
# AWS UG Feb Meetup

2015-02-12

henning.jacobs@zalando.de

zalando

# One of Europe's largest online fashion retailers

**15** countries

**14+ million** active customers

**2.2 billion €** revenue 2014

**640+ million** visits in Q1/2 2014

# In Germany

More than 5000

Employees


Zalando
SOLr
SHOP
MOBILE
LOUNGE
3rd PARTY
SERVICE 2
SERVICE 1
SERVICE 3
SERVICE 5
SERVICE 6
TOOLING
POSTGRE SQL
BI
LOGISTICS

zalando

# Some more numbers

**200+** deployment units (WARs)

**1300+** production Tomcat instances

**80+** database master instances

**300+** developers and 200+ QA and PM

zalando

# Our Situation

- ~40 development teams
- Each team should
  - be able to **use any AWS service**
  - have an **isolated sandbox**
  - be able to **access services** of other teams
  - have **separate billing** & accounting

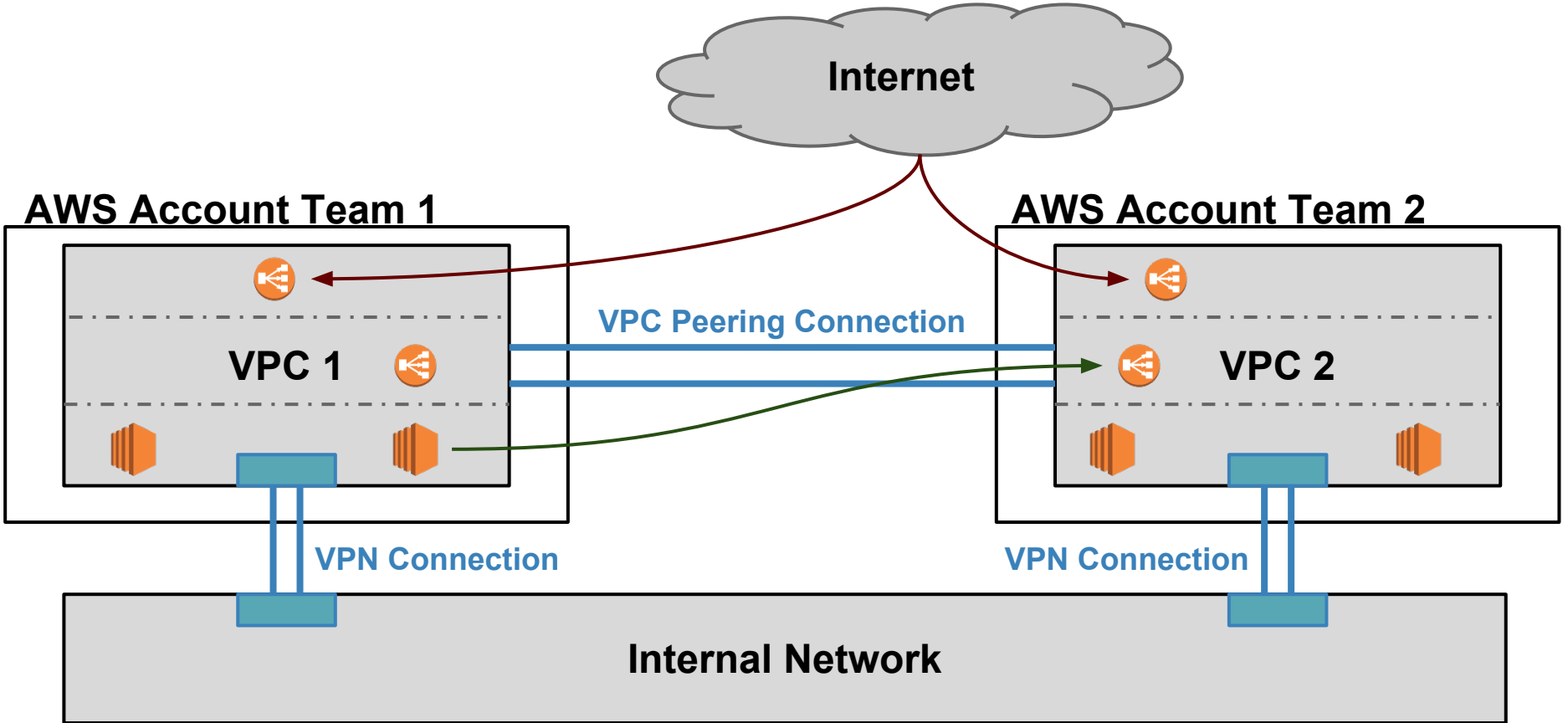
zalando

# Our Approach

- One AWS **account per team**
- Every team has “**full rights**” in their account
- Some common infrastructure
  - Docker registry
  - audit/monitoring
  - security

zalando

# Our Approach



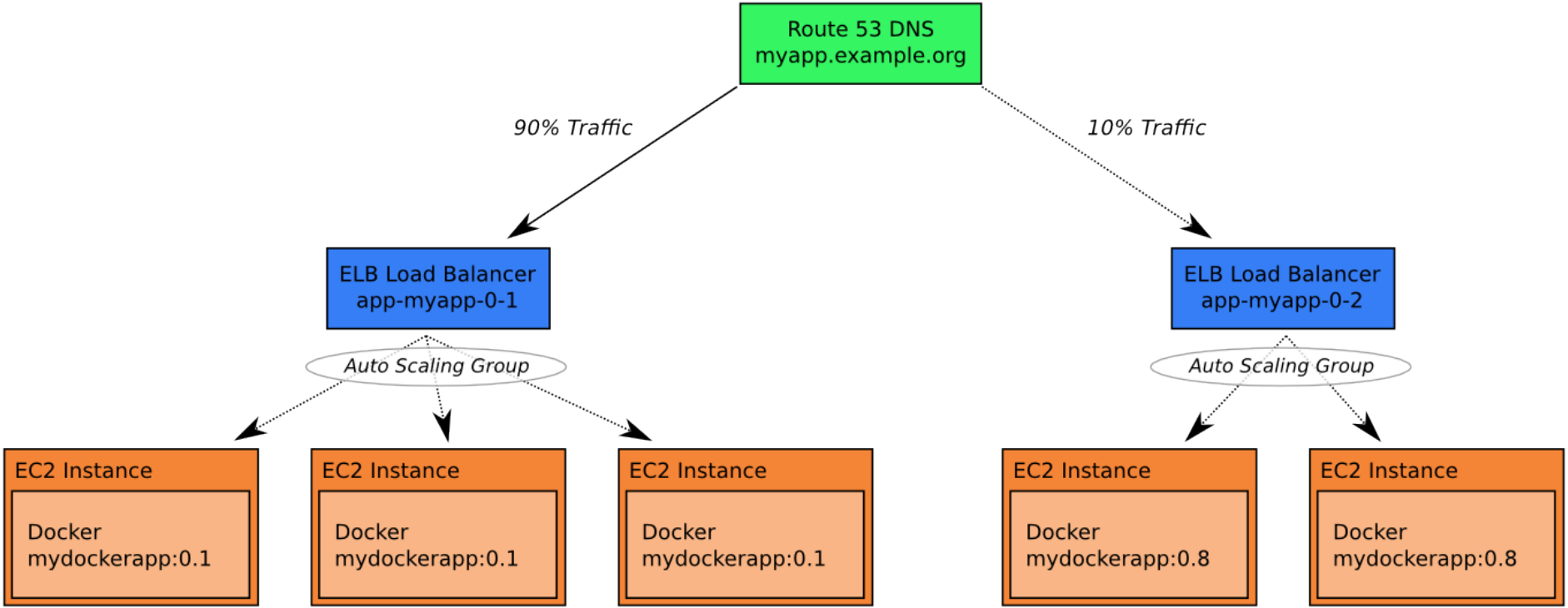
zalando

# Our current AWS Setup

- 13 different AWS team accounts
- Mostly application prototypes
- Some internal production use cases

Currently working on infrastructure tooling...

# AWS Minion


Route 53 DNS
myapp.example.org
90% Traffic
10% Traffic
ELB Load Balancer
app-myapp-0-1
ELB Load Balancer
app-myapp-0-2
Auto Scaling Group
Auto Scaling Group
EC2 Instance
Docker
mydockerapp:0.1
EC2 Instance
Docker
mydockerapp:0.1
EC2 Instance
Docker
mydockerapp:0.1
EC2 Instance
Docker
mydockerapp:0.8
EC2 Instance
Docker
mydockerapp:0.8

zalando

# Minion Workflow

- **Create** application
- **Deploy** application version
- **Route traffic** to new application version

# Create Application

```
~/workspace/clojure-web-hello-world (master) $ minion app create aws-minion-manifest.yaml
Checking whether application clojure-web-hello-world exists.. OK
Creating key pair for application clojure-web-hello-world.. OK
Creating application security group app-clojure-web-hello-world.. OK
Creating LB security group app-clojure-web-hello-world-lb.. OK
Creating IAM role and instance profile.. OK
```

- Creates **Security Group**
- Creates **IAM Role**

# Build & Push Docker Image

```
$ docker build -t
registry.example.org/hjacobs/hello-world:0.1 .
```

```
$ docker push registry.example.org/hjacobs/hello-world:0.1
```

```
~/workspace/clojure-web-hello-world (master) $ minion images
```

| Repository | Tag | Image |
|---|---|---|
| ahanin/ahanin-helloworld | 0.1 | ahanin/ahanin-helloworld:0.1 |
| ahanin/ahanin-helloworld | latest | ahanin/ahanin-helloworld:latest |
| ahanin/ahanin-heloworld | 0.1 | ahanin/ahanin-heloworld:0.1 |
| brandsolutions/analytics-play | 0.1 | brandsolutions/analytics-play:0.1 |
| ci_cd/jenkins | 0.01 | ci_cd/jenkins:0.01 |
| hackweek/christophs-playground | 1.0 | hackweek/christophs-playground:1.0 |
| hackweek/helloworld | 1.0-hseffler-SNAPSHOT | hackweek/helloworld:1.0-hseffler-SNAPSHOT |
| hackweek/helloworld | latest | hackweek/helloworld:latest |
| hjacobs/clojure-web-hello-world | 0.1 | hjacobs/clojure-web-hello-world:0.1 |
| hjacobs/docker-registry | 0.9.0hjacobs1 | hjacobs/docker-registry:0.9.0hjacobs1 |
| **hjacobs/hello-world** | **0.1** | **hjacobs/hello-world:0.1** |
| hjacobs/opengrok | 0.1 | hjacobs/opengrok:0.1 |
| hjacobs/registry | 0.8.1 | hjacobs/registry:0.8.1 |
| hjacobs/techblog | 1.0 | hjacobs/techblog:1.0 |

# Deploy Application Version

```
~/workspace/clojure-web-hello-world (master) $ minion ver create clojure-web-hello-world 2.0 hjacobs/hello-world:0.1
Checking Docker registry [redacted].. OK
Creating launch configuration for clojure-web-hello-world version 2.0.. OK
Creating load balancer for clojure-web-hello-world version 2.0.. OK
Creating auto scaling group for clojure-web-hello-world version 2.0.. OK
DNS name of load balancer is internal-app-clojure-web-hello-world-2-0-1088772597.eu-west-1.elb.amazonaws.com
Configuring DNS name clojure-web-hello-world-2-0.[redacted] .. OK
Waiting for instance start and LB.. . . . . OK
Waiting for LB members to become active.. . . . . OK
Application version URL is https://clojure-web-hello-world-2-0.[redacted]
```

- Creates **Auto Scaling Group**
- Creates **Load Balancer**
- Creates **DNS** myapp-<VER>.example.org

# Version & Instances

```
~/workspace/clojure-web-hello-world (master) $ minion ver
Application Name        Ver.  Docker Image              Instance States  Desired#  Weight  Created
clojure-web-hello-world 2.0   hjacobs/hello-world:0.1   1x InService     1                 9m ago
```

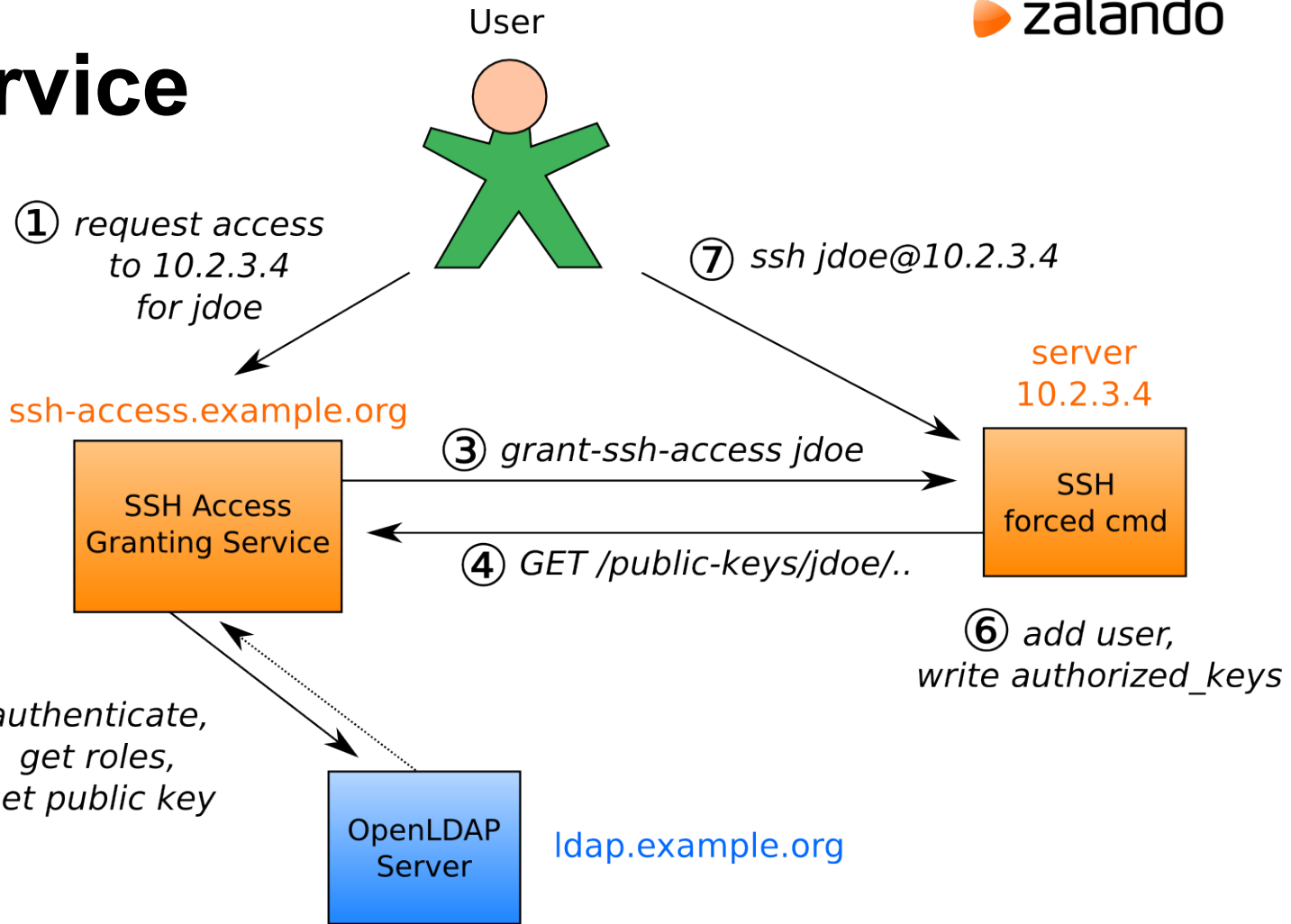
```
~/workspace/clojure-web-hello-world (master) $ minion instances
Application Name        Ver.  Instance Id  Team     Public Ip  Private Ip     State    Launched
clojure-web-hello-world 2.0   i-fc0ded18  hjacobs             10.144.81.204  RUNNING  12m ago
```

# Route Traffic

```
~/workspace/clojure-web-hello-world (master) $ minion ver traffic clojure-web-hello-world 2.0 100
Calculating new weights.. OK
| Application Name | Version | Identifier | Old Weight | Delta | Compensation | New Weight | Current |
| clojure-web-hello-world | 2.0 | app-clojure-web-hello-world-2-0 | 0.0 | 100.0 | | 100.0 | < |
Setting weights for clojure-web-hello-world.[redacted]... OK
```

- Changes **weights in Route 53**
- Exposes application as **myapp.example.org**

# SSH Service


User
① request access to 10.2.3.4 for jdoe
⑦ ssh jdoe@10.2.3.4
server 10.2.3.4
ssh-access.example.org
SSH Access Granting Service
③ grant-ssh-access jdoe
④ GET /public-keys/jdoe/..
SSH forced cmd
⑥ add user, write authorized_keys
② authenticate, get roles,
⑤ get public key
OpenLDAP Server
ldap.example.org

# How to get access to a single server

```
$ curl \
  -u jdoe \
  -H Content-Type:application/json \
  -d '{"hostname": "10.1.2.3",
       "reason": "troubleshooting app XY"}' \
https://ssh-access.example.org/access-requests

"Access to host 10.1.2.3 for
 user jdoe was granted."
```

# Work in progress...

- **AMI** with custom cloud config
- **Audit** tooling
- **Logging** / **monitoring** infrastructure
- **Security** framework for Service-to-Service authentication

zalando

# Links

**AWS Minion** – Manage Docker apps on EC2
https://github.com/zalando/aws-minion

**AWS Overlord** – Account infrastructure management tool
https://github.com/zalando/aws-overlord

**SSH Access Granting Service** – Distribute SSH public keys
https://github.com/zalando/ssh-access-granting-service

**AWS AMI Creator** – Simple script to create AMIs
https://github.com/zalando/aws-ami-creator

zalando

# Discussion groups

- **VPN & VPC**: Andreas
- **Security**: Tobi
- **Storage**: Hendrik
- **Deployment**: Henning
- **Beginners**: Rodrigo