Kubernetes i CloudNative introduction

Wojciech Barczyński - SMACC.io | Hypatos.ai

27 March 2019

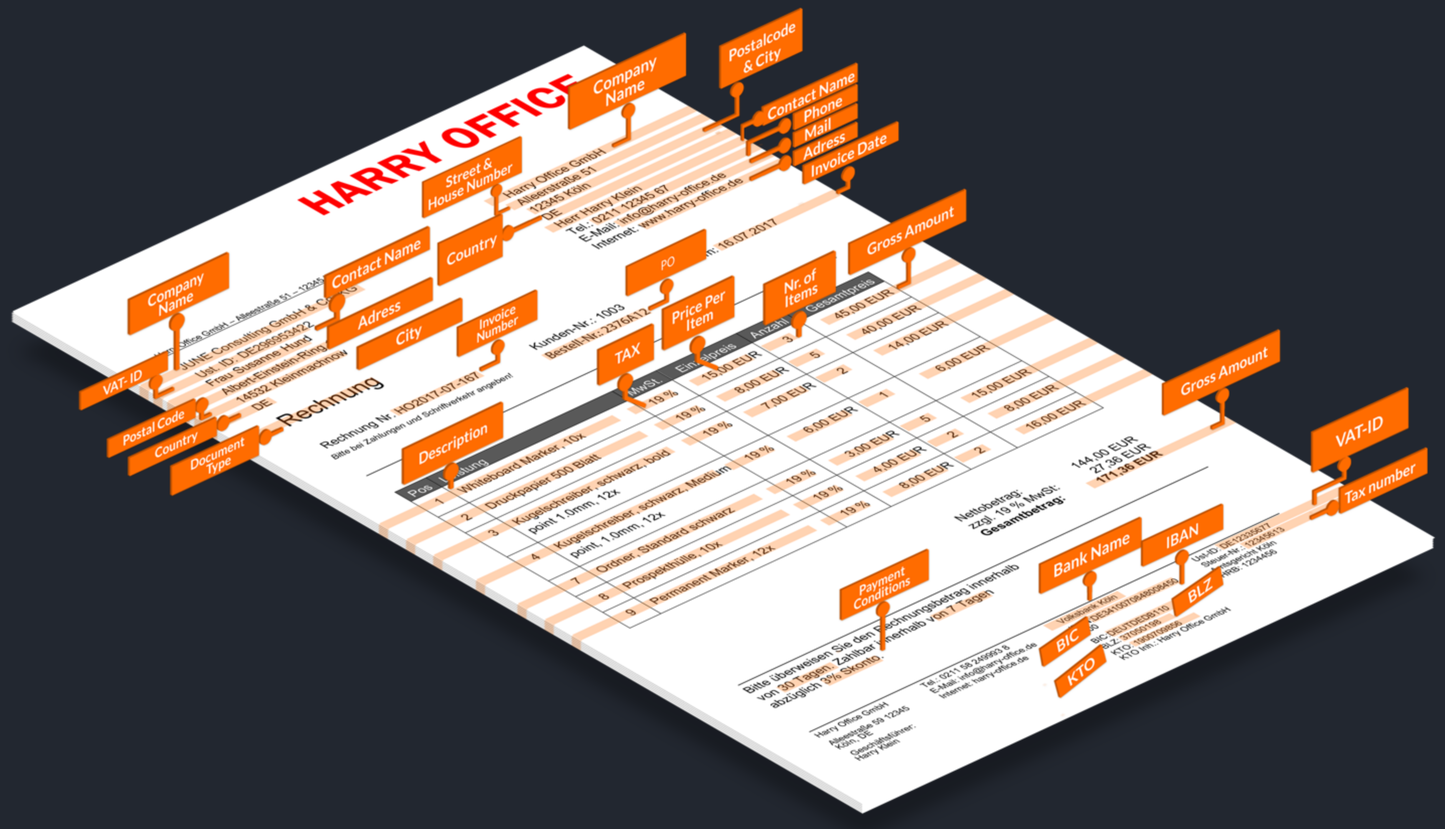
WOJCIECH BARCZYŃSKI

- Software Developer & System Engineer
- Now at Machine-Learning FinTech
- Since 2016 working with K8S

Github: wojciech12 | Linked: IN | HP: wbarczynski.pl | T: @wbarczynski

Hypatos

Problem SMACC solves



SAP

Microsoft Dynamics

DATEV

SMACC

Beginnings

... - 2014

- Everybody talks *services* but....
- No simple platform for operating services
- Mesosphere!
- Docker swarm not perfect
- We learn so much in Google (Borg) and other companies

Docker like RoR change

New Linux, New Application Server!

OnPremise

CLOUD NATIVE COMPUTING FOUNDATION

kubernetes

Prometheus

OPENTRACING

fluentd

Istio

etcd

https://github.com/cncf/artwork

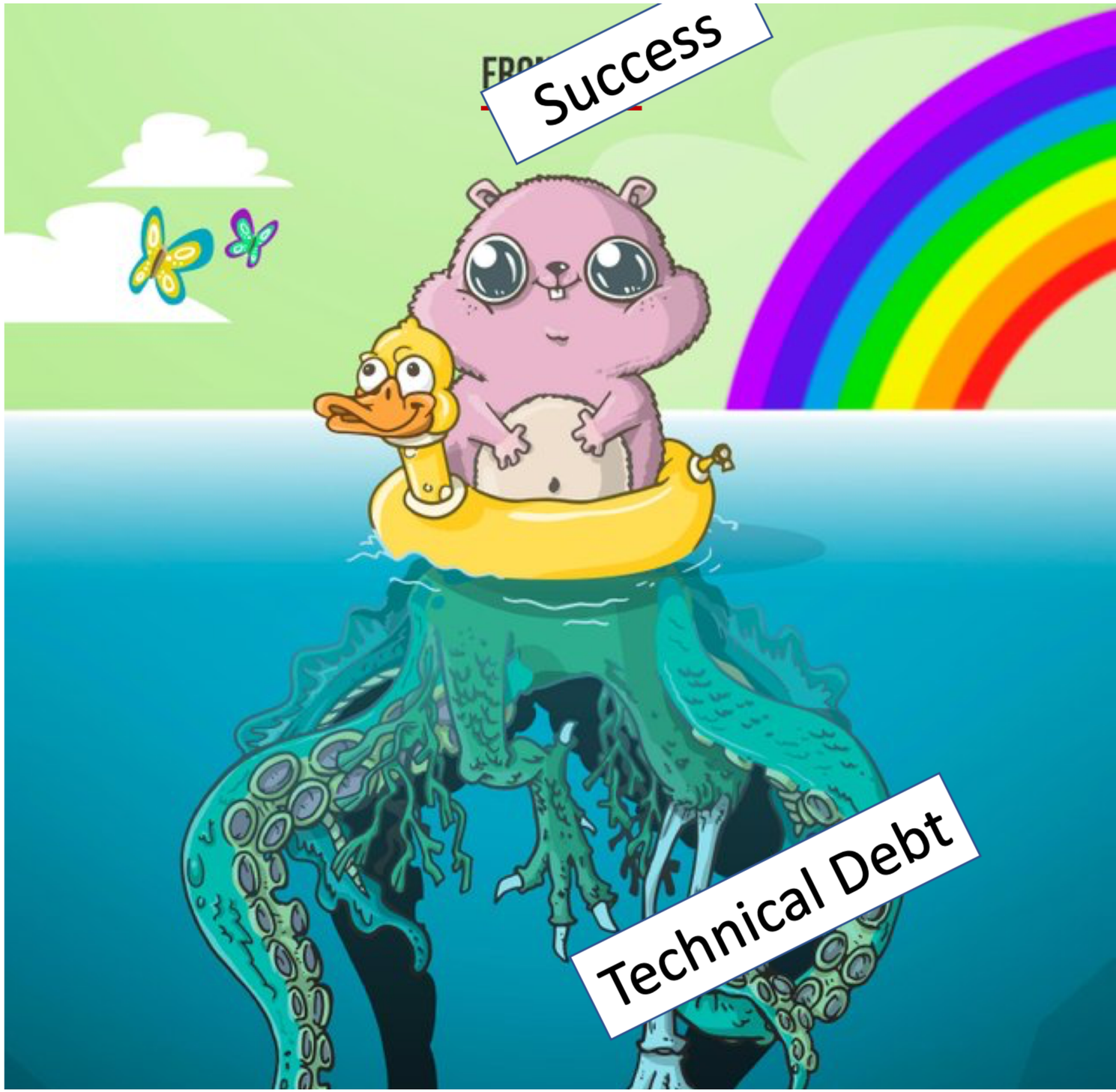
Slow delivery

Continuous Deployment?

Fear

Frustration

XX% Idle Machines



2016

SALTSTACK

ANSIBLE

CHEF

puppet

AWS
CloudFormation

HashiCorp
Terraform

Black (Blue) Box

Infrastructure (almost) invisible

Easy* Continuous Deployment

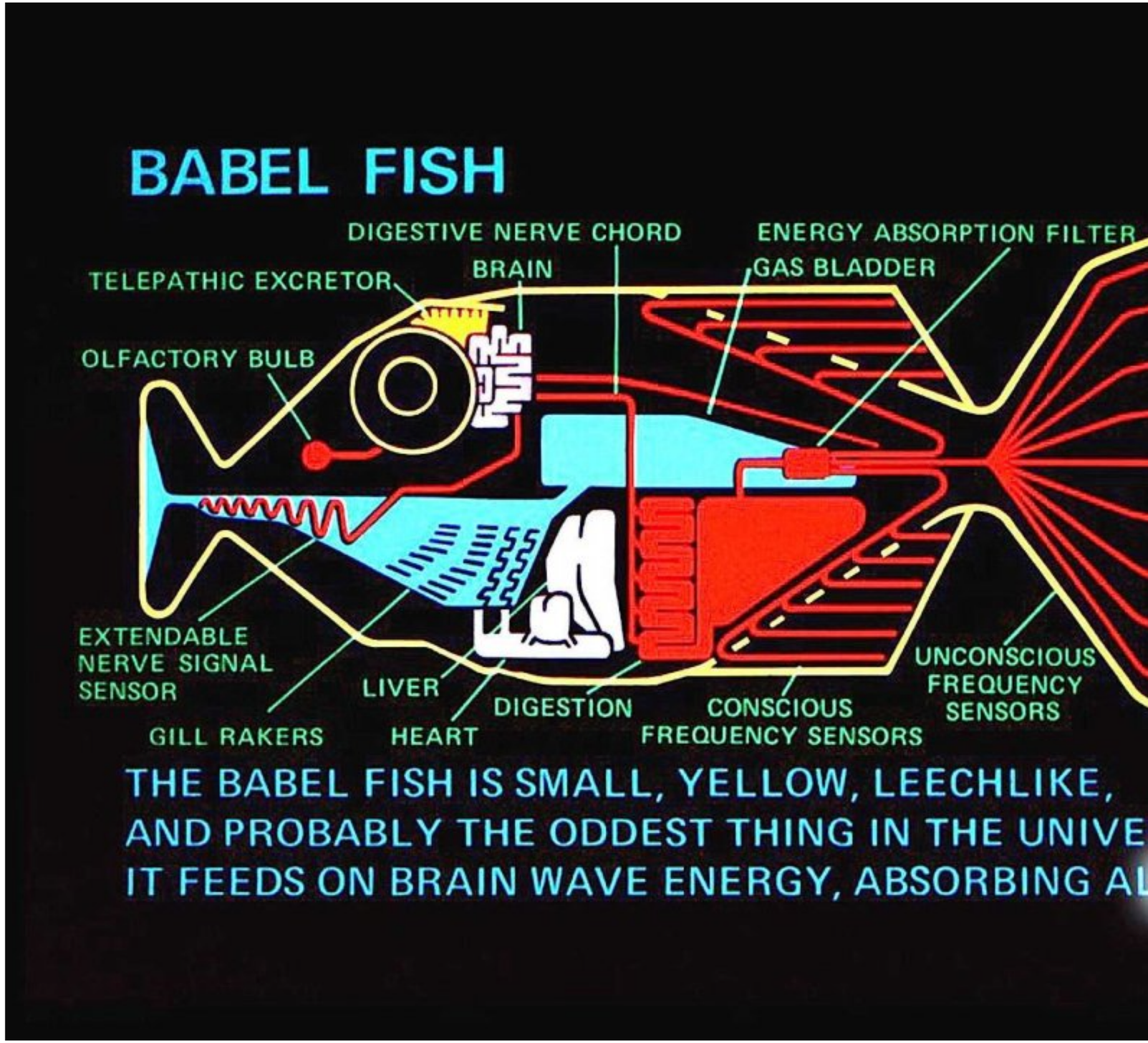
https://en.wikipedia.org/wiki/File:Dr_Who_(316350537).jpg

Common

Language

Artifacts

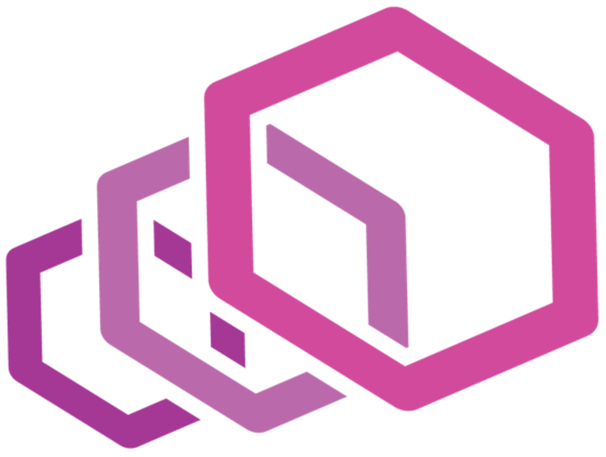
Platform



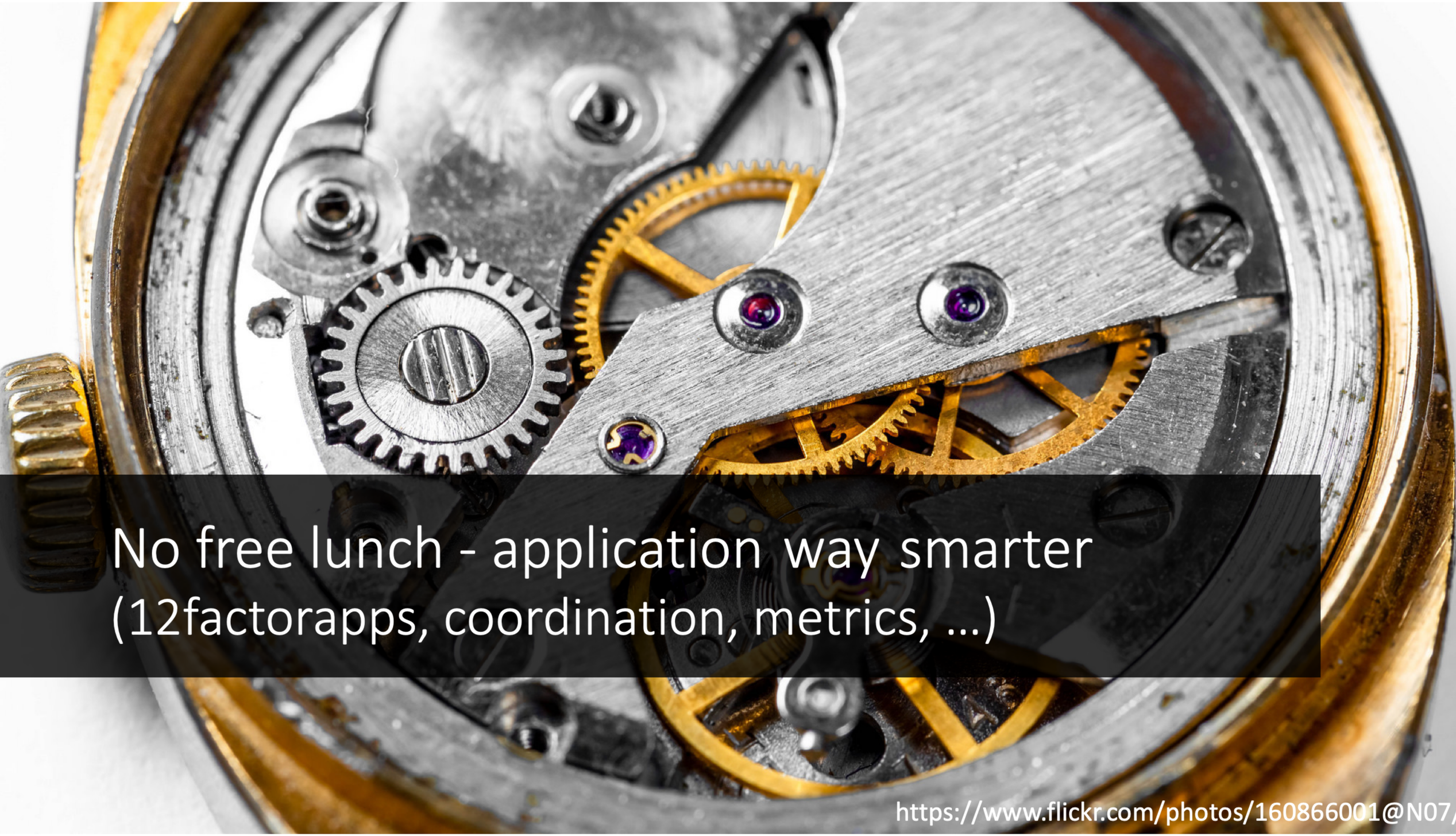
Learn-as-you-go

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

traefik



envoy



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

https://www.flickr.com/photos/160866001@N07/

KUBERNETES

- Service oriented
- Battery for 12factor apps
- Orchiestration
- Focus on simplicity [*]

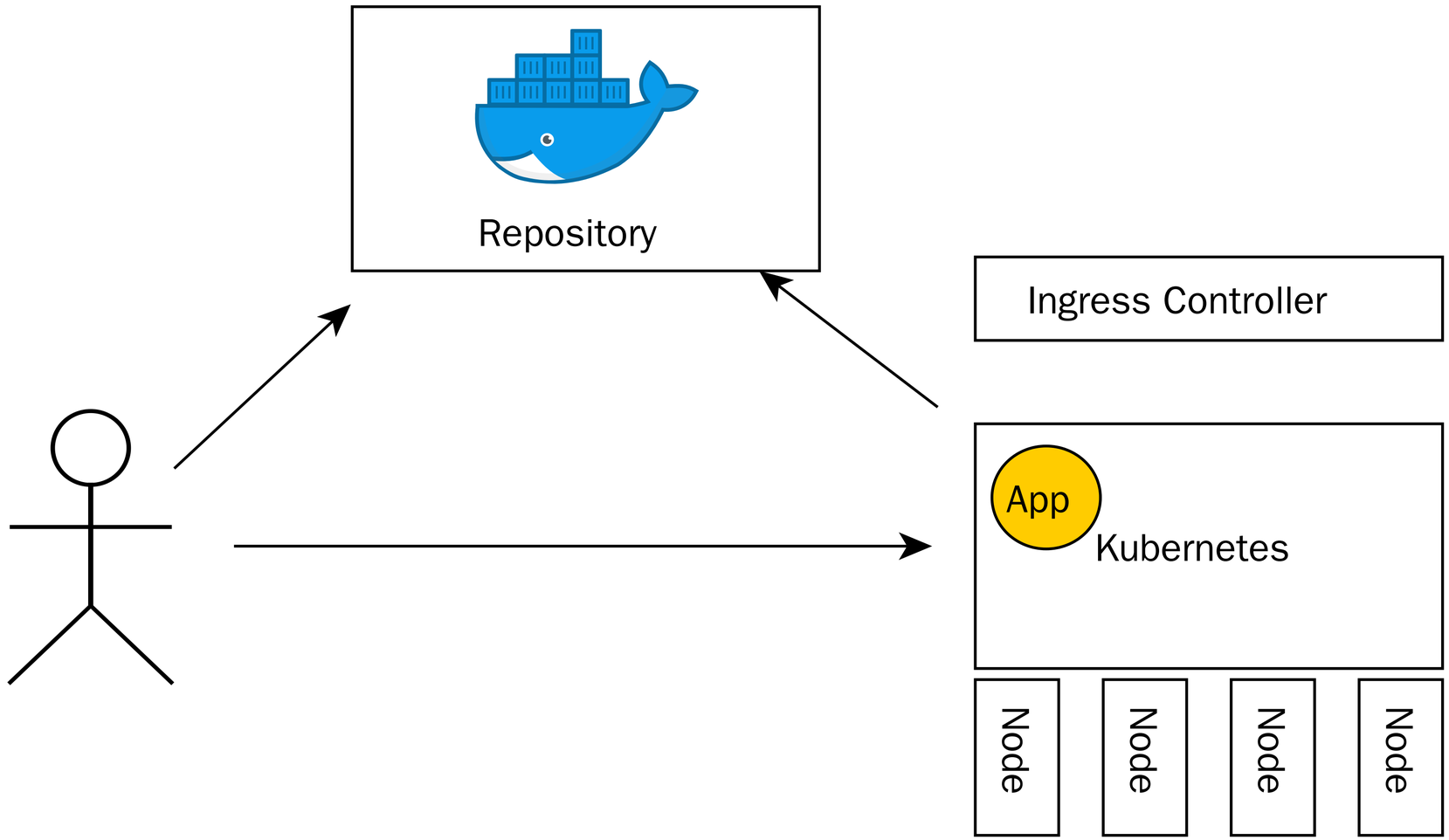
[*] In spirit what Docker brought us

The twelve factor apps

- Services to operate service, test, etc.
- Services with low TCO.
- Published by Heroku
- ps. No free lunch

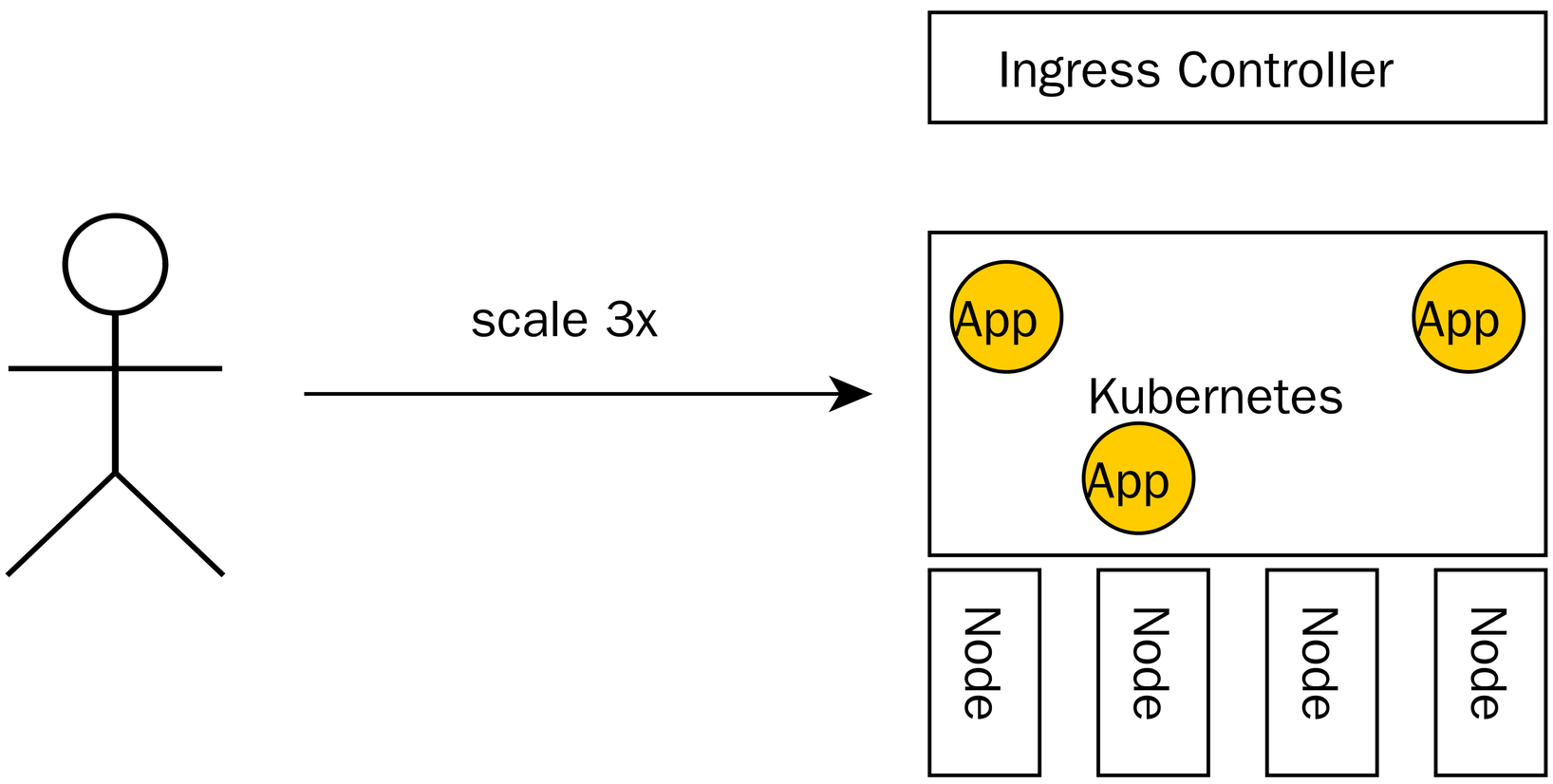
https://12factor.net

KUBERNETES



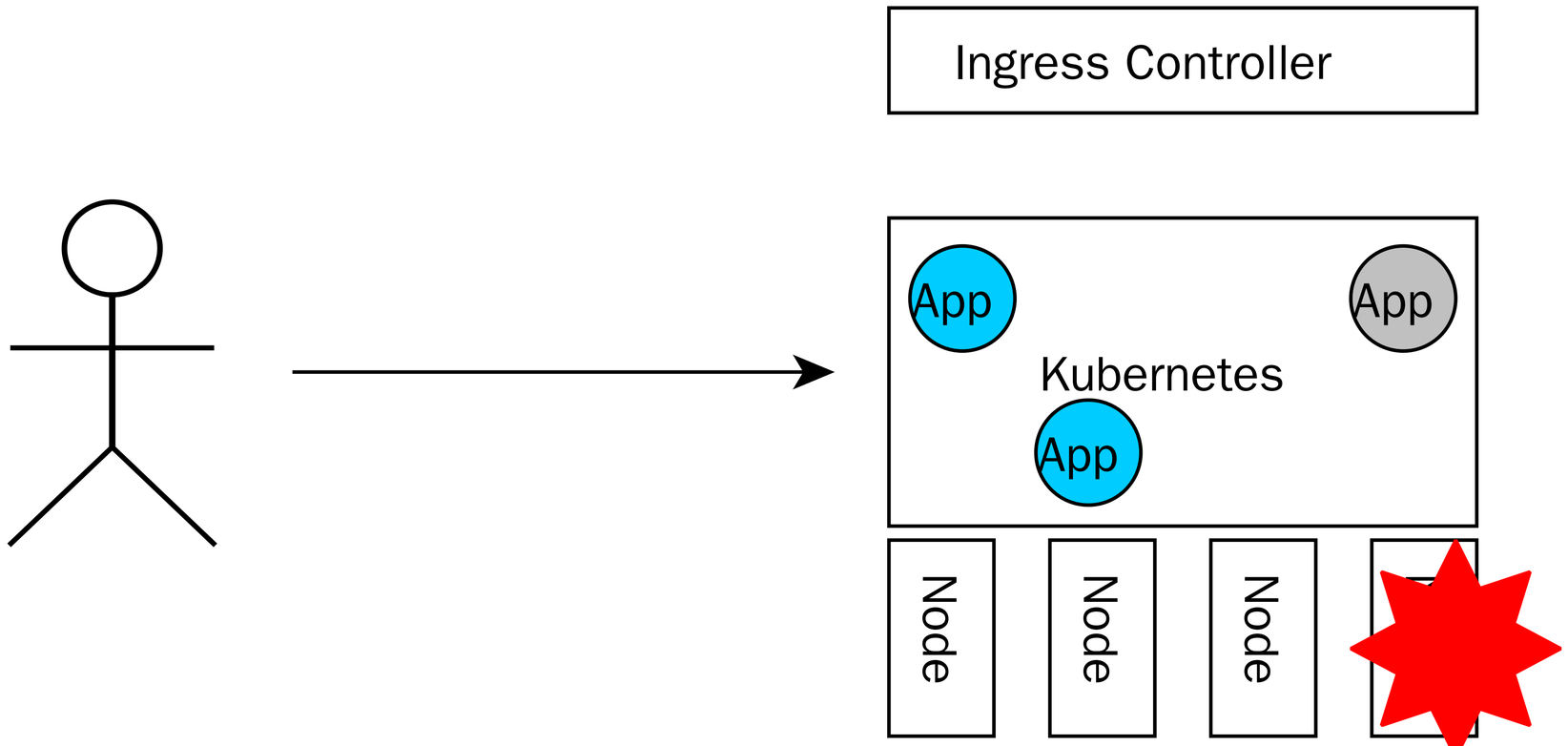

```
make docker_push; kubectl create -f app-srv-dpl.yaml
```

SCALE UP! SCALE DOWN!



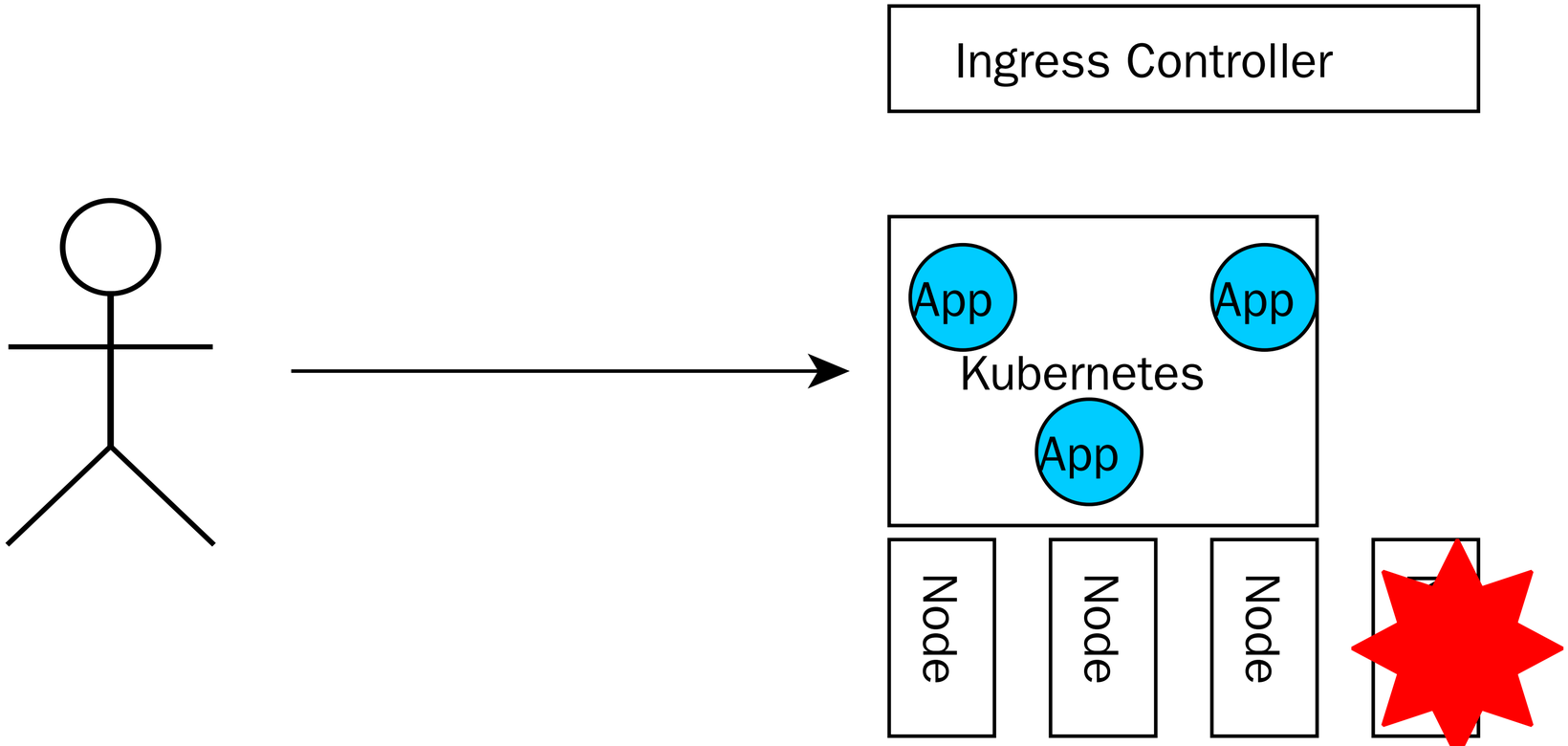

```
kubectl --replicas=3 -f app-srv-dpl.yaml
```

RESILIENCE!



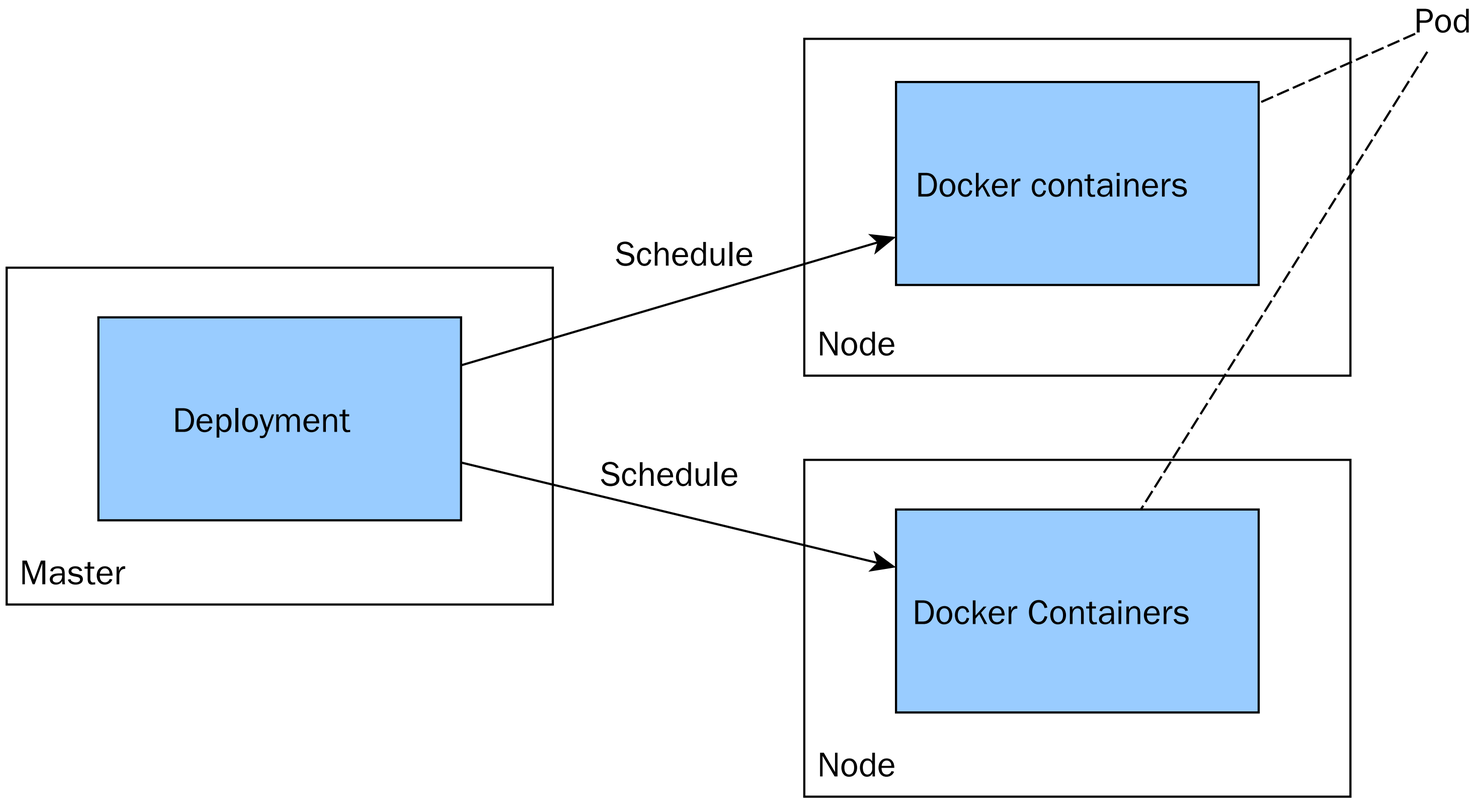

```
kubectl --replicas=3 -f app-srv-dpl.yaml
```

RESILIENCE!




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

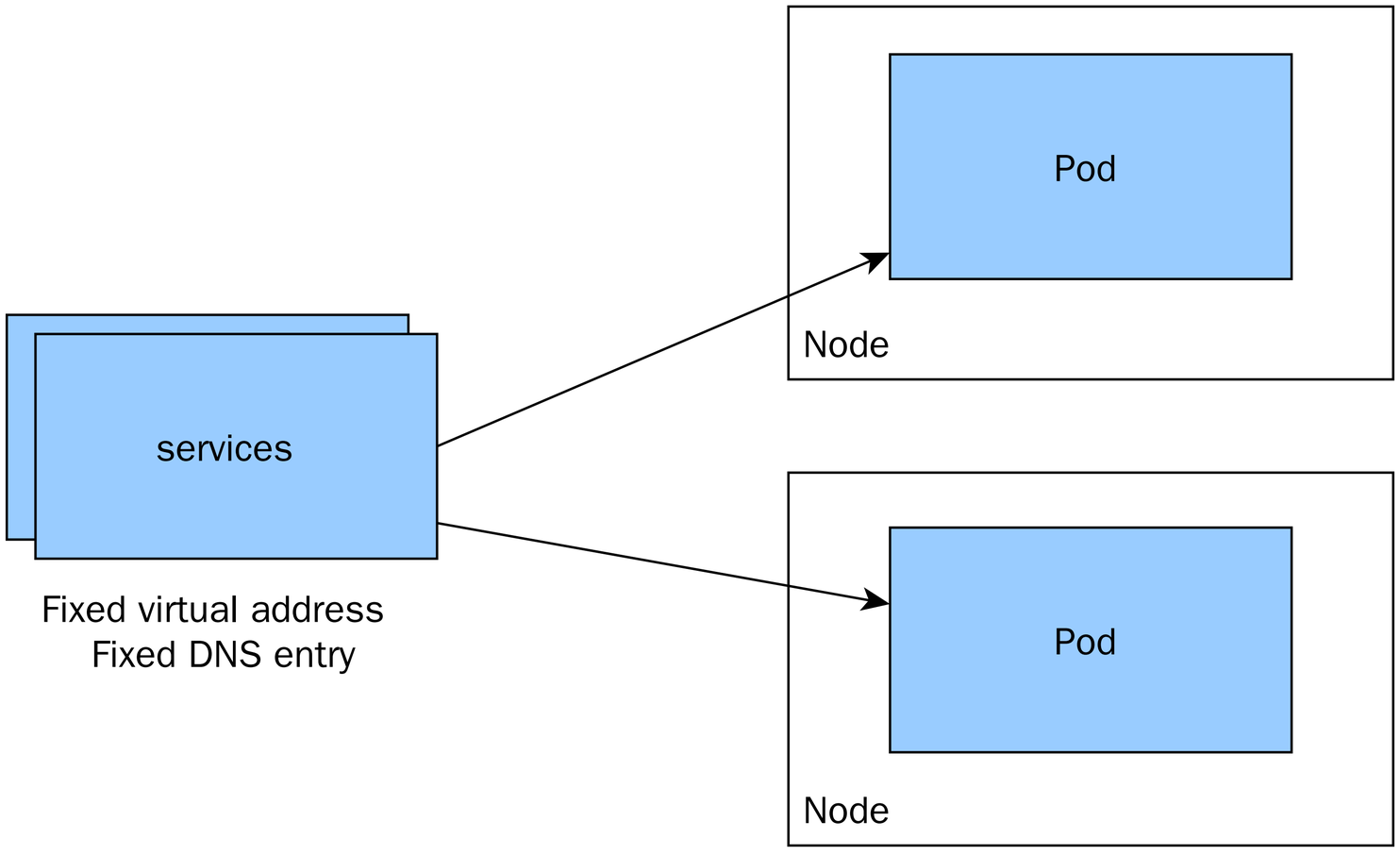
DEPLOYMENT AND PODS



deployment.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-api
  labels:
    app: demo-api
spec:
  replicas: 3
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: demo-api
  template:
```

SERVICE AND PODS



Service matches pods based on labels

service.yml

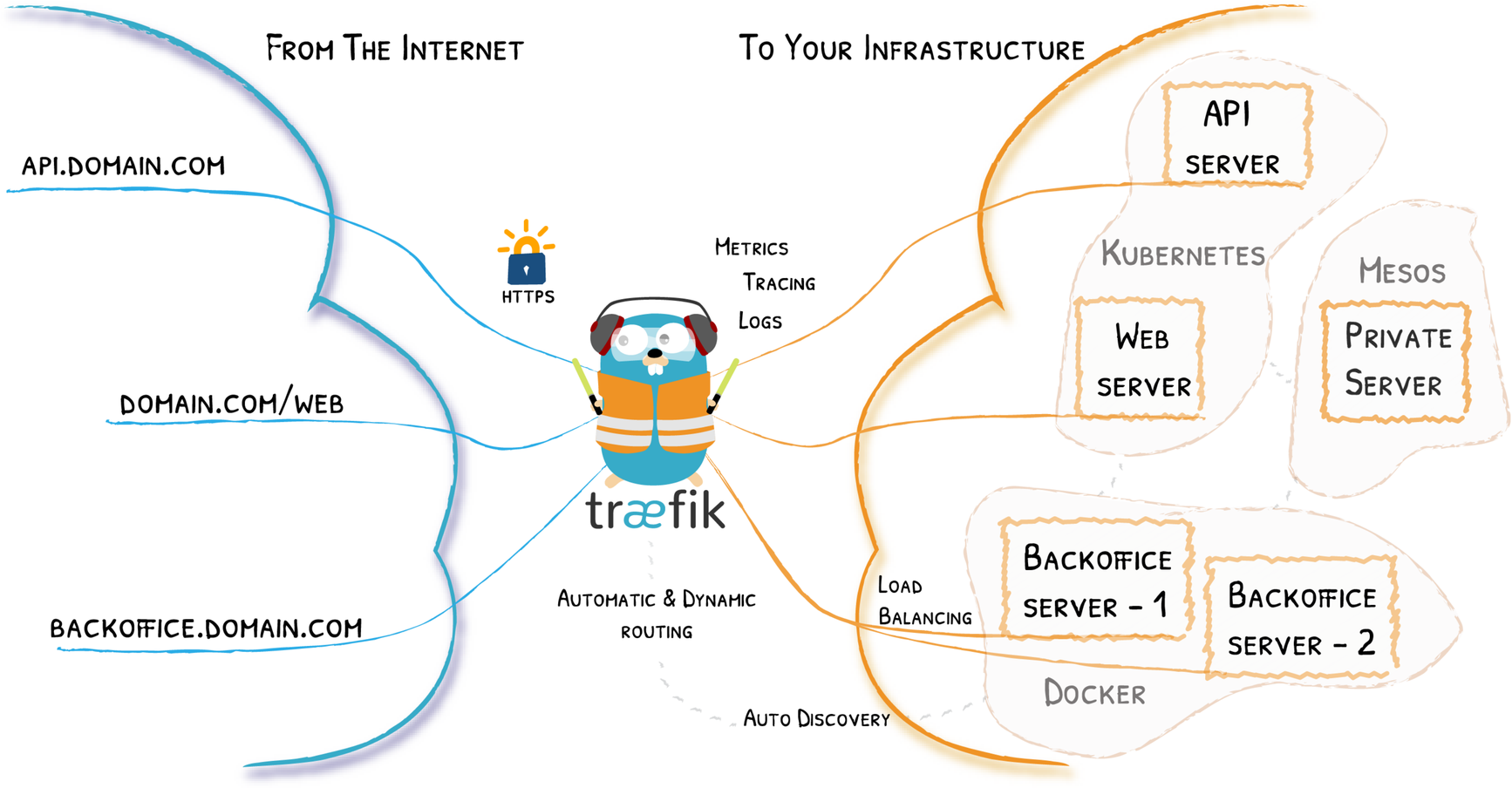
```yaml
apiVersion: v1
kind: Service
metadata:
  name: demo-api
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: demo-api
  type: LoadBalancer
```

SERVICE

`curl http://users/list`

Load balanced inside cluster!

HOW GET USER REQUESTS?



INGRESS

Pattern	Target App Service
api.smacc.io/v1/users	users-v1
api.smacc.io/v2/users	users-v2
smacc.io	web

BASIC CONCEPTS

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

SERVICE DISCOVERY AND LABELING

- names in DNS:
  `curl http://users/list`
- labels:
  `name=value`
- annotations:
  `prometheus.io/scrape: "true"`

SERVICE DISCOVERY

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

INTEGRATION WITH MONITORING

```yaml
apiVersion: v1
kind: Service
metadata:
  name: iam
  labels:
    app: iam
    tier: backend
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/scheme: "http"
    prometheus.io/path: "metrics"
    prometheus.io/port: "8088"
```

INGRESS CONFIGURATION

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: iam
  annotations:
    kubernetes.io/ingress.class: traefik
    traefik.frontend.rule.type: PathPrefixStrip
spec:
  rules:
  - host: example.org
    http:
      paths:
        - path: /iam
          backend:
```

All hail the git!

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

+ all tools -> CRD

simple semantic

- Complex release strategies are (relatively) easy
- Continoues Deployment is simply to implement!

github.com/wojciech12/talk_zero_downtime_deployment_with_kubernetes

CLOUD NATIVE
COMPUTING FOUNDATION

PROMETHEUS

Prometheus

Best in class monitoring and alerting

FLUENTD

fluentd

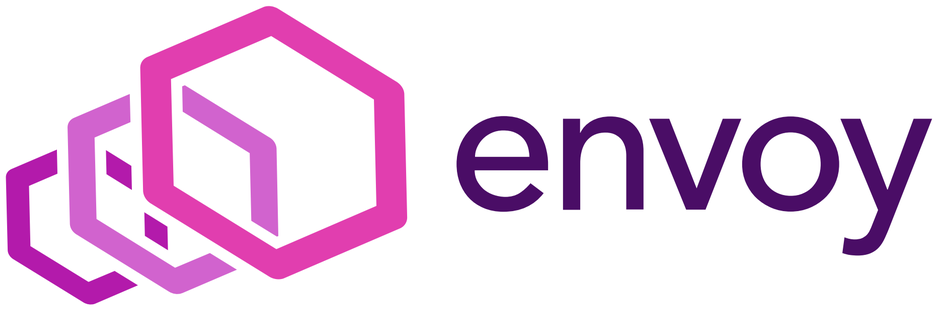
combined with ElasticSearch and Kibana

ETCD

Distributed key-value store.

Alternative: Consule, Zookeeper

INGRESS/PROXY



other: Nginx, Traefik

HOTTEST!

Not in CloudNative Foundation

TRACING

JAEGER

the last piece for observability, read this blog

K8S PACKAGE MANAGER

I do not recommend to use

MUCH MORE

- Kubeless/OpenFaaS for Serverless
- Kubeflow/Pachyderm for Machine Learning
- VirtualMachines with K8S - kata, kubevirt

COMMON

- Simple (reasonable) to run
- Focus on simplicity in operating at scale

HOW TO START

AS-A-SERVICE

- Google - GCE
- Azure - AKS
- Amazon - EKS

A lot of installers as well. See also rancher.

AT HOME

- Mini-kube
- ...

A lot of installers as well

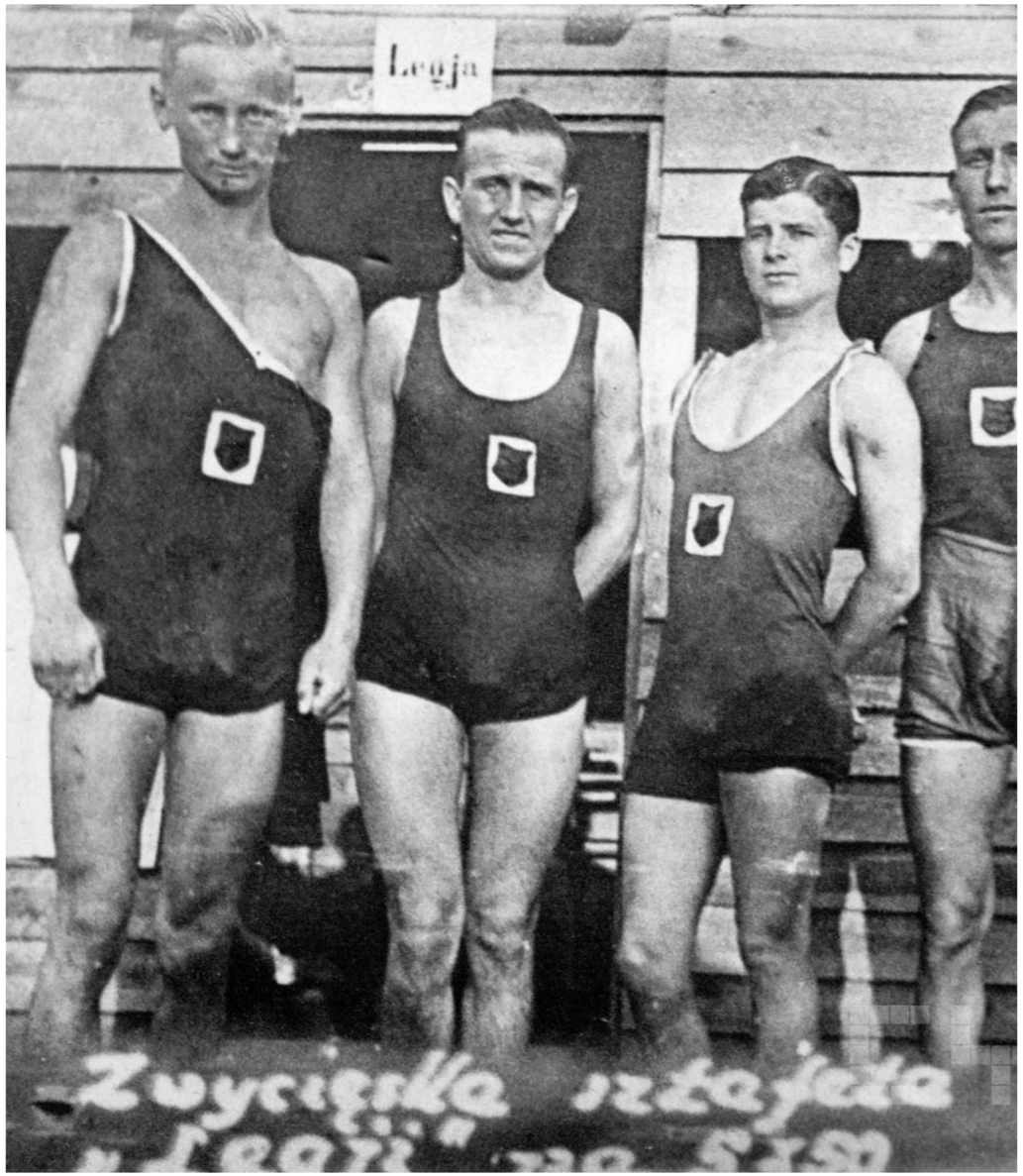
Keep it simple

https://www.flickr.com/photos/bruno_brujah/

Keep everybody in the process

1. Teach the team Kubernetes definitions

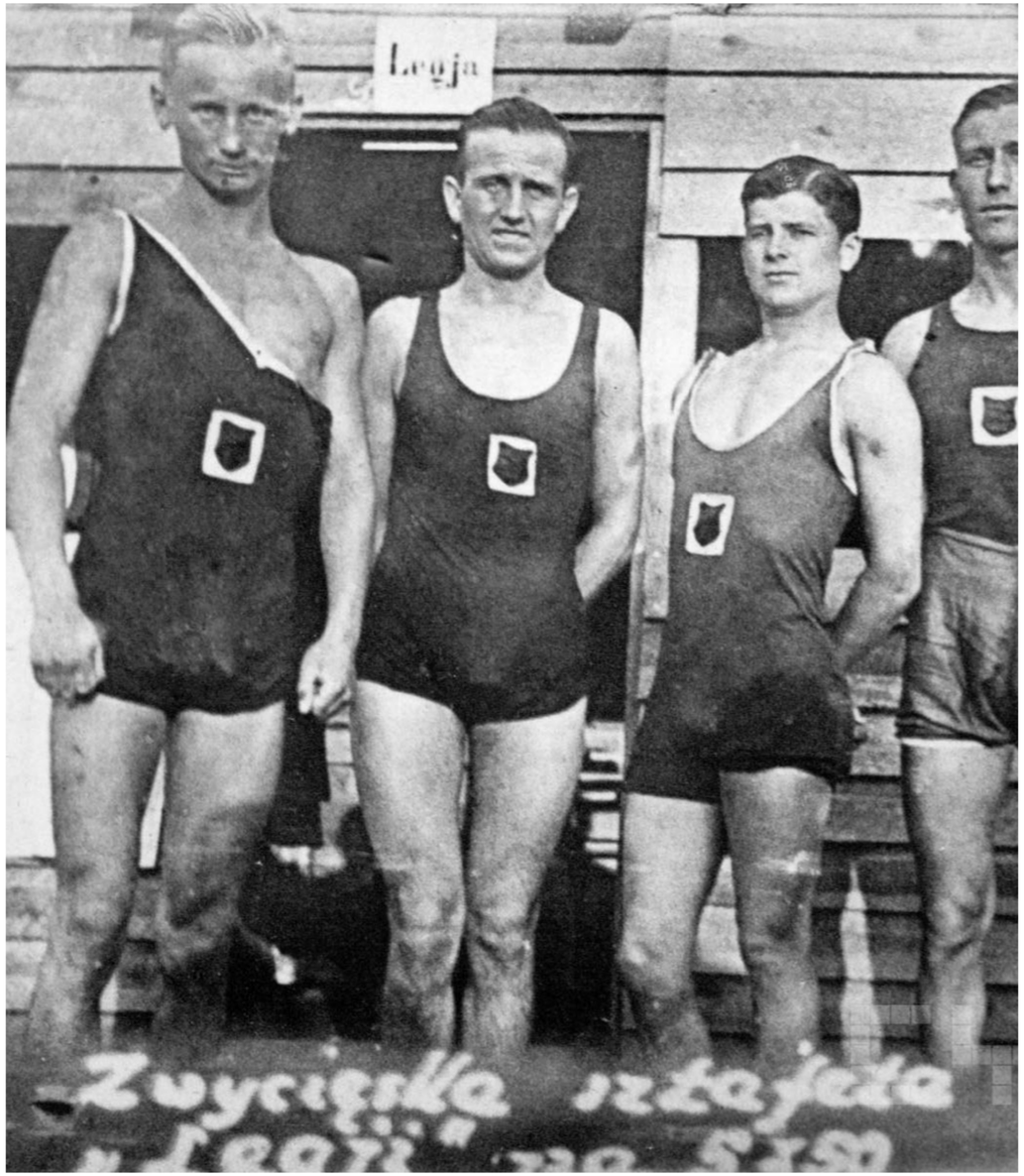
2. Keep the process understandable



Keep everybody in the process

3. Keep K8S files, … easy to read

4. Do not terrorize with how-amazing-Kubernetes-is :D



and hide it!

your-platform.git

- `git tag` and `push`
- branches: dev, staging, master
- Generated k8s files
- PR to production

Similar to the Kelsey Hightower approach

KUBERNETES

- Pure and generated
  from simple templates with ${ENV_VAR}
- 2x kubernetes operators

GIT REPO

```
|- tools
|     |- kube-service.yaml
|     \- kube-deployment.yaml
|
|- Dockerfile
|- VERSION
\- Makefile
```

Makefile

```
SERVICE_NAME=v-connector
DOCKER_REGISTRY=eu.gcr.io

test: test_short test_integration

run_local:

release: docker_build docker_push

kube_create_config:

deploy:
```

Copy&Paste from the project to project

Effective Kubernetes

https://www.flickr.com/photos/bruno_brujah/

- Start
  with small iterations
- Learn-as-you-go

Effective Kubernetes

https://www.flickr.com/photos/bruno_brujah/

- Simplicity
- Keep your Kubernetes understandable

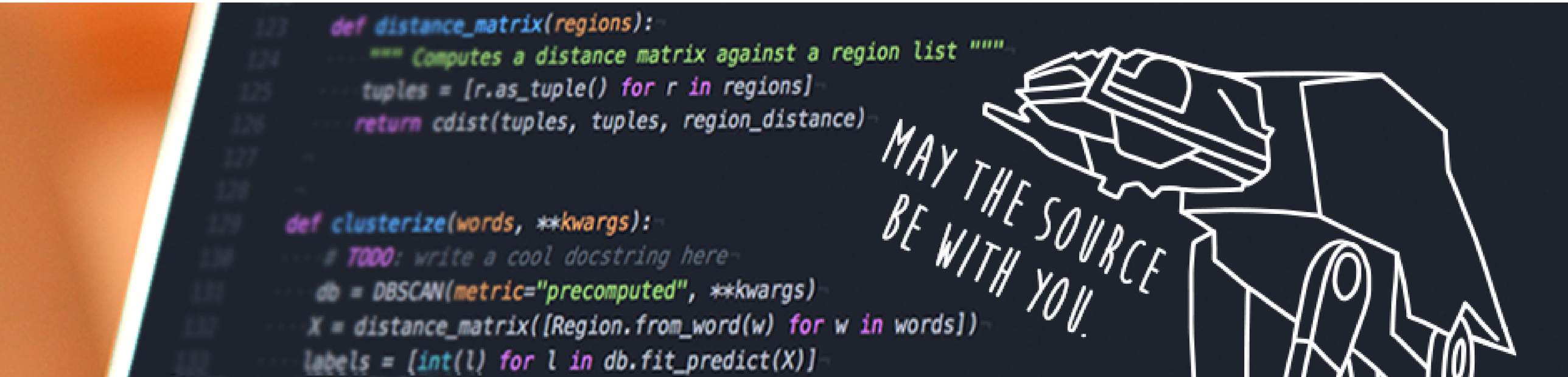
Hope k8s team keep it this way

Kubernetes VS Startup

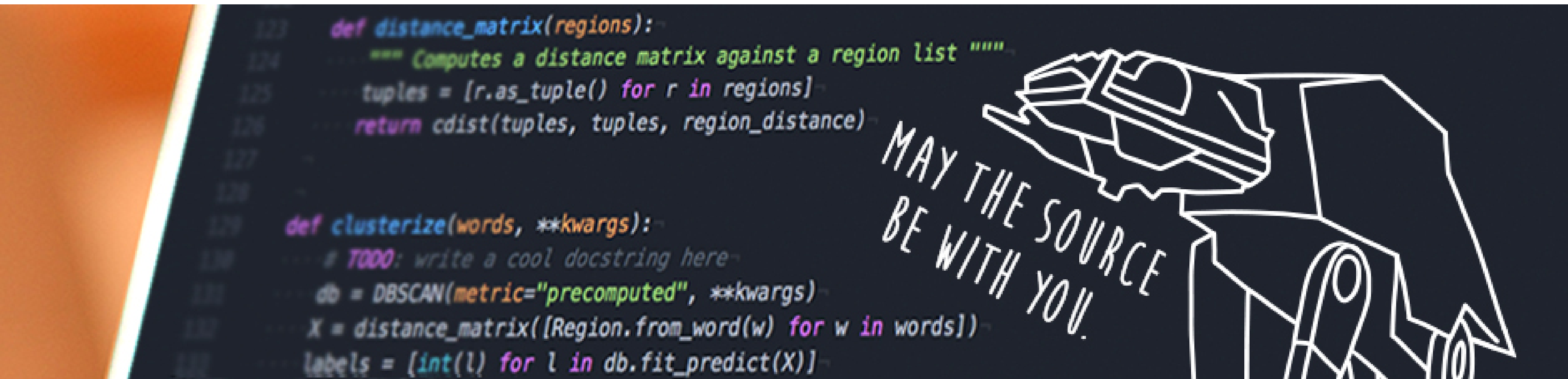
- VM with application run in dockers
- After MVP version X, go to k8s
- Learn as-you-go
- Do not get high on all CN projects :)

THANK YOU. QUESTIONS?

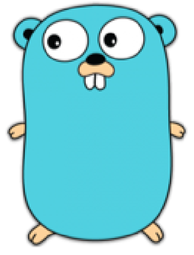
btw. https://github.com/hypatos/jobs



github.com/wojciech12



Hypatos Go



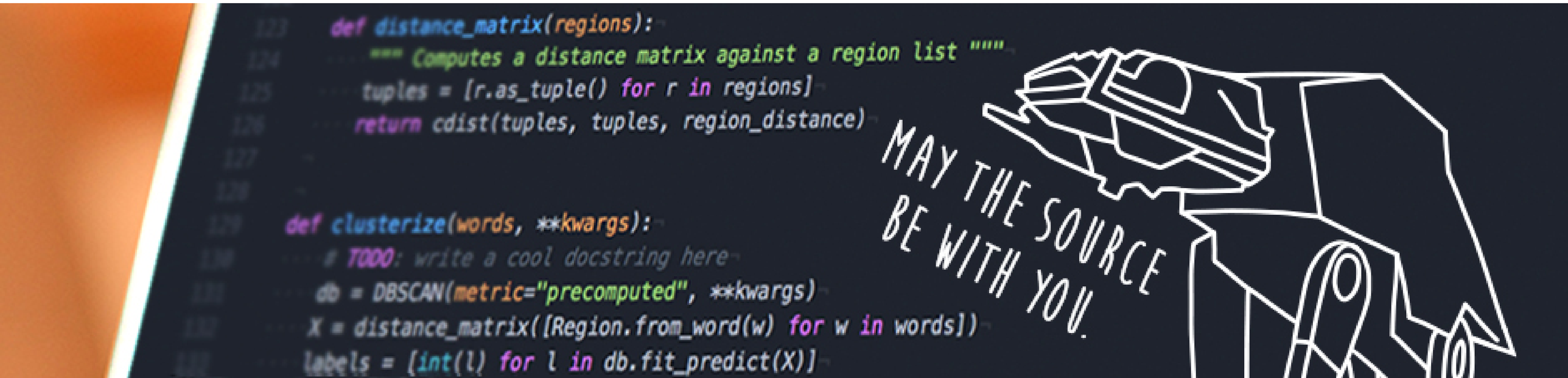
PYTORCH

TensorFlow™

amazon web services™

Azure

BACKUP SLIDES



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

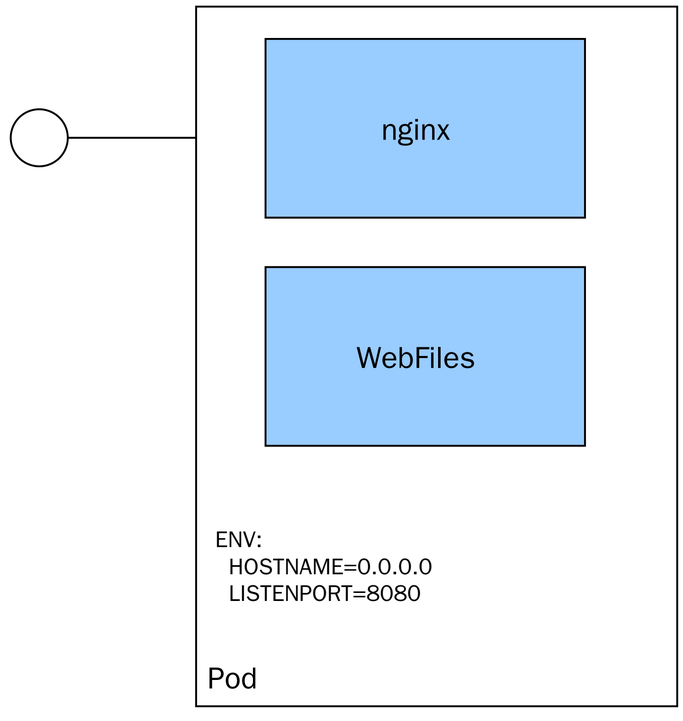
BACKUPS:

- old-school backups
- replications

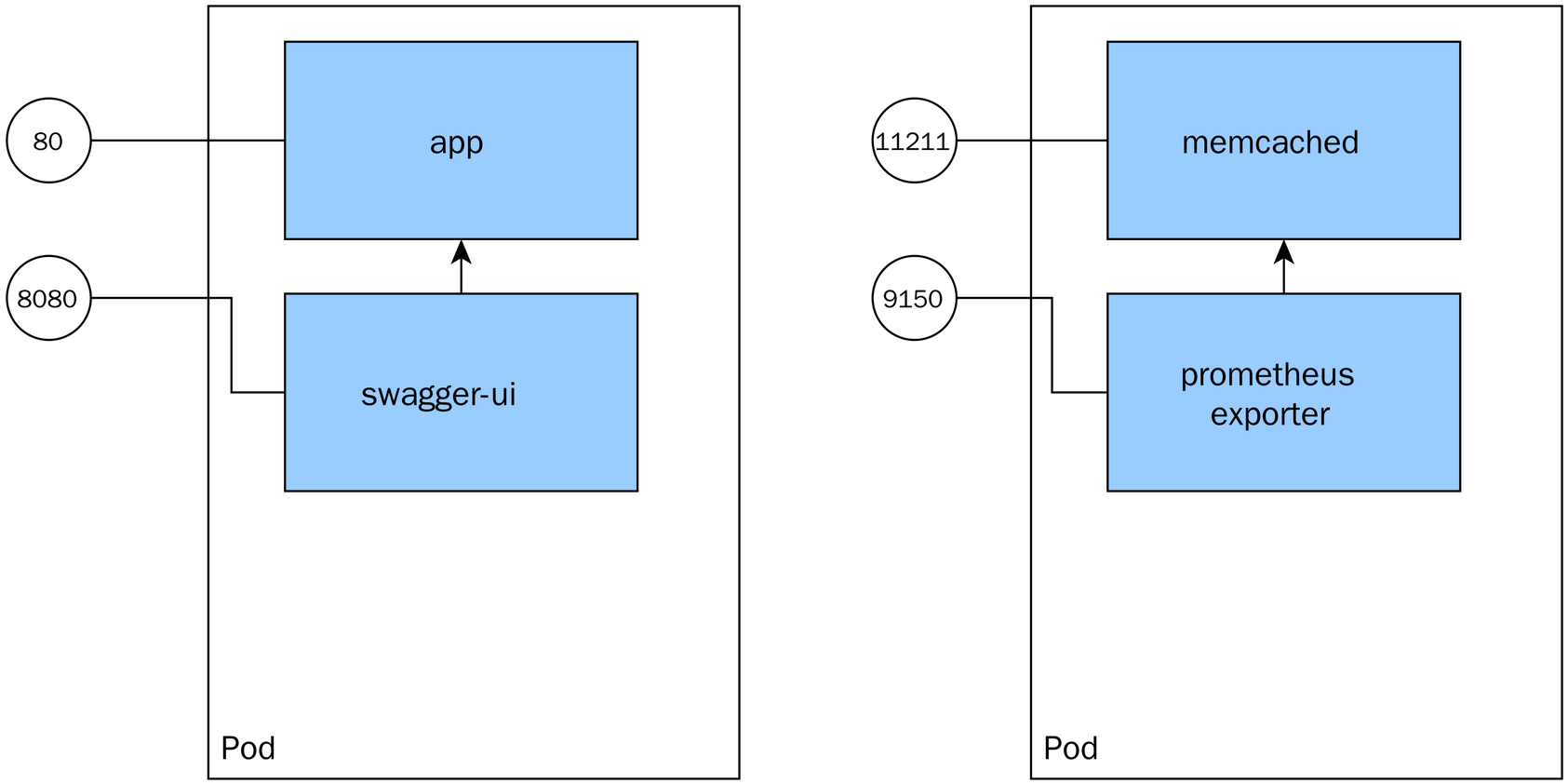
KUBERNETES CONCEPTS+

PODS

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



SIDE-CARS


80
app
8080
swagger-ui
Pod
11211
memcached
9150
prometheus
exporter
Pod