WebMagic Informatica

Terraform Interview Questions

WebMagic Informatica
webmagicinformatica.com
+91 7021 4791 26

Q1. What do you understand by Terraform in AWS?

Terraform is a part of the AWS DevOps Competency and is also an AWS Partner Network (APN) advanced technology partner. It is similar to AWS Cloud Formation in the sense that it is also an "infrastructure as code" tool that allows you to create, update, and version your AWS infrastructure.

Q2. What are the key features of Terraform?

Terraform helps you manage all of your infrastructure as code and construct it as and when needed. Here are its key main features:

A console that allows users to observe functions The ability to translate HCL code into JSON format A configuration language that supports interpolation A module count that keeps track of the number of modules applied to the infrastructure.

Q3. Define IAC?

IAC or Infrastructure as Code allows you to build, change, and manage your infrastructure through coding instead of manual processes. The configuration files are created according to your infrastructure specifications and these configurations can be edited and distributed securely within an organisation.

Q5. What are the most useful Terraform commands?

Some of the most useful Terraform commands are:

- `terraform init` - initializes the current directory
- `terraform refresh` - refreshes the state file
- `terraform output` - views Terraform outputs
- `terraform apply` - applies the Terraform code and builds stuff
- `terraform destroy` - destroys what has been built by Terraform
- `terraform graph` - creates a DOT-formatted graph
- `terraform plan` - a dry run to see what Terraform will do

Q6. Are callbacks possible with Terraform on Azure?

By using the Azure Event Hubs, callbacks are possible on Azure. Terraform's Azure supplier provides effortless functionality to users. Microsoft Azure Cloud Shell provides an already-installed Terraform occurrence.

Q7. What is Terraform init?

Terraform init is a control to initialise an operational index that contains Terraform pattern files. This control can be looped multiple times. It is the first command that should be run after writing the new Terraform design.

Q8. What is Terraform D?

Terraform D is a plugin used on most in-service systems and Windows. Terraform init by default searches the following directories for plugins.

Q9. Is history the same as it is on the web while using TFS API to provide resources?

Yes, the narration is similar to on the web because the UI keeps the API as the base. The whole thing that is on the UI is available during other methods and the API.

Q10. Why is Terraform used for DevOps?

Terraform uses a JSON-like configuration language called the HashiCorp Configuration Language (HCL). HCL has a very simple syntax that makes it easy for DevOps teams to define and enforce infrastructure configurations across multiple clouds and on-premises data centres.

Define null resource in Terraform. null_resource implements the standard resource library, but no further action is taken. The triggers argument allows an arbitrary set of values that will cause the replacement of resources when changed.

Q11. What do you mean by Terraform cloud?

Terraform Cloud is a platform that enables teams to use Terraform together, either on-demand or in response to various events. It is deeply integrated with Terraform's workflows and data, unlike a general-purpose continuous integration system. It includes easy access to shared state and secret data, detailed policy controls for updating infrastructure and governing the contents of Terraform, a private registry for sharing Terraform modules, and lots more.

Q12. What do you understand by the terraform backend?

Each Terraform configuration can specify a backend, which defines two main things:

Where operations are performed Where the state is stored (Terraform keeps track of all the resources created in a state file)

Q13. What are the version controls supported by Terraform besides GitHub?

The version controls supported GitLab EE, GitLab CE, and Bucket cloud.

Q14. Name some major competitors of Terraform?

Some of the top competitors and alternatives to Terraform are Azure Management Tools, Morpheus, CloudHealth, Turbonomic, and CloudBolt.

Q15. Explain the uses of Terraform CLI and list some basic CLI commands.

The Terraform Command-Line Interface (CLI) is used to manage infrastructure and interact with Terraform state, configuration files, providers, etc.

Here are some basic CLI commands:

- `terraform init` - initializes the current directory
- `terraform refresh` - refreshes the state file
- `terraform output` - views Terraform outputs
- `terraform apply` - applies the Terraform code and builds stuff
- `terraform destroy` - destroys what has been built by Terraform
- `terraform graph` - creates a DOT-formatted graph
- `terraform plan` - a dry run to see what Terraform will do

Q16. What are modules in Terraform?

A jug for numerous resources that are used jointly is known as a module in Terraform. The root module includes resources mentioned in the .tf files and is required for every Terraform.

Q17. What is a Private Module Registry?

A Private Module Registry is a feature from Terraform Cloud that allows users to share Terraform modules across the organization. You can enforce rules or "sentinel policies" on the registry that specify how members of your organization can use the modules.

Q18. Is Terraform usable for an on-prem infrastructure?

Yes, Terraform can be used for on-prem infrastructure. As there are a lot of obtainable providers, we can decide which suits us the best. All that we need is an API.

Q19. Does Terraform support multi-provider deployments?

Yes, multi-provider deployments are supported by Terraform, which includes on-prem like Openstack, VMware, and we can manage SDN even using Terram too.

Q20. How is duplicate resource error ignored during terraform apply?

We can try the following options:

Delete those resources from the cloud provider(API) and recreate them using Terraform Delete those resources from Terraform code to stop its management with it

Carry out a terraform import of the resource and remove the code that is trying to recreate them

Name all version controls supported by Terraform The supported version controls are:

Azure DevOps Services Azure DevOps Server Bitbucket Server Bitbucket Cloud Gitlab EE and CE Gitlab.com GitHub Enterprise GitHub.com (OAuth) GitHub.com

Q20. What are some of the built-in provisioners available in Terraform?

Here is the list of built-in provisioners in Terraform:

Salt-masterless Provisioner Remote-exec Provisioner Puppet Provisioner Local-exec Provisioner Habitat Provisioner File Provisioner Chef Provisioner

Q21. Which command destroys Terraform managed infrastructure?

The given command is used for this purpose:

terraform destroy [options] [dir]

Q22. Tell us about some notable Terraform applications.

The applications of Terraform are pretty broad due to its facility of extending its abilities for resource manipulation. Some of the unique applications are:

Software demos development Resource schedulers Multi-cloud deployment Disposable environment creations Multi-tier applications development Self-service clusters Setup of Heroku App

Q23. What are the components of Terraform architecture?

The Terraform architecture includes the following features:

Sub-graphs Expression Evaluation Vertex Evaluation Graph Walk Graph Builder State Manager Configuration Loader CLI (Command Line interface) Backend

Q24. Define Resource Graph in Terraform.

A resource graph is a visual representation of the resources. It helps modify and create independent resources simultaneously. Terraform establishes a plan for the configuration of the graph to generate plans and refresh the state. It creates structure most efficiently and effectively to help us understand the drawbacks.

Q25. Can you provide a few examples where we can use for Sentinel policies?

Sentinels are a powerful way to implement a variety of policies in Terraform. Here are a few examples:

Enforce explicit ownership in resources Restrict roles the cloud provider can assume Review an audit trail for Terraform Cloud operations Forbid only certain resources, providers, or data sources Enforce mandatory tagging on resources Restrict how modules are used in the Private Module Registry

Q26. What are the various levels of Sentinel enforcement?

Sentinel has three enforcement levels - advisory, soft mandatory, and hard mandatory.

Advisory - Logged but allowed to pass. An advisory is issued to the user when they trigger a plan that violates the policy. Soft Mandatory - The policy must pass unless an override is specified. Only administrators have the ability to override. Hard Mandatory - The policy must pass no matter what. This policy cannot be overridden unless it is removed. It is the default enforcement level in Terraform.

Q27. How to Store Sensitive Data in Terraform?

Terraform requires credentials to communicate with your cloud provider's API. But most of the time, these credentials are saved in plaintext on your desktop. GitHub is exposed to thousands of API and cryptographic keys every day. Hence, your API keys should never be stored in Terraform code directly. You should use encrypted storage to store all your passwords, TLS certificates, SSH keys, and anything else that shouldn't be stored in plain text.

Q28. What is Terragrunt, and what are its uses?

Terragrunt is a thin wrapper that provides extra tools to keep configurations DRY, manage remote state and work with multiple Terraform modules. It is used for:

Working with multiple AWS accounts Executing Terraform commands on multiple modules Keeping our CLI flags DRY Keeping our remote state configuration DRY Keeping our Terraform code DRY

Q29. Explain State File Locking?

State file locking is a Terraform mechanism in which operations on a specific state file are blocked to avoid conflicts between multiple users performing the same process. When one user releases the lock, then only the other one can operate on that state. This helps in preventing state file corruption. This is a backend operation.

Q30. What do you understand by a Tainted Resource?

A tainted resource is a resource that is forced to be destroyed and recreated on the next command. When a resource is marked as tainted, the state files are updated, but nothing changes on the infrastructure. The terraform plan shows that help will be destroyed and recreated. The changes get implemented when the next apply happens.

Q31. How to lock Terraform module versions?

A proven way of locking Terraform module versions is using the Terraform module registry as a source. We can use the 'version' attribute in the module of the Terraform configuration file. As the Github repository is being used as a source, we need to specify versions, branch, and query strings with '?ref'.

Q32. What is Terraform Core?

Tell us some of its primary responsibilities. Terraform Core is a binary written statically compiled by using the Go programming language. The compiled binary offers an entry point for the users of Terraform. The primary responsibilities include:

Reading and interpolation of modules and configuration files by Infrastructure as code functionalities Resource Graph Construction Plugin communication through RPC Plan execution Management of resource state

Q33. Give the terraform configuration for creating a single EC2 instance on AWS.

This is the Terraform configuration for creating a single EC2 instance on AWS:

```
provider “aws” {
region = “”
}
resource “aws_instance” “example” {
ami = "ami-213123585"
instance_type = "t2.micro"
tags {
Name = "example"
}
}
```

Q33. How will you upgrade plugins on Terraform?

Run 'terraform init' with '-upgrade' option. This command rechecks the releases.hashicorp.com to find new acceptable provider versions. It also downloads available provider versions. ".terraform/plugins/_" is the automatic downloads directory.

Q34. How will you make an object of one module available for the other module at a high level?

An output variable is defined in resource configuration. Declare the output variable of module_A. Create a file variable.tf for module B. Establish the input variable inside this file having the same name as the key defined in module_B. Replicate the process for making variables available to other modules

Q35. What are some of the latest Terraform Azure Provider factors?

The latest versions involve new data resources and Azurem_batch_certificate, which helps in managing the certificate. This resource is used for controlling the prefix in networking. There is fixing of bugs, and azurerm_app_service has also been enhanced.

Q36. How will you control and handle rollbacks when something goes wrong?

I need to recommit the previous code version to be the new and current version in my VCS. This would trigger a terraform run, which would be responsible for running the old code. As Terraform is more declarative, I will make sure all things in the code roll back to the old code. I would use the State Rollback Feature of Terraform Enterprise to roll back to the latest state if the state file got corrupted.

Terraform is an open-source infrastructure as code software tool created by HashiCorp. Users define and provide data center infrastructure using a declarative configuration language known as HashiCorp Configuration Language, or optionally JSON.

More Interview Question: https://webmagicinformatica.com/interview-questions/