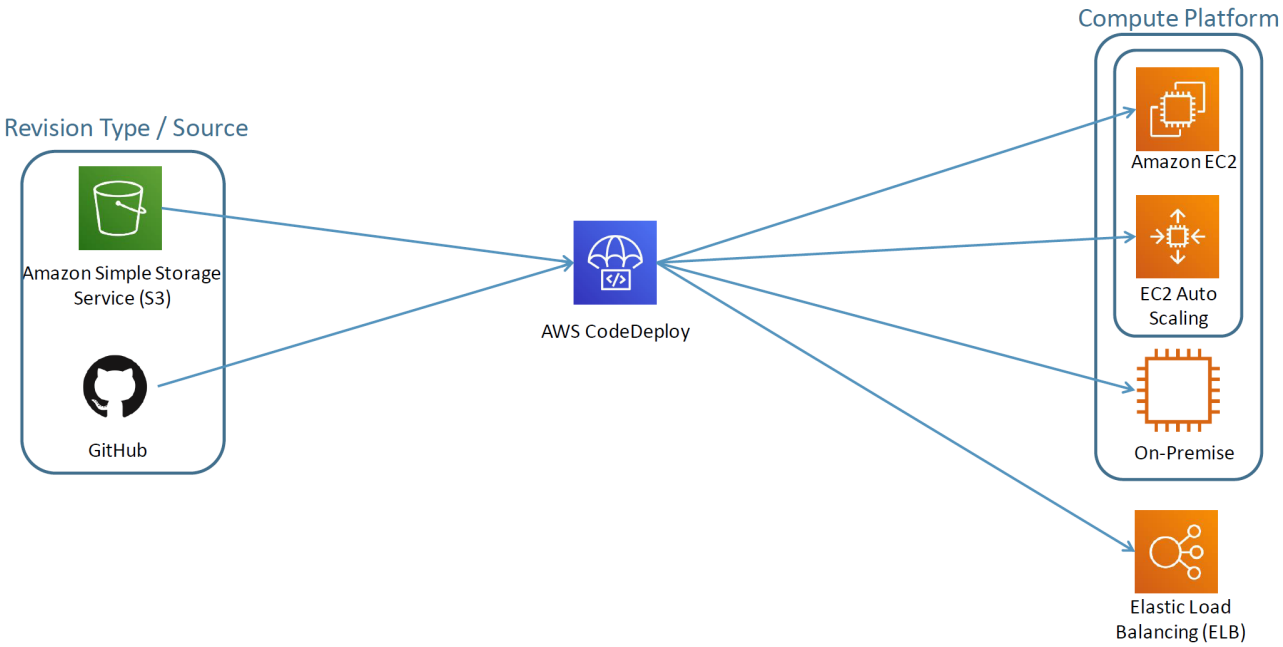
3. CI/CD in AWS - Part II

Code Deploy

CodeDeploy is a development service that automates application deployments to EC2 instances, On-premises instances, AWS lambda, AWS ECS

New features can be released rapidly, avoiding downtime



Pre-requisites

1. Create Service Role for CodeDeploy - To handle the EC2 instances
2. Create an IAM instance profile - For EC2 instance, which will have permission to access the S3 bucket to bring the artifact to the EC2

Service Role
IAM Service Create Role AWS Service CodeDeploy

Select trusted entity

Trusted entity type

- (●) **AWS service** — Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ( ) **AWS account** — Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ( ) **Web identity** — Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ( ) **SAML 2.0 federation** — Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ( ) **Custom trust policy** — Create a custom trust policy to enable others to perform actions in this account.

Use case

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

Common use cases

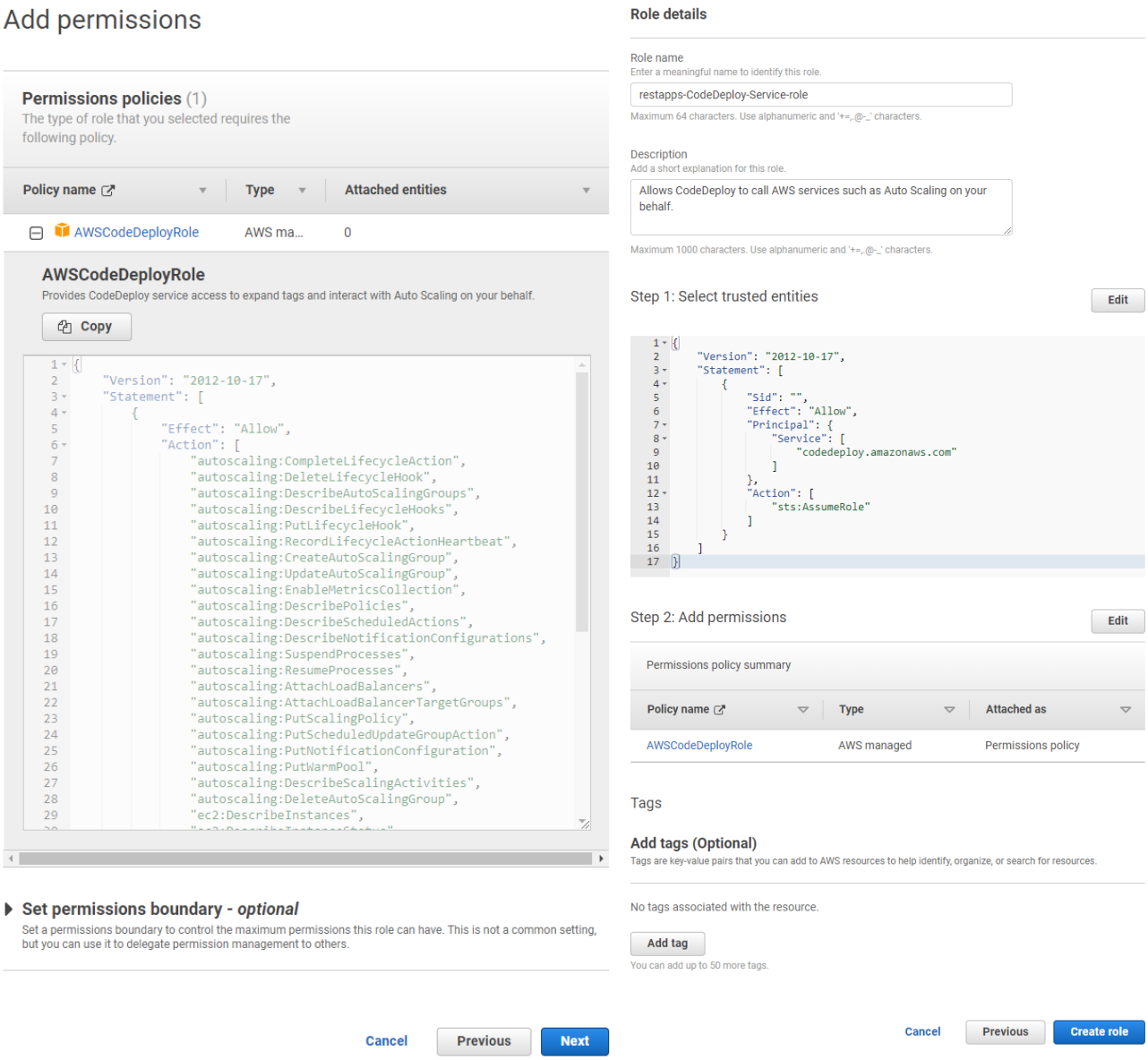
- ( ) **EC2** — Allows EC2 instances to call AWS services on your behalf.
- ( ) **Lambda** — Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

CodeDeploy ▾

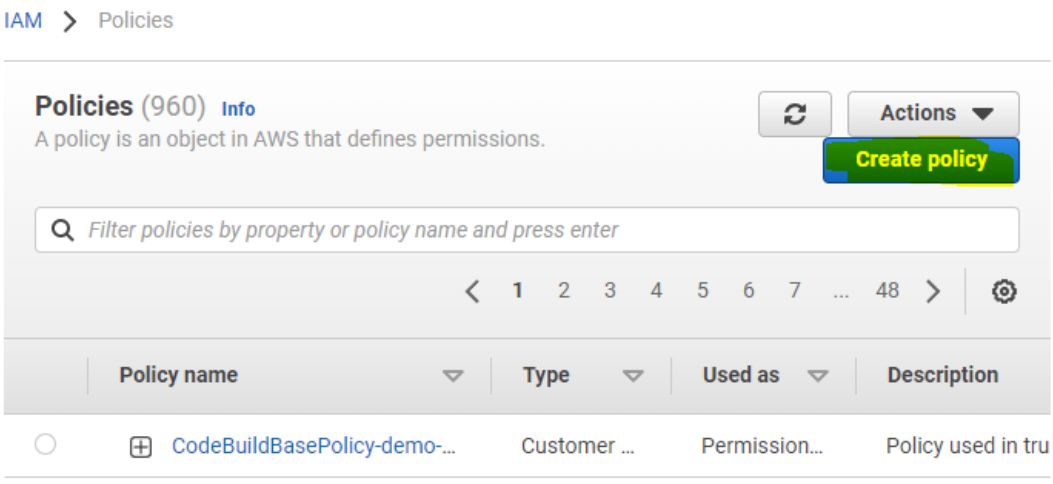
- (●) **CodeDeploy** — Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.
- ( ) **CodeDeploy for Lambda** — Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.
- ( ) **CodeDeploy - ECS** — Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS services on your behalf.

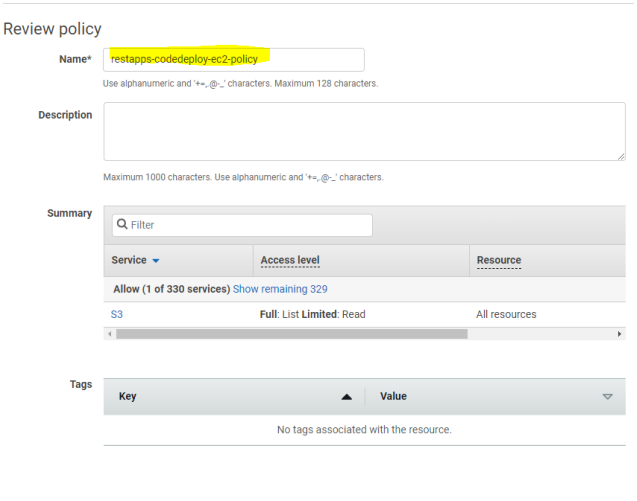
Cancel **Next**

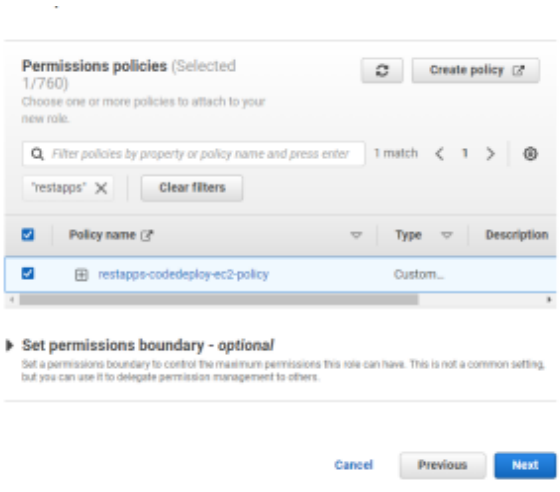


IAM Role

IAM Service Create Policy



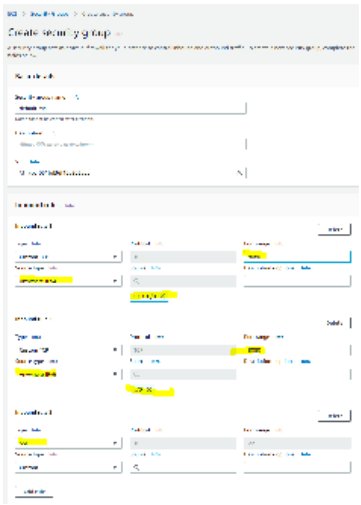


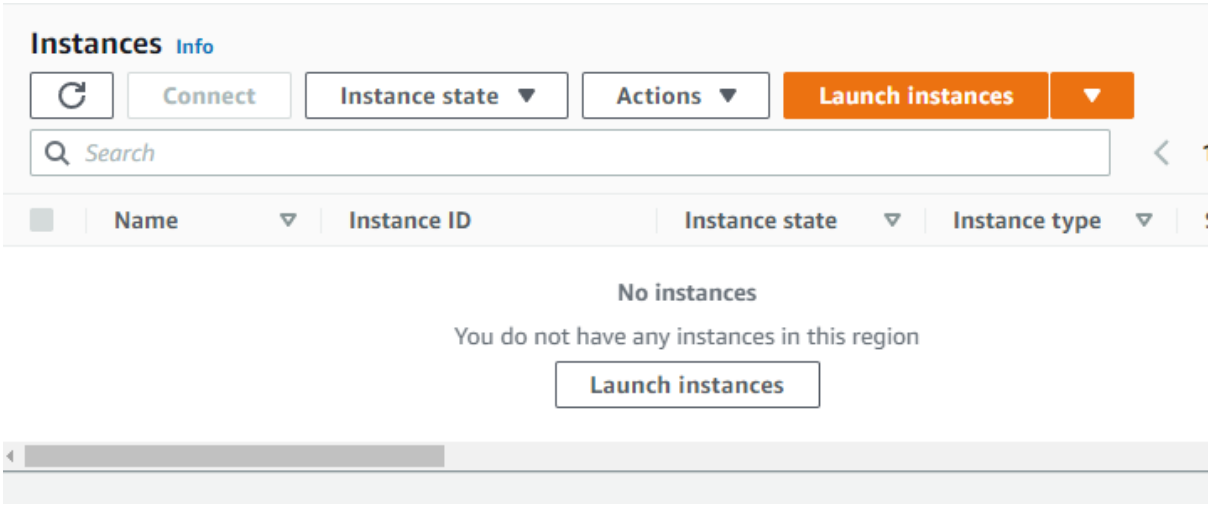


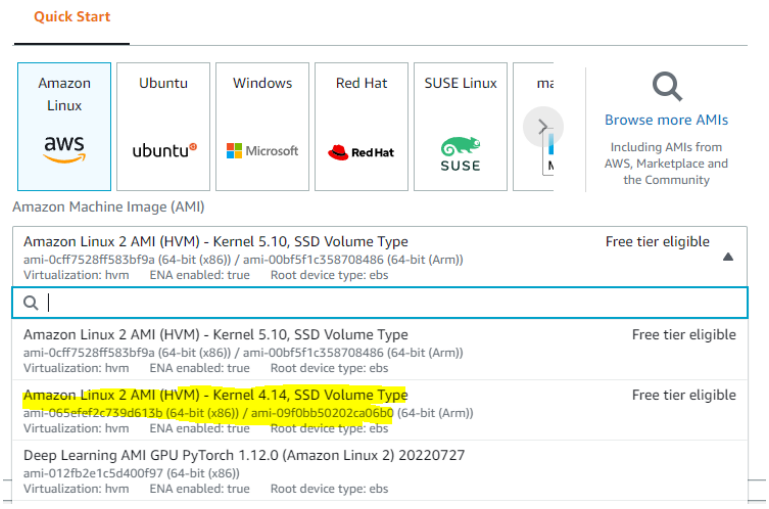
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action":[
                "s3:Get*",
                "s3:List*"
            ],
            "Effect" : "Allow",
            "Resource" : "*"
        }
    ]
}
```

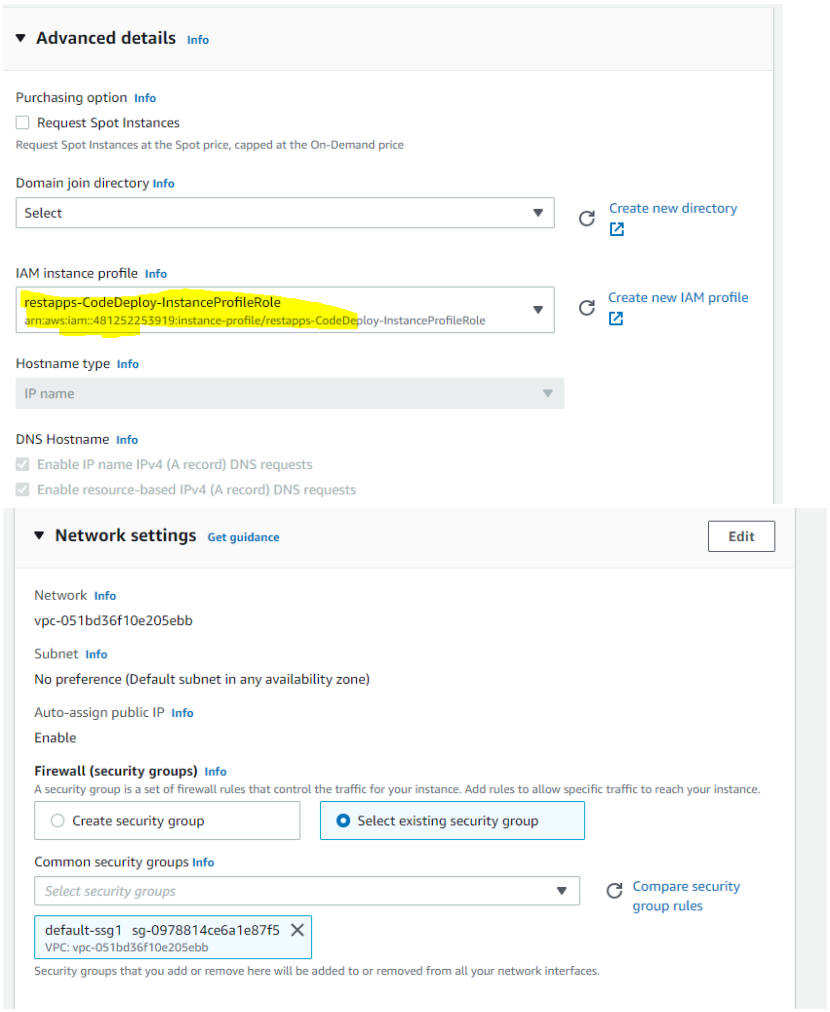
Create an EC2 VM

Step 1: Create EC2 VM
Access EC2 service Security Groups Create Security Groups




Instances Info
Connect
Instance state
Actions
Launch instances
Search
Name
Instance ID
Instance state
Instance type
No instances
You do not have any instances in this region
Launch instances

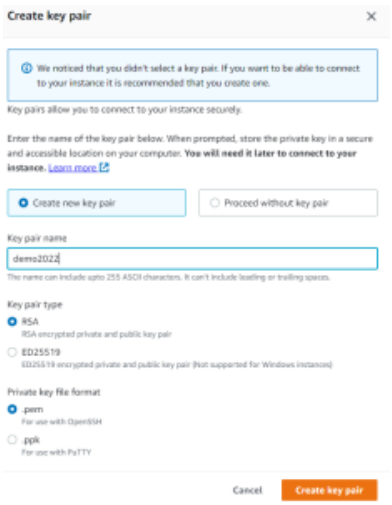

Quick Start
Amazon Linux
Ubuntu
Windows
Red Hat
SUSE Linux
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type
ami-0cff7528ff583bf9a (64-bit (x86)) / ami-00bf5f1c358708486 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Free tier eligible
Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type
ami-0cff7528ff583bf9a (64-bit (x86)) / ami-00bf5f1c358708486 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Free tier eligible
Amazon Linux 2 AMI (HVM) - Kernel 4.14, SSD Volume Type
ami-065efef2c739d613b (64-bit (x86)) / ami-09f0bb50202ca06b0 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Free tier eligible
Deep Learning AMI GPU PyTorch 1.12.0 (Amazon Linux 2) 20220727
ami-012fb2e1c5d400f97 (64-bit (x86))
Virtualization: hvm ENA enabled: true Root device type: ebs



UserData What needs to be in the VM, mentioned in the UserData. Region for code deploy is us-east-1 because the services are launched in us-east-1

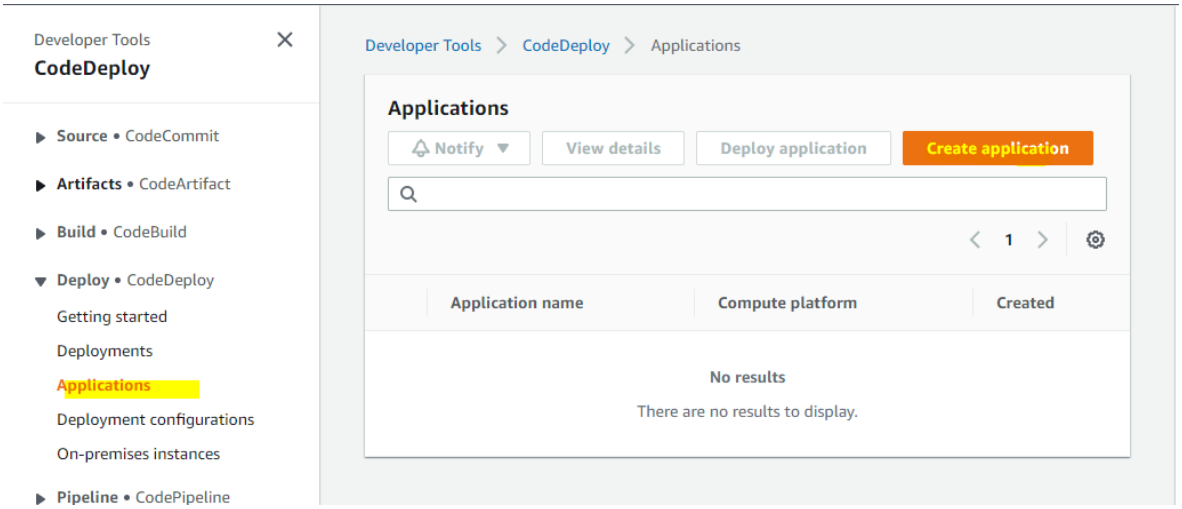
```
#!/bin/bash
yum update
yum -y install ruby
yum -y install wget
yum -y erase java-1.7.0-openjdk.x86_64
yum -y install java-1.8.0-openjdk.x86_64
yum -y install java-1.8.0-openjdk-devel
sudo amazon-linux-extras install tomcat9
sudo systemctl enable tomcat
cd /home/ec2-user
wget https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

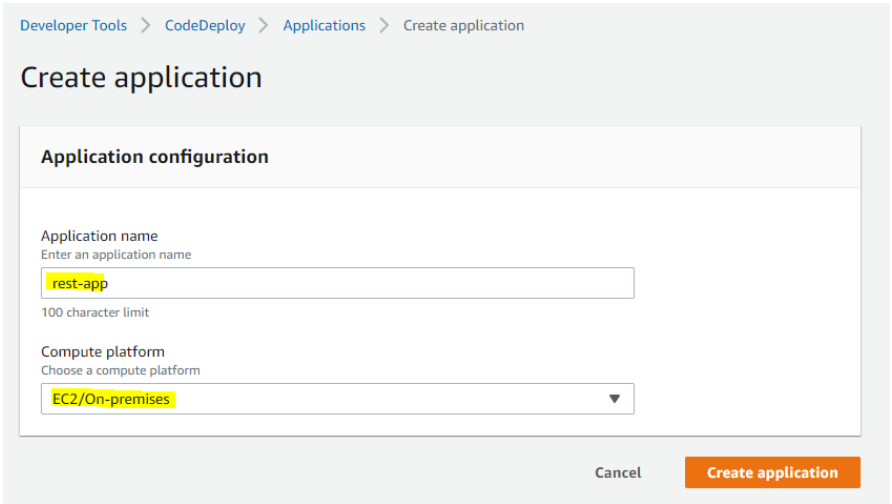
To SSH or Putty into the instance, a key-pair can be created like below,

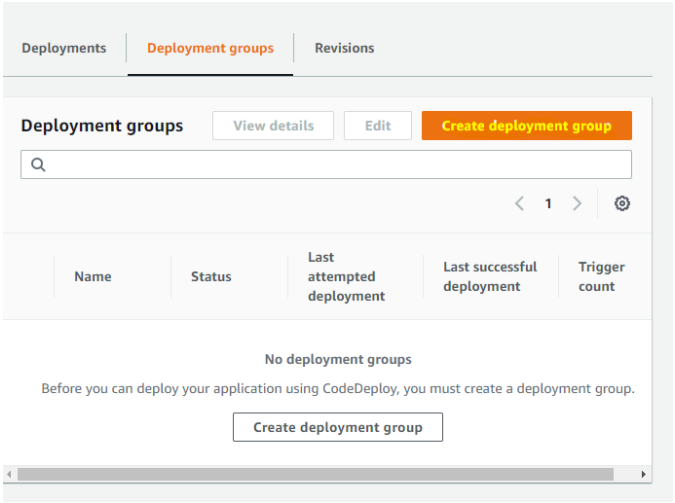


Create CodeDeploy files and scripts

First we setup the CodeDeploy Application AWS CodeDeploy Service Applications






Deployments
Deployment groups
Revisions
Deployment groups
View details
Edit
Create deployment group
1
Name
Status
Last attempted deployment
Last successful deployment
Trigger count
No deployment groups
Before you can deploy your application using CodeDeploy, you must create a deployment group.
Create deployment group

Deployment group name

Enter a deployment group name

dev-env

100 character limit

Service role

Enter a service role

Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

arn:aws:iam::481252253919:role/restapps-CodeDeploy-Service-role

Deployment type

Choose how to deploy your application

- (•) In-place
  Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update
- ( ) Blue/green
  Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

Environment configuration

Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment

- [ ] Amazon EC2 Auto Scaling groups
- [x] Amazon EC2 instances
  1 unique matched instance. Click here for details

You can add up to three groups of tags for EC2 instances to this deployment group.
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 - *optional*	
AppName	Dev	Remove tag

Add tag

+ Add tag group

- [ ] On-premises instances

Matching instances

1 unique matched instance. Click here for details

Install AWS CodeDeploy Agent

- ○ Never
- ○ Only once
- ◉ Now and schedule updates

Basic scheduler | Cron expression

14 Days

Deployment settings

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 is deployed and the success or failure conditions for a deployment.

CodeDeployDefault.AllAtOnce or Create deployment configuration

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.

☐ Enable load balancing

▼ Advanced - optional

Triggers

Delete trigger | Create trigger

	Name	Events	Type
○	restapp-deployment-trigger	DeploymentStart, DeploymentSuccess, DeploymentFailure, DeploymentStop, DeploymentReady, DeploymentRollback	AWS SNS

Alarms

Delete alarm | Add alarm

Name

No Amazon CloudWatch alarms have been created for this deployment group

☐ Ignore alarm configuration
Skips the step of checking Amazon CloudWatch alarms during the deployment process

☐ Continue deployment even if alarm status is unavailable
Permits deployment to run when alarm data cannot be retrieved from Amazon Cloudwatch

Rollbacks

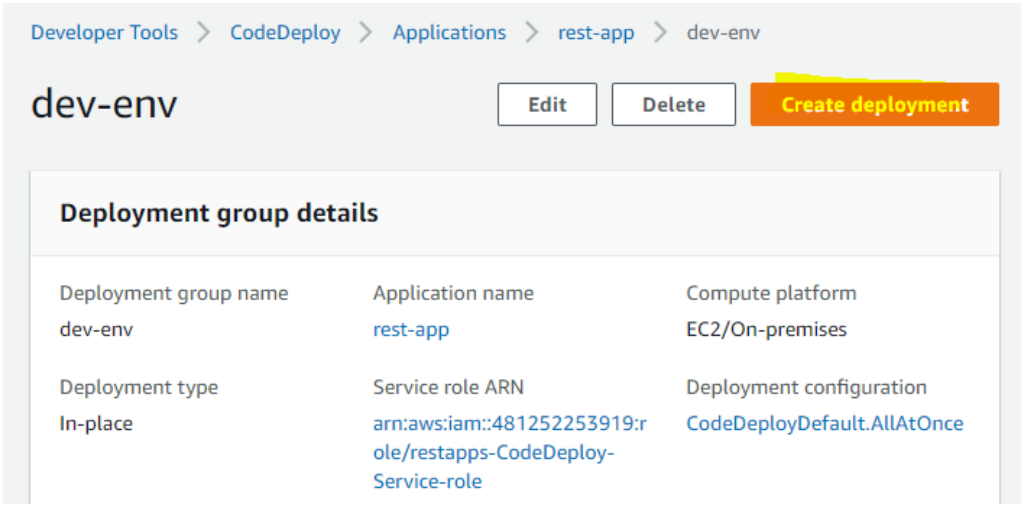
Enable deployment rollbacks for this deployment group

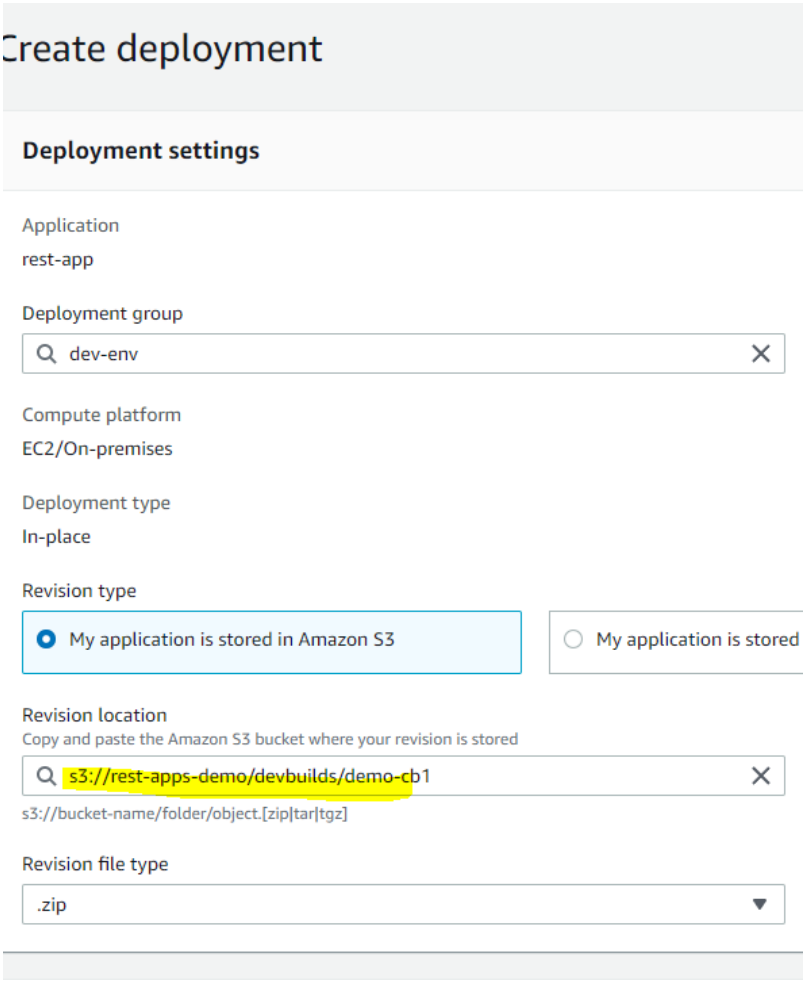
☐ Roll back when a deployment fails

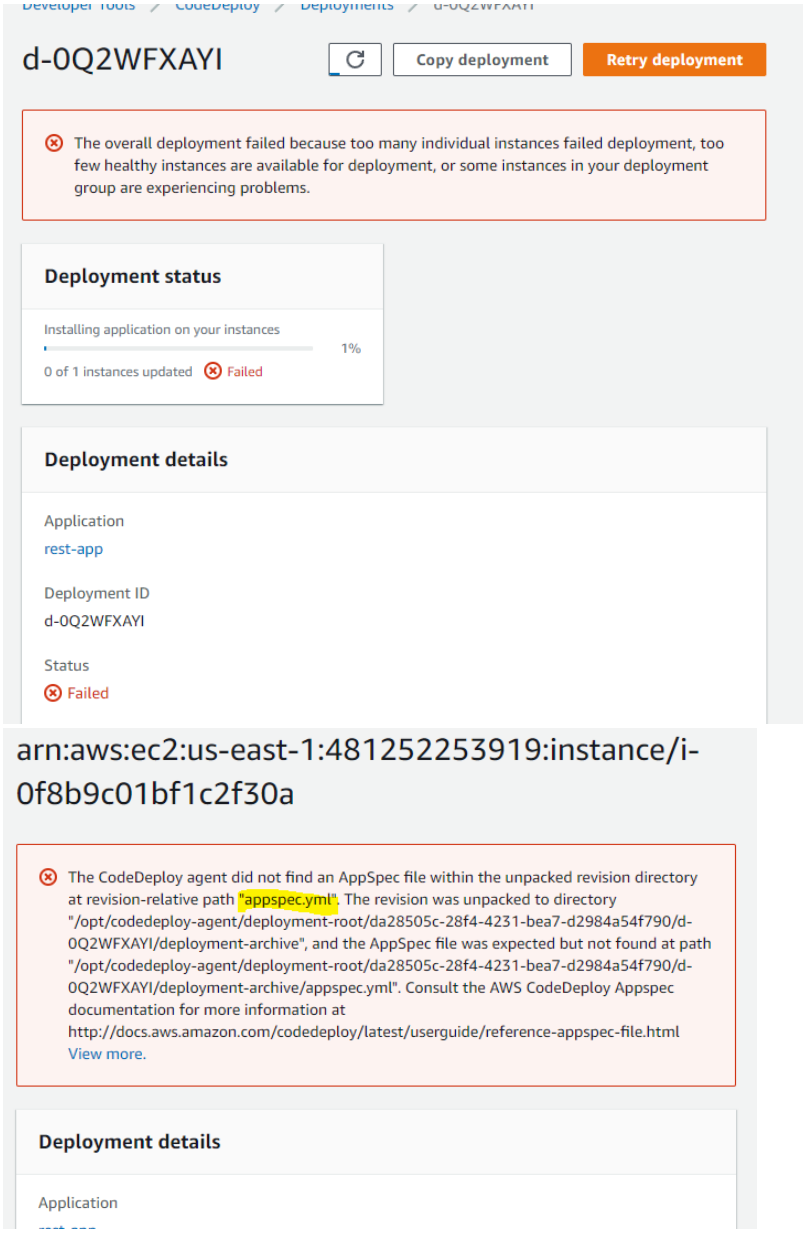
☐ Roll back when alarm thresholds are met

☑ Disable rollbacks

Create Deployment







Step 1: Create appspec.yml – which execute the scripts while deploying

Create **appspec.yml**, and add it in the project folder

```
version: 0.0
os: linux
files:
  - source: target/demo.war
    destination: /usr/share/tomcat/webapps/
hooks:
  BeforeInstall:
    - location: scripts/before_install.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/after_install.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

Step 2: Create a `script` folder in the project directory, add before_install script, start-up script, shutdown script in the Script folder

before_install Script

```
#!/bin/bash
rm -rf /usr/share/tomcat/webapps/demo*
```

after_install Script

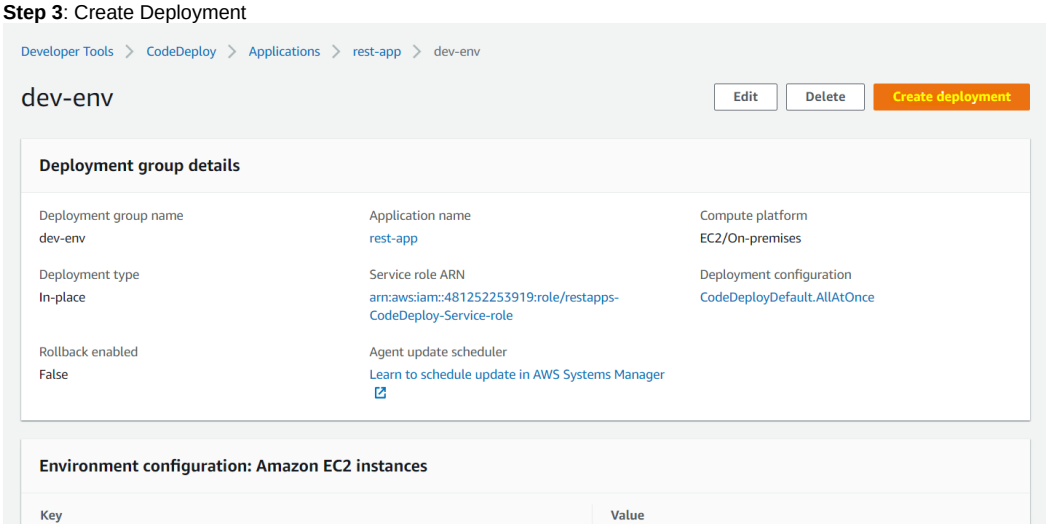
start_server Script

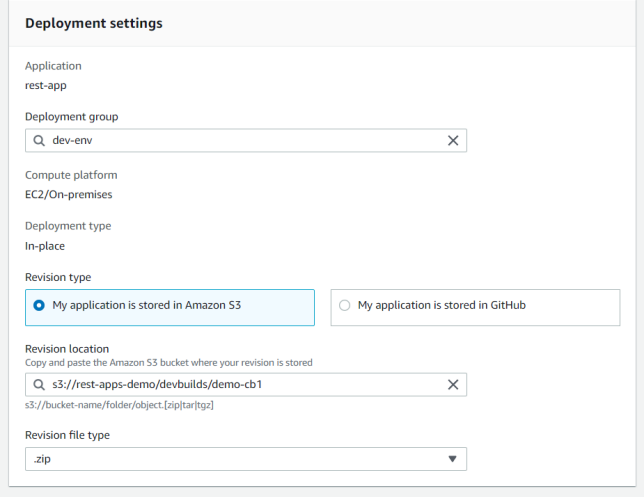
```
#!/bin/bash
sudo systemctl start tomcat
```

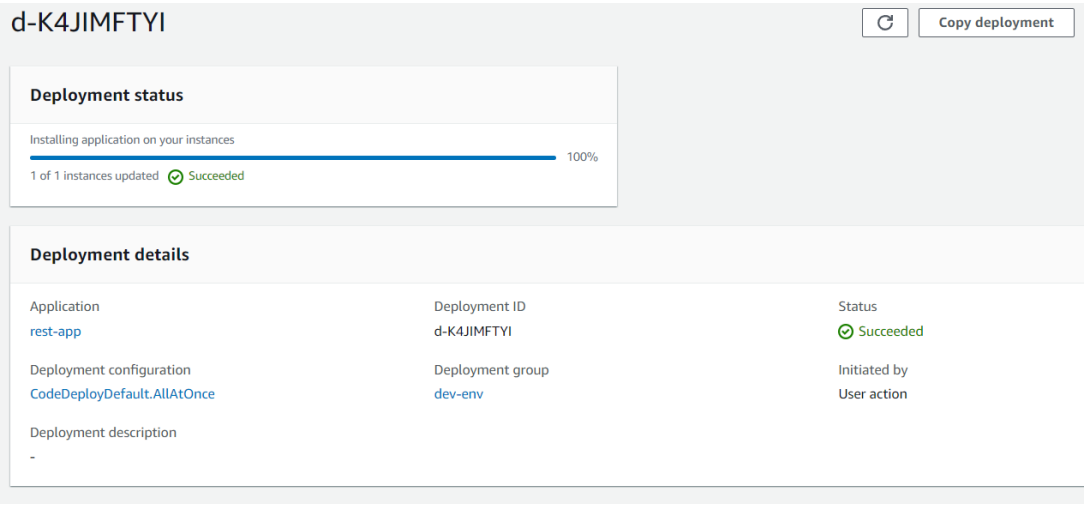
stop_server Script

```
#!/bin/bash
isExistApp=`pgrep java`
if [[ -n  $isExistApp ]]; then
  sudo systemctl stop tomcat
fi
```

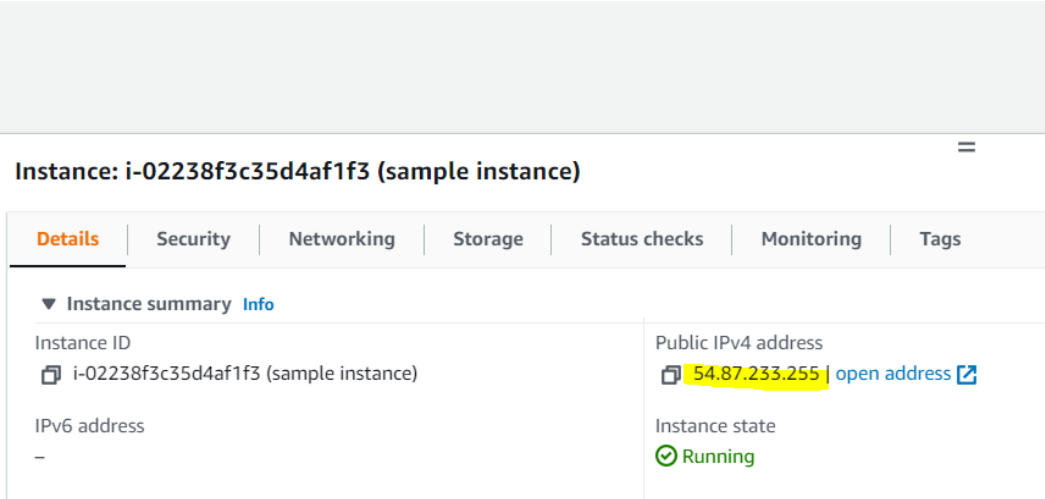
Step 3: Create Deployment







Step 4: App can be accessed on *{public ip of ec2 instance} : {tomcat-portnumber} / {projectname} / {api}*



Hi from Project - v5