ska-sdp-dataproduct-dashboard Documentation

Release 0.4.3

Snehal Valame

Aug 08, 2023

CONTENTS:

The *ska-sdp-dataproduct-dashboard* repository contains the code for the dashboard that is used to access Data Products in a storage volume.

CHAPTER

ONE

OVERVIEW

The data product dashboard is used to list and download data products that are saved on a shared data volume within the environment where it is deployed.

1.1 Usage

The dashboard contains a table of all the data products in the shared data volume, as seen by the Data Product API. This table has built-in functionality for sorting and filtering. When a data product is selected, its metadata is displayed on the right as well as the download options for the selected product.

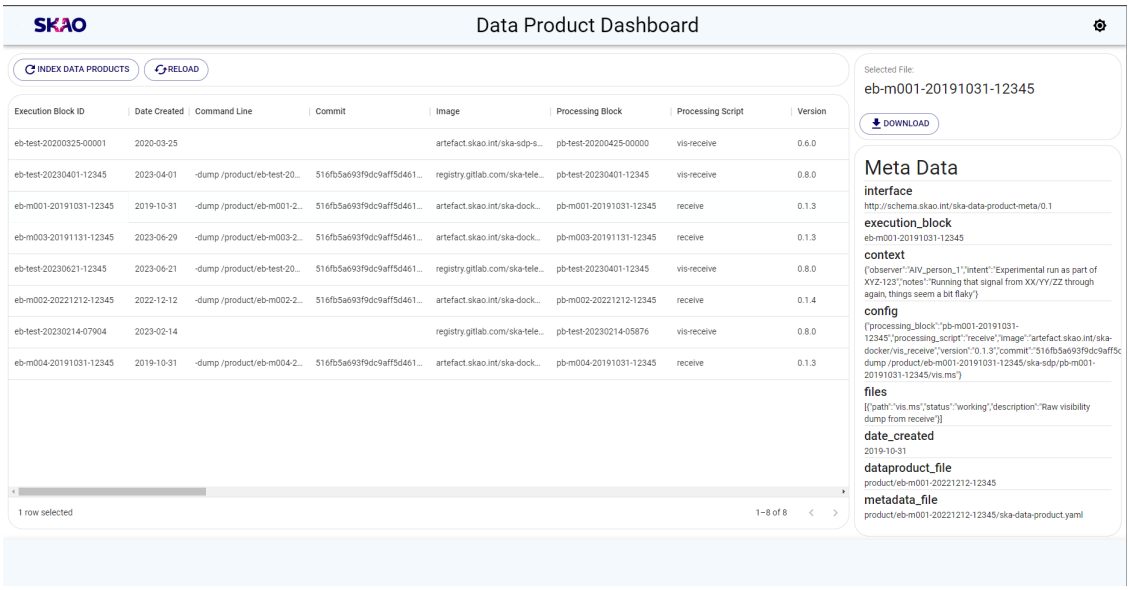

Fig. 1: Example Data Product Dashboard

Above the table is a panel which contains functions related to the data store in use. These can include the following:

- **Index Data Products**

  This button will re-index the shared data volume from disk into the metadata store. This allows the user to find items that might have been added to the volume without being ingested by the API.

- **Reload**

  This function will update the table on the dashboard with the latest list of data products in the metadata store. This allows the user to find items that have been ingested by the API since the dashboard last loaded.

When the data product API has access to an Elasticsearch backend, additional search functionality will become available. This allows the user to use Elasticsearch to search for a key value pair within the metadata.

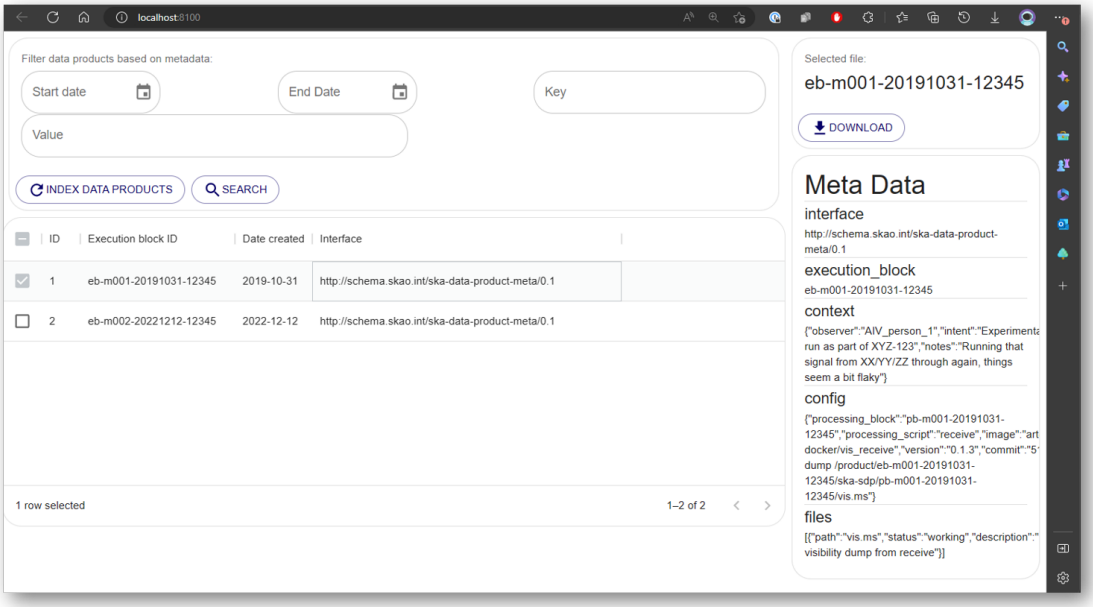

Fig. 2: Example Data Product Dashboard with Elasticsearch enabled.

The application can be run as a standalone front-end application or used as a remote (Webpack 5 Module) within the SKA Portal SKA Landing Page.

CHAPTER

TWO

DEVELOPER GUIDE

This document complements the guidelines set out in the SKA telescope developer portal

Tooling Pre-requisites

This project requires **Node** and **YARN** to install and run. To install please follow the instructions for your operating system at nodejs downloads.

Alternatively, the official Node docker image can be used. Instructions can be found on the official Node docker image site.

2.1 Development setup

To run the application directly on your host machine

All the following notes assume you are at the command prompt for your chosen environment.

Confirm Node and YARN are installed and configured correctly, both the following commands should return the relevant version number.

```
> node --version
> yarn --version
```

Clone the repository and its submodules:

```
git clone git@gitlab.com:ska-telescope/sdp/ska-sdp-dataproduct-dashboard.git
git submodule update --init --recursive
```

Scripts for running, testing, and building the application are defined in the scripts section of the package.json file. These are run using YARN

To run the application locally on your host machine, install all the necessary project dependencies with the YARN package manager:

```
> yarn install
```

2.2 Running scripts

You should now be able to run the scripts defined in the package.json within the project directory.

Running the application in development mode

The app can be run with the node environment set to NODE_ENV=test (allowing Istanbul to instrument the code) and webpack serve with –mode development. You can access your application at http://localhost:8100. The app will recompile and restart if you make any edits to the source files.

```
> yarn start
```

Running the application tests using Cypress

Cypress has been set up to provide component and end to end testing. For information on the use of Cypress, see Cypress component-testing.

Code coverage is implemented with Istanbul and NYC for instrumenting the code, and cobertura reporter as it is used for reporting for the Gitlab CI of coverage statistics.

Cypress can be opened in a browser by running:

```
> yarn cypress:open
```

Or alternatively unit and end to end tests tests can be run headless by:

```
> yarn test:component:headless
> yarn test:e2e:headless
```

Code coverage can be viewed by opening the *build/coverage/index.html* in a browser after running:

```
> yarn test:coverage:report
```

Running the production code

The build script builds the app for production to the *dist* folder. The build is minified and any JSX is transpiled to JavaScript. Your app is ready to be deployed!

```
> yarn build
```

Running the application inside a docker container on your host machine

To run the application using docker, build the docker file in the root directory and run the container exposing port 8100.

```
docker build -t ska-sdp-dataproduct-dashboard .
docker run -p 8100:8100 ska-sdp-dataproduct-dashboard
```

The project will then be accessible at the URL http://localhost:8100/

CHAPTER

THREE

DEPLOYMENT GUIDE

3.1 Kubernetes Deployment

This is the documentation for the Data Product Dashboard Helm Chart.

3.1.1 Usage

The data product dashboard is intended to be deployed as a standalone deployment, running as a service accessible to other deployments through its API or to users through the dashboard URL. Typical deployments are done from within the GitLab pipelines, deploying into pre-configured environments to one of three namespaces (ci-dev, integration or staging)

During development, developers can deploy the development branches into the ci-dev namespace from the Gitlab pipeline:

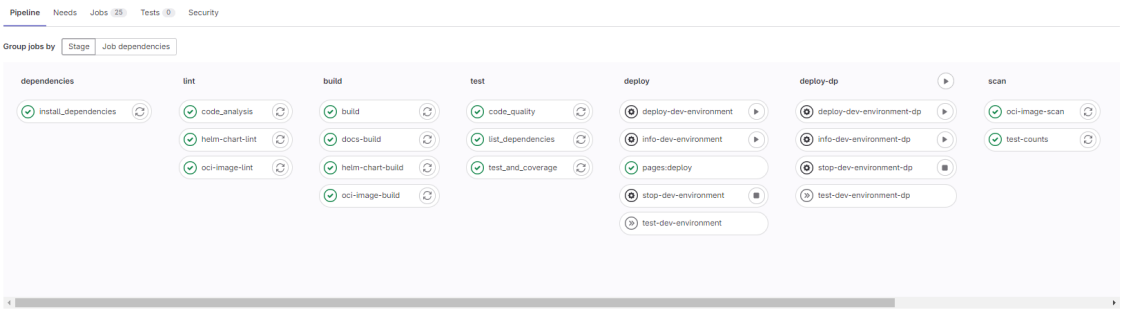

Fig. 1: Deployment from pipeline on dev branch

From the master branch, the application can be deployed into the integration or staging namespace of each environment.

The deployed Data Product Dashboard should then be accessible at: "https://sdhp.stfc.skao.int/\protect\T1\textdollarKUBE_NAMESPACE/dashboard/", and the backend should be accessible at: "https://sdhp.stfc.skao.int/\protect\T1\textdollarKUBE_NAMESPACE/api/"

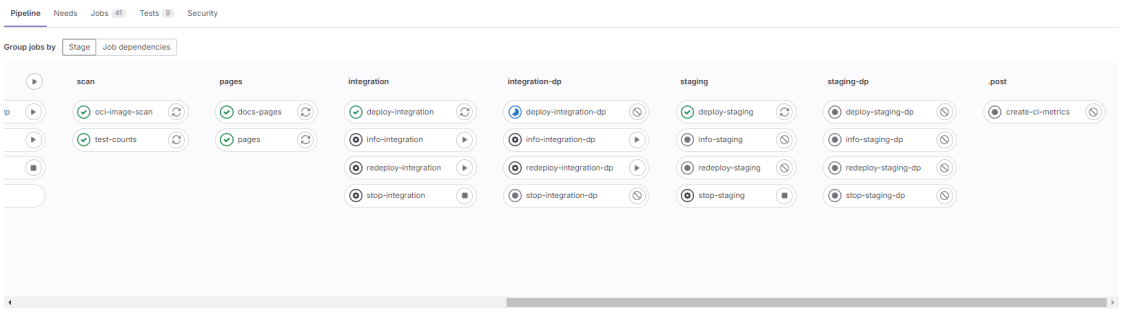

Fig. 2: Deployment from pipeline on master branch

3.2 Steps to run the system locally in Minikube

The following steps will assume that you have the repo checked out, or have the chart locally.

1. Start Minikube if it is not already running:

```
minikube start
minikube status
```

If needed, build images, tag and load them to Minikube.

```
docker build -t ska-sdp-dataproduct-dashboard .
docker images
docker tag [Image ID] ska-sdp-dataproduct-dashboard:[Tag]
minikube image load ska-sdp-dataproduct-dashboard:[Tag]
minikube image ls
```

2. Change to the chart directory in the repository: `cd charts/ska-sdp-dataproduct-dashboard/`. Make the needed changes to image versions and enable the deployments as required in the values files. Then update the Helm dependencies.

```
helm dependency update .
helm dependency build
```

3. Create a new namespace (optional): `kubectl create namespace [namespace]`
4. Install the helm chart with the following values:

   helm install [namespace] . -n [namespace] –set helmdeploy.namespace=[namespace] –values values_local_deployment.yaml

On a system with limited resources / slow connection, run with the following additional flags:

```
helm install [namespace] . -n [namespace] --set helmdeploy.namespace=[namespace] --
↪values values_local_deployment.yaml --set diagnosticMode.enabled=true --timeout=60m
```

Once the above is complete you will have the following running:

- The Data Product API
- The Data Product Dashboard

To be able to access the API and the dashboard run the following:

```
kubectl -n [namespace] port-forward service/ska-sdp-dataproduct-api 8000:8000
kubectl -n [namespace] port-forward service/ska-sdp-dataproduct-dashboard 8100:8100
```

You should now be able to access the API and the Dashboard on the following URL's:

- http://localhost:8000/filelist
- http://localhost:8100/

To get data onto the PV:

```
    kubectl get pod -n [namespace]
kubectl cp [host path]/ska-sdp-dataproduct-api/tests/test_files/product [ska-sdp-
↪dataproduct-api pod]:/usr/data -n [namespace]
```