Harmonogram

9:00 - 10:30 I blok
10:30 - 10:45 Przerwa 15 min
10:45 - 12:00 II blok
12:00 - 12:15 Przerwa 15 min
12:15 - 13:15 III Blok
13:15 - 14:00 Przerwa obiadowa 45m
14:00 - 15:30 IV blok
15:30 - 15:45 Przerwa 15 min
15:45 - 17:00 V blok

Slides: https://docs.google.com/presentation/d/1S-Rs9FObiDcs5ry3KJa4n7ou6M6mH42jwPPW572hXqU/edit?slide=id.g31bcb11b552_0_208#slide=id.g31bcb11b552_0_208

Stacjonarne:

- Docker i Kubernetes - stacjonarnie:
- https://www.alx.pl/ankiety/7654wvvk

- Wprowadzenie do Devops - CI/CD, GIT, Jenkins, Ansible, Docker, Kubernetes - stacjonarnie:
- https://www.alx.pl/ankiety/7652efep

ZDALNIE (ZOOM)

- Wprowadzenie do Devops - CI/CD, GIT, Jenkins, Ansible, Docker, Kubernetes - zdalnie:
- https://www.alx.pl/ankiety/7653mnno
- Docker i Kubernetes - zdalnie:
- https://www.alx.pl/ankiety/7655heaq
- Kubernetes w praktyce:
- https://www.alx.pl/ankiety/7656gsiu

Docker

docker version

hostname
ip a

ls /
cat /etc/os-release

ps aux

docker run -it debian bash

hostname
ip a
apt-get update
apt-get install iproute2

systemctl status docker.service
ls -la /var/run/docker.sock
ls -la /var/run/docker.pid

pidof dockerd
cat /var/run/docker.pid
docker info

Docker image registires

https://hub.docker.com/
https://hub.docker.com/_/debian

docker pull quay.io/prometheus/node-exporter

docker search ubuntu

docker search ubuntu --limit 10 --filter=stars=3

Docker tags

docker pull nginx
docker pull redis/redis-stack
docker images

docker pull redis/redis-stack:7.4.0-v8

docker pull redis/redis-stack@sha256:f3a4ca8891fcef481109e663463206d1639a870cba2e5a49a696363abf4e7f95

docker pull redis/redis-stack:7.2.0-v20

docker run

docker run --rm -p 8080:80 nginx

http://localhost:8080/

ip addr show dev eth0

docker run -d --name webserver -p 8081:80 nginx
docker ps

https://github.com/moby/moby/blob/master/internal/namesgenerator/names-generator.go
http://localhost:8081/

docker run -d --name redis-rocks -p 8001:8001 redis/redis-stack
http://localhost:8001/

http://localhost:8001/redis-stack/workbench
SET training01 k8s-rocks
GET training01

docker ps
docker ps -a
docker stop redis-rocks
docker ps -a

Container start/stop/restart

docker create -p 8080:80 --name nginx007 nginx
docker start containerID

curl localhost:8080

docker ps -a
docker rm fervent_robinson

docker images
docker image rm debian:latest

remove debian container
docker rm b6b24e22b38d
docker image rm debian:latest

docker container stop 686b7d7722d1
docker container start 686b7d7722d1

running container
docker rm --force nginx007

docker exec -it redis-rocks /bin/sh
redis-cli

SET Training01 redis-test
GET Traininig01

docker exec -it redis-rocks whoami

docker run --name hello hello-world

docker logs hello

docker restart hello
docker logs hello

Container writable-layer

docker rm --force redis-rocks

docker run -d --name redis-rocks -p 8001:8001 redis/redis-stack

docker volume ls

docker exec -it redis-rocks ls -R /data/

http://localhost:8001/redis-stack/workbench
SET mode writable-layer

docker restart redis-rocks

SET mode writable-layer
SAVE

docker exec -it redis-rocks ls -R /data/

docker restart redis-rocks

Bind mount

docker run -d --name redis-rocks -p 8001:8001 -v /redis-data/:/data redis/redis-stack

docker exec -it redis-rocks redis-cli SAVE

ls /redis-data/

Create volume

docker volume create redis-vol

docker volume inspect redis-vol

docker run -d --name redis-rocks -p 8001:8001 -v redis-vol:/data redis/redis-stack

docker exec -it redis-rocks redis-cli SAVE

docker volume inspect redis-vol | jq ".[].Mountpoint"

sudo ls -la /var/lib/docker/volumes/redis-vol/_data

docker run -d --name redis-rocks --restart unless-stopped -p 8001:8001 -v redis-vol:/data redis/redis-stack

docker ps
systemctl restart docker.service
docker ps
docker start webserver

volume-nocopy

NFS volume

sudo apt-get install -y nfs-kernel-server
sudo systemctl status nfs-kernel-server.service

sudo mkdir /nfs
sudo chown nobody:nogroup /nfs

ip addr show dev eth0
sudo vim /etc/exports

```
echo "/nfs 10.4.0.0/24(rw)" | sudo tee -a /etc/exports

cat /etc/exports

sudo systemctl restart nfs-kernel-server

nc 10.4.0.XXX 2049 -v
sudo mkdir /nfs-dump
 sudo mount 10.4.0.10:/nfs /nfs-dump/

 docker volume ls

 docker volume create --driver local \
  --opt type=nfs \
  --opt o=addr=10.4.0.XXX,rw \
  --opt device=:/nfs \
  redis-nfs

  docker rm --force redis-rocks

  docker run -d --name redis-rocks --restart unless-stopped  -p 8001:8001 -v redis-nfs:/data redis/redis-stack

  docker exec -it redis-rocks redis-cli SAVE

  docker volume inspect redis-nfs

 docker inspect redis-rocks
 df -h

 open /nfs
```

Network

```
 docker network ls

 docker inspect bridge
 ipcalc 172.17.0.0/16

 docker run --rm -d --name nginx02 nginx
 docker run --rm -it --name curl101 nicolaka/netshoot

 docker inspect bridge
```

curl 172.17.0.4

doesn't resolve
curl nginx02

docker network create demo01

docker run --rm -it --network demo01 --name curl102 nicolaka/netshoot
docker inspect demo01

docker run --rm -d --network demo01 --name nginx03 nginx

in netshoot container
curl nginx03

Dockerfile

mkdir -p /home/kurs/src/docker-website
cd /home/kurs/src/docker-website

touch Dockerfile

```
FROM nginx:1.27.3

LABEL maintainer="nginx@ninja.pl"

ENV NINJA_NGINX_VERSION="0.1"

# comment
RUN apt-get update && \
apt-get install -y \
iproute2 \
wget \
&& rm -rf /var/lib/apt/lists/

COPY index.html /usr/share/nginx/html/index.html
```

docker build -t nginx-ninja:v1 .

docker images

docker run --rm --name ninja01 -p 8080:80 nginx-ninja:v1

docker build --build-arg DEST_FILE=ninja.html -t nginx-ninja:v2 .

docker run --rm --name ninja01 -p 8080:80 nginx-ninja:v2

http://localhost:8080/ninja.html

Size matters - go app image build

git clone https://github.com/max-mulawa/materials.git
cd materials/docker/go-app

```
FROM golang:1.24

LABEL maintainer="random dev"

RUN apt-get update && \
 apt-get install -y curl

WORKDIR /src

COPY go.mod go.sum ./
RUN go mod download

COPY cmd/ ./cmd/
COPY pkg/ ./pkg/

RUN CGO_ENABLED=0 go build -o ./api cmd/web/main.go

EXPOSE 8080

ENTRYPOINT ["/src/api"]
```

docker build -t go-app:v1 .

docker images

docker run --rm --name goapp -p 8080:8080 go-app:v1

http://localhost:8080/

ps aux | grep api

Multi-stage build

```
FROM golang:1.24 AS build

LABEL maintainer="random dev"

RUN apt-get update && \
 apt-get install -y curl

WORKDIR /src

COPY go.mod go.sum ./
RUN go mod download

COPY cmd/ ./cmd/
COPY pkg/ ./pkg/

RUN CGO_ENABLED=0 go build -o ./api cmd/web/main.go

# end result (image build)
FROM gcr.io/distroless/static-debian12
COPY --from=build /src/api /app/api

USER nonroot

EXPOSE 8080

ENTRYPOINT ["/app/api"]


docker build -t go-app:v2 .

docker pull gcr.io/distroless/static-debian12

docker run --rm --name goapp -p 8080:8080 go-app:v2

docker login -u mulawam
dckr_pat_doQZzGaaECUl2OhjlSXAtdqS47A

docker tag go-app:v2 mulawam/go-app:v2-xx
docker images

docker push mulawam/go-app:v2-xx

docker images -f "dangling=true"
```

yaml 101

```
fullName: "Tyson Fury"
country: "NO"
active: true

heavyweightChamption:
- 2015
- 2020
record: [30, 3, 1]

features:
  weight: 240
  height: 204


fights:
  - with: Usyk
    result: loss
    date: 2024-12-22
  - with: Usyk
    result: loss
    date: 2023-12-05

notes: >
  Line 1
  Line 1 cnt

notes: |
  Line 1
  Line 2
```

cat fighter.yaml | yq

Docker compose

```
echo -n "mypass" > /tmp/db-password.txt

services:
  postgres:
    image: postgres:16
    ports:
     - 5432:5432
```

```
    volumes:
      - db-data:/var/lib/postgresql/data
      - ./messages.sql:/docker-entrypoint-initdb.d/messages.sql
    secrets:
      - db-password
    environment:
      - POSTGRES_PASSWORD_FILE=/run/secrets/db-password
      - POSTGRES_USER=api
  api:
    image: go-app:v2
    ports:
    - 8080:8080
    secrets:
    - db-password
    environment:
     - DB_USER=api
     - DB_PASSWORD_FILE=/run/secrets/db-password
     - DB_HOST=postgres
     - DB_PORT=5432
volumes:
  db-data:
secrets:
  db-password:
    file: /tmp/db-password.txt
```

```
docker compose up
curl localhost:8080/messages

curl -X POST -H 'Content-Type: application/json' -d '{"id":1,"title":"my title", "description":"my body"}'
http://localhost:8080/messages

curl localhost:8080/messages
```

Published port and nftables

```
docker run --rm -p 8080:80 nginx
ps aux | grep docker-proxy

sudo nft list ruleset
sudo nft list table ip nat
curl 10.4.0.XX:8080
sudo nft list table ip nat
```

CGroups and memory limits

docker run -d --name mem-test polinux/stress stress --vm 1 --vm-bytes 15M

docker run -d --name mem-test -m 10mb polinux/stress stress --vm 1 --vm-bytes 15M

docker rm -f mem-test
docker run -d --name mem-test -m 10mb polinux/stress stress --vm 1 --vm-bytes 8M

docker stats mem-test

mount | grep cgrou

find /sys/fs/cgroup -name *ContainerID*

pidof stress

cat /sys/fs/cgroup/system.slice/docker-792c1735f2bd973796846c6b10d137fe6719ec777adffb86a7d56a0ff398253d.scope/memory.max

cat /sys/fs/cgroup/system.slice/docker-792c1735f2bd973796846c6b10d137fe6719ec777adffb86a7d56a0ff398253d.scope/cgroup.procs

Kubernetes

Landscape

https://landscape.cncf.io/

https://sonobuoy.io/

https://k8s.af/

https://kubernetes.io/releases/

kubectl install

mkdir k8s-setup

cd k8s-setup

KUBECTL_VERSION=v1.33.4
curl -LO https://dl.k8s.io/release/$KUBECTL_VERSION/bin/linux/amd64/kubectl

chmod +x kubectl

sudo mv kubectl /usr/local/bin/
kubectl version --client

alias k="kubectl"

source <(kubectl completion bash)
complete -o default -F __start_kubectl k

source ~/.bashrc
k version --client

minikube install

MINIKUBE_VERSION=v1.37.0
curl -LO https://github.com/kubernetes/minikube/releases/download/$MINIKUBE_VERSION/minikube-linux-amd64

chmod +x minikube-linux-amd64
sudo mv minikube-linux-amd64 /usr/local/bin/minikube
minikube status
minikube version

minikube completion bash | sudo tee -a /usr/share/bash-completion/completions/minikube > /dev/null

minikube 3 node cluster

export K8S_VERSION=1.33.4

minikube start --nodes 3 --memory=2Gb --kubernetes-version=$K8S_VERSION

htop
kubectl get nodes
minikube status

docker ps
docker network ls

minikube ssh

Basic Pod/Deployment/Service

mkdir basic
touch pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: basic-nginx
  labels:
    training: devops
spec:
  containers:
    - name: c1
      image: nginx:1.27
```

k create -f pod.yaml
kubectl get pods

kubectl get pods -o wide

k port-forward pods/basic-nginx 8080:80
http://localhost:8080/

touch deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: basic-nginx-ha
spec:
  replicas: 3
  selector:
    matchLabels:
      training: devops
      company: pirates
  template:
     metadata:
        labels:
          training: devops
          company: pirates
          ver: "123"
```

```
    spec:
     containers:
       - name: c1
         image: nginx
```

```
k create -f deployment.yaml
```

```
k get pods -l company=pirates
k get deployments.apps
```

```
k get deployment,replicaset
```

```
k get pod --show-labels
```

```
apiVersion: v1
kind: Service
metadata:
  name: basic-nginx
spec:
  type: NodePort
  selector:
    training: devops
    company: pirates
  ports:
    - port: 8080
      targetPort: 80
      nodePort: 30080
```

```
k create -f service.yaml
```

```
minikube ip
minikube ip -n minikube-m02
```

```
curl 192.168.49.XX:30080
```

```
k logs deployments/basic-nginx-ha --all-pods=true --prefix --since 20s
```

Deployment and rollout strategy

```
apiVersion: apps/v1
kind: Deployment
metadata:
   name: infov6
   annotations:
```

```
    kubernetes.io/change-cause: "upgrade to 6.7.1"
spec:
  replicas: 2
  selector:
    matchLabels:
      tier: frontend
      env: dev
  template: # pod definition goes here
      metadata:
        labels:
          tier: frontend
          env: test # broken
      spec:
        containers:
        - name: info
          image: stefanprodan/podinfo:6.6.3
          ports:
          - containerPort: 9898
        - name: net-tools
          image: nicolaka/netshoot
          command: ["/bin/bash"]
          args: ["-c", "while true; do curl localhost:9898/version; sleep 15;done"]

k create -f deploy.yaml
k get deploy

k rollout status deployment infov6
k rollout history deployment infov6
k get pod -l tier=frontend
k rollout restart deployment infov6

k replace -f deploy.yaml
k rollout status deployment infov6
```

Rollback

```
# add non existing podinfo image version
 k replace -f deploy.yaml
k get deployments.apps
k get pods

k describe pod infov6-59b7f8c6bf-9pvv9

k rollout undo deployment infov6
k get deployments.apps
```

k get pod infov6-6db85895bc-7hhmz -o yaml

```
        image: stefanprodan/podinfo:6.7.1
```

k replace -f deploy.yaml

Config Maps

mkdir cm

k create configmap podinfo-config --from-literal version=v1 --from-literal msg="Welcome to devops"

k get cm podinfo-config -o yaml

touch values.yaml

```
database:
  host: db01.cluster
  tls: disabled
```

touch status.json

```
{
   "level":"Warning"
}
```

k create configmap podinfo-file-config --from-file val.yaml=values.yaml --from-file status.json

k get cm

touch pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: podinfo-cm
spec:
  containers:
    - name: pi
      image: stefanprodan/podinfo:6.7.1
      command:
        - ./podinfo
        - --config-path=/configuration
```

```
    volumeMounts:
      - name: config-files
        mountPath: /configuration
        readOnly: true
    ports:
      - name: http
        containerPort: 9898
        protocol: TCP
    env:
     - name: PODINFO_UI_MESSAGE
       valueFrom:
         configMapKeyRef:
           name: podinfo-config
           key: msg
  volumes:
    - name: config-files
      configMap:
        name: podinfo-file-config
```

```
k create -f pod.yaml

k exec -it pod/podinfo-cm -- ls /configuration

k exec -it pod/podinfo-cm -- cat /configuration/status.json | jq

k port-forward  pod/podinfo-cm 9898:9898
http://localhost:9898/configs
http://localhost:9898/
```

Namespaces

```
k get namespaces
k get pods --namespace default
k get pods

k get pods -n kube-system
k get pods -n kube-system -o wide

k get daemonsets.apps -n kube-system
 k get nodes

 k create ns myapi
kubectl get ns
```

```
k get pods -n myapi

k create deployment mydeploy --image stefanprodan/podinfo --replicas 3
k get deployments.apps

k create deployment mydeploy -n myapi  --image stefanprodan/podinfo --replicas 3
k get pods -n myapi

k create deployment mydeploy -n myapi  --image stefanprodan/podinfo --replicas 3 --dry-run=client

k create deployment mydeploy -n myapi  --image stefanprodan/podinfo --replicas 3 --dry-run=client -oyaml

k describe pod mydeploy-c96f9c65d-dv6fv -n myapi

k edit deployments.apps -n myapi mydeploy
```

Services (ClusterIP service)

```
mkdir svc
k create deployment --image nginx --replicas 2 web01

k delete deployments.apps web01
k expose deployment web01 --port 8080 --target-port 80

k run -it --rm pod-tmp --image nicolaka/netshoot -- sh

nslookup web01
curl web01:8080

curl web01.default.svc.cluster.local.:8080
```

Load balancing test (k6)

```
cat <<EOF | tee k6-clusterip.js
import http from 'k6/http';

export const options = {
  vus: 2, //virtual users run concurrently
  iterations: 100, // number of requests throughtout the test
  duration: '10s', //duration of the test execution
  noConnectionReuse: true, // disable keep-alive connections
};
```

```
export default function () {
  http.get('http://web01:8080');
}
EOF

k run -i --rm --image grafana/k6 k6-test-run -- run - < k6-clusterip.js

k get pods,svc -l app=web01

k logs --selector app=web01 --prefix --timestamps --since 5m --tail 500 | grep pod/web01-7d78477f84-xvgzk/nginx | wc -l

k get pod -l app=web01
```

Secrets and private Docker image repository

```
mkdir secrets
cd secrets

touch deploy.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-v1
spec:
  selector:
    matchLabels:
      app: app-v1
  template:
    metadata:
      labels:
        app: app-v1
    spec:
      containers:
        - name: c1
          image: mulawam/go-app:v2-mm
      imagePullSecrets:
       - name: docker-reg

k create -f deploy.yaml
 k get pod -l app=app-v1

 k describe pod app-v1-6f84598564-c65sp | tail
```

```
docker login -u mulawam
dckr_pat_doQZzGaaECUl2OhjlSXAtdqS47A

 cat ~/.docker/config.json
echo "bXVsYXdhbTpkY2tyX3BhdF9kb1FaekdhYUVDVWwyT2hqbFNYQXRkcVM0N0E=" | base64 -d

kubectl create secret generic docker-reg \
    --from-file=.dockerconfigjson=$HOME/.docker/config.json \
    --type=kubernetes.io/dockerconfigjson

k get secrets
k get secrets docker-reg -oyaml

k replace -f deploy.yaml
k get pods -l app=app-v1
```

https://developer.hashicorp.com/vault/tutorials/kubernetes/kubernetes-raft-deployment-guide
https://github.com/bitnami-labs/sealed-secrets

Storage

```
mkdir storage && cd storage
```

Statefulset and emptyDir

```
echo -n root | base64

touch secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: redis-secret
type: Opaque
data:
  password: cm9vdA==

k create -f secret.yaml

k get secrets
```

```
touch redis.yaml

apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: redis
  labels:
    app: redis
spec:
  serviceName: redis
  selector:
    matchLabels:
      service: redis
  template:
    metadata:
      labels:
        service: redis
    spec:
      containers:
      - name: redis
        image: redis/redis-stack:6.2.6-v19
        ports:
        - containerPort: 6379
          name: redis
        - containerPort: 8001
          name: insights
        env:
        - name: REDIS_PASSWORD
          valueFrom:
            secretKeyRef:
              name: redis-secret
              key: password
        - name: REDIS_ARGS
          value: "--requirepass $(REDIS_PASSWORD)"
        volumeMounts:
        - name: redis-data
          mountPath: /data
      volumes:
      - name: redis-data
        emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: redis
spec:
```

```
  ports:
  - port: 6379
    name: redis
  # this makes its headless and without IP
  clusterIP: None
  selector:
    service: redis
```

```
k create -f redis.yaml

k get pods -l service=redis
k get statefulset

k exec -it redis-0 -- sh

redis-cli --user default --pass root
SAVE

# other terminal
k exec redis-0 -- ls /data
```

Hostpath

```
.....
      volumes:
      - name: redis-data
        hostPath:
          type: DirectoryOrCreate
          path: /tmp/redis-data

k replace -f redis.yaml
k get pods -l service=redis

 k exec redis-0 -- redis-cli --user default --pass root SAVE

 k get pods -l service=redis -owide
minikube ssh -n minikube-m03

ls /tmp/redis-data/ -la
```

StorageClass, PV and PVC on minikube

```
 k get pods -A
k get pods -n kube-system
k get pods -A | grep storage

k get storageclass

k get pvc,pv

k get persistentvolume,persistentvolumeclaims

touch pvc.yaml

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: redis-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: standard
  resources:
    requests:
       storage: 1Gi

k create -f pvc.yaml

k get pvc,pv
```

https://kubernetes.io/docs/concepts/storage/storage-classes/

```
...
      volumes:
      - name: redis-data
        persistentVolumeClaim:
          claimName: redis-pvc

k replace -f redis.yaml
k get pods -l service=redis
k exec redis-0 -- redis-cli --user default --pass root SAVE

k describe persistentvolume/pvc-6bdd2b06-b4f9-4e36-b50f-af042fb94c1f

minikube ssh -n minikube-m03
k delete -f redis.yaml
k get pods -l service=redis
```

k get pv,pvc

Ingress Gateway API

mkdir ingress && cd ingress

minikube cluster with Calico CNI

minikube delete
docker network rm minikube

export K8S_VERSION=1.33.4

minikube start --memory=2Gb --kubernetes-version=$K8S_VERSION --cni=false --network-plugin=cni --extra-config=kubeadm.pod-network-cidr=192.168.0.0/16 --subnet=172.16.0.0/24

k get pods -A

export CALICO_VERSION=3.30.2
k create -f https://raw.githubusercontent.com/projectcalico/calico/v$CALICO_VERSION/manifests/operator-crds.yaml

k api-resources | grep calico

k create -f https://raw.githubusercontent.com/projectcalico/calico/v$CALICO_VERSION/manifests/tigera-operator.yaml

k get ns

k get pod -n tigera-operator

k create -f https://raw.githubusercontent.com/projectcalico/calico/v$CALICO_VERSION/manifests/custom-resources.yaml

watch kubectl get tigerastatus

k rollout restart deployment -n kube-system coredns

k get nodes

minikube node add
minikube node add

k get nodes

touch app.yaml

Service (type LoadBalancer) with Metallb

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-multi
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      tier: frontend
  template:
    metadata:
      labels:
        app: nginx
        tier: frontend
    spec:
      containers:
        - name: ng
          image: nginx
---
apiVersion: v1
kind: Service
metadata:
  name: lb-svc
spec:
  type: LoadBalancer
  selector:
     app: nginx
     tier: frontend
  ports:
    - targetPort: 80
      port: 8081
```

```
 k create -f app.yam
 k get svc
```

```
 minikube addons enable metallb
```

k get all -n metallb-system

172.16.0.0/24

k get cm -n metallb-system config -o yaml > metallb-config.yaml

```
apiVersion: v1
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 172.16.0.100-172.16.0.120
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"config":"address-pools:\n- name: default\n  protocol: layer2\n  addresses:\n
- -\n"},"kind":"ConfigMap","metadata":{"annotations":{},"name":"config","namespace":"metallb-system"}}
  creationTimestamp: "2025-11-06T15:56:11Z"
  name: config
  namespace: metallb-system
  resourceVersion: "3333"
  uid: b1615b91-c7e8-42d3-8da8-810c07aa0cfe
```

k replace -f metallb-config.yaml

k rollout restart deployment -n metallb-system controller

curl 172.16.0.100:8081

Calico Gateway API (Ingress)

minikube status
minikube start

k get nodes

k rollout restart daemonset -n calico-system calico-node

k get pods -A -owide

minikube node delete minikube-m0X

```
k get pods -A -owide | grep m03

touch tigera-gateway.yaml

apiVersion: operator.tigera.io/v1
kind: GatewayAPI
metadata:
 name: tigera-secure

 k create -f tigera-gateway.yaml

 watch kubectl get tigerastatus

  k get ns

  k get all -n tigera-gateway

  k get pod -A -owide

  touch podinfo-gateway.yaml

apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: podinfo-gateway
  namespace: default
spec:
  gatewayClassName: tigera-gateway-class
  listeners:
  - name: http
    port: 80
    protocol: HTTP
    hostname: podinfo.public
  - name: https
    port: 443
    protocol: HTTPS
    hostname: podinfo.public
    tls:
      mode: Terminate
      certificateRefs:
      - name: podinfo-public-tls
        kind: Secret

sudo apt install mkcert libnss3-tools -y
mkcert --install
```

mkcert podinfo.public

k create secret tls podinfo-public-tls --cert ./podinfo.public.pem --key podinfo.public-key.pem
k get secrets

k create -f podinfo-gateway.yaml

k get all -n tigera-gateway

k replace --force -f metallb-config.yaml

touch podinfo-app.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pod-info-public
spec:
  replicas: 3
  selector:
    matchLabels:
      component: pod
      app: info-public
  template:
    metadata:
      labels:
        component: pod
        app: info-public
    spec:
      containers:
        - name: c1
          image: stefanprodan/podinfo
---
apiVersion: v1
kind: Service
metadata:
    name: podinfo-public-svc
spec:
  selector:
    component: pod
    app: info-public
  ports:
    - name: http
      port: 8080
      targetPort: 9898
```

k create -f podinfo-app.yaml

k get pods

touch http-route.yaml

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: podinfo-http-route
  namespace: default
spec:
  parentRefs:
  - name: podinfo-gateway
    namespace: default
    port: 80
    sectionName: http
  hostnames:
  - podinfo.public
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /
    backendRefs:
    - name: podinfo-public-svc
      port: 8080
```

k get pod -n metallb-system
k get gateway

curl -v -H "Host: podinfo.public" http://172.16.0.101

touch https-route.yaml

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: podinfo-route
  namespace: default
spec:
  parentRefs:
  - name: podinfo-gateway
    namespace: default
    port: 443
    sectionName: https
```

```
  hostnames:
  - podinfo.public
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /
    backendRefs:
    - name: podinfo-public-svc
      port: 8080
```

```
k create -f https-route.yaml

echo "172.16.0.101 podinfo.public" | sudo tee -a /etc/hosts

curl https://podinfo.public

k get httproute
k get pods -n tigera-gateway

curl https://podinfo.public
```

https://kind.sigs.k8s.io/
https://github.com/kelseyhightower/kubernetes-the-hard-way

https://github.com/kubernetes-sigs/kubespray

```
export CP="10.4.0.XXX" # control plane
export W1="10.4.0.XXX" # worker 1
export W2="10.4.0.XXX" # worker 2

source ~/.bashrc

echo $CP $W1 $W2

cd

ssh-keygen

for ip in $CP $W1 $W2; do
  ssh-copy-id -o StrictHostKeyChecking=accept-new $ip
done

for ip in $CP $W1 $W2; do
  ssh ${ip} hostname
```

```
  ssh ${ip} ip addr show dev eth0
done

ssh $CP

sudo ufw status

swapon --show
sudo sysctl net.ipv4.ip_forward
sudo sysctl net.ipv4.ip_forward=1
sudo sysctl net.ipv4.ip_forward

echo "net.ipv4.ip_forward = 1" | sudo tee /etc/sysctl.d/k8s.conf

sudo modprobe overlay

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
EOF

sudo apt-get update

sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

{
   echo \
   "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
   $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
   sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
}
sudo apt-get update

sudo apt-get -y install containerd.io

containerd config default

containerd config default | sudo tee /etc/containerd/config.toml

cat /etc/containerd/config.toml

sudo sed -i "s/SystemdCgroup = false/SystemdCgroup = true/g" /etc/containerd/config.toml

cat /etc/containerd/config.toml | grep Cgroup
```

```
sudo systemctl restart containerd
sudo systemctl status containerd

export K8S_MINOR_VERSION=1.33
curl -fsSL https://pkgs.k8s.io/core:/stable:/v$K8S_MINOR_VERSION/deb/Release.key | sudo gpg --dearmor
-o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v$
{K8S_MINOR_VERSION}/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list

https://semver.org/

sudo apt-get update

sudo apt-cache madison kubeadm

export K8S_VERSION=1.33.4
sudo apt-get install -y kubelet=$K8S_VERSION-1.1 kubeadm=$K8S_VERSION-1.1
kubectl=$K8S_VERSION-1.1

sudo apt-mark hold kubelet kubeadm kubectl

sudo kubeadm init phase preflight

sudo sed -i "s/pause:3.8/pause:3.10/g" /etc/containerd/config.toml

sudo systemctl restart containerd
sudo systemctl status containerd

cat <<EOF | tee kubeadm.config
apiVersion: kubeadm.k8s.io/v1beta4
kind: ClusterConfiguration
kubernetesVersion: v1.33.4
apiServer:
  extraArgs:
  - name: advertise-address
    value: ${CP}
networking:
  podSubnet: 192.168.0.0/16
  serviceSubnet: 10.96.0.0/12
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
mode: nftables
EOF
```

```
rm kubeadm.config
exit

hostname
echo $CP

scp ./kubeadm.config $CP:/home/kurs
ssh $CP ls /home/kurs

ssh $CP

sudo kubeadm init --config kubeadm.config --dry-run | more

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

  kubectl get nodes

kubectl describe node 00control

kubectl get pod -A -o wide

scp $CP:/home/kurs/.kube/config $HOME/.kube/config

cat ~/.kube/config

k get nodes

mkdir setup-calico && cd setup-calico

export CALICO_VERSION=3.30.2
kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v$CALICO_VERSION/manifests/operator-crds.yaml
kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v$CALICO_VERSION/manifests/tigera-operator.yaml

k get -n tigera-operator all

wget https://raw.githubusercontent.com/projectcalico/calico/v$CALICO_VERSION/manifests/custom-resources.yaml
cat custom-resources.yaml

sed -i "s/VXLANCrossSubnet/VXLAN/g" ./custom-resources.yaml
```

```
sed -i '/calicoNetwork:/a\ \ \ \ linuxDataplane: Nftables' ./custom-resources.yaml

k create -f custom-resources.yaml

k get nodes
kubectl get tigerastatus
kubectl get nodes
```

Worker nodes setup

```
tee depend.sh > /dev/null <<'DEPS'
#!/bin/bash
set -euo pipefail

export DEBIAN_FRONTEND=noninteractive

sudo sysctl net.ipv4.ip_forward=1

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.ipv4.ip_forward = 1
EOF

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
EOF

sudo modprobe overlay

sudo apt-get update
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

{
   echo \
   "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
   $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
   sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
}

sudo apt-get update
sudo apt-get -y install containerd.io
```

```
sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml

sudo sed -i "s/SystemdCgroup = false/SystemdCgroup = true/g" /etc/containerd/config.toml
sudo sed -i "s/pause:3.8/pause:3.10/g" /etc/containerd/config.toml

sudo systemctl restart containerd

export K8S_MINOR_VERSION=1.33 # K8S_MINOR_VERSION tag

curl -fsSL https://pkgs.k8s.io/core:/stable:/v$K8S_MINOR_VERSION/deb/Release.key | sudo gpg --dearmor
--yes --batch -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v$
{K8S_MINOR_VERSION}/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

export K8S_VERSION=1.33.4 # K8S_VERSION tag
sudo apt-get install -y kubelet=$K8S_VERSION-1.1 kubeadm=$K8S_VERSION-1.1
kubectl=$K8S_VERSION-1.1
sudo apt-mark hold kubelet kubeadm kubectl


kubectl version --client=true
kubeadm version
kubelet --version
DEPS

for host in $W1 $W2; do
  scp ./depend.sh $host:./depend.sh | tee -a deps-install.log
  echo "Running depend.sh on $host ..." | tee -a deps-install.log
  ssh $host ./depend.sh | tee -a deps-install.log
done

# check the log output
cat deps-install.log | grep -E "(kubeadm|kubelet|10\.)"

ssh $W1 kubeadm version
ssh $W2 kubeadm version

ssh $CP sudo kubeadm token create --print-join-command

JOIN_CMD=$(ssh $CP sudo kubeadm token create --print-join-command)
echo $JOIN_CMD
```

```
for worker in $W1 $W2; do
  echo "joining $worker node to k8s cluster" | tee -a join-cluster.log
  ssh $worker sudo $JOIN_CMD | tee -a join-cluster.log
done

k get nodes

sudo apt-get install curl gpg apt-transport-https --yes
curl -fsSL https://packages.buildkite.com/helm-linux/helm-debian/gpgkey | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/helm.gpg] https://packages.buildkite.com/helm-linux/helm-debian/any/ any main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list
sudo apt-get update
sudo apt-get install helm

helm version

https://k9scli.io/
```

Kubernetes Dashboard

```
helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/

helm install dashboard kubernetes-dashboard/kubernetes-dashboard --create-namespace --namespace kubernetes-dashboard

helm list -A

k get all -n kubernetes-dashboard

k port-forward -n kubernetes-dashboard svc/dashboard-kong-proxy 8443:443

https://localhost:8443/


k get serviceaccounts -n tigera-operator
k get clusterrole | grep admin

touch rbac.yaml

# Service Account with the name admin-user in namespace kubernetes-dashboard first
apiVersion: v1
kind: ServiceAccount
metadata:
```

```
  name: admin-user
  namespace: kubernetes-dashboard
---
# In most cases after provisioning the cluster using kops, kubeadm or any other popular tool, the ClusterRole cluster-admin already exists in the cluster.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

k create -f rbac.yaml

k get sa -n kubernetes-dashboard

k create -n kubernetes-dashboard token admin-user

fg
ctrl+C

Dynamic storage (NFS server)

NFS Server on jumpbox

```
sudo apt install -y nfs-kernel-server
sudo systemctl status nfs-kernel-server
sudo mkdir -p /nfs
sudo chown nobody:nogroup /nfs

echo "/nfs 10.4.0.0/24(rw)" | sudo tee -a /etc/exports
sudo systemctl restart nfs-kernel-server
```

NFS client on workers

```
for host in $CP $W1 $W2; do
  echo "Installing NFS client sh on $host ..." | tee -a nfs-install.log
  ssh $host sudo apt-get install -y nfs-common | tee -a nfs-install.log
done
```

NFS Provider installed through helm chart

https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner/blob/master/charts/nfs-subdir-external-provisioner/README.md

ip addr show dev eth0

https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner/blob/master/charts/nfs-subdir-external-provisioner/values.yaml#L10C1-L12C21

helm repo add nfs-subdir-external-provisioner https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner

```
helm install nfs-subdir-external-provisioner nfs-subdir-external-provisioner/nfs-subdir-external-provisioner \
  --create-namespace \
  --namespace nfs-provisioner \
  --set nfs.server=10.4.0.XXX \
  --set nfs.path=/nfs/
```

helm list -A

k rollout restart deployment nfs-subdir-external-provisioner -n nfs-provisioner

k get storageclass

kubectl patch storageclass nfs-client -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": "true"}}}'

```
k get storageclass
k get all -n nfs-provisioner
```

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: redis
  labels:
    app: redis
spec:
  serviceName: redis
```

```
  selector:
    matchLabels:
      service: redis
  template:
    metadata:
      labels:
        service: redis
    spec:
      containers:
      - name: redis
        image: redis/redis-stack:6.2.6-v19
        ports:
        - containerPort: 6379
          name: redis
        - containerPort: 8001
          name: insights
        env:
        - name: REDIS_PASSWORD
          valueFrom:
            secretKeyRef:
              name: redis-secret
              key: password
        - name: REDIS_ARGS
          value: "--requirepass $(REDIS_PASSWORD)"
        volumeMounts:
        - name: redis-data
          mountPath: /data
      volumes:
      - name: redis-data
  volumeClaimTemplates:
    - metadata:
        name: redis-data
      spec:
        accessModes: [ "ReadWriteOnce" ]
        storageClassName: "nfs-client"
        resources:
          requests:
            storage: 1Gi
---
apiVersion: v1
kind: Service
metadata:
  name: redis
spec:
  ports:
  - port: 6379
```

```
    name: redis
  # this makes its headless and without IP
  clusterIP: None
  selector:
    service: redis
```

k create -f secret.yaml -f redis.yaml

k get pvc,pv

k exec redis-0 -- redis-cli --user default --pass root SAVE

tree /nfs

https://github.com/gimlet-io/onechart

Network Policies

CNI

https://github.com/flannel-io/flannel
https://cilium.io/use-cases/cni/

Policies

mkdir policies
cd policies

touch manifest.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: experiments
  labels:
    experiment: v1
---
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: pod-info
  namespace: experiments
spec:
  replicas: 2
  selector:
    matchLabels:
      component: pod
      app: info
  template:
    metadata:
      labels:
        component: pod
        app: info
    spec:
      containers:
        - name: c1
          image: stefanprodan/podinfo
---
apiVersion: v1
kind: Service
metadata:
    name: podinfo-svc
    namespace: experiments
spec:
  selector:
    component: pod
    app: info
  ports:
    - name: http
      port: 9898
      targetPort: 9898

k create -f manifest.yaml

k get all -n experiments

k run -n default -it --rm test-default --image nicolaka/netshoot

curl podinfo-svc
curl podinfo-svc.experiments:9898

cat /etc/resolv.conf
nslookup podinfo-svc.experiments

curl podinfo-svc.experiments.svc.cluster.local.:9898
```

k run -it --rm test-exper -n experiments --image nicolaka/netshoot -- sh
curl podinfo-svc:9898

touch ingress-policy.yaml

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: only-experiments
  namespace: experiments
spec:
  podSelector:
    matchLabels:
      app: info
      component: pod
  policyTypes:
    - Ingress
```

k get networkpolicies -n experiments

k describe networkpolicies/only-experiments -n experiments

k -n calico-system port-forward svc/whisker 8081:8081 &

http://localhost:8081/flow-logs/denied-flows

curl --connect-timeout 2s podinfo-svc:9898

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: only-experiments
  namespace: experiments
spec:
  podSelector:
    matchLabels:
      app: info
      component: pod
  policyTypes:
    - Ingress
  ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            experiment: default
```

k label namespaces default experiment=defaul
k get ns default --show-labels

k replace -f ingress-policy.yaml

curl podinfo-svc.experiments.svc.cluster.local.:9898
curl --connect-timeout 2s podinfo-svc:9898

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: only-experiments
  namespace: experiments
spec:
  podSelector:
    matchLabels:
      app: info
      component: pod
  policyTypes:
    - Ingress
  ingress:
    - from:
      - namespaceSelector:
          matchExpressions:
           - key: experiment
             operator: In
             values: ["default","v1"]
          # matchLabels:
          #   experiment: default
```

curl podinfo-svc.experiments.svc.cluster.local.:9898
curl --connect-timeout 2s podinfo-svc:9898

https://docs.tigera.io/calico/latest/network-policy/encrypt-cluster-pod-traffic