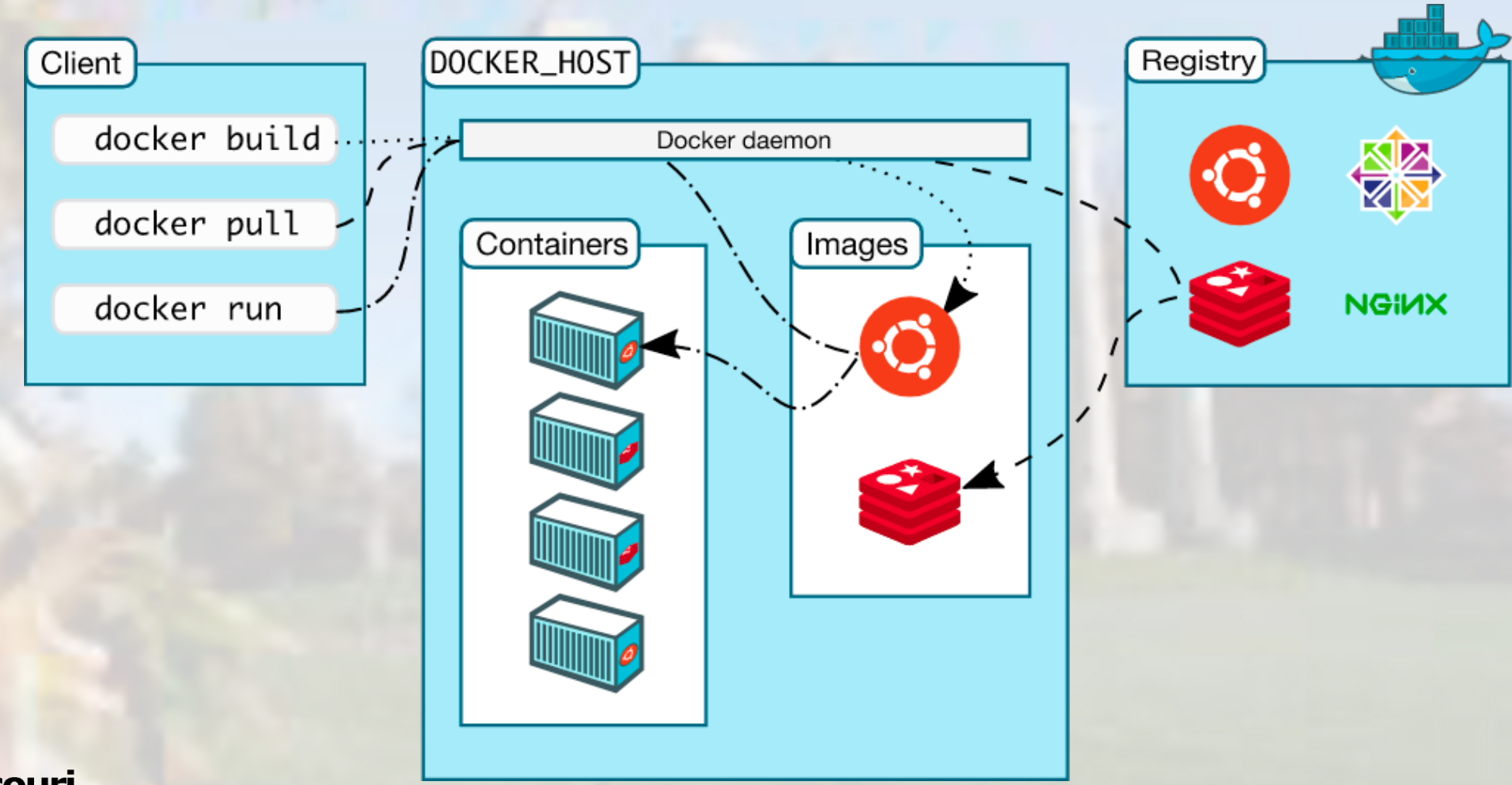
Bring Your Own Container Image: GitLab CI/CD and Automating Image Creation for Kubernetes

University of Missouri

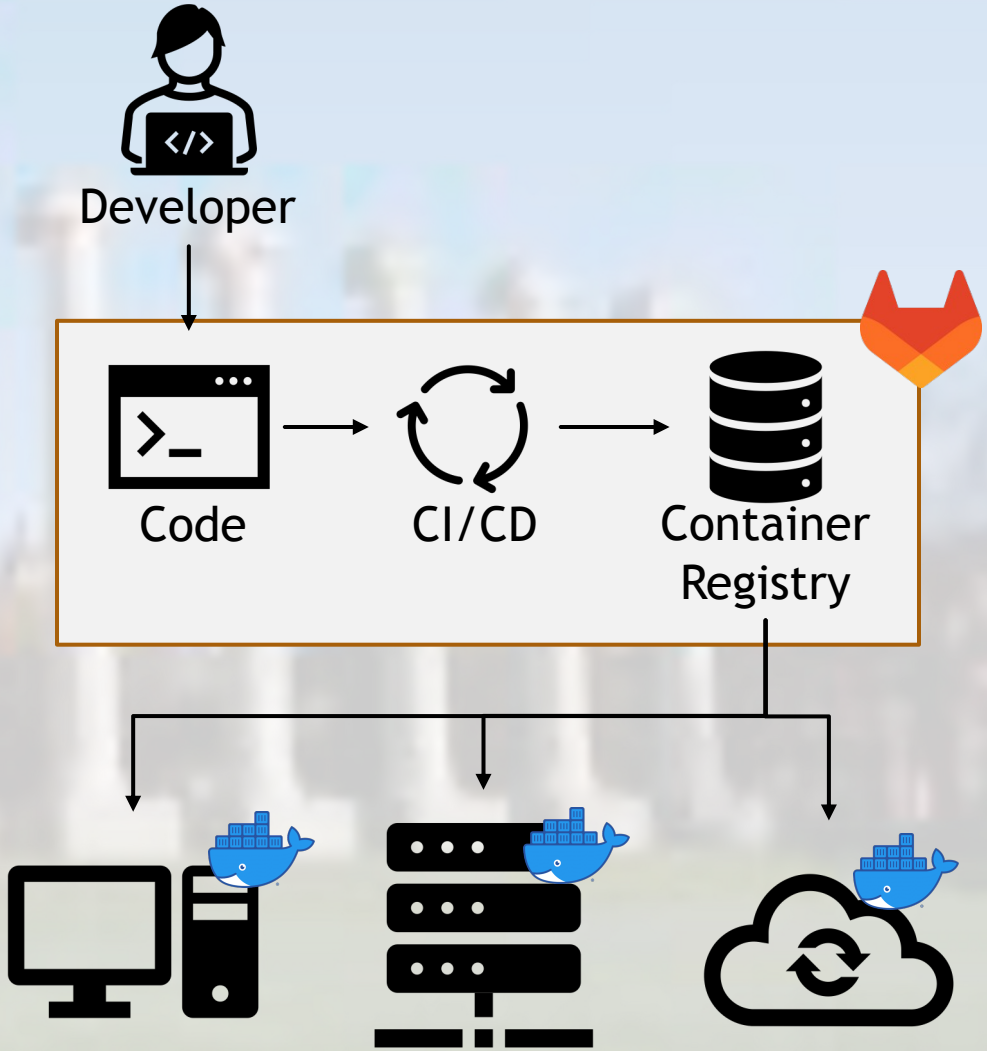
Containerization & Kubernetes

- Kubernetes requires a publicly available image, and we need the flexibility to use own our images
- **Nautilus** Gitlab
  - VCS
  - Create easily maintained and developed custom images using GitLab CI



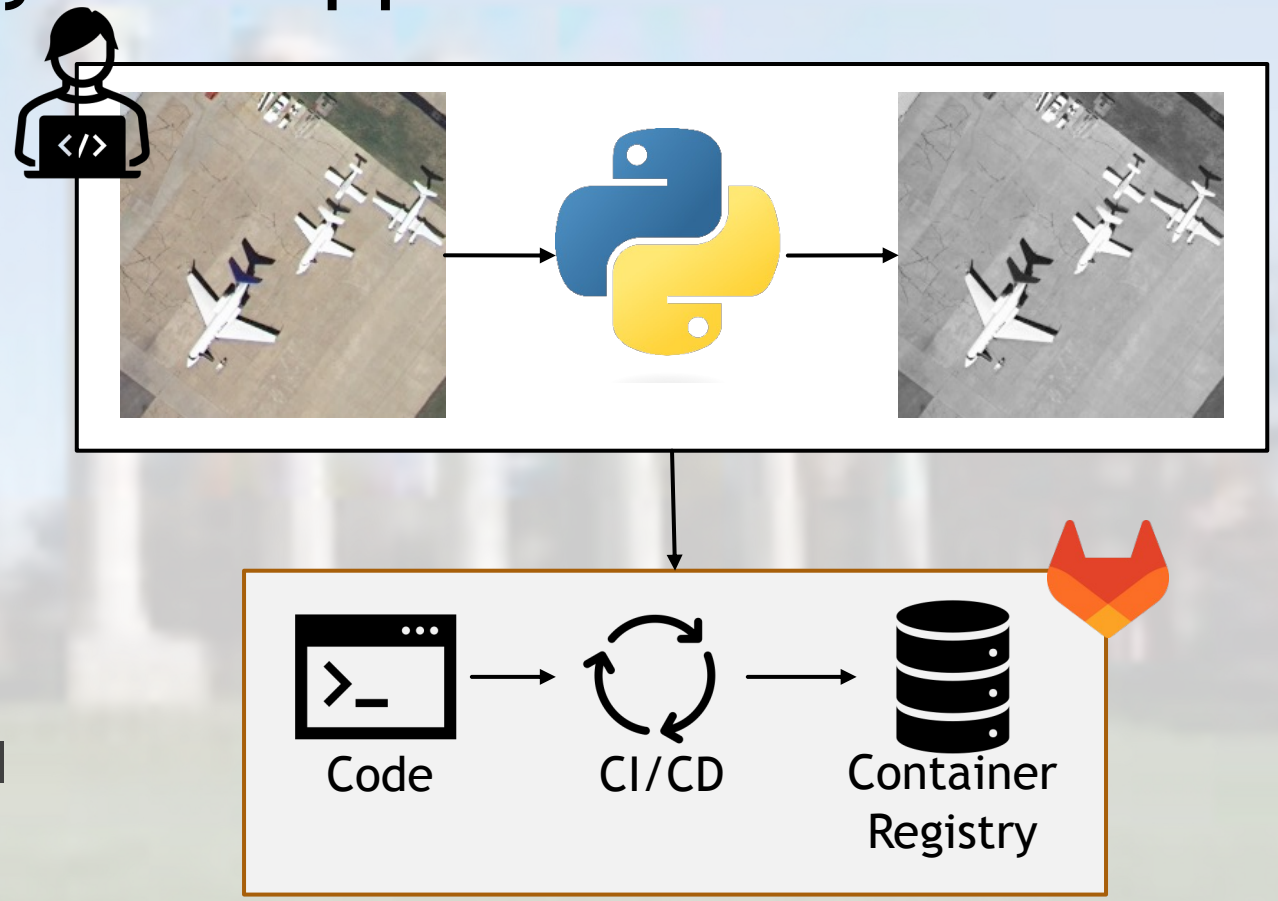
Automation with GitLab

- GitLab is a common tool for CI/CD with Docker because it contains VCS, CI/CD and a built-in image registry with access restrictions
  - Only those added to the repository can access the code, CI/CD build instructions, and the container registry
- In our previous example, the VCS, CI/CD, and Registry providers were all independent, but GitLab contains all three of these on a per-repository or per-group basis



Automation in Action: GitLab CI + Registry for Python Application

- We configure GitLab to utilize CI/CD and the Container Registry to automatically build and push a container each time we push a new commit to the repository
- Each commit has its own tag, and latest is automatically updated to point to the most recent commit
- Let's view the repository, the Gitlab CI config, and the container registry!

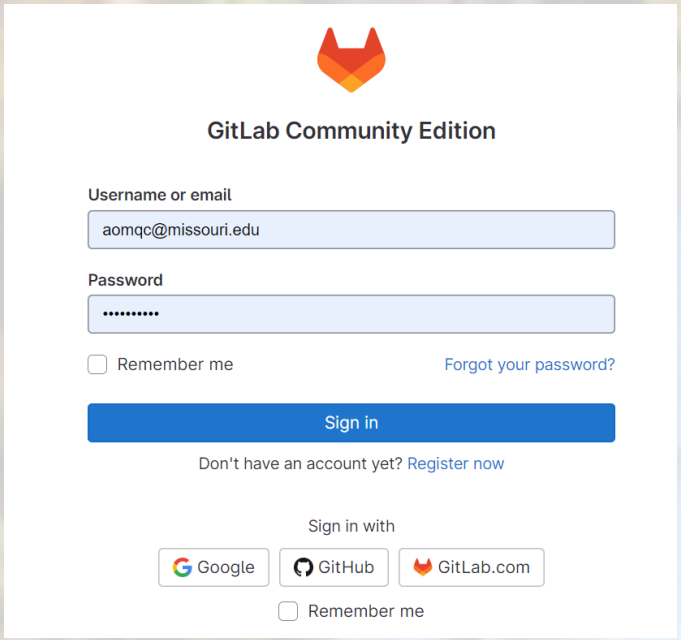


Containerization
Nautilus GitLab

https://gitlab.nrp-nautilus.io/

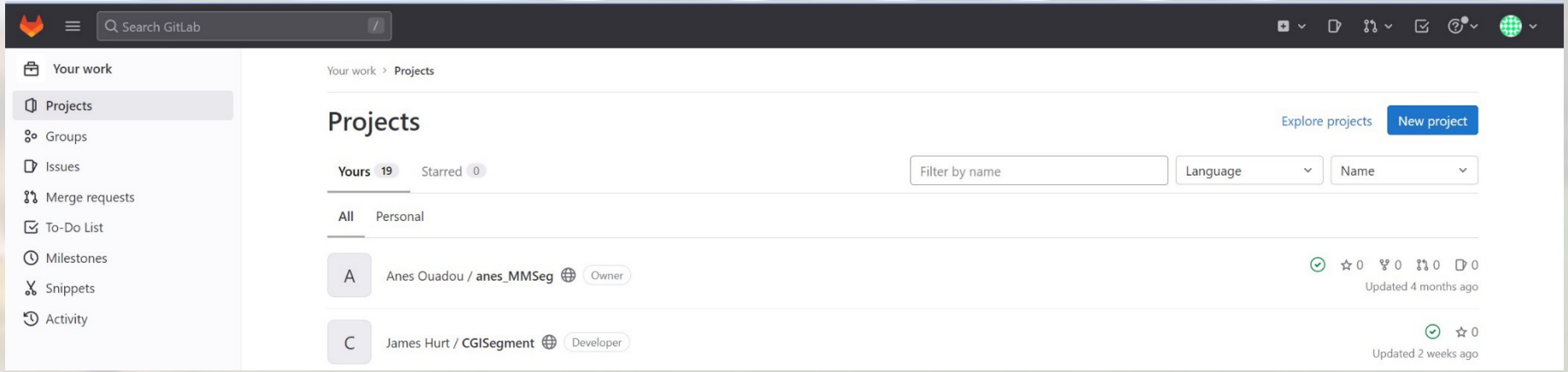
- Git Version Control System
- CI/CD services for building and storing container images to use on Nautilus

Creating an NRP GitLab Account



Create an account on Nautilus GitLab

Note: You may need to have your university system added by NRP Admins to use NRP GitLab



Creating a Custom Container Image

Create new repository

- Code & other files can be private, but the repository and the container registry MUST be public
- Open the repository in the Web IDE or clone it locally

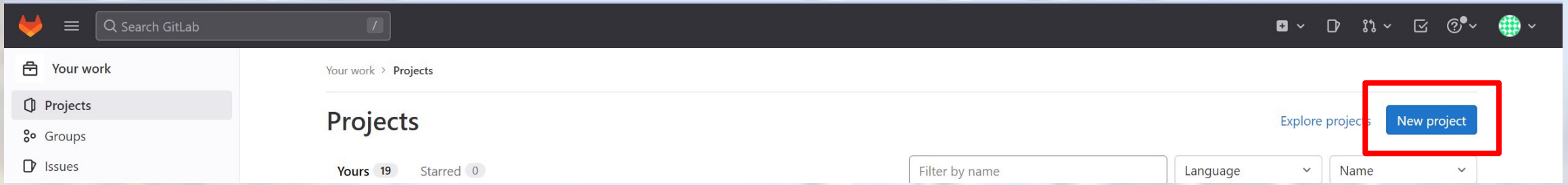
Create Gitlab CI YAML File

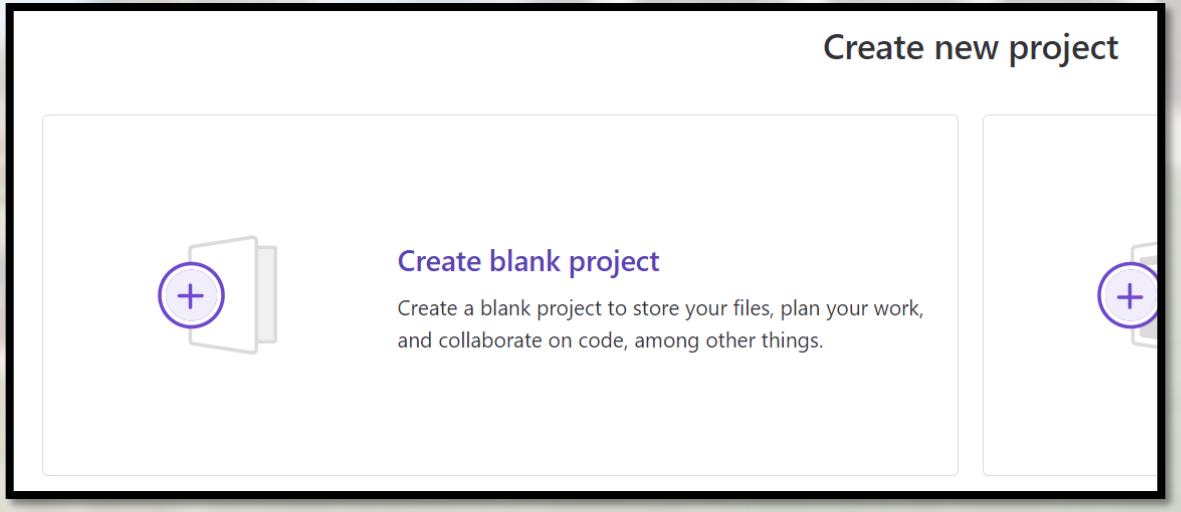
- Create a file called .gitlab-ci.yml
- Copy its content from this link: .gitlab-ci.yml
- Commit and push to your repository

Create Dockerfile

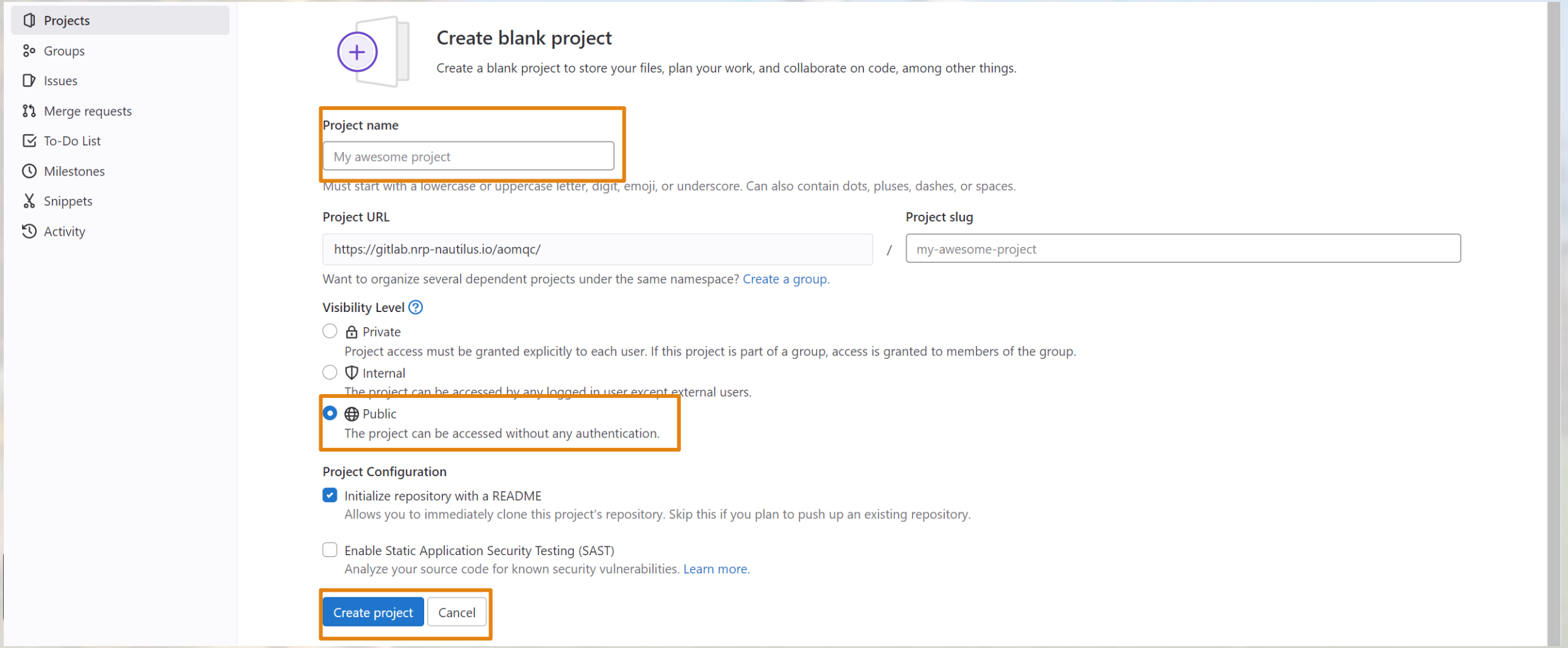
- Write the appropriate commands to build the image you need
- Use a community container as your base
- Commit and push to your repository

NRP GitLab
Creating a New Repository





NRP GitLab
Creating a New Repository



- We need to create:
  - Gitlab CI YAML
  - Dockerfile

C **CGISegment**
Project ID: 2316 Leave project

Star 0 Fork 0

568 Commits 1 Branch 0 Tags 1.1 MB Project Storage

Fix Docker image
Alex Hurt authored 6 months ago
e98e742e

master cgisegment / + Find file Web IDE Clone

README CI/CD configuration Wiki

Name	Last commit	Last update
cgisegment	Wrong indentation for final test set aggregation	9 months ago
sample_configs	Updating the sample configs	1 year ago
tests	Getting Tests Up-To-Date	1 year ago
.gitignore	*ipynb	1 year ago
.gitlab-ci.yml	Add gitlab CI YML	7 months ago
Dockerfile	Fix Docker image	6 months ago
README.md	Remove ref	7 months ago
main.py	SegFormer and Lawin	10 months ago

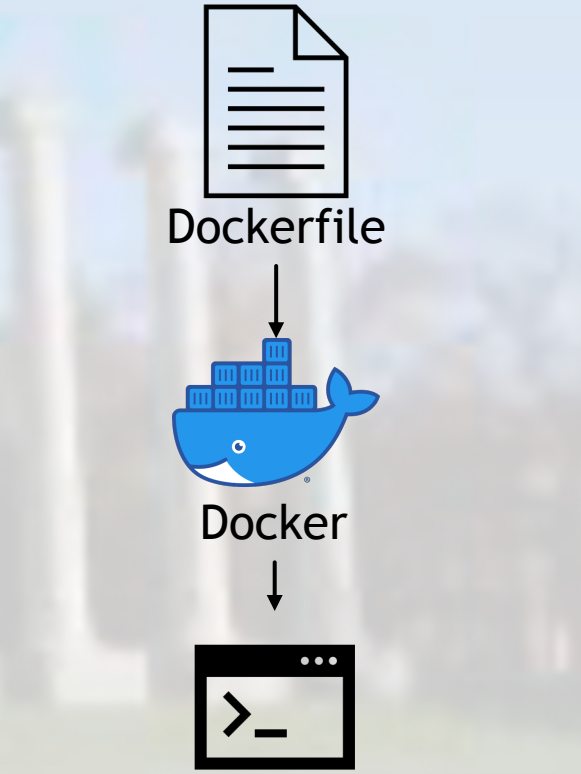
Creating the GitLab CI YAML

https://ucsd-prp.gitlab.io/userdocs/development/gitlab/

```yaml
image: gcr.io/kaniko-project/executor:debug

stages:
- build-and-push

build-and-push-job:
  stage: build-and-push
  variables:
    GODEBUG: "http2client=0"
  script:
  - echo "{\"auths\":{\"$CI_REGISTRY\":{\"username\":\"$CI_REGISTRY_USER\",\"passwo
  - /kaniko/executor --cache=true --push-retry=10 --context $CI_PROJECT_DIR --docke
```

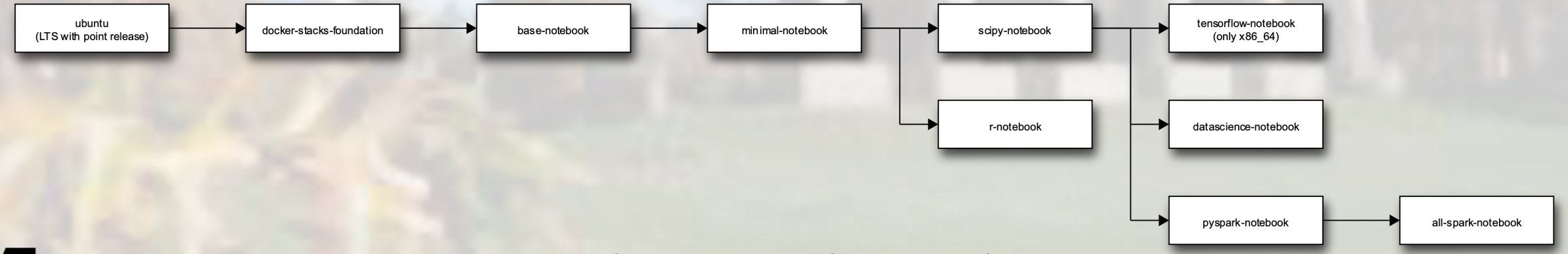
Dockerfiles

- Recall: Containers and container images enable reliable portability of software
- We need a way to build container images in a distributed and standardized way, so that anyone with a container runtime can build and run our software package (container image)
- Dockerfiles are the template, or recipe, for how a container image will be built
- Dockerfiles use a custom format and set of commands to describe how a container image will be built



Using Community Published Container Images

- Container images are extensible!
  - Containers can be built from other images
- We can then build hierarchical docker containers, where each container has a specific set of dependencies and packages installed
- Base images enable rapid and reproducible building of even complex docker containers by leveraging the open source, published docker images



Example: Jupyter Docker Images[1]

1. https://jupyter-docker-stacks.readthedocs.io/en/latest/using/selecting.html

Dockerfile: Common Commands

- Dockerfiles function via a set of directives and their respective arguments:

```
DIRECTIVE arg1 arg2 …
```

- FROM
  - The FROM command defines the starting point for the Docker build process. If you are building custom software from the ground up, you may be setting this to a linux operating system, such as ubuntu.
  - If you are working in a framework or programming language, this may also be a certain version of that framework, such as python:3.8, node:8, pytorch:1.8, etc.
  - The docker image specified in the from command must be present either on the public docker hub repository, or be visible to the build context via a full URL.
- ENV & ARG
  - The ENV and ARG commands define environment variables during the build process. There are 2 key differences between them:
  - ENV commands persist to the final container, so ENV key value will persist to the launched container, while ARG key value will not
  - ARG commands can be overridden at build time, which allows for templating of Dockerfiles
  - The ENV command is very useful for tasks like ensuring your executable is present on the PATH of the container, while ARG is useful for things like ensuring you are building the correct version of the software.

Dockerfile: Common Commands

- CMD and ENTRYPOINT
  - The CMD and ENTRYPOINT commands both set the command to be run when the container starts via a docker run command.
  - There is 1 key difference between them: a CMD can be overridden via either a downstream Dockerfile (i.e., someone uses your container in their FROM command) or via the command line during container startup. An ENTRYPOINT cannot be as easily overridden, and requires a special flag to be overridden.
  - Best practice is to set a CMD unless you have reasoning behind using ENTRYPOINT
- COPY
  - The COPY command is how local files are copied into the container. Recall that Docker containers are mini virtual machines, meaning that it has its own filesystem. In order to build and run software in docker, we often need to copy our code and scripts into the container. To do this, we use the COPY command
- RUN
  - The RUN command tells the docker build process to run a command inside the container. This could be everything from installing a package with apt (RUN apt install) to creating and removing files in the container.
  - The commands available for the RUN command are determined by the base container, i.e. apt will only be available in debian based containers.

Sample Dockerfile

```
ARG PYVERSION=3.8
FROM python:${PYVERSION}

RUN mkdir -p /workspace
WORKDIR /workspace

COPY /requirements.txt /workspace
RUN pip install -r ./requirements.txt

COPY /*.py /workspace/
CMD /bin/bash
```

- Create a build argument for the python version that defaults to 3.8
- Start FROM an existing image in a Docker *registry*. In this case, python
- Create a directory named /workspace and set it as the working directory
- Copy a file named “requirements.txt” from the build directory to /workspace in the container
- Use pip to install the requirements defined in requirements.txt
- Copy all python files in the build directory to the /workspace directory in the container
- Set the default command to run when the container starts to /bin/bash

Writing Dockerfiles: Multiple Paths to the Same Goal

Two different styles of writing Docker file

main / deeplearning_pytorch / Dockerfile — Find file | Blame | History | Permalink

Dockerfile 1.24 KiB — Open in Web IDE | Replace | Delete

```
FROM nvidia/cuda:11.2.2-cudnn8-runtime-ubuntu20.04

RUN chmod 1777 /tmp && chmod 1777 /var/tmp
RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y vim curl ca-certificates amqp-tools wget graphviz
RUN apt-get -y install git

RUN apt-get install -y python3
RUN apt-get install -y python3-pip

RUN pip install --upgrade protobuf==3.20.1

RUN pip install numpy scipy
RUN pip install matplotlib
RUN pip install pyyaml
RUN pip install pytest-shutil
RUN pip install rasterio
RUN pip install pandas
RUN pip install geopandas
RUN pip install geojson
RUN pip install rtree
RUN pip install pillow
RUN pip install jinja2
RUN pip install bs4
RUN pip install lxml
RUN pip install pydot
RUN pip install pydotplus

RUN apt-get install -y binutils libproj-dev gdal-bin
```

master / cgisegment / Dockerfile — Find file | Blame | History | Permalink

Fix Docker image
Alex Hurt authored 6 months ago — e98e742e

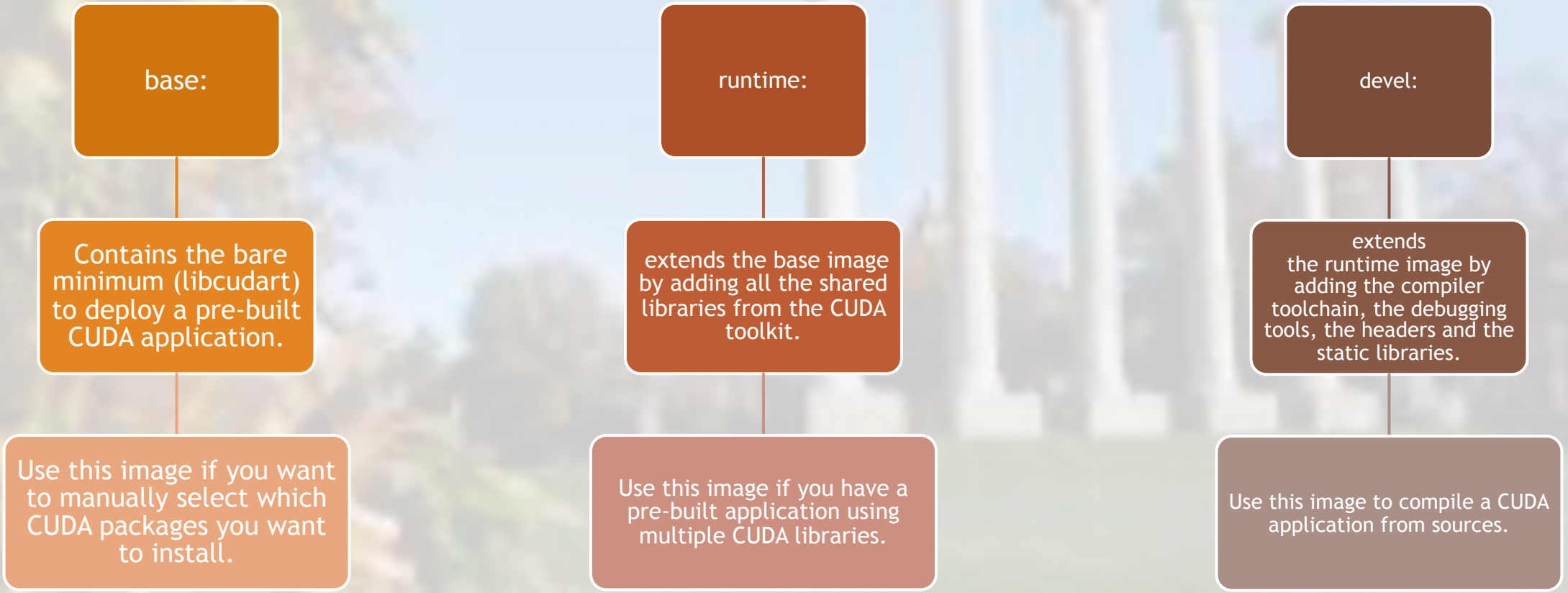
Dockerfile 1.29 KiB — Open in Web IDE | Replace | Delete

```
FROM pytorch/pytorch:1.12.1-cuda11.3-cudnn8-devel

##########
# env
##########
RUN mkdir -p /workspace
ENV CGISEGMENT /workspace/cgisegment/
ENV DEBIAN_FRONTEND noninteractive
ENV DISTRO ubuntu2004
ENV ARCH x86_64
ENV TORCH_HOME /pytorch_data


#######
# Install dependencies
#######
RUN apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/cuda/repos/$DISTRO/$ARCH/3bf863cc.pub
RUN apt update -y && apt install -y \
        git \
        build-essential \
        libsm6 \
        libxext6 \
        libxrender-dev \
        libgl1-mesa-glx \
        libglib2.0-0

#########
# Copy files
#########
RUN mkdir -p ${CGISEGMENT}
COPY /main.py ${CGISEGMENT}
COPY /cgisegment ${CGISEGMENT}/cgisegment
COPY /requirements.txt /tmp/requirements.txt
```

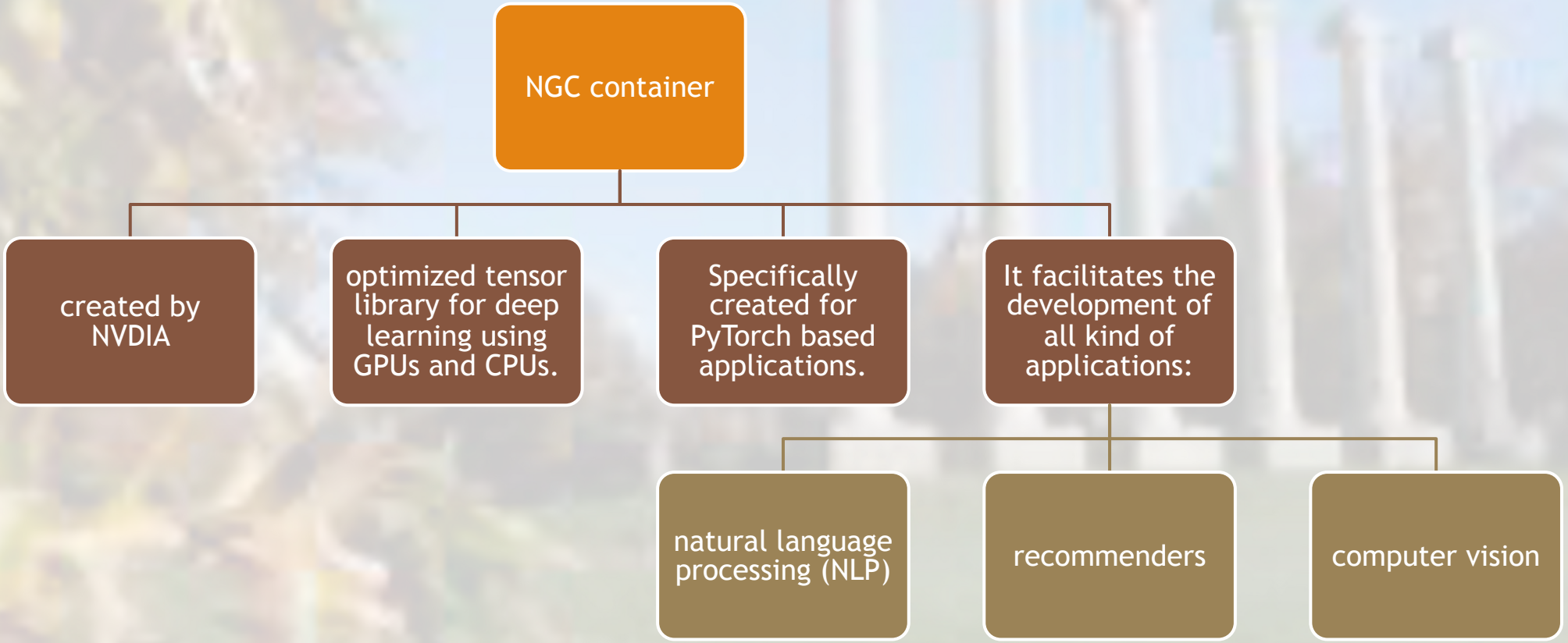
Nautilus Gitlab: Building GPU-enabled Containers

- To utilize the GPUs in a cluster, GPU-enabled image is a MUST
- GPU containers can be built from community developed and published base containers, such as nvidia/cuda and the NVIDIA NGC
- CUDA GPU base images comes in three variations:
  - base
  - runtime
  - devel

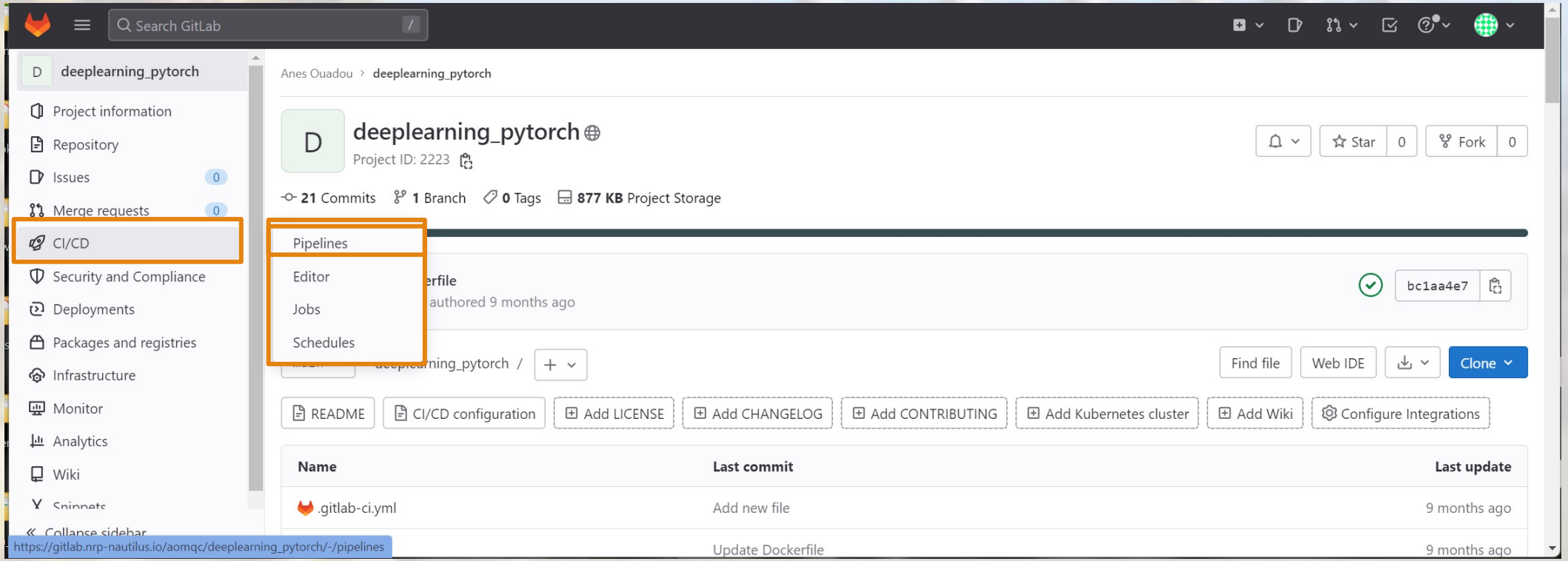
NVIDIA CUDA Containers: Three Different Types



NVIDIA NGC Containers: Optimized But Inflexible

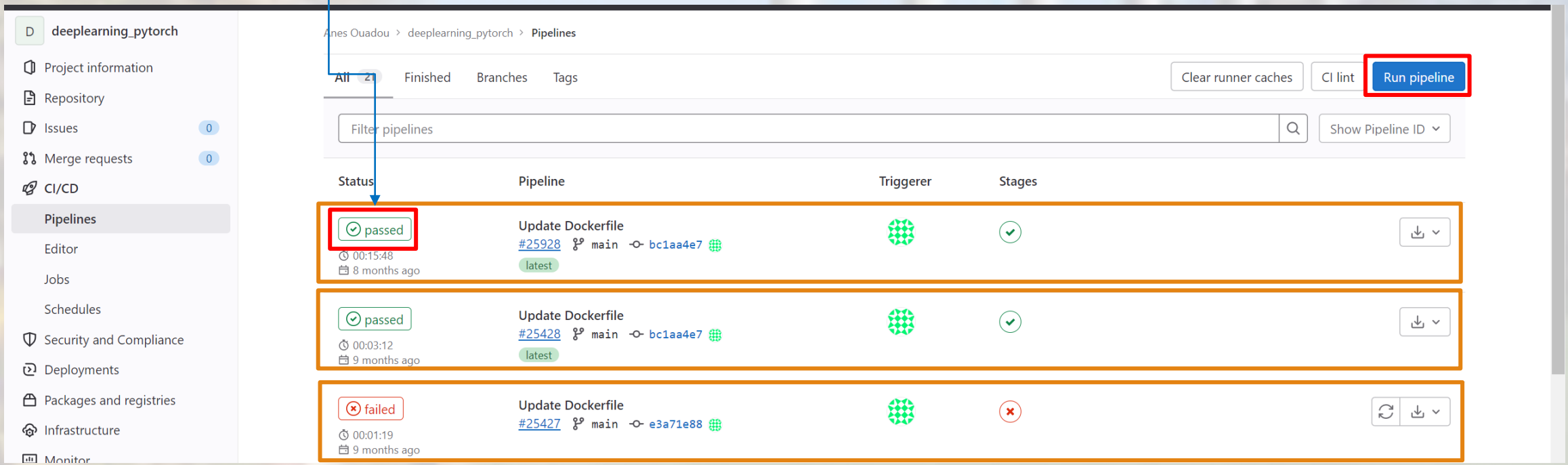


Nautilus GitLab: Viewing CI/CD Pipelines

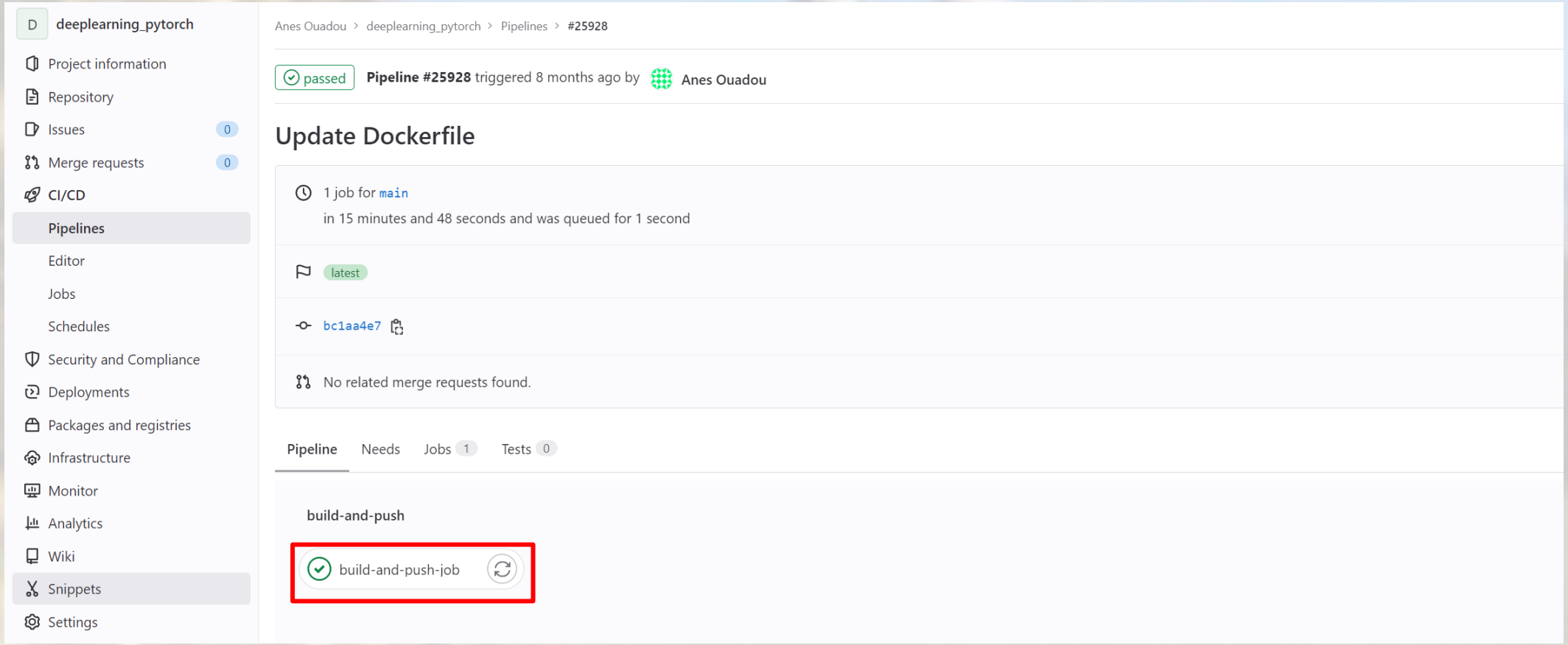


Nautilus Gitlab: Viewing Pipeline Status

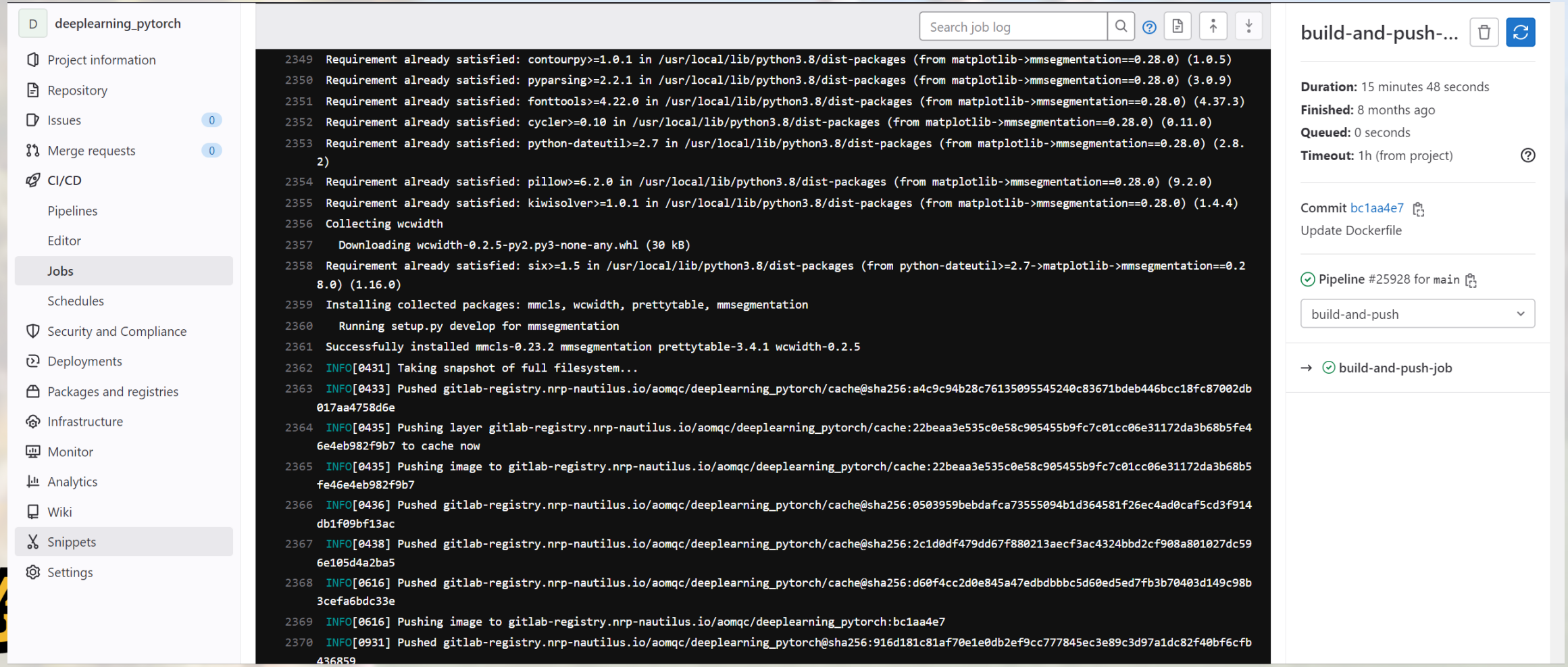
The image is being built the status is: Running



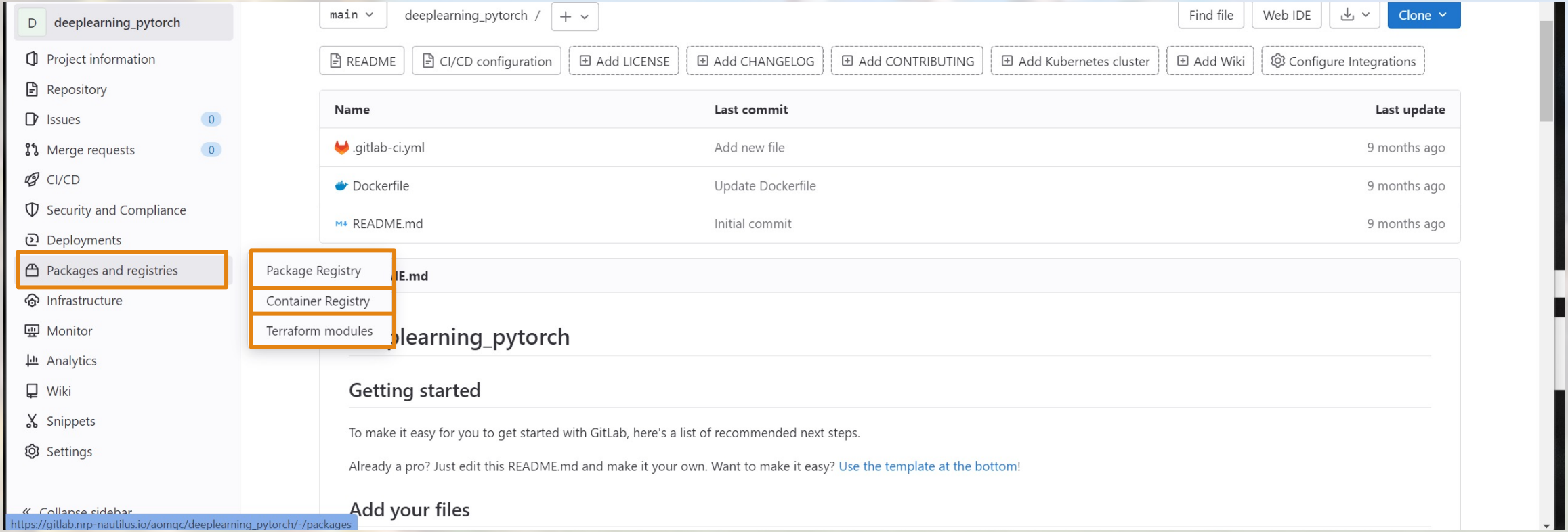
Nautilus GitLab: Viewing Job Status



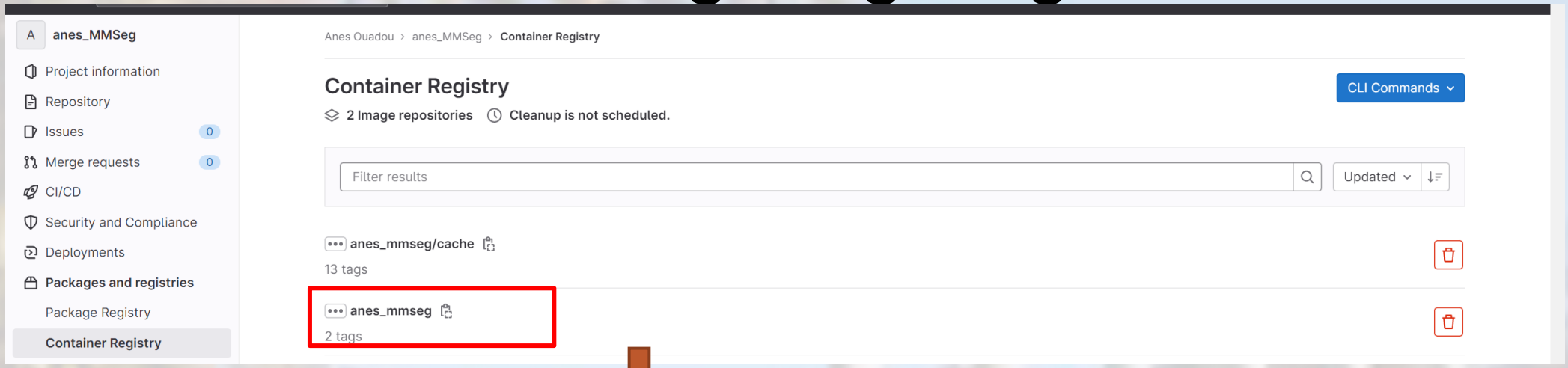
Nautilus GitLab: Reviewing Job Output



Nautilus GitLab: Viewing the Container Registry

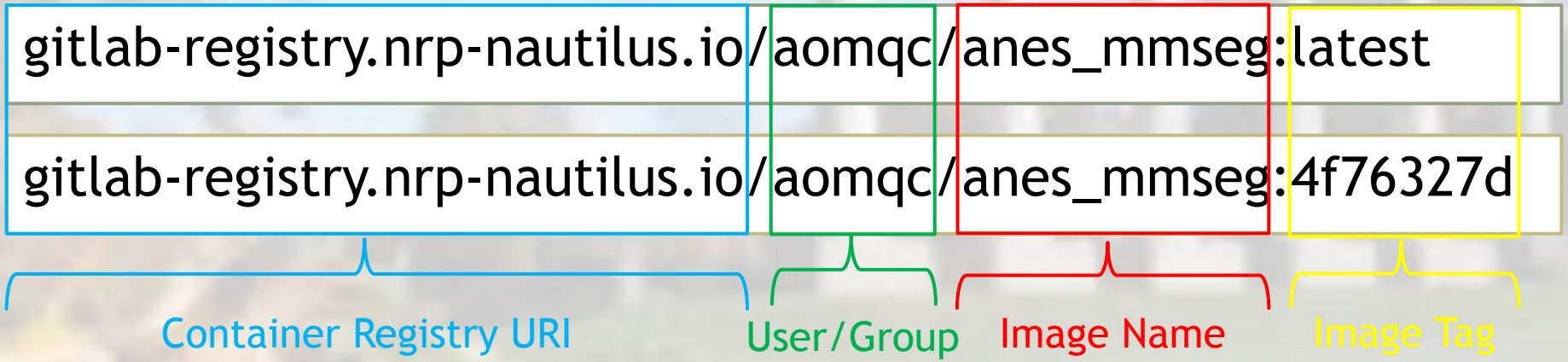


Nautilus GitLab: Viewing Image Tags



Nautilus GitLab: Understanding Container Image URIs

- The list of images will always have one image labeled latest and at least one image with a randomly generated tag



Note: Never use the latest tag in your YAML files, as it causes collision issues on nodes in the cluster

Nautilus GitLab: Using Your Container Image on Nautilus

```
apiVersion: batch/v1
kind: Job

metadata:
  name: convert-xview-train

spec:
  backoffLimit: 0
  template:
    spec:
      restartPolicy: Never
      containers:
        - name: dsc
          image: gitlab-registry.nrp-nautilus.io/jhurt/dsc:72de1d25
          imagePullPolicy: IfNotPresent
          command: ["python3",  "/output/scripts/myscript.py"]
          resources:
            limits:
              memory: 16Gi
              cpu: 16
            requests:
              memory: 16Gi
              cpu: 16
          volumeMounts:
            - mountPath: /data
              name: jhurt-data
      volumes:
        - name: jhurt-data
          persistentVolumeClaim:
            claimName: jhurt-data
```

Hands-On Example: Image Processing (CPU)

- Register for a Gitlab Account on NRP
- Login to your Account
- Create a new public repository
- Add your Python code
- Add your Dockerfile
- Add your Gitlab CI YAML
- Commit & Push to trigger the CI build and push of your container image
- Use your container image in a pod on Nautilus

https://gitlab.nrp-nautilus.io/jhurt/python-ci

Hands-On Example: Matrix Multiplication (GPU)

- Create a new public repository
- Add your **GPU-enabled** Python code
- Add your **CUDA-based** Dockerfile
- Add your Gitlab CI YAML
- Commit & Push to trigger the CI build and push of your container image
- Use your container image in a pod on Nautilus

https://gitlab.nrp-nautilus.io/jhurt/gpu-ci