GETTING STARTED WITH GITOPS

RUNE JUHL, ATEA

TABLE OF CONTENTS

- Who am I?
- What is GitOps?
- Technology overview
- Getting our hands dirty
- Job done!

WHO AM I?

- Rune Juhl, open source architect at Atea
- using Linux since late 1990s
- sold my first webapp before I was a teenager
  - ASP 2.0 and a MS Access database
- a geek!
  - `emacs` user
  - loves Clojure (`(str :lisp ["is"] 'awesome)`)
  - prefers functional programming
- crazy about free and open-source software
- likes cooking 🍳, sludgy doom metal 🤘, reading sci-fi 🚀 and hard fiction 🥼

OPEN SOURCE AT ATEA

- a new department in Atea Consulting
- we do “anything open source”, pretty much
- we have our license specialist in the department
  - expertise needed for proper advice and best price
- Red Hat Platinum partner
  - …but we work with Ubuntu and SuSE platforms too

WHAT IS GITOPS?

A guiding set of principles:

- complete system described declaratively
- desired system state is managed through Git
- changes in Git are automatically applied
- `$something` ensures correctness

Sources:

- https://www.weave.works/technologies/gitops/

WHAT'S GITOPS, REALLY?

- automated processes; no manual intervention needed
  - CI/CD pipelines take care of building packages and images
  - a GitOps controller takes care of state and consistency
- configuration is text files
  - generate them!
  - transform them!
  - reuse them!

BENEFITS OF GITOPS

- transparency
  - operations team can see exactly what is deployed, in which versions
- reproducibility
  - easy to spin up a copy of production for e.g. recovery test
- no hidden surprises
  - unmanaged resources can be detected by looking at differences between repo and deployment
- easy to revert
  - broke it all? `git revert HEAD^^` can do the trick

GITOPS VISUALIZED

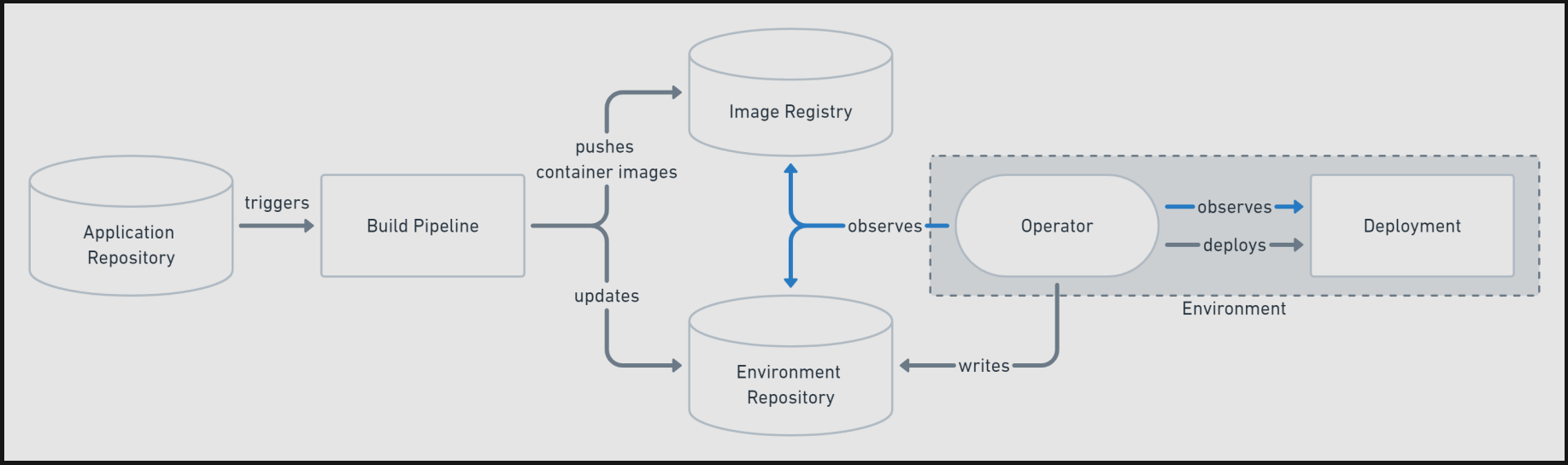


Figure 1: Pull-based deployments. https://www.gitops.tech/

TECHNOLOGY OVERVIEW

- short talk, lots of different pieces
- let’s have an introduction of the technologies we use

WHAT IS DOCKER?

WHAT IS DOCKER?

- a software delivery format
  - a Docker image provides the system libraries required
  - a Docker container is the result when you package an application

Example `Dockerfile`:

```
FROM bitnami/java:17-debian-11
MAINTAINER Rune Juhl Jacobsen <rune.juhl@atea.dk>
COPY ./target/uberjar/awesome-hello-standalone.jar /usr/local/lib/
ENTRYPOINT ["java", "-jar", "/usr/local/lib/awesome-hello-standalone.jar"]
```

WHAT'S THE BENEFIT OF DOCKER?

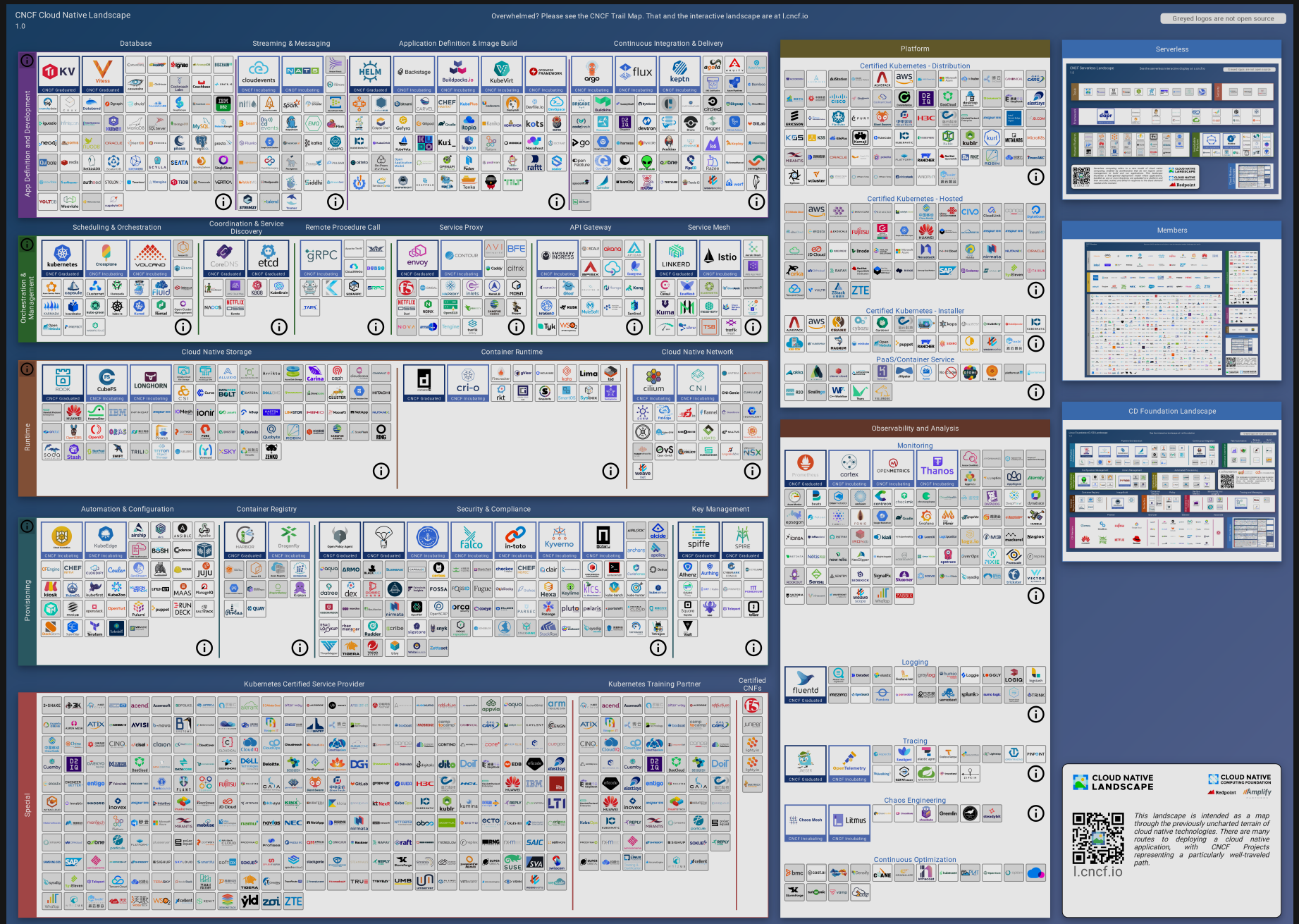
- apps are self-contained
  - containers are packaged with required libraries
- easy to run, no lengthy install processes
- easy to manage and version
- repeatability!

WHAT IS KUBERNETES?

WHAT IS KUBERNETES?

- a container orchestration platform
  - originally a Google project, now managed by *Cloud Native Computing Foundation* (CNCF)
  - takes care of keeping things running
    - e.g. automatic restarts and back-off
- very extensible; can run on anything, anywhere, in any configuration
- everything is configured through a RPC API using JSON

KUBERNETES IS HUGE!


CNCF Cloud Native Landscape
1.0
Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io
Greyed logos are not open source
App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Runtime
Cloud Native Storage
Container Runtime
Cloud Native Network
Provisioning
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Special
Kubernetes Certified Service Provider
Kubernetes Training Partner
Certified CNFs
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Continuous Optimization
Serverless
Members
CD Foundation Landscape
CLOUD NATIVE LANDSCAPE
CLOUD NATIVE COMPUTING FOUNDATION
Redpoint
Amplify
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.
l.cncf.io

MULTIPLE VENDORS AND DISTRIBUTIONS

- Amazon Elastic Kubernetes Service (EKS)
- Azure Kubernetes Service (AKS)
- Google Kubernetes Engine (GKE)
- K3s
- SuSE Rancher
- Red Hat OpenShift
- VMWare Tanzu
- …and more

WHAT IS ARGO-CD?

WHAT IS ARGO-CD?

- a GitOps delivery tool
- ensures that your Kubernetes cluster is kept in line with configuration
- periodically refreshes Git repos used and applies pending changes

GETTING OUR HANDS DIRTY

- it doesn’t have to be (that) hard to get started with GitOps on Kubernetes
- practice makes perfect!
  - breaking stuff in testing is a lot more fun than breaking production

The goal: Getting a simple web application to run in Kubernetes using Argo-CD.

In the following: YAML files are truncated, find complete files at https://github.com/runejuhl

TEST STACK

- K3s is an opinionated Kubernetes distribution
  - slimmed down Kubernetes, removing legacy features + more
  - runs anywhere, including on your laptop and on Raspberry Pi
- Argo-CD is an all-in-one GitOps solution
  - also used in Red Hat OpenShift
- Argo-CD Autopilot is an opinionated way of installing Argo-CD and structuring contents

TEST SETUP

```
# Install K3s
curl -sfL https://get.k3s.io | sh -
# Copy kubectl config to own home
mkdir -p ~/.kube && sudo cat /etc/rancher/k3s/k3s.yaml > ~/.kube/config

# Create a namespace for Argo-CD
kubectl create namespace argocd
# Install Argo-CD
kubectl apply -n argocd -f \
  https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

# Create a Git repo and a personal access token and bootstrap using Argo-CD
# Autopilot
export GIT_TOKEN=github_pat_PcZ...IP0
export GIT_REPO=https://github.com/runejuhl/demo-gitops.git
argocd-autopilot repo bootstrap
```

CREATING A HELM CHART

- Helm is a package manager for Kubernetes
- it provides a way of templating the resources we want to create in Kubernetes
- as always in Kubernetes-land there are alternatives (e.g. kustomize)
- easy to create a skeleton chart:

```
helm create "${chart-name}"
```

TEMPLATING MAKES HELM FLEXIBLE

```
{{- if .Values.ingress.enabled -}}
{{- $fullName := include "chart.fullname" . -}}
{{- $svcPort := .Values.service.port -}}
{{- if and .Values.ingress.className (not
    (semverCompare ">=1.18-0" .Capabilities.KubeVersion.GitVersion)) }}
  {{- if not (hasKey .Values.ingress.annotations "kubernetes.io/ingress.class") }}
  {{- $_ := set .Values.ingress.annotations "kubernetes.io/ingress.class"
            .Values.ingress.className}}
  {{- end }}
{{- end }}
{{- if semverCompare ">=1.19-0" .Capabilities.KubeVersion.GitVersion -}}
apiVersion: networking.k8s.io/v1
{{- else if semverCompare ">=1.14-0" .Capabilities.KubeVersion.GitVersion -}}
apiVersion: networking.k8s.io/v1beta1
```

A MINIMAL APPLICATION

For a minimal application to run a Helm chart needs to define 4 things:

- a **`Chart.yaml` file**
  contains metadata
- a **`Service` resource**
  defines the policies for how Kubernetes can access our application
- an **`Ingress` resource**
  defines how our load balancer should route incoming requests to our application
- a **`Deployment` resource**
  handles running the containers needed, in the amount needed

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: awesome-awesome-hello
  labels: ...
spec:
  type: ClusterIP
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app.kubernetes.io/name: awesome-hello
    app.kubernetes.io/instance: awesome
```

Ingress

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: awesome-awesome-hello
  labels: ...
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
    - host: "awesome-hello.local"
      http:
        paths:
          - path: /
            pathType: ImplementationSpecific
            backend:
              service:
                name: awesome-awesome-hello
                port:
                  number: 80
```

Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: awesome-awesome-hello
  labels: ...
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: awesome-hello
      app.kubernetes.io/instance: awesome
  template:
    metadata:
      labels:
        app.kubernetes.io/name: awesome-hello
        app.kubernetes.io/instance: awesome
    spec:
      containers:
        - name: awesome-hello
          image: "runejuhl/awesome-hello:latest"
          imagePullPolicy: Always
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
          livenessProbe:
            httpGet:
              path: /health
              port: http
          readinessProbe:
```

```
        httpGet:
          path: /health
          port: http
```

A COMPLETE HELM CHART

- chart can be hosted on a plain HTTP server or live in a Git repo

```
$ tree
.
├── awesome-hello
│   ├── Chart.yaml
│   ├── templates
│   │   ├── deployment.yaml
│   │   ├── _helpers.tpl
│   │   ├── ingress.yaml
│   │   ├── NOTES.txt
│   │   └── service.yaml
│   ├── values-production.yaml
│   └── values.yaml
└── wordpress
    ├── Chart.yaml
    └── values-production.yaml
```

CONNECTING THE PIECES

- we need to instruct Argo-CD to handle our application
- everything is templates, so let’s make a template!

ARGO-CD ApplicationSet

```yaml
kind: ApplicationSet
spec:
  generators:
  - git:
      files:
      - path: helm-apps/**/config.yaml
      repoURL: https://github.com/runejuhl/demo-gitops.git
      template:
        metadata: {}
        spec:
          destination: {}
          project: ""
          source:
            repoURL: ""
template:
  spec:
    destination:
      namespace: '{{ destNamespace }}'
      server: '{{ destServer }}'
    project: helm-apps
    source:
      repoURL: https://github.com/runejuhl/demo-charts.git
      targetRevision: HEAD
      path: '{{ chart }}'
      helm:
        valueFiles:
        - 'values-{{ environment }}.yaml'
```

https://github.com/runejuhl/demo-gitops/blob/main/projects/apps.yaml

ADDING THE APPLICATION CONFIG

```yaml
appName: awesome-hello
destNamespace: awesome-hello
destServer: 'https://kubernetes.default.svc'
chart: awesome-hello
environment: production
```

https://github.com/runejuhl/demo-gitops/blob/main/helm-apps/awesome-hello/config.yaml

RESULT


APP HEALTH
Healthy
CURRENT SYNC STATUS
Synced
MORE
To HEAD (fce1697)
Author:
Comment:
Rune Juhl (Atea) <rune.juhl@atea.dk> -
Use upstream registry
LAST SYNC RESULT
Sync OK
MORE
To fce1697
Succeeded 3 hours ago (Wed Nov 02 2022 11:48:23 GMT+0100)
Author:
Comment:
Rune Juhl (Atea) <rune.juhl@atea.dk> -
Use upstream registry
100%
helm-apps-awesome-hello
3 hours
helm-apps-awesome-hello
svc
3 hours
helm-apps-awesome-hello
deploy
3 hours
rev:1
helm-apps-awesome-hello
ing
3 hours
helm-apps-awesome-hello
ep
3 hours
helm-apps-awesome-hello-x4...
endpointslice
3 hours
helm-apps-awesome-hello-66...
rs
3 hours
rev:1
helm-apps-awesome-hello-66...
pod
3 hours
running
1/1
helm-apps-awesome-hello-66...
pod
3 hours
running
1/1
helm-apps-awesome-hello-66...
pod
3 hours
running
1/1

JOB DONE!

Questions? Comments?

Source code:

- https://github.com/runejuhl/awesome-hello
- https://github.com/runejuhl/demo-gitops
- https://github.com/runejuhl/demo-charts

Contact me!

rune.juhl@atea.dk