Features of Terraform

Terraform works by building a graph database that provides operators with insight into resource dependencies. It also generates an execution plan that allows operators to see what sequence of steps Terraform will take when a setting is applied or a change is made.

Alternatives to Terraform

Two examples of alternative tools to **Terraform are Pulumi and SaltStack.**

What are Kubernetes architecture components?

The main components of a Kubernetes cluster include:

1. **Nodes**: Nodes are VMs or physical servers that host containerized applications. Each node in a cluster can run one or more application instance. There can be as few as one node, however, a typical Kubernetes cluster will have several nodes (and deployments with hundreds or more nodes are not uncommon).
2. **Image Registry**: Container images are kept in the registry and transferred to nodes by the control plane for execution in container pods.
3. **Pods**: Pods are where containerized applications run. They can include one or more containers and are the smallest unit of deployment for applications in a Kubernetes cluster. .
4. **How does Terraform work?**

   Terraform allows users to define their entire infrastructure simply by using configuration files and version control. When a command is given to deploy and run a server, database or load balancer, Terraform parses the code and translates it into an application programming interface (API) call to the resource provider. Because Terraform is open source, developers are always able to extend the tool's usefulness by writing new plugins or compiling different versions of existing plugins.

AWS CodeBuild

Benefits

- Fully managed build service
- Continuous scaling
- Pay as you go
- Extensible
- Secure
- Enables continuous integration and delivery

Why Nagios?

It is used for monitoring the performance issues of servers. Easily find the root cause of any problem. It is also used to detect all the possible networks.

What is Amazon EKS?

Amazon Elastic Kubernetes Service (Amazon EKS) lets you deploy and manage Kubernetes on AWS, without having to run Kubernetes directly on EC2 machines

Advanced DevOps. The term “Advanced” here indicates the level of expertise, skills, tools required to do some additional automation and Integration in the CI/CD pipeline such as:

- *Managing Infrastructure as a Code and Functions Integration with the Cloud Eco System and Agility needs*
- *Clustering*
- *Dealing with Feature Flags*
- *Automatic Rollback, Provisioning*
- *Deployment Testing, Injecting Faults*

Advanced devops is when you are using various tool stack in your environment and leveraging one click deployments to production i.e. from requirement gathering to making your application live to production and not only this you are doing monitoring of your applications like checking health,avoiding downtime of your applications using monitoring tools.

Infrastructure as code is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.
Infrastructure as code (IaC) means to manage your IT infrastructure using configuration files.

Infrastructure as Code Benefits:

- **Speed**

The first significant benefit IaC provides is speed. Infrastructure as code enables you to quickly set up your complete infrastructure by running a script. IaC can make the entire software development lifecycle more efficient.

- **Consistency**

Manual processes result in mistakes, period. Humans are fallible. Our memories fault us. *IaC solves that problem by having the config files themselves be the single source of truth.* That way, you guarantee the same configurations will be deployed over and over, without discrepancies.

- **Accountability**

This one is quick and easy. Since you can version IaC configuration files like any source code file, you have full traceability of the changes each configuration suffered. No more guessing games about who did what and when.

What Are Amazon S3 Buckets?
Amazon Simple Storage Service (S3) *is a public cloud storage platform that is highly scalable, fast, reliable, and inexpensive.*
Amazon S3 buckets are a part of Amazon Web Services (AWS) and come with a user interface that enables users to store and retrieve data from anywhere on the web. In addition

to simple storage, *S3 buckets can be used to host static HTML websites as well as complex dynamic web applications.*

Many organizations use S3 buckets for backup and recovery, and for storing large amounts of data for analytics and other purposes.

Amazon S3 Security Features

One of the most important security features provided by AWS is the event logs, which can be enabled and disabled via the interface. By enabling logging, organizations can keep track of how their data is accessed, shared, modified, or removed.
The logs record a wide range of events, which include the date and time content is accessed and the protocols that are used (HTTP, FTP, etc.). The logs also include HTTP status codes, Turnaround time, and HTTP request messages.

SonarQube an *open source platform for continuous inspection of code quality* to perform automatic reviews with static analysis of code to:

- Detect Bugs
- Code Smells
- Security Vulnerabilities
- Centralize Quality

SonarQube can be **used** in combination with Azure DevOps. If you do not know SonarQube, it is tool that centralizes static code analysis and unit test coverage. *It can be used across multiple languages and for a single project up to enterprise scale.* SonarQube can be used as a SaaS product or hosted on your own instance.

AWS CodeBuild is a fully *managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy.* With CodeBuild, you don't need to provision, manage, and scale your own build servers.

Benefits

- Fully managed build service
- Continuous scaling
- Pay as you go
- Extensible
- Secure
- Enables continuous integration and delivery

AWS CodeBuild belongs to a family of AWS Code Services, which you can use to create complete, automated software release workflows for continuous integration and delivery (CI/CD). You can also integrate CodeBuild into your existing CI/CD workflow. For example, you can use CodeBuild as a worker node for your existing Jenkins server setup for distributed builds.

- Secure

What is a Lambda function?

The code you run on AWS Lambda is called a "Lambda function."

AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. You can **use** AWS Lambda to extend other AWS services with custom logic, or create your own back end services that operate at AWS scale, performance, and security.

What is serverless computing?

Serverless computing is a cloud-based service where a cloud provider manages the server.

The cloud provider dynamically allots compute storage and resources as needed to execute each line of code.

With serverless computing, the service provider takes care of all the infrastructure (server-side IT), which means all your team needs to do is write code. (Serverless computing is also known as serverless architecture, and it relates closely to functions-as-a-service, or FaaS.)

Serverless framework

Serverless frameworks are *free, open-sourced software for coding*. It builds, compiles and packages code for serverless implementation. It then will deploy it to the cloud.

Developers **use** serverless framework to simplify and speed the coding process for their teams. It also makes scaling easier across developers and helps to decrease configuration time.

A couple leading serverless frameworks are:
AWS Serverless Application Model (SAM) consists of template specification and a command line interface (CLI).
Serverless has both free and paid options.

What is a CI/CD pipeline?

The primary goal of a CI/CD pipeline is to automate the software development lifecycle (SDLC).

The pipeline will cover many aspects of a software development process, from writing the code and running tests to delivery and deployment. Simply stated, a CI/CD pipeline integrates automation and continuous monitoring into the development lifecycle. This kind of pipeline, which encompasses all the stages of the software development life cycle and connects each stage, is collectively called a CI/CD pipeline.

Stages in a CI/CD pipeline

A CI/CD pipeline can be divided into four main stages:

1. **Source**
2. **Build**
3. **Test**
4. **Deployment**

Source stage:
This is the first stage of any CI/CD pipeline. In this stage, the CI/CD pipeline will get *triggered by any change in the program* or a preconfigured flag in the code repository (repo). This stage focuses on source control, covering version control and tracking changes.

Build stage:
This second stage of the pipeline *combines the source code* with all its dependencies to aexecutable/runnable instance of the development.
This stage covers:

- Software builds
- Other kinds of buildable objects, such as Docker containers
- This stage is the most important one. Failure in a build here could indicate a fundamental issue in the underlying code.

Test stage:
The test stage incorporates all the automated testing to validate the behavior of the software. The goal of this *stage is to prevent software bugs from reaching end-users.* Multiple types of testing from integration testing to functional testing can be incorporated into this stage. This stage will also expose any errors with the product.

Deploy stage:
This is the final stage of the pipeline. After passing all the previous stages, the package is now *ready to be deployed.* In this stage, the package is deployed to proper environments as first to a staging environment for further quality assurance (QA) and then to a production environment.

Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the Java programming language. It is used to implement CI/CD workflows, called pipelines.

Advantages of Jenkins include:

1. It is an open-source tool with great community support.
2. It is easy to install.
3. It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share it with the community.
4. It is free of cost.
5. It is built with Java and hence, it is portable to all the major platforms.

HashiCorp Terraform is an *open source infrastructure as code (IaC) software tool that allows DevOps engineers to programmatically provision the physical resources an application requires to run.*
Terraform users define and enforce infrastructure configurations by using a JSON-like configuration language called **HCL (HashiCorp Configuration Language).** HCL's simple syntax makes it easy for DevOps teams to provision and re-provision infrastructure across multiple cloud and on-premises data centers.

How does Terraform work?

Terraform allows users to define their entire infrastructure simply by using configuration files and version control. When a command is given to deploy and run a server, database or load balancer, Terraform parses the code and translates it into an application programming interface (API) call to the resource provider. Because Terraform is open source, developers are always able to extend the tool's usefulness by writing new plugins or compiling different versions of existing plugins.

Terraform has two important components: **Terraform Core and Terraform Plugins.**
Terraform Core oversees the *reading and interpolation of resource plan executions, resource graphs, state management features and configuration files.*
Terraform Plugins are responsible for *defining resources for specific services.* This includes authenticating infrastructure providers and initializing the libraries used to make API calls.

The Terraform Registry acts as a centralized repository for module sharing and enables the discovery and distribution of Terraform modules to users. The Registry is available in two variants:

Public Registry -- holds services that interact with an API to expose and manage specific resource and community-contributed modules.
Private Registry -- holds services for modules used internally within an organization.

What is Terraform used for?

1. **External resource management** -- Terraform supports public and private cloud infrastructure, as well as network appliances and software as a service (SaaS) deployments.
2. **Multi-cloud deployment** -- the software tool's native ability to support multiple cloud services helps increase fault tolerance.
3. **Multi-tier applications** -- Terraform allows each resource collection to easily be scaled up or down as needed.
4. **Self-service clusters** -- the registries make it easy for users to find prepackaged configurations that can be used as is or modified to meet a particular need.
5. **Software defined networking (SDN)** -- Terraform's readability makes it easy for network engineers to codify the configuration for an SDN.
6. **Resource scheduler** -- Terraform modules can stop and start resources on AWS and allow Kubernetes to schedule Docker containers.
7. **Disposable environments** -- modules can be used to create an ad hoc, throwaway test environment for code before it's put into production.

Features of Terraform

Terraform works by building a graph database that provides operators with insight into resource dependencies. It also generates an execution plan that allows operators to see what sequence of steps Terraform will take when a setting is applied or a change is made.

The advantages of using Terraform include the ability to:

- translate HCL code into JSON;
- support multiple cloud platforms;
- make incremental changes to resources;
- provide support for software-defined networking;
- import existing resources to a Terraform state; and
- lock modules before applying state changes to ensure that only one person can make changes at a time

There are some disadvantages, however, to using Terraform.

- New releases and updates may have bugs.
- States have to be in sync with the infrastructure at all times.
- If users don't opt to use JSON, they will have to learn a new language, HCL.
- It doesn't have error handling.
- Renaming resources and moving them deeper into modules can be difficult.

Alternatives to Terraform

Two examples of alternative tools to **Terraform are Pulumi and SaltStack.**

Pulumi is an infrastructure-as-code upstart designed with a tool set specifically made to move users away from Terraform. Pulumi supports cloud-native platforms, like Kubernetes, and adds Terraform-like features such as CrossGuard.

SaltStack is an event-driven automation and IaC tool that helps IT organizations manage and secure cloud infrastructure. The tool can be used to automate the efficient orchestration of an enterprise DevOps workflow.

Kubernetes is an *open-source container orchestration platform that supports numerous containers to work together, thereby reducing operational load.* Features of this platform include rolling deployment, auto-scaling, volume storage, and computer resource. It can run in a data center, or in a public, private, or hybrid cloud.

benefits of Kubernetes with DevOps

Quick delivery of software with improved compliance Facilitates continual enhancement Enhances collaboration and transparency across the team(s) that are responsible for delivering the software. Effectively reduce development costs and security risks

What is Kubernetes Architecture?

Kubernetes is *an architecture that offers a loosely coupled mechanism for service discovery across a cluster. A Kubernetes cluster has one or more control planes, and one or more compute nodes*.

Overall, the control plane is responsible for managing the overall cluster, exposing the application program interface (API), and for scheduling the initiation and shutdown of compute nodes based on a desired configuration. Each of the compute nodes runs a container runtime like Docker along with an agent, kubelet, which communicates with the control plane. Each node can be bare metal servers, or on-premises or cloud-based virtual machines (VMs).

What are Kubernetes architecture components?

The main components of a Kubernetes cluster include:

5. **Nodes**: Nodes are VMs or physical servers that host containerized applications. Each node in a cluster can run one or more application instance. There can be as few as one node, however, a typical Kubernetes cluster will have several nodes (and deployments with hundreds or more nodes are not uncommon).
6. **Image Registry**: Container images are kept in the registry and transferred to nodes by the control plane for execution in container pods.
7. **Pods**: Pods are where containerized applications run. They can include one or more containers and are the smallest unit of deployment for applications in a Kubernetes cluster. .