Infrastructure as a Code(IaaC) - Self Notes prepared by Kamalesh

Explanation: Infrastructure as Code (IaC) is a DevOps practice that involves managing infrastructure such as networks, virtual machines, load balancers, and connection topologies using a descriptive model. IaC uses versioning and DevOps methodology to define and deploy infrastructure. It allows teams to work together with a unified set of practices and tools to deliver applications and their supporting infrastructure rapidly and reliably at scale.

- IaC helps avoid manual configuration and enforces consistency by representing desired environment states via well-documented code in formats such as JSON.
- Infrastructure deployments with IaC are repeatable and prevent runtime issues caused by configuration drift or missing dependencies.
- Release pipelines execute the environment descriptions and version configuration models to configure target environments .
- IaC also helps DevOps teams test applications in production-like environments early in the development cycle.
- The cloud dynamically provisions and tears down environments based on IaC definitions.
- The infrastructure code itself can be validated and tested to prevent common deployment issues.

Tools used in IaaC:

- Terraform
- Ansible
- Chef
- Puppet
- SaltStack
- AWS CloudFormation
- Azure Resource Manager
- Google Cloud Deployment Manager

These tools help automate the provisioning and configuration of infrastructure, which can lead to faster and more reliable deployments.

Terraform:

- Terraform is an Infrastructure as Code (IaC) tool developed by HashiCorp. It allows engineers to define their software infrastructure in code. This means that you can manage your infrastructure in the same way that you manage your application code - with version control, automation, and collaboration.

Infrastructure as a Code(IaaC) - Self Notes prepared by Kamalesh

- HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle.
- Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.
- One of the key features of Terraform is its declarative syntax. This means that you define what your desired end state is, and Terraform figures out the best way to achieve that.
- It lets you define resources and infrastructure in human-readable, declarative configuration files, and manages your infrastructure's lifecycle.
- Terraform can manage infrastructure on multiple cloud platforms. The human-readable configuration language helps you write infrastructure code quickly.
- Terraform's state allows you to track resource changes throughout your deployments. You can commit your configurations to version control to safely collaborate on infrastructure.
- Users define and provide data center infrastructure using a declarative configuration language known as HashiCorp Configuration Language (HCL), or optionally JSON.
- Terraform manages external resources (such as public cloud infrastructure, private cloud infrastructure, network appliances, software as a service, and platform as a service) with "providers".
- **HashiCorp Configuration Language** is a structured configuration language to describe the infrastructure that terraform manages. Its main purpose is to declare the resources which represent infrastructure objects.
- A Terraform configuration is a complete document in the Terraform language that tells Terraform how to manage a given collection of infrastructure.
- HCL has expanded as a more general configuration language. It's visually similar to JSON with additional data structures and capabilities built-in.

Head over to this for HCL notes: [Introduction to HCL and HCL tooling - Octopus Deploy]

Software download link: [Download Terraform software here (HashiCorp)]
- Edit the system environmental variables before installing the software.

- While it is true that YAML is supported in Terraform, it is important to note that it is not a direct replacement for HashiCorp Configuration Language (HCL).

  Terraform primarily uses HCL for writing its configuration files, which have a specific syntax and structure. HCL is specifically designed for infrastructure-as-code (IaC) purposes and provides features that make it easier to describe resources and dependencies.

  However, Terraform also allows the use of YAML as an alternative configuration format. This can be useful if you are more comfortable working with YAML or if you have existing YAML-based infrastructure definitions.

When using YAML with Terraform, you need to convert YAML into the equivalent HCL syntax using a tool called `hcl2json`. This tool allows you to convert YAML configurations to HCL JSON syntax, which can then be used by Terraform.

In summary, while you can use YAML with Terraform, it requires converting the YAML syntax to HCL JSON syntax, and it is not a direct replacement for HCL.

- Terraform primarily uses HCL for its configuration1. However, you can use YAML for certain parts of your Terraform configuration.
- For example, you can manage your infrastructure using both Terraform and YAML files. This can be done by parsing the YAML configuration in Terraform using the `yamldecode` function.

The core Terraform workflow consists of three stages:

- Write: You define resources, which may be across multiple cloud providers and services. For example, you might create a configuration to deploy an application on virtual machines in a Virtual Private Cloud (VPC) network with security groups and a load balancer.
- Plan: Terraform creates an execution plan describing the infrastructure it will create, update, or destroy based on the existing infrastructure and your configuration.
- Apply: On approval, Terraform performs the proposed operations in the correct order, respecting any resource dependencies. For example, if you update the properties of a VPC and change the number of virtual machines in that VPC, Terraform will recreate the VPC before scaling the virtual machines.

Ansible(Configuration Management tool)

- Ansible is an open-source, cross-platform tool for resource provisioning automation that DevOps professionals use for continuous delivery of software code.
- It is the most preferred DevOps tool for orchestration, automation, configuration, and managing IT infrastructure.
- Ansible is the first human-readable automation language that can be read and written across IT.

Infrastructure as a Code(IaaC) - Self Notes prepared by Kamalesh

- It is a simple, but powerful, server and configuration management tool that can help with CI/CD, which is one aspect of DevOps.
- It is an open-source software platform utilized for automating configuration management.

Steps to Install Ansible in windows:

1. Run this command in command prompt under administrator authentication:

   -> pip install ansible **#This will install all the configuration and dependencies of ansible in the install**

2. **Run this command to check the version of ansible:**

   -> Ansible –version