Immutable Infrastructure

The New App Deployment

AXEL FONTAINE
@axelfontaine
axel@boxfuse.com
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
Flyway
flywaydb.org
Let’s start with a small story.
Heater that gives off a little bit of light

Light that gives off a little bit of heat
Edison Screw
Simple, stable, standards-compliant interface with a clear contract

My responsibility

The electricity company’s responsibility
Simple, stable, standards-compliant interface with a clear contract
POLL:
what type of infrastructure are you running on?

• On Premise
• Colocation
• Root Server
• Cloud
How did this evolve?
sometime in the 20th century ...
Challenges

- Power, Network, Cooling
- Physical Security
- Physical Space
- Procurement, Vendor Management
- Capacity Planning
- Financing
- OS + Patches
- App + Updates
ON PREM = + +

Our responsibility
Our responsibility + COLO = + Their responsibility
COLO = Windows NT Server + [Server] + [Server]

Simple, stable, standards-compliant interface:

(19” Rack, AC Power, Ethernet, ...)
COLO = Windows NT Server + [image of server] + [blue box labeled: Undifferentiated Heavy Lifting] 

Our responsibility can change as long as it complies with the interface contract.
ROOT SERVER

\[\text{ROOT SERVER} = \text{Undifferentiated Heavy Lifting} + \text{Our responsibility}\]

Our responsibility can change as long as it complies with the interface contract.
ROOT SERVER = Software <-> Hardware

+ Undifferentiated Heavy Lifting

Simple, stable, standards-compliant interface
Could this be our industry’s **Edison Screw**?

Simple, stable, standards-compliant interface
Let’s talk about software
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
why aren’t we doing the same for the layers this is running on ????
what could possibly go wrong in these other layers ???
missing software
wrong name
bad version
incorrect permissions
critical resource in use
what aren’t we holding our servers to the same standards as our applications ???
Multiple instances in multiple Environments
Multiple instances in multiple **Environments**

- All instances should be as similar as possible (any difference is a potential source of errors)
- That also includes your local Dev environment!
- Must be able to reliably provision new ones (and recreate existing ones from scratch)
If I had asked my customers what they wanted they would have said a faster horse.

Henry Ford
fast forward to 2015 ...
Every day, AWS adds enough server capacity to power the whole $5B enterprise Amazon.com was in 2003. Weekends included.
Control Plane

Data Plane
Control Plane

Data Plane
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
it is time to rethink the faster horse
but there is one big problem left ...
Multiple GB Network Cable
Running servers in production should be like going **backpacking**. You take the bare minimum with you. Anything else is going to hurt.

A Wise Man
what is really adding **business value** ???
Multiple GB

40 – 80 MB
Bootable App → Network Cable
who is this for ???
12-factor app
demo
What are the implications ???
Focus shift

Instance  ➡️  Service

Individual instances become disposable
Treat servers like *cattle* instead of pets
high uptime is a liability

The longer an instance is up, the harder it becomes to recreate exactly (and it will fail eventually!)
How to solve service discovery?

Use a stable entry point with an internal registry.
What about security?

When was the last time your toaster got hacked?
What about security?

- Smallest possible attack surface
- Vastly reduced implications due to low uptime and transient nature of instances
- Very difficult to exploit other systems because essential tooling is missing
what about configuration???

- Bake as much configuration as possible for all environments directly in the Bootable App
- Use environment detection and auto-configuration
- Pass remaining configuration at startup and expose it as environment variables
what about the database ???
what about the **database** ???

Bootable App

---

Flyway

Database
what about the **database** ???

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

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
• …
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
Bootable App

Hypervisor

Hardware

On Prem / Cloud

Bootable App

OS+Container Runtime

Hardware

On Prem

Bootable App

OS+Container Runtime

Hypervisor

Hardware

Cloud

Only makes sense if you cannot afford $9.60/month granularity
Bootable App

Hypervisor

Hardware

On Prem / Cloud

Bootable App

OS+Container Runtime

Hardware

On Prem

Bootable App

OS+Container Runtime

Hypervisor

Hardware

Cloud

Only makes sense if you cannot afford 1.3 cents /hour granularity
summary
• One immutable unit
• Regenerated after every change
• Promoted from Environment to Environment

Classic Mistake: Build per Environment
Bootable App

- One immutable unit
- Regenerated after every change
- Promoted from Environment to Environment

Classic Mistake: Build per Environment
Thanks !