LTIMindtree

Whitepaper

Accelerate Digital Transformation with Self-Provisioning Catalog-based Infrastructure as a Code Strategy

Contents

Introduction

89%

of companies already plan to adopt a digital-first business strategy

Success in transformation demands a focus on three critical technological pillars: the cloud, IT infrastructure, and security and compliance. To accomplish agility and other digital transformation goals, it is necessary to streamline management and configuration workflows across all three pillars. Streamlining workflows increases responsiveness and frees up the resources and support needed for innovation.

Organizations will face technical obstacles and traps as they embark on their digital transformation path via cloud. The correct tools and someone to help you along the way are critical components of a successful digital transformation through the cloud.

Moving to the cloud means doing things in a new way. The basics of data centers are changing; instead of fixed structures, we now have flexible ones. Security and networking are based on who you are, and they are not just a number. We're also using automatic self-service systems instead of old-fashioned ticketing.

This switch helps things get done faster and makes it quicker to bring products to market. However, it also creates many old tools and processes outdated. Companies need to adapt to flexible and efficient workflows at every level, to make the most of cloud computing. This is especially true for the infrastructure layer, where we're changing how we set up and manage things. We're now using a system that codes infrastructure, keeps a good record, and manages the whole life of that infrastructure. These changes lay the foundation for crucial shared cloud services. Teams in organizations use these services to cut costs, reduce risks, and speed up their work.

LTIMindtree

Executive summary

This paper explores making the first part of how a cloud works better, specifically looking at the infrastructure layer. The main idea is to get the most out of it by using something called "Infrastructure as Code" in the framework of how the cloud works. It's like having a plan that grows as you get better at it.

Adopting Infrastructure as Code (IaC) is one of the best approaches to automating infrastructure. IaC is the process of controlling and providing infrastructure through readable definition files rather than real hardware configurations or interactive configuration tools. It applies to bare metal servers, virtualized systems, and the cloud.

Terraform for IaC and ServiceNow for IT service and operations management are widely used tools by most organizations.

Terraform automates cloud/on-prem infrastructure provisioning and security with declarative infrastructure and policy as code. Infrastructure and policies are executed within a consistent workflow across the entire infrastructure after it is codified, shared, and versioned.

ServiceNow handles incidents, service requests, problems, changes, and catalog regular IT service requests.

For catalog-based IaC setup and provisioning, we have considered Terraform and ServiceNow tools as the key components in our journey towards catalog-based Infrastructure as Code.

Common challenges at the infrastructure layer

Addressing challenges at the infrastructure layer becomes paramount as organizations transition from on-premises setups to the cloud. New demands emerge, and operators must navigate these complexities:

Scale

Teams aim to swiftly adjust their infrastructure capacity, accommodating fluctuations without errors amid potentially extensive configuration changes.

Variety

Unified provisioning workflows are sought across various platforms, streamlining operations and ensuring a cohesive approach to diverse environments.

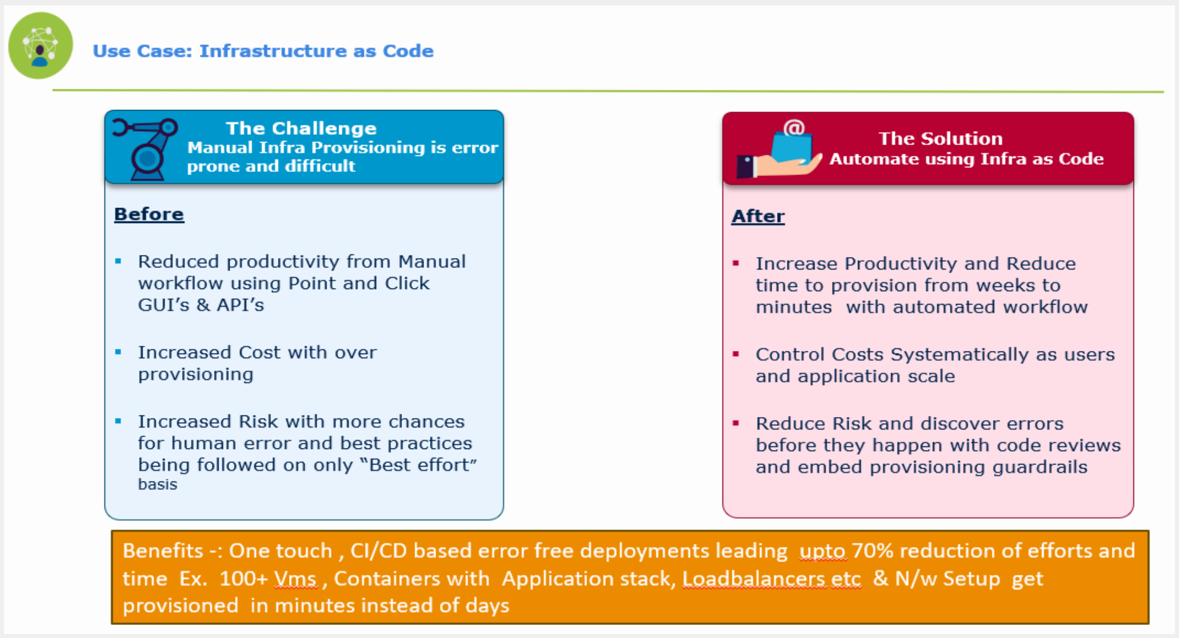
Dependencies

Within the provisioning workflow, teams seek the inclusion and automation of existing services and dependencies, enhancing efficiency and reducing manual intervention.

Use case for self-provisioning catalog-based Infrastructure as Code

At the core of a cloud operating model lies infrastructure provisioning, driven by the principles of Infrastructure as Code (IaC). By translating infrastructure into code, teams can clearly define the desired end state of a deployment. This not only ensures consistency across deployments but also enables tracking and auditing of changes made to the code. The transformative shift to IaC establishes a foundation for efficient and error-resistant cloud operations, aligning with the evolving needs of dynamic and diverse cloud environments.

Catalog-based Infrastructure as Code provides organizations with a powerful solution to manage and scale their infrastructure efficiently. Maintaining a catalog of reusable components and automating provisioning processes can help the company meet customer demands, reduce operational overhead, and ensure the consistency and security of its IT Systems.



Use Case: Infrastructure as Code

The Challenge
Manual Infra Provisioning is error prone and difficult

Before

- Reduced productivity from Manual workflow using Point and Click GUI's & API's
- Increased Cost with over provisioning
- Increased Risk with more chances for human error and best practices being followed on only "Best effort" basis

The Solution
Automate using Infra as Code

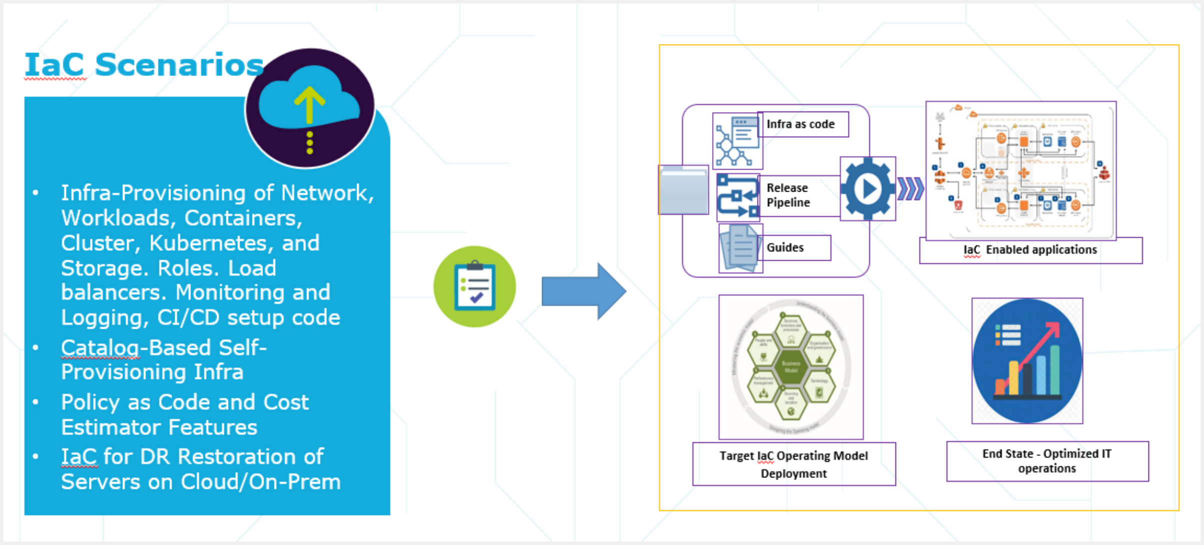
After

- Increase Productivity and Reduce time to provision from weeks to minutes with automated workflow
- Control Costs Systematically as users and application scale
- Reduce Risk and discover errors before they happen with code reviews and embed provisioning guardrails

Benefits -: One touch , CI/CD based error free deployments leading upto 70% reduction of efforts and time Ex. 100+ Vms , Containers with Application stack, Loadbalancers etc & N/w Setup get provisioned in minutes instead of days

Figure 1 denotes the problem statement faced by many organizations, along with a solution.

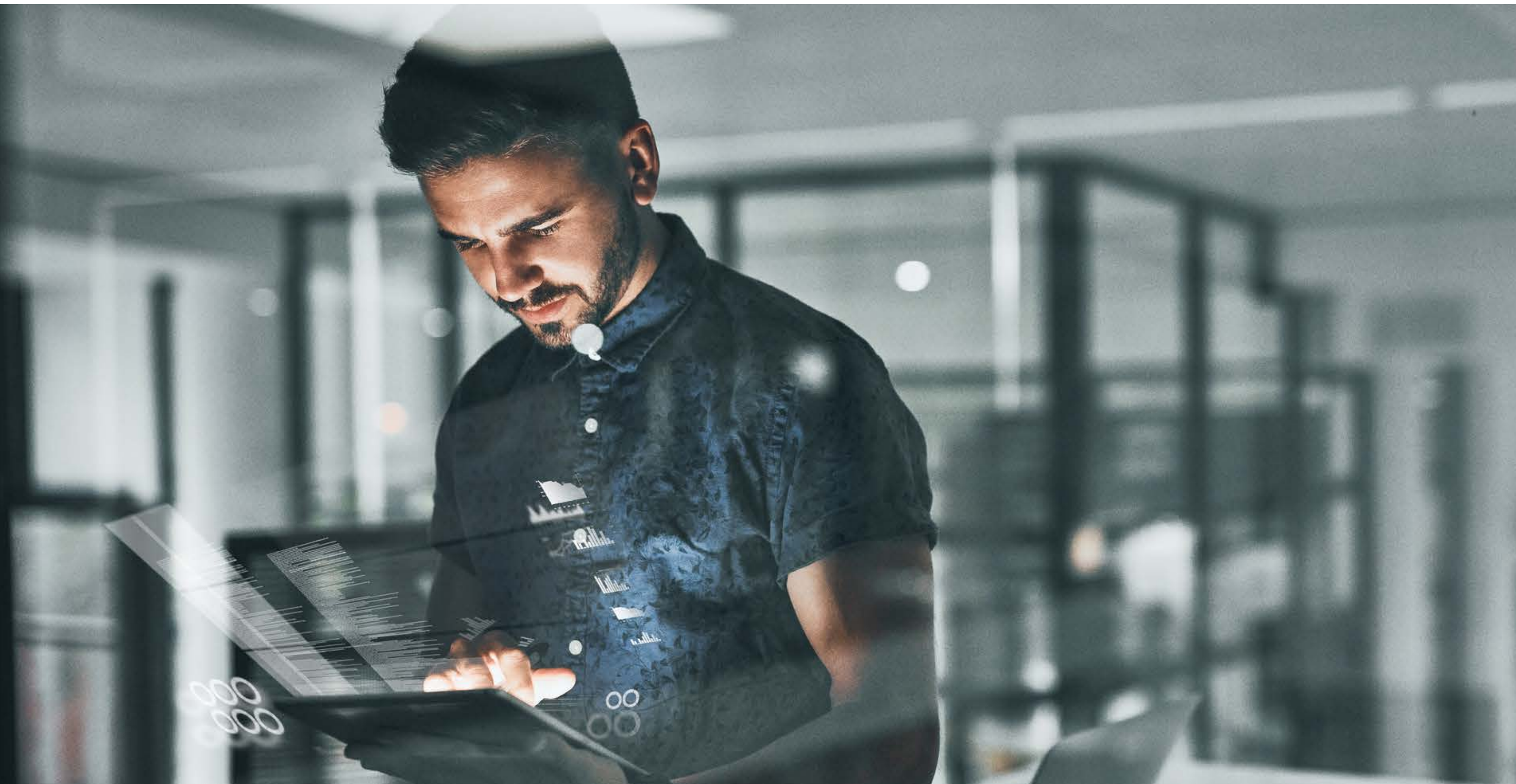
Scenarios for IaC and future mode of operations


Infra as code
Release Pipeline
Guides
IaC Enabled applications
Target IaC Operating Model Deployment
End State - Optimized IT operations

IaC Scenarios

- Infra-Provisioning of Network, Workloads, Containers, Cluster, Kubernetes, and Storage. Roles. Load balancers. Monitoring and Logging, CI/CD setup code
- Catalog-Based Self-Provisioning Infra
- Policy as Code and Cost Estimator Features
- IaC for DR Restoration of Servers on Cloud/On-Prem

Figure 2 denotes key scenarios for IaC implementation along with future modes of operations.



Key benefits of using IaC

IaC uses a code to define, deploy, and maintain infrastructure resources. This approach offers many benefits, such as increased efficiency, reliability, and agility in managing IT infrastructure. Some key benefits of using Infrastructure as Code are:

Faster time to production/market

IaC automation dramatically speeds the process of provisioning infrastructure for development, testing, and production (and for scaling or taking down production infrastructure as needed).

Improved consistency—less ‘configuration drift’

Configuration drift occurs when ad-hoc configuration changes and updates result in mismatched development, test, and deployment environments. This can result in issues at deployment, security vulnerabilities, and risks when developing applications. IaC prevents drift by provisioning the same environment every time.

Faster, more efficient development

By simplifying provisioning and ensuring infrastructure consistency, IaC can confidently accelerate every phase of the software delivery lifecycle. Developers can quickly provision sandboxes and continuous integration/continuous deployment (CI/CD) environments.

Protection against churn

Provisioning is delegated to a few skilled engineers or SMEs to maximize efficiency in organizations without IaC. If one of these specialists leaves the organization, others are sometimes left to reconstruct the process. IaC ensures that provisioning intelligence always remains with the organization.

Lower costs and improved ROI

In addition to dramatically reducing the time, effort, and specialized skill required to provision and scale infrastructure, IaC lets organizations take maximum advantage of cloud computing’s consumption-based cost structure. It also enables developers to spend less time on plumbing and more time developing innovative, mission-critical software solutions.

Portability

IaC makes it easier to move infrastructure configurations across different cloud providers or environments. This portability is beneficial for organizations employing a multi-cloud or hybrid cloud strategy.

Collaboration

IaC promotes collaboration between development and operations teams. Both teams can work on the same codebase, improving communication and ensuring that infrastructure requirements align with application development goals.

In summary, Infrastructure as Code enhances automation, consistency, collaboration, and efficiency in managing and deploying infrastructure, making it a crucial practice in modern DevOps and cloud-based environments.

RoadMap for implementing Infrastructure as Code layer with maturity model

The maturity model stages for the infrastructure as code layer involve the strategic development of three key components as organizations advance in their infrastructure implementation:

Best-in-class provisioning workflow

- Adopting: The initial phase focused on integrating new provisioning workflows.
- Standardizing: Establishing uniform practices for increased efficiency.
- Scaling: Expanding the workflow capabilities for broader applications.

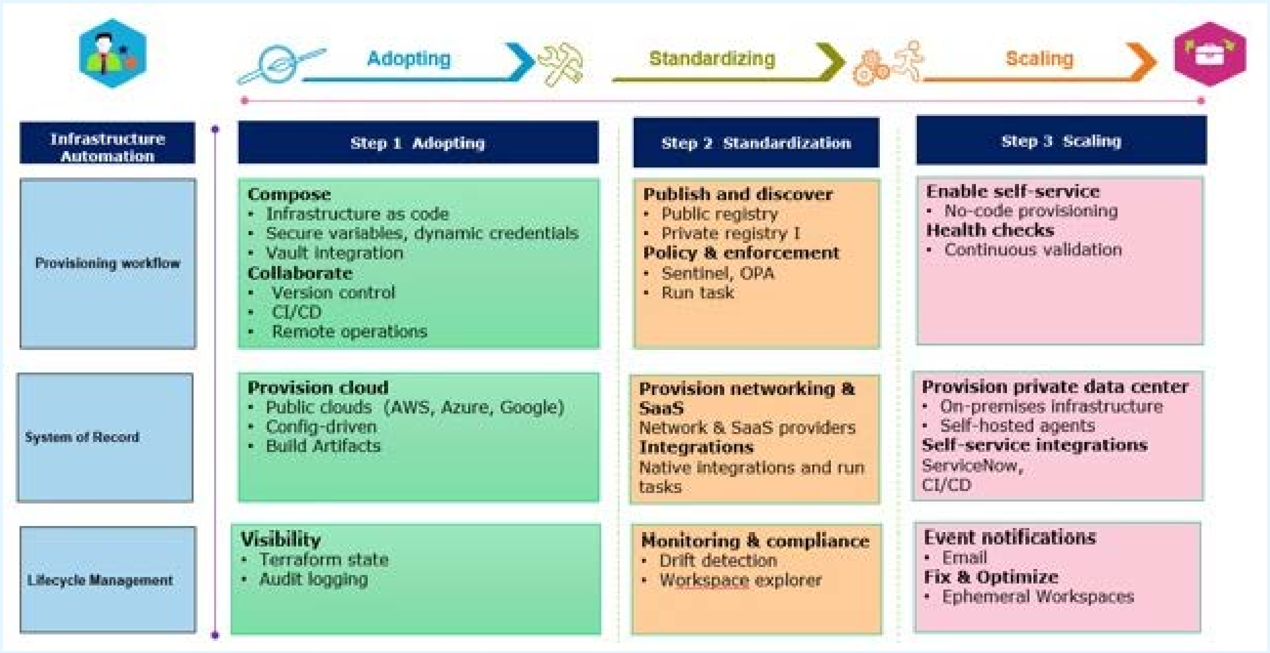
System of record for enhanced visibility (Day 2 operations)

- Adopting: Introduction of a foundational system for recording operations.
- Standardizing: Implementing consistent practices to enhance visibility.
- Scaling: Elevating the system to meet the evolving needs of operations.

Resource management system for infrastructure lifecycle

- Adopting: Initiating a system to manage the lifecycles of infrastructure components.
- Standardizing: Implementing standardized procedures for efficient management.
- Scaling: Expanding the system's capabilities to handle diverse infrastructure lifecycles.

In each of these critical areas, organizations are advised to navigate through the maturity model stages—adopting, standardizing, and scaling—to formulate a robust implementation roadmap. This strategic approach ensures a systematic and well-orchestrated progression toward an optimized and scalable Infrastructure as Code layer.



Infrastructure Automation	Step 1 Adopting	Step 2 Standardization	Step 3 Scaling
Provisioning workflow	**Compose** • Infrastructure as code • Secure variables, dynamic credentials • Vault integration **Collaborate** • Version control • CI/CD • Remote operations	**Publish and discover** • Public registry • Private registry I **Policy & enforcement** • Sentinel, OPA • Run task	**Enable self-service** • No-code provisioning **Health checks** • Continuous validation
System of Record	**Provision cloud** • Public clouds (AWS, Azure, Google) • Config-driven • Build Artifacts	**Provision networking & SaaS** Network & SaaS providers **Integrations** Native integrations and run tasks	**Provision private data center** • On-premises infrastructure • Self-hosted agents **Self-service integrations** ServiceNow, CI/CD
Lifecycle Management	**Visibility** • Terraform state • Audit logging	**Monitoring & compliance** • Drift detection • Workspace explorer	**Event notifications** • Email **Fix & Optimize** • Ephemeral Workspaces

Figure 3 denotes the implementation road map for IaC with maturity model stages.

Building blocks for Infrastructure as Code layer

8.1. Infrastructure as Code tool enablement

Terraform is a preferred tool for automating both cloud and on-premises infrastructure. It uses infrastructure as code to handle provisioning and ensure compliance in the cloud operating model.

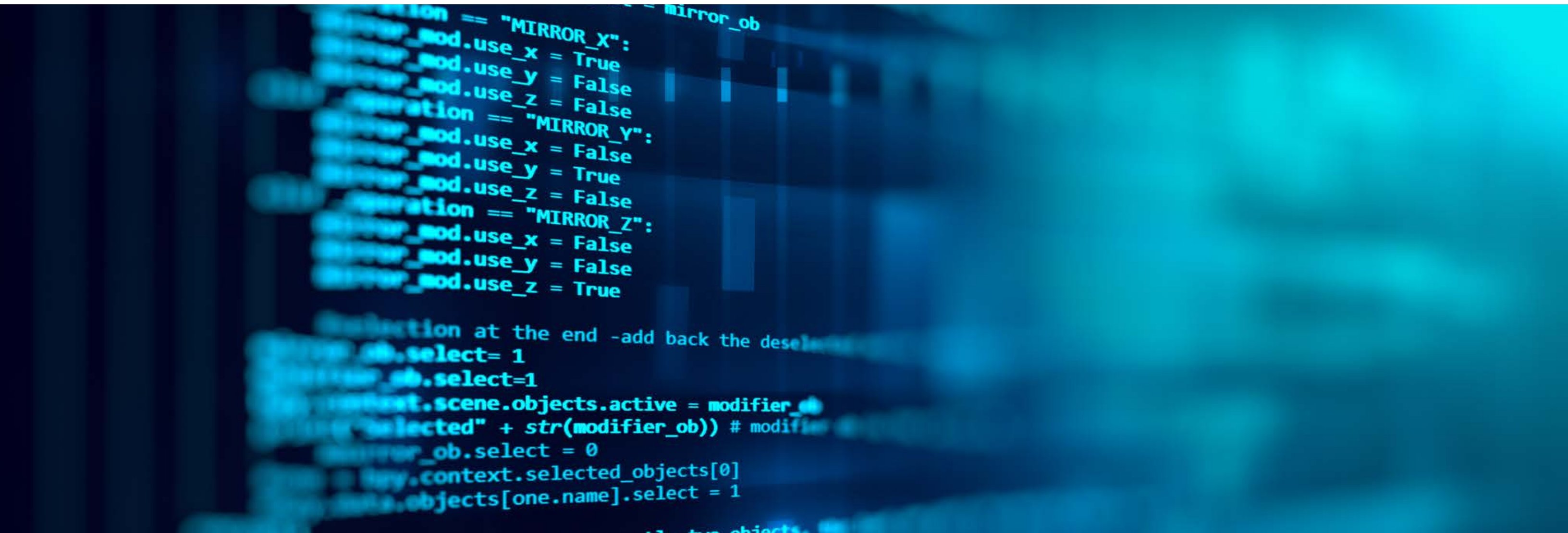
Terraform automates the provisioning of cloud infrastructure and services using an infrastructure-as-code approach. Users specify their desired infrastructure and services in a configuration file using version control. Terraform then translates these configurations into API calls, automating resource provisioning to minimize errors and build failures. With open-source providers, it supports quick creation and compatibility with various types of infrastructure. It is the core for automating cloud infrastructure, using infrastructure and policy as code for compliance and management in the cloud operating model.

Organizations transitioning to this model aim for both agility and control, swiftly delivering products to markets and internal users. They aspire to a state where workflows are consistent across technologies, infrastructure, and service provisioning are automated, and security aligns with the speed and scope of automation. To reach this state, organizations must:

- Enable real-time control and proactive policy enforcement
- Remove manual processes and bottlenecks
- Centralize management and control across technologies

8.2. IaC architecture and catalog-based demand execution

A reference architecture for catalog-based IaC is depicted below. Its primary constituents are ServiceNow, VCS repository CI/CD pipelines, and Terraform to create infra and application resources for on-prem and multi-cloud environments.



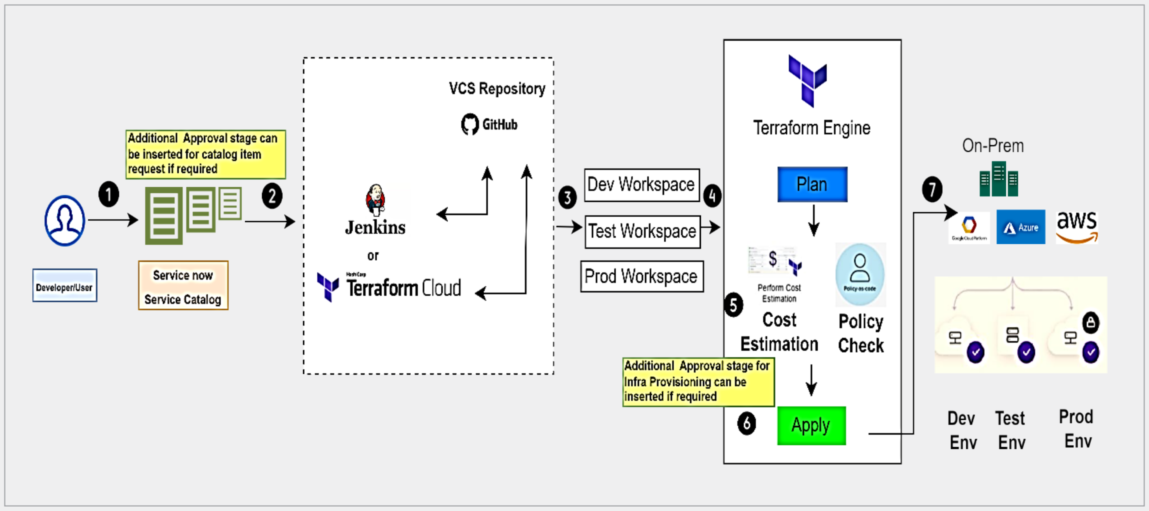


Figure 4 denotes reference architecture for catalog-based Infrastructure as Code using Terraform.

1. The user goes to the catalog and picks an item.
2. A request is triggered from ServiceNow to Terraform.
3. Workspaces are created in Terraform.

1. Terraform backend triggers Plan.
2. Cost and Policy checks are done
3. Based on SUCCESS, Apply is initiated.
4. Required Infrastructure is provisioned in On-Prem or Cloud as per request

8.3. Provisioning

The Operating Model often begins by helping operations teams move from focusing solely on setting up specific servers tied to similar infrastructure to using workflows that support a shift-left IT approach using Terraform for Infrastructure as Code. This allows for on-demand capacity from different Cloud and service providers. To handle key aspects of provisioning, like the quantity and distribution of services, the temporary and unchangeable nature of resources, and deploying to various environments, organizations are shifting towards an automation-centered operating model for cloud infrastructure.

8.4. Compliance

Traditional methods of preventing infractions rely on IT as the gatekeeper to infrastructure. Policies are typically understood informally within the IT team and are not explicitly coded. If policies are enforced, it's done manually by the gatekeeping organization, hindering the speed and self-service advantages offered by cloud infrastructure. While some teams automate policy enforcement, they often use post-provisioning scans, introducing risks during the period before the infrastructure is scanned for compliance. Lastly, some organizations opt for a least common denominator approach to enforcement, assuming all infrastructure is susceptible to specific risks, neglecting unique vulnerabilities in different parts of the organizational infrastructure.

Terraform manages cloud compliance and enhances automation by enforcing policies during the provisioning process. This proactive policy enforcement reduces risks, controls costs, and boosts productivity through automation. This helps minimize risks as organizations scale in the cloud.

Security and compliances aspects handled by Infrastructure as Code are as below:

Enhancing security measures: Creating and enforcing policies to prevent breaches is crucial. Examples of security concerns include:

- Limiting vulnerable app versions
- Restricting resources with public IP addresses
- Prohibiting security groups with egress 0.0.0.0
- Allowing the use of only approved modules

Ensuring regulatory compliance: Establishing and enforcing policies to prevent regulatory noncompliance is vital. Examples of regulatory concerns encompass:

- GDPR
- FedRamp regulations
- HIPAA regulations
- PCI standards

8.4.1. Sentinel policy as code

Using Sentinel Policy as Code, various teams (security, compliance, audit, finance, operations) define policies within Terraform. These policies are checked against each Terraform plan before execution, allowing for preventive and proactive policy implementation. This ensures adherence to best practices and compliance with regulations. Sentinel Policies offer different enforcement levels, ranging from Advisory (warning without prevention) to Soft Mandatory (requires override to break) and Hard Mandatory (prevents provisioning if a policy is violated).

8.4.2. Automated policy enforcement

Terraform ensures that Sentinel policies are applied to workspaces before provisioning so that once an organization sets a rule in Sentinel, no infrastructure can be created that violates it—enforcement happens automatically.

8.4.3. Cost management via policy enforcement

Sentinel enables the creation of cost-focused policies that are automatically applied in the Terraform workflow. Administrators can then approve major changes or block specific workspaces from surpassing set limits. For instance, policies can prevent unnecessary provisioning of large, costly machines or enforce overall budget constraints for team deployments without specifying machine types. Moreover, policies can dynamically allow a certain level of monthly change but cap it at a specified monetary limit.

8.4.4. Audit logging

Terraform Enterprise provides detailed audit logs for organizations seeking insights into the resources managed by Terraform. These logs record information whenever any resource is changed, allowing teams to track who made changes and what those changes were. Many organizations use audit logging to meet and maintain regulatory compliance.

8.5. Workflow integrations with Version Control Systems (VCS)

Many users incorporate Terraform's cloud management capabilities into their existing workflows or toolchains. Terraform facilitates this by integrating with major Version Control Systems (VCS), Continuous Integration/Continuous Deployment (CI/CD) tools, and service management tools. Additionally, it supports a comprehensive REST API. These integrations enable organizations to maintain operational consistency without disruptingproductivity.

Terraform users describe infrastructure using a straightforward, human-readable language called HCL (HashiCorp Configuration Language). Users can create their unique HCL configuration files or use existing templates from the public module registry.

Typically, these configuration files are stored in a Version Control System (VCS) repository linked to a Terraform workspace. This connection allows users to apply software engineering practices to version and refine infrastructure as code. VCS and Terraform Cloud serve as a delivery pipeline. Terraform integrates with Azure DevOps, BitBucket, Github, and Gitlab.

When you make changes in a linked Version Control System (VCS) repository, Terraform will automatically initiate a plan in all workspaces connected to that repository. You can review this plan for safety and accuracy in the Terraform UI before applying it to set up the designated infrastructure.

8.6. Continuous Integration, Continuous Delivery (CI/CD) pipeline

Terraform is compatible with various CI/CD pipelines like Jenkins, Circle, Travis, Gitlab, and Github. Many users use Terraform's programmability to automate much of their provisioning workflow, ensuring compliance through policy as code. Terraform's API-driven runs enable flexible provisioning workflows using an infrastructure-as-code approach, suitable for any organization. In a continuous integration (CI) system, changes in Terraform code are monitored, and Terraform Cloud's REST API is utilized for provisioning. This allows organizations to include various actions in their CI pipeline for infrastructure provisioning while benefiting from Terraform Cloud's features like private modules, state management, policy as code (Sentinel), and more.

8.7. Catalog-based IT Service Management (ITSM)

Terraform offers a seamless integration with ServiceNow. ServiceNow facilitates digital workflow management, promoting efficient collaboration within teams through a user-friendly interaction process. The ServiceNow Service Catalog acts as a storefront where various services can be ordered by different individuals in the organization. This often includes requests for cloud resources, such as developers needing machines for code testing or the finance IT team requiring infrastructure for new accounting software. Organizations using the ServiceNow Service Catalog can submit these requests through ServiceNow, directing them to the appropriate team for Cloud Infrastructure. Terraform automates provisioning through infrastructure as code, ensuring security, compliance, and cost-sensitive policies are applied to all resources during provisioning. This enables teams less focused on coding to easily adopt top-notch provisioning workflows and tools while gaining proficiency in infrastructure as code.

Next steps to get started on the Infrastructure as Code journey

Next Steps

Workshops to get started

- Assessment Workshop to generate esate details and scope for infra as code automation
- Establishing scope and phase plans

Build and Implementation

- Build and Deploy IaC as per defined scope

Figure 5 denotes the potential next steps to get started on the Infrastructure as Code journey.

Conclusion

As organizations transition to the cloud operating model, they initially encounter the challenge of setting up cloud infrastructure. Many believe that while the cloud provides speed, it also introduces new security risks, often leading to the perception that addressing these risks might slow down processes. However, Terraform offers a robust alternative by blending infrastructure as code for provisioning with policy as code for compliance and management. This allows organizations to maintain both agility and control as they build expertise in infrastructure provisioning, compliance, and management.

Using a maturity model allows these teams to create a detailed roadmap, outlining the organization's current position and what steps are needed to reach its goals. This roadmap aids teams across the organization in making quicker and better-informed decisions. Additionally, it facilitates benchmarking progress along the way.

External References

1. Gartner Says 89% of Board Directors Say Digital is Embedded in All Business Growth Strategies, Gartner, Oct 19, 2022: https://www.gartner.com/en/newsroom/press-releases/2022-10-19-gartner-says-89-percent-of-board-directors-say-digital-is-embedded-in-all-business-growth-strategies

About the author

Sandip Bhide

Sandip Bhide has over 22+ years of IT experience. He has led complex, large-scale solutions/ Implementation programs in technology, consulting, presales and delivery. He has extensive experience in cloud transformation projects across domains in BFSI, Life Science focusing on digital transformation, cloud native services, and application modernization. He is part of the TechOffice and specializes in Cloud, IaC, and DevSecOps streams.