CKA Crash Course

Intro

Your Instructor

- Sander van Vugt
- Author of many Linux, Ansible and Kubernetes related courses on this platform
- mail@sandervanvugt.nl
- Founder of the Living Open Source Foundation: livingopensource.net

The audience

- Some Kubernetes knowledge is required
- Ideally, would have attended my CKAD class or done the CKAD exam
- Good knowledge of Linux

This course

- This is not an introduction course to Kubernetes
- To successfully attend this course and prepare for the CKA exam, you should have good basic knowledge of Kubernetes already
- To acquire this knowledge, use one of the following live courses
  - Managing Containers on Linux
  - Kubernetes in 4 Hours
  - Certified Kubernetes Application Developer (CKAD) Crash Course

Poll Question 1

Rate your own Kubernetes knowledge/experience

- Just beginning
- Unstructured working knowledge
- Knowledge at CKAD level
- Knowledge at CKA level

Poll Question 2

Have you attended my CKAD class?

- yes
- no

Poll Question 3

Rate your own expertise regarding Kubernetes

- none
- poor
- average
- above average
- very confident

Required Lab Environment

- 4 virtual machines
- 1 control node
  - 2 vCPUs
  - 2 GB RAM
  - 20 GB disk
  - No swap
- 3 worker nodes
  - 1 vCPU
  - 1 GB RAM
  - 20 GB disk
  - No swap
- Disable firewalld

Course resources

- https://github.com/sandervanvugt/cka

CKA Crash Course

Understanding this Course Philosophy

How NOT to learn for CKA

- You won't learn by memorizing bullet points on slides
- You won't learn by copy-pasting lines of code in a demo environment
- You won't learn by getting all the exam questions and memorize the answers

How to Learn for CKA

- Understand what you're doing and what the different objects are all about
- Practice to become fast in it
- Use available resources to find information in case you don't know it

How this is applied in this course

- This course is scenario based
- We work with sample questions, which could look like the questions that you'll see on the exam
- I'll use my virtual whiteboard to explain concepts behind the objects you have to work with
- You'll use available resources to figure out how to do it – just the way you would do it on the exam
- You will have 5 minutes for each of the scenarios, then we'll discuss
- And by the end of this course you've build understanding and skills to get you successfully through the exam
- When needed, a slide with a procedure description is added

CKA Crash Course

Generic Exam Tips

Before you Sign up

- See https://training.linuxfoundation.org/certification/certified-kubernetes-administrator-cka/ for the current objectives
- Notice that the exam follows the latest Kubernetes release, so a new update is published every 3 months!
- With the exam voucher, one free retake is included

Documentation is available!

While doing the exam

- One browser tab with access to docs.kubernetes.io is allowed
- Use **kubectl explain** for any additional details
- Notice that some documentation is outdated / inaccurate

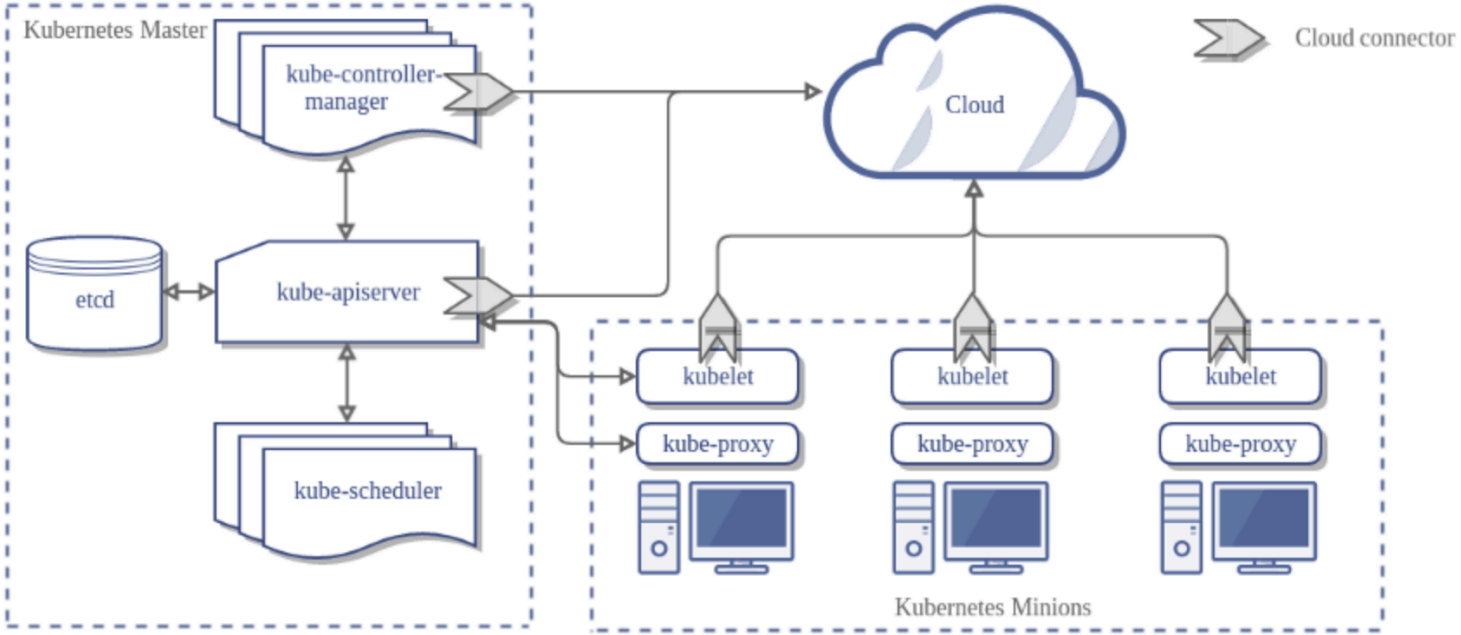
CKA Crash Course

1. Creating a Cluster

1. Creating a Cluster

1.1 Understanding Kubernetes Architecture

Architecture Overview



Understanding the Master Node

- kube-apiserver: front-end of the cluster that services REST operations and connects to the etcd database
- kube-scheduler: schedules pods on specific nodes based on labels, taints and tolerations set for the pods
- etcd: a B+tree key-value store that keeps the current cluster state. Organized with master and following instances of the database
- kube-controller-manager: manages current state of the cluster
- cloud-controller-manager: interacts with outside cloud managers
- Different optional add-ons
  - DNS
  - Dashboard
  - Cluster level resource monitoring
  - Cluster level logging

Understanding the Worker Nodes

- kubelet: passes requests to the container engine to ensure that Pods are available
- kube-proxy: runs on every node and uses iptables to provide an interface to connect to Kubernetes components
- container runtime: takes care of actually running the containers
- supervisord: monitors and guarantees the availability of the kubelet and docker processes
- network agent: implements a software defined networking solution, such as weave
- logging: the CNCF project Fluentd is used for unified logging in the cluster. A fluentd agent must be installed on the K8s nodes

1. Creating a Cluster

1.2 Understanding node Networking Requirements

Cluster Node Requirements

- To set up a Kubernetes on-premise cluster, **kubeadm** is used
- You'll need a minimum of one node, for the setup used in this course you'll need 4 nodes
- Use Centos 7.x or Ubuntu for best support
- The control node needs 2 CPUs
- All nodes need 1 GiB for a test environment, a minimum of 2 GiB or (much) more is required for production environments
- Install without using swap space
- Shut off the firewall

Software Installation

- Before starting installation, you need a container runtime
- Different runtimes are supported, in this course we'll use **docker**
- To make installation easy, use **git clone** https://github.com/sandervanvugt/cka which provides 2 scripts
  - **setup-docker.sh** installs the Docker container runtime
  - **setup-kubetools.sh** installs the Kubernetes tools
- As root, run these scripts on all nodes
- On all nodes, use **systemctl enable --now docker** before continuing

Installing a Pod Network Add-on

- A network add-on must be installed for pods to communicate
- CNI is the Container Network Interface. It works with add-ons to implement networking
- Different project exist for offering Kubernetes network support, which requires support for the following types of networking:
  - container-to-container
  - pod-to-pod
  - pod-to-service
  - external-to-service
- Look for an add-on that supports *network-policy* as well as *RBAC* (both covered later in this course)
- In this course, we'll use the Weave add-on

Common Pod Networking Plugins

- Flannel: a layer 3 IPv4 network between cluster nodes. Can use several backend mechanisms such as VXLAN
- Weave: a common add-on for a CNI-enabled Kubernetes cluster
- Calico: a layer 3 network solution that uses IP encapsulation and is used in Kubernetes, OpenStack, OpenShift, Docker and others
- AWS VPC

Exercise 1

- Use **kubeadm** to create a cluster. control.example.com is set up as cluster controller node, worker{1..3} are set up as worker nodes
- The task is completed if **kubectl get nodes** shows all nodes in a ready state

CKA Crash Course

2. Creating a Pod

Exercise 1

- Create a Pod that runs the latest version of the alpine image. This pod should be configured to run the sleep 3600 command repeatedly and it should be created in the mynamespace namespace

CKA Crash Course

3. Creating a Multi-container Pod

Exercise 3

- Configure a Pod that runs containers based on the 3 following images
  - redis
  - nginx
  - busybox

CKA Crash Course

4. Creating a Pod with an init Container

Exercise 4

- Configure a Pod that runs 2 containers. The first container should create the file /data/runfile.txt. The second container should only start once this file has been created. The second container should run the "sleep 10000" command as its task

CKA Crash Course

5. Creating a Deployment

Exercise 5

- Create a Deployment with the name nginx-ex5. The deployment should start 5 replicas and use the nginx image

CKA Crash Course

6. Creating a Manifest File

Exercise 6

- Create a Deployment using a Manifest File. The deployment should run a Busybox container that executes the "sleep 10000" command, and it should start 3 replicas. Run the manifest file to create the desired configuration. After running it, make sure that all API objects created by the deployment are removed again.

CKA Crash Course

7. Creating a Pod with Persistent Volume

Exercise 7

- Create a Pod that is based on the nginx image. This pod should use local storage non-persistent way.

CKA Crash Course

8. Configure a ConfigMap

Understanding ConfigMap

- ConfigMaps can be used to separate dynamic data from static data in a Pod
- They are not encoded or encrypted
- They can be used in three different ways:
  - Make variables available within a Pod
  - Provide command line arguments
  - Mount them on the location where the application expects to find a configuration file
- Secrets are encoded ConfigMaps which can be used to store sensitive data
- ConfigMaps must be created before the pods that are using them

Understanding ConfigMap Sources

- ConfigMaps can be created from different sources
  - Directories: uses multiple files in a directory
  - Files: puts the contents of a file in the ConfigMap
  - Literal Values: useful to provide variables and command arguments that are to be used by a Pod
- Since Kubernetes 1.14 the kustomization.yaml generator can be used
  - This used the configMapGenerator API object to generate the ConfigMap based on input data in the kustomization.yaml file

Procedure Overview

- Start by defining the ConfigMap and create it
  - Consider the different sources that can be used for ConfigMaps
  - **kubectl create cm myconf --from-file=my.conf**
  - **kubectl create cm variables --from-env-file=variables**
  - **kubectl create cm special --from-literal=VAR3=planets --from-literal=VAR4=moon**
  - Verify creation, using **kubectl describe cm <cmname>**
- Use **--from-file** to put the contents of a config file in the configmap
- Use **--from-env-file** to define variables
- Use **--from-literal** to define variables or command line arguments

Procedure Overview - 2

- To include *variables* from a ConfigMap:

```
envFrom:
  - configMapRef:
      name: ConfigMapName
```

- To include *config files* from a ConfigMap:

```
volumes:
  - configMap:
      name: ConfigMapName
      items:
      - key: my-custom.conf
        path: default.conf
```

Demo: Creating a ConfigMap from a File

- Show contents of variables (in github)
- Create the ConfigMap: **kubectl create cm variables --from-env-file=variables**
- Verify creation: **kubectl describe cm variables**
- Create a Pod: **kubectl create –f cm-test-pod1.yml**
- Check that the variables are available: **kubectl logs po/test** (or whatever the name is)

Demo: Configuring a ConfigMap from a Literal

- **kubectl create cm morevars --from-literal=VAR3=planets --from-literal=VAR4=moon**
- **kubectl get cm/morevars**

Demo: Using ConfigMaps for ConfigFiles

- Create the ConfigMap: **kubectl create cm nginx-cm --from-file nginx-custom-config.conf**
- Check the contents of the ConfigMap: **kubectl get configmap/nginx-cm –o yaml**
- Next, create the Pod: **kubectl create -f nginx-cm.yml**
- Check the config file: **kubectl exec -it nginx-cm /bin/bash**
- **cat /etc/nginx/conf.d/default.conf**

Exercise 14

- Create a ConfigMap that defines the variable myuser=mypassword. Create a Pod that runs Alpine, and uses this variable from the configMap

CKA Crash Course

9. Creating a Persistent Volume

Exercise 9

- Create a Persistent Volume that uses local host storage. This PV should be accessible from all name spaces. Run a Pod with the name pv-pod that uses this persistent volume from the "myvol" namespace

10. Running a Pod in a Namespace

exercise 10

- In the run-once namespace, run a Pod with the name xxazz-pod, using the alpine image and the command sleep 3600. Create the namespace if needed. Ensure that the task in the Pod runs once, and after running it once, the Pod stops

CKA Crash Course

11. Create and Upgrade a Deployment

Exercise 11

- Create a Deployment that runs Nginx, based on the 1.14 version. After creating it, enable recording, and perform a rolling upgrade to upgrade to the latest version of Nginx. After successfully performing the upgrade, undo the upgrade again.

CKA Crash Course

12. Find Pods by Label

Exercise 12

- Find all Pods that have the label app set to working. Use kubectl features to sort this list on host name, and write the resulting output to the file /var/exam/sortlist.txt

CKA Crash Course

13. Expose a Pod

Exercise 13

- Expose the Nginx Pod such that it can be reached by external users. To expose it, use a cluster IP address

CKA Crash Course

14. Creating and Using a Secret

Exercise 8

- Create a secret that contains the variable definition userpw=s3cr3tPW
- Configure one Pod to use that secret by mounting it in the /etc/secret directory. Configure another Pod that uses that secret in a way that shows userpw is scrambled when the Pod YAML code is dumped

CKA Crash Course

15. Configure a Daemonset

Exercise 15

- Run a Daemonset, that ensures your Nginx application runs on every host. The name of the application should be nginx-ds, and it should run the latest version of the nginx image

16. Create a HostPath PV with max Storage of 2 GiB

Create a PV

- Create a PV that is accessible by multiple namespaces at the same time. It should allocate 2GiB of disk storage on the local host

17. Enable a Node to Run Pods Again

Exercise 17

- Analyze the nodes in the local cluster. One of the nodes is not available to run Pods. Enable it to run Pods again

18. Mark a Node as Unavailable

Exercise 18

- In your cluster, mark the node worker2 as unavailable. Ensure that all Pods are moved away from the local node and are started again somewhere else

19. Put a Node in Maintenance Mode

Exercise 19

- Put a node in maintenance mode, such that no new pods will be scheduled on it

20. Finding the Pod with the Highest CPU Load

Exercise 20

- Find the Pod with the highest CPU load and write its name to the file /var/exam/cpu-pods.txt

Configuring Monitoring and **kubectl top**

- **git clone https://github.com/kubernetes-incubator/metrics-server.git**
- **kubectl create –f metrics-server/deploy/1.8+/**
- **kubectl –n kube-system get pods** # look for metrics-server
- **kubectl –n kube-system edit deployment metrics-server**
  - In spec.template.spec.containers.args, add
    - --kubelet-insecure-tls
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
  - Under dnsPolicy, add a line that reads **hostNetwork: true**
- **kubectl –n kube-system logs metrics-server<TAB>** should show "Generating self-signed cert" and "Serving securely on [::]443

Running **kubectl top**

- **kubectl top pods --all-namespaces** will show most active Pods
- Give it time to collect the metrics – should work in about 60 seconds

CKA Crash Course

21. Backing up the etcd Database

Exercise 21

- Create a backup of the etcd database. API version 3 is used for the current database. Write the backup to /var/exam/etcd-backup

CKA Crash Course

22. Pod and Service DNS Connectivity

Testing DNS - 1

1. Create busybox.yaml

```
apiVersion: v1
kind: Pod
metadata:
 name: busybox2
 namespace: default
spec:
 containers:
 - image: busybox
   name: busy
   command:
     - sleep
     - "3600"
```

2. Create the pod, using **kubectl create -f busybox.yaml**
3. Use **kubectl get svc** to validate the name of any exposed service
4. Use **kubectl exec -ti busybox2 -- nslookup kubernetes**
5. You'll see the IP address being resolved, thus providing proof that DNS is working correctly
6. Note that services can be resolved through DNS, Pods cannot by default

Testing DNS - 2

- **kubectl create –f pod-and-service-dns.yaml**
- **kubectl exec –it busybox2 --nslookup default-subdomain**
- **kubectl exec –it busybox2 --nslookup busybox-1**

Understanding DNS

- CoreDNS is the default DNS service in Kubernetes
- Its IP address is exposed by a service kube-dns that lives in the kube-system namespace
- This service IP address should match the contents of /etc/resolv.conf on the Pod nodes
- When starting a container, the Kubelet passes DNS to it, using **--cluster-dns=<dns-service-ip>**
- Also, the kubelet is configured with its local DNS domain, using **–cluster-domain=<default-local-domain>**

Analyzing DNS - 1

- Check Pod DNS resolver: **kubectl exec podname -- cat /etc/resolv.conf**
  - The nameserver should match the IP address of the core-DNS service
  - You may have a search path, containing **search default.svc.cluster.local svc.cluster.local cluster.local …** (you should see this path being queried also)
- Use **kubectl get pods --namespace=kube-system –l k8s-app=kube-dns** to verify the CoreDNS Pod is up
- If pods fail: **kubectl –n kube-system describe pods core-dns-nnn**
- Check logs in CoreDNS Pods: **for p in $(kubectl get pods --namespace=kube-system –l k8s-app=kube-dns –o name); do kubectl logs --namespace=kube-system $p; done**

Analyzing DNS – 2

- Verify the DNS service is up: **kubectl get svc –n kube-system**
- Verify the endpoints are exposed: **kubectl get ep kube-dns –n kube-system**

Troubleshooting DNS

- Disable all firewalling on all nodes**: iptables –F && iptables –t nat -F && iptables –t mangle –F && iptables -X**
- Restart Dockerd**: systemctl restart docker**
- Remove core-dns Pods: **kubectl delete pod –n kube-system –l k8s-app=kube-dns**, they are automatically recreated
- Remove your network plugin pod and re-install
- Replace network plugin: Calico is doing better than Weave

Exercise 22

- Start a Pod that runs busybox image. Use the name busy22 for this Pod. Expose this Pod on a cluster IP address. Configure the Pod and Service such that DNS name resolution is possible, and use the **nslookup** command to look up the names. Write the output of the DNS lookup command to the file /var/exam/dnsnames.txt

23. Configuring a Node to Autostart a Pod

Exercise 23

- Configure your node worker3 to automatically start a Pod that runs an nginx web server, using the name auto-web. Put the manifest file in /etc/kubernetes/manifests

CKA Crash Course

24. Troubleshooting Cluster Connectivity

Troubleshooting the Cluster

- Restoring .kube/config
  - **sudo cp –I /etc/kubernetes/admin.conf $HOME/.kube/config**
  - **sudo chown $(id –u):$(id –g) $HOME/.kube/config**
- Check the kubelet service: **systemctl status kubelet**

Exercise 24

- You currently are incapable of accessing the cluster, getting a service unavailable error message. Troubleshoot this problem

CKA Crash Course

Where to go from here

Where to go from here

- Consult training.linuxfoundation.org/certification/certified-Kubernetes-administrator-cka/
- Hopefully you're ready for the exam now! Time to order your voucher:
  - https://www.itgilde-academy.com/training-category/certification/
  - https://linuxfoundation.org