Introduction to Cloud Computing – Exercise 6

Scope: Troubleshooting, Kubernetes Dashboard, Data configuration, DaemonSet

Admission:

To perform the following exercises, you should have a virtual machine with minikube installed and running.

1. Troubleshooting

In this task you will learn how to deal with diagnosing what is happening in your deployment. Three techniques for checking the status of our deployment will be presented.

Create a test environment by running two deployments.

```
Kubectl create –f helloworld-deployment.yaml
```

And

```
Kubectl create –f helloworld-with-bad-pod.yaml
```

Then see the available deplyments and pods. You will see that one of the deployments is unavailable, and one of the deployments has an incorrect status.

The first technique to diagnose the state of our deployment is the *kubectl* describe command. To see the status of the bad-helloworld-deployment deployment, call it with the command *kubectl describe*

Is there an error in the information displayed?

Then execute *kubectl describe* for poda bad-helloworld-deployment.

Is there an error in the description? If so, which one?

The second way is to analyze logs. Invoke again all deployments and pods. Then call

```
Kubectl logs NAME_OF_POD
```

You should see all the logs present in this pod.

The third method is to call commands directly in the podium (going inside). To do this, call the command

```
Kubectl exec –it DEPLOYMENT_NAME /bin/bash
```

After calling the command, you will notice that we are "inside" the pod. In this way, you can see configuration files or, for example, verify that all processes are running.

In the case of a deployment consisting of multiple containers, to connect to a particular container, you need to add argument *–c*.

2. Kubernetes Dashboard

Read https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

What does Kubernetes Dashboard do?

You can install Kubernetes Dashboards using the instructions on the website, or you can use minikube add-ons to do so. Make sure minikube is running

```
Minikube status
```

To install minikube add-ons, first see the list of add-ons

```
Minikube addons list
```

The addition that interests us is *the dashboard.* To turn it on, call

```
Minikube addons enabled dashboard
```

The add-on should start, but you should get a message that one more add-on should be run. Run it.

After starting, call

```
Minikube addons list
```

Verify that all required add-ons are enabled. To run the add-on, call

```
Minikube dashboard
```

If the browser does not start on its own, go to the previous labs to get the port under which the add-on is running.

Familiarize yourself with the add-on through changes on the dashboard.

3. Data configuration

Apps require data to be passed to them during deployment. An example of this can be the url that we want to provide to our application or, for example, log level. Instead of entering these values in the code, we can use *config maps* to pass the values to the container.

See reader-deployment.yaml

Indicate where thevariables are passed to the container. Then call

```
Kubectl create –f reader-deployment.yaml
```

And see the active pods. When the pod will be in the running status, call

```
Kubectl logs POD_NAME
```

As you can see, the logs have *an error level.* Now let's create a configmap. Invoke command

```
Kubectl create configmap logger –-from-literal=log_level=debug
```

With this command stwwe used a new configuration mapthat passes a variable log_level with the value of debug to the container.

See how the yaml file changed (open *reader-configmap-deployment,yaml)*

What is the difference between the yaml reader-deployment.yaml and *reader-configmap-deployment.yaml* files ?

To view all configuration maps

```
Kubectl get configmaps
```

And to preview a specific map, call

```
Kubectl get configmap/MAP_NAME –o yaml
```

Check the logs on the new post running with reader-configmap-deployment.yaml, post them in the report

4. DaemonSet

In previous labs, we ran applications that were stateless, in this case we will run a regular busybox

https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/

See *the daemonset.yaml* file. You can see that the kind*:* parameter in this case is DaemonSet. For more information, please visit
https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/

What does DaemonSet mode mean?

To create a new DaemonSet callaj

```
Kubectl create –f daemonset.yaml
```

You can preview current deployments with the command.

```
Kubectl get daemonsets
```

You will notice that displays more information than in the case of pod, and there is a column that has information about *NodeSelector* but is empty. In the next part we will configure it. View all pods to see if our DaemonSet is working properly.

The next step is to add the label to our minikube node.

```
Kubectl label nodes/minikube infra=development –overwrite
```

To see the available labels fora noda call

```
Kubectl get nodes –-show-labels
```

You should see our added label. See the daemonset-infra-development.yaml file

Are there node selector settings in the configuration file? Point to a place.

Call

```
Kubectl create –f daemonset-infra-development.yaml
```

Display again the daemon sets. Is the node selector filled?

Read d ae monset-infra-prod.yaml and run this DaemonSet. After starting, display the available DaemonSety, does the newly created one work properly? Why yes/no?

Task – run dae monset-insta-prod

- Remove daemonset-insta-prod

- Add a new label to Node MiniKube

- edit the configuration file by changing node selector

- add a screenshot to the report