
AWS
re:Invent

CON306

Optimizing your applications for scale on Amazon ECS

Adam Keller

Developer Advocate, Containers

AWS

AWS re:Invent

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

Agenda

Build for scale

Scaling the process

Scaling operations

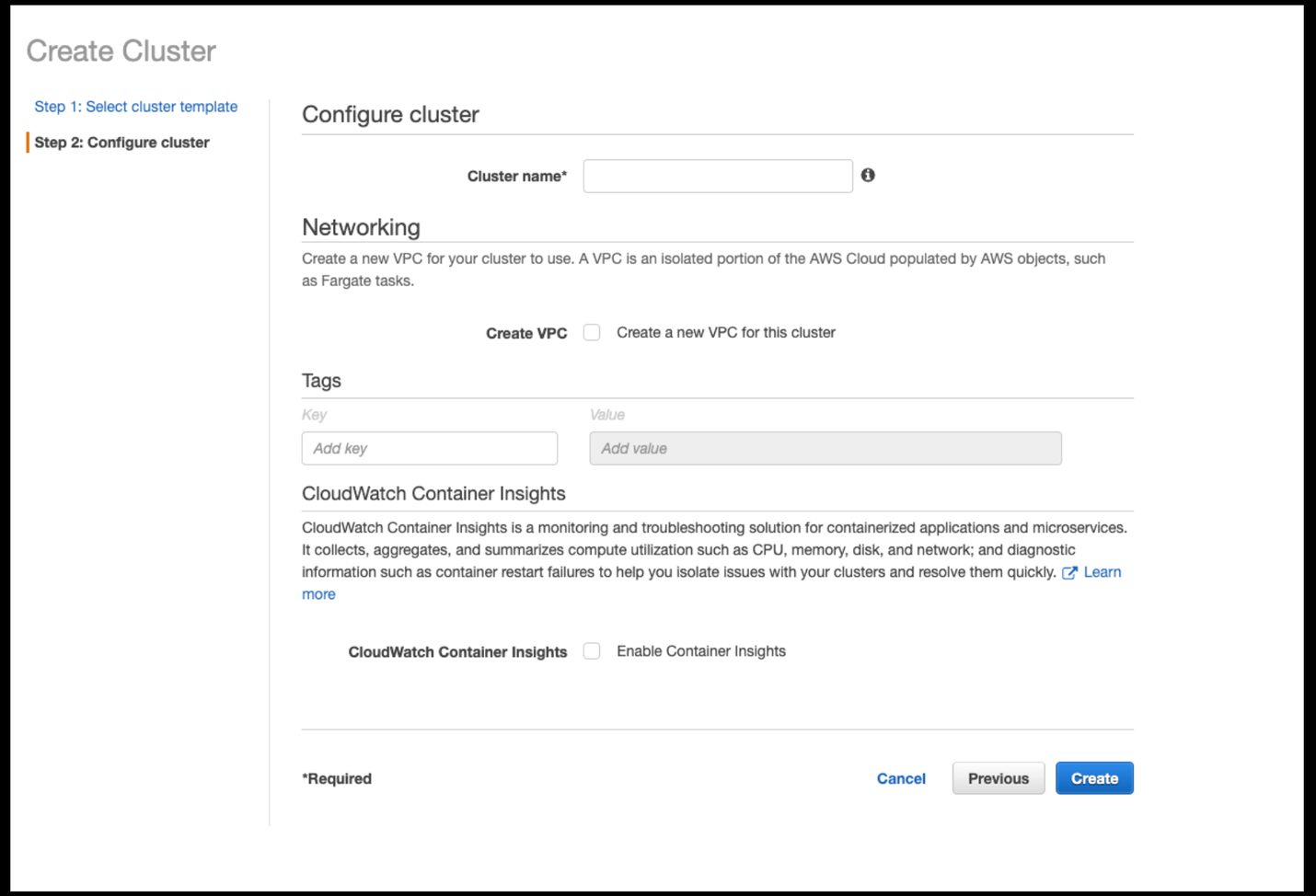


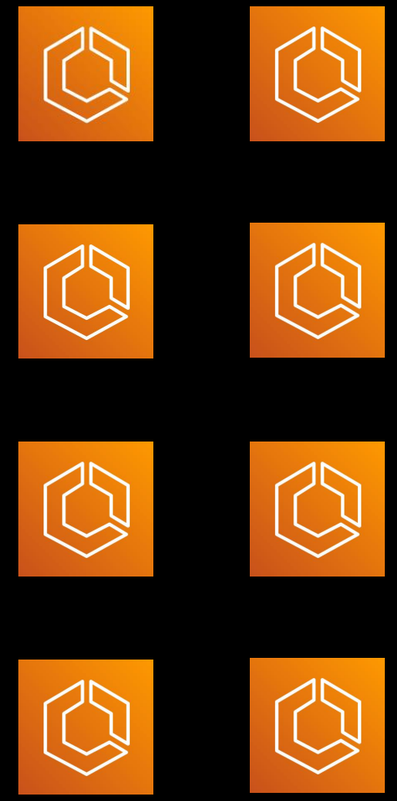
AWS re:Invent

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

We all start somewhere







Build for scale



AWS re:Invent

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

Automate from the beginning



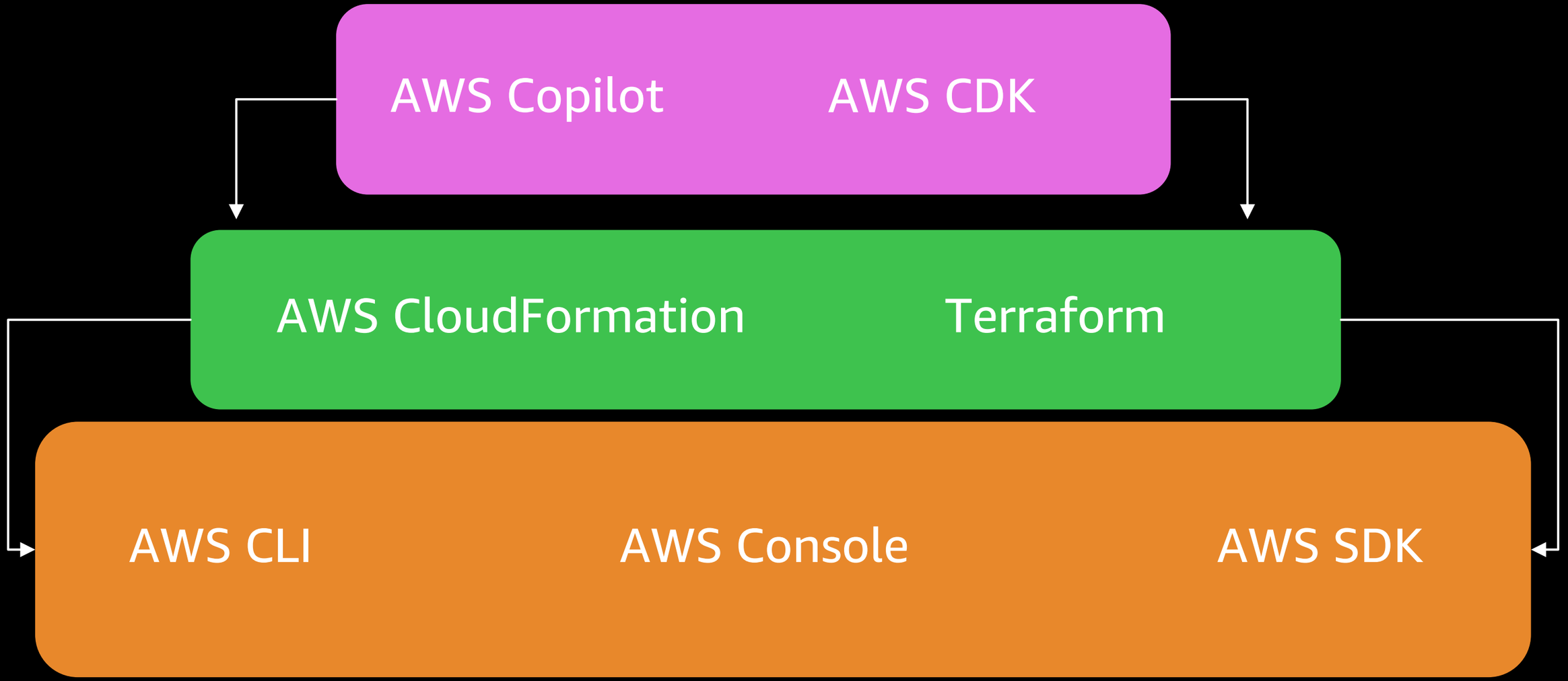
Build to your team’s strengths

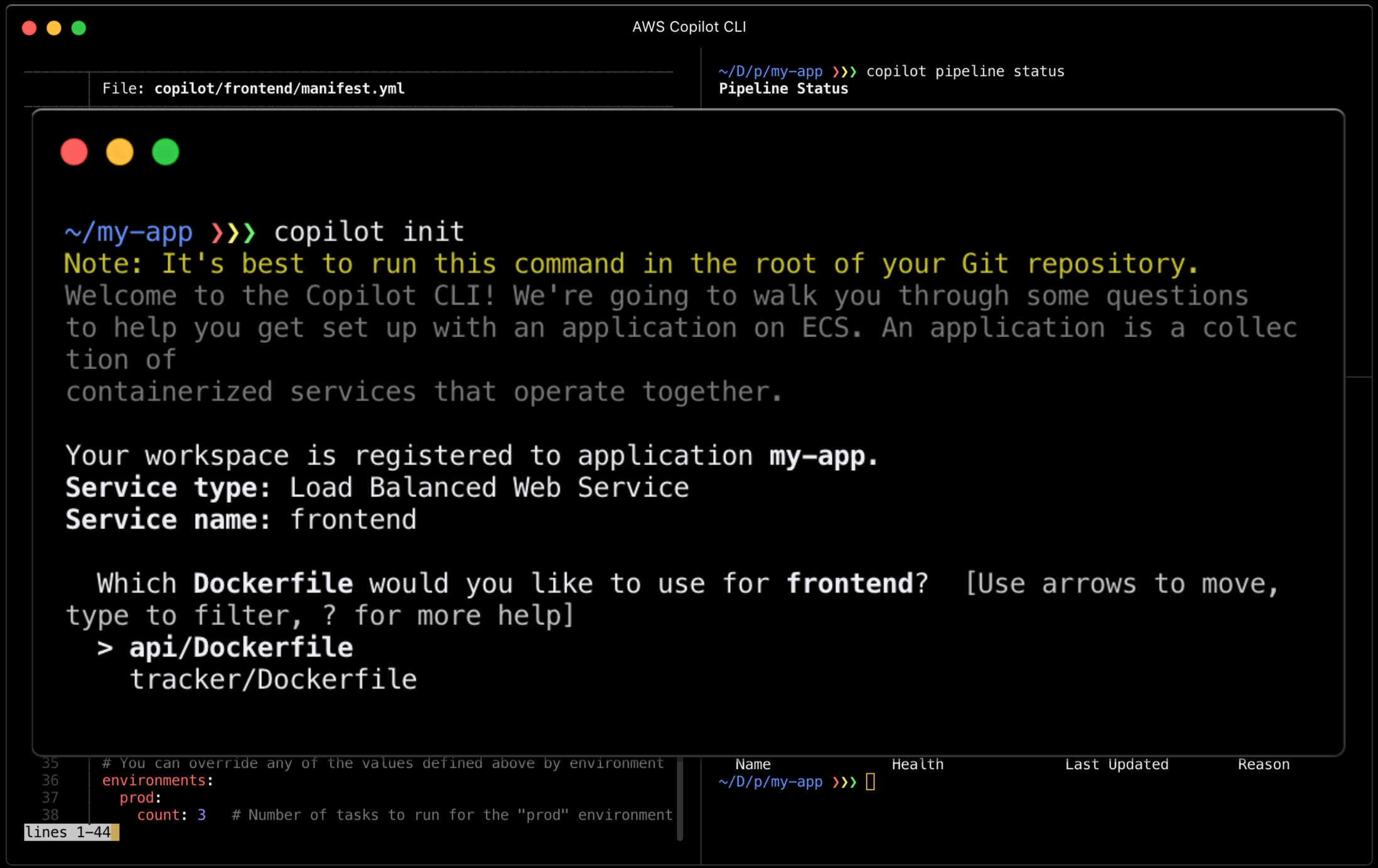
Manage your environments as code

Use opinionated tools

GitOps

Choosing the right tool for the job




AWS Copilot CLI
File: copilot/frontend/manifest.yml
~/D/p/my-app ❯❯❯ copilot pipeline status
Pipeline Status
~/my-app ❯❯❯ copilot init
Note: It's best to run this command in the root of your Git repository.
Welcome to the Copilot CLI! We're going to walk you through some questions
to help you get set up with an application on ECS. An application is a collec
tion of
containerized services that operate together.
Your workspace is registered to application my-app.
Service type: Load Balanced Web Service
Service name: frontend
Which Dockerfile would you like to use for frontend? [Use arrows to move,
type to filter, ? for more help]
> api/Dockerfile
tracker/Dockerfile
35 # You can override any of the values defined above by environment
36 environments:
37 prod:
38 count: 3 # Number of tasks to run for the "prod" environment
lines 1-44
Name Health Last Updated Reason
~/D/p/my-app ❯❯❯

AWS CDK

```python
from aws_cdk import core, aws_ecs_patterns, aws_ecs


class AlbFargateStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        # The code that defines your stack goes here
        self.wordpress_lb_service = aws_ecs_patterns.ApplicationLoadBalancedFargateService(
            self, "WordPressALBService",
            cpu=1,
            memory_limit_mib=512,
            listener_port=80,
            task_image_options=aws_ecs_patterns.ApplicationLoadBalancedTaskImageOptions(
                image=aws_ecs.ContainerImage.from_registry('wordpress:latest'),
                container_port=8080
            )
        )

```

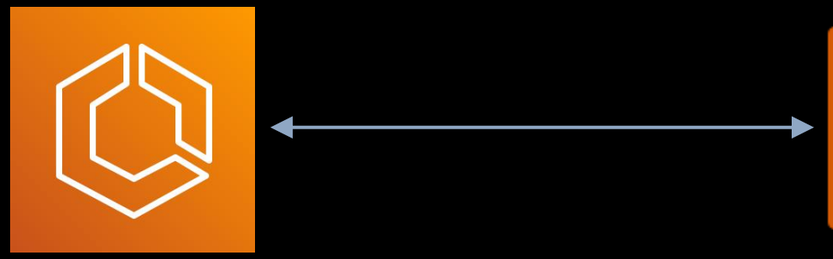
1,000+ lines of CloudFormation

Understanding your application

- Stateful vs. stateless
- Think monitoring / tracing
  - What are the most important metrics?
  - What does scaling look like?
- Health checks



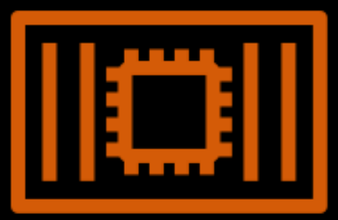
Container health checks



Amazon Elastic Container Service (Amazon ECS)



```
[ "CMD-SHELL", "./health.sh || exit 1" ]
```







Database



Application load balancer



```
/health
```

Scale the process

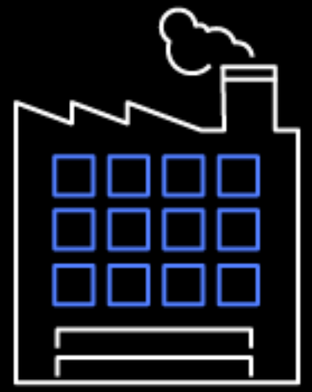


AWS re:Invent

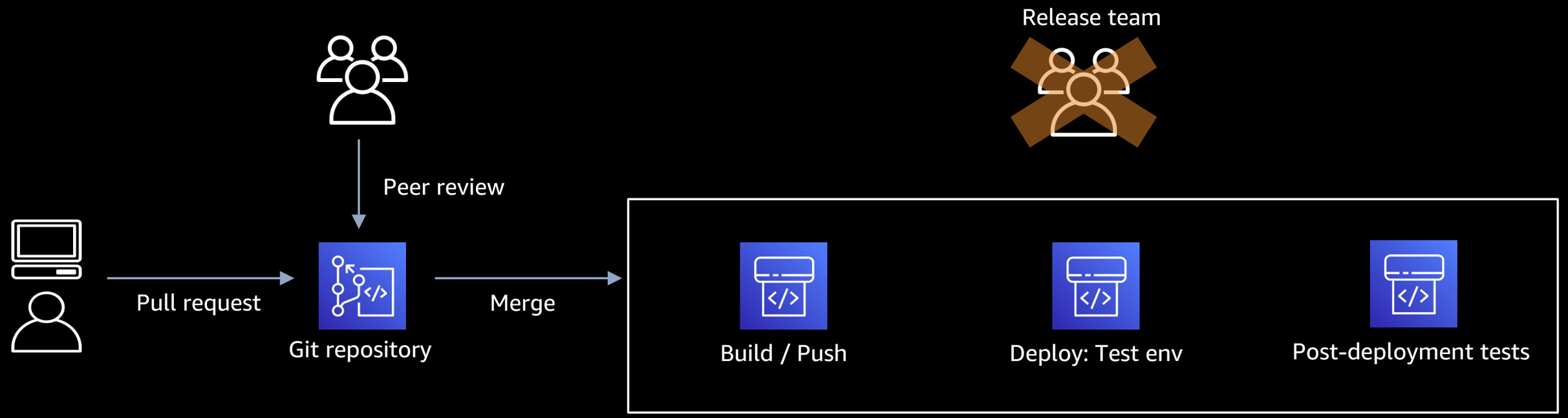
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Modern deployments

- The evolution of deployments
- CI / CD to GitOps
- Monitoring and testing is critical



GitOps



Single source of truth

Change review / deploy via git

Modern deployment strategies

- Manually modifying in the console is not a strategy
- Rolling deployments
- AWS CodeDeploy blue / green, canary, linear

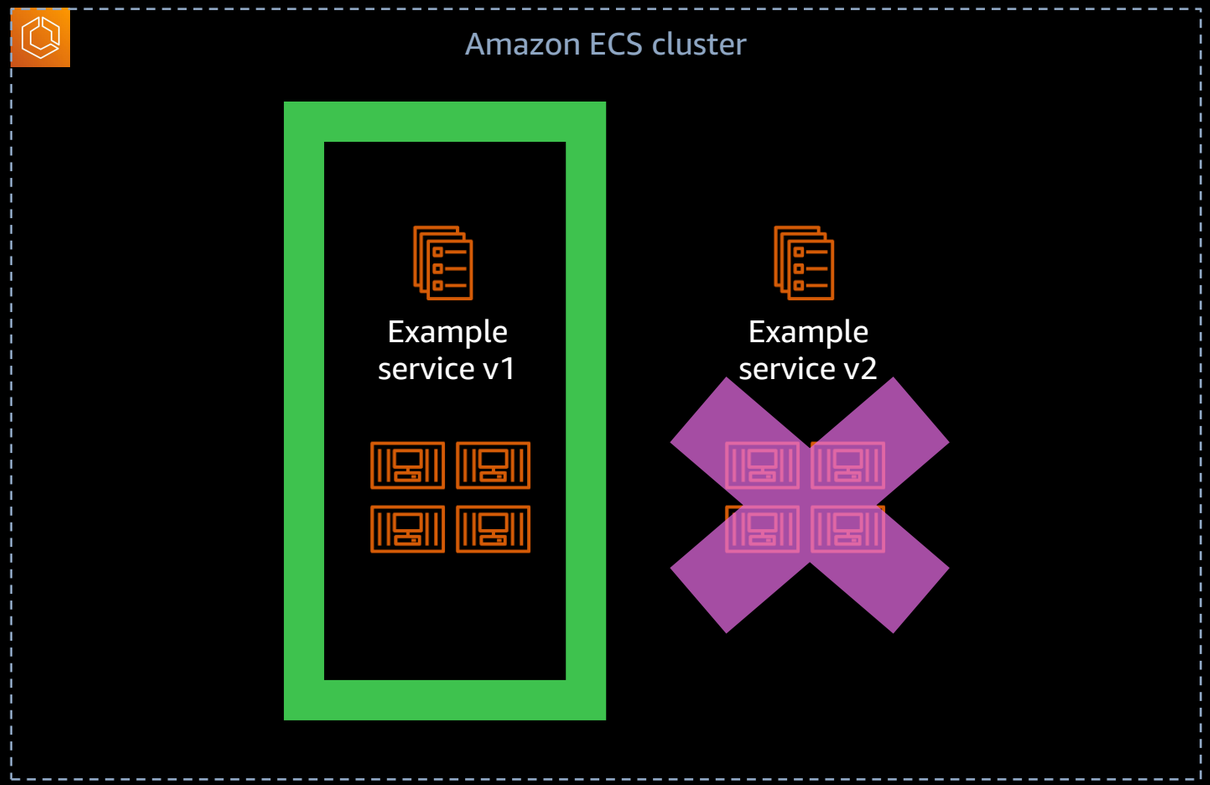
Rolling deployments

Min healthy
100%

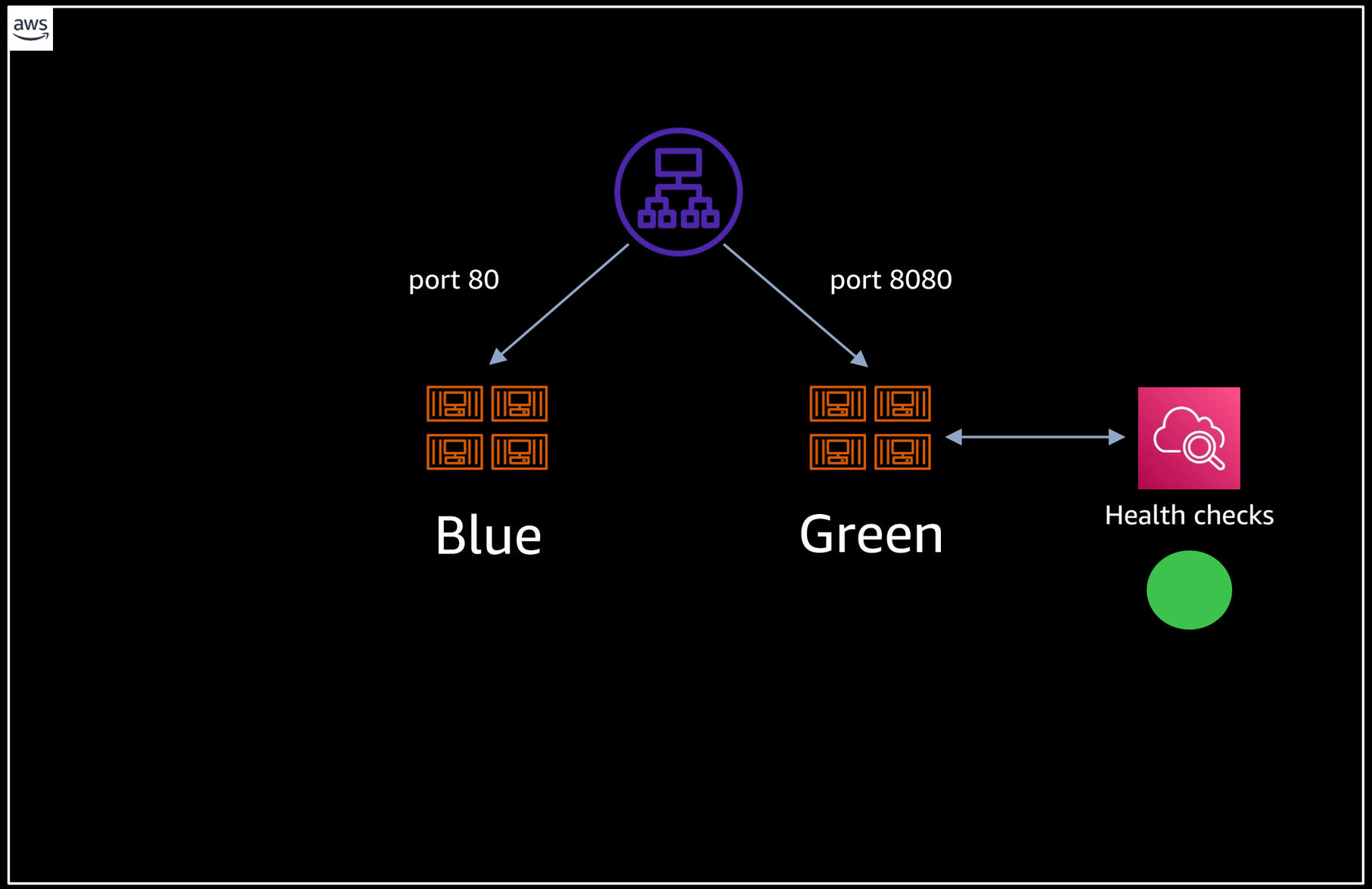
Max healthy
200%



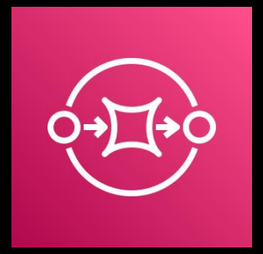
Deployment circuit breaker



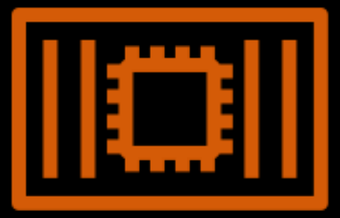
Blue / green deployments



Service Auto Scaling



Approximate number of messages visible



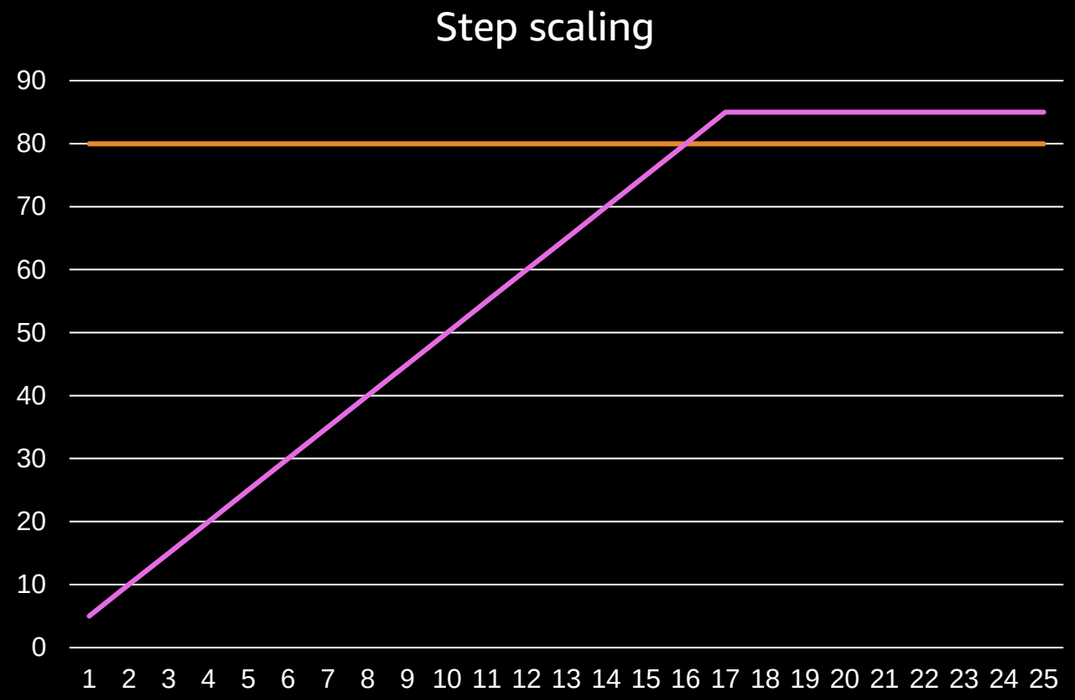
CPU
Memory

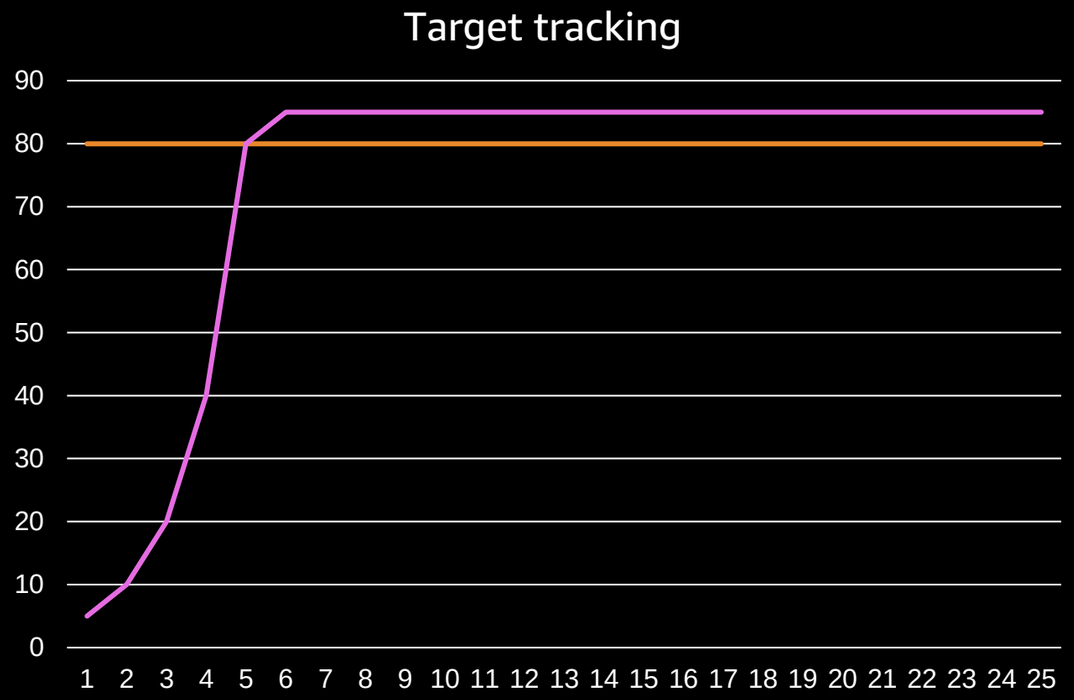


Active connection count
Request count

Custom metrics

Step vs. target Auto Scaling





Scale the infrastructure



AWS re:Invent

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

Application first



Tenets for application first:

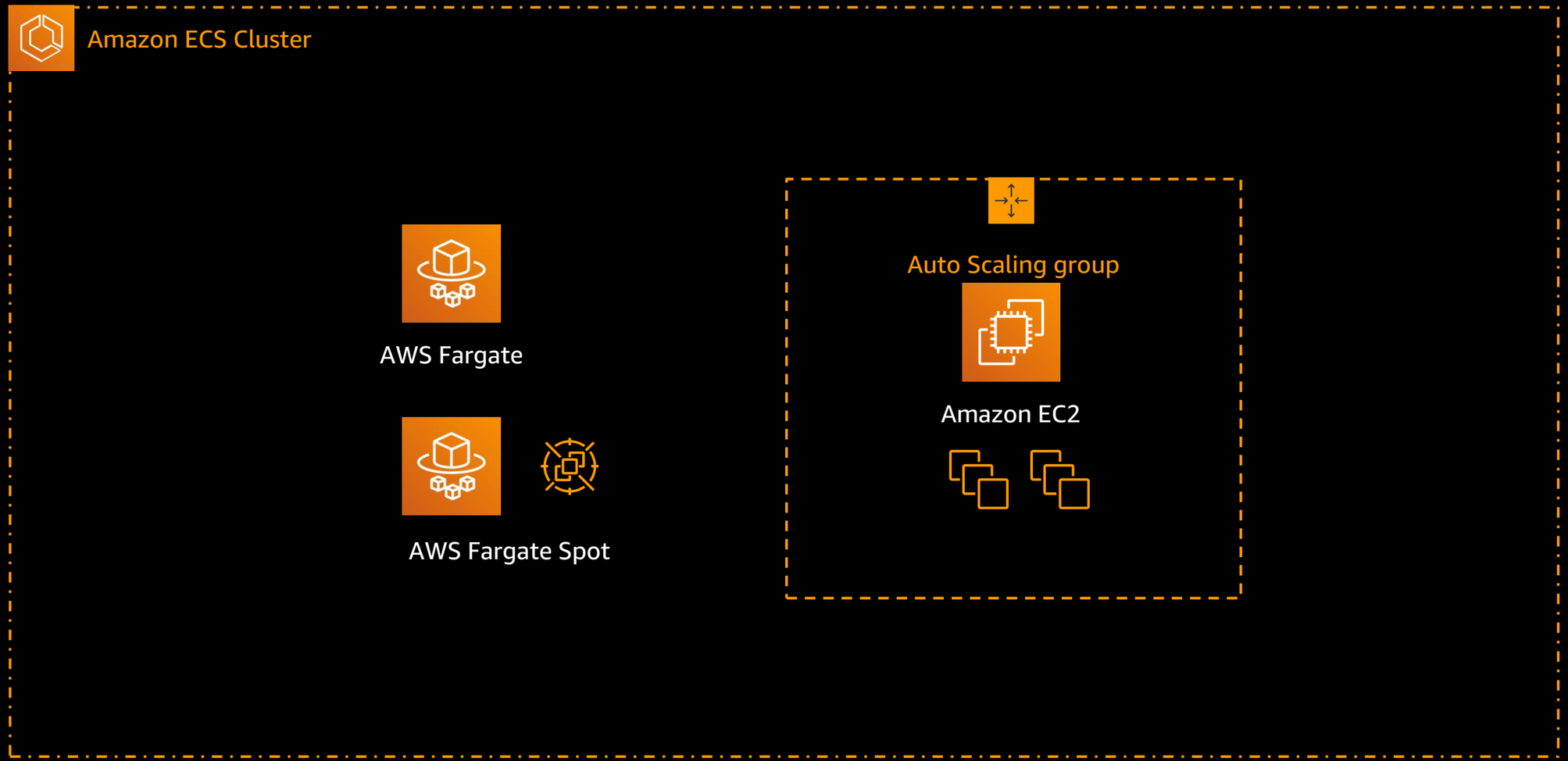
- Applications own their requirements
- Infrastructure responds to application requirements

Question for the audience

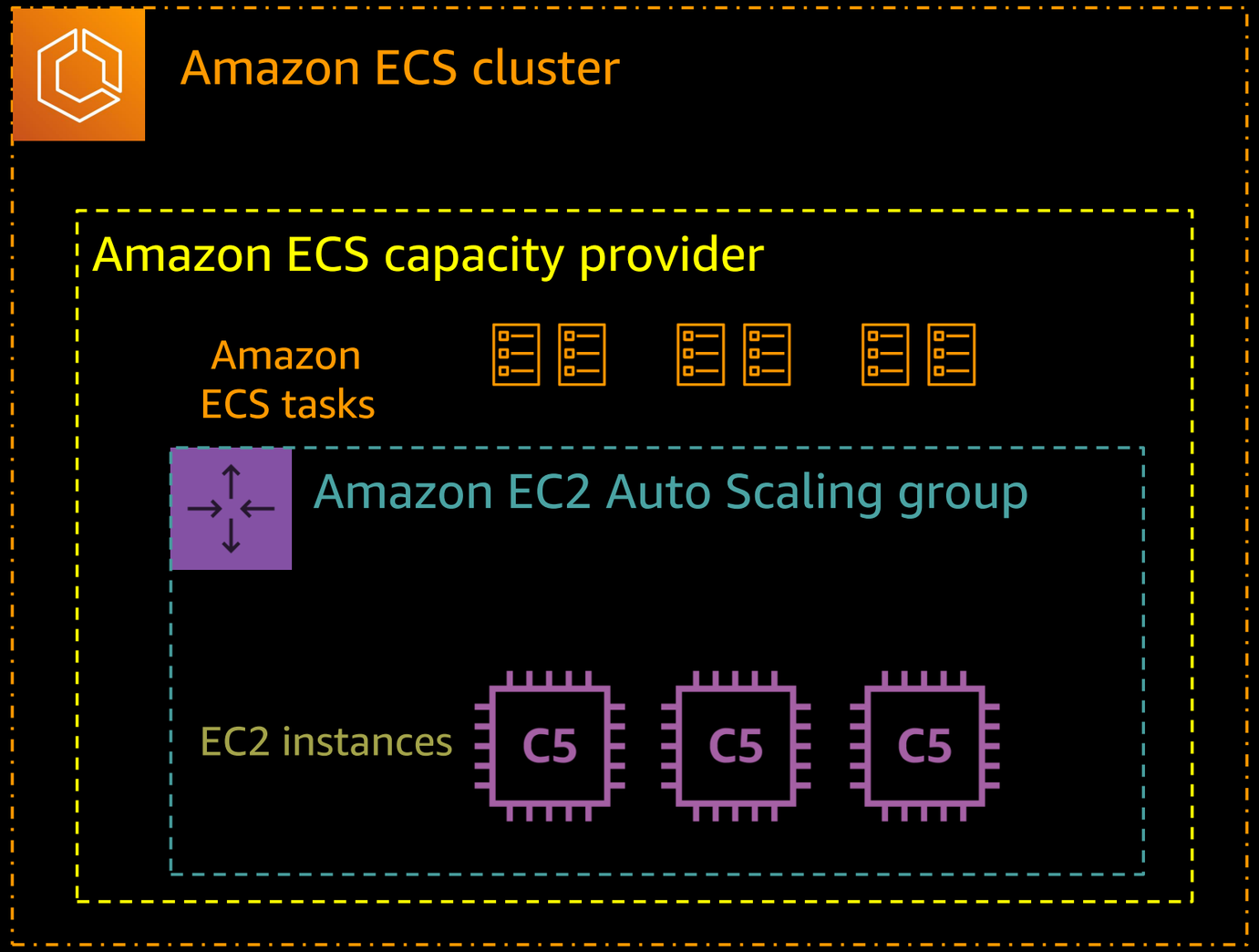
When was the last time you upgraded the control plane for your Amazon ECS clusters?

Answer: Never! The control plane is fully managed, and updates are available without having to perform any upgrades.

Amazon ECS capacity providers



Amazon ECS capacity providers setup



Create cluster (one time)

Create Auto Scaling group (one time)

Create capacity provider (one time)

Run tasks

Instances start

Tasks placed

Capacity providers strategies: Fargate and Fargate Spot

Example strategy:

Fargate: base=1, weight=1

Fargate Spot: base=0, weight=4

6 total tasks:

Fargate =2 tasks

Fargate Spot=4 tasks



Capacity provider strategy: Splitting scale out

Capacity provider strategy:

Controls how tasks are assigned to CPs

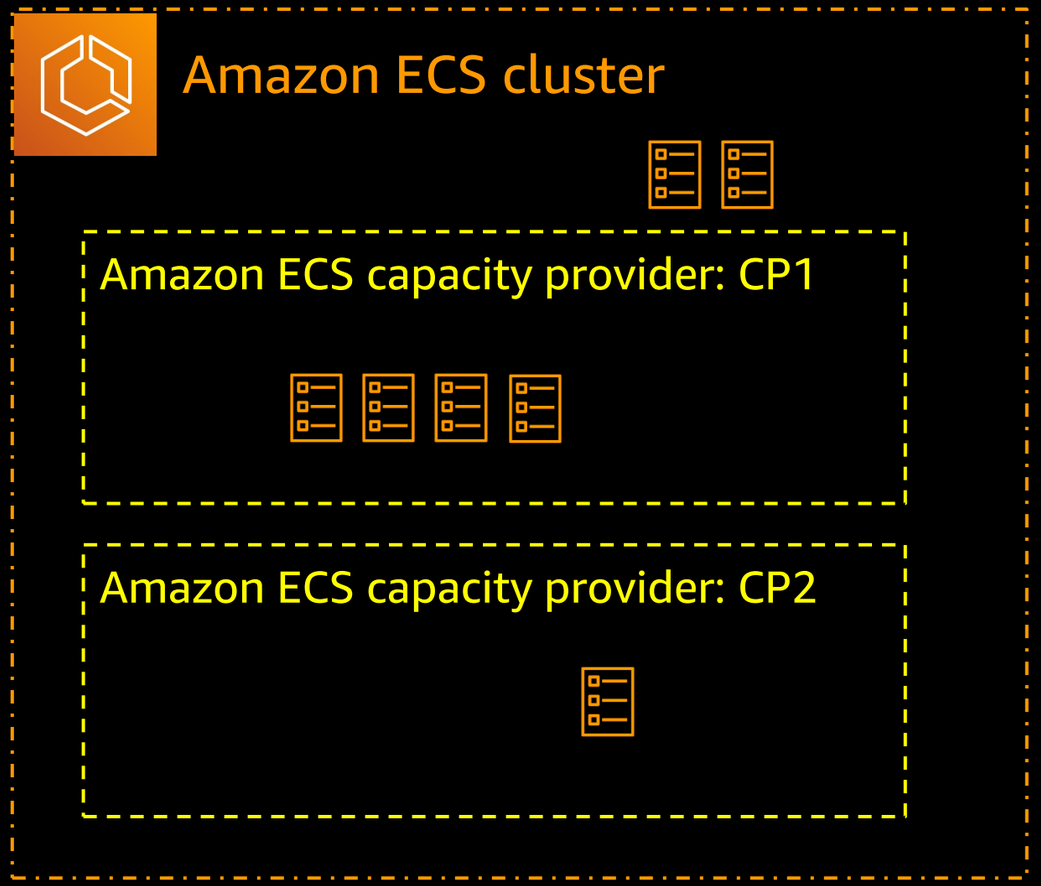
Example strategy:

CP1: base=3, weight=1

CP2: base=0, weight=1

7 total tasks:

CP1=5 tasks CP2=3 task



Example metrics:

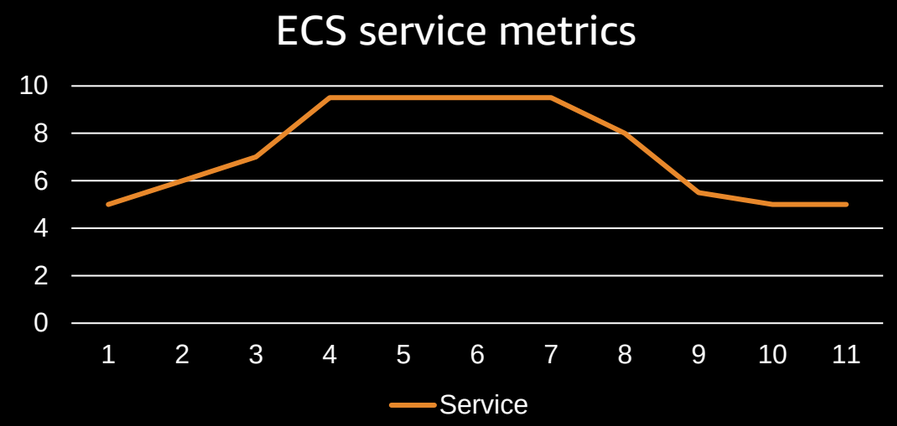
CPU, Memory, Network Traffic, Queue Depth

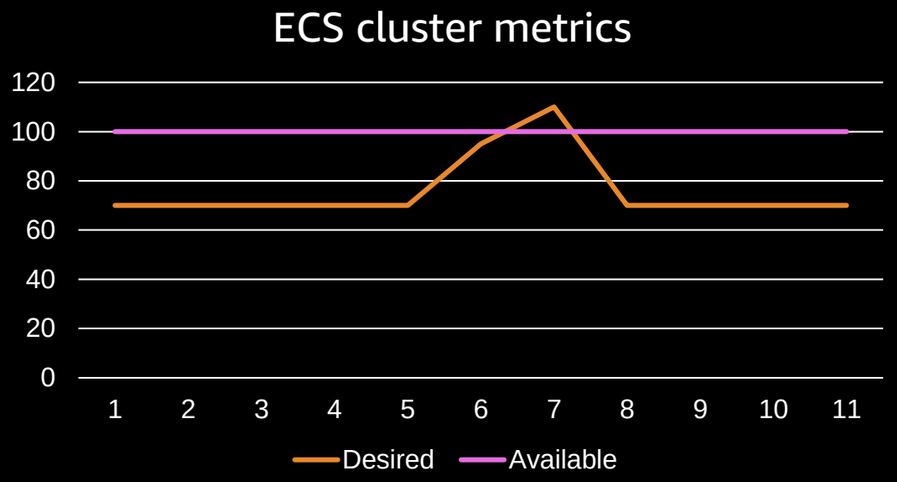
Amazon CloudWatch

AWS Fargate

Amazon ECS cluster

Amazon ECS service





Placement strategies



Binpack

Random

Spread

Understand service quotas

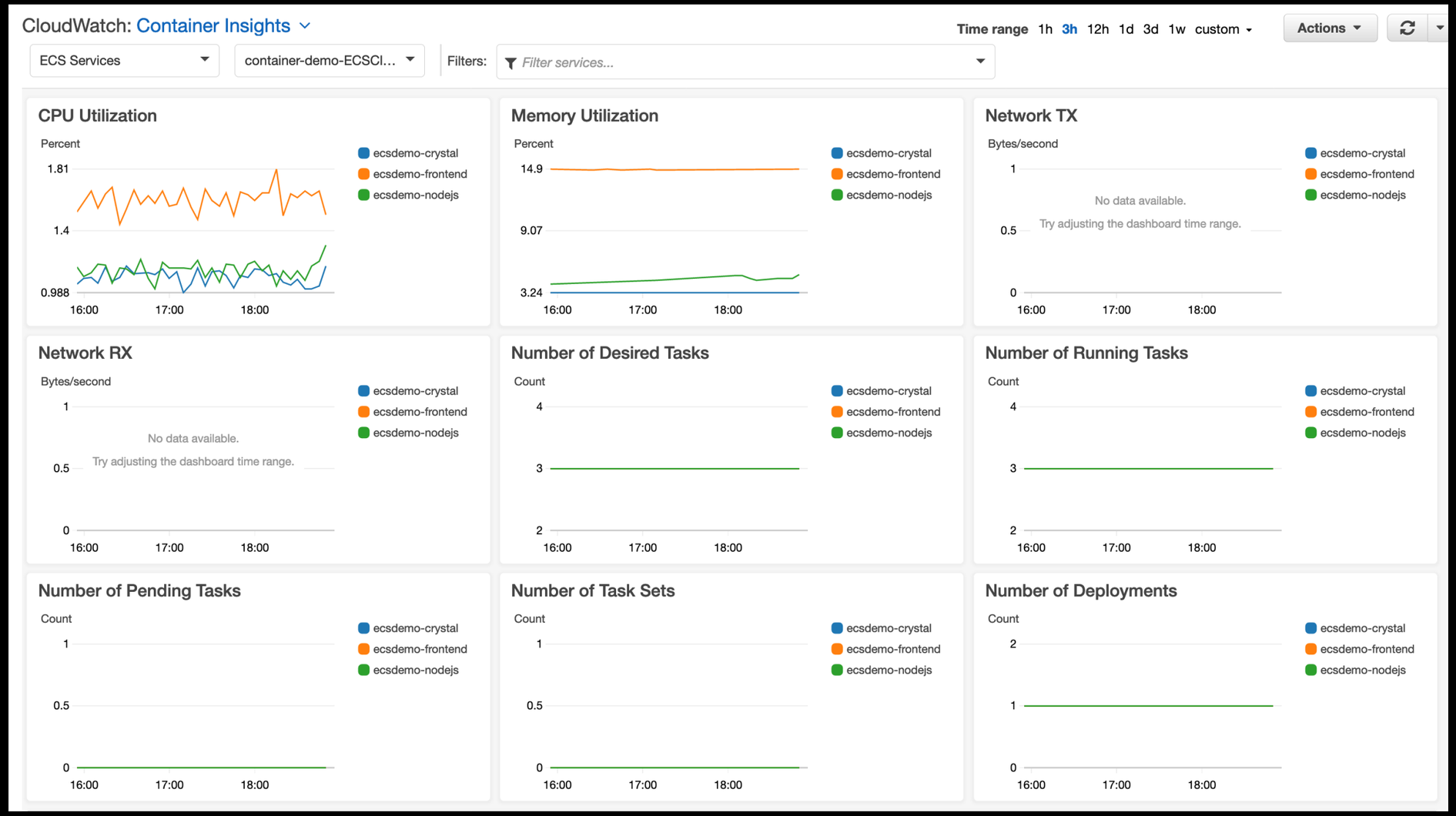
- Adjustable vs. not
- Plan ahead
- Amazon CloudWatch alarms

Monitoring / tracing

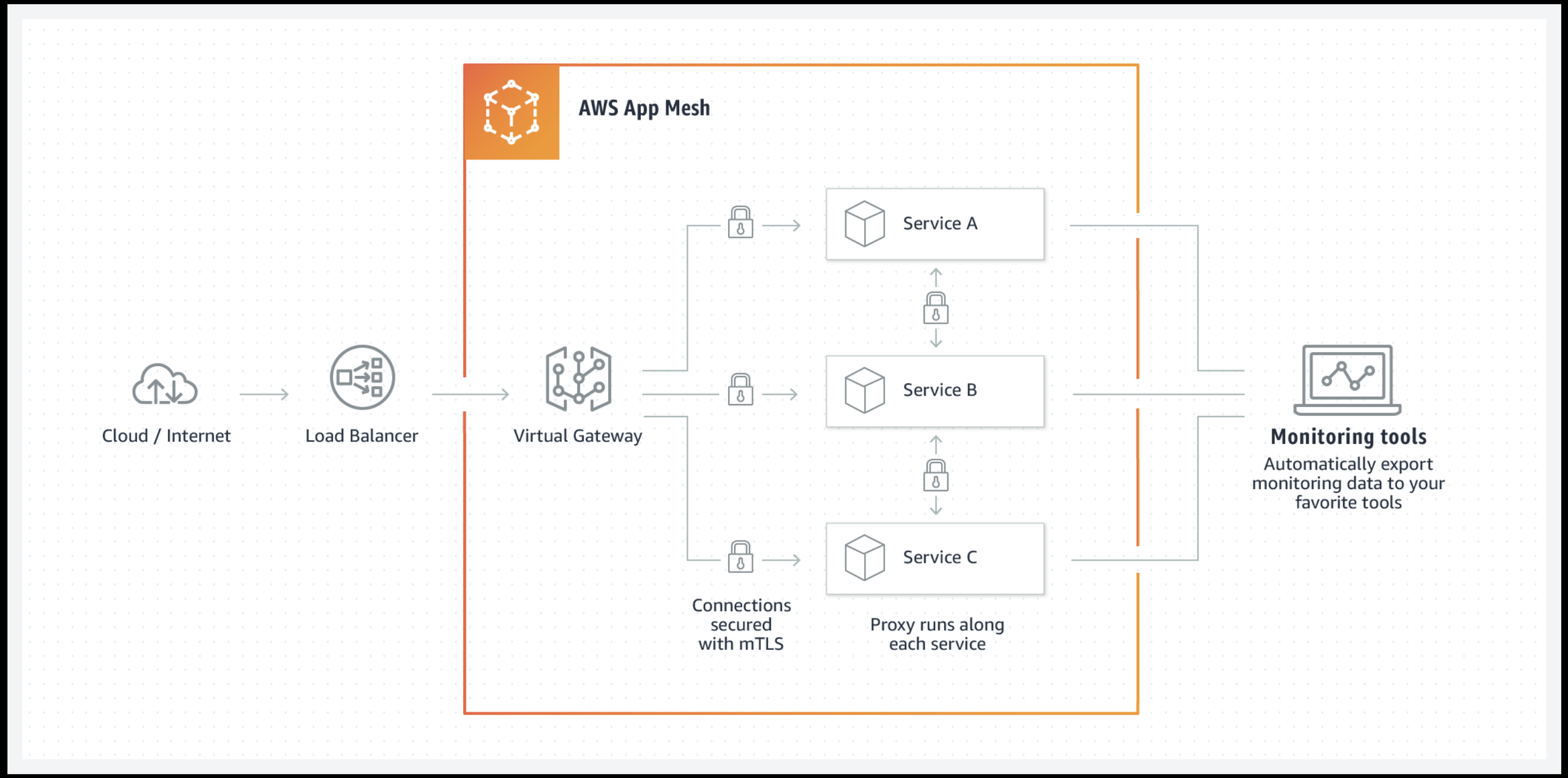


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

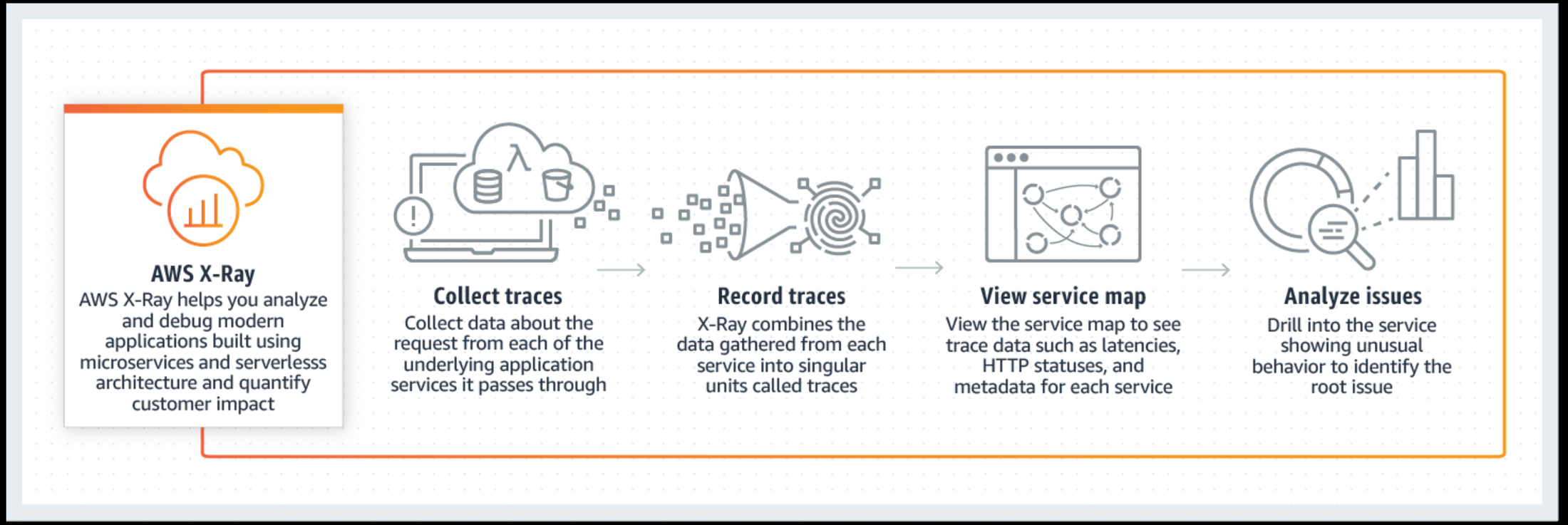
Amazon CloudWatch container insights



AWS App Mesh


AWS App Mesh
Cloud / Internet
Load Balancer
Virtual Gateway
Service A
Service B
Service C
Connections secured with mTLS
Proxy runs along each service
Monitoring tools
Automatically export monitoring data to your favorite tools

AWS X-Ray


AWS X-Ray
AWS X-Ray helps you analyze and debug modern applications built using microservices and serverlesss architecture and quantify customer impact
Collect traces
Collect data about the request from each of the underlying application services it passes through
Record traces
X-Ray combines the data gathered from each service into singular units called traces
View service map
View the service map to see trace data such as latencies, HTTP statuses, and metadata for each service
Analyze issues
Drill into the service showing unusual behavior to identify the root issue

Partner ecosystem

- AWS Containers Competency Partners
  - https://aws.amazon.com/containers/partner-solutions/

Resources

- ECS workshop
  - https://ecsworkshop.com
- Containers from the couch
  - https://containersfromthecouch.com
- AWS containers road map
  - https://github.com/aws/containers-roadmap/projects/1

Thank you!

Adam Keller

Developer Advocate, Containers
AWS

Twitter: @realadamjkeller
Github: adamjkeller

© 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.