Hands-on II: Container building and deployment in heterogeneous environments

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

2023.11.06

HUAWEI

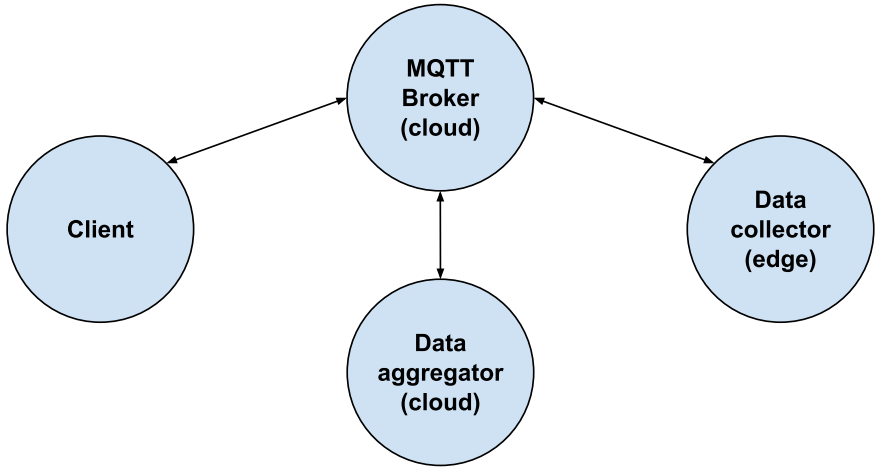
Part 2

Goal: Deploy on the cluster a MQTT data aggregator (cloud), a data collector (**edge**) and client (check figure).

In the GitHub repository (https://github.com/rho770/oehi-cc-training.git) you will find:

- Python scripts for the aggregator, collector, and client.
- Dockerfiles to build the images for aggregator, collector, and client.
- YAML files to deploy the containers on the k8s cluster.

You can run the Python scripts from shell rather than deploying the containers, however this procedure will not be described in the following slides.



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.