Git Webhooks with AWS Services

Linking Your Git Repository to Amazon S3 and AWS Services for Continuous Code Integration, Testing, and Deployment

Quick Start Reference Deployment

Jay McConnell, Karthik Thirugnanasambandam, and Santiago Cardenas
Amazon Web Services

September 2017

Contents

About This Guide

This Quick Start deployment guide discusses architectural considerations and configuration steps for deploying Git webhooks to push code to an Amazon Simple Storage Service (Amazon S3) bucket on the Amazon Web Services (AWS) Cloud. It also provides links for viewing and launching the AWS CloudFormation template that automates the deployment.

The guide is for IT infrastructure architects, administrators, and DevOps professionals who are planning to implement AWS services that use Amazon S3 as a source in the AWS Cloud. Examples of such services include AWS CodePipeline, AWS CodeBuild, and AWS CodeDeploy.

This Quick Start creates webhook endpoints and deploys an AWS Lambda function to push your code to Amazon S3. It does not create or configure webhooks, because the process varies depending on which Git software you are using. For specific guidance, consult your Git service documentation.

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 and Git webhooks, you can launch the Quick Start to build the architecture shown in Figure 1. The deployment takes approximately 15 minutes. If you're new to AWS or to Git webhooks, please review the implementation details and follow the step-by-step instructions 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

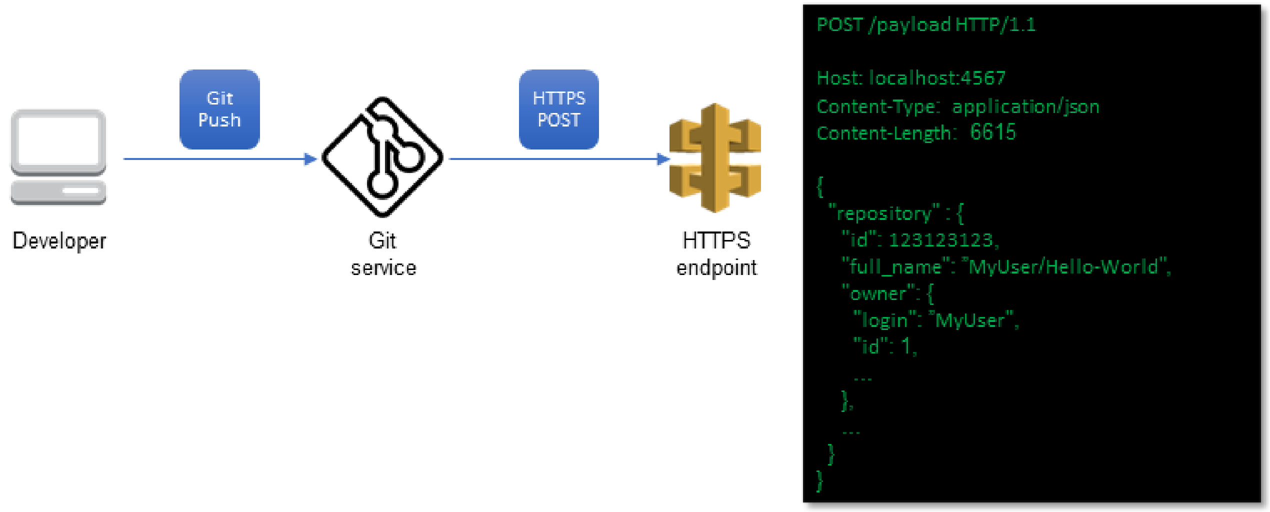
Quick Starts are automated reference deployments for key technologies on the AWS Cloud, based on AWS best practices for security and high availability.

Overview

Using Webhooks on AWS

This Quick Start implements HTTPS endpoints and code that helps you link AWS services with webhooks. Git webhooks enable event-driven integration between Git services and external applications. This Quick Start uses this integration to receive an event when commits are pushed to a Git repository.

As Figure 1 illustrates, when code is pushed to your repository, the Git service sends an HTTPS POST to the endpoints configured by the Quick Start. The POST contains JSON data about the push operation, including the repository details that the Quick Start uses to fetch the latest version of your code.



```
POST /payload HTTP/1.1

Host: localhost:4567
Content-Type:  application/json
Content-Length:  6615

{
 "repository" : {
   "id": 123123123,
   "full_name": "MyUser/Hello-World",
   "owner": {
    "login": "MyUser",
    "id": 1,
    ...
   },
   ...
 }
}
```

Figure 1: Using webhooks for code commits

This Quick Start implements the required code to trigger a Lambda function that zips up the code in your repository and places the .zip file in Amazon S3. When this function is triggered by a Git webhook, it provides a convenient way to bridge Git services with AWS services like AWS CodePipeline and AWS CodeBuild, so that builds and pipeline executions occur automatically when you commit your code to a Git repository. Linking your existing code repositories to the AWS Cloud in this way enables your code to be continuously integrated, tested, built, and deployed on the AWS Cloud with each change.

Important The Lambda functions that are deployed by this Quick Start must be able to communicate with your Git repository. For example, you can use a SaaS-based Git service that Lambda can reach through the internet.

AWS Services

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

- API Gateway – Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. You can create an API that acts as a "front door" for applications to access data, business logic, or functionality from your back-end services, such as workloads running on Amazon Elastic Compute Cloud (Amazon EC2), code running on Lambda, or any web application. API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management. This Quick Start uses API Gateway to receive the webhook request from your Git service and to forward it to Lambda.
- Lambda – AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume—there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service with zero administration. Just upload your code and Lambda runs and scales 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. This Quick Start implementation uses a Lambda function to retrieve your code from API Gateway and to place it in Amazon S3.
- KMS – AWS Key Management Service (AWS KMS) helps create and control the encryption keys used to encrypt your data, and uses hardware security modules (HSMs) to protect the security of your keys. AWS KMS is integrated with several other AWS services to help you protect the data you store with these services. AWS KMS is also integrated with AWS CloudTrail to provide you with logs of all key usage to help meet your regulatory and compliance needs. In this Quick Start implementation, KMS is used to encrypt the Secure Shell (SSH) private key that is generated at launch. The **GitPull** method uses the private key for SSH authentication of your Git repository.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is object storage with a simple web service interface to store and retrieve any amount of data from anywhere on the web. It is designed to deliver 99.999999999% durability. This Quick Start uses Amazon S3 to store code files and SSH private keys.
- 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.

In addition to these services, you'll want to be familiar with setting up and securing webhooks in your Git product. For information, see the product documentation for these technologies.

Costs 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.

The Quick Start provides an Amazon API Gateway endpoint and several 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. Depending on your configuration, the 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.

Architecture

Deploying this Quick Start builds the following environment in the AWS Cloud.

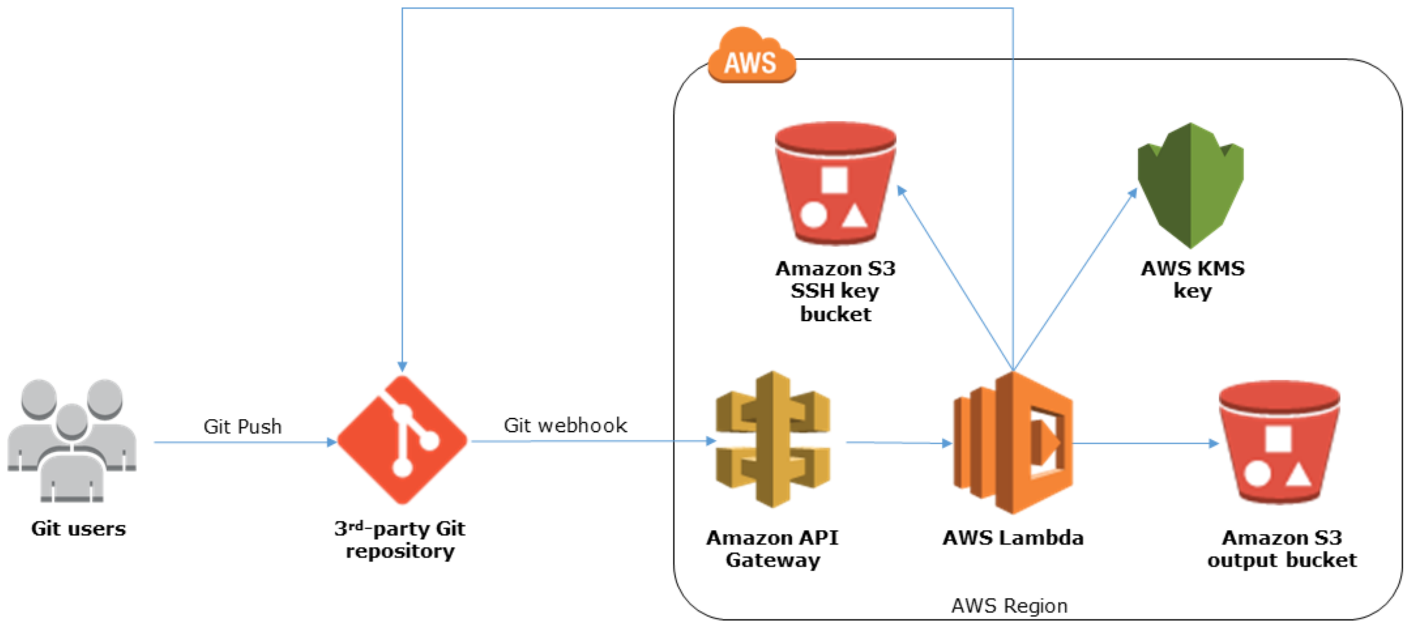


Figure 2: Webhook endpoint architecture on AWS

The Quick Start deployment sets up a serverless AWS Cloud environment that includes the following components.

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

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

- An AWS KMS key to encrypt the SSH private key 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 keypair, 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, make sure to copy the S3 bucket contents before you delete the CloudFormation stack.

Webhook Endpoints

This Quick Start includes two endpoints for connecting to your Git repository. Each endpoint has different benefits and limitations. We recommend that you consider your use case, repository size, and other requirements before choosing which endpoint to use.

- **Zip download endpoint** – Uses the Git provider's HTTP API to download a zipped copy of the current state of the repository. This endpoint has the following benefits:
  - No need for external libraries
  - Smaller Lambda function code
  - Large repository size limit (500 MB)

- **Git pull endpoint** – Uses SSH to pull from the Git repository. The repository contents are then zipped and uploaded to Amazon S3. This endpoint has the following benefits:
  - Efficient for repositories with a high volume of commits, because each time the API is triggered, it downloads only the changed files
  - Suitable for any Git server that supports hooks and SSH; doesn't depend on personal access tokens or OAuth2
  - More extensible because it uses a standard Git library

Best Practices

The architecture built by this Quick Start supports AWS best practices for security.

SSH Keys

SSH keys are generated at stack creation, and are then encrypted using AWS KMS and the encrypted copy stored in Amazon S3. When you use the Git pull endpoint, the private key is fetched by the Lambda function, decrypted, and used to authenticate against your Git service to perform a clone of the repository. We don't recommend (a) reusing SSH keys for multiple services, or (b) launching another instance of this Quick Start for each repository that you wish to clone to Amazon S3; this ensures that each repository uses unique keys.

Webhook Security

Different Git services provide varying ways to authenticate against an endpoint. The Git pull endpoint supports webhook secrets (used by GitHub Enterprise, GitLab, and other Git repository managers) as well as source IP address whitelisting. The zip download endpoint supports personal access tokens (as used by GitHub Enterprise and GitLab) and OAuth2 (used by Bitbucket). We recommend that you set up at least one of these security mechanisms to protect your webhook API endpoint. For more information about how this Quick Start utilizes these mechanisms, see the parameters in the Deployment Steps section of this guide. For product-specific guidance on how to configure these security mechanisms, refer to your Git product's documentation.

Deployment Steps

The procedure for an end-to-end deployment 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 and choose a region.

Step 2. Launch the Quick Start

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

Step 3. Configure your Git repository

After deployment is complete, configure your Git service to create the webhook.

Step 4. Configure an AWS service to connect to the object in Amazon S3

Configure an AWS service like AWS CodePipeline or AWS CodeBuild to use the S3 object as a source, and to automatically take action when the object is updated by a commit to the Git repository.

Step 5. Test a commit

Before putting the webhook into production, test your deployment to ensure that the object is being updated in Amazon S3 and that your configured service is triggered.

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 the Quick Start 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.

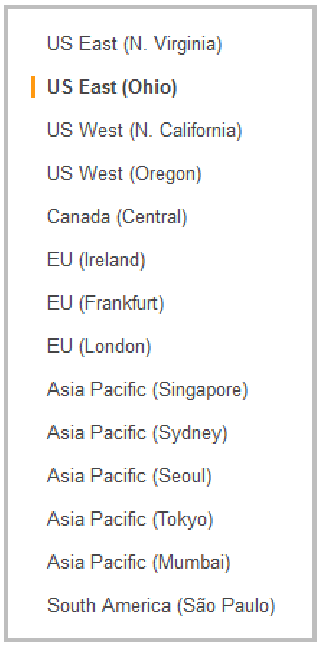


Figure 3: Choosing an AWS Region

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.

Step 2. Launch the Quick Start

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

Launch

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 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**.

   Parameters are grouped in four major categories and described in the following tables.

   General Settings:

Parameter label (name)	Default	Description
Output S3 Bucket Name (OutputBucketName)	—	(Optional) The name of the S3 bucket where the zip file output should be placed. If you leave this parameter blank, the Quick Start will automatically generate a bucket name, which will be included in the outputs of this stack. This S3 bucket is the location that you'd like to use as a source for your integrations. The full key is in the format: *output-bucket-name*/*git-user*/*git-repository*/ *git-user*_*git-repository*.zip For more information about using this key, see step 4.
Custom Domain Name (CustomDomainName)	—	(Optional) A custom domain name for the webhook endpoint. If you leave this parameter blank, API Gateway will create a domain name for you. The full URLs for both webhook endpoints will be available in the outputs of this stack.

   Git Pull Settings:

Parameter label (name)	Default	Description
API Secret (ApiSecret)	—	(Optional, Git pull method only) The webhook secret you want to use for security. If the webhook payload contains a matching secret, IP range authentication is bypassed. This value cannot contain a comma (,), backslash (\), or quotation mark ("). Webhook secrets are used by GitHub Enterprise, GitLab, and other Git services that provide API secrets. The secret must be in the webhook payload headers in one of the following formats: • X-Git-Token or X-Gitlab-Token headers with the secret in plain text.

Parameter label (name)	Default	Description
		• X-Hub-Signature header. The value must be a HMAC-SHA1 hexdigest of the JSON payload. The SHA1 key used to create the digest is the value you must provide for this parameter. Some Git services may provide these values only during the setup of the webhook; in these cases, leave this parameter blank and follow the steps outlined in the Adding an API Secret After Deployment section.
Allowed IPs (AllowedIps)	131.103.20.160/27, 165.254.145.0/26, 104.192.143.0/24	(Optional, Git pull method only) Comma-separated list of IP CIDR blocks for source IP authentication. The Bitbucket Cloud IP ranges are provided as the default.

Zip Download Settings:

Parameter label (name)	Default	Description
Git Personal Access Token (GitToken)	—	(Optional, zip download method only) The personal access token used by GitHub Enterprise and GitLab. This token is sent to the Git server's API as an authorization token when requesting the zip archive.
OAuth2 Key (OauthKey)	—	(Optional, zip download method only) The OAuth2 key used by Bitbucket Cloud. An access token is retrieved by sending the OAuth2 key and secret to the Bitbucket Cloud API. This token is then used to retrieve the code zip file.
OAuth2 Secret (OauthSecret)	—	(Optional, zip download method only) The OAuth2 secret used by Bitbucket Cloud. An access token is retrieved by sending the OAuth2 key and secret to the Bitbucket Cloud API. This token is then used to retrieve the code zip file.

AWS Quick Start Configuration:

Parameter label (name)	Default	Description
Quick Start S3 Bucket Name (QSS3BucketName)	aws-quickstart	The S3 bucket you have created for your copy of Quick Start assets, if you decide to customize or extend the Quick Start for your own use. The bucket name can include numbers, lowercase letters, uppercase letters, and hyphens, but should not start or end with a hyphen.
Quick Start S3 Key Prefix (QSS3KeyPrefix)	quickstart-git2s3/	The S3 key name prefix used to simulate a folder for your copy of Quick Start assets, if you decide to customize or extend the Quick Start for your own use. This prefix can include numbers, lowercase letters, uppercase letters, hyphens, and forward slashes.

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 webhook resources are ready.
9. The **Outputs** tab for the stack contain the two webhook endpoint URLs, the output bucket name, and the public SSH key, as illustrated in Figure 4.

Adding an API Secret After Deployment

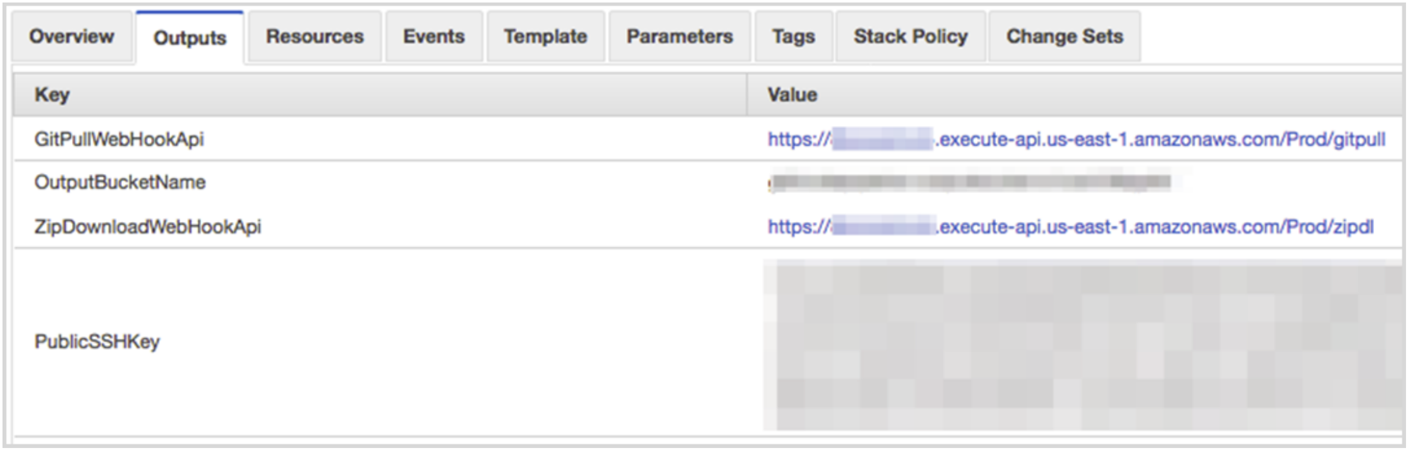
In some cases, your Git service may provide security mechanisms like API secrets when you create the webhook. In these cases, you can launch the Quick Start with a blank parameter value for the **API Secret** parameter, and then update the stack to provide the value of the parameter. Follow these steps:

1. In the AWS Cloudformation console, select the stack.
2. Choose **Actions**, and then choose **Update Stack**.
3. Keep the default to use the current template.
4. On the **Specify Details** page, change the **API Secret** parameter setting.
5. Choose **Next** twice.
6. Under **Capabilities**, select the check box to acknowledge that the template will create IAM resources, and then choose **Update**.

When the status is **UPDATE_COMPLETE**, the stack has been updated with the webhook secret you specified for security.

Step 3. Configure Your Git Repository

After you have successfully deployed the Quick Start, you can configure the service that will use the S3 object as a source. Figure 4 shows the **Outputs** tab in the AWS CloudFormation console, which displays the outputs for configuring your Git webhook.



Key	Value
GitPullWebHookApi	https://[redacted].execute-api.us-east-1.amazonaws.com/Prod/gitpull
OutputBucketName	[redacted]
ZipDownloadWebHookApi	https://[redacted].execute-api.us-east-1.amazonaws.com/Prod/zipdl
PublicSSHKey	[redacted]

Figure 4: Outputs tab after deployment

- **GitPullWebHookApi** is the webhook endpoint to use if you opt for the Git pull method described in the Webhook Endpoints section of this guide.
- **ZipDownloadWebHookApi** is the webhook endpoint to use if you opt for the zip download method described in the Webhook Endpoints section of this 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 4. Configure an AWS Service to Connect to the S3 Object

After you have successfully deployed the Quick Start, you can configure the desired service to use the S3 object as a source. As previously illustrated in Figure 4, the **Outputs** tab in the AWS CloudFormation console includes the **OutputBucketName** output. This output is an S3 key that forms the base of the path to your code zip file. The S3 key is in this 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.

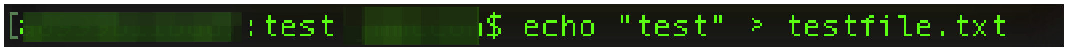
The exact process for linking an AWS service differs from service to service. For step-by-step instructions, consult the service documentation. For easy reference, here are links for the two typical services:

- AWS CodePipeline – Simple pipeline tutorial
- AWS CodeBuild – Walkthrough for creating deployable source code

Step 5. Test a Commit

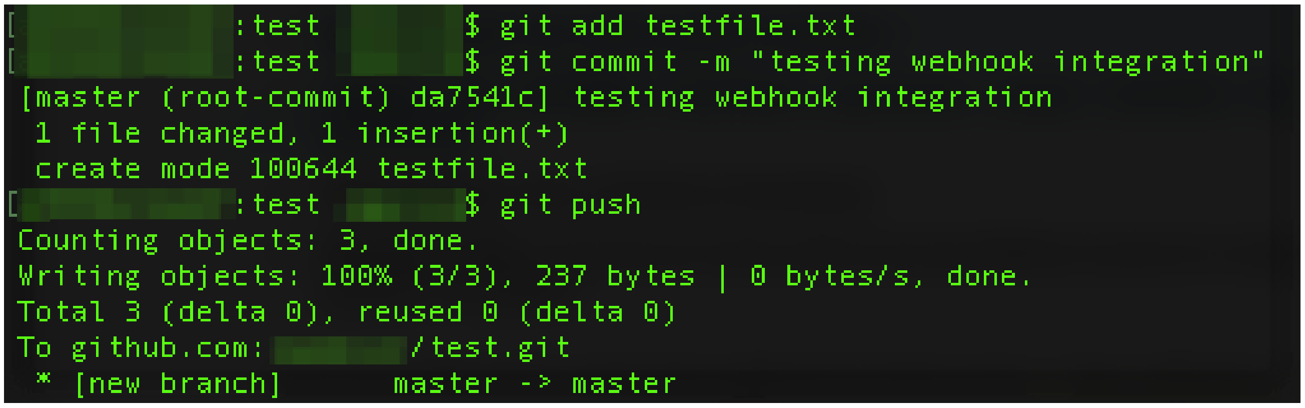
Before putting the webhook into production, you should test your deployment.

1. Modify a file in your repository.



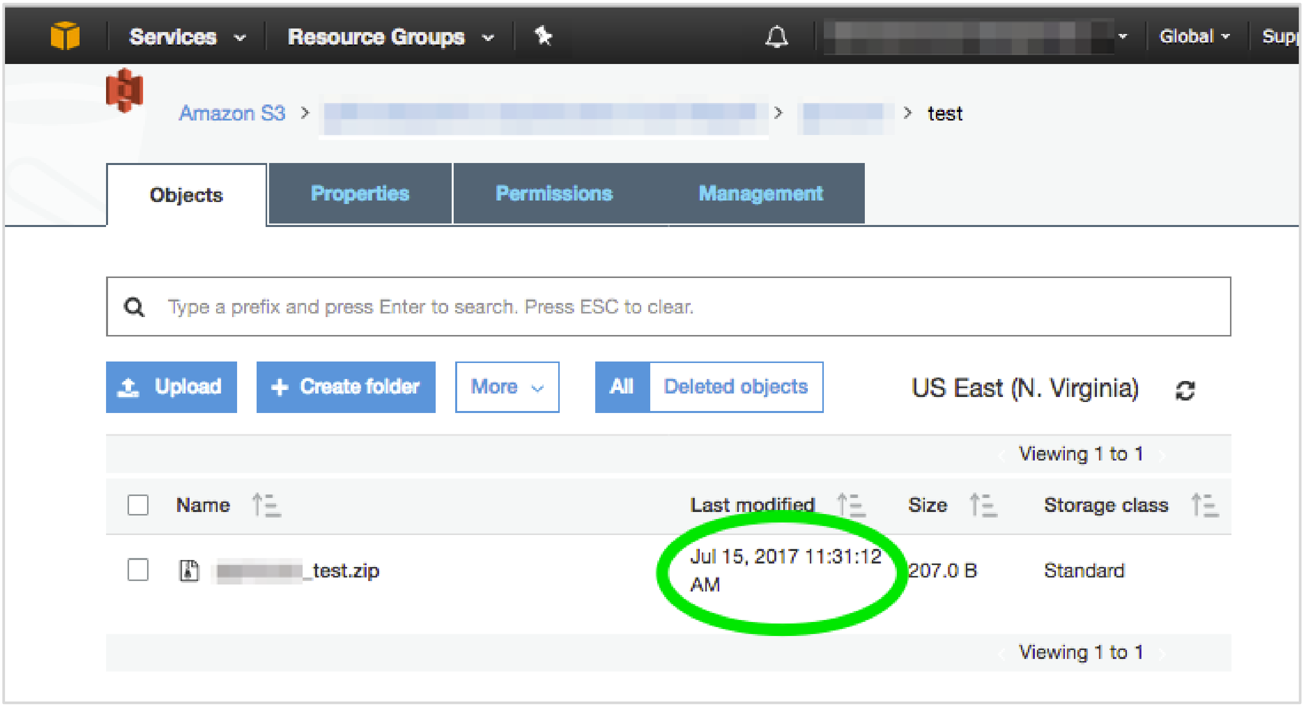
```
[[redacted]:test [redacted]$ echo "test" > testfile.txt
```

2. Commit and push the changes.



```
[[redacted]:test [redacted]$ git add testfile.txt
[[redacted]:test [redacted]$ git commit -m "testing webhook integration"
[master (root-commit) da7541c] testing webhook integration
 1 file changed, 1 insertion(+)
 create mode 100644 testfile.txt
[[redacted]:test [redacted]$ git push
Counting objects: 3, done.
Writing objects: 100% (3/3), 237 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:[redacted]/test.git
 * [new branch]      master -> master
```

3. Wait a few minutes and check your S3 bucket for a new (or updated) object with a key that matches your repository path.



Name	Last modified	Size	Storage class
_test.zip	Jul 15, 2017 11:31:12 AM	207.0 B	Standard

Figure 5: Checking for S3 bucket updates after a commit

Troubleshooting

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. (You will want to look at the Lambda console's **Monitoring** tab.)

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

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

- In your Git service's 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 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.

Additional Resources

AWS services

- AWS CloudFormation
  https://aws.amazon.com/documentation/cloudformation/
- AWS Lambda
  https://aws.amazon.com/documentation/lambda/
- Amazon API Gateway
  https://aws.amazon.com/documentation/apigateway/
- Amazon S3
  https://aws.amazon.com/documentation/s3/
- AWS CodePipeline
  https://aws.amazon.com/documentation/codepipeline/
- AWS CodeBuild
  https://aws.amazon.com/documentation/codebuild/

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 reference deployments

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

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.

Document Revisions

Date	Change	In sections
September 2017	Initial publication	—