Continuous Integration & Deployment with Github, CircleCI and AWS CodeDeploy

Continuous Deployment is a software development practice in which every code change goes through the entire pipeline and is put into production, automatically, resulting in many production deployments every day.

Continuous Deployment Workflow

AMI

1. Developer commits code changes to GitHub repository.
2. CircleCI will trigger a new build on commit notification.
3. CircleCI will run the build steps from CircleCI config file `.circleci/config.yml` from your repository. Build steps should do the following:
    a. Install `awscli` in your primary container by following the AWS CLI documentation [http://docs.aws.amazon.com/cli/latest/userguide/installing.html].
    b. Download HasiCorp Packer binary and make sure it is executable.
    c. Validate packer template.
    d. Build AMI and register it with AWS.

Web Application

1. Developer commits code changes to GitHub repository.
2. CircleCI will trigger a new build on commit notification.

3. CircleCI will run the build steps from CircleCI config file `.circleci/config.yml` from your repository. Build steps should do the following:
    a. Install `awscli` in your primary container by following the AWS CLI documentation [http://docs.aws.amazon.com/cli/latest/userguide/installing.html].
    b. Run unit tests.
    c. Build your artifacts if all tests are successful.
    d. Zip your artifacts and upload it to AWS S3 bucket dedicated for code deploy.
    e. Call AWS CodeDeploy to deploy the latest revision of your application to the EC2 instances.

IAM Setup

CodeDeploy-EC2-S3 Policy for the Server (EC2)

`CodeDeploy-EC2-S3` policy allows EC2 instances to get data from S3 buckets. This is required for EC2 instances to download latest application revision.

> **Note**
>
> Replace `*` with appropriate ARN name to create secure policies.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
```

```
    ]
}
```

CircleCI-Upload-To-S3 Policy for CircleCI to Upload to AWS S3

`CircleCI-Upload-To-S3` policy allows CircleCI to upload artifiacts from latest successful build to dedicated S3 bucket used by code deploy.

> **Note**
>
> Replace `*` with appropriate ARN name to create secure policies.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

CirlceCI-Code-Deploy Policy for CirlceCI to Call CodeDeploy

`CirlceCI-Code-Deploy` policy allows CirlceCI to call CodeDeploy APIs to initiate application deployment on EC2 instances.

> **Note**
>
> 1. Replace `AWS_REGION` with AWS region.
> 2. Replace `AWS_ACCOUNT_ID` with your account id.

3. Replace `CODE_DEPLOY_APPLICATION_NAME` with your Code Deploy Application Name (Note: We haven't created Code Deploy Application yet. So note down the application name for later steps.)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:RegisterApplicationRevision",
        "codedeploy:GetApplicationRevision"
      ],
      "Resource": [

"arn:aws:codedeploy:AWS_REGION:AWS_ACCOUNT_ID:application:CODE_DEPL

      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetDeployment"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource": [

"arn:aws:codedeploy:AWS_REGION:AWS_ACCOUNT_ID:deploymentconfig:Code


"arn:aws:codedeploy:AWS_REGION:AWS_ACCOUNT_ID:deploymentconfig:Code


"arn:aws:codedeploy:AWS_REGION:AWS_ACCOUNT_ID:deploymentconfig:Code
```

```
        ]
      }
    ]
}
```

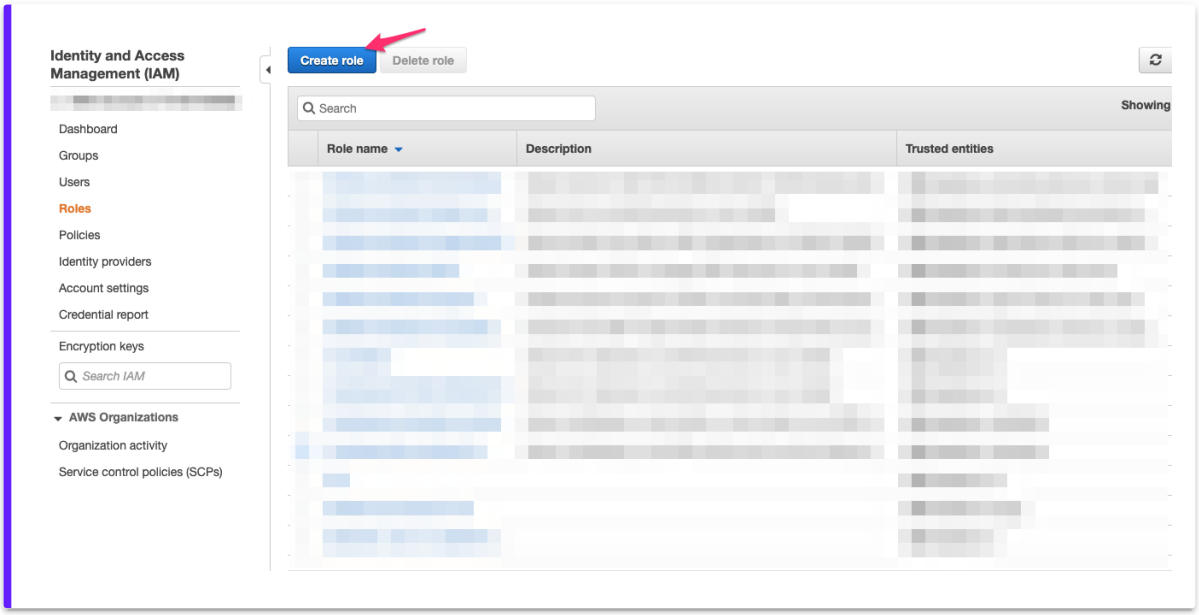
Create New User for CircleCI

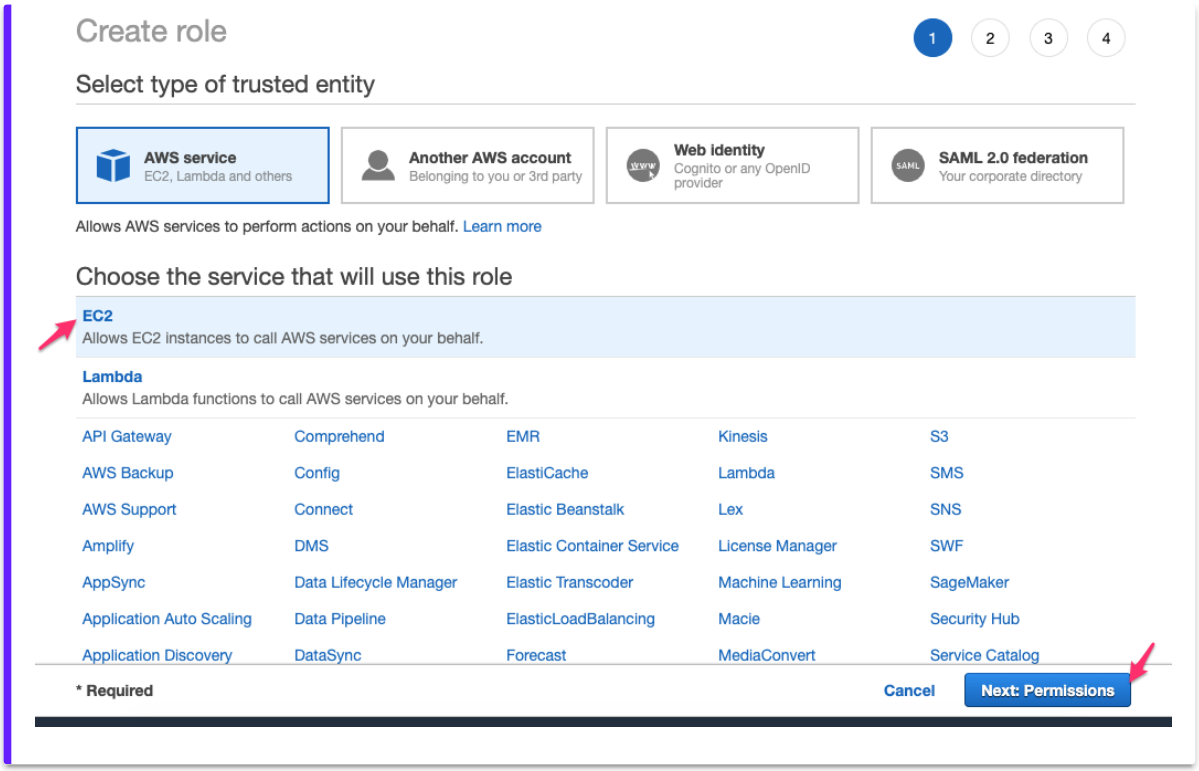
Create a new user `circleci` with programmatic access only. Attach following IAM policies to this newly created user:

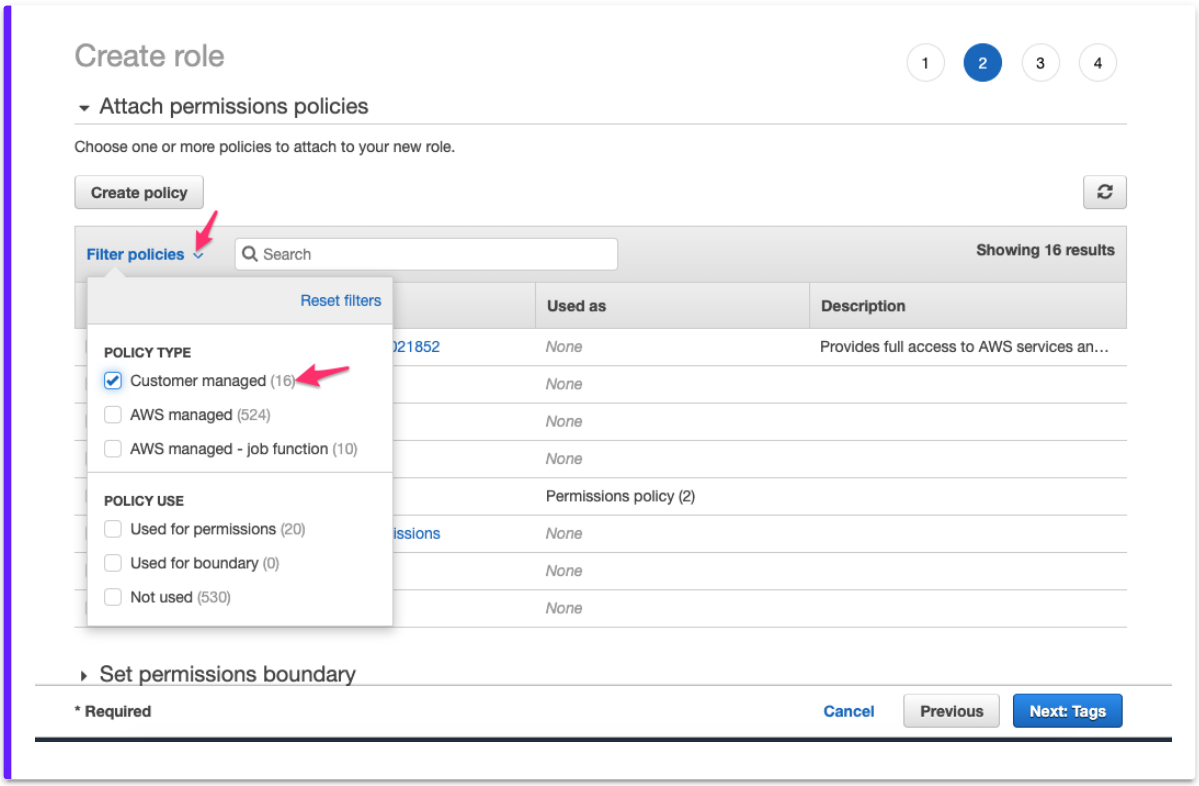
1. CirlceCI-Upload-To-S3
2. CirlceCI-Code-Deploy
3. circleci-ec2-ami [https://www.packer.io/docs/builders/amazon.html#iam-task-or-instance-role]

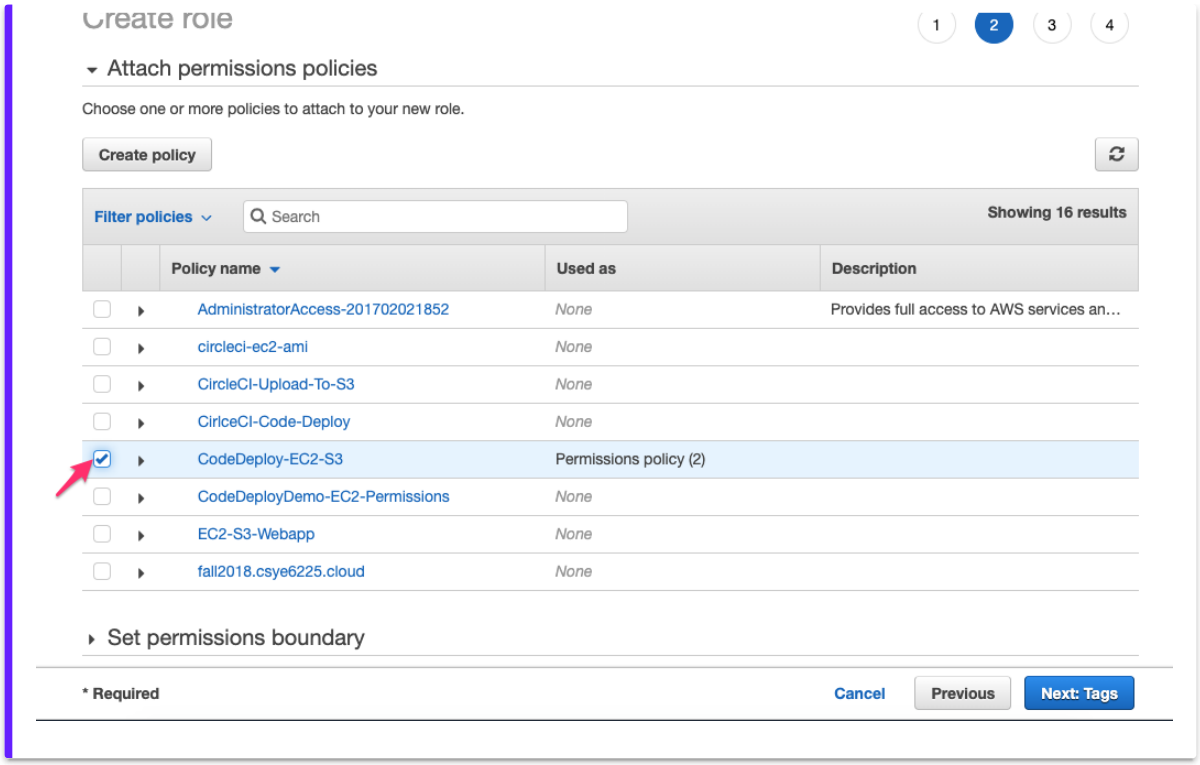
CodeDeployEC2ServiceRole IAM Role for EC2 Instance(s)

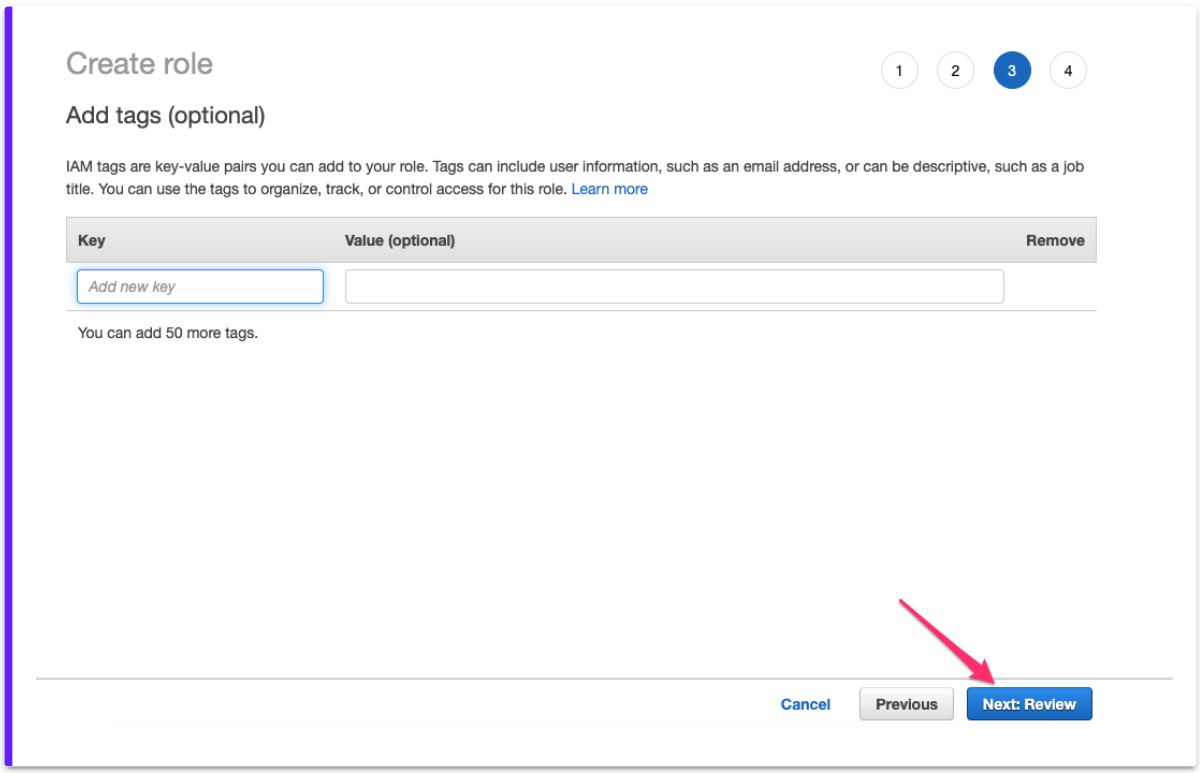
Create a new role `CodeDeployEC2ServiceRole` for EC2 instances that will be used to host your web application.

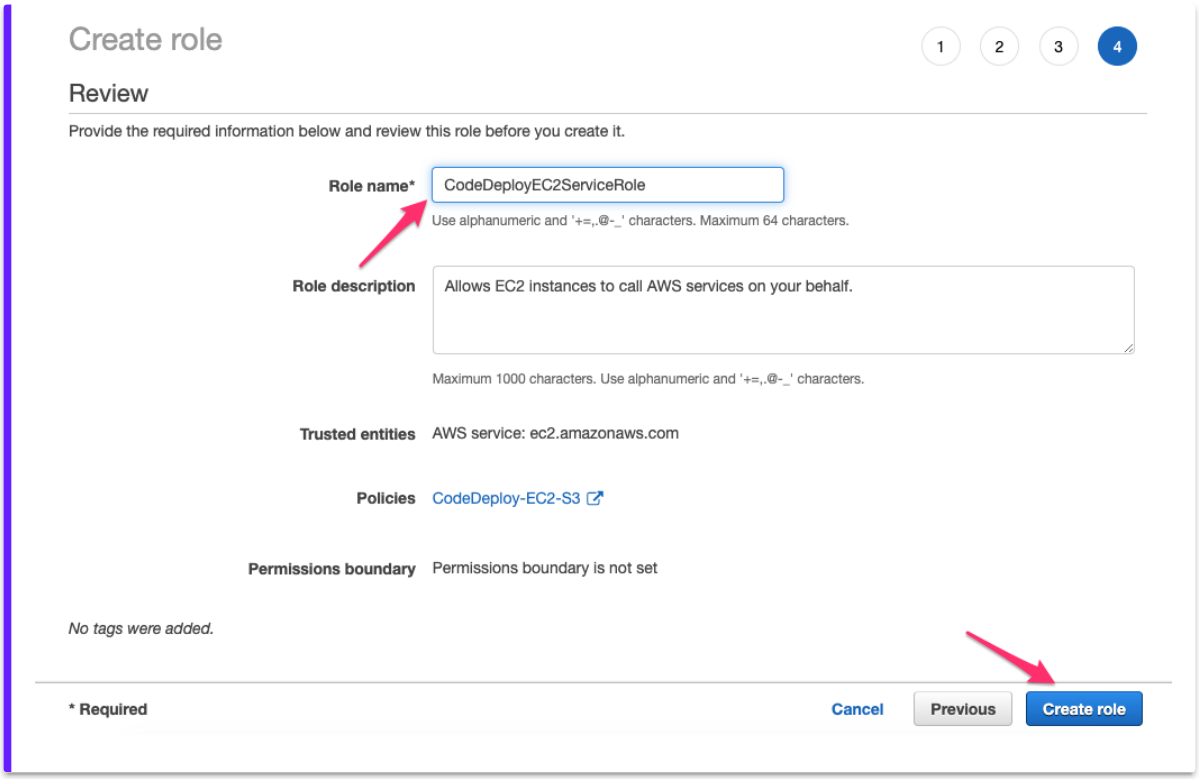



Create role
1 2 3 4
Select type of trusted entity
AWS service
EC2, Lambda and others
Another AWS account
Belonging to you or 3rd party
Web identity
Cognito or any OpenID provider
SAML 2.0 federation
Your corporate directory
Allows AWS services to perform actions on your behalf. Learn more
Choose the service that will use this role
EC2
Allows EC2 instances to call AWS services on your behalf.
Lambda
Allows Lambda functions to call AWS services on your behalf.
API Gateway
Comprehend
EMR
Kinesis
S3
AWS Backup
Config
ElastiCache
Lambda
SMS
AWS Support
Connect
Elastic Beanstalk
Lex
SNS
Amplify
DMS
Elastic Container Service
License Manager
SWF
AppSync
Data Lifecycle Manager
Elastic Transcoder
Machine Learning
SageMaker
Application Auto Scaling
Data Pipeline
ElasticLoadBalancing
Macie
Security Hub
Application Discovery
DataSync
Forecast
MediaConvert
Service Catalog
* Required
Cancel
Next: Permissions


Create role
1 2 3 4
Attach permissions policies
Choose one or more policies to attach to your new role.
Create policy
Filter policies
Search
Showing 16 results
Reset filters
POLICY TYPE
Customer managed (16)
AWS managed (524)
AWS managed - job function (10)
POLICY USE
Used for permissions (20)
Used for boundary (0)
Not used (530)
Used as
Description
None
Provides full access to AWS services an…
Permissions policy (2)
Set permissions boundary
* Required
Cancel
Previous
Next: Tags

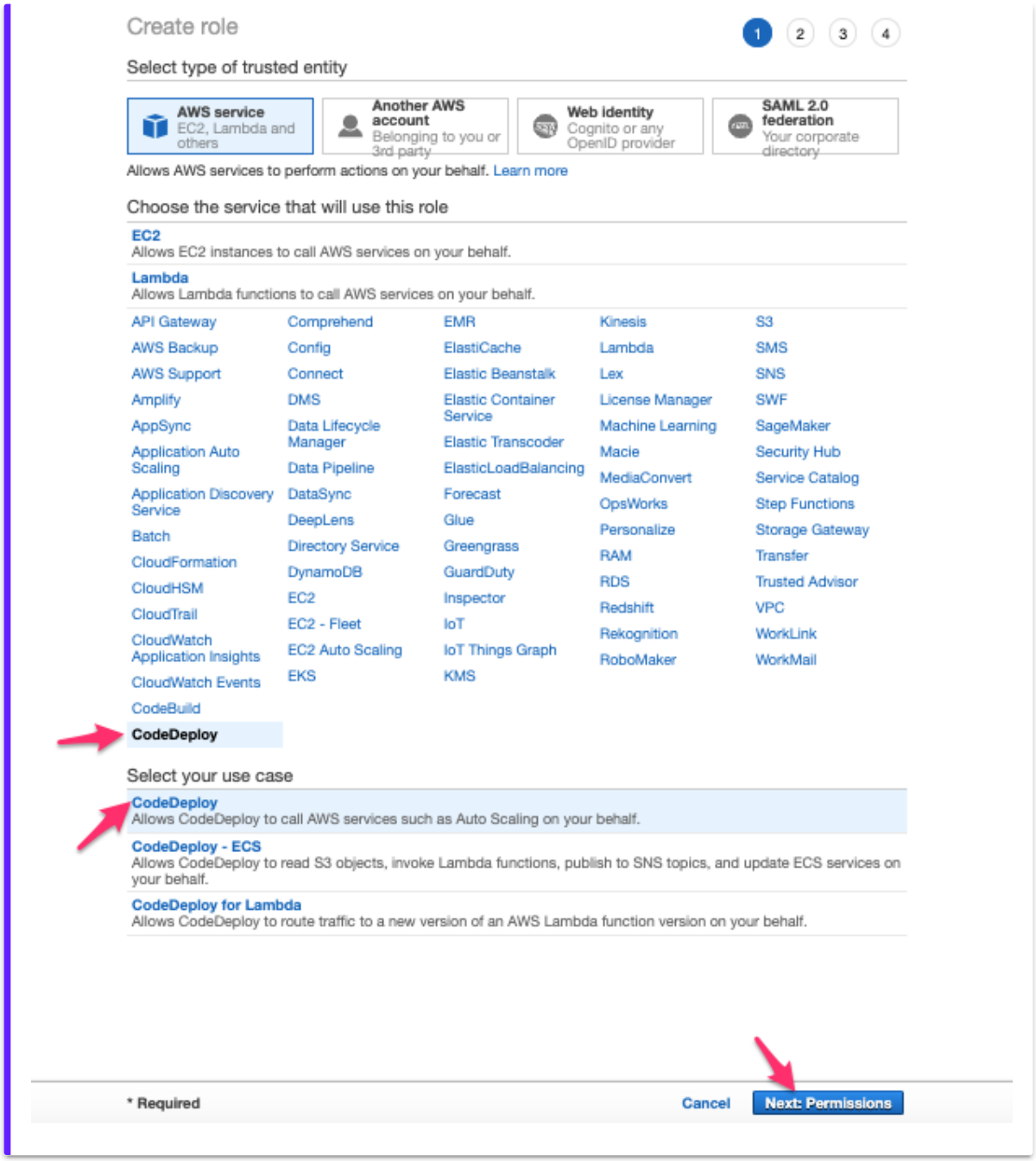

Create role
1
2
3
4
Attach permissions policies
Choose one or more policies to attach to your new role.
Create policy
Filter policies
Search
Showing 16 results
Policy name
Used as
Description
AdministratorAccess-201702021852
None
Provides full access to AWS services an…
circleci-ec2-ami
None
CircleCI-Upload-To-S3
None
CirlceCI-Code-Deploy
None
CodeDeploy-EC2-S3
Permissions policy (2)
CodeDeployDemo-EC2-Permissions
None
EC2-S3-Webapp
None
fall2018.csye6225.cloud
None
Set permissions boundary
* Required
Cancel
Previous
Next: Tags

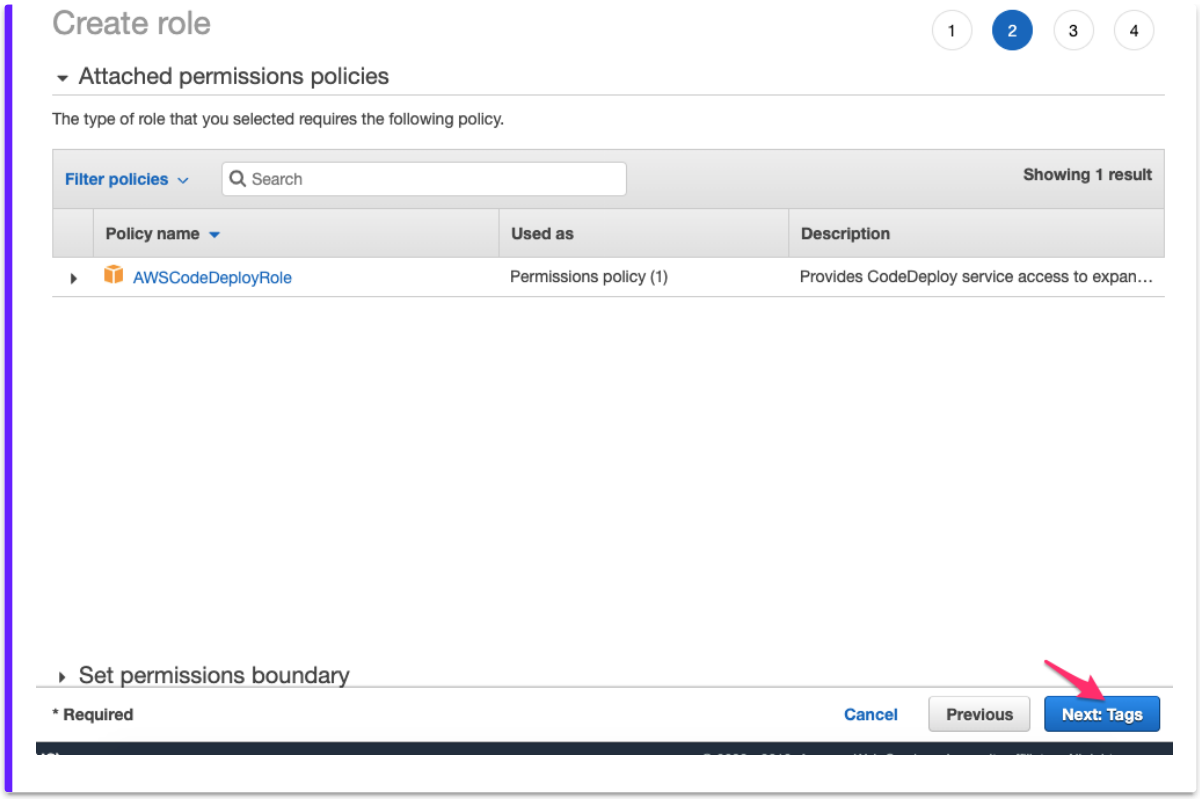

Create role
1
2
3
4
Add tags (optional)
IAM tags are key-value pairs you can add to your role. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this role. Learn more
Key
Value (optional)
Remove
Add new key
You can add 50 more tags.
Cancel
Previous
Next: Review

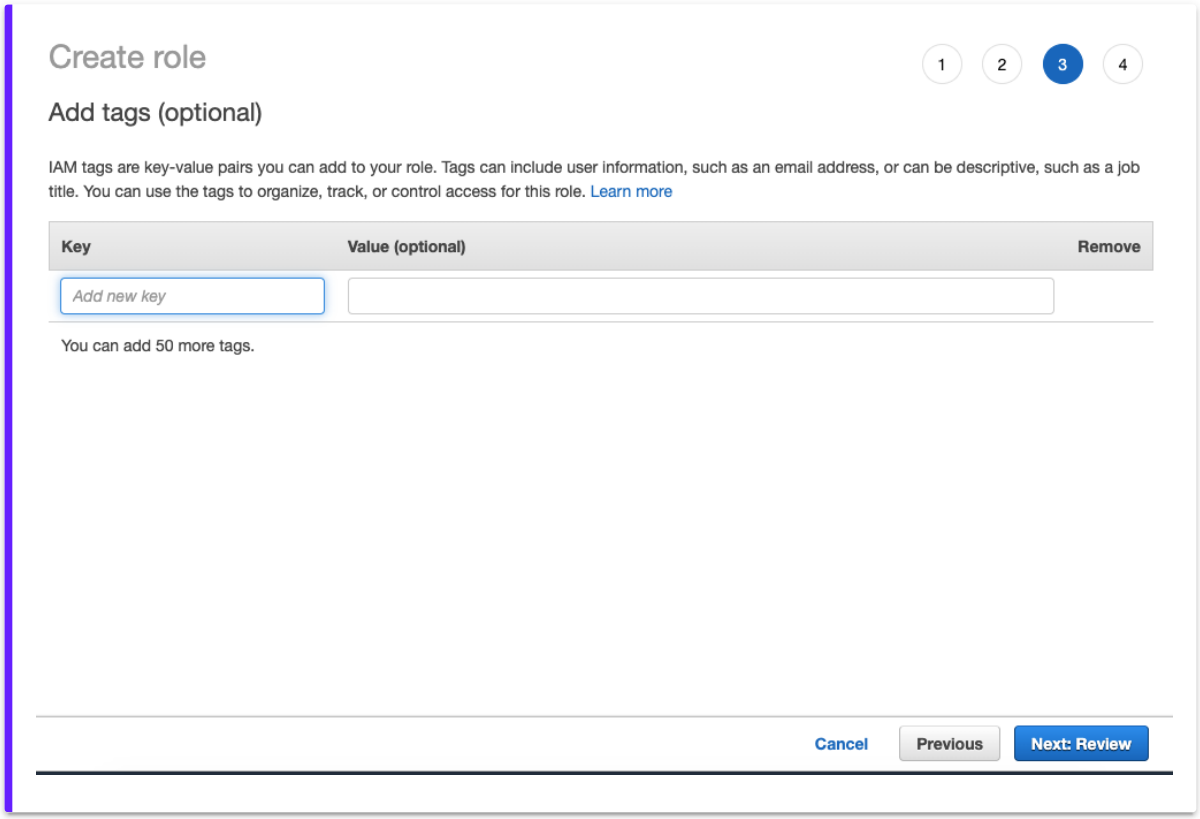


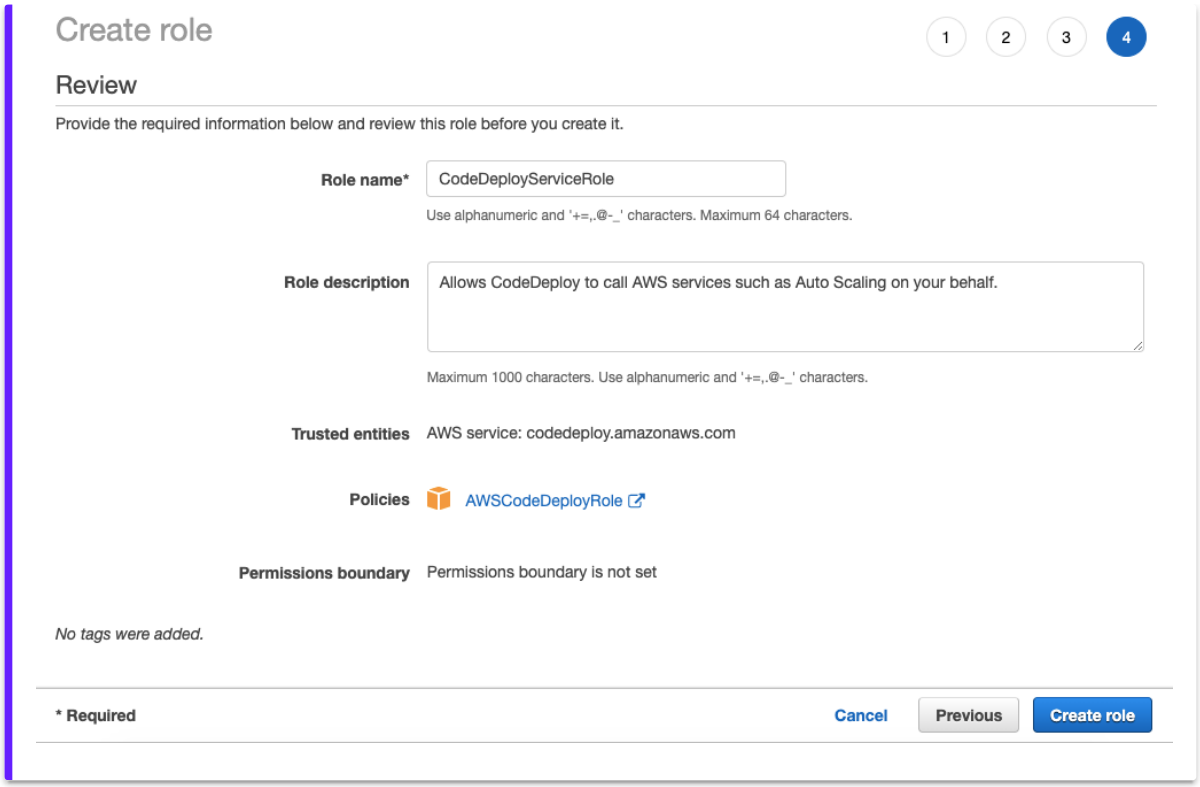
CodeDeployServiceRole IAM Role for CodeDeploy

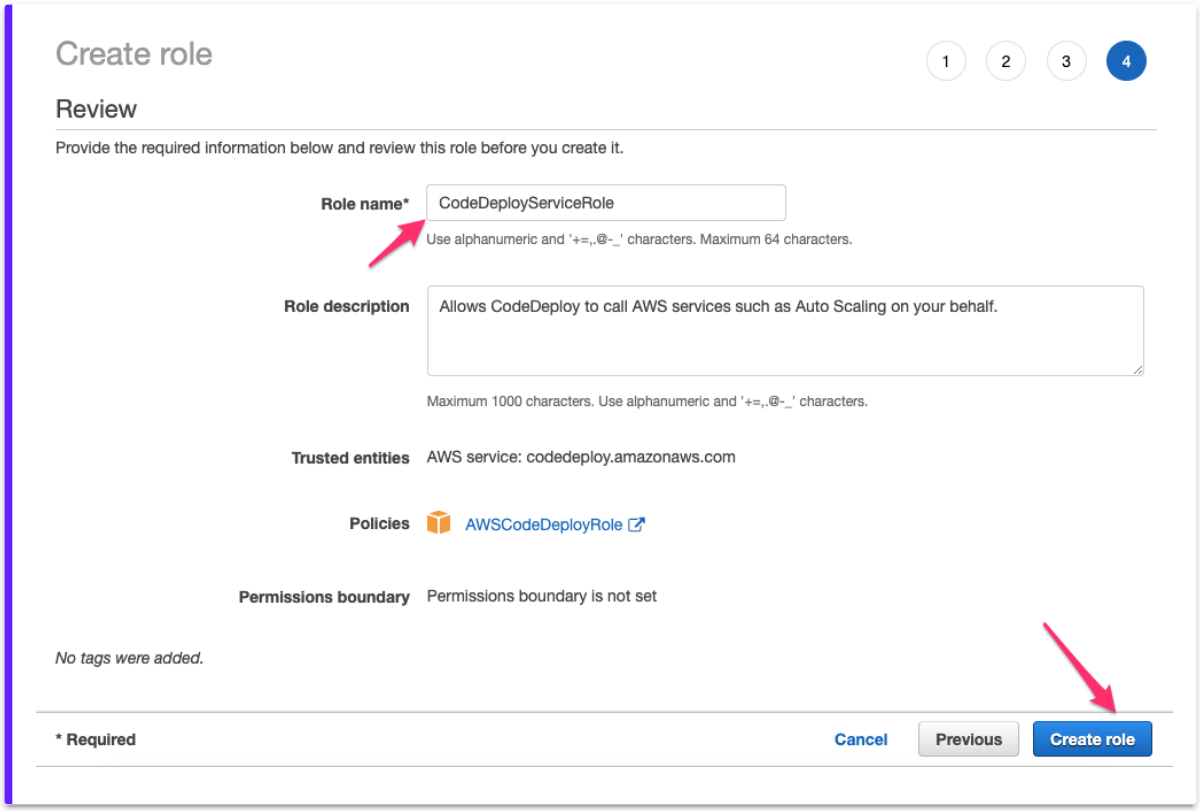
Create a new role `CodeDeployServiceRole` for EC2 instances that will be used to host your web application.


Create role
1 2 3 4
Select type of trusted entity
AWS service
EC2, Lambda and others
Another AWS account
Belonging to you or 3rd party
Web identity
Cognito or any OpenID provider
SAML 2.0 federation
Your corporate directory
Allows AWS services to perform actions on your behalf. Learn more
Choose the service that will use this role
EC2
Allows EC2 instances to call AWS services on your behalf.
Lambda
Allows Lambda functions to call AWS services on your behalf.
API Gateway
AWS Backup
AWS Support
Amplify
AppSync
Application Auto Scaling
Application Discovery Service
Batch
CloudFormation
CloudHSM
CloudTrail
CloudWatch Application Insights
CloudWatch Events
CodeBuild
CodeDeploy
Comprehend
Config
Connect
DMS
Data Lifecycle Manager
Data Pipeline
DataSync
DeepLens
Directory Service
DynamoDB
EC2
EC2 - Fleet
EC2 Auto Scaling
EKS
EMR
ElastiCache
Elastic Beanstalk
Elastic Container Service
Elastic Transcoder
ElasticLoadBalancing
Forecast
Glue
Greengrass
GuardDuty
Inspector
IoT
IoT Things Graph
KMS
Kinesis
Lambda
Lex
License Manager
Machine Learning
Macie
MediaConvert
OpsWorks
Personalize
RAM
RDS
Redshift
Rekognition
RoboMaker
S3
SMS
SNS
SWF
SageMaker
Security Hub
Service Catalog
Step Functions
Storage Gateway
Transfer
Trusted Advisor
VPC
WorkLink
WorkMail
Select your use case
CodeDeploy
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.
CodeDeploy - ECS
Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS services on your behalf.
CodeDeploy for Lambda
Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.
* Required
Cancel
Next: Permissions


Create role
1
2
3
4
Attached permissions policies
The type of role that you selected requires the following policy.
Filter policies
Search
Showing 1 result
Policy name
Used as
Description
AWSCodeDeployRole
Permissions policy (1)
Provides CodeDeploy service access to expan…
Set permissions boundary
* Required
Cancel
Previous
Next: Tags


Create role
1
2
3
4
Add tags (optional)
IAM tags are key-value pairs you can add to your role. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this role. Learn more
Key
Value (optional)
Remove
Add new key
You can add 50 more tags.
Cancel
Previous
Next: Review


Create role
1
2
3
4
Review
Provide the required information below and review this role before you create it.
Role name*
CodeDeployServiceRole
Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities
AWS service: codedeploy.amazonaws.com
Policies
AWSCodeDeployRole
Permissions boundary
Permissions boundary is not set
No tags were added.
* Required
Cancel
Previous
Create role


Create role
1
2
3
4
Review
Provide the required information below and review this role before you create it.
Role name*
CodeDeployServiceRole
Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities
AWS service: codedeploy.amazonaws.com
Policies
AWSCodeDeployRole
Permissions boundary
Permissions boundary is not set
No tags were added.
* Required
Cancel
Previous
Create role

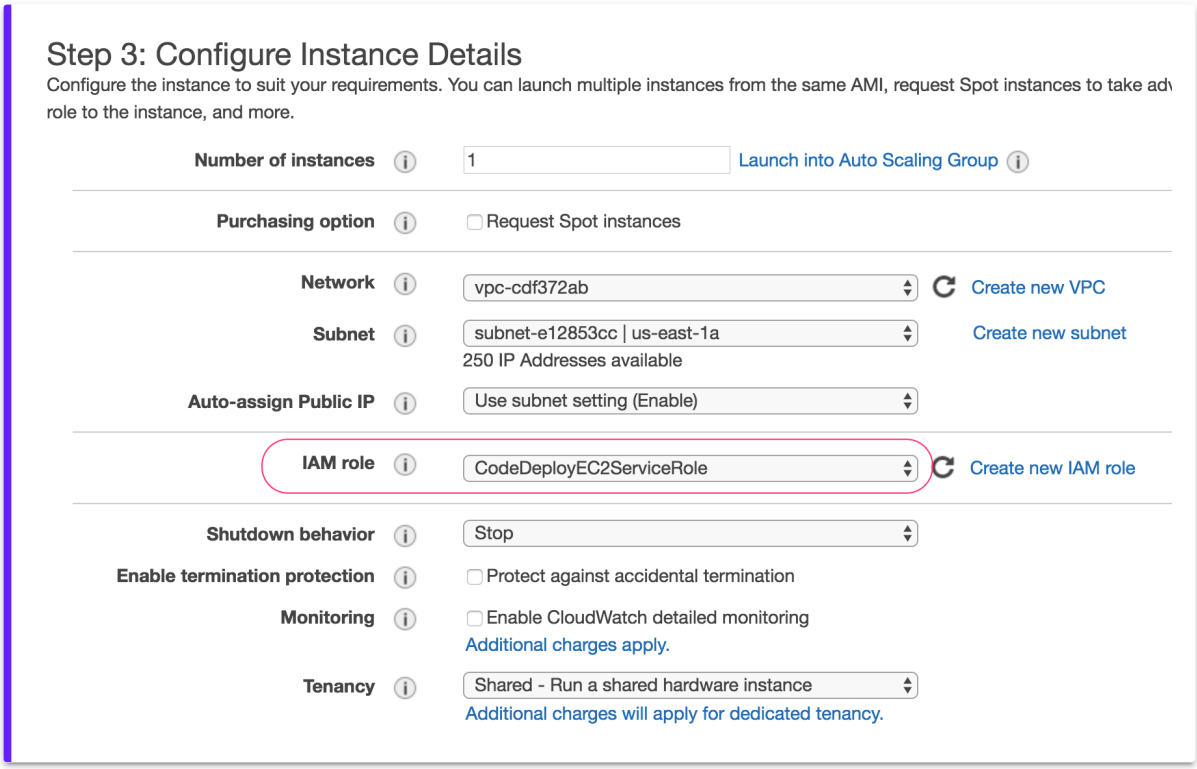
Create S3 bucket for CodeDeploy

> **Note**
>
> This setup can be done through AWS console.

Create a S3 bucket in same region as your EC2 instance. Bucket name should be `code-deploy.yourdomain.tld` where `yourdomain.tld` should be replaced with your domain name.

Create EC2 Instance to Host Application

Create one or more EC2 instance to host your application. This EC2 instance must have the IAM Role **CodeDeployEC2ServiceRole**. Tag the instance with `KEY` and `VALUE` of your choice. You will need the `KEY` and `VALUE` later when creating CodeDeploy application.



Install CodeDeploy Agent

1. Install [https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-linux.html] Code Deploy Agent from `us-east-1` region.

CodeDeploy Appspec

Create AWS CodeDeploy App Spec file. The AppSpec file is used to manage each deployment as a series of lifecycle events. Note that the **appspec.yml** file should be in root of your repository. See App Spec documentation here [http://docs.aws.amazon.com/codedeploy/latest/userguide/writing-app-spec.html].

```
version: 0.0
os: linux

files:
  - source: ./build/libs/ROOT.war
    destination: /var/lib/tomcat7/webapps

hooks:
  AfterInstall:
   - location: ./restartTomcat.sh
     timeout: 180
     runas: centos
```

Create CodeDeploy Application

1. Application Name - **csye6225-webapp**
2. Compute Platform - **EC2/On-premises**

Create CodeDeploy Deployment Group

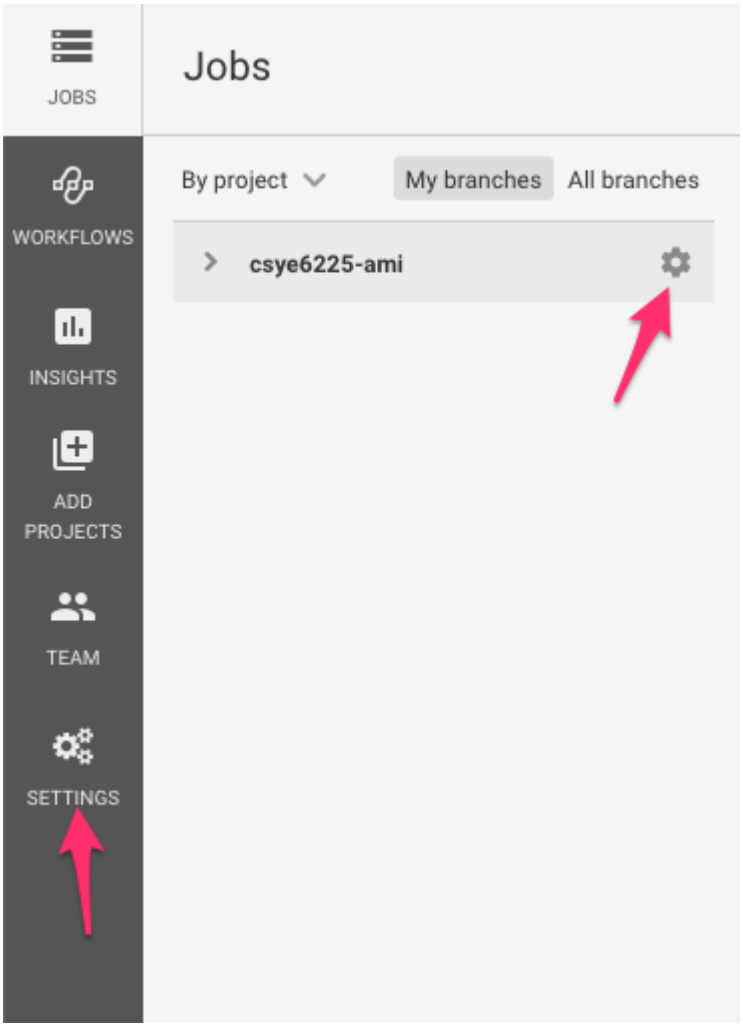
1. Deployment group name - **csye6225-webapp-deployment**

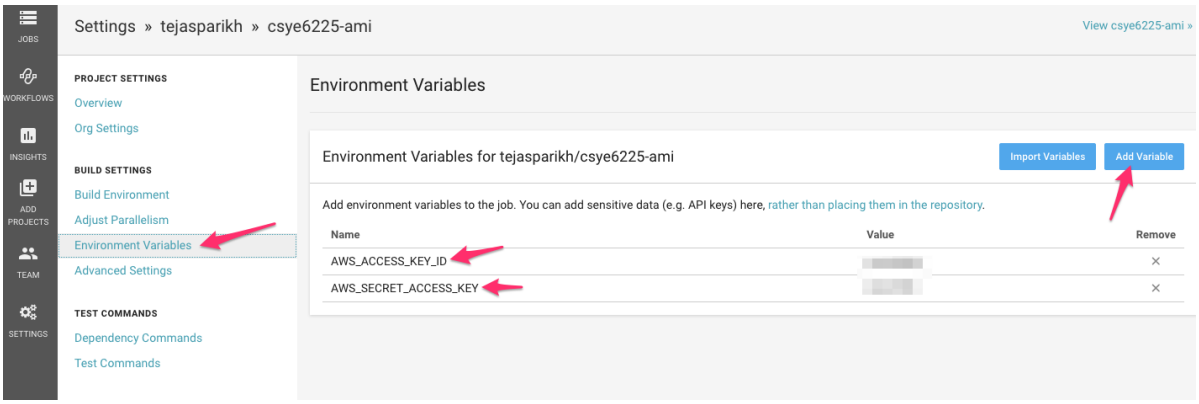
2. Service role - **CodeDeployServiceRole**
3. Deployment type - **In-place**
4. Environment Configuration - **Amazon EC2 Instances**
    a. Provide the tag group key and values.
5. Deployment settings - **CodeDeployDefault.AllAtOnce**
6. Load Balancer - **disabled**
7. Rollback - **Roll back when a deployment fails**

Everything else can be left to default values.

Setting CircleCI Environment Variable

1. Add your AWS access keys [https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys] to CircleCI as either project environment variables [https://circleci.com/docs/2.0/env-vars/#setting-an-environment-variable-in-a-project] or context environment variables [https://circleci.com/docs/2.0/env-vars/#setting-an-environment-variable-in-a-context].
2. Store your Access Key ID in a variable called `AWS_ACCESS_KEY_ID` and your Secret Access Key in a variable called `AWS_SECRET_ACCESS_KEY`.





CircleCI Config Example

Repo: https://github.com/tejasparikh/csye6225-spring2019-ami
[https://github.com/tejasparikh/csye6225-spring2019-ami]

```
version: 2
jobs:
  build:
    docker:
      - image: circleci/python:2.7-jessie
    steps:
      - checkout
      - run:
          name: Install packages
          command: sudo apt-get update && sudo apt-get install
wget zip unzip -y
      - run:
          name: Install awscli
          command: sudo pip install awscli
      - run:
          name: Download packer
          command: |
            wget -q
https://releases.hashicorp.com/packer/1.3.4/packer_1.3.4_linux_amd6

            unzip packer*.zip
            chmod +x packer
      - run:
          name: Validate Packer Template
          command: ./packer validate ubuntu-ami-template.json
      - run:
          name: Build AMI
          command: |
            ./packer build \
            -var "aws_region=${AWS_REGION}" \
            -var "subnet_id=${AWS_SUBNET_ID}" \
            ubuntu-ami-template.json
```

Documentation

AWS CLI

- CodeDeploy [https://docs.aws.amazon.com/cli/latest/reference/deploy/index.html]
- S3 API [https://docs.aws.amazon.com/cli/latest/reference/s3api/index.html]

EC2

- Instance Metadata and User Data [http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html]
- EC2 UserData [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html#cfn-ec2-instance-userdata]

Spring

- Set the active Spring profiles [https://docs.spring.io/spring-boot/docs/current/reference/html/howto-properties-and-configuration.html#howto-set-active-spring-profiles]

IAM & CloudFormation

- AWS::IAM::Role [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-role.html]
- AWS::IAM::Policy [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-policy.html]
- AWS::S3::Bucket [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-s3-bucket.html]
- Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances

[http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html]

AWS CodeDeploy

- AWS CodeDeploy AppSpec File Reference [http://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html]
- AWS CodeDeploy [http://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html]
- CodeDeploy [https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html]
- Overview of CodeDeploy Deployment Types [https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html#welcome-deployment-overview]
- Troubleshooting CodeDeploy [https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting.html]

CircleCI

- CircleCI [https://circleci.com/]
- CircleCI Documentation [https://circleci.com/docs]
- Using Environment Variables [https://circleci.com/docs/2.0/env-vars/]
- Pre-Built CircleCI Docker Images [https://circleci.com/docs/2.0/circleci-images/]
- Concepts [https://circleci.com/docs/2.0/concepts/#section=getting-started]

Trigger CircleCI Build Without GitHub Commit

You can trigger CircleCI job via api from command line using curl. See https://circleci.com/docs/2.0/api-job-trigger/ [https://circleci.com/docs/2.0/api-

job-trigger/].

```
curl -u ${CIRCLE_API_USER_TOKEN} \
     -d build_parameters[CIRCLE_JOB]=build \
     https://circleci.com/api/v1.1/project/<vcs-
type>/<org>/<repo>/tree/<branch>
```

Example API Call

Here's an example api call for triggering build for https://github.com/tejasparikh/csye6225-spring2019-ami [https://github.com/tejasparikh/csye6225-spring2019-ami] repository

```
curl -u 0a1d67cdO_PERSONAL_OR_PROJECT_TOKEN_cdbc356b0f5 \
    -d build_parameters[CIRCLE_JOB]=build \

https://circleci.com/api/v1.1/project/github/tejasparikh/csye6225-
spring2019-ami/tree/master
```

Note

you may get prompted for password "Enter host password for user" just hit enter key and it should work.

Output of the API call

```
{
  "compare" : null,
  "previous_successful_build" : {
    "build_num" : 6,
    "status" : "success",
    "build_time_millis" : 323972
  },
  "build_parameters" : {
    "CIRCLE_JOB" : "build"
  },
  "oss" : true,
  "committer_date" : null,
  "body" : null,
  "usage_queued_at" : "2019-03-10T21:23:16.702Z",
```

```
  "fail_reason" : null,
  "retry_of" : null,
  "reponame" : "csye6225-spring2019-ami",
  "ssh_users" : [ ],
  "build_url" : "https://circleci.com/gh/tejasparikh/csye6225-
spring2019-ami/7",
  "parallel" : 1,
  "failed" : null,
  "branch" : "master",
  "username" : "tejasparikh",
  "author_date" : null,
  "why" : "api",
  "user" : {
    "is_user" : true,
    "login" : "tejasparikh",
    "avatar_url" :
"https://avatars2.githubusercontent.com/u/25620460?v=4",
    "name" : "Tejas Parikh",
    "vcs_type" : "github",
    "id" : 25620460
  },
  "vcs_revision" : "01aff86f5e5dad1c9658a65c1a554cdd609d30d2",
  "vcs_tag" : null,
  "build_num" : 7,
  "infrastructure_fail" : false,
  "committer_email" : null,
  "previous" : {
    "build_num" : 6,
    "status" : "success",
    "build_time_millis" : 323972
  },
  "status" : "not_running",
  "committer_name" : null,
  "retries" : null,
  "subject" : null,
  "vcs_type" : "github",
  "timedout" : false,
  "dont_build" : null,
  "lifecycle" : "not_running",
  "no_dependency_cache" : false,
  "stop_time" : null,
  "ssh_disabled" : true,
  "build_time_millis" : null,
  "picard" : null,
  "circle_yml" : {
```

```
    "string" : "version: 2\njobs:\n  build:\n    docker:\n      -
image: circleci/python:2.7-jessie\n    steps:\n      - checkout\n
- run:\n          name: Install packages\n          command: sudo
apt-get update && sudo apt-get install wget zip unzip -y\n      -
run:\n          name: Install awscli\n          command: sudo pip
install awscli\n      - run:\n          name: Download packer\n
command: |\n            wget -q
https://releases.hashicorp.com/packer/1.3.4/packer_1.3.4_linux_amd6
 \n            unzip packer*.zip  \n            chmod +x packer\n
- run:\n          name: Validate Packer Template\n
command: ./packer validate ubuntu-ami-template.json\n      -
run:\n          name: Build AMI\n          command: |\n
./packer build \\\n            -var \"aws_region=${AWS_REGION}\"
\\\n            -var \"subnet_id=${AWS_SUBNET_ID}\" \\\n
ubuntu-ami-template.json\n\n"
  },
  "messages" : [ ],
  "is_first_green_build" : false,
  "job_name" : null,
  "start_time" : null,
  "canceler" : null,
  "platform" : "2.0",
  "outcome" : null,
  "vcs_url" : "https://github.com/tejasparikh/csye6225-
spring2019-ami",
  "author_name" : null,
  "node" : null,
  "canceled" : false,
  "author_email" : null
}
```


Troubleshooting EC2 User Data Script & CodeDeploy

1. Instance will be online before user data script execution completes. You can follow the log in `/var/log/cloud-init.log` to see what AWS is doing. This behavior is normal and expected.
2. Look for `util.py[DEBUG]: Running command ['/var/lib/cloud/instance/scripts/part-001'] with allowed return codes [0] (shell=False, capture=False)` in `/var/log/cloud-init.log` to see user data script execution result code.

3. Full path for shell script created from instance's user data is `/var/lib/cloud/instance/scripts/part-001` . You can run this script manually on shell `./part-001` as root user to follow the execution and see the error.
4. `CodeDeploy fails : HEALTH_CONSTRAINTS The overall deployment failed because too many individual instances failed deployment, too few healthy instances are available for deployment, or some instances in your deployment group are experiencing problems." (HEALTH_CONSTRAINTS)` - This is a very generic error and you have to usually click on deployment to get down to exact issue. Also check logs in `/var/log/aws/codedeploy-agent` on your EC2 instance if code deploy made it that far. Usually students encounter errors due to incorrect path for files or deployment directory. Also make sure that IAM role is attached to the EC2 instance.
5. See https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting-deployments.html [https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting-deployments.html] for debugging CodeDeploy issues.
6. `Download Bundle failed - The specified key does not exist.` - This error indicates that file does not exist in S3 bucket.
7. Creating AWS IAM Role using cloudformation does not create RolePolicies. [https://stackoverflow.com/questions/43300573/creating-aws-iam-role-using-cloudformation-does-not-create-rolepolicies]