



Microservice Architecture on AWS using AWS Lambda and Docker Containers

Danilo Poccia – AWS Technical Evangelist
 @danilop

Why **Micro**services?

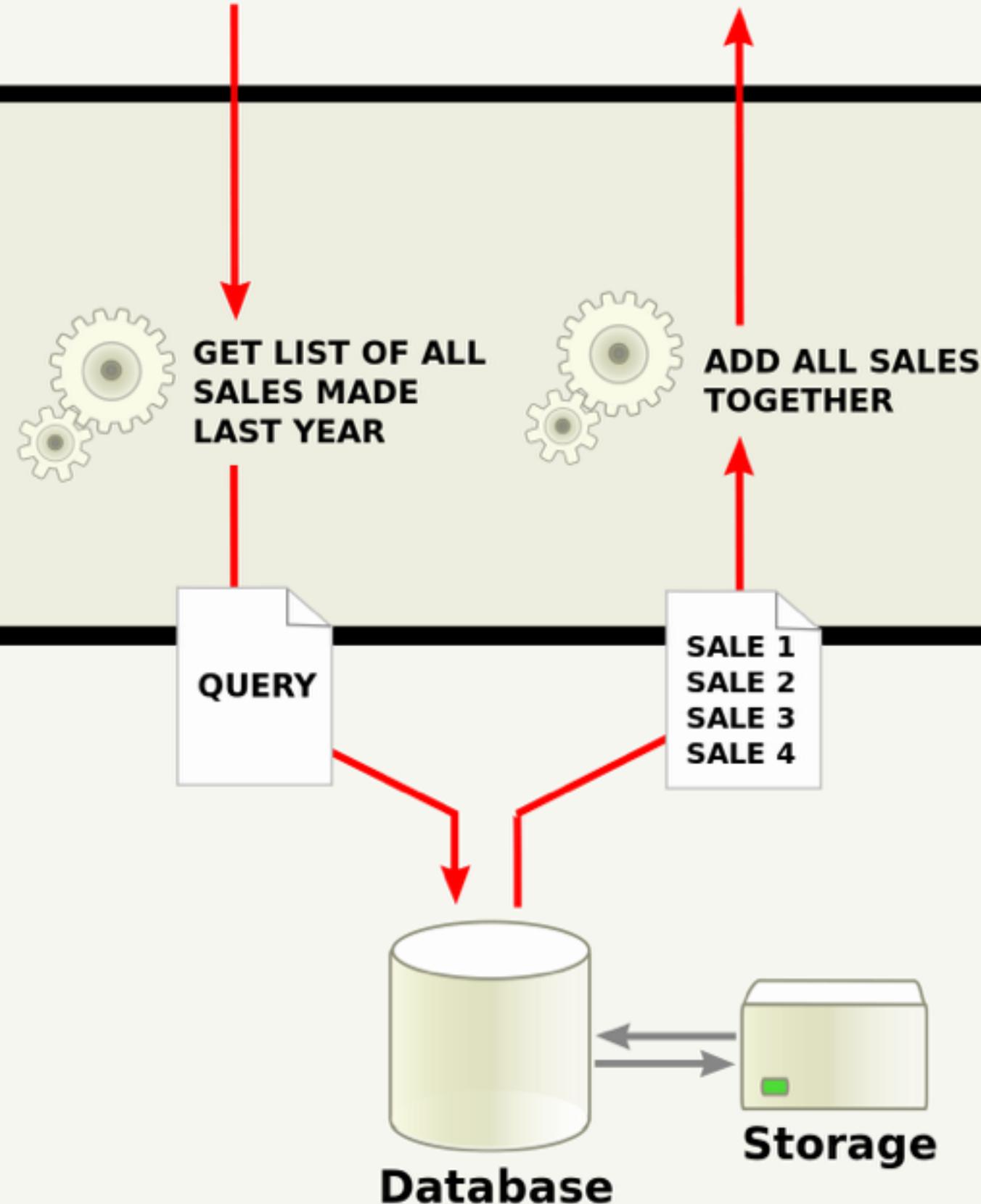
Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.



Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.



Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

As a Project scales Complexity arises

“Complexity arises when the dependencies among the elements become important.”

*Complex Adaptive Systems:
An Introduction to Computational Models of Social Life
Scott E. Page, John H. Miller*

How to design
smaller services?

Business Domain

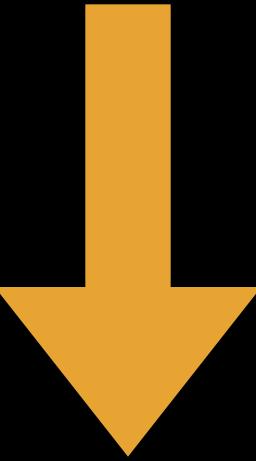
+

Loosely Coupled

+

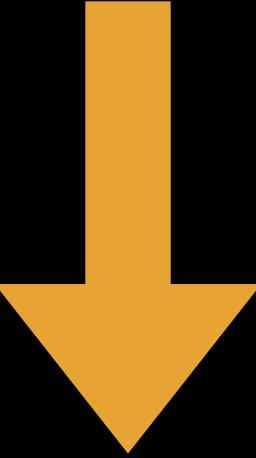
Bounded Context

Microservices



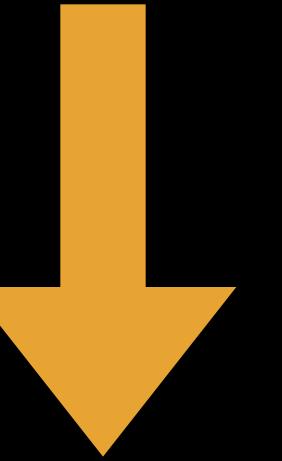
Independent Deployment

Microservices



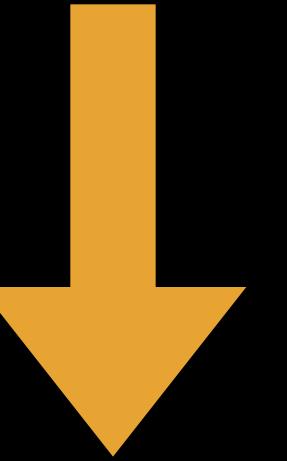
Choose the Right Tool

Microservices



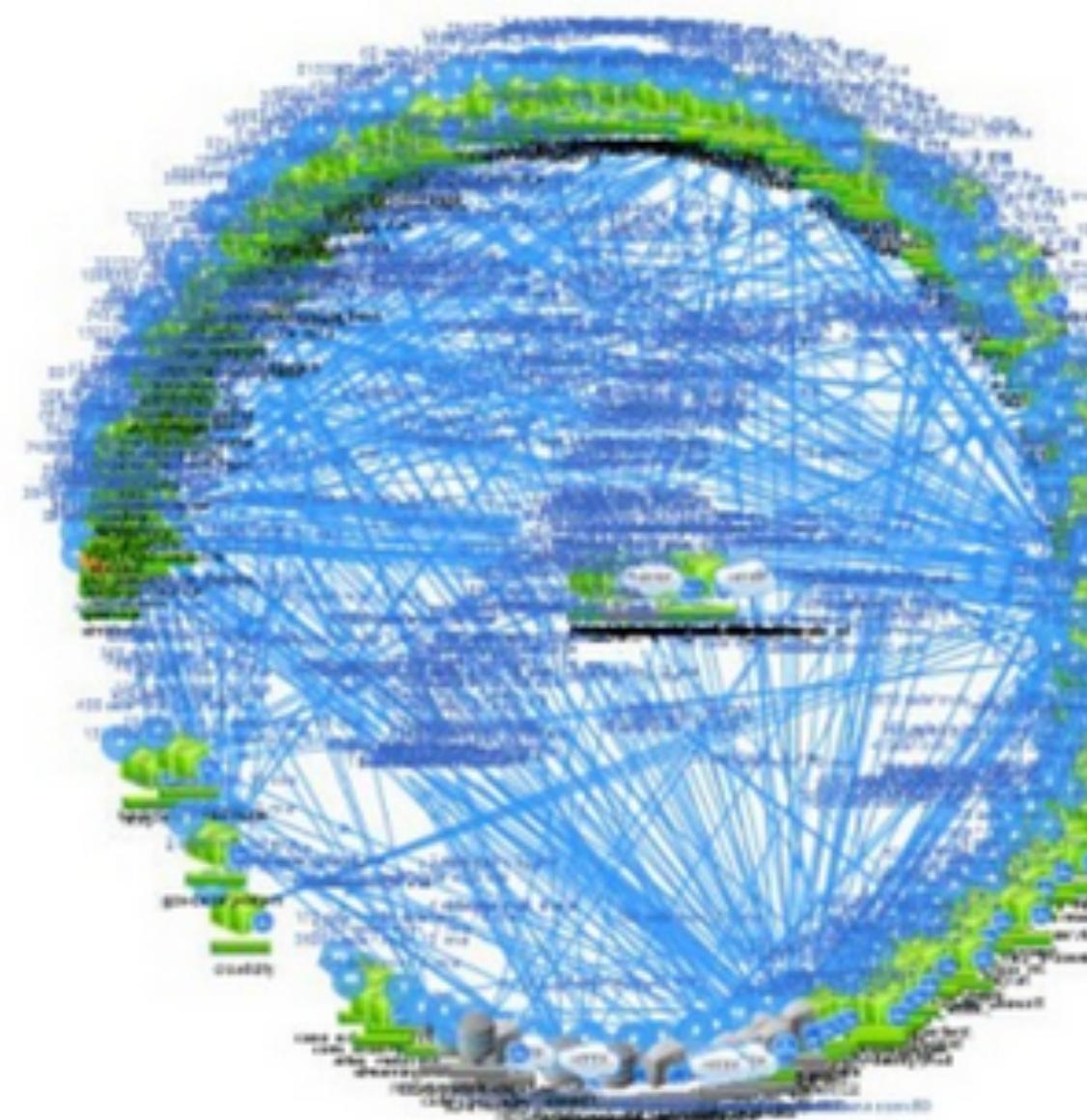
Adopt New Technologies

Microservices

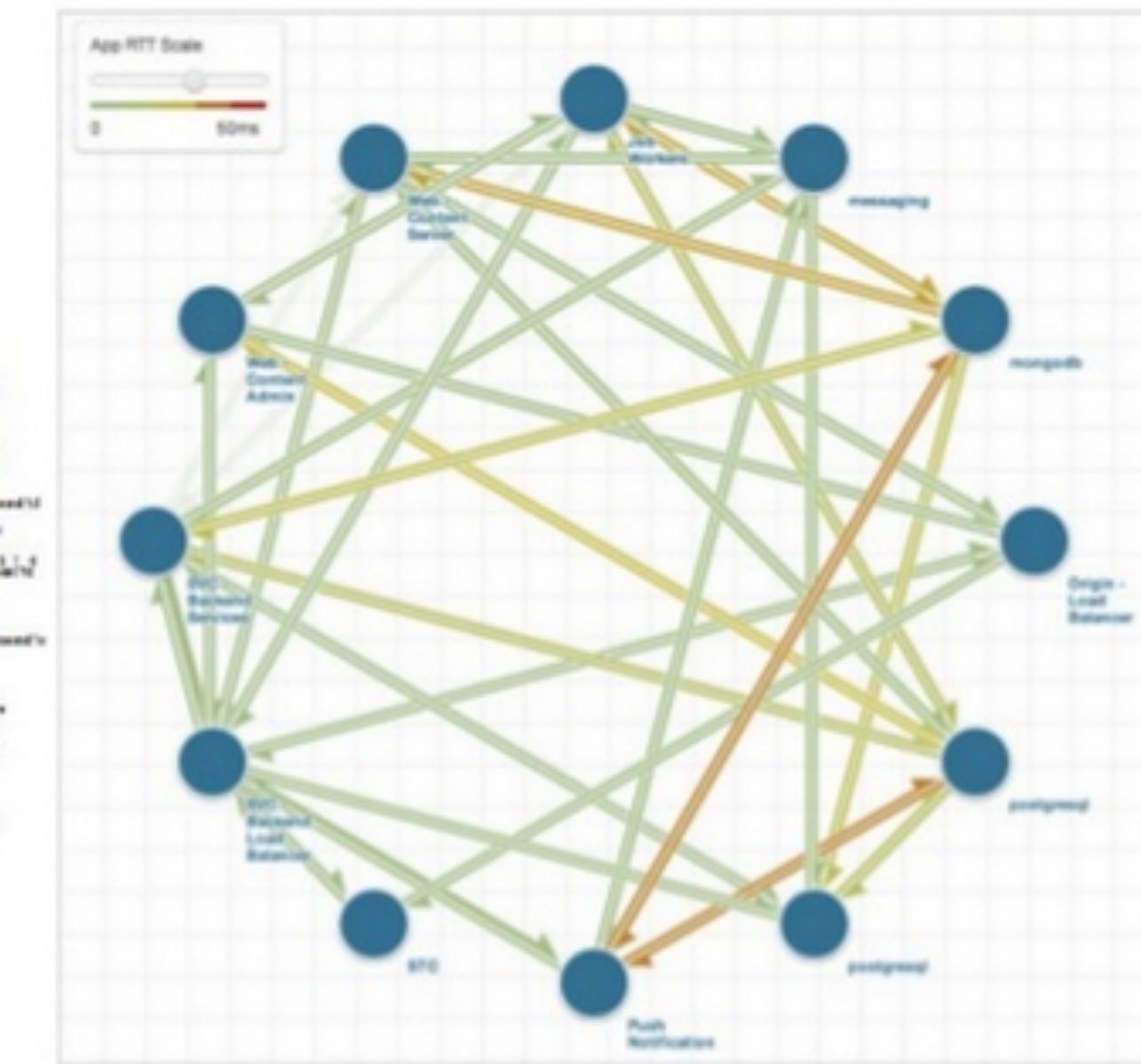


Culture of Automation

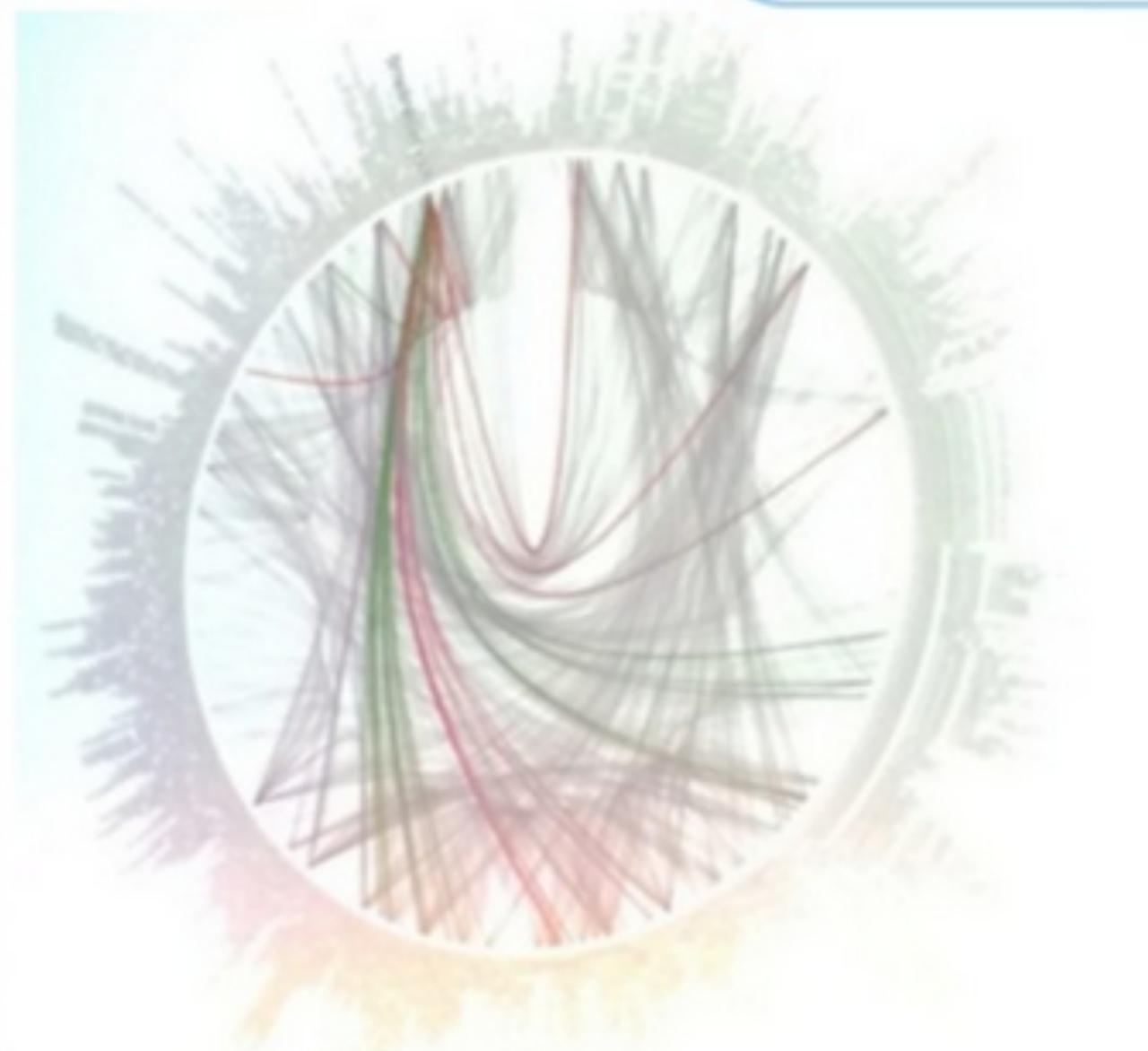
“Death Star” Architecture Diagrams



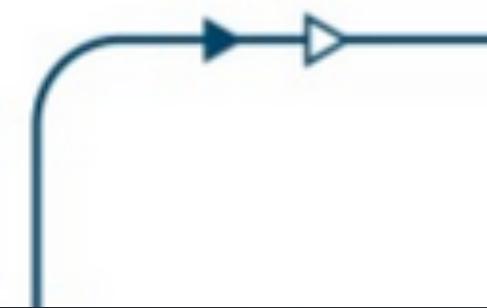
Netflix



Gilt Groupe (12 of 450)



Twitter



As visualized by Appdynamics, Boundary.com and Twitter internal tools

Adrian Cockcroft, Technology Fellow at Battery Ventures
<http://www.slideshare.net/adriancockcroft/goto-berlin>

How small is **small**?

“something that could be
rewritten in **two weeks**”

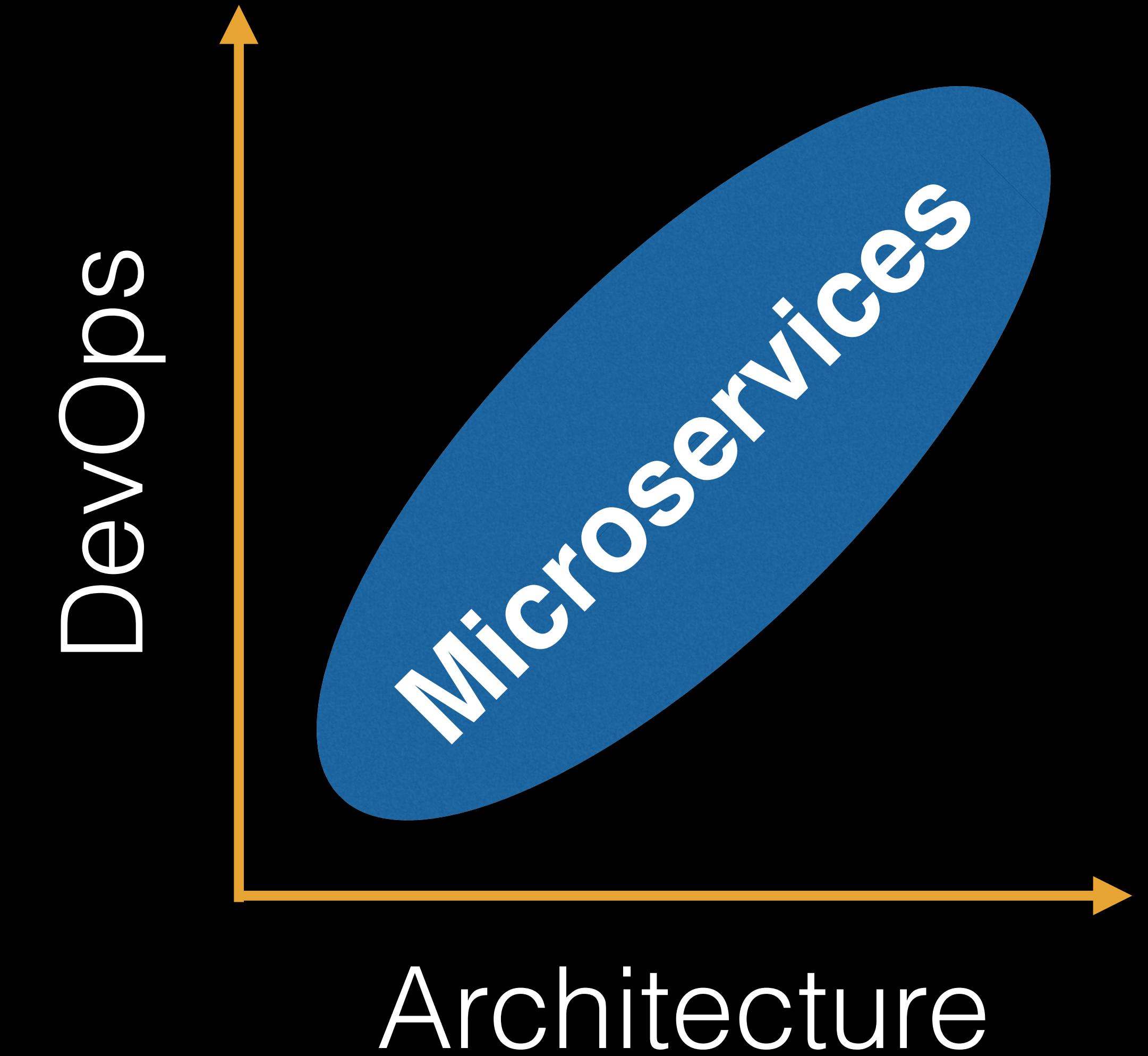
Two Pizza Teams



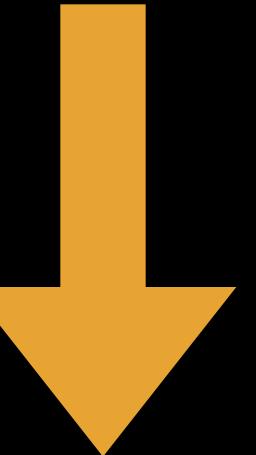
Conway's law

“organizations which design systems ...
are constrained to produce designs which
are copies of the communication
structures of these organizations”

Melvin Conway, 1968



Distributed Systems



Independent Scalability Auto Scaling

Security

Least Privileges

Single Sign-On

Confused Deputy Problem
(for downstream calls
after authentication)

Testing

Automate

Service Tests

End-to-end Tests

Synthetic Monitoring

Monitoring

Expose Service Metrics

Standard Log Format

Correlation ID

From Log collection to
Real-Time Event Routing

Amazon CloudWatch Logs
Amazon Kinesis

Understand your Trends

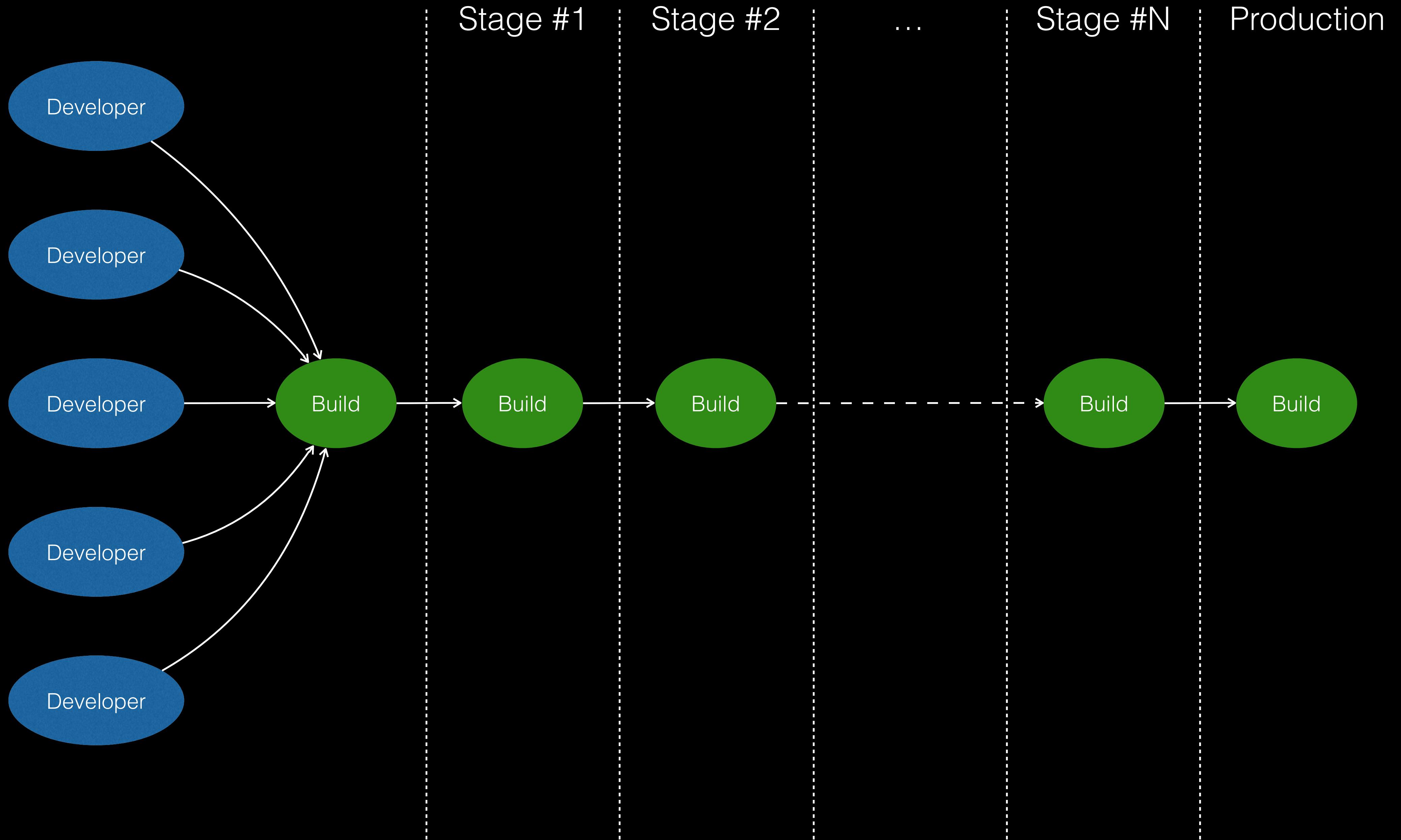
Self-Describing System

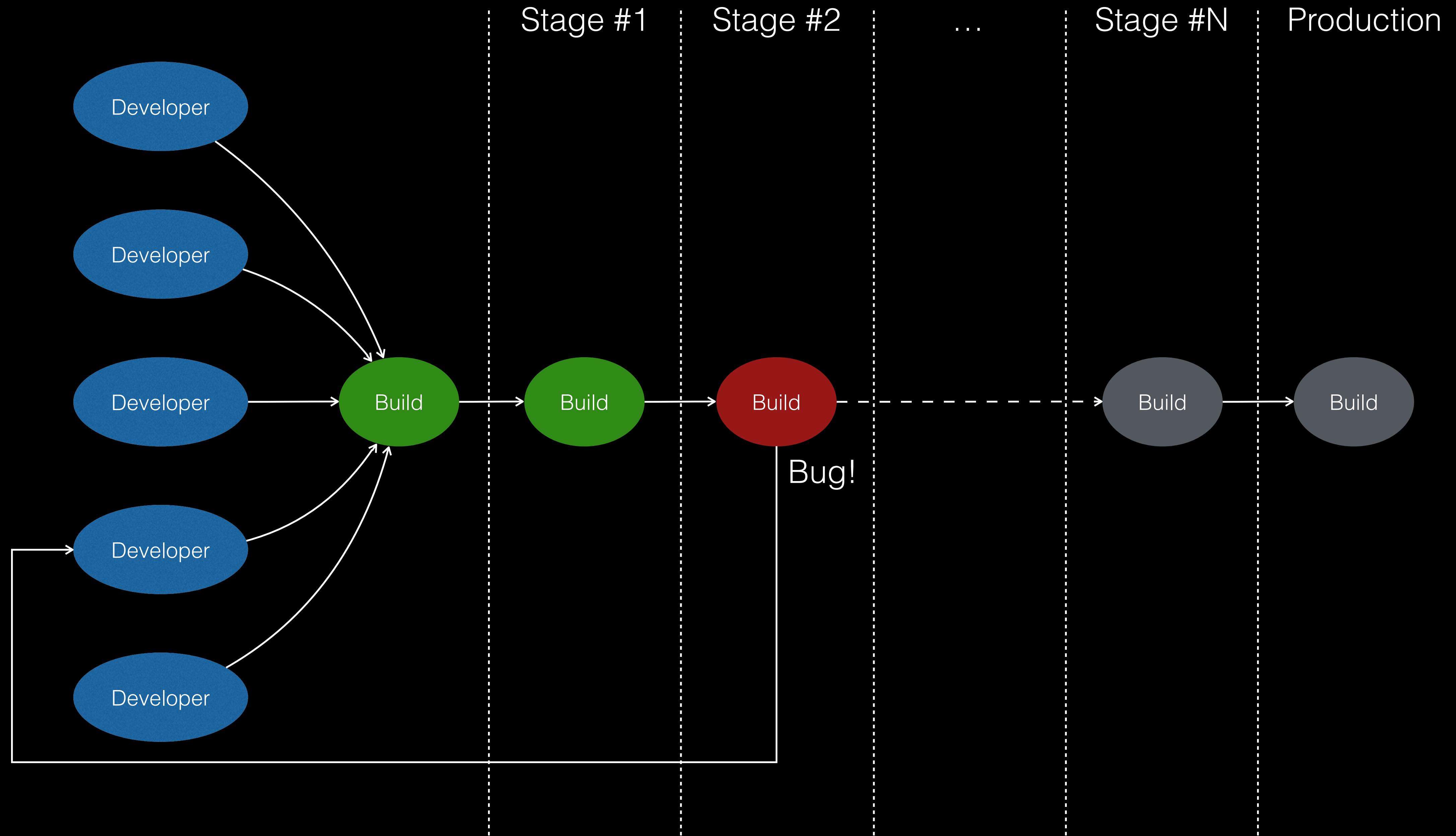
Discovery

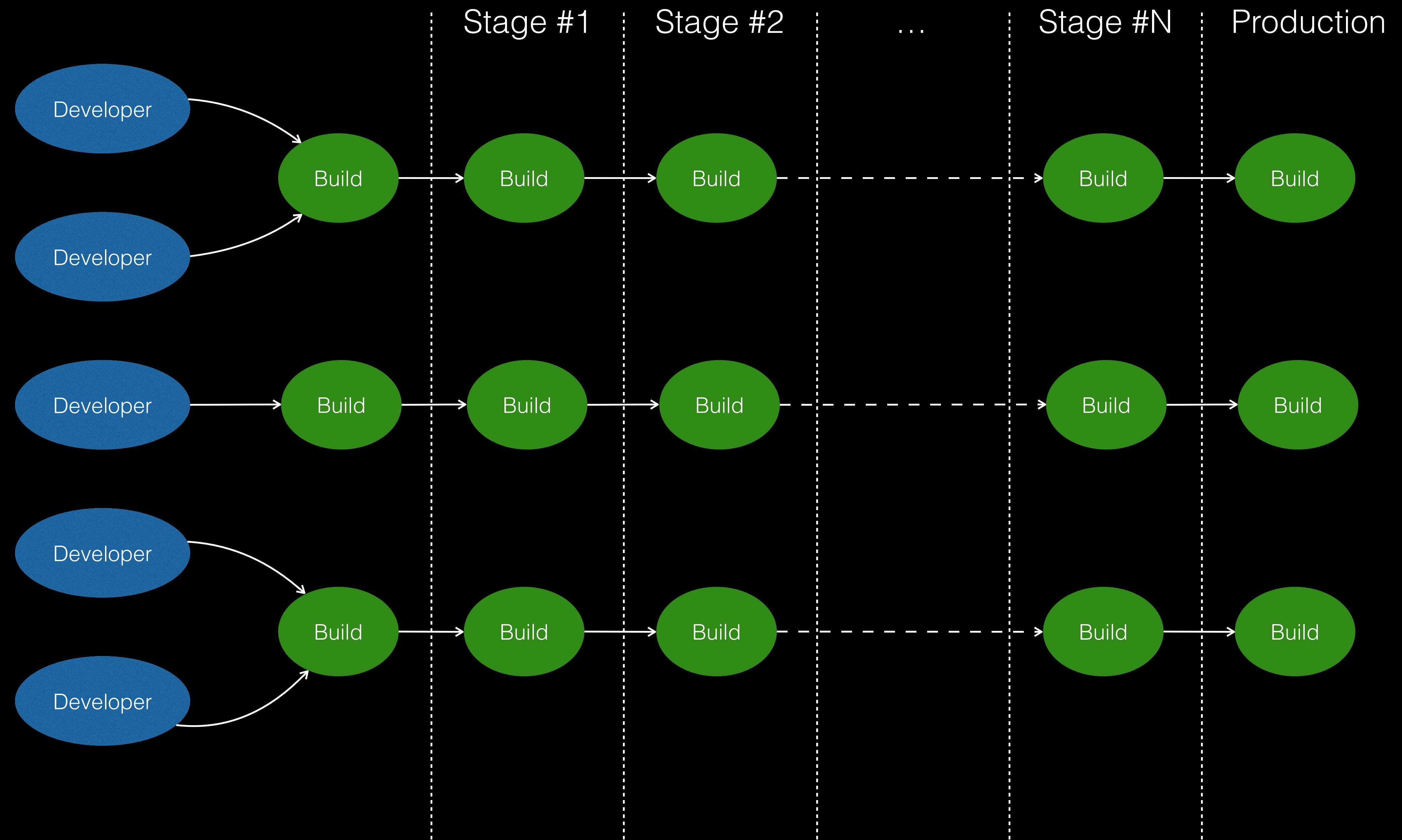
Amazon Route 53
(DNS)

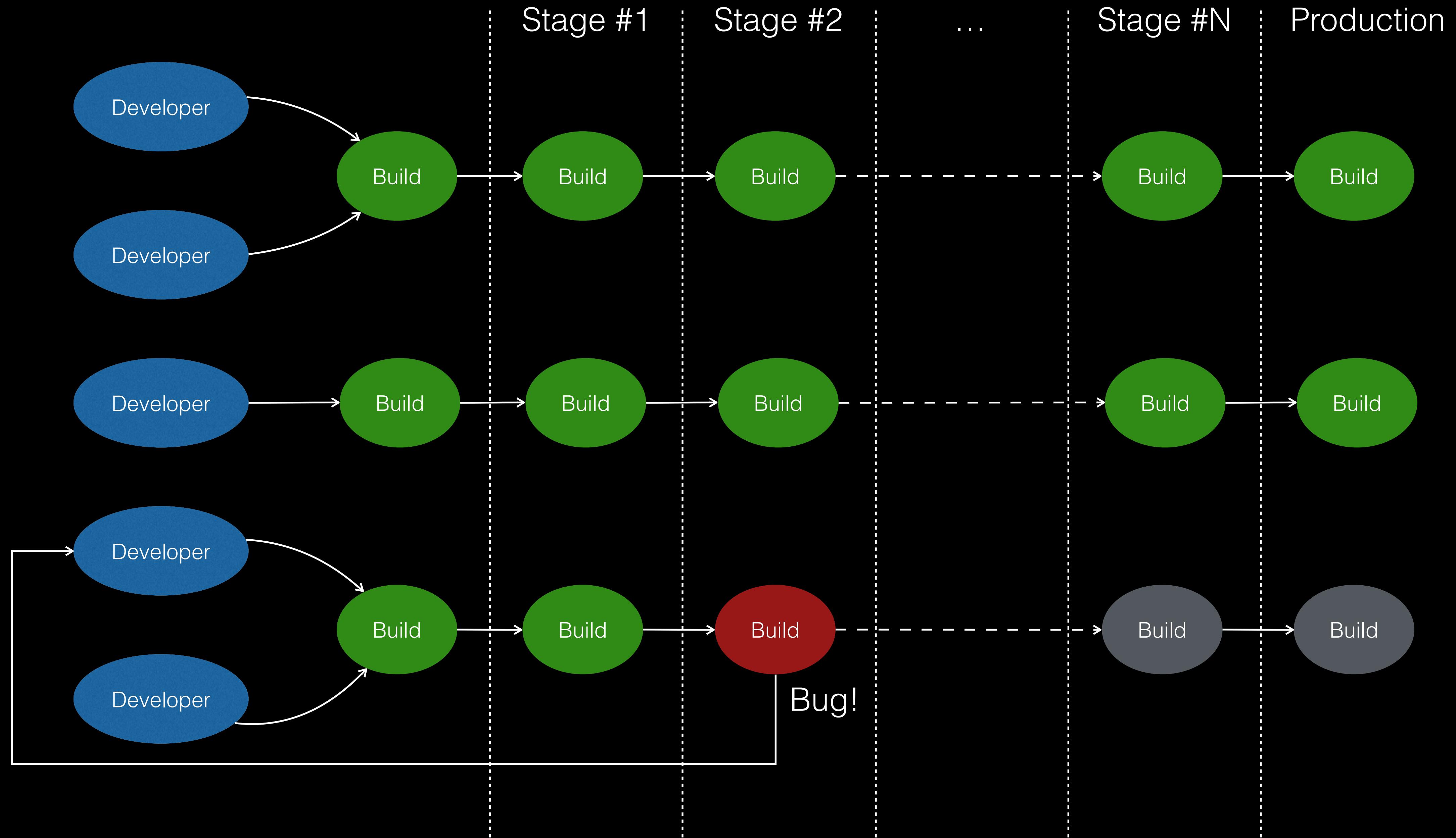
AWS Resource
Tagging

Deployment Pipeline

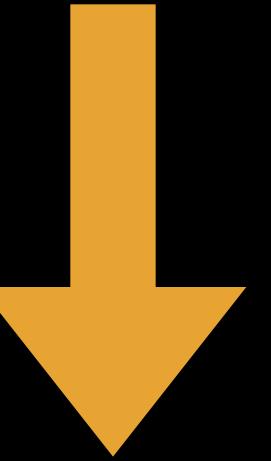








Design for Failure



Degradate Functionality

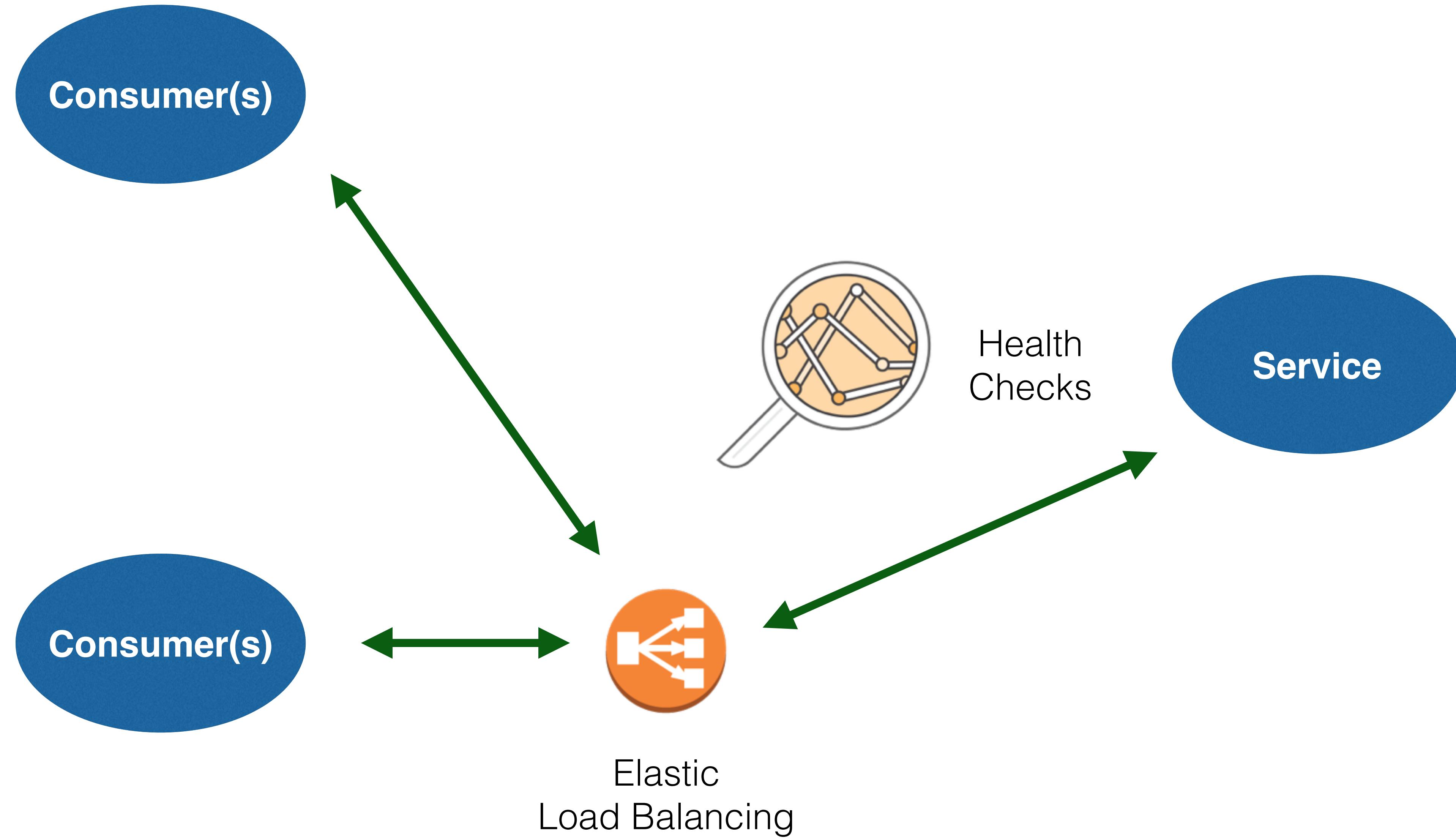
Circuit Breakers



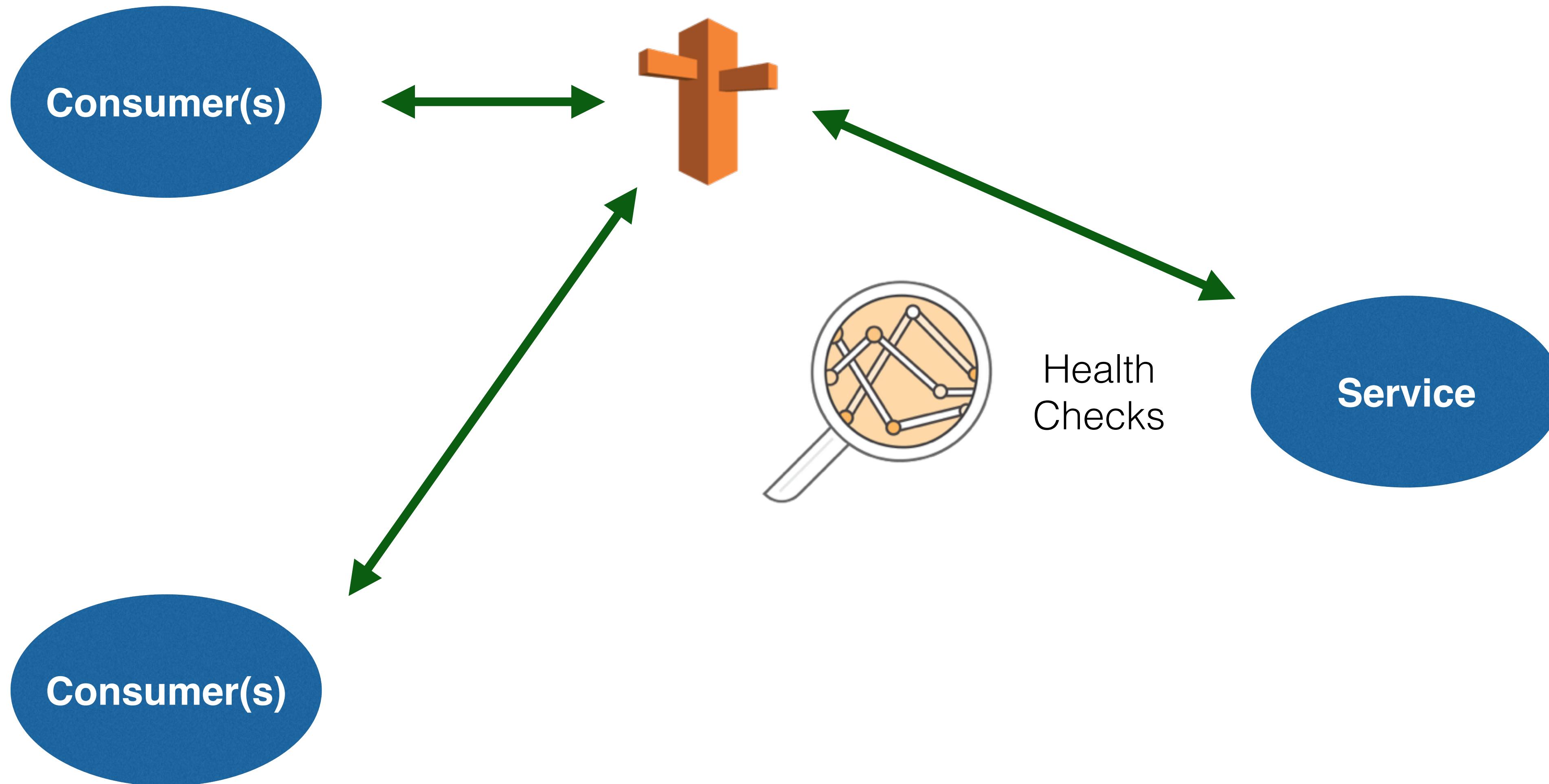
"Jtecul" by own - Own work.

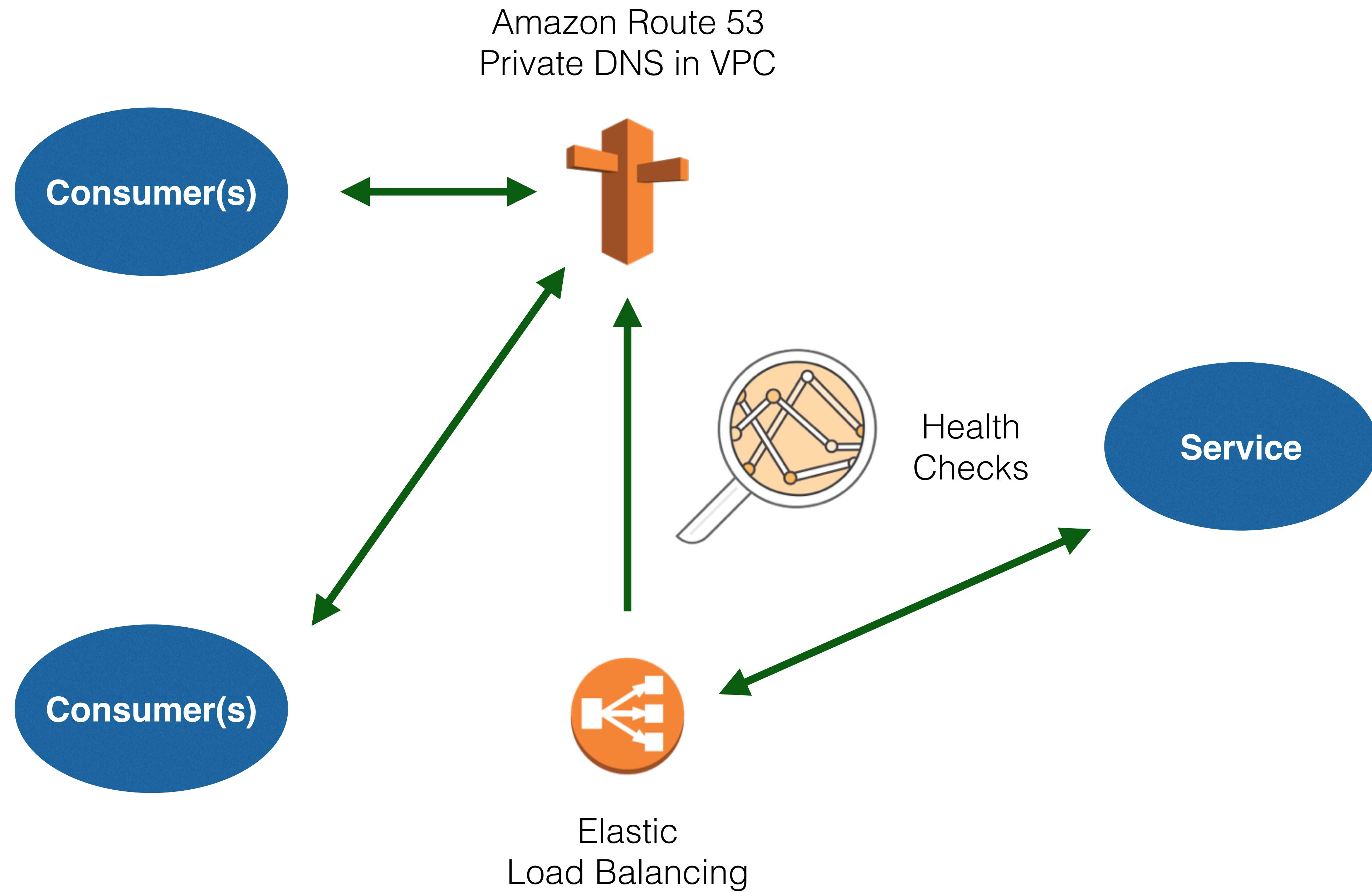
Licensed under CC BY-SA 3.0 via Wikimedia Commons

<http://commons.wikimedia.org/wiki/File:Jtecul.jpg#/media/File:Jtecul.jpg>

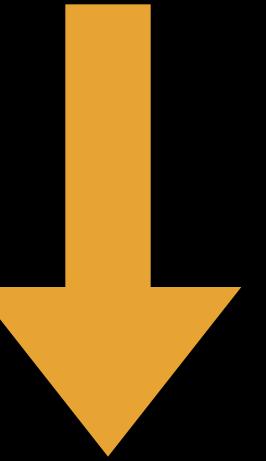


Amazon Route 53
Private DNS in VPC



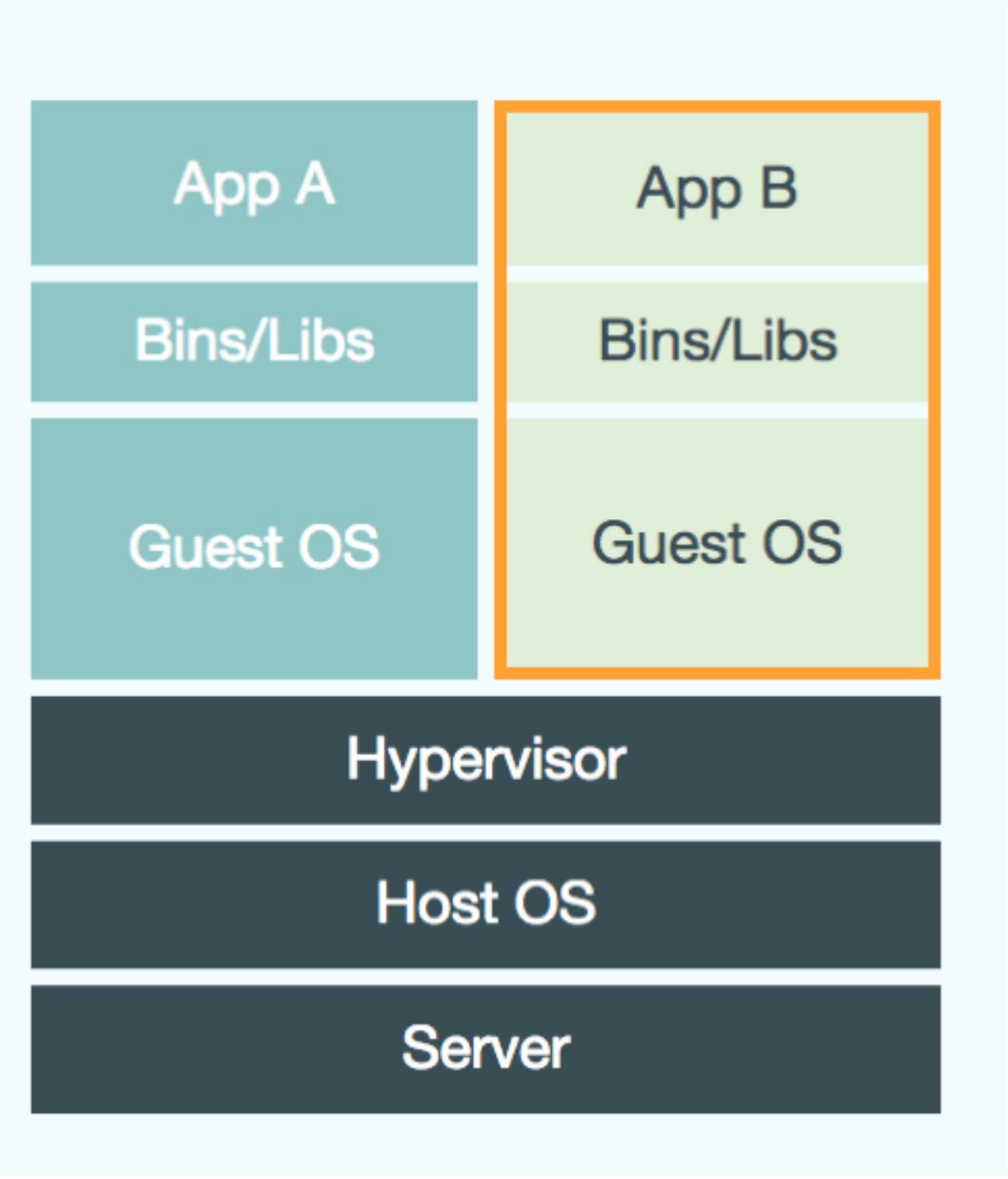


Disposable Infrastructure

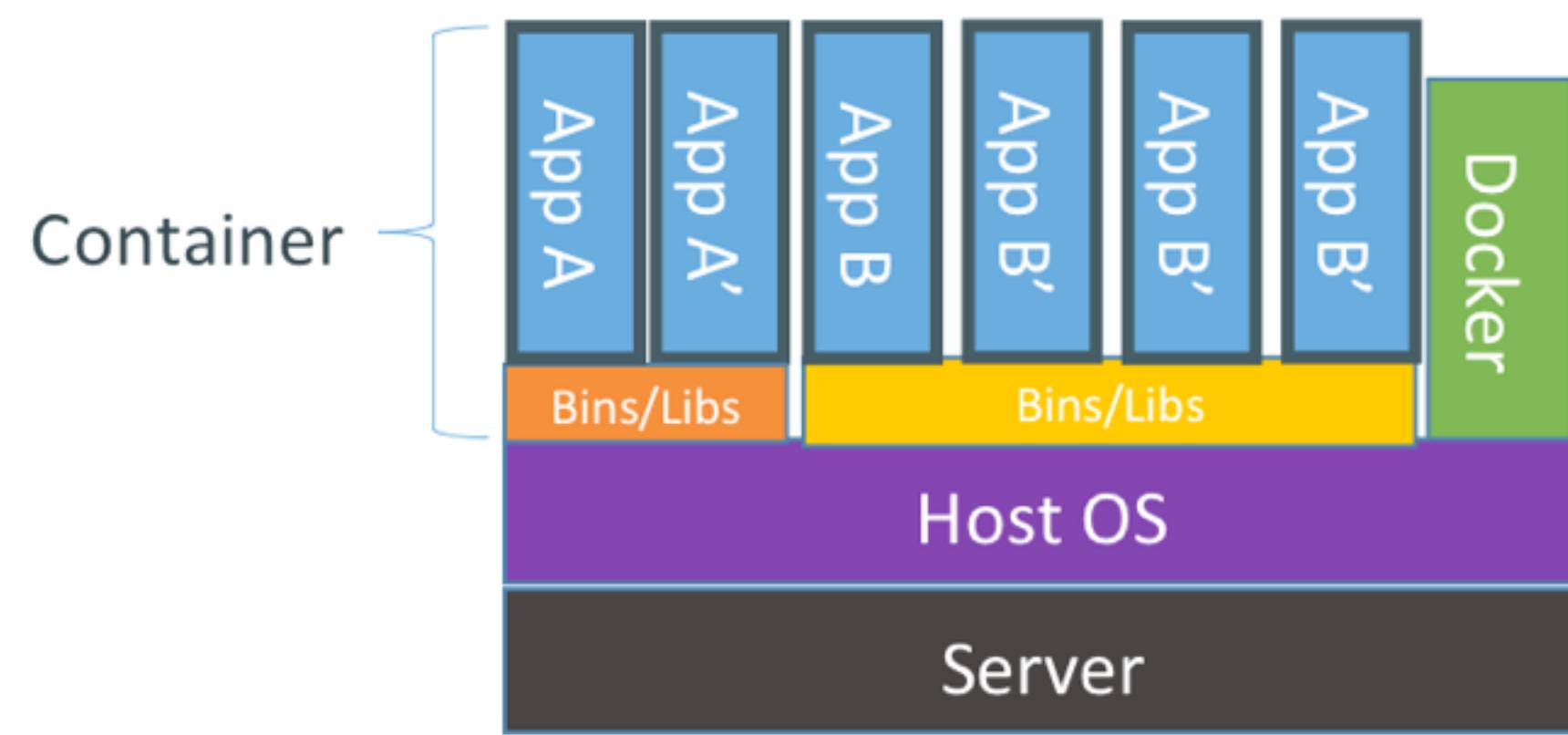


Immutable Deployments

Why Docker?



Virtual Machine



Container

Docker on AWS

**Amazon
Linux**

A supported and maintained Linux image provided by Amazon Web Services

**AWS
Elastic
Beanstalk**

For deploying and scaling web applications and services

**Amazon EC2
Container Service**

Highly scalable, high performance container management service

Amazon EC2 instances

Docker daemon

Amazon ECS agent

Amazon EC2 Container Service

Key Components

Container Instances

Clusters

Tasks

Task Definitions

The screenshot shows the GitHub repository page for `aws/amazon-ecs-agent`. The repository has 59 commits, 2 branches, 1 release, and 6 contributors. The master branch is selected. The commit history shows several changes related to the agent's functionality and documentation. The README.md file is also visible.

Code

- Issues (6)
- Pull Requests (2)
- Wiki
- Pulse
- Graphs

SSH clone URL
git@github.com:aws/amazon-ecs-agent

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

[Clone in Desktop](#) [Download ZIP](#)

Commits

File	Description	Date
agent	Handle default cluster when restoring state	21 days ago
misc	Initial VolumesFrom support	27 days ago
scripts	Handle default cluster when restoring state	21 days ago
.gitignore	Minor doc fix	a month ago
.travis.yml	Simple travis.yml; run non-integ tests	a month ago
CHANGELOG.md	Bump version to 0.0.3	21 days ago
LICENSE	Update copyright year	a month ago
Makefile	Initial VolumesFrom support	27 days ago
NOTICE	Update copyright year	a month ago
README.md	Merge v0.0.3 agent changes into master	21 days ago
VERSION	Bump version to 0.0.3	21 days ago

README.md

Amazon ECS Container Agent

The Amazon ECS Container Agent is software developed for the [Amazon EC2 Container Service](#). It runs on Container Instances and starts containers on behalf of Amazon ECS.

<https://github.com/aws/amazon-ecs-agent>

Regional

Resource pool

Grouping of Container Instances

Start empty, dynamically scalable

Amazon EC2 Container Service

Key Components

Container Instances

Clusters

Tasks

Task Definitions

Amazon EC2 Container Service

Unit of work

Grouping of related Containers

Run on Container Instances

Key Components

Container Instances

Clusters
Tasks

Task Definitions

```
[  
  {  
    "image": "mysql",  
    "name": "db",  
    "cpu": 10,  
    "memory": 500,  
    ...
```

Amazon EC2 Container Service

Key Components

Container Instances

Clusters

Tasks

Task Definitions

Tasks are defined via Task Definitions

```
[  
 {  
   "image": "tutum/wordpress-stackable",  
   "name": "wordpress",  
   "cpu": 10,  
   "memory": 500,  
   "essential": true,  
   "links": [  
     "db"  
   ],  
   "entryPoint": [  
     "/bin/sh",  
     "-c"  
   ],  
   "environment": [  
     ...  
   ],  
   "portMappings": [  
     {  
       "containerPort": 80,  
       "hostPort": 80  
     }  
   ]  
 },
```

```
{  
   "image": "mysql",  
   "name": "db",  
   "cpu": 10,  
   "memory": 500,  
   "essential": true,  
   "entryPoint": [  
     "/entrypoint.sh"  
   ],  
   "environment": [  
     {  
       "name": "MYSQL_ROOT_PASSWORD",  
       "value": "pass"  
     }  
   ],  
   "portMappings": []  
 }
```

Tasks are defined via Task Definitions

From Docker Hub

10 CPU Units (1024 is full CPU),
500 Megabytes of Memory

Environment Variables

No external ports exposed

```
{  
  "image": "mysql",  
  "name": "db",  
  "cpu": 10,  
  "memory": 500,  
  "essential": true,  
  "entryPoint": [  
    "/entrypoint.sh"  
,  
  "environment": [  
    {  
      "name": "MYSQL_ROOT_PASSWORD",  
      "value": "pass"  
    }  
,  
  "portMappings": []  
]
```

Tasks are defined via Task Definitions

```
[  
 {  
   "image": "tutum/wordpress-stackable",  
   "name": "wordpress",  
   "cpu": 10,  
   "memory": 500,  
   "essential": true, ←  
   "links": [  
     "db" ←  
   ],  
   "entryPoint": [  
     "/bin/sh",  
     "-c"  
   ],  
   "environment": [  
     ...  
   ],  
   "portMappings": [  
     {  
       "containerPort": 80,  
       "hostPort": 80  
     }  
   ]  
 },  
 ]
```

Essential to our Task

Docker link to mysql container

Expose port 80 in container
to port 80 on host

Why AWS Lambda?



AWS Lambda

**Event driven,
fully managed compute**

AWS Lambda

**Focus on business logic,
not infrastructure**



Customer uploads code,
AWS Lambda handles:

Capacity
Scaling
Deployment
Fault tolerance
Monitoring
Logging

...



AWS Lambda

Automatic scaling

Customers pay only
for what they use,
no over/under provisioning

AWS Lambda



Fine-grained pricing

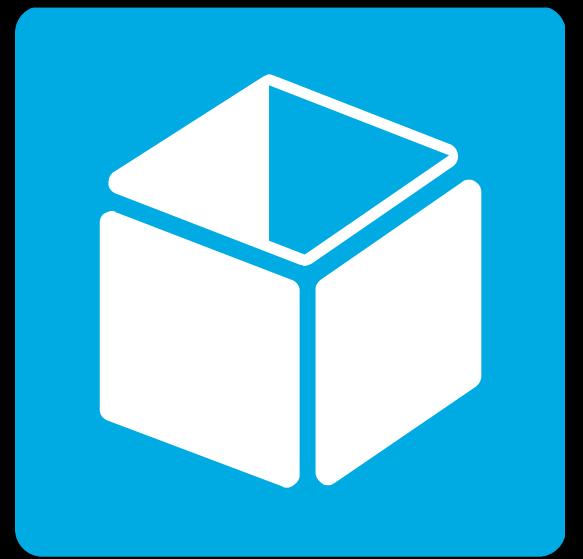
Price compute time by 100ms,
even short jobs make sense

Low request charge

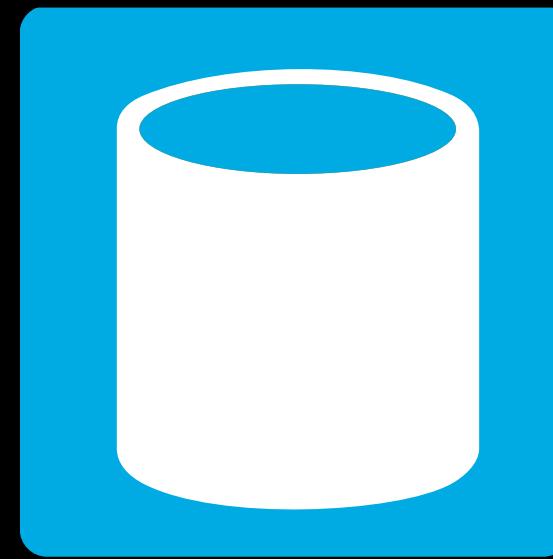
No hourly, daily, or monthly minimums

Free tier

Events come in many **different** shapes & sizes



S3 event
notifications



DynamoDB
Streams

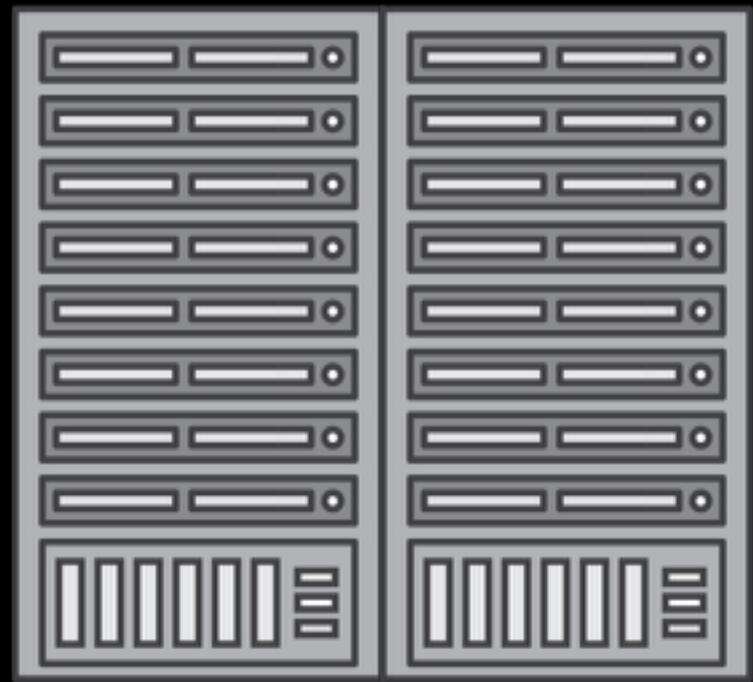


Kinesis events



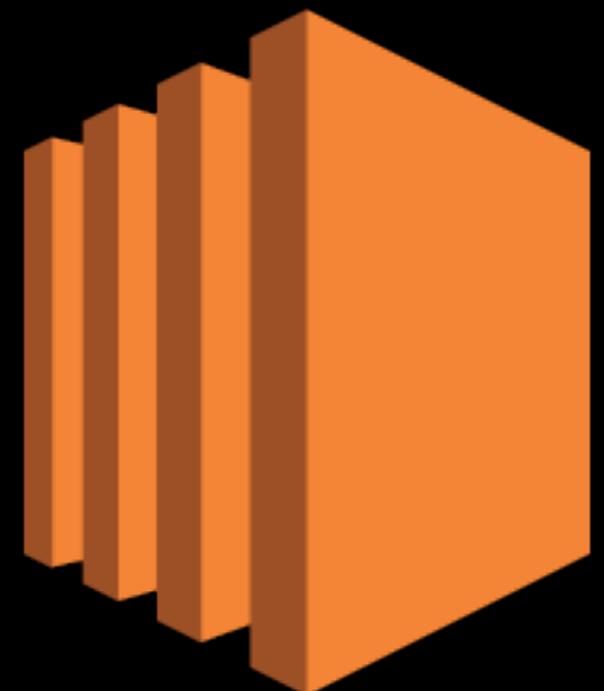
Custom
events

Weeks



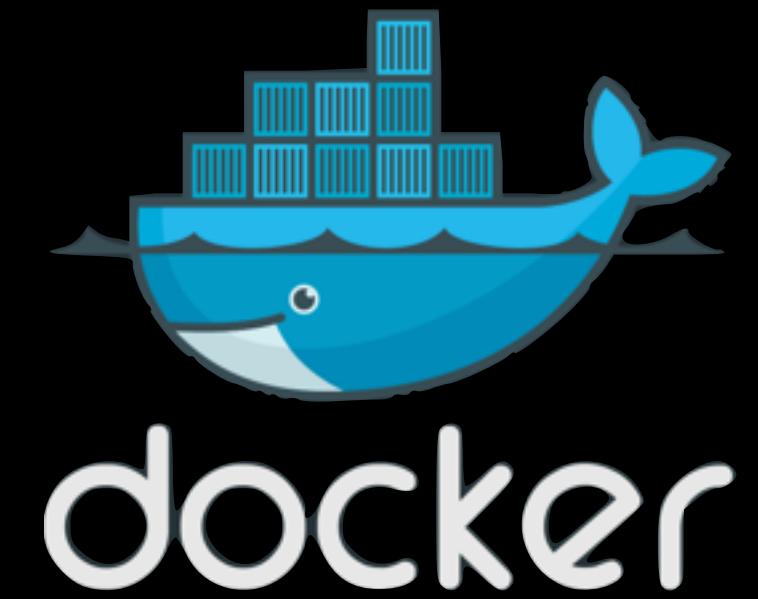
On-Premises

Minutes



Amazon EC2

Seconds



Containers

Milliseconds



AWS Lambda

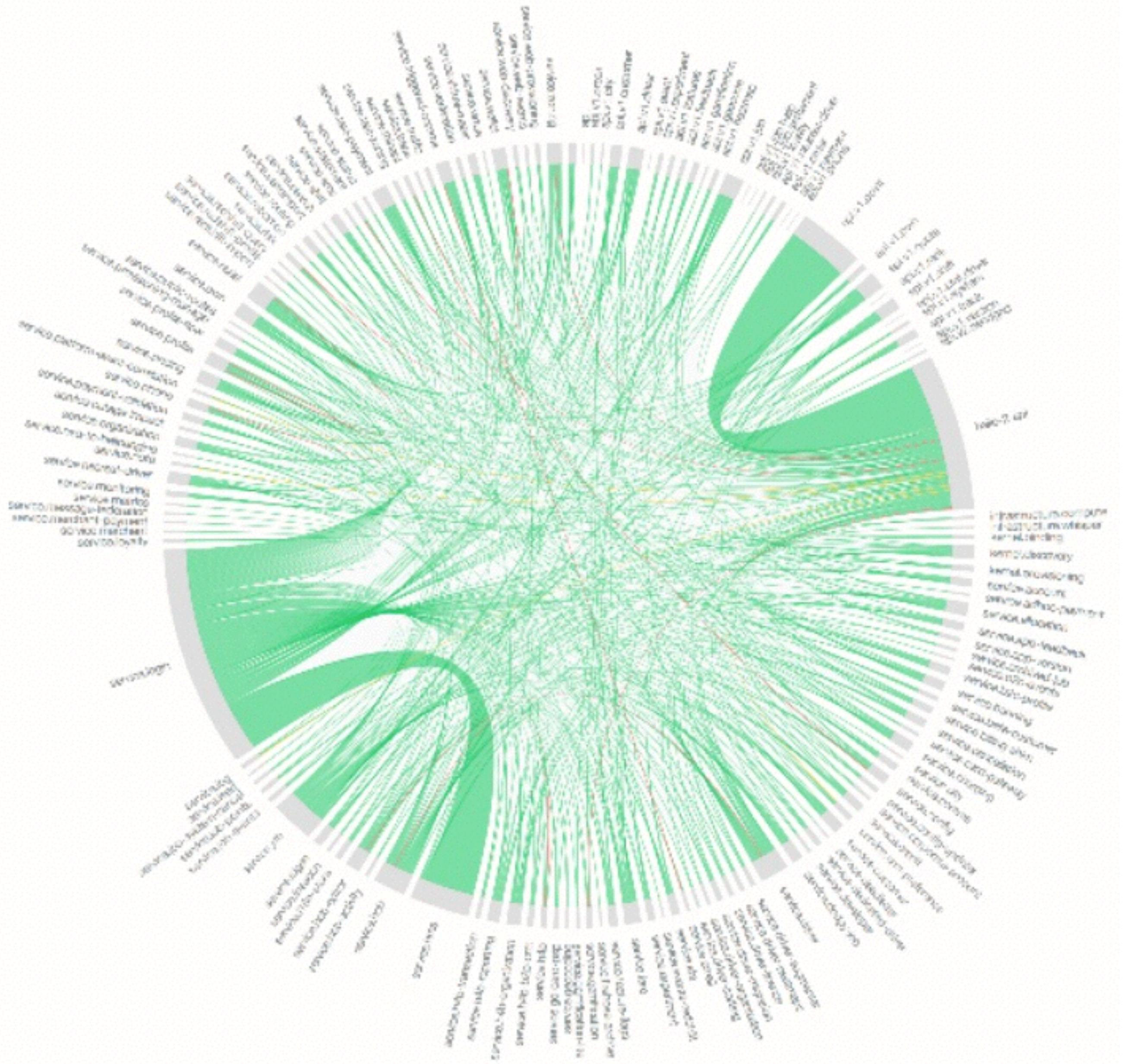
Netflix Architecture

Asgard, Amimator, Hystrix,
Cassandra, JVM, Docker, ...



Hailo Architecture

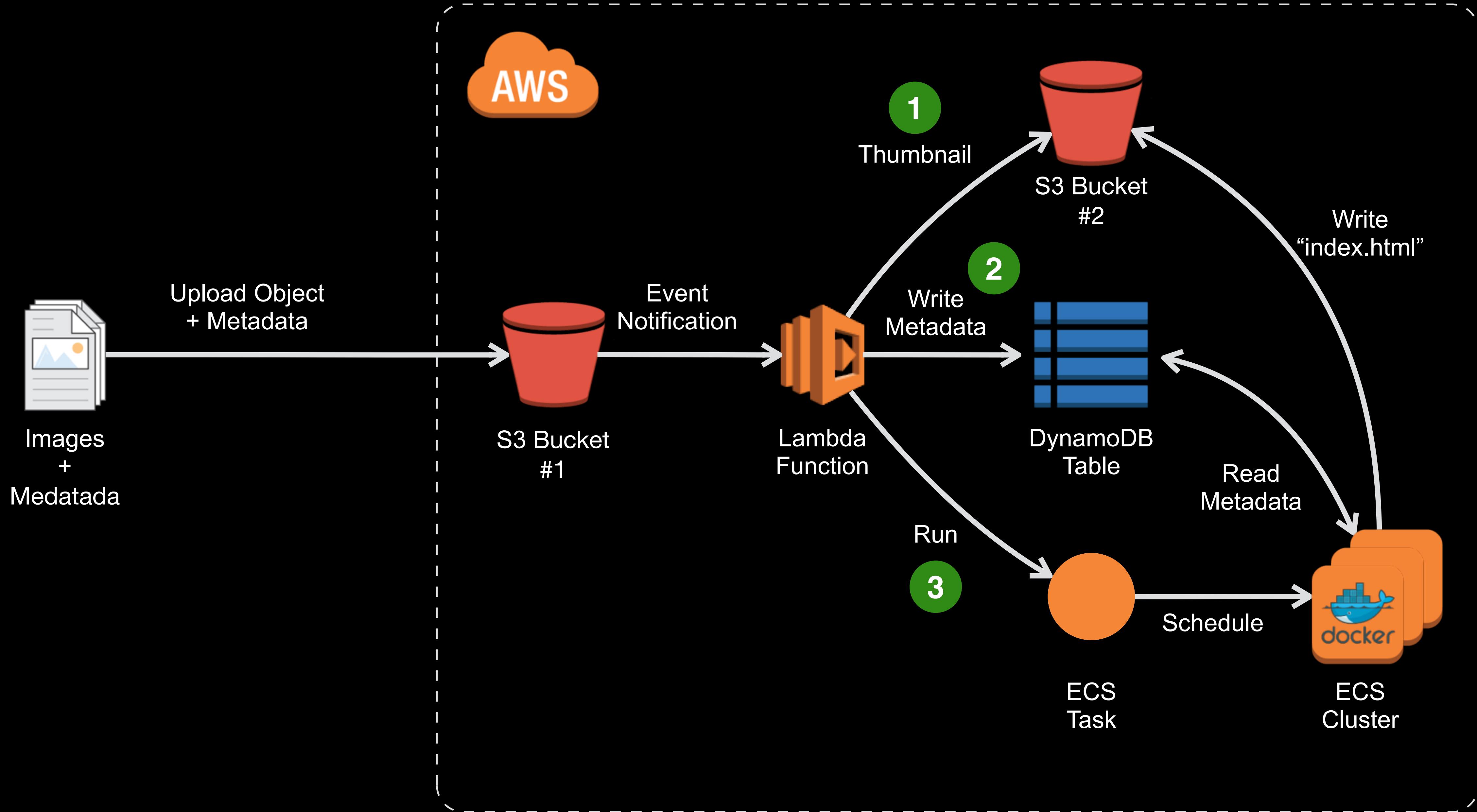
Hubot, Janky, Jenkins,
Go, RabbitMQ, Cassandra,
Docker, ...



On AWS

Demo Architecture

Content Management System
Prototype



<demo>

...

</demo>

Iтеративный
Продолжительное улучшение
Кайзен



 @danilop