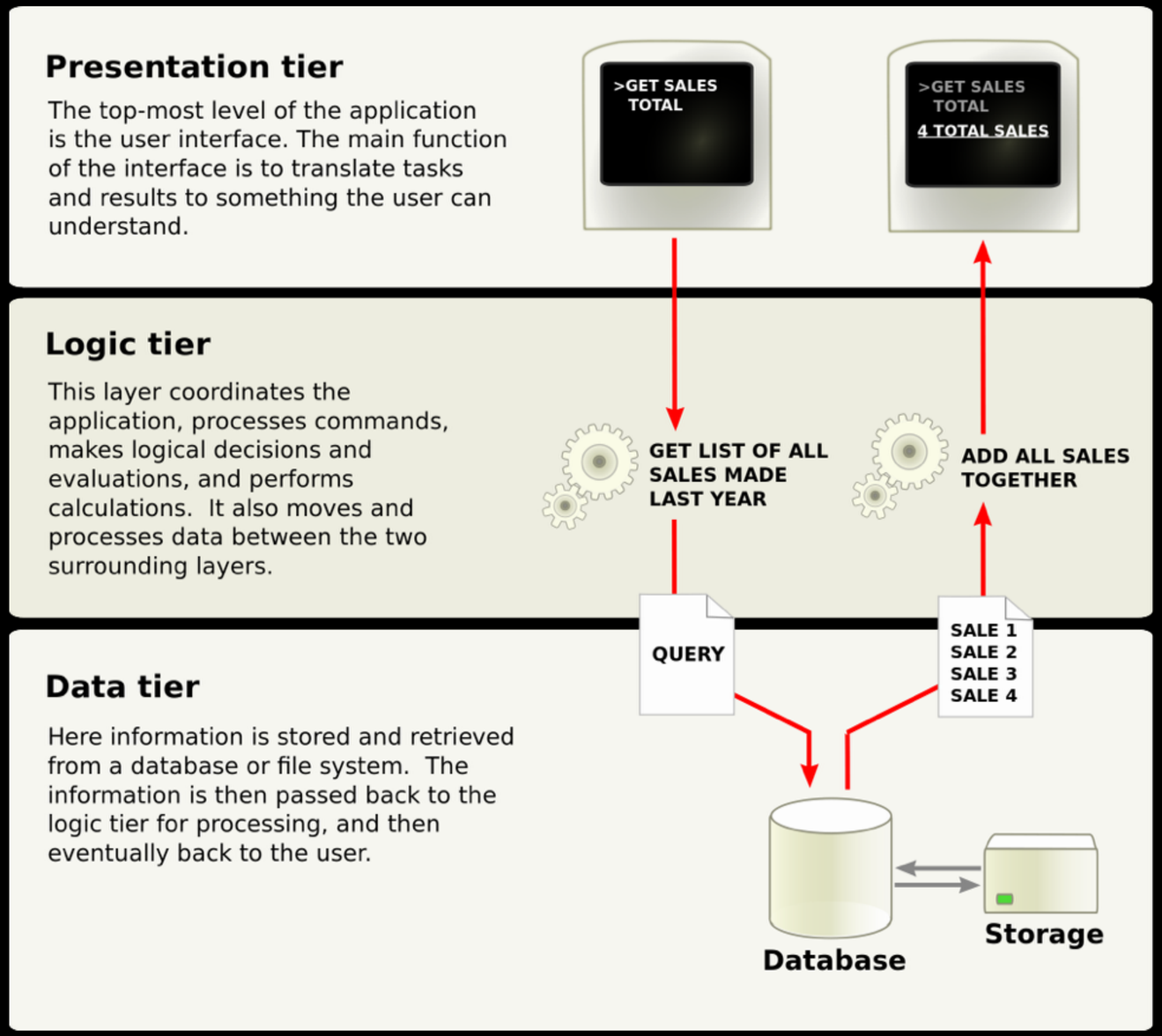
Microservice Architecture on AWS using AWS Lambda and Docker Containers

Danilo Poccia – AWS Technical Evangelist
@danilop

Why **Micro**services?



http://en.wikipedia.org/wiki/Multitier_architecture

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

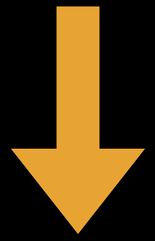
+

Loosely Coupled

+

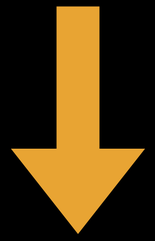
Bounded Context

Microservices



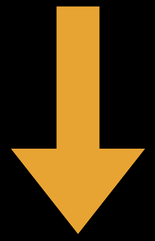
Independent Deployment

Microservices



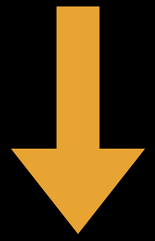
Choose the Right Tool

Microservices



Adopt New Technologies

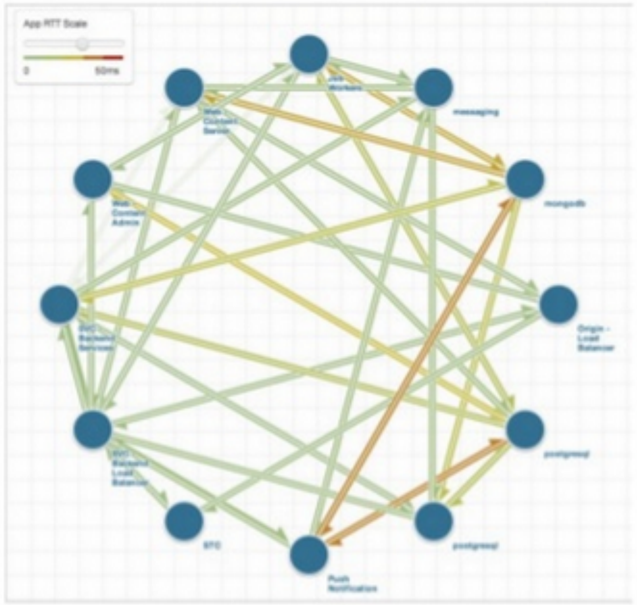
Microservices



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**?

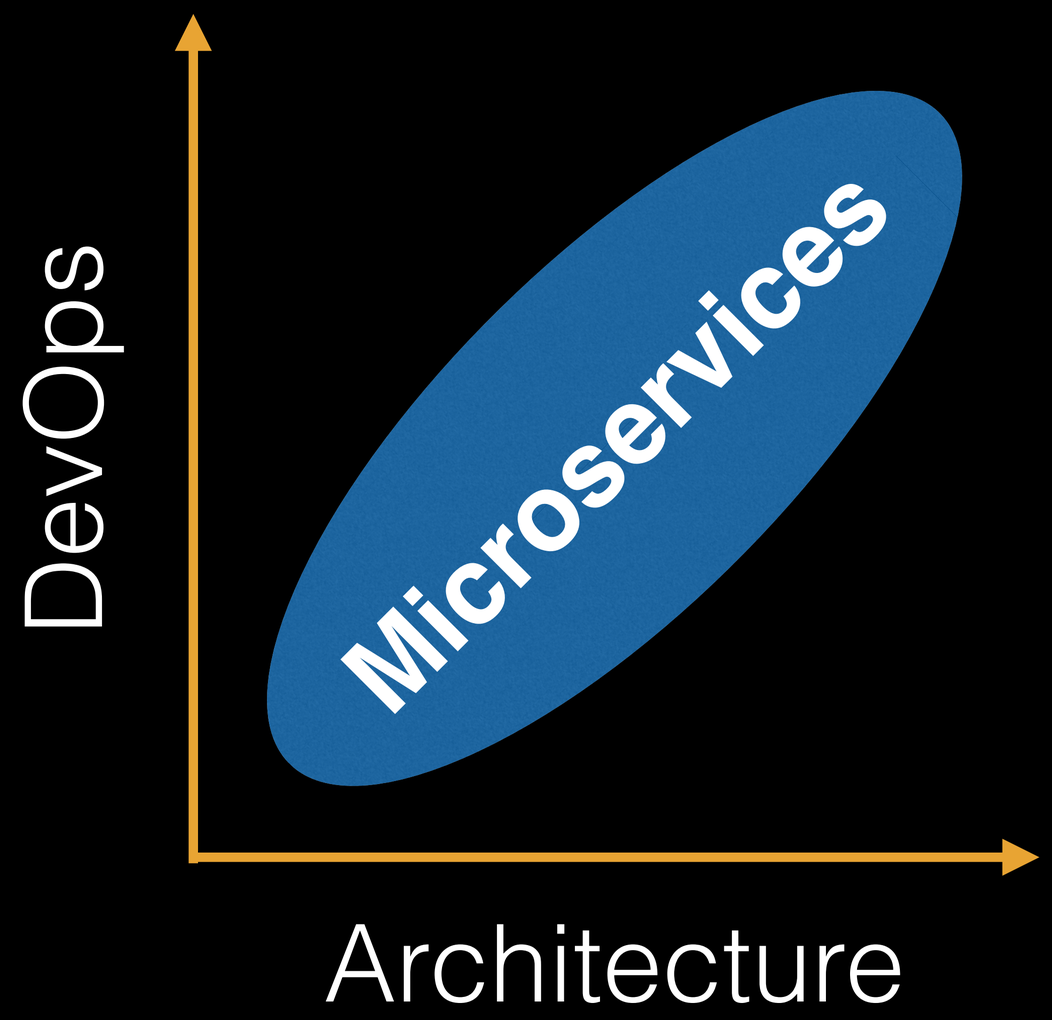
“something that could be rewritten in **two weeks**”

Two Pizza Teams

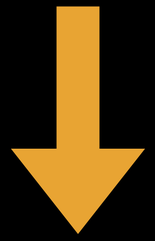
Conway's law

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

Melvin Conway, 1968


DevOps
Microservices
Architecture

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

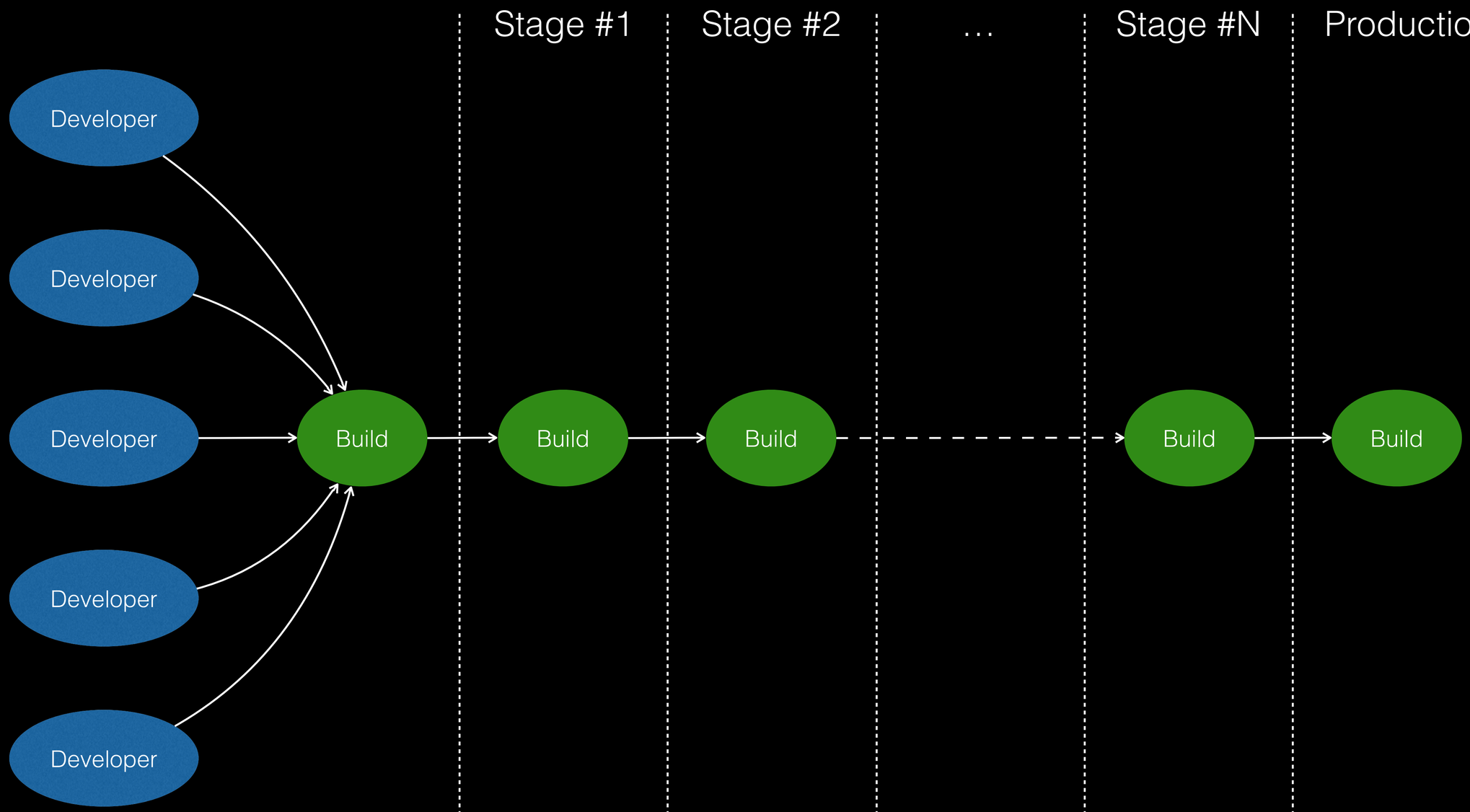
Discovery

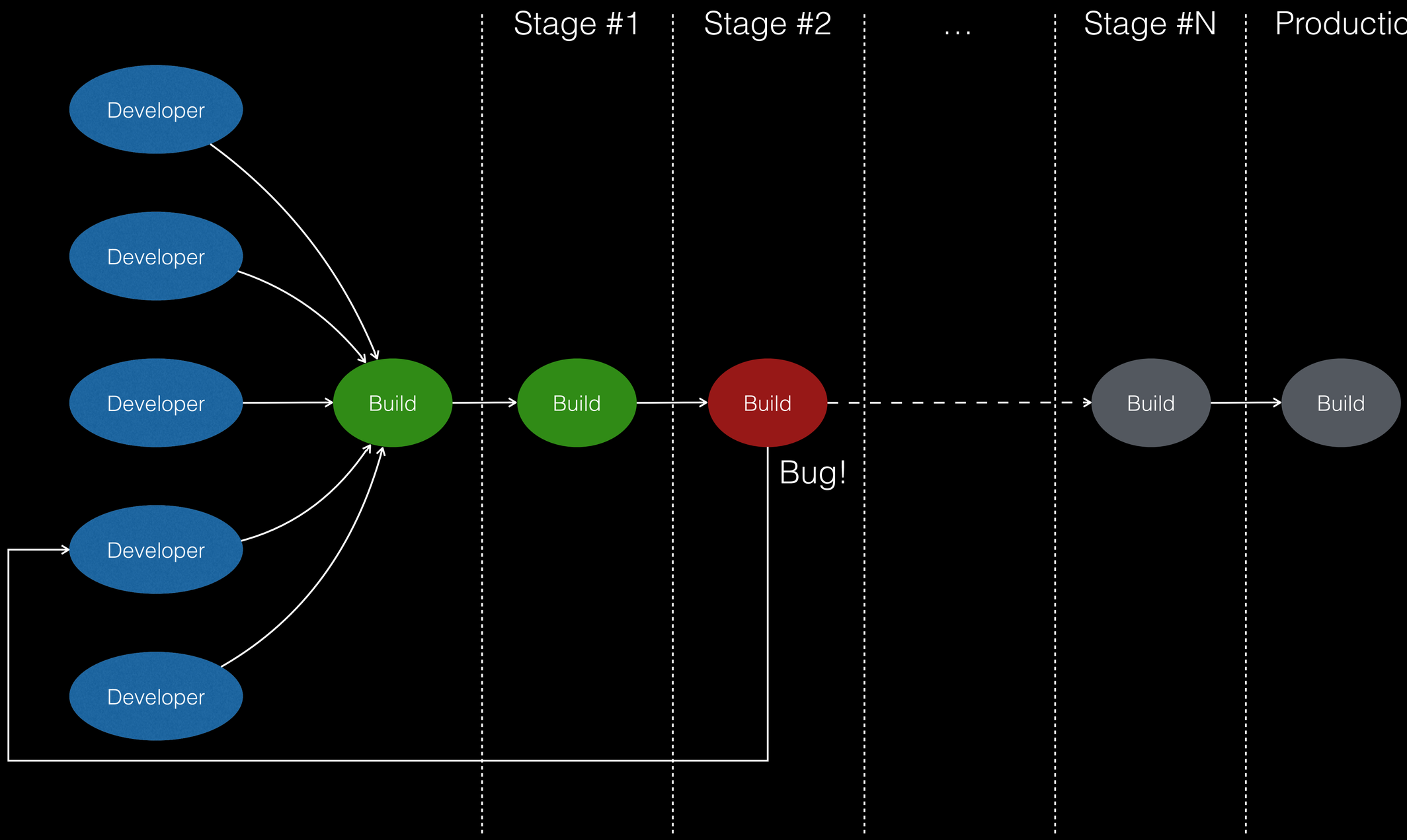
Self-Describing System

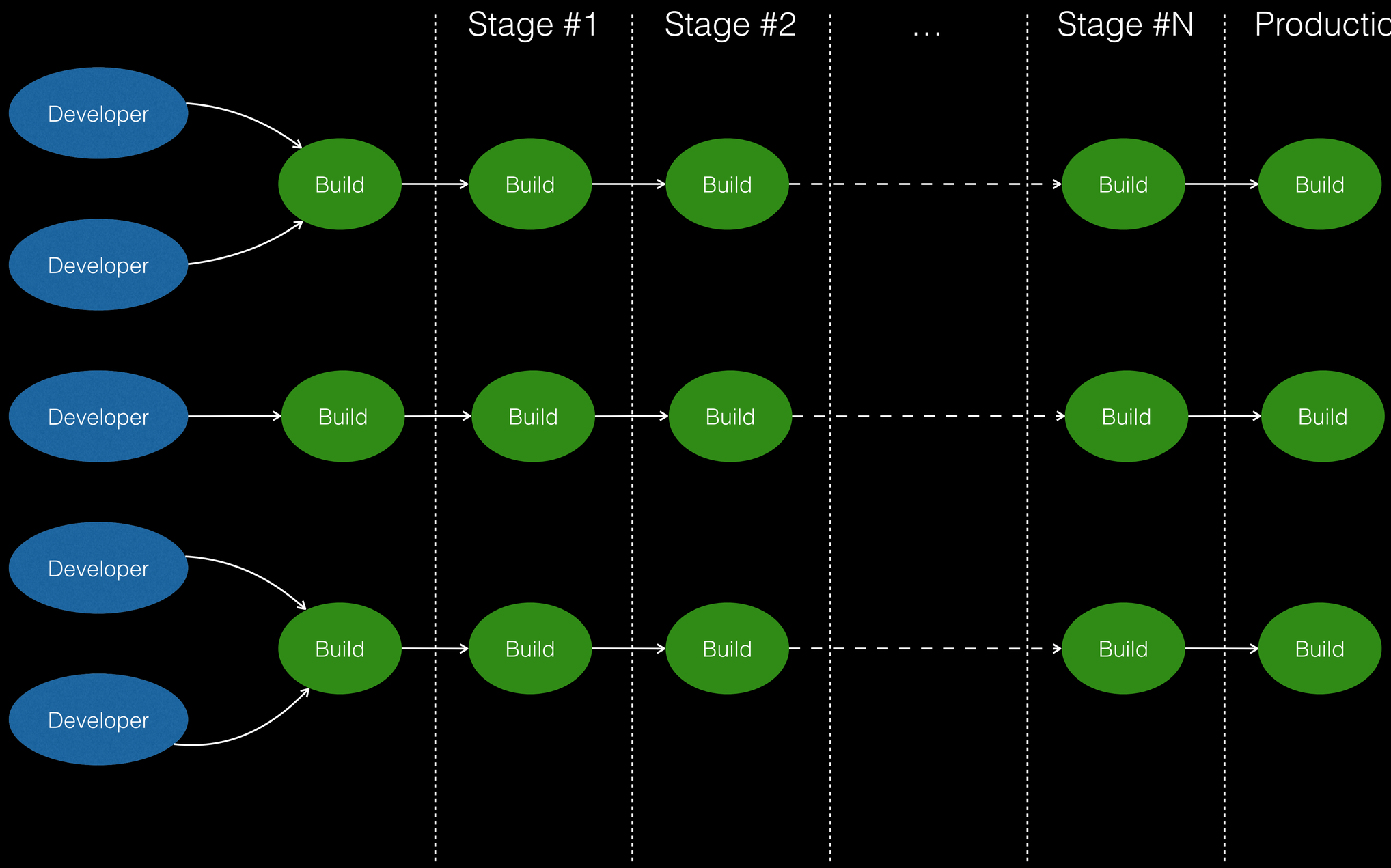
Amazon Route 53
(DNS)

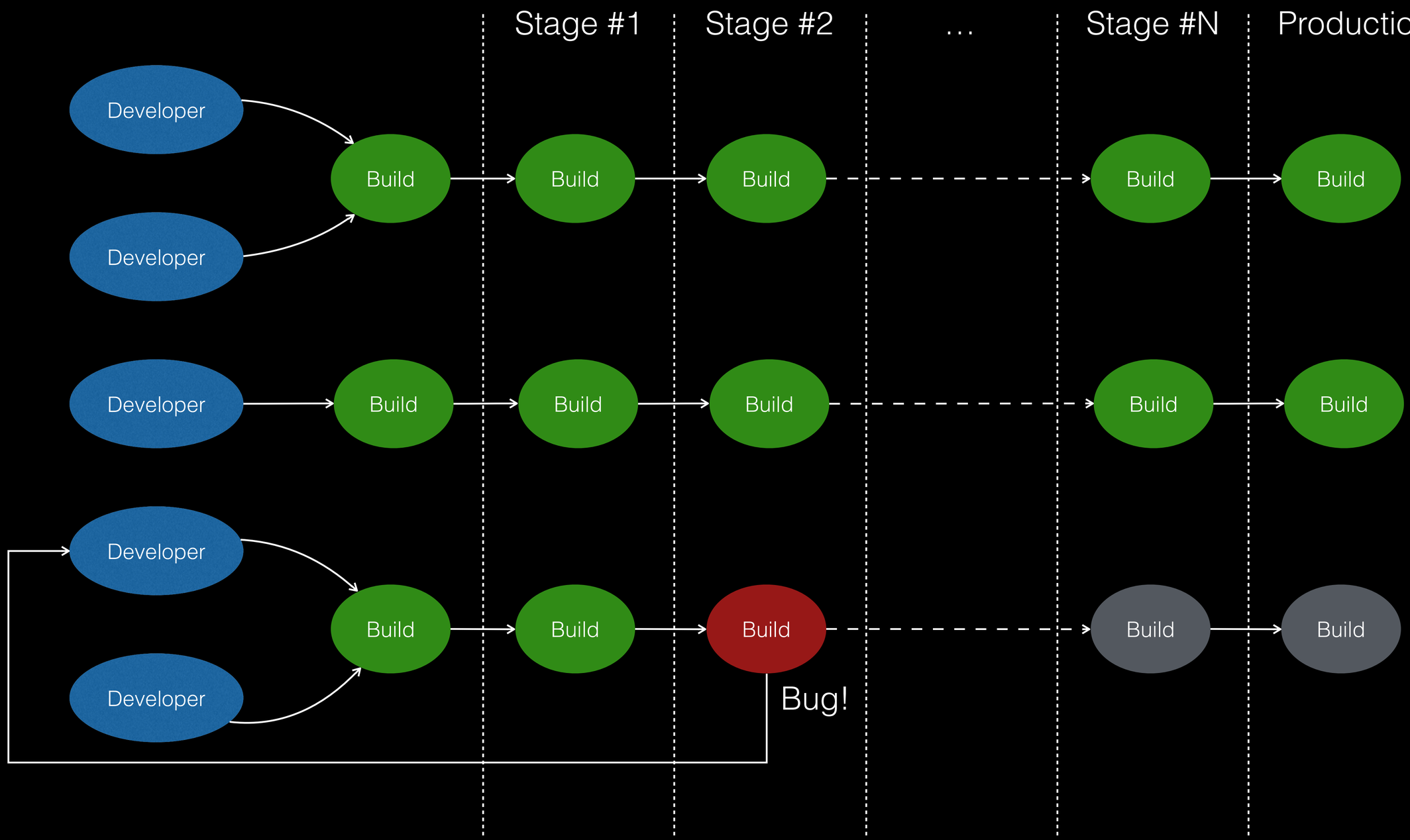
AWS Resource
Tagging

Deployment Pipeline

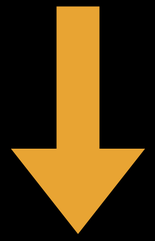

Stage #1
Stage #2
…
Stage #N
Production
Developer
Developer
Developer
Developer
Developer
Build
Build
Build
Build
Build


Stage #1
Stage #2
…
Stage #N
Production
Developer
Developer
Developer
Developer
Developer
Build
Build
Build
Build
Build
Bug!


Stage #1
Stage #2
…
Stage #N
Production
Developer
Developer
Developer
Developer
Developer
Developer
Build
Build
Build
Build
Build
Build
Build
Build
Build
Build
Build
Build
Build
Build
Build

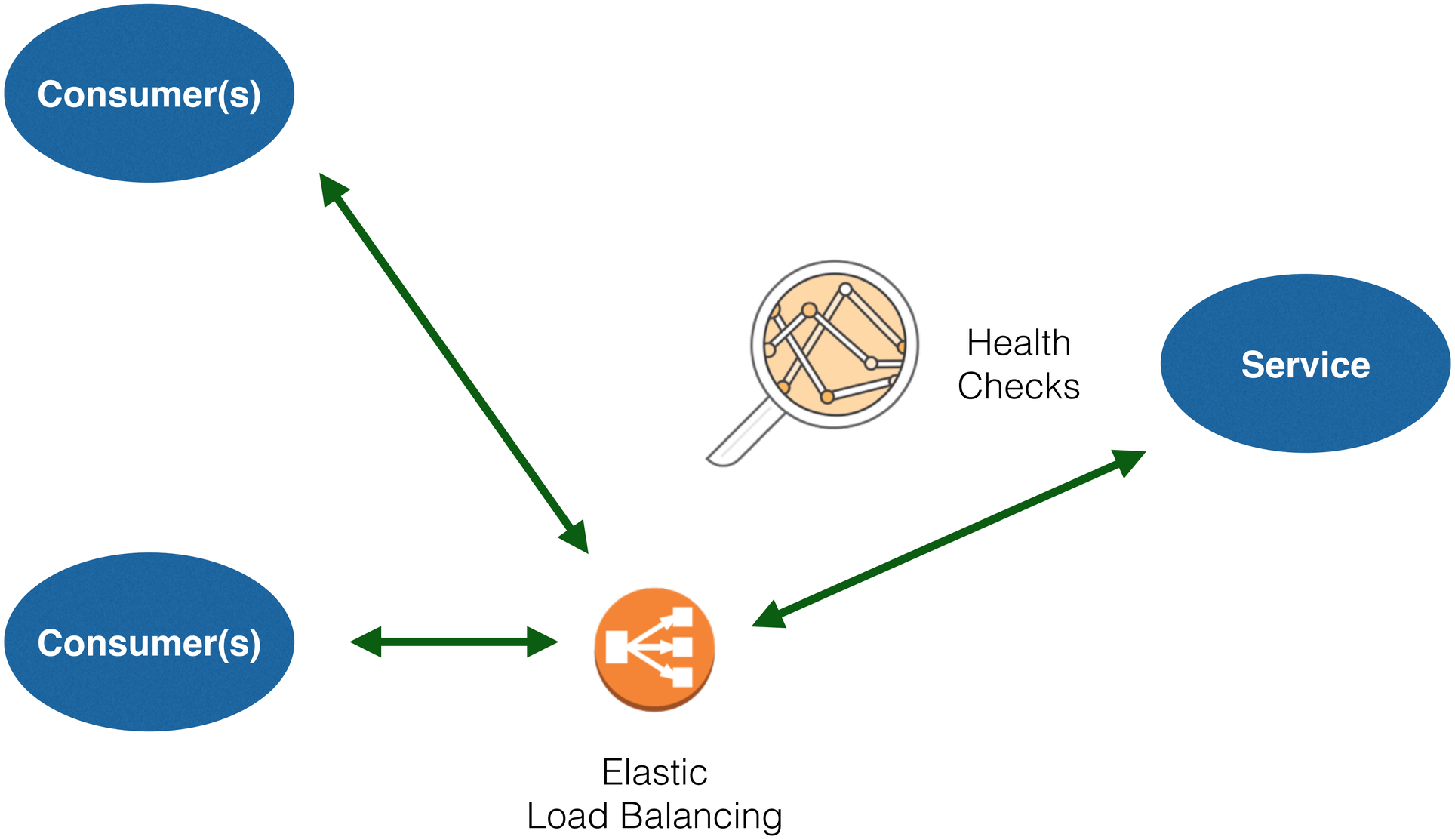

Stage #1
Stage #2
…
Stage #N
Production
Developer
Developer
Build
Build
Build
Build
Build
Developer
Developer
Build
Build
Build
Build
Build
Developer
Developer
Build
Build
Build
Build
Build
Bug!

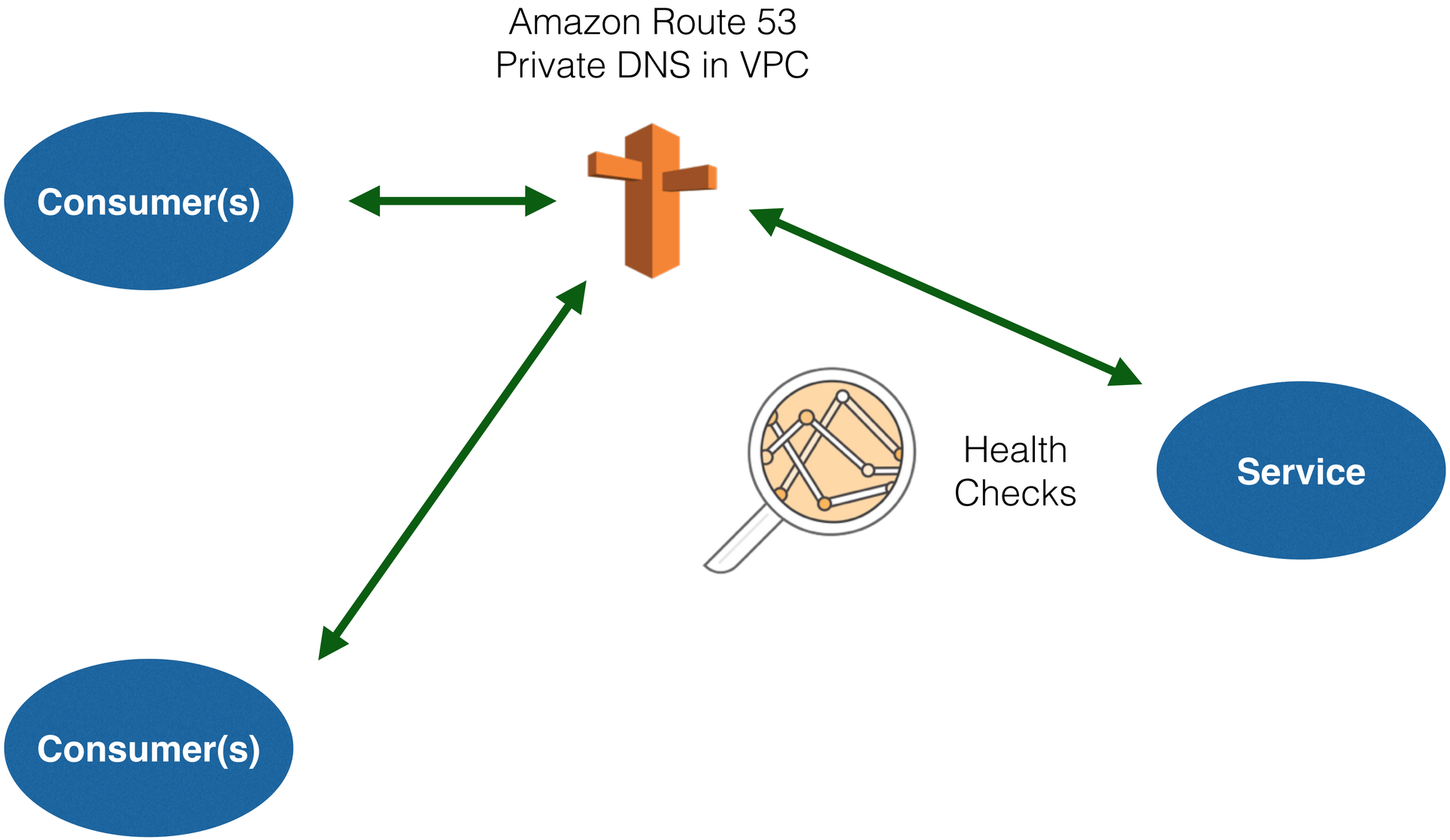
Design for Failure

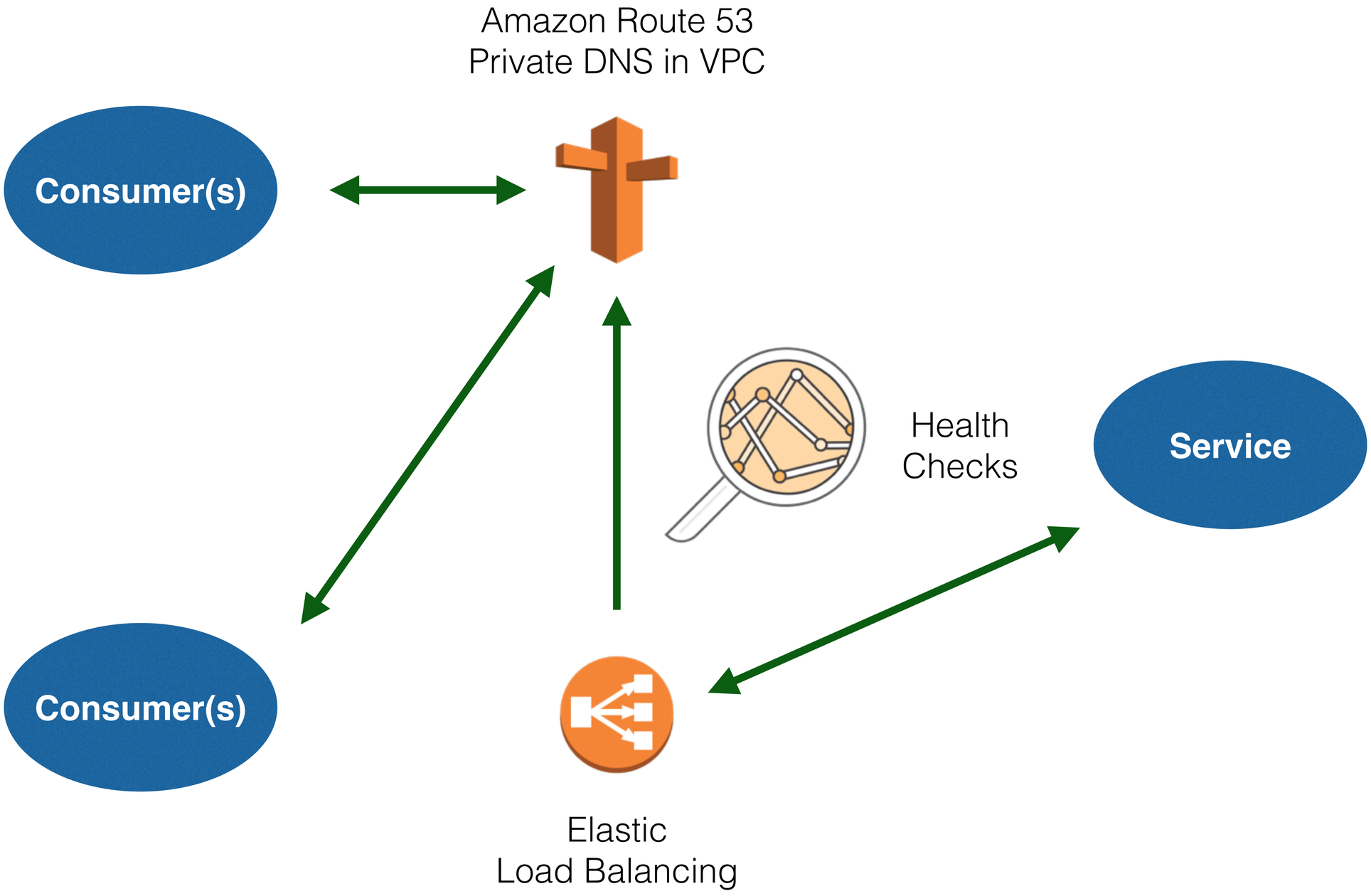


Degrade Functionality

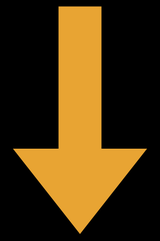
Circuit Breakers


Consumer(s)
Consumer(s)
Health
Checks
Service
Elastic
Load Balancing


Amazon Route 53
Private DNS in VPC
Consumer(s)
Consumer(s)
Health
Checks
Service

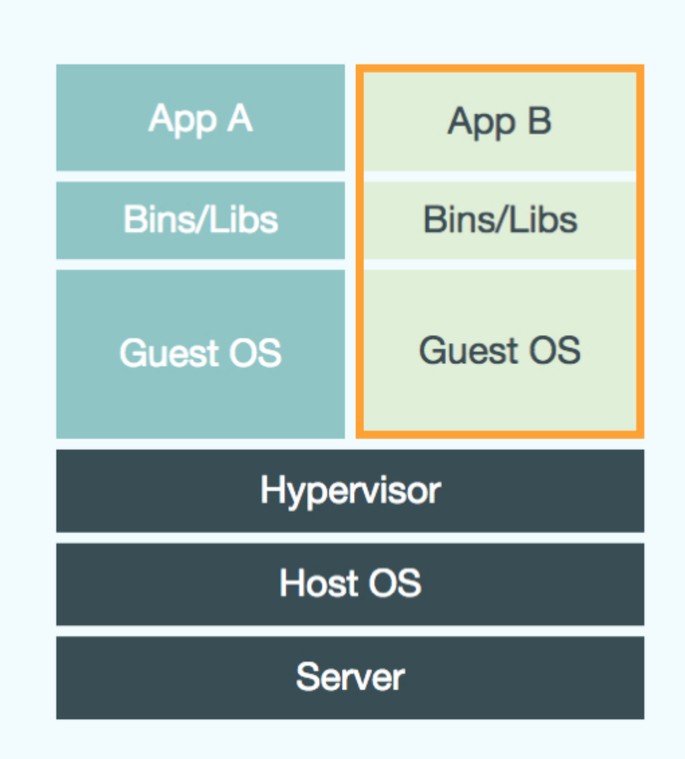

Amazon Route 53
Private DNS in VPC
Consumer(s)
Consumer(s)
Health
Checks
Service
Elastic
Load Balancing

Disposable Infrastructure

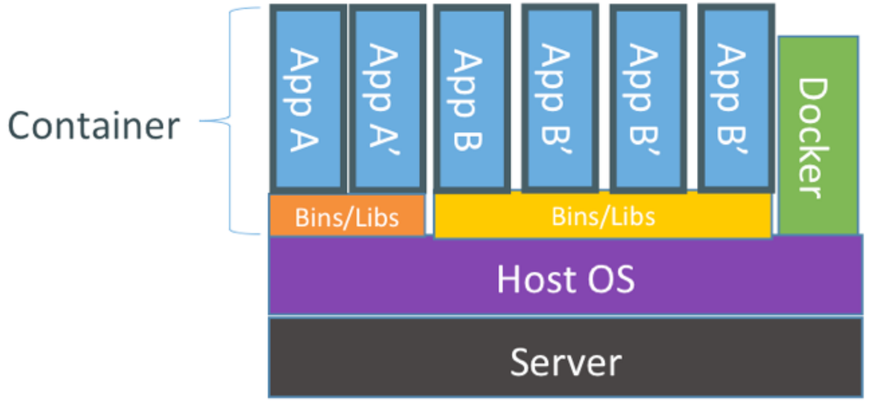


Immutable Deployments

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

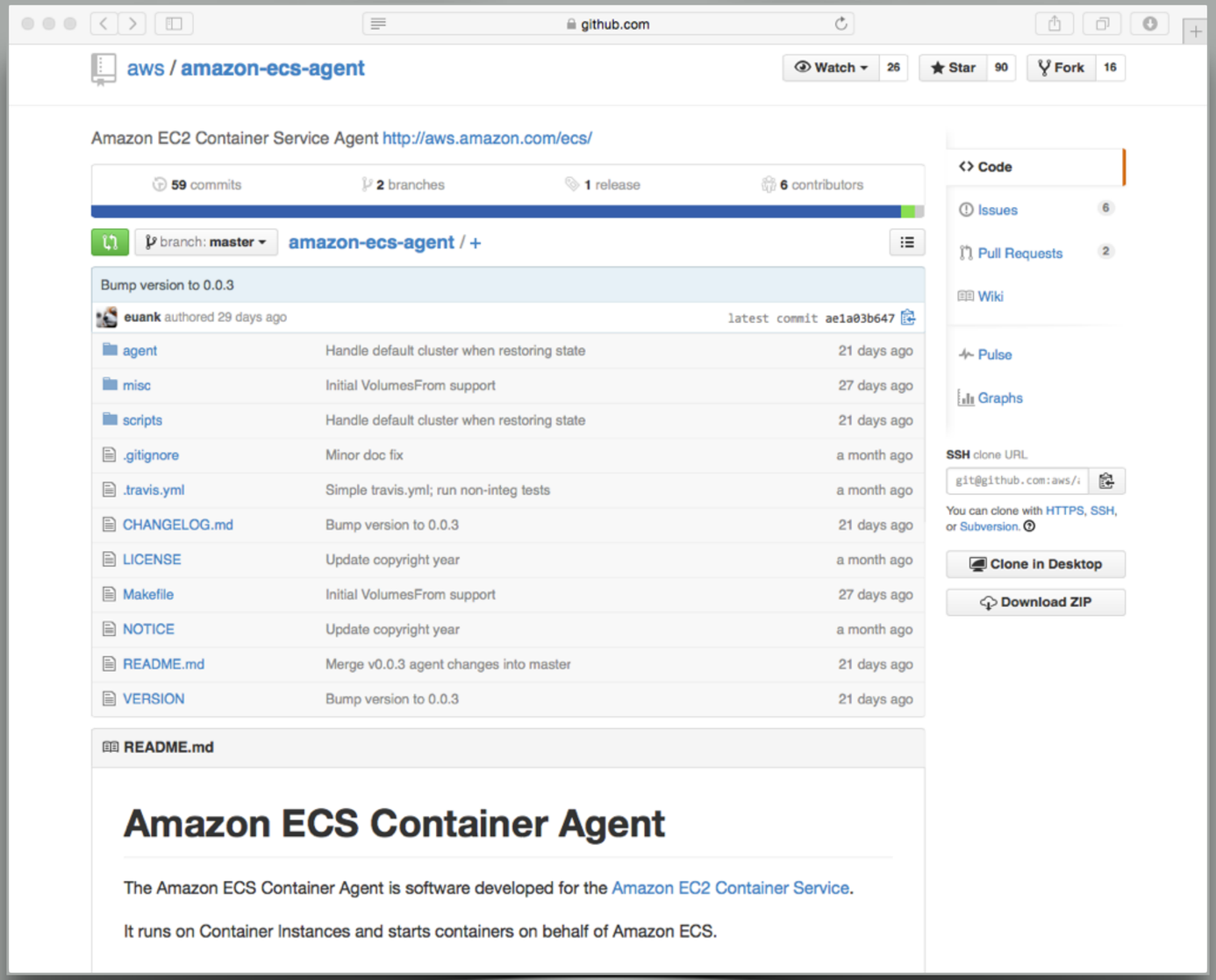
Docker daemon

Amazon ECS agent

Amazon EC2 Container Service

Key Components

Container Instances
Clusters
Tasks
Task Definitions



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

Unit of work

Grouping of related Containers

Run on Container Instances

Amazon EC2 Container Service

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

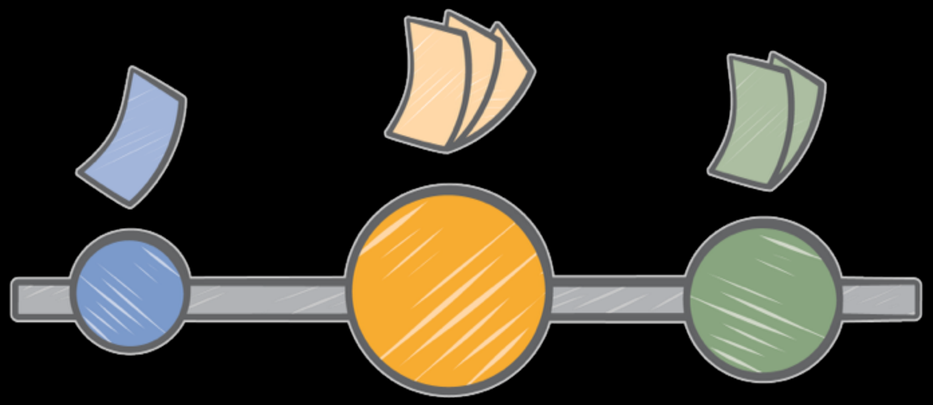
Customer uploads code,
AWS Lambda handles:

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

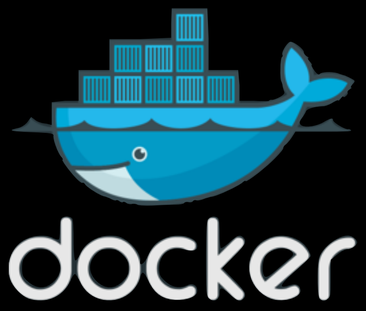
Weeks

On-Premises

Minutes

Amazon EC2

Seconds



Containers

Milliseconds

AWS Lambda

On AWS

Netflix Architecture

Asgard, Amimator, Hystrix,
Cassandra, JVM, Docker, …

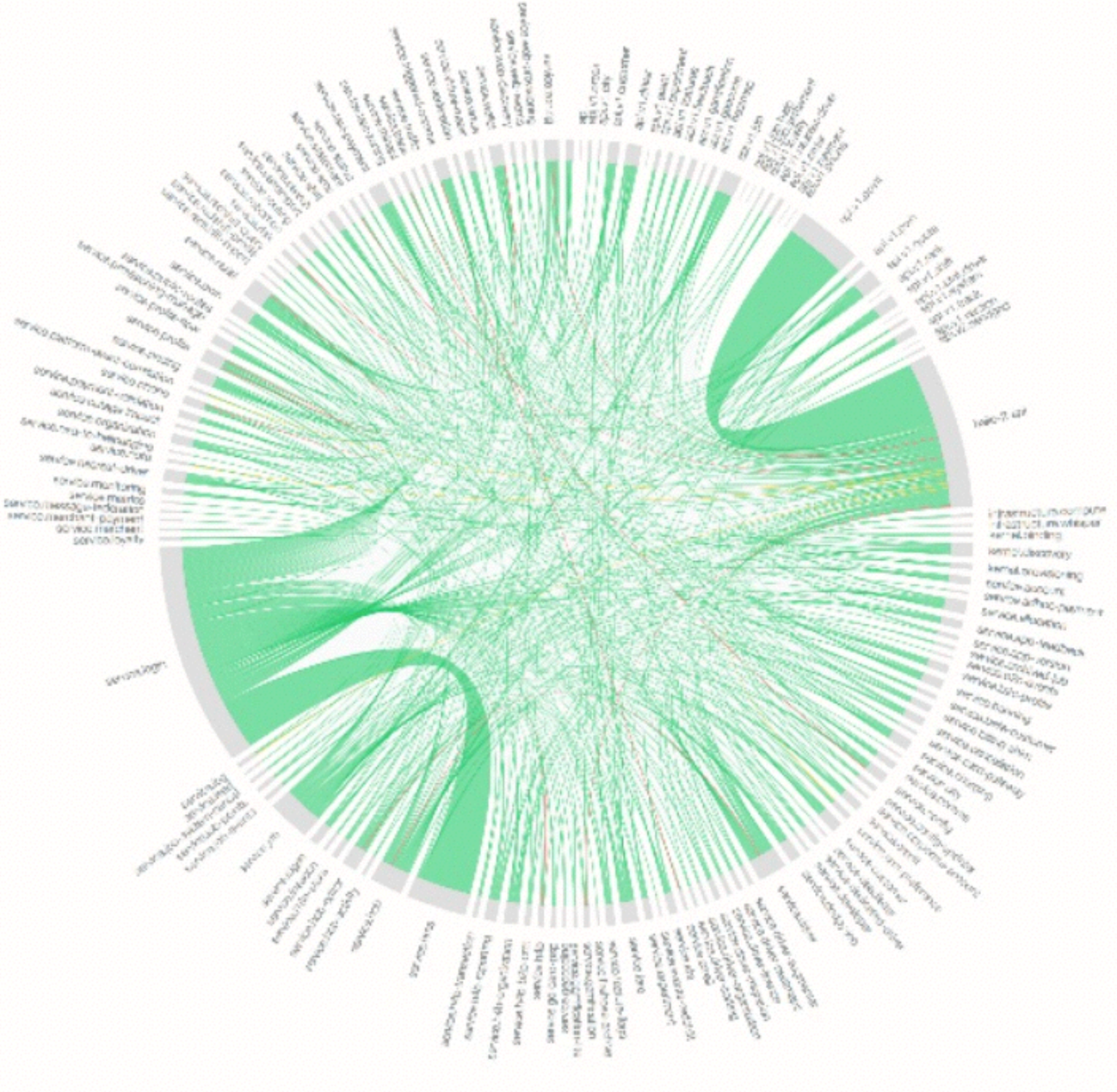

amazon
web services™
CHAOS
MONKEY

On AWS

Hailo Architecture

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

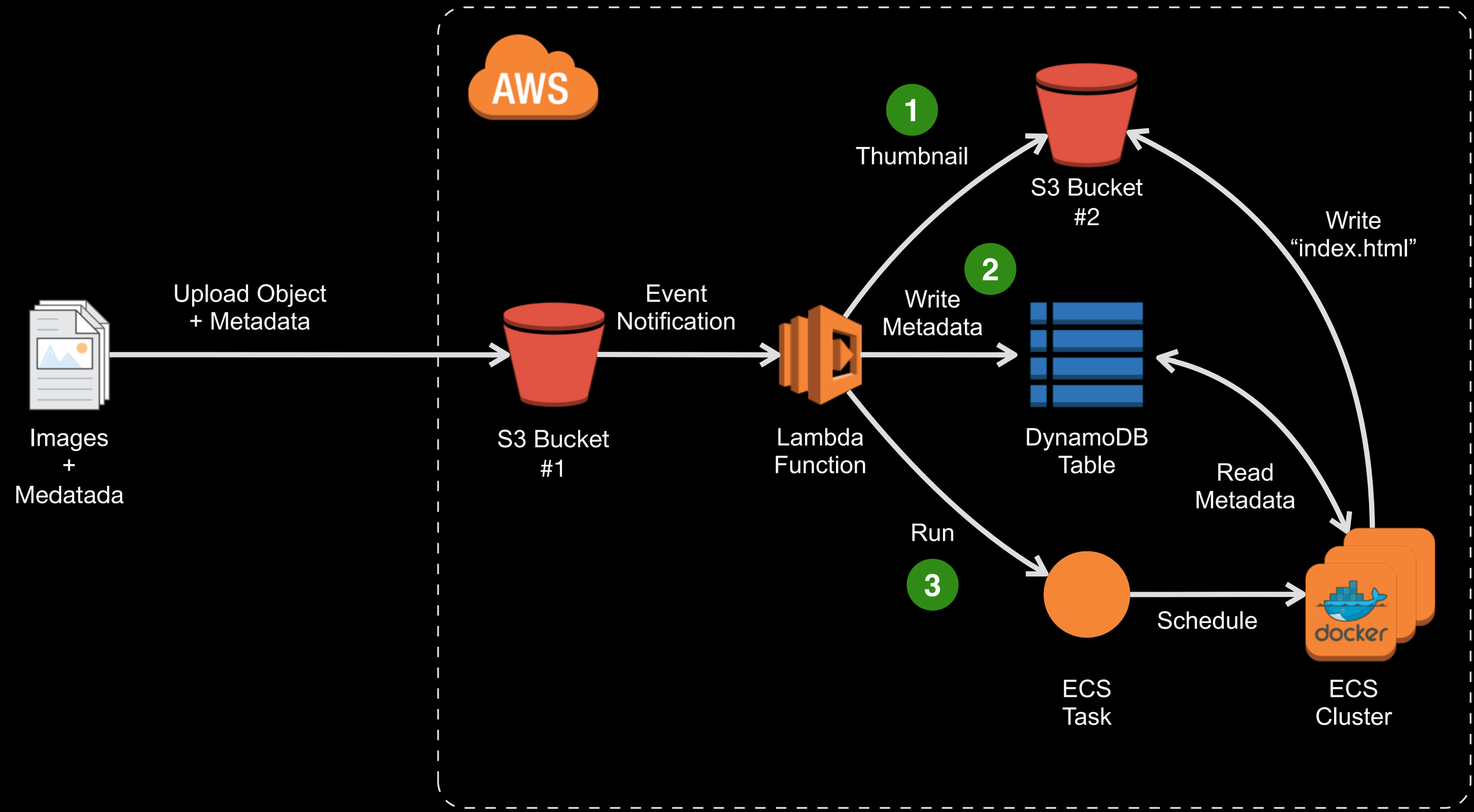
HAIL O™



On AWS

Demo Architecture

Content Management System Prototype


AWS
1
Thumbnail
S3 Bucket
#2
Write
“index.html”
2
Write
Metadata
Upload Object
+ Metadata
Event
Notification
Images
+
Medatada
S3 Bucket
#1
Lambda
Function
DynamoDB
Table
Read
Metadata
Run
3
Schedule
docker
ECS
Task
ECS
Cluster

<demo>

. . .

</demo>

Iterate

Continuous Improvement

Kaizen

@danilop