

Deep Dive #2

Algorithm containers

Anne Mickan

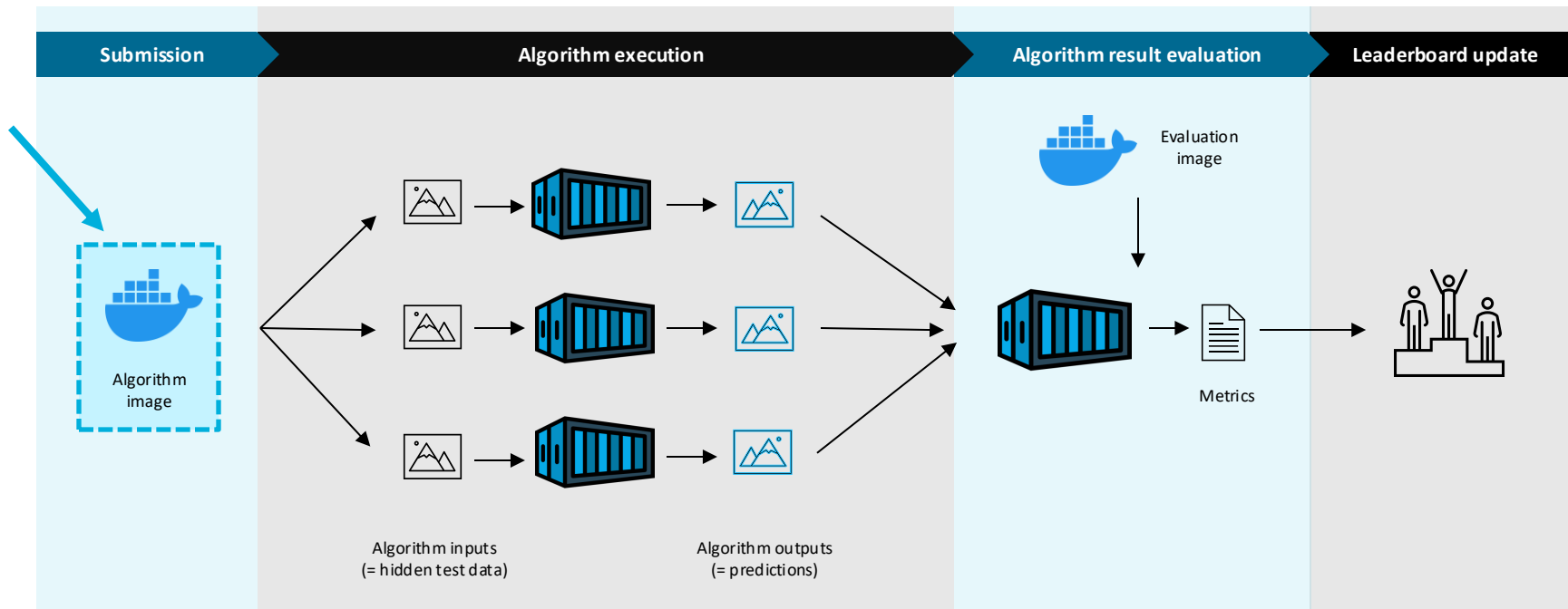
Research Software Engineer @grand-challenge.org

Radboudumc

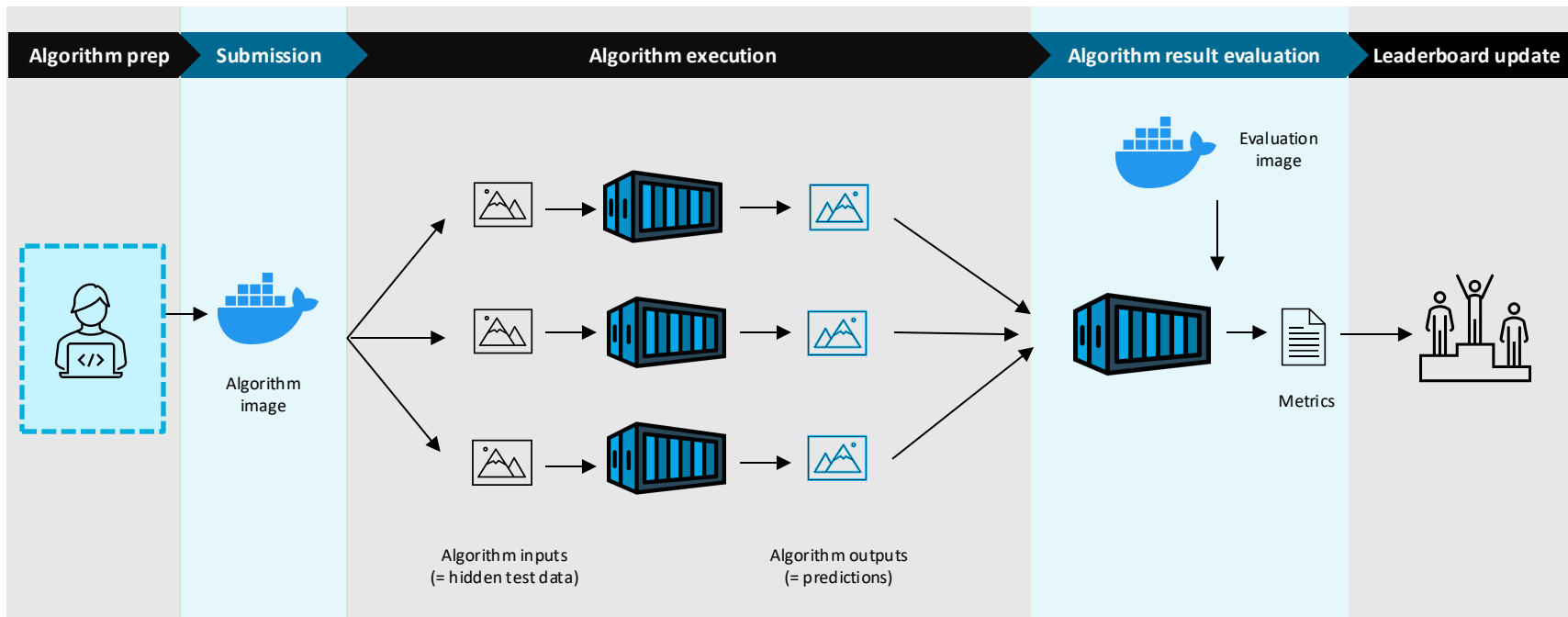
Agenda

| Time | Topic |
|---------------|--|
| 10:30 – 11:00 | Welcome and introduction to challenges on GC |
| 11:00 - 11:45 | Deep dive #1: uploading and managing hidden test data |
| 11:45 - 13:00 | Lunch Break |
| 13:00 - 14:15 | Deep dive #2: algorithm containers |
| 14:15 - 14:30 | Short Break |
| 14:30 - 16:00 | Deep dive #3: custom evaluation methods & leaderboard set-up |
| 16:00 - 17:00 | Wrap-up , Q&A |

Challenge submission workflow



Challenge submission workflow





~_ (ツ) _ /~

IT WORKS
on my machine

version 1.x

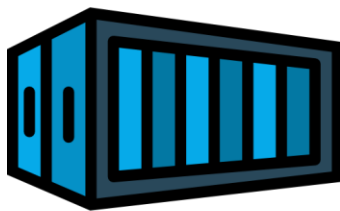


version 2.x

Solution: containers



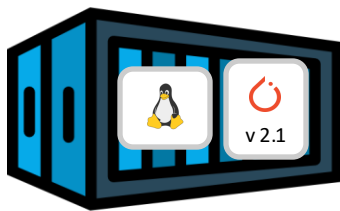
Solution: containers



Container

Lightweight, isolated environment that holds everything your software needs to run
(i.e., your code, dependencies, system tools, settings)

Solution: containers

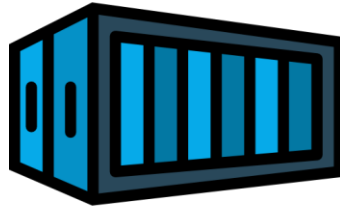


Container



Ensures that code runs anywhere

Solution: containers

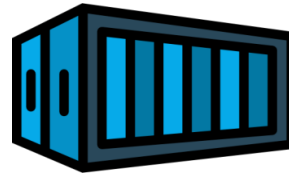


Docker Container

Solution: containers



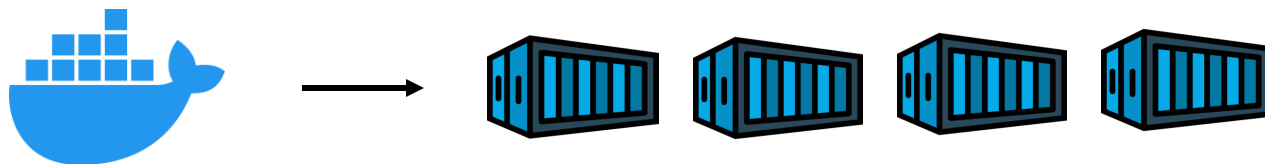
Docker Image



Docker Container

- Snapshot of your application
- Contains **instructions** and **ingredients** needed to create a container

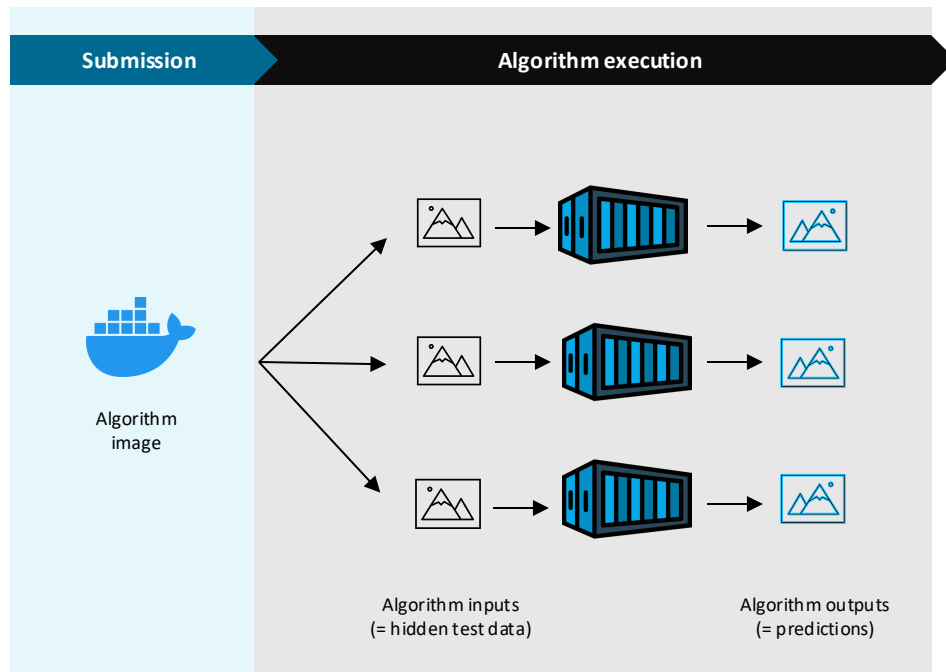
Solution: containers



Docker Image

- Snapshot of your application
- Contains **instructions** and **ingredients** needed to create a container

Challenge submission workflow



- Portability
- Consistency
- Scalability
- Efficiency

A concrete example

- Algorithm that segments vessels from a color fundus image

A concrete example

- Algorithm that segments vessels from a color fundus image

Inputs



Color fundus image

`/input/images/color-fundus/*`

42

Age in months

`/input/age-in-months.json`

A concrete example

- Algorithm that segments vessels from a color fundus image

Inputs



Color fundus image

42 Age in months

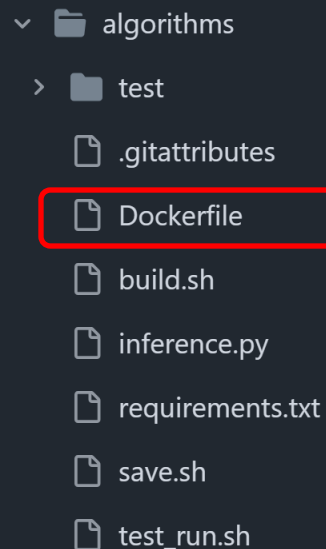
Output



Binary vessel segmentation

`/output/images/binary-vessel-segmentation/`*

A concrete example

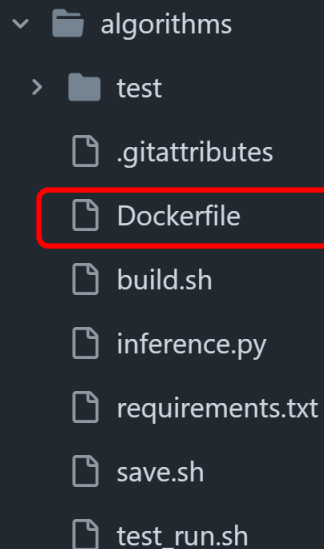


A file explorer interface showing a directory structure. The 'algorithms' folder is expanded, showing a 'test' subfolder and several files. The 'Dockerfile' file is highlighted with a red rectangular border.

- ▼ algorithms
 - > test
 - .gitattributes
 - Dockerfile
 - build.sh
 - inference.py
 - requirements.txt
 - save.sh
 - test_run.sh

```
FROM --platform=linux/amd64 python:3.11-slim AS example-algorithm-amd64
```


A concrete example



A file explorer window showing a directory structure. The 'algorithms' folder is expanded, showing a 'test' subfolder and several files. The 'Dockerfile' file is highlighted with a red rectangular border.

- algorithms
 - test
 - .gitattributes
 - Dockerfile
 - build.sh
 - inference.py
 - requirements.txt
 - save.sh
 - test_run.sh

```
FROM --platform=linux/amd64 python:3.11-slim AS example-algorithm-amd64

# Ensures that Python output to stdout/stderr is not buffered: prevents missing information when terminating
ENV PYTHONUNBUFFERED=1
```

A concrete example

```

▼  folder algorithms
    >  folder test
        .gitattributes
        Dockerfile
        build.sh
        inference.py
        requirements.txt
        save.sh
        test_run.sh

```

```

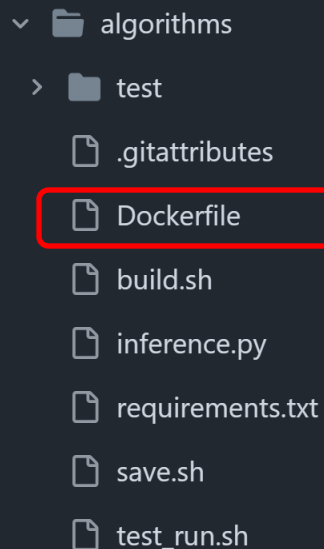
FROM --platform=linux/amd64 python:3.11-slim AS example-algorithm-amd64

# Ensures that Python output to stdout/stderr is not buffered: prevents missing information when terminating
ENV PYTHONUNBUFFERED=1

RUN groupadd -r user && useradd -m --no-log-init -r -g user user
USER user

```

A concrete example



A file explorer interface showing a directory structure. The 'algorithms' folder is expanded, showing a 'test' subfolder and several files. The 'Dockerfile' file is highlighted with a red rectangular border.

- algorithms
 - test
 - .gitattributes
 - Dockerfile**
 - build.sh
 - inference.py
 - requirements.txt
 - save.sh
 - test_run.sh

```
FROM --platform=linux/amd64 python:3.11-slim AS example-algorithm-amd64

# Ensures that Python output to stdout/stderr is not buffered: prevents missing information when terminating
ENV PYTHONUNBUFFERED=1

RUN groupadd -r user && useradd -m --no-log-init -r -g user user
USER user

WORKDIR /opt/app
```

A concrete example

✓  algorithms

>  test

 .gitattributes


 Dockerfile

 build.sh

 inference.py

 requirements.txt

 save.sh

 test_run.sh



```
FROM --platform=linux/amd64 python:3.11-slim AS example-algorithm-amd64
```

```
# Ensures that Python output to stdout/stderr is not buffered: prevents missing information when terminating  
ENV PYTHONUNBUFFERED=1
```

```
RUN groupadd -r user && useradd -m --no-log-init -r -g user user  
USER user
```

```
WORKDIR /opt/app
```

```
COPY --chown=user:user requirements.txt /opt/app/
```

A concrete example

```

  ✓  folder algorithms
    >  folder test
        .gitattributes
        Dockerfile
        build.sh
        inference.py
        requirements.txt
        save.sh
        test_run.sh

```

```

FROM --platform=linux/amd64 python:3.11-slim AS example-algorithm-amd64

# Ensures that Python output to stdout/stderr is not buffered: prevents missing information when terminating
ENV PYTHONUNBUFFERED=1

RUN groupadd -r user && useradd -m --no-log-init -r -g user user
USER user

WORKDIR /opt/app

COPY --chown=user:user requirements.txt /opt/app/

# You can add any Python dependencies to requirements.txt
RUN python -m pip install \
    --user \
    --no-cache-dir \
    --no-color \
    --requirement /opt/app/requirements.txt

```

A concrete example

```

v  folder algorithms
>  folder test
   file .gitattributes
   file Dockerfile
   file build.sh
   file inference.py
   file requirements.txt
   file save.sh
   file test_run.sh

```

```

FROM --platform=linux/amd64 python:3.11-slim AS example-algorithm-amd64

# Ensures that Python output to stdout/stderr is not buffered: prevents missing information when terminating
ENV PYTHONUNBUFFERED=1

RUN groupadd -r user && useradd -m --no-log-init -r -g user user
USER user

WORKDIR /opt/app

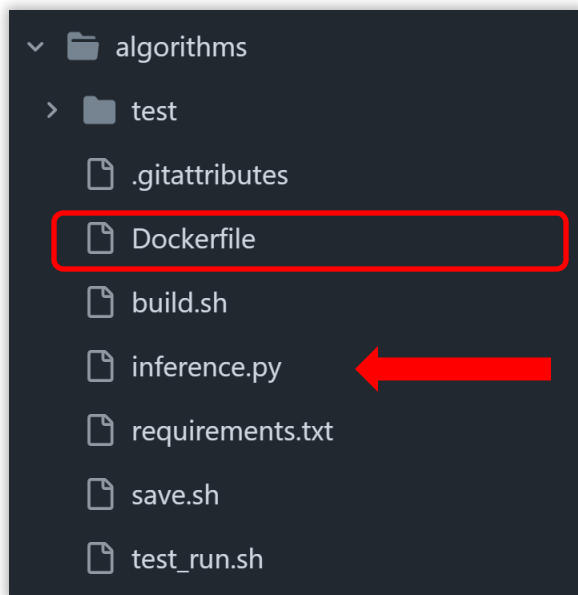
COPY --chown=user:user requirements.txt /opt/app/

# You can add any Python dependencies to requirements.txt
RUN python -m pip install \
    --user \
    --no-cache-dir \
    --no-color \
    --requirement /opt/app/requirements.txt

COPY --chown=user:user inference.py /opt/app/

```

A concrete example



```
FROM --platform=linux/amd64 python:3.11-slim AS example-algorithm-amd64

# Ensures that Python output to stdout/stderr is not buffered: prevents missing information when terminating
ENV PYTHONUNBUFFERED=1

RUN groupadd -r user && useradd -m --no-log-init -r -g user user
USER user

WORKDIR /opt/app

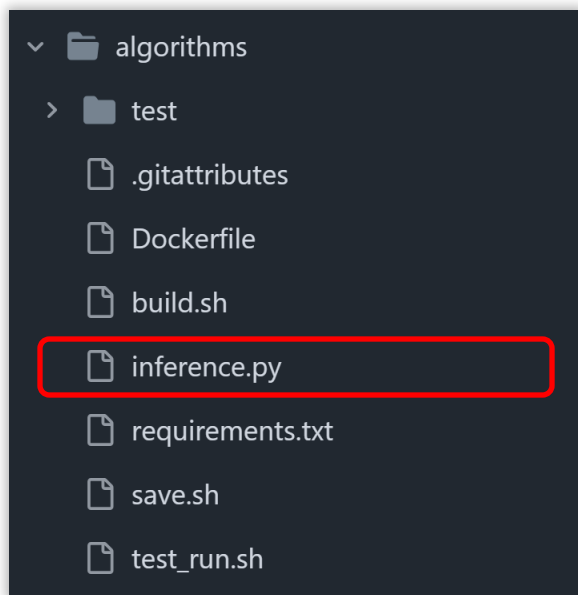
COPY --chown=user:user requirements.txt /opt/app/

# You can add any Python dependencies to requirements.txt
RUN python -m pip install \
    --user \
    --no-cache-dir \
    --no-color \
    --requirement /opt/app/requirements.txt

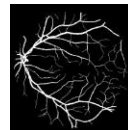
COPY --chown=user:user inference.py /opt/app/

ENTRYPOINT ["python", "inference.py"]
```

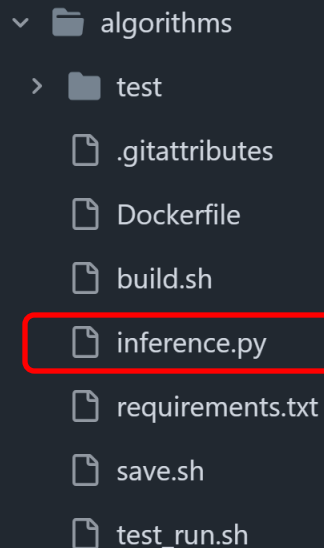
A concrete example



42



A concrete example



A file explorer window showing a directory structure. The 'algorithms' folder is expanded, showing a 'test' subfolder and several files. The 'inference.py' file is highlighted with a red rectangular border.

- algorithms
 - test
 - .gitattributes
 - Dockerfile
 - build.sh
 - inference.py
 - requirements.txt
 - save.sh
 - test_run.sh

```
## Read the inputs
# Fundus image
input_fundus_image = load_image(
    location= Path("/input") / "images" / "color-fundus",
)
# Dummy patient metadata which we will ignore
# This is just to demonstrate the possibility of having multiple different inputs
input_age_in_months = load_json_file(
    location=Path("/input") / "age-in-months.json",
)
```

A concrete example

```

v algorithms
  > test
    .gitattributes
    Dockerfile
    build.sh
    inference.py
    requirements.txt
    save.sh
    test_run.sh

```

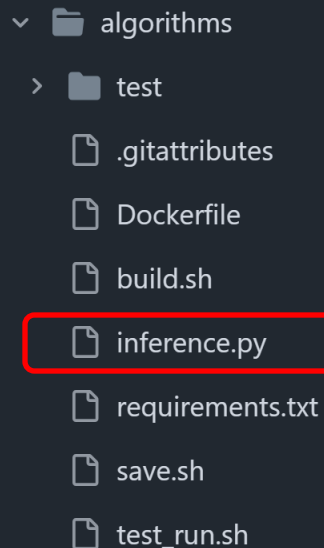
```

## Read the inputs
# Fundus image
input_fundus_image = load_image(
    location= Path("/input") / "images" / "color-fundus",
)
# Dummy patient metadata which we will ignore
# This is just to demonstrate the possibility of having multiple different inputs
input_age_in_months = load_json_file(
    location=Path("/input") / "age-in-months.json",
)

# Process inputs and generate predictions:
# For this example, we will simply convert the image to a binary mask
output_vessel_segmentation = convert_to_binary_mask(image=input_fundus_image)

```

A concrete example



A file explorer window showing the project structure. The 'algorithms' folder is expanded, showing a 'test' subfolder and several files. The 'inference.py' file is highlighted with a red rectangular border.

- algorithms
 - test
 - .gitattributes
 - Dockerfile
 - build.sh
 - inference.py
 - requirements.txt
 - save.sh
 - test_run.sh

```
## Read the inputs
# Fundus image
input_fundus_image = load_image(
    location=Path("/input") / "images" / "color-fundus",
)
# Dummy patient metadata which we will ignore
# This is just to demonstrate the possibility of having multiple different inputs
input_age_in_months = load_json_file(
    location=Path("/input") / "age-in-months.json",
)

# Process inputs and generate predictions:
# For this example, we will simply convert the image to a binary mask
output_vessel_segmentation = convert_to_binary_mask(image=input_fundus_image)

# Save your output
write_image_to_file(
    location=Path("/output") / "images/binary-vessel-segmentation",
    image=output_vessel_segmentation,
)
```

A concrete example

```

v  folder algorithms
  > folder test
    .gitattributes
    Dockerfile
    build.sh
    inference.py
    requirements.txt
    save.sh
    test_run.sh

```

```

## Read the inputs
# Fundus image
input_fundus_image = load_image(
    location= Path("/input") / "images" / "color-fundus",
)
# Dummy patient metadata which we will ignore
# This is just to demonstrate the possibility of having multiple different inputs
input_age_in_months = load_json_file(
    location=Path("/input") / "age-in-months.json",
)

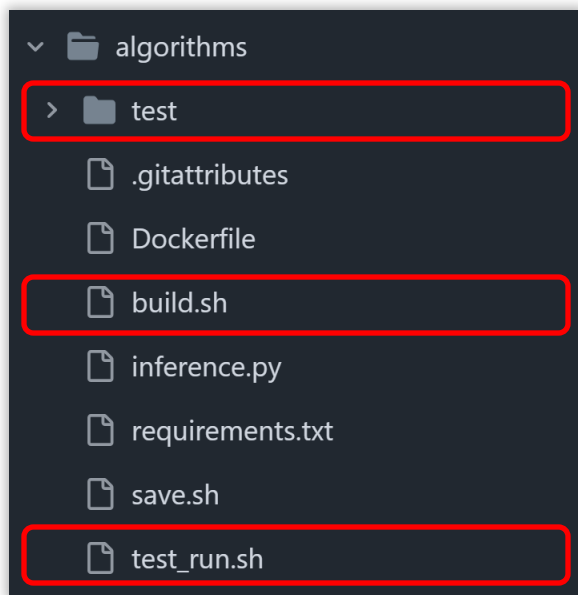
# Process inputs and generate predictions:
# For this example, we will simply convert the image to a binary mask
output_vessel_segmentation = convert_to_binary_mask(image=input_fundus_image)

# Save your output
write_image_to_file(
    location=Path("/output") / "images/binary-vessel-segmentation",
    image=output_vessel_segmentation,
)

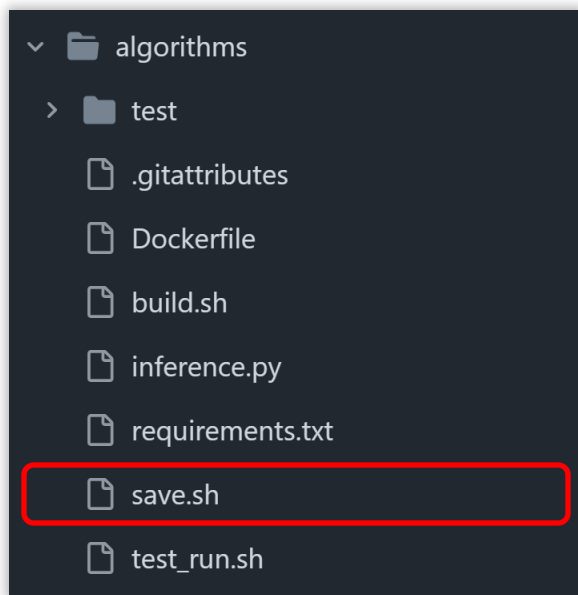
return 0

```

A concrete example

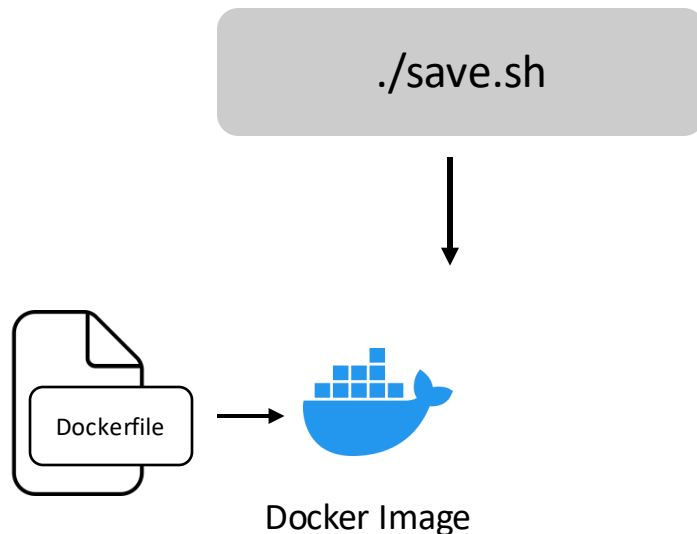
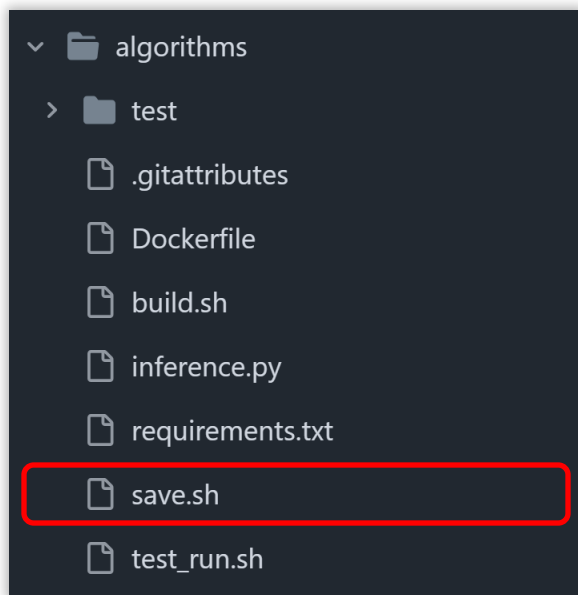


A concrete example

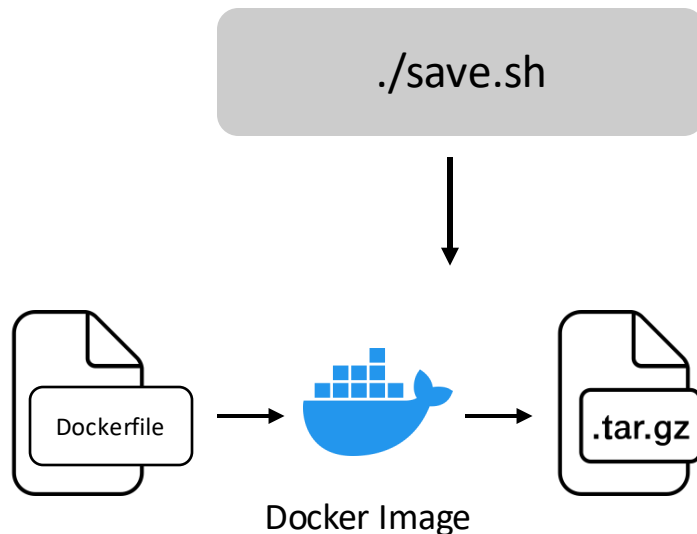
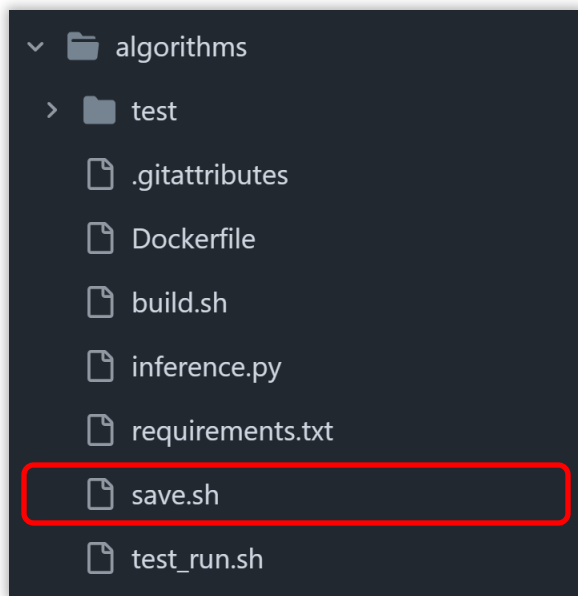


`./save.sh`

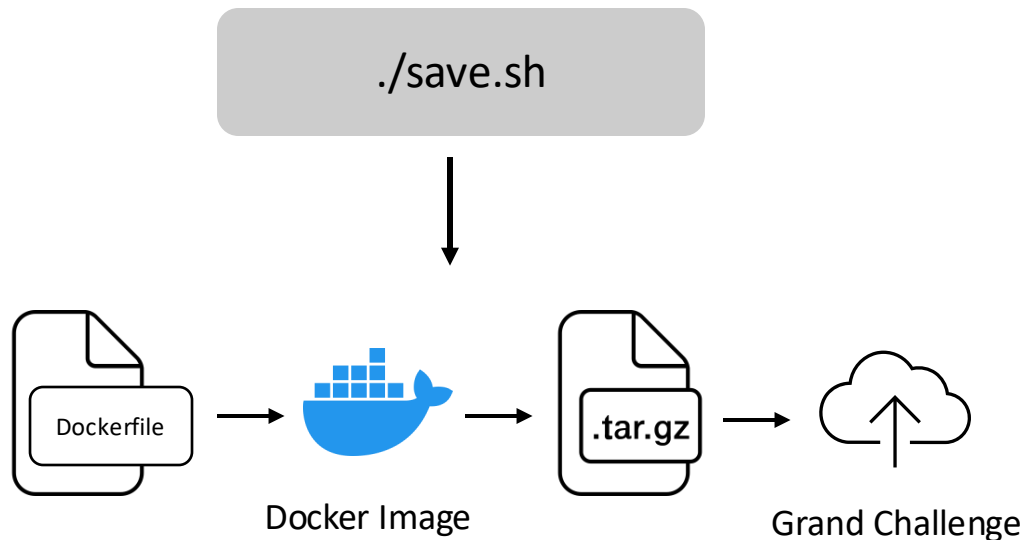
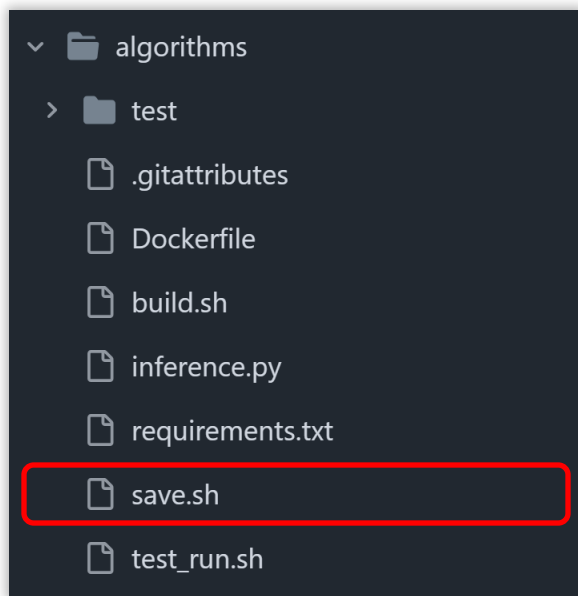
A concrete example



A concrete example



A concrete example



Algorithms on GC

Algorithms

We have made several machine learning algorithms available that you can try out by uploading your own anonymised medical imaging data. Please [contact us](#) if you would like to make your own algorithm available here.

+ Add a new algorithm

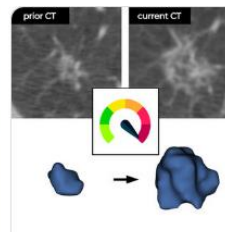
Filter Algorithms

4912 algorithms found



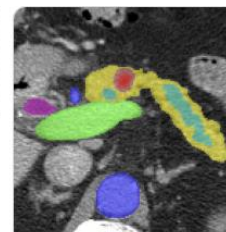
DataScientX (N. Debs, A. Routier, et al.; France) algorithm trained on PI-CAI: Private and Public Training Dataset

[🔗](#) [📄](#) [PI-CAI](#)



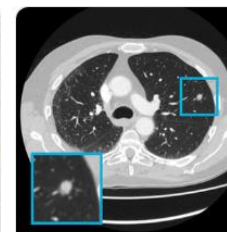
Deep learning to estimate pulmonary nodule malignancy risk using a current and a prior CT image

[📄](#) [Article](#)



Pancreatic Ductal Adenocarcinoma Detection in CT

[📄](#) [Article](#)



Lung nodule detection for routine clinical CT scans

[📄](#) [Article](#)

Creating an algorithm on GC

[Info](#) [Forum](#) [Teams](#) [Submit](#) [Leaderboard](#) [Statistics](#) [Admin](#)

[Test Phase](#) [All Submissions](#)

Test Phase Submission

Create a new submission

The algorithms for this phase need to implement all of the following input-output combinations: [+ Manage your algorithms](#)

| | Inputs | Outputs |
|---|--|---|
| 1 | <ul style="list-style-type: none">Color Fundus Image (Image)Age in months (Integer) | <ul style="list-style-type: none">Binary vessel segmentation (Segmentation) |

Algorithm*

Select one of your algorithms to submit as a solution to this phase. See above for information regarding the necessary configuration of the algorithm.

Save

Creating an algorithm on GC

Title*

Threshold Algorithm

Description

Short description of this algorithm, max 1024 characters. This will appear in the info modal on the algorithm overview list.

Job requires gpu type

No GPU

What GPU to attach to this algorithms inference jobs

Job requires memory gb*

8

How much memory to assign to this algorithms inference jobs

Uploading an Algorithm Image

Information

Containers

Models

Editors

Users

Requests 0

Try-out Algorithm

Results

Container Images


Link GitHub Repo


Upload a Container


To re-activate a previously uploaded container image, click on the info button next to it and then on "Make active image for algorithm".


| | | | |
|--------------|---|------------------|----------|
| <div>ⓘ</div> | Algorithm Image 9e6c6903 (SHA256: 103e844d) | Import Completed | Inactive |
| <div>ⓘ</div> | Algorithm Image eff86d2b (SHA256: 9b4e33dc) | Import Completed | Active |


Uploading an Algorithm Image


 Information


 Containers


 Models

 Editors


 Users


 Requests 0

 Try-out Algorithm


 Results

Container Images

 Link GitHub Repo


 Upload a Container

To re-activate a previously uploaded container image, click on the info button next to it and then on "Make active image for algorithm".

 Algorithm Image 9e6c6903 (SHA256: 103e844d)

Import Completed

Inactive

 Algorithm Image eff86d2b (SHA256: 9b4e33dc)

Import Completed

Active

Uploading an Algorithm Image

Information

Containers

Models

Editors

Users

Requests 0

Try-out Algorithm

Results

Container Images


Link GitHub Repo


Upload a Container


To re-activate a previously uploaded container image, click on the info button next to it and then on "Make active image for algorithm".


| | | | |
|--------------|---|------------------|----------|
| <div>ⓘ</div> | Algorithm Image 9e6c6903 (SHA256: 103e844d) | Import Completed | Inactive |
| <div>ⓘ</div> | Algorithm Image eff86d2b (SHA256: 9b4e33dc) | Import Completed | Active |


Uploading an Algorithm Image


 Information


 Containers


 Models

 Editors


 Users


 Requests 0

 Try-out Algorithm



 Results

Container Images


 Link GitHub Repo


 Upload a Container


To re-activate a previously uploaded container image, click on the info button next to it and then on "Make active image for algorithm".


| | | | |
|---|---|------------------|----------|
|  | Algorithm Image 9e6c6903 (SHA256: 103e844d) | Import Completed | Inactive |
|  | Algorithm Image eff86d2b (SHA256: 9b4e33dc) | Import Completed | Active |


Uploading an Algorithm Image


 Information


 Containers


 Models

 Editors


 Users


 Requests 0

 Try-out Algorithm



 Results

Container Images


 Link GitHub Repo


 Upload a Container


To re-activate a previously uploaded container image, click on the info button next to it and then on "Make active image for algorithm".


| | | | |
|---|---|------------------|----------|
|  | Algorithm Image 9e6c6903 (SHA256: 103e844d) | Import Completed | Inactive |
|  | Algorithm Image eff86d2b (SHA256: 9b4e33dc) | Import Completed | Active |


Uploading an Algorithm Image


 Information


 Containers


 Models

 Editors


 Users


 Requests 0

 Try-out Algorithm



 Results

Container Images

 Link GitHub Repo

 Upload a Container

To re-activate a previously uploaded container image, click on the info button next to it and then on "Make active image for algorithm".

| | | | |
|---|---|------------------|----------|
|  | Algorithm Image 9e6c6903 (SHA256: 103e844d) | Import Completed | Inactive |
|  | Algorithm Image eff86d2b (SHA256: 9b4e33dc) | Import Completed | Active |

Uploading an Algorithm Image

Try-out Algorithm

Select the data that you would like to run the algorithm on.

As an editor for this algorithm you can test and debug your algorithm in total 5 times per unique algorithm image. You share these credits with all other editors of this algorithm. Once you have reached the limit, any extra jobs will be deducted from your personal algorithm credits, of which you get 1000 per month. Using this algorithm requires 20 credits per job. You can currently create up to 55 jobs for this algorithm.

Color-Fundus-Image*

Choose data source...

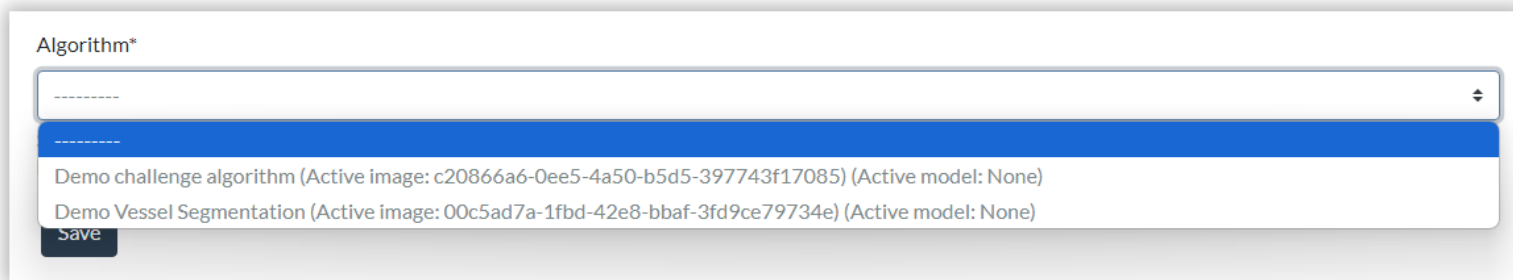
Age-In-Months*

Age in months

Submit

By running this algorithm you agree to the [General Terms of Service](#).

Make a submission



The screenshot shows a web form for submitting an algorithm. At the top, there is a label "Algorithm*" followed by a text input field containing several dashes. Below the input field is a dropdown menu with a blue highlight on the first option. The dropdown list contains two entries: "Demo challenge algorithm (Active image: c20866a6-0ee5-4a50-b5d5-397743f17085) (Active model: None)" and "Demo Vessel Segmentation (Active image: 00c5ad7a-1fbd-42e8-bbaf-3fd9ce79734e) (Active model: None)". At the bottom left of the form, there is a dark blue button labeled "Save".

- GC allows one submission per unique (SHA256) algorithm image per challenge phase
- Container won't have access to internet

Questions & Answers



Hands-on #2: algorithms

1. Create an algorithm on GC
2. Build, export and upload an algorithm image
3. Try-out algorithm on GC

Agenda

| Time | Topic |
|---------------|--|
| 10:30 – 11:00 | Welcome and introduction to challenges on GC |
| 11:00 - 11:45 | Deep dive #1: uploading and managing hidden test data |
| 11:45 - 13:00 | Lunch Break |
| 13:00 - 14:15 | Deep dive #2: algorithm containers |
| 14:15 - 14:30 | Short Break |
| 14:30 - 16:00 | Deep dive #3: custom evaluation methods & leaderboard set-up |
| 16:00 - 17:00 | Wrap-up , Q&A |