zalando

Auto-scaling your API

Insights and Tips from the Zalando Team

Sean Patrick Floyd - @oldJavaGuy

Luis Mineiro - @voidmaze

JBCNConf 2015
Barcelona, Spain

ONE of EUROPE'S LARGEST ONLINE FASHION RETAILERS

15 countries
3 fulfilment centers
15+ million active customers
2.2+ billion € revenue 2014
130+ million visits per month
8.000+ employees

Tech hubs in Berlin, Dublin, Dortmund and Helsinki

Visit us: tech.zalando.com

Our Scale

2 datacenters
thousands of production instances
serving **15** countries

Based on https://flic.kr/p/bX5E4c

Zalando stack

Credits to our colleague Kolja Wilcke

zalando

Credits to our colleague Kolja Wilcke

ENVÍO Y DEVOLUCIÓN GRATUITOS | DERECHO DE DEVOLUCIÓN DE 100 DÍAS | ATENCIÓN AL CLIENTE GRATUITA 900 802 463

MUJER HOMBRE NIÑO

zalando

Iniciar sesión | Favoritos | Cesta

News & Style | Ropa | Zapatos | Deporte | Bolsos y Complementos | Premium | Marcas | Rebajas

Busca aquí: zapatos, ropa, marcas...

ELIGE UNA SECCIÓN

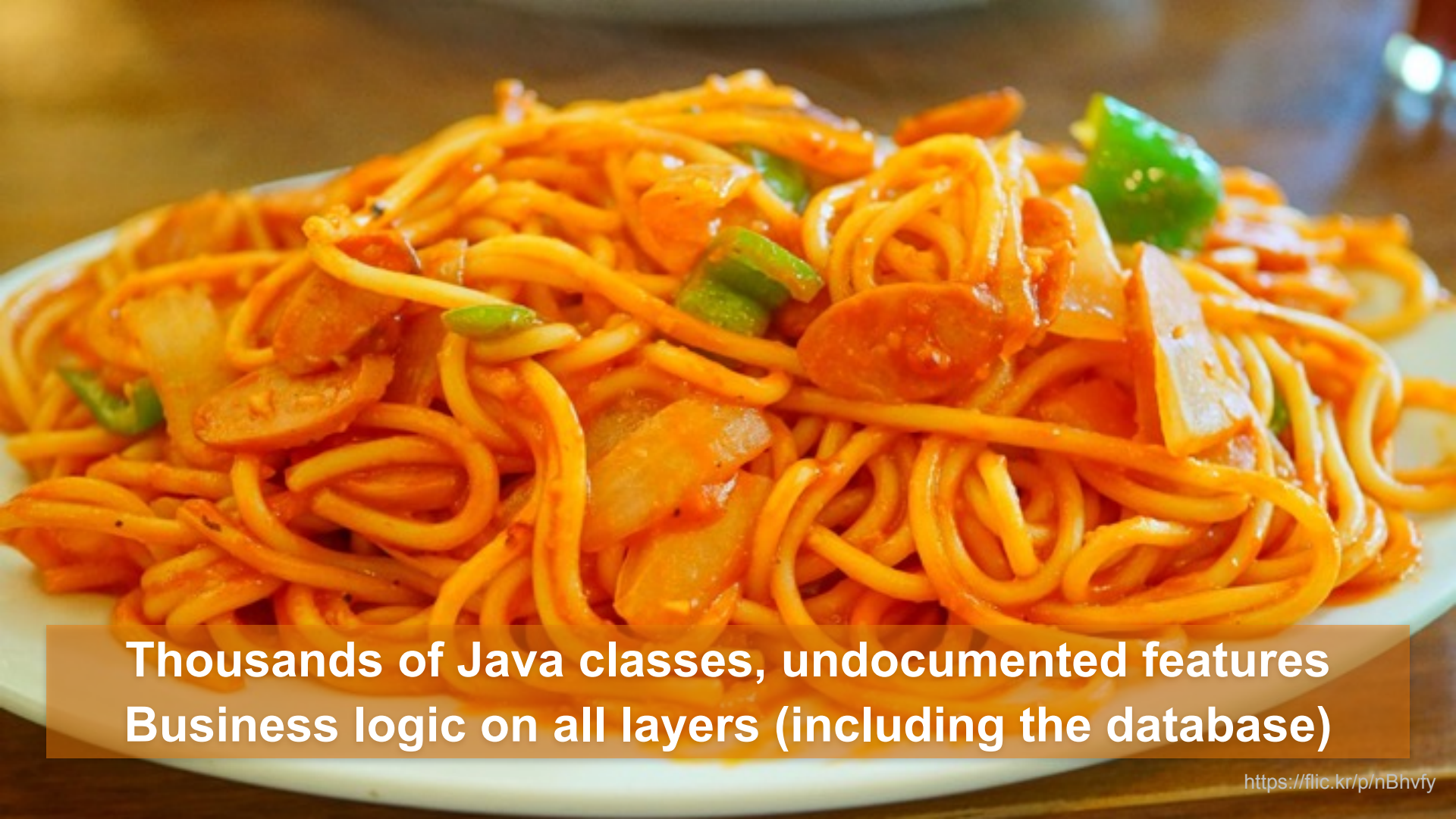
zalando

Credits to our colleague Kolja Wilcke

The Shop Monolith

We call it “**Jimmy**”

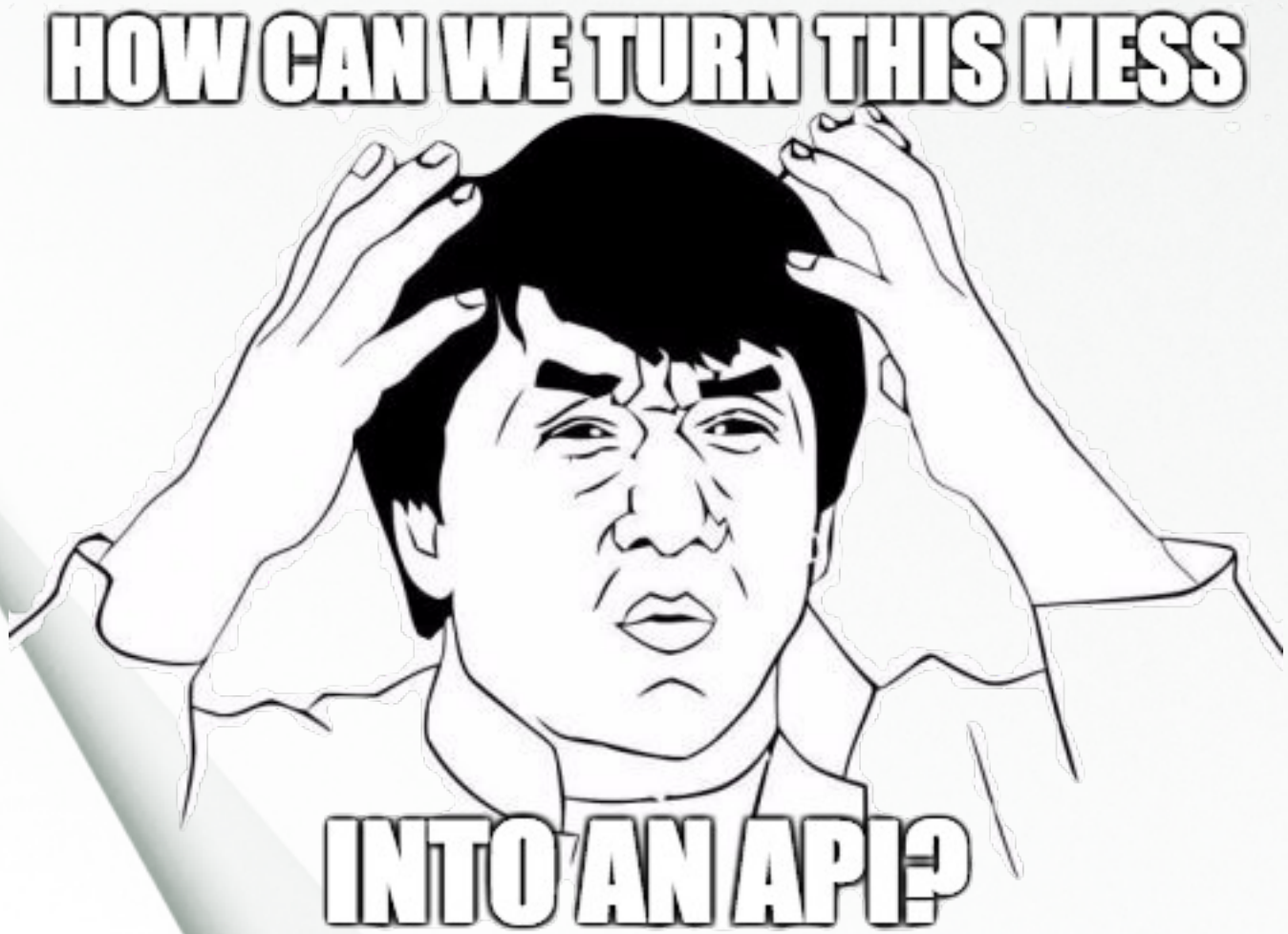
http://blog.codinghorror.com/new-programming-jargon/


Thousands of Java classes, undocumented features
Business logic on all layers (including the database)
https://flic.kr/p/nBhvfy

It’s 2013…

Zalando wants to sponsor hackathons…

We need a REST API!


HOW CAN WE TURN THIS MESS
INTO AN API?

Hackathon API

zalando

First Step

Let’s create some REST-(ish) Spring controllers inside our **monolithic web application**

Deploy a couple of Jimmy instances just to serve requests for this “API”

Pros

The infrastructure is already there!

A few days of coding and we’re all set

Cons

This “API” cannot be deployed independently of Jimmy.

Jimmy is infested with business logic everywhere.

New Requirements


WE HAVE A HACKATHON API?
LET'S USE IT FOR SOME OTHER STUFF AS WELL
imgflip.com

New Requirements

We needed this new API for our existing frontends, also (very high traffic).

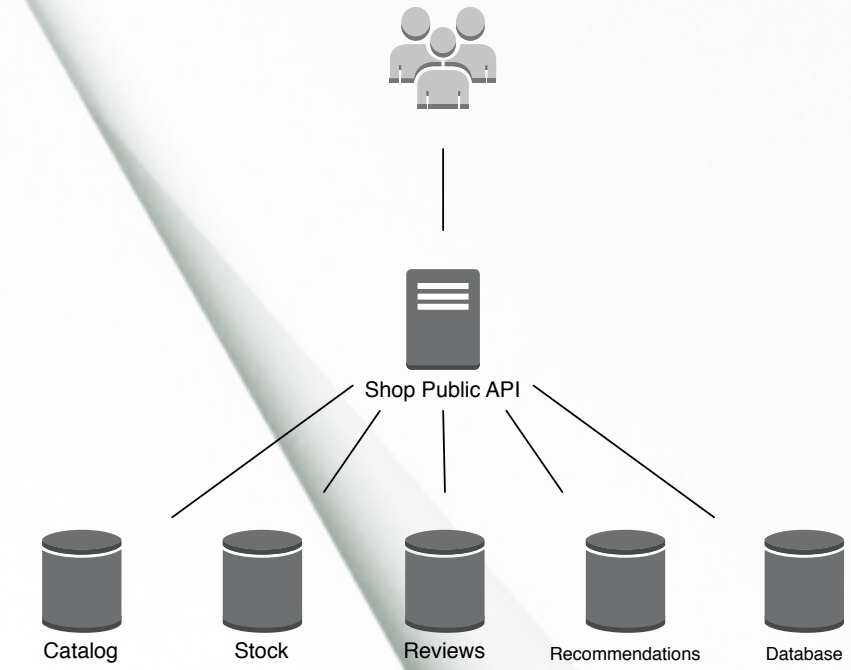
Some third-party apps also wanted to use this API as their backend.

Shop Public API

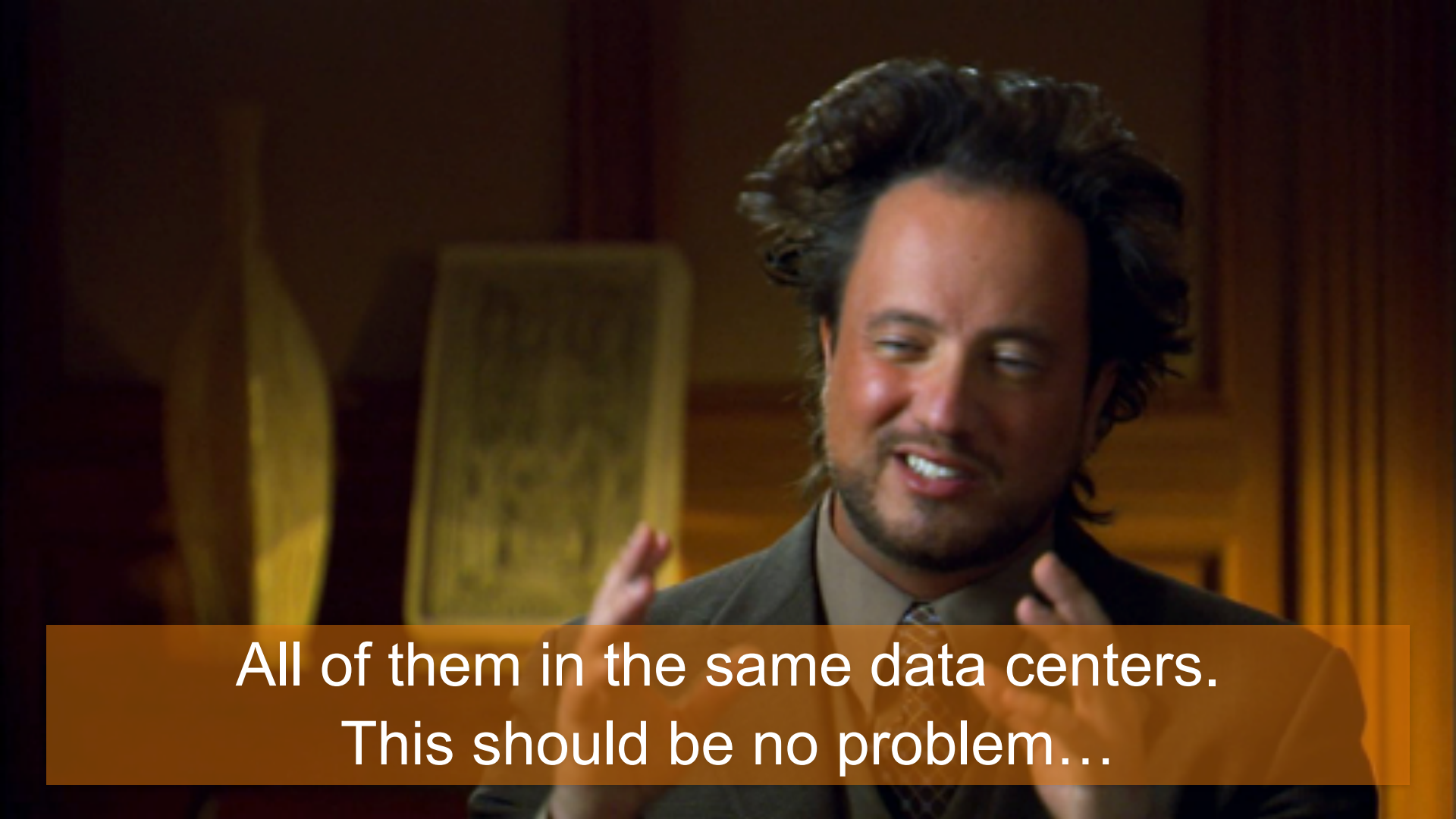
We decided to build a “real” API as a separate standalone project.

Of course, there were some dependencies…

Data Sources



1. Catalog - **SOLr**
2. Stock - **Memcached**
3. Reviews - **REST**(-ish) API
4. Recommendations - **RPC**
5. Database - **PostgreSQL**


All of them in the same data centers.
This should be no problem…

Tech Company

Paradigm Shift

Until 2014
We’re a Fashion Company that has a lot of tech knowledge

2015
Suddenly we’re a Tech Company, providing “Fashion as a Service”

We established 5 principles

API FIRST

API First

Document and peer review your API in a format like Swagger before writing a single line of code.

Ideally, either generate either your server interfaces or your test data (or both) from the Swagger spec

REST

REST

Manipulate resources, don't call methods

Expose nouns, not verbs

Use HTTP verbs
GET, PUT, POST, DELETE, PATCH

SAAS

Identity Management

Our backend services don’t expose APIs yet

Company-wide IAM strategy is not ready yet

Can only expose read-only features for now

MICRO-SERVICES

Microservices

We already have a Service-Oriented Architecture

It was mostly SOAP, though…

And definitely not micro

CLOUD

Road to AWS

Some challenges ahead…

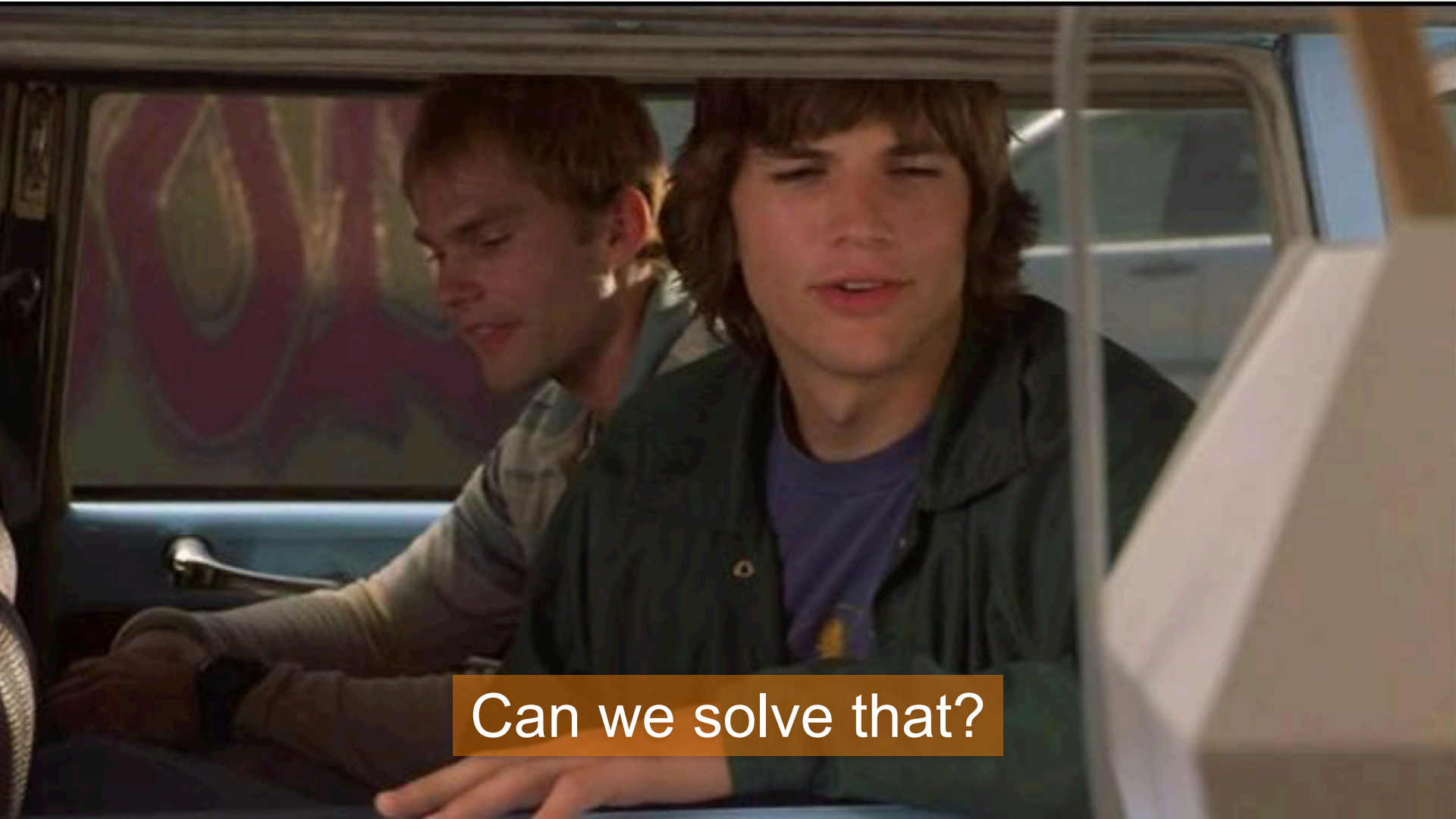
1. Backend services not available yet
2. SOLr also not available
3. We can’t just move the databases there

How we did it

Challenges

Catalog and Stock datasources are latency critical

and they’re owned by different teams!


Can we solve that?

Step 1 - move critical datasources to AWS

“Move” Data Sources to AWS

We **can’t** just **move** them!

Jimmy also needs them in the data centers,

you insensitive clod!

Ok… we just **replicate** them.


STEAMED RICE
SOUPS
WONTON SOUP
GG FLOWER SOUP
HOT AND SOUR SO
EEF NOODLE SOUP
4.55
UR PORK
CHINESE
FOOOOOD
and then…?

Replicating Data Sources

SOLr has its own replication mechanism

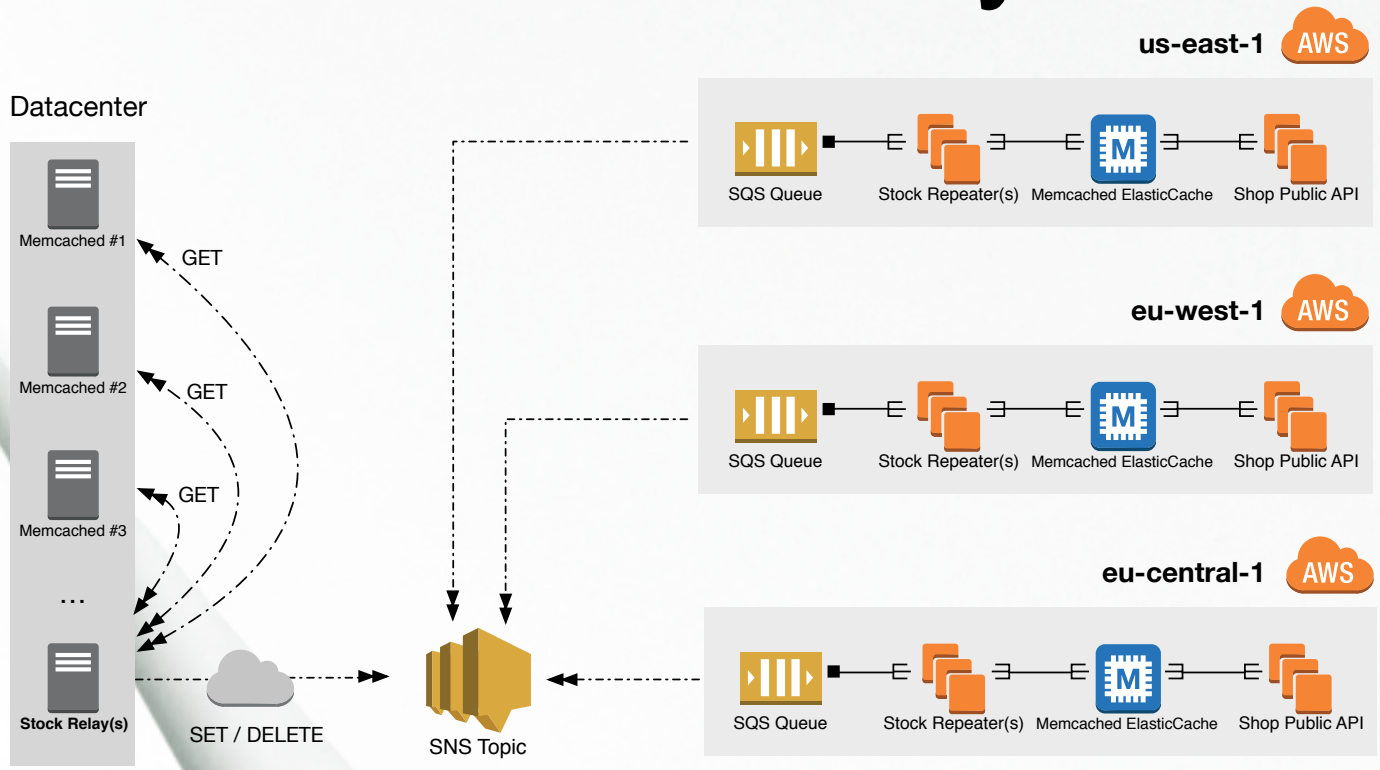
over HTTP

memcached should be easy …


You wish!

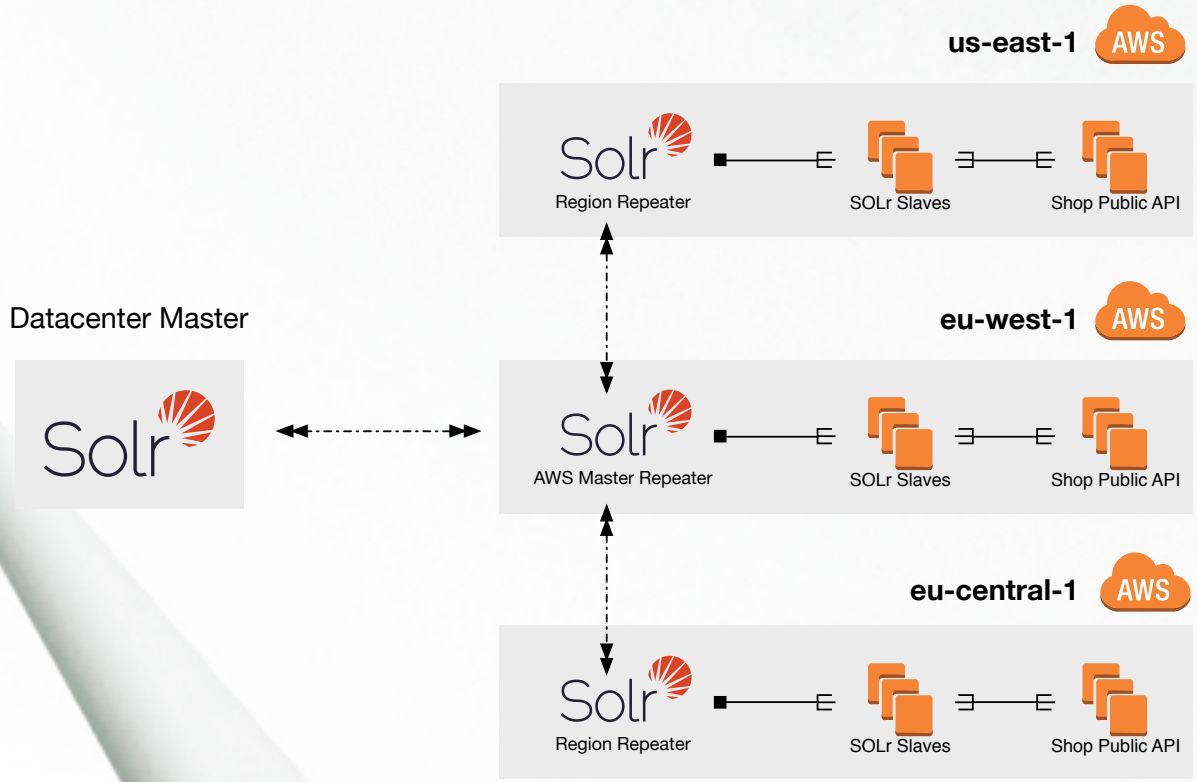
Step 2 - Build memcached replication

Stock Relay

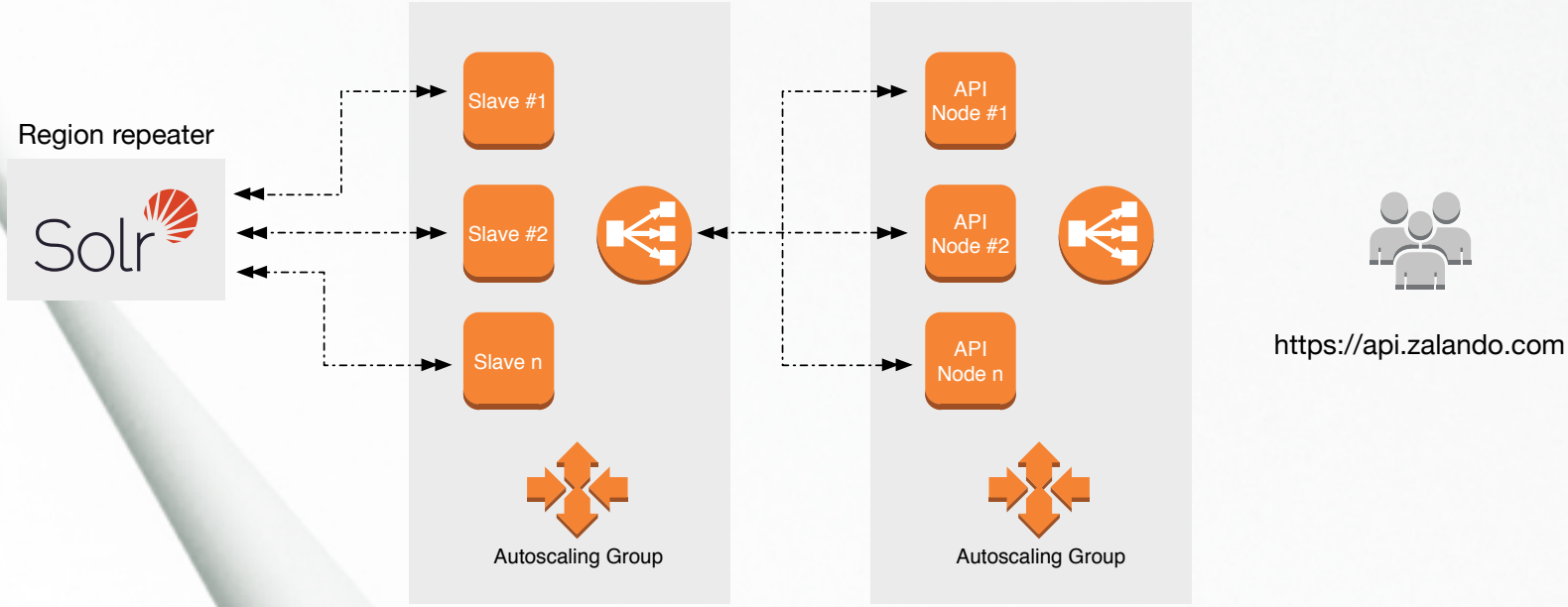

us-east-1
AWS
Datacenter
Memcached #1
Memcached #2
Memcached #3
...
Stock Relay(s)
GET
GET
GET
SET / DELETE
SNS Topic
SQS Queue
Stock Repeater(s)
Memcached ElasticCache
Shop Public API
eu-west-1
AWS
SQS Queue
Stock Repeater(s)
Memcached ElasticCache
Shop Public API
eu-central-1
AWS
SQS Queue
Stock Repeater(s)
Memcached ElasticCache
Shop Public API


STEAMED RICE
SOUPS
WONTON SOUP
GG FLOWER SOUP
HOT AND SOUR SO
EEF NOODLE SOUP
4.55
UR PORK
CHINESE
FOOOOOD
and then…?

AWS SOLr Repeater

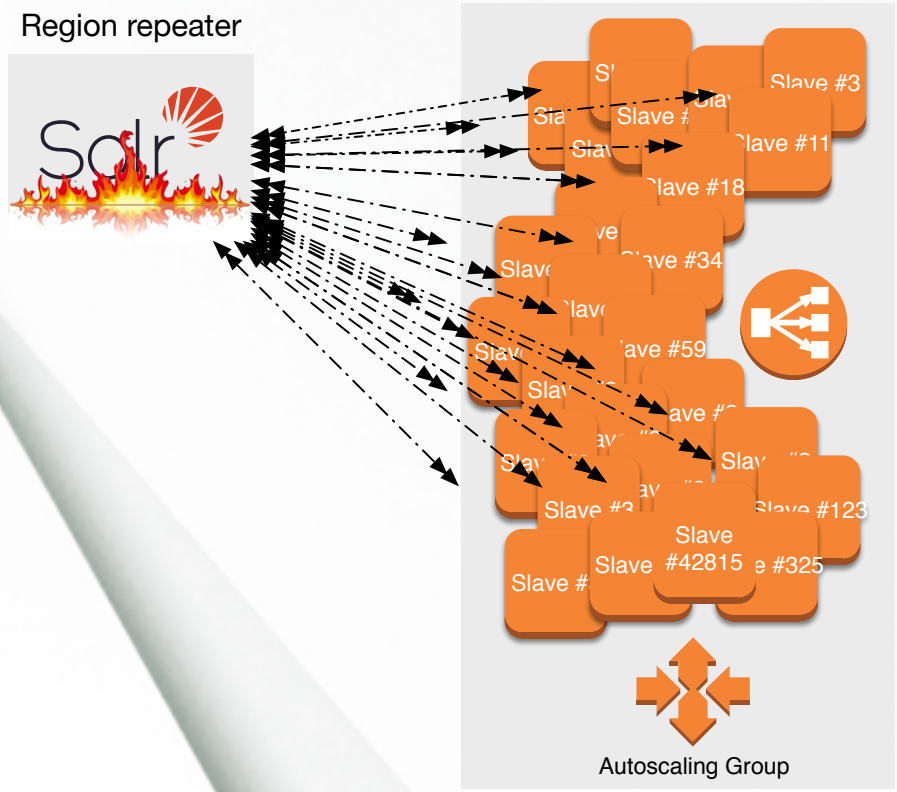


Autoscaling SOLr and API



http://www.docstoc.com/docs/109290533/Lucid-Imagination# by Erick Erickson

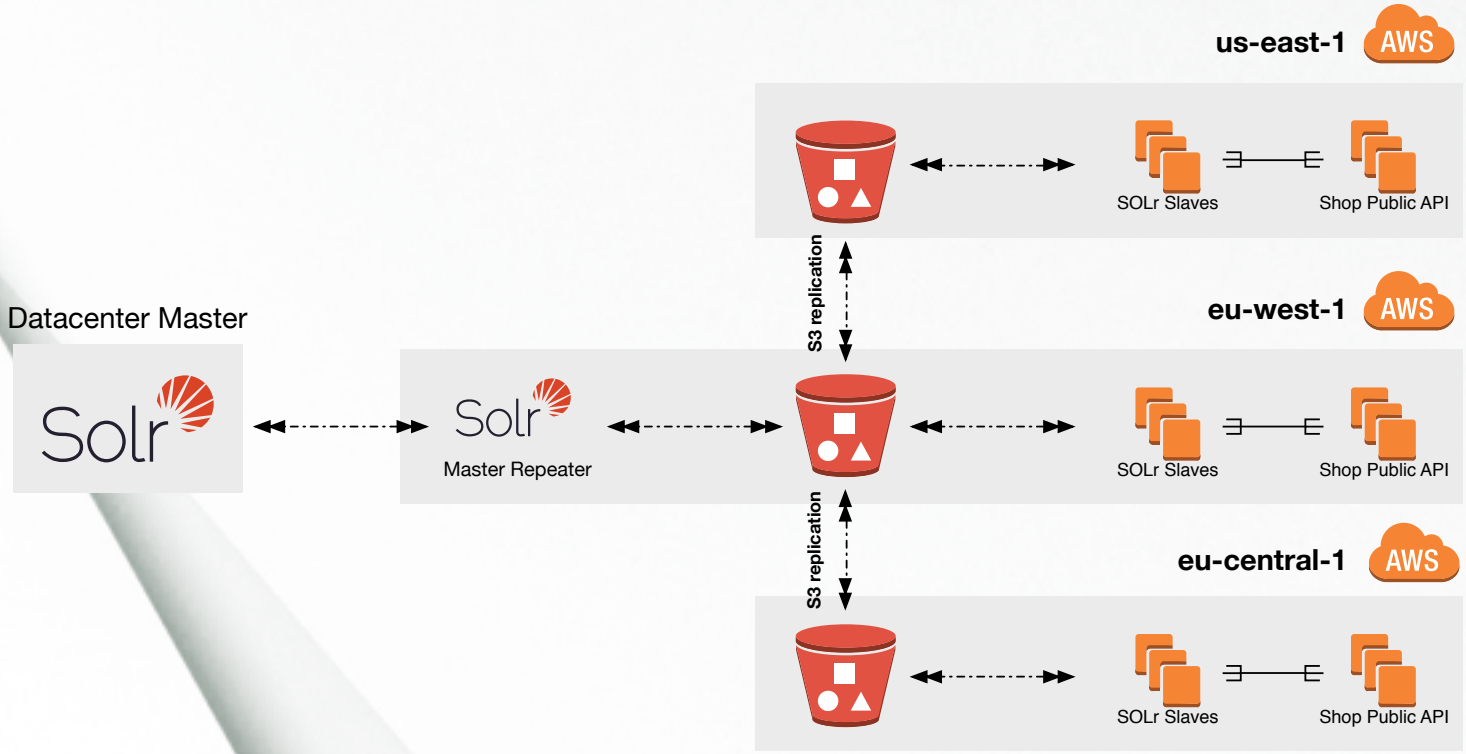
Scaling Limitations



Edited from https://flic.kr/p/zuBXM

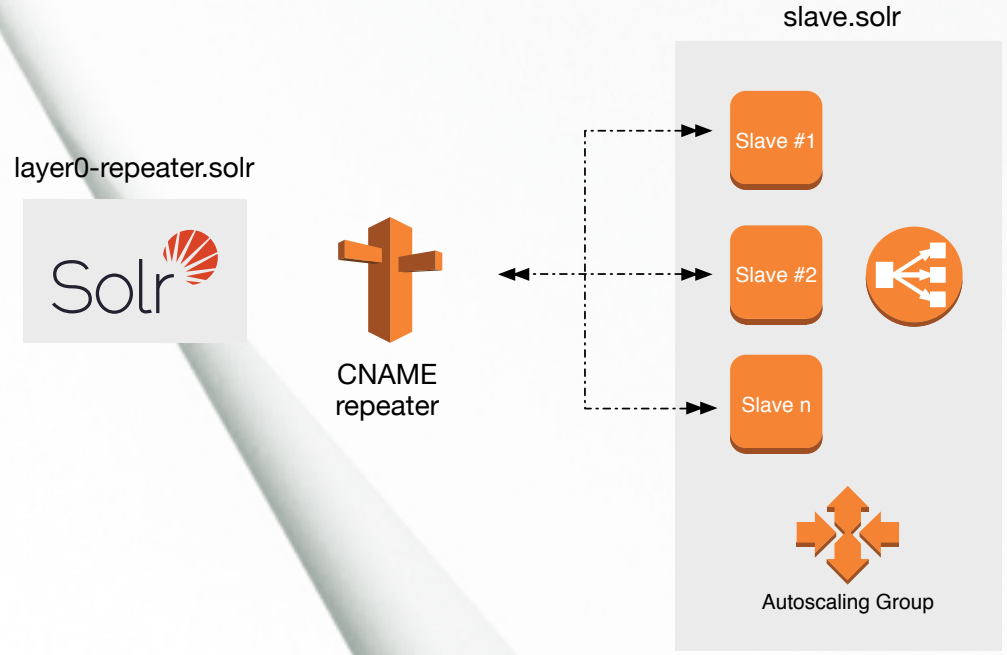

Y U no scale!

S3 Bucket for Replication



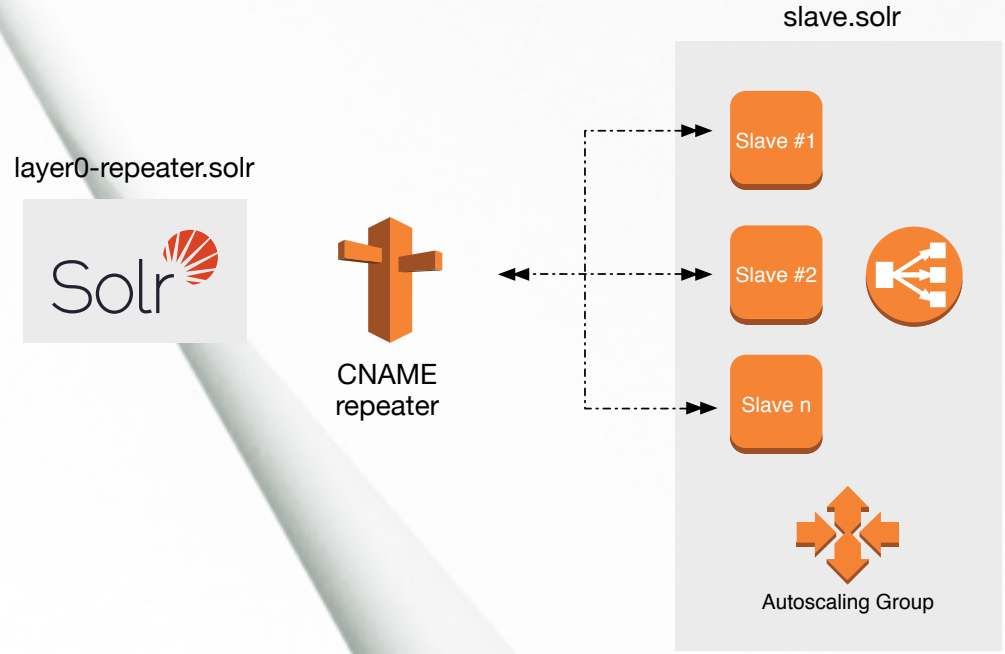

I see...

“Onion layers” for Replication



- Start with a single repeater layer - **layer0.solr**
- Setup a Route53 CNAME repeater that links to it - **repeater.solr**
- Entire slave fleet in the ASG is always configured to replicate from **repeater.solr**

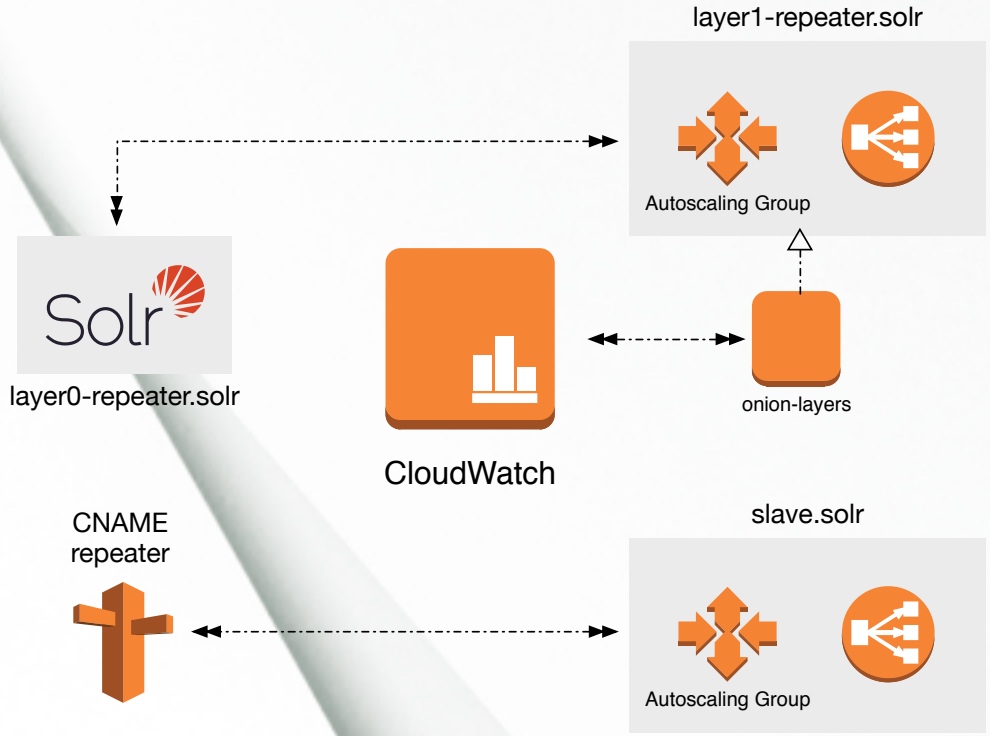
“Onion layers” for Replication



CloudWatch

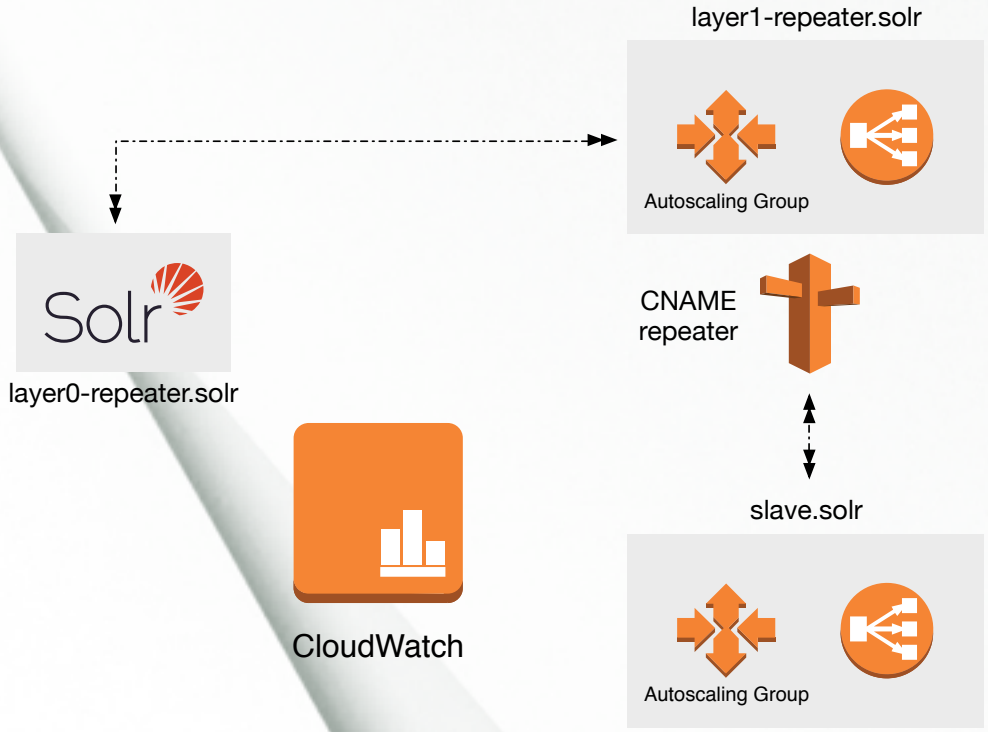
- Setup CloudWatch alarms for relevant metrics in the repeater layer. Give them some slack space.
- Configure it to send notifications to the **onion-layers-topic**.
- Bring up your instance of **onion-layers**.

“Onion layers” for Replication



- **onion-layers** creates a new ASG layer. Calculates ***currentLayer* = *currentLayer* + *1***
- The new layer starts replicating from **layer<current*Layer-1*>-repeater**
- Adds a new Route53 recordset **layer<*currentLayer*>-repeater**.

“Onion layers” for Replication



- After replication, the CNAME **repeater** is updated to link to **layer<*currentLayer*>-repeater**.
- **onion-layers** acts on alarms for the connection between **current layer** and **previous layer**.
- You can still have your own AS alarms for the connection between **slaves** and **current layer**.


Yes. These tools will be open-sourced soon

Thank you for listening

Check out our blog https://tech.zalando.com
Our many open source products https://github.com/zalando
The STUPS stack https://stups.io

Got more questions? You can reach us on twitter
@ZalandoTech
We're hiring!

Special thanks to Jessie Dude.
No Continuum Transfunctioners were harmed during the production of these slides.