Kubernetes in 4 Hours

Sander van Vugt

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
- Running Applications in Pods and Containers
- Exposing Applications using Services
- Using Volumes to Provide Storage
- If time allows: Managing ConfigMaps

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

- See the setup guide in the course resources for different setup options
- Or use any other Kubernetes solution:
  - Docker Desktop: enable Kubernetes support from the main dashboard window
  - Any public cloud based solution (I like Google GCE, but many others exist)
  - Anything else that runs Kubernetes

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
- 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

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, RedHat Podman is upcoming
- Kubernetes adds Pods to manage containers

Container needs in the Datacenter

- Decoupled storage
- 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 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 in 4 Hours

Installing a Kubernetes Test Cluster

Minikube Overview

- Minikube offers a complete test environment that runs on Linux, OS-X or Windows
- Other test environments can also be used
- In all cases, you'll need to have the **kubectl** client on your management platform

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.sh** script and follow instructions
- Warning: currently Mac OS Big Sur is giving problems using nested virtualization and the script will fail on Big Sure

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

- *Pods*: the basic unit in Kubernetes, represents a set of containers that share common resources
- *Deployments*: the application itself, standard entity that is rolled out with Kubernetes
- *Services*: make deployments accessible from the outside by providing a single IP/port combination.
- *Persistent Volumes*: persistent (networked) storage

Understanding the Pod

- Kubernetes manages Pods, not containers
- A Pod is using *namespaces* to ensure that resources in the Pod can communicate easily
- 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

Accessing and Using the Cluster

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 deployment
- 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 nginx --image=nginx**
- Use **kubectl** combined with instructions in a YAML file to do anything you'd like
- **kubectl create -f <name>.yaml**
- **kubectl get pods**
- **kubectl describe pods** shows all details about a pod, including information about containers running within

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 many examples!

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
- Kubernetes core services run in the **kube-system** namespace

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 -it PODNAME -- /bin/sh** gives access to a shell running within a Pod

Kubernetes in 4 Hours

Working with Deployments

Understanding Deployments

- The Deployment implements replication and update strategy
- Replica set defines how many instances of a Pod you'll be running
- Update strategy defines which type of update you'll be using

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

Demo

MANUALLY SETTING

- **kubectl label deployment ghost state=demo**
- **kubectl get deployments --show-labels**
- **kubectl get deployments --selector state=demo**

AUTOMATED

- **kubectl create deployment nginx --image=nginx**
- **kubectl describe deployment nginx** –> look for label
- **kubectl describe pod nginx-xxx**
- **kubectl label pod nginx-xxx app-** → will remove the auto-assigned run label and start a new pod to meet the requirements
- **kubectl get all --selector app=nginx**

Pearson

Kubernetes in 4 Hours

Accessing Services from Outside

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

Understanding Service Types

- **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

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

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 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

Understanding Persistent Volumes

- A Persistent Volume is a storage abstraction that is used to store persistent data
- Using PVC with Persistent Volumes creates portable Kubernetes storage that allow you to use volumes, regardless of the specific storage provider
- StorageClass can be used as the default store in a cluster

Demo

1. **kubectl create -f morevolumes.yaml**
2. **kubectl get pods morevol2**
3. **kubectl describe pods morevol2 | less** ## verify there are two containers in the pod
4. **kubectl exec -ti morevol2 -c centos1 -- touch /centos1/test**
5. **kubectl exec -ti morevol2 -c centos2 -- ls -l /centos2**

Kubernetes in 3 Hours

Optional Topic: Using ConfigMaps

Understanding ConfigMaps

- ConfigMaps can be used to separate dynamic data from static data in a Pod
- Secrets are encoded ConfigMaps which can be used to store sensitive data
- ConfigMaps must be created before the pods that are using them

Using ConfigMaps

- Make variables available within a Pod
- Provide command line arguments
- Mount them on the location where the application expects to find a configuration file

Creating ConfigMaps - 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=cow --from-literal=VAR4=goat**
  - 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

Demo: Creating ConfigMaps for Config Files

- 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**

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
  - Building Microservices with Containers
- Or one of the following recorded courses
  - Getting Started with Kubernetes 2/ed
  - Hands on Kubernetes
  - Certified Kubernetes Application Developer
  - Certified Kubernetes Administrator
  - Modern Container-Based DevOps: Managing Microservices using Kubernetes and Docker