**KUBERNETES NOTES:**

**Document - Link to Code Repository**

**1. Official CKA Exam Blueprint:** https://github.com/cncf/curriculum

**2. Our CKA GitHub Repository Link:**

<https://github.com/zealvora/certified-kubernetes-administrator>

Domain 1: Core Concepts

<https://docs.google.com/document/d/1XpXl9Xm0_4sUtf3gKUA0TFp4qwCvXnu8YO9_0B4OUZo/edit?usp=sharing>

**Notes -Workloads & Scheduling**

https://docs.google.com/document/d/1Uoebi-g-Arua9myZCzK3Kx-rpJsPZ9NK9oyWG8MgdxY/edit?usp=sharing

**Notes - Services and Networking**

https://docs.google.com/document/d/1QHNHlvPTJSTIk\_sLI-c\_qm1n4CbG5RufYJmtw1vycbo/edit?usp=sharing

**Notes - Security**

<https://docs.google.com/document/d/1bfWCFQgJ6IhOZbSaliMRaMDIW8c3DX0KyVz96rRtv4E/edit?usp=sharing>

**Notes - Storage**

<https://docs.google.com/document/d/1BWhLJSJUXRZWpu2YkhXJsY_fF0DN-4OzHDUD51p72eE/edit?usp=sharing>

**Document - MSSP**

**Digital Ocean Credit Referral Code:**

https://m.do.co/c/74dcb0137794

**Document - kubectl**

https://kubernetes.io/docs/tasks/tools/

download kubectl in your linux server

login to server as root

chmod +x ./kubectl

mkdir .kube

cd ~/.kube and create a file config and add the downloaded config file from the master k8s node to connect then cd .. to root

cp kubectl /urs/local/bin (this will run the kubectl in any directory)

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

chmod +x ./kubectl

mkdir .kube

cd ~/.kube/

vim config

cd ..

cp kubectl /usr/local/bin/

kubectl get nodes

**Docker Installation:**

yum -y install docker

systemctl start docker

systemctl enable docker

K8S container = pod

kubectl run mywebserver --image=nginx (to create a pod (container))

kubectl exec -it mywebserver -- bash (to get into your container)

kubectl delete pod mywebserver (to delete pod)

\*\*\*\*\*\*\*where kubectl\*\*\*\*\*\*\*\*\* to locate it in your system

**Document - API Resources for writing YAML files**

**Documentation:**

<https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19/>

<https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#metadata>

**newpod.yaml file**

1. apiVersion: v1
2. kind: Pod
3. metadata:
4. name: nginxwebserver
5. spec:
6. containers:
7. - image: nginx
8. name: democontainer

Creating Kubernetes Clusters

STEPS

1. Provision three servers 1 with medium as Control and two t2.micro as worker nodes
2. Vim /etc/hosts and add all the three servers private IP address and their names
3. Enable the overlay and br\_netfilter with the { cat << EOF | sudo tee /etc/modules-load.d/containerd.conf \ overlay \ br\_netfilter \
4. Run sudo modprobe overlay and sudo modprobe nr\_netfilter (to enable them)

Kubernetes Fundamentals

Resources:

In this lesson, we're going to be looking at Kubernetes resources. So, here's just a quick breakdown of what we're going to be talking about. First, we'll answer the simple question:

What are resources? We'll focus in on the Pods resource. We'll talk about the concept of resource types, as well as Kubernetes objects, and we'll do a quick hands-on demonstration

where we'll explore this concept in our Kubernetes cluster. So, let's start by answering the question:

What are resources?

Well, resources are simply objects of a certain type in the Kubernetes API, and we interact with our cluster by creating and modifying these objects. So if you want to run a container in your Kubernetes cluster, you're going to be creating some type of object via the Kubernetes API, and that's called a resource. If you want to make a change to your application running in the cluster or some form of configuration for the application, you're going to be modifying different objects via the Kubernetes API. So, that's what we're talking about when we talk about resources. Arguably, the most important type of resource in Kubernetes is the Pods resource. A Pod is the most basic resource in Kubernetes. So, we have our Pods resource there, and the Pod represents a group of one or more containers. So, essentially, each Pod object defines at least one container, but it can possibly be more than one. And if you want to run some containers in Kubernetes, you can do that by creating a Pod object. So, here we have our Pods resource, which is a list of different Pod objects, and we have each one of our Pod objects there. And when we create a Pod object that defines some containers, Kubernetes will go ahead and actually run those containers. **Here's just a quick study tip for the KCNA exam**. If you want to run a task before a Pod's main container starts up, you can use a special type of container called an init container. And that's just part of the configuration of a Pod. Just a little tip of something to pay attention to for the purposes of the exam. Now, when we talk about Kubernetes resources, there are multiple different types of resources that we can create in our cluster. So, all objects or resources have a resource type. Sometimes those terms, object and resource, are used a little bit interchangeably. Technically, they mean different things, but in practice, they kind of refer to the same thing when you're actually interacting with Kubernetes. But they all have a resource type, and the object's resource type determines the specific Kubernetes functionality that that object controls. So if I create a Pod, that's going to spin up some containers. If I create a deployment, that does something else, that's a different resource type.

If I create a service, that has, yet again, some additional functionality. And we're going to be talking in a little more detail about Pods and deployments and services and what each of those types does as we proceed through the course. But right now, it's just important to understand that each resource type is going to control certain functionality within the cluster. Another study tip, just something to be aware of for the purposes of the exam: You can use a particular command to list all of the available resource types in the cluster. And that is the kubectl api-resources command. So if you just want to see all the different options, all the different resource types that are available, just use that command to list them. Another important command to keep mind is this command that you can use to get documentation about a resource. And that is kubectl explain. That command will give you all of the available fields for a resource and documentation on what it does and all the configuration options for that resource. Now, let's talk about Kubernetes objects. A Kubernetes object is a piece of data that represents the state of a cluster. So, we have our Pods resource type there, and we have multiple Pods. And each one of those Pods is a Kubernetes object. When we create or modify an object, Kubernetes is going to work constantly to implement the desired state expressed in the object data. And that data also tells us about the current actual state of the object. So, you can look at the data within a Pod object to gain information about the state of those containers, or, in many cases, even make changes to what's going on in the cluster. Another study tip is you can define your own custom resource types using a CustomResourceDefinition. So if you want to extend the Kubernetes API and create your own custom resource type, there's a CustomResourceDefinition which is itself a resource. You can create a CustomResourceDefinition object. That'll create your own custom resource type. Now, let's go ahead and dive into our cluster and just work with these Kubernetes resources and see what it actually looks like in practice. I'm logged into my Kubernetes control plane node here, and I'm going to start by just running this kubectl api-resources command. Here, I can see a list of all the different available resources. There's quite a few resource types available. So, that just gives me an idea of all the resource types that are available in the cluster. And I also want to check out that kubectl explain command, and I'll explain the Pod resource. Now in kubectl, I can do pod or pods with an s. They'll both do the same thing. So, I'm just going to do kubectl explain pod. And this gives me some documentation on that Pod's resource. And I can see all the available fields for the Pod object. So now, let's go ahead and interact with a real Kubernetes object by creating a Pod. And that's going to, of course, run a container within our cluster. So, there are multiple different ways to do this, but I'm going to do it by first creating a YAML file, and I'm going to call it my-pod.yml. And inside this YAML file, I can basically define my Pod object. So, I'll just paste in this information so you don't have to watch me type everything out, but this is just a standard YAML format for that Pod resource type. The kind here indicates the resource type I want to create. So, that's going to be a Pod. Here is the name of my Pod. It's just going to be my-pod. Here is the Pod specification. This is the configuration for my Pod object. And I'm going to be running one container here, and it's going to be called nginx, and it's going to use the nginx:stabl image. So, I'll go ahead and just save it and exit that file. And then I can use the command kubectl apply to go ahead and basically apply configuration based on that file. So, that's going to create that Pod object. I'll just pass in the file name there with -f, and it says my Pod was created. So, I've just created a Kubernetes object. Of course, I can also view my object. I can do kubectl get pod or get pods. Both of those commands will do the same thing. And I can see my Pod there. And it is in the running status. That means it's up and running. It seems to be working as expected. So, I'm now running a container simply by creating that Pod object. And, of course, I can delete my object as well. So, I'll do kubectl delete pod my-pod. And once that is deleted, that will go ahead and destroy that container. So by interacting with these Kubernetes objects, I'm interacting with what my cluster is actually doing in terms of running containers. Now, before I end the lesson, I want to go through a few quick study tips, just things to focus on as you're studying for the KCNA exam. First, a resource is an object of a certain type in the Kubernetes API. You can list all available resource types in a cluster with the command kubectl api-resources. You can get documentation for a resource type using kubectl explain. You can use an init container to run a task before a Pod's main container starts up.

And you can create your own custom resource types using a CustomResourceDefinition. So, that was just a quick look at Kubernetes resources. We talked about what resources are, we took a closer look at the Pods resource, we talked about resource types and the concept of Kubernetes objects, and we did a quick hands-on demonstration. That's all for this lesson.

References:

<https://github.com/ACloudGuru/content-kcna/blob/main/S02_L02_Introducing%20Kubernetes%20Resources.md>

K8s API Terminology

<https://kubernetes.io/docs/reference/using-api/api-concepts/#standard-api-terminology>

Understanding K8s ogjects

<https://kubernetes.io/docs/concepts/overview/working-with-objects/kubernetes-objects/>

Pods

<https://kubernetes.io/docs/concepts/workloads/pods/>

Init Containers

<https://kubernetes.io/docs/concepts/workloads/pods/init-containers/>

Kubectl Cheat sheat

<https://kubernetes.io/docs/reference/kubectl/cheatsheet/>