AWSTemplateFormatVersion: 2010-09-09

Description: Creating ECS service

Parameters:

AppName:

Type: String

Description: Name of app requiring ELB exposure

Default: simple-app

AppContainerPort:

Type: Number

Description: Container port of app requiring ELB exposure

Default: '80'

AppHostPort:

Type: Number

Description: Host port of app requiring ELB exposure

Default: '80'

ServiceName:

Type: String

LoadBalancerName:

Type: String

HealthCheckGracePeriodSeconds:

Type: String

Resources:

cluster:

Type: AWS::ECS::Cluster

taskdefinition:

Type: AWS::ECS::TaskDefinition

Properties:

ContainerDefinitions:

- Name: !Ref AppName

Image: 445261605856.dkr.ecr.us-west-1.amazonaws.com/hello-repository:latest

Cpu: '10'

PortMappings:

- ContainerPort: !Ref AppContainerPort

HostPort: !Ref AppHostPort

Memory: '500'

Essential: 'true'

service:

Type: AWS::ECS::Service

Properties:

Cluster: !Ref cluster

DeploymentConfiguration:

MaximumPercent: 200

MinimumHealthyPercent: 100

DesiredCount: 0

HealthCheckGracePeriodSeconds: !Ref HealthCheckGracePeriodSeconds

LoadBalancers:

- ContainerName: !Ref AppName

ContainerPort: !Ref AppContainerPort

LoadBalancerName: !Ref elb

PlacementStrategies:

- Type: binpack

Field: memory

- Type: spread

Field: host

PlacementConstraints:

- Type: memberOf

Expression: 'attribute:ecs.availability-zone != us-east-1d'

- Type: distinctInstance

TaskDefinition: !Ref taskdefinition

ServiceName: !Ref ServiceName

Role: !Ref Role

elb:

Type: AWS::ElasticLoadBalancing::LoadBalancer

Properties:

LoadBalancerName: !Ref LoadBalancerName

Listeners:

- InstancePort: !Ref AppHostPort

LoadBalancerPort: '80'

Protocol: HTTP

Subnets:

- !Ref Subnet1

DependsOn: GatewayAttachment

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: 10.0.0.0/24

Subnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

CidrBlock: 10.0.0.0/25

InternetGateway:

Type: AWS::EC2::InternetGateway

GatewayAttachment:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

InternetGatewayId: !Ref InternetGateway

VpcId: !Ref VPC

Role:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: 2008-10-17

Statement:

- Sid: ''

Effect: Allow

Principal:

Service: ecs.amazonaws.com

Action: 'sts:AssumeRole'

ManagedPolicyArns:

- 'arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceRole'

Outputs:

Cluster:

Value: !Ref cluster

AWSTemplateFormatVersion: '2010-09-09'

Parameters:

KeyName:

Type: AWS::EC2::KeyPair::KeyName

Description: Name of an existing EC2 KeyPair to enable SSH access to the ECS instances.

VpcId:

Type: AWS::EC2::VPC::Id

Description: Select a VPC that allows instances access to the Internet.

SubnetId:

Type: List<AWS::EC2::Subnet::Id>

Description: Select at two subnets in your selected VPC.

DesiredCapacity:

Type: Number

Default: '1'

Description: Number of instances to launch in your ECS cluster.

MaxSize:

Type: Number

Default: '1'

Description: Maximum number of instances that can be launched in your ECS cluster.

InstanceType:

Description: EC2 instance type

Type: String

Default: t2.micro

AllowedValues: [t2.micro, t2.small, t2.medium, t2.large, m3.medium, m3.large,

m3.xlarge, m3.2xlarge, m4.large, m4.xlarge, m4.2xlarge, m4.4xlarge, m4.10xlarge,

c4.large, c4.xlarge, c4.2xlarge, c4.4xlarge, c4.8xlarge, c3.large, c3.xlarge,

c3.2xlarge, c3.4xlarge, c3.8xlarge, r3.large, r3.xlarge, r3.2xlarge, r3.4xlarge,

r3.8xlarge, i2.xlarge, i2.2xlarge, i2.4xlarge, i2.8xlarge]

ConstraintDescription: Please choose a valid instance type.

Mappings:

AWSRegionToAMI:

us-east-1:

AMIID: ami-eca289fb

us-east-2:

AMIID: ami-446f3521

us-west-1:

AMIID: ami-9fadf8ff

us-west-2:

AMIID: ami-7abc111a

eu-west-1:

AMIID: ami-a1491ad2

eu-central-1:

AMIID: ami-54f5303b

ap-northeast-1:

AMIID: ami-9cd57ffd

ap-southeast-1:

AMIID: ami-a900a3ca

ap-southeast-2:

AMIID: ami-5781be34

Resources:

ECSCluster:

Type: AWS::ECS::Cluster

EcsSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: ECS Security Group

VpcId: !Ref 'VpcId'

EcsSecurityGroupHTTPinbound:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !Ref 'EcsSecurityGroup'

IpProtocol: tcp

FromPort: '80'

ToPort: '80'

CidrIp: 0.0.0.0/0

EcsSecurityGroupSSHinbound:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !Ref 'EcsSecurityGroup'

IpProtocol: tcp

FromPort: '22'

ToPort: '22'

CidrIp: 0.0.0.0/0

EcsSecurityGroupALBports:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !Ref 'EcsSecurityGroup'

IpProtocol: tcp

FromPort: '31000'

ToPort: '61000'

SourceSecurityGroupId: !Ref 'EcsSecurityGroup'

CloudwatchLogsGroup:

Type: AWS::Logs::LogGroup

Properties:

LogGroupName: !Join ['-', [ECSLogGroup, !Ref 'AWS::StackName']]

RetentionInDays: 14

taskdefinition:

Type: AWS::ECS::TaskDefinition

Properties:

Family: !Join ['', [!Ref 'AWS::StackName', -ecs-demo-app]]

ContainerDefinitions:

- Name: simple-app

Cpu: '10'

Essential: 'true'

Image: httpd:2.4

Memory: '300'

LogConfiguration:

LogDriver: awslogs

Options:

awslogs-group: !Ref 'CloudwatchLogsGroup'

awslogs-region: !Ref 'AWS::Region'

awslogs-stream-prefix: ecs-demo-app

MountPoints:

- ContainerPath: /usr/local/apache2/htdocs

SourceVolume: my-vol

PortMappings:

- ContainerPort: 80

- Name: busybox

Cpu: 10

Command: ['/bin/sh -c "while true; do 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>'' > top; /bin/date > date ;

echo ''</div></body></html>'' > bottom; cat top date bottom > /usr/local/apache2/htdocs/index.html

; sleep 1; done"']

EntryPoint: [sh, -c]

Essential: false

Image: busybox

Memory: 200

LogConfiguration:

LogDriver: awslogs

Options:

awslogs-group: !Ref 'CloudwatchLogsGroup'

awslogs-region: !Ref 'AWS::Region'

awslogs-stream-prefix: ecs-demo-app

VolumesFrom:

- SourceContainer: simple-app

Volumes:

- Name: my-vol

ECSALB:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: ECSALB

Scheme: internet-facing

LoadBalancerAttributes:

- Key: idle\_timeout.timeout\_seconds

Value: '30'

Subnets: !Ref 'SubnetId'

SecurityGroups: [!Ref 'EcsSecurityGroup']

ALBListener:

Type: AWS::ElasticLoadBalancingV2::Listener

DependsOn: ECSServiceRole

Properties:

DefaultActions:

- Type: forward

TargetGroupArn: !Ref 'ECSTG'

LoadBalancerArn: !Ref 'ECSALB'

Port: '80'

Protocol: HTTP

ECSALBListenerRule:

Type: AWS::ElasticLoadBalancingV2::ListenerRule

DependsOn: ALBListener

Properties:

Actions:

- Type: forward

TargetGroupArn: !Ref 'ECSTG'

Conditions:

- Field: path-pattern

Values: [/]

ListenerArn: !Ref 'ALBListener'

Priority: 1

ECSTG:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

DependsOn: ECSALB

Properties:

HealthCheckIntervalSeconds: 10

HealthCheckPath: /

HealthCheckProtocol: HTTP

HealthCheckTimeoutSeconds: 5

HealthyThresholdCount: 2

Name: ECSTG

Port: 80

Protocol: HTTP

UnhealthyThresholdCount: 2

VpcId: !Ref 'VpcId'

ECSAutoScalingGroup:

Type: AWS::AutoScaling::AutoScalingGroup

Properties:

VPCZoneIdentifier: !Ref 'SubnetId'

LaunchConfigurationName: !Ref 'ContainerInstances'

MinSize: '1'

MaxSize: !Ref 'MaxSize'

DesiredCapacity: !Ref 'DesiredCapacity'

CreationPolicy:

ResourceSignal:

Timeout: PT15M

UpdatePolicy:

AutoScalingReplacingUpdate:

WillReplace: 'true'

ContainerInstances:

Type: AWS::AutoScaling::LaunchConfiguration

Properties:

ImageId: !FindInMap [AWSRegionToAMI, !Ref 'AWS::Region', AMIID]

SecurityGroups: [!Ref 'EcsSecurityGroup']

InstanceType: !Ref 'InstanceType'

IamInstanceProfile: !Ref 'EC2InstanceProfile'

KeyName: !Ref 'KeyName'

UserData:

Fn::Base64: !Sub |

#!/bin/bash -xe

echo ECS\_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config

yum install -y aws-cfn-bootstrap

/opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource ECSAutoScalingGroup --region ${AWS::Region}

service:

Type: AWS::ECS::Service

DependsOn: ALBListener

Properties:

Cluster: !Ref 'ECSCluster'

DesiredCount: '1'

LoadBalancers:

- ContainerName: simple-app

ContainerPort: '80'

TargetGroupArn: !Ref 'ECSTG'

Role: !Ref 'ECSServiceRole'

TaskDefinition: !Ref 'taskdefinition'

ECSServiceRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Statement:

- Effect: Allow

Principal:

Service: [ecs.amazonaws.com]

Action: ['sts:AssumeRole']

Path: /

Policies:

- PolicyName: ecs-service

PolicyDocument:

Statement:

- Effect: Allow

Action: ['elasticloadbalancing:DeregisterInstancesFromLoadBalancer', 'elasticloadbalancing:DeregisterTargets',

'elasticloadbalancing:Describe\*', 'elasticloadbalancing:RegisterInstancesWithLoadBalancer',

'elasticloadbalancing:RegisterTargets', 'ec2:Describe\*', 'ec2:AuthorizeSecurityGroupIngress']

Resource: '\*'

ServiceScalingTarget:

Type: AWS::ApplicationAutoScaling::ScalableTarget

DependsOn: service

Properties:

MaxCapacity: 2

MinCapacity: 1

ResourceId: !Join ['', [service/, !Ref 'ECSCluster', /, !GetAtt [service, Name]]]

RoleARN: !GetAtt [AutoscalingRole, Arn]

ScalableDimension: ecs:service:DesiredCount

ServiceNamespace: ecs

ServiceScalingPolicy:

Type: AWS::ApplicationAutoScaling::ScalingPolicy

Properties:

PolicyName: AStepPolicy

PolicyType: StepScaling

ScalingTargetId: !Ref 'ServiceScalingTarget'

StepScalingPolicyConfiguration:

AdjustmentType: PercentChangeInCapacity

Cooldown: 60

MetricAggregationType: Average

StepAdjustments:

- MetricIntervalLowerBound: 0

ScalingAdjustment: 200

ALB500sAlarmScaleUp:

Type: AWS::CloudWatch::Alarm

Properties:

EvaluationPeriods: '1'

Statistic: Average

Threshold: '10'

AlarmDescription: Alarm if our ALB generates too many HTTP 500s.

Period: '60'

AlarmActions: [!Ref 'ServiceScalingPolicy']

Namespace: AWS/ApplicationELB

Dimensions:

- Name: LoadBalancer

Value: !GetAtt

- ECSALB

- LoadBalancerFullName

ComparisonOperator: GreaterThanThreshold

MetricName: HTTPCode\_ELB\_5XX\_Count

EC2Role:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Statement:

- Effect: Allow

Principal:

Service: [ec2.amazonaws.com]

Action: ['sts:AssumeRole']

Path: /

Policies:

- PolicyName: ecs-service

PolicyDocument:

Statement:

- Effect: Allow

Action: ['ecs:CreateCluster', 'ecs:DeregisterContainerInstance', 'ecs:DiscoverPollEndpoint',

'ecs:Poll', 'ecs:RegisterContainerInstance', 'ecs:StartTelemetrySession',

'ecs:Submit\*', 'logs:CreateLogStream', 'logs:PutLogEvents']

Resource: '\*'

AutoscalingRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Statement:

- Effect: Allow

Principal:

Service: [application-autoscaling.amazonaws.com]

Action: ['sts:AssumeRole']

Path: /

Policies:

- PolicyName: service-autoscaling

PolicyDocument:

Statement:

- Effect: Allow

Action: ['application-autoscaling:\*', 'cloudwatch:DescribeAlarms', 'cloudwatch:PutMetricAlarm',

'ecs:DescribeServices', 'ecs:UpdateService']

Resource: '\*'

EC2InstanceProfile:

Type: AWS::IAM::InstanceProfile

Properties:

Path: /

Roles: [!Ref 'EC2Role']

Outputs:

ecsservice:

Value: !Ref 'service'

ecscluster:

Value: !Ref 'ECSCluster'

ECSALB:

Description: Your ALB DNS URL

Value: !Join ['', [!GetAtt [ECSALB, DNSName]]]

taskdef:

Value: !Ref 'taskdefinition'

For tomcat ecs cluster

AWSTemplateFormatVersion: '2010-09-09'

Parameters:

KeyName:

Type: AWS::EC2::KeyPair::KeyName

Description: Name of an existing EC2 KeyPair to enable SSH access to the ECS instances.

VpcId:

Type: AWS::EC2::VPC::Id

Description: Select a VPC that allows instances access to the Internet.

SubnetId:

Type: List<AWS::EC2::Subnet::Id>

Description: Select at two subnets in your selected VPC.

DesiredCapacity:

Type: Number

Default: '1'

Description: Number of instances to launch in your ECS cluster.

MaxSize:

Type: Number

Default: '1'

Description: Maximum number of instances that can be launched in your ECS cluster.

InstanceType:

Description: EC2 instance type

Type: String

Default: t2.micro

AllowedValues: [t2.micro, t2.small, t2.medium, t2.large, m3.medium, m3.large,

m3.xlarge, m3.2xlarge, m4.large, m4.xlarge, m4.2xlarge, m4.4xlarge, m4.10xlarge,

c4.large, c4.xlarge, c4.2xlarge, c4.4xlarge, c4.8xlarge, c3.large, c3.xlarge,

c3.2xlarge, c3.4xlarge, c3.8xlarge, r3.large, r3.xlarge, r3.2xlarge, r3.4xlarge,

r3.8xlarge, i2.xlarge, i2.2xlarge, i2.4xlarge, i2.8xlarge]

ConstraintDescription: Please choose a valid instance type.

Mappings:

AWSRegionToAMI:

us-east-1:

AMIID: ami-eca289fb

us-east-2:

AMIID: ami-446f3521

us-west-1:

AMIID: ami-9fadf8ff

us-west-2:

AMIID: ami-7abc111a

eu-west-1:

AMIID: ami-05cc68a00d392447a

eu-central-1:

AMIID: ami-54f5303b

ap-northeast-1:

AMIID: ami-9cd57ffd

ap-southeast-1:

AMIID: ami-a900a3ca

ap-southeast-2:

AMIID: ami-5781be34

Resources:

ECSCluster:

Type: AWS::ECS::Cluster

EcsSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: ECS Security Group

VpcId: !Ref 'VpcId'

EcsSecurityGroupHTTPinbound:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !Ref 'EcsSecurityGroup'

IpProtocol: tcp

FromPort: '80'

ToPort: '80'

CidrIp: 0.0.0.0/0

EcsSecurityGroupSSHinbound:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !Ref 'EcsSecurityGroup'

IpProtocol: tcp

FromPort: '22'

ToPort: '22'

CidrIp: 0.0.0.0/0

EcsSecurityGroupALBports:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !Ref 'EcsSecurityGroup'

IpProtocol: tcp

FromPort: '31000'

ToPort: '61000'

SourceSecurityGroupId: !Ref 'EcsSecurityGroup'

CloudwatchLogsGroup:

Type: AWS::Logs::LogGroup

Properties:

LogGroupName: !Join ['-', [ECSLogGroup, !Ref 'AWS::StackName']]

RetentionInDays: 14

taskdefinition:

Type: AWS::ECS::TaskDefinition

Properties:

Family: !Join ['', [!Ref 'AWS::StackName', -ecs-demo-app]]

ContainerDefinitions:

- Name: simple-app

Cpu: '10'

Essential: 'true'

Image: tomcat:7.0

Memory: '300'

LogConfiguration:

LogDriver: awslogs

Options:

awslogs-group: !Ref 'CloudwatchLogsGroup'

awslogs-region: !Ref 'AWS::Region'

awslogs-stream-prefix: ecs-demo-app

PortMappings:

- ContainerPort: 8080

ECSALB:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: ECSALB

Scheme: internet-facing

LoadBalancerAttributes:

- Key: idle\_timeout.timeout\_seconds

Value: '30'

Subnets: !Ref 'SubnetId'

SecurityGroups: [!Ref 'EcsSecurityGroup']

ALBListener:

Type: AWS::ElasticLoadBalancingV2::Listener

DependsOn: ECSServiceRole

Properties:

DefaultActions:

- Type: forward

TargetGroupArn: !Ref 'ECSTG'

LoadBalancerArn: !Ref 'ECSALB'

Port: '80'

Protocol: HTTP

ECSALBListenerRule:

Type: AWS::ElasticLoadBalancingV2::ListenerRule

DependsOn: ALBListener

Properties:

Actions:

- Type: forward

TargetGroupArn: !Ref 'ECSTG'

Conditions:

- Field: path-pattern

Values: [/]

ListenerArn: !Ref 'ALBListener'

Priority: 1

ECSTG:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

DependsOn: ECSALB

Properties:

HealthCheckIntervalSeconds: 10

HealthCheckPath: /

HealthCheckProtocol: HTTP

HealthCheckTimeoutSeconds: 5

HealthyThresholdCount: 2

Name: ECSTG

Port: 80

Protocol: HTTP

UnhealthyThresholdCount: 2

VpcId: !Ref 'VpcId'

ECSAutoScalingGroup:

Type: AWS::AutoScaling::AutoScalingGroup

Properties:

VPCZoneIdentifier: !Ref 'SubnetId'

LaunchConfigurationName: !Ref 'ContainerInstances'

MinSize: '1'

MaxSize: !Ref 'MaxSize'

DesiredCapacity: !Ref 'DesiredCapacity'

CreationPolicy:

ResourceSignal:

Timeout: PT15M

UpdatePolicy:

AutoScalingReplacingUpdate:

WillReplace: 'true'

ContainerInstances:

Type: AWS::AutoScaling::LaunchConfiguration

Properties:

ImageId: !FindInMap [AWSRegionToAMI, !Ref 'AWS::Region', AMIID]

SecurityGroups: [!Ref 'EcsSecurityGroup']

InstanceType: !Ref 'InstanceType'

IamInstanceProfile: !Ref 'EC2InstanceProfile'

KeyName: !Ref 'KeyName'

UserData:

Fn::Base64: !Sub |

#!/bin/bash -xe

echo ECS\_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config

yum install -y aws-cfn-bootstrap

/opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource ECSAutoScalingGroup --region ${AWS::Region}

service:

Type: AWS::ECS::Service

DependsOn: ALBListener

Properties:

Cluster: !Ref 'ECSCluster'

DesiredCount: '1'

LoadBalancers:

- ContainerName: simple-app

ContainerPort: '8080'

TargetGroupArn: !Ref 'ECSTG'

Role: !Ref 'ECSServiceRole'

TaskDefinition: !Ref 'taskdefinition'

ECSServiceRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Statement:

- Effect: Allow

Principal:

Service: [ecs.amazonaws.com]

Action: ['sts:AssumeRole']

Path: /

Policies:

- PolicyName: ecs-service

PolicyDocument:

Statement:

- Effect: Allow

Action: ['elasticloadbalancing:DeregisterInstancesFromLoadBalancer', 'elasticloadbalancing:DeregisterTargets',

'elasticloadbalancing:Describe\*', 'elasticloadbalancing:RegisterInstancesWithLoadBalancer',

'elasticloadbalancing:RegisterTargets', 'ec2:Describe\*', 'ec2:AuthorizeSecurityGroupIngress']

Resource: '\*'

ServiceScalingTarget:

Type: AWS::ApplicationAutoScaling::ScalableTarget

DependsOn: service

Properties:

MaxCapacity: 2

MinCapacity: 1

ResourceId: !Join ['', [service/, !Ref 'ECSCluster', /, !GetAtt [service, Name]]]

RoleARN: !GetAtt [AutoscalingRole, Arn]

ScalableDimension: ecs:service:DesiredCount

ServiceNamespace: ecs

ServiceScalingPolicy:

Type: AWS::ApplicationAutoScaling::ScalingPolicy

Properties:

PolicyName: AStepPolicy

PolicyType: StepScaling

ScalingTargetId: !Ref 'ServiceScalingTarget'

StepScalingPolicyConfiguration:

AdjustmentType: PercentChangeInCapacity

Cooldown: 60

MetricAggregationType: Average

StepAdjustments:

- MetricIntervalLowerBound: 0

ScalingAdjustment: 200

ALB500sAlarmScaleUp:

Type: AWS::CloudWatch::Alarm

Properties:

EvaluationPeriods: '1'

Statistic: Average

Threshold: '10'

AlarmDescription: Alarm if our ALB generates too many HTTP 500s.

Period: '60'

AlarmActions: [!Ref 'ServiceScalingPolicy']

Namespace: AWS/ApplicationELB

Dimensions:

- Name: LoadBalancer

Value: !GetAtt

- ECSALB

- LoadBalancerFullName

ComparisonOperator: GreaterThanThreshold

MetricName: HTTPCode\_ELB\_5XX\_Count

EC2Role:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Statement:

- Effect: Allow

Principal:

Service: [ec2.amazonaws.com]

Action: ['sts:AssumeRole']

Path: /

Policies:

- PolicyName: ecs-service

PolicyDocument:

Statement:

- Effect: Allow

Action: ['ecs:CreateCluster', 'ecs:DeregisterContainerInstance', 'ecs:DiscoverPollEndpoint',

'ecs:Poll', 'ecs:RegisterContainerInstance', 'ecs:StartTelemetrySession',

'ecs:Submit\*', 'logs:CreateLogStream', 'logs:PutLogEvents']

Resource: '\*'

AutoscalingRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Statement:

- Effect: Allow

Principal:

Service: [application-autoscaling.amazonaws.com]

Action: ['sts:AssumeRole']

Path: /

Policies:

- PolicyName: service-autoscaling

PolicyDocument:

Statement:

- Effect: Allow

Action: ['application-autoscaling:\*', 'cloudwatch:DescribeAlarms', 'cloudwatch:PutMetricAlarm',

'ecs:DescribeServices', 'ecs:UpdateService']

Resource: '\*'

EC2InstanceProfile:

Type: AWS::IAM::InstanceProfile

Properties:

Path: /

Roles: [!Ref 'EC2Role']

Outputs:

ecsservice:

Value: !Ref 'service'

ecscluster:

Value: !Ref 'ECSCluster'

ECSALB:

Description: Your ALB DNS URL

Value: !Join ['', [!GetAtt [ECSALB, DNSName]]]

taskdef:

Value: !Ref 'taskdefinition'