Cheat sheets and checklists

- Student name: Xander Van der Linden
- GitHub repo: https://github.com/HOGENT-MLOps/mlops-2425-xvanderlinden

Basic commands

Task	Command
Change directory	`cd DIRECTORY`
List files	`ls -l`
Create directory	`mkdir DIRECTORY`
Create empty file	`touch FILE`
Copy file	`cp FILE DEST`
Move file	`mv FILE DEST`

Docker commands

Task	Command
List all containers	`docker ps -a`
List all images	`docker images`
Stop a container	`docker stop CONTAINER`
Remove a container	`docker rm CONTAINER`

Git workflow

Simple workflow for a personal project without other contributors:

Task	Command
Current project status	`git status`
Select files to be committed	`git add FILE...`
Commit changes to local repository	`git commit -m 'MESSAGE'`
Push local changes to remote repository	`git push`
Pull changes from remote repository to local	`git pull`

Checklist network configuration

1. Is the IP address correct? `ip a`
2. Is the router/default gateway correct? `ip r -n`

3. Is a DNS-server available? `cat /etc/resolv.conf`

Labo 1 Docker

Task	Command
Change directory	`cd .\resources\02-dockerlab\`
Start Portainer Container	`docker compose -f .\docker-compose.portainer.yml up -d`
Build Docker Image	`docker build -t webapp .`
Run Docker Image	`docker run -d -p 3000:3000 --name webapp_container webapp`
Fetch Animal Data (GET)	`curl http://localhost:3000/animals`
Add Animal Data (POST)	`curl -X POST -H "Content-Type: application/json" -d '{"name":"Lion","age":4}' http://localhost:3000/animals`
Remove Docker Container	`docker rm webapp_container`
Start Webapp and Database	`docker compose up -d`
Stop Docker Container	`docker stop webapp_container`
Update .gitignore	`echo "dockerlab/webapp/database" >> .gitignore`
Copy Docker Compose File	`cp docker-compose.yml docker-compose-sqlite.yml`
Run Tests	`docker compose run test`

Labo 2 CI/CD

Task	Command
Set global username	`git config --global user.name "Your Name"`
Set global email	`git config --global user.email "Your@Name.com"`
Check Git configuration	`git config --global --list`
Initialize a Git repository	`git init`
Add all files to staging	`git add .`
Commit with a message	`git commit -m "Your commit message"`

Task	Command
Push to the main branch	`git push origin main`
Clone a repository	`git clone <repository-url>`
Set remote origin	`git remote add origin <repository-url>`
Install project dependencies	`yarn install`
Run linter	`yarn lint`
Run tests	`yarn test`
Create a Docker image	`docker build -t <image-name> .`
Run a Docker container	`docker run -p 3000:3000 <image-name>`
Push Docker image to Docker Hub	`docker push <dockerhub-username>/<image-name>`
Pull Docker image from Docker Hub	`docker pull <dockerhub-username>/<image-name>`

Labo 3 - MLFlow Cheatsheet

Taak	Commando
Gebruik Python's ingebouwde `venv` module om een virtuele omgeving aan te maken	`python3 -m venv venv`
Activeer de virtuele omgeving (Windows)	`.\venv\Scripts\Activate`
Controleer het pad naar Python	`Get-Command python`
Controleer het pad naar pip	`Get-Command pip`
Start de Prefect server (Linux/macOS)	`export PREFECT_HOME=$(pwd)/prefect_home`
Start de Prefect server (Windows)	`$Env:PREFECT_HOME = "$(Get-Location)/prefect_home"`
Start de Prefect server	`prefect server start`
Start de MLFlow server	`mlflow server`
Zet MLFlow system metrics logging op true	`export MLFLOW_ENABLE_SYSTEM_METRICS_LOGGING=true`
Start MLFlow met logging	`python -m mlflow.server`
Log een MLFlow experiment	`mlflow.start_run()`
Log metrics met MLFlow	`mlflow.log_metric("metric_name", value)`
Log een model met MLFlow	`mlflow.log_artifact("pad_naar_model")`
Zet de MLFlow tracking URI	`mlflow.set_tracking_uri("http://localhost:5000")`

Labo 4 - Kubernetes

Taak	Commando
Real-time service opvragen(zo goed als)	`while (1) { kubectl get all; sleep 1 }`
Toon alle resources	`kubectl get all`
Toon de deployments	`kubectl get deployments`
Toon de nodes in de cluster	`kubectl get nodes`
Toon de pods in de cluster	`kubectl get pods`
Toon de kubectl configuratie	`kubectl config view`
Maak een "hello-node" deployment	`kubectl create deployment hello-node --image=agnhost:2.39 -- /agnhost netexec --http-port=8080`
Exposeer als LoadBalancer service	`kubectl expose deployment hello-node --type=LoadBalancer --port=8080`
Toon de services	`kubectl get services`
Open de service via Minikube	`minikube service hello-node`
Start Minikube	`minikube start`
Open Minikube dashboard	`minikube dashboard`
Toon logs van een pod	`kubectl logs hello-node-66d457cb86-k45gr`
Apply deployment YAML	`kubectl apply -f echo-deployment.yml`
Apply service YAML	`kubectl apply -f echo-service.yml`
Open de echo-service via Minikube	`minikube service echo-service`
Apply alle resources via YAML	`kubectl apply -f echo-all.yml`
Open alle services via Minikube	`minikube service echo-all-service`
Toon gedetailleerde podinfo inclusief node	`kubectl get pods -o wide`
Maak herhaalde cURL-aanroepen naar een service	`for ($i = 1; $i -le 10; $i++) { curl http://127.0.0.1:59623/ }`
Voeg een label toe aan een pod	`kubectl label pod <pod_name> application_type=demo`
Verander label met `--overwrite` optie	`kubectl label pod <pod_name> application_type=production --overwrite`
Toon pods met labels	`kubectl get pods --show-labels`
Toon jobs met labels	`kubectl get jobs --show-labels`

Taak	Commando
Toon pods van een specifieke job	`kubectl get pods -l job-name=pi`
Toon beschrijving van cronjob	`kubectl describe cronjob hello`
Toon alle cronjobs	`kubectl get cronjobs`
Schaal deployment met 5 replicas	`kubectl scale deployment frontend --replicas=5`
Toon de pods na schaling	`kubectl get pods`