

# HashiCorp Terraform - Terraform Enterprise API LifeCycle

## Pre-Requisites

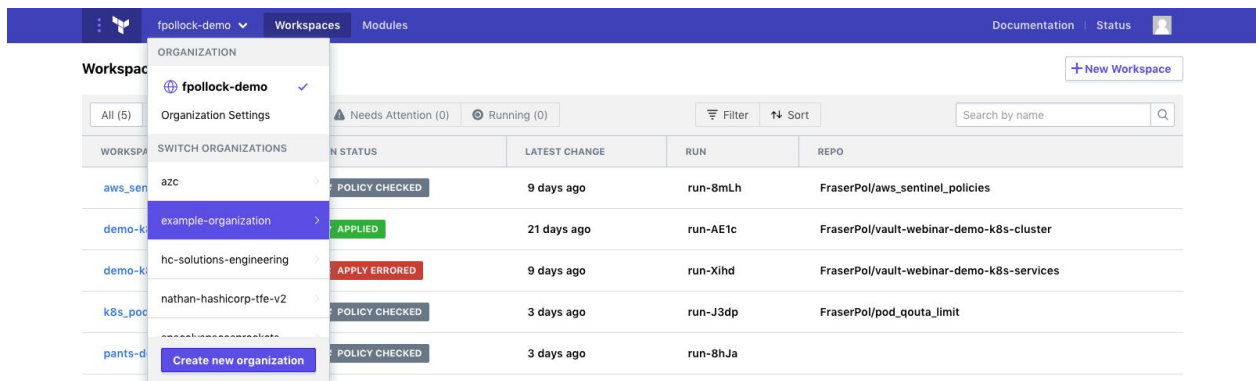
- An Organization is created on Private Terraform Enterprise or SaaS Terraform Enterprise
- User has the permissions and capability to get an **organization token** as well as their team/user token
- We'll be using [JQ](#) for the duration of this tutorial but its not required

This guide will go over the API-driven Run workflow for (P)TFE. You can use the following [URL](#) to follow along the logic or skip right into coding.

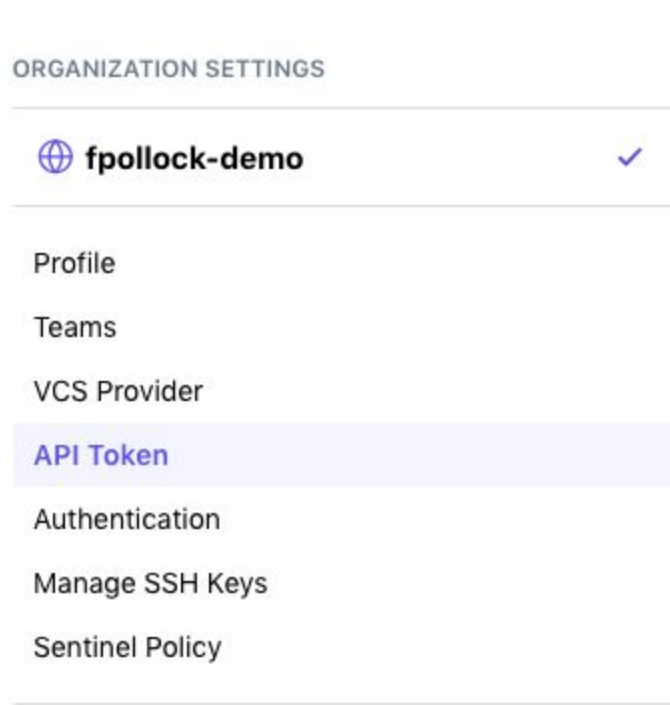
## Step 1 - Create a Workspace

The user must create a [workspace](#). Note: Workspace creation is restricted to members of the owners team, the owners team [service account](#), and the [organization service account](#). To interact with the API you need a authorization token. To create a workspace this needs to be an organization token. This can be acquired through the API or (P)TFE UI as follows;

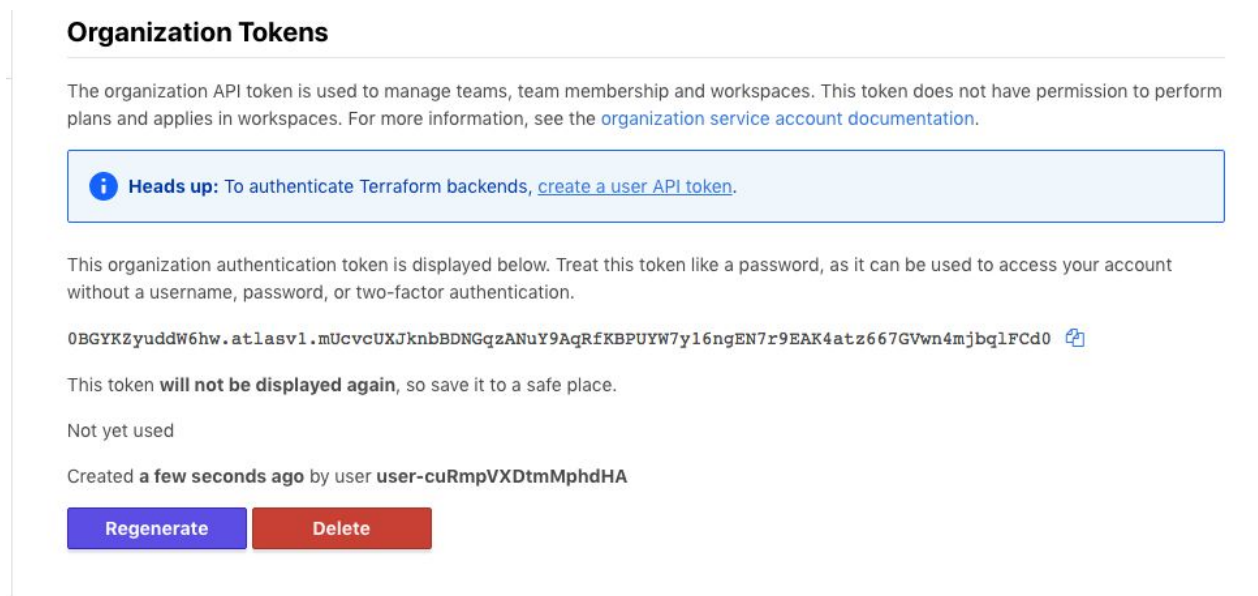
1. From the home page of Terraform Enterprise GUI select your organization drop down and “Organization Settings”



2. In the “Organization Settings” page then select “API Token”



3. Finally under “Organization Tokens” select either “Generate” if this is your first time or select “Regenerate” if you have created a token before

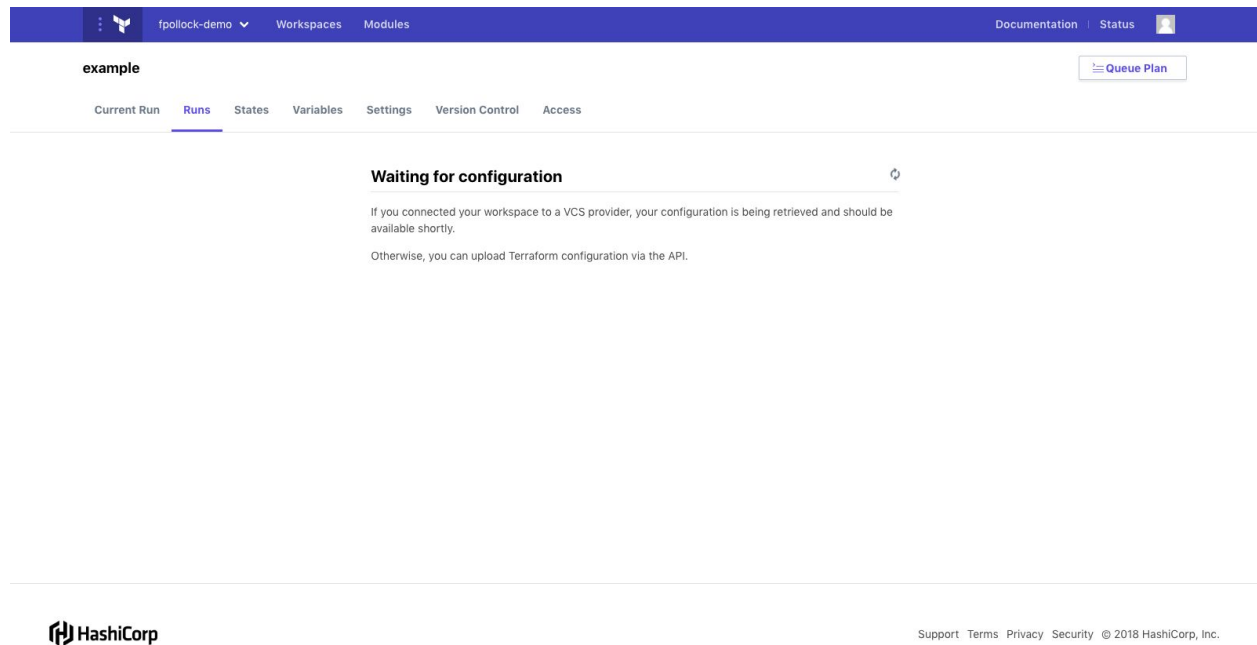


Once you have the token the next step is to actually create the Workspace. This particular guide is intended to create a Workspace without VSC backing for use in a CI/CD pipeline. The code to complete that is [here](#).

**Note - The response will contain a “id” field similar to “ws-o9L7sVHQMzAALeZa”. This will be used in later POST.**

## **Step 2 - Post Variables**

Once a workspace is created through the API you’ll have a blank canvas to continue. If you look at your Workspace via the GUI you should now see a similar landing page



The next step for our ‘example’ Workspace is to configure our variables. There are two ways to do this. One is through the TFE-CLI which you can review [here](#). The way we will be doing it is via the [API](#).

The code required to do this can be found [here](#). Remember you’ll need the workspace ID we got from our first step.

**NOTE - TFE allows variables to be marked as sensitive, make sure you hide those cloud credentials!**

**NOTE - To loop through this in a more effective manner you can do a for statement, use [this](#) example to guide you.**

**NOTE - If you are doing this for PTFE ensure you have a filter for your organization and workspace!**

[Here](#) is an example -

[https://\\$address/api/v2/vars?filter%5Borganization%5D%5Bname%5D=\\${organization}&filter%5Bworkspace%5D%5Bname%5D=\\${workspace}](https://$address/api/v2/vars?filter%5Borganization%5D%5Bname%5D=${organization}&filter%5Bworkspace%5D%5Bname%5D=${workspace})

### **Step 3 - Creating a Configuration, Run and Apply**

Performing a run on a new configuration is a multi-step process.

1. [Create a configuration version on the workspace.](#)
2. [Upload configuration files to the configuration version.](#)
3. [Create a run on the workspace](#); this is done automatically when a configuration file is uploaded.
4. [Create and queue an apply on the run](#); if auto-apply is not enabled.

This endpoint cannot be accessed with [organization tokens](#). You must access it with a [user token](#) or [team token](#). To get a user token we'll go back into our "Organization Settings" then into "API Tokens" and finally this time we'll select "User Tokens"

The screenshot shows the HashiCorp Terraform Cloud interface. At the top, there's a navigation bar with 'fpollock-demo' selected under 'Workspaces'. A dropdown menu for 'ORGANIZATION' is open, showing a list of organizations: 'fpollock-demo' (checked), 'azc', 'example-organization', 'hc-solutions-engineering', 'nathan-hashicorp-tfe-v2', and 'xxxxxxxxxxxxxxxxxxxx'. Below the list is a 'Create new organization' button. The main content area is titled 'Waiting for configuration' and contains the text: 'If you connected your workspace to a VCS provider, your configuration is being retrieved and should be available shortly. Otherwise, you can upload Terraform configuration via the API.' The footer includes the HashiCorp logo and the text 'Support Terms Privacy Security © 2018 HashiCorp, Inc.' The URL in the address bar is 'https://app.terraform.io/app/fpollock-demo/settings/profile'.

ORGANIZATION SETTINGS

fpollock-demo ✓

Profile

Teams

VCS Provider

API Token

Authentication

Manage SSH Keys

Sentinel Policy

## API Tokens

### User Tokens

A user API token has the same permission level as your user account. It is the only type of token which can be granted access to multiple organizations. Generate user API tokens on your [user settings page](#).

### Team Tokens

Team API tokens are used by services like a CI/CD pipeline to perform plans and applies on a workspace. To generate a team API token, go to the [Teams](#) page in your organization settings, select a team, and generate a new token at the bottom of its page.

### Organization Tokens

The organization API token is used to manage teams, team membership and workspaces. This token does not have permission to perform plans and applies in workspaces. For more information, see the [organization service account documentation](#).

**Heads up:** To authenticate Terraform backends, [create a user API token](#).

Treat this token like a password, as it can be used to access your account without a username, password, or two-factor authentication.

Not yet used

Created an hour ago by user user-cuRmpVXDtmMphdHA

[Regenerate](#) [Delete](#)

Give our token a name and click “Generate Token”

USER SETTINGS

Profile

Password

Two Factor Authentication

Tokens

## Tokens

### Generate new token

DESCRIPTION

example\_token [Generate token](#)

Use this description to help identify the use of the token in the future.

### Tokens (1)

Automation Test [Created 3 days ago](#) [Delete](#)

This token will not be displayed again, so save it to a safe place.

Now that we have a token to authenticate we can start to go through the steps to invoke a run. First we need to [Create a configuration version on the workspace](#). The code to complete this can be found [here](#).

**Note: The upload-url will be needed later, that's where we're going to post our configuration later!**

Now that we've got a configuration version we need to populate it with our Terraform code. The most significant difference in this workflow is that TFE *does not* fetch configuration files from version control. Instead, your own tooling must upload the configurations as a .tar.gz file. This allows you to work with configurations from unsupported version control systems, automatically generate Terraform configurations from some other source of data, or build a variety of other integrations.

The most important thing to note here is that when we upload our configuration we must make sure the main.tf is at the root of the uploaded folder as such;

- Example.tar.gz
  - main.tf
  - variables.tf
  - output.tf
  - modules
    - network
    - app

Once we've successfully zipped up our Terraform configuration the the next step is to upload it. The code to accomplish this is [here](#).

**NOTE - The upload URL should have came from your creating a configuration step**

If in the first step you followed our example the run will be automatically que'd

Key path	Type	Default	Description
data.attributes.auto-queue-runs	boolean	true	When true, runs are queued automatically when the configuration version is uploaded.

If not you'll need to follow the extra steps to [create a run on the workspace](#).

Finally the last thing we need to do is "Confirm the run" by [Creating and queueing an apply on the run](#).

You can do this by listing all the runs in a workspace or by checking our last output for the run\_id. The steps to apply are listed [here](#).

Congratulations! You just created and invoked a run in TFE with the API. If you wanted to do an additional run you could start again at the step where we created the configuration and repeat.

If you want additional references or code to review you can see the API Driven Workflow logic [here](#) the entirety of the TFE API Documents [here](#) and also a reference guide [here](#).