

```
--region aws-region \
--query 'Stacks[0].Outputs[?OutputKey==`LoadBalancerURL`].OutputValue' \
--output text
```

Output:

```
http://ecs-tutorial-stack-alb-0123456789.aws-region.elb.amazonaws.com
```

Paste the DNS name into a browser to view a webpage that displays a sample Amazon ECS application.

## Step 3: Clean up

To clean up the resources you created, run the following command.

```
aws cloudformation delete-stack \
--stack-name ecs-stack
```

The delete-stack command initiates deletion of the AWS CloudFormation stack that was created in this tutorial, deleting all the resources in the stack. To verify deletion, you can repeat the procedure in [Step 2: Verify Amazon ECS resource creation](#). The list of ARNs in the outputs will no longer include a task definition called ecs-tutorial-stack-task or a cluster called ecs-tutorial-stack-cluster. The list-services call will fail.

## AWS CloudFormation example templates for Amazon ECS

You can create Amazon ECS clusters, task definitions, and services using AWS CloudFormation. The following topics include templates that demonstrate how to create resources with different configurations. You can create these resources with these templates by using the AWS CloudFormation console or the AWS CLI.

AWS CloudFormation templates are text files in the JSON or YAML format that describe the resources that you want to provision in your AWS CloudFormation stacks. If you're unfamiliar with either the JSON or YAML format, or both, you can use AWS Infrastructure Composer to get started using AWS CloudFormation templates. For more information, see [Create templates visually with Infrastructure Composer](#) in the *AWS CloudFormation User Guide*.

The following topics list example templates for Amazon ECS task definitions, clusters, and services.

### Topics

- [Task definitions](#)
- [Capacity providers](#)
- [Clusters](#)
- [Services](#)
- [IAM roles for Amazon ECS](#)

## Task definitions

A task definition is a blueprint for your application that describes the parameters and one or more containers that form your application. The following are example AWS CloudFormation templates for Amazon ECS task definitions. For more information about Amazon ECS task definitions, see [Amazon ECS task definitions](#).

### Fargate Linux task definition

You can use the following template to create a sample Fargate Linux task.

JSON

```
{  
    "AWSTemplateFormatVersion": "2010-09-09",  
    "Description": "ECS Task Definition with parameterized values",  
    "Parameters": {  
        "ContainerImage": {  
            "Type": "String",  
            "Default": "public.ecr.aws/docker/library/httpd:2.4",  
            "Description": "The container image to use for the task"  
        },  
        "ContainerCpu": {  
            "Type": "Number",  
            "Default": 256,  
            "Description": "The number of CPU units to reserve for the container",  
            "AllowedValues": [256, 512, 1024, 2048, 4096]  
        },  
        "ContainerMemory": {  
            "Type": "Number",  
            "Default": 512,  
            "Description": "The amount of memory (in MiB) to reserve for the container",  
            "AllowedValues": [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]  
        },  
    }  
}
```

```
"TaskFamily": {  
    "Type": "String",  
    "Default": "task-definition-cfn",  
    "Description": "The name of the task definition family"  
},  
"ContainerName": {  
    "Type": "String",  
    "Default": "sample-fargate-app",  
    "Description": "The name of the container"  
},  
"ContainerPort": {  
    "Type": "Number",  
    "Default": 80,  
    "Description": "The port number on the container"  
},  
"HostPort": {  
    "Type": "Number",  
    "Default": 80,  
    "Description": "The port number on the host"  
},  
"ExecutionRoleArn": {  
    "Type": "String",  
    "Default": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole",  
    "Description": "The ARN of the task execution role"  
},  
"LogGroup": {  
    "Type": "String",  
    "Default": "/ecs/fargate-task-definition",  
    "Description": "The CloudWatch log group for container logs"  
},  
"NetworkMode": {  
    "Type": "String",  
    "Default": "awsvpc",  
    "Description": "The Docker networking mode to use",  
    "AllowedValues": ["awsvpc", "bridge", "host", "none"]  
},  
"OperatingSystemFamily": {  
    "Type": "String",  
    "Default": "LINUX",  
    "Description": "The operating system for the task",  
    "AllowedValues": ["LINUX", "WINDOWS_SERVER_2019_FULL",  
    "WINDOWS_SERVER_2019_CORE", "WINDOWS_SERVER_2022_FULL", "WINDOWS_SERVER_2022_CORE"]  
},  
},
```

```
"Resources": {
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "ContainerDefinitions": [
                {
                    "Command": [
                        "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon ECS.</p> </div></body></html>' > /usr/local/apache2/htdocs/index.html && -foreground\""
                    ],
                    "EntryPoint": [
                        "sh",
                        "-c"
                    ],
                    "Essential": true,
                    "Image": {"Ref": "ContainerImage"},
                    "LogConfiguration": {
                        "LogDriver": "awslogs",
                        "Options": {
                            "mode": "non-blocking",
                            "max-buffer-size": "25m",
                            "awslogs-create-group": "true",
                            "awslogs-group": {"Ref": "LogGroup"},
                            "awslogs-region": {"Ref": "AWS::Region"},
                            "awslogs-stream-prefix": "ecs"
                        }
                    },
                    "Name": {"Ref": "ContainerName"},
                    "PortMappings": [
                        {
                            "ContainerPort": {"Ref": "ContainerPort"},
                            "HostPort": {"Ref": "HostPort"},
                            "Protocol": "tcp"
                        }
                    ]
                }
            ],
            "Cpu": {"Ref": "ContainerCpu"},
            "ExecutionRoleArn": {"Ref": "ExecutionRoleArn"},
            "Family": {"Ref": "TaskFamily"},
```

```
"Memory": {"Ref": "ContainerMemory"},  
"NetworkMode": {"Ref": "NetworkMode"},  
"RequiresCompatibilities": [  
    "FARGATE"  
],  
"RuntimePlatform": {  
    "OperatingSystemFamily": {"Ref": "OperatingSystemFamily"}  
}  
}  
}  
}  
},  
"Outputs": {  
    "TaskDefinitionArn": {  
        "Description": "The ARN of the created task definition",  
        "Value": {"Ref": "ECSTaskDefinition"}  
    }  
}  
}  
}
```

## YAML

```
AWSTemplateFormatVersion: 2010-09-09  
Description: 'ECS Task Definition to deploy a sample app'  
Parameters:  
  ContainerImage:  
    Type: String  
    Default: 'public.ecr.aws/docker/library/httpd:2.4'  
    Description: The container image to use for the task  
  ContainerCpu:  
    Type: Number  
    Default: 256  
    Description: The number of CPU units to reserve for the container  
    AllowedValues: [256, 512, 1024, 2048, 4096]  
  ContainerMemory:  
    Type: Number  
    Default: 512  
    Description: The amount of memory (in MiB) to reserve for the container  
    AllowedValues: [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]  
  TaskFamily:  
    Type: String  
    Default: 'task-definition-cfn'  
    Description: The name of the task definition family  
  ContainerName:
```

```
Type: String
Default: 'sample-fargate-app'
Description: The name of the container
ContainerPort:
  Type: Number
  Default: 80
  Description: The port number on the container
HostPort:
  Type: Number
  Default: 80
  Description: The port number on the host
ExecutionRoleArn:
  Type: String
  Default: 'arn:aws:iam::111122223333:role/ecsTaskExecutionRole'
  Description: The ARN of the task execution role
LogGroup:
  Type: String
  Default: '/ecs/fargate-task-definition'
  Description: The CloudWatch log group for container logs
NetworkMode:
  Type: String
  Default: 'awsvpc'
  Description: The Docker networking mode to use
  AllowedValues: ['awsvpc', 'bridge', 'host', 'none']
OperatingSystemFamily:
  Type: String
  Default: 'LINUX'
  Description: The operating system for the task
  AllowedValues: ['LINUX', 'WINDOWS_SERVER_2019_FULL', 'WINDOWS_SERVER_2019_CORE',
'WINDOWS_SERVER_2022_FULL', 'WINDOWS_SERVER_2022_CORE']
Resources:
  ECSTaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      ContainerDefinitions:
        - Command:
          - >-
            /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
App</title> <style>body {margin-top: 40px; background-color:
#333;} </style> </head><body> <div
style=color:white;text-align:center> <h1>Amazon ECS Sample
App</h1> <h2>Congratulations!</h2> <p>Your application is now
running on a container in Amazon ECS.</p> </div></body></html>' >
/usr/local/apache2/htdocs/index.html && httpd-foreground"
```

```
EntryPoint:
  - sh
  - '-c'
Essential: true
Image: !Ref ContainerImage
LogConfiguration:
  LogDriver: awslogs
  Options:
    mode: non-blocking
    max-buffer-size: 25m
    awslogs-create-group: 'true'
    awslogs-group: !Ref LogGroup
    awslogs-region: !Ref AWS::Region
    awslogs-stream-prefix: ecs
Name: !Ref ContainerName
PortMappings:
  - ContainerPort: !Ref ContainerPort
    HostPort: !Ref HostPort
    Protocol: tcp
Cpu: !Ref ContainerCpu
ExecutionRoleArn: !Ref ExecutionRoleArn
Family: !Ref TaskFamily
Memory: !Ref ContainerMemory
NetworkMode: !Ref NetworkMode
RequiresCompatibilities:
  - FARGATE
RuntimePlatform:
  OperatingSystemFamily: !Ref OperatingSystemFamily
Outputs:
  TaskDefinitionArn:
    Description: The ARN of the created task definition
    Value: !Ref ECSTaskDefinition
```

## Amazon EFS task definition

You can use the following template to create a task that uses an Amazon EFS file system that you created. For more information about using Amazon EFS volumes with Amazon ECS, see [Use Amazon EFS volumes with Amazon ECS](#).

JSON

```
{
```

```
"AWSTemplateFormatVersion": "2010-09-09",
"Description": "Create a task definition for a web server with parameterized values.",
"Parameters": {
    "ExecutionRoleArn": {
        "Type": "String",
        "Default": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
        "Description": "The ARN of the task execution role"
    },
    "NetworkMode": {
        "Type": "String",
        "Default": "awsvpc",
        "Description": "The Docker networking mode to use",
        "AllowedValues": ["awsvpc", "bridge", "host", "none"]
    },
    "TaskFamily": {
        "Type": "String",
        "Default": "my-ecs-task",
        "Description": "The name of the task definition family"
    },
    "ContainerCpu": {
        "Type": "String",
        "Default": "256",
        "Description": "The number of CPU units to reserve for the container",
        "AllowedValues": ["256", "512", "1024", "2048", "4096"]
    },
    "ContainerMemory": {
        "Type": "String",
        "Default": "512",
        "Description": "The amount of memory (in MiB) to reserve for the container",
        "AllowedValues": ["512", "1024", "2048", "3072", "4096", "5120", "6144", "7168", "8192"]
    },
    "ContainerName": {
        "Type": "String",
        "Default": "nginx",
        "Description": "The name of the container"
    },
    "ContainerImage": {
        "Type": "String",
        "Default": "public.ecr.aws/nginx/nginx:latest",
        "Description": "The container image to use for the task"
    },
    "ContainerPort": {
```

```
"Type": "Number",
"Default": 80,
"Description": "The port number on the container"
},
"InitProcessEnabled": {
    "Type": "String",
    "Default": "true",
    "Description": "Whether to enable the init process inside the container",
    "AllowedValues": ["true", "false"]
},
"EfsVolumeName": {
    "Type": "String",
    "Default": "efs-volume",
    "Description": "The name of the EFS volume"
},
"EfsContainerPath": {
    "Type": "String",
    "Default": "/usr/share/nginx/html",
    "Description": "The path in the container where the EFS volume will be mounted"
},
"LogGroup": {
    "Type": "String",
    "Default": "LogGroup",
    "Description": "The CloudWatch log group for container logs"
},
"LogStreamPrefix": {
    "Type": "String",
    "Default": "efs-task",
    "Description": "The prefix for the log stream"
},
"EfsFilesystemId": {
    "Type": "String",
    "Default": "fs-1234567890abcdef0",
    "Description": "The ID of the EFS filesystem"
},
"EfsRootDirectory": {
    "Type": "String",
    "Default": "/",
    "Description": "The root directory in the EFS filesystem"
},
"EfsTransitEncryption": {
    "Type": "String",
    "Default": "ENABLED",
```

```
        "Description": "Whether to enable transit encryption for EFS",
        "AllowedValues": ["ENABLED", "DISABLED"]
    },
},
"Resources": {
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "ExecutionRoleArn": {"Ref": "ExecutionRoleArn"},
            "NetworkMode": {"Ref": "NetworkMode"},
            "RequiresCompatibilities": ["FARGATE"],
            "Family": {"Ref": "TaskFamily"},
            "Cpu": {"Ref": "ContainerCpu"},
            "Memory": {"Ref": "ContainerMemory"},
            "ContainerDefinitions": [
                {
                    "Name": {"Ref": "ContainerName"},
                    "Image": {"Ref": "ContainerImage"},
                    "Essential": true,
                    "PortMappings": [
                        {
                            "ContainerPort": {"Ref": "ContainerPort"},
                            "Protocol": "tcp"
                        }
                    ],
                    "LinuxParameters": {
                        "InitProcessEnabled": {"Ref": "InitProcessEnabled"}
                    },
                    "MountPoints": [
                        {
                            "SourceVolume": {"Ref": "EfsVolumeName"},
                            "ContainerPath": {"Ref": "EfsContainerPath"}
                        }
                    ],
                    "LogConfiguration": {
                        "LogDriver": "awslogs",
                        "Options": {
                            "mode": "non-blocking",
                            "max-buffer-size": "25m",
                            "awslogs-group": {"Ref": "LogGroup"},
                            "awslogs-region": {"Ref": "AWS::Region"},
                            "awslogs-create-group": "true",
                            "awslogs-stream-prefix": {"Ref": "LogStreamPrefix"}
                        }
                    }
                }
            ]
        }
    }
}
```

```
        }
    ],
    "Volumes": [
        {
            "Name": {"Ref": "EfsVolumeName"},
            "EFSVolumeConfiguration": {
                "FilesystemId": {"Ref": "EfsFilesystemId"},
                "RootDirectory": {"Ref": "EfsRootDirectory"},
                "TransitEncryption": {"Ref": "EfsTransitEncryption"}
            }
        }
    ]
},
"Outputs": {
    "TaskDefinitionArn": {
        "Description": "The ARN of the created task definition",
        "Value": {"Ref": "ECSTaskDefinition"}
    }
}
}
```

## YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create a task definition for a web server with parameterized values.
Parameters:
  ExecutionRoleArn:
    Type: String
    Default: arn:aws:iam::123456789012:role/ecsTaskExecutionRole
    Description: The ARN of the task execution role
  NetworkMode:
    Type: String
    Default: awsvpc
    Description: The Docker networking mode to use
    AllowedValues: [awsvpc, bridge, host, none]
  TaskFamily:
    Type: String
    Default: my-ecs-task
    Description: The name of the task definition family
  ContainerCpu:
```

```
Type: String
Default: "256"
Description: The number of CPU units to reserve for the container
AllowedValues: ["256", "512", "1024", "2048", "4096"]

ContainerMemory:
Type: String
Default: "512"
Description: The amount of memory (in MiB) to reserve for the container
AllowedValues: ["512", "1024", "2048", "3072", "4096", "5120", "6144", "7168",
"8192"]

ContainerName:
Type: String
Default: nginx
Description: The name of the container

ContainerImage:
Type: String
Default: public.ecr.aws/nginx/nginx:latest
Description: The container image to use for the task

ContainerPort:
Type: Number
Default: 80
Description: The port number on the container

InitProcessEnabled:
Type: String
Default: "true"
Description: Whether to enable the init process inside the container
AllowedValues: ["true", "false"]

EfsVolumeName:
Type: String
Default: efs-volume
Description: The name of the EFS volume

EfsContainerPath:
Type: String
Default: /usr/share/nginx/html
Description: The path in the container where the EFS volume will be mounted

LogGroup:
Type: String
Default: LogGroup
Description: The CloudWatch log group for container logs

LogStreamPrefix:
Type: String
Default: efs-task
Description: The prefix for the log stream

EfsFilesystemId:
```

```
Type: String
Default: fs-1234567890abcdef0
Description: The ID of the EFS filesystem

EfsRootDirectory:
Type: String
Default: /
Description: The root directory in the EFS filesystem

EfsTransitEncryption:
Type: String
Default: ENABLED
Description: Whether to enable transit encryption for EFS
AllowedValues: [ENABLED, DISABLED]

Resources:
ECSTaskDefinition:
Type: AWS::ECS::TaskDefinition
Properties:
ExecutionRoleArn: !Ref ExecutionRoleArn
NetworkMode: !Ref NetworkMode
RequiresCompatibilities:
- FARGATE
Family: !Ref TaskFamily
Cpu: !Ref ContainerCpu
Memory: !Ref ContainerMemory
ContainerDefinitions:
- Name: !Ref ContainerName
Image: !Ref ContainerImage
Essential: true
PortMappings:
- ContainerPort: !Ref ContainerPort
Protocol: tcp
LinuxParameters:
InitProcessEnabled: !Ref InitProcessEnabled
MountPoints:
- SourceVolume: !Ref EfsVolumeName
ContainerPath: !Ref EfsContainerPath
LogConfiguration:
LogDriver: awslogs
Options:
mode: non-blocking
max-buffer-size: 25m
awslogs-group: !Ref LogGroup
awslogs-region: !Ref AWS::Region
awslogs-create-group: "true"
awslogs-stream-prefix: !Ref LogStreamPrefix
```

```
Volumes:  
  - Name: !Ref EfsVolumeName  
    EFSVolumeConfiguration:  
      FilesystemId: !Ref EfsFilesystemId  
      RootDirectory: !Ref EfsRootDirectory  
      TransitEncryption: !Ref EfsTransitEncryption  
  
Outputs:  
  TaskDefinitionArn:  
    Description: The ARN of the created task definition  
    Value: !Ref ECSTaskDefinition
```

## Capacity providers

Capacity providers are associated with an Amazon ECS cluster and are used to manage compute capacity for your workloads.

### Create a capacity provider for Amazon ECS Managed Instances

By default, Amazon ECS provides a capacity provider that automatically selects the most cost-optimized general-purpose instance types. However, you can create custom capacity providers to specify instance requirements such as instance types, CPU manufacturers, accelerator types, and other requirements. You can use the following template to create a capacity provider for Amazon ECS Managed Instances that satisfies the specified memory and CPU requirements.

#### JSON

```
{  
  "AWSTemplateFormatVersion": "2010-09-09",  
  "Resources": {  
    "MyCapacityProvider": {  
      "Type": "AWS::ECS::CapacityProvider",  
      "Properties": {  
        "ManagedInstancesProvider": {  
          "InfrastructureRoleArn": "arn:aws:iam::123456789012:role/  
ECSInfrastructureRole",  
          "InstanceLaunchTemplate": {  
            "Ec2InstanceProfileArn":  
              "arn:aws:iam::123456789012:instance-profile/ecsInstanceProfile",  
              "NetworkConfiguration": null,  
              "Subnets": [  
                "subnet-12345678"
```

```
        ],
        "SecurityGroups": [
            "sg-87654321"
        ]
    },
    "StorageConfiguration": {
        "StorageSizeGiB": 30
    },
    "InstanceRequirements": {
        "VCpuCount": {
            "Min": 1,
            "Max": 4
        },
        "MemoryMiB": {
            "Min": 2048,
            "Max": 8192
        }
    }
}
}
```

## YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyCapacityProvider:
    Type: AWS::ECS::CapacityProvider
    Properties:
      ManagedInstancesProvider:
        InfrastructureRoleArn: "arn:aws:iam::123456789012:role/
ECSInfrastructureRole"
      InstanceLaunchTemplate:
        Ec2InstanceProfileArn: "arn:aws:iam::123456789012:instance-profile/
ecsInstanceProfile"
        NetworkConfiguration:
          Subnets:
            - "subnet-12345678"
      SecurityGroups:
        - "sg-87654321"
      StorageConfiguration:
```

```
  StorageSizeGiB: 30
  InstanceRequirements:
    VCpuCount:
      Min: 1
      Max: 4
    MemoryMiB:
      Min: 2048
      Max: 8192
```

## Clusters

An Amazon ECS cluster is a logical grouping of tasks or services. You can use the following templates to create clusters with different configurations. For more information about Amazon ECS clusters, see [Amazon ECS clusters](#).

### Create an empty cluster with default settings

You can use the following template to create an empty cluster with default settings.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "ECSCluster": {
      "Type": "AWS::ECS::Cluster",
      "Properties": {
        "ClusterName": "MyEmptyCluster"
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
    Properties:
      ClusterName: MyEmptyCluster
```

## Create an empty cluster with managed storage encryption and enhanced Container Insights

You can use the following template to create a cluster with cluster-level managed storage and enhanced Container Insights enabled. Cluster-level encryption applies to Amazon ECS managed data volumes such as Amazon EBS volumes. For more information about Amazon EBS encryption, see [Encrypt data stored in Amazon EBS volumes attached to Amazon ECS tasks](#). For more information about using Container Insights with enhanced observability, see [Monitor Amazon ECS containers using Container Insights with enhanced observability](#).

JSON

```
{  
    "AWSTemplateFormatVersion": "2010-09-09",  
    "Resources": {  
        "Cluster": {  
            "Type": "AWS::ECS::Cluster",  
            "Properties": {  
                "ClusterName": "EncryptedEnhancedCluster",  
                "ClusterSettings": [  
                    {  
                        "Name": "containerInsights",  
                        "Value": "enhanced"  
                    }  
                ],  
                "Configuration": {  
                    "ManagedStorageConfiguration": {  
                        "KmsKeyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"  
                    }  
                }  
            }  
        }  
    }  
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09  
Resources:  
  Cluster:  
    Type: AWS::ECS::Cluster  
    Properties:
```

```
ClusterName: EncryptedEnhancedCluster
ClusterSettings:
  - Name: containerInsights
    Value: enhanced
Configuration:
  ManagedStorageConfiguration:
    KmsKeyId: a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

## Create a cluster with the AL2023 Amazon ECS-Optimized-AMI

You can use the following template to create a cluster that uses a capacity provider that launches AL2023 instances on Amazon EC2.

### Important

For the latest AMI IDs, see [Amazon ECS-optimized AMI](#) in the *Amazon Elastic Container Service Developer Guide*.

## JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "EC2 ECS cluster that starts out empty, with no EC2 instances yet. An ECS capacity provider automatically launches more EC2 instances as required on the fly when you request ECS to launch services or standalone tasks.",
  "Parameters": {
    "InstanceType": {
      "Type": "String",
      "Description": "EC2 instance type",
      "Default": "t2.medium",
      "AllowedValues": [
        "t1.micro",
        "t2.2xlarge",
        "t2.large",
        "t2.medium",
        "t2.micro",
        "t2.nano",
        "t2.small",
        "t2.xlarge",
        "t3.2xlarge",
        "t3.4xlarge",
        "t3.8xlarge",
        "t3.12xlarge",
        "t3.16xlarge"
      ]
    }
  }
}
```

```
        "t3.large",
        "t3.medium",
        "t3.micro",
        "t3.nano",
        "t3.small",
        "t3.xlarge"
    ],
},
"DesiredCapacity": {
    "Type": "Number",
    "Default": "0",
    "Description": "Number of EC2 instances to launch in your ECS cluster."
},
"MaxSize": {
    "Type": "Number",
    "Default": "100",
    "Description": "Maximum number of EC2 instances that can be launched in
your ECS cluster."
},
"ECSAMI": {
    "Description": "The Amazon Machine Image ID used for the cluster",
    "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
    "Default": "/aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended/image_id"
},
"VpcId": {
    "Type": "AWS::EC2::VPC::Id",
    "Description": "VPC ID where the ECS cluster is launched",
    "Default": "vpc-1234567890abcdef0"
},
"SubnetIds": {
    "Type": "List<AWS::EC2::Subnet::Id>",
    "Description": "List of subnet IDs where the EC2 instances will be
launched",
    "Default": "subnet-021345abcdef67890"
}
},
"Resources": {
    "ECSCluster": {
        "Type": "AWS::ECS::Cluster",
        "Properties": {
            "ClusterSettings": [
                {
                    "Name": "containerInsights",

```

```
        "Value": "enabled"
    }
}
},
"ECSAutoScalingGroup": {
    "Type": "AWS::AutoScaling::AutoScalingGroup",
    "DependsOn": [
        "ECSCluster",
        "EC2Role"
    ],
    "Properties": {
        "VPCZoneIdentifier": {
            "Ref": "SubnetIds"
        },
        "LaunchTemplate": {
            "LaunchTemplateId": {
                "Ref": "ContainerInstances"
            },
            "Version": {
                "Fn::GetAtt": [
                    "ContainerInstances",
                    "LatestVersionNumber"
                ]
            }
        },
        "MinSize": 0,
        "MaxSize": {
            "Ref": "MaxSize"
        },
        "DesiredCapacity": {
            "Ref": "DesiredCapacity"
        },
        "NewInstancesProtectedFromScaleIn": true
    },
    "UpdatePolicy": {
        "AutoScalingReplacingUpdate": {
            "WillReplace": "true"
        }
    }
},
"ContainerInstances": {
    "Type": "AWS::EC2::LaunchTemplate",
    "Properties": {
```

```
        "LaunchTemplateName": "asg-launch-template-2",
        "LaunchTemplateData": {
            "ImageId": {
                "Ref": "ECSAMI"
            },
            "InstanceType": {
                "Ref": "InstanceType"
            },
            "IamInstanceProfile": {
                "Name": {
                    "Ref": "EC2InstanceProfile"
                }
            },
            "SecurityGroupIds": [
                {
                    "Ref": "ContainerHostSecurityGroup"
                }
            ],
            "UserData": {
                "Fn::Base64": {
                    "Fn::Sub": "#!/bin/bash -xe\n echo ECS_CLUSTER=\${{ECSCluster}} >> /etc/ecs/ecs.config\n yum install -y aws-cfn-bootstrap\n /opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource ContainerInstances --configsets full_install --region ${AWS::Region} &\n"
                }
            },
            "MetadataOptions": {
                "HttpEndpoint": "enabled",
                "HttpTokens": "required"
            }
        }
    },
    "EC2InstanceProfile": {
        "Type": "AWS::IAM::InstanceProfile",
        "Properties": {
            "Path": "/",
            "Roles": [
                {
                    "Ref": "EC2Role"
                }
            ]
        }
    }
},
```

```
"CapacityProvider": {
    "Type": "AWS::ECS::CapacityProvider",
    "Properties": {
        "AutoScalingGroupProvider": {
            "AutoScalingGroupArn": {
                "Ref": "ECSAutoScalingGroup"
            },
            "ManagedScaling": {
                "InstanceWarmupPeriod": 60,
                "MinimumScalingStepSize": 1,
                "MaximumScalingStepSize": 100,
                "Status": "ENABLED",
                "TargetCapacity": 100
            },
            "ManagedTerminationProtection": "ENABLED"
        }
    }
},
"CapacityProviderAssociation": {
    "Type": "AWS::ECS::ClusterCapacityProviderAssociations",
    "Properties": {
        "CapacityProviders": [
            {
                "Ref": "CapacityProvider"
            }
        ],
        "Cluster": {
            "Ref": "ECScluster"
        },
        "DefaultCapacityProviderStrategy": [
            {
                "Base": 0,
                "CapacityProvider": {
                    "Ref": "CapacityProvider"
                },
                "Weight": 1
            }
        ]
    }
},
"ContainerHostSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Access to the EC2 hosts that run containers",

```

```
        "VpcId": {
            "Ref": "VpcId"
        }
    },
    "EC2Role": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "ec2.amazonaws.com"
                            ]
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ]
                    }
                ]
            },
            "Path": "/",
            "ManagedPolicyArns": [
                "arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role",
                "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceStateforCore"
            ]
        }
    },
    "ECSTaskExecutionRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "ecs-tasks.amazonaws.com"
                            ]
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ]
                    }
                ]
            }
        }
    }
}
```

```
        "sts:AssumeRole"
    ],
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": {
                "Fn::Sub": "arn:${AWS::Partition}:ecs:
${AWS::Region}:${AWS::AccountId}:*"
            }
        },
        "StringEquals": {
            "aws:SourceAccount": {
                "Fn::Sub": "${AWS::AccountId}"
            }
        }
    }
},
"Path": "/",
"ManagedPolicyArns": [
    "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
]
}
},
],
"Outputs": {
    "ClusterName": {
        "Description": "The ECS cluster into which to launch resources",
        "Value": "ECSCluster"
    },
    "ECSTaskExecutionRole": {
        "Description": "The role used to start up a task",
        "Value": "ECSTaskExecutionRole"
    },
    "CapacityProvider": {
        "Description": "The cluster capacity provider that the service should
use to request capacity when it wants to start up a task",
        "Value": "CapacityProvider"
    }
}
}
```

## YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: EC2 ECS cluster that starts out empty, with no EC2 instances yet. An ECS capacity provider automatically launches more EC2 instances as required on the fly when you request ECS to launch services or standalone tasks.

Parameters:
  InstanceType:
    Type: String
    Description: EC2 instance type
    Default: t2.medium
    AllowedValues:
      - t1.micro
      - t2.2xlarge
      - t2.large
      - t2.medium
      - t2.micro
      - t2.nano
      - t2.small
      - t2.xlarge
      - t3.2xlarge
      - t3.large
      - t3.medium
      - t3.micro
      - t3.nano
      - t3.small
      - t3.xlarge
  DesiredCapacity:
    Type: Number
    Default: '0'
    Description: Number of EC2 instances to launch in your ECS cluster.
  MaxSize:
    Type: Number
    Default: '100'
    Description: Maximum number of EC2 instances that can be launched in your ECS cluster.
  ECSAMI:
    Description: The Amazon Machine Image ID used for the cluster
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/image_id
  VpcId:
    Type: AWS::EC2::VPC::Id
    Description: VPC ID where the ECS cluster is launched
    Default: vpc-1234567890abcdef0
```

```
SubnetIds:  
  Type: List<AWS::EC2::Subnet::Id>  
  Description: List of subnet IDs where the EC2 instances will be launched  
  Default: subnet-021345abcdef67890  
Resources:  
  ECSCluster:  
    Type: AWS::ECS::Cluster  
    Properties:  
      ClusterSettings:  
        - Name: containerInsights  
          Value: enabled  
  ECSAutoScalingGroup:  
    Type: AWS::AutoScaling::AutoScalingGroup  
    DependsOn:  
      - ECSCluster  
      - EC2Role  
    Properties:  
      VPCZoneIdentifier: !Ref SubnetIds  
      LaunchTemplate:  
        LaunchTemplateId: !Ref ContainerInstances  
        Version: !GetAtt ContainerInstances.LatestVersionNumber  
      MinSize: 0  
      MaxSize: !Ref MaxSize  
      DesiredCapacity: !Ref DesiredCapacity  
      NewInstancesProtectedFromScaleIn: true  
    UpdatePolicy:  
      AutoScalingReplacingUpdate:  
        WillReplace: 'true'  
  ContainerInstances:  
    Type: AWS::EC2::LaunchTemplate  
    Properties:  
      LaunchTemplateName: asg-launch-template-2  
      LaunchTemplateData:  
        ImageId: !Ref ECSAMI  
        InstanceType: !Ref InstanceType  
        IamInstanceProfile:  
          Name: !Ref EC2InstanceProfile  
        SecurityGroupIds:  
          - !Ref ContainerHostSecurityGroup  
      UserData: !Base64  
        Fn::Sub: |  
          #!/bin/bash -xe  
          echo ECS_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config  
          yum install -y aws-cfn-bootstrap
```

```
/opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource
ContainerInstances --configsets full_install --region ${AWS::Region} &
    MetadataOptions:
        HttpEndpoint: enabled
        HttpTokens: required
    EC2InstanceProfile:
        Type: AWS::IAM::InstanceProfile
        Properties:
            Path: /
            Roles:
                - !Ref EC2Role
    CapacityProvider:
        Type: AWS::ECS::CapacityProvider
        Properties:
            AutoScalingGroupProvider:
                AutoScalingGroupArn: !Ref ECSAutoScalingGroup
            ManagedScaling:
                InstanceWarmupPeriod: 60
                MinimumScalingStepSize: 1
                MaximumScalingStepSize: 100
                Status: ENABLED
                TargetCapacity: 100
            ManagedTerminationProtection: ENABLED
    CapacityProviderAssociation:
        Type: AWS::ECS::ClusterCapacityProviderAssociations
        Properties:
            CapacityProviders:
                - !Ref CapacityProvider
            Cluster: !Ref ECSCluster
            DefaultCapacityProviderStrategy:
                - Base: 0
                    CapacityProvider: !Ref CapacityProvider
                    Weight: 1
    ContainerHostSecurityGroup:
        Type: AWS::EC2::SecurityGroup
        Properties:
            GroupDescription: Access to the EC2 hosts that run containers
            VpcId: !Ref VpcId
    EC2Role:
        Type: AWS::IAM::Role
        Properties:
            AssumeRolePolicyDocument:
                Statement:
                    - Effect: Allow
```

```
Principal:  
  Service:  
    - ec2.amazonaws.com  
Action:  
  - sts:AssumeRole  
Path: /  
ManagedPolicyArns:  
  - arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role  
  - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore  
ECSTaskExecutionRole:  
  Type: AWS::IAM::Role  
Properties:  
  AssumeRolePolicyDocument:  
    Statement:  
      - Effect: Allow  
      Principal:  
        Service:  
          - ecs-tasks.amazonaws.com  
    Action:  
      - sts:AssumeRole  
  Condition:  
    ArnLike:  
      aws:SourceArn: !Sub arn:${AWS::Partition}:ecs:${AWS::Region}:  
${AWS::AccountId}:*  
    StringEquals:  
      aws:SourceAccount: !Sub ${AWS::AccountId}  
Path: /  
ManagedPolicyArns:  
  - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy  
Outputs:  
  ClusterName:  
    Description: The ECS cluster into which to launch resources  
    Value: ECSCluster  
  ECSTaskExecutionRole:  
    Description: The role used to start up a task  
    Value: ECSTaskExecutionRole  
  CapacityProvider:  
    Description: The cluster capacity provider that the service should use to  
request capacity when it wants to start up a task  
    Value: CapacityProvider
```

## Services

You can use an Amazon ECS service to run and maintain a specified number of instances of a task definition simultaneously in an Amazon ECS cluster. If one of your tasks fails or stops, the Amazon ECS service scheduler launches another instance of your task definition to replace it. This helps maintain your desired number of tasks in the service. The following templates can be used to deploy services. For more information about Amazon ECS services, see [Amazon ECS services](#).

### Deploy a load balanced web application

The following template creates an Amazon ECS service with two tasks that run on Fargate. The tasks each have an NGINX container. The template also creates an Application Load Balancer that distributes application traffic and an Application Auto Scaling policy that scales the application based on CPU utilization. The template also creates the networking resources necessary to deploy the application, the logging resources for container logs, and an Amazon ECS task execution IAM role. For more information about the task execution role, see [Amazon ECS task execution IAM role](#). For more information about auto scaling, see [Automatically scale your Amazon ECS service](#).

JSON

```
{  
    "AWSTemplateFormatVersion": "2010-09-09",  
    "Description": "[AWS Docs] ECS: load-balanced-web-application",  
    "Parameters": {  
        "VpcCidr": {  
            "Type": "String",  
            "Default": "10.0.0.0/16",  
            "Description": "CIDR block for the VPC"  
        },  
        "ContainerImage": {  
            "Type": "String",  
            "Default": "public.ecr.aws/ecs-sample-image/amazon-ecs-sample:latest",  
            "Description": "Container image to use in task definition"  
        },  
        "PublicSubnet1Cidr": {  
            "Type": "String",  
            "Default": "10.0.1.0/24",  
            "Description": "CIDR block for public subnet 1"  
        },  
        "PublicSubnet2Cidr": {  
            "Type": "String",  
            "Default": "10.0.2.0/24",  
            "Description": "CIDR block for public subnet 2"  
        }  
    },  
    "Resources": {  
        "ALB": {  
            "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",  
            "Properties": {  
                "Name": "my-load-balancer",  
                "Protocol": "HTTP",  
                "Port": 80, "TargetGroupArn": { "Fn::GetAtt": "TaskTargetGroup", "Path": "TargetGroupArn" },  
                "HealthCheck": {  
                    "Interval": 30, "Timeout": 5, "Path": "/healthcheck", "Port": "80", "Protocol": "HTTP"  
                },  
                "Subnets": { "Fn::GetAZs": "AWS::Region" }  
            }  
        },  
        "TaskDefinition": {  
            "Type": "AWS::ECS::TaskDefinition",  
            "Properties": {  
                "ContainerDefinitions": [ {  
                    "Name": "nginx",  
                    "Image": { "Fn::Ref": "ContainerImage" },  
                    "PortMappings": [ { "ContainerPort": 80, "HostPort": 80 } ]  
                } ],  
                "Family": "my-task-definition",  
                "Memory": 512,  
                "NetworkMode": "awsvpc",  
                "RequiresCompatibilities": [ "FARGATE" ],  
                "TaskRoleArn": { "Fn::GetAtt": "TaskExecutionRole", "Path": "Arn" }  
            }  
        },  
        "TaskTargetGroup": {  
            "Type": "AWS::ECS::TargetGroup",  
            "Properties": {  
                "Port": 80, "Protocol": "HTTP", "TargetType": "IP",  
                "VpcId": { "Fn::GetAtt": "Vpc", "Path": "VpcId" }  
            }  
        },  
        "TaskExecutionRole": {  
            "Type": "AWS::IAM::Role",  
            "Properties": {  
                "AssumeRolePolicyDocument": {  
                    "Statement": [ {  
                        "Action": "sts:AssumeRole",  
                        "Effect": "Allow", "Principal": "task.amazonaws.com"  
                    } ]  
                },  
                "Policies": [ {  
                    "PolicyName": "TaskExecutionPolicy",  
                    "PolicyDocument": {  
                        "Statement": [ {  
                            "Action": "logs:CreateLogGroup",  
                            "Effect": "Allow", "Resource": "arn:aws:logs:us-east-1:  
                        } ]  
                    } ]  
                }  
            }  
        },  
        "Vpc": {  
            "Type": "AWS::EC2::VPC",  
            "Properties": {  
                "CidrBlock": { "Fn::Ref": "VpcCidr" },  
                "EnableDnsSupport": true, "EnableDnsResolution": true,  
                "InstanceTenancy": "default",  
                "SubnetMappings": [ {  
                    "SubnetType": "public", "VpcId": { "Fn::Ref": "Vpc" },  
                    "AllocationId": { "Fn::GetAtt": "PublicSubnet1", "Path": "AllocationId" },  
                    "Primary": true  
                }, {  
                    "SubnetType": "public", "VpcId": { "Fn::Ref": "Vpc" },  
                    "AllocationId": { "Fn::GetAtt": "PublicSubnet2", "Path": "AllocationId" }  
                } ]  
            }  
        },  
        "PublicSubnet1": {  
            "Type": "AWS::EC2::Subnet",  
            "Properties": {  
                "CidrBlock": { "Fn::Ref": "PublicSubnet1Cidr" },  
                "VpcId": { "Fn::Ref": "Vpc" }  
            }  
        },  
        "PublicSubnet2": {  
            "Type": "AWS::EC2::Subnet",  
            "Properties": {  
                "CidrBlock": { "Fn::Ref": "PublicSubnet2Cidr" },  
                "VpcId": { "Fn::Ref": "Vpc" }  
            }  
        }  
    },  
    "Outputs": {  
        "TaskDefinitionArn": {  
            "Description": "ARN of the task definition.",  
            "Value": { "Fn::GetAtt": "TaskDefinition", "Path": "Arn" }  
        },  
        "TaskExecutionRoleArn": {  
            "Description": "ARN of the task execution role.",  
            "Value": { "Fn::GetAtt": "TaskExecutionRole", "Path": "Arn" }  
        },  
        "LoadBalancerArn": {  
            "Description": "ARN of the application load balancer.",  
            "Value": { "Fn::GetAtt": "ALB", "Path": "Arn" }  
        }  
    }  
}
```

```
        "Default": "10.0.2.0/24",
        "Description": "CIDR block for public subnet 2"
    },
    "PrivateSubnet1Cidr": {
        "Type": "String",
        "Default": "10.0.3.0/24",
        "Description": "CIDR block for private subnet 1"
    },
    "PrivateSubnet2Cidr": {
        "Type": "String",
        "Default": "10.0.4.0/24",
        "Description": "CIDR block for private subnet 2"
    },
    "ServiceName": {
        "Type": "String",
        "Default": "tutorial-app",
        "Description": "Name of the ECS service"
    },
    "ContainerPort": {
        "Type": "Number",
        "Default": 80,
        "Description": "Port on which the container listens"
    },
    "DesiredCount": {
        "Type": "Number",
        "Default": 2,
        "Description": "Desired number of tasks"
    },
    "MinCapacity": {
        "Type": "Number",
        "Default": 1,
        "Description": "Minimum number of tasks for auto scaling"
    },
    "MaxCapacity": {
        "Type": "Number",
        "Default": 10,
        "Description": "Maximum number of tasks for auto scaling"
    }
},
"Resources": {
    "VPC": {
        "Type": "AWS::EC2::VPC",
        "Properties": {
            "CidrBlock": {
```

```
        "Ref": "VpcCidr"
    },
    "EnableDnsHostnames": true,
    "EnableDnsSupport": true,
    "Tags": [
        {
            "Key": "Name",
            "Value": {
                "Fn::Sub": "${AWS::StackName}-vpc"
            }
        }
    ]
},
"InternetGateway": {
    "Type": "AWS::EC2::InternetGateway",
    "Properties": {
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-igw"
                }
            }
        ]
    }
},
"InternetGatewayAttachment": {
    "Type": "AWS::EC2::VPCGatewayAttachment",
    "Properties": {
        "InternetGatewayId": {
            "Ref": "InternetGateway"
        },
        "VpcId": {
            "Ref": "VPC"
        }
    }
},
"PublicSubnet1": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
        "VpcId": {
            "Ref": "VPC"
        }
    }
},
```

```
        "AvailabilityZone": {
            "Fn::Select": [
                0,
                {
                    "Fn::GetAZs": ""
                }
            ]
        },
        "CidrBlock": {
            "Ref": "PublicSubnet1Cidr"
        },
        "MapPublicIpOnLaunch": true,
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-public-subnet-1"
                }
            }
        ]
    },
    "PublicSubnet2": {
        "Type": "AWS::EC2::Subnet",
        "Properties": {
            "VpcId": {
                "Ref": "VPC"
            },
            "AvailabilityZone": {
                "Fn::Select": [
                    1,
                    {
                        "Fn::GetAZs": ""
                    }
                ]
            },
            "CidrBlock": {
                "Ref": "PublicSubnet2Cidr"
            },
            "MapPublicIpOnLaunch": true,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": {

```

```
        "Fn::Sub": "${AWS::StackName}-public-subnet-2"
    }
}
],
},
"PrivateSubnet1": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
        "VpcId": {
            "Ref": "VPC"
        },
        "AvailabilityZone": {
            "Fn::Select": [
                0,
                {
                    "Fn::GetAZs": ""
                }
            ]
        },
        "CidrBlock": {
            "Ref": "PrivateSubnet1Cidr"
        },
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-private-subnet-1"
                }
            }
        ]
    }
},
"PrivateSubnet2": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
        "VpcId": {
            "Ref": "VPC"
        },
        "AvailabilityZone": {
            "Fn::Select": [
                1,
                {
                    "Fn::GetAZs": ""
                }
            ]
        }
    }
}
```

```
        }
    ],
},
"CidrBlock": {
    "Ref": "PrivateSubnet2Cidr"
},
"Tags": [
    {
        "Key": "Name",
        "Value": {
            "Fn::Sub": "${AWS::StackName}-private-subnet-2"
        }
    }
]
},
"NatGateway1EIP": {
    "Type": "AWS::EC2::EIP",
    "DependsOn": "InternetGatewayAttachment",
    "Properties": {
        "Domain": "vpc",
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-nat-eip-1"
                }
            }
        ]
    }
},
"NatGateway2EIP": {
    "Type": "AWS::EC2::EIP",
    "DependsOn": "InternetGatewayAttachment",
    "Properties": {
        "Domain": "vpc",
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-nat-eip-2"
                }
            }
        ]
    }
}
```

```
        },
    },
    "NatGateway1": {
        "Type": "AWS::EC2::NatGateway",
        "Properties": {
            "AllocationId": {
                "Fn::GetAtt": [
                    "NatGateway1EIP",
                    "AllocationId"
                ]
            },
            "SubnetId": {
                "Ref": "PublicSubnet1"
            },
            "Tags": [
                {
                    "Key": "Name",
                    "Value": {
                        "Fn::Sub": "${AWS::StackName}-nat-1"
                    }
                }
            ]
        }
    },
    "NatGateway2": {
        "Type": "AWS::EC2::NatGateway",
        "Properties": {
            "AllocationId": {
                "Fn::GetAtt": [
                    "NatGateway2EIP",
                    "AllocationId"
                ]
            },
            "SubnetId": {
                "Ref": "PublicSubnet2"
            },
            "Tags": [
                {
                    "Key": "Name",
                    "Value": {
                        "Fn::Sub": "${AWS::StackName}-nat-2"
                    }
                }
            ]
        }
    }
}
```

```
        },
    },
    "PublicRouteTable": {
        "Type": "AWS::EC2::RouteTable",
        "Properties": {
            "VpcId": {
                "Ref": "VPC"
            },
            "Tags": [
                {
                    "Key": "Name",
                    "Value": {
                        "Fn::Sub": "${AWS::StackName}-public-routes"
                    }
                }
            ]
        }
    },
    "DefaultPublicRoute": {
        "Type": "AWS::EC2::Route",
        "DependsOn": "InternetGatewayAttachment",
        "Properties": {
            "RouteTableId": {
                "Ref": "PublicRouteTable"
            },
            "DestinationCidrBlock": "0.0.0.0/0",
            "GatewayId": {
                "Ref": "InternetGateway"
            }
        }
    },
    "PublicSubnet1RouteTableAssociation": {
        "Type": "AWS::EC2::SubnetRouteTableAssociation",
        "Properties": {
            "RouteTableId": {
                "Ref": "PublicRouteTable"
            },
            "SubnetId": {
                "Ref": "PublicSubnet1"
            }
        }
    },
    "PublicSubnet2RouteTableAssociation": {
        "Type": "AWS::EC2::SubnetRouteTableAssociation",
```

```
    "Properties": {
        "RouteTableId": {
            "Ref": "PublicRouteTable"
        },
        "SubnetId": {
            "Ref": "PublicSubnet2"
        }
    }
},
"PrivateRouteTable1": {
    "Type": "AWS::EC2::RouteTable",
    "Properties": {
        "VpcId": {
            "Ref": "VPC"
        },
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-private-routes-1"
                }
            }
        ]
    }
},
"DefaultPrivateRoute1": {
    "Type": "AWS::EC2::Route",
    "Properties": {
        "RouteTableId": {
            "Ref": "PrivateRouteTable1"
        },
        "DestinationCidrBlock": "0.0.0.0/0",
        "NatGatewayId": {
            "Ref": "NatGateway1"
        }
    }
},
"PrivateSubnet1RouteTableAssociation": {
    "Type": "AWS::EC2::SubnetRouteTableAssociation",
    "Properties": {
        "RouteTableId": {
            "Ref": "PrivateRouteTable1"
        },
        "SubnetId": {
```

```
        "Ref": "PrivateSubnet1"
    }
}
},
"PrivateRouteTable2": {
    "Type": "AWS::EC2::RouteTable",
    "Properties": {
        "VpcId": {
            "Ref": "VPC"
        },
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-private-routes-2"
                }
            }
        ]
    }
},
"DefaultPrivateRoute2": {
    "Type": "AWS::EC2::Route",
    "Properties": {
        "RouteTableId": {
            "Ref": "PrivateRouteTable2"
        },
        "DestinationCidrBlock": "0.0.0.0/0",
        "NatGatewayId": {
            "Ref": "NatGateway2"
        }
    }
},
"PrivateSubnet2RouteTableAssociation": {
    "Type": "AWS::EC2::SubnetRouteTableAssociation",
    "Properties": {
        "RouteTableId": {
            "Ref": "PrivateRouteTable2"
        },
        "SubnetId": {
            "Ref": "PrivateSubnet2"
        }
    }
},
"ALBSecurityGroup": {
```

```
"Type": "AWS::EC2::SecurityGroup",
"Properties": {
    "GroupName": {
        "Fn::Sub": "${AWS::StackName}-alb-sg"
    },
    "GroupDescription": "Security group for Application Load Balancer",
    "VpcId": {
        "Ref": "VPC"
    },
    "SecurityGroupIngress": [
        {
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIp": "0.0.0.0/0",
            "Description": "Allow HTTP traffic from internet"
        }
    ],
    "SecurityGroupEgress": [
        {
            "IpProtocol": -1,
            "CidrIp": "0.0.0.0/0",
            "Description": "Allow all outbound traffic"
        }
    ],
    "Tags": [
        {
            "Key": "Name",
            "Value": {
                "Fn::Sub": "${AWS::StackName}-alb-sg"
            }
        }
    ]
},
"ECSSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupName": {
            "Fn::Sub": "${AWS::StackName}-ecs-sg"
        },
        "GroupDescription": "Security group for ECS tasks",
        "VpcId": {
            "Ref": "VPC"
        }
    }
}
```

```
        },
        "SecurityGroupIngress": [
            {
                "IpProtocol": "tcp",
                "FromPort": {
                    "Ref": "ContainerPort"
                },
                "ToPort": {
                    "Ref": "ContainerPort"
                },
                "SourceSecurityGroupId": {
                    "Ref": "ALBSecurityGroup"
                },
                "Description": "Allow traffic from ALB"
            }
        ],
        "SecurityGroupEgress": [
            {
                "IpProtocol": -1,
                "CidrIp": "0.0.0.0/0",
                "Description": "Allow all outbound traffic"
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-ecs-sg"
                }
            }
        ]
    },
    "ApplicationLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Name": {
                "Fn::Sub": "${AWS::StackName}-alb"
            },
            "Scheme": "internet-facing",
            "Type": "application",
            "Subnets": [
                {
                    "Ref": "PublicSubnet1"
                }
            ]
        }
    }
},
```

```
        },
        {
            "Ref": "PublicSubnet2"
        }
    ],
    "SecurityGroups": [
        {
            "Ref": "ALBSecurityGroup"
        }
    ],
    "Tags": [
        {
            "Key": "Name",
            "Value": {
                "Fn::Sub": "${AWS::StackName}-alb"
            }
        }
    ]
},
"ALBTARGETGROUP": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
        "Name": {
            "Fn::Sub": "${AWS::StackName}-tg"
        },
        "Port": {
            "Ref": "ContainerPort"
        },
        "Protocol": "HTTP",
        "VpcId": {
            "Ref": "VPC"
        },
        "TargetType": "ip",
        "HealthCheckIntervalSeconds": 30,
        "HealthCheckPath": "/",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 5,
        "HealthyThresholdCount": 2,
        "UnhealthyThresholdCount": 5,
        "Tags": [
            {
                "Key": "Name",
                "Value": {

```

```
        "Fn::Sub": "${AWS::StackName}-tg"
    }
}
]
}
},
"ALBListener": {
    "Type": "AWS::ElasticLoadBalancingV2::Listener",
    "Properties": {
        "DefaultActions": [
            {
                "Type": "forward",
                "TargetGroupArn": {
                    "Ref": "ALBTTargetGroup"
                }
            }
        ],
        "LoadBalancerArn": {
            "Ref": "ApplicationLoadBalancer"
        },
        "Port": 80,
        "Protocol": "HTTP"
    }
},
"ECSCluster": {
    "Type": "AWS::ECS::Cluster",
    "Properties": {
        "ClusterName": {
            "Fn::Sub": "${AWS::StackName}-cluster"
        },
        "CapacityProviders": [
            "FARGATE",
            "FARGATE_SPOT"
        ],
        "DefaultCapacityProviderStrategy": [
            {
                "CapacityProvider": "FARGATE",
                "Weight": 1
            },
            {
                "CapacityProvider": "FARGATE_SPOT",
                "Weight": 4
            }
        ],
    }
},
```

```
"ClusterSettings": [
    {
        "Name": "containerInsights",
        "Value": "enabled"
    }
],
"Tags": [
    {
        "Key": "Name",
        "Value": {
            "Fn::Sub": "${AWS::StackName}-cluster"
        }
    }
]
},
"ECSTaskExecutionRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "RoleName": {
            "Fn::Sub": "${AWS::StackName}-task-execution-role"
        },
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ecs-tasks.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "ManagedPolicyArns": [
            "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-task-execution-role"
                }
            }
        ]
    }
}
```

```
        }
    ]
}
},
"ECSTaskRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "RoleName": {
            "Fn::Sub": "${AWS::StackName}-task-role"
        },
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ecs-tasks.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-task-role"
                }
            }
        ]
    }
},
"LogGroup": {
    "Type": "AWS::Logs::LogGroup",
    "Properties": {
        "LogGroupName": {
            "Fn::Sub": "/ecs/${AWS::StackName}"
        },
        "RetentionInDays": 7
    }
},
"TaskDefinition": {
    "Type": "AWS::ECS::TaskDefinition",
    "Properties": {
```

```
        "Family": {
            "Fn::Sub": "${AWS::StackName}-task"
        },
        "Cpu": "256",
        "Memory": "512",
        "NetworkMode": "awsvpc",
        "RequiresCompatibilities": [
            "FARGATE"
        ],
        "ExecutionRoleArn": {
            "Fn::GetAtt": [
                "ECSTaskExecutionRole",
                "Arn"
            ]
        },
        "TaskRoleArn": {
            "Fn::GetAtt": [
                "ECSTaskRole",
                "Arn"
            ]
        },
        "ContainerDefinitions": [
            {
                "Name": {
                    "Ref": "ServiceName"
                },
                "Image": {
                    "Ref": "ContainerImage"
                },
                "PortMappings": [
                    {
                        "ContainerPort": {
                            "Ref": "ContainerPort"
                        },
                        "Protocol": "tcp"
                    }
                ],
                "Essential": true,
                "LogConfiguration": {
                    "LogDriver": "awslogs",
                    "Options": {
                        "awslogs-group": {
                            "Ref": "LogGroup"
                        }
                    },
                    "LogGroup": {
                        "Ref": "LogGroup"
                    }
                }
            }
        ]
    }
}
```

```
        "awslogs-region": {
            "Ref": "AWS::Region"
        },
        "awslogs-stream-prefix": "ecs"
    }
},
"HealthCheck": {
    "Command": [
        "CMD-SHELL",
        "curl -f http://localhost/ || exit 1"
    ],
    "Interval": 30,
    "Timeout": 5,
    "Retries": 3,
    "StartPeriod": 60
}
}
],
"Tags": [
{
    "Key": "Name",
    "Value": {
        "Fn::Sub": "${AWS::StackName}-task"
    }
}
]
}
},
"ECSService": {
    "Type": "AWS::ECS::Service",
    "DependsOn": "ALBListener",
    "Properties": {
        "ServiceName": {
            "Fn::Sub": "${AWS::StackName}-service"
        },
        "Cluster": {
            "Ref": "ECScluster"
        },
        "TaskDefinition": {
            "Ref": "TaskDefinition"
        },
        "DesiredCount": {
            "Ref": "DesiredCount"
        },
        "MinCount": {
            "Ref": "MinCount"
        },
        "MaxCount": {
            "Ref": "MaxCount"
        }
    }
}
```

```
"LaunchType": "FARGATE",
"PlatformVersion": "LATEST",
"NetworkConfiguration": {
    "AwsVpcConfiguration": {
        "AssignPublicIp": "DISABLED",
        "SecurityGroups": [
            {
                "Ref": "ECSSecurityGroup"
            }
        ],
        "Subnets": [
            {
                "Ref": "PrivateSubnet1"
            },
            {
                "Ref": "PrivateSubnet2"
            }
        ]
    }
},
"LoadBalancers": [
    {
        "ContainerName": {
            "Ref": "ServiceName"
        },
        "ContainerPort": {
            "Ref": "ContainerPort"
        },
        "TargetGroupArn": {
            "Ref": "ALBTargetGroup"
        }
    }
],
"DeploymentConfiguration": {
    "MaximumPercent": 200,
    "MinimumHealthyPercent": 50,
    "DeploymentCircuitBreaker": {
        "Enable": true,
        "Rollback": true
    }
},
"EnableExecuteCommand": true,
"Tags": [
    {

```

```
        "Key": "Name",
        "Value": {
            "Fn::Sub": "${AWS::StackName}-service"
        }
    ],
}
},
"ServiceScalingTarget": {
    "Type": "AWS::ApplicationAutoScaling::ScalableTarget",
    "Properties": {
        "MaxCapacity": {
            "Ref": "MaxCapacity"
        },
        "MinCapacity": {
            "Ref": "MinCapacity"
        },
        "ResourceId": {
            "Fn::Sub": "service/${ECSCluster}/${ECSService.Name}"
        },
        "RoleARN": {
            "Fn::Sub": "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService"
        },
        "ScalableDimension": "ecs:service:DesiredCount",
        "ServiceNamespace": "ecs"
    }
},
"ServiceScalingPolicy": {
    "Type": "AWS::ApplicationAutoScaling::ScalingPolicy",
    "Properties": {
        "PolicyName": {
            "Fn::Sub": "${AWS::StackName}-cpu-scaling-policy"
        },
        "PolicyType": "TargetTrackingScaling",
        "ScalingTargetId": {
            "Ref": "ServiceScalingTarget"
        },
        "TargetTrackingScalingPolicyConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "ECSServiceAverageCPUUtilization"
            },
            "TargetValue": 70,
        }
    }
}
```

```
        "ScaleOutCooldown": 300,
        "ScaleInCooldown": 300
    }
}
},
"Outputs": {
    "VPCId": {
        "Description": "VPC ID",
        "Value": {
            "Ref": "VPC"
        },
        "Export": {
            "Name": {
                "Fn::Sub": "${AWS::StackName}-VPC-ID"
            }
        }
    },
    "LoadBalancerURL": {
        "Description": "URL of the Application Load Balancer",
        "Value": {
            "Fn::Sub": "http://${ApplicationLoadBalancer.DNSName}"
        },
        "Export": {
            "Name": {
                "Fn::Sub": "${AWS::StackName}-ALB-URL"
            }
        }
    },
    "ECSClusterName": {
        "Description": "Name of the ECS Cluster",
        "Value": {
            "Ref": "ECSCluster"
        },
        "Export": {
            "Name": {
                "Fn::Sub": "${AWS::StackName}-ECS-Cluster"
            }
        }
    },
    "ECSServiceName": {
        "Description": "Name of the ECS Service",
        "Value": {
            "Fn::GetAtt": [

```

```
        "ECSService",
        "Name"
    ],
},
"Export": {
    "Name": {
        "Fn::Sub": "${AWS::StackName}-ECS-Service"
    }
}
},
"PrivateSubnet1": {
    "Description": "Private Subnet 1 ID",
    "Value": {
        "Ref": "PrivateSubnet1"
    },
    "Export": {
        "Name": {
            "Fn::Sub": "${AWS::StackName}-Private-Subnet-1"
        }
    }
},
"PrivateSubnet2": {
    "Description": "Private Subnet 2 ID",
    "Value": {
        "Ref": "PrivateSubnet2"
    },
    "Export": {
        "Name": {
            "Fn::Sub": "${AWS::StackName}-Private-Subnet-2"
        }
    }
}
}
```

## YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: '[AWS Docs] ECS: load-balanced-web-application'

Parameters:
  VpcCidr:
    Type: String
```

```
  Default: '10.0.0.0/16'
  Description: CIDR block for the VPC
ContainerImage:
  Type: String
  Default: 'public.ecr.aws/ecs-sample-image/amazon-ecs-sample:latest'
  Description: Container image to use in task definition

PublicSubnet1Cidr:
  Type: String
  Default: '10.0.1.0/24'
  Description: CIDR block for public subnet 1

PublicSubnet2Cidr:
  Type: String
  Default: '10.0.2.0/24'
  Description: CIDR block for public subnet 2

PrivateSubnet1Cidr:
  Type: String
  Default: '10.0.3.0/24'
  Description: CIDR block for private subnet 1

PrivateSubnet2Cidr:
  Type: String
  Default: '10.0.4.0/24'
  Description: CIDR block for private subnet 2

ServiceName:
  Type: String
  Default: 'tutorial-app'
  Description: Name of the ECS service

ContainerPort:
  Type: Number
  Default: 80
  Description: Port on which the container listens

DesiredCount:
  Type: Number
  Default: 2
  Description: Desired number of tasks

MinCapacity:
  Type: Number
```

```
Default: 1
Description: Minimum number of tasks for auto scaling
```

```
MaxCapacity:
Type: Number
Default: 10
Description: Maximum number of tasks for auto scaling
```

#### Resources:

```
# VPC and Networking
```

##### VPC:

```
Type: AWS::EC2::VPC
Properties:
  CidrBlock: !Ref VpcCidr
  EnableDnsHostnames: true
  EnableDnsSupport: true
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-vpc'
```

```
# Internet Gateway
```

##### InternetGateway:

```
Type: AWS::EC2::InternetGateway
Properties:
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-igw'
```

##### InternetGatewayAttachment:

```
Type: AWS::EC2::VPGatewayAttachment
Properties:
  InternetGatewayId: !Ref InternetGateway
  VpcId: !Ref VPC
```

```
# Public Subnets for ALB
```

##### PublicSubnet1:

```
Type: AWS::EC2::Subnet
Properties:
  VpcId: !Ref VPC
  AvailabilityZone: !Select [0, !GetAZs '']
  CidrBlock: !Ref PublicSubnet1Cidr
  MapPublicIpOnLaunch: true
  Tags:
    - Key: Name
```

```
Value: !Sub '${AWS::StackName}-public-subnet-1'

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs '']
    CidrBlock: !Ref PublicSubnet2Cidr
    MapPublicIpOnLaunch: true
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-public-subnet-2'

# Private Subnets for ECS Tasks
PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [0, !GetAZs '']
    CidrBlock: !Ref PrivateSubnet1Cidr
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-private-subnet-1'

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs '']
    CidrBlock: !Ref PrivateSubnet2Cidr
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-private-subnet-2'

# NAT Gateways for private subnet internet access
NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-nat-eip-1'
```

```
NatGateway2EIP:  
  Type: AWS::EC2::EIP  
  DependsOn: InternetGatewayAttachment  
  Properties:  
    Domain: vpc  
  Tags:  
    - Key: Name  
      Value: !Sub '${AWS::StackName}-nat-eip-2'  
  
NatGateway1:  
  Type: AWS::EC2::NatGateway  
  Properties:  
    AllocationId: !GetAtt NatGateway1EIP.AllocationId  
    SubnetId: !Ref PublicSubnet1  
  Tags:  
    - Key: Name  
      Value: !Sub '${AWS::StackName}-nat-1'  
  
NatGateway2:  
  Type: AWS::EC2::NatGateway  
  Properties:  
    AllocationId: !GetAtt NatGateway2EIP.AllocationId  
    SubnetId: !Ref PublicSubnet2  
  Tags:  
    - Key: Name  
      Value: !Sub '${AWS::StackName}-nat-2'  
  
# Route Tables  
PublicRouteTable:  
  Type: AWS::EC2::RouteTable  
  Properties:  
    VpcId: !Ref VPC  
  Tags:  
    - Key: Name  
      Value: !Sub '${AWS::StackName}-public-routes'  
  
DefaultPublicRoute:  
  Type: AWS::EC2::Route  
  DependsOn: InternetGatewayAttachment  
  Properties:  
    RouteTableId: !Ref PublicRouteTable  
    DestinationCidrBlock: 0.0.0.0/0  
    GatewayId: !Ref InternetGateway
```

```
PublicSubnet1RouteTableAssociation:  
  Type: AWS::EC2::SubnetRouteTableAssociation  
  Properties:  
    RouteTableId: !Ref PublicRouteTable  
    SubnetId: !Ref PublicSubnet1  
  
PublicSubnet2RouteTableAssociation:  
  Type: AWS::EC2::SubnetRouteTableAssociation  
  Properties:  
    RouteTableId: !Ref PublicRouteTable  
    SubnetId: !Ref PublicSubnet2  
  
PrivateRouteTable1:  
  Type: AWS::EC2::RouteTable  
  Properties:  
    VpcId: !Ref VPC  
    Tags:  
      - Key: Name  
        Value: !Sub '${AWS::StackName}-private-routes-1'  
  
DefaultPrivateRoute1:  
  Type: AWS::EC2::Route  
  Properties:  
    RouteTableId: !Ref PrivateRouteTable1  
    DestinationCidrBlock: 0.0.0.0/0  
    NatGatewayId: !Ref NatGateway1  
  
PrivateSubnet1RouteTableAssociation:  
  Type: AWS::EC2::SubnetRouteTableAssociation  
  Properties:  
    RouteTableId: !Ref PrivateRouteTable1  
    SubnetId: !Ref PrivateSubnet1  
  
PrivateRouteTable2:  
  Type: AWS::EC2::RouteTable  
  Properties:  
    VpcId: !Ref VPC  
    Tags:  
      - Key: Name  
        Value: !Sub '${AWS::StackName}-private-routes-2'  
  
DefaultPrivateRoute2:  
  Type: AWS::EC2::Route  
  Properties:
```

```
RouteTableId: !Ref PrivateRouteTable2
DestinationCidrBlock: 0.0.0.0/0
NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:
Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
  RouteTableId: !Ref PrivateRouteTable2
  SubnetId: !Ref PrivateSubnet2

# Security Groups
ALBSecurityGroup:
Type: AWS::EC2::SecurityGroup
Properties:
  GroupName: !Sub '${AWS::StackName}-alb-sg'
  GroupDescription: Security group for Application Load Balancer
  VpcId: !Ref VPC
  SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 80
      ToPort: 80
      CidrIp: 0.0.0.0/0
      Description: Allow HTTP traffic from internet
  SecurityGroupEgress:
    - IpProtocol: -1
      CidrIp: 0.0.0.0/0
      Description: Allow all outbound traffic
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-alb-sg'

ECSSecurityGroup:
Type: AWS::EC2::SecurityGroup
Properties:
  GroupName: !Sub '${AWS::StackName}-ecs-sg'
  GroupDescription: Security group for ECS tasks
  VpcId: !Ref VPC
  SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: !Ref ContainerPort
      ToPort: !Ref ContainerPort
      SourceSecurityGroupId: !Ref ALBSecurityGroup
      Description: Allow traffic from ALB
  SecurityGroupEgress:
```

```
- IpProtocol: -1
  CidrIp: 0.0.0.0/0
  Description: Allow all outbound traffic
Tags:
- Key: Name
  Value: !Sub '${AWS::StackName}-ecs-sg'

# Application Load Balancer
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: !Sub '${AWS::StackName}-alb'
    Scheme: internet-facing
    Type: application
    Subnets:
      - !Ref PublicSubnet1
      - !Ref PublicSubnet2
    SecurityGroups:
      - !Ref ALBSecurityGroup
  Tags:
- Key: Name
  Value: !Sub '${AWS::StackName}-alb'

ALBTARGETGROUP:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    Name: !Sub '${AWS::StackName}-tg'
    Port: !Ref ContainerPort
    Protocol: HTTP
    VpcId: !Ref VPC
    TargetType: ip
    HealthCheckIntervalSeconds: 30
    HealthCheckPath: /
    HealthCheckProtocol: HTTP
    HealthCheckTimeoutSeconds: 5
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 5
  Tags:
- Key: Name
  Value: !Sub '${AWS::StackName}-tg'

ALBLISTENER:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
```

```
DefaultActions:
  - Type: forward
    TargetGroupArn: !Ref ALBTargetGroup
    LoadBalancerArn: !Ref ApplicationLoadBalancer
    Port: 80
    Protocol: HTTP

# ECS Cluster
ECSCluster:
  Type: AWS::ECS::Cluster
  Properties:
    ClusterName: !Sub '${AWS::StackName}-cluster'
    CapacityProviders:
      - FARGATE
      - FARGATE_SPOT
    DefaultCapacityProviderStrategy:
      - CapacityProvider: FARGATE
        Weight: 1
      - CapacityProvider: FARGATE_SPOT
        Weight: 4
    ClusterSettings:
      - Name: containerInsights
        Value: enabled
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-cluster'

# IAM Roles
ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Sub '${AWS::StackName}-task-execution-role'
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-task-execution-role'
```

```
ECSTaskRole:  
  Type: AWS::IAM::Role  
  Properties:  
    RoleName: !Sub '${AWS::StackName}-task-role'  
    AssumeRolePolicyDocument:  
      Version: '2012-10-17'  
      Statement:  
        - Effect: Allow  
        Principal:  
          Service: ecs-tasks.amazonaws.com  
        Action: sts:AssumeRole  
    Tags:  
      - Key: Name  
      Value: !Sub '${AWS::StackName}-task-role'  
  
# CloudWatch Log Group  
LogGroup:  
  Type: AWS::Logs::LogGroup  
  Properties:  
    LogGroupName: !Sub '/ecs/${AWS::StackName}'  
    RetentionInDays: 7  
  
# ECS Task Definition  
TaskDefinition:  
  Type: AWS::ECS::TaskDefinition  
  Properties:  
    Family: !Sub '${AWS::StackName}-task'  
    Cpu: '256'  
    Memory: '512'  
    NetworkMode: awsvpc  
    RequiresCompatibilities:  
      - FARGATE  
    ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn  
    TaskRoleArn: !GetAtt ECSTaskRole.Arn  
    ContainerDefinitions:  
      - Name: !Ref ServiceName  
        Image: !Ref ContainerImage  
        PortMappings:  
          - ContainerPort: !Ref ContainerPort  
            Protocol: tcp  
        Essential: true  
    LogConfiguration:  
      LogDriver: awslogs
```

```
    Options:
        awslogs-group: !Ref LogGroup
        awslogs-region: !Ref AWS::Region
        awslogs-stream-prefix: ecs
    HealthCheck:
        Command:
            - CMD-SHELL
            - curl -f http://localhost/ || exit 1
        Interval: 30
        Timeout: 5
        Retries: 3
        StartPeriod: 60
    Tags:
        - Key: Name
          Value: !Sub '${AWS::StackName}-task'

# ECS Service
ECSService:
    Type: AWS::ECS::Service
    DependsOn: ALBListener
    Properties:
        ServiceName: !Sub '${AWS::StackName}-service'
        Cluster: !Ref ECSCluster
        TaskDefinition: !Ref TaskDefinition
        DesiredCount: !Ref DesiredCount
        LaunchType: FARGATE
        PlatformVersion: LATEST
        NetworkConfiguration:
            AwsVpcConfiguration:
                AssignPublicIp: DISABLED
                SecurityGroups:
                    - !Ref ECSSecurityGroup
            Subnets:
                - !Ref PrivateSubnet1
                - !Ref PrivateSubnet2
        LoadBalancers:
            - ContainerName: !Ref ServiceName
              ContainerPort: !Ref ContainerPort
              TargetGroupArn: !Ref ALBTARGETGroup
        DeploymentConfiguration:
            MaximumPercent: 200
            MinimumHealthyPercent: 50
        DeploymentCircuitBreaker:
            Enable: true
```

```
    Rollback: true
    EnableExecuteCommand: true # For debugging
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-service'

# Auto Scaling Target
ServiceScalingTarget:
  Type: AWS::ApplicationAutoScaling::ScalableTarget
  Properties:
    MaxCapacity: !Ref MaxCapacity
    MinCapacity: !Ref MinCapacity
    ResourceId: !Sub 'service/${ECSCluster}/${ECSService.Name}'
    RoleARN: !Sub 'arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService'
    ScalableDimension: ecs:service:DesiredCount
    ServiceNamespace: ecs

# Auto Scaling Policy - CPU Utilization
ServiceScalingPolicy:
  Type: AWS::ApplicationAutoScaling::ScalingPolicy
  Properties:
    PolicyName: !Sub '${AWS::StackName}-cpu-scaling-policy'
    PolicyType: TargetTrackingScaling
    ScalingTargetId: !Ref ServiceScalingTarget
    TargetTrackingScalingPolicyConfiguration:
      PredefinedMetricSpecification:
        PredefinedMetricType: ECSServiceAverageCPUUtilization
        TargetValue: 70.0
        ScaleOutCooldown: 300
        ScaleInCooldown: 300

Outputs:
  VPCId:
    Description: VPC ID
    Value: !Ref VPC
    Export:
      Name: !Sub '${AWS::StackName}-VPC-ID'

  LoadBalancerURL:
    Description: URL of the Application Load Balancer
    Value: !Sub 'http://${ApplicationLoadBalancer.DNSName}'
    Export:
```

```
Name: !Sub '${AWS::StackName}-ALB-URL'

ECSClusterName:
  Description: Name of the ECS Cluster
  Value: !Ref ECSCluster
  Export:
    Name: !Sub '${AWS::StackName}-ECS-Cluster'

ECSServiceName:
  Description: Name of the ECS Service
  Value: !GetAtt ECSService.Name
  Export:
    Name: !Sub '${AWS::StackName}-ECS-Service'

PrivateSubnet1:
  Description: Private Subnet 1 ID
  Value: !Ref PrivateSubnet1
  Export:
    Name: !Sub '${AWS::StackName}-Private-Subnet-1'

PrivateSubnet2:
  Description: Private Subnet 2 ID
  Value: !Ref PrivateSubnet2
  Export:
    Name: !Sub '${AWS::StackName}-Private-Subnet-2'
```

## Deploy a service with ECS Exec enabled

You can use the following template to deploy a service with ECS Exec enabled. The service runs in a cluster with a KMS key for encrypting ECS Exec sessions and a log configuration for redirecting execute command session logs to an Amazon S3 bucket. For more information, see [Monitor Amazon ECS containers with ECS Exec](#).

### JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "ECS Cluster with Fargate Service and Task Definition and ECS Exec enabled.",
  "Parameters": {
    "ClusterName": {
      "Type": "String",
```

```
        "Default": "CFNCluster",
        "Description": "Name of the ECS Cluster"
    },
    "S3BucketName": {
        "Type": "String",
        "Default": "amzn-s3-demo-bucket",
        "Description": "S3 bucket for ECS execute command logs"
    },
    "KmsKeyId": {
        "Type": "String",
        "Default": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "Description": "KMS Key ID for ECS execute command encryption"
    },
    "ContainerImage": {
        "Type": "String",
        "Default": "public.ecr.aws/docker/library/httpd:2.4",
        "Description": "Container image to use for the task"
    },
    "ContainerCpu": {
        "Type": "Number",
        "Default": 256,
        "AllowedValues": [256, 512, 1024, 2048, 4096],
        "Description": "CPU units for the container (256 = 0.25 vCPU)"
    },
    "ContainerMemory": {
        "Type": "Number",
        "Default": 512,
        "AllowedValues": [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192],
        "Description": "Memory for the container (in MiB)"
    },
    "DesiredCount": {
        "Type": "Number",
        "Default": 1,
        "Description": "Desired count of tasks in the service"
    },
    "SecurityGroups": {
        "Type": "List<AWS::EC2::SecurityGroup::Id>",
        "Description": "Security Group IDs for the ECS Service"
    },
    "Subnets": {
        "Type": "List<AWS::EC2::Subnet::Id>",
        "Description": "Subnet IDs for the ECS Service"
    },
    "ServiceName": {
```

```
        "Type": "String",
        "Default": "cfn-service",
        "Description": "Name of the ECS service"
    },
    "TaskFamily": {
        "Type": "String",
        "Default": "task-definition-cfn",
        "Description": "Family name for the task definition"
    },
    "TaskExecutionRoleArn": {
        "Type": "String",
        "Description": "ARN of an existing IAM role for ECS task execution",
        "Default": "arn:aws:iam::1112222333:role/ecsTaskExecutionRole"
    },
    "TaskRoleArn": {
        "Type": "String",
        "Description": "ARN of an existing IAM role for ECS tasks",
        "Default": "arn:aws:iam::1112222333:role/execTaskRole"
    }
},
"Resources": {
    "ECSCluster": {
        "Type": "AWS::ECS::Cluster",
        "Properties": {
            "ClusterName": {"Ref": "ClusterName"},
            "Configuration": {
                "ExecuteCommandConfiguration": {
                    "Logging": "OVERRIDE",
                    "LogConfiguration": {
                        "S3BucketName": {"Ref": "S3BucketName"}
                    },
                    "KmsKeyId": {"Ref": "KmsKeyId"}
                }
            },
            "Tags": [
                {
                    "Key": "Environment",
                    "Value": {"Ref": "AWS::StackName"}
                }
            ]
        }
    },
    "ECSTaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "ContainerDefinitions": [
                {
                    "Name": "mycontainer",
                    "Image": "myimage:latest",
                    "Memory": 128,
                    "Essential": true,
                    "PortMappings": [
                        {
                            "ContainerPort": 8080
                        }
                    ],
                    "Environment": [
                        {
                            "Name": "MYENV",
                            "Value": "myvalue"
                        }
                    ],
                    "MountPoints": [
                        {
                            "ContainerPath": "/var/www/html",
                            "HostPath": "/var/www/html",
                            "ReadOnly": false
                        }
                    ],
                    "LogConfiguration": {
                        "LogDriver": "awslogs",
                        "Options": {
                            "awslogs-region": "us-east-1",
                            "awslogs-group": "mygroup",
                            "awslogs-stream-prefix": "mystream"
                        }
                    }
                }
            ],
            "Memory": 512,
            "NetworkMode": "awsvpc",
            "TaskRoleArn": {"Ref": "TaskRoleArn"}
        }
    }
}
```

```
"Properties": {
    "ContainerDefinitions": [
        {
            "Command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon ECS.</p> </div></body></html>' > /usr/local/apache2/htdocs/index.html && httpd-foreground\""
            ],
            "EntryPoint": [
                "sh",
                "-c"
            ],
            "Essential": true,
            "Image": {"Ref": "ContainerImage"},
            "LogConfiguration": {
                "LogDriver": "awslogs",
                "Options": {
                    "mode": "non-blocking",
                    "max-buffer-size": "25m",
                    "awslogs-create-group": "true",
                    "awslogs-group": {"Fn::Sub": "/ecs/${AWS::StackName}"},
                    "awslogs-region": {"Ref": "AWS::Region"},
                    "awslogs-stream-prefix": "ecs"
                }
            },
            "Name": "sample-fargate-app",
            "PortMappings": [
                {
                    "ContainerPort": 80,
                    "HostPort": 80,
                    "Protocol": "tcp"
                }
            ]
        }
    ],
    "Cpu": {"Ref": "ContainerCpu"},
    "ExecutionRoleArn": {"Ref": "TaskExecutionRoleArn"},
    "TaskRoleArn": {"Ref": "TaskRoleArn"},
    "Family": {"Ref": "TaskFamily"},
    "Memory": {"Ref": "ContainerMemory"},
```

```
        "NetworkMode": "awsvpc",
        "RequiresCompatibilities": [
            "FARGATE"
        ],
        "RuntimePlatform": {
            "OperatingSystemFamily": "LINUX"
        },
        "Tags": [
            {
                "Key": "Name",
                "Value": {"Fn::Sub": "${AWS::StackName}-TaskDefinition"}
            }
        ]
    },
    "ECSService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
            "ServiceName": {"Ref": "ServiceName"},
            "Cluster": {"Ref": "ECSCluster"},
            "DesiredCount": {"Ref": "DesiredCount"},
            "EnableExecuteCommand": true,
            "LaunchType": "FARGATE",
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "AssignPublicIp": "ENABLED",
                    "SecurityGroups": {"Ref": "SecurityGroups"},
                    "Subnets": {"Ref": "Subnets"}
                }
            },
            "TaskDefinition": {"Ref": "ECSTaskDefinition"},
            "Tags": [
                {
                    "Key": "Name",
                    "Value": {"Fn::Sub": "${AWS::StackName}-Service"}
                }
            ]
        }
    },
    "Outputs": {
        "ClusterName": {
            "Description": "The name of the ECS cluster",
            "Value": {"Ref": "ECSCluster"}
        }
    }
},
```

```
  },
  "ServiceName": {
    "Description": "The name of the ECS service",
    "Value": {"Ref": "ServiceName"}
  },
  "TaskDefinitionArn": {
    "Description": "The ARN of the task definition",
    "Value": {"Ref": "ECSTaskDefinition"}
  },
  "ClusterArn": {
    "Description": "The ARN of the ECS cluster",
    "Value": {"Fn::GetAtt": ["ECSCluster", "Arn"]}
  },
  "StackName": {
    "Description": "The name of this stack",
    "Value": {"Ref": "AWS::StackName"}
  },
  "StackId": {
    "Description": "The unique identifier for this stack",
    "Value": {"Ref": "AWS::StackId"}
  },
  "Region": {
    "Description": "The AWS Region where the stack is deployed",
    "Value": {"Ref": "AWS::Region"}
  },
  "AccountId": {
    "Description": "The AWS Account ID",
    "Value": {"Ref": "AWS::AccountId"}
  }
}
}
```

## YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: ECS Cluster with Fargate Service and Task Definition and ECS Exec
enabled.
Parameters:
  ClusterName:
    Type: String
    Default: CFNCluster
    Description: Name of the ECS Cluster
  S3BucketName:
```

```
Type: String
Default: amzn-s3-demo-bucket
Description: S3 bucket for ECS execute command logs

KmsKeyId:
Type: String
Default: a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
Description: KMS Key ID for ECS execute command encryption

ContainerImage:
Type: String
Default: public.ecr.aws/docker/library/httpd:2.4
Description: Container image to use for the task

ContainerCpu:
Type: Number
Default: 256
AllowedValues: [256, 512, 1024, 2048, 4096]
Description: CPU units for the container (256 = 0.25 vCPU)

ContainerMemory:
Type: Number
Default: 512
AllowedValues: [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]
Description: Memory for the container (in MiB)

DesiredCount:
Type: Number
Default: 1
Description: Desired count of tasks in the service

SecurityGroups:
Type: List<AWS::EC2::SecurityGroup::Id>
Description: Security Group IDs for the ECS Service

Subnets:
Type: List<AWS::EC2::Subnet::Id>
Description: Subnet IDs for the ECS Service

ServiceName:
Type: String
Default: cfn-service
Description: Name of the ECS service

TaskFamily:
Type: String
Default: task-definition-cfn
Description: Family name for the task definition

TaskExecutionRoleArn:
Type: String
Description: ARN of an existing IAM role for ECS task execution
Default: 'arn:aws:iam::111122223333:role/ecsTaskExecutionRole'

TaskRoleArn:
```

```
Type: String
Description: ARN of an existing IAM role for ECS tasks
Default: 'arn:aws:iam::111122223333:role/execTaskRole'

Resources:
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: !Ref ClusterName
      Configuration:
        ExecuteCommandConfiguration:
          Logging: OVERRIDE
          LogConfiguration:
            S3BucketName: !Ref S3BucketName
            KmsKeyId: !Ref KmsKeyId
      Tags:
        - Key: Environment
          Value: !Ref AWS::StackName

  ECSTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
        - Command:
          - >-
            /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon ECS.</p> </div></body></html>' > /usr/local/apache2/htdocs/index.html && httpd-foreground"
      EntryPoint:
        - sh
        - '-c'
      Essential: true
      Image: !Ref ContainerImage
      LogConfiguration:
        LogDriver: awslogs
      Options:
        mode: non-blocking
        max-buffer-size: 25m
        awslogs-create-group: 'true'
        awslogs-group: !Sub /ecs/${AWS::StackName}
        awslogs-region: !Ref AWS::Region
        awslogs-stream-prefix: ecs
```

```
Name: sample-fargate-app
PortMappings:
  - ContainerPort: 80
    HostPort: 80
    Protocol: tcp
Cpu: !Ref ContainerCpu
ExecutionRoleArn: !Ref TaskExecutionRoleArn
TaskRoleArn: !Ref TaskRoleArn
Family: !Ref TaskFamily
Memory: !Ref ContainerMemory
NetworkMode: awsvpc
RequiresCompatibilities:
  - FARGATE
RuntimePlatform:
  OperatingSystemFamily: LINUX
Tags:
  - Key: Name
    Value: !Sub ${AWS::StackName}-TaskDefinition
ECSService:
  Type: AWS::ECS::Service
  Properties:
    ServiceName: !Ref ServiceName
    Cluster: !Ref ECSCluster
    DesiredCount: !Ref DesiredCount
    EnableExecuteCommand: true
    LaunchType: FARGATE
    NetworkConfiguration:
      AwsVpcConfiguration:
        AssignPublicIp: ENABLED
        SecurityGroups: !Ref SecurityGroups
        Subnets: !Ref Subnets
    TaskDefinition: !Ref ECSTaskDefinition
    Tags:
      - Key: Name
        Value: !Sub ${AWS::StackName}-Service
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref ECSCluster
  ServiceName:
    Description: The name of the ECS service
    Value: !Ref ServiceName
  TaskDefinitionArn:
    Description: The ARN of the task definition
```

```
  Value: !Ref ECSTaskDefinition
ClusterArn:
  Description: The ARN of the ECS cluster
  Value: !GetAtt ECSCluster.Arn
StackName:
  Description: The name of this stack
  Value: !Ref AWS::StackName
StackId:
  Description: The unique identifier for this stack
  Value: !Ref AWS::StackId
Region:
  Description: The AWS Region where the stack is deployed
  Value: !Ref AWS::Region
AccountId:
  Description: The AWS Account ID
  Value: !Ref AWS::AccountId
```

## Deploy service that uses Amazon VPC Lattice

You can use the following template to get started with creating an Amazon ECS service with VPC Lattice. You may need to complete the following additional steps to set up VPC Lattice:

- Update your security group's inbound rules for VPC Lattice to allow the inbound rule vpc-lattice prefix and to allow traffic on port 80.
- Associate VPC for the service to a VPC Lattice service network.
- Configure a private or public hosted zone with Amazon Route 53.
- Configure listeners and listener rules in a VPC Lattice service.
- Verify health check configurations of the target group.

For more information about using VPC Lattice with Amazon ECS, see [Use Amazon VPC Lattice to connect, observe, and secure your Amazon ECS services](#).

### JSON

```
{  
  "AWSTemplateFormatVersion": "2010-09-09",  
  "Description": "The template used to create an ECS Service with VPC Lattice.",  
  "Parameters": {  
    "ECSClusterName": {
```

```
"Type": "String",
"Default": "vpc-lattice-cluster"
},
"ECSServiceName": {
    "Type": "String",
    "Default": "vpc-lattice-service"
},
"SecurityGroupIDs": {
    "Type": "List<AWS::EC2::SecurityGroup::Id>",
    "Description": "Security Group IDs for the ECS Service"
},
"SubnetIDs": {
    "Type": "List<AWS::EC2::Subnet::Id>",
    "Description": "Subnet IDs for the ECS Service"
},
"VpcID": {
    "Type": "AWS::EC2::VPC::Id",
    "Description": "VPC ID for the resources"
},
"ContainerImage": {
    "Type": "String",
    "Default": "public.ecr.aws/docker/library/httpd:2.4",
    "Description": "Container image to use for the task"
},
"TaskCpu": {
    "Type": "Number",
    "Default": 256,
    "AllowedValues": [256, 512, 1024, 2048, 4096],
    "Description": "CPU units for the task"
},
"TaskMemory": {
    "Type": "Number",
    "Default": 512,
    "AllowedValues": [512, 1024, 2048, 4096, 8192, 16384],
    "Description": "Memory (in MiB) for the task"
},
"LogGroupName": {
    "Type": "String",
    "Default": "/ecs/vpc-lattice-task",
    "Description": "CloudWatch Log Group name"
},
"EnableContainerInsights": {
    "Type": "String",
    "Default": "enabled",
```

```
"AllowedValues": ["enabled", "disabled"],
"Description": "Enable or disable CloudWatch Container Insights for the cluster"
},
},
"Resources": {
"ECSCluster": {
"Type": "AWS::ECS::Cluster",
"Properties": {
"ClusterName": {"Ref": "ECSClusterName"},
"ClusterSettings": [
{
"Name": "containerInsights",
"Value": {"Ref": "EnableContainerInsights"}
}
],
"Tags": [
{
"Key": "Name",
"Value": {"Ref": "ECSClusterName"}
}
]
}
},
"ECSTaskExecutionRole": {
"Type": "AWS::IAM::Role",
"Properties": {
"AssumeRolePolicyDocument": {
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Principal": {
"Service": "ecs-tasks.amazonaws.com"
},
>Action": "sts:AssumeRole"
}
]
},
"ManagedPolicyArns": [
"arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
]
}
},
"TaskLogGroup": {
```

```
"Type": "AWS::Logs::LogGroup",
"DeletionPolicy": "Retain",
"UpdateReplacePolicy": "Retain",
"Properties": {
    "LogGroupName": {"Ref": "LogGroupName"}, 
    "RetentionInDays": 30
},
},
"VpcLatticeTaskDefinition": {
    "Type": "AWS::ECS::TaskDefinition",
    "Properties": {
        "ContainerDefinitions": [
            {
                "Command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title>
<style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
<div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
<h2>Congratulations!</h2> <p>Your application is now running on a container in
Amazon ECS.</p> </div></body></html>' > /usr/local/apache2/htdocs/index.html &&
httpd-foreground\""
                ],
                "EntryPoint": [
                    "sh",
                    "-c"
                ],
                "Essential": true,
                "Image": {"Ref": "ContainerImage"},
                "LogConfiguration": {
                    "LogDriver": "awslogs",
                    "Options": {
                        "mode": "non-blocking",
                        "max-buffer-size": "25m",
                        "awslogs-create-group": "true",
                        "awslogs-group": {"Ref": "LogGroupName"},
                        "awslogs-region": {"Ref": "AWS::Region"},
                        "awslogs-stream-prefix": "ecs"
                    }
                },
                "Name": "vpc-lattice-container",
                "PortMappings": [
                    {
                        "ContainerPort": 80,
                        "HostPort": 80,
                        "Protocol": "tcp",
                        "Subnets": [{"Ref": "SubnetList"}]
                    }
                ]
            }
        ]
    }
}
```

```
        "Name": "vpc-lattice-port"
    }
]
}
],
"Cpu": {"Ref": "TaskCpu"},
"ExecutionRoleArn": {"Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]},
"Family": "vpc-lattice-task-definition",
"Memory": {"Ref": "TaskMemory"},
"NetworkMode": "awsvpc",
"RequiresCompatibilities": [
    "FARGATE"
],
"RuntimePlatform": {
    "OperatingSystemFamily": "LINUX"
}
},
"ECSService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
        "Cluster": {"Ref": "ECSCluster"},
        "TaskDefinition": {"Ref": "VpcLatticeTaskDefinition"},
        "LaunchType": "FARGATE",
        "ServiceName": {"Ref": "ECSServiceName"},
        "SchedulingStrategy": "REPLICA",
        "DesiredCount": 2,
        "AvailabilityZoneRebalancing": "ENABLED",
        "NetworkConfiguration": {
            "AwsvpcConfiguration": {
                "AssignPublicIp": "ENABLED",
                "SecurityGroups": {
                    "Ref": "SecurityGroupIDs"
                },
                "Subnets": {
                    "Ref": "SubnetIDs"
                }
            }
        },
        "PlatformVersion": "LATEST",
        "VpcLatticeConfigurations": [
            {
                "RoleArn": "arn:aws:iam::111122223333:role/ecsInfrastructureRole",
                "PortName": "vpc-lattice-port",
            }
        ]
    }
}
```

```
"TargetGroupArn": {
    "Ref": "TargetGroup1"
}
],
],
"DeploymentConfiguration": {
    "DeploymentCircuitBreaker": {
        "Enable": true,
        "Rollback": true
    },
    "MaximumPercent": 200,
    "MinimumHealthyPercent": 100
},
"DeploymentController": {
    "Type": "ECS"
},
"ServiceConnectConfiguration": {
    "Enabled": false
},
"Tags": [],
"EnableECSManagedTags": true
},
},
"TargetGroup1": {
    "Type": "AWS::VpcLattice::TargetGroup",
    "Properties": {
        "Type": "IP",
        "Name": "first-target-group",
        "Config": {
            "Port": 80,
            "Protocol": "HTTP",
            "VpcIdentifier": {"Ref": "VpcID"},
            "HealthCheck": {
                "Enabled": true,
                "Path": "/"
            }
        },
        "Tags": [
            {
                "Key": "ecs-application-networking/ServiceName",
                "Value": {"Ref": "ECSServiceName"}
            },
            {
                "Key": "ecs-application-networking/ClusterName",
                "Value": {"Ref": "ECSClusterName"}
            }
        ]
    }
}
```

```
        "Value": {"Ref": "ECSClusterName"}  
    },  
    {  
        "Key": "ecs-application-networking/TaskDefinition",  
        "Value": {"Ref": "VpcLatticeTaskDefinition"}  
    },  
    {  
        "Key": "ecs-application-networking/VpcId",  
        "Value": {"Ref": "VpcID"}  
    }  
]  
}  
},  
"  
Outputs": {  
    "ClusterName": {  
        "Description": "The cluster used to create the service.",  
        "Value": {  
            "Ref": "ECSCluster"  
        }  
    },  
    "ClusterArn": {  
        "Description": "The ARN of the ECS cluster",  
        "Value": {  
            "Fn::GetAtt": ["ECSCluster", "Arn"]  
        }  
    },  
    "ECSService": {  
        "Description": "The created service.",  
        "Value": {  
            "Ref": "ECSService"  
        }  
    },  
    "TaskDefinitionArn": {  
        "Description": "The ARN of the task definition",  
        "Value": {  
            "Ref": "VpcLatticeTaskDefinition"  
        }  
    }  
}
```

## YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: The template used to create an ECS Service with VPC Lattice.

Parameters:
  ECSClusterName:
    Type: String
    Default: vpc-lattice-cluster
  ECSServiceName:
    Type: String
    Default: vpc-lattice-service
  SecurityGroupIDs:
    Type: List<AWS::EC2::SecurityGroup::Id>
    Description: Security Group IDs for the ECS Service
  SubnetIDs:
    Type: List<AWS::EC2::Subnet::Id>
    Description: Subnet IDs for the ECS Service
  VpcID:
    Type: AWS::EC2::VPC::Id
    Description: VPC ID for the resources
  ContainerImage:
    Type: String
    Default: public.ecr.aws/docker/library/httpd:2.4
    Description: Container image to use for the task
  TaskCpu:
    Type: Number
    Default: 256
    AllowedValues: [256, 512, 1024, 2048, 4096]
    Description: CPU units for the task
  TaskMemory:
    Type: Number
    Default: 512
    AllowedValues: [512, 1024, 2048, 4096, 8192, 16384]
    Description: Memory (in MiB) for the task
  LogGroupName:
    Type: String
    Default: /ecs/vpc-lattice-task
    Description: CloudWatch Log Group name
  EnableContainerInsights:
    Type: String
    Default: 'enhanced'
    AllowedValues: ['enabled', 'disabled', 'enhanced']
    Description: Enable or disable CloudWatch Container Insights for the cluster
```

```
Resources:
# ECS Cluster
ECSCluster:
  Type: AWS::ECS::Cluster
  Properties:
    ClusterName: !Ref ECSClusterName
    ClusterSettings:
      - Name: containerInsights
        Value: !Ref EnableContainerInsights
    Tags:
      - Key: Name
        Value: !Ref ECSClusterName

# IAM Roles
ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy

# CloudWatch Logs
TaskLogGroup:
  Type: AWS::Logs::LogGroup
  DeletionPolicy: Retain
  UpdateReplacePolicy: Retain
  Properties:
    LogGroupName: !Ref LogGroupName
    RetentionInDays: 30

# Task Definition
VpcLatticeTaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    ContainerDefinitions:
      - Command:
          - >-
```

```
        /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon ECS.</p> </div></body></html>' > /usr/local/apache2/htdocs/index.html && httpd-foreground"
EntryPoint:
  - sh
  - '-c'
Essential: true
Image: !Ref ContainerImage
LogConfiguration:
  LogDriver: awslogs
  Options:
    mode: non-blocking
    max-buffer-size: 25m
    awslogs-create-group: 'true'
    awslogs-group: !Ref LogGroupName
    awslogs-region: !Ref 'AWS::Region'
    awslogs-stream-prefix: ecs
Name: vpc-lattice-container
PortMappings:
  - ContainerPort: 80
    HostPort: 80
    Protocol: tcp
    Name: vpc-lattice-port
Cpu: !Ref TaskCpu
ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
Family: vpc-lattice-task-definition
Memory: !Ref TaskMemory
NetworkMode: awsvpc
RequiresCompatibilities:
  - FARGATE
RuntimePlatform:
  OperatingSystemFamily: LINUX

ECSService:
  Type: AWS::ECS::Service
  Properties:
    Cluster: !Ref ECSCluster
    TaskDefinition: !Ref VpcLatticeTaskDefinition
    LaunchType: FARGATE
    ServiceName: !Ref ECSServiceName
```

```
SchedulingStrategy: REPLICA
DesiredCount: 2
AvailabilityZoneRebalancing: ENABLED
NetworkConfiguration:
  AwsVpcConfiguration:
    AssignPublicIp: ENABLED
    SecurityGroups: !Ref SecurityGroupIDs
    Subnets: !Ref SubnetIDs
PlatformVersion: LATEST
VpcLatticeConfigurations:
  - RoleArn: arn:aws:iam::111122223333:role/ecsInfrastructureRole
    PortName: vpc-lattice-port
    TargetGroupArn: !Ref TargetGroup1
DeploymentConfiguration:
  DeploymentCircuitBreaker:
    Enable: true
    Rollback: true
    MaximumPercent: 200
    MinimumHealthyPercent: 100
  DeploymentController:
    Type: ECS
  ServiceConnectConfiguration:
    Enabled: false
  Tags: []
  EnableECSManagedTags: true

TargetGroup1:
  Type: AWS::VpcLattice::TargetGroup
  Properties:
    Type: IP
    Name: first-target-group
  Config:
    Port: 80
    Protocol: HTTP
    VpcIdentifier: !Ref VpcID
  HealthCheck:
    Enabled: true
    Path: /
  Tags:
    - Key: ecs-application-networking/ServiceName
      Value: !Ref ECSServiceName
    - Key: ecs-application-networking/ClusterName
      Value: !Ref ECSClusterName
    - Key: ecs-application-networking/TaskDefinition
```

```
    Value: !Ref VpcLatticeTaskDefinition
    - Key: ecs-application-networking/VpcId
      Value: !Ref VpcID
```

#### Outputs:

##### ClusterName:

```
  Description: The cluster used to create the service.
  Value: !Ref ECSCluster
```

##### ClusterArn:

```
  Description: The ARN of the ECS cluster
  Value: !GetAtt ECSCluster.Arn
```

##### ECSService:

```
  Description: The created service.
  Value: !Ref ECSService
```

##### TaskDefinitionArn:

```
  Description: The ARN of the task definition
  Value: !Ref VpcLatticeTaskDefinition
```

## Deploy service with a volume configuration

The following template includes a volume configuration in the service definition. Amazon ECS supports configuring the following data volumes by using a volume configuration at launch: Amazon EBS volumes. For more information about Amazon EBS volumes, see [Use Amazon EBS volumes with Amazon ECS](#).

### JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "The template used to create an ECS Service that includes a volume configuration. The configuration is used to create Amazon EBS volumes for attachment to the tasks. One volume is attached per task.",
  "Parameters": {
    "ECSClusterName": {
      "Type": "String",
      "Default": "volume-config-cluster",
      "Description": "Name of the ECS cluster"
    },
    "SecurityGroupIDs": {
      "Type": "List<AWS::EC2::SecurityGroup::Id>",
      "Description": "Security Group IDs for the ECS Service"
    }
  },
  "Resources": {
    "VpcLatticeTaskDefinition": {
      "Type": "AWS::ECS::TaskDefinition",
      "Properties": {
        "ContainerDefinitions": [
          {
            "Name": "app",
            "Image": "my-image:latest",
            "MountPoints": [
              {
                "ContainerPath": "/data",
                "SourceVolume": "data"
              }
            ]
          }
        ],
        "Memory": 512,
        "NetworkMode": "awsvpc",
        "RequiresCompatibilities": [
          "FARGATE"
        ],
        "TaskRoleArn": "arn:aws:iam::123456789012:role/task-role"
      }
    }
  }
}
```