KubeCon | CloudNativeCon

China 2018

peytz&co

Using Kubernetes

for handling second screen experience of european tv show

About us

Key Qualifications

- 12+ years' experience with development and operations
- Automating everything
- Been handling Kubernetes clusters in production for 2 years

Thomas Hector
Head of IT Operations

Key Qualifications

- 20+ year of experience in web-development & operation/hosting
- OpenSource advocate
- Extensive hand-on experience with Kubernetes and Containerization

Jan-Erik Revsbech
CTO

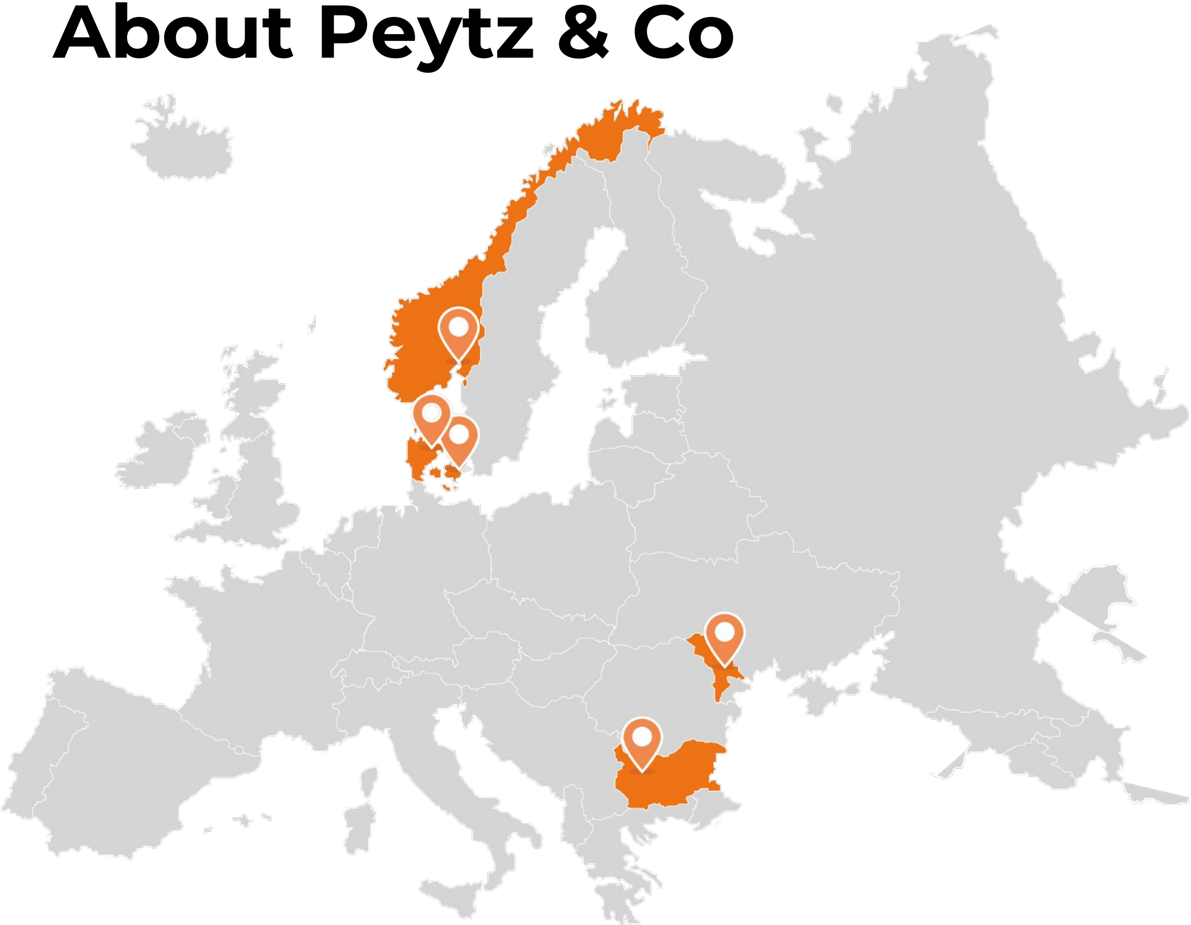
@peytzco

@Peytz & Co

@Peytz & Co

peytz&co

About Peytz & Co

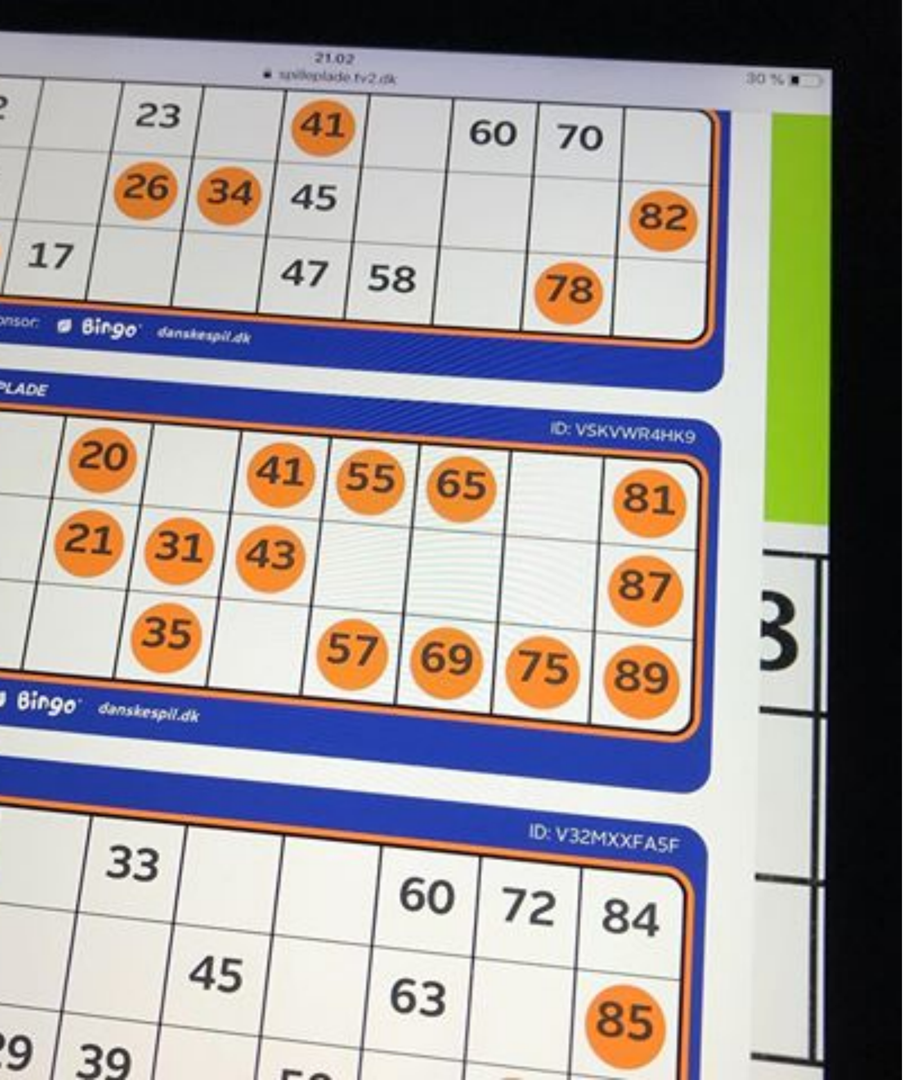


- 5 offices
- +100 employees
- OpenSource
- Infrastructure projects
- Custom built IT solutions
- 15 years' experience

peytz&co

Danish TV Show

TV2



The task

Danish televison broadcaster TV2 asked us

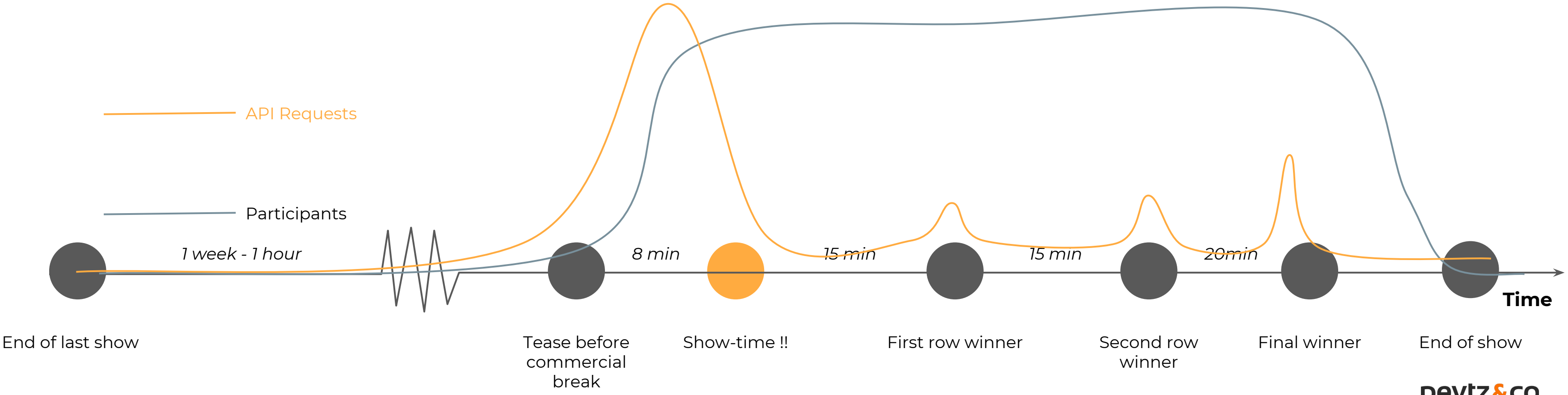
- To create an application which works on all devices
- To create an admin interface to run the show
- The application should be able to scale quickly but keep lowest cost possible
- Users must identify themselves (log in)
- Gametickets must be e-mailed to users

Challenge: Timing!

Weekly show - 9 weeks in a row

Attempt to get people to sign up early - unsuccessful

Hundreds of thousands users signing up during commercial break.



peytz&co

Application architecture

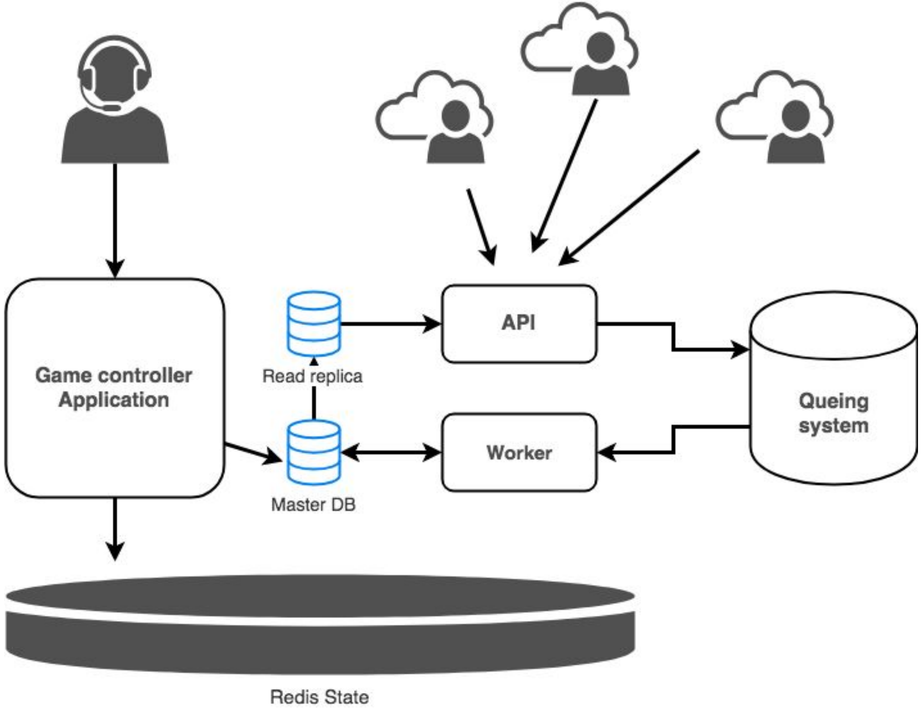
Application architecture

PHP-based Admin application

Shared state in Redis and SQL database

Scalable userfacing API in GoLang

Queuing system



Application technologies

Best of breed-technologies

Right tool for the right job

Easy prototyping with php - good performance with GoLang

HAPROXY
MariaDB
golang
NGINX
php
NSQ
redis

Scaling and orchestration

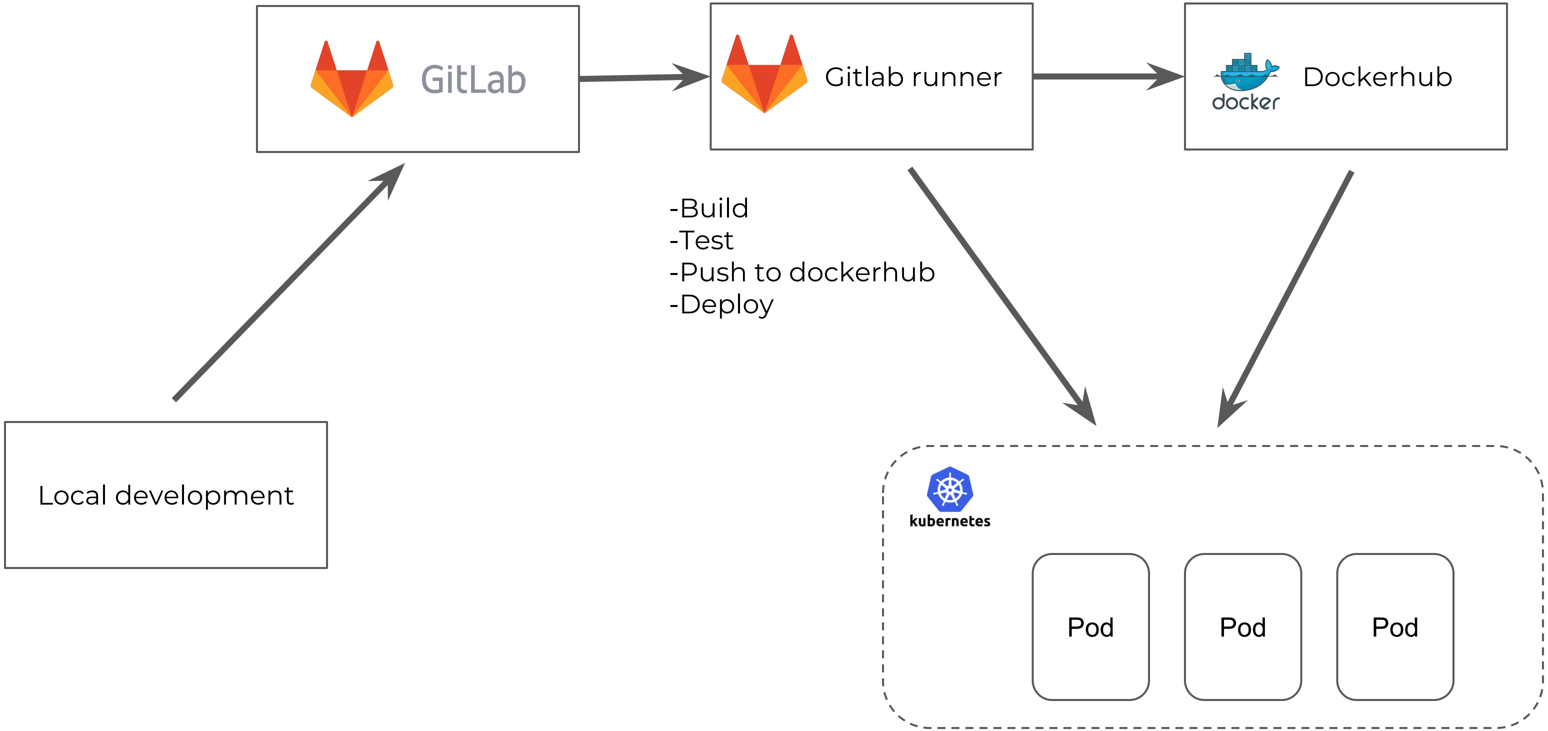
Containerized development and production environment

Cloud orchestration with Kubernetes

Hosted on AWS

kubernetes docker aws

Deployment, CI and CD



-Build
-Test
-Push to dockerhub
-Deploy

Application infrastructure

The services around Kubernetes

Experience

Take control

Easy to manage and scale

aws

Amazon ElastiCache

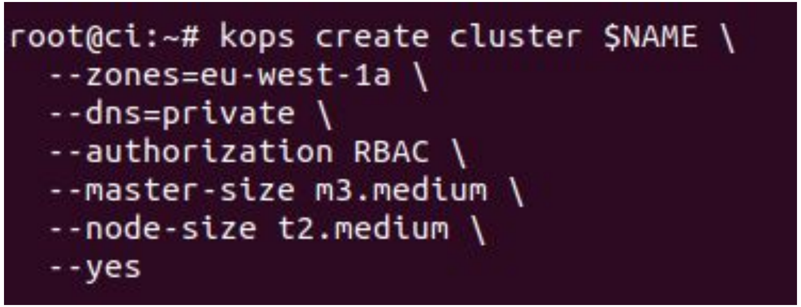
amazon RDS

kops

KOPS: Installation

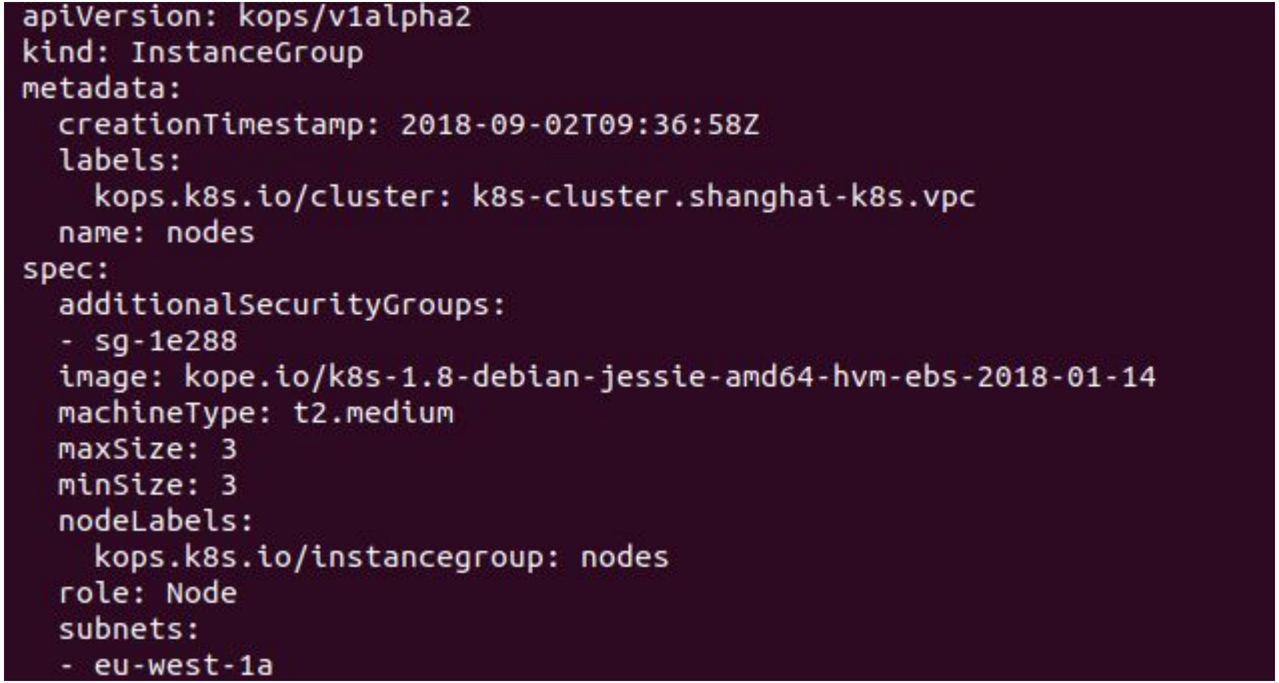
- Create **AWS user**
- Make a **CI host**
- wget and **install kops** and **kubectl**
- Configure **AWS cli**
- Create **S3** Bucket
- Export **KOPS** variables
- Setup **Route53**

KOPS: Create cluster



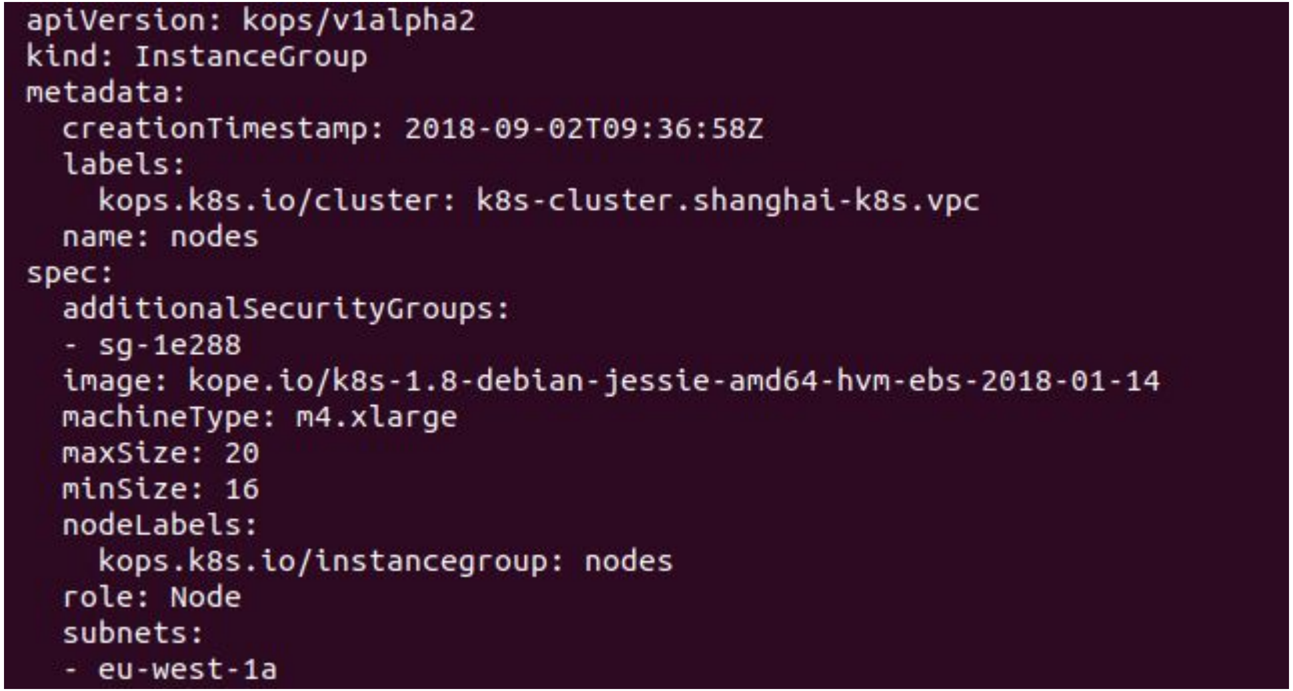
```
root@ci:~# kops create cluster $NAME \
  --zones=eu-west-1a \
  --dns=private \
  --authorization RBAC \
  --master-size m3.medium \
  --node-size t2.medium \
  --yes
```

KOPS: Edit ig nodes



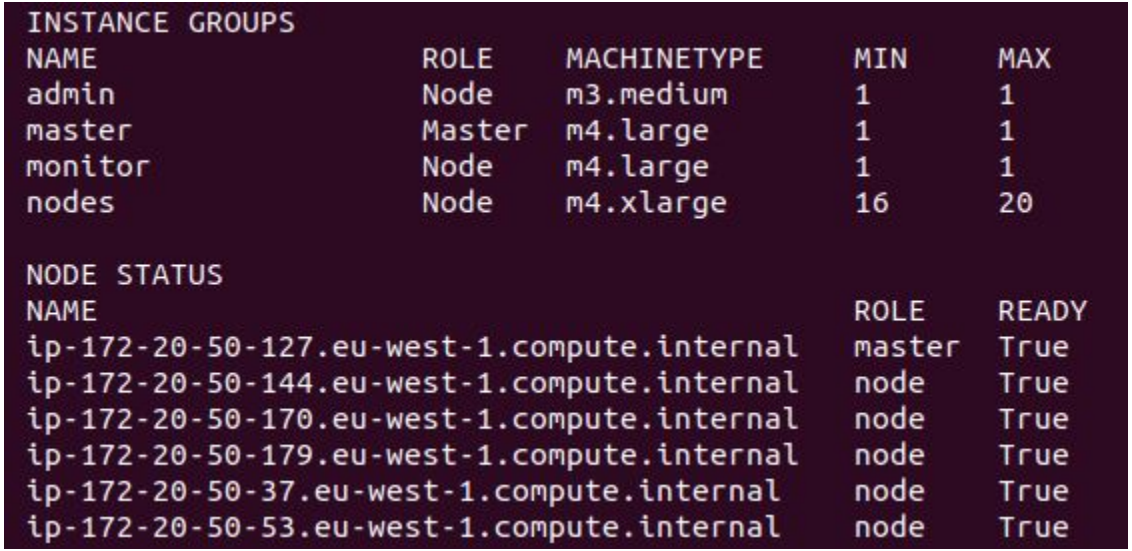
```
apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
  creationTimestamp: 2018-09-02T09:36:58Z
  labels:
    kops.k8s.io/cluster: k8s-cluster.shanghai-k8s.vpc
  name: nodes
spec:
  additionalSecurityGroups:
  - sg-1e288
  image: kope.io/k8s-1.8-debian-jessie-amd64-hvm-ebs-2018-01-14
  machineType: t2.medium
  maxSize: 3
  minSize: 3
  nodeLabels:
    kops.k8s.io/instancegroup: nodes
  role: Node
  subnets:
  - eu-west-1a
```

KOPS: Edit ig nodes



```
apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
  creationTimestamp: 2018-09-02T09:36:58Z
  labels:
    kops.k8s.io/cluster: k8s-cluster.shanghai-k8s.vpc
  name: nodes
spec:
  additionalSecurityGroups:
  - sg-1e288
  image: kope.io/k8s-1.8-debian-jessie-amd64-hvm-ebs-2018-01-14
  machineType: m4.xlarge
  maxSize: 20
  minSize: 16
  nodeLabels:
    kops.k8s.io/instancegroup: nodes
  role: Node
  subnets:
  - eu-west-1a
```

KOPS: Validate cluster



```
INSTANCE GROUPS
NAME                    ROLE    MACHINETYPE     MIN     MAX
admin                   Node    m3.medium       1       1
master                  Master  m4.large        1       1
monitor                 Node    m4.large        1       1
nodes                   Node    m4.xlarge       16      20

NODE STATUS
NAME                                            ROLE    READY
ip-172-20-50-127.eu-west-1.compute.internal     master  True
ip-172-20-50-144.eu-west-1.compute.internal     node    True
ip-172-20-50-170.eu-west-1.compute.internal     node    True
ip-172-20-50-179.eu-west-1.compute.internal     node    True
ip-172-20-50-37.eu-west-1.compute.internal      node    True
ip-172-20-50-53.eu-west-1.compute.internal      node    True
```

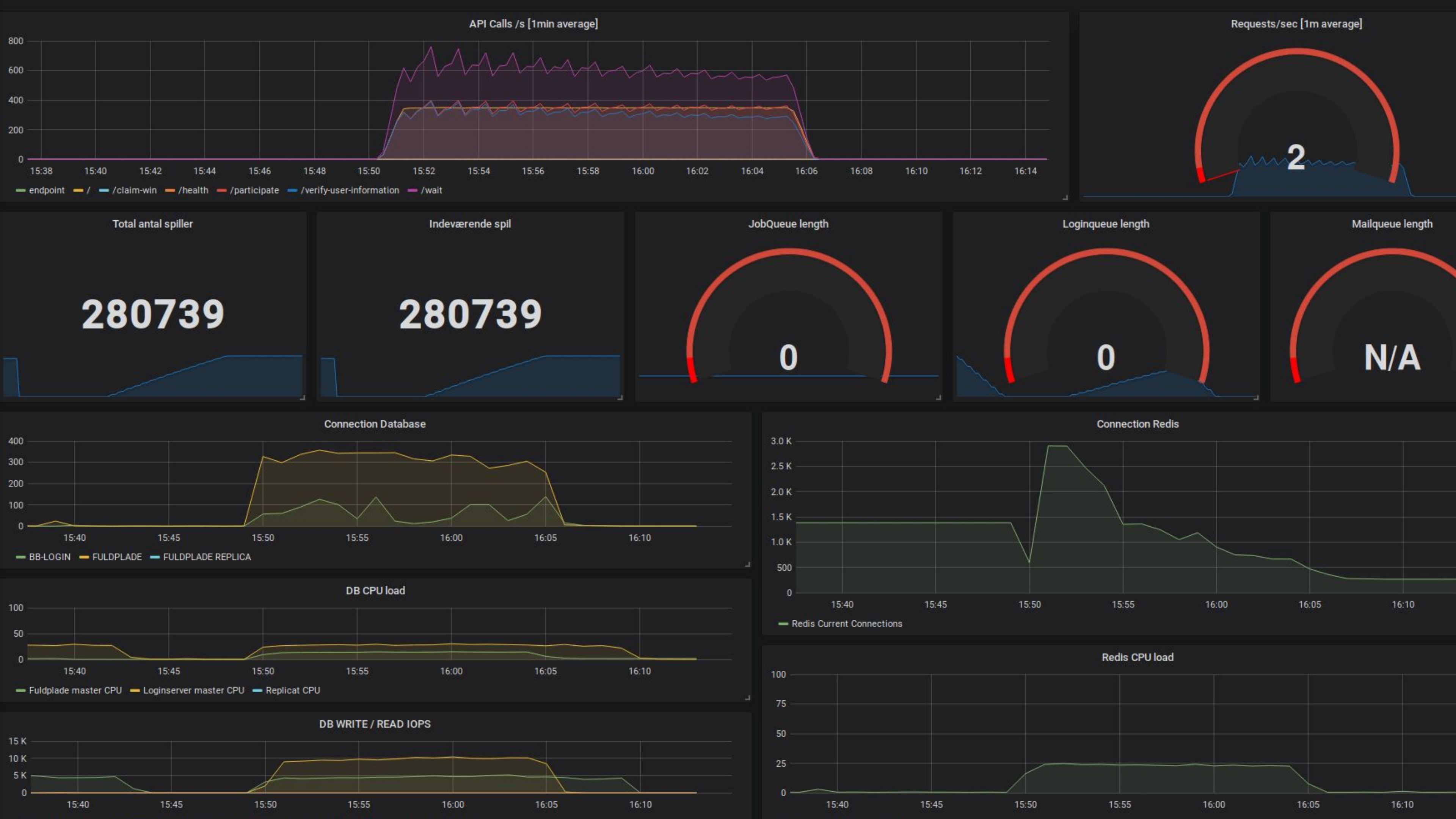
Monitoring

Prometheus autodiscovery

Grafana dashboards

Graylogs search and dashboards

Grafana Prometheus graylog

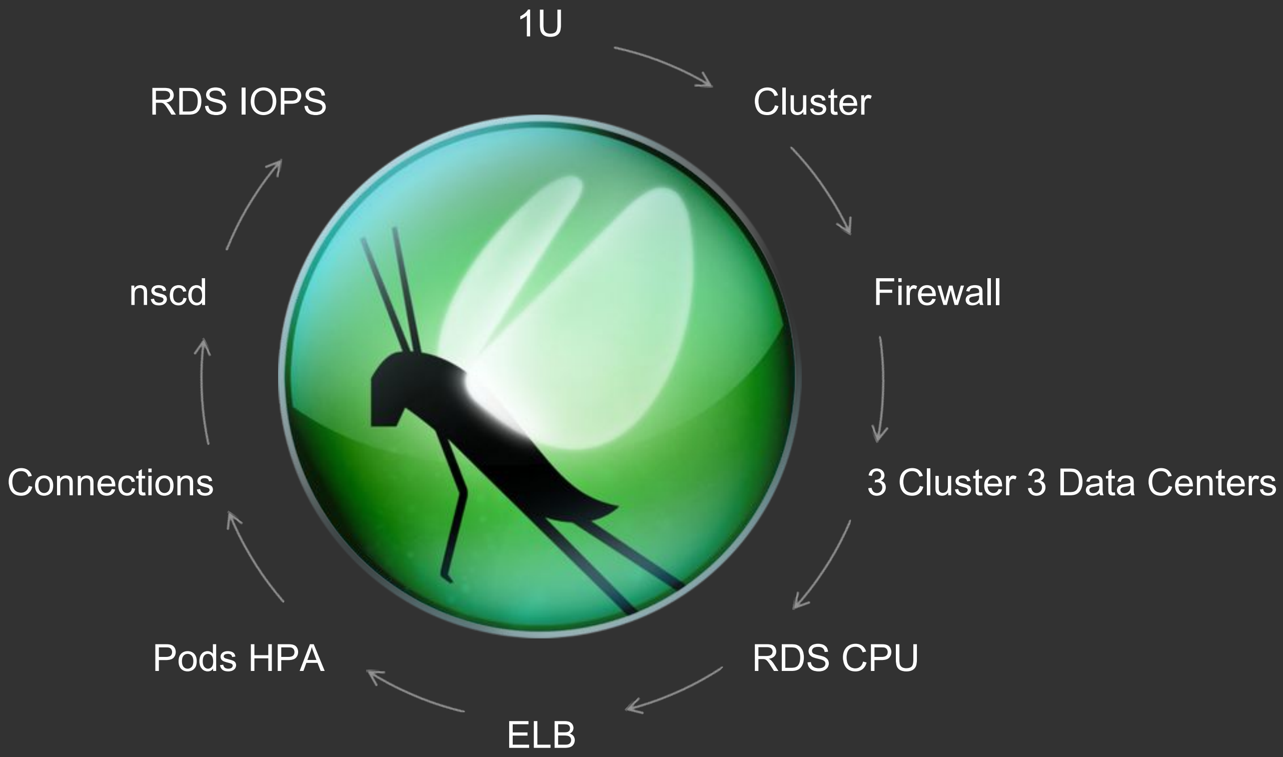

API Calls /s [1min average]
800
600
400
200
0
15:38
15:40
15:42
15:44
15:46
15:48
15:50
15:52
15:54
15:56
15:58
16:00
16:02
16:04
16:06
16:08
16:10
16:12
16:14
endpoint
/
/claim-win
/health
/participate
/verify-user-information
/wait
Requests/sec [1m average]
2
Total antal spiller
280739
Indeværende spil
280739
JobQueue length
0
Loginqueue length
0
Mailqueue length
N/A
Connection Database
400
300
200
100
0
15:40
15:45
15:50
15:55
16:00
16:05
16:10
BB-LOGIN
FULDPLADE
FULDPLADE REPLICA
Connection Redis
3.0 K
2.5 K
2.0 K
1.5 K
1.0 K
500
0
15:40
15:45
15:50
15:55
16:00
16:05
16:10
Redis Current Connections
DB CPU load
100
50
0
15:40
15:45
15:50
15:55
16:00
16:05
16:10
Fuldplade master CPU
Loginserver master CPU
Replicat CPU
Redis CPU load
100
75
50
25
0
15:40
15:45
15:50
15:55
16:00
16:05
16:10
DB WRITE / READ IOPS
15 K
10 K
5 K
0
15:40
15:45
15:50
15:55
16:00
16:05
16:10

Performance test

Performancetest: LOCUST

- Test as code
- Highly scalable
- Less hardware consuming

LOCUST: Fast Forward



1U

Cluster

Firewall

3 Cluster 3 Data Centers

RDS CPU

ELB

Pods HPA

Connections

nscd

RDS IOPS

Evaluation & conclusions

Evaluation

Kubernetes and docker for fast scaling

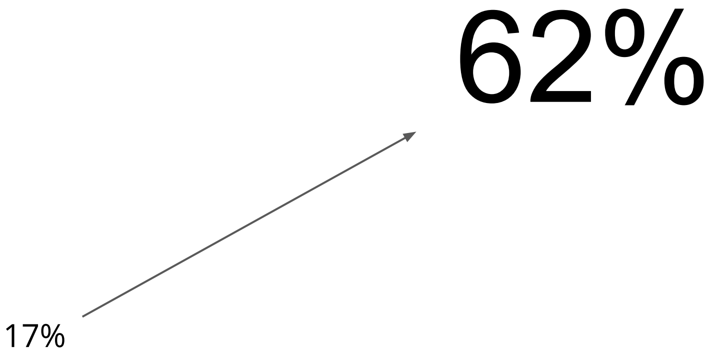
Low price tag on operations

No bottlenecks throughout the season

peytz&co

Record breaking

Second screen interaction



62%

17%

peytz&co

Conclusion

- Documentation and procedures
- Prometheus
- Scaling
- Best practice
- Locust

Thank you for the attention

Any questions?

Thomas Hector

th@peytz.dk

in @thomas-rasch-hector

Jan-Erik Revsbech

jer@peytz.dk

in @janerikrrevsbech

peytz&co