
AWS
re:Invent

CON212

# Securing your Amazon ECS applications: Best practices

Tori Hara

Sr. Developer Advocate, Containers

AWS

AWS re:Invent

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

# Agenda

- Amazon Elastic Container Service (Amazon ECS) overview
- Securing your Amazon ECS tasks
- Operating your Amazon ECS tasks securely
- Takeaways





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

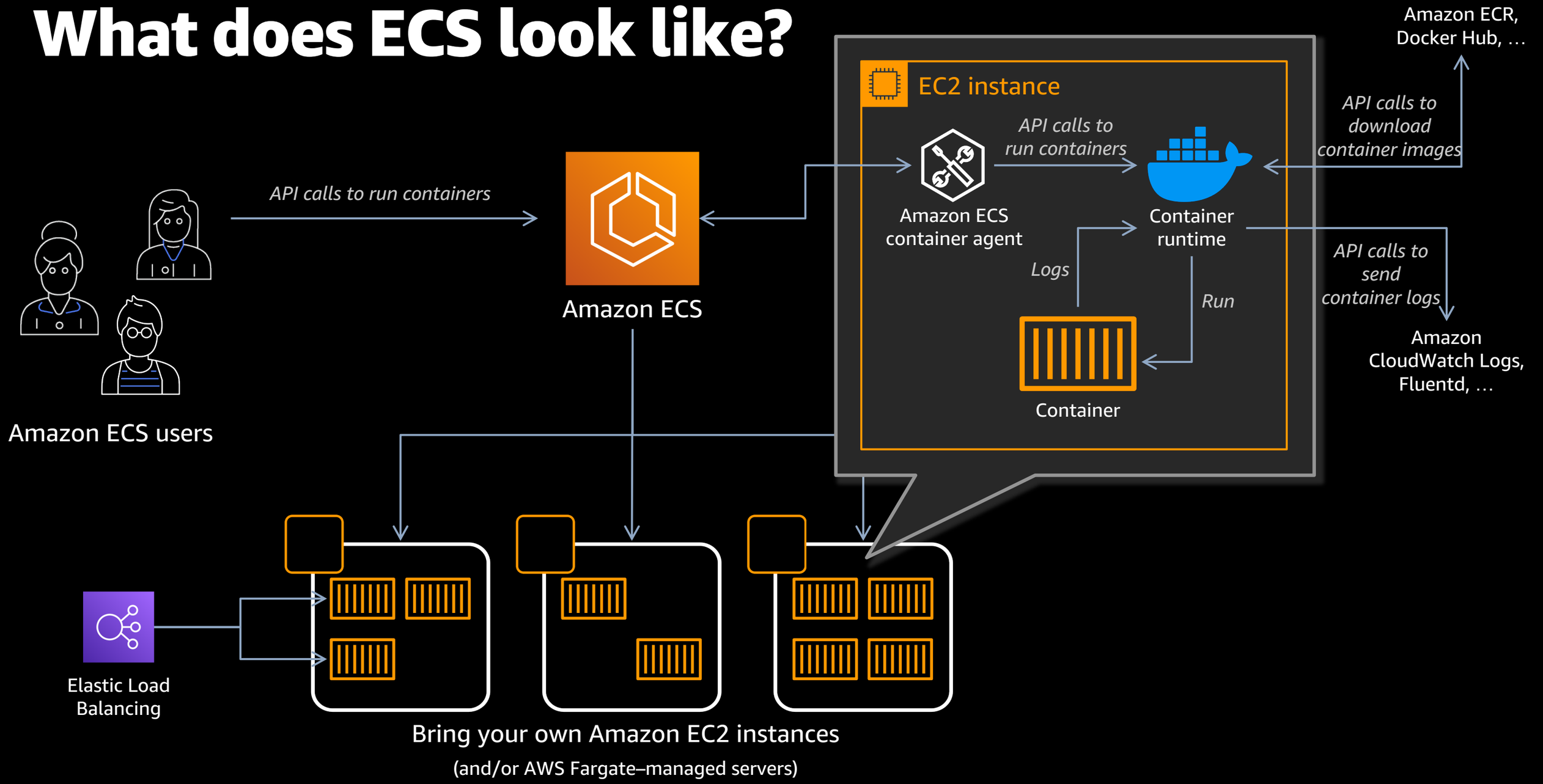
# Amazon ECS overview



AWS re:Invent

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

# What does ECS look like?



# Amazon ECS constructs: Task and service



## Amazon ECS task

Defined by a “task definition”

A group of 1 or more containers

The smallest deployable unit

Configure networking, storage, parameters, IAM roles, and compute resources

Similar to the *docker run* command on your local machine



## Amazon ECS service

Defined by a “service definition”

Enables you to run and maintain a required number of Amazon ECS tasks

Automatically handles Amazon ECS task failures by replacing them with new Amazon ECS tasks

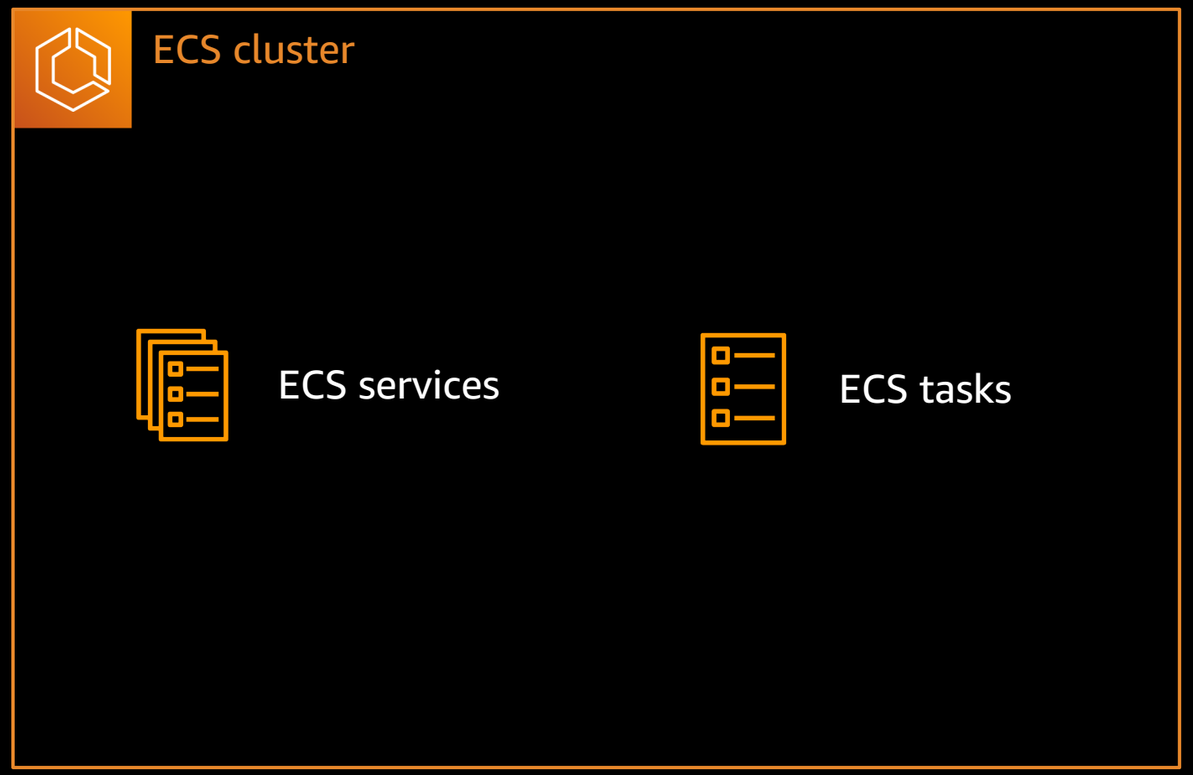
Configure Amazon VPC networking, deployment type, optional load balancers, and service discovery

# Amazon ECS constructs: Cluster

A logical grouping of Amazon ECS tasks or services in a region

Multiple clusters can be created

Clusters / services / tasks work as IAM permission boundaries



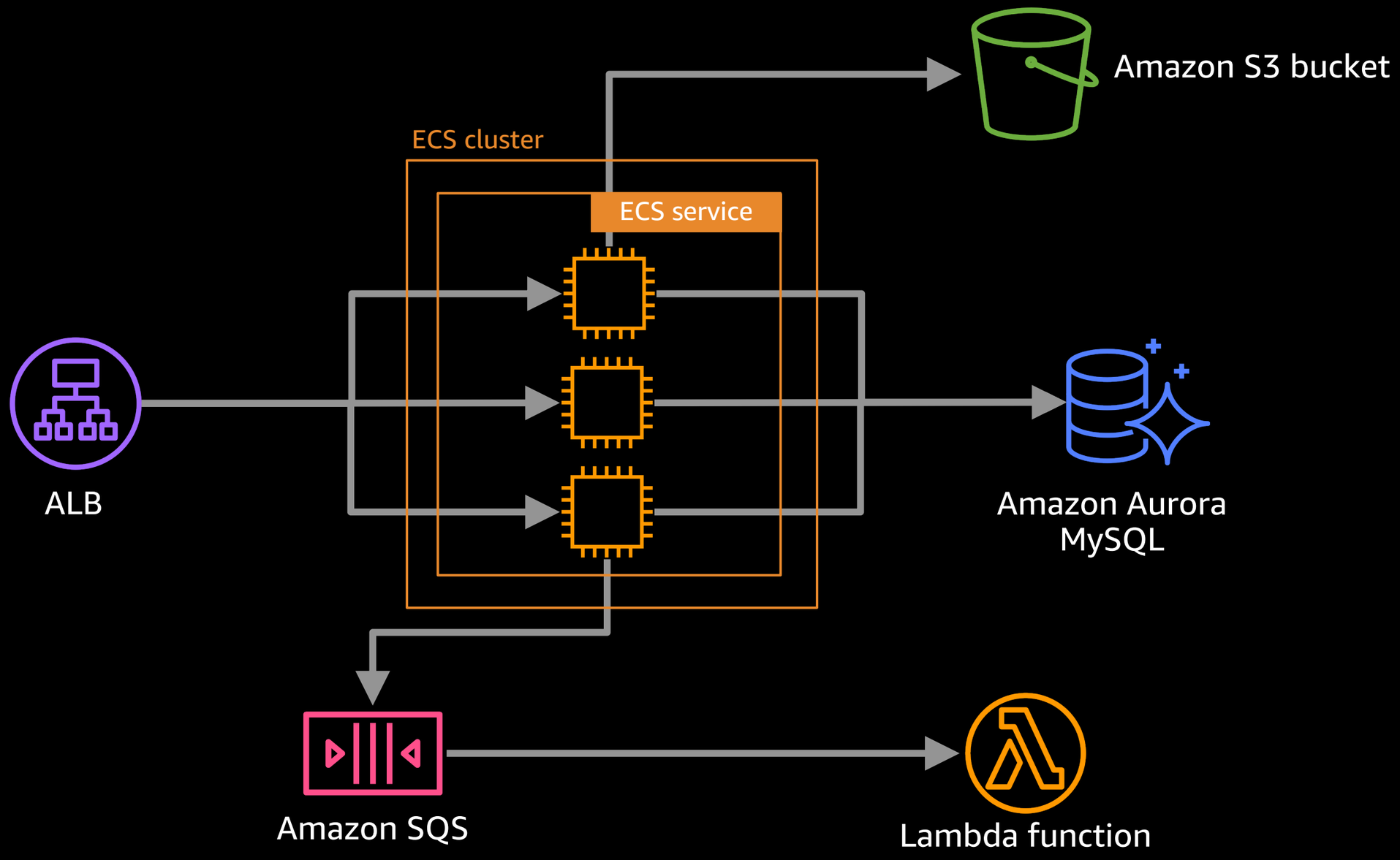
# Securing your Amazon ECS tasks



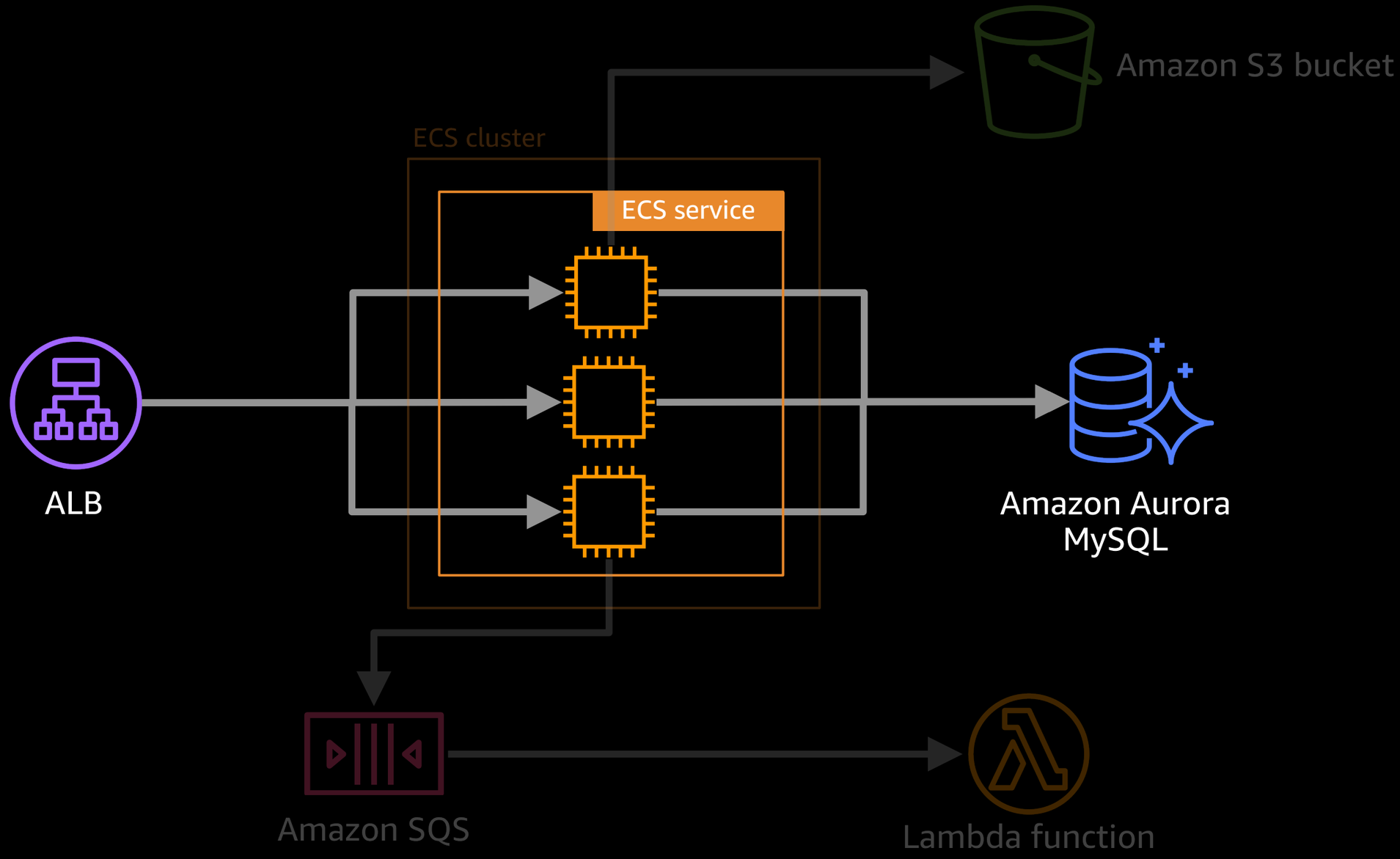
AWS re:Invent

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

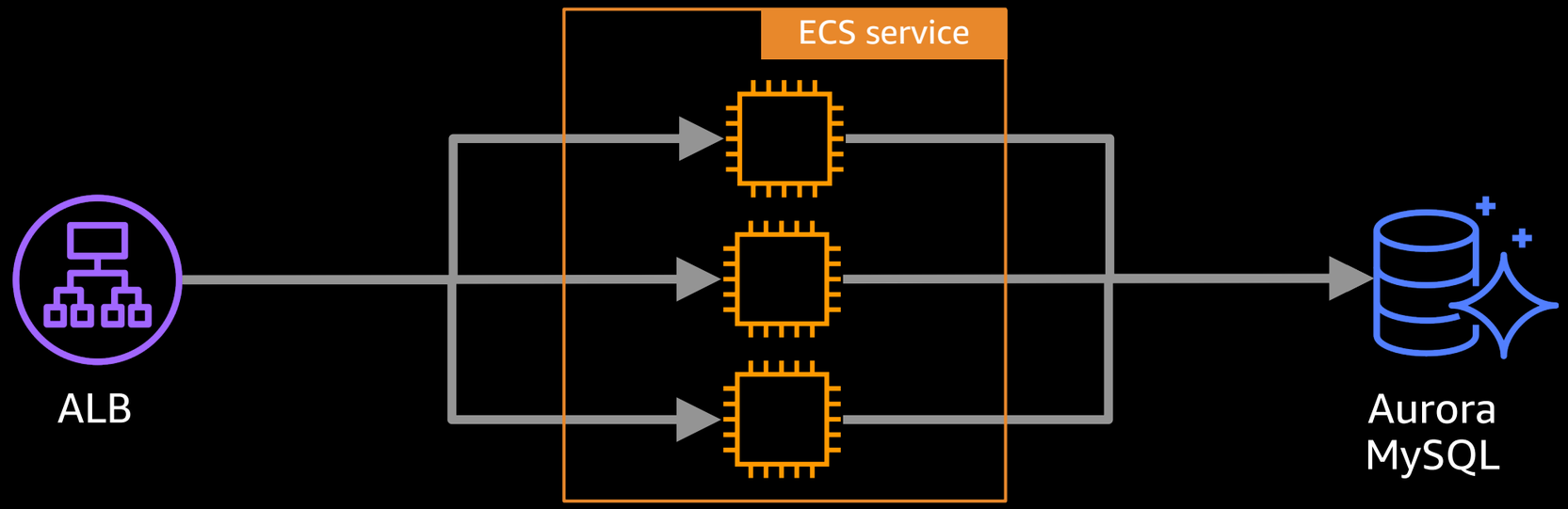
# Meet your Amazon ECS app



# Step 1: Security groups

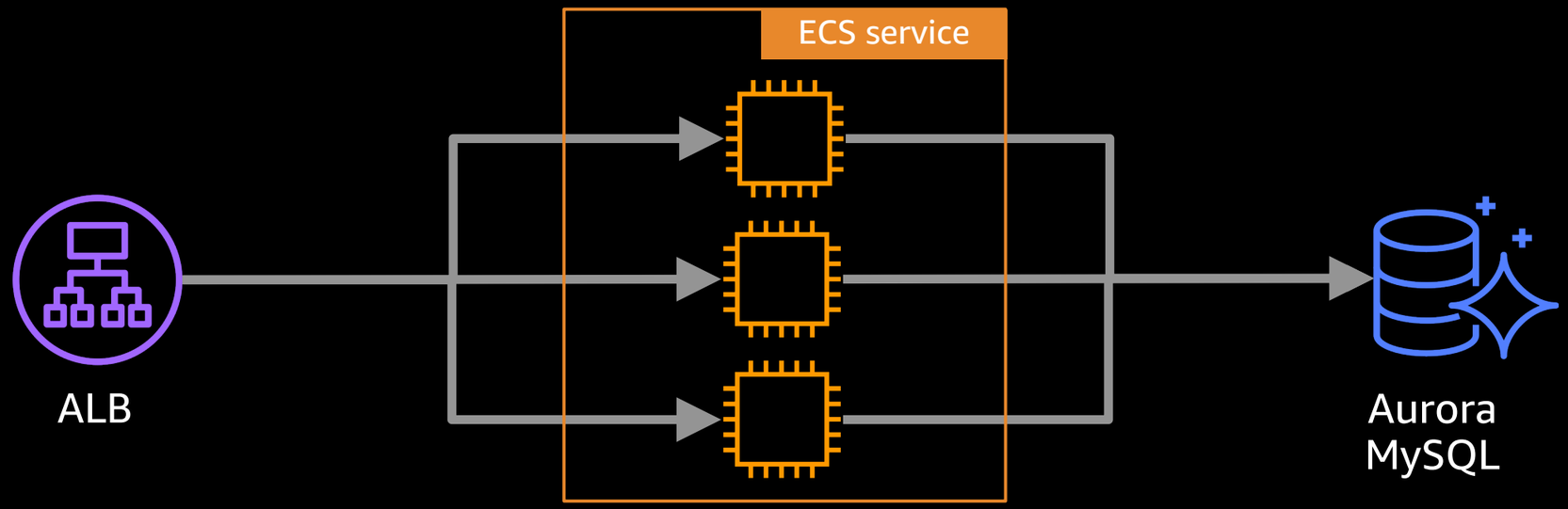


# Step 1: Security groups



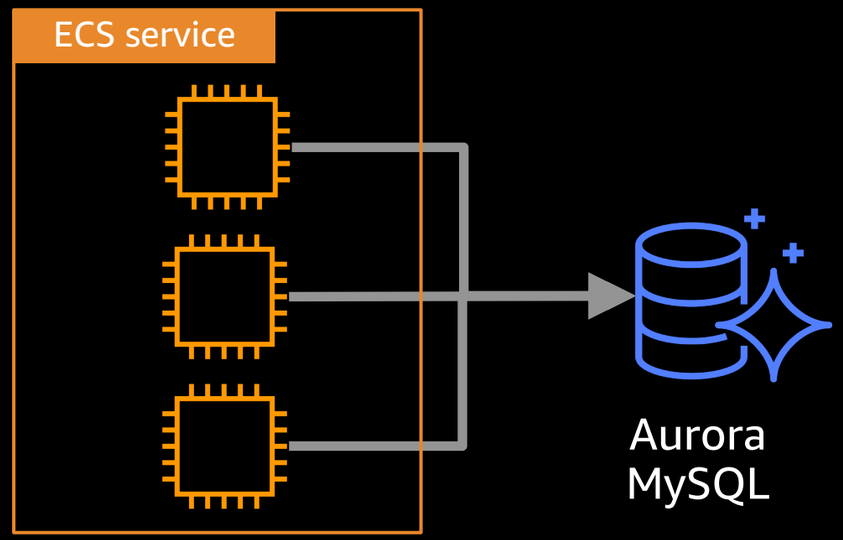

| | sg-ALB | sg-ECSTask | sg-AuroraDB |
|---|---|---|---|
| Allowed type and port | HTTPS: 443 | HTTP: 80 | MySQL/Aurora: 3306 |
| Allowed source | 0.0.0.0/0 (Public access) | 10.0.0.0/16 (From within the VPC) | 10.0.0.0/16 (From within the VPC) |

# Step 1: Security groups




| | sg-ALB | sg-ECSTask | sg-AuroraDB |
|---|---|---|---|
| Allowed type and port | HTTPS: 443 | HTTP: 80 | MySQL/Aurora: 3306 |
| Allowed source | 0.0.0.0/0 (Public access) | sg-ALB (Only from the ALB) | sg-ECSTask (Only from the ECS tasks) |

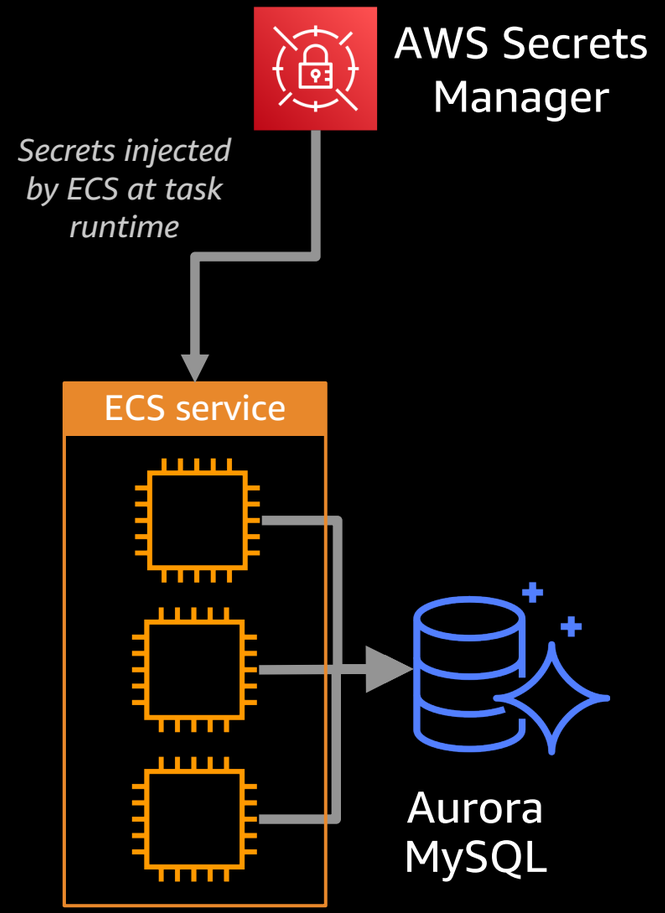
# Step 2: Secrets




ECS task definition

```
{
   ... snip ...
   "containerDefinitions":[{
      "name":"my-web-app",
      "image":"my-web-app:v1",
         ... snip ...
      "environment": [{
         "name": "DB_USER",
         "value": "myappdbuser"
      },{
         "name": "DB_PASSWD",
         "value": "mysupersecretpasswd"
      },{
         "name": "DB_HOST",
         "value": "my-db...rds.amazonaws.com"
      },{
         "name": "DB_PORT",
         "value": "3306"
      }],
         ... snip ...
   }],
   ... snip ...
}
```

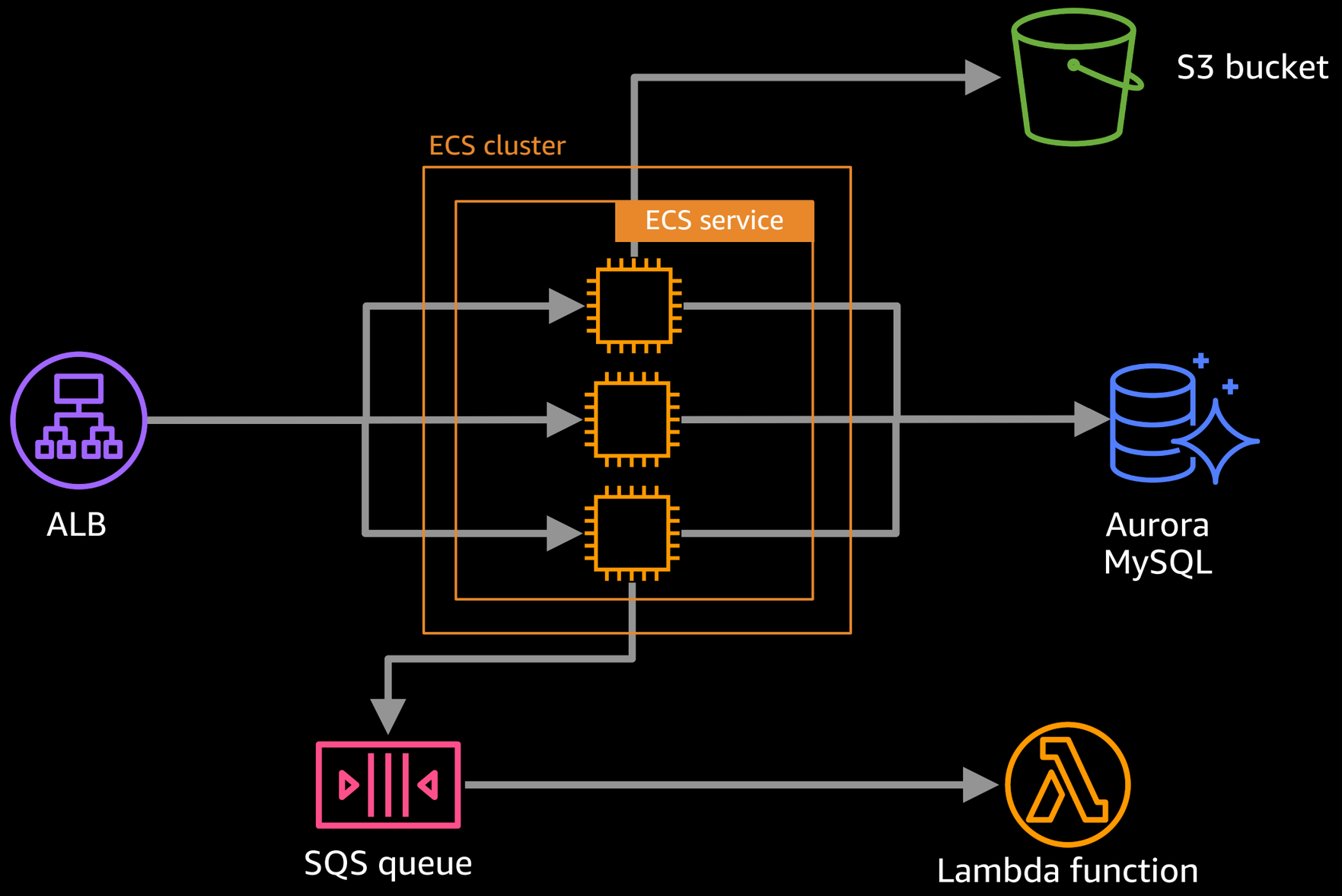
# Step 2: Secrets



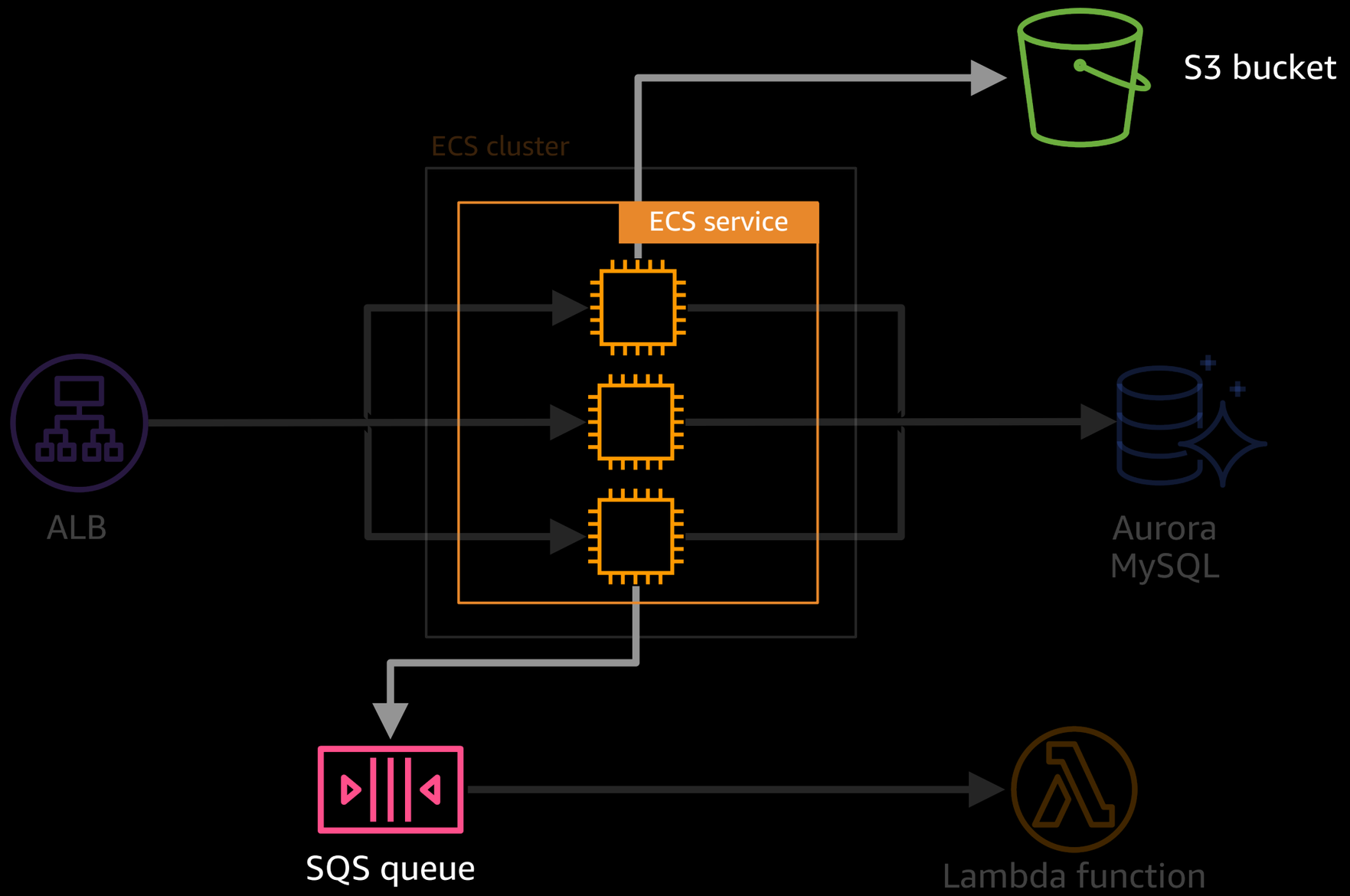

ECS task definition

```
{
  ... snip ...
  "containerDefinitions":[{
    "name":"my-app",
    "image":"my-web-app:v1",
      ... snip ...
    "secrets": [{
       "name": "DB_USER",
       "valueFrom": ”...secretsmanager:...:my-db-secret:username::"
    },{
       "name": "DB_PASSWD",
       "valueFrom": ”...secretsmanager:...:my-db-secret:password::"
    }],
    "environment": [{
       "name": "DB_HOST",
       "value": ”my-db...rds.amazonaws.com"
    },{
       "name": "DB_PORT",
       "value": ”3306"
    }]
      ... snip ...
}
```

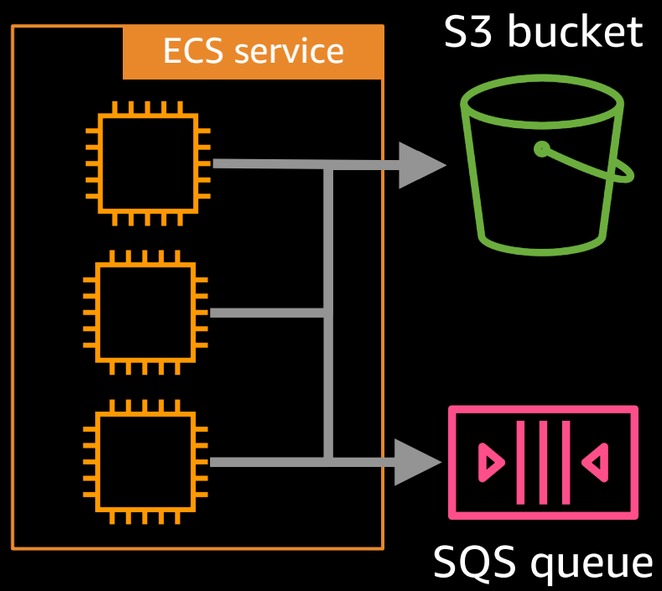
# Meet your Amazon ECS app (again)



# Step 3: IAM roles



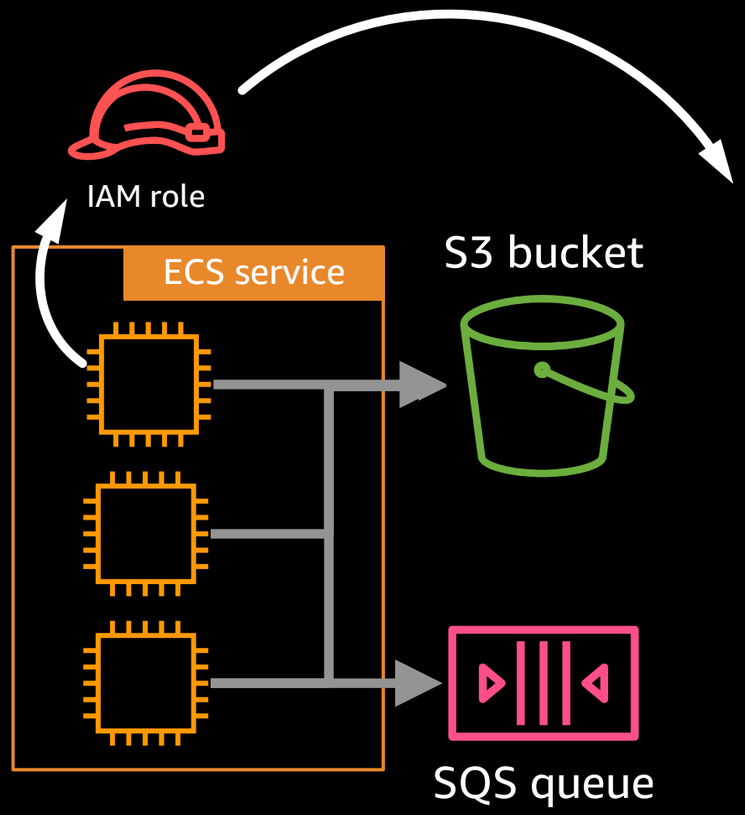
# Step 3: ECS task-level IAM roles




ECS task definition

```
{
   ... snip ...
   "taskRoleArn": "",
   "executionRoleArn": "",
   "containerDefinitions":[{
      ... snip ...
   }],
   ... snip ...
}
```

# Step 3: EC2-level IAM role

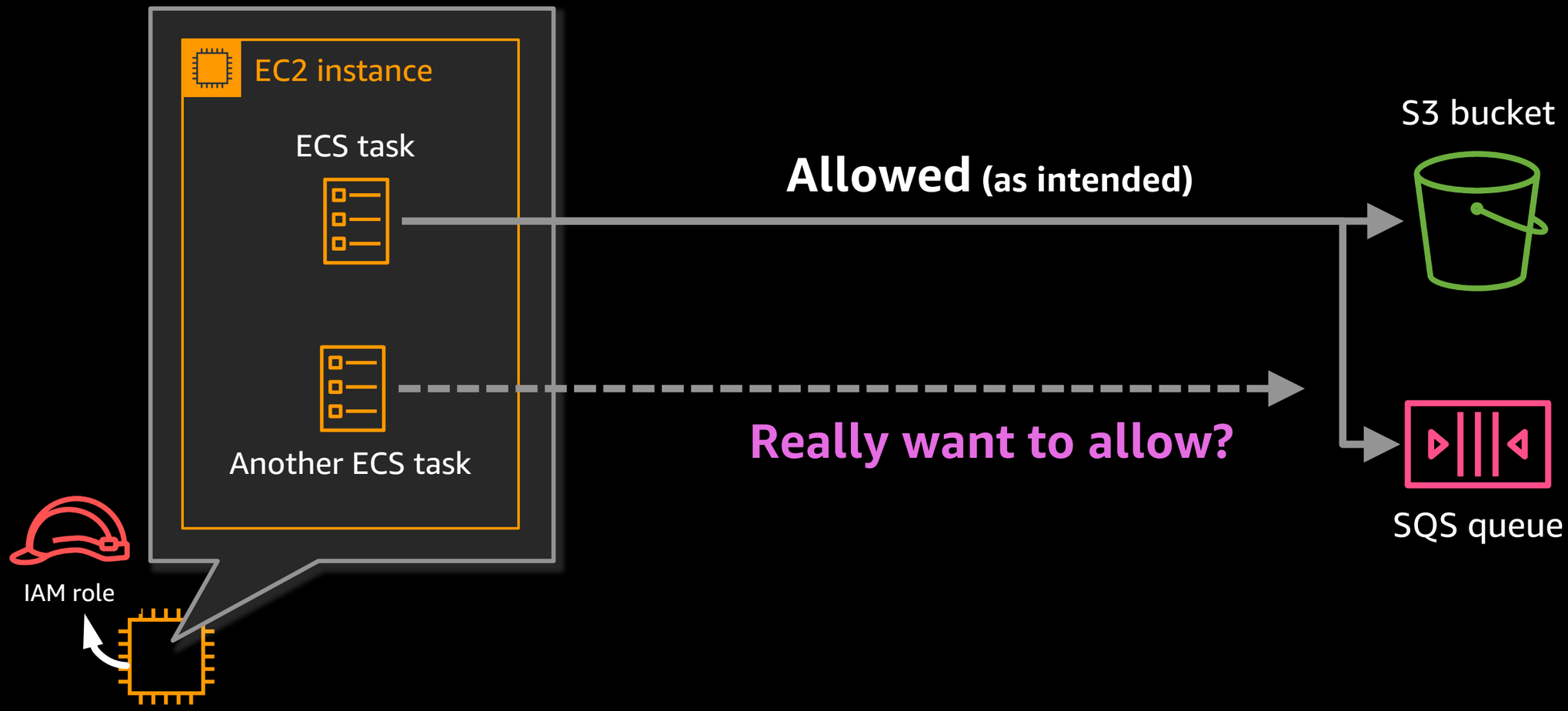



EC2 instance IAM role's policies

```
{
  "AttachedPolicies": [{
    "PolicyName": "AmazonS3FullAccess",
    "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
  },
  {
    "PolicyName": "AmazonSQSFullAccess",
    "PolicyArn": "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
  },
  {
    "PolicyName": "AmazonEC2ContainerServiceforEC2Role",
    "PolicyArn": "arn:aws:iam::aws:policy/service-
role/AmazonEC2ContainerServiceforEC2Role"
  }]
}
```

# Step 3: EC2-level IAM role

All ECS tasks on the same EC2 instance can use the EC2 instance role



# Step 3: IAM roles for tasks

S3 write-only policy

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:ListBucket"],
    "Resource": ["arn:aws:s3:::your-bucket"]
  },{
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:DeleteObject"
    ],
    "Resource": ["arn:aws:s3:::your-bucket/*"]
  }]
}
```

SQS write-only policy

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sqs:SendMessage",
    "Resource": "arn:aws:sqs:...:your-queue-name"
  }]
}
```

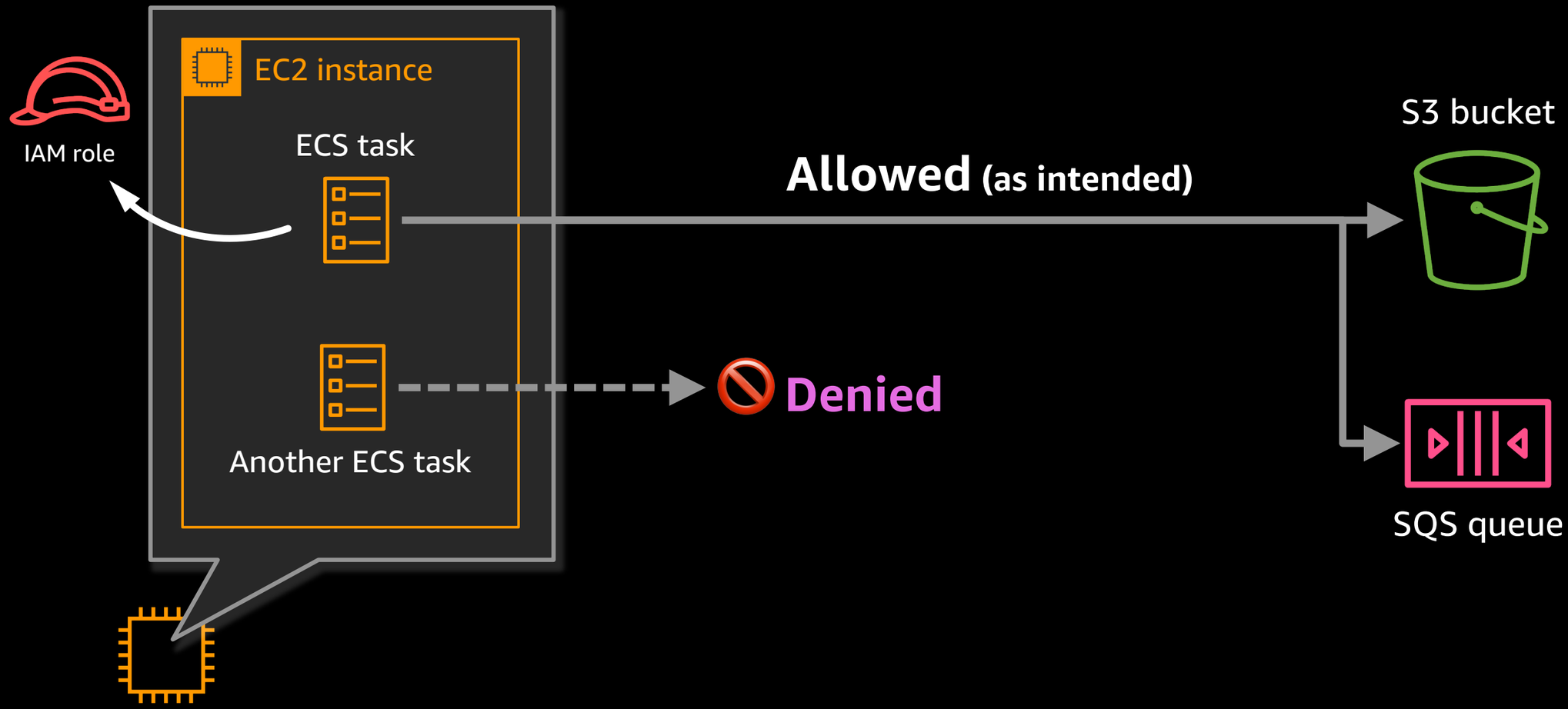
ECS task role

```
{
  "AttachedPolicies": [{
    "PolicyName": "S3WriteOnlyPolicy",
    "PolicyArn": "S3WriteOnlyPolicy-ARN-here"
  },{
    "PolicyName": "SQSWriteOnlyPolicy",
    "PolicyArn": "SQSWriteOnlyPolicy-ARN-here"
  }]
}
```

ECS task definition

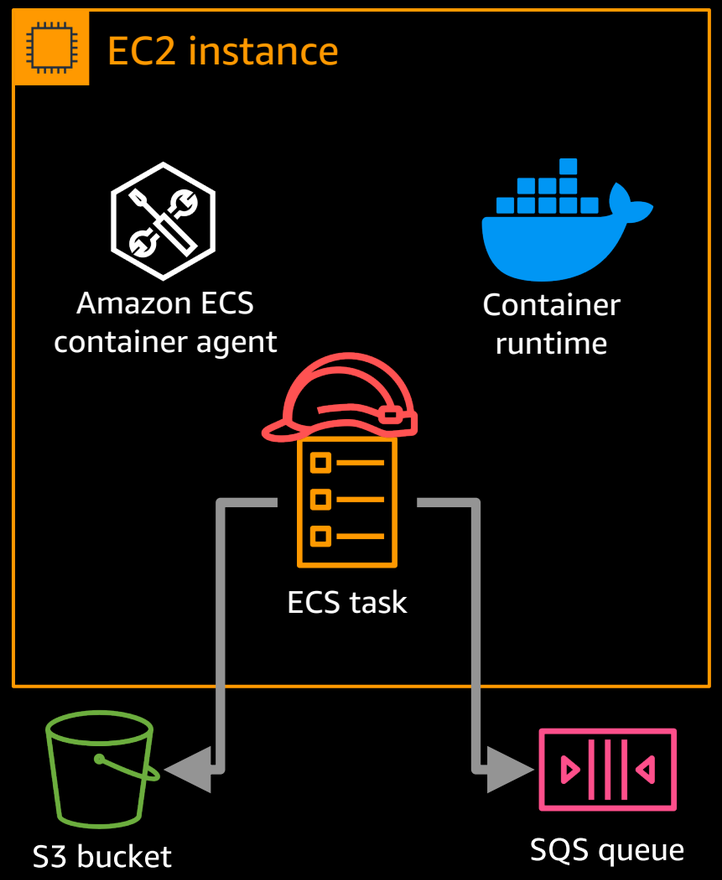
```
{
    ... snip ...
    "taskRoleArn": ”TaskRole-ARN-here",
    "executionRoleArn": "",
    "containerDefinitions":[{
        ... snip ...
    }],
    ... snip ...
}
```

# Step 3: IAM roles for tasks

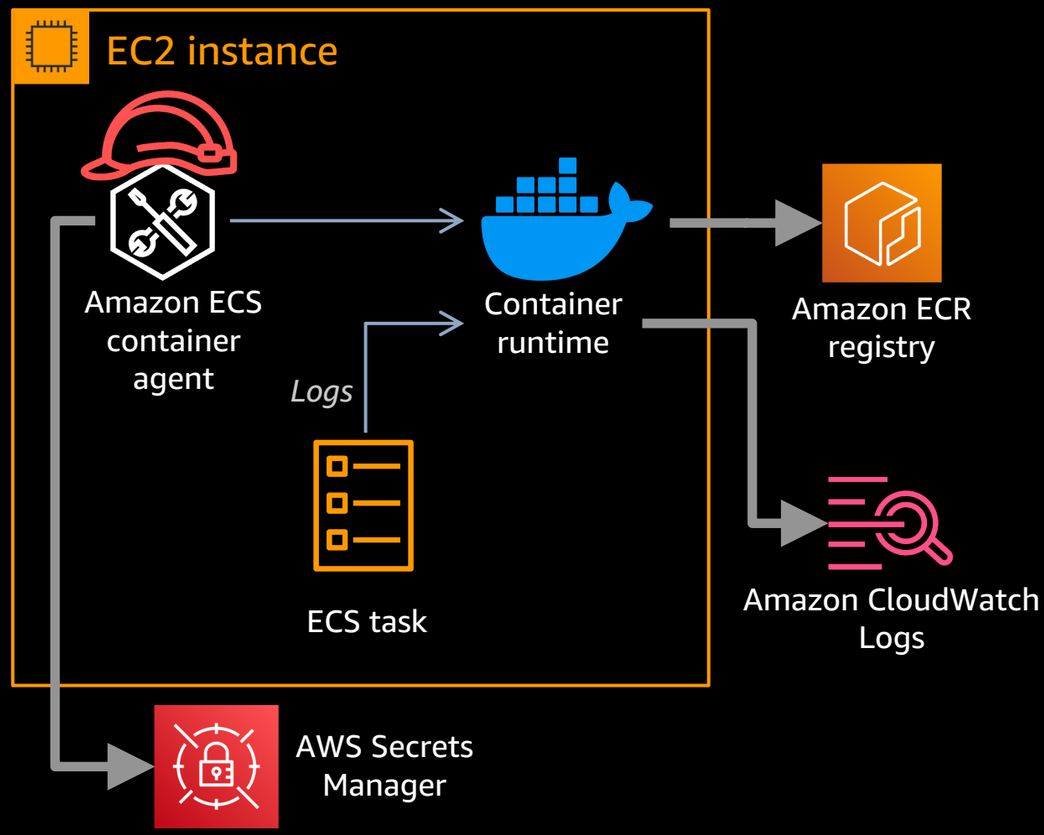


# Step 3: “Task Execution Role”?

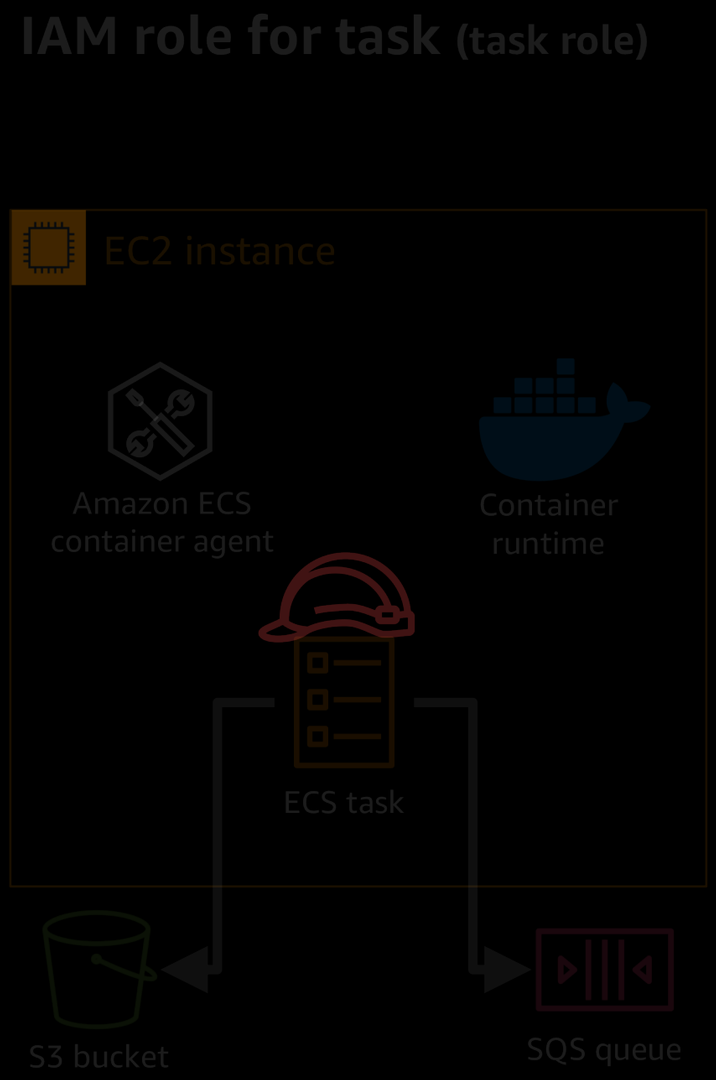
## IAM role for task (task role)

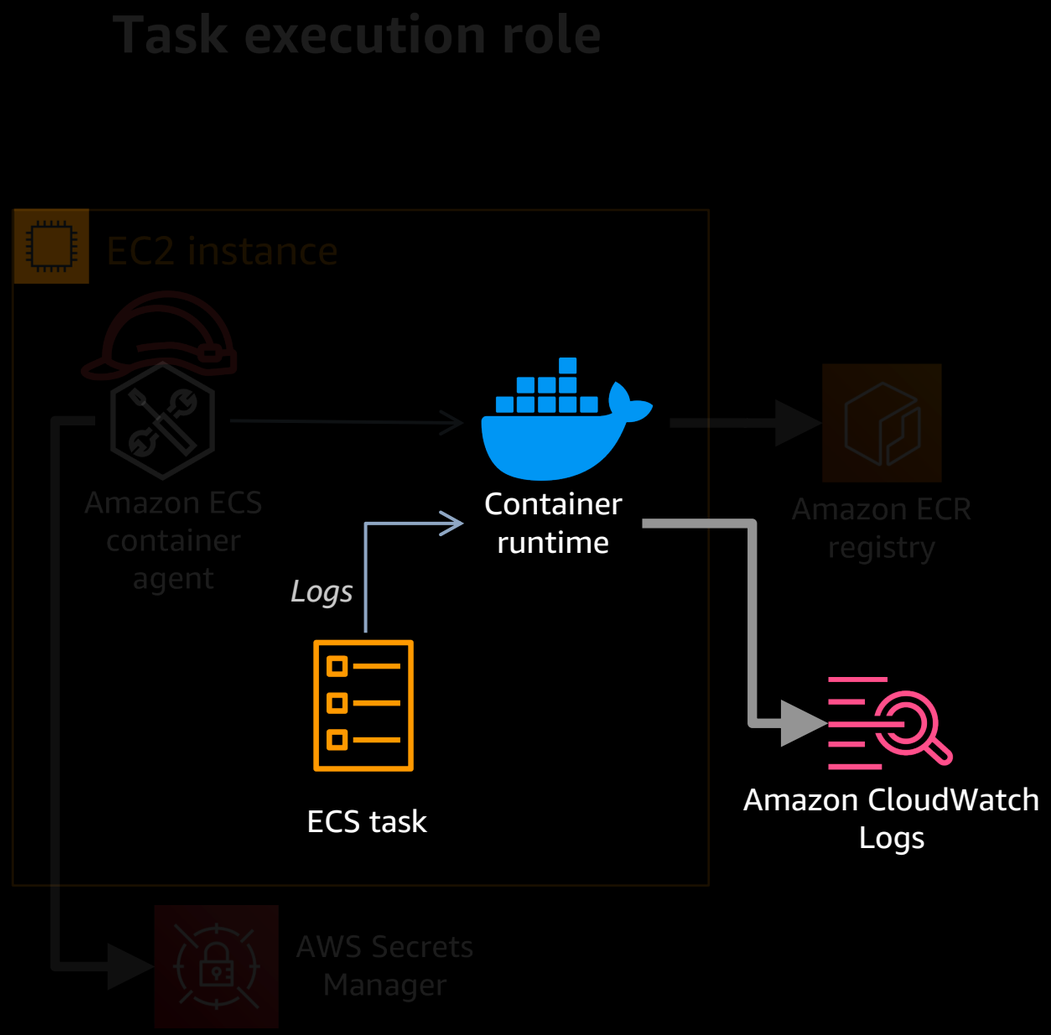



## Task execution role

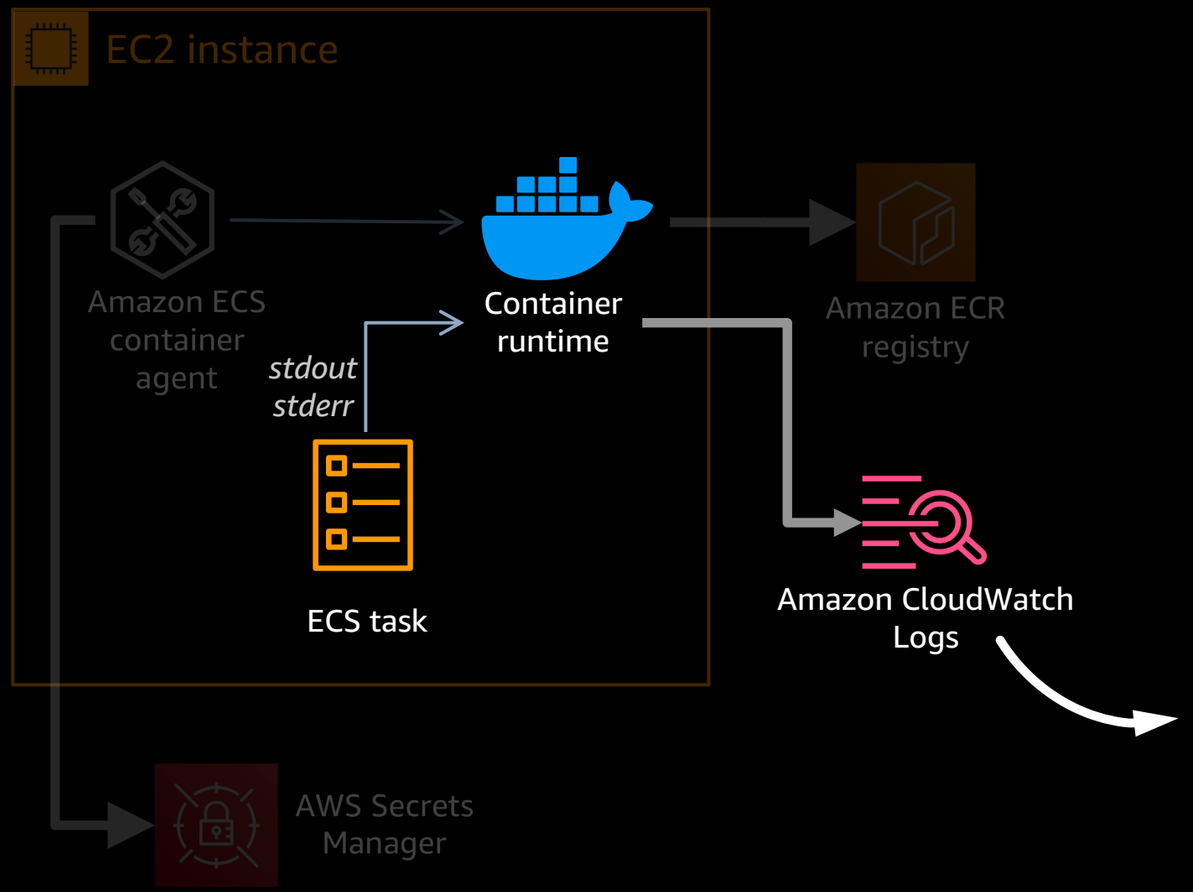


# Step 4: Logs from apps



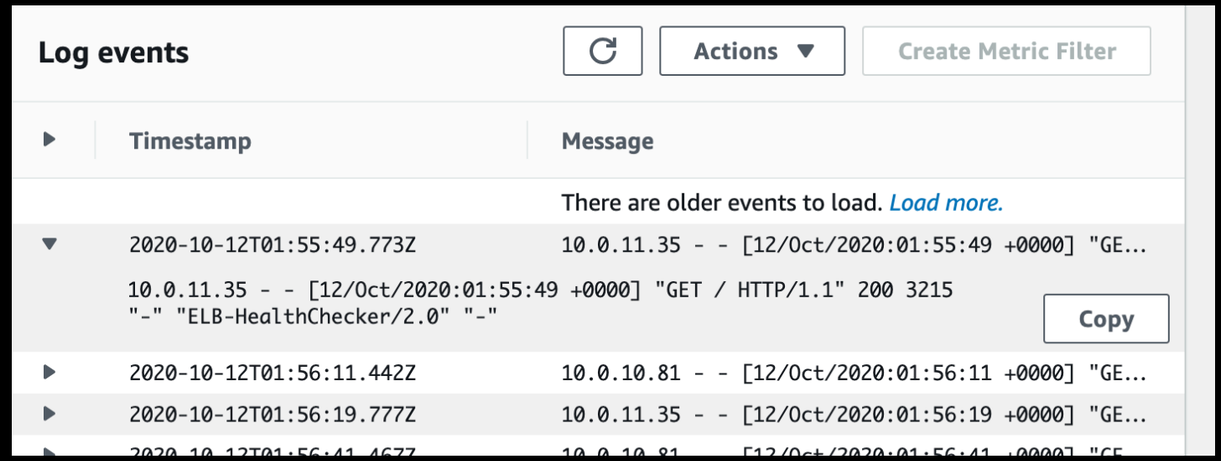



# Step 4: Logs from apps

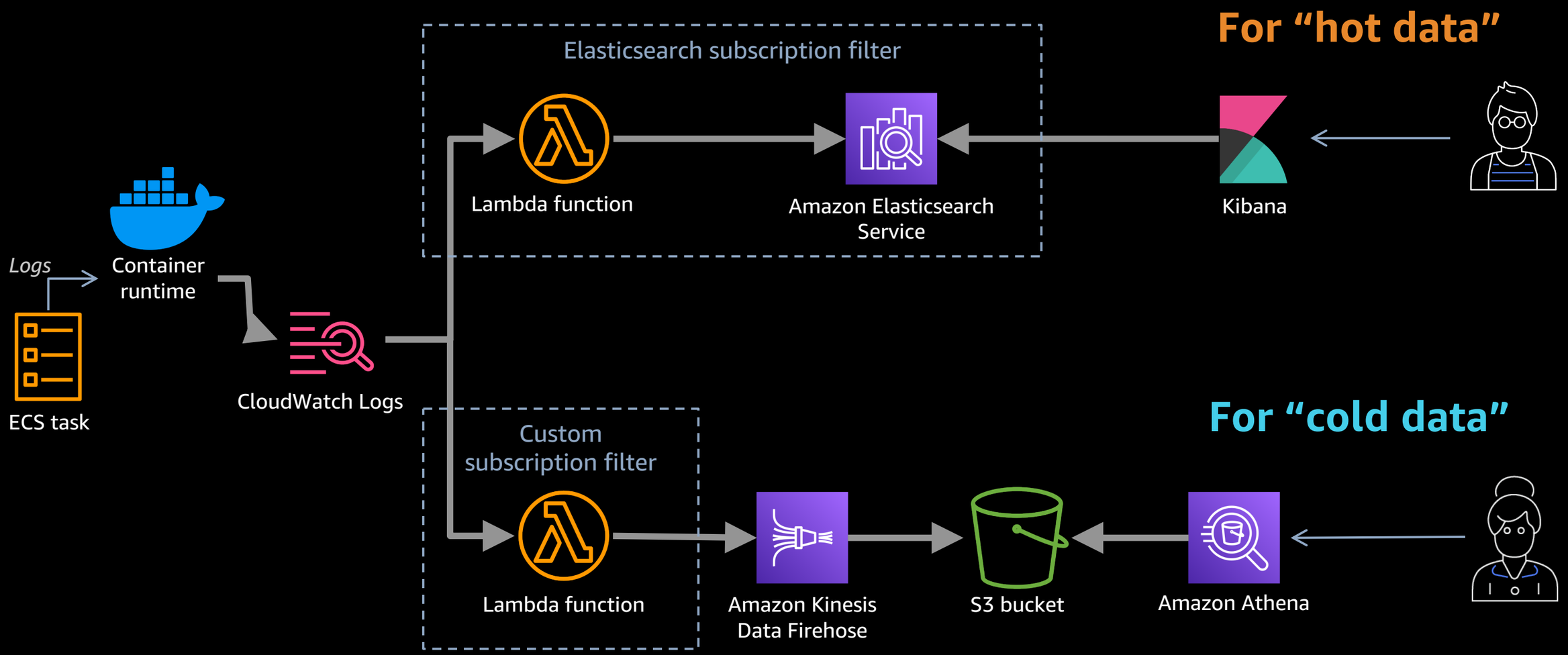



ECS task definition

```
{
  ... snip ...
  "containerDefinitions":[{
    "name":"my-web-app",
    "image":"my-web-app:v1",
      ... snip ...
    "logConfiguration": {
      "logDriver": "awslogs",
      "options": {
        "awslogs-group": "/ecs/my-web-app",
        "awslogs-region": "us-west-2",
        "awslogs-stream-prefix": "ecs"
    }}
  }], ... snip ...
}
```



# Step 4: Logs from apps

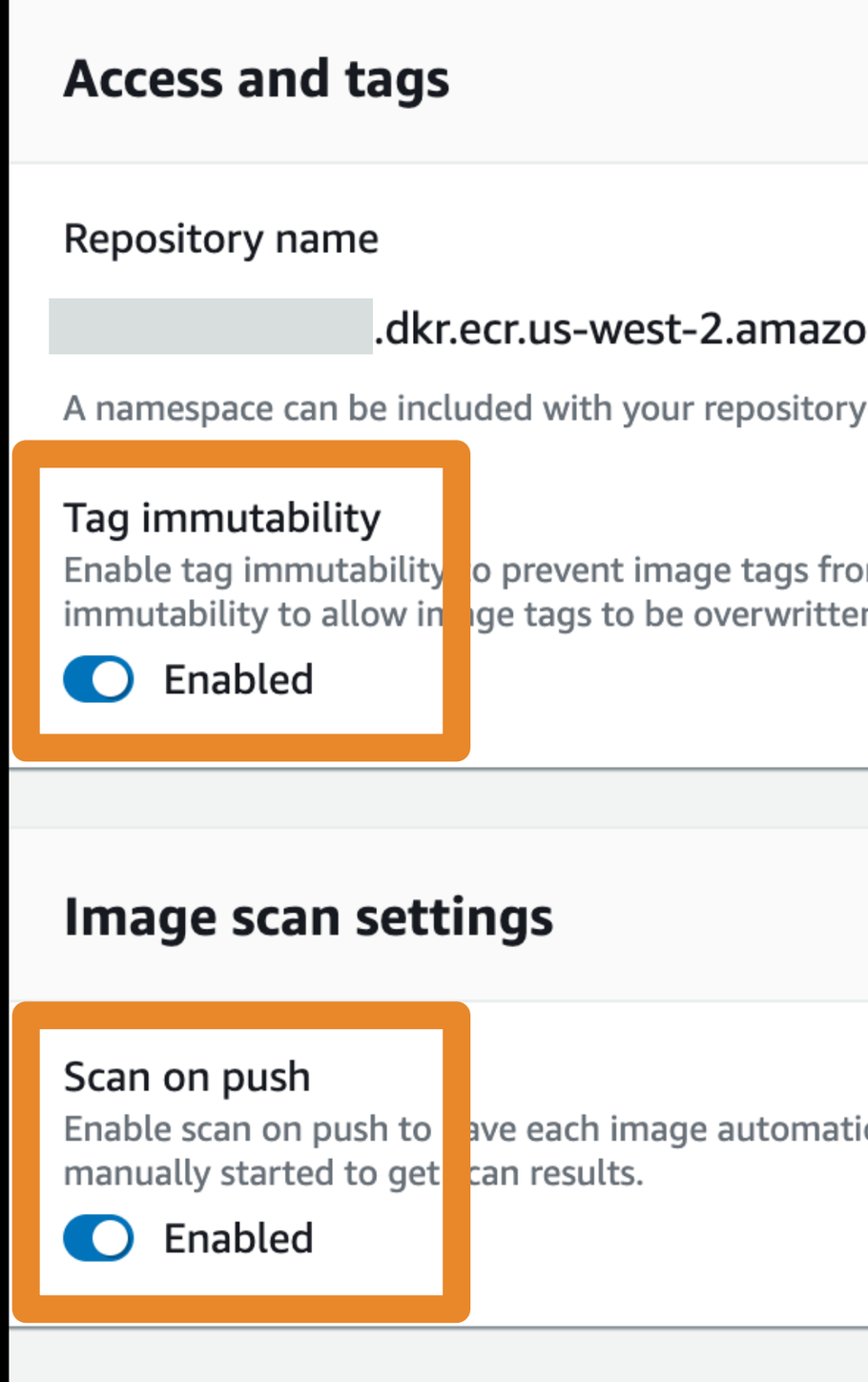



See the Amazon ECS feature “FireLens” for more advanced log handling and routing:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/using_firelens.html

# Step 5: Enabling ECR features

1. Immutable tags
   - Image tags can be overwritten by push
   - Use immutable tags to use expected container images at any given time

2. Image scanning
   - “Scan on push” for automatic scanning
   - Scheduled rescanning is also recommended to handle CVEs after image creation; see also https://aws.amazon.com/blogs/containers/amazon-ecr-native-container-image-scanning/



# Securing your Amazon ECS app



**Container images**

- ECR immutable tags and image scanning



**Containers**

- No privileged mode
- Non–root user
- Read-only file system



**ECS tasks**

- Security groups
- Secrets
- Task role and task execution role
- Container logs

See also “Best practices for writing Dockerfiles” by Docker, Inc.:
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

# Operating your Amazon ECS tasks securely



AWS re:Invent

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

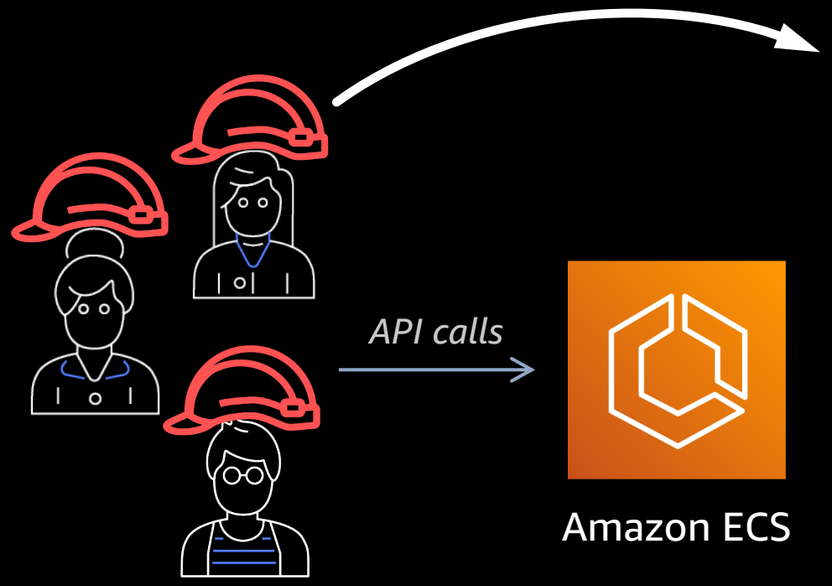
# Operating your Amazon ECS tasks securely

1. Access-control for Amazon ECS resources

2. Auditing AWS API calls

3. Validating AWS resources to keep everything compliant

# Step 6: Access control

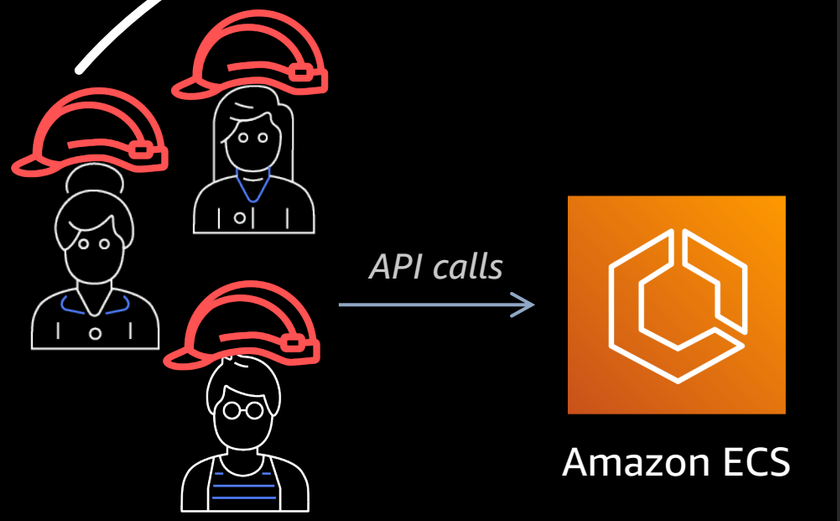



**e.g., Grant read-only access to specific cluster**

IAM policy

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": ”ReadOnlyAccess",
    "Effect": "Allow",
    "Action": [
      "ecs:DescribeCluster”
    ],
    "Resource": "arn:aws:ecs:region:account-id:cluster/our-cluster"
  }]
}
```

# Step 6: Access control

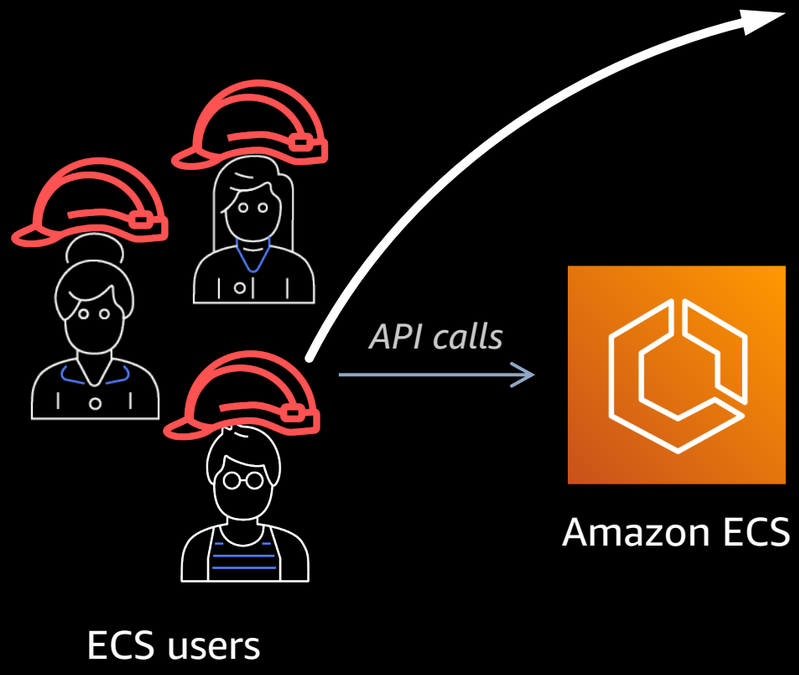



**e.g., Grant read-only access to specific cluster resources**

IAM policy

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": ”ReadOnlyClusterResources",
    "Effect": "Allow",
    "Action": [
      "ecs:Describe*",
      "ecs:List*"
    ],
    "Condition": {
      "ArnEquals": {
        "ecs:cluster": "arn:aws:ecs:region:account-id:cluster/our-cluster"
      }
    },
    "Resource": "*"
  }]
}
```

# Step 6: Access control




**e.g., Grant full access to Amazon ECS resources**

**only if resource "Owner" tag and user "Team" tag values match**

IAM Policy

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "ECSFullAccessIfOwner",
    "Effect": "Allow",
    "Action": "ecs:*",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "ecs:ResourceTag/Owner": "${aws:PrincipalTag/Team}"
      }
    }
  }]
}
```

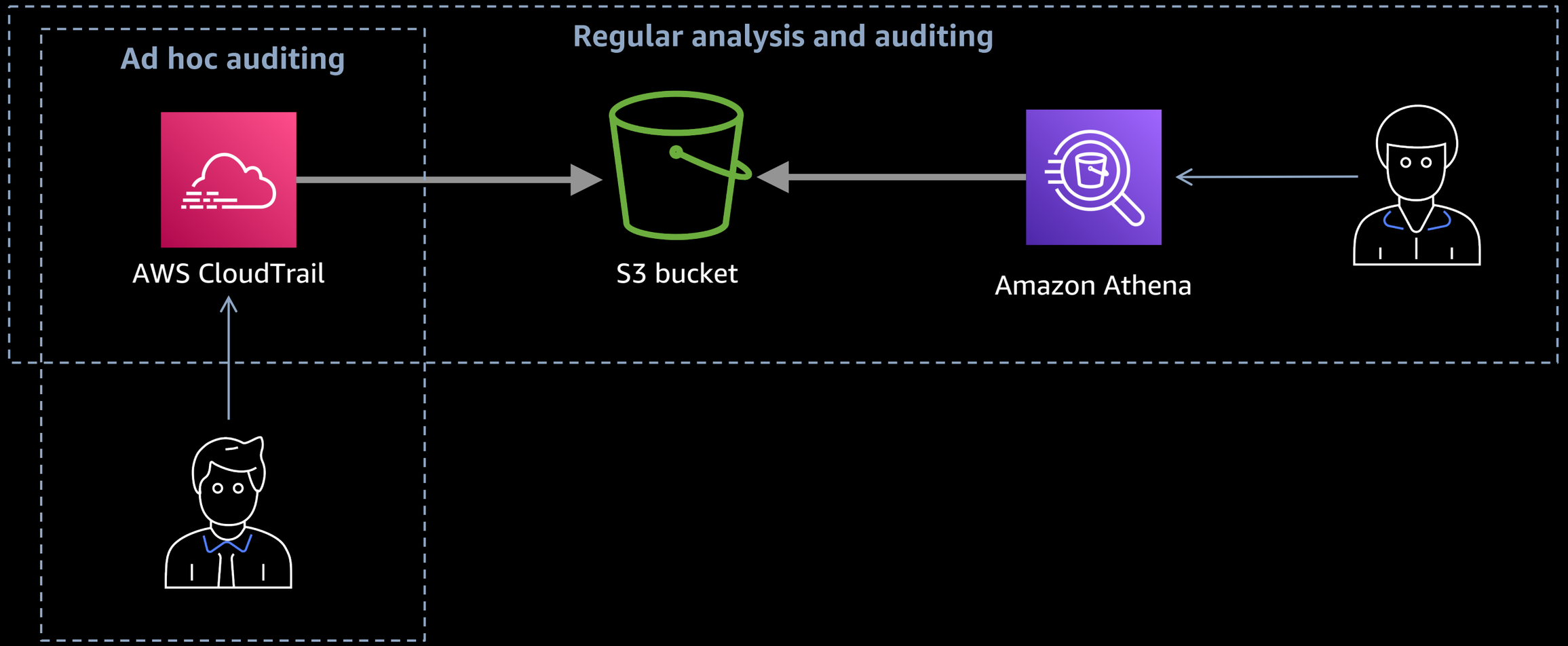
# Step 7: Audit AWS API calls

- AWS CloudTrail allows you to keep track of AWS API call events in your AWS account
- Recorded events will have
  - eventTime
  - userIdentity
  - eventSource (represents an AWS service)
  - eventName (represents an API action name)
  - and more!

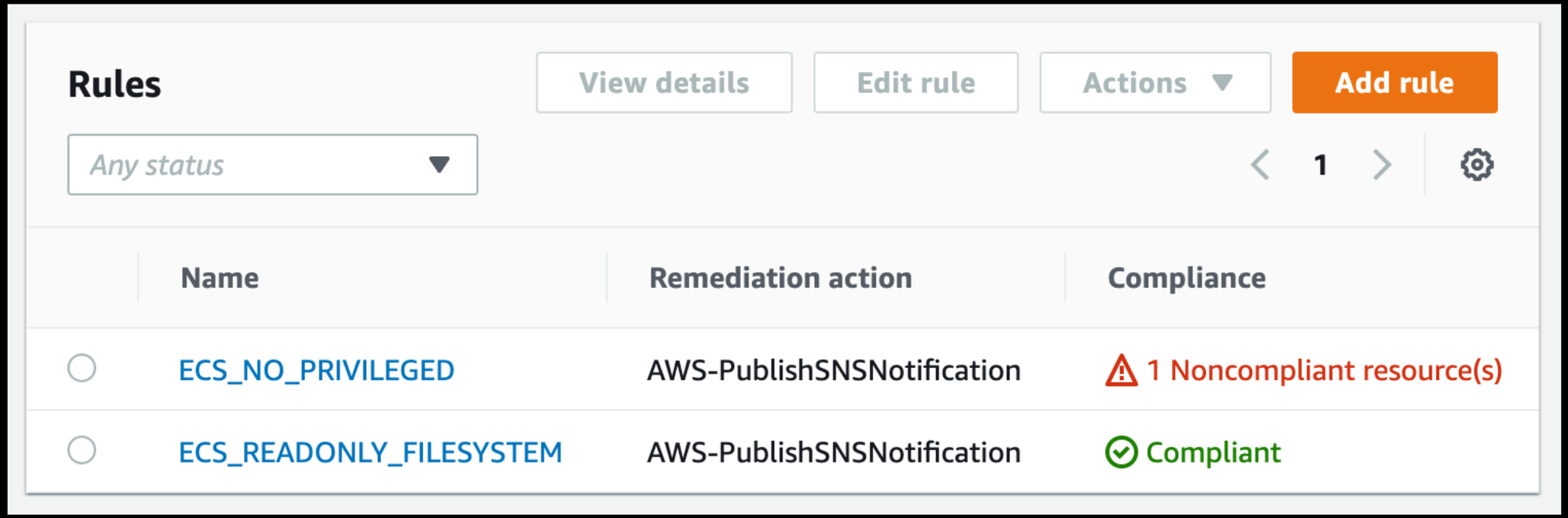
Learn more:
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-event-reference-record-contents.html



# Step 7: Audit Amazon ECS API calls



# Step 8: Validate your resources




Use AWS Config to validate your configurations to be compliant

- IAM resources can be automatically remediated
- Amazon ECS is not officially supported by AWS Config yet
- Write custom rules to validate Amazon ECS resources such as task definitions

For more examples, go to https://github.com/awslabs/aws-config-rules

# Steps to further secure workloads



AWS re:Invent

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

# Automate everything

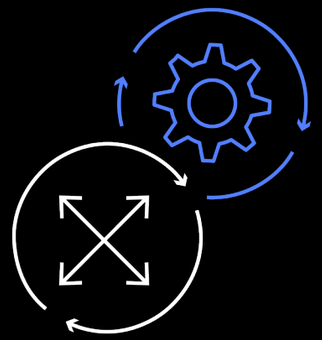



- Avoid manual operations, to process everything robustly and securely
- CI/CD pipelines help automate building, testing, and deployment

**Related session**

CON210 – Developing CI/CD pipelines with Amazon ECS and AWS Fargate

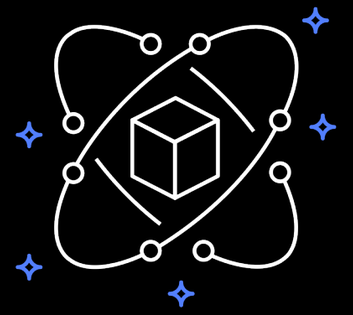
# Security benefits of AWS Fargate



**AWS manages and patches hosts for you**



**No privileged or SSH access**



**Secure isolation boundaries**

## Related sessions

CON203 – Choosing your container data plane on AWS

CON216 – AWS Fargate: Are serverless containers right for you?

# Takeaways





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

# Takeaways

## Secure your Amazon ECS applications

- Harden Amazon ECS tasks throughout their life cycles
- Keep operations robust

## One step further

- Automate everything as much as possible
- Use AWS Fargate to harden your workloads with less effort

## Get started today

- All Amazon ECS features covered today are available without installing any additional software

# Thank you!

Tori Hara

toricls

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.