😊 Deployment Through Code Deploy

AWS CodeDeploy is a fully managed deployment service that automates the process of deploying applications to various compute services like Amazon EC2, AWS Lambda, or on-premises servers. Its primary goal is to make it easier to update and release software securely and reliably, while minimizing downtime during deployments.

Key Features of CodeDeploy:

1. **Supports Various Compute Platforms**:
    - **EC2/On-Premises**: Deploy applications to EC2 instances, on-premises servers, or hybrid environments.
    - **Lambda**: Automate deployment of updated AWS Lambda functions.
    - **ECS (Containers)**: Deploy updates to Amazon ECS services running on clusters.
2. **Deployment Strategies**:
    - **In-place Deployment**: Updates existing instances with the new application version.
    - **Blue/Green Deployment**: Switches traffic between old and new application versions, enabling rollback with minimal disruption.
3. **Customizable Hooks**:
    - CodeDeploy supports lifecycle event hooks, allowing you to run custom scripts during various phases of deployment (e.g., BeforeInstall, AfterInstall, ApplicationStart).
4. **Rollback**:
    - Automatically or manually roll back to a previous version if deployment issues are detected.
5. **Monitoring and Logging**:
    - Tracks the progress of deployments and integrates with Amazon CloudWatch for monitoring.
6. **Platform Agnostic**:
    - Works with applications written in any language and supports any type of application architecture.

Common Use Cases:

1. **Application Updates**: Deploy new versions of web apps, APIs, or back-end services.
2. **Configuration Changes**: Apply updates to configuration files or system settings.

3. **Continuous Deployment**: Integrate with CI/CD pipelines for automated software release processes.
4. **Hybrid Deployments**: Manage deployments across a combination of cloud and on-premises environments.

Components of CodeDeploy:

1. **Application**: The container for the resources you want to deploy (e.g., Lambda function, EC2 instances).
2. **Deployment Group**: Defines the targets (e.g., EC2 instances, on-premises servers) and configurations for the deployment.
3. **AppSpec File**:
    - The configuration file (appspec.yml or appspec.json) that defines the deployment actions and lifecycle hooks.
4. **Deployment Configuration**:
    - Specifies the deployment strategy (e.g., AllAtOnce, HalfAtATime, OneAtATime).

Workflow Example:

1. Prepare your application and specify deployment settings in an **AppSpec file**.
2. Upload the application code and AppSpec file to an S3 bucket or a Git-based repository (e.g., CodeCommit).
3. Create a **deployment group** with your target servers or Lambda functions.
4. Initiate a deployment in CodeDeploy.
5. CodeDeploy orchestrates the deployment based on your configuration, applying updates, running hooks, and monitoring the process.

Benefits:

- **Automation**: Reduces manual intervention in deployment processes.
- **Flexibility**: Supports various environments and deployment types.
- **Reliability**: Reduces downtime and ensures consistent updates.
- **Scalability**: Handles deployments across hundreds or thousands of instances.

If you're integrating AWS services into your workflow, CodeDeploy is a powerful tool to standardize and automate deployment processes.

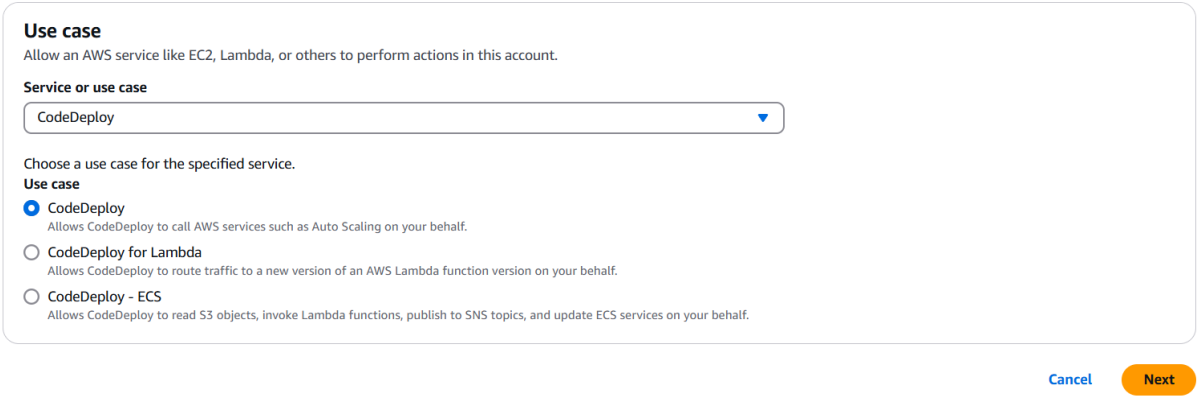
Code Deploy is a very extensive service. It provides a lot of features. We explored it at a high-level overview by just fetching the data from S3 and deploying it to the EC2 instance. You can also configure the environment; you can start the application after deployment and so on.

😄 To begin with the Lab:

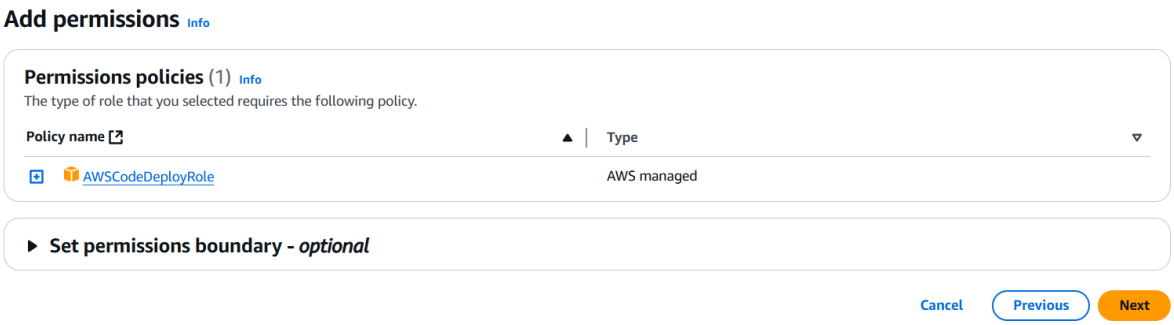
1. Now to work with this service there 5 practical steps or the prerequisites to work with this service.

   a. **Create an IAM role for Code Deploy with S3ReadOnly Access**
   b. **Create IAM Role for EC2 with S3 ReadOnlyAccess**
   c. **Launch EC2 instance with Appropriate Role**
   d. **Install Code Deploy Agent in EC2**
   e. **Configure Code Deploy Service**

2. So, we are going to follow the practical steps to move forward with this lab. Let's start with the first step which is to create the IAM role for Code Deploy service.
3. Navigate to IAM go to roles and click on Create role. In the use case choose Code deploy service and click on next.



4. It has a pre-build permission choose this and click on Next.



5. Give it a name and click on Create role.

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

codeDeployRole

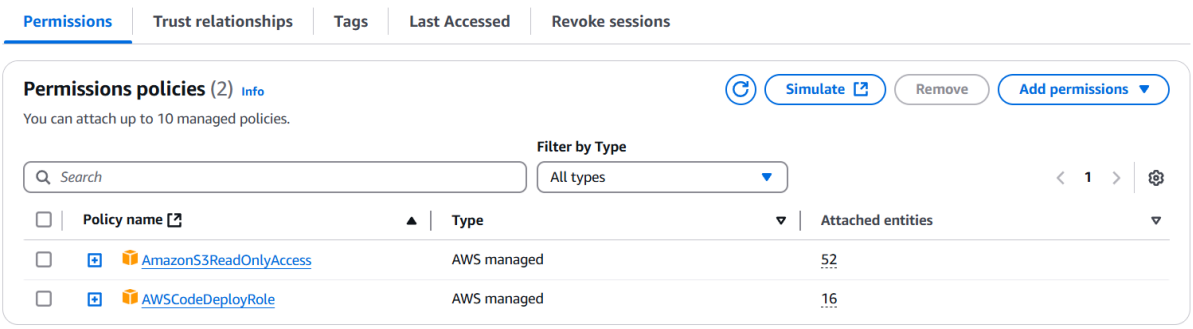
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description
Add a short explanation for this role.

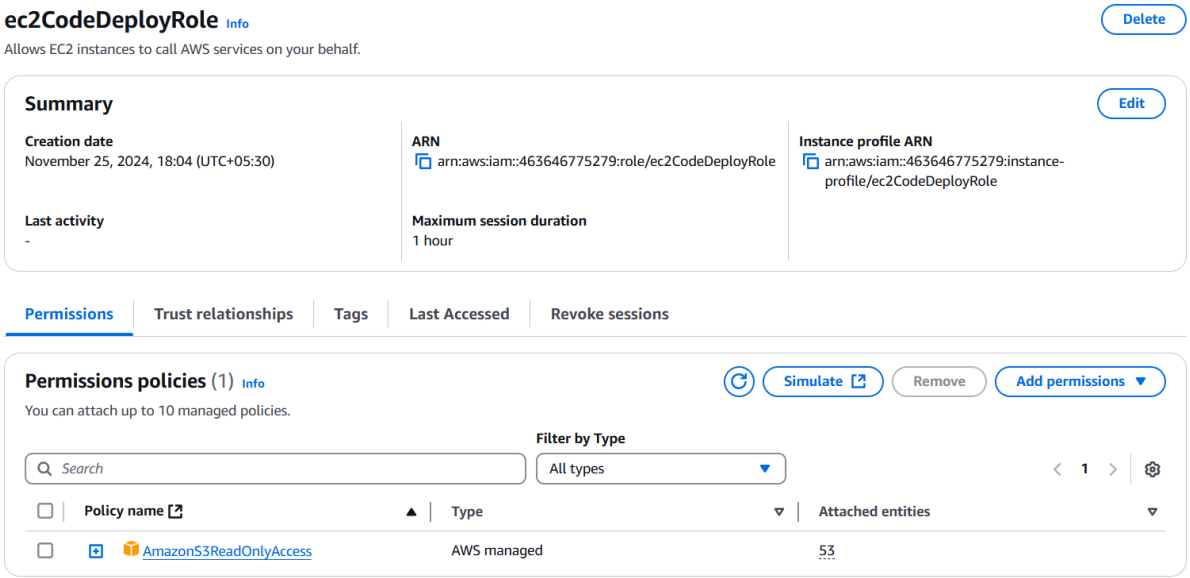
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[{}]!#$%^*();"'`

6. Open your role and then add permission for S3 read-only access as shown below.

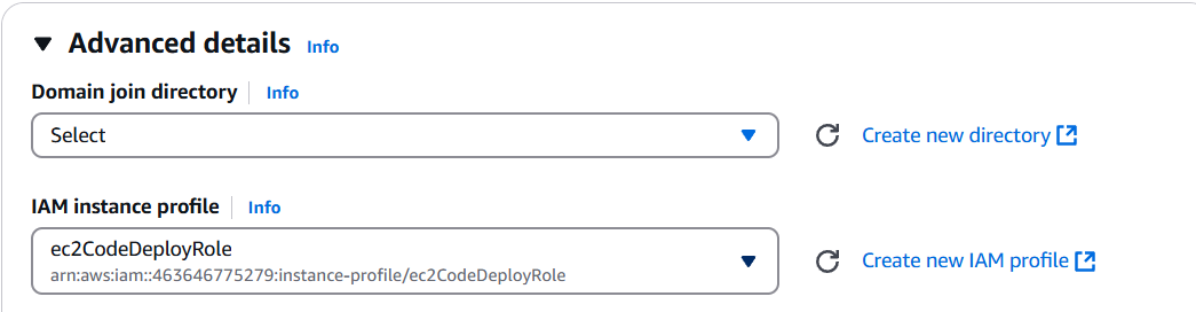


7. For Step 2 we need to create an IAM role for the EC2 instance. Again, click on Create role. Choose EC2 as your use case and click on next. Then you need to attach the S3 read-only permission and create your role.



8. Our Steps 1 and 2 are completed now we need to create an EC2 instance with the appropriate permission.

9. We are going to create our EC2 instance with the default permission and OS just give it a name, create a key pair, and expand the advanced details tab. Here you need to give the IAM role that we created and then create our EC2 instance.



10. Once our EC2 is created we will now install the code deploy agent on it. For that first, we need to connect our instance. Then we need to follow this link below to install the agent.

    https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-linux.html

11. You can also use the commands mentioned below to install the code deploy agent.

    sudo yum install ruby

    sudo yum install wget

    cd /home/ec2-user

    wget https://aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com/latest/install

    chmod +x ./install

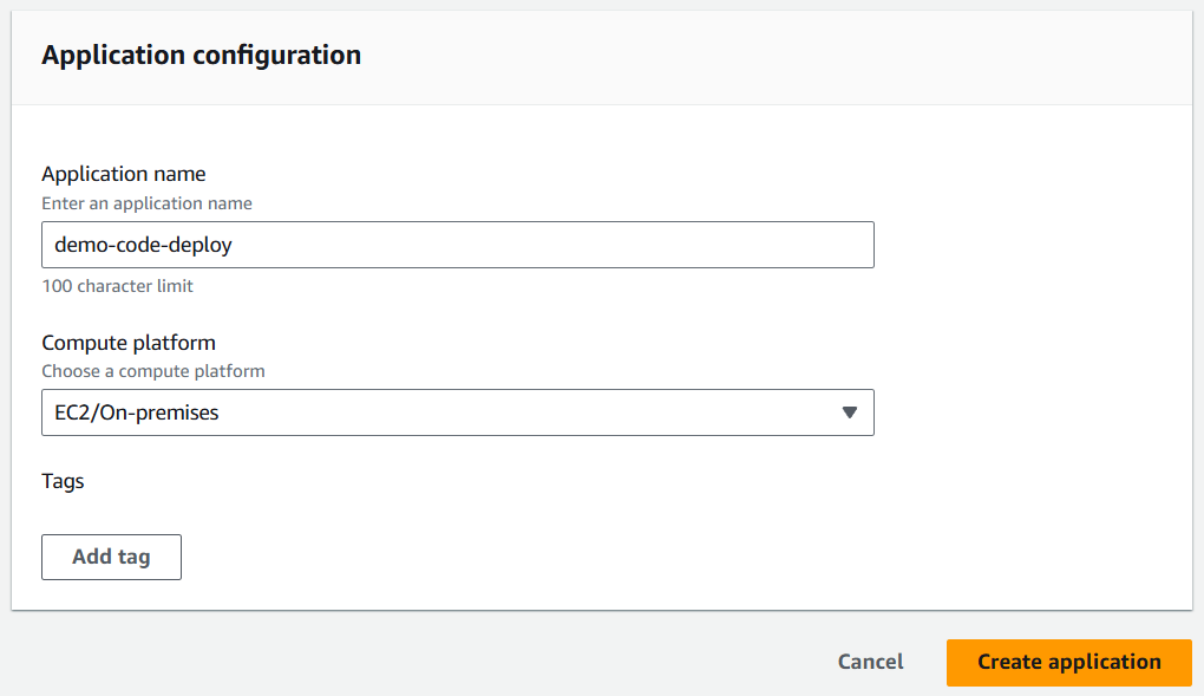
    sudo ./install auto

    systemctl status codedeploy-agent

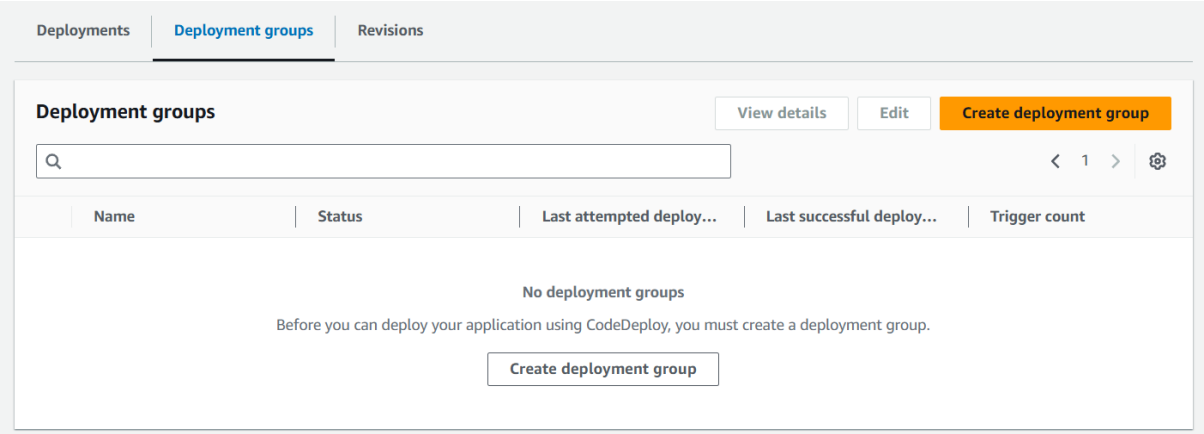
```
[ec2-user@ip-172-31-5-236 ~]$ systemctl status codedeploy-agent
● codedeploy-agent.service - AWS CodeDeploy Host Agent
     Loaded: loaded (/usr/lib/systemd/system/codedeploy-agent.service; enabled; preset: disabled)
     Active: active (running) since Mon 2024-11-25 13:06:17 UTC; 2min 1s ago
   Main PID: 26550 (ruby)
      Tasks: 3 (limit: 1111)
     Memory: 65.8M
        CPU: 974ms
     CGroup: /system.slice/codedeploy-agent.service
             ├─26550 "codedeploy-agent: master 26550"
             └─26552 "codedeploy-agent: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller of master 26550"

Nov 25 13:06:16 ip-172-31-5-236.ap-south-1.compute.internal systemd[1]: Starting codedeploy-agent.service - AWS CodeDeploy Host Agent...
Nov 25 13:06:17 ip-172-31-5-236.ap-south-1.compute.internal systemd[1]: Started codedeploy-agent.service - AWS CodeDeploy Host Agent.
[ec2-user@ip-172-31-5-236 ~]$
```

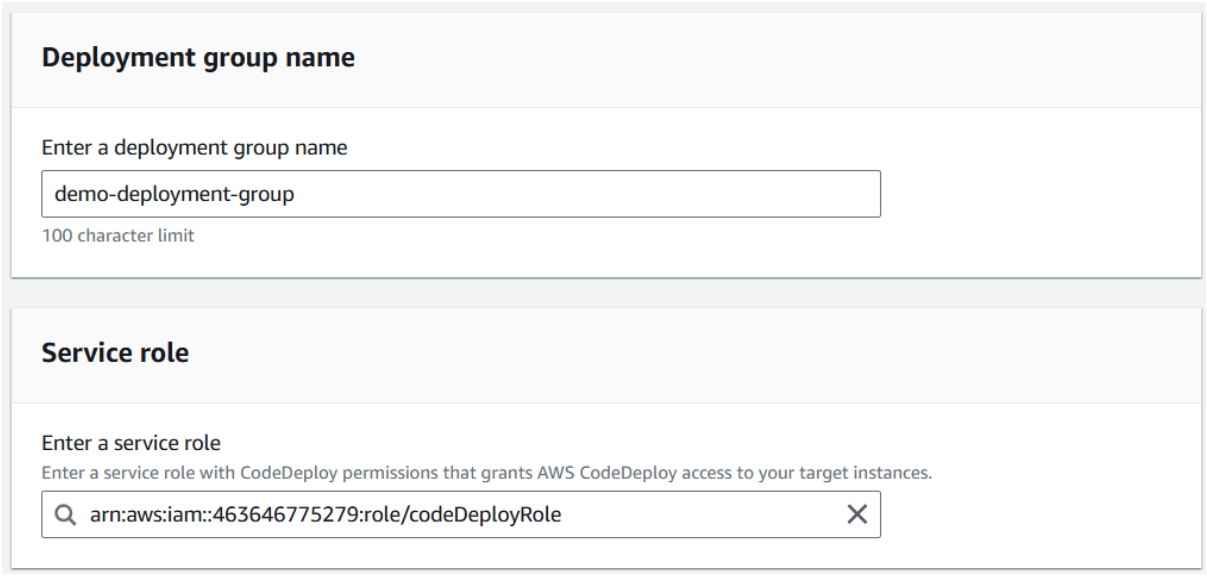
12. Now search for Code Deploy and click on create Application.
13. Here we need to give our application a name and choose our compute platform as EC2. Click on Create application.



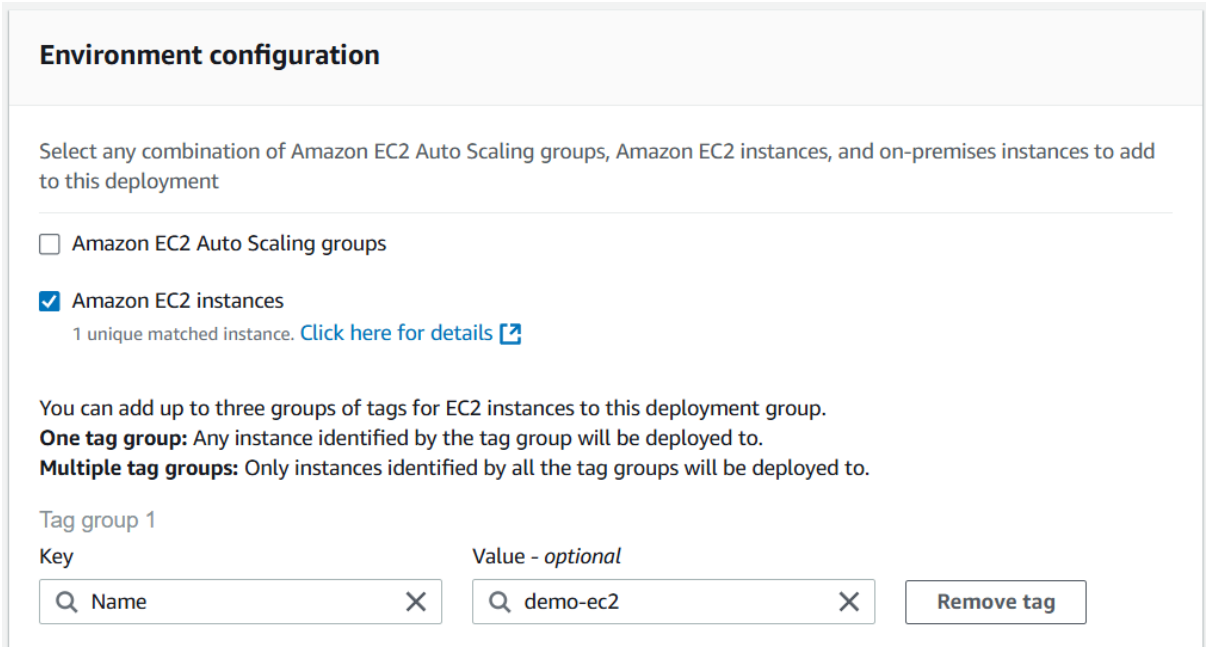
14. From our application we need to create a deployment group. So, click on it.



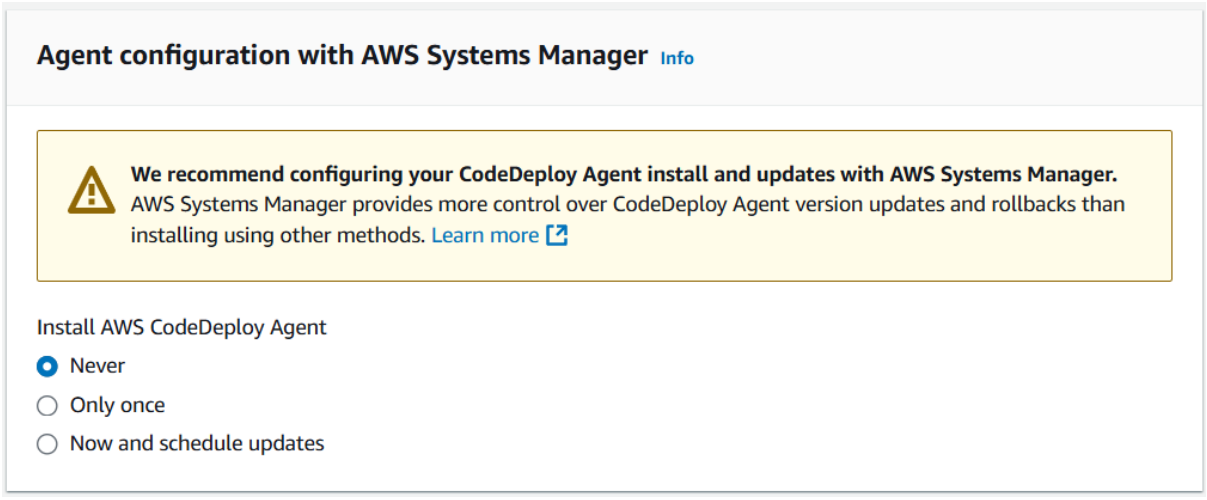
15. Give it name and in the service role choose the IAM role which we created in the step 1.

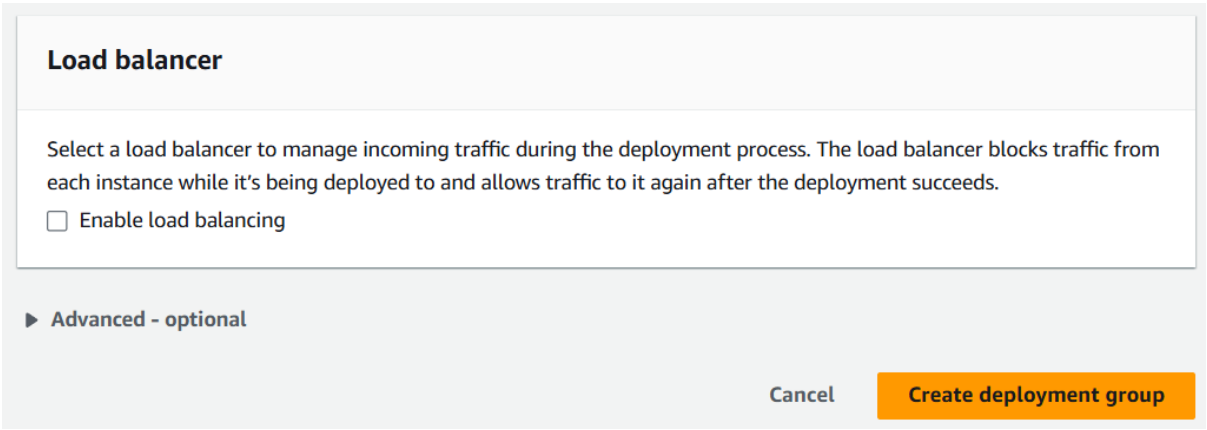


16. For the environment configuration choose Amazon EC2 instances and from the tag group choose the name of your EC2 instance.

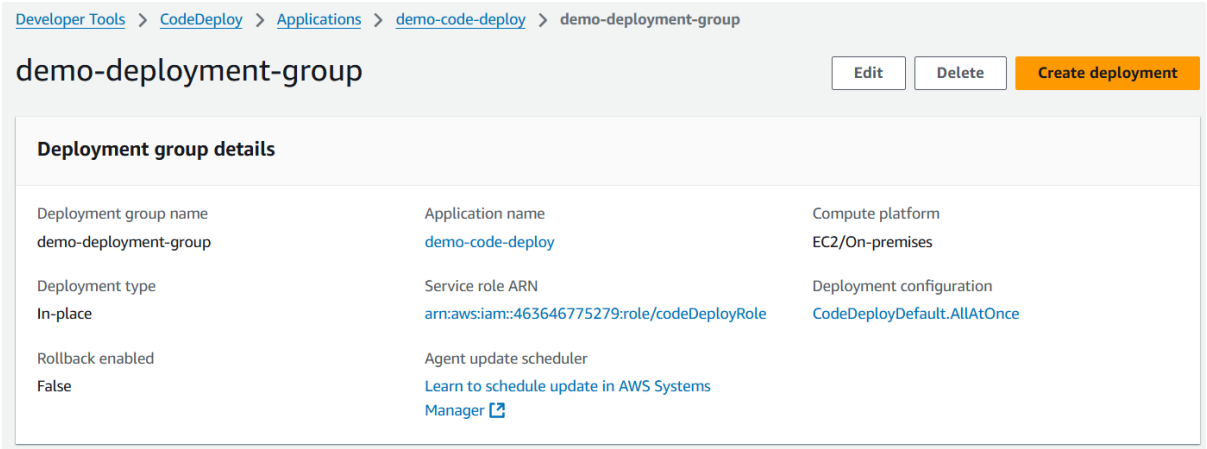


17. In the end for code deploy agent installation choose never. Also, disable the load balancer and click on Create deployment group.

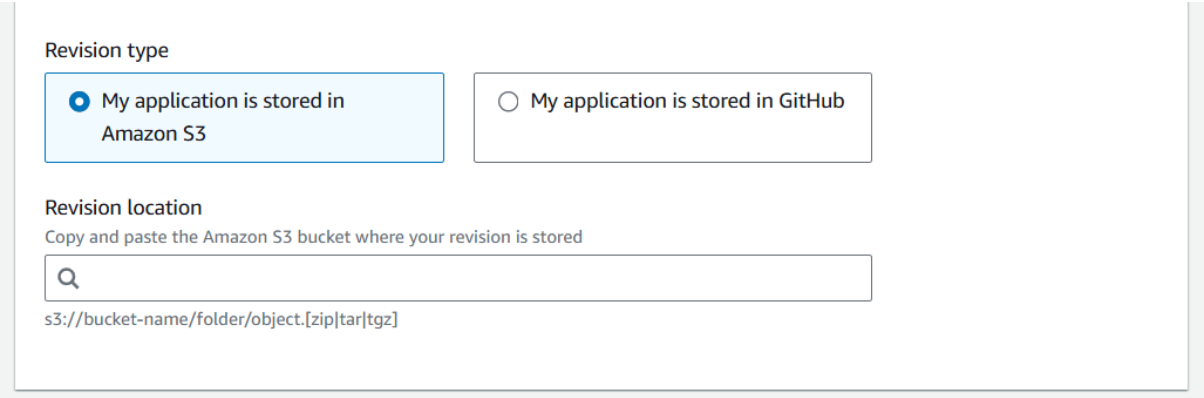




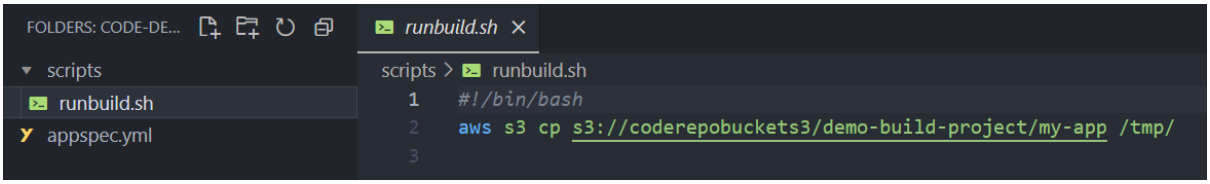
18. Our deployment group is created now we need to create the deployment.



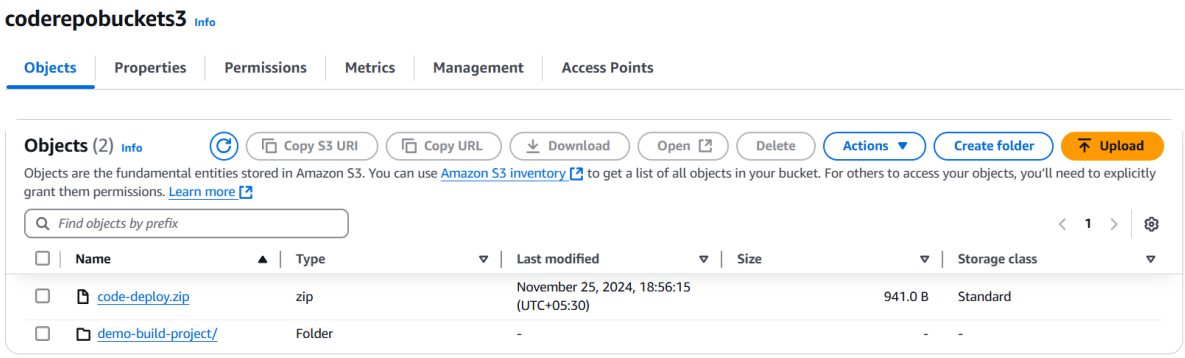
19. Now the important part here is to give the location of our application which is stored in S3 bucket.



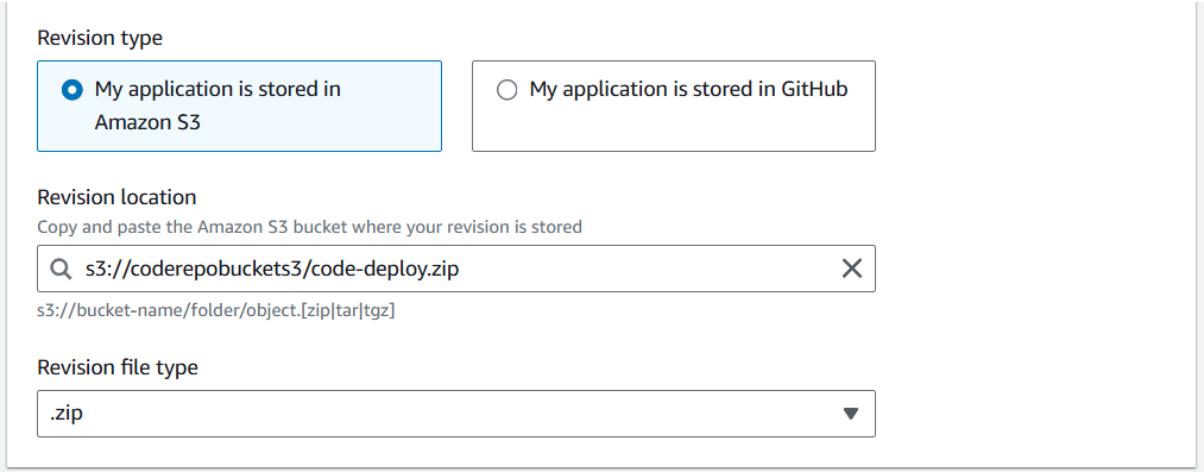
20. For that I have created an appspec.yml file and a runbuild.sh file. So, you have to open the runbuild.sh file and change the bucket name with your bucket name that has the binary of my-app which was created when we built our project using code build.



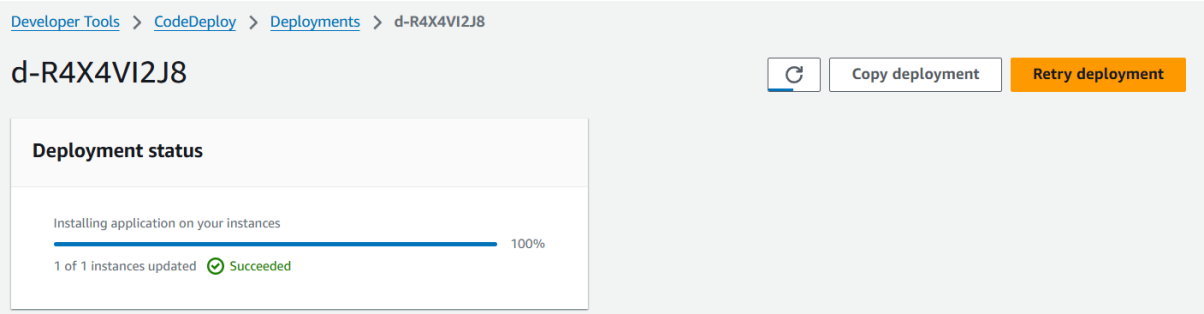
21. After that zip the content and you will also get this zip file to use from the GitHub with this document.
22. So, I have reused the bucket that has my build project and in the same bucket, I have uploaded the zip file.



23. Then copy the S3 URI of the zip file and paste it in the create deployment page. You will also notice that it has detected that it is a zip file.



24. Click on Create Deployment and in no time your deployment will succeed.



25. In the end connect your instance and go to the tmp directory and here you will find your my-app binary.