

ECS Tasks Definitions

IAM Role	1
Role 1: netflix-task-role	1
Role 2: ecs-task-execution-role	3
ECS Task Definitions	4
JSON reference	7
CloudWatch Log Group	9

IAM Role

ECS Fargate is the serverless container orchestrator. It needs 2 IAM roles.

1. A role for our netflix application to call **secrets manager** to get the RDS credentials.
2. A role for Fargate cluster to pull the docker image and run the application.

Let's create these one by one.

Role 1: netflix-task-role

- This is for our application

Trusted entity type

The screenshot shows the 'Trusted entity type' section of the AWS IAM Role creation wizard. It lists five options: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Each option has a brief description below it.

Trusted entity type	Description
<input checked="" type="radio"/> AWS service	Allow AWS services like EC2, Lambda, or others to perform actions in this account.
<input type="radio"/> AWS account	Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
<input type="radio"/> Web identity	Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
<input type="radio"/> SAML 2.0 federation	Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
<input type="radio"/> Custom trust policy	Create a custom trust policy to enable others to perform actions in this account.

- Our ECS Task would be making AWS API calls

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Elastic Container Service

Choose a use case for the specified service.

Use case

Elastic Container Service

Allows ECS to create and manage AWS resources on your behalf.

Elastic Container Service Autoscale

Allows Auto Scaling to access and update ECS services.

Elastic Container Service Task

Allows ECS tasks to call AWS services on your behalf.

EC2 Role for Elastic Container Service

Allows EC2 instances in an ECS cluster to access ECS.

- We need these 2 permissions

- ECS tasks will have to register themselves to Target Groups
 - They will have to access Secrets Manager for credentials

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AmazonEC2ContainerServiceRole	AWS managed	Permissions policy
SecretsManagerReadWrite	AWS managed	Permissions policy

- Let's give a name for the role

Role details

Role name

Enter a meaningful name to identify this role.

netflix-task-role

Maximum 64 characters. Use alphanumeric and '+,-,@,_' characters.

Description

Role 2: ecs-task-execution-role

- This is for Fargate cluster

Trusted entity type

AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity
Allows users federated by the web identity provider to assume actions in this account.

SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

- Select ECS & Task Execution Role

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Elastic Container Service

Choose a use case for the specified service.

Use case

Allows ECS to create and manage AWS service resources required to encrypt Amazon ECS Service Connect

Task Execution Role for Elastic Container Service
Allows access to other AWS service resources that are required to run Amazon ECS tasks.

EC2 Role for ECS Managed Instances

- role name

Role details

Role name
Enter a meaningful name to identify this role.

ecs-task-execution-role

Maximum 64 characters. Use alphanumeric and '+,-,.,@,_' characters.

- Create the Role. Once created, ensure that you have this policy

The screenshot shows a list of permissions policies. At the top, it says "Permissions policies (1) [Info](#)". Below that, a note says "You can attach up to 10 managed policies." A search bar is present. The table has two columns: "Policy name" and "Type". One policy is listed: "AmazonECSTaskExecutionRolePolicy" (AWS managed).

Policy name	Type
AmazonECSTaskExecutionRolePolicy	AWS managed

ECS Task Definitions

We need to create task-definitions for every single microservice we have! This task definition is similar to Kubernetes deployment yaml file. It contains

- image name
 - CPU / memory
 - IAM role
- Create task definition for movie-service

The screenshot shows the "Task definition configuration" page. It asks for a "Task definition family" and provides a "Info" link. A note says "Specify a unique task definition family name." A text input field contains "movie-service". A note below says "Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed."

- Select Fargate

Launch type | [Info](#)

Selection of the launch type will change task definition parameters.

AWS Fargate

Serverless compute for containers.

Amazon EC2 instances

Self-managed infrastructure using Amazon EC2 instances.

- CPU architecture

OS, Architecture, Network mode

Network mode is used for tasks and is dependent on the compute type selected.

Operating system/Architecture | [Info](#)

Linux/X86_64

- CPU / Memory requirements

Task size | [Info](#)

Specify the amount of CPU and memory to reserve for your task.

CPU

.5 vCPU

Memory

1 GB

- IAM Roles

▼ Task roles - *conditional*

Task role | [Info](#)

A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the [IAM console](#)

netflix-task-role

Task execution role | [Info](#)

A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task exec

ecs-task-execution-role

- Provide docker image details. Copy Image URI from ECR or Select image tag

Select Amazon ECR image

Private repository

Select the repository containing the image you want to use.

127549748682.dkr.ecr.us-east-1.amazonaws.com/movie-service

Images (1/3)

Find Images

	Image tag	Image digest ↗	Pushed a
<input checked="" type="radio"/>	latest	<input type="text"/> sha256:2648bdeca7ebd7cc8c3f0ab...	November
<input type="radio"/>	-	<input type="text"/> sha256:b299db7fe63e06c765a31d...	November
<input type="radio"/>	-	<input type="text"/> sha256:f688f5bb0127f2793f2b61...	November

Select image by

Image digest

Use the SHA256 digest to reference this image.

Image tag

Use a human-readable tag to reference this image.

Image tag

latest



- container port 8080

Port mappings | [Info](#)

Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol	Remove
8080	TCP	container-port-protocol	HTTP	

- Environment variables

▼ Environment variables - optional

Environment variables | [Info](#)

Add individually

Add environment variables using plain text values or secrets from AWS Secrets Manager or Parameter Store.

Key	Value type	Value
SPRING_PROFILES_ACTIVE	<input type="text"/> Value	<input type="text"/> prod

[Add environment variable](#)

- Everything else is optional. Click on “Create”
- Repeat the same for “customer-service”

- We should have 2 task definitions created

The screenshot shows the 'Task definitions' page in the AWS ECS console. At the top, it says 'Task definitions (2)'. Below that is a search bar labeled 'Filter task definitions' and a filter button set to 'Active'. There are two entries in the table:

Task definition	Status of last revision
customer-service	ACTIVE
movie-service	ACTIVE

JSON reference

```
{
  "family": "movie-service",
  "containerDefinitions": [
    {
      "cpu": 0,
      "environment": [
        {
          "name": "SPRING_PROFILES_ACTIVE",
          "value": "prod"
        }
      ],
      "essential": true,
      "image": "127549748682.dkr.ecr.us-east-1.amazonaws.com/movie-service:latest",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/movie-service",
          "awslogs-create-group": "true",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "ecs"
        }
      },
      "mountPoints": [],
      "name": "movie-service",
      "portMappings": [
        {
          "containerPort": 8080,
          "hostPort": 8080,
          "protocol": "tcp"
        }
      ]
    }
  ]
}
```

```
        }
    ],
    "systemControls": [],
    "volumesFrom": []
}
],
"taskRoleArn": "arn:aws:iam::127549748682:role/netflux-task-role",
"executionRoleArn": "arn:aws:iam::127549748682:role/ecs-task-execution-role",
"networkMode": "awsvpc",
"volumes": [],
"placementConstraints": [],
"requiresCompatibilities": [
    "FARGATE"
],
"cpu": "512",
"memory": "1024"
}
```

```
{
    "family": "customer-service",
    "containerDefinitions": [
        {
            "cpu": 0,
            "environment": [
                {
                    "name": "SPRING_PROFILES_ACTIVE",
                    "value": "prod"
                }
            ],
            "essential": true,
            "image": "127549748682.dkr.ecr.us-east-1.amazonaws.com/customer-service:latest",
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/customer-service",
                    "awslogs-create-group": "true",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            },
            "mountPoints": [],
            "name": "customer-service",
            "portMappings": [
                {
                    "containerPort": 8080,
                    "hostPort": 8080,
                    "protocol": "tcp"
                }
            ],
            "systemControls": []
        }
    ]
}
```

```

        "volumesFrom": [],
    },
],
"taskRoleArn": "arn:aws:iam::127549748682:role/netflux-task-role",
"executionRoleArn": "arn:aws:iam::127549748682:role/ecs-task-execution-role",
"networkMode": "awsvpc",
"volumes": [],
"placementConstraints": [],
"requiresCompatibilities": [
    "FARGATE"
],
"cpu": "512",
"memory": "1024"
}

```

CloudWatch Log Group

- CloudWatch is the monitoring service provided by AWS. When applications run, they generate logs that need a place to be stored and managed.
- For this purpose, we create Log Groups in CloudWatch corresponding to each ECS task definition, such as:
 - /ecs/movie-service
 - /ecs/customer-service
- These log groups help organize and monitor logs for each service independently.

The screenshot shows the AWS CloudWatch Log Groups interface. On the left, there's a navigation sidebar with 'CloudWatch' selected. Under 'Logs', 'Log groups' is also selected. The main area displays 'Log groups (2)'. It includes a search bar and a table with two entries:

<input type="checkbox"/> Log group	Log class	Anomaly d...
<input type="checkbox"/> /ecs/movie-service	Standard	Configure
<input type="checkbox"/> /ecs/customer-service	Standard	Configure