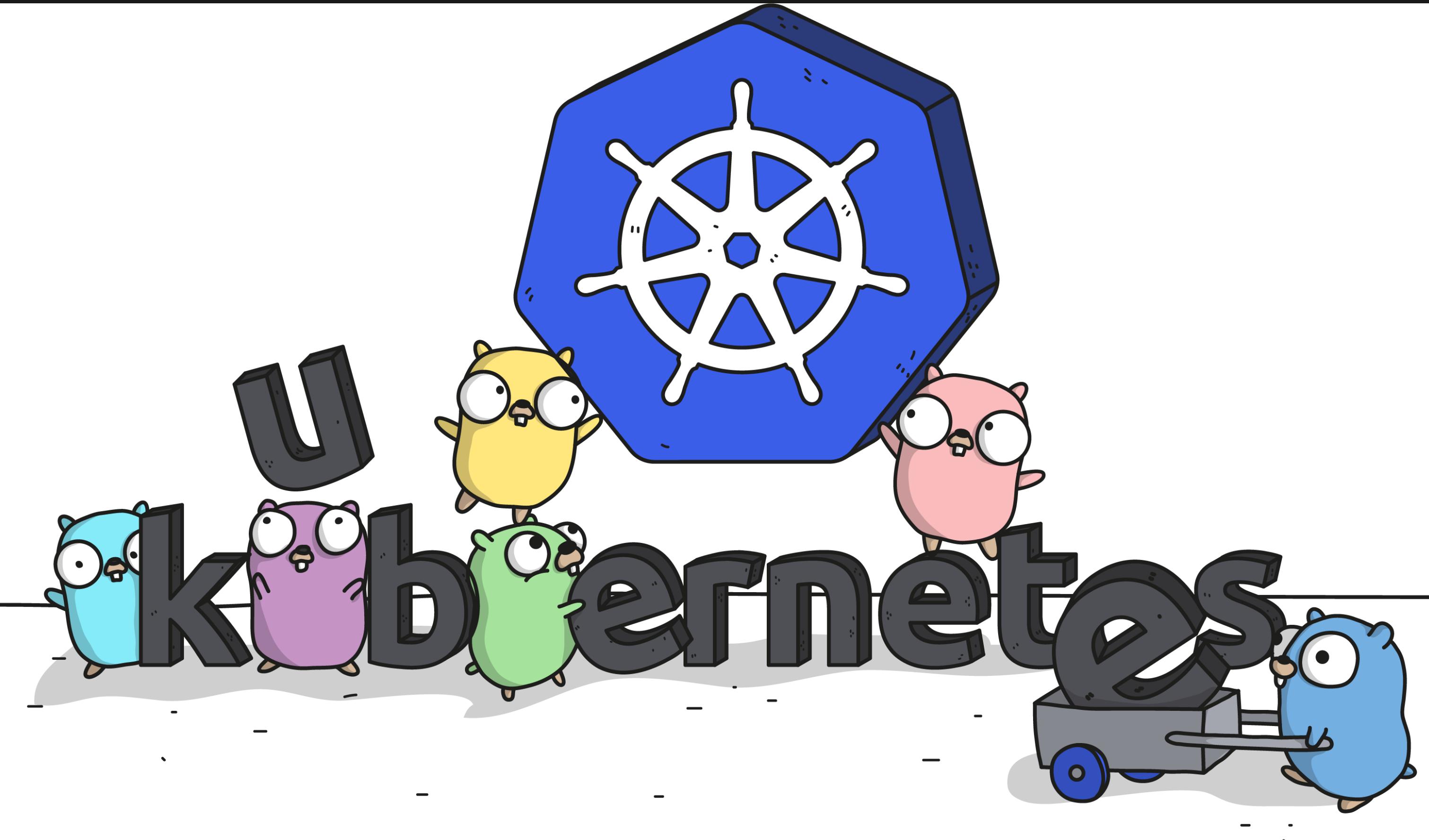


# KUBERNETES INTRODUCTION



Wojciech Barczyński  
[wbarczynski+trainings@gmail.com](mailto:wbarczynski+trainings@gmail.com)

# High delivery performance

- Lead Time
- Deployment frequency
- Mean time to Recovery
- Change Fail Percent



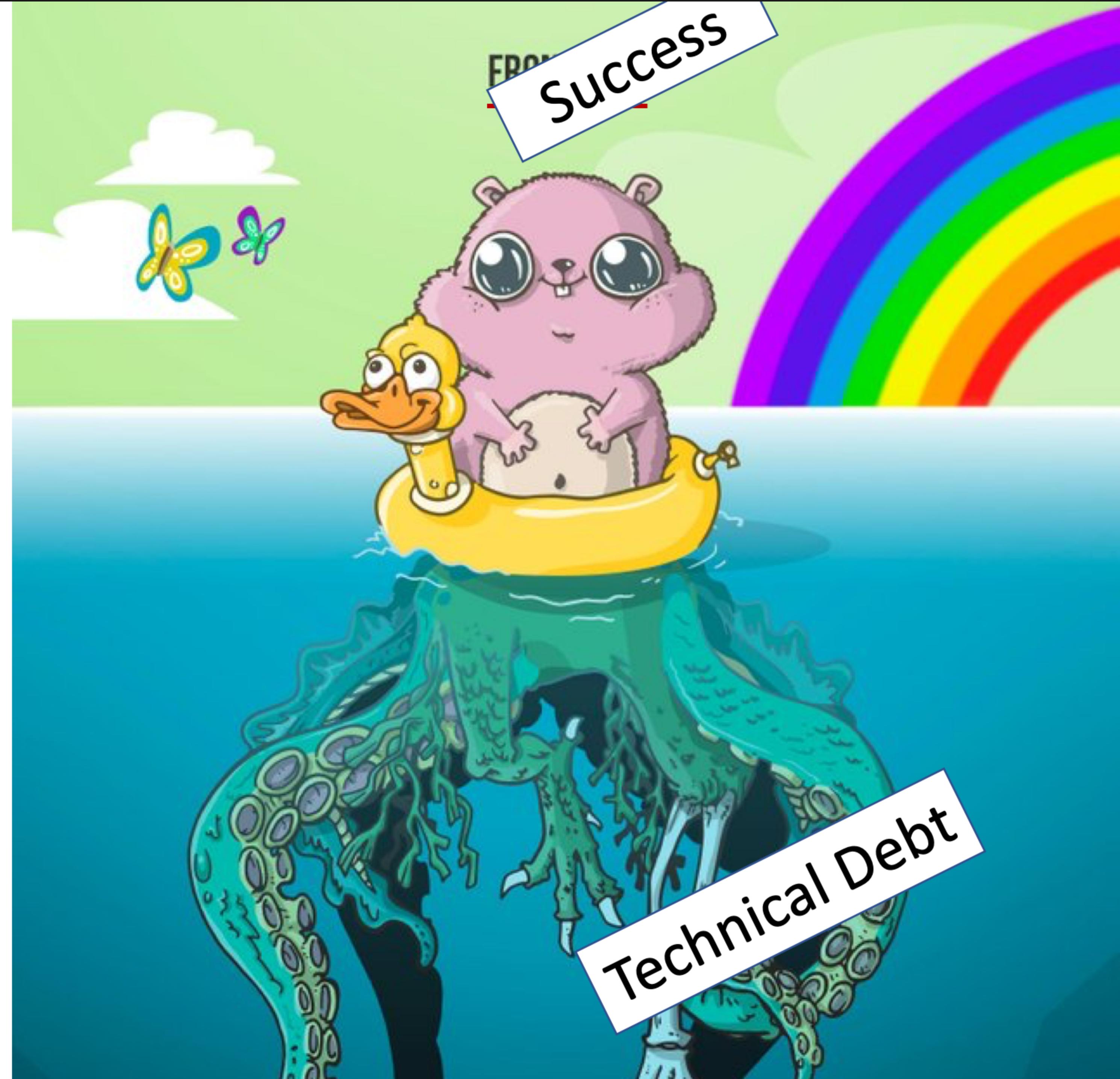
# STORY

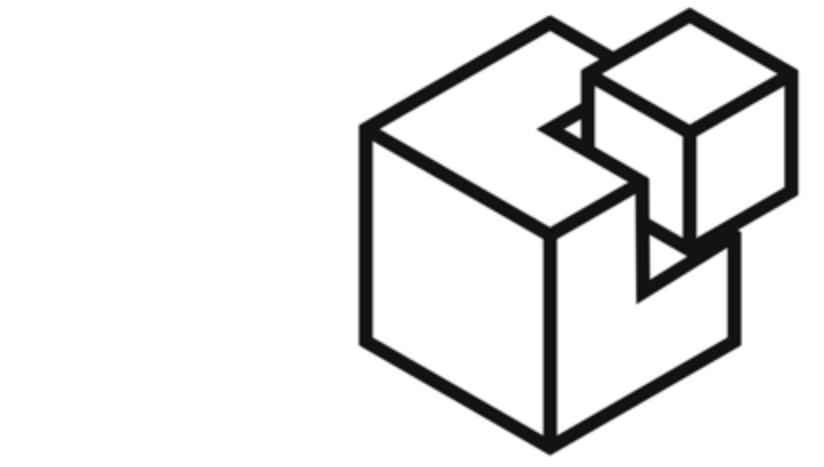
Kubernetes + Go + ...

- Codility
- SMACC/Hypatos - Fintech / ML - since 2017
- Lyke - Mobile Fashion app - since 2016

I do not like Infra

Slow delivery  
Continuous Deployment?  
Fear  
Frustration  
XX% Idle Machines





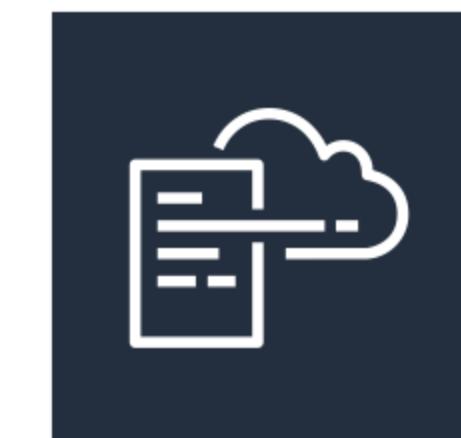
SALTSTACK



ANSIBLE



CHEF



AWS  
CloudFormation

2016



HashiCorp  
**Terraform**

# Black (Blue) Box

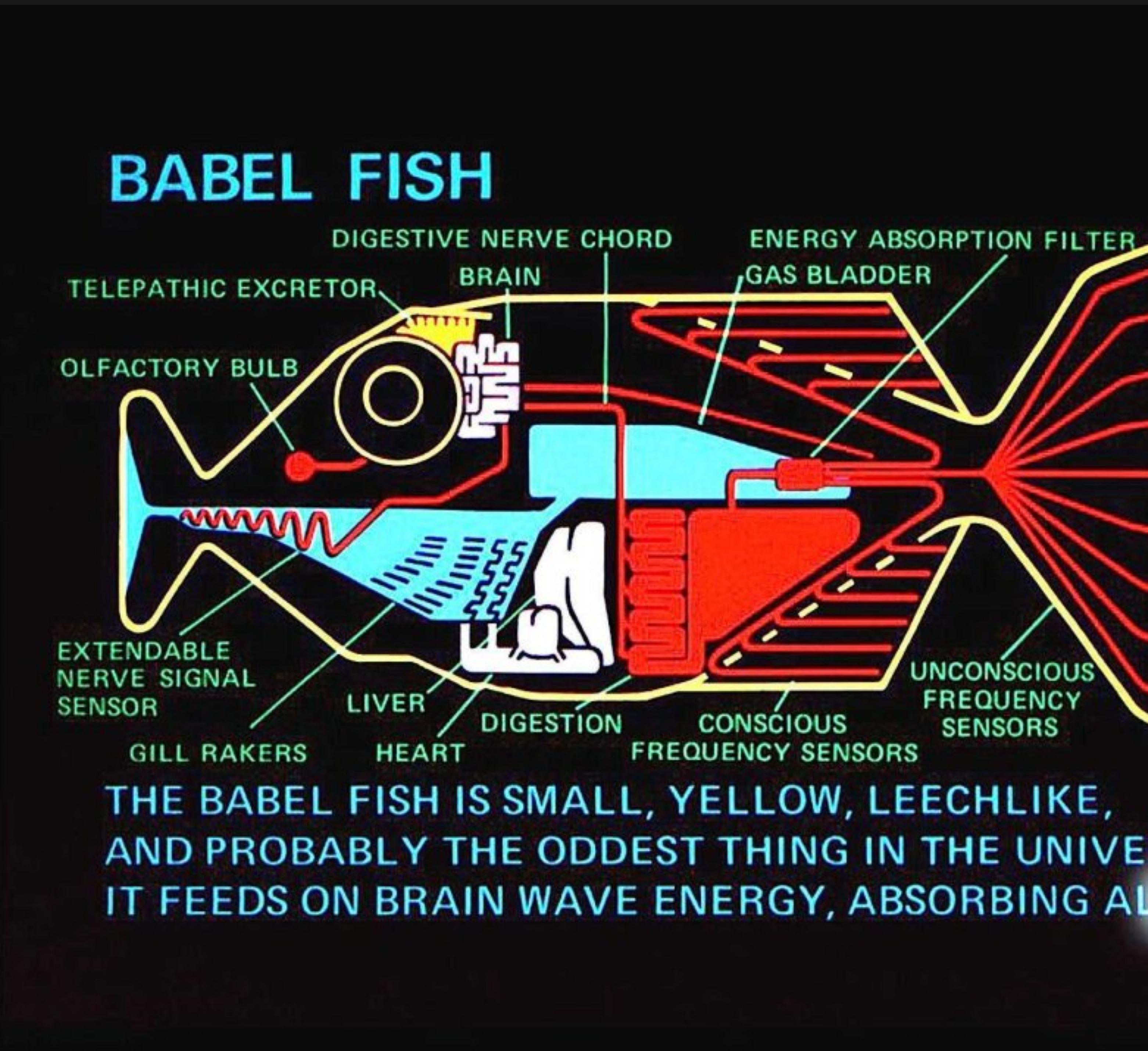
## Infrastructure (almost) invisible

## Easy\* Continuous Deployment



[https://en.wikipedia.org/wiki/File:Dr\\_Who\\_\(316350537\).jpg](https://en.wikipedia.org/wiki/File:Dr_Who_(316350537).jpg)

Common  
Language  
Artifacts  
Platform

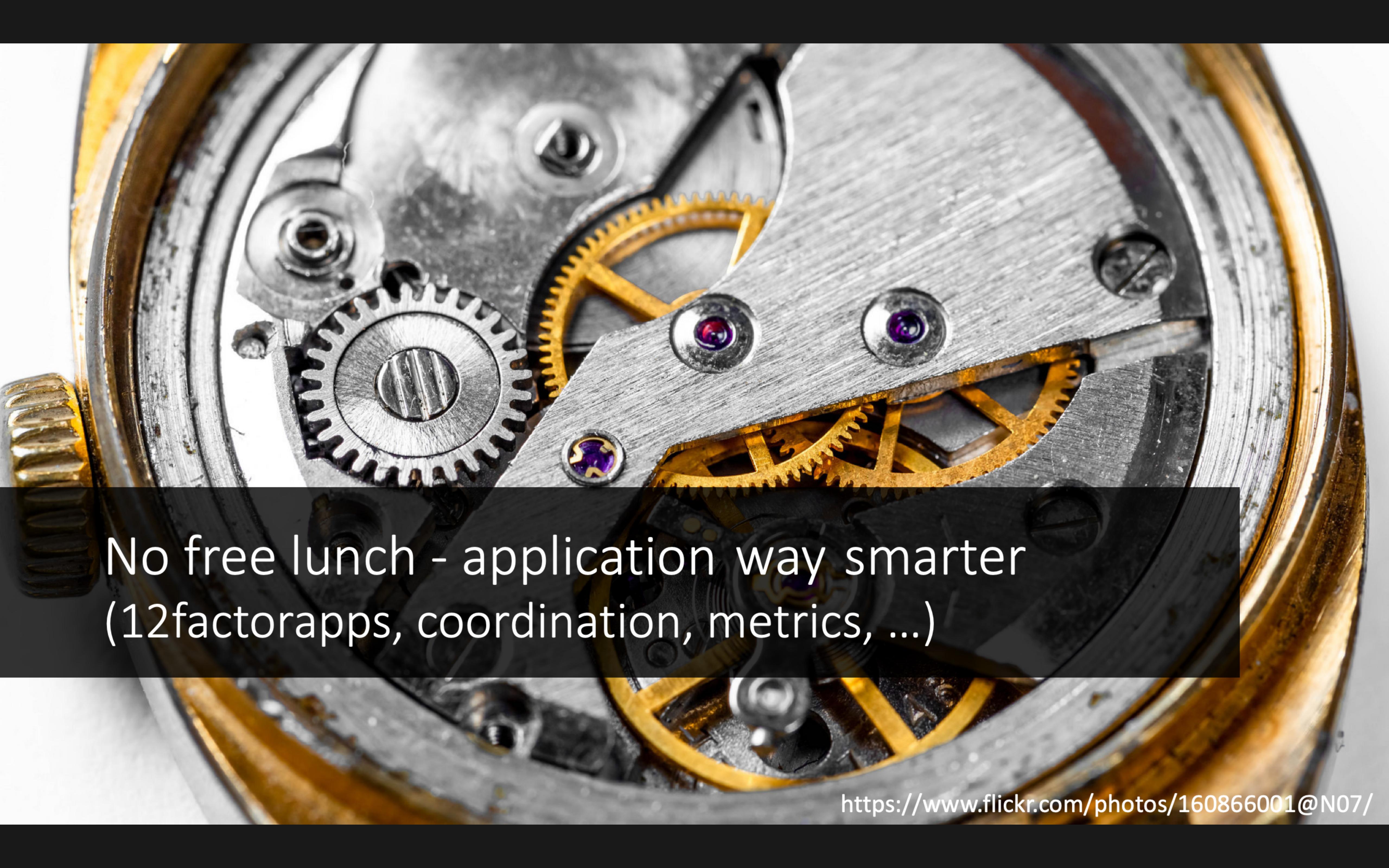


# Learn-as-you-go

1. Deploy Cloud-Native app
2. Make a Hell of Mistakes
3. Get it right or Postpone



envoy

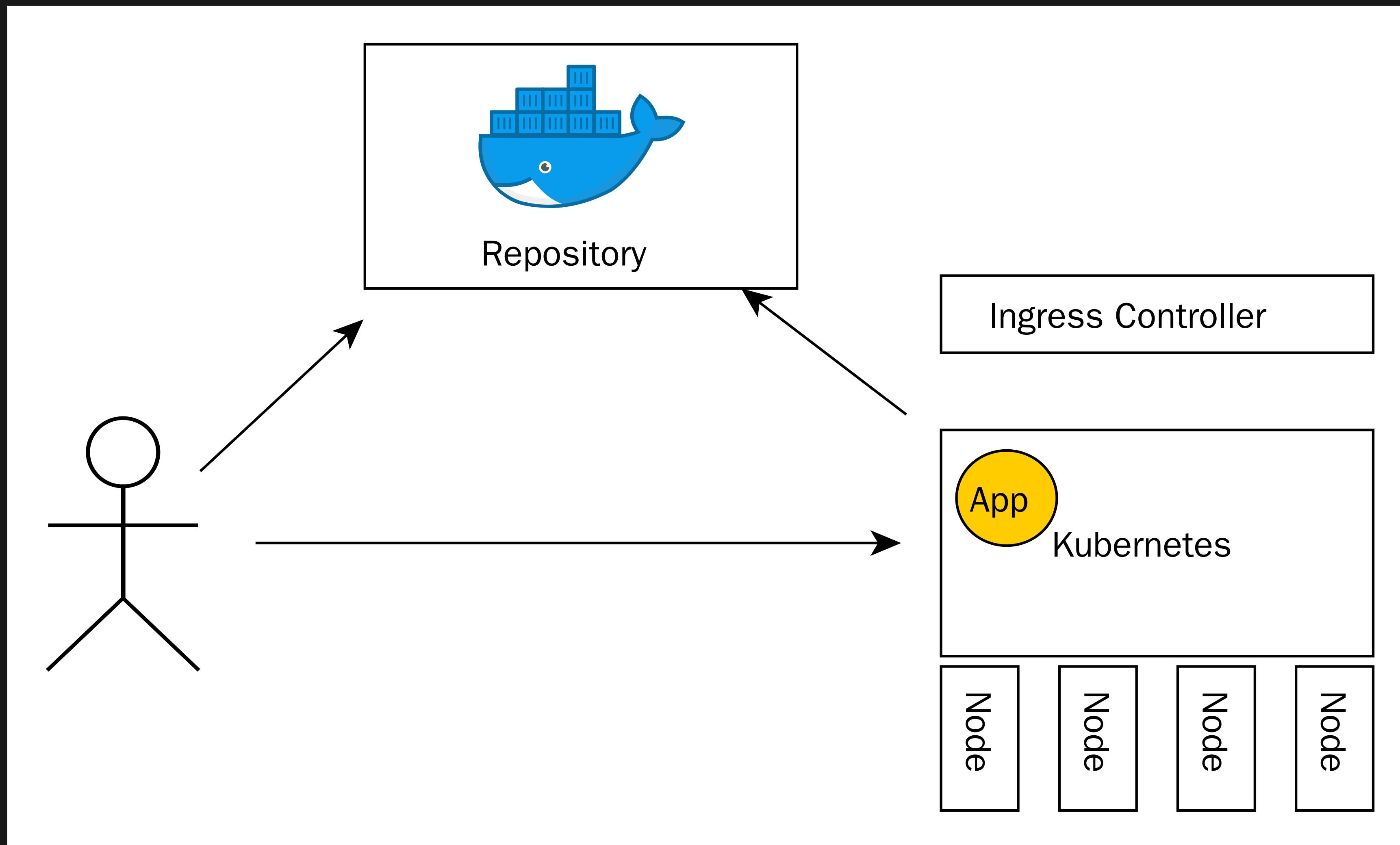
A close-up photograph of a mechanical watch's internal mechanism. The image shows various metal gears, some with gold-colored plating, meshing together. Bright blue and red gemstones, likely rubies, are used as bearings for the gears. The background is dark, making the metallic components stand out.

No free lunch - application way smarter  
(12factorapps, coordination, metrics, ...)

# KUBERNETES

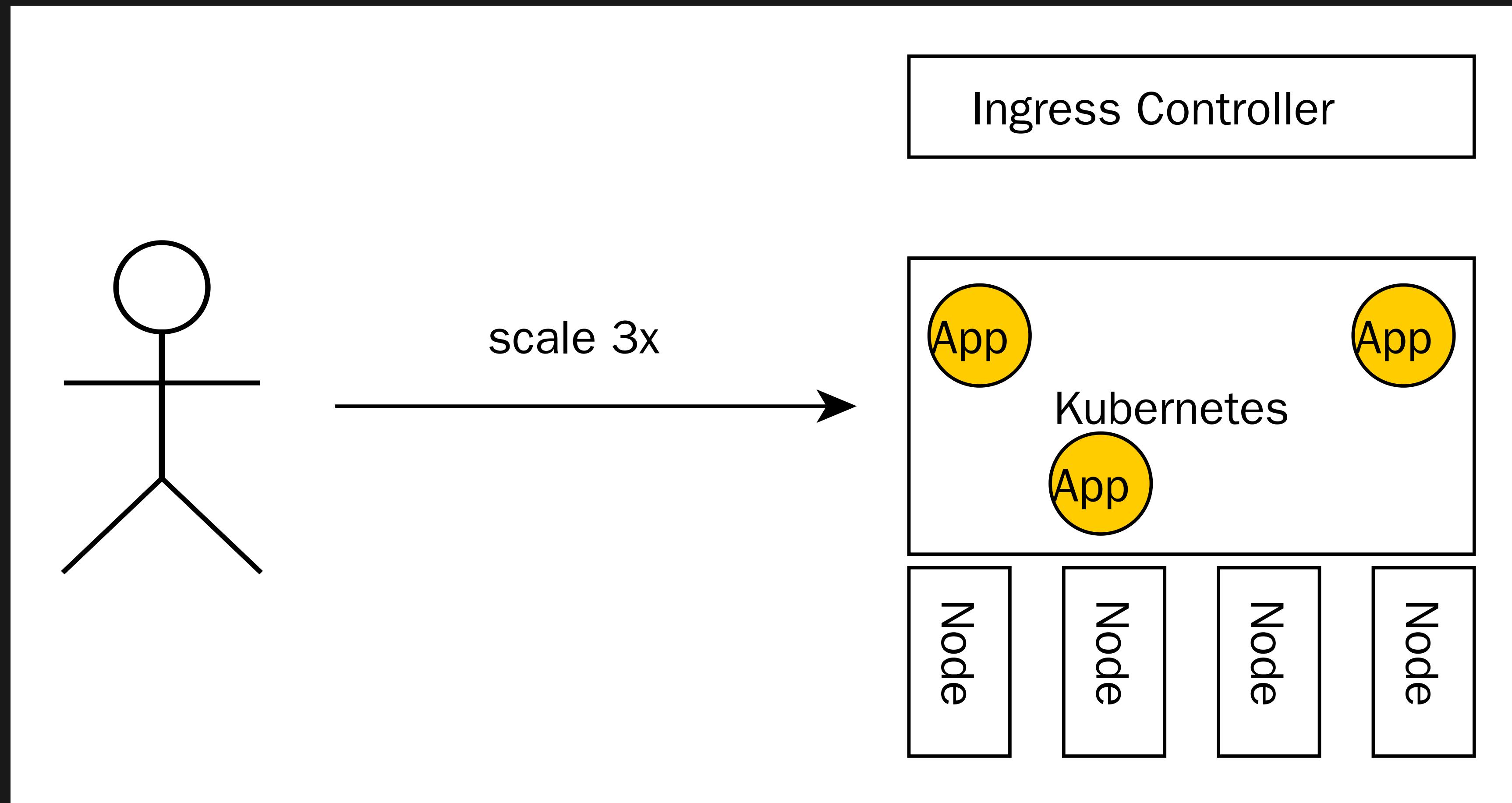
- Container management
- Battery for 12factor apps
- ...heading to integration platform
- ...becoming a framework for your Xubernetes

# KUBERNETES



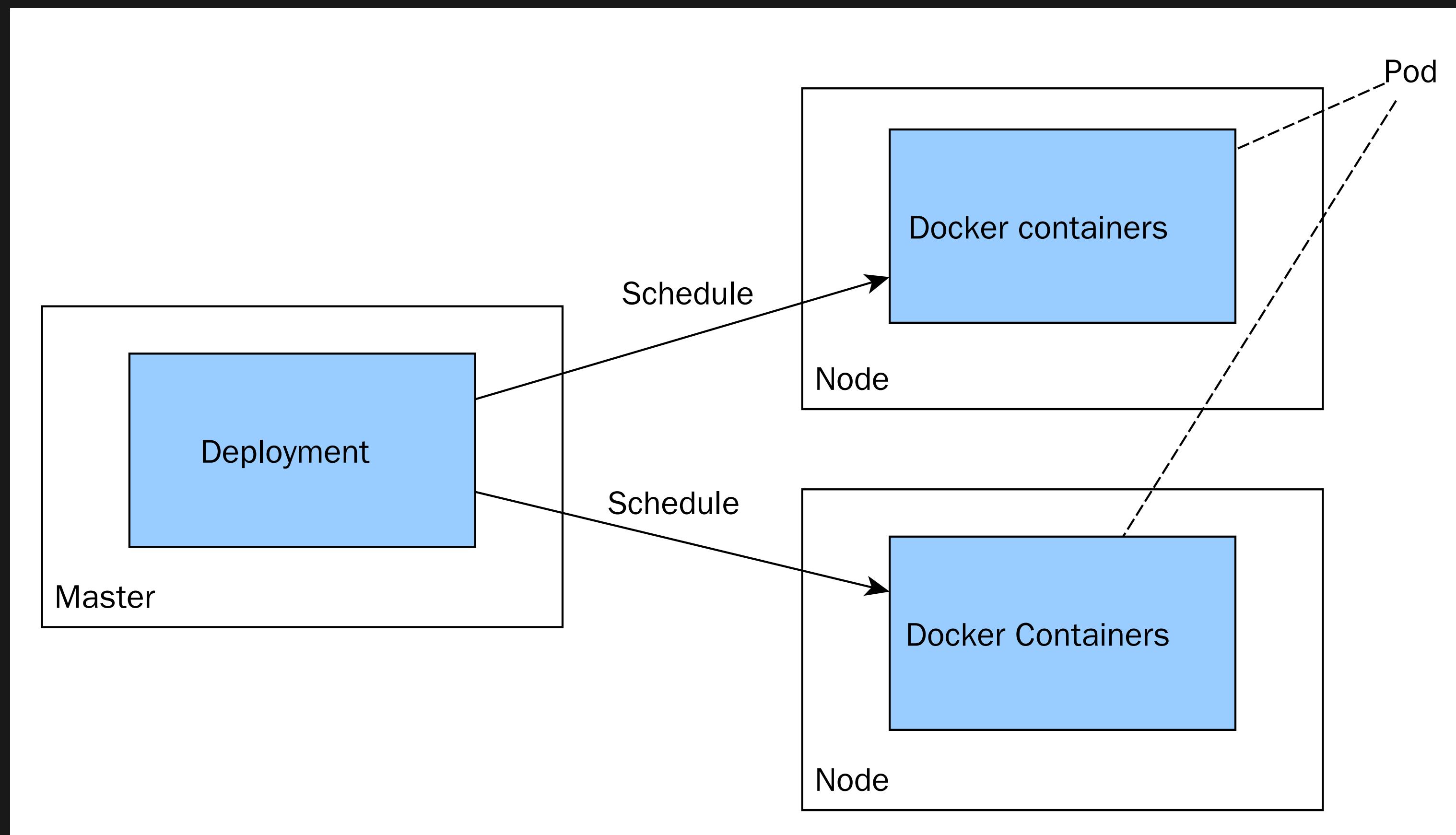
```
make docker_push; kubectl create -f app-srv-dpl.yaml
```

# SCALE UP! SCALE DOWN!

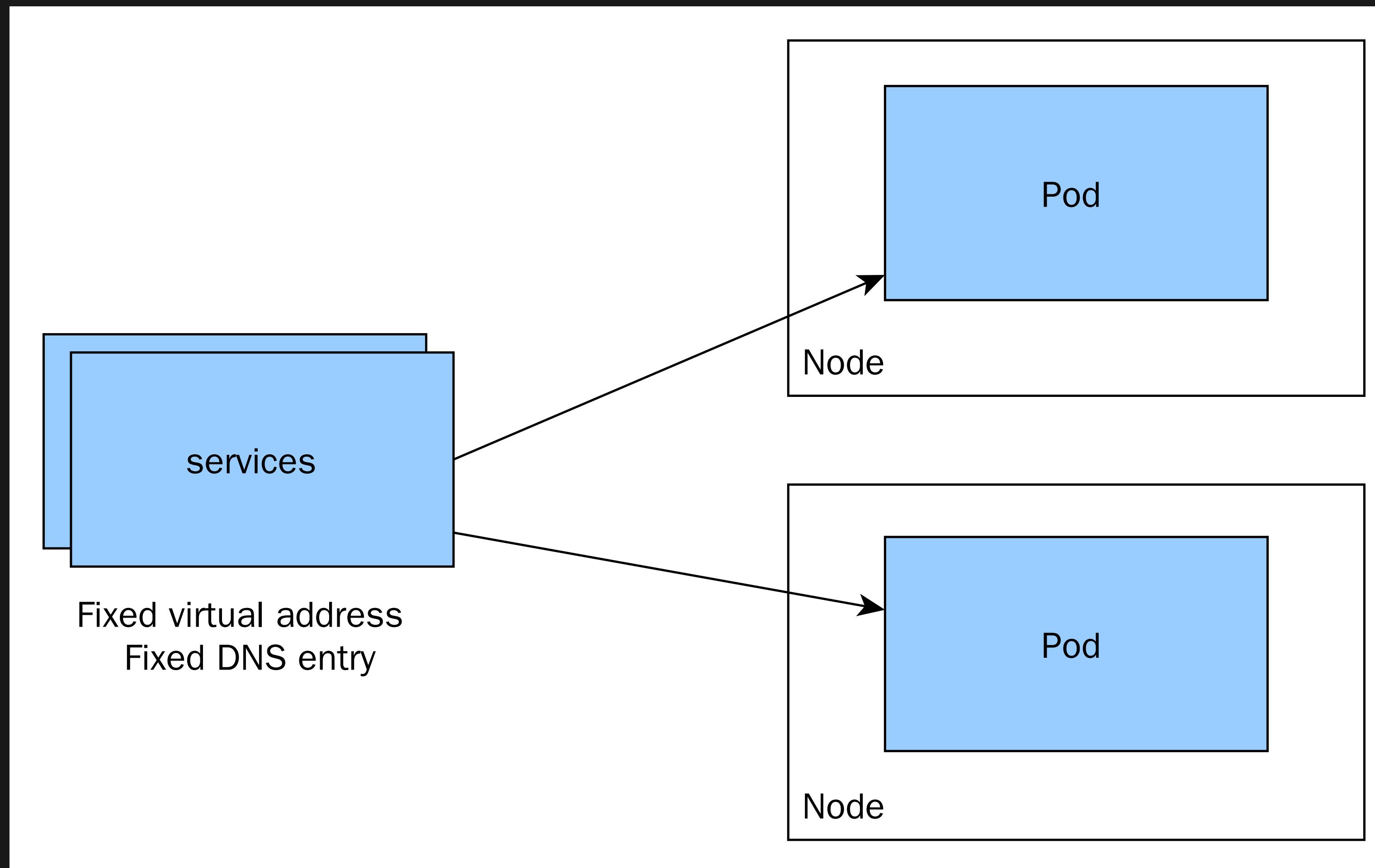


```
kubectl --replicas=3 -f app-srv-dpl.yaml
```

# DEPLOYMENT AND PODS



# SERVICE AND PODS



Service matches pods based on labels

# INGRESS

Pattern

api.example.io/v1/users

Target App Service

users-v1

---

api.example.io/v2/users

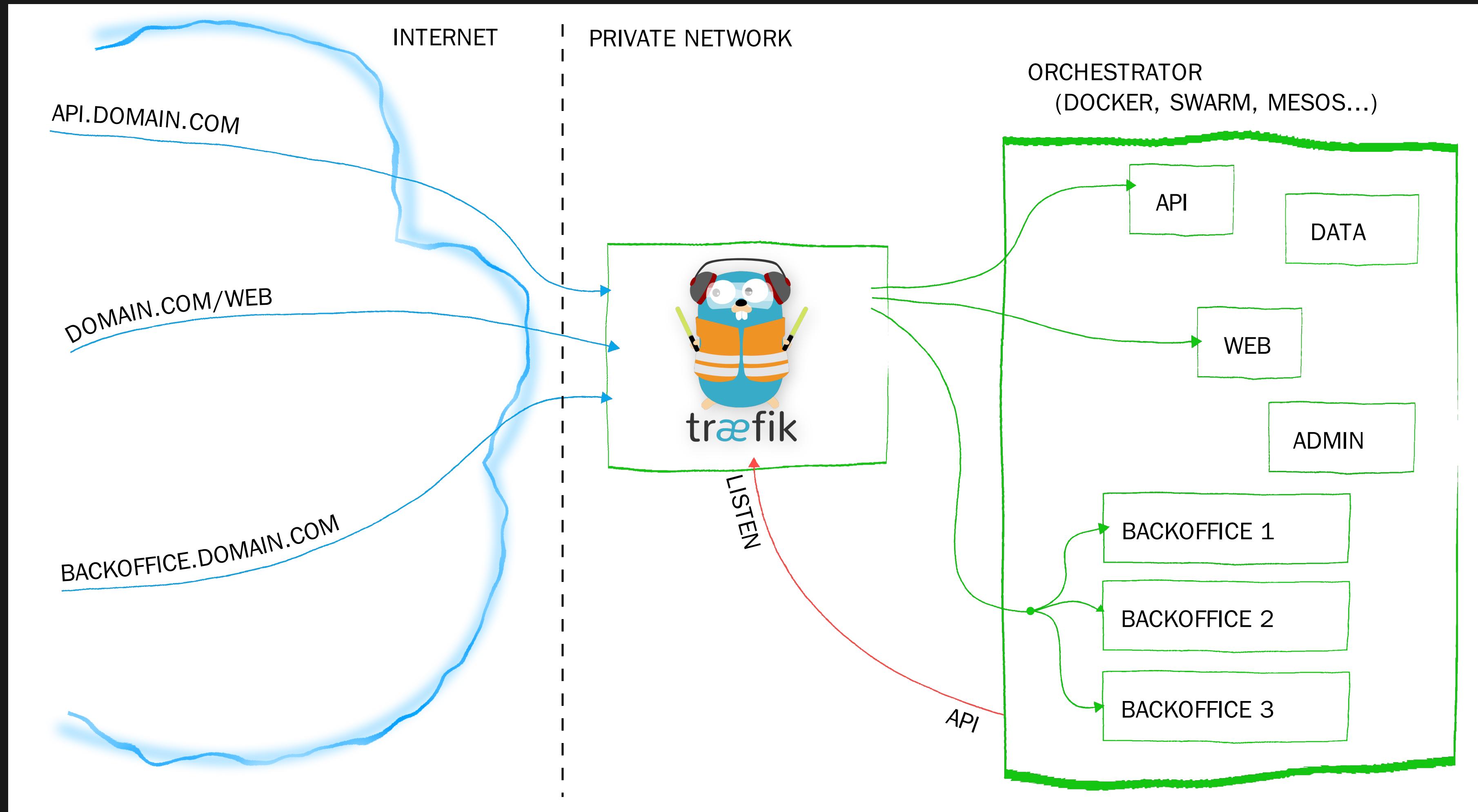
users-v2

---

example.io

web

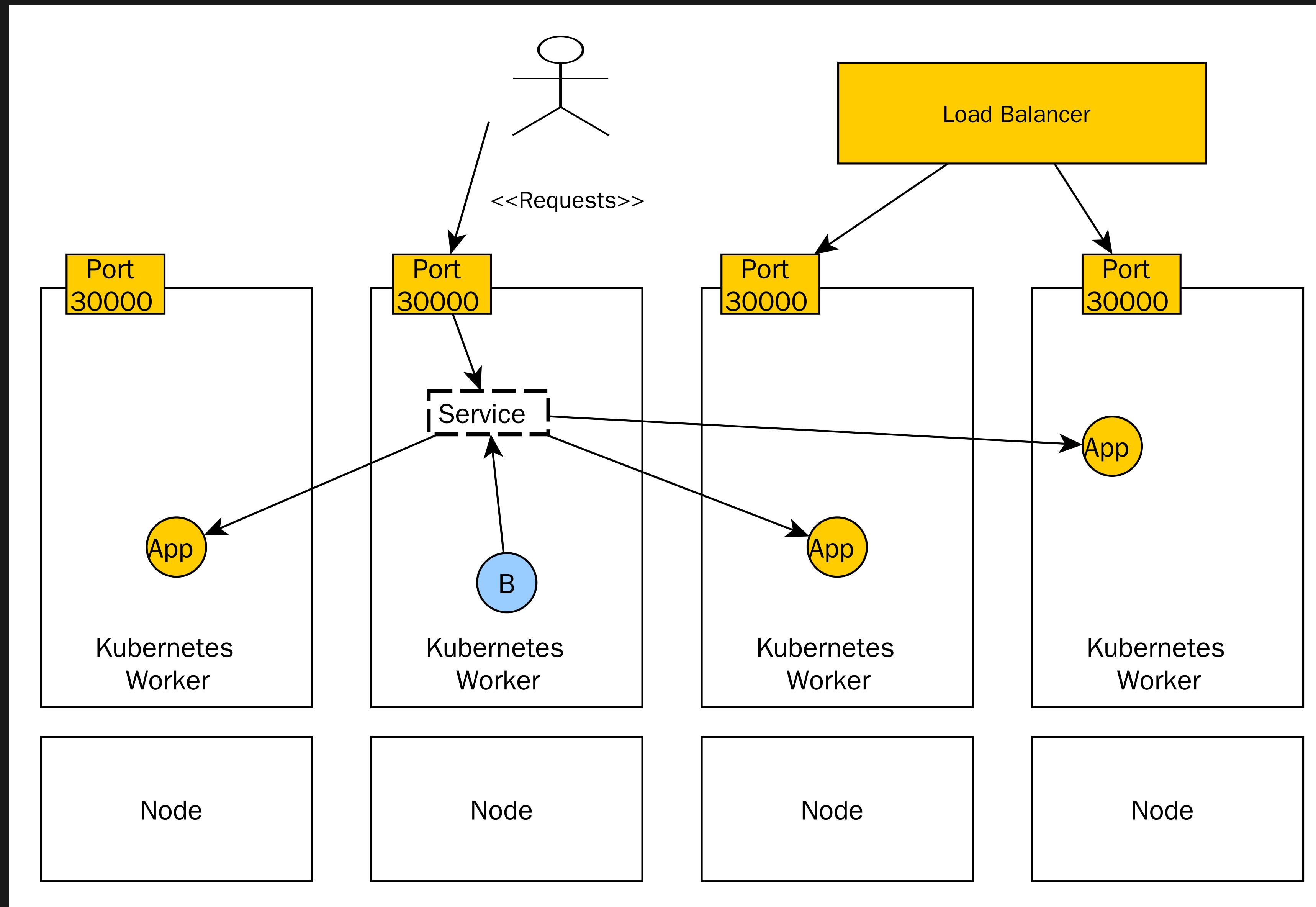
# INGRESS CONTROLLER



## BASIC CONCEPTS

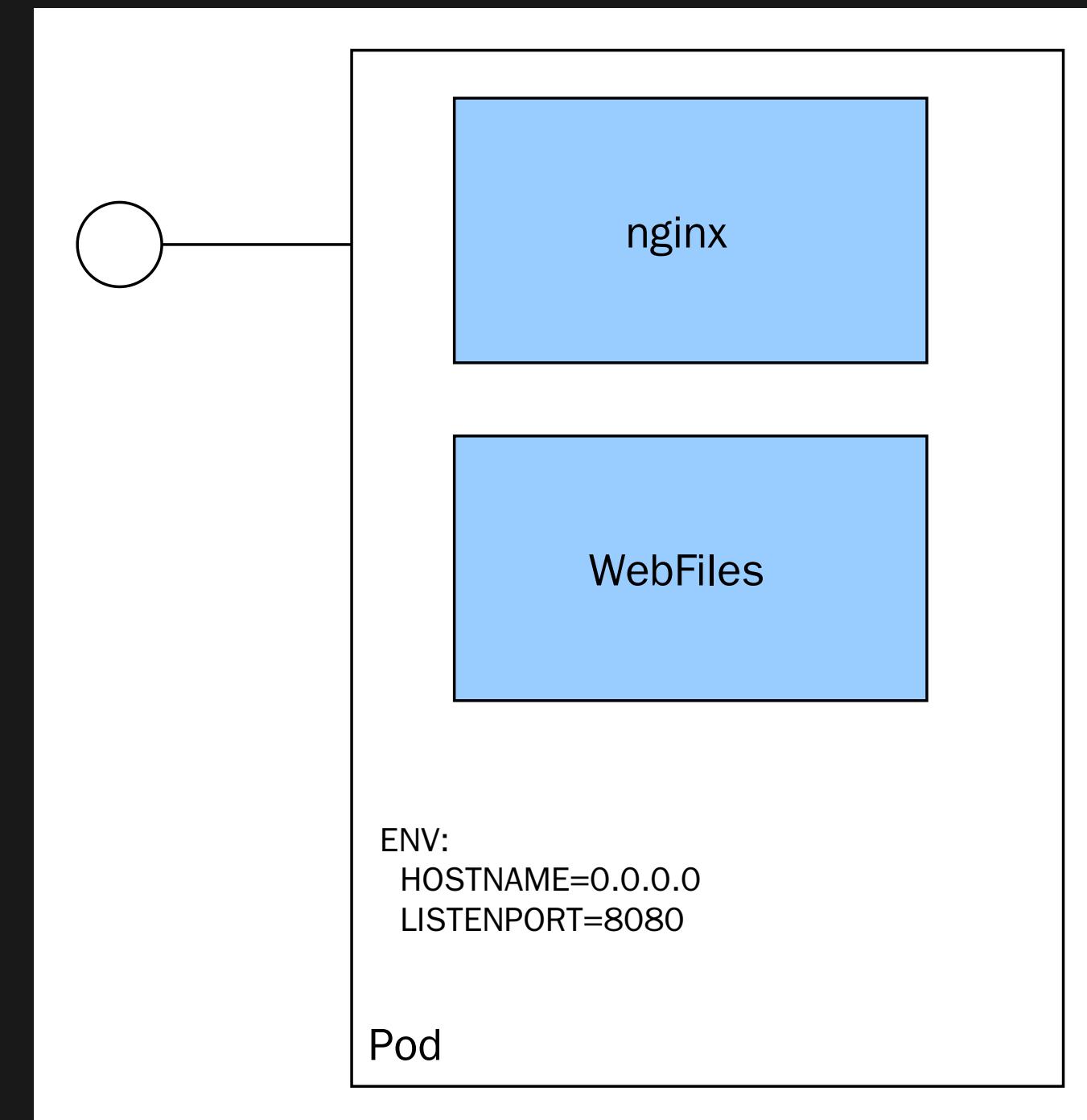
Name	Purpose	
Service	Interface	Entry point (Service Name)
Deployment	Factory	How many pods, which pods
Pod	Implementation	1+ docker running

# LOAD-BALANCING

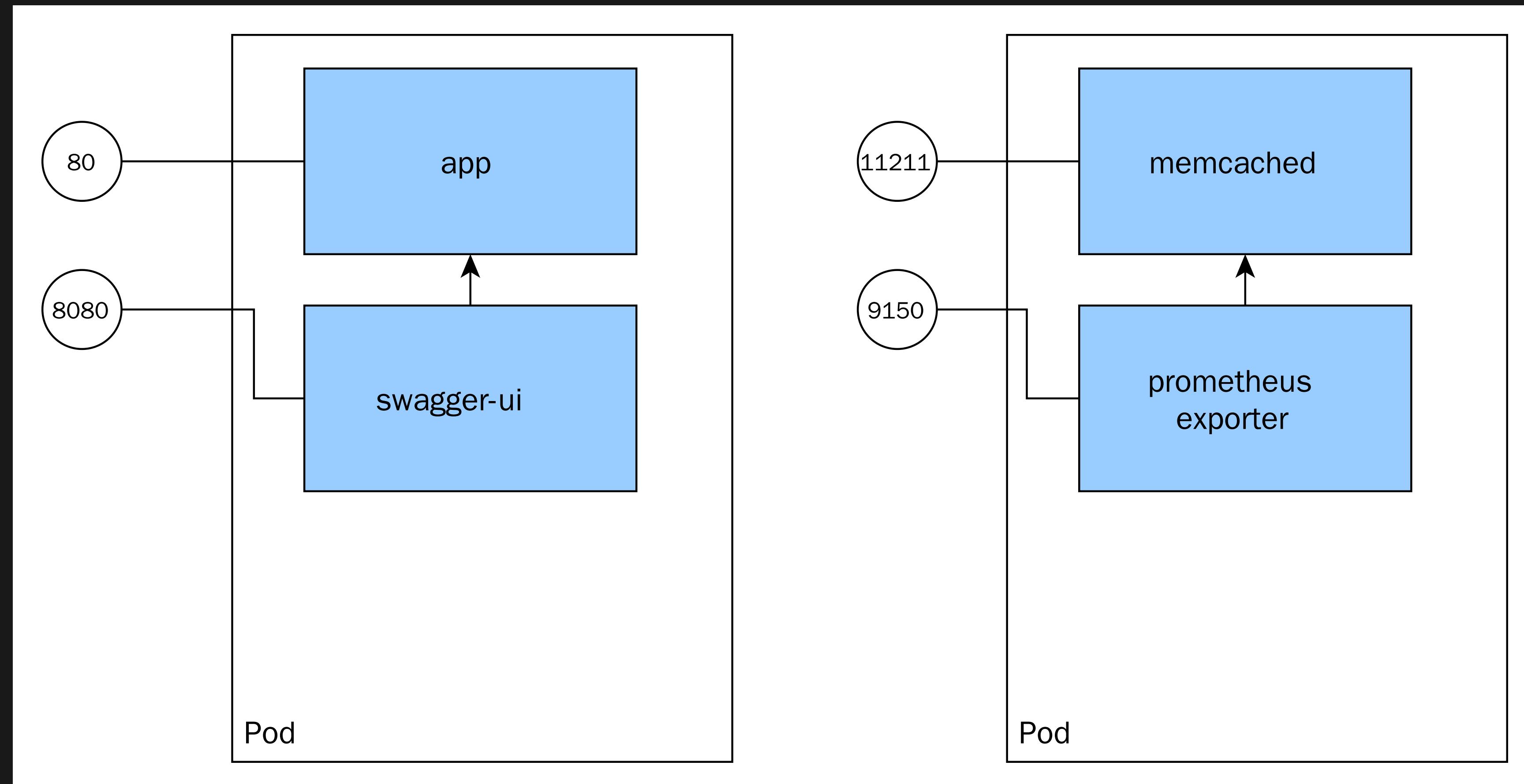


# PODS

- See each other on localhost
- Live and die together
- Can expose multiple ports



# SIDE-CARS



# SERVICE DISCOVERY AND LABELING

- names in DNS:

```
curl http://users/list
```

- labels:

```
name=value
```

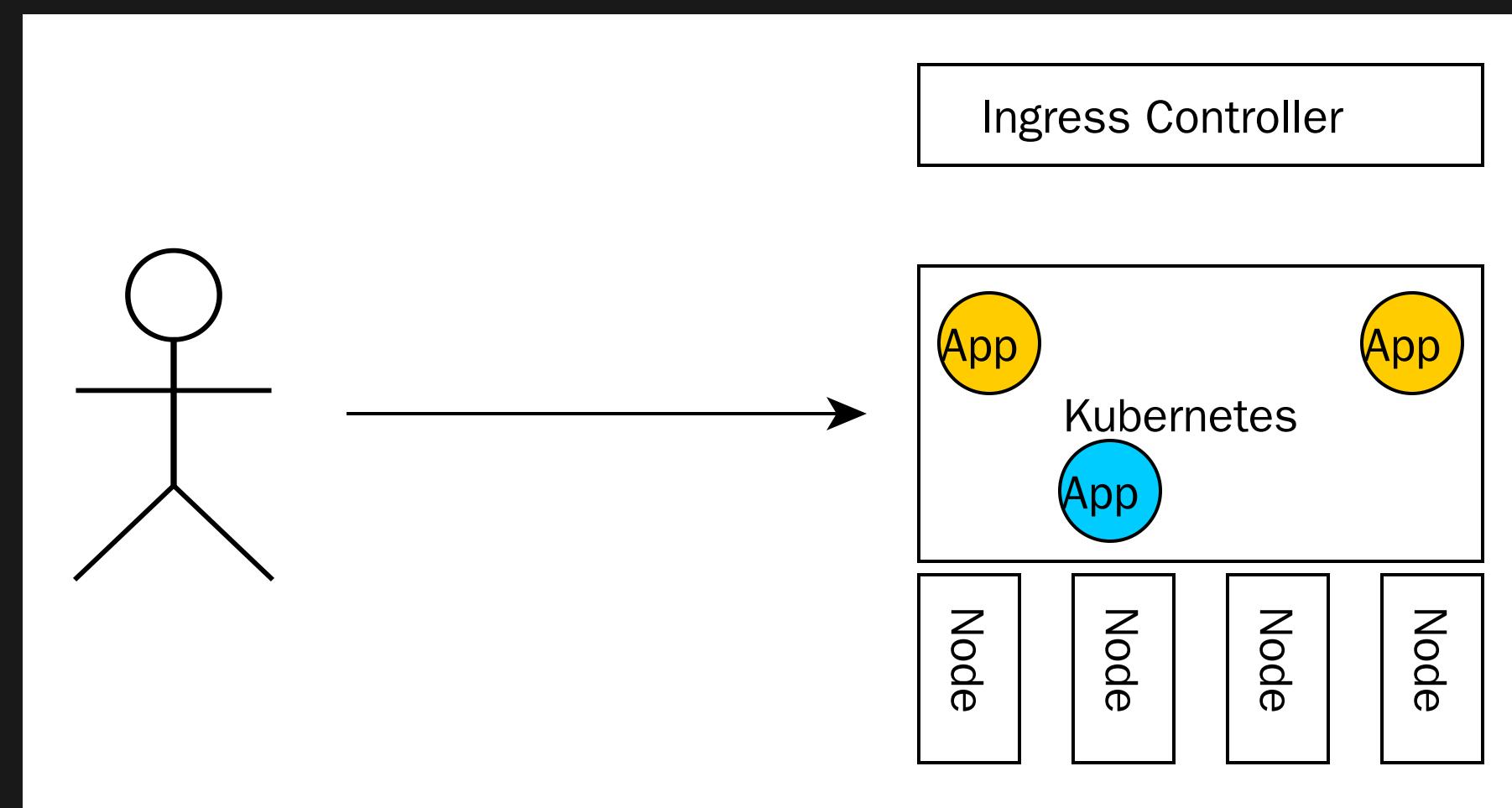
- annotations:

```
prometheus.io/scrape: "true"
```

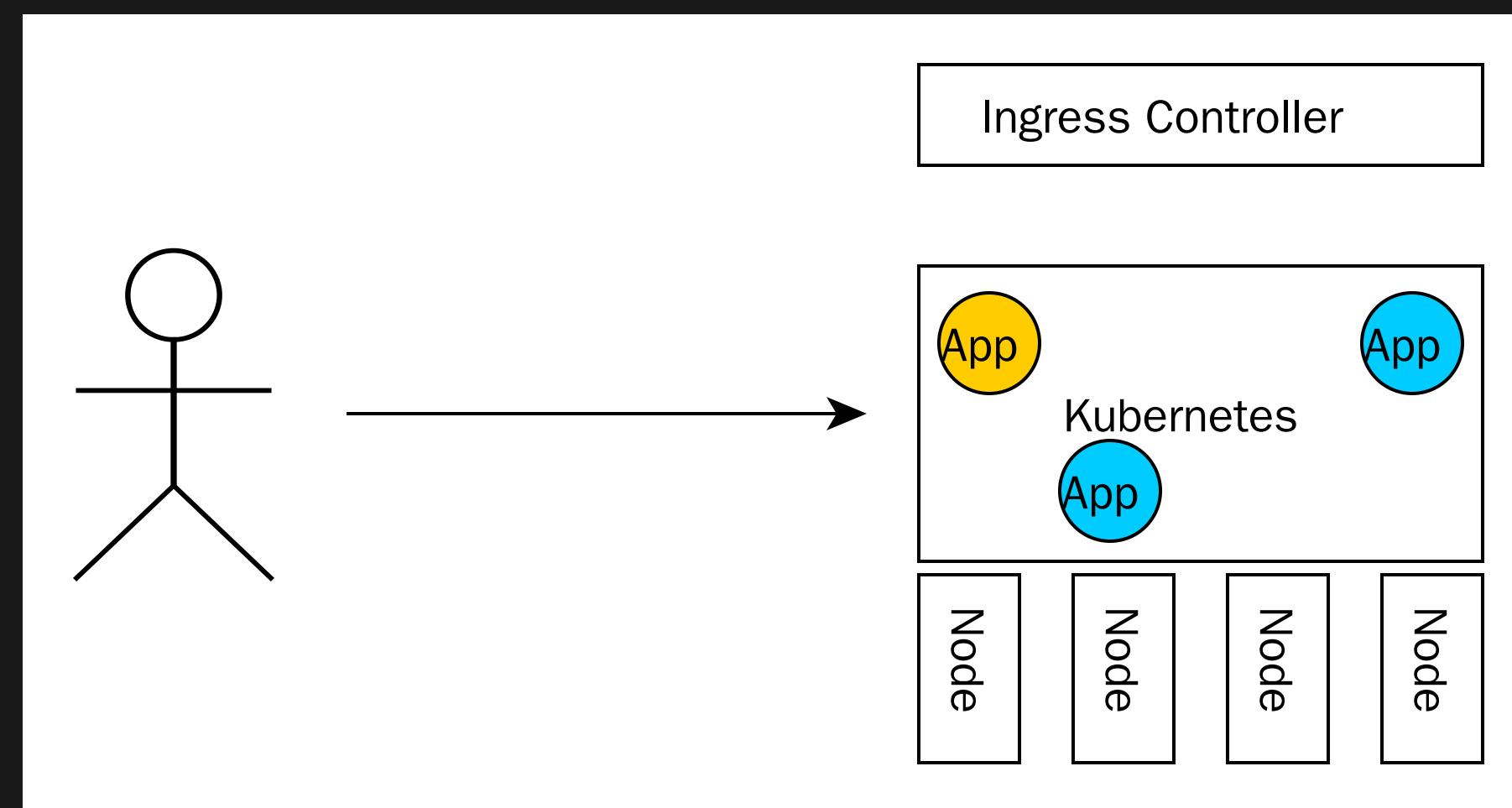
# SERVICE DISCOVERY

- loosely couple components
- auto-wiring with logging and monitoring
- drop-in installation (traefik, prometheus, ...)

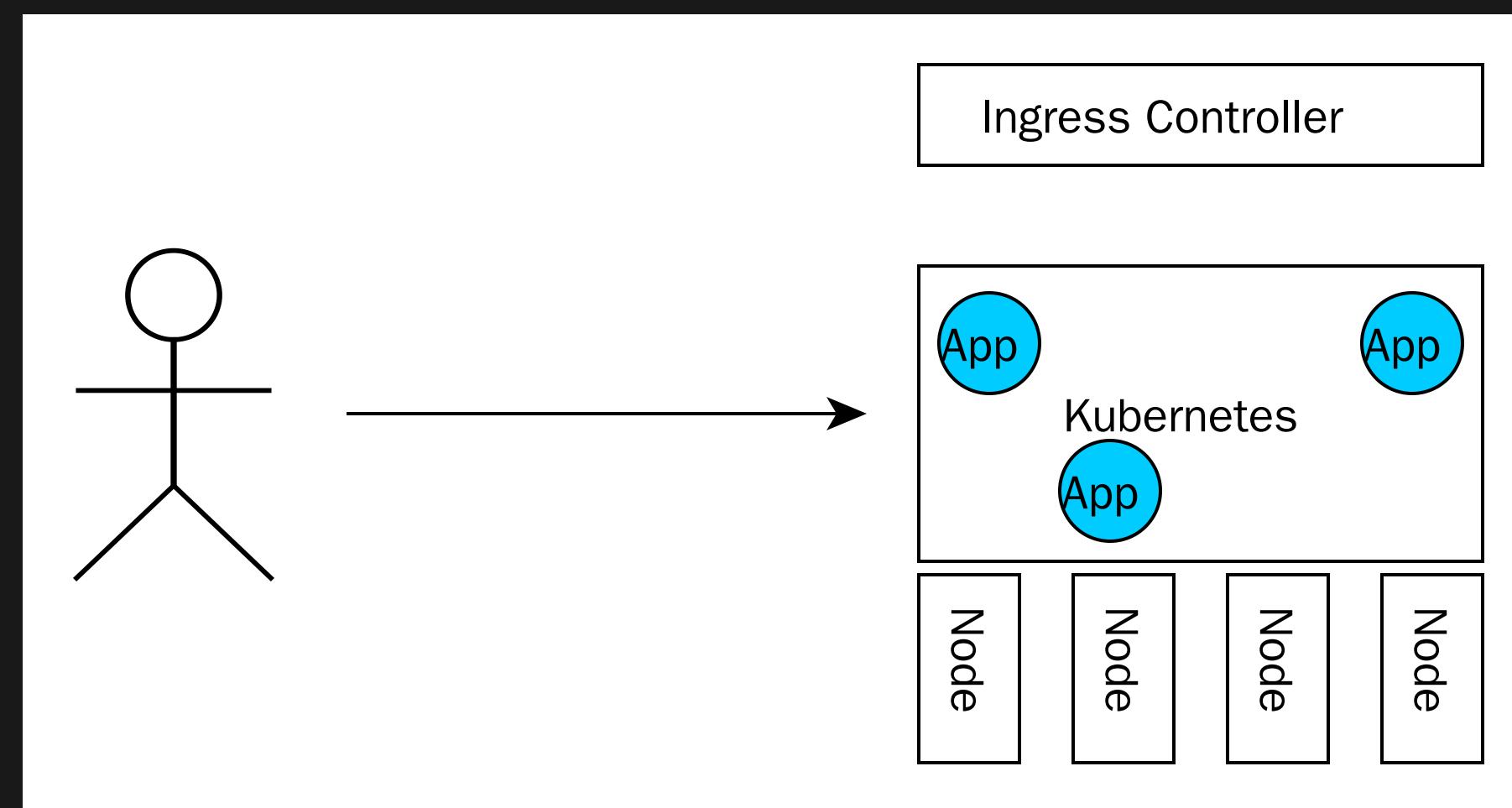
# EXAMPLE: ROLLING UPDATES



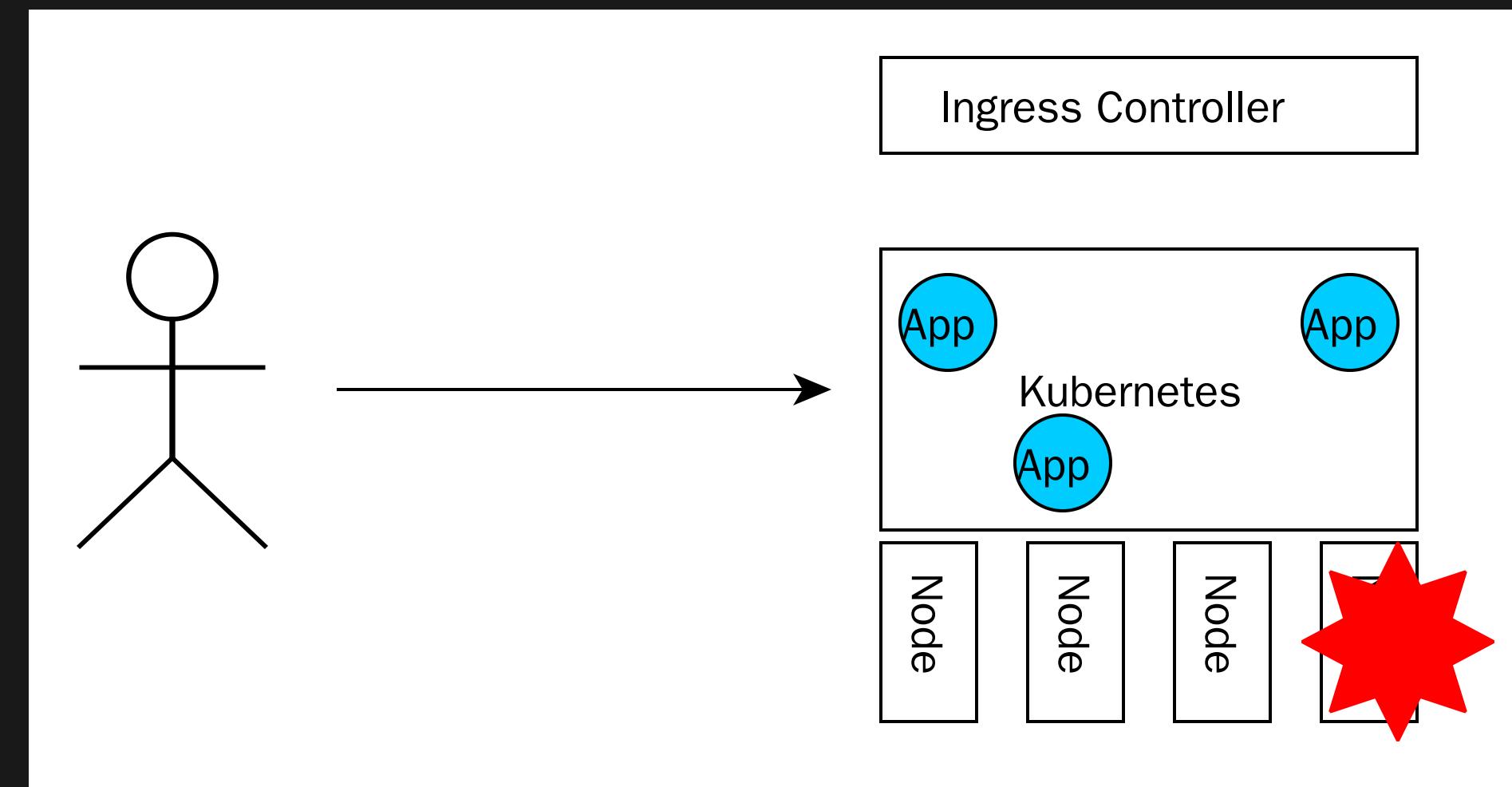
# EXAMPLE: ROLLING UPDATES



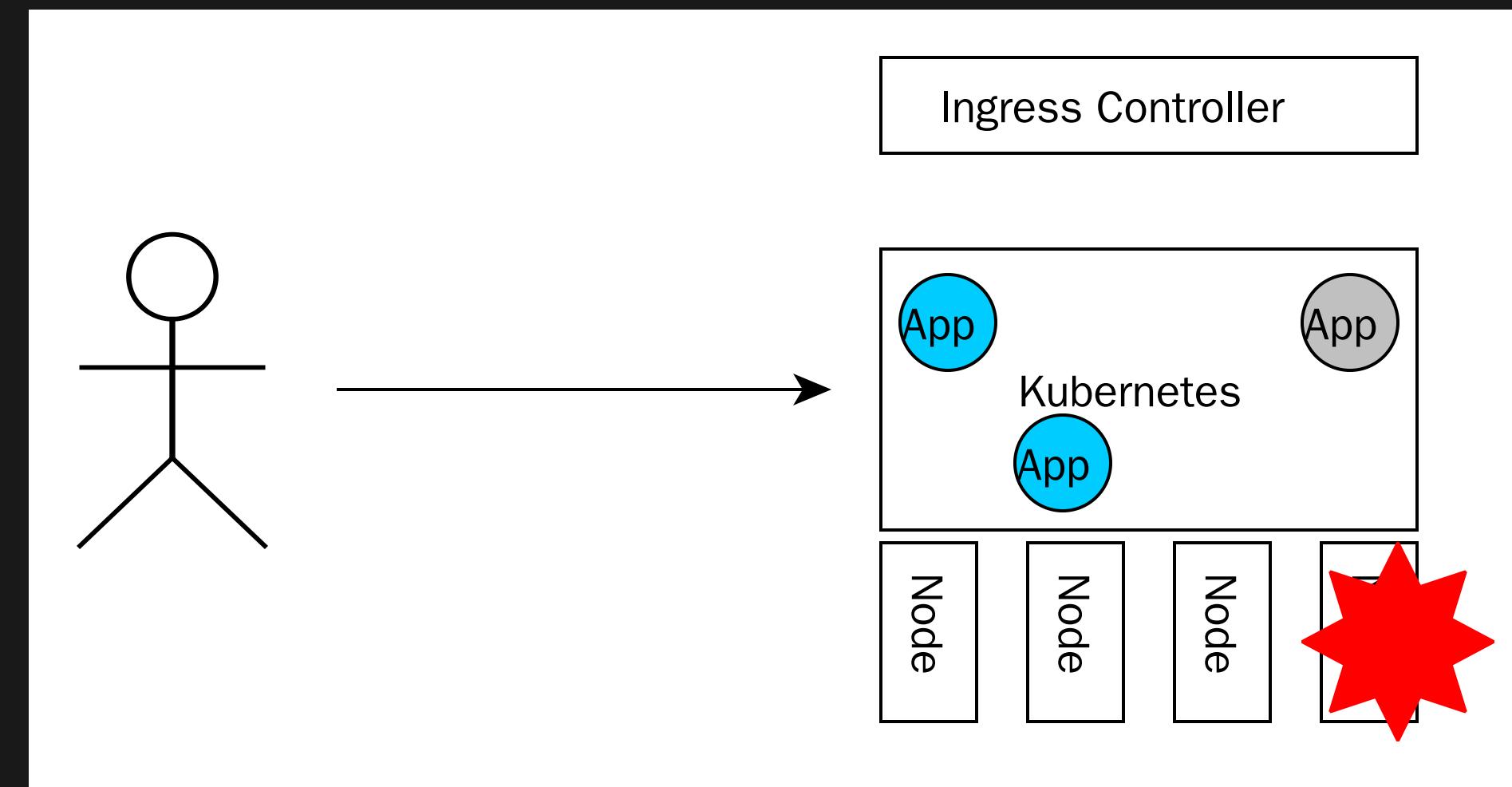
# EXAMPLE: ROLLING UPDATES



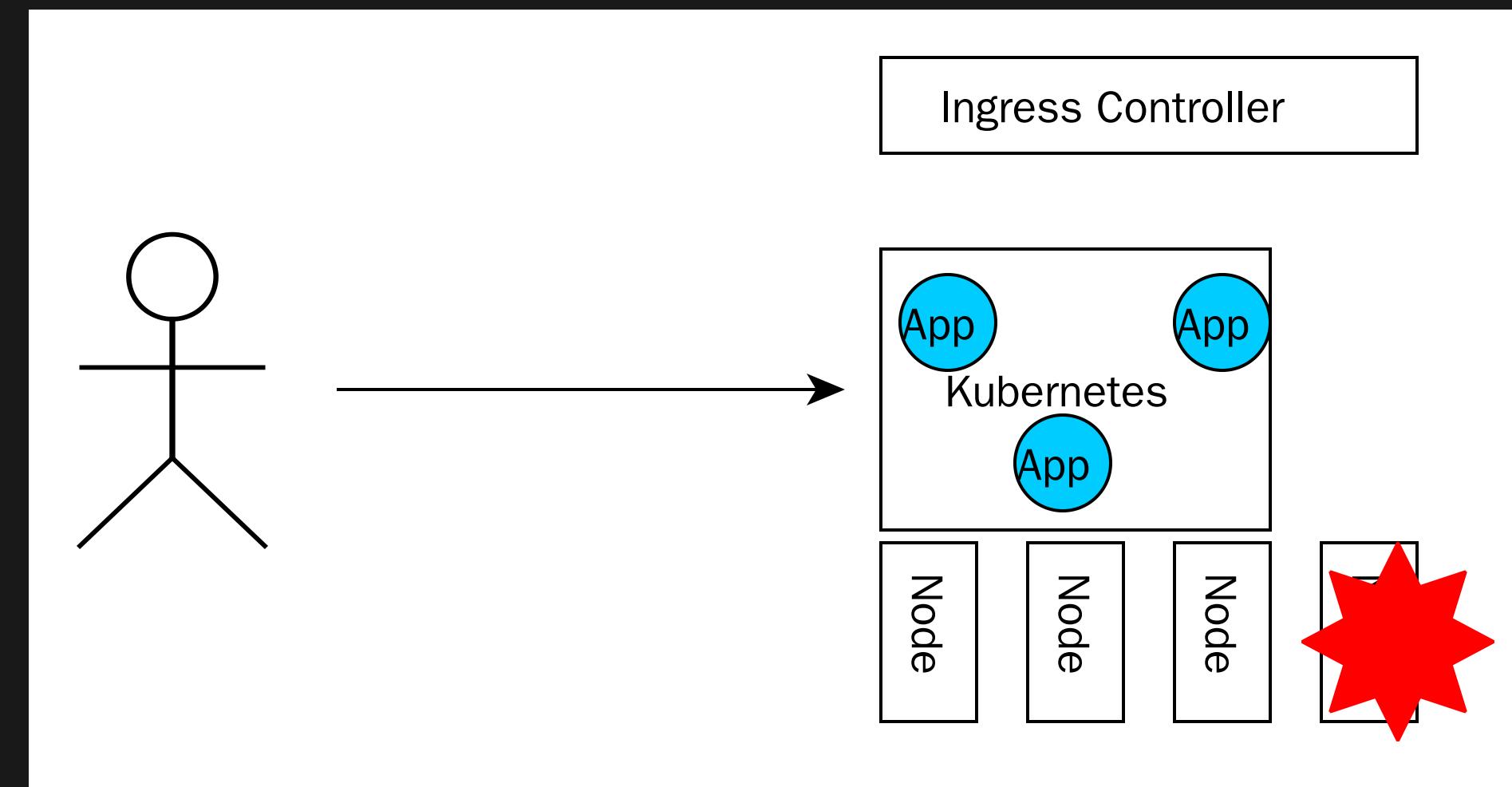
# RESILIENT INFRA



# RESILIENT INFRA



# RESILIENT INFRA



# ALL HAIL THE GIT!

- Yaml
- integration:  
monitoring, alarming, ingress-controller
- ...
- Devs can forget about infra... almost
- DevOps Culture Dream!
  - + all tools -> CRD

# Effective Kubernetes

- Start with small iterations
- Learn as-you-go
- Keep Kubernetes Understable

Hope the k8s community keep it this way

# Effective Kubernetes

- Move configuration to runtime
- Do not terrorize your devs with K8S
- No free lunch... app must be smarter

# THANK YOU. QUESTIONS?

```
123 def distance_matrix(regions):
124     """ Computes a distance matrix against a region list """
125     tuples = [r.as_tuple() for r in regions]
126     return cdist(tuples, tuples, region_distance)
127
128
129 def clusterize(words, **kwargs):
130     # TODO: write a cool docstring here
131     db = DBSCAN(metric="precomputed", **kwargs)
132     X = distance_matrix([Region.from_word(w) for w in words])
133     labels = [int(l) for l in db.fit_predict(X)]
```



[https://github.com/wojciech12/workshop\\_kubernetes\\_and\\_cloudnative](https://github.com/wojciech12/workshop_kubernetes_and_cloudnative)

```
123 def distance_matrix(regions):
124     """ Computes a distance matrix against a region list """
125     tuples = [r.as_tuple() for r in regions]
126     return cdist(tuples, tuples, region_distance)
127
128
129 def clusterize(words, **kwargs):
130     # TODO: write a cool docstring here
131     db = DBSCAN(metric="precomputed", **kwargs)
132     X = distance_matrix([Region.from_word(w) for w in words])
133     labels = [int(l) for l in db.fit_predict(X)]
```



# BACKUP SLIDES

```
123 def distance_matrix(regions):
124     """ Computes a distance matrix against a region list """
125     tuples = [r.as_tuple() for r in regions]
126     return cdist(tuples, tuples, region_distance)
127
128
129 def clusterize(words, **kwargs):
130     # TODO: write a cool docstring here
131     db = DBSCAN(metric="precomputed", **kwargs)
132     X = distance_matrix([Region.from_word(w) for w in words])
133     labels = [int(l) for l in db.fit_predict(X)]
```



# INFRA TOOLS

- Prometheus + AlertMnager + Grafana
- Traefik
- Kafka - Yolean/kubernetes-kafka
- Vault on Etcd - banzaicloud/bank-vaults

# DATA STORES

- Kafka - Yolean/kubernetes-kafka
- Etcd - coreos/etcd-operator
- DB: PSQL and Mongo

## BACKUPS:

- old-school backups with ARK and Restic
- replications across clouds

# 1. CLEAN UP

- Single script for repo - Makefile [1]
- Resurrect the README

[1] With zsh or bash auto-completion plug-in in your terminal.

## 2. GET BACK ALL THE KNOWLEDGE

Extract from:

- Puppet, ... → Dockerfile
- Running Instances → Dockerfile, README.rst
- Nagios, ... → README.rst + checks /

### 3. INTRODUCE RUN\_LOCAL

- make run\_local
- A nice section on how to run in README.rst
- with docker-compose

The most crucial point.

## 4. GET TO KUBERNETES

- make kube\_create\_config
- make kube\_apply
- Generate the yaml files if your envs differ  
secrets from gopass (password manager)

## 5. CONTINUOUS DEPLOYMENT

Travis:

- the same Makefile as dev use

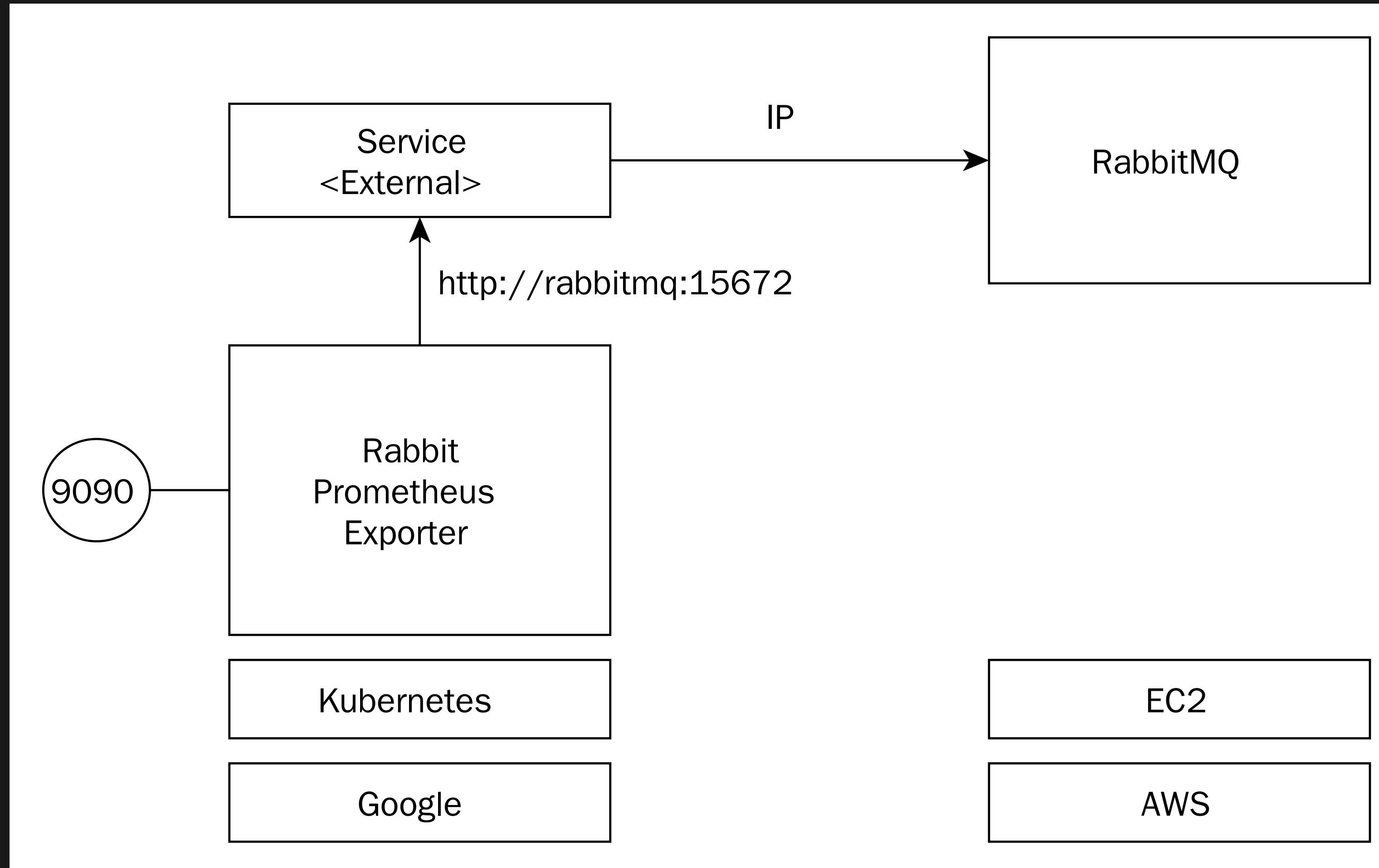
## 6. KEEP IT RUNNING

Bridge the new with old:

- Use External Services in Kubernetes
- Expose k8s in the Legacy [1]

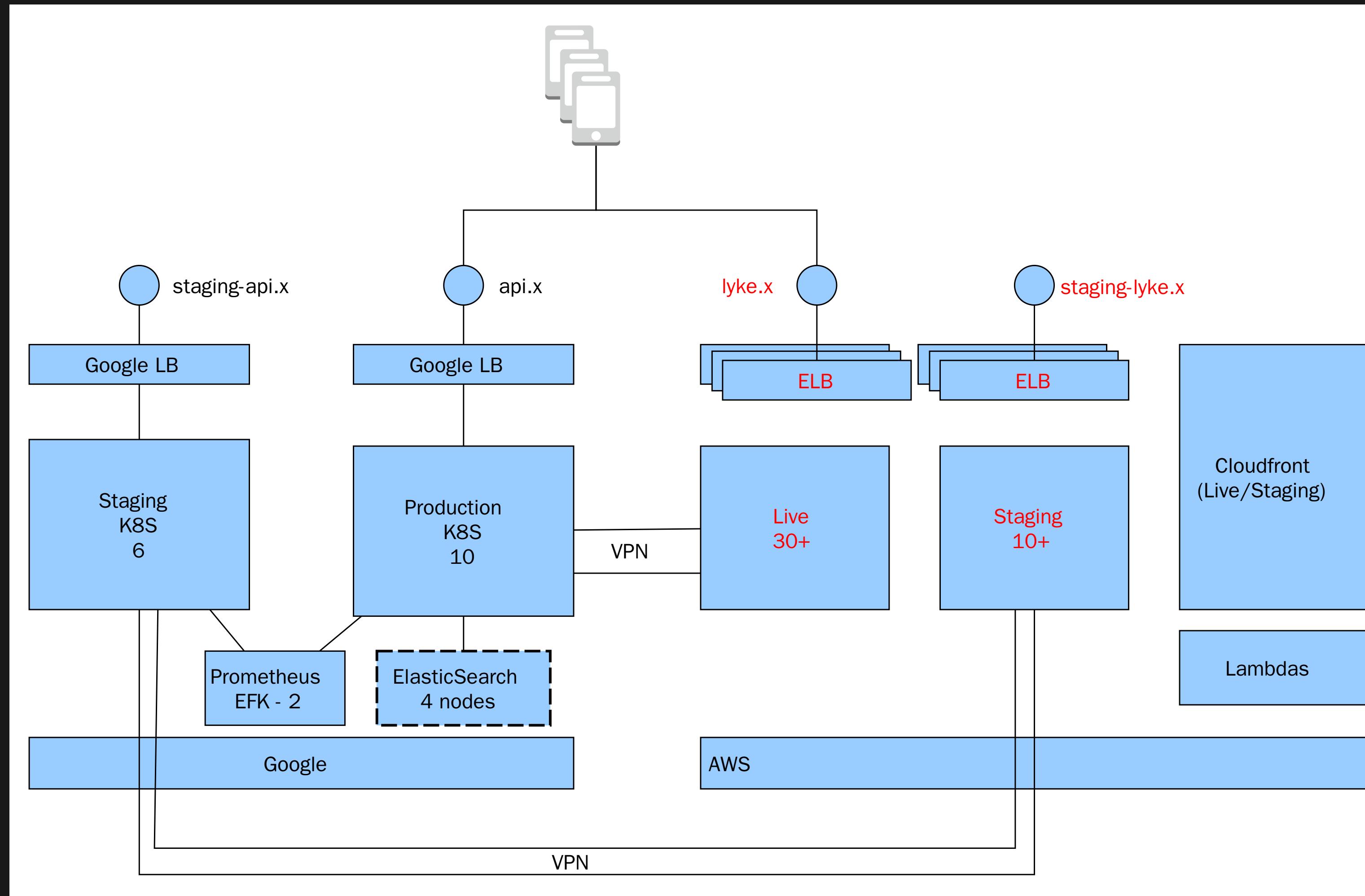
[1] hard coded IP:PORT, considered: K8S events to Consul

# Bridge the new with old



Monitor legacy with new stack

# Architecture During Migration



## 7. INTRODUCE SMOKE-TEST

```
TARGET_URL=127.0.0 make smoke_test  
TARGET_URL=api.example.com/users make smoke_test
```

## 8. SERVICE SELF-AWARE

Add to old services:

1. metrics/
2. health/
3. info/

## 9. MOVE TO MICRO-SERVICES

Offload Legacy:

- Keep the lights on
- New functionality to micro-services

## 10. GET PERFORMANCE TESTING

- wrk for evaluating performance
- load test the real system