­­

Nextgen cloud

Insight Enterprises

Contents

[Purpose 2](#_Toc98860543)

[Service Catalog Repository Structure 2](#_Toc98860544)

[Tfvars Repository Structure 7](#_Toc98860545)

[Triggering Pipeline in the ADO 8](#_Toc98860546)

[Terraform State File Management 9](#_Toc98860547)

[Conclusion 10](#_Toc98860548)

# Purpose

This document provides an overview of the repositories and the pipelines structure that are present in the project. This document also provides instructions on how to trigger the pipelines and explains how terraform state files are managed through the pipelines.

The following image represents the project repository structure.

Graphical user interface, text, application, email

Description automatically generated

There are three repositories in the project:

|  |  |
| --- | --- |
| Repository Name | Description |
| service\_catalog | This repository contains three folders (i.e., terraform, pipelines, and templates).   * **Terraform** folder contains the codes for all the service catalog items. * **Pipelines** folder contains the pipeline codes for all the service catalogs. * **Templates** folder contains pipeline code for deploying and destroying the resources in single or multiple subscription. |
| tfvars | This repository contains the tfvars and yaml files which are needed to trigger the pipelines for all the service catalogs. |
| reference | This repository contains the updated tfvars files for all the service catalogs. |

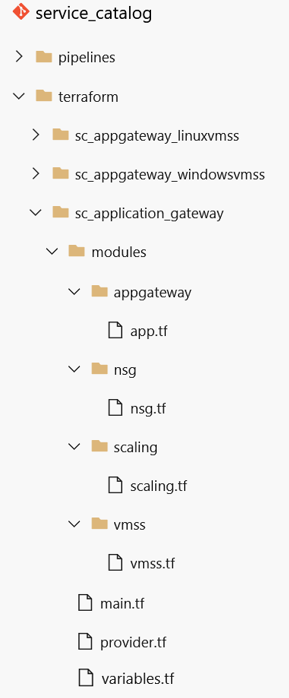
# Service Catalog Repository Structure

The following image represents the service catalog repository structure:

Graphical user interface, text, application

Description automatically generated

The **Terraform** folder contains multiple sub-folders containing terraform codes for all the service catalog items. The following image represents the file structure of the application gateway, which is one of the service catalog folders present in terraform folder.

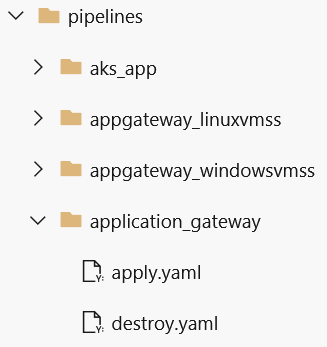


In **sc\_application\_gateway,** there is a sub folder called **modules,** which contains multiple sub-folders containing a terraform file which are required to build application gateway resources. The **main.tf** file runs the files present in the modules folder. The **variables.tf** file contains the variables that are required to run the main.tf file. The **provider.tf** file contains code snippets for Azure provider and its versions. It also contains code snippets for managing the terraform state files.

The following image represents the code structures of the terraform files

Graphical user interface, text, application

Description automatically generated



The **Pipeline** folder contains multiple sub-folders which contains pipeline codes for the respective service catalog items. Each sub-folder contains two yaml files, **apply.yaml** and **destroy.yaml**. The following image represents the file structure of the Notification Hub, which is one of the sub-folders presents in pipeline folder.

The **apply.yaml** file helps in building up the resources for the specific service catalog item. The following image represents the code structure of the apply.yaml file.

Graphical user interface, application

Description automatically generated

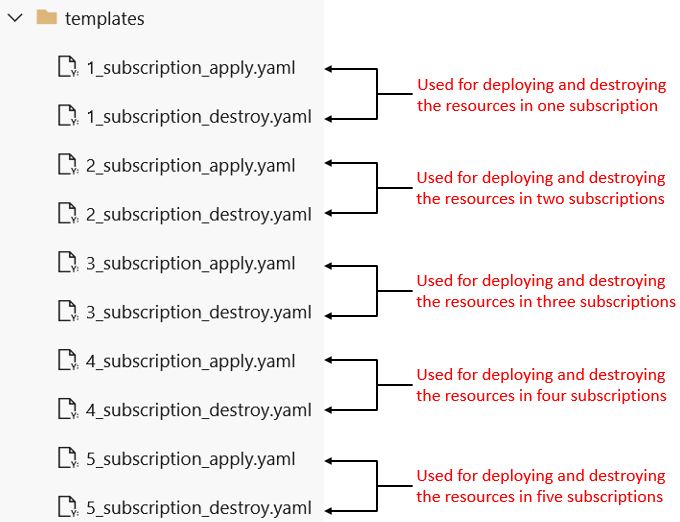
The **destroy.yaml** file helps in destroying up the resources for the specific catalog item. The following image represents the code structure of the destroy.yaml file.

Graphical user interface, chart

Description automatically generated

The **templates** folder contains the pipeline codes for deploying and destroying the resources in single or multiple subscriptions. The following files are present in the templates folder:

* **1\_subscription\_apply.yaml –** Used for deploying the resources in one subscription.
* **1\_subscription\_destroy.yaml –** Used for destroying the resources in one subscription.
* **2\_subscription\_apply.yaml –** Used for deploying the resources in two subscription.
* **2\_subscription\_destroy.yaml –** Used for destroying the resources in two subscription.
* **3\_subscription\_apply.yaml –** Used for deploying the resources in three subscription.
* **3\_subscription\_destroy.yam –** Used for destroying the resources in three subscription.
* **4\_subscription\_apply.yaml –** Used for deploying the resources in four subscription.
* **4\_subscription\_destroy.yaml –** Used for destroying the resources in four subscription.
* **5\_subscription\_apply.yaml –** Used for deploying the resources in five subscription.
* **5\_subscription\_destroy.yaml –** Used for destroying the resources in five subscription.



The following image represent the code structure of the **1\_subscription\_apply.yaml** file.

Table

Description automatically generated

The pipeline code for the **2\_subscription\_apply.yaml, 3\_subscription\_apply.yaml, 4\_subscription\_apply.yaml** and **5\_subscription\_apply.yaml** file are also quite similar but it deploys the resources in multiple subscriptions.

The following image represents the code structure of the **1\_subscription\_destroy.yaml** file.



The pipeline code for the **2\_subscription\_destroy.yaml, 3\_subscription\_destroy.yaml, 4\_subscription\_destroy.yaml** and **5\_subscription\_destroy.yaml** file are also quite similar but it destroys the resources in multiple subscriptions.

# Tfvars Repository Structure

The tfvars repository contains the tfvars and yaml files to trigger the pipeline for the respective service catalog. The following image represents the files structure of one of the service catalog folders present in the tfvars repository.

Diagram

Description automatically generated

The code structure of both the **create.yaml** and **destroy.yaml** is same. The following image represents the code structure of the create.yaml file where the deployment is happening in single subscription.

Chart, line chart

Description automatically generated

The following image represents the code structure of the create.yaml file where the deployment is happening in multiple subscriptions.

Diagram, table

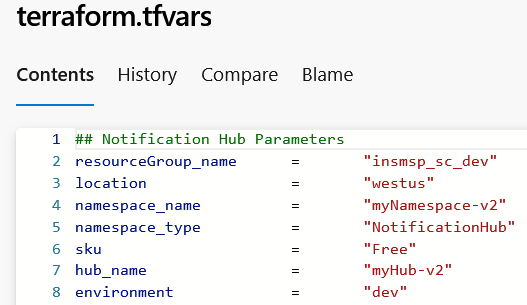
Description automatically generated

**Note:** Maximum value of total\_subscriptions is 5.

# Triggering Pipeline in the ADO

Perform the following steps to trigger the pipeline to deploy/destroy a particular service catalog:

1. Open the Project in the ADO. Navigate to the tfvars repository and open the service catalog folder that needs to be deploy.
2. Open the tfvars file and edit the parameters as per the requirements and commit the changes. The figure below depicts the structure of tfvars file.



1. Open the **create.yaml** file and set the app name, total number of subscriptions and name of these subscriptions as per the requirements and commit the changes. This will trigger the pipeline.
2. **To destroy that service catalog, copy the code from the create.yaml file and paste it in the destroy.yaml file and commit the changes. This will trigger the pipeline.**

**Note: After the successful execution of the pipeline through all the stages, we will receive a notification on the mail as well as on Microsoft Teams regarding the status of the pipeline.**

# Terraform State File Management

Terraform stores state file regarding the managed infrastructure and the configuration. This state file is used by Terraform to map the real-world resources to the configuration, keep track of metadata, and improve performance for large infrastructures. This state file is stored by default in a local file named **terraform.tfstate**  
During the execution of pipelines, **terraform.tfstate** file is stored in the Azure storage account.

**Prerequisites for managing terraform state files:**

* Resource Group
* Resource Group Location
* Storage Account
* Container

The following image represents the code snippet of the **1\_subscription\_apply.yaml** file where terraform state files are managed.

Text

Description automatically generated

All the parameters that are required for the terraform state file management are present in the **Library** which is present in the **Pipeline** section. Also, the terraform state management structure follows the same structure as the tfvars repository as shown in the diagram.

Graphical user interface, application

Description automatically generated

# Conclusion

In this document we discussed on the service catalog repositories and the pipelines structures. We discussed on how to trigger the pipelines to deploy or destroy a particular service catalog. We also discussed on how terraform state files are managed through pipelines.