DEPLOYMENT STRATEGIES FOR YOUR MICRO-SERVICE WITH K8S

+ HOW TO PREPARE YOUR SERVICE

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

WOJCIECH BARCZYŃSKI

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

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]

AGENDA.

- Mikroserwises
- Dlaczego kubernetes?
- Strategie deploymentu
- Jak przygotować mikroserwis?

MICRO-SERVICES

- Scalling the team
- ...
- Scalling the products

MICRO-SERVICES

- Independent
- Following 12factor app
- Self-aware

12FACTOR APP

- Heroku - 2011
- App easy to run in production
- Low TCO
- Easy to manage

MICRO-SERVICES

- */healthz*
- */metrics*
- */readiness*
- */info*

MICRO-SERVICES

- Do not need to share code
- Share the same conventions
- Every git repos looks familiar
- Might be in different tech

MICRO-SERVICES

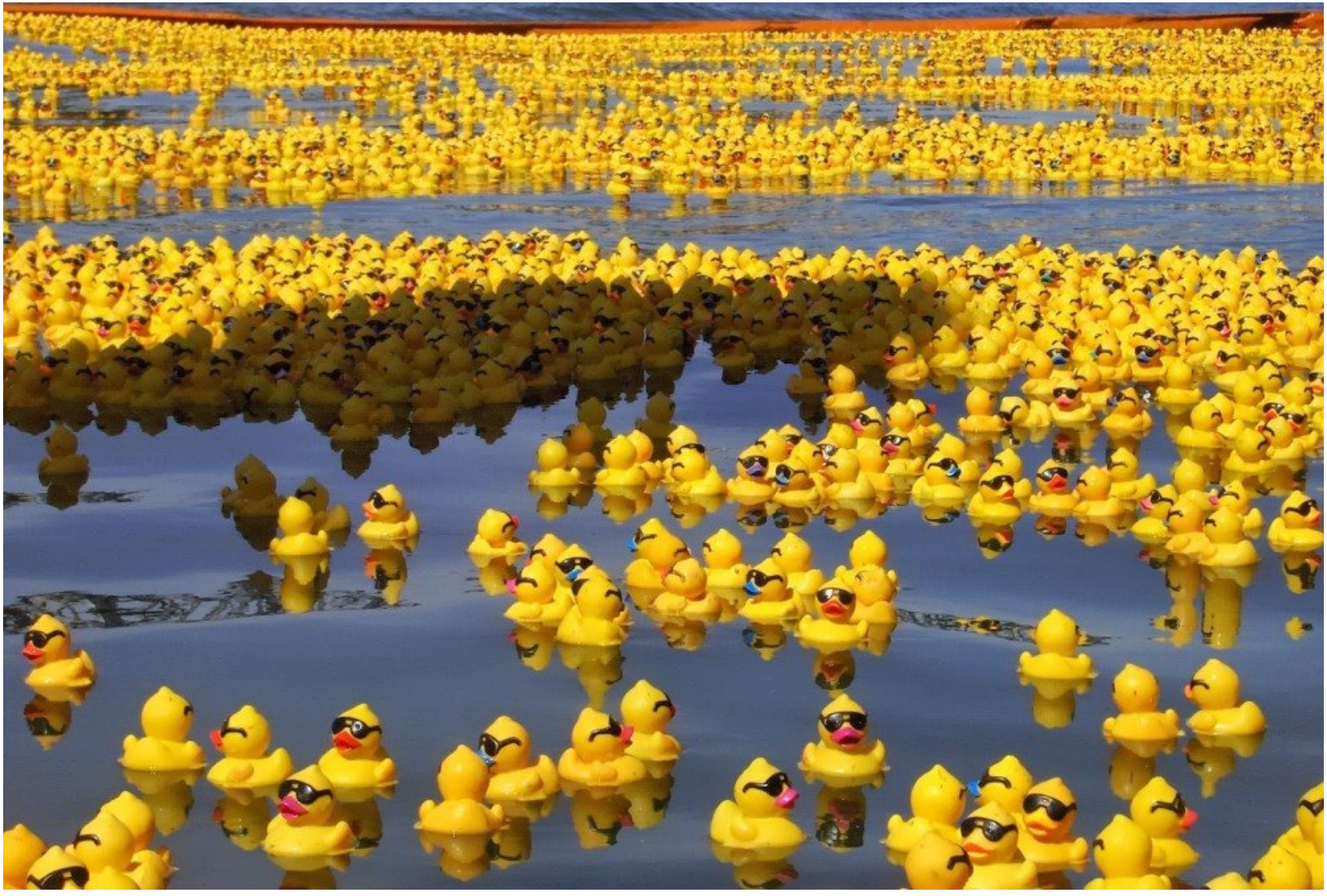
- Not a silver bullet :)

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

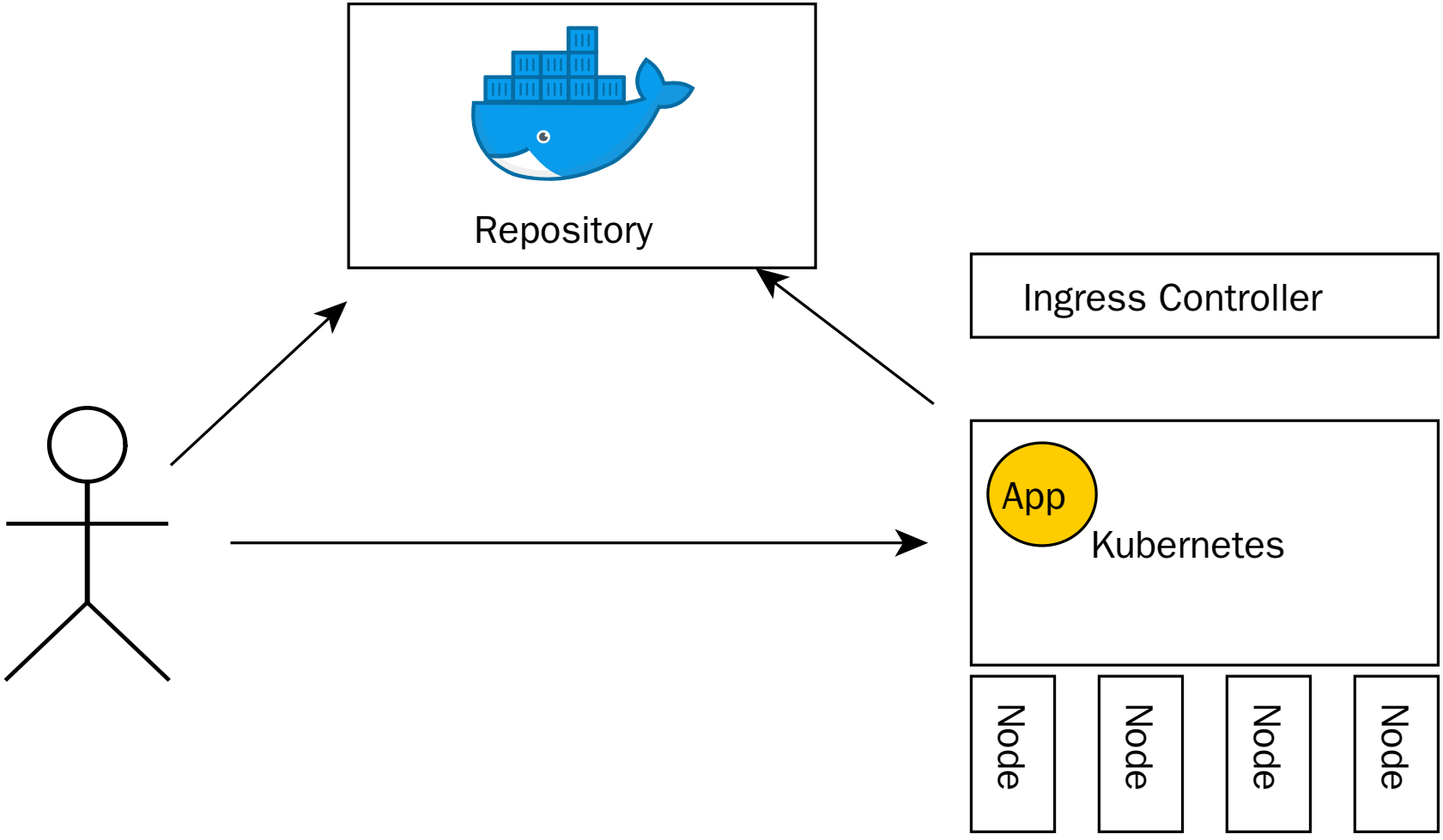
KUBERNETES

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

KUBERNETES

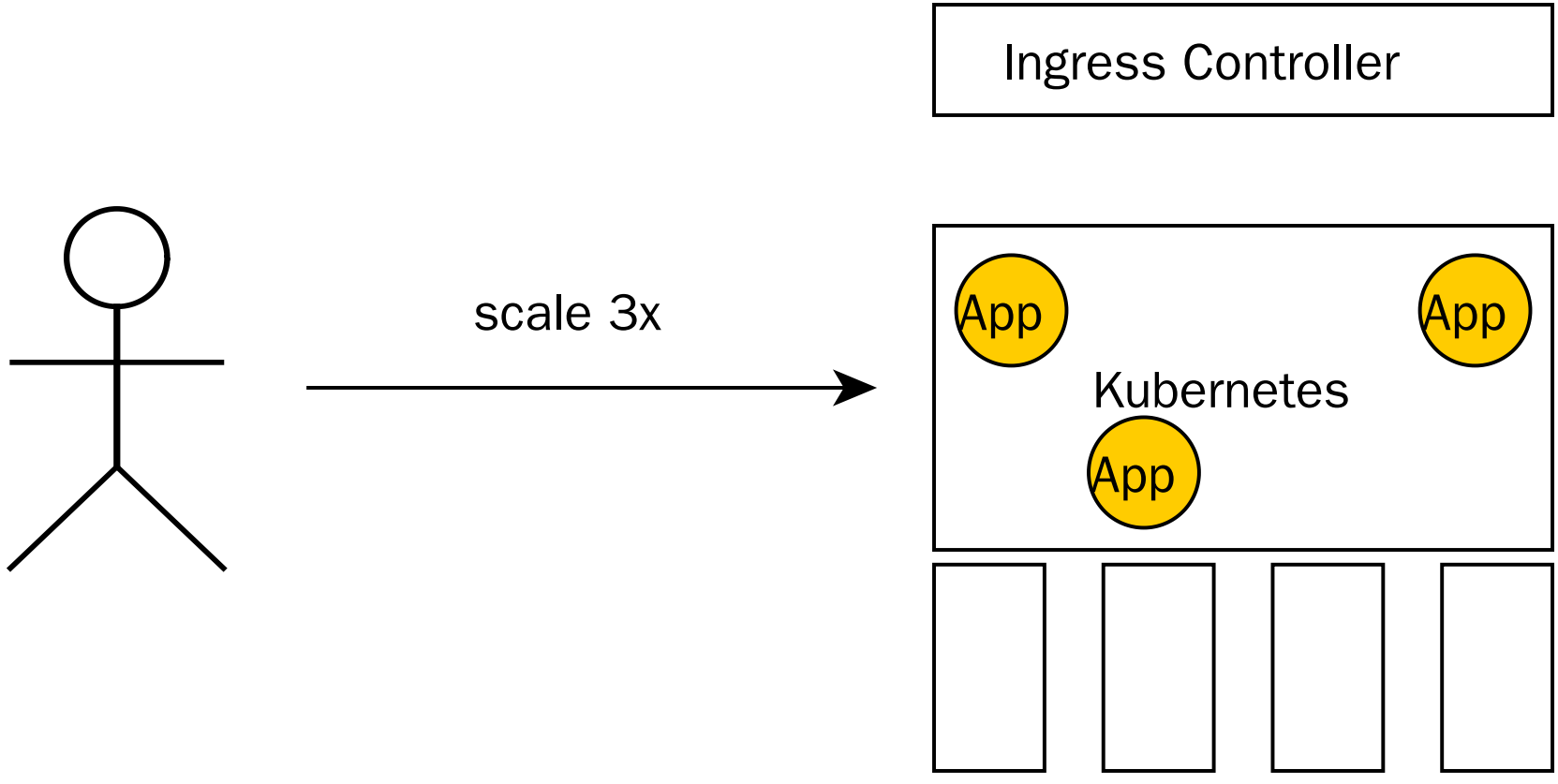
- Batteries for your 12factory apps
- Service discovery, meta-data support
- Utilize resources to nearly 100%

KUBERNETES



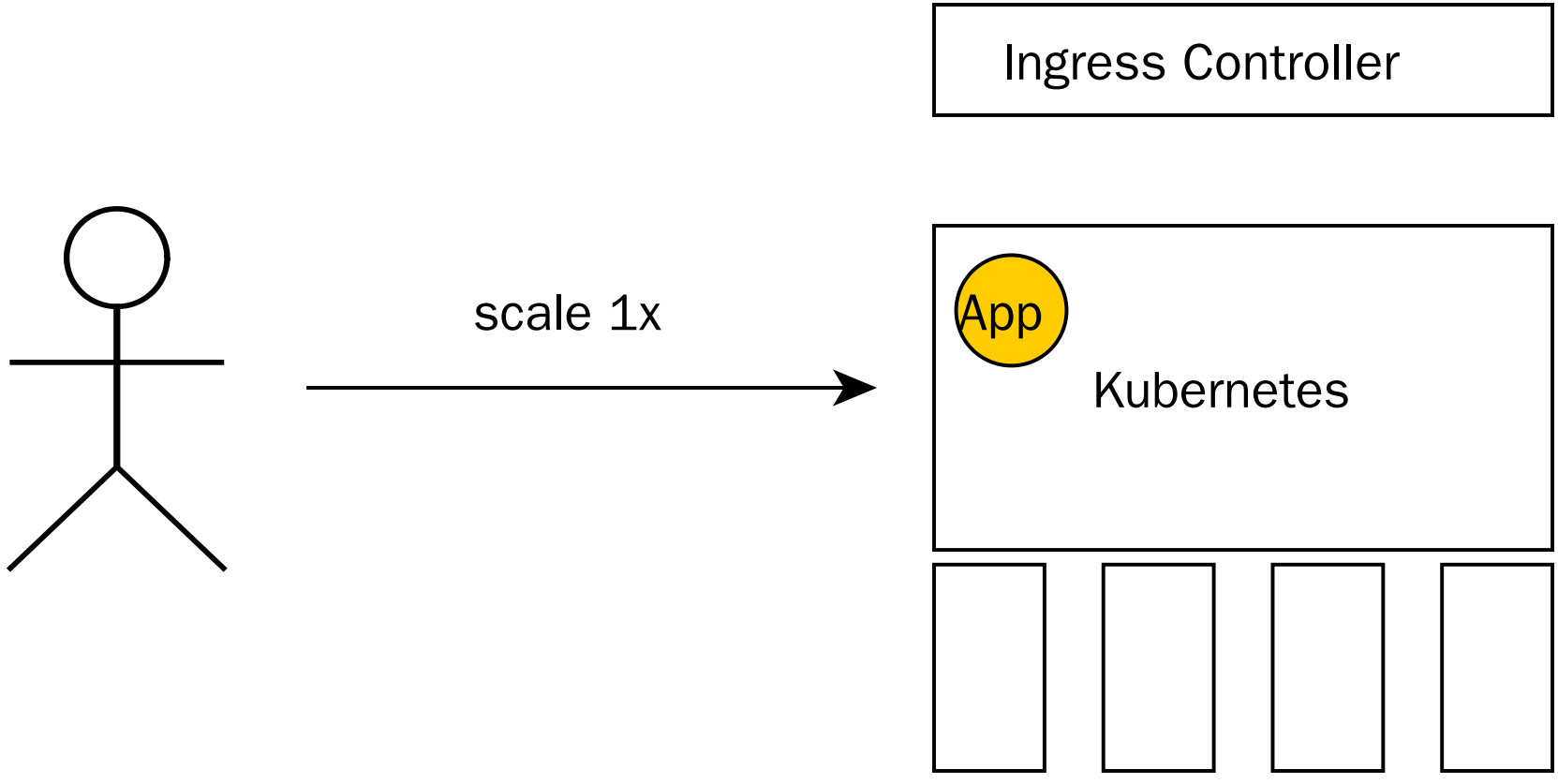

```
make docker_push; kubectl create -f app-srv-dpl.yaml
```

SCALE UP! SCALE DOWN!



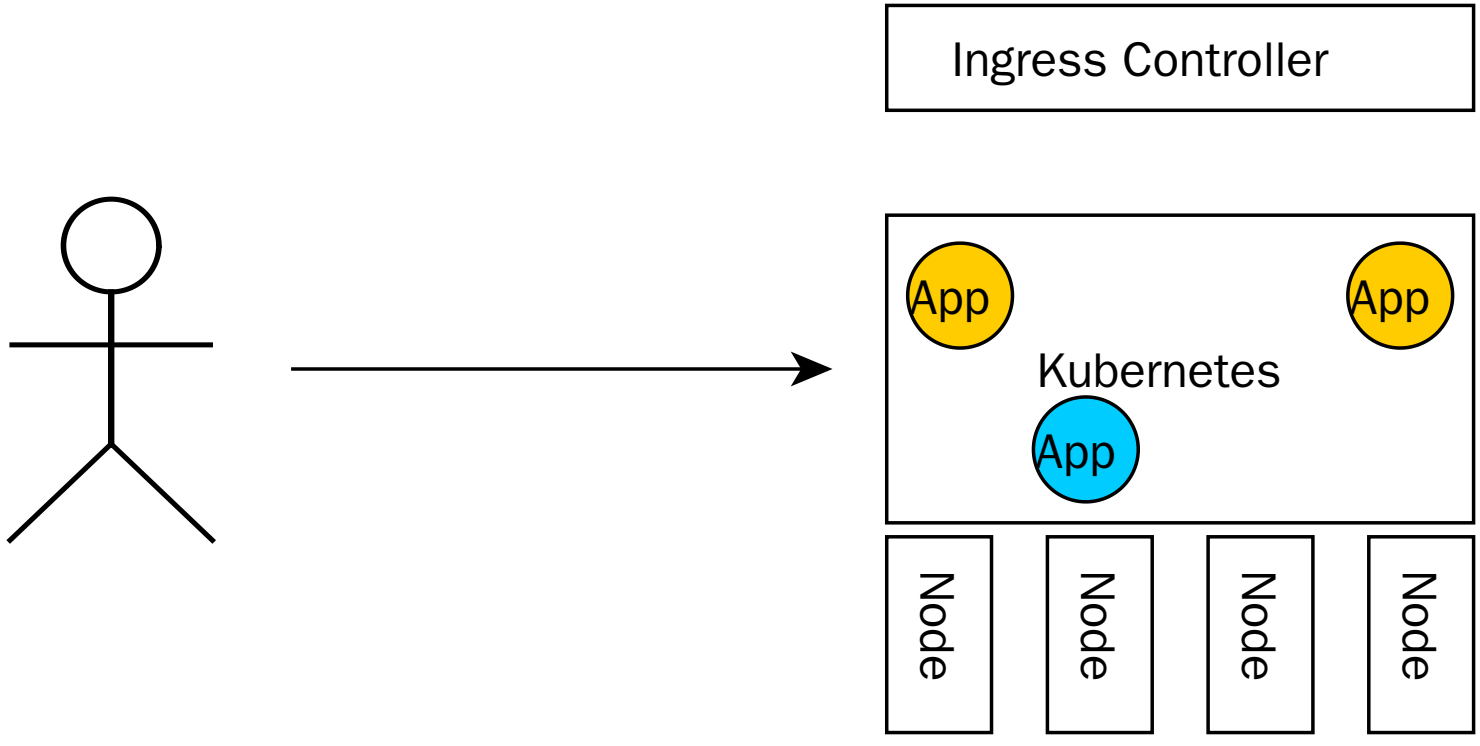

```
kubectl --replicas=3 -f app-srv-dpl.yaml
```

SCALE UP! SCALE DOWN!



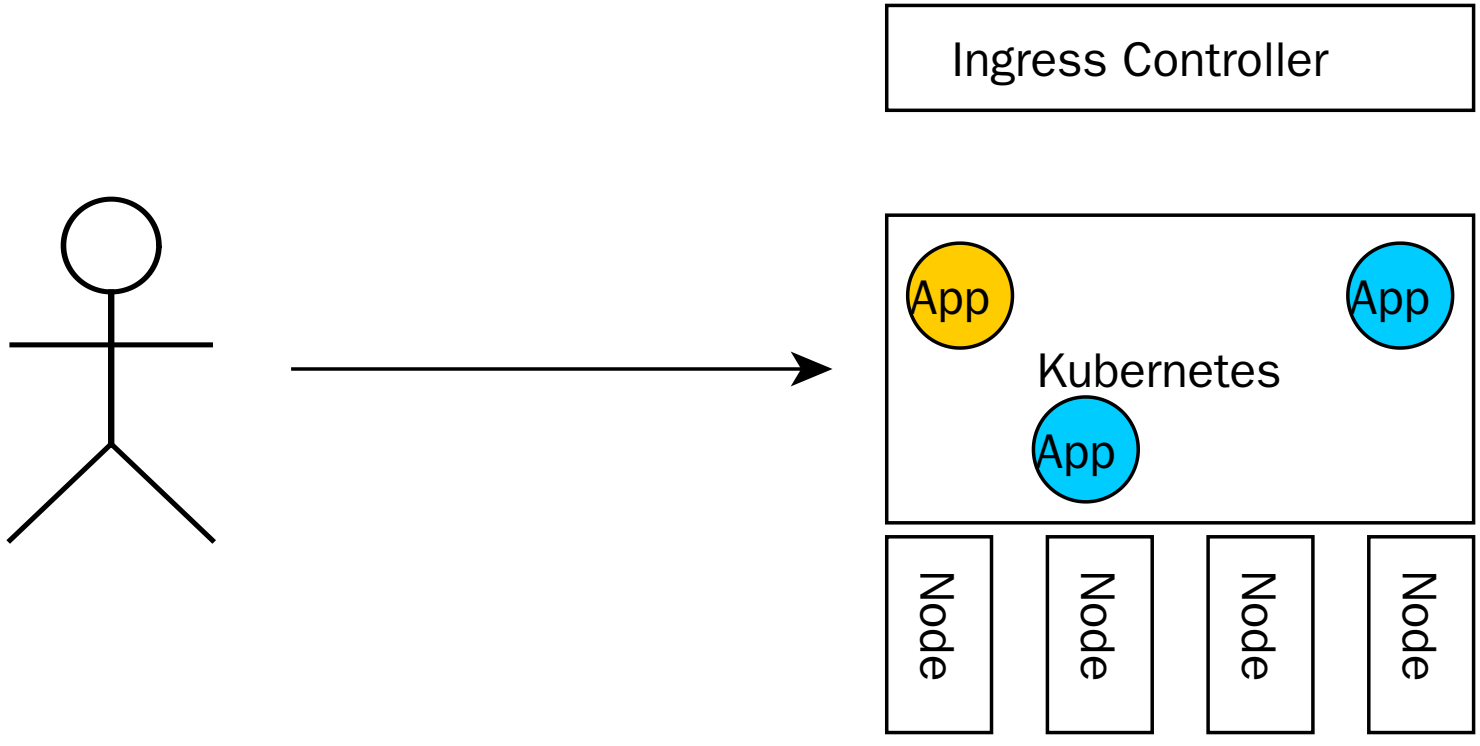

```
kubectl --replicas=1 -f app-srv-dpl.yaml
```

ROLLING UPDATES!



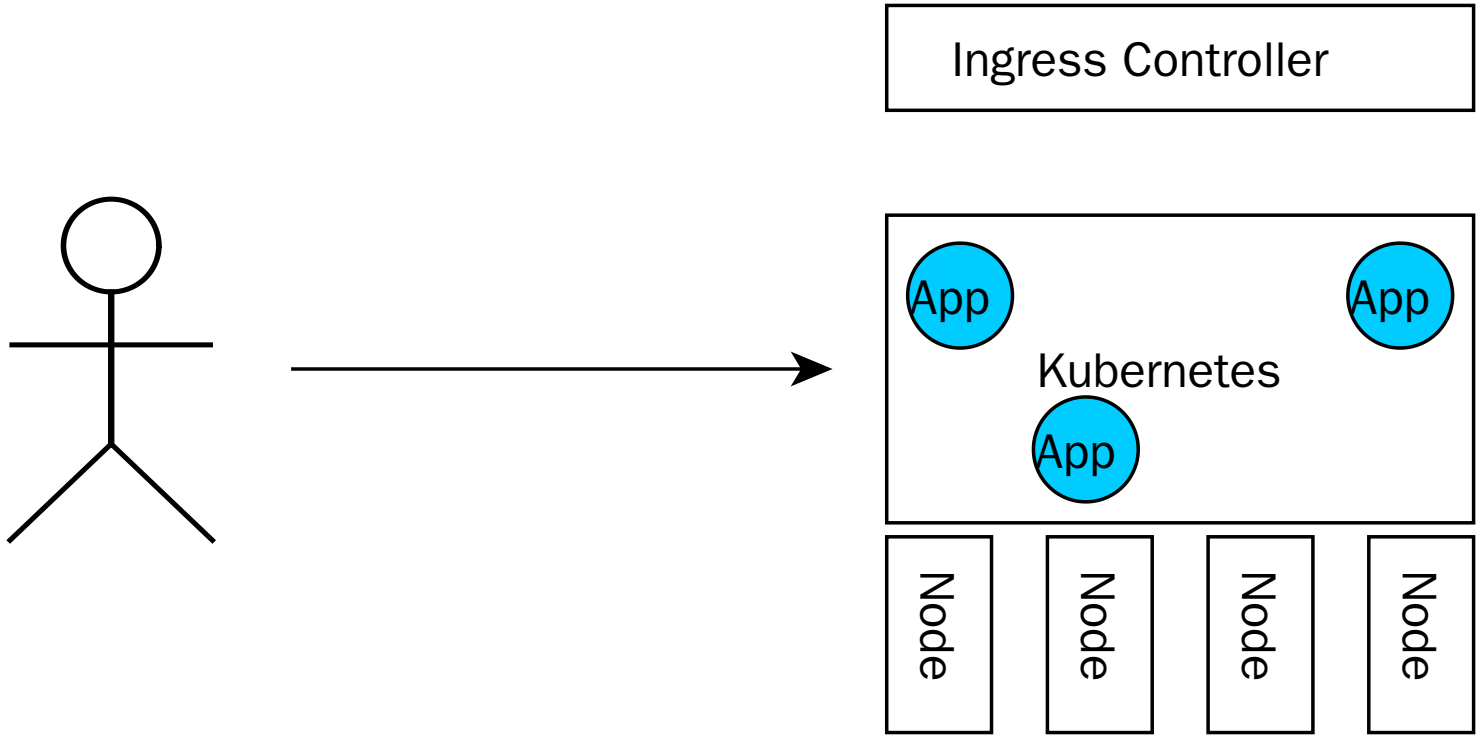

```
kubectl set image deployment/app app=app:v2.0.0
```

ROLLING UPDATES!



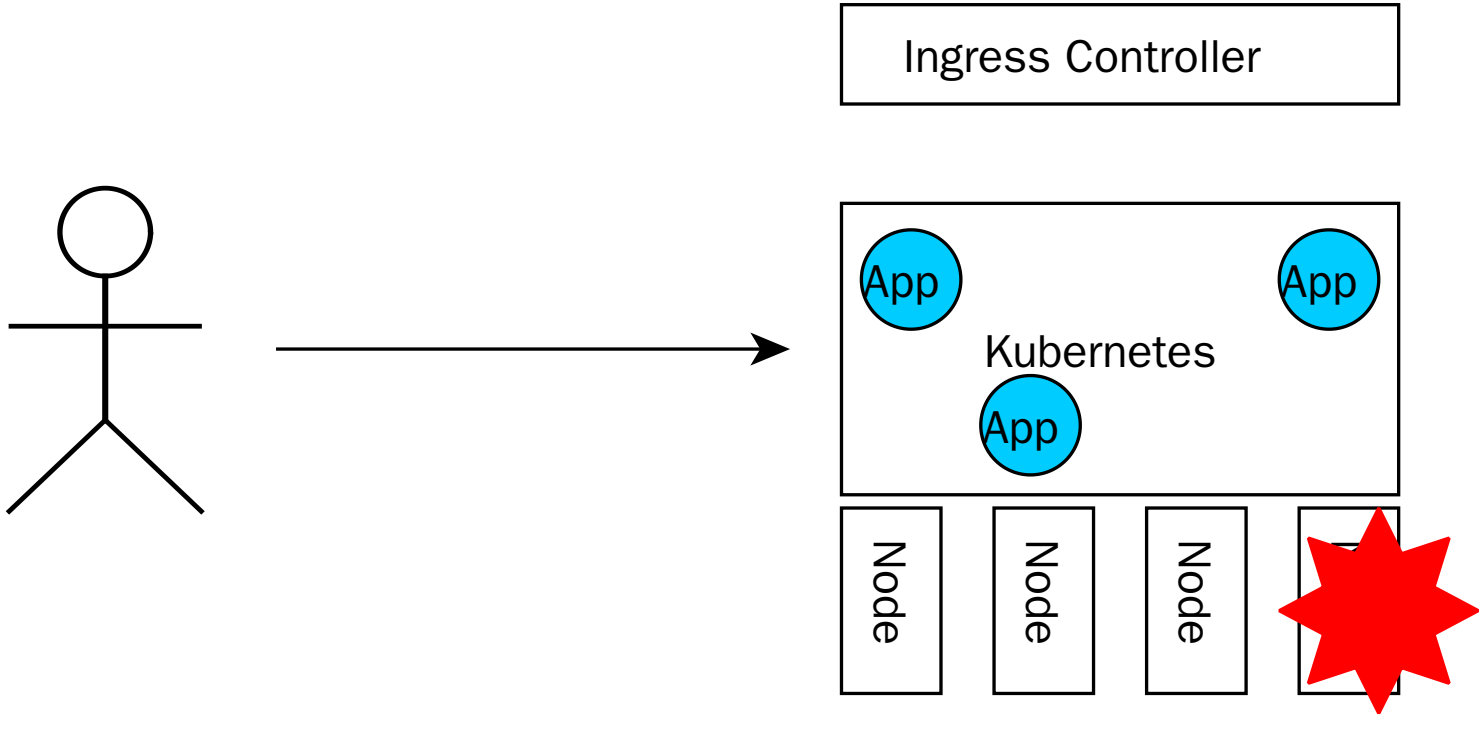

```
kubectl set image deployment/app app=app:v2.0.0
```

ROLLING UPDATES!

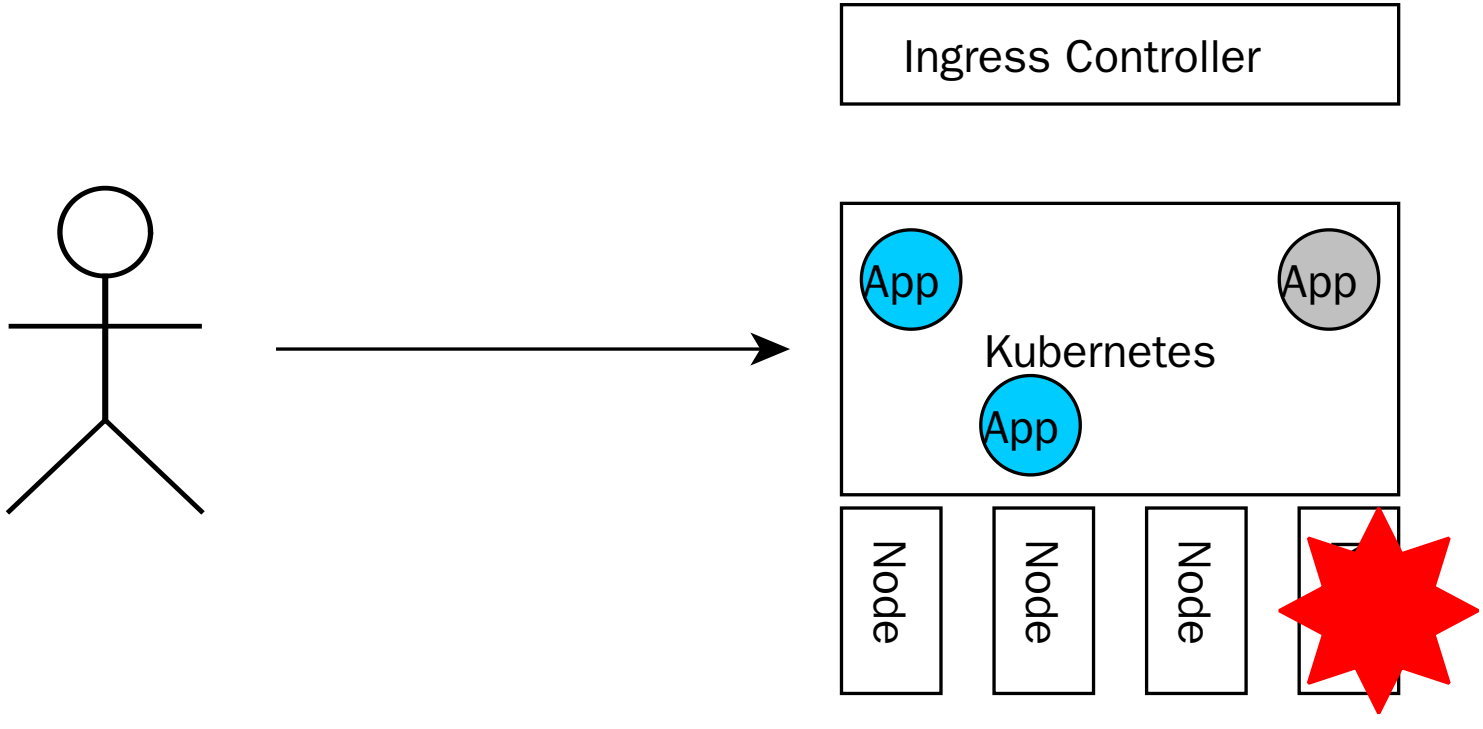



```
kubectl set image deployment/app app=app:v2.0.0
```

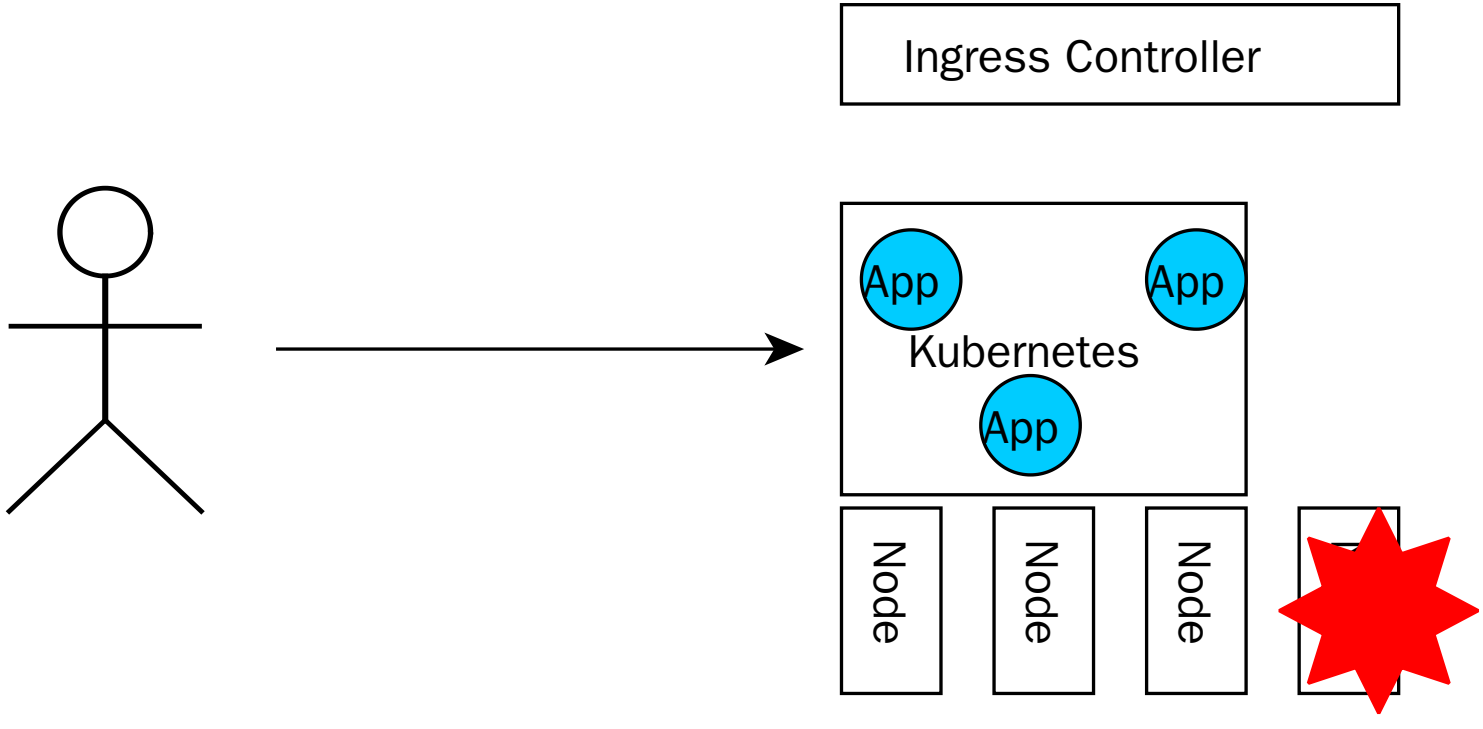
RESISTANCE!



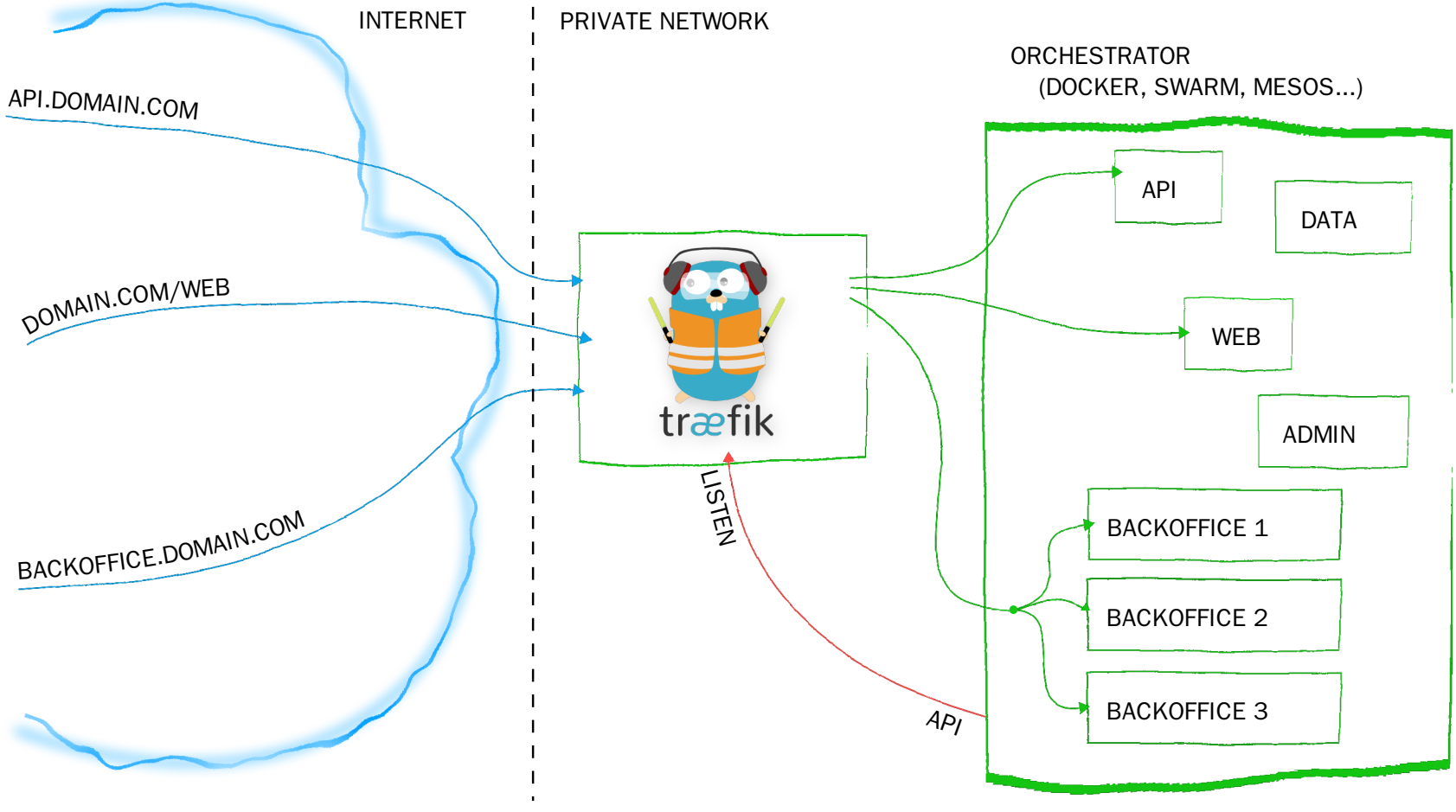
RESISTANCE!



RESISTANCE!



HOW GET USER REQUESTS?

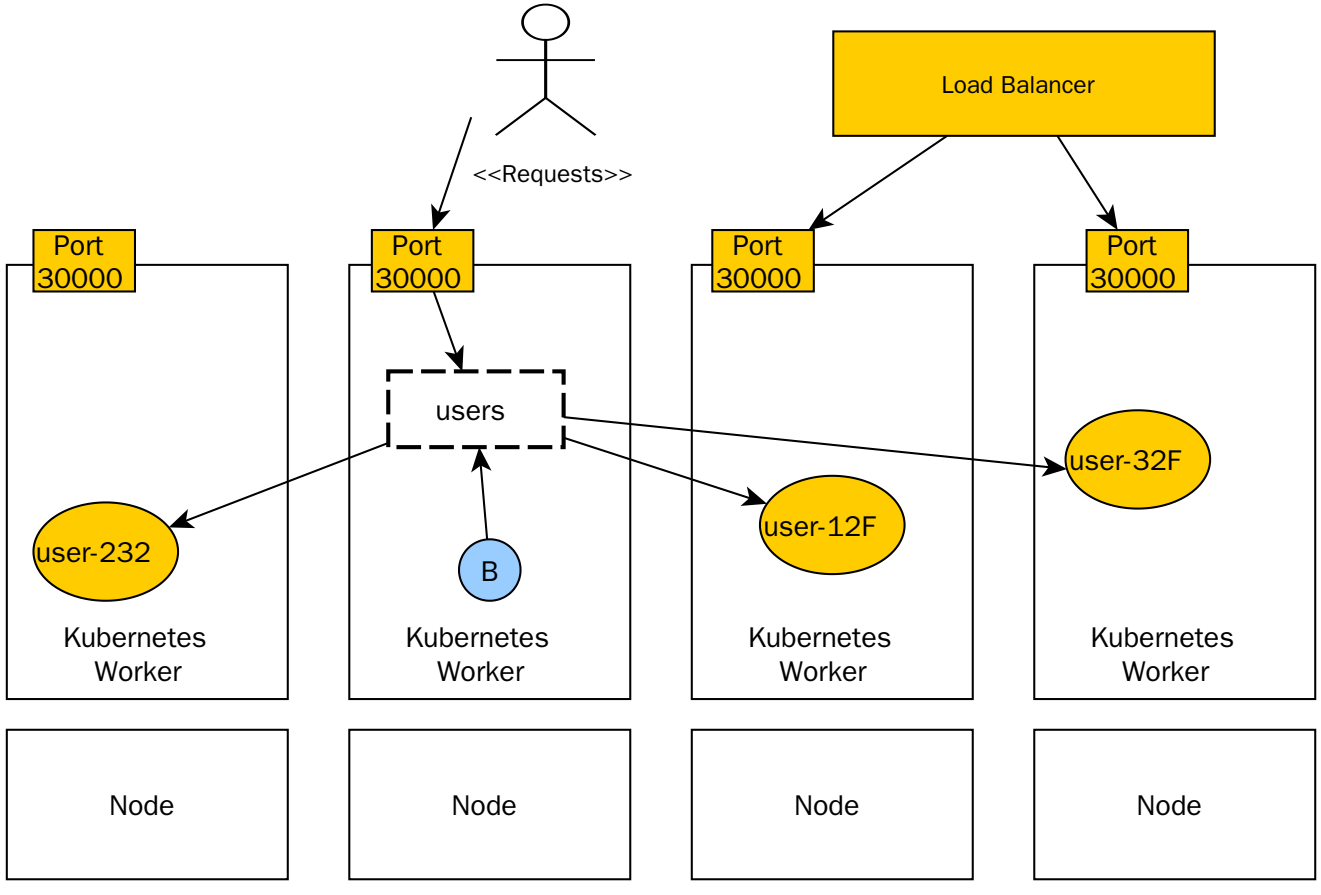


Ingress Controller

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"
  ```

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!

DEPLOYMENT STRATEGIES

STRATEGIES

We will see:

- Replace (downtime visible)
- Rolling updates
- Blue Green
- Canary

OTHER

We will not cover:

- Feature toggles
- A/B like
- Shadow deployment

FIRST THE HOMEWORK

Need to support:

- liveness - am I dead?
- readiness - can I serve requests?

KUBE LIVENESS PROBE

```
livenessProbe:
  httpGet:
    path: /model
    port: 8000
    httpHeaders:
      - name: X-Custom-Header
        value: Awesome
  initialDelaySeconds: 600
  periodSeconds: 5
  timeoutSeconds: 18
  successThreshold: 1
  failureThreshold: 3
```

LIVENESS PROBE

- our pod gets restarted
- too many restarts -> CrashLoop

KUBE READINESS PROBE

```
readinessProbe:
  exec:
    command:
    - cat
     - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 5
```

YOUR APP SHOULD ON STOP

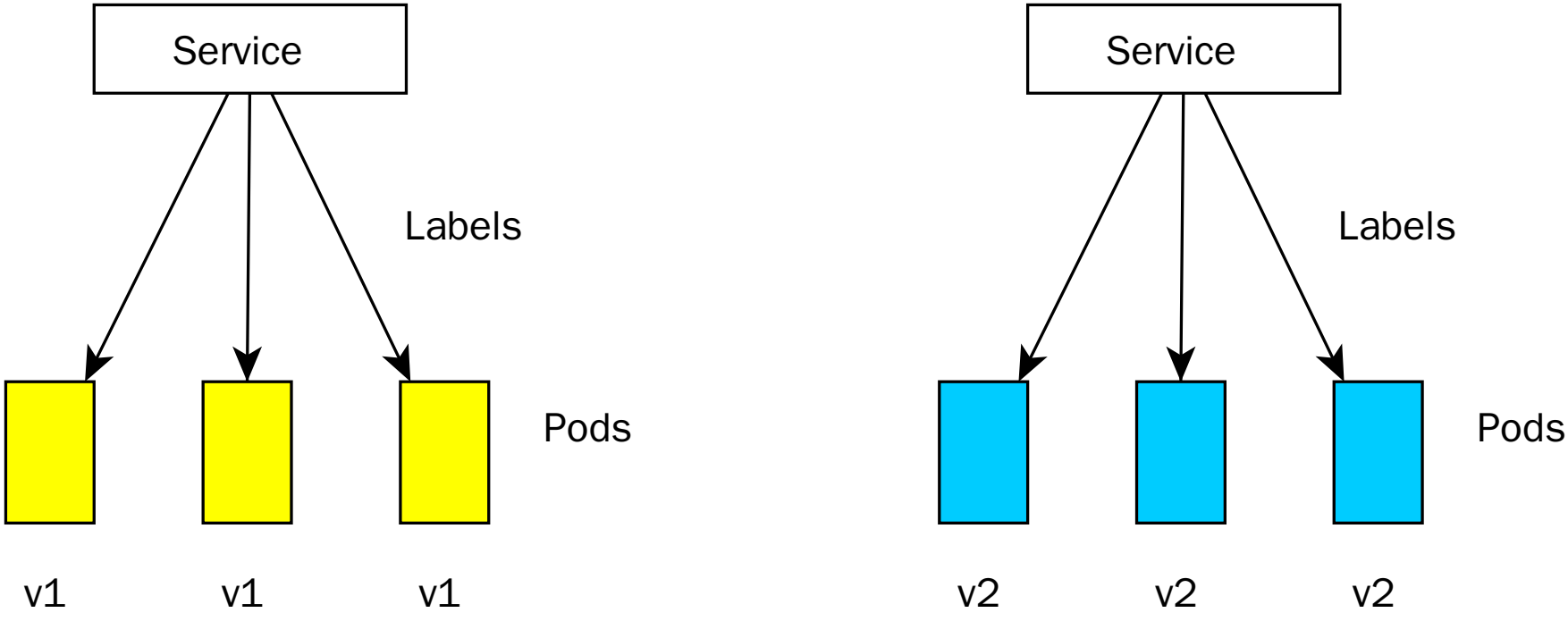
- when we get SIGTERM signal
- app gives 500 on readinessProbe
- app does not receive new requests
- app graceful shutdown
- kuberenetes forces kill if 30 limit exceeded

ALWAYS

Implement readiness for:

- ML Model-based components
- slow starting time

DEMO - RECREATE



DEMO - RECREATE

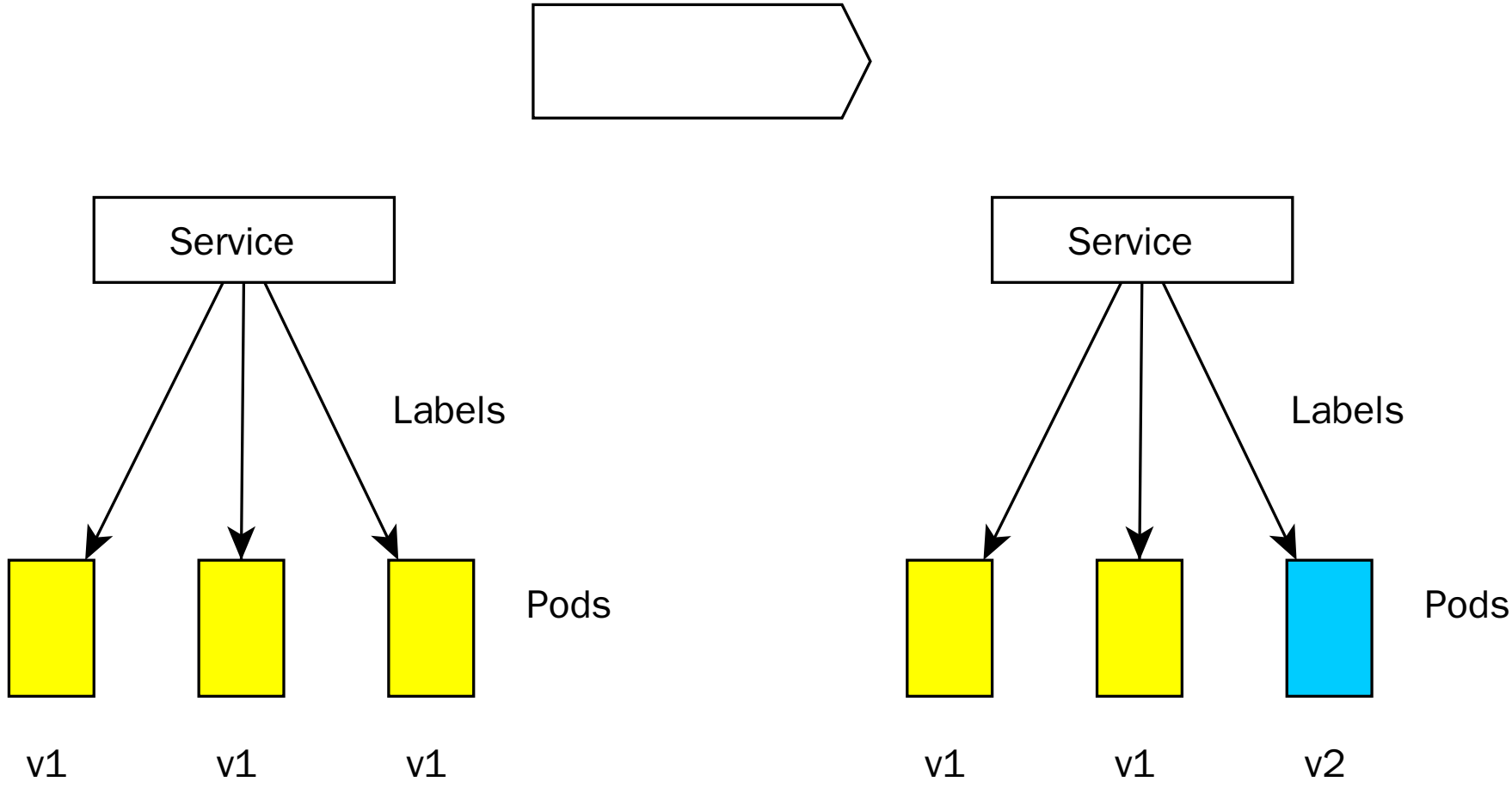
```
spec:
  replicas: 3
  strategy:
    type: Recreate
```

```
kubectl set image  deployment/demo-api \
    app=wojciech11/api-status:2.0.0
```

DEMO - RECREATE

- quick
- downtime visible

DEMO - ROLLING UPDATES



DEMO - ROLLING UPDATES

```
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 2
    maxUnavailable: 0
```

docs

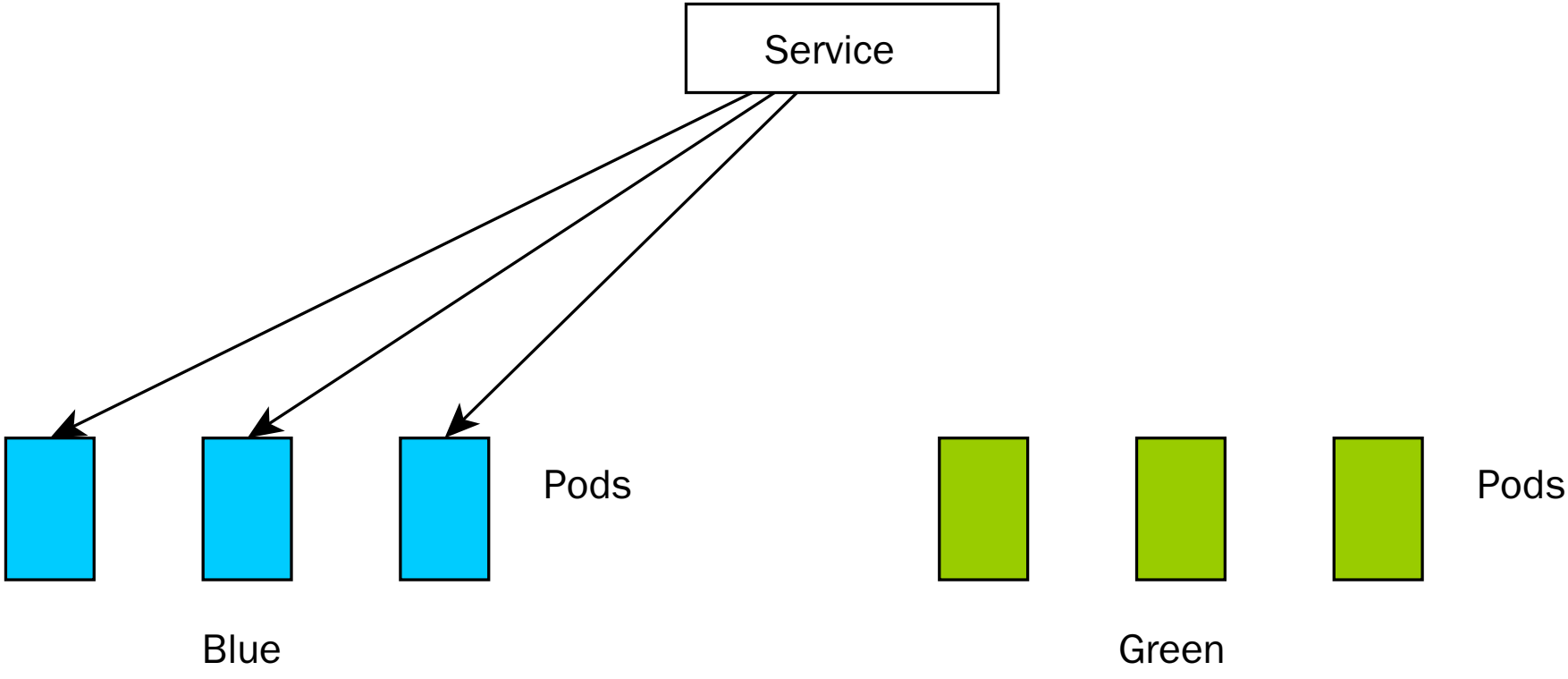
DEMO - ROLLING UPDATES

```
kubectl set image  deployment/demo-api
    app=wojciech11/api-status:2.0.0
```

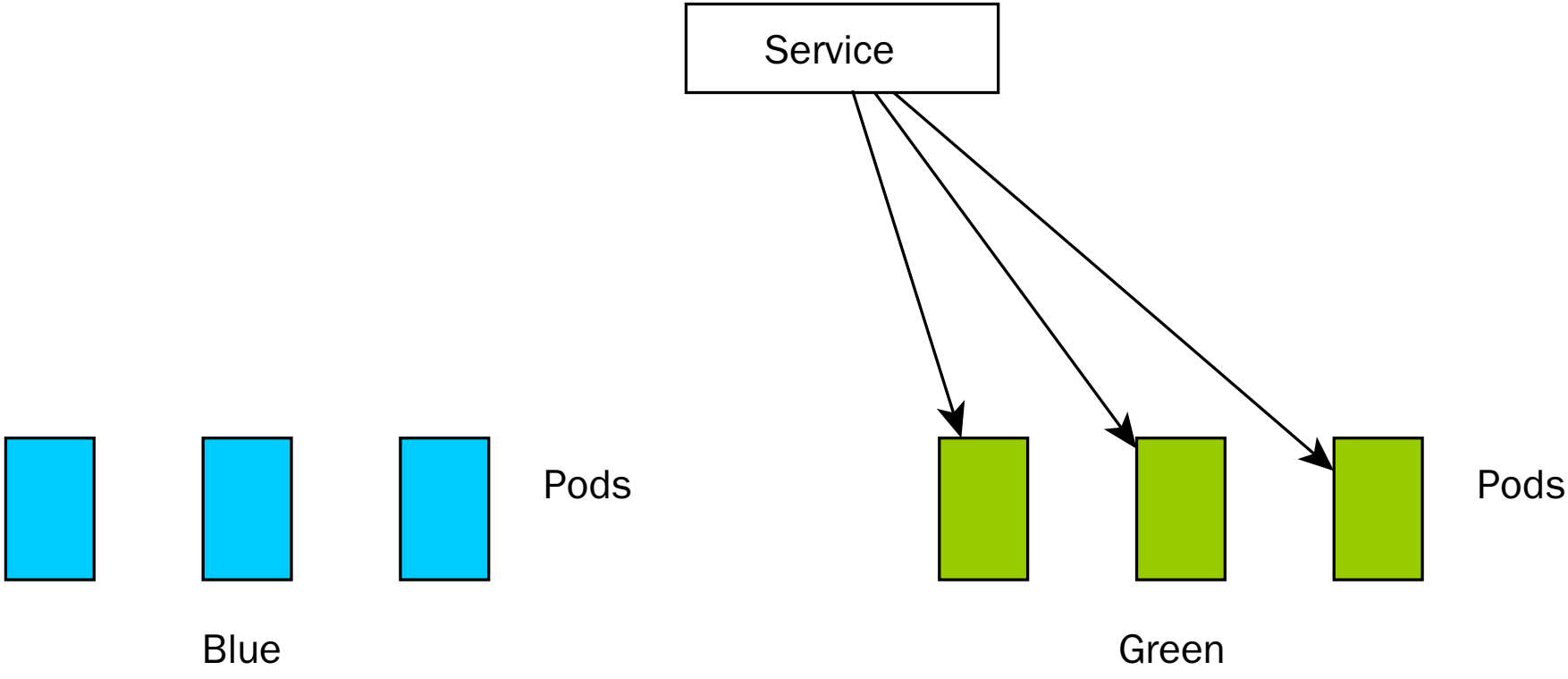
DEMO - ROLLING UPDATES

- the most popular

DEMO - GREEN/BLUE



DEMO - GREEN/BLUE



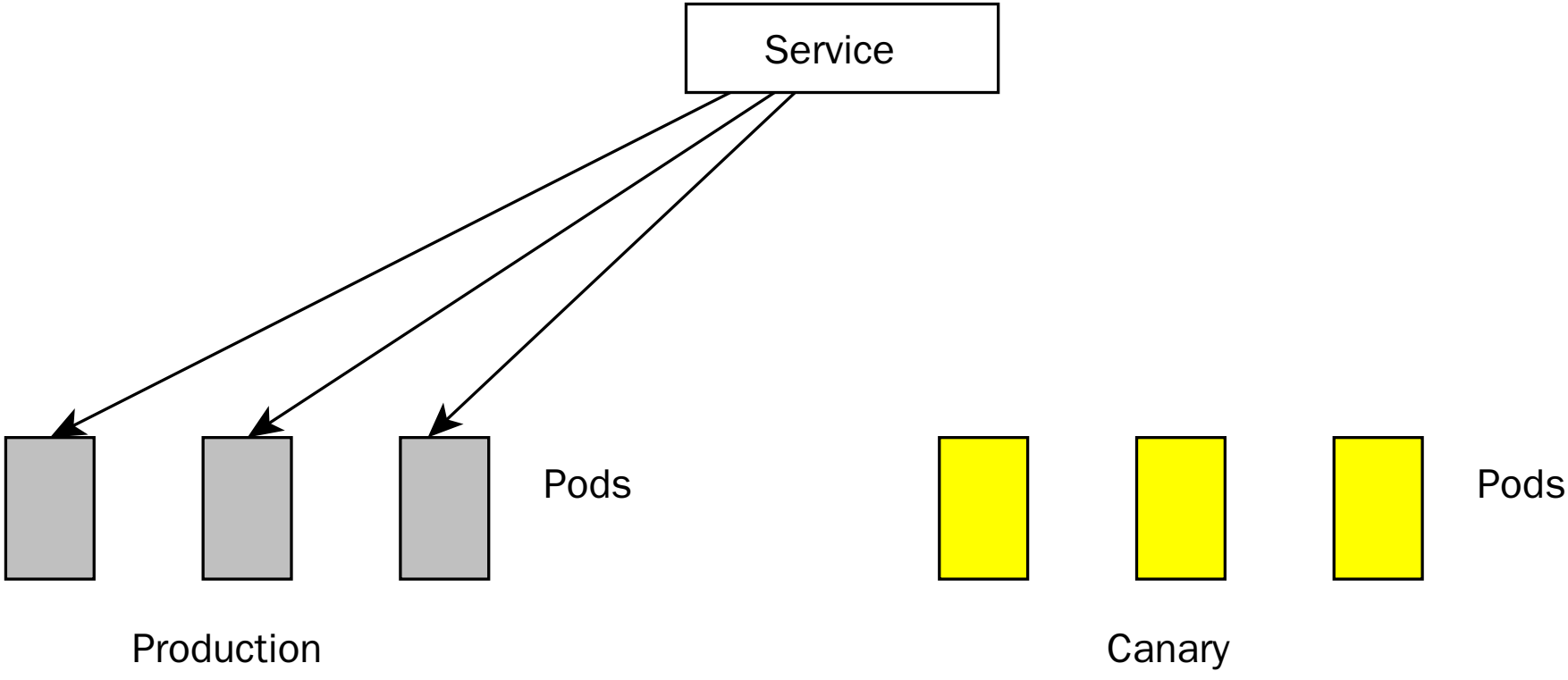
DEMO - GREEN/BLUE

```
kubectl patch service api-status \
    -p '{"spec":{"selector": {"label": "green"} }}'
```

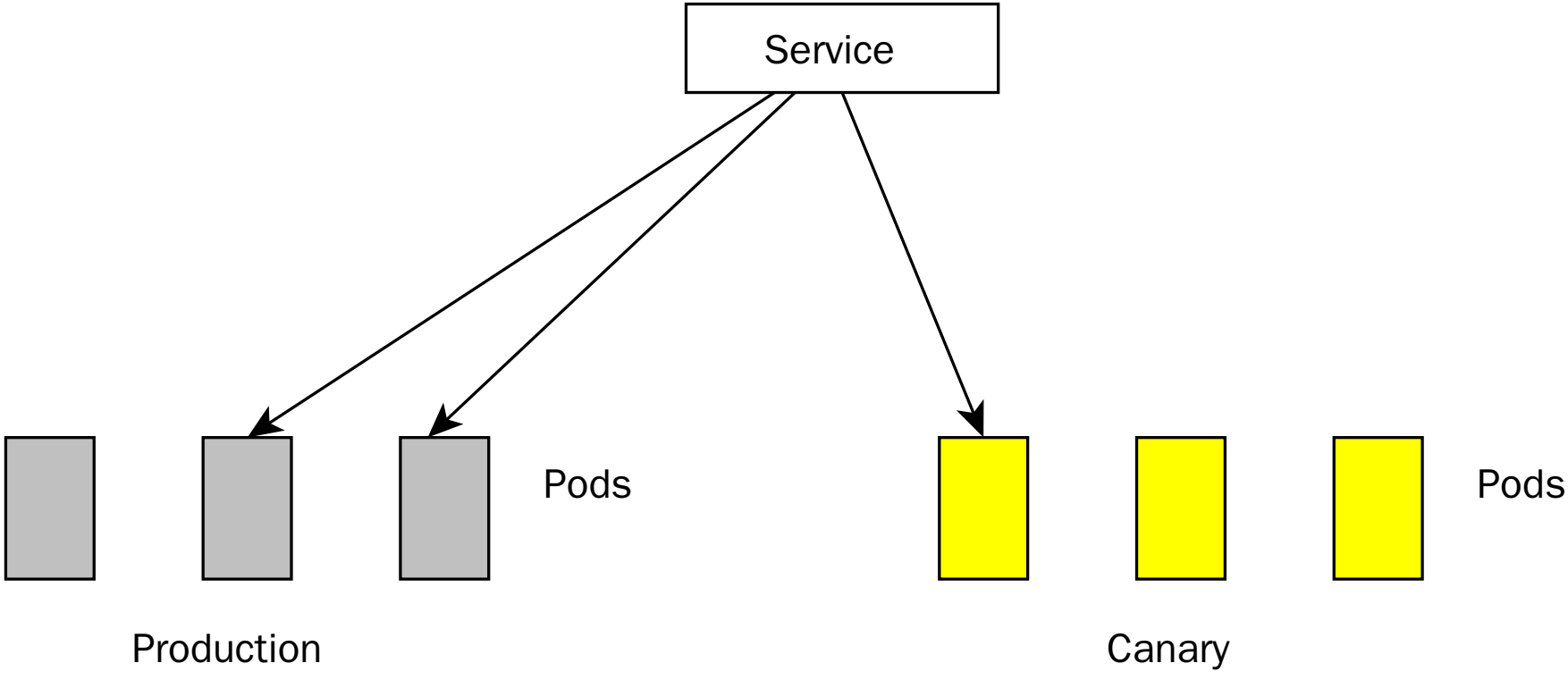
DEMO - GREEN/BLUE

- For big changes
- Less common
- Might be implemented with *Ingress*

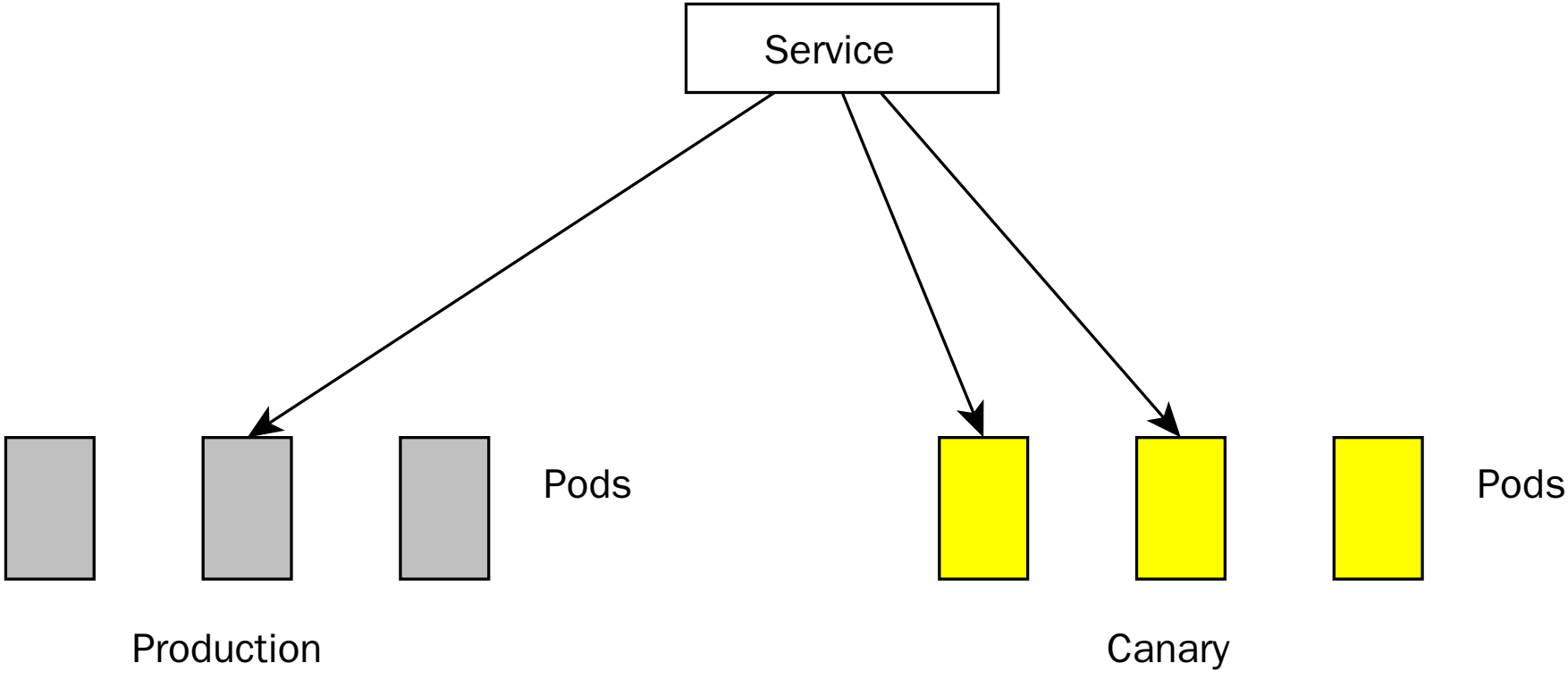
DEMO - CANARY



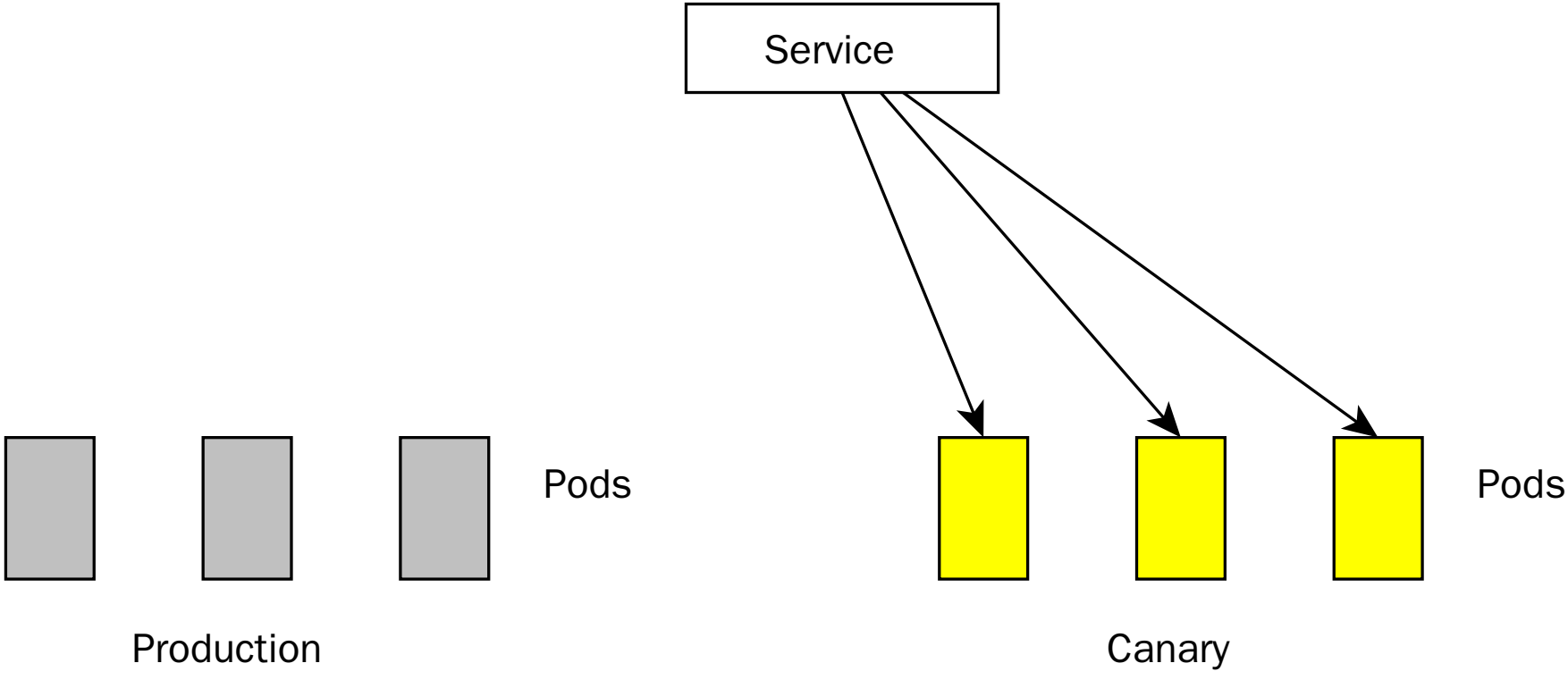
DEMO - CANARY



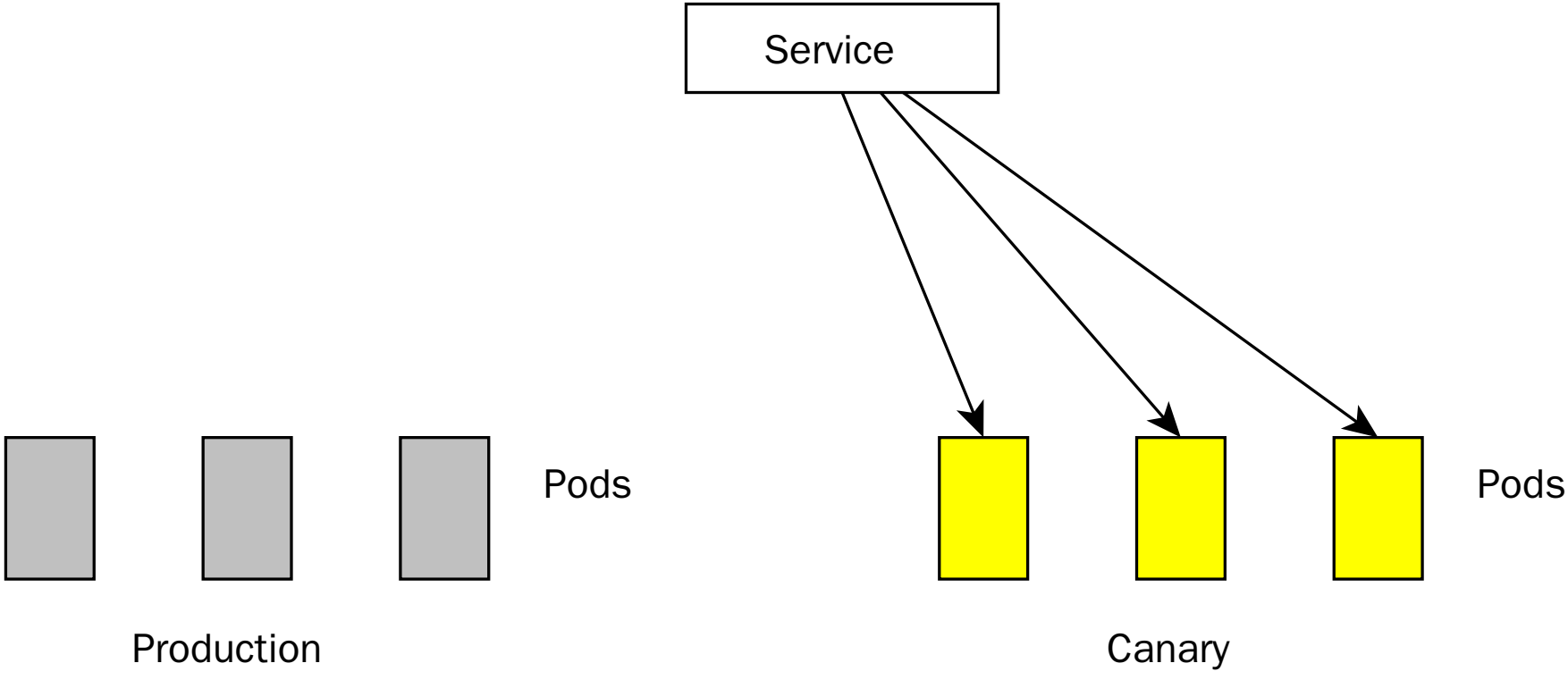
DEMO - CANARY



DEMO - CANARY



DEMO - CANARY



```
kubectl scale --replicas=3 deploy/api-status-nginx-blue
kubectl scale --replicas=1 deploy/api-status-nginx-green

# no errors, let's continoue
kubectl scale --replicas=2 deploy/api-status-nginx-blue
kubectl scale --replicas=2 deploy/api-status-nginx-green
```

DEMO - CANARY

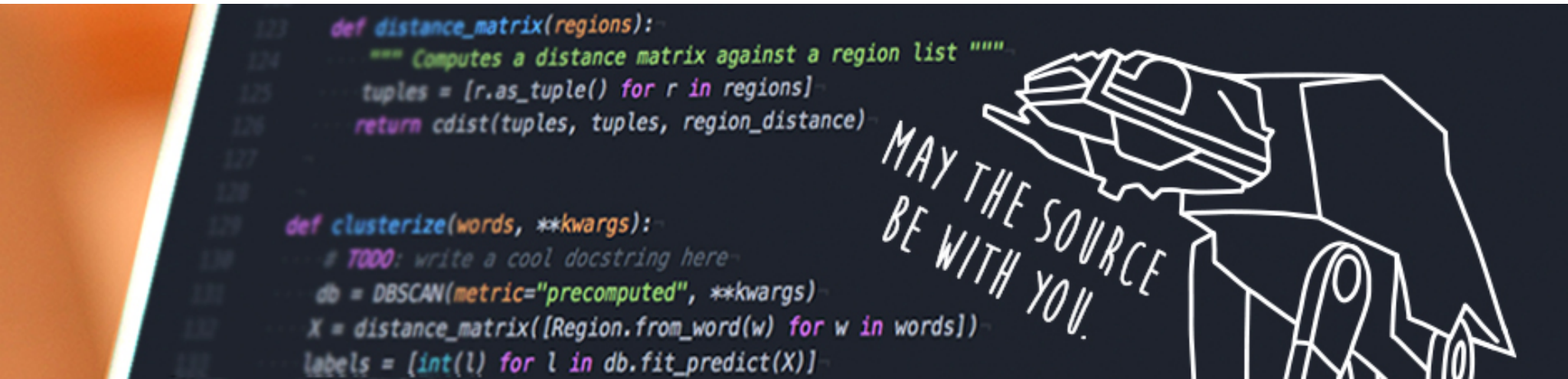
- manually
- with help of Traefik / Istio / ...

SUMMARY

- kubernetes simple semantic
- easy deployment of your applications
- will work for any application type

DZIĘKUJĘ. PYTANIA?

ps. We are hiring.



SMACC

Go

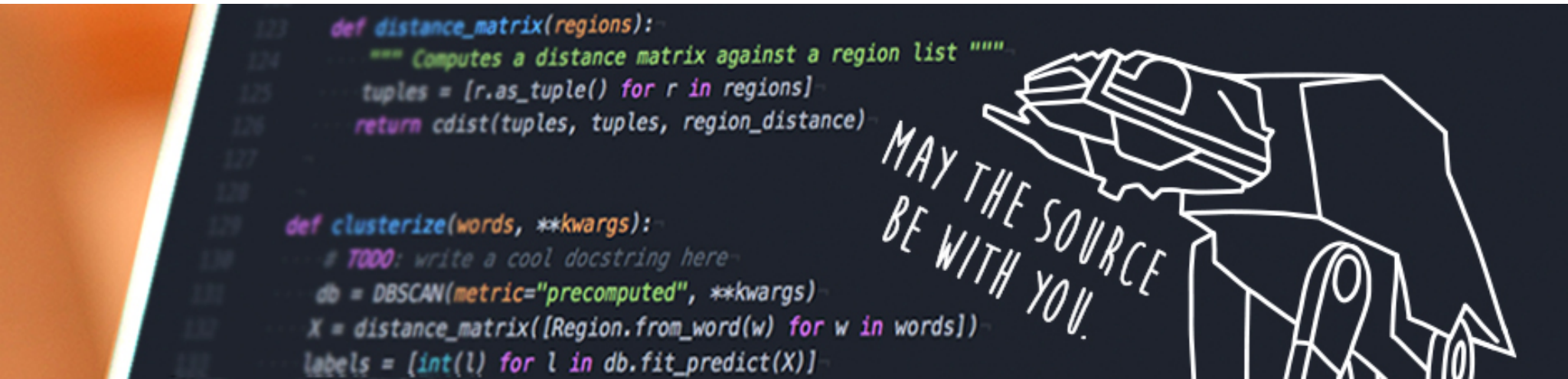
PYTORCH

TensorFlow ™

amazon web services™

Azure

BACKUP SLIDES



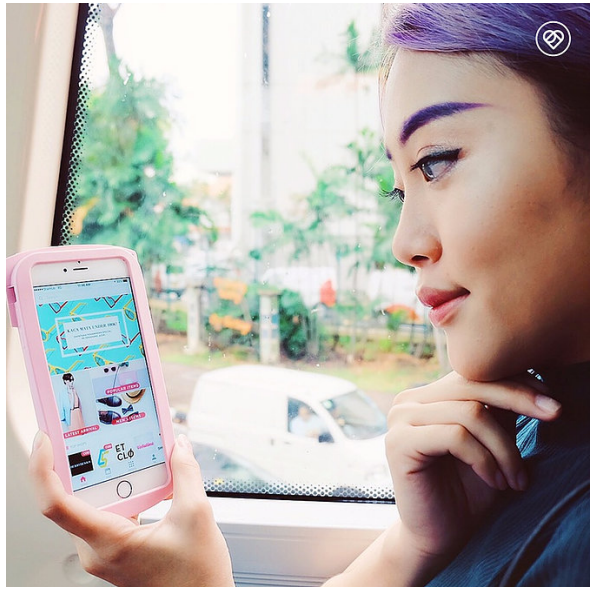
STORY

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

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


SMACC
Hypatos

SMACC

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

STORY

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

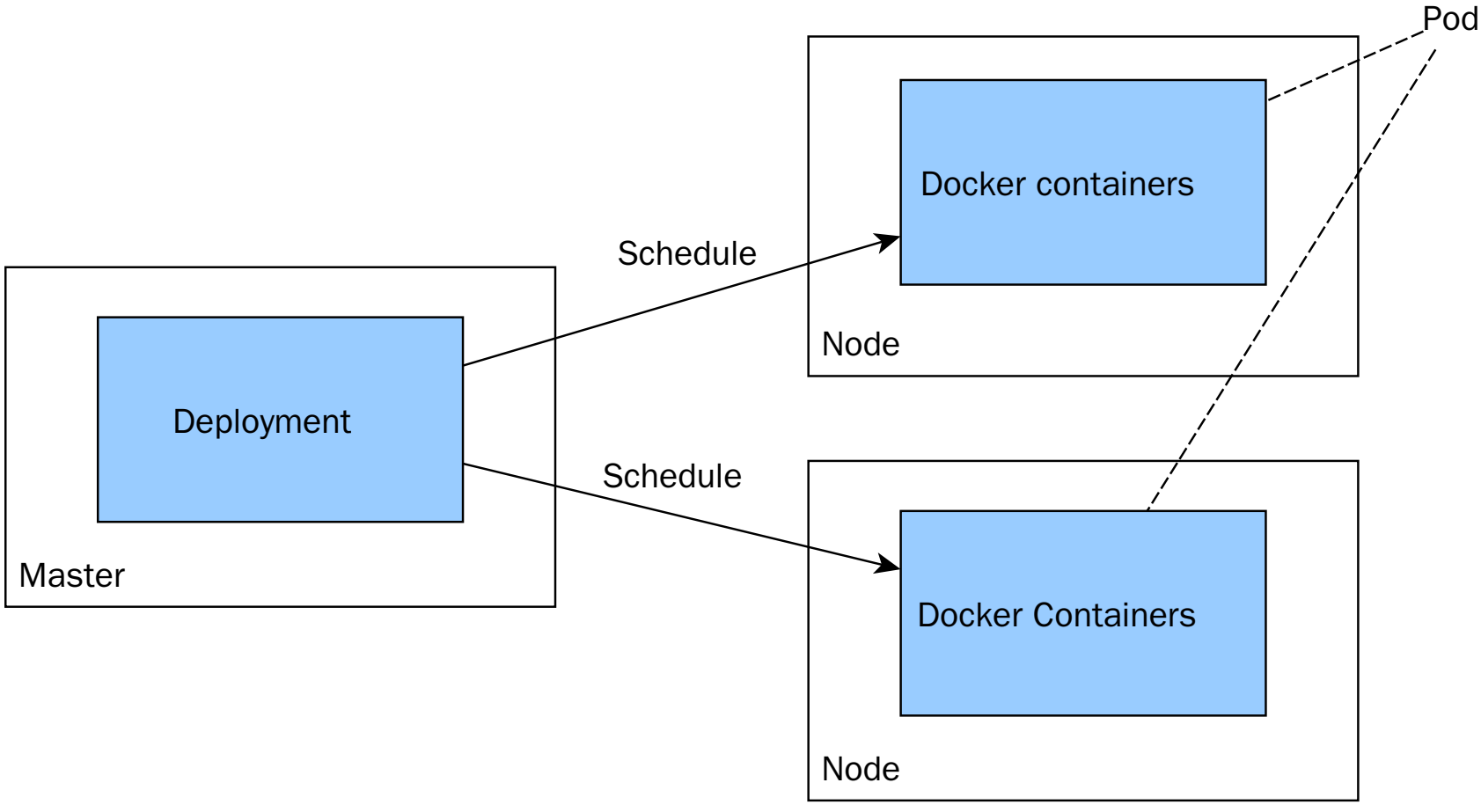
DIFFERENCE ☠

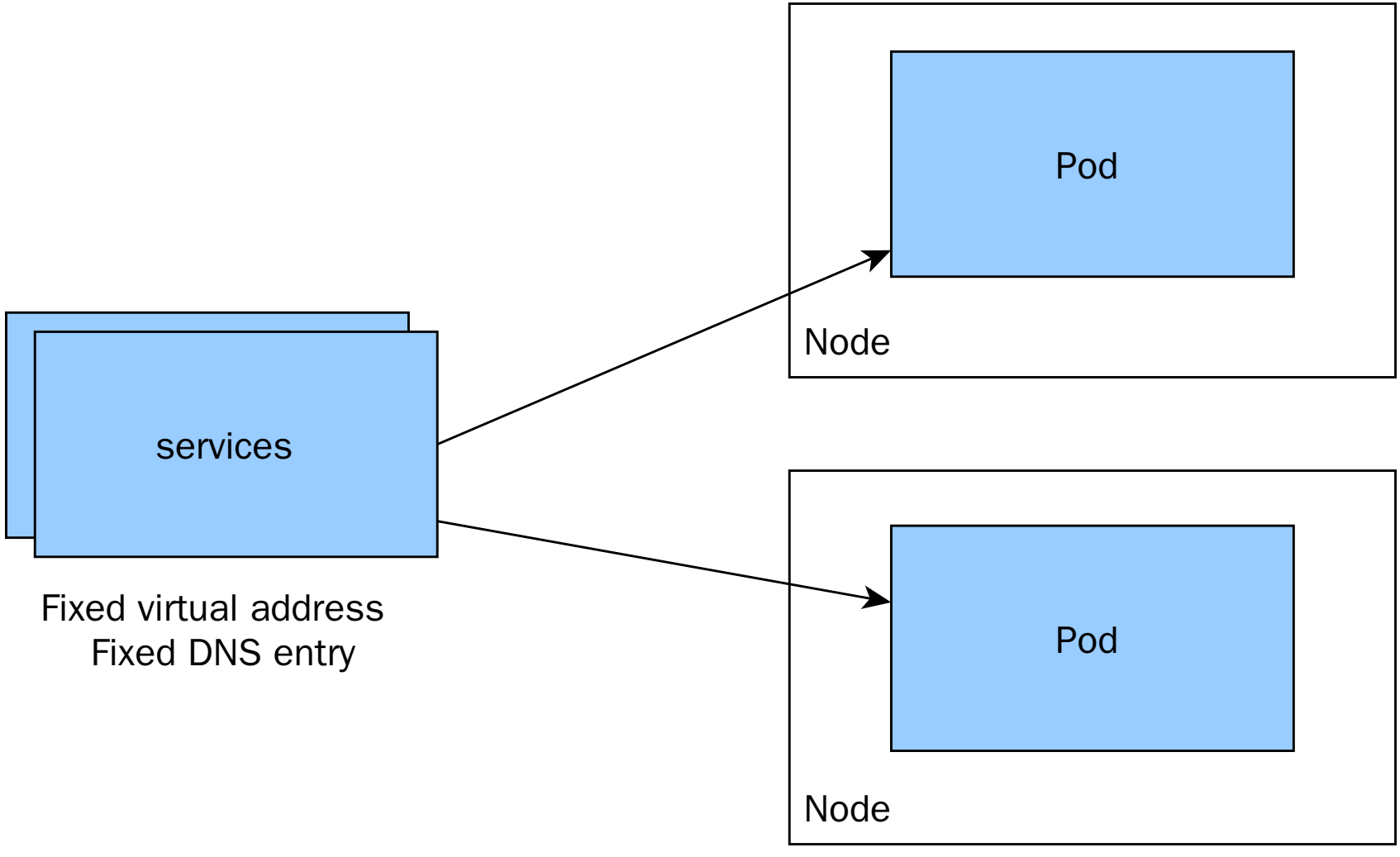
- Two teams in Berlin and Warsaw
- Me in Warsaw

APPROACH

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

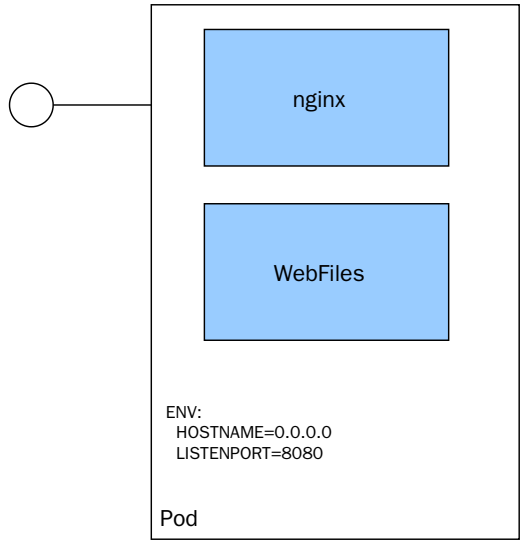
KUBERNETES CONCEPTS


Pod
Docker containers
Schedule
Node
Deployment
Schedule
Master
Docker Containers
Node

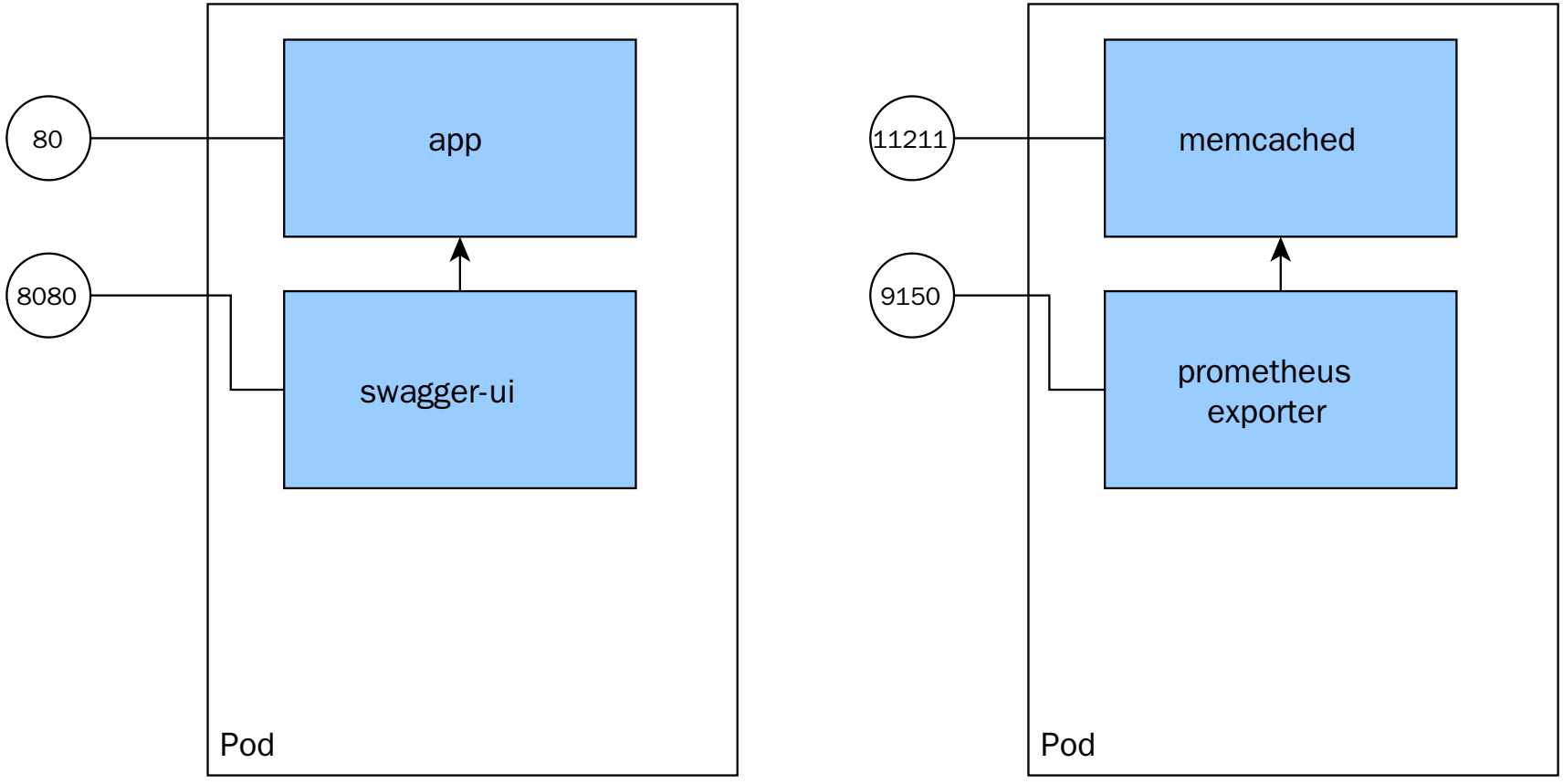

Pod
Node
services
Fixed virtual address
Fixed DNS entry
Pod
Node

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

BASIC CONCEPTS

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