Blue-Green Deployments to AWS Elastic Beanstalk on the AWS Cloud

Using AWS CodePipeline for Deploying to AWS Elastic Beanstalk Environments

Quick Start Reference Deployment

Kirankumar Chandrashekar, Shivansh Singh
Amazon Web Services

May 2018
Last update: June 2018 (revisions)

aws

Contents

About This Guide

This Quick Start deployment guide discusses architectural considerations and configuration steps for deploying an application to an Amazon Web Services (AWS) Elastic Beanstalk environment on the AWS Cloud. The Quick Start creates a continuous integration/continuous delivery (CI/CD) pipeline using CodePipeline for a cost-effective, fault-tolerant architecture. It also provides an AWS CloudFormation template that you can use to automate the deployment.

The guide is for IT infrastructure architects, administrators, and DevOps professionals who are planning to implement or extend their deployments to applications that are hosted in an Elastic Beanstalk environment.

Quick Links

The links in this section are for your convenience. Before you launch the Quick Start, please review the architecture, configuration, network security, and other considerations discussed in this guide.

- If you have an AWS account, and you're already familiar with AWS services such as AWS Elastic Beanstalk, AWS Lambda, CodePipeline, and AWS CodeBuild, you can launch the Quick Start to build the architecture shown in Figure 1 for a new or an existing Elastic Beanstalk environment. The deployment takes approximately 15 minutes. If you're new to AWS or to Elastic Beanstalk, CodePipeline, and Git webhooks, please review the implementation details and follow the deployment steps provided later in this guide.

  Launch Quick Start

  If you want to take a look under the covers, you can view the AWS CloudFormation template that automates the deployment.

  View template

About Quick Starts

Quick Starts are automated reference deployments for key workloads on the AWS Cloud. Each Quick Start launches, configures, and runs the AWS compute, network, storage, and other services required to deploy a specific workload on AWS, using AWS best practices for security and availability.

Overview

CI/CD Pipeline for Blue-Green Deployment on AWS

From a disaster recovery and development perspective, when an application is developed and deployed to an Elastic Beanstalk environment, having two separate, but identical, environments—blue and green—helps increase availability and reduce risk. In this case, the blue environment is the production environment that normally handles live traffic. The CI/CD pipeline architecture creates a clone (green) of the live Elastic Beanstalk environment (blue). The pipeline then swaps the URLs between the two environments.

While CodePipeline deploys application code to the original environment—and testing and maintenance take place—the temporary clone environment handles the live traffic.

Suppose deployment to the blue environment fails because of issues with the application code. While the code is being corrected and recommitted to the repository, the green environment serves the live traffic, and there is no downtime. Once deployment to the blue environment is successful, and code review and code testing are completed, the pipeline once again swaps the URLs between the green and blue environments. The blue environment starts serving the live traffic again, and the pipeline terminates the temporarily created green environment. Not having to continuously run parallel environments saves costs.

This solution also allows integration with Git repositories via Git webhooks and enables automatic code deployment, immediately after the code is committed to the Git repository.

> **Note** To implement Git to S3 integration, which triggers the pipeline automatically after committing the code to a Git repository, use the Git Webhooks with AWS services deployment guide for Git Webhooks with AWS services.

Cost and Licenses

You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using the Quick Start.

This Quick Start launches a CodePipeline that interacts with the Lambda functions, CodeBuild service, Amazon Simple Notification Service (SNS), and Elastic Beanstalk environment to facilitate cost-efficient deployments.

Optionally you can enable the Git Webhooks with AWS Services Quick Start with this Quick Start, for linking your Git Repository to Amazon S3. Git to Amazon S3 provides an Amazon API Gateway endpoint and Lambda functions to handle the download, zipping, and deployment of code to Amazon S3.

AWS CodePipeline carries a cost for each active pipeline; see AWS CodePipeline pricing. CodeBuild and Amazon SNS use pay-as-you-go pricing; for details, see AWS CodeBuild and Amazon SNS. Depending on your configuration, the Git Webhooks with AWS Services Quick Start may deploy an AWS Key Management Service (AWS KMS) key; for pricing, see AWS Key Management Service pricing. API Gateway, Amazon S3, and Lambda costs vary depending on how often you commit code to your repository. Each commit triggers a request to the Lambda execution in API Gateway; for details, see the pricing pages for API Gateway, Amazon S3, and Lambda.

AWS Services

The core AWS components used by this Quick Start include the following services. (If you are new to AWS, see the Getting Started section of the AWS documentation.)

- AWS CodePipeline – CodePipeline is a continuous integration and continuous delivery service for fast and reliable application and infrastructure updates. CodePipeline builds, tests, and deploys your code every time there is a code change, based on the release process models you define. This enables you to rapidly and reliably deliver features and updates. You can easily build out an end-to-end solution by using our pre-built plugins for popular third-party services like GitHub or integrating your own custom plugins into any stage of your release process. With CodePipeline, you only pay for what you use. There are no upfront fees or long-term commitments.
- AWS Lambda – With AWS Lambda, you can run code without provisioning or managing servers. You pay only for the compute time you consume—there's no charge when your code isn't running. You can run code for virtually any type of application or backend service—all with zero administration. Just upload your code and Lambda takes care of

everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

- AWS CodeBuild – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy. CodeBuild eliminates the need to provision, manage, and scale your own build servers. It provides prepackaged build environments for the most popular programming languages and build tools such as Apache Maven, Gradle, and more. You can also customize build environments in CodeBuild to use your own build tools. CodeBuild scales automatically to meet peak build requests.
- AWS CloudFormation – AWS CloudFormation gives you an easy way to create and manage a collection of related AWS resources, and provision and update them in an orderly and predictable way. You use a template to describe all the AWS resources (e.g., EC2 instances) that you want. You don't have to individually create and configure the resources or figure out dependencies—AWS CloudFormation handles all of that.
- AWS IAM - AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.
- Amazon S3 - Amazon Simple Storage Service (Amazon S3) is storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. You can accomplish these tasks using the simple and intuitive web interface of the AWS Management Console.
- Amazon SNS - Amazon Simple Notification Service (SNS) is a flexible, fully managed pub/sub messaging and mobile notifications service for coordinating the delivery of messages to subscribing endpoints and clients. With SNS you can fan-out messages to a large number of subscribers, including distributed systems and services, and mobile devices. It is easy to set up, operate, and reliably send notifications to all your endpoints – at any scale. Get started using SNS in a matter of minutes using the AWS Management Console, AWS Command Line Interface, or using the AWS SDK with just three simple APIs. SNS eliminates the complexity and overhead associated with managing and operating dedicated messaging software and infrastructure.
- AWS Elastic Beanstalk - AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS. You can simply upload your code, and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, automatic scaling to

application health monitoring. You retain full control over the AWS resources powering your application and can access the underlying resources at any time. There is no additional charge for Elastic Beanstalk - you pay only for the AWS resources needed to store and run your applications.

Architecture

Deploying this Quick Start with default parameters builds the following architecture, shown in Figure 1, for a blue-green deployment to an Elastic Beanstalk environment.

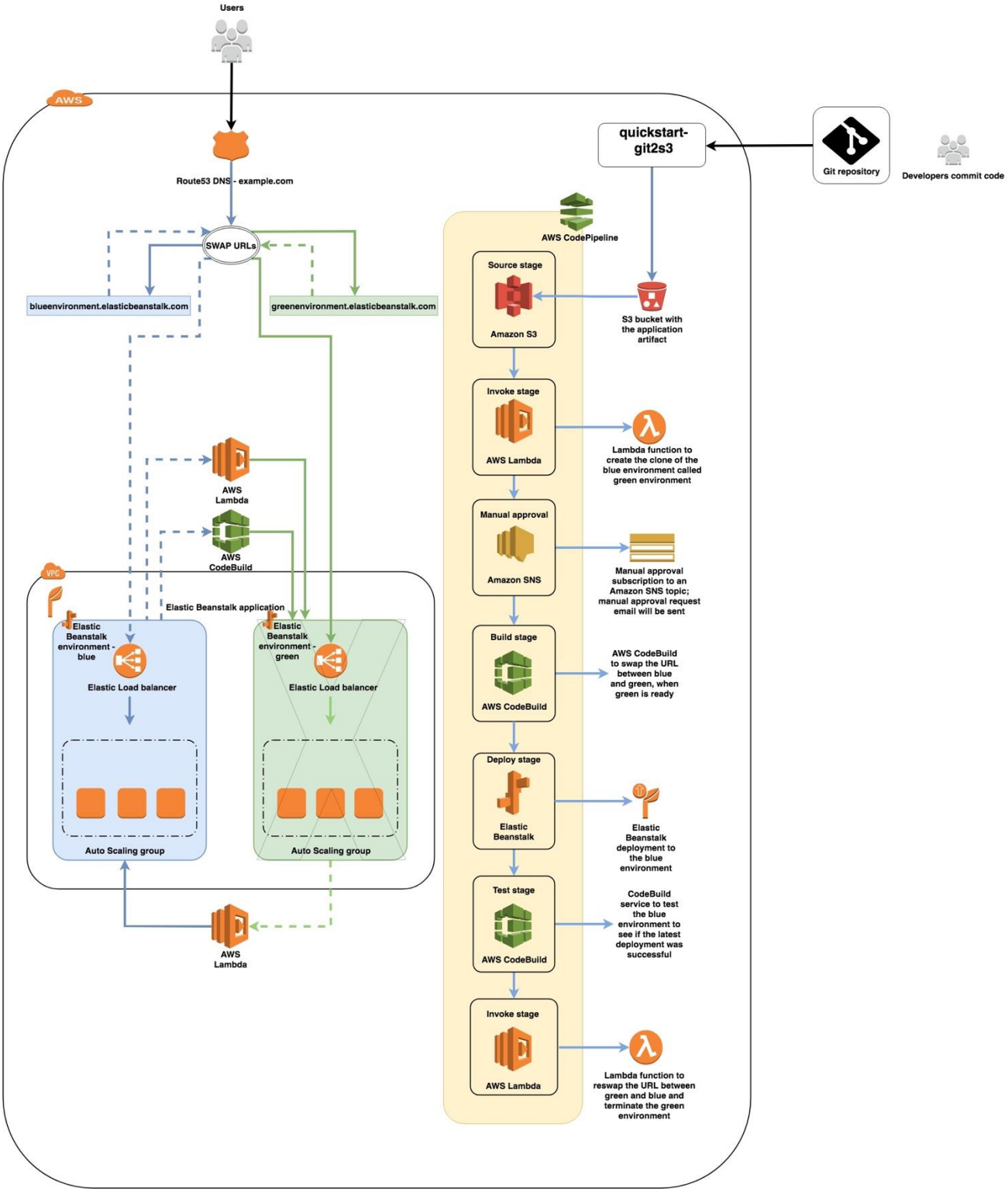

Figure 1: Quick Start architecture for blue/green deployments to Elastic Beanstalk

The Quick Start sets up the following components:

- A pipeline created by CodePipeline, which is triggered when the S3 bucket referenced in the source stage is uploaded with a new version of the application package required for deploying to the Elastic Beanstalk environment.
- Lambda functions to clone the blue environment as the green environment, to re-swap the URLs, and to terminate the green environment when deployment and testing for the blue environment are complete.
- CodeBuild projects to perform the initial URL swap between the blue and green environments and to test the deployment of the blue environment.
- An SNS topic for setting up the CodePipeline approval stage and for sending the approval email.
- IAM roles for the Lambda functions, CodeBuild projects, CodePipeline, and service role for the Elastic Beanstalk sample application.
- Separate Amazon S3 buckets for the CodePipeline Artifact Store, the Lambda Function assets, and the CodeBuild assets. You can have an optional S3 bucket for storing the zip package that contains application code for the Elastic Beanstalk deployment.
- If an existing Elastic Beanstalk environment and the application where it exists is not provided, this Quick Start creates a sample environment named **BlueEnvironment** in the application named **BlueGreenBeanstalkApplication**.

If Git to S3 integration is enabled (when the parameter GitToS3integration is set to **True**), it creates the following components in addition to the preceding ones:

- An API Gateway endpoint to accept webhook requests from Git.
- Lambda functions to connect to the Git service, either over Secure Shell (SSH) or through the Git service's endpoint. These functions zip the code and upload it to Amazon S3.

> **Important** The Lambda functions that this Quick Start deploys must be able to communicate with your Git repository. For example, you can use a software as a service (SaaS)-based Git service that the Lambda service can reach through the internet.

- An AWS KMS key to encrypt the SSH private key that is used to connect to the repository over SSH.

- Two S3 buckets: One bucket stores the zipped contents of your Git repository, and the second bucket stores the AWS KMS-encrypted SSH private keys that are generated during stack creation. Note that the first bucket has versioning enabled, and all previous versions are retained indefinitely. If you'd like to manage the retention period for old versions, follow the instructions in the Amazon S3 documentation.
- Several IAM roles required for the Lambda functions and API Gateway. The inline permissions attached to these roles are scoped using the least privilege model.
- Two Lambda-backed AWS CloudFormation custom resources. One resource generates an SSH key pair, encrypts it using AWS KMS, and stores it in Amazon S3. The second resource deletes the content of the two S3 buckets on stack deletion.

> **Note** If you need backups, copy the contents of the Amazon S3 bucket before you delete the CloudFormation stack.

For more information about Git to S3 integration, see Git Webhooks with AWS services.

Blue-Green Deployment Process

When the code is pushed to your Git repository, you can use the Git Webhooks with AWS Services Quick Start to upload the code to an S3 bucket.

Using the Git Webhooks with AWS Quick Start is optional. You can also upload the zipped application package to the S3 bucket that's used as the source stage in CodePipeline. When the code is uploaded to this S3 bucket, the deployment starts. The source stage pulls the zipped package from the S3 bucket and creates an artifact that is used for deployment.

If the blue environment is serving the live traffic, and the green environment is the clone, the pipeline automatically performs the following actions shown in Figure 2:

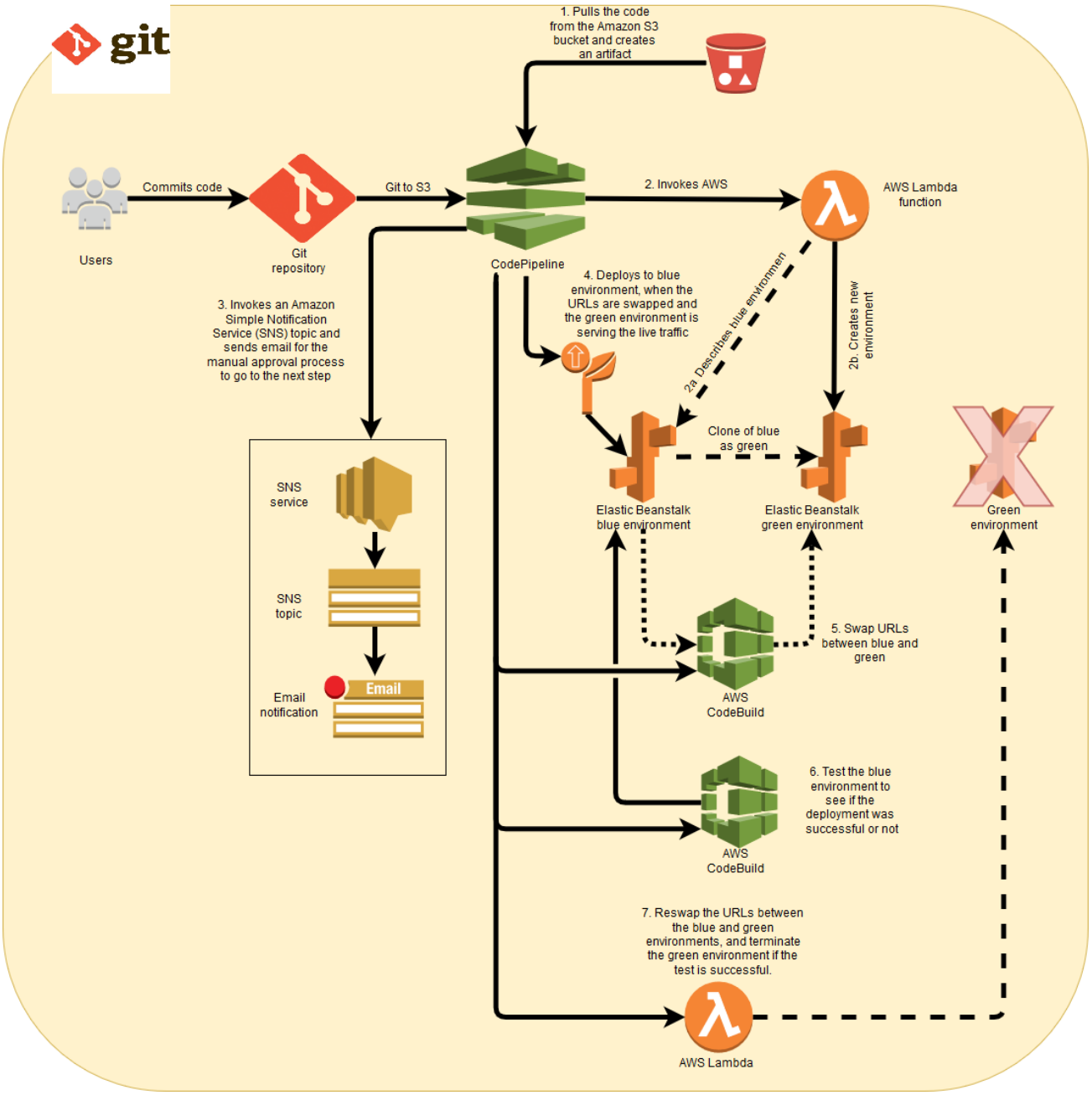

Figure 2: Blue-green deployment process

1. The first stage of the pipeline pulls the artifact from the source bucket and provides it for deployment to the Elastic Beanstalk environment.
2. The second stage triggers a Lambda function that clones the blue environment, which results in a green environment.
3. The third stage waits for manual approval before proceeding to the next stage.

4. The fourth stage swaps the URLs between the blue and the green environments using a CodeBuild project. Once this is complete, the green environment serves the live traffic.
5. The fifth stage performs the deployment to the blue environment.
6. If the deployment is successful, the sixth stage triggers a test on the blue environment to see if it can access a 200 OK in the response. If the response is other than 200 OK, the pipeline doesn't proceed, and marks this stage as failed.
7. If the test stage is successful, another Lambda function is triggered in the seventh stage, which again swaps the URLs between the blue and green environments and then terminates the green environment. The blue environment gets back its initial Elastic Beanstalk CNAME that it initially had for serving the live traffic.

Best Practices

> **Note** This Quick Start is intended for an already existing Elastic Beanstalk environment that you can use as a blue environment. The sample Elastic Beanstalk environment that is created via this Quick Start is for illustration and demonstration purposes only.

The architecture built by this Quick Start supports AWS best practices for high availability and security. We recommend that you adhere to the following best practices:

- Do not make any changes to the AWS resources created by this Quick Start directly on the AWS Management Console, command-line interface, or SDK, especially when the pipeline is running.
- Use a valid email address for the approval stage of the pipeline. This stage is crucial for the pipeline to proceed, swap the URLs, and perform the deployments.

Planning the Deployment

Deployment Options

This Quick Start provides two deployment options, using the same template, depending on whether the GitToS3integration parameter is set to **true** or **false**:

- **Deploy with Git to S3 integration.** This option builds a new AWS environment consisting of the pipeline created by AWS CodePipeline, along with other infrastructure components. It also builds an API Gateway interface and Lambda functions that allow an automatic trigger to the pipeline via webhooks, when the code is committed to the Git repository. The Git to S3 integration automatically uploads the code into the S3 bucket location referenced by the source stage of the pipeline, which automatically triggers code deployment on the pipeline.
- **Deploy without Git to S3 integration.** This option builds a new AWS environment consisting of the pipeline created by CodePipeline, along with other infrastructure components. The pipeline gets automatically triggered when the zip file package is uploaded to the S3 bucket location that is referenced in the source stage.

For both options, you can create a sample Elastic Beanstalk environment to test the solution. You can also modify the sample template for Elastic Beanstalk or use your own. By modifying the template, you can replace the Elastic Beanstalk environment with your own environment configuration within the same Quick Start stack. The environment that this stack creates can be provided as input to the pipeline stack as a blue environment.

Prerequisites

Create an Elastic Beanstalk environment in your AWS account, or make sure that you already have an environment that can be considered a blue environment. To create an Elastic Beanstalk environment, follow the steps in AWS Elastic Beanstalk environment.

If you do not have an Elastic Beanstalk environment in your account, you can launch this Quick Start with the sample application, which deploys a PHP sample solution. The Quick Start will deploy a sample application by default, in this case.

To have a different solution stack as part of the sample environment, provide the appropriate values for the parameters **SolutionStackForNewBeanstalkEnv**, **AppPackageS3Bucket** and **AppPackageS3key** within the Mappings section of bluegreen-deployment-master.template when you launch the CloudFormation stack for this Quick Start.

Deployment Steps

The procedure for deploying a blue-green architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Prepare an AWS account

Sign up for an AWS account, choosing a region, and requesting increases for account limits, if necessary.

Step 2. (Optional) Make a note of an existing Elastic Beanstalk environment and application

Make a note of an existing Elastic Beanstalk environment that can be considered a blue environment for the pipeline. You can skip this step if you want to launch a sample environment and application.

Step 3. Launch the Quick Start

Launch the AWS CloudFormation template into your AWS account, specify parameter values, and create the stack.

Step 4. (Optional) If Git to S3 integration is enabled, configure your Git repository

After the deployment of the AWS CloudFormation template is complete, configure your Git service to create the webhook. You can ignore this, if you aren't using Git to S3 integration.

Step 5. Test a commit

If Git to S3 integration is enabled, commit the application code to the Git repository. The pipeline should start the deployment automatically when Git webhooks invokes the API Gateway. This allows the Lambda function to pull the code and upload the zip package to the S3 bucket referenced by the source bucket.

If Git to S3 integration is not enabled, upload the zip package that contains the application code to the S3 bucket for the source stage, referenced by CodePipeline, so that the pipeline starts automatically.

Step 1. Prepare Your AWS Account

1. If you don't already have an AWS account, create one at https://aws.amazon.com by following the on-screen instructions. Part of the sign-up process involves receiving a phone call and entering a PIN using the phone keypad.
2. Use the region selector in the navigation bar to choose the AWS Region where you want to deploy a blue-green architecture on AWS. For more information, see Regions and

Availability Zones. Regions are dispersed and located in separate geographic areas. Each Region includes at least two Availability Zones that are isolated from one another but connected through low-latency links.

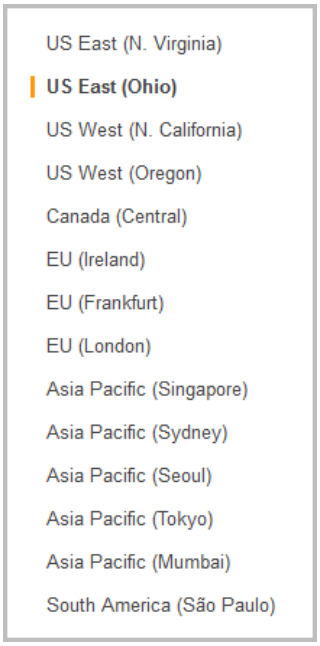

Figure 3: Choosing an AWS Region

Consider choosing a region closest to your data center or corporate network to reduce network latency between systems running on AWS and the systems and users on your corporate network. The template is launched in the US East (Ohio) Region by default.

Step 2. (Optional) Make a Note of an Existing Elastic Beanstalk Environment and Application

1. Navigate to the Elastic Beanstalk service on the console.
2. Click the environment where you want to apply the blue-green deployment architecture. Note down the name of the Elastic Beanstalk environment and the Elastic Beanstalk application where it exists.
3. You can ignore this if you would like to test this architecture with a sample Elastic Beanstalk environment and application.

Step 3. Launch the Quick Start

1. Launch the AWS CloudFormation template into your AWS account.

   You can also download the template to use it as a starting point for your own implementation. The stack takes approximately 15 minutes to create.

> **Note** You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using this Quick Start. Prices are subject to change. See the pricing pages for each AWS service you will be using in this Quick Start for full details.

2. Check the region that's displayed in the upper-right corner of the navigation bar, and change it if necessary. This is where the network infrastructure for the blue-green deployment on the AWS Cloud will be built. The template is launched in the US East (Ohio) Region by default.
3. On the **Select Template** page, keep the default setting for the template URL, and then choose **Next**.
4. On the **Specify Details** page, change the stack name if needed. Review the parameters for the template. Provide values for the parameters that require input. For all other parameters, review the default settings and customize them as necessary. When you finish reviewing and customizing the parameters, choose **Next**.

   In the following tables, parameters are listed by category and described for deploying with Git to S3 integration or without this integration, depending on whether the GitToS3integration parameter is set to **true** or **false**.

- **Parameters for deploying with or without Git to S3 integration**

  AWS Elastic Beanstalk configuration:

Parameter label (name)	Default	Description
Beanstalk Source Stage S3 Bucket (BeanstalkSourceStageS3BucketName)		**(Optional)** The S3 bucket for the source stage of the pipeline that CodePipeline uses to pull the application package. If the GitToS3integration parameter value is set to **false** and if this value is left blank, a bucket will be created. If a bucket name is provided, that will be the bucket CodePipeline will look in to get the application source package. If the GitToS3integration parameter value is set to **true**, a new bucket will be created with this name to put the source package (zip file). If left blank, a bucket name will be automatically generated. Provide this value only when the GitToS3integration parameter is set to **true**.
Beanstalk Source Stage S3 Bucket key (BeanstalkSourceStageS3BucketKey)		The path to the Elastic Beanstalk application source package (in .zip format) that will trigger the source stage in the pipeline. If the GitToS3integration parameter is set to **true**, this parameter is the S3 bucket key where the Git webhook will place the zip file as input to the source stage in the pipeline. The key should be in the format git-user/git-repository/git-user_git-repository.zip. This value differs from one Git repo to another. For more information, see the deployment steps in the Git Webhooks with AWS Services Quick Start deployment guide.
Existing Blue Environment Name (ExistingBlueEnvironmentName)		**(Optional)** The name of the Elastic Beanstalk environment that's designated as the blue environment. Leave this value blank, if you want to create a new sample environment with the PHP solution stack. Otherwise, provide a value to use for blue-green deployment.
Existing Beanstalk Application Name ExistingBeanstalkApplicationName)		**(Optional)** Required if the parameter ExistingBlueEnvironmentName is provided. Give the name of the existing Elastic Beanstalk application where the existing Elastic Beanstalk environment is

Parameter label (name)	Default	Description
		running. Leaving this value blank creates a new sample application. You must provide a value for this parameter if you provide a value for ExistingBlueEnvironmentName.
Green Environment Name (GreenEnvironmentName)	GreenEnvironment	The name of the green environment, which will be the clone of the blue environment. Traffic will be routed temporarily to the green environment, until the blue environment is finished with the deployment and testing. After successful deployment to the blue environment, traffic will be routed back to the blue environment, and the green environment will be terminated.

CodePipeline parameters:

Parameter label (name)	Default	Description
Admin Email Address (AdministratorEmail)	*Requires input*	The administrator email address for the manual approval stage in the pipeline. The Amazon SNS subscription email will be sent to this address.
Name of the Pipeline (NameofthePipeline)	BlueGreenCICDPipeline	The name of the CI/CD pipeline used in the blue-green deployment.

Git2s3 setup parameters. Valid only when GitToS3integration parameter is set to true. Ignore otherwise:

Parameter label (name)	Default	Description
Git to S3 Integration (GitToS3integration)	false	Select **true** if you want to set up Git to S3 integration. If not, leave the default value as **false**. This will allow you to use an existing Git repository when deploying an application to Elastic Beanstalk.
Custom Domain Name (CustomDomainName)	—	**(Optional)** Use a custom domain name for the webhook endpoint. If left blank, API Gateway will create a domain name for you. Provide this value only when the **GitToS3integration** parameter is set to **true**.

Parameter label (name)	Default	Description
API Secret (ApiSecret)	—	**(Optional)** This applies only to the gitpull method. WebHook Secrets for use with GitHub Enterprise and GitLab. If a secret is matched IP range authentication is bypassed. Cannot contain: , \ "Provide this value only when the GitToS3integration parameter is set to **true**.
Allowed IPs (AllowedIps)	131.103.20.160/27 165.254.145.0/26 104.192.143.0/24	**(Optional)** This applies only to the gitpull method. This is a comma-separated list of IP CIDR blocks for source IP authentication. Bitbucket Cloud IP ranges are provided as defaults. Provide this value only when the GitToS3integration parameter is set to **true**.
Git Personal Access Token (GitToken)	—	**(Optional)** This applies to the zipdl method only. Personal access token, needed for GitHub Enterprise and GitLab. Provide this value only when the GitToS3integration parameter is set to **true**.
OAuth2 Key (OauthKey)	—	**(Optional)** This applies to the zipdl method only. The OAuth2 Key needed for Bitbucket. Provide this value only when the GitToS3integration parameter is set to **true**.
OAuth2 Secret (OauthSecret)	—	**(Optional)** This applies to the zipdl method only. The OAuth2 Secret needed for Bitbucket. Provide this value only when the GitToS3integration parameter is set to **true**.

AWS Quick Start Configuration:

Parameter label (name)	Default	Description
Quick Start S3 Bucket Name (QSS3BucketName)	aws-quickstart	The S3 bucket name for the Quick Start assets. The Quick Start bucket name can include numbers, lowercase letters, uppercase letters, and hyphens (-). It cannot start or end with a hyphen (-).
Quick Start S3 Key Prefix (QSS3KeyPrefix)	quickstart-codepipeline-bluegreen-deployment/	The S3 key prefix for the Quick Start assets. The Quick Start key prefix can include numbers, lowercase letters, uppercase letters, hyphens (-), and forward slash (/).

5. On the **Options** page, you can specify tags (key-value pairs) for resources in your stack and set advanced options. When you're done, choose **Next**.
6. On the **Review** page, review and confirm the template settings. Under **Capabilities**, select the check box to acknowledge that the template will create IAM resources.
7. Choose **Create** to deploy the stack.
8. Monitor the status of the stack. When the status is **CREATE_COMPLETE**, the CI/CD pipeline architecture for blue-green deployment is ready.

Step 4. (Optional) If Git to S3 Integration Is Enabled, Configure Your Git Repository

> **Note** For detailed information, see the Git Webhooks with AWS Services Quick Start deployment guide.

If the **GitToS3integration** parameter is set to **true**, you need to configure the webhooks for your Git repository.

Figure 4 shows the **Outputs** tab in the AWS CloudFormation console, which displays the outputs for configuring your Git webhook.

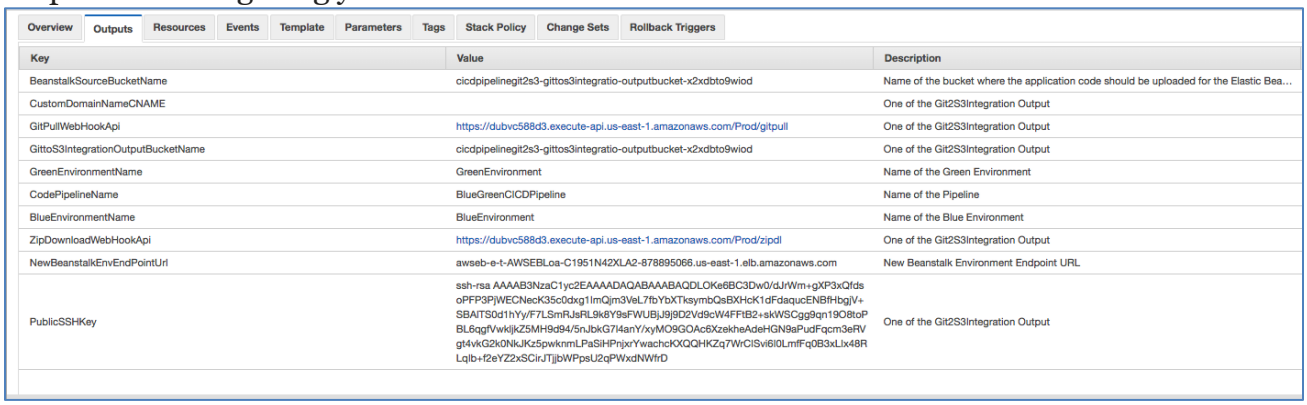

Figure 4: Outputs tab for configuring a Git webhook, when GitToS3integration is set to true

GitPullWebHookApi is the webhook endpoint to use if you choose the Git pull method described in the Webhook Endpoints section of the Git Webhooks with AWS Services Quick Start deployment guide.

ZipDownloadWebHookApi is the webhook endpoint to use if you choose the zip download method described in the Webhook Endpoints section of the Git Webhooks with AWS Services Quick Start deployment guide.

PublicSSHKey is the public SSH key that you use to connect to your repository if you're using the Git pull endpoint. This key can be configured as a read-only machine user or as a deployment key in your Git service. The exact process to set up webhooks differs from service to service. For step-by-step instructions, consult your Git service's documentation.

Step 5. Test a Commit

Check that the **GitToS3integration** parameter is set to **true** and that the webhooks are set up correctly. If so, when you commit the code to the Git repository, your application code will be zipped and placed in the S3 bucket that is displayed in the **Outputs** section of the CloudFormation stack called **BeanstalkSourceBucketName**. This bucket should have a key in the following format:

S3://output-bucket-name/git-user/git-repository/git-user_git-repository.zip

where:

- git-user is the owner or path prefix of the repository. In some Git services, this may be an organization name.

- Some Git services do not return a Git user or organization for a repository. In these cases, you can omit the git-user parts of the path.

  Because this is the S3 bucket and key referred to in the source stage of the S3 bucket, the pipeline triggers and starts the deployment.

If the **GitToS3integration** parameter is set to **false**, manually upload the zip file that contains the application package to the S3 bucket that is displayed in the **Outputs** section of the AWS CloudFormation stack called **BeanstalkSourceBucketName**, shown in Figure 5. The pipeline will automatically start the deployment process. The key name for the zip file should align with the value provided for the **BeanstalkSourceStageS3BucketKey** parameter when you create the stack.

> **Note** You can update the value of the **BeanstalkSourceStageS3BucketKey** parameter by performing stack updates.

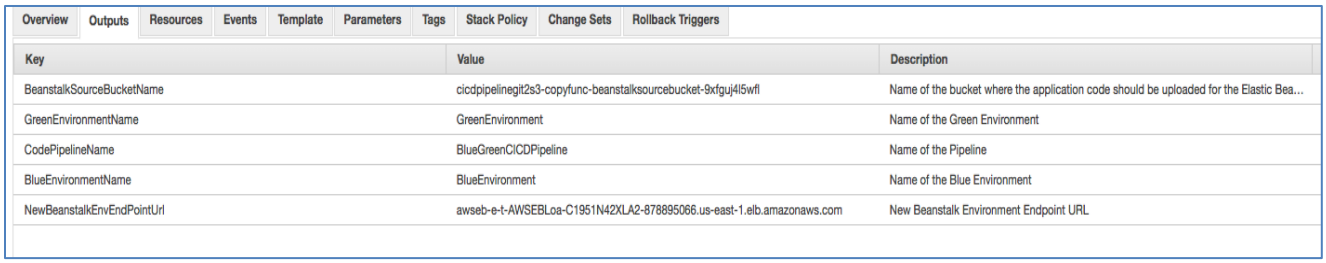

Figure 5: Outputs section when GitToS3integration is set to false

> **Note** If you are using the sample environment and you want to test this Quick Start, download the zip files that contain the Elastic Beanstalk sample application from the Elastic Beanstalk documentation.

Troubleshooting

Q. I encountered a CREATE_FAILED error when I launched the Quick Start.

A. When you deploy the Quick Start, if you encounter a CREATE_FAILED error instead of the CREATE_COMPLETE status code, we recommend that you relaunch the template with **Rollback on failure** set to **No**. (This setting is under **Advanced** in the AWS CloudFormation console, **Options** page.) With this setting, the stack's state will be retained so you can troubleshoot the issue. (Look at the AWS Lambda console's **Monitoring** tab.)

> **Important** When you set Rollback on failure to **No**, you will continue to incur AWS charges for this stack. Please make sure to delete the stack when you finish troubleshooting.

If your commits are not being pushed through to Amazon S3 when using Git to S3 integration, check the following:

- In your Git webhooks configuration, check that your configured security parameters and the endpoint are correct. Consult the Git service documentation for detailed guidance on configuration.
- Check the Lambda logs for errors. These are stored in Amazon CloudWatch Logs. To access the logs, open the endpoint's Lambda function in the AWS Lambda console, navigate to the **Monitoring** tab, and then choose **View logs in CloudWatch**.

For additional information, see Troubleshooting AWS CloudFormation on the AWS website. If the problem you encounter isn't covered on that page or in the table, please visit the AWS Support Center.

For additional information, see Troubleshooting AWS CloudFormation on the AWS website. If the problem you encounter isn't covered on that page or in the table, please visit the AWS Support Center.

Q. I encountered a size limitation error when I deployed the AWS CloudFormation templates.

A. We recommend that you launch the Quick Start templates from the links in this guide or from another S3 bucket. If you deploy the templates from a local copy on your computer or from a non-S3 location, you might encounter template size limitations when you create the stack. For more information about AWS CloudFormation limits, see the AWS documentation.

Q. The source stage of the pipeline failed, with an error similar to the following:

> Invalid action configuration
> Could not access the Amazon S3 object: "cicdpipelinegit2s3-copyfunc-beanstalksourcebucket-9xfguj4l5wfl/application-code.zip". Make sure that the names of the bucket and object are both correct.

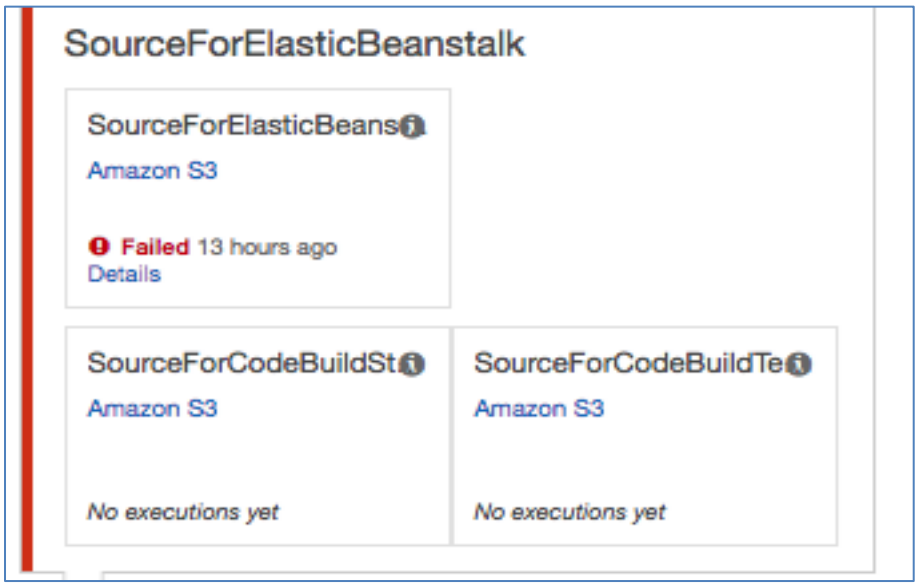

Figure 6: Error related to failure of source stage

This might be due to not uploading the application package file to the S3 bucket for the source stage. In this case, the pipeline expects a file named cicdpipelinegit2s3-copyfunc-beanstalksourcebucket-9xfguj4l5wfl/application-code.zip within the S3 bucket. If it is not available, the source stage fails.

Q. I want to use the sample Elastic Beanstalk environment, and I need the Python solution stack.

A. We recommend that you create an Elastic Beanstalk environment outside of this Quick Start, and provide the names of the environment and application as parameters for the pipeline to use. The sample Elastic Beanstalk application comes with a PHP solution stack. If you want to change it, you must modify the values in the mappings section of bluegreen-deployment-master.template, to match the values that appear in the Elastic Beanstalk documentation.

Q. What do I do when deploying to the blue environment fails and the green environment is still serving the live traffic?

A. If this is an application-related issue, commit the fixed code to the Git repository again or to the S3 bucket that the pipeline is referencing, and let the pipeline start from the

beginning. In this case, if the green environment already has the blue environment's URL, the URLs will not be swapped again. They will be re-swapped only when deploying to the blue environment succeeds, and the pipeline transitions to the next stage.

GitHub Repository

You can visit our GitHub repository to download the templates and scripts for this Quick Start, to post your comments, and to share your customizations with others.

Additional Resources

AWS services

- AWS CloudFormation
  https://aws.amazon.com/documentation/cloudformation/
- AWS CodePipeline
  https://aws.amazon.com/documentation/codepipeline/
- AWS CodeBuild
  https://aws.amazon.com/documentation/codebuild/
- AWS Lambda
  https://aws.amazon.com/documentation/lambda
- Amazon SNS
  https://aws.amazon.com/documentation/sns/
- AWS Elastic Beanstalk
  https://aws.amazon.com/documentation/elastic-beanstalk/
- Amazon S3
  https://aws.amazon.com/documentation/s3/

Git webhooks

- GitHub Developer Repository Webhooks API
  https://developer.github.com/v3/repos/hooks/
- Atlassian Bitbucket Webhooks documentation
  https://confluence.atlassian.com/bitbucket/manage-webhooks-735643732.html
- GitLab Webhooks documentation
  https://docs.gitlab.com/ce/user/project/integrations/webhooks.html

Quick Start Git to S3

- Git Webhooks with AWS services
  https://aws.amazon.com/quickstart/architecture/git-to-s3-using-webhooks/
- Git Webhooks with AWS services documentation
  https://aws-quickstart.s3.amazonaws.com/quickstart-git2s3/doc/git-to-amazon-s3-using-webhooks.pdf

Quick Start reference deployments

- AWS Quick Start home page
  https://aws.amazon.com/quickstart/

Document Revisions

Date	Change	In sections
June 2018	Removed “optional” from the parameter description for BeanstalkSourceStageS3BucketKey to reflect template update	Parameters for deploying with or without Git to S3 integration
May 2018	Initial publication	—

Notices

This document is provided for informational purposes only. It represents AWS’s current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS’s products or services, each of which is provided “as is” without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

The software included with this paper is licensed under the Apache License, Version 2.0 (the "License"). You may not use this file except in compliance with the License. A copy of the License is located at http://aws.amazon.com/apache2.0/ or in the "license" file accompanying this file. This code is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.