Data Science Workflow

Lecture 1: Introduction, K8s, Docker, Git

M.Sc. Teodor Chiaburu

FB 6, Data Science, Berliner Hochschule für Technik

5. Juni 2025

Introduction

Hi, I'm Teodor Chiaburu. Call me Teo ;-)

- Bachelors in Maths and Masters in Data Science (both at BHT)
- Currently in my final PhD year (XAI, Uncertainty Quantization, Computer Vision...)
- Taught ML and Numerical Analysis
- Passionate about dancing Salsa and cooking

Introduction

Hi, I'm Teodor Chiaburu. Call me Teo ;-)

- Bachelors in Maths and Masters in Data Science (both at BHT)
- Currently in my final PhD year (XAI, Uncertainty Quantization, Computer Vision...)
- Taught ML and Numerical Analysis
- Passionate about dancing Salsa and cooking

Let's connect

- My website
- LinkedIn
- Mail: chiaburu.teodor@bht-berlin.de

Introduction

Hi, I'm Teodor Chiaburu. Call me Teo ;-)

- Bachelors in Maths and Masters in Data Science (both at BHT)
- Currently in my final PhD year (XAI, Uncertainty Quantization, Computer Vision...)
- Taught ML and Numerical Analysis
- Passionate about dancing Salsa and cooking

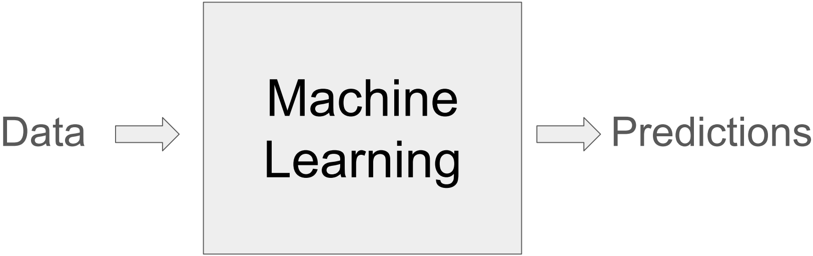
Let's connect

- My website
- LinkedIn
- Mail: chiaburu.teodor@bht-berlin.de

What is this course about?

Check *Course_Info.pdf* on Moodle!

ML in Academia

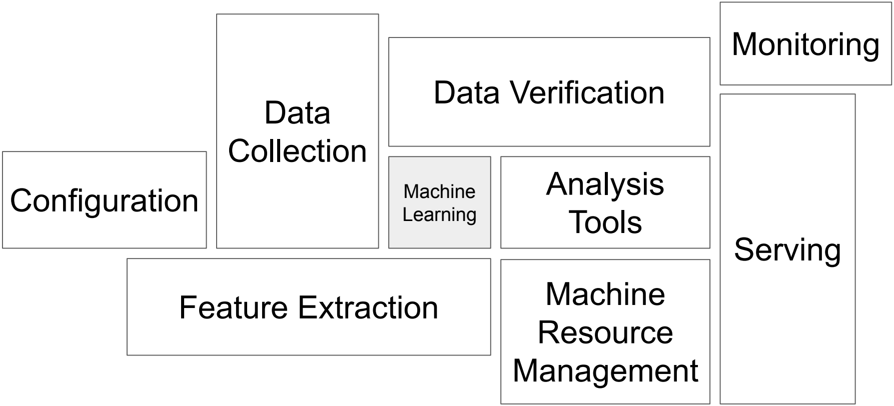


Data

Machine Learning

Predictions

ML in Production



Configuration

Data Collection

Data Verification

Machine Learning

Analysis Tools

Feature Extraction

Machine Resource Management

Monitoring

Serving

Adapted from *Sculley et al, Hidden Technical Debt in ML Systems, NeurIPS 2015*

Deploying with Kubernetes 101

Goals

By the end of this day, you'll be able to:

Deploying with Kubernetes 101

Goals

By the end of this day, you'll be able to:

- Understand what Kubernetes (K8s) is and what problems it solves.

Deploying with Kubernetes 101

Goals

By the end of this day, you'll be able to:

- Understand what Kubernetes (K8s) is and what problems it solves.
- Understand core K8s concepts e.g. Pods, Deployments, Jobs...

Deploying with Kubernetes 101

Goals

By the end of this day, you'll be able to:

- Understand what Kubernetes (K8s) is and what problems it solves.
- Understand core K8s concepts e.g. Pods, Deployments, Jobs...
- Connect to our university's cluster.

Deploying with Kubernetes 101

Goals

By the end of this day, you'll be able to:

- Understand what Kubernetes (K8s) is and what problems it solves.
- Understand core K8s concepts e.g. Pods, Deployments, Jobs...
- Connect to our university's cluster.
- SSH into your running pods.

Deploying with Kubernetes 101

Goals

By the end of this day, you'll be able to:

- Understand what Kubernetes (K8s) is and what problems it solves.
- Understand core K8s concepts e.g. Pods, Deployments, Jobs...
- Connect to our university's cluster.
- SSH into your running pods.
- Deploy pods and volumes using `kubectl`.

Deploying with Kubernetes 101

Goals

By the end of this day, you'll be able to:

- Understand what Kubernetes (K8s) is and what problems it solves.
- Understand core K8s concepts e.g. Pods, Deployments, Jobs...
- Connect to our university's cluster.
- SSH into your running pods.
- Deploy pods and volumes using `kubectl`.
- Use port forwarding to connect to your pods and work remotely.

What is Kubernetes, really?

Kubernetes (K8s) is an open-source system for managing containerized apps. Think of it like an OS for clusters.

It deploys, scales and keeps your apps alive. Basically, your app's 24/7 babysitter.

Official documentation: `https://kubernetes.io/docs`

Abbildung: K8s Logo

What Is Kubernetes, Really?

- *Kubernetes* (Greek) = helmsman, pilot, person steering a ship[1]
- Abbreviation *K8s* - 8 letters between 'K' and 's' ($8 = 2^3$ is also a perfect cube)
- Exactly on this day - 6th June - 11 years ago, the first commit of Kubernetes was pushed to GitHub[2]
  - Since then, it already grew to the second largest open source software community in the world[3] (which one is the largest? :-))

[1] https://kubernetes.io/docs/concepts/overview/

[2] https://kubernetes.io/blog/2024/06/06/10-years-of-kubernetes/

[3] https://www.cncf.io/reports/kubernetes-project-journey-report/

So what is THE (BHT) CLUSTER?

- Let's have a peek inside: Grafana

So what is THE (BHT) CLUSTER?

- Let's have a peek inside: Grafana
- This is what it sounds like: youtube video

So what is THE (BHT) CLUSTER?

- Let’s have a peek inside: Grafana
- This is what it sounds like: youtube video
- The Cluster Docs (our official BHT cluster documentation, most of your questions already have an answer there)

So what is THE (BHT) CLUSTER?

- Let's have a peek inside: Grafana
- This is what it sounds like: youtube video
- The Cluster Docs (our official BHT cluster documentation, most of your questions already have an answer there)
- Dashboard for your own workspace: Headlamp

So what is THE (BHT) CLUSTER?

- Let's have a peek inside: Grafana
- This is what it sounds like: youtube video
- The Cluster Docs (our official BHT cluster documentation, most of your questions already have an answer there)
- Dashboard for your own workspace: Headlamp

Notes

1) Our cluster is made up of real physical machines, but you can also run K8s on virtual machines/nodes in the cloud e.g. on AWS (like Netflix does).

So what is THE (BHT) CLUSTER?

- Let's have a peek inside: Grafana
- This is what it sounds like: youtube video
- The Cluster Docs (our official BHT cluster documentation, most of your questions already have an answer there)
- Dashboard for your own workspace: Headlamp

Notes

1) Our cluster is made up of real physical machines, but you can also run K8s on virtual machines/nodes in the cloud e.g. on AWS (like Netflix does).
2) You can experiment on your computer with **minikube**, which creates a local single-node K8s cluster: `https://minikube.sigs.k8s.io/docs/`. Or invest in a Raspberry Pi and build a cute little cluster there (plenty of tutorials online).

Kubernetes core concepts

- **Pod**: Smallest deployable unit (usually 1 container).

Kubernetes core concepts

- **Pod**: Smallest deployable unit (usually 1 container).
- **Job**: One-off tasks that run to completion and then stop.

Kubernetes core concepts

- **Pod**: Smallest deployable unit (usually 1 container).
- **Job**: One-off tasks that run to completion and then stop.
- **Deployment**: Keeps desired number of pods alive.

Kubernetes core concepts

- **Pod**: Smallest deployable unit (usually 1 container).
- **Job**: One-off tasks that run to completion and then stop.
- **Deployment**: Keeps desired number of pods alive.
- **PVC**: **P**ersistent **V**olume **C**laim, requests persistent storage.

Kubernetes core concepts

- **Pod**: Smallest deployable unit (usually 1 container).
- **Job**: One-off tasks that run to completion and then stop.
- **Deployment**: Keeps desired number of pods alive.
- **PVC**: **P**ersistent **V**olume **C**laim, requests persistent storage.
- **Secret**: Stores sensitive info (like SSH keys).

Kubernetes core concepts

- **Pod**: Smallest deployable unit (usually 1 container).
- **Job**: One-off tasks that run to completion and then stop.
- **Deployment**: Keeps desired number of pods alive.
- **PVC**: **P**ersistent **V**olume **C**laim, requests persistent storage.
- **Secret**: Stores sensitive info (like SSH keys).
- Others: Namespaces, Cronjobs, Services ...

Prerequisites for working on the cluster[a]

You will need:

☑ BHT account (duh...)

[a] *Note:* what I will show you is an example of remote prototyping workflow on the cluster via VSCode. There are many more ways of working with the cluster.

Prerequisites for working on the cluster[a]

[a] *Note:* what I will show you is an example of remote prototyping workflow on the cluster via VSCode. There are many more ways of working with the cluster.

You will need:

- ☑ BHT account (duh...)
- ☑ Cluster Documentation (see above)

Prerequisites for working on the cluster[a]

[a] *Note:* what I will show you is an example of remote prototyping workflow on the cluster via VSCode. There are many more ways of working with the cluster.

You will need:

- ☑ BHT account (duh...)
- ☑ Cluster Documentation (see above)
- ☑ Internet access with a VPN connection: `https://doku.bht-berlin.de/zugang/vpn`

Prerequisites for working on the cluster[a]

You will need:

- ☑ BHT account (duh...)
- ☑ Cluster Documentation (see above)
- ☑ Internet access with a VPN connection: `https://doku.bht-berlin.de/zugang/vpn`
- ☑ Access rights (I took care of this)

[a] *Note:* what I will show you is an example of remote prototyping workflow on the cluster via VSCode. There are many more ways of working with the cluster.

Prerequisites for working on the cluster[a]

[a] *Note:* what I will show you is an example of remote prototyping workflow on the cluster via VSCode. There are many more ways of working with the cluster.

You will need:

- ☑ BHT account (duh...)
- ☑ Cluster Documentation (see above)
- ☑ Internet access with a VPN connection: `https://doku.bht-berlin.de/zugang/vpn`
- ☑ Access rights (I took care of this)
- ☑ Install **kubectl**: `https://kubernetes.io/docs/tasks/tools/`

Prerequisites for working on the cluster[a]

You will need:

- ☑ BHT account (duh...)
- ☑ Cluster Documentation (see above)
- ☑ Internet access with a VPN connection: `https://doku.bht-berlin.de/zugang/vpn`
- ☑ Access rights (I took care of this)
- ☑ Install **kubectl**: `https://kubernetes.io/docs/tasks/tools/`
- ☑ Configure BHT login for **kubectl** (see Moodle and docs)

[a] *Note:* what I will show you is an example of remote prototyping workflow on the cluster via VSCode. There are many more ways of working with the cluster.

Prerequisites for working on the cluster[a]

[a] *Note:* what I will show you is an example of remote prototyping workflow on the cluster via VSCode. There are many more ways of working with the cluster.

You will need:

- ☑ BHT account (duh...)
- ☑ Cluster Documentation (see above)
- ☑ Internet access with a VPN connection: `https://doku.bht-berlin.de/zugang/vpn`
- ☑ Access rights (I took care of this)
- ☑ Install **kubectl**: `https://kubernetes.io/docs/tasks/tools/`
- ☑ Configure BHT login for **kubectl** (see Moodle and docs)
- ☑ Install **docker** (see Moodle)

Prerequisites for working on the cluster[a]

[a] *Note:* what I will show you is an example of remote prototyping workflow on the cluster via VSCode. There are many more ways of working with the cluster.

You will need:

- ☑ BHT account (duh...)
- ☑ Cluster Documentation (see above)
- ☑ Internet access with a VPN connection: `https://doku.bht-berlin.de/zugang/vpn`
- ☑ Access rights (I took care of this)
- ☑ Install **kubectl**: `https://kubernetes.io/docs/tasks/tools/`
- ☑ Configure BHT login for **kubectl** (see Moodle and docs)
- ☑ Install **docker** (see Moodle)
- ☑ *Recommended*: VSCode (with extensions for Python, Containers, Kubernetes, remote-ssh). Feel free to install WSL as well, if on Windows, but you're also fine without it (Linux fans will frown now).

Prerequisites for working on the cluster[a]

[a] *Note:* what I will show you is an example of remote prototyping workflow on the cluster via VSCode. There are many more ways of working with the cluster.

You will need:

- ☑ BHT account (duh...)
- ☑ Cluster Documentation (see above)
- ☑ Internet access with a VPN connection: `https://doku.bht-berlin.de/zugang/vpn`
- ☑ Access rights (I took care of this)
- ☑ Install **kubectl**: `https://kubernetes.io/docs/tasks/tools/`
- ☑ Configure BHT login for **kubectl** (see Moodle and docs)
- ☑ Install **docker** (see Moodle)
- ☑ *Recommended*: VSCode (with extensions for Python, Containers, Kubernetes, remote-ssh). Feel free to install WSL as well, if on Windows, but you're also fine without it (Linux fans will frown now).
- □ *Bonus*: K9s for monitoring your cluster work: `https://k9scli.io/`

Running Python code on the cluster

Example: Printing 'Hello from the cluster!' using a Job

Step 1: Download the course folder *DSWorkflow* from Moodle. Every week we will add new lecture material to this folder.

Step 2: Open the folder in VSCode and check the file *remote-job.yml*.

Step 3: Send it to the cluster:

```
kubectl apply -f remote-job.yml
```

- This will print the message once, then exit!

Step 4: Check your pod's identifier:

```
kubectl get pods
```

Step 5: Check the printed statement:

```
kubectl logs dsw-job-[identifier]
```

Step 6: Clear the job (pod identifier not needed here):

```
kubectl delete job dsw-job
```

Great 'Job'![a]

[a] Pun intended.

You've just printed `"Hello from the cluster!"` ✓

But what if we want to do more than that?

- Write real code directly on the cluster.
- Save files and results persistently.
- Access everything remotely from VSCode.
- ...

Let's start coding like pros ;-)

SSH access to your pod: setup steps

To connect to your Kubernetes pod via SSH, you'll need two things:

- An SSH key pair (for authentication)
- A Kubernetes secret that injects the public key into the pod

Step 1: Generate a new SSH key pair (recommended: with a passphrase)

```
ssh-keygen
```

- This creates two files: id_[rsa] (private) and id_[rsa].pub (public)
- Save them in ~/.ssh/

Step 2: Upload your public key to Kubernetes as a secret

```
kubectl create secret generic dsw-secret \
  --from-file=authorized_keys=~/.ssh/id_[rsa].pub
```

- Replace [rsa] with your actual extension
- This creates a secret that your pod can later mount to enable SSH access.
- Check that the secret was created on the cluster:

```
kubectl get secrets
```

Step 3: Create a local SSH config file (if you don't already have one)

- Create it in /.ssh/config
- Delete the .txt extension if you created it as a text file first

Step 4: Add the following block:

```
Host kubernetes
    HostName localhost
    Port 44414
    User root
    IdentityFile ~/.ssh/id_[rsa]
```

- `HostName localhost`: we connect through port-forwarding.
- `Port 44414`: forwarded to port 22 on the pod.
- `User root`: SSH as root (you'll configure this in the pod).
- `IdentityFile`: your private key for authentication.

Docker: It works on my machine, and yours, and theirs...

What is Docker?

- A way to package your code + all dependencies into a portable unit called **image**.
- When you run an image, you create a **container** - an isolated environment where your app lives e.g. our Moodle platform is also based on a public Docker image.

Image vs Container:

- `Image`: Like a class - blueprint of your app.
- `Container`: Like an object - a running instance of the image.

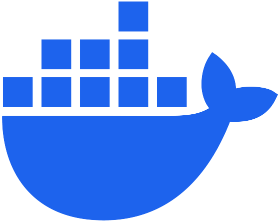


Abbildung: Docker Logo

Writing and building a docker image

Setup on Windows/Mac/Linux:

- Install **Docker Desktop**.
- Make sure it's running before using Docker in the terminal.
- Test with:

```
docker info
```

A Dockerfile defines how to build your image. Let's check ours from Moodle. Make sure:

- The file `sshd_config` is in the same folder as your `Dockerfile`.

Build the image (takes about 5 mins):

```
docker buildx build --platform linux/amd64 \
  -t teochiaburu/dsw-python:3.11.9 .
```

Running, inspecting and pushing the image

Start a container from your image:

```
docker run teochiaburu/dsw-python:3.11.9
```

Useful commands:

- Check containers: `docker ps`
- View logs: `docker logs <container_id>`
- Inspect config: `docker inspect <container_id>`
- Enter container shell: `docker exec -it <container_id> /bin/bash`
- Stop a container: `docker stop <container_id>`

Push image to DockerHub (takes 3-4 mins):

```
docker push teochiaburu/dsw-python:3.11.9
```

Alternatively: Use our university's Docker registry (see cluster docs).

Applying your PVC and Deployment

Check the files *storage.yml* and *remote-deployment.yml*.

Notes about Persistent Volumes (PV)

- Think of it as a **virtual hard disk** managed by the cluster.
- Unlike container file systems, a PV's data **persists across pod restarts**.

Notes about PVC

- A **request for storage** - like saying "Hey Kubernetes, I need 5Gb of space!"
- PVC is a request to K8s to find a suitable PV, it's not the storage itself. Pods (or containers within pods) are what use the PVC to access the storage. They mount the PVC as a volume and that's how containers can read and write data.
- You **must attach** a PVC to a pod/container to make it useful.
- You can mount multiple PVCs into a single pod e.g. shared models + user-specific training results.
- You can **increase** the size of a PVC (if the storage class supports it), but not decrease it. So always start with less and add more space later if necessary.

Applying your PVC and Deployment

```
kubectl apply -f storage.yml

kubectl get pvc # check pvc status

kubectl describe pvc dsw-pvc # check if it's bound to a PV

kubectl apply -f remote-deployment.yml

kubectl get deployments

kubectl get pods

kubectl describe pod [pod-name] # check that your pod is using the volume
```

Port-Forwarding to access pods via VSCode

Why Port-Forwarding?

- Kubernetes pods are not directly reachable from your local machine.
- We use `kubectl port-forward` to map a port on your computer (e.g. 44414) to port 22 (SSH) inside the pod.
- This creates a secure bridge so that VSCode can connect.

Workflow to connect VSCode to the pod via SSH

1. **Start port-forwarding**:

```
kubectl port-forward [pod-name] 44414:22
```

2. **Keep this terminal open!** It's your live SSH bridge.
3. Now you are able to run 'ssh kubernetes' in a new shell, which will open in your home directory as a root user; try 'cd ..' and check the other folders in your pod.
4. In VSCode: `Ctrl + Shift + P` → `Remote-SSH: Connect to Host...` → choose `kubernetes`.

Port-Forwarding to access pods via VSCode

Note: Reinstall extensions each time

- Since pods are ephemeral, every time you connect you'll need to reinstall extensions e.g. Python.
- Shortcut: `Ctrl + Shift + P` → `Remote: Install Local Extensions in 'SSH: kubernetes'` → you can click all of them
- Reload Window (you will now notice that Python code is highlighted differently, since the Python extension is active)

Installing the libraries in your environment

Method 1: In the VSCode console:

```
python3 -m venv .dsw_env

source .dsw_env/bin/activate

which python

pip install -r src/Lecture_1/requirements.txt
```

Method 2:

- `Ctrl + Shift + P` → `Python: Create Environment...` and select the requirements file from there.
- Default name *.venv* and always on the topmost folder level (same as *src*); you can change this afterwards.
- You need to activate the environment *.venv* in the shell first or run a script by clicking on the play button in the top right corner (it will activate the environment automatically)

Check .venv/lib, you will find your libraries there.

Cloning your repo into the PVC

Step 1: Initialize git in your course's local folder, then push to GitHub/Lab.
Step 2: On the cluster, clone the repo under *storage/courses* (you may need to type in your GitHub/Lab credentials every time, because new IP).
Step 3: You have to set your account's default identity before your first push from the cluster, so GitHub/Lab recognizes you as the repo owner pushing from the cluster:

```
git config --global user.email "you@example.com"
git config --global user.name "Your␣Name"
```

You can pick this name such that you can later recognize whether the push came from your local machine or the cluster.

IMPORTANT! Make sure to store everything you need later in the PVC, otherwise it will be lost!! Example:

```
cd ~
mkdir temp_folder
cd temp_folder
echo Hello > hello.txt
cat hello.txt
```

This file disappears when you shut down the pod.

For notebook fans

You can work in VSCode on the cluster with Jupyter notebooks, as well. Your environment just needs to include the juypter dependancies.

Alternatively: BHT JupyterHub

Shutting down your application

IMPORTANT! Always close your deployment pod once you're done!

- Unlike Jobs, Deployments don't shut down automatically once your code is run to completion. They keep running, blocking resources on the cluster (people will hate you for that...).

```
kubectl scale deployment [name] --replicas=0
```

- You can rescale it later back to 'replicas=1', so you don't need to delete your deployment every time and reapply it afterwards (you can, though).
- *Note:* deleting the deployment also deletes the pod it was running in; deleting the pod does not delete the deployment itself.