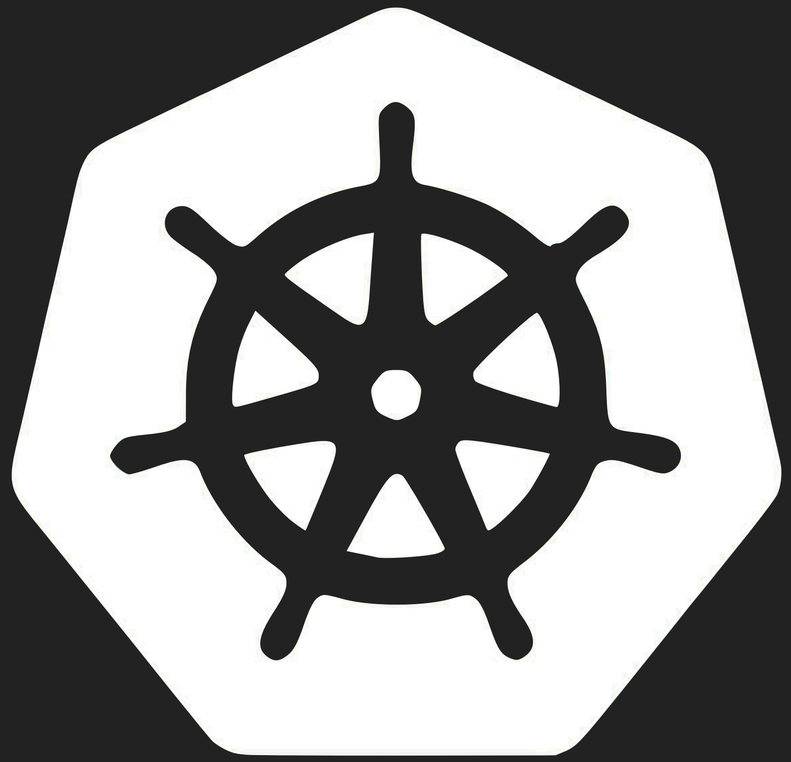


AN INTRODUCTION TO CONTAINER ORCHESTRATION

FROM 0 TO KUBERNETES

HI, I'M HEIKO

- I'm a Senior Consultant from a company you never heard
- I help putting EVs in the Cloud
- (Event) Photography
- Longboarding

/in/heikoborchers/

@archang3l_media

@archang3l_media@chaos.social

HI, I'M CEDI

I'm a Senior SRE at $BigTech

I do resiliency engineering

(Analog) Photography

Brazilian Jiu-Jitsu

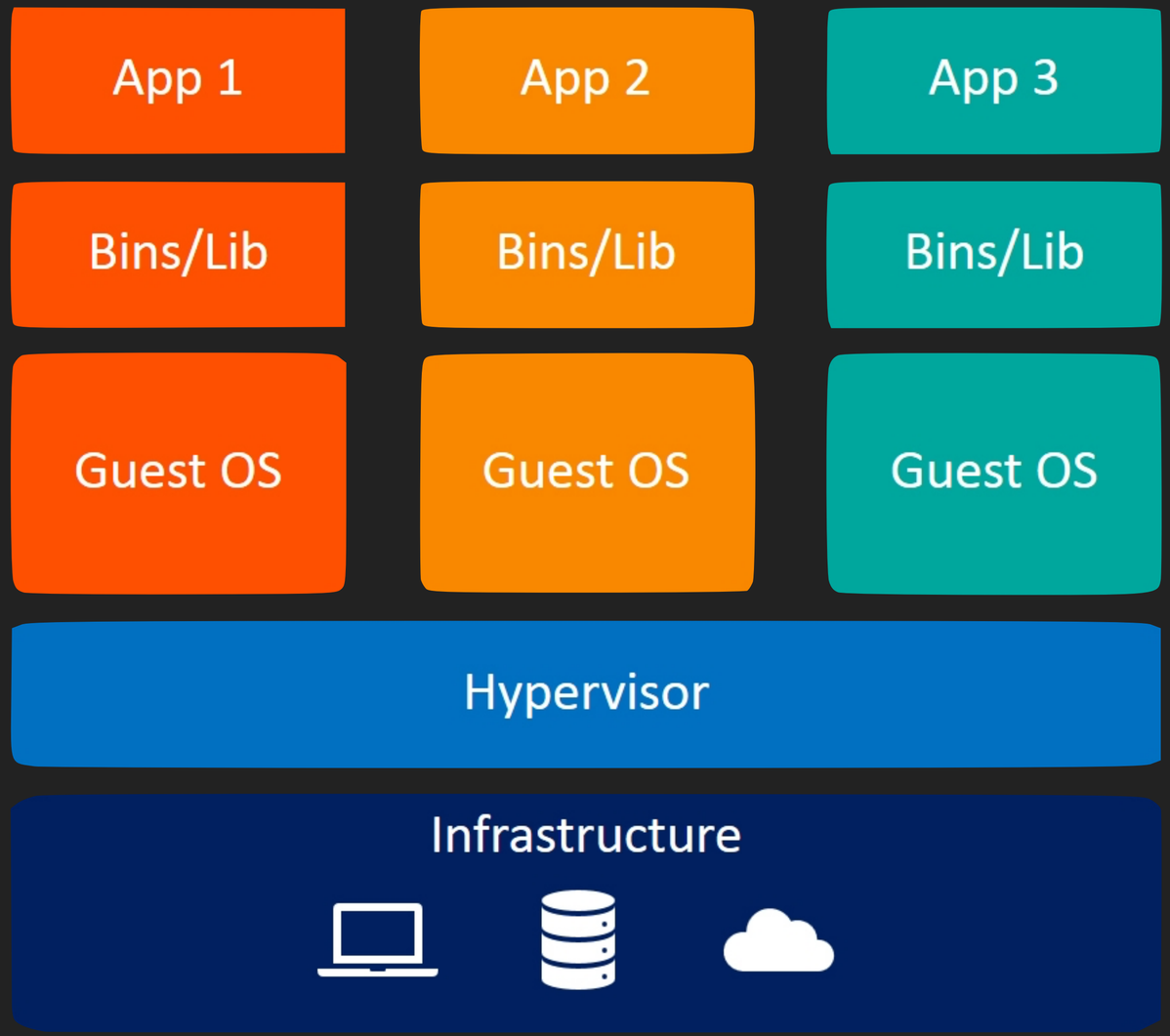
/cedi

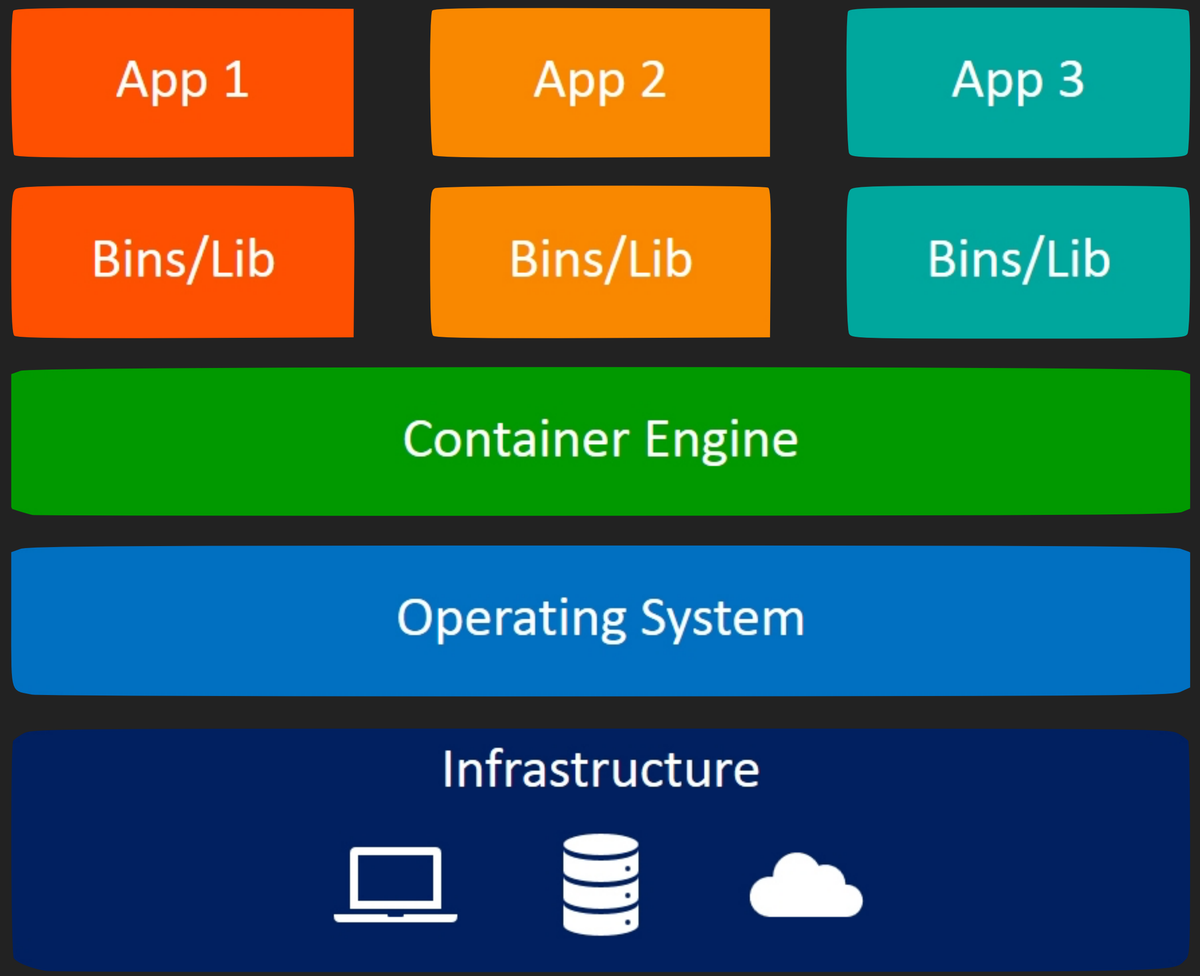
/in/cekienzl

@cedi@hachyderm.io

CONTAINERS, CONTAINERS EVERYWHERE

CONTAINER & VM


App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure


App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Container Engine
Operating System
Infrastructure

source

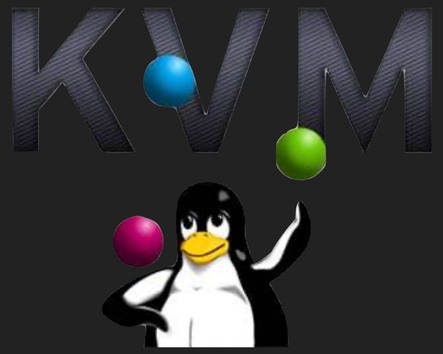
PRO

- Consistency
- Automation
- Stability
- Scalability

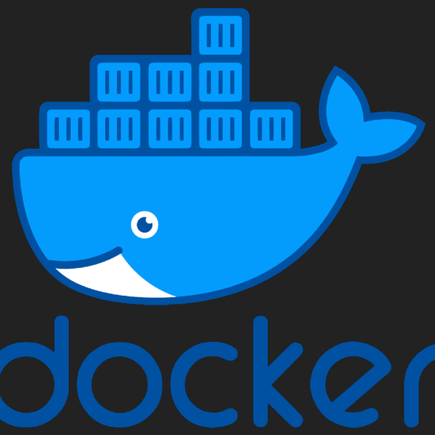
CON

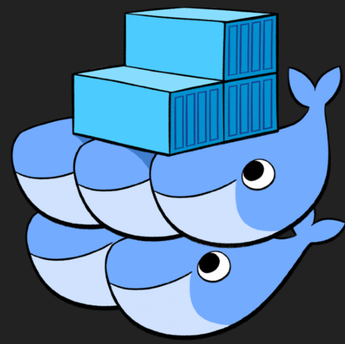
- Yet another layer...
- Learning Curve
- Documentation
- Plethora of tools

VM ORCHESTRATION



CONTAINER ORCHESTRATION

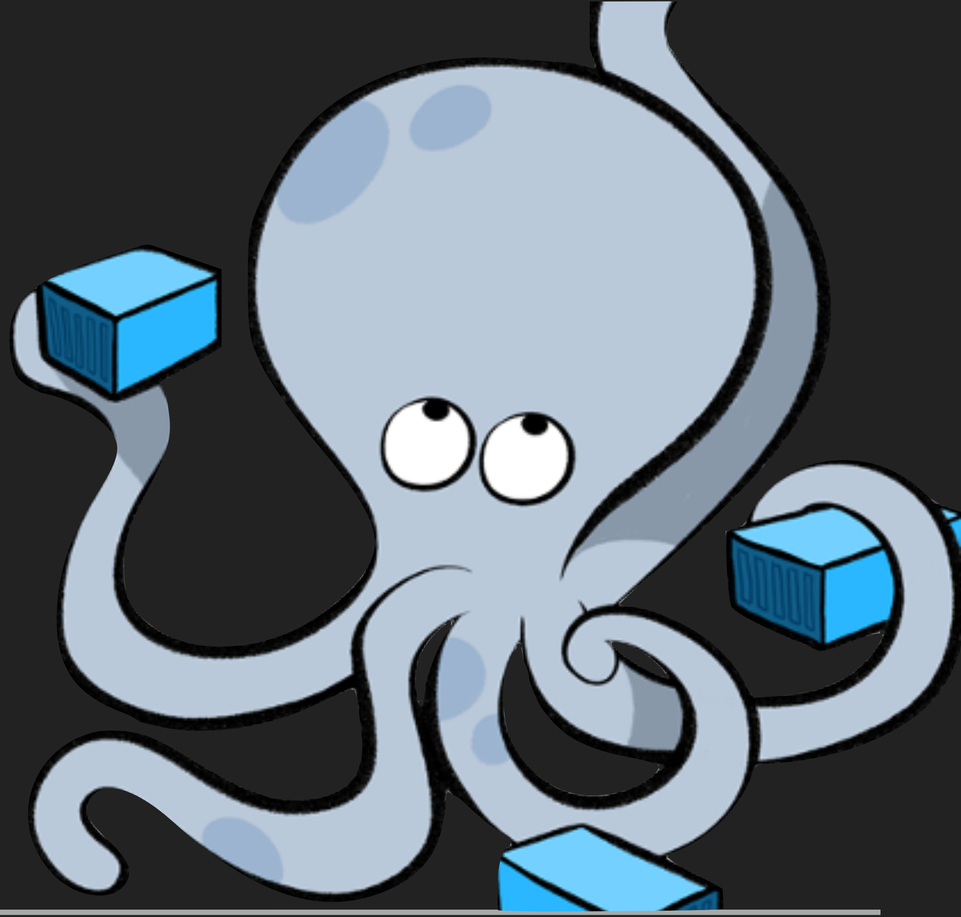




CONTAINER

ORCHESTRATION

CONTAINER ORCHESTRATION



DOCKER COMPOSE

COMPOSE IS A TOOL FOR DEFINING AND RUNNING MULTI-CONTAINER DOCKER APPLICATIONS. WITH COMPOSE, YOU USE A YAML FILE TO CONFIGURE YOUR APPLICATION'S SERVICES. THEN, WITH A SINGLE COMMAND, YOU CREATE AND START ALL THE SERVICES FROM YOUR CONFIGURATION.

Docker Compose Documentation

DOCKER-COMPOSE

File: **docker-compose_clean.yml**

```yaml
version: '3.5'
services:
  gitlab:
    image: 'gitlab/gitlab-ce:latest'
    restart: always
    hostname: 'gitlab.example.com'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://gitlab.example.com'
        registry_external_url 'https://registry.gitlab.example.com'
        gitlab_rails['gitlab_shell_ssh_port'] = 2224
        gitlab_rails['smtp_enable'] = "true"
        gitlab_rails['smtp_address'] = "some-mail-server.example.com"
        gitlab_rails['smtp_port'] = "587"
        gitlab_rails['smtp_user_name'] = "noreply@example.com"
        gitlab_rails['smtp_password'] = "example_password"
        gitlab_rails['smtp_domain'] = "example.com"
        gitlab_rails['smtp_authentication'] = ""
        gitlab_rails['smtp_tls'] = "true"
        gitlab_rails['registry_enabled'] = "true"
        gitlab_rails['registry_host'] = "registry.gitlab.example.com"
        gitlab_rails['registry_port'] = "5000"
        gitlab_rails['gitlab_default_projects_features_container_registry'] = "true"
        gitlab_rails['registry_path'] = "/opt/storage/gitlab/gitlab-rails/shared/registry"
        nginx['real_ip_header'] = "X-Real-IP"
        nginx['real_ip_recursive'] = "on"
        nginx['listen_https'] = false
        nginx['listen_port'] = 80
        registry_nginx['ssl_certificate'] = "/gitlab/certs/fullchain.pem"
        registry_nginx['ssl_certificate_key'] = "/gitlab/certs/privkey.pem"
    ports:
      - "8081:80"
      - "2224:22"
      - "5000:5000"
    volumes:
      - "/opt/storage/gitlab/config:/etc/gitlab"
      - "/opt/storage/gitlab/logs:/var/log/gitlab"
      - "/opt/storage/gitlab/data:/var/opt/gitlab"
      - "/gitlab/certs:/gitlab/certs"
```

PRO

- Easy
- Multiple Environments on a single host
- Stability
- Reproducible
- Declarative / Infrastructure as Code

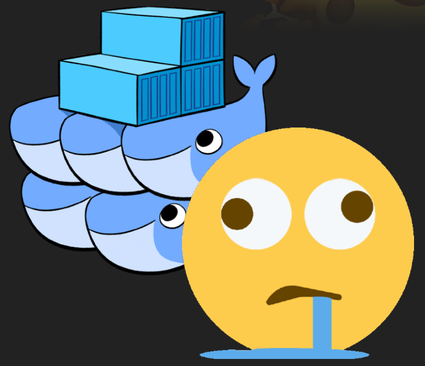
CON

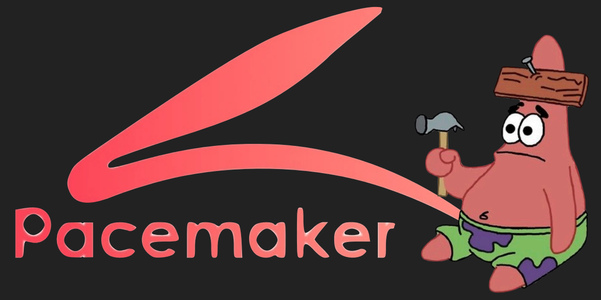
- Single Host Deployments

CONTAINER ORCHESTRATION

CLOUD SOLUTIONS

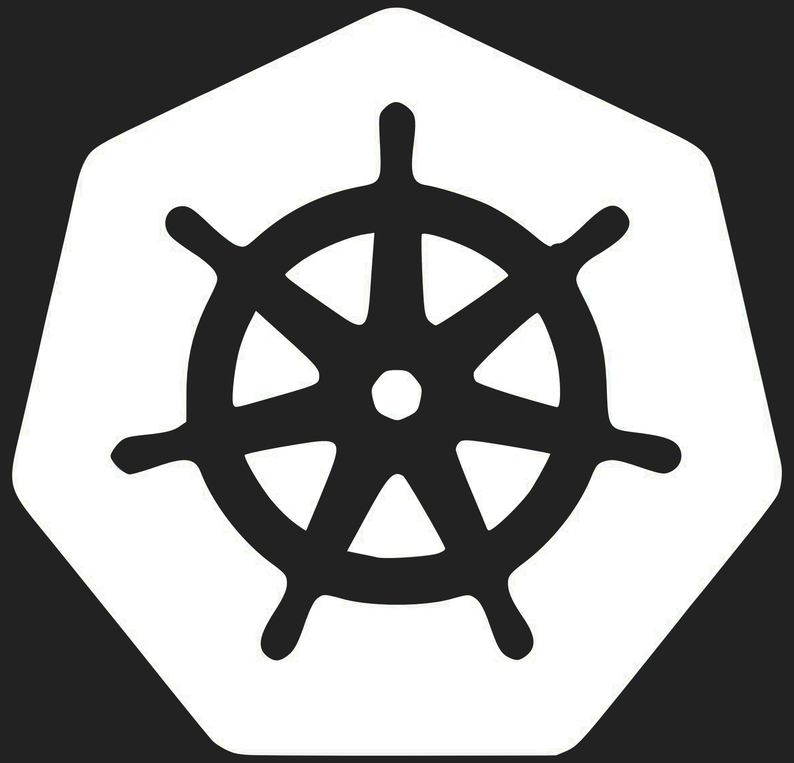
YES, BUT...





CONTAINER ORCHESTRATION

HONORABLE MENTIONS



KUBERNETES

NOW, WHAT _IS_ KUBERNETES?

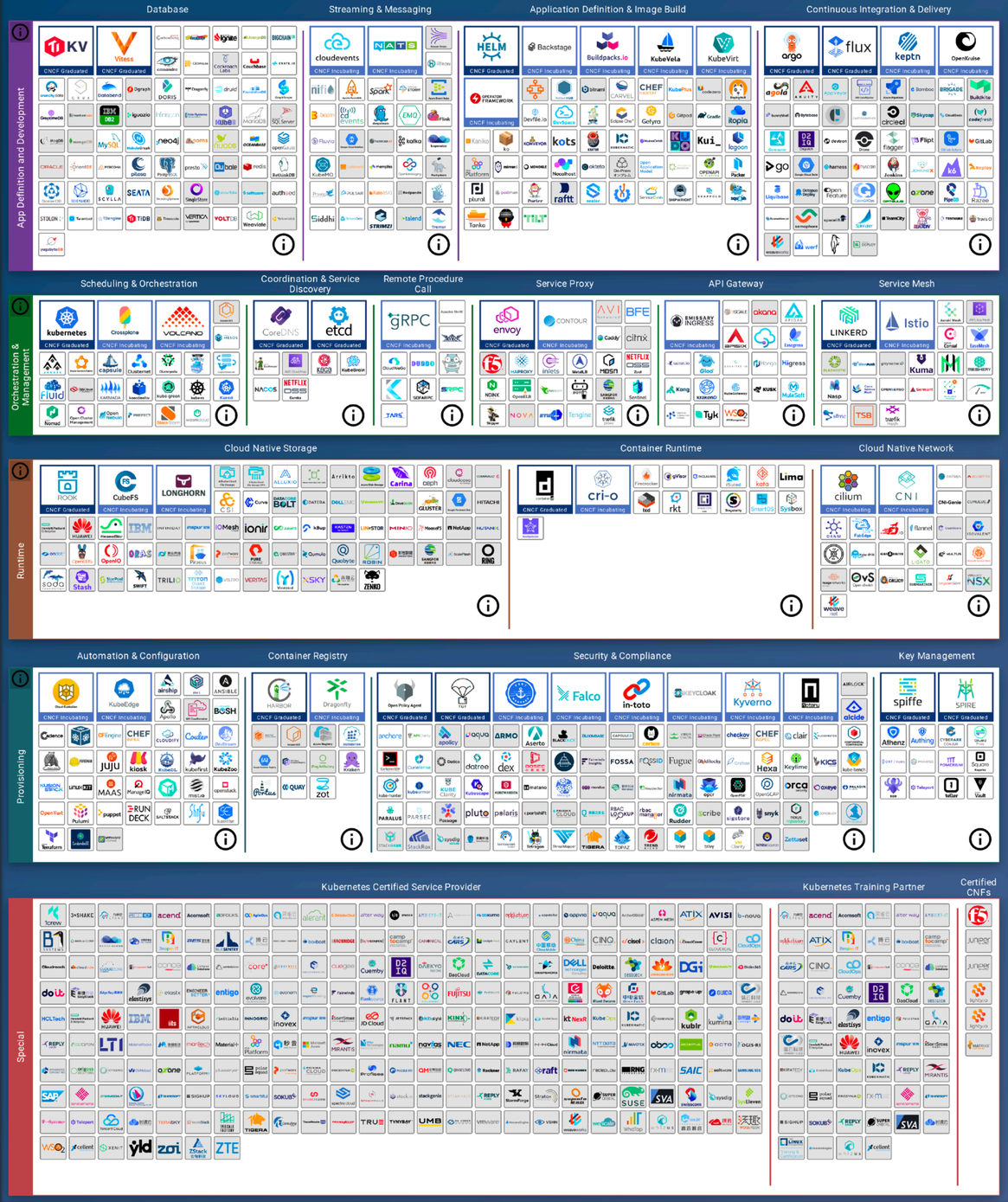
SPOILER ALERT,

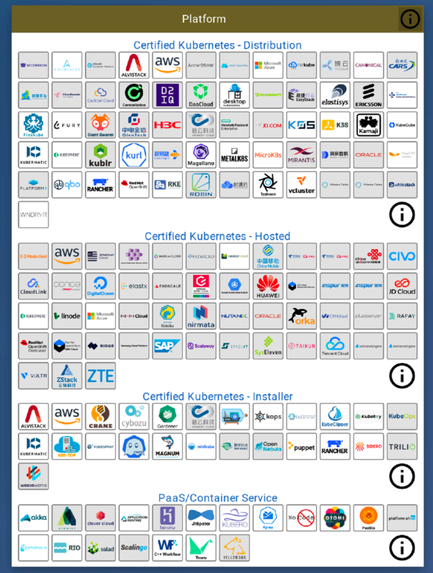
IT'S NOT A SINGLE TOOL

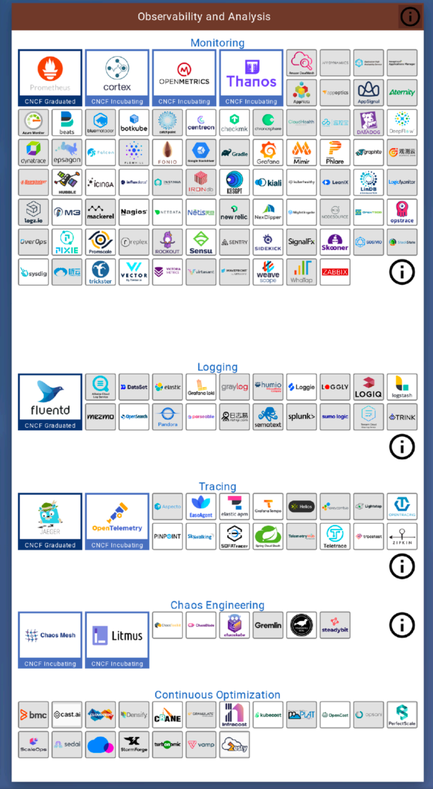
CNCF Cloud Native Landscape
1.0

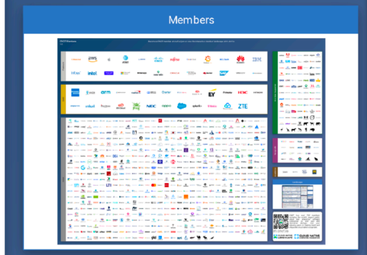
Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

Greyed logos are not open source









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

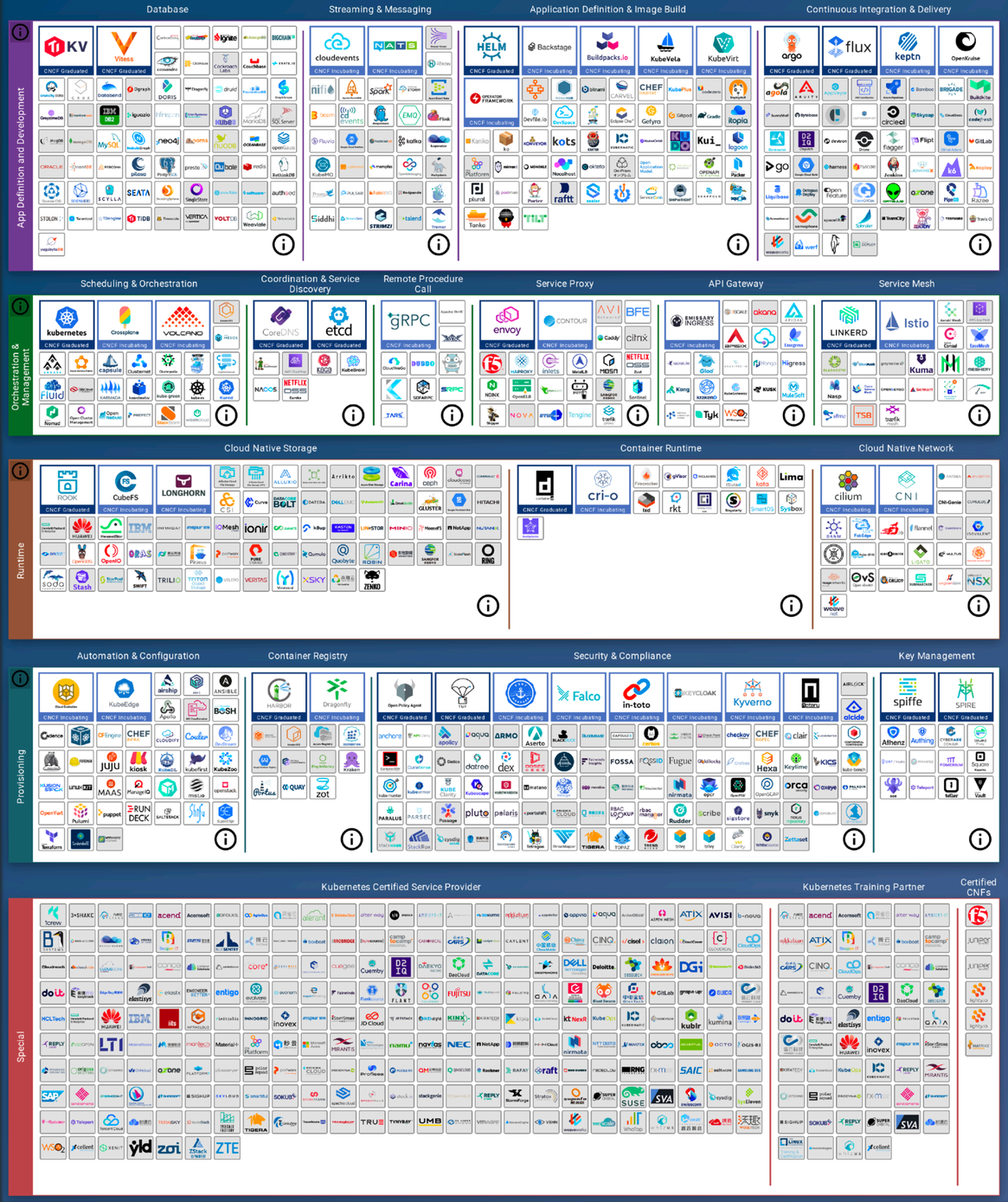
KUBERNETES DEFINES A SET OF BUILDING BLOCKS ("PRIMITIVES") THAT COLLECTIVELY PROVIDE MECHANISMS THAT DEPLOY, MAINTAIN, AND SCALE APPLICATIONS

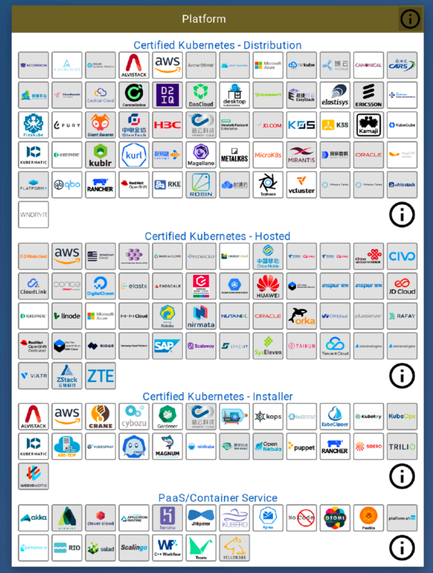
Wikipedia

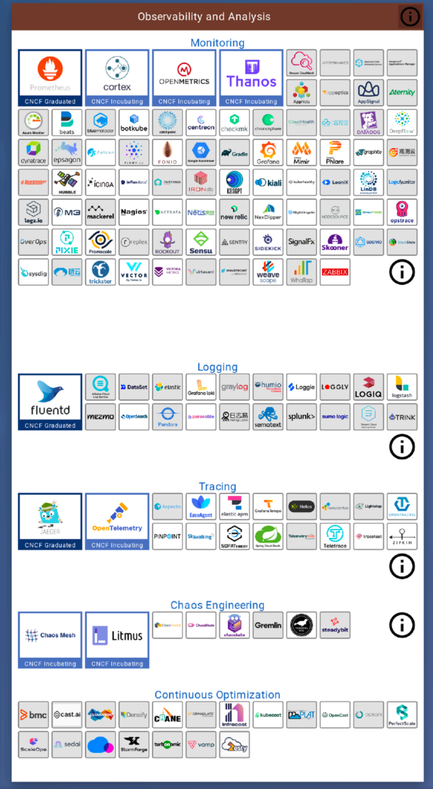
CNCF Cloud Native Landscape
1.0

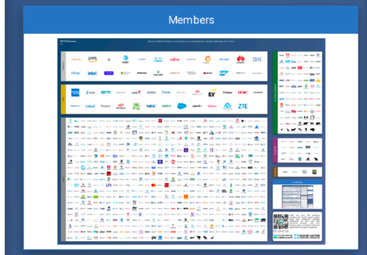
Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

Greyed logos are not open source









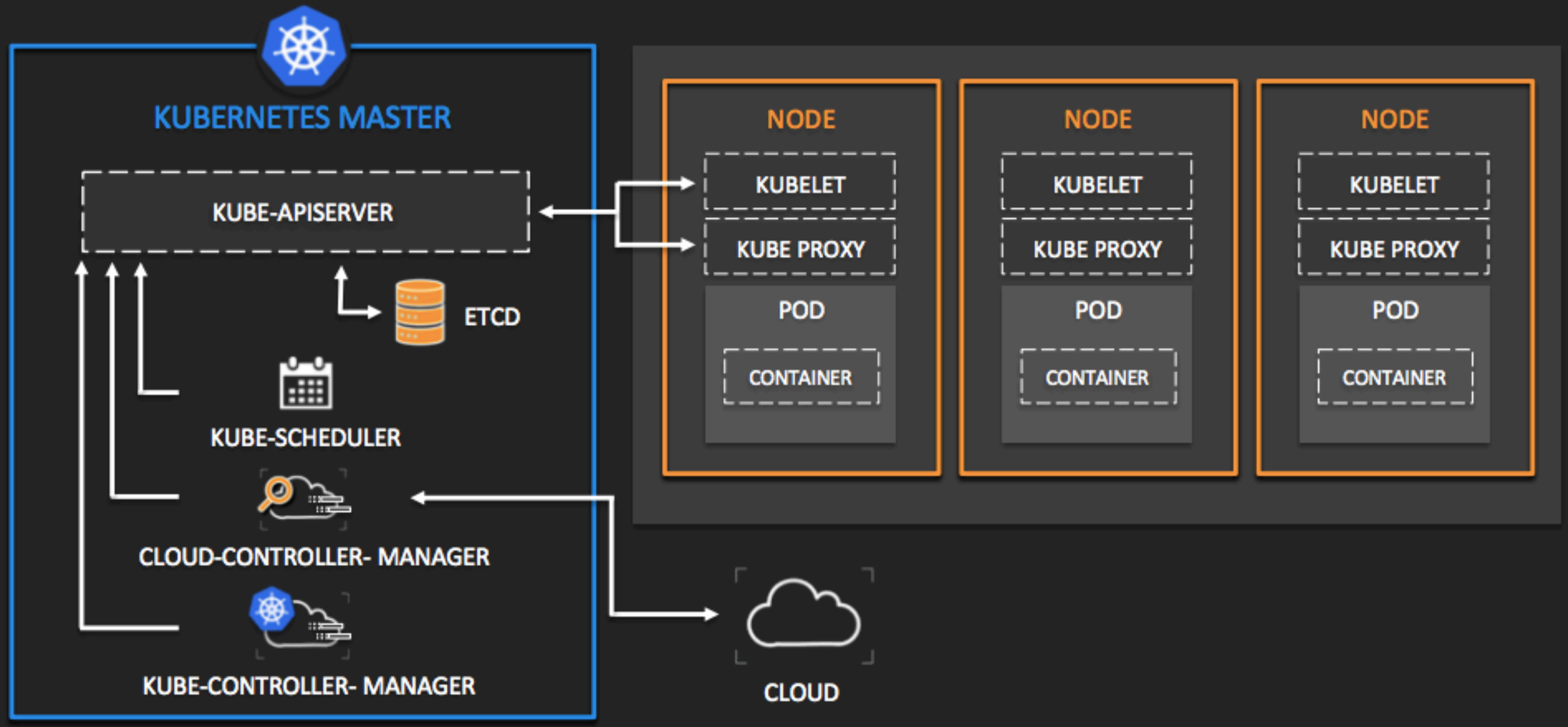
CLOUD NATIVE LANDSCAPE

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.

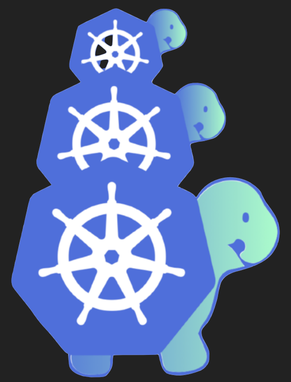
l.cncf.io

BASIC KUBERNETES ARCHITECTURE

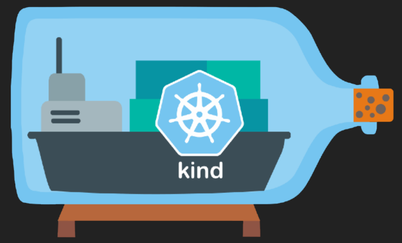


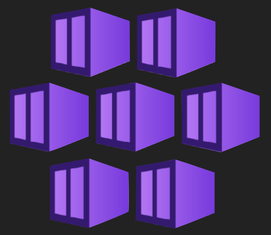
MicroK8s

KOS



kops





MIRANTIS

SAP

KUBERNETES

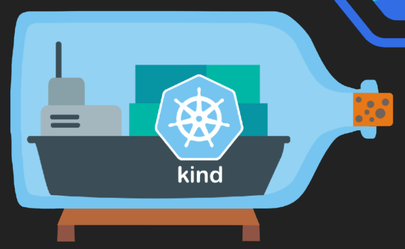
DISTRIBUTIONS

Red Hat OpenShift on **IBM Cloud**

DISTRIBUTIONS

DEVELOPMENT

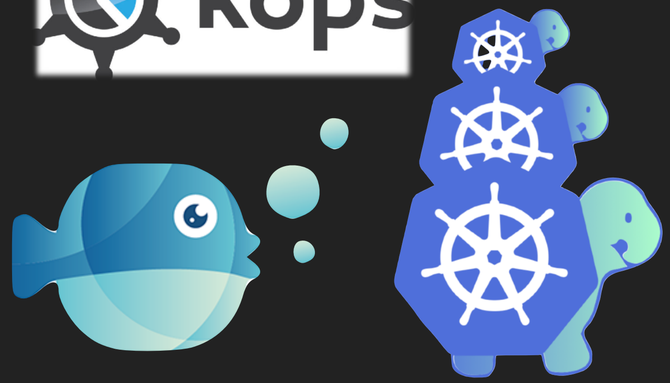
MicroK8s



SINGLE NODE / EDGE

K0S

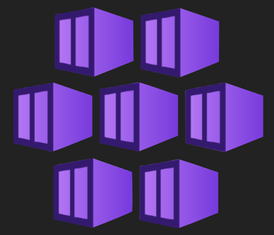
PRODUCTION

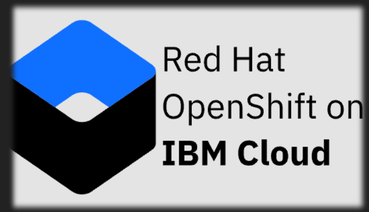


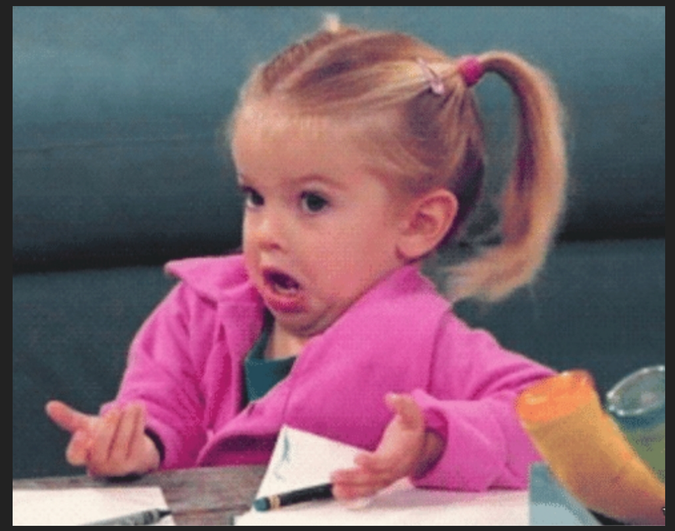
DISTRIBUTIONS... CONTINUED...

ENTERPRISE

CLOUD







KUBERNETES

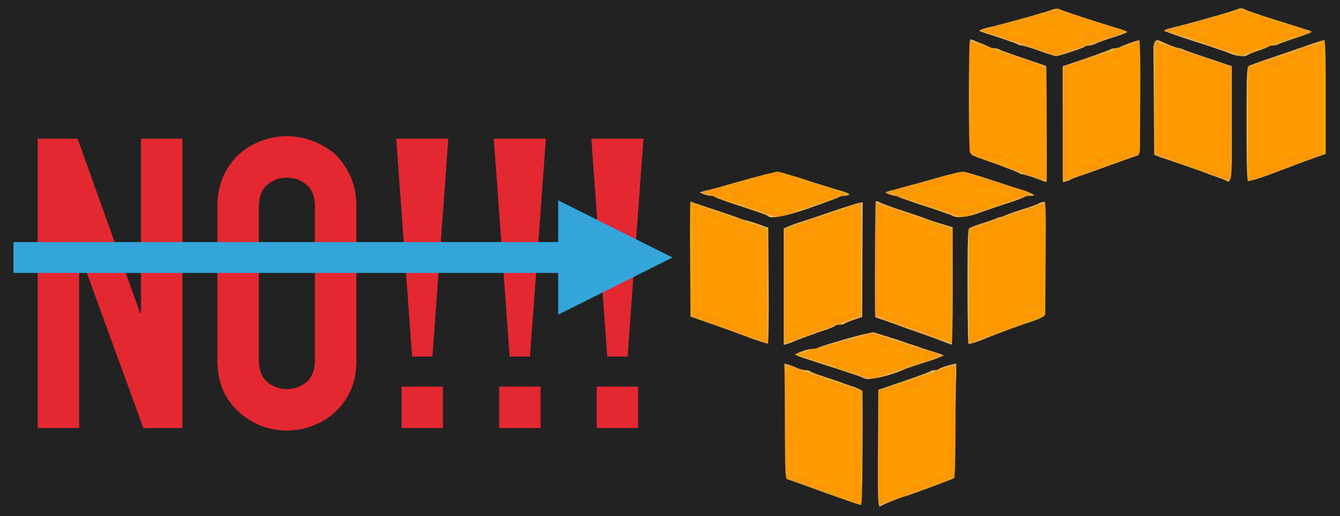
BUT WHEN?

KUBERNETES, BUT WHEN?

LETS PLAY A GAME...

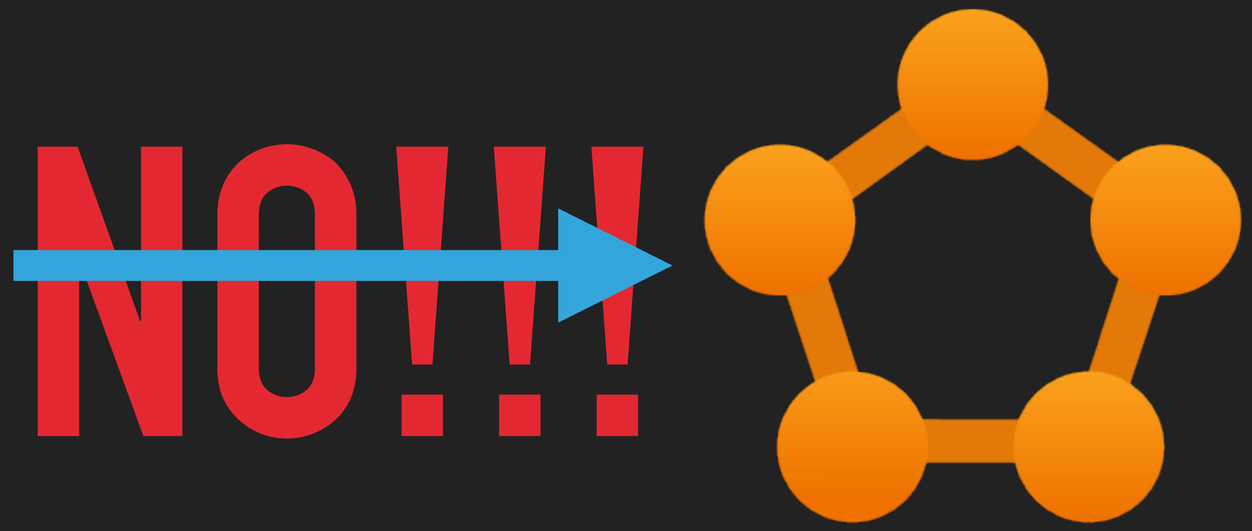
IS KUBERNETES THE RIGHT CHOICE FOR

SERVING STATIC CONTENT?


NO!!!

IS KUBERNETES THE RIGHT CHOICE FOR

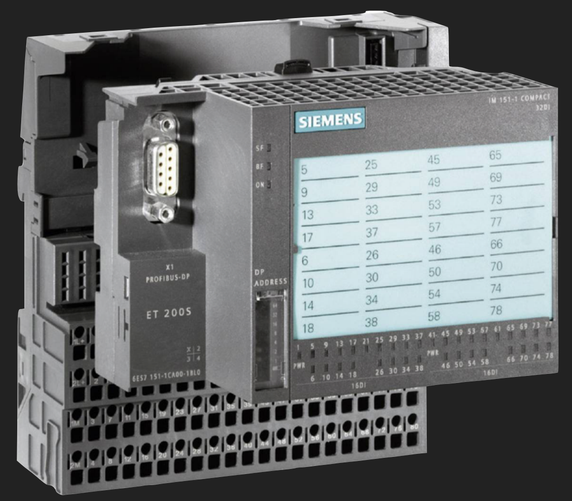
DATABASES?


NO!!!

IS KUBERNETES THE RIGHT CHOICE FOR

PLANT / FACTORY / PHYSICAL SAFETY CONTROL SOFTWARE?

NO!!!



IS KUBERNETES THE RIGHT CHOICE FOR

STATEFUL WORKLOAD?

IT DEPENDS

NO*

* when state is kept in the software

YES*

* when state is kept external (in a database cluster or object storage)

IS KUBERNETES THE RIGHT CHOICE FOR

YOUR COMPANIES (LEGACY/ ENTERPRISE) JAVA APPLICATION?

MAYBE?

PRO

- Improve density (in comparison to running each Application on a single host)
- Isolation (no JVM dependency chaos)
- State is probably (hopefully) stored somewhere else anyway

CON

- It will not scale (better) than before
- Low utilization of Kubernetes native features

IS KUBERNETES THE RIGHT CHOICE FOR

EDGE COMPUTING?

MAYBE?

PRO

- Running your services in a standardized way close to the consumer
- Distributed sync
- State can (and should) be stored centrally

CON

- Managing multiple Kubernetes Clusters requires much more planning
- Multi-Cluster deployments are hard to facilitate
- Latency constraints for inter-cluster communication

IS KUBERNETES THE RIGHT CHOICE FOR

STATELESS MICRO-SERVICES?

YES!

IS KUBERNETES THE RIGHT CHOICE FOR

SERVERLESS COMPUTING?

OPENFAAS

* but you need additional software

DEFAULT TO KUBERNETES ONLY WHEN THERE'S NOT A BETTER OPTION FOR YOUR WORKLOADS.

Karl Isenberg

EVALUATE CAREFULLY

IS KUBERNETES REALLY THE RIGHT CHOICE FOR YOU?

- What are your functional infrastructure requirements?
- What are your scaling requirements?
- How do you handle state in your workload?
- Do you have the operational capacity to operate a Kubernetes cluster?

EVALUATE CAREFULLY

POSSIBLE ALTERNATIVES

- IaaS: Hetzner, Azure VMSS, AWS EC2
- PaaS: Heroku, Azure WebApps, AWS Elastic Beanstalk
- CaaS: DigitalOcean Droplets, Azure Container Instances, AWS Fargate
- FaaS: Azure Functions, AWS Lambda

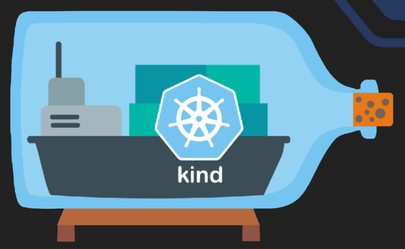
KUBERNETES

BUT HOW?

DISTRIBUTIONS

DEVELOPMENT

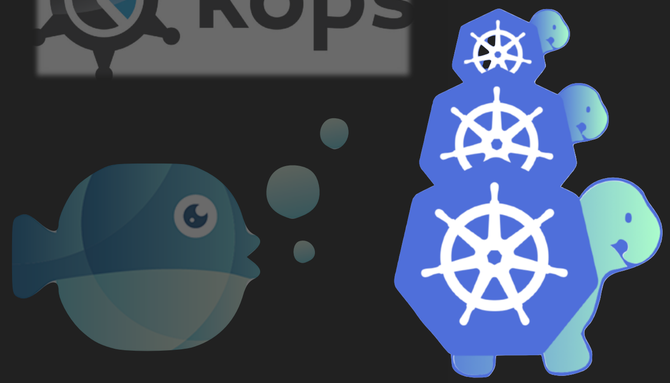
MicroK8s



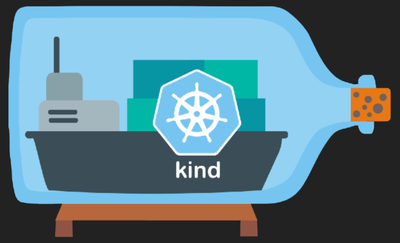
SINGLE NODE / EDGE

K0S

PRODUCTION



GET STARTED WITH A DEV ENVIRONMENT



- Make your first steps with “kind” on your local machine

SINGLE NODE-, EDGE-, (AND HOMELAB) DEPLOYMENT

- K3s on a single node is good enough for your home-lab
- Multi-node K3s is probably enough for most use-cases!
- Use a systemd service unit to keep k3s running
- Use the K3s Ansible Playbook

..de/ansible/k3s-ansible

cedi@mae ~/src/cedi/av0de/ansible/k3s-ansible master≡ ?1 ~1 zsh

```
❯ bat roles/k3s/master/templates/k3s.service
```

File: roles/k3s/master/templates/k3s.service

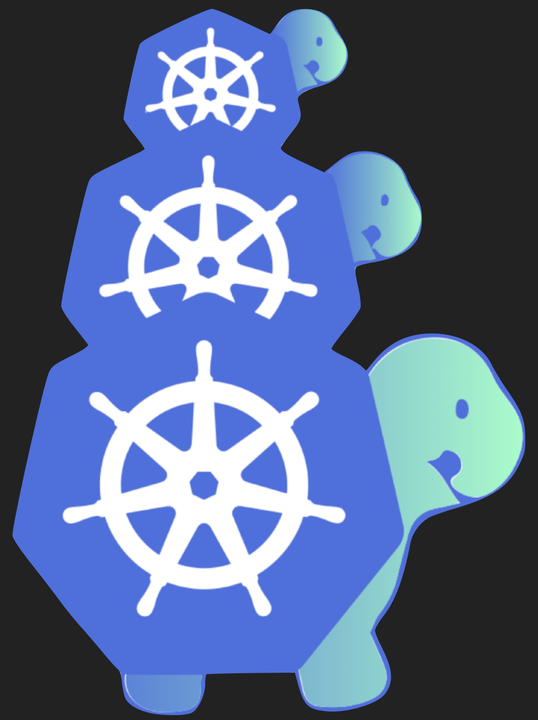
```
[Unit]
Description=Lightweight Kubernetes
Documentation=https://k3s.io
After=network-online.target

[Service]
Type=notify
ExecStartPre=-/sbin/modprobe br_netfilter
ExecStartPre=-/sbin/modprobe overlay
ExecStart=/usr/local/bin/k3s server --data-dir /var/lib/rancher/k3s --flannel-backend=none --disable-network-policy --disable
 traefik
KillMode=process
Delegate=yes
# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=1048576
LimitNPROC=infinity
LimitCORE=infinity
TasksMax=infinity
TimeoutStartSec=0
Restart=always
RestartSec=5s

[Install]
WantedBy=multi-user.target
```

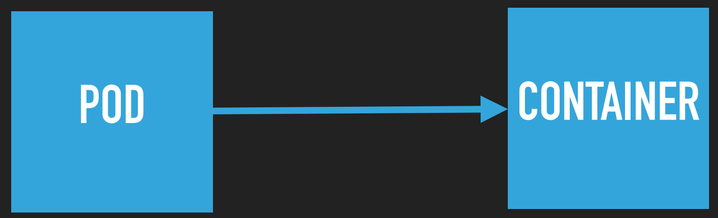
IF YOU REALLY WANT TO GO ALL THE WAY

- Use the Cluster API
  - Declarative Management isn't just nice for workload but for entire clusters too
- Read the documentation of clustermgr
- Since most of you deploy to hetzner anyway
  - ccl.pw/cluster-api-hetzner
- This isn't gonna be easy! But we warned you :)

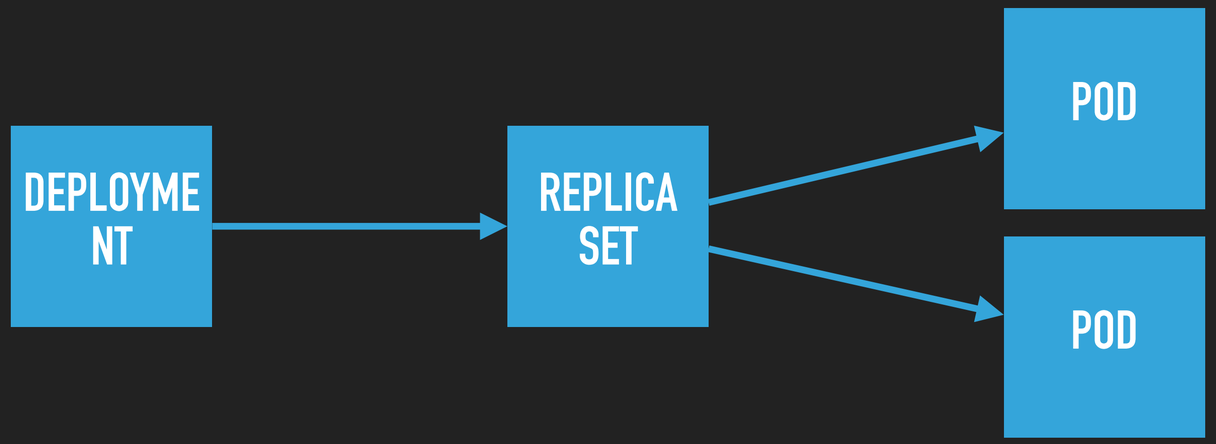


DEPLOY WORKLOAD

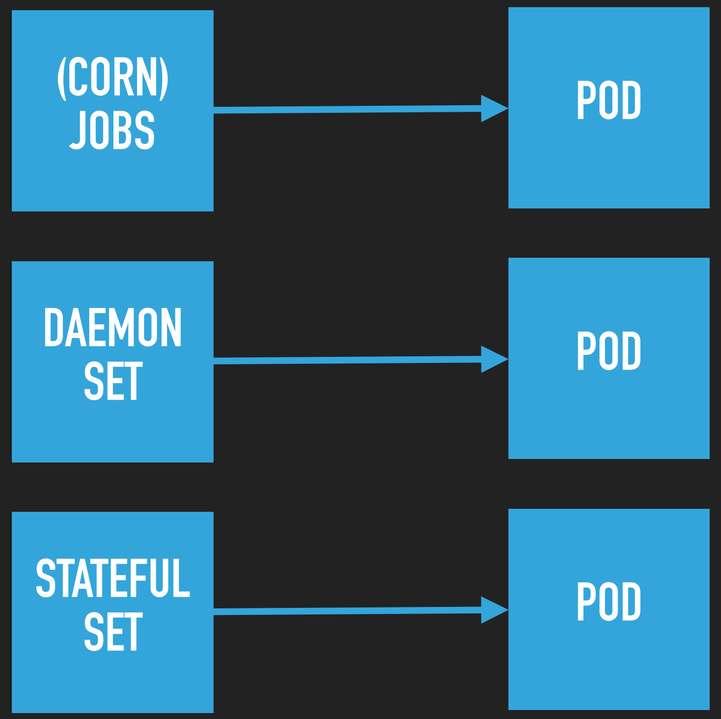
PODS



MANAGING PODS



MANAGING PODS



kubectl

A WORD ON “KUBECTL APPLY -F” AND “HELM INSTALL”

PLEASE DON’T*

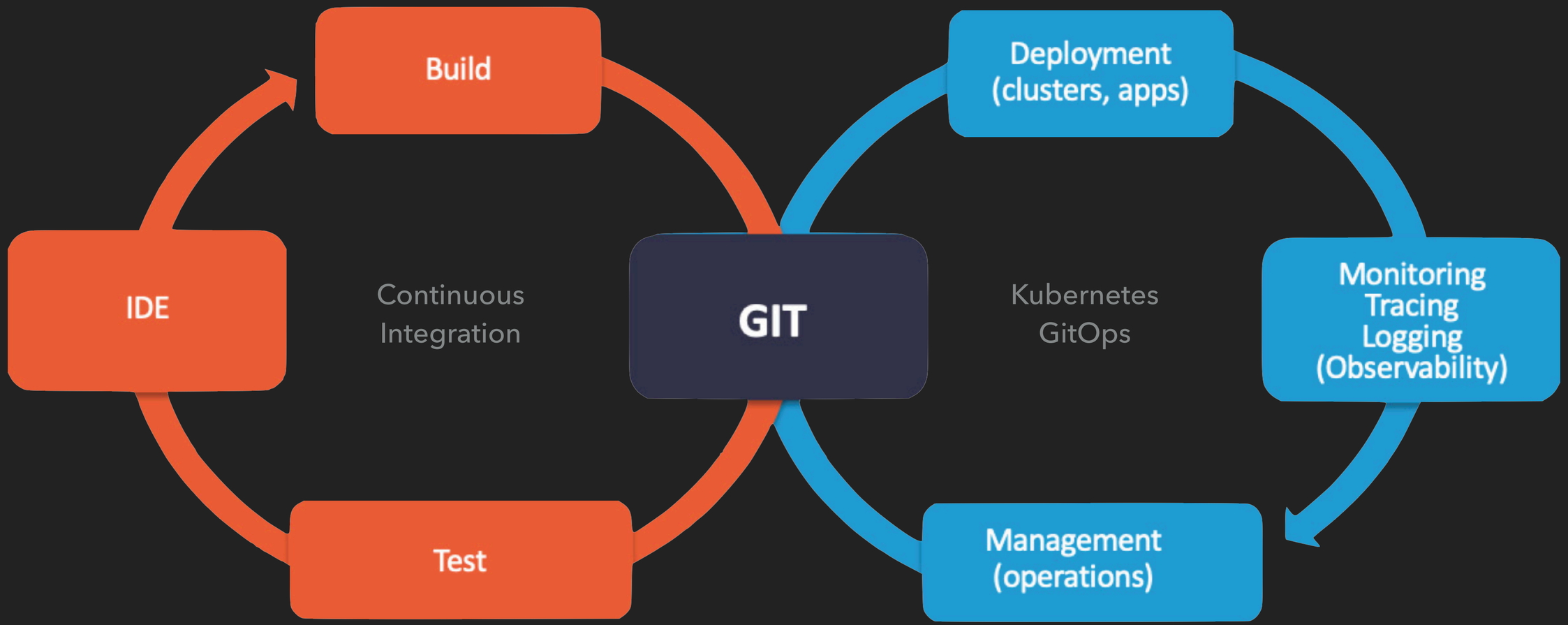
* I mean, it’s fine on dev. But certainly not on prod

DON'T INSTALL ANYTHING MANUALLY ON YOUR (PROD) CLUSTER

- Not reproducible
- Leads to "Snowflake" Deployments
- Hard to audit (what, when, and why?)
- Keeping a historical record of changes is hard
- Almost impossible to do proper secrets management

BUT HOW?

GIT-OPS TO THE RESCUE


Build
IDE
Test
Continuous
Integration
GIT
Deployment
(clusters, apps)
Monitoring
Tracing
Logging
(Observability)
Management
(operations)
Kubernetes
GitOps
DEVELOPMENT
DEPLOYMENT

PUSH

- Continuous Delivery Pipeline applies configuration to Kubernetes Cluster
- Requires your CD Pipeline to have access to the Cluster
- Does not detect configuration drift

PULL

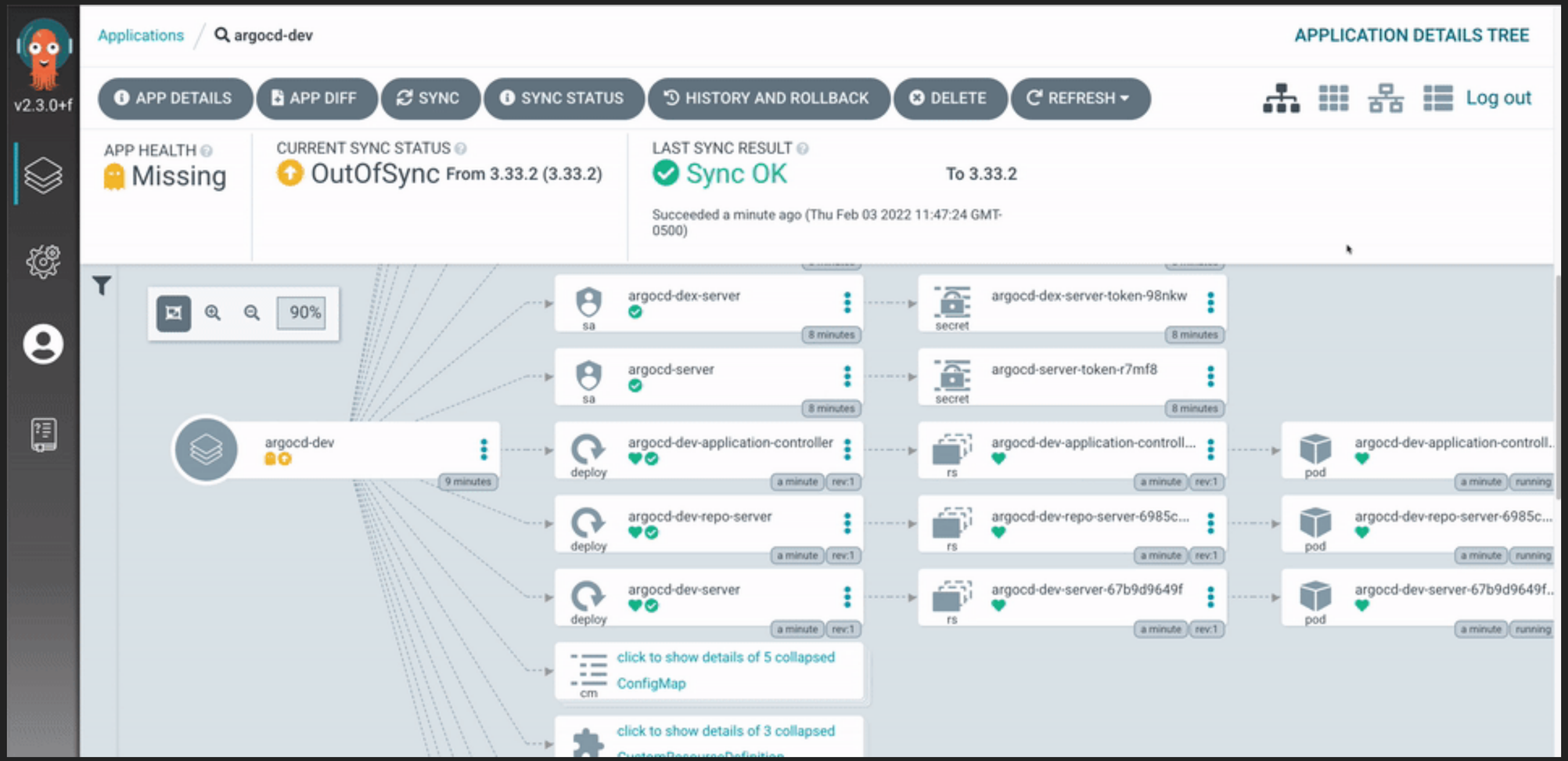
- GitOps Operator runs inside the Kubernetes Cluster and pulls changes from Git and applies them to the Cluster from the inside
- Requires the GitOps Operator to have access to the Repository
- Can detect configuration drift and revert manual changes

GIT-OPS

ARGO-CD

ARGO-CD

- Easy to use
- Declarative
- Nice looking management UI
- Easily extensible with your favorite secret management solution
- Avoid config drift
  - reconcile loop can roll-back manual changes automatically


v2.3.0+f
Applications
argocd-dev
APPLICATION DETAILS TREE
APP DETAILS
APP DIFF
SYNC
SYNC STATUS
HISTORY AND ROLLBACK
DELETE
REFRESH
Log out
APP HEALTH
Missing
CURRENT SYNC STATUS
OutOfSync From 3.33.2 (3.33.2)
LAST SYNC RESULT
Sync OK
To 3.33.2
Succeeded a minute ago (Thu Feb 03 2022 11:47:24 GMT-0500)
90%
argocd-dev
9 minutes
argocd-dex-server
sa
8 minutes
argocd-dex-server-token-98nkw
secret
8 minutes
argocd-server
sa
8 minutes
argocd-server-token-r7mf8
secret
8 minutes
argocd-dev-application-controller
deploy
a minute
rev:1
argocd-dev-application-controll...
rs
a minute
rev:1
argocd-dev-application-controll..
pod
a minute
running
argocd-dev-repo-server
deploy
a minute
rev:1
argocd-dev-repo-server-6985c...
rs
a minute
rev:1
argocd-dev-repo-server-6985c...
pod
a minute
running
argocd-dev-server
deploy
a minute
rev:1
argocd-dev-server-67b9d9649f
rs
a minute
rev:1
argocd-dev-server-67b9d9649f..
pod
a minute
running
click to show details of 5 collapsed ConfigMap
cm
click to show details of 3 collapsed

GIT-OPS

SECRET MANAGEMENT

SECRET MANAGEMENT

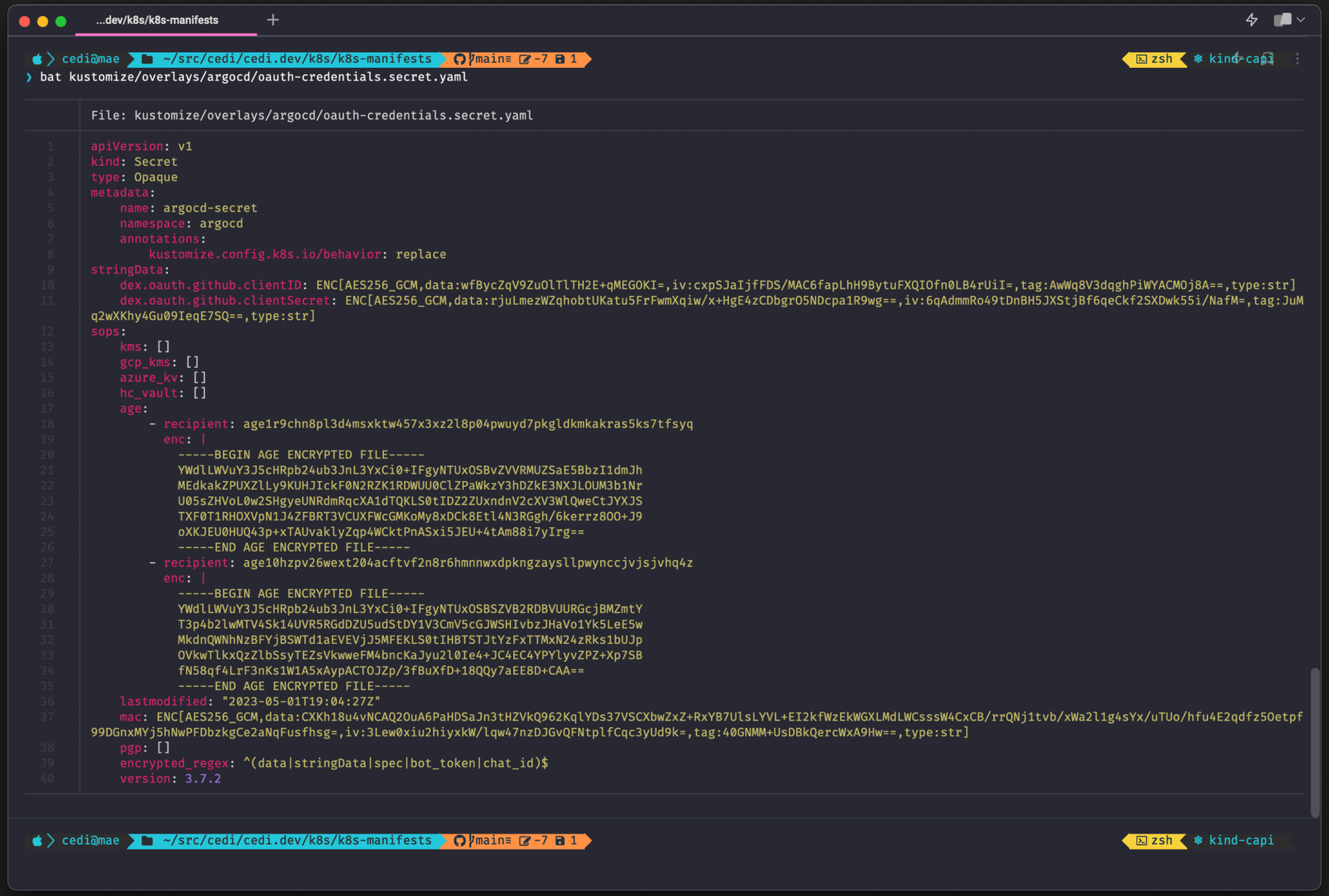
GIT-CRYPT

- Encrypts whole files
- Based on PGP
- Git-Diff won’t work at all

MOZILLA SOPS

- Encrypts only YAML Value
- Can use a multitude of credential providers and encryption methods (PGP, AGE, Vault, …)
- Git-diff works (to some extent)

SOPS


...dev/k8s/k8s-manifests
cedi@mae ~/src/cedi/cedi.dev/k8s/k8s-manifests main≡ -7 1
zsh kind-capi
bat kustomize/overlays/argocd/oauth-credentials.secret.yaml
File: kustomize/overlays/argocd/oauth-credentials.secret.yaml
apiVersion: v1
kind: Secret
type: Opaque
metadata:
name: argocd-secret
namespace: argocd
annotations:
kustomize.config.k8s.io/behavior: replace
stringData:
dex.oauth.github.clientID: ENC[AES256_GCM,data:wfBycZqV9ZuOlTlTH2E+qMEGOKI=,iv:cxpSJaIjfFDS/MAC6fapLhH9BytuFXQIOfn0LB4rUiI=,tag:AwWq8V3dqghPiWYACMOj8A==,type:str]
dex.oauth.github.clientSecret: ENC[AES256_GCM,data:rjuLmezWZqhobtUKatu5FrFwmXqiw/x+HgE4zCDbgrO5NDcpa1R9wg==,iv:6qAdmmRo49tDnBH5JXStjBf6qeCkf2SXDwk55i/NafM=,tag:JuMq2wXKhy4Gu09IeqE7SQ==,type:str]
sops:
kms: []
gcp_kms: []
azure_kv: []
hc_vault: []
age:
- recipient: age1r9chn8pl3d4msxktw457x3xz2l8p04pwuyd7pkgldkmkakras5ks7tfsyq
enc: |
-----BEGIN AGE ENCRYPTED FILE-----
YWdlLWVuY3J5cHRpb24ub3JnL3YxCi0+IFgyNTUxOSBvZVVRMUZSaE5BbzI1dmJh
MEdkakZPUXZlLy9KUHJIckF0N2RZK1RDWUU0ClZPaWkzY3hDZkE3NXJLOUM3b1Nr
U05sZHVoL0w2SHgyeUNRdmRqcXA1dTQKLS0tIDZ2ZUxndnV2cXV3WlQweCtJYXJS
TXF0T1RHOXVpN1J4ZFBRT3VCUXFWcGMKoMy8xDCk8Etl4N3RGgh/6kerrz8OO+J9
oXKJEU0HUQ43p+xTAUvaklyZqp4WCktPnASxi5JEU+4tAm88i7yIrg==
-----END AGE ENCRYPTED FILE-----
- recipient: age10hzpv26wext204acftvf2n8r6hmnnwxdpkngzaysllpwynccjvjsjvhq4z
enc: |
-----BEGIN AGE ENCRYPTED FILE-----
YWdlLWVuY3J5cHRpb24ub3JnL3YxCi0+IFgyNTUxOSBSZVB2RDBVUURGcjBMZmtY
T3p4b2lwMTV4Sk14UVR5RGdDZU5udStDY1V3CmV5cGJWSHIvbzJHaVo1Yk5LeE5w
MkdnQWNhNzBFYjBSWTd1aEVEVjJ5MFEKLS0tIHBTSTJtYzFxTTMxN24zRks1bUJp
OVkwTlkxQzZlbSsyTEZsVkwweFM4bncKaJyu2l0Ie4+JC4EC4YPYlyvZPZ+Xp7SB
fN58qf4LrF3nKs1W1A5xAypACTOJZp/3fBuXfD+18QQy7aEE8D+CAA==
-----END AGE ENCRYPTED FILE-----
lastmodified: "2023-05-01T19:04:27Z"
mac: ENC[AES256_GCM,data:CXKh18u4vNCAQ2OuA6PaHDSaJn3tHZVkQ962KqlYDs37VSCXbwZxZ+RxYB7UlsLYVL+EI2kfWzEkWGXLMdLWCsssW4CxCB/rrQNj1tvb/xWa2l1g4sYx/uTUo/hfu4E2qdfz5Oetpf99DGnxMYj5hNwPFDbzkgCe2aNqFusfhsg=,iv:3Lew0xiu2hiyxkW/lqw47nzDJGvQFNtplfCqc3yUd9k=,tag:40GNMM+UsDBkQercWxA9Hw==,type:str]
pgp: []
encrypted_regex: ^(data|stringData|spec|bot_token|chat_id)$
version: 3.7.2
cedi@mae ~/src/cedi/cedi.dev/k8s/k8s-manifests main≡ -7 1
zsh kind-capi

NETWORKING

HOW TO SERVE YOUR APPLICATION

THE ABSOLUTE BASICS

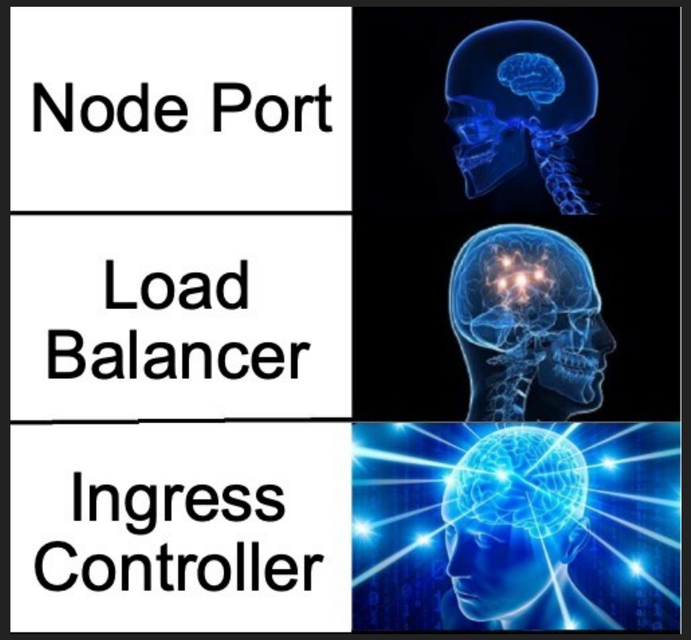
- Pod-Spec exposes a Port
- A Service selects Pods via labels
- Service specifies an exposed Port
- The Service ensures traffic is served to the Pod on the Pods Port

SERVICES

```yaml
apiVersion: v1
kind: Service
metadata:
  name: kuard-service
spec:
  selector:
    app: kuard
  ports:
  - port: 80
    targetPort: 8080
```

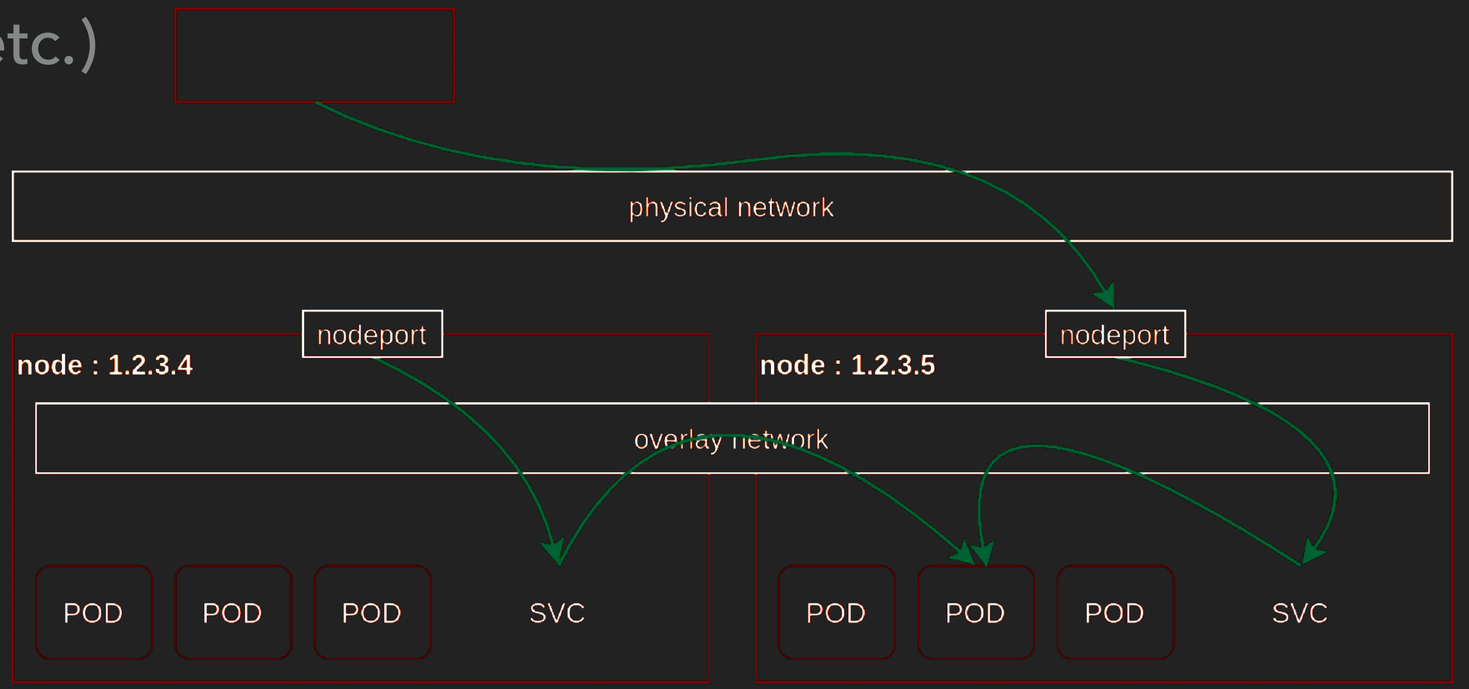
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kuard-deployment
  labels:
    app: kuard
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuard
  template:
    metadata:
      labels:
        app: kuard
    spec:
      containers:
        - image: gcr.io/kuar-demo/kuard-amd64:blue
          name: kuard
          ports:
            - containerPort: 8080
              name: http
```

HOW TO SERVE CUSTOMER TRAFFIC



EASY MODE: NODE PORT

- No Load-Balancing and auto-failover
- Available Node-Port Range is 30000-32767
- No reserved ports (80, 443, etc.)



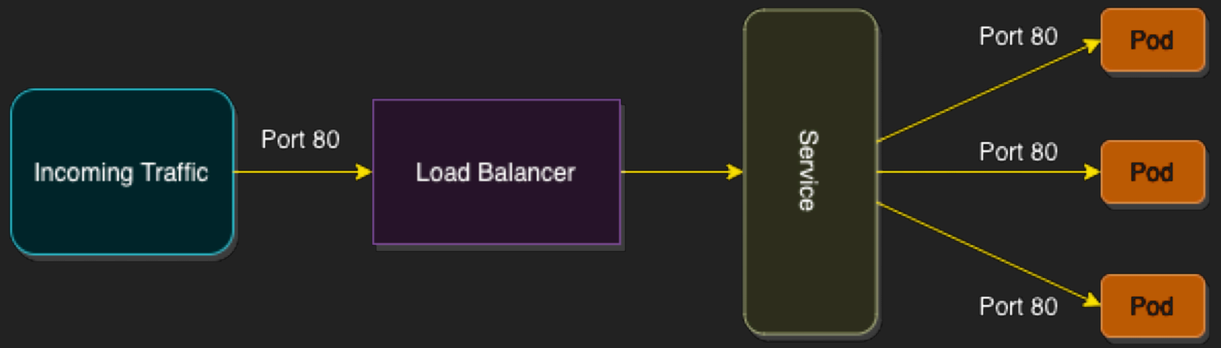
NODE-PORT

```yaml
apiVersion: v1
kind: Service
metadata:
  name: kuard-service
spec:
  selector:
    app: kuard
  ports:
  - port: 80
    targetPort: 8080
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kuard-deployment
  labels:
    app: kuard
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuard
  template:
    metadata:
      labels:
        app: kuard
    spec:
      containers:
        - image: gcr.io/kuar-demo/kuard-amd64:blue
          name: kuard
          ports:
            - containerPort: 8080
              name: http
```

ADVANCED MODE: LOADBALANCER

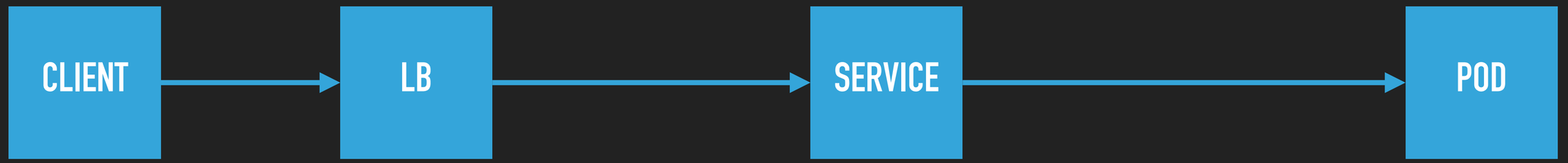
- Expensive (usually each LoadBalancer is billed separately, for existence & traffic)
- Certificate management is limited by the Cloud-Provider capabilities
- Requires a Cloud-Provider with LoadBalancer support
- TLS usually terminated at the LB
- Only works for domains, not paths



LOADBALANCER

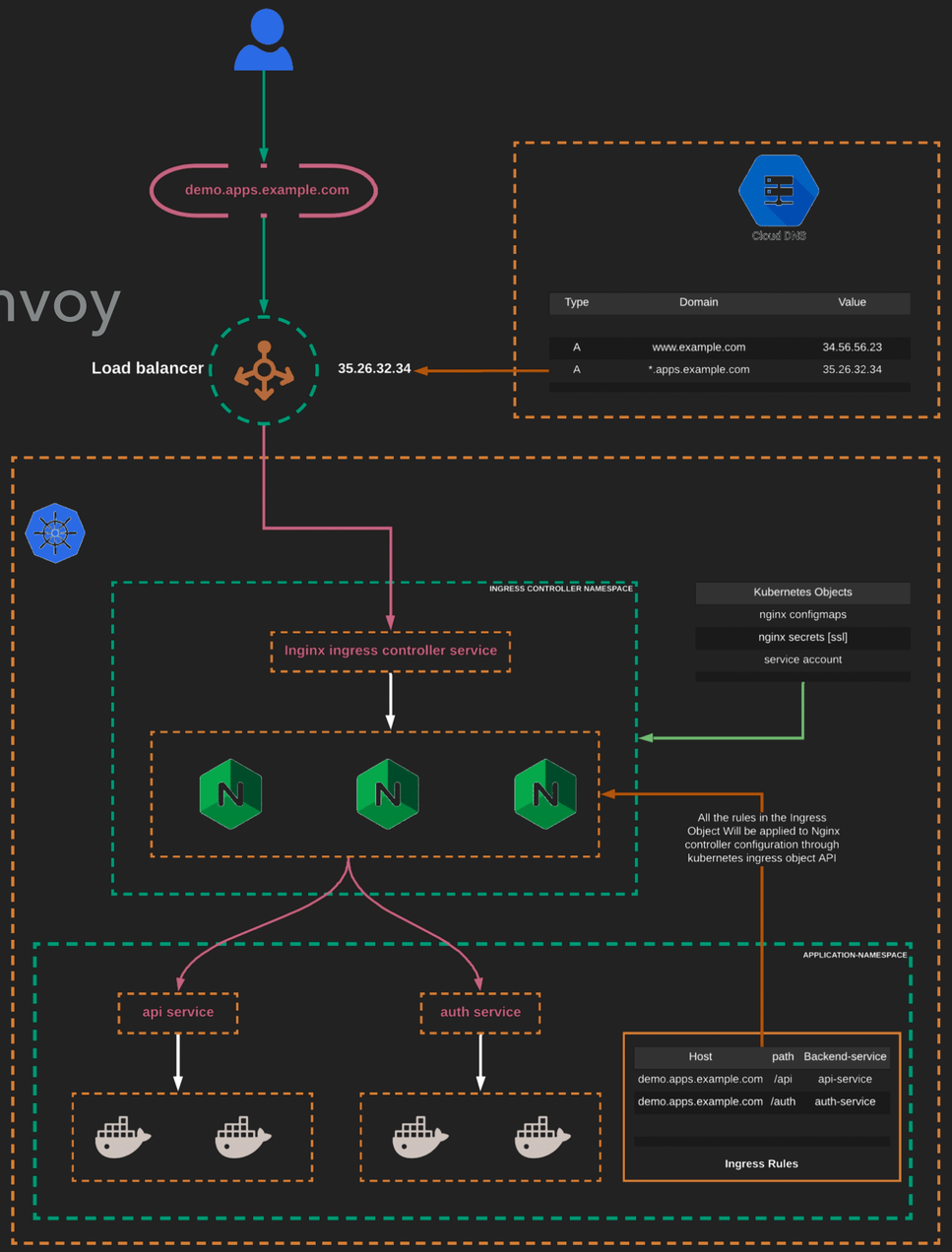
```yaml
apiVersion: v1
kind: Service
metadata:
  name: kuard-service
spec:
  selector:
    app: kuard
  ports:
  - port: 80
    targetPort: 8080
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kuard-deployment
  labels:
    app: kuard
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuard
  template:
    metadata:
      labels:
        app: kuard
    spec:
      containers:
        - image: gcr.io/kuar-demo/kuard-amd64:blue
          name: kuard
          ports:
            - containerPort: 8080
              name: http
```



EXPERT LEVEL: INGRESS CONTROLLER

- 99,9% of the time simply use ingress-nginx or envoy
- Domain Auto discovery
- Certificates can be issued automatically
- High performance
- Works on URL paths



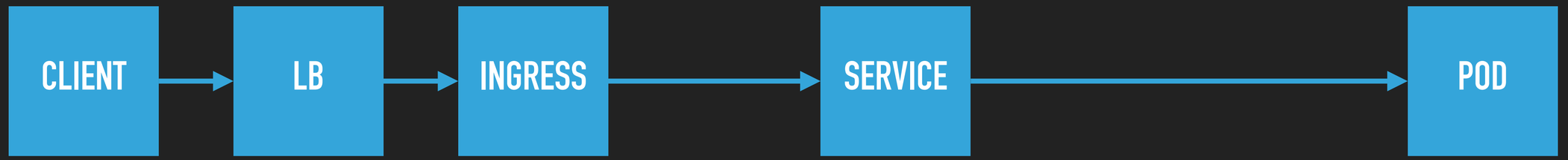
envoy

INGRESS

```yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tailnet-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: nginx
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
spec:
  rules:
    - host: test.cedi.dev
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: kuard-service
              port:
                number: 80
  tls:
    - secretName: test-ts-secret
      hosts:
        - test.cedi.dev
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: kuard-service
spec:
  selector:
    app: kuard
  ports:
  - port: 80
    targetPort: 8080
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kuard-deployment
  labels:
    app: kuard
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuard
  template:
    metadata:
      labels:
        app: kuard
    spec:
      containers:
        - image: gcr.io/kuar-demo/kuard-amd64:blue
          name: kuard
          ports:
            - containerPort: 8080
              name: http
```



WRAP-UP

- Take a Pi, install K3s
- Deploy your workload with Helm and or Kustomize
- Secure your secrets with SOPS and don’t leak them to Git
- Keep your Cluster reconciled with ArgoCD
- Use ingress-nginx to serve incoming requests

CONGRATS, NOW YOU HAVE A K8S CLUSTER TO MAINTAIN

/archang3l_media /in/heikoborchers

@archang3l_media@chaos.social

/cedi /in/cekienzl

@cedi@hachyderm.io

THANKS FOR LISTENING!

Q&A TIME

RELATED TALKS AND TOPICS

- Decoding Site Reliability Engineering: An Exploration of SRE, DevOps, and Platform Engineering
  - ccl.pw/decoding-sre
- Modern Observability - Scalable Observability with the LGTM Stack: Harnessing the Power of Loki, Grafana, Tempo, and Mimir
  - ccl.pw/modern-o11y
- Understanding Alerting - How to come up with a good enough alerting strategy (GPN 20)
  - ccl.pw/alerting
- Kubernetes, the good, the bad and the Ugly (GPN 20)
  - ccl.pw/k8s-the-good-the-bad-the-ugly