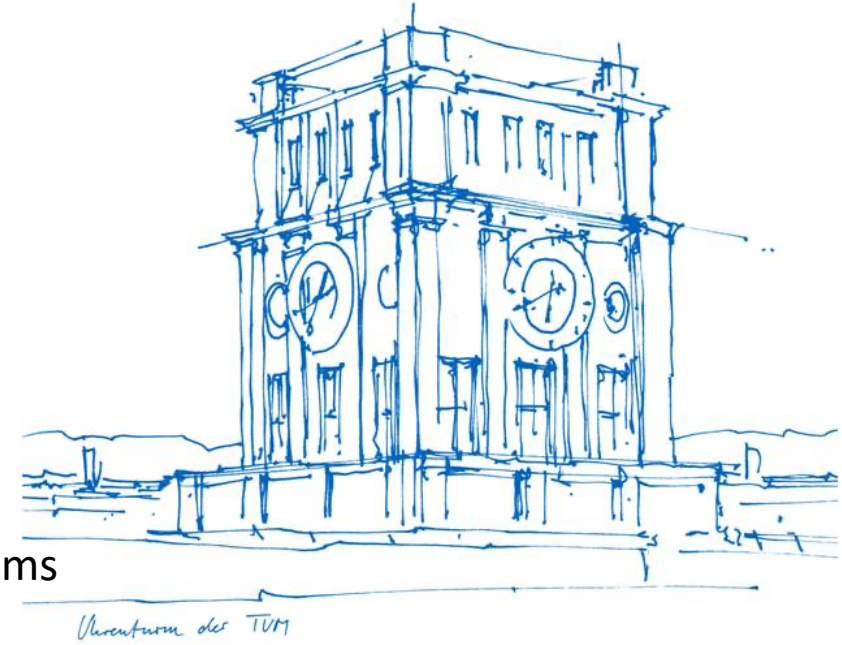
Cloud Computing Exercise – 4

Application Deployment using Kubernetes



Anshul Jindal (M.Sc. Informatics)

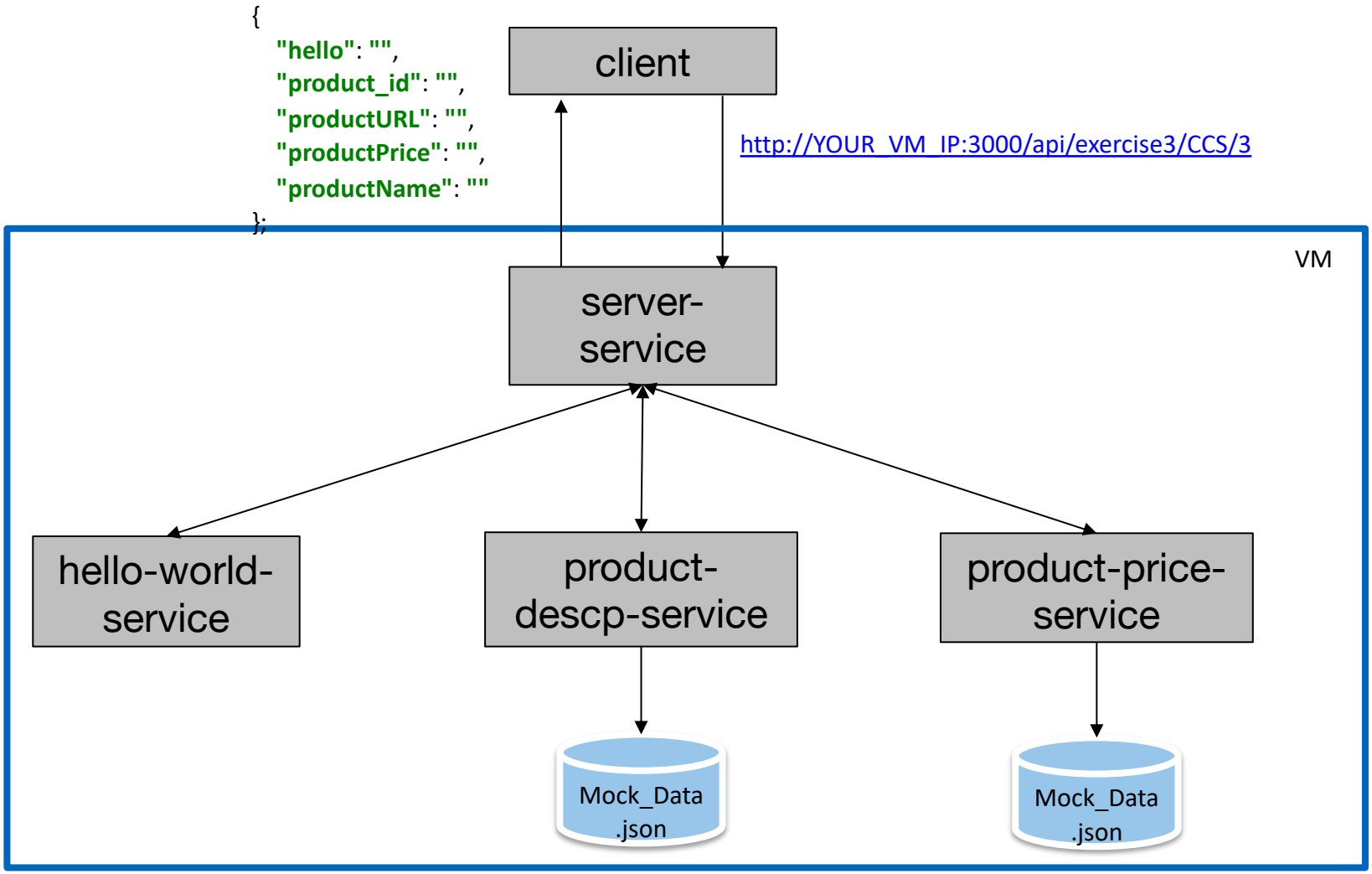
anshul.jindal@tum.de

Chair of Computer Architecture and Parallel Systems

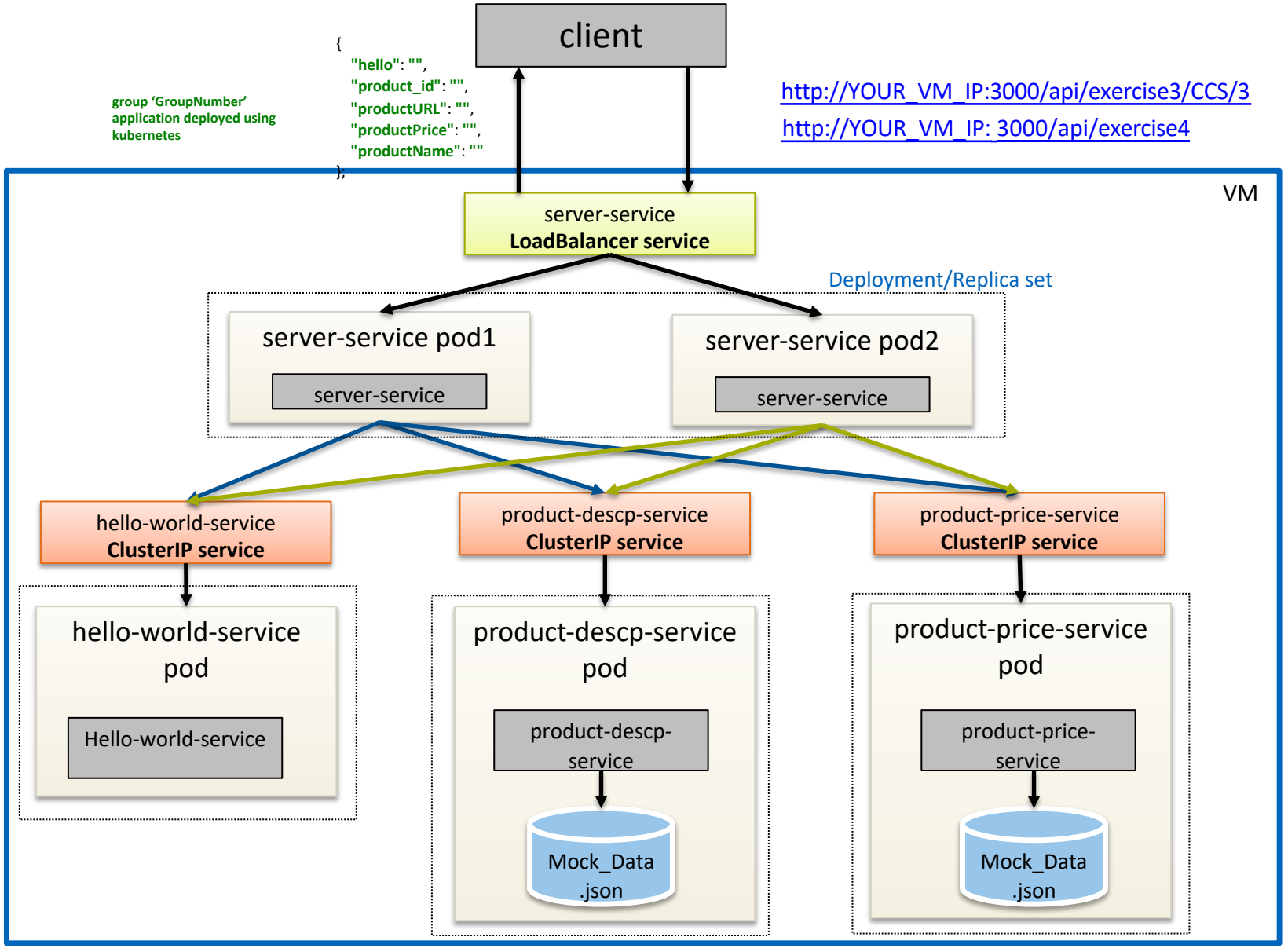
Technical University of Munich (TUM), Germany

Exercise 4

Exercise3 Application Architecture



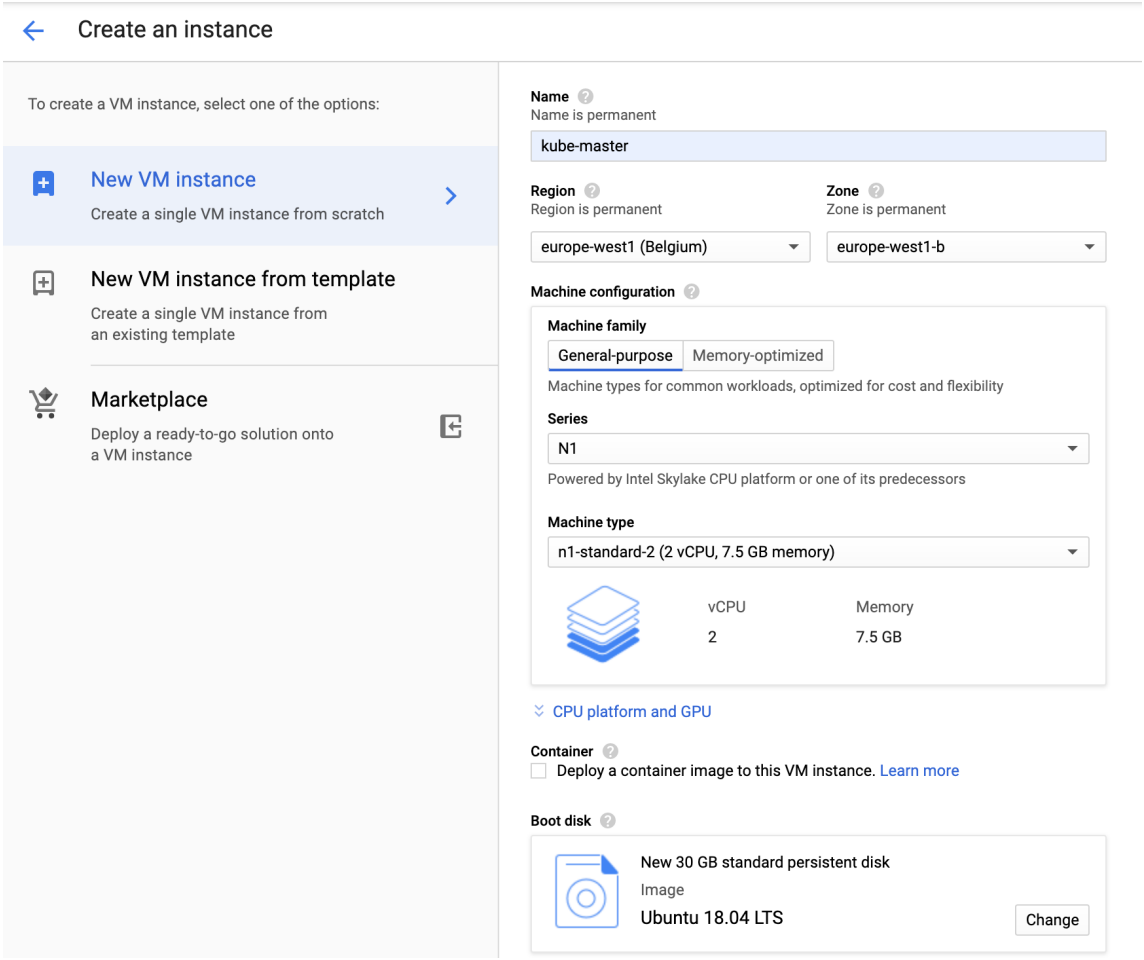
Exercise 4 Application Architecture



Kubernetes Installation
and
Running the application

Launching an EC2 Master Instance

- Create a new VM on GCP (select instance > = n1-standard-2)



- Kubernetes cluster operate on the below mentioned ports, so enable these.
  - 30000-32767 (node port range)
  - 8001, 443, 6443 (for Kubernetes communication)

SSH to Master

SSH into Master VM

```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Dec  2 22:26:17 UTC 2019

  System load:  0.12              Processes:           95
  Usage of /:   3.9% of 28.90GB   Users logged in:     0
  Memory usage: 6%                IP address for ens4: 10.132.0.4
  Swap usage:   0%

0 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

anshul_j6110@kube-master:~$
```

Install docker and Kubernetes

1. Install packages to allow apt to use a repository over HTTPS

```
sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
software-properties-common
```

2. Add Docker's official GPG (GNU Privacy Guard) key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
```

3. Use the following command to set up the stable repository.

```
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
```

Install docker and Kubernetes Cont..

4. Switch to root user

```
sudo su root
```

5. Add Kubernetes repositories

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
```

6. Switch to the normal user

```
su <orginal user name>
```

7. Update the apt package index.

```
sudo apt-get update
```

8. Install the latest version of Docker by using this command.

```
sudo apt-get install -y docker-ce
```

9. Installation kubeadm, kubernetes and kubectl

```
sudo apt-get install -y kubelet kubeadm kubernetes-cni
```

Installation

We will be using kubeadm to deploy the kubernetes Cluster.

- Install Docker, Kubernetes, Kubeadm and Kubectl on Master and Slave nodes
  (As part of the exercise we are not using slave nodes)
- Check the Installation by running kubectl command, you would get something like this

```
anshul_j6110@kube-master:~$ kubectl
kubectl controls the Kubernetes cluster manager.

 Find more information at: https://kubernetes.io/docs/reference/kubectl/overview

Basic Commands (Beginner):
  create         Create a resource from a file or from stdin.
  expose         Take a replication controller, service, deployment or pod and e
  run            Run a particular image on the cluster
  set            Set specific features on objects

Basic Commands (Intermediate):
  explain        Documentation of resources
  get            Display one or many resources
  edit           Edit a resource on the server
  delete         Delete resources by filenames, stdin, resources and names, or b

Deploy Commands:
  rollout        Manage the rollout of a resource
  scale          Set a new size for a Deployment, ReplicaSet, Replication Contro
  autoscale      Auto-scale a Deployment, ReplicaSet, or ReplicationController

Cluster Management Commands:
  certificate    Modify certificate resources.
  cluster-info   Display cluster info
```

Step 2 - **Configuring Kubernetes**

Initialize the Master Node using kubeadm init command **(need to be run as root)**

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
anshul_j6110@kube-master:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.16.3
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. Th
/docs/setup/cri/
        [WARNING SystemVerification]: this Docker version is not on the list of validated v
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connec
[preflight] You can also perform this action in beforehand using 'kubeadm config images pul
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubea
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
```

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.132.0.5:6443 --token if7uyp.6kfghli2a73d7lzy \
    --discovery-token-ca-cert-hash sha256:8481b25a6b4afbfe424997f64336154e2c81ed1f22b2ef26cca364ccd6a76b2a
```

To be run on the slave nodes for joining the kubernetes cluster

Step 2 - **Configuring Kubernetes Cont..**

- Before going forward, you should create a new user and add it to sudoers and run the following commands on it:

```
sudo mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- Check everything is running fine by running command kubectl get nodes

```
anshul_j6110@kube-master:~$ kubectl get nodes
NAME          STATUS     ROLES    AGE    VERSION
kube-master   NotReady   master   6m5s   v1.16.3
```

Step 3 - **Installing the Pod Network**

- Master is up so we need to install the pod network.
- It is necessary to do this before you try to deploy any applications to your cluster, and before kube-dns will start up.
- See the add-ons page for a complete list of available network add-ons. To install an add-on run this command: Example: kubectl apply -f <add-on-name.yaml>
- We will be installing flannel, which provides networking and network policy.

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/2140ac876ef134e0ed5af15c65e414cf26827915/Documentation/kube-flannel.yml

```
anshul_j6110@kube-master:~$ kubectl apply -f https://raw.githubuserconte
podsecuritypolicy.policy/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds-amd64 created
daemonset.apps/kube-flannel-ds-arm64 created
daemonset.apps/kube-flannel-ds-arm created
daemonset.apps/kube-flannel-ds-ppc64le created
daemonset.apps/kube-flannel-ds-s390x created
```

Step 4 – **Status Check**

- Check the status of pods run the following command.

kubectl get pods --all-namespaces

```
anshul_j6110@kube-master:~$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                   READY   STATUS    RESTARTS   AGE
kube-system   coredns-5644d7b6d9-5ql4n               1/1     Running   0          12m
kube-system   coredns-5644d7b6d9-mm5gd               1/1     Running   0          12m
kube-system   etcd-kube-master                       1/1     Running   0          11m
kube-system   kube-apiserver-kube-master             1/1     Running   0          11m
kube-system   kube-controller-manager-kube-master    1/1     Running   0          11m
kube-system   kube-flannel-ds-amd64-ncwzw            1/1     Running   0          74s
kube-system   kube-proxy-42xrt                       1/1     Running   0          12m
kube-system   kube-scheduler-kube-master             1/1     Running   0          11m
```

- You can also run this command to check the status of pods:

watch kubectl get pods –all-namespaces

It automatically gets refreshed after 2 seconds

- Check the status of node using the command kubectl get nodes

```
anshul_j6110@kube-master:~$ kubectl get nodes
NAME          STATUS   ROLES    AGE   VERSION
kube-master   Ready    master   14m   v1.16.3
```

Step 5 – **Joining the nodes**

- By default, your cluster will not schedule pods on the master for security reasons.
- If you want to be able to schedule pods on the master, e.g. a single-machine Kubernetes cluster for development, run the following command on master:

kubectl taint nodes --all node-role.kubernetes.io/master-

```
anshul_j6110@kube-master:~$ kubectl taint nodes --all node-role.kubernetes.io/master-
node/kube-master untainted
```

(Worker)

- Slave nodes can be joined by running the kubeadm join command as taken note while doing kubeadm init on master node.

Kubernetes Workloads

- Workloads within Kubernetes are higher level objects that manage Pods or other higher level objects.

ReplicaSet

- Primary method of managing pod replicas and their lifecycle.
- Includes their scheduling, scaling, and deletion.
- Their job is simple: Always ensure the desired number of pods are running.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-example
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
template:
    <pod template>
```

Deployment

- Declarative method of managing Pods via ReplicaSets.
- Provide rollback functionality and update control.
- Updates are managed through the pod-template-hash label.

Pod Template

```
template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
    - name: nginx
      image: nginx
```

Kubernetes Workloads Cont..

Deployment

- revisionHistoryLimit: The number of previous iterations of the Deployment to retain.
- strategy: Describes the method of updating the Pods based on the type. Valid options are
  - **Recreate:**

    All existing Pods are killed before the new ones are created.
  - **RollingUpdate:**

    Cycles through updating the Pods according to the parameters:
    - maxSurge: how many additional replicas to spin up while updating.
    - maxUnavailable: how many may be unavailable during the update.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-example
spec:
  replicas: 3
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    <pod template>
```

More..

Step 7 – **Running your containerized services**

- We create deployments for each service, the hello-world deployment file looks like : (kubernetes_files/deployments/hello-world.yml)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world-deployment
  labels:
    app: hello-world
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image: HUB_ID/microservice:hello
          ports:
            - containerPort: 9001
```

Type of workload

Name of deployment

Labels for referring

Specification about pod

Number of replicas

Template of the pod

Specification of the container

Container Name

Image name

Container Port

Step 7 – **Running your containerized images**

Before creating deployments :

- Add the image name in kubernetes_files/deployments/hello-world.yml file.
- Complete the missing product-descp.yml, product-price.yml and server.yml files.

After that run, the deployments for the microservices using the command

kubectl apply -f kubernetes_files/deployments/< file_name>.yml

```
anshul_j6110@kube-master:~/Assignment$ kubectl apply -f kubernetes_files/deployments/hello-world.yml
deployment.apps/hello-world-deployment created
anshul_j6110@kube-master:~/Assignment$ kubectl apply -f kubernetes_files/deployments/product-descp.yml
deployment.apps/product-descp-deployment created
anshul_j6110@kube-master:~/Assignment$ kubectl apply -f kubernetes_files/deployments/product-price.yml
deployment.apps/product-price-deployment created
anshul_j6110@kube-master:~/Assignment$ kubectl apply -f kubernetes_files/deployments/server.yml
deployment.apps/server-deployment created
```

Step 7 – **Running your containerized images Cont..**

- Check the status of all the pods in the deployments by running the command.

kubectl get pods --all-namespaces

```
anshul_j6110@kube-master:~/Assignment$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                    READY   STATUS    RESTARTS   AGE
default       hello-world-deployment-7bbc8f69b9-mclj6 1/1     Running   0          4m36s
default       product-descp-deployment-744fbff996-8r9t9 1/1   Running   0          4m22s
default       product-price-deployment-5d68fc7cd7-46cxg 1/1   Running   0          4m17s
default       server-deployment-f6dd896f5-mck9g       1/1     Running   0          4m11s
kube-system   coredns-5644d7b6d9-f8q74                1/1     Running   0          14m
kube-system   coredns-5644d7b6d9-n45b5                1/1     Running   0          14m
kube-system   etcd-kube-master                        1/1     Running   0          13m
kube-system   kube-apiserver-kube-master              1/1     Running   0          13m
kube-system   kube-controller-manager-kube-master     1/1     Running   0          13m
kube-system   kube-flannel-ds-amd64-8z6kf             1/1     Running   0          7m37s
kube-system   kube-proxy-mc4w8                        1/1     Running   0          14m
kube-system   kube-scheduler-kube-master              1/1     Running   0          13m
```

- Check the status of deployments:

kubectl get deployments --all-namespaces

```
anshul_j6110@kube-master:~/Assignment$ kubectl get deployments --all-namespaces
NAMESPACE     NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
default       hello-world-deployment  1/1     1            1           5m52s
default       product-descp-deployment 1/1    1            1           5m38s
default       product-price-deployment 1/1    1            1           5m33s
default       server-deployment       1/1     1            1           5m27s
kube-system   coredns                 2/2     2            2           15m
```

Step 7 – **Running your containerized images Cont..**

- As all the microservices are running in different pods so we need to create kube-services for each of them to complete the interaction.
- All the kube-services are in (kubernetes_files/services/)

```
apiVersion: v1
kind: Service
metadata:
  name: hello-world-service
spec:
  selector:
    app: hello-world
  ports:
    - protocol: TCP
      port: 9001
      targetPort: 9001
```

Type of workload

Name of the kube-service. It is same as in the docekr-compose.yml file for last exercise

Name of the pod to connect this with.

Container Port and VM port

Step 7 – **Running your containerized images Cont..**

Before creating services :

- Complete the missing product-descp.yml, product-price.yml files in kubernetes_files/services/.
- We expose our server microservice to the outside world so its type is **LoadBalancer**.

After that, run the kube-services for the microservices using the command :

kubectl apply -f kubernetes_files/services/<file_name>.yml

```
anshul_j6110@kube-master:~/Assignment$ kubectl apply -f kubernetes_files/services/hello-world.yml
service/hello-world-service created
anshul_j6110@kube-master:~/Assignment$ kubectl apply -f kubernetes_files/services/product-descp.yml
service/product-descp-service created
anshul_j6110@kube-master:~/Assignment$ kubectl apply -f kubernetes_files/services/product-price.yml
service/product-price-service created
anshul_j6110@kube-master:~/Assignment$ kubectl apply -f kubernetes_files/services/server.yml
service/server-service created
```

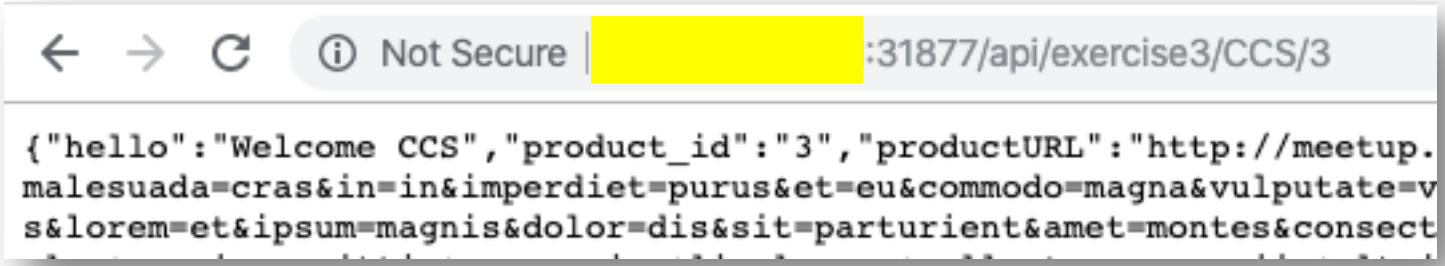
Step 7 – **Running your containerized images Cont..**

- Get the kube-services by running the command

kubectl get services --all-namespaces

```
anshul_j6110@kube-master:~/Assignment$ kubectl get services --all-namespaces
NAMESPACE     NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
default       hello-world-service     ClusterIP      10.106.123.151   <none>        9001/TCP
default       kubernetes              ClusterIP      10.96.0.1        <none>        443/TCP
default       product-descp-service   ClusterIP      10.111.88.181    <none>        9002/TCP
default       product-price-service   ClusterIP      10.109.99.70     <none>        9003/TCP
default       server-service          LoadBalancer   10.105.60.60     <pending>     3000:31877/TCP
kube-system   kube-dns                ClusterIP      10.96.0.10       <none>        53/UDP,53/TCP,9153/TCP
```

- Here the external-IP is your **VM public IP** and the port number is **31877**.
- Your Application would be running at address http://VM_IP:PORTNUMBER/api/exercise3/CCS/3
- http://VM_IP:PORTNUMBER/api/exercise4



Step 8 – **Scaling your deployment**

- Before Scaling

```
anshul_j6110@kube-master:~/Assignment$ kubectl get deployment server-deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
server-deployment   1/1     1            1           19m
```

- Scaling can be done by following command:

kubectl scale deployment <deployment_name> --replicas=<replicaNumber>

```
anshul_j6110@kube-master:~/Assignment$ kubectl scale deployment server-deployment --replicas=2
deployment.apps/server-deployment scaled
anshul_j6110@kube-master:~/Assignment$ kubectl get deployment server-deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
server-deployment   2/2     2            2           21m
```

Deleting and Resetting the Cluster

Do this step only if you need to redeploy the Kubernetes cluster or some workloads.

- To delete the service and deployment you can run the following command:

```
kubectl delete service,deployment <deployment_Name>
```

- Reset all kubeadm installed state, run the following command on master

```
kubeadm reset
```

- Delete the configuration file

```
sudo rm -r $HOME/.kube/config
```

Tasks to be Completed

Tasks to be completed

As part of the exercise4, following are the tasks to be completed:

1. Add an API endpoint in your Server Microservice and push the images to docker-hub:
    1. **/api/exercise4: Send a message “group # application deployed using kubernetes”**
2. Install docker and Kubernetes on the VM.
3. After installation run this application on the VM using Kubernetes as explained in previous slides:
    a. Start Kubernetes Cluster
    b. Install Pod Network
    c. Enable Pod Scheduling on Master node.
    d. Run all microservices deployments.
    e. Create kube services for all the microservices.
    f. Scale Server microservice to have 2 replicas
    g. Expose Kubernetes API : sudo kubectl proxy --address='0.0.0.0' --port=8001 --accept-hosts='^*$’&
    h. Check the status and port number of microservices.
    i. Visit the URL to test the application: http://YOUR_VM_IP:PORTNUMBER/api/exercise3/CCS/3
    http://YOUR_VM_IP:PORTNUMBER/api/exercise4

Submission

Submission Instructions

To submit your application results you need to follow this :

1. Open the Cloud Class server url : https://cloudcom.caps.in.tum.de/
2. Login with your provided username and password.
3. After logging in, you will find the button for **exercise4**
4. Click on it and a form will come up where you must provide
   - VM ip on which your application is running
   - Port number of the Server application

Example:

10.0.23.1

32465

5. Then click submit.
6. You will get the correct submission from server if everything is done correctly. (multiple productids will be tested while submission of the code).

Deadline for submission: Check the submission server

Thank you for your attention!