Exercise 7: Platform as a Service with Heroku and OpenShift

Welcome to exercise 7. This time we will have the following lessons:

1. Concepts of Platform as a Service
2. PaaS with OpenShift

Lessons learned

Lesson 1: Concepts of Platform as a Service

So far we have been working with *Infrastructure as a Service* and automated as much work as possible. The approach of *Platform as a Service (PaaS)* is even one step further: the automation of infrastructure and deployment is invisible for the user.

Research: What is Platform as a Service?

Do some research about Platform as a Service.

Question: Platform as a Service

- What are benefits of PaaS compared to IaaS?
- What are drawbacks and limitations of PaaS compared to IaaS?

Research: Platform as a Service with Heroku

One proprietary popular PaaS providers is Heroku [1]. Have a look at the product web page and the documentation of Heroku [2]. Heroku supports automatic scaling of applications [3].

[1] https://www.heroku.com/

[2] https://devcenter.heroku.com/articles/how-heroku-works

[3] https://blog.heroku.com/heroku-autoscaling

Question: Heroku

- How does Heroku allow you to deploy an application like Mediawiki?
- Which metric and what statistical pattern is used for autoscaling?

Lesson 2: OpenShift

OpenShift is available as a hosted service and as proprietary and open source software [1] for private hostings. OpenShift uses Kubernetes to provide PaaS, similar to Heroku.

[1] https://www.openshift.org/

Task: Install OpenShift

The terraform-openshift.zip provides terraform scripts, which installs OpenShift on bwCloud automatically. Download and unzip the archive, change the provider.tf to fit your *username and tenant*, change the *key name* in instances.tf. The installer needs ssh access between the VMs. Place your *bwCloud private key* on the file `init_openshift_master`.

Then use `terraform apply` to install your own OpenShift instance. Terraform deploys four virtual machines on bwCloud: an openshift-master and three openshift-nodes. OpenShift will be installed via Cloud-init asynchronously, this will take some time! Cloud-init will trigger the ansible based installation of OpenShift [1].

Check the output of cloud-init to validate that the installation was successful, then access the OpenShift dashboard at http://PUBLIC_IP_Master:8443/. Besides this graphical web UI, OpenShift offers the CLI tool `oc`, which is installed already in the openshift-master vm.

[1] https://docs.openshift.org/latest/install_config/install/advanced_install.html

Task: Install test application on OpenShift

On the web dashboard, you'll be asked to create a new project. Do so and name it e.g. `clouds-exercise`. On the next page titled “Add to Project” select “Deploy Image”, select “Image Name”, provice “ghost” and press enter. Press the create button of the appearing form panel. Click on “Continue to overview.” to follow the deployment of the ghost blog application.

While OpenShift is downloading the Docker image and creating the Docker container, we need to create a route to be able to access the ghost application. In the overview , click the link in the top right corner “create router”. Specify as hostname `bwcloud-vmXYZ.rz.uni-ulm.de`, the hostname of your openshift master, then press the create button. On the overview page, you will now have a link to the deployed ghost application.

Task: Maintaining the test application

In the overview dashboard, the ghost application can be scaled up and down by pressing the two arrows next to the circle with the number of pods running for the deployment. OpenShift has also automatic health checks and even autoscaling.

In the menu bar, go to Applications > Deployments, then select the ghost deployment. In the Actions menu, edit the health checks and enable an autoscaler.

Question: Maintaining OpenShift Deployments

- What types of health checks does OpenShift offer?
- According to which metrics does the autoscaler work?