Prague AWS User Group

Running Docker Containers on AWS (with ECS)

Vladimir Simek, Solutions Architect @ AWS
20 September 2017

Agenda

Why Containers?

Cluster Management

Running Services

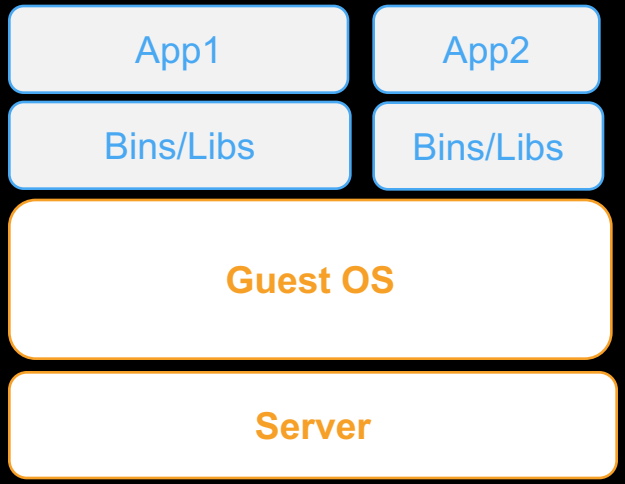
Placement Options

Security and Monitoring

Demo

Why Containers?

What are Containers?



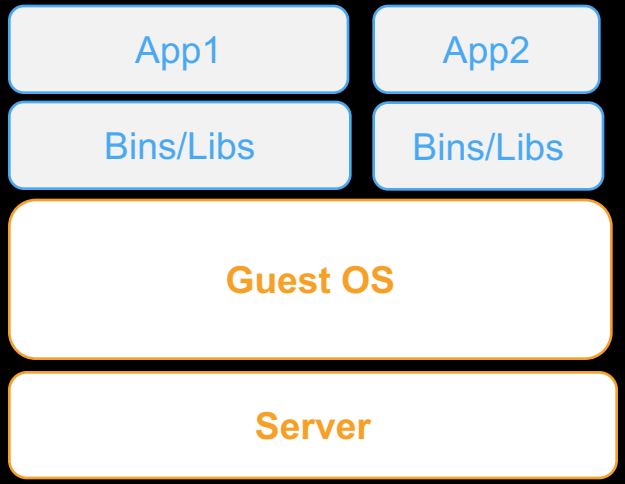
OS virtualization

Process isolation

Images

Automation

Container advantages



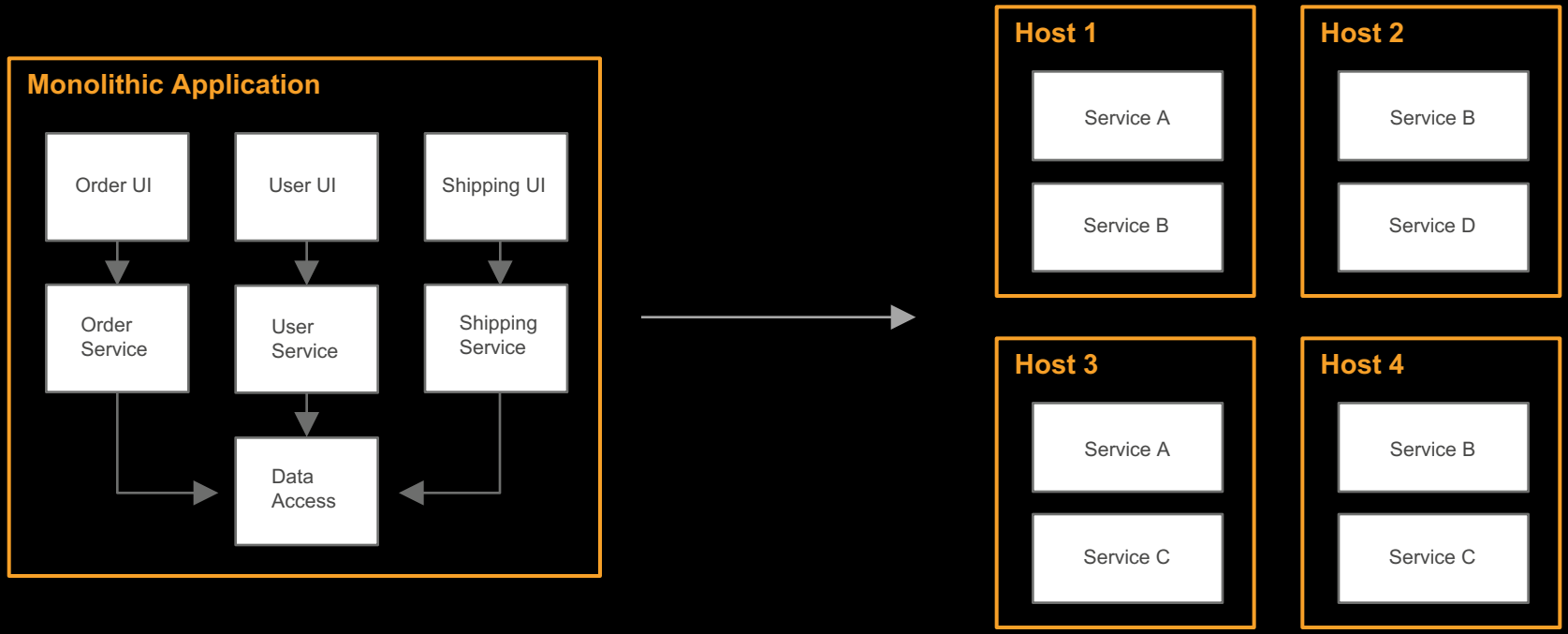
Portable

Flexible

Fast

Efficient

Services evolve to microservices



Containers are natural for microservices

Simple to model

Any app, any language

Image is the version

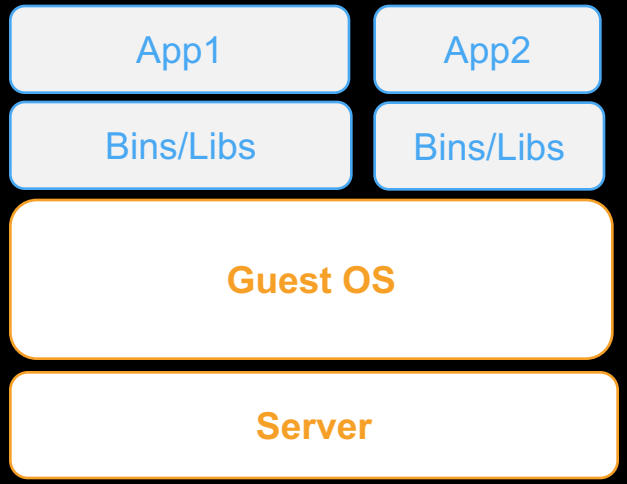
Test & deploy same artifact

Stateless servers decrease change risk

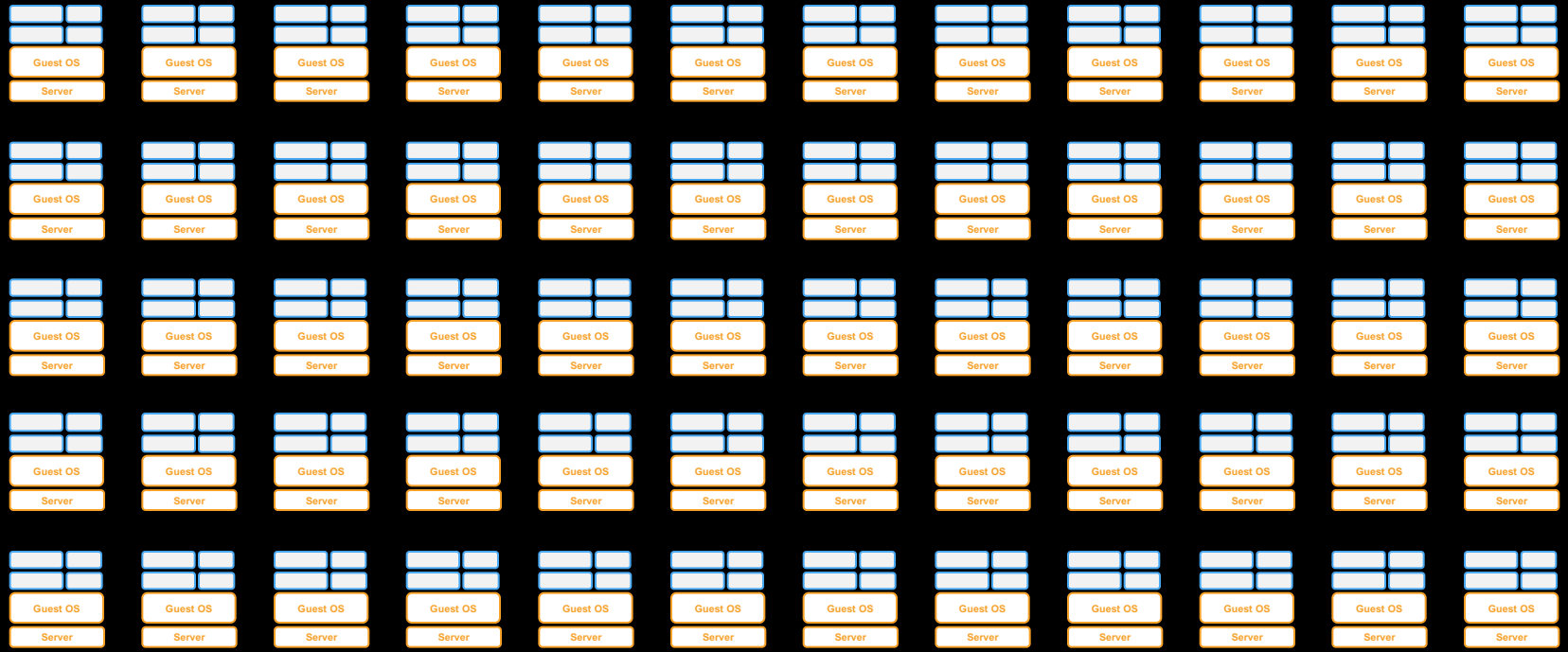
Scheduling


7a
8a
9a
10a
Pick up dry cleaning
11a
12p
Eat lunch
1p
2p
Bike to the store
3p
4p
Relax
5p
6p
7p

Scheduling one resource is straightforward



Scheduling a cluster is hard

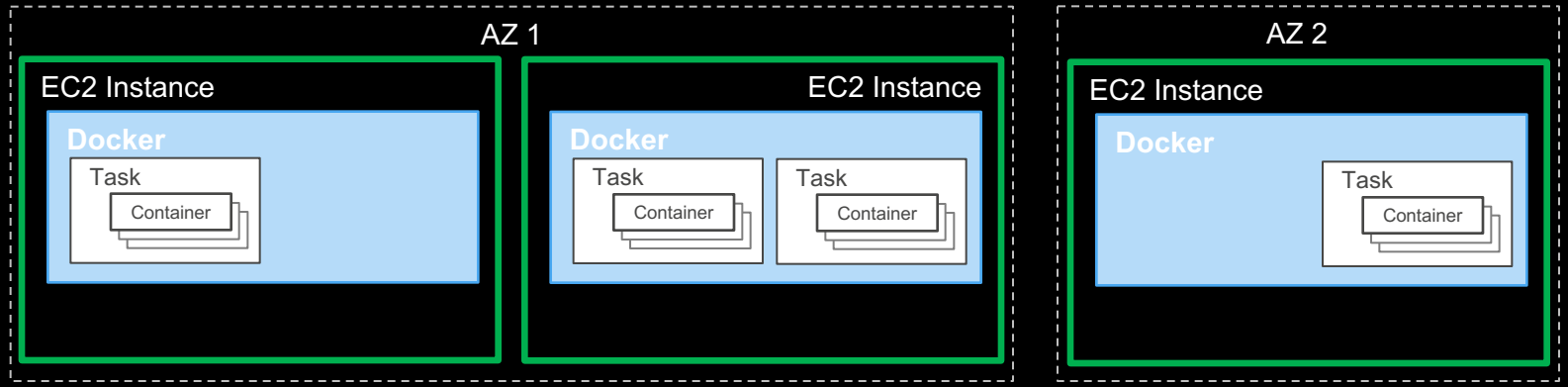


What is Amazon ECS?

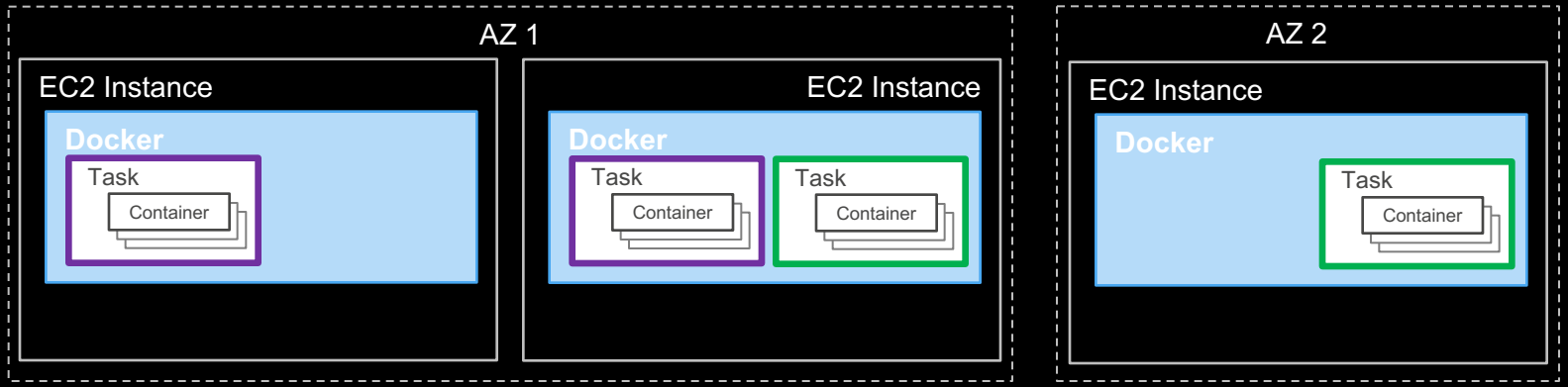
Amazon EC2 Container Service (ECS) is a highly scalable, high performance **container management service**. You can use Amazon ECS to **schedule** the placement of containers across your cluster. You can also integrate your own **scheduler** or **third-party scheduler** to meet business or application specific requirements.

Cluster Management

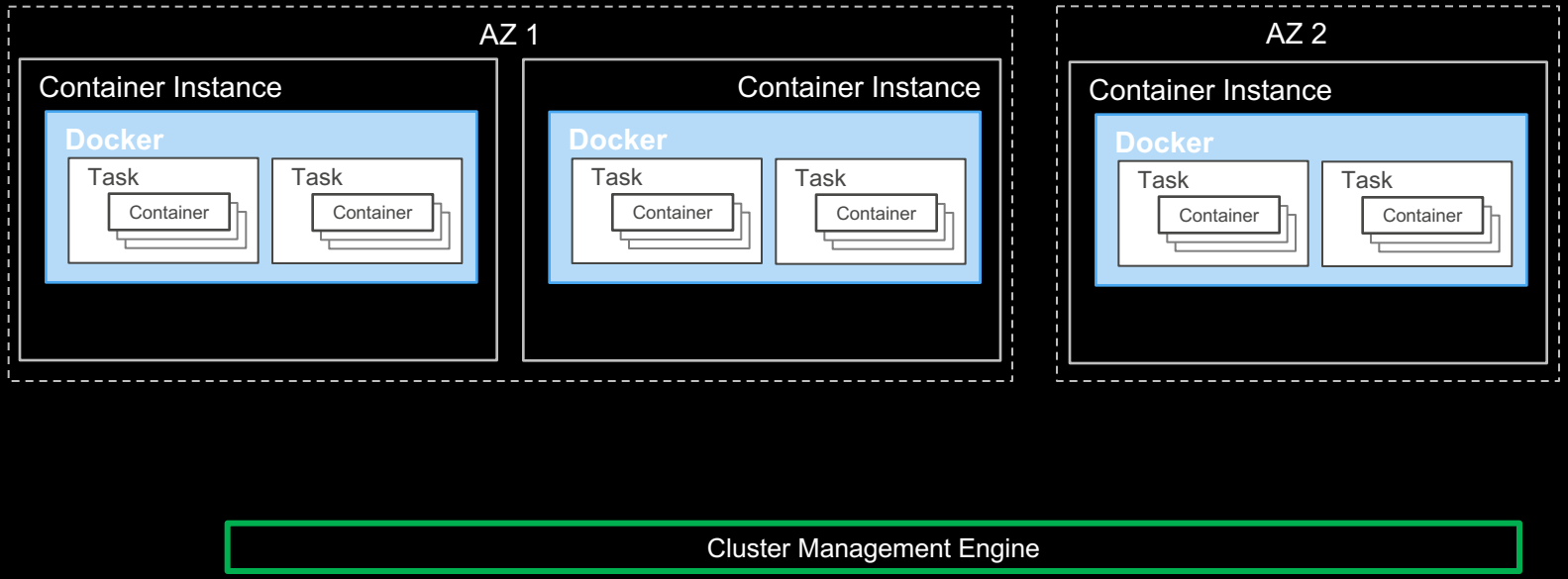
Cluster Management: Resource Management



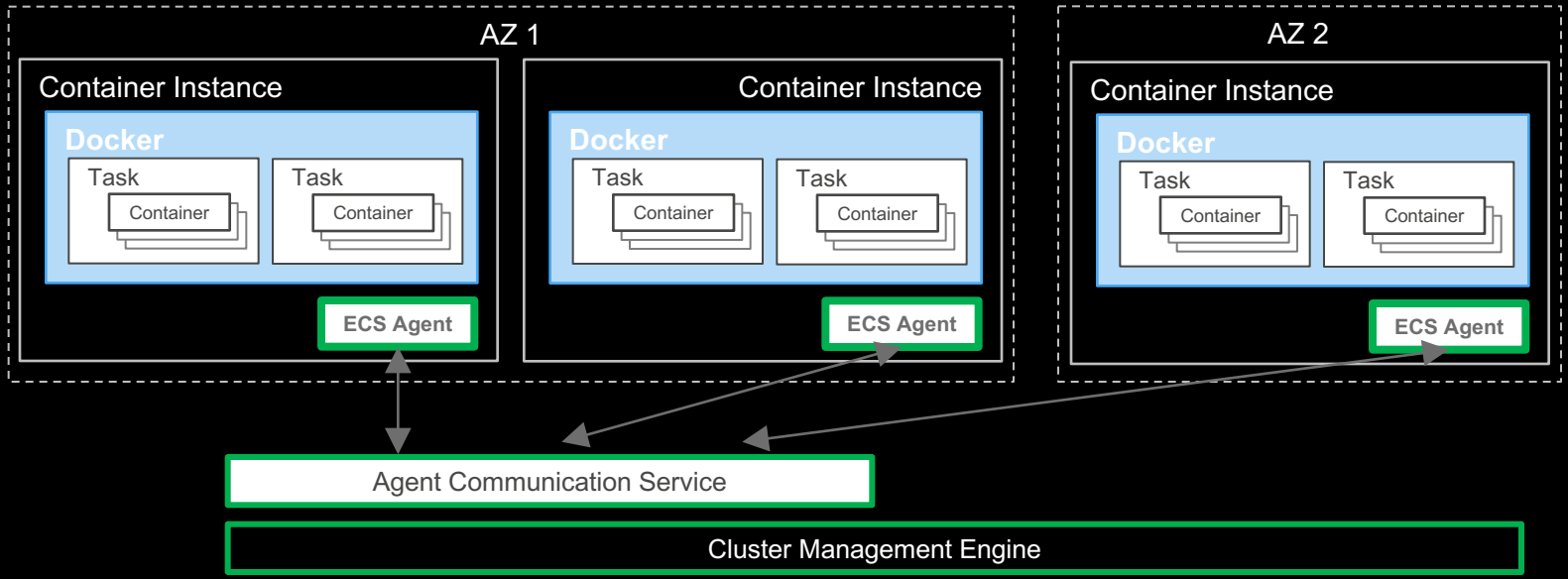
Cluster Management: Scheduling



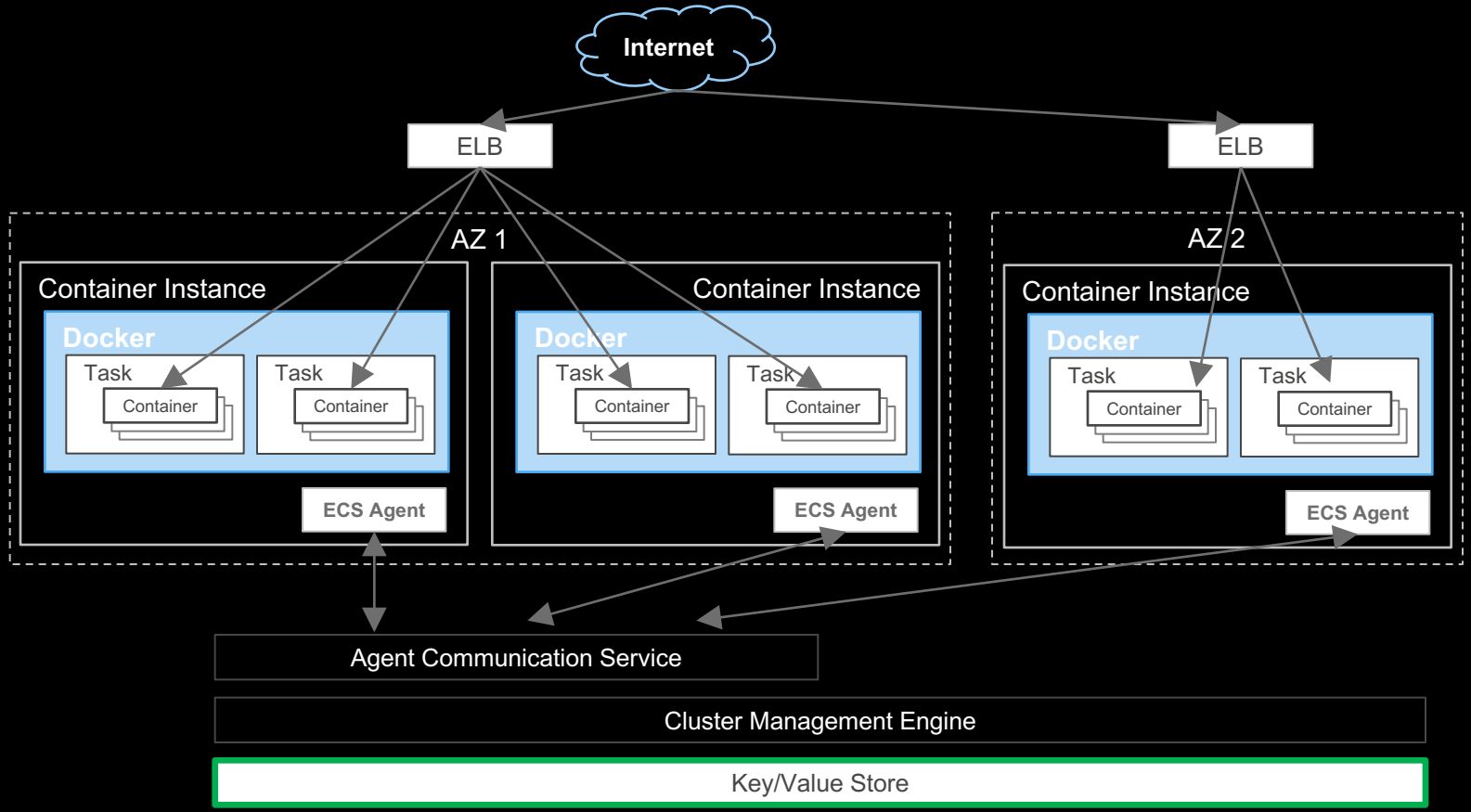
Amazon ECS: Resource Management



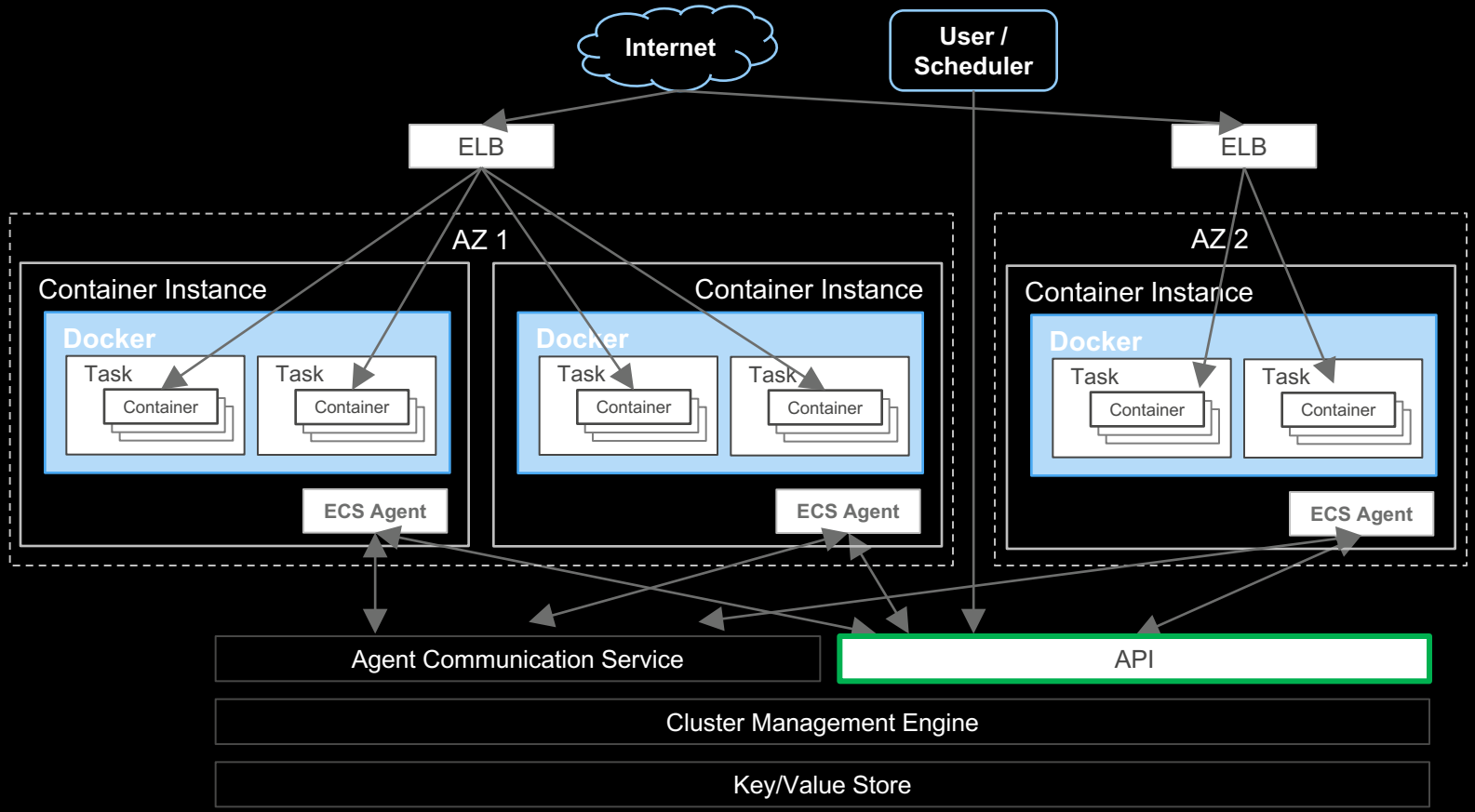
Amazon ECS: Agent Communication



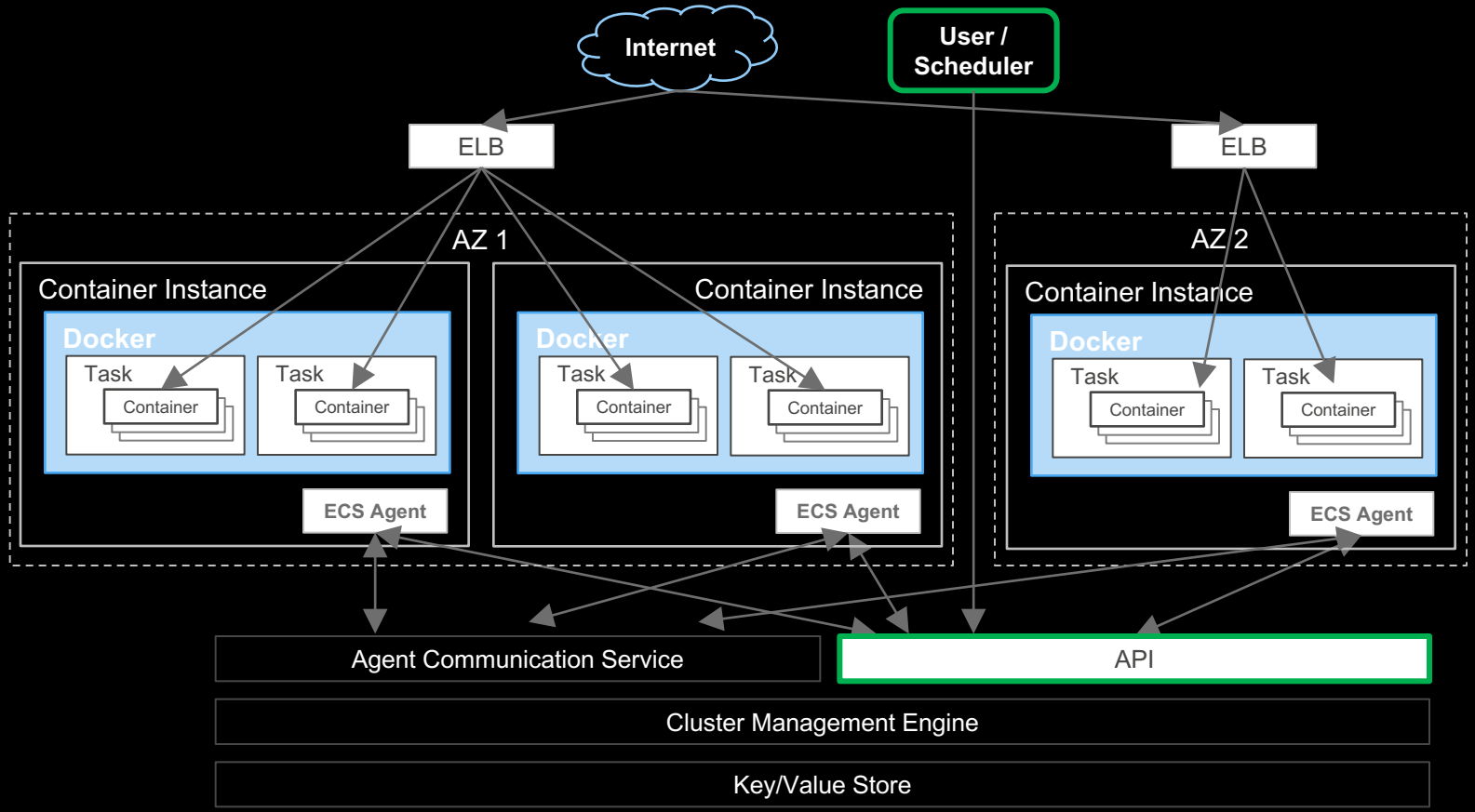
Amazon ECS: Key/Value Store



Amazon ECS: APIs

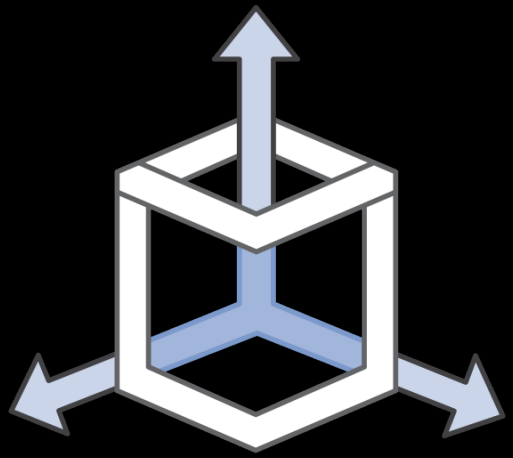


Amazon ECS: Scheduling



Benefits

Easily Manage Clusters for Any Scale



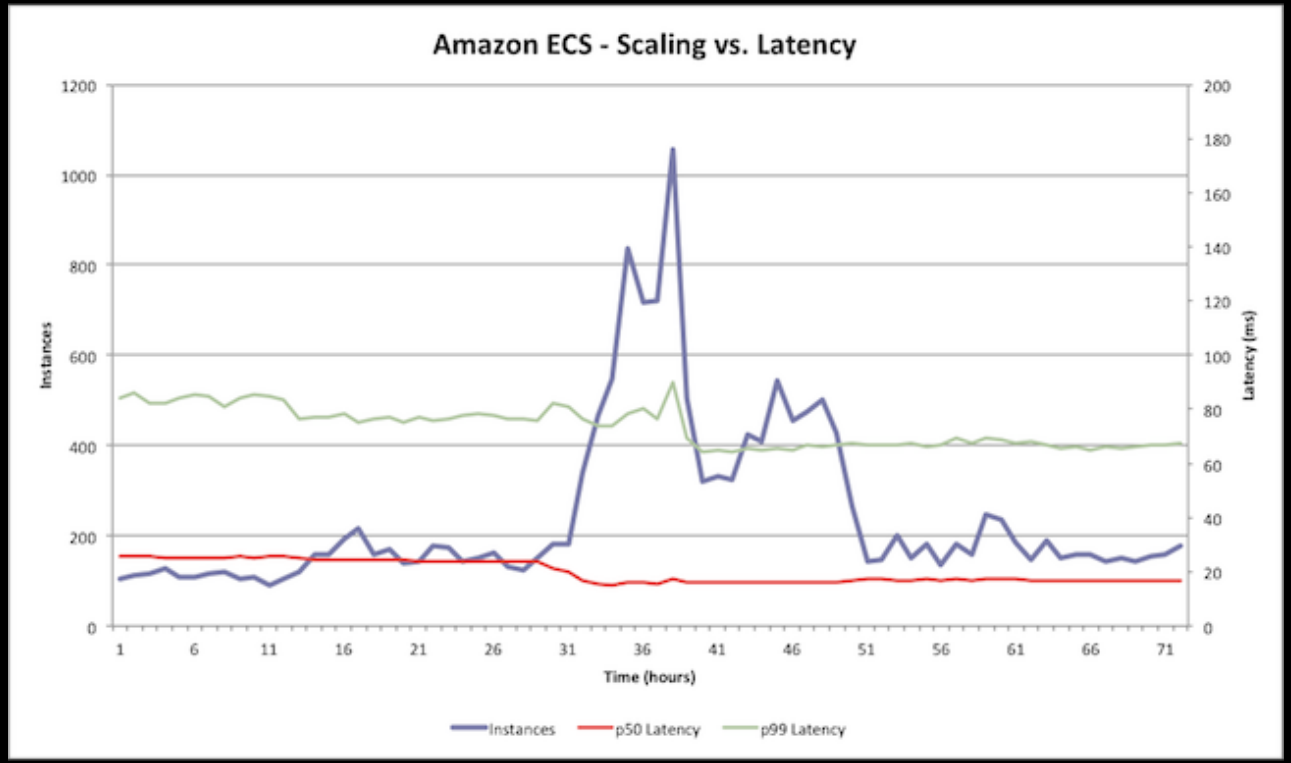
Nothing to run

Complete state

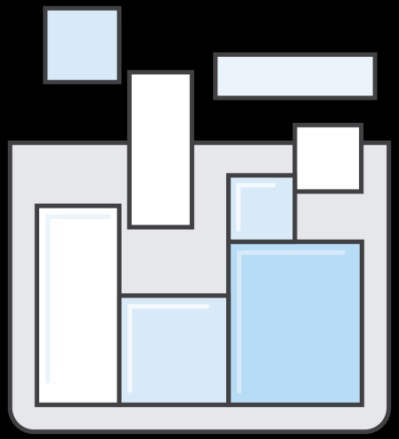
Control and monitoring

Scale

Scalable


Amazon ECS - Scaling vs. Latency
Instances
Latency (ms)
Time (hours)
0
200
400
600
800
1000
1200
0
20
40
60
80
100
120
140
160
180
200
1
6
11
16
21
26
31
36
41
46
51
56
61
66
71
Instances
p50 Latency
p99 Latency

Flexible Container Placement

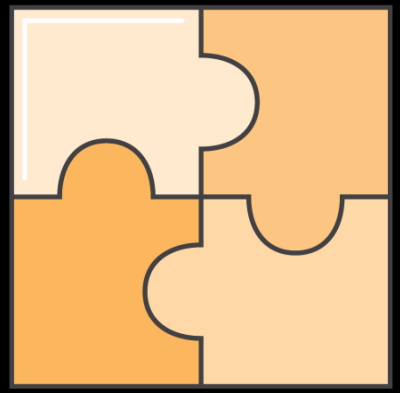


Applications

Batch jobs

Multiple schedulers

Designed for use with other AWS services



Elastic Load Balancing

Amazon Elastic Block Store

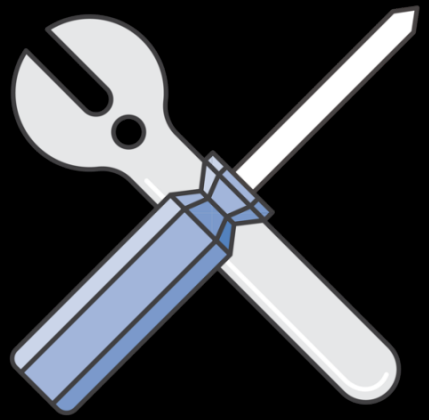
Amazon Virtual Private Cloud

Amazon CloudWatch

AWS Identity and Access Management

AWS CloudTrail

Extensible



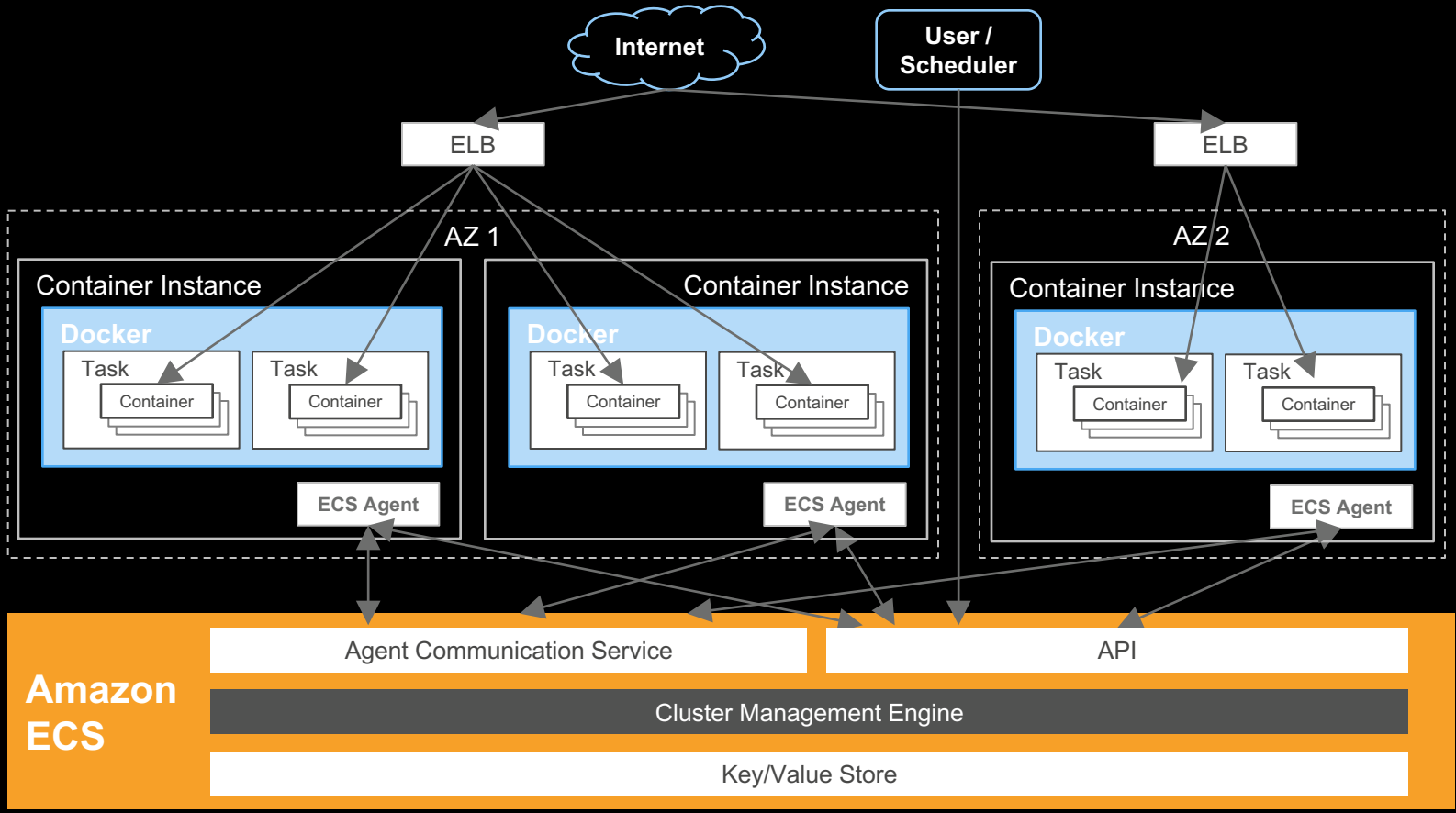
Comprehensive APIs

Custom schedulers

Open source agent and CLI

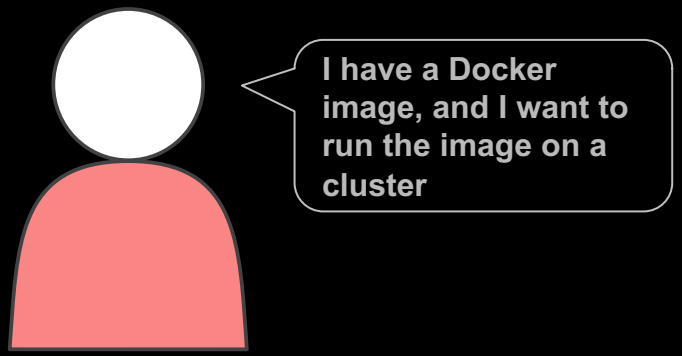
Blox

Amazon ECS

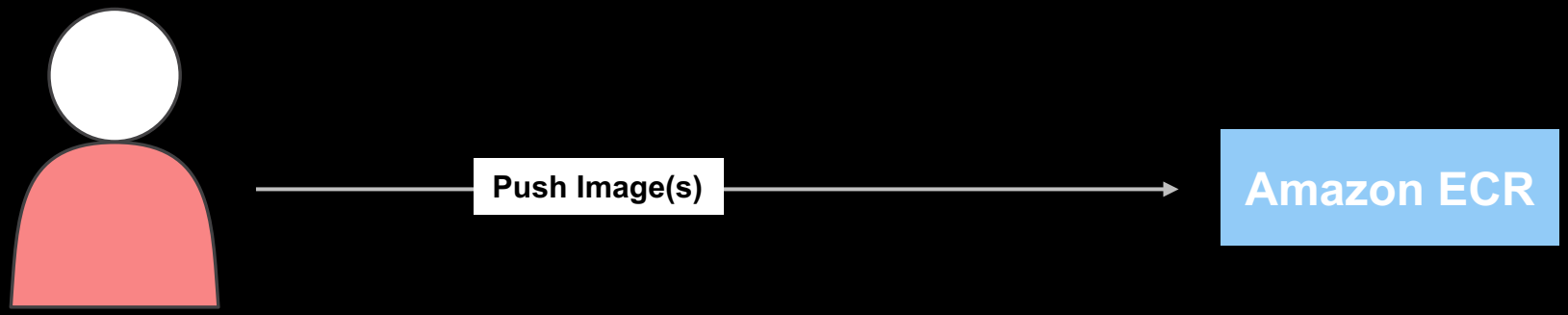

Internet
User / Scheduler
ELB
ELB
AZ 1
AZ 2
Container Instance
Container Instance
Container Instance
Docker
Docker
Docker
Task
Task
Task
Task
Task
Task
Container
Container
Container
Container
Container
Container
ECS Agent
ECS Agent
ECS Agent
Amazon ECS
Agent Communication Service
API
Cluster Management Engine
Key/Value Store

ECS Workflow

Typical User Workflow



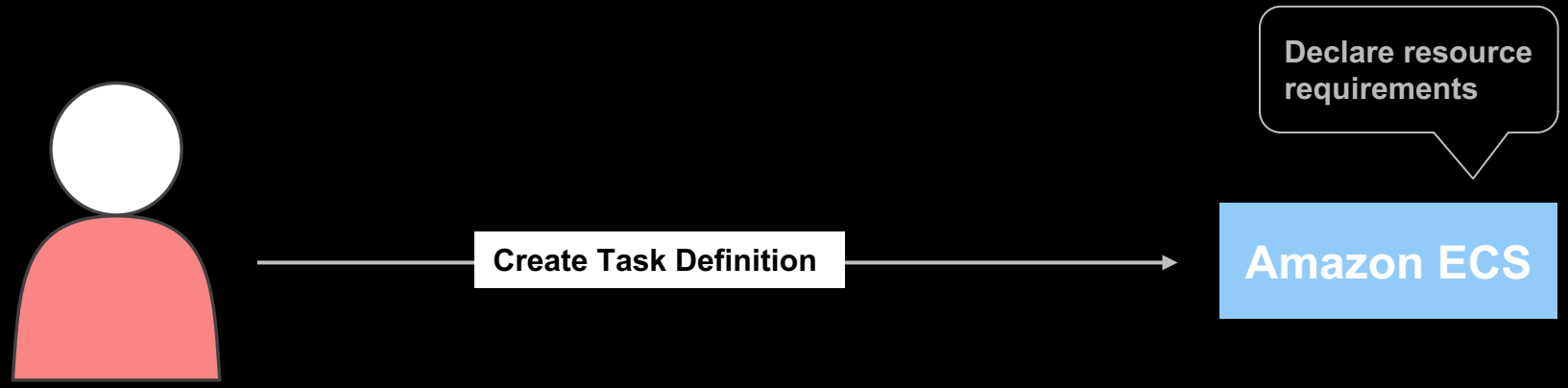
Typical User Workflow



Push Image to ECR

```
$ docker push my-account.dkr.ecr.us-east-
1.amazonaws.com/myImage
```

Typical User Workflow



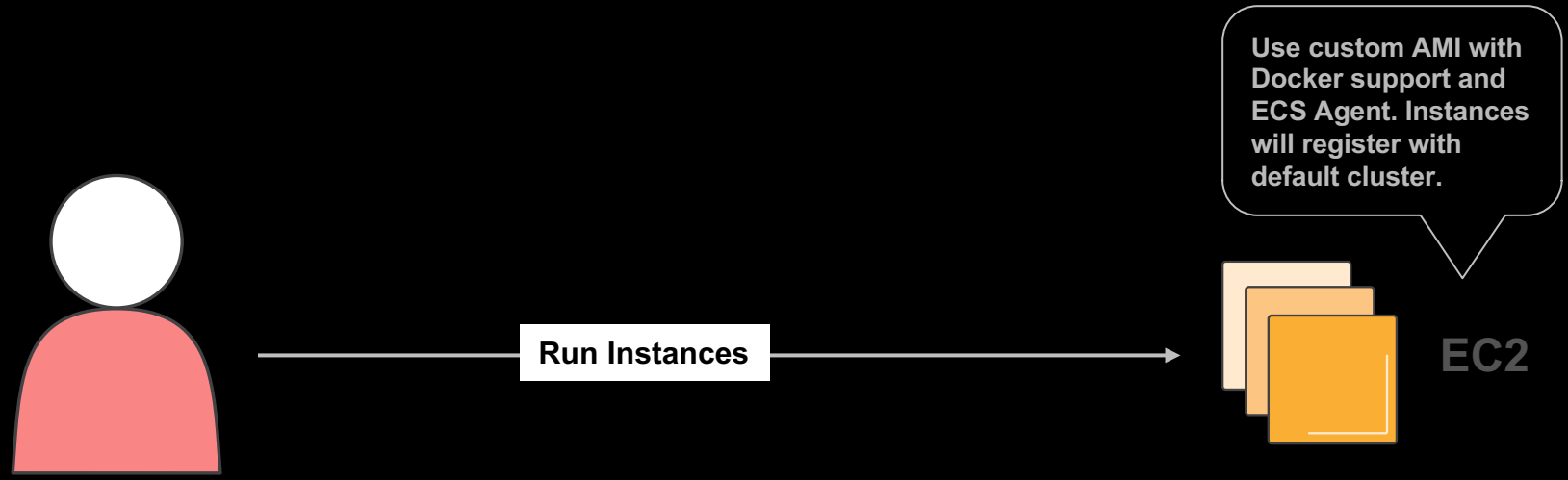
Register task definition

```
$ aws ecs register-task-definition --cli-
input-json file://task-defintion.json
```

Task definition


```
{
    "taskDefinition": {
        "status": "INACTIVE",
        "family": "curler",
        "volumes": [],
        "taskDefinitionArn": "arn:aws:ecs:us-west-2:123456778901:task-definition/curler:1",
        "containerDefinitions": [
            {
                "environment": [],
                "name": "curler",
                "mountPoints": [],
                "image": "curl:latest",
                "cpu": 100,
                "portMappings": [],
                "entryPoint": [],
                "memory": 256,
                "command": [
                    "curl -v http://example.com/"
                ],
                "essential": true,
                "volumesFrom": []
            }
        ],
        "revision": 1
    }
}
```

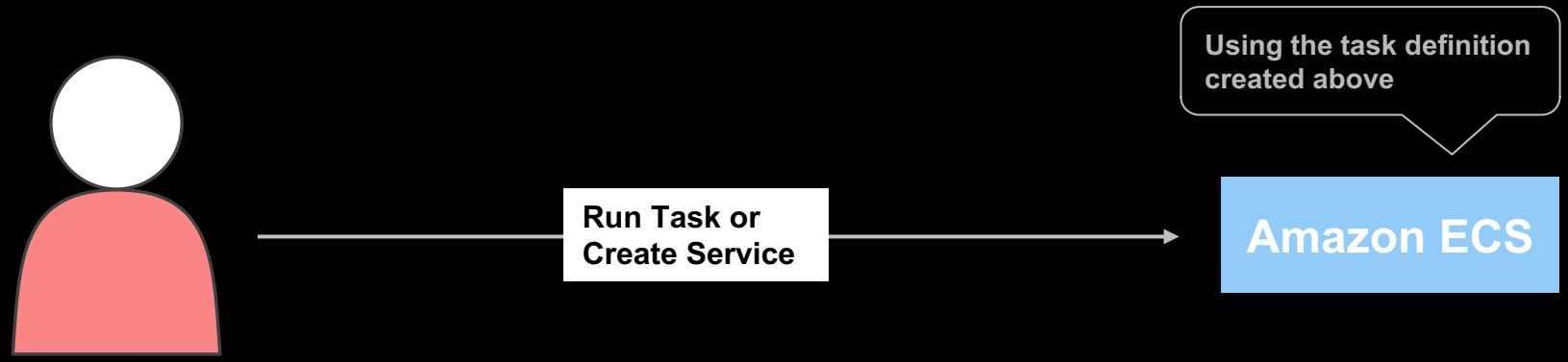
Typical User Workflow



Create cluster

```
$ aws ecs create-cluster --cluster-name
"my_cluster"
```

Typical User Workflow



Create service

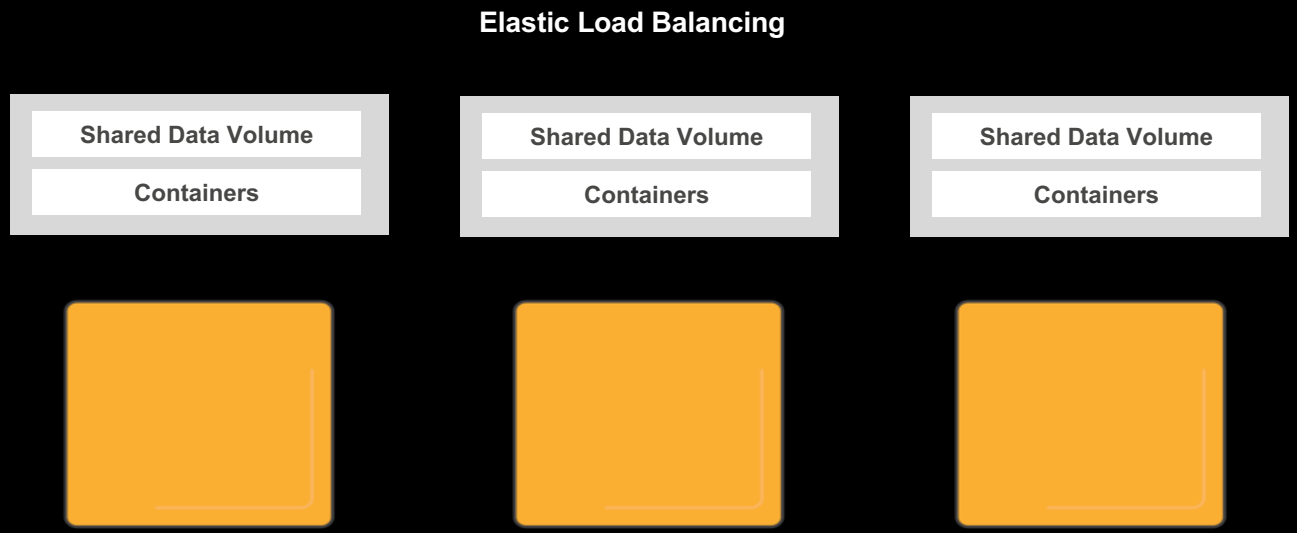
```
$ aws ecs create-service --service-name ecs-simple-service
--task-definition ecs-demo --desired-count 10
```

Create Service

Load Balance traffic across containers

Automatically recover unhealthy containers

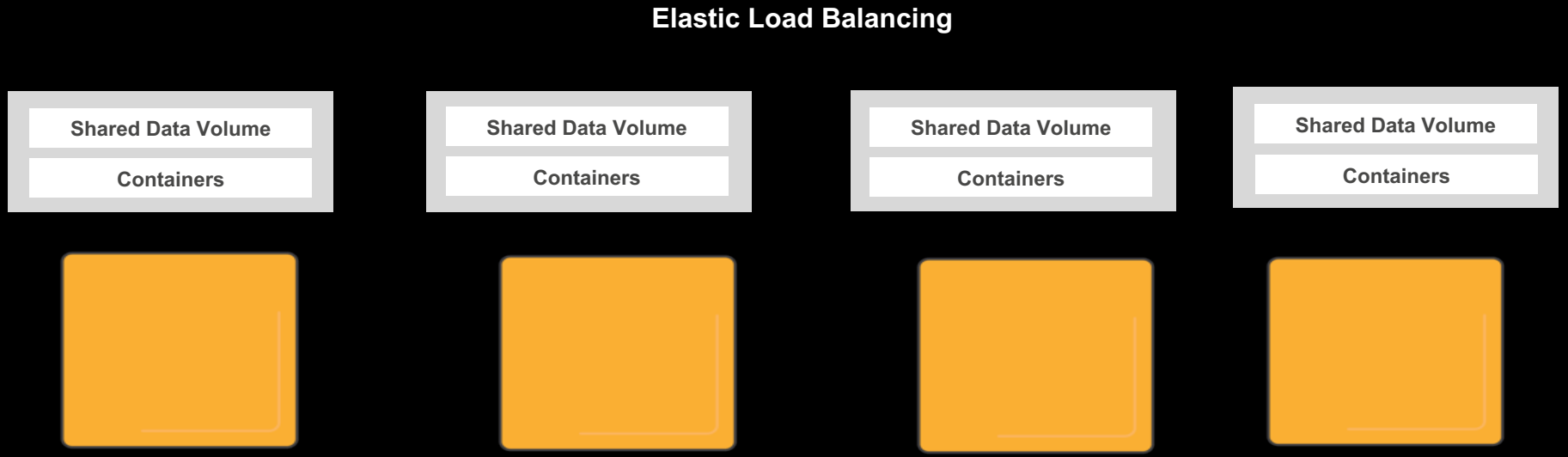
Discover services



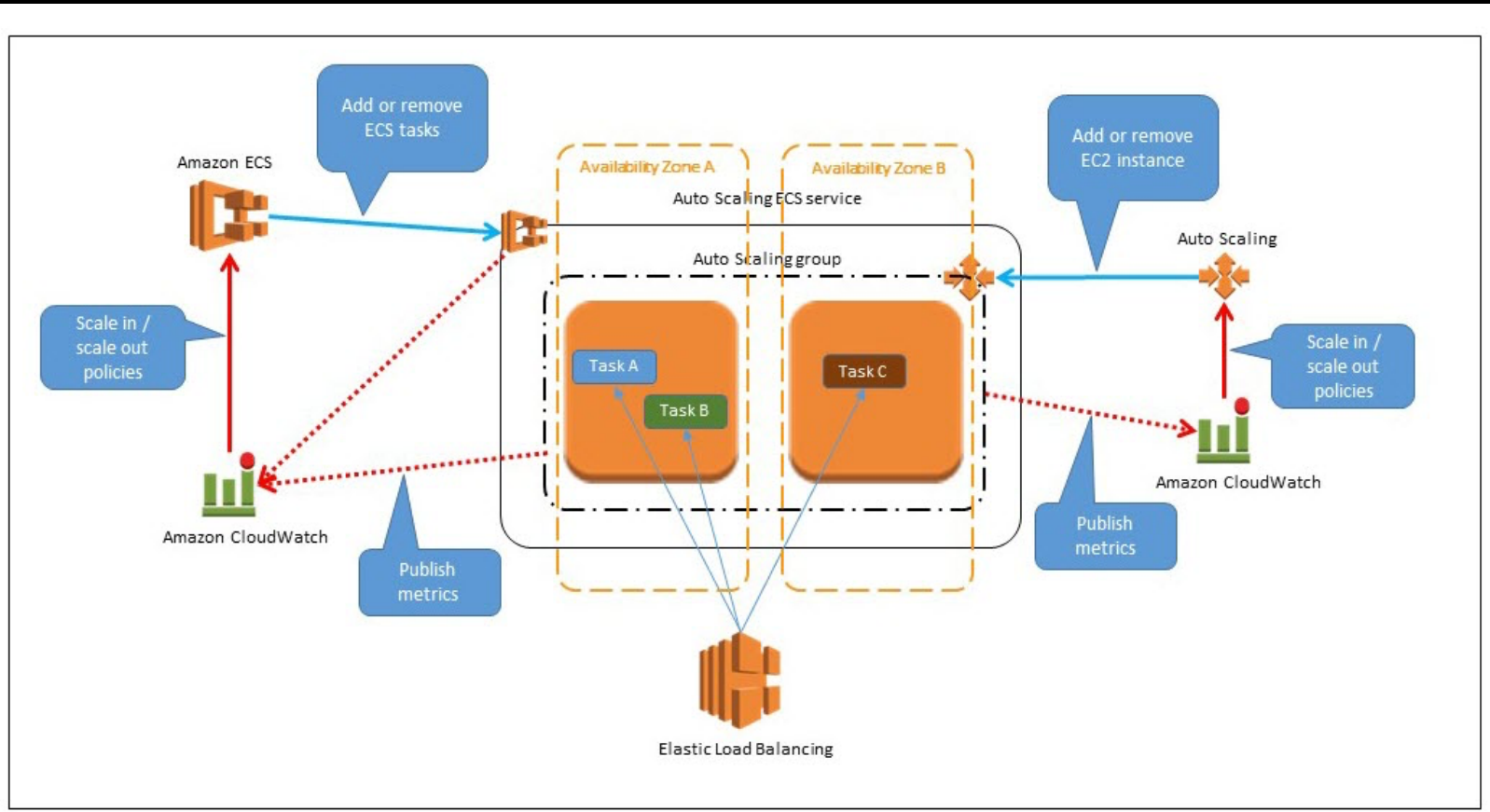
Scale Service

Scale out

Scale in



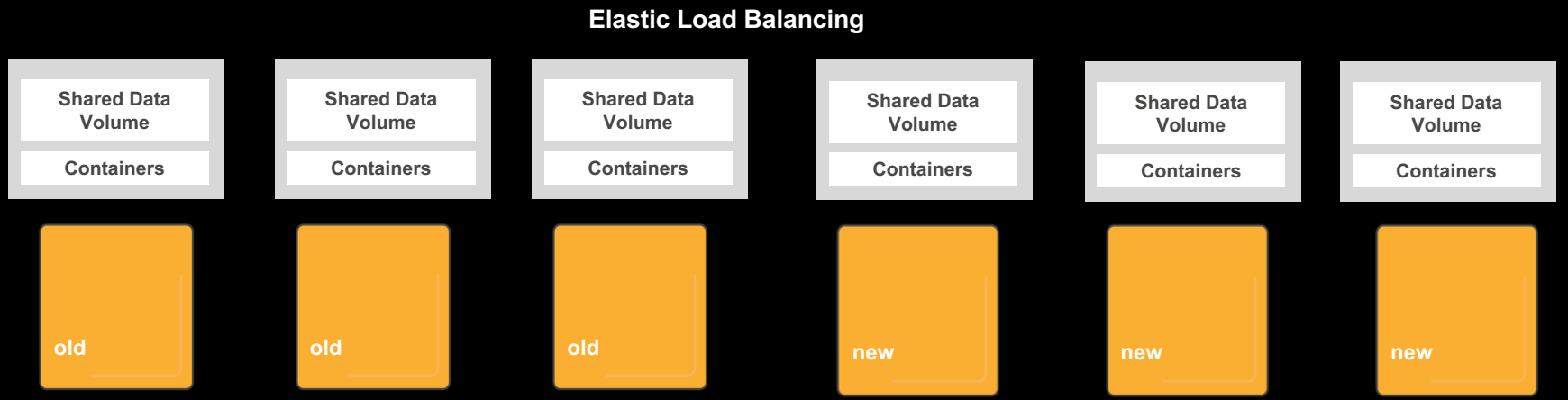
Scale Service



Update Service

Deploy new version

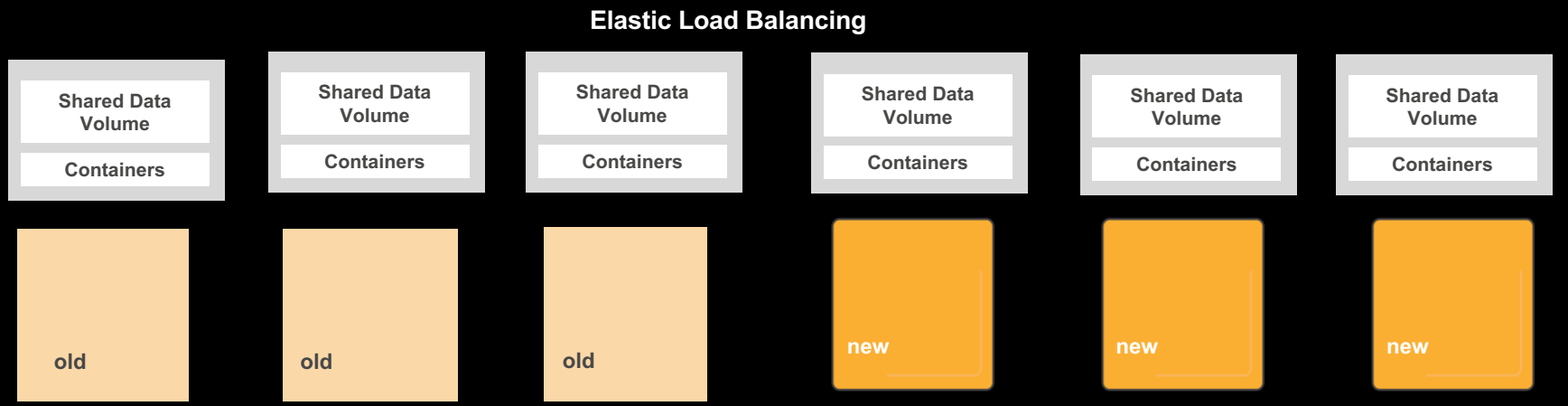
Drain connections



Update Service (cont.)

Deploy new version

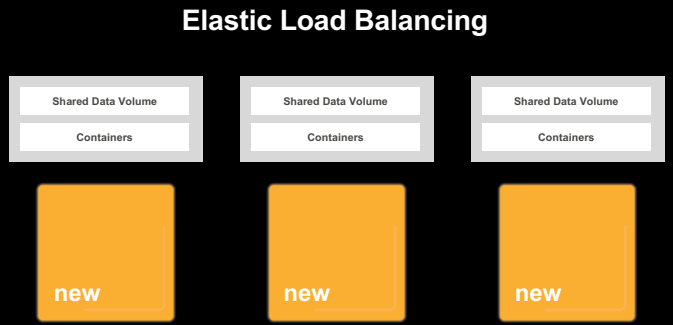
Drain connections



Update Service (cont.)

Deploy new version

Drain connections



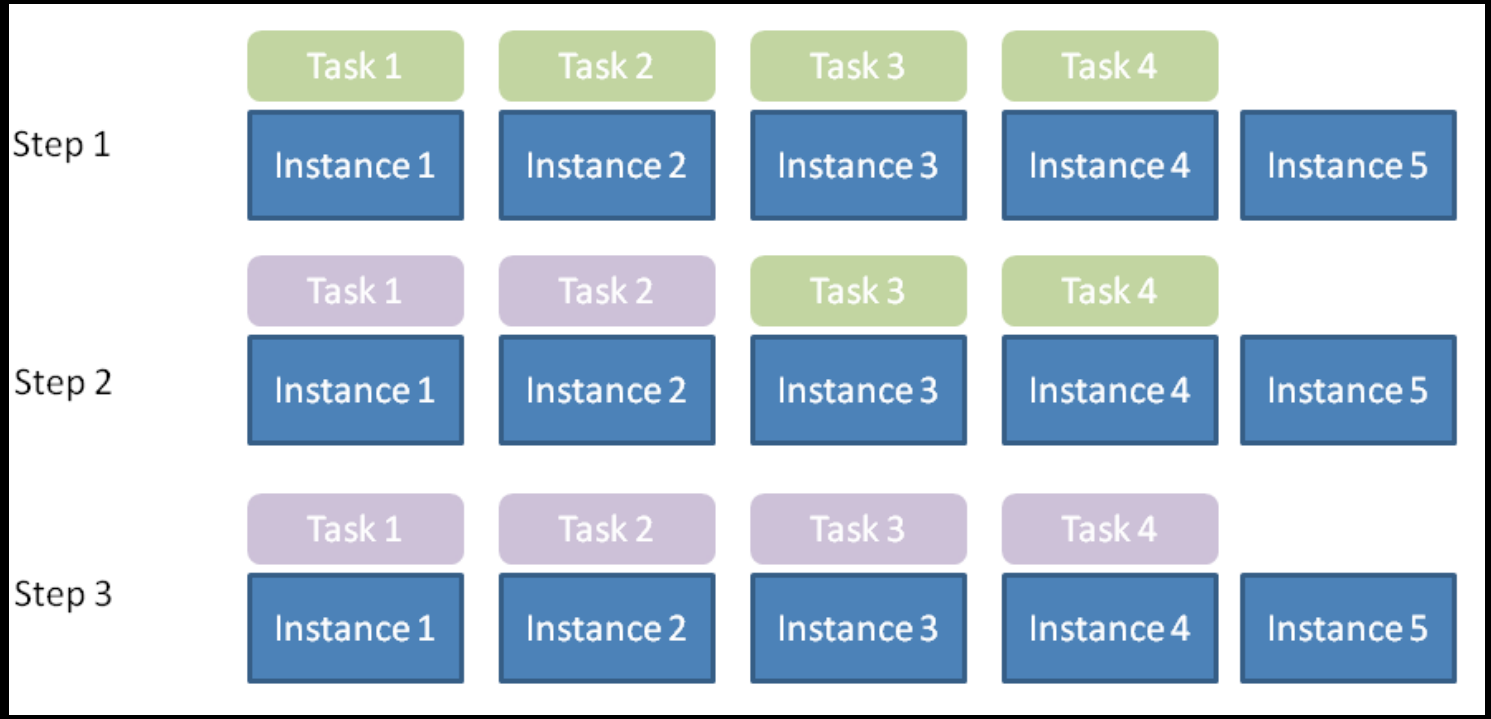
Update Service (cont.)

Specify a deployment configuration for your service:

- *minimumHealthyPercent*: lower limit (as a percentage of the service's desiredCount) of the number of running tasks that must remain running in a service during a deployment.
- *maximumPercent*: upper limit (as a percentage of the service's desiredCount) of the number of running tasks that can be running in a service during a deployment.

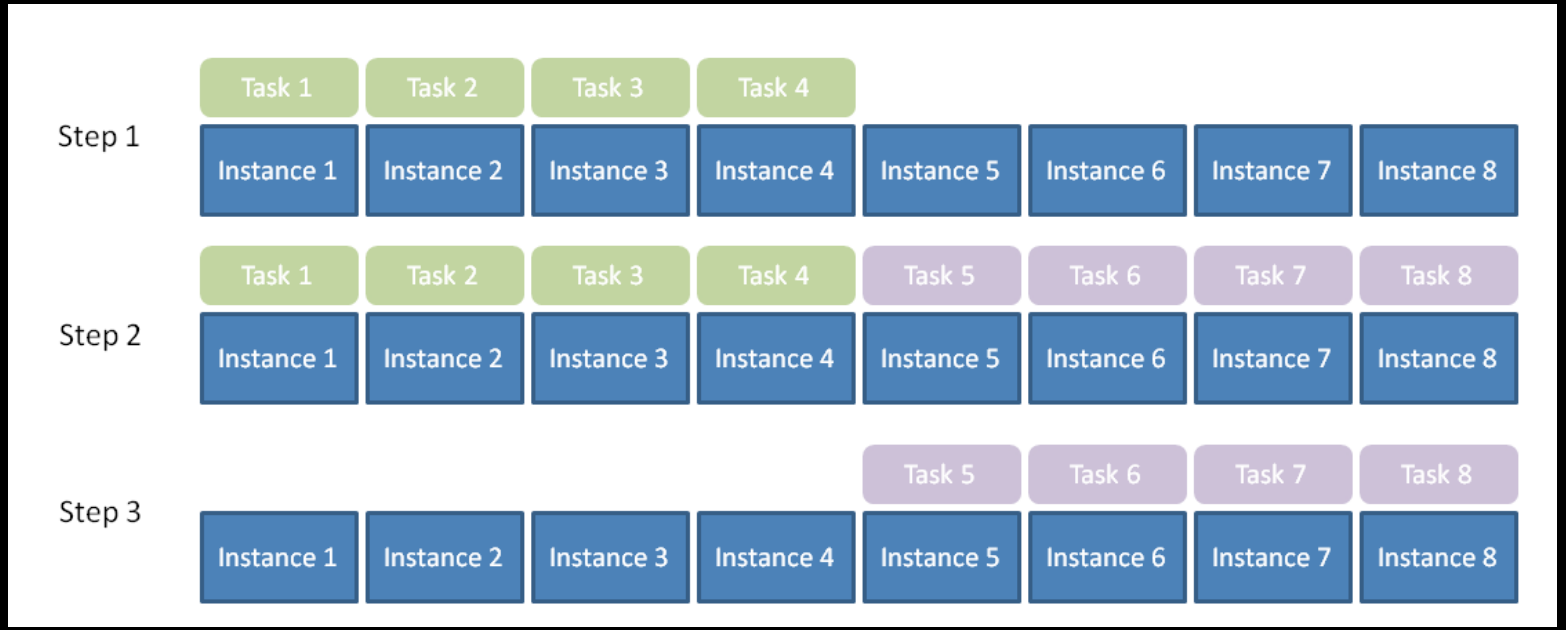
Update Service (cont.)

Deploy using the least space: *minimumHealthyPercent* = 50%, *maximumPercent* = 100%



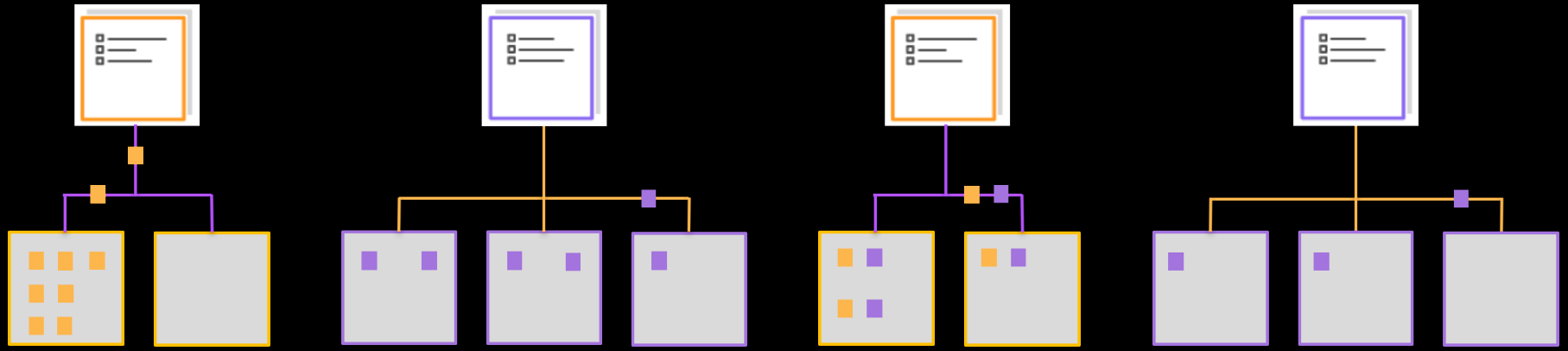
Update Service (cont.)

Deploy quickly without reducing service capacity: *minimumHealthyPercent* = 100%, *maximumPercent* = 200%



Task Placement Examples

Supported Placement Strategies



Binpacking

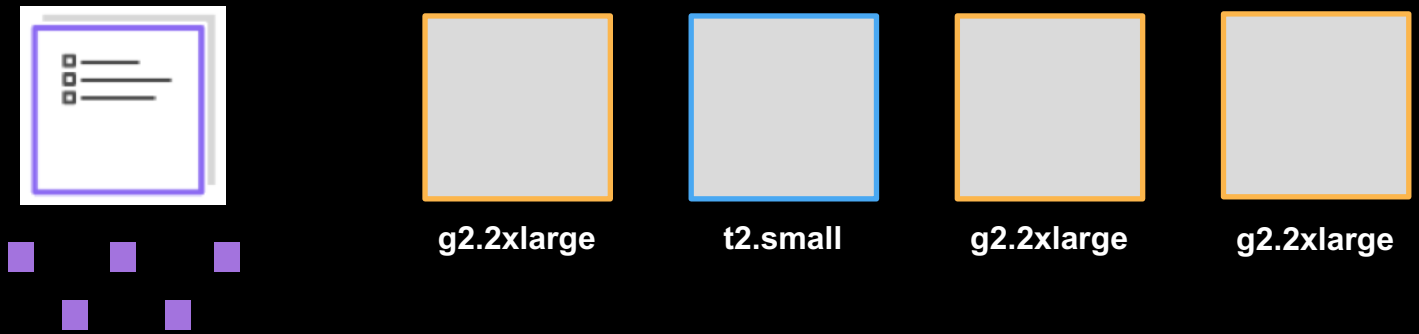
Spread

Affinity

Distinct Instance

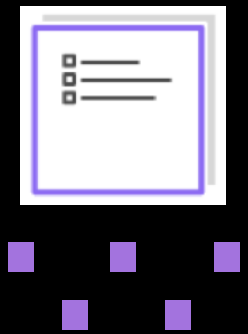
Placement: Targeting Instance Type

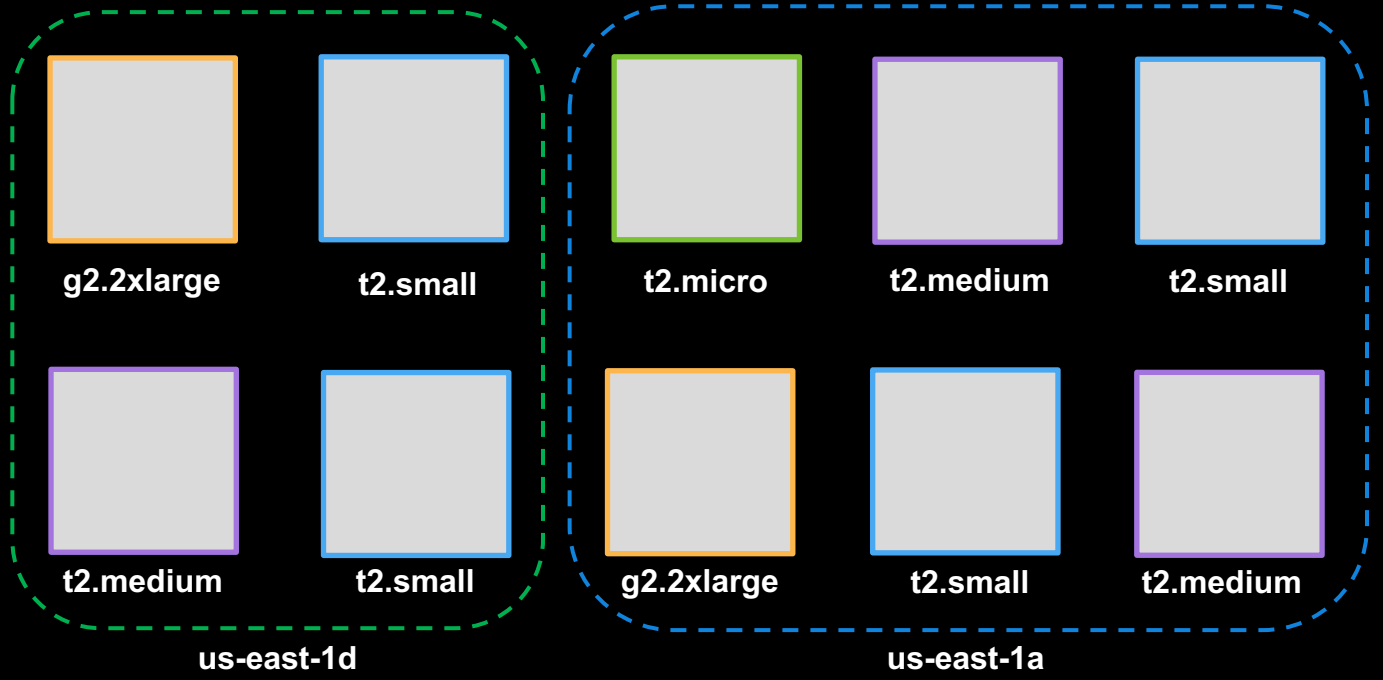
```
aws ecs run-task --cluster ecs-demo --task-definition myapp --count 5 --placement-constraints
type="memberOf",expression="attribute:ecs.instance-type == g2.2xlarge"
```



Placement: Targeting Instance Type & Zone

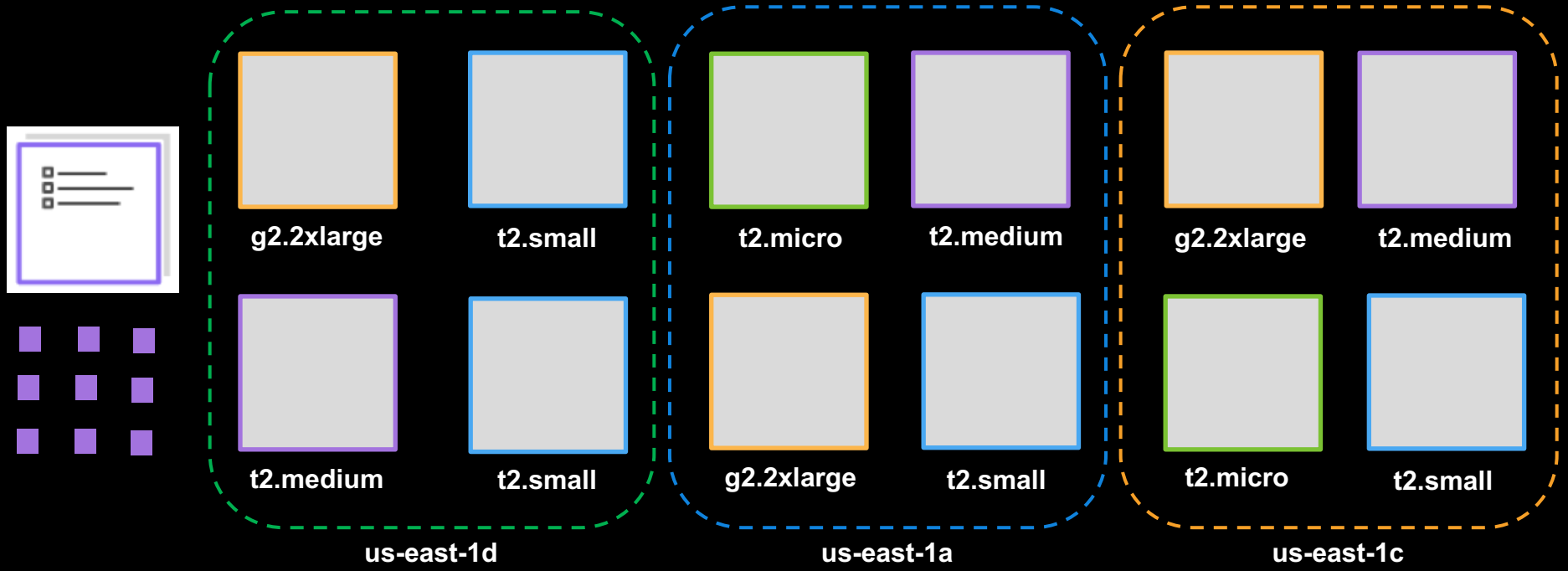
```
aws ecs run-task --cluster ecs-demo --task-definition myapp --count 5 --placement-constraints
type="memberOf",expression="(attribute:ecs.instance-type == t2.small or
attribute:ecs.instance-type == t2.medium) and attribute:ecs.availability-zone != us-east-1d"
```





Placement: Availability Zone Spread

```
aws ecs run-task --cluster ecs-demo --task-definition myapp --count 9 --placement-strategy
type="spread",field="attribute:ecs.availability-zone"
```



Security

IAM Roles

IAM roles for **container instances**:

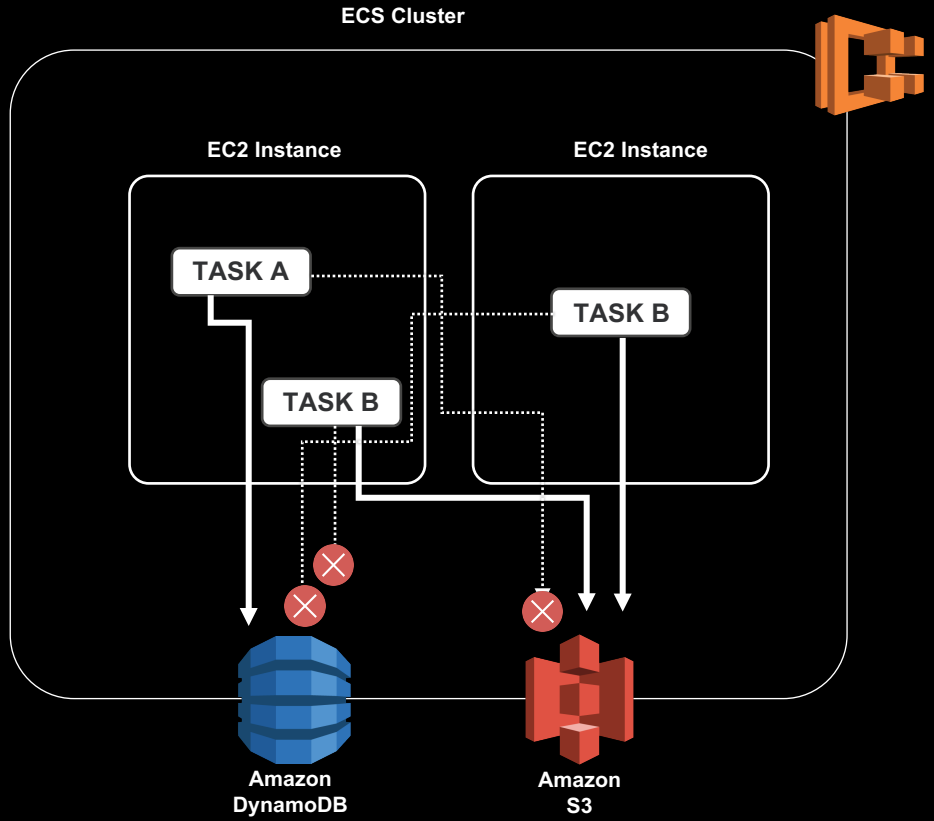
- Bound to the ECS container instance
- Applies to all containers running on the host
- Pulling images from ECR
- CloudWatch Logs

IAM roles for **tasks**:

- Bound to specific ECS tasks
- Task-specific access to AWS services

Tip Use principle of least privilege – prefer **IAM roles for tasks** where applicable

IAM Roles For Tasks



Monitoring & Logging

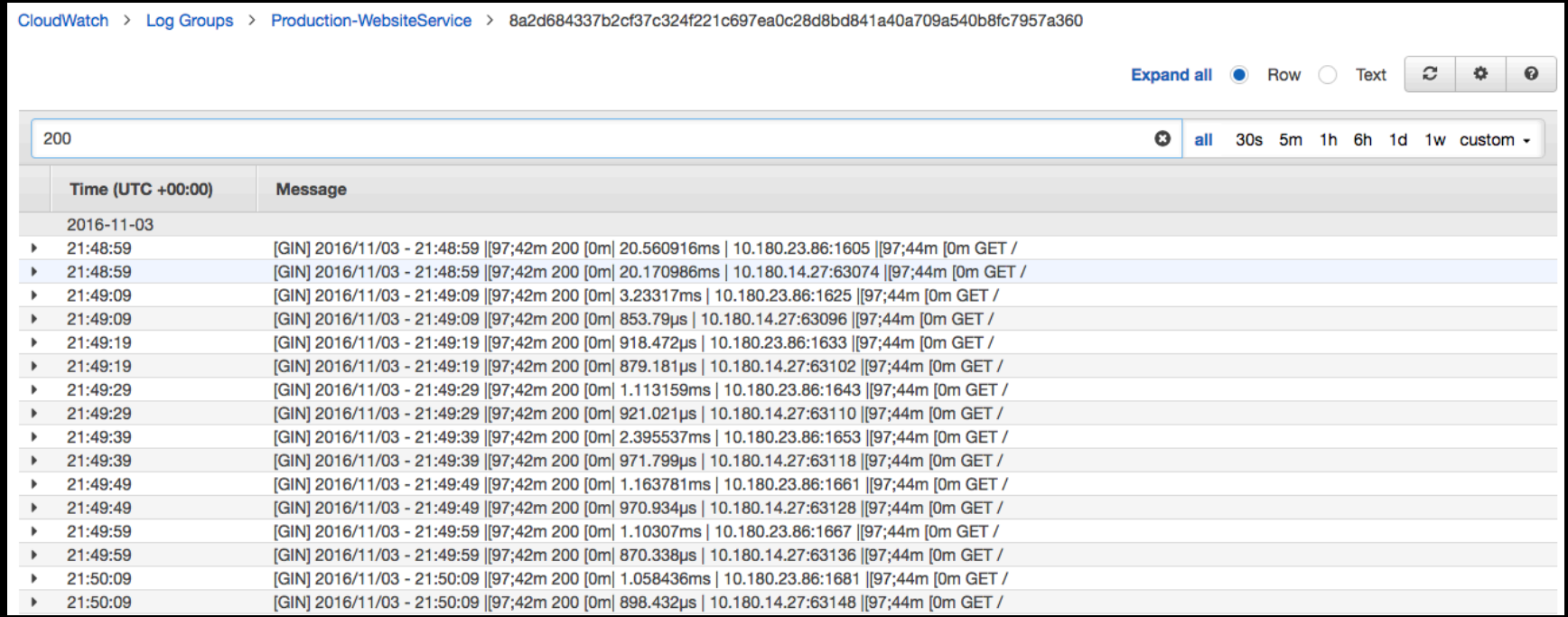
Monitoring with CloudWatch

Centralized Logging with CloudWatch Logs

```
{
  "image": ”nginx:latest",
  ...
  "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
      "awslogs-group": ”nginx",
      "awslogs-region": "us-east-1"
    }
  }
}
```

- Defined within the task definition
- Available log drivers
  - awslogs
  - fluentd
  - gelf
  - journald
  - json-file
  - splunk
  - Syslog
- Submit a pull request on ECS agent GitHub repo if you would like others

Centralized Logging with CloudWatch Logs



Demo

AWS Customers using ECS in production

14	3500	500 million
Data Processing Services	Peak Container Instances	Compute Hours Used This Year

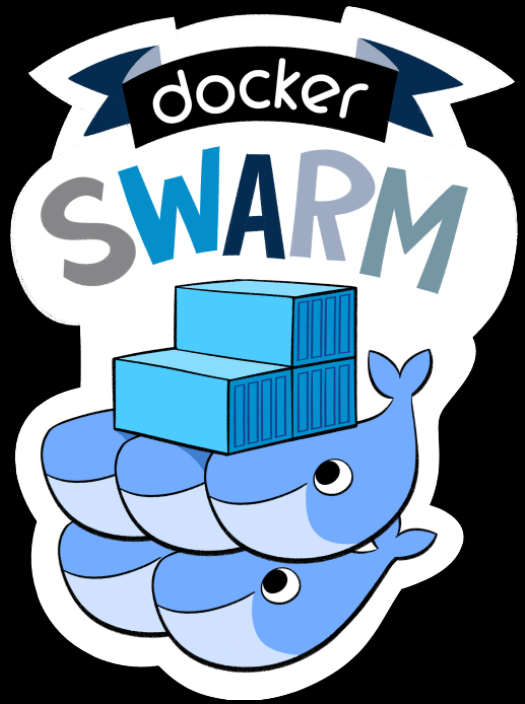
- 50 Microservices - 10 containers/service
- Costs down 40% by using Spot Instances
- Faster Deployment Times

“ In November 2015 we moved our Docker container architecture to Amazon ECS, and for the first time ever in December we were able to celebrate a new year in which our system could handle the huge number of requests without any crashes or interruptions—an accomplishment that we were extremely proud of. We had faced the biggest night on the calendar without any downtime.”

Docker on AWS – just ECS?

kubernetes

MESOS



AWS & CNCF

Questions?

Thank you

vladsim@amazon.com