CloudNative **Aalborg** presents

Introduction to Kubernetes

by Henrik Høegh - Praqma

PRAQMA
Now part of Eficode

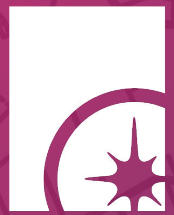
Automating Kubernetes with fluxCD

followed by

Observability using Prometheus, Grafana Loki and Jaeger

by Arne Mejlholm - Sparnord

Hosted by



spar nord

CLOUD NATIVE
COMPUTING FOUNDATION

CloudNative **Aalborg**

#CloudNativeAalborg
#CloudNativeNordics

Join Slack  @ https://www.cloudnativenordics.com/

CLOUD NATIVE
COMPUTING FOUNDATION

Cloud Native **Aalborg** *presents*

Introduction to Kubernetes

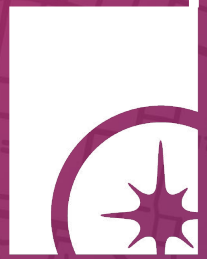
and

Automating installation of Kubernetes

and

Observability

A word from tonight's sponsor



Food

/sponsored by

spar NORD

CLOUD NATIVE
COMPUTING FOUNDATION

Do you want to host a Meetup?

Arne Mejlholm
IT Developer
Sparnord

Simon Bengtsson
Systems Engineer
Sparnord

Camilla Beck Larsen
Systems Engineer
Sparnord

Allan Højgaard Jensen
Cloud Platform Architect
Oracle

come talk to us...

Do you want to speak at a Meetup?

Arne Mejlholm
IT Developer
Sparnord

Simon Bengtsson
Systems Engineer
Sparnord

Camilla Beck Larsen
Systems Engineer
Sparnord

Allan Højgaard Jensen
Cloud Platform Architect
Oracle

come talk to us...

Cloud Native Aalborg

We need your feedback...

- what can we do better?
- any ideas for future meetups?

- provide feedback on: 

Tweet! Tweet!

#CloudNativeAalborg
#CloudNativeNordics

Make some noise about our awesome community… @CloudNativeFdn, #kubernetes, etc.

CLOUD NATIVE
COMPUTING FOUNDATION

Cloud Native Nordics Community

CLOUD NATIVE
NORDICS

CLOUD NATIVE
NORDICS

Continue the discussions and meet Cloud Natives from Denmark, Sweden, Norway, and Finland

www.cloudnativenordics.com

Sponsored by

Introduction to Kubernetes

/by Henrik Høegh
DevOps Consutant Praqma

CLOUD NATIVE
COMPUTING FOUNDATION


PRAQMA

PRAQMA

An Introduction to Kubernetes

(Greek (Κυβερνήτης)-> kee-ver-NEE-tees)

About me

Name: Henrik Høegh
Twitter: @Hoeghh

Doing:

- CLOUD NATIVE
- ATLASSIAN

Contact: heh@praqma.net

PRAQMA
Now part of Eficode

CODE-CONF

Continuous Delivery and DevOps Conference

28 OCT 2019
Copenhagen

HENRIK HØEGH

Storage Operator in Kubernetes - Getting your Feet Wet

DevOps Consultant, Praqma

With Kubernetes becoming the standard way of orchestrating containers, we still need to solve the problem of on-prem storage. Storage and state is one of the most challenging tasks to solve when operating in an on-prem Cloud Native environment. Traditionally companies has been using external storage line NFS, but there are certain problems with this approach. In this talk Henrik will talk about the problems of external storage, how to move storage inside the cluster and automate it with the Rook operator, followed by a live demo. By the end of the talk, the listener will be able to get started with Cloud Native storage in Kubernetes and how to take advantage and flexibility Rook provides.

Twitter: @HenrikHoegh GitHub: hoeghh Blog: Praqma Stories

Read less

ceph

ROOK

Projects I'm working on

https://www.praqma.com/products/ask/

Agenda

Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Why Kubernetes

- **History of scaling**
  - One process <-> one core
  - One process <-> multi cores
  - One process, many threads <-> multi core
- **Future of scaling**
  - Monolith <-> multi machines
  - Horizontal scaling
  - Microservices

Agenda

Why Kubernetes

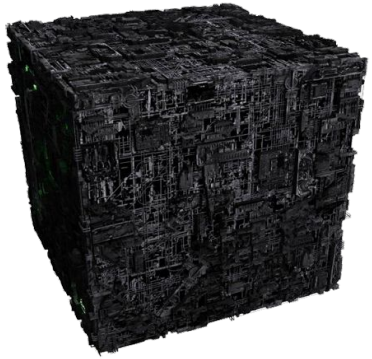
Shoulders of giants

Building blocks

Cluster architecture

Shoulders of giants

Google Borg and **Docker**.



docker

Agenda

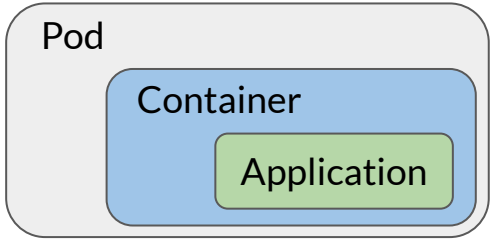
Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Pods




```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'Application']
```

PRAQMA

https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/

Agenda

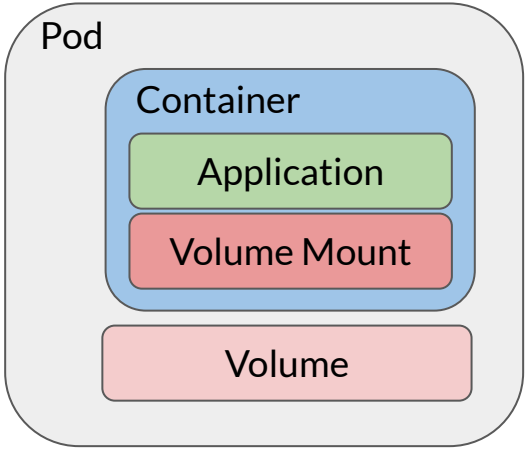
Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Volumes



https://kubernetes.io/docs/concepts/storage/volumes

Agenda

Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Volumes

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'Application']
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

https://kubernetes.io/docs/concepts/storage/volumes

Agenda

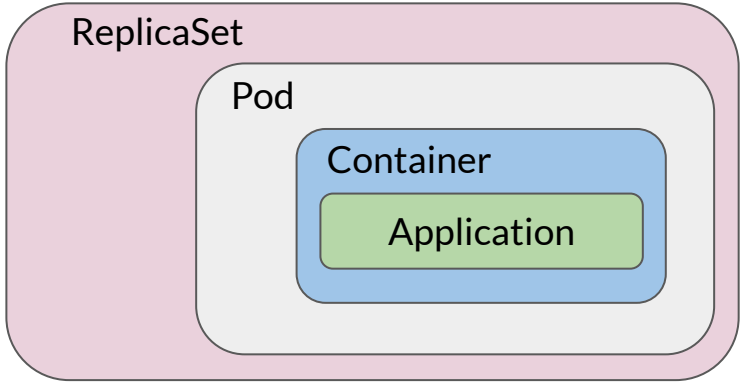
Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - ReplicaSets



https://kubernetes.io/docs/concepts/workloads/controllers/replicaset

Agenda

- Why Kubernetes
- Shoulders of giants
- **Building blocks**
- Cluster architecture

Building blocks - ReplicaSets

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-pod
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp-pod
        image: busybox
```

https://kubernetes.io/docs/concepts/workloads/controllers/replicaset

Agenda

Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - ReplicaSets

```
kubectl get replicaset
NAME        DESIRED   CURRENT   READY   AGE
Myapp-pod   3         0         0       6s

kubectl get replicaset
NAME        DESIRED   CURRENT   READY   AGE
Myapp-pod   3         3         3       18s

kubectl scale --replicas=5 rs/Myapp-pod
```

https://kubernetes.io/docs/concepts/workloads/controllers/replicaset

Agenda

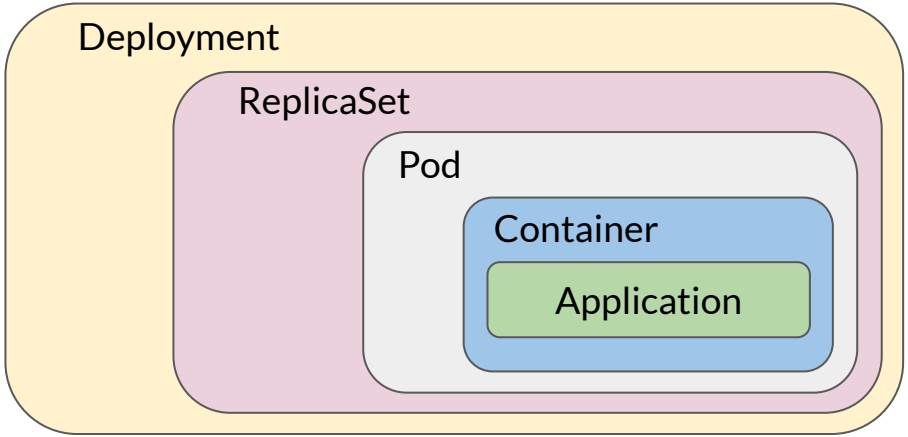
Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Deployment



https://kubernetes.io/docs/concepts/workloads/controllers/deployment

Agenda

- Why Kubernetes
- Shoulders of giants
- **Building blocks**
- Cluster architecture

Building blocks - Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  replicas: 3
 strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 2 # Max added
      maxUnavailable: 0
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp
        image: busybox
```

https://kubernetes.io/docs/concepts/workloads/controllers/deployment

PRAQMA

Agenda

Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Deployment

```
kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
Myapp-pod           3         0         0            0           1s

kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
Myapp-pod           3         3         3            3           18s
```

Updating an image version / tag:

```
kubectl --record \
    set image deployment Myapp-pod \
    myapp-container=busybox:1.30.1

kubectl edit deployment Myapp-pod
```

PRAQMA

https://kubernetes.io/docs/concepts/workloads/controllers/deployment

Agenda

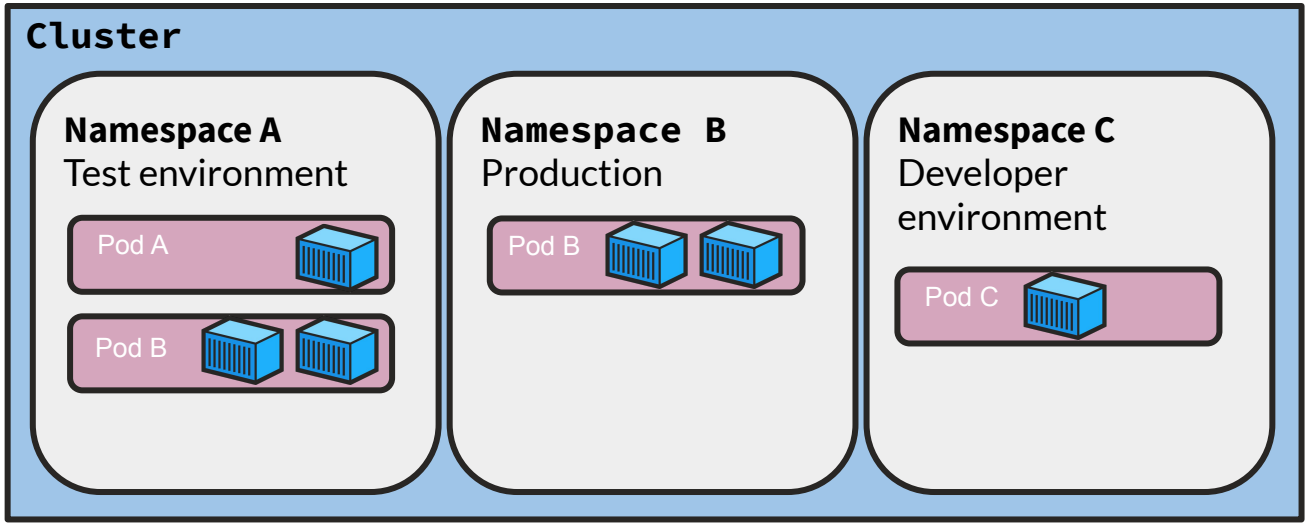
Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Namespaces



PRAQMA

https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

Agenda

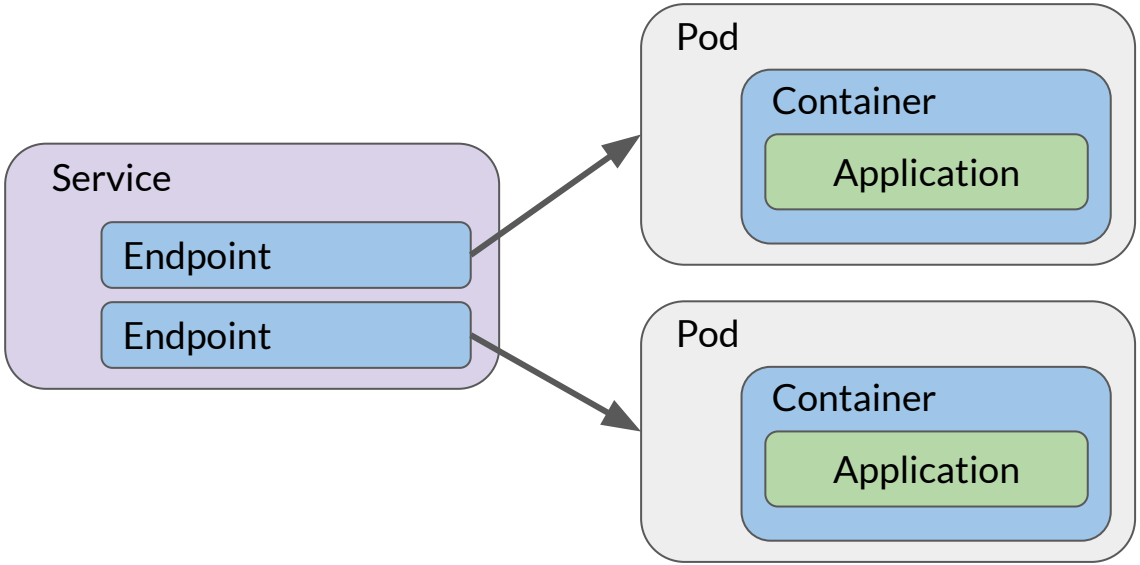
Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Service



https://kubernetes.io/docs/concepts/services-networking/service

Agenda

Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp
        image: busybox
```

PRAQMA

https://kubernetes.io/docs/concepts/services-networking/service

Agenda

Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Service DNS

```
[svc-name].[namespace].svc.cluster.local
```

```
$ curl myapp-svc.prod.svc.cluster.local:80
```

PRAQMA

https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/

Agenda

Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Building blocks - Service types

- ClusterIP (“None”?)
- NodePort
- ExternalName
- LoadBalancer

https://kubernetes.io/docs/concepts/services-networking/service

Agenda

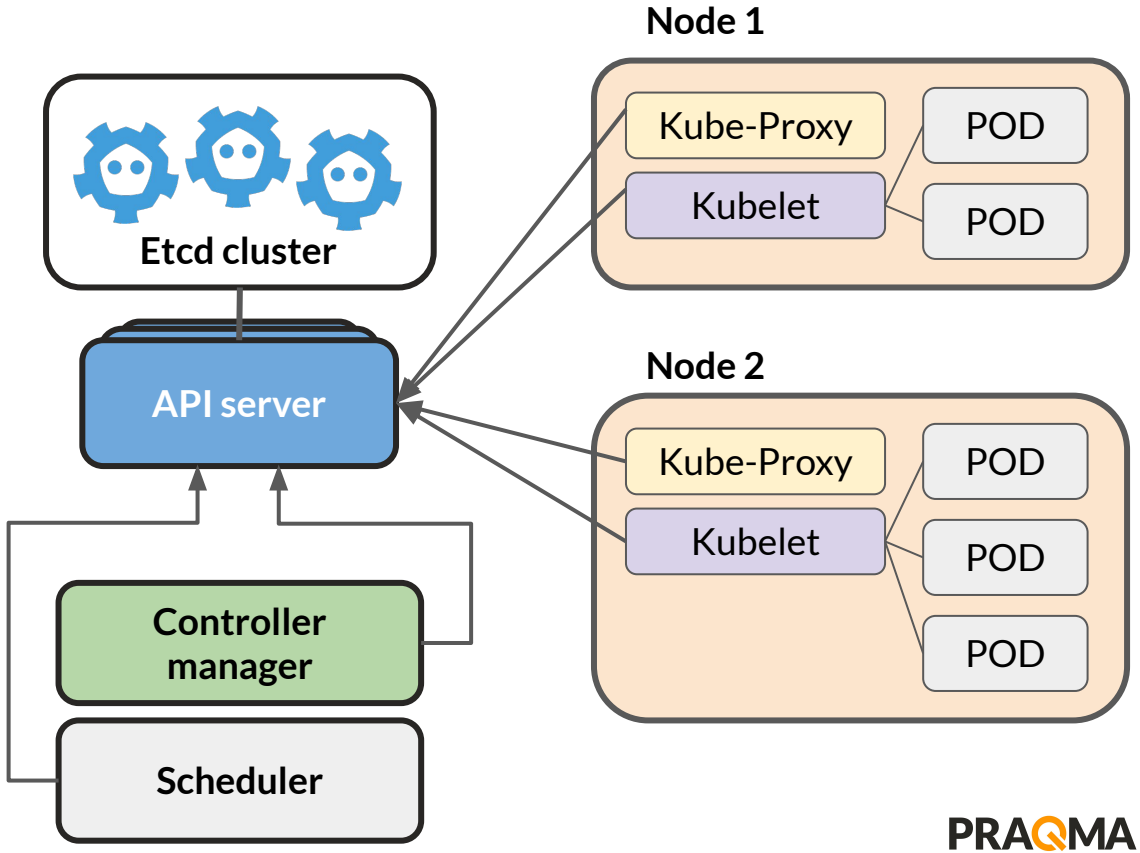
Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Cluster architecture



Agenda

Why Kubernetes

Shoulders of giants

Building blocks

Cluster architecture

Other stuff

Things I didn't cover, that you should look up:

- Configmaps & Secrets
- Persistent Volumes & Claims
- Ingress
- Other controllers e.g.
  - Statefulsets
  - Daemonsets

PRAQMA

Thank you

Remember to tweet!

@HenrikHoegh

Automating Kubernetes
with fluxCD

/by Arne Mejlholm
IT Developer Sparnord

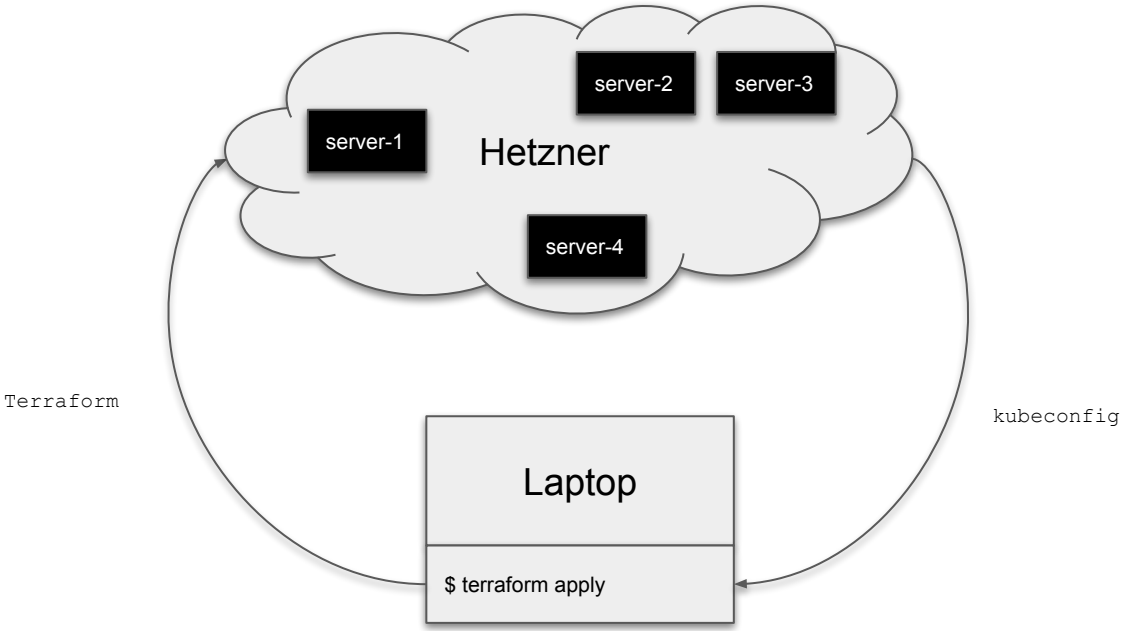
CLOUD NATIVE
COMPUTING FOUNDATION

Infrastructure as code

- Last time we created servers manually
- Sounds more like pets than cattle
- Lets automate that!

Terrraform

- Use Infrastructure as Code to provision and manage any cloud, infrastructure, or service


server-2
server-3
server-1
Hetzner
server-4
Terraform
kubeconfig
Laptop
$ terraform apply

```
############### Variables ################
# Define Hetzner provider
provider "hcloud" {
  token = "<hetzner_cloud_token_goes_here>"
}

# Obtain ssh key data
data "hcloud_ssh_key" "ssh_key" {
  fingerprint = "<ssh_key_goes_here>"
}
```

```
# Create Debian 10 server
resource "hcloud_server" "server-1" {
  name = "server-1"
  image = "debian-10"
  server_type = "cx11"
  ssh_keys  = ["${data.hcloud_ssh_key.ssh_key.id}"]

  connection {
    host = "${hcloud_server.server-1.ipv4_address}"
    type = "ssh"
    user = "root"
    private_key = "${file("~/.ssh/id_rsa")}"
  }
```

```
provisioner "remote-exec" {
  inline = ["# Connected!"]
}

provisioner "local-exec" {
  command = "k3sup install --ip ${hcloud_server.server-1.ipv4_address} && mv kubeconfig ~/.kube/config"
}

provisioner "local-exec" {
  command = "kubectl create ns apps"
}

provisioner "local-exec" {
  command = "kubectl create secret generic wisdom-service-secret -n apps --from-env-file=secret.env"
}
```

```
############## Variables ###############
# Define Hetzner provider
provider "hcloud" {
  token = "<hetzner_cloud_token_goes_here>"
}

# Obtain ssh key data
data "hcloud_ssh_key" "ssh_key" {
  fingerprint = "<ssh_key_goes_here>"
}

# Create Debian 10 server
resource "hcloud_server" "server-1" {
  name = "server-1"
  image = "debian-10"
  server_type = "cx11"
  ssh_keys  = ["${data.hcloud_ssh_key.ssh_key.id}"]

  connection {
    host = "${hcloud_server.server-1.ipv4_address}"
    type = "ssh"
    user = "root"
    private_key = "${file("~/.ssh/id_rsa")}"
  }

  provisioner "remote-exec" {
    inline = ["# Connected!"]
  }

  provisioner "local-exec" {
    command = "k3sup install --ip ${hcloud_server.server-1.ipv4_address} && mv kubeconfig ~/.kube/config"
  }

  provisioner "local-exec" {
    command = "kubectl create ns apps"
  }

  provisioner "local-exec" {
    command = "kubectl create secret generic wisdom-service-secret -n apps --from-env-file=secret.env"
  }
```

GitOps

GitOps is a way to do Kubernetes cluster management and application delivery.

It works by using Git as a single source of truth for declarative infrastructure and applications.

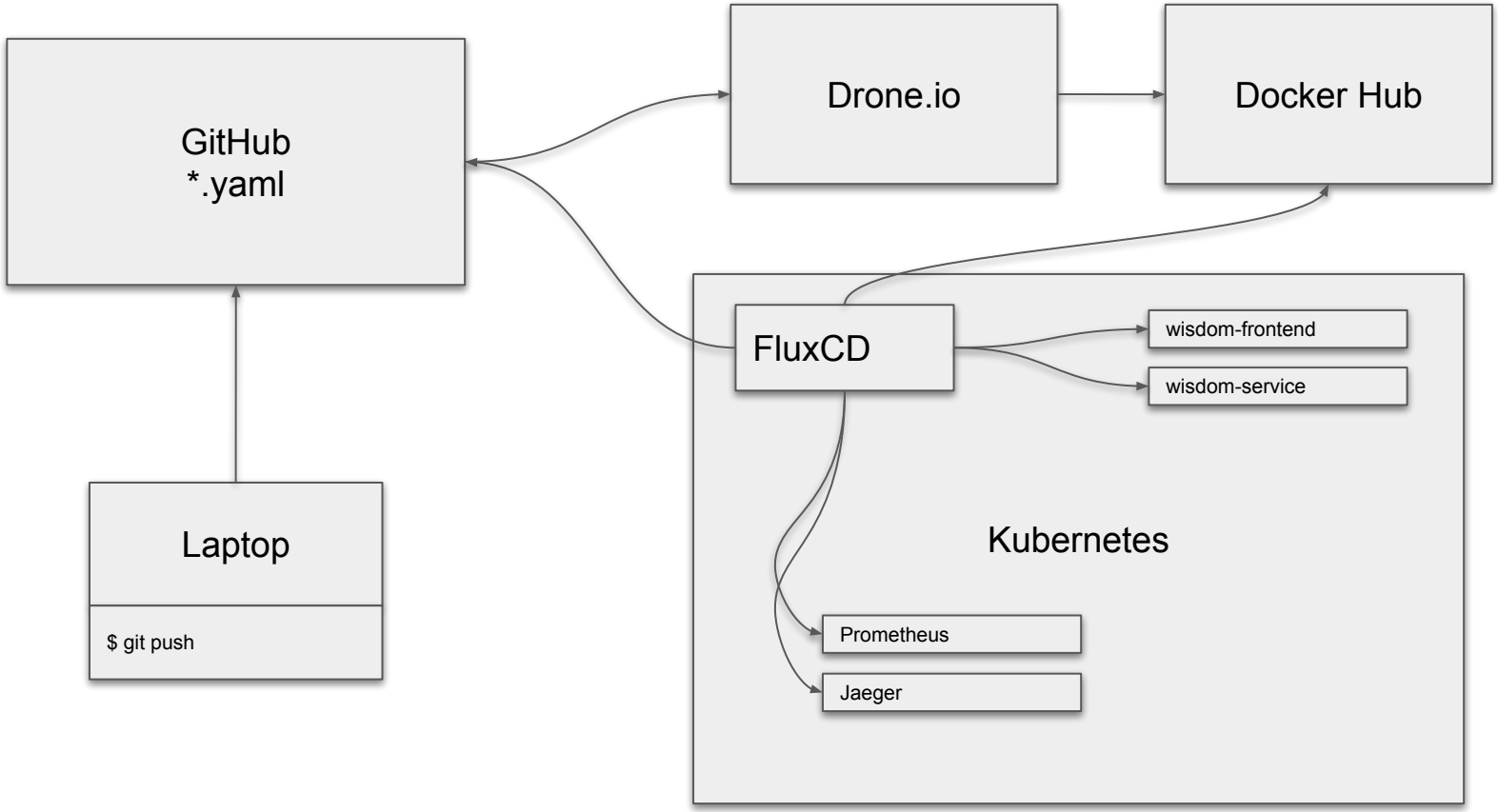
FluxCD

- The GitOps operator for Kubernetes

```
fluxctl install --git-user=mejlholm \
--git-email=mejlholm@users.noreply.github.com \
--git-url=git@github.com:Cloud-Native-Aalborg/Meetup-2 \
--git-paths=k8s,wisdom-frontend/deploy,wisdom-service/deploy
--namespace=flux \
| kubectl apply -f -
```

Drone.io

- Drone is a self-service Continuous Delivery platform for busy development teams.


GitHub
*.yaml
Drone.io
Docker Hub
FluxCD
wisdom-frontend
wisdom-service
Kubernetes
Prometheus
Jaeger
Laptop
$ git push

Lets see that in practice

- Demo time

Observability

with Prometheus, Grafana Loki and Jaeger

/by Arne Mejlholm

IT Developer Sparnord

CLOUD NATIVE COMPUTING FOUNDATION

How are our applications doing?

- What do we do if our applications are misbehaving?

- How many times are our application being used?

Lets install some observability tools:

- Jaeger
- Prometheus
- Grafana Loki

Jaeger

JAEGER

Jaeger: open source, **end-to-end distributed tracing**

Monitor and troubleshoot transactions in complex distributed systems

Prometheus

Power your **metrics** and alerting with a leading open-source monitoring solution.

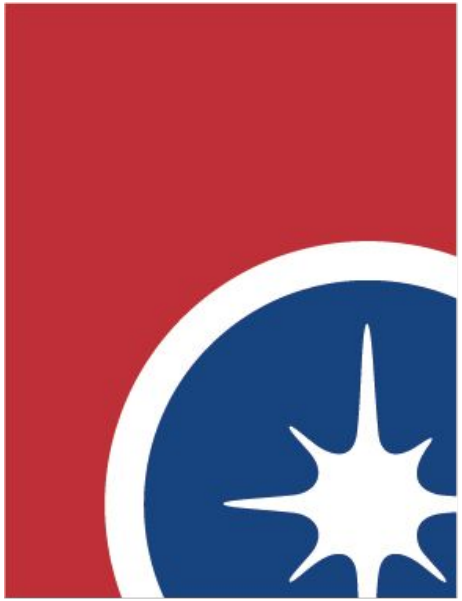
Grafana Loki

Loki is a horizontally-scalable, highly-available, multi-tenant **log aggregation system** inspired by Prometheus.

Lets see that in practice

- Demo time

A big thank you to this evening's host



spar NORD

Networking

/by You

Cloudnatives in Aalborg

CloudNative **Aalborg**

#CloudNativeAalborg
#CloudNativeNordics

Join Slack  @ https://www.cloudnativenordics.com/

CLOUD NATIVE
COMPUTING FOUNDATION