EFFECTIVE PLATFORM BUILDING WITH KUBERNETES

Wojciech Barczyński - SMACC.io | Hypatos.ai

18 December 2018

WOJCIECH BARCZYŃSKI

- Lead Software Developer & System Engineer
- Organizer Golang Warsaw Meetup
- Visiting Lecturer at WSB and ALK
- 2019 ➡ Trainings & Consultancy

STORY

Kubernetes + Go + ...

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

I do not like Infra

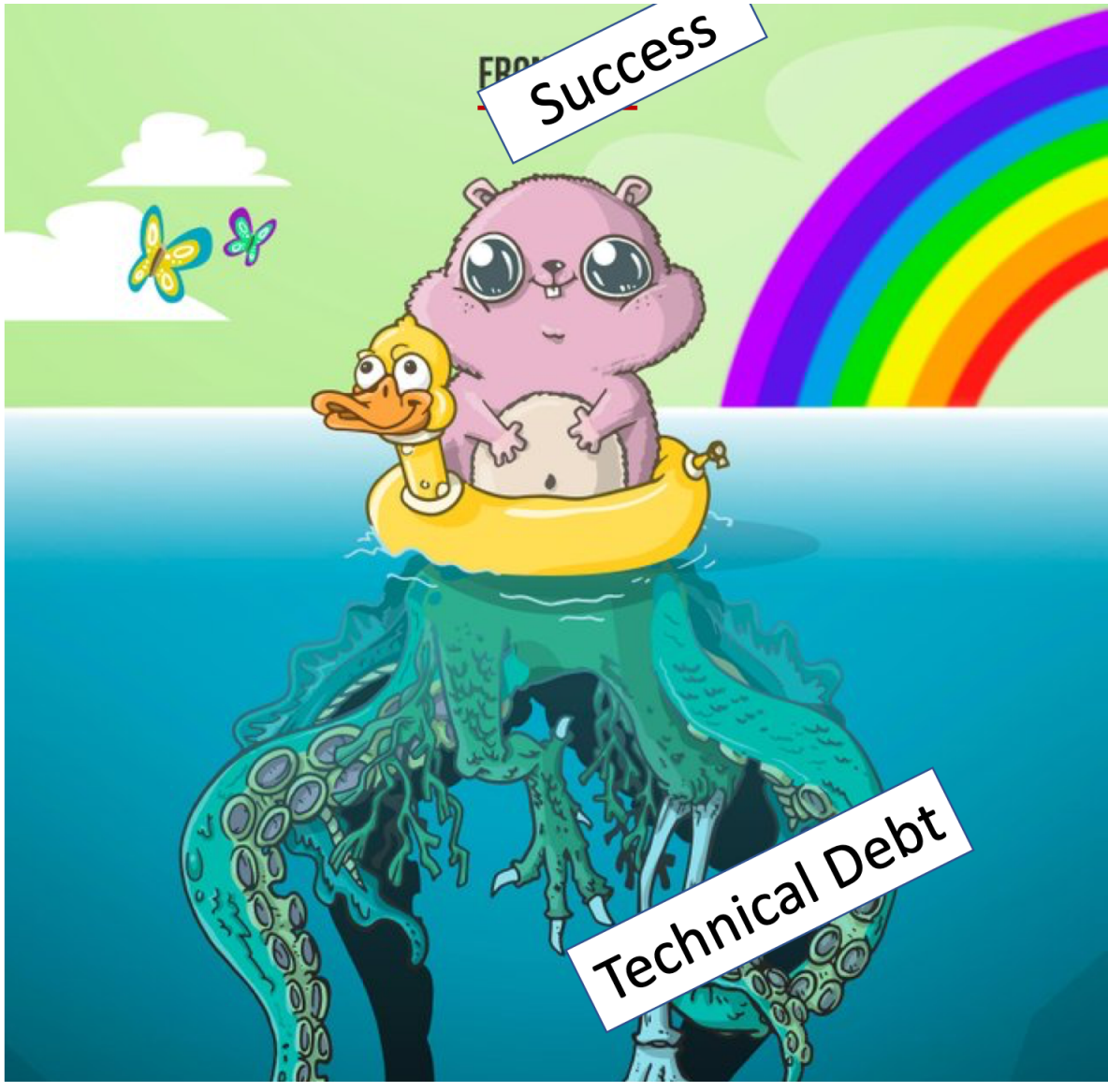
Slow delivery

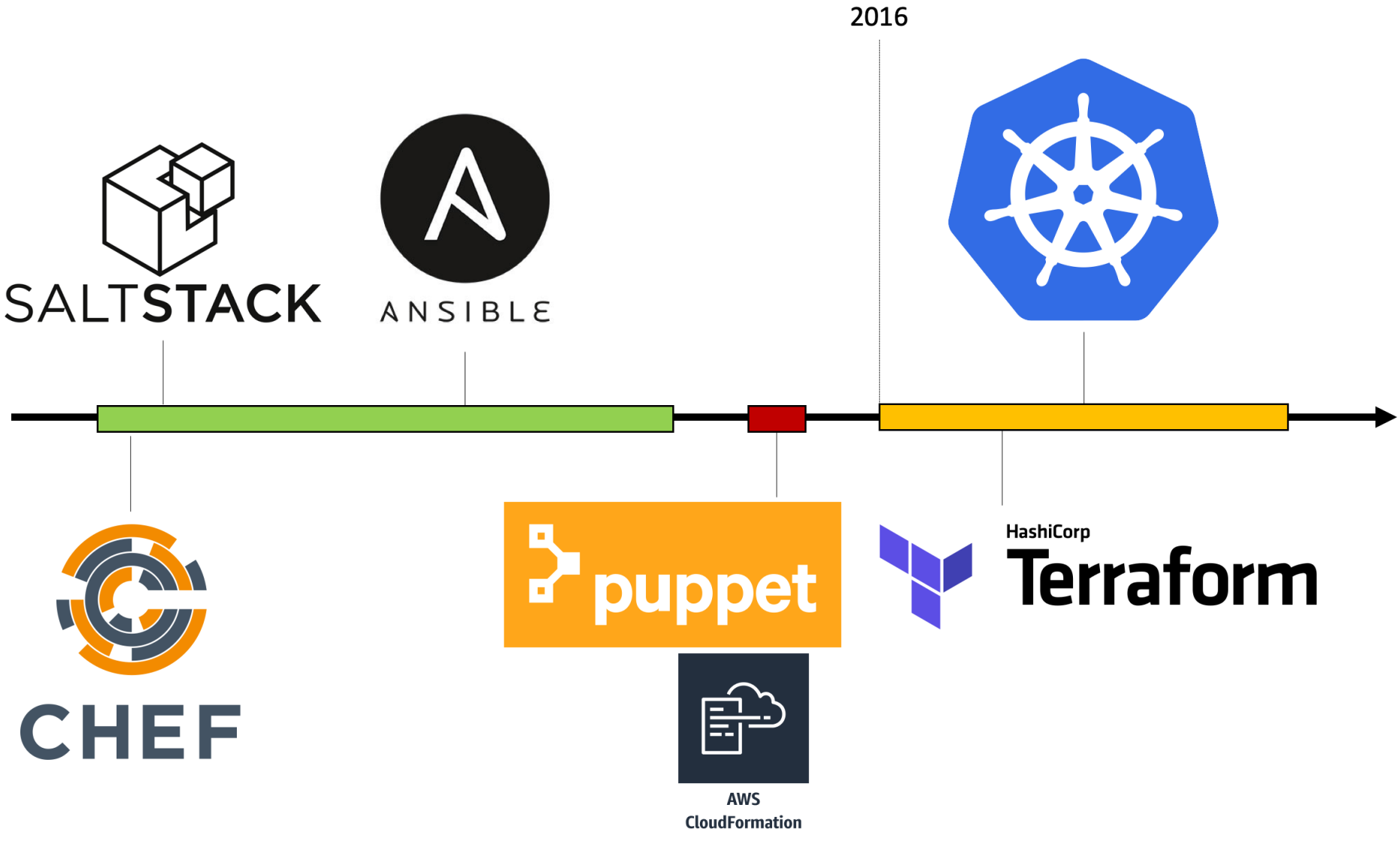
Continuous Deployment?

Fear

Frustration

XX% Idle Machines




2016
SALTSTACK
ANSIBLE
CHEF
puppet
AWS
CloudFormation
HashiCorp
Terraform

Black (Blue) Box

Infrastructure (almost) invisible

Easy* Continuous Deployment

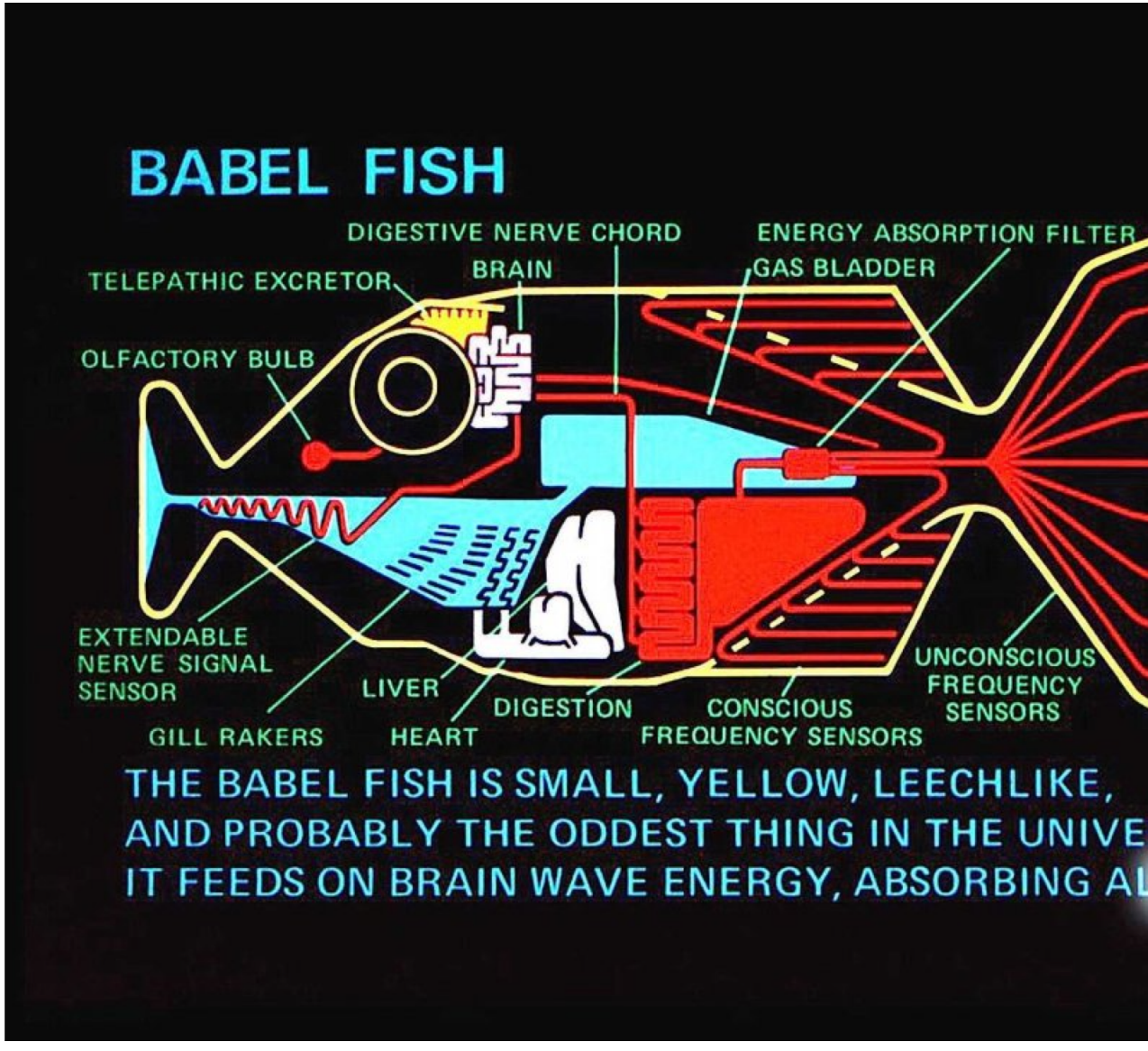
https://en.wikipedia.org/wiki/File:Dr_Who_(316350537).jpg

Common

Language

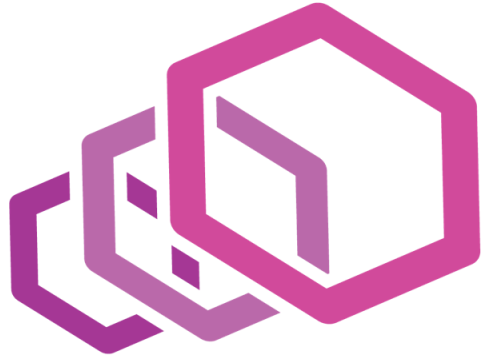
Artifacts

Platform

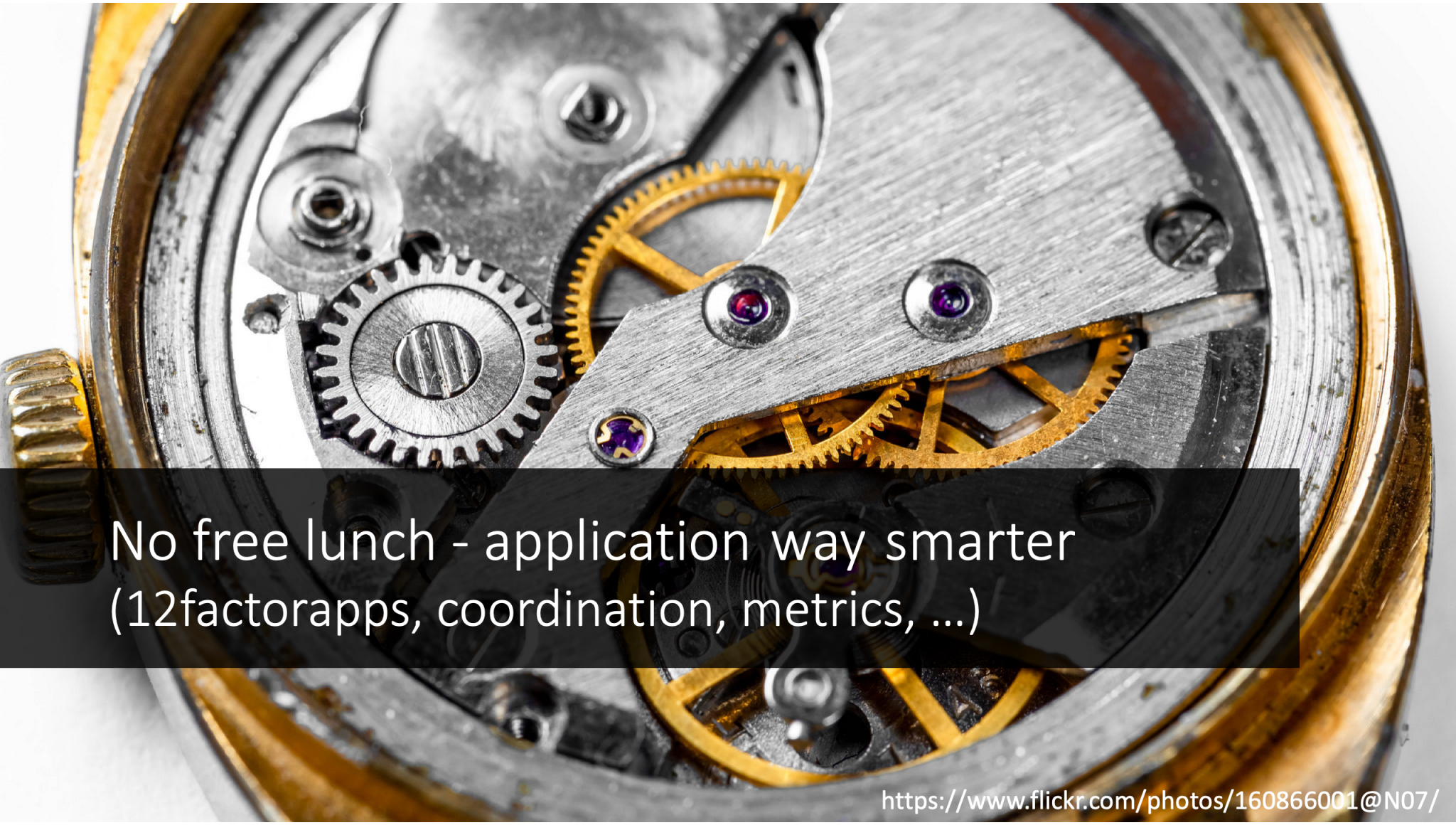


Learn-as-you-go

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



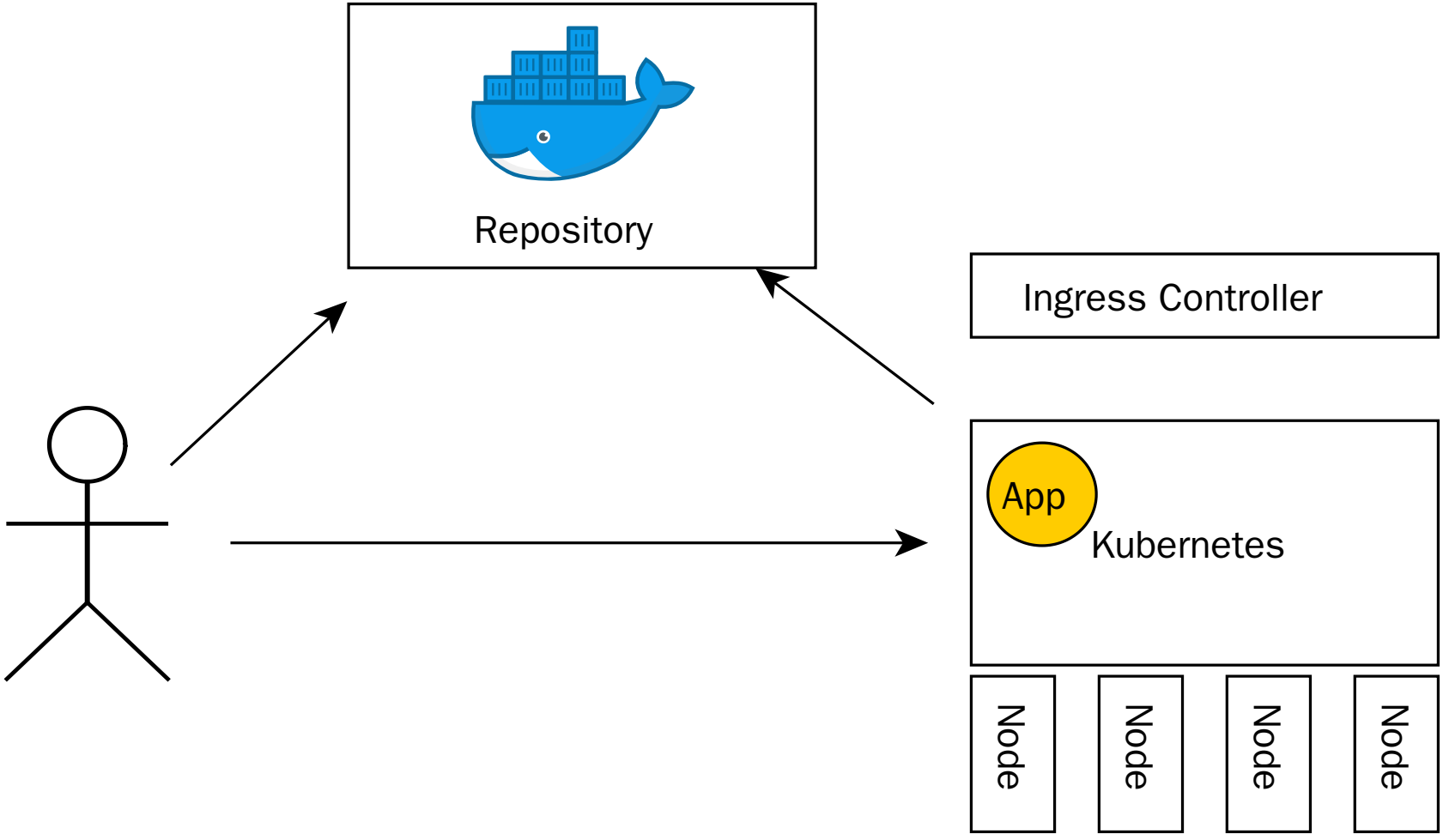
envoy


No free lunch - application way smarter
(12factorapps, coordination, metrics, …)
https://www.flickr.com/photos/160866001@N07/

KUBERNETES

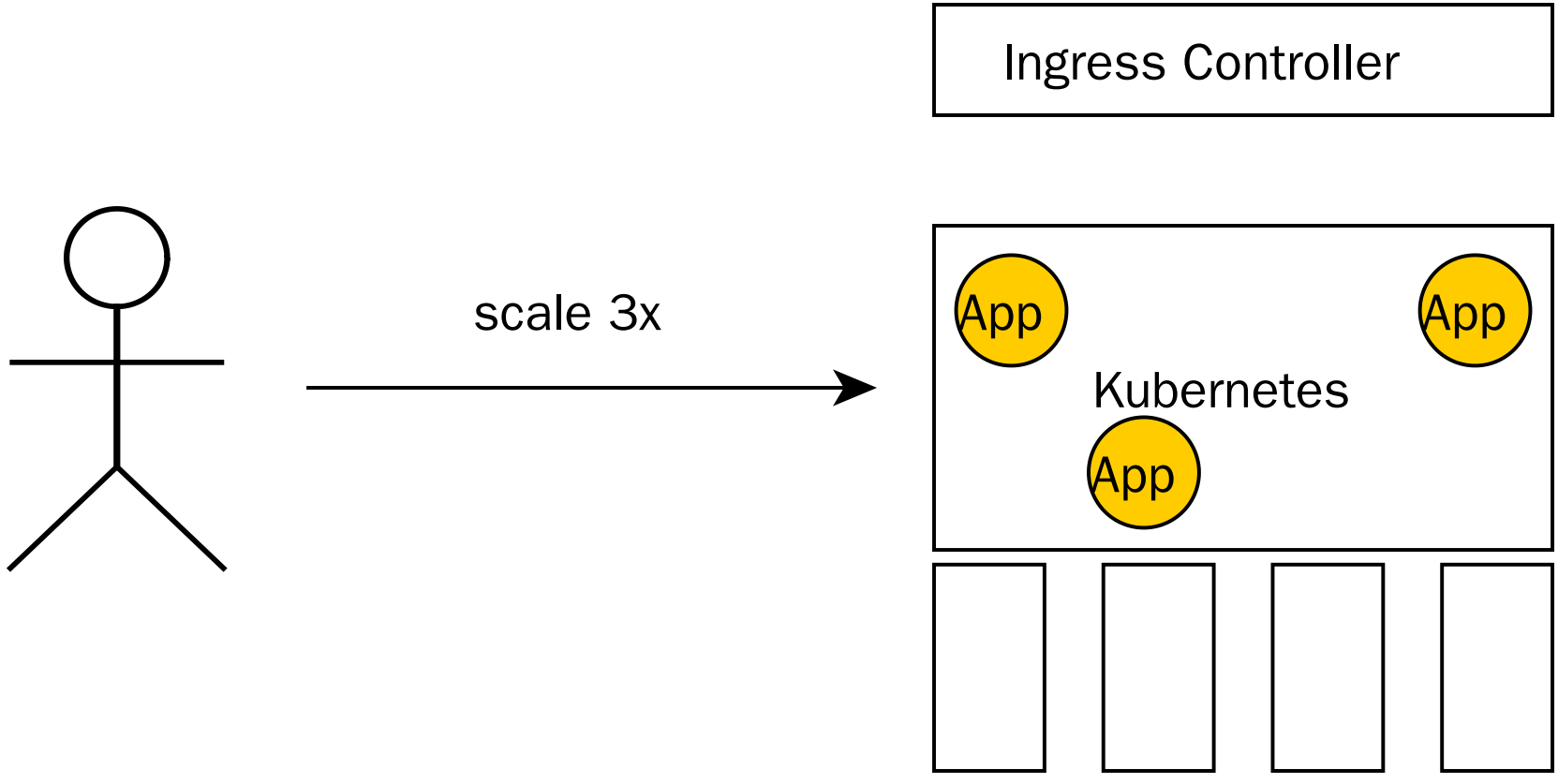
- Container management
- Battery for 12factor apps
- ...heading to integration platform
- ...becoming a framework for your Xubernetes

KUBERNETES



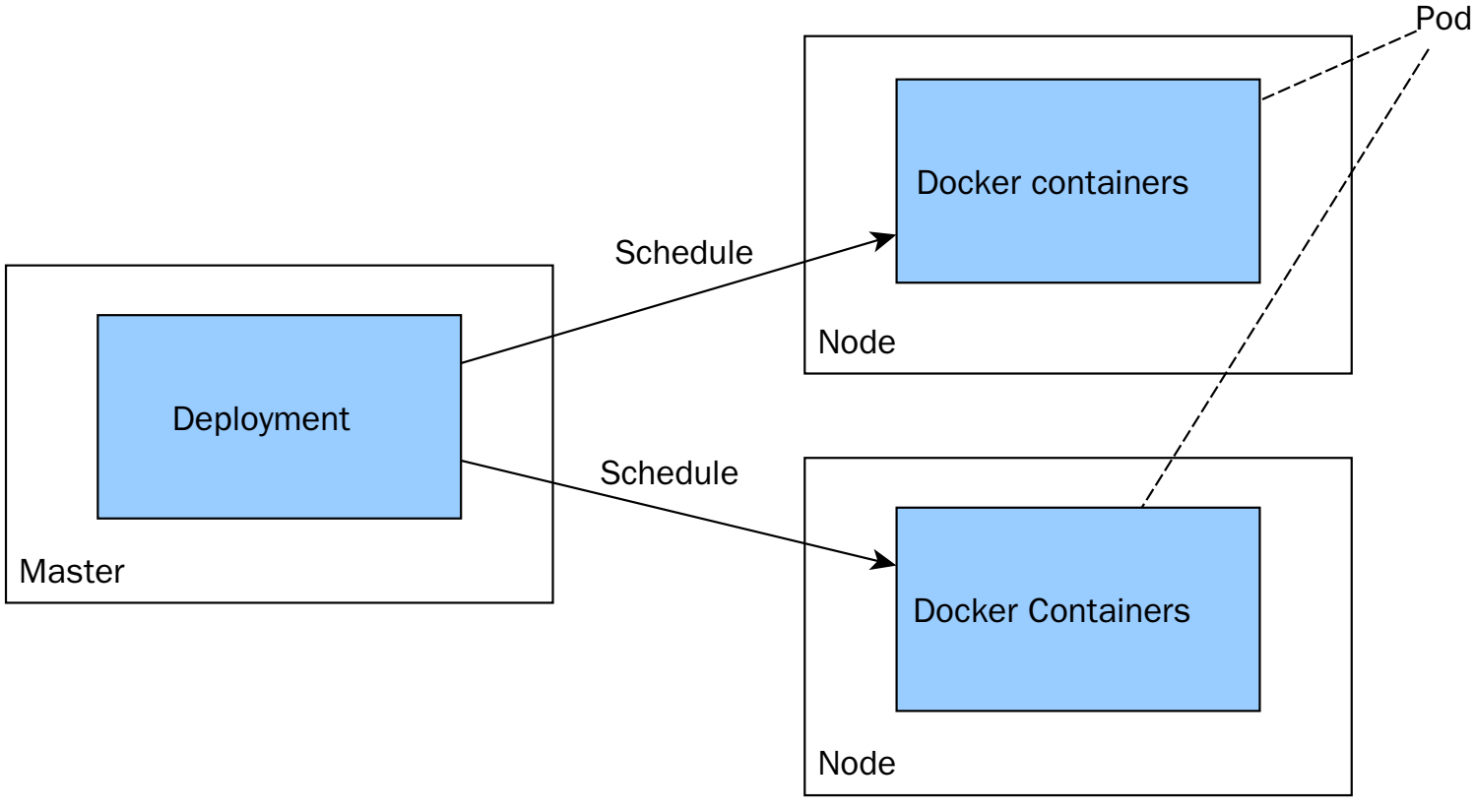

```
make docker_push; kubectl create -f app-srv-dpl.yaml
```

SCALE UP! SCALE DOWN!

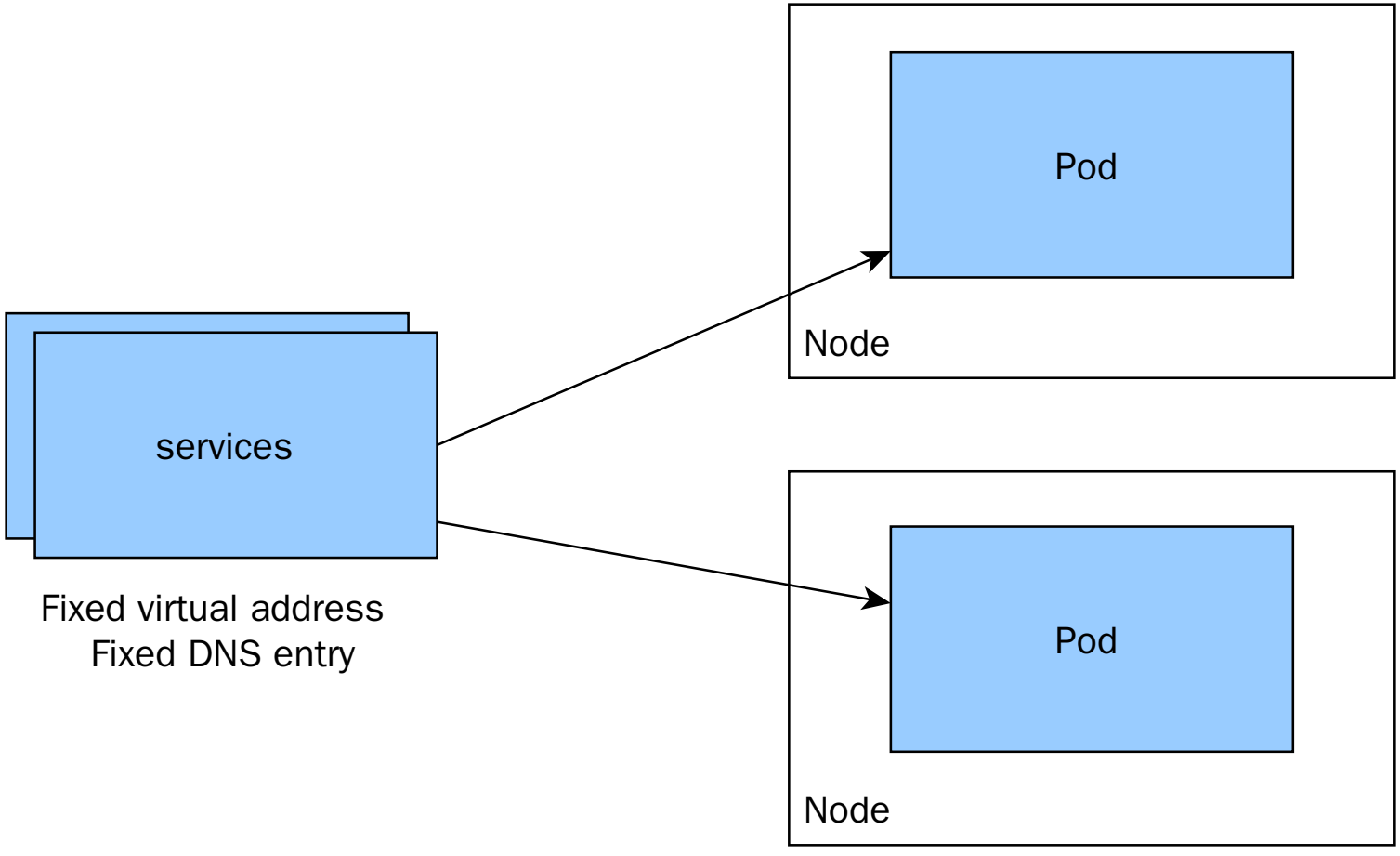



```
kubectl --replicas=3 -f app-srv-dpl.yaml
```

DEPLOYMENT AND PODS



SERVICE AND PODS



Service matches pods based on labels

INGRESS

Pattern	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

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

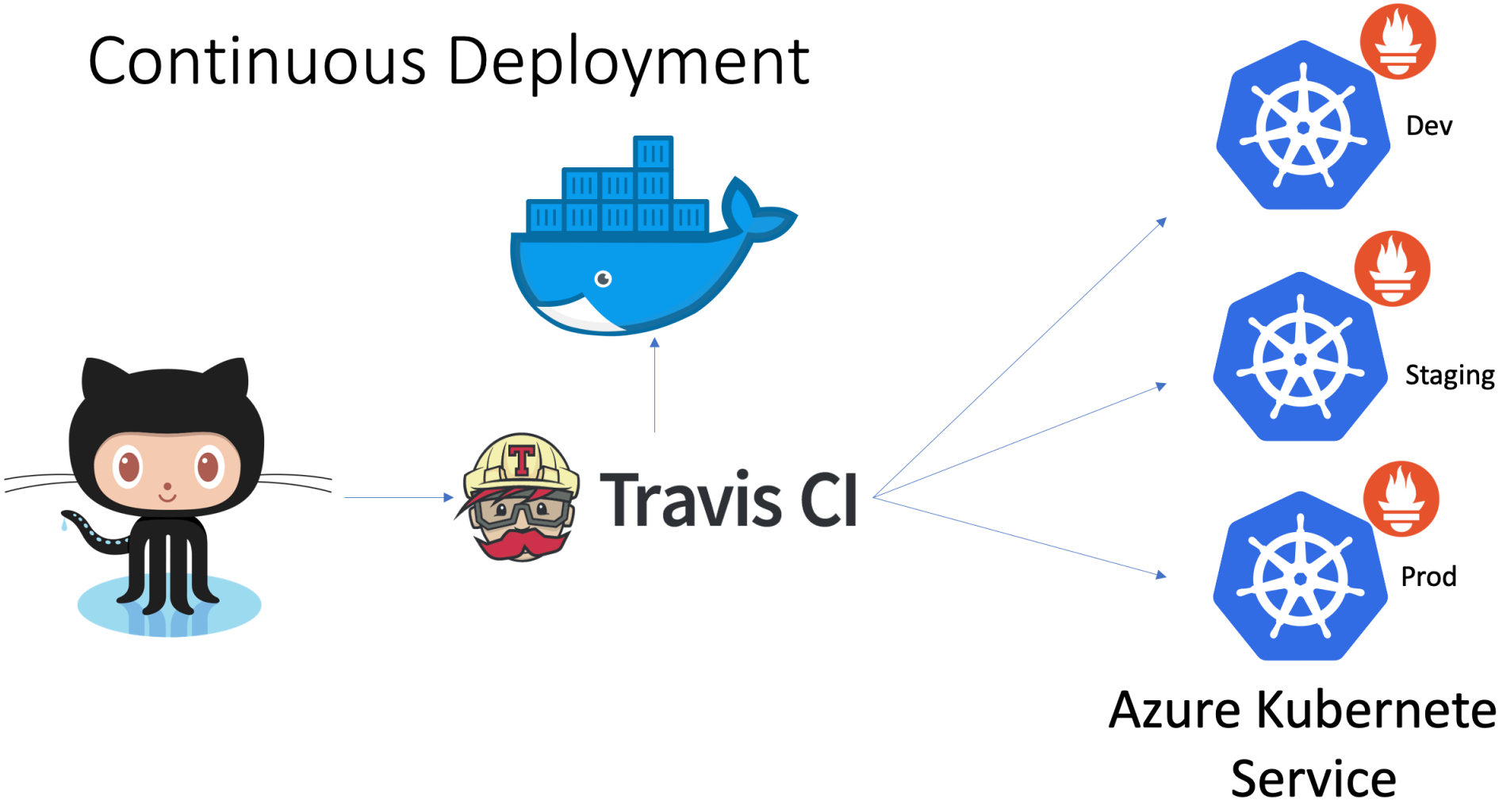
ALL HAIL THE GIT!

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

+ all tools -> CRD


Keep it simple
https://www.flickr.com/photos/bruno_brujah/

Continuous Deployment



Continuous Deployment

1. make run_local
2. Code on master → develop env

Travis CI

Development

Continuous Deployment

3. Git tag → staging env
4. PR accepted → production env

Travis CI

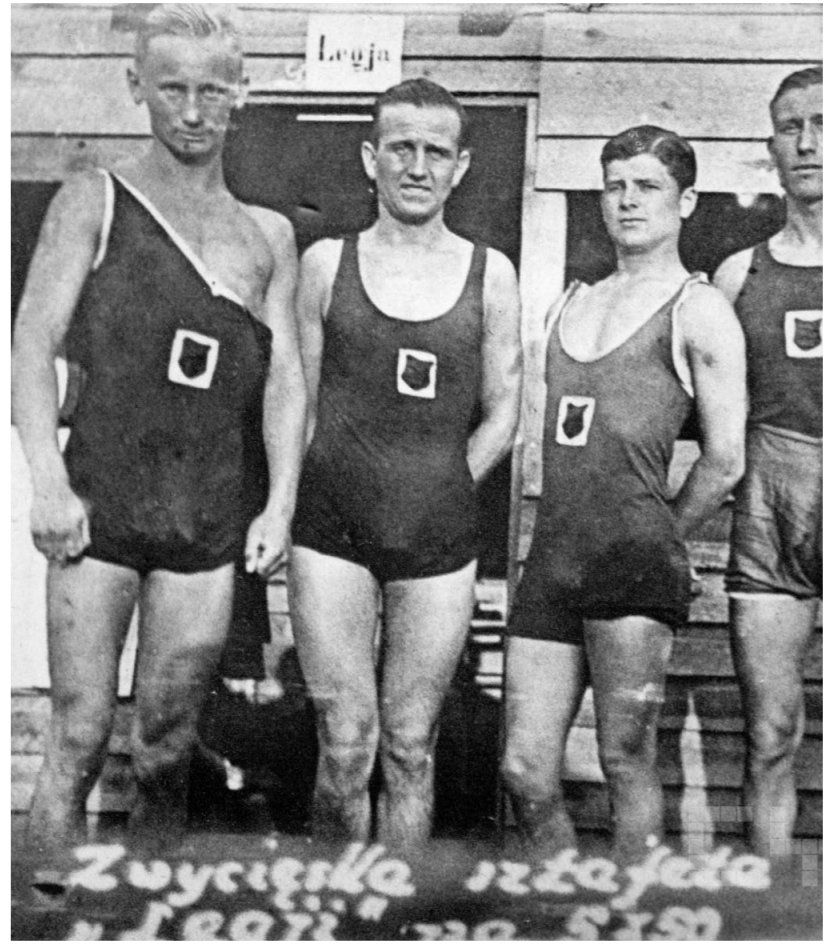
Staging

Production

Keep everybody in the process

1. Teach the team Kubernetes definitions

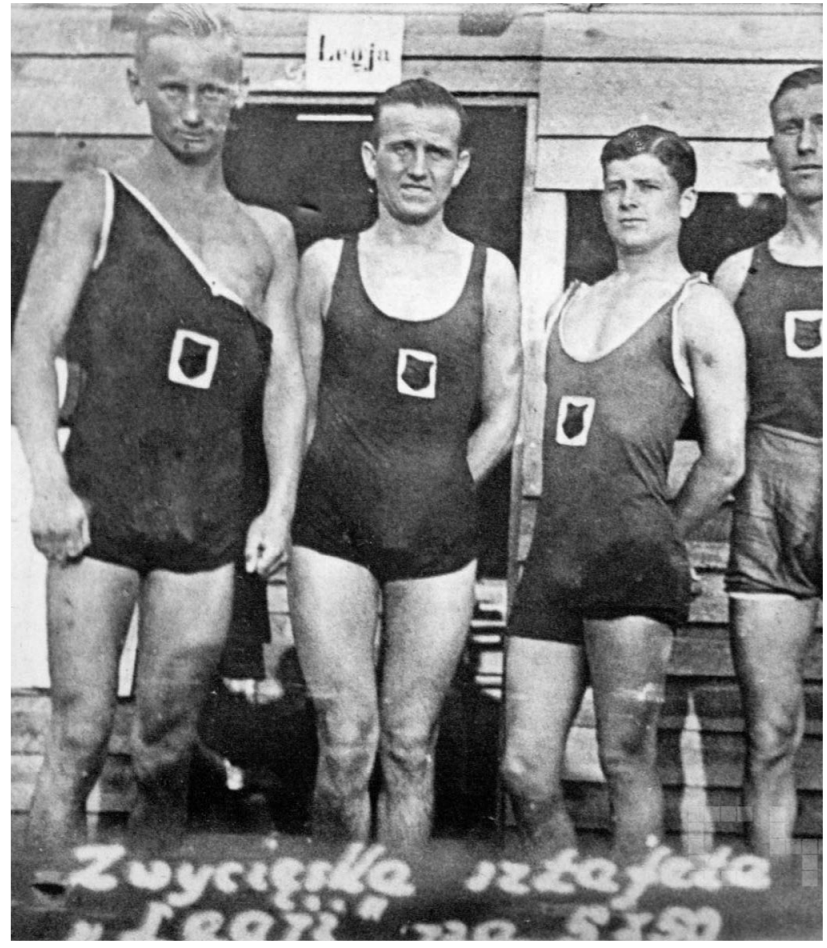
2. Keep the process understandable



Keep everybody in the process

3. Keep K8S files, … easy to read

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



Keep everybody in the process

Copy & Paste:

1. Makefile
2. Kubernetes files
3. TravisCI

```
curl https://github.com/smacc-ci/deploy.sh | bash
```

Conventions over tools!

- Common conventions for repos
- No a single deploying tool
- No encrypted data in repo

ps. Only when you are really really ready.

ALL IN GIT

smacc-platform.git

- branches: dev, staging, master
- Generated k8s files
- ENV variables for each of environment

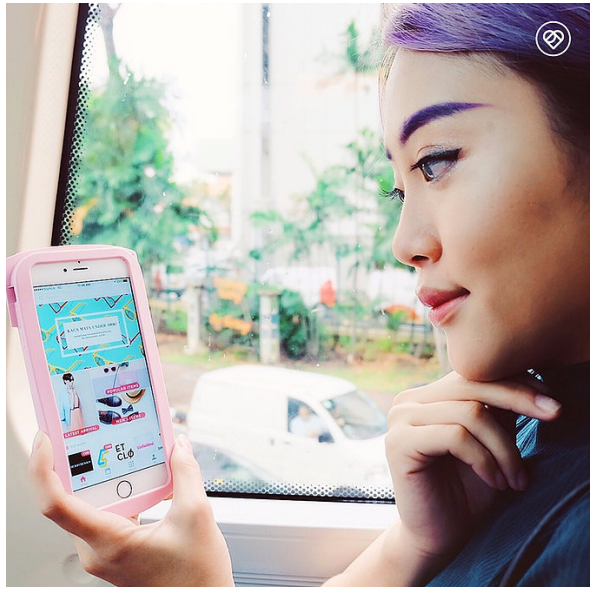
Similar to the Kelsey Hightower approach

KUBERNETES

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

LYKE

- E-commerce Indonesia
- Mobile-only
- 50k+ users
- 2M downloads
- Top 10 Fashion Apps Google Play Store



http://www.news.getlyke.com/single-post/2016/12/02/Introducing-the-New-Beautiful-LYKE

Now JollyChic Indonesia

CHALLENGES

- 50+ VMs in Amazon, 1 VM - 1 App
- 65% Idle machine $$$
- Puppet with manual deployment process
- Fear, Forgotten components
- Performance issues

APPROACH

1. Simplify Infrastructure
2. Change the Development Practices
   (12factor + kubernetes)
3. Make everybody learn Kubernetes
4. Change the Work Organization

see: Conway's law

SIMPLIFY

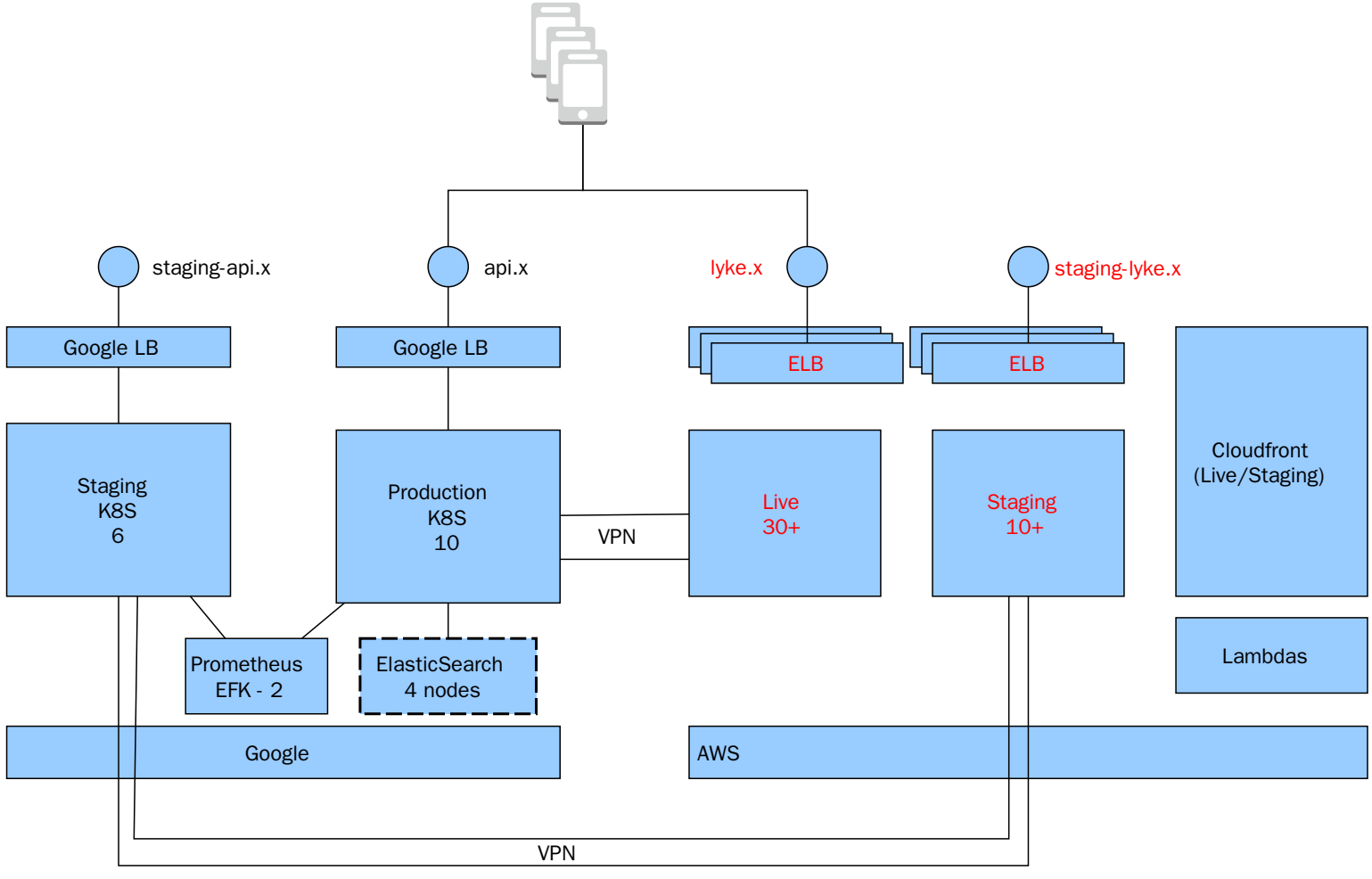
1. Kubernetes with Google Kubernetes Engine
2. Terraform for all new
3. Database-as-a-service

SIMPLIFY

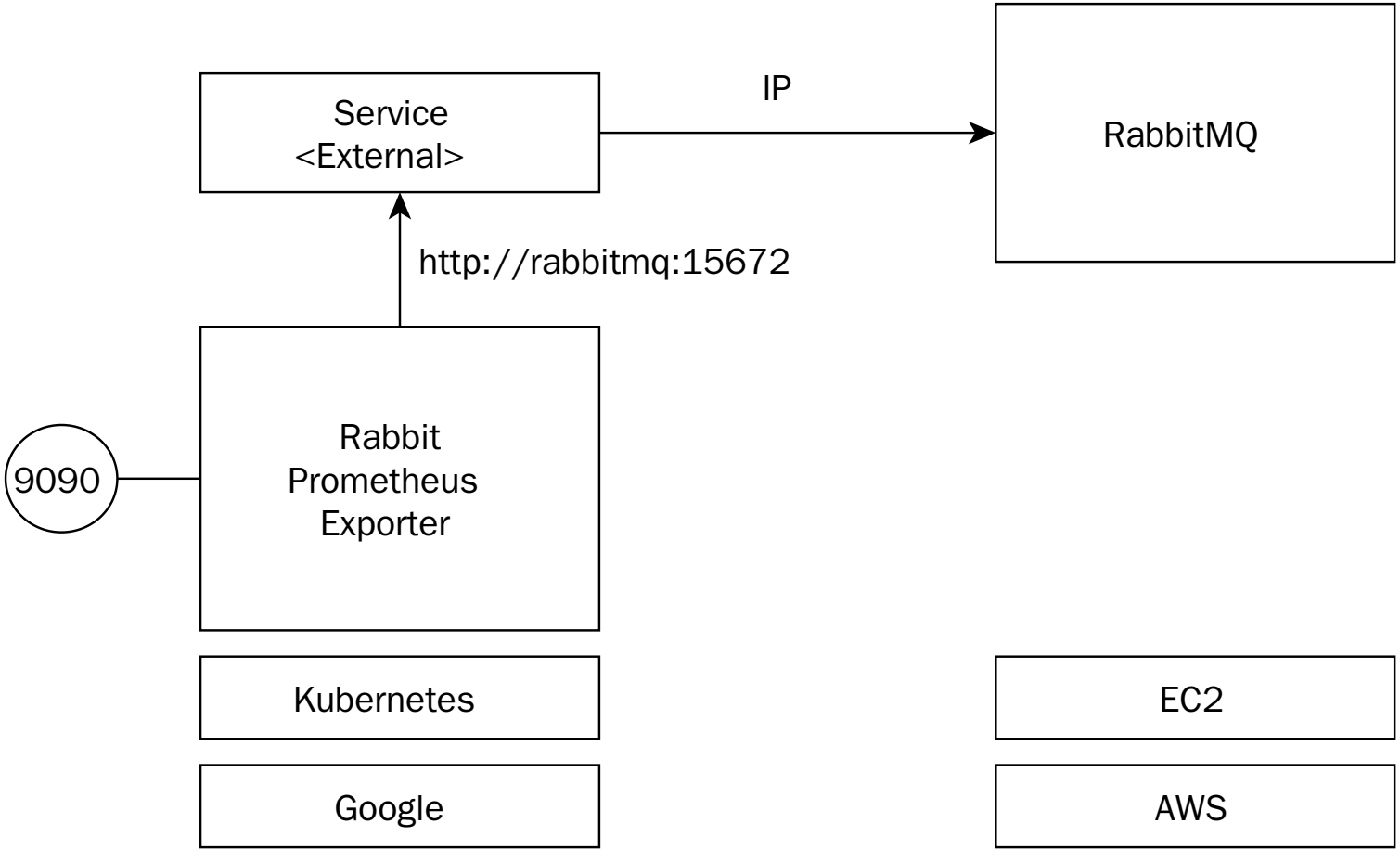
1. Prometheus, AlertManager, and Grafana
2. Elasticsearch-Fluentd-Kibana
3. Google Identity-Aware-Proxy
   to protect all dev dashboards

One person efford

Architecture During Migration



Bridge the new with old



Monitor legacy with new stack

CONTINUOUS DEPLOYMENT

- branch-based:
  - master
  - staging
  - production
- repo independent

Good transition strategy

CONTINUOUS DEPLOYMENT

- Tests
- Build docker
- Deploy to Google Container Registry and k8s
- No config/secrets applied

Vault planned; moving secrets to runtime

GIT REPO

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

Makefile

```
SERVICE_NAME=v-connector
GCP_DOCKER_REGISTRY=eu.gcr.io
test: test_short test_integration

run_local:

docker_build: docker_push

kube_create_config:

kube_apply:

kube_deploy:
```

Copy&Paste from the project to project

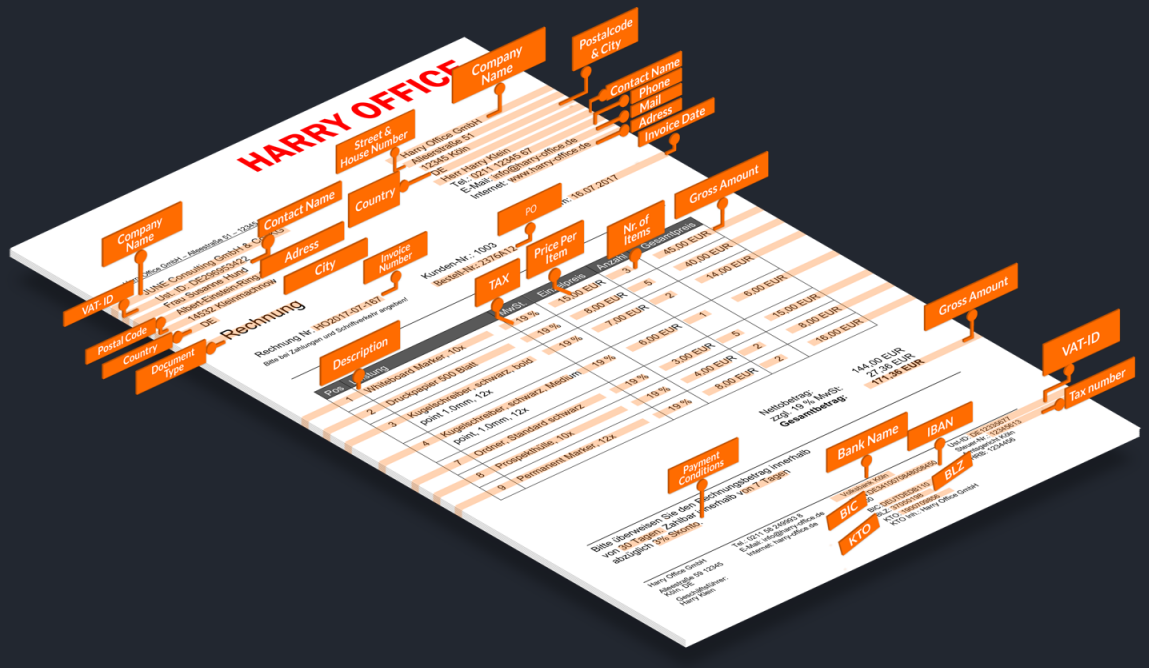
WHAT WORKED

1. Copy&Paste Makefile and k8s yaml
2. Separate deployments a good **transition** strategy

SMACC

- Machine Learning FinTech
- SaaS and API platform
- From Enterprise (Deutsche Bank, AoK) to SME
- Well-known FinTech Startup in Germany

Problem SMACC solves



Hypatos

SAP

Microsoft Dynamics

DATEV

SMACC

GET TO K8S

- Legacy on AWS, experiments with AWS ECS :/
- Self-hosted K8S on ProfitBricks
- ooo... K8S can use Azure Active Directory :D

RUN EVERYWHERE!

- Get to Microsoft ScaleUp, welcome Azure
- Luckily - Azure Kubernetes Service
- With GPU integration :)
- Our OnPrem = Our OnCloud

KEEP IT SIMPLE

- az aks CLI for setting k8s - README.rst
- Terraform for everything else
- Secrets: 1Password and gopass.pw

Terraform also sets our AWS

MIGRATION WAS EASY...

- Tech-wise **yes**
- Team-wise **no**

DIFFERENCE ☠

- Two teams in Berlin and Warsaw
- Me in Warsaw

NEW EXPERIENCE

Team overwhelmed with k8s bits in:

- TravisCI, Makefiles, Yamls
- feel it is too much hasle

Maybe Keylsey Hightower was right ;)

APPROACH

- Simplify, Simplify
- Hide K8S magic
- `git tag` driven Continoues Deployment

Repo .travis.yml

```
language: go
go:
- '1.16'
services:
  - docker
install:
  - curl -sL https://${GITHUB_TOKEN}@raw.githubusercontent.com
  - if [ -f "tools/travis/install.sh" ]; then bash tools/travi
script:
  - dep ensure
  - make lint
  - make test
  - if [ -z "${TRAVIS_TAG}" ]; then make snapshot; fi;
deploy:
  - provider: script # deploy to docker repo
```

Makefile

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

Makefile only tasks for dev

CONTINUOUS DEPLOYMENT

- Github
- TravisCI
- hub.docker.com
- AKS

CONTINUOUS DEPLOYMENT

1. `git tag` and `push`
2. smacc-platform.git
3. Deploy to staging
4. PR to production

WHAT WORKED

- Hiding k8s
- Understandable CD process

WOULD DO DIFFERENT

- More sensitive to feedback

NEXT

- Scale our ML trainings on the top of k8s - kubeflow
- Use CRD to bring exteranl tools to k8s
- Keeping an eye on Istio
- Deployment tool based on missy

Effective Kubernetes

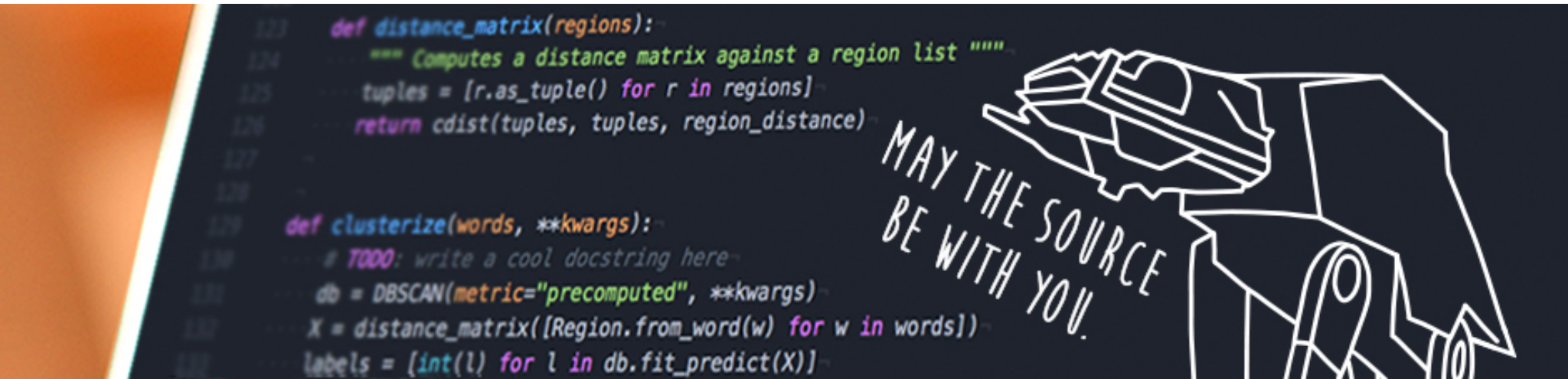
- Start with small iterations
- Learn as-you-go
- Keep Kubernetes Understable

Hope the k8s community keep it this way

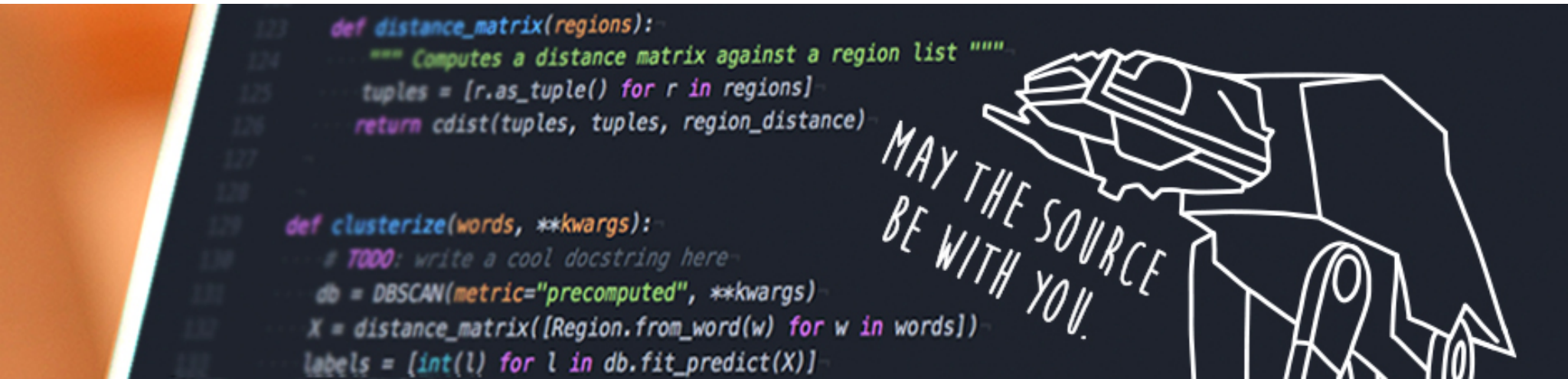
Effective Kubernetes

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

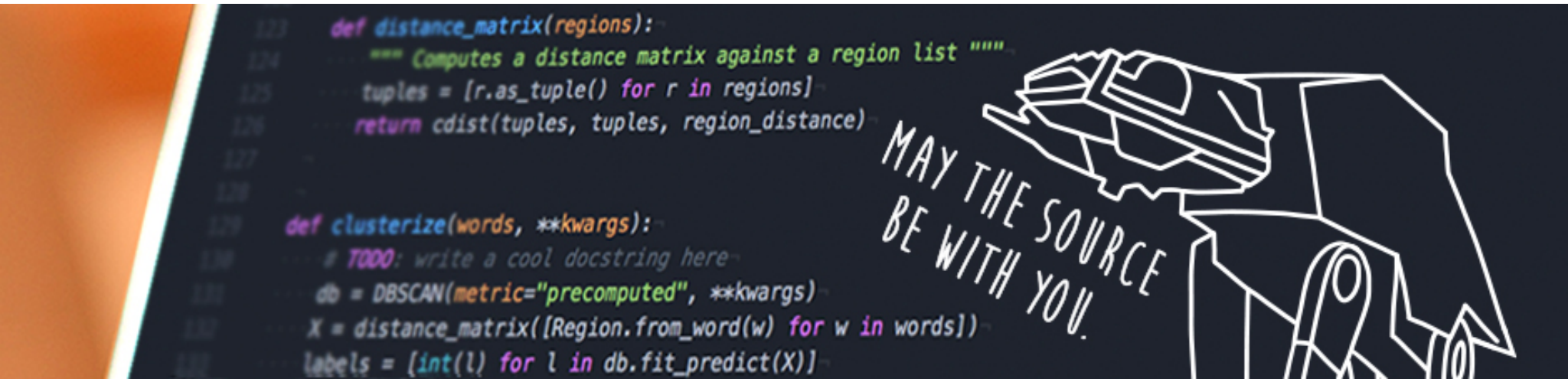
Big thanks to my colleages in Lyke and SMACC



THANK YOU. QUESTIONS?



github.com/wojciech12/talk_effective_kubernetes



SMACC

Go

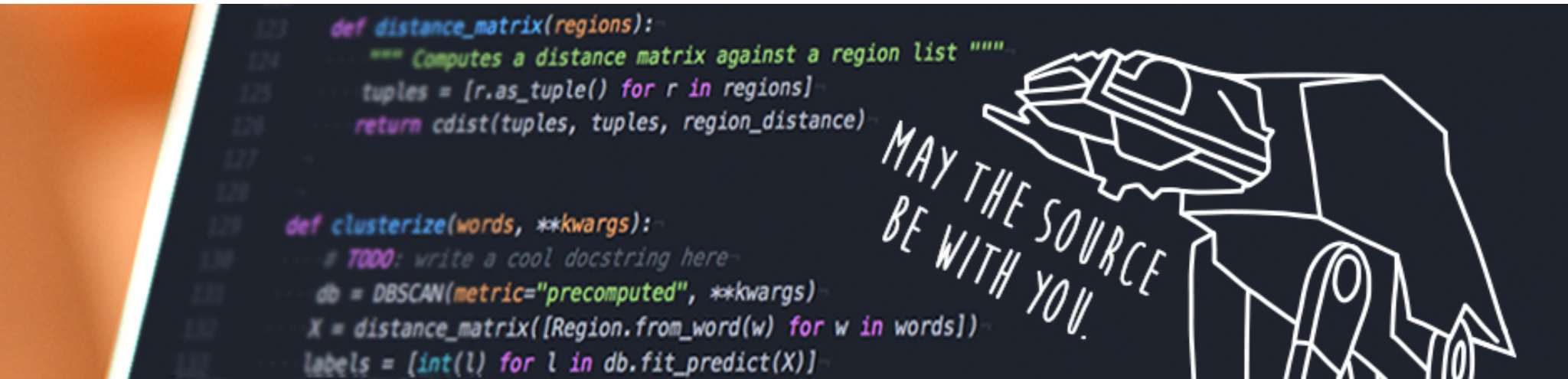
PYTORCH

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

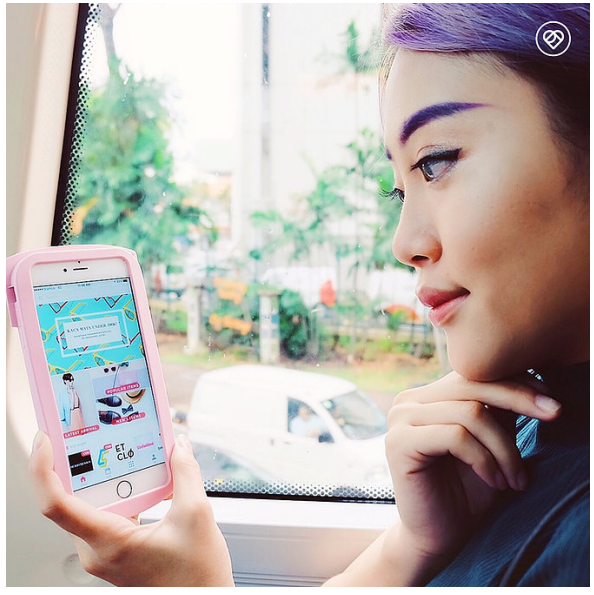
BACKUPS:

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

LYKE

LYKE

- E-commerce
- Mobile-only
- 50k+ users
- 2M downloads
- Top 10 Fashion Apps
  w Google Play Store



http://www.news.getlyke.com/single-post/2016/12/02/Introducing-the-New-Beautiful-LYKE

Now JollyChic Indonesia

GOOD PARTS

- Fast Growth
- A/B Testing
- Data-driven
- Product Manager, UI Designer, Mobile Dev, and tester - one body

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

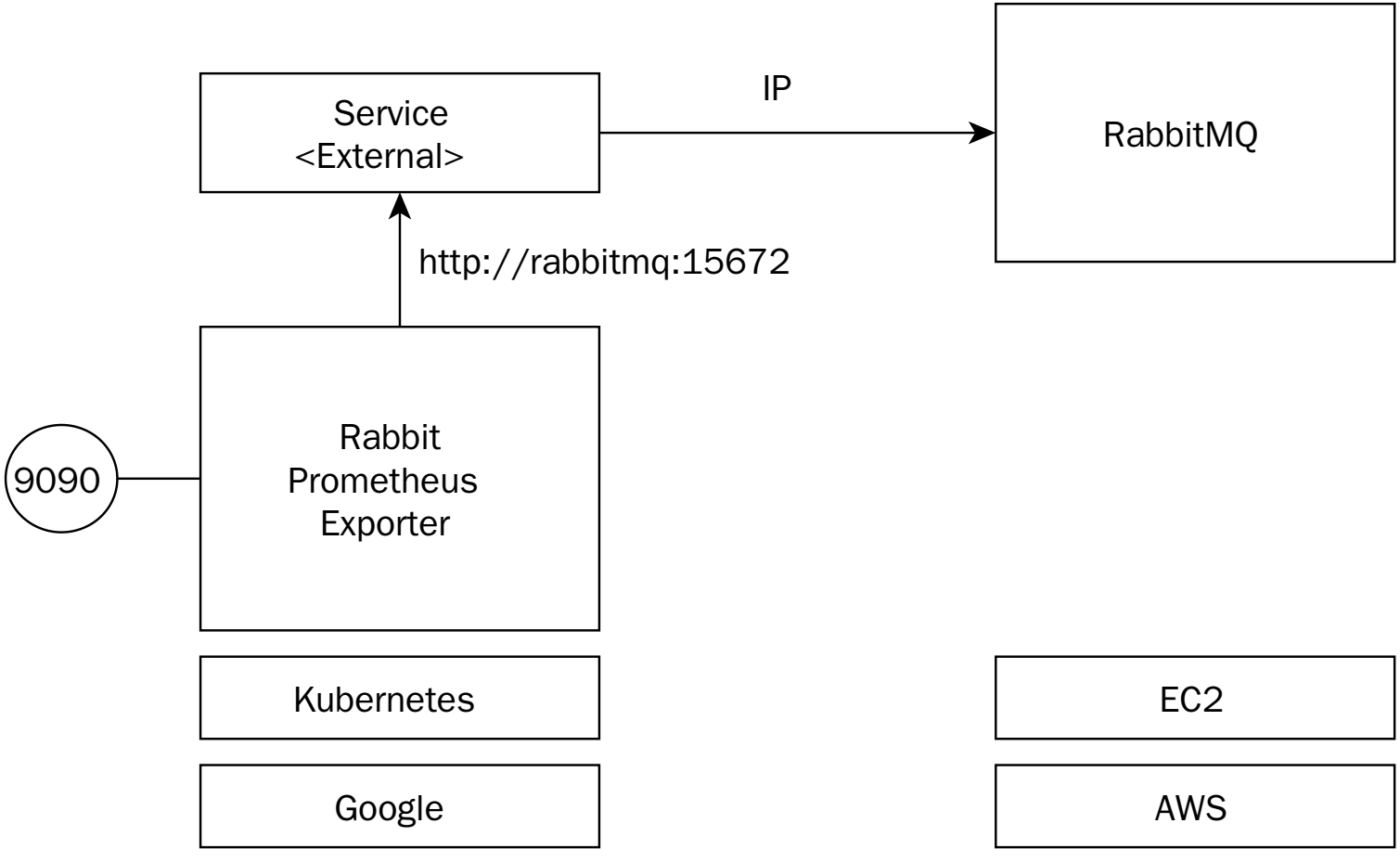
6. KEEP IT RUNNING

Bridge the new with old:

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

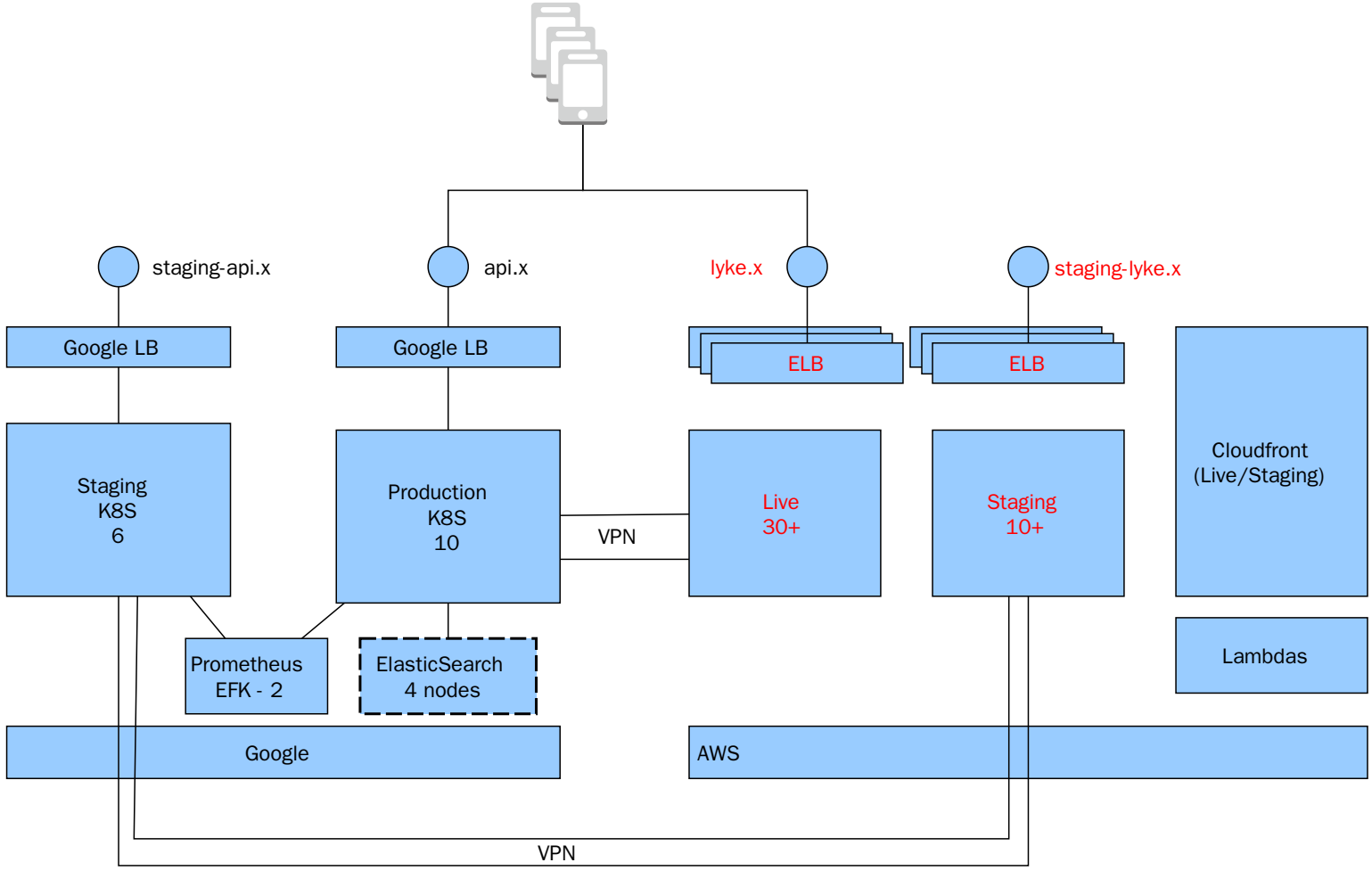
[1] hard coded IP:PORT, considered: K8S events to Consul

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

WHAT WORKED

1. Copy&Paste Makefile and k8s yaml
2. Separate deployments a good transition strategy

WHAT DID NOT WORK

1. Too many PoC, cut to 2 weeks max
2. Doing it with too small steps
3. Push back to k8s yaml [*]
4. Alert rules too hard to write

[*] With coaching, I thought, it is OK

DO DIFFERENT

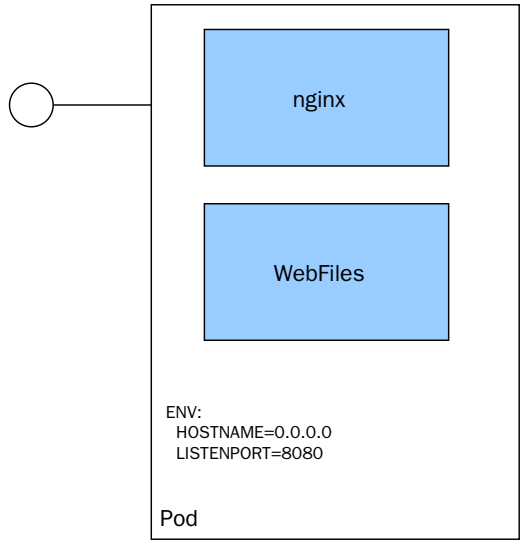
1. Move data day one
2. Make devs know it is a transition stage
3. Teach earlier about resources
4. EFK could wait
5. All-hands for a paid-XXX% weekend for migration


SMACC
Hypatos

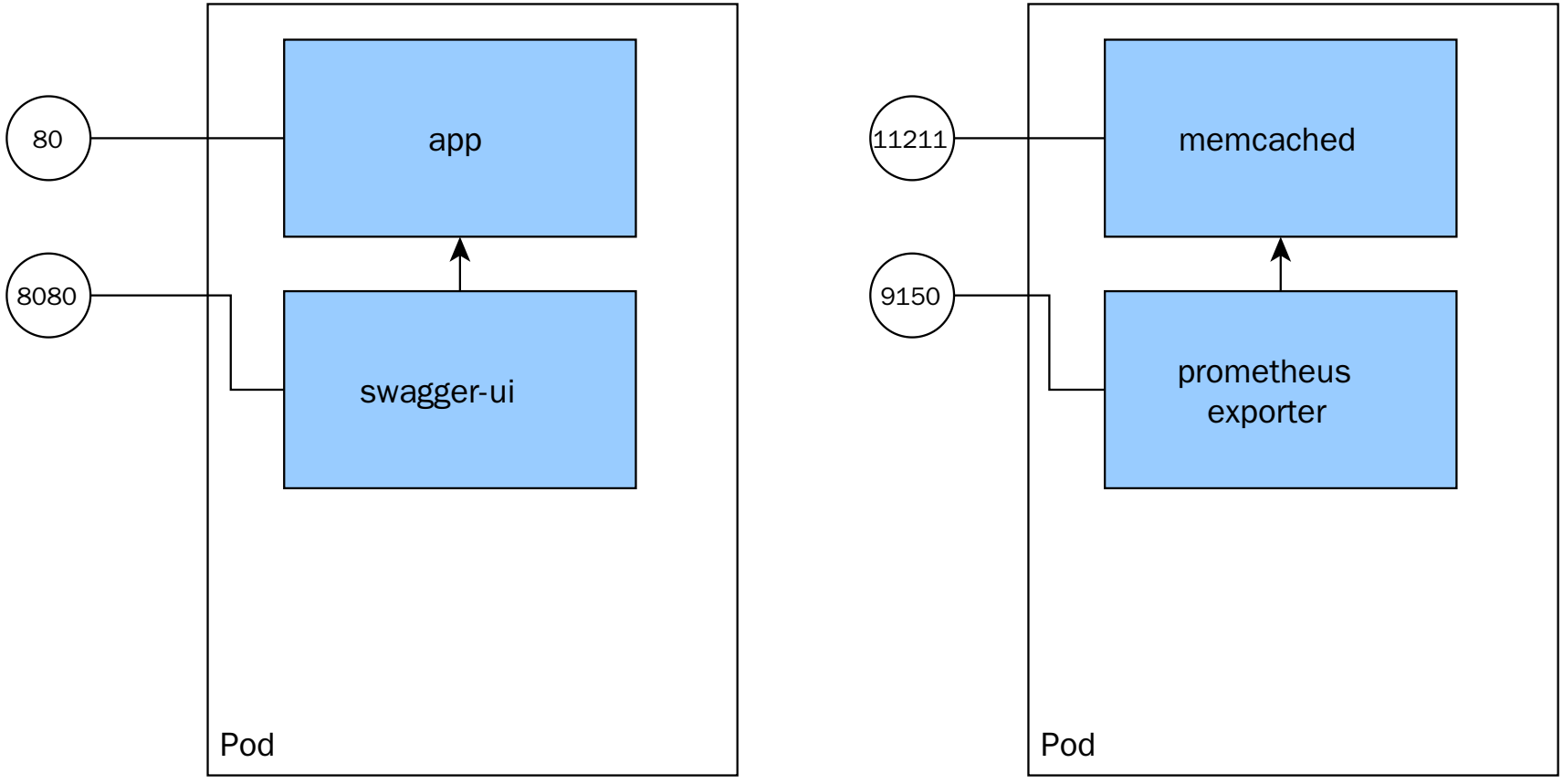
KUBERNETES CONCEPTS+

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