Serverless Computing

DC Posch
dcposch@dcpos.ch
@dcposch

1. a bit of history...
2. what’s serverless?
3. serverless backends
4. future

A bit of history...

A bit of history

A bit of history

A bit of history

“Pets vs cattle”

```
pets   = servers with names
cattle = servers with numbers
```

A bit of history

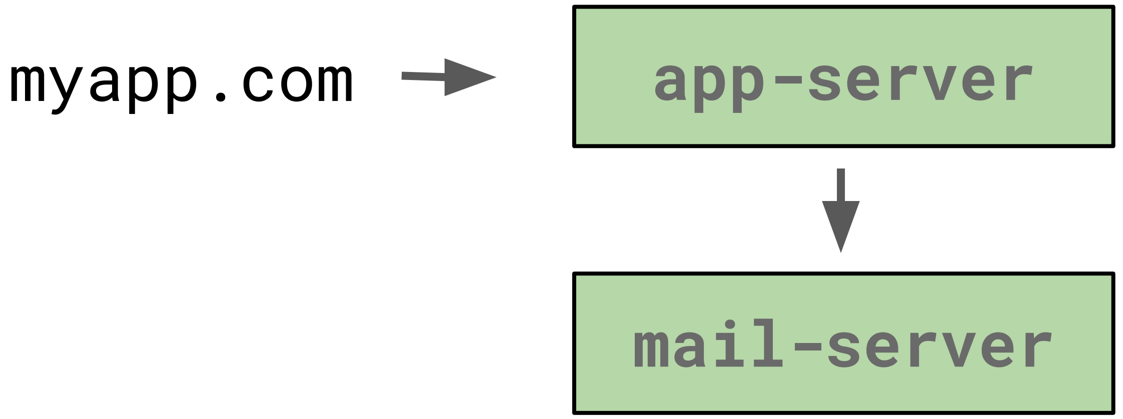
“Pets”

```
Then: host.ucla.edu -> physical server
Now:  host.ucla.edu -> some ec2 instance
```

Either way, if one server dies, the site stops working :(

A bit of history

“Pets”



A bit of history

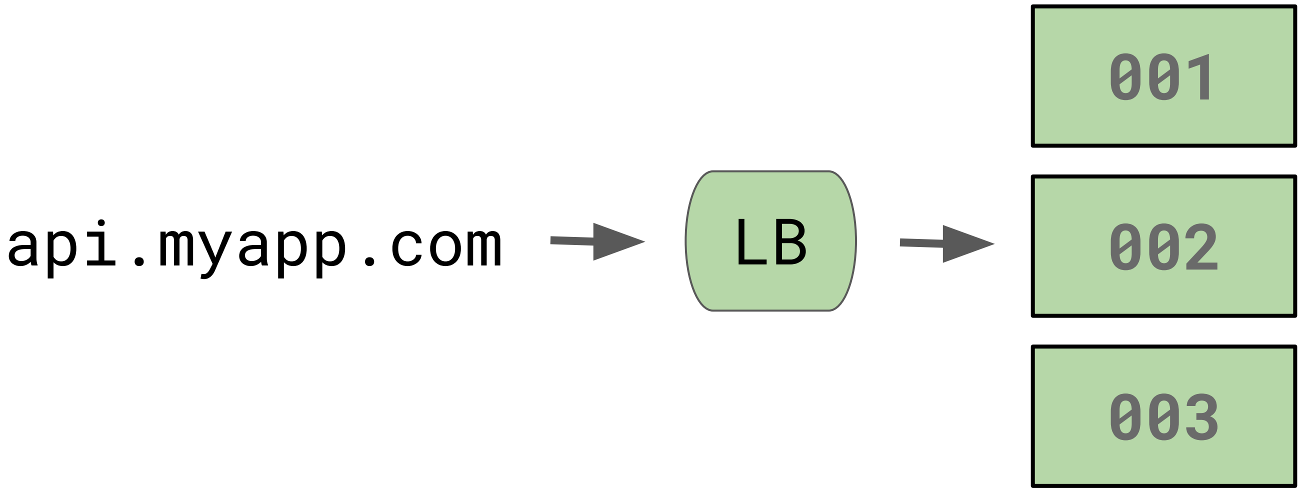
“Cattle”

```
host.ucla.edu -> load balancer
              -> 2+ identical instances
```

One dies, no big deal.

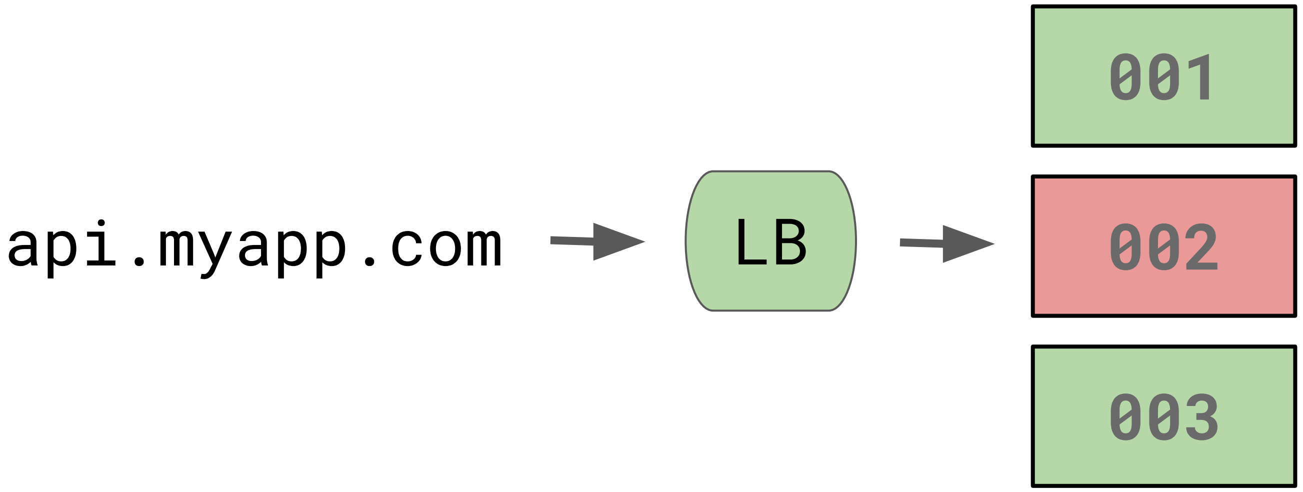
A bit of history

“Cattle”



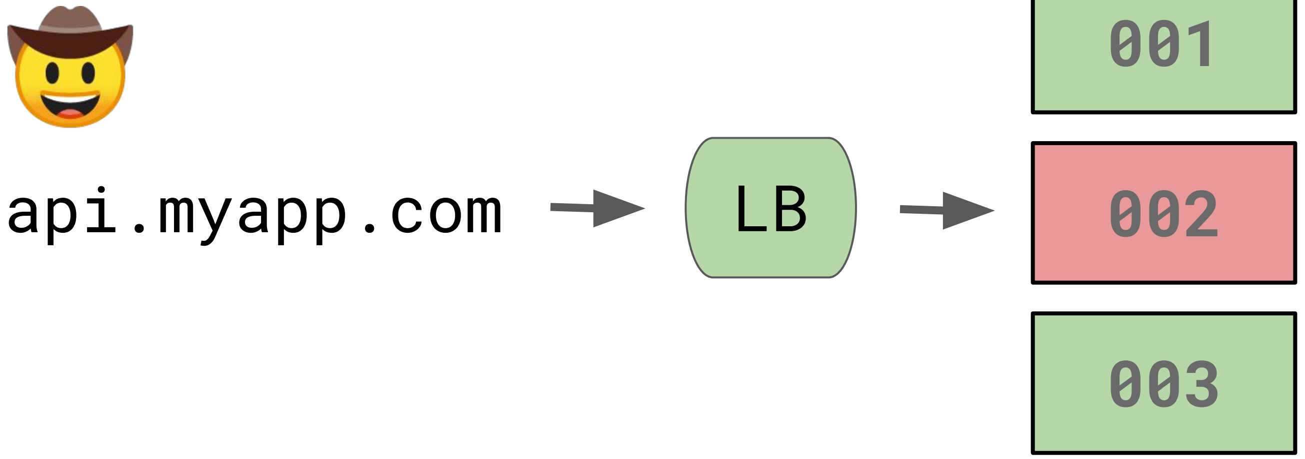
A bit of history

“Cattle”



A bit of history

“Cattle”



A bit of history

servers with names

» servers with numbers

» ?

A bit of history

servers with names

» servers with numbers

» **the new hotness... not caring how many servers are running at all.**

1. a bit of history...
2. **what’s serverless?**
3. serverless backends
4. edge computing

What’s serverless?

What’s serverless?

for example: you just define an HTTP handler, then tell Amazon to handle the incoming traffic.

What’s serverless?

AWS Lambda

First and most popular. Upload your code, define when to run it.

Demo time

What’s serverless?

ec2: pay per instance, per hour

lambda: pay per invocation

What’s serverless?

“Declarative computing”

Pay only what you use

Scale down to zero - no work, no cost

Automatically scale up

Why serverless?

Why serverless?

Scale up automatically

If app gets popular, load balancing,

running more instances -> automatic.

Why serverless?

Scale down to zero

Good for the long tail. You can run a small app for ~$0/mo.

Why serverless?

Less work

No starting, stopping, SSH-ing into, patching, upgrading servers. No estimating how many instance you need. Etc.

Demo time

1. a bit of history...
2. what’s serverless?
3. **serverless backends**
4. edge computing

Serverless backends

Serverless backends

Functions (Lambda, etc) are **stateless**.

No disk. Can’t store anything between invocations.

Serverless backends

Functions (Lambda, etc) are **stateless**.

✓ Great for scaling, reliability

✘ Can’t store any data

Serverless backends

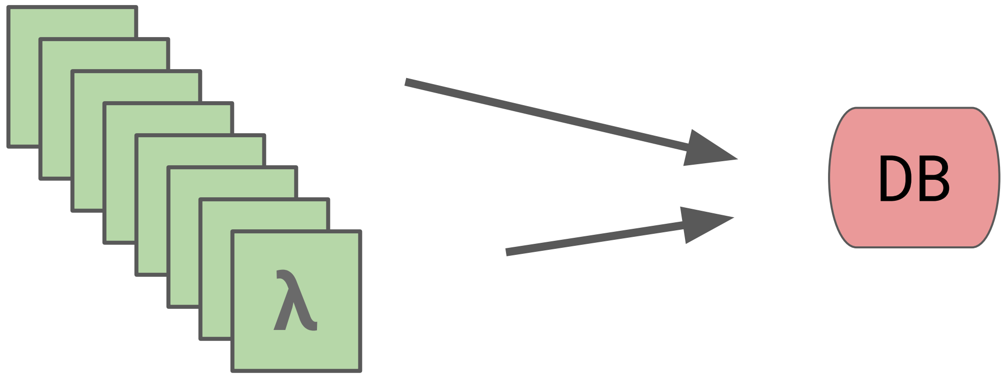
Need a place to keep state.

- Traditional databases

Serverless backends

Need a place to keep state.

- Traditional databases :/



Serverless backends

Need a place to keep state.

- Traditional databases
- Object stores, like S3

Serverless backends

Need a place to keep state.

- Traditional databases
- Object stores, like S3
- Serverless DBs, like Dynamo

Serverless backends

Need a place to keep state.

- Traditional databases
- Object stores, like S3
- Serverless DBs, like Dynamo
- Serverless backends, like Firebase

Demo time

1. a bit of history...
2. what’s serverless?
3. serverless backends
4. **future**

Future

Future

Open source / open standards

Today, serverless is a fairly new technology. Lambda, GCF, S3, Dynamo, Firebase are all closed source :(

Future

Open source / open standards

Knative looks promising.

Future

Open source / open standards

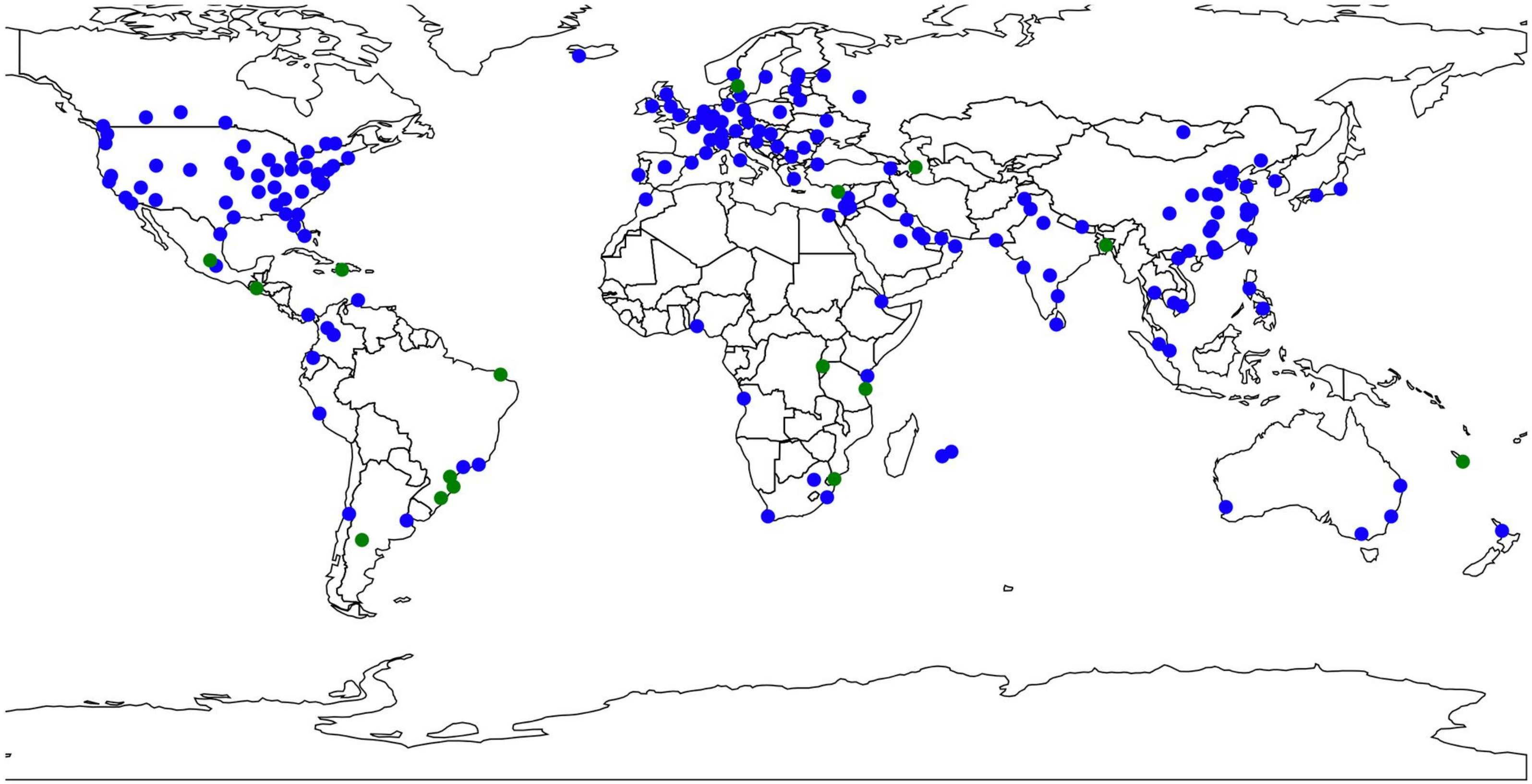
Using today’s proprietary serverless is fine for many projects.

Future

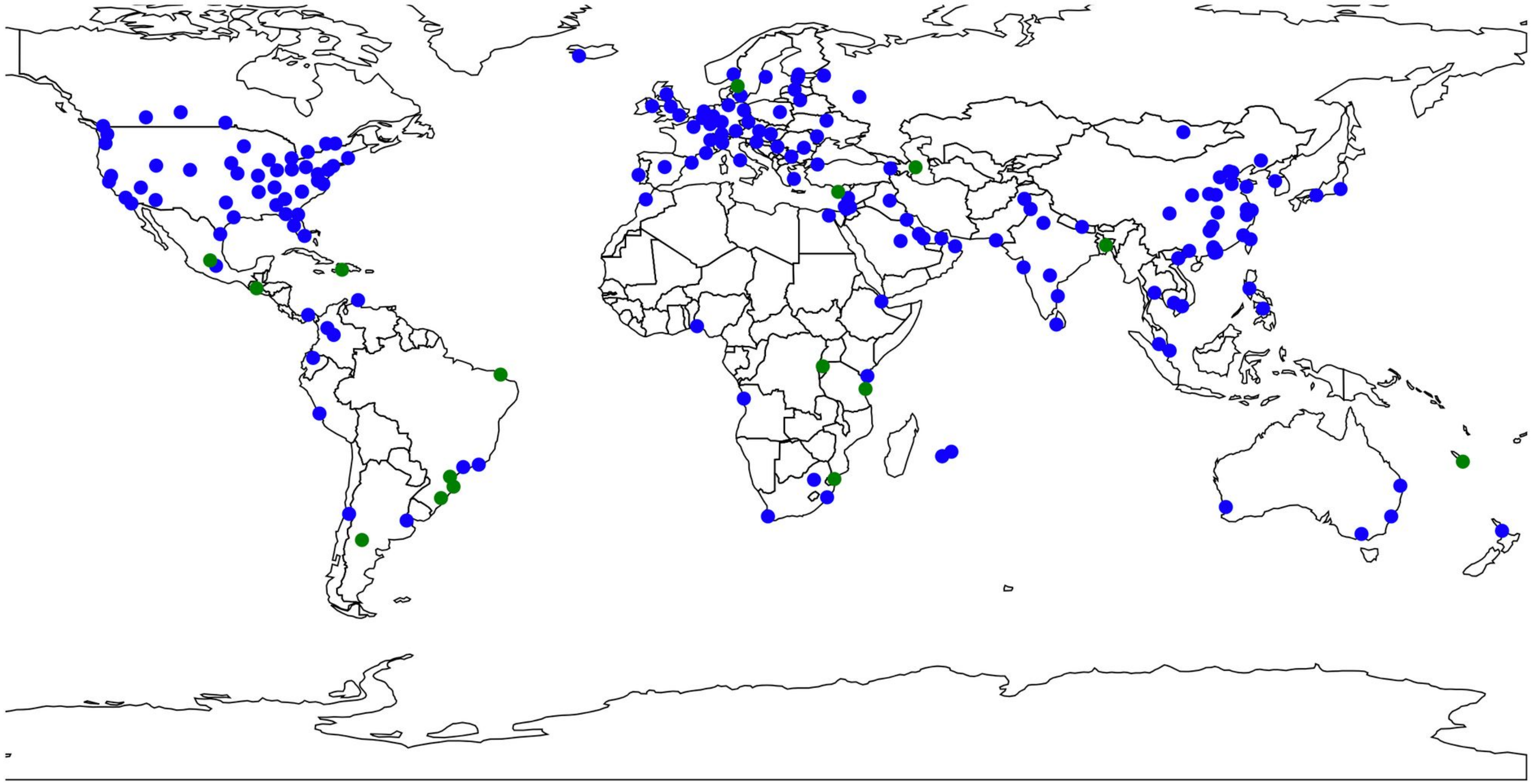
Edge computing.

For example, Cloudflare Workers.

Future - edge computing



Future - most of world in <10ms



Conclusion

Questions?

DC Posch
dcposch@dcpos.ch
@dcposch