Customizations for AWS Control Tower

AWS Implementation Guide

Lalit Grover

Aijun Peng

January 2020

Last updated: October 2020 (see Revisions)

Contents

About This Guide

This implementation guide discusses architectural considerations and configuration steps for deploying the Customizations for AWS Control Tower solution in the Amazon Web Services (AWS) Cloud. It includes a link to an AWS CloudFormation template that launches, configures, and runs the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects and developers who have practical experience architecting in the AWS Cloud.

Overview

Amazon Web Services (AWS) offers a diverse array of services and features that allow for flexible control of AWS cloud computing resources and the associated AWS account(s) that manage them. Given the large number of design choices available, the manual process of setting up and configuring a multi-account AWS environment can be a time-consuming task. AWS Control Tower helps customers set up a landing zone in their AWS accounts based on best practices for ongoing governance over AWS workloads. Before deploying this solution, customers must have an AWS Control Tower landing zone deployed in their account.

The Customizations for AWS Control Tower solution combines AWS Control Tower and other highly-available, trusted AWS services to help customers quickly set up a secure, multi-account AWS environment using AWS best practices. This solution enables customers to easily add customizations to their AWS Control Tower landing zone using an AWS CloudFormation template and service control policies (SCPs). You can deploy the custom template and policies to individual accounts and organizational units (OUs) within your organization. This solution integrates with AWS Control Tower lifecycle events to ensure that resource deployments stay in sync with the customer's landing zone. For example, when a new account is created using the AWS Control Tower account factory, the solution ensures that all resources attached to the account's OUs will be automatically deployed.

Cost

You are responsible for the cost of the AWS services used while running this solution. As of the date of publication, the cost for running this solution depends on the number of AWS CodePipeline executions, the duration of AWS CodeBuild runs, the number and duration of AWS Lambda functions, and the number of Amazon EventBridge events published. For example, if you execute 100 builds in one month using **build.general1.small** where each build runs for five minutes, then the approximate cost for running this solution is **$3.00 per month**. For full details, refer to the pricing webpage for each AWS service you will be using in this solution.

The Amazon Simple Storage Service (Amazon S3) bucket and AWS CodeCommit Git-based repository resources are retained after the solution template is deleted to protect the customer configuration. Depending on the option selected, you are charged based on the amount of data stored in the S3 bucket and the number of Git requests (not applicable to Amazon S3 resource). Refer to Amazon S3 and AWS CodeCommit pricing for details.

Architecture Overview

Deploying this solution builds the following environment in the AWS Cloud.

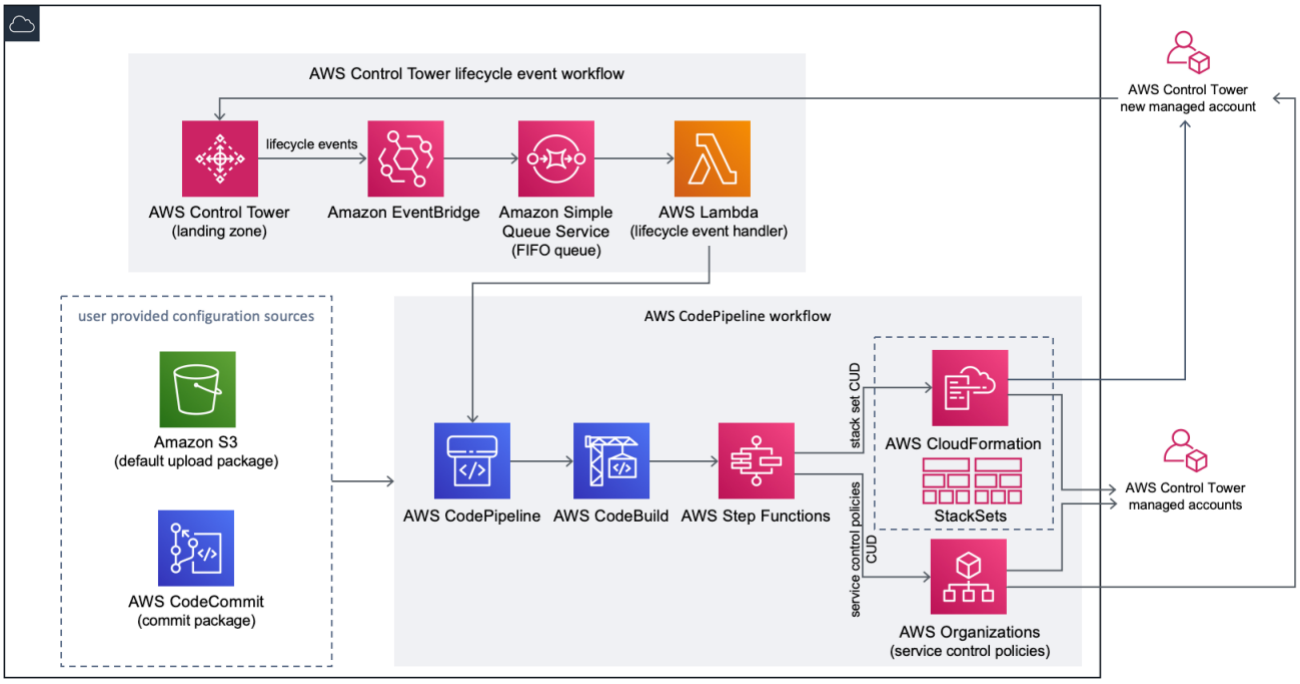

Figure 1: Customizations for AWS Control Tower architecture

This solution includes an AWS CloudFormation template you deploy in your AWS account that launches all the components necessary to build the workflows that enable you to customize your AWS Control Tower landing zone.

> **Note:** The solution must be deployed in the same region and account where your AWS Control Tower landing zone is deployed. For information about setting up an AWS Control Tower landing zone, refer to Getting Started with AWS Control Tower in the *AWS Control Tower User Guide*.

Once the solution is deployed, the custom resources are packaged and uploaded to the code pipeline source using Amazon Simple Storage Service (Amazon S3), and triggers the service control policies (SCPs) state machine and the AWS CloudFormation StackSets state machine to deploy the SCPs at the OU level or stack instances at the OU and/or account level.

> **Note:** By default, this solution creates an Amazon S3 bucket to store the pipeline source, but you can change the location to an AWS CodeCommit repository. For more information, refer to Appendix B.

The solution deploys two workflows: an AWS CodePipeline workflow and an AWS Control Tower lifecycle event workflow. The AWS CodePipeline workflow configures AWS CodePipeline, AWS CodeBuild projects, and AWS Step Functions to orchestrate the management of AWS CloudFormation StackSets and SCPs in your organization.

Uploading the configuration package triggers the code pipeline to run the following stages:

- **Build Stage** – validates the contents of the configuration package using AWS CodeBuild.
- **SCP Stage** – triggers the service control policy state machine to make AWS Organizations API calls to create SCPs.
- **AWS CloudFormation Stage** – triggers the stack set state machine to deploy the resources specified in the accounts and/or OUs list provided in the manifest file.

> **Note:** For information about customizing the configuration package, refer to the Customizations for AWS Control Tower Developer Guide.

Each stage in the code pipeline invokes the stack set and SCP step function, and deploys custom stack sets and SCPs to the target individual accounts or the entire organizational units.

When a new managed account is created in AWS Control Tower, the AWS Control Tower lifecycle event triggers the AWS CodePipeline workflow. You can customize the configuration package using this workflow which consists of an Amazon EventBridge event rule, an Amazon Simple Queue Service (Amazon SQS) first-in first-out (FIFO) queue, and an AWS Lambda function. When a matching lifecycle event is detected by the Amazon EventBridge event rule, it passes the event to the Amazon SQS FIFO queue, triggers the AWS Lambda function, and invokes the code pipeline to perform downstream stack sets and the SCPs deployments.

Solution Components

Amazon Simple Storage Service

The solution creates an Amazon Simple Storage Service (Amazon S3) `custom-control-tower-configuration-<account-ID>-<region>` bucket with a sample configuration `_custom-control-tower-configuration.zip` file. This zip file provides a sample manifest and the related sample templates for describing the folder structure and developing a custom configuration package that customizes the AWS Control Tower landing zone environment. The sample manifest identifies the stack sets and service control policies (SCPs) configurations to implement into new and existing accounts. You can use this sample configuration package to develop and upload a custom package that will trigger the solution's

configuration pipeline. For information about customizing the configuration file, refer to the Customizations for AWS Control Tower Developer Guide.

AWS CodeCommit

Based on the customer input in the CloudFormation template, the solution can also create an AWS CodeCommit repository with the sample configuration explained in the Amazon Simple Storage Service section.

Amazon Simple Queue Service

The solution uses an Amazon Simple Queue Service (Amazon SQS) FIFO queue to capture lifecycle events from Amazon EventBridge, and triggers an AWS Lambda function to invoke AWS CodePipeline to deploy AWS CloudFormation StackSets or AWS Organizations SCPs.

AWS CodePipeline

AWS CodePipeline validates, tests, and implements changes based on updates to the configuration package using either the default Amazon S3 bucket or AWS CodeCommit repository. For more information about changing the configuration source control to AWS CodeCommit, refer to Appendix B. The pipeline includes stages to validate and manage the configuration files and templates, core accounts, AWS Organizations service control policies, and AWS CloudFormation StackSets. For more information about the pipeline stages in this solution, refer to the Customizations for AWS Control Tower Developer Guide.

AWS Key Management Service

The solution creates an AWS Key Management Service (AWS KMS) `CustomControlTowerKMSKey` encryption key. This key is used to encrypt objects in the Amazon S3 configuration bucket, Amazon SQS queue, and sensitive parameters in the AWS Systems Manager Parameter Store. By default, only roles provisioned by the Customizations for AWS Control Tower solution have permission to perform encrypt or decrypt operations with this key. Administrators must be added to the `CustomControlTowerKMSKey` policy to access the configuration file, FIFO queue, or Parameter Store `SecureString` values. Automatic key rotation is enabled by default.

AWS Lambda

The solution uses AWS Lambda functions to trigger the installation components during the initial installation and deployment of AWS CloudFormation StackSets or AWS Organizations SCPs during an AWS Control Tower lifecycle event.

AWS Systems Manager Parameter Store

AWS Systems Manager Parameter Store is used to store the solution's configuration parameters. These parameters are used for integrating related configuration templates, such as configuring each account to log AWS CloudTrail data to a centralized Amazon S3 bucket. Additionally, administrators can leverage the Systems Manager Parameter Store to view the solution's inputs and parameters in one centralized location.

Amazon Simple Notification Service

The solution uses Amazon Simple Notification Service (Amazon SNS) topics to publish notifications during the workflow such as pipeline approval. Amazon SNS is launched only when you choose to receive pipeline approval notifications.

Deployment Considerations

Customizations for AWS Control Tower Initial Deployment

The solution must be launched in the same region and account where AWS Control Tower landing zone is deployed. By default, this solution creates and runs the custom configuration package through a configuration pipeline.

Configuration Source

By default, the template creates an Amazon Simple Storage Service (Amazon S3) bucket to store the sample configuration package as a zip file `_custom-control-tower-configuration.zip`. The S3 bucket is version controlled and you can update the configuration package as needed. For information about updating the configuration package, refer to Appendix A.

> **Note:** The sample configuration package filename begins with an underscore (_) so that AWS CodePipeline is not automatically triggered. When you have completed the customization of the configuration package, ensure you upload the `custom-control-tower-configuration.zip` without the underscore (_) in order to trigger the deployment in AWS CodePipeline.

You can change the storage location of the configuration package from the S3 bucket to an AWS CodeCommit Git repository by selecting the `AWS CodeCommit` option in the CloudFormation parameter. This option enables you to easily manage version control.

> **Note:** When using the default S3 bucket, the configuration package should be available as a zip file. When using the AWS CodeCommit repository, the configuration package should be placed in the repository without zipping the files. For information

about creating and storing the configuration package in AWS CodeCommit, refer to the Customizations for AWS Control Tower Developer Guide.

You can use the sample configuration package to create your own custom configuration source. When you are ready to deploy your custom configurations, manually upload the configuration package to either the S3 bucket or the AWS CodeCommit repository. The pipeline is automatically triggered when the configuration file is uploaded.

Pipeline Configuration Parameters

The AWS CloudFormation template provides the option to manually approve the deployment of configuration changes. By default, manual approval is disabled. For more information, refer to Step 1. Launch the Stack.

When enabled, the configuration pipeline validates the customizations made to the AWS Control Tower file manifest and templates, then pauses the process until manual approval is granted. Once manual approval is received, the deployment executes the remaining pipeline stages that is needed to implement the Customizations for AWS Control Tower solution.

These parameter can be used to keep the customizations for the AWS Control Tower configuration from executing by rejecting the first attempt to run through the pipeline. It can also be used for manual validation of customizations for the AWS Control Tower configuration changes as a final control before implementation.

Customizations for AWS Control Tower Update

If you have previously deployed the solution, you must update the solution's CloudFormation stack to get the latest version of the solution's framework. For details, refer to Update the Stack.

AWS CloudFormation Template

This solution uses AWS CloudFormation to automate the deployment of the Customizations for AWS Control Tower solution in the AWS Cloud. It includes the following AWS CloudFormation template, which you can download before deployment.

View template

custom-control-tower-initiation.template: This template deploys an AWS CodePipeline, AWS CodeBuild projects, AWS Step Functions, AWS Lambda functions, an Amazon EventBridge event rule, an AWS Simple Queue Service queue, and an Amazon Simple Storage Service bucket with a sample configuration package, but you can also customize the template based on your specific needs.

Automated Deployment

Before you launch the automated deployment, please review the considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 15 minutes

Prerequisites

The solution requires AWS Control Tower to be deployed in your AWS account in the same region and account where AWS Control Tower landing zone is deployed. If you do not have a landing zone set up, refer to Getting Started with AWS Control Tower in the *AWS Control Tower User Guide*.

What We'll Cover

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

Step 1. Launch the Stack

- Launch the AWS CloudFormation template into your AWS account.
- Review the template parameters, and adjust if necessary.

Step 2. Create a Custom Package

- Create a custom configuration package.

Step 1. Launch the Stack

This automated AWS CloudFormation template deploys the Customizations for AWS Control Tower solution in the AWS Cloud.

> **Note**: You are responsible for the cost of the AWS services used while running this solution. Refer to the Cost section for more details. For full details, refer to the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button to the right to launch the `custom-control-tower-initiation` AWS CloudFormation template.
   You can also download the template as a starting point for your own implementation.

   Launch Solution

2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.

> **Note**: This solution must be launched in the same region and account where you deployed AWS Control Tower landing zone. AWS Control Tower is available in specific AWS Regions only. Therefore, you must launch this solution in a Region where this service is available. For the most current service availability by Region, refer to the AWS Control Tower FAQs.

3. On the **Create stack** page, verify that the correct template URL shows in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following parameters.

Pipeline Configuration		
Parameter	**Default**	**Description**
Pipeline Approval Stage	`No`	Choose whether to change the pipeline configuration from the default automated approval stage to a manual approval stage. For more information, refer to the Customizations for AWS Control Tower Developer Guide.
Pipeline Approval Email Address	<Optional Input>	The email address for approval notifications. To use this parameter, you must set the **Pipeline Approval Stage** parameter to `Yes`.
AWS CodePipeline Source	`Amazon S3`	The source for AWS CodePipeline to help customers easily select where they wish to store and configure the solution's customizations.

AWS CodeCommit Setup		
Parameter	**Default**	**Description**
Existing CodeCommit Repository?	`No`	Choose whether to use an existing CodeCommit Git repository. If you choose `Yes`, you must set the **CodePipeline Source** parameter to `AWS CodeCommit`.
CodeCommit Repository Name	`custom-control-tower-configuration`	The Git repository name. To use this parameter, you must set the **AWS CodePipeline Source** parameter to `AWS CodeCommit`. This name is used to create a new Git repository, and must be unique. If you provide the name of an existing Git repository, you must set the **Existing CodeCommit Repository?** parameter to `Yes` and enter the exact name of that repository.
CodeCommit Branch Name	`master`	The Git branch where the customization package is stored. Git repositories can have many branches. This is the default name given to the branch in the Git repository. To use this parameter, you must set the **CodePipeline Source** parameter to `AWS CodeCommit`.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template might create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately 15 minutes.

Step 2. Create a Custom Package

You are now ready to set up a secure, multi-account AWS environment using AWS best practices. Using the launched stack, you can easily add customizations to your AWS Control Tower landing zone and service control policies (SCPs) by customizing the included configuration package. For detailed instructions on creating a custom package, refer to the Customizations for AWS Control Tower Developer Guide.

> **Note:** The pipeline will not execute without uploading the custom configuration package.

Update the Stack

If you have previously deployed the solution, follow this procedure to update the Customizations for AWS Control Tower CloudFormation stack to get the latest version of the solution's framework.

1. Sign in to the AWS Cloudformation console, select your existing Customizations for AWS Control Tower CloudFormation stack, and select **Update**.
2. Select **Replace current template**.
3. Under **Specify template**:
    a. Select **Amazon S3 URL**.
    b. Copy the link of the latest template.
    c. Paste the link in the **Amazon S3 URL** box.
    d. Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**. Choose **Next** again.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. Refer to Step 1. Launch the Stack for details about the parameters.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template might create AWS Identity and Access Management (IAM) resources.
8. Choose **View change set** and verify the changes.
9. Choose **Update stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should see a status of UPDATE_COMPLETE in approximately 15 minutes.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden when AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

AWS Key Management Service

Customizations for AWS Control Tower creates an AWS Key Management Service (AWS KMS) `CustomControlTowerKMSKey` encryption key. This key is used to encrypt objects in the Amazon Simple Storage Service (Amazon S3) configuration bucket, Amazon Simple Queue Service (Amazon SQS) FIFO queue, and sensitive parameters in the AWS Systems Manager Parameter Store.

Additional Resources

- AWS CloudFormation
- AWS Control Tower
- AWS CloudFormation StackSets
- Amazon Simple Storage Service
- AWS CodePipeline
- AWS CodeBuild
- AWS CodeCommit
- Amazon EventBridge
- AWS Lambda
- AWS Step Functions
- Amazon Simple Queue Service
- AWS Systems Manager Parameter Store
- Amazon Simple Notification Service

Appendix A: Using Amazon S3 as the Configuration Source

When the Customizations for AWS Control Tower solution is deployed, an initial configuration `_custom-control-tower-configuration.zip` file is deployed in an Amazon Simple Storage Service (Amazon S3) `custom-control-tower-configuration-<account-ID>-<region>` bucket.

> **Note** If you choose to download and modify this file, zip the changes, save as a new file named *`custom-control-tower-configuration.zip`*, and upload it back to the same S3 bucket.
>
> The S3 bucket is the default source of the pipeline. When default settings are used, uploading a configuration zip file without the underscore prefix in the file name to the S3 bucket will automatically execute the code pipeline configuration updates.

The zip file is protected using Server-Side Encryption (SSE) with AWS Key Management Service (AWS KMS), and denial of use of the KMS key. To access the file, use the following procedure to update the KMS Key Policy with the role(s) that should be granted access, either an administrator role, a user, or both.

1. Navigate to the AWS Key Management Service console.
2. In **Customer Managed Keys**, select **CustomControlTowerKMSKey**.
3. Select the **Key policy** tab. Then, select **Edit**.
4. In the **Edit key policy** page, find the **Allow Use of the key** section in the code, and add the role(s) that should be granted access:
   - To add an administration role:
     `arn:aws:iam::<account-ID>:role/<administrator-role>`
   - To add a user:
     `arn:aws:iam::<account-ID>:user/<username>`
5. Select **Save Changes**.
6. Navigate to the Amazon S3 console, find the S3 bucket containing the configuration zip file, and select download.
7. Make the necessary configuration changes to the manifest file and template files. For information about customizing the manifest and template files, refer to Customizations for AWS Control Tower Developer Guide.

8. Upload your changes:
    a. Zip the modified configuration files, and name the file: `custom-control-tower-configuration.zip`.
    b. Upload the file to Amazon S3 using SSE with the AWS KMS master-key: `CustomControlTowerKMSKey`.

Appendix B: Collection of Operational Metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Timestamp:** Data-collection timestamp
- **State Machine Execution Count:** Incrementally counts the number of times the solution's state machine runs

Note that AWS will own the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, complete one of the following tasks:

- Modify the AWS CloudFormation template mapping section as follows:

```
AnonymousData:
    SendAnonymousData:
       Data: Yes
```

to

```
AnonymousData:
    SendAnonymousData:
        Data: No
```

OR

- After the solution has been deployed, find the `/org/primary/metrics_flag` SSM parameter key in the Parameter Store console and set the value to `No`.

Source Code

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

Revisions

Date	Change
January 2020	Initial Release
March 2020	Bug fixes
July 2020	For information about updates and changes for v1.2.0, refer to the CHANGELOG.md file in the GitHub repository.
October 2020	Added instructions for updating the solution CloudFormation stack and bug fixes. For information about updates and changes for v1.2.1, refer to the CHANGELOG.md file in the GitHub repository.