Deploying Software to Amazon EC2 with AWS CodeDeploy

**SPL-TF-200-DODSCD-1 - Version 1.0.1**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab overview**

In this lab, you learn and practice the concepts and configuration of AWS CodeDeploy to install a web application package on Amazon Compute Cloud (Amazon EC2) instances in both developing/testing and production environments. The lab covers different topics like deployment options, configuration parameters, and monitoring of the AWS CodeDeploy service.

OBJECTIVES

By the end of this lab, you should be able to do the following:

* Configure Amazon Simple Notification Service (Amazon SNS) to provide deployment notifications.
* Configure CodeDeploy to deploy different versions of the application code to developing/testing and production environments.
* Configure CodeDeploy to deploy application code to EC2 instances using in-place and blue/green deployments.
* Monitor and troubleshoot CodeDeploy deployments issues.
* Explore different CodeDeployment features.

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab, student should have a basic knowledge of:

* Navigating through the AWS Management Console.
* Software Development Lifecycle (SDLC).
* AWS Cloud9.

ICON KEY

* **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
* **Note:** A hint, tip, or important guidance.
* **Learn more:** Where to find more information.
* **Task complete:** A conclusion or summary point in the lab.
* **Command:** A command that you must run.
* **Expected output:** A sample output that you can use to verify the output of a command or edited file.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING:** **Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

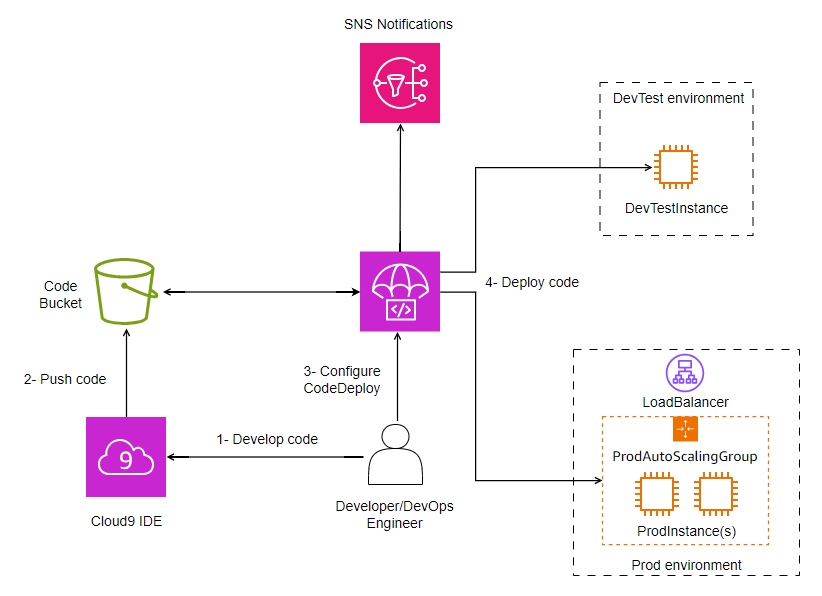
In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

LAB ENVIRONMENT

As a software developer/DevOps engineer, you have been assigned the task to deploy a web application code to an Amazon EC2 instances using AWS CodeDeploy.

The following diagram shows the basic architecture of the lab environment:



*Image description: The preceding figure depicts the lab environment. A software developer/DevOps engineer uses Cloud9 IDE to develop and push the code to an Amazon Simple Storage Service (Amazon S3) bucket. The software developer/DevOps engineer also uses AWS CodeDeploy to deploy the code to the DevTest environment represented by an EC2 instance and also to the Prod environment represented by an EC2 auto-scaling group behind an application load balancer.*

SERVICES USED IN THIS LAB

**AWS CodeDeploy**

AWS CodeDeploy is a fully managed deployment service that automates software deployments to various compute services, such as Amazon Elastic Compute Cloud (EC2), Amazon Elastic Container Service (ECS), AWS Lambda, and your on-premises servers. Use CodeDeploy to automate software deployments, eliminating the need for error-prone manual operations.

**Amazon EC2**

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to handle compute-heavy tasks, such as monthly or yearly processes, or spikes in website traffic. When usage decreases, you can reduce capacity (scale down) again.

**Amazon Simple Storage Service (Amazon S3)**

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance. It stores and protect any amount of data for a range of use cases, such as data lakes, websites, cloud-native applications, backups, archive, machine learning, and analytics.

**AWS Cloud9**

AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more, so you don’t need to install files or configure your development machine to start new projects.

**Amazon Simple Notification Service (Amazon SNS)**

Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers). Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel. Clients can subscribe to the SNS topic and receive published messages using a supported endpoint type, such as Amazon Data Firehose, Amazon SQS, AWS Lambda, HTTP, email, mobile push notifications, and mobile text messages (SMS).

AWS SERVICES NOT USED IN THIS LAB

AWS service capabilities used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

**Task 1: Subscribe to SNS topic**

In this task, you subscribe to an Amazon SNS topic. The SNS topic is used to alert the subscribed DevOps engineers about CodeDeploy deployment notifications. The SNS topic is already created when the lab is provisioned.

1. At the top of the AWS Management Console, in the search bar, search for and choose

SNS

.

1. In the left navigation pane, choose **Topics**.

**Note:** You may need to expand the left navigation pane by choosing the menu icon  that is located on the left side of the console.

1. From the list of **Topics**, choose the link for the **CodeDeployTopic**.
2. Choose the **Create subscription** .
3. In the **Create subscription** page, configure the following:

* For **Protocol**, select **Email**.
* For **Endpoint**, enter a valid email address that you can access.

**Note:** In your personal AWS environment, this might be an email alias for all the DevOps engineers or another notification protocol used for notifications alerts. Individuals receive an email and have to confirm their subscription prior to receiving future notifications from the topic.

1. Choose the **Create subscription** .

A banner message displays a message indicating that subscription to CodeDeployTopic is created successfully.

1. Open the inbox of the email address you entered for the subscription.
2. Locate a recent message from **AWS Notifications**[**no-reply@sns.amazonaws.com**](mailto:no-reply@sns.amazonaws.com).

**Note:** It may take up to 5 minutes to receive the email depending on your email server.

1. Select the Confirm subscription hyperlink contained in the email.

The browser opens a new page confirming the subscription.

1. Close the Amazon SNS topic subscription confirmation page.

**Task complete:** You have successfully subscribed to an Amazon SNS topic. With a subscription, Amazon SNS pushes new messages from this topic to the subscribed email address.

**Task 2: Configure CodeDeploy to push the initial version of the application code to the Developing/Testing environment**

In this task, you deploy the first version of the web application to the DevTest instance using CodeDeploy and verify the deployment. You also troubleshoot and remediate deployment issues in the AppSpec file, and you explore the use of CodeDeploy environment variables.

First, you review the web application code.

TASK 2.1: REVIEW THE WEB APPLICATION CODE AND CODEDEPLOY APPSPEC FILE

Open the Cloud9 environment.

1. Copy the **Cloud9Url** value that is listed to the left of these instructions. Paste the value into a new browser tab, and then press **Enter**.

**Consider:** Familiarize yourself with the AWS Cloud9 IDE layout.

* The top-left section contains the various menu items.
* The left-hand side is the *Files* pane where you see your file structure. The top level of your file tree is named *Cloud9 - /home/ec2-user/environment*.
* Below the menu section is where you modify your files.
  + Feel free to close the **Welcome** tab.
* The bottom section is the terminal session. This is where you run commands throughout the lab.

1. In the **Files** pane, expand (if not already expanded) the **Cloud9 - /home/ec2-user/environment** directory and then expand the **web-app-v1** directory.

The *web-app-v1* directory is where version 1 of the web application code is stored.

1. Locate and choose the **appspec.yml** file in the **web-app-v1** directory.

The file opens in a new tab in the top-half of your Cloud9 environment.

The AppSpec file is used by CodeDeploy to manage a deployment. If your application uses the EC2/On-Premises compute platform, the AppSpec file must be a YAML-formatted file named **appspec.yml** and it must be placed in the root of the directory structure of an application’s source code. Otherwise, deployments fail. It is used by CodeDeploy to determine:

* What it should install onto your instances from your application revision in Amazon S3 or GitHub.
* Which lifecycle event hooks to run in response to deployment lifecycle events.

After you have a completed AppSpec file, you bundle it, along with the content to deploy, into an archive file (zip, tar, or compressed tar).

**Learn more:** Refer to [CodeDeploy AppSpec file reference](https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file.html) in the **Additional resources** section for more information.

Notice that the *appspec* file specifies the target environment (Linux in this example) and the source and destination location of the code to be installed (*/app* for the source and */var/www/html/* for the destination)

Notice that the *appspec.yml* file also references other script files that are ran by CodeDeploy during the different lifecycle event hooks of the deployment. These script files are stored in the *scripts* directory. Feel free to expand the *scripts* directory and explore these scripts in your Cloud9 environment.

Here is a summary of which lifecycle event hooks are taking place in this deployment:

* *ApplicationStop*: The script is stopping the *httpd* service if it is already running or available on the instance.
* *AfterInstall*: This lifecycle event takes place after the application code is installed. You can use this event to install any dependencies and/or configure any required parameters. For the web application example used in this lab, *httpd* is installed as a dependency required to run the web application. The *modify\_index* script is used to configure some parameters and is discussed later in the lab.
* *ApplicationStart*: This lifecycle event is used to start any required services. The script used in this lab is simply starting the *httpd* service.

**Note:** The lifecycle event hooks run in a specific order regardless of the order they are listed in the *appspec.yml* file.

**Learn more:** Refer to [AppSpec ‘hooks’ section](https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html) in the **Additional resources** section for more information.

Finally, review the web application code.

1. In the **Files** pane, expand the **app** directory.

The *app* directory has a simple web application code structured as follows:

* *index.html* file: This is your main HTML file.
* *css* directory: This is where web page style settings are stored.
* *static* directory: This is where the web page static assets are stored.

**Note:** The web application used in this lab is a simple web page. The purpose of the lab is to explore and practice deploying software using CodeDeploy.

1. Locate and choose the **index.html** file in the **app** directory.

The file opens in a new tab in the top-half of your Cloud9 environment.

Notice that the *body* section of the file has some environment variables (STAGE\_VALUE, APP\_VERSION, and INSTANCE\_ID). The values of these variables are retrieved and substituted when the *modify\_index* script runs in the *AfterInstall* lifecycle hook of the *appspec.yml* file. You explore these variables when deploying the code later in this task.

Next, you create the code package and upload it to an S3 bucket to prepare for the deployment.

TASK 2.2: CREATE THE CODE PACKAGE AND UPLOAD TO AN S3 BUCKET

When creating a deployment using CodeDeploy, you need to package your code and ensure that the *appspec* file is at the root of the code package. Then, you need to upload or push your code package to a repository that is compatible with target deployment platform. For EC2 instances, the repository can be an S3 bucket or GitHub.

For the purpose of this lab, you package your code as *.zip* and use an S3 bucket as your repository. The S3 bucket is already created during lab provisioning. The name of the bucket is the value of the *CodeBucketName* listed to the left of these instructions.

Now, create a *.zip* package of your code including the *appspec.yml* file and *scripts* directory.

1. **Command:** To create the zip archive for version 1 of the code, in the **Cloud9** environment terminal session, from the **~/environment** prompt, enter the following command:

cd web-app-v1 ; zip -r ../webappv1.zip . \* ; cd ..

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\* EXAMPLE OUTPUT \*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

adding: app/ (stored 0%)

adding: app/css/ (stored 0%)

adding: app/css/index.css (deflated 46%)

adding: app/index.html (deflated 43%)

adding: app/static/ (stored 0%)

adding: app/static/powered-by-aws.png (stored 0%)

adding: appspec.yml (deflated 60%)

adding: scripts/ (stored 0%)

adding: scripts/install\_dependencies (deflated 12%)

adding: scripts/modify\_index (deflated 45%)

adding: scripts/start\_server (stored 0%)

adding: scripts/stop\_server (deflated 17%)

The archive *webappv1.zip* archive is created in the *~/environment* directory.

Before uploading the *.zip* archive to the S3 bucket, set the S3 bucket name as an environment variable so you can reuse the variable for all your queries from the AWS CLI.

1. **Command:** To set the S3 bucket name as an environment variable named *code\_bucket*, run the following command on the **~/environment** prompt:

code\_bucket=$(aws s3api list-buckets --query 'Buckets[?starts\_with(Name, `codebucket-`)].[Name]' --output text)

**Expected output:**

None, unless there is an error.

1. **Command:** To verify that the environment variable is correctly set, run the following command on the **~/environment** prompt:

echo $code\_bucket

**Expected output:** Your bucket name is different from what is seen below.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\* EXAMPLE OUTPUT \*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

codebucket-us-west-2-4087364191214953

1. Verify that the output of the above command matches the value of the **CodeBucketName** listed to the left of these instructions.

**Note:** If your Cloud9 environment is stopped, the environment variable setting is lost and need to be re-configured.

1. **Command:** To upload the **webappv1.zip** archive to the S3 bucket, run the following command on the **~/environment** prompt:

aws s3 cp webappv1.zip s3://$code\_bucket/webappv1.zip

**Expected output:** Your bucket name is different from what is seen below.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\* EXAMPLE OUTPUT \*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

upload: ./webappv1.zip to s3://codebucket-us-west-2-4087364191214953/webappv1.zip

1. Close any files tabs that are open in the top-half section of the Cloud9 environment by choosing **x** in the file tab.

Your web application code package is uploaded and ready for deployment.

TASK 2.3: CONFIGURE CODEDEPLOY TO DEPLOY THE WEB APPLICATION TO THE DEVTEST ENVIRONMENT

Next, you configure CodeDeploy to deploy your web application. CodeDeploy configuration includes the following components:

* *Application*: This is a place holder to configure an application name and a compute platform that you intend to deploy the application to.
* *Deployment group*: This is a logical group mapped under the *application* and includes configuration items like the service role, deployment type, environment configuration, deployment settings, and other deployment parameters.
* *Deployment*: This is a specific deployment instance that is mapped under the *deployment group* and points to the code package that is being deployed. You can also set some deployment parameters.

You examine the different configuration options above as you configure them in this task.

First, start by creating an application.

1. At the top of the AWS Management Console, in the search bar, search for and choose

CodeDeploy

.

1. In the left navigation pane, in the **Deploy . CodeDeploy** section, choose **Applications**.
2. Choose **Create application** .
3. In the **Create application** page, configure the following:

* For **Application name**, enter

webapp

.

* For **Compute platform**, select **EC2/On-Premises**.

**Consider:** Notice that you can also use CodeDeploy to deploy code to AWS Lambda and Amazon Elastic Container Service (Amazon ECS). For the purpose of this lab, you use EC2 instances as the deployment platform.

1. Choose **Create application** .

A banner message displays a message indicating that the application is created.

**Learn more:** Refer to [Create an application with CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/applications-create.html) in the **Additional resources** section for more information.

Now, create a deployment group under the *webapp* application. This deployment group is for deploying the code to the *DevTest* environment. The *DevTest* environment is represented by a single EC2 instance.

1. In the **webapp** page, choose **Create deployment group** .
2. In the **Create deployment group**, configure the following:

* For **Deployment group name**, enter

DevTest

* For **Service role**, choose the **CodeDeployRole**

**Consider:** The CodeDeploy service role is assumed by the CodeDeploy service. It must have all the required AWS Identity and Access Management (IAM) permissions required for the service to deploy the code. The *CodeDeployRole* was pre-created during the lab provisioning.

**Learn more:** Refer to [Create a service role for CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/getting-started-create-service-role.html) in the **Additional resources** section for more information.

* For **Deployment type**, select **In-place**.

**Consider:** In *in-place* mode, CodeDeploy updates the instances in the deployment group with the latest application revisions. During a deployment, each instance is be briefly taken offline for its update. This is a suitable option for developing and testing environments but you may want to take a different approach for production. You examine the *blue/green* option in a later task in the lab.

* For **Environment configuration**, select **Amazon EC2 instances**.
* For **Key** under **Tag group 1**, choose **Name**.
* For **Value-optional** under **Tag group 1**, choose **DevTestInstance**.

**Consider:**

* You can identify the target EC2 instance(s) using and instance tag(s) or an auto scaling group of EC2 instances. For this task, you are deploying the code to a single EC2 instance in the DevTest environment which is not a member of an auto scaling group.
* The *key/value* pair tag of *Name/DevTestInstance* is used as the tag to identify the *DevTestInstance*.
* For CodeDeploy to deploy code to EC2 instances, the instances must have the CodeDeploy agent installed and running. All the instances in this lab are already provisioned with the CodeDeploy agent during lab provisioning.

**Learn more:** Refer to [Install the CodeDeploy agent](https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install.html) in the **Additional resources** section for more information.

* For **Loadbalancer**, deselect **Enable load balancing**.

**Consider:** This option is not required for the *DevTest* instance as it is not sitting behind a load balancer.

1. Expand the **Advanced - optional** section.
2. For **Triggers**, choose **Create trigger** .
3. In the **Create deployment trigger** dialogue window, configure the following:

* For **Trigger name**, enter

Deployment failure

.

* For **Events**, select **Deployment fails**.
* For **Amazon SNS topics**, choose the topic ARN that has **CodeDeployTopic** in its name.

**Consider:** You can create multiple triggers and select multiple events for each trigger. For the purpose of this lab, you only configure a trigger for deployment failures.

1. Choose **Create trigger** .
2. Choose **Cancel** to close the **Create deployment trigger** dialogue window
3. Choose **Create deployment group** .

A banner message displays a message indicating that the deployment group is created.

In few minutes, you should receive an email indicating that AWS CodeDeploy notification was successfully set.

**Learn more:** Refer to [Working with deployment groups in CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-groups.html) in the **Additional resources** section for more information.

Now, create a deployment to point to the code package to be deployed.

1. In the **DevTest** page, choose **Create deployment** .

Note that some of the deployment are already set as they are inherited from the deployment group configuration.

1. In the **Create deployment** page, configure the following:

* For **Revision type**, select **My application is stored in Amazon S3**.
* For **Revision location**, copy and paste the value of **WebappV1S3Uri** listed to the left of these instructions. This is the S3 URI of the package you uploaded to the S3 bucket.
* For **Revision file type**, select **.zip**.

**Consider:** Check the other options of file types that you can use to deploy your code.

* For **Deployment description**, type

1.0

.

* For **Additional deployment behavior settings**, leave as default. This section allows to configure how to handle failures and duplicate contents options.
* For **Deployment group overrides**, leave as default. This is an option to override the deployment group settings of how many instances to deploy at a time. Since there is only one instance. The default setting of *AllAtOnce* is fine.
* For **Rollback configuration overrides**, leave as default. This option allows you to configure if and when rollback to the previous version is invoked.

1. Choose **Create deployment** .

A banner message displays a message indicating that the deployment is created.

CodeDeploy assigns a unique ID to each deployment (something like d-5F27KVTG4) which you can see at the top of the deployment page.

**Learn more:** Refer to [Working with deployments in CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments.html) in the **Additional resources** section for more information.

After few seconds, the *Deployment status* in the the deployment page indicates that the deployment *Failed*.

After few minutes, you should receive an email on the email account you subscribe to the SNS topics indicating that a CodeDeploy deployment had FAILED with the details of the deployment.

You need to troubleshoot and remediate the deployment issue.

TASK 2.4: TROUBLESHOOT AND REMEDIATE DEPLOYMENT ISSUE

For each deployment, CodeDeploy allows you to monitor the deployment status and lifecycle events using the AWS Management Console which can help you troubleshooting deployment issues. The CodeDeploy agent also writes detailed deployment logs into the EC2 instance.

Start by examining the lifecycle events in the console.

1. In the deployment page where the deployment failed, scroll down to **Deployment lifecycle events**. Notice that it displays the instance ID with a **Failed** status.
2. Choose the **View events** link under the **Events** column.

The browser displays a new page which shows a message stating that the *script/modify\_index* script does not exist at specified location.

1. Scroll down to the section that displays the lifecycle events. Notice the order of lifecycle events execution and the status of each event. Notice that the first event that failed is the *AfterInstall* event. It indicates that a *ScriptMissing* message under the error code column of that event.
2. Choose the **ScriptMissing** link for the **AfterInstall** event.

The browser displays the **Event Logs** page which indicates that *scripts/install\_dependencies* script was successfully completed while the next script in the event *script/modify\_index* was not. This is because the script was not find.

1. Choose the browser tab of your Cloud9 environment.
2. In the **Files** pane, locate and choose the **appspec.yml** file in the **web-app-v1** directory.

The file opens in a new tab in the top-half of your Cloud9 environment.

Notice that the script location in the second step in the *AfterInstall* hook is pointing to *script/modify\_index* while the directory storing the scripts is named *scripts* rather than *script*.

1. Modify the location of the second step in the *AfterInstall* hook to the following:

- location: scripts/modify\_index

After modifying the above, your *AfterInstall* should like the snippet below:

AfterInstall:

- location: scripts/install\_dependencies

timeout: 300

runas: root

- location: scripts/modify\_index

timeout: 300

runas: root

Do not modify any other line in the file.

1. In the Cloud9 menu, choose **File**, then choose **Save** to save your changes.

Now you need to create a new zip archive and upload it to the S3 bucket.

1. **Command:** To create the zip archive after modifying the code, in the **Cloud9** environment terminal session, from the **~/environment** prompt, enter the following command:

cd web-app-v1 ; zip -r ../webappv1.zip . \* ; cd ..

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\* EXAMPLE OUTPUT \*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

adding: app/ (stored 0%)

adding: app/css/ (stored 0%)

adding: app/css/index.css (deflated 46%)

adding: app/index.html (deflated 43%)

adding: app/static/ (stored 0%)

adding: app/static/powered-by-aws.png (stored 0%)

adding: appspec.yml (deflated 60%)

adding: scripts/ (stored 0%)

adding: scripts/install\_dependencies (deflated 12%)

adding: scripts/modify\_index (deflated 45%)

adding: scripts/start\_server (stored 0%)

adding: scripts/stop\_server (deflated 17%)

The archive *webappv1.zip* archive is created in the *~/environment* directory.

1. **Command:** To reupload the new **webappv1.zip** archive to the S3 bucket, run the following command on the **~/environment** prompt:

aws s3 cp webappv1.zip s3://$code\_bucket/webappv1.zip

**Expected output:** Your bucket name is different from what is seen below.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\* EXAMPLE OUTPUT \*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

upload: ./webappv1.zip to s3://codebucket-us-west-2-4087364191214953/webappv1.zip

**Learn more:** Refer to [Troubleshooting CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting.html) in the **Additional resources** section for more information.

Now, re-attempt the deployment.

1. Return to the browser tab of CodeDeploy.
2. In the left navigation pane, in the **Deploy . CodeDeploy** section, under **Deployments**, choose **Deployment**.

The browser opens the page of the failed deployment.

1. Choose **Retry deployment** .

A banner message displays a message indicating that the deployment retry is restarted.

After few seconds, the deployment status shows that the deployment had succeeded.

1. To verify the web application deployment on the **DevTestInstance**, copy the **DevTestInstanceUrl** value listed to the left of these instructions, paste it in a new browser tab, and then press **Enter**.

The web application is displayed with the *Hello World* heading and a welcome message to the webapp.

Next, examine how the *AfterInstall* hook scripts are leveraging the use of the CodeDeploy environment variables.

TASK 2.5: EXAMINE LEVERAGING THE USE OF CODEDEPLOY ENVIRONMENT VARIABLES

Examine the web application *Hello World* page. Notice the following:

* The *Deployment stage* has a value of *DevTest* which is the same as the deployment group name.
* The *Application version* has a value of *1.0* which is the same as the deployment description.

As stated earlier, the *index.html* file has some dynamic contents. The *modify\_index* script file runs simple commands to retrieve the some CodeDeploy environment variables from the CodeDeploy agent during the deployment and adds these values to the *index.html* file to appear in the web page. The instance ID is also added in the same way, however, it is retrieved via the EC2 instance metadata service and it is not a CodeDeploy environment variables.

Another use of the CodeDeploy variables is in the *install\_dependencies* script. The script installs the *httpd* service and then sets the *httpd* service to run on port 9090 if the stage is equal to *DevTest* (which is retrieved from the deployment group name). Otherwise, the *httpd* service runs on the standard port 80. This is a common scenario where you can use CodeDeploy environment variables to set a custom port number for the service depending on a the variable value.

1. Feel free to view and examine the **modify\_index** and **install\_dependencies** files which can be found under the **web-app-v1/scripts** folder in your Cloud9 environment.

**Note:** The web page in the lab displays the stage, application version, and instance ID is only used for teaching and demonstration purpose. You should not disclose these values in your public web pages unless there is a valid reason to do so.

**Learn more:** Refer to [Using CodeDeploy Environment Variables](https://aws.amazon.com/blogs/devops/using-codedeploy-environment-variables/) in the **Additional resources** section for more information.

**Task complete:** You have successfully deployed the first version of the web application to the DevTest instance using CodeDeploy and verified the deployment. During the deployment, you also troubleshooted and remediated deployment issues in the AppSpec file, and you explored the use of CodeDeploy environment variables.

**Task 3: Configure CodeDeploy to push the initial version of the application code to the Prod environment**

In this task, you deploy the first version of the web application to the Prod instances using in-place deployment type and verify the deployment.

Your production environment has two auto scaling groups. Each group has two instances. You deploy the first version of the code to the instances in the *ProdBlueASG* auto scaling group. The other auto scaling group is used in a later task when deploying the updated version of the code using blue/green deployment.

Here is a summary of requirements to deploy the code to the Prod instances:

* CodeDeploy application: You use the same application created earlier *webapp*.
* CodeDeploy deployment group: You need to create a new deployment group to push the code to the Prod instances in the auto scaling group.
* CodeDeploy deployment: You need to create a new deployment of the new deployment group.

First, create a new deployment group.

1. Choose the browser tab of the CodeDeploy console.
2. In the left navigation pane, in the **Deploy . CodeDeploy** section, choose **Applications**.
3. In the **Applications** page, choose the link for the **webapp**.
4. In the **webapp** page, choose **Create deployment group** .
5. In the **Create deployment group**, configure the following:

* For **Deployment group name**, enter

Prod-In-Place

* For **Service role**, choose the **CodeDeployRole**
* For **Deployment type**, select **In-place**.
* For **Environment configuration**, select **Amazon EC2 Auto Scaling groups**, and then select **ProdBlueASG** in the drop-down menu.

**Consider:** For this deployment group, since the target instances are members of an auto scaling group, you identify the target instances by selecting their auto scaling group.

* For **Loadbalancer**, select **Enable load balancing**.
* For **Load balancer type**, select **Application Load Balancer or Network Load Balancer**.
* For **Choose target groups**, select **Autoscale-tg**.

**Consider:** This option is required for the *Prod* instances as they are placed behind a load balancer. The load balancer and target group resources are already created during lab provisioning.

1. Expand the **Advanced - optional** section.
2. For **Triggers**, choose **Create trigger** .
3. In the **Create deployment trigger** dialogue window, configure the following:

* For **Trigger name**, enter

Deployment failure

.

* For **Events**, select **Deployment fails**.
* For **Amazon SNS topics**, choose the topic ARN that has **CodeDeployTopic** in its name.

1. Choose **Create trigger** .
2. Choose **Cancel** to close the **Create deployment trigger** dialogue window
3. Choose **Create deployment group** .

A banner message displays a message indicating that the deployment group is created.

In few minutes, you should receive an email indicating that AWS CodeDeploy notification was successfully set.

Now, create a deployment to point to the code package to be deployed.

1. In the **Prod-In-Place** page, choose **Create deployment** .

Note that some of the deployment are already set as they are inherited from the deployment group configuration.

1. In the **Create deployment** page, configure the following:

* For **Revision type**, select **My application is stored in Amazon S3**.
* For **Revision location**, copy and paste the value of **WebappV1S3Uri** listed to the left of these instructions. This is the S3 URI of the package you uploaded to the S3 bucket.
* For **Revision file type**, select **.zip**.
* For **Deployment description**, type

1.0

.

* For **Additional deployment behavior settings**, leave as default.
* For **Deployment group overrides**, leave as default.
* For **Rollback configuration overrides**, leave as default.

1. Choose **Create deployment** .

A banner message displays a message indicating that the deployment is created.

After 5-10 minutes, the deployment status shows that the deployment had succeeded. While waiting for the deployment to be completed, you can view the progress of deployment on each of the two instances in the *Deployment lifecycle events* section of the deployment page.

1. To verify the web application deployment on the **ProdInstance**, copy the **ProdLoadBalancerUrl** value listed to the left of these instructions, paste it in a new browser tab, and then press **Enter**.

The web application is displayed with the *Hello World* heading and a welcome message to the webapp. Notice that the URL does not have a custom port. This means it is running on the default HTTP port 80 as the script did not modify the port. Recall that the port is modified to *9090* only if the deployment group name is *DevTest*.

1. Verify that web page has a *Deployment stage* value of *Prod-In-Place* and *Application version* value of *1.0*.
2. Attempt refreshing the web page multiple times. Notice that the *Instance ID* is interchanging as there are two instances in this auto scaling group behind the load balancer.

**Task complete:** You have successfully deployed the first version of the web application to the Prod instances using using in-place deployment type and verified the deployment.

**Task 4: Deploy an updated version of the application code to the Developing/Testing environment**

The software development team have decided to make some minor updates to the web application and the developer already created the application files and placed them in the *web-app-v2/* directory of your Cloud9 environment. They directory also includes the *appspec.yml* file and *script* directory with all the required deployment scripts.

**Note:** In real-life scenarios, you use a different branch in your source control repository and update the application version. Hence, you do not have two separate folders for the application versions. However, for the purpose of this lab, the code is provided to you in separate folders since the version control repository is out of the scope of this lab.

In this task, you use CodeDeploy to push the updated version of the web application to the *DevTest* instance and verify the deployment.

**Note:**

* The updates made to the application are only related to the web page background color (green instead of blue)
* The *appspec.yml* file is correctly configured and pointing to the correct locations of the script files.

TASK 4.1: CREATE THE UPDATED CODE PACKAGE AND UPLOAD IT TO S3 BUCKET

1. Choose the browser tab of your Cloud9 environment.
2. **Command:** To create the zip archive for version 2 of the code, in the **Cloud9** environment terminal session, from the **~/environment** prompt, enter the following command:

cd web-app-v2 ; zip -r ../webappv2.zip . \* ; cd ..

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\* EXAMPLE OUTPUT \*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

adding: app/ (stored 0%)

adding: app/css/ (stored 0%)

adding: app/css/index.css (deflated 45%)

adding: app/index.html (deflated 43%)

adding: app/static/ (stored 0%)

adding: app/static/powered-by-aws.png (stored 0%)

adding: appspec.yml (deflated 60%)

adding: scripts/ (stored 0%)

adding: scripts/install\_dependencies (deflated 12%)

adding: scripts/modify\_index (deflated 45%)

adding: scripts/start\_server (stored 0%)

adding: scripts/stop\_server (deflated 17%)

The archive *webappv2.zip* archive is created in the *~/environment* directory.

1. **Command:** To upload the **webappv2.zip** archive to the S3 bucket, run the following command on the **~/environment** prompt:

aws s3 cp webappv2.zip s3://$code\_bucket/webappv2.zip

**Expected output:** Your bucket name is different from what is seen below.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\* EXAMPLE OUTPUT \*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

upload: ./webappv2.zip to s3://codebucket-us-west-2-4087364191214953/webappv2.zip

1. Close any files tabs that are open in the top-half section of the Cloud9 environment by choosing **x** in the file tab.

Your updated version of the web application code package is uploaded and ready for deployment.

TASK 4.2: CONFIGURE CODEDEPLOY TO DEPLOY THE UPDATED VERSION OF WEB APPLICATION TO THE DEVTEST ENVIRONMENT

Next, you configure CodeDeploy to deploy your web application. Since you are deploying the new version to the same *DevTest* instance, you can reuse the same existing *DevTest* deployment group. However, you need to configure a new deployment pointing to the new version of the code.

**Note:** In real-life scenario, you can consider creating a new deployment group, reuse the existing group, or modify the existing group depending on the use case of your deployment.

1. Choose the browser tab of the CodeDeploy console.
2. In the left navigation pane, in the **Deploy . CodeDeploy** section, choose **Applications**.
3. In the **Applications** page, choose the link for the **webapp**.
4. In the **webapp** page, choose the link of the **DevTest** deployment group.
5. In the **DevTest** page, choose **Create deployment** .

Note that some of the deployment are already set as they are inherited from the deployment group configuration.

1. In the **Create deployment** page, configure the following:

* For **Revision type**, select **My application is stored in Amazon S3**.
* For **Revision location**, copy and paste the value of **WebappV2S3Uri** listed to the left of these instructions. This is the S3 URI of the package you uploaded to the S3 bucket.
* For **Revision file type**, select **.zip**.
* For **Deployment description**, type

2.0

.

* For **Additional deployment behavior settings**, select **Overwrite the content**.
* For **Deployment group overrides**, leave as default.
* For **Rollback configuration overrides**, deselect **Disable rollbacks** and select **Roll back when a deployment fails**.

1. Choose **Create deployment** .

A banner message displays a message indicating that the deployment is created.

After few seconds, the deployment status shows that the deployment had succeeded. You can view the progress of deployment on the target instance in the *Deployment lifecycle events* section of the deployment page.

1. To verify the web application deployment on the **DevTestInstance**, copy the **DevTestInstanceUrl** value listed to the left of these instructions, paste it in a new browser tab, and then press **Enter**.

The web application is displayed with the *Hello World* heading and a welcome message to the webapp.

1. Verify that web page has a *Deployment stage* value of *DevTest* and *Application version* value of *2.0*.

Notice that the web page background color has changed now to a green color instead of blue in the previous version.

**Task complete:** You have successfully deployed the updated version of the web application to the DevTest instance using CodeDeploy and verified the deployment.

**Task 5: Deploy an updated version of the application code to the production environment using blue/green deployment type**

In this task, you use CodeDeploy to push the updated version of the web application to the *Prod* instances using blue/green deployment type and verify the deployment.

UNDERSTANDING BLUE/GREEN DEPLOYMENT TYPE

In the previous tasks, you used in-place deployment type to deploy the code on the target instances. With in-place type, CodeDeploy takes the target instance offline, deploys the code and run all related lifecycle hooks, then brings the instance back online if the deployment succeeded. In some scenarios, you do not want to have any downtime in your application. In this use case, you can use blue/green deployment. With blue/green deployment, you must have another replica of the target instances that you want to deploy the code to. The existing instances ares known as *original* and considered as the *blue* deployment, while new instances are known as *replacement* and considered as the *green* deployment. During the deployment, CodeDeploy deploys the new code to the replacement (green) instances while keeping the original (blue) instances intact. This means that the web application is up and running during the deployment but using the previous version of the code. Once the deployment succeeds on the green instances, the traffic can be rerouted automatically or manually depending on the deployment settings.

In blue/green deployment type, the replacement instances can be created using one of the following options:

* *Automatically copy Amazon EC2 Auto Scaling group*: When you create the deployment, CodeDeploy provisions the replacement instances by creating an Amazon EC2 Auto Scaling group and deploy the new application revision to it. CodeDeploy creates the Auto Scaling group by copying an existing group you specify in the configuration.
* *Manually provision instances*: The replacements instances are already provisioned. You specify the original instances in the deployment group configuration, then you specify the replacement instances when you create the deployment and CodeDeploy pushes the new code to them. You can define them using tags or as members of an auto scaling group.

For the purpose of this lab, the target replacement instances are already provisioned as members of another auto scaling group named *ProdGreenASG*.

1. Choose the browser tab of CodeDeploy console.
2. In the left navigation pane, in the **Deploy . CodeDeploy** section, choose **Applications**.
3. In the **Applications** page, choose the link for the **webapp**.

Since this deployment is using blue/green deployment type, you need to create a new deployment group.

1. In the **webapp** page, choose **Create deployment group** .
2. In the **Create deployment group**, configure the following:

* For **Deployment group name**, enter

Prod-Blue-Green

* For **Service role**, choose the **CodeDeployRole**
* For **Deployment type**, select **Blue/green**.
* For **Environment configuration**, select **Manually provision instances**, then select **Amazon EC2 Auto Scaling groups**, and then select **ProdBlueASG** in the drop-down menu.

**Consider:** In the option above, you specify the instances you want to replace (original instances). Then, you specify the replacement instances when you create the deployment.

1. For the **Deployment settings** section, configure the following:

* For **Traffic rerouting**, select **I will choose whether to reroute traffic**.
* For **Days**, select **0**.
* For **Hours**, select **0**.
* For **Minutes**, select **10**.
* Leave the rest of settings in this section as default.

**Consider:** Even though you set a time for when to reroute traffic and terminate original instances, you can still manually invoke these actions before the timer is due.

1. For the **Load balancer** section, configure the following:

* For **Load balancer type**, select **Application Load Balancer or Network Load Balancer**.
* For **Choose target groups**, select **Autoscale-tg**.

1. Expand the **Advanced - optional** section.
2. For **Triggers**, choose **Create trigger** .
3. In the **Create deployment trigger** dialogue window, configure the following:

* For **Trigger name**, enter

Deployment failure

.

* For **Events**, select **Deployment fails**.
* For **Amazon SNS topics**, choose the topic ARN that has **CodeDeployTopic** in its name.

1. Choose **Create trigger** .
2. Choose **Cancel** to close the **Create deployment trigger** dialogue window
3. Choose **Create deployment group** .

A banner message displays a message indicating that the deployment group is created.

In few minutes, you should receive an email indicating that AWS CodeDeploy notification was successfully set.

Now, create a deployment to point to the code package to be deployed.

1. In the **Prod-Blue-Green** page, choose **Create deployment** .

Note that some of the deployment are already set as they are inherited from the deployment group configuration.

1. In the **Create deployment** page, configure the following:

* For **Revision type**, select **My application is stored in Amazon S3**.
* For **Revision location**, copy and paste the value of **WebappV2S3Uri** listed to the left of these instructions. This is the S3 URI of the package you uploaded to the S3 bucket.
* For **Revision file type**, select **.zip**.
* For **Environment configuration**, select **Amazon EC2 Auto Scaling groups**, and then select **ProdGreenASG** from the drop-down menu.

**Consider:** In the option above, you specify the replacement instances where the new version of code is deployed. These instances must be already provisioned and have the CodeDeploy agent installed and running before you create the deployment.

* For **Deployment description**, type

2.0

.

* For **Additional deployment behavior settings**, leave as default.
* For **Deployment group overrides**, leave as default.
* For **Rollback configuration overrides**, select **Disable rollbacks**.

1. Choose **Create deployment** .

A banner message displays a message indicating that the deployment is created.

After 1-2 minutes, the deployment status shows that *Step 2 Installing application on replacement instances* had succeeded. This indicates that the new version is successfully installed on the replacements instances. However, the traffic is not yet rerouted to the replacement instances.

In real-life scenarios, you would use this phase to perform any kind of checks and tests on the replacement environment to verify the deployment. If the replacement instances are behind a separate load balancer, you can even reroute the traffic and test by accessing the web page. You can then start redistributing production traffic from the original to replacement instances using DNS weighted traffic rule. This all depends on your requirements in terms of downtime and the cost of the extra replacement resources.

**Learn more:** Refer to [Deployment methods](https://docs.aws.amazon.com/whitepapers/latest/practicing-continuous-integration-continuous-delivery/deployment-methods.html) in the **Additional resources** section for more information.

You can wait for the 10 minutes timer you set for the traffic to be rerouted automatically or invoke the reroute manually. For the lab purpose. You manually invoke traffic reroute.

1. In the deployment page, choose **Reroute traffic** .

This invokes the *Rerouting traffic to replacement instances* step.

You can monitor the traffic rerouting in the **Traffic shifting progress** section of the deployment page. It displays the traffic rerouting between the original and replacement instances.

1. Wait until the status of *Step 3 Rerouting traffic to replacement instances* had succeeded. This may take 5-10 minutes.
2. To verify the web application deployment on the **ProdInstance**, copy the **ProdLoadBalancerUrl** value listed to the left of these instructions, paste it in a new browser tab, and then press **Enter**.

The new web page should have a *Deployment stage* value of *Prod-Blue-Green* and *Application version* value of *2.0*. It also has a green color background.

1. Attempt refreshing the web page multiple times. Notice that the you can only see the new version of the code and the *Instance ID* is interchanging as there are two instances in the replacement auto scaling group behind the load balancer.

At this point, you complete your testing and verification then decide to terminate the original instances. This can be done manually or automatically by waiting for the one hour timer which you set in the deployment configuration.

For the purpose of this lab, you terminate the original instances manually.

1. In the deployment page, choose **Terminate** .

This invokes the *Terminating original instances* step.

1. Wait until the status of *Step 4 Terminating original instances* had succeeded. This indicates that your original instances are terminated.

**Task complete:** You have successfully used CodeDeploy to push the updated version of the web application to the production instances using blue/green deployment type and verified the deployment.

**Conclusion**

You have successfully done the following:

* Configured Amazon Simple Notification Service (Amazon SNS) to provide deployment notifications.
* Configured CodeDeploy to deploy different versions of the application code to developing/testing and production environments.
* Configured CodeDeploy to deploy application code to EC2 instances using in-place and blue/green deployments.
* Monitored and troubleshooted CodeDeploy deployments issues.
* Explored different CodeDeployment features.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional Resources**

* [CodeDeploy AppSpec file reference](https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file.html)
* [AppSpec ‘hooks’ section](https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html)
* [Create an application with CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/applications-create.html)
* [Install the CodeDeploy agent](https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install.html)
* [Working with deployment groups in CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-groups.html)
* [Working with deployments in CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments.html)
* [Troubleshooting CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting.html)
* [Using CodeDeploy Environment Variables](https://aws.amazon.com/blogs/devops/using-codedeploy-environment-variables/)
* [Create a service role for CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/getting-started-create-service-role.html)
* [Deployment methods](https://docs.aws.amazon.com/whitepapers/latest/practicing-continuous-integration-continuous-delivery/deployment-methods.html)

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).