TFE Standard Operating Procedures Runbook

Target Audience: TFE Systems Administrators, Cloud Engineers, DevOps-Oriented Engineers

Contact:
Ryan Butler, Senior Solutions Architect
Jesse Adelman, Engineer
Jim Raymonds, Project Manager
April 20, 2020

AHEAD
401 North Michigan Ave., Suite 3400
Chicago, IL 60611
312.924.4492 (office)
800.294.5141 (fax)
www.ThinkAHEAD.com

Table of Contents

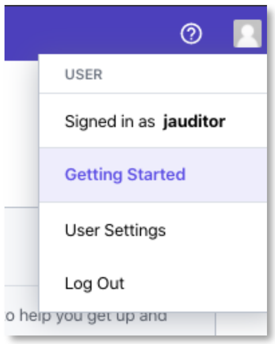
Purpose of This Document

This document has been designed as a general guide for typical, common tasks in Terraform Enterprise. However, in all cases, HashiCorp documentation is the primary source of truth to reference. At the time of this document's writing, HashiCorp's Terraform Enterprise documentation can be found at

https://www.terraform.io/docs/enterprise/index.html
and
https://www.terraform.io/docs/cloud/index.html.

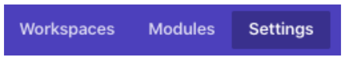
New to Terraform?

Please take some time to walk through the "Getting Started" document, under your User Profile menu in the upper right:

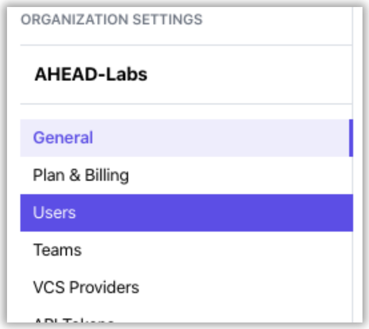


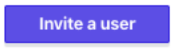
Within the TFE Application Interface

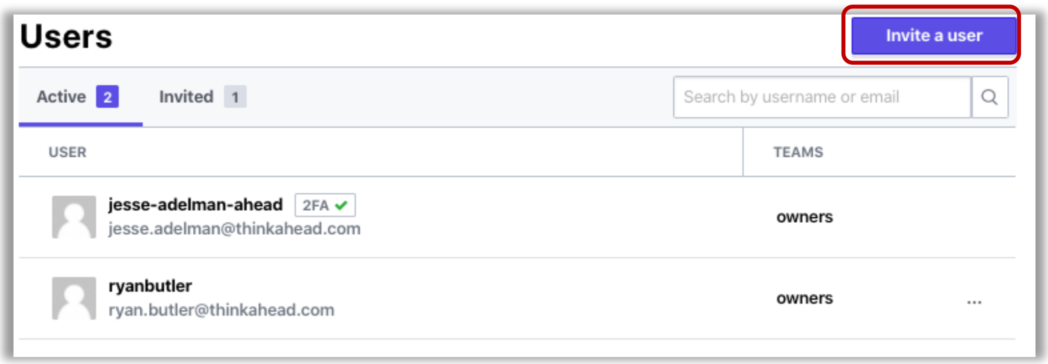
Adding a new user

1. In the top left navigation bar, select **Settings**: 

2. Then, select **Users** from the left column:

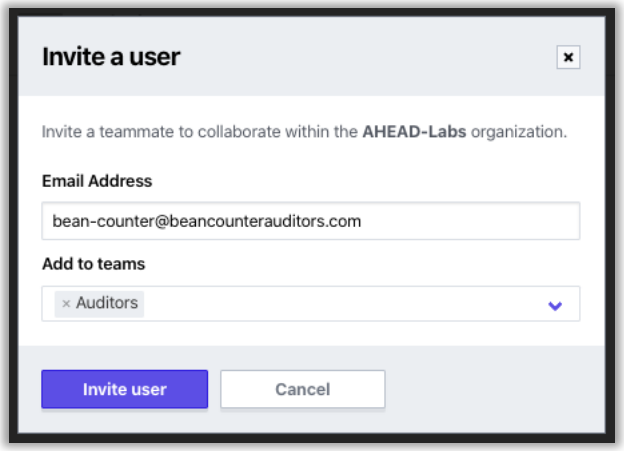


3. In the **Users** dialog page, select  as seen below:



USER	TEAMS
jesse-adelman-ahead 2FA ✓ jesse.adelman@thinkahead.com	owners
ryanbutler ryan.butler@thinkahead.com	owners

4. Enter the user's email address and add them to the appropriate Team:



Invite a teammate to collaborate within the **AHEAD-Labs** organization.

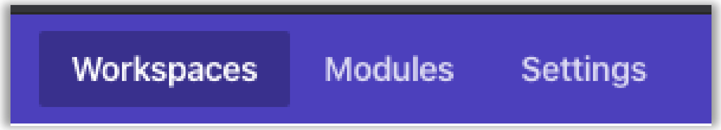
Email Address: bean-counter@beancounterauditors.com

Add to teams: × Auditors

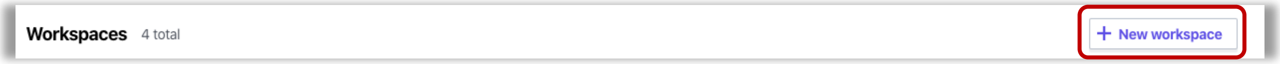
5. Once the user accepts the email invitation, they will be added. Please urge new users to go through the "Getting Started" flow mentioned earlier in this RunBook!

Creating a Workspace

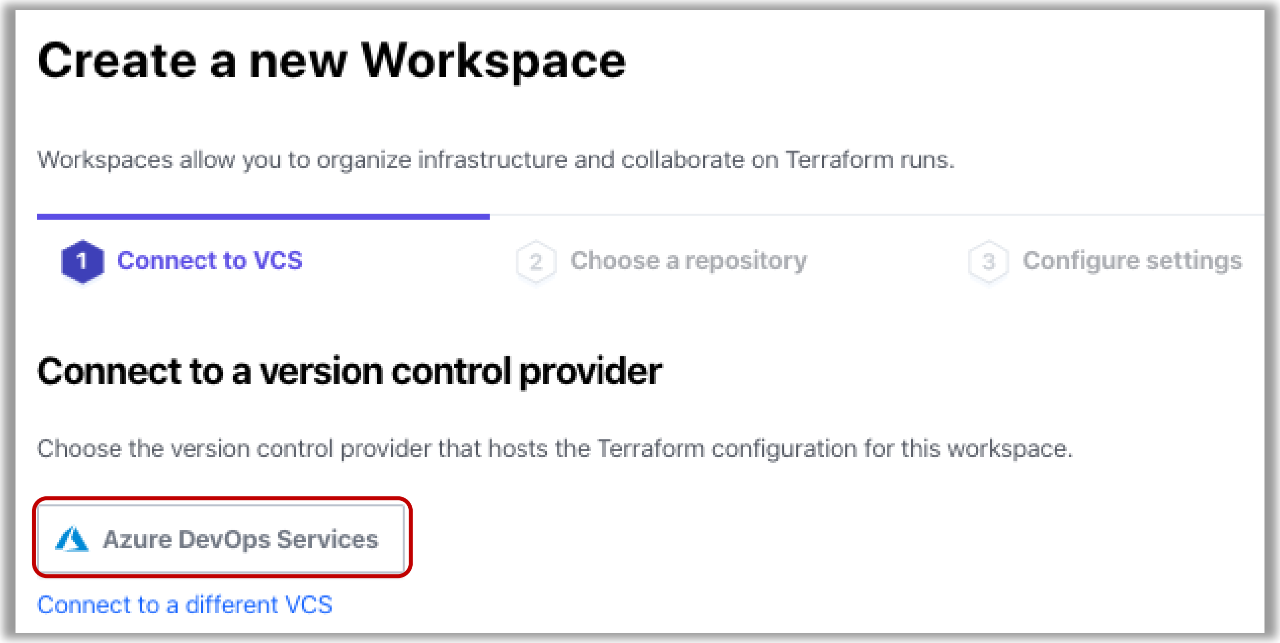
1. Click to select **Workspaces** in the top navigation bar



2. Next, click **+ New Workspace** at the right



3. You will now enter a three-step "Wizard" web flow. Select **Azure DevOps Services**.



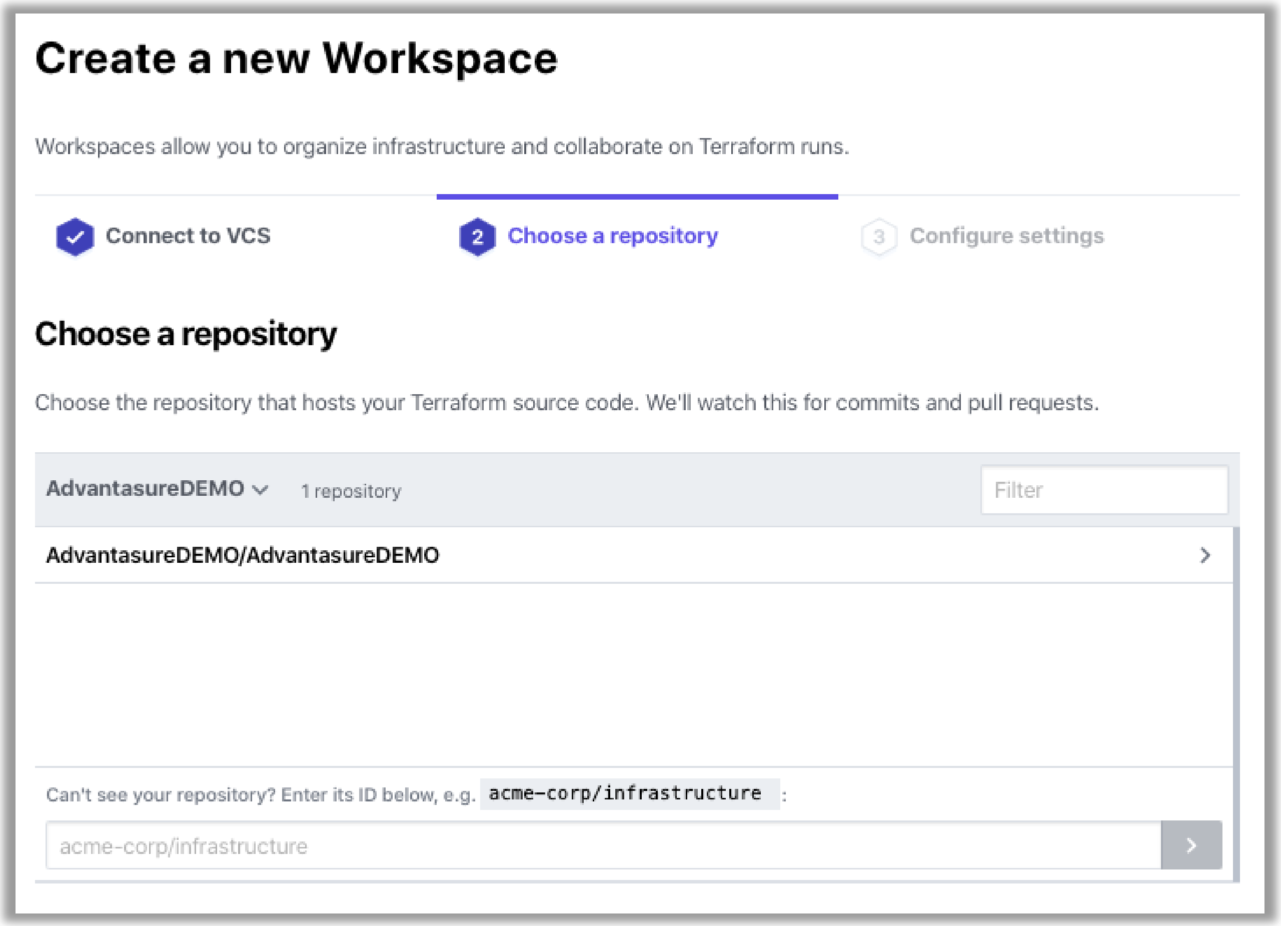
Create a new Workspace

Workspaces allow you to organize infrastructure and collaborate on Terraform runs.

Connect to a version control provider

Choose the version control provider that hosts the Terraform configuration for this workspace.

4. Proceed to Step 2, **Choose a repository**:



Create a new Workspace

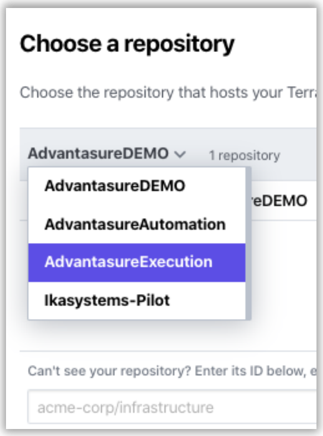
Workspaces allow you to organize infrastructure and collaborate on Terraform runs.

Choose a repository

Choose the repository that hosts your Terraform source code. We'll watch this for commits and pull requests.

Can't see your repository? Enter its ID below, e.g. `acme-corp/infrastructure` :

5. Click and hold the drop-down Repository menu and select **AdvantasureExecution**:

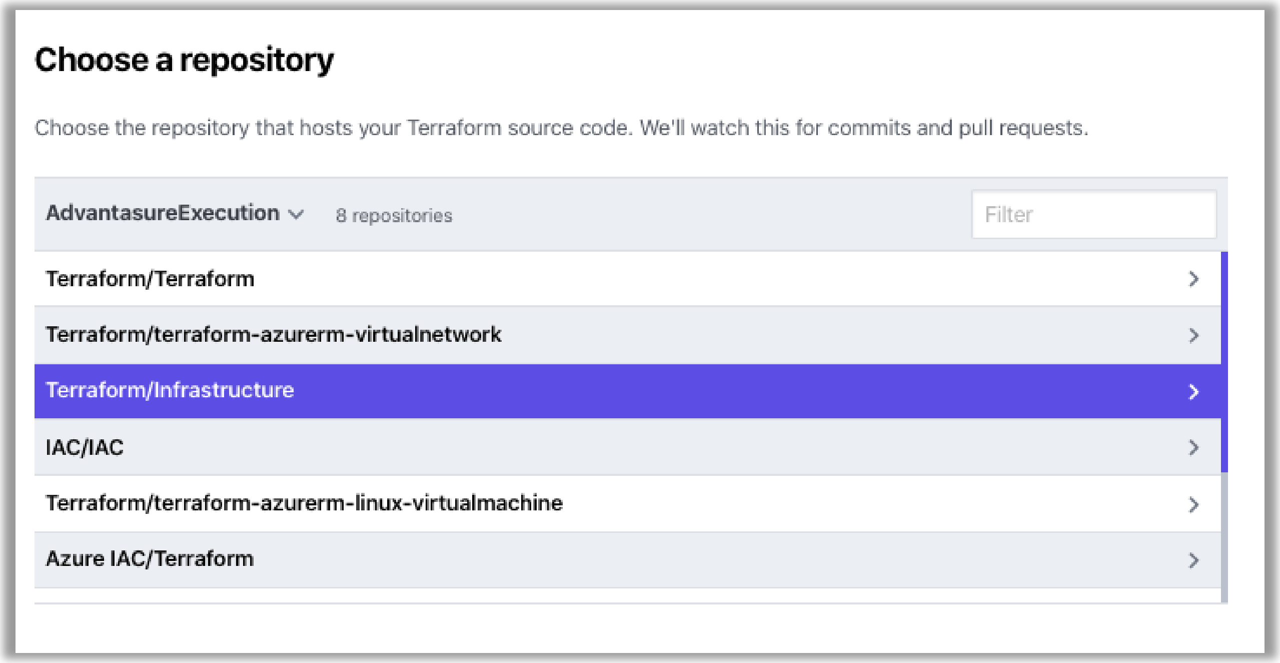


Choose a repository

Choose the repository that hosts your Terr

Can't see your repository? Enter its ID below, e

6. The **Terraform/Infrastructure** repository will now be available. As of this writing, this will be the Example "root" module for deploying a single Linux VM instance in Azure (this list may change over time).

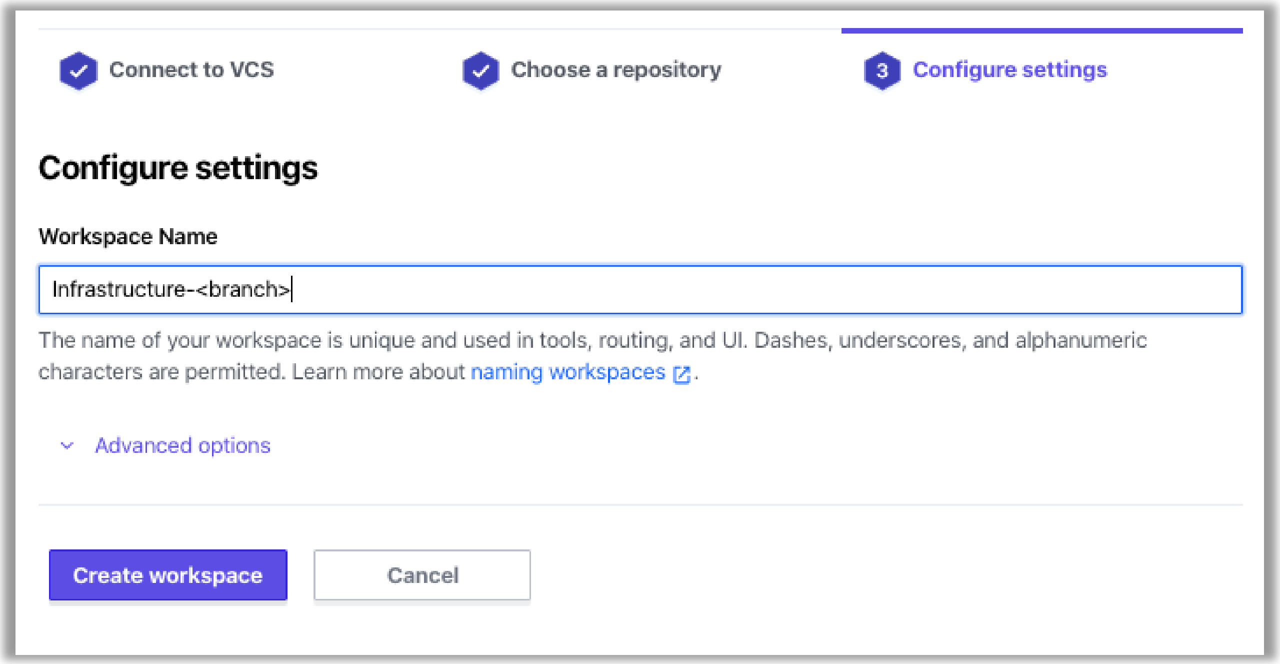


Choose a repository

Choose the repository that hosts your Terraform source code. We'll watch this for commits and pull requests.

Terraform/Terraform
Terraform/terraform-azurerm-virtualnetwork
Terraform/Infrastructure
IAC/IAC
Terraform/terraform-azurerm-linux-virtualmachine
Azure IAC/Terraform

7. Proceed to Step 3, **Configure Settings**:



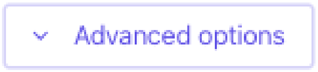
Configure settings

Workspace Name

Infrastructure-<branch>

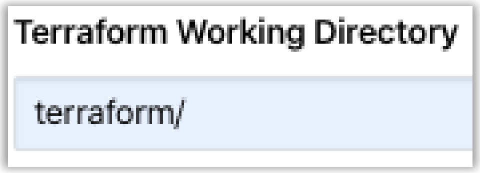
The name of your workspace is unique and used in tools, routing, and UI. Dashes, underscores, and alphanumeric characters are permitted. Learn more about naming workspaces.

Here, you'll enter the Workspace name. The convention shown here is to take the name of the Git Repo and append "-<branch>" to the end, where <branch> is the name of the git branch this Workspace will represent (could typically be 'dev', 'qa', 'prod', 'production', etc.).

8. Next, click 

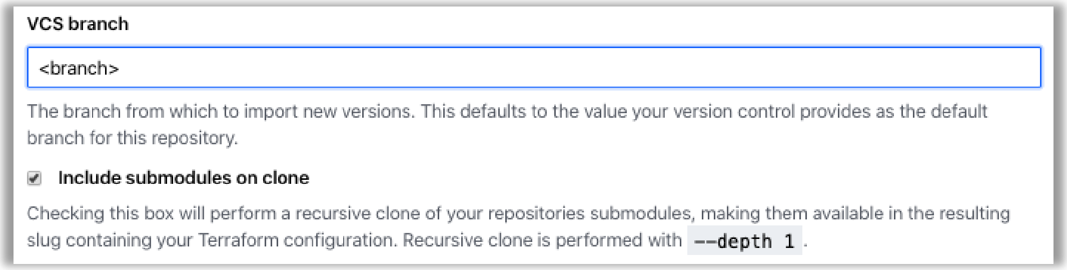
.

9. Then, enter **terraform/** under **Terraform Working Directory**, like so:


Terraform Working Directory

terraform/

10. When you fill in the above field, the rest of the form will change. Next, fill in the branch information, to match the <branch> you entered above, and check the box **Include submodules on clone**:

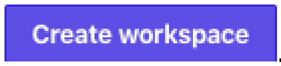

VCS branch

<branch>

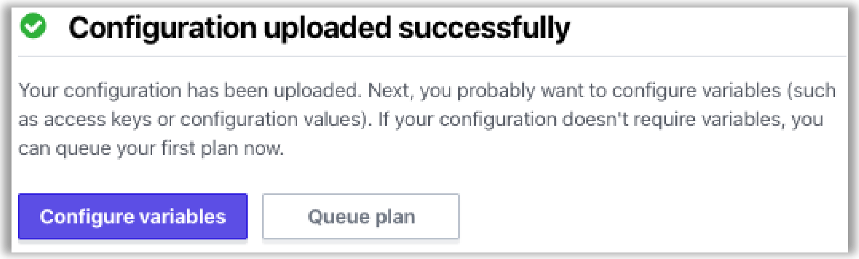
The branch from which to import new versions. This defaults to the value your version control provides as the default branch for this repository.

☑ Include submodules on clone

Checking this box will perform a recursive clone of your repositories submodules, making them available in the resulting slug containing your Terraform configuration. Recursive clone is performed with `--depth 1`.

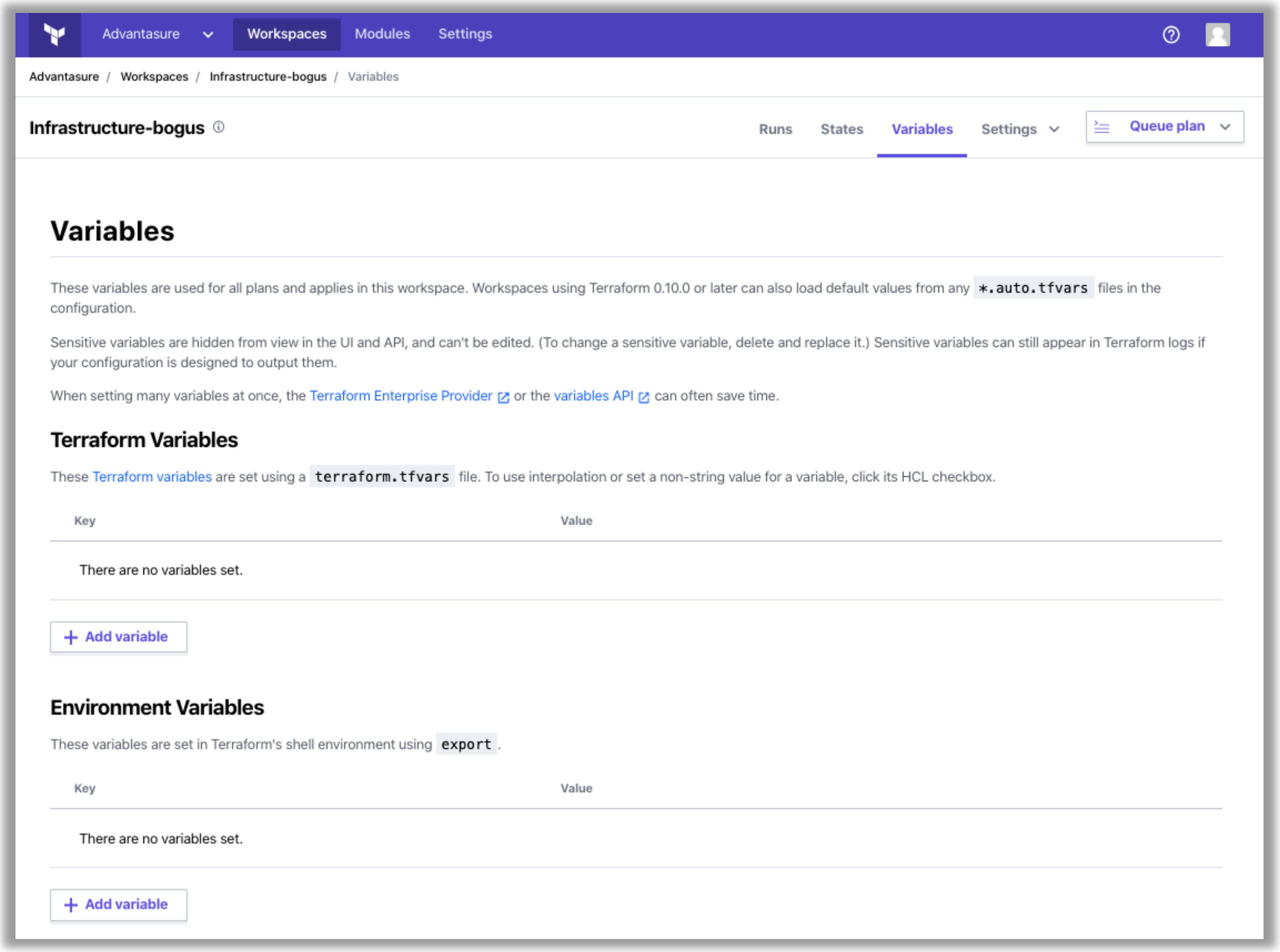
11. Finally, click 

. Then, you should see this in the middle of the page. Click **Configure variables**:


Configuration uploaded successfully

Your configuration has been uploaded. Next, you probably want to configure variables (such as access keys or configuration values). If your configuration doesn't require variables, you can queue your first plan now.

12. The next screen is for configuring Environment Variables that permit access to the IaaS provider (in our case, Azure):



Variables

These variables are used for all plans and applies in this workspace. Workspaces using Terraform 0.10.0 or later can also load default values from any `*.auto.tfvars` files in the configuration.

Sensitive variables are hidden from view in the UI and API, and can't be edited. (To change a sensitive variable, delete and replace it.) Sensitive variables can still appear in Terraform logs if your configuration is designed to output them.

When setting many variables at once, the Terraform Enterprise Provider or the variables API can often save time.

Terraform Variables

These Terraform variables are set using a `terraform.tfvars` file. To use interpolation or set a non-string value for a variable, click its HCL checkbox.

Key	Value
There are no variables set.	

+ Add variable

Environment Variables

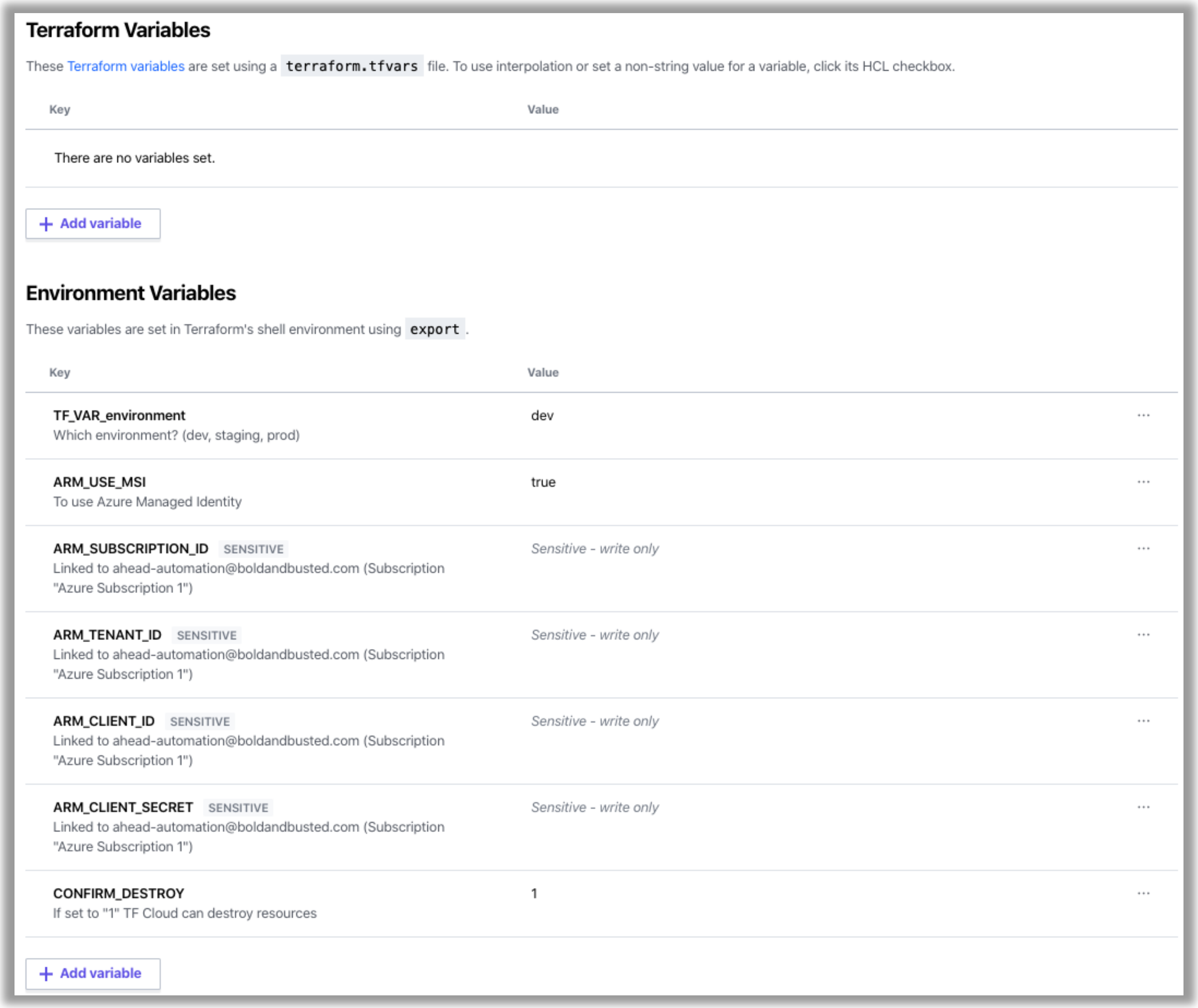
These variables are set in Terraform's shell environment using `export`.

Key	Value
There are no variables set.	

+ Add variable

The following screenshot is an example “Environment Variables” settings page from a configured development Workspace with Azure.

Note: This example has “CONFIRM_DESTROY” set to “1” (or “True”). This means that for this Workspace, there will be no confirmation before resources managed by Terraform are completely destroyed! AHEAD recommends that this setting is selected ***only on true development environments the destruction of which will have zero customer impact***.



Terraform Variables

These Terraform variables are set using a `terraform.tfvars` file. To use interpolation or set a non-string value for a variable, click its HCL checkbox.

Key	Value
There are no variables set.	

+ Add variable

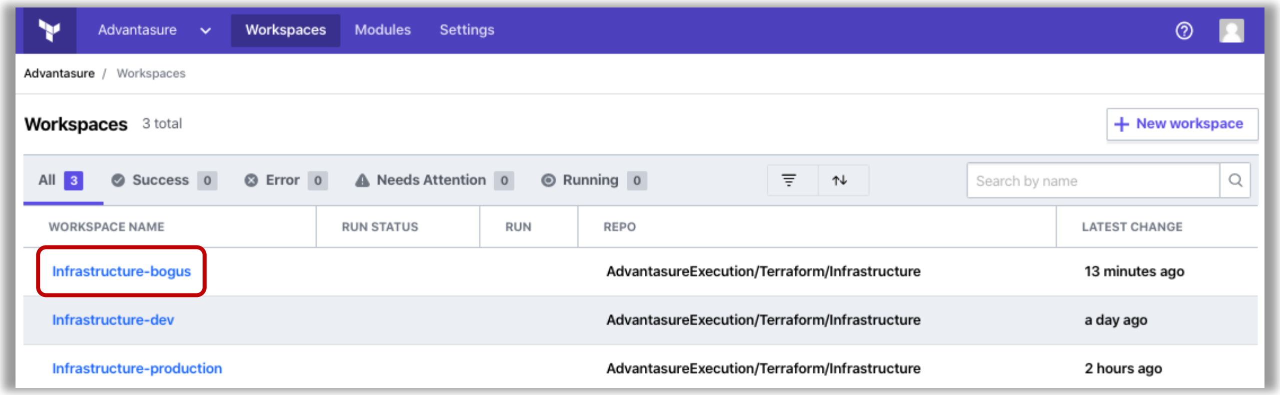
Environment Variables

These variables are set in Terraform's shell environment using `export`.

Key	Value
TF_VAR_environment Which environment? (dev, staging, prod)	dev
ARM_USE_MSI To use Azure Managed Identity	true
ARM_SUBSCRIPTION_ID SENSITIVE Linked to ahead-automation@boldandbusted.com (Subscription "Azure Subscription 1")	*Sensitive - write only*
ARM_TENANT_ID SENSITIVE Linked to ahead-automation@boldandbusted.com (Subscription "Azure Subscription 1")	*Sensitive - write only*
ARM_CLIENT_ID SENSITIVE Linked to ahead-automation@boldandbusted.com (Subscription "Azure Subscription 1")	*Sensitive - write only*
ARM_CLIENT_SECRET SENSITIVE Linked to ahead-automation@boldandbusted.com (Subscription "Azure Subscription 1")	*Sensitive - write only*
CONFIRM_DESTROY If set to "1" TF Cloud can destroy resources	1

+ Add variable

Deleting a workspace



Workspaces 3 total

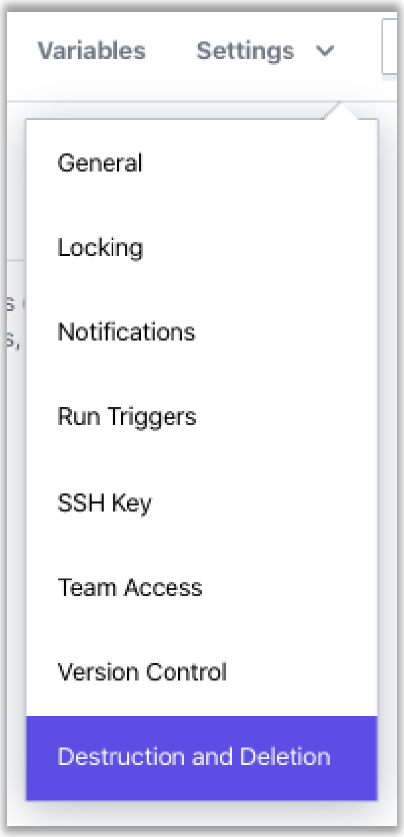
+ New workspace

All 3 | Success 0 | Error 0 | Needs Attention 0 | Running 0 | Search by name

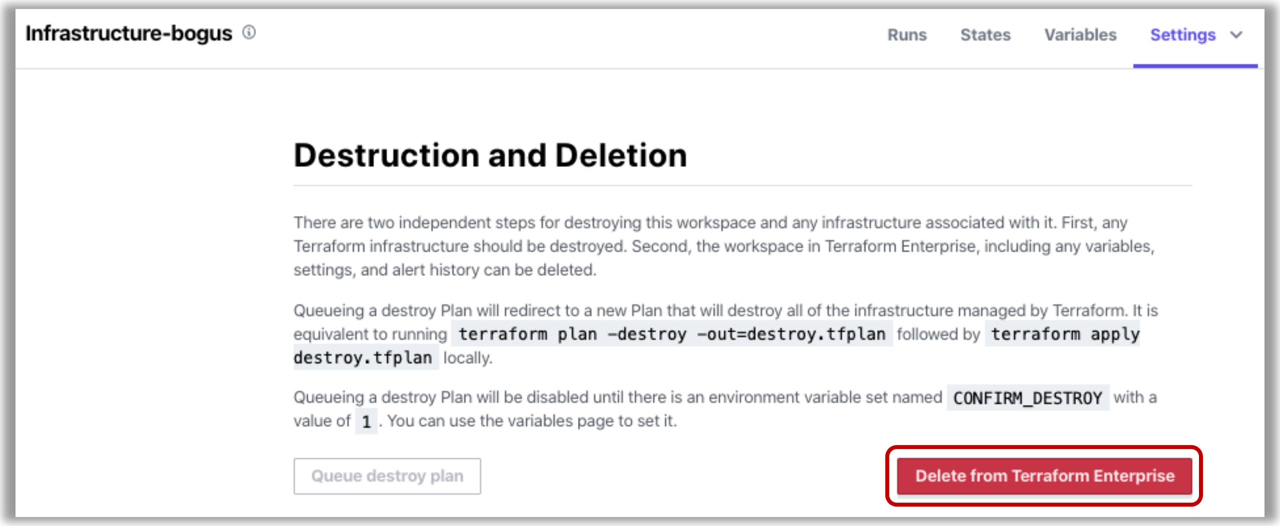
WORKSPACE NAME	RUN STATUS	RUN	REPO	LATEST CHANGE
Infrastructure-bogus			AdvantasureExecution/Terraform/Infrastructure	13 minutes ago
Infrastructure-dev			AdvantasureExecution/Terraform/Infrastructure	a day ago
Infrastructure-production			AdvantasureExecution/Terraform/Infrastructure	2 hours ago

1. In this example, we will delete “infrastructure-bogus” in the list above. First, select the workspace to be deleted from that list by clicking it.

2. Next, go to the **Settings** menu and select **Destruction and Deletion**:



3. Then, you'll see this page:



Destruction and Deletion

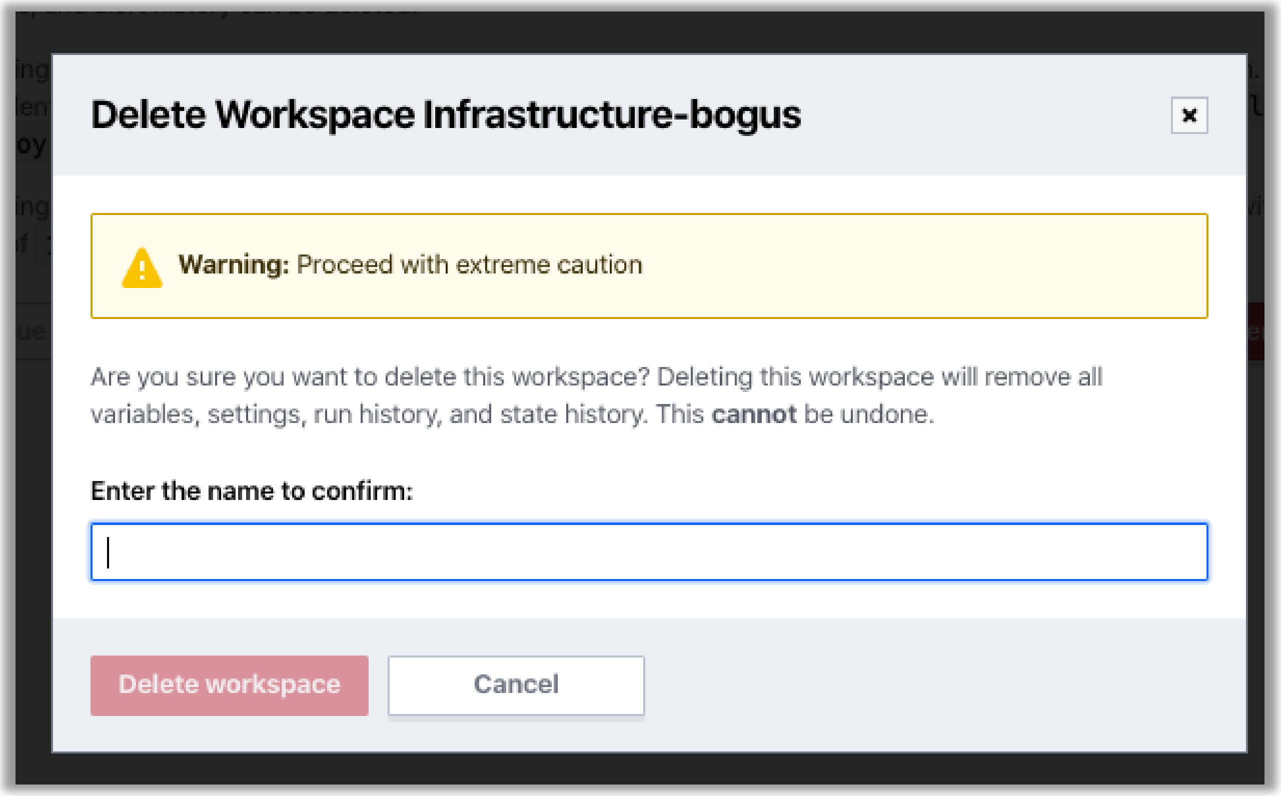
There are two independent steps for destroying this workspace and any infrastructure associated with it. First, any Terraform infrastructure should be destroyed. Second, the workspace in Terraform Enterprise, including any variables, settings, and alert history can be deleted.

Queueing a destroy Plan will redirect to a new Plan that will destroy all of the infrastructure managed by Terraform. It is equivalent to running `terraform plan -destroy -out=destroy.tfplan` followed by `terraform apply destroy.tfplan` locally.

Queueing a destroy Plan will be disabled until there is an environment variable set named `CONFIRM_DESTROY` with a value of `1`. You can use the variables page to set it.

4. If you are certain about this operation, click the red button **Delete from Terraform Enterprise**. **NOTE** that this will **NOT** destroy the associated infrastructure, only the Workspace as defined in Terraform Enterprise. If you wish to destroy the infrastructure as well, you will need to follow the instructions shown above to set the **CONFIRM_DESTROY** Environment Variable to enable the ability to destroy the built infrastructure with the "Queue destroy plan" button shown above.

5. Next, you will see this pop-over dialog requiring you to enter the name of the Workspace before Terraform Enterprise will permit a deletion of the Workspace.



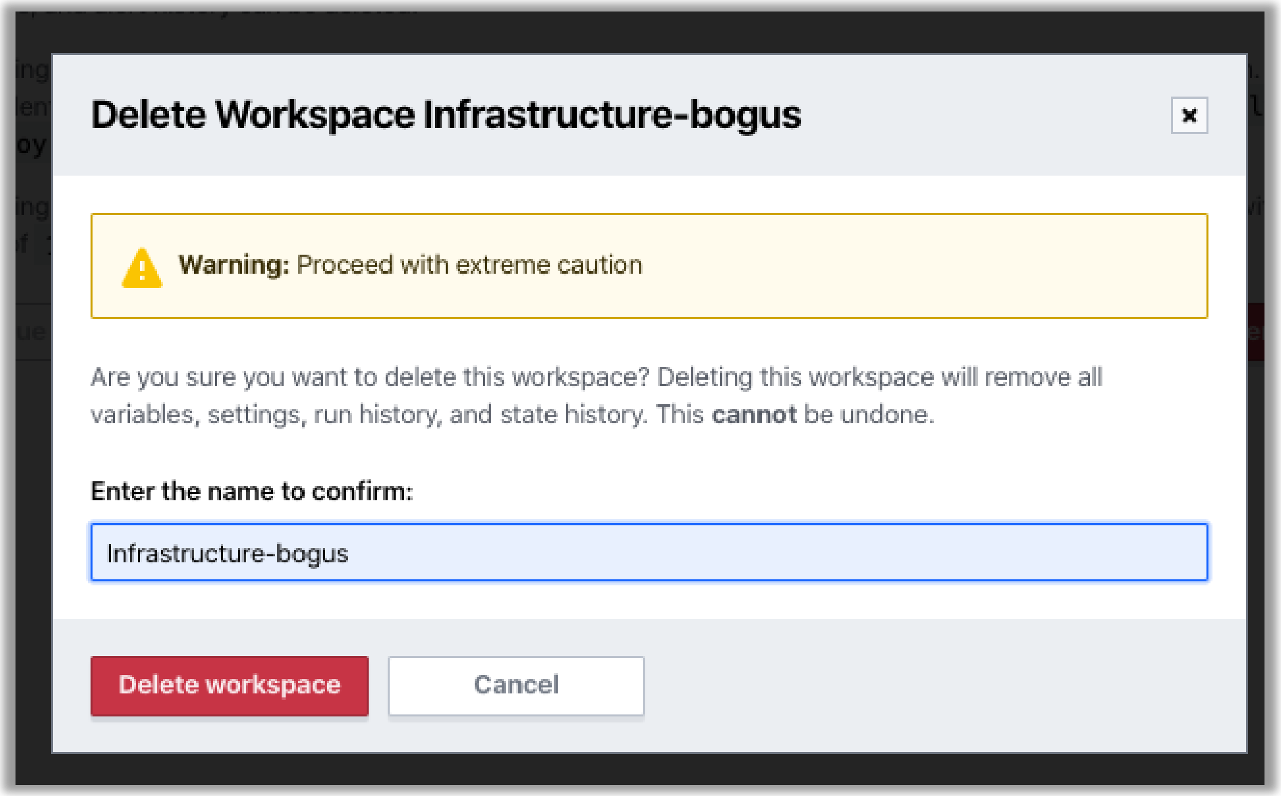
Delete Workspace Infrastructure-bogus

Warning: Proceed with extreme caution

Are you sure you want to delete this workspace? Deleting this workspace will remove all variables, settings, run history, and state history. This **cannot** be undone.

Enter the name to confirm:

6. Type the name of the Workspace to enable the “Delete workspace” red button.



Delete Workspace Infrastructure-bogus

Warning: Proceed with extreme caution

Are you sure you want to delete this workspace? Deleting this workspace will remove all variables, settings, run history, and state history. This **cannot** be undone.

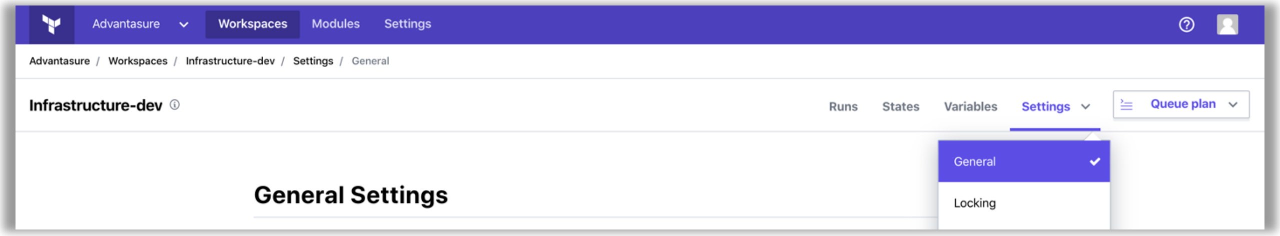
Enter the name to confirm:

Infrastructure-bogus

7. Click the **Delete Workspace** button. You should see a Green-tinged “Success” popover dialog appear in the lower left. You have deleted the Workspace.

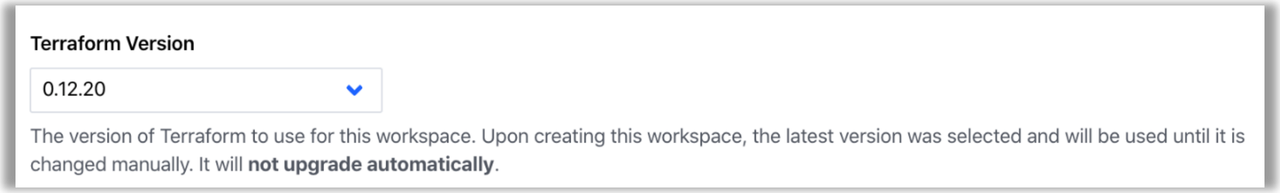
Changing the Terraform CLI version to use

1. Under **Workspaces** on the top navigation bar, select a Workspace. There, in the Settings drop-down menu towards the right, select **General**:



2. Scroll down until you see **Terraform Version**. Select the preferred version of Terraform from the dropdown list.

   Note: Setting “latest” will always use the latest version of Terraform on each plan and apply.



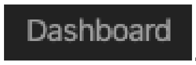
Terraform Version

0.12.20

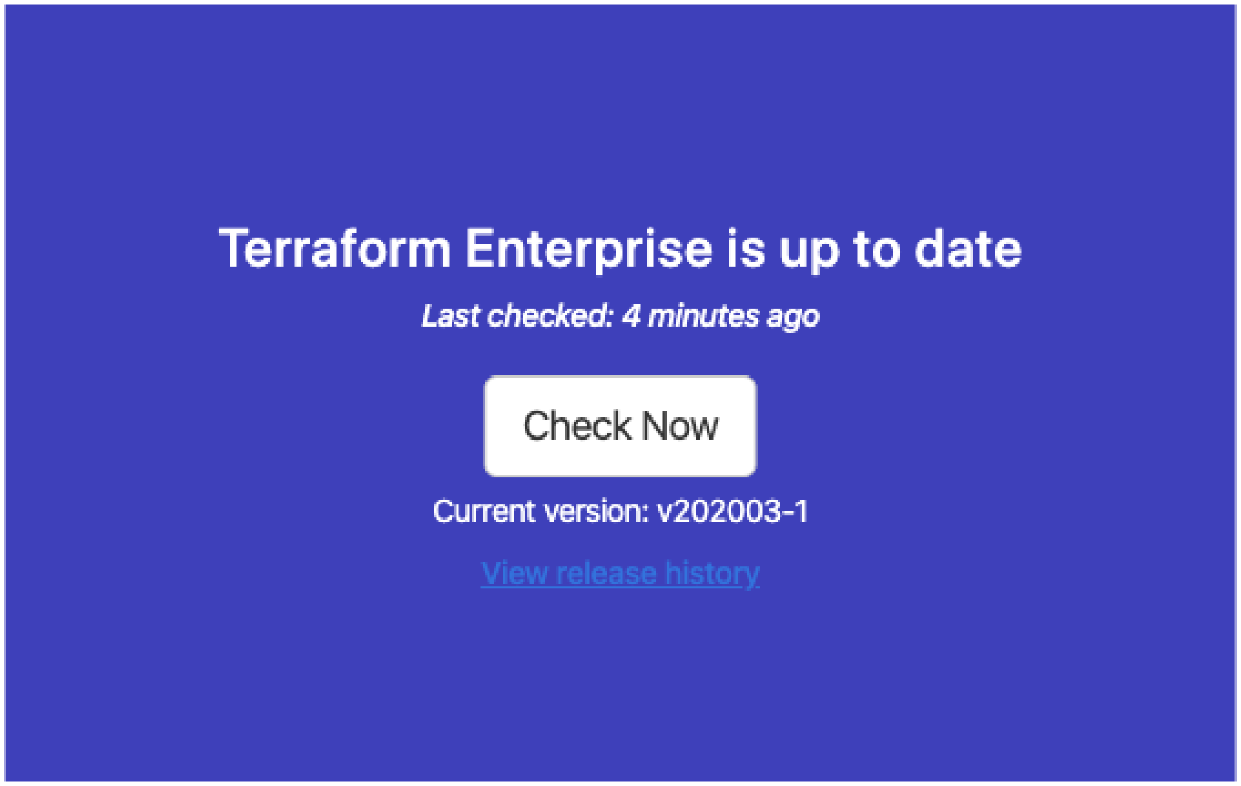
The version of Terraform to use for this workspace. Upon creating this workspace, the latest version was selected and will be used until it is changed manually. It will **not upgrade automatically**.

Within the TFE Admin Console (seen in the installation tasks)

Upgrading TFE to the latest version

1. On the TFE Admin Console (located at the same base URL as the console, but at port 8800), select .

   Then, click the **Check Now** button in the central pane in the Dashboard, as seen below:



Terraform Enterprise is up to date

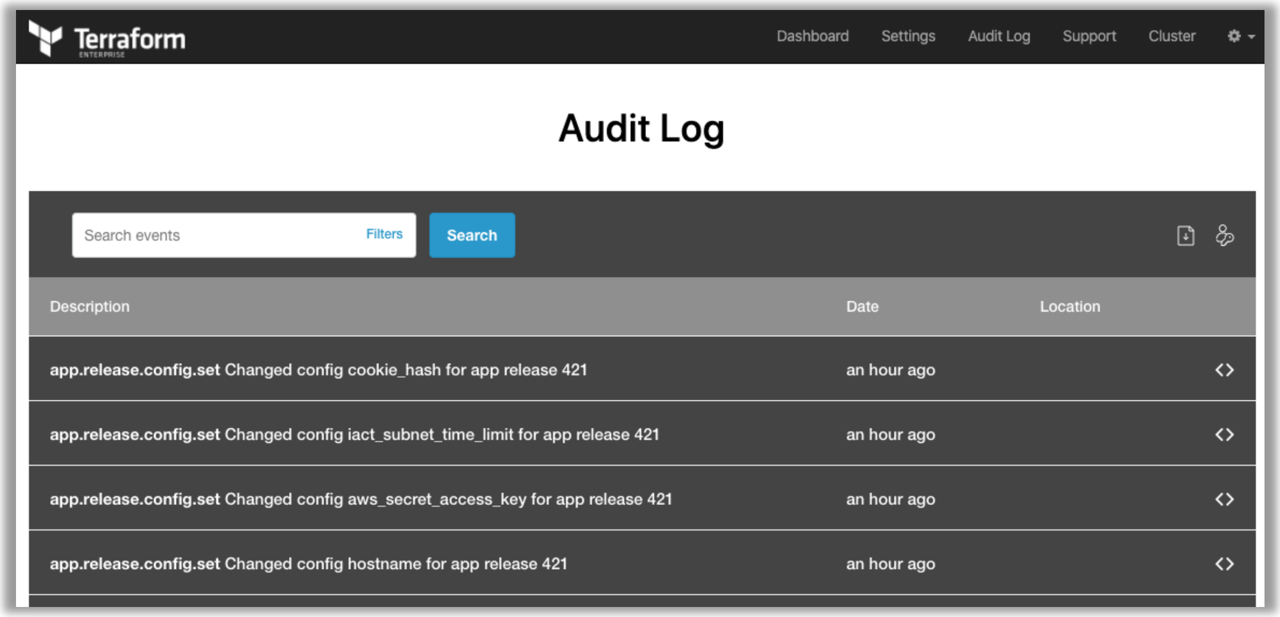
Last checked: 4 minutes ago

Current version: v202003-1

3. Before proceeding, the best practice includes the following important steps:
   a. Take a volume-level snapshot of the disk storage on the instance, including any attached drives.
   b. Make a backup of the PostgreSQL external database.
   c. Make a backup of the associated Azure Blob Storage.
4. Proceed with the upgrade process, as prompted.

Viewing the Audit Logs

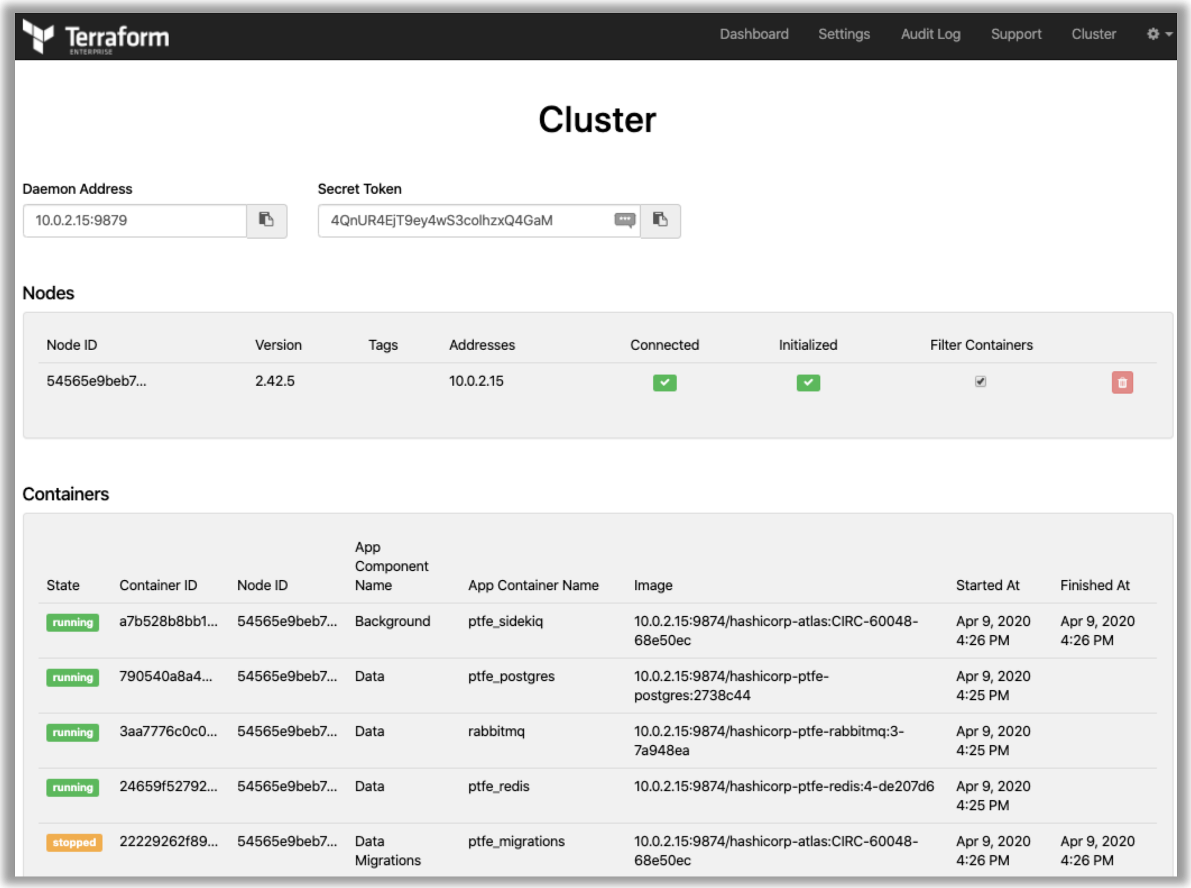
5. On the Admin Console, select **Audit Log** from the upper left navigation bar. This will bring you to the screen below. Note the **Filters** available in the **Search** field. The default is to only show log entries that Create, Update, or Delete. Modify the Filters to see also Read log entries.



Description	Date	Location
app.release.config.set Changed config cookie_hash for app release 421	an hour ago	
app.release.config.set Changed config iact_subnet_time_limit for app release 421	an hour ago	
app.release.config.set Changed config aws_secret_access_key for app release 421	an hour ago	
app.release.config.set Changed config hostname for app release 421	an hour ago	

Viewing the status of the Docker Containers

6. In the Admin Console, select **Cluster** from the navigation menu in the upper right of the page. You'll see a page like the one below, which has detailed information on the status of the various Docker containers that comprise Terraform Enterprise:



Cluster

Daemon Address: 10.0.2.15:9879

Secret Token: 4QnUR4EjT9ey4wS3colhzxQ4GaM

Nodes

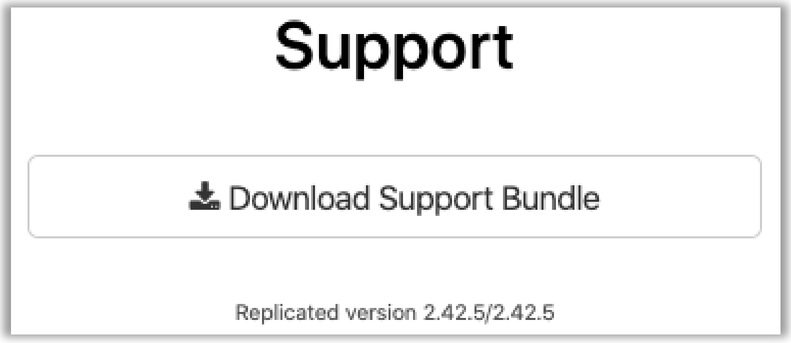
Node ID	Version	Tags	Addresses	Connected	Initialized	Filter Containers
54565e9beb7...	2.42.5		10.0.2.15	✓	✓	✓

Containers

State	Container ID	Node ID	App Component Name	App Container Name	Image	Started At	Finished At
running	a7b528b8bb1...	54565e9beb7...	Background	ptfe_sidekiq	10.0.2.15:9874/hashicorp-atlas:CIRC-60048-68e50ec	Apr 9, 2020 4:26 PM	Apr 9, 2020 4:26 PM
running	790540a8a4...	54565e9beb7...	Data	ptfe_postgres	10.0.2.15:9874/hashicorp-ptfe-postgres:2738c44	Apr 9, 2020 4:25 PM	
running	3aa7776c0c0...	54565e9beb7...	Data	rabbitmq	10.0.2.15:9874/hashicorp-ptfe-rabbitmq:3-7a948ea	Apr 9, 2020 4:25 PM	
running	24659f52792...	54565e9beb7...	Data	ptfe_redis	10.0.2.15:9874/hashicorp-ptfe-redis:4-de207d6	Apr 9, 2020 4:25 PM	
stopped	22229262f89...	54565e9beb7...	Data Migrations	ptfe_migrations	10.0.2.15:9874/hashicorp-atlas:CIRC-60048-68e50ec	Apr 9, 2020 4:26 PM	Apr 9, 2020 4:26 PM

Preparing for a HashiCorp Support Request

1. In the Admin Console, in the upper right navigation bar, click **Support**, and then click **Download Support Bundle**. The file will be saved to your browser's Downloads folder.



2. Next, e-mail this file to support@hashicorp.com or use their Zendesk-backed Support Portal at https://support.hashicorp.com/. A support ticket will then be created.

Connecting TFE Remote State to a local Terraform CLI environment

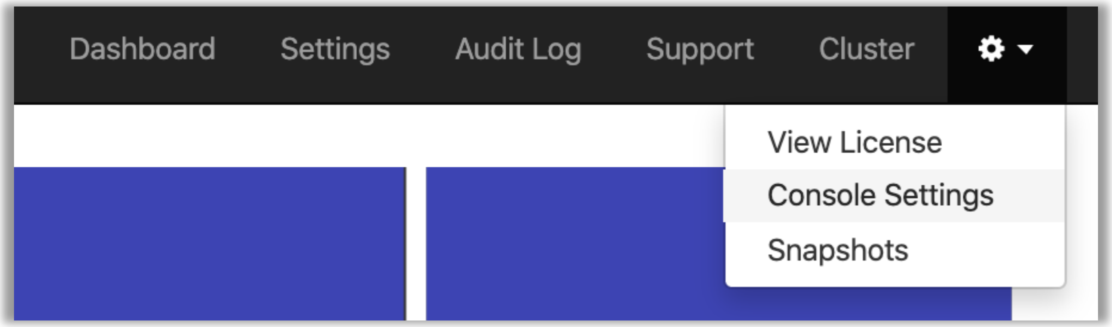
For those who prefer a CLI-based workflow, take a look at the instructions here: https://www.terraform.io/docs/cloud/run/cli.html

TIP: Include the stanzas mentioned here within a file named 'override.tf' in your 'root' module. This file will then be excluded from the code repo via .gitignore.

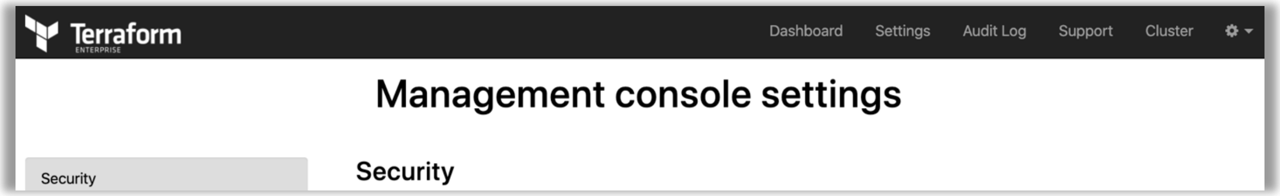
Uploading TLS certificates for HTTP client use

Prerequisite: You have the SSL cert private key and SSL Certificate files on your local workstation (for the Upload Files option seen below).

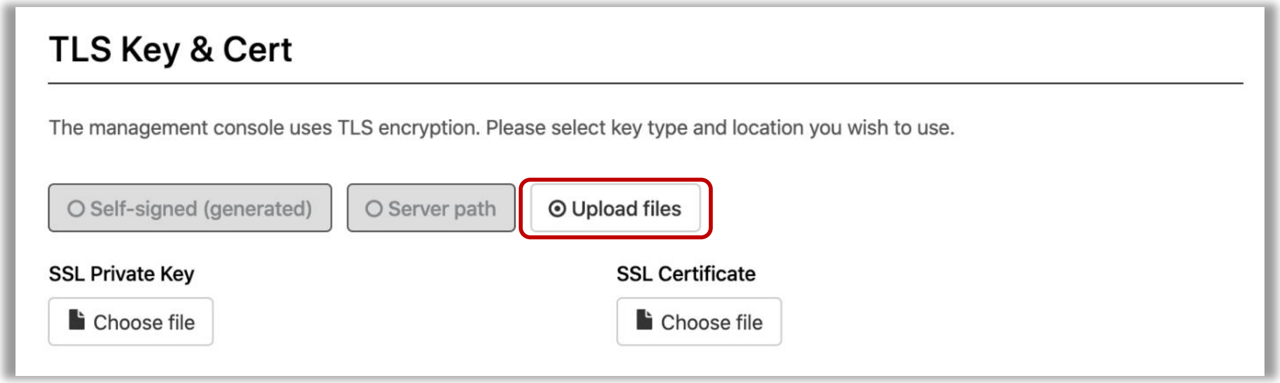
1. In the Admin Console, go to the upper right, select the "gear" dropdown menu, and select **Console Settings**. These are the setting that apply to the TFE Web Application Console itself, NOT the Admin Console.



2. You should see **Management console settings** at the top of the page under the navigation bar.



3. Next, scroll down to **TLS Key & Cert**. Select **(*) Upload Files**, as seen below:



TLS Key & Cert

The management console uses TLS encryption. Please select key type and location you wish to use.

Self-signed (generated) Server path [✓ Upload files]

SSL Private Key
Choose file

SSL Certificate
Choose file

4. Upload the files. Then, scroll to the bottom of the page and click **Save**. If validation tests on your configuration pass, you should be prompted to restart Terraform Enterprise. When you are ready, click **Restart Now**. You may need to clear your browser cache to see the new SSL certificate in use.