What is Kubernetes and why we use it?

What is Kubernetes and why we use it?

- Automated Container Management
  - Automates deployment, scaling, and operations of application containers across clusters of hosts.
- Ensures Reliability
  - Maintains high availability, facilitates scaling, and manages rollouts and rollbacks.
- Resource Optimization
  - Efficiently manages resource allocation, maximizing utility and minimizing waste

What is Kubernetes and why we use it?

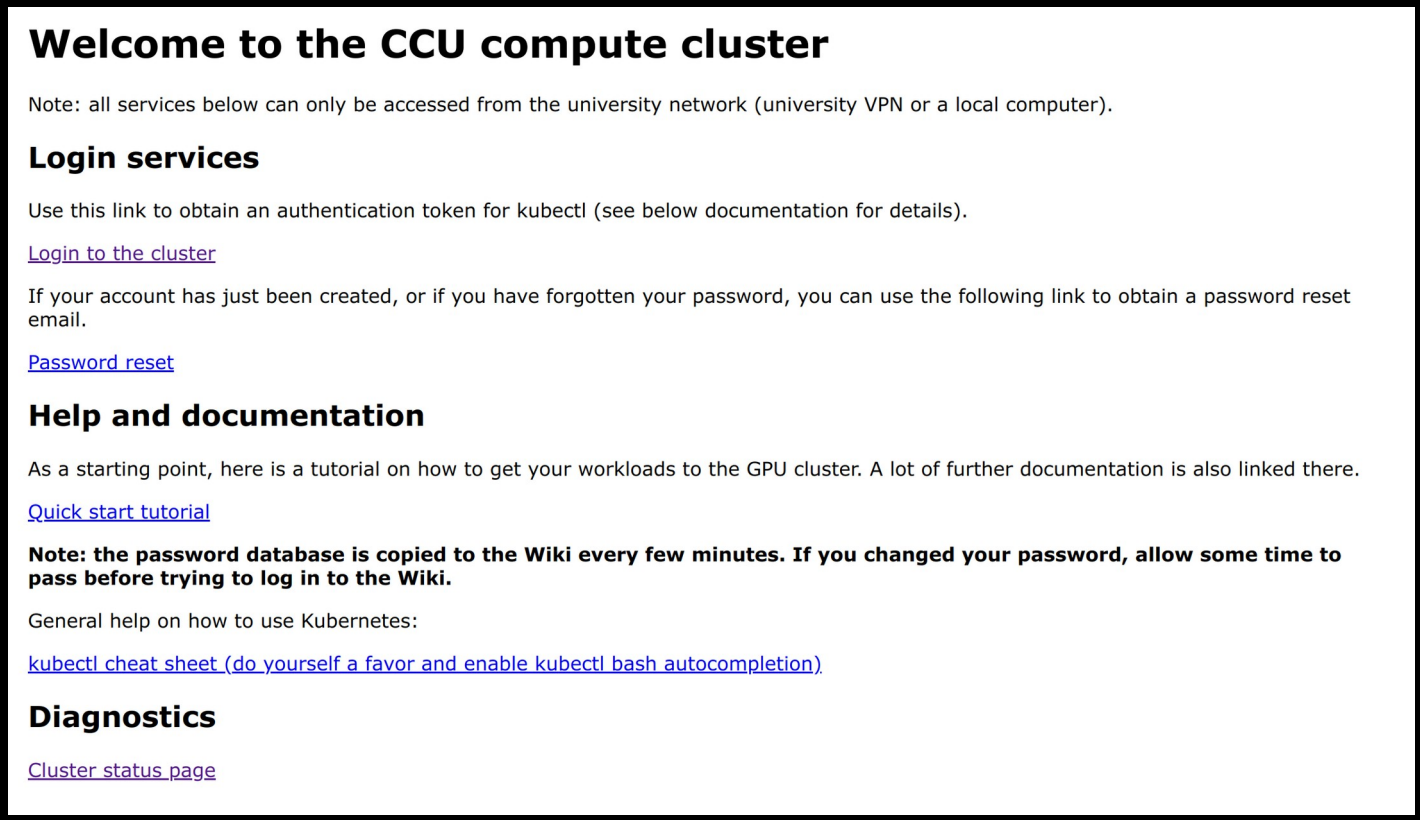
- Automated Container Management
  - Automates deployment, scaling, and operations of application containers across clusters of hosts.
- Ensures Reliability
  - Maintains high availability, facilitates scaling, and manages rollouts and rollbacks.
- Resource Optimization
  - Efficiently manages resource allocation, maximizing utility and minimizing waste
- All major cloud computing solutions use it

Who here already tried to use our cluster?

Step by step guide to our cluster

- After receiving the login details, you will be able to log in here:

  https://ccu-k8s.inf.uni-konstanz.de



Welcome to the CCU compute cluster

Note: all services below can only be accessed from the university network (university VPN or a local computer).

Login services

Use this link to obtain an authentication token for kubectl (see below documentation for details).

Login to the cluster

If your account has just been created, or if you have forgotten your password, you can use the following link to obtain a password reset email.

Password reset

Help and documentation

As a starting point, here is a tutorial on how to get your workloads to the GPU cluster. A lot of further documentation is also linked there.

Quick start tutorial

Note: the password database is copied to the Wiki every few minutes. If you changed your password, allow some time to pass before trying to log in to the Wiki.

General help on how to use Kubernetes:

kubectl cheat sheet (do yourself a favor and enable kubectl bash autocompletion)

Diagnostics

Cluster status page

Wiki

- At the CCU wiki there is a Quick Start guide
  - It will improve significantly in the near future
- The Log in section is well described, steps are basically:
  - Install kubectl in your computer
  - Login in the first link
  - Use the Full Kubeconfig section
  - Alter as described in the wiki
- If everything went properly, type:

  kubectl config use-context ccu-k8s

  kubectl get pods
  - Answer will be: No resources found in namespace user-your-name

Repository access (soon in the wiki)

- Once your kubectl is configured, you should login to the cluster repository:

  sudo docker login ccu-k8s.inf.uni-konstanz.de:32250

- Check that you have a file at:

  /home/calovi/.docker/config.json

- Create a secret

  kubectl create secret generic YOURTAG --from-file=.dockerconfigjson=/home/calovi/.docker/config.json –type=kubernetes.io/dockerconfigjson

- The YOURTAG is what is going to be used in every script submission to the cluster

Pods and Jobs

- When submitting a script to the cluster one usually submit a Pod, or a Job
  - Pod is a continually running container, like the jupyter notebook server, or a session that you want to get into the VM and test what it works
  - Job, is just a sequence of commands, that is run, and once it is finished it automatically closes

    (having the care to save your data in your permanent storage)
- Pods once created, will remain active, and using resources until deleted

Best practices

- Whenever testing a new container use Pods. Test it, run it, check that everything is running well
- Delete your pod to free resources for the rest of the community
- You will receive a permanent storage in the cluster under your namespace (login), the rule of thumb is:
  - Large files should be stored in the permanent storage
  - Many small training files, should be copied to the pod/job temporary storage

Best Practices

- Do you have a data management plan already?
  - If so, soon we will have tutorials how to mount your hot backup drive directly into the cluster

Ilja Werner

Don’t forget to create a data management plan

Best Practices

- Do you have a data management plan already?
  - If so, soon we will have tutorials how to mount your hot backup drive directly into the cluster
- Your storage WILL fail eventually, better be prepared

Ilja Werner

Don’t forget to create a data management plan

Ok, but what is Kubernetes and how do we use it?

- A typical script for a pod will look like this
- Don’t get intimidated, most of the things you don’t have to change

```
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-test-pod
spec:
  containers:
  - name: ubuntu
    image: ubuntu:20.04
    command: ["sleep", "1d"]
    resources:
      requests:
        cpu: 100m
        memory: 100Mi
      limits:
        cpu: 1
        memory: 1Gi
    volumeMounts:
      - mountPath: /abyss/home
        name: cephfs-home
        readOnly: false
      - mountPath: /abyss/shared
        name: cephfs-shared
        readOnly: false
      - mountPath: /abyss/datasets
        name: cephfs-datasets
        readOnly: true
  volumes:
    - name: cephfs-home
      hostPath:
        path: "/cephfs/abyss/home/<your-username>"
        type: Directory
    - name: cephfs-shared
      hostPath:
        path: "/cephfs/abyss/shared"
        type: Directory
    - name: cephfs-datasets
      hostPath:
        path: "/cephfs/abyss/datasets"
        type: Directory
```

Kubectl cheat sheet

- kubectl apply -f Your_Script.yaml
  - How to submit your pod or job to the cluster
- kubectl get pods
  - Describe all your running pods
- kubectl describe pods Your_Script
  - Shows the basic status of your pod and possibly why it failed to be mounted
- kubectl log pods/Your_Pod
  - Terminal messages of your pod, possible errors and other messages
- kubectl delete pods Your_Pod
- kubectl delete -f Your_script.yaml
  - If using the script option, kills all nested services
  - Deleting your Pod from the server
    - Deletes everything that was not saved in your permanent storage
- Kubectl exec -it Your_Pod /bin/bash
  - Same as docker, get inside a running pod

Cluster nodes

- The cluster has different nodes
- Each node can have a different type of GPU, with different attributes

CCU name	Access	Platform	GPUs	Labels	Taints
imp	all	Dual Xeon Rack	4 x Titan Xp @ 12 GB	gpumem=12, gpuarch=nvidia-titan, nvidia-compute-capability-sm70=true	
dretch	all	Dual Xeon Rack	4 x Titan RTX @ 24 GB	gpumem=24, gpuarch=nvidia-titan, nvidia-compute-capability-sm70=true	
belial	exc-cb	Supermicro	8 x Quadro RTX 6000 @ 24 GB	gpumem=24, gpuarch=nvidia-rtx, nvidia-compute-capability-sm75=true	gpumem=24:NoSchedule
fierna	exc-cb	Supermicro	8 x Quadro RTX 6000 @ 24 GB	gpumem=24, gpuarch=nvidia-rtx, nvidia-compute-capability-sm75=true	gpumem=24:NoSchedule
vecna	exc-cb, inf	nVidia DGX-2	16 x V100 @ 32 GB	gpumem=32, gpuarch=nvidia-v100, nvidia-compute-capability-sm80=true	gpumem=32:NoSchedule
zariel	trr161	nVidia DGX A100	8 x A100 @ 40 GB	gpumem=40, gpuarch=nvidia-a100, nvidia-compute-capability-sm80=true	gpumem=40:NoSchedule
tiamat	exc-cb	Supermicro	4 x A100 @ 40 GB	gpumem=40, gpuarch=nvidia-a100, nvidia-compute-capability-sm80=true	gpumem=40:NoSchedule
asmodeus	all	Supermicro	4 x A100 HGX 320 GB, subdivided in 8 GPUs @ 40 GB	gpumem=40, gpuarch=nvidia-a100, nvidia-compute-capability-sm80=true	gpumem=40:NoSchedule
demogorgon	exc-cb	Delta	8 x A40 @ 48 GB	gpumem=48, gpuarch=nvidia-a40, nvidia-compute-capability-sm80=true	gpumem=48:NoSchedule

Taints

- To access a specific node, one can do it by describing correct “taint”
  - Detailed info at the wiki
- One can select a node by name as well by using the nodeSelector

Port forwarding

- At the moment, to have access to the graphical interface inside a pod, one would have to access it via ssh -X
  - Better solutions are being developed, but were not ready for today
- In the Dockerfile_Jupyter one can see how I managed to install ssh and to have access to it
- The container enables the usual port 22 for ssh within the container, but to access it, one has to create a rule for forwarding the port to our local machine

  kubectl port-forward <pod-name> <dest-port>:<source-port>
- If everything was set up appropriately, one can then access it by:
  - ssh -X -p <dest-port> root@localhost
- More commonly this would be used to access browser related pods like jupyter-notebook servers

Port forwarding

- WARNING
  - For most situations, graphical interface to the pod is complete unnecessary
  - Labelling data should be done at your local machine and files transferred to your permanent/temporary storage
  - You can be putting your machine and the cluster at risk depending on how secure your machine is

    (unlikely, but good practice to be cautious)

- ssh -X is a VERY slow way to handle visual interface, if needed check with Ilja and me to see if our alternative is already in place

Questions, break?