EFFECTIVE PLATFORM BUILDING WITH KUBERNETES.

IS K8S NEW LINUX?

Wojciech Barczynski - SMACC.io | Hypatos.ai
1 October 2018

WOJCIECH BARCZYŃSKI

- Lead Software Engineer & System Engineer
- Python i Golang
- Hobby: uczenie inżynierii oprogramowania

BACKGROUND

- Machine Learning FinTech
- Before:
  1 z 10 Indonesian mobile e-commerce
- Spent 3.5y datacenters z Openstack (0 ->21)
- SAP R&D

STORY

- Lyke - [12.2016 - 07.2017] - GKE
- SMACC - [10.2017 - present] - OnPrem + AKS

NIE LUBIE INFRA :D

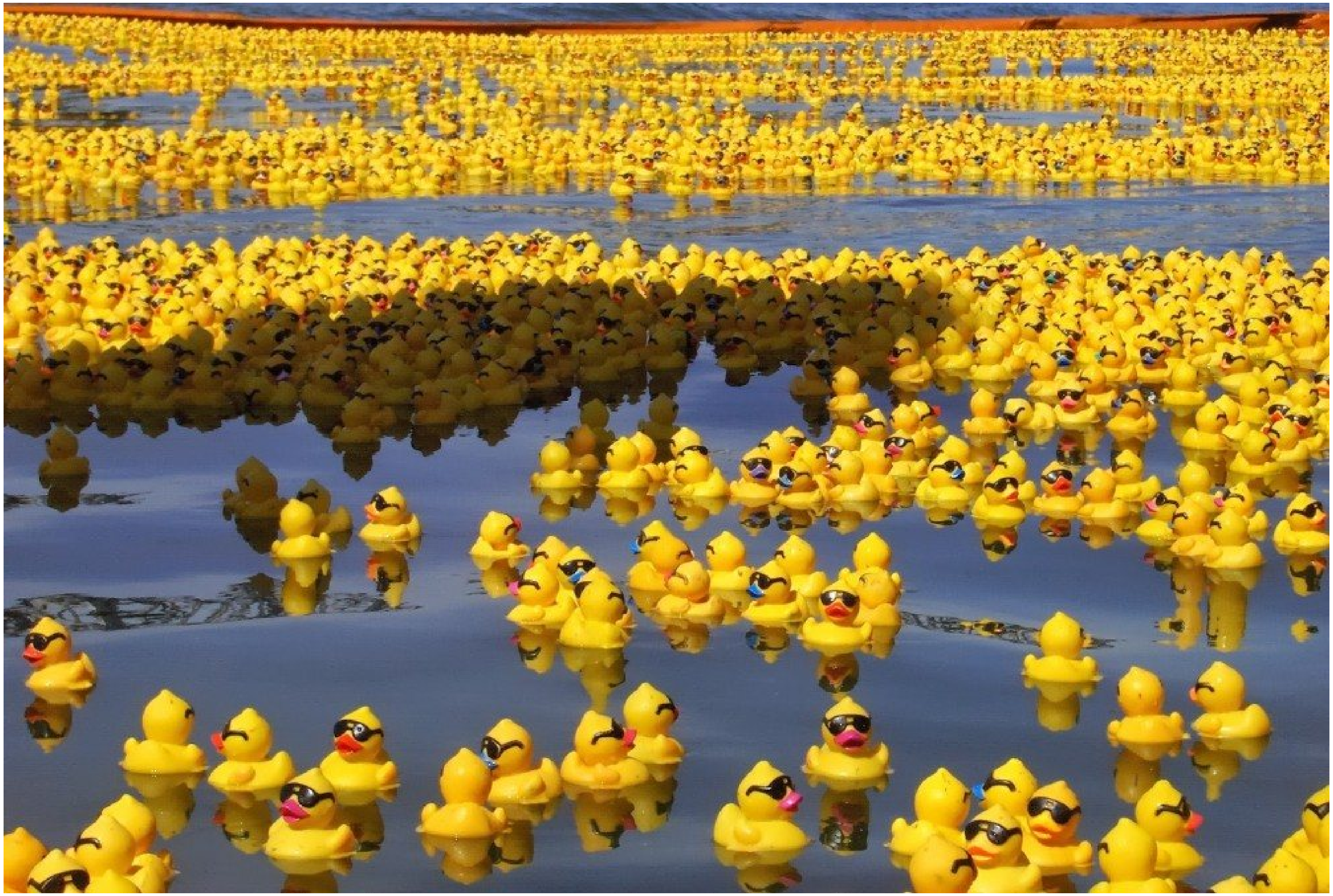
- Fabric, Ansible
- Puppet... Chef
- ...
- CloudFormations...

Terraform <3

NIE LUBIE INFRA :D

- Admistracja jest trudna i kosztowna
- Virtualne Maszyny, ansible, salt, etc.
- Za dużo ruchomych części
- Nie kończąca się standaryzacja

MIKROSERWISY AAA!



I...

- Cloud is not so cheap - $$$
- Cloud IaaS lock-in is real

A GDYBYŚ

- nie musiał myśleć o IaaS
- nigdy więcej logowania się na VM
- mniej dyskusji o CI / CD
- DC jako czarna skrzynka

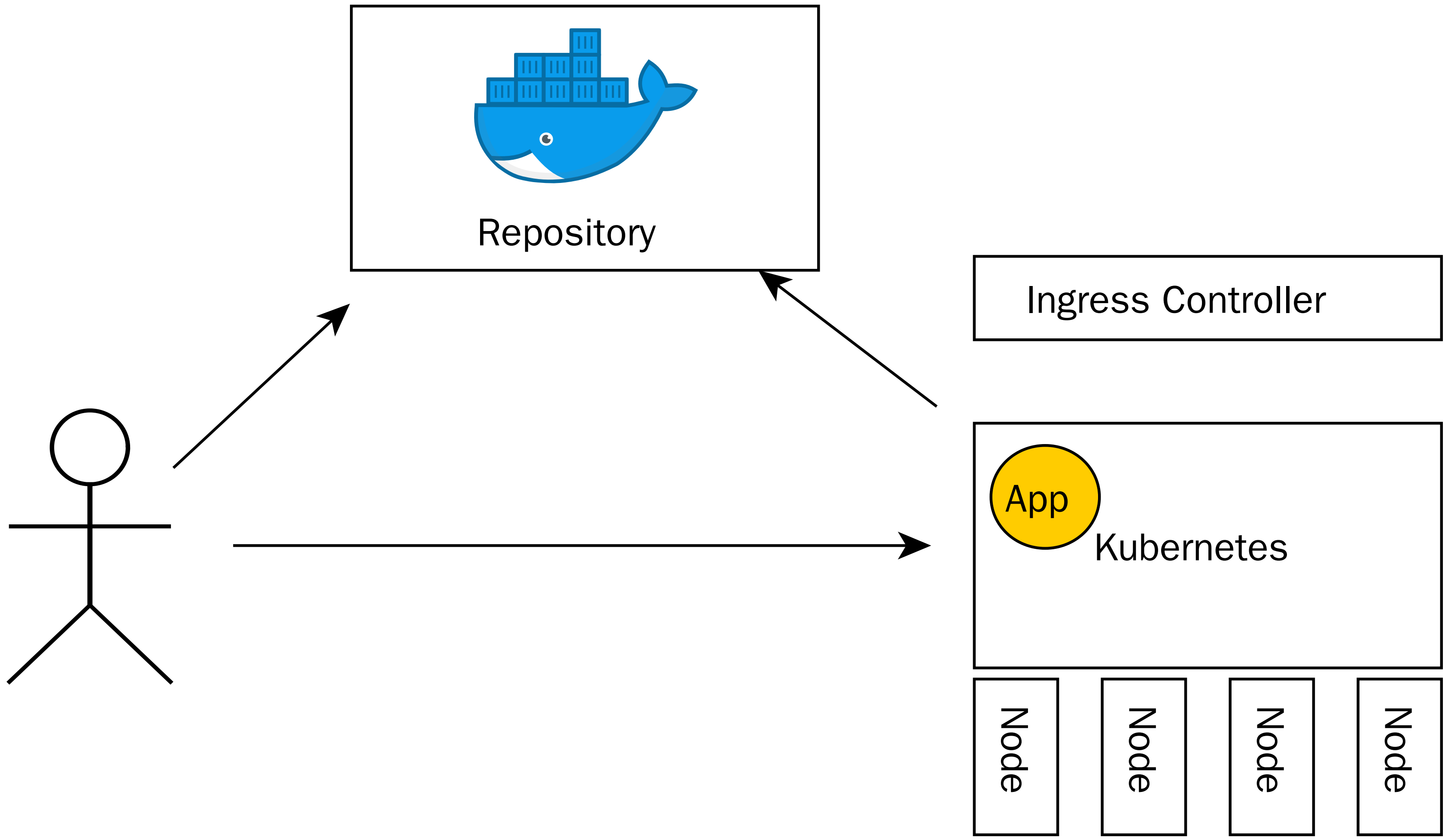
KUBERNETES

- Container management
- Service and application mindset
- Simple Semantic*
- Independent from IaaS provider

KUBERNETES

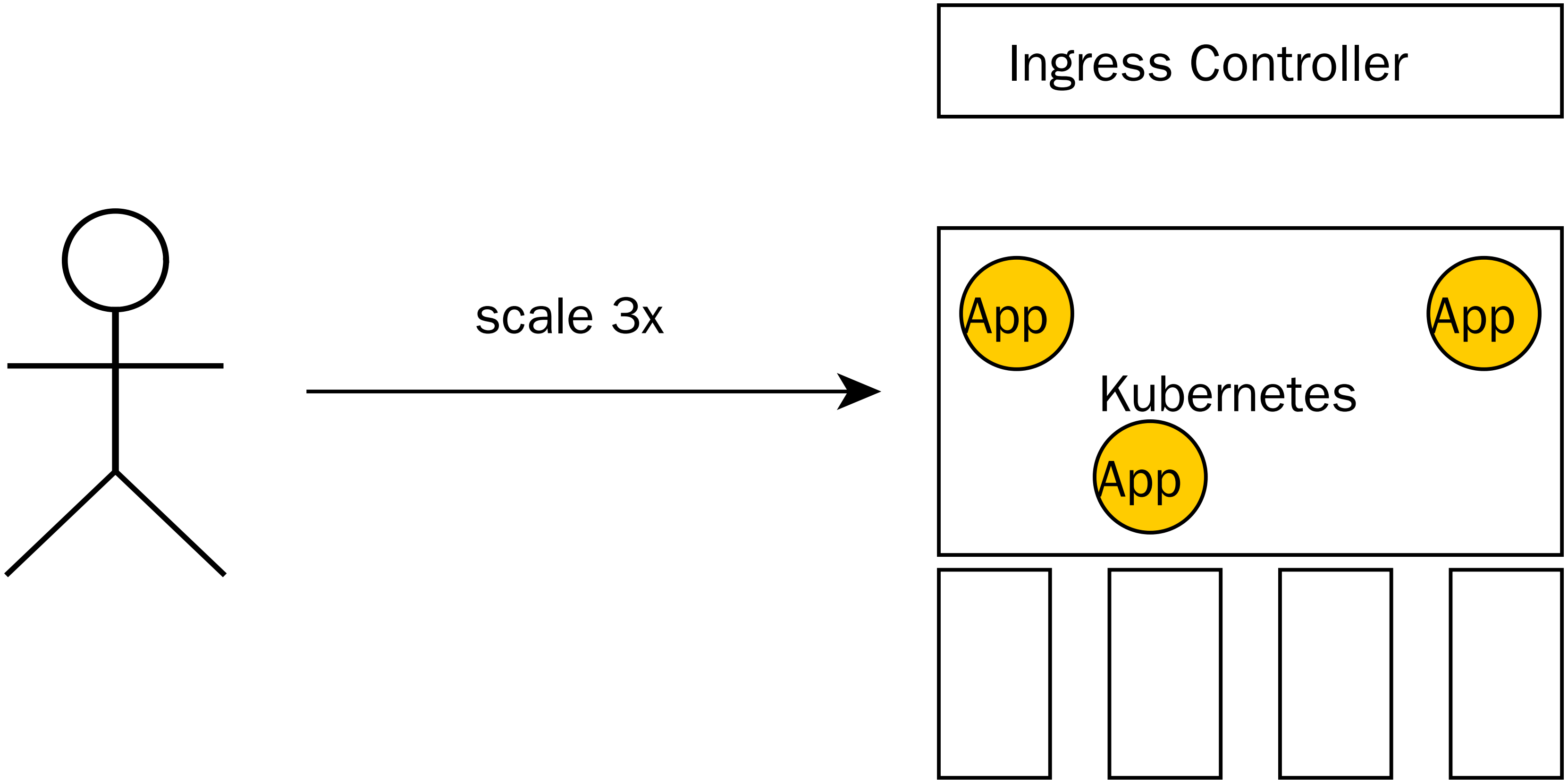
- Batteries for your 12factory apps
- Service discovery, meta-data support
- Utilize resources to nearly 100%

KUBERNETES



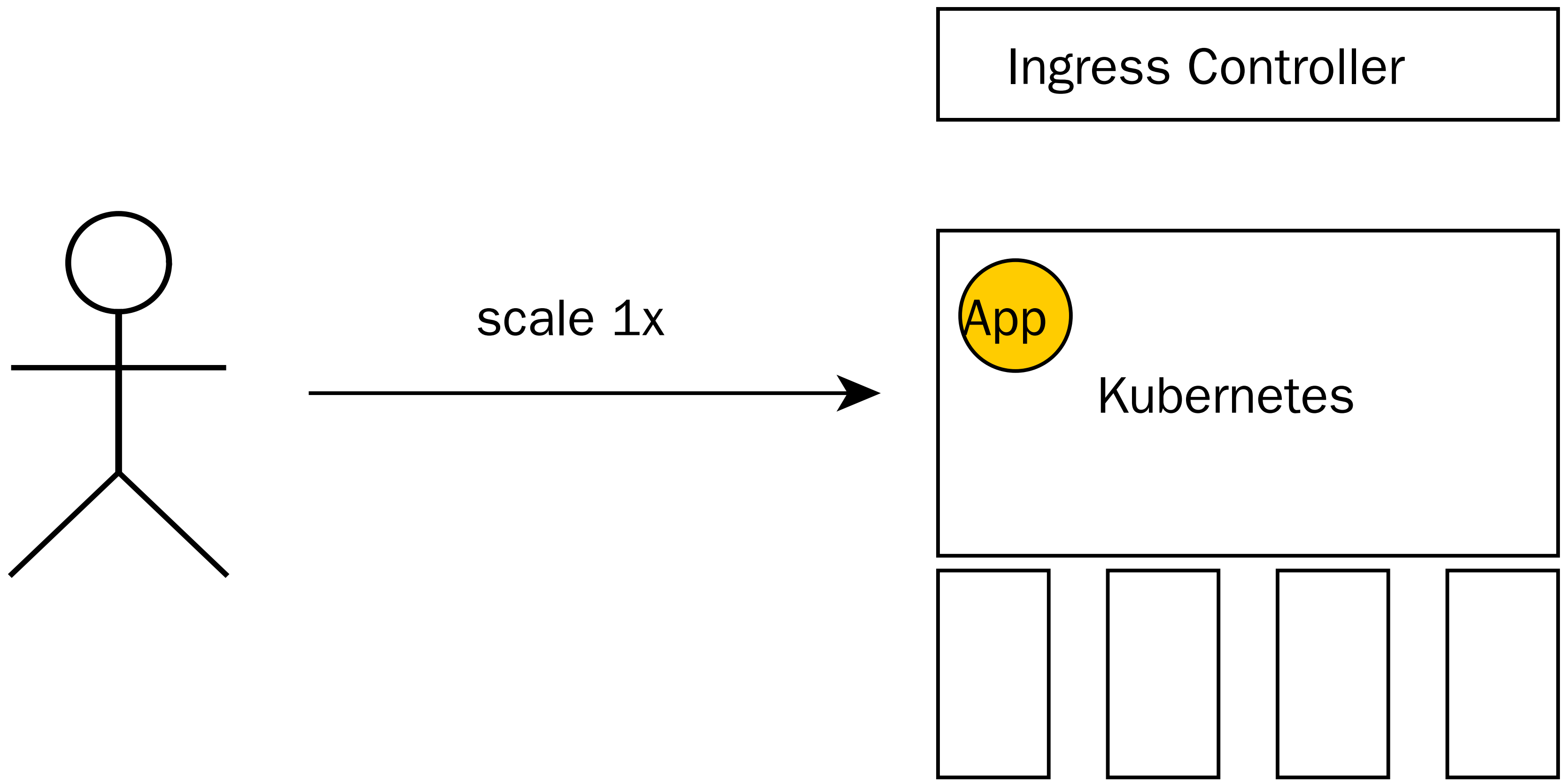
```
make docker_push; kubectl create -f app-srv-dpl.yaml
```

SCALE UP! SCALE DOWN!



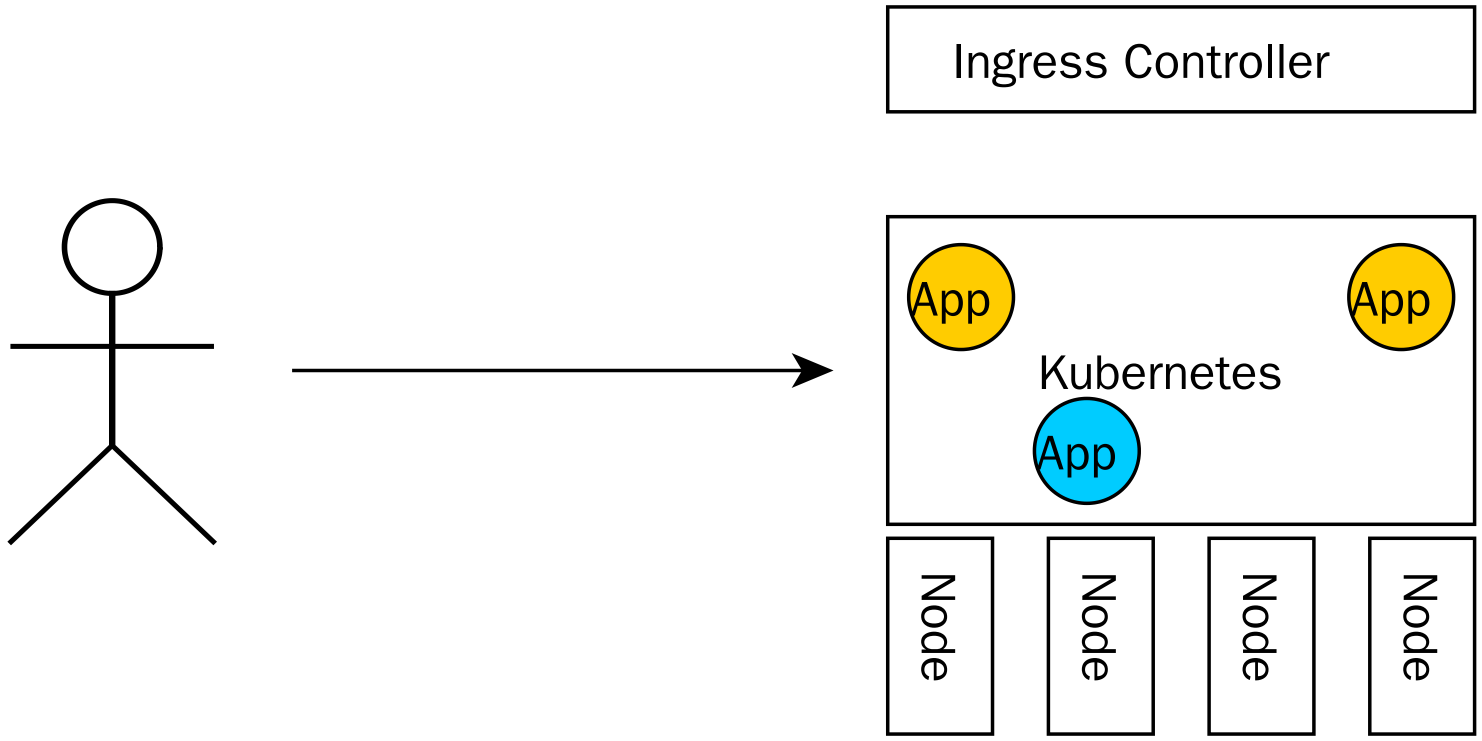
```
kubectl --replicas=3 -f app-srv-dpl.yaml
```

SCALE UP! SCALE DOWN!



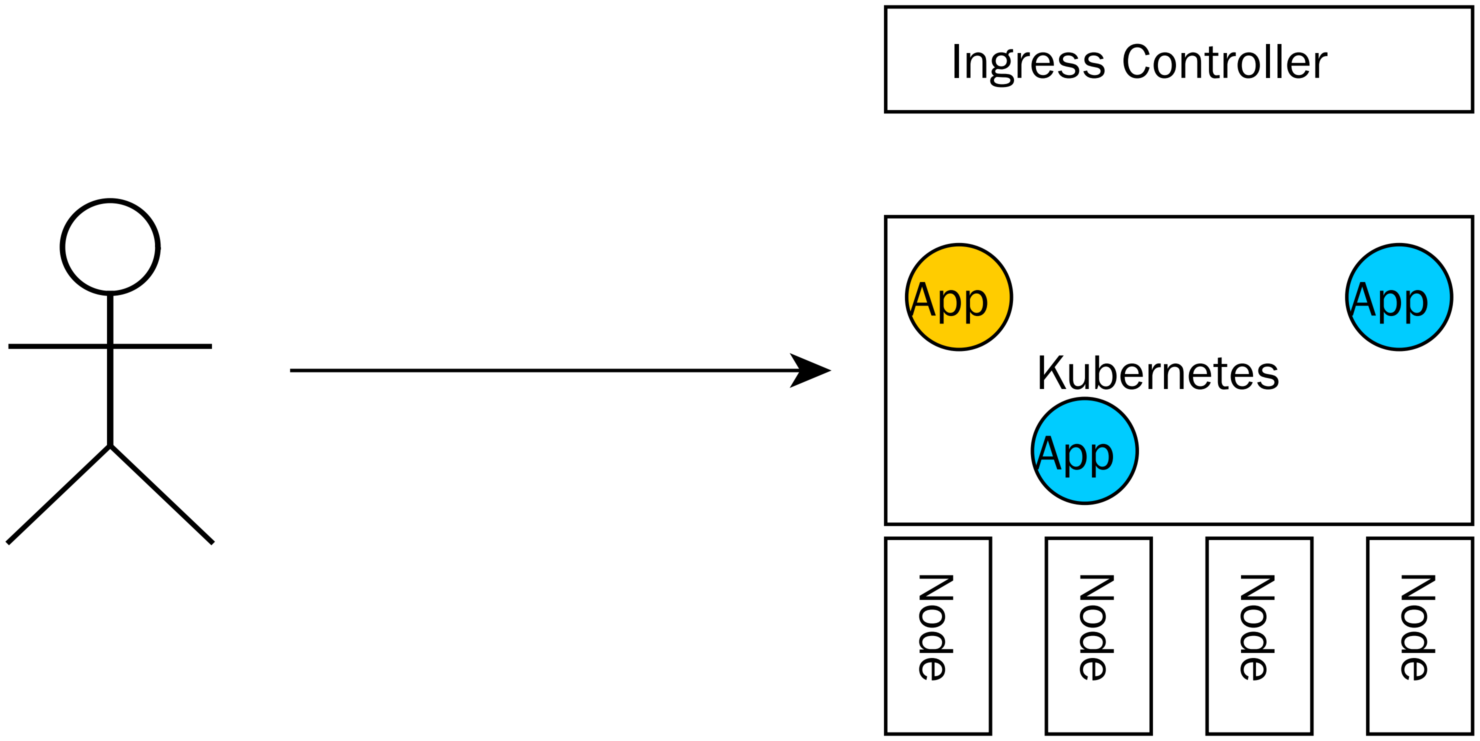
```
kubectl --replicas=1 -f app-srv-dpl.yaml
```

ROLLING UPDATES!



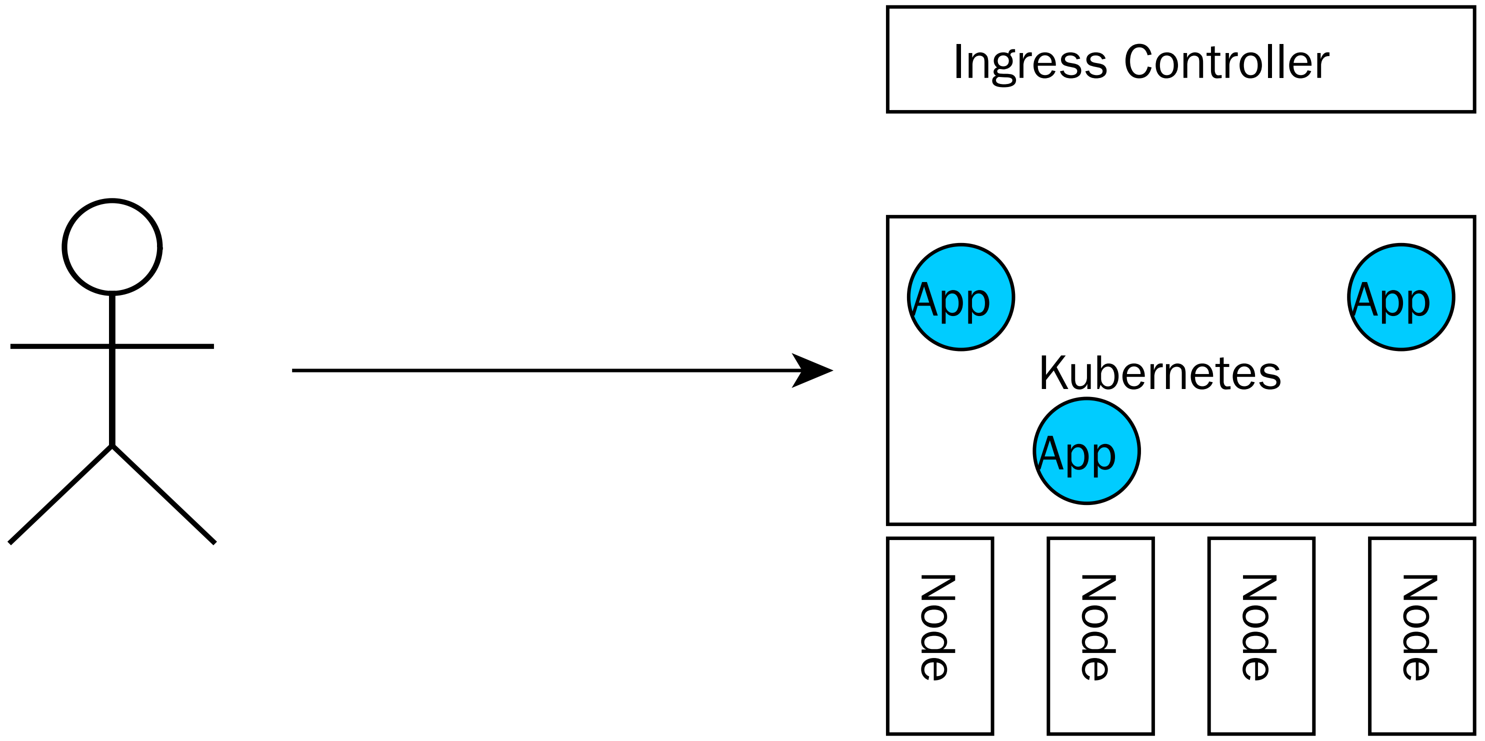
```
kubectl set image deployment/app app=app:v2.0.0
```

ROLLING UPDATES!



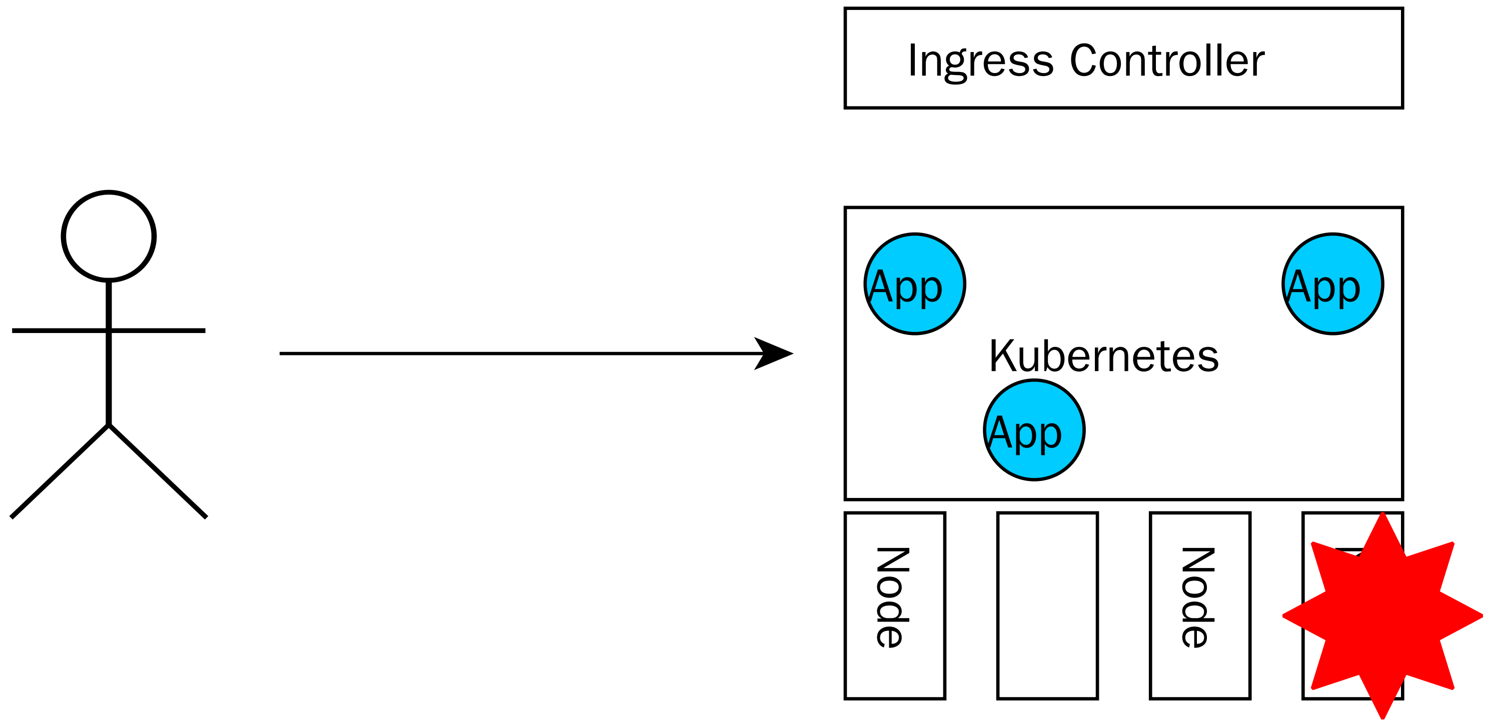
```
kubectl set image deployment/app app=app:v2.0.0
```

ROLLING UPDATES!

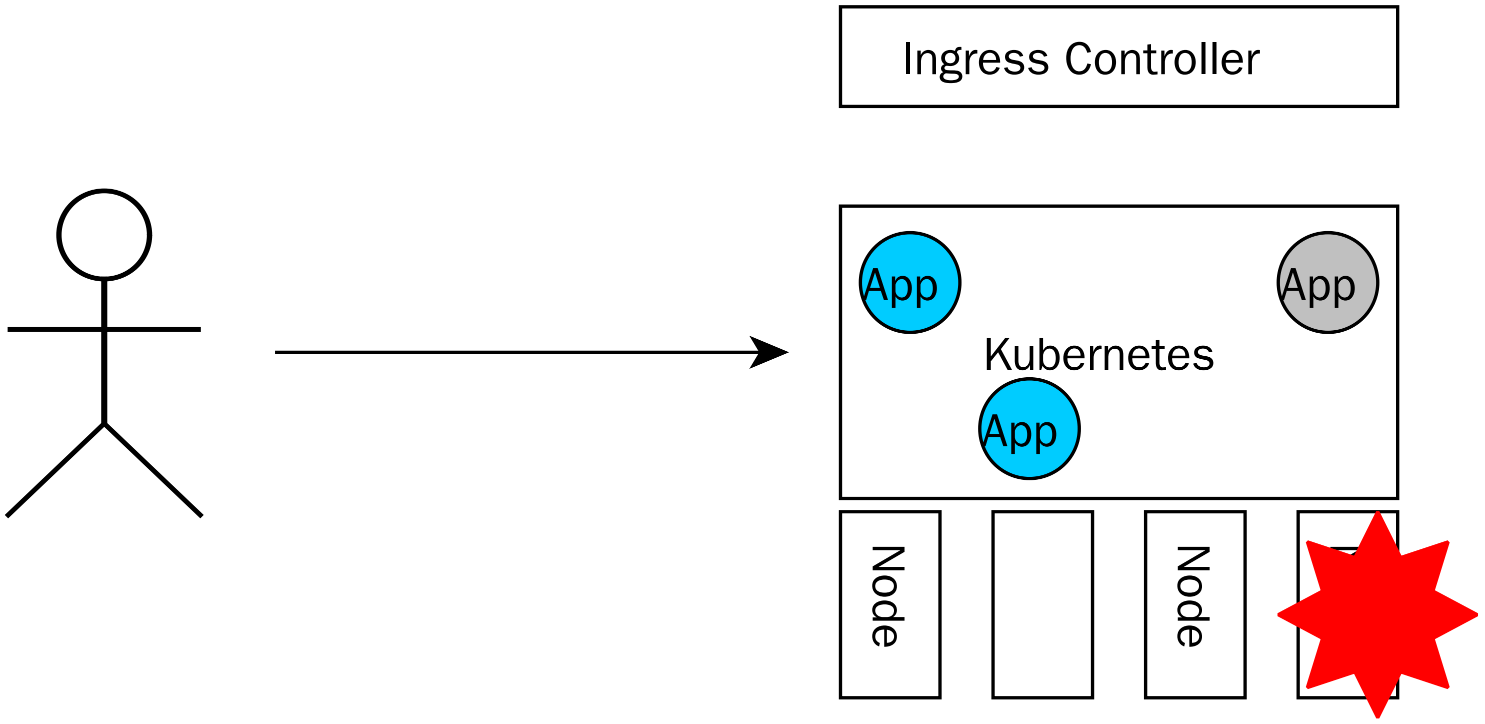


```
kubectl set image deployment/app app=app:v2.0.0
```

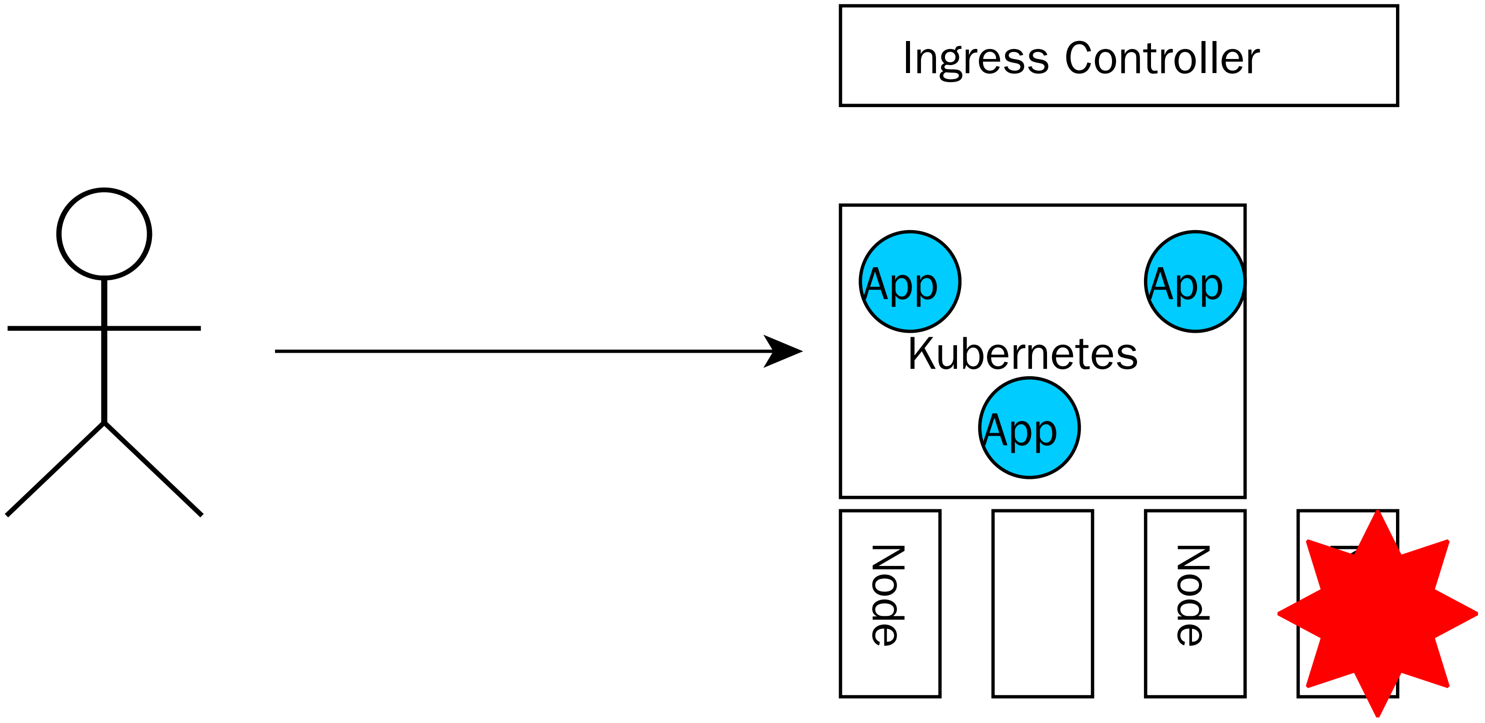
RESISTANCE!



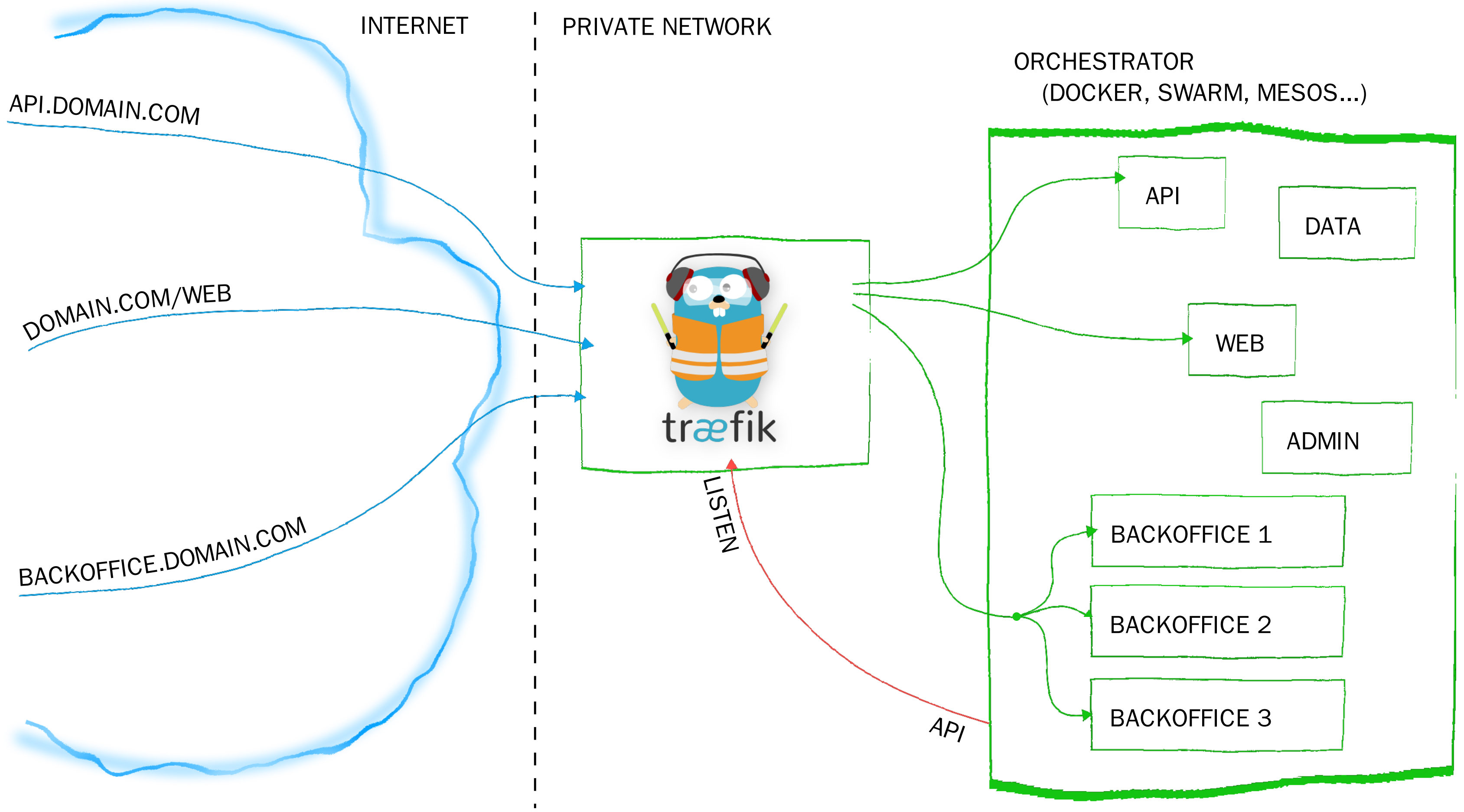
RESISTANCE!



RESISTANCE!



HOW GET USER REQUESTS?

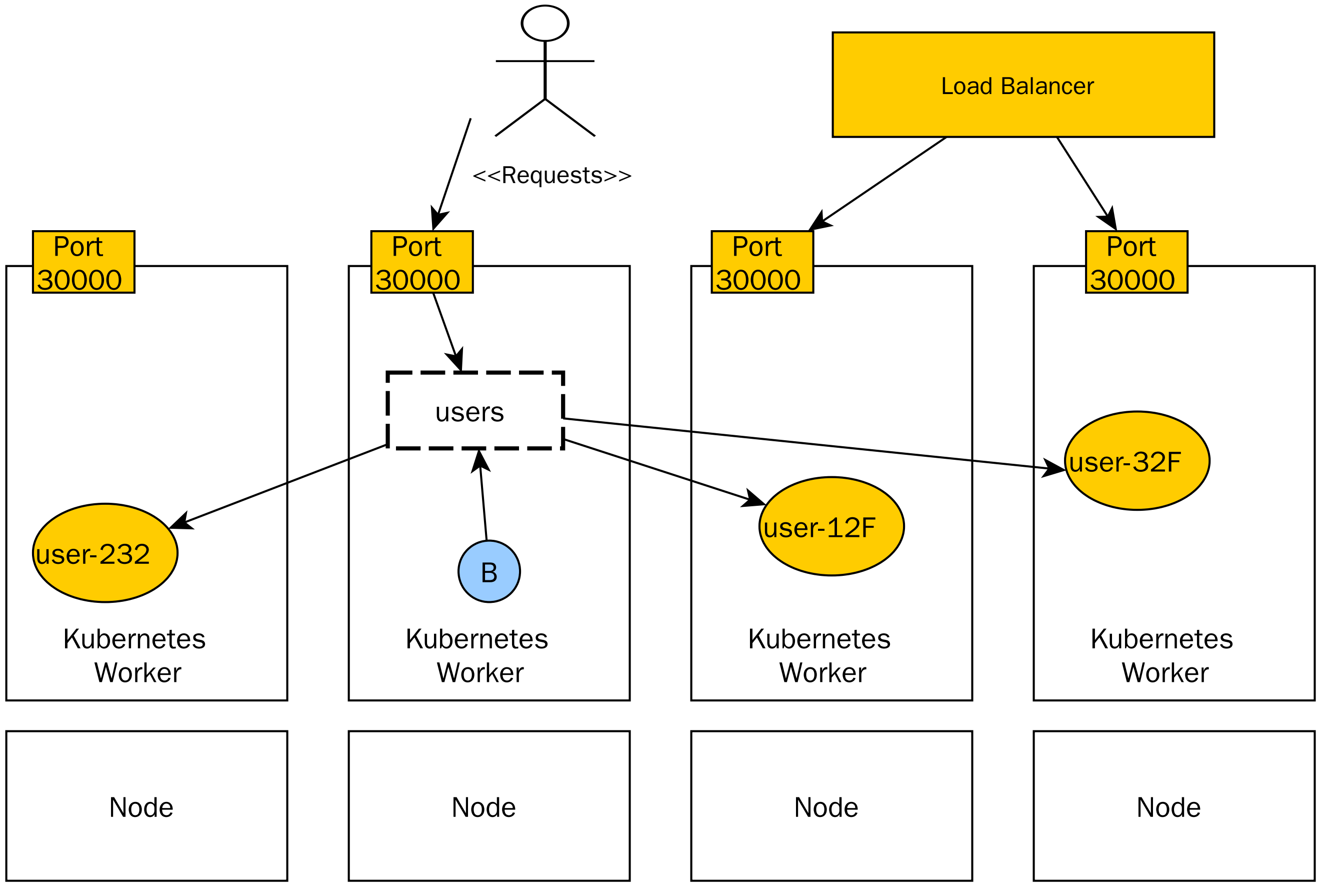


Ingress Controller

INGRESS

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

LOAD BALANCING



SERVICE DISCOVERY

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

AUTO-WIRING

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: memcached-0
  labels:
    app: memcached
    role: shard-0
    tier: backend
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/scheme: "http"
    prometheus.io/path: "metrics"
    prometheus.io/port: "9150"
spec:
```

https://github.com/skarab7/kubernetes-memcached

AUTO-WIRING

```
groups:
- name: apps
  rules:
  - alert: ProductionInstanceDown
    expr: up{env = 'production'}  == 0
    for: 2m
    labels:
      severity: opsgenie
    annotations:
      summary: "Instance {{ $labels.instance }} of {{ $labels.
      description: "{{ $labels.instance }} of job {{ $labels.j
```

DROP-IN

- traefik / Ingress / Envoy
- prometheus
- audit checks
- ...

MORE

- readiness probe
- liveness probe
- resource quotas

ALL IN GIT

- all in yaml
- integration with monitoring, alarming
- integration with ingress-controller
- ...
- devs can forget about infrastructure... almost

DevOps Culture Dream!

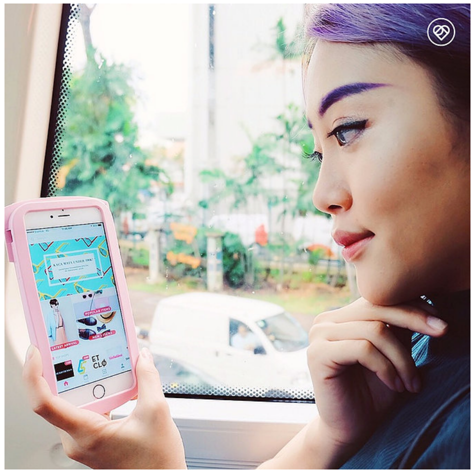
I..

- łatwo zaimplementować Continuous Deployment
- i ukryć detale

LYKE

LYKE

- Mobile E-commerce
- 50k+ użytkowników
- 2M downloads
- Top 10 Fashion Apps



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

Teraz: JollyChic Indonesia

GOOD PARTS

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

CHALLENGES

- 50+ VMs in Amazon
- 1 VM - 1 App, idle machine
- Puppet, hilarious (manual) deployment process
- Fear
- Forgotten components
- Performance issues

APPROACH

1. Simplify infrastructure
2. Change the Development practices
3. Change the work organization

see: Conway's law

SIMPLIFY

1. Kubernetes with Google Kubernetes Engine
2. Terraform for all new

SIMPLIFY

1. Monitoring: Prometheus + Grafana
2. Loging: Elasticsearch-Fluentd-Kibana
3. Google Identity-Aware-Proxy to protect tools
4. 3rd party SaaS: statuscake and opsgenie

CONTINUOUS DEPLOYMENT

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

TRAVISCI

1. Tests
2. Build docker
3. Deploy to Google Container Registry
4. Deploy to k8s only new docker
5. no config applied

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

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

- Puppet … ➡ Dockerfile
- Check the 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
- Use: `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

5. CONTINUOUS DEPLOYMENT

Travis:

- use the same Makefile as a developer

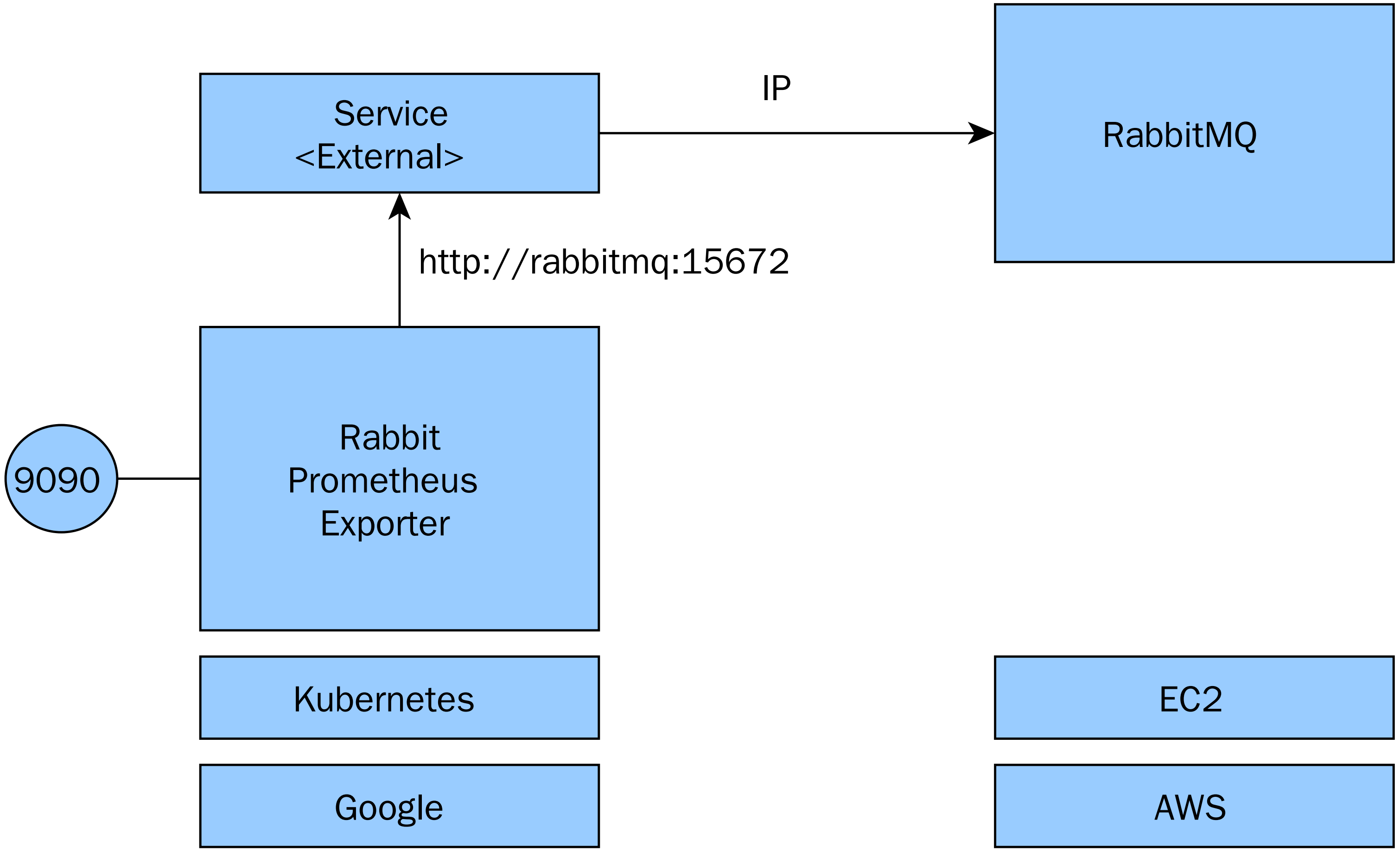
6. KEEP IT RUNNING

Bridge the new with old:

- Use external services in Kubernetes
- Optional: Expose k8s in the Legacy [1]

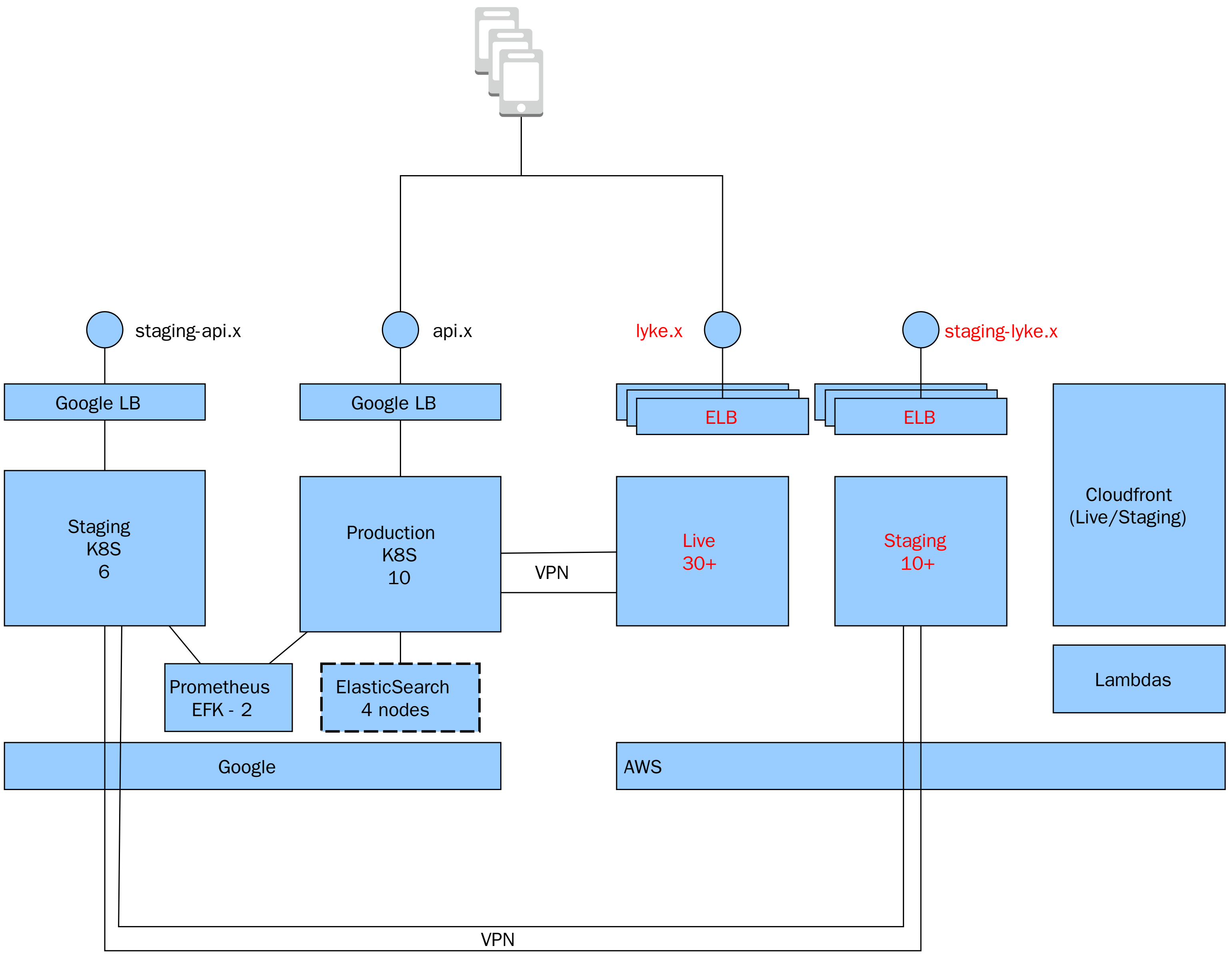
[1] feeding K8S events to HashiCorp consul

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. MOVE TO MICRO-SERVICES

To offload the biggest components:

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

9. SERVICE SELF-CONSCIOUSNESS

Add to old services:

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

10. GET PERFORMANCE TESTING

- introduce *wrk* for evaluating performance
- load test the real system

WHAT WORKED

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

WHAT DID NOT WORK

1. Too many PoC ➡ 2 weeks max
2. Do it with smaller chunks
3. Alert rules too hard to write
4. Push-back to k8s yaml [*]

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

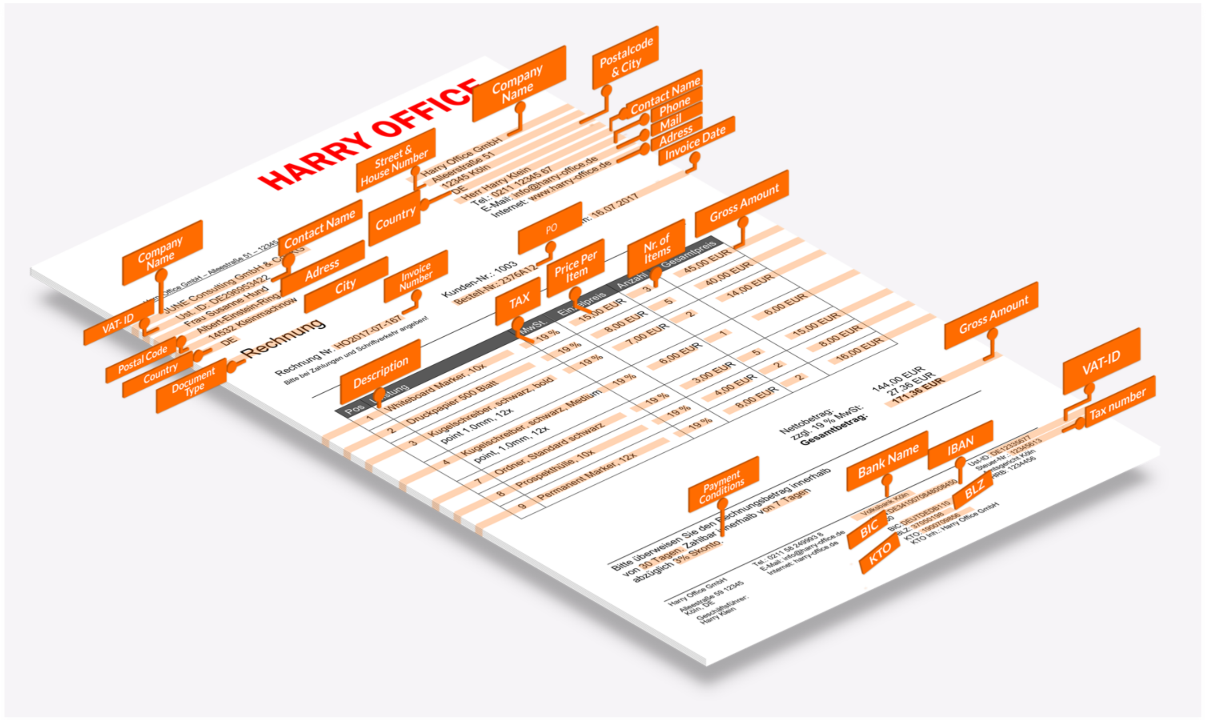
DO DIFFERENT

1. Move dev and staging data immediately
2. Let devs know it is a transition stage
3. Teach earlier about quotas
4. EFK could wait
5. Big migration - a paid-XXX% weekend

SMACC

Hypatos

SMACC.io



SAP

Microsoft Dynamics

DATEV

SMACC

Klienci

Deutsche Bank

Deloitte.

EY

AOK
Die Gesundheitskasse.

HelloFresh

accenture

BMB
巴三巴
GRUPPE

McKinsey&Company

SMACC

STORY

- Legacy on AWS
- Experiments with AWS ECS :/
- Self-hosted K8S on ProfitBricks
- Get to Microsoft ScaleUp, welcome Azure!

LUCKILY - AKS

Azure ❤

Nie było, aż tak różowo ☠

AZURE KUBERNETES SERVICE

- Niezależne od IaaS
- Our OnPrem = Our OnCloud
- Konsolidacja naszych mikroserwisów
- Plug & play, e.g., monitoring

PROSTOTA

- az aks CLI do ustawienia k8s - README.rst
- Terraform dla wszystkiego wokól
- 1Password i gopass.pw

TF dla AWS

RÓŻNICA ☠

- Dwa zespoły: Berlin i Warszawa
- Ja w Warszawie

NOWE DOŚWIADCZENIE

- devs really do not like TravisCI ... k8s yamls
- ciężkie przejście z ProfitBricks do AKS

ROZWIĄZANIE

- prościej :)
- copy&paste dla wszystkiego
- ukryć deployment na k8s, usunąć magic
- deployment on *tag*

Similar to the Kelsey Hightower approach

Repo .travis.yml

```
language: go
go:
- '1.10'
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
```

Makefile

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

CONTINUOUS DEPLOYMENT

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

PROCESS

1. `git tag` and push
2. Generate deploy, ingress, and svc kubernetes files
3. Merge PR -> produciton

KUBERNETES

- Pure, generated, kubernetes config
- 2x kubernetes operators

NEXT

- Acceptance tests
- Scale our Machine Learnings
- Deployment tool based on missy
- Keeping an eye on Istio

Kubernetes - Linux

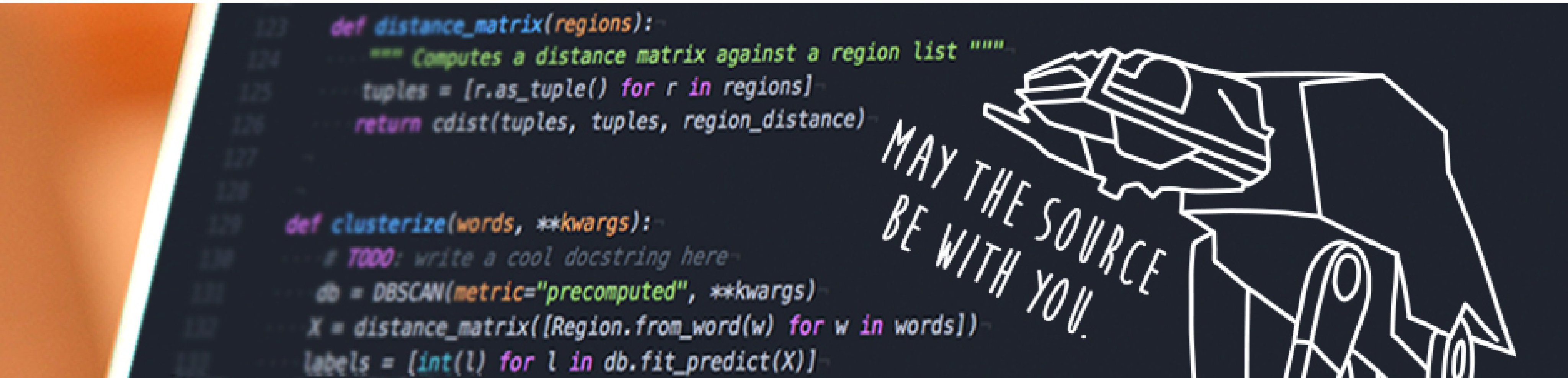
- Not a silver bullet, but damn close
- Common runtime dla onPrem i onCloud
- Z kubevirt - zastąpić może Openstack

Kubernetes - Linux

- The biggest asset - the API
- With service discovery - an integration platform

DZIĘKUJĘ. PYTANIA?

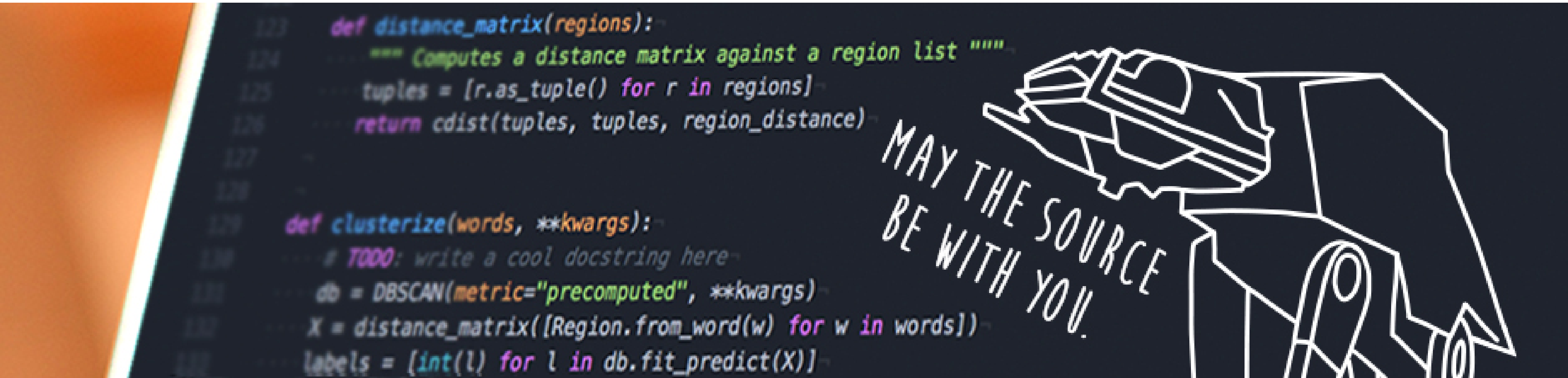
ps. We are hiring.



```
def distance_matrix(regions):
    """ Computes a distance matrix against a region list """
    tuples = [r.as_tuple() for r in regions]
    return cdist(tuples, tuples, region_distance)


def clusterize(words, **kwargs):
    # TODO: write a cool docstring here
    db = DBSCAN(metric="precomputed", **kwargs)
    X = distance_matrix([Region.from_word(w) for w in words])
    labels = [int(l) for l in db.fit_predict(X)]
```

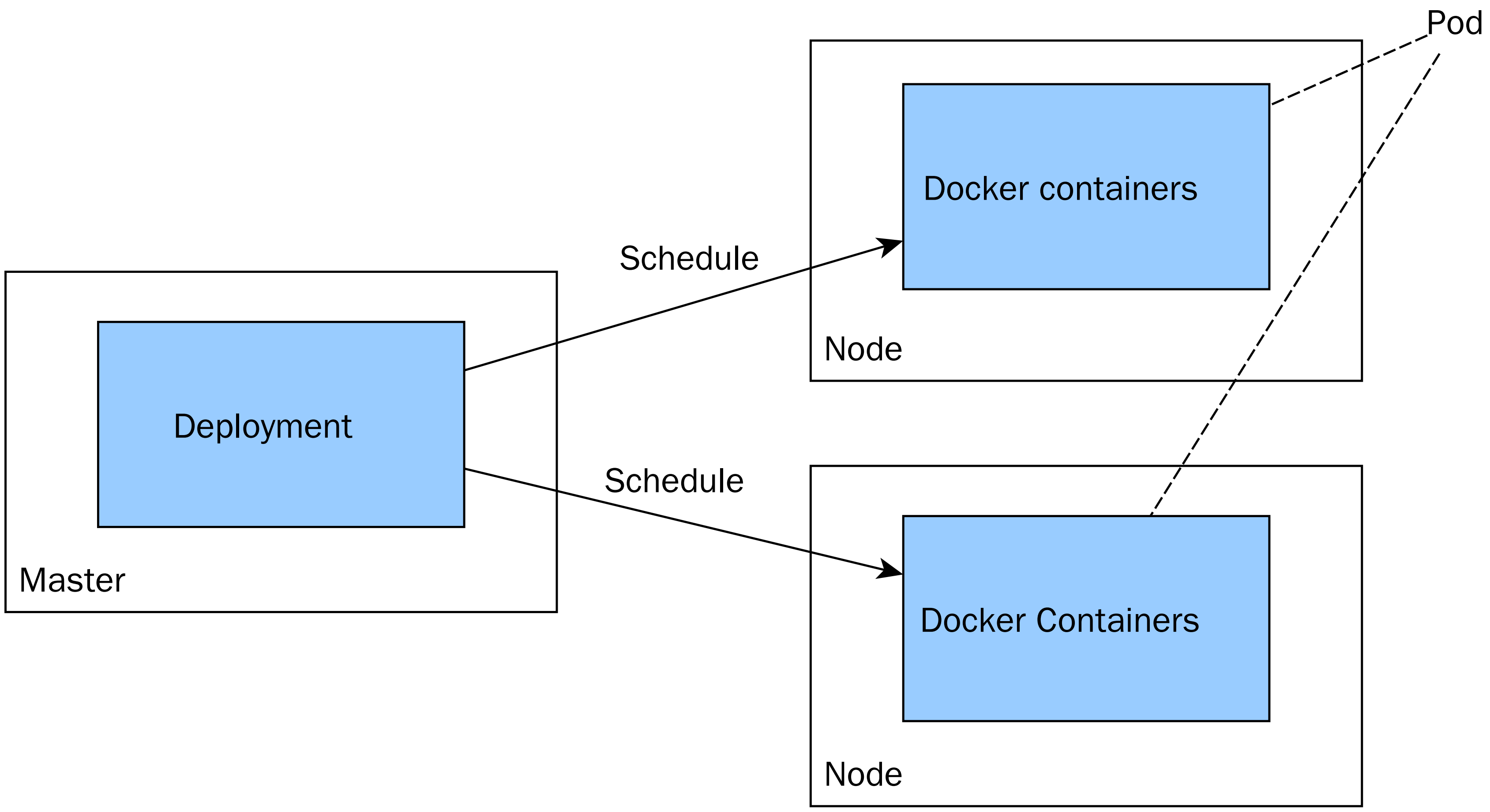
BACKUP SLIDES

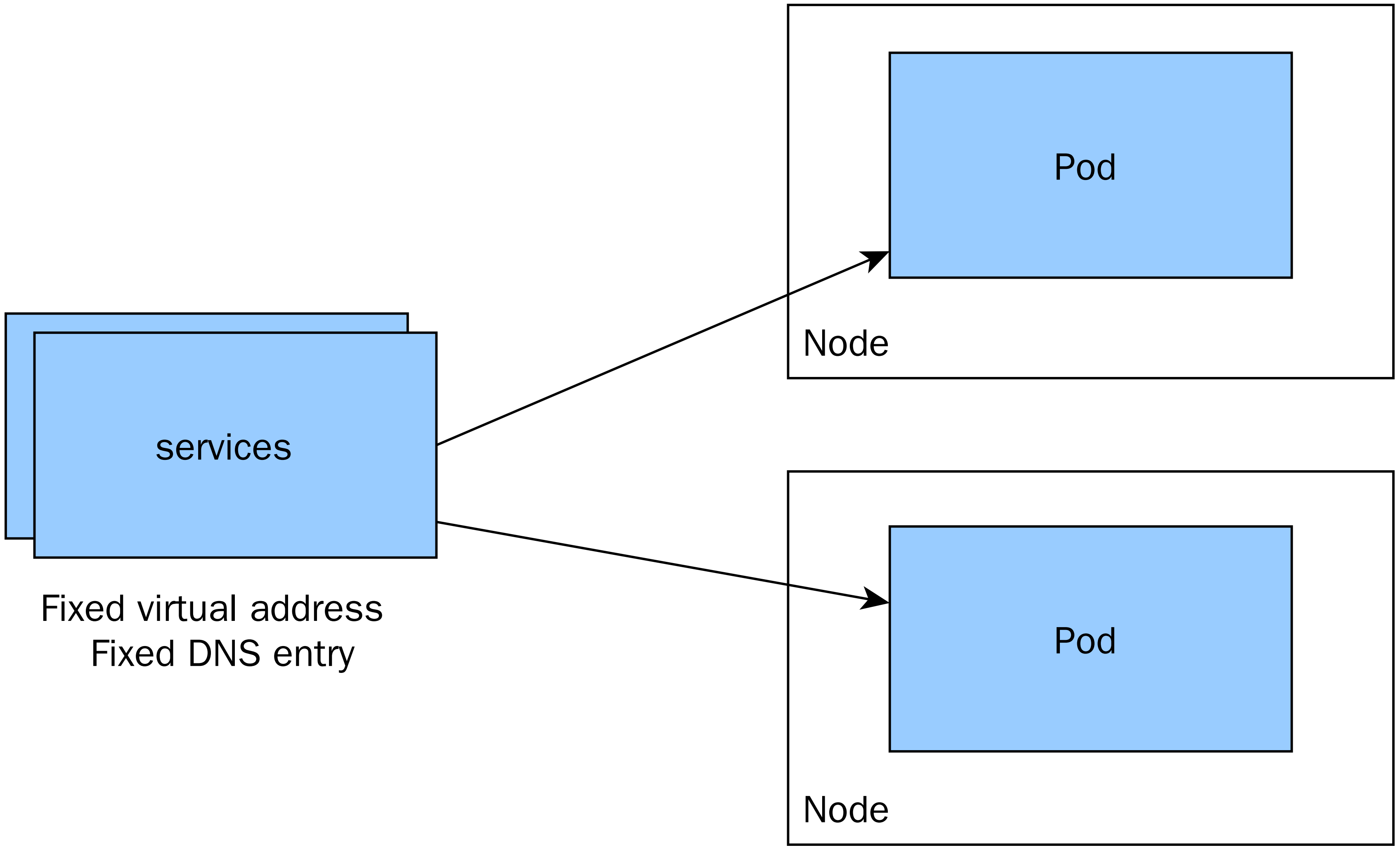


```
def distance_matrix(regions):
    """ Computes a distance matrix against a region list """
    tuples = [r.as_tuple() for r in regions]
    return cdist(tuples, tuples, region_distance)


def clusterize(words, **kwargs):
    # TODO: write a cool docstring here
    db = DBSCAN(metric="precomputed", **kwargs)
    X = distance_matrix([Region.from_word(w) for w in words])
    labels = [int(l) for l in db.fit_predict(X)]
```

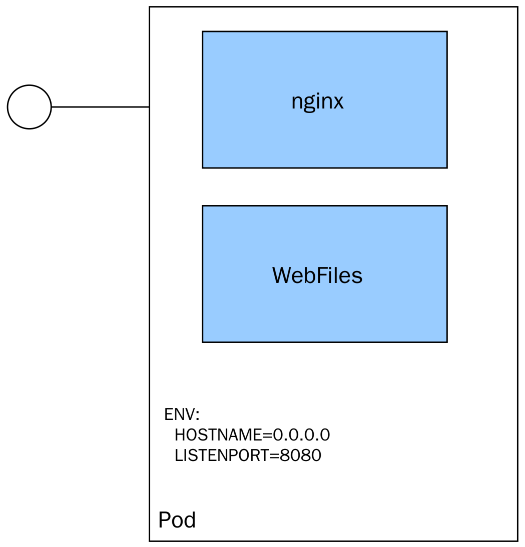
KUBERNETES CONCEPTS


Pod
Docker containers
Node
Schedule
Deployment
Master
Schedule
Docker Containers
Node

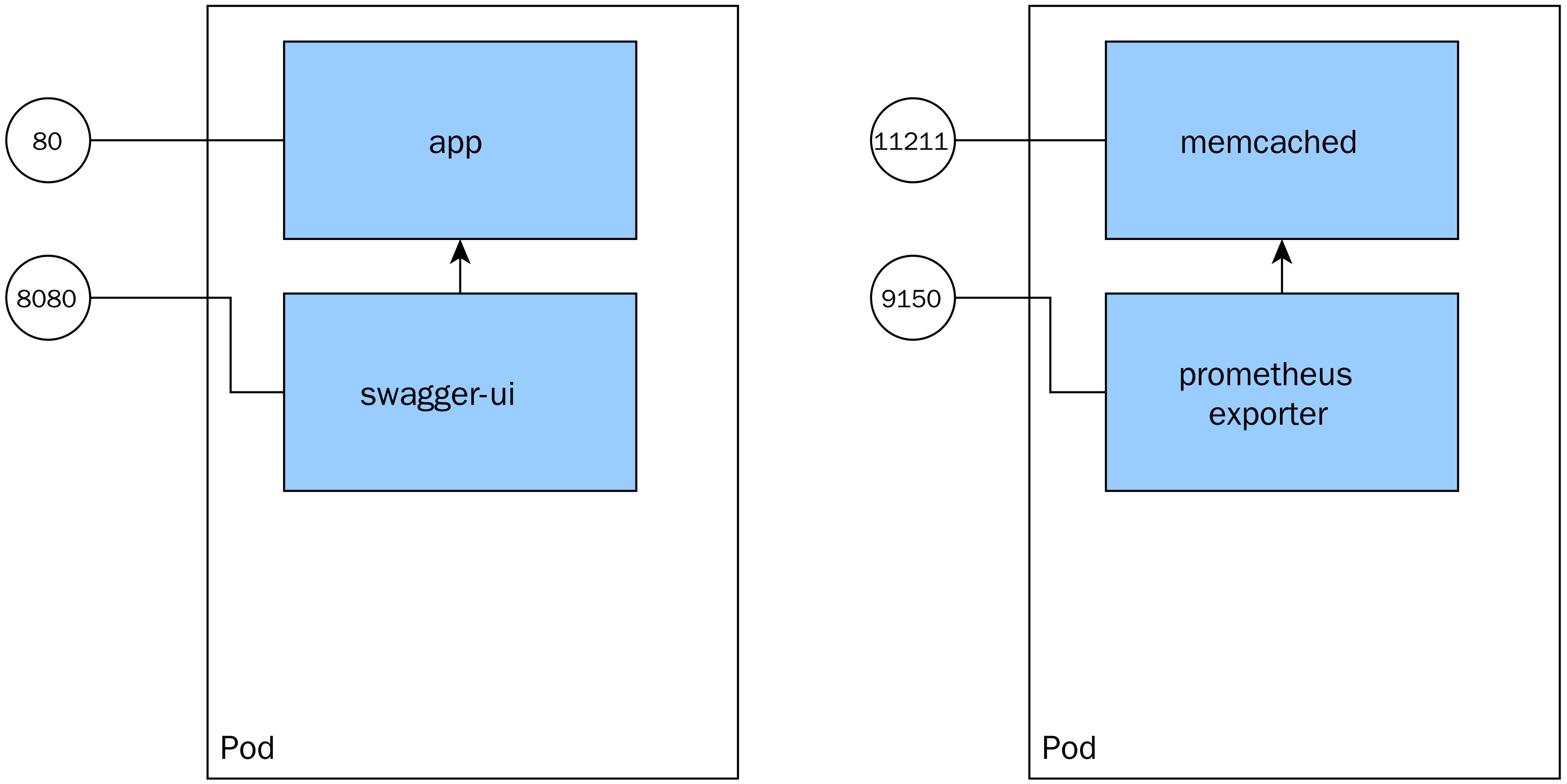

services
Fixed virtual address
Fixed DNS entry
Pod
Node
Pod
Node

PODS

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



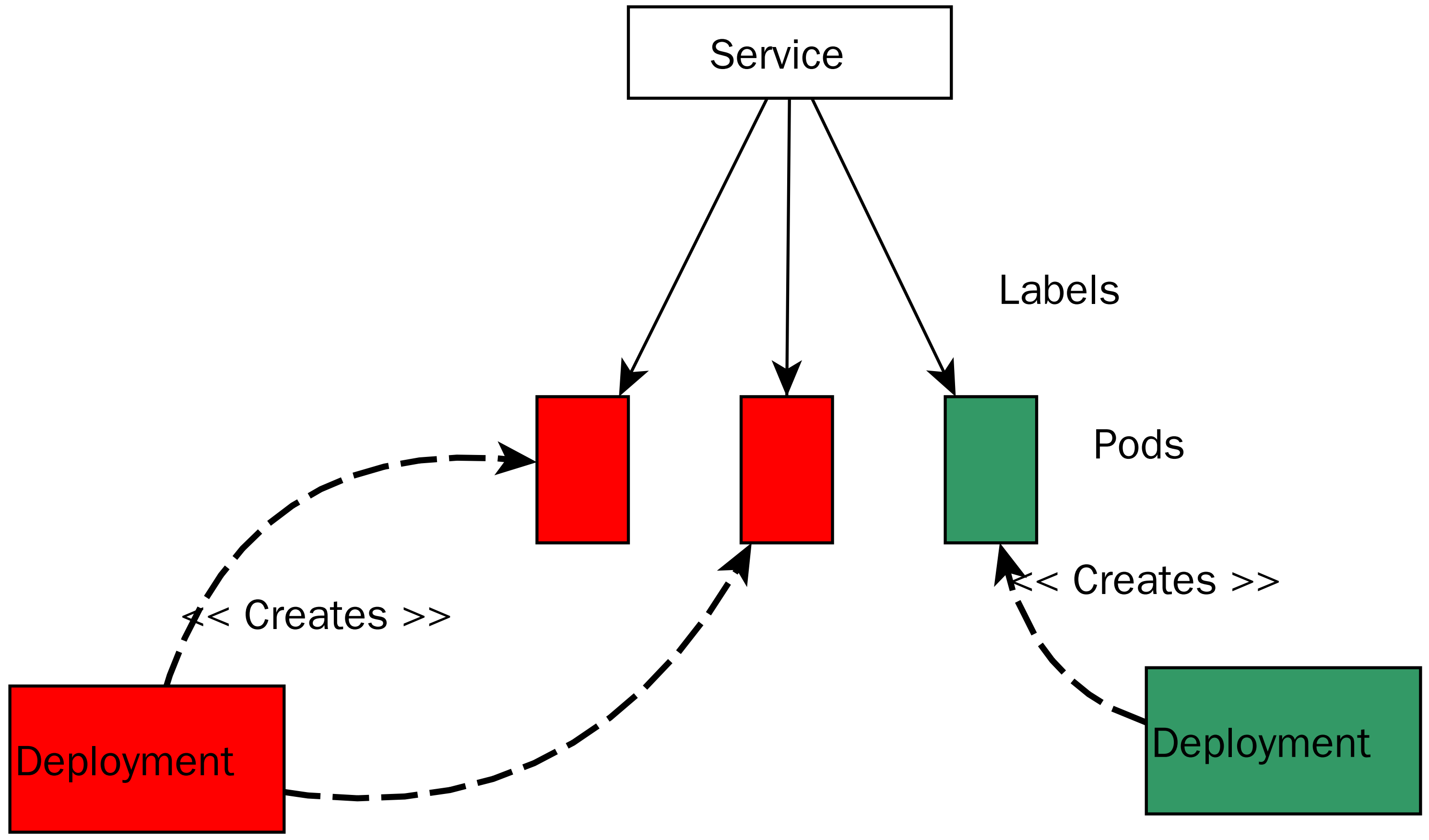
SIDE-CARS


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

BASIC CONCEPTS

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

ROLLING RELEASE WITH DEPLOYMENTS



Also possible

KUBERNETES CONCEPTS

MACHINE LEARNING WORKLOADS

- Rediness probe
- Liveness probe

K8S CLIENTS

- golang
- python