TerraSnow Enterprise Documentation

Release 0.0.1

Plus3 IT Systems

Aug 27, 2018

Contents

1 Overview 3

2 Contents 5

3 Project Flow Diagrams 21

4 Supported Versions of ServiceNow 25

Enables the deployment of AWS resources from ServiceNow via Terraform Enterprise

CHAPTER 1

Overview

TerraSnow Enterprise is a collection of scripts that enable the deployment of Terraform resources from a ServiceNow instance via Terraform Enterprise. It was designed to simplify cloud resource consumption at the user level and to operate within a multi-tenant AWS environment.

This project contains a terraform template to deploy a Ngnix reverse proxied, Flask based endpoint that handles Gitlab Tag and Push events by creating a ServiceNow Terraform Module Catalog Item.

CHAPTER 2

Contents

2.1 Installation

TerraSnow is maintained in two separate repositories.

The first maintains the TerraSnow Enterprise Instance and backing code as a terraform deployable host. It is maintained in this project's repo under `/scripting_host`.

The second repository hosts the accompanying ServiceNow scoped application.

2.1.1 Overview

1. *Configure ServiceNow*
   (a) *Create the TerraSnow Configuration File*
   (b) *Configure/Deploy your MID Server*
   (c) *Install the TerraSnow Scoped Application*
   (d) *Associate the TerraSnow application with your MID server*
   (e) *Create the TerraSnow API User*
   (f) *Collect the required sys_ids*
2. *Configure Terraform Enterprise*
3. *Upload the TerraSnow Configuration file to S3*
4. *Deploy the TerraSnow Instance*
5. Proceed to the usage section to create your first terraform catalog item

2.1.2 Assumptions

- Working familiarity with terraform module development and terraform based resource deployment within AWS
- Passing understanding of ServiceNow application development
- Familiarity with AWS IAM role assignment and configuration

2.1.3 Requirements

- Admin access to the target AWS account
- The latest version of terraform installed on the machine from which the TerraSnow Instance template will be deployed
- Pre-configured Gitlab instance
- Pre-configured ServiceNow Instance
- Web console access to Terraform Enterprise
- A MID Server ..

  NOTE:

  The MID Server must be deployed in the target AWS account with an instance role that has been granted assume role privileges to the accounts in which terraform resources will be deployed.

  The MID Server must be associated with your ServiceNow instance

  All of these services can be used with TerraSnow Enterprise in their SaaS form(s) with the exception of the ServiceNow MidServer.

2.1.4 Setup

1. Clone the TerraSnow Enterprise repo to your workstation.
2. Create a file called `config.ini` in the root of the `terrasnow-enterprise` directory. See the *Configuration File* section of this document for this file's structure.
3. Fork the TerraSnow Enterprise Scoped Application project into a personal repo.

   NOTE: The scoped application can be loaded into your ServiceNow instance directly from this repo. However, you will be unable to commit any local changes you make unless you follow these steps to point your application at a different repository.

Configure ServiceNow

The following instructions outline the steps required to configure your ServiceNow instance for use with the TerraSnow application.

Create the TerraSnow Configuration File

NOTE: Terraform Enterprise and ServiceNow environment specific details are stored within a configuration file.

These settings are pulled by the TerraSnow Instance automatically and as needed.

This file must be stored in an S3 bucket that is read-accessible by the TerraSnow instance (configurable via the associated AWS Instance Role). Additionally, it is recommended that this file be stored in an encrypted S3 bucket due to its sensitive nature.

The expected file structure is as follows:

File Name: `config.ini`

Contents:

```
[SERVICENOW]
INSTANCE_NAME=
SN_API_USER_NAME=
SN_API_USER_PWD=
TF_CATALOG=
CATEGORY=
TFE_WORKFLOW=
SYS_PACKAGE=

[TERRAFORM_ENTERPRISE]
INSTANCE_NAME=
ATLAS_TOKEN=
```

ServiceNow

Overview of the `config.ini` settings for ServiceNow specific information

Value	Description
INSTANCE_NAME	url of the target ServiceNow instance ex: https://mysninstance.com
SN_API_USER_NAME	user name of the user performing API actions against ServiceNow
SN_API_USER_PWD	password of the user performing API actions against ServiceNow
TF_CATALOG	sys_id of the target Catalog
CATEGORY	sys_id of the target Category
TFE_WORKFLOW	sys_id of the TF catalog item order workflow
SYS_PACKAGE	sys_id of the TerraSnow scoped application

Terraform enterprise

Overview of the `config.ini` settings for Terraform Enterprise specific information

Value	Description
INSTANCE_NAME	url of the target TFE instance ex: https://app.terraform.io
ATLAS_TOKEN	User API access token to create and populate TFE workspaces

Configure/Deploy your MID Server

Deploy a MID server into the target AWS environment. This mid server will be making the API calls against the TerraSnow Instance in order to deploy resources against TerraForm Enterprise.

> **NOTE:**
>
> The MID Server must be deployed in the target AWS account with an instance role that has been granted assume role privileges to the accounts in which terraform resources will be deployed.

The MID Server must be associated with your ServiceNow instance

Install the TerraSnow Scoped Application

Import the TerraSnow scoped application from your personal repo by following the official ServiceNow instructions for Importing applications from source control.

Associate the TerraSnow application with your MID server

1. From your ServiceNow instance navigate to `Service Mapping` > `MID Servers`
2. Select the MID server that was deployed to your target AWS account.
3. Select the `Supported Applications` tab and click `Edit...`
4. Add `terraform-snow` and confirm that it now shows in the `Supported Applications` list

Create the TerraSnow API User

1. Create an account on the ServiceNow instance that has the following roles:

role_name	requirement
admin	place_holder
api_analytics_read	place_holder
catalog_editor	place_holder
catalog	place_holder
catalog_admin	place_holder
credential_admin	place_holder
rest_api_explorer	place_holder
user_criteria_admin	place_holder
web_service_admin	place_holder

1. Add the user name and the user's password to the values for `SN_API_USER_NAME` and `SN_API_USER_PWD` respectively in `config.ini`

   NOTE: You *must* login to your ServiceNow instance with this user at least once and select the `terraform-snow` application scope. If you fail to do so TerraSnow catalog items will be created in the global scope.

Collect the required sys_ids

From the Terraform Resources Catalog

1. Within the TerraSnow Scoped application locate the Terraform Template Catalog (`Terraform Resources Scoped App`)
2. Copy the `sys_id` of the catalog (Retrievable from the sys_id option of the right click context menu in the catalog list view) and update the value of `TF_CATALOG` in `config.ini`
3. Copy the `sys_id` of the terraform resources catalog category (retrievable from the sys_id option of the right click context menu in the catalog Categories tab) and update the value of `CATEGORY` in `config.ini`

From the Terraform Resources Workflow

1. Locate the `terrasnow-enterprise - scoped` workflow within the TerraSnow scoped application
2. From the workflow properties context menu, right click and copy its `sys_id`
3. Update the value of `TFE_WORKFLOW` in `config.ini`

Configure Terraform Enterprise

NOTE: Testing and development was done against Terraform Enterprise using a single Organization.

Generate a user API Token

1. Generate an API token for a Terraform Enterprise user: `TFE console > User Settings > Tokens`
2. Update the value of `ATLAS_TOKEN` in `config.ini`

Upload the configuration file to S3

Requirements:

1. This bucket *must* be private
2. The IAM Instance Role that is assigned to the TerraSnow Instance must have read access to this bucket

Recommendations:

1. Ensure the bucket is encrypted.
2. Configure the bucket with versioning to prevent inadvertent loss of information

Deploy the TerraSnow Instance

This instance will perform all the 'heavy lifting' when it comes to building the catalog item(s) within ServiceNow as well as the Workspace creation within Terraform Enterprise when the catalog item is ordered.

Deployment

NOTE: Successful deployment requires that the environment specific configuration file has been populated with the correct information and uploaded to S3.

1. Navigate to the scripting_host folder and create a `terraform.tfvars` file specific to the target AWS env
2. Configure the local env to target the correct AWS account either via the AWS cli or by modifying the provider block in `main.tf`
3. Run `terraform apply`
4. Proceed to the usage section for catalog item creation and gitlab repo configuration.

API Endpoints

On successful deployment the instance is configured with the following endpoints:

Endpoint	Description
`/`	Sends `200` regardless of content, used for testing
`/aws-assume-role-webhook`	Listens for AWS assume role data, creates the required TFE credential env vars
`/gitlab-webhook`	Listens for tag update events sent from gitlab and creates the associated SN catalog item
`/tfe-run-webhook`	Listens for workflow run events, uploads the source terraform module to the target workspace to trigger a TFE workflow event
`/variables-webhook`	Listens for ServiceNow variables creation requests, sends associated API call to SN to create the variable
`/workflow-webhook`	Listens for TFE workspace creation events, creates an empty workspace

Create your first Terraform Catalog item

See the usage section of this guide for more details.

2.2 Usage

2.2.1 Requirements

- The Installation procedures have been completed successfully.
- A working Gitlab instance

  Notes:

  - Module repositories must meet the same requirements as those outlined for addition to the Terraform Module Private Registry
  - This project was successfully tested against the watchmaker lx-instance module.
  - The source terraform module has a separate `main.tf` and `variables.tf` file. Variables not defined in `variables.tf` will not be included in the resulting ServiceNow catalog item.

2.2.2 ServiceNow TF catalog item creation

1. Create a Gitlab repo with the `terraform-<PROVIDER>-<MODULE_NAME>` name format
2. Add the TerraSnow instance public key to the repo (available at the `https://YOUR_TERRASNOW_INSTANCE/pub-key/key.txt`) and grant the TerraSnow instance read access to the repo.
3. Add a version tag to the project before commit that follows the PEP 440 standard (ex: 1.0.2)
4. Add the TerraSnow instance url as a webhook under `Repo > Settings > Integrations`
   - Select `Tag push events` and `Enable SSL verification`

- Paste in the TerraSnow instance gitlab webhook url: `http://YOUR_TERRASNOW_INSTANCE/gitlab-webhook`
- Click the `Add Webhook` button to complete

5. Kick off the ServiceNow catalog item build process by either manually triggering the webhook or incrementing the project's version tag: `git tag -a v0.0.1 -m 'test' && git push origin --tags`
6. The Terraform resource catalog item should now be available for order via the target ServiceNow instance.

   Note: There is an issue with the ServiceNow catalog item OnLoad client scripts associating with their target variables (see the comments in the embedded client scripts for more details). Unfortunately, this is a manual step for now.

2.2.3 Deploying TF resources from ServiceNow

Navigate to the TerraForm ServiceNow Catalog via either the Terraform service portal `https://YOUR_SNOW_INSTANCE/tfcm` or via the application studio.

On order, the workflow should be triggered and a workspace will be created on the target Terraform Enterprise instance.

Current workspace naming convention is the ServiceNow REQ sys_id but this may be updated in a future release

2.3 Scripting Host

Documentation that outlines the configuration of the terraform deployable scripting host.

2.3.1 Module Input Variables

Variable: `subnet_id` **Description**: The target subnet id for the TerraSnow instance

Variable: `env_type` **Description**: Suffix added to the instance name (dev, test, prod, etc.)

Variable: `alias_name` **Description**: Value used in building the instance name and the instance domain name.

Variable: `target_r53_zone` **Description**: Target route 53 zone in which to build the resulting domain name entry.

Variable: `pub_access_sg` **Description**: The security group within the target AWS account that allows public access.

Variable: `priv_access_vpc_id` **Description**: ID of the VPC that provides private access within the target AWS account.

Variable: `priv_alb_subnets` **Description**: List of subnets that are backed by the private ALB.

Variable: `subnet_id` **Description**: The id of the security group in which to place the instance

Variable: `sg_allow_inbound_from` **Description**: Source security group to allow inbound traffic into the instance's private security group.

Variable: `instance_type` **Description**: AWS instance type (t2.micro, t2.medium, etc.)

Variable: `key_name` **Description**: SSH public key used to login to the TerraSnow instance.

Variable: `instance_role` **Description**: Role to associate with the TerraForm Scripting host instance. Requires read access to the S3 bucket where the TerraSnow configuration file is stored.

Variable: `private_gitlab_server` **Description**: "hostname of the gitlab server. ex: gitlab.mydomain.net. Passed as a variable into the TerraSnow host initialization script. Used to add the gitlab host as a trusted ssh endpoint and enable use of `git clone` via SSH.

2.3.2 Outputs

Variable: `_private_ip` **Value**: IPv4 IP address **Description**: The private IP address of the TerraSnow instance

Variable: `aws_assume_role_webhook` **Value**: https://INSTANCE_FQDN/aws-assume-role-webhook **Description**: The AWS assume role API endpoint of the TerraSnow instance

Variable: `gitlab_webhook` **Value**: https://INSTANCE_FQDN/gitlab-webhook **Description**: The gitlab webhook endpoint of the TerraSnow instance

Variable: `pub_deployment_key` **Value**: https://INSTANCE_FQDN/pub-key/key.txt **Description**: The web accessible path to the public key of the TerraSnow instance. This key is added to the target gitlab repo as a deploy key with read access to enable the TerraSnow instance to successfully `git clone`.

Variable: `tfe_workflow_webhook` **Value**: https://INSTANCE_FQDN/workflow-webhook **Description**: The Terraform Enterprise workspace API endpoint of the TerraSnow instance.

Variable: `sn_variables_webhook` **Value**: http://INSTANCE_FQDN/variables-webook **Description**: The webhook that triggers the ServiceNow catalog item variables.

2.3.3 Overview

The included terraform module will deploy the following resources.

Terraform Enterprise Scripting Host

Description: An EC2 instance of the size of your choosing (via the `instance_type` variable).

Requirements:

- An IAM role that at a minimum has read access to the S3 bucket where the TerraSnow configuration file is stored.
- An AWS environment that has a security group that provides public access. Port 443 is required as all communications done with the TerraSnow api endpoints are over https via the TerraSnow alb.

Application Load balancer

Description: Created via the included `alb` module. An ALB that proxies http connections from the TerraSnow instance to https. Backed by an AWS issued https certificate.

Requirements: A separate public access security group within the target AWS account.

TerraSnow Initialization Script

Description: A bash script that will install and configure the flask application on an EC2 instance.

Requirements: The EC2 instance on which this script is run will require internet access.

2.4 API Reference

2.4.1 TerraSnow API endpoints

Endpoint	Description
`/`	Sends `200` regardless of content, used for testing
`/aws-assume-role-webhook`	Listens for AWS assume role data, creates the required TFE credential env vars
`/gitlab-webhook`	Listens for tag update events sent from gitlab and creates the associated SN catalog item
`/tfe-run-webhook`	Listens for workflow run events, uploads the source terraform module to the target workspace to trigger a TFE workflow event
`/variables-webhook`	Listens for ServiceNow variables creation requests, sends associated API call to SN to create the variable
`/workflow-webhook`	Listens for TFE workspace creation events, creates an empty workspace

Assume Role

Listens for AWS assume role data, and creates the following TFE workspace environment variables:

- AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY (created with `is_senative=True`)
- AWS_DEFAULT_REGION
- AWS_SESSION_TOKEN

Request Syntax

```
{
  "data": [
    {
     "region": "us-east-1",
     "org_name": "MyTFEorg",
     "workspace_name": "ws-123456ASDFhjklmn",
     "role": "arn:aws:iam::0123456789123:role/target_role",
     "duration": "900"
    }
   ]
}
```

Parameters

- region *(string)* – **[REQUIRED]** – Target region for resource creation.
- org_name *(string)* – **[REQUIRED]** – Name of the target TFE region
- workspace_name *(string)* – **[REQUIRED]** – Id of the target TFE workspace
- role *(string)* – **[REQUIRED]** – The target AWS role to assume. This role requires the necessary permissions to deploy the source terraform template in the target account.
- duration *(string)* – **[REQUIRED]** – Maps to the `DurationSections` option in boto3's `assume_role` and is subject to the same limitations. Set to 15 minutes by default.

Returns

The response contains the TFE api responses for each environment variable that is created within the target TFE workspace.

```
{
  "access_key_id": "TFE VARIABLE CREATION RESPONSE",
  "secret_access_key": "TFE VARIABLE CREATION RESPONSE",
  "region": "TFE VARIABLE CREATION RESPONSE",
  "aws_session_token": "TFE VARIABLE CREATION RESPONSE"
}
```

Gitlab

Designed to be triggered on Gitlab tag update events. This endpoint triggers a query against the target ServiceNow instance for a catalog item of the source terraform module. If a ServiceNow catalog item is found and its version is less than the current repo's version tag a new ServiceNow catalog item will be created and the previous version's catalog item will be disabled, otherwise no actions are taken.

Request Syntax

Expects the standard gitlab tag update request body

Returns

```
{
  "Status": "200"
}
```

TFE run

This endpoint will query the target workspace for the configuration upload url, `git clone` the target repo from Gitlab, and upload the resulting zip of your repo to the workspace. Currently workspace creation sets `Auto Apply` to true so any change in the configuration will trigger a Plan and Apply events.

Request Syntax

```
{
   "data" : [
    {
    "project_name": "terraform-aws-lx-instance",
    "repo_url": "git@your_gitlab_instance:gitlab.user/terraform-aws-lx-instance.git",
    "module_version": "vx.y.z",
    "workspace_id": "ws-123456ASDFhjklmn",
    "region": "us-east-1"
    }
   ]
  }
```

Parameters

- project_name *(string)* – **[REQUIRED]** – Name of your terraform module project.
- repo_url *(string)* – **[REQUIRED]** – SSH URI to the target gitlab repo containing your terraform module
- module_version *(string)* – **[REQUIRED]** – specific version tag of your repo that you want to associate the workspace with.

- workspace_id *(string)* – **[REQUIRED]** – target TFE workspace id
- region *(string)* – **[REQUIRED]** – target AWS region in which your terraform resources will be deployed.

Returns

If successful:

```
{
  "Status": "SUCCESS"
}
```

In the event of an error TerraSnow will return the response given by the TFE instance against it's call to `PUT https://archivist.terraform.io/v1/object/<UNIQUE OBJECT ID>`

Workflow

Listens for TFE workspace events, creates an empty TFE workspace and backs it with your source repo and version tag

Request Syntax

```
{
  "data" :
     [
       {
         "region": "us-east-1",
         "org_name": "your_tfe_org",
         "workspace_name": "your_tfe_workspace_name",
         "repo_id": "gitlab.user/tf_project",
         "repo_version": "x.y.z",
         "action": "CREATE"
       }
    ]
}
```

Parameters

- region *(string)* – **[REQUIRED]** – target AWS region in which the terraform resources will be deployed
- org_name *(string)* – **[REQUIRED]** – the target TFE organization name
- workspace_name *(string)* – **[REQUIRED]** – the target TFE workspace name
- repo_id *(string)* – **[REQUIRED]** – the id of the source terraform module's repo
- repo_version *(string)* – **[REQUIRED]** – the target version tag of the terraform module's repo
- action *(string)* – **[REQUIRED]** – the desired action on the target workspace, accepts CREATE or DELETE

Returns

TerraSnow simply passes back the response to the workspace creation api endpoint from the TFE instance.

From the official TFE workspace api documentation:

```
{
  "data": {
    "id": "ws-SihZTyXKfNXUWuUa",
    "type": "workspaces",
    "attributes": {
```

(continues on next page)

(continued from previous page)


```
      "name": "workspace-2",
      "environment": "default",
      "auto-apply": false,
      "locked": false,
      "created-at": "2017-11-02T23:55:16.142Z",
      "working-directory": null,
      "terraform-version": "0.10.8",
      "can-queue-destroy-plan": true,
      "vcs-repo": {
        "identifier": "skierkowski/terraform-test-proj",
        "branch": "",
        "oauth-token-id": "ot-hmAyP66qk2AMVdbJ",
        "ingress-submodules": false
      },
      "permissions": {
        "can-update": true,
        "can-destroy": false,
        "can-queue-destroy": false,
        "can-queue-run": false,
        "can-update-variable": false,
        "can-lock": false,
        "can-read-settings": true
      }
    },
    "relationships": {
      "organization": {
        "data": {
          "id": "my-organization",
          "type": "organizations"
        }
      },
      "ssh-key": {
        "data": null
      },
      "latest-run": {
        "data": null
      }
    },
    "links": {
      "self": "/api/v2/organizations/my-organization/workspaces/workspace-2"
    }
  }
}
```


2.5 ServiceNow Catalog Item

The details below include descriptions of the variables, client scripts, and script includes utilized in each ServiceNow terraform resource catalog item. Unless otherwise stated variables and client scripts are created automatically.

2.5.1 Variables

ServiceNow catalog item variables are automatically populated with the default values of their terraform module counterparts. Variables that are defined in the terraform module without a default value are created as required ServiceNow

catalog item variables.

The provided terraform module's variable description is populated in both the ServiceNow variable question text and tool tip.

tfv

Type: String

Description: Denotes the prefix given to the variables included in the terraform module's variables.tf file.

adv_toggle

Type: CheckBox

Description: Advanced mode toggle that is used to show/hide catalog item variables that are not marked as required.

Roles

Type: Select Box

Description: Used in conjunction with the `populateAWSRoleInfoOnLoad.js` client script. Contains the AWS account information for the user's select role.

gen_OS_Type

Type: String

Description: not currently in use

gen_aws_role

Type: String

Description: Holds the ARN of the role selected from the Roles dropdown. Auto filled via the `enableAfterPopulateRolesOnChange.js` OnChange event

gen_AwsAccountInfo

Type: Multi Line Text

Description: Used to hold a JSON object of AWS account info. Details on how this information is populated are not currently documented. More information to follow in a later release.

gen_module_version

Type: String

Description: The version of the terraform module as provided in the Gitlab repo tag event.

gen_region

Type: String

Description: The target region in which AWS resources will be provisioned. Populated via the `enableAfterPopulateRolesOnChange.js` OnChange event.

gen_org_name

Type: String

Description: The name of the Terraform Enterprise Organization. Currently populated from the TerraSnow configuration file.

gen_repo_url

Type: String

Description: SSH URI to the gitlab repo

2.5.2 Client Scripts

This project contains several ServiceNow client scripts contained within the `/sn_javascript` directory that support ease of use when ordering a terraform resource catalog item.

createDiaplyToggleOnChange.js

Type: OnChange

Associated Variable: adv_toggle

Description: Used to show or hide 'advanced'/default terraform module options (those variables included in the the terraform module that were provided with default values.)

hideGenericVariablesOnLoad.js

Type: OnLoad

Description: Hides variables prefixed with `gen_` on the catalog item load event.

populateAWSRoleInfoOnLoad.js

Type: OnLoad

Description: invokes the `populateAWSRoleInfoScriptInclude` to populate the roles variable dropdown. This variable's selection value is then passed to TerraSnow via the `/aws-assume-role-webhook` endpoint.

enableAfterPopulateRolesOnChange.js

Type: OnChange

Associated Variable: Roles

Description: popluates the gen_aws_role, gen_region variables on selection of the AWS role provided in the Roles variable dropdown

2.5.3 Script Includes

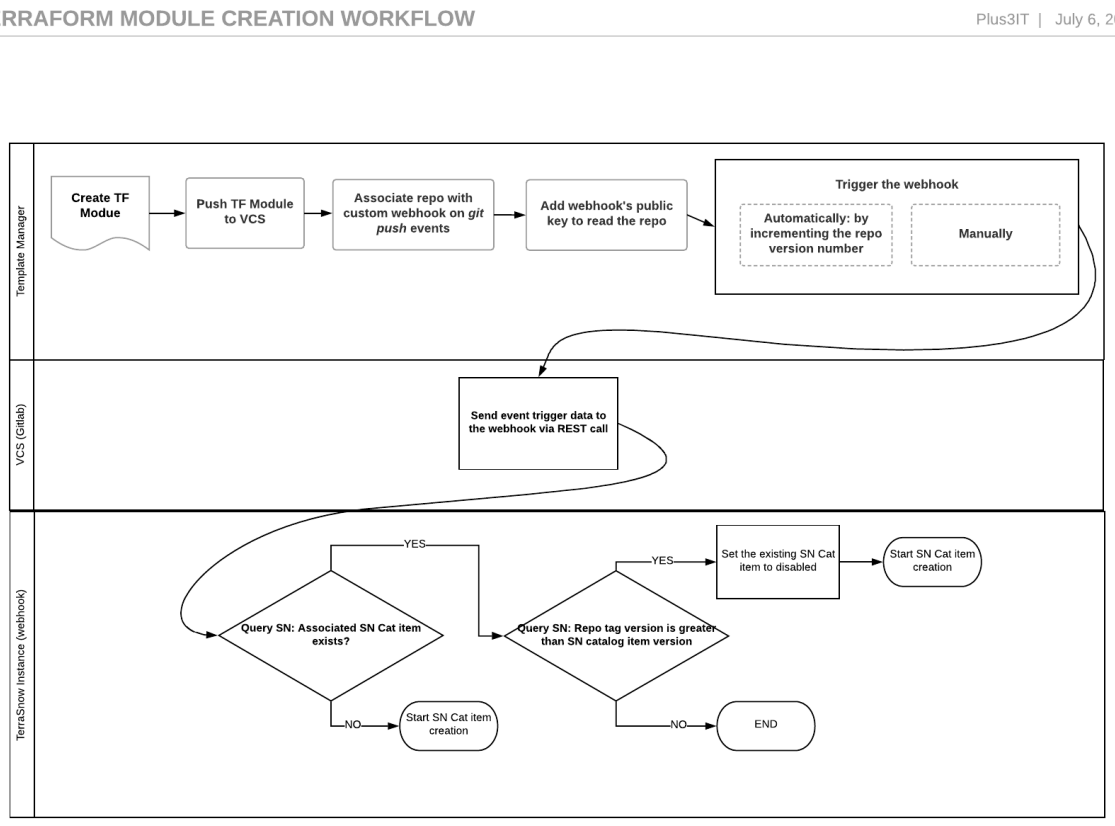
populateAWSRoleInfoScriptInclude.js

Description: Queries a custom table for the ServiceNow user's associated Active directory group, their default AWS region, and the AWS account ARN that has been associated with that Active Directory group. Returns a JSON object containing this information.

CHAPTER 3

Project Flow Diagrams

3.1 Terraform Module Creation Workflow



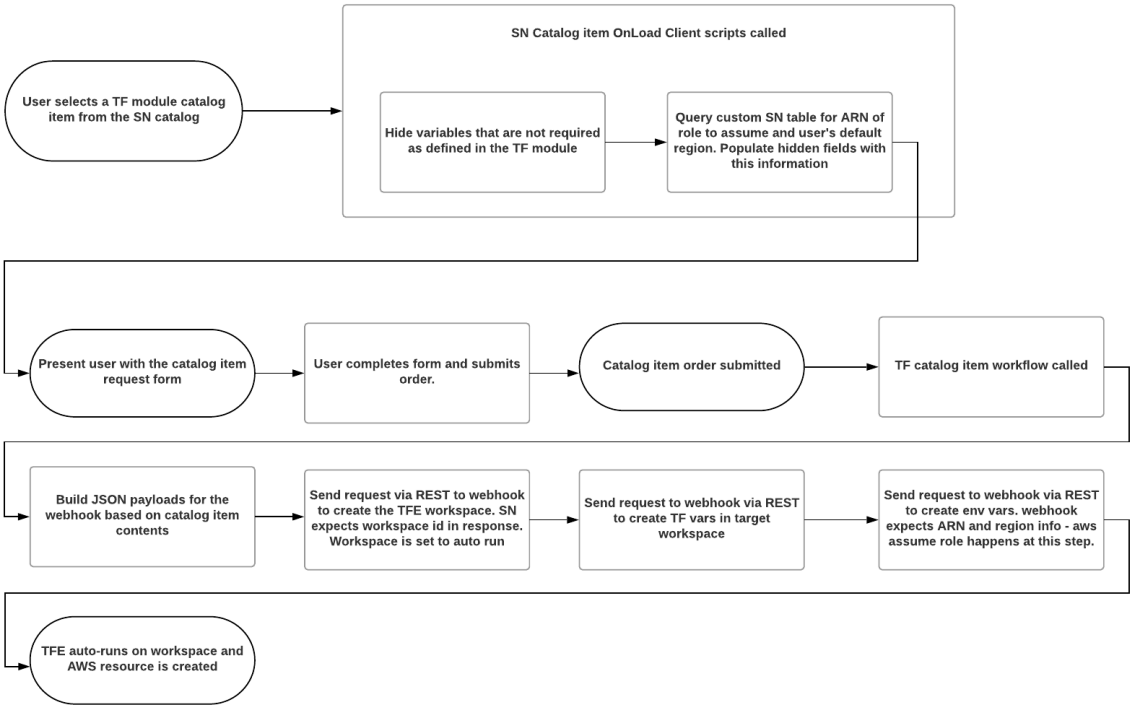
3.2 ServiceNow Catalog Item Order

SERIVCENOW CATALOG ITEM ORDER

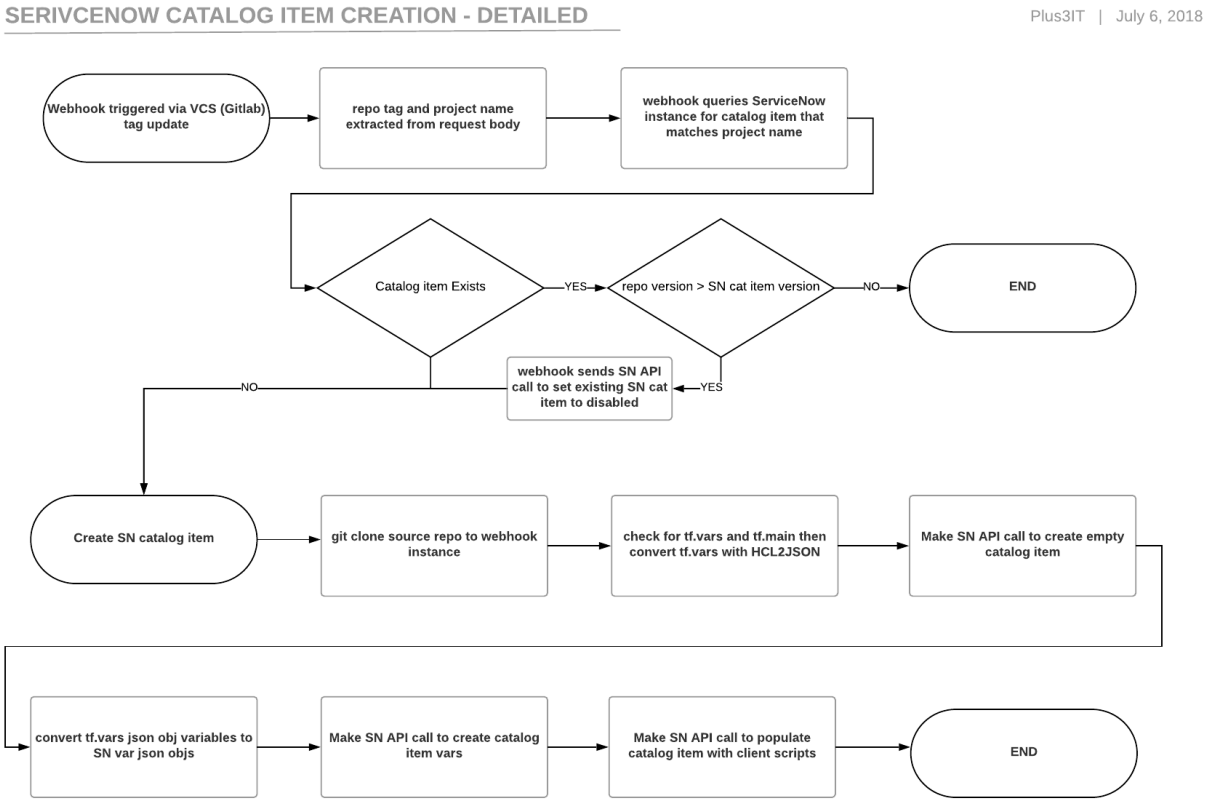
Plus3IT | July 6, 2018

PreRequisities/Assumptions:

- Webhook is deployed
- TFE instance has been configured properly
- TF module source repo is configured properly and the associated SN catalog item already exists
- Terrasnow scoped application has been deployed on the target ServiceNow instance



3.3 ServiceNow Catalog Item Creation - Detailed



CHAPTER 4

Supported Versions of ServiceNow

- Jakarta (tested working)