DevOps + Infrastructure

TRACK SUPPORTED BY

[Logo]

Drupal Europe
Darmstadt, Germany
10 - 14 September 2018
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
• 1,800 contributors
• Google, Microsoft, Red Hat and Huawei
• Cloud Native Computing Foundation
  • https://www.cncf.io/
What is Kubernetes?

- Container orchestrator
- Automation
What is Kubernetes?

- Master
- API Server
What is Kubernetes?

- Nodepool(s)
- Same machine type
What is Kubernetes?

- Node
  - “Machine” (VM)
  - COS
  - Fleeting
  - Pods
Pods vs Containers

What is this “Pod” you speak of?
Pods vs Containers

Pod != Container
I’m a container!

I’m a pod!
Pods vs Containers: example

PHP-fpm

NGINX
Pods vs Containers: example

Pod: www

PHP-fpm

/var/www/example.com

Nginx

localhost
Pods vs Containers: example

- ElasticSearch
- Kibana
Pods vs Containers: example

Pod: elastic

```
| elastic | /var/lib/elastic | /var/lib/kibana | kibana |
```
Pods vs Containers: example

Pod: *elastic*

```
elastic
```
```
/var/lib/elastic
```

Pod: *kibana*

```
/var/lib/kibana
```
```
kibana
```
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
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
Will resolve as hostname
www:80
Labels & label selectors

Connecting the dots
Labels & label selectors

- Services connect pods
- Key value
- Get operations by label
Labels & label selectors

```
elastic-master-1
  elastic
  app=elasticsearch

elastic-data-1
  elastic
  app=elasticsearch

elastic-data-2
  elastic
  app=elasticsearch

Drupal
  elastic:9200

app=elasticsearch
Service: elastic
port: 9200
```
Labels & label selectors
Labels & label selectors

- **elastic-master-1**
  - app=elasticsearch
  - role=master

- **elastic-data-1**
  - app=elasticsearch
  - role=data

- **elastic-data-2**
  - app=elasticsearch
  - role=data

- **Service: elastic-internal**
  - app=elasticsearch
  - role=data
  - port: 9200
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

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

Watch out here!
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

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

varnish:80,443

www:80

cron

mysql:3306

https://example.com
Deploying

All that good stuff
Jenkins

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

https://example.com
Blue-green canary

• Use-case: canary

https://example.com
Blue-green canary

• Use-case: canary

Codebase v1  Codebase v1  Codebase v1  Codebase v2  Codebase v2  Codebase v2

https://example.com

https://internal.example.com
Blue-green canary

• Use-case: canary

https://example.com

https://internal.example.com
Blue-green canary

• Use-case: canary

[start diagram]

Codebase v1

Codebase v1

Codebase v1

Codebase v2

Codebase v2

Codebase v2

[cloud]

https://example.com
Blue-green canary

• Use-case: canary

https://example.com
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/