Lab 5 – Containers and Kubernetes

# Basics:

* Introduction to containers, e.g. Docker: <https://docs.docker.com/get-started/>
  + Lightweight virtualization
  + Key concepts:
    - Dockerfile, Image, Container, Image repositories
* Kubernetes, container orchestration: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>
  + Cluster, Pods, Services etc.
  + Used to manage a cluster of hosts/containers
  + Why do we need orchestration? What functionalities are provided by k8s, which exceed those of a simple dockerd daemon?
* Alternative implementations:
  + Containers: apptainer, podman, lxc
  + Container orchestration: OpenShift, Rancher, Nomad, commercial services: e.g. AWS ECS, Azure AKS, Google GKE

# Prerequisites

* Preferably a Linux environment (Linux VM, or Windows/WSL 2)
* Docker
* Kubernetes cluster. While installing a full-fledged cluster is quite complicated, it’s possible to prepare a development “mini k8s cluster” with the help of
  + Minikube: [https://minikube.sigs.k8s.io/docs/start](https://minikube.sigs.k8s.io/docs/start/) (simple, only 1-node cluster)
  + Kind: [https://kind.sigs.k8s.io](https://kind.sigs.k8s.io/) (Kubernetes in Docker, possible to emulate multiple nodes on a single machine)
  + Or others…

# Assignments

Assignments were performed in **WSL** in the **Ubuntu** operating system

1. Dockerize AWS-CLI (3p)
2. Create a Dockerfile with [aws-cli](https://aws.amazon.com/cli/), built from [source files](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-source-install.html) (using tar.gz file / make).
   * Use a [multi-stage build](https://docs.docker.com/build/building/multi-stage/) to create a small image. Also use a [small base image, e.g., based on Alpine Linux](https://hub.docker.com/_/python/).
   * Read more on [Python and Docker multistage build.](https://www.blogfoobar.com/post/2018/02/10/python-and-docker-multistage-build)
   * Follow [best practices for writing Dockerfiles](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/). In particular read about [ADD or COPY](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#add-or-copy).

nano Dockerfile

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

My main example for creating the image was the AWS instructions. To improve the Dockerfile, I concretized the Alpine version and removed the groff installation

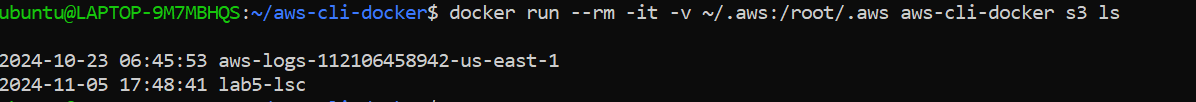
Running image locally

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

1. Build an image based on your Dockerfile and test it.
   * Use [a volume](https://docs.docker.com/storage/volumes/) (-v option) to make AWS credentials available in the container.

Checking buckets on AWS with container

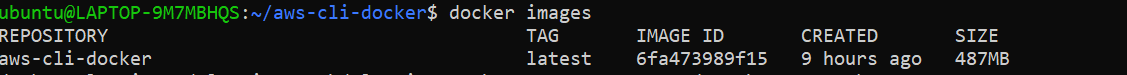


Checking buckets on AWS

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Size of the container image



1. Kubernetes deployment (5p)
   1. Create a k8s cluster using Amazon Elastic Kubernetes Service (EKS)
      * You can also use minikube, kind, or any other Kubernetes distribution, or existing cluster.
      * Minikube, by default, uses its own internal Docker daemon. This daemon doesn’t know anything about images built previously. Prepare your environment by directing it to access the internal docker daemon by using the $(minikube docker-env) command and rebuild your images. This way images will be available within the k8s cluster. (<https://medium.com/bb-tutorials-and-thoughts/how-to-use-own-local-doker-images-with-minikube-2c1ed0b0968>)

My main example was the AWS instructions for creating cluster with AWS console

Cluster

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

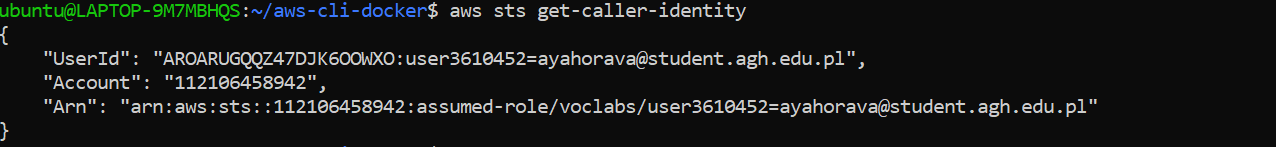
Opis wygenerowany automatycznie

Nodes group

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Cluster authentication



Create a kubeconfig file for the cluster



Testing configuration

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

* 1. Using Helm, install an [NFS server and provisioner](https://github.com/kubernetes-sigs/nfs-ganesha-server-and-external-provisioner) in the cluster.
     + Go to charts/nfs-server-provisioner for a README.
     + Pay attention to configuration parameters, in particular, override storageClass.name which denotes the name of the StorageClass that you’ll have to use when creating Persistent Volume Claims.

Installing Helm

Obraz zawierający tekst, Czcionka, oprogramowanie, zrzut ekranu

Opis wygenerowany automatycznie

Installing nfs

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Changing StorageClass

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

* 1. Create a [Persistent Volume Claim](https://kubernetes.io/docs/concepts/storage/persistent-volumes/) which will bind to a NFS Persistent Volume [provisioned dynamically](https://kubernetes.io/docs/concepts/storage/persistent-volumes/#provisioning) by the provisioner installed in the previous step.

nano nfs-pvc.yaml

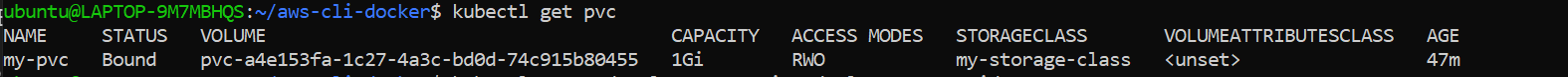
Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Running nfs-pvc.yaml



Checking pvc



* 1. Create a [Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/) with a HTTP server (e.g., apache or nginx). The web content directory should be mounted as a volume using the PVC created in the previous step.

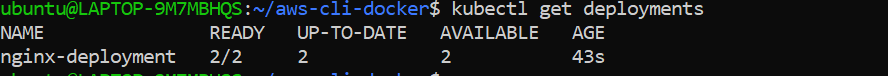
nano nginx-deployment.yaml

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

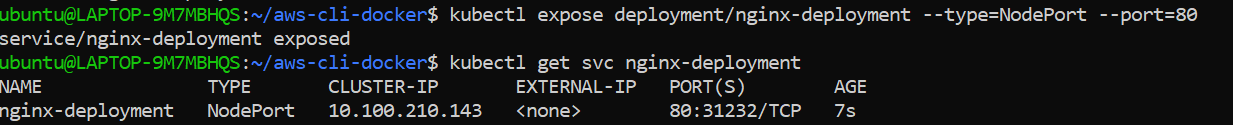
Opis wygenerowany automatycznie

Running deployment

kubectl apply -f nginx-deployment.yaml

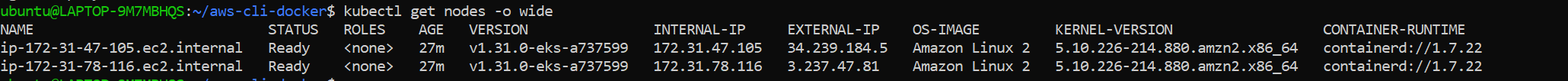


* 1. Create a [Service](https://kubernetes.io/docs/concepts/services-networking/) associated with the Pod(s) of the HTTP server Deployment.



Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie



* 1. Create a [Job](https://kubernetes.io/docs/concepts/workloads/controllers/job/) which mounts the PVC and copies a sample content through the shared NFS PV.

Job configuration

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Nfs-test.yaml

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

* 1. Test the HTTP server by showing the sample web content in a browser.

Test in console

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Test HTTP

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Repository: <https://github.com/honeyAsya/LSC>

Commands:

nano Dockerfile

docker build -t aws-cli-docker .

docker run --rm -it aws-cli-docker –version

aws configure

aws s3api create-bucket --bucket lab5-lsc --region us-east-1

aws s3 ls

aws sts get-caller-identity

curl -fsSL -o get\_helm.sh <https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3>

curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash % Total % Received % Xferd

helm repo add nfs-ganesha-server-and-external-provisioner <https://kubernetes-sigs.github.io/nfs-ganesha-server-and-external-provisioner/>

helm install my-release nfs-ganesha-server-and-external-provisioner/nfs-server-provisioner

helm upgrade my-release nfs-ganesha-server-and-external-provisioner/nfs-server-provisioner --set

storageClass.name=my-storage-class

nano nfs-pvc.yaml

kubectl apply -f nfs-pvc.yaml

kubectl get pvc my-pvc

nano nginx-deployment.yaml

kubectl apply -f nginx-deployment.yaml

nano nginx-service.yaml

kubectl apply -f nginx-service.yaml

kubectl get nodes curl <http://3.237.47.81:31232>

nano job.yaml

kubectl apply -f job.yaml

nano nfs-test.yaml

kubectl apply -f nfs-test.yaml

kubectl exec -it nfs-test -- /bin/sh

cat /mnt/sample.txt