Kubernetes

Nicole Heinimann
heinimann@informatik.uni-leipzig.de

What is Kubernetes?

Easy!

“Kubernetes (commonly stylized as K8s) is an open-source container orchestration system for automating software deployment, scaling, and management.“

What is Kubernetes?

What is Kubernetes?

✨magic✨

Cloud Native

Docker on Steroids

The thing my SRE friend won’t
shut up about

Very easy and intuitive

PaaS

Way too
complicated

Computing as LEGO

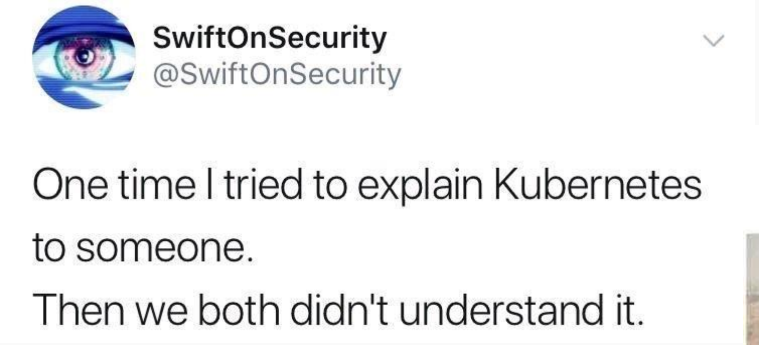
A data center OS

Nightmare fuel

An agnostic platform

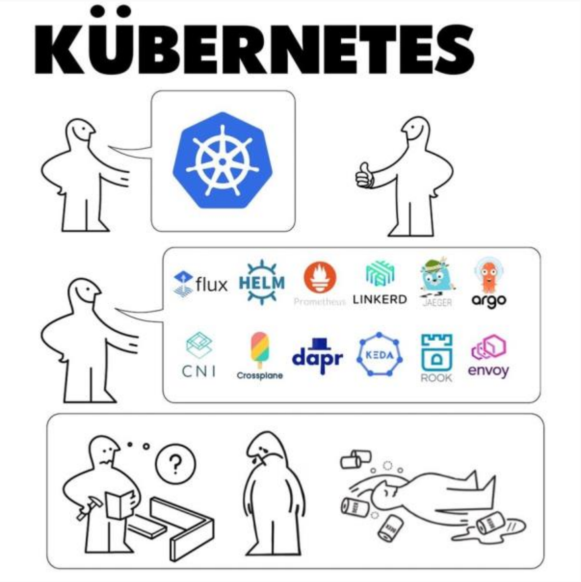
The best thing since
sliced bread

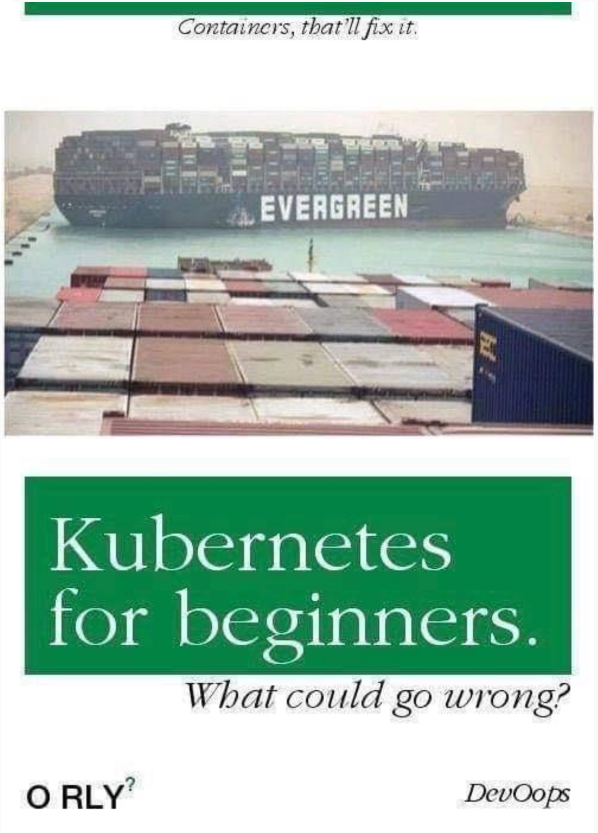
Declarative Deployment



SwiftOnSecurity
@SwiftOnSecurity

One time I tried to explain Kubernetes to someone.
Then we both didn't understand it.





Containers, that'll fix it.

Kubernetes for beginners.

What could go wrong?

O RLY?

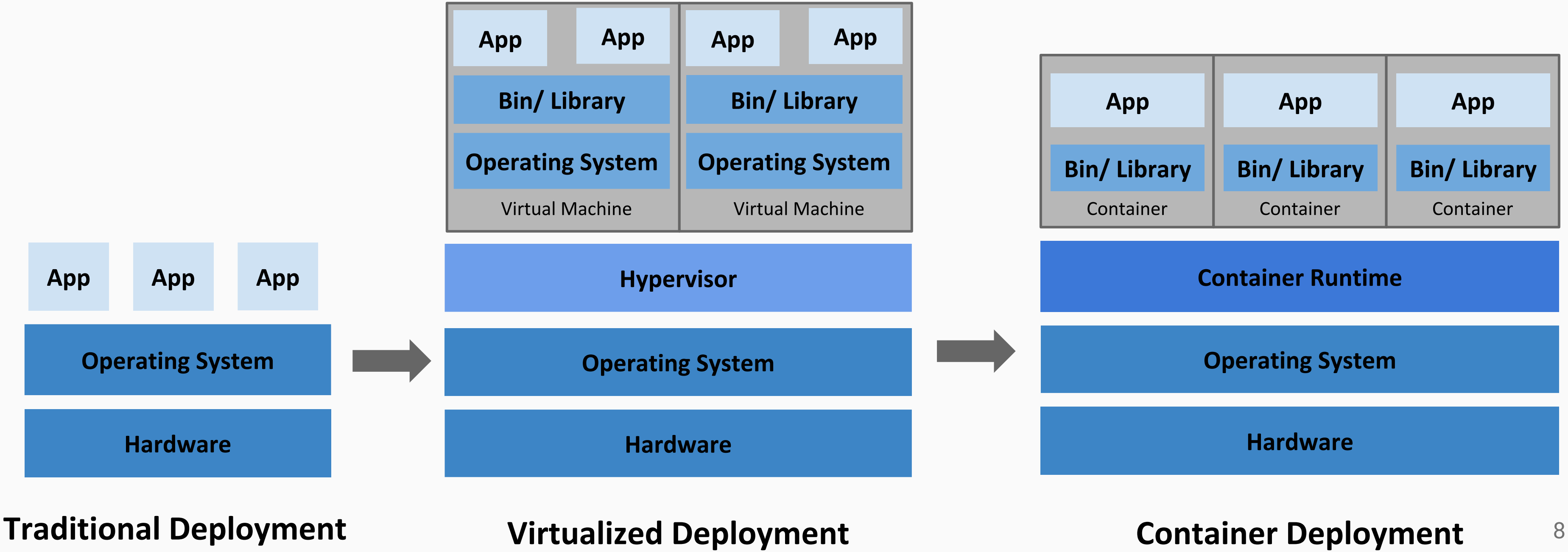
DevOops

ARE Y'ALL WITH THE CULT
WE'RE NOT A CULT
WE BELIEVE YOU CAN DEFINE ANY COMPLEX DEPLOYMENT USING SOLELY YAML AND GITOPS
YEAH, THIS IS IT

That doesn’t seem helpful…

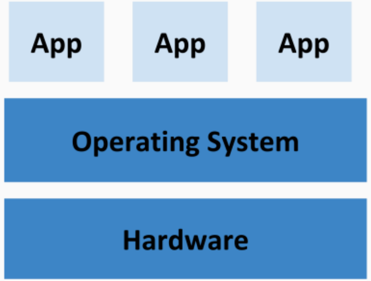
➡ To understand Kubernetes, we first need to understand why it even exists.

A short history of deployment



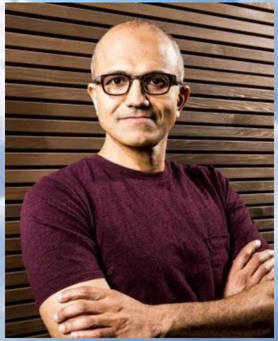
The old days

- Software running directly on servers
- Maintaining whole infrastructure
- “Throw across the wall deployment”
- Lack of automated deployment
- Conflicting software, no isolation
- Language specific tooling required



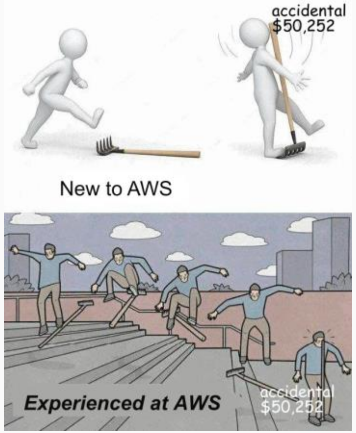
Traditional Deployment

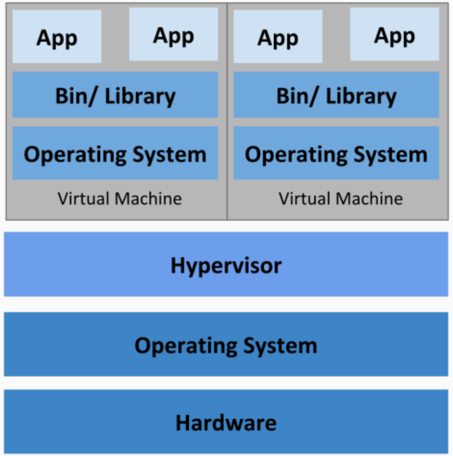
The Cloud



VMs in the Cloud

- Infrastructure as a Service
- Easier Scaling, Resilience and Failover
- Dependency Problems still present
- Inefficient Resource Utilization
- ”AWS bill problem”
- Still requires
  - Network admin knowledge
  - Storage/DB admin knowledge
  - Linux admin knowledge
  - Cloud specific knowledge





App App App App

Bin/ Library Bin/ Library

Operating System Operating System

Virtual Machine Virtual Machine

Hypervisor

Operating System

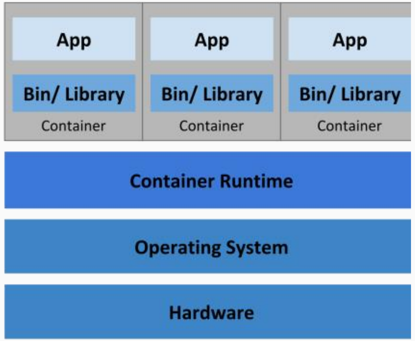
Hardware

Virtualized Deployment

Containerization

- Solving the “Works on my Machine”
- Integrates nicely in CI/CD
- Less overhead
- Docker run and all is good (?)
- But:

  Now deployment requires docker specific skills in addition to all the previous skills



Container Deployment

Help, my docker container isn’t reachable

OK LISTEN CAREFULLY

```
sudo sysctl -w
section.tunable=9000
mount -fvaxuaaa /run/*
../walk/
docker pull -f --force
FORCE=TRUE.’
```

DID YOU HIT ENTER?

Yeah I hit enter

WRONG, I WASN’T DONE!

```
cat “In the realm of the
invisible sun, shadows dance with
the light they cannot touch.” >
/etc/docker/seccomp.conf
sudo systemctl restart docker
```

Two hard problems looking for a solution…

- Deploying and Maintaining Software Systems
- Managing the underlying Infrastructure

Deploying and Maintaining Software Systems

Containers are great but…

- Scaling
- Failover
- Networking
- Observability
- Service Discovery
- Resource Management
- Logging
- Access Management
- …

Are still hard!

Managing the underlying Infrastructure

Actual physical computers are…

A. Weird and non-standard
B. Bad at computing
C. Complicated
D. Rarely in a coherent state
E. Buggy and prone to failure
F. All of the above

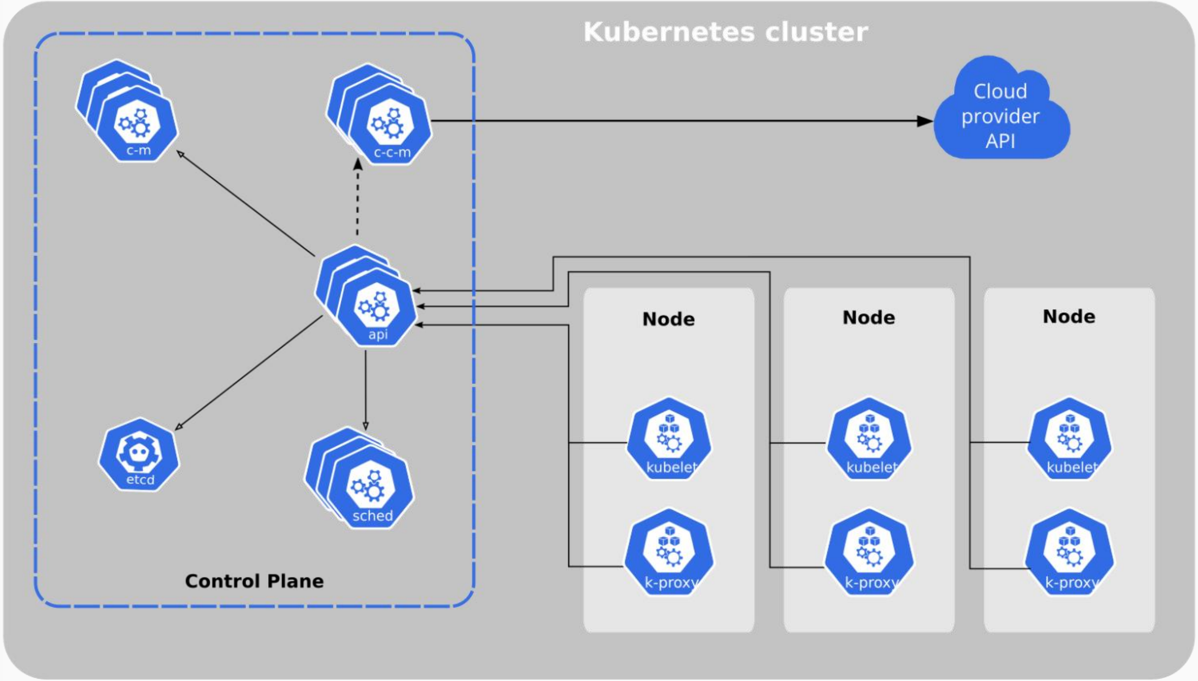
How does Kubernetes solve this?

- Kubernetes abstracts infrastructure and deployment through heavy use of containerization, virtualized networks and ephemeral storage.
- Deployments are declarative, not imperative
- Users don’t need to worry how the definitions are fulfilled.

How does K8s solve this?

The control plane manages a cluster of worker nodes.

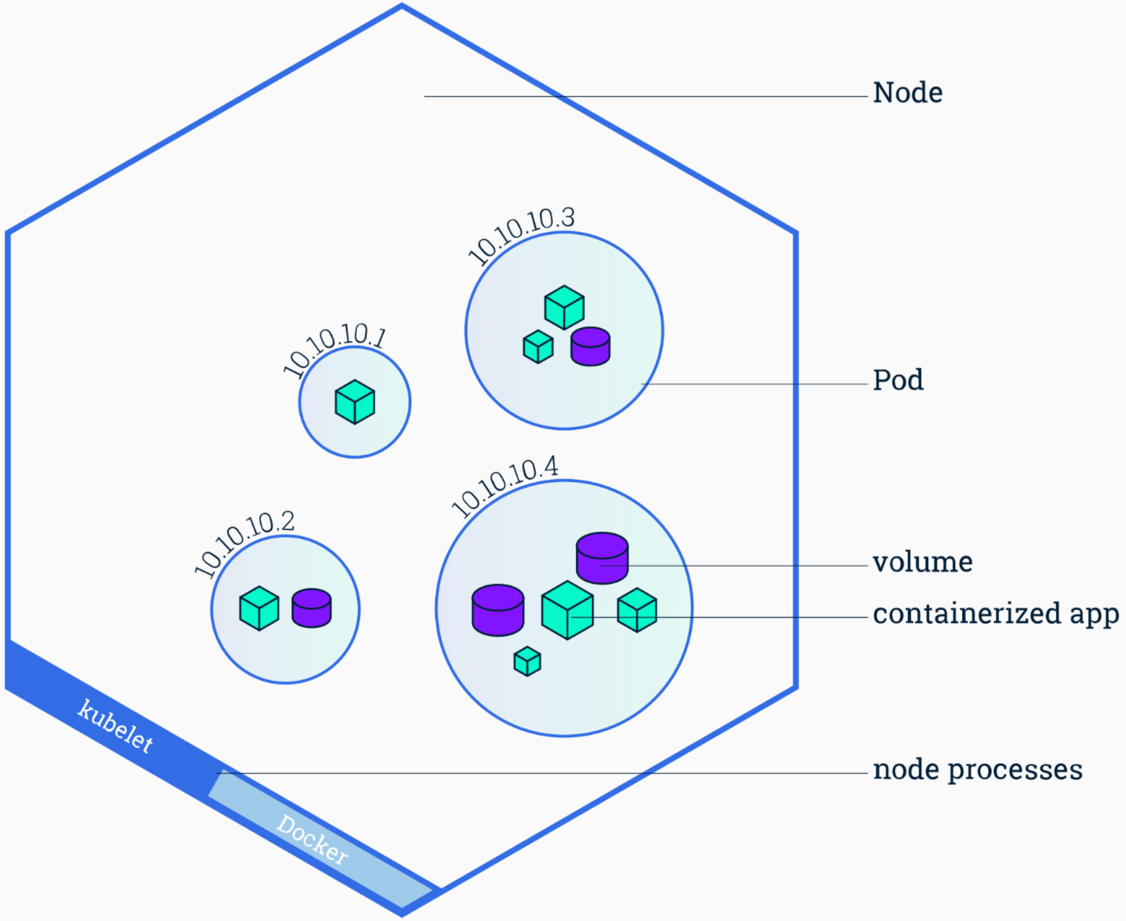
The cluster is exposed as a ***virtual platform*** to provide resources for the architecture of a Software System.



How does K8s solve this?

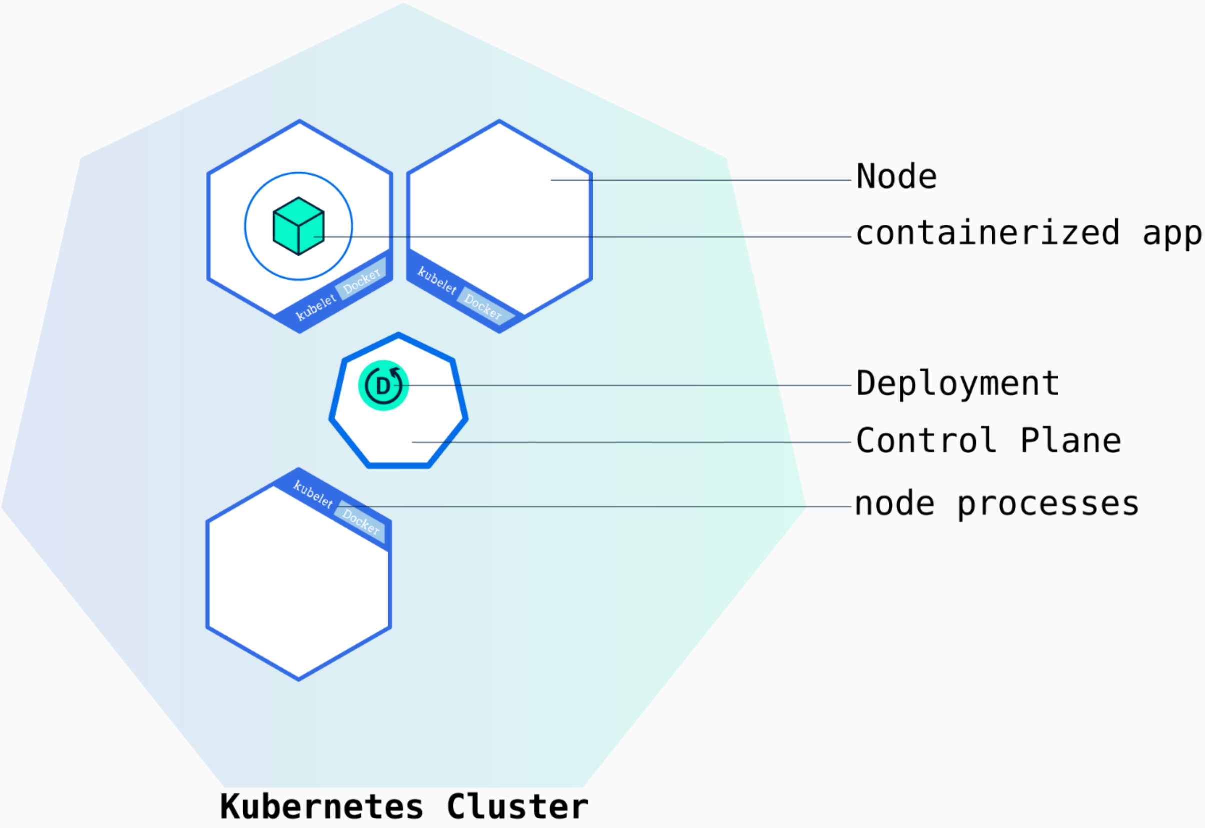
Kubernetes defines a set of building blocks (ressources).

The characteristics of concrete resources are defined in YAML.



How does K8s solve this?

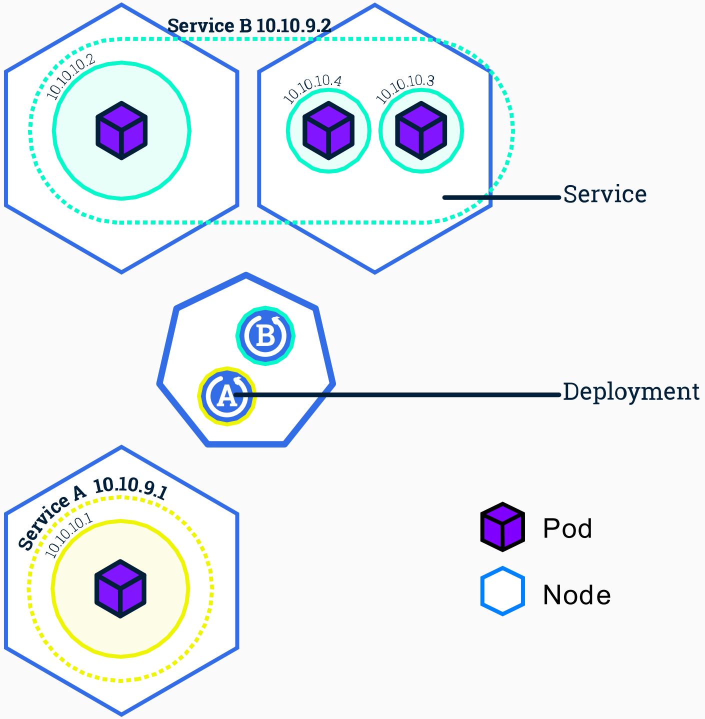
Higher level resources are built from these primitive resources.



How does K8s solve this?

To deploy a Software System:

1. Write a YAML definition of your architecture
2. ***Apply*** this definition to a Kubernetes cluster with ***kubectl*** or the ***API***.
3. Kubernetes ensures that those definitions are always fulfilled.



Where do I get one of these clusters?

- Host yourself
- Rent from a cloud vendor

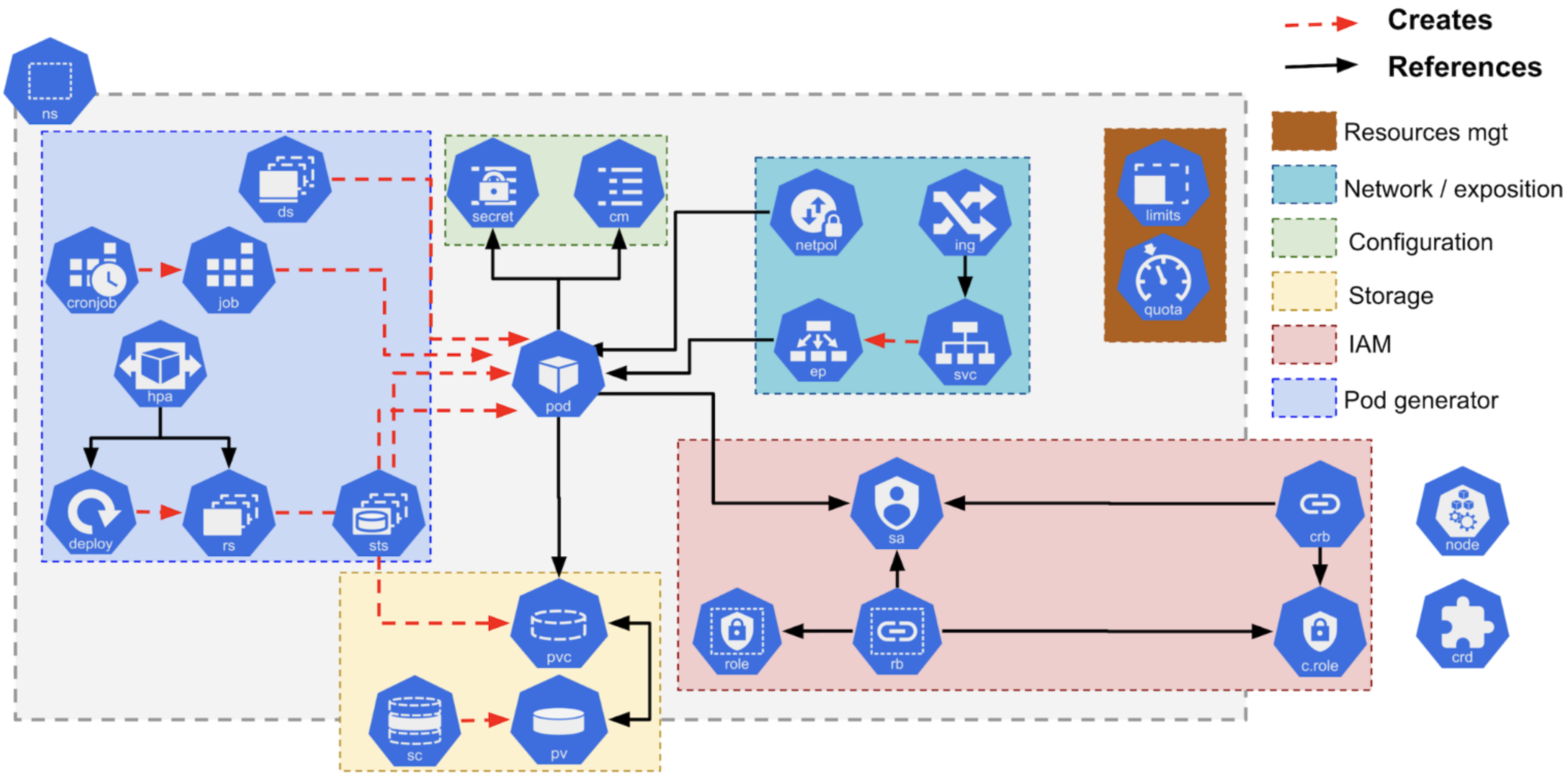
Google Cloud

Azure

aws

Kubernetes Resources

Kubernetes Resources Map



Namespace

- Useful for isolating
- Usually one deployment per namespace
- Almost every other resource is attached to a namespace

```yaml
apiVersion: v1
kind: Namespace
metadata:

  name: myspace
```

ConfigMap

- Small (1MB max) immutable configuration storage
- Config itself in a YAML
- Can be mounted as a ***Volume***
- Can be accessed through environment variables

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  # property-like keys; each key maps to a value
  player_initial_lives: "3"
  ui_properties_file_name: "user-
interface.properties"
  # file-like keys
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow

    allow.textmode=true
```

Secret

- Similar to a ***ConfigMap***
- Values are protected
- A change causes a container to be recreated
- Types for common use cases

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-sa-sample
  annotations:
    kubernetes.io/service-account.name: "sa-name"
type: kubernetes.io/service-account-token
data:
  mySecretValue: YmFyCg==
```

Persistent Volume

- Most simple unit of persistent storage
- Requires a storage provider
- Can be attached to ***Pods***
- Only persistent volumes need to be defined!
- Requires a Persistent Volume and a Claim to attach it to a ***Pod***

Ephemeral Volumes are prefered and inferred from *Pod* definitions!

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs
spec:
  capacity:
    storage: 1Mi
  accessModes:
    - ReadWriteMany
  nfs:
    server: nfs.srv.org
    path: "/"
  mountOptions:
    - nfsvers=4.2
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: nfs
spec:
 accessModes:
   - ReadWriteMany
 storageClassName: ""
 resources:
   requests:
     storage: 1Mi
 volumeName: nfs
```

Pod

- Most basic unit of computation
- Contains container(s)
- May contain an ***InitContainer***
- References the ***Volumes*** attached to a container
- Has its own IP address

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - image: nginx:1.14.2
    ports:
    - containerPort: 80
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

Deployment

- Consists of metadata and a template for one or more ***Pods***
- Defines the number of replicas of said ***Pod***
- Often labeled to allow the connection to a ***Service***

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods
matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Service

- Connects a computation resource with a FQDN.
- Every computation resource needs a ***Service*** to be routable.
- Provides automatically load balancing to the ***Pods*** in a ***Deployments***
- Contains the ports of used to access a ***Service***

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Ingress

- Exposes a ***Service*** to the external network
- Maps external URLs and FQDNs to internal ***Services***
- Usually terminates TLS
- Requires an ***Ingress Controller*** to be usable.

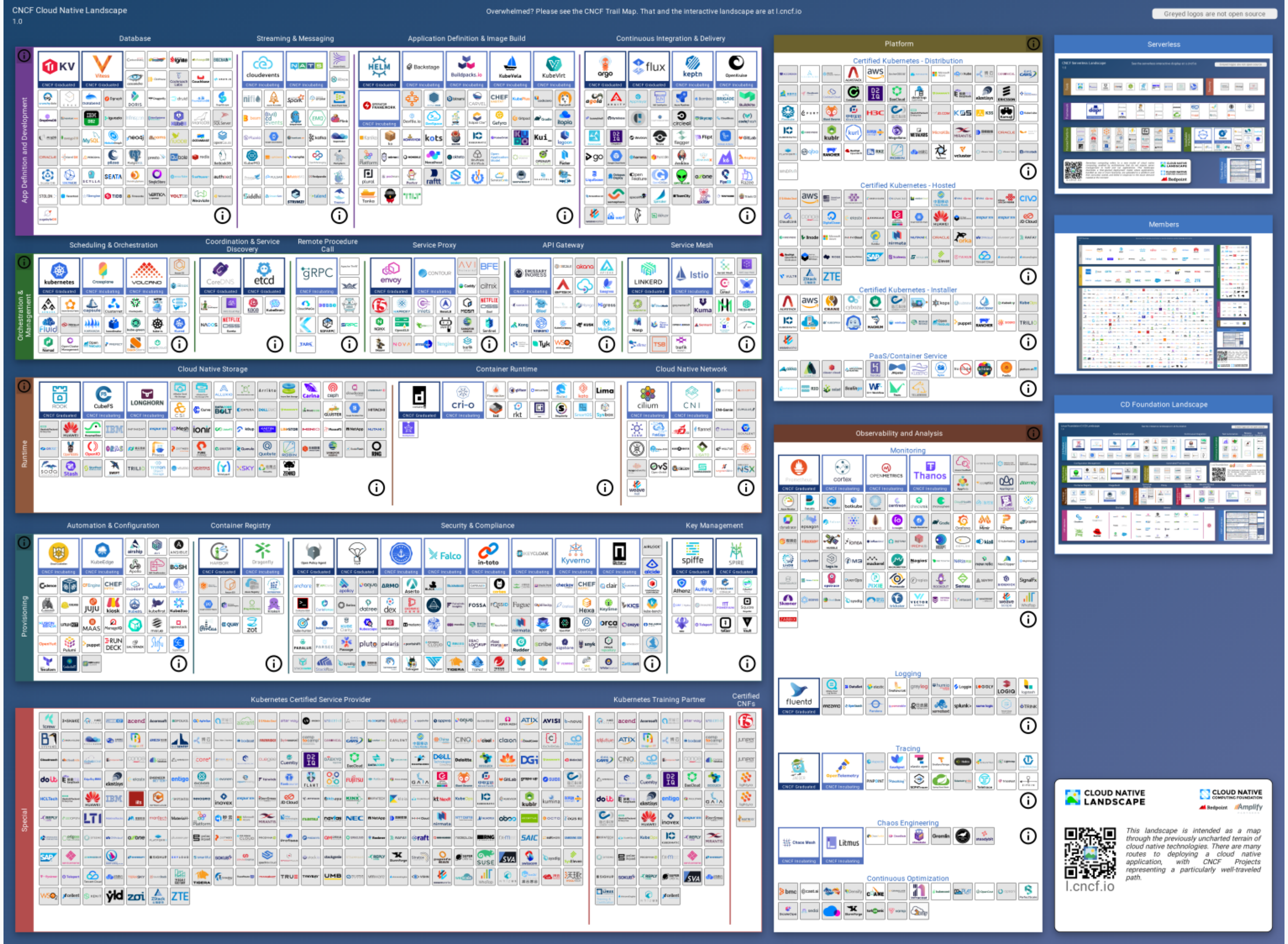
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

Custom Resources and Operators

- The Kubernetes is extensible through custom resource definitions (CRD).
- CRDs are managed through operators.
- Operators
    - Plugins for Kubernetes
    - Run on the Cluster with extra privileges
    - Hook into the Kubernetes API

```yaml
apiVersion: acid.zalan.do/v1
kind: postgresql
metadata:
  name: acid-minimal-cluster
  namespace: default
spec:
  teamId: "acid"
  volume:
    size: 1Gi
  numberOfInstances: 2
  users:
    zalando:  # database owner
    - superuser
    - createdb
    foo_user: []  # role for application foo
  databases:
    foo: zalando  # dbname: owner
  preparedDatabases:
    bar: {}
  postgresql:
    version: "14"
```

The Kubernetes Ecosystem


CNCF Cloud Native Landscape
1.0
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
l.cncf.io

Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

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.

Helm

- Package Manager for Kubernetes
- Templating YAML Files
- De-Facto Standard to install external software on a cluster

- Provides Certificates
- Letsencrypt Support
- Useful to secure Ingresses
- Prerequisite for many other extensions

CERT
MANAGER

Cluster Management

- Provided either by Cloud Vendor or Selfhosted
- Rancher de-facto Standard outside the RedHat Openshift Ecosystem.

RANCHER®

Networking

- Provided via CNI-Plugins
- Essential components in Kubernetes
- Enable network connectivity for containers within a cluster
- Manage IP addresses, routes, etc.
- Facilitate communication between containers, pods, and external networks

cilium

PROJECT CALICO

Storage

- Required for Persistent Volumes
- Require a CSI Driver
- Externally Provided by
  - Cloud Vendor
  - NFS, iSCSI, etc.
- Hosted in the Cluster itself
  - Longhorn
  - Rook (Ceph)

Ingress Providers

- Required for Ingresses
- Can bring their own extensions to Kubernetes networking
- Less relevant in the cloud as an external Load Balancer is often provided

NGINX®

træfik proxy

HAPROXY

Continuous Deployment

- Specifically designed for Kubernetes
- Almost always following a GitOps approach

Service Mesh

Provide advanced networking capabilities:

- Traffic isolation
- mTLS
- Traffic Observability
- Rate Limiting
- A/B test routing
- Additional observability

Istio

LINKERD

Observability

- Collecting metrics:
  - Resource utilization
  - Response time
  - Outage
- Sending alerts

JAEGER

Prometheus

Grafana

Logging

- Collecting logs from all Containers
- All Information accessible in one location
- Popular solutions:
  - ELK stack
  - EFK stack

Container Registry

- ”Dockerhub at home”
- Faster image pulls and caching
- Provide additional capabilities:
    - Signing
    - Vulnerability scans
    - Autocleaning

HARBOR

Dragonfly

Kubernetes for AI enable Software

Why Kubernetes for AI enabled Software?

In deploying AI enabled software systems, the issues intrinsic to traditional systems are frequently magnified:

- The need for scalability to accommodate larger models
- Increased frequency of deployments due to regular updates
- A higher rate of rollbacks due to potential issues with model quality
- The expense of inference necessitates efficient hardware utilization

Kubernetes design philosophy

Declarative over imperative deployments

Treat deployments like cattle, not pets

Operations must be atomic

Resilience and redundancy aren't optional

Extensibility over batteries included

GitOps and Kubernetes

What is GitOps?

Builds on

- Monorepo
- Infrastructure/Configuration as Code
- CI/CD
- Deployment as Declaration

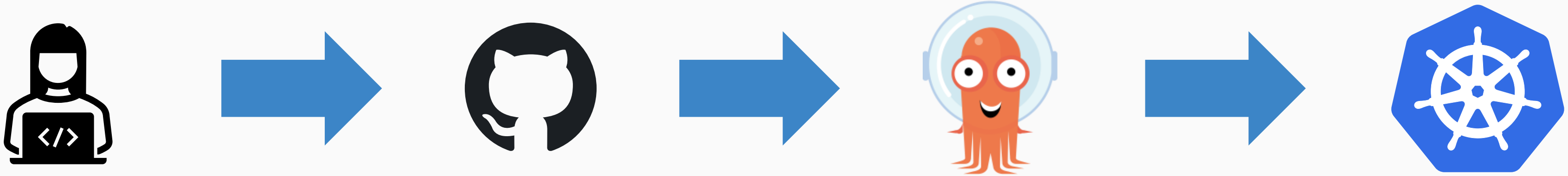
To enable a **Single Source of Truth** for Deployment

Why GitOps

Single Source of Truth enables:

- Unified Development Flow
- Coherent State
- Easier Rollbacks and Debugging
- Centralized resource management

How does GitOps work?



DevOps Engineer commits to the repository

CD monitoring picks up changes from Git Host

CD adjusts deployment in the K8s cluster

Challenges of GitOps

- Initial Learning Curve
- Secrets Management

  → Secrets Provider (Hashicorp Vault, etc.)
- Complex dynamic deployments

  → Dedicated deployment pipelines
- Requires Strict Git Discipline

HashiCorp
Vault

Kubernetes as a Platform for AI

Why extend Kubernetes

- Flexible on-demand scaling
- Isolation through Containerization
- Workload management

→ Needed for managing big data, training computationally intensive models and scalable inference.

- Unified Developer Experience
- Extendible API
- Operator Framework

→ Makes the life of developers and users much easier

Kubernetes as an “AI Platform”-Platform

Many AI Development and Deployment Tools use Kubernetes at their core or are adapted to fit into the ecosystem

Apache Airflow

FEAST

Kubeflow

jupyterhub

mlflow™

APACHE Spark™

JupyterHub

- Multi-User Notebooks with simultaneous access
- Secured, isolated workspaces
- Allows customizable setups,

  supporting different kernels and configurations
- Enables Data Scientists to access large computational resources

  (including GPUs) on their end device.

Kubeflow

- Kubernetes-Native
- All in one platform
- End-to-End ML workflow from data ingestion and preprocessing to model training, serving, and monitoring
- Component-Based Architecture:
    - Jupyter notebooks
    - Training jobs operators
    - Serving stack for deploying models

! Extremely hard to set up outside of cloud vendors !

Apache Spark

- Unified computing engine for big data processing and analytics
- Supports SQL, streaming and ML
- In-memory processing
- Bindings for Java, Scala and Python
- Adapted to Kubernetes to utilize its resource management capabilities

Apache Airflow

- Platform to programmatically author, schedule and monitor tasks in pipelines
- Pipelines can be dynamically triggered
- Utilizes directed acyclic graphs (DAGs)
- Extensible via plugins
- Scales task execution across multiple worker

…and many more

SELDON

Weights & Biases

feathr

Weaviate

Milvus

Elyra

KServe

DELTA

a deep learning language technology platform

Security in Containers and Kubernetes

Security issues of Containers

- Containers are “Root Filesystems in a Tarball”

  → Libraries frozen in time
- Still huge attack surface
- Horizontal escalation
- Supply chain risk
- Leaky isolation

Mitigation: Minimalist containers

Utilizing a multi-stage build approach:

1. Build in a rich environment
2. Execute artifacts in a minimalist container
   - Alpine
   - Distroless

Mitigation: Reduced Privileges

- Running as non-root in a container
- Setting CPU and memory limits
- Enforcing a security confinement (Seccomp + SELinux)
- Rootless containers with user privileges

Mitigation: Signing

- Signing containers to provide integrity and authorship
- Can prevent supply chain attacks

  …but only if those signatures are enforced!
- Long Term Certificates can cause further problems

  → Sigstore and Cosign

sigstore

Further Mitigations

- Containers as actual VMs:
    - Firecracker
    - Kata Containers
    - gVisor

- Container scanning?

→ Highly unreliable

Kubernetes Demo