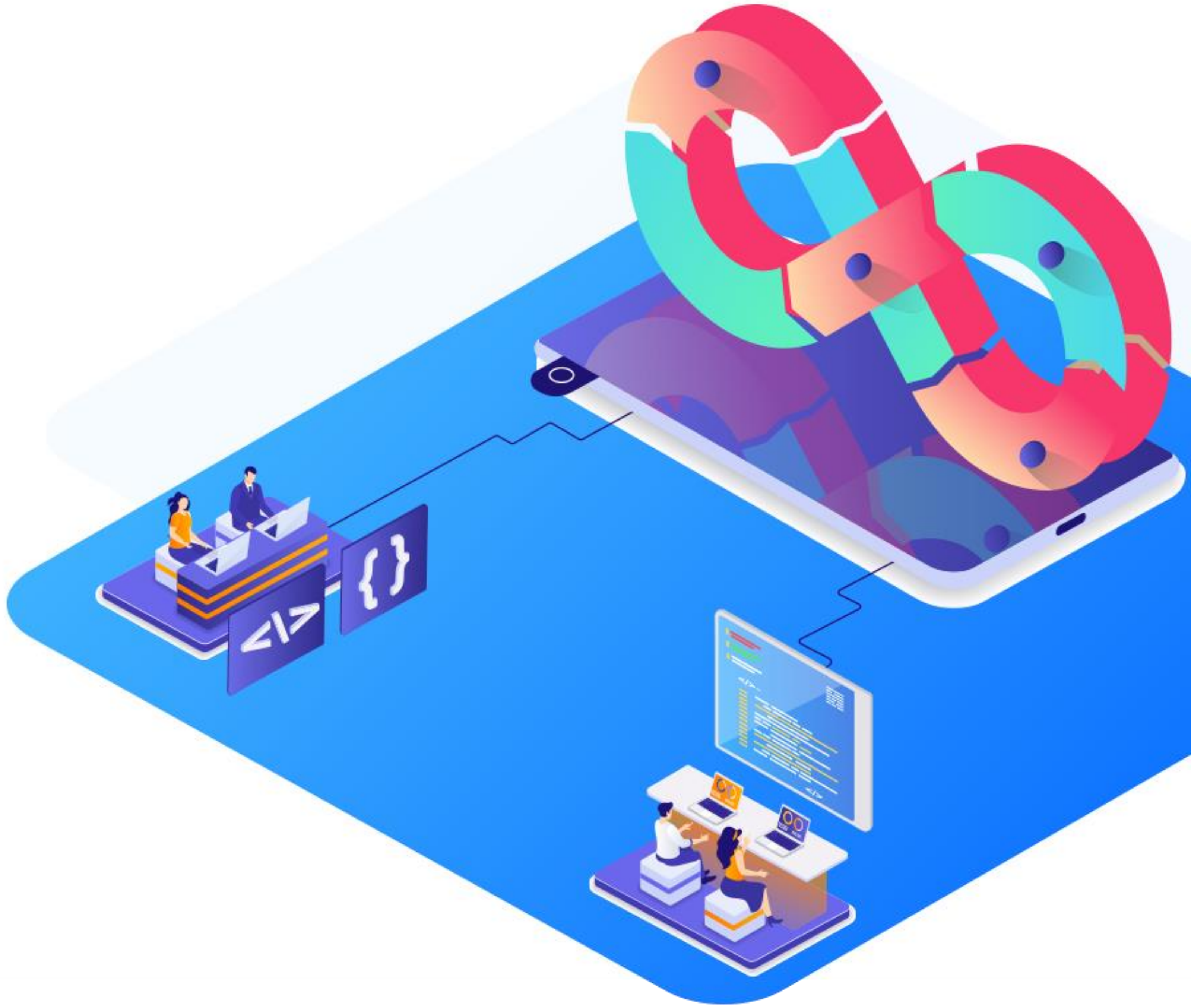
Configuration Management with Ansible and Terraform



Terraform Cloud

Learning Objectives

By the end of this lesson, you will be able to:

- Evaluate the benefits of using Terraform Cloud for team-based infrastructure management, focusing on its capabilities for enhancing collaboration, security, and consistency
- Integrate Terraform Cloud with a version control system (VCS) to automate infrastructure workflows
- Execute the private registry's functions
- Outline the role and implementation of Sentinel policies in Terraform Cloud, emphasizing how they enforce compliance and governance across Terraform projects

Learning Objectives

By the end of this lesson, you will be able to:

- Create a workspace in Terraform Cloud, configure it to manage Terraform configurations and state, utilizing VCS integration for real-time updates and collaboration
- Perform the initial setup of a Terraform Cloud environment by configuring backends, setting up workspaces, and establishing secure management of state files
- Construct effective Sentinel policies that ensure infrastructure compliance with organizational standards and regulatory requirements, using Terraform Cloud's policy-as-code framework

Introduction to Terraform Cloud and State

What Is Terraform Cloud?

It is a platform developed by HashiCorp that helps to manage Terraform code. It enhances collaboration between developers and DevOps engineers.

It allows organizations to enable audit logging, continuous validation, and automated configuration drift detection.

Why Use HCP Terraform Cloud ?

HashiCorp Terraform Cloud offers several advantages for managing infrastructure as code, particularly in team environments.

01 Provides secure access to shared state and secret data, centralizing sensitive information management

02 Features robust controls for approving infrastructure changes, enhancing security

03 Offers a private registry for sharing Terraform modules within your organization

Terraform Cloud: Getting Started

Steps for setting up Terraform Cloud:

01	Sign up
02	Create organization
03	Create first workspace
04	Configure variables

Terraform Cloud: Getting Started

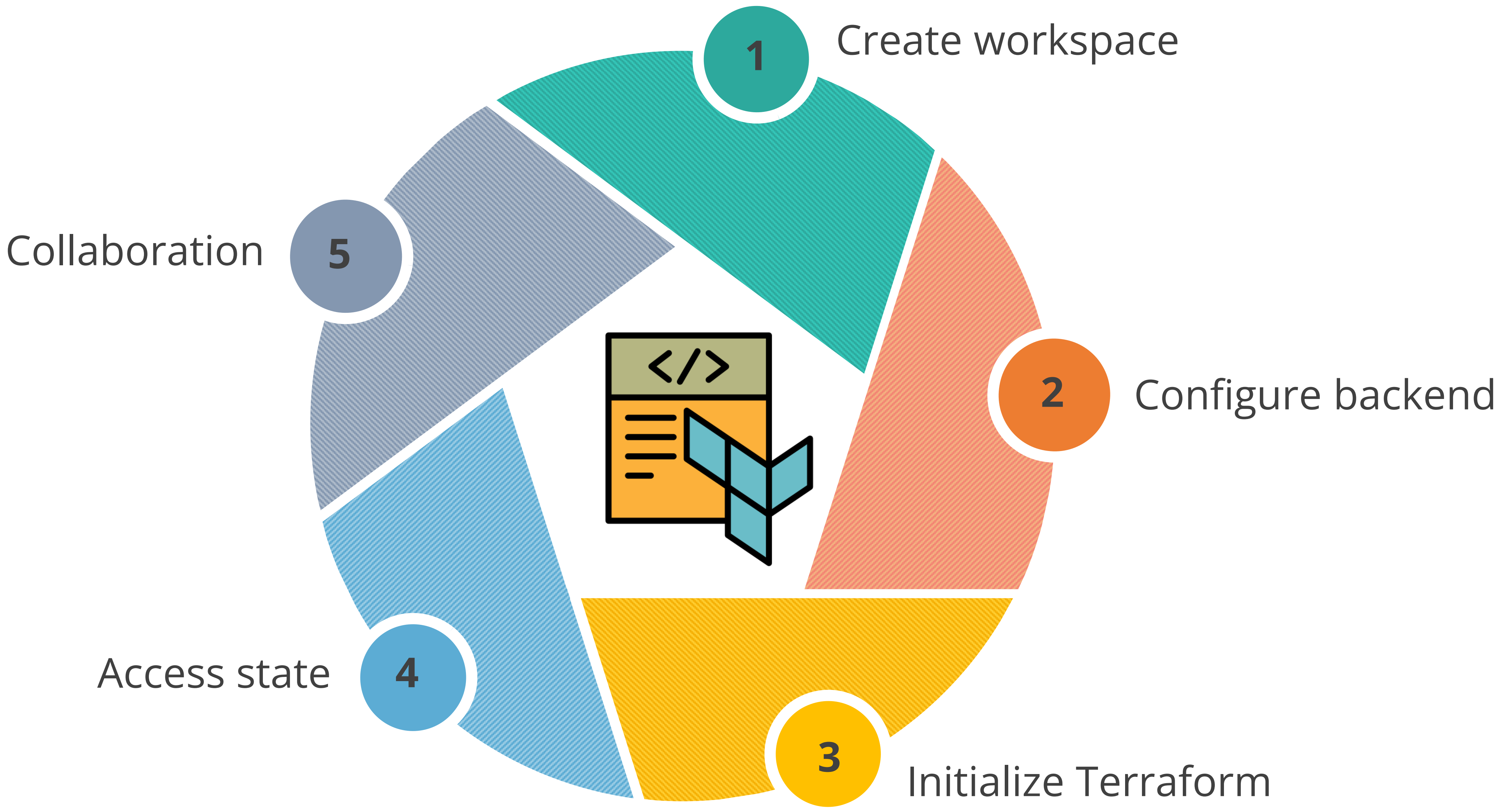
05	Set up workflows
06	Manage access
07	Explore advanced features
08	Monitor and maintain

Terraform Cloud: Remote State

It involves storing the state file of your Terraform configuration in a remote shared store. This can be accessed by your team to ensure everyone is working with the latest configuration.

Setting up Remote State in Terraform Cloud

Here are the steps involved in setting up remote state in Terraform cloud:



Remote State: Benefits

It simplifies infrastructure management by offering a secure, robust, and collaborative platform, especially valuable in team environments.

Centralization	It centrally stores state files, allowing team-wide access to current state information.
Security	State file access is managed by user roles, ensuring only authorized changes to infrastructure.
Locking	Terraform Cloud automatically locks state files during operations, preventing conflicting changes.

Quick Check

As a project manager in a software development company, you are evaluating the implementation of Terraform Cloud for your team's infrastructure management. Which benefit of using Terraform Cloud's remote state feature is most crucial for enhancing team collaboration?

A. It simplifies infrastructure management

B. It centrally stores state files, allowing team-wide access to current state information

C. It manages access based on user roles

D. It automatically locks state files during operations

Workspaces, Version Control, and Variables

Terraform Cloud: Workspaces

These are a fundamental organizational unit that facilitates managing Infrastructure as Code (IaC) with distinct sets of configurations and resources.

1. Each workspace has its own configurations and state, supporting separate management of different deployment stages.

2. Each workspace is linked with Version Control Systems (VCS), which triggers automatic Terraform executions whenever changes are made to associated repositories.

3. Each workspace has fine-grained access controls, ensuring only authorized users can make changes.

Workspaces : Contents

Though Terraform workspaces and local working directories serve the same purpose, they store data differently.

Component	Local Terraform	HCP Terraform
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

Workspaces : Usage Scenarios

Development or testing

Developers use separate workspaces for safe testing of new features without affecting main infrastructure

Multi-region deployments

Distinct workspaces manage resources across various regions, providing customized configurations and autonomous management.

Module testing

Workspaces enable testing of Terraform modules under various conditions to ensure robust functionality

Terraform Cloud: Secure Variables

It refers to sensitive values that are marked as confidential within Terraform configurations to prevent exposure in logs or Terraform Cloud UI.

This includes credentials, API keys, passwords, and other sensitive data.

Secure Variables: Types

The features of the two types of secure variables you can manage in HCP Terraform are:'

Environment variables	Terraform variables
• Configures the execution environment for Terraform runs	• Defines parameters within Terraform configurations
• Applies globally to the workspace	• Specific to Terraform modules or resources
• Manages dynamic credentials for cloud providers	• Input configurable options without altering core configuration

Terraform Cloud: Version Control Integration(VCS)

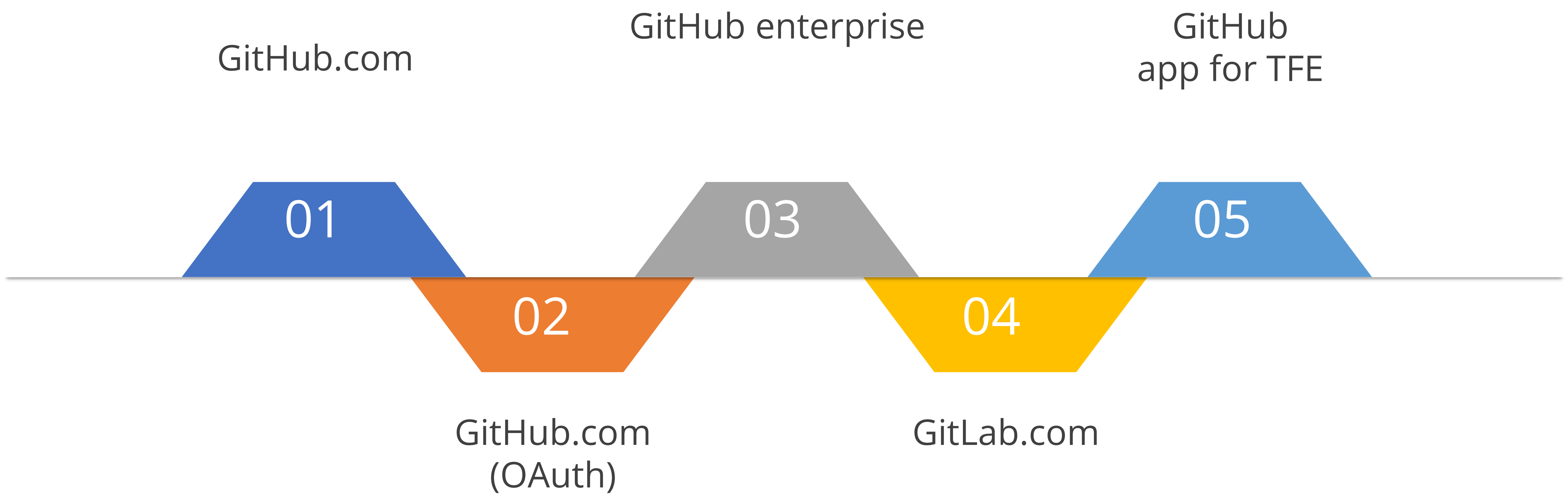
It allows you to connect your Terraform configuration directly to a repository in a VCS.

The key features include:

1. Automated workflow
2. Code review and collaboration
3. Security and access control
4. Change tracking and audit trail

Version Control Integration: Supported VCS Providers

The following VCS providers are compatible with Terraform:



Version Control Integration: Supported VCS Providers

01 Bitbucket cloud

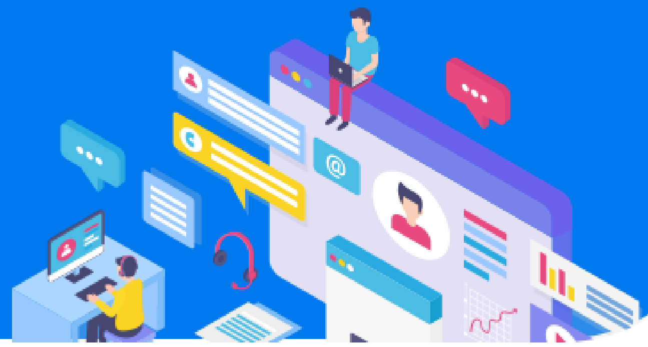
02 Bitbucket data centre

03 Azure DevOps server

04 Azure DevOps services

05 GitLab EE and CE

Assisted Practice



Working with Workspaces Terraform Cloud **Duration: 15 Min.**

Problem statement:

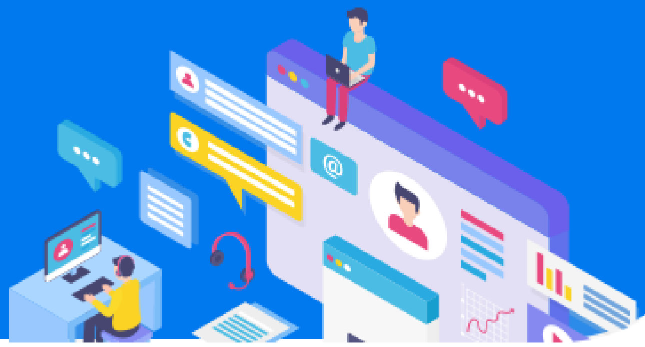
To use Terraform Cloud to manage infrastructure by creating and managing workspaces for deploying resources

Outcome:

you will successfully use Terraform Cloud to manage infrastructure by creating and managing workspaces for deploying resources.

Note: Refer to the demo document for detailed steps

Assisted Practice: Guidelines



Steps to be followed:

1. Sign in to Terraform Cloud platform
2. Create an organization and workspace
3. Initialize Terraform
4. Plan and apply the configurations
5. Select and create workspaces

Assisted Practice

Working with variables and versions on Terraform Cloud **Duration: 10 Min.**

Problem statement:

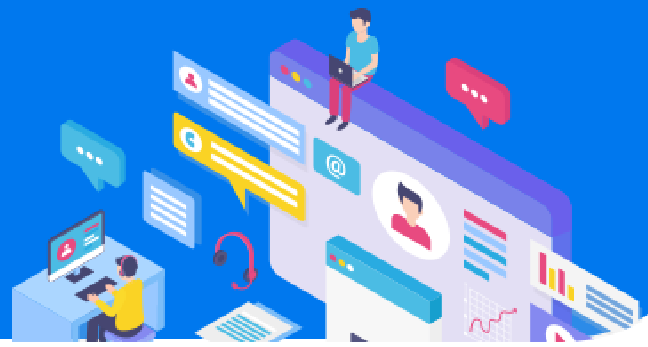
To use Terraform Cloud for managing infrastructure by creating workspaces and defining variables to ensure consistent and reliable deployments

Outcome:

you can effectively work with variables and versions in Terraform Cloud, ensuring a collaborative and managed approach to infrastructure provisioning.

Note: Refer to the demo document for detailed steps

Assisted Practice: Guidelines



Steps to be followed:

1. Sign in to Terraform Cloud platform
2. Create an organization and workspace
3. Configure Terraform CLI for Terraform Cloud
4. Define and use variables on Terraform Cloud
5. Run the Terraform plan and apply

Quick Check

As a Terraform Cloud workspace administrator, you need to set up secure management for cloud provider credentials. Which type of variable should you primarily use to securely manage these credentials across your Terraform runs?

A. Terraform input variables

B. Environment variables

C. Local variables

D. Plaintext variables

Registry and VCS

What Is a Registry?

It is a public repository of Terraform modules and providers that can be used by Terraform users to find and share reusable components for their infrastructure projects.

It includes solutions developed by HashiCorp, third-party vendors, and the Terraform community.

Registry: Navigation

It is organized into categories for modules, providers, and policies to simplify navigation. To navigate The steps for navigating the registry are given below::

01	Click on a provider or module card to access detailed information about that item
02	Utilize the search bar at the top of the registry for quick access
03	Filter search results by specific tiers to quickly find the most relevant modules or providers

Registry: User Account Management

Follow instructions for users interested in publishing modules, providers, and policy libraries.'

GitHub integration

Sign in to the Terraform registry using a GitHub account to publish content

Sign-in process

Click the 'Sign in' button and follow the prompts to authorize and access your GitHub account

Publishing access

Follow the detailed instructions that are available for users interested in publishing modules, providers, and policy libraries

Registry: Getting Help and Support

Resources for users to access help and support in Terraform Cloud:

Maintenance and support	Modules, providers, and policies are maintained by HashiCorp, trusted partners, or the Terraform community
Provider tiers and namespaces	Important references for understanding the support and reliability level of each provider
Issue reporting	Use **Report an Issue** to flag problems or contribute to GitHub repositories

Terraform Cloud: Private Registry

It works similarly to the public Terraform Registry and helps you share Terraform providers and Terraform modules across your organization.

It serves as a centralized repository for storing and versioning Terraform modules, which are collections of Terraform resources that are used to create infrastructure as code (IaC).

Implementing a Private Registry

To set up a private registry, configure your Terraform cloud workspace to include a private module registry. This setup typically requires you to:

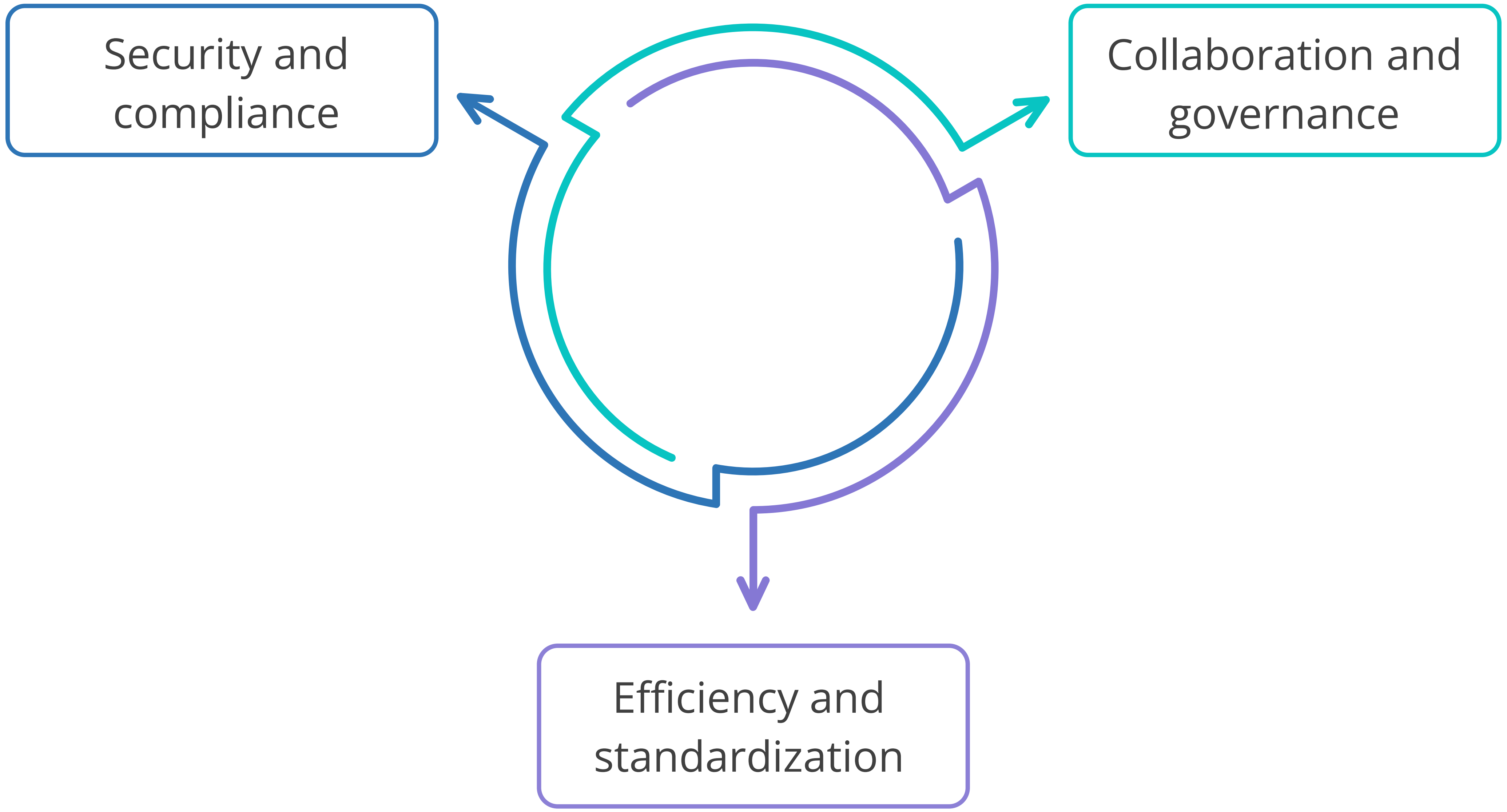
01 Connect your Terraform Cloud workspace to your VCS

02 Configure access controls and permissions according to your organization’s policies

03 Populate the registry with initial modules and establish guidelines for module development and maintenance

Benefits of Using a Private Registry

Private registries have the following advantages:



Terraform Cloud: Sentinel Policy

It provides a powerful layer of policy enforcement that helps ensure your infrastructure meets the necessary standards and regulations before it's provisioned.

The key features of Sentinel policy are:

Policy as code

Written in high-level declarative language. Designed to be readable and writable by individuals who are not necessarily developers.

Simulated testing

Sentinel includes a testing framework that allows policy authors to write tests for their policies.

Sentinel Imports

HCP Terraform provides four imports to define policy rules for the plan, configuration, state, and run associated with a policy check:

- **tfplan**
- **tfconfig**
- **tfstate**
- **tfrun**

Sentinel Policies: Functions and Idioms

The following functions and idioms will be useful as you start writing Sentinel policies for Terraform :

1. Iterate over modules to find resources
2. Validate resource attributes
3. Write rules
4. Validate multiple conditions in a single policy

Terraform Cloud: VCS Workflow

In addition to the CLI-driven workflow, HCP Terraform offers a VCS-driven workflow that automatically triggers runs based on changes to your VCS repositories.

The key features of VCS workflow are:

It enables collaboration within teams by establishing your shared repositories as the source of truth for infrastructure configuration.

It enables teams to manage their infrastructure as code in a systematic and controlled manner.

Benefits of Using VCS Workflow

The advantages of using VCS Workflow are given below:

Enhanced collaboration and review

Consistency and reliability

Audit and compliance

Assisted Practice

Working with Private Registry on Terraform Cloud **Duration: 15 Min.**

Problem statement:

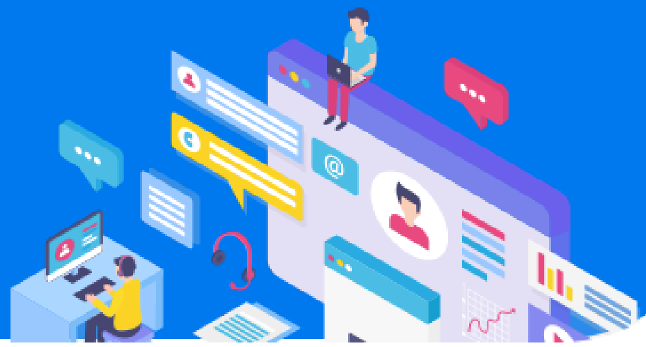
To demonstrate how to work with a private module registry on Terraform Cloud, enabling the use and management of modules within a private organization

Outcome:

You will successfully set up a private module registry on Terraform Cloud, publish a module, and use it in a Terraform configuration.

Note: Refer to the demo document for detailed steps

Assisted Practice: Guidelines



Steps to be followed:

1. Set up Terraform Cloud account and organization
2. Create and publish a module to the private registry
3. Initialize the Terraform Configuration Repository

Quick Check

You have been tasked with setting up a private Registry in your company's Terraform Cloud environment to enhance module management and security. After initial planning, you are determining the first critical step in the implementation process. What should you prioritize to start the setup effectively?

A. Import all existing public modules from the Terraform Public Registry
B. Connect your Terraform Cloud workspace to your VCS
C. Conduct a training session for all IT staff on how to use Terraform
D. Update all existing modules to the latest version before importing

Key Takeaways

- Enhances team collaboration and consistency in handling Terraform configurations.
- Automates Terraform runs to consistently apply infrastructure changes after code updates.
- Secures sensitive data with strict access controls to ensure only authorized changes.
- Offers a private registry for sharing and managing Terraform modules, boosting reusability and uniformity.

Setting up Variables and Versions in Terraform Cloud

Duration: 25 Min.

Project agenda: To set up variables and versions in Terraform cloud to customize infrastructure configurations, track changes, and maintain consistent infrastructure states

Description: As a DevOps engineer at tech company, you will enhance infrastructure management using Terraform Cloud. This project involves setting up variables to customize configurations and implementing version control to track changes and maintain consistent states. By leveraging Terraform Cloud's capabilities, you will ensure flexible, reusable deployments and efficient management, reducing manual interventions and enhancing scalability.

Setting up Variables and Versions in Terraform Cloud

Duration: 25 Min.

Perform the following:

1. Log in to the Terraform cloud
2. Create an organization
3. Create a new workspace
4. Add environment variables
5. Add Terraform variables
6. Set up the version

Expected deliverables: A fully configured set of variables in Terraform Cloud for customizable infrastructure and a version-controlled environment ensuring consistent and trackable changes to infrastructure configurations.

Thank You