Semester **01**

System Provisioning & Configuration Management.

Module # **02**

Provisioning Tools

Module Learning Objectives

Upon successful conclusion of this module, you will be able to:

- Recall the role of provisioning tools in automating the deployment and management of infrastructure.
- Understand the functioning of provisioning tools and their varied use cases.
- Apply the acquired knowledge by focusing on Terraform as a specific provisioning tool.
- Engage in a hands-on experience to actively apply Terraform for provisioning infrastructure.

Module Topics

Introduction to Provisioning Tools

- Ansible, Chef, Puppet and Terraform
- Use Cases and Comparison
- Hands-on with Terraform
- Basic Terraform Commands
- Provisioning Infrastructure on a Cloud Provider

2.1 Introduction to Provisioning Tools

01 Provisioning tools are crucial for automating the deployment and management of infrastructure.

02 Provisioning tools automate the deployment and management of infrastructure.

03 They ensure consistency and efficiency in system administration and configuration management.

2.1.1 Ansible

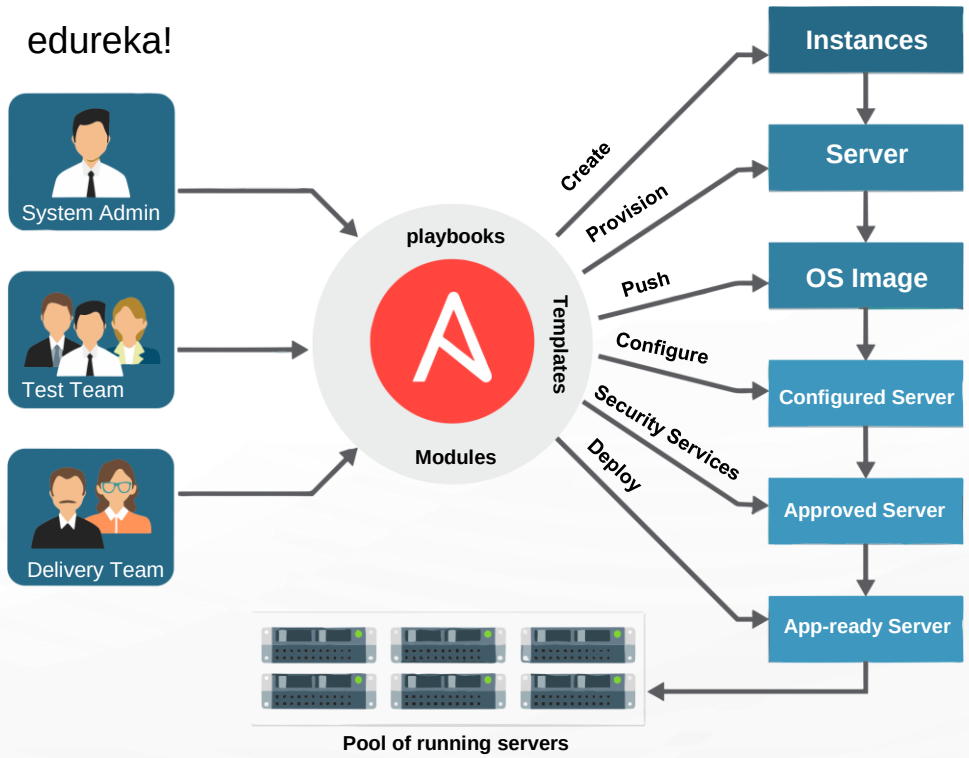
Background

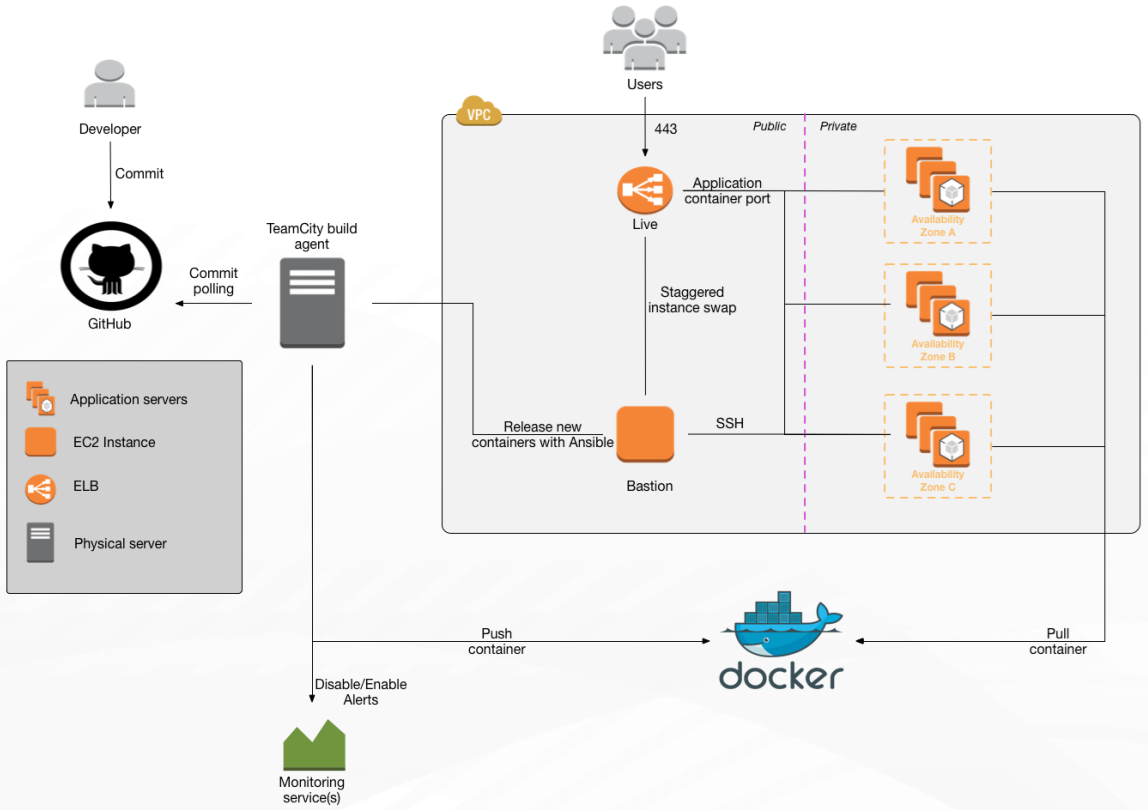
- Ansible was created by **Michael DeHaan**, a former Red Hat employee, and it was **first released in 2012**.

Philosophy and Design

- Ansible is an **open-source automation tool** used for configuration management, application deployment, task automation, and orchestration.

2.1.1 Ansible | Use cases

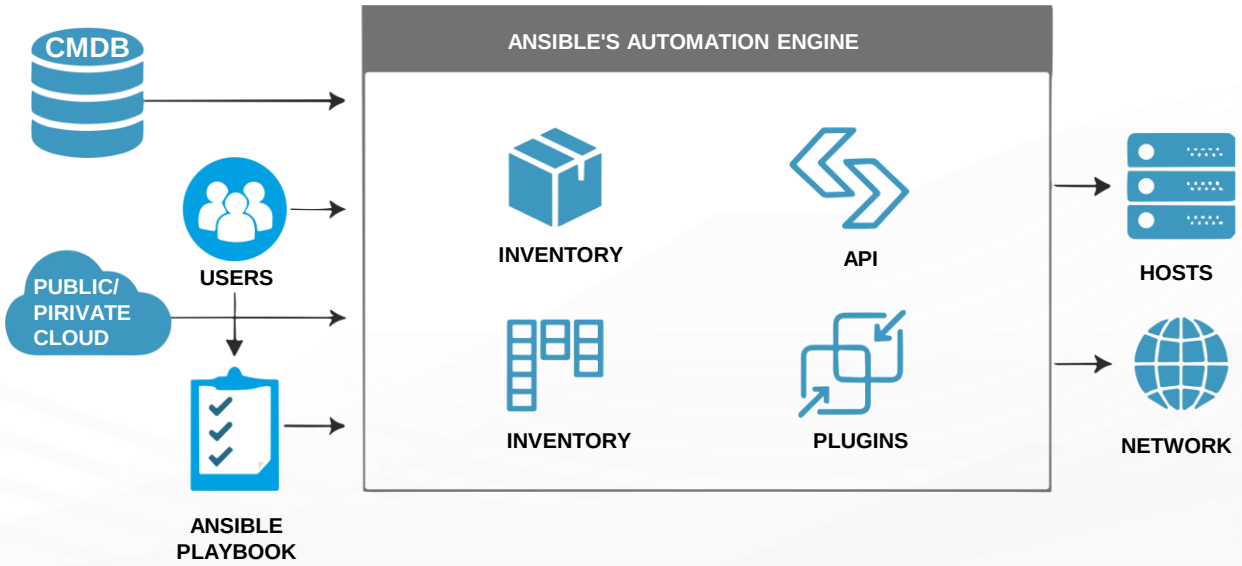


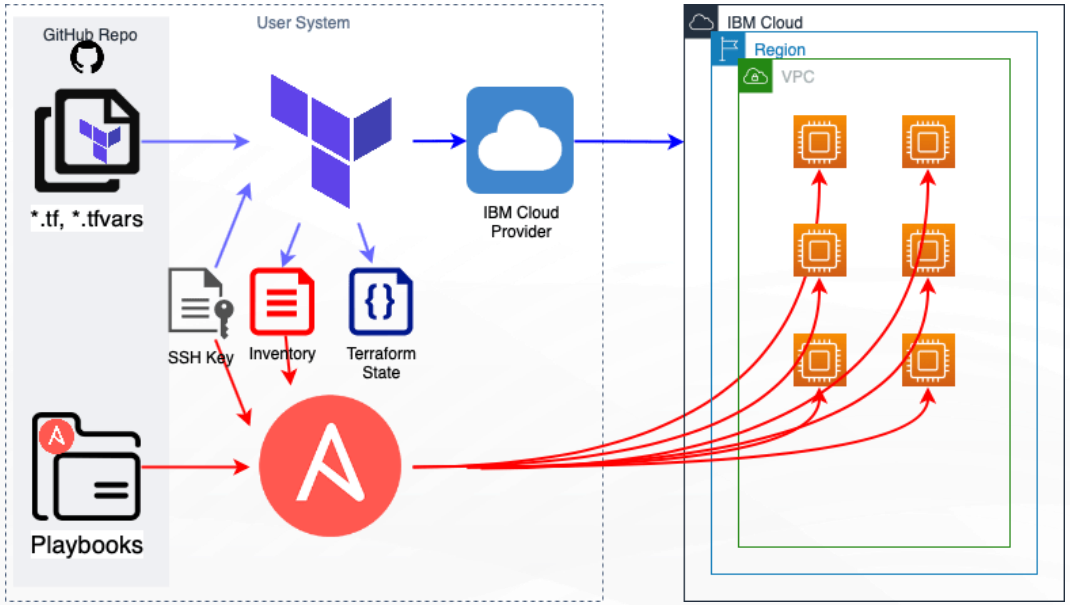


1. **Configuration Management:** Ansible is widely used for configuration management tasks, ensuring that servers and systems are configured consistently.

2. **Application Deployment:** It simplifies the process of deploying and managing applications across different environments.

2.1.1 Ansible | Use cases

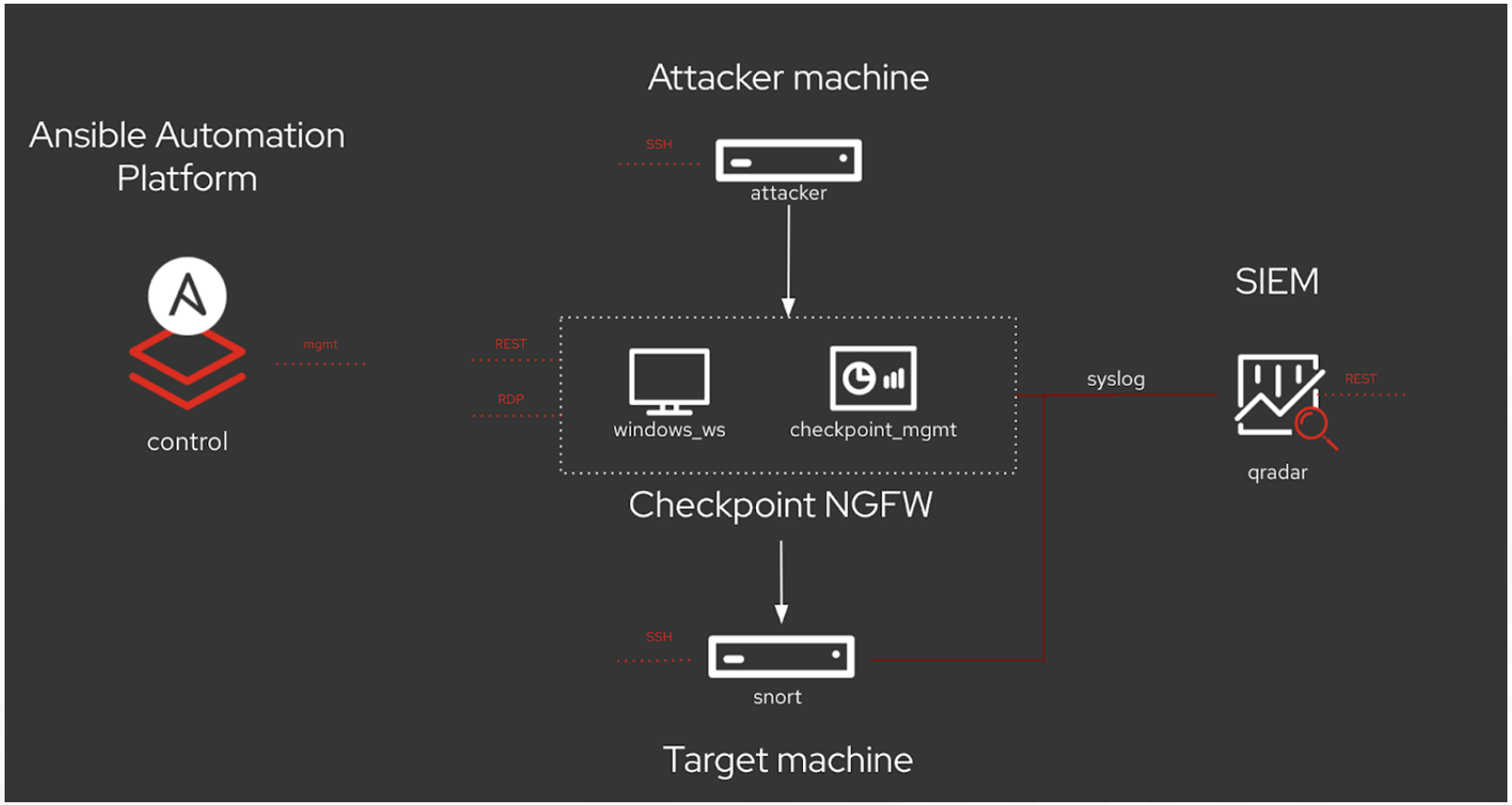




3. **Orchestration:** Ansible facilitates the orchestration of complex workflows and tasks, allowing the automation of multi-step processes.

4. **Infrastructure as Code (IaC):** Ansible supports Infrastructure as Code principles, enabling the definition and versioning of infrastructure using code.

2.1.1 Ansible | Use cases



5. Security Compliance Automation: Ansible ensures systems comply with security policies and standards.

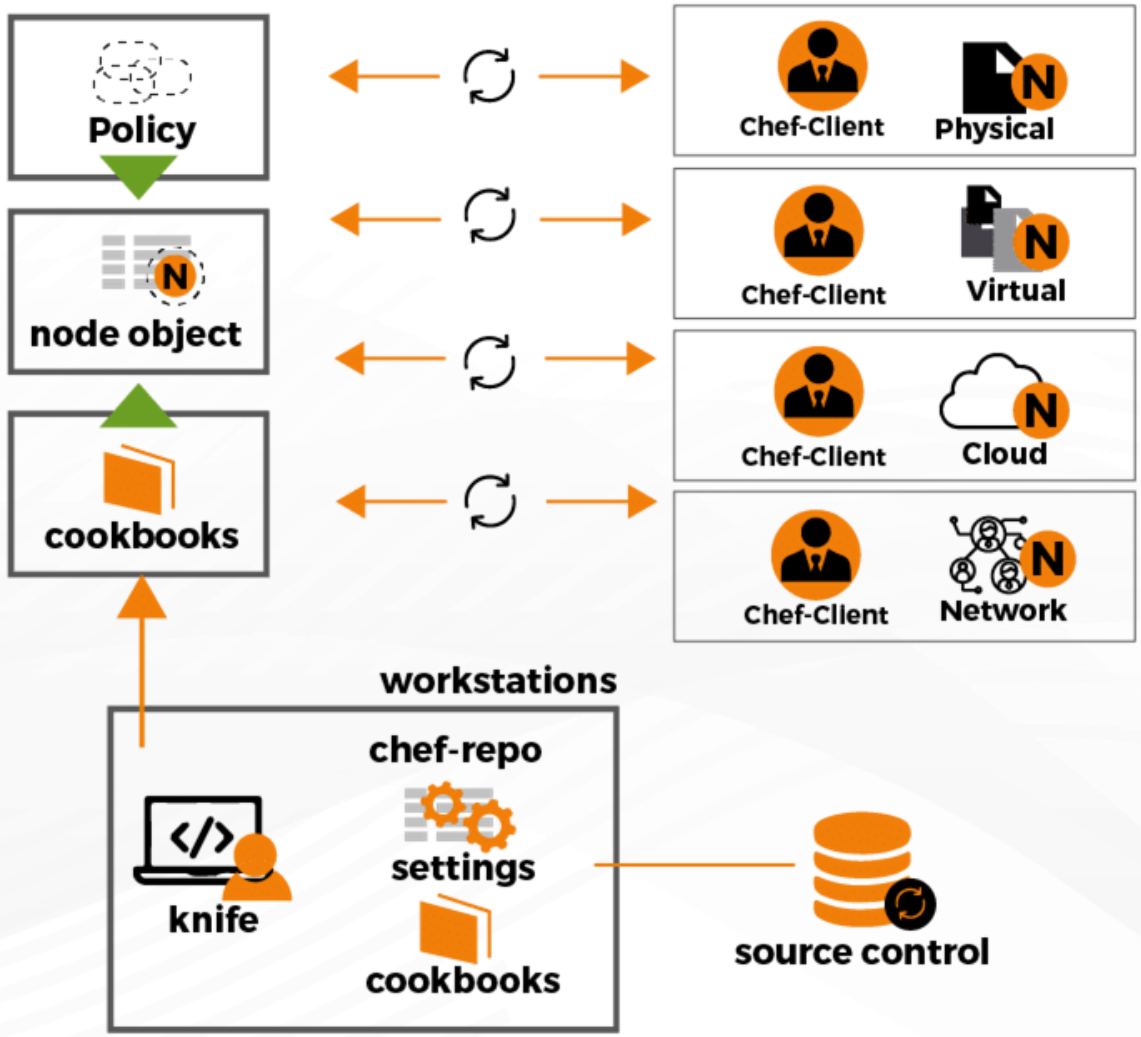
2.1.2 Chef

Background

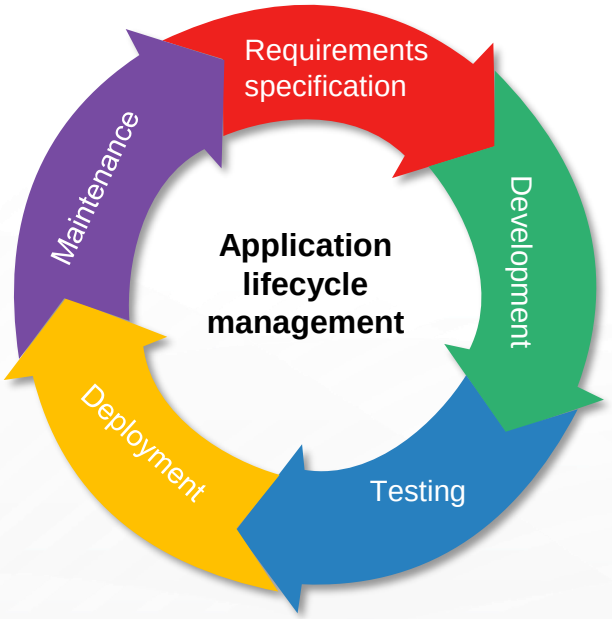
- Chef was created by Adam Jacob, Jesse Robbins, Barry Steinglass, and others and was first released in 2009 by Chef Software, Inc.

Philosophy and Design

- Chef uses a client-server architecture, where the infrastructure to be managed is defined in code and stored on a central Chef server.



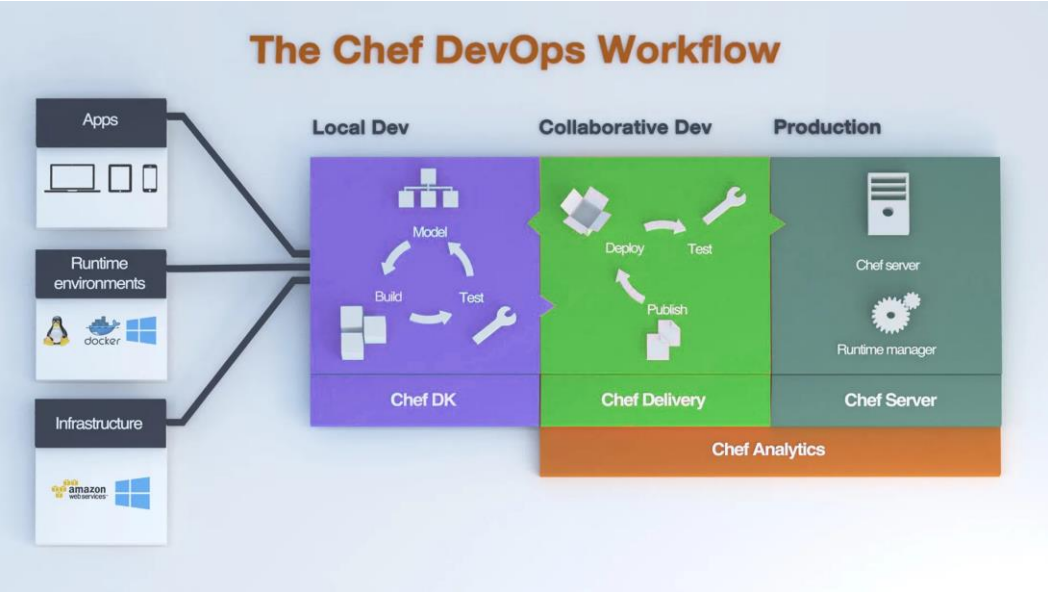
2.1.2 Chef | Use cases

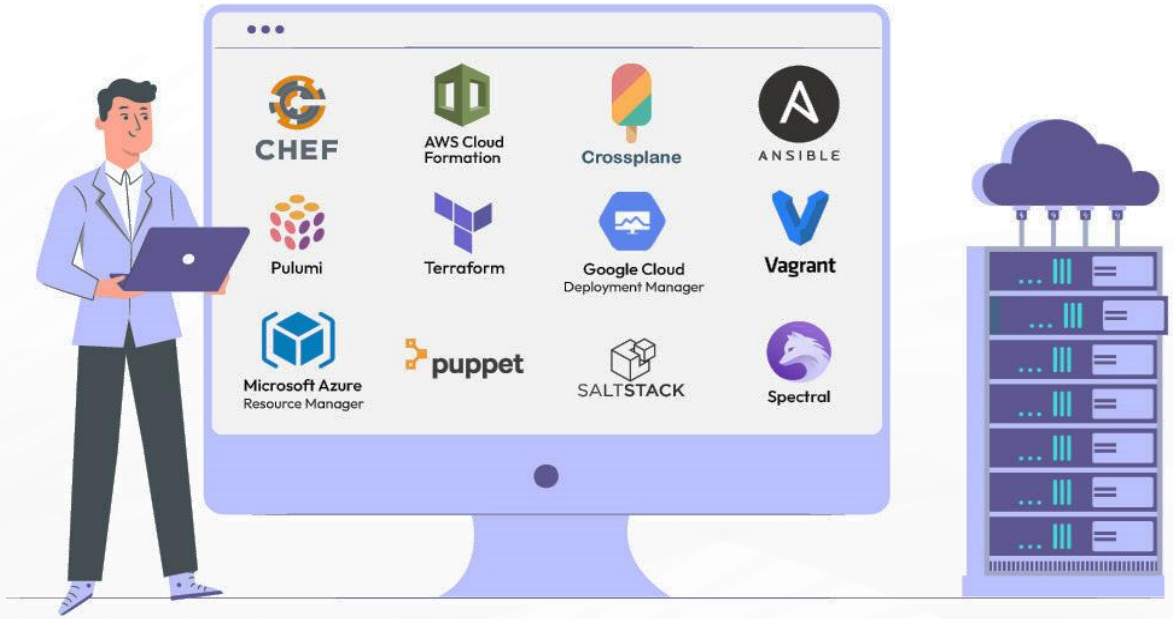


1. **Configuration Management:** Chef is commonly used for defining and maintaining the desired state of server configurations, ensuring consistency across large-scale environments.

2. **Application Lifecycle Management:** Chef facilitates the automated deployment of applications by defining the necessary configurations and dependencies.

2.1.2 Chef | Use cases





3. **Continuous Delivery and DevOps:** Chef supports continuous delivery practices by automating the entire software delivery process, from development to production.

2. **Infrastructure as Code (IaC)**: Chef enables the codification of infrastructure through recipes and cookbooks, allowing teams to version control

2.1.2 Chef | Use cases

5.Compliance Automation: Chef Automate, an add-on product, provides features for compliance automation.

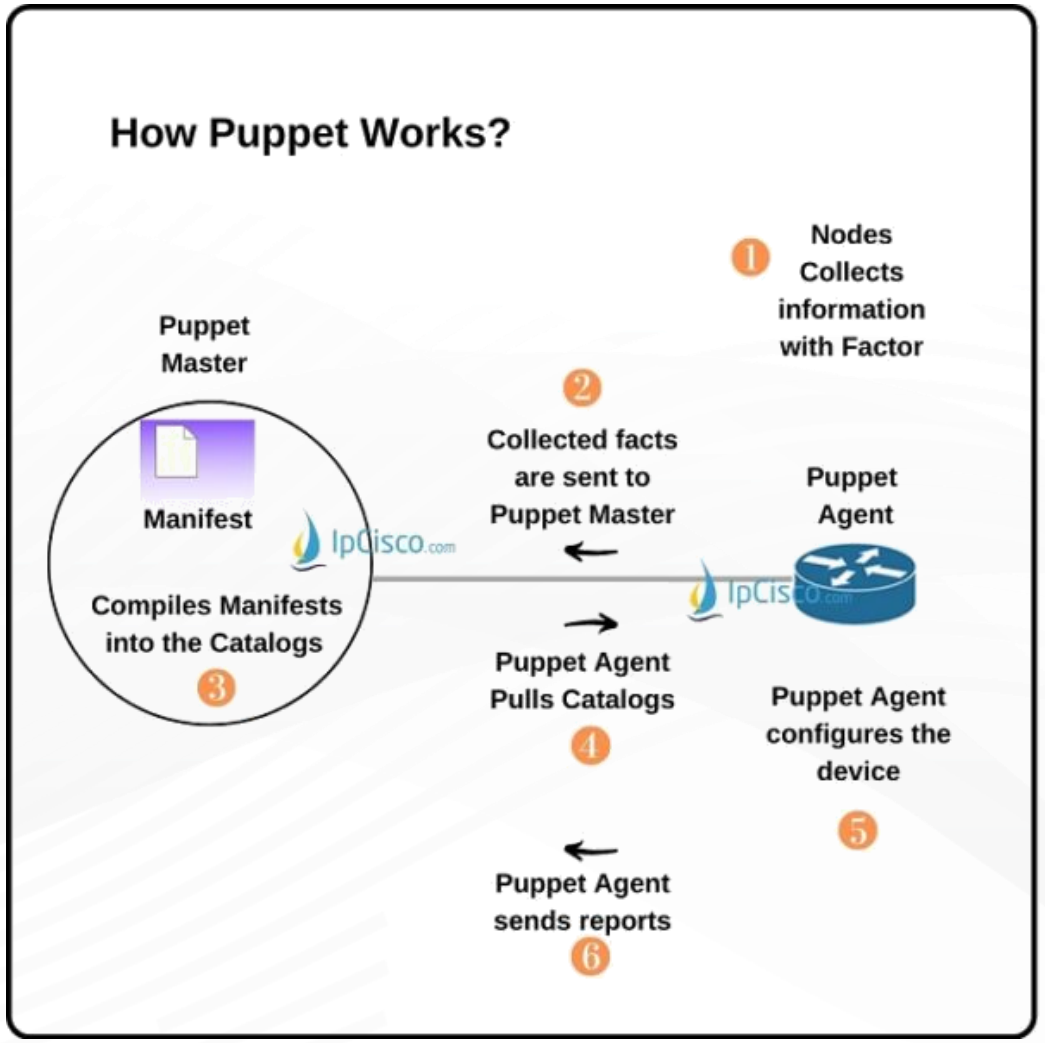
2.1.3 Puppet

Background

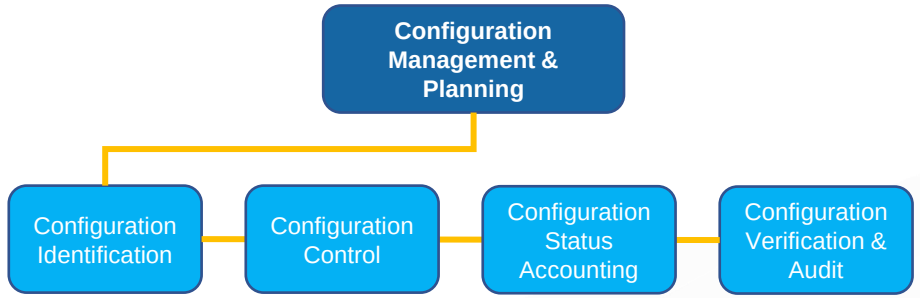
- Puppet was created by Luke Kanies and first released in 2005. Luke Kanies founded Puppet Labs (now known as Puppet, Inc.) to further develop and commercialize the Puppet tool.

Philosophy and Design

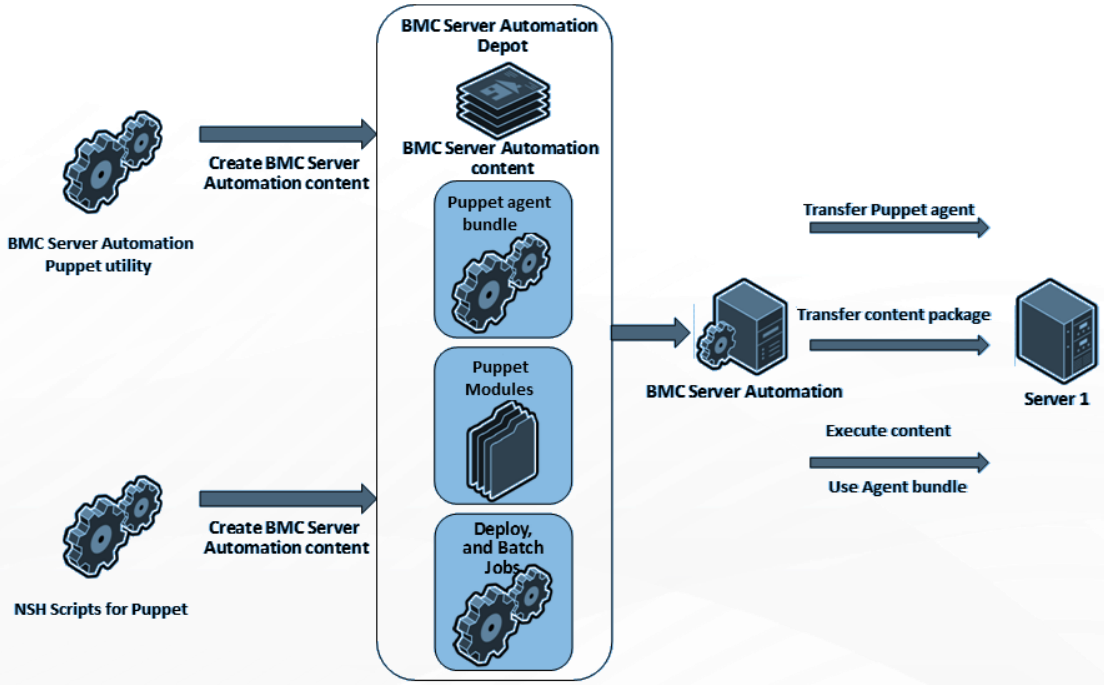
- Puppet follows a client-server or master-agent architecture. The Puppet master server stores configuration information, while Puppet agents run on each node in the infrastructure.



2.1.3 Puppet | Use cases



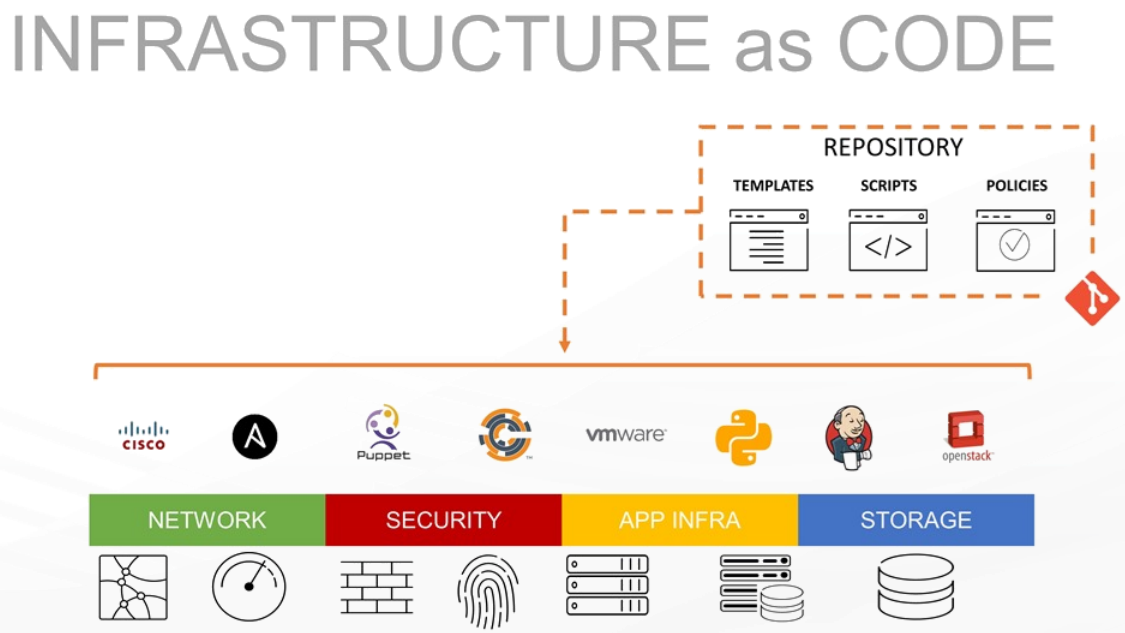
- Performance Measurements
- Hazards and Incidents Analysis
- Access and Backup Storage
- Version Control
- Resources Management
- Changes Approval Process
- Impact Analyses
- Roles & Responsibilities



1. **Configuration Management:** Puppet is commonly used for ensuring that servers and systems are configured consistently and according to organizational policies.

2. **Application Deployment:** Puppet facilitates the automation of deploying and managing applications by defining and enforcing the required configurations.

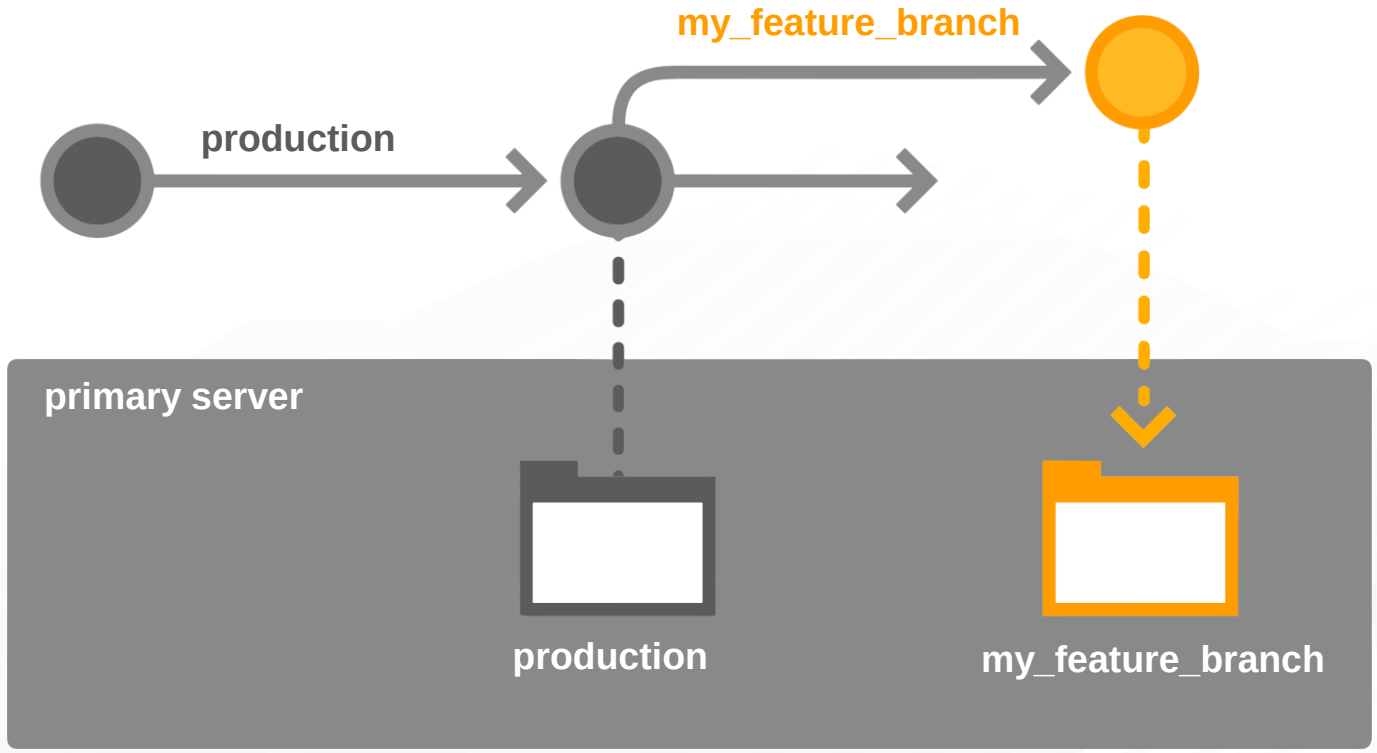
2.1.3 Puppet | Use cases



3. **Infrastructure as Code (IaC):** Puppet supports Infrastructure as Code principles, allowing users to express infrastructure configurations in code, version control them, and apply changes systematically.

4. **Security Compliance:** Puppet provides a framework for defining and enforcing security configurations.

2.1.3 Puppet | Use cases



5. **Orchestration and Workflow Automation:** Puppet can be used for orchestrating workflows that involve multiple servers or components.

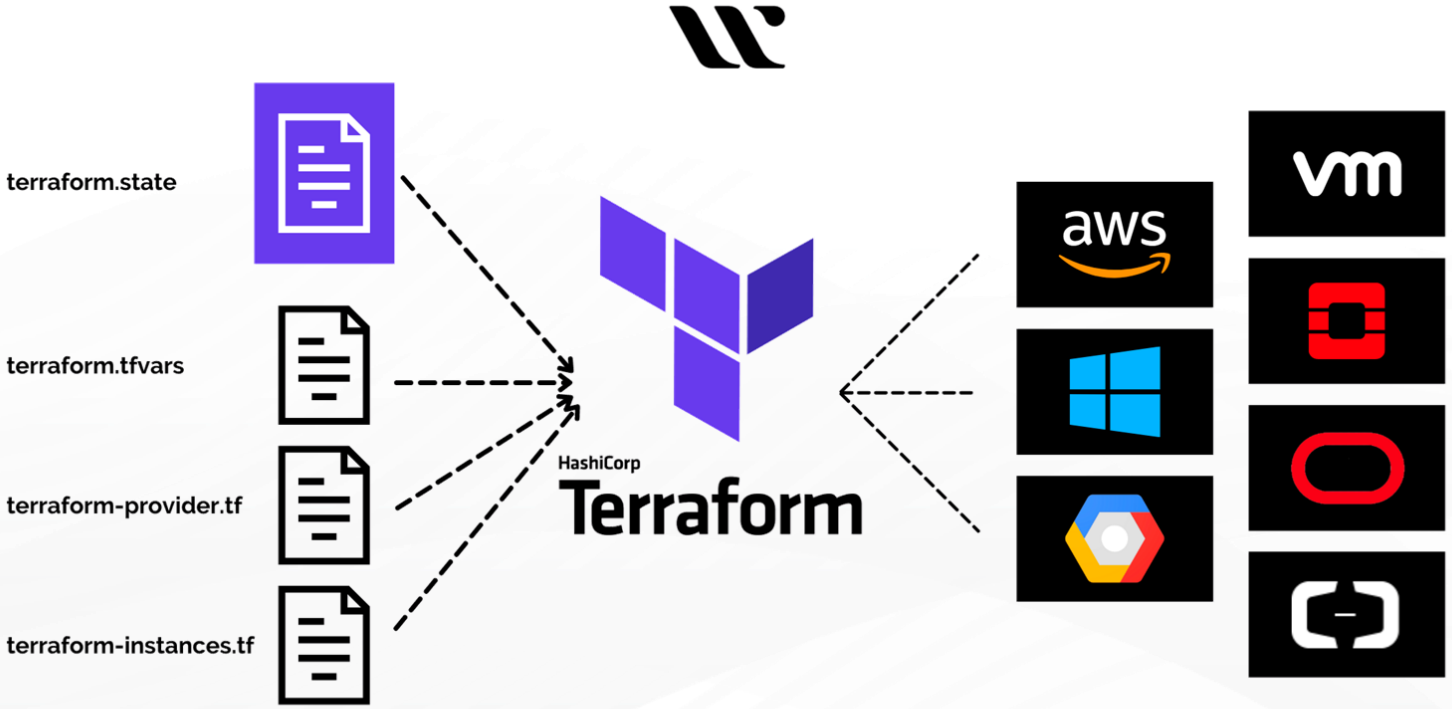
2.1.4 Terraform

Background

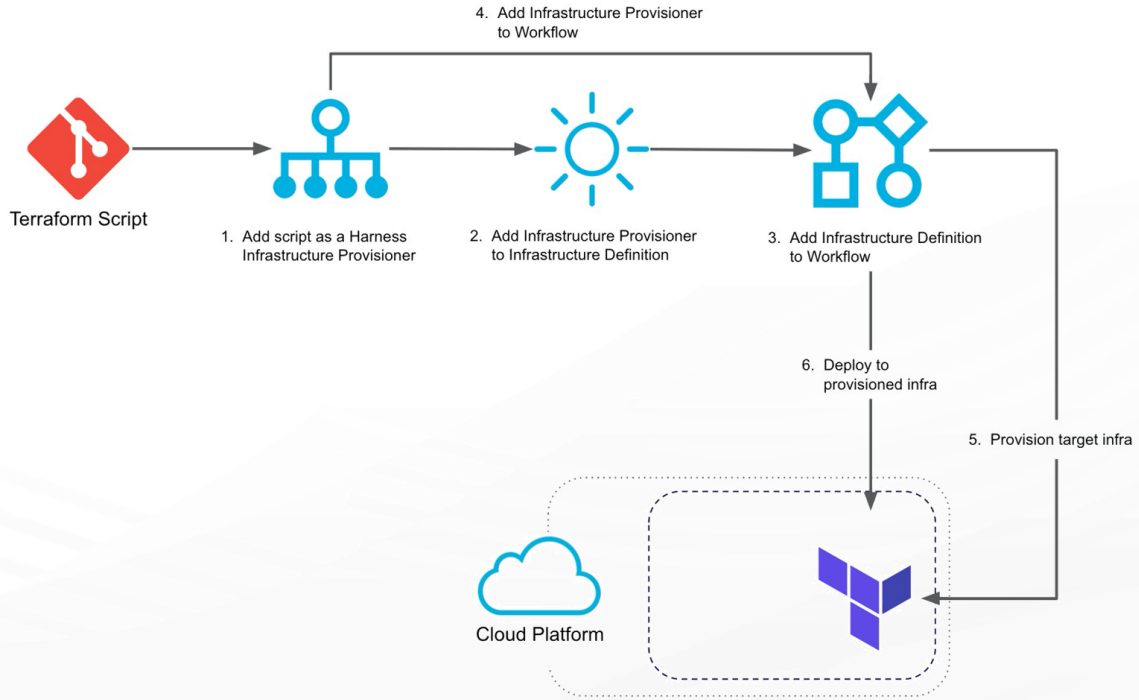
- Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp.

Philosophy and Design

- Terraform uses HashiCorp Configuration Language (HCL), a declarative language, for defining infrastructure configurations.



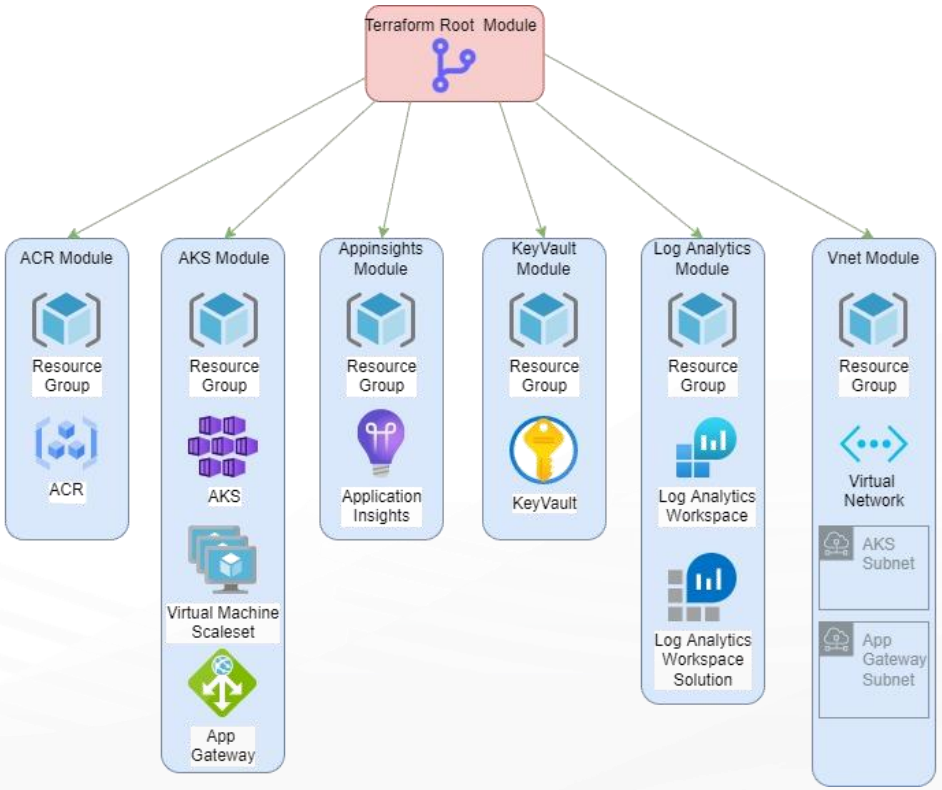
2.1.4 Terraform | Use cases

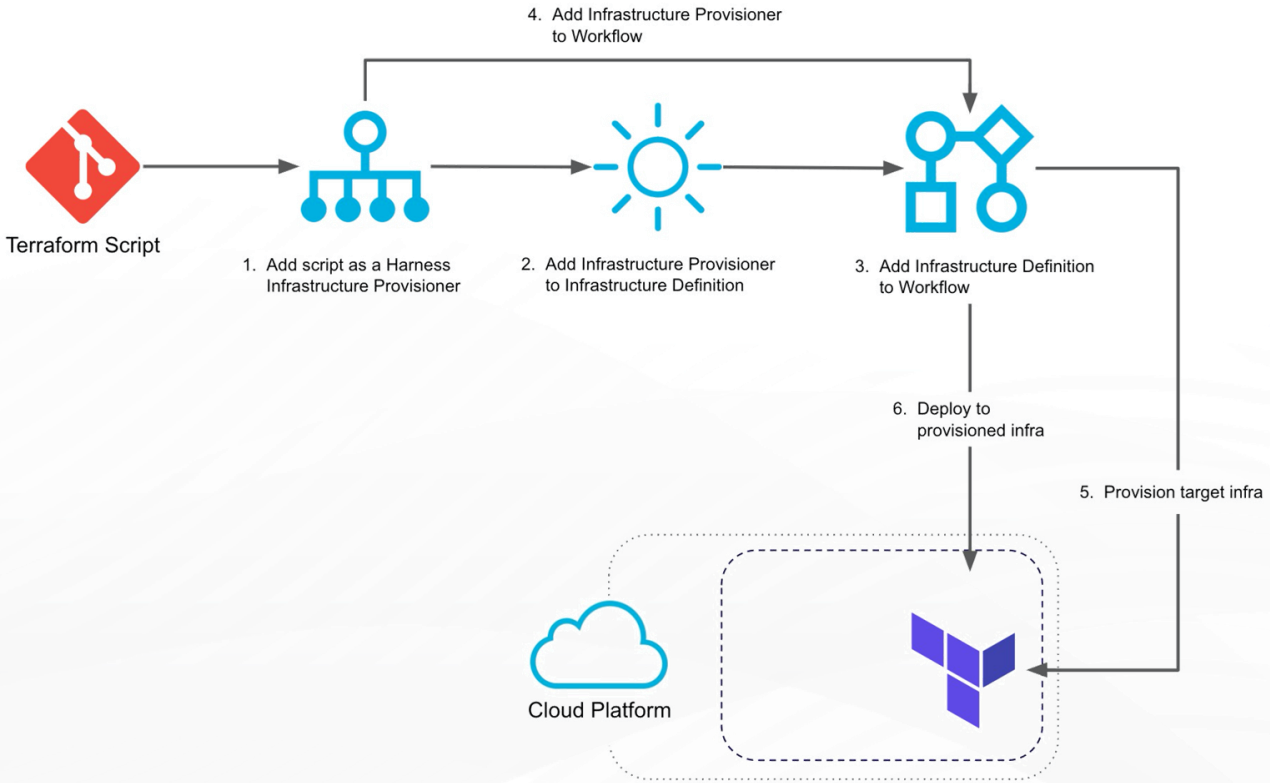


1. **Cloud Infrastructure Provisioning:** Terraform enables users to define cloud resources, such as virtual machines, storage, networking components, and more, in a declarative configuration.

2. **Multi-Cloud Deployments:** Terraform's multi-cloud support allows users to define infrastructure configurations that can be deployed seamlessly across different cloud providers.

2.1.4 Terraform | Use cases

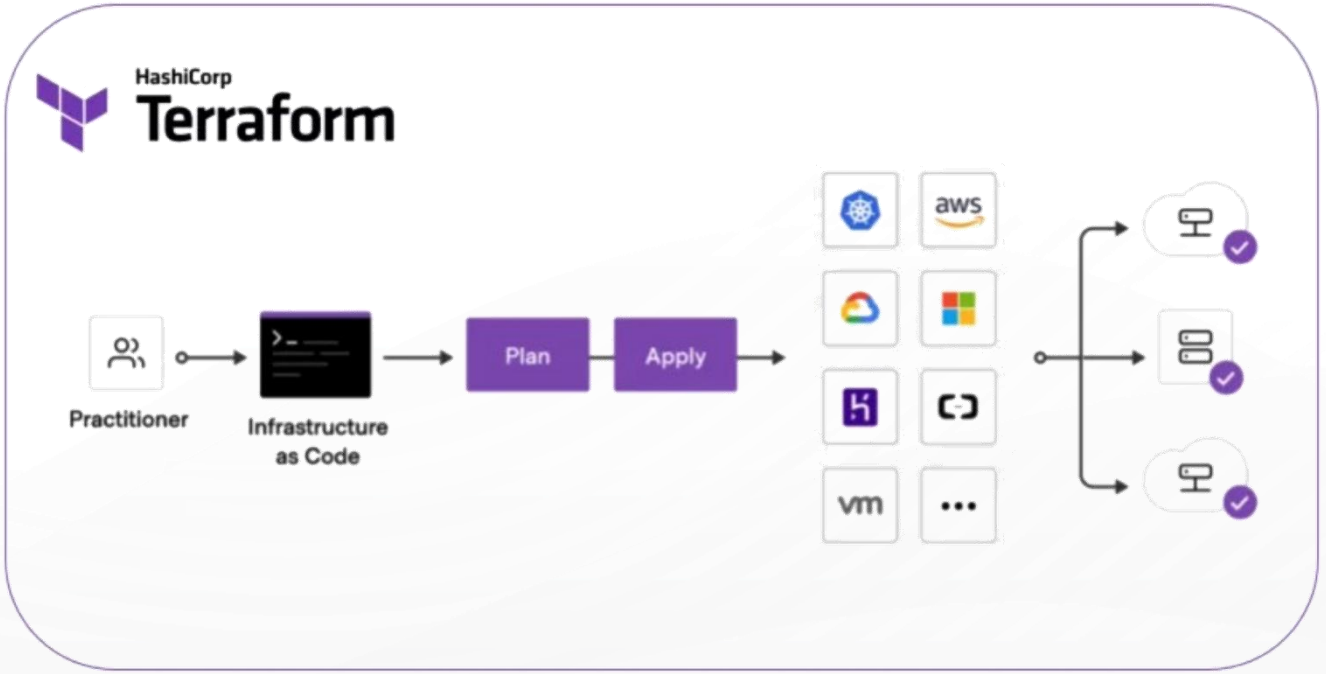




3. **Infrastructure Modules for Reusability**: Terraform modules allow users to encapsulate and share infrastructure components as reusable building blocks.

4. **Environment Provisioning:** Terraform can be used to define and provision environments needed for development, testing, and staging.

2.1.4 Terraform | Use cases



5. **Infrastructure as Code:** Organizations can use Terraform to define and provision servers, networks, and other resources in their own data centers, applying the benefits of Infrastructure as Code to their entire infrastructure.

2.2.0 Basic Terraform Commands

terraform init	Initializes a Terraform working directory by downloading providers and modules.
terraform plan	Generates an execution plan describing the changes to be made.
terraform apply	Applies the changes specified in the execution plan.
terraform destroy	Destroys the Terraform-managed infrastructure.
terraform validate	Validates the configuration files for syntax and other errors.
terraform fmt	Rewrites Terraform configuration files to a consistent format.

2.2.0 Basic Terraform Commands (Contd.)

terraform show	Displays the current state or a saved plan visually.
terraform state	Used for advanced state management, including import and manipulation.
terraform refresh	Updates the Terraform state file against real resources in the provider.
terraform output	Displays the output values from the Terraform configuration.
terraform import	Imports existing infrastructure into Terraform state.
terraform workspace	Manages Terraform workspaces for different environments.

2.2.2 Hands-on | Terraform Basics.

Activity 1: Terraform Basics

Description:

- Dive into Terraform with this hands-on activity. From exploring key concepts to configuring a random pet resource, participants navigate Terraform workflow—initializing, planning, applying changes, and cleaning up resources. Gain practical insights into Infrastructure as Code fundamentals in a guided journey through Terraform core functionalities.

Output from the Activity:

- Witness the creation of a random pet resource. Explore terraform.tfstate to understand state management. Cleanup resources gracefully with terraform destroy. Solidify your Terraform foundation and discover the power of declarative infrastructure management.

Click here to start the lab activity

2.2.3 Hands-on | Setting Up a Terraform Working Directory.

Activity 2: Setting Up a Terraform Working Directory

Description:

- This hands-on activity guides you in creating a project directory, defining infrastructure code using AWS, and initializing the Terraform project. Build a solid understanding of foundational Terraform concepts and workflows in a structured and engaging step-by-step experience.

Output from the Activity:

- Witness the birth of an AWS EC2 instance. Explore .terraform and terraform.tfstate for insights into Terraform's inner workings. Practice planning, applying changes, and resource cleanup, gaining proficiency in Terraform's essential commands. Elevate your Infrastructure as Code skills with this immersive, hands-on journey.

Click here to start the lab activity

2.2.4 Hands-on | Initializing, planning, applying, and managing Terraform changes.

Activity 3: Initializing, planning, applying, and managing Terraform changes.

Description:

- This hands-on activity immerses you in the complete workflow—initializing a project, planning changes, applying modifications, and managing Terraform state. Gain practical insights into resource manipulation and state management in a structured and guided experience.

Output from the Activity:

- Witness the evolution of infrastructure. Navigate terraform plan insights, apply changes, and grasp state intricacies. Resolve errors, inspect resource states, and master Terraform's state commands. Cap off the activity with resource cleanup, solidifying your expertise in Terraform's complete workflow.

Click here to start the lab activity

2.2.1 Hands-on | Provisioning Infrastructure on AWS with Terraform.

Activity 4: I Provisioning Infrastructure on AWS with Terraform.

Description:

- This hands-on activity explian provision AWS infrastructure using Terraform. From configuring the AWS provider to creating an S3 bucket, this activity offers practical insights into Terraform's cloud orchestration capabilities, enabling you to seamlessly manage cloud resources with Infrastructure as Code.

Output from the Activity:

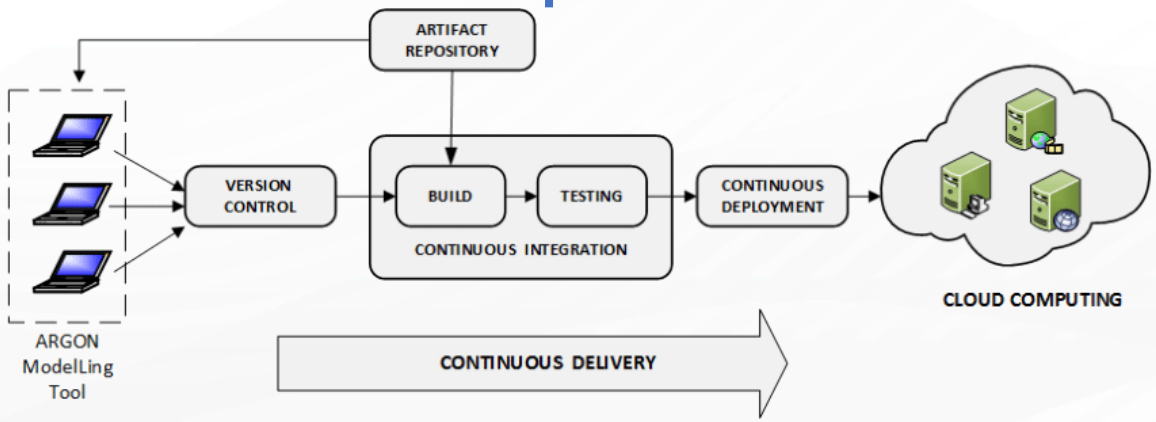
- Witness the birth of an S3 bucket on AWS. Review and apply changes, validate provisions on the AWS Console, and grasp Terraform's state management. Conclude with resource cleanup, solidifying your proficiency in Terraform's cloud provisioning prowess. Elevate your Infrastructure as Code expertise through this immersive, practical experience.

Click here to start the lab activity

2.2.5 Provisioning Infrastructure on a Cloud Provider.

Use the 'provider' block in Terraform to specify the target cloud provider.

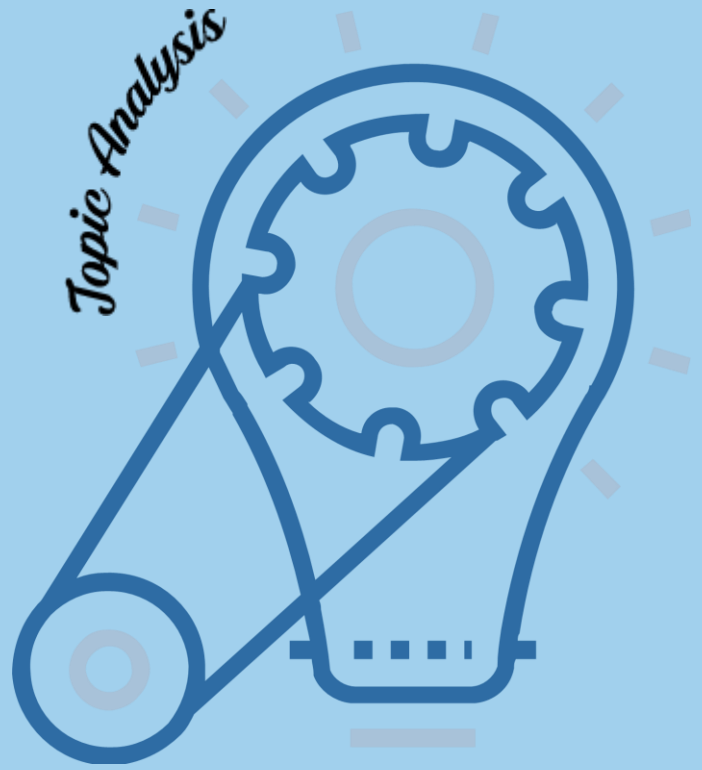
Define the infrastructure components using Terraform's declarative configuration language.



Use 'terraform init' and 'terraform apply' to provision the defined infrastructure on the cloud provider.'

Verify the provisioned infrastructure on the cloud provider platform.

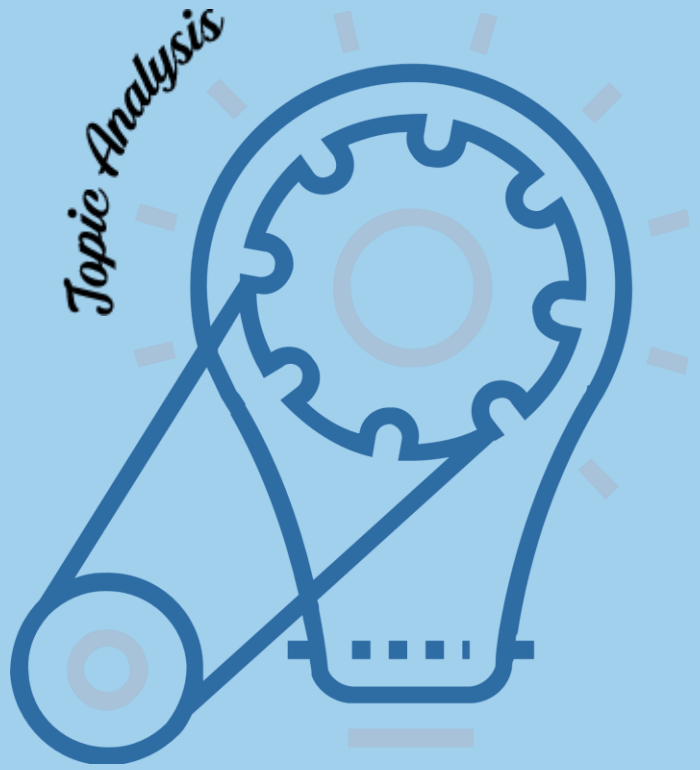
What did you grasp?



1. What is the primary benefit of using provisioning tools, such as Ansible, Chef, Puppet, and Terraform, in real-world IT scenarios?

   A. Ensuring Manual Intervention
   B. Adapting to Diverse Environments
   C. Slowing Down Deployment Processes
   D. Ignoring Security and Compliance

What did you grasp?



1. What does the terraform ‘apply’ command do?

   A. Initializes a Terraform working directory.
   B. Generates an execution plan.
   C. Destroys the Terraform-managed infrastructure.
   D. Applies the changes specified in the execution plan.

In a nutshell, we learnt:

- Introduction to Provisioning Tools
- Exploration of Key Tools
- Hands-on with Terraform
- Provisioning Infrastructure on a Cloud Provider

End of Module

Next Module: