SDD: ACME Corp Cloud 2.0 with HCP Terraform

Created: Aug 04, 2025
Current Version: 1.1.0

Status: In-Review | Approved | Obsolete
Owner: maxwell.castro@ibm.com
Contributors:
Other stakeholders:
Approvers: james.anderton@ibm.com

1. Overview

This document outlines a solution design for ACME Corp's "Cloud 2.0" initiative, which aims to minimize cloud costs and enforce uniform provisioning. The proposed solution is based on the HCP Terraform solution, which will serve as the new standard for infrastructure provisioning across all business units. The solution provides a unified platform to address current challenges, including the lack of comprehensive governance and cross-cloud support in ACME's existing IaC tools.

2. Current Scenario

ACME Corp's current infrastructure provisioning for its customer-facing website relies on a decentralized, multi-faceted approach. This includes a combination of cloud-native Infrastructure as Code (IaC) tools, such as AWS CloudFormation and Azure Resource Manager (ARM) templates, complemented by a suite of custom-developed scripts. While this approach has enabled individual teams to deploy infrastructure, it suffers from several key deficiencies. Specifically, the lack of a unified governance framework and standardized workflows across different cloud providers has resulted in inconsistencies, operational inefficiencies, and a significant management overhead, which directly impedes the strategic objectives of cost reduction and enterprise-wide standardization.

3. Requirement Details

The key requirements for the new provisioning platform, as per the Cloud 2.0 initiative, are:

- **Cost Reduction:** The platform must help minimize cloud costs by enabling granular control over resource provisioning, enforcing budget constraints through policy, and optimizing resource utilization. This includes preventing over-provisioning, ensuring resources are tagged for cost allocation, and identifying idle or underutilized assets. The company is particularly focused on achieving a measurable ROI and reducing overall cloud spend.
- **Standardization:** It must enforce provisioning in a uniform way across all business units and cloud environments. This means establishing consistent naming conventions, resource configurations, and security baselines, reducing configuration drift and ensuring compliance. The goal is to eliminate the ad-hoc provisioning practices that currently exist.
- **Reusable Code:** A dedicated team of experts will be responsible for developing and maintaining a library of reusable infrastructure code components (modules). These modules should encapsulate best practices, security standards, and common configurations, allowing developers to consume pre-approved building blocks rather than writing infrastructure code from scratch.
- **Common Registry:** The platform must offer a centralized, easily accessible common registry for developers to source these reusable components. This registry should provide versioning, documentation, and discovery capabilities to promote module adoption and consistency.
- **Multi-cloud Support:** The solution should provide a unified approach to managing infrastructure across multiple cloud providers (e.g., AWS, Azure, GCP). This is critical to avoid vendor lock-in and to leverage the strengths of different cloud platforms without introducing operational complexity. The company is particularly interested in how this will handle cloud-specific implementation details.
- **Governance:** The platform must include comprehensive governance capabilities to ensure compliance with internal policies and security best practices. This includes policy enforcement at various stages of the provisioning lifecycle (e.g., pre-plan, post-plan, pre-apply) to prevent non-compliant deployments.
- **Workflow Automation:** The solution needs to automate infrastructure

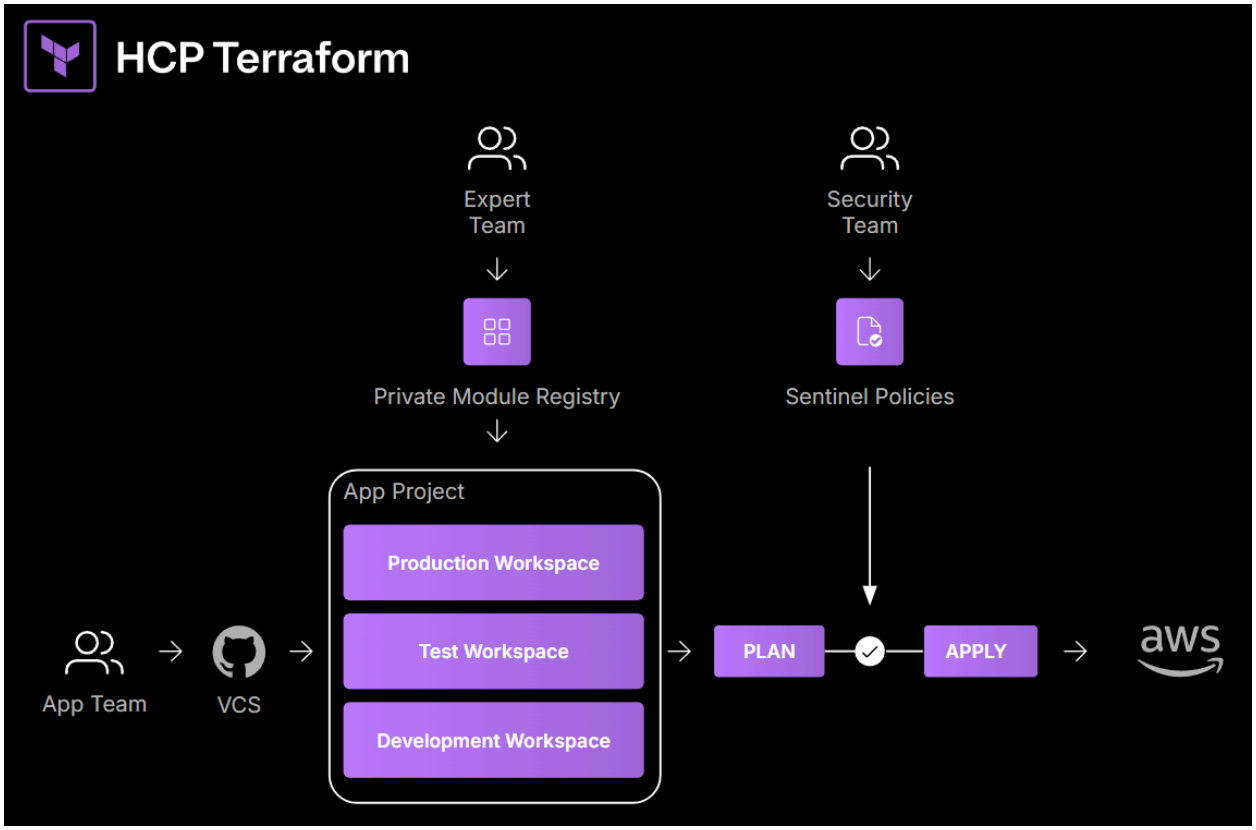
provisioning workflows and integrate seamlessly with existing CI/CD pipelines. This involves automating plan and apply operations, enabling GitOps-style workflows, and reducing manual intervention to accelerate deployment times and minimize human error.
- **Operational Reliability:** The platform must provide assurance about reliability and operational supportability. This includes features like remote state management for consistency and collaboration, drift detection to identify unauthorized changes, and robust monitoring and observability options for tracking infrastructure changes and troubleshooting issues. The company is particularly concerned about single points of failure and the ability to track changes.

4. Assumptions and Prerequisites

- A dedicated team will be assembled to develop and maintain the private module registry.
- Executive sponsorship is in place to support the adoption of a new platform.
- The team is willing to invest in the necessary training to become proficient with Terraform and HCP Terraform.
- ACME Corp has a version control system (e.g., Git) in place that can be integrated with HCP Terraform.

5. High-Level Design

The high-level design involves establishing a centralized HCP Terraform organization with a development workspace. This setup uses a GitOps-style workflow with VCS integration. A private module registry will be created to host reusable infrastructure components, such as a networking module and a compute module, developed by a core team of experts. Sentinel policies will be configured to enforce governance and security rules across all workspaces. The workflow will be driven by a VCS, where code changes trigger automated Terraform runs in the corresponding workspaces, thereby providing a standardized and auditable deployment pipeline.



6. Low-Level Design

The project structure for the Terraform configuration and modules is as follows:

```
acme-demo/
├── README.md
├── backend.tf
├── main.tf
├── outputs.tf
└── variables.tf
```

- **Module Development:**
    - The core team (Expert Team) will create and publish Terraform modules for common infrastructure components.
    - There are two published modules:
        - *The networking module*: creates an AWS VPC and a public subnet. It takes inputs for the VPC CIDR block, subnet CIDR block, availability zone, and a name prefix. The outputs.tf file

for this module exposes the vpc_id and subnet_id.
        - *The compute module*: provisions an AWS EC2 instance and its variables include ami_id, instance_type, and a list of allowed_ssh_cidrs to pass the Sentinel policy check.
- **Governance:** A Sentinel policy named *ec2-instance_type-check* is created in HCP Terraform and it is designed to enforce allowed EC2 instance types. The policy's enforcement level is set to soft mandatory.
- **CI/CD Integration:** The HCP Terraform will be integrated with ACME's existing Version Control Provider. Runs will be triggered based on changes to configuration in repositories segregated by branches, allowing automated plan, based on pull request merges, and manually approved apply.

7. Impact Analysis

The adoption of Terraform Cloud will fundamentally transform ACME Corp's infrastructure provisioning practices. The current heterogeneous approach, which relies on a mix of cloud-native IaC tools and custom scripts, will be replaced by a standardized, unified, and governed platform. This transition is expected to yield significant positive impacts across the organization:

- **Enhanced Standardization and Consistency**: By centralizing infrastructure definitions in a private module registry and enforcing a VCS-driven workflow, ACME will establish a single source of truth for all infrastructure. This will eliminate configuration drift, ensure all deployments adhere to a uniform standard, and dramatically reduce the likelihood of manual errors.
- **Accelerated Time to Market**: The use of pre-approved, reusable modules will allow development teams to provision complex infrastructure with minimal effort and in a fraction of the time it would take to write the code from scratch. This increased velocity will enable business units to innovate and deploy new services more rapidly.
- **Strengthened Governance and Security Posture**: The implementation of Sentinel policy as code will move governance left in the development lifecycle. Policies will be evaluated on every proposed change (terraform plan), automatically preventing non-compliant or insecure deployments before they are even provisioned. This proactive approach will significantly bolster the company's security posture and ensure continuous compliance.
- **Improved Collaboration and Operational Efficiency:** A centralized platform with remote state management will enable multiple teams to work on shared

infrastructure without the risk of state conflicts. The auditable history of every Terraform run will provide a clear trail of all infrastructure changes, simplifying troubleshooting and enhancing operational efficiency for the SRE and DevOps teams.

- **Measurable Cost Savings:** The combination of policy enforcement (e.g., preventing oversized instances), standardization, and automation will lead to a demonstrable reduction in cloud spend and operational overhead. This will directly contribute to the CIO's goal of reducing Total Cost of Ownership (TCO).

8. Out-of-Scope

- Migration of all existing infrastructure to Terraform is not part of the initial project scope and will be handled in a phased approach.
- Detailed development of all reusable modules for every possible use case is not included; the initial focus will be on the most common components.

9. Risks and Mitigation

- **Risk:** Adoption challenges from technical teams due to the introduction of a new platform and workflow.
  - **Mitigation:** A pilot project will be conducted to build internal buy-in by demonstrating the clear superiority and benefits of the new platform over existing tools.
- **Risk:** The learning curve for the team to become proficient with Terraform could slow down the project.
  - **Mitigation:** The implementation will start with a core set of easy-to-use modules, allowing developers to get started quickly without deep expertise. The documentation also suggests that the learning curve is reduced by the use of the private module registry.
- **Risk:** Unforeseen requirements or project scope creep from various business units could cause project delays.
  - **Mitigation:** The project will be scoped to a specific pilot team initially, with a phased onboarding of other teams to gather feedback and refine requirements incrementally.