NobleProg

OpenShift Github Training

Trainer: Alexander Kolin

NobleProg

Welcome

Alexander Kolin

Software/System Architect

Web-Developer

DevOps-Engineer

Cloud & Migration Consultant

NobleProg

Course objectives

Course objectives

Day 1

- Github introduction
- Container introduction
- Container Lab
- Kubernetes Introduction
- OpenShift overview
- Deployment
- Openshift networking

Course objectives

Day 1

- Github introduction
- Container introduction
- Container Lab
- Kubernetes Introduction
- OpenShift overview
- Deployment
- Openshift networking

Day 2

- Github Actions
- Kustomize
- Advanced deployment
- Storage

NobleProg

Course objectives

Day 1

- Github introduction
- Container introduction
- Container Lab
- Kubernetes Introduction
- OpenShift overview
- Deployment
- Openshift networking

Day 2

- Github Actions
- Kustomize
- Advanced deployment
- Storage

Day 3

- Configmaps & Secrets
- Monitoring
- Capstone project
- Wrap Up & Questions

NobleProg

Github

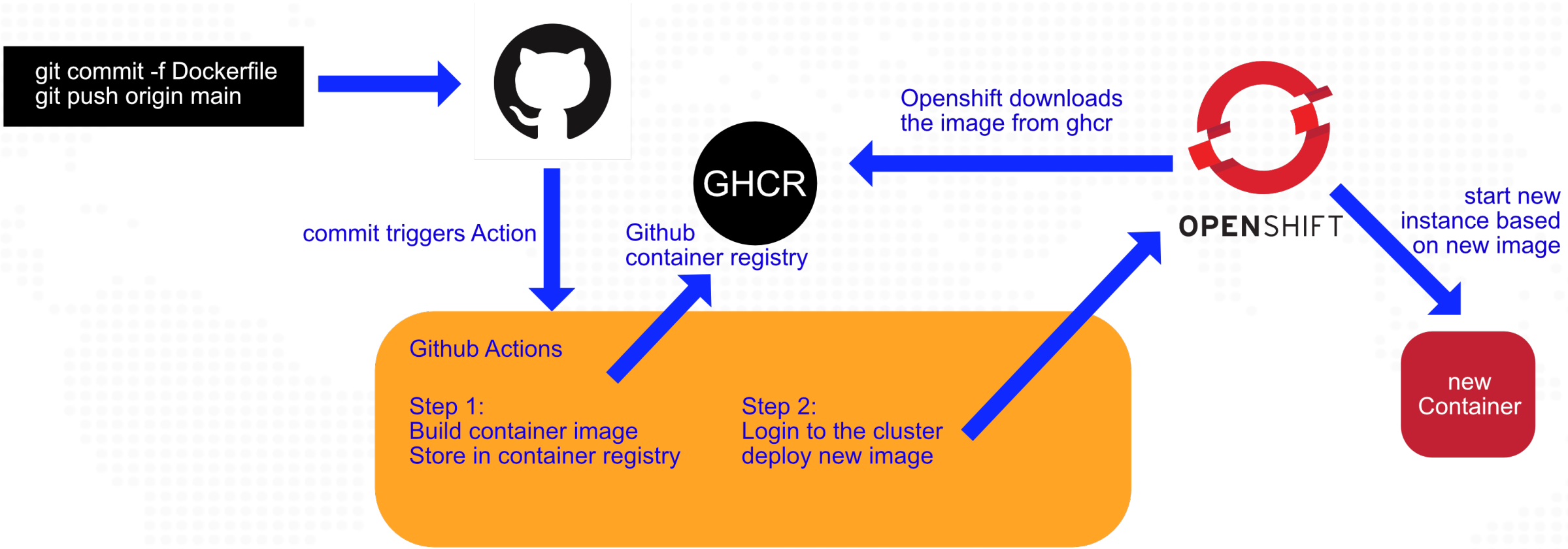
- Git Server
- Container registry
- Deployment Workload provider
- Opensource

- Open terminal and type:

  $ git clone https://github.com/kolinrr/openshift.git

NobleProg

Github

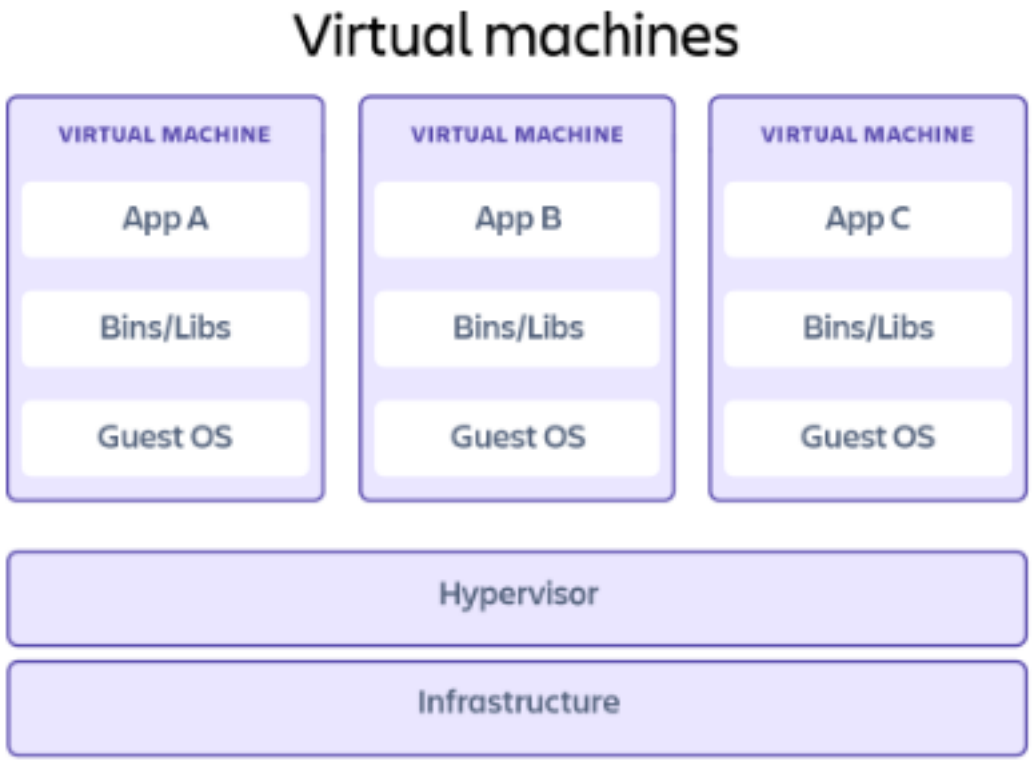


NobleProg

Container basics

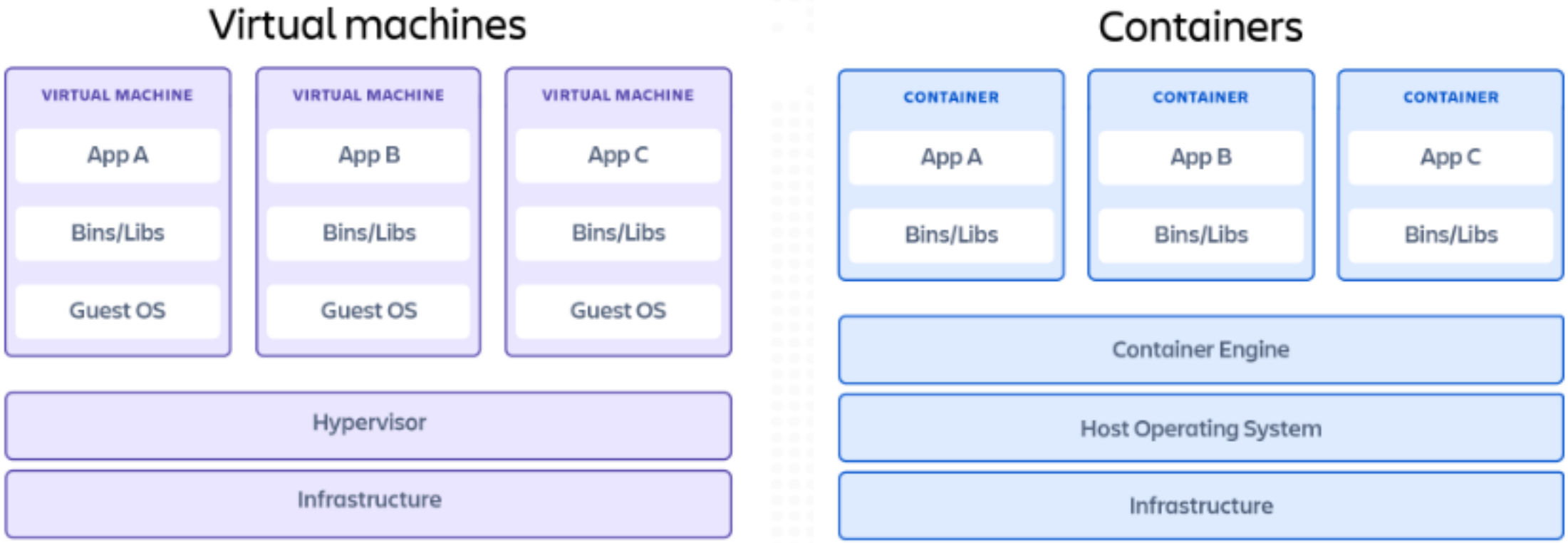
(c) Atlassian https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms

Container basics



(c) Atlassian https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms

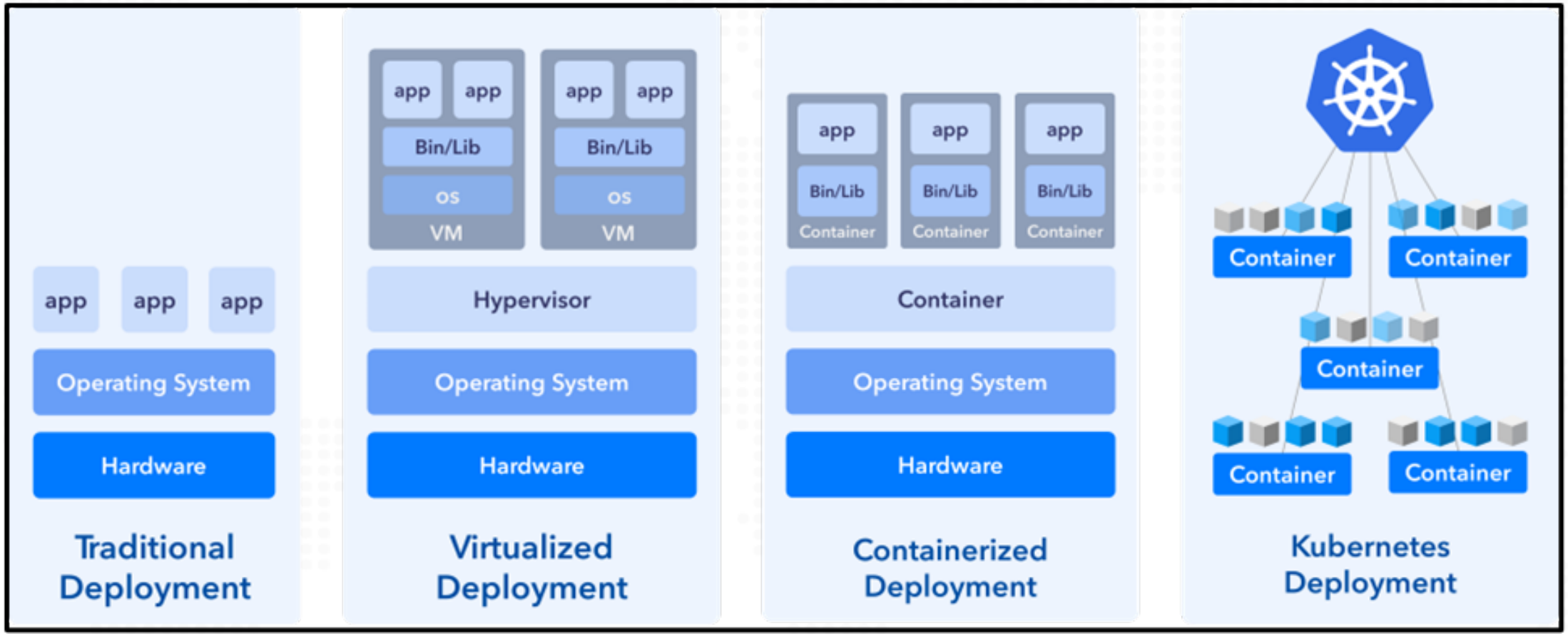
Container basics



(c) Atlassian https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms

NobleProg

Cloud solutions



(c) https://www.cloudthat.com/resources/blog/introduction-to-kubernetes-on-vsphere-part-1

NobleProg

How To Container

- Docker vs Podman
  - Podman is Daemonless Tool - Easier to use in automation (DinD)
  - Podman developed by Redhat
  - Rootles - more secure
  - very similar to docker
  - Allows to create a pod - multiple container in one instance
- Docker registry
  - hub.docker.com - official Docker-Registry
  - ghcr.io - Github

Container Lab

Container vs Image?!

Container Lab

Container vs Image?!

Docker

$ docker pull *image-name*

$ docker build -t *image-name* -f my-dockerfile

$ docker run -d --name *container-name* --p 8080:80 *image-name*

$ docker ps //process status

$ docker logs *container-id or container-name*

$ docker exec -it *container-id or container-name* /bin/bash

$ docker stop *container-id or container-name*

NobleProg

Container Lab

Container vs Image?!

Docker

$ docker pull *image-name*

$ docker build -t *image-name* -f my-dockerfile

$ docker run -d --name *container-name* --p 8080:80 *image-name*

$ docker ps //process status

$ docker logs *container-id or container-name*

$ docker exec -it *container-id or container-name* /bin/bash

$ docker stop *container-id or container-name*

… and Podman?

NobleProg

Container Lab

Container vs Image?!

Docker

$ docker pull *image-name*

$ docker build -t *image-name* -f my-dockerfile

$ docker run -d --name *container-name* --p 8080:80 *image-name*

$ docker ps //process status

$ docker logs *container-id or container-name*

$ docker exec -it *container-id or container-name* /bin/bash

$ docker stop *container-id or container-name*

… and Podman?

$ podman pull *image-name*

$ … build -t *image-name* -f my-dockerfile

$ … run -d --name *container-name* --p 8080:80 *image-name*

$ … ps //process status

$ … logs *container-id or container-name*

$ … exec -it *container-id or container-name* /bin/bash

$ … stop *container-id or container-name*

NobleProg

Container Lab

Container vs Image?!

Docker

$ docker pull *image-name*

$ docker build -t *image-name* -f my-dockerfile

$ docker run -d --name *container-name* --p 8080:80 *image-name*

$ docker ps //process status

$ docker logs *container-id or container-name*

$ docker exec -it *container-id or container-name* /bin/bash

$ docker stop *container-id or container-name*

… and Podman?

$ podman pull *image-name*

$ … build -t *image-name* -f my-dockerfile

$ … run -d --name *container-name* --p 8080:80 *image-name*

$ … ps //process status

$ … logs *container-id or container-name*

$ … exec -it *container-id or container-name* /bin/bash

$ … stop *container-id or container-name*

Try out: podman pull bitnami/nginx

NobleProg

Container Lab

Create own container

Dockerfile

```
git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/container
```

Container Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/container

Dockerfile

Create own container

Open your terminal

$ cd openshift/Day1/container/

$ cat Dockerfile

$ podman build -t myimage -f Dockerfile

$ podman images

Container Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/container

Create own container

Dockerfile

Open your terminal

$ cd openshift/Day1/container/

$ cat Dockerfile

$ podman build -t myimage -f Dockerfile

$ podman images

$ podman run -d --name mycontainer --p 8080:80 myimage

$ podman ps

$ podman logs mycontainer

NobleProg

Container Lab

Create own container

Open your terminal

$ cd openshift/Day1/container/

$ cat Dockerfile

$ podman build -t myimage -f Dockerfile

$ podman images

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/container

Dockerfile

```
FROM bitnami/nginx:latest
```

$ podman run -d --name mycontainer --p 8080:80 myimage

$ podman ps

$ podman logs mycontainer

NobleProg

Container Lab

Create own custom container

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/container/custom

Dockerfile

Container Lab

Create own custom container

Open your terminal

$ cd openshift/Day1/container/custom

$ cat Dockerfile

$ podman build -t customimage -f Dockerfile

$ podman images

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/container/custom

Dockerfile

NobleProg

Container Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/container/custom

Dockerfile

Create own custom container

Open your terminal

$ cd openshift/Day1/container/custom

$ cat Dockerfile

$ podman build -t customimage -f Dockerfile

$ podman images

$ podman run -d --name customcontainer --p 8080:80 customimage

$ podman ps

$ podman logs customcontainer

NobleProg

Container Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/container/custom

Dockerfile

```
FROM bitnami/nginx:latest
```

Create own custom container

Open your terminal

$ cd openshift/Day1/container/custom

$ cat Dockerfile

$ podman build -t customimage -f Dockerfile

$ podman images

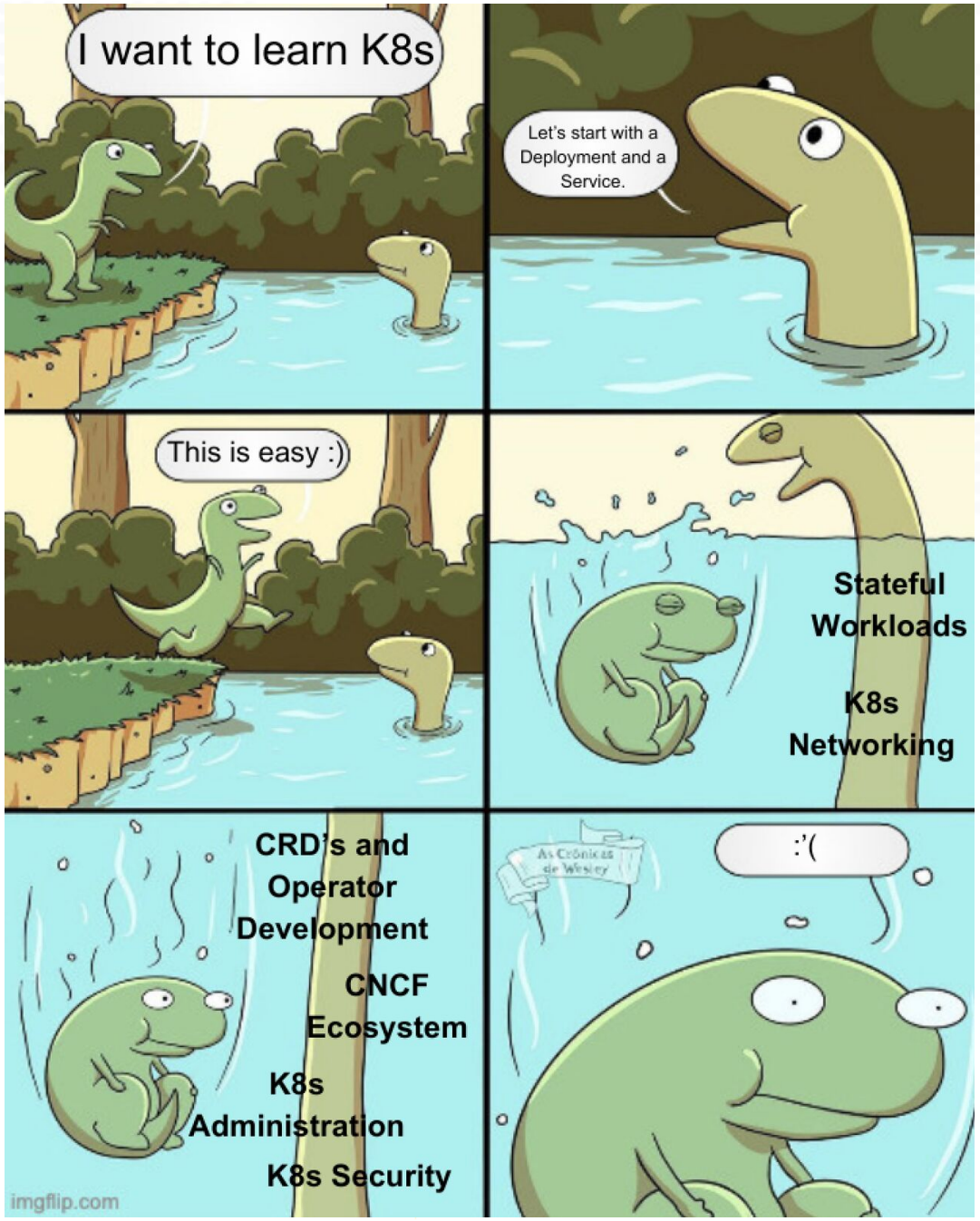
$ podman run -d --name customcontainer --p 8080:80 customimage

$ podman ps

$ podman logs customcontainer

NobleProg

Lunch Break: 30 minutes


I want to learn K8s
Let's start with a Deployment and a Service.
This is easy :)
Stateful Workloads
K8s Networking
CRD's and Operator Development
CNCF Ecosystem
K8s Administration
K8s Security
:'(
imgflip.com

Kubernetes

- Container orchestrations platform
- IaC
- Runs on bare metal, (v)cloud and hybrid
- Scaling
- K8s?!

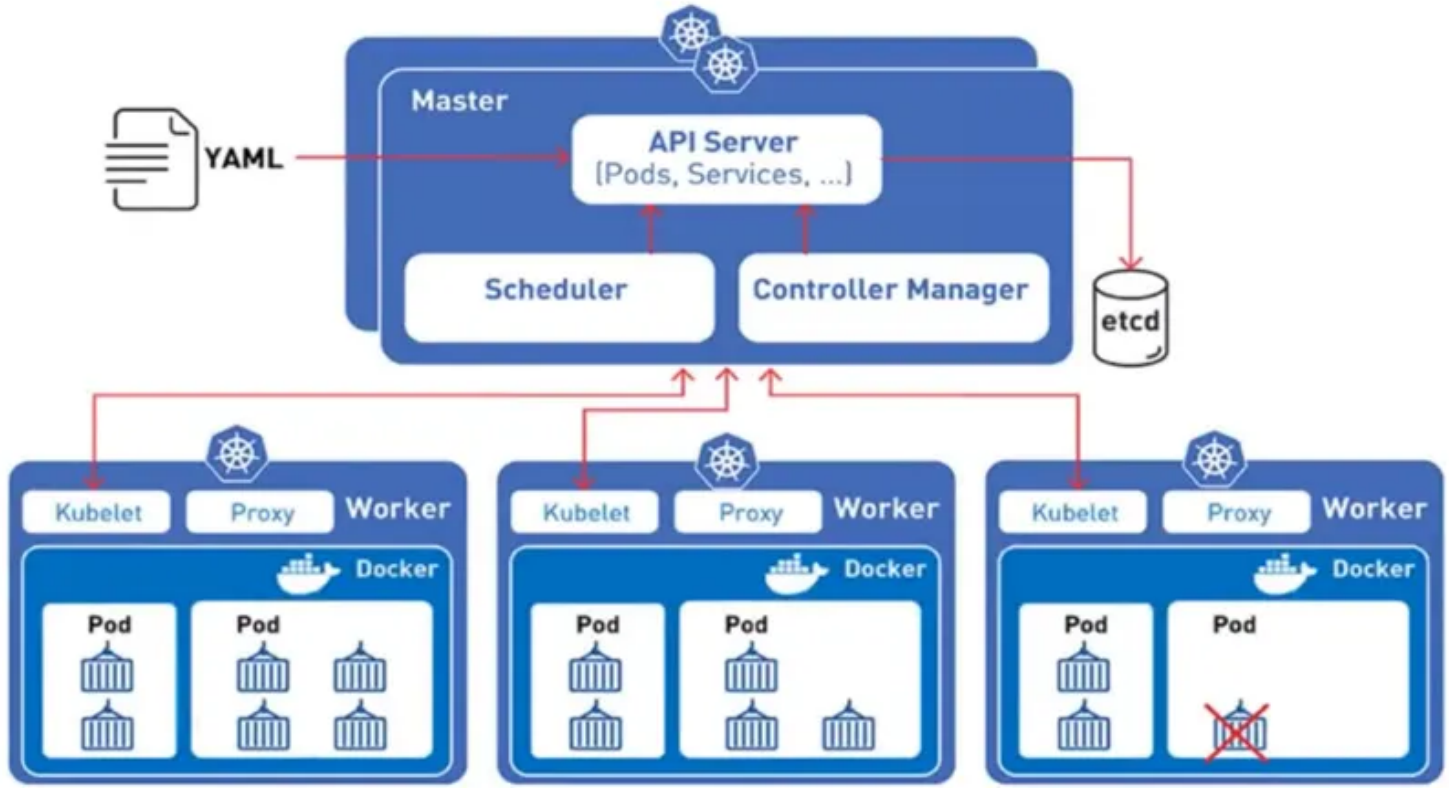
Kubernetes

- Container orchestrations platform
- IaC
- Runs on bare metal, (v)cloud and hybrid
- Scaling
- K8s?!

kubernetes
12345678
k8s

NobleProg

Kubernetes Architecture



(c) https://www.csharp.com/article/getting-started-with-kubernetes-part2/

NobleProg

Openshift

- Kubernetes distribution from RedHat
- Couple of Inbox features: Automation, routing
- Security
- Usability & extended GUI
- Hybrid- & Multi-Cloud support
- Enterprise support

Openshift Deployment

- Openshift/Kubernetes Basics:
  - Container -> Pod -> Deployment / ReplicaSet / StatefulSet / DaemonSet / Job / CronJob

Openshift Deployment

- Openshift/Kubernetes Basics:
  - Container -> Pod -> Deployment / ReplicaSet / StatefulSet / DaemonSet / Job / CronJob
- Openshift/Kubernetes CLI

Openshift Deployment

- Openshift/Kubernetes Basics:
  - Container -> Pod -> Deployment / ReplicaSet / StatefulSet / DaemonSet / Job / CronJob

- Openshift/Kubernetes CLI

```
$ oc get pods

$ oc get deployments

$ oc apply -f my.yaml

$ oc describe pod/deployment/service… *name*

$ oc scale deployment *name* --replicas=2

$ oc exec -it my-pod -c container-x -- /bin/bash

$ oc logs my-pod -c nginx-container

$ oc delete pod/deployment/service… *name*
```

NobleProg

Openshift Lab

- crc start -p $HOME/pull-secret
- oc login -u kubeadmin
- crc console

Openshift Lab

- my-pod.yaml

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/deployment

Openshift Lab

- my-pod.yaml

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/deployment

```
$ oc apply -f my-pod.yaml
$ oc get pods -n <namespace>
$ oc describe pod/deployment/service… *name*
$ oc exec -it my-pod -c container-x -- /bin/bash
$ oc logs my-pod -c nginx-container
$ oc delete pod/deployment/service… *name*
$ oc port-forward pod/nginx-pod 1337:8080
```

NobleProg

Openshift Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/deployment

- my-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: bitnami/nginx
      ports:
        - containerPort: 8080
```

```
$ oc apply -f my-pod.yaml

$ oc get pods -n <namespace>

$ oc describe pod/deployment/service… *name*

$ oc exec -it my-pod -c container-x -- /bin/bash

$ oc logs my-pod -c nginx-container

$ oc delete pod/deployment/service… *name*

$ oc port-forward pod/nginx-pod 1337:8080
```

NobleProg

Openshift Lab

- my-deployment.yaml

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/deployment

Openshift Lab

- my-deployment.yaml

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/deployment

```
$ oc apply -f my-deployment.yaml
$ oc get pods -n <namespace>
$ oc get deployments
$ oc describe pod/deployment/service… *name*
$ oc scale deployment *name* --replicas=2
$ oc exec -it my-pod -c container-x -- /bin/bash
$ oc logs my-pod -c nginx-container
$ oc delete pod/deployment/service… *name*
$ oc port-forward pod/nginx-pod 1337:8080
```

NobleProg

Openshift Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/deployment

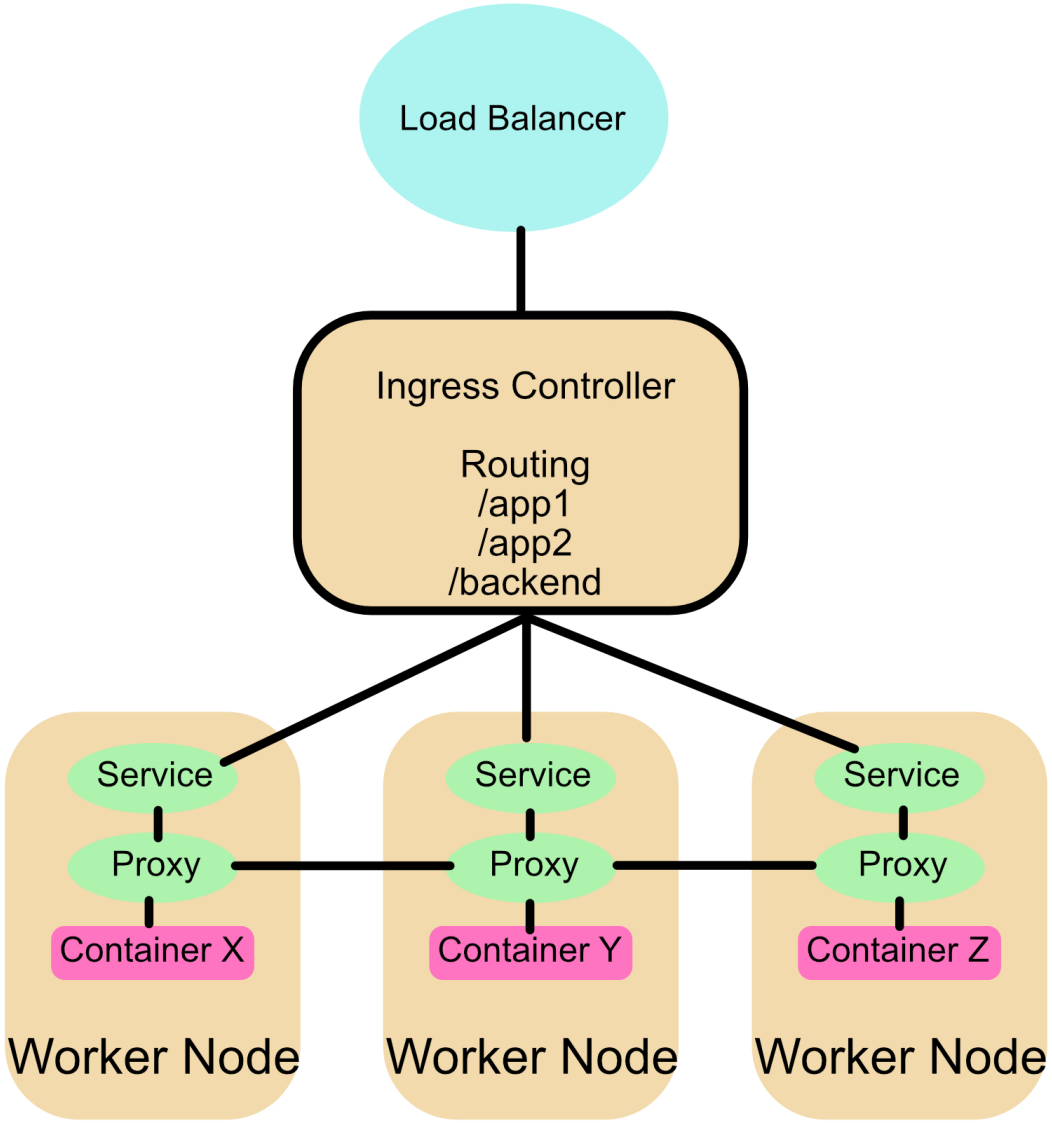
- my-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: bitnami/nginx:latest
        ports:
        - containerPort: 8080
        imagePullPolicy: IfNotPresent
```

```
$ oc apply -f my-deployment.yaml

$ oc get pods -n <namespace>

$ oc get deployments

$ oc describe pod/deployment/service… *name*

$ oc scale deployment *name* --replicas=2

$ oc exec -it my-pod -c container-x -- /bin/bash

$ oc logs my-pod -c nginx-container

$ oc delete pod/deployment/service… *name*

$ oc port-forward pod/nginx-pod 1337:8080
```

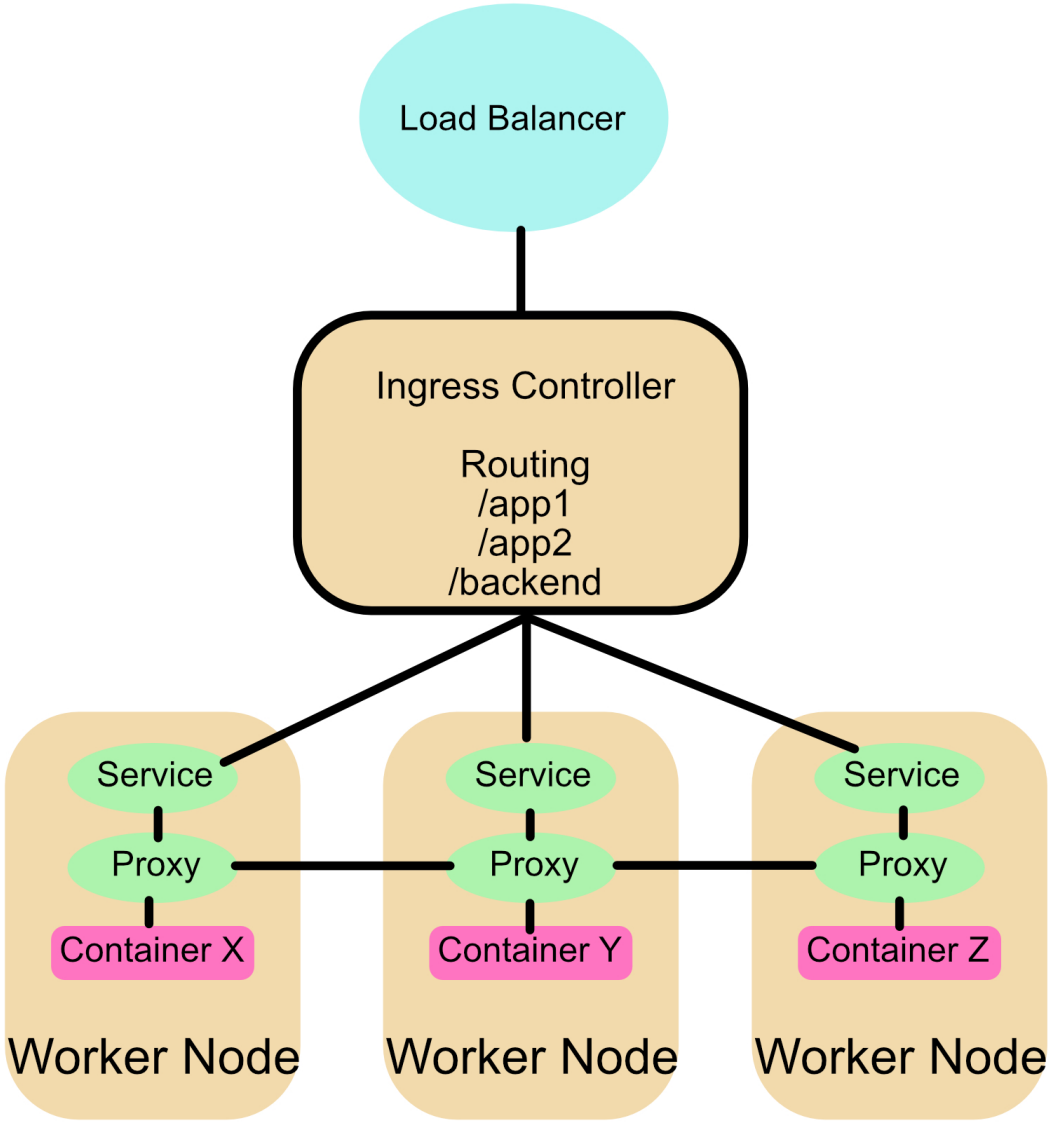
NobleProg

Openshift Networking



Openshift Networking

K8s-Service:
- Role of Internal DNS-Service on the Cluster
- Port mapping
- Types:
  - ClusterIP (default)
  - NodePort
  - LoadBalancer (bind external Service like AWS)
  - ExternalName (internal DNS)



NobleProg

Openshift Networking

K8s-Service:

- Role of Internal DNS-Service on the Cluster
- Port mapping
- Types:
  - ClusterIP (default)
  - NodePort
  - LoadBalancer (bind external Service like AWS)
  - ExternalName (internal DNS)

Openshift Networking

K8s-Service:

- Role of Internal DNS-Service on the Cluster
- Port mapping
- Types:
  - ClusterIP (default)
  - NodePort
  - LoadBalancer (bind external Service like AWS)
  - ExternalName (internal DNS)

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8082
```

NobleProg

Openshift Networking

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8082
```

K8s-Service:

- Role of Internal DNS-Service on the Cluster
- Port mapping
- Types:
  - ClusterIP (default)
  - NodePort
  - LoadBalancer (bind external Service like AWS)
  - ExternalName (internal DNS)

Ingress-Controller:

- Highlevel Reverse-Proxy routing between **Load Balancer** and **Services**
  - **mydomain.com/app1:80 -> Service app1:8080**
  - **mydomain.com/backend:80 -> Service backend:8081**

NobleProg

Openshift Networking

K8s-Service:
- Role of Internal DNS-Service on the Cluster
- Port mapping
- Types:
  - ClusterIP (default)
  - NodePort
  - LoadBalancer (bind external Service like AWS)
  - ExternalName (internal DNS)

Ingress-Controller:

- Highlevel Reverse-Proxy routing between **Load Balancer** and **Se**
  - **mydomain.com/app1:80 -> Service app1:8080**
  - **mydomain.com/backend:80 -> Service backend:8081**

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8082
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: myingress
spec:
  rules:
    - host: myapp.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service
                port:
                  number: 8080
```

NobleProg

Openshift Networking

K8s-Service:
- Role of Internal DNS-Service on the Cluster
- Port mapping
- Types:
  - ClusterIP (default)
  - NodePort
  - LoadBalancer (bind external Service like AWS)
  - ExternalName (internal DNS)

Ingress-Controller:

- Highlevel Reverse-Proxy routing between **Load Balancer** and **Se**
  - **mydomain.com/app1:80 -> Service app1:8080**
  - **mydomain.com/backend:80 -> Service backend:8081**

Route:

- Same as Ingress, already included in openshift

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8082
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: myingress
spec:
  rules:
    - host: myapp.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service
                port:
                  number: 8080
```

NobleProg

Openshift Networking

K8s-Service:

- Role of Internal DNS-Service on the Cluster
- Port mapping
- Types:
  - ClusterIP (default)
  - NodePort
  - LoadBalancer (bind external Service like AWS)
  - ExternalName (internal DNS)

Ingress-Controller:

- Highlevel Reverse-Proxy routing between **Load Balancer** and **Se**
  - **mydomain.com/app1:80 -> Service app1:8080**
  - **mydomain.com/backend:80 -> Service backend:8081**

Route:

- Same as Ingress, already included in openshift

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8082
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: myingress
spec:
  rules:
    - host: myapp.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service
                port:
                  number: 8080
```

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: myroute
spec:
  host: myapp.example.com
  to:
    kind: Service
    name: nginx-service
  port:
    targetPort: 8080
```

NobleProg

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

```
$ oc apply -f service.yaml

$ oc get service

$ oc describe pod/deployment/service… *name*

$ oc port-forward service/nginx 1337:8080
```

NobleProg

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

```
$ oc apply -f service.yaml

$ oc get service

$ oc describe pod/deployment/service… *name*

$ oc port-forward service/nginx 1337:8080
```

NobleProg

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

```
$ oc apply -f service.yaml

$ oc get service

$ oc describe pod/deployment/service… *name*

$ oc port-forward service/nginx 1337:8080
```

NobleProg

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 30080
```

```
$ oc apply -f service.yaml

$ oc get service

$ oc describe pod/deployment/service… *name*

$ oc port-forward service/nginx 1337:8080
```

NobleProg

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

```
$ oc apply -f service.yaml

$ oc get service

$ oc describe pod/deployment/service… *name*
```

NobleProg

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-external
spec:
  type: ExternalName
  externalName: google.com
```

```
$ oc apply -f service.yaml

$ oc get service

$ oc describe pod/deployment/service… *name*
```

NobleProg

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

```
$ oc apply -f route.yaml

$ oc get route

$ oc describe pod/deployment/service… *name*
```

NobleProg

Openshift Networking - Lab

git clone https://github.com/kolinrr/openshift.git

cd openshift/Day1/network

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: myroute
spec:
  to:
    kind: Service
    name: nginx
  port:
    targetPort: 8080
```

```
$ oc apply -f route.yaml

$ oc get route

$ oc describe pod/deployment/service… *name*
```

NobleProg

Openshift Sandbox

- Get own sandbox: https://developers.redhat.com/developer-sandbox

Github Actions & Openshift Deployment Lab

- Register / Login Github https://github.com/ ***avoid caps in name**
- Fork https://github.com/kolinrr/openshift/fork
- git clone https://github.com/*myuser*/openshift
- git remote set-url origin https://*myuser*@github.com/*myuser*/openshift.git
- Generate new PAT : https://github.com/settings/tokens
- git commit -m „init commit“
- git push
- Provide Token as password

NobleProg

Finish!

Questions?