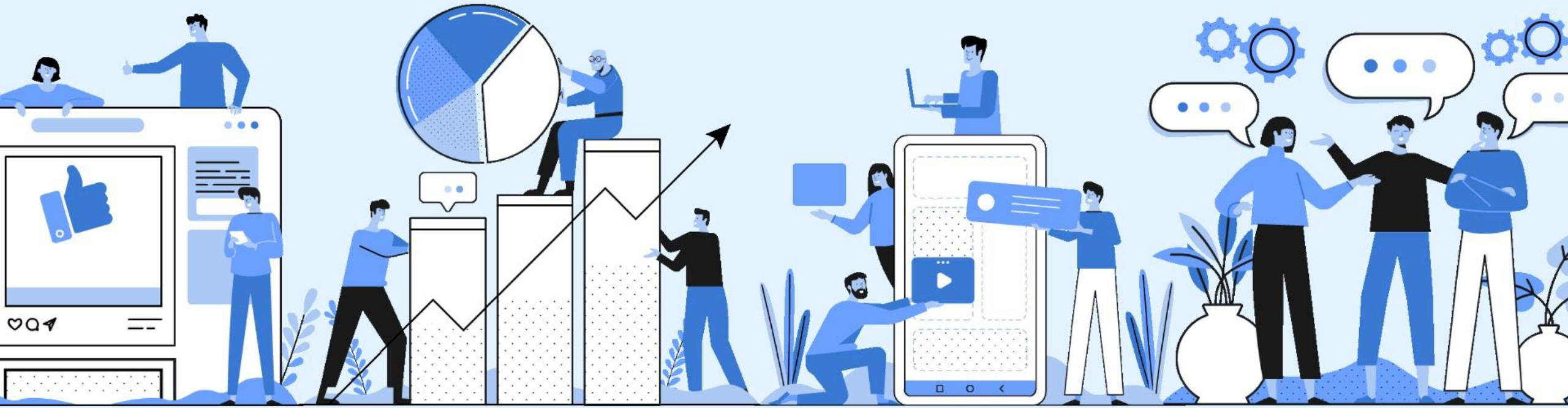
Infrastructure Orchestration

Cloud Computing – Your Ticket to Limitless Innovation.



Infrastructure Automation: Seven DevOps tools for orchestration secrets management and monitoring

Infrastructure automation can have many benefits, including:

- ✓ **Security**: Automation can reduce human errors that can lead to security threats
- ✓ **Cost optimization**: Automation can help optimize costs and the quality of IT architecture
- ✓ **Automate deployments**: Automation can increase confidence in deployments and understanding of infrastructure configuration
- ✓ **Cost visibility**: Automation can reduce errors and deviations, and decrease the chances of infrastructure incompatibility
- ✓ **Accelerate automation**: Automation can streamline repetitive tasks such as provisioning, configuring, deploying, and decommissioning
- ✓ **Consistency across environments**: Automation can simplify provisioning and ensure infrastructure consistency

Infrastructure Automation: Seven DevOps tools for orchestration secrets management and monitoring

Infrastructure automation can have many benefits, including:

- ✓ Faster DevOps adoption
- ✓ Improved user experience via self-service portals
- ✓ More efficient workflows
- ✓ Faster updates

Infrastructure Orchestration tools

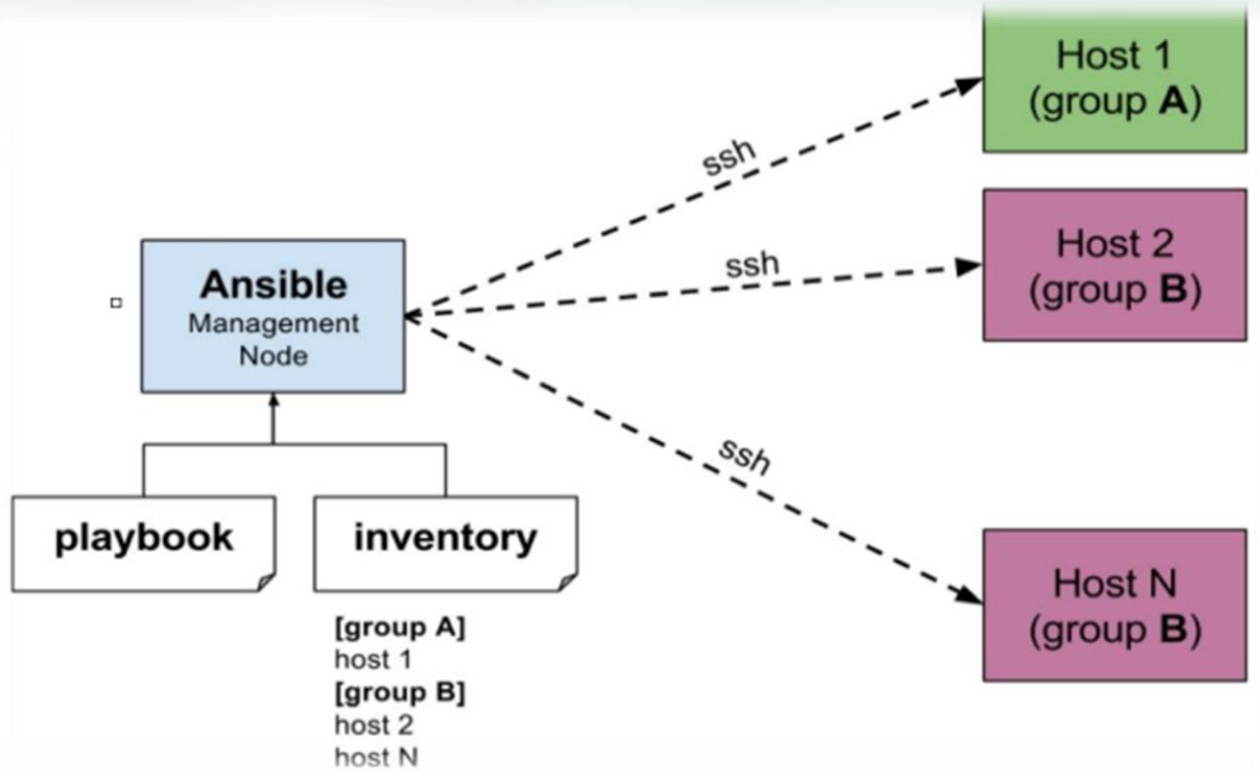
- ✓ **Ansible,**
- ✓ **Chef,**
- ✓ **Puppet,**
- ✓ **Docker,**
- ✓ **Terraform,**
- ✓ **Aws Cloud Formation,**
- ✓ **Jenkins**

Ansible

- ✓ **Ansible** is simple open source IT engine which automates application deployment, intra service orchestration, cloud provisioning and many other IT tools.
- ✓ Ansible is easy to deploy because it does not use any agents or custom security infrastructure. Ansible works by connecting your nodes through ssh(by default).
- ✓ But if you want other method for connection like Kerberos, Ansible gives that option to you.
- ✓ Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e. YAML
- ✓ It's a human-readable data serialization language & is commonly used for configuration files, but could be used in many applications where data is being stored.
- ✓ After connecting to your nodes, Ansible pushes small programs called as "Ansible Modules".
- ✓ Ansible runs that modules on your nodes and removes them when finished.
- ✓ Ansible manages your inventory in simple text files (These are the hosts file).

How Ansible Works?

- ✓ Ansible works by connecting to your nodes and pushing out small programs, called "**Ansible modules**" to them.
- ✓ Ansible then executes these modules (over SSH(Linux) windows(winrm)), and removes them when finished.
- ✓ Your library of modules can reside on any machine, and there are no servers, daemons, or databases required.
- ✓ **The management node** in the above picture is the controlling node (managing node) which controls the entire execution of the playbook.
- ✓ It's the node from which you are running the installation.
- ✓ The inventory file provides the list of hosts where the Ansible modules needs to be run and the management node does a SSH connection and executes the small modules on the hosts machine and installs the product/software



Beauty of Ansible is that it removes the modules once those are installed so effectively it connects to host machine , executes the instructions and if it's successfully installed removes the code which was copied on the host machine which was executed.

Chef

- ✓ Chef is an open source technology developed by Opscode. Adam Jacob, co-founder of Opscode is known as the founder of Chef.
- ✓ This technology uses Ruby encoding to develop basic building blocks like recipe and cookbooks.
- ✓ It is developed on the basis of Ruby DSL language.
- ✓ Chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.
- ✓ Chef have got its own convention for different building blocks, which are required to manage and automate infrastructure.

Chef

Following are the most prominent features of Chef –

- ✓ Chef uses popular Ruby language to create a domain-specific language.
- ✓ Chef does not make assumptions on the current status of a node. It uses its mechanisms to get the current status of machine.
- ✓ Chef is ideal for deploying and managing the cloud server, storage, and software.

Advantages of Chef

Chef offers the following advantages –

- ✓ Lower barrier for entry – As Chef uses native Ruby language for configuration, a standard configuration language it can be easily picked up by anyone having some development experience.
- ✓ Excellent integration with cloud – Using the knife utility, it can be easily integrated with any of the cloud technologies. It is the best tool for an organization that wishes to distribute its infrastructure on multi-cloud environment.

Chef

Disadvantages of Chef

Some of the major drawbacks of Chef are as follows –

- ✓ One of the huge disadvantages of Chef is the way cookbooks are controlled. It needs constant babying so that people who are working should not mess up with others cookbooks.
- ✓ Only Chef solo is available.
- ✓ In the current situation, it is only a good fit for AWS cloud.
- ✓ It is not very easy to learn if the person is not familiar with Ruby.
- ✓ Documentation is still lacking.

Chef

Resource

It is the basic component of a recipe used to manage the infrastructure with different kind of states. There can be multiple resources in a recipe, which will help in configuring and managing the infrastructure.

package – Manages the packages on a node

service – Manages the services on a node

user – Manages the users on the node

group – Manages groups

template – Manages the files with embedded Ruby template

cookbook_file – Transfers the files from the files subdirectory in the cookbook to a location on the node

file – Manages the contents of a file on the node

directory – Manages the directories on the node

execute – Executes a command on the node

cron – Edits an existing cron file on the node

Chef

Attribute

- ✓ They are basically settings. They can be thought of as a key value pair of anything which one wants to use in the cookbook. There are several different kinds of attributes that can be applied, with a different level of precedence over the final settings that the node operates under.

File

- ✓ It's a subdirectory within the cookbook that contains any static file which will be placed on the nodes that uses the cookbooks. A recipe then can be declared as a resource that moves the files from that directory to the final node.

Templates

- ✓ They are similar to files, but they are not static. Template files end with the .ebr extension, which means they contain embedded Ruby. They are mainly used to substitute an attribute value into the files to create the final file version that will be placed on the node.

Metadata.rb

- ✓ It is used to manage the metadata about the package. This includes details like the name and details of the package. It also includes things such as dependency information that tells which cookbooks this cookbook needs to operate. This allows the Chef server to build the run-list of the node correctly and ensures that all of the pieces are transferred correctly.

Chef

Default Cookbook Structure

```
C:\chef\cookbooks\nginx>tree
Folder PATH listing for volume Local Disk
Volume serial number is BE8B-6427
C:.
├───attributes
├───definitions
├───files
│   └───default
├───libraries
├───providers
├───recipes
├───resources
├───templates
│   └───default
```

Puppet

Chef and Puppet are automation platforms that allow you to use code to automate the configurations of your servers. They have many similar capabilities and use cases, but there are some differences:

Use case: Puppet is known to be better for managing large-scale deployments across data centers and the cloud. Chef is widely used to manager smaller, less complex infrastructure.

Configuration language
Chef uses Ruby DSL (Domain Specific Language), a developer-oriented language that's challenging to learn. Puppet employs the PuppetDSL, another difficult to learn language.

Interoperability
Both the Chef server and the Ruby server work only on a Linux/Unix machine.

- Using Puppet is like writing configuration files whereas using Chef is like programming the control of your nodes.
- Chef offers flexibility and a strong focus on infrastructure as code.
- Puppet excels in managing large-scale environments.

AWS CloudFormation

- ✓ CloudFormation is a service provided by AWS for designing our own infrastructure using code, i.e. infrastructure as code.
- ✓ Currently, CloudFormation supports two languages, JSON and YAML.You can write your code with one of the languages.
- ✓ CloudFormation comes with great features, being able to update your infrastructure whenever you want and also having the ability to delete the stack in case you don’t need it.
- ✓ A fascinating feature of CloudFormation is that it saves more time in building infrastructure and helps in focusing on the development.
- ✓ It is also possible to replicate our infrastructure in a short amount of time.
- ✓ It eliminates human error and works according to the code you have written. It consists of two main components,Stack and Templates.

AWS CloudFormation

- ✓ CloudFormation Template
- ✓ 1.It consists of various sections like
  - AWS Template Format Version
  - Description
  - Metadata
  - Parameters
  - Mappings
  - Conditions
  - Resources
  - (Required Field)
  - Outputs
- ✓ It is not mandatory that the template requires all the above-mentioned sections. By using only the Resources section, we will be able to create a template.
- ✓ The resources section plays an important role in the template creation.
- ✓ For example, to create an EC2 instance, a template shall consist of various parameters such as key name, image ID, instance type.
- ✓ It is also possible to create two resources in the same template and refer to one from another, i.e. attaching an elastic IP with an EC2 instance.

Jenkins

- ✓ Jenkins is an open source automation tool written in Java programming language that allows continuous integration.
- ✓ Jenkins **builds** and **tests** our software projects which continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.
- ✓ It also allows us to continuously deliver our software by integrating with a large number of testing and deployment technologies.
- ✓ Jenkins offers a straightforward way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks.
- ✓ With the help of Jenkins, organizations can speed up the software development process through automation. Jenkins adds development life-cycle processes of all kinds, including build, document, test, package, stage, deploy static analysis and much more.
- ✓ Jenkins achieves CI (Continuous Integration) with the help of plugins. Plugins is used to allow the integration of various DevOps stages. If you want to integrate a particular tool, you have to install the plugins for that tool.

What is Continuous Integration?

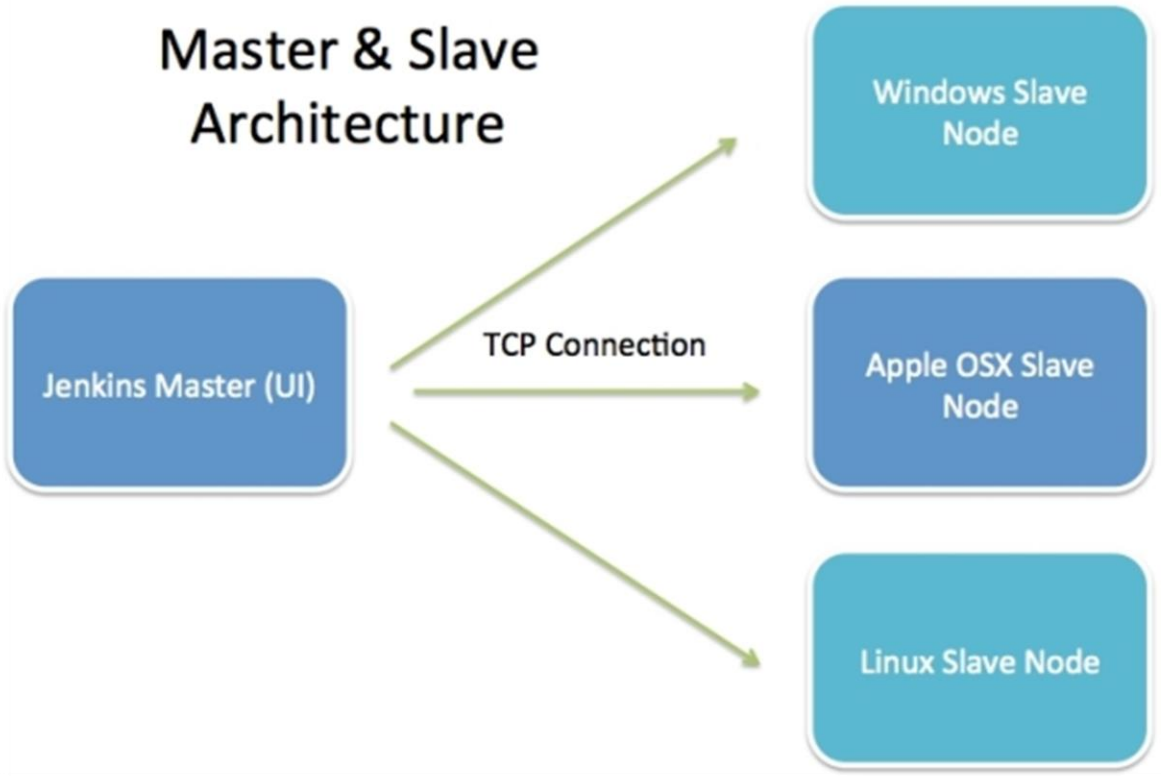
- ✓ **Continuous Integration (CI)** is a development practice in which the developers are needs to commit changes to the source code in a shared repository at regular intervals. Every commit made in the repository is then built. This allows the development teams to detect the problems early.
- ✓ Continuous integration requires the developers to have regular builds. The general practice is that whenever a code commit occurs, a build should be triggered.

How Jenkins works:

- ✓ First of all, a developer commits the code to the source code repository. Meanwhile, the Jenkins checks the repository at regular intervals for changes.
- ✓ Soon after a commit occurs, the Jenkins server finds the changes that have occurred in the source code repository. Jenkins will draw those changes and will start preparing a new build.
- ✓ If the build fails, then the concerned team will be notified.
- ✓ If built is successful, then Jenkins server deploys the built in the test server.
- ✓ After testing, Jenkins server generates a feedback and then notifies the developers about the build and test results.
- ✓ It will continue to verify the source code repository for changes made in the source code and the whole process keeps on repeating.

Jenkins Architecture

- ✓ Jenkins follows **Master-Slave** architecture to manage distributed builds. In this architecture, slave and master communicate through TCP/IP protocol.
- ✓ Jenkins architecture has two components:
  - ❑ Jenkins Master/Server
  - ❑ Jenkins Slave/Node/Build Server



Jenkins Master

- ❑ The main server of Jenkins is the Jenkins Master. It is a web dashboard which is nothing but powered from a war file. By default it runs on 8080 port. With the help of Dashboard, we can configure the jobs/projects but the build takes place in Nodes/Slave. By default one node (slave) is configured and running in Jenkins server. We can add more nodes using IP address, user name and password using the ssh, jnlp or webstart methods.

✓ **The server's job or master's job is to handle:**

- Scheduling build jobs.
- Dispatching builds to the nodes/slaves for the actual execution.
- Monitor the nodes/slaves (possibly taking them online and offline as required).
- Recording and presenting the build results.
- A Master/Server instance of Jenkins can also execute build jobs directly.

✓ **Jenkins Slave**

- Jenkins slave is used to execute the build jobs dispatched by the master. We can configure a project to always run on a particular slave machine, or particular type of slave machine, or simple let the Jenkins to pick the next available slave/node.
- As we know Jenkins is developed using Java is platform independent thus Jenkins Master/Servers and Slave/nodes can be configured in any servers including Linux, Windows, and Mac.

In programming, build is a compiled version of a program.
Compile means to convert a program into a lower-level form or machine-code.

Terraform

- ✓ Terraform is an Infrastructure as Code (IaC) tool. It's primarily used by DevOps teams to automate infrastructure tasks.
- ✓ Terraform is an open-source tool that helps create, change, and improve infrastructure. It's used to provision and manage infrastructure on-premises and in the cloud.
- ✓ Terraform can manage infrastructure on any platform or service that supports an API.
- ❑ **Some of the main use cases of Terraform include:**
  - ✓ Provisioning cloud resources
  - ✓ Automating infrastructure deployments
  - ✓ Versioning infrastructure
- ✓ Terraform is written in the Go language and created by HashiCorp. Users define and provide data center infrastructure using a declarative configuration language known as HashiCorp Configuration Language (HCL), or optionally JSON

What is Terraform Used For?

- ✓ Terraform can be connected with different infrastructure hosts and achieve **complex management scenarios** and **compliance across multiple clouds**. Its configuration can be easily packaged, shared, and reused in the form of Terraform modules.

How Does Terraform Work?

- ✓ Terraform can be thought of as consisting of **two main parts**: ***Terraform Core*** *and* ***Plugins***.

- ✓ ***Terraform Core*** takes into consideration the current state and evaluates it against your desired configuration. It then proposes a plan to add or remove infrastructure components as needed. Next, it takes care of provisioning or decommissioning any resources if you choose to apply the plan.

- ✓ ***Terraform Plugins*** provide a mechanism for Terraform Core to communicate with your infrastructure host or SaaS providers. Terraform Providers and Provisioners are examples of plugins as mentioned above. Terraform Core communicates with the plugins via Remote Procedure Call (RPC).

What is Terraform Used For?

- ✓ Terraform can be connected with different infrastructure hosts and achieve **complex management scenarios** and **compliance across multiple clouds**. Its configuration can be easily packaged, shared, and reused in the form of Terraform modules.

How Does Terraform Work?

- ✓ Terraform can be thought of as consisting of **two main parts**: ***Terraform Core*** *and* ***Plugins***.

- ✓ ***Terraform Core*** takes into consideration the current state and evaluates it against your desired configuration. It then proposes a plan to add or remove infrastructure components as needed. Next, it takes care of provisioning or decommissioning any resources if you choose to apply the plan.

- ✓ ***Terraform Plugins*** provide a mechanism for Terraform Core to communicate with your infrastructure host or SaaS providers. Terraform Providers and Provisioners are examples of plugins as mentioned above. Terraform Core communicates with the plugins via Remote Procedure Call (RPC).