Infrastructure Provisioning - Terraform Cloud

Overview

Terraform Cloud is a platform that performs Terraform runs to provision infrastructure, either on demand or in response to various events. Unlike a general-purpose continuous integration (CI) system, it is deeply integrated with Terraform's workflows and data, which allows it to make Terraform significantly more convenient and powerful.

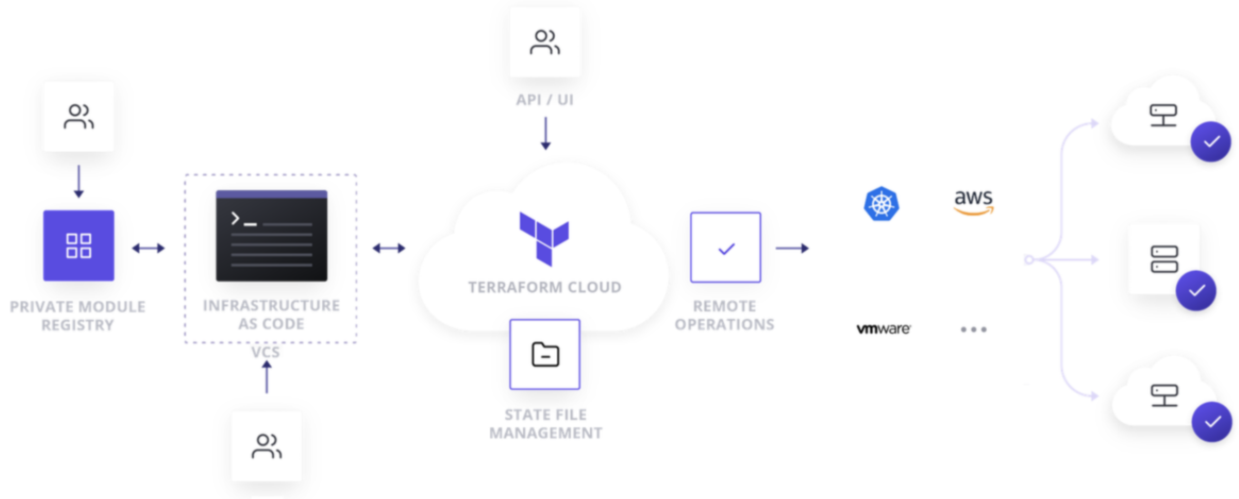
Terraform Cloud is used for the following:

- Provision and manage infrastructure.
- Help teams use Terraform together.
- Manage Terraform runs in a consistent and reliable environment.
- Includes easy access to shared state and secret data, access controls for approving changes to infrastructure, a private registry for sharing Terraform modules, and detailed policy controls for governing the contents of Terraform configurations.

Terraform cloud is available as a hosted service at https://app.terraform.io. Terraform cloud

Content

Terraform Cloud



Overview of features:

Terraform Workflow

- Terraform Cloud runs Terraform CLI to provision infrastructure.
- Since teams must share responsibilities and awareness to avoid single points of failure, working with Terraform in a team requires a remote workflow. At minimum, state must be shared; ideally, Terraform should execute in a consistent remote environment.
- The foundations of this workflow are remote Terraform execution, a workspace-based organizational model, version control integration, command-line integration, remote state management with cross-workspace data sharing, and a private Terraform module registry.

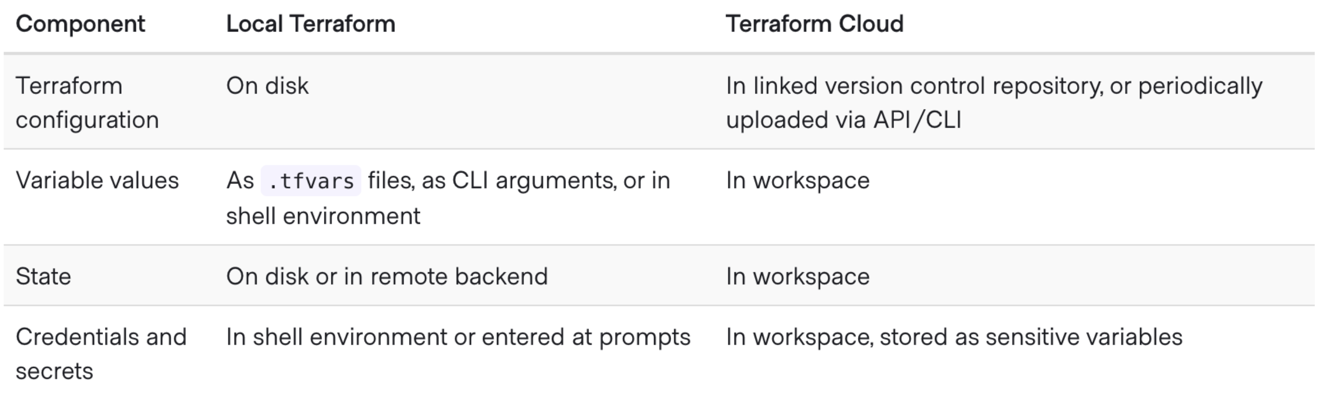
Remote Terraform Execution

- Terraform Cloud runs Terraform on disposable virtual machines in its own cloud infrastructure. Remote Terraform execution is sometimes referred to as "remote operations."
- **Remote execution helps provide consistency and visibility for critical provisioning operations**. It also enables powerful features like Sentinel policy enforcement, cost estimation, notifications, version control integration, and more.
  - Terraform runs and remote operations:
    - Remote runs can be **initiated by webhooks from your VCS provider**, by UI controls within Terraform Cloud, by API calls, or by Terraform CLI. When using Terraform CLI to perform remote operations, the progress of the run is streamed to the user's terminal, to provide an experience equivalent to local operations.

Workspaces

- Terraform's local workflow manages a **collection of infrastructure** with a *persistent working directory,* which contains configuration, state data, and variables. Practitioners can use separate directories to organize infrastructure resources into meaningful groups, and Terraform will use content from whichever directory it is invoked from.
- Terraform Cloud organizes infrastructure with *workspaces* instead of directories. Each workspace contains everything necessary to manage a given collection of infrastructure, and Terraform uses that content whenever it executes in the context of that workspace.
  - Workspaces are **collections of infrastructure**
  - When run **locally**, Terraform manages each collection of infrastructure with a persistent working **directory**, which contains a configuration, state data, and variables. Since Terraform CLI uses content from the directory it runs in, you can organize infrastructure resources into meaningful groups by keeping their configurations in separate directories.

Workspace Content



Component	Local Terraform	Terraform Cloud
Terraform configuration	On disk	In linked version control repository, or periodically uploaded via API/CLI
Variable values	As `.tfvars` files, as CLI arguments, or in shell environment	In workspace
State	On disk or in remote backend	In workspace
Credentials and secrets	In shell environment or entered at prompts	In workspace, stored as sensitive variables

Remote State Management, Data Sharing, and Run Triggers

- Terraform Cloud acts as a remote backend for your Terraform state. **State storage is tied to workspaces**, which helps keep state associated with the configuration that created it.
- Terraform Cloud also enables you to share information **between workspaces** with root-level **outputs**. Separate groups of infrastructure resources often need to share a small amount of information, and workspace outputs are an ideal interface for these dependencies.
- Any workspace that uses remote operations can use `terraform_remote_state` data sources to access other workspaces' outputs, without any additional configuration or authentication. And since new information from one workspace might change the desired infrastructure state in another, you can create workspace-to-workspace run triggers to ensure downstream workspaces react when their dependencies change.

Version Control Integration

- Like other kinds of code, infrastructure-as-code belongs in version control, so Terraform Cloud is designed to work directly with your version control system (VCS) provider.
- Each workspace can be linked to a VCS repository that contains its Terraform configuration, optionally specifying a branch and subdirectory. Terraform Cloud automatically retrieves configuration content from the repository, and will also watch the repository for changes:
  - When new commits are merged, linked workspaces automatically run Terraform plans with the new code.
  - When pull requests are opened, linked workspaces run speculative plans with the proposed code changes and post the results as a pull request check; reviewers can see at a glance whether the plan was successful, and can click through to view the proposed changes in detail.

Command Line Integration

- Remote CLI-driven runs use the current working directory's Terraform configuration and the remote workspace's variables, so you don't need to obtain production cloud credentials just to preview a configuration change.

Private Module Registry

- Teams can benefit greatly by codifying commonly used infrastructure patterns into reusable **modules**
- Terraform CLI can already fetch modules from arbitrary VCS sources, but Terraform Cloud improves this with a private module registry. Users throughout your organization can browse a directory of internal modules, and can specify flexible version constraints for the modules they use in their configurations. Easy versioning lets downstream teams use modules with confidence, and frees upstream teams to iterate faster.
- The private registry uses your VCS as the source of truth, relying on Git tags to manage module versions. Tell Terraform Cloud which repositories contain modules, and the registry handles the rest.

Access Control and Governance

- Larger organizations are more complex, and tend to use access controls and explicit policies to help manage that complexity. Terraform Cloud's paid upgrade plans provide extra features to help meet the control and governance needs of large organizations.
- Team-Based Permissions System
  - With Terraform Cloud's team management, you can define groups of users that match your organization's real-world teams and assign them only the permissions they need. When combined with the access controls your VCS provider already offers for code, workspace permissions are an effective way to follow the principle of least privilege.

- Sentinel Policy
  - Terraform Cloud embeds the Sentinel policy-as-code framework, which lets you define and enforce granular policies for how your organization provisions infrastructure. You can limit the size of compute VMs, confine major updates to defined maintenance windows, and much more.
  - Policies can act as firm requirements, advisory warnings, or soft requirements that can be bypassed with explicit approval from your compliance team.
- Cost Estimation
  - Before making changes to infrastructure in the major cloud providers, Terraform Cloud can display an estimate of its total cost, as well as any change in cost caused by the proposed updates. Cost estimates can also be used in Sentinel policies to provide warnings for major price shifts.

Resources

Hands-on exercises:

- https://learn.hashicorp.com/terraform?track=cloud-gettingstarted#cloud-gettingstarted
- https://learn.hashicorp.com/terraform/tfc/tfc_migration