Hands-on I: Container building and deployment in homogeneous environments

Vittorio Cozzolino
E-mail: vittorio.cozzolino@huawei.com
Senior Software Engineer
Munich/Dublin Research Center

2023.11.06

DECICE Workshop
Agenda

Part 1

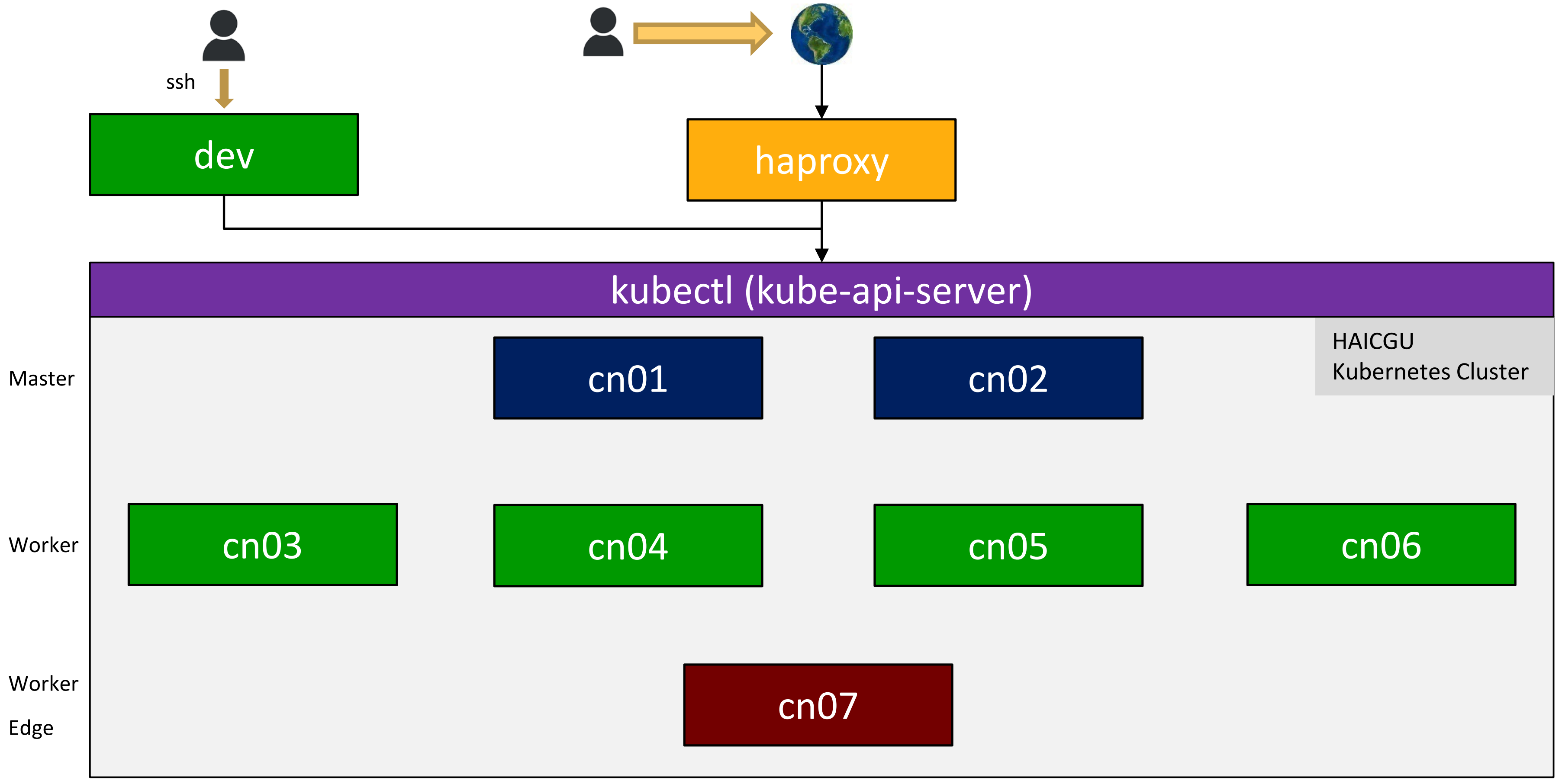
- HAICGU cluster overview
- Login HAICGU cluster
- MQTT application
    - Build containers
    - Push the containers to internal registry
    - Deploy containers on Kubernetes using YAML files
    - Check containers logs and status

Part 2

- Intro about KubeEdge
- Deploy MQTT application on KubeEdge
    - Build and push aggregator container
    - Modify YAML
    - Deploy on edge node

DECICE Workshop – Part 1

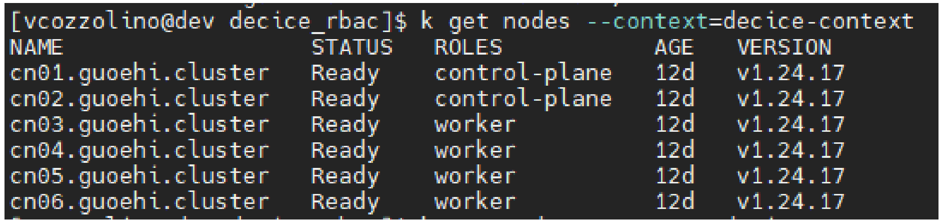
HAICGU Cluster



DECICE Workshop – Part 1

Login HAICGU Cluster

1. SSH to dev node with pub/private key
2. Run this command to copy the k8s cluster config into your home folder
   - `cd && mkdir .kube && cp ../kubernetes/config ~/.kube`
3. Run the command → `kubectl get nodes –A` and check the output. You should see what's shown below.
4. Not all users have the same access rights to the kubernetes cluster:
   1. DECICE users have limited access to the cluster (limited to ``decice`` namespace).
   2. Admins have no visibility restrictions of cluster resources.
5. In this tutorial, we will deploy everything in the `decice` namespace.



```
[vcozzolino@dev decice_rbac]$ k get nodes --context=decice-context
NAME                   STATUS   ROLES           AGE   VERSION
cn01.guoehi.cluster    Ready    control-plane   12d   v1.24.17
cn02.guoehi.cluster    Ready    control-plane   12d   v1.24.17
cn03.guoehi.cluster    Ready    worker          12d   v1.24.17
cn04.guoehi.cluster    Ready    worker          12d   v1.24.17
cn05.guoehi.cluster    Ready    worker          12d   v1.24.17
cn06.guoehi.cluster    Ready    worker          12d   v1.24.17
```

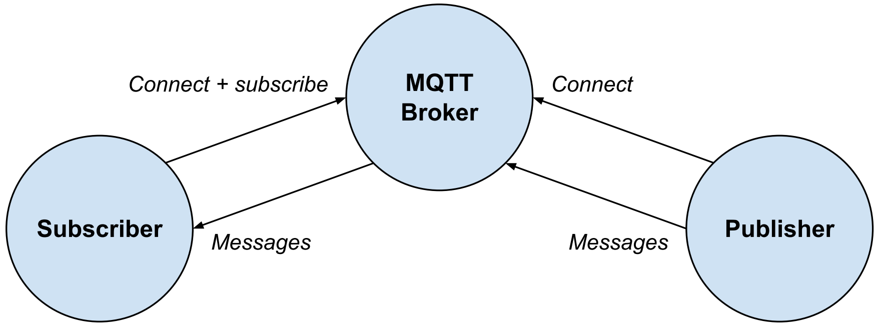
DECICE Workshop – Part 1
Build and Push MQTT Application

Goal: Deploy on the cluster a MQTT publisher and subscriber and check that they can write/read on a predefined topic. On the k8s cluster, a MQTT broker (mosquito) is already installed and running.

In the GitHub repository (https://github.com/rho770/oehi-cc-training.git) you will find:
- Python scripts for the publisher and subscriber.
- Dockerfiles to build the images for publisher and subscriber.
- YAML files to deploy the containers on the k8s cluster.

Note: You can run the Python scripts from shell rather than deploying the containers, however this procedure will not be described in the following slides. To do so, remember to run this command from shell first:

```
module load GCC Python paho-mqtt
```



DECICE Workshop – Part 1
Build and Push MQTT Application

1. Make sure that you are in your home folder → `cd`
2. Pull the repository with → `git clone https://github.com/rho770/oehi-cc-training.git ~/demo/`
3. Move to the demo folder → `cd ~/demo/ workflow1`.
4. In the folder, there are both deploy and undeploy scripts that can be used to automated the steps illustrated below.
5. **Check/Edit** the Python code (optional).
6. **Build** the publisher image with → `podman build -f publisher.dockerfile . -t cn04:30500/"$USER"-publisher:latest`
   - you might need sudo rights to run the podman command!
7. Build the subscriber image with → `podman build -f subscriber.dockerfile . -t cn04:30500/"$USER"-subscriber:latest`
8. **Push** the images:
   1. Inside the Kubernetes cluster, we have deployed a private Docker image registry for DECICE.
   2. To push an image to the private registry, run the commands below:
      1. Push the subscriber image with → `podman push cn04:30500/"$USER"-subscriber:latest`
      2. Push the publisher image with → `podman push cn04:30500/"$USER"-publisher:latest`

9. **Deploy** your containers on Kubernetes using the provided YAML file (*):
   1. Deploy the MQTT publisher → `envsubst < yaml/publisher.yaml | kubectl create -f -`
   2. Deploy the MQTT subscriber → `envsubst < yaml/subscriber.yaml | kubectl create -f -`
10. Check the logs from the pods:
    1. Find out the name of the pods by running first kubectl get pods –n decice
       - `kubectl logs –n decice –f publisher_pod_name`
       - `kubectl logs –n decice –f subscriber_pod_name`
11. To delete the resources created, run the following:
    1. Undeploy the MQTT publisher → `envsubst < yaml/publisher.yaml | kubectl delete -f -`
    2. Undeploy the MQTT subscriber → `envsubst < yaml/subscriber.yaml | kubectl delete -f -`

Extra: Have a look at the YAML files to understand how pods are assigned to k8s nodes and where the connection details for the broker are located.
(*): We use the shell command `envsubst` to replace environment variables in the YAML file with their actual value. In this case, the current shell user username.

DECICE Workshop – Part 2
MQTT Data Aggregation and Collection at the Edge

1. Make sure that you are in your home folder → `cd`
2. Pull the repository with → `git clone https://github.com/rho770/oehi-cc-training.git ~/demo/`
3. Move to the demo folder → `cd ~/demo/ workflow2.`
4. In the folder, there are both deploy and undeploy scripts that can be used to automated the steps illustrated below.
5. **Check/Edit** the Python code (optional).
6. **Build** the aggregator image with → `podman build -f aggregator_cloud.dockerfile . -t cn04:30500/"$USER"-aggregator:latest`
7. Build the collector image with → `podman build -f collector_edge.dockerfile . -t cn04:30500/"$USER"-collector:latest`
8. Build the client image with → `podman build -f client.dockerfile . -t cn04:30500/"$USER"-client-wf:latest`
9. **Push** the images:
   1. To push an image to the private registry, run the commands below:
      - Push the aggregator image with → `podman push cn04:30500/"$USER"-aggregator:latest`
      - Push the collector image with → `podman push cn04:30500/"$USER"-collector:latest`
      - Push the client image with → `podman push cn04:30500/"$USER"-client-wf:latest`
10. **Deploy** your containers on Kubernetes using the provided YAML file:
    1. Deploy the MQTT cloud aggregator → `envsubst < yaml/aggregator_cloud.yaml | kubectl create -f –`
    2. Deploy the MQTT edge collector → `envsubst < yaml/collector_edge.yaml | kubectl create -f -`
    3. Deploy the MQTT client → `envsubst < yaml/client.yaml | kubectl create -f -`
11. Check the logs from the pods:
    1. Find out the name of the pods by running first kubectl get pods –n decice
       - `kubectl logs –n decice –f client_pod_name`
12. To delete the resources created, run the following:
    1. Undeploy the MQTT aggregator → `envsubst < yaml/aggregator_cloud.yaml | kubectl delete -f -`
    2. Undeploy the MQTT collector → `envsubst < yaml/collector_edge.yaml | kubectl delete -f -`
    3. Undeploy the MQTT client → `envsubst < yaml/client.yaml | kubectl delete -f -`

Extra: Have a look at the YAML files to understand how we scheduled the collector on the KubeEdge edge node.