



Enter a new world of web development where everything is serverless

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

WAQ 18

Bastian Widmer - @dasrecht | @amazeeio

Bonjour WAQ18!

Bonjour WAQ18!

Pardon my Français - or the absence of it.

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



Overview

- What do you do?
- Past
- Your Goal
- Serverlessness
- Our way into the new world
- To the future and beyond!
- How to get started



\$> whoami bastian

- System Engineer at amaze.io
- Containers in Production 🧟🤖
- Zurich, Switzerland
- English, German, Swiss-German and a bit of French
- @dasrecht
- Too many side projects!*
- TEDxBern
- DevOpsDays Zurich
- CommunityRack.org
- Running TOR nodes for fun
- Working with real containers

* this list is not complete!



\$>

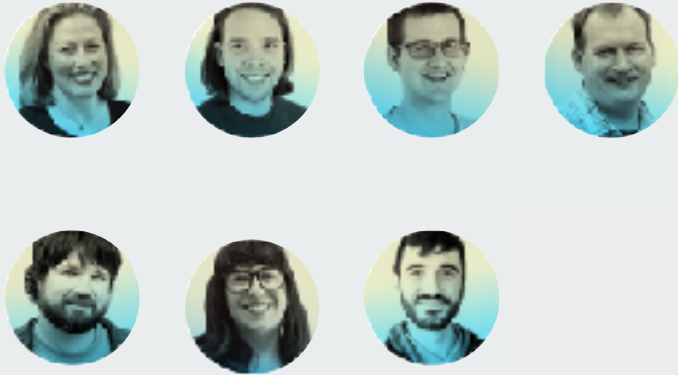




amazee.io?



amazee.io



- Fully Open Sourced Hosting Platform for Drupal Web Projects
- Hosting since 8 years
- We're a remote team of 7
 - Zurich, Switzerland
 - Barcelona, Spain
 - Austin, TX
 - Portland, OR
 - Melbourne
- Hosting in 16 different countries

Back in the day™



Windows Explorer window showing two folders: 'Program Files\Internet Explorer' and 'Program Files\Internet Explorer\resources'. The left folder contains files like help, installation, iellbak, iellbak.bak, iellbak.bak2, iellbak.bak3, iellbak.bak4, iellbak.bak5, iellbak.bak6, iellbak.bak7, iellbak.bak8, iellbak.bak9, iellbak.bak10, iellbak.bak11, iellbak.bak12, iellbak.bak13, iellbak.bak14, iellbak.bak15, iellbak.bak16, iellbak.bak17, iellbak.bak18, iellbak.bak19, iellbak.bak20, iellbak.bak21, iellbak.bak22, iellbak.bak23, iellbak.bak24, iellbak.bak25, iellbak.bak26, iellbak.bak27, iellbak.bak28, iellbak.bak29, iellbak.bak30, iellbak.bak31, iellbak.bak32, iellbak.bak33, iellbak.bak34, iellbak.bak35, iellbak.bak36, iellbak.bak37, iellbak.bak38, iellbak.bak39, iellbak.bak40, iellbak.bak41, iellbak.bak42, iellbak.bak43, iellbak.bak44, iellbak.bak45, iellbak.bak46, iellbak.bak47, iellbak.bak48, iellbak.bak49, iellbak.bak50, iellbak.bak51, iellbak.bak52, iellbak.bak53, iellbak.bak54, iellbak.bak55, iellbak.bak56, iellbak.bak57, iellbak.bak58, iellbak.bak59, iellbak.bak60, iellbak.bak61, iellbak.bak62, iellbak.bak63, iellbak.bak64, iellbak.bak65, iellbak.bak66, iellbak.bak67, iellbak.bak68, iellbak.bak69, iellbak.bak70, iellbak.bak71, iellbak.bak72, iellbak.bak73, iellbak.bak74, iellbak.bak75, iellbak.bak76, iellbak.bak77, iellbak.bak78, iellbak.bak79, iellbak.bak80, iellbak.bak81, iellbak.bak82, iellbak.bak83, iellbak.bak84, iellbak.bak85, iellbak.bak86, iellbak.bak87, iellbak.bak88, iellbak.bak89, iellbak.bak90, iellbak.bak91, iellbak.bak92, iellbak.bak93, iellbak.bak94, iellbak.bak95, iellbak.bak96, iellbak.bak97, iellbak.bak98, iellbak.bak99, iellbak.bak100. The right folder contains files like iellbak.bak101, iellbak.bak102, iellbak.bak103, iellbak.bak104, iellbak.bak105, iellbak.bak106, iellbak.bak107, iellbak.bak108, iellbak.bak109, iellbak.bak110, iellbak.bak111, iellbak.bak112, iellbak.bak113, iellbak.bak114, iellbak.bak115, iellbak.bak116, iellbak.bak117, iellbak.bak118, iellbak.bak119, iellbak.bak120, iellbak.bak121, iellbak.bak122, iellbak.bak123, iellbak.bak124, iellbak.bak125, iellbak.bak126, iellbak.bak127, iellbak.bak128, iellbak.bak129, iellbak.bak130, iellbak.bak131, iellbak.bak132, iellbak.bak133, iellbak.bak134, iellbak.bak135, iellbak.bak136, iellbak.bak137, iellbak.bak138, iellbak.bak139, iellbak.bak140, iellbak.bak141, iellbak.bak142, iellbak.bak143, iellbak.bak144, iellbak.bak145, iellbak.bak146, iellbak.bak147, iellbak.bak148, iellbak.bak149, iellbak.bak150, iellbak.bak151, iellbak.bak152, iellbak.bak153, iellbak.bak154, iellbak.bak155, iellbak.bak156, iellbak.bak157, iellbak.bak158, iellbak.bak159, iellbak.bak160, iellbak.bak161, iellbak.bak162, iellbak.bak163, iellbak.bak164, iellbak.bak165, iellbak.bak166, iellbak.bak167, iellbak.bak168, iellbak.bak169, iellbak.bak170, iellbak.bak171, iellbak.bak172, iellbak.bak173, iellbak.bak174, iellbak.bak175, iellbak.bak176, iellbak.bak177, iellbak.bak178, iellbak.bak179, iellbak.bak180, iellbak.bak181, iellbak.bak182, iellbak.bak183, iellbak.bak184, iellbak.bak185, iellbak.bak186, iellbak.bak187, iellbak.bak188, iellbak.bak189, iellbak.bak190, iellbak.bak191, iellbak.bak192, iellbak.bak193, iellbak.bak194, iellbak.bak195, iellbak.bak196, iellbak.bak197, iellbak.bak198, iellbak.bak199, iellbak.bak200.

Microsoft
Internet Explorer 5

! This product is...
Copyright 1999 Microsoft Corporation. All rights reserved.
This program is protected by United States and other
copyright laws.

What is your goal?

Write code

Deploy code

~~Manage Infrastructure~~

Enjoy that things just work!

Learn something along the way.

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

— one of my colleagues, 2012



2015

- The bleeding edge gets dull after a while
- 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 and updating is a pain!
- Let's take look at Docker
- 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
 - Working towards V1.0.0 Release

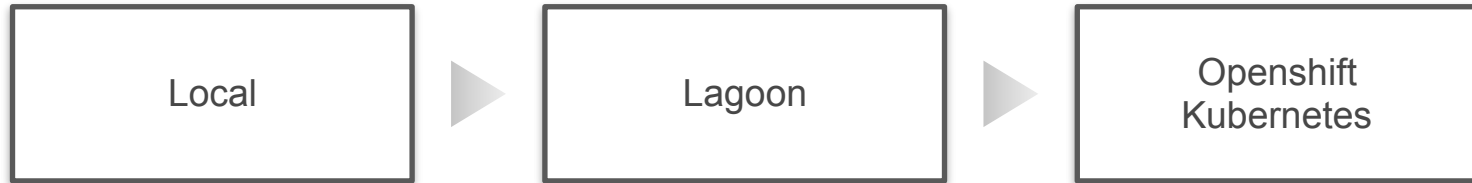


Lagoon?

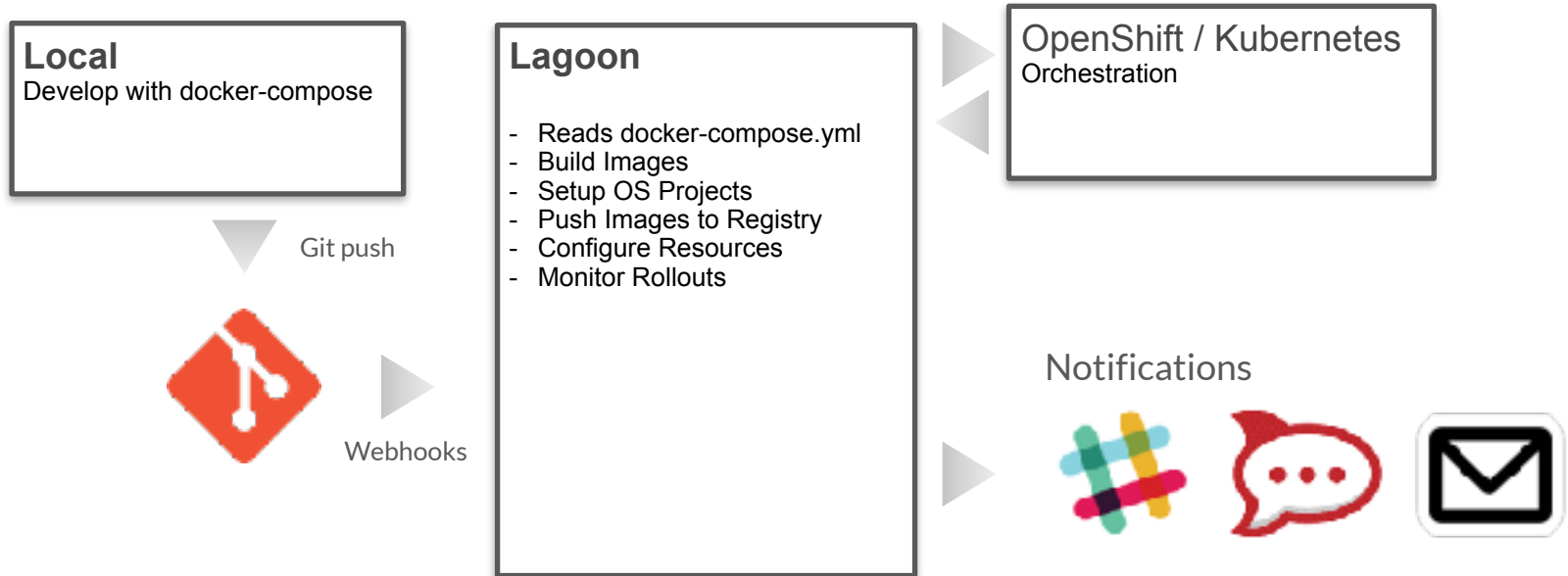
- 4. Iteration of our Hosting Stack
- Microservices
- Deployment Pipeline for Drupal Web Projects
- Local Development Environment
- Infrastructure/Services Defined in Code



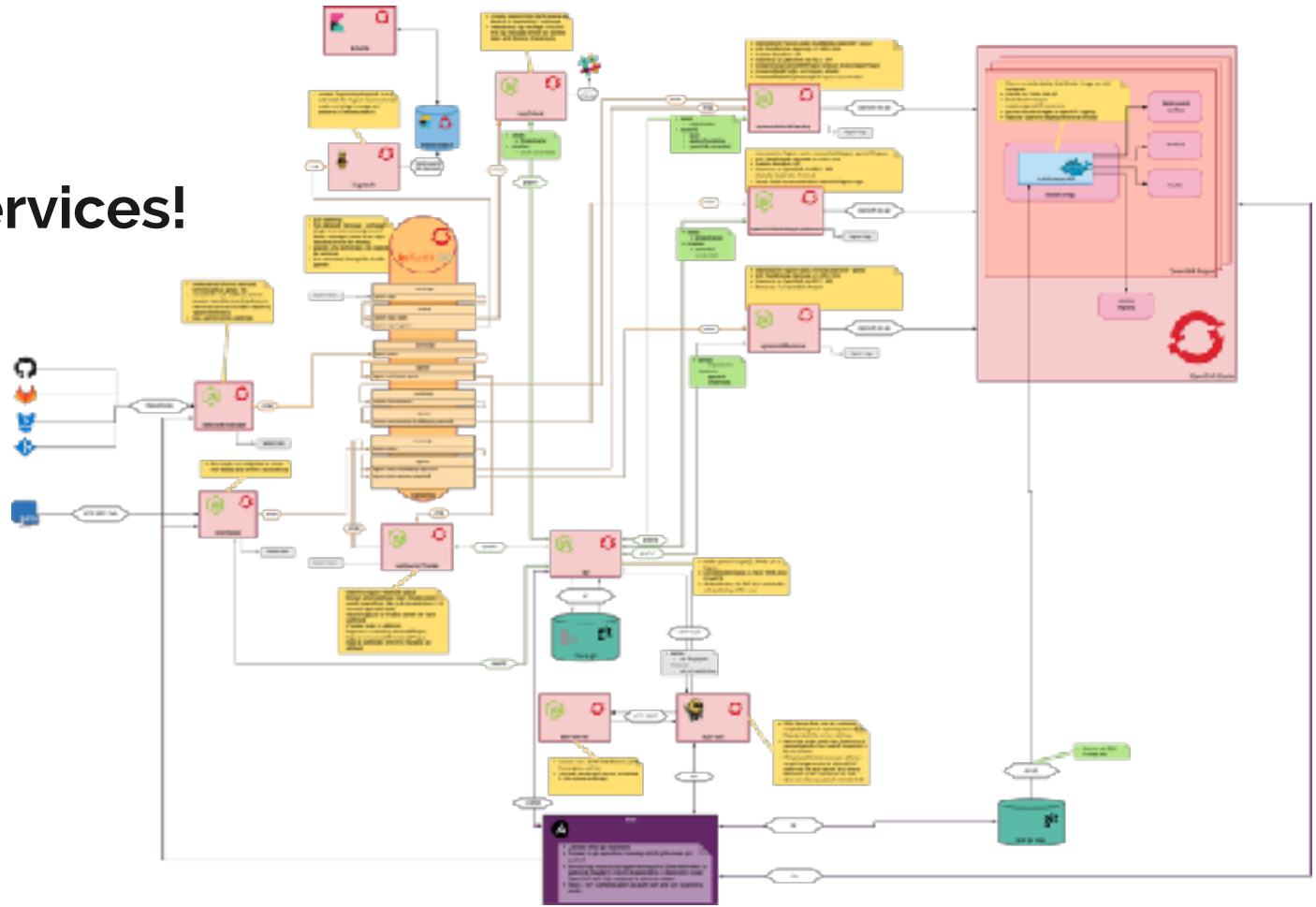
TL;DR



More details



Microservices!



And yes! It's all
opensource!

The screenshot shows the GitHub repository page for 'amazon/lagoon'. A red box highlights the repository name in the header. Below the header, statistics are displayed: 94,903 commits, 812 branches, 0.6 releases, 44,28 contributors, and 4 forks. A table lists various sub-repositories with their commit counts and last commit dates.

Repository	Commits	Last commit
amazon-lagoon	94,903	2 days ago
amazon-lagoon-frontend	1,100	2 days ago
amazon-lagoon-backend	1,100	2 days ago
amazon-lagoon-docs	1,100	2 days ago
amazon-lagoon-frontend-test	1,100	2 days ago
amazon-lagoon-backend-test	1,100	2 days ago
amazon-lagoon-docs-test	1,100	2 days ago
amazon-lagoon-frontend-ci	1,100	2 days ago
amazon-lagoon-backend-ci	1,100	2 days ago
amazon-lagoon-docs-ci	1,100	2 days ago
amazon-lagoon-frontend-dev	1,100	2 days ago
amazon-lagoon-backend-dev	1,100	2 days ago
amazon-lagoon-docs-dev	1,100	2 days ago

COMPLEXITY



Complexity of orchestrated services

- If you don't master traditional infrastructure - You will have a hard time
- Orchestrated micro-services 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)

Serverless?

Spoiler alert: Serverless still involves servers!



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)



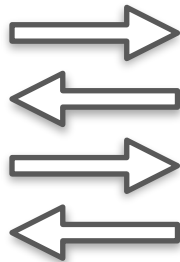
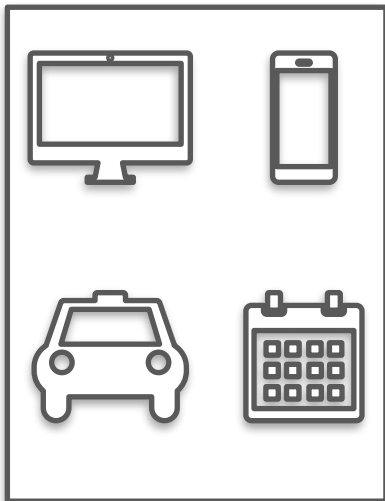
Functions as a Service (FaaS)

- Deployscript triggers a function saving the logs to an Object Storage,
- Someone calling our Emergency phone - Triggers a function and forwards the Call



Functions as a Service (FaaS)

Event Sources



Functions



Backend Services





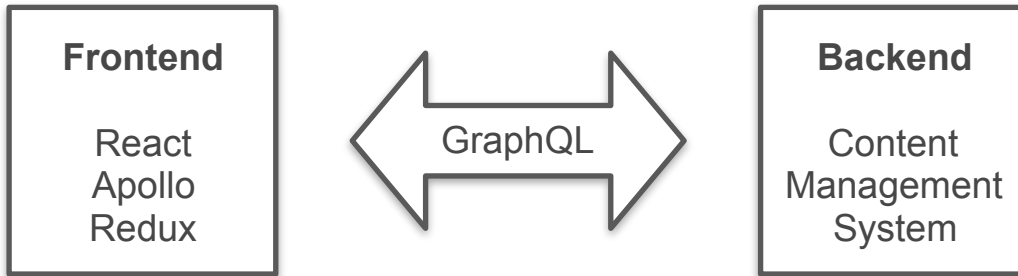
Serverless

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 memory or cpu usage
- Embrace third-party services
- Loosely coupled

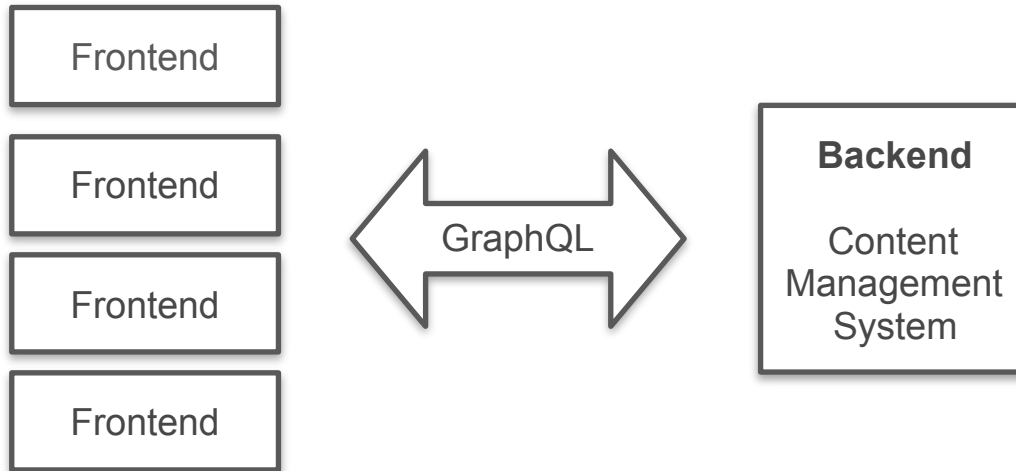


Loose Coupling / Decoupling

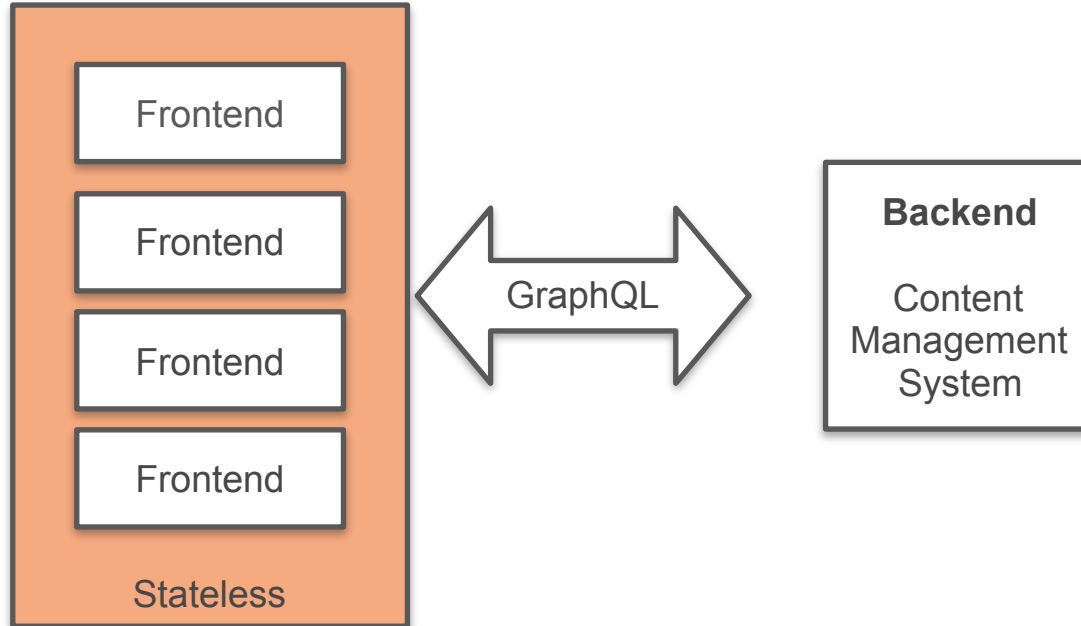




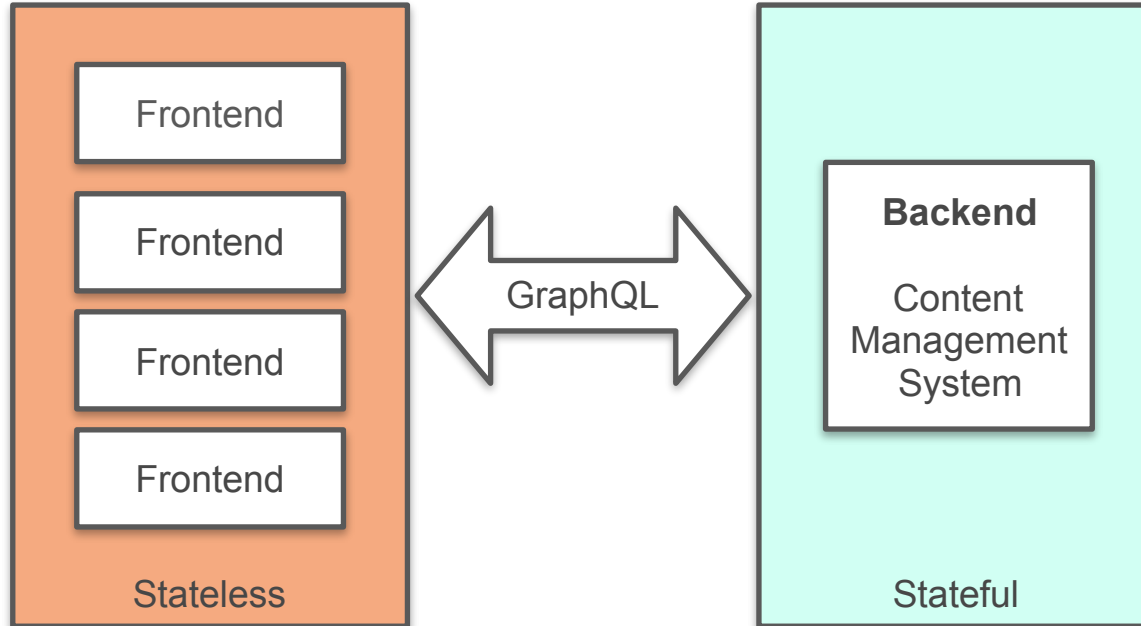
Loose Coupling / Decoupling



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?

Cloud Native - Adopting the new mindset



Cloud Native - Built for leveraging Cloud Services

- Need Block Storage - API CALL: I need 500GB SSD storage with 5000 IOPS
- Saving Objects - API CALL: store my files
- Too much load - API CALL: I need 2 Compute nodes with 96GB of RAM and 16 CPUs
- Maintenance? - Create a new compute node, schedule the containers/functions there, toss away the old machine (cattle vs pets)

To sum it up: How do you get started?



How do you get started?

- Automate, Experiment, Learn
- Embrace third-party services
- 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
- You will fail and learn - that's normal
- Build on opensource and if you can, also opensource

Thank you for your attention!

Bastian Widmer - @dasrecht | @amazeeio





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/>