Meetup

Introduction to Kubernetes

ON2IT ZERO TRUST INNOVATORS

This is a Meetup

(not a one-way presentation)

- We most likely won't be done on time
- Questions are welcome and encouraged
- There are no bad questions
- Interact with each-other
- Get dirty, fix and break things

ZERO TRUST INNOVATORS

Thank you

Agenda

- Brief overview of containers
- What is container orchestration and why do we need it
- Kubernetes concepts (basics)
- Hands-on: Installing Kubernetes
- Hands-on: Creating an Ingress controller
- Hands-on: Whack-A-Pod
- Hands-on: Expand your cluster
- Hands-on: Run your own
- Conclusion

Containers (& container images)

- A (container) image is a complete package of software incl. dependencies
  - Defined by a Dockerfile
  - ‘compiled’ by the ‘docker build’-command into a layered file system + meta data
- A container is a runtime instance of an image
- More lightweight than a VM
  - kernel is shared, and host OS is already booted
  - Starting/stopping a container only involves starting/killing a process
- Containers run on a container-runtime (i.e. Docker)
- Ideally stateless
- Upgrade of software is ideally a new container

- Populair in ‘devops’ and ‘ci/cd’ environments

Container Runtime

- Makes containers run
- Will eventually restart containers
- Provides connectivity between containers on the SAME host
- Provides connectivity between container- and the ‘real’ world (port-nat)

- Most well known runtime: Docker

Container Orchestration

- Provisioning and deployment of containers
- Updating containers (rolling updates)
- Redundancy
- Scaling up (and down)
- Movement of containers
- Allocation of resources
- External exposure of services (running in the container world)
- Load balancing / service discovery
- Health monitoring of containers and hosts

- Abstraction, Abstraction, Abstraction (i.e. container runtime, network, storage)

Container Orchestrators

- Docker Swarm
- Mesos (Management) / Marathon (Container Orchestration)
- Rancher (Management) / Cattle (Container Orchestration)
- Kubernetes 
- Vmware PKS, Redhat Openshift (Kubernetes)

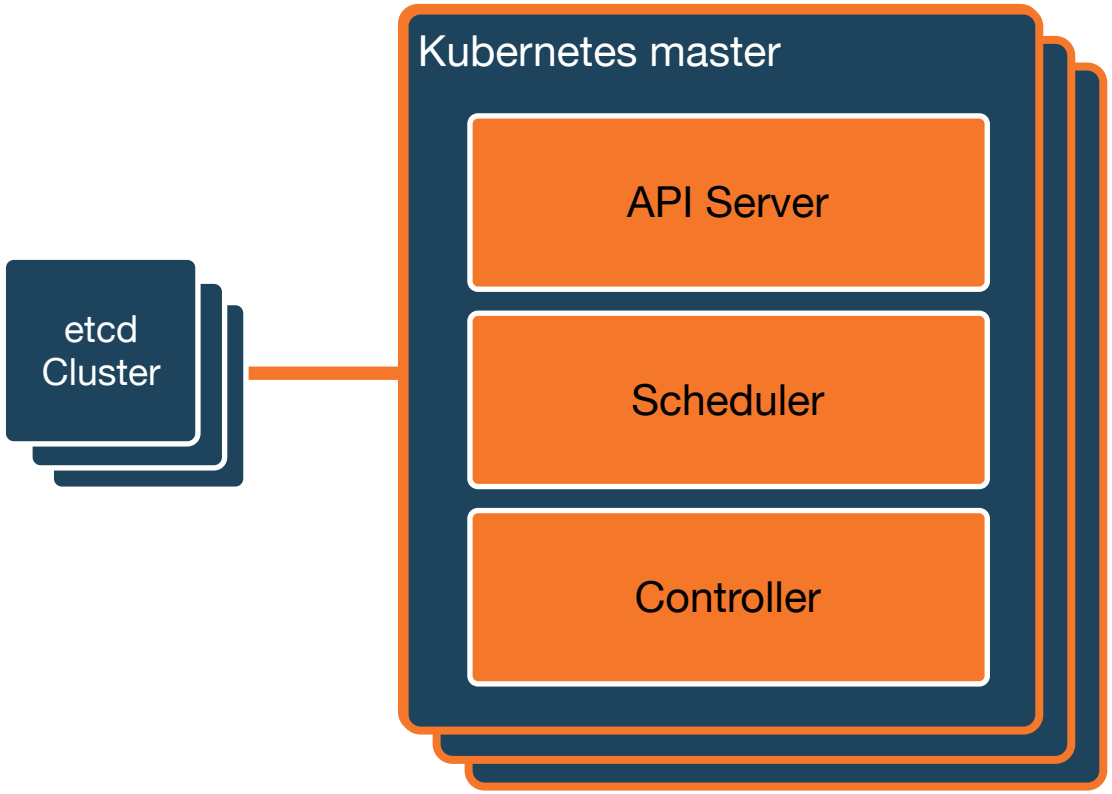
Kubernetes

- Predecessor to Borg
- Initially developed and used by Google
- Now part of CNCF 

- Google is still a big sponsor of Kubernetes project
- It is the ‘de facto’ standard for running/orchestrating containers.
- All major cloud providers have support for Kubernetes.

Kubernetes components

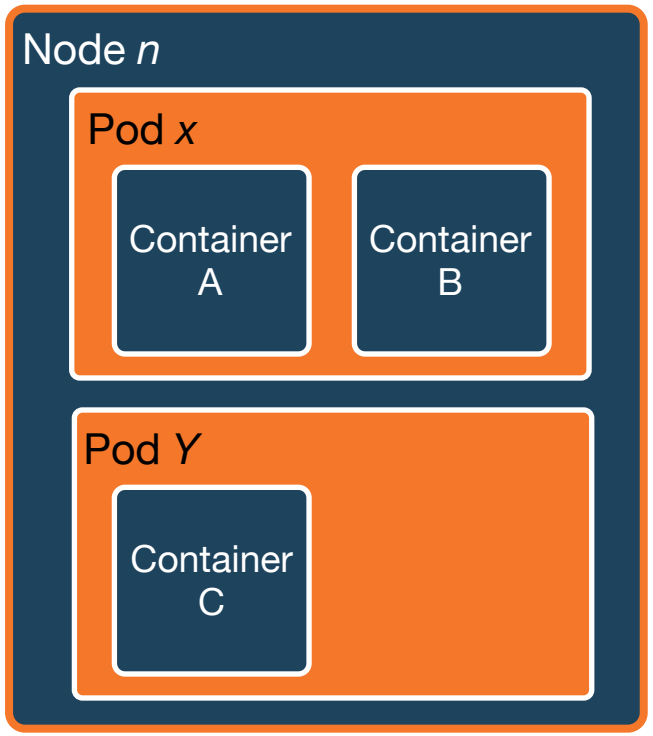
- Kubernetes Master(s)
  - etcd
  - API Service
  - Scheduler
  - Controller
    - Node controller
    - Replication controller
    - Endpoint controller
    - Service Account and Token controller



Kubernetes concepts

- Kubelet (node agent)

- Nodes (Kubernetes hosts)
- Pods ≈ Container(s)
- Services

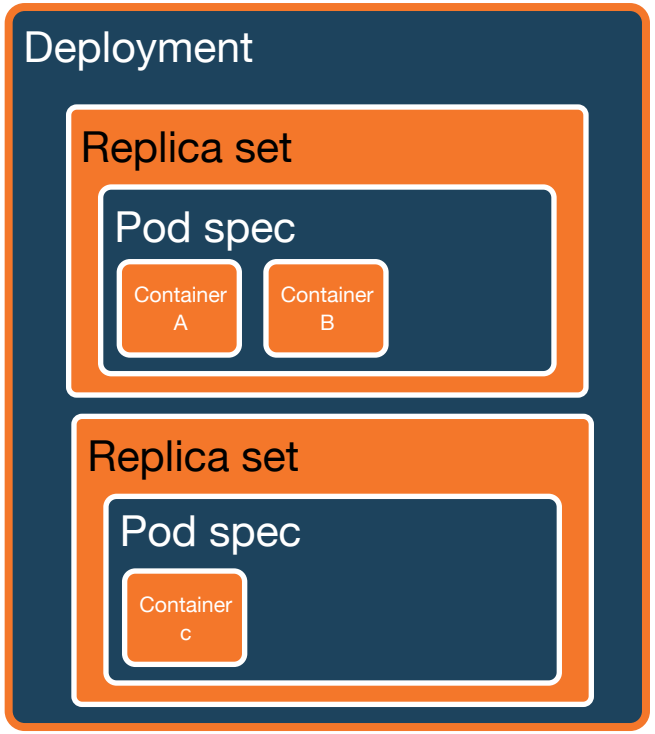


Kubernetes concepts

- Deployments
  - Replicaset
  - Pods
  - …
- Written in yaml
- kubectl apply –f deploy.yaml

“desired state management”

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```



Kubernetes ‘getting started’

- CRI = Container Runtime (we will use Docker)
- CNI = Container Network (we will user Flannel)
  - This makes it possible to stretch container networks across multiple nodes

- kubeadm – bootstrap Kubernetes cluster conform best practices

- kubectl – Kubernetes command-line tool

Hands-On
“Accessing th LAB”

10 min

Hands-On
“Single Node Cluster”

30 min

Single Node Cluster

- https://github.com/on2itsecurity/meetup-kubernetes-introduction
- On the master node (10.10.10.x1):
  - Install CRI - Docker 18.06 (recommend version by Kubernetes)
  - Install kubeadm, kubelet and kubectl
  - Deploy the node
  - Install CNI - Flannel (simple L2 = vxlan)
  - Convert to single node

Explore the environment

- kubectl get all --all-namespaces -o wide
- kubectl get nodes
- kubectl get pods
- kubectl get services
- kubectl get deployments
- kubectl get componentstatus

Hands-On
“Ingress Controller”

15 min

Ingress controller (nginx-ingress)

- Deploy the ‘nginx-ingress’ controller
- Check the service / nodeport

Hands-On

`kubectl get componentstatuses`

“Whack-A-POD”

30 min

Deploy Whack-A-Pod

- Deploy and access the game.
- Goal: Make the blinking light of the board stop
- Play the game and check if you can find out what happens
  - What happens with the pods ?
  - Why does the service fail now and then ?

Hint: The next and advanced view are bit less overwhelming

Hands-On
“Expanding the cluster”

15 min

Add a second node to the cluster

- On the worker node (10.10.10.x2):
  - Install the bare minimum tools: CRI, Kubelet, Kubeadm
  - Join the cluster

- Play ‘Whack-A-Pod’ again and use the advanced view.
  - Can you find the difference with the single-node setup ?

Hands-On
“Run your own workload”

15 min

Final remarks

- Container orchestration makes life easier
  - Kubernetes is the ‘de facto’ standard -> No Vendor Lock-in (for cloud)
- Kubernetes ≈ General Purpose ’Platform-as-a-Service’
  - i.e. multi- / hybrid cloud scenarios
- “Desired State” is the new normal
- Next Meetup: deeper dive on Kubernetes

ZERO TRUST INNOVATORS

Committed to innovation that delivers leading edge Zero Trust cyber security solutions, managed security services and support to our clients.