Terraform Enterprise Provider

The Terraform Enterprise provider is used to interact with the many resources supported by Terraform Enterprise (https://www.hashicorp.com/products/terraform). It supports both the SaaS version of Terraform Enterprise (app.terraform.io (https://app.terraform.io)) and private instances.

Use the navigation to the left to read about the available resources.

Authentication

This provider requires a Terraform Enterprise API token in order to manage resources.

To manage the full selection of resources, provide a user token (/docs/cloud/users-teams-organizations/users.html#api-tokens) from an account with appropriate permissions. This user should belong to the "owners" team of every Terraform Enterprise organization you wish to manage.

> **Note:** You can use an organization or team token (/docs/cloud/users-teams-organizations/api-tokens.html) instead of a user token, but it will limit which resources you can manage. Organization and team tokens cannot manage resources across multiple organizations, and organization tokens cannot manage certain resource types (like SSH keys). See the Terraform Enterprise API documentation (/docs/cloud/api/index.html) for more details about access to specific resources.

There are three ways to provide the required token:

- On the CLI, omit the `token` argument and set a `credentials` block in your CLI config file (/docs/commands/cli-config.html#credentials).
- Set the `TFE_TOKEN` environment variable.
- In a Terraform Enterprise workspace, set `token` in the provider configuration. Use an input variable for the token and mark the corresponding variable in the workspace as sensitive.

Example Usage

```
# Configure the Terraform Enterprise Provider
provider "tfe" {
  hostname = "${var.hostname}"
  token    = "${var.token}"
}

# Create an organization
resource "tfe_organization" "org" {
  # ...
}
```

Argument Reference

The following arguments are supported:

- `hostname` - (Optional) The Terraform Enterprise hostname to connect to. Defaults to `app.terraform.io`. Can be overridden by setting the `TFE_HOSTNAME` environment variable.
- `token` - (Optional) The token used to authenticate with Terraform Enterprise. Only set this argument when running in a Terraform Enterprise workspace; for CLI usage, omit the token from the configuration and set it as `credentials` in the CLI config file (/docs/commands/cli-config.html#credentials) or set the `TFE_TOKEN` environment variable. See Authentication above for more.

Data Source: tfe_ssh_key

Use this data source to get information about a SSH key.

Example Usage

```
data "tfe_ssh_key" "test" {
  name         = "my-ssh-key-name"
  organization = "my-org-name"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name of the SSH key.
- `organization` - (Required) Name of the organization.

Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the SSH key.

Data Source: tfe_team_access

Use this data source to get information about team permissions for a workspace.

Example Usage

```
data "tfe_team_access" "test" {
  team_id      = "my-team-id"
  workspace_id = "my-workspace-id"
}
```

Argument Reference

The following arguments are supported:

- `team_id` - (Required) ID of the team.
- `workspace_id` - (Required) ID of the workspace.

Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` The team access ID.
- `access` - The type of access granted.

Data Source: tfe_team

Use this data source to get information about a team.

Example Usage

```
data "tfe_team" "test" {
  name         = "my-team-name"
  organization = "my-org-name"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name of the team.
- `organization` - (Required) Name of the organization.

Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the team.

Data Source: tfe_workspace

Use this data source to get information about a workspace.

Example Usage

```
data "tfe_workspace" "test" {
  name         = "my-workspace-name"
  organization = "my-org-name"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name of the workspace.
- `organization` - (Required) Name of the organization.

Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The workspace's human-readable ID, which looks like `<ORGANIZATION>/<WORKSPACE>`.
- `external_id` - The workspace's opaque external ID, which looks like `ws-<RANDOM STRING>`.
- `auto_apply` - Indicates whether to automatically apply changes when a Terraform plan is successful.
- `file_triggers_enabled` - Indicates whether runs are triggered based on the changed files in a VCS push (if `true`) or always triggered on every push (if `false`).
- `operations` - Indicates whether the workspace is using remote execution mode.
- `queue_all_runs` - Indicates whether all runs should be queued.
- `ssh_key_id` - The ID of an SSH key assigned to the workspace.
- `terraform_version` - The version of Terraform used for this workspace.
- `trigger_prefixes` - List of repository-root-relative paths which describe all locations to be tracked for changes.
- `vcs_repo` - Settings for the workspace's VCS repository.
- `working_directory` - A relative path that Terraform will execute within.

The `vcs_repo` block contains:

- `identifier` - A reference to your VCS repository in the format `:org/:repo` where `:org` and `:repo` refer to the organization and repository in your VCS provider.

- `ingress_submodules` - Indicates whether submodules should be fetched when cloning the VCS repository.
- `oauth_token_id` - OAuth token ID of the configured VCS connection.

Data Source: tfe_workspace_ids

Use this data source to get a map of (external) workspace IDs.

Example Usage

```
data "tfe_workspace_ids" "app-frontend" {
  names        = ["app-frontend-prod", "app-frontend-dev1", "app-frontend-staging"]
  organization = "my-org-name"
}

data "tfe_workspace_ids" "all" {
  names        = ["*"]
  organization = "my-org-name"
}
```

Argument Reference

The following arguments are supported:

- `names` - (Required) A list of workspace names to search for. Names that don't match a real workspace will be omitted from the results, but are not an error.

  To select *all* workspaces for an organization, provide a list with a single asterisk, like `["*"]`. No other use of wildcards is supported.

- `organization` - (Required) Name of the organization.

Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `ids` - A map of workspace names and their human-readable IDs, which look like `<ORGANIZATION>/<WORKSPACE>`.
- `external_ids` - A map of workspace names and their opaque external IDs, which look like `ws-<RANDOM STRING>`.

tfe_notification_configuration

Terraform Cloud can be configured to send notifications for run state transitions. Notification configurations allow you to specify a URL, destination type, and what events will trigger the notification. Each workspace can have up to 20 notification configurations, and they apply to all runs for that workspace.

Example Usage

Basic usage:

```
resource "tfe_organization" "test" {
  name  = "my-org-name"
  email = "admin@company.com"
}

resource "tfe_workspace" "test" {
  name         = "my-workspace-name"
  organization = "${tfe_organization.test.id}"
}

resource "tfe_notification_configuration" "test" {
  name                     = "my-test-notification-configuration"
  enabled                  = true
  destination_type         = "generic"
  triggers                 = ["run:created", "run:planning", "run:errored"]
  url                      = "https://example.com"
  workspace_external_id    = "${tfe_workspace.test.external_id}"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name of the notification configuration.
- `destination_type` - (Required) The type of notification configuration payload to send. Valid values are `generic` or `slack`.
- `enabled` - (Optional) Whether the notification configuration should be enabled or not. Disabled configurations will not send any notifications. Defaults to `false`.
- `token` - (Optional) A write-only secure token for the notification configuration, which can be used by the receiving server to verify request authenticity when configured for notification configurations with a destination type of `generic`. A token set for notification configurations with a destination type of `slack` is not allowed and will result in an error. Defaults to `null`.
- `triggers` - (Optional) The array of triggers for which this notification configuration will send notifications. Valid values are `run:created`, `run:planning`, `run:needs_attention`, `run:applying` `run:completed`, `run:errored`. If omitted, no notification triggers are configured.

- `url` - (Required) The HTTP or HTTPS URL of the notification configuration where notification requests will be made.
- `workspace_external_id` - (Required) The external id of the workspace that owns the notification configuration.

Attributes Reference

- `id` - The ID of the notification configuration.

Import

Notification configurations can be imported; use `<NOTIFICATION CONFIGURATION ID>` as the import ID. For example:

```
terraform import tfe_notification_configuration.test nc-qV9JnKRkmtMa4zcA
```

tfe_oauth_client

An OAuth Client represents the connection between an organization and a VCS provider.

Note: This resource does not currently support creation of Bitbucket Server OAuth clients.

Example Usage

Basic usage:

```
resource "tfe_oauth_client" "test" {
  organization     = "my-org-name"
  api_url          = "https://api.github.com"
  http_url         = "https://github.com"
  oauth_token      = "my-vcs-provider-token"
  service_provider = "github"
}
```

Argument Reference

The following arguments are supported:

- `organization` - (Required) Name of the organization.
- `api_url` - (Required) The base URL of your VCS provider's API (e.g. `https://api.github.com` or `https://ghe.example.com/api/v3`).
- `http_url` - (Required) The homepage of your VCS provider (e.g. `https://github.com` or `https://ghe.example.com`).
- `oauth_token` - (Required) The token string you were given by your VCS provider.
- `service_provider` - (Required) The VCS provider being connected with. Valid options are `github`, `github_enterprise`, `bitbucket_hosted`, `gitlab_hosted`, `gitlab_community_edition`, or `gitlab_enterprise_edition`.

Attributes Reference

- `id` - The ID of the OAuth client.
- `oauth_token_id` - The ID of the OAuth token associated with the OAuth client.

tfe_organization

Manages organizations.

Example Usage

Basic usage:

```
resource "tfe_organization" "test" {
  name  = "my-org-name"
  email = "admin@company.com"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name of the organization.
- `email` - (Required) Admin email address.
- `session_timeout_minutes` - (Optional) Session timeout after inactivity. Defaults to `20160`.
- `session_remember_minutes` - (Optional) Session expiration. Defaults to `20160`.
- `collaborator_auth_policy` - (Optional) Authentication policy (`password` or `two_factor_mandatory`). Defaults to `password`.
- `owners_team_saml_role_id` - (Optional) The name of the "owners" team.

Attributes Reference

- `id` - The name of the organization.

Import

Organizations can be imported; use `<ORGANIZATION NAME>` as the import ID. For example:

```
terraform import tfe_organization.test my-org-name
```

tfe_organization_token

Generates a new organization token, replacing any existing token. This token can be used to act as the organization service account.

Example Usage

Basic usage:

```
resource "tfe_organization_token" "test" {
  organization = "my-org-name"
}
```

Argument Reference

The following arguments are supported:

- `organization` - (Required) Name of the organization.
- `force_regenerate` - (Optional) If set to `true` , a new token will be generated even if a token already exists. This will invalidate the existing token!

Attributes Reference

- `id` - The ID of the token.
- `token` - The generated token.

Import

Organization tokens can be imported; use `<ORGANIZATION NAME>` as the import ID. For example:

```
terraform import tfe_organization_token.test my-org-name
```

tfe_policy_set

Sentinel Policy as Code is an embedded policy as code framework integrated with Terraform Enterprise.

Policy sets are groups of policies that are applied together to related workspaces. By using policy sets, you can group your policies by attributes such as environment or region. Individual policies that are members of policy sets will only be checked for workspaces that the policy set is attached to.

Example Usage

Basic usage (VCS-based policy set):

```
resource "tfe_policy_set" "test" {
  name                   = "my-policy-set"
  description            = "A brand new policy set"
  organization           = "my-org-name"
  policies_path          = "policies/my-policy-set"
  workspace_external_ids = ["${tfe_workspace.test.external_id}"]

  vcs_repo {
    identifier         = "my-org-name/my-policy-set-repository"
    branch             = "master"
    ingress_submodules = false
    oauth_token_id     = "${tfe_oauth_client.test.oauth_token_id}"
  }
}
```

Using manually-specified policies:

```
resource "tfe_policy_set" "test" {
  name                   = "my-policy-set"
  description            = "A brand new policy set"
  organization           = "my-org-name"
  policy_ids             = ["${tfe_sentinel_policy.test.id}"]
  workspace_external_ids = ["${tfe_workspace.test.external_id}"]
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name of the policy set.
- `description` - (Optional) A description of the policy set's purpose.
- `global` - (Optional) Whether or not policies in this set will apply to all workspaces. Defaults to `false`. This value *must not* be provided if `workspace_external_ids` are provided.
- `organization` - (Required) Name of the organization.

- `policies_path` - (Optional) The sub-path within the attached VCS repository to ingress when using `vcs_repo`. All files and directories outside of this sub-path will be ignored. This option can only be supplied when `vcs_repo` is present. Forces a new resource if changed.
- `policy_ids` - (Optional) A list of Sentinel policy IDs. This value *must not* be provided if `vcs_repo` is provided.
- `vcs_repo` - (Optional) Settings for the policy sets VCS repository. Forces a new resource if changed. This value *must not* be provided if `policy_ids` are provided.
- `workspace_external_ids` - (Optional) A list of workspace external IDs. This value *must not* be provided if `global` is provided.

Note: When neither `vcs_repo` or `policy_ids` is not specified, the current default is to create an empty non-VCS policy set.

The `vcs_repo` block supports:

- `identifier` - (Required) A reference to your VCS repository in the format `:org/:repo` where `:org` and `:repo` refer to the organization and repository in your VCS provider.
- `branch` - (Optional) The repository branch that Terraform will execute from. Default to `master`.
- `ingress_submodules` - (Optional) Whether submodules should be fetched when cloning the VCS repository. Defaults to `false`.
- `oauth_token_id` - (Required) Token ID of the VCS Connection (OAuth Conection Token) to use.

Attributes Reference

- `id` - The ID of the policy set.

Import

Policy sets can be imported; use `<POLICY SET ID>` as the import ID. For example:

```
terraform import tfe_policy_set.test polset-wAs3zYmWAhYK7peR
```

tfe_sentinel_policy

Sentinel Policy as Code is an embedded policy as code framework integrated with Terraform Enterprise.

Policies are configured on a per-organization level and are organized and grouped into policy sets, which define the workspaces on which policies are enforced during runs.

Example Usage

Basic usage:

```
resource "tfe_sentinel_policy" "test" {
  name         = "my-policy-name"
  description  = "This policy always passes"
  organization = "my-org-name"
  policy       = "main = rule { true }"
  enforce_mode = "hard-mandatory"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name of the policy.
- `description` - (Optional) A description of the policy's purpose.
- `organization` - (Required) Name of the organization.
- `policy` - (Required) The actual policy itself.
- `enforce_mode` - (Required) The enforcement level of the policy. Valid values are `advisory`, `hard-mandatory` and `soft-mandatory`. Defaults to `soft-mandatory`.

Attributes Reference

- `id` - The ID of the policy.

Import

Sentinel policies can be imported; use `<ORGANIZATION NAME>/<POLICY ID>` as the import ID. For example:

```
terraform import tfe_sentinel_policy.test my-org-name/pol-wAs3zYmWAhYK7peR
```

tfe_ssh_key

This resource represents an SSH key which includes a name and the SSH private key. An organization can have multiple SSH keys available.

Example Usage

Basic usage:

```
resource "tfe_ssh_key" "test" {
  name         = "my-ssh-key-name"
  organization = "my-org-name"
  key          = "private-ssh-key"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name to identify the SSH key.
- `organization` - (Required) Name of the organization.
- `key` - (Required) The text of the SSH private key.

Attributes Reference

- `id` The ID of the SSH key.

Import

Because the Terraform Enterprise API does not return the private SSH key content, this resource cannot be imported.

tfe_team_access

Associate a team to permissions on a workspace.

Example Usage

Basic usage:

```
resource "tfe_team" "test" {
  name         = "my-team-name"
  organization = "my-org-name"
}

resource "tfe_workspace" "test" {
  name         = "my-workspace-name"
  organization = "my-org-name"
}

resource "tfe_team_access" "test" {
  access       = "read"
  team_id      = "${tfe_team.test.id}"
  workspace_id = "${tfe_workspace.test.id}"
}
```

Argument Reference

The following arguments are supported:

- `access` - (Required) Type of access to grant. Valid values are `admin`, `read`, `plan`, or `write`.
- `team_id` - (Required) ID of the team to add to the workspace.
- `workspace_id` - (Required) The workspace to which the team will be added, specified as a human-readable ID (`<ORGANIZATION>/<WORKSPACE>`).

Attributes Reference

- `id` The team access ID.

Import

Team accesses can be imported; use `<ORGANIZATION NAME>/<WORKSPACE NAME>/<TEAM ACCESS ID>` as the import ID. For example:

```
terraform import tfe_team_access.test my-org-name/my-workspace-name/tws-8S5wnRbRpogw6apb
```

tfe_team

Manages teams.

Example Usage

Basic usage:

```
resource "tfe_team" "test" {
  name         = "my-team-name"
  organization = "my-org-name"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name of the team.
- `organization` - (Required) Name of the organization.

Attributes Reference

- `id` The ID of the team.

Import

Teams can be imported; use `<ORGANIZATION NAME>/<TEAM ID>` as the import ID. For example:

```
terraform import tfe_team.test my-org-name/team-uomQZysH9ou42ZYY
```

tfe_team_member

Add or remove a user from a team.

NOTE on managing team memberships: Terraform currently provides two resources for managing team memberships. The tfe_team_member (/docs/providers/tfe/r/team_member.html) resource can be used multiple times as it manages the team membership for a single user. The tfe_team_members (/docs/providers/tfe/r/team_members.html) resource, on the other hand, is used to manage all team memberships for a specific team and can only be used once. Both resources cannot be used for the same team simultaneously.

Example Usage

Basic usage:

```
resource "tfe_team" "test" {
  name         = "my-team-name"
  organization = "my-org-name"
}

resource "tfe_team_member" "test" {
  team_id  = "${tfe_team.test.id}"
  username = "sander"
}
```

Argument Reference

The following arguments are supported:

- `team_id` - (Required) ID of the team.
- `username` - (Required) Name of the user to add.

Import

A team member can be imported; use `<TEAM ID>/<USERNAME>` as the import ID. For example:

```
terraform import tfe_team_member.test team-47qC3LmA47piVan7/sander
```

tfe_team_members

Manages users in a team.

NOTE on managing team memberships: Terraform currently provides two resources for managing team memberships. The tfe_team_member (/docs/providers/tfe/r/team_member.html) resource can be used multiple times as it manages the team membership for a single user. The tfe_team_members (/docs/providers/tfe/r/team_members.html) resource, on the other hand, is used to manage all team memberships for a specific team and can only be used once. Both resources cannot be used for the same team simultaneously.

Example Usage

Basic usage:

```
resource "tfe_team" "test" {
  name         = "my-team-name"
  organization = "my-org-name"
}

resource "tfe_team_members" "test" {
  team_id   = "${tfe_team.test.id}"
  usernames = ["admin", "sander"]
}
```

Argument Reference

The following arguments are supported:

- `team_id` - (Required) ID of the team.
- `usernames` - (Required) Names of the users to add.

Attributes Reference

- `id` - The ID of the team.

Import

Team members can be imported; use `<TEAM ID>` as the import ID. For example:

```
terraform import tfe_team_members.test team-47qC3LmA47piVan7
```

tfe_team_token

Generates a new team token and overrides existing token if one exists.

Example Usage

Basic usage:

```
resource "tfe_team" "test" {
  name         = "my-team-name"
  organization = "my-org-name"
}

resource "tfe_team_token" "test" {
  team_id = "${tfe_team.test.id}"
}
```

Argument Reference

The following arguments are supported:

- `team_id` - (Required) ID of the team.
- `force_regenerate` - (Optional) If set to `true`, a new token will be generated even if a token already exists. This will invalidate the existing token!

Attributes Reference

- `id` - The ID of the token.
- `token` - The generated token.

Import

Team tokens can be imported; use `<TEAM ID>` as the import ID. For example:

```
terraform import tfe_team_token.test team-47qC3LmA47piVan7
```

tfe_variable

Creates, updates and destroys variables.

Example Usage

Basic usage:

```
resource "tfe_organization" "test" {
  name  = "my-org-name"
  email = "admin@company.com"
}

resource "tfe_workspace" "test" {
  name         = "my-workspace-name"
  organization = "${tfe_organization.test.id}"
}

resource "tfe_variable" "test" {
  key          = "my_key_name"
  value        = "my_value_name"
  category     = "terraform"
  workspace_id = "${tfe_workspace.test.id}"
}
```

Argument Reference

The following arguments are supported:

- `key` - (Required) Name of the variable.
- `value` - (Required) Value of the variable.
- `category` - (Required) Whether this is a Terraform or environment variable. Valid values are `terraform` or `env`.
- `hcl` - (Optional) Whether to evaluate the value of the variable as a string of HCL code. Has no effect for environment variables. Defaults to `false`.
- `sensitive` - (Optional) Whether the value is sensitive. If true then the variable is written once and not visible thereafter. Defaults to `false`.
- `workspace_id` - (Required) The workspace that owns the variable, specified as a human-readable ID (`<ORGANIZATION>/<WORKSPACE>`).

Attributes Reference

- `id` - The ID of the variable.

Import

Variables can be imported; use `<ORGANIZATION NAME>/<WORKSPACE NAME>/<VARIABLE ID>` as the import ID. For example:

```
terraform import tfe_variable.test my-org-name/my-workspace-name/var-5rTwnSaRPogw6apb
```

tfe_workspace

Provides a workspace resource.

Example Usage

Basic usage:

```
resource "tfe_workspace" "test" {
  name         = "my-workspace-name"
  organization = "my-org-name"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) Name of the workspace.
- `organization` - (Required) Name of the organization.
- `auto_apply` - (Optional) Whether to automatically apply changes when a Terraform plan is successful. Defaults to `false`.
- `file_triggers_enabled` - (Optional) Whether to filter runs based on the changed files in a VCS push. If enabled, the working directory and trigger prefixes describe a set of paths which must contain changes for a VCS push to trigger a run. If disabled, any push will trigger a run. Defaults to `true`.
- `operations` - (Optional) Whether to use remote execution mode. When set to `false`, the workspace will be used for state storage only. Defaults to `true`.
- `queue_all_runs` - (Optional) Whether all runs should be queued. When set to `false`, runs triggered by a VCS change will not be queued until at least one run is manually queued. Defaults to `true`.
- `ssh_key_id` - (Optional) The ID of an SSH key to assign to the workspace.
- `terraform_version` - (Optional) The version of Terraform to use for this
- `trigger_prefixes` - (Optional) List of repository-root-relative paths which describe all locations to be tracked for changes. workspace. Defaults to the latest available version.
- `working_directory` - (Optional) A relative path that Terraform will execute within. Defaults to the root of your repository.
- `vcs_repo` - (Optional) Settings for the workspace's VCS repository.

The `vcs_repo` block supports:

- `identifier` - (Required) A reference to your VCS repository in the format `:org/:repo` where `:org` and `:repo` refer to the organization and repository in your VCS provider.

- `branch` - (Optional) The repository branch that Terraform will execute from. Default to `master`.
- `ingress_submodules` - (Optional) Whether submodules should be fetched when cloning the VCS repository. Defaults to `false`.
- `oauth_token_id` - (Required) Token ID of the VCS Connection (OAuth Conection Token) to use.

Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The workspace's human-readable ID, which looks like `<ORGANIZATION>/<WORKSPACE>`.
- `external_id` - The workspace's opaque external ID, which looks like `ws-<RANDOM STRING>`.

Import

Workspaces can be imported; use `<ORGANIZATION NAME>/<WORKSPACE NAME>` as the import ID. For example:

```
terraform import tfe_workspace.test my-org-name/my-workspace-name
```