Kubernetes in 4 Hours

Sander van Vugt

Pearson

About your instructor

- This course is presented by Sander van Vugt
  - mail@sandervanvugt.com
  - www.sandervanvugt.com
- Course resources are available at https://github.com/sandervanvugt/kubernetes

Agenda

- Understanding Kubernetes
- Kubernetes Installation and Configuration
- Using Kubernetes to Manage Containers

Expectations

- This class is for people new to Kubernetes
- I'll teach you how to get started and deploy applications on Kubernetes
- Don't expect much information about advanced topics

Lab instructions

- This course contains lab instructions which you can follow along or run later at your own convenience
- You need a running Kubernetes installation to follow aong
- To get *the same experience*, use the following recommended minimal settings
    - Linux host (I'm using Ubuntu LTS 20.04) with hardware virtualization enabled
    - At least 8GB of RAM (more is always better)
    - At least 40 GB of available disk space (more is always better)
- While running labs in this course, do NOT run another hypervisor on the host at the same time

Alternative Lab Solutions

- Docker Desktop: enable Kubernetes support from the main dashboard window
- Any public cloud based solution (I like Google GCE, but others work also)
- Anything else that runs Kubernetes

Question Handling Policy

- If you want me to answer your question, post in Q&A please. I will answer it if it meets the requirements, but you may have to wait a bit
- Questions about personal issues "when I do this, I get error message xyz" only in group chat please
- If you just missed something, please watch the recording. I cannot repeat topics I just explained as that wastes valuable time
- Questions about topics that you should have known in group chat only please
- Off-topic questions are welcome in the Q&A at the end of this course

Poll Question 1

- How would you rate your own Linux knowledge and experience?
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5

Poll question 2

- How would you rate your own knowledge about containers
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5

Poll question 3

- How would you rate your own Kubernetes knowledge and experience?
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5

Poll question 4

- Where are you from?
  - India
  - Asia (other countries)
  - Africa
  - North or Central America
  - South America
  - Europe
  - Australia / Pacific
  - Netherlands

Kubernetes in 4 Hours

What is Kubernetes?

What is Kubernetes?

- Kubernetes is rapidly evolving open-source software, written in Go, for automating deployment, scaling, and management (orchestration) of containerized applications
  - See kubernetes.io for more details
  - A new major release every 3 months!
- It is about running multiple connected containers across different hosts, where continuity of service is guaranteed
- The solution is based on technology that Google has been using for many years in their datacenters
  - Google Borg inspired the development of Kubernetes
  - Borg has been used for over 15 years for hosting internal Google applications

Kubernetes Orchestration tasks

- Schedule containers to run on specific hosts using Pods
- Join hosts that are running containers in an etcd cluster
- Take care of scalability
- Make storage available
- Expose containers using a service

Other container management solutions

- Docker Swarm
- Rancher
- Red Hat OpenShift
  - Integrated Kubernetes and adds devops workflow services on top of it
- Other management solutions are normally based on Kubernetes

What are Containers?

- Containers provide a way to package, ship and run applications
  - "Just a fancy way to run an application"
- Docker is a leading solution in containers
  - easily build containers
  - share container images in a simple way by using Docker registries
- Docker Inc offers 2 methods to manage containers in the datacenter
  - Docker swarm: currently part of Mirantis
  - Kubernetes
- Notice that Docker isn't the only container solution around, and even if Kubernetes now focusses on Docker, other container platforms will be integrated also

Container needs in the Datacenter

- A methodology to build, test and verify container images
- A cluster of hosts to run the containers
- Monitoring and self-healing of containers
- A solution for updates and rollbacks
- A flexible network that can self-extend if that is needed

About the Kubernetes Host Platform

Kubernetes can be offered through different host platforms

- As a hosted service in public cloud
- As a set of processes on a Linux host
- As a set of containers running on a Linux host
- Integrated in a minimized container OS as provided by CoreOS or Atomic
- As an all-in-one solution, running on Minikube

CNCF: Standardization on K8s

- Cloud Native Computing Foundation (CNCF) is a governing body that solves issues faced by any cloud native application (so not just Kubernetes)
- Google donated Kubernetes to the Cloud Native Computing Foundation, which is a foundation in Linux Foundation
- CNCF owns the copyright of Kubernetes
- Kubernetes itself uses an Apache license
- Developers need to sign a Contributor License Agreement with CNCF
- CNCF ensures that Kubernetes solutions are highly standardized

Installing a Kubernetes Test Cluster

Minikube Overview

- Minikube offers a complete test environment that runs on Linux, OS-X or Windows
- In this course we'll focus on Minikube as it is easy to setup and has no further dependencies
- In all cases, you'll need to have the **kubectl** client on your management platform
- As mentioned before, any other Kubernetes stack that you have available will do for the labs in this course

Installing Minikube

- A scripted installation is provided for Fedora Workstation as well as Ubuntu 20.04 only
- Install either of these with at least 4 GB RAM and 20 GB disk space (8 GB and 40GB recommended)
- Use **git clone https://github.com/sandervanvugt/kubernetes**
- From there, use the **kube-setup** script and follow instructions

Testing Minikube

- **kubectl cluster-info**: shows cluster information
- **kubectl get nodes**: shows nodes currently available
- **minikube ssh**: logs in to the Minikube host
- **docker ps**: shows all Docker processes on the MK host
- **ps aux | grep localkube**: shows the localkube process on MK host

Working with Images in Kubernetes

- To start using Kubernetes, you'll need container images
- You can create your container image and push it to a public or private registry before referring to it in a Kubernetes pod
- Or you can use public container registries to pull images from
  - Use Docker Hub or any public cloud container registry
  - Docker Hub images make sense in a small private deployment
  - Public cloud container registries make sense if you're running Kubernetes from a Public cloud

Running Your First Application

- From **minikube dashboard**, click +CREATE in the upper right corner
- Specify **httpd** as the container image as well as the container name
- This will pull the Docker image and run it in the minikube environment

Kubernetes in 4 Hours

Understanding Kubernetes Resource Types

Understanding Main Kubernetes Resource Types

- Resource types are defined in the Kubernetes APIs
- *Pods*: the basic unit in Kubernetes, represents a set of containers that share common resources such as an IP address and storage volumes
- *Deployments*: standard entity that is rolled out with Kubernetes
- *Services*: make deployments accessible from the outside by providing a single IP/port combination. Services by default provide access to pods in round-robin fashion using a load balancer
- *Persistent Volumes*: persistent (networked) storage that can be mounted within a container by using a Persistent Volume Claim
- *Many* more are available

Understanding the Pod

- Kubernetes manages Pods, not containers
- A Pod is using *namespaces* to ensure that resources in the Pod can communicate easily
    - In a pre-container environment, containers in a Pod would be implemented as applications on the same computer
- Containers can be put together in a Pod, together with Pod-specific storage, but a typical pod runs one container only

Kubernetes in 4 Hours

Kubernetes Beyond Minikube

K8s in Public Cloud

- Kubernetes is commonly used in public cloud
  - AWS
  - Azure
  - Google Cloud
- Installation in private cloud (OpenStack and others) is also common
- Alternatively, Kubernetes can be installed on premise in local datacenter using **kubeadm**

Installing an on-premise Cluster

- **kubeadm** helps you building a physical cluster, running in your local datacenter
- The documentation is here: https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/
- Rough procedure outline
  - Run **kubeadm init** on the head node
  - Create a network
    - **kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"**
  - Run **kubeadm join --token <token> <head-node-IP-address>** to join worker nodes

Kubernetes in 4 Hours

Accessing and Using the Cluster

Pearson

Methods to access the cluster

- The **kubectl** command line utility provides convenient administrator access, allowing you to run many tasks against the cluster
- Direct API access allows developers to address the cluster using API calls from custom scripts
- The Kubernetes Console offers a web based interface

Using kubectl

- The **kubectl** command is the generic command that allows you to manage all aspects of pods and containers
  - It provides functionality that normally is provided through the **docker** command, but talks to pods instead of containers
- Use **kubectl create** to create pods
- Or **kubectl get ...** or one of the many other options to get information about pods
- Start with **kubectl completion -h**

Kubernetes in 4 Hours

Managing Pods

Managing pods with kubectl

- Use **kubectl run** to run a *pod* based on a default image
  - **kubectl run ghost --image=ghost:0.9**
- Use **kubectl** combined with instructions in a YAML file to do anything you'd like
- **kubectl create -f <name>.yaml**
- **kubectl get pods**
- (see sample code on next slide)
- **kubectl describe pods** shows all details about a pod, including information about containers running within
  - For instance, **kubectl describe pods newhttpd**

Using kubectl in a declarative way

- The recommended way to work with kubectl, is by writing your manifest files and using **kubectl apply -f manifest.yaml** to the current objects in your cluster
- This *declarative* methodology is giving you much more control than the *imperative* methodology where you create all from the CLI
  - Get current state of an object: **kubectl get deployments nginx -o yaml**
  - Push settings from a new manifest: **kubectl replace -f nginx.yaml**
  - Apply settings from a manifest: **kubectl apply -f nginx.yaml**

Creating YAML Files

- YAML files are used in declarative way
- Use **kubectl run mypod --image=nginx --dry-run=client -o yaml > mypod.yaml** to easily generate a YAML file
- Use **kubectl explain** for more information about properties to be used in the YAML files
- Consult kubernetes.io/docs for additional information

Understanding Namespaces

- Namespaces create isolated environments: Pods running in one namespace have no direct access to Pods running in another namespace
- Use namespaces to create virtual datacenters
- Some key resources are non-namespaces as they define essential Kubernetes functionality
- Kubernetes core services run in the **kube-system** namespace

Using Namespaces

- The following commands allow you to work with namespaces
  - **kubectl get ns**
  - **kubectl create ns demo**
  - **kubectl get ns/demo -o yaml**
- As seen in the previous slide, you can add namespaces when creating a pod, thus ensuring that a pod is available in a specific namespace only

Getting More information about Pods

- **kubectl describe** is showing cluster information about Pods
- **kubectl logs** is giving access to the Pod application STDOUT
- **kubectl get pods podname -o yaml** shows detailed information about what is going on in a Pod
- **kubectl exec PODNAME -- /bin/sh** gives access to a shell running within a Pod

Kubernetes in 4 Hours

Working with Deployments

Understanding Replica Sets

- The pod is the most basic entity in Kubernetes
- To determine how many instances of a pod you want to run, you need replica sets
- Replica sets are managed through deployments
- Use **kubectl scale** or **kubectl create … --replicas=n** to run a specific number of replicas for an application

Kubernetes in 4 Hours

Using Scaling and Rollback

Scaling deployments

- **kubectl scale deployment nginx --replicas=3**
  - Notice that scaling deployments may fail because of limited availability of resources
- **kubectl get deployments nginx -o json**
  - Notice that different output formats are available, use **-o yaml** if you prefer seeing it in yaml, or **-o json** for json
  - In the output look for the labels; a run label is automatically added
- **kubectl get pods -Lrun** filters on pods with the label running
  - The run label is used to determine if sufficient replicas are available
  - Remove one "run" label and you'll notice that a new pod will be created immediately
  - Use **kubectl label pods <name> run-** to remove the label run

Understanding Labels

- Labels are name tags used in Deployments that can be set on objects
- Labels are set automatically on most resources, and are used internally by Kubernetes to connect resources
- Use **kubectl get all --show-labels** to see all labels
- Use **kubectl get all --selector app=nginx** to see all resources with a specific label

Kubernetes in 4 Hours

Accessing Services from Outside

Accessing Applications in a pod

- To access an application running in a pod, port-forward can be used
- Run the port-forward argument to **kubectl** as a background process, this will export the pod port locally
  - **kubectl port-forward deployment/nginx 8080:80 &** where
  - **curl localhost:8080**

Understanding Services

- Pods in a deployment don't provide a uniform IP address to access all of them
- To provide such a unique IP address, use services
- Which pods are added to a service is normally determined by using a label selector
- Using labels makes services very flexible: you don't rely on the existence of a specific pod, but of a pod that caries a specific label, which makes it easy to replace pods

Creating Services

- When running a deployment, Pods have an internal network address
- This address is dynamically allocated and cannot be used to address the deployment as it addressed the pod
- The service exposes applications running in Pods on an external IP address
- Different types of services can be used to determine how the service can be accessed

Understanding Service Types

- Different service types determine how services are accessed
  - **ClusterIP** is the default and provides internal access only: useful for internal connections
  - **NodePort** assigns a random port ID and exposes it on the nodes that run the service
  - **LoadBalancer** is available in public cloud. May be used in private cloud, if Kubernetes provides a plugin for that cloud type
  - **externalName** exposes the service using a name
- To access a cluster IP service locally (without exposing it), you may also use **kubectl proxy**
  - After starting kubectl proxy, the service is accessible on the host where **kubectl proxy** was started

Demo: Using Services - 1

- **kubectl create deployment nginxsvc --image=nginx**
- **kubectl scale deployment nginxsvc --replicas=3**
- **kubectl expose deployment nginxsvc --port=80**
- **kubectl describe svc nginxsvc** # look for endpoints
- **kubectl get svc nginx -o=yaml**
- **kubectl get svc**
- **kubectl get endpoints**

Demo: Using Services - 2

- **minikube ssh**
- **curl http://svc-ip-address**
- **exit**
- **kubectl edit svc nginxsvc**

```
  …
  protocol: TCP
  nodePort: 32000
type: NodePort
```

- **kubectl get svc**
- (from host): **curl http://$(minikube ip):32000**

Understanding Ingress

- Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster
- Traffic can be defined using Ingress rules
- Ingress also takes care of SSL/TLS termination
- To use Ingress, an Ingress controller is required
- Use **minikube addons list / minikube addons enable**

Demo: Using Ingress - 1

- minikube addons enable ingress
- kubectl get deployment
- kubectl get svc nginxsvc
- curl http://$(minikube ip):32000
- vim nginxsvc-ingress.yaml
- kubectl apply -f nginxsvc-ingress.yaml
- kubectl get ingress
- sudo vim /etc/hosts
  - $(minikube ip)   nginxsvc.info
- curl nginxsvc.info

Demo: Using Ingress - 2

- **kubectl edit svc nginxsvc**
  - remove **nodePort: 32000**
  - change to **type: ClusterIP**
- **vim nginxsvc-ingress.yaml**
  - change to **servicePort: 80**
- **kubectl apply -f nginxsvc-ingress.yaml**
- **curl nginxsvc.info**

Kubernetes in 4 Hours

Working with Volumes

Understanding Volumes

- Volumes can be mounted in a specific location in the container to provide persistent storage
- Volumes typically are created using YAML files while creating the VMs
- Volumes can be internal to a Pod, and as such are a part of the Pod specification
- Persistent Volumes are a different API object and have been added to decouple storage from the pods that need it

Managing Volumes (demo)

- Creating the Volume
  - **kubectl create -f volumes.yaml**
  - **kubectl get pods**
- After creation, test that you can use it
  - **kubectl exec -ti vol -c centos -- touch /test/myfile**
  - **kubectl exec -ti vol -c centos -- ls -l /test**
  - Note: try this command without --

Understanding Persistent Volumes

- A Persistent Volume is a storage abstraction that is used to store persistent data
  - Use **persistentVolume** to define it
  - Different types (NFS, iSCSI, CephFS and many more) are available
- Using claims with Persistent Volumes creates portable Kubernetes storage that allow you to use volumes, regardless of the specific storage provider
  - Use **persistentVolumeClaims**
  - The persistent volume claim talks to the available backend storage provider and dynamically uses volumes that are available on that storage type
- Kubernetes will use Persistent Volumes according to the availability of the requested volume accessModes and capacity

Kubernetes in 3 Hours

Additional Features

Interesting Kubernetes Features not in this course

- Quota set resource limitations at a namespace level
- Secrets: like configMaps, to work with sensitive data in a way that the data is not readable
- Helm: the Kubernetes Package Manager, applications are packages in a chart and published in a repository as a tarball, allows to group all the different object types that make up an app into one package
- Custom Resource Definitions: the option to create your own objects in the API
- ConfigMaps can be used to decouple site specific configuration from the YAML code
- Kustomization.yaml provides a complex installation script to make setup tasks easier

Kubernetes in 3 Hours

Summary

Next Steps

- To learn more, consider one of the following live courses
  - CKAD Crash Course
  - CKA Crash Course
- Or one of the following recorded courses
  - Getting Started with Kubernetes 2/ed
  - Hands on Kubernetes
  - Certified Kubernetes Application Developer
  - Certified Kubernetes Administrator