Architecting for the

Cloud

@axelfontaine
Let us know what you think

Click ‘engage’ to rate sessions and ask questions

Follow us on Twitter @GOTOber
About Axel Fontaine

• Founder and CEO of Boxfuse
• Over 15 years industry experience
• Continuous Delivery expert
• Regular speaker at tech conferences
• JavaOne RockStar in 2014

@axelfontaine
about

questions
POLL:
what type of infrastructure are you running on?

• On Premise
• Colocation
• Root Server
• Cloud
what is special about the cloud ??
Every day, AWS adds enough server capacity to power the whole $7B enterprise Amazon.com was in 2004. Weekends included.
Control Plane

Data Plane


"Advanced Test Reactor" by Argonne National Laboratory - originally posted to Flickr as Advanced Test Reactor core, Idaho National Laboratory. Licensed under CC BY-SA 2.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Advanced_Test_Reactor.jpg
benefits of the cloud

✓ Shift to a world of abundance (no more resource scarcity)

✓ Clean Control Plane/Data Plane split with API-based provisioning

✓ Cost-based Architectures with the ability to turn infrastructure off
moving to the cloud
lift & shift
(= the naïve approach)
Congratulations! You now have:

- A more expense Hetzner/OVH
- Lots of (too much?) trust in your cloud provider
- Potential legal trouble due to data privacy laws

lift & shift
(= the naïve approach)
understanding the cloud
availability zones
building blocks
The hard **Truth** about Security

1. Always breakable with infinite time & resources

2. Must make it more complicated/expensive to break than it’s worth (use defense in depth!)

3. Has a usability cost

4. Almost always about the data
Trusting your neighbors is good. But it’s even better to put a good lock on the door.

Werner Vogels
CTO of an online book shop
Data in Motion

TLS / SSL
Data in Use & at Rest

Client-side encryption
Client-side encryption

- Encrypt sensitive & personally identifiable data
- Use different Encryption key for each field/record
- Encrypt Encryption Key using Key encrypting Key
- Secure & Rotate the Key encrypting Key
Key Management

- In App: €
- KMS: €€
- HSM: €€€€€
Querying Encrypted Data

Other clear text field

<table>
<thead>
<tr>
<th>Id</th>
<th>Encrypted</th>
<th>123</th>
<th>#!azw\b</th>
</tr>
</thead>
<tbody>
<tr>
<td>456</td>
<td>67ftf6&amp; )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Exact Match => Hmac

<table>
<thead>
<tr>
<th>Hmac</th>
<th>Encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>5841545832</td>
<td>#!azw\b</td>
</tr>
<tr>
<td>0219237127</td>
<td>67ftf6&amp; )</td>
</tr>
</tbody>
</table>

Range => Lower fidelity

<table>
<thead>
<tr>
<th>Low Fi</th>
<th>Encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>48.5</td>
<td>#!azw\b</td>
</tr>
<tr>
<td>37.2</td>
<td>67ftf6&amp; )</td>
</tr>
</tbody>
</table>

=> Use transparent persistence layer converters!
Compute
POLL:
which level of automation are you at?

• Build
• Unit Tests
• Continuous Integration
• Acceptance Tests
• Continuous Deployment (Code)
• Continuous Deployment (Code + DB + Configuration)
• Infrastructure
• One immutable unit
• Regenerated after every change
• Promoted from Environment to Environment

Classic Mistake: Build per Environment
Image ➔ Fully Baked ➔ Provisioned on Startup ➔ Instance

Question mark
- Every Instance 100% identical
- Fastest startup
- Launch always succeeds

**NETFLIX**

- Fully Baked
- Provisioned on Startup

**Most people**
✔ One immutable unit
✔ Regenerated after every change
✔ Promoted from environment to environment

NETFLIX

Most people

Fully Baked

Provisioned on Startup
✓ One immutable unit
✓ Regenerated after every change
✓ Promoted from environment to environment

Image

Fully Baked
✓ One immutable unit
✓ Regenerated after every change
✓ Promoted from environment to environment
- One immutable unit
- Regenerated after every change
- Promoted from environment to environment

Image

Fully Baked
keep your instances stateless
high uptime is a liability

The longer an instance is up, the harder it becomes to recreate exactly (and it will fail eventually!)
Focus shift

Instance → Service

Individual instances become disposable
Treat servers like **cattle** instead of pets
What are the implications ???
scaling

Image → Instance
types of scaling

- up
- down
- in
- out
scaling triggers for different types of services

- sync => load
- async => queue depth
- cron => time
scaling & costs

prefer **smaller granularity**
instance types

General Purpose

CPU

RAM

Disk
How to solve service discovery?

Use a stable entry point with an internal registry.
what about configuration???

- Bake as much configuration as possible for all environments directly in the Image

- Use environment detection and auto-configuration

- Pass remaining configuration at startup and expose it as environment variables

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC_URL</td>
<td>jdbc:...</td>
</tr>
<tr>
<td>ENV</td>
<td>prod</td>
</tr>
</tbody>
</table>
what about the database???

- Keep persistent state out of the instance, including the database
- Use one of the many good hosted solutions available like Amazon RDS or Google Cloud SQL
- Use a database migration tool to update the schema on application startup
what about the logs ???

ssh me@myserver1
tail -f server.log

ssh me@myserver2
tail -f server.log

ssh me@myserver3
tail -f server.log
Ship logs to a **central log server**

where they can be

• aggregated
• stored and backed up
• indexed
• searched through a nice web UI

Many good hosted solutions

• Loggly
• Logentries
• Papertrail
• ...

=> Think about **data privacy!**
what about sessions ???

Keep session in an encrypted and signed cookie

• avoids session timeouts
• avoids server clustering & session replication
• avoids sticky sessions & server affinity
what about rolling out new versions???
what about containers ???
understanding modern CPUs

Both Intel and AMD have hardware support for virtualization

- isolation
- performance
Hardware

VM

Image
Hypervisor
Hardware

Image
OS+Container
Runtime
Hardware

Container

on prem

your responsibility
cloud

Only makes sense if you cannot afford 8.75€/month granularity

container images | container scheduling | containers | container volumes | container networking

machine images | instance scheduling | instances | instance volumes | instance networking

your responsibility

cloud responsibility
cloud

Only makes sense if you cannot afford 0.01€/hour granularity

container images
container scheduling
containers
container volumes
container networking

your responsibility

machine images
instance scheduling
instances
instance volumes
instance networking

cloud responsibility
summary

- Put a good **lock on the door** (use encryption!)
- Use **fully baked** images (build once!)
- Treat servers like **cattle** (disposable!)
• Fully baked images generated in seconds (not minutes or hours)

• Optimized for JVM apps (Spring Boot, Dropwizard, Tomcat, TomEE, ...)

• Minimal images just 1% of size of regular OS (measured in MB not GB)

• Images work on VirtualBox & AWS (environment parity from dev to prod)

• Zero downtime updates on AWS (fully automatic blue/green deployments)
final disclaimer

no animals were harmed while making this talk 😊
Please remember to rate session.

Thank you!
Thanks !