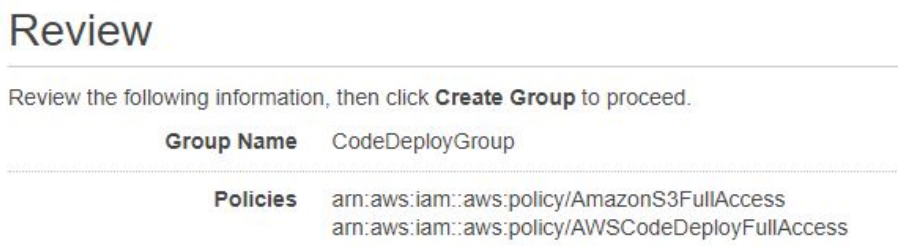
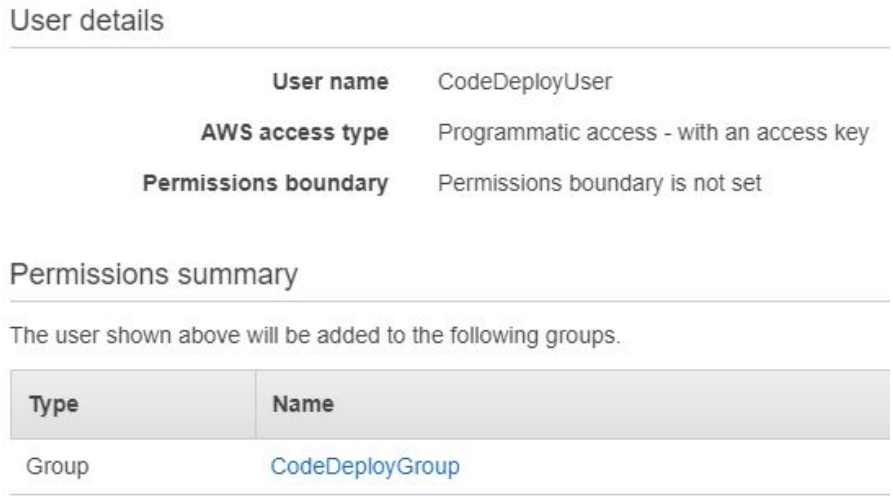
Auto-Deploy to EC2 with AWS CodeDeploy from Bitbucket Pipelines

STEP 1. Log in to AWS Management Console
https://console.aws.amazon.com/console/home

STEP 2. Create an IAM group
Go to Services > Find and select 'IAM' > Groups > Create New Group
Create an IAM group called *CodeDeployGroup* > Next Step > Find and select the following permissions *AmazonS3FullAccess* and *AWSCodeDeployFullAccess* > Next Step > Create Group



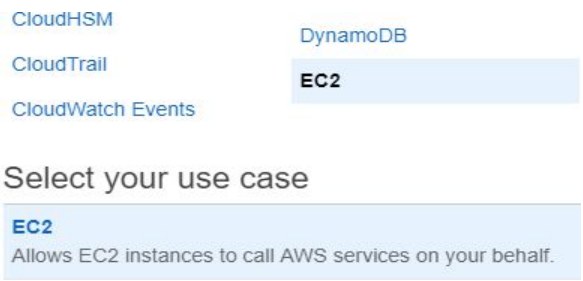
STEP 3. Create an IAM user
Go to Users > Add user > User name: *CodeDeployUser* > Access type: Programmatic access > Next: Permissions > Add user to our created group *CodeDeployGroup* > Next: Tags > Next: Review > Create user > You will see page with keys.



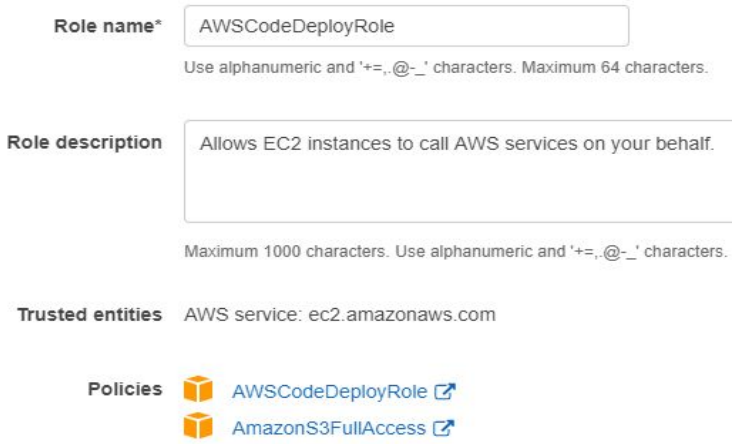
Download .csv file > Make note of this user's access key. It will be required later > Close

STEP 4. Create a role

We need to create a role that can be associated to EC2 instances and interact with CodeDeploy Go to Roles > Create role > Select type of trusted entity: AWS service > Select EC2 from list > Select your use case: EC2 > Next: Permissions



Find and select *AWSCodeDeployRole* and *AmazonS3FullAccess* > Next: Tags > Next: Review > Role name: *AWSCodeDeployRole* > Create role

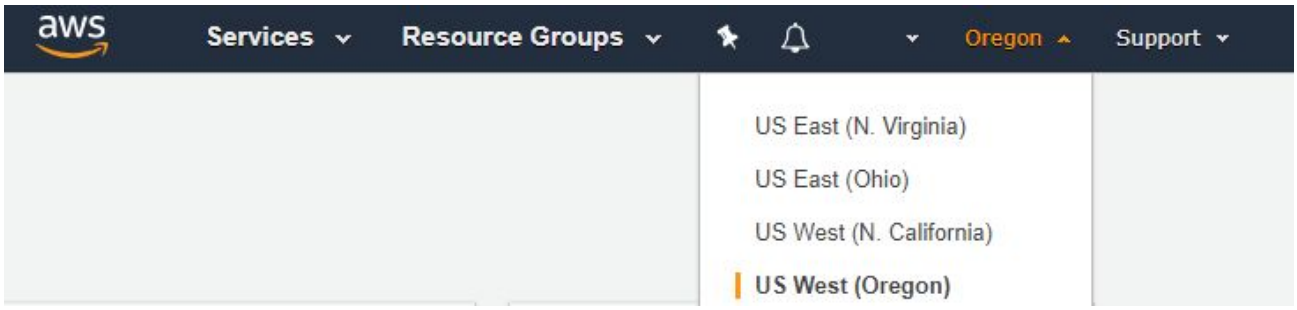


After creating the role, we need to edit the **Trust Relationship.** Change the region to the region you are working out of. Open created *AWSCodeDeployRole* role in Roles tab > Trust relationship > Edit trust relationship > Update Trust Policy as in the image below.

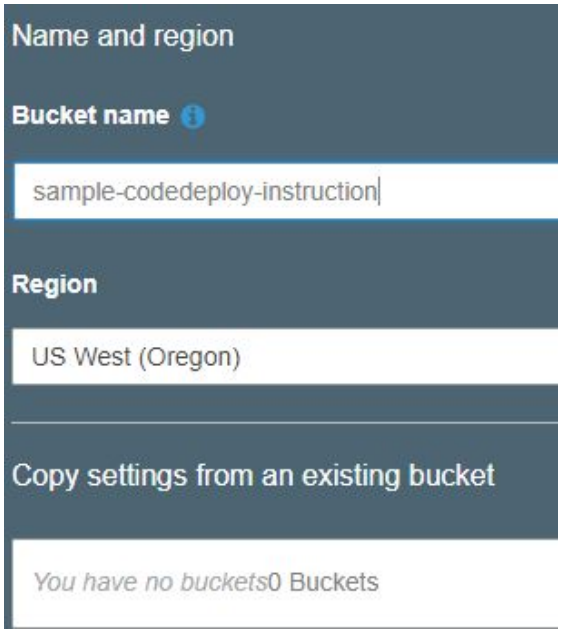
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
            "ec2.amazonaws.com",
            "codedeploy.us-west-2.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

STEP 5. Create S3 bucket

Services > Find S3 > Create bucket > Bucket name: *sample-codedeploy-instruction* >

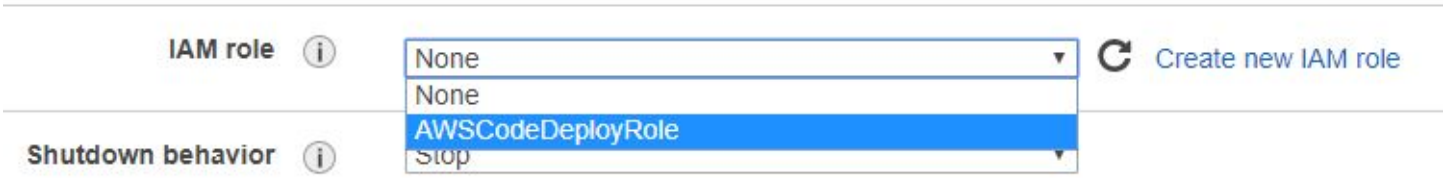


> Region: Select your region > Avoid 2, 3, 4 steps > Create



STEP 6. Create EC2 Instance

Services > Find EC2 > Launch Instance > (I choose **Ubuntu Server 16.04 LTS** for instruction) > Select > Next: Configure Instance Details > Select an IAM role that we created (AWSCodeDeployRole) >



> Next: Add storage > Next: Add Tags > Add Tag > Key: Name, Value: code-deploy-instance >

Make note of key and value. It will be required later. > Next: Configure Security Group >
> Assign a security group: Create a new security group and fill as in the picture below >

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
Custom TCP F	TCP	3000	Anywhere	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

> Review and Launch > Launch > Create a new key pair > Key pair name: ec2-key > Download Key Pair and save it > then, from drop list you need to select 'Choose an existing key pair' and select created 'ec2-key' > Launch Instances

STEP 7. Install PuTTY
https://www.putty.org/ It installs both PuTTY and PuTTYgen which we'll use to convert *.pem to *.ppk. At the end of step 6 we got the key ec2-key.pem.
Open PuTTYgen > File > Load private key > Select All Files (*.*), find ec2-key.pem and open it > OK > Save private key > Yes > call it the same as .pem key > close PuTTYgen

STEP 8. Install CodeDeploy Agent on EC2 instance (Ubuntu 16.04)
In AWS console > Services > EC2 > Instances > Copy Public DNS (IPv4)
Open PuTTY > Host Name: ubuntu@-paste-here-your IPv4- > Port: 22 > Connection type: SSH
Connection > SSH > Auth > Browse your converted private key 'ec2-key.ppk' > Open > Yes

Now install CodeDeploy agent as per your instance type:
Linux Server:
http://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-linux.html

Ubuntu Server:
http://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-ubuntu.html

Windows Server:
http://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-windows.html

In this guide we use second link

We need to execute a few commands in PuTTY to install CodeDeploy Agent

1. sudo apt-get update
2. sudo apt-get install ruby
3. sudo apt-get install wget
4. cd /home/ubuntu
5. wget https://*bucket-name*.s3.amazonaws.com/latest/install

   where *bucket-name* is 'aws-codedeploy-us-west-2' in our case.

Region name	*bucket-name* replacement	Region identifier
US East (Ohio)	aws-codedeploy-us-east-2	us-east-2
US East (N. Virginia)	aws-codedeploy-us-east-1	us-east-1
US West (N. California)	aws-codedeploy-us-west-1	us-west-1
US West (Oregon)	aws-codedeploy-us-west-2	us-west-2
Canada (Central)	aws-codedeploy-ca-central-1	ca-central-1
EU (Ireland)	aws-codedeploy-eu-west-1	eu-west-1
EU (London)	aws-codedeploy-eu-west-2	eu-west-2
EU (Paris)	aws-codedeploy-eu-west-3	eu-west-3
EU (Frankfurt)	aws-codedeploy-eu-central-1	eu-central-1
Asia Pacific (Tokyo)	aws-codedeploy-ap-northeast-1	ap-northeast-1
Asia Pacific (Seoul)	aws-codedeploy-ap-northeast-2	ap-northeast-2
Asia Pacific (Singapore)	aws-codedeploy-ap-southeast-1	ap-southeast-1
Asia Pacific (Sydney)	aws-codedeploy-ap-southeast-2	ap-southeast-2
Asia Pacific (Mumbai)	aws-codedeploy-ap-south-1	ap-south-1
South America (São Paulo)	aws-codedeploy-sa-east-1	sa-east-1

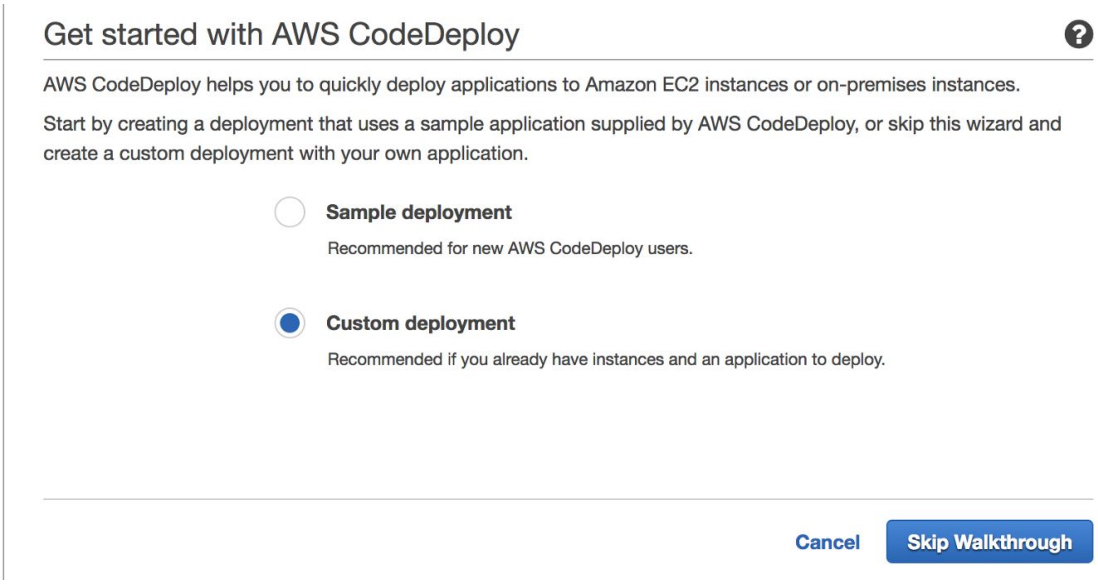
wget https://aws-codedeploy-us-west-2.s3.amazonaws.com/latest/install

6. chmod +x ./install
7. sudo ./install auto
8. sudo service codedeploy-agent start
9. sudo service codedeploy-agent status

STEP 9. CodeDeploy

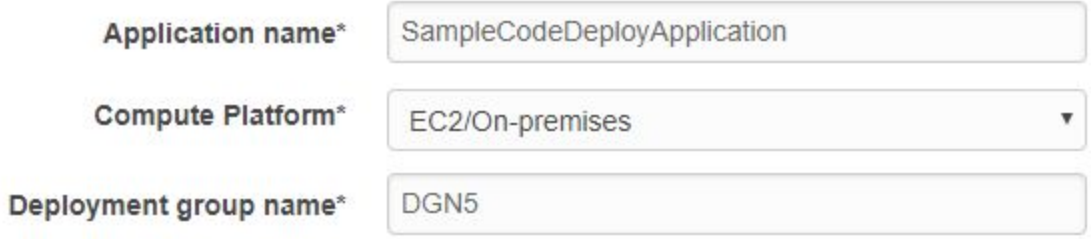
Service > Find CodeDeploy > Return to the old experience > Get started now

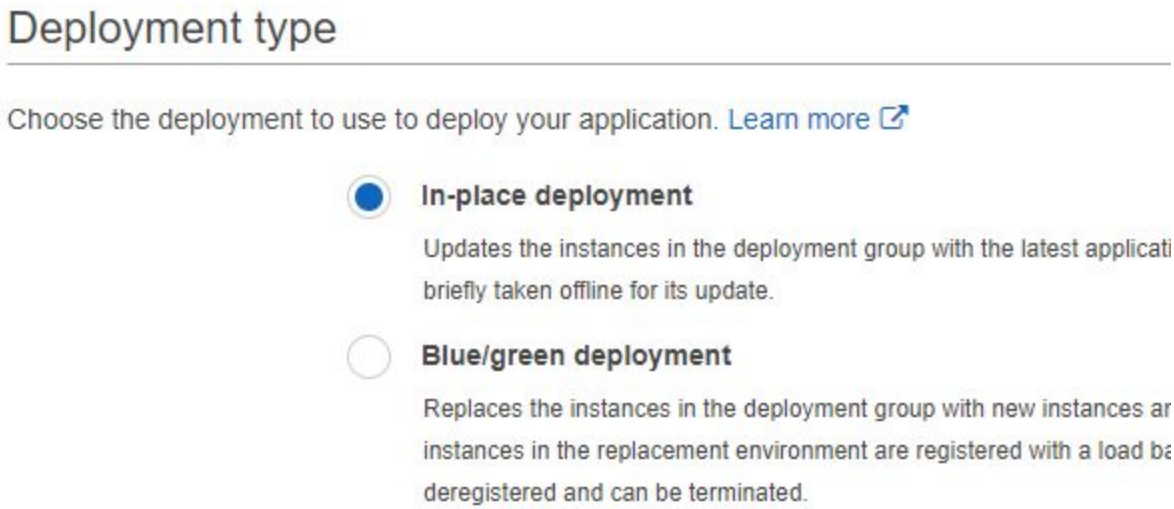
If this is your first time in CodeDeploy, you will get to the following screen.



Select Custom deployment > Skip Walkthrough

Next please fill in as shown in the following pictures:





Select Amazon EC2 instances

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instance

Auto Scaling groups | **Amazon EC2 instances** | On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. Learn more

One tag group : Any instance identified by the tag group will be deployed to.

Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Name	code-deploy-instance	1	
2				

Add tag group

Matching instances

Right now these instances match the criteria you specified. There might be more or fewer instances that match your criteria when a depl runs.

1 to 1 of 1 instances

Instance ID	Status	Filter types	Association
i-0b7[redacted]139	Running	Name:code-deploy-instance	EC2

Disable load balancing

Enable load balancing

Load balancer Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

Application Load Balancer or Network Load Balancer

Choose a target group

Classic Load Balancer

Select Deployment configuration: CodeDeployDefault.OneAtATime

Deployment configuration

Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application will be deployed and the success or failure conditions for a deployment.

Deployment configuration CodeDeployDefault.OneAtATime or **Create deployment configuration**

Routes traffic to one instance in the replacement environment at a time. Succeeds if traffic is successfully rerouted to all replacement instances. Fails after the very first rerouting failure. Allows the deployment to succeed for some instances, even if the overall deployment fails.

Service role

Select a service role that grants AWS CodeDeploy access to the instances.

Service role ARN* arn:aws:iam::111[redacted]32:role/AWSCodeDeployF

You can find Service role ARN* > Go to Services > IAM > Roles > Open AWSCodeDeployRole that we created in STEP 4 > Copy to clipboard.

Roles > AWSCodeDeployRole

Summary

Role ARN	arn:aws:iam::111[redacted]32:role/AWSCodeDeployRole
Role description	Allows EC2 instances to call AWS services on your behalf. | Edit
Instance Profile ARNs	arn:aws:iam::111[redacted]32:instance-profile/AWSCodeDeployRole
Path	/
Creation time	2019-01-11 17:53 UTC+0200
Maximum CLI/API session duration	1 hour Edit

Copy to clipboard

STEP 10. Bitbucket

Bitbucket repository settings > Pipelines | Settings > Enable Pipelines

Bitbucket repository settings > Repository variables. Now we need to tell Bitbucket about all of this information. Set the following Environment variables for the repository. The AWS_SECRET_ACCESS_KEY and AWS_ACCESS_KEY_ID are for the user created at the beginning, not your personal AWS user account.

Repository variables

Environment variables added on the repository level can be accessed by any users with push permissions in the repository. To access a variable, put the $ symbol in front of its name. For example, access AWS_SECRET by using $AWS_SECRET. For more information, see account variables.

Repository variables override variables added on the account level. View account variables

Name	Value	Add
S3_BUCKET	sample-codedeploy-instruction	
DEPLOYMENT_GROUP_NAME	DGN5	
DEPLOYMENT_CONFIG	CodeDeployDefault.OneAtATime	
AWS_DEFAULT_REGION	us-west-2	
APPLICATION_NAME	SampleCodeDeployApplication	
AWS_ACCESS_KEY_ID	AKIAI[redacted]YGRXA	
AWS_SECRET_ACCESS_KEY	YvJecbL[redacted]x0NvOz7	

STEP 11. Adding files in repository for auto-deploy(we use nodejs and express for this guide)

It is assumed that you have already installed node and npm on your EC2 instance.
We need the following files in our project directory: appspec.yml, bitbucket-pipelines.yml, codedeploy_deploy.py, scripts/startApp.sh. You can modify these files according to your project dependencies and needs.

You can download these files from AWS bitbucket repository - https://bitbucket.org/awslabs/aws-codedeploy-bitbucket-pipelines-python/src/master/

This is how these files look for this guide:

appspec.yml

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/
hooks:
  AfterInstall:
    - location: scripts/startApp.sh
      timeout: 300
      runas: root
```

bitbucket-pipelines.yml

```
image: python:3.5.1

pipelines:
  branches:
    master:
      - step:
          script:
            - apt-get update # required to install zip
            - apt-get install -y zip # required for packaging up the application
            - pip install boto3==1.3.0 # required for codedeploy_deploy.py
            - zip -r /tmp/artifact.zip * # package up the application for deployment
            - python codedeploy_deploy.py # run the deployment script
```

script/startApp.ch

```
fuser -k 3000/tcp
cd /var/www/html
mkdir -p logs
npm -v
npm install
```

codedeploy_deploy.py

In most cases, this file does not need to be edited.

After that, we need to add these files to the repository. As soon as they are added, the process of automatic deployment begins. Everytime you push new changes on bitbucket repository the process of automatic deployment begins. You can track this process in the repository ‘Pipelines’ tab.