Lab5: Hosting a web service using Kubernetes cloud orchestration

Please **fill in the report** and submit the **pdf** to NYUClasses

# Objectives

* Understand the concept of container and the reason for cloud orchestration
* Get familiar with the Kubernetes and how to use the Kubernetes orchestration tool.
* Experience with hosting a popular cloud service using Kubernetes
* Use Kubernetes to deploy your WordPress server with a trivial loadbalancer
* Understand that you can manage Kubernetes services on Google Cloud Platform (Google Kubernetes Engine)

# Experiments Tasks

## Basics

1. Go through the Kubernetes introduction to get the general idea of Kubernetes

<https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>

## Install Tools

1. Install Docker:

<https://docs.docker.com/engine/install/ubuntu/>

1. Install kubectl:

<https://kubernetes.io/docs/tasks/tools/install-kubectl/>

1. Install Minikube:

<https://kubernetes.io/docs/tasks/tools/install-minikube/>

1. Install Helm:

<https://github.com/helm/helm>

## Host a Wordpress server on Kubernetes Cluster

* 1. Make sure the installation is complete
  2. Run a single-node Kubernetes cluster locally
  3. Deploy your WordPress server using helm  
     <https://hub.helm.sh/charts/bitnami/wordpress>
  4. Use kubectl commands to interact with Kubernetes cluster
  5. Verify your deployment of wordpress both on your browser and terminal
  6. Monitor the status of your Kubernetes cluster

1. **Reports**
   1. **General**

1. (a) What is a container from an operating systems perspective?   
 (b) Why are cgroups important to containers?

|  |
| --- |
| a. From operating systems perspective, the containing in the operating system are used to packet applications, and then to make sure they can run in the isolated environments.  b. Cgroups named control groups. As we know, the container can be used to packet application and make them independent. And according to achieve this. There is a linux control groups exits which called Cgroups to be used to processes the resources to be grouped, isolated and managed as a unit.  We also know that if there is a bad bug exits, then the whole mechanism and process may be broken. However, Cgroups can be used to limit and isolate the CPU, memory, disk I/O and network usage of one or more processes. So that if there exits bug, it will not break all the process or mechanism. It only break the process that are bundled in a group. Which means that the cgroup can bundle processes together with the same set of limit. In another word, there will be lots of group exits in a system. They can also be hierarchial. The subgroup will inherit the limits if its parents.  There are four features for cgroup:  Resource limiting: there is some kind of limitation set to groups. The limitation can be memory limit, location limit and so on. Which is used to limited the group not exceed the limitation.  Prioritization: one or more groups may be use fewer or more the throughput of CPUs or disk I/O  Accounting: it can be used to monitor and measure the usage of a group resources.  Control: group of processes can be controlled, which means the group of processes can be used to stopped and restarted.  Because Cgroup is used to solve the main feature of the container, and it can prevent all the system interrupted. So Cgroup is important to containers. |

2. In Docker what does the ENTRYPOINT command allow you to do?

|  |
| --- |
| ENTRYPOINT will run the container as a executable, also it allows to specify a command with parameter. Which means that when we run the container the image can also be executed without specifying. Sometimes we want to run some images when we run the Docker. However we cannot use docker run <command> to run this image. If we did this and this image is not been used by ENTRYPOINT, then it will return ERROR. So we have to use ENTRYPOINT in the Dockerfile to mention the image. So we can auto using the image when we run the Docker. |

3. (a) In Kubernetes, what is a pod and why is it useful?   
(b) What is a label and why is it useful?

|  |
| --- |
| a. Pod in Kubernetes is the smallest deployable units which can be created and managed. A pod is a group of one or more containers. They will have shared storage or network resources. The contents of Pod can be co-located and co-scheduled, and these contents can run in a shared context.  Pod is useful is because Kubernetes works with Pods rather than containers. Kubernetes uses Pods to run codes and images in clusters. So, the containers in the pod can run on the same machine. Because we mentioned pod is a group of one or more containers. And they share the network resources in the same pod. This can communicate between containers via the file system or over the network. So it will allow multi-container pod design patterns. And this pattern can separate the system as isolated. It can also keep the code free from the log collection, external proxy service and metrics.  b. Label is key or value pairs which can be attached to the objects like pods. If there are some attribute can be related to the users, then the labels can be used to specify these attribute of objects.  For any given object, the key must be unique, so we can use specific keys to locate the object which can be meaningful to the users.   * + - 1. Also labels can be used to organize and select subsets of objects. Which means labels can let us to organize Kubernetes workloads which are in the clusters.       2. Also it can let us selectively filter the kubectl output which is just the object I want to use.       3. Labels can be used to attach objects at creation time and it can be modified at any given time.       4. Labels can also enable us to understand the API objects’ layer and hierarchies.       5. For the Open Policy Agent, we can use labels to make it more flexible.   So all of these are the reason why label is useful. |

1. (a) In Kubernetes, what is a service and why is it useful?  
   (b) What is a ReplicaSet and why is it useful?

|  |
| --- |
| a. Service in Kubernetes is an abstract way to an application which is running on a set of Pods as a network service.  Service is useful is because service is a group of pods and it can run on the cluster. Also services are cheap. Also in the cluster we can have as many services as we want. Also service has an important feature which can be standardized across the cluster such as the load-balancing, service discovery between applications, and also to support zero-downtime application deployments. As I mentioned services is a group of pods, so service can define the pods which will process data for the service. We can say this process is created by the pod label query. The pod label query is in the services. It can match pods which are created by one or more replication controllers. So this is why service is usful for the Kubernetes, because it has a tight connection with pods.  b. ReplicaSet：We need the ReplicaSet to maintain the stable set of replica Pods running at any given time. We can say that the ReplicaSet can guarantee the specified number of Pods running in the process.  The ReplicaSet is useful is because, if we do not use ReplicaSet, we need to create multiple manifests for the number of pods. We need to do lots of works for that. Also, for the ReplicaSet, we have a Label Selector. We know that labels can be used to specify the attributes of objects. And those objects can be meaningful and useful to users. Accoridng to this way, if we use ReplicaSet, the we do not need to do lots of work for deploy replicas of single application. It does not need to change the core system. So ReplicaSet can be more convenient to control the pods. |

1. What is the difference between a pod and a node?

|  |
| --- |
| Pod: pod is used for managing the container. For every pods, there should be one or more containers in it. And all the containers in the same pods will have tight relationship. They will share the resources in the same pod. We can also say the pod is the atomic units on the Kubernetes platforms. Each pod need to be scheduled on Nodes when we create a Deployment on Kubernetes. When the Node Failure. The Pods in it will be scheduled on other available Nodes in the cluster.  Node: we should know that Pod always runs on a Node. So we can say that a Node can be virtual machine or physical machine. However, for Node, there should be Master manages other Nodes. On the Kubernetes, the Master Node automatically schedule pods to the Nodes in cluster.  So we can say that the difference between a pad and a node can be:  a pod can have one or more containers in it. However for Node, there can be one or more pods in it.  Pods does not have master to control it, but for node it have master node to control it.  Pods can be atomic units on Kubernetes, but for Node, it can be larger than pod.  Containers in the same node can share the resources, but for node there is no resources for it to be shared with pods on it. |

**3.2 Verify your installation**

1. Screenshots of running docker successfully

sudo docker run hello-world

|  |
| --- |
|  |

1. Screenshots showing that you can start minikube successfully

minikube start --driver=docker

minikube status

|  |
| --- |
|  |

1. Screenshots of verifying kubectl configuration

kubectl cluster-info

|  |
| --- |
|  |

**3.3 Deploy your WordPress**

1. Explain the commands

|  |
| --- |
| (a)  kubectl get pods: list the name and all details information for the all pods in plain-text output format. kubectl get services: list the name and all details information for the all services in plain-text output format. (b) kubectl describe deployment: we can get deployment detail information. We can see the pods are brought up and removed in which order. kubectl logs <pod-name>: pod-name is the name of the pod we want to see. So this command is to show the logs for pod which has the pod-name from a container in a pod. |

1. Screenshots of your terminals showing that your pod is up (Hint: use a command in 9a)

|  |
| --- |
|  |

1. Screenshots of your terminals showing your running service along with the external IP address that can be used to access your pod. (Hint: use commands in 9a)

|  |
| --- |
|  |

1. (a)Screenshots of your browser running your wordpress. What is your wordpress URL?   
   (b)Screenshots of login to your wordpress as admin

|  |
| --- |
| * + - * 1. <http://﻿10.100.70.105/> |

1. What happens after you manually delete a pod using the following command?

kubectl delete pods *<your-pod-name>*

|  |
| --- |
| When we delete a pods. Then there will be new pod to do the same thing as the previous one did.  I check my pods with ﻿kubectl get pods then I get    When I delete the pods with kubectl delete pods ﻿my-release-wordpress-7b8666479f-58x4j  Then the terminal shows    Also the dashboard will change to    Then use kubectl get pods the terminal shows    Then I found the name of the pod has been changed.  Then I will delete another pod with kubectl delete pods my-release-mariadb-0    And the dash board also changed    However, when I type kubectl get pods the terminal shows    The name of this pod which has been deleted does not changed, but the age is changed which means that this pod is the new pod when I delete the previous one.  And After I delete either of these two pods, the dash board will turn to green in the end, because the deleted pods will be replaced by another pod to do the same thing as it did before. All of this is because the ReplicaSet. This is the main pod which will not change. So if there Is one sub pod has been deleted. The main pod will allocate another sub pod to replace the deleted one. |

1. Screenshots of your Kubernetes Dashboard. What is the health status of your workloads?

|  |
| --- |
| First stage, nothing changed on the dashboard:    After delete the *my-release-wordpress-7b8666479f-58x4j pod*:    When the old pod *my-release-wordpress-7b8666479f-58x4j* is deleted and the new pod ﻿my-*release-wordpress-7b8666479f-m2ptj* has been added in , and waiting for running.    When the new pod ﻿*my-release-wordpress-7b8666479f-m2ptj* has finished running:    When I delete the pod named ﻿*my-release-mariadb-0*    When the new pod *my-release-mariadb-0* has been added and waiting for finishing running:  Here, even the name is the same as previous one, but the age can be different.      When the pod *my-release-mariadb-0* has been finished running:    So according to delete either of these two pods, we can find whatever we did on these two pods. Finally the status of the dashboard will be recovery. This is because when we delete s pod, there will be another node to be created to replace it and wait for finishing running. After the running is finished, then the process will be recovery. So the dashboard will be green again. |

1. Challenges you’ve encountered while doing this experiment and explain how you manage to solve them. If you do not experience any problem, simply say no problem.

|  |
| --- |
| First when I run the tunnel it always tell me there is no space. So I create another virtual machine with more space to solve this problem.  Second, it can always show no connection when I type the external IP address on the web browser. Then I use minikube dashboard to see the status and wait for all those fore sets become green. Then I type the IP in the web bar. So I can achieve the website. |

**We have zero tolerance to forged or fabricated data!!** A single piece of forged/fabricated data would bring the total score down to zero.