
DrupalEurope
Darmstadt, Germany
Sep 10 - 14, 2018

Drupal Europe
Darmstadt, Germany
10 - 14 September 2018

DevOps + Infrastructure

TRACK SUPPORTED BY

About me

- Nils Peeters
- DevOps Engineer
- nils@scalecity.io
- https://www.linkedin.com/in/nilspeeters/
- www.scalecity.io

Containerized Drupal, Kubernetes and blue/green

Down the rabbit hole

Agenda

- Kubernetes
  - What, how and why
- Q&A
- Drupal 8
  - Containerized
- Q&A
- Deploying
  - CI/CD
  - Rolling-update
  - Blue-green
- Q&A

Kubernetes

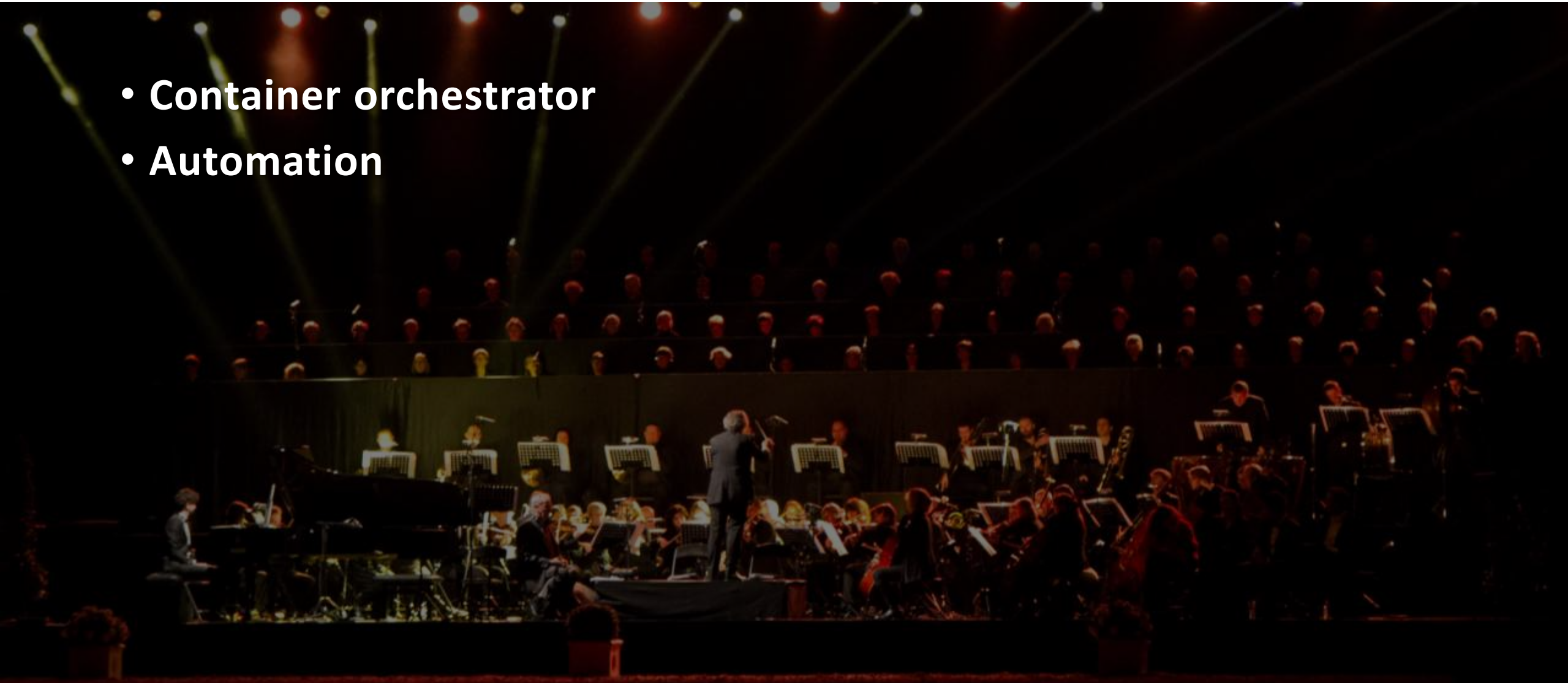
The what, how and why

What is Kubernetes?

- Open source
- Borg

- 70 000 commits
- 1800 contributors
- Google, Microsoft, Red Hat and Huawei
- Cloud Native Computing Foundation
  - https://www.cncf.io/

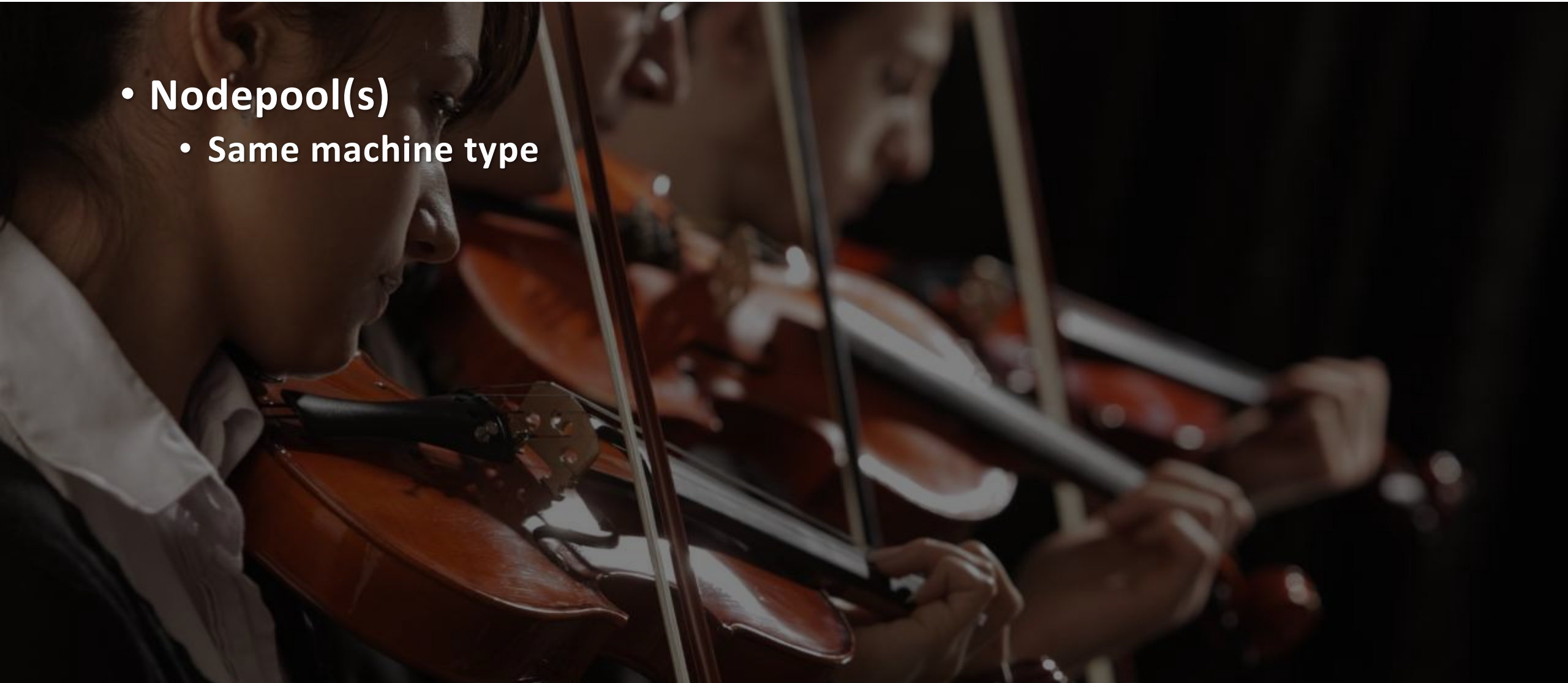
What is Kubernetes?



- **Container orchestrator**
- **Automation**

What is Kubernetes?

What is Kubernetes?



- **Nodepool(s)**
  - **Same machine type**

What is Kubernetes?

- **Node**
  - **“Machine” (VM)**
  - **COS**
  - **Fleeting**
  - **Pods**

SCALECITY

Pods vs Containers

What is this “Pod” you speak of?

Pods vs Containers

Image reference:

Technology X

I’m a container!

Technology X

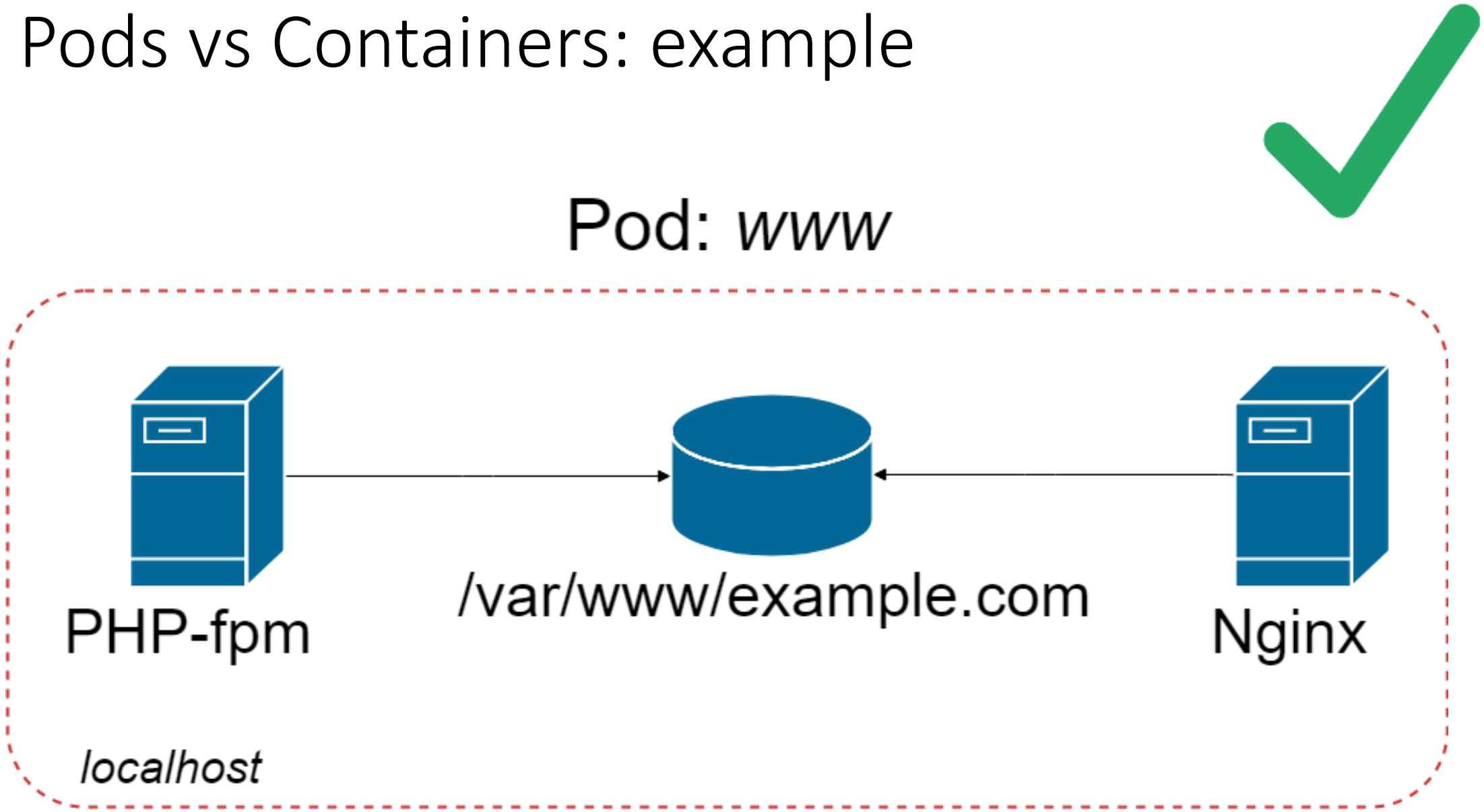
I’m a pod!

Pods vs Containers: example

PHP-fpm

Nginx

Pods vs Containers: example

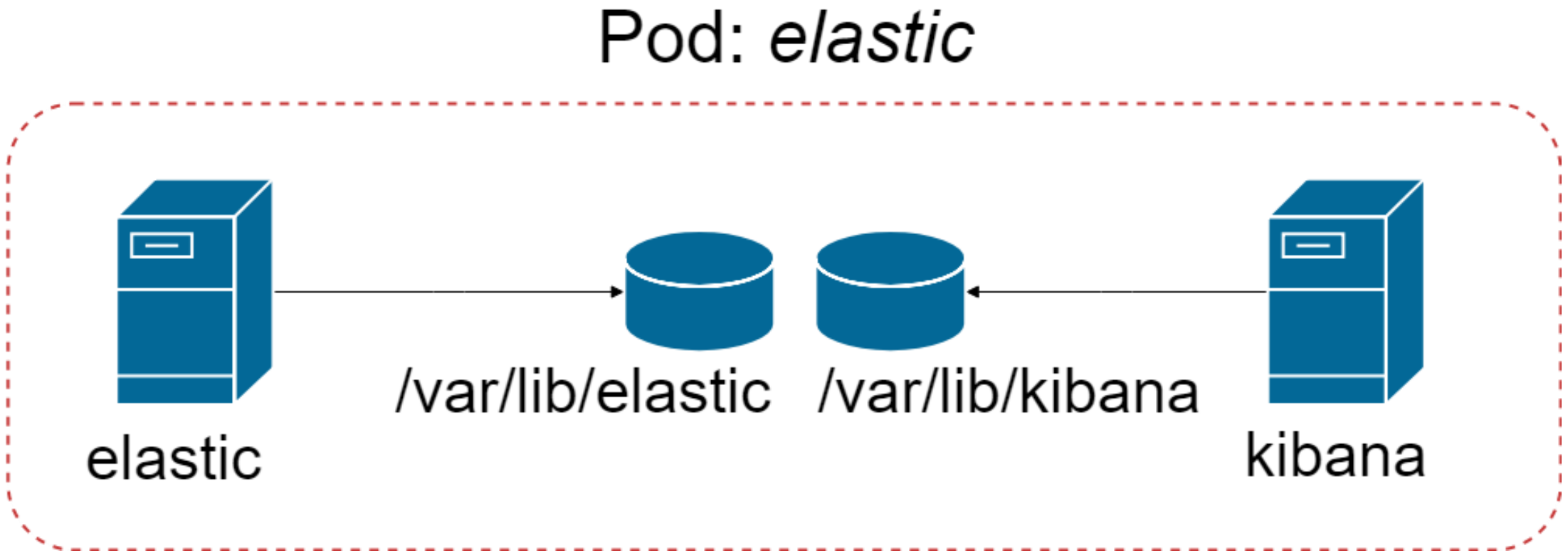


Pods vs Containers: example

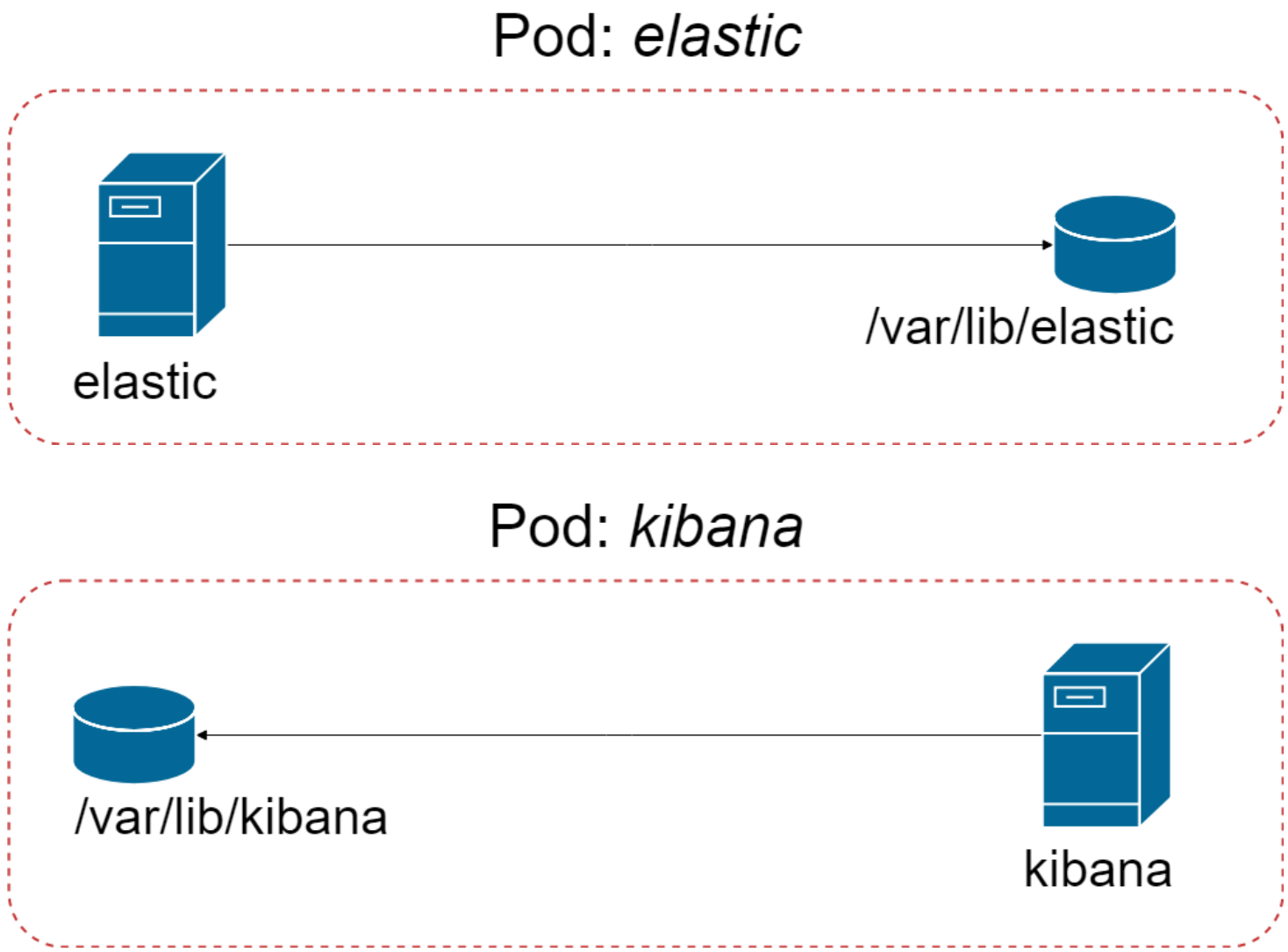
ElasticSearch

Kibana

Pods vs Containers: example



Pods vs Containers: example



Pod vs Containers

```
apiVersion: v1
kind: Pod
metadata:
  name: www
spec:
  containers:
  - name: my-nginx
    image: docker.example.com/nginx:1.14
    ports:
    - containerPort: 8080
 - name: my-php
    image: docker.example.com/php:7.1-fpm
    ports:
    - containerPort: 9000
```

SCALECITY

Services

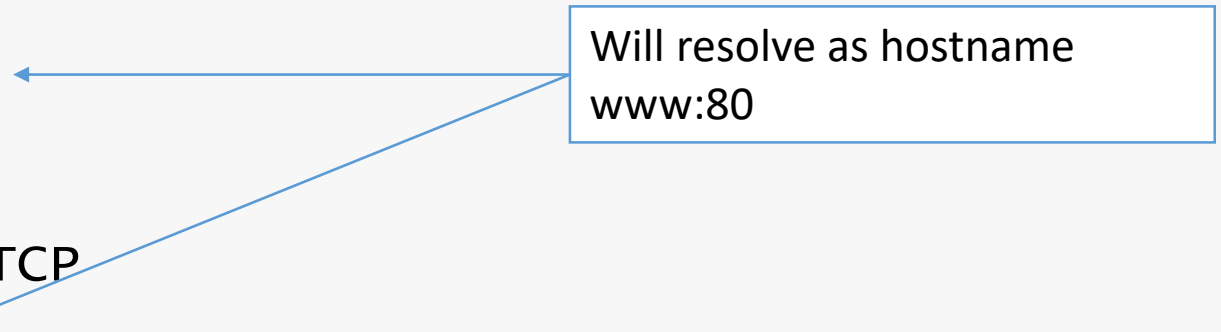
Connecting the Pods

Services

- **Networking component**
- **Labels**
- **Name = hostname**

Services

```
kind: Service
apiVersion: v1
metadata:
  name: www
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
```



SCALECITY

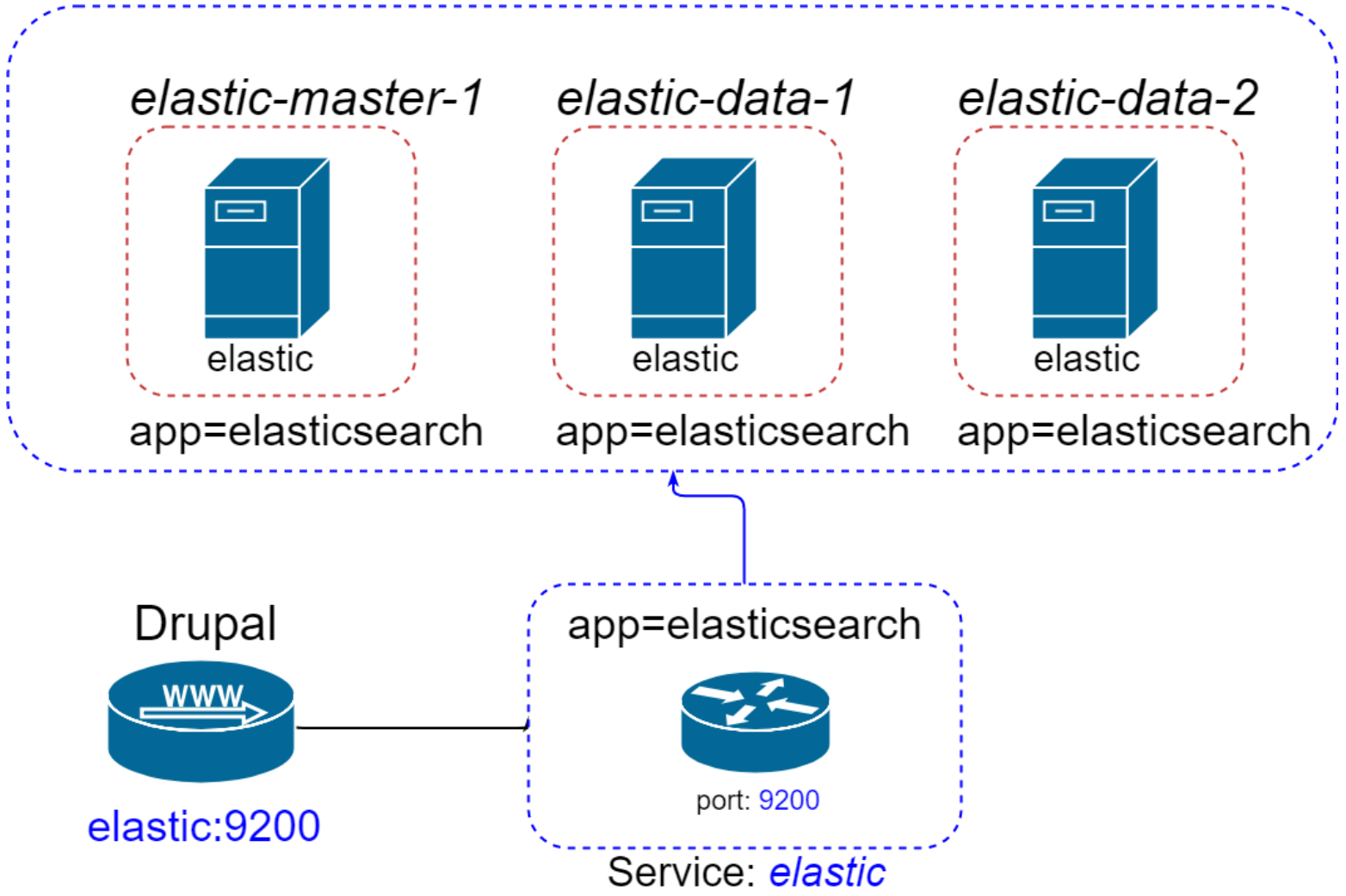
Labels & label selectors

Connecting the dots

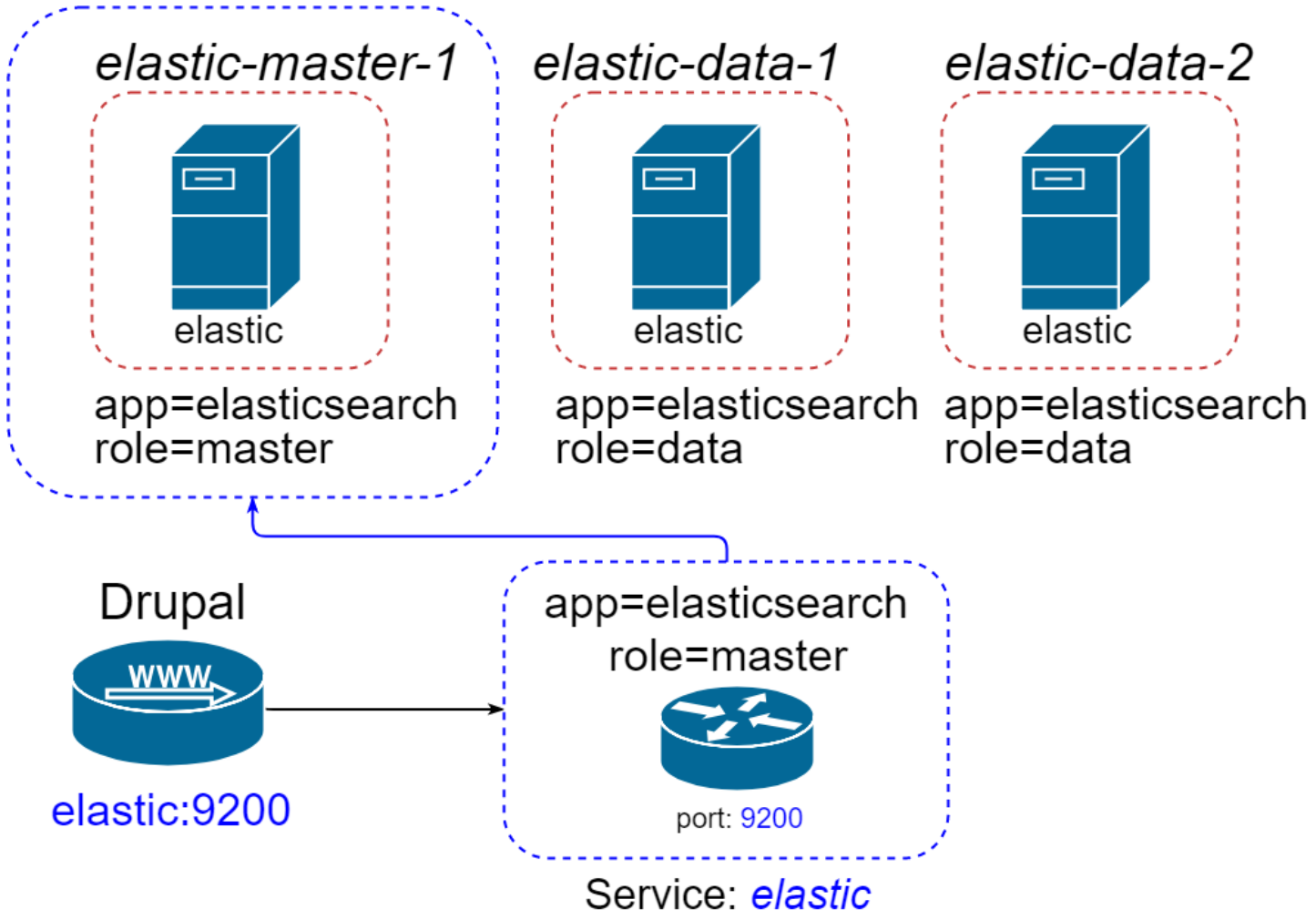
Labels & label selectors

- **Services connect pods**
- **Key value**
- **Get operations by label**

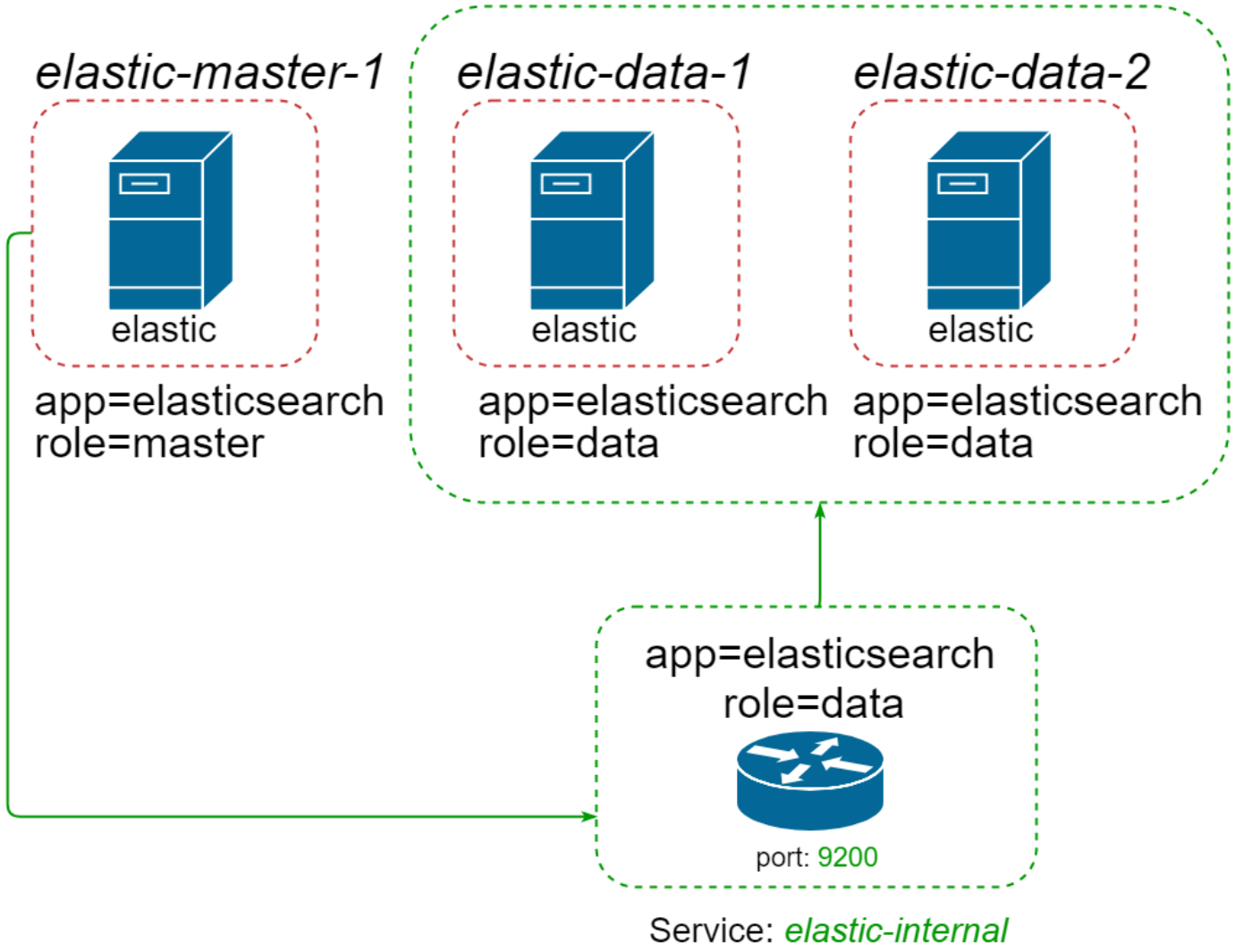
Labels & label selectors



Labels & label selectors



Labels & label selectors

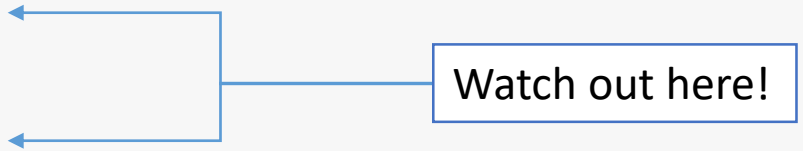


Labels & label selectors

```
apiVersion: v1
kind: Pod
metadata:
  name: elastic
  labels:
    app: elastic
    role: master
spec:
  containers:
  - name: my-apache
    image: docker.example.com/elasticsearch:6.4
    ports:
    - containerPort: 8080
```

Labels & label selectors

```
apiVersion: v1
kind: Service
metadata:
  name: elastic
  labels:
    name: elastic
spec:
  selector:
    app: elastic
    role: master
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
```

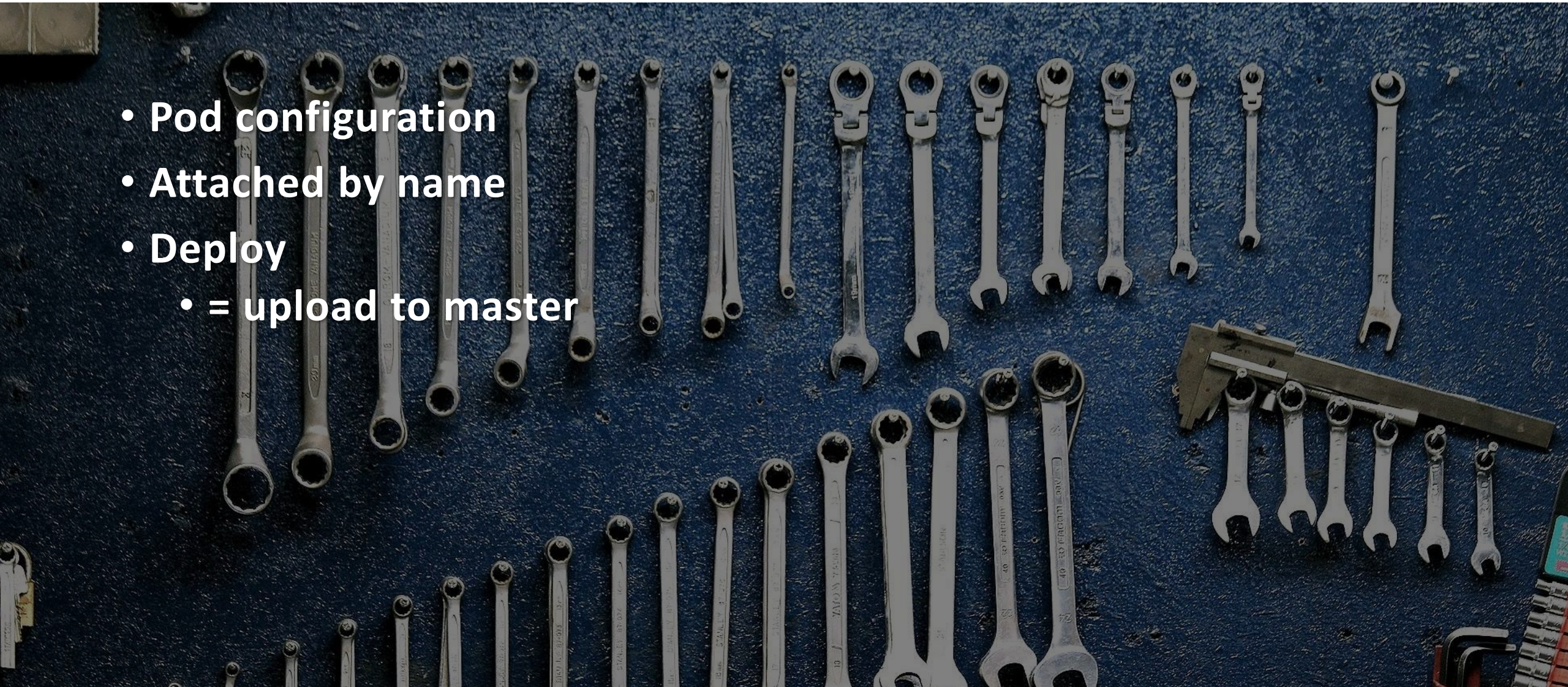


SCALECITY

Configmaps & Secrets

You pods act my way, or the highway

Configmaps



- Pod configuration
- Attached by name
- Deploy
  - = upload to master

Secrets

- **Configmap + encryption**
- **Great for stuff like API keys**

Configmaps / Secrets

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-apache-config
data:
  my.conf: |
    <VirtualHost *:8080>
        DocumentRoot "/www/example1"
        ServerName www.example.com
    </VirtualHost>
```

Configmaps / Secrets

```
apiVersion: v1
kind: Pod
spec:
  containers:
  - name: my-apache
    image: docker.example.com/apache:2.4
    ports:
    - containerPort: 8080
    volumeMounts:
    - name: sites-enabled-config
      mountPath: /etc/apache2/sites-enabled/default.conf
      subPath: my.conf
  volumes:
    - name: sites-enabled-config
      configMap:
        name: my-apache-config
```

SCALECITY

Drupal 8

Containerized

Component breakdown – D8

Container breakdown – D8

PHP-fpm

MySQL

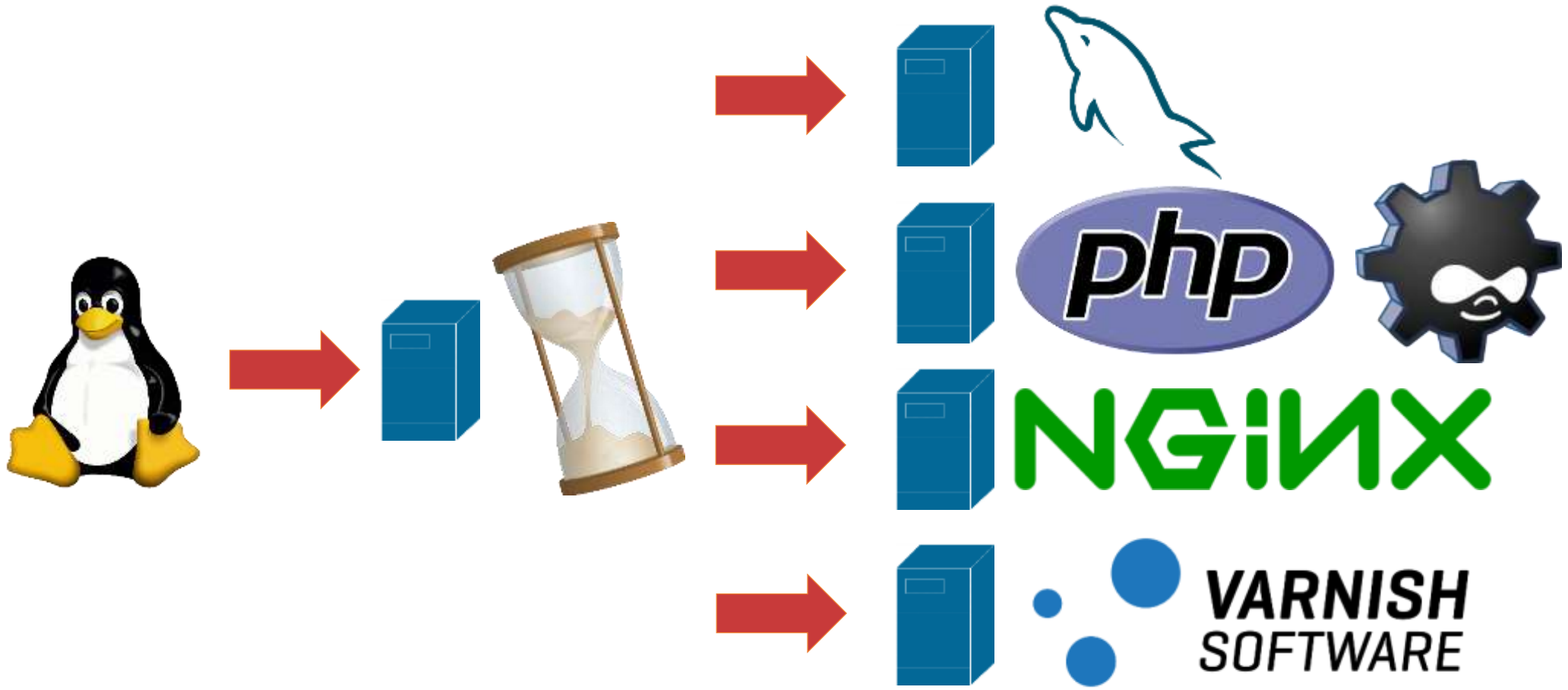
Nginx

Varnish

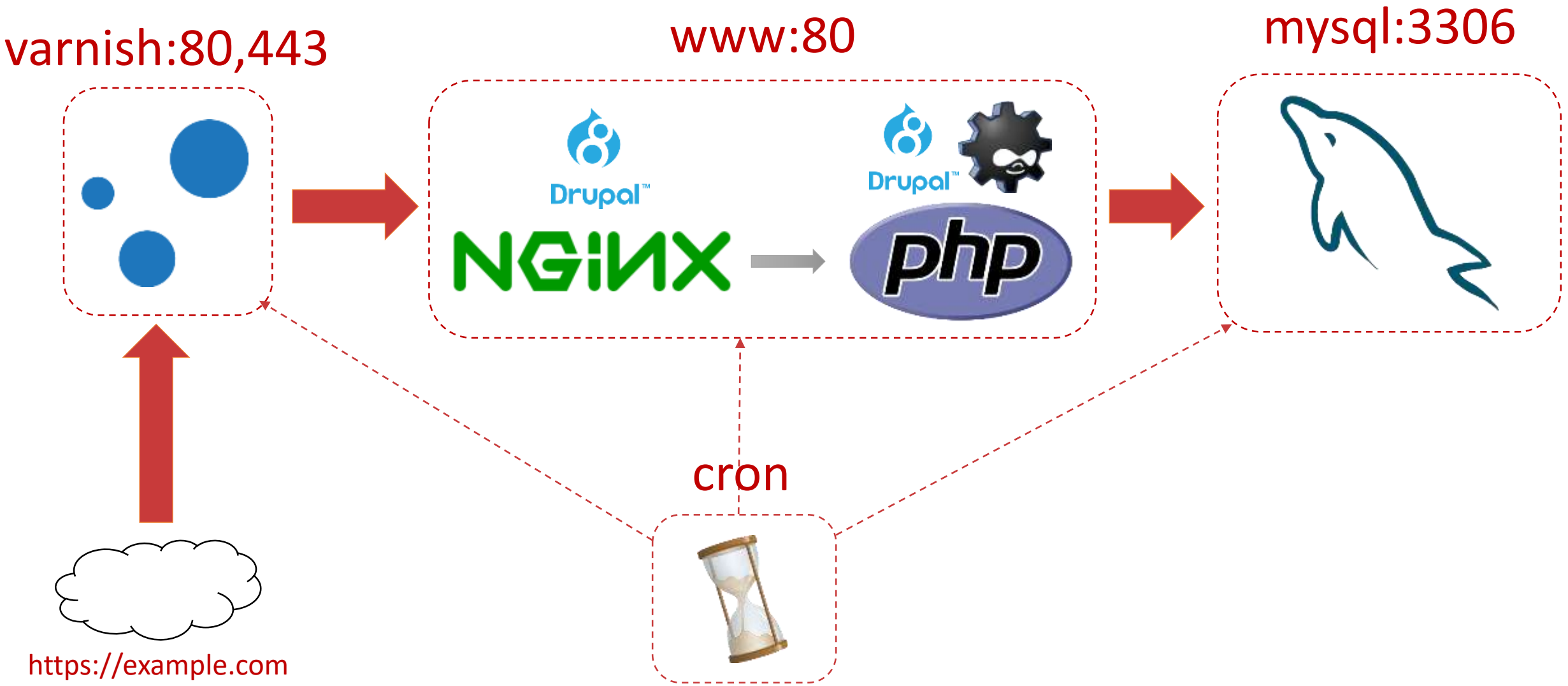
Container breakdown – D8

Linux cron

Container buildup: cron



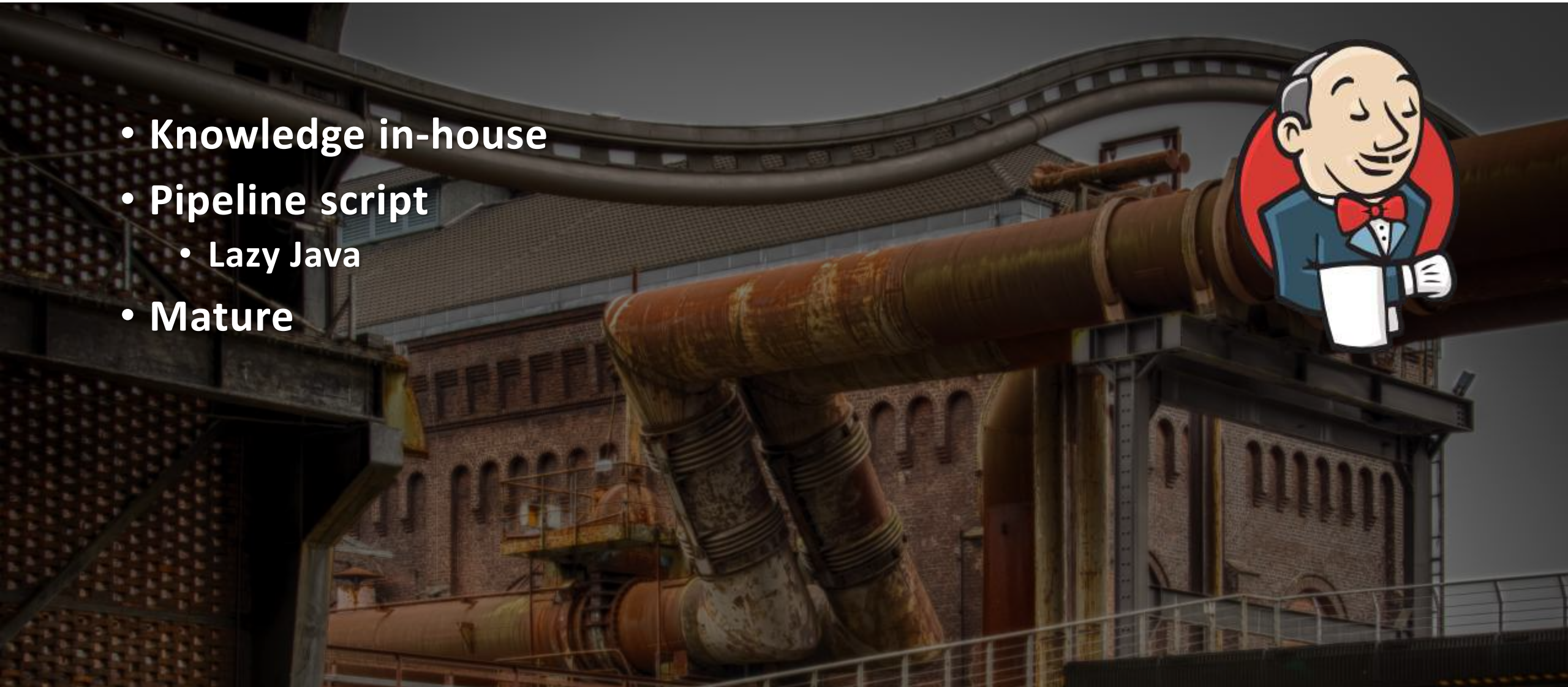
Pod buildup & dataflow



Deploying

All that good stuff

Jenkins



- Knowledge in-house
- Pipeline script
  - Lazy Java
- Mature

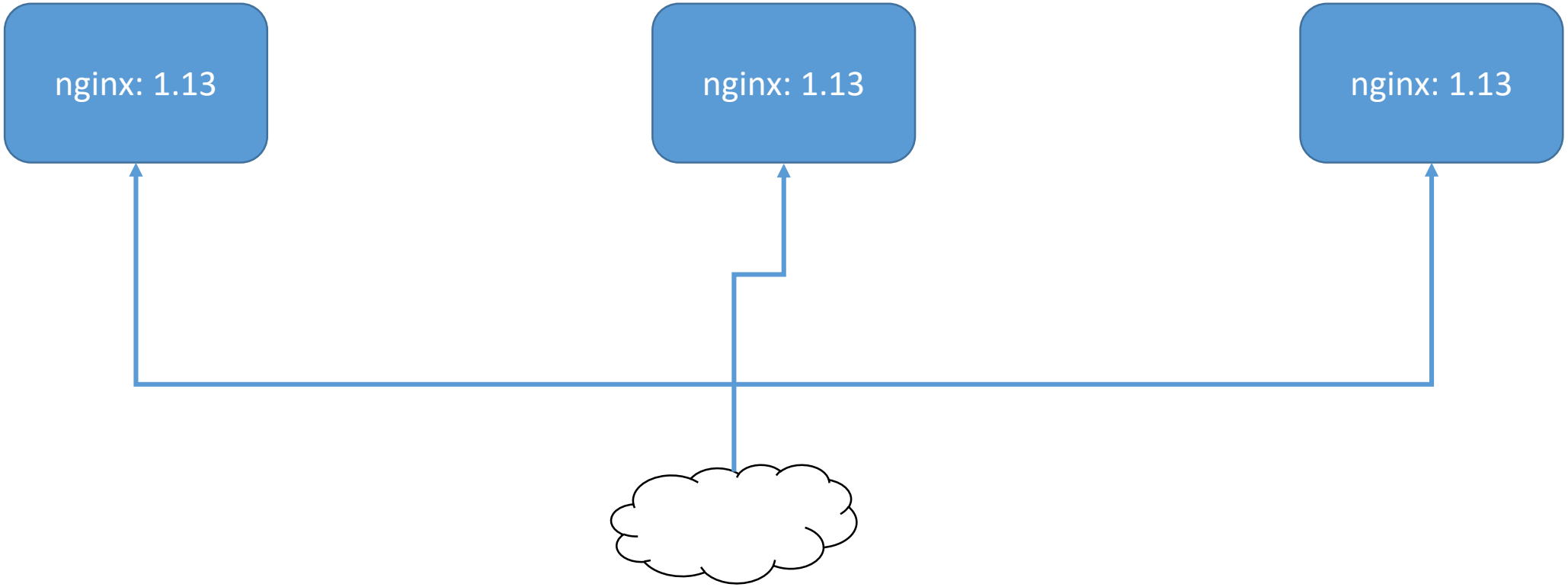
SCALECITY

Rolling update

Default deploy method in Kubernetes

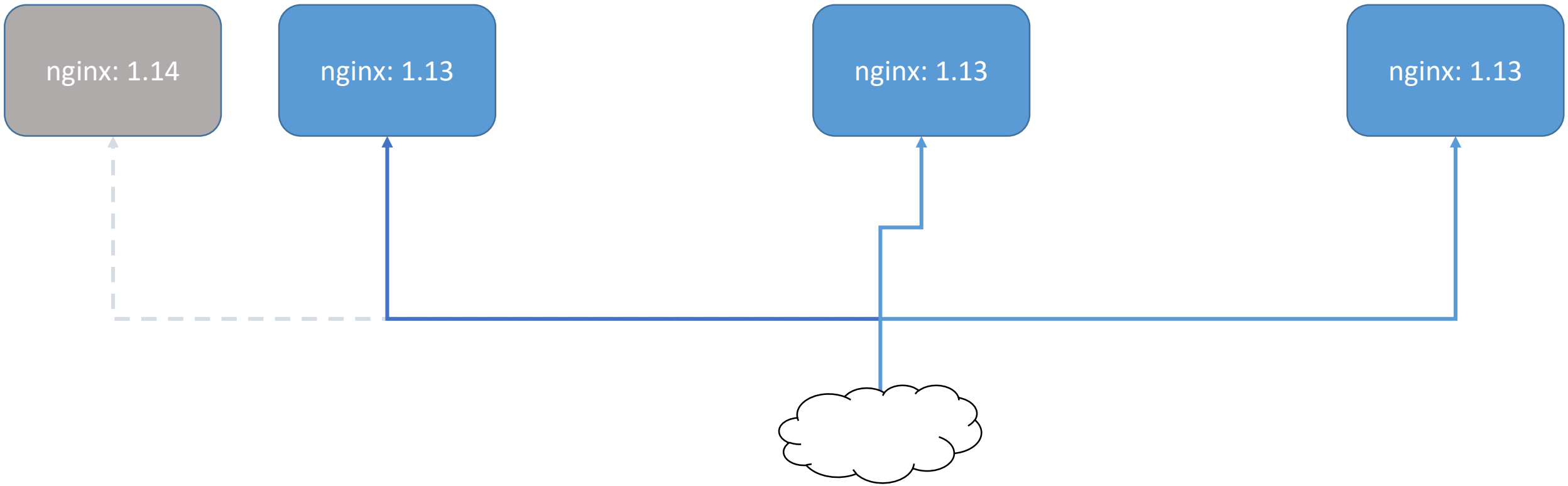
Rolling update

- Use-case: upgrading nginx from 1.13 to 1.14



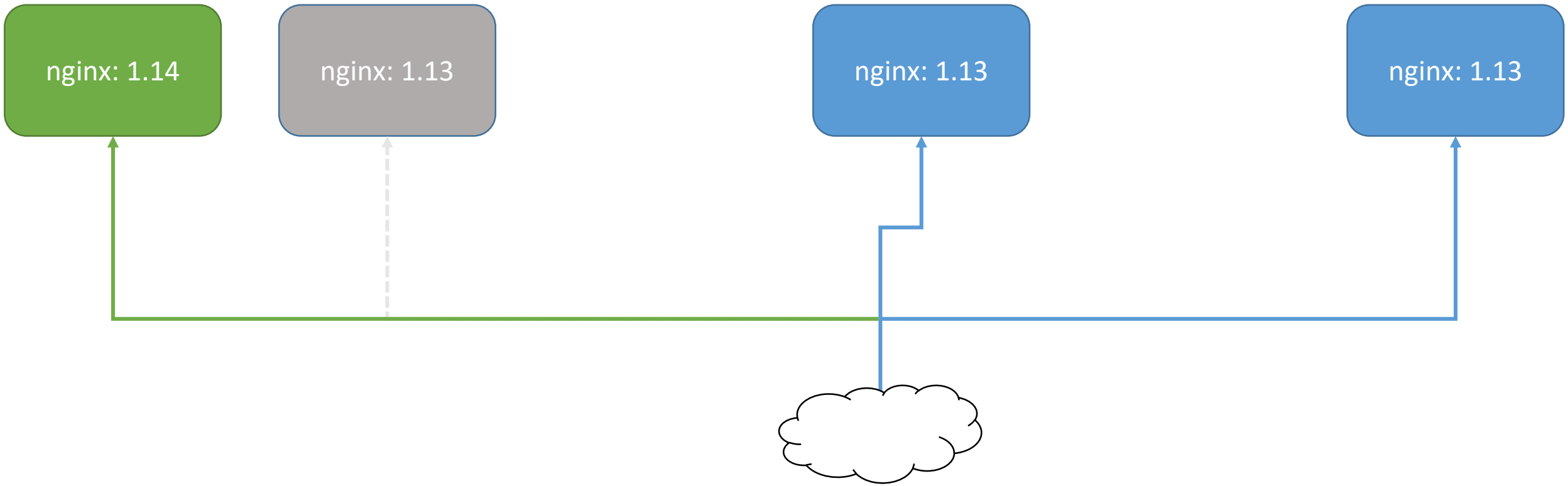
Rolling update

- Use-case: upgrading nginx from 1.13 to 1.14



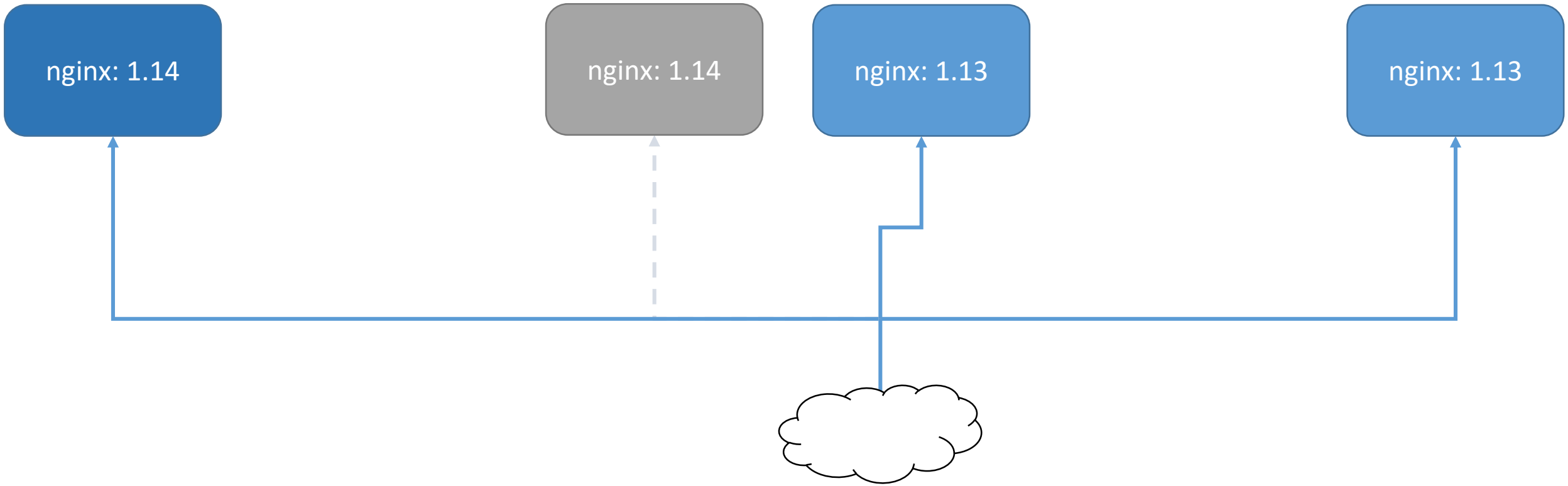
Rolling update

- Use-case: upgrading nginx from 1.13 to 1.14



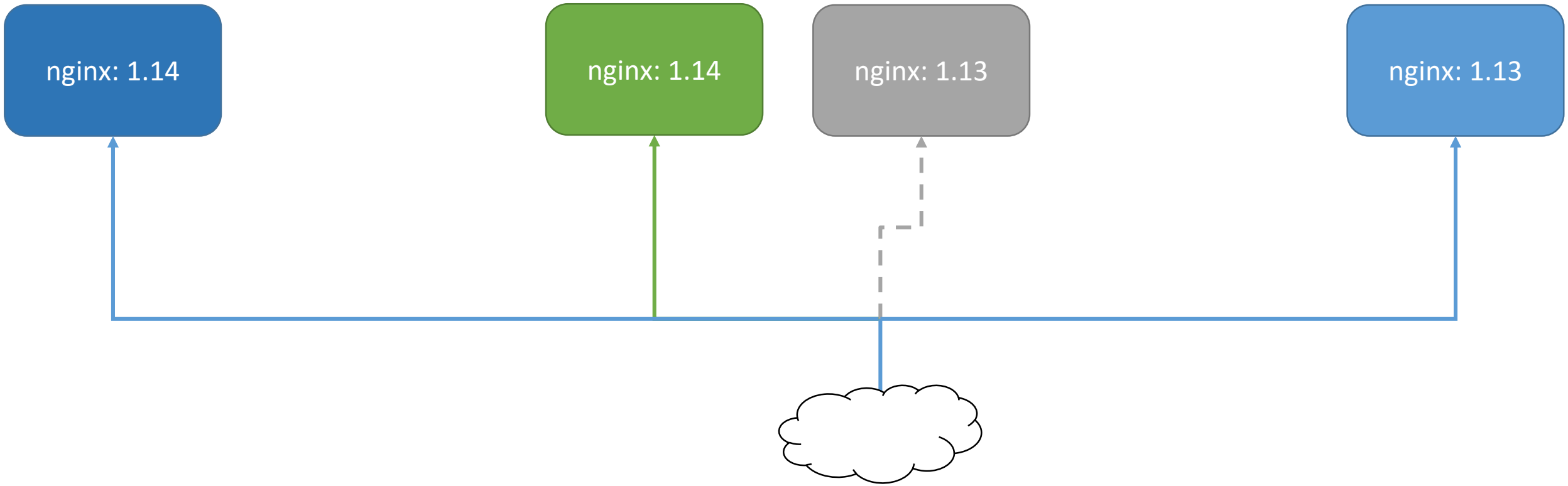
Rolling update

- Use-case: upgrading nginx from 1.13 to 1.14



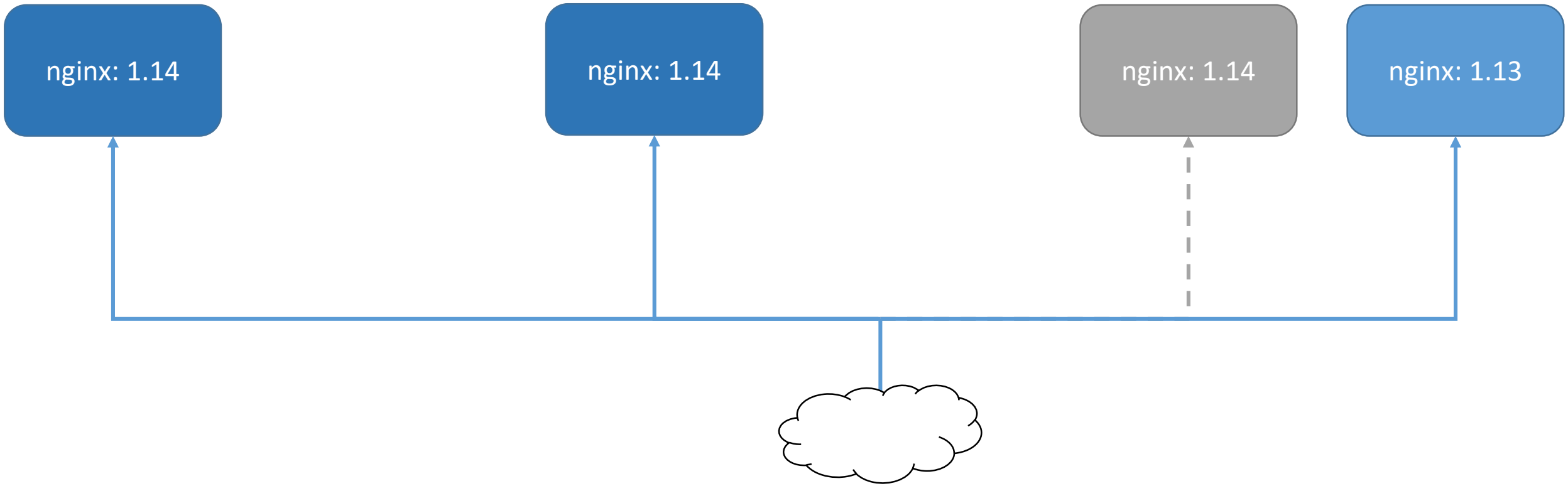
Rolling update

- Use-case: upgrading nginx from 1.13 to 1.14



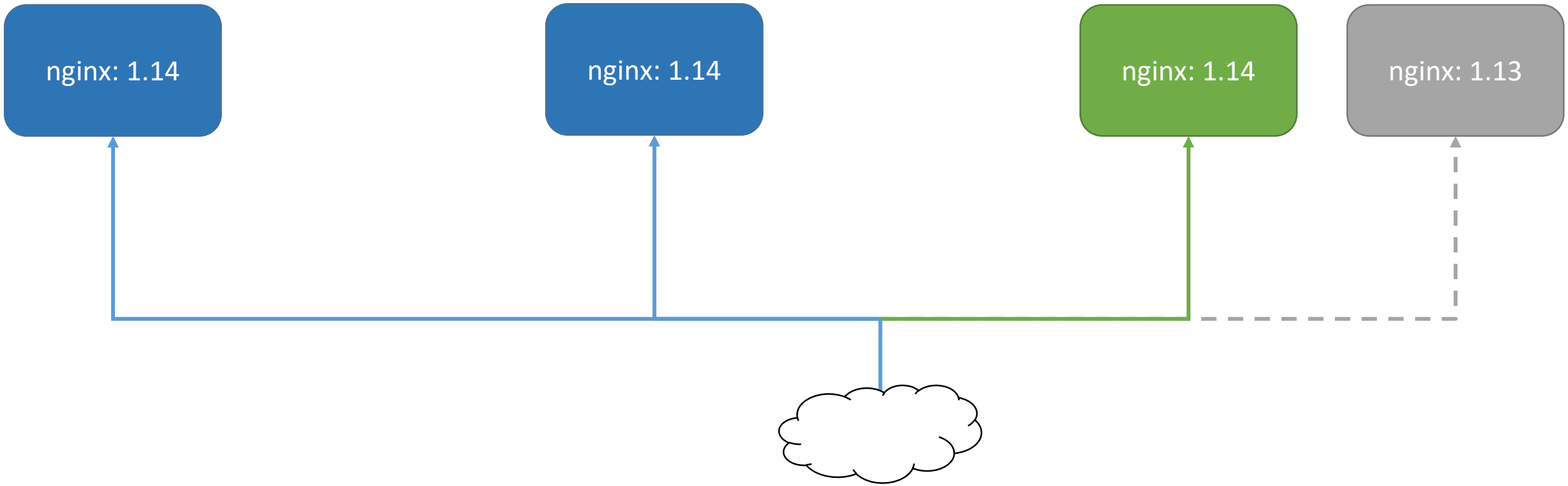
Rolling update

- Use-case: upgrading nginx from 1.13 to 1.14



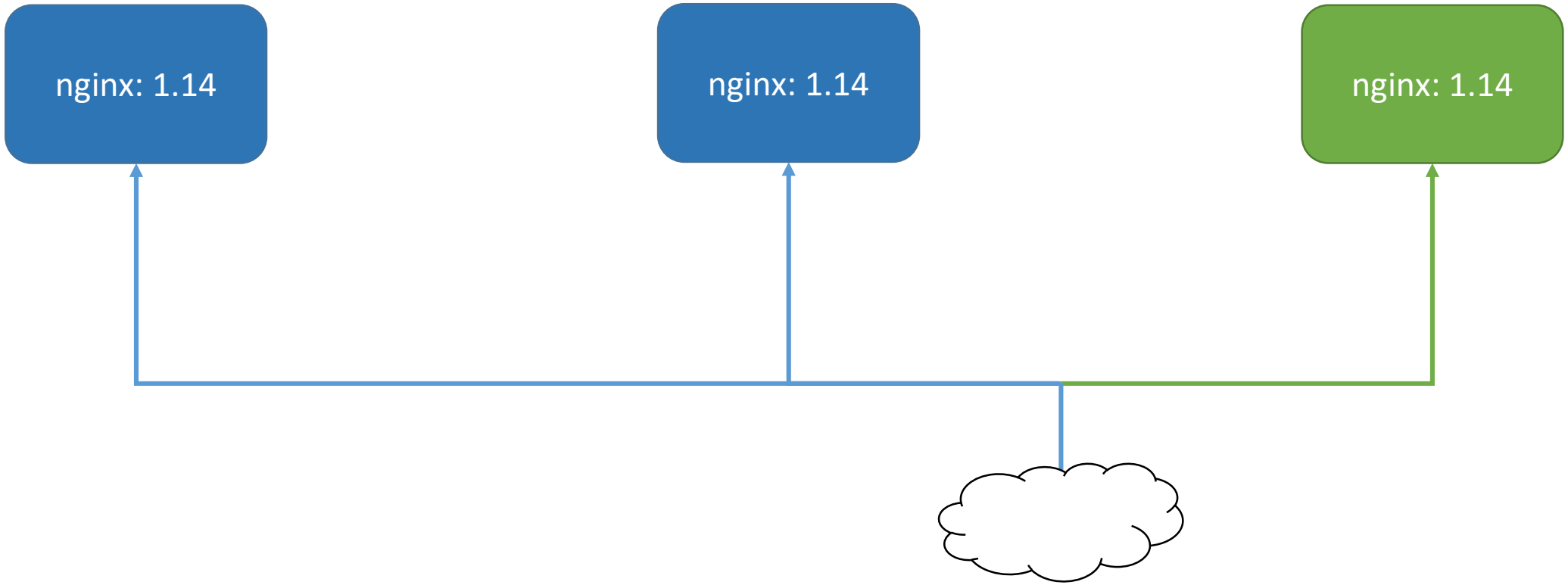
Rolling update

- Use-case: upgrading nginx from 1.13 to 1.14



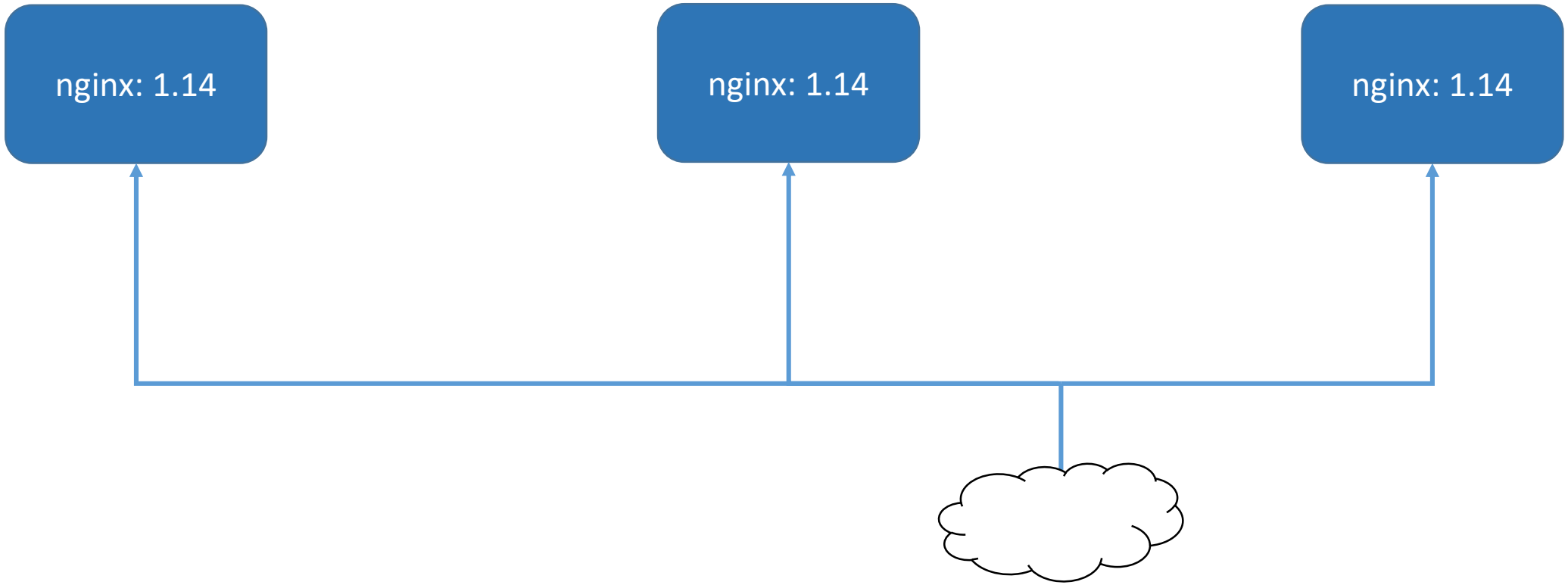
Rolling update

- Use-case: upgrading nginx from 1.13 to 1.14



Rolling update

- Use-case: upgrading nginx from 1.13 to 1.14

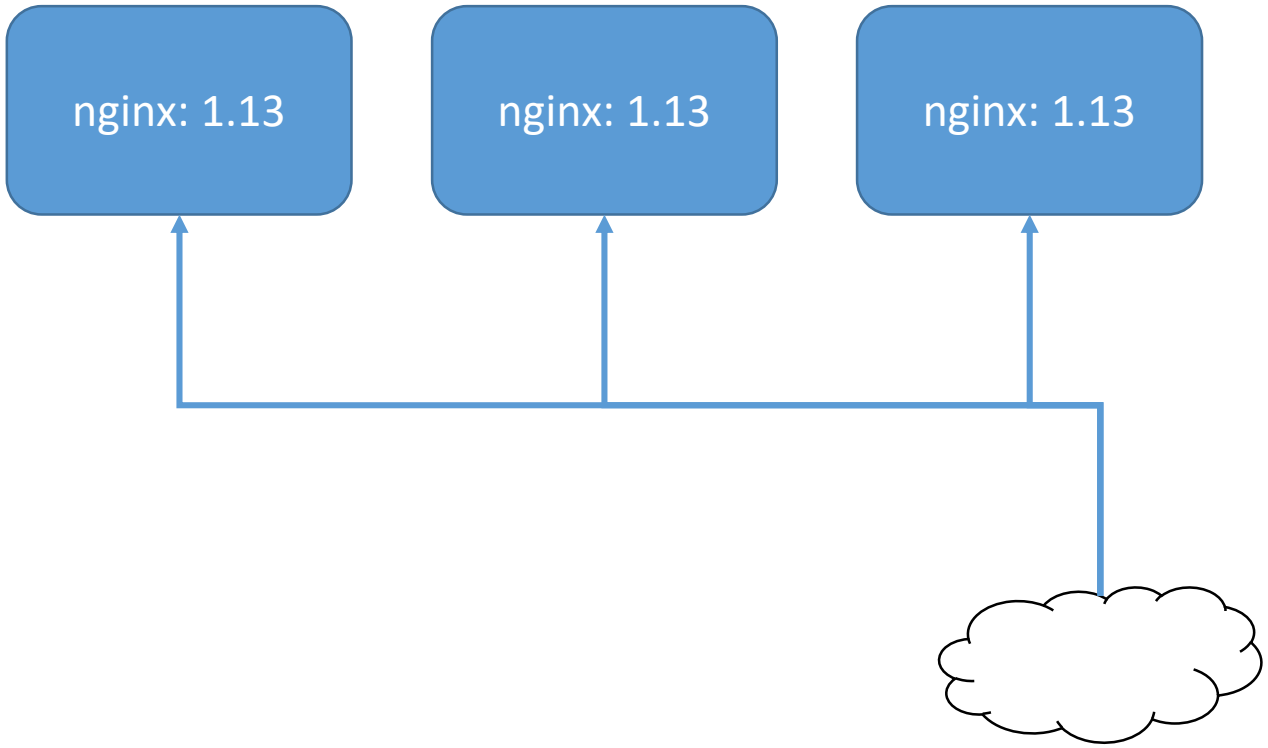


Blue-green

Not available in vanilla kubernetes

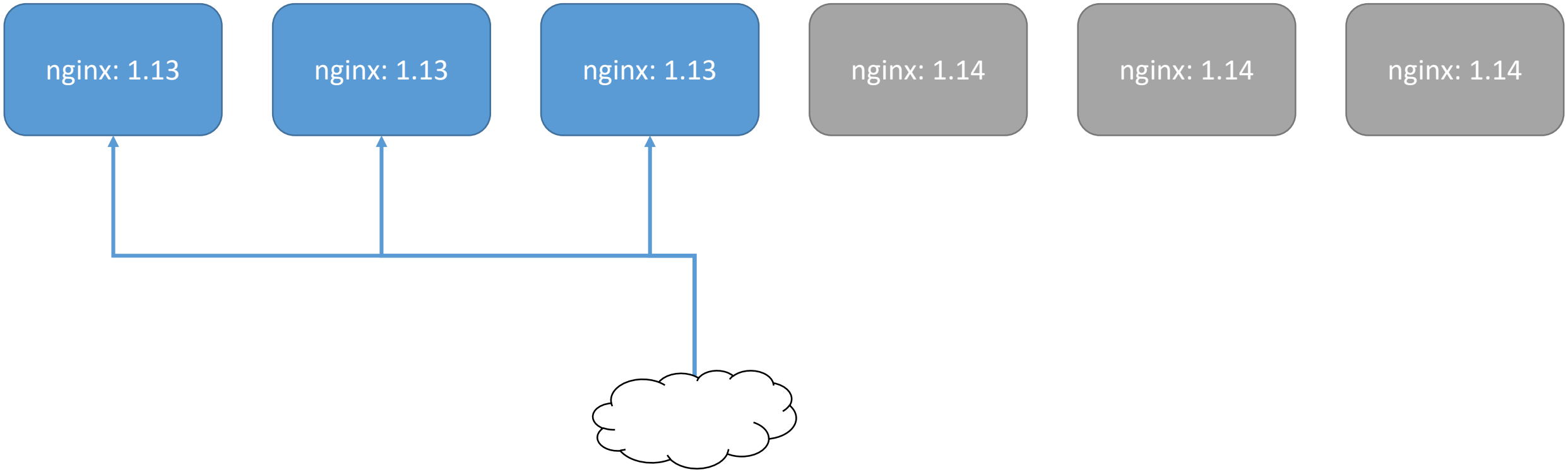
Blue-green

- Use-case: upgrading nginx from 1.13 to 1.14



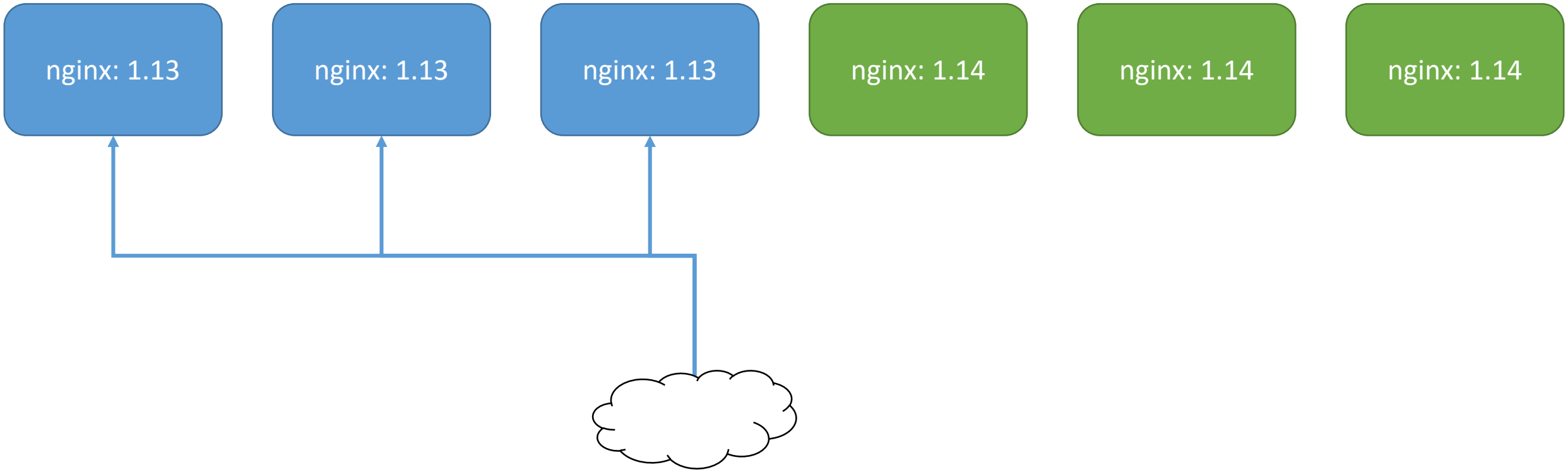
Blue-green

- Use-case: upgrading nginx from 1.13 to 1.14



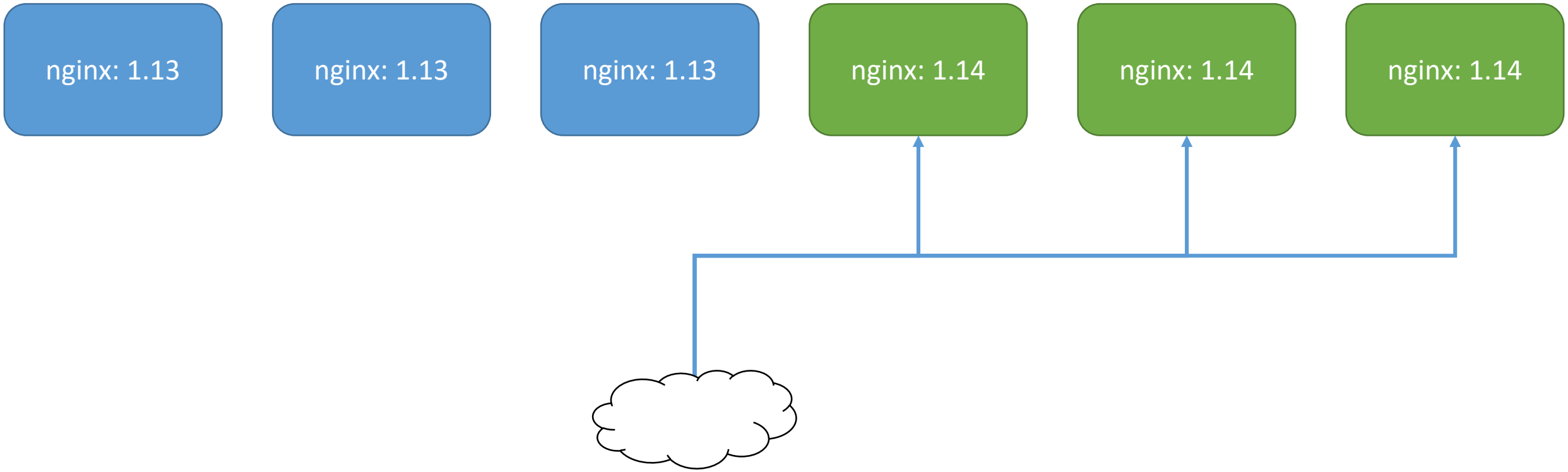
Blue-green

- Use-case: upgrading nginx from 1.13 to 1.14



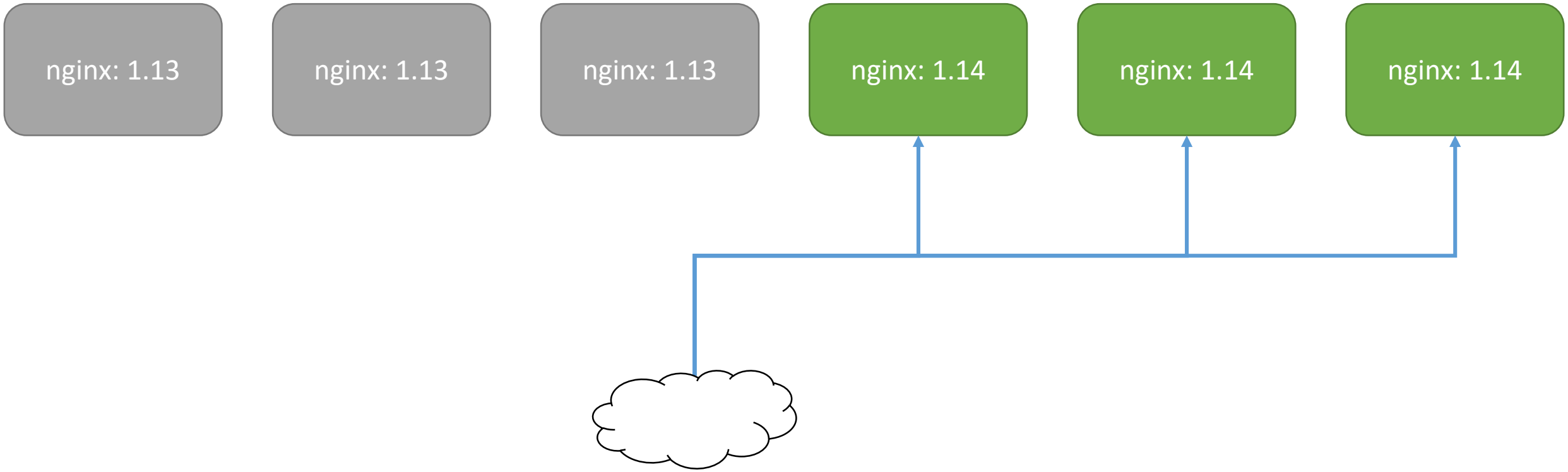
Blue-green

- Use-case: upgrading nginx from 1.13 to 1.14



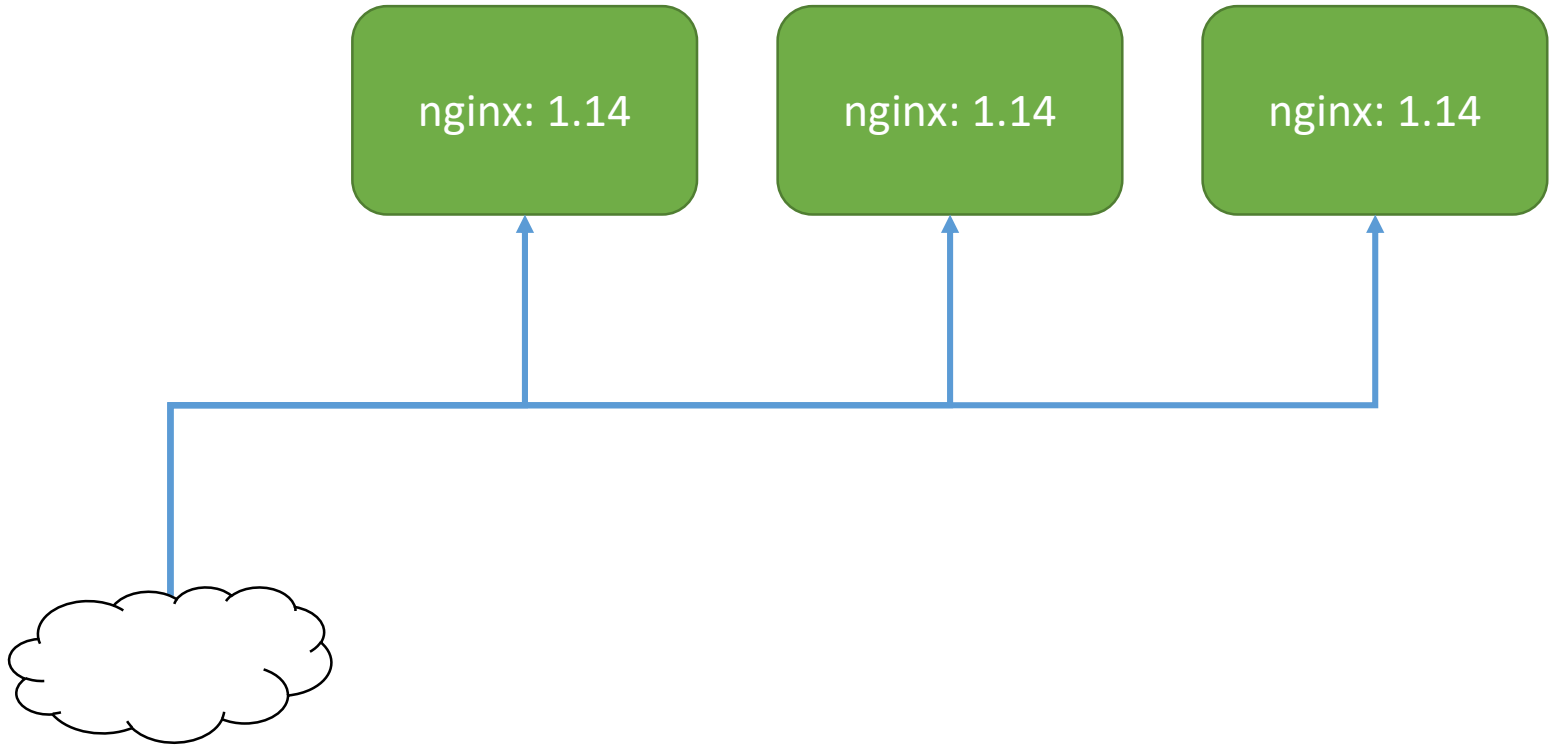
Blue-green

- Use-case: upgrading nginx from 1.13 to 1.14



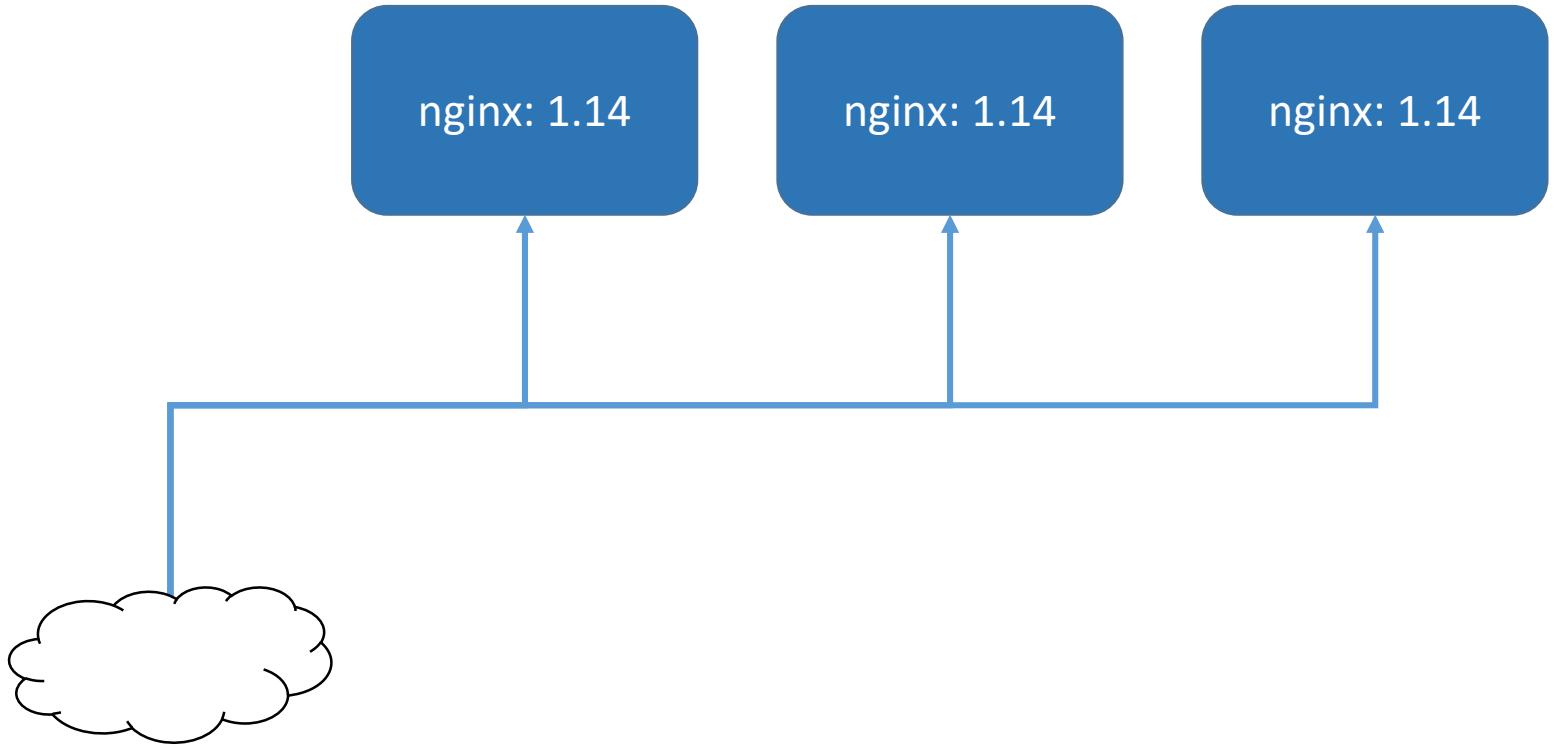
Blue-green

- Use-case: upgrading nginx from 1.13 to 1.14



Blue-green

- Use-case: upgrading nginx from 1.13 to 1.14



Why now

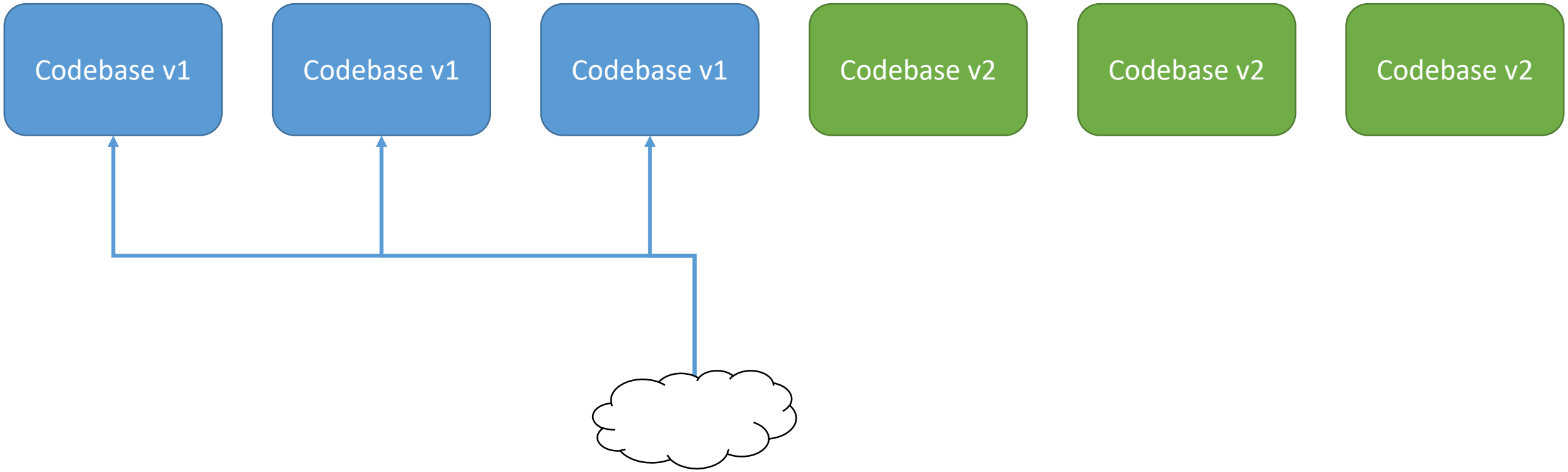
- **Not new**
- **Duplicate entire infrastructure**
- **Cheaper**
- **Stateless**

Advantages

- **Never in a mixed state**
- **Less downtime vs classic deployment**

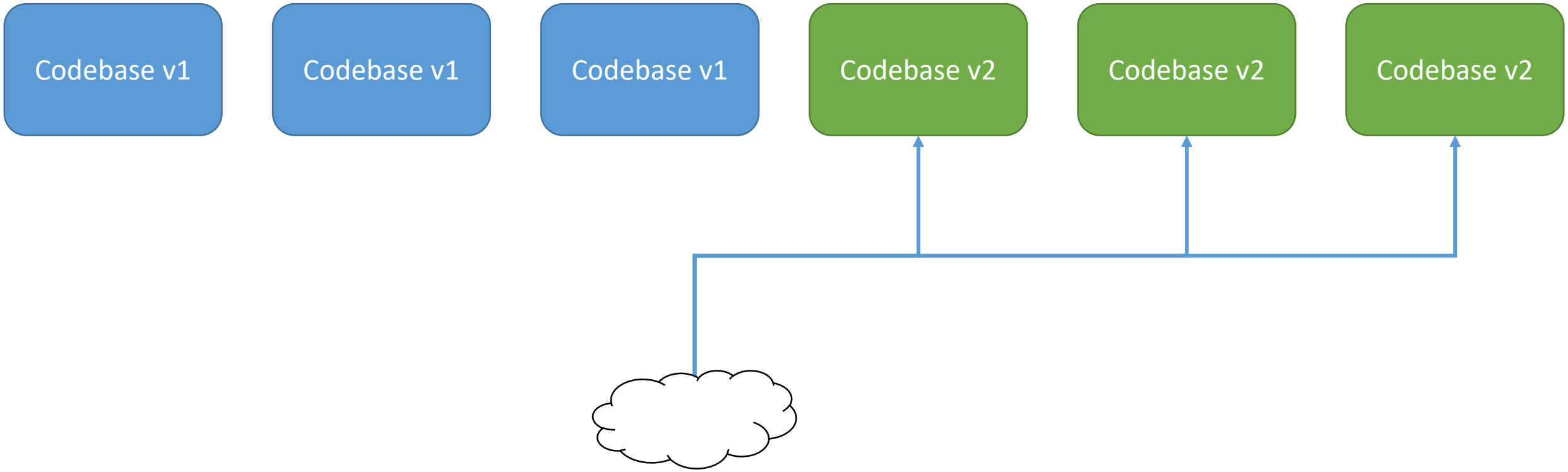
Blue-green

- Use-case: upgrading drupal



Blue-green

- Use-case: upgrading drupal

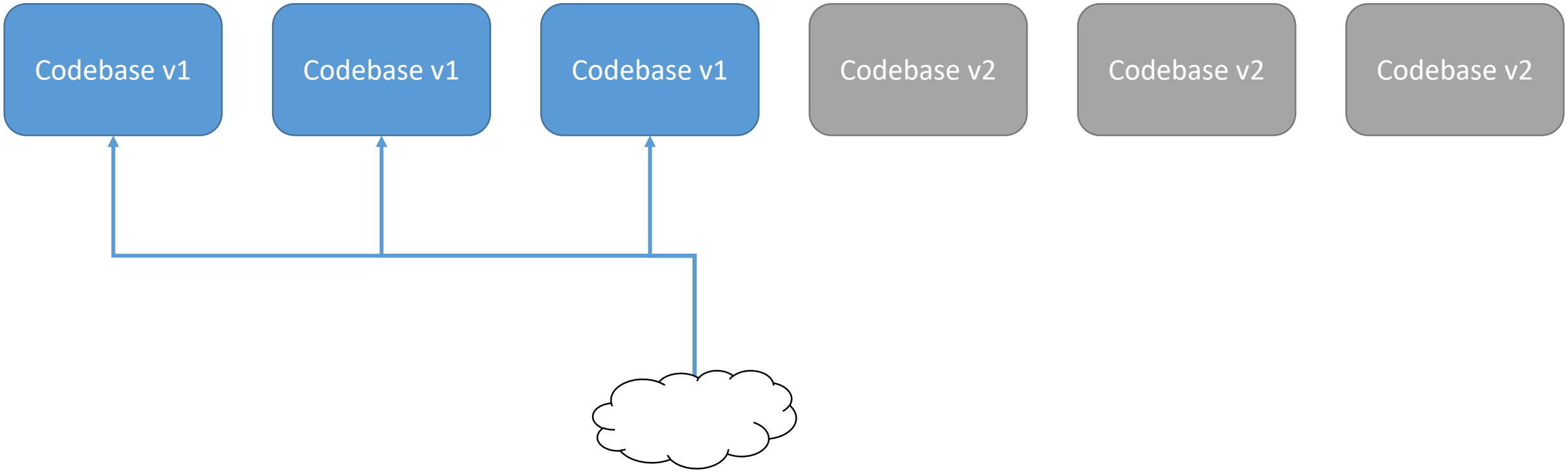


Advantages

- Never in a mixed state
- Less downtime vs classic deployment
- Safety nets
  - Rollback

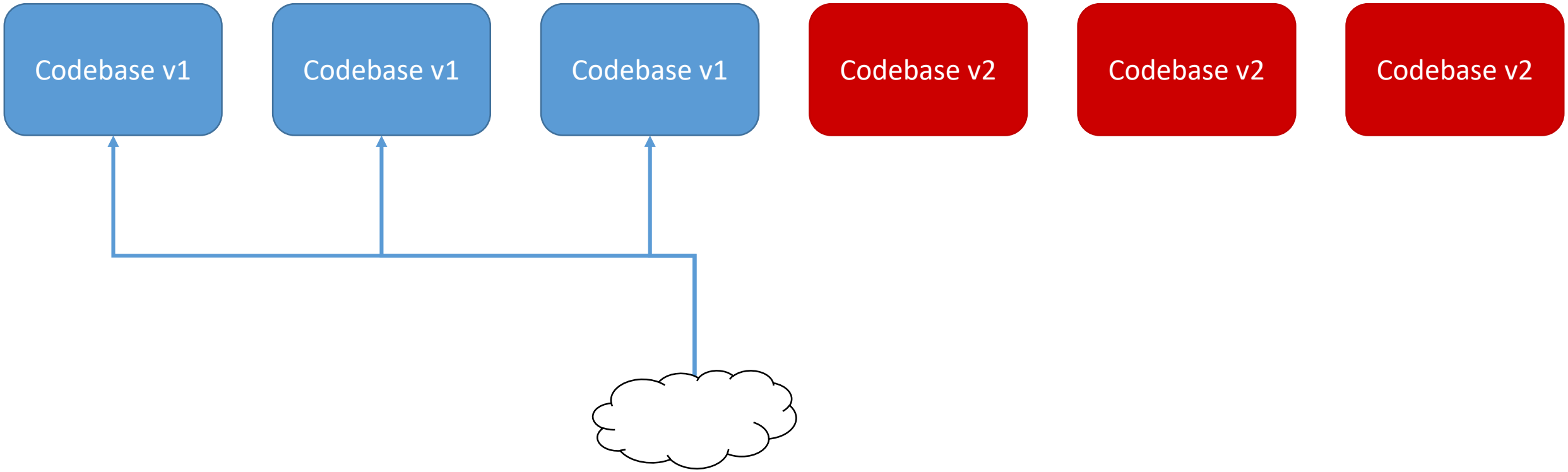
Blue-green gone wrong

- Use-case: fallback on faulty code



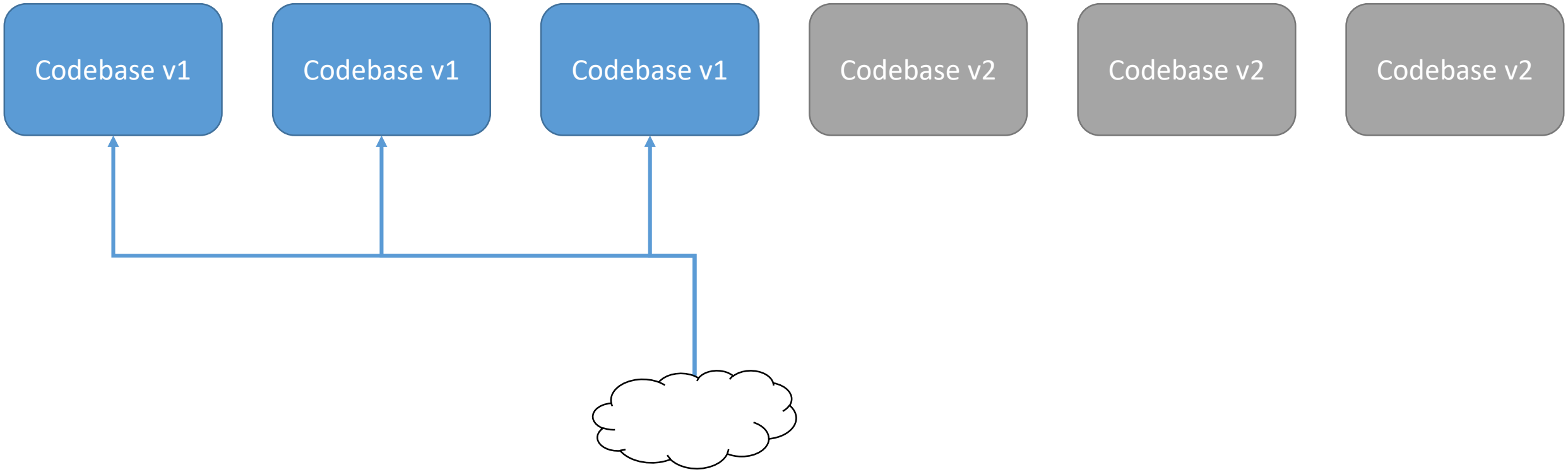
Blue-green gone wrong

- Use-case: fallback on faulty code



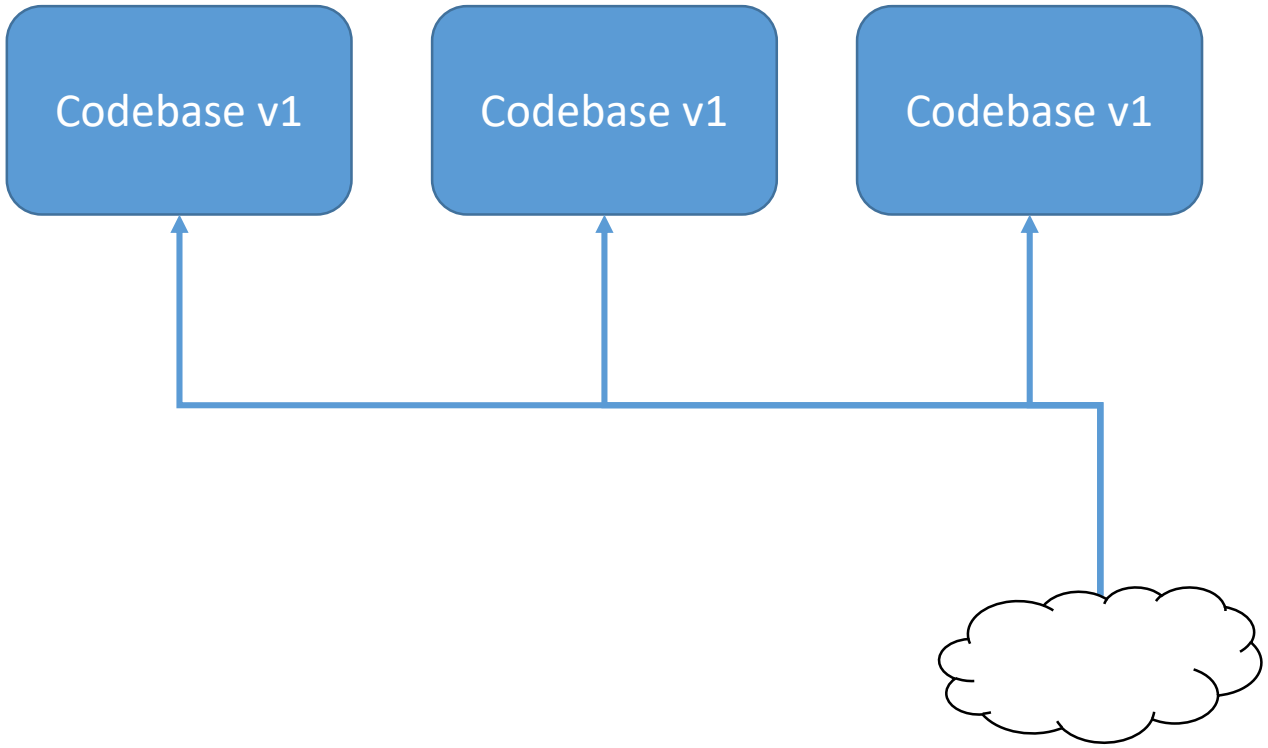
Blue-green gone wrong

- Use-case: fallback on faulty code



Blue-green gone wrong

- Use-case: fallback on faulty code

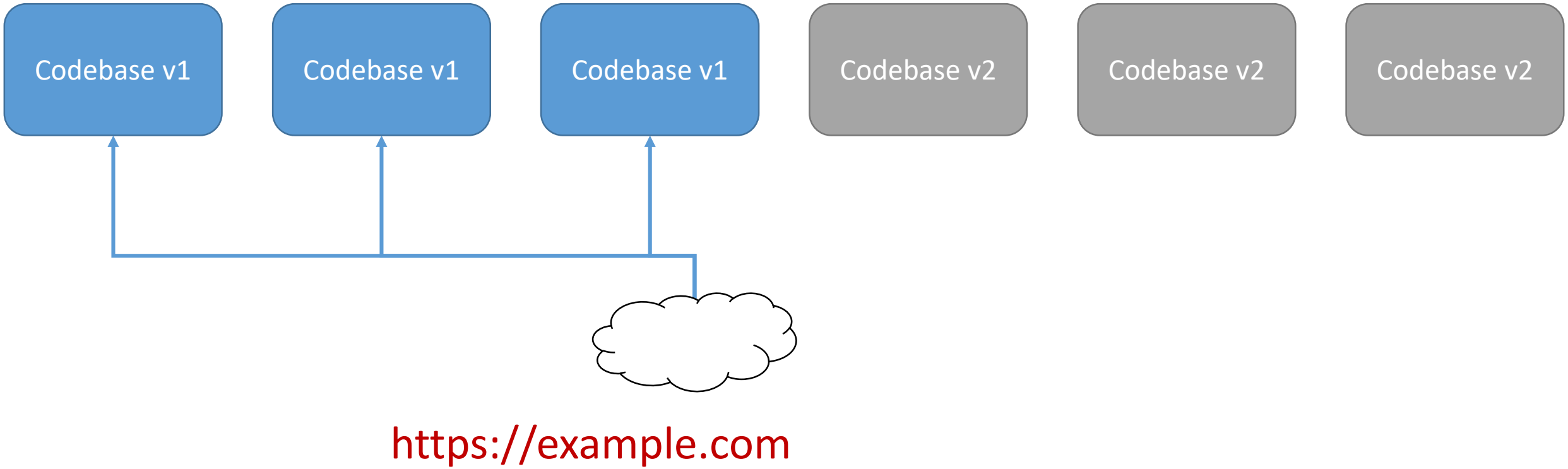


Advantages

- Never in a mixed state
- Less downtime vs classic deployment
- Safety nets
  - Rollback
  - Canary

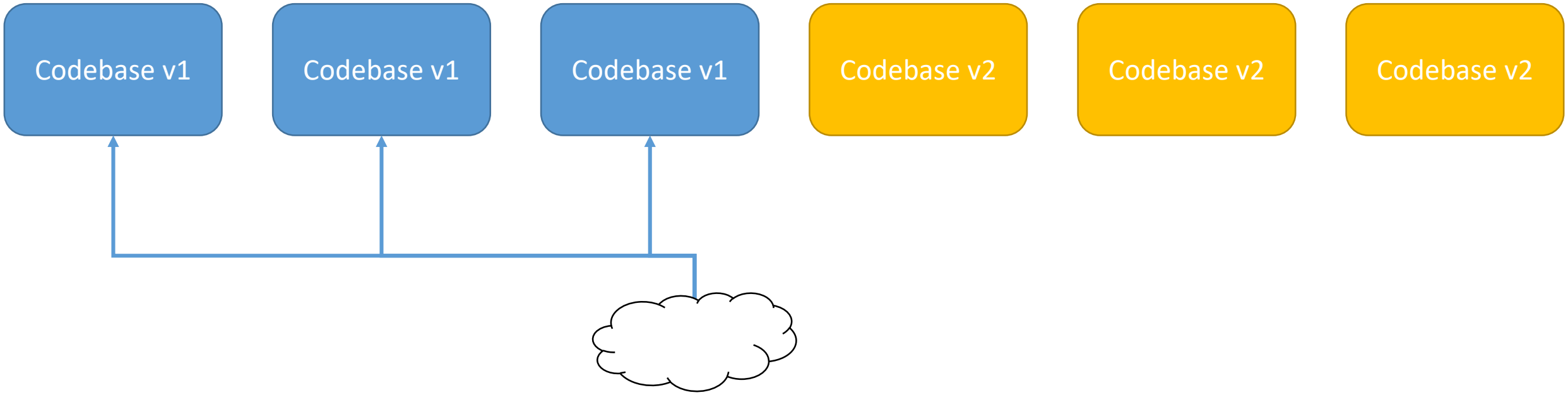
Blue-green canary

- Use-case: canary



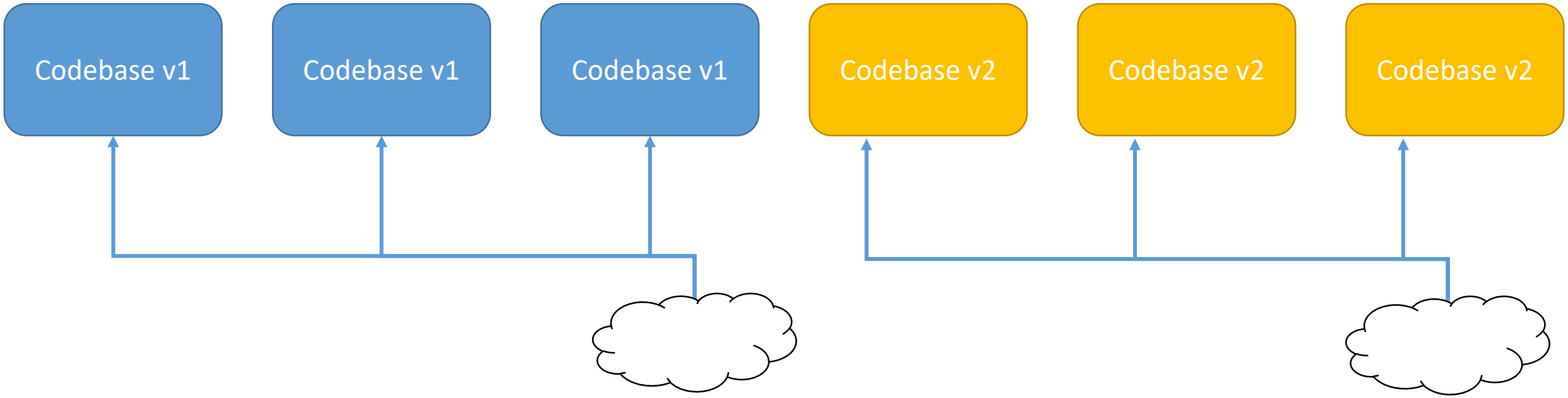
Blue-green canary

- Use-case: canary



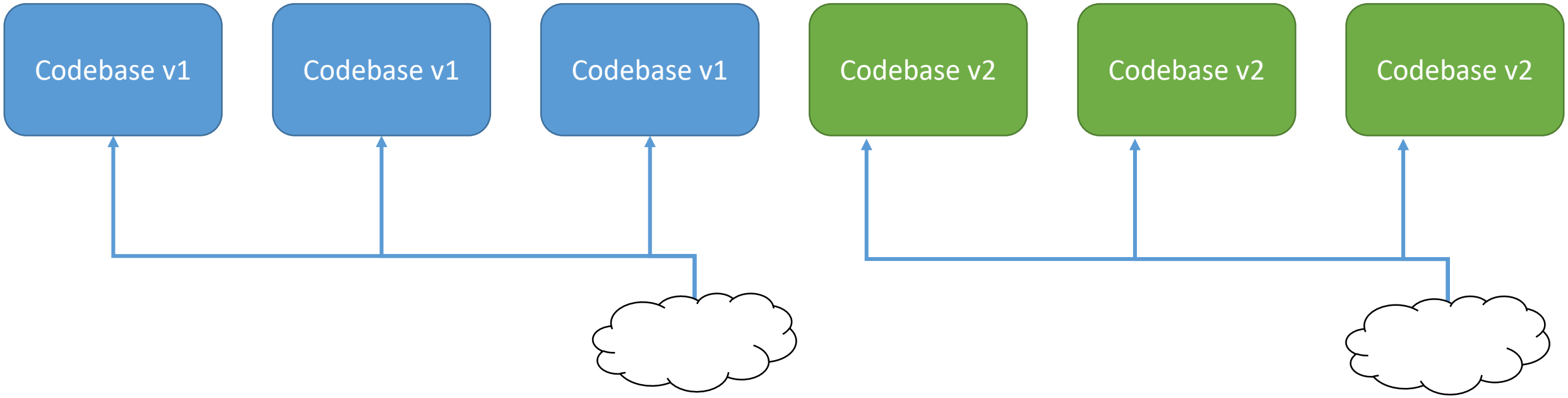
Blue-green canary

- Use-case: canary



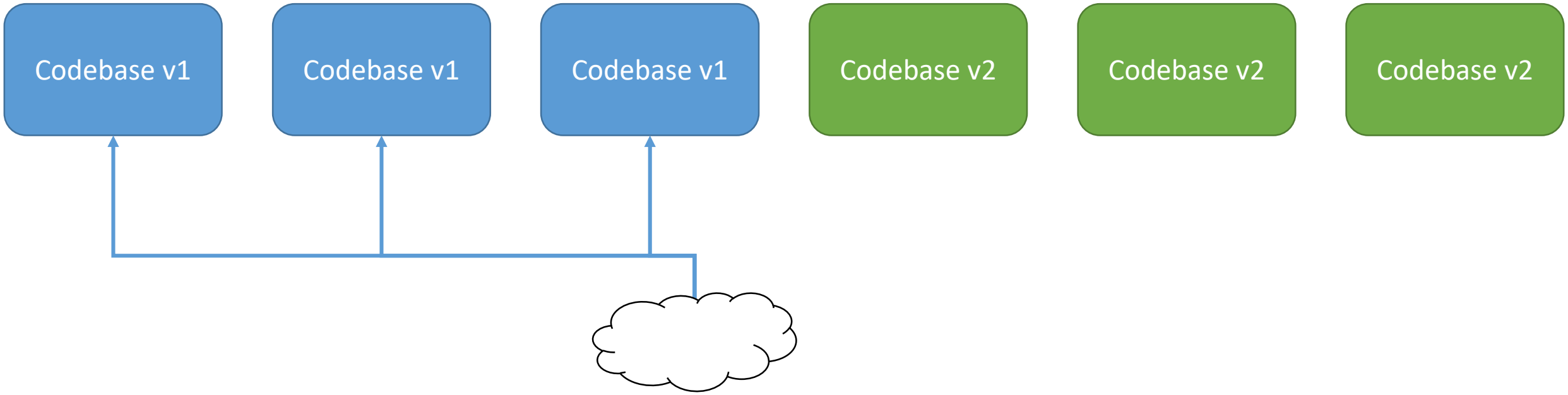
Blue-green canary

- Use-case: canary



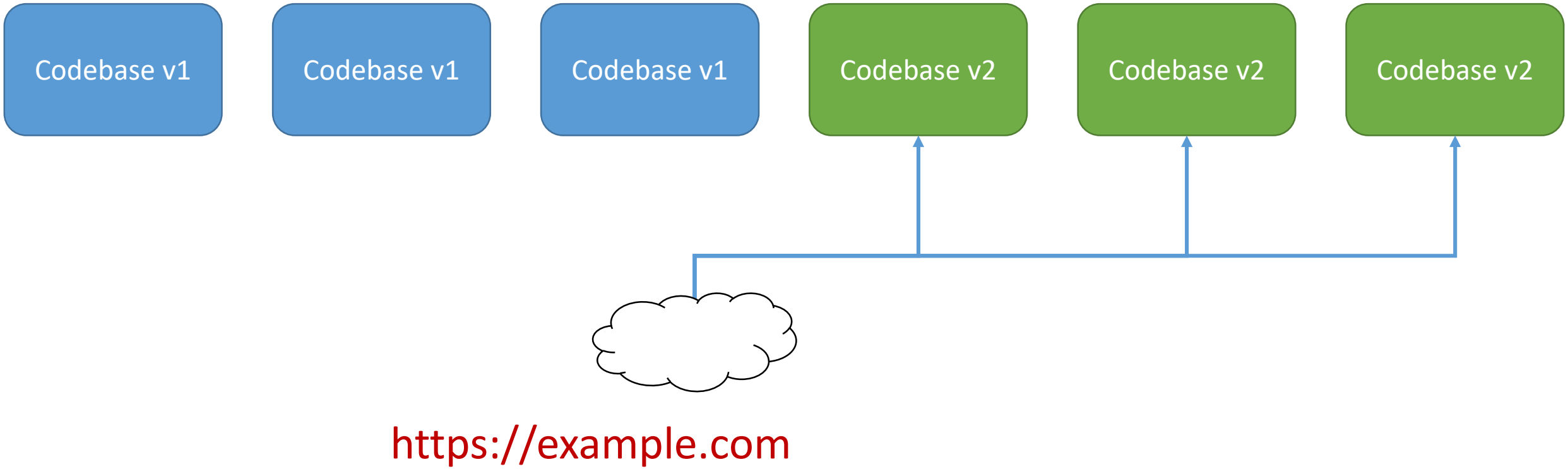
Blue-green canary

- Use-case: canary



Blue-green canary

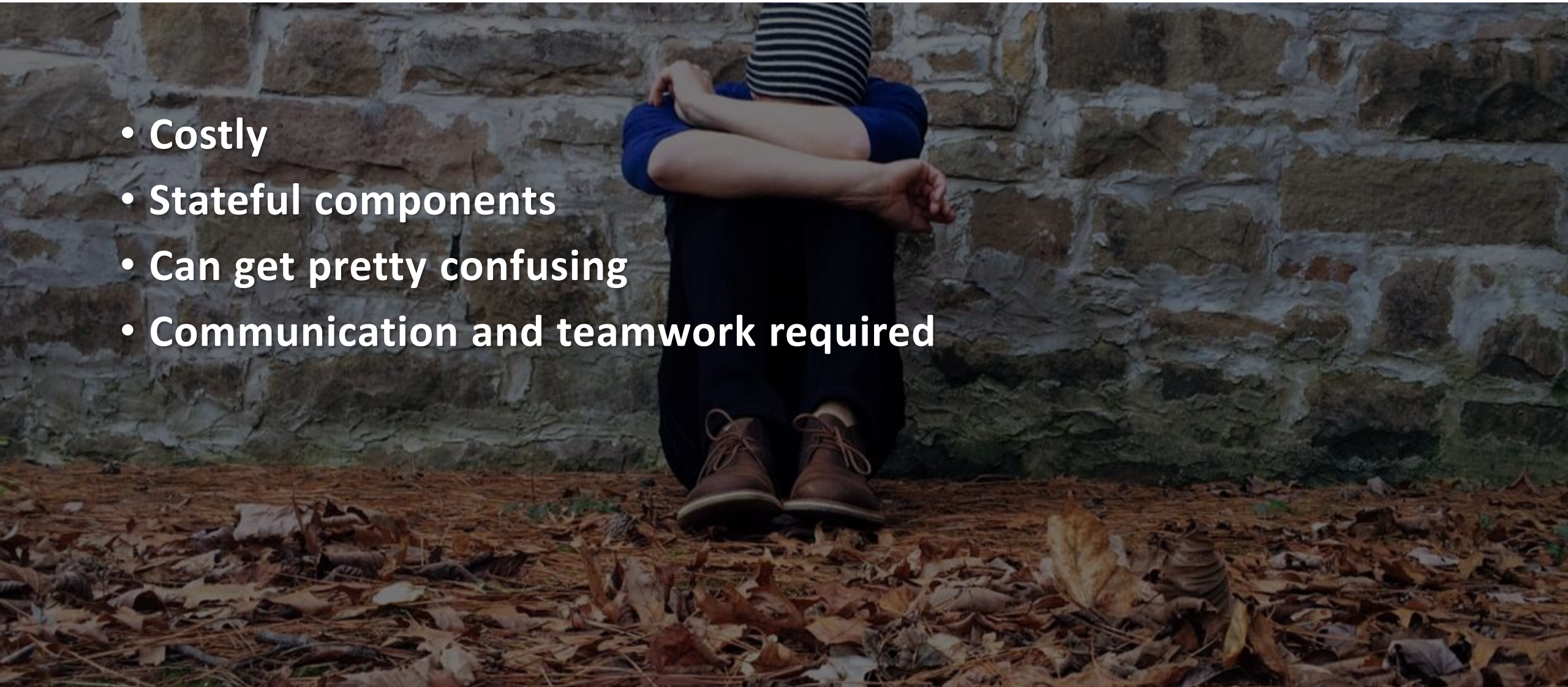
- Use-case: canary



Advantages

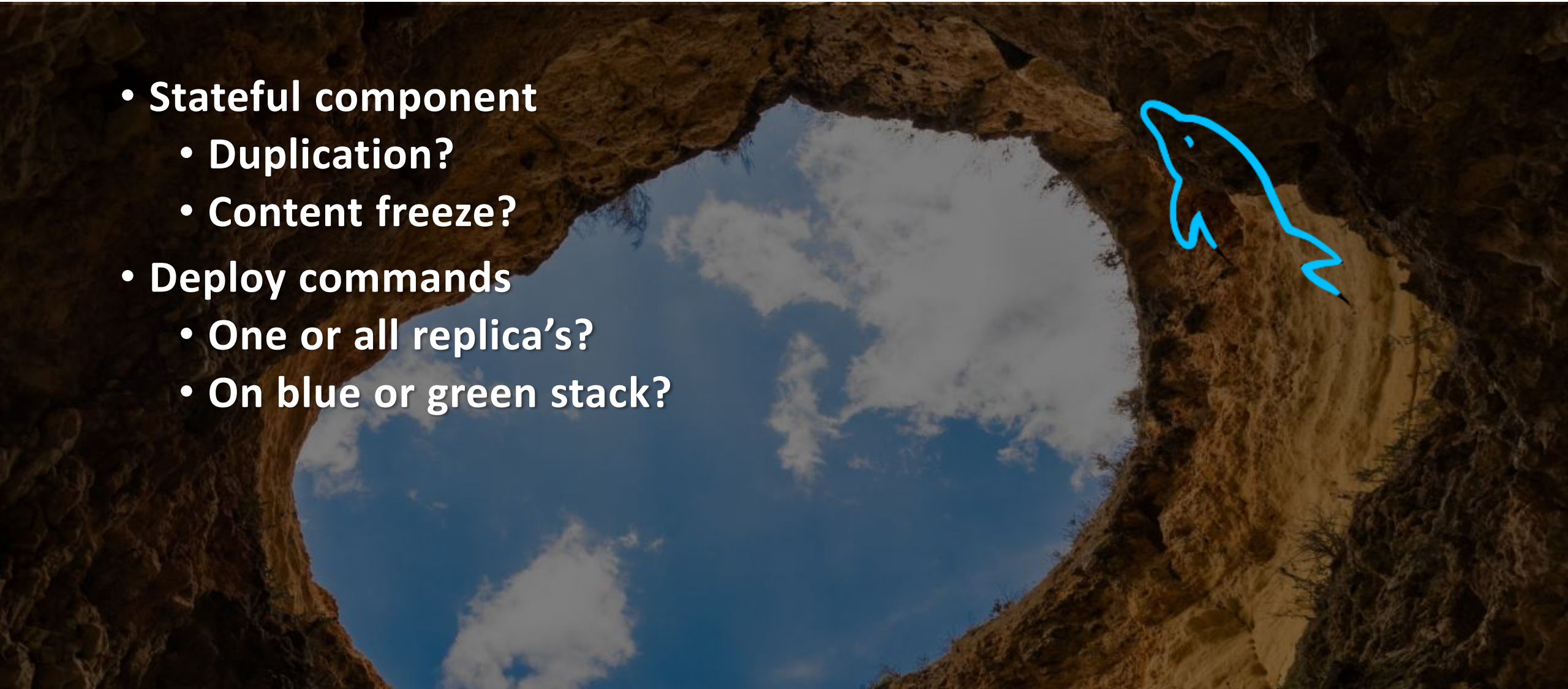
- Never in a mixed state
- Less downtime vs classic deployment
- Safety nets
  - Rollback
  - Canary

Issues



- Costly
- Stateful components
- Can get pretty confusing
- Communication and teamwork required

Drupal pitfalls



- Stateful component
  - Duplication?
  - Content freeze?
- Deploy commands
  - One or all replica’s?
  - On blue or green stack?

Thanks!

Nils Peeters
DevOps Engineer @ ScaleCity
nils@scalecity.io
https://www.linkedin.com/in/nilspeeters/

Become a Drupal contributor Friday from 9am

- First timers workshop
- Mentored contribution
- General contribution

Thanks!

Nils Peeters
DevOps Engineer @ ScaleCity
nils@scalecity.io
https://www.linkedin.com/in/nilspeeters/