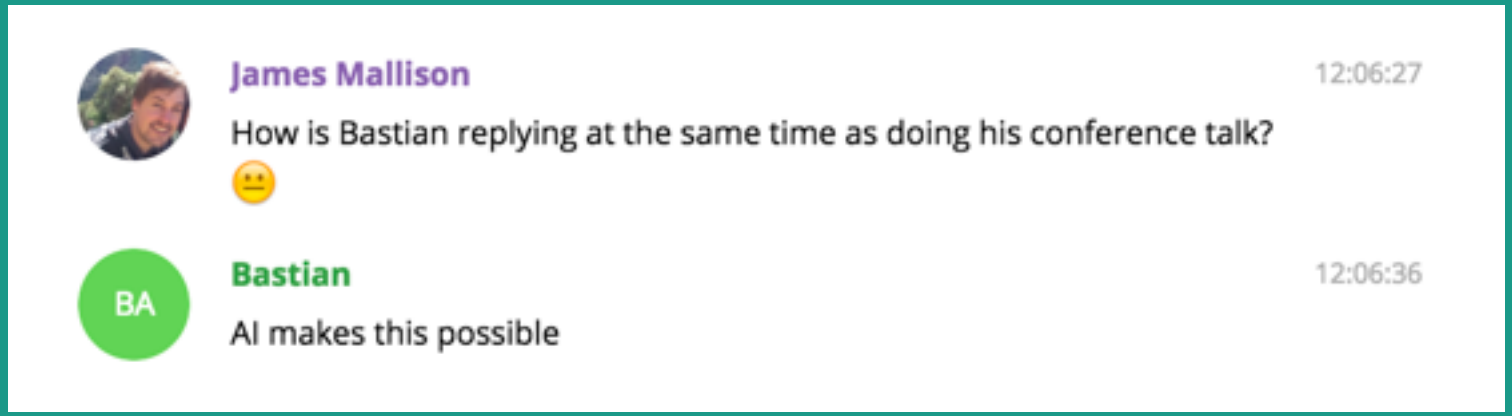
Enter a new world of web development where everything is serverless

What's possible and how will infrastructure be shaped in the future

PHPDay 2018 Vernona
Bastian Widmer - @dasrecht | @amazeeio

Hi I'm Bastian

Hi I'm Bastian - the other one



$> whoami bastian

- System Engineer at amazee.io
- Containers in Production 👻🤖
- Zurich, Switzerland
- @dasrecht
- Too many side projects!*
  - TEDxBern
  - DevOpsDays Zurich
  - CommunityRack.org
  - Running TOR nodes for fun
  - Working with real containers

* this list is not complete!

$>


Caldeira
1633
Toggenburger

We will talk about: serverless, concepts, containers, infrastructure and modern software architecture

Back in the day™

Back in the day™

- Editor
- Storage Medium
- FTP Client
- Server
- Browser

What about today?

What about today?

- New technologies
- Containers
- Microservices
- Security
- Ops not being fast enough to catch up with Dev?

COMPLEXITY

Wouldn't it be cool to define our infrastructure directly in our project repository?

— one of my colleagues, 2012

2015

- Full rebuild of the hosting infrastructure
- Configuration management
- Per-Commit Deployments
- Local Dev Environment built with the same tools based on vagrant

2016

- It’s not flexible enough
- VMs use a lot of space

- Let’s give Docker a chance
- Building tooling around Docker - pygmy

- **Why using Containers just locally?**

Pygmy: https://github.com/amazeeio/pygmy

2017
2018

- 2017
  - First Website running on Docker!
  - Eureka! This actually works!
  - Pull-Request Environments
  - Open Source?
  - Open Source! - Lagoon
- 2018
  - Actively migrating customers over to lagoon
  - Working towards V1.0.0 Release

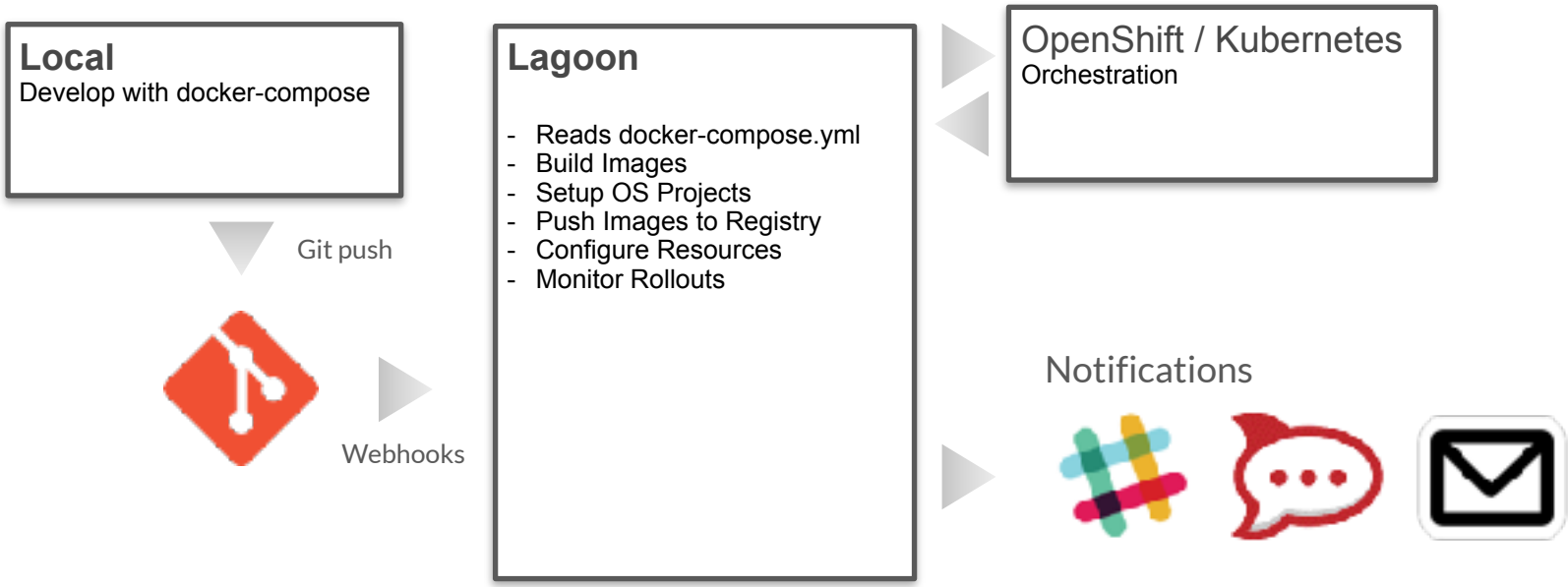
Lagoon?

- 4. Iteration of our hosting stack
- Fully opensourced
- On top of Kubernetes / Openshift
- Microservices
- Deployment pipeline for web projects
- Local development environment
- Infrastructure/Services defined in code

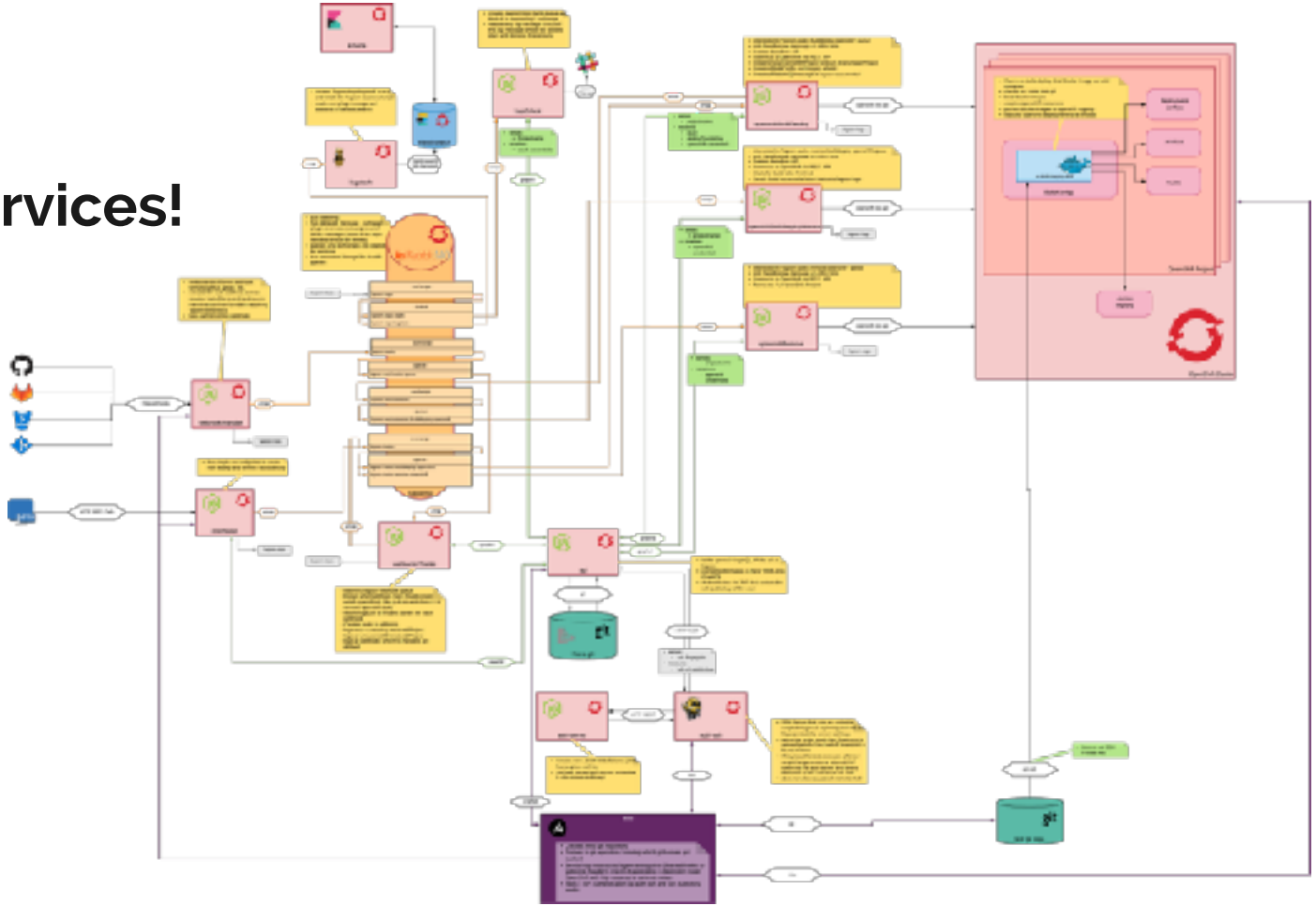
- https://github.com/amazeeio/lagoon

TL;DR

More details



Microservices!



A word of wisdom: Complexity of orchestrated systems

- If you don’t master traditional infrastructure - You will have a hard time
- Orchestrated microservices are a living thing
- You still need skilled people that know how to engineer on top of cloud services
- (and you don’t get rid of servers - you just treat them differently)

Spoiler: There are Platforms as a Service for that

- Build on top of opensource projects like Kubernetes and Openshift
- Kubernetes has become the de-facto standard in orchestrating containers
- Every big Cloud Vendor has a Kubernetes (K8S) offering

- Functions as a Service in Kubernetes: OpenFAAS, Fission, Kubeless

Serverless?

Spoiler alert: Serverless still involves servers!

Hard Truth: You might not go all serverless

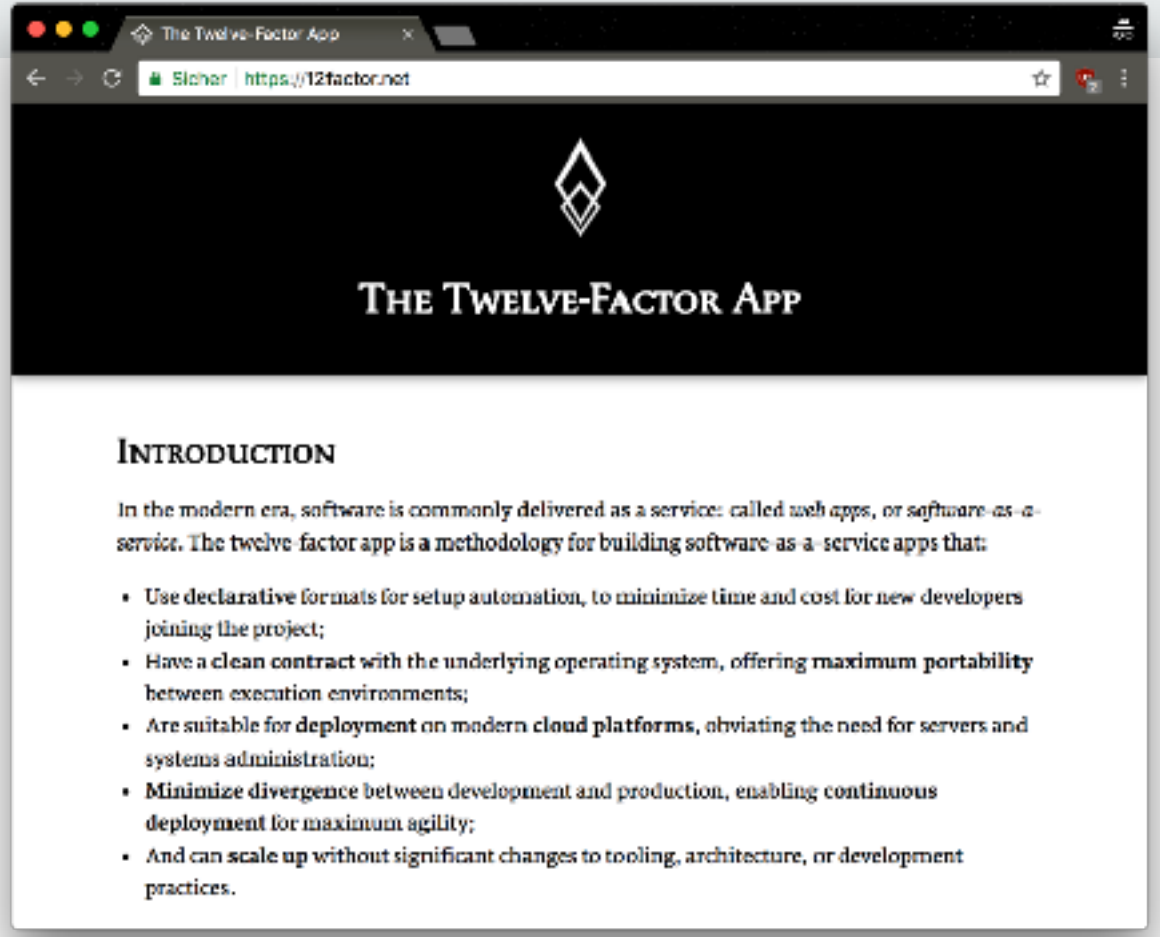
Hard Truth: You might not go all serverless

unless:

Hard Truth: You might not go all serverless

unless:
- You can pull of a full rebuild
- Your service is stateless
- You follow the 12Factor methodology

12factor.net



Serverless

A Serverless solution is one that costs you nothing to run if nobody is using it (excluding data storage)

Paul Johnston : https://medium.com/@PaulDJohnston/a-simple-definition-of-serverless-8492adfb175a

Functions as a Service
vs.
Serverless

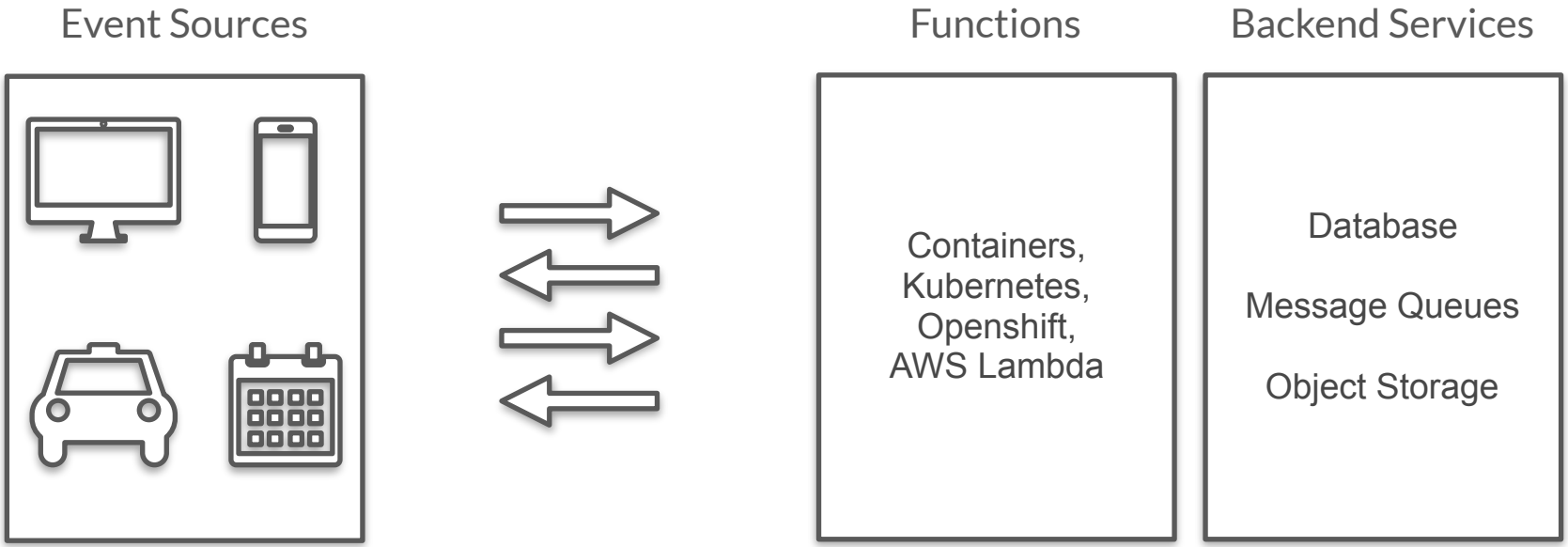
Functions as a Service (FaaS)

- Concept
- You write the code
- Event-based execution
- Stateless
- Short-lived (Milliseconds — Minutes)
- Pay per use
- Uses a compute service to run the code (no servers)
- Most of the cloud vendors have proprietary approaches to FaaS

Functions as a Service (FaaS)

- Deployscript triggers a function saving the logs to an object storage
- Someone calling our emergency phone - Triggers a function and forwards the phonecall
- You left your stove on for too long and an alert gets sent out to your phone
- You finished a workout and your armband talks to an api and saves the data.

Functions as a Service (FaaS)

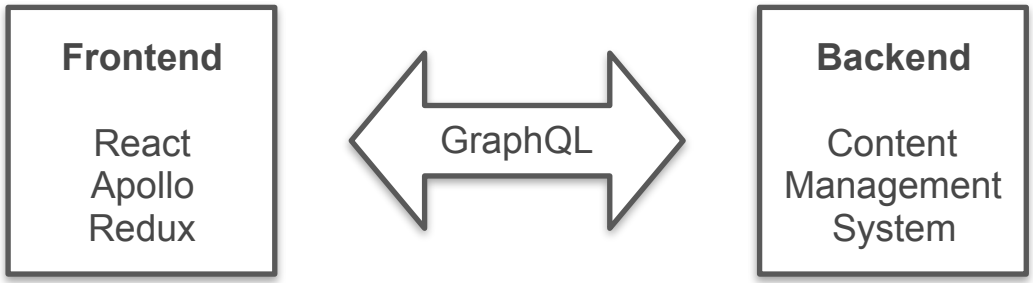


Serverless

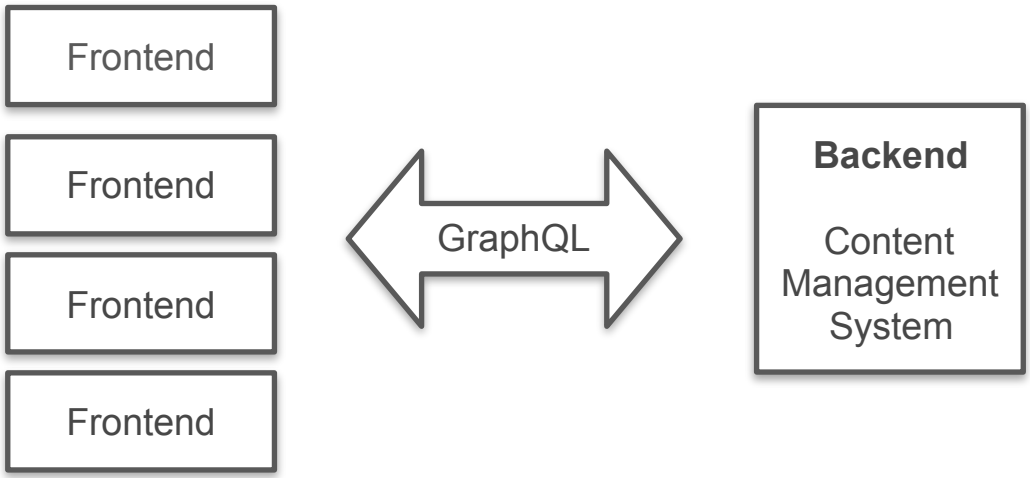
FaaS but enriched by:

- Auto-scaling based on demand
- Scaling down to zero running instances when it’s not used
- You don’t need to worry about scaling
- Embrace third-party services
- Loosely coupled

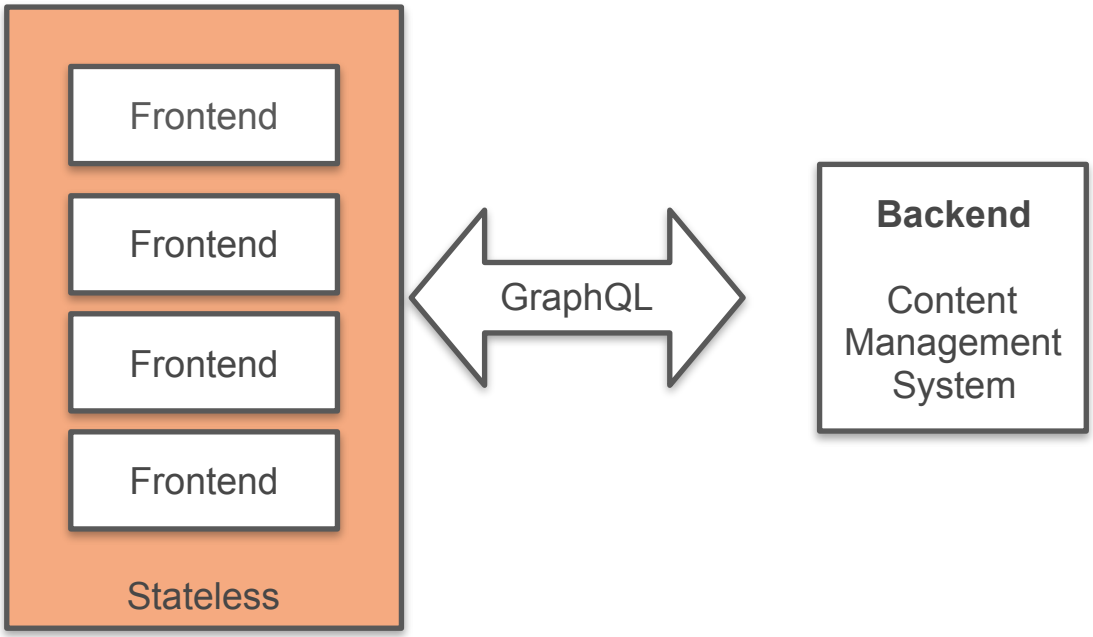
Loose Coupling / Decoupling



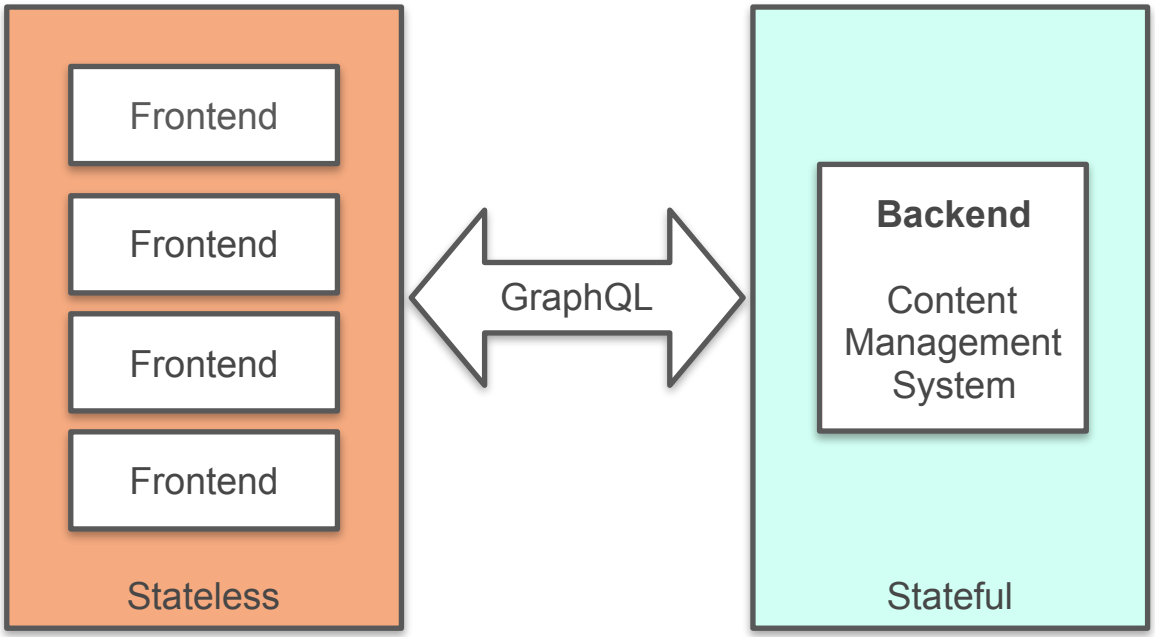
Loose Coupling / Decoupling



Loose Coupling / Decoupling



Loose Coupling / Decoupling



Serverless-ish

As soon as you have traditional stateful applications you will not have a serverless application.

We host websites with databases. So much for serverless?

Let’s call it Serverless-*ish*

Serverless-ish

But, there are no running costs beside the storage if you don’t use the application. We remove the containers.

And spin them back up if there’s demand for it.

We are on our way. It’s a journey after all.

FUTURE

In the cloud, traditional concepts don't hold up anymore.

Don’t get attached to your infrastructure!

Don't get attached to your infrastructure!
Don't give your servers names!

Don't get attached to your infrastructure!
Don't give your servers names!
Never!

Don't get attached to your infrastructure!
Don't give your servers names!
Never!

Seriously...Don't!

But why?

Pets vs. Cattle Metaphor

Pets

- Sometimes manually built
- Have names „webserver“, „billing“
- Are managed with care
- If they fail people are sad
- Think about it like your office coffee machine

Cattle/Herd

- Built via automation
- web01, web02, web03, web04
- Managed automatically
- Self-healing / orchestrated
- If they fail, they get replaced

Monitoring?
Uptime?

Monitoring

- Is my service reachable?
- Is the usage pattern within set boundaries (response time, response codes)
- Anomalies? Act on out-of-band errors
- Negative Uptime - Is the container/function running for too long?

Take aways

Serverless - The takeaways:

- Automate, Experiment, Fail, Learn, Share
- Serverless is an approach to modern application design
- Embrace third-party services wherever possible
- Don’t get attached to your infrastructure (Cloud-Computing isn’t a one time thing)
- Make small steps (e.g. break up your application in several services)
- Start with the parts that don’t hurt you if they fail
- Build on opensource and if you can, also opensource your things
- Work towards the 12 factor app methodology

Thank you for your attention!

Bastian Widmer - @dasrecht | @amazeeio

Rate my talk:

http://s.nrdy.ch/phpday-verona

Resources

- WS_FTP Pro Screenshot - https://www.cnet.com/products/ws-ftp-pro-7-5/review/
- IE Screenshot - https://guidebookgallery.org/splashes/internetexplorer
- Serverless https://martinfowler.com/articles/serverless.html
- Cattles and Pets - https://twitter.com/noggin143/status/354666097691205633
- Cattles and Pets - http://cloudscaling.com/blog/cloud-computing/the-history-of-pets-vs-cattle/
- The Power of Serverless - https://thepowerofserverless.info/
- 12 Factor App - https://12factor.net/

Further Readings

- https://loige.co/from-bare-metal-to-serverless/
- https://www.fullstackpython.com/serverless.html