Configuration Management Tools - Lab Assignment 1

Welcome to your first assignment. The work here should teach you to related topics to what we have been discussing in lectures. The structure is quite simple.

3 lab assignments will be given during the course, each consisting of **five tasks**. Typically an assignment is supposed to be done **within a week**, but will have a **two week deadline**.
If every task is completed satisfactory, **tasks 1-3 will yield 1** point each and are required to pass the course, **task 4 will yield you 1 extra point** and **task five will yield 3 extra points** counting towards a higher grade. **All assignments are done individually**.

Additionally there will be one **mini-project yielding 4 points**, the project is done in group and is required.

Start by making a private repo for the course on github, then invite “MaxAndersson” to your repo. Please keep you commits small and keep your repo up to date. This way I can follow your progress, give you feedback etc. If you need help then you can always refer to the commit you are working on. Your tag your commits with A1 for Lab Assignment 1 and push to github, then send me the commit SHA that can be found by typing “git log” into your terminal. Tip: If you want to clone another repo into your existing one have a look at using git submodules instead.

Written material should be available by the deadline and to get practical parts approved will be done by showing me what you have done, either during workshop or schedule an online session with me by sending an email to max@scaleoutsystems.com

Passing criteria:

- 13 points for passing grade (G)
- 22 points for pass with distinction (VG)

All written material should be handed in as a separate **pdf or markup** in your repo, containing your **name** and **assignment number** in both the title and filename

Task 1: Describe/Explain the following:

- IaaS, PaaS & SaaS (hint: Nist)
- OS-level virtualization
- cgroups
- Copy-on-Write (COW) & Snapshots
- High-Availability
- Idempotency
- Mutable vs Immutable Infrastructure
- Configuration Management vs. Orchestration
- Procedural vs Declarative
- Git Submodule

- Ansible
    - Inventory File
    - Playbook
- Kubernetes
    - Compare Stateful and Stateless Applications
    - ReplicaSets,Deployments, Pods & Services

Task 2: Time to build & deploy!

- Subtask 1:
    - Clone the project from github (https://github.com/yhl17cmt/Demo1WebApp)
    - Write a multi-stage Dockerfile to build with the maven image and deploy with the tomcat image
    - Make sure it runs.
    - Question 1: Why might you want to make a multi-stage build ?
- Subtask 2:
    - Clone the project from github (https://github.com/yhl17cmt/business-data)
    - Write a Dockerfile from mysql or mariadb image and load the sql dump by mounting a volume.

      Hint:
      $ docker run --rm -it -v \
      $(pwd)/Northwind_Database.sql:/docker-entrypoint-initdb.d/dump.sql -e \
      "MYSQL_ALLOW_EMPTY_PASSWORD=true" mariadb
    - Make sure it works.
    - Question 2: Mounting database dumps like this can be a good solution to use for a test database, but why might you not want to do it like this for a production database?
- Subtask 3:
    - Deploy the containers built in subtask(ST) 1 & 2 with ansible using a docker module and preferably a dynamic inventory file for something like openstack, vagrant etc..

Task 3: Changing environment. Now it's time to take a look at something else. We will take a look at setting up Kubernetes. Don't worry that we haven't talked that much about the inner workings of kubernetes yet. This task focuses more on getting things up and running.
For local development it's common to use alternatives such as minikube to get up and running with kubernetes, and there are also managed kubernetes clusters available from all major cloud providers in the market.

But for this task we will set up kubernetes with the help of ansible that we have already gotten a little familiar with. Luckily we do not have to do everything ourselves, we will be using something called kubespray. Here are a couple links to get you started

- https://kubernetes.io/docs/setup/custom-cloud/kubespray/ - Kubernetes Documentation
- https://github.com/yhl17cmt/kubespray - Kubespray fork

To help you along the way there is a guide appended to the end of this document to help you get started.

Task 4: Define a **Deployment** equal to T2 and deploy a **Pod/ReplicaSet** to your cluster and expose it as a **Service**.

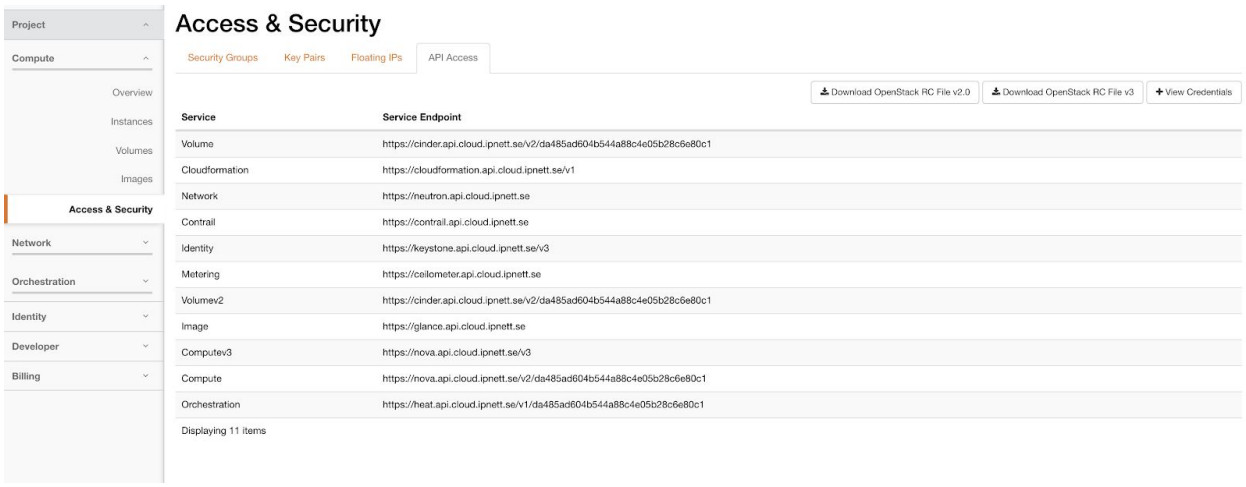
Task 5: Write a short essay min. 1-2 pages (11 pt, Arial or Times New Roman, reference system of choice) about **Microservices** including, but not limited to:

- Benefits and Disadvantages compared to a Monolithic architecture
- A suitable Use Case for Microservices
- Two or more architectural patterns explained.

Installing Kubernetes with Kubespray on OpenStack.

Obtaining a openstack profile

- Login to the openstack portal with your credentials.
- Navigate to Access & Security
- Press the Download OpenStack RC File v3.0



To continue with the process add the variables to your profile.

source <<file you just downloaded>>

Download kubespray

Download kubespray from source or run git clone.
Kubespray you can find here.

- https://github.com/yhl17cmt/kubespray

If you want to know the differances between Kubespray and other popular projects you can read this. https://github.com/yhl17cmt/kubespray/blob/master/docs/comparisons.md

Proceed by cloning or forking the repository.
git clone https://github.com/yhl17cmt/kubespray

Install dependencies

sudo apt update

sudo apt install ansible

Install the dependencies for kubespray
Change directory to the kubespray root folder.
Install dependencies using
python3 -m pip install -r requirements.txt

Setup the openstack cli

To find out the variables from openstack download the openstack client.
It will work with the RC file sourced that you previously downloaded.

If you are on a freshly installed Ubuntu system for example. Install
apt install python3-dev python3-pip

To install the openstack client
python3 -m pip install python-openstackclient

Then you are able to access the openstack CLI from a terminal by using
openstack

Try it out with
openstack help

Configure your cluster

From the kubespray root folder you are now ready to make a copy of the sample project and start configuring.

export CLUSTER=myclustername
cp -LRp contrib/terraform/openstack/sample-inventory \
inventory/$CLUSTER
cd inventory/$CLUSTER
ln -sf ../../contrib/terraform/openstack/hosts
ln -sf ../../contrib

Now it’s time to configure your cluster.
Start by editing the cluster.tf file

<< cluster.tf file here.

Basic setup

Set public_key_path
Give an ssh key you can use to access the nodes.
If you dont have any key you need to generate one. (See other documentation).

Set image name
image = "ubuntu-18.04-server-cloudimg-amd64-20190212.1"

Set user
For ubuntu set
ssh_user = “ubuntu”

network_name = “your-network-here” # will be created
external_net = "71b10496-2617-47ae-abbc-36239f0863bb" # need to be provided. You find this by running the openstack cli

Select your subnet_cidr.
 For example
subnet_cidr = "192.168.100.0/24"   # You might need to change to another subnet.
floatingip_pool = "public-v4"   # use the existing pool in openstack. Find this by using the openstack cli or GUI.
Select your cluster parameters number of nodes etc.

Deploy cluster resources using ansible

terraform init ../../contrib/terraform/openstack
terraform apply -var-file=cluster.tf ../../contrib/terraform/openstack

Deploy kubernetes on openstack infrastructure

Start by setting the cloud_provider to openstack in
inventory/$CLUSTER/group_vars/all/all.yml

Your cluster is default configured to use the network module “calico” if you want to edit the cluster go through the group_vars mainly all.yml and k8s-cluster/k8s-cluster.yml files to configure according to your needs.

ansible-playbook --become -i inventory/safespring/hosts cluster.yml --flush-cache

Optional (get openstack parameters for cluster.tf)

Using the openstack client you are to be able to access uuid for flavor and network.
Openstack flavor list
Openstack images list
Get the name of the floating ip pool
For Safespring cloud (cluster-v4)
Openstack floating ip pool list
If problems here you can check the security groups and ensure that
ansible-playbook --become -i inventory/safespring-test/hosts cluster.yml --flush-cache

Troubleshooting.
If there are errors. Check first that the ports are open.

Common problem!
You may need to manually add security groups to the nodes for example ssh access.
ICMP is generally also not allowed thus making the playbook ping to fail.
Add ICMP to a security group for the hosts temporarily for creation if desired.

Login to your master node.
sudo cp /etc/kubernetes/admin.conf $HOME/.
sudo chown $(id -u):$(id -g) admin.conf
export KUBECONFIG=admin.conf
If you want to continue using this cluster config you can add this to your environment by placing the export … line in your .bashrc or .bash_profile

Test your deployment

Test your deploy with hello-world.
kubectl create deployment hello-node --image=gcr.io/hello-minikube-zero-install/hello-node
kubectl expose deployment hello-node --type=NodePort--port=33200
Navigate to any of the worker nodes on the port you specifed or use curl.

Install nfs-common on all nodes
Set a default security group with ingress 111 and 2049 and egress all for proper communication.

References

https://docs.openstack.org/newton/user-guide/common/cli-install-openstack-command-line-clients.html