Serverless CI/CD for the Enterprise on the AWS Cloud

Quick Start Reference Deployment

February 2020

Andy Warzon, Forrest Brazeal, and Charlie Guse, Trek10
Jay Yeras and Jay McConnell, Amazon Web Services

Visit our GitHub repository for source files and to post feedback, report bugs, or submit feature ideas for this Quick Start.

Contents

This Quick Start was created by Trek10 in collaboration with Amazon Web Services (AWS). Trek10 is an AWS Partner Network (APN) Partner.

Quick Starts are automated reference deployments that use AWS CloudFormation templates to deploy key technologies on AWS, following AWS best practices.

Overview

This Quick Start reference deployment guide provides step-by-step instructions for deploying serverless CI/CD (continuous integration and continuous delivery) for the enterprise on the AWS Cloud.

This Quick Start is for users who want to get started in developing serverless applications on AWS, need an enterprise-ready deployment pipeline, and want to reduce the undifferentiated heavy lifting associated with building, managing, and securing their own, custom-built deployment pipeline. To learn more about serverless deployment, see the Trek10 website.

Enterprises that are ready to embrace serverless application development will get the most benefit from this Quick Start reference deployment. Engineers, team leads, and executives who are already familiar with DevSecOps best practices can take advantage of the improved process flow, decreased time to market, increased security and accountability, and lowered total cost of ownership (TCO) provided by the Quick Start architecture.

Serverless CI/CD for enterprises on AWS

Serverless workflows involve cloud-based builds and deployments. This is a change from traditional development processes and can feel uncomfortable and unfamiliar for many developers. This Quick Start eases the transition to the cloud by automating AWS Lambda-ready builds and deployments without the need for manual code packaging.

Serverless collaboration can be challenging, so this Quick Start uses dynamic pipelines that test code automatically whenever changes are pushed to a feature branch. This approach, based on GitFlow, enables multiple developers within a team to test and deploy their serverless applications in the cloud simultaneously, without stepping on each other's work.

The Quick Start uses several AWS services to enable multiple development teams within an organization to collaborate securely and efficiently on serverless application deployments. It uses AWS services to provide artifact storage, parameter management, automated testing, builds, deployment, and change management for serverless applications across multiple AWS accounts. The basic pipeline can be augmented with additional deployment, testing, or approval steps based on enterprise requirements. The Quick Start also includes a minimal sample serverless application that uses the AWS Serverless Application Model (AWS SAM).

After deploying the Quick Start, you should be able to:

- Push your serverless application code and configuration securely to a source control repository.
- Automatically build and test your serverless application in an AWS development or staging environment.
- Perform automated or manual validation and approval of the code changes.
- Securely manage deployment parameters.
- Deploy your serverless application in a production environment.

Core concepts

This Quick Start adheres to several core concepts of DevOps:

- Repeatability: The application deployment process is fully automated, and all components of the build and deployment pipeline are code-defined. Additionally, when a feature branch is created, the creation and deletion of all deployment infrastructure for that feature branch is fully automated.

- Immutability: The build artifacts are generated once and are then deployed to staging and production. AWS CloudFormation also manages deployment with built-in rollback.
- Security: All AWS Identity and Access Management (IAM) roles are code-defined and scoped to least privilege. Separate AWS accounts are used for the development, shared services, and production environments to provide a security barrier and to limit the blast radius. AWS Key Management Service (AWS KMS) is used to encrypt artifacts. Sensitive application values are stored in AWS Secrets Manager. An example of retrieving those sensitive values at runtime is included in the sample application.
- Low overhead: All systems created in this Quick Start are pay-per-use, managed AWS services. There are no virtual machines to secure or patch, and no capacity to right-size or maintain.

Deployment frameworks

The AWS Serverless Application Model (AWS SAM) is an open-source framework that builds on top of AWS CloudFormation to help you build and deploy serverless applications on AWS. It includes the ability to define common serverless components more concisely in YAML, supports local testing, and provides a command-line interface (CLI) for building and deploying your serverless applications.

In this Quick Start, the sample project is defined in an AWS SAM template, and an AWS SAM package is transformed into an AWS CloudFormation template for the final deployment. However, there are other framework options for defining and deploying serverless applications, and you can adapt this Quick Start to use other frameworks. For example:

- Serverless Framework is a widely adopted toolkit for deploying serverless applications. It has a plug-in ecosystem and multi-cloud capabilities.
- Apex supports a few unique capabilities, such as shimming in applications in other languages, and can deploy an existing Express.js application to AWS Lambda.

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.

The AWS CloudFormation template for this Quick Start includes configuration parameters that you can customize. For cost estimates, see the pricing pages for each AWS service you will be using. Prices are subject to change.

Tip After you deploy the Quick Start, we recommend that you enable the AWS Cost and Usage Report to track costs associated with the Quick Start. This report delivers billing metrics to an S3 bucket in your account. It provides cost estimates based on usage throughout each month, and finalizes the data at the end of the month. For more information about the report, see the AWS documentation.

Architecture

Deploying this Quick Start builds the serverless CI/CD environment illustrated in Figure 1 in the AWS Cloud.

This Quick Start follows AWS multi-account best practices for isolation of resources. After you prepare separate AWS accounts for development, production, and shared services (as explained in step 1 of the deployment section), this Quick Start sets up the following:

- IAM users, roles, and groups in your AWS development, production, and shared services accounts to control access to pipeline actions and deployed resources.
- Dynamic branch pipelines for deploying and testing new feature code in Git branches, using AWS CodePipeline.
- A master code pipeline that deploys to multiple AWS accounts, using AWS CodePipeline.
- Amazon Simple Storage Service (Amazon S3) buckets for pipeline artifacts.
- AWS Secrets Manager to store sensitive configuration data in a central location.
- An AWS CodeCommit repository for storing application code.
- AWS CodeBuild and AWS CodeDeploy configurations for building, deploying, and testing serverless applications.
- A sample serverless application that uses AWS Lambda, Amazon API Gateway, and Amazon DynamoDB.
- Integration with other Amazon services such as AWS Lambda, AWS Key Management Service (AWS KMS), and Amazon Simple Notification Service (Amazon SNS).

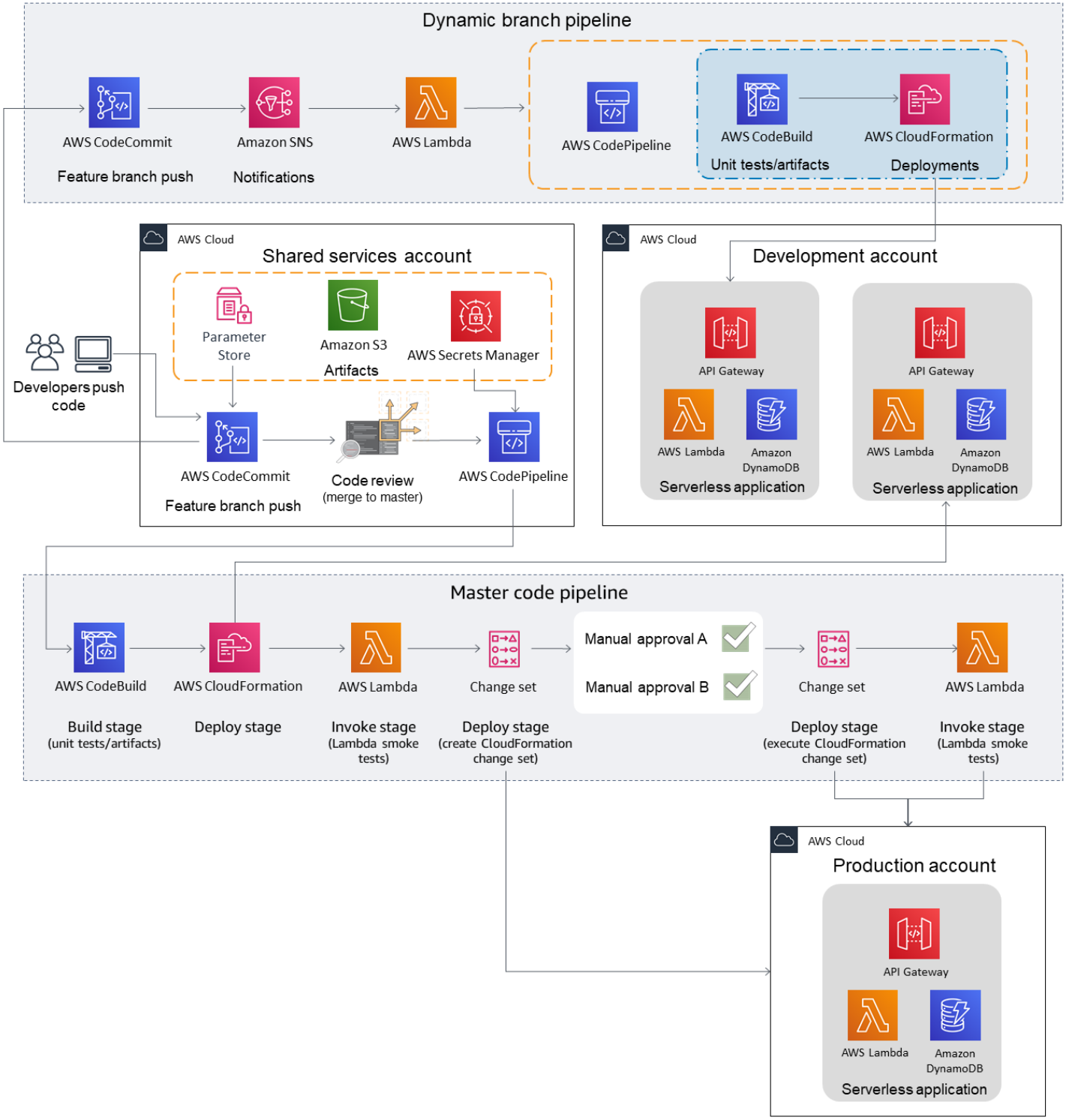

Figure 1: Quick Start architecture for serverless CI/CD on AWS

Planning the deployment

Specialized knowledge

This Quick Start assumes familiarity with the following concepts:

- Version control concepts, using Git or another distributed source code management tool.
- CI/CD best practices, including automated testing and code promotion.
- Serverless concepts and services, including AWS Lambda and AWS SAM.

This deployment guide also requires a moderate level of familiarity with these AWS services: CodePipeline, CodeCommit, CodeDeploy, Amazon S3, and IAM. If you're new to AWS, visit the Getting Started Resource Center and the AWS Training and Certification website for materials and programs that can help you develop the skills to design, deploy, and operate your infrastructure and applications on the AWS Cloud.

Technical requirements

Before you launch the Quick Start, your account must be configured as specified in the following table. Otherwise, deployment might fail.

AWS accounts	This Quick Start follows AWS multi-account best practices for isolation of resources. You must have three distinct AWS accounts ready to use: • Shared services account: This account hosts the core deployment infrastructure and application source code. • Development account: A staging environment for developing your application code is deployed into this account. • Production account: The final production code for your application is deployed into this account. To prepare these accounts, follow the instructions in step 1 of the deployment steps.
Resources	If necessary, request service limit increases for the following resources. You might need to do this if you already have an existing deployment that uses these resources, and you think you might exceed the default limits with this deployment. For default limits, see the AWS documentation. AWS Trusted Advisor offers a service limits check that displays your usage and limits for some aspects of some services.

Resource	This deployment uses
IAM roles	6 in the shared services account, and 4 each in the production and development accounts.

Regions	This deployment includes AWS CodePipeline, AWS CodeBuild, AWS CodeCommit, and AWS Secrets Manager, which aren't currently supported in all AWS Regions. For a current list of supported regions, see AWS Regions and Endpoints in the AWS documentation.
IAM permissions	To deploy the Quick Start, you must log in to the AWS Management Console with IAM permissions for the resources and actions the templates will deploy. The *AdministratorAccess* managed policy within IAM provides sufficient permissions, although your organization may choose to use a custom policy with more restrictions.

Deployment steps

Step 1. Prepare your AWS accounts

1. If you don't already have an AWS account, create one at https://aws.amazon.com by following the on-screen instructions. This will be your master AWS account.

   Your AWS account is automatically signed up for all AWS services. You are charged only for the services you use.

2. Set up sub-accounts for shared services, development, and production:

   a. Sign in to your AWS account at https://aws.amazon.com.

   b. Open the AWS Organizations console at https://console.aws.amazon.com/organizations/.

   c. Follow the instructions in the AWS documentation to create an organization.

   d. Follow the instructions in the AWS documentation to create three sub-accounts: shared services, development, and production. If you have a large development organization, consider creating separate sets of these three accounts for each business unit or logical grouping of applications.

3. Save the AWS account IDs for all three accounts (shared services, development, and production). You will use these in a later step. For additional information, see Finding Your AWS Account ID in the AWS documentation.

Step 2. Enable cross-account access

> **Notes** The instructions in this section reflect the older version of the AWS CloudFormation console. If you're using the redesigned console, some of the user interface elements might be different.
>
> 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. For full details, see the pricing pages for each AWS service you will be using in this Quick Start. Prices are subject to change.

You need to create an IAM role in the development and production accounts to enable cross-account access from the shared services account. The Quick Start includes an AWS CloudFormation template that automatically creates this role for you.

IN THE DEVELOPMENT ACCOUNT

1. Sign in to the development account as a user with IAM permission to create a CloudFormation stack and an IAM role.
2. Launch the AWS CloudFormation template for cross-account access.

   The deployment takes about 2 minutes to complete.

3. Check the AWS Region that's displayed in the upper-right corner of the navigation bar, and change it if necessary. The template is launched in the US East (N. Virginia) Region by default.

> **Note** This deployment includes CodePipeline, CodeBuild, CodeCommit, and Secrets Manager, which aren't currently supported in all AWS Regions. For a current list of supported regions, see the AWS Regions and Endpoints webpage.

4. On the **Select Template** page, keep the default setting for the template URL, and then choose **Next**.

5. 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.

   View template

Parameter label (name)	Default	Description
Shared services account ID (`CentralAwsAccountId`)	*Requires input*	The AWS account ID of the shared services account, from step 1.3. For guidance, see Finding Your AWS Account ID in the AWS documentation.
Child account role name (`ChildAccountRoleName`)	ChildAccountRole	The name of the role to create in the development account. This name must be unique in the development account.

   When you finish reviewing and customizing the parameters, choose **Next**.

6. 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**.
7. On the **Review** page, review and confirm the template settings. Under **Capabilities**, select the two check boxes to acknowledge that the template will create IAM resources and that it might require the capability to auto-expand macros.
8. Choose **Create** to deploy the stack.
9. Monitor the status of the stack. When the status is **CREATE_COMPLETE**, the deployment is done.
10. Sign out of the development account.

IN THE PRODUCTION ACCOUNT

1. Sign in to the production account as a user with IAM permissions to create a CloudFormation stack and an IAM role.
2. Check the AWS Region that's displayed in the upper-right corner of the navigation bar. If necessary, change it to the AWS Region you used in the development account.
3. Launch the AWS CloudFormation template for cross-account access.

The deployment takes about 2 minutes to complete.

4. On the **Select Template** page, keep the default setting for the template URL, and then choose **Next**.
5. 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.

   View template

Parameter label (name)	Default	Description
Shared services account ID (`CentralAwsAccountId`)	*Requires input*	The AWS account ID of the shared services account, from step 1.3. For guidance, see Finding Your AWS Account ID in the AWS documentation.
Child account role name (`ChildAccountRoleName`)	ChildAccountRole	The name of the role to create in the production account. This name must be unique in the production account.

   When you finish reviewing and customizing the parameters, choose **Next**.

6. 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**.
7. On the **Review** page, review and confirm the template settings. Under **Capabilities**, select the two check boxes to acknowledge that the template will create IAM resources and that it might require the capability to auto-expand macros.
8. Choose **Create** to deploy the stack.
9. Monitor the status of the stack. When the status is **CREATE_COMPLETE**, the deployment is complete.
10. Sign out of the production account.

Step 3. Deploy resources

1. Sign in to the shared services AWS account as a user with IAM permissions to create resources in several AWS services. We recommend using the *AdministratorAccess* managed policy.
2. Check the AWS Region that's displayed in the upper-right corner of the navigation bar. If necessary, change it to the AWS Region you used in step 2.
3. Launch the AWS CloudFormation template to deploy resources across all three accounts.

The deployment takes 10-15 minutes to complete.

4. On the **Select Template** page, keep the default setting for the template URL, and then choose **Next**.
5. On the **Specify Details** page, change the stack name if needed. Review the parameters (described in the following tables) 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**.

   View template

Application configuration:

Parameter label (name)	Default	Description
Application name (`AppName`)	Sample	The application name to use for the repository and child stack name.

Accounts configuration:

Parameter label (name)	Default	Description
Development account ID (child) (`DevAwsAccountId`)	*Requires input*	The AWS account ID of the development account, from step 1.3. For guidance, see Finding Your AWS Account ID in the AWS documentation.
Development account role name	ChildAccountRole	The name of the role created by the template in the development account. Use the value for

Parameter label (name)	Default	Description
(DevChildAccountRoleName)		**ChildAccountRoleName** that you specified for the development account in step 2.
Production account ID (child) (ProdAwsAccountId)	*Requires input*	The AWS account ID of the production account, from step 1.3. For guidance, see Finding Your AWS Account ID in the AWS documentation.
Production account role name (ProdChildAccountRoleName)	ChildAccountRole	The name of the role created by the template in the production account. Use the value for **ChildAccountRoleName** that you specified for the production account in step 2.

Pipeline configuration:

Parameter label (name)	Default	Description
Build image (BuildImageName)	aws/codebuild/nodejs:10.1.0	The Docker image for the sample application build. Keep the default setting to try the CI/CD environment out with the sample application that is included with this Quick Start. To customize the environment to your own needs, specify any CodeBuild-managed Docker image or your own Docker image. For additional information, see the Build specification section later in this guide.

AWS Quick Start configuration:

Note We recommend that you keep the default settings for the following two parameters, unless you are customizing the Quick Start templates for your own deployment projects. Changing the settings of these parameters will automatically update code references to point to a new Quick Start location. For additional details, see the AWS Quick Start Contributor's Guide.

Parameter label (name)	Default	Description
Quick Start S3 bucket name (QSS3BucketName)	aws-quickstart	The S3 bucket you 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-trek10-serverless-enterprise-cicd/	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.

6. 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**.
7. On the **Review** page, review and confirm the template settings. Under **Capabilities**, select the two check boxes to acknowledge that the template will create IAM resources and that it might require the capability to auto-expand macros.
8. Choose **Create** to deploy the stack.
9. Monitor the status of the stack. When the status is **CREATE_COMPLETE**, the deployment is done.
10. Sign out of the shared services account.

Step 4. Start a CI/CD pipeline with sample application code

The `/sample-project` folder of the GitHub repository for this Quick Start includes a sample application, which enables you to easily test the CI/CD environment set up by the Quick Start and gives you a reference point for building your own serverless applications with the AWS SAM. This application is a miniature microservice that uses Amazon API Gateway, an AWS Lambda function in node.js, and Amazon DynamoDB to expose a REST interface and simple key-value data store.

There are multiple ways to configure your environment to push code to AWS CodeCommit. If you would like to configure this manually, review the CodeCommit User Guide to determine the the appropriate setup for your situation, and then commit the code in `/sample-project` to the master Git branch to trigger the pipeline.

If you would like to configure your environment quickly, follow these steps:

1. Install the AWS Command Line Interface (AWS CLI) in your environment and configure it to authenticate to your AWS shared services account. For information about installing the AWS CLI, see the AWS documentation.
2. Confirm that you have Git and **ssh-keygen** installed in your environment.
3. In the GitHub repository for this Quick Start, open the `scripts/deployment-variables.sh` file and make sure that the variables `STACK_NAME` and `LOCAL_REPO_FOLDER` are set correctly. (The other variables in that script file are not used in the git setup, they are used for a different, optional script.)
4. Navigate to the root of the cloned repository and run `bash scripts/git-setup.sh`, which will:
    a. Create an SSH public/private key pair.

b. Upload the public key to AWS so that it is associated with your CodeCommit repository.

c. Create a local directory and clone the (empty) CodeCommit repository locally.

d. Copy the sample application code into this directory, commit it, and push it to AWS CodeCommit.

This will trigger the initial pipeline deployment.

5. Navigate to CodePipeline in your AWS shared services account to view the pipeline steps and progress.

Customizing and extending the Quick Start

Project structure and larger team scale-up

This Quick Start is meant to be a starting point for larger organizations. Although it implements a deployment pipeline for only a single application, it can be adapted to a larger set of applications and repeated to scale to many DevOps teams across a larger enterprise. To understand this, let's take a closer look at the project structure.

In the root of the GitHub repository for this Quick Start, the file `ChildAccountRole.template.yaml` is used to minimally configure the child accounts (development and production) to allow access from the shared services account and to allow the automation to complete the setup in subsequent steps.

In the `/templates` folder, the master template `full-stack.template.yaml` nests several sub-templates, which are also in that folder. The master template uses the AWS CLI command **cloudformation deploy** to package all those templates and associated Lambda functions, and to create the infrastructure.

This Quick Start contains all the infrastructure for one organizational unit in a larger organization that would manage some logical set of applications.

- The shared services account-level infrastructure is defined in `account.template.yaml` and deployed once for that given account and team.
- Similarly, the template `cross-account.template.yaml` prepares the child accounts further and is used once for that set of development and production accounts.
- `Project.template.yaml` and `pipeline.template.yaml` are deployed once per project or application. Although this Quick Start includes only one application, it would typically be used to manage some logical set of applications through a single shared services

account and development/production accounts. In this case, `project.template.yaml` and `pipeline.template.yaml` would be used to launch multiple stacks, one for each project or application.

Build specification

In the sample application included with this Quick Start, the file `buildspec.build.yaml` defines the details of the build operation. In this simple node.js scenario, dependencies are installed before the AWS SAM is used to package up the final deployment artifact. Customizing the dependencies section will be a key part of applying this Quick Start to your use case: For example, you might include a build process for a compiled language or install Python dependencies.

To get the build environment you need, you can select from the Docker images managed by AWS for CodeBuild or bring your own Docker image. The image is defined with the **Build image** (`BuildImageName`) parameter in `/templates/full-stack.template.yaml` and configured in step 3 of this guide.

Automated testing

Automated testing is another core DevOps concept that this Quick Start fully supports through AWS CodeBuild. You can add unit test commands to the build specification file, as described in the previous section.

This Quick Start includes an additional smoke test step in the deployment pipeline, as an example of other types of automated testing that can be done. The smoke test is defined in a CodeBuild build specification file (`buildspec.smoketest.yaml` in the sample project directory), which can be customized with any Docker image and commands to meet your automated testing requirements.

Secrets management

AWS Secrets Manager is a fully managed service for storing, accessing, and rotating application secrets with fine-grained access control. This Quick Start:

- Creates a sample secret for your application in the `project.template.yaml` file in the `/templates` folder of the GitHub repository.
- Creates an AWS KMS encryption key for encrypting the secret.
- Defines fine-grained IAM policies for accessing the secret and for using the KMS key to decrypt it.

- Includes example code for retrieving that secret during the AWS Lambda runtime, in the sample project (`sample-project/api/handler.js`).

This secret can be a single string, or it can be a set of key-value pairs or even arbitrary JSON. If you want to have multiple secrets but still easily manage fine-grained permissions, you can use secret name-spacing and wildcards. For example, with secrets named `app1/secret1` and `app1/secret2`, the application can be given permission to access `app1/*` secrets.

Troubleshooting

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

A. If AWS CloudFormation fails to create the stack, 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 and the instance will be left running, so you can troubleshoot the issue.

> **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.

For additional information, see Troubleshooting AWS CloudFormation on the AWS website.

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.

For further assistance

In addition to this Quick Start guide, Trek10 offers architectural guidance, engineering, and 24/7 operational support for AWS. If you are interested in a further engagement with Trek10 to deploy and manage your serverless application infrastructure, check out the Serverless Developer Acceleration offering from Trek10, or contact Trek10 at https://www.trek10.com/contact.

Send us feedback

To post feedback, submit feature ideas, or report bugs, use the **Issues** section of the GitHub repository for this Quick Start. If you'd like to submit code, please review the Quick Start Contributor's Guide.

Additional resources

AWS resources

- Getting Started Resource Center
- AWS General Reference
- AWS Glossary

AWS services

- AWS CloudFormation
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- AWS KMS
- AWS Secrets Manager

Serverless application development

- Introduction to serverless computing
- AWS Serverless Application Model
- AWS Serverless Application Repository
- Additional example AWS SAM apps

Other Quick Start reference deployments

- AWS Quick Start home page

Document revisions

Date	Change	In sections
February 2020	Instructions for script files	Step 4

Date	Change	In sections
April 2019	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.