ID2207 HT23 Modern Methods in Software Engineering (50928)

DataCloud Project

by Eugen Lucchiari Hartz

October 20, 2023

Task 1

A pipeline was developed based on the specifications. The used programming language is JavaScript. The implemented pipeline uses Docker containers to perform a series of data processing tasks. It starts by unzipping files using a dedicated container, followed by transforming TSV to CSV, splitting large CSV files, cleaning and preprocessing the data using a standalone executable from Grafterizer and finally convert Datagraft CSV files to ArangoDB values based on external transformation JSON from Grafterizer.
For the step where the pipeline cleans and preprocesses a CSV file using a stand alone executable from Grafterizer the container contains besides a script to execute Grafterizer also the Grafterizer tool found in a GitHub repo which provides a Dockerfile and a docker compose file. For the last step where the pipeline converts a Datagraft CSV files to ArangoDB values based on external transformation JSON from Grafterizer the idea is to use the Grafterizer GUI and tools to define a transformation that specifies how the Datagraft CSV files should be transformed to ArangoDB values.

Each processing step is implemented within a container which specifies the Docker image. Each container includes scripts written in JavaScript and/or other tools to be able to execute the specific data processing task. To install needed tools such as csv writer the node package manager (npm) was used. The pipeline is able to take input files as argument and process it to get a specified output. Furthermore, this pipeline follows a linear flow because each step produces an output that serves as an input for the next step. Therefore, I also implemented a bash script which executes each Docker container one after the other and accordingly automates the entire pipeline.

Time estimations/measurements in hours
- 3 hours learn Javascript
- 3 hours investigate about Docker

- 1 hour investigate about Bash
- 2 hours research and setup of Grafterizer
- 1 hours research ArangoDB
- 7 to write the code and implement the pipeline

Total for Task 1: 17h

Task 2

In this task the pipeline was implemented using a tool called Argo-Workflow. For setting up a Kubernetes cluster I used Docker Desktop. Furthermore, the argo CLI has been used. For implementing it I created a YAML file where each step is as before in Task 1 in a Docker container and the output of one step serves as the input for the next step.
Accordingly, the workflow specifies an entry point and templates for a directed acyclic graph (DAG) of tasks. The tasks include unzipping, converting TSV to CSV, splitting data, transforming, and sending data to ArangoDB. Each task is a container step utilizing specific Docker images which have been pushed to Docker Hub to make them available for Argo.

Time measurements in hours
- 4 hours to do investigations about Argo workflow
- 1 hour to do investigations about Kubernetes
- 2 hours to do more investigations about Docker (and how it is used with Argo)
- 4 hours to set it up and implement

Total for Task 2: 11h

Task 3

In this task the pipeline was implemented using the provided DEF-PIPE tool of the DataCloud project. Each processing step is again encapsulated within a container and there is a start and an end point. Furthermore, a DSL text file and a json file have been created with the help of this tool to describe the pipeline. Besides the start and the end point the pipeline consists of the steps “Unzip”, “TSVtoCSV”, “Split”, “Transform” and “ToArango”.

Time measurements in hours
- 1 hour to do investigations about the provided DEF-PIPE tool
- 1 hour to implement it

Total for Task 3: 2h

Task 4

4.1

In this task the Tasks 1-3 should be repeated with another provided pipeline that should again be described manually, with Argo and with the DEF-PIPE tool.

For doing task 1 the used programming language is again JavaScript and again the implemented pipeline uses Docker containers to perform a series of data processing tasks.
It begins with a "Start" element, serving as a data source, and ends with an "End" element, marking the conclusion of the pipeline. The second step, "GenerateSampleData," creates sample data by invoking a container implementation using the image "registry.cloud/tellucare-edge" and specific environmental parameters.
The third step, "ReceiveDataFromMQTT," processes data from an MQTT source, utilizing the "registry.cloud/tellucare-api:latest" image and associated environmental parameters, including authentication details for a RabbitMQ server.
Besides that, the "CreateNotification" step generates notifications through a container implementation using the "registry.sintef.cloud/application-logic:latest" image.
The pipeline includes a "FilterNotifications" step, which performs data processing using the "notification-filter:latest" image, specifying trigger conditions and resource requirements for execution.
This pipeline follows a linear data flow from one step to another.
In summary, the pipeline initializes with data generation, proceeds to ingest data from an MQTT source, creates notifications, and finally filters and processes the notifications.

Time estimations/measurements in hours

- **6 hours to do task 4.1**

4.2

For task 2 the pipeline has again been implemented using Argo Workflow. For the Kubernetes cluster I again used Docker Desktop and the argo CLI. As before the main important file is the created pipeline.yaml file. The five main steps are the same as in 4.1: "start", "generate-sample-data" "receive-data-from-mqtt" "create-notification" and "filter-notifications" and "end”. Each step is associated with a specific container, specifying the Docker image to be used and the command to execute within the container.

Time estimations/measurements in hours

- **4 hours to do task 4.2**

4.3

For this task again the provided DEF-PIPE tool of the DataCloud project has been used. The pipeline was given and the FilterNotifications step got included as last step. With the tool a json file can be generated which provides information about the pipeline. As already mentioned before the five main steps are the following: "start", "generate-sample-data" "receive-data-from-mqtt" "create-notification" and "filter-notifications" and "end”. Each step is associated with a specific container. In summary, the pipeline describes a data processing workflow where data is sourced, transformed, and processed through a series of containerized steps with specified execution requirements and parameters

Time estimations/measurements in hours

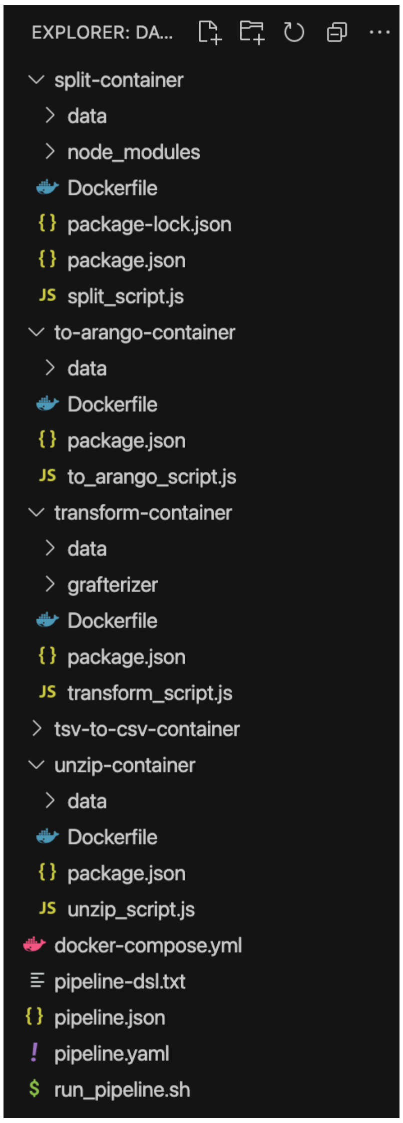
- **1 hour to do task 4.3**

Total for Task 4.1, 4.2 and 4,3: 11 hours

Appendix

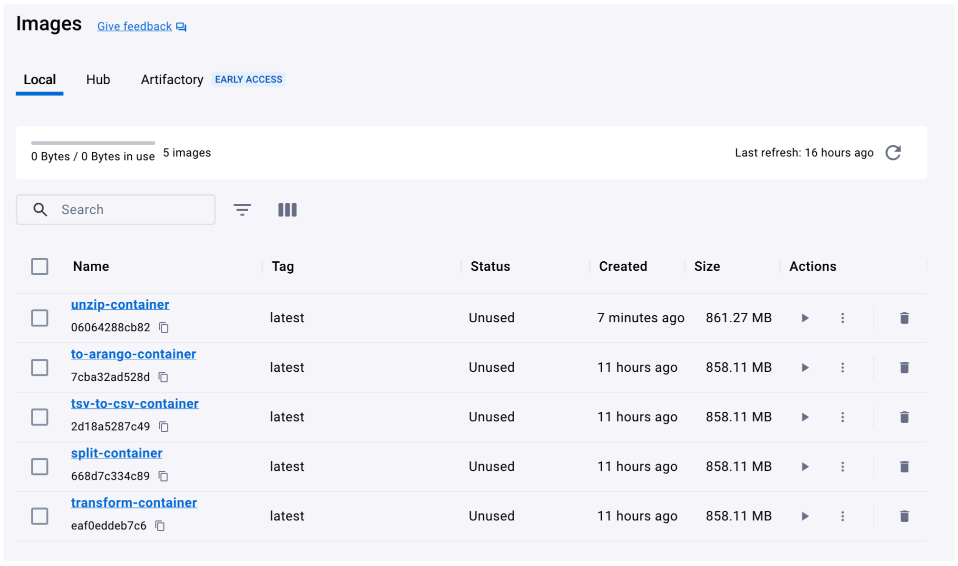
Appendix contains explanations, notes and screenshots I took while working on the different tasks.

Task 1



```
EXPLORER: DA...
v split-container
  > data
  > node_modules
  Dockerfile
  package-lock.json
  package.json
  split_script.js
v to-arango-container
  > data
  Dockerfile
  package.json
  to_arango_script.js
v transform-container
  > data
  > grafterizer
  Dockerfile
  package.json
  transform_script.js
> tsv-to-csv-container
v unzip-container
  > data
  Dockerfile
  package.json
  unzip_script.js
docker-compose.yml
pipeline-dsl.txt
pipeline.json
pipeline.yaml
run_pipeline.sh
```

Docker images successfully created with the docker build -t container-name . command


Images Give feedback

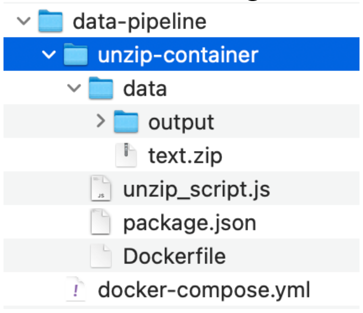
Local Hub Artifactory EARLY ACCESS

0 Bytes / 0 Bytes in use 5 images Last refresh: 16 hours ago

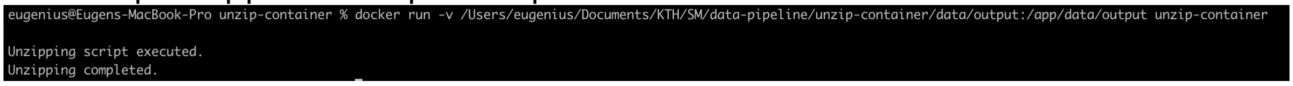
Search

Name	Tag	Status	Created	Size	Actions
unzip-container 06064288cb82	latest	Unused	7 minutes ago	861.27 MB	
to-arango-container 7cba32ad528d	latest	Unused	11 hours ago	858.11 MB	
tsv-to-csv-container 2d18a5287c49	latest	Unused	11 hours ago	858.11 MB	
split-container 668d7c334c89	latest	Unused	11 hours ago	858.11 MB	
transform-container eaf0eddeb7c6	latest	Unused	11 hours ago	858.11 MB	

before executing, we want to unzip text.zip

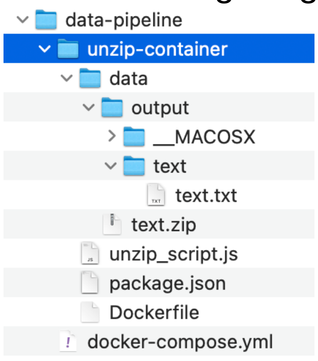

```
data-pipeline
  unzip-container
    data
      output
      text.zip
    unzip_script.js
    package.json
    Dockerfile
  docker-compose.yml
```

can be executed in the shell like this:
docker run -v /Users/eugenius/Documents/KTH/SM/data-pipeline/unzip-container/data/output:/app/data/output unzip-container

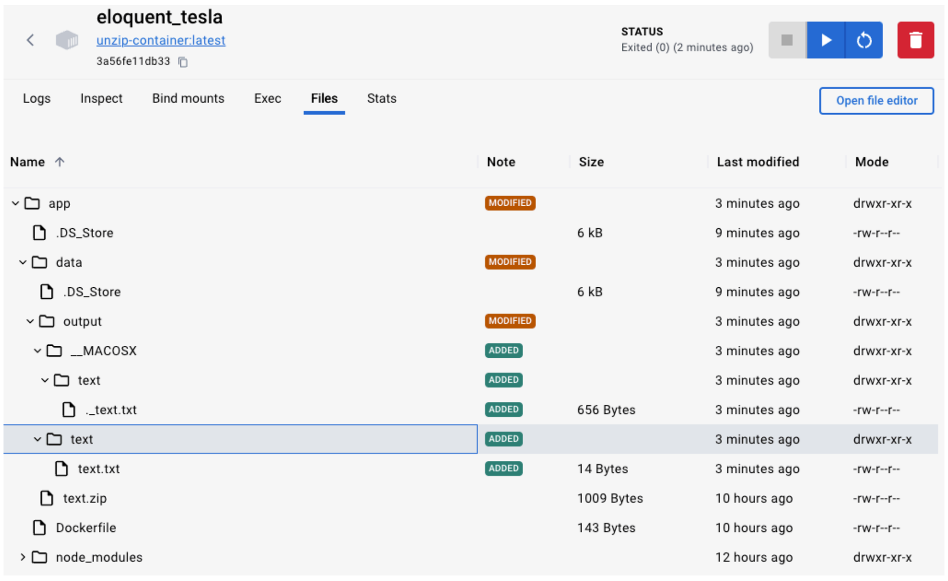

```
eugenius@Eugens-MacBook-Pro unzip-container % docker run -v /Users/eugenius/Documents/KTH/SM/data-pipeline/unzip-container/data/output:/app/data/output unzip-container

Unzipping script executed.
Unzipping completed.
```

after executing we get the unzipped file


```
data-pipeline
  unzip-container
    data
      output
        __MACOSX
        text
          text.txt
      text.zip
    unzip_script.js
    package.json
    Dockerfile
  docker-compose.yml
```

when we run the unzip container in the docker we can also execute it and get the unzipped file



eloquent_tesla
unzip-container:latest
3a56fe11db33

STATUS
Exited (0) (2 minutes ago)

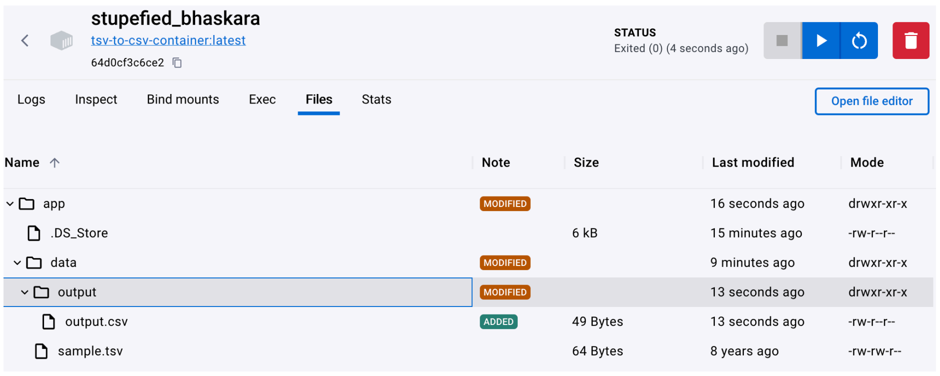
Logs Inspect Bind mounts Exec Files Stats

Open file editor

Name	Note	Size	Last modified	Mode
app	MODIFIED		3 minutes ago	drwxr-xr-x
.DS_Store		6 kB	9 minutes ago	-rw-r--r--
data	MODIFIED		3 minutes ago	drwxr-xr-x
.DS_Store		6 kB	9 minutes ago	-rw-r--r--
output	MODIFIED		3 minutes ago	drwxr-xr-x
__MACOSX	ADDED		3 minutes ago	drwxr-xr-x
text	ADDED		3 minutes ago	drwxr-xr-x
._text.txt	ADDED	656 Bytes	3 minutes ago	-rw-r--r--
text	ADDED		3 minutes ago	drwxr-xr-x
text.txt	ADDED	14 Bytes	3 minutes ago	-rw-r--r--
text.zip		1009 Bytes	10 hours ago	-rw-r--r--
Dockerfile		143 Bytes	10 hours ago	-rw-r--r--
node_modules			12 hours ago	drwxr-xr-x

similiar procedure now to test the other scripts and containers

tsv-to-csv-container



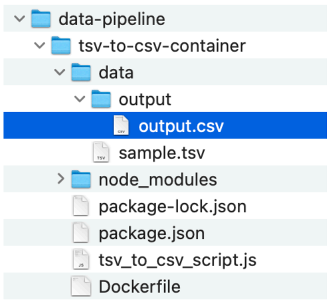
stupefied_bhaskara
tsv-to-csv-container:latest
64d0cf3c6ce2

STATUS
Exited (0) (4 seconds ago)

Logs Inspect Bind mounts Exec Files Stats

Open file editor

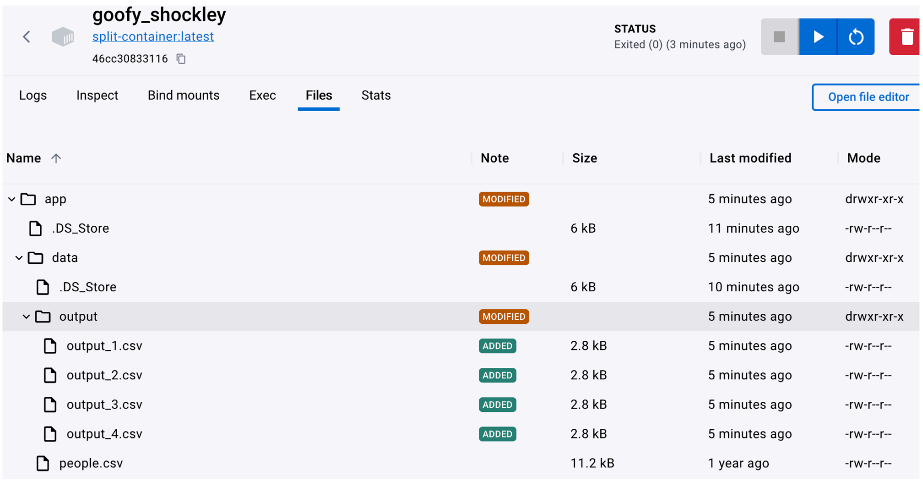
Name	Note	Size	Last modified	Mode
app	MODIFIED		16 seconds ago	drwxr-xr-x
.DS_Store		6 kB	15 minutes ago	-rw-r--r--
data	MODIFIED		9 minutes ago	drwxr-xr-x
output	MODIFIED		13 seconds ago	drwxr-xr-x
output.csv	ADDED	49 Bytes	13 seconds ago	-rw-r--r--
sample.tsv		64 Bytes	8 years ago	-rw-rw-r--



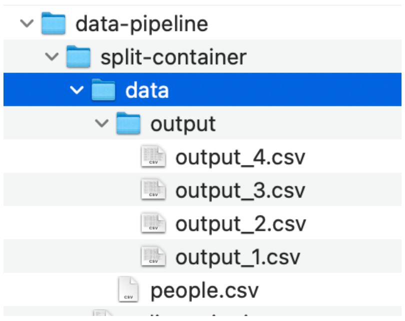
```
data-pipeline
  tsv-to-csv-container
    data
      output
        output.csv
      sample.tsv
    node_modules
    package-lock.json
    package.json
    tsv_to_csv_script.js
    Dockerfile
```

split-container

people.csv has 100 rows, number of rows per output file is set to 25, so we get 4 output files



Name	Note	Size	Last modified	Mode
app	MODIFIED		5 minutes ago	drwxr-xr-x
.DS_Store		6 kB	11 minutes ago	-rw-r--r--
data	MODIFIED		5 minutes ago	drwxr-xr-x
.DS_Store		6 kB	10 minutes ago	-rw-r--r--
output	MODIFIED		5 minutes ago	drwxr-xr-x
output_1.csv	ADDED	2.8 kB	5 minutes ago	-rw-r--r--
output_2.csv	ADDED	2.8 kB	5 minutes ago	-rw-r--r--
output_3.csv	ADDED	2.8 kB	5 minutes ago	-rw-r--r--
output_4.csv	ADDED	2.8 kB	5 minutes ago	-rw-r--r--
people.csv		11.2 kB	1 year ago	-rw-r--r--



```
data-pipeline
  split-container
    data
      output
        output_4.csv
        output_3.csv
        output_2.csv
        output_1.csv
      people.csv
```

transform-container

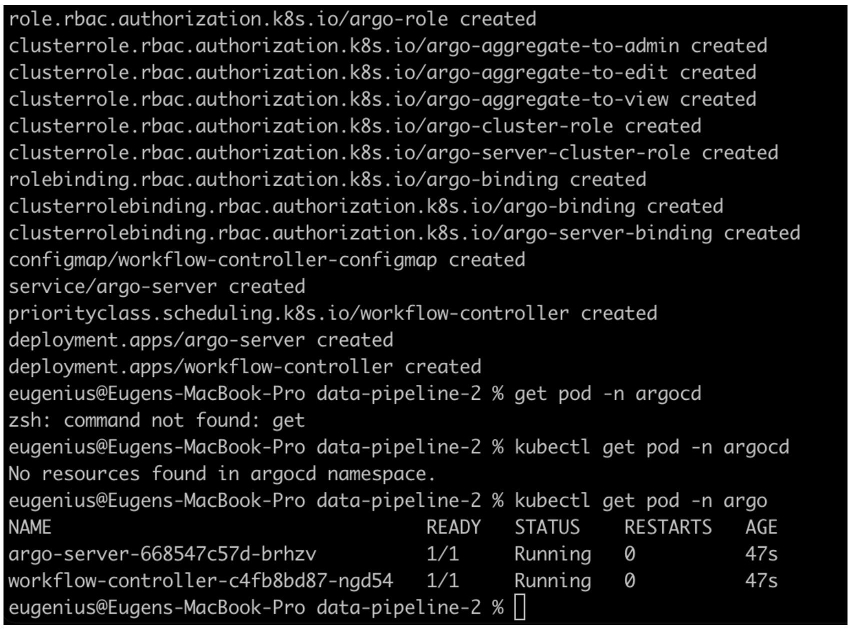
I found this grafterizer github repo https://github.com/datagraft/grafterizer
I do not know the tool but assume that the presented GUI can be used to clean and preprocess a CSV file by integrating it to my script and using the provided docker-compose file.

to-arango-container

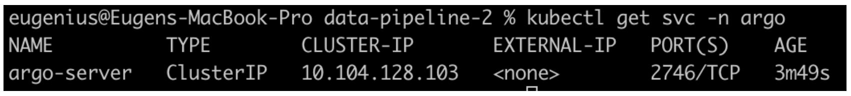
container contains besides a script to execute Grafterizer also the Grafterizer tool found in a GitHub repo which provides a Dockerfile and a docker compose file. For the last step where the pipeline converts a Datagraft CSV files to ArangoDB values based on external transformation JSON from Grafterizer the idea is to use the Grafterizer GUI and tools to define a transformation that specifies how the Datagraft CSV files should be transformed to ArangoDB values.

Task 2

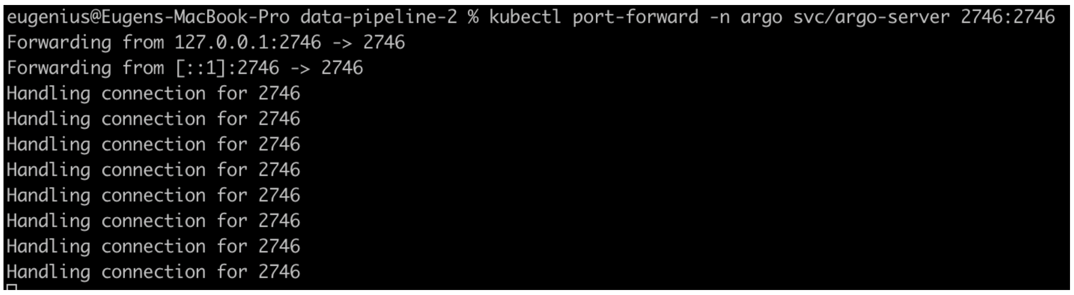
https://www.youtube.com/watch?v=MeU5_k9ssrs&t=1702s



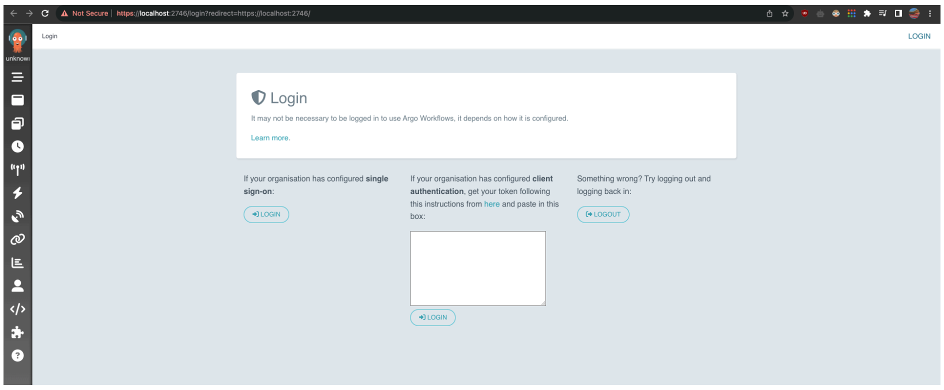
```
role.rbac.authorization.k8s.io/argo-role created
clusterrole.rbac.authorization.k8s.io/argo-aggregate-to-admin created
clusterrole.rbac.authorization.k8s.io/argo-aggregate-to-edit created
clusterrole.rbac.authorization.k8s.io/argo-aggregate-to-view created
clusterrole.rbac.authorization.k8s.io/argo-cluster-role created
clusterrole.rbac.authorization.k8s.io/argo-server-cluster-role created
rolebinding.rbac.authorization.k8s.io/argo-binding created
clusterrolebinding.rbac.authorization.k8s.io/argo-binding created
clusterrolebinding.rbac.authorization.k8s.io/argo-server-binding created
configmap/workflow-controller-configmap created
service/argo-server created
priorityclass.scheduling.k8s.io/workflow-controller created
deployment.apps/argo-server created
deployment.apps/workflow-controller created
eugenius@Eugens-MacBook-Pro data-pipeline-2 % get pod -n argocd
zsh: command not found: get
eugenius@Eugens-MacBook-Pro data-pipeline-2 % kubectl get pod -n argocd
No resources found in argocd namespace.
eugenius@Eugens-MacBook-Pro data-pipeline-2 % kubectl get pod -n argo
NAME                                  READY   STATUS    RESTARTS   AGE
argo-server-668547c57d-brhzv          1/1     Running   0          47s
workflow-controller-c4fb8bd87-ngd54   1/1     Running   0          47s
eugenius@Eugens-MacBook-Pro data-pipeline-2 % 
```



```
eugenius@Eugens-MacBook-Pro data-pipeline-2 % kubectl get svc -n argo
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
argo-server   ClusterIP   10.104.128.103   <none>        2746/TCP   3m49s
```



```
eugenius@Eugens-MacBook-Pro data-pipeline-2 % kubectl port-forward -n argo svc/argo-server 2746:2746
Forwarding from 127.0.0.1:2746 -> 2746
Forwarding from [::1]:2746 -> 2746
Handling connection for 2746
Handling connection for 2746
Handling connection for 2746
Handling connection for 2746
Handling connection for 2746
Handling connection for 2746
Handling connection for 2746
Handling connection for 2746
```


Not Secure
https://localhost:2746/login?redirect=https://localhost:2746/
unknown
Login
LOGIN

Login

It may not be necessary to be logged in to use Argo Workflows, it depends on how it is configured.

Learn more.

If your organisation has configured **single sign-on**:

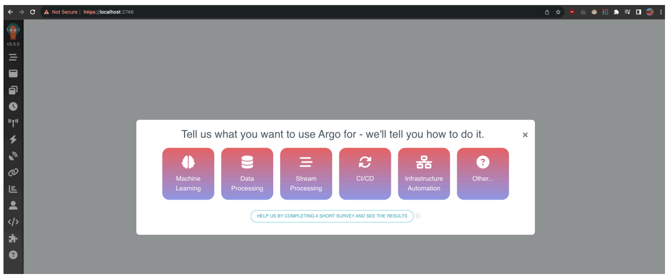
LOGIN

If your organisation has configured **client authentication**, get your token following this instructions from here and paste in this box:

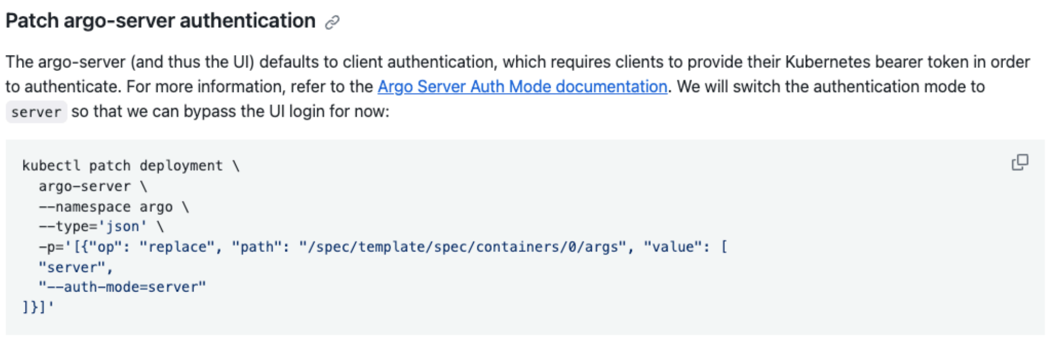
LOGIN

Something wrong? Try logging out and logging back in:

LOGOUT


Not Secure | https://localhost:2746
v3.5.0
Tell us what you want to use Argo for - we'll tell you how to do it.
Machine Learning
Data Processing
Stream Processing
CI/CD
Infrastructure Automation
Other...
HELP US BY COMPLETING A SHORT SURVEY AND SEE THE RESULTS

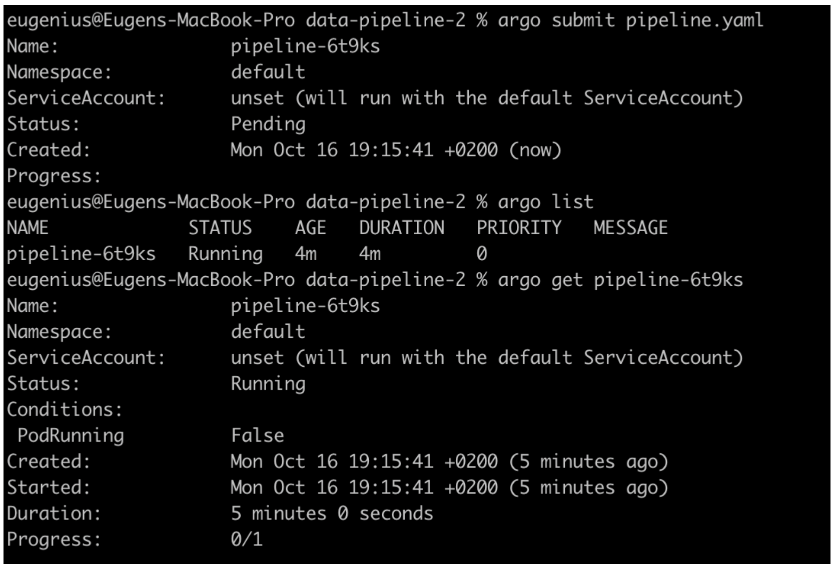
Important for authentication to not have the forbidden error:



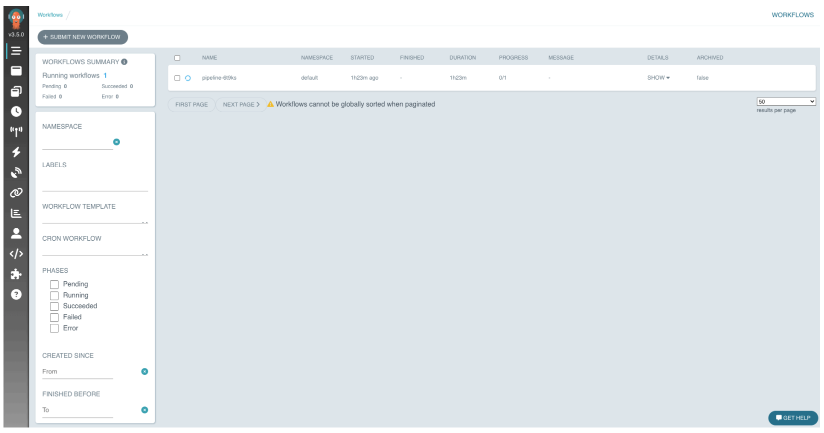
Patch argo-server authentication

The argo-server (and thus the UI) defaults to client authentication, which requires clients to provide their Kubernetes bearer token in order to authenticate. For more information, refer to the Argo Server Auth Mode documentation. We will switch the authentication mode to `server` so that we can bypass the UI login for now:

```
kubectl patch deployment \
  argo-server \
  --namespace argo \
  --type='json' \
  -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/args", "value": [
  "server",
  "--auth-mode=server"
]}]'
```



```
eugenius@Eugens-MacBook-Pro data-pipeline-2 % argo submit pipeline.yaml
Name:                pipeline-6t9ks
Namespace:           default
ServiceAccount:      unset (will run with the default ServiceAccount)
Status:              Pending
Created:             Mon Oct 16 19:15:41 +0200 (now)
Progress:
eugenius@Eugens-MacBook-Pro data-pipeline-2 % argo list
NAME             STATUS    AGE   DURATION   PRIORITY   MESSAGE
pipeline-6t9ks   Running   4m    4m         0
eugenius@Eugens-MacBook-Pro data-pipeline-2 % argo get pipeline-6t9ks
Name:                pipeline-6t9ks
Namespace:           default
ServiceAccount:      unset (will run with the default ServiceAccount)
Status:              Running
Conditions:
 PodRunning          False
Created:             Mon Oct 16 19:15:41 +0200 (5 minutes ago)
Started:             Mon Oct 16 19:15:41 +0200 (5 minutes ago)
Duration:            5 minutes 0 seconds
Progress:            0/1
```

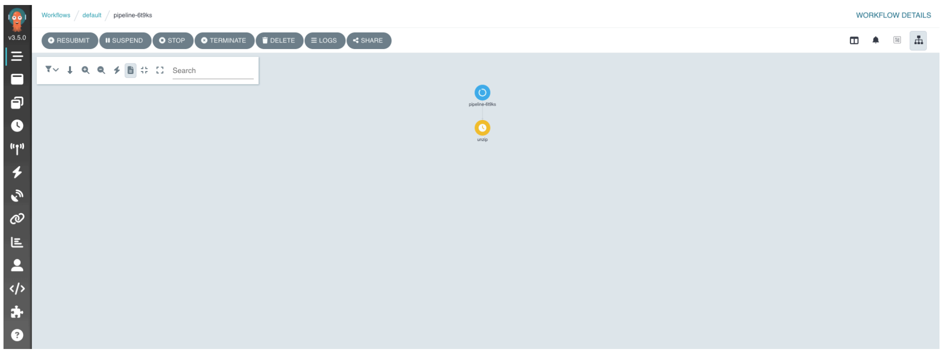


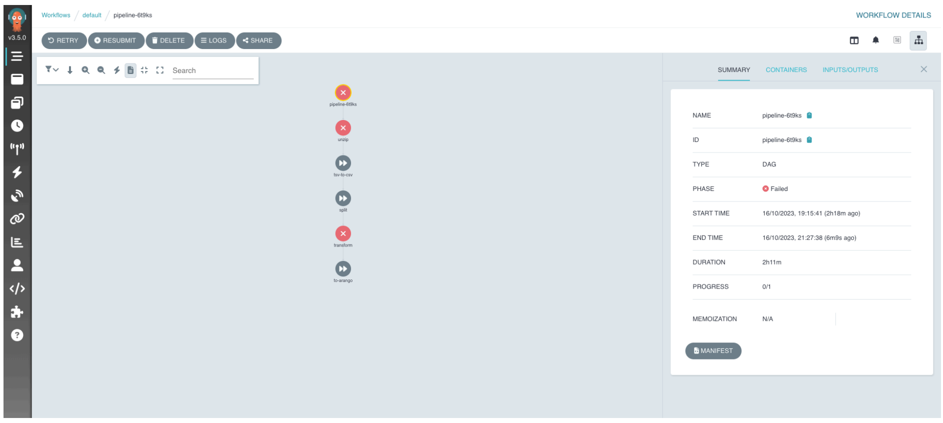
Running workflows 1
Pending 0 Succeeded 0
Failed 0 Error 0

	NAME	NAMESPACE	STARTED	FINISHED	DURATION	PROGRESS	MESSAGE	DETAILS	ARCHIVED
	pipeline-6t9ks	default	1h23m ago	-	1h23m	0/1	-	SHOW	false

Workflows cannot be globally sorted when paginated

50 results per page

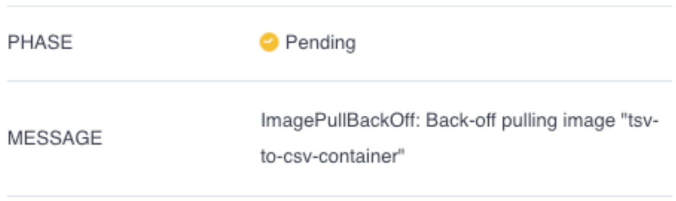




NAME	pipeline-6t9ks
ID	pipeline-6t9ks
TYPE	DAG
PHASE	Failed
START TIME	16/10/2023, 19:15:41 (2h18m ago)
END TIME	16/10/2023, 21:27:38 (6m9s ago)
DURATION	2h11m
PROGRESS	0/1
MEMOIZATION	N/A

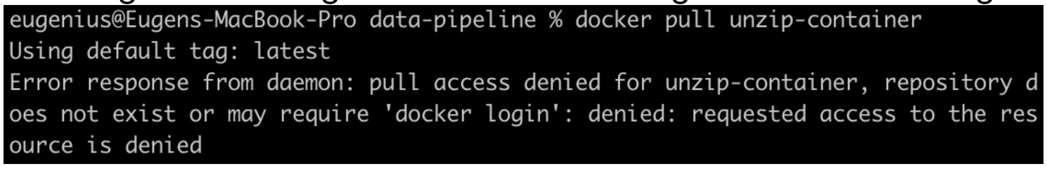
Often received error messages:

- ErrImagePull: rpc error: code = Unknown desc = Error response from daemon: pull access denied for unzip-container, repository does not exist or may require 'docker login': denied: requested access to the resource is denied
- Error response from daemon: pull access denied for unzip-container, repository does not exist or may require 'docker login': denied: requested access to the resource is denied
- Also when trying to pull docker image: Error response from daemon: pull access denied for unzip-container, repository does not exist or may require 'docker login': denied: requested access to the resource is denied
- also this here:

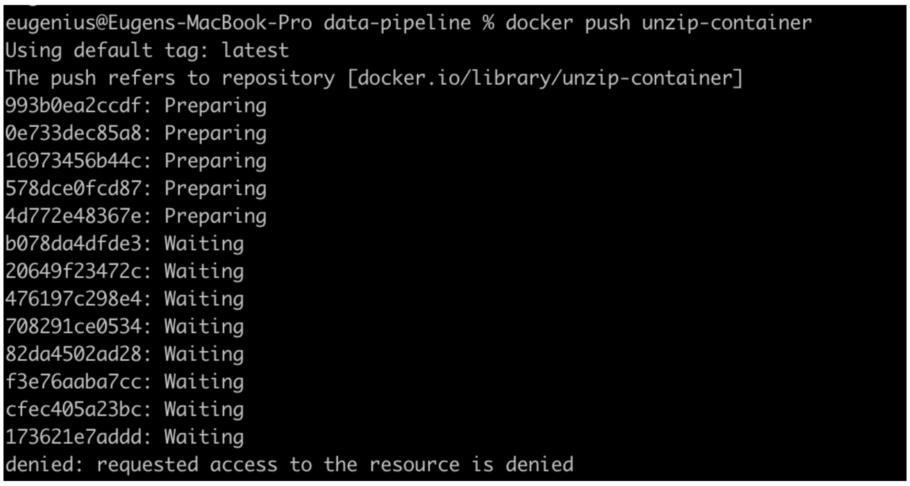


PHASE	Pending
MESSAGE	ImagePullBackOff: Back-off pulling image "tsv-to-csv-container"

- -> create Docker account and push docker image to Docker Hub so that Argo Workflow gets access to the image. See the following screenshots:

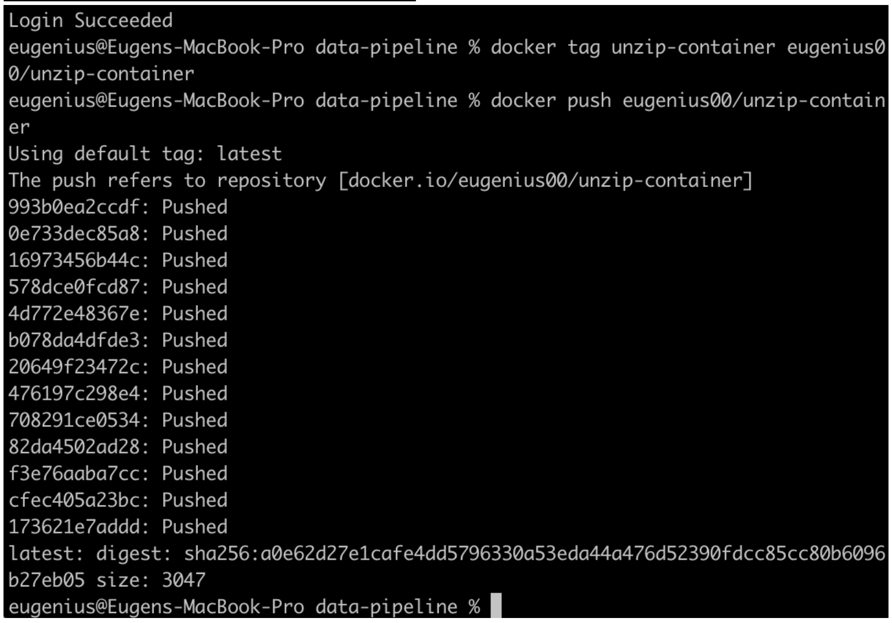


```
eugenius@Eugens-MacBook-Pro data-pipeline % docker pull unzip-container
Using default tag: latest
Error response from daemon: pull access denied for unzip-container, repository d
oes not exist or may require 'docker login': denied: requested access to the res
ource is denied
```

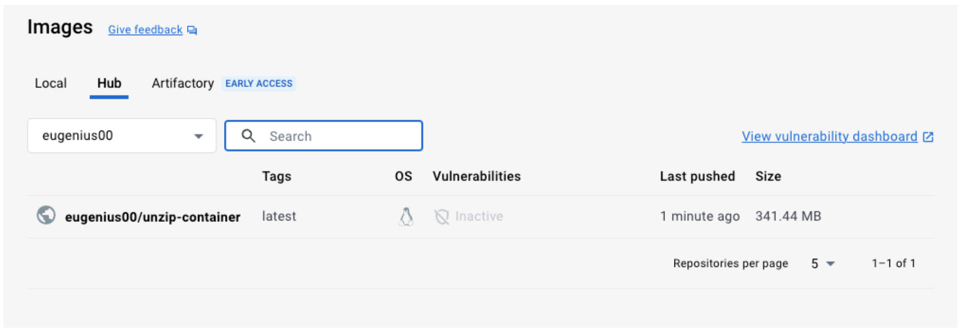


```
eugenius@Eugens-MacBook-Pro data-pipeline % docker push unzip-container
Using default tag: latest
The push refers to repository [docker.io/library/unzip-container]
993b0ea2ccdf: Preparing
0e733dec85a8: Preparing
16973456b44c: Preparing
578dce0fcd87: Preparing
4d772e48367e: Preparing
b078da4dfde3: Waiting
20649f23472c: Waiting
476197c298e4: Waiting
708291ce0534: Waiting
82da4502ad28: Waiting
f3e76aaba7cc: Waiting
cfec405a23bc: Waiting
173621e7addd: Waiting
denied: requested access to the resource is denied
```

https://stackoverflow.com/questions/41984399/denied-requested-access-to-the-resource-is-denied-docker



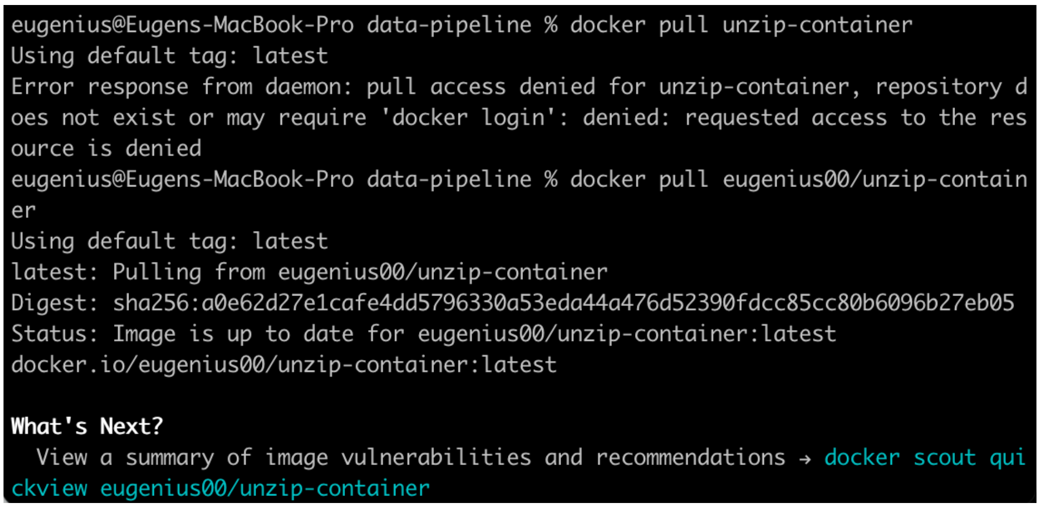
```
Login Succeeded
eugenius@Eugens-MacBook-Pro data-pipeline % docker tag unzip-container eugenius0
0/unzip-container
eugenius@Eugens-MacBook-Pro data-pipeline % docker push eugenius00/unzip-contain
er
Using default tag: latest
The push refers to repository [docker.io/eugenius00/unzip-container]
993b0ea2ccdf: Pushed
0e733dec85a8: Pushed
16973456b44c: Pushed
578dce0fcd87: Pushed
4d772e48367e: Pushed
b078da4dfde3: Pushed
20649f23472c: Pushed
476197c298e4: Pushed
708291ce0534: Pushed
82da4502ad28: Pushed
f3e76aaba7cc: Pushed
cfec405a23bc: Pushed
173621e7addd: Pushed
latest: digest: sha256:a0e62d27e1cafe4dd5796330a53eda44a476d52390fdcc85cc80b6096
b27eb05 size: 3047
eugenius@Eugens-MacBook-Pro data-pipeline %
```



	Tags	OS	Vulnerabilities	Last pushed	Size
eugenius00/unzip-container	latest		Inactive	1 minute ago	341.44 MB

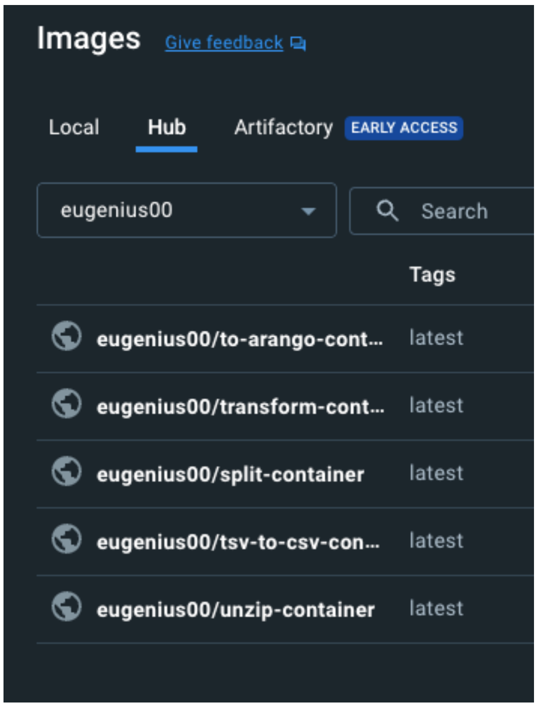
Repositories per page 5 1–1 of 1

Prove that when doing it with the image in the Docker Hub it works to pull the image, so Argo might have access to it

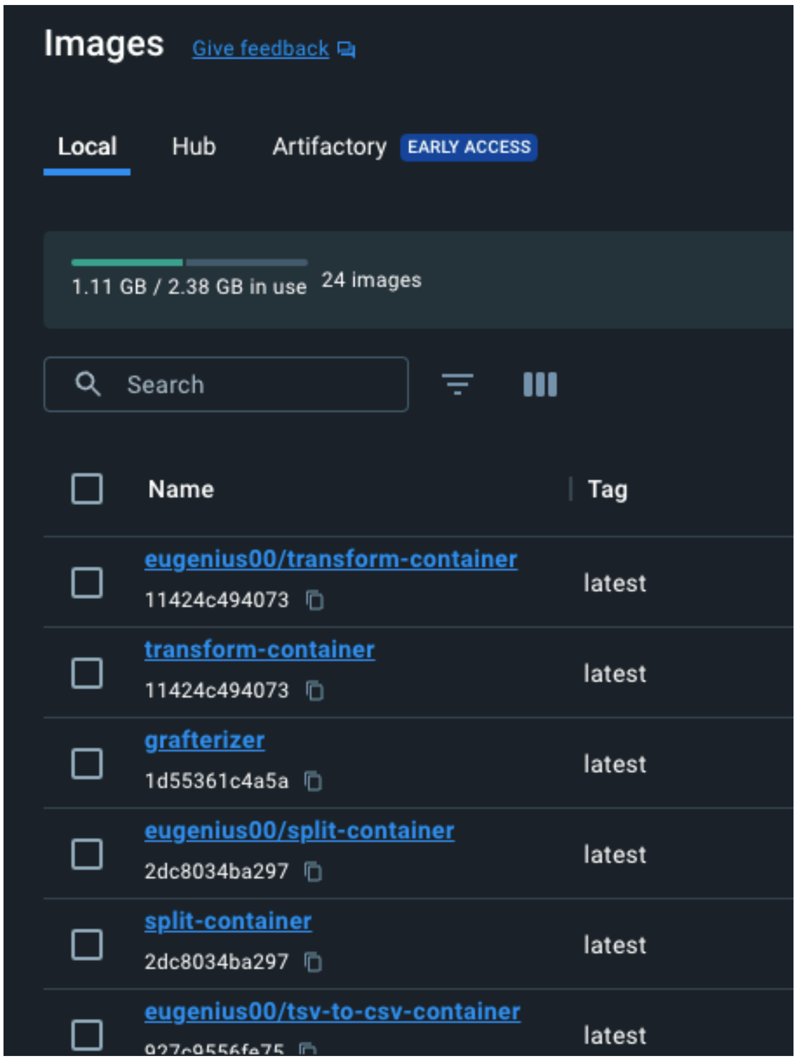


```
eugenius@Eugens-MacBook-Pro data-pipeline % docker pull unzip-container
Using default tag: latest
Error response from daemon: pull access denied for unzip-container, repository d
oes not exist or may require 'docker login': denied: requested access to the res
ource is denied
eugenius@Eugens-MacBook-Pro data-pipeline % docker pull eugenius00/unzip-contain
er
Using default tag: latest
latest: Pulling from eugenius00/unzip-container
Digest: sha256:a0e62d27e1cafe4dd5796330a53eda44a476d52390fdcc85cc80b6096b27eb05
Status: Image is up to date for eugenius00/unzip-container:latest
docker.io/eugenius00/unzip-container:latest

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout qui
ckview eugenius00/unzip-container
```



	Tags
eugenius00/to-arango-cont...	latest
eugenius00/transform-cont...	latest
eugenius00/split-container	latest
eugenius00/tsv-to-csv-con...	latest
eugenius00/unzip-container	latest



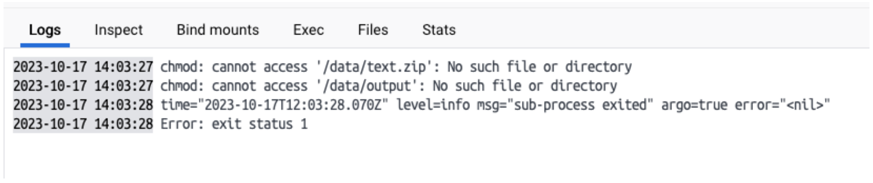
Images Give feedback

Local | Hub | Artifactory EARLY ACCESS

1.11 GB / 2.38 GB in use 24 images

Name	Tag
eugenius00/transform-container 11424c494073	latest
transform-container 11424c494073	latest
grafterizer 1d55361c4a5a	latest
eugenius00/split-container 2dc8034ba297	latest
split-container 2dc8034ba297	latest
eugenius00/tsv-to-csv-container 927c9556fe75	latest

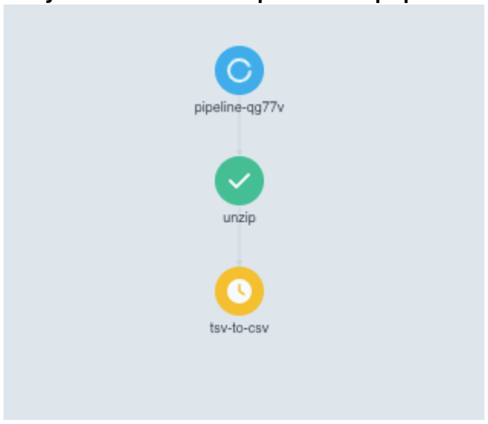
Often errors with files that cannot be found:



Logs | Inspect | Bind mounts | Exec | Files | Stats

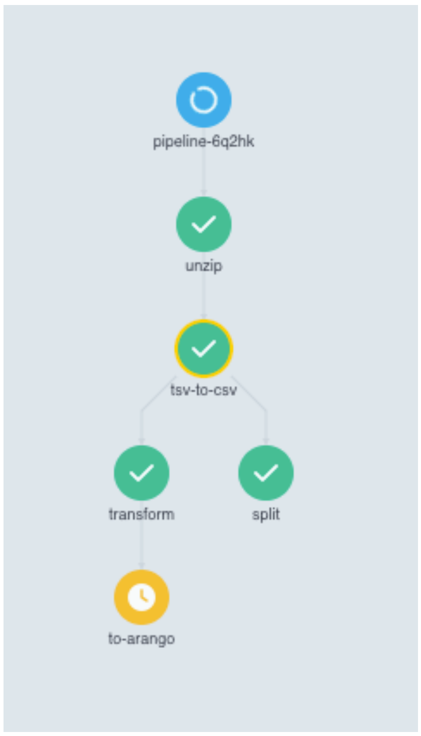
```
2023-10-17 14:03:27 chmod: cannot access '/data/text.zip': No such file or directory
2023-10-17 14:03:27 chmod: cannot access '/data/output': No such file or directory
2023-10-17 14:03:28 time="2023-10-17T12:03:28.070Z" level=info msg="sub-process exited" argo=true error="<nil>"
2023-10-17 14:03:28 Error: exit status 1
```

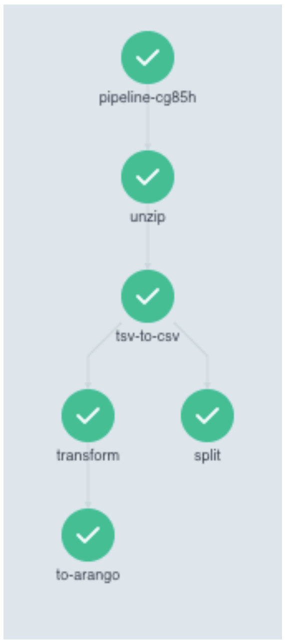
Adjusted and improved pipeline.yaml file



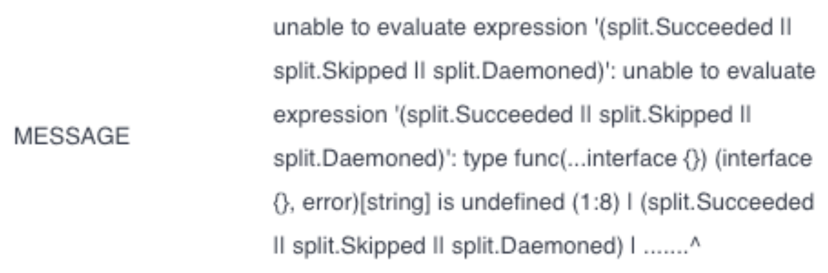
Similar approach with the other containers

transform with tsv-to-csv as dependency as we only need one file as output as



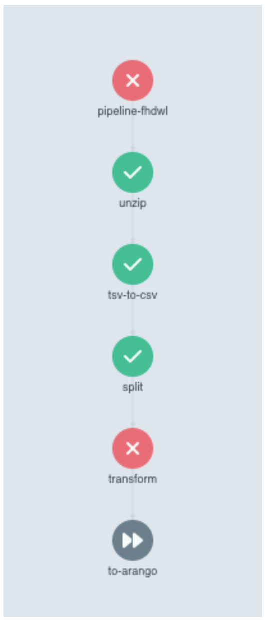


They pipeline works but the dependency of transform should be split which returns the following error

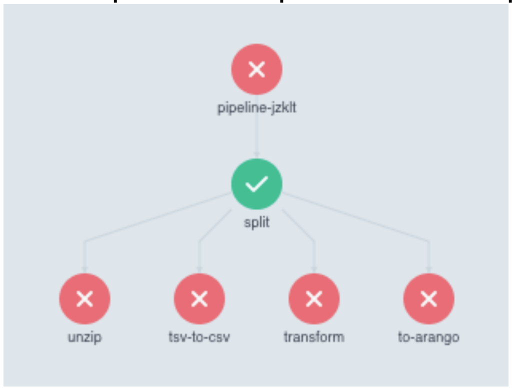


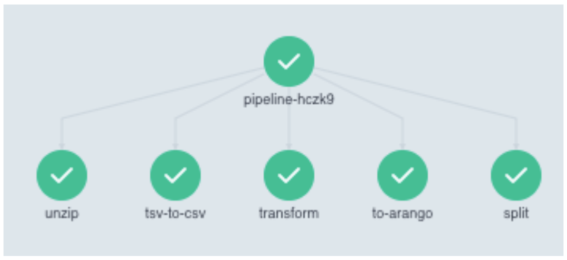
MESSAGE

```
unable to evaluate expression '(split.Succeeded ||
split.Skipped || split.Daemoned)': unable to evaluate
expression '(split.Succeeded || split.Skipped ||
split.Daemoned)': type func(...interface {}) (interface
{}, error)[string] is undefined (1:8) | (split.Succeeded
|| split.Skipped || split.Daemoned) | .......^
```

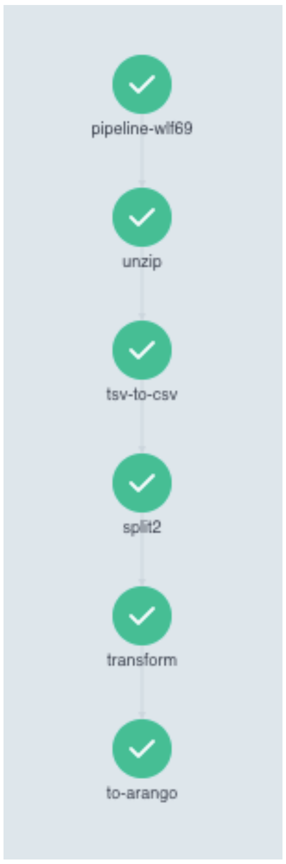


test to prove that problem is at split-container



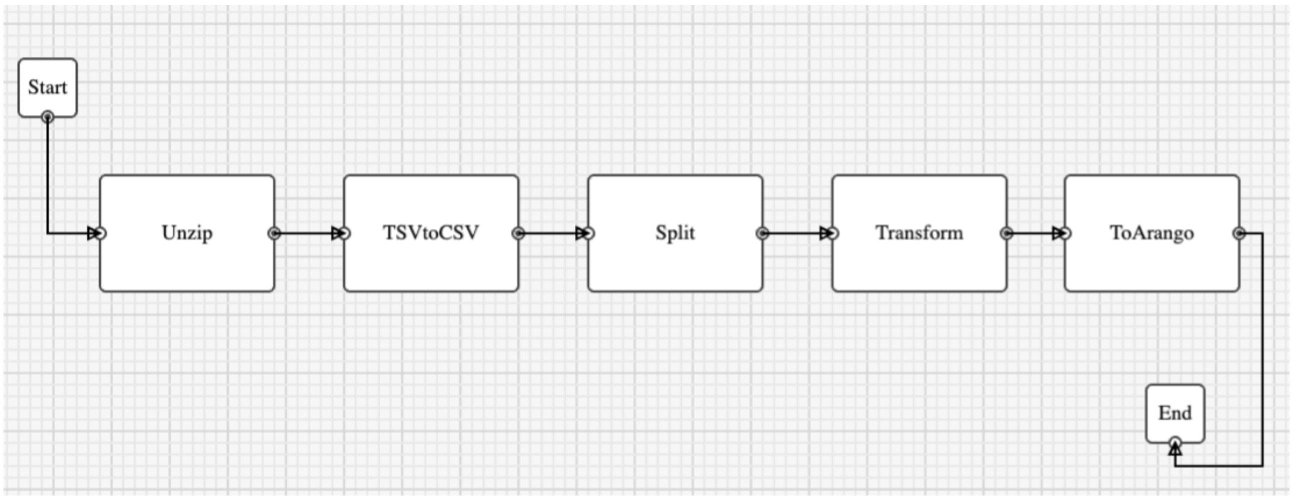


managed to fix the issue by renaming split



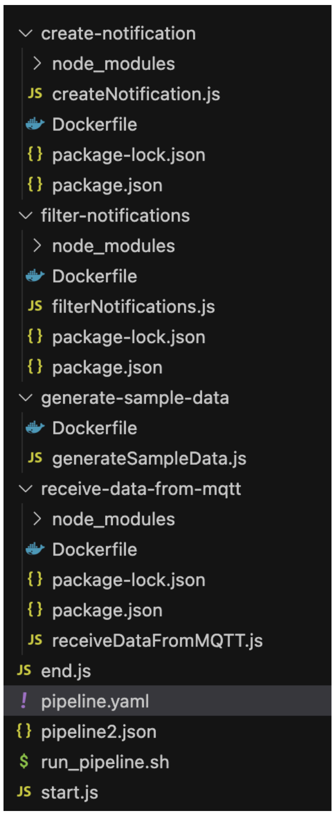
Task 3

DEF-PIPE tool of the DataCloud project
https://crowdserv.sys.kth.se/repo



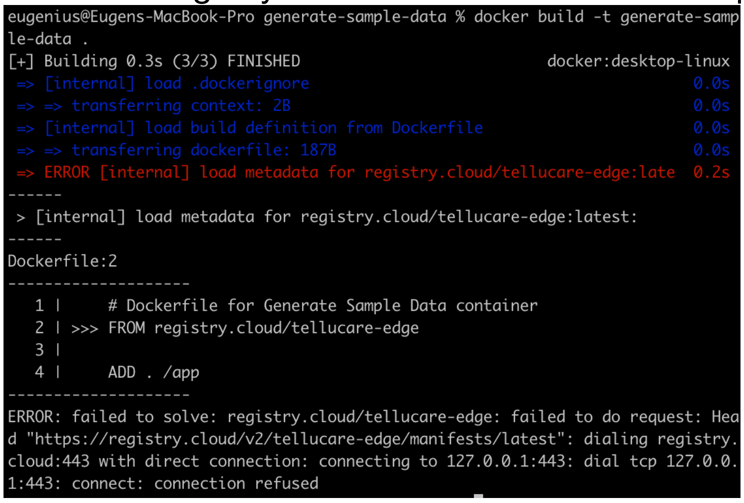
Task 4

Similar structure of the repo as in Task 1



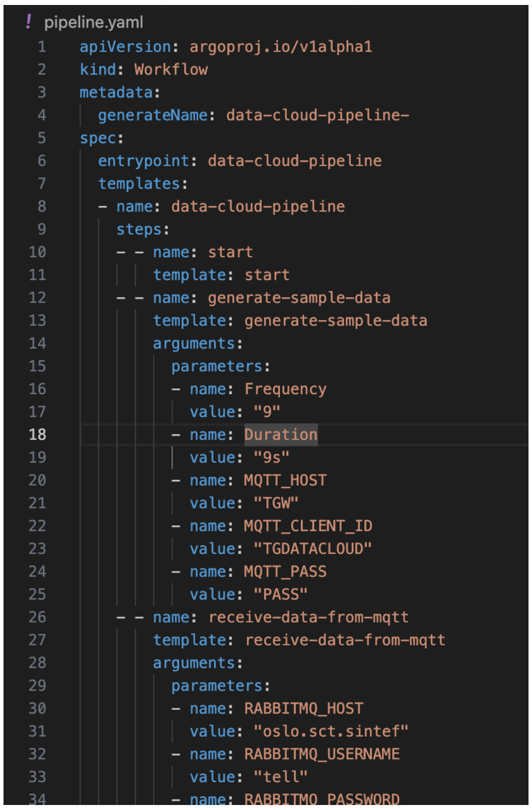
```
create-notification
  node_modules
  createNotification.js
  Dockerfile
  package-lock.json
  package.json
filter-notifications
  node_modules
  Dockerfile
  filterNotifications.js
  package-lock.json
  package.json
generate-sample-data
  Dockerfile
  generateSampleData.js
receive-data-from-mqtt
  node_modules
  Dockerfile
  package-lock.json
  package.json
  receiveDataFromMQTT.js
end.js
pipeline.yaml
pipeline2.json
run_pipeline.sh
start.js
```

When trying to build the Docker images I got this error because we don't have access to registry.cloud which is defined in the provided pipeline



```
eugenius@Eugens-MacBook-Pro generate-sample-data % docker build -t generate-samp
le-data .
[+] Building 0.3s (3/3) FINISHED                               docker:desktop-linux
 => [internal] load .dockerignore                                              0.0s
 => => transferring context: 2B                                                0.0s
 => [internal] load build definition from Dockerfile                           0.0s
 => => transferring dockerfile: 187B                                           0.0s
 => ERROR [internal] load metadata for registry.cloud/tellucare-edge:late  0.2s
------
 > [internal] load metadata for registry.cloud/tellucare-edge:latest:
------
Dockerfile:2
--------------------
   1 |     # Dockerfile for Generate Sample Data container
   2 | >>> FROM registry.cloud/tellucare-edge
   3 |
   4 |     ADD . /app
--------------------
ERROR: failed to solve: registry.cloud/tellucare-edge: failed to do request: Hea
d "https://registry.cloud/v2/tellucare-edge/manifests/latest": dialing registry.
cloud:443 with direct connection: connecting to 127.0.0.1:443: dial tcp 127.0.0.
1:443: connect: connection refused
```

Accordingly, to actually execute the pipeline this is something where more information would be needed but in this case as it doesn't need to be executed it is fine.
The yaml file again has been developed for Argo. Here is just a small excerpt



pipeline.yaml

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: data-cloud-pipeline-
spec:
  entrypoint: data-cloud-pipeline
  templates:
  - name: data-cloud-pipeline
    steps:
    - - name: start
        template: start
    - - name: generate-sample-data
        template: generate-sample-data
        arguments:
          parameters:
          - name: Frequency
            value: "9"
          - name: Duration
            value: "9s"
          - name: MQTT_HOST
            value: "TGW"
          - name: MQTT_CLIENT_ID
            value: "TGDATACLOUD"
          - name: MQTT_PASS
            value: "PASS"
    - - name: receive-data-from-mqtt
        template: receive-data-from-mqtt
        arguments:
          parameters:
          - name: RABBITMQ_HOST
            value: "oslo.sct.sintef"
          - name: RABBITMQ_USERNAME
            value: "tell"
          - name: RABBITMQ_PASSWORD
```