Set up a Continuous Deployment Pipeline

using AWS CodePipeline

In this tutorial, you will learn how to create an automated software release pipeline that deploys a live sample app. You will create the pipeline using AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change. You will use your GitHub account, an Amazon Simple Storage Service (S3) bucket, or an AWS CodeCommit repository as the source location for the sample app's code. You will also use AWS Elastic Beanstalk as the deployment target for the sample app. Your completed pipeline will be able to detect changes made to the source repository containing the sample app and then automatically update your live sample app.

Continuous deployment allows you to deploy revisions to a production environment automatically without explicit approval from a developer, making the entire software release process automated.

Everything done in this tutorial is free tier eligible.

Step 1: Create a deployment environment

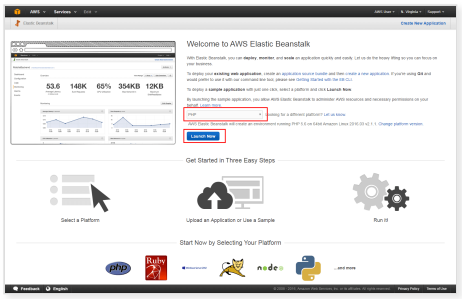
Your continuous deployment pipeline will need a target environment containing virtual servers, or Amazon EC2 instances, where it will deploy sample code. You will prepare this environment before creating the pipeline.

a. To simplify the process of setting up and configuring EC2 instances for this tutorial, you will spin up a sample environment using AWS Elastic Beanstalk. Elastic Beanstalk lets you easily host web applications without needing to launch, configure, or operate virtual servers on your own. It automatically provisions and operates the infrastructure (e.g. virtual servers, load balancers, etc.) and provides the application stack (e.g. OS, language and framework, web and application server, etc.) for you.

- Click here to open the AWS Elastic Beanstalk console.

---

b. Choose *PHP* from the drop-down menu and then click **Launch Now**.

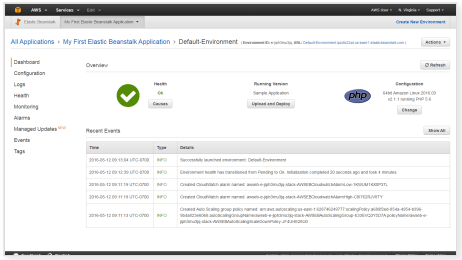


(click to zoom)

Note: If you have created an Elastic Beanstalk application before, click: **Create New Application** on the upper-right corner. Name your application and create a new *web server environment*. Select *PHP* as your platform and *Single Instance* as your environment type. If you are planning to remote login to your instances, select a key pair. Otherwise, leave default values for the remaining options and create the environment for your continuous deployment pipeline.

---

c. Elastic Beanstalk will begin creating a sample environment for you to deploy your application to. It will create an Amazon EC2 instance, a security group, an Auto Scaling group, an Amazon S3 bucket, Amazon CloudWatch alarms, and a domain name for your application.



Step 2: Get a copy of the sample code

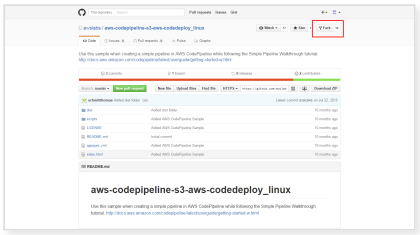
In this step, you will retrieve a copy of the sample app's code and choose a source to host the code. The pipeline takes code from the source and then performs actions on it.

You can use one of three options as your source: a GitHub repository, an Amazon S3 bucket, or an AWS CodeCommit repository. Select your preference and follow the steps below:

GitHub

a. If you would like to use your GitHub account:

- Visit our GitHub repository containing the sample code at https://github.com/awslabs/aws-codepipeline-s3-aws-codedeploy_linux.
- Fork a copy of the repository to your own GitHub account by clicking the **Fork** button in the upper-right corner.



(click to zoom)

Amazon S3

AWS CodeCommit

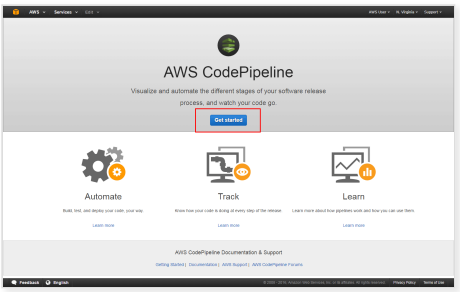
Step 3: Create your pipeline

deployment environment.

---

a. Click here to open the AWS CodePipeline console.

- On the Welcome page, click **Create pipeline**.
- If this is your first time using AWS CodePipeline, an introductory page appears instead of Welcome. Click **Get Started**.



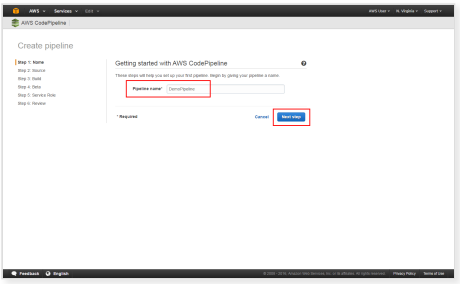
(click to zoom)

---

b. On the *Step 1: Name* page:

- **Pipeline name:** enter the name for your pipeline, *DemoPipeline.*
- Click **Next step**.

Note: After you create a pipeline, you cannot change its name.



(click to zoom)

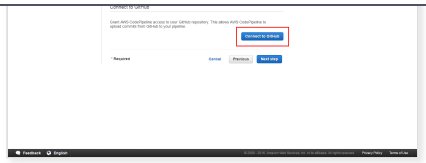
---

c. On the *Step 2: Source* page, select the location of the source you selected and follow the steps below:

GitHub

Source Provider: GitHub

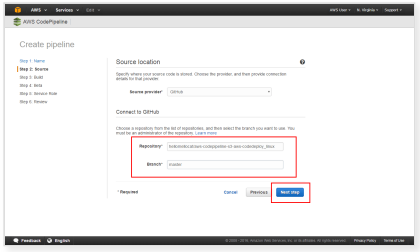
- A new browser window will open to connect you to GitHub. If prompted to sign in, provide your GitHub credentials.
- You will be asked to authorize application access to your account. Choose **Authorize application**.



(click to zoom)

Specify the repository and branch:

- **Repository:** In the drop-down list, choose the GitHub repository you want to use as the source location for your pipeline. Click the forked repository in your GitHub account containing the sample code called *aws-codepipeline-s3-aws-codedeploy_linux*.
- **Branch:** In the drop-down list, choose the branch you want to use, *master.*
- Click **Next step**.



(click to zoom)

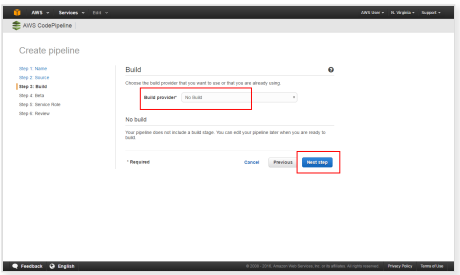
Amazon S3

AWS CodeCommit

---

d. A true continuous deployment pipeline requires a build stage, where code is compiled and unit tested. CodePipeline lets you plug your preferred build provider into your pipeline. However, in this tutorial you will skip the build stage.

- In *Step 3: Build* page, choose **No Build.**

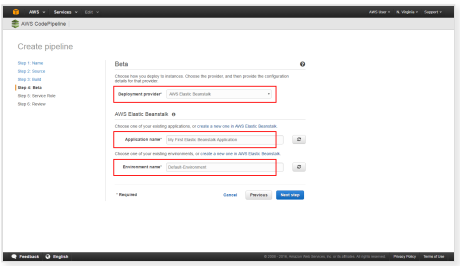


(click to zoom)

e. In the *Step 4: Beta* page:

- Deployment provider: Click **AWS Elastic Beanstalk.**
- Application name: Click **My First Elastic Beanstalk Application.**
- Environment name: Click **Default-Environment.**
- Click **Next step**.

Note: The name "Beta" is simply the name given by default to this stage of the pipeline, just as "Source" was the name given to the first stage of the pipeline.

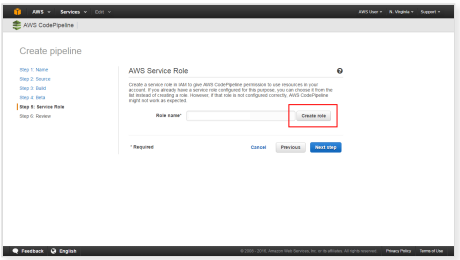


(click to zoom)

f. In the *Step 5: Service Role* page:

- Service Role: Click **Create role.**
- You will be redirected to an IAM console page that describes the AWS-CodePipeline-Service role that will be created for you. Click **Allow**
- After you create the role, you are returned to the *Step 5: Service Role* page where *AWS-CodePipeline-Service* appears in Role name. Click **Next step**.

Note: Service role creation is only required the first time you create a pipeline in AWS CodePipeline. If a service role has already been created, you will be able to choose it from the drop-down list of roles. Because the drop-down list will display all IAM service roles associated with your account, if you choose a name different from the default, be sure that

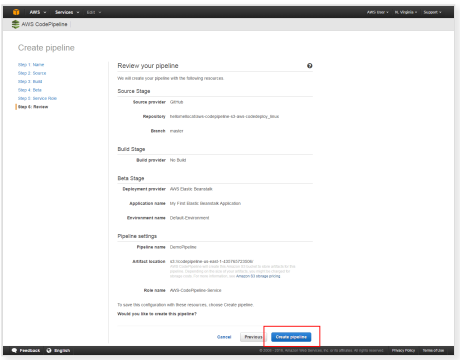


(click to zoom)

Step 4: Activate your pipeline to deploy your code

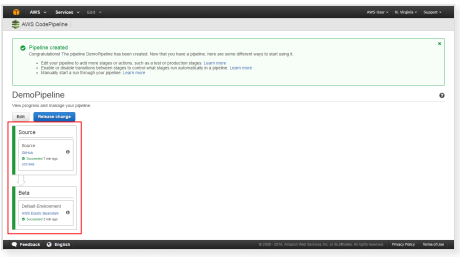
In this step, you will launch your pipeline. Once your pipeline has been created, it will start to run automatically. First, it detects the sample app code in your source location, bundles up the files, and then move them to the second stage that you defined. During this stage, it passes the code to Elastic Beanstalk, which contains the EC2 instance that will host your code. Elastic Beanstalk handles deploying the code to the EC2 instance.

a. In the *Step 6: Review* page, review the information and click **Create pipeline**.



(click to zoom)

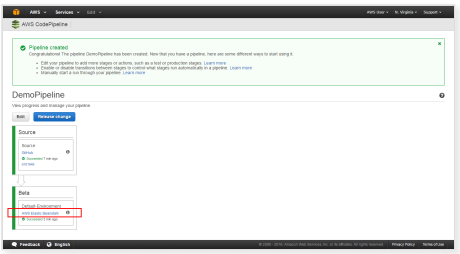
b. After your pipeline is created, the pipeline status page appears and the pipeline automatically starts to run. You can view progress as well as success and failure messages as the pipeline performs each action.



(click to zoom)

To verify your pipeline ran successfully, monitor the progress of the pipeline as it moves through each stage. The status of each stage will change from *No executions yet* to *In*

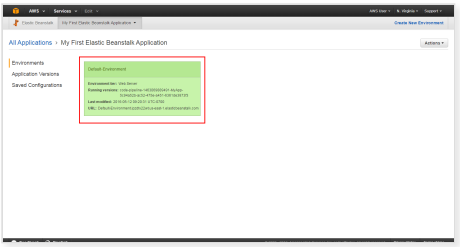
c. In the status area for the Beta stage, click **AWS Elastic Beanstalk**.



(click to zoom)

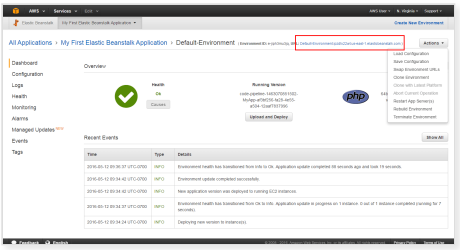
d. The AWS Elastic Beanstalk console opens with the details of the deployment.

- Click the environment you created earlier, called **Default-Environment**.



(click to zoom)

e. Click the URL that appears in the upper-right part of the page to view the sample website you deployed.



(click to zoom)

Step 5: Commit a change and then update your app

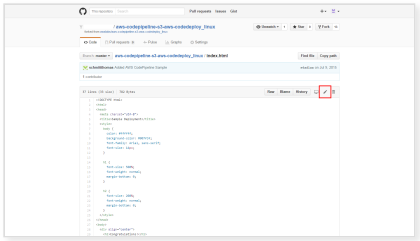
deploying it to your EC2 instance via Elastic Beanstalk.

Note that the sample web page you deployed refers to AWS CodeDeploy, a service that automates code deployments. In CodePipeline, CodeDeploy is an alternative to using Elastic Beanstalk for deployment actions. Let's update the sample code so that it correctly states that you deployed the sample using Elastic Beanstalk.

GitHub

a. Visit your own copy of the repository that you forked in GitHub.

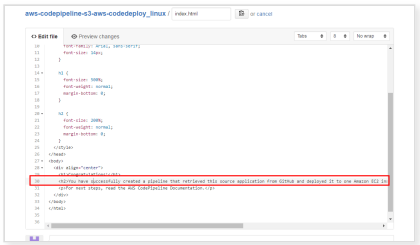
- Open *index.html*
- Select the **Edit** icon.



(click to zoom)

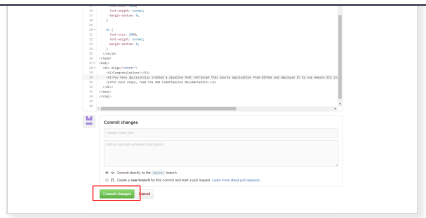
b. Update the webpage by copying and pasting the following text on line 30:

```
You have successfully created a pipeline that retri
eved this source application from GitHub and deploy
ed it to one Amazon EC2 instance using AWS Elastic
 Beanstalk. You're one step closer to practicing co
ntinuous deployment!
```



(click to zoom)

c. Commit the change to your repository.



(click to zoom)

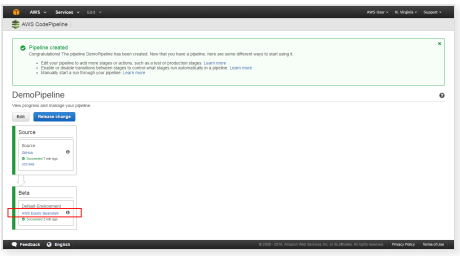
Amazon S3

AWS CodeCommit

---

d. Return to your pipeline in the CodePipeline console. In a few minutes, you should see the Source change to blue, indicating that the pipeline has detected the changes you made to your source repository. Once this occurs, it will automatically move the updated code to Elastic Beanstalk.

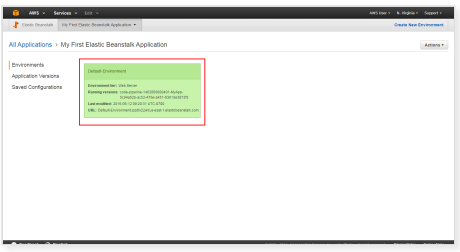
- After the pipeline status displays *Succeeded*, in the status area for the Beta stage, click **AWS Elastic Beanstalk**.



(click to zoom)

---

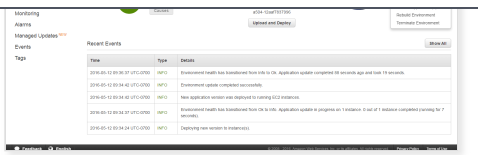
e. The AWS Elastic Beanstalk console opens with the details of the deployment. Select the environment you created earlier, called **Default-Environment**.



(click to zoom)

---

updated automatically through the continuous deployment pipeline!



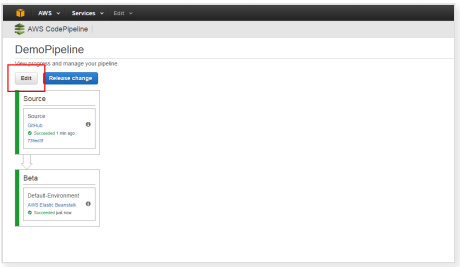
(click to zoom)

Step 6: Clean up your resources

To avoid future charges, you will delete all the resources you launched throughout this tutorial, which includes the pipeline, the Elastic Beanstalk application, and the source you set up to host the code.

a. First, you will delete your pipeline:

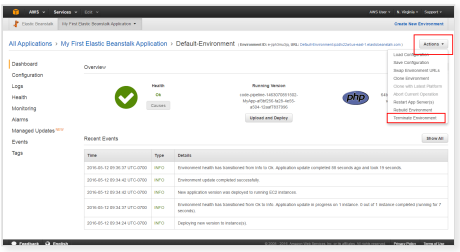
- In the pipeline view, click **Edit**.
- Click **Delete**.
- Type in the name of your pipeline and click **Delete**.



(click to zoom)

b. Second, delete your Elastic Beanstalk application:

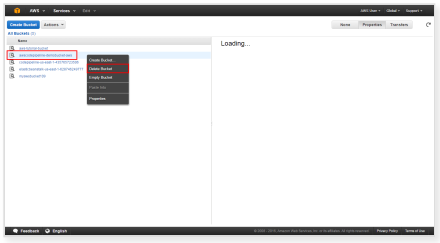
- Visit the Elastic Beanstalk console.
- Click **Actions.**
- Then click **Terminate Environment**.



(click to zoom)

c. If you created an S3 bucket for this tutorial, delete the bucket you created:

- Visit the S3 console.
- Right-click the bucket name and select **Delete Bucket**.
- When a confirmation message appears, enter the bucket name and then click **Delete**.



(click to zoom)

AWS CodeCommit

Congratulations!

You have successfully created an automated software release pipeline using AWS CodePipeline! Using CodePipeline, you created a pipeline that uses GitHub, Amazon S3, or AWS CodeCommit as the source location for application code and then deploys the code to an Amazon EC2 instance managed by AWS Elastic Beanstalk. Your pipeline will automatically deploy your code every time there is a code change. You are one step closer to practicing continuous deployment!

Next Steps

Now that you have learned to create a simple pipeline using AWS CodePipeline, you can learn more by visiting the following resources.

and an AWS CodeDeploy application to deploy the built code to a staging server.

- Quickly spin up a four-stage pipeline with a Jenkins build server by using our Pipeline Starter Kit.
- Learn more about continuous delivery.

Was this page helpful?

Yes No

Legal

Create an AWS Account

Amazon is an Equal Opportunity Employer: *Minority / Women / Disability / Veteran / Gender Identity / Sexual Orientation / Age.*

Language

عربي |
Bahasa Indonesia |
Deutsch |
English |
Español |
Français |
Italiano |
Português |
Tiếng Việt |
Türkçe |
Русский |
ไทย |
日本語 |
한국어 |
中文 (简体) |
中文 (繁體)

Privacy
|
Site Terms
|