
AWS
re:Invent

CON222

How Disney+ deploys globally with Amazon ECS

Zach Bintliff

Engineering Manager, Developer Productivity Engineering

Disney+

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Who we are

How we use Amazon ECS

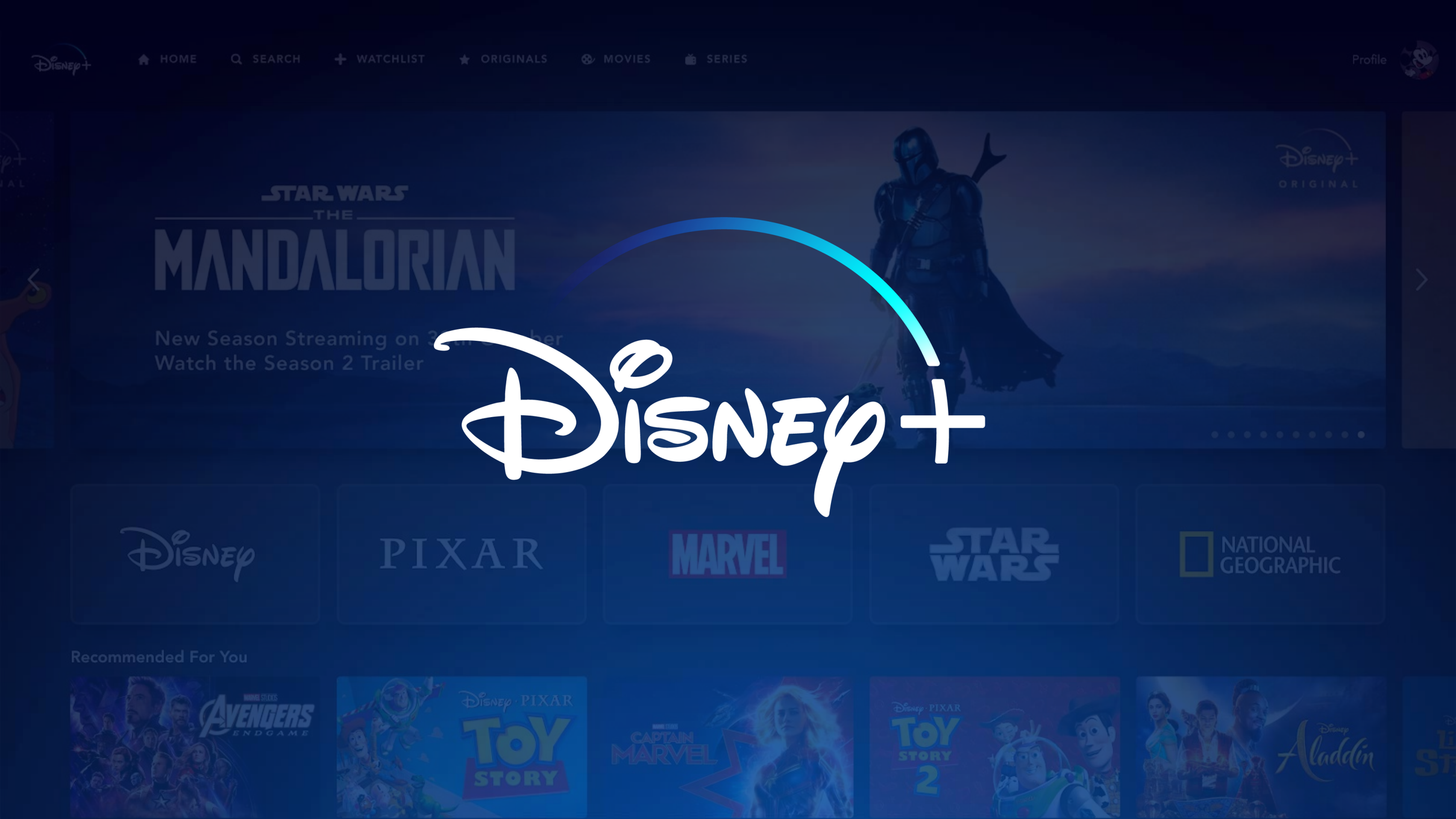
Goals of the platform

Features the platform allows

How developers interact with the platform



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.


HOME
SEARCH
WATCHLIST
ORIGINALS
MOVIES
SERIES
Profile
STAR WARS
THE
MANDALORIAN
New Season Streaming on
Watch the Season 2 Trailer
Disney+
ORIGINAL
Disney+
Disney
PIXAR
MARVEL
STAR WARS
NATIONAL GEOGRAPHIC
Recommended For You
AVENGERS
ENDGAME
Disney · PIXAR
TOY STORY
CAPTAIN MARVEL
Disney · PIXAR
TOY STORY 2
Disney
Aladdin

Disney+ developer productivity

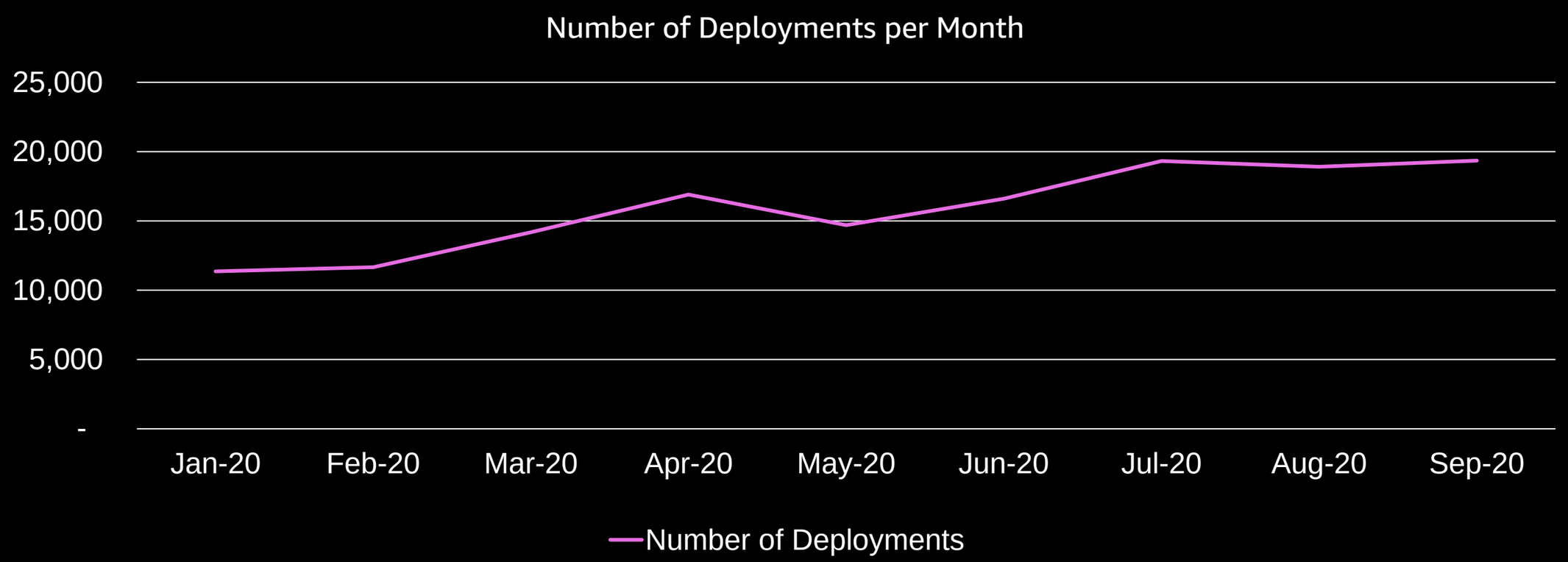
- Goal: To help teams operate in the cloud
- Provide tools to easily manage build, deploy, and compute
- Focusing on deployment

Some numbers

Over 330k
deployments since 2017

Over 700
unique applications deployed

~40k
ECS tasks deployed across our clusters



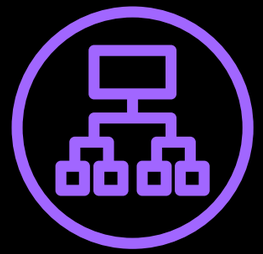
Choosing Amazon ECS

- Robust scheduler
- Many clusters across many regions
- Little overhead, other than managing compute layer



Building a platform – “Glue code as a service”

- More to deploying production workloads than the containers
- Connecting ECS, load balancing, target groups, logging, and DNS
- Availability requirements
- Best practices



Application load balancer



Amazon CloudWatch



Amazon Route 53

Technologies used

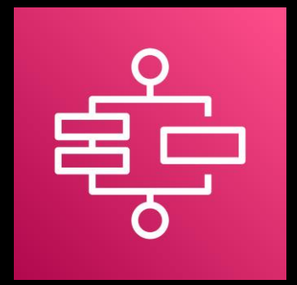
- Amazon ECS
- AWS CloudFormation to safely create resources
- AWS Step Functions as orchestration layer



Amazon Elastic Container Service



AWS CloudFormation



AWS Step Functions

Goals of the platform

- Make it easy to deploy
- Make it difficult to make a mistake
- Default best practices



Goals: Make it easy to deploy

- Require as few as parameters as possible

Worker Application

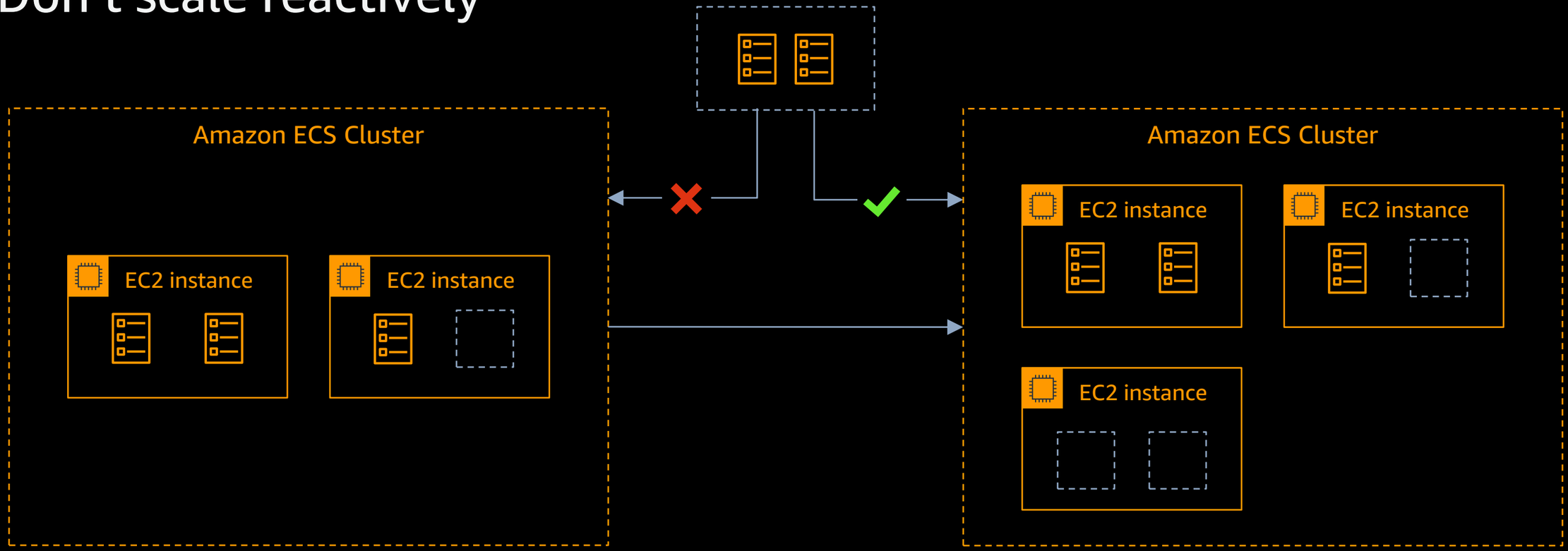
```
---
applicationName: ""
environment: ""
namespace: ""
region: ""
service:
  desiredCount: 0
task:
  containerDefinitions:
  - cpu: 0
    essential: false
    image: ""
    memory: 0
    name: ""
```

Load Balanced Application

```
---
applicationName: ""
environment: ""
loadBalancerConfigNameList:
- ""
namespace: ""
region: ""
service:
  desiredCount: 0
task:
  containerDefinitions:
  - cpu: 0
    essential: false
    image: ""
    memory: 0
    name: ""
    portMappings:
    - containerPort: 0
      hostPort: 0
      protocol: ""
```

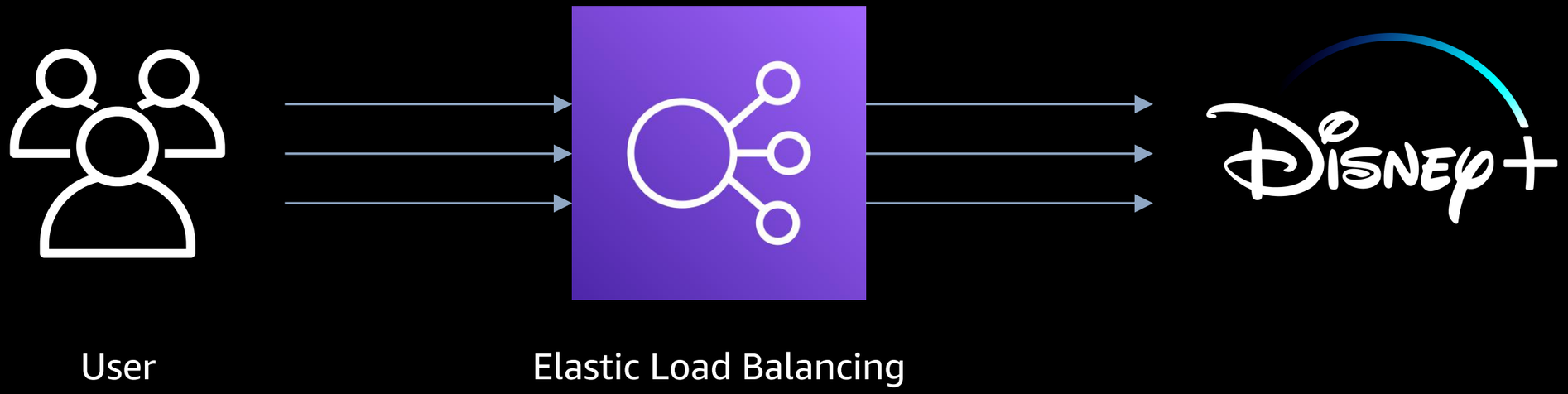
Goals: Make it easy to deploy

- Preemptive cluster scaling
- Don’t scale reactively



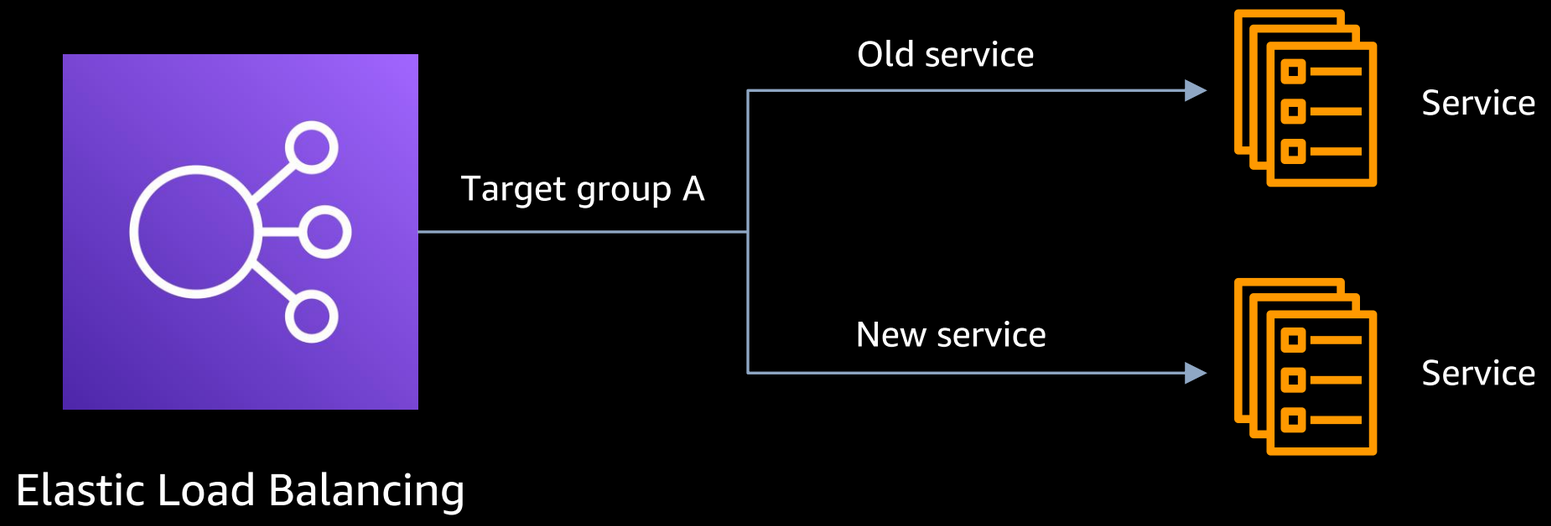
Goals: Make it difficult to make a mistake

- Load balancer preservation



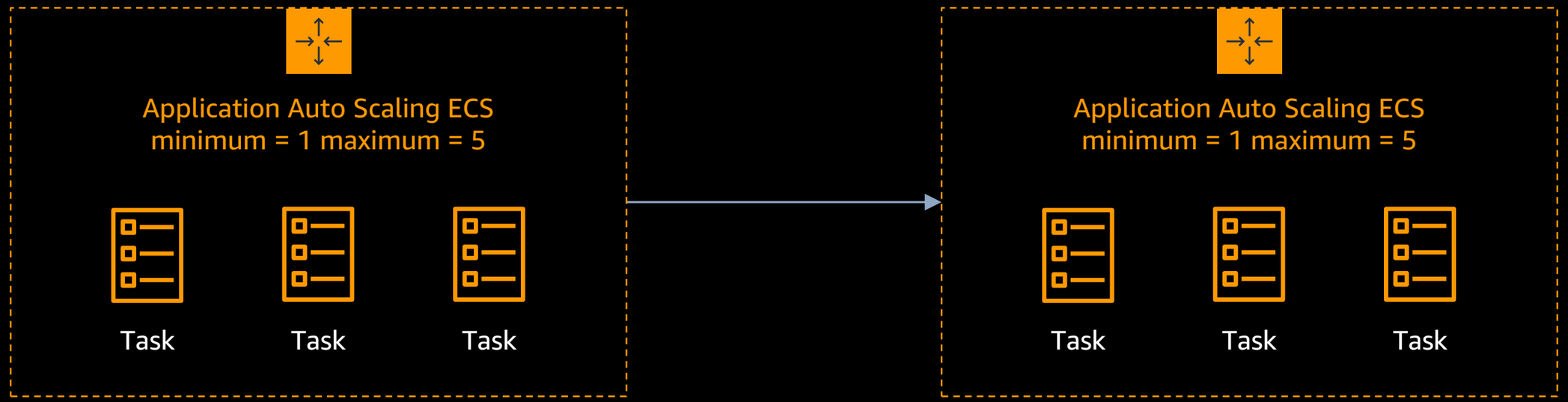
Goals: Make it difficult to make a mistake

- Blue/green deployment



Goals: Make it difficult to make a mistake

- Safely determine desired count of scaling applications



Goals: Make it difficult to make a mistake

- Robust health checking
- Three levels of health checking
  - Task count
  - Container checks
  - ALB checks
- Times vary across different deployments

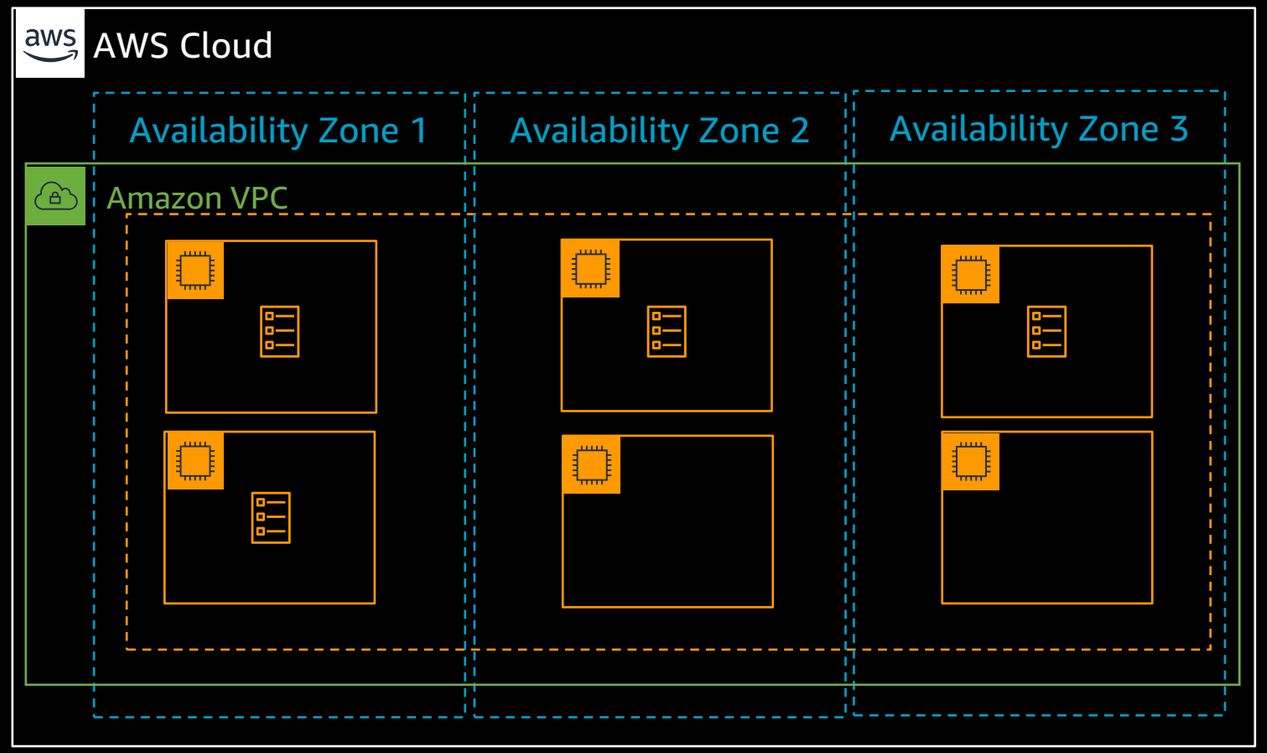
Goals: Make it difficult to make a mistake

- Detailed events through entire pipeline
- When things do fail, explain why

```
- status: DEPLOY-FAIL
   message: 'Task Stop Reason: Task failed to start; Container demo
   Status Reason: CannotPullContainerError: The named manifest is not known to the.
                  registry.;'
```

Goals: Default best practices

- Placement constraints
- Availability zone spread
- Instance spread



Goals: Default best practices

- Tagging and container labels
- Default logging configuration



Amazon CloudWatch



Logs

Goals: Default best practices

- Deterministic DNS name
- Create and manage a Route 53 ALIAS Record
- Allows app teams to better control their endpoints



Amazon Route 53

How developers interact with platform

- Provide a CLI to be used locally and on Jenkins
- Use own central authentication
- No AWS context or AWS credentials needed
- Provide a declarative YAML that represents your deployment

```
$ deployer create-deployment --spec-file file://definition.yaml
```

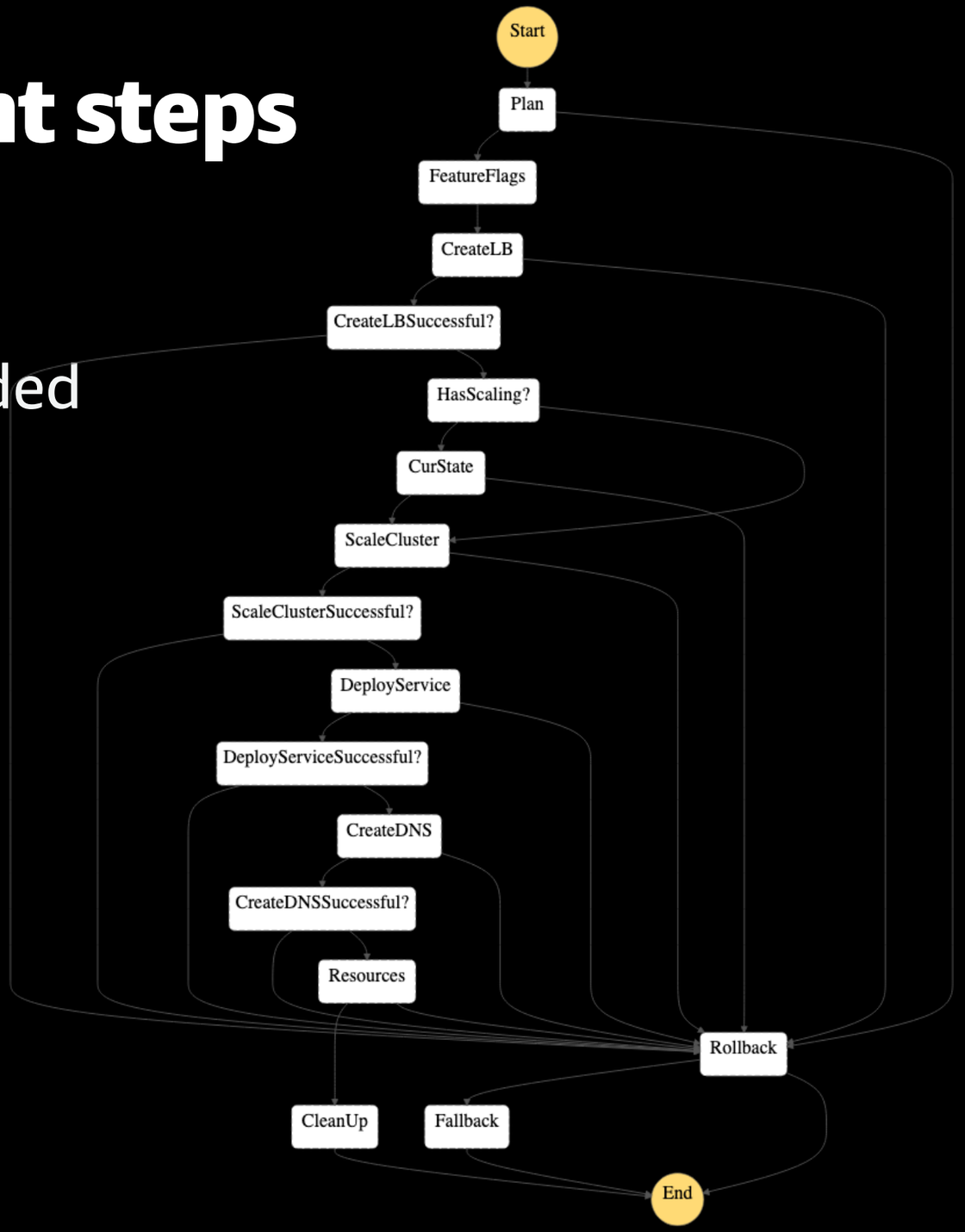
Other features

- Canary deployments using weighted target groups
- Ability to diff two different deployments to track changes
- Deployment lineage for easier rollback

```
$ deployer deployment-diff --id 123
    portMappings:
      -
~             containerPort: 2020 => 8080
              hostPort: 0
~             protocol: TCP => UDP
```

Recap: Platform deployment steps

- Map to specific account/cluster
- Creates an ALB and target group if needed
- Scales cluster if needed
- Creates the ECS Service
- Run health checks
- Update DNS
- Tear down old service
- Always rollback



Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.