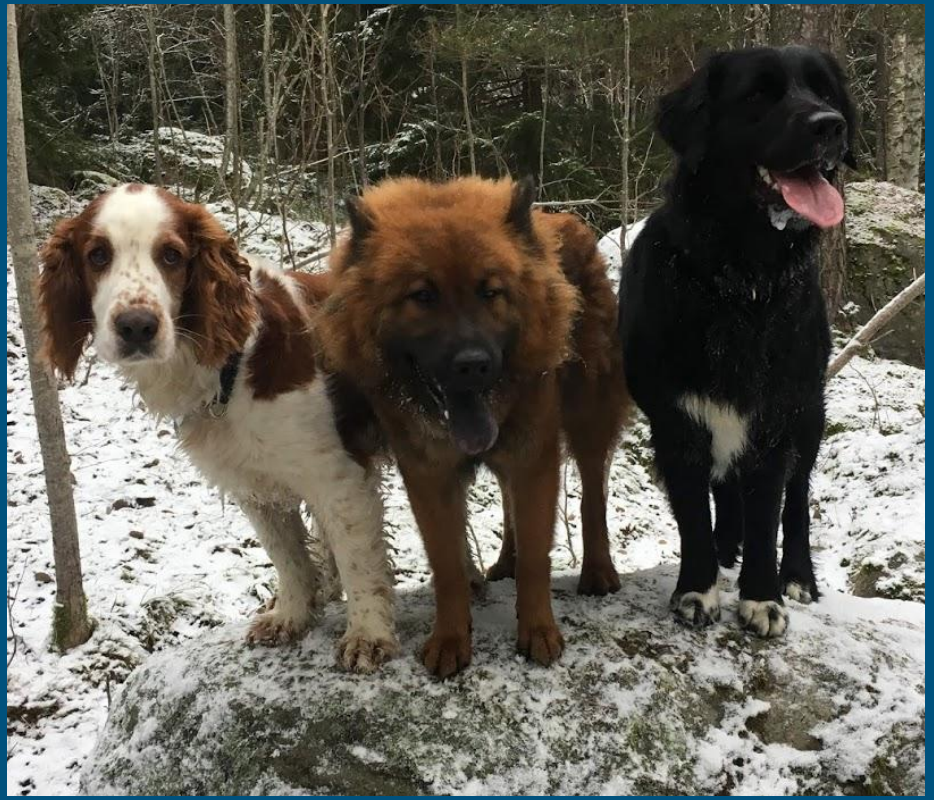
How a team of 4 wrote, maintained and operated 50+ services

If you expect cool technical solutions, prepare to be disappointed

2-4 developers

An average of more than 1 service per week

Dropwizard java mongodb



Why so many services?

We needed to scale! In the future!

Many teams and millions of requests per second!

Why so many services?

~~We needed to scale! In the future!~~

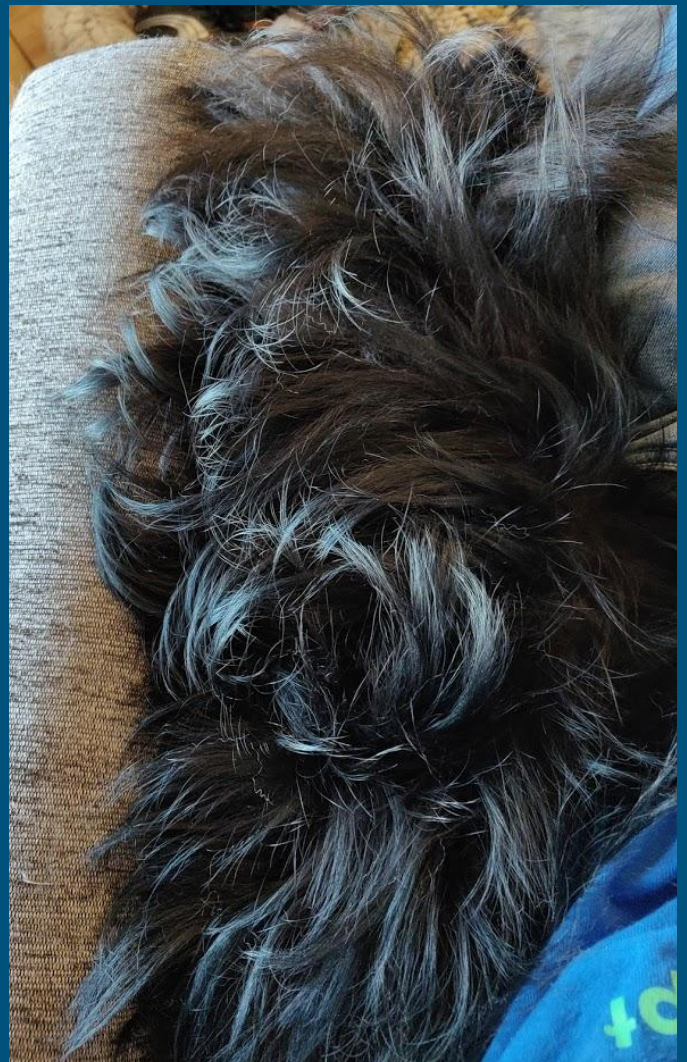
~~Many teams and millions of requests per second!~~

We wanted boundaries between domains.

Why so many services?

We rewrote entire services several times.

Not because they where wrong, but the world changed

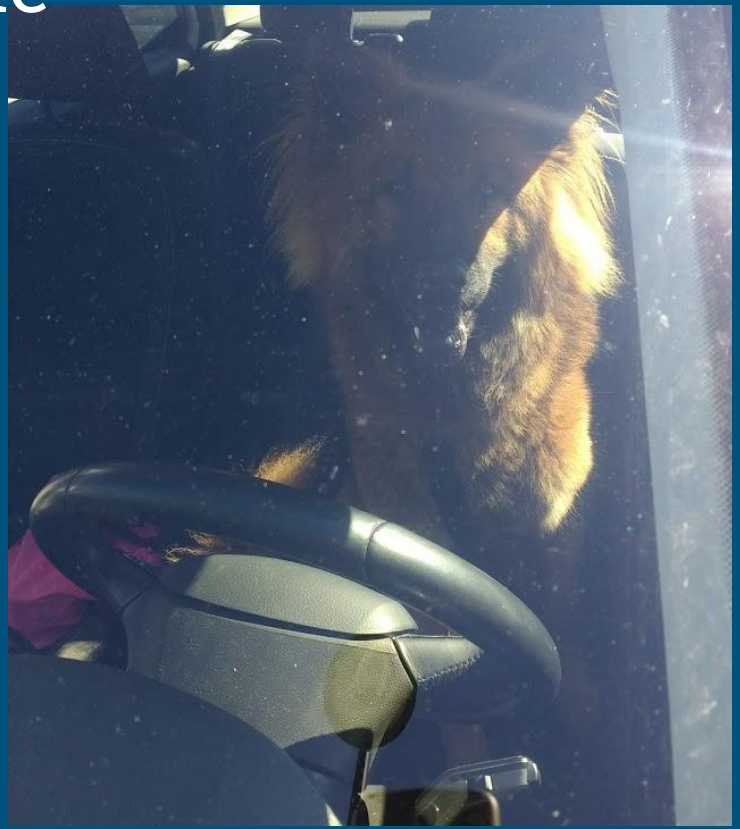


As little control as possible

Hosted mongodb

Sns/sqs/kinesis

Kops to create a kubernetes cluster



Create a new service

Creating a new service in 3:ish steps

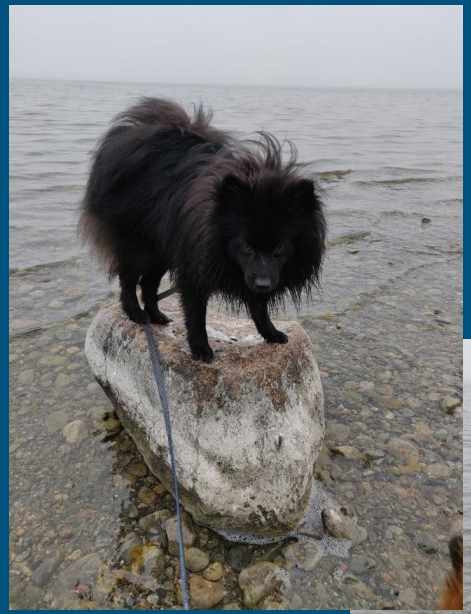
Create a new service

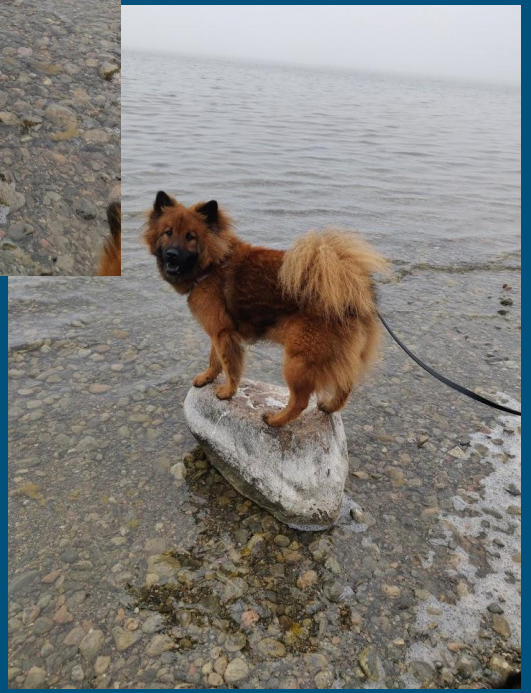
- Select old service
- Press ctrl+c
- Press ctrl+v

Create a new service

Search and replace old name with new name

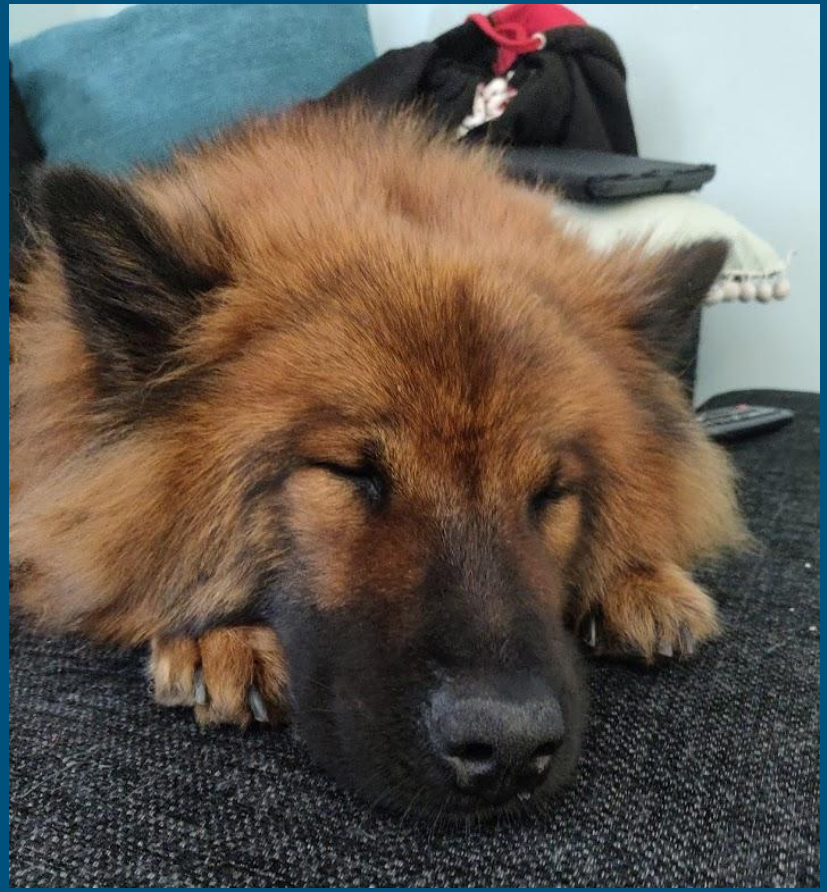
Replace business logic





Create a new service

Extremly booring, but not error prone



Create a new service

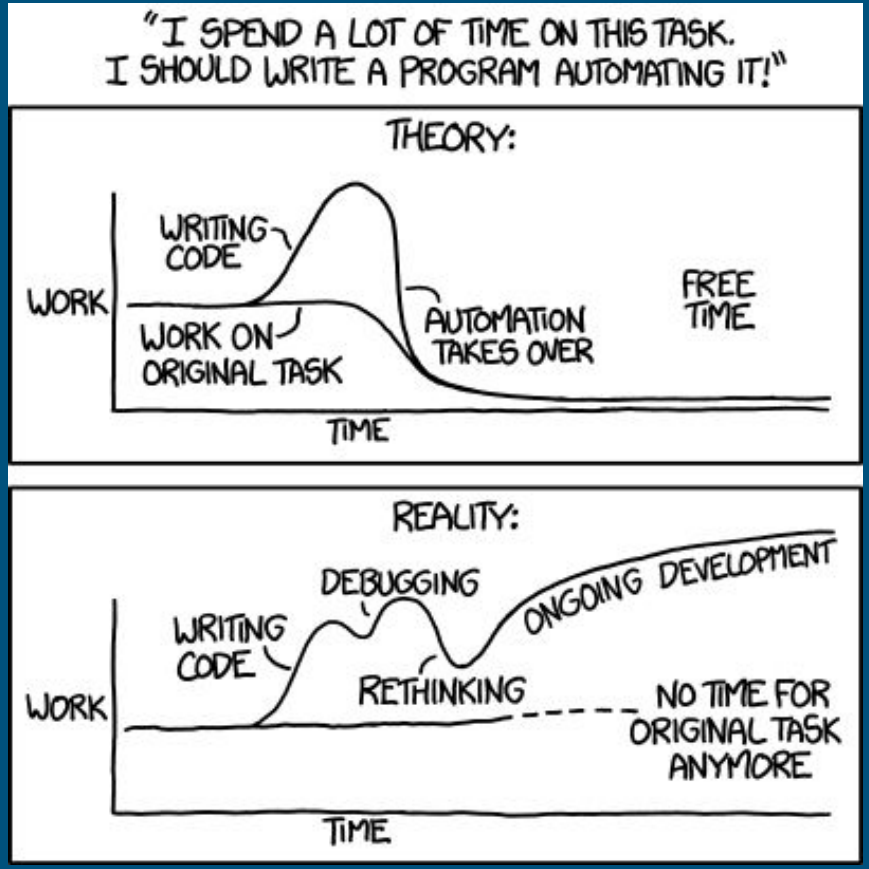
Yes we could have a mvn archetype and a pipeline and a…

Don’t automate boring stuff that is easy to get right.

Dont automate because you can

- No deployment pipelines (until we felt we needed them)
- No Cross service integration tests or full blown “staging” tests

https://xkcd.com/1319/



Shared things

Small common library

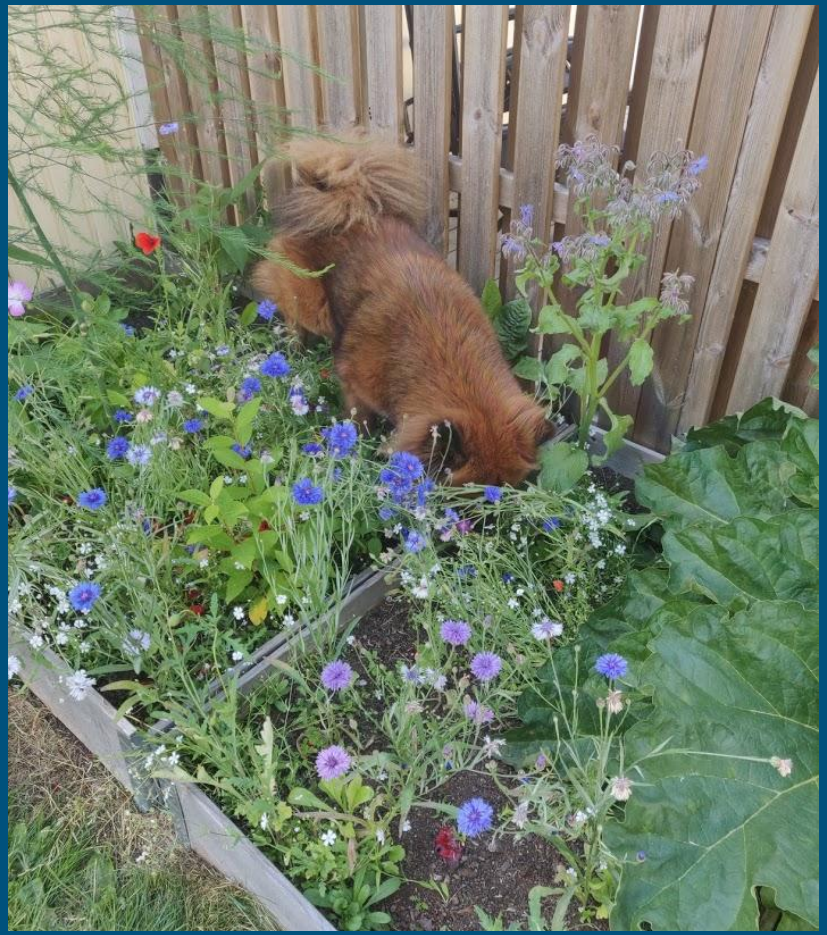
Proxy for verifying authentication etc

Some services, users, groups, comodity booring things

Everything is familiar

Everything looks the same

Experiments are great! But update old services



Everything is familiar

Migrated everything from java to kotlin

Changed from sns/sqs to kinesis and then back again.

Not worth using something if it's not worth updating everything.

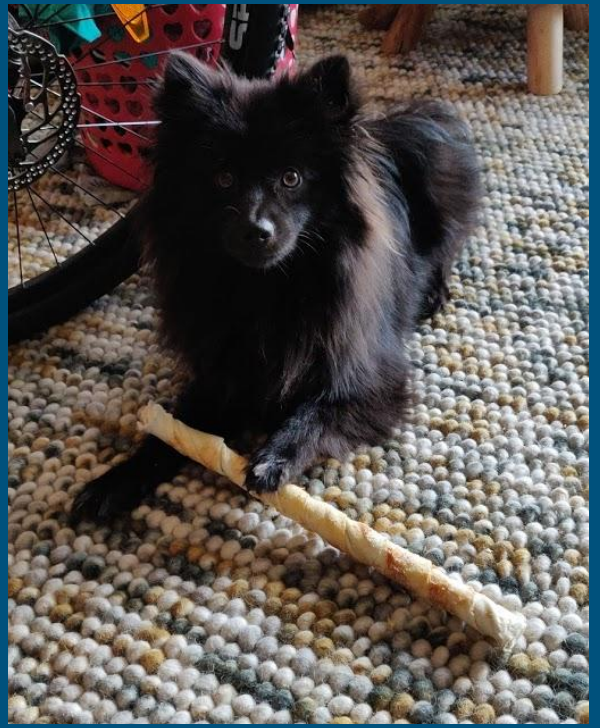
No meetings

(except standup and retro)

Full days of coding, day after day

Small isolated system

Making a isolated one team system compared to a multiteam huge fancy thing is like planning a TV dinner compared to a wedding.



No fancy solutions
No snowflakes
Great prestigeless team