EFFECTIVE PLATFORM BUILDING WITH KUBERNETES.

IS K8S NEW LINUX?

Wojciech Barczynski - SMACC.io | Hypatos.ai
Wrzesień 2018

WOJCIECH BARCZYŃSKI

- Lead Software Engineer & System Engineer
- Interests: working software
- Hobby: teaching software engineering

SMACC

Hypatos

BACKGROUND

- ML FinTech ➡ micro-services and k8s
- Before:
  1 z 10 Indonesian mobile e-commerce (Rocket Internet)
- Spent 3.5y with Openstack, 1000+ nodes, 21 data centers
- I do not like INFRA :D

STORY

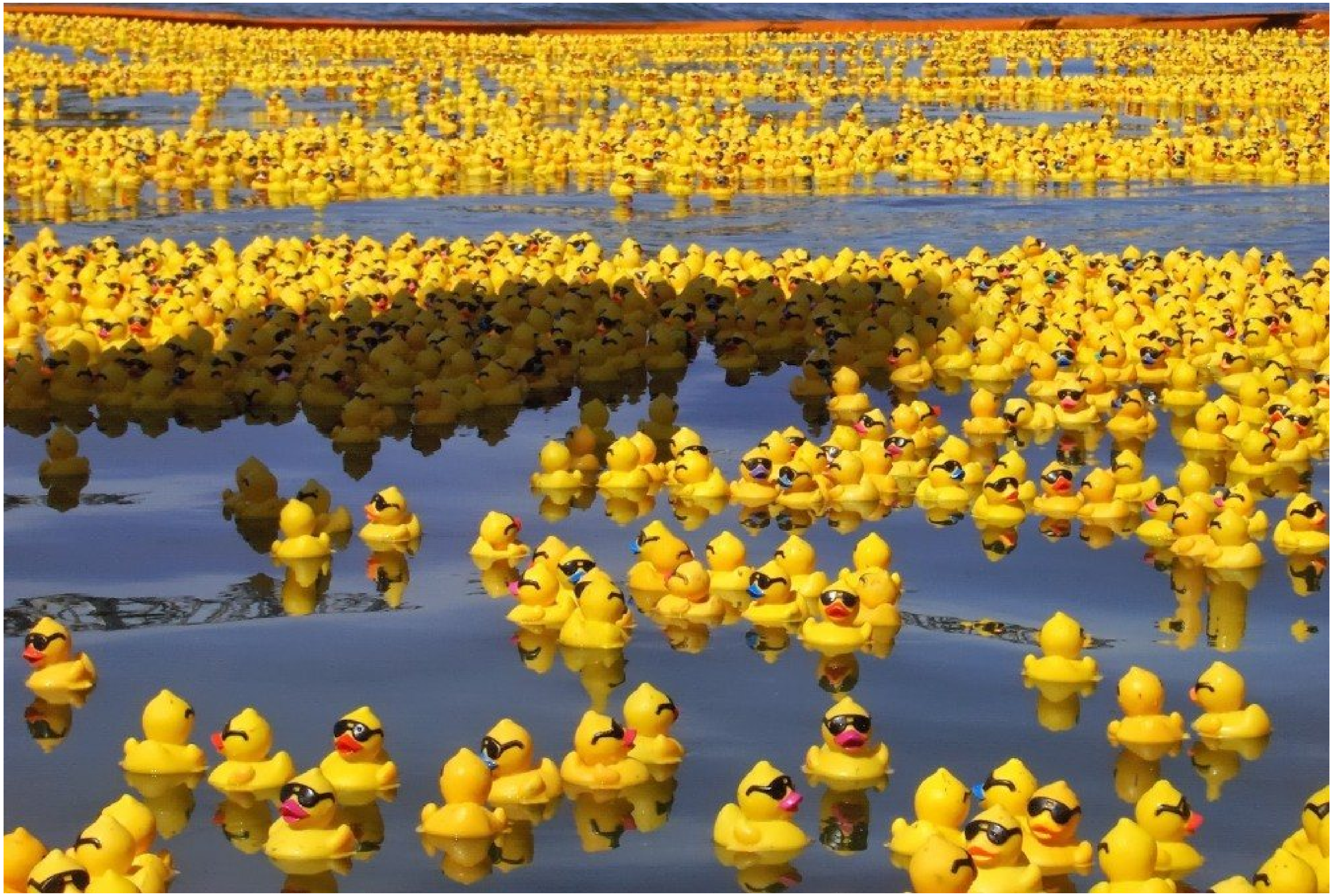
- Lyke - [12.2016 - 07.2017]
- SMACC - [10.2017 - present]

KUBERNETES

WHY?

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

MIKROSERWISY AAA!



WHY?

- Cloud is not so cheap - $$$

IMAGINE

- do not need to think about IaaS
- no login on a VM
- less gold plating your CI / CD ...
- DC as a black box

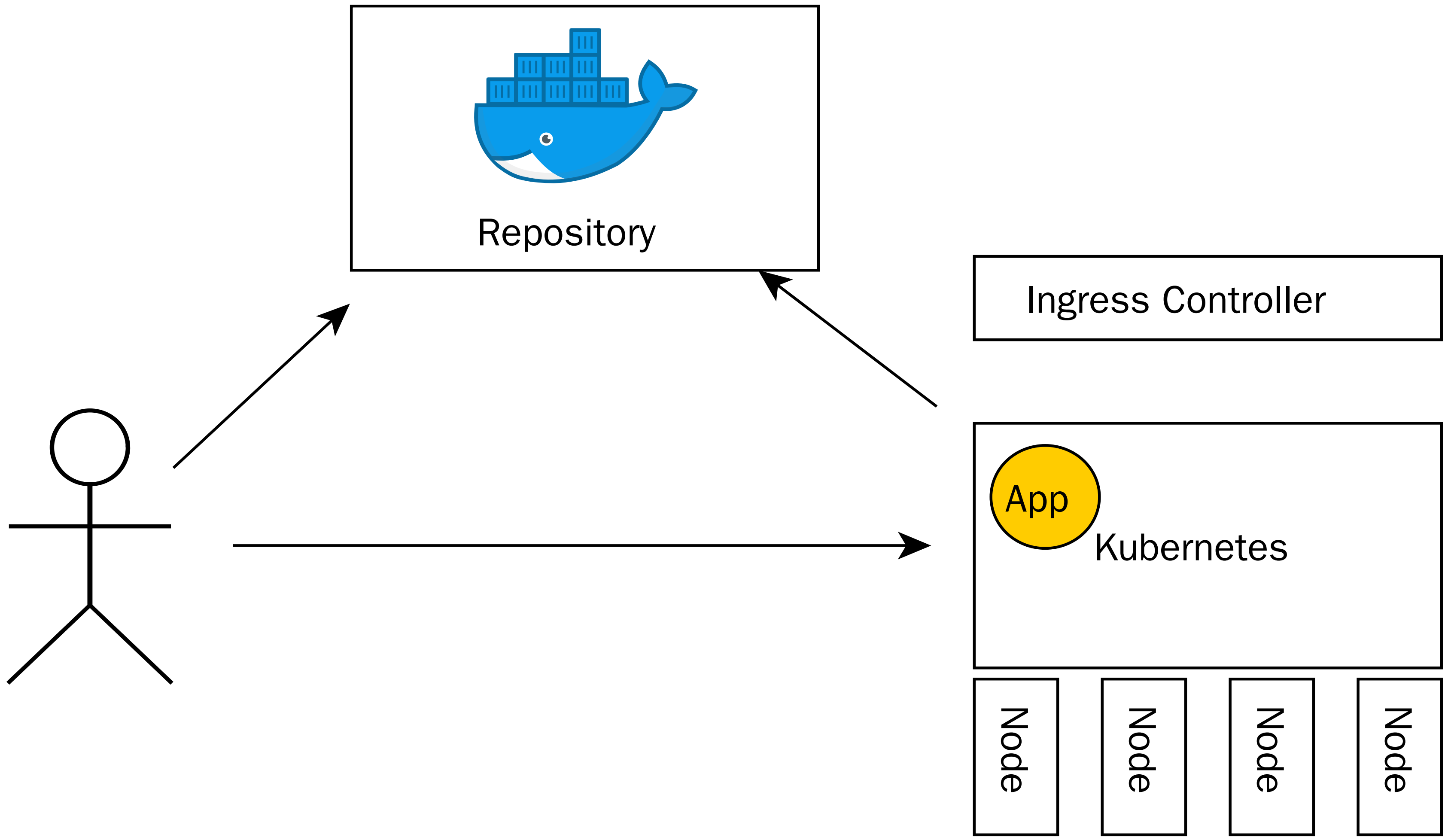
KUBERNETES

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

KUBERNETES

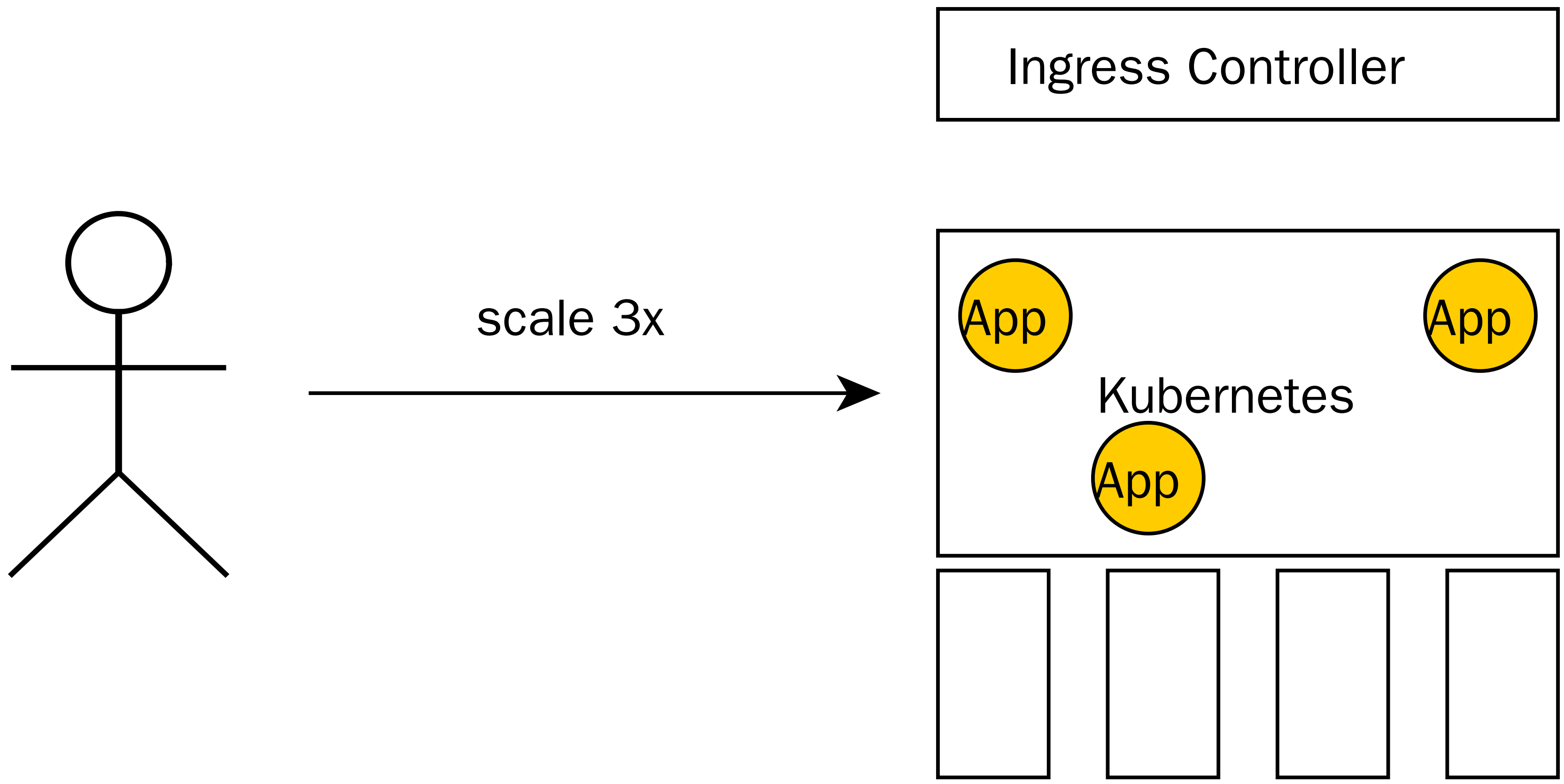
- Batteries for your 12factory apps
- Service discovery, meta-data support
- Utilize resources to nearly 100%

KUBERNETES



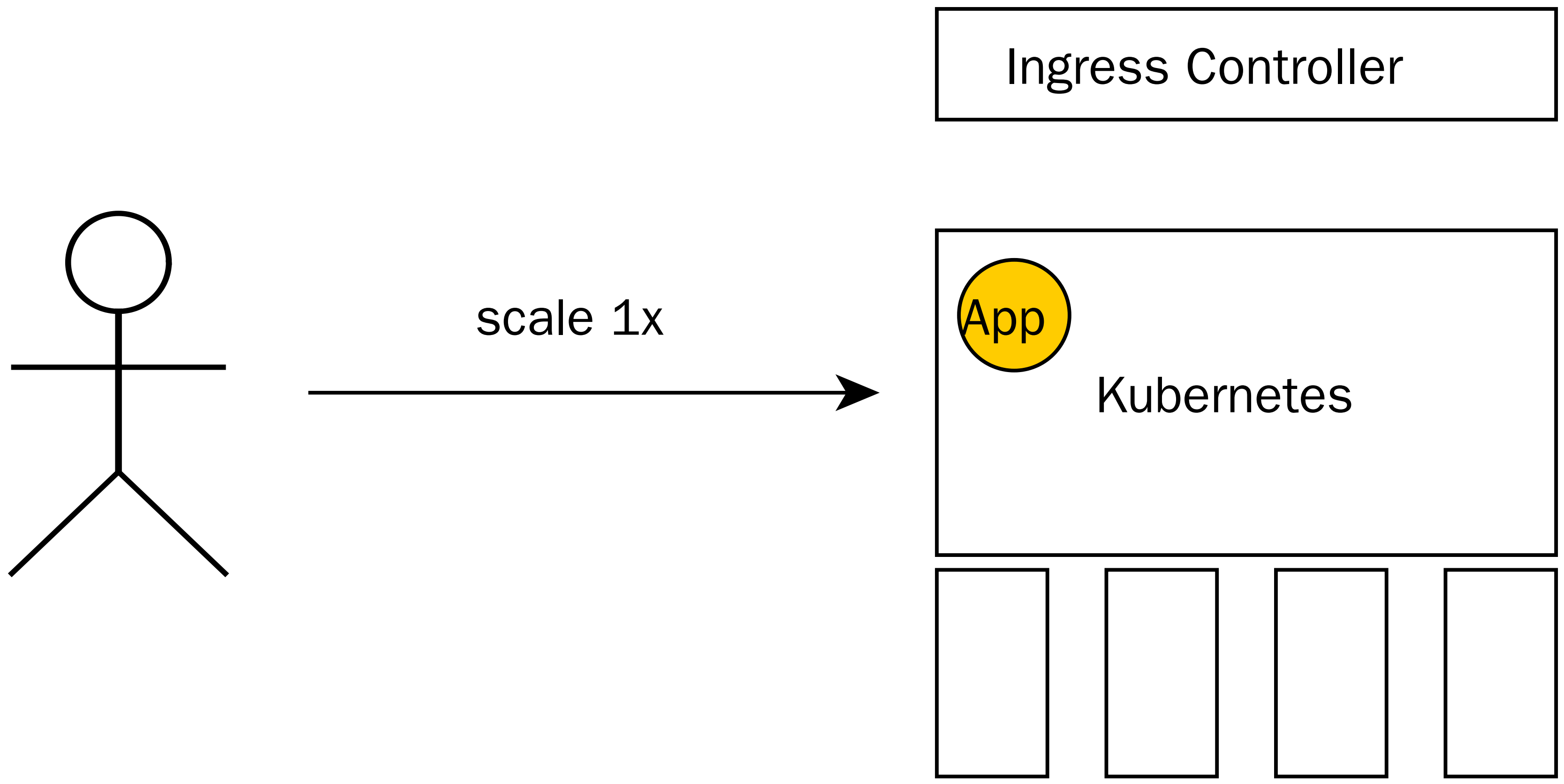
```
make docker_push; kubectl create -f app-srv-dpl.yaml
```

SCALE UP! SCALE DOWN!



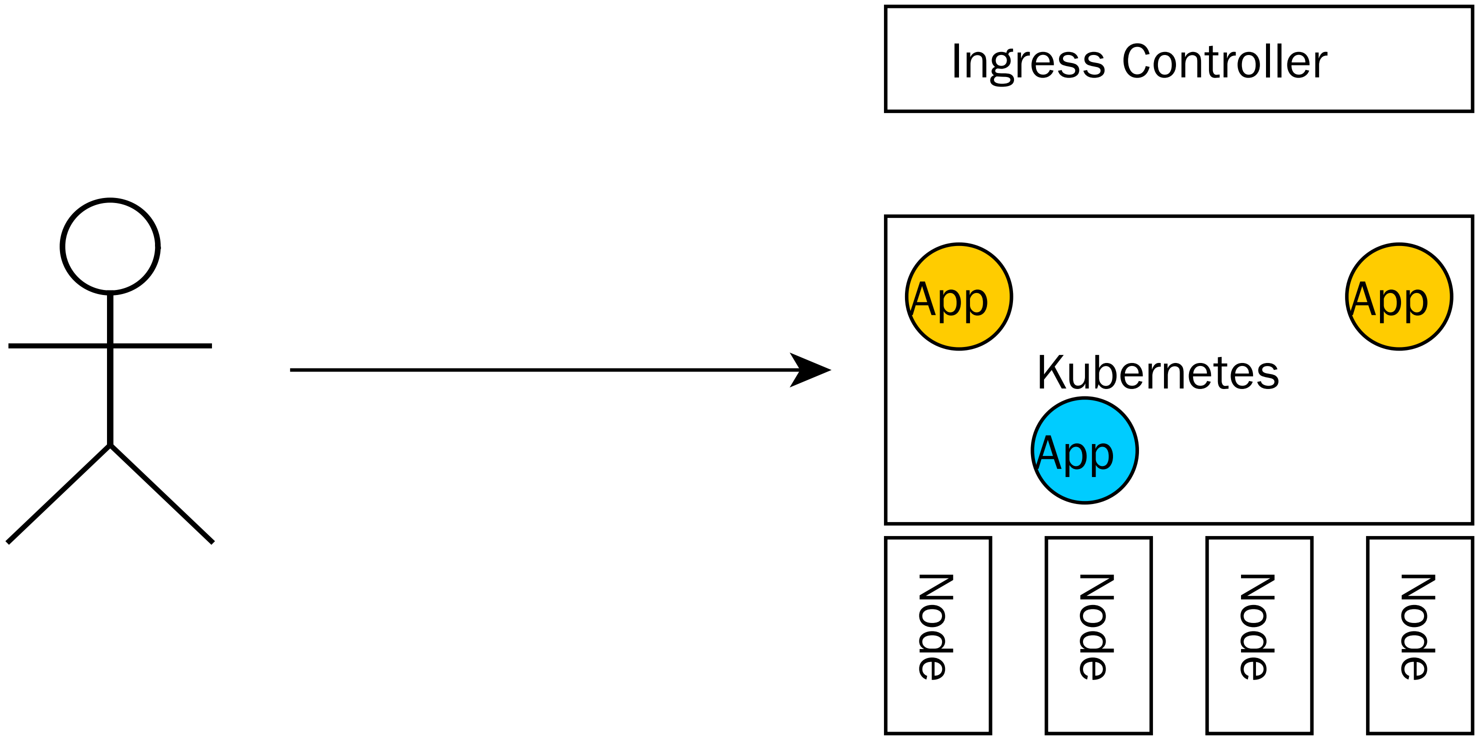
```
kubectl --replicas=3 -f app-srv-dpl.yaml
```

SCALE UP! SCALE DOWN!



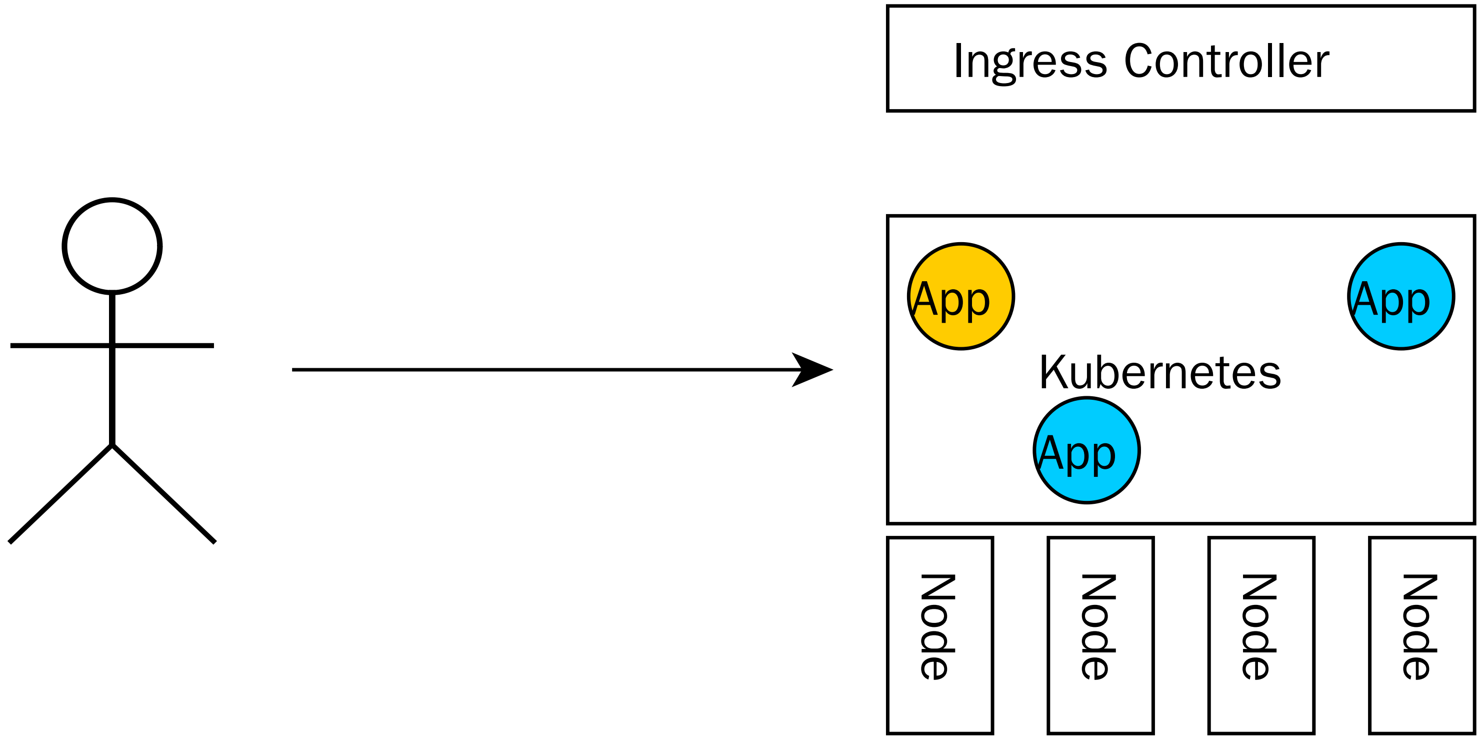
```
kubectl --replicas=1 -f app-srv-dpl.yaml
```

ROLLING UPDATES!

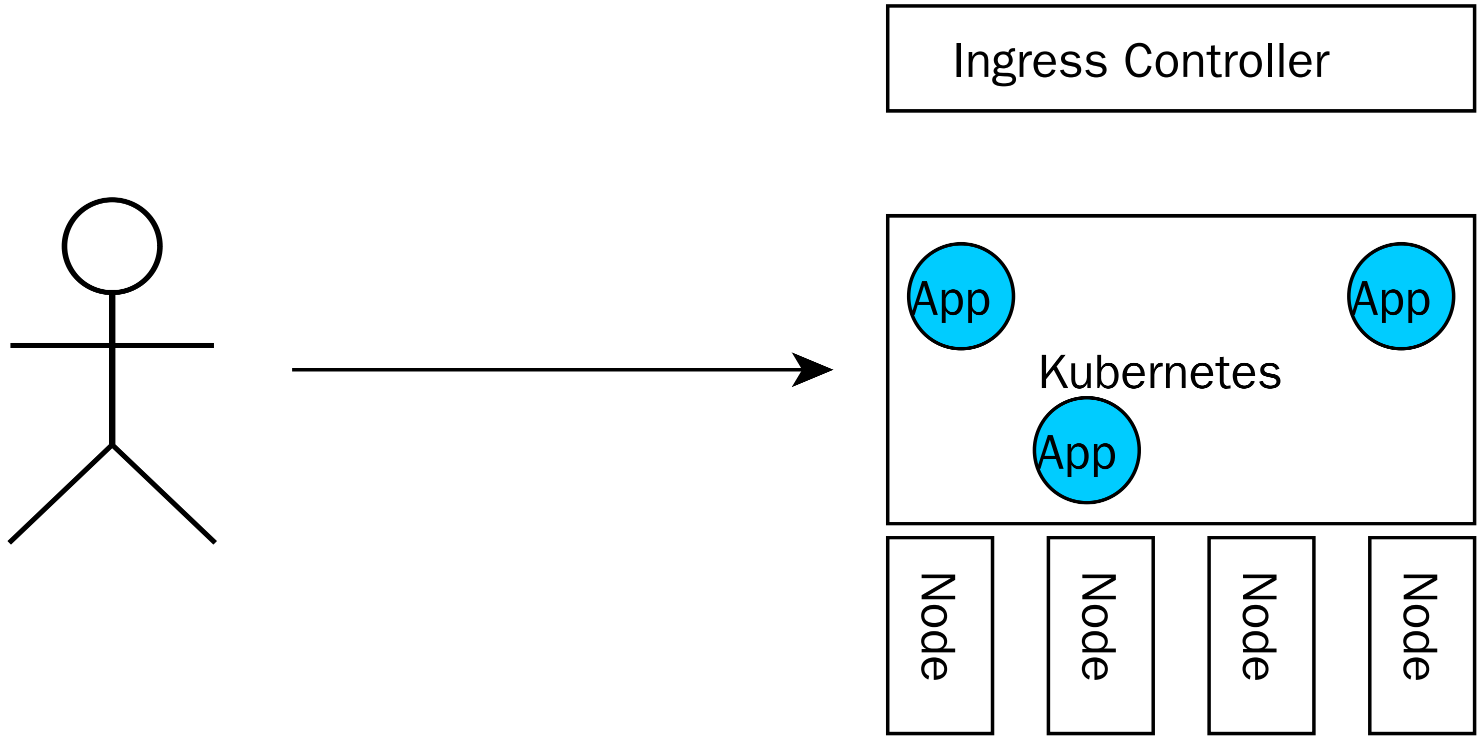


```
kubectl set image deployment/app app=app:v2.0.0
```

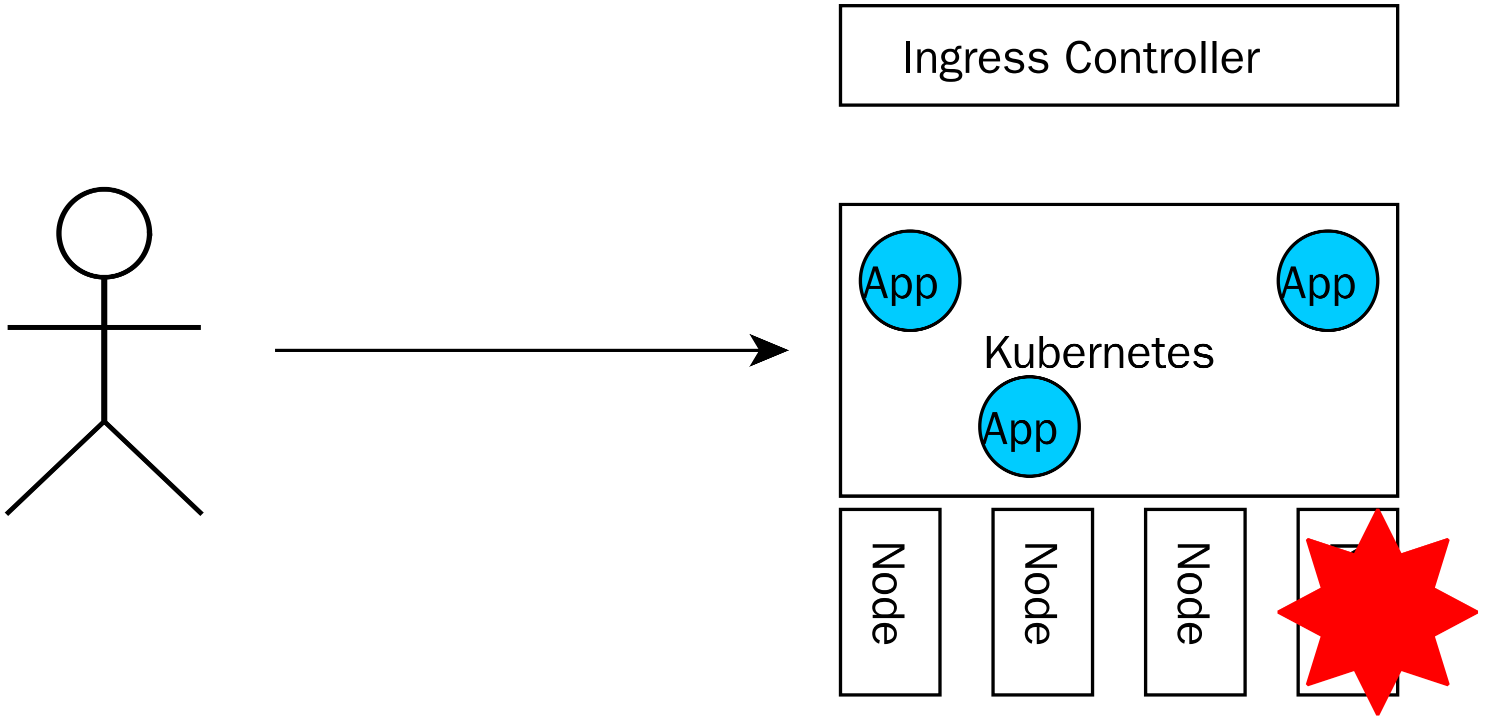
ROLLING UPDATES!



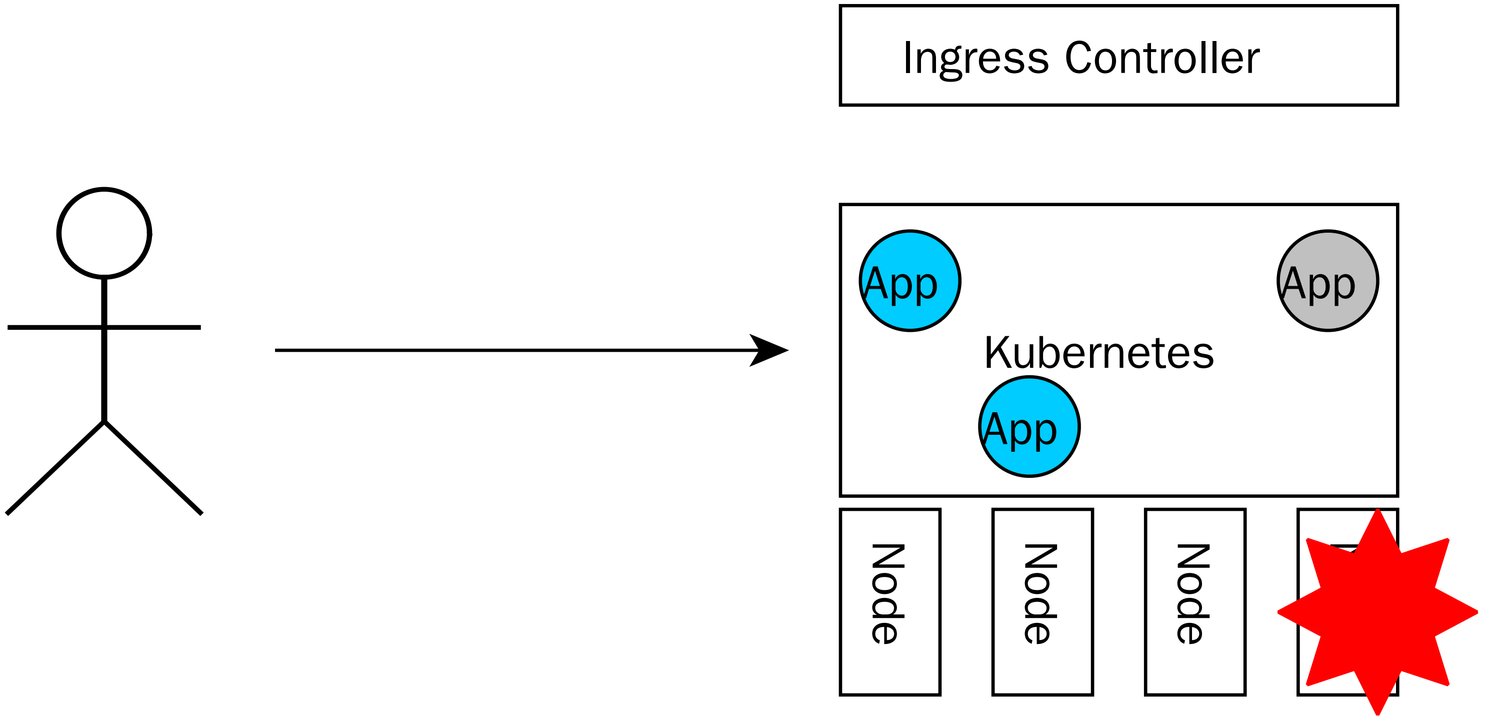
ROLLING UPDATES!



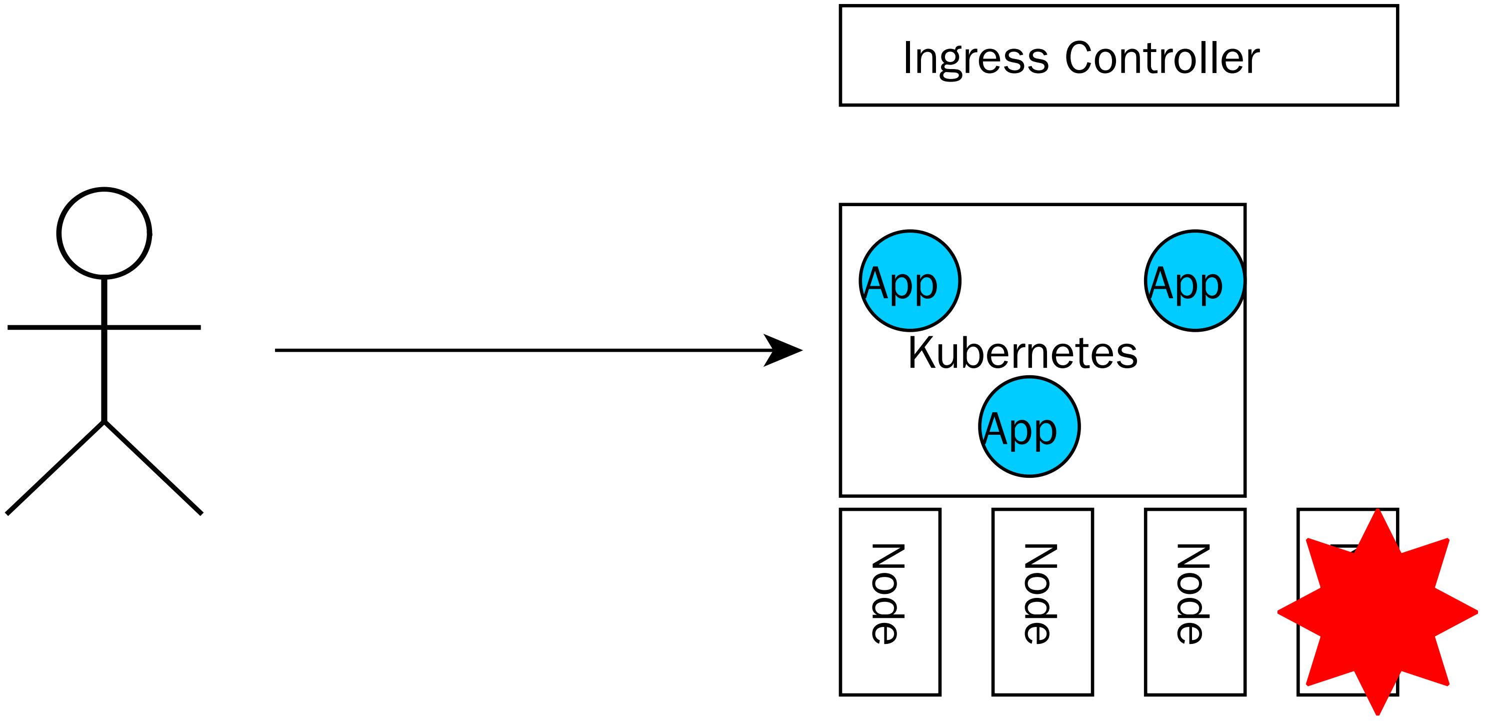
RESISTANCE!



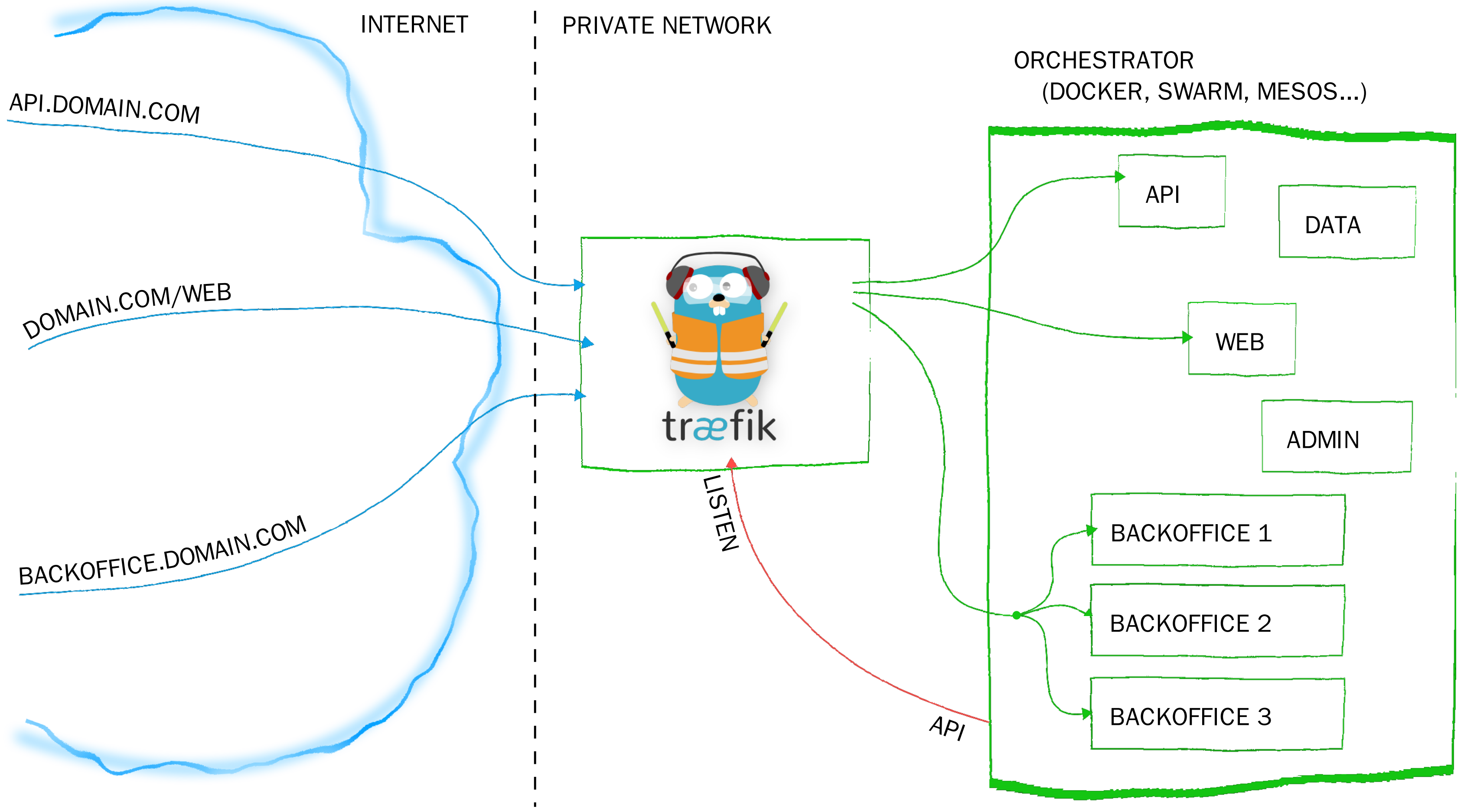
RESISTANCE!



RESISTANCE!



HOW GET USER REQUESTS?

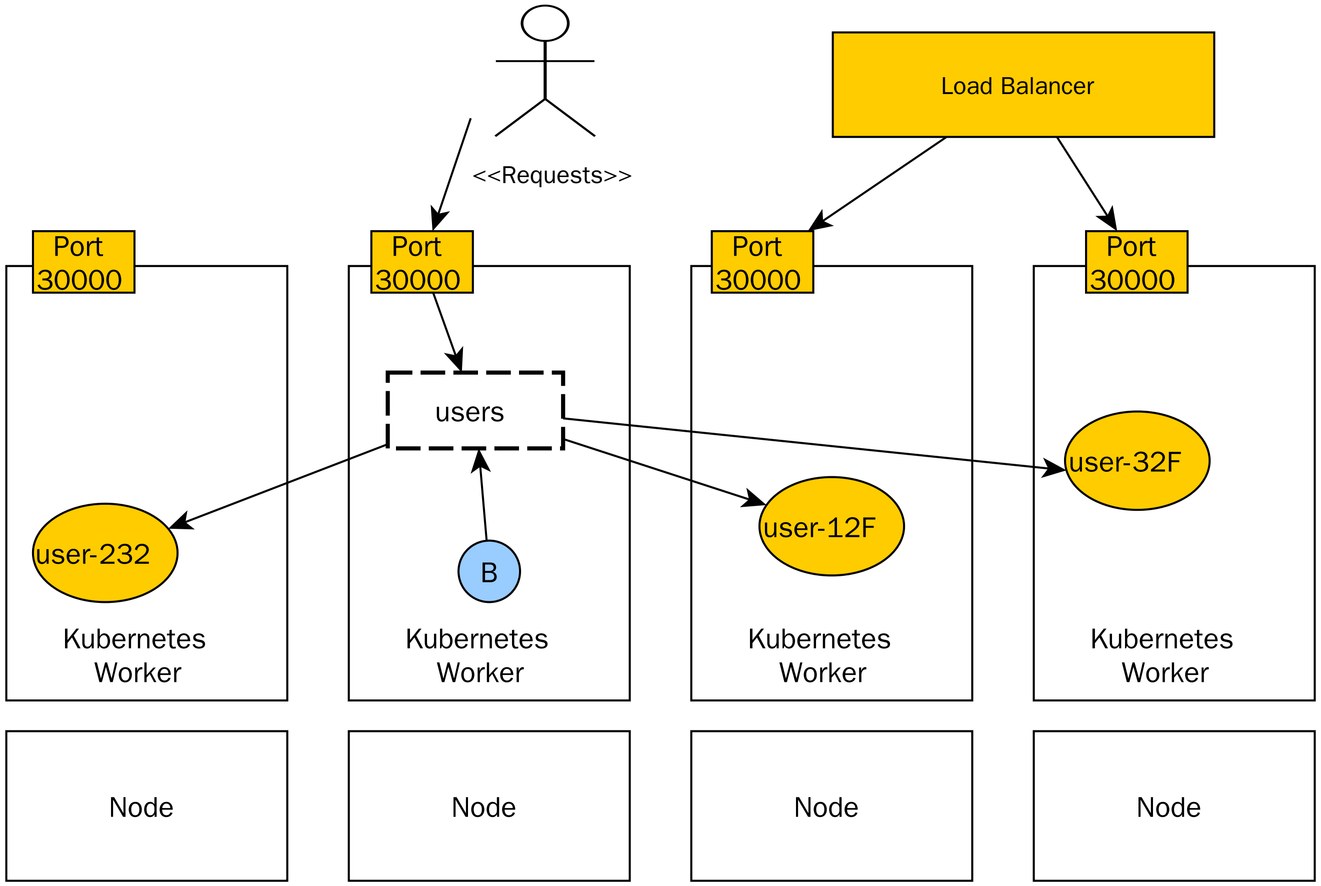


Ingress Controller

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"`

SERVICE DISCOVERY

- loosely couple components
- auto-wiring with logging and monitoring

DROP-IN

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

THE BEST PART

All live in git:

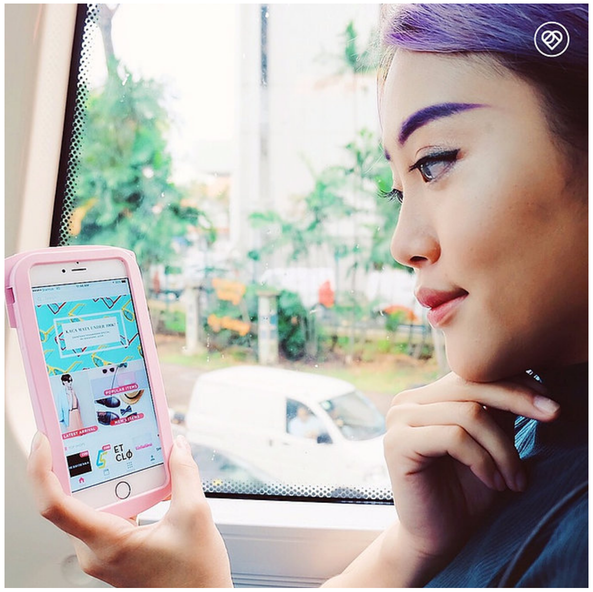
- all in Yaml
- integration with monitoring, alarming
- integration with ingress-controller
- ...
- Devs can forget about infrastructure... almost

DevOps Culture Dream!

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

CHALLENGES

- 50+ VMs in Amazon, 1 VM - 1 App, idle machine
- Puppet, hilarious (manual) deployment process
- Fear
- Forgotten components
- sometimes 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. Prometheus, AlertManager, and Grafana
2. Elasticsearch-Fluentd-Kibana
3. Google Identity-Aware-Proxy to protect all dev dashboards
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, Chef, ... ➡ 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

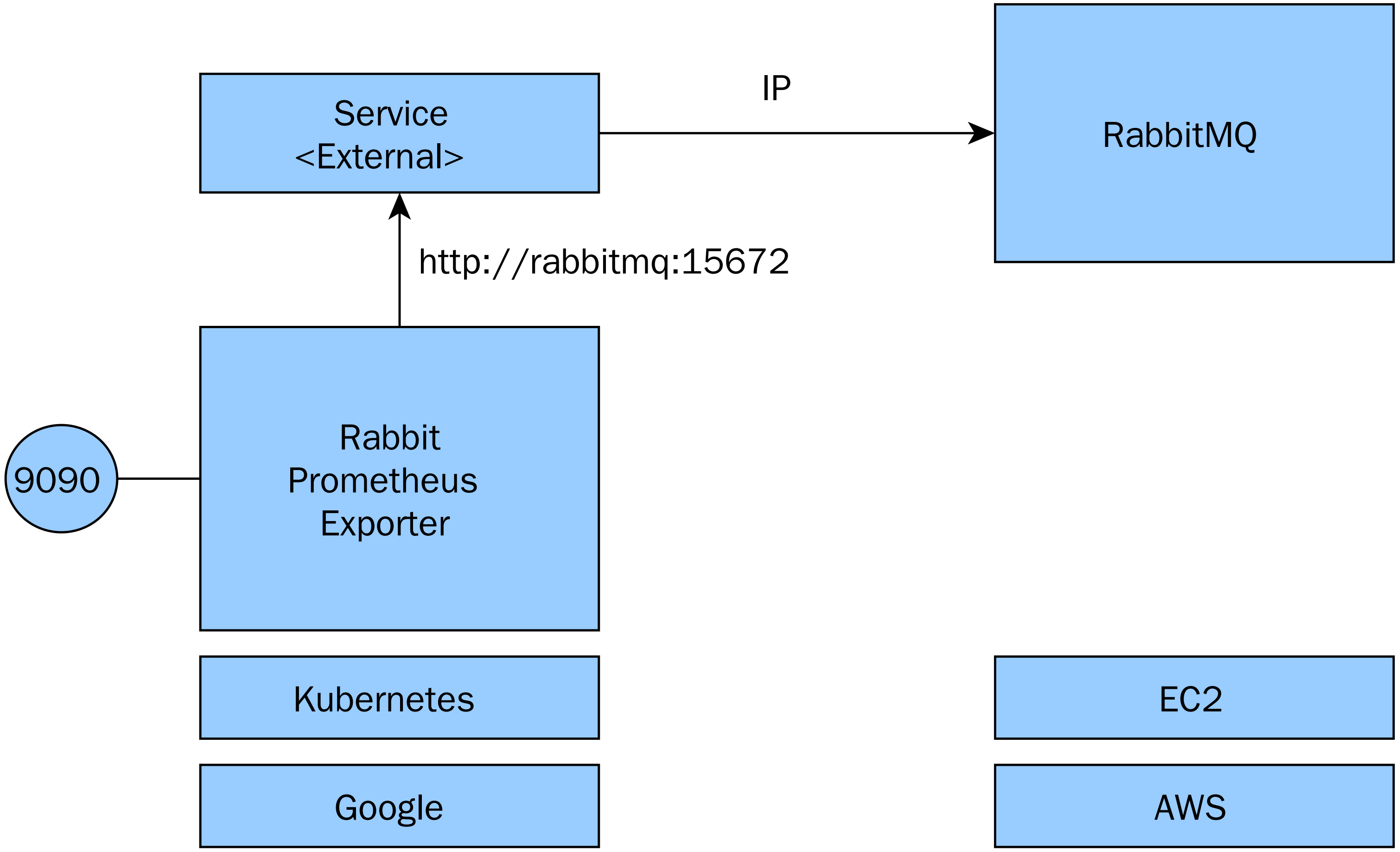
6. KEEP IT RUNNING

Bridge the new with old:

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

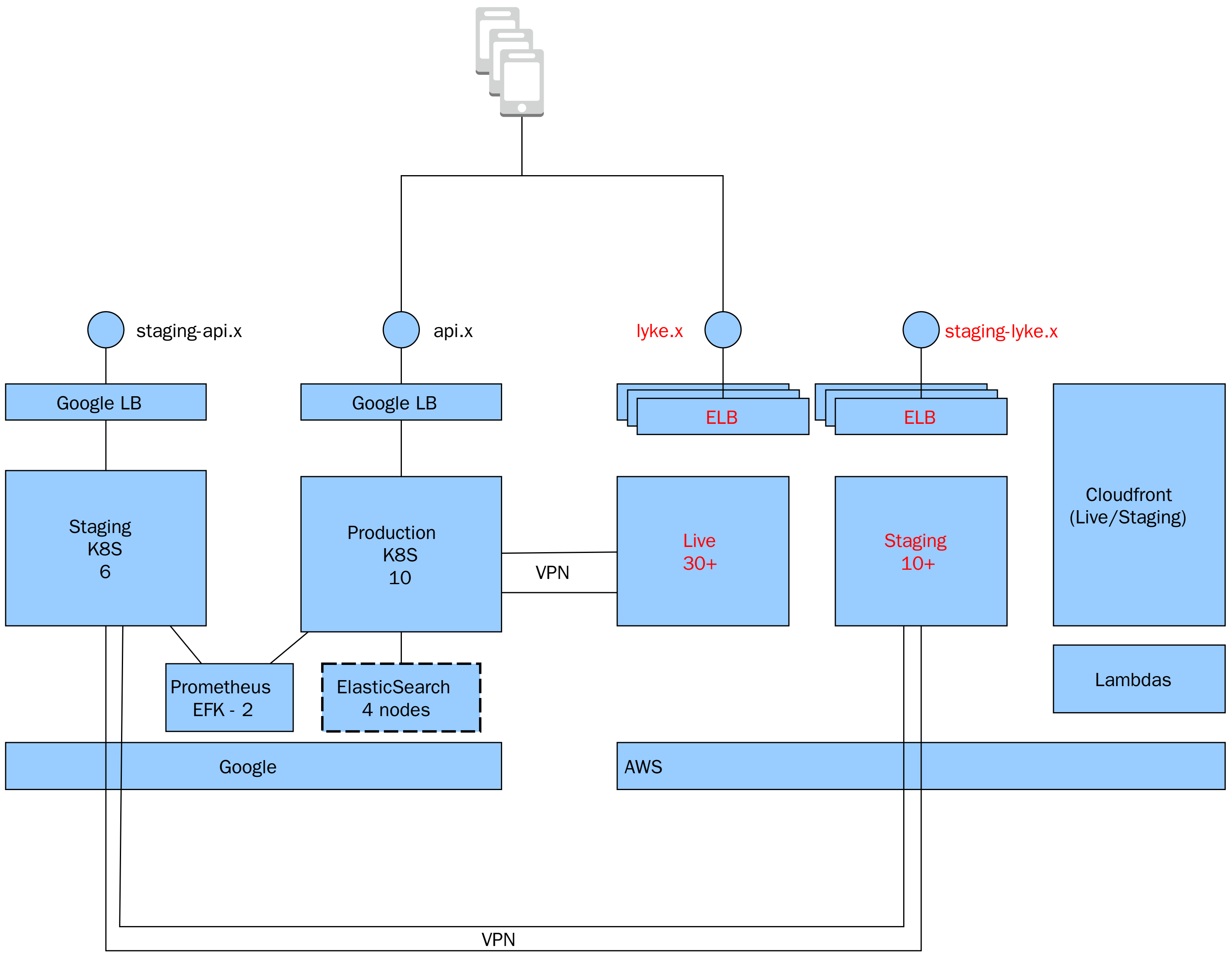
[1] feeding K8S events to HashiCorp consul

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. C&P Makefile and k8s between repos
2. Separate deployments a good transition strategy

WHAT DID NOT WORK

1. Too many PoC, should cut them to 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

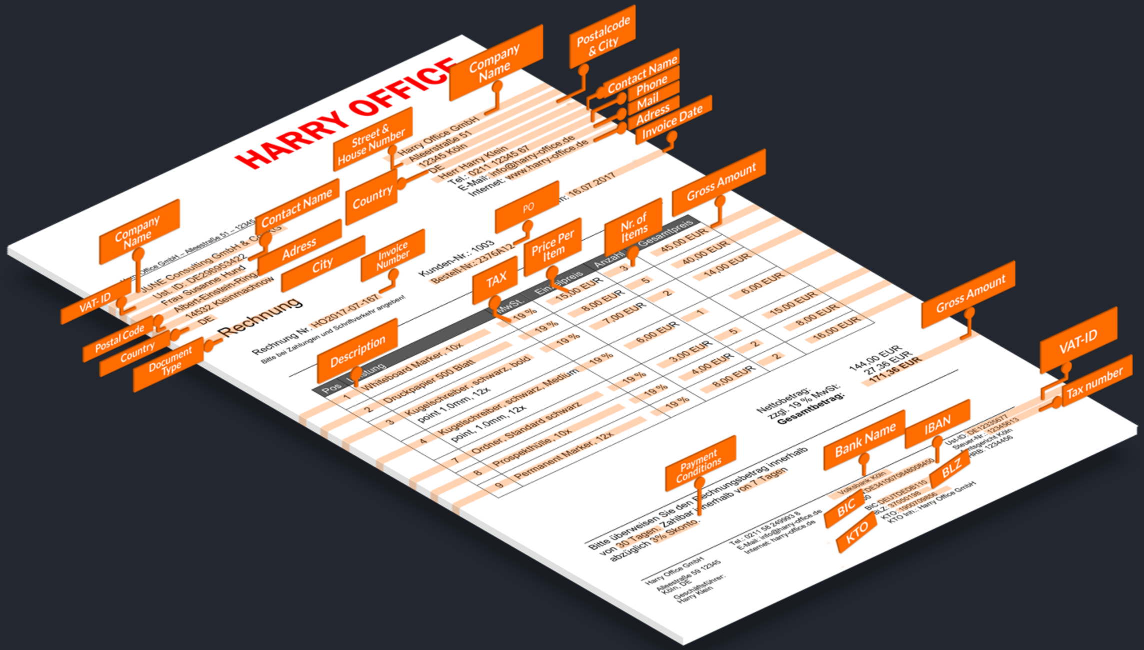
1. Move dev and staging data immediately
2. Let devs know it is a transition stage
3. Teach earlier about resources
4. EFK could wait
5. World-stop for a paid-XXX% weekend for migration

SMACC

Hypatos

Hypatos

Problem SMACC solves



SAP

Microsoft Dynamics

DATEV

SMACC

Hypatos

Global clients

SMACC

STORY

- Legacy on AWS, experiments with AWS ECS :/
- Self-hosted K8S on ProfitBricks
- Get to Microsoft ScaleUp, welcome Azure
- Luckily - AKS

AZURE KUBERNETES SERVICE

- Independent from IaaS
- Our OnPrem = Our OnCloud
- Consolidation of our micro-services
- Plug and play, e.g., monitoring

SIMPLICITY

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

TF also sets our AWS

DIFFERENCE ☠

- Two teams in Berlin and Warsaw
- Me in Warsaw

NEW EXPERIENCE

- devs really do not like TravisCI ... k8s yamls
- transition from PB to AKS was painful

SOLUTION

- make everything ligther
- c&p without modifications
- hide the k8s, remove magic
- deploy on *tag*

Similar to the Kelsey Hightower approach

Repo .travis.yml

```yaml
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

PROCESS

1. Generate deploy, ingress, and svc kubernetes files
2. Commit to smacc-platform.git on **staging** branch
3. Deploy to staging environment

PROCESS

1. Create PR in smacc-platform.git for **production** branch
2. On merge, deploy to production

smacc-platform

- 3 independent branches: dev, staging, and master
- Target for other scripts

KUBERNETES

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

WHAT WORKED

- Hiding k8s
- Go for ubuntu-based docker images

WOULD DO DIFFERENT

- More sensitive to feedback

NEXT

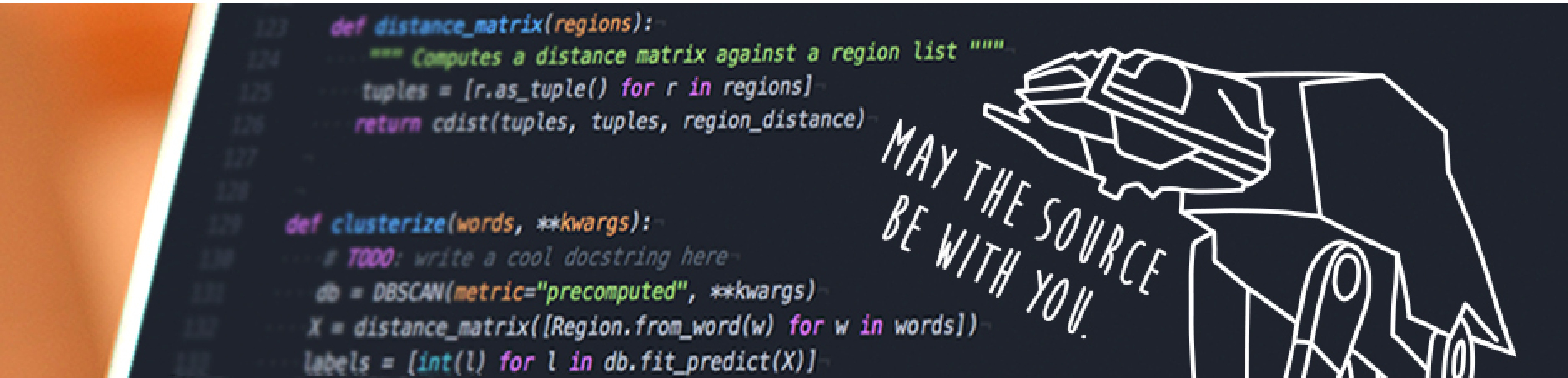
- Acceptance tests on every deployment
- Scale our ML trainings on the top of k8s
- Deployment tool based on missy
- Keeping an eye on Istio

K8S - Linux

- Kubernetes not a silver bullet, but damn close
- Common runtime for onPrem and onCloud
- The biggest asset - the API
- With service discovery - an integration platform
- With kubevirt - might replace your Openstack

DZIĘKUJĘ. PYTANIA?

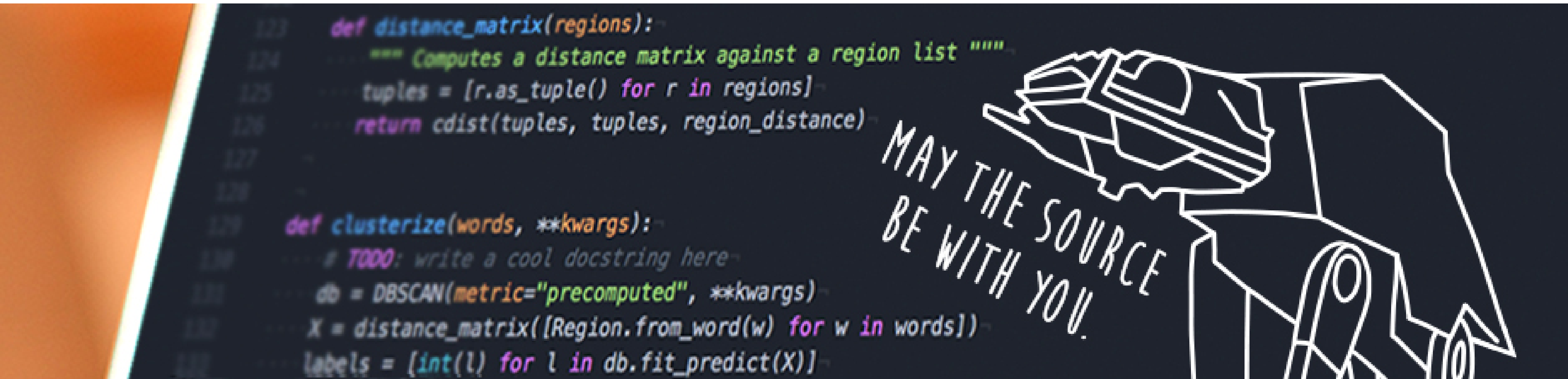
ps. We are hiring.



```
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)]
```

HIRING

- Senior Polyglot Software Engineers
- Experienced System Engineers
- Front-end Engineers
- 1 Data-Driven Product Manager

Apply: hello-warsaw@smacc.io,
Questions? wojciech.barczynski@smacc.io, FB or LI

We will teach you Go if needed. No k8s or ML, we will take care of that.

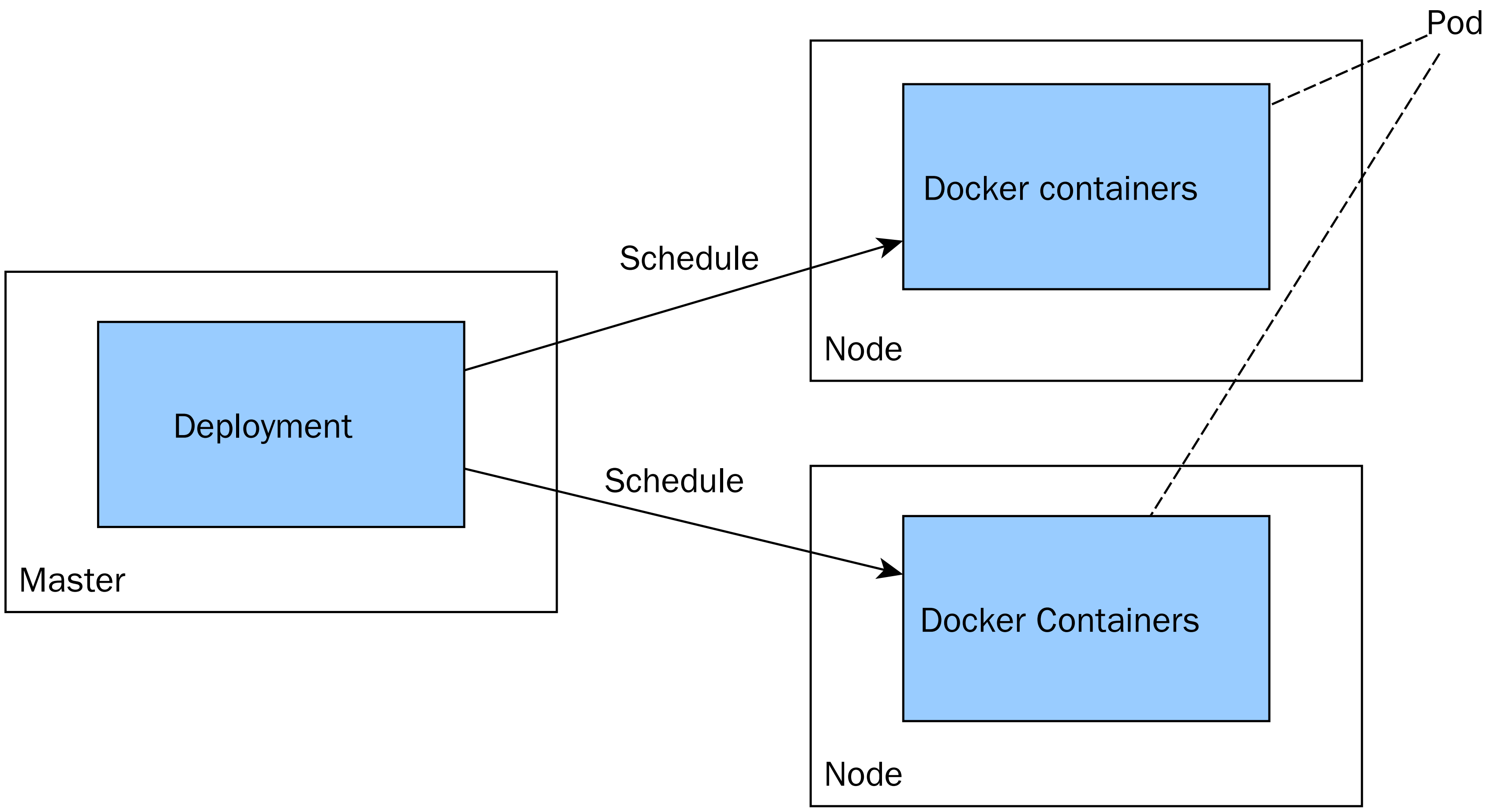
0.1 ➡ 1.0

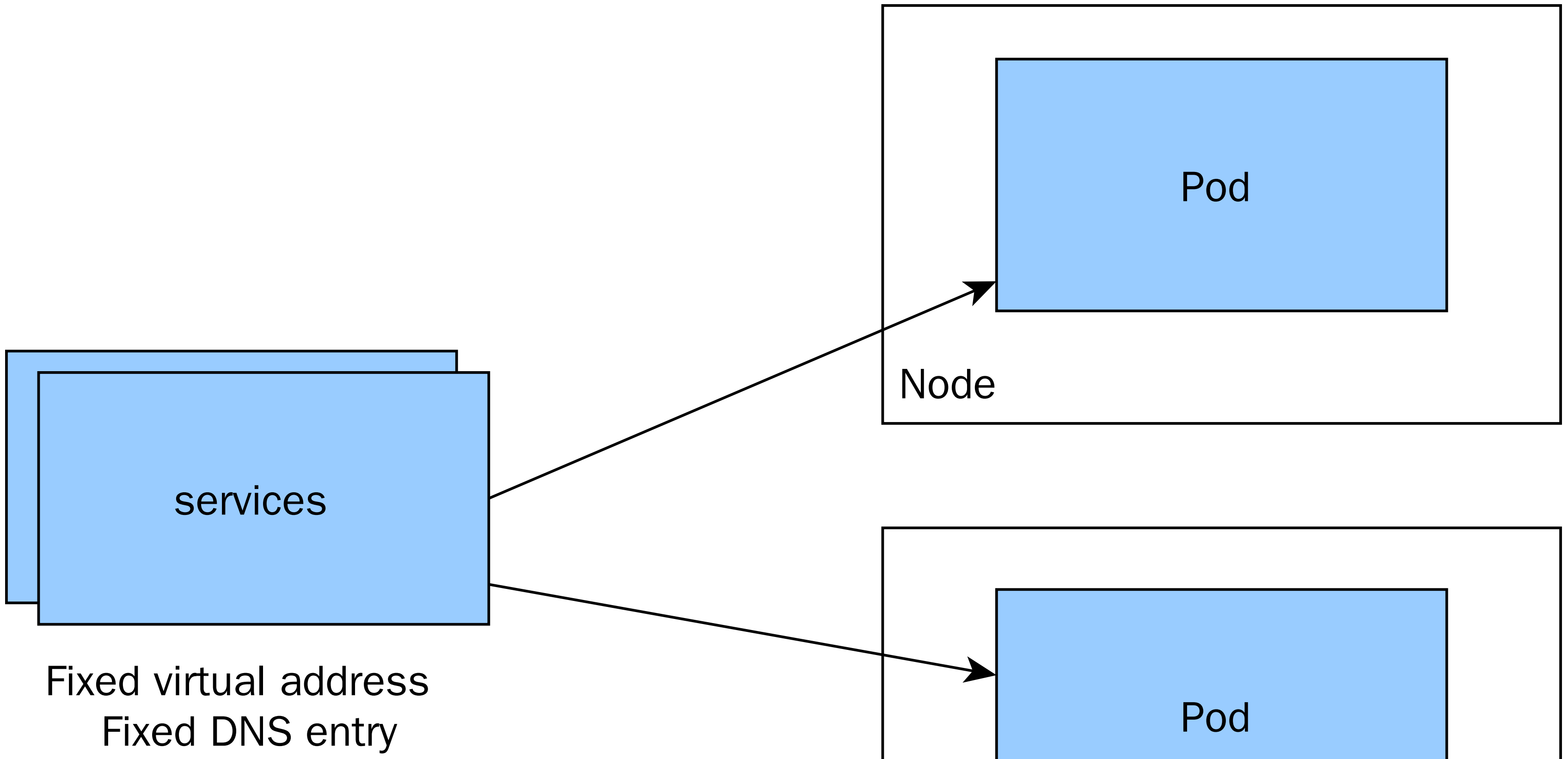
CHANGE THE WORK ORGANIZATION

- From Scrum
- To Kanban

For the next talk

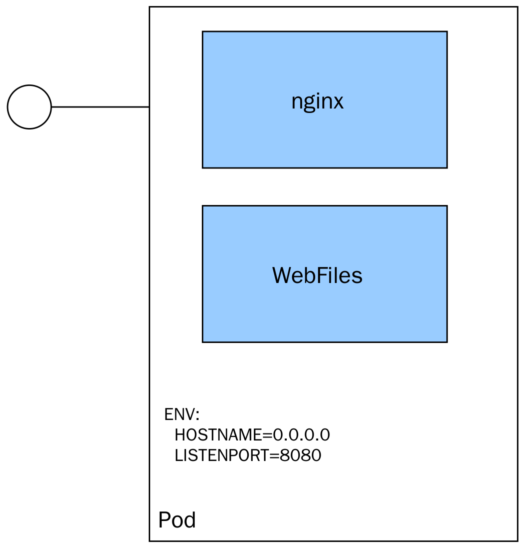
KUBERNETES CONCEPTS


Pod
Docker containers
Schedule
Node
Deployment
Schedule
Master
Docker Containers
Node

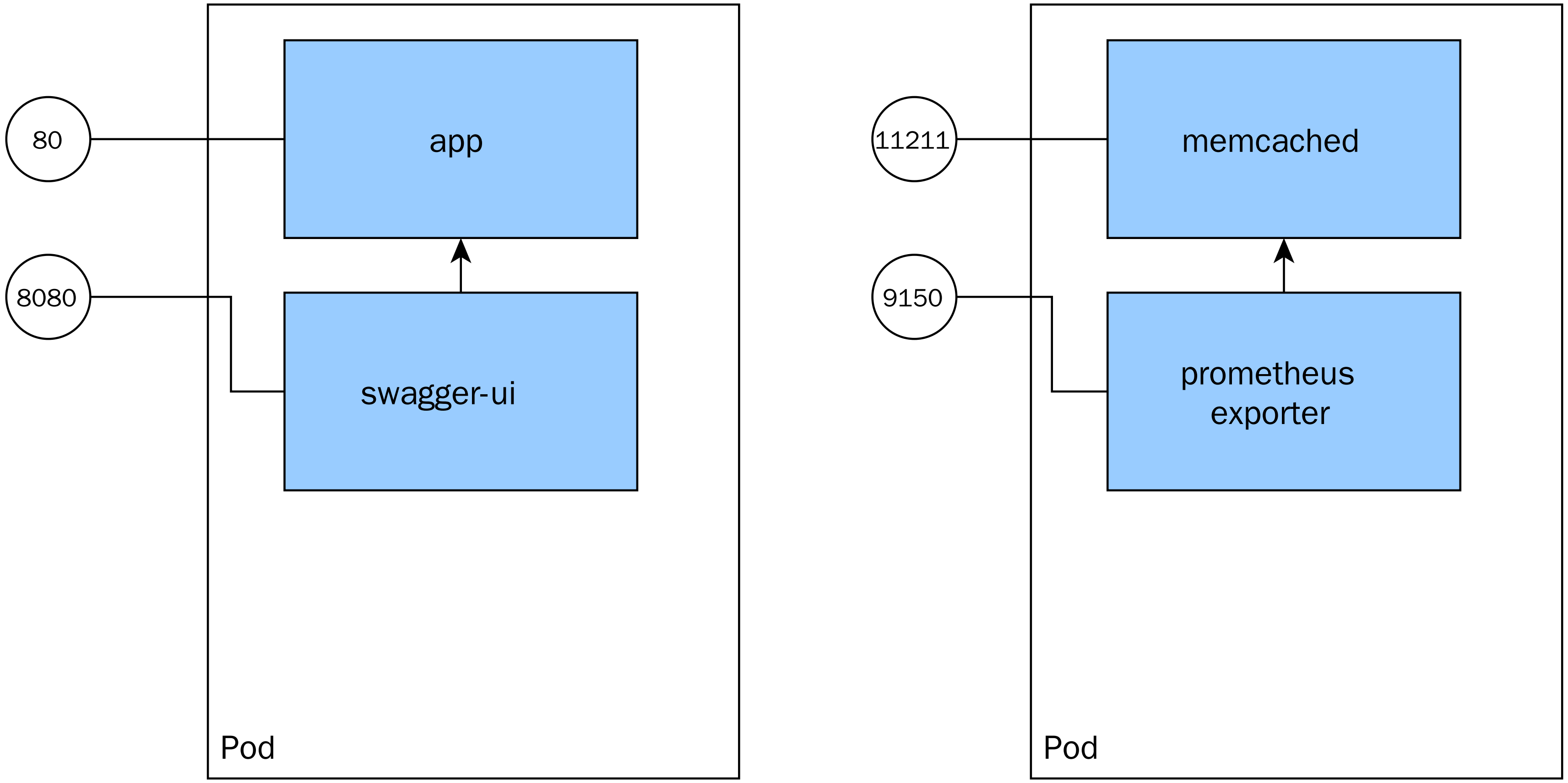

services
Fixed virtual address
Fixed DNS entry
Pod
Node
Pod
Node

PODS

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



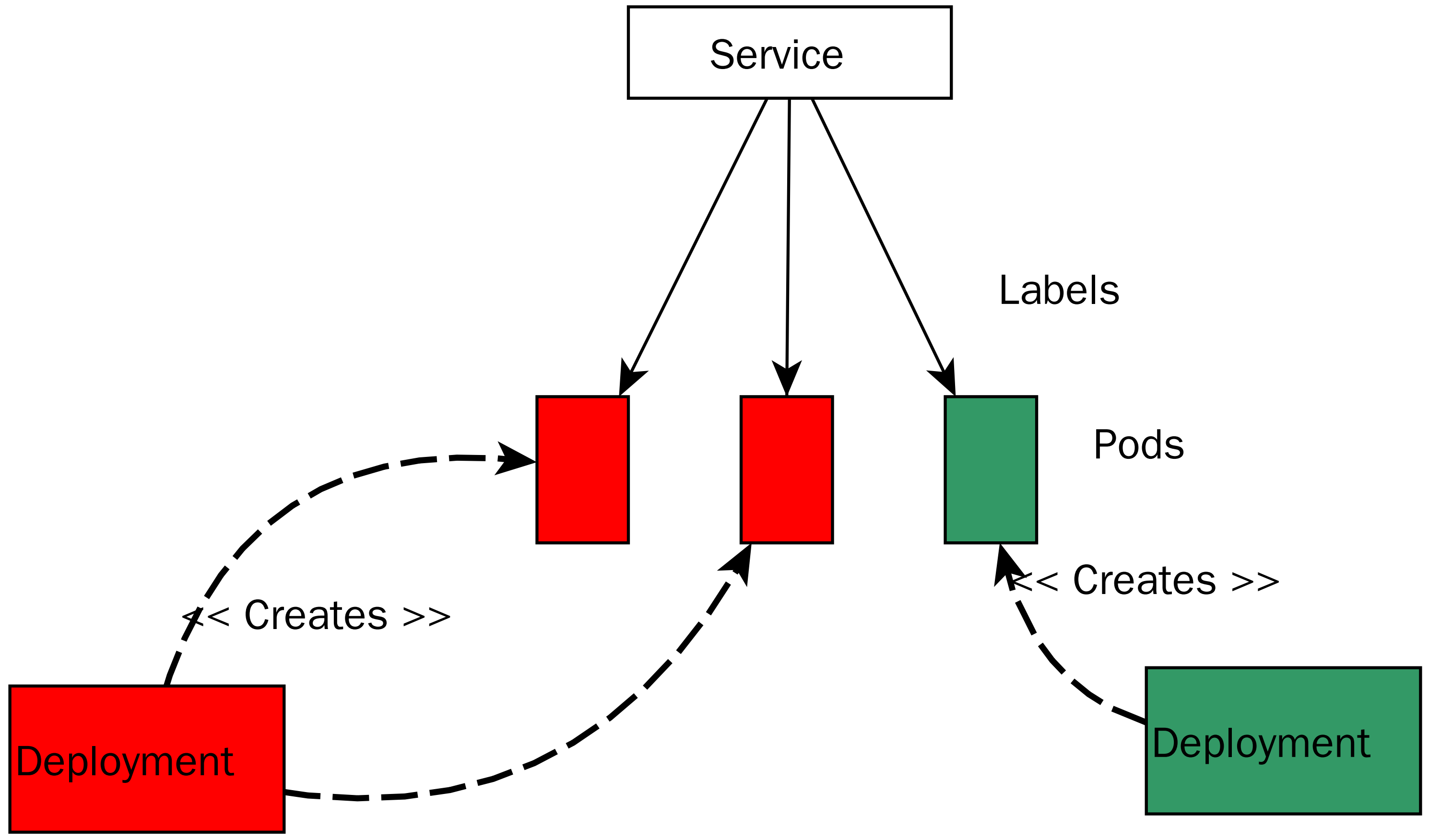
SIDE-CARS


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