ScanCode.io

nexB Inc.

May 08, 2024

GETTING STARTED

Welcome to the very start of your ScanCode.io journey! In this documentation you'll find information on:

- An overview of ScanCode.io and ScanPipe
- Installation instructions
- Tutorials to get you started
- Reference documentation about the ScanPipe concepts, Pipelines, Pipes and more
- How to make technical contributions to the project and the community

CHAPTER

ONE

SCANCODE.IO OVERVIEW

ScanCode.io is a server to script and automate the process of **Software Composition Analysis (SCA)** to identify any open source components and their license compliance data in an application's codebase. ScanCode.io can be used for various use cases, such as Docker container and VM composition analyses, among other applications.

1.1 Why ScanCode.io?

Modern software is built from many open source packages assembled with new code. Knowing which free and open source code package is in use matters because:

- You're required to **know the license of third-party code** before using it, and
- You want to avoid using buggy, outdated or vulnerable components.

It's usually convenient to include and reuse new code downloaded from the internet; however, it's often surprisingly hard to get a proper inventory of all third-party code origins and licenses used in a software project. There are some great tools available to scan your code and help uncover these details. For example, when you reuse only a few FOSS components in a single project, running one of these tools, such as the **ScanCode-toolkit**, manually along with a spreadsheet might be enough to manage your software composition analysis.

However, when you scale up, running automated and reproducible analysis pipelines that are adapted to a software project's unique context and technology platform can be difficult. This will require deploying and running multiple specialized tools and merge their results with a consistent workflow. Moreover, when reusing thousands of open source packages is becoming commonplace, code scans pipelines need to be scripted as code is running on servers backed by a shared database, not on a laptop.

For instance, when you analyze Docker container images, there could be hundreds to thousands of system packages, such as Debian, RPM, Alpine, and application packages, including npm, PyPI, Rubygems, Maven, installed in an image side-by-side with your own code. Taking care of all this can be an extremely hard task, and that's when **ScanCode.io** comes into play to help organizing these complex code analysis as scripted pipelines and store their results in a database for automated code analysis.

1.2 What is ScanPipe?

ScanPipe is a developer-friendly framework and application that helps software analysts and engineers build and manage real-life software composition analysis projects as scripted pipelines.

ScanPipe provides a unified framework to the infrastructure that is required to execute and organize these software composition analysis projects.

1.3 Should I use ScanPipe?

If you are working on a software composition analysis project, or you are planning to start a new one, consider the following questions:

1. **Automation**: Is the project part of a larger compliance program (as opposed to a one-off) and that you require automation?
2. **Complexity**: Does the project use many third-party components or technologies?
3. **Reproducibility**: Is it important that the results are reproducible, traceable, and auditable?

If you answered **"yes"** to any of the above, keep reading - ScanPipe can help you. If the answer is **"no"** to all of the above, which is a valid scenario, e.g., when you are doing small-scale analysis, ScanPipe may provide only limited benefit for you.

The first set of available pipelines helps automate the analysis of Docker container images and virtual machine (VM) disk images that often harbor comprehensive software stacks from an operating system with its kernel through system and application packages to original and custom applications.

1.4 Dependencies and Internal Tools

ScanCode.io is essentially a Django-based application wrapper around the ScanCode Toolkit scanning engine.

The **Django framework** is leveraged for many aspects of ScanCode.io:

- *User Interface*
- *REST API*
- *Command Line Interface*
- *Data Models*

Note: Multiple applications from the Django eco-system are also included, see the setup.cfg file for an exhaustive list of dependencies.

The second essential part of ScanCode.io is the **ScanCode Toolkit**, which is used for archives extraction and as the scanning engine.

The nexB container-inspector library is also a key component of ScanCode.io as this tool is used to analyse Docker images, containers, root filesystems, and virtual machine images.

Note: As a common practice, ScanCode.io releases usually follow ScanCode Toolkit releases to ensure the latest improvements of the scanning engines are included in the latest release of ScanCode.io.

CHAPTER

TWO

INSTALLATION

Welcome to **ScanCode.io** installation guide! This guide describes how to install ScanCode.io on various platforms. Please read and follow the instructions carefully to ensure your installation is functional and smooth.

The **preferred ScanCode.io installation** is to *Run with Docker* as this is the simplest to setup and get started. Running ScanCode.io with Docker **guarantee the availability of all features** with the **minimum configuration** required. This installation **works across all Operating Systems**.

Alternatively, you can install ScanCode.io locally as a development server with some limitations and caveats. Refer to the *Local development installation* section.

2.1 Run with Docker

2.1.1 Get Docker

The first step is to download and **install Docker on your platform**. Refer to Docker documentation and choose the best installation path for your system: Get Docker.

2.1.2 Build the Image

ScanCode.io is distributed with `Dockerfile` and `docker-compose.yml` files required for the creation of the Docker image.

Warning: On **Windows**, ensure that git `autocrlf` configuration is set to `false` before cloning the repository:

```
git config --global core.autocrlf false
```

Clone the git ScanCode.io repo, create an **environment file**, and **build the Docker image**:

```
git clone https://github.com/nexB/scancode.io.git && cd scancode.io
make envfile
docker compose build
```

Warning: As the `docker-compose` v1 command is officially deprecated by Docker, you will only find references to the `docker compose` v2 command in this documentation.

2.1.3 Run the App

Run your image as a container:

```
docker compose up
```

At this point, the ScanCode.io app should be running at port 80 on your Docker host. Go to http://localhost/ on a web browser to **access the web UI**.

An overview of the web application usage is available at *User Interface*.

Note: Congratulations, you are now ready to use ScanCode.io, and you can move onto the **Tutorials** section starting with the *Analyze Docker Image (Web UI)* tutorial.

Tip: ScanCode.io will take advantage of all the CPUs made available by your Docker configuration for faster processing.

Make sure to allow enough memory to support each CPU processes.

A good rule of thumb is to allow **2 GB of memory per CPU**. For example, if Docker is configured for 8 CPUs, a minimum of 16 GB of memory is required.

Tip: By default, ScanCode.io starts only 1 worker, which means only 1 pipeline will be executed at a time. If you wish to start more workers, use the following command, replacing the number 2 with the desired number of workers:

```
docker compose up --scale worker=2
```

Warning: To access a dockerized ScanCode.io app from a remote location, the `ALLOWED_HOSTS` and `CSRF_TRUSTED_ORIGINS` settings need to be provided in your `.env` file, for example:

```
ALLOWED_HOSTS=.your-domain.com
CSRF_TRUSTED_ORIGINS=https://*.your-domain.com
```

Refer to ALLOWED_HOSTS settings and CSRF_TRUSTED_ORIGINS settings for more details.

Tip: If you run ScanCode.io on desktop or laptop, it may come handy to pause/unpause or suspend your local ScanCode.io system. For this, use these commands:

```
docker compose pause   # to pause/suspend
docker compose unpause  # to unpause/resume
```

2.1.4 Upgrade the App

Update your local ScanCode.io repo, and **build the Docker image**:

```
cd scancode.io
git pull
docker compose build
```

Warning: The Docker image has been updated to run as a non-root user. If you encounter "permissions" issues while running the ScanCode.io Docker images following the `docker compose build`, you will need to update the the permissions of the `/var/scancodeio/` directory of the Docker volumes using:

```
docker compose run -u 0:0 web chown -R app:app /var/scancodeio/
```

See also https://github.com/nexB/scancode.io/issues/399

Note: You need to rebuild the image whenever ScanCode.io's source code has been modified or updated.

2.1.5 Execute a Command

Note: Refer to the *Command Line Interface* section for the full list of commands.

A `scanpipe` command can be executed through the `docker compose` command line interface with:

```
docker compose exec -it web scanpipe COMMAND
```

2.1.6 Use alternative HTTP ports

By default, the application is accessible on port 80 for HTTP and 443 for HTTPS requests. This assumes that these ports are not already occupied by another application. You can customize both of these ports by adjusting the following variables in the `.env` file, located in the root of the application directory, next to the `docker-compose.yml` file:

```
NGINX_PUBLISHED_HTTP_PORT=8080
NGINX_PUBLISHED_HTTPS_PORT=8443
```

2.2 Offline installation with Docker

ScanCode.io can be installed and operated on a server that is not connected to the internet, such as an "airgapped" or isolated server.

To achieve this, Docker images are initially built on a machine with internet access and subsequently transferred to the "offline" server for isolated installation.

Note: `docker` and `docker compose` are required on both the local machine and the server.

2.2.1 Build the offline installation package

Build and save the offline installation package with docker images, configuration and scripts on your local machine:

```
make offline-package
```

A tarball `scancodeio-offline-package-VERSION.tar` will be created in the *dist/* directory.

2.2.2 Install on an offline server

Copy the tarball to the server then extract it replacing `VERSION` with the actual version value:

```
tar -xf scancodeio-offline-package-VERSION.tar
```

Change to the extracted `build/` directory:

```
cd build
```

Load the docker Images:

```
docker load --input scancodeio-images.tar.gz
```

2.2.3 Run on an offline server

Run the App by starting the ScanCode.io services:

```
docker compose --file docker-compose-offline.yml up
```

And visit the web UI at: http://localhost/project/

Note: The nginx service (webserver) requires the port 80 to be available on the host. In case the port 80 is already in used, you will encounter the following error:

```
ERROR: for build_nginx_1 Cannot start service nginx: driver failed programming ...
```

You can attempt to stop potential running services blocking the port 80 with the following commands on the host before starting ScanCode.io services:

```
sudo systemctl stop nginx
sudo systemctl stop apache2
```

2.3 Local development installation

2.3.1 Supported Platforms

ScanCode.io has been tested and is supported on the following operating systems:

1. **Debian-based** Linux distributions
2. **macOS** 10.14 and up

Warning: On **Windows** ScanCode.io can **only** be *Run with Docker*.

2.3.2 Pre-installation Checklist

Before you install ScanCode.io, make sure you have the following prerequisites:

- **Python: versions 3.10 to 3.12** found at https://www.python.org/downloads/
- **Git**: most recent release available at https://git-scm.com/
- **PostgreSQL**: release 11 or later found at https://www.postgresql.org/ or https://postgresapp.com/ on macOS

2.3.3 System Dependencies

In addition to the above pre-installation checklist, there might be some OS-specific system packages that need to be installed before installing ScanCode.io.

On **Linux**, several **system packages are required** by the ScanCode toolkit. Make sure those are installed before attempting the ScanCode.io installation:

```
sudo apt-get install \
    build-essential python3-dev libssl-dev libpq-dev \
    bzip2 xz-utils zlib1g libxml2-dev libxslt1-dev libpopt0 \
    libgpgme11 libdevmapper1.02.1 libguestfs-tools
```

See also ScanCode-toolkit Prerequisites for more details.

For the *Collect Codebase Symbols (addon)* pipeline, Universal Ctags is needed.

- On **Linux** install it using:

  ```
  sudo apt-get install universal-ctags
  ```

- On **MacOS** install Universal Ctags using Homebrew:

  ```
  brew install universal-ctags
  ```

For the *Collect Source Strings (addon)* pipeline, gettext is needed.

- On **Linux** install it using:

  ```
  sudo apt-get install gettext
  ```

- On **MacOS** install gettext using Homebrew:

  ```
  brew install gettext
  ```

2.3.4 Clone and Configure

- Clone the ScanCode.io GitHub repository:

```
git clone https://github.com/nexB/scancode.io.git && cd scancode.io
```

- Inside the *scancode.io/* directory, install the required dependencies:

```
make dev
```

Note: You can specify the Python version during the `make dev` step using the following command:

```
make dev PYTHON_EXE=python3.11
```

When `PYTHON_EXE` is not specified, by default, the `python3` executable is used.

Tip: When running M1 based MacOS, you can also install SCIO in x86 mode using rosetta:

```
softwareupdate --install-rosetta
arch -x86_64 /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/
↪Homebrew/install/master/install.sh)"
arch -x86_64 /usr/local/Homebrew/bin/brew install python@3.12
make dev PYTHON_EXE=/usr/local/bin/python3.12
(. bin/activate; pip install psycopg[binary])
```

- Create an environment file:

```
make envfile
```

2.3.5 Database

PostgreSQL is the preferred database backend and should always be used on production servers.

- Create the PostgreSQL user, database, and table with:

```
make postgresdb
```

Warning: The `make postgres` command is assuming that your PostgreSQL database template is using the `en_US.UTF-8` collation. If you encounter database creation errors while running this command, it is generally related to an incompatible database template.

You can either update your template to fit the ScanCode.io default, or provide custom values collation using the `POSTGRES_INITDB_ARGS` variable such as:

```
make postgresdb POSTGRES_INITDB_ARGS=\
    --encoding=UTF-8 --lc-collate=en_US.UTF-8 --lc-ctype=en_US.UTF-8
```

Note: You can also use a **SQLite** database for local development as a single user with:

```
make sqlitedb
```

Warning: Choosing SQLite over PostgreSQL has some caveats. Check this link for more details.

2.3.6 Tests

You can validate your ScanCode.io installation by running the tests suite:

```
make test
```

2.3.7 Web Application

A web application is available to create and manage your projects from a browser; you can start the local webserver and access the app with:

```
make run
```

Then open your web browser and visit: http://localhost:8001/ to access the web application.

Warning: `make run` is provided as a simplified way to run the application with one **major caveat**: pipeline runs will be **executed synchronously** on HTTP requests and will leave your browser connection or API calls opened during the pipeline execution. See also the *SCANCODEIO_ASYNC* setting.

Warning: This setup is **not suitable for deployments** and **only supported for local development**. It is highly recommended to use the *Run with Docker* setup to ensure the availability of all the features and the benefits from asynchronous workers for pipeline executions.

An overview of the web application usage is available at *User Interface*.

2.3.8 Upgrading

If you already have the ScanCode.io repo cloned, you can upgrade to the latest version with:

```
cd scancode.io
git pull
make dev
make migrate
```

2.4 Helm Chart [Beta]

Warning: The Helm Chart support for ScanCode.io is a community contribution effort. It is only tested on a few configurations and still under development. We welcome improvement suggestions and issue reports at ScanCode.io GitHub repo.

2.4.1 Requirements

Helm must be installed to use the charts. Please refer to Helm's documentation to get started.

Requires:

- Kubernetes cluster running with appropriate permissions (depending on your cluster)
- `kubectl` set up to connect to the cluster
- `helm`

Tested on:

- minikube v1.25.1:

```
$ minikube version
minikube version: v1.25.1
commit: 3e64b11ed75e56e4898ea85f96b2e4af0301f43d
```

- helm v3.8.1:

```
$ helm version
version.BuildInfo{Version:"v3.8.1",
GitCommit:"5cb9af4b1b271d11d7a97a71df3ac337dd94ad37",
GitTreeState:"clean", GoVersion:"go1.17.5"}
```

2.4.2 Installation

Once Helm is properly set up, add the `scancode-kube` repo as follows:

```
# clone github repository
git clone git@github.com:xerrni/scancode-kube.git

# create kubernetes namespace
kubectl create namespace scancode

# configure values.yaml file
vi values.yaml

# install helm dependencies
helm dependency update

# check if dependencies are installed
helm dependency list

# sample output
```

(continues on next page)

(continued from previous page)

```
# NAME            VERSION REPOSITORY                          STATUS
# nginx           9.x.x   https://charts.bitnami.com/bitnami      ok
# postgresql      11.x.x  https://charts.bitnami.com/bitnami      ok
# redis           16.x.x  https://charts.bitnami.com/bitnami      ok

# install scancode helm charts
helm install scancode ./ --namespace scancode

# wait until all pods are in Running state
# afterwards cancel this command as it will run forever
kubectl get pods -n scancode --watch

# sample output
# NAME                                      READY   STATUS    RESTARTS   AGE
# scancode-nginx-f4d79f44d-4vhlv            1/1     Running   0          5m28s
# scancode-postgresql-0                     1/1     Running   0          5m28s
# scancode-redis-master-0                   1/1     Running   0          5m28s
# scancode-scancodeio-web-5786df657c-khrgb  1/1     Running   0          5m28s
# scancode-scancodeio-worker-0              1/1     Running   1          5m28s

# expose nginx frontend
minikube service --url=true -n scancode scancode-nginx
```

2.5 Gitpod

Warning: The Gitpod support for ScanCode.io is a community contribution effort. We welcome improvement suggestions and issue reports at ScanCode.io GitHub repo.

2.5.1 Installation

- Create a new Workspace and open it in VSCode Browser or your preferred IDE. Provide the ScanCode.io GitHub repo URL: https://github.com/nexB/scancode.io
- Open the "TERMINAL" window and create the `.env` file with:

```
make envfile
```

- Open the generated `.env` file and add the following settings:

```
ALLOWED_HOSTS=.gitpod.io
CSRF_TRUSTED_ORIGINS=https://*.gitpod.io
```

2.5.2 Run the App

- Build and run the app container:

  ```
  docker compose build
  docker compose up
  ```

At this stage, the ScanCode.io app is up and running. To access the app, open the “PORTS” window and open the address for port 80 in your browser.

CHAPTER

THREE

USER INTERFACE

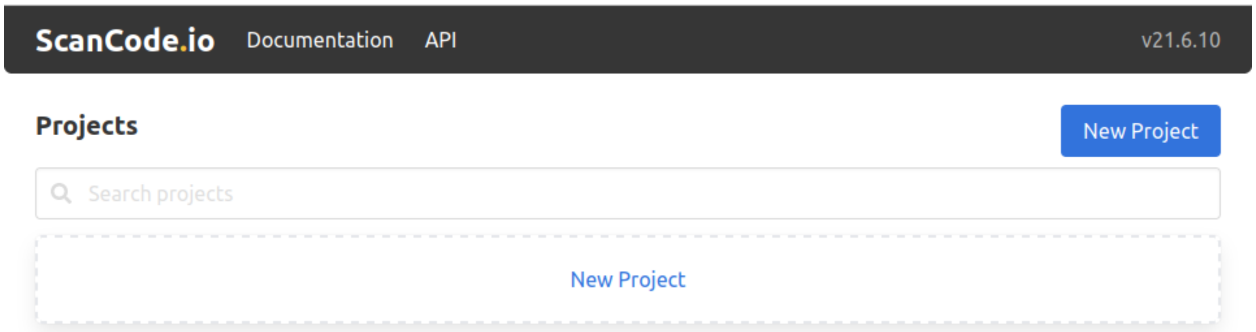
As mentioned in the installation guide, ScanCode.io offers a web application to create and manage your projects from a browser. You'll get access to this visual interface when you successfully install ScanCode.io locally.

To access the web application, open your web browser and visit http://localhost/ or http://localhost:8001/ if you run on a local development setup.

Note: All the capabilities offered by the ScanCode.io Web Interface are also available as *Command Line Interface* and in the *REST API*.

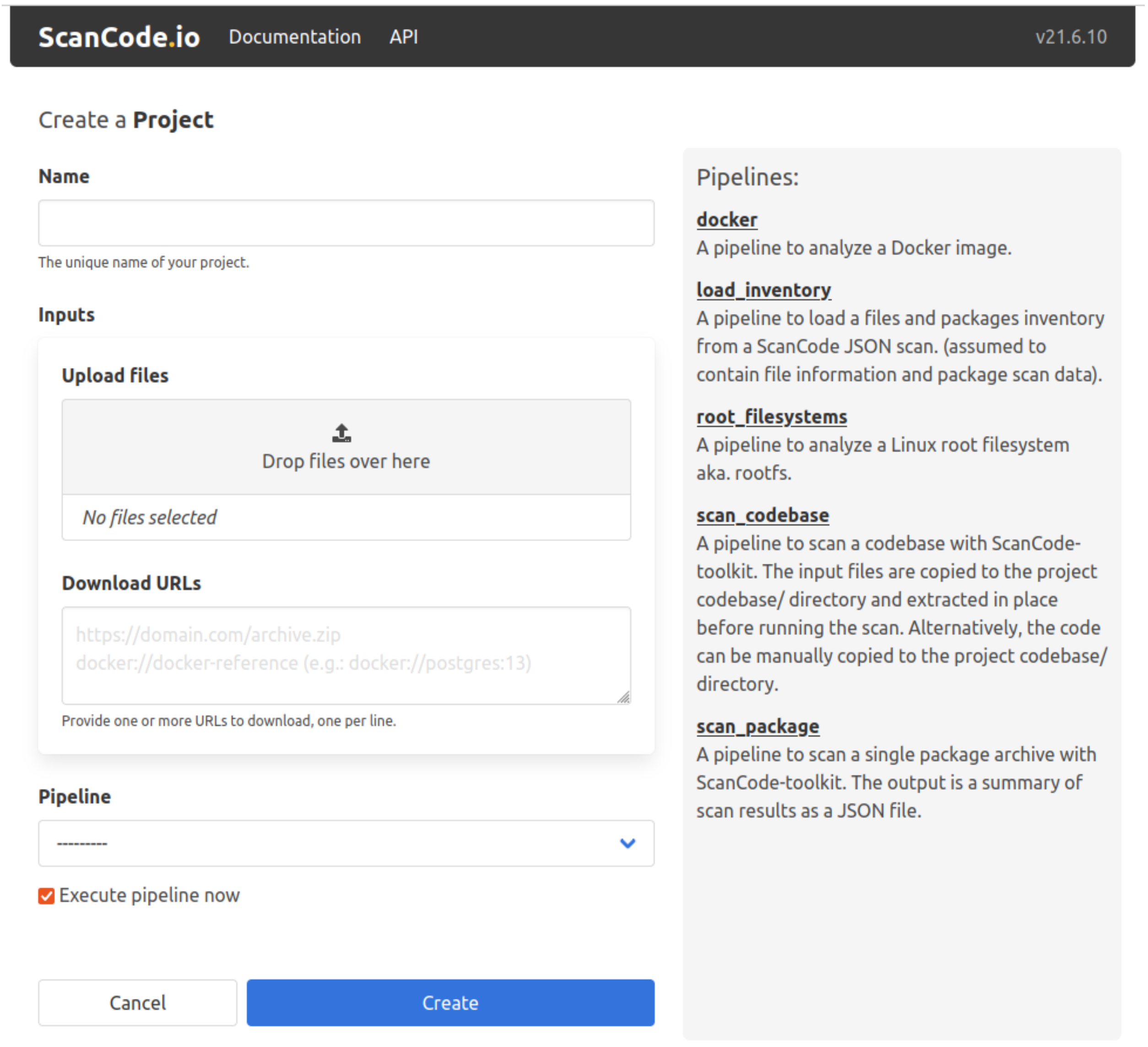
3.1 Home Screen

When you open the web application for the first time, the home screen will appear. From this screen, you'll be able to create a new project, search your existing projects, view or download scan results, access documentation, and more.



3.2 Creating a New Project

Creating a project is the first step that needs to be performed before you can start using ScanCode.io. There are two **"New Project"** buttons on the home screen, as shown in the previous screenshot. To create a new project, click on either button, and you will be directed to the **"Create a Project"** page.



Create a **Project**

Name

The unique name of your project.

Inputs

Upload files

Drop files over here

No files selected

Download URLs

https://domain.com/archive.zip
docker://docker-reference (e.g.: docker://postgres:13)

Provide one or more URLs to download, one per line.

Pipeline

☑ Execute pipeline now

Pipelines:

docker
A pipeline to analyze a Docker image.

load_inventory
A pipeline to load a files and packages inventory from a ScanCode JSON scan. (assumed to contain file information and package scan data).

root_filesystems
A pipeline to analyze a Linux root filesystem aka. rootfs.

scan_codebase
A pipeline to scan a codebase with ScanCode-toolkit. The input files are copied to the project codebase/ directory and extracted in place before running the scan. Alternatively, the code can be manually copied to the project codebase/ directory.

scan_package
A pipeline to scan a single package archive with ScanCode-toolkit. The output is a summary of scan results as a JSON file.

As shown above, creating a project involves filling in the following input fields:

3.2.1 Name

To create a project, you must provide a unique name for the new project.

Warning: A project name can't be changed or edited once the project has been created.

3.2.2 Inputs

You can upload files available on your machine or add links to desired input files, such as package archives or Docker images, as the input to your project. If you are providing more than one URL as the project input, make sure to add one URL per line.

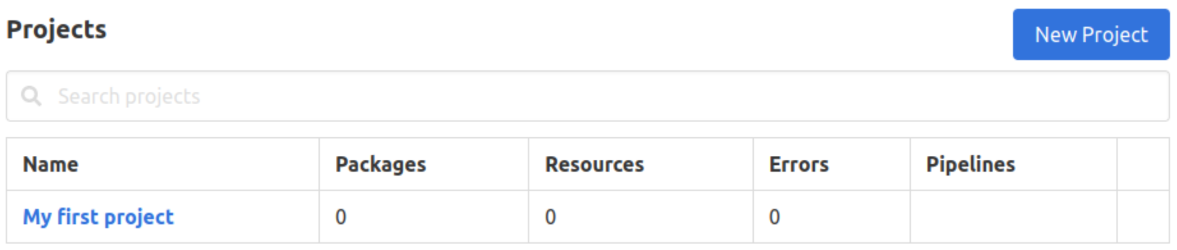
Tip: Docker images can be provided as inputs to be fetched using the `docker://docker-reference` syntax in the **"Download URLs"** field. For example: `docker://postgres:13`

3.2.3 Pipeline

Use the drop-down list to select one of the available pipelines depending on your use-case. When you add a pipeline, you can check the **"Execute pipeline now"** checkbox, which will add and execute the selected pipeline in one operation.

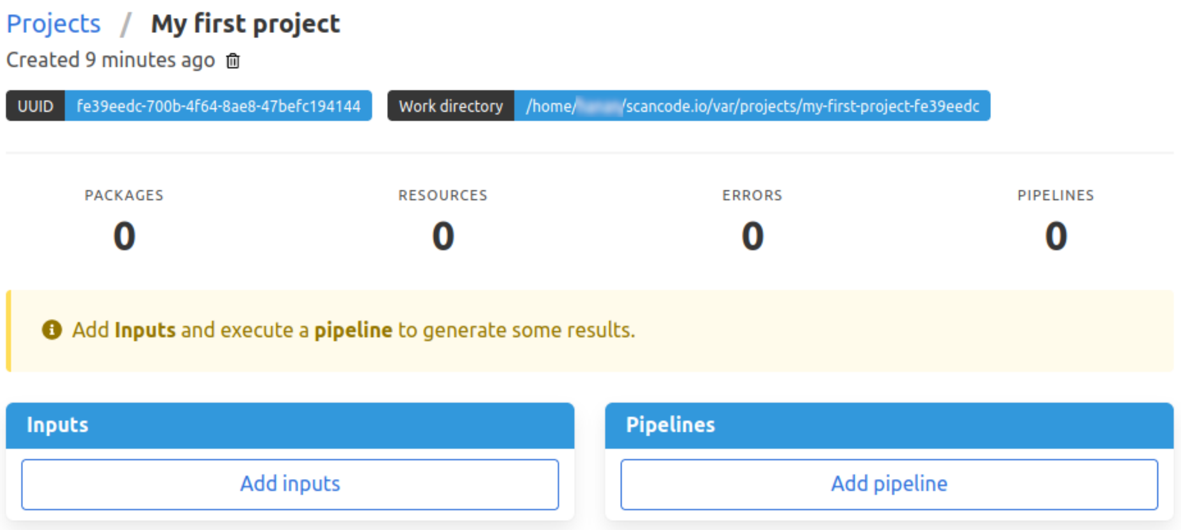
Note: If you're not sure of which pipeline to select, refer to the pipelines details on the right pane of the **"Create a Project"** page, as shown above.

You can still create a new project while leaving the **Inputs** and **Pipeline** fields blank; however, it's mandatory to provide a project **Name**!



Name	Packages	Resources	Errors	Pipelines	
My first project	0	0	0		

Once successfully created, you can later add any needed inputs and pipelines to your project by clicking the **"Add inputs"** and **"Add pipeline"** buttons.



Created 9 minutes ago

UUID fe39eedc-700b-4f64-8ae8-47befc194144

Work directory /home/[redacted]/scancode.io/var/projects/my-first-project-fe39eedc

PACKAGES 0 RESOURCES 0 ERRORS 0 PIPELINES 0

Add **Inputs** and execute a **pipeline** to generate some results.

Warning: You will not be able to add any extra inputs once a pipeline has been run on the project. However, you still can add and run extra pipelines as needed!

Within each project, you can view your project details, review the results of the pipeline execution, or download the output files.

Note: Please refer to the *Output Files* page for more details about your scan results.

3.3 Project Settings

The project settings form provides a convenient interface for editing essential project details and adding relevant notes. With this form, you have the ability to modify the project name, as well as include any additional notes you deem necessary.

In addition to managing project information, the form also offers configuration options that are related to the extraction process. You can specify a list of items to be ignored during pipeline execution, ensuring that only relevant content is considered. Furthermore, you have the option to customize the attribution template according to your specific requirements.

Tip: To generate the project configuration file `scancode-config.yml`, navigate to the Project Settings UI and click on the **Config file** link in the left section under **Download**.

3.3.1 Archive a Project

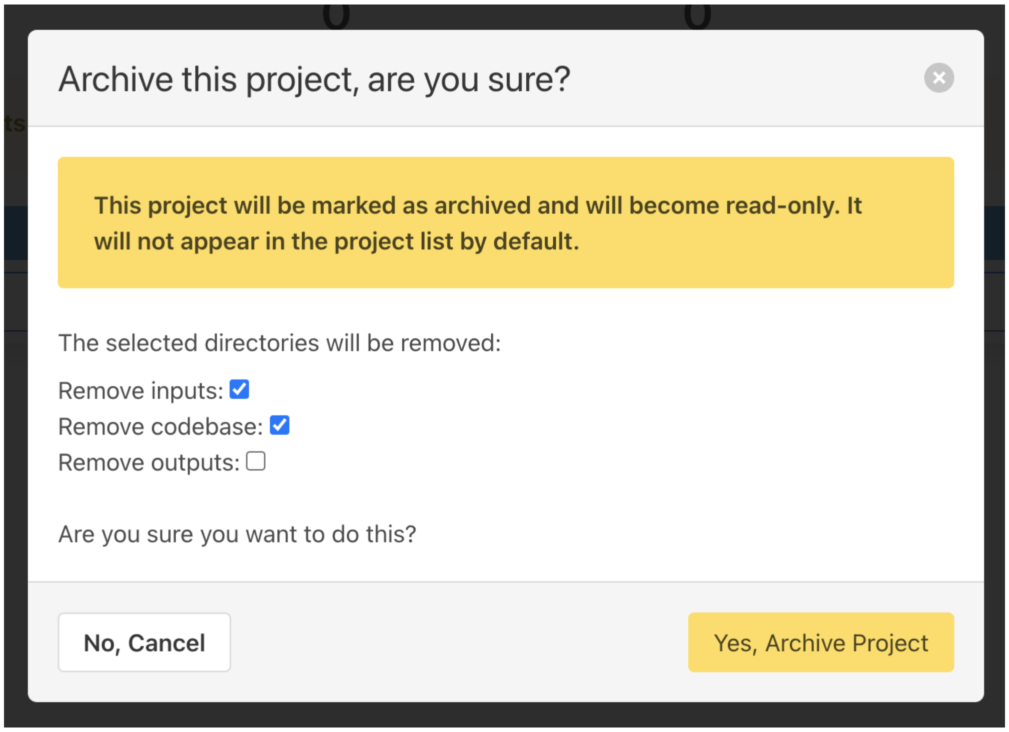
After a project is complete, you may want to archive it to prevent any further modification to that project.

Archiving projects also makes navigating existing projects easier as the archived projects are hidden by default from the project list.

Selected *Project workspace* directories can be removed during the archive operation.

Tip: The project results are stored in the database and available to generate outputs at any time.

Note: A project cannot be archived if one of its related run is queued or already running.


Archive this project, are you sure?

This project will be marked as archived and will become read-only. It will not appear in the project list by default.

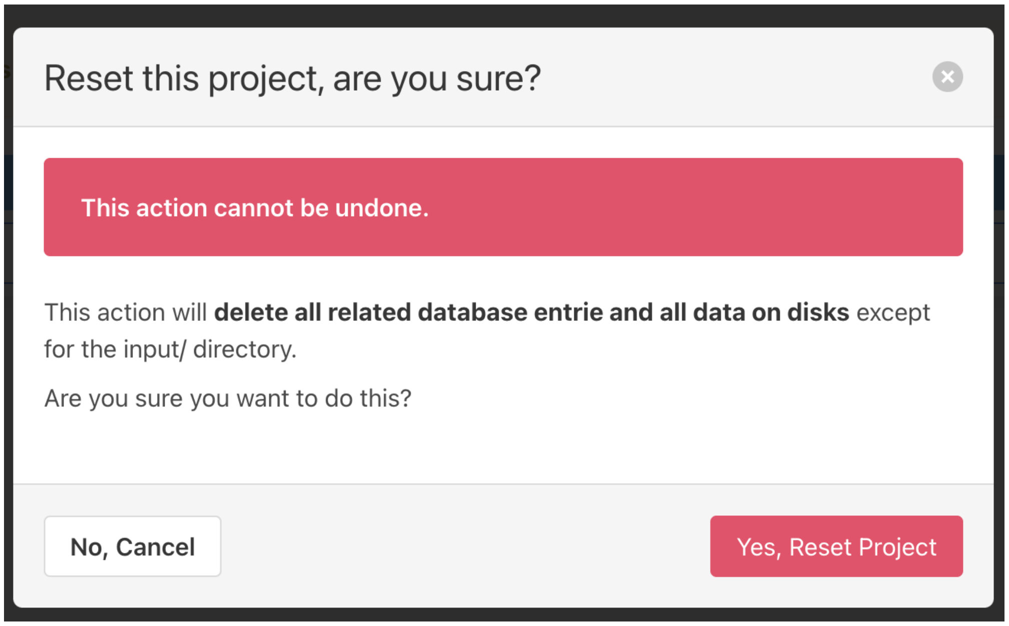
The selected directories will be removed:

Remove inputs: ✓
Remove codebase: ✓
Remove outputs: ☐

Are you sure you want to do this?

3.3.2 Reset a Project

The reset allows to **wipe all database entries and all data on disks** related to a project while keeping the *input/* files. It can be used to re-run pipelines on a clean slate of the project without having to re-upload input files.


Reset this project, are you sure?

This action cannot be undone.

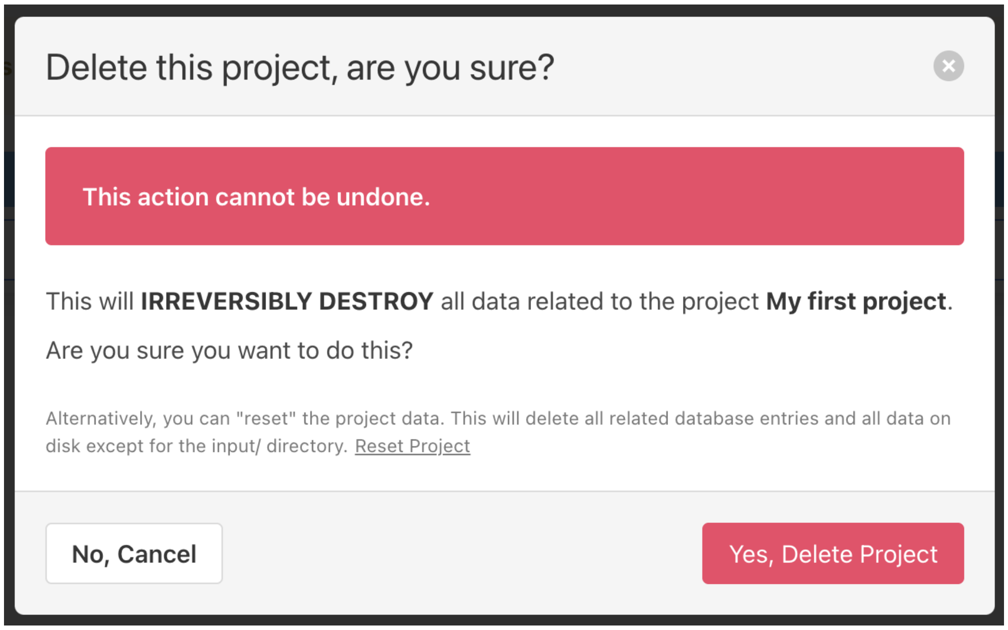
This action will **delete all related database entrie and all data on disks** except for the input/ directory.

Are you sure you want to do this?

3.3.3 Delete a Project

If any of your projects is no longer needed, you can delete it from the project's details page. Deleting old projects also makes navigating existing projects easier. Simply to delete any project, click on the trash icon under the project's name.

Warning: Projects get permanently deleted and cannot be restored.



Delete this project, are you sure?

This action cannot be undone.

This will **IRREVERSIBLY DESTROY** all data related to the project **My first project**.

Are you sure you want to do this?

Alternatively, you can "reset" the project data. This will delete all related database entries and all data on disk except for the input/ directory. Reset Project

3.4 Search Syntax

When searching on objects list views, you can use a powerful search syntax to refine your queries and find exactly what you're looking for. This guide will walk you through the search syntax and provide examples to help you get started.

3.4.1 Basic Search

- **Single Term**: Enter a single word to search for exact matches.

  Example: `file.txt`

- **Quoted Phrases**: Use double quotes to search for exact phrases.

  Example: `"name version"`

3.4.2 Advanced Search

- **Field Searches**: Specify fields to narrow down your search. Use `field_name:` followed by your search term.

  Example: `name:file.txt`
- **Negation**: Use a hyphen (-) before a field name to exclude results.

  Example: `-name:file.txt`
- **Lookup Types**: Use lookup types to perform specific searches.
  - =: Exact match
  - `^`: Starts with
  - `$`: Ends with
  - ~: Contains
  - >: Greater than
  - <: Less than

Example: `path^:dir1`

3.4.3 Combining Queries

Multiple queries are combined with the `AND` operator:

Example: `name:file.txt status:scanned`

3.4.4 Examples

Here are some examples of advanced searches:

Search for Resources:

- Find resources by name and license_expression: `name:LICENSE detected_license_expression:mit`
- Find resources by path ending: `path$:directory_without_slash`

CHAPTER

FOUR

FAQS

You can't find what you're looking for? Below you'll find answers to a few of our frequently asked questions.

4.1 How can I run a scan?

You simply start by creating a *new project* and run the appropriate pipeline.

ScanCode.io offers several *Built-in Pipelines* depending on your input, see above.

4.2 Which pipeline should I use?

Selecting the right pipeline for your needs depends primarily on the type of input data you have available. Here are some general guidelines based on different input scenarios:

- If you have a **Docker image** as input, use the *analyze_docker_image* pipeline.
- For a full **codebase compressed as an archive**, optionally also with it's **pre-resolved dependenices**, and want to detect all the packages present linked with their respective files, use the *scan_codebase* pipeline.
- If you have a **single package archive**, and you want to get information on licenses, copyrights and package metadata for it, opt for the *scan_single_package* pipeline.
- When dealing with a **Linux root filesystem** (rootfs), the *analyze_root_filesystem_or_vm_image* pipeline is the appropriate choice.
- For processing the results of a **ScanCode-toolkit scan** or **ScanCode.io scan**, use the *load_inventory* pipeline.
- When you want to import **SPDX/CycloneDX SBOMs or ABOUT files** into a project, use the *load_sbom* pipeline.
- When you have **lockfiles or other package manifests** in a codebase and you want to resolve packages from their package requirements, use the *resolve_dependencies* pipeline.
- When you have application **package archives/codebases** and optionally also their **pre-resolved dependenices** and you want to **inspect packages** present in the package manifests and dependency, use the *inspect_packages* pipeline.
- For scenarios involving both a **development and deployment codebase**, consider using the *map_deploy_to_develop* pipeline.
- For getting the DWARF debug symbol compilation unit paths when available from an elf binary. use the *inspect_elf_binaries* pipeline.

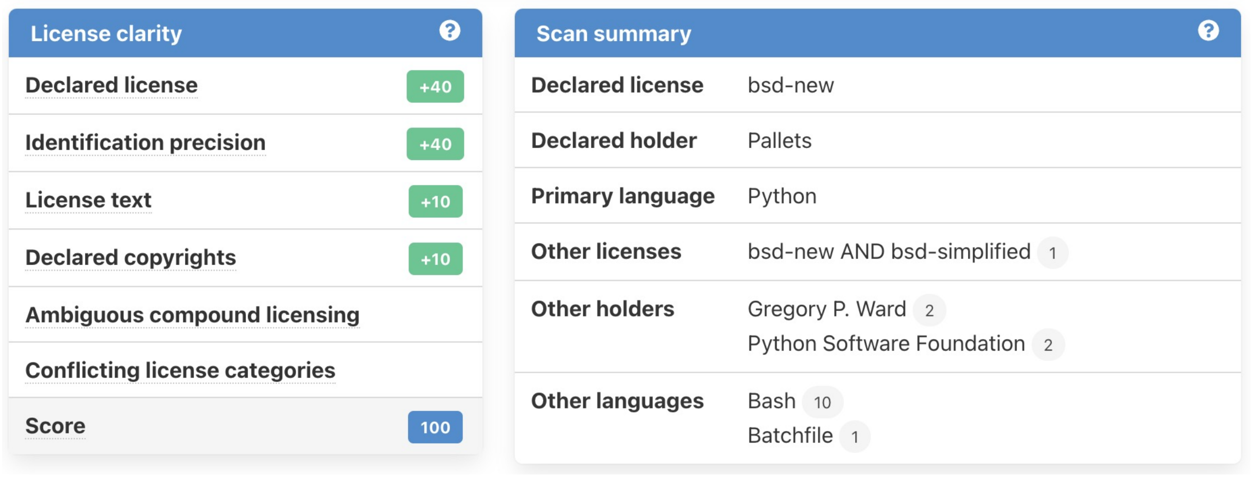
These pipelines will automatically execute the necessary steps to scan and create the packages, dependencies, and resources for your project based on the input data provided.

After executing one of the pipelines mentioned above, you have the option to **augment your project's data** by executing additional pipelines, often referred to as **addon** pipelines. These additional pipelines offer further enhancements and modifications to your existing data, allowing for more comprehensive analysis and insights.

- If you wish to **find vulnerabilities** for packages and dependencies, you can use the *find_vulnerabilities* pipeline. Note that setting up *VulnerableCode* is required for this pipeline to function properly.
- To **populate PurlDB with the packages discovered in your project**, use the *populate_purldb* pipeline. Before executing this pipeline, make sure to set up *PurlDB*.
- To **match your project codebase resources to MatchCode.io for Package matches**, utilize the *match_to_matchcode* pipeline. It's essential to set up *MatchCode.io* before executing this pipeline.

4.3 What is the difference between scan_codebase and scan_single_package pipelines?

The key differences are that the *scan_single_package* pipeline treats the input as if it were a single package, such as a package archive, and computes a **License clarity** and a **Scan summary** to aggregate the package scan data:



License clarity	
Declared license	+40
Identification precision	+40
License text	+10
Declared copyrights	+10
Ambiguous compound licensing	
Conflicting license categories	
Score	100

Scan summary	
Declared license	bsd-new
Declared holder	Pallets
Primary language	Python
Other licenses	bsd-new AND bsd-simplified 1
Other holders	Gregory P. Ward 2 Python Software Foundation 2
Other languages	Bash 10 Batchfile 1

In contrast, the *scan_codebase* pipeline is more of a general purpose pipeline and make no such single package assumption. It does not not compute such summary.

You can also have a look at the different steps for each pipeline from the *Built-in Pipelines* documentation.

4.4 Can I run multiple pipelines in parallel?

Yes, you can run multiple pipelines in parallel by starting your Docker containers with the desired number of workers using the following command:

```
docker compose up --scale worker=2
```

Note: You can also add extra workers by running the command while the ScanCode.io services are already running. For example, to add 2 extra workers to the 2 currently running ones, use the following command:

```
sudo docker compose up --scale worker=4
```

4.5 Can I pause/resume a running pipeline?

You can stop/terminate a running pipeline but it will not be possible to resume it. Although, as a workaround if you run ScanCode.io on desktop or laptop, you can pause/unpause the running Docker containers with:

```
docker compose pause  # to pause/suspend
docker compose unpause  # to unpause/resume
```

4.6 What tool does ScanCode.io use to analyze docker images?

The following tools and libraries are used during the docker images analysis pipeline:

- container-inspector and debian-inspector for handling containers and distros.
- fetchcode-container to download containers and images.
- scancode-toolkit for application package scans and system package scans.
- extractcode for universal and reliable archive extraction.
- Specific handling of windows containers is done in scancode-toolkit to process the windows registry.
- Secondary libraries and plugins from scancode-plugins.

The pipeline documentation is available at *Analyse Docker Image* and its source code at docker.py. It is hopefully designed to be simple and readable code.

4.7 Am I able to run ScanCode.io on Windows?

Yes, you can use the *Run with Docker* installation. However, please be sure to carefully read the warnings, as running on Windows may have certain limitations or challenges.

4.8 Is it possible to compare scan results?

At the moment, you can only download full reports in JSON and XLSX formats. Please refer to our *Output Files* section for more details on the output formats.

4.9 How can I trigger a pipeline scan from a CI/CD, such as Jenkins, TeamCity or Azure Devops?

You can refer to the *Automation* to automate your projects management.

Also, A new GitHub action is available at scancode-action repository to run ScanCode.io pipelines from your GitHub Workflows.

4.10 How to tag input files?

Certain pipelines, including the *Map Deploy To Develop*, require input files to be tagged. This section outlines various methods to tag input files based on your project management context.

4.10.1 Using download URLs as inputs

You can provide tags using the "#<fragment>" section of URLs. This tagging method is universally applicable in the User Interface, REST API, and Command Line Interface.

Example:

```
https://url.com/sources.zip#from
https://url.com/binaries.zip#to
```

4.10.2 Uploading local files

There are multiple ways to tag input files when uploading local files:

- **User Interface:** Utilize the "Edit flag" link in the "Inputs" panel of the Project details view.
- **REST API:** Use the "upload_file_tag" field in addition to the "upload_file" field.
- **Command Line Interface:** Tag uploaded files using the "filename:tag" syntax. Example: `--input-file path/filename:tag`.

4.11 How to fetch files from private sources and protected by credentials?

Several *Fetch Authentication* settings are available to define the credentials required to access your private files, depending on the authentication type:

- *Basic authentication*
- *Digest authentication*
- *HTTP request headers*
- *.netrc file*
- *Docker private repository*

Example for GitHub private repository files:

```
SCANCODEIO_FETCH_HEADERS="github.com=Authorization=token <YOUR_TOKEN>"
```

Example for Docker private repository:

```
SCANCODEIO_SKOPEO_CREDENTIALS="registry.com=user:password"
```

CHAPTER

FIVE

CONTRIBUTING TO SCANCODE.IO

Thank you so much for being so interested in contributing to ScanCode.io. We are always on the lookout for enthusiastic contributors like you who can make our project better, and we're willing to lend a helping hand if you have any questions or need guidance along the way. That being said, here are a few resources to help you get started.

Note: By contributing to the ScanCode.io project, you agree to the Developer Certificate of Origin.

5.1 Do Your Homework

Before adding a contribution or create a new issue, take a look at the project's README, read through our documentation, and browse existing issues, to develop some understanding of the project and confirm whether a given issue/feature has previously been discussed.

5.2 Ways to Contribute

Contributing to the codebase is not the only way to add value to ScanCode.io or join our community. Below are some examples to get involved:

5.2.1 First Timers

You are here to help, but you're a new contributor! No worries, we always welcome newcomer contributors. We maintain some good first issues and encourage new contributors to work on those issues for a smooth start.

Warning: "Can I work on this issue?"

You do not need our permission to work on an open issue. A good start is to present your understanding of the problem/bug and how you would fix it. Providing some code using a pull request will come handy, but being able to explain a solution is always a good start.

Make sure to read through this page and follow the recommendations.

Warning: "Is this issue is open?"

Unless closed, yes it is open.

5.2.2 Report Issues

- Report a new bug; just remember to be as specific as possible.
- Create a new issue to request a feature, submit a feedback, or ask a question.
- Look into existing bugs, try to reproduce the issue on your side, and discuss solutions in the comments.

Note: Make sure to check existing issues, and *FAQs* to confirm whether a given issue or a question has previously been discussed.

5.2.3 Code Contributions

Code is contributed to the codebase using **pull requests**. A pull request should always be attached to an existing issue. When there is no existing issues, start by creating one to discuss potential solutions and implementation details before writing any code.

We use several conventions to ensure code quality regarding format, testing, and attribution.

Make sure to follow those conventions before submitting your code:

1. **Code validation**

   We use PEP8 conventions. A command is available to automatically format your code:

   ```
   make valid
   ```

2. **Unit tests**

   We write tests, a lot of tests, thousands of tests. When fixing bugs or adding new features, you should add tests too. You can run the test suite with:

   ```
   make test
   ```

3. **Commit messages and Developer Certificate of Origin**

   Follow the instructions at Writing good Commit Messages and check some examples.

   You must include a "Signed-off-by" to your commit messages:

   ```
   Signed-off-by: Philippe Ombredanne <pombredanne@nexb.com>
   ```

4. **Your code is now ready to be pushed as a PR**

Note: Pull requests that are not passing the automated integration tests are unlikely to be reviewed. Focus on making all the "Checks" to pass before asking for a code review.

5.2.4 Documentation Improvements

Documentation is a critical aspect of any project that is usually neglected or overlooked. We value any suggestions to improve ScanCode.io documentation.

Tip: Our documentation is treated like code. Make sure to check our writing guidelines to help guide new users.

5.2.5 Other Ways

You want to contribute to other aspects of the ScanCode.io project, and you can't find what you're looking for! You can always discuss new topics, ask questions, and interact with us and other community members on Gitter.

5.3 Helpful Resources

- Review our comprehensive guide for more details on how to add quality contributions to our codebase and documentation
- Check this free resource on how to contribute to an open source project on github
- Follow this wiki page on how to write good commit messages
- Pro Git book
- How to write a good bug report

CHAPTER

SIX

CHANGELOG

6.1 v34.5.0 (unreleased)

- Display the current path location in the “Codebase” panel as a navigation breadcrumbs. https://github.com/nexB/scancode.io/issues/1158
- Fix a rendering issue in the dependency details view when for_package or datafile_resource fields do not have a value. https://github.com/nexB/scancode.io/issues/1177
- Add a new *CollectPygmentsSymbolsAndStrings* pipeline (addon) for collecting source symbol, string and comments using Pygments. https://github.com/nexB/scancode.io/pull/1179
- Workaround an issue with the cyclonedx-python-lib that does not allow to load SBOMs that contains properties with no values. https://github.com/nexB/scancode.io/issues/1185
- Add a new *CollectTreeSitterSymbolsAndStrings* pipeline (addon) for collecting source symbol and string using tree-sitter. https://github.com/nexB/scancode.io/pull/1181
- Fix *inspect_packages* pipeline to properly link discovered packages and dependencies to codebase resources of package manifests where they were found. Also correctly assign the datasource_ids attribute for packages and dependencies. https://github.com/nexB/scancode.io/pull/1180
- Add “Product name” and “Product version” as new project settings. https://github.com/nexB/scancode.io/issues/1197
- Add “Product name” and “Product version” as new project settings. https://github.com/nexB/scancode.io/issues/1197
- Raise the minimum RAM required per CPU code in the docs. A good rule of thumb is to allow **2 GB of memory per CPU**. For example, if Docker is configured for 8 CPUs, a minimum of 16 GB of memory is required. https://github.com/nexB/scancode.io/issues/1191
- Add value validation for the search complex query syntax. https://github.com/nexB/scancode.io/issues/1183

6.2 v34.4.0 (2024-04-22)

- Upgrade Gunicorn to v22.0.0 security release.
- Display the list of fields available for the advanced search syntax in the modal UI. https://github.com/nexB/scancode.io/issues/1164
- Add support for CycloneDX 1.6 outputs and inputs. Also, the CycloneDX outputs can be downloaded as 1.6, 1.5, and 1.4 spec versions. https://github.com/nexB/scancode.io/pull/1165
- Update matchcode-toolkit to v4.1.0

- Add a new function *scanpipe.pipes.matchcode.fingerprint_codebase_resources()*, which computes approximate file matching fingerprints for text files using the new *get_file_fingerprint_hashes* function from matchcode-toolkit.
- Rename the *purldb-scan-queue-worker* management command to *purldb-scan-worker*.
- Add *docker-compose.purldb-scan-worker.yml* to run ScanCode.io as a PurlDB scan worker service.

6.3 v34.3.0 (2024-04-10)

- Associate resolved packages with their source codebase resource. https://github.com/nexB/scancode.io/issues/1140
- Add a new *CollectSourceStrings* pipeline (addon) for collecting source string using xgettext. https://github.com/nexB/scancode.io/pull/1160

6.4 v34.2.0 (2024-03-28)

- Add support for Python 3.12 and upgrade to Python 3.12 in the Dockerfile. https://github.com/nexB/scancode.io/pull/1138
- Add support for CycloneDX XML inputs. https://github.com/nexB/scancode.io/issues/1136
- Upgrade the SPDX schema to v2.3.1 https://github.com/nexB/scancode.io/issues/1130

6.5 v34.1.0 (2024-03-27)

- Add support for importing CycloneDX SBOM 1.2, 1.3, 1.4 and 1.5 spec formats. https://github.com/nexB/scancode.io/issues/1045
- The pipeline help modal is now available from all project views: form, list, details. The docstring are converted from markdown to html for proper rendering. https://github.com/nexB/scancode.io/pull/1105
- Add a new *CollectSymbols* pipeline (addon) for collecting codebase symbols using Universal Ctags. https://github.com/nexB/scancode.io/pull/1116
- Capture errors during the *inspect_elf_binaries* pipeline execution. Errors on resource inspection are stored as project error message instead of global pipeline failure. The problematic resource path is stored in the message details and displayed in the message list UI as a link to the resource details view. https://github.com/nexB/scancode.io/issues/1121 https://github.com/nexB/scancode.io/issues/1122
- Use the *package_only* option in scancode *get_package_data* API in *inspect_packages* pipeline, to skip license and copyright detection in extracted license and copyright statements found in package metadata. https://github.com/nexB/scancode-toolkit/pull/3689
- Rename the `match_to_purldb` pipeline to `match_to_matchcode`, and add MatchCode.io API settings to ScanCode.io settings.
- In the DiscoveredPackage model, rename the "datasource_id" attribute to "datasource_ids" and add a new attribute "datafile_paths". This is aligned with the scancode-toolkit Package model, and package detection information is now stored correctly. Also update the UI for discovered packages to show the corresponding package datafiles and their datasource IDs. A data migration is included to facilitate the migration of existing data. https://github.com/nexB/scancode.io/issues/1099

- Add PurlDB tab, displayed when the PURLDB_URL settings is configured. When loading the package details view, a request is made on the PurlDB to fetch and and display any available data. https://github.com/nexB/scancode.io/issues/1125
- Create a new management command *purldb-scan-queue-worker*, that runs scancode.io as a Package scan queue worker for PurlDB. *purldb-scan-queue-worker* gets the next available Package to be scanned and the list of pipeline names to be run on the Package from PurlDB, creates a Project, fetches the Package, runs the specified pipelines, and returns the results to PurlDB. https://github.com/nexB/scancode.io/pull/1078 https://github.com/nexB/purldb/issues/236
- Update matchcode-toolkit to v4.0.0

6.6 v34.0.0 (2024-03-04)

- Add ability to "group" pipeline steps to control their inclusion in a pipeline run. The groups can be selected in the UI, or provided using the "pipeline_name:group1,group2" syntax in CLI and REST API. https://github.com/nexB/scancode.io/issues/1045
- **Refine pipeline choices in the "Add pipeline" modal based on the project context.**
  - When there is at least one existing pipeline in the project, the modal now includes all addon pipelines along with the existing pipeline for selection.
  - In cases where no pipelines are assigned to the project, the modal displays all base (non-addon) pipelines for user selection.

  https://github.com/nexB/scancode.io/issues/1071
- Rename pipeline for consistency and precision: * scan_codebase_packages: inspect_packages

  Restructure the inspect_manifest pipeline into: * load_sbom: for loading SPDX/CycloneDX SBOMs and ABOUT files * resolve_dependencies: for resolving package dependencies * inspect_packages: gets package data from package manifests/lockfiles

  A data migration is included to facilitate the migration of existing data. Only the new names are available in the web UI but the REST API and CLI are backward compatible with the old names. https://github.com/nexB/scancode.io/issues/1034 https://github.com/nexB/scancode.io/discussions/1035
- Remove "packageFileName" entry from SPDX output. https://github.com/nexB/scancode.io/issues/1076
- Add an add-on pipeline for collecting DWARF debug symbol compilation unit paths when available from elfs. https://github.com/nexB/purldb/issues/260
- Extract all archives recursively in the *scan_single_package* pipeline. https://github.com/nexB/scancode.io/issues/1081
- Add URL scheme validation with explicit error messages for input URLs. https://github.com/nexB/scancode.io/issues/1047
- All supported *output_format* can now be downloaded using the results_download API action providing a value for the new *output_format* parameter. https://github.com/nexB/scancode.io/issues/1091
- Add settings related to fetching private files. Those settings allow to define credentials for various authentication types. https://github.com/nexB/scancode.io/issues/620 https://github.com/nexB/scancode.io/issues/203
- Update matchcode-toolkit to v3.0.0

6.7 v33.1.0 (2024-02-02)

- **Rename multiple pipelines for consistency and precision:**
  - docker: analyze_docker_image
  - root_filesystems: analyze_root_filesystem_or_vm_image
  - docker_windows: analyze_windows_docker_image
  - inspect_manifest: inspect_packages
  - deploy_to_develop: map_deploy_to_develop
  - scan_package: scan_single_package

  A data migration is included to facilitate the migration of existing data. Only the new names are available in the web UI but the REST API and CLI are backward compatible with the old names. https://github.com/nexB/scancode.io/issues/1044
- Generate CycloneDX SBOM in 1.5 spec format, migrated from 1.4 previously. The Package vulnerabilities are now included in the CycloneDX SBOM when available. https://github.com/nexB/scancode.io/issues/807
- Improve the inspect_manifest pipeline to accept archives as inputs. https://github.com/nexB/scancode.io/issues/1034
- Add support for "tagging" download URL inputs using the "#<fragment>" section of URLs. This feature is particularly useful in the map_develop_to_deploy pipeline when download URLs are utilized as inputs. Tags such as "from" and "to" can be specified by adding "#from" or "#to" fragments at the end of the download URLs. Using the CLI, the uploaded files can be tagged using the "filename:tag" syntax while using the *–input-file* arguments. In the UI, tags can be edited from the Project details view "Inputs" panel. On the REST API, a new *upload_file_tag* field is available to use along the *upload_file*. https://github.com/nexB/scancode.io/issues/708

6.8 v33.0.0 (2024-01-16)

- Upgrade Django to version 5.0 and drop support for Python 3.8 and 3.9 https://github.com/nexB/scancode.io/issues/1020
- Fetching "Download URL" inputs is now delegated to an initial pipeline step that is always run as the start of a pipeline. This allows to run pipelines on workers running from a remote location, external to the main ScanCode.io app server. https://github.com/nexB/scancode.io/issues/410
- Migrate the Project.input_sources field into a InputSource model. https://github.com/nexB/scancode.io/issues/410
- Refactor run_scancode to not fail on scan errors happening at the resource level, such as a timeout. Project error message are created instead. https://github.com/nexB/scancode.io/issues/1018
- Add support for the SCANCODEIO_SCAN_FILE_TIMEOUT setting in the scan_package pipeline. https://github.com/nexB/scancode.io/issues/1018
- Add support for non-archive single file in the scan_package pipeline. https://github.com/nexB/scancode.io/issues/1009
- Do not include "add-on" pipelines in the "New project" form choices. https://github.com/nexB/scancode.io/issues/1041
- Display a "Run pipelines" button in the "Pipelines" panel. Remove the ability to run a single pipeline in favor of running all "not started" project pipeline. https://github.com/nexB/scancode.io/issues/997

- In "map_deploy_to_develop" pipeline, add support for path patterns in About file attributes documenting resource paths. https://github.com/nexB/scancode.io/issues/1004
- Fix an issue where the pipeline details cannot be fetched when using URLs that include credentials such as "user:pass@domain". https://github.com/nexB/scancode.io/issues/998
- Add a new pipeline, `match_to_purldb`, that check CodebaseResources of a Project against PurlDB for Package matches.

6.9 v32.7.0 (2023-10-25)

- Display the `Run.scancodeio_version` in the Pipeline run modal. When possible this value is displayed as a link to the diff view between the current ScanCode.io version and the version used when the Pipeline was run. https://github.com/nexB/scancode.io/issues/956
- Improve presentation of the "Resources detected license expressions" project section. https://github.com/nexB/scancode.io/issues/937
- Add ability to sort by Package URL in package list https://github.com/nexB/scancode.io/issues/938
- Fix an issue where the empty project settings were overriding the settings loaded from a config file. https://github.com/nexB/scancode.io/issues/961
- Control the execution order of Pipelines within a Project. Pipelines are not allowed to start anymore unless all the previous ones within a Project have completed. https://github.com/nexB/scancode.io/issues/901
- Add support for webhook subscriptions in project clone. https://github.com/nexB/scancode.io/pull/910
- Add resources license expression summary panel in the project details view. This panel displays the list of licenses detected in the project and include links to the resources list. https://github.com/nexB/scancode.io/pull/355
- Add the `tag` field on the DiscoveredPackage model. This new field is used to store the layer id where the package was found in the Docker context. https://github.com/nexB/scancode.io/issues/919
- Add to apply actions, such as archive, delete, and reset to a selection of project from the main list. https://github.com/nexB/scancode.io/issues/488
- Add new "Outputs" panel in the Project details view. Output files are listed and can be downloaded from the panel. https://github.com/nexB/scancode.io/issues/678
- Add a step in the `deploy_to_develop` pipelines to create "local-files" packages with from-side resource files that have one or more relations with to-side resources that are not part of a package. This allows to include those files in the SBOMs and attribution outputs. https://github.com/nexB/scancode.io/issues/914
- Enable sorting the packages list by resources count. https://github.com/nexB/scancode.io/issues/978

6.10 v32.6.0 (2023-08-29)

- Improve the performance of the codebase relations list view to support large number of entries. https://github.com/nexB/scancode.io/issues/858
- Improve DiscoveredPackageListView query performances refining the prefetch_related. https://github.com/nexB/scancode.io/issues/856
- Fix the `map_java_to_class` d2d pipe to skip if no `.java` file is found. https://github.com/nexB/scancode.io/issues/853
- Enhance Package search to handle full `pkg:` purls and segment of purls. https://github.com/nexB/scancode.io/issues/859

- Add a new step in the `deploy_to_develop` pipeline where we tag archives as processed, if all the resources in their extracted directory is mapped/processed. https://github.com/nexB/scancode.io/issues/827
- Add the ability to clone a project. https://github.com/nexB/scancode.io/issues/874
- Improve perceived display performance of projects charts and stats on home page. The charts are displayed when the number of resources or packages are less than 5000 records. Else, a button to load the charts is displayed. https://github.com/nexB/scancode.io/issues/844
- Add advanced search query system to all list views. Refer to the documentation for details about the search syntax. https://github.com/nexB/scancode.io/issues/871
- Migrate the ProjectError model to a global ProjectMessage. 3 level of severity available: INFO, WARNING, and ERROR. https://github.com/nexB/scancode.io/issues/338
- Add label/tag system that can be used to group and filters projects. https://github.com/nexB/scancode.io/issues/769

6.11 v32.5.2 (2023-08-14)

Security release: This release addresses the security issue detailed below. We encourage all users of ScanCode.io to upgrade as soon as possible.

- GHSA-6xcx-gx7r-rccj: Reflected Cross-Site Scripting (XSS) in license endpoint The `license_details_view` function was subject to cross-site scripting (XSS) attack due to inadequate validation and sanitization of the key parameter. The license views were migrated class-based views are the inputs are now properly sanitized. Credit to @0xmpij for reporting the vulnerability. https://github.com/nexB/scancode.io/security/advisories/GHSA-6xcx-gx7r-rccj https://github.com/nexB/scancode.io/issues/847
- Add bandit analyzer and Django "check –deploy" to the check/validation stack. This helps to ensure that we do not introduce know code vulnerabilities and deployment issues to the codebase. https://github.com/nexB/scancode.io/issues/850
- Migrate the run_command function into a safer usage of the subprocess module. Also fix various warnings returned by the bandit analyzer. https://github.com/nexB/scancode.io/issues/850
- Replace the `scancode.run_scancode` function by a new `run_scan` that interact with scancode-toolkit scanners without using subprocess. This new function is used in the `scan_package` pipeline. The `SCANCODE_TOOLKIT_CLI_OPTIONS` settings was renamed `SCANCODE_TOOLKIT_RUN_SCAN_ARGS`. Refer to the documentation for the next "dict" syntax. https://github.com/nexB/scancode.io/issues/798

6.12 v32.5.1 (2023-08-07)

Security release: This release addresses the security issue detailed below. We encourage all users of ScanCode.io to upgrade as soon as possible.

- GHSA-2ggp-cmvm-f62f: Command injection in docker image fetch process The `fetch_docker_image` function was subject to potential injection attack. The user inputs are now sanitized before calling the subprocess function. Credit to @0xmpij for reporting the vulnerability. https://github.com/nexB/scancode.io/security/advisories/GHSA-2ggp-cmvm-f62f

—

- Add support for multiple input URLs, and adding multiple pipelines in the project creation REST API. https://github.com/nexB/scancode.io/issues/828

- Update the `fetch_vulnerabilities` pipe to make the API requests by batch of purls. https://github.com/nexB/scancode.io/issues/835
- Add vulnerability support for discovered dependencies. The dependency data is loaded using the `find_vulnerabilities` pipeline backed by a VulnerableCode database. https://github.com/nexB/scancode.io/issues/835
- Fix root filesystem scanning for installed packages and archived Linux distributions. Allows the scan to discover system packages from *rpmdb.sqlite* and other sources. https://github.com/nexB/scancode.io/pull/840

6.13 v32.5.0 (2023-08-02)

WARNING: After upgrading the ScanCode.io codebase to this version, and following the `docker compose build`, the permissions of the `/var/scancodeio/` directory of the Docker volumes require to be updated for the new `app` user, using: `docker compose run -u 0:0 web chown -R app:app /var/scancodeio/`

- Run Docker as non-root user using virtualenv. WARNING: The permissions of the `/var/scancodeio/` directory in the Docker volumes require to be updated for the new `app` user. https://github.com/nexB/scancode.io/issues/399
- Add column sort and filters in dependency list view. https://github.com/nexB/scancode.io/issues/823
- Add a new `ScanCodebasePackage` pipeline to scan a codebase for packages only. https://github.com/nexB/scancode.io/issues/815
- Add new `outputs` REST API action that list projects output files including an URL to download the file. https://github.com/nexB/scancode.io/issues/678
- Add support for multiple to/from input files in the `deploy_to_develop` pipeline. https://github.com/nexB/scancode.io/issues/813
- Add the ability to delete and download project inputs. Note that the inputs cannot be modified (added or deleted) once a pipeline run as started on the project. https://github.com/nexB/scancode.io/issues/813
- Fix root_filesystem data structure stored on the Project `extra_data` field. This was causing a conflict with the expected docker images data structure when generating an XLSX output. https://github.com/nexB/scancode.io/issues/824
- Fix the SPDX output to include missing detailed license texts for LicenseRef. Add `licensedb_url` and `scancode_url` to the SPDX `ExtractedLicensingInfo` `seeAlsos`. Include the `Package.notice_text` as the SPDX `attribution_texts`. https://github.com/nexB/scancode.io/issues/841

6.14 v32.4.0 (2023-07-13)

- Add support for license policies and complaince alert for Discovered Packages. https://github.com/nexB/scancode.io/issues/151
- Refine the details views and tabs: - Add a “Relations” tab in the Resource details view - Disable empty tabs by default - Display the count of items in the tab label - Improve query performances for details views https://github.com/nexB/scancode.io/issues/799
- Upgrade vulnerablecode integration: - Add `affected_by_vulnerabilities` field on `DiscoveredPackage` model. - Add UI for showing package vulnerabilities in details view. - Add packages filtering by `is_vulnerable`. - Include vulnerability data in the JSON results. https://github.com/nexB/scancode.io/issues/600

- Add multiple new filtering option to list views table headers. Refactored the way to define filters using the table_columns view attribute. https://github.com/nexB/scancode.io/issues/216 https://github.com/nexB/scancode.io/issues/580 https://github.com/nexB/scancode.io/issues/506
- Update the CycloneDX BOM download file extension from `.bom.json` to `.cdx.json`. https://github.com/nexB/scancode.io/issues/785
- SPDX download BOM do not include codebase resource files by default anymore. https://github.com/nexB/scancode.io/issues/785
- Add archive_location to the LAYERS worksheet of XLSX output. https://github.com/nexB/scancode.io/issues/773
- Add "New Project" button to Project details view. https://github.com/nexB/scancode.io/issues/763
- Display image type files in the codebase resource details view in a new "Image" tab.
- Add `slug` field on the Project model. That field is used in URLs instead of the `uuid`. https://github.com/nexB/scancode.io/issues/745
- Fix the ordering of the Codebase panel in the Project details view. https://github.com/nexB/scancode.io/issues/795
- Do not rely on the internal `id` PK for package and dependency details URLs. Package details URL is now based on `uuid` and the dependency details URL is based on `dependency_uid`. https://github.com/nexB/scancode.io/issues/331
- Add a "License score" project setting that can be used to limit the returned license matches with a score above the provided one. This is leveraging the ScanCode-toolkit `--license-score` option, see: https://scancode-toolkit.readthedocs.io/en/stable/cli-reference/basic-options.html#license-score-option https://github.com/nexB/scancode.io/issues/335

6.15 v32.3.0 (2023-06-12)

- Upgrade ScanCode-toolkit to latest v32.0.x Warning: This upgrade requires schema and data migrations (both included). It is recommended to reset and re-run the pipelines to benefit from the latest ScanCode detection improvements. Refer to https://github.com/nexB/scancode-toolkit/blob/develop/CHANGELOG.rst#v3200-next-roadmap for the full list of changes. https://github.com/nexB/scancode.io/issues/569
- Add a new `deploy_to_develop` pipeline specialized in creating relations between the development source code and binaries or deployed code. This pipeline is expecting 2 archive files with "from-" and "to-" filename prefixes as inputs: 1. "from-[FILENAME]" archive containing the development source code 2. "to-[FILENAME]" archive containing the deployment compiled code https://github.com/nexB/scancode.io/issues/659
- Add ability to configure a Project through a new "Settings" form in the UI or by providing a ".scancode-config.yml" configuration file as one of the Project inputs. The "Settings" form allows to rename a Project, add and edit the notes, as well as providing a list of patterns to be ignored during pipeline runs, the choice of extracting archives recursively, and the ability to provide a custom template for attribution. https://github.com/nexB/scancode.io/issues/685 https://github.com/nexB/scancode.io/issues/764
- Add `notes` field on the Project model. Notes can be updated from the Project settings form. Also, notes can be provided while creating a project through the CLI using the a new `--notes` option. https://github.com/nexB/scancode.io/issues/709
- Add a mapper function to relate .ABOUT files during the d2d pipeline. https://github.com/nexB/scancode.io/issues/740
- Enhance the file viewer UI of the resource details view. A new search for the file content was added. Also, it is now possible to expand the file viewer in full screen mode. https://github.com/nexB/scancode.io/issues/724

- Refine the breadcrumb UI for details view. https://github.com/nexB/scancode.io/issues/717
- Move the "Resources status" panel from the run modal to the project details view. https://github.com/nexB/scancode.io/issues/370
- Improve the speed of Project `reset` and `delete` using the _raw_delete model API. https://github.com/nexB/scancode.io/issues/729
- Specify `update_fields` during each `save()` related to Run tasks, to force a SQL UPDATE in order to avoid any data loss when the model fields are updated during the task execution. https://github.com/nexB/scancode.io/issues/726
- Add support for XLSX input in the `load_inventory` pipeline. https://github.com/nexB/scancode.io/issues/735
- Add support for unknown licenses in attribution output. https://github.com/nexB/scancode.io/issues/749
- Add `License` objects to each of the package for attribution generation. https://github.com/nexB/scancode.io/issues/775
- The "Codebase" panel can now be used to browse the Project's codebase/ directory and open related resources details view. https://github.com/nexB/scancode.io/issues/744

6.16 v32.2.0 (2023-04-25)

- Enhance the `update_or_create_package` pipe and add the ability to assign multiple codebase resources at once. https://github.com/nexB/scancode.io/issues/681
- Add new command line option to create-project and add-input management commands to copy the content of a local source directory to the project codebase work directory. https://github.com/nexB/scancode.io/pull/672
- Include the ScanCode-toolkit version in the output headers. https://github.com/nexB/scancode.io/pull/670
- Enhance the `output` management command to support providing multiple formats at once. https://github.com/nexB/scancode.io/issues/646
- Improve the resolution of CycloneDX BOM and SPDX document when the file extension is simply `.json`. https://github.com/nexB/scancode.io/pull/688
- Add support for manifest types using ScanCode-toolkit handlers. https://github.com/nexB/scancode.io/issues/658
- Enhance the Resource details view to use the tabset system and display all available data including the content viewer. https://github.com/nexB/scancode.io/issues/215
- Add a "layers" data sheet in the xlsx output for docker pipeline run. https://github.com/nexB/scancode.io/issues/578
- Move the `cyclonedx` and `spdx` root modules into the `pipes` module. https://github.com/nexB/scancode.io/issues/657
- Remove the admin app and views. https://github.com/nexB/scancode.io/issues/645
- Enhance the `resolve_about_packages` pipe to handle filename and checksum values.
- Split the pipes unit tests into their own related submodule.
- Upgrade ScanCode Toolkit to v31.2.6 https://github.com/nexB/scancode.io/issues/693

6.17 v32.1.0 (2023-03-23)

- Add support for ScanCode.io results in the "load_inventory" pipeline. https://github.com/nexB/scancode.io/issues/609
- Add support for CycloneDX 1.4 to the "inspect-manifest" pipeline to import SBOM into a Project. https://github.com/nexB/scancode.io/issues/583
- Add fields in CycloneDX BOM output using the component properties. See registered properties at https://github.com/nexB/aboutcode-cyclonedx-taxonomy https://github.com/nexB/scancode.io/issues/637
- Upgrade to Python 3.11 in the Dockerfile. https://github.com/nexB/scancode.io/pull/611
- Refine the "Command Line Interface" documentation about the `scanpipe` command usages in the Docker context. Add the /app workdir in the "PYTHONPATH" env of the Docker file to make the `scanpipe` entry point available while running `docker compose` commands. https://github.com/nexB/scancode.io/issues/616
- Add new tutorial about the "find vulnerabilities" pipeline and the vulnerablecode integration in the documentation. https://github.com/nexB/scancode.io/issues/600
- Rewrite the CLI tutorials for a Docker-based installation. https://github.com/nexB/scancode.io/issues/440
- Use CodebaseResource `path` instead of `id` as slug_field in URL navigation. https://github.com/nexB/scancode.io/issues/242
- Remove dead code related to the project_tree view https://github.com/nexB/scancode.io/issues/623
- Update `scanpipe.pipes.ProjectCodebase` and related code to work properly with current Project/CodebaseResource path scheme. https://github.com/nexB/scancode.io/pull/624
- Add `SCANCODEIO_PAGINATE_BY` setting to customize the number of items displayed per page for each object type. https://github.com/nexB/scancode.io/issues/563
- Add setting for per-file timeout. The maximum time allowed for a file to be analyzed when scanning a codebase is configurable with SCANCODEIO_SCAN_FILE_TIMEOUT while the maximum time allowed for a pipeline to complete can be defined using SCANCODEIO_TASK_TIMEOUT. https://github.com/nexB/scancode.io/issues/593

6.18 v32.0.1 (2023-02-20)

- Upgrade ScanCode-toolkit and related dependencies to solve installation issues. https://github.com/nexB/scancode.io/pull/586
- Add support for Python 3.11 https://github.com/nexB/scancode.io/pull/611
- Populate `documentDescribes` field with Package and Dependency SPDX IDs in SPDX BOM output. https://github.com/nexB/scancode.io/issues/564

6.19 v32.0.0 (2022-11-29)

- Add a new "find vulnerabilities" pipeline to lookup vulnerabilities in the VulnerableCode database for all project discovered packages. Vulnerability data is stored in the extra_data field of each package. More details about VulnerableCode at https://github.com/nexB/vulnerablecode/ https://github.com/nexB/scancode.io/issues/101
- Add a new "inspect manifest" pipeline to resolve packages from manifest, lockfile, and SBOM. The resolved packages are created as discovered packages. Support PyPI "requirements.txt" files, SPDX document as JSON ".spdx.json", and AboutCode ".ABOUT" files. https://github.com/nexB/scancode.io/issues/284
- Generate SBOM (Software Bill of Materials) compliant with the SPDX 2.3 specification as a new downloadable output. https://github.com/nexB/scancode.io/issues/389
- Generate CycloneDX SBOM (Software Bill of Materials) as a new downloadable output. https://github.com/nexB/scancode.io/issues/389
- Display Webhook status in the Run modal. The WebhookSubscription model was refined to capture delivery data. https://github.com/nexB/scancode.io/issues/389
- Display the current active step of a running pipeline in the "Pipeline" section of the project details view, inside the run status tag. https://github.com/nexB/scancode.io/issues/300
- Add proper pagination for API actions: resources, packages, dependencies, and errors.
- Refine the fields ordering in API Serializers based on the toolkit order. https://github.com/nexB/scancode.io/issues/546
- Keep the current filters state when submitting a search in list views. https://github.com/nexB/scancode.io/issues/541
- Improve the performances of the project details view to load faster by deferring the the charts rendering. This is especially noticeable on projects with a large amount of codebase resources and discovered packages. https://github.com/nexB/scancode.io/issues/193
- Add support for filtering by "Other" values when filtering from the charts in the Project details view. https://github.com/nexB/scancode.io/issues/526
- `CodebaseResource.for_packages` now returns a list of `DiscoveredPackage.package_uid` or `DiscoveredPackage.package_url` if `DiscoveredPackage.package_uid` is not present. This is done to reflect the how scancode-toolkit's JSON output returns `package_uid``s in the ``for_packages` field for Resources.
- Add the model DiscoveredDependency. This represents Package dependencies discovered in a Project. The `scan_codebase` and `scan_packages` pipelines have been updated to create DiscoveredDepdendency objects. The Project API has been updated with new fields:
  - `dependency_count` - The number of DiscoveredDependencies associated with the project.
  - `discovered_dependencies_summary` - A mapping that contains following fields:
    - `total` - The number of DiscoveredDependencies associated with the project.
    - `is_runtime` - The number of runtime dependencies.
    - `is_optional` - The number of optional dependencies.
    - `is_resolved` - The number of resolved dependencies.

  These values are also available on the Project view. https://github.com/nexB/scancode.io/issues/447
- The `dependencies` field has been removed from the DiscoveredPackage model.
- Create directory CodebaseResources in the rootfs pipeline. https://github.com/nexB/scancode.io/issues/515

- Add ProjectErrors when the DiscoveredPackage could not be fetched using the provided *package_uid* during the *assemble_package* step instead of failing the whole pipeline. https://github.com/nexB/scancode.io/issues/525
- Escape paths before using them in regular expressions in `CodebaseResource.walk()`. https://github.com/nexB/scancode.io/issues/525
- Disable multiprocessing and threading by default on macOS ("spawn" start method). https://github.com/nexB/scancode.io/issues/522

6.20 v31.0.0 (2022-08-25)

- WARNING: Drop support for Python 3.6 and 3.7. Add support for Python 3.10. Upgrade Django to version 4.1 series.
- Upgrade ScanCode-toolkit to version 31.0.x. See https://github.com/nexB/scancode-toolkit/blob/develop/CHANGELOG.rst for an overview of the changes in the v31 compared to v30.
- Implement run status auto-refresh using the htmx JavaScript library. The statuses of queued and running pipeline are now automatically refreshed in the project list and project details views every 10 seconds. A new "toast" type of notification is displayed along the status update. https://github.com/nexB/scancode.io/issues/390
- Ensure the worker service waits for migrations completion before starting. To solve this issue we install the wait-for-it script available in Debian by @vishnubob and as suggested in the Docker documentation. In the docker-compose.yml, we let the worker wait for the web processing to be complete when gunicorn exposes port 8000 and web container is available. Reference: https://docs.docker.com/compose/startup-order/ Reference: https://github.com/vishnubob/wait-for-it Reference: https://tracker.debian.org/pkg/wait-for-it https://github.com/nexB/scancode.io/issues/387
- Add a "create-user" management command to create new user with its API key. https://github.com/nexB/scancode.io/issues/458
- Add a "tag" field on the CodebaseResource model. The layer details are stored in this field in the "docker" pipeline. https://github.com/nexB/scancode.io/issues/443
- Add support for multiple inputs in the LoadInventory pipeline. https://github.com/nexB/scancode.io/issues/451
- Add new SCANCODEIO_REDIS_PASSWORD environment variable and setting to optionally set Redis instance password.
- Ensure a project cannot be deleted through the API while a pipeline is running. https://github.com/nexB/scancode.io/issues/402
- Display "License clarity" and "Scan summary" values as new panel in the project details view. The summary is generated during the *scan_package* pipeline. https://github.com/nexB/scancode.io/issues/411
- Enhance Project list view page:
  - 20 projects are now displayed per page
  - Creation date displayed under the project name
  - Add ability to sort by date and name
  - Add ability to filter by pipeline type
  - Add ability to filter by run status

  https://github.com/nexB/scancode.io/issues/413

- Correctly extract symlinks in docker images. We now use the latest container-inspector to fix symlinks extraction in docker image tarballs. In particular broken symlinks are not treated as an error anymore and symlinks are extracted correctly. https://github.com/nexB/scancode.io/issues/471 https://github.com/nexB/scancode.io/issues/407
- Add a Package details view including all model fields and resources. Display only 5 resources per package in the list view. https://github.com/nexB/scancode.io/issues/164 https://github.com/nexB/scancode.io/issues/464
- Add the ability to filter by empty and none values providing the "EMPTY" magic value to any filters. https://github.com/nexB/scancode.io/issues/296
- CodebaseResource.name now contains both the bare file name with extension, as opposed to just the bare file name without extension. Using a name stripped from its extension was something that was not used in other AboutCode project or tools. https://github.com/nexB/scancode.io/issues/467
- Export current results as XLSX for resource, packages, and errors list views. https://github.com/nexB/scancode.io/issues/48
- Add support for .tgz extension for input files in Docker pipeline https://github.com/nexB/scancode.io/issues/499
- Add support for resource missing file content in details view. Refine the annotation using the new className instead of type. https://github.com/nexB/scancode.io/issues/495
- Change the worksheet names in XLSX output, using the "PACKAGES", "RESOURCES", "DEPENDENCIES", and "ERRORS" names. https://github.com/nexB/scancode.io/issues/511
- Update application Package scanning step to reflect the updates in scancode-toolkit package scanning.
  - Package data detected from a file are now stored on the CodebaseResource.package_data field.
  - A second processing step is now done after scanning for Package data, where Package Resources are determined and DiscoveredPackages and DiscoveredDependencies are created.

  https://github.com/nexB/scancode.io/issues/444

6.21 v30.2.0 (2021-12-17)

- Add authentication for the Web UI views and REST API endpoint. The autentication is disabled by default and can be enabled using the SCANCODEIO_REQUIRE_AUTHENTICATION settings. When enabled, users have to authenticate through a login form in the Web UI, or using their API Key in the REST API. The API Key can be viewed in the Web UI "Profile settings" view ince logged-in. Users can be created using the Django "createsuperuser" management command. https://github.com/nexB/scancode.io/issues/359
- Include project errors in XLSX results output. https://github.com/nexB/scancode.io/issues/364
- Add input_sources used to fetch inputs to JSON results output. https://github.com/nexB/scancode.io/issues/351
- Refactor the update_or_create_package pipe to support the ProjectError system and fix a database transaction error. https://github.com/nexB/scancode.io/issues/381
- Add webhook subscription available when creating project from REST API. https://github.com/nexB/scancode.io/issues/98
- Add the project "reset" feature in the UI, CLI, and REST API. https://github.com/nexB/scancode.io/issues/375
- Add a new GitHub action that build the docker-compose images and run the test suite. This ensure that the app is properly working and tested when running with Docker. https://github.com/nexB/scancode.io/issues/367
- Add –no-install-recommends in the Dockerfile apt-get install and add the *linux-image-amd64* package. This packages makes available the kernels required by extractcode and libguestfs for proper VM images extraction. https://github.com/nexB/scancode.io/issues/367

- Add a new *list-project* CLI command to list projects. https://github.com/nexB/scancode.io/issues/365

6.22 v30.1.1 (2021-11-23)

- Remove the –no-install-recommends in the Dockerfile apt-get install to include required dependencies for proper VM extraction. https://github.com/nexB/scancode.io/issues/367

6.23 v30.1.0 (2021-11-22)

- Synchronize QUEUED and RUNNING pipeline runs with their related worker jobs during worker maintenance tasks scheduled every 10 minutes. If a container was taken down while a pipeline was running, or if pipeline process was killed unexpectedly, that pipeline run status will be updated to a FAILED state during the next maintenance tasks. QUEUED pipeline will be restored in the queue as the worker redis cache backend data is now persistent and reloaded on starting the image. Note that internaly, a running job emits a "heartbeat" every 60 seconds to let all the workers know that it is properly running. After 90 seconds without any heartbeats, a worker will determine that the job is not active anymore and that job will be moved to the failed registry during the worker maintenance tasks. The pipeline run will be updated as well to reflect this failure in the Web UI, the REST API, and the command line interface. https://github.com/nexB/scancode.io/issues/130
- Enable redis data persistence using the "Append Only File" with the default policy of fsync every second in the docker-compose. https://github.com/nexB/scancode.io/issues/130
- Add a new tutorial chapter about license policies and compliance alerts. https://github.com/nexB/scancode.io/issues/337
- Include layers in docker image data. https://github.com/nexB/scancode.io/issues/175
- Fix a server error on resource details view when the compliance alert is "missing". https://github.com/nexB/scancode.io/issues/344
- Migrate the ScanCodebase pipeline from *scancode.run_scancode* subprocess to *scancode.scan_for_application_packages* and *scancode.scan_for_files*. https://github.com/nexB/scancode.io/issues/340

6.24 v30.0.1 (2021-10-11)

- Fix a build failure related to dependency conflict. https://github.com/nexB/scancode.io/issues/342

6.25 v30.0.0 (2021-10-8)

- Upgrade ScanCode-toolkit to version 30.1.0
- Replace the task queue system, from Celery to RQ. https://github.com/nexB/scancode.io/issues/176
- Add ability to delete "not started" and "queued" pipeline tasks. https://github.com/nexB/scancode.io/issues/176
- Add ability to stop "running" pipeline tasks. https://github.com/nexB/scancode.io/issues/176
- Refactor the "execute" management command and add support for –async mode. https://github.com/nexB/scancode.io/issues/130

- Include codebase resource data in the details of package creation project errors. https://github.com/nexB/scancode.io/issues/208
- Add a SCANCODEIO_REST_API_PAGE_SIZE setting to control the number of objects returned per page in the REST API. https://github.com/nexB/scancode.io/issues/328
- Provide an "add input" action on the Project endpoint of the REST API. https://github.com/nexB/scancode.io/issues/318

6.26 v21.9.6

- Add ability to "archive" projects, from the Web UI, API and command line interface. Data cleanup of the project's input, codebase, and output directories is available during the archive operation. Archived projects cannot be modified anymore and are hidden by default from the project list. A project cannot be archived if one of its related run is queued or already running. https://github.com/nexB/scancode.io/issues/312
- Remove the run_extractcode pipe in favor of extractcode API. https://github.com/nexB/scancode.io/issues/312
- The *scancode.run_scancode* pipe now uses an optimal number of available CPUs for multiprocessing by default. The exact number of parallel processes available to ScanCode.io can be defined using the SCANCODEIO_PROCESSES setting. https://github.com/nexB/scancode.io/issues/302
- Renamed the SCANCODE_DEFAULT_OPTIONS setting to SCANCODE_TOOLKIT_CLI_OPTIONS. https://github.com/nexB/scancode.io/issues/302
- Log the outputs of run_scancode as progress indication. https://github.com/nexB/scancode.io/issues/300

6.27 v21.8.2

- Upgrade ScanCode-toolkit to version 21.7.30
- Add new documentation chapters and tutorials on the usage of the Web User Interface. https://github.com/nexB/scancode.io/issues/241
- Add ability to register custom pipelines through a new SCANCODEIO_PIPELINES_DIRS setting. https://github.com/nexB/scancode.io/issues/237
- Add a pipeline *scan_package.ScanPackage* to scan a single package archive with ScanCode-toolkit. https://github.com/nexB/scancode.io/issues/25
- Detected Package dependencies are not created as Package instance anymore but stored on the Package model itself in a new *dependencies* field. https://github.com/nexB/scancode.io/issues/228
- Add the extra_data field on the DiscoveredPackage model. https://github.com/nexB/scancode.io/issues/191
- Improve XLSX creation. We now check that the content is correctly added before calling XlsxWriter and report and error if the truncated can be truncated. https://github.com/nexB/scancode.io/issues/206
- Add support for VMWare Photon-based Docker images and rootfs. This is an RPM-based Linux distribution

6.28 v21.6.10

- Add support for VM image formats extraction such as VMDK, VDI and QCOW. See https://github.com/nexB/extractcode#archive-format-kind-file_system for the full list of supported extensions. The new extraction feature requires the installation of *libguestfs-tools*, see https://github.com/nexB/extractcode#adding-support-for-vm-images-extraction for installation details. https://github.com/nexB/scancode.io/issues/132
- Add the ability to disable multiprocessing and threading entirely through the SCANCODEIO_PROCESSES setting. Use 0 to disable multiprocessing and use -1 to also disable threading. https://github.com/nexB/scancode.io/issues/185
- Missing project workspace are restored on reports (xlsx, json) creation. This allow to download reports even if the project workspace (input, codebase) was deleted. https://github.com/nexB/scancode.io/issues/154
- Add ability to search on all list views. https://github.com/nexB/scancode.io/issues/184
- Add the is_binary, is_text, and is_archive fields to the CodebaseResource model. https://github.com/nexB/scancode.io/issues/75

6.29 v21.5.12

- Adds a new way to fetch docker images using skopeo provided as a plugin using docker:// reference URL-like pointers to a docker image. The syntax is docker://<docker image> where <docker image> is the string that would be used in a "docker pull <docker image>" command. Also rename scanpipe.pipes.fetch.download() to fetch_http() https://github.com/nexB/scancode.io/issues/174
- Pipeline status modals are now loaded asynchronously and available from the project list view.
- Fix an issue accessing codebase resource content using the scan_codebase and load_inventory pipelines. https://github.com/nexB/scancode.io/issues/147

6.30 v21.4.28

- The installation local timezone can be configured using the TIME_ZONE setting. The current timezone in now included in the dates representation in the web UI. https://github.com/nexB/scancode.io/issues/142
- Fix pipeline failure issue related to the assignment of un-saved (not valid) packages. https://github.com/nexB/scancode.io/issues/162
- Add a new QUEUED status to differentiate a pipeline that is in the queue for execution from a pipeline execution not requested yet. https://github.com/nexB/scancode.io/issues/130
- Refactor the multiprocessing code for file and package scanning. All database related operation are now executed in the main process as forking the existing database connection in sub-processes is a source of issues. Add progress logging for scan_for_files and scan_for_application_packages pipes. https://github.com/nexB/scancode.io/issues/145
- Links from the charts to the resources list are now also filtered by in_package/not_in_package if enabled on the project details view. https://github.com/nexB/scancode.io/issues/124
- Add ability to filter on codebase resource detected values such as licenses, copyrights, holders, authors, emails, and urls. https://github.com/nexB/scancode.io/issues/153
- Filtered list views from a click on chart sections can now be opened in a new tab using ctrl/meta + click. https://github.com/nexB/scancode.io/issues/125

- Add links to codebase resource and to discovered packages in list views.

6.31 v21.4.14

- Implement timeout on the scan functions, default to 120 seconds per resources. https://github.com/nexB/scancode.io/issues/135
- Fix issue with closing modal buttons in the web UI. https://github.com/nexB/scancode.io/issues/116 https://github.com/nexB/scancode.io/issues/141

6.32 v21.4.5

- Add support for Docker and VM images using RPMs such as Fedora, CentOS, RHEL, and openSUSE linux distributions. https://github.com/nexB/scancode.io/issues/6
- Add a compliance alert system based on license policies provided through a policies.yml file. The compliance alerts are computed from the license_expression and stored on the codebase resource. When the policy feature is enabled, the compliance alert values are displayed in the UI and returned in all the downloadable results. The enable and setup the policy feature, refer to https://scancodeio.readthedocs.io/en/latest/scancodeio-settings.html#scancode-io-settings https://github.com/nexB/scancode.io/issues/90
- Add a new codebase resource detail view including the file content. Detected value can be displayed as annotation in the file source. https://github.com/nexB/scancode.io/issues/102
- Download URLs can be provided as inputs on the project form. Each URL is fetched and added to the project input directory. https://github.com/nexB/scancode.io/issues/100
- Run celery worker with the "threads" pool implementation. Implement parallelization with ProcessPoolExecutor for file and package scans. Add a SCANCODEIO_PROCESSES settings to control the multiprocessing CPUs count. https://github.com/nexB/scancode.io/issues/70
- Optimize "tag" type pipes using the update() API in place of save() on the QuerySet iteration. https://github.com/nexB/scancode.io/issues/70
- Use the extractcode API for the Docker pipeline. This change helps with performance and results consistency between pipelines. https://github.com/nexB/scancode.io/issues/70
- Implement cache to prevent scanning multiple times a duplicated codebase resource. https://github.com/nexB/scancode.io/issues/70
- Create the virtualenv using the virtualenv.pyz app in place of the bundled "venv". https://github.com/nexB/scancode.io/issues/104
- Consistent ordering for the pipelines, now sorted alphabetically.

6.33 v1.1.0 (2021-02-16)

- Display project extra data in the project details view. https://github.com/nexB/scancode.io/issues/88
- Add a @profile decorator for profiling pipeline step execution. https://github.com/nexB/scancode.io/issues/73
- Support inputs as tarballs in root_filesystem pipelines. The input archives are now extracted with extractcode to the codebase/ directory. https://github.com/nexB/scancode.io/issues/96
- Improve support for unknown distros in docker and root_filesystem pipelines. The pipeline logs the distro errors on the project instead of failing. https://github.com/nexB/scancode.io/issues/97
- Implement Pipeline registration through distribution entry points. Pipeline can now be installed as part of external libraries. With this change pipelines are no longer referenced by the Python script path, but by their registered name. This is a breaking command line API change. https://github.com/nexB/scancode.io/issues/91
- Add a "Run Pipeline" button in the Pipeline modal of the Project details view. Pipelines can now be added from the Project details view. https://github.com/nexB/scancode.io/issues/84
- Upgrade scancode-toolkit to version 21.2.9
- Allow to start the pipeline run immediately on addition in the *add_pipeline* action of the Project API endpoint. https://github.com/nexB/scancode.io/issues/92
- Rename the pipes.outputs module to pipes.output for consistency.
- Remove the dependency on Metaflow. WARNING: The new Pipelines syntax is not backward compatible with v1.0.x https://github.com/nexB/scancode.io/issues/82

6.34 v1.0.7 (2021-02-01)

- Add user interface to manage Projects from a web browser All the command-line features are available https://github.com/nexB/scancode.io/issues/24
- Log messages from Pipeline execution on a new Run instance *log* field https://github.com/nexB/scancode.io/issues/66
- Add support for scancode pipes and Project name with whitespaces
- Add a profile() method on the Run model for profiling pipeline execution https://github.com/nexB/scancode.io/issues/73

6.35 v1.0.6 (2020-12-23)

- Add a management command to delete a Project and its related work directories https://github.com/nexB/scancode.io/issues/65
- Add CSV and XLSX support for the *output* management command https://github.com/nexB/scancode.io/issues/46
- Add a to_xlsx output pipe returning XLSX compatible content https://github.com/nexB/scancode.io/issues/46
- Add a "status" management command to display Project status information https://github.com/nexB/scancode.io/issues/66
- Fix the env_file location to run commands from outside the root dir https://github.com/nexB/scancode.io/issues/64

- Add utilities to save project error in the database during Pipeline execution https://github.com/nexB/scancode.io/issues/64
- Install psycopg2-binary instead of psycopg2 on non-Linux platforms https://github.com/nexB/scancode.io/issues/64

6.36 v1.0.5 (2020-12-07)

- Add minimal license list and text views https://github.com/nexB/scancode.io/issues/32
- Add admin actions to export selected objects to CSV and JSON The output content, such as included fields, can be configured for CSV format https://github.com/nexB/scancode.io/issues/48 https://github.com/nexB/scancode.io/issues/49
- Add –list option to the graph management command. Multiple graphs can now be generated at once.
- Add ProjectCodebase to help walk and navigate Project CodebaseResource loaded from the Database Add also a get_tree function compatible with scanpipe.CodebaseResource and commoncode.Resource https://github.com/nexB/scancode.io/issues/52
- Add support for running ScanCode.io as a Docker image https://github.com/nexB/scancode.io/issues/9
- Add support for Python 3.7, 3.8, and 3.9 https://github.com/nexB/scancode.io/issues/54

6.37 v1.0.4 (2020-11-17)

- Add a to_json output pipe returning ScanCode compatible content https://github.com/nexB/scancode.io/issues/45
- Improve Admin UI for efficient review: display, navigation, filters, and ability to view file content https://github.com/nexB/scancode.io/issues/36
- Add Pipelines and Pipes documentation using Sphinx autodoc Fix for https://github.com/nexB/scancode.io/issues/38
- Add new ScanCodebase pipeline for codebase scan Fix for https://github.com/nexB/scancode.io/issues/37
- Upgrade Django, Metaflow, and ScanCode-toolkit to latest versions

6.38 v1.0.3 (2020-09-24)

- Add ability to resume a failed pipeline from the run management command Fix for https://github.com/nexB/scancode.io/issues/22
- Use project name as argument to run a pipeline Fix for https://github.com/nexB/scancode.io/issues/18
- Add support for "failed" task_output in Run.get_run_id method Fix for https://github.com/nexB/scancode.io/issues/17

6.39 v1.0.2 (2020-09-18)

- Add documentation and tutorial For https://github.com/nexB/scancode.io/issues/8
- Add a create-project, add-input, add-pipeline, run, output management commands to expose ScanPipe features through the command line Fix for https://github.com/nexB/scancode.io/issues/13
- Always return the Pipeline subclass/implementation from the module inspection Fix for https://github.com/nexB/scancode.io/issues/11

6.40 v1.0.1 (2020-09-12)

- Do not fail when collecting system packages in Ubuntu docker images for layers that do not install packages by updating to a newer version of ScanCode Toolkit Fix for https://github.com/nexB/scancode.io/issues/1

6.41 v1.0.0 (2020-09-09)

- Initial release

CHAPTER

SEVEN

ANALYZE DOCKER IMAGE (WEB UI)

This tutorial aims to show you how to scan a docker image input file using the ScanCode Web UI while introducing you to the interface's various features.

Tip: This tutorial is intended for anyone who prefers interacting with a visual interface when working with ScanCode.io. If you prefer using the command line, you can check our command line tutorial: *Analyze Codebase (Command Line)*.

Note: This tutorial assumes you have a current version of ScanCode.io installed locally on your machine with an access to the ScanCode Web UI. If you do not have them already, you can take a look at our *Installation* guide for instructions.

7.1 Requirements

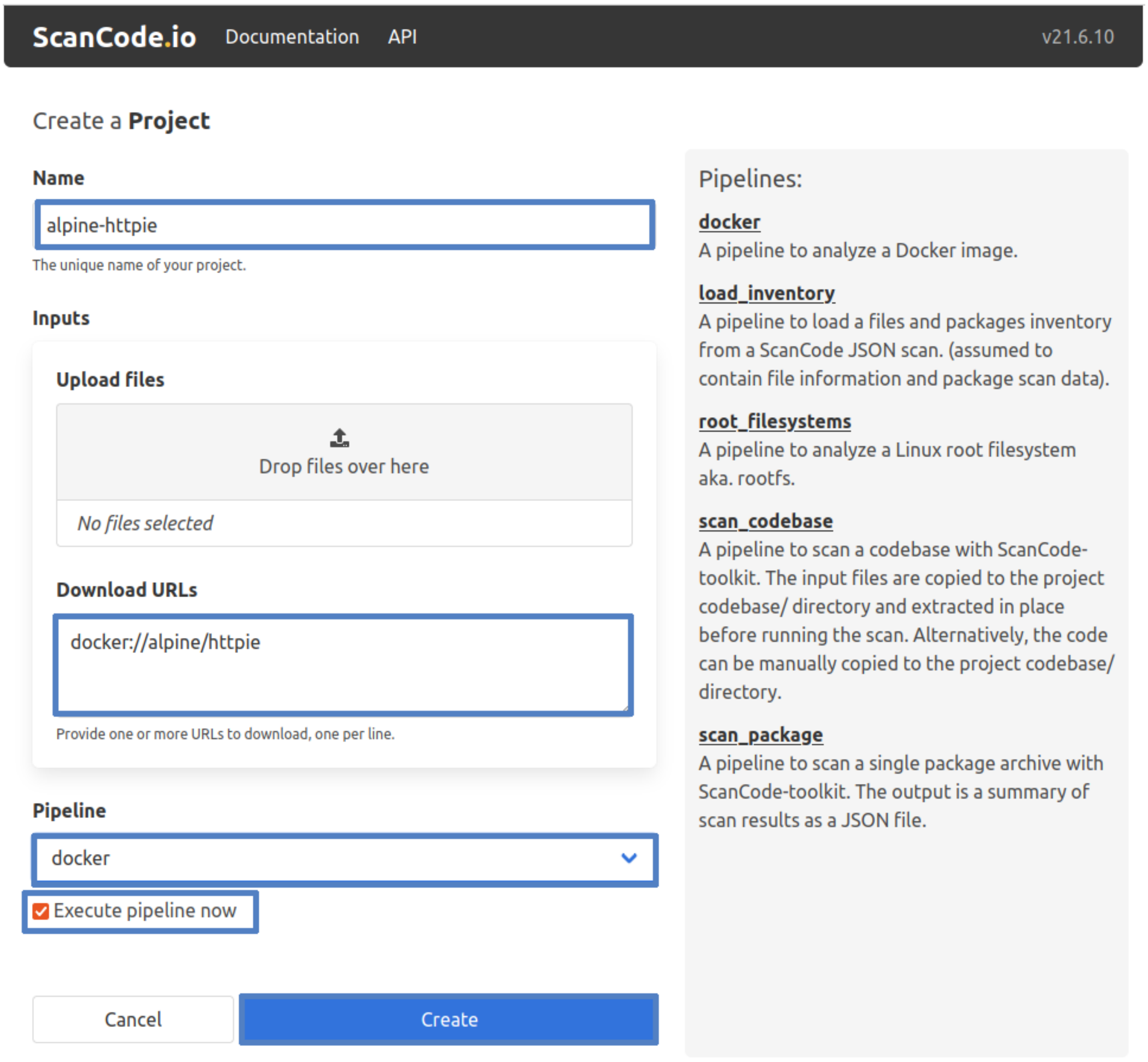
We'll assume that you have:

- Installed **ScanCode.io** locally
- Access to the web application from your preferred browser on http://localhost/ or http://localhost:8001/ if you run on a local development setup.

Tip: You can view our *User Interface* section for general information about the ScanCode.io Web UI.

7.2 Instructions

- From the homepage, click on the **"New Project"** button to create a new project named `alpine-httpie`. You will be directed to the **"Create a Project"** page where you need to fill in the new project's details.
- Paste the input Docker image's URL, docker://alpine/httpie, in the **"Download URL"** field, which fetches the image from the provided URL.
- Use the **"Pipeline"** dropdown list, add the `analyze_docker_image` pipeline to your project.
- You can add and execute the `analyze_docker_image` pipeline in one operation by checking the **"Execute pipeline now"** checkbox.



Create a **Project**

Name

alpine-httpie

The unique name of your project.

Inputs

Upload files

Drop files over here

No files selected

Download URLs

docker://alpine/httpie

Provide one or more URLs to download, one per line.

Pipeline

docker

✓ Execute pipeline now

Pipelines:

docker
A pipeline to analyze a Docker image.

load_inventory
A pipeline to load a files and packages inventory from a ScanCode JSON scan. (assumed to contain file information and package scan data).

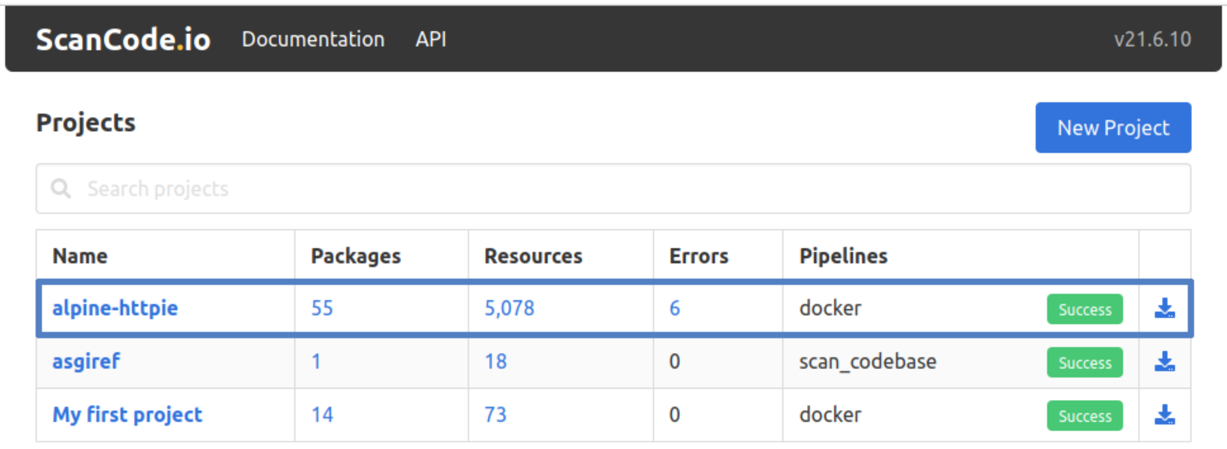
root_filesystems
A pipeline to analyze a Linux root filesystem aka. rootfs.

scan_codebase
A pipeline to scan a codebase with ScanCode-toolkit. The input files are copied to the project codebase/ directory and extracted in place before running the scan. Alternatively, the code can be manually copied to the project codebase/ directory.

scan_package
A pipeline to scan a single package archive with ScanCode-toolkit. The output is a summary of scan results as a JSON file.

Note: You can create a new project while leaving the **Inputs** and **Pipeline** fields blank; however, it's required to provide a project **Name**!

- Finally, click the **"Create"** button



Name	Packages	Resources	Errors	Pipelines	
alpine-httpie	55	5,078	6	docker Success	
asgiref	1	18	0	scan_codebase Success	
My first project	14	73	0	docker Success	

Note: Please note that when you choose to create a new project and execute the pipeline in one operation, the process may take few minutes before it completes.

The previous screenshot shows the ScanCode.io home screen with the new "alpine-httpie" project and other existing projects. The home screen also shows a summary of the number of **Packages**, **Code Resources**, and **Errors**—if any—discovered during the scan process. It also contains any **Pipelines** used and their execution status, i.e.:

- **Not started**
- **Queued**
- **Running**
- **Success**
- **Failure**

Plus, the ability to download the generated results in **JSON** and **Excel (XLSX)** file formats, covered in *Output Files*.

Tip: Refer to the complementary *Review Scan Results (Web UI)* page, to understand this tutorial's scan results/output.

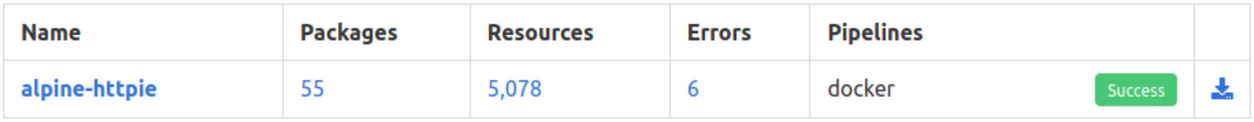
CHAPTER

EIGHT

REVIEW SCAN RESULTS (WEB UI)

This chapter is complementary to the *Analyze Docker Image (Web UI)* tutorial, and the output included here represents the generated results of the tutorial's pipeline run. The goal here is to guide you on how to understand and review your scan results using the ScanCode.io web interface.

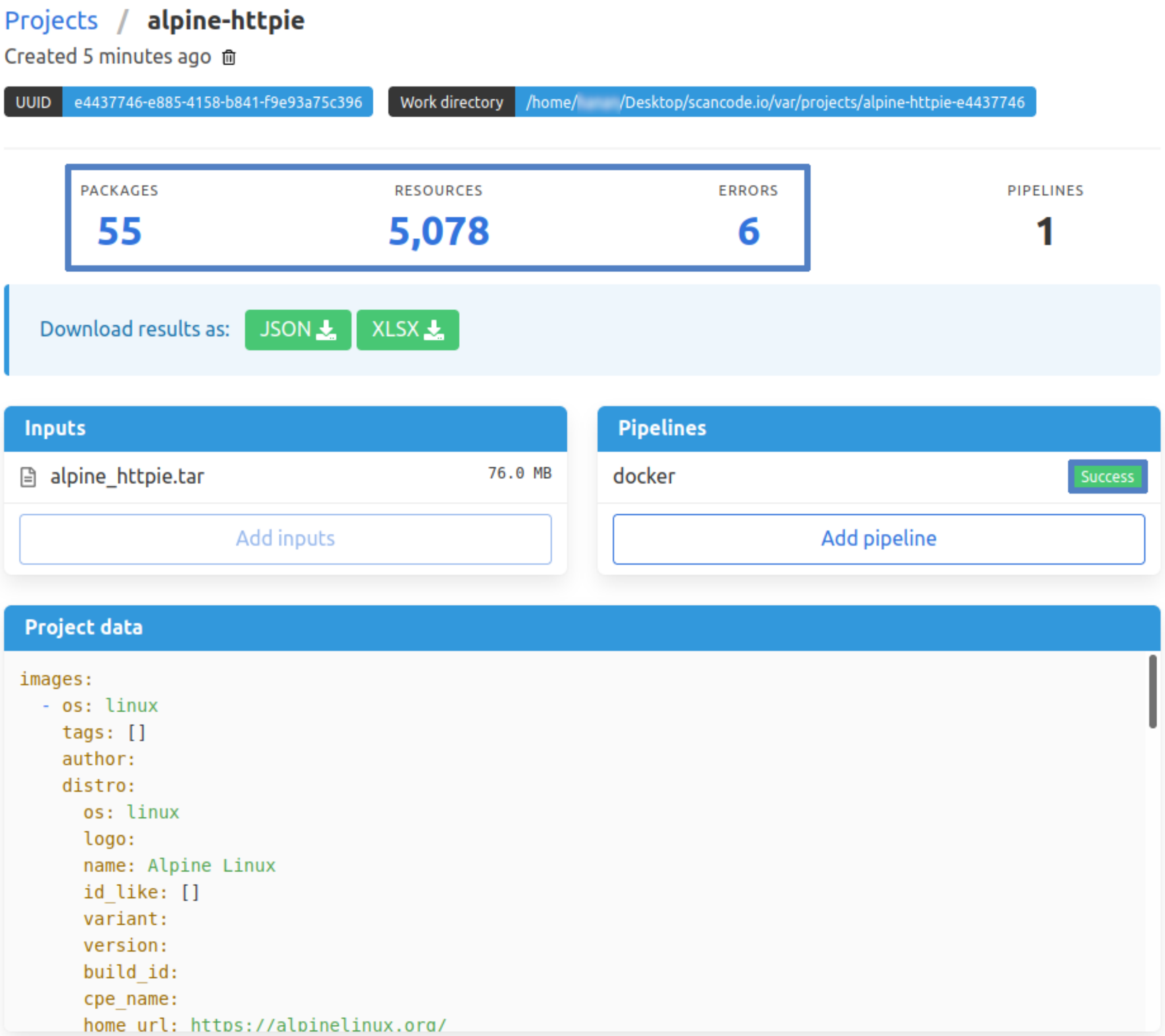
Tip: As a perquisite, follow the *Analyze Docker Image (Web UI)* tutorial to have a better understanding of the information included here.



Name	Packages	Resources	Errors	Pipelines	
alpine-httpie	55	5,078	6	docker Success	

On the homepage, you can click on the project name in the summary table to access a detailed project output page. You can also click any of the numbers underneath the **Packages**, **Resources**, or **Errors** fields to be directed to the field's corresponding information. Further, clicking on the pipeline's execution status —**Success** in this case— expands some extra pipeline details, Resources status, and Run log.

Note: You can also view output-related information, in graphical representation, in each project page. Plus, other project data and input details.



Projects / **alpine-httpie**

Created 5 minutes ago

UUID e4437746-e885-4158-b841-f9e93a75c396 Work directory /home/[redacted]/Desktop/scancode.io/var/projects/alpine-httpie-e4437746

PACKAGES	RESOURCES	ERRORS	PIPELINES
55	5,078	6	1

Download results as: JSON XLSX

Inputs

alpine_httpie.tar 76.0 MB

Add inputs

Pipelines

docker Success

Add pipeline

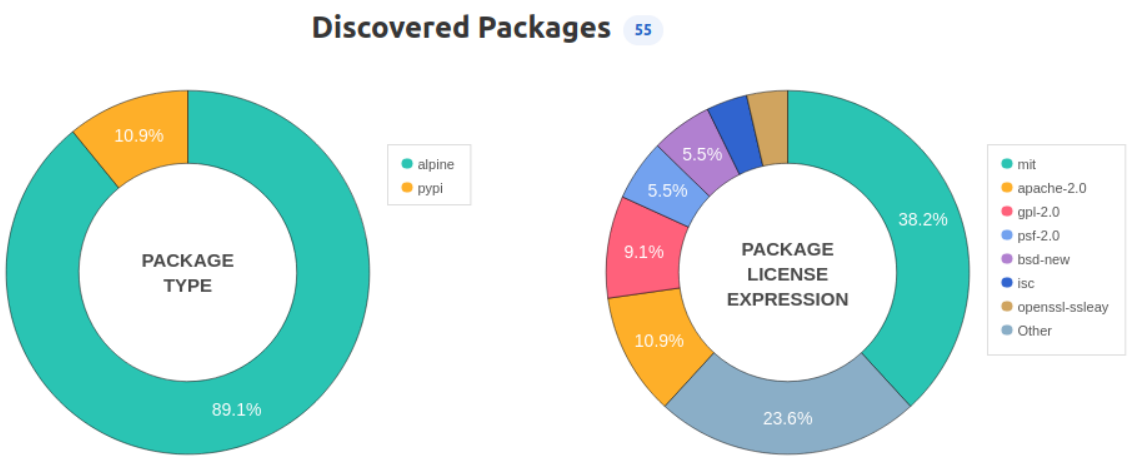
Project data

```
images:
  - os: linux
    tags: []
    author:
    distro:
      os: linux
      logo:
      name: Alpine Linux
      id_like: []
      variant:
      version:
      build_id:
      cpe_name:
      home_url: https://alpinelinux.org/
```

In general, the output of any pipeline run includes details about:

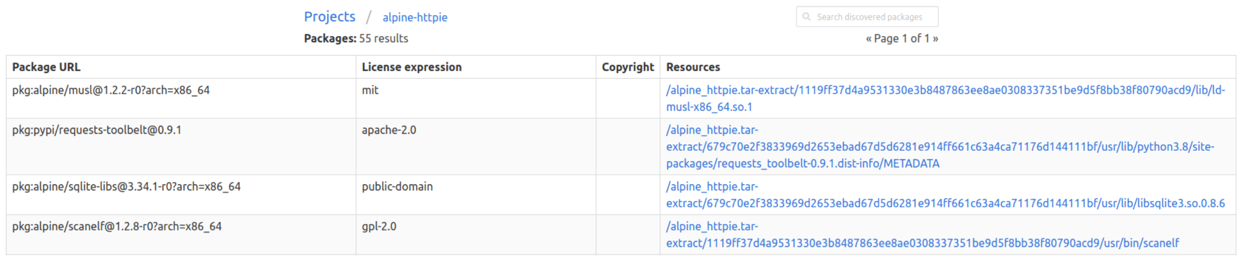
8.1 Packages

A summary of **Discovered Packages** in a graphical format is shown in the project page that includes two Pie/Doughnut charts, which filter all packages found by their **Type** and **License Expression**:



The two circular charts are interactive and can be filtered further by clicking any of the categories—**Type** and **License Expression**—on the right of each graph. Also, you can expand more details about any category separately, in tabular format, when you click its corresponding slice colour.

In addition, there is also an overall detailed table that includes all packages found and code resources within each package, which can be accessed by clicking on the **Packages** number field.



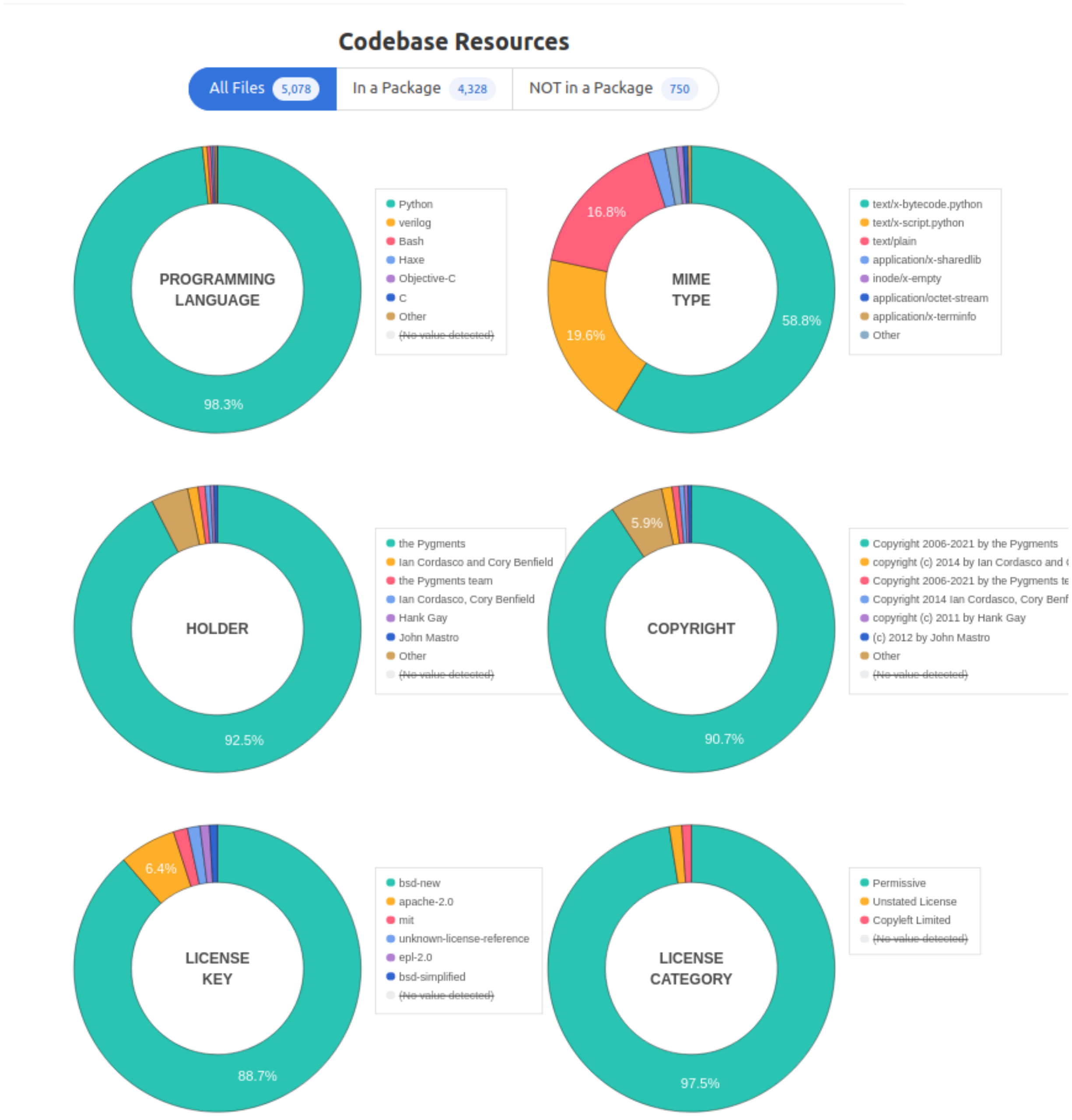
Packages: 55 results

Package URL	License expression	Copyright	Resources
pkg:alpine/musl@1.2.2-r0?arch=x86_64	mit		/alpine_httpie.tar-extract/1119ff37d4a9531330e3b8487863ee8ae0308337351be9d5f8bb38f80790acd9/lib/ld-musl-x86_64.so.1
pkg:pypi/requests-toolbelt@0.9.1	apache-2.0		/alpine_httpie.tar-extract/679c70e2f3833969d2653ebad67d5d6281e914ff661c63a4ca71176d144111bf/usr/lib/python3.8/site-packages/requests_toolbelt-0.9.1.dist-info/METADATA
pkg:alpine/sqlite-libs@3.34.1-r0?arch=x86_64	public-domain		/alpine_httpie.tar-extract/679c70e2f3833969d2653ebad67d5d6281e914ff661c63a4ca71176d144111bf/usr/lib/libsqlite3.so.0.8.6
pkg:alpine/scanelf@1.2.8-r0?arch=x86_64	gpl-2.0		/alpine_httpie.tar-extract/1119ff37d4a9531330e3b8487863ee8ae0308337351be9d5f8bb38f80790acd9/usr/bin/scanelf

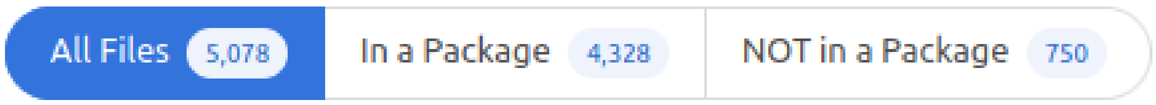
8.2 Resources

Similar to **Packages**, the total number of discovered **Codebase Resources** is shown on both ScanCode.io homepage and the "alpine-httpie" project page. Clicking on this number reveals a detailed table for all found code resources.

Further, the project page offers a group of Doughnut charts that filter code resources by **Programming Language**, **Mime Type**, **Holder**, **Copyright**, **License Key**, and **License Category**.

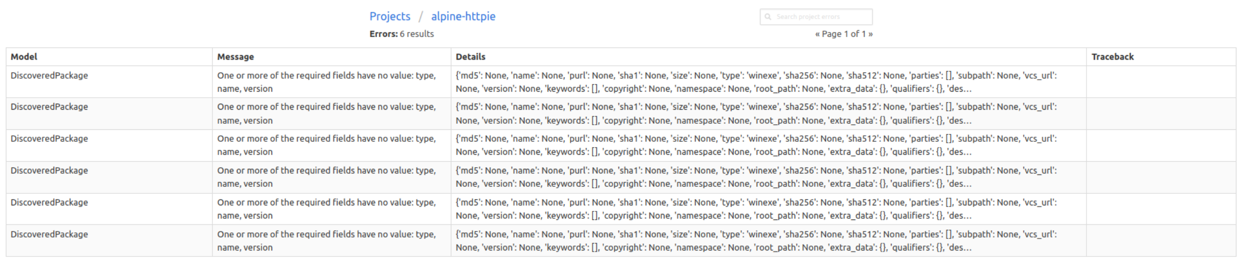


Note: The charts above show all discovered Codebase files by default regardless of their existence within a package. You can still only view a subset, i.e., **In a Package** or **Not in a Package**



8.3 Errors

In addition to discovered packages and codebase resources, the ScanCode.io homepage shows the number of existing errors, which you can click for detailed description of each error.



Projects / alpine-httpie

Errors: 6 results

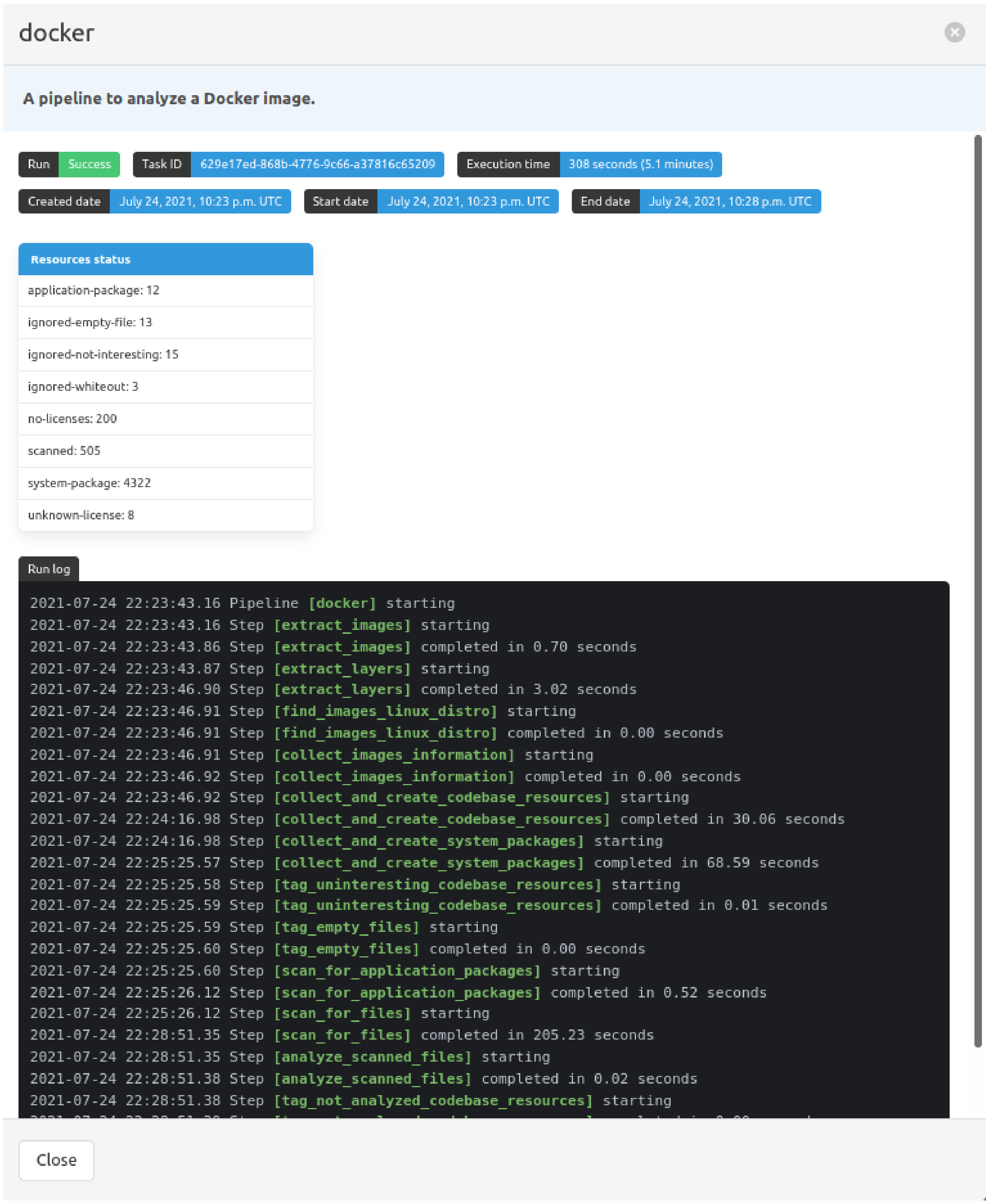
Search project errors

« Page 1 of 1 »

Model	Message	Details	Traceback
DiscoveredPackage	One or more of the required fields have no value: type, name, version	{'md5': None, 'name': None, 'purl': None, 'sha1': None, 'size': None, 'type': 'winexe', 'sha256': None, 'sha512': None, 'parties': [], 'subpath': None, 'vcs_url': None, 'version': None, 'keywords': [], 'copyright': None, 'namespace': None, 'root_path': None, 'extra_data': {}, 'qualifiers': {}, 'des...	
DiscoveredPackage	One or more of the required fields have no value: type, name, version	{'md5': None, 'name': None, 'purl': None, 'sha1': None, 'size': None, 'type': 'winexe', 'sha256': None, 'sha512': None, 'parties': [], 'subpath': None, 'vcs_url': None, 'version': None, 'keywords': [], 'copyright': None, 'namespace': None, 'root_path': None, 'extra_data': {}, 'qualifiers': {}, 'des...	
DiscoveredPackage	One or more of the required fields have no value: type, name, version	{'md5': None, 'name': None, 'purl': None, 'sha1': None, 'size': None, 'type': 'winexe', 'sha256': None, 'sha512': None, 'parties': [], 'subpath': None, 'vcs_url': None, 'version': None, 'keywords': [], 'copyright': None, 'namespace': None, 'root_path': None, 'extra_data': {}, 'qualifiers': {}, 'des...	
DiscoveredPackage	One or more of the required fields have no value: type, name, version	{'md5': None, 'name': None, 'purl': None, 'sha1': None, 'size': None, 'type': 'winexe', 'sha256': None, 'sha512': None, 'parties': [], 'subpath': None, 'vcs_url': None, 'version': None, 'keywords': [], 'copyright': None, 'namespace': None, 'root_path': None, 'extra_data': {}, 'qualifiers': {}, 'des...	
DiscoveredPackage	One or more of the required fields have no value: type, name, version	{'md5': None, 'name': None, 'purl': None, 'sha1': None, 'size': None, 'type': 'winexe', 'sha256': None, 'sha512': None, 'parties': [], 'subpath': None, 'vcs_url': None, 'version': None, 'keywords': [], 'copyright': None, 'namespace': None, 'root_path': None, 'extra_data': {}, 'qualifiers': {}, 'des...	
DiscoveredPackage	One or more of the required fields have no value: type, name, version	{'md5': None, 'name': None, 'purl': None, 'sha1': None, 'size': None, 'type': 'winexe', 'sha256': None, 'sha512': None, 'parties': [], 'subpath': None, 'vcs_url': None, 'version': None, 'keywords': [], 'copyright': None, 'namespace': None, 'root_path': None, 'extra_data': {}, 'qualifiers': {}, 'des...	

8.4 Other Information

Clicking on the pipeline's execution status —**Success** in this case— opens a new window with some extra pipeline-specific details, such start and end date, launch, execution time and status, run log, etc.



A pipeline to analyze a Docker image.

Run Success | Task ID 629e17ed-868b-4776-9c66-a37816c65209 | Execution time 308 seconds (5.1 minutes)

Created date July 24, 2021, 10:23 p.m. UTC | Start date July 24, 2021, 10:23 p.m. UTC | End date July 24, 2021, 10:28 p.m. UTC

Resources status
application-package: 12
ignored-empty-file: 13
ignored-not-interesting: 15
ignored-whiteout: 3
no-licenses: 200
scanned: 505
system-package: 4322
unknown-license: 8

Run log

```
2021-07-24 22:23:43.16 Pipeline [docker] starting
2021-07-24 22:23:43.16 Step [extract_images] starting
2021-07-24 22:23:43.86 Step [extract_images] completed in 0.70 seconds
2021-07-24 22:23:43.87 Step [extract_layers] starting
2021-07-24 22:23:46.90 Step [extract_layers] completed in 3.02 seconds
2021-07-24 22:23:46.91 Step [find_images_linux_distro] starting
2021-07-24 22:23:46.91 Step [find_images_linux_distro] completed in 0.00 seconds
2021-07-24 22:23:46.91 Step [collect_images_information] starting
2021-07-24 22:23:46.92 Step [collect_images_information] completed in 0.00 seconds
2021-07-24 22:23:46.92 Step [collect_and_create_codebase_resources] starting
2021-07-24 22:24:16.98 Step [collect_and_create_codebase_resources] completed in 30.06 seconds
2021-07-24 22:24:16.98 Step [collect_and_create_system_packages] starting
2021-07-24 22:25:25.57 Step [collect_and_create_system_packages] completed in 68.59 seconds
2021-07-24 22:25:25.58 Step [tag_uninteresting_codebase_resources] starting
2021-07-24 22:25:25.59 Step [tag_uninteresting_codebase_resources] completed in 0.01 seconds
2021-07-24 22:25:25.59 Step [tag_empty_files] starting
2021-07-24 22:25:25.60 Step [tag_empty_files] completed in 0.00 seconds
2021-07-24 22:25:25.60 Step [scan_for_application_packages] starting
2021-07-24 22:25:26.12 Step [scan_for_application_packages] completed in 0.52 seconds
2021-07-24 22:25:26.12 Step [scan_for_files] starting
2021-07-24 22:28:51.35 Step [scan_for_files] completed in 205.23 seconds
2021-07-24 22:28:51.35 Step [analyze_scanned_files] starting
2021-07-24 22:28:51.38 Step [analyze_scanned_files] completed in 0.02 seconds
2021-07-24 22:28:51.38 Step [tag_not_analyzed_codebase_resources] starting
```

Tip: The next tutorial chapter *Analyze Docker Image (Command Line)* will explore the interaction with ScanCode.io through a command line interface.

CHAPTER

NINE

ANALYZE DOCKER IMAGE (COMMAND LINE)

In this tutorial, you will learn by example how to use ScanCode.io to analyze a test Docker image by following the steps below and, along the way, learn some of the ScanCode.io basic commands.

Note: This tutorial assumes you have a recent version of ScanCode.io installed locally on your machine and **running with Docker**. If you do not have it installed, see our *Installation* guide for instructions.

9.1 Requirements

To successfully complete this tutorial, you first need to:

- Install **ScanCode.io** locally
- Have **Shell access** on the machine where ScanCode.io is installed

9.2 Instructions

- Create a new directory in your home directory that will be used to put the input code to be scanned.

```
$ mkdir -p ~/codedrop/
```

- Download the following **test Docker image** and save it to the *~/codedrop/* directory: 30-alpine-nickolashkraus-staticbox-latest.tar

```
$ curl https://github.com/nexB/scancode.io-tutorial/releases/download/sample-images/30-
↪alpine-nickolashkraus-staticbox-latest.tar --output ~/codedrop/30-alpine-
↪nickolashkraus-staticbox-latest.tar
```

- Create an alias to the `scanpipe` command executed through the `docker compose` command line interface with:

```
$ alias scanpipe="docker compose -f ${PWD}/docker-compose.yml run --volume ~/codedrop/:/
↪codedrop:ro web scanpipe"
```

- Create a new project named `staticbox`:

```
$ scanpipe create-project staticbox
```

```
>> Project staticbox created with work directory /var/scancodeio/workspace/projects/
↪staticbox-d4ed9405
```

Note: New projects work directory are created inside the location defined in *SCANCODEIO_WORKSPACE_LOCATION* setting. Default to the */var/scancodeio/workspace/* directory.

- Add the test Docker image tarball to the project workspace's *input/* directory:

```
$ scanpipe add-input --project staticbox \
    --input-file /codedrop/30-alpine-nickolashkraus-staticbox-latest.tar
```

```
>> File copied to the project inputs directory:
   - 30-alpine-nickolashkraus-staticbox-latest.tar
```

Note: The command output will let you know that the Docker image file was copied to the project's *input/* directory. Alternatively, you can copy files manually to the *input/* directory to include entire directories.

- Add the `analyze_docker_image` pipeline to your project:

```
$ scanpipe add-pipeline --project staticbox analyze_docker_image
```

```
>> Pipeline analyze_docker_image added to the project
```

- Check the status of the pipeline added to your project:

```
$ scanpipe show-pipeline --project staticbox
```

```
>> [NOT_STARTED] analyze_docker_image
```

Note: The `scanpipe show-pipeline` command lists all the pipelines added to the project and their execution status. You can use this to get a quick overview of the pipelines that have been already running, pipelines with **"SUCCESS"** or **"FAILURE"** status, and those will be running next, pipelines with **"NOT_STARTED"** status as shown below.

- Run the `analyze_docker_image` pipeline on this project. In the output, you will be shown the pipeline's execution progress:

```
$ scanpipe execute --project staticbox
```

```
>> Pipeline analyze_docker_image run in progress...
   Pipeline [analyze_docker_image] starting
   Step [extract_images] starting
   Step [extract_images] completed in 0.18 seconds
   Step [extract_layers] starting
   [...]
   Pipeline completed
   analyze_docker_image successfully executed on project staticbox
```

- Executing the `show-pipeline` command again will also confirm the success of the pipeline execution - **"[SUCCESS] analyze_docker_image"** status:

```
$ scanpipe show-pipeline --project staticbox
```

```
>> [SUCCESS] analyze_docker_image
```

- Get the results of the pipeline execution as a JSON file using the `output` command:

```
$ scanpipe output --project staticbox --format json --print > staticbox_results.json
```

- Finally, open the `staticbox_results.json` file in your preferred text editor/file viewer.

Note: To understand the output of the pipeline execution, see our *Output Files* section for details.

Tip: The `inputs` and `pipelines` can be provided directly at once when calling the `create-project` command. The `--execute` option is also available to start the pipeline execution right after the project creation. For example, the following command will create a project named `staticbox2`, download the test Docker image to the project's *input/* directory, add the `analyze_docker_image` pipeline, and execute the pipeline in one operation:

```
$ scanpipe create-project staticbox2 \
    --input-url https://github.com/nexB/scancode.io-tutorial/releases/download/sample-
↪images/30-alpine-nickolashkraus-staticbox-latest.tar \
    --pipeline analyze_docker_image \
    --execute
```

CHAPTER

TEN

ANALYZE CODEBASE (COMMAND LINE)

The focus of this tutorial is to guide you through scanning a codebase package using ScanCode.io.

Note: This tutorial assumes you have a recent version of ScanCode.io installed locally on your machine and **running with Docker**. If you do not have it installed, see our *Installation* guide for instructions.

10.1 Requirements

Before you follow the instructions in this tutorial, you need to:

- Install **ScanCode.io** locally
- Have **Shell access** on the machine where ScanCode.io is installed

10.2 Instructions

- Create a new directory in your home directory that will be used to put the input code to be scanned.

```
$ mkdir -p ~/codedrop/
```

- Download the following **package archive** and save it to the *~/codedrop/* directory: asgiref-3.3.0-py3-none-any.whl

```
$ curl https://files.pythonhosted.org/packages/c0/e8/
↪578887011652048c2d273bf98839a11020891917f3aa638a0bc9ac04d653/asgiref-3.3.0-py3-none-
↪any.whl --output ~/codedrop/asgiref-3.3.0-py3-none-any.whl
```

- Create an alias to the `scanpipe` command executed through the `docker compose` command line interface with:

```
$ alias scanpipe="docker compose -f ${PWD}/docker-compose.yml run --volume ~/codedrop/:/
↪codedrop:ro web scanpipe"
```

- Create a new project named `asgiref`:

```
$ scanpipe create-project asgiref
```

```
>> Project asgiref created with work directory /var/scancodeio/workspace/projects/
↪asgiref-35519104
```

- Add the package archive to the project workspace's *input/* directory:

```
$ scanpipe add-input --project asgiref --input-file /codedrop/asgiref-3.3.0-py3-none-any.
↪whl
```

```
>> File copied to the project inputs directory:
  - asgiref-3.3.0-py3-none-any.whl
```

- Add the `scan_codebase` pipeline to your project:

```
$ scanpipe add-pipeline --project asgiref scan_codebase
```

```
>> Pipeline scan_codebase added to the project
```

Note: The content of the *input/* directory will be copied in the *codebase/* directory where `extractcode` will be executed before running `scancode`. Alternatively, the codebase content can be manually copied to the *codebase/* directory in which case the `--input` option can be omitted.

- Run the `scan_codebase` pipeline on your project. The pipeline execution progress is shown within the following command's output:

```
$ scanpipe execute --project asgiref
```

```
>> Pipeline scan_codebase run in progress..
   Pipeline [scan_codebase] starting
   Step [copy_inputs_to_codebase_directory] starting
   Step [copy_inputs_to_codebase_directory] completed in 0.00 seconds
   Step [extract_archives] starting
   [...]
   Pipeline completed
   scan_codebase successfully executed on project asgiref
```

- Finally, export the scan results as JSON format:

```
$ scanpipe output --project asgiref --format json --print > asgiref-3.3.0_results.
↪json
```

Tip: The `inputs` and `pipelines` can be provided at the same time when calling the `create-project` command. For instance, the following command will create a new project named `asgiref2`, add the package archive as the project input, add the `scan_codebase` pipeline to the project, and execute it:

```
$ scanpipe create-project asgiref2 \
    --input-file /codedrop/asgiref-3.3.0-py3-none-any.whl \
    --pipeline scan_codebase \
    --execute
```

```
>> Project asgiref2 created with work directory /var/scancodeio/workspace/projects/
↪asgiref2-bea7a5e9
   File copied to the project inputs directory:
   - asgiref-3.3.0-py3-none-any.whl
```

(continues on next page)

(continued from previous page)

```
Start the scan_codebase pipeline execution...
[...]
Pipeline completed
scan_codebase successfully executed on project asgiref2
```

CHAPTER

ELEVEN

ANALYSE PACKAGE ARCHIVE (REST API)

This tutorial complements the *REST API* section, and the aim here is to show the API features while analyzing a package archive.

Tip: As a pre-requisite, check our *REST API* chapter for more details on REST API and how to get started.

Instructions:

- First, let's create a new project called `boolean.py-3.8`.
- We'll be using this package as the project input.
- We can add and execute the scan_single_package pipeline on our new project.

Note: Whether you follow this tutorial and previous instructions using cURL or Python script, the final results should be the same.

11.1 Using cURL

- In your terminal, insert the following:

```
api_url="http://localhost/api/projects/"
content_type="Content-Type: application/json"
data='{
    "name": "boolean.py-3.8",
    "input_urls": "https://github.com/bastikr/boolean.py/archive/refs/tags/v3.8.zip",
    "pipeline": "scan_single_package",
    "execute_now": true
}'

curl -X POST "$api_url" -H "$content_type" -d "$data"
```

Note: You have to set the api_url to http://localhost:8001/api/projects/ if you run on a local development setup.

Tip: You can provide the data using a json file with the text below, which will be passed in the -d parameter of the curl request:

```json
{
    "name": "boolean.py-3.8",
    "input_urls": "https://github.com/bastikr/boolean.py/archive/refs/tags/v3.8.zip",
    "pipeline": "scan_single_package",
    "execute_now": true
}
```

While in the same directory as your JSON file, here called `boolean.py-3.8_cURL.json`, create your new project with the following curl request:

```
curl -X POST "http://localhost/api/projects/" -H "Content-Type: application/json" -d␣
↪@boolean.py-3.8_cURL.json
```

If the new project has been successfully created, the response should include the project's details URL value among the returned data.

```json
{
    "name": "boolean.py-3.8",
    "url": "http://localhost/api/projects/11de938f-fb86-4178-870c-99f4952b8881/",
    "[...]": "[...]"
}
```

If you click on the project url, you'll be directed to the new project's instance page that allows you to perform extra actions on the project including deleting it.

Note: Refer to our *REST API* section for more information about these extra actions.

11.2 Using Python script

Tip: To interact with REST APIs, we will be turning to the requests library.

- To follow the above instructions and create a new project, start up the Python interpreter by typing `python` in your terminal.
- If you are seeing the prompt >>>, you can execute the following commands:

```python
import requests

api_url = "http://localhost/api/projects/"
data = {
    "name": "boolean.py-3.8",
    "input_urls": "https://github.com/bastikr/boolean.py/archive/refs/tags/v3.8.zip",
    "pipeline": "scan_single_package",
    "execute_now": True,
}
response = requests.post(api_url, data=data)
response.json()
```

The JSON response includes a generated UUID for the new project.

```
# print(response.json())
{
    "name": "boolean.py-3.8",
    "url": "http://localhost/api/projects/11de938f-fb86-4178-870c-99f4952b8881/",
    "[...]": "[...]",
}
```

Note: Alternatively, you can create a Python script with the above commands/text. Then, navigate to the same directory as your Python file and run the script to create your new project. However, no response will be shown on the terminal, and to access a given project details, you need to visit the projects' API endpoint.

Tip: You can check the *REST API* section for more details on how to view and download your scan results.

CHAPTER

TWELVE

LICENSE POLICIES AND COMPLIANCE ALERTS

In this tutorial, we'll introduce ScanCode.io's **license policies** and **compliance alerts** system and use the **results of a pipeline run** to demonstrate an example of the license policies and compliance alerts output.

As already mentioned, ScanCode.io automates the process of **Software Composition Analysis "SCA"** to identify existing open source components and their license compliance data in an application's codebase.

ScanCode.io also gives users the ability to define a set of **license policies** to have their projects checked against with a **compliance system**.

12.1 Creating Policies Files

A valid policies file is required to **enable compliance-related features**.

The policies file, by default `policies.yml`, is a **YAML file** with a structure similar to the following:

```yaml
license_policies:
-   license_key: mit
    label: Approved License
    compliance_alert: ''
-   license_key: mpl-2.0
    label: Restricted License
    compliance_alert: warning
-   license_key: gpl-3.0
    label: Prohibited License
    compliance_alert: error
```

- In the above policies file, licenses are referenced by the `license_key`, such as mit and gpl-3.0, which represents the ScanCode license key to match against detected licenses in the scan results.
- A policy is defined with a `label` and a `compliance_alert`. The labels can be customized to your prefered wording.
- The `compliance_alert` accepts 3 values:
  - `''` (empty string)
  - `warning`
  - `error`

12.2 Policies File Location

By default, ScanCode.io will look for a `policies.yml` file at the root of its codebase.

Alternatively, you can configure the location of policies files using the dedicated *SCANCODEIO_POLICIES_FILE* setting in your `.env` file.

Tip: Check out our *Application Settings* section for a comprehensive list of settings including policies file setting.

12.3 How Does The Compliance Alert Work?

The compliance system works by following a `Precedence of Policies` principal allowing the highest precedence policy to be applied in case of resources or packages with complex license expressions:

- **error > warning > missing > '' (empty string)**

This principal means a given resource with `error AND warning AND ''` license expression would have an overall compliance alert of `error`.

Warning: The `missing` compliance alert value is applied for licenses not present in the policies file.

12.4 Example Output

Create a `policies.yml` file in the root directory of your ScanCode.io codebase, with the following content:

```
license_policies:
-   license_key: mit
    label: Approved License
    compliance_alert: ''
-   license_key: gpl-3.0
    label: Prohibited License
    compliance_alert: error
```

Run the following command to create a project and run the `scan_codebase` pipeline:

```
$ scanpipe create-project cuckoo-filter-with-policies \
    --input-url https://files.pythonhosted.org/packages/75/fc/
↪f5b2e466d763dcc381d5127b73ffc265e8cdaf39ddafa422b7896e625432/cuckoo_filter-1.0.6.tar.
↪gz \
    --pipeline scan_codebase \
    --execute
```

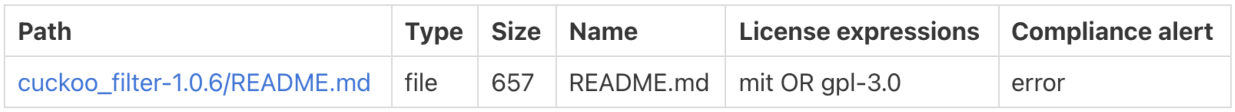
Generate results:

```
$ scanpipe output --project cuckoo-filter-with-policies
```

The computed compliance alerts are now included in the results, available for each detected licenses, and computed at the codebase resource level, for example:

```
{
  "for_packages": [],
  "compliance_alert": "error",
  "path": "cuckoo_filter-1.0.6.tar.gz-extract/cuckoo_filter-1.0.6/README.md",
  "licenses": [
    {
      "key": "mit",
      "name": "MIT License",
      "policy": {
        "label": "Recommended License",
        "compliance_alert": ""
      },
    }
    {
      "key": "gpl-3.0",
      "name": "GNU General Public License 3.0",
      "policy": {
        "label": "Prohibited License",
        "compliance_alert": "error"
    },
  ],
  "license_expressions": [
    "mit OR gpl-3.0",
  ],
  "status": "scanned",
  "name": "README",
  "[...]": "[...]"
}
```

The compliance alert are also displayed in the Web UI:



Path	Type	Size	Name	License expressions	Compliance alert
cuckoo_filter-1.0.6/README.md	file	657	README.md	mit OR gpl-3.0	error

CHAPTER

THIRTEEN

FIND VULNERABILITIES (WEB UI)

This tutorial aims to show you how to integrate VulnerableCode with ScanCode.io and how to discover vulnerable packages using the `find_vulnerabilities` pipeline.

Note: This tutorial assumes that you have a working installation of ScanCode.io. If you don't, please refer to the *Installation* page.

13.1 Configure VulnerableCode integration

Warning: The `find_vulnerabilities` pipeline requires access to a VulnerableCode database.

You have the option to either deploy your instance of VulnerableCode or connect to the public instance.

To configure your local environment, set the `VULNERABLECODE_URL` in your `.env` file:

```
VULNERABLECODE_URL=https://public.vulnerablecode.io/
```

Restarting the services is required following any changes to .env:

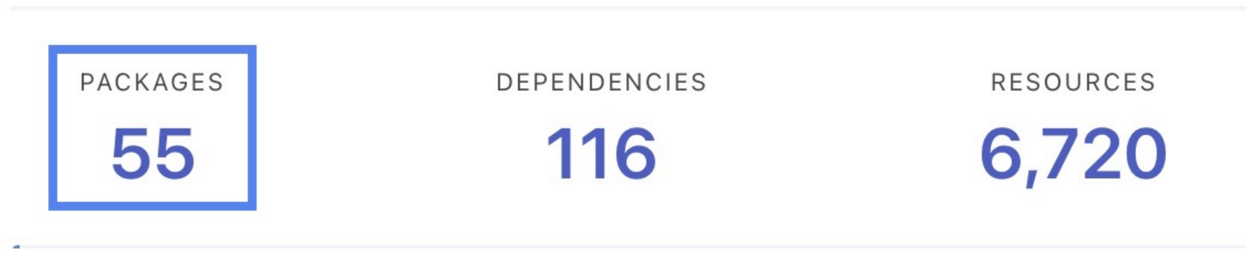
```
docker compose restart web worker
```

13.2 Run the `find_vulnerabilities` pipeline

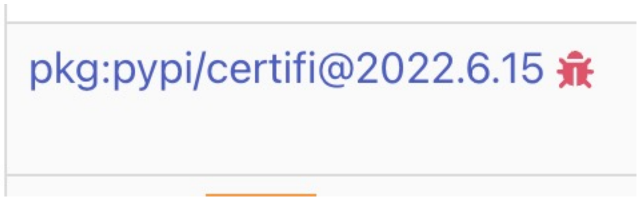
Open any of your existing projects containing a few detected packages.

Note: If you do not have any projects available, please start with this tutorial: *Analyze Docker Image (Web UI)*

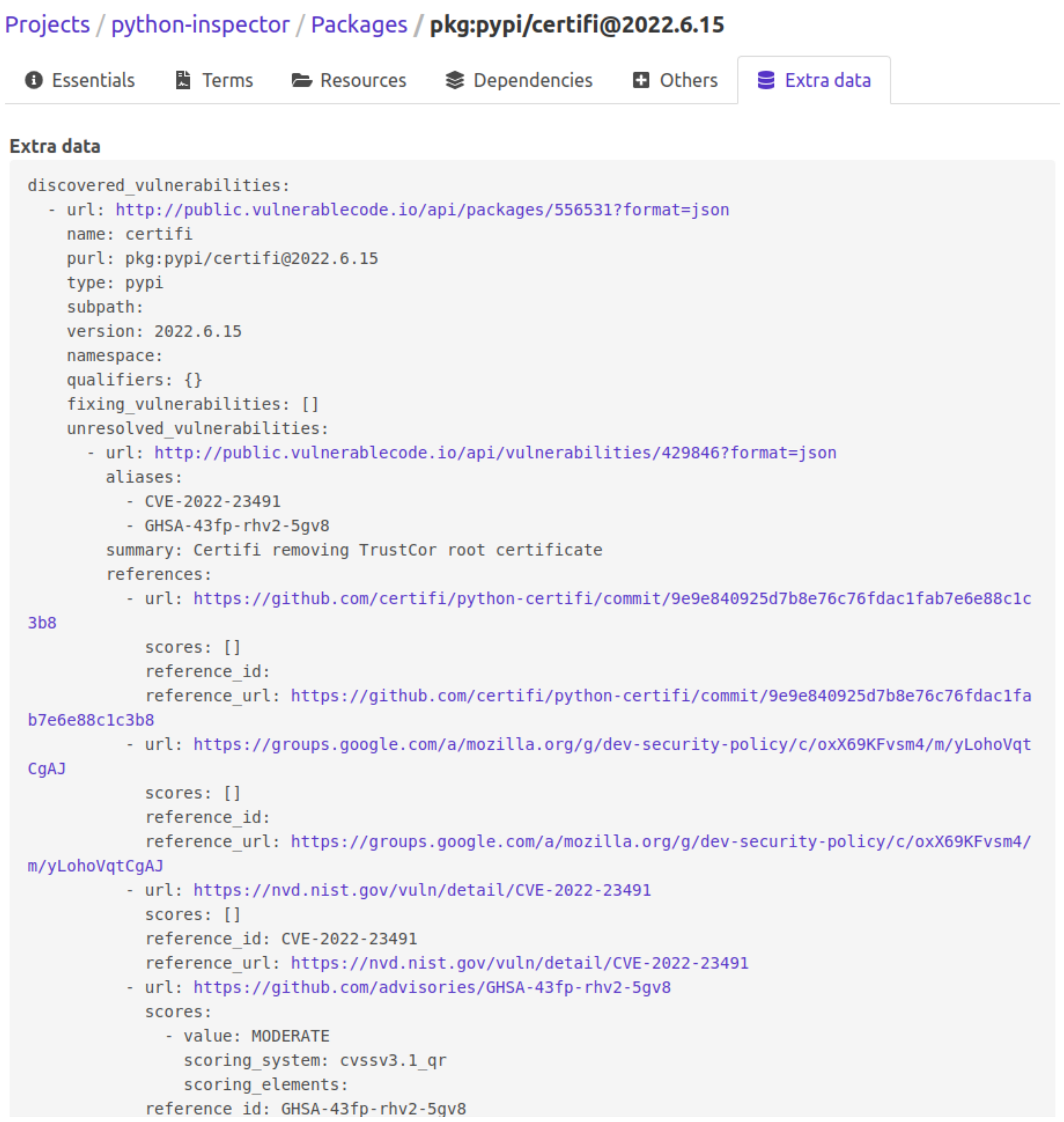
- Click on the **"Add pipeline"** button and select the **"find_vulnerabilities"** pipeline from the dropdown list. Check the **"Execute pipeline now"** option and validate with the **"Add pipeline"** button.
- Once the pipeline run completes with success, you can reach the **Packages** list view by clicking the count number under the **"PACKAGES"** header:



- A red bug icon is displayed next to all packages for which declared vulnerabilities were found:



- Click red bug icon to reach the vulnerability details for this package:


Projects / python-inspector / Packages / pkg:pypi/certifi@2022.6.15
Essentials
Terms
Resources
Dependencies
Others
Extra data

Extra data

```
discovered_vulnerabilities:
  - url: http://public.vulnerablecode.io/api/packages/556531?format=json
    name: certifi
    purl: pkg:pypi/certifi@2022.6.15
    type: pypi
    subpath:
    version: 2022.6.15
    namespace:
    qualifiers: {}
    fixing_vulnerabilities: []
    unresolved_vulnerabilities:
      - url: http://public.vulnerablecode.io/api/vulnerabilities/429846?format=json
        aliases:
          - CVE-2022-23491
          - GHSA-43fp-rhv2-5gv8
        summary: Certifi removing TrustCor root certificate
        references:
          - url: https://github.com/certifi/python-certifi/commit/9e9e840925d7b8e76c76fdac1fab7e6e88c1c
3b8
            scores: []
            reference_id:
            reference_url: https://github.com/certifi/python-certifi/commit/9e9e840925d7b8e76c76fdac1fa
b7e6e88c1c3b8
          - url: https://groups.google.com/a/mozilla.org/g/dev-security-policy/c/oxX69KFvsm4/m/yLohoVqt
CgAJ
            scores: []
            reference_id:
            reference_url: https://groups.google.com/a/mozilla.org/g/dev-security-policy/c/oxX69KFvsm4/
m/yLohoVqtCgAJ
          - url: https://nvd.nist.gov/vuln/detail/CVE-2022-23491
            scores: []
            reference_id: CVE-2022-23491
            reference_url: https://nvd.nist.gov/vuln/detail/CVE-2022-23491
          - url: https://github.com/advisories/GHSA-43fp-rhv2-5gv8
            scores:
              - value: MODERATE
                scoring_system: cvssv3.1_qr
                scoring_elements:
            reference_id: GHSA-43fp-rhv2-5gv8
```

CHAPTER

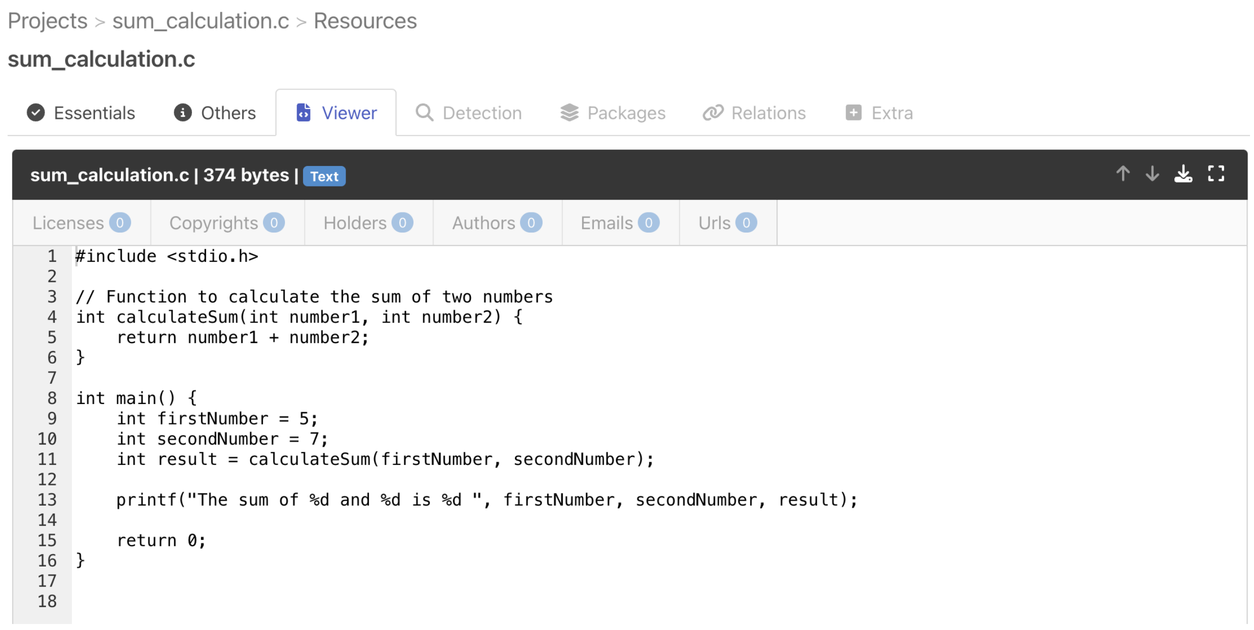
FOURTEEN

SYMBOL AND STRING COLLECTION (WEB UI)

In this tutorial we'll introduce the different addon pipeline that can be used for collecting symbols and strings from codebase resources.

Note: This tutorial assumes that you have a working installation of ScanCode.io. If you don't, please refer to the *Installation* page.

Throughout this tutorial, we will use this resource for symbol and string collection.



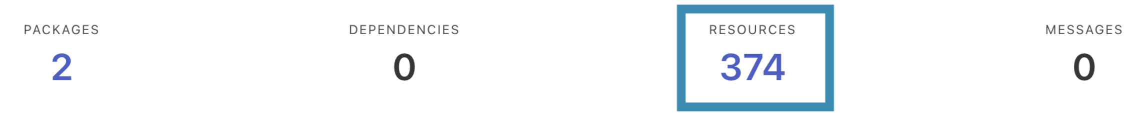
```
#include <stdio.h>

// Function to calculate the sum of two numbers
int calculateSum(int number1, int number2) {
    return number1 + number2;
}

int main() {
    int firstNumber = 5;
    int secondNumber = 7;
    int result = calculateSum(firstNumber, secondNumber);

    printf("The sum of %d and %d is %d ", firstNumber, secondNumber, result);

    return 0;
}
```

14.1 Ctags Symbols

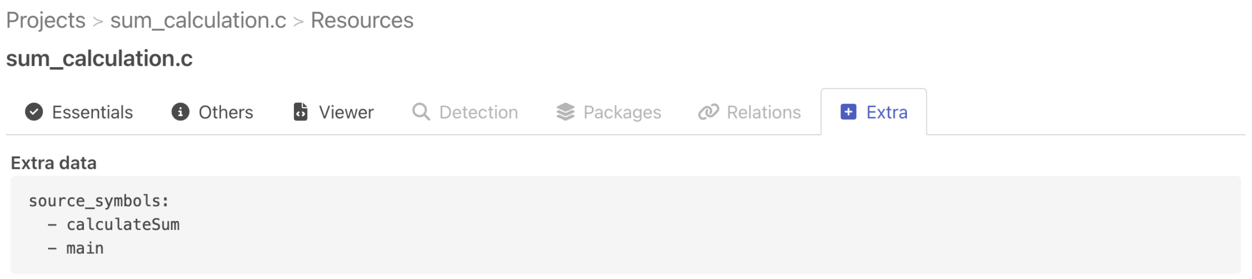
- Open any existing projects containing a few resources.
- Click on the **"Add pipeline"** button and select the **"collect_symbols"** pipeline from the dropdown list. Check the **"Execute pipeline now"** option and validate with the **"Add pipeline"** button.

Warning: The `collect_symbols` pipeline requires `universal-ctags`, please refer to the *System Dependencies*.

- Once the pipeline run completes with success, you can reach the **Resources** list view by clicking the count number under the **"RESOURCES"** header:



- Click on any code file and go to **Extra** tab, to get the resource symbols.



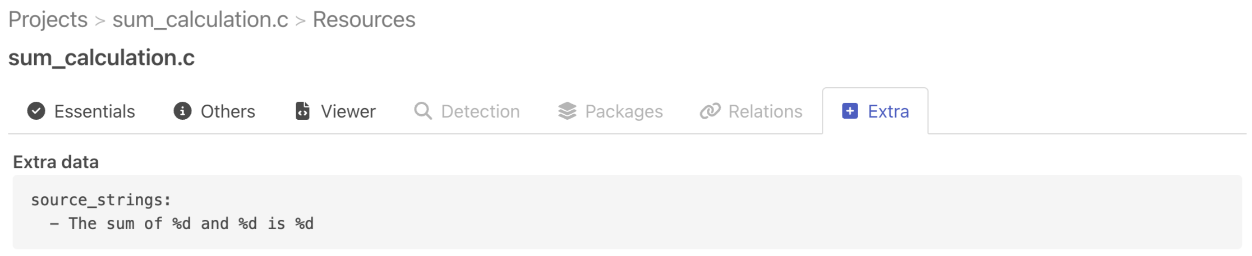
```
source_symbols:
  - calculateSum
  - main
```

14.2 Xgettext Strings

- Open any existing projects containing a few resources.
- Click on the **"Add pipeline"** button and select the **"collect_source_strings"** pipeline from the dropdown list. Check the **"Execute pipeline now"** option and validate with the **"Add pipeline"** button.

Warning: The `collect_source_strings` pipeline requires `gettext`, please refer to the *System Dependencies*.

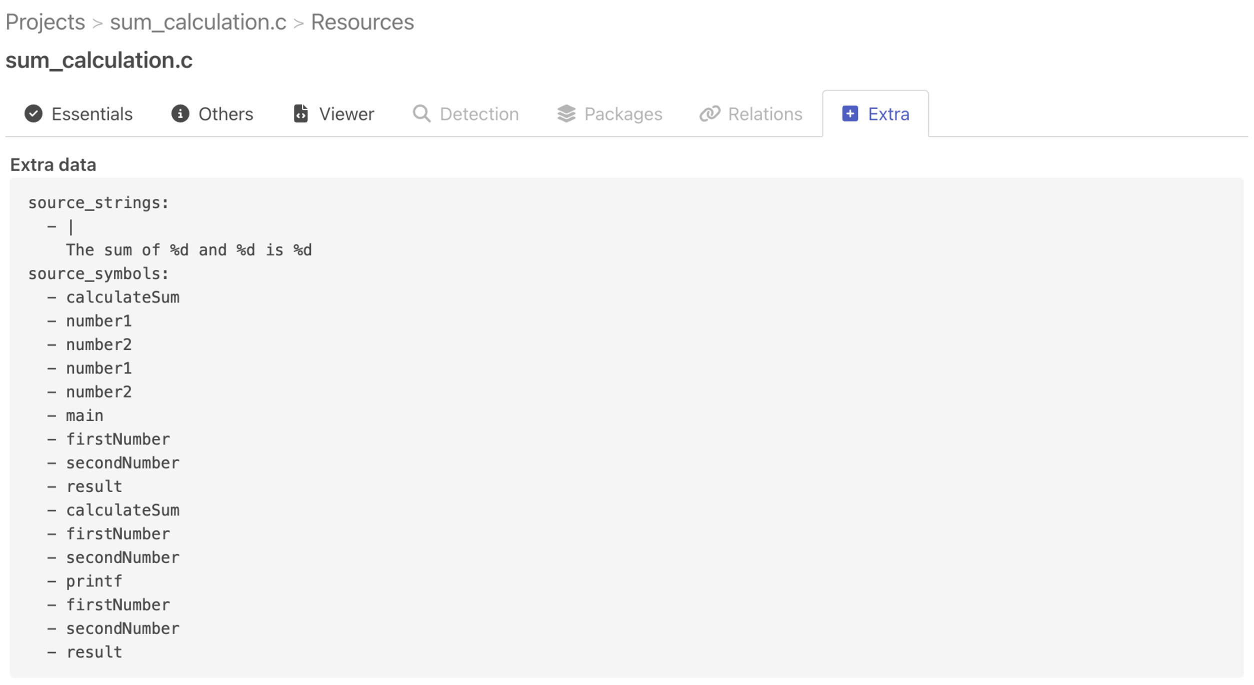
- Once the pipeline run completes with success, you can reach the **Resources** list view by clicking the count number under the **"RESOURCES"** header:
- Click on any code file and go to **Extra** tab, to get the resource strings.



```
source_strings:
  - The sum of %d and %d is %d
```

14.3 Tree-Sitter Symbols and Strings

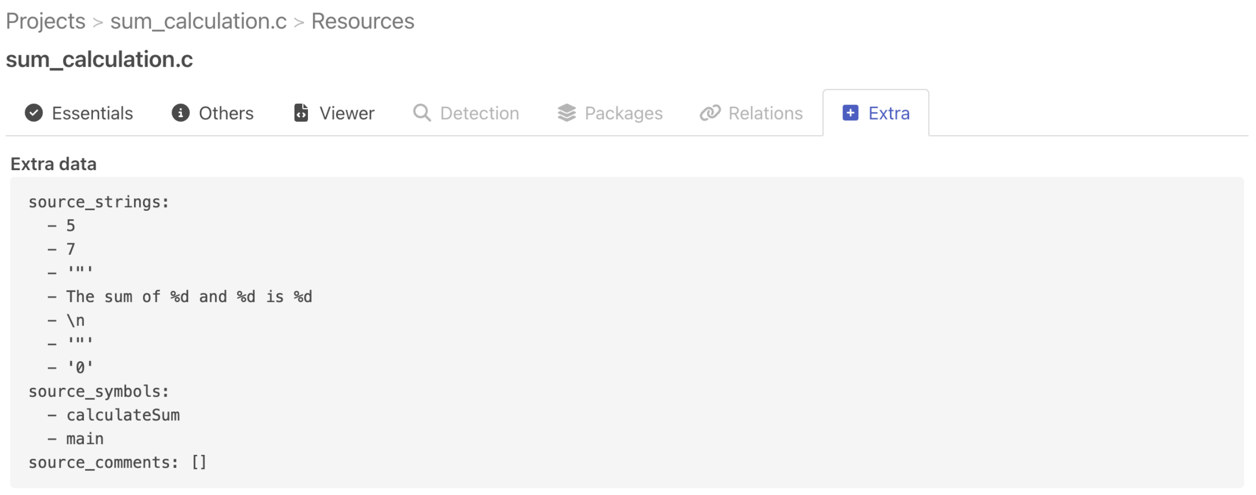
- Open any existing projects containing a few resources.
- Click on the **"Add pipeline"** button and select the **"collect_tree_sitter_symbols"** pipeline from the dropdown list. Check the **"Execute pipeline now"** option and validate with the **"Add pipeline"** button.
- Once the pipeline run completes with success, you can reach the **Resources** list view by clicking the count number under the **"RESOURCES"** header:
- Click on any code file and go to **Extra** tab, to get the resource symbols and strings.



```
source_strings:
  - |
    The sum of %d and %d is %d
source_symbols:
  - calculateSum
  - number1
  - number2
  - number1
  - number2
  - main
  - firstNumber
  - secondNumber
  - result
  - calculateSum
  - firstNumber
  - secondNumber
  - printf
  - firstNumber
  - secondNumber
  - result
```

14.4 Pygments Symbols and Strings

- Open any existing projects containing a few resources.
- Click on the **"Add pipeline"** button and select the **"collect_pygments_symbols"** pipeline from the dropdown list. Check the **"Execute pipeline now"** option and validate with the **"Add pipeline"** button.
- Once the pipeline run completes with success, you can reach the **Resources** list view by clicking the count number under the **"RESOURCES"** header:
- Click on any code file and go to **Extra** tab, to get the resource symbols and strings.



```
source_strings:
  - 5
  - 7
  - '"'
  - The sum of %d and %d is %d
  - \n
  - '"'
  - '0'
source_symbols:
  - calculateSum
  - main
source_comments: []
```

CHAPTER

FIFTEEN

SCANPIPE CONCEPTS

15.1 Project

A **project** encapsulates the analysis of software code:

- It has a *Project workspace*, which is a directory that contains the software code files under analysis.
- It makes use of one or more **code analysis** *Pipelines* scripts to automate the code analysis process.
- It tracks *Codebase Resources*, i.e. its **code files and directories**
- It tracks *Discovered Packages*, i.e. **system and application packages** origin and license discovered in the codebase.

In the database, **a project is identified by its unique name**.

Note: Multiple analysis pipelines can be run on a single project.

15.2 Project workspace

A project workspace is the root directory where **a project's files are stored**.

The following directories exist under the workspace directory:

- *input/* contains all uploaded files used as the input of a project, such as a codebase archive.
- *codebase/* contains files and directories - i.e. resources - tracked as CodebaseResource records in the database.
- *output/* contains any output files created by the pipelines, including reports, scan results, etc.
- *tmp/* is a scratch pad for temporary files generated during pipelines runs.

15.3 Pipelines

A pipeline is a Python script that contains a series of steps, which are executed sequentially to **perform a code analysis**.

It usually starts with the uploaded input files, which might need to be extracted first. Then, it generates `CodebaseResource` records in the database accordingly.

Those resources can then be **analyzed, scanned, and matched** as needed. Analysis results and reports are eventually posted at the end of a pipeline run.

All *Built-in Pipelines* are located in the `scanpipe.pipelines` module. Each pipeline consists of a Python script and includes one subclass of the `Pipeline` class. Each step is a method of the `Pipeline` class. The execution order of the steps - or the sequence of steps execution - is declared through the `steps` class attribute.

Tip: Refer to *Custom Pipelines* for details about adding custom pipelines to ScanCode.io.

Note: You can assign one or more pipelines to a project as a sequence.

15.4 Pipes

As mentioned above, pipelines include a group of operations—Pipes—that are combined in a chain-like fashion and executed in orderly manner. Pipes are simply the building blocks of a given pipeline.

For example, the following operations—Steps—are included in the RootFS pipeline, and they are leveraging pipes to accomplish pre-defined tasks:

```
from scanpipe.pipelines import Pipeline
from scanpipe.pipes import flag
from scanpipe.pipes import rootfs
from scanpipe.pipes import scancode

class RootFS(Pipeline):
    [...]

    def flag_empty_files(self):
        """
        Flags empty files.
        """
        flag.flag_empty_files(self.project)

    def scan_for_application_packages(self):
        """
        Scans unknown resources for packages information.
        """
        scancode.scan_for_application_packages(self.project)
```

Note: All **built-in pipes** are located in the `scanpipe.pipes` module. Pipes are grouped by type in modules, e.g. `codebase`, `input`, `output`, `scancode`.

Refer to our *Pipes* section for information about available pipes and their usage.

15.5 Codebase Resources

A project `Codebase Resources` are records of its **code files and directories**. `CodebaseResource` is a database model and each record is identified by its path under the project workspace.

The following are some of the `CodebaseResource` attributes:

- A **status**, which is used to track the analysis status for this resource.
- A **type**, such as a file, a directory or a symlink
- Various attributes to track detected **copyrights**, **license expressions**, **copyright holders**, and **related packages**.

Note: Please note that ScanCode-toolkit use the same attributes and attribute names for files.

15.6 Discovered Packages

A project `Discovered Packages` are records of the **system and application packages** discovered in the code under analysis. `DiscoveredPackage` is a database model and each record is identified by its `Package URL`. `Package URL` is a fundamental effort to create informative identifiers for software packages, such as Debian, RPM, npm, Maven, or PyPI packages. See https://github.com/package-url for more details.

The following are some of the `DiscoveredPackage` attributes:

- A type, name, version (all Package URL attributes)
- A homepage_url, download_url, and other URLs
- Checksums, such as SHA1, MD5
- Copyright, license_expression, and declared_license

Note: Please note that ScanCode-toolkit use the same attributes and attribute names for packages.

CHAPTER SIXTEEN

BUILT-IN PIPELINES

Pipelines in ScanCode.io are Python scripts that facilitate code analysis by executing a sequence of steps. The platform provides the following built-in pipelines:

Tip: If you are unsure which pipeline suits your requirements best, check out the *Which pipeline should I use?* section for guidance.

16.1 Pipeline Base Class

class `scanpipe.pipelines.`**`Pipeline`**

Main class for all pipelines including common step methods.

`flag_empty_files``()`

Flag empty files.

`flag_ignored_resources``()`

Flag ignored resources based on Project `ignored_patterns` setting.

`extract_archives``()`

Extract archives located in the codebase/ directory with extractcode.

16.2 Analyse Docker Image

class `scanpipe.pipelines.docker.`**`Docker`**

Analyze Docker images.

`extract_images``()`

Extract images from input tarballs.

`extract_layers``()`

Extract layers from input images.

`find_images_os_and_distro``()`

Find the operating system and distro of input images.

`collect_images_information``()`

Collect and store image information in a project.

collect_and_create_codebase_resources()

Collect and labels all image files as CodebaseResources.

collect_and_create_system_packages()

Collect installed system packages for each layer based on the distro.

flag_uninteresting_codebase_resources()

Flag files that don't belong to any system package.

16.3 Analyze Root Filesystem or VM Image

class scanpipe.pipelines.root_filesystem.**RootFS**

Analyze a Linux root filesystem, also known as rootfs.

extract_input_files_to_codebase_directory()

Extract root filesystem input archives with extractcode.

find_root_filesystems()

Find root filesystems in the project's codebase/.

collect_rootfs_information()

Collect and stores rootfs information on the project.

collect_and_create_codebase_resources()

Collect and label all image files as CodebaseResource.

collect_and_create_system_packages()

Collect installed system packages for each rootfs based on the distro. The collection of system packages is only available for known distros.

flag_uninteresting_codebase_resources()

Flag files—not worth tracking—that don't belong to any system packages.

scan_for_application_packages()

Scan unknown resources for packages information.

match_not_analyzed_to_system_packages()

Match files with "not-yet-analyzed" status to files already belonging to system packages.

match_not_analyzed_to_application_packages()

Match files with "not-yet-analyzed" status to files already belonging to application packages.

scan_for_files()

Scan unknown resources for copyrights, licenses, emails, and urls.

analyze_scanned_files()

Analyze single file scan results for completeness.

flag_not_analyzed_codebase_resources()

Check for any leftover files for sanity; there should be none.

16.4 Analyse Docker Windows Image

class `scanpipe.pipelines.docker_windows.`**`DockerWindows`**

Analyze Windows Docker images.

`flag_known_software_packages``()`

Flag files from known software packages by checking common install paths.

`flag_uninteresting_codebase_resources``()`

Flag files that are known/labelled as uninteresting.

`flag_program_files_dirs_as_packages``()`

Report the immediate subdirectories of `Program Files` and `Program Files (x86)` as packages.

`flag_data_files_with_no_clues``()`

Flag data files that have no clues on their origin as uninteresting.

16.5 Collect Pygments Source Symbols (addon)

class `scanpipe.pipelines.collect_pygments_symbols.`**`CollectPygmentsSymbolsAndStrings`**

Collect codebase symbols using pygments and keep them in extra data field.

Also collect strings and comments.

`collect_and_store_pygments_symbols_and_strings``()`

Collect symbols, strings and comments from codebase files using pygments and store them in the extra data field.

16.6 Collect Source Strings (addon)

class `scanpipe.pipelines.collect_source_strings.`**`CollectSourceStrings`**

Collect source strings from codebase files and keep them in extra data field.

`collect_and_store_resource_strings``()`

Collect source strings from codebase files using gettext and store them in the extra data field.

16.7 Collect Codebase Symbols (addon)

class `scanpipe.pipelines.collect_symbols.`**`CollectSymbols`**

Collect symbols from codebase files and keep them in extra data field.

`collect_and_store_resource_symbols``()`

Collect symbols from codebase files using Ctags and store them in the extra data field.

16.8 Collect Tree-Sitter Source Symbols (addon)

`class scanpipe.pipelines.collect_tree_sitter_symbols.`**`CollectTreeSitterSymbolsAndStrings`**

Collect codebase symbols using tree-sitter and keep them in extra data field.

Also collect strings.

`collect_and_store_tree_sitter_symbols_and_strings``()`

Collect symbols and strings from codebase files using tree-sitter and store them in the extra data field.

16.9 Find Vulnerabilities (addon)

Warning: This pipeline requires access to a VulnerableCode database. Refer to *VULNERABLECODE* to configure access to VulnerableCode in your ScanCode.io instance.

`class scanpipe.pipelines.find_vulnerabilities.`**`FindVulnerabilities`**

Find vulnerabilities for packages and dependencies in the VulnerableCode database.

Vulnerability data is stored on each package and dependency instance.

`check_vulnerablecode_service_availability``()`

Check if the VulnerableCode service if configured and available.

`lookup_packages_vulnerabilities``()`

Check for vulnerabilities for each of the project's discovered package.

`lookup_dependencies_vulnerabilities``()`

Check for vulnerabilities for each of the project's discovered dependency.

16.10 Inspect ELF Binaries (addon)

`class scanpipe.pipelines.inspect_elf_binaries.`**`InspectELFBinaries`**

Inspect ELF binaries and collect DWARF paths.

`collect_dwarf_source_path_references``()`

Collect DWARF paths from ELF files and set values on the extra_data field.

16.11 Inspect Packages

`class scanpipe.pipelines.inspect_packages.`**`InspectPackages`**

Inspect a codebase for packages and pre-resolved dependencies.

This pipeline inspects a codebase for application packages and their dependencies using package manifests and dependency lockfiles. It does not resolve dependencies, it does instead collect already pre-resolved dependencies from lockfiles, and direct dependencies (possibly not resolved) as found in package manifests' dependency sections.

See documentation for the list of supported package manifests and dependency lockfiles: https://scancode-toolkit.readthedocs.io/en/stable/reference/available_package_parsers.html

scan_for_application_packages()

Scan resources for package information to add DiscoveredPackage and DiscoveredDependency objects from detected package data.

16.12 Load Inventory

class scanpipe.pipelines.load_inventory.**LoadInventory**

Load JSON/XLSX inventory files generated with ScanCode-toolkit or ScanCode.io.

Supported format are ScanCode-toolkit JSON scan results, ScanCode.io JSON output, and ScanCode.io XLSX output.

An inventory is composed of packages, dependencies, resources, and relations.

get_inputs()

Locate all the supported input files from the project's input/ directory.

build_inventory_from_scans()

Process JSON scan results files to populate packages, dependencies, and resources.

16.13 Load SBOM

class scanpipe.pipelines.load_sbom.**LoadSBOM**

Load package data from one or more SBOMs.

Supported SBOMs: - SPDX document - CycloneDX BOM Other formats: - AboutCode .ABOUT files for package curations.

get_sbom_inputs()

Locate all the SBOMs among the codebase resources.

get_packages_from_sboms()

Get packages data from SBOMs.

create_packages_from_sboms()

Create the packages and dependencies from the SBOM, in the database.

16.14 Resolve Dependencies

class scanpipe.pipelines.resolve_dependencies.**ResolveDependencies**

Resolve dependencies from package manifests and lockfiles.

This pipeline collects lockfiles and manifest files that contain dependency requirements, and resolves these to a concrete set of package versions.

Supports resolving packages for: - Python: using python-inspector, using requirements.txt and setup.py manifests as inputs

get_manifest_inputs()

Locate package manifest files with a supported package resolver.

get_packages_from_manifest()

Resolve package data from lockfiles/requirement files with package requirements/dependenices.

create_resolved_packages()

Create the resolved packages and their dependencies in the database.

16.15 Map Deploy To Develop

Warning: This pipeline requires input files to be tagged with the following:

- "from": For files related to the source code (also known as "develop").
- "to": For files related to the build/binaries (also known as "deploy").

Tagging your input files varies based on whether you are using the REST API, UI, or CLI. Refer to the *How to tag input files?* section for guidance.

class scanpipe.pipelines.deploy_to_develop.**DeployToDevelop**

Establish relationships between two code trees: deployment and development.

This pipeline requires a minimum of two archive files, each properly tagged with:

- **from** for archives containing the development source code.
- **to** for archives containing the deployment compiled code.

When using download URLs as inputs, the "from" and "to" tags can be provided by adding a "#from" or "#to" fragment at the end of the download URLs.

When uploading local files:

- **User Interface:** Use the "Edit flag" link in the "Inputs" panel of the Project details view.
- **REST API:** Utilize the "upload_file_tag" field in addition to the "upload_file".
- **Command Line Interface:** Tag uploaded files using the "filename:tag" syntax, for example, `--input-file path/filename:tag`.

get_inputs()

Locate the `from` and `to` input files.

extract_inputs_to_codebase_directory()

Extract input files to the project's codebase/ directory.

collect_and_create_codebase_resources()

Collect and create codebase resources.

fingerprint_codebase_directories()

Compute directory fingerprints for matching

flag_whitespace_files()

Flag whitespace files with size less than or equal to 100 byte as ignored.

map_about_files()

Map `from/` .ABOUT files to their related `to/` resources.

`map_checksum()`
: Map using SHA1 checksum.

`match_archives_to_purldb()`
: Match selected package archives by extension to PurlDB.

`find_java_packages()`
: Find the java package of the .java source files.

`map_java_to_class()`
: Map a .class compiled file to its .java source.

`map_jar_to_source()`
: Map .jar files to their related source directory.

`map_javascript()`
: Map a packed or minified JavaScript, TypeScript, CSS and SCSS to its source.

`map_elf()`
: Map ELF binaries to their sources.

`map_go()`
: Map Go binaries to their sources.

`match_directories_to_purldb()`
: Match selected directories in PurlDB.

`match_resources_to_purldb()`
: Match selected files by extension in PurlDB.

`map_javascript_post_purldb_match()`
: Map minified javascript file based on existing PurlDB match.

`map_javascript_path()`
: Map javascript file based on path.

`map_javascript_colocation()`
: Map JavaScript files based on neighborhood file mapping.

`map_thirdparty_npm_packages()`
: Map thirdparty package using package.json metadata.

`map_path()`
: Map using path similarities.

`flag_mapped_resources_archives_and_ignored_directories()`
: Flag all codebase resources that were mapped during the pipeline.

`perform_house_keeping_tasks()`

On deployed side

- **PurlDB match files with `no-java-source` and empty status,**
  if no match is found update status to `requires-review`.
- Update status for uninteresting files.
- Flag the dangling legal files for review.

On devel side

- Update status for not deployed files.

match_purldb_resources_post_process()

Choose the best package for PurlDB matched resources.

remove_packages_without_resources()

Remove packages without any resources.

scan_unmapped_to_files()

Scan unmapped/matched `to/` files for copyrights, licenses, emails, and urls and update the status to *requires-review*.

scan_mapped_from_for_files()

Scan mapped `from/` files for copyrights, licenses, emails, and urls.

create_local_files_packages()

Create local-files packages for codebase resources not part of a package.

flag_deployed_from_resources_with_missing_license()

Update the status for deployed from files with missing license.

16.16 Match to MatchCode (addon)

Warning: This pipeline requires access to a MatchCode.io service. Refer to *MATCHCODE.IO* to configure access to MatchCode.io in your ScanCode.io instance.

`class scanpipe.pipelines.match_to_matchcode.`**`MatchToMatchCode`**

Match the codebase resources of a project against MatchCode.io to identify packages.

This process involves:

1. Generating a JSON scan of the project codebase
2. Transmitting it to MatchCode.io and awaiting match results
3. Creating discovered packages from the package data obtained
4. Associating the codebase resources with those discovered packages

Currently, MatchCode.io can only match for archives, directories, and files from Maven and npm Packages.

This pipeline requires a MatchCode.io instance to be configured and available. There is currently no public instance of MatchCode.io. Reach out to nexB, Inc. for other arrangements.

check_matchcode_service_availability()

Check if the MatchCode.io service if configured and available.

send_project_json_to_matchcode()

Create a JSON scan of the project Codebase and send it to MatchCode.io.

poll_matching_results()

Wait until the match results are ready by polling the match run status.

create_packages_from_match_results()

Create DiscoveredPackages from match results.

16.17 Populate PurlDB (addon)

Warning: This pipeline requires access to a PurlDB service. Refer to *PURLDB* to configure access to PurlDB in your ScanCode.io instance.

class scanpipe.pipelines.populate_purldb.**PopulatePurlDB**

Populate PurlDB with discovered project packages and their dependencies.

populate_purldb_with_discovered_packages()

Add DiscoveredPackage to PurlDB.

populate_purldb_with_discovered_dependencies()

Add DiscoveredDependency to PurlDB.

16.18 Scan Codebase

class scanpipe.pipelines.scan_codebase.**ScanCodebase**

Scan a codebase for application packages, licenses, and copyrights.

This pipeline does not further scan the files contained in a package for license and copyrights and only considers the declared license of a package. It does not scan for system (Linux distro) packages.

copy_inputs_to_codebase_directory()

Copy input files to the project's codebase/ directory. The code can also be copied there prior to running the Pipeline.

collect_and_create_codebase_resources()

Collect and create codebase resources.

scan_for_application_packages()

Scan unknown resources for packages information.

scan_for_files()

Scan unknown resources for copyrights, licenses, emails, and urls.

16.19 Scan Single Package

class scanpipe.pipelines.scan_single_package.**ScanSinglePackage**

Scan a single package archive (or package manifest file).

This pipeline scans a single package for package metadata, declared dependencies, licenses, license clarity score and copyrights.

The output is a summary of the scan results in JSON format.

get_package_input()

Locate the package input in the project's input/ directory.

collect_input_information()

Collect and store information about the project input.

extract_input_to_codebase_directory()
Copy or extract input to project codebase/ directory.

run_scan()
Scan extracted codebase/ content.

load_inventory_from_toolkit_scan()
Process a JSON Scan results to populate codebase resources and packages.

make_summary_from_scan_results()
Build a summary in JSON format from the generated scan results.

CHAPTER

SEVENTEEN

CUSTOM PIPELINES

Pipelines are Python scripts; each contains a set of instructions that have to be executed in an orderly manner—pipe-like nature—to perform a code analysis.

- A pipeline is a **Python class** that lives in a Python module as a `.py` **file**.
- A pipeline class **always inherits** from the `Pipeline` base class *Pipeline Base Class*, or from other existing pipeline classes, such as the *Built-in Pipelines*.
- A pipeline **defines sequence of steps**—execution order of the steps—using the `steps` classmethod.

See *Pipelines* for more details.

17.1 Pipeline registration

Built-in pipelines are located in *scanpipe/pipelines/* directory and are registered during the ScanCode.io installation.

Whereas custom pipelines are added as Python files `.py` in the directories defined in the *SCANCODEIO_PIPELINES_DIRS* setting. Custom pipelines are registered at runtime.

17.2 Create a Pipeline

Create a new Python file `my_pipeline.py`, and make sure to include the full path of the new pipeline directory in the *SCANCODEIO_PIPELINES_DIRS* setting.

```python
from scanpipe.pipelines import Pipeline

class MyPipeline(Pipeline):

    @classmethod
    def steps(cls):
        return (
            cls.step1,
            cls.step2,
        )

    def step1(self):
        pass

    def step2(self):
        pass
```

Tip: You can view the *scanpipe/pipelines/* directory for more pipeline examples.

17.3 Modify Existing Pipelines

Existing pipelines are flexible and can be reused as a base for custom pipelines , i.e. be customized. For instance, you can override existing steps, add new ones, or remove any of them.

```python
from scanpipe.pipelines.scan_codebase import ScanCodebase

class MyCustomScan(ScanCodebase):

    @classmethod
    def steps(cls):
        return (
            # Original steps from the ScanCodebase pipeline
            cls.copy_inputs_to_codebase_directory,
            cls.extract_archives,
            cls.collect_and_create_codebase_resources,
            cls.flag_empty_files,
            cls.flag_ignored_resources,
            cls.scan_for_application_packages,
            cls.scan_for_files,

            # My extra steps
            cls.extra_step1,
            cls.extra_step2,
        )

    def extra_step1(self):
        pass

    def extra_step2(self):
        pass
```

17.4 Custom Pipeline Example

The example below shows a custom pipeline that is based on the built-in *Scan Codebase* pipeline with an extra reporting step.

Add the following code snippet to a Python file and register the path of the file's directory in the *SCANCODEIO_PIPELINES_DIRS*.

```python
from collections import defaultdict

from jinja2 import Template

from scanpipe.pipelines.scan_codebase import ScanCodebase

```

(continues on next page)

(continued from previous page)

```python
class ScanAndReport(ScanCodebase):
    """
    Runs the ScanCodebase built-in pipeline steps and generate a licenses report.
    """

    @classmethod
    def steps(cls):
        return ScanCodebase.steps() + (
            cls.report_licenses_with_resources,
        )

    # Set to True to extract recursively nested archives in archives.
    extract_recursively = False

    # See https://jinja.palletsprojects.com/en/3.0.x/templates/ for documentation
    report_template = """
    {% for matched_text, paths in resources.items() -%}
        {{ matched_text }}

        {% for path in paths -%}
            {{ path }}
        {% endfor %}

    {% endfor %}
    """

    def report_licenses_with_resources(self):
        """
        Retrieves codebase resources and generates a licenses report file using
        a Jinja template.
        """
        resources = self.project.codebaseresources.has_license_detections()

        resources_by_matched_text = defaultdict(list)
        for resource in resources:
            for detection_data in resource.license_detections:
                for match in detection_data.get("matches", []):
                    matched_text = match.get("matched_text")
                    resources_by_matched_text[matched_text].append(resource.path)

        template = Template(self.report_template, lstrip_blocks=True, trim_blocks=True)
        report_stream = template.stream(resources=resources_by_matched_text)
        report_file = self.project.get_output_file_path("license-report", "txt")
        report_stream.dump(str(report_file))
```

17.5 Pipeline Packaging

Once you created a custom pipeline, you'll want to package it as a Python module for easier distribution and reuse. You can check the Packaging Python Project tutorial at PyPA, for standard packaging instructions.

After you have packaged your own custom pipeline successfully, you need to specify the entry point of the pipeline in the *setup.cfg* file.

```
[options.entry_points]
scancodeio_pipelines =
    pipeline_name = pipeline_module:Pipeline_class
```

Note: Remember to replace `pipeline_module` with the name of the Python module containing your custom pipeline.

17.6 Pipeline Packaging Example

The example below shows a standard pipeline packaging procedure for the custom pipeline created in *Custom Pipeline Example*.

A typical directory structure for the Python package would be:

```
.
├── CHANGELOG.rst
├── LICENSE
├── MANIFEST.in
├── pyproject.toml
├── README.rst
├── setup.cfg
├── setup.py
└── src
    └── scancodeio_scan_and_report_pipeline
        ├── __init__.py
        └── pipelines
            ├── __init__.py
            └── scan_and_report.py
```

Add the following code snippet to your *setup.cfg* file and specify the entry point to the pipeline under the `[options.entry_points]` section.

```
[metadata]
license_files =
    LICENSE
    CHANGELOG.rst

name = scancodeio_scan_and_report_pipeline
author = nexB. Inc. and others
author_email = info@aboutcode.org
license = Apache-2.0

# description must be on ONE line https://github.com/pypa/setuptools/issues/1390
```

(continues on next page)

(continued from previous page)

```
description =  Generates a licenses report file from a template in ScanCode.io
long_description = file:README.rst
url = https://github.com/nexB/scancode.io
classifiers =
    Development Status :: 4 - Beta
    Intended Audience :: Developers
    Programming Language :: Python :: 3
    Programming Language :: Python :: 3 :: Only
keywords =
    scancodeio
    pipelines

[options]
package_dir=
    =src
packages=find:
include_package_data = true
zip_safe = false
python_requires = >=3.10
setup_requires = setuptools_scm[toml] >= 4

[options.packages.find]
where=src

[options.entry_points]
scancodeio_pipelines =
    pipeline_name = scancodeio_scan_and_report_pipeline.pipelines.scan_and_
↪report:ScanAndReport
```

Tip: Take a look at Google License Classifier pipeline for ScanCode.io for a complete example on packaging a custom tool as a pipeline.

17.7 Pipeline Publishing to PyPI

After successfully packaging a pipeline, you may consider distributing it—as a plugin—via PyPI. Ensure a directory structure similar to the *Pipeline Packaging Example* with all package files correctly configured.

Tip: See the Python packaging tutorial at PyPA for a detailed setup guide.

Next step involves generating the distribution archives for the package. Make sure you have the latest version of `build` installed on your system.

```
pip install --upgrade build
```

Now run the following command from within the same directory where the `pyproject.toml` is located:

```
python -m build
```

Once completed, you should have two files inside the *dist/* directory with the `.tar.gz` and `.whl` extensions.

Note: Remember to create an account on PyPI before uploading your distribution archive to PyPI.

You can use `twine` to upload the package to PyPI. To install twine, run the following command:

```
pip install twine
```

Finally, you can upload your package to PyPI with the next command:

```
twine upload dist/*
```

Once successfully uploaded, your pipeline package should be viewable on PyPI under the name specified in your manifest.

To make your pipeline available in your instance of ScanCode.io, you need to install the package from PyPI. For example, to install the package described in the *Pipeline Packaging Example*, run:

```
bin/pip install scancodeio_scan_and_report_pipeline
```

CHAPTER

EIGHTEEN

PIPES

18.1 Generic

`scanpipe.pipes.`**`make_codebase_resource`**`(`*project*, *location*, *save=True*, ***extra_fields*`)`

Create a CodebaseResource instance in the database for the given `project`.

The provided `location` is the absolute path of this resource. It must be rooted in *project.codebase_path* as only the relative path within the project codebase/ directory is stored in the database.

Extra fields can be provided as keywords arguments to this function call:

```
make_codebase_resource(
    project=project,
    location=resource.location,
    rootfs_path=resource.path,
    tag=layer_tag,
)
```

In this example, `rootfs_path` is an optional path relative to a rootfs root within an Image/VM filesystem context. e.g.: "/var/log/file.log"

All paths use the POSIX separators.

If a CodebaseResource already exists in the `project` with the same path, the error raised on save() is not stored in the database and the creation is skipped.

`scanpipe.pipes.`**`get_resource_codebase_root`**`(`*project*, *resource_path*`)`

Return "to" or "from" depending on the resource location in the codebase.

`scanpipe.pipes.`**`yield_resources_from_codebase`**`(`*project*`)`

Yield CodebaseResource instances, including their `info` data, ready to be inserted in the database using `save()` or `bulk_create()`.

`scanpipe.pipes.`**`collect_and_create_codebase_resources`**`(`*project*, *batch_size=5000*`)`

Collect and create codebase resources including the "to/" and "from/" context using the resource tag field.

The default `batch_size` can be overriden, although the benefits of a value greater than 5000 objects are usually not significant.

`scanpipe.pipes.`**`update_or_create_resource`**`(`*project*, *resource_data*`)`

Get, update or create a CodebaseResource then return it.

`scanpipe.pipes.`**`update_or_create_package`**(*project*, *package_data*, *codebase_resources=None*)

Get, update or create a DiscoveredPackage then return it. Use the *project* and *package_data* mapping to lookup and creates the DiscoveredPackage using its Package URL and package_uid as a unique key. The package can be associated to *codebase_resources* providing a list or queryset of resources.

`scanpipe.pipes.`**`create_local_files_package`**(*project*, *defaults*, *codebase_resources=None*)

Create a local-files package using provided `defaults` data.

`scanpipe.pipes.`**`update_or_create_dependency`**(*project*, *dependency_data*, *for_package=None*, *datafile_resource=None*, *datasource_id=None*, *strip_datafile_path_root=False*)

Get, update or create a DiscoveredDependency then returns it. Use the *project* and *dependency_data* mapping to lookup and creates the DiscoveredDependency using its dependency_uid and for_package_uid as a unique key.

If *strip_datafile_path_root* is True, then *DiscoveredDependency.create_from_data()* will strip the root path segment from the *datafile_path* of *dependency_data* before looking up the corresponding CodebaseResource for *datafile_path*. This is used in the case where Dependency data is imported from a scancode-toolkit scan, where the root path segments are not stripped for *datafile_path*.

`scanpipe.pipes.`**`get_or_create_relation`**(*project*, *relation_data*)

Get or create a CodebaseRelation then return it. The support for update is not useful as there is no fields on the model that could be updated.

`scanpipe.pipes.`**`normalize_path`**(*path*)

Return a normalized path from a *path* string.

`scanpipe.pipes.`**`strip_root`**(*location*)

Return the provided *location* without the root directory.

`scanpipe.pipes.`**`filename_now`**(*sep='-'*)

Return the current date and time in iso format suitable for filename.

`scanpipe.pipes.`**`count_group_by`**(*queryset*, *field_name*)

Return a summary of all existing values for the provided *field_name* on the *queryset*, including the count of each entry, as a dictionary.

`scanpipe.pipes.`**`get_bin_executable`**(*filename*)

Return the location of the *filename* executable binary.

`class` `scanpipe.pipes.`**`LoopProgress`**(*total_iterations*, *logger*, *progress_step=10*)

A context manager for logging progress in loops.

Usage:

```
total_iterations = 100
logger = print  # Replace with your actual logger function

progress = LoopProgress(total_iterations, logger, progress_step=10)
for item in progress.iter(iterator):
    "Your processing logic here"

with LoopProgress(total_iterations, logger, progress_step=10) as progress:
    for item in progress.iter(iterator):
        "Your processing logic here"
```

`__init__`(*total_iterations*, *logger*, *progress_step=10*) `scanpipe.pipes.`**`get_text_str_diff_ratio`**`(`*str_a*, *str_b*`)`

Return a similarity ratio as a float between 0 and 1 by comparing the text content of the `str_a` and `str_b`.

Return None if any of the two resources str is empty.

`scanpipe.pipes.`**`get_resource_diff_ratio`**`(`*resource_a*, *resource_b*`)`

Return a similarity ratio as a float between 0 and 1 by comparing the text content of the CodebaseResource `resource_a` and `resource_b`.

Return None if any of the two resources are not readable as text.

`scanpipe.pipes.`**`poll_until_success`**`(`*check*, *sleep=10*, ***kwargs*`)`

Given a function *check*, which returns the status of a run, return True when the run instance has completed successfully.

Return False when the run instance has failed, stopped, or gone stale.

The arguments for *check* need to be provided as keyword argument into this function.

18.2 Codebase

`scanpipe.pipes.codebase.`**`get_resource_fields`**`(`*resource*, *fields*`)`

Return a mapping of fields from *fields* and values from *resource*

`scanpipe.pipes.codebase.`**`get_resource_tree`**`(`*resource*, *fields*, *codebase=None*, *seen_resources={}*`)`

Return a tree as a dictionary structure starting from the provided *resource*.

The following classes are supported for the input *resource* object:

- scanpipe.models.CodebaseResource
- commoncode.resource.Resource

The data included for each child is controlled with the *fields* argument.

The *codebase* is only required in the context of a commoncode *Resource* input.

seen_resources is used when get_resource_tree() is used in the context of get_codebase_tree(). We keep track of child Resources we visit in *seen_resources*, so we don't visit them again in get_codebase_tree().

`scanpipe.pipes.codebase.`**`get_codebase_tree`**`(`*codebase*, *fields*`)`

Return a tree as a dictionary structure starting from the root resources of the provided *codebase*.

The following classes are supported for the input *codebase* object:

- scanpipe.pipes.codebase.ProjectCodebase
- commoncode.resource.Codebase
- commoncode.resource.VirtualCodebase

The data included for each child is controlled with the *fields* argument.

`scanpipe.pipes.codebase.`**`get_basic_virtual_codebase`**`(`*resources_qs*`)`

Return a VirtualCodebase created from CodebaseResources in *resources_qs*.

The only Resource fields that are populated are path, sha1, size, and is_file. This is intended for use with scanpipe.pipes.matchcode.fingerprint_codebase_directories

`class` `scanpipe.pipes.codebase.`**`ProjectCodebase`**`(`*project*`)`

Represents the codebase of a project stored in the database. A Codebase is a tree of Resources.

`__init__`(*project*)

18.3 Compliance

`scanpipe.pipes.compliance.`**`flag_compliance_files`**(*project*)

Flag compliance files status for the provided *project*.

`scanpipe.pipes.compliance.`**`analyze_compliance_licenses`**(*project*)

Scan compliance licenses status for the provided *project*.

18.4 CycloneDX

`scanpipe.pipes.cyclonedx.`**`resolve_license`**(*license*)

Return license expression/id/name from license item.

`scanpipe.pipes.cyclonedx.`**`get_declared_licenses`**(*licenses*)

Return resolved license from list of LicenseChoice.

`scanpipe.pipes.cyclonedx.`**`get_checksums`**(*component*)

Return dict of all the checksums from a component.

`scanpipe.pipes.cyclonedx.`**`get_external_references`**(*component*)

Return dict of reference urls from list of *component.external_references*.

`scanpipe.pipes.cyclonedx.`**`get_properties_data`**(*component*)

Return the properties as dict, extracted from *component.properties*.

`scanpipe.pipes.cyclonedx.`**`validate_document`**(*document*)

Check the validity of this CycloneDX document.

The validator is loaded from the document specVersion property.

`scanpipe.pipes.cyclonedx.`**`is_cyclonedx_bom`**(*input_location*)

Return True if the file at *input_location* is a CycloneDX BOM.

`scanpipe.pipes.cyclonedx.`**`cyclonedx_component_to_package_data`**(*cdx_component*)

Return package_data from CycloneDX component.

`scanpipe.pipes.cyclonedx.`**`get_components`**(*bom*)

Return list of components from CycloneDX BOM.

`scanpipe.pipes.cyclonedx.`**`delete_tools`**(*cyclonedx_document_json*)

Remove the `tools` section, if defined, from the SBOM as it can be in the way of loading a SBOM that is valid regarding the spec, but fails the deserialization.

The `metadata.tools` as an array was deprecated in 1.5 and replaced by an object structure where you can define a list of `components` and `services`.

The new structure is not yet supported by the cyclonedx-python-lib, neither for serialization (output) nor deserialization (input). https://github.com/CycloneDX/cyclonedx-python-lib/issues/578

The tools are not used anyway in the context of loading the SBOM component data as packages.

`scanpipe.pipes.cyclonedx.`**`delete_empty_dict_property`**(*cyclonedx_document_json*)

Remove dict entry where keys are defined but no values are set, such as `{"name": ""}`.

Class like cyclonedx.model.contact.OrganizationalEntity raise a NoPropertiesProvidedException while it is not enforced in the spec.

See https://github.com/CycloneDX/cyclonedx-python-lib/issues/600

`scanpipe.pipes.cyclonedx.`**`resolve_cyclonedx_packages`**(*input_location*)

Resolve the packages from the *input_location* CycloneDX document file.

18.5 Deploy to develop

`scanpipe.pipes.d2d.`**`get_inputs`**(*project*)

Locate the `from` and `to` input files in project inputs/ directory. The input source can be flagged using a "from-" / "to-" prefix in the filename or by adding a "#from" / "#to" fragment at the end of the download URL.

`scanpipe.pipes.d2d.`**`get_extracted_path`**(*resource*)

Return the `-extract/` extracted path of provided `resource`.

`scanpipe.pipes.d2d.`**`get_extracted_subpath`**(*path*)

Return the path segments located after the last `-extract/` segment.

`scanpipe.pipes.d2d.`**`get_best_path_matches`**(*to_resource*, *matches*)

Return the best `matches` for the provided `to_resource`.

`scanpipe.pipes.d2d.`**`get_from_files_for_scanning`**(*resources*)

Return resources in the "from/" side which has been mapped to the "to/" side, but are not mapped using ABOUT files.

`scanpipe.pipes.d2d.`**`map_checksum`**(*project*, *checksum_field*, *logger=None*)

Map using checksum.

`scanpipe.pipes.d2d.`**`map_java_to_class`**(*project*, *logger=None*)

Map to/ compiled Java .class(es) to from/ .java source using Java fully qualified paths and indexing from/ .java files.

`scanpipe.pipes.d2d.`**`get_indexable_qualified_java_paths_from_values`**(*resource_values*)

Yield tuples of (resource id, fully-qualified Java path) for indexable classes from a list of `resource_data` tuples of "from/" side of the project codebase.

These `resource_data` input tuples are in the form:
(resource.id, resource.name, resource.extra_data)

And the output tuples look like this example::
(123, "org/apache/commons/LoggerImpl.java")

`scanpipe.pipes.d2d.`**`get_indexable_qualified_java_paths`**(*from_resources_dot_java*)

Yield tuples of (resource id, fully-qualified Java class name) for indexable classes from the "from/" side of the project codebase using the "java_package" Resource.extra_data.

`scanpipe.pipes.d2d.`**`find_java_packages`**(*project*, *logger=None*)

Collect the Java packages of Java source files for a `project`.

Multiprocessing is enabled by default on this pipe, the number of processes can be controlled through the SCANCODEIO_PROCESSES setting.

Note: we use the same API as the ScanCode scans by design scanpipe.pipes.d2d.**scan_for_java_package**(*location*, *with_threading=True*)

Run a Java package scan on provided `location`.

Return a dict of scan `results` and a list of `errors`.

scanpipe.pipes.d2d.**save_java_package_scan_results**(*codebase_resource*, *scan_results*, *scan_errors*)

Save the resource Java package scan results in the database as Resource.extra_data. Create project errors if any occurred during the scan.

scanpipe.pipes.d2d.**map_jar_to_source**(*project*, *logger=None*)

Map .jar files to their related source directory.

scanpipe.pipes.d2d.**map_path**(*project*, *logger=None*)

Map using path suffix similarities.

scanpipe.pipes.d2d.**get_project_resources_qs**(*project*, *resources*)

Return a queryset of CodebaseResources from *project* containing the CodebaseResources from *resources* . If a CodebaseResource in *resources* is an archive or directory, then their descendants are also included in the queryset.

Return None if *resources* is empty or None.

scanpipe.pipes.d2d.**create_package_from_purldb_data**(*project*, *resources*, *package_data*, *status*)

Create a DiscoveredPackage instance from PurlDB `package_data`.

Return a tuple, containing the created DiscoveredPackage and the number of CodebaseResources matched to PurlDB that are part of that DiscoveredPackage.

scanpipe.pipes.d2d.**match_purldb_package**(*project*, *resources_by_sha1*, *enhance_package_data=True*, ***kwargs*)

Given a mapping of lists of CodebaseResources by their sha1 values, *resources_by_sha1*, send those sha1 values to purldb packages API endpoint, process the matched Package data, then return the number of CodebaseResources that were matched to a Package.

scanpipe.pipes.d2d.**match_purldb_resource**(*project*, *resources_by_sha1*, *package_data_by_purldb_urls=None*, ***kwargs*)

Given a mapping of lists of CodebaseResources by their sha1 values, *resources_by_sha1*, send those sha1 values to purldb resources API endpoint, process the matched Package data, then return the number of CodebaseResources that were matched to a Package.

package_data_by_purldb_urls is a mapping of package data by their purldb package instance URLs. This is intended to be used as a cache, to avoid retrieving package data we retrieved before.

scanpipe.pipes.d2d.**match_purldb_directory**(*project*, *resource*)

Match a single directory resource in the PurlDB.

scanpipe.pipes.d2d.**match_sha1s_to_purldb**(*project*, *resources_by_sha1*, *matcher_func*, *package_data_by_purldb_urls*)

Process *resources_by_sha1* with *matcher_func* and return a 3-tuple contaning an empty defaultdict(list), the number of matches and the number of sha1s sent to purldb.

scanpipe.pipes.d2d.**match_purldb_resources**(*project*, *extensions*, *matcher_func*, *chunk_size=1000*, *logger=None*)

Match against PurlDB selecting codebase resources using provided `package_extensions` for archive type files, and `resource_extensions`.

Match requests are sent off in batches of 1000 SHA1s. This number is set using *chunk_size*.

`scanpipe.pipes.d2d.`**`match_purldb_directories`**`(`*project*, *logger=None*`)`

Match against PurlDB selecting codebase directories.

`scanpipe.pipes.d2d.`**`map_javascript`**`(`*project*, *logger=None*`)`

Map a packed or minified JavaScript, TypeScript, CSS and SCSS to its source.

`class scanpipe.pipes.d2d.`**`AboutFileIndexes`**`(`*regex_by_about_path: dict*, *ignore_regex_by_about_path: dict*, *about_resources_by_path: dict*, *about_pkgdata_by_path: dict*, *mapped_resources_by_aboutpath: dict*`)`

About file indexes are used to create packages from About files and map the resources described in them to the respective packages created, using regex path patterns and other About file data.

`classmethod` **`create_indexes`**`(`*project*, *from_about_files*, *logger=None*`)`

Return an ABOUT file index, containing path pattern mappings, package data, and resources, created from *from_about_files*, the About file resources.

`get_matched_about_path``(`*to_resource*`)`

Map *to_resource* using the about file index, and if mapped, return the path string to the About file it was mapped to, and if not mapped or ignored, return None.

`map_deployed_to_devel_using_about``(`*to_resources*`)`

Return mapped resources which are mapped using the path patterns in About file indexes. Resources are mapped for each About file in the index, and their status is updated accordingly.

`get_about_file_companions``(`*about_path*`)`

Given an `about_path` path string to an About file, get CodebaseResource objects for the companion license and notice files.

`create_about_packages_relations``(`*project*`)`

Create packages using About file package data, if the About file has mapped resources on the to/ codebase and creates the mappings for the package created and mapped resources.

`__init__``(`*regex_by_about_path: dict*, *ignore_regex_by_about_path: dict*, *about_resources_by_path: dict*, *about_pkgdata_by_path: dict*, *mapped_resources_by_aboutpath: dict*`)` → None

`scanpipe.pipes.d2d.`**`map_about_files`**`(`*project*, *logger=None*`)`

Map `from/` .ABOUT files to their related `to/` resources.

`scanpipe.pipes.d2d.`**`map_javascript_post_purldb_match`**`(`*project*, *logger=None*`)`

Map minified javascript file based on existing PurlDB match.

`scanpipe.pipes.d2d.`**`map_javascript_path`**`(`*project*, *logger=None*`)`

Map javascript file based on path.

`scanpipe.pipes.d2d.`**`map_javascript_colocation`**`(`*project*, *logger=None*`)`

Map JavaScript files based on neighborhood file mapping.

`scanpipe.pipes.d2d.`**`flag_processed_archives`**`(`*project*`)`

Flag package archives as processed if they meet the following criteria:

1. They have no assigned status.
2. They are identified as package archives.
3. All resources inside the corresponding archive '-extract' directory have an assigned status.

This function iterates through the package archives in the project and checks whether all resources within their associated '-extract' directory have statuses. If so, it updates the status of the package archive to "archive-processed".

`scanpipe.pipes.d2d.`**`map_thirdparty_npm_packages`**`(`*project*, *logger=None*`)`

Map thirdparty package using package.json metadata.

`scanpipe.pipes.d2d.`**`get_from_files_related_with_not_in_package_to_files`**`(`*project*`)`

Return from-side resource files that have one or more relations with to-side resources that are not part of a package. Only resources with a `detected_license_expression` value are returned.

`scanpipe.pipes.d2d.`**`create_local_files_packages`**`(`*project*`)`

Create local-files packages for codebase resources not part of a package.

Resources are grouped by license_expression within a local-files packages.

`scanpipe.pipes.d2d.`**`match_resources_with_no_java_source`**`(`*project*, *logger=None*`)`

Match resources with `no-java-source` to PurlDB, if no match is found update status to `requires-review`.

`scanpipe.pipes.d2d.`**`match_unmapped_resources`**`(`*project*, *matched_extensions=None*, *logger=None*`)`

Match resources with empty status to PurlDB, if unmatched update status as `requires-review`.

`scanpipe.pipes.d2d.`**`flag_undeployed_resources`**`(`*project*`)`

Update status for undeployed files.

`scanpipe.pipes.d2d.`**`scan_unmapped_to_files`**`(`*project*, *logger=None*`)`

Scan unmapped/matched `to/` files for copyrights, licenses, emails, and urls and update the status to *requires-review*.

`scanpipe.pipes.d2d.`**`flag_deployed_from_resources_with_missing_license`**`(`*project*, *doc_extensions=None*`)`

Update the status for deployed from files with missing license.

`scanpipe.pipes.d2d.`**`handle_dangling_deployed_legal_files`**`(`*project*, *logger*`)`

Scan the legal files with empty status and update status to *REVIEW_DANGLING_LEGAL_FILE*.

`scanpipe.pipes.d2d.`**`save_scan_legal_file_results`**`(`*codebase_resource*, *scan_results*, *scan_errors*`)`

Save the legal resource scan results with *REVIEW_DANGLING_LEGAL_FILE* status in the database. Create project errors if any occurred during the scan.

`scanpipe.pipes.d2d.`**`flag_whitespace_files`**`(`*project*`)`

Flag whitespace files with size less than or equal to 100 byte as ignored.

`scanpipe.pipes.d2d.`**`match_purldb_resources_post_process`**`(`*project*, *logger=None*`)`

Choose the best package for PurlDB matched resources.

`scanpipe.pipes.d2d.`**`map_paths_resource`**`(`*to_resource*, *from_resources*, *from_resources_index*, *map_types*, *logger=None*`)`

Map paths found in the `to_resource` extra_data to paths of the `from_resources` CodebaseResource queryset using the precomputed `from_resources_index` path index.

`scanpipe.pipes.d2d.`**`process_paths_in_binary`**`(`*to_resource*, *from_resources*, *from_resources_index*, *map_type*, *paths_in_binary*`)`

Process list of paths in binary and Yield either: - a tuple of (unique key for a relationship, `CodebaseRelation` object) - Or a path if it was not mapped

`scanpipe.pipes.d2d.`**`count_path_segments`**`(`*path*`)`

Return the number of path segments in POSIX `path` string

`scanpipe.pipes.d2d.`**`sort_matched_from_resources`**`(`*matched_from_resources*`)`

Return the sorted list of `matched_from_resources` based on path length and path.

`scanpipe.pipes.d2d.`**`is_invalid_match`**(*match*, *matched_path_length*)

Check if the match is invalid based on the `matched_path_length` and the number of resource IDs.

`scanpipe.pipes.d2d.`**`map_elfs`**(*project*, *logger=None*)

Map ELF binaries to their sources in `project`.

`scanpipe.pipes.d2d.`**`get_elf_file_dwarf_paths`**(*location*)

Retrieve dwarf paths for ELF files.

`scanpipe.pipes.d2d.`**`get_go_file_paths`**(*location*)

Retrieve Go file paths.

`scanpipe.pipes.d2d.`**`map_go_paths`**(*project*, *logger=None*)

Map Go binaries to their source in `project`.

18.6 Docker

`scanpipe.pipes.docker.`**`get_tarballs_from_inputs`**(*project*)

Return the tarballs from the *project* input/ work directory. Supported file extensions: *.tar*, *.tar.gz*, *.tgz*.

`scanpipe.pipes.docker.`**`extract_images_from_inputs`**(*project*)

Collect all the tarballs from the *project* input/ work directory, extracts each tarball to the tmp/ work directory and collects the images.

Return the *images* and an *errors* list of error messages that may have happened during the extraction.

`scanpipe.pipes.docker.`**`extract_image_from_tarball`**(*input_tarball*, *extract_target*, *verify=True*)

Extract images from an `input_tarball` to an `extract_target` directory Path object and collects the extracted images.

Return the *images* and an *errors* list of error messages that may have happened during the extraction.

`scanpipe.pipes.docker.`**`extract_layers_from_images`**(*project*, *images*)

Extract all layers from the provided *images* into the *project* codebase work directory.

Return an *errors* list of error messages that may occur during the extraction.

`scanpipe.pipes.docker.`**`extract_layers_from_images_to_base_path`**(*base_path*, *images*)

Extract all layers from the provided *images* into the *base_path* work directory.

Return an *errors* list of error messages that may occur during the extraction.

`scanpipe.pipes.docker.`**`get_image_data`**(*image*, *layer_path_segments=2*)

Return a mapping of image-related data given an *image*. Keep only `layer_path_segments` trailing layer location segments (or keep the locations unmodified if `layer_path_segments` is 0).

`scanpipe.pipes.docker.`**`get_layer_tag`**(*image_id*, *layer_id*, *layer_index*, *id_length=6*)

Return a "tag" crafted from the provided *image_id*, *layer_id*, and *layer_index*. The purpose of this tag is to be short, clear and sortable.

For instance, given an image with an id:
785df58b6b3e120f59bce6cd10169a0c58b8837b24f382e27593e2eea011a0d8

and two layers from bottom to top as:
0690c89adf3e8c306d4ced085fc16d1d104dcfddd6dc637e141fa78be242a707
7a1d89d2653e8e4aa9011fd95034a4857109d6636f2ad32df470a196e5dd1585

we would get these two tags:
img-785df5-layer-01-0690c8 img-785df5-layer-02-7a1d89

`scanpipe.pipes.docker.`**`create_codebase_resources`**(*project*, *image*)

Create the CodebaseResource for an *image* in a *project*.

`scanpipe.pipes.docker.`**`create_system_package`**(*project*, *purl*, *package*, *layer*, *layer_tag*)

Create system package and related resources.

`scanpipe.pipes.docker.`**`scan_image_for_system_packages`**(*project*, *image*)

Given a *project* and an *image* - this scans the *image* layer by layer for installed system packages and creates a DiscoveredPackage for each.

Then for each installed DiscoveredPackage file, check if it exists as a CodebaseResource. If exists, relate that CodebaseResource to its DiscoveredPackage; otherwise, keep that as a missing file.

`scanpipe.pipes.docker.`**`flag_whiteout_codebase_resources`**(*project*)

Tag overlayfs/AUFS whiteout special files CodebaseResource as "ignored-whiteout". See https://github.com/opencontainers/image-spec/blob/master/layer.md#whiteouts for details.

`class scanpipe.pipes.docker.`**`Layer`**(*layer_tag*, *created_by*, *layer_id*, *image_id*, *created*, *size*, *author*, *comment*, *archive_location*)

`archive_location`
Alias for field number 8

`author`
Alias for field number 6

`comment`
Alias for field number 7

`created`
Alias for field number 4

`created_by`
Alias for field number 1

`image_id`
Alias for field number 3

`layer_id`
Alias for field number 2

`layer_tag`
Alias for field number 0

`size`
Alias for field number 5

`scanpipe.pipes.docker.`**`get_layers_data`**(*project*)

Get list of structured layers data from project extra_data field.

18.7 ELF

scanpipe.pipes.elf.**collect_dwarf_source_path_references**(*resource*)

Collect and store the DWARF debug paths of the provided ELF `resource`.

18.8 Fetch

scanpipe.pipes.fetch.**run_command_safely**(*command_args*)

Execute the external commands following security best practices.

This function is using the subprocess.run function which simplifies running external commands. It provides a safer and more straightforward API compared to older methods like subprocess.Popen.

- This does not use the Shell (shell=False) to prevent injection vulnerabilities.
- The command should be provided as a list of `command_args` arguments.
- Only full paths to executable commands should be provided to avoid any ambiguity.

WARNING: If you're incorporating user input into the command, make sure to sanitize and validate the input to prevent any malicious commands from being executed.

As `check` is True, if the exit code is non-zero, it raises a CalledProcessError.

scanpipe.pipes.fetch.**get_request_session**(*uri*)

Return a Requests session setup with authentication and headers.

scanpipe.pipes.fetch.**fetch_http**(*uri*, *to=None*)

Download a given *uri* in a temporary directory and return the directory's path.

exception scanpipe.pipes.fetch.**FetchDockerImageError**

scanpipe.pipes.fetch.**get_docker_image_platform**(*docker_url*)

Return a platform mapping of a docker reference. If there are more than one, return the first one by default.

scanpipe.pipes.fetch.**fetch_docker_image**(*docker_url*, *to=None*)

Fetch a docker image from the provided Docker image *docker_url*, using the "docker://reference" URL syntax. Return a *Download* object.

Docker references are documented here: https://github.com/containers/skopeo/blob/0faf16017/docs/skopeo.1.md#image-names

scanpipe.pipes.fetch.**get_fetcher**(*url*)

Return the fetcher function based on the provided *url* scheme.

scanpipe.pipes.fetch.**fetch_url**(*url*)

Fetch provided *url* and returns the result as a *Download* object.

scanpipe.pipes.fetch.**fetch_urls**(*urls*)

Fetch provided *urls* list. The *urls* can also be provided as a string containing one URL per line. Return the fetched URLs as *downloads* objects and a list of *errors*.

scanpipe.pipes.fetch.**check_urls_availability**(*urls*)

Check the accessibility of a list of URLs.

18.9 Flag

`scanpipe.pipes.flag.`**`flag_empty_files`**`(`*project*`)`

Flag empty files as ignored.

`scanpipe.pipes.flag.`**`flag_ignored_directories`**`(`*project*`)`

Flag directories as ignored.

`scanpipe.pipes.flag.`**`flag_ignored_patterns`**`(`*project*, *patterns*`)`

Flag codebase resource as `ignored` status from list of `patterns`.

`scanpipe.pipes.flag.`**`analyze_scanned_files`**`(`*project*`)`

Set the status for CodebaseResource to unknown or no license.

`scanpipe.pipes.flag.`**`flag_not_analyzed_codebase_resources`**`(`*project*`)`

Flag codebase resource as *not-analyzed*.

`scanpipe.pipes.flag.`**`flag_mapped_resources`**`(`*project*`)`

Flag all codebase resources that were mapped during the d2d pipeline.

18.10 Input

`scanpipe.pipes.input.`**`copy_input`**`(`*input_location*, *dest_path*`)`

Copy the `input_location` to the `dest_path`.

`scanpipe.pipes.input.`**`copy_inputs`**`(`*input_locations*, *dest_path*`)`

Copy the provided `input_locations` to the `dest_path`.

`scanpipe.pipes.input.`**`move_input`**`(`*input_location*, *dest_path*`)`

Move the provided `input_location` to the `dest_path`.

`scanpipe.pipes.input.`**`move_inputs`**`(`*inputs*, *dest_path*`)`

Move the provided `inputs` to the `dest_path`.

`scanpipe.pipes.input.`**`get_tool_name_from_scan_headers`**`(`*scan_data*`)`

Return the `tool_name` of the first header in the provided `scan_data`.

`scanpipe.pipes.input.`**`is_archive`**`(`*location*`)`

Return True if the file at `location` is an archive.

`scanpipe.pipes.input.`**`load_inventory_from_toolkit_scan`**`(`*project*, *input_location*`)`

Create packages, dependencies, and resources loaded from the ScanCode-toolkit scan results located at `input_location`.

`scanpipe.pipes.input.`**`load_inventory_from_scanpipe`**`(`*project*, *scan_data*`)`

Create packages, dependencies, resources, and relations loaded from a ScanCode.io JSON output provided as `scan_data`.

`scanpipe.pipes.input.`**`get_worksheet_data`**`(`*worksheet*`)`

Return the data from provided `worksheet` as a list of dict.

`scanpipe.pipes.input.`**`clean_xlsx_field_value`**`(`*model_class*, *field_name*, *value*`)`

Clean the `value` for compatibility with the database `model_class`.

`scanpipe.pipes.input.`**`clean_xlsx_data_to_model_data`**`(`*model_class*, *xlsx_data*`)`

Clean the `xlsx_data` for compatibility with the database `model_class`.

`scanpipe.pipes.input.`**`load_inventory_from_xlsx`**`(`*project*, *input_location*`)`

Create packages, dependencies, resources, and relations loaded from XLSX file located at `input_location`.

18.11 JS

`scanpipe.pipes.js.`**`is_source_mapping_in_minified`**`(`*resource*, *map_file_name*`)`

Return True if a string contains a source mapping in its last 5 lines.

`scanpipe.pipes.js.`**`sha1`**`(`*content*`)`

Calculate the SHA-1 hash of a string.

`scanpipe.pipes.js.`**`source_content_sha1_list`**`(`*map_file*`)`

Return list containing sha1 of sourcesContent.

`scanpipe.pipes.js.`**`load_json_from_file`**`(`*location*`)`

Return the deserialized json content from `location`.

`scanpipe.pipes.js.`**`get_map_sources`**`(`*map_file*`)`

Return source paths from a map file.

`scanpipe.pipes.js.`**`get_map_sources_content`**`(`*map_file*`)`

Return sources contents from a map file.

`scanpipe.pipes.js.`**`get_minified_resource`**`(`*map_resource*, *minified_resources*`)`

Return the corresponding minified_resource given a `map_resource` Resource object and a `minified_resources` query set of minified JS Resource. Return None if it cannot be found.

`scanpipe.pipes.js.`**`get_js_map_basename_and_extension`**`(`*filename*`)`

Return a 2-tuple pf (basename, extension) of a JavaScript/TypeScript related file. Return None otherwise.

18.12 JVM

Support for JVM-specific file formats such as .class and .java files.

`scanpipe.pipes.jvm.`**`get_java_package`**`(`*location*, *java_extensions=('.java',)*, ***kwargs*`)`

Return a Java package as a mapping with a single "java_package" key, or `None` from the .java source code file at `location`.

Only look at files with an extension in the `java_extensions` tuple.

Note: this is the same API as a ScanCode Toolkit API scanner function by design.

`scanpipe.pipes.jvm.`**`find_java_package`**`(`*lines*`)`

Return a mapping of `{'java_package': <value>}` or `None` from an iterable or text `lines`.

For example:

```
>>> lines = ["   package    foo.back ;  # dsasdasdasdasdasda.asdasdasd"]
>>> assert find_java_package(lines) == {"java_package": "foo.back"}
```

scanpipe.pipes.jvm.**get_normalized_java_path**(*path*)

Return a normalized .java file path for `path` .class file path string. Account for inner classes in that their .java file name is the name of their outer class.

For example:

```
>>> get_normalized_java_path("foo/org/common/Bar$inner.class")
'foo/org/common/Bar.java'
>>> get_normalized_java_path("foo/org/common/Bar.class")
'foo/org/common/Bar.java'
```

scanpipe.pipes.jvm.**get_fully_qualified_java_path**(*java_package*, *filename*)

Return a fully qualified java path of a .java `filename` in a `java_package` string. Note that we use "/" as path separators.

For example:

```
>>> get_fully_qualified_java_path("org.common" , "Bar.java")
'org/common/Bar.java'
```

18.13 MatchCode

exception scanpipe.pipes.matchcode.**MatchCodeIOException**

scanpipe.pipes.matchcode.**is_configured**()

Return True if the required MatchCode.io settings have been set.

scanpipe.pipes.matchcode.**is_available**()

Return True if the configured MatchCode.io server is available.

scanpipe.pipes.matchcode.**request_get**(*url*, *payload=None*, *timeout=60*)

Wrap the HTTP request calls on the API.

scanpipe.pipes.matchcode.**save_directory_fingerprints**(*project*, *virtual_codebase*, *to_codebase_only=False*)

Save directory fingerprints from directory Resources in *virtual_codebase* to the directory CodebaseResources from *project* that have the same path.

If *to_codebase_only* is True, then we are only saving the directory fingerprints for directories from the to/ codebase of a d2d project.

scanpipe.pipes.matchcode.**fingerprint_codebase_directories**(*project*, *to_codebase_only=False*)

Compute directory fingerprints for the directories from *project*.

These directory fingerprints are used for matching purposes on matchcode.

If *to_codebase_only* is True, the only directories from the *to/* codebase are computed.

scanpipe.pipes.matchcode.**fingerprint_codebase_resource**(*location*, *with_threading=True*, ***kwargs*)

Compute fingerprints for the resource at *location* using the scancode-toolkit direct API.

Return a dictionary of scan *results* and a list of *errors*.

scanpipe.pipes.matchcode.**save_resource_fingerprints**(*resource*, *scan_results*, *scan_errors=None*)

Save computed fingerprints from *scan_results* to *resource.extra_data*. Create project errors if any occurred during the scan.

`scanpipe.pipes.matchcode.`**`fingerprint_codebase_resources`**`(`*project*, *resource_qs=None*, *progress_logger=None*, *to_codebase_only=False*`)`

Compute fingerprints for the resources from *project*.

These resource fingerprints are used for matching purposes on matchcode.

Multiprocessing is enabled by default on this pipe, the number of processes can be controlled through the SCANCODEIO_PROCESSES setting.

If *to_codebase_only* is True, the only resources from the *to/* codebase are computed.

`scanpipe.pipes.matchcode.`**`send_project_json_to_matchcode`**`(`*project*, *timeout=60*, *api_url=None*`)`

Given a *project*, create a JSON scan of the *project* CodebaseResources and send it to MatchCode.io for matching. Return a tuple containing strings of the url to the particular match run and the url to the match results.

`scanpipe.pipes.matchcode.`**`get_run_url_status`**`(`*run_url*, ***kwargs*`)`

Given a *run_url*, which is a URL to a ScanCode.io Project run, return its status, otherwise return None.

`scanpipe.pipes.matchcode.`**`poll_run_url_status`**`(`*run_url*, *sleep=10*`)`

Given a URL to a scancode.io run instance, *run_url*, return True when the run instance has completed successfully.

Raise a MatchCodeIOException when the run instance has failed, stopped, or gone stale.

`scanpipe.pipes.matchcode.`**`get_match_results`**`(`*run_url*`)`

Given the *run_url* for a pipeline running the matchcode matching pipeline, return the match results for that run.

`scanpipe.pipes.matchcode.`**`map_match_results`**`(`*match_results*`)`

Given *match_results*, which is a mapping of ScanCode.io codebase results, return a defaultdict(list) where the keys are the package_uid of matched packages and the value is a list containing the paths of Resources associated with the package_uid.

`scanpipe.pipes.matchcode.`**`create_packages_from_match_results`**`(`*project*, *match_results*`)`

Given *match_results*, which is a mapping of ScanCode.io codebase results, use the Package data from it to create DiscoveredPackages for *project* and associate the proper Resources of *project* to the DiscoveredPackages.

18.14 Output

`scanpipe.pipes.output.`**`safe_filename`**`(`*filename*`)`

Convert the provided *filename* to a safe filename.

`scanpipe.pipes.output.`**`get_queryset`**`(`*project*, *model_name*`)`

Return a consistent QuerySet for all supported outputs (json, xlsx, csv, …)

`scanpipe.pipes.output.`**`queryset_to_csv_file`**`(`*queryset*, *fieldnames*, *output_file*`)`

Output csv content generated from the provided *queryset* objects to the *output_file*. The fields to be included as columns and their order are controlled by the *fieldnames* list.

`scanpipe.pipes.output.`**`queryset_to_csv_stream`**`(`*queryset*, *fieldnames*, *output_stream*`)`

Output csv content generated from the provided *queryset* objects to the *output_stream*. The fields to be included as columns and their order are controlled by the *fieldnames* list.

`scanpipe.pipes.output.`**`to_csv`**`(`*project*`)`

Generate output for the provided *project* in csv format. Since the csv format does not support multiple tabs, one file is created per object type. The output files are created in the *project* output/ directory. Return a list of paths of the generated output files.

scanpipe.pipes.output.**to_json**(*project*)

Generate output for the provided *project* in JSON format. The output file is created in the *project* output/ directory. Return the path of the generated output file.

scanpipe.pipes.output.**queryset_to_xlsx_worksheet**(*queryset*, *workbook*, *exclude_fields=()*)

Add a new worksheet to the `workbook` `xlsxwriter.Workbook` using the `queryset`. The `queryset` "model_name" is used as a name for the "worksheet". Exclude fields listed in the `exclude_fields` sequence of field names.

Add an extra trailing "xlsx_errors" column with conversion error messages if any. Return a number of conversion errors.

scanpipe.pipes.output.**to_xlsx**(*project*)

Generate output for the provided `project` in XLSX format. The output file is created in the `project` "output/" directory. Return the path of the generated output file.

Note that the XLSX worksheets contain each an extra "xlsx_errors" column with possible error messages for a row when converting the data to XLSX exceed the limits of what can be stored in a cell.

scanpipe.pipes.output.**to_spdx**(*project*, *include_files=False*)

Generate output for the provided `project` in SPDX document format. The output file is created in the `project` "output/" directory. Return the path of the generated output file.

scanpipe.pipes.output.**get_cyclonedx_bom**(*project*)

Return a CycloneDX *Bom* object filled with provided *project* data. See https://cyclonedx.org/use-cases/#dependency-graph

scanpipe.pipes.output.**sort_bom_with_schema_ordering**(*bom_as_dict*, *schema_version*)

Sort the `bom_as_dict` using the ordering from the `schema_version`.

scanpipe.pipes.output.**to_cyclonedx**(*project*, *version='1.6'*)

Generate output for the provided `project` in CycloneDX BOM format. The output file is created in the `project` "output/" directory. Return the path of the generated output file.

scanpipe.pipes.output.**render_template**(*template_string*, *context*)

Render a Django `template_string` using the `context` dict.

scanpipe.pipes.output.**render_template_file**(*template_location*, *context*)

Render a Django template at `template_location` using the `context` dict.

scanpipe.pipes.output.**get_attribution_template**(*project*)

Return a custom attribution template if provided or the default one.

scanpipe.pipes.output.**make_unknown_license_object**(*license_symbol*)

Return a `License` object suitable for the provided `license_symbol`, that is representing a license key unknown by the current toolkit licensed index.

scanpipe.pipes.output.**get_package_expression_symbols**(*parsed_expression*)

Return the list of `license_symbols` contained in the `parsed_expression`. Since unknown license keys are missing a `License` set in the `wrapped` attribute, a special "unknown" `License` object is injected.

scanpipe.pipes.output.**get_package_data_for_attribution**(*package*, *licensing*)

Convert the `package` instance into a dictionary of values usable during attribution generation.

scanpipe.pipes.output.**get_unique_licenses**(*packages*)

Return a list of unique License symbol objects preserving ordering. Return an empty list if the packages do not have licenses.

Replace by the following one-liner once this toolkit issues is fixed: https://github.com/nexB/scancode-toolkit/issues/3425 licenses = set(license for package in packages for license in package["licenses"])

`scanpipe.pipes.output.`**`to_attribution`**`(`*project*`)`

Generate attribution for the provided `project`. The output file is created in the `project` "output/" directory. Return the path of the generated output file.

Custom template can be provided in the *codebase/.scancode/templates/attribution.html* location.

The model instances are converted into data dict to prevent any data leak as the attribution template is customizable.

18.15 PathMap

`class scanpipe.pipes.pathmap.`**`Match`**`(`*matched_path_length*, *resource_ids*`)`

`matched_path_length:` `int`

Alias for field number 0

`resource_ids:` `list`

Alias for field number 1

`scanpipe.pipes.pathmap.`**`find_paths`**`(`*path*, *index*`)`

Return a Match for the longest paths matched in the `index` automaton for a POSIX `path` string. Return None if there is not matching paths found.

`scanpipe.pipes.pathmap.`**`build_index`**`(`*resource_id_and_paths*, *with_subpaths=True*`)`

Return an index (an index) built from a `resource_id_and_paths` iterable of tuples of (resource_id int, resource_path string).

If *with_subpaths*` is True, index all suffixes of the paths, other index and match only each complete path.

For example, for the path "samples/JGroups/src/RouterStub.java", the suffixes are:

samples/JGroups/src/RouterStub.java

JGroups/src/RouterStub.java

src/RouterStub.java

RouterStub.java

`scanpipe.pipes.pathmap.`**`add_path`**`(`*resource_id*, *segments*, *segments_count*, *index*`)`

Add the `resource_id` path represented by its list of reversed path `segments` with `segments_count` segments to the `index` automaton.

`scanpipe.pipes.pathmap.`**`add_subpaths`**`(`*resource_id*, *segments*, *segments_count*, *index*`)`

Add all the `resource_id` subpaths "suffixes" of the resource path as represented by its list of reversed path `segments` with `segments_count` segments to the `index` automaton.

`scanpipe.pipes.pathmap.`**`get_reversed_path_segments`**`(`*path*`)`

Return reversed segments list given a POSIX `path` string. We reverse based on path segments separated by a "/".

Note that the inputh `path` is assumed to be normalized, not relative and not containing double slash.

For example:: >>> assert get_reversed_path_segments("a/b/c.js") == ["c.js", "b", "a"] `scanpipe.pipes.pathmap.`**`convert_segments_to_path`**`(`*segments*`)`

Return a path string is suitable for indexing or matching given a `segments` sequence of path segment strings. The resulting reversed path is prefixed and suffixed by a "/" irrespective of whether the original path is a file or directory and had such prefix or suffix.

For example:: >>> assert convert_segments_to_path(["c.js", "b", "a"]) == "/c.js/b/a/"

18.16 PurlDB

`exception` `scanpipe.pipes.purldb.`**`PurlDBException`**

`scanpipe.pipes.purldb.`**`is_configured`**`()`

Return True if the required PurlDB settings have been set.

`scanpipe.pipes.purldb.`**`is_available`**`()`

Return True if the configured PurlDB server is available.

`scanpipe.pipes.purldb.`**`request_get`**`(`*url*, *payload=None*, *timeout=60*`)`

Wrap the HTTP request calls on the API.

`scanpipe.pipes.purldb.`**`collect_response_results`**`(`*response*, *data*, *timeout=60*`)`

Return all results from a purldb API response.

`scanpipe.pipes.purldb.`**`match_packages`**`(`*sha1_list*, *enhance_package_data=False*, *timeout=60*, *api_url=None*`)`

Match a list of SHA1 in the PurlDB for package-type files.

If *enhance_package_data* is True, then purldb will enhance Package data for matched Packages, if possible.

`scanpipe.pipes.purldb.`**`match_resources`**`(`*sha1_list*, *timeout=60*, *api_url=None*`)`

Match a list of SHA1 in the PurlDB for resource files.

`scanpipe.pipes.purldb.`**`match_directory`**`(`*fingerprint*, *timeout=60*, *api_url=None*`)`

Match directory content fingerprint in the PurlDB for a single directory resource.

`scanpipe.pipes.purldb.`**`submit_purls`**`(`*packages*, *timeout=60*, *api_url=None*`)`

Submit list of dict where each dict has either resolved PURL i.e. PURL with version or version-less PURL along with vers range to PurlDB for indexing.

`scanpipe.pipes.purldb.`**`feed_purldb`**`(`*packages*, *chunk_size*, *logger=<bound method Logger.info of <Logger scanpipe.pipes.purldb (INFO)>>*`)`

Feed PurlDB with list of PURLs for indexing.

`scanpipe.pipes.purldb.`**`get_unique_resolved_purls`**`(`*project*`)`

Return PURLs from project's resolved DiscoveredDependencies.

`scanpipe.pipes.purldb.`**`get_unique_unresolved_purls`**`(`*project*`)`

Return PURLs from project's unresolved DiscoveredDependencies.

`scanpipe.pipes.purldb.`**`populate_purldb_with_discovered_packages`**`(`*project*, *logger=<bound method Logger.info of <Logger scanpipe.pipes.purldb (INFO)>>*`)`

Add DiscoveredPackage to PurlDB.

`scanpipe.pipes.purldb.`**`populate_purldb_with_discovered_dependencies`**`(`*project*, *logger=<bound method Logger.info of <Logger scanpipe.pipes.purldb (INFO)>>*`)`

Add DiscoveredDependency to PurlDB.

`scanpipe.pipes.purldb.`**`get_package_by_purl`**`(`*package_url*`)`

Get a Package details entry providing its *package_url*.

`scanpipe.pipes.purldb.`**`find_packages`**`(`*payload*`)`

Get Packages using provided *payload* filters on the PurlDB package list.

`scanpipe.pipes.purldb.`**`get_next_download_url`**`(`*timeout=60*, *api_url=None*`)`

Return the ScannableURI UUID, download URL, and pipelines for the next Package to be scanned from PurlDB

Return None if the request was not successful

`scanpipe.pipes.purldb.`**`send_results_to_purldb`**`(`*scannable_uri_uuid*, *scan_results_location*, *scan_summary_location*, *project_extra_data*, *timeout=60*, *api_url=None*`)`

Send project results to purldb for the package handeled by the ScannableURI with uuid of *scannable_uri_uuid*

`scanpipe.pipes.purldb.`**`update_status`**`(`*scannable_uri_uuid*, *status*, *scan_log=''*, *timeout=60*, *api_url=None*`)`

Update the status of a ScannableURI on a PurlDB scan queue

`scanpipe.pipes.purldb.`**`create_project_name`**`(`*download_url*, *scannable_uri_uuid*`)`

Create a project name from *download_url* and *scannable_uri_uuid*

`scanpipe.pipes.purldb.`**`poll_run_status`**`(`*project*, *sleep=10*`)`

Poll the status of all runs of *project*. Raise a PurlDBException with a message containing the log of the run if the run has stopped, failed, or gone stale, otherwise return None.

`scanpipe.pipes.purldb.`**`get_run_status`**`(`*run*, ***kwargs*`)`

Refresh the values of *run* and return its status

18.17 Resolve

`scanpipe.pipes.resolve.`**`resolve_manifest_resources`**`(`*resource*, *package_registry*`)`

Get package data from resource.

`scanpipe.pipes.resolve.`**`get_packages`**`(`*project*, *package_registry*, *manifest_resources*, *model=None*`)`

Get package data from package manifests/lockfiles/SBOMs or get package data for resolved packages from package requirements.

`scanpipe.pipes.resolve.`**`create_packages_and_dependencies`**`(`*project*, *packages*, *resolved=False*`)`

Create DiscoveredPackage and DiscoveredDependency objects for packages detected in a package manifest, lockfile or SBOM.

If resolved, create packages out of resolved dependencies, otherwise create dependencies.

`scanpipe.pipes.resolve.`**`get_packages_from_manifest`**`(`*input_location*, *package_registry=None*`)`

Resolve packages or get packages data from a package manifest file/ lockfile/SBOM at *input_location*.

`scanpipe.pipes.resolve.`**`get_manifest_resources`**`(`*project*`)`

Get all resources in the codebase which are package manifests.

`scanpipe.pipes.resolve.`**`resolve_pypi_packages`**`(`*input_location*`)`

Resolve the PyPI packages from the *input_location* requirements file.

`scanpipe.pipes.resolve.`**`resolve_about_package`**`(`*input_location*`)`

Resolve the package from the `input_location` .ABOUT file.

`scanpipe.pipes.resolve.`**`populate_license_notice_fields_about`**`(`*package_data*, *about_data*`)`

Populate `package_data` with license and notice attributes from `about_data`.

`scanpipe.pipes.resolve.`**`resolve_about_packages`**`(`*input_location*`)`

Wrap `resolve_about_package` to return a list as expected by the InspectManifest pipeline.

`scanpipe.pipes.resolve.`**`convert_spdx_expression`**`(`*license_expression_spdx*`)`

Return an ScanCode license expression from a SPDX *license_expression_spdx* string.

`scanpipe.pipes.resolve.`**`resolve_spdx_packages`**`(`*input_location*`)`

Resolve the packages from the *input_location* SPDX document file.

`scanpipe.pipes.resolve.`**`get_default_package_type`**`(`*input_location*`)`

Return the package type associated with the provided *input_location*. This type is used to get the related handler that knows how process the input.

`scanpipe.pipes.resolve.`**`set_license_expression`**`(`*package_data*`)`

Set the license expression from a detected license dict/str in provided *package_data*.

18.18 RootFS

`exception` `scanpipe.pipes.rootfs.`**`DistroNotFound`**

`exception` `scanpipe.pipes.rootfs.`**`DistroNotSupported`**

`class` `scanpipe.pipes.rootfs.`**`RootFs`**`(`*location*, *distro=None*`)`

A root filesystem.

`classmethod from_project_codebase``(`*project*`)`

Return RootFs objects collected from the project's "codebase" directory. Each directory in the input/ is considered as the root of a root filesystem.

`get_resources``(`*with_dir=False*`)`

Return a Resource for each file in this rootfs.

`get_installed_packages``(`*packages_getter*`)`

Return tuples of (package_url, package) for installed packages found in this rootfs layer using the *packages_getter* function or callable.

The *packages_getter()* function should:

- Accept a first argument string that is the root directory of filesystem of this rootfs
- Return tuples of (package_url, package) where package_url is a package_url string that uniquely identifies a package; while, a *package* is an object that represents a package (typically a scancode- toolkit packagedcode.models.Package class or some nested mapping with the same structure).

The *packages_getter* function would typically query the system packages database, such as an RPM database or similar, to collect the list of installed system packages.

`__init__`(*location*, *distro=None*) → None

Method generated by attrs for class RootFs.

`scanpipe.pipes.rootfs.`**`get_resources`**(*location*, *with_dir=False*)

Return the Resource found in the *location* in root directory of a rootfs.

`scanpipe.pipes.rootfs.`**`create_codebase_resources`**(*project*, *rootfs*)

Create the CodebaseResource for a *rootfs* in *project*.

`scanpipe.pipes.rootfs.`**`has_hash_diff`**(*install_file*, *codebase_resource*)

Return True if one of available hashes on both *install_file* and *codebase_resource*, by hash type, is different. For example: Alpine uses SHA1 while Debian uses MD5, we prefer the strongest hash that's present.

`scanpipe.pipes.rootfs.`**`package_getter`**(*root_dir*, ***kwargs*)

Return installed package objects.

`scanpipe.pipes.rootfs.`**`scan_rootfs_for_system_packages`**(*project*, *rootfs*)

Given a *project* Project and a *rootfs* RootFs, scan the *rootfs* for installed system packages, and create a DiscoveredPackage for each.

Then for each installed DiscoveredPackage file, check if it exists as a CodebaseResource. If exists, relate that CodebaseResource to its DiscoveredPackage; otherwise, keep that as a missing file.

`scanpipe.pipes.rootfs.`**`get_resource_with_md5`**(*project*, *status*)

Return a queryset of CodebaseResource from a *project* that has a *status*, a non-empty size, and md5.

`scanpipe.pipes.rootfs.`**`match_not_analyzed`**(*project*, *reference_status='system-package'*, *not_analyzed_status='not-analyzed'*)

Given a *project* Project : 1. Build an MD5 index of files assigned to a package that has a status of *reference_status* 2. Attempt to match resources with status *not_analyzed_status* to that index 3. Relate each matched CodebaseResource to the matching DiscoveredPackage and set its status.

`scanpipe.pipes.rootfs.`**`flag_uninteresting_codebase_resources`**(*project*)

Flag any file that do not belong to any system package and determine if it's: - A temp file - Generated - Log file of sorts (such as var) using few heuristics

`scanpipe.pipes.rootfs.`**`flag_ignorable_codebase_resources`**(*project*)

Flag codebase resource using the glob patterns from commoncode.ignore of ignorable files/directories, if their paths match an ignorable pattern.

`scanpipe.pipes.rootfs.`**`flag_data_files_with_no_clues`**(*project*)

Flag CodebaseResources that have a file type of *data* and no detected clues to be uninteresting.

`scanpipe.pipes.rootfs.`**`flag_media_files_as_uninteresting`**(*project*)

Flag CodebaseResources that are media files to be uninteresting.

`scanpipe.pipes.rootfs.`**`get_rootfs_data`**(*root_fs*)

Return a mapping of rootfs-related data given a `root_fs`.

18.19 ScanCode

`scanpipe.pipes.scancode.logger = <Logger scanpipe.pipes (INFO)>`

Utilities to deal with ScanCode toolkit features and objects.

exception `scanpipe.pipes.scancode.InsufficientResourcesError`

`scanpipe.pipes.scancode.get_max_workers`(*keep_available*)

Return the *SCANCODEIO_PROCESSES* if defined in the setting, or returns a default value based on the number of available CPUs, minus the provided *keep_available* value.

On operating system where the multiprocessing start method is not "fork", but for example "spawn", such as on macOS, multiprocessing and threading are disabled by default returning -1 *max_workers*.

`scanpipe.pipes.scancode.extract_archive`(*location*, *target*)

Extract a single archive or compressed file at *location* to the *target* directory.

Return a list of extraction errors.

Wrapper of the *extractcode.api.extract_archive* function.

`scanpipe.pipes.scancode.extract_archives`(*location*, *recurse=False*)

Extract all archives at *location* and return errors.

Archives and compressed files are extracted in a new directory named "<file_name>-extract" created in the same directory as each extracted archive.

If *recurse* is True, extract nested archives-in-archives recursively.

Return a list of extraction errors.

Wrapper of the *extractcode.api.extract_archives* function.

`scanpipe.pipes.scancode.get_resource_info`(*location*)

Return a mapping suitable for the creation of a new CodebaseResource.

`scanpipe.pipes.scancode.scan_file`(*location*, *with_threading=True*, *min_license_score=0*, ***kwargs*)

Run a license, copyright, email, and url scan on a provided *location*, using the scancode-toolkit direct API.

Return a dictionary of scan *results* and a list of *errors*.

`scanpipe.pipes.scancode.scan_for_package_data`(*location*, *with_threading=True*, *package_only=False*, ***kwargs*)

Run a package scan on provided *location* using the scancode-toolkit direct API.

Return a dict of scan *results* and a list of *errors*.

`scanpipe.pipes.scancode.save_scan_file_results`(*codebase_resource*, *scan_results*, *scan_errors*)

Save the resource scan file results in the database. Create project errors if any occurred during the scan.

`scanpipe.pipes.scancode.save_scan_package_results`(*codebase_resource*, *scan_results*, *scan_errors*)

Save the resource scan package results in the database. Create project errors if any occurred during the scan.

`scanpipe.pipes.scancode.scan_resources`(*resource_qs*, *scan_func*, *save_func*, *scan_func_kwargs=None*, *progress_logger=None*)

Run the *scan_func* on the codebase resources of the provided *resource_qs*. The *save_func* is called to save the results.

Multiprocessing is enabled by default on this pipe, the number of processes can be controlled through the *SCANCODEIO_PROCESSES* setting. Multiprocessing can be disabled using *SCANCODEIO_PROCESSES=0*, and threading can also be disabled *SCANCODEIO_PROCESSES=-1*

The codebase resources QuerySet is chunked in 2000 results at the time, this can result in a significant reduction in memory usage.

Note that all database related actions are executed in this main process as the database connection does not always fork nicely in the pool processes.

`scanpipe.pipes.scancode.`**`scan_for_files`**(*project*, *resource_qs=None*, *progress_logger=None*)

Run a license, copyright, email, and url scan on files without a status for a *project*.

Multiprocessing is enabled by default on this pipe, the number of processes can be controlled through the SCAN-CODEIO_PROCESSES setting.

`scanpipe.pipes.scancode.`**`scan_for_application_packages`**(*project*, *assemble=True*, *package_only=False*, *progress_logger=None*)

Run a package scan on resources without a status for a *project*, and add them in their respective *package_data* attribute. Then create DiscoveredPackage and DiscoveredDependency instances from the detected package data optionally. If the *assemble* argument is set to *True*, DiscoveredPackage and DiscoveredDependency instances are created and added to the project by assembling resource level package_data, and resources which belong in the DiscoveredPackage instance, are assigned to that package.

Multiprocessing is enabled by default on this pipe, the number of processes can be controlled through the SCAN-CODEIO_PROCESSES setting.

`scanpipe.pipes.scancode.`**`add_resource_to_package`**(*package_uid*, *resource*, *project*)

Relate a DiscoveredPackage to *resource* from *project* using *package_uid*.

Add a ProjectMessage when the DiscoveredPackage could not be fetched using the provided *package_uid*.

`scanpipe.pipes.scancode.`**`assemble_packages`**(*project*)

Create instances of DiscoveredPackage and DiscoveredDependency for *project* from the parsed package data present in the CodebaseResources of *project*, using the respective package handlers for each package manifest type.

`scanpipe.pipes.scancode.`**`process_package_data`**(*project*)

Create instances of DiscoveredPackage and DiscoveredDependency for *project* from the parsed package data present in the CodebaseResources of *project*.

Here package assembly though package handlers are not performed, instead package/dependency objects are created directly from package data.

`scanpipe.pipes.scancode.`**`get_packages_with_purl_from_resources`**(*project*)

Yield Dependency or PackageData objects created from detected package_data in all the project resources. Both Dependency and PackageData objects have the *purl* attribute with a valid purl.

`scanpipe.pipes.scancode.`**`get_pretty_params`**(*args*)

Format provided `args` for the `pretty_params` run_scan argument.

`scanpipe.pipes.scancode.`**`run_scan`**(*location*, *output_file*, *run_scan_args*)

Scan the *location* content and write the results into an *output_file*.

`scanpipe.pipes.scancode.`**`get_virtual_codebase`**(*project*, *input_location*)

Return a ScanCode virtual codebase built from the JSON scan file located at the *input_location*.

`scanpipe.pipes.scancode.`**`create_codebase_resources`**(*project*, *scanned_codebase*)

Save the resources of a ScanCode *scanned_codebase* scancode.resource.Codebase object to the database as a CodebaseResource of the *project*. This function can be used to expend an existing *project* Codebase with new CodebaseResource objects as the existing objects (based on the *path*) will be skipped.

`scanpipe.pipes.scancode.`**`create_discovered_packages`**`(`*project*, *scanned_codebase*`)`

Save the packages of a ScanCode *scanned_codebase* scancode.resource.Codebase object to the database as a DiscoveredPackage of *project*.

`scanpipe.pipes.scancode.`**`create_discovered_dependencies`**`(`*project*, *scanned_codebase*, *strip_datafile_path_root=False*`)`

Save the dependencies of a ScanCode *scanned_codebase* scancode.resource.Codebase object to the database as a DiscoveredDependency of *project*.

If *strip_datafile_path_root* is True, then *DiscoveredDependency.create_from_data()* will strip the root path segment from the *datafile_path* of *dependency_data* before looking up the corresponding CodebaseResource for *datafile_path*. This is used in the case where Dependency data is imported from a scancode-toolkit scan, where the root path segments are not stripped for *datafile_path*.

`scanpipe.pipes.scancode.`**`set_codebase_resource_for_package`**`(`*codebase_resource*, *discovered_package*`)`

Assign the *discovered_package* to the *codebase_resource* and set its status to "application-package".

`scanpipe.pipes.scancode.`**`get_license_matches_grouped`**`(`*project*`)`

Return a dictionary of all license_matches of a given `project` grouped by `resource.detected_license_expression`.

`scanpipe.pipes.scancode.`**`make_results_summary`**`(`*project*, *scan_results_location*`)`

Extract selected sections of the Scan results, such as the *summary license_clarity_score*, and *license_matches* related data. The *key_files* are also collected and injected in the *summary* output.

18.20 SPDX

`scanpipe.pipes.spdx.`**`SPDX_SCHEMA_URL`** `=`
`'https://github.com/spdx/spdx-spec/raw/development/v2.3.1/schemas/spdx-schema.json'`

Generate SPDX Documents. Spec documentation: https://spdx.github.io/spdx-spec/v2.3/

Usage:

```python
import pathlib
from scanpipe.pipes import spdx

creation_info = spdx.CreationInfo(
    person_name="John Doe",
    person_email="john@starship.space",
    organization_name="Starship",
    tool="SPDXCode-1.0",
)

package1 = spdx.Package(
    spdx_id="SPDXRef-package1",
    name="lxml",
    version="3.3.5",
    license_concluded="LicenseRef-1",
    checksums=[
        spdx.Checksum(
            algorithm="SHA1", value="10c72b88de4c5f3095ebe20b4d8afbedb32b8f"
        ),
```

(continues on next page)

(continued from previous page)

```
        spdx.Checksum(algorithm="MD5", value="56770c1a2df6e0dc51c491f0a5b9d865"),
    ],
    external_refs=[
        spdx.ExternalRef(
            category="PACKAGE-MANAGER",
            type="purl",
            locator="pkg:pypi/lxml@3.3.5",
        ),
    ]
)

document = spdx.Document(
    name="Document name",
    namespace="https://[CreatorWebsite]/[pathToSpdx]/[DocumentName]-[UUID]",
    creation_info=creation_info,
    packages=[package1],
    extracted_licenses=[
        spdx.ExtractedLicensingInfo(
            license_id="LicenseRef-1",
            extracted_text="License Text",
            name="License 1",
            see_alsos=["https://license1.text"],
        ),
    ],
    comment="This document was created using SPDXCode-1.0",
)

# Display document content:
print(document.as_json())

# Validate document
schema = pathlib.Path(spdx.SPDX_JSON_SCHEMA_LOCATION).read_text()
document.validate(schema)

# Write document to a file:
with open("document_name.spdx.json", "w") as f:
    f.write(document.as_json())
```

class `scanpipe.pipes.spdx.`**`CreationInfo`**(*person_name: str = '', organization_name: str = '', tool: str = '', person_email: str = '', organization_email: str = '', license_list_version: str = '3.20', comment: str = '', created: str = <factory>*)

One instance is required for each SPDX file produced. It provides the necessary information for forward and backward compatibility for processing tools.

`comment``: str = ''`

Identify when the SPDX document was originally created. The date is to be specified according to combined date and time in UTC format as specified in ISO 8601 standard. Format: YYYY-MM-DDThh:mm:ssZ

`as_dict``()`

Return the data as a serializable dict.

`get_creators_spdx``()`

Return the *creators* list from related field values.

`static get_creators_dict`(*creators_data*)

Return the *creators* dict from SPDX data.

`__init__`(*person_name: str = ''*, *organization_name: str = ''*, *tool: str = ''*, *person_email: str = ''*, *organization_email: str = ''*, *license_list_version: str = '3.20'*, *comment: str = ''*, *created: str = <factory>*) → None

`class scanpipe.pipes.spdx.Checksum`(*algorithm: str*, *value: str*)

The checksum provides a mechanism that can be used to verify that the contents of a File or Package have not changed.

`as_dict()`

Return the data as a serializable dict.

`__init__`(*algorithm: str*, *value: str*) → None

`class scanpipe.pipes.spdx.ExternalRef`(*category: str*, *type: str*, *locator: str*, *comment: str = ''*)

An External Reference allows a Package to reference an external source of additional information, metadata, enumerations, asset identifiers, or downloadable content believed to be relevant to the Package.

`as_dict()`

Return the data as a serializable dict.

`__init__`(*category: str*, *type: str*, *locator: str*, *comment: str = ''*) → None

`class scanpipe.pipes.spdx.ExtractedLicensingInfo`(*license_id: str*, *extracted_text: str*, *name: str = ''*, *comment: str = ''*, *see_alsos: ~typing.List[str] = <factory>*)

An ExtractedLicensingInfo represents a license or licensing notice that was found in a package, file or snippet. Any license text that is recognized as a license may be represented as a License rather than an ExtractedLicensingInfo.

`as_dict()`

Return the data as a serializable dict.

`__init__`(*license_id: str*, *extracted_text: str*, *name: str = ''*, *comment: str = ''*, *see_alsos: ~typing.List[str] = <factory>*) → None

`class scanpipe.pipes.spdx.Package`(*spdx_id: str*, *name: str*, *download_location: str = 'NOASSERTION'*, *license_declared: str = 'NOASSERTION'*, *license_concluded: str = 'NOASSERTION'*, *copyright_text: str = 'NOASSERTION'*, *files_analyzed: bool = False*, *version: str = ''*, *supplier: str = ''*, *originator: str = ''*, *homepage: str = ''*, *filename: str = ''*, *description: str = ''*, *summary: str = ''*, *source_info: str = ''*, *release_date: str = ''*, *built_date: str = ''*, *valid_until_date: str = ''*, *primary_package_purpose: str = ''*, *comment: str = ''*, *license_comments: str = ''*, *checksums: ~typing.List[~scanpipe.pipes.spdx.Checksum] = <factory>*, *external_refs: ~typing.List[~scanpipe.pipes.spdx.ExternalRef] = <factory>*, *attribution_texts: ~typing.List[str] = <factory>*)

Packages referenced in the SPDX document.

`as_dict()`

Return the data as a serializable dict.

static date_to_iso(*date_str*)

Convert a provided *date_str* to the SPDX format: *YYYY-MM-DDThh:mm:ssZ*.

__init__(*spdx_id: str*, *name: str*, *download_location: str = 'NOASSERTION'*, *license_declared: str = 'NOASSERTION'*, *license_concluded: str = 'NOASSERTION'*, *copyright_text: str = 'NOASSERTION'*, *files_analyzed: bool = False*, *version: str = ''*, *supplier: str = ''*, *originator: str = ''*, *homepage: str = ''*, *filename: str = ''*, *description: str = ''*, *summary: str = ''*, *source_info: str = ''*, *release_date: str = ''*, *built_date: str = ''*, *valid_until_date: str = ''*, *primary_package_purpose: str = ''*, *comment: str = ''*, *license_comments: str = ''*, *checksums: ~typing.List[~scanpipe.pipes.spdx.Checksum] = <factory>*, *external_refs: ~typing.List[~scanpipe.pipes.spdx.ExternalRef] = <factory>*, *attribution_texts: ~typing.List[str] = <factory>*) → None

class scanpipe.pipes.spdx.**File**(*spdx_id: str*, *name: str*, *checksums: ~typing.List[~scanpipe.pipes.spdx.Checksum] = <factory>*, *license_concluded: str = 'NOASSERTION'*, *copyright_text: str = 'NOASSERTION'*, *license_in_files: ~typing.List[str] = <factory>*, *contributors: ~typing.List[str] = <factory>*, *notice_text: str = ''*, *types: ~typing.List[str] = <factory>*, *attribution_texts: ~typing.List[str] = <factory>*, *comment: str = ''*, *license_comments: str = ''*)

Files referenced in the SPDX document.

as_dict()

Return the data as a serializable dict.

__init__(*spdx_id: str*, *name: str*, *checksums: ~typing.List[~scanpipe.pipes.spdx.Checksum] = <factory>*, *license_concluded: str = 'NOASSERTION'*, *copyright_text: str = 'NOASSERTION'*, *license_in_files: ~typing.List[str] = <factory>*, *contributors: ~typing.List[str] = <factory>*, *notice_text: str = ''*, *types: ~typing.List[str] = <factory>*, *attribution_texts: ~typing.List[str] = <factory>*, *comment: str = ''*, *license_comments: str = ''*) → None

class scanpipe.pipes.spdx.**Relationship**(*spdx_id: str*, *related_spdx_id: str*, *relationship: str*, *comment: str = ''*)

Represent the relationship between two SPDX elements. For example, you can represent a relationship between two different Files, between a Package and a File, between two Packages, or between one SPDXDocument and another SPDXDocument.

as_dict()

Return the SPDX relationship as a serializable dict.

__init__(*spdx_id: str*, *related_spdx_id: str*, *relationship: str*, *comment: str = ''*) → None

class scanpipe.pipes.spdx.**Document**(*name: str, namespace: str, creation_info: ~scanpipe.pipes.spdx.CreationInfo, packages: ~typing.List[~scanpipe.pipes.spdx.Package], spdx_id: str = 'SPDXRef-DOCUMENT', version: str = '2.3', data_license: str = 'CC0-1.0', comment: str = '', files: ~typing.List[~scanpipe.pipes.spdx.File] = <factory>, extracted_licenses: ~typing.List[~scanpipe.pipes.spdx.ExtractedLicensingInfo] = <factory>, relationships: ~typing.List[~scanpipe.pipes.spdx.Relationship] = <factory>*)

Collection of section instances each of which contains information about software organized using the SPDX format.

as_dict()

Return the SPDX document as a serializable dict.

as_json(*indent=2*)

Return the SPDX document as serialized JSON.

static **safe_document_name**(*name*)

Convert provided *name* to a safe SPDX document name.

validate(*schema*)

Check the validity of this SPDX document.

__init__(*name: str, namespace: str, creation_info: ~scanpipe.pipes.spdx.CreationInfo, packages: ~typing.List[~scanpipe.pipes.spdx.Package], spdx_id: str = 'SPDXRef-DOCUMENT', version: str = '2.3', data_license: str = 'CC0-1.0', comment: str = '', files: ~typing.List[~scanpipe.pipes.spdx.File] = <factory>, extracted_licenses: ~typing.List[~scanpipe.pipes.spdx.ExtractedLicensingInfo] = <factory>, relationships: ~typing.List[~scanpipe.pipes.spdx.Relationship] = <factory>*) → None

scanpipe.pipes.spdx.**validate_document**(*document*, *schema=PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/scancodeio/e packages/scanpipe/pipes/schemas/spdx-schema-2.3.json')*)

SPDX document validation. Requires the *jsonschema* library.

scanpipe.pipes.spdx.**is_spdx_document**(*input_location*)

Return True if the file at *input_location* is a SPDX Document.

18.21 Symbols

exception scanpipe.pipes.symbols.**UniversalCtagsNotFound**

scanpipe.pipes.symbols.**collect_and_store_resource_symbols**(*project*, *logger=None*)

Collect symbols from codebase files using Ctags and store them in the extra data field.

scanpipe.pipes.symbols.**collect_and_store_pygments_symbols_and_strings**(*project*, *logger=None*)

Collect symbols, strings and comments from codebase files using pygments and store them in the extra data field.

scanpipe.pipes.symbols.**collect_and_store_tree_sitter_symbols_and_strings**(*project*, *logger=None*)

Collect symbols from codebase files using tree-sitter and store them in the extra data field.

18.22 VulnerableCode

scanpipe.pipes.vulnerablecode.**is_configured**()

Return True if the required VulnerableCode settings have been set.

scanpipe.pipes.vulnerablecode.**is_available**()

Return True if the configured VulnerableCode server is available.

`scanpipe.pipes.vulnerablecode.`**`chunked`**(*iterable*, *chunk_size*)

Break an *iterable* into lists of *chunk_size* length.

```
>>> list(chunked([1, 2, 3, 4, 5], 2))
[[1, 2], [3, 4], [5]]
>>> list(chunked([1, 2, 3, 4, 5], 3))
[[1, 2, 3], [4, 5]]
```

`scanpipe.pipes.vulnerablecode.`**`get_purls`**(*packages*)

Return the PURLs for the given list of *packages*.

`scanpipe.pipes.vulnerablecode.`**`request_get`**(*url*, *payload=None*, *timeout=None*)

Wrap the HTTP request calls on the API.

`scanpipe.pipes.vulnerablecode.`**`get_vulnerabilities_by_purl`**(*purl*, *timeout=None*, *api_url=None*)

Get the list of vulnerabilities providing a package *purl*.

`scanpipe.pipes.vulnerablecode.`**`get_vulnerabilities_by_cpe`**(*cpe*, *timeout=None*, *api_url=None*)

Get the list of vulnerabilities providing a package or component *cpe*.

`scanpipe.pipes.vulnerablecode.`**`bulk_search_by_purl`**(*purls*, *timeout=None*, *api_url=None*)

Bulk search of vulnerabilities using the provided list of *purls*.

`scanpipe.pipes.vulnerablecode.`**`bulk_search_by_cpes`**(*cpes*, *timeout=None*, *api_url=None*)

Bulk search of vulnerabilities using the provided list of *cpes*.

`scanpipe.pipes.vulnerablecode.`**`fetch_vulnerabilities`**(*packages*, *chunk_size=1000*, *logger=<bound method Logger.info of <Logger scanpipe.pipes.vulnerablecode (INFO)>>*)

Fetch and store vulnerabilities for each provided `packages`. The PURLs are used for the lookups in batch of `chunk_size` per request.

18.23 Windows

`scanpipe.pipes.windows.`**`package_getter`**(*root_dir*, ***kwargs*)

Return installed package objects.

`scanpipe.pipes.windows.`**`flag_uninteresting_windows_codebase_resources`**(*project*)

Flag known uninteresting files as uninteresting.

`scanpipe.pipes.windows.`**`flag_installed_package_files`**(*project*, *root_dir_pattern*, *package*, *q_objects=None*)

For all CodebaseResources from *project* whose *rootfs_path* starts with *root_dir_pattern*, add *package* to the discovered_packages of each CodebaseResource and set the status.

`scanpipe.pipes.windows.`**`flag_known_software`**(*project*)

Find Windows software in *project* by checking CodebaseResources to see if their rootfs_path is under a known software root directory. If there are CodebaseResources that are under a known software root directory, a DiscoveredPackage is created for that software package and all files under that software package's root directory are considered installed files for that package.

Currently, we are only checking for Python and openjdk in Windows Docker image layers.

If a version number cannot be determined for an installed software Package, then a version number of "nv" will be set.

`scanpipe.pipes.windows.`**`flag_program_files`**(*project*)

Report all subdirectories of Program Files and Program Files (x86) as Packages.

If a Package is detected in this manner, then we will attempt to determine the version from the path. If a version cannot be determined, a version of *nv* will be set for the Package.

CHAPTER

NINETEEN

PROJECT CONFIGURATION

You have two options for configuring your project: using the user interface settings or providing a `scancode-config.yml` configuration file.

To configure your project using the user interface, refer to the instructions in the *Project Settings* section.

Alternatively, you can provide a `scancode-config.yml` configuration file as part of the project inputs with the `--input-file` option when using the *Command Line Interface*.

Tip: You can generate the project configuration file from the *Project Settings* UI.

CHAPTER TWENTY

DATA MODELS

This section is a collection of concepts or notations for describing the structure of the ScanCode.io Data Model and providing details about all fields included in the output files.

20.1 Project

`class scanpipe.models.Project`

The Project encapsulates all analysis processing. Multiple analysis pipelines can be run on the same project.

Parameters

- **uuid** (*UUIDField*) – Primary key: UUID
- **extra_data** (*JSONField*) – Extra data. Optional mapping of extra data key/values.
- **created_date** (*DateTimeField*) – Created date. Creation date for this project.
- **name** (*CharField*) – Name. Name for this project.
- **slug** (*SlugField*) – Slug
- **work_directory** (*CharField*) – Work directory. Project work directory location.
- **is_archived** (*BooleanField*) – Is archived. Archived projects cannot be modified anymore and are not displayed by default in project lists. Multiple levels of data cleanup may have happened during the archive operation.
- **notes** (*TextField*) – Notes
- **settings** (*JSONField*) – Settings

Relationship fields:

Parameters

- **labels** (`TaggableManager` to `Tag`) – Tags. A comma-separated list of tags. (related name: `project`)
- **tagged_items** (`GenericRelation` to `UUIDTaggedItem`) – Tagged items (related name: +)

Reverse relationships:

Parameters

- **projectmessages** (Reverse `ForeignKey` from *ProjectMessage*) – All projectmessages of this project (related name of *project*)

- **inputsources** (Reverse ForeignKey from InputSource) – All inputsources of this project (related name of project)
- **runs** (Reverse ForeignKey from *Run*) – All runs of this project (related name of *project*)
- **codebaseresources** (Reverse ForeignKey from *CodebaseResource*) – All codebaseresources of this project (related name of *project*)
- **codebaserelations** (Reverse ForeignKey from *CodebaseRelation*) – All codebaserelations of this project (related name of *project*)
- **discoveredpackages** (Reverse ForeignKey from *DiscoveredPackage*) – All discoveredpackages of this project (related name of *project*)
- **discovereddependencies** (Reverse ForeignKey from *DiscoveredDependency*) – All discovereddependencies of this project (related name of *project*)
- **webhooksubscriptions** (Reverse ForeignKey from WebhookSubscription) – All webhooksubscriptions of this project (related name of project)

add_downloads(*downloads*)

Move the given *downloads* to the current project's input/ directory and adds the *input_source* for each entry.

add_error(*description=''*, *model=''*, *details=None*, *exception=None*, *resource=None*)

Create an ERROR ProjectMessage record using for this project.

add_info(*description=''*, *model=''*, *details=None*, *exception=None*, *resource=None*)

Create an INFO ProjectMessage record for this project.

add_input_source(*download_url=''*, *filename=''*, *is_uploaded=False*, *tag=''*)

Create a InputFile entry for the current project, given a *download_url* or a *filename*.

add_message(*severity*, *description=''*, *model=''*, *details=None*, *exception=None*, *resource=None*)

Create a ProjectMessage record for this Project.

The `model` attribute can be provided as a string or as a Model class. A `resource` can be provided to keep track of the codebase resource that was analyzed when the error occurred.

add_pipeline(*pipeline_name*, *execute_now=False*, *selected_groups=None*)

Create a new Run instance with the provided *pipeline* on the current project.

If *execute_now* is True, the pipeline task is created. on_commit() is used to postpone the task creation after the transaction is successfully committed. If there isn't any active transactions, the callback will be executed immediately.

add_upload(*uploaded_file*, *tag=''*)

Write the given *upload* to the current project's input/ directory and adds the *input_source*.

add_uploads(*uploads*)

Write the given *uploads* to the current project's input/ directory and adds the *input_source* for each entry.

add_warning(*description=''*, *model=''*, *details=None*, *exception=None*, *resource=None*)

Create a WARNING ProjectMessage record for this project.

add_webhook_subscription(*target_url*)

Create a new WebhookSubscription instance with the provided *target_url* for the current project.

archive(*remove_input=False*, *remove_codebase=False*, *remove_output=False*)

Set the project *is_archived* field to True.

The *remove_input*, *remove_codebase*, and *remove_output* can be provided during the archive operation to delete the related work directories.

The project cannot be archived if one of its related run is queued or already running.

`clear_tmp_directory()`

Delete the whole content of the tmp/ directory. This is called at the end of each pipeline Run, and it doesn't store any content that might be needed for further processing in following pipeline Run.

`clone`(*clone_name*, *copy_inputs=False*, *copy_pipelines=False*, *copy_settings=False*, *copy_subscriptions=False*, *execute_now=False*)

Clone this project using the provided `clone_name` as new project name.

`copy_input_from`(*input_location*)

Copy the file at *input_location* to the current project's input/ directory.

`delete`(**args*, ***kwargs*)

Delete the *work_directory* along project-related data in the database.

`delete_related_objects()`

Delete all related object instances using the private *_raw_delete* model API. This bypass the objects collection, cascade deletions, and signals. It results in a much faster objects deletion, but it needs to be applied in the correct models order as the cascading event will not be triggered. Note that this approach is used in Django's *fast_deletes* but the scanpipe models are cannot be fast-deleted as they have cascades and relations.

`get_codebase_config_directory()`

Return the `.scancode` config directory if available in the *codebase* directory.

`get_enabled_settings()`

Return the enabled settings with non-empty values.

`get_env`(*field_name=None*)

Return the project environment loaded from the `.scancode/config.yml` config file, when available, and overriden by the `settings` model field.

`field_name` can be provided to get a single entry from the env.

`get_input_config_file()`

Return the `scancode-config.yml` file from the input/ directory if available.

`get_inputs_with_source()`

Return an input list including the filename, download_url, and size data.

`get_latest_output`(*filename*)

Return the latest output file with the "filename" prefix, for example "scancode-<timestamp>.json".

`get_next_run()`

Return the next non-executed Run instance assigned to current project.

`get_output_file_path`(*name*, *extension*)

Return a crafted file path in the project output/ directory using given *name* and *extension*. The current date and time strings are added to the filename.

This method ensures the proper setup of the work_directory in case of a manual wipe and re-creates the missing pieces of the directory structure.

`get_output_files_info()`

Return files form the output work directory including the name and size.

`static get_root_content`(*directory*)

Return a list of all files and directories of a given *directory*. Only the first level children will be listed.

`get_settings_as_yml()`

Return the `settings` file content as yml, suitable for a config file.

`inputs`(*pattern='**/*'*, *extensions=None*)

Return all files and directories path of the input/ directory matching a given *pattern*. The default **/* pattern means "this directory and all subdirectories, recursively". Use the * pattern to only list the root content. The returned paths can be limited to the provided list of `extensions`.

`move_input_from`(*input_location*)

Move the file at *input_location* to the current project's input/ directory.

`reset`(*keep_input=True*)

Reset the project by deleting all related database objects and all work directories except the input directory—when the *keep_input* option is True.

`save`(**args*, ***kwargs*)

Save this project instance. The workspace directories are set up during project creation.

`setup_work_directory()`

Create all the work_directory structure and skips if already existing.

`start_pipelines()`

Start the next "not started" pipeline execution.

`walk_codebase_path()`

Return files and directories path of the codebase/ directory recursively.

`write_input_file`(*file_object*)

Write the provided *file_object* to the project's input/ directory.

`WORK_DIRECTORIES = ['input', 'output', 'codebase', 'tmp']`

`can_change_inputs`

Return True until one pipeline run has started its execution on the project. Always return False when the project is archived.

`can_start_pipelines`

Return True if at least one "not started" pipeline is assigned to this project and if no pipeline runs is currently "queued or running". "not started". Always return False when the project is archived.

`property codebase_path`

Return the *codebase* directory as a Path instance.

`codebaserelations`

Type: Reverse `ForeignKey` from *`CodebaseRelation`*

All codebaserelations of this project (related name of *`project`*)

`codebaseresources`

Type: Reverse `ForeignKey` from *`CodebaseResource`*

All codebaseresources of this project (related name of *`project`*)

`created_date`

Type: `DateTimeField`

Created date. Creation date for this project.

`dependency_count`
Return the number of dependencies related to this project.

`discovereddependencies`
Type: Reverse `ForeignKey` from *`DiscoveredDependency`*

All discovereddependencies of this project (related name of *`project`*)

`discoveredpackages`
Type: Reverse `ForeignKey` from *`DiscoveredPackage`*

All discoveredpackages of this project (related name of *`project`*)

`extra_data`
Type: `JSONField`

Extra data. Optional mapping of extra data key/values.

`file_count`
Return the number of **file** resources related to this project.

`file_in_package_count`
Return the number of **file** resources **in a package** related to this project.

`file_not_in_package_count`
Return the number of **file** resources **not in a package** related to this project.

`has_single_resource`
Return True if we only have a single CodebaseResource associated to this project, False otherwise.

`property input_files`
Return list of files' relative paths in the input/ directory recursively.

`property input_path`
Return the *input* directory as a Path instance.

`property input_root`
Return a list of all files and directories of the input/ directory. Only the first level children will be listed.

`property input_sources`

`inputsources`
Type: Reverse `ForeignKey` from `InputSource`

All inputsources of this project (related name of `project`)

`is_archived`
Type: `BooleanField`

Is archived. Archived projects cannot be modified anymore and are not displayed by default in project lists. Multiple levels of data cleanup may have happened during the archive operation.

`labels = <taggit.managers._TaggableManager object>`

`message_count`
Return the number of messages related to this project.

`name`
Type: `CharField`

Name. Name for this project.

notes

Type: `TextField`

Notes

property output_path

Return the *output* directory as a Path instance.

property output_root

Return a list of all files and directories of the output/ directory. Only first level children will be listed.

package_count

Return the number of packages related to this project.

projectmessages

Type: Reverse `ForeignKey` from *`ProjectMessage`*

All projectmessages of this project (related name of *`project`*)

relation_count

Return the number of relations related to this project.

resource_count

Return the number of resources related to this project.

runs

Type: Reverse `ForeignKey` from *`Run`*

All runs of this project (related name of *`project`*)

settings

Type: `JSONField`

Settings

slug

Type: `SlugField`

Slug

tagged_items

Type: Reverse `GenericRelation` from *`Project`*

All + of this Label (related name of *`tagged_items`*)

property tmp_path

Return the *tmp* directory as a Path instance.

uuid

Type: `UUIDField`

Primary key: UUID

vulnerable_dependency_count

Return the number of vulnerable dependencies related to this project.

vulnerable_package_count

Return the number of vulnerable packages related to this project.

webhooksubscriptions

Type: Reverse `ForeignKey` from `WebhookSubscription`

All webhooksubscriptions of this project (related name of `project`)

work_directory

Type: `CharField`

Work directory. Project work directory location.

property work_path

Return the *work_directory* as a Path instance.

20.2 CodebaseResource

class scanpipe.models.**CodebaseResource**

A project Codebase Resources are records of its code files and directories. Each record is identified by its path under the project workspace.

These model fields should be kept in line with *commoncode.resource.Resource*.

Parameters

- **id** (*AutoField*) – Primary key: ID
- **md5** (*CharField*) – MD5. MD5 checksum hex-encoded, as in md5sum.
- **sha1** (*CharField*) – SHA1. SHA1 checksum hex-encoded, as in sha1sum.
- **sha256** (*CharField*) – SHA256. SHA256 checksum hex-encoded, as in sha256sum.
- **sha512** (*CharField*) – SHA512. SHA512 checksum hex-encoded, as in sha512sum.
- **extra_data** (*JSONField*) – Extra data. Optional mapping of extra data key/values.
- **detected_license_expression** (*TextField*) – Detected license expression. The license expression summarizing the license info for this resource, combined from all the license detections
- **detected_license_expression_spdx** (*TextField*) – Detected license expression spdx. The detected license expression for this file, with SPDX license keys
- **license_detections** (*JSONField*) – License detections. List of license detection details.
- **license_clues** (*JSONField*) – License clues. List of license matches that are not proper detections and potentially just clues to licenses or likely false positives. Those are not included in computing the detected license expression for the resource.
- **percentage_of_license_text** (*FloatField*) – Percentage of license text. Percentage of file words detected as license text or notice.
- **copyrights** (*JSONField*) – Copyrights. List of detected copyright statements (and related detection details).
- **holders** (*JSONField*) – Holders. List of detected copyright holders (and related detection details).
- **authors** (*JSONField*) – Authors. List of detected authors (and related detection details).
- **emails** (*JSONField*) – Emails. List of detected emails (and related detection details).
- **urls** (*JSONField*) – Urls. List of detected URLs (and related detection details).

- **compliance_alert** (*CharField*) – Compliance alert. Indicates how the license expression complies with provided policies.
- **path** (*CharField*) – Path. The full path value of a resource (file or directory) in the archive it is from.
- **rootfs_path** (*CharField*) – Rootfs path. Path relative to some root filesystem root directory. Useful when working on disk images, docker images, and VM images.Eg.: "/usr/bin/bash" for a path of "tarball-extract/rootfs/usr/bin/bash"
- **status** (*CharField*) – Status. Analysis status for this resource.
- **size** (*BigIntegerField*) – Size. Size in bytes.
- **tag** (*CharField*) – Tag
- **type** (*CharField*) – Type. Type of this resource as one of: file, directory, symlink
- **name** (*CharField*) – Name. File or directory name of this resource with its extension.
- **extension** (*CharField*) – Extension. File extension for this resource (directories do not have an extension).
- **programming_language** (*CharField*) – Programming language. Programming language of this resource if this is a code file.
- **mime_type** (*CharField*) – Mime type. MIME type (aka. media type) for this resource. See https://en.wikipedia.org/wiki/Media_type
- **file_type** (*CharField*) – File type. Descriptive file type for this resource.
- **is_binary** (*BooleanField*) – Is binary
- **is_text** (*BooleanField*) – Is text
- **is_archive** (*BooleanField*) – Is archive
- **is_key_file** (*BooleanField*) – Is key file
- **is_media** (*BooleanField*) – Is media
- **package_data** (*JSONField*) – Package data. List of Package data detected from this CodebaseResource

Relationship fields:

Parameters
project (ForeignKey to *Project*) – Project (related name: *codebaseresources*)

Reverse relationships:

Parameters

- **related_to** (Reverse ForeignKey from *CodebaseRelation*) – All related to of this codebase resource (related name of *from_resource*)
- **related_from** (Reverse ForeignKey from *CodebaseRelation*) – All related from of this codebase resource (related name of *to_resource*)
- **discovered_packages** (Reverse ManyToManyField from *DiscoveredPackage*) – All discovered packages of this codebase resource (related name of *codebase_resources*)
- **dependencies** (Reverse ForeignKey from *DiscoveredDependency*) – All dependencies of this codebase resource (related name of *datafile_resource*)

class Type(*value*, *names=None*, *, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

List of CodebaseResource types.

DIRECTORY = 'directory'

FILE = 'file'

SYMLINK = 'symlink'

add_package(*discovered_package*)

Assign the *discovered_package* to this *codebase_resource* instance.

as_spdx()

Return this CodebaseResource as an SPDX Package entry.

children(*codebase=None*)

Return a QuerySet of direct children CodebaseResource objects using a database query on the current CodebaseResource *path*.

Paths are returned in lower-cased sorted path order to reflect the behavior of the *commoncode.resource.Resource.children()* https://github.com/nexB/commoncode/blob/main/src/commoncode/resource.py

codebase is not used in this context but required for compatibility with the commoncode.resource.VirtualCodebase class API.

create_and_add_package(*package_data*)

Create a DiscoveredPackage instance using the *package_data* and assigns it to the current CodebaseResource instance.

Errors that may happen during the DiscoveredPackage creation are capture at this level, rather that in the DiscoveredPackage.create_from_data level, so resource data can be injected in the ProjectMessage record.

classmethod create_from_data(*project*, *resource_data*)

Create and returns a DiscoveredPackage for a *project* from the *package_data*. If one of the values of the required fields is not available, a "ProjectMessage" is created instead of a new DiscoveredPackage instance.

descendants()

Return a QuerySet of descendant CodebaseResource objects using a database query on the current CodebaseResource *path*. The current CodebaseResource is not included.

get_compliance_alert_display(*, *field=<django.db.models.CharField: compliance_alert>*)

Shows the label of the `compliance_alert`. See `get_FOO_display()` for more information.

get_path_segments_with_subpath()

Return a list of path segment name along its subpath for this resource.

Such as:

```
[
    ('root', 'root'),
    ('subpath', 'root/subpath'),
    ('file.txt', 'root/subpath/file.txt'),
]
```

get_raw_url()

Return the URL to access the RAW content of the resource.

`get_spdx_types()`

`get_type_display(*, field=<django.db.models.CharField: type>)`

Shows the label of the `type`. See `get_FOO_display()` for more information.

`has_parent()`

Return True if this CodebaseResource has a parent CodebaseResource or False otherwise.

`parent(codebase=None)`

Return the parent CodebaseResource object for this CodebaseResource or None.

codebase is not used in this context but required for compatibility with the commoncode.resource.Codebase class API.

`parent_path()`

Return the parent path for this CodebaseResource or None.

`siblings(codebase=None)`

Return a sequence of sibling Resource objects for this Resource or an empty sequence.

codebase is not used in this context but required for compatibility with the commoncode.resource.Codebase class API.

`walk(topdown=True)`

Return all descendant Resources of the current Resource; does not include self.

Traverses the tree top-down, depth-first if *topdown* is True; otherwise traverses the tree bottom-up.

`authors`

Type: `JSONField`

Authors. List of detected authors (and related detection details).

`compliance_alert`

Type: `CharField`

Compliance alert. Indicates how the license expression complies with provided policies.

Choices:

- `ok`
- `warning`
- `error`
- `missing`

`copyrights`

Type: `JSONField`

Copyrights. List of detected copyright statements (and related detection details).

`dependencies`

Type: Reverse `ForeignKey` from `DiscoveredDependency`

All dependencies of this codebase resource (related name of `datafile_resource`)

`detected_license_expression`

Type: `TextField`

Detected license expression. The license expression summarizing the license info for this resource, combined from all the license detections

detected_license_expression_spdx

Type: `TextField`

Detected license expression spdx. The detected license expression for this file, with SPDX license keys

discovered_packages

Type: Reverse `ManyToManyField` from *`DiscoveredPackage`*

All discovered packages of this codebase resource (related name of *`codebase_resources`*)

emails

Type: `JSONField`

Emails. List of detected emails (and related detection details).

extension

Type: `CharField`

Extension. File extension for this resource (directories do not have an extension).

extra_data

Type: `JSONField`

Extra data. Optional mapping of extra data key/values.

property file_content

Return the content of the current Resource file using TextCode utilities for optimal compatibility.

file_type

Type: `CharField`

File type. Descriptive file type for this resource.

property for_packages

Return the list of all discovered packages associated to this resource.

holders

Type: `JSONField`

Holders. List of detected copyright holders (and related detection details).

id

Type: `AutoField`

Primary key: ID

is_archive

Type: `BooleanField`

Is archive

is_binary

Type: `BooleanField`

Is binary

property is_dir

Return True, if the resource is a directory.

property is_file

Return True, if the resource is a file.

is_key_file

Type: BooleanField

Is key file

is_media

Type: BooleanField

Is media

property is_symlink

Return True, if the resource is a symlink.

is_text

Type: BooleanField

Is text

license_clues

Type: JSONField

License clues. List of license matches that are not proper detections and potentially just clues to licenses or likely false positives. Those are not included in computing the detected license expression for the resource.

license_detections

Type: JSONField

License detections. List of license detection details.

license_expression_field = 'detected_license_expression'

property location

Return the location of the resource as a string.

property location_path

Return the location of the resource as a Path instance.

md5

Type: CharField

MD5. MD5 checksum hex-encoded, as in md5sum.

mime_type

Type: CharField

Mime type. MIME type (aka. media type) for this resource. See https://en.wikipedia.org/wiki/Media_type

name

Type: CharField

Name. File or directory name of this resource with its extension.

property name_without_extension

Return the name of the resource without it's extension.

package_data

Type: JSONField

Package data. List of Package data detected from this CodebaseResource

path

Type: `CharField`

Path. The full path value of a resource (file or directory) in the archive it is from.

percentage_of_license_text

Type: `FloatField`

Percentage of license text. Percentage of file words detected as license text or notice.

programming_language

Type: `CharField`

Programming language. Programming language of this resource if this is a code file.

project

Type: `ForeignKey` to *`Project`*

Project (related name: *`codebaseresources`*)

project_id

Internal field, use *`project`* instead.

related_from

Type: Reverse `ForeignKey` from *`CodebaseRelation`*

All related from of this codebase resource (related name of *`to_resource`*)

related_to

Type: Reverse `ForeignKey` from *`CodebaseRelation`*

All related to of this codebase resource (related name of *`from_resource`*)

rootfs_path

Type: `CharField`

Rootfs path. Path relative to some root filesystem root directory. Useful when working on disk images, docker images, and VM images.Eg.: "/usr/bin/bash" for a path of "tarball-extract/rootfs/usr/bin/bash"

sha1

Type: `CharField`

SHA1. SHA1 checksum hex-encoded, as in sha1sum.

sha256

Type: `CharField`

SHA256. SHA256 checksum hex-encoded, as in sha256sum.

sha512

Type: `CharField`

SHA512. SHA512 checksum hex-encoded, as in sha512sum.

size

Type: `BigIntegerField`

Size. Size in bytes.

property spdx_id

status

Type: `CharField`

Status. Analysis status for this resource.

tag

Type: `CharField`

Tag

type

Type: `CharField`

Type. Type of this resource as one of: file, directory, symlink

Choices:

- `file`
- `directory`
- `symlink`

urls

Type: `JSONField`

Urls. List of detected URLs (and related detection details).

20.3 DiscoveredPackage

class scanpipe.models.DiscoveredPackage

A project's Discovered Packages are records of the system and application packages discovered in the code under analysis. Each record is identified by its Package URL. Package URL is a fundamental effort to create informative identifiers for software packages, such as Debian, RPM, npm, Maven, or PyPI packages. See https://github.com/package-url for more details.

Parameters

- **id** (*AutoField*) – Primary key: ID
- **type** (*CharField*) – Type. A short code to identify the type of this package. For example: gem for a Rubygem, docker for a container, pypi for a Python Wheel or Egg, maven for a Maven Jar, deb for a Debian package, etc.
- **namespace** (*CharField*) – Namespace. Package name prefix, such as Maven groupid, Docker image owner, GitHub user or organization, etc.
- **name** (*CharField*) – Name. Name of the package.
- **version** (*CharField*) – Version. Version of the package.
- **qualifiers** (*CharField*) – Qualifiers. Extra qualifying data for a package such as the name of an OS, architecture, distro, etc.
- **subpath** (*CharField*) – Subpath. Extra subpath within a package, relative to the package root.
- **md5** (*CharField*) – MD5. MD5 checksum hex-encoded, as in md5sum.
- **sha1** (*CharField*) – SHA1. SHA1 checksum hex-encoded, as in sha1sum.
- **sha256** (*CharField*) – SHA256. SHA256 checksum hex-encoded, as in sha256sum.

- **sha512** (*CharField*) – SHA512. SHA512 checksum hex-encoded, as in sha512sum.
- **extra_data** (*JSONField*) – Extra data. Optional mapping of extra data key/values.
- **compliance_alert** (*CharField*) – Compliance alert. Indicates how the license expression complies with provided policies.
- **affected_by_vulnerabilities** (*JSONField*) – Affected by vulnerabilities
- **filename** (*CharField*) – Filename. File name of a Resource sometimes part of the URI properand sometimes only available through an HTTP header.
- **primary_language** (*CharField*) – Primary language. Primary programming language.
- **description** (*TextField*) – Description. Description for this package. By convention the first line should be a summary when available.
- **release_date** (*DateField*) – Release date. The date that the package file was created, or when it was posted to its original download source.
- **homepage_url** (*CharField*) – Homepage URL. URL to the homepage for this package.
- **download_url** (*CharField*) – Download URL. A direct download URL.
- **size** (*BigIntegerField*) – Size. Size in bytes.
- **bug_tracking_url** (*CharField*) – Bug tracking URL. URL to the issue or bug tracker for this package.
- **code_view_url** (*CharField*) – Code view URL. a URL where the code can be browsed online.
- **vcs_url** (*CharField*) – VCS URL. A URL to the VCS repository in the SPDX form of: "git", "svn", "hg", "bzr", "cvs", https://github.com/nexb/scancode-toolkit.git@405aaa4b3 See SPDX specification "Package Download Location" at https://spdx.org/spdx-specification-21-web-version#h.49x2ik5
- **repository_homepage_url** (*CharField*) – Repository homepage URL. URL to the page for this package in its package repository. This is typically different from the package homepage URL proper.
- **repository_download_url** (*CharField*) – Repository download URL. Download URL to download the actual archive of code of this package in its package repository. This may be different from the actual download URL.
- **api_data_url** (*CharField*) – API data URL. API URL to obtain structured data for this package such as the URL to a JSON or XML api its package repository.
- **copyright** (*TextField*) – Copyright. Copyright statements for this package. Typically one per line.
- **holder** (*TextField*) – Holder. Holders for this package. Typically one per line.
- **declared_license_expression** (*TextField*) – Declared license expression. The license expression for this package typically derived from its extracted_license_statement or from some other type-specific routine or convention.
- **declared_license_expression_spdx** (*TextField*) – Declared license expression spdx. The SPDX license expression for this package converted from its declared_license_expression.
- **license_detections** (*JSONField*) – License detections. A list of LicenseDetection mappings typically derived from its extracted_license_statement or from some other type-specific routine or convention.

- **other_license_expression** (*TextField*) – Other license expression. The license expression for this package which is different from the declared_license_expression, (i.e. not the primary license) routine or convention.
- **other_license_expression_spdx** (*TextField*) – Other license expression spdx. The other SPDX license expression for this package converted from its other_license_expression.
- **other_license_detections** (*JSONField*) – Other license detections. A list of LicenseDetection mappings which is different from the declared_license_expression, (i.e. not the primary license) These are detections for the detection for the license expressions in other_license_expression.
- **extracted_license_statement** (*TextField*) – Extracted license statement. The license statement mention, tag or text as found in a package manifest and extracted. This can be a string, a list or dict of strings possibly nested, as found originally in the manifest.
- **notice_text** (*TextField*) – Notice text. A notice text for this package.
- **datasource_ids** (*JSONField*) – Datasource ids. The identifiers for the datafile handlers used to obtain this package.
- **datafile_paths** (*JSONField*) – Datafile paths. A list of Resource paths for package datafiles which were used to assemble this pacakage.
- **file_references** (*JSONField*) – File references. List of file paths and details for files referenced in a package manifest. These may not actually exist on the filesystem. The exact semantics and base of these paths is specific to a package type or datafile format.
- **parties** (*JSONField*) – Parties. A list of parties such as a person, project or organization.
- **uuid** (*UUIDField*) – UUID
- **missing_resources** (*JSONField*) – Missing resources
- **modified_resources** (*JSONField*) – Modified resources
- **package_uid** (*CharField*) – Package uid. Unique identifier for this package.
- **keywords** (*JSONField*) – Keywords
- **source_packages** (*JSONField*) – Source packages
- **tag** (*CharField*) – Tag

Relationship fields:

Parameters

- **project** (ForeignKey to *Project*) – Project (related name: *discoveredpackages*)
- **codebase_resources** (ManyToManyField to *CodebaseResource*) – Codebase resources (related name: *discovered_packages*)

Reverse relationships:

Parameters
dependencies (Reverse ForeignKey from *DiscoveredDependency*) – All dependencies of this discovered package (related name of *for_package*)

add_resources(*codebase_resources*)

Assign the *codebase_resources* to this *discovered_package* instance.

as_cyclonedx()

Return this DiscoveredPackage as an CycloneDX Component entry.

as_spdx()

Return this DiscoveredPackage as an SPDX Package entry.

classmethod clean_data(*data*)

Return the *data* dict keeping only entries for fields available in the model.

classmethod create_from_data(*project*, *package_data*)

Create and returns a DiscoveredPackage for a *project* from the *package_data*. If one of the values of the required fields is not available, a "ProjectMessage" is created instead of a new DiscoveredPackage instance.

classmethod extract_purl_data(*package_data*)

get_compliance_alert_display(**, field=<django.db.models.CharField: compliance_alert>*)

Shows the label of the `compliance_alert`. See `get_FOO_display()` for more information.

get_declared_license_expression()

Return this package license expression.

Use *declared_license_expression* when available or compute the expression from *declared_license_expression_spdx*.

get_declared_license_expression_spdx()

Return this package license expression using SPDX keys.

Use *declared_license_expression_spdx* when available or compute the expression from *declared_license_expression*.

affected_by_vulnerabilities

Type: `JSONField`

Affected by vulnerabilities

api_data_url

Type: `CharField`

API data URL. API URL to obtain structured data for this package such as the URL to a JSON or XML api its package repository.

bug_tracking_url

Type: `CharField`

Bug tracking URL. URL to the issue or bug tracker for this package.

code_view_url

Type: `CharField`

Code view URL. a URL where the code can be browsed online.

codebase_resources

Type: `ManyToManyField` to `CodebaseResource`

Codebase resources (related name: `discovered_packages`)

compliance_alert

Type: `CharField`

Compliance alert. Indicates how the license expression complies with provided policies.

Choices:

- ok

- `warning`
- `error`
- `missing`

`copyright`

Type: `TextField`

Copyright. Copyright statements for this package. Typically one per line.

`datafile_paths`

Type: `JSONField`

Datafile paths. A list of Resource paths for package datafiles which were used to assemble this pacakage.

`datasource_ids`

Type: `JSONField`

Datasource ids. The identifiers for the datafile handlers used to obtain this package.

`declared_license_expression`

Type: `TextField`

Declared license expression. The license expression for this package typically derived from its extracted_license_statement or from some other type-specific routine or convention.

`declared_license_expression_spdx`

Type: `TextField`

Declared license expression spdx. The SPDX license expression for this package converted from its declared_license_expression.

`dependencies`

Type: Reverse `ForeignKey` from *`DiscoveredDependency`*

All dependencies of this discovered package (related name of *`for_package`*)

`description`

Type: `TextField`

Description. Description for this package. By convention the first line should be a summary when available.

`download_url`

Type: `CharField`

Download URL. A direct download URL.

`extra_data`

Type: `JSONField`

Extra data. Optional mapping of extra data key/values.

`extracted_license_statement`

Type: `TextField`

Extracted license statement. The license statement mention, tag or text as found in a package manifest and extracted. This can be a string, a list or dict of strings possibly nested, as found originally in the manifest.

`file_references`

Type: `JSONField`

File references. List of file paths and details for files referenced in a package manifest. These may not actually exist on the filesystem. The exact semantics and base of these paths is specific to a package type or datafile format.

`filename`

Type: `CharField`

Filename. File name of a Resource sometimes part of the URI properand sometimes only available through an HTTP header.

`holder`

Type: `TextField`

Holder. Holders for this package. Typically one per line.

`homepage_url`

Type: `CharField`

Homepage URL. URL to the homepage for this package.

`id`

Type: `AutoField`

Primary key: ID

`keywords`

Type: `JSONField`

Keywords

`license_detections`

Type: `JSONField`

License detections. A list of LicenseDetection mappings typically derived from its extracted_license_statement or from some other type-specific routine or convention.

`license_expression_field = 'declared_license_expression'`

`md5`

Type: `CharField`

MD5. MD5 checksum hex-encoded, as in md5sum.

`missing_resources`

Type: `JSONField`

Missing resources

`modified_resources`

Type: `JSONField`

Modified resources

`name`

Type: `CharField`

Name. Name of the package.

namespace

Type: `CharField`

Namespace. Package name prefix, such as Maven groupid, Docker image owner, GitHub user or organization, etc.

notice_text

Type: `TextField`

Notice text. A notice text for this package.

other_license_detections

Type: `JSONField`

Other license detections. A list of LicenseDetection mappings which is different from the declared_license_expression, (i.e. not the primary license) These are detections for the detection for the license expressions in other_license_expression.

other_license_expression

Type: `TextField`

Other license expression. The license expression for this package which is different from the declared_license_expression, (i.e. not the primary license) routine or convention.

other_license_expression_spdx

Type: `TextField`

Other license expression spdx. The other SPDX license expression for this package converted from its other_license_expression.

package_uid

Type: `CharField`

Package uid. Unique identifier for this package.

parties

Type: `JSONField`

Parties. A list of parties such as a person, project or organization.

primary_language

Type: `CharField`

Primary language. Primary programming language.

project

Type: `ForeignKey` to *`Project`*

Project (related name: *`discoveredpackages`*)

project_id

Internal field, use *`project`* instead.

property purl

Return the Package URL.

qualifiers

Type: `CharField`

Qualifiers. Extra qualifying data for a package such as the name of an OS, architecture, distro, etc.

release_date

Type: `DateField`

Release date. The date that the package file was created, or when it was posted to its original download source.

repository_download_url

Type: `CharField`

Repository download URL. Download URL to download the actual archive of code of this package in its package repository. This may be different from the actual download URL.

repository_homepage_url

Type: `CharField`

Repository homepage URL. URL to the page for this package in its package repository. This is typically different from the package homepage URL proper.

resources

Return the assigned codebase_resources QuerySet as a list.

sha1

Type: `CharField`

SHA1. SHA1 checksum hex-encoded, as in sha1sum.

sha256

Type: `CharField`

SHA256. SHA256 checksum hex-encoded, as in sha256sum.

sha512

Type: `CharField`

SHA512. SHA512 checksum hex-encoded, as in sha512sum.

size

Type: `BigIntegerField`

Size. Size in bytes.

source_packages

Type: `JSONField`

Source packages

property spdx_id

subpath

Type: `CharField`

Subpath. Extra subpath within a package, relative to the package root.

tag

Type: `CharField`

Tag

type

Type: `CharField`

Type. A short code to identify the type of this package. For example: gem for a Rubygem, docker for a container, pypi for a Python Wheel or Egg, maven for a Maven Jar, deb for a Debian package, etc.

uuid

Type: `UUIDField`

UUID

vcs_url

Type: `CharField`

VCS URL. A URL to the VCS repository in the SPDX form of: "git", "svn", "hg", "bzr", "cvs", https://github.com/nexb/scancode-toolkit.git@405aaa4b3 See SPDX specification "Package Download Location" at https://spdx.org/spdx-specification-21-web-version#h.49x2ik5

version

Type: `CharField`

Version. Version of the package.

20.4 DiscoveredDependency

class `scanpipe.models.`**`DiscoveredDependency`**

A project's Discovered Dependencies are records of the dependencies used by system and application packages discovered in the code under analysis.

Parameters

- **id** (*AutoField*) – Primary key: ID
- **type** (*CharField*) – Type. A short code to identify the type of this package. For example: gem for a Rubygem, docker for a container, pypi for a Python Wheel or Egg, maven for a Maven Jar, deb for a Debian package, etc.
- **namespace** (*CharField*) – Namespace. Package name prefix, such as Maven groupid, Docker image owner, GitHub user or organization, etc.
- **name** (*CharField*) – Name. Name of the package.
- **version** (*CharField*) – Version. Version of the package.
- **qualifiers** (*CharField*) – Qualifiers. Extra qualifying data for a package such as the name of an OS, architecture, distro, etc.
- **subpath** (*CharField*) – Subpath. Extra subpath within a package, relative to the package root.
- **affected_by_vulnerabilities** (*JSONField*) – Affected by vulnerabilities
- **dependency_uid** (*CharField*) – Dependency uid. The unique identifier of this dependency.
- **extracted_requirement** (*CharField*) – Extracted requirement. The version requirements of this dependency.
- **scope** (*CharField*) – Scope. The scope of this dependency, how it is used in a project.
- **datasource_id** (*CharField*) – Datasource id. The identifier for the datafile handler used to obtain this dependency.
- **is_runtime** (*BooleanField*) – Is runtime
- **is_optional** (*BooleanField*) – Is optional
- **is_resolved** (*BooleanField*) – Is resolved

Relationship fields:

Parameters

- **`project`** (`ForeignKey` to *`Project`*) – Project (related name: *`discovereddependencies`*)
- **`for_package`** (`ForeignKey` to *`DiscoveredPackage`*) – For package (related name: *`dependencies`*)
- **`datafile_resource`** (`ForeignKey` to *`CodebaseResource`*) – Datafile resource (related name: *`dependencies`*)

`as_spdx`()

Return this Dependency as an SPDX Package entry.

`classmethod create_from_data`(*project*, *dependency_data*, *for_package=None*, *datafile_resource=None*, *datasource_id=None*, *strip_datafile_path_root=False*)

Create and returns a DiscoveredDependency for a *project* from the *dependency_data*.

If *strip_datafile_path_root* is True, then *create_from_data()* will strip the root path segment from the *datafile_path* of *dependency_data* before looking up the corresponding CodebaseResource for *datafile_path*. This is used in the case where Dependency data is imported from a scancode-toolkit scan, where the root path segments are not stripped for *datafile_path*.

`affected_by_vulnerabilities`

Type: `JSONField`

Affected by vulnerabilities

`datafile_path`

`datafile_resource`

Type: `ForeignKey` to *`CodebaseResource`*

Datafile resource (related name: *`dependencies`*)

`datafile_resource_id`

Internal field, use *`datafile_resource`* instead.

`datasource_id`

Type: `CharField`

Datasource id. The identifier for the datafile handler used to obtain this dependency.

`dependency_uid`

Type: `CharField`

Dependency uid. The unique identifier of this dependency.

`extracted_requirement`

Type: `CharField`

Extracted requirement. The version requirements of this dependency.

`for_package`

Type: `ForeignKey` to *`DiscoveredPackage`*

For package (related name: *`dependencies`*)

`for_package_id`

Internal field, use *`for_package`* instead.

for_package_uid

id

Type: AutoField

Primary key: ID

is_optional

Type: BooleanField

Is optional

is_resolved

Type: BooleanField

Is resolved

is_runtime

Type: BooleanField

Is runtime

name

Type: CharField

Name. Name of the package.

namespace

Type: CharField

Namespace. Package name prefix, such as Maven groupid, Docker image owner, GitHub user or organization, etc.

property package_type

project

Type: ForeignKey to *Project*

Project (related name: *discovereddependencies*)

project_id

Internal field, use *project* instead.

property purl

qualifiers

Type: CharField

Qualifiers. Extra qualifying data for a package such as the name of an OS, architecture, distro, etc.

scope

Type: CharField

Scope. The scope of this dependency, how it is used in a project.

property spdx_id

subpath

Type: CharField

Subpath. Extra subpath within a package, relative to the package root.

type

Type: `CharField`

Type. A short code to identify the type of this package. For example: gem for a Rubygem, docker for a container, pypi for a Python Wheel or Egg, maven for a Maven Jar, deb for a Debian package, etc.

version

Type: `CharField`

Version. Version of the package.

20.5 CodebaseRelation

class scanpipe.models.**CodebaseRelation**

Relation between two CodebaseResource.

Parameters

- **uuid** (*UUIDField*) – Primary key: UUID
- **extra_data** (*JSONField*) – Extra data. Optional mapping of extra data key/values.
- **map_type** (*CharField*) – Map type

Relationship fields:

Parameters

- **project** (`ForeignKey` to *Project*) – Project (related name: `codebaserelations`)
- **from_resource** (`ForeignKey` to *CodebaseResource*) – From resource (related name: *related_to*)
- **to_resource** (`ForeignKey` to *CodebaseResource*) – To resource (related name: *related_from*)

extra_data

Type: `JSONField`

Extra data. Optional mapping of extra data key/values.

from_resource

Type: `ForeignKey` to *CodebaseResource*

From resource (related name: *related_to*)

from_resource_id

Internal field, use *from_resource* instead.

map_type

Type: `CharField`

Map type

project

Type: `ForeignKey` to *Project*

Project (related name: *codebaserelations*)

project_id

Internal field, use *project* instead.

property score

property status

to_resource

Type: `ForeignKey` to *`CodebaseResource`*

To resource (related name: *`related_from`*)

to_resource_id

Internal field, use *`to_resource`* instead.

uuid

Type: `UUIDField`

Primary key: UUID

20.6 ProjectMessage

class scanpipe.models.**ProjectMessage**

Stores messages such as errors and exceptions raised during a pipeline run.

Parameters

- **uuid** (*`UUIDField`*) – Primary key: UUID
- **severity** (*`CharField`*) – Severity. Severity level of the message.
- **description** (*`TextField`*) – Description. Description.
- **model** (*`CharField`*) – Model. Name of the model class.
- **details** (*`JSONField`*) – Details. Data that caused the error.
- **traceback** (*`TextField`*) – Traceback. Exception traceback.
- **created_date** (*`DateTimeField`*) – Created date

Relationship fields:

Parameters

project (`ForeignKey` to *`Project`*) – Project (related name: *`projectmessages`*)

class Severity(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

ERROR = 'error'

INFO = 'info'

WARNING = 'warning'

get_severity_display(*, *field=<django.db.models.CharField: severity>*)

Shows the label of the *`severity`*. See `get_FOO_display()` for more information.

created_date

Type: `DateTimeField`

Created date

`description`

Type: `TextField`

Description. Description.

`details`

Type: `JSONField`

Details. Data that caused the error.

`model`

Type: `CharField`

Model. Name of the model class.

`project`

Type: `ForeignKey` to *`Project`*

Project (related name: *`projectmessages`*)

`project_id`

Internal field, use *`project`* instead.

`severity`

Type: `CharField`

Severity. Severity level of the message.

Choices:

- `info`
- `warning`
- `error`

`traceback`

Type: `TextField`

Traceback. Exception traceback.

`uuid`

Type: `UUIDField`

Primary key: UUID

20.7 Run

`class scanpipe.models.`**`Run`**

The Database representation of a pipeline execution.

Parameters

- **uuid** (*UUIDField*) – Primary key: UUID
- **task_id** (*UUIDField*) – Task id
- **task_start_date** (*DateTimeField*) – Task start date
- **task_end_date** (*DateTimeField*) – Task end date
- **task_exitcode** (*IntegerField*) – Task exitcode

- **task_output** (*TextField*) – Task output
- **log** (*TextField*) – Log
- **pipeline_name** (*CharField*) – Pipeline name. Identify a registered Pipeline class.
- **created_date** (*DateTimeField*) – Created date
- **scancodeio_version** (*CharField*) – Scancodeio version
- **description** (*TextField*) – Description
- **current_step** (*CharField*) – Current step
- **selected_groups** (*JSONField*) – Selected groups

Relationship fields:

Parameters
`project` (`ForeignKey` to *Project*) – Project (related name: *runs*)

deliver_project_subscriptions()

Triggers related project webhook subscriptions.

execute_task_async()

Enqueues the pipeline execution task for an asynchronous execution.

get_diff_url()

Return a GitHub diff URL between this Run commit at the time of execution and the current commit of the ScanCode.io app instance. The URL is only returned if both commit are available and if they differ.

get_previous_runs()

Return all the previous Run instances regardless of their status.

make_pipeline_instance()

Return a pipelines instance using this Run pipeline_class.

profile(*print_results=False*)

Return computed execution times for each step in the current Run.

If *print_results* is provided, the results are printed to stdout.

set_current_step(*message*)

Set the `message` value on the `current_step` field. Truncate the value at 256 characters.

set_scancodeio_version()

Set the current ScanCode.io version on the `scancodeio_version` field.

start()

Start the pipeline execution when allowed or raised an exception.

sync_with_job()

Synchronise this Run instance with its related RQ Job. This is required when a Run gets out of sync with its Job, this can happen when the worker or one of its processes is killed, the Run status is not properly updated and may stay in a Queued or Running state forever. In case the Run is out of sync of its related Job, the Run status will be updated accordingly. When the run was in the queue, it will be enqueued again.

property can_start

Return True if this Run is allowed to start its execution.

Run are not allowed to start when any of their previous Run instances within the pipeline has not completed (not started, queued, or running). This is enforced to ensure the pipelines are run in a sequential order.

created_date

Type: `DateTimeField`

Created date

current_step

Type: `CharField`

Current step

description

Type: `TextField`

Description

log

Type: `TextField`

Log

property pipeline_class

Return this Run pipeline_class.

pipeline_name

Type: `CharField`

Pipeline name. Identify a registered Pipeline class.

project

Type: `ForeignKey` to *`Project`*

Project (related name: *`runs`*)

project_id

Internal field, use *`project`* instead.

scancodeio_version

Type: `CharField`

Scancodeio version

selected_groups

Type: `JSONField`

Selected groups

task_end_date

Type: `DateTimeField`

Task end date

task_exitcode

Type: `IntegerField`

Task exitcode

task_id

Type: `UUIDField`

Task id

`task_output`

Type: `TextField`

Task output

`task_start_date`

Type: `DateTimeField`

Task start date

`uuid`

Type: `UUIDField`

Primary key: UUID

CHAPTER

TWENTYONE

OUTPUT FILES

Whether you use the command line or the web application to run your scans, the generated results are available for review or export in **JSON**, **Excel (XLSX)**, **SPDX**, and **CycloneDX** file formats. You can also produce the **Attribution** as an HTML file.

Tip: Check our *Data Models* section for more details about all fields included in the output files.

21.1 Creating Output Files

21.1.1 Command Line

You can output the scan results using the `output` command while specifying the output file format with the `--format` option:

```
$ scanpipe output --project PROJECT --format {json,xlsx,spdx,cyclonedx,attribution}
```

Note: The previous command will output the scan results in a file format – as specified – with the output files created in the `PROJECT`'s *output/* directory. By default, JSON output files are created when no file format is given.

Warning: When running with Docker, ensure that the output files workspace is assigned to a volume to be accessible on the host machine.

To add local input files to a project using the *Command Line Interface*, additional arguments need to be passed to the `docker compose` command.

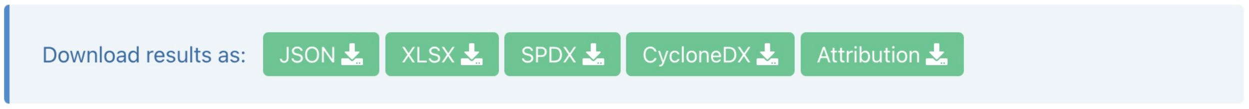
For example, using the following command will mount and make available the projects workspace on the host at `~/projects/`:

```
mkdir ~/projects/
docker compose run --volume ~/projects/:/var/scancodeio/workspace/projects/ \
    web scanpipe output --project my_project --format json
```

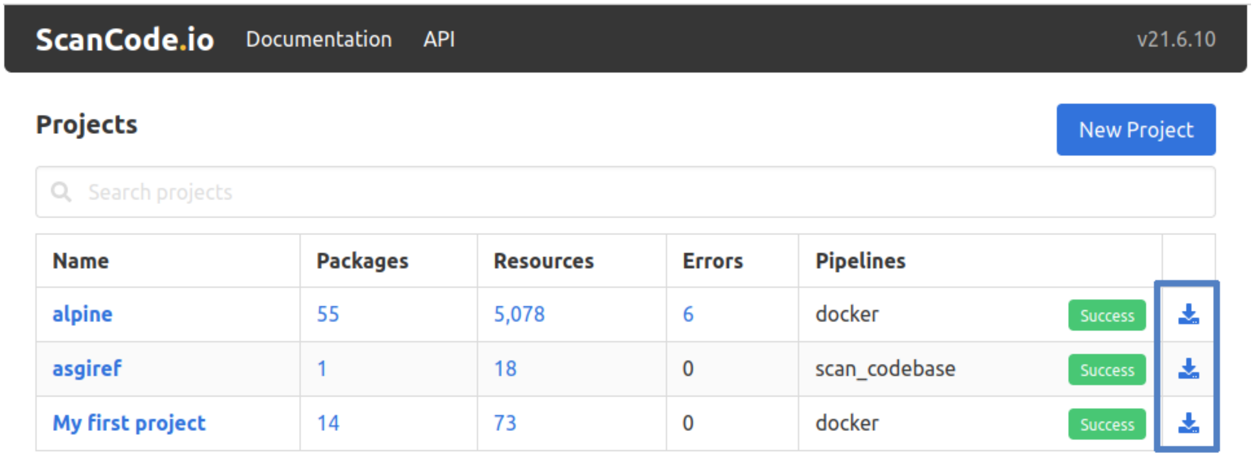
Alternatively, you can also locate the Docker volumes directory on your host machine. For instance, on Linux, it's typically found at: `/var/lib/docker/volumes/`.

21.1.2 Web UI

When using the ScanCode.io web application, you can download the results of your project in your preferred output format within the project page.



You can also download the generated results—for any existing project—from the ScanCode.io home screen.



Name	Packages	Resources	Errors	Pipelines		
alpine	55	5,078	6	docker	Success	
asgiref	1	18	0	scan_codebase	Success	
My first project	14	73	0	docker	Success	

21.2 Understanding Output Files

As previously mentioned, the output file format is set using the `--format` option to either JSON or XLSX data files. Regardless of the format, the data included in either output file remains almost the same.

21.2.1 JSON

The JSON file starts with some general information about the scan process, including the scan tool, scan date, input file details, pipeline used, etc., as shown below

```
{
  "headers": [
    {
      "tool_name": "scanpipe",
      "tool_version": "21.6.10",
      "notice": "Generated with ScanCode and provided on an \"AS IS\" BASIS, WITHOUT␣
↪WARRANTIES\nOR CONDITIONS OF ANY KIND, either express or implied. No content created␣
↪from\nScanCode should be considered or used as legal advice. Consult an Attorney\nfor␣
↪any legal advice.\nScanCode is a free software code scanning tool from nexB Inc. and␣
↪others.\nVisit https://github.com/nexB/scancode-toolkit/ for support and download.",
      "uuid": "f06e257e-3126-4220-87c7-13583ced38a0",
      "created_date": "2021-06-12T19:51:26.218Z",
      "input_files": [
        "30-alpine-nickolashkraus-staticbox-latest.tar"
      ],
```

(continues on next page)

(continued from previous page)

```
  "runs": [
    {
      "pipeline_name": "analyze_docker_image",
      "description": "A pipeline to analyze a Docker image.",
      "uuid": "5f1ec0c5-91ed-45c8-ab3d-beae44018716",
      "created_date": "2021-06-13T00:50:18.367560Z",
      "task_id": "e2085ee9-5804-4065-9a35-e25a883b8b62",
      "task_start_date": "2021-06-13T01:20:47.663939Z",
      "task_end_date": "2021-06-13T01:20:56.486136Z",
      "task_exitcode": 0,
      "task_output": "",
      "log": "2021-06-13 01:20:47.66 Pipeline [analyze_docker_image] starting\n2021-06-
↪13 01:20:47.66 Step [extract_images] starting\n2021-06-13 01:20:47.72 Step [extract_
↪images] completed in 0.05 seconds\n2021-06-13 01:20:47.72 Step [extract_layers]␣
↪starting\n2021-06-13 01:20:47.84 Step [extract_layers] completed in 0.12 seconds\n2021-
↪06-13 01:20:47.84 Step [find_images_linux_distro] starting\n2021-06-13 01:20:47.84␣
↪Step [find_images_linux_distro] completed in 0.00 seconds\n2021-06-13 01:20:47.85 Step␣
↪[collect_images_information] starting\n2021-06-13 01:20:47.85 Step [collect_images_
↪information] completed in 0.00 seconds\n2021-06-13 01:20:47.85 Step [collect_and_
↪create_codebase_resources] starting\n2021-06-13 01:20:48.65 Step [collect_and_create_
↪codebase_resources] completed in 0.79 seconds\n2021-06-13 01:20:48.65 Step [collect_
↪and_create_system_packages] starting\n2021-06-13 01:20:50.89 Step [collect_and_create_
↪system_packages] completed in 2.24 seconds\n2021-06-13 01:20:50.89 Step [flag_
↪uninteresting_codebase_resources] starting\n2021-06-13 01:20:50.90 Step [tag_
↪uninteresting_codebase_resources] completed in 0.00 seconds\n2021-06-13 01:20:50.90␣
↪Step [tag_empty_files] starting\n2021-06-13 01:20:50.91 Step [tag_empty_files]␣
↪completed in 0.00 seconds\n2021-06-13 01:20:50.91 Step [scan_for_application_packages]␣
↪starting\n2021-06-13 01:20:50.98 Step [scan_for_application_packages] completed in 0.
↪07 seconds\n2021-06-13 01:20:50.98 Step [scan_for_files] starting\n2021-06-13 01:20:56.
↪46 Step [scan_for_files] completed in 5.48 seconds\n2021-06-13 01:20:56.46 Step␣
↪[analyze_scanned_files] starting\n2021-06-13 01:20:56.47 Step [analyze_scanned_files]␣
↪completed in 0.00 seconds\n2021-06-13 01:20:56.47 Step [tag_not_analyzed_codebase_
↪resources] starting\n2021-06-13 01:20:56.48 Step [tag_not_analyzed_codebase_resources]␣
↪completed in 0.00 seconds\n2021-06-13 01:20:56.48 Pipeline completed\n",
      "execution_time": 8
    }
  ],
  "extra_data": {
    "images": [
      {
        "os": "linux",
        "tags": [
          "nickolashkraus/staticbox:latest"
        ],
        "author": null,
        "distro": {
          "os": "linux",
          "logo": null,
          "name": "Alpine Linux",
          "id_like": [],
          "variant": null,
          "version": null,
```

(continues on next page)

(continued from previous page)

```
          "build_id": null,
          "cpe_name": null,
          "home_url": "https://alpinelinux.org/",
          "extra_data": {},
          "identifier": "alpine",
          "variant_id": null,
          "version_id": "3.11.3",
          "pretty_name": "Alpine Linux v3.11",
          "support_url": null,
          "architecture": "amd64",
          "bug_report_url": "https://bugs.alpinelinux.org/",
          "version_codename": null,
          "documentation_url": null,
          "privacy_policy_url": null
        },
        "labels": {},
        "sha256": null,
        "comment": null,
        "created": "2020-02-04T20:14:21.37837804Z",
        "history": [
          {
            "created": "2020-01-18T01:19:37.02673981Z",
            "created_by": "/bin/sh -c #(nop) ADD␣
↪file:e69d441d729412d24675dcd33e04580885df99981cec43de8c9b24015313ff8e in / "
          },
          {
            "created": "2020-01-18T01:19:37.187497623Z",
            "created_by": "/bin/sh -c #(nop)  CMD [\"/bin/sh\"]",
            "empty_layer": true
          },
          {
            "created": "2020-02-04T20:14:18.651799654Z",
            "created_by": "/bin/sh -c #(nop) COPY␣
↪file:0534399d8928526e71db5a2dd096bfa0548c3ea036b678eb596a76d2ddc2bdbf in /staticbox/
↪bin/busybox "
          },
          {
            "created": "2020-02-04T20:14:20.986239348Z",
            "created_by": "/bin/sh -c for f in /bin/*; do if [[ -h $f  ]]; then ln -sf /
↪staticbox/bin/busybox /staticbox/bin/$(basename $f); fi done"
          },
          {
            "created": "2020-02-04T20:14:21.37837804Z",
            "created_by": "/bin/sh -c #(nop)  ENV PATH=/staticbox/bin:/usr/local/sbin:/
↪usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
            "empty_layer": true
          }
        ],
        "variant": null,
        "image_id": "7656d1f7594c21d805a02a8d7183506490949113 0ed7add6357b28d512f8d213",
        "os_version": null,
        "architecture": "amd64",
```

(continues on next page)

(continued from previous page)

```
        "image_format": "docker",
        "config_digest":
↪"sha256:7656d1f7594c21d805a02a8d7183506490949113 0ed7add6357b28d512f8d213",
        "docker_version": "18.03.1-ee-3"
      }
    ]
  }
}],
}
```

The JSON results file also lists information about any *packages* discovered during the scan process with information about each individual *package* similar to the following:

```
"packages": [
  {
    "purl": "pkg:alpine/musl@1.1.24-r0?arch=x86_64",
    "type": "alpine",
    "namespace": "",
    "name": "musl",
    "version": "1.1.24-r0",
    "qualifiers": "arch=x86_64",
    "subpath": "",
    "primary_language": "",
    "description": "the musl c library (libc) implementation",
    "release_date": "2019-11-15",
    "homepage_url": "http://www.musl-libc.org/",
    "download_url": "",
    "size": 376511,
    "sha1": "",
    "md5": "",
    "bug_tracking_url": "",
    "code_view_url": "",
    "vcs_url": "git+http://git.alpinelinux.org/aports/commit/?
↪id=ba05f40c20ddc515f748f205f01befbba3a88feb",
    "copyright": "",
    "license_expression": "mit",
    "declared_license": "MIT",
    "notice_text": "",
    "missing_resources": [
      "/lib/libc.musl-x86_64.so.1"
    ],
    "modified_resources": [],
    "keywords": [],
    "source_packages": [
      "pkg:alpine/musl@1.1.24-r0"
    ]
  }
]
```

The results will also include all of the or files (codebase resources) found.

Note: Please note that these files might or might not be included within a package.

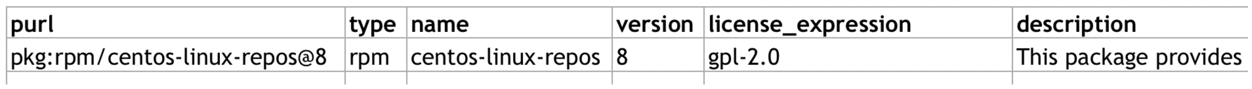
```
"files": [{
  "for_packages": [
    "pkg:alpine/busybox@1.31.1-r9?arch=x86_64"
  ],
  "compliance_alert": "",
  "path": "/30-alpine-nickolashkraus-staticbox-latest.tar-extract/
↪5216338b40a7b96416b8b9858974bbe4acc3096ee60acbc4dfb1ee02aecceb10/bin/busybox",
  "size": 841288,
  "sha1": "593739e717ef3e8833034614576e03d189be30a1",
  "md5": "0234c668c5c93317e3f055fdd44f0943",
  "copyrights": [],
  "holders": [],
  "authors": [],
  "licenses": [],
  "license_expressions": [],
  "emails": [],
  "urls": [],
  "status": "system-package",
  "type": "file",
  "extra_data": {},
  "name": "busybox",
  "extension": "",
  "programming_language": "",
  "mime_type": "application/x-pie-executable",
  "file_type": "ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically␣
↪linked, interpreter /lib/ld-musl-x86_64.so.1, stripped",
  "is_binary": true,
  "is_text": false,
  "is_archive": false
}]
```

21.2.2 Excel (XLSX)

ScanCode.io can produce the scan results in a .xlsx file format, which will include two Excel sheets for the Discovered Packages and the Codebase Resources.

Note: Unlike the JSON file, the XLSX output file does not include any general information about the scan process, tool, date, etc.

The **Discovered Packages** data sheet includes details about all packages found:



purl	type	name	version	license_expression	description
pkg:rpm/centos-linux-repos@8	rpm	centos-linux-repos	8	gpl-2.0	This package provides

while the **Codebase Resources** sheet includes information about each individual file:

path	status	type	size	mime_type	for_packages
/etc/centos-release	system-package	file	30	text/plain	pkg:rpm/centos-linux-release@8.3

21.2.3 Attribution

ScanCode.io can generate attribution notices of the discovered packages of a project. The output format is a HTML page.

The default template output can be customized providing your own template in the *.scancode* config directory *SCANCODEIO_CONFIG_DIR*.

You usually want to start with a copy of the default template available at `scanpipe/templates/scanpipe/attribution.html` and add your modifications.

You can then place your custom template file into the *.scancode* config directory in your input files, such as it will end up at `codebase/.scancode/templates/attribution.html` on extraction.

The following variable are available as the template context:

- `project`
- `packages`
- `licenses`

Refer to *Data Models* for the full details of available fields.

CHAPTER

TWENTYTWO

COMMAND LINE INTERFACE

The `scanpipe` command can be executed using the Docker Compose command line interface.

If the Docker Compose stack is already running, you can execute the command as follows:

```
docker compose exec -it web scanpipe COMMAND
```

If the ScanCode.io services are not currently running, you can use the following command:

```
docker compose run --rm web scanpipe COMMAND
```

Additionally, you can start a new Docker container and execute multiple `scanpipe` commands within a `bash` session:

```
docker compose run web bash
scanpipe COMMAND
scanpipe COMMAND
...
```

Warning: In order to add local input files to a project using the Command Line Interface, extra arguments need to be passed to the `docker compose` command.

For instance `--volume /path/on/host:/target/path/in/container:ro` will mount and make available the host path inside the container (`:ro` stands for read only).

```
docker compose run --volume /home/sources:/sources:ro \
    web scanpipe create-project my_project --input-file="/sources/image.tar"
```

Note: In a local development installation, the `scanpipe` command is directly available as an entry point in your virtualenv and is located at `<scancode.io_root_dir>/bin/scanpipe`.

22.1 $ scanpipe –help

Lists all sub-commands available, including Django built-in commands. ScanPipe's own commands are listed under the [scanpipe] section:

```
$ scanpipe --help
...
[scanpipe]
    add-input
    add-pipeline
    archive-project
    create-project
    delete-project
    execute
    list-project
    output
    show-pipeline
    status
```

22.2 $ scanpipe <subcommand> –help

Displays help for the provided sub-command.

For example:

```
$ scanpipe create-project --help
usage: scanpipe create-project [--input-file INPUTS_FILES]
    [--input-url INPUT_URLS] [--copy-codebase SOURCE_DIRECTORY]
    [--pipeline PIPELINES] [--execute] [--async]
    name

Create a ScanPipe project.

positional arguments:
  name                  Project name.
```

22.3 $ scanpipe create-project <name>

Creates a ScanPipe project using `<name>` as a Project name. The project name must be unique.

Optional arguments:

- `--pipeline PIPELINES` Pipelines names to add on the project.

Tip: Use the "pipeline_name:group1,group2" syntax to select steps groups:

```
--pipeline map_deploy_to_develop:Java,JavaScript
```

- `--input-file INPUTS_FILES` Input file locations to copy in the *input/* work directory.

Tip: Use the “filename:tag” syntax to **tag** input files: `--input-file path/filename:tag`

- `--input-url INPUT_URLS` Input URLs to download in the *input/* work directory.

Tip: Use the “url#tag” syntax to tag downloaded files: `--input-url https://url.com/filename#tag`

- `--copy-codebase SOURCE_DIRECTORY` Copy the content of the provided source directory into the *codebase/* work directory.
- `--execute` Execute the pipelines right after project creation.
- `--async` Add the pipeline run to the tasks queue for execution by a worker instead of running in the current thread. Applies only when `--execute` is provided.

Warning: Pipelines are added and are executed in order.

22.4 *$ scanpipe list-project [–search SEARCH] [–include-archived]*

Lists ScanPipe projects.

Optional arguments:

- `--search SEARCH` Limit the projects list to this search results.
- `--include-archived` Include archived projects.

Tip: Only the project names are listed by default. You can display more details about each project by providing the `--verbosity 2` or `--verbosity 3` options.

22.5 *$ scanpipe add-input –project PROJECT [–input-file FILES] [–input-url URLS]*

Adds input files in the project’s work directory.

- `--input-file INPUTS_FILES` Input file locations to copy in the *input/* work directory.

Tip: Use the “filename:tag” syntax to **tag** input files: `--input-file path/filename:tag`

- `--input-url INPUT_URLS` Input URLs to download in the *input/* work directory.

Tip: Use the “url#tag” syntax to tag downloaded files: `--input-url https://url.com/filename#tag`

- `--copy-codebase SOURCE_DIRECTORY` Copy the content of the provided source directory into the *codebase/* work directory.

For example, assuming you have created beforehand a project named "foo", this will copy `~/docker/alpine-base.tar` to the foo project *input/* directory:

```
$ scanpipe add-input --project foo --input-file ~/docker/alpine-base.tar
```

Warning: Make sure to mount your local sources volume in the Docker setup:

`--volume /host/sources:/sources:ro --input-file /sources/image.tar`

You can also provide URLs of files to be downloaded to the foo project *input/* directory:

```
$ scanpipe add-input --project foo --input-url https://github.com/nexB/scancode.io-
↪tutorial/releases/download/sample-images/30-alpine-nickolashkraus-staticbox-latest.tar
```

Note: Docker images can be provided as input using their Docker reference with the `docker://docker-reference` syntax. For example:

```
$ [...] --input-url docker://redis
$ [...] --input-url docker://postgres:13
$ [...] --input-url docker://docker.elastic.co/elasticsearch/elasticsearch-oss:7.10.2
```

See https://docs.docker.com/engine/reference/builder/ for more details about references.

22.6 *$ scanpipe add-pipeline –project PROJECT PIPELINE_NAME [PIPELINE_NAME . . .]*

Adds the `PIPELINE_NAME` to a given `PROJECT`. You can use more than one `PIPELINE_NAME` to add multiple pipelines at once.

Warning: Pipelines are added and are executed in order.

For example, assuming you have created beforehand a project named "foo", this will add the docker pipeline to your project:

```
$ scanpipe add-pipeline --project foo analyze_docker_image
```

Tip: Use the "pipeline_name:group1,group2" syntax to select steps groups:

`--pipeline map_deploy_to_develop:Java,JavaScript`

22.7 $ scanpipe execute –project PROJECT

Executes the next pipeline of the `PROJECT` project queue.

Optional arguments:

- `--async` Add the pipeline run to the tasks queue for execution by a worker instead of running in the current thread.

22.8 $ scanpipe show-pipeline –project PROJECT

Lists all the pipelines added to the `PROJECT` project.

22.9 $ scanpipe status –project PROJECT

Displays status information about the `PROJECT` project.

Note: The full logs of each pipeline execution are displayed by default. This can be disabled providing the `--verbosity 0` option.

22.10 $ scanpipe output –project PROJECT –format {json,csv,xlsx,spdx,cyclonedx,attribution}

Outputs the `PROJECT` results as JSON, XLSX, CSV, SPDX, CycloneDX, and Attribution. The output files are created in the `PROJECT` *output/* directory.

Multiple formats can be provided at once:

```
$ scanpipe output --project foo --format json xlsx spdx cyclonedx attribution
```

Optional arguments:

- `--print` Print the output to stdout instead of creating a file. This is not compatible with the XLSX and CSV formats. It cannot be used when multiple formats are provided.

Refer to *Mount projects workspace* to access your outputs on the host machine when running with Docker.

22.11 $ scanpipe archive-project –project PROJECT

Archives a project and remove selected work directories.

Optional arguments:

- `--remove-input` Remove the *input/* directory.
- `--remove-codebase` Remove the *codebase/* directory.
- `--remove-output` Remove the *output/* directory.
- `--no-input` Does not prompt the user for input of any kind.

22.12 *$ scanpipe reset-project –project PROJECT*

Resets a project removing all database entrie and all data on disks except for the input/ directory.

Optional arguments:

- `--no-input` Does not prompt the user for input of any kind.

22.13 *$ scanpipe delete-project –project PROJECT*

Deletes a project and its related work directories.

Optional arguments:

- `--no-input` Does not prompt the user for input of any kind.

22.14 *$ scanpipe create-user <username>*

Note: This command is to be used when ScanCode.io's authentication system *SCANCODEIO_REQUIRE_AUTHENTICATION* is enabled.

Creates a user and generates an API key for authentication.

You will be prompted for a password. After you enter one, the user will be created immediately.

The API key for the new user account will be displayed on the terminal output.

```
User <username> created with API key: abcdef123456
```

The API key can also be retrieved from the *Profile settings* menu in the UI.

Warning: Your API key is like a password and should be treated with the same care.

By default, this command will prompt for a password for the new user account. When run non-interactively with the `--no-input` option, no password will be set, and the user account will only be able to authenticate with the REST API using its API key.

Optional arguments:

- `--no-input` Does not prompt the user for input of any kind.

CHAPTER

TWENTYTHREE

REST API

To get started with the REST API, visit the **projects' API endpoint** at http://localhost/api/projects/ or http://localhost:8001/api/projects/ if you run on a local development setup.

23.1 Authentication

When the authentication setting *SCANCODEIO_REQUIRE_AUTHENTICATION* is enabled on a ScanCode.io instance (disabled by default), you will have to include an authentication token `API key` in the Authorization HTTP header of each request.

The key should be prefixed by the string literal "Token" with whitespace separating the two strings. For example:

```
Authorization: Token abcdef123456
```

Warning: Your API key is like a password and should be treated with the same care.

Example of a cURL-style command line using an API Key for authentication:

```
curl -X GET http://localhost/api/projects/ -H "Authorization:Token abcdef123456"
```

Example of a Python script:

```python
import requests

api_url = "http://localhost/api/projects/"
headers = {
    "Authorization": "Token abcdef123456",
}
params = {
    "page": "2",
}
response = requests.get(api_url, headers=headers, params=params)
response.json()
```

23.2 Project list

An API endpoint that provides the ability to list, get, and create projects.

GET /api/projects/

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "name": "alpine/httpie",
            "url": "/api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/",
            "uuid": "6461408c-726c-4b70-aa7a-c9cc9d1c9685",
            "created_date": "2021-07-27T08:43:06.058350+02:00",
            "next_run": null,
            "runs": []
        }
    ]
}
```

The project list can be filtered by `name`, `uuid`, and `is_archived` fields. For example:

```
api_url="http://localhost/api/projects/"
content_type="Content-Type: application/json"
payload="name=project_name"

curl -X GET "$api_url?$payload" -H "$content_type"
```

23.3 Create a project

Using cURL:

```
api_url="http://localhost/api/projects/"
content_type="Content-Type: application/json"
data='{
    "name": "project_name",
    "input_urls": "https://download.url/package.archive",
    "pipeline": "scan_single_package",
    "execute_now": true
}'

curl -X POST "$api_url" -H "$content_type" -d "$data"
```

Note: To **upload a file** as the input of the project, you have to use the cURL "form emulation" mode with the following syntax:

```
api_url="http://localhost/api/projects/"
upload_file="/path/to/the/archive.zip"

```

(continues on next page)

(continued from previous page)

```
curl -F "name=project_name" \
     -F "pipeline=scan_single_package" \
     -F "execute_now=True" \
     -F "upload_file=@$upload_file" \
     "$api_url"
```

Tip: To upload more than one file, you can use the *Add input* endpoint of the project.

Tip: To tag the `upload_file`, you can provide the tag value using the `upload_file_tag` field.

Using Python and the **"requests"** library:

```python
import requests

api_url = "http://localhost/api/projects/"
data = {
    "name": "project_name",
    "input_urls": "https://download.url/package.archive",
    "pipeline": "scan_single_package",
    "execute_now": True,
}
response = requests.post(api_url, data=data)
response.json()
```

Note: To **upload a file** as the input of the project, you have to provide the `files` argument to the `requests.post` call:

```python
import requests

api_url = "http://localhost/api/projects/"
data = {
    "name": "project_name",
    "pipeline": "scan_single_package",
    "execute_now": True,
}
files = {"upload_file": open("/path/to/the/archive.zip", "rb")}
response = requests.post(api_url, data=data, files=files)
response.json()
```

You have the flexibility to explicitly set the filename, content_type, and headers for your uploaded files using the following code:

```python
files = {"upload_file": ("inventory.json", file_contents, "application/json")}
```

For more information on this topic, refer to the following link: https://docs.python-requests.org/en/latest/user/quickstart/#post-a-multipart-encoded-file

When creating a project, the response will include the project's details URL value among the returned data. You can make a GET request to this URL, which returns all available information about the project, including the status of any pipeline run:

```
{
    "name": "project_name",
    "url": "/api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/",
    "uuid": "6461408c-726c-4b70-aa7a-c9cc9d1c9685",
    "created_date": "2021-07-21T16:06:29.132795+02:00"
}
```

23.4 Project details

The project details view returns all information available about a project.

`GET /api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/`

```
{
    "name": "alpine/httpie",
    "url": "/api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/",
    "uuid": "6461408c-726c-4b70-aa7a-c9cc9d1c9685",
    "created_date": "2021-07-27T08:43:06.058350+02:00",
    "[...]": "[...]",
    "codebase_resources_summary": {
        "application-package": 1
    },
    "discovered_packages_summary": {
        "total": 1,
        "with_missing_resources": 0,
        "with_modified_resources": 0
    }
}
```

23.5 Managing projects

Multiple **actions** are available to manage projects:

23.5.1 Add input

This action adds provided `input_urls` or `upload_file` to the `project`.

`POST /api/projects/d4ed9405-5568-45ad-99f6-782a9b82d1d2/add_input/`

Data:

- `input_urls`: A list of URLs to download
- `upload_file`: A file to upload
- `upload_file_tag`: An optional tag to add on the uploaded file

Using cURL to provide download URLs:

```
api_url="http://localhost/api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/add_input/"
content_type="Content-Type: application/json"
data='{
    "input_urls": [
        "https://github.com/nexB/debian-inspector/archive/refs/tags/v21.5.25.zip",
        "https://github.com/package-url/packageurl-python/archive/refs/tags/0.9.4.tar.gz"
   ]
}'

curl -X POST "$api_url" -H "$content_type" -d "$data"
```

```
{
    "status": "Input(s) added."
}
```

Using cURL to upload a local file:

```
api_url="http://localhost/api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/add_input/"
upload_file="/path/to/the/archive.zip"

curl -X POST "$api_url" -F "upload_file=@$upload_file"
```

```
{
    "status": "Input(s) added."
}
```

23.5.2 Add pipeline

This action adds a selected `pipeline` to the `project`. If the `execute_now` value is True, the pipeline execution will start immediately during the pipeline addition.

`POST /api/projects/d4ed9405-5568-45ad-99f6-782a9b82d1d2/add_pipeline/`

Data:

- `pipeline`: The pipeline name
- `execute_now`: `true` or `false`

Tip: Use the "pipeline_name:group1,group2" syntax to select steps groups:

`"pipeline": "map_deploy_to_develop:Java,JavaScript"`

Using cURL:

```
api_url="http://localhost/api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/add_pipeline/
↪"
content_type="Content-Type: application/json"
data='{
    "pipeline": "analyze_docker_image",
    "execute_now": true
}'
```

(continues on next page)

(continued from previous page)

```
curl -X POST "$api_url" -H "$content_type" -d "$data"
```

```
{
    "status": "Pipeline added."
}
```

23.5.3 Archive

This action archive a project and remove selected work directories.

`POST /api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/archive/`

Data:

- `remove_input`: `true`
- `remove_codebase`: `true`
- `remove_output`: `false`

```
{
    "status": "The project project_name has been archived."
}
```

23.5.4 Reset

This action will delete all related database entrie and all data on disks except for the *input/* directory.

`POST /api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/reset/`

```
{
    "status": "All data, except inputs, for the project_name project have been removed."
}
```

23.5.5 Errors

This action lists all errors that were logged during any pipeline execution on a given `project`.

`GET /api/projects/6461408c-726c-4b70-aa7a-c9cc9d1c9685/errors/`

```
[
    {
        "uuid": "d4ed9405-5568-45ad-99f6-782a9b82d1d2",
        "model": "CodebaseResource",
        "[...]": "[...]",
        "message": "ERROR: for scanner: packages:",
        "created_date": "2021-04-27T22:38:30.762731+02:00"
    }
]
```

23.5.6 File content

This displays the content of a `project` file resource provided using the `?path=<resource_path>` argument.

`GET /api/projects/d4ed9405-5568-45ad-99f6-782a9b82d1d2/file_content/?path=setup.py`

```
{
    "file_content": "#!/usr/bin/env python\n# -*- encoding: utf-8 -*-\n\n..."
}
```

23.5.7 Packages

Lists all `packages` of a given `project`.

`GET /api/projects/d4ed9405-5568-45ad-99f6-782a9b82d1d2/packages/`

```
[
    {
        "purl": "pkg:deb/debian/libdb5.3@5.3.28%2Bdfsg1-0.5?arch=amd64",
        "type": "deb",
        "namespace": "debian",
        "name": "libdb5.3",
        "version": "5.3.28+dfsg1-0.5",
        "[...]": "[...]"
    }
]
```

23.5.8 Resources

This action lists all `resources` of a given `project`.

`GET /api/projects/d4ed9405-5568-45ad-99f6-782a9b82d1d2/resources/`

```
[
    {
        "for_packages": [
            "pkg:deb/debian/bash@5.0-4?arch=amd64"
        ],
        "path": "/bin/bash",
        "size": 1168776,
        "[...]": "[...]"
    }
]
```

23.5.9 Results

Displays the results as JSON content compatible with ScanCode data format.

`GET /api/projects/d4ed9405-5568-45ad-99f6-782a9b82d1d2/results/`

```
{
    "headers": [
        {
            "tool_name": "scanpipe",
            "tool_version": "21.8.2",
            "[...]": "[...]"
        }
    ]
}
```

23.5.10 Results (Download)

Finally, use this action to download the project results in the provided `output_format` as an attachment file.

Data:

- output_format: `json`, `xlsx`, `spdx`, `cyclonedx`, `attribution`

`GET /api/projects/d4ed9405-5568-45ad-99f6-782a9b82d1d2/results_download/?output_format=cyclonedx`

Tip: Refer to *Output Files* to learn more about the available output formats.

23.6 Run details

The run details view returns all information available about a pipeline run.

`GET /api/runs/c4d09fe5-c133-4c03-8286-6894ee5ffaab/`

```
{
    "url": "http://localhost/api/runs/8d5c3962-5fca-47d7-b8c8-47a19247714e/",
    "pipeline_name": "scan_single_package",
    "status": "success",
    "description": "A pipeline to scan a single package archive with ScanCode-toolkit.",
    "project": "http://localhost/api/projects/cd5b0459-303f-4e92-99c4-ea6d0a70193e/",
    "uuid": "8d5c3962-5fca-47d7-b8c8-47a19247714e",
    "created_date": "2021-10-01T08:44:05.174487+02:00",
    "task_exitcode": 0,
    "[...]": "[...]",
    "execution_time": 12
}
```

23.7 Managing pipeline runs

Multiple **actions** are available to manage pipeline runs:

23.7.1 Start pipeline

This action starts (send to the task queue) a pipeline run for execution.

`POST /api/runs/8d5c3962-5fca-47d7-b8c8-47a19247714e/start_pipeline/`

```
{
    "status": "Pipeline pipeline_name started."
}
```

23.7.2 Stop pipeline

This action stops a "running" pipeline.

`POST /api/runs/8d5c3962-5fca-47d7-b8c8-47a19247714e/stop_pipeline/`

```
{
    "status": "Pipeline pipeline_name stopped."
}
```

23.7.3 Delete pipeline

This action deletes a "not started" or "queued" pipeline run.

`POST /api/runs/8d5c3962-5fca-47d7-b8c8-47a19247714e/delete_pipeline/`

```
{
    "status": "Pipeline pipeline_name deleted."
}
```

CHAPTER TWENTYFOUR

AUTOMATION

To **automate ScanCode.io scans and schedule** them for regular execution or in response to **specific events**, such as commits or releases, you can explore various available options:

24.1 1. Utilize an external ScanCode.io server (REST API)

If you have access to an external ScanCode.io server, you can interact with it programmatically through the *REST API* to **trigger scans automatically**.

You can use the following Python script as a base and execute it from various automation methods such as a cron job or a git hook:

```python
from datetime import datetime
from os import getenv

import requests

# Configure the following variables to your needs
PROJECT_NAME = f"scan-{datetime.now().isoformat()}"
INPUT_URLS = [
    "https://github.com/nexB/scancode.io/archive/refs/tags/v32.4.0.zip",
]
PIPELINES = [
    "inspect_packages",
    "find_vulnerabilities",
]
EXECUTE_NOW = True


def create_project():
    session = requests.Session()

    # ScanCode.io server location
    SCANCODEIO_URL = getenv("SCANCODEIO_URL", default="").rstrip("/")
    if not SCANCODEIO_URL:
        raise ValueError("SCANCODEIO_URL value missing from the env")

    # Optional authentication
    SCANCODEIO_API_KEY = getenv("SCANCODEIO_API_KEY")
    if SCANCODEIO_API_KEY:
```

(continues on next page)

(continued from previous page)

```
        session.headers.update({"Authorization": f"Token {SCANCODEIO_API_KEY}"})

    projects_api_url = f"{SCANCODEIO_URL}/api/projects/"
    project_data = {
        "name": PROJECT_NAME,
        "input_urls": INPUT_URLS,
        "pipeline": PIPELINES,
        "execute_now": EXECUTE_NOW,
    }

    response = session.post(projects_api_url, data=project_data)
    print(response.json())


if __name__ == "__main__":
    create_project()
```

Note: Before running this script, ensure that the environment variables `SCANCODEIO_URL` and `SCANCODEIO_API_KEY` (when authentication is enabled) are set correctly. You can set the environment variables within the script command itself using the following format:

```
SCANCODEIO_URL="https://..." SCANCODEIO_API_KEY="apikey..." python script.py
```

By providing the required environment variables in this manner, you can execute the script with the appropriate configurations and credentials.

24.2 2. Integrating ScanCode.io with GitHub Workflows

Seamlessly integrate ScanCode.io into your GitHub Workflows to enable automated scans as an integral part of your development process.

Visit the scancode-action repository to explore and learn more about the GitHub Action for ScanCode.io. The repository provides detailed information, usage instructions, and configuration options to help you incorporate code scanning effortlessly into your workflows.

24.3 3. Run a Local ScanCode.io app on your machine (management commands)

To automate scans within your local environment, you can run the ScanCode.io app directly on your machine and leverage the *Command Line Interface*.

For instance, you can create a project and trigger it using the following command in a crontab:

```
docker compose exec -it web scanpipe create-project scan-$(date +"%Y-%m-%dT%H:%M:%S") \
  --pipeline scan_single_package \
  --input-url https://github.com/package-url/packageurl-python/archive/refs/heads/main.
↪zip \
  --execute
```

By executing this command, you initiate the project creation process, and the scan will be triggered automatically based on the specified pipeline and input URL.

CHAPTER

TWENTYFIVE

APPLICATION SETTINGS

ScanCode.io is configured with environment variables stored in a `.env` file.

The `.env` file is created at the root of the ScanCode.io codebase during its installation. You can configure your preferences using the following settings in the `.env` file.

Note: ScanCode.io is based on the Django web framework and its settings system. The list of settings available in Django is documented at Django Settings.

Tip: Settings specific to ScanCode.io are all prefixed with `SCANCODEIO_`.

Restarting the services is required following any changes to .env:

```
docker compose restart web worker
```

25.1 Instance settings

25.1.1 DATABASE

The database can be configured using the following settings:

```
SCANCODEIO_DB_HOST=localhost
SCANCODEIO_DB_NAME=scancodeio
SCANCODEIO_DB_USER=user
SCANCODEIO_DB_PASSWORD=password
SCANCODEIO_DB_PORT=5432
```

25.1.2 SCANCODEIO_REQUIRE_AUTHENTICATION

By default, the ScanCode.io Web UI and REST API are available without any authentication.

The authentication system can be enable with this settings:

```
SCANCODEIO_REQUIRE_AUTHENTICATION=True
```

Once enabled, all the Web UI views and REST API endpoints will force the user to login to gain access.

A management command *$ scanpipe create-user <username>* is available to create users and generate their API key for authentication.

See *Authentication* for details on using the `API key` authentication system in the REST API.

25.1.3 SCANCODEIO_WORKSPACE_LOCATION

This setting defines the workspace location of a given project. The **workspace** is the directory where **all of the project's files are stored** , such as input, codebase, and output files:

```
SCANCODEIO_WORKSPACE_LOCATION=/var/scancodeio/workspace/
```

It defaults to a *var/* directory in the local ScanCode.io codebase.

See *Project workspace* for more details.

25.1.4 SCANCODEIO_CONFIG_DIR

The location of the *.scancode/* configuration directory within the project codebase.

Default: `.scancode`

This directory allows to provide configuration files and customization for a ScanCode.io project directly through the codebase files.

For example, to provide a custom attribution template to your project, add it in a *.scancode/* directory located at the root of your codebase before uploading it to ScanCode.io. The expected location of the attribution template is:

```
.scancode/templates/attribution.html
```

25.1.5 SCANCODEIO_PROCESSES

By default, multiprocessing is enabled and configured to use an optimal number of CPUs available on the machine. You can control the number of parallel processes available to ScanCode.io using the SCANCODEIO_PROCESSES setting:

```
SCANCODEIO_PROCESSES=4
```

Multiprocessing can be disabled entirely using "0":

```
SCANCODEIO_PROCESSES=0
```

To disable both multiprocessing and threading, use "-1":

```
SCANCODEIO_PROCESSES=-1
```

Note: Multiprocessing and threading are disabled by default on operating system where the multiprocessing start method is not "fork", such as on macOS.

25.1.6 SCANCODEIO_ASYNC

When enabled, pipeline runs are **executed asynchronously**, meaning that users can continue using the app while the pipeline are run in the background.

The ASYNC mode is **enabled by default in a "Run with Docker" configuration** but **disabled in a "Local development" setup**.

It is possible to enable ASYNC mode in a "local development" setup with the following setting:

```
SCANCODEIO_ASYNC=True
```

Once enabled, pipeline runs will be sent to a task queue instead of being executed synchronously in the web server process.

Warning: The ASYNC mode required a **Redis server** and running a **tasks worker** using `$ make worker`.

On macOS, the ASYNC mode requires the following line in your environment:

```
export OBJC_DISABLE_INITIALIZE_FORK_SAFETY=YES
```

25.1.7 SCANCODEIO_TASK_TIMEOUT

Maximum time allowed for a pipeline to complete. The pipeline run will be stopped and marked as failed if that limit is reached.

The value is a string with specify unit including hour, minute, second (e.g. "1h", "3m", "5s"):

```
SCANCODEIO_TASK_TIMEOUT=24h
```

Default: `24h`

25.1.8 SCANCODEIO_SCAN_FILE_TIMEOUT

Maximum time allowed for a file to be analyzed when scanning a codebase.

The value unit is second and is defined as an integer:

```
SCANCODEIO_SCAN_FILE_TIMEOUT=120
```

Default: `120` (2 minutes)

25.1.9 SCANCODEIO_PIPELINES_DIRS

This setting defines any additional locations that ScanCode.io will search in for pipelines. It usually includes a list of comma-separated strings containing full paths of additional pipelines directories:

```
SCANCODEIO_PIPELINES_DIRS=/var/scancodeio/pipelines/,/home/user/pipelines/
```

25.1.10 SCANCODEIO_POLICIES_FILE

This setting defines the location of the policies file, or `policies.yml`. A valid policies file is required to enable compliance-related features.

```
license_policies:
-   license_key: mit
    label: Approved License
    compliance_alert: ''
-   license_key: mpl-2.0
    label: Restricted License
    compliance_alert: warning
-   license_key: gpl-3.0
    label: Prohibited License
    compliance_alert: error
```

- Licenses are referenced by the `license_key`.
- A Policy is defined with `label` and `compliance_alert`.
- The `compliance_alert` accepts 3 values: ‘’ for an empty string, warning, and error.

Note: When the policy feature is enabled, the `compliance_alert` values are displayed in the UI and returned in all downloadable results.

Tip: Check out the *License Policies and Compliance Alerts* tutorial for in-depth coverage of this feature.

25.1.11 SCANCODEIO_PAGINATE_BY

The number of objects display per page for each object type can be customized with the following setting:

```
SCANCODEIO_PAGINATE_BY=project=30,error=50,resource=100,package=100,dependency=100
```

25.1.12 SCANCODEIO_REST_API_PAGE_SIZE

A numeric value indicating the number of objects returned per page in the REST API:

```
SCANCODEIO_REST_API_PAGE_SIZE=100
```

Default: `50`

Warning: Using a large page size may have an impact on performances.

25.1.13 SCANCODEIO_LOG_LEVEL

By default, only a minimum of logging messages is displayed in the console, mostly to provide some progress about pipeline run execution.

Default: `INFO`

The `DEBUG` value can be provided to this setting to see all ScanCode.io debug messages to help track down configuration issues for example. This mode can be enabled globally through the `.env` file:

```
SCANCODEIO_LOG_LEVEL=DEBUG
```

Or, in the context of running a *scanpipe command*:

```
$ SCANCODEIO_LOG_LEVEL=DEBUG bin/scanpipe [command]
```

The web server can be started in DEBUG mode with:

```
$ SCANCODEIO_LOG_LEVEL=DEBUG make run
```

25.1.14 TIME_ZONE

A string representing the time zone for the current ScanCode.io installation. By default the `UTC` time zone is used:

```
TIME_ZONE=Europe/Paris
```

Note: You can view a detailed list of time zones here.

25.2 External services (integrations)

25.2.1 PURLDB

A public instance of **PurlDB** is accessible at https://public.purldb.io/.

Alternatively, you can deploy your own instance of PurlDB by following the instructions provided in the documentation at https://purldb.readthedocs.io/.

To configure your local environment, set the `PURLDB_URL` in your `.env` file:

```
PURLDB_URL=https://public.purldb.io/
```

While using the public PurlDB instance, providing an API key is optional. However, if authentication is enabled on your PurlDB instance, you can provide the API key using `PURLDB_API_KEY`:

```
PURLDB_API_KEY=insert_your_api_key_here
```

Note: Once the PurlDB is configured, a new "PurlDB" tab will be available in the discovered package details view.

25.2.2 VULNERABLECODE

You have the option to either deploy your instance of VulnerableCode or connect to the public instance.

To configure your local environment, set the `VULNERABLECODE_URL` in your `.env` file:

```
VULNERABLECODE_URL=https://public.vulnerablecode.io/
```

When using the public VulnerableCode instance, providing an API key is optional. However, if authentication is enabled on your VulnerableCode instance, you can provide the API key using `VULNERABLECODE_API_KEY`:

```
VULNERABLECODE_API_KEY=insert_your_api_key_here
```

25.2.3 MATCHCODE.IO

There is currently no public instance of MatchCode.io.

Alternatively, you can deploy your own instance of MatchCode.io by following the instructions provided in the documentation at https://purldb.readthedocs.io/.

To configure your local environment, set the `MATCHCODEIO_URL` in your `.env` file:

```
MATCHCODEIO_URL=https://<Address to MatchCode.io host>/
```

If authentication is enabled on your MatchCode.io instance, you can provide the API key using `MATCHCODEIO_API_KEY`:

```
MATCHCODEIO_API_KEY=insert_your_api_key_here
```

25.3 Fetch Authentication

Several settings are available to define the credentials required to access your private files, depending on the authentication type: Basic, Digest, Token header, etc.

Note: The provided credentials are enabled for all projects on the ScanCode.io instance.

Warning: Ensure that the provided `host` values are fully qualified, including the domain and subdomain.

25.3.1 SCANCODEIO_FETCH_BASIC_AUTH

You can provide credentials for input URLs protected by Basic Authentication using the `host=user,password` syntax:

```
SCANCODEIO_FETCH_BASIC_AUTH="www.host1.com=user,password;www.host2.com=user,password;"
```

25.3.2 SCANCODEIO_FETCH_DIGEST_AUTH

You can provide credentials for input URLs protected by Digest Authentication using the `host=user,password` syntax:

```
SCANCODEIO_FETCH_DIGEST_AUTH="www.host1.com=user,password;www.host2.com=user,password;"
```

25.3.3 SCANCODEIO_FETCH_HEADERS

When authentication credentials can be provided through HTTP request headers, you can use the following syntax:

```
SCANCODEIO_FETCH_HEADERS="www.host1.com=Header1=value,Header2=value;"
```

Example for a GitHub private repository:

```
SCANCODEIO_FETCH_HEADERS="raw.github.com=Authorization=token <YOUR_TOKEN>"
```

25.3.4 SCANCODEIO_NETRC_LOCATION

If your credentials are stored in a .netrc file, you can provide its location on disk using:

```
SCANCODEIO_NETRC_LOCATION="~/.netrc"
```

25.3.5 SCANCODEIO_SKOPEO_CREDENTIALS

You can define the username and password for Skopeo to access containers private registries using the `host=user:password` syntax:

```
SCANCODEIO_SKOPEO_CREDENTIALS="host1=user:password,host2=user:password"
```

25.3.6 SCANCODEIO_SKOPEO_AUTHFILE_LOCATION

Specify the path of the Skopeo authentication file using the following setting:

```
SCANCODEIO_SKOPEO_AUTHFILE_LOCATION="/path/to/auth.json"
```

CHAPTER

TWENTYSIX

RECOGNIZED DISTROS, OS, AND IMAGES

26.1 Archives Formats

ScanCode.io recognizes and **can extract most archive formats**; however, it offers special support for VM and container image formats:

- **Docker image tarbal** archives using a Docker image layout
- **Virtual machine images** using **QCOW** and **VHDI** image format

26.2 Operating Systems Detection

When scanning for Docker or virtual machine (VM) images, one of the first tasks of a pipeline after extracting an archive is to **detect the operating system**. For Linux, this also includes detecting the installed Linux distribution, which checks for certain files such as `/etc/os-release` on Linux. The detected OS—distro—is then used to determine **which system packages are likely installed**, such as RPM or Debian packages.

For each recognized OS, a pipeline collects the following information:

- OS and image details
- Installed system packages metadata, license and their files
- Installed application packages metadata, license and their files
- Details for files not part of a package

26.3 Installed System Packages

ScanCode.io recognizes the following OS technology combinations; some of which may be only used for certain pipelines:

- **Debian-based** Linux distros: Debian, Ubuntu and Debian-derivative
- **RPM-based** Linux distros: RHEL, Fedora, openSUSE/SUSE
- **Alpine** Linux distros

For the above three flavors, the *analyze_docker_image* and *analyze_root_filesystem_or_vm_image* pipelines support comprehensive detection of installed system packages, their provenance, their license metadata, and their installed files.

- For **Windows**, the *analyze_windows_docker_image* pipeline supports Windows Docker images with extensive detection of installed Windows packages, programs, and the majority of installed files.

- **Distroless** Docker images system packages are detected with the *analyze_docker_image* pipeline; package and license metadata are also detected. However, some work needs to be done to achieve comprehensive support and fix the issue of system packages ot tracking their installed files. Check this open GitHub issue for more details.
- **Yocto** and **OpenWRT** Linux VM images are partially supported; adding more support is currently in progress.
- Other distros and OS will be scanned; however, we might not be able to detect system installed packages and may return a larger volume of file-level detections.

CHAPTER

TWENTYSEVEN

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

INDEX