**Terraform Interview Questions & Answers**

1. **What do you mean by Terraform?**

Terraform is open-source communication as a system software tool created by HashiCorp It is an instrument for building, altering and versioning transportation safely and professionally. Terraform can direct existing and accepted service providers as well as convention in-house solutions.

1. **Name the components of Terraform?**

The logical separation of Terraform into separate structures refers to two separate components. The two components are the Terraform Core and Terraform Plugins. The Terraform Core utilizes distant process calls for communicating with Terraform Plugins. Also, Terraform Core offers varied ways of discovering and loading plugins according to supplies. The Terraform Plugins symbolize a completion for a specific service such as bash or AWS.

1. **How does Terraform help in discovering plugins?**

The authority “Terraform init” helps Terraform interpret configuration files in the operational directory. Then, Terraform finds out the essential plugins and searches for installed plugins in diverse locations. In addition, Terraform also downloads extra plugins at times. Then, it decides the plugin versions for using and writes a security device file for ensuring that Terraform will employ the identical plugin versions.

1. **Can I add policies to the open-source or pro version of Terraform enterprise?**

You cannot insert policies to the open-source description of Terraform Enterprise. The equal also goes for the Enterprise Pro version. The finest version of Terraform Enterprise only could contact the lookout policies.

1. **Do you think callbacks possible with Terraform on Azure?**

Callbacks are probable on Azure with Terraform by using the Azure Event Hubs. Azure supplier of Terraform provides this functionality effortlessly to users. Most imperative of all, Azure Cloud Shell of Microsoft provides an already installed occurrence of Terraform.

1. **What are the versions controls supported on Terraform?**

GitHub is the essential version manages supported on Terraform. Also, you can find the hold up of GitLab CE and GitLab EE on Terraform. In addition, Terraform also ropes the Bucket Cloud.

1. **Define Modules in Terraform?**

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

1. **Name some major competitors of Terraform?**

Some of them are:

* Packer
* Cloud Foundry
* Ansible
* Kubernetes

1. **What provisioners are used in Terraform?**

Provisioners in Terraform can be used to arrange servers or other transportation objects for repair. It models exact actions on the restricted machine as well as on distant machines.

1. **Define Terraform init?**

The Terraform init is a control that is used to initialize an operational index containing Terraform pattern files. This is the first authority that should be sprint after writing a new Terraform design or cloning an obtainable one from account control. It is safe to lope this control multiple times.

1. **What do you mean by Terragrunt?**

Terragrunt is a thin covering for Terraform that implements the practices advocated by the Terraform: Up and Running book. We've established Terragrunt obliging as it encourages versioned modules and reusability for diverse environments with some useful features, counting recursive code implementation in subdirectories.

1. **Explain the uses of Terraform CLI?**

Terraform is forbidden via a very simple to employ the command-line interface; it is only a solitary command-line application: Terraform. This application then takes a subcommand such as "relate" or "plan". The total list of subcommands is in the steering to the left.

1. **What do you mean by Terraform cloud?**

Terraform Cloud is a SaaS that we hold up that in its place when you run Terraform you still might run it on your restricted machine, but now it saves and retrieves the condition file from Terraform Cloud which is operation over here. Terraform Cloud removes a lot of of the complexities in difficult to preserve your own Terraform state files in a multi-team.

1. **Define null resource in Terraform?**

The null resource implements the average resource lifecycle but takes no extra action. The trigger argument permits specifying a subjective set of values that, when misrepresented will source the reserve to be replaced.

1. **What do you mean by Terraform D?**

Terraform. D/plugins on most in-service systems and %APPDATA%\Terraform. d\plugins on Windows. By default, Terraform init searches the next directories for plugins.

1. **Define Terraform provider?**

Terraform is used to manage and inform infrastructure resources such as bodily machines, VMs, network switches, containers, and more. A provider is accountable for thoughtful API interactions and revealing resources.

1. **How do you use count Terraform?**

We just employ [count. index] to energetically identify the current iteration of the script.

1. **How to check the installed version of Terraform?**

To test the version, commence Windows Power Shell and go through Terraform -version.

1. **How to Terraform work?**

Terraform creates an implementation plan define, what it will do to attain the preferred state, and then executes it to construct the described infrastructure. As the configuration changes, Terraform is talented to decide what changed and generate incremental execution plans which can be practical.

1. **What is Terraform in AWS?**

Terraform by HashiCorp, an AWS Partner Network Advanced Technology Partner and associate of the AWS DevOps capability is an "infrastructure as code" tool comparable to AWS Cloud Formation that permits you to produce an update, and story your Amazon Web Services infrastructure.

1. **Name some major features of Terraform?**

Some of them are:

* Execution Plans
* Change Automation
* Resource Graph
* Infrastructure as code

1. **Tell me the reasons to choose Terraform for DevOps?**

To decide to Terraform for DevOps one significant reason people think Terraform is to direct their infrastructure as code. Infrastructure as code is also a primary and base for DevOps practices such as account control, policy review, continuous addition and continuous operation.

1. **Can Terraform be used for on-prem infrastructure?**

Yes, Terraform can be utilized for on-prem infrastructure. There are a lot of obtainable providers. You can decide any one of them which suits you most excellent. Many also build client Terraform providers for themselves; all wanted is just an API.

1. **Define LAC?**

LaC is a short form to the term “Infrastructure as Code”. IaC refers to a scheme whereby developers can run and provision the computer data centre’s mechanically instead of getting into a physical process. Terraform, for example, is a case tool of IaC.

1. **While using TEF’s API for the provision of resources, will history be the same as it is on the web?**

Yes. The narration will be similar as it is on the web because the UI is residential keeping API as the base. So the whole thing that is on the UI could be availed during the API and the other method around.

So, above are the mentioned interview questions & answers for [DevOps jobs](https://www.shine.com/job-search/devops-jobs), candidates should go through it and search more to clear their job interview.

--------------------------------------------------------------------------------------------------------------------------------------

### 1. Define Terraform.

Answer: The first entry among Terraform interview questions always deals with the definition of Terraform. Terraform is a tool for creating, changing, and versioning infrastructure with higher safety and efficiency. Terraform is now popular all over the world as an important addition to the chain of important DevOps tools. Terraform provides essential functionalities of managing solutions for in-house issues. The facility of ‘infrastructure as a code’ model in Terraform is also one of the popular reasons for its adoption.

### 2. What are the reasons to choose Terraform for DevOps?

Answer: Candidates would definitely encounter this mention among top Terraform interview questions. The foremost reason to choose Terraform for DevOps is evident in the improvement of quality and efficiency in software delivery. Terraform supports automation and helps in running infrastructure as code. Another potential reason for choosing Terraform is the facility for implementing almost any type of coding principle.

Also Read: [Top 30 Kubernetes Interview Questions](https://www.whizlabs.com/blog/top-kubernetes-interview-questions/)

### 3. What are the notable features of Terraform?

Answer: The features of Terraform would be one of the common topics of the latest Terraform interview questions. The key features of Terraform are as follows.

1. In-built graphing features for visualization of infrastructure
2. Friendly custom syntax helps in improving efficiency
3. The ability for understanding resource relationships
4. Contribution of updates and new features by the Open Source Project
5. Capability for breaking down a configuration into smaller parts for ease of organization and maintenance

### 4. How does Terraform work?

Answer: The working of Terraform is a formidable topic that churns out many relevant Terraform interview questions. The best answer to this question would be to point towards the plugin-based architecture of Terraform. The plugin-based architecture helps developers in extending functionalities of Terraform. Developers could write new plugins or compile the modified versions of current plugins.

### 5. What are the notable applications of Terraform?

Answer: The use cases of Terraform are also important aspects of the best Terraform interview questions. Generally, the applications of Terraform are very broad due to the facility for extending the abilities of Terraform for resource manipulation. Here are some of the notable applications of Terraform.

* Heroku App setup
* Self-service clusters
* Development of multi-tier applications
* Creation of disposable environments
* Multi-cloud deployment
* Resource schedulers
* Developing software demos

**Ansible and Terraform are the two names that are prominent in the DevOps landscape now. Let’s understand the**[**Ansible vs Terraform**](https://www.whizlabs.com/blog/ansible-vs-terraform/)**battle!**

### 6. What is the process for making the object of one module available for another module at a higher level?

Answer: This entry can be one of the difficult Terraform interview questions. Here is the process to make an object of one module available for another module at a higher level.

Define an output variable in a resource configuration.

Declare the output variable of module\_1 for use in another module’s configuration

Create a new key name and ensure that it is equivalent to the output variable of module\_1

Now, create a file ‘variable.tf’ for module\_2

Set up an input variable inside the ‘variable.tf’ the file that would enable dynamic configuration of the resource in a module

In order to ensure the availability of the variable to another module, repeat the process again. The reason for this is the restricted scope of the particular variable to module\_2.

### 7. Do you know about the new factors in the latest v1.24.0 and v1.25.0 Terraform Azure Provider?

Answer: Candidates should prepare for such Terraform technical interview questions. You can find multiple new resources in the latest versions of Terraform alongside new data resources. For example, Azurerm\_batch\_certificate. The new resource can support the management of certificates in the Azure batch. Furthermore, it also helps in the management of public IP and the prefix in networking. Another data resource in the latest versions is the Azurerm\_firewall that helps in accessing data for a particular firewall existing already. In addition, the new versions also involve a lot of bug fixes. You can also find improvement in the azurerm\_app\_service resource in the latest versions.

### 8. Does Terraform support themes?

Answer: Candidates could land up with this entry among Terraform technical interview questions generally. The answer implies that the v0.3.1 of Terraform supports Gtk themes. We can use the command “**cp/usr/wherever/THEMENAME/gtk/gtkrc $HOME/.gtkrc**” for enabling gtk theme in a system. If the command shows error in opening the theme files or in the event of failure of the files to open, edits in the .gtkrc are mandatory. After editing, you have to attach the line “**pixmap\_path/usr/wherever/THEMENAME/gtk**” at the start of the file name. Now, the theme could load at startup.

### 9. What are the components of Terraform?

Answer: The structure of Terraform is another notable point for the best Terraform interview questions. The logical division of Terraform into distinct structures refers to two distinct components. The two components are the Terraform Core and Terraform Plugins. The Terraform Core utilizes remote procedure calls (RPCs) for communicating with Terraform Plugins. In addition, Terraform Core also offers diverse ways of discovering and loading plugins according to requirements. The Terraform Plugins represent an implementation for a specific service such as bash or AWS or provisioner.

### 10. What are the primary responsibilities of Terraform Core?

Answer: This is one of the basic Terraform interview questions that you can face. The Terraform Core is a statically-compiled binary written by using the Go programming language. The compiled binary offers an entry-point for Terraform users. The primary responsibilities of the Terraform Core are as follows.

* Resource state management
* Execution of plans
* Communication with plugins through RPC
* Construction of Resource Graph
* Infrastructure as code functionalities for reading and interpolation of configuration files and modules

Check: [Most Common Jenkins Interview Questions & Answers](https://www.whizlabs.com/blog/top-jenkins-interview-questions/)

### 11. What is the Terraform Plugins?

Answer: Candidates should prepare for Terraform interview questions based on this topic. Terraform Plugins are executable binaries written in Go programming language. Plugins are basically the providers and provisioners in Terraform configurations. Terraform has various in-built provisioner plugins, and users have to discover provider plugins dynamically according to their requirements. The Terraform plugins help in domain-specific implementation of the service they represent.

### 12. What are the primary responsibilities of the provider and provisioner plugins?

Answer: Candidates could find this entry as a follow up to Terraform interview questions on architecture or the Terraform plugins. The primary responsibilities for provider plugins are as follows.

* Authentication with infrastructure provider
* Definition of resources that map to particular services
* Initialization of libraries used for making API calls

The primary responsibility of provisioner plugins is the execution of commands or scripts on a specific resource after creation or upon its destruction.

### 13. How does Terraform help in discovering plugins?

Answer: This entry is one of the most popular Terraform interview questions. The command “**terraform init**” helps Terraform read configuration files in the working directory. Then, Terraform finds out the necessary plugins and searches for installed plugins in different locations. In addition, Terraform also downloads additional plugins at times. Then, it decides the plugin versions for using and writes a lock file for ensuring that Terraform will use the same plugin versions.

### 14. What are the different behaviors of Terraform plugins during discovery?

Answer: This question is one of the tricky Terraform interview questions that can confuse many expert candidates. The behavior of the plugins depends on their type. The three kinds of plugins are built-in provisioners, providers by HashiCorp, and third-party providers and provisioners. The in-built provisioner plugins are always available in the Terraform binary. The providers by HashiCorp download automatically if not installed already. Regarding the third-party providers and provisioners, you have to install them manually.

### 15. What is the Terraform configuration for creating a single EC2 instance on AWS?

Answer: Candidates could land up with this interesting entry among Terraform DevOps interview questions. The following Terraform configuration helps in creating a single EC2 instance on AWS.

provider "aws" {

region = "ap-south-1"

}

resource "aws\_instance"

"example" {

ami = "ami-4fc58420"

instance\_type = "t2.micro"

tags {

 Name = "terraform-example"

}

}

It is required to be fully prepared before going for a DevOps interview. Prepare with these [Top DevOps Interview Questions](https://www.whizlabs.com/blog/top-devops-interview-questions/) to ace the interview!

### 16. Why does POVRay render fields and does not display sometimes?

Answer: This is one of the critical Terraform DevOps interview questions. The primary reason for the failure of default export to POVRay could be the lower version of POVRay. Version 3.0 of POVRay does not support the display. The same reason could be evident in the case of failure in a display of POVRay without error reports. Terraform works effectively with the 3.1 version of POVRay. So, you should use the –pov30 switch for informing Terraform about the issue. You can check the version of POVRay by typing “**POVRay**” and observing the first line of the output.

### 17. How can I check if the POVRay install is compatible with Terraform?

Answer: Candidates could find tough Terraform interview questions like this one. You can try “**povray+l tf\_land.pov**” to check whether the POVRay installs on your system is ok with Terraform. You can find two outcomes – one good and one bad. The good message is an error “**tf\_land.pov:26:error: Error opening TGA image**”. The “**tf\_land.pov**” denotes the file distributed in the root directory of terraforming. It means that you can find the included file in the POV on your system. The second message is about ‘**colors.inc**,’ that indicates the absence of files in your POV. As a result, you can clearly find out whether the POVRay installs works with Terraform or not.

### 18. What if I encounter a serious error and want to rollback?

Answer: Candidates should find this entry among practical Terraform interview questions. The answer is recommitting the previous version of the code for making it the new and current version in a VCS (Version Control System). As a result, a terraform run triggers and runs the old code. It is essential to ensure that the old code contains all entities provisioned in the code for rollback. If the state file has been subject to corruption from a recent Terraform run, then you can opt for State Rollback Feature in Terraform Enterprise. It can help you to roll back to the previous latest state. The validation for this process is the versioning of every state change.

### 19. Can I add policies to the open-source or Pro version of Terraform Enterprise?

Answer: This is one of the most popular Terraform interview questions coming up in recent interviews. First of all, you cannot add policies to the open-source version of Terraform Enterprise. The same also goes for the Enterprise Pro version. The Premium version of Terraform Enterprise only could access the sentinel policies.

### 20. What are the ways to lock Terraform module versions?

Answer: Candidates could find such technical Terraform interview questions difficult. Preparing for such questions in advance gives candidates a definite advantage. The answer is that there is a proven way of locking Terraform module versions. Using the Terraform module registry as a source, you can use the ‘**version**’ attribute in the module in the Terraform configuration file. Using a GitHub repository as a source, you have to specify branch, versions, and query string with ‘?ref’.

### 21. Are callbacks possible with Terraform on Azure?

Answer: This is also one of the new Terraform interview questions for aspiring candidates. Callbacks are possible on Azure with Terraform by using the Azure Event Hubs. AzureRM provider of Terraform provides this functionality easily to users. Most important of all, Azure Cloud Shell of Microsoft provides an already installed instance of Terraform.

### 22. Can I use Terraform for on-premises infrastructure?

Answer: Candidates could find this question as one of the tough ones based on real experience. The answer to this question clearly implies the feasibility of using Terraform with on-premises infrastructure. Many providers offer this functionality, and you can choose one according to your requirements. Certain providers also have APIs for accessing Terraform in on-premises infrastructure.

### 23. What are the version controls supported on Terraform?

Answer: The candidate could find this entry as one of the important Terraform interview questions. Although this question may seem to be the simplest of the lot, it is essential to point out the precise responses. First of all, GitHub is the basic version control supported on Terraform. In addition, you can also find the support of GitLab CE and GitLab EE on Terraform. Furthermore, Terraform also supports the Bucket Cloud.

### 24. Is there any similarity between the management of Azure Availability Zones and management by other available cloud providers?

Answer: The response to this question would directly refer to the fact that availability zones have different areas and zones. Every availability zone has a specific power source and network. Any region with an enabled availability zone would have three different availability zones. You need to note that AzureRM Terraform provider does not have any accessible resources for the management of Azure Availability Zones. However, this is the present scenario, and AzureRM Terraform provider could include some improvements in the future for this issue.

### 25. How can I upgrade plugins on Terraform?

Answer: The answer would start off with running ‘terraform init’ with the ‘-upgrade’ option. The command helps in rechecking the releases.hashicorp.com to find out new acceptable provider versions. The command also downloads the available provider versions. Such types of actions are evident in the case of providers which have their acceptable versions in the automatic downloads directory. The automatic downloads directory is “**.terraform/plugins/<OS>\_<ARCH>.**”

In case of installation of any acceptable version of a specific provider in another location, the ‘**terraform init -upgrade**’ command will not download a new version.

--------------------------------------------------------------------------------------------------------------------------

1.**What is Terraform?**

Terraform is an open-source infrastructure as a code software tool created by HashiCorp It is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.

2.**How to Terraform work?**

Terraform produce an execution plan delineate, what it will do to reach the desired state, and then executes it to build the described infrastructure. As the configuration changes, Terraform is able to determine what changed and create incremental execution plans which can be applied.

3.**Explain the uses of Terraform CLI?**

Terraform is controlled via a very easy to use the command-line interface (CLI) it is only a single command-line application: terraform. This application then takes a subcommand such as “apply” or “plan”. The complete list of subcommands is in the navigation to the left.

4.**what are the features of Terraform?**  
Some features of Terraform are:-

1. **Graphing –** Its features of graphing that are built-in are helpful in visualizing the infrastructure.
2. **Custom Syntax –** Its custom syntax is very friendly which aids in enhancing efficiency.
3. **Resource Relationships –** A very beneficial feature of terraforming is that it is able to understand resource relationships.
4. **Updates –** The updates and features are added by the Open Source Project. It does so with a group of lots of contributors.
5. **Improved Maintenance –** It is capable of breaking down the configuration into small parts or chunks for improving the organization and the maintenance.

5.**What are the components of Terraform?**

Terraform is another notable point for the best Terraform interview questions. The logical division of Terraform into distinct structures refers to two distinct components.

The two components are**the Terraform Core** and **Terraform Plugins**.

1. The Terraform Core utilizes remote procedure calls (RPCs) for communicating with Terraform Plugins. In addition, Terraform Core also offers diverse ways of discovering and loading plugins according to requirements.
2. The Terraform Plugins represent an implementation for a specific service such as bash or AWS or provisioner.

6.**What are the reasons to choose Terraform for DevOps?**

To choose to Terraform for DevOps one important reason people consider Terraform is to manage their infrastructure as code. Infrastructure as code is also a key and foundation for DevOps practices such as version control, code review, continuous integration, and continuous deployment.

7.**Define Modules in Terraform.**

Modules in terraforming is a container for multiple resources that are used together. Every Terraform configuration has at least one module, known as its root module, which consists of the resources defined in the . tf files in the main working directory.

8.**what is the Terraform cloud?**

Terraform Cloud is a SaaS that we support—that instead when you run Terraform you still could run it on your local machine, but now it saves and retrieves the state file from Terraform Cloud—which is running over here. Terraform Cloud removes many of the complexities in trying to maintain your own Terraform state files in a multi-team.

9.**How to check the installed version of Terraform?**

We can useterraform -versioncommand to identify the version which we are running.

10.**Explain the Provisioners in Terraform.**

Provisioners are used to model specific actions on the local machine or on a remote machine in order to prepare servers or other infrastructure objects for service.

11.**What is the difference between Terraform and CloudFormation?**

CloudFormation covers almost all bits and parts of AWS. Terraform covers the most important AWS resources as well. But on top of that Terraform can provision infrastructure at other cloud providers as well as 3rd party services

12.**What is the null resource in Terraform?**

The null\_resource resource implements the standard resource lifecycle but takes no further action. The triggers argument allows specifying an arbitrary set of values that, when changed, will cause the resource to be replaced.

13.**How do you use count terraform?**

we just use **[count. index]** to dynamically specify the current iteration of the script!

14.**What is a terraform provider?**

Terraform is used to create, manage, and update infrastructure resources such as physical machines, VMs, network switches, containers, and more. A provider is responsible for understanding API interactions and exposing resources.

15.**List Some notable applications of Terraform.**  
List some of the notable applications of Terraform.

1. Heroku App setup
2. Self-service clusters
3. Development of multi-tier applications
4. Creation of disposable environments
5. Multi-cloud deployment
6. Resource schedulers
7. Developing software demos

16.**Where is terraform D?**

Terraform. d/plugins on most operating systems and %APPDATA%\terraform. d\plugins on Windows. By default, terraform init searches the following directories for plugins.

17. **What if I encounter a serious error and want to rollback?**

Recommitting the previous version of the code for making it the new and current version in a VCS (Version Control System). As a result, a terraform run triggers and runs the old code. It is essential to ensure that the old code contains all entities provisioned in the code for rollback. If the state file has been subject to corruption from a recent Terraform run, then you can opt for State Rollback Feature in Terraform Enterprise. It can help you to roll back to the previous latest state. The validation for this process is the versioning of every state change.

18. **Can I use Terraform for on-premises infrastructure?**

Yes, Terraform supports **multi**–**provider** deployments which include on-prem like VMware, Openstack and even using Terrarm we can manage SDN (Software Defined Network) too.

19. **What is terraform init?**

The **terraform init** is a command that is used to initialize a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It is safe to run this command multiple times.

**Syntax**

terraform init [options] [DIR]

20. **What is terraform state?**

**Terraform state** is a command that is used for advanced state management. The state is a necessary requirement for Terraform to function. Terraform must store state about your managed infrastructure and configuration. This state is used by Terraform to map real-world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures. Syntax

terraform state [options] [args]

21. **What is the use of terraform apply command?**

The **terraform apply command** is **used** to **apply** the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a **terraform** plan execution plan.

22. **Which command is used to destroy Terraform-managed infrastructure?**

The “**terraform destroy command”**is used to destroy the Terraform-managed infrastructure

23. **What is the use of fmt command in Terraform?**

**fmt**tool will take care of formatting. TO validate our configuration formatting and make them neat by running

terraform fmt -diff

The formatting **command** rewrites **Terraform** configuration files in a canonical format and style.

24. **How to ignore duplicate resource error during terraform apply?**

It is a little different situation,

Possible causes of this could be:

* Someone has executed Terraform code and we don’t have a shared/updated state
* someone has created them manually
* a Terraform destroy failed in a way that deleted the resources for the API but failed to save the update state

The solution depends on what we need as the desired state. we can try with:

* delete those resources from our Terraform code to stop managing them with it
* delete those resources from the API ( cloud provider ) and recreate them with Terraform
* Perform a terraform import of those resources and remove the terraform code that is trying to recreate them (NOT RECOMMENDED)
* use terraform apply --target=xxx to apply only resources you need to apply (NOT RECOMMENDED)

25. **Share a few Terraform CLI Commands which you use only your day to day routine?**

Few common Terraform commands which I use based on need like:

* **Apply:** builds or changes infrastructure.
* **Console:** Interactive console for Terraform interpolations.
* **destroy:** Destroy Terraform-managed infrastructure.
* **env:** Workspace management
* **fmt:** Rewrites config files to canonical format
* **get:** Download and install modules for the configuration
* **graph:** Build a visible graph of Terraform resources Import: existing infrastructure into Terraform
* **Init:** Initialize a Terraform working directory
* **output:** Read output from a state file plan: Generate and show an execution plan validate: Validates the Terraform files
* **version:** Prints the Terraform version
* **Workspace:** Workspace management

26. **In which programming language Terraform is written?**

It was written using “Go programming language.”

27. **Is there a way to bulk import the state of current cloud subscription into Terraform state?**

We can use the terraform import command to import individual resources into our Terraform state, but there is not currently a bulk import tool.

28. **Can you shortly explain the best practices for cloning an infrastructure with Terraform?**

If we are referring to the promotion, then we could have dev and prod branches of a repository with Terraform code and then point dev and prod workspaces against those branches and promote changes from dev to prod to clone, but set different values of Terraform variables to account for differences between dev and prod.

If by “clone” you mean bring existing infrastructure not provisioned by Terraform under Terraform management, then we can use the terraform import command to import individual resources into our Terraform state.

or If we just want to duplicate an existing infrastructure with Terraform, then we would need to map all of our existing resources into a Terraform template.

29. **Give a configuration of for creating a single E2C instance in Amazon Web Services ( AWS ).**

provider “aws” { region = “ap-south-1” } resource “aws\_instance” “example” { ami = “ami-4fc58420” instance\_type = “t2.micro” tags { Name = “terraform-example” } }

30. **What are all version controls are supported by Terraform?**

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

------------------------------------------------------------------------------------------------------------------------

1. **What is Terraform?**

Terraform is an open-source infrastructure as a code software tool created by HashiCorp It is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.

2.**How to Terraform work?**

Terraform produce an execution plan delineate, what it will do to reach the desired state, and then executes it to build the described infrastructure. As the configuration changes, Terraform is able to determine what changed and create incremental execution plans which can be applied.

3.**Explain the uses of Terraform CLI?**

Terraform is controlled via a very easy to use the command-line interface (CLI) it is only a single command-line application: terraform. This application then takes a subcommand such as "apply" or "plan". The complete list of subcommands is in the navigation to the left.

4.**what are the features of Terraform?**  
Some features of Terraform are:-

1. **Graphing -** Its features of graphing that are built-in are helpful in visualizing the infrastructure.
2. **Custom Syntax -** Its custom syntax is very friendly which aids in enhancing efficiency.
3. **Resource Relationships -** A very beneficial feature of terraforming is that it is able to understand resource relationships.
4. **Updates -** The updates and features are added by the Open Source Project. It does so with a group of lots of contributors.
5. **Improved Maintenance -** It is capable of breaking down the configuration into small parts or chunks for improving the organization and the maintenance.

5.**What are the components of Terraform?**

Terraform is another notable point for the best Terraform interview questions. The logical division of Terraform into distinct structures refers to two distinct components.

The two components are**the Terraform Core** and **Terraform Plugins**.

1. The Terraform Core utilizes remote procedure calls (RPCs) for communicating with Terraform Plugins. In addition, Terraform Core also offers diverse ways of discovering and loading plugins according to requirements.
2. The Terraform Plugins represent an implementation for a specific service such as bash or AWS or provisioner.

6.**What are the reasons to choose Terraform for DevOps?**

To choose to Terraform for DevOps one important reason people consider Terraform is to manage their infrastructure as code. Infrastructure as code is also a key and foundation for DevOps practices such as version control, code review, continuous integration, and continuous deployment.

7.**Define Modules in Terraform.**

Modules in terraforming is a container for multiple resources that are used together. Every Terraform configuration has at least one module, known as its root module, which consists of the resources defined in the . tf files in the main working directory.

8.**what is the Terraform cloud?**

Terraform Cloud is a SaaS that we support—that instead when you run Terraform you still could run it on your local machine, but now it saves and retrieves the state file from Terraform Cloud—which is running over here. Terraform Cloud removes many of the complexities in trying to maintain your own Terraform state files in a multi-team.

9.**How to check the installed version of Terraform?**

To check the version, launch Windows PowerShell and enter: terraform -version.

10.**Explain the Provisioners in Terraform.**

Provisioners are used to model specific actions on the local machine or on a remote machine in order to prepare servers or other infrastructure objects for service.

11.**What is the difference between Terraform and CloudFormation?**

CloudFormation covers almost all bits and parts of AWS. Terraform covers the most important AWS resources as well. But on top of that Terraform can provision infrastructure at other cloud providers as well as 3rd party services

12.**What is the null resource in Terraform?**

The null\_resource resource implements the standard resource lifecycle but takes no further action. The triggers argument allows specifying an arbitrary set of values that, when changed, will cause the resource to be replaced.

13.**How do you use count terraform?**

we just use **[count. index]** to dynamically specify the current iteration of the script!

14.**What is a terraform provider?**

Terraform is used to create, manage, and update infrastructure resources such as physical machines, VMs, network switches, containers, and more. A provider is responsible for understanding API interactions and exposing resources.

15.**List Some notable applications of Terraform.**  
List some of the notable applications of Terraform.

1. Heroku App setup
2. Self-service clusters
3. Development of multi-tier applications
4. Creation of disposable environments
5. Multi-cloud deployment
6. Resource schedulers
7. Developing software demos

16.**Where is terraform D?**

Terraform. d/plugins on most operating systems and %APPDATA%\terraform. d\plugins on Windows. By default, terraform init searches the following directories for plugins.

--------------------------------------------------------------------------------------------------------------------

### ****Q1. What are the reasons for choosing Terraform for DevOps?****

Terraform is a tool for Infrastructure as Code (IaC). It is used for defining and provisioning the complete infrastructure. Here are some of the most important reasons why I would choose Terraform:

* It can do complete orchestration and not just configuration management (like Ansible and Puppet).
* Has an amazing support of almost all the popular cloud providers like AWS, Azure, GCP, DigitalOcean etc.
* Easily manages the configuration of an immutable (dynamic) infrastructure.
* Provide immutable infrastructure where configuration changes smoothly.
* Works on HCL (HashiCorp configuration language), which is very easy to learn and understand.
* Easily portable from one provider to another.
* Works on masterless, Client only architecture. You just need to install Terraform client and it will take care of the rest using the APIs.

### ****Q2. How does Terraform work?****

There are four stages in a Terraform lifecycle – terraform init, terraform plan, terraform apply, and terraform destroy. So, the flow is first you initialize, then you plan, after that you apply and finally destroy.

* Terraform init is used to initialize the working directory which has all the terraform configuration files.
* Terraform plan is used to create an execution in order to reach a desired state of the infrastructure. This is created to check whether the expectation of reaching the desired state will be met or not, without changing any real state of resources.
* Terraform apply basically executes the execution plan created by terraform plan to reach the desired state of the infrastructure.
* Terraform destroy is used to remove all the resources on the infrastructure and destroy it.

### ****Q3. Explain the working of Terraform Core.****

Terraform core is the entry point of the whole terraform architecture. It is responsible for reading all the configurations and create a dependency graph out of it. Once the terraform plan command is executed, the terraform core loads all the needed configuration files from the disk and also the last known state of the resources. It then begins a refresh operation and tells the terraform provider plugin to read all the resources. After the read operation, the terraform core checks if there is any difference in the last known state and current state. It then presents the changes in the output of terraform plan on the terminal.

Terraform configurations and terraform state are the two inputs to the terraform core. Terraform configuration has the details of what needs to be created and provisioned on the infrastructure and terraform state is to keep the up-to-date status of the infrastructure.

### ****Q4. What are the primary responsibilities of the provider and provisioner plugins?****

Providers and Provisioners are used in Terraform using plugins.

The responsibilities of provider plugins are:

* Creating, reading, updating and deleting resources
* Adding new resource type to an existing provider
* Writing a new provider in case the existing provider is not enough to manage the resources
* Understanding the API interactions and exposing the resources
* Help in local utilities for tasks like generating random numbers for unique resource names
* Authentication of the provider of infrastructure

The responsibilities of provisioner plugins are:

* To pass data to VMs and other compute resources
* To launch configuration management products
* To execute scripts on a local or a remote machine to create or destroy resources
* To bootstrap a resource, to perform clean-up before destroying the resource

### ****Q5. How does Terraform help in discovering plugins and what are the different behaviours of it?****

After running the terraform init command, terraform reads all the configuration files from the working directory to check which plugins are required depending on the configuration for the infrastructure. Then it searches for those plugins, checks where they are installed, sometimes it downloads additional plugins if required and decides which plugin version must be used. Then terraform writes a lock file to make sure the same plugin version is used when the next terraform init command runs.

There are three different ways in which terraform plugins behave in order to discover plugins:

1. They use built-in provisioners. These are used where plugins are always available and included in terraform binary
2. They use providers distributed by Hashicorp. The plugins here are automatically downloaded if they are not installed already
3. They use 3rd party providers and provisioners. Here the necessary plugins are manually installed

### ****Q6. What are the ways to lock Terraform module versions?****

Here you need to use the proven way to lock the terraform module version. You can use the terraform module registry as a source and provide the attribute as ‘version’ in the module in a terraform configuration file. If you are using the GitHub repository as a source, then you need to specify the branch, version and query string with ‘? ref’.

### ****Q7. What is Terragrunt in Terraform?****

Terragrunt is a tool by Gruntwork which work on the principle of DRY (Don’t Repeat Yourself). It is a thin wrapper which has some extra tools using which you can keep the configurations DRY. You can use it for terraform modules and terraform states also.

By using Terragrunt, you write your codes on Terraform only once, even if you have multiple environments. You do not write configuration codes for every environment. It helps you get rid of duplicate code in the backend. Using Terragrunt, you can manage the terraform state once by defining it in the root directory and all the child modules can inherit it. In case you want to apply something on multiple modules, using terragrunt, you can do that using a single command which will make the changes in all the modules.

Terragrunt helps in encouraging versioned modules and reusability for multiple environments. It also comes with additional features such as lifecycle hooks which adds flexibility in using terraform. This tool also supports continuous deployment practices natively. Here the code is packaged, versioned and reused by multiple environments in the CICD pipelines.

### ****Q8. What is the difference between Terraform and CloudFormation?****

Terraform is a product by Hashicorp and CloudFormation is a product of Amazon Web services. Both are similar in many ways as they are used to do similar tasks but there are multiple differences between them.

HashiCorp Configuration Language (HCL) is used by Terraform to keep it human-readable and it is machine friendly. On the other hand, CloudFormation uses JSON or YAML for defining the configuration.

Terraform provides amazing native support for modules. Terraform registry has plenty of open-source modules readily available to be used. Comparative to Terraform, the support of the module in CloudFormation is not that great. It has some features to modularize your template, but most of the time, they are not used, it is up to you if you want to use them.

CloudFormation is an AWS service so it works smoothly with AWS infrastructure but takes time to support new AWS service capabilities. Terraform supports AWS resources and is faster than CloudFormation most of the time when new working with new AWS features. Terraform also have better support for other cloud providers such as Microsoft Azure and Google Cloud Platform and many other 3rd parties.

One more very important difference between the two is that while applying and update, terraform will show you all the changes that will happen, it will drill down and share all the module information. Whereas in CloudFormation, while updating, it only tells about the nested stacks but does not drill down into the details.

### ****Q9. How would you recover from a failed apply in Terraform?****

The usual way to represent “rolling back” in Terraform is to put your configuration in version control and commit before each change, and then you can use your version control system’s features to revert to an older configuration if needed. You always need to recommit the previous version code for it to be the new version in the version control system.

Not all changes can be rolled back purely by reverting a version control system change though. For example, if you added a new provider block and resources for that provider all in one commit and then applied the result, in order to rollback, you would need to change the configuration to still include the provider block but not include any of the resource blocks, so you would need to adjust the configuration during the revert. Terraform will then use the remaining provider block to configure the provider to run the destroy actions, after which you can finally remove the provider block too.

If the state file gets corrupted from the latest terraform run, then you can use terraform enterprise and its features of state rollback to go to the previous latest state which was working fine and was in a good state. This is possible in the enterprise version because every state change is versioned there.

### ****Q10. How to ignore duplicate resource error during terraform apply?****

It seems to be the case that when a resource conflict occurs (the IAM roles in my case), Terraform will ignore certain resource types previously saved in its state and attempt to recreate those resources from scratch.

Possible causes of this could be:

* You or someone else has executed your Terraform code and you don’t have a shared/updated state
* Someone has created them manually
* A Terraform destroy failed in a way that deleted the resources for the API but failed to save the update state

Solutions depend on what you need. You can:

* Delete those resources from your Terraform code to stop managing them with it
* Delete those resources from the API (cloud provider) and recreate them with Terraform
* Perform a terraform import of those resources and remove the terraform code that is trying to recreate them (NOT RECOMMENDED)
* Use terraform apply –target=xxx to apply only resources you need to apply (NOT RECOMMENDED)

### ****Q11. Share a few Terraform CLI Commands which you use for your day-to-day routine?****

Below are a few Terraform CLI commands which are often used on a daily basis:

* **Init:**This command is usedto initialize the working directory which has all the terraform configuration files.
* **Get:**This command is used to download an update of the module mentioned in the root module.
* **Plan:**This command is used to create an execution in order to reach a desired state of the infrastructure.
* **Apply:**This command is used to execute the execution plan created by terraform plan to reach the desired state of the infrastructure.
* **Destroy:**It isused to remove all the resources on the infrastructure and destroy it.
* **Graph:**Using this CLI command, you can visualize an execution plan or generate a visual representation of a configuration.
* **Validate:**This command is used to validate the configuration file if they are syntactically consistent.
* **Workspace:**It is used to manage the workspaces.
* **Fmt:**It is used to rewrite the terraform configuration files in a canonical format and style.

### ****Q12. What is an Execution Plan in Terraform?****

One of the important stages in a terraform lifecycle is the terraform plan state. This is the stage where the execution plan is created. Terraform plan creates an execution plan which has the details about what all things will get executed once the apply command runs. The execution plan helps you in visualizing what changes are about to happen, and there are no surprises later.

### ****Q13. What is a Resource Graph in Terraform?****

A resource graph is a graph build by terraform for all the resources. It also creates and modifies any non-dependent resources parallelly. Terraform creates a dependency graph from the configurations files and walks the graph to generate plans, refresh state etc. The resource graph is used by terraform in building infrastructure as efficiently as possible, and it provides insights to the operators about the dependencies in their infrastructure.

### ****Q14. What is the Terraform cloud?****

Terraform Cloud is an application that manages Terraform runs in a consistent and reliable environment. It includes easy access to the shared state and secret data, a private registry for sharing Terraform modules, access controls for approving changes to infrastructure, detailed policy controls for governing the Terraform configuration contents, etc.

Terraform Cloud is a hosted service and available at https://app.terraform.io. It offers free accounts for small teams and has paid plans with additional feature sets for medium-sized businesses.

For large enterprises, there is a separate terraform cloud product – Terraform Enterprise. You get a private instance of the Terraform Cloud application for an enterprise, with no resource limits and with additional enterprise-grade architectural features like SAML single sign-on and audit logging.

### ****Q15. Are callbacks possible with Terraform on Azure?****

Yes, callbacks are possible with Terraform on Azure, it is done using Azure Events Hub. Terraform has a provider called AzureRM which provides the callback functionality.

### ****Q16. What is a Remote Backend in Terraform?****

The remote backend in terraform is used to store the state of terraform and can also run operations in terraform cloud. Remote backend multiple terraform commands such as init, plan, apply, destroy (terraform version >= v0.11.12), get, output, providers, state (sub-commands: list, mv, pull, push, rm, show) , taint, untaint, validate and many more. It can work with a single remote terraform cloud workspace or even multiple workspaces. For running remote operations like terraform plan or terraform apply, you can use terraform cloud’s run environment.

### ****Q17. What is State File Locking?****

State file locking is a mechanism in terraform where operation on a specific state file is blocked by multiple callers to avoid any conflict between the team members. Once the lock from one caller is released, then only any other caller can operate on that state file after taking a lock on it. This helps in preventing any corruption of the state file. It is a backend operation, so the acquiring of lock on a state file in backend. If it takes more time than expected to acquire a lock on the state file, you will get a status message as an output.

### ****Q18. What is a Tainted Resource?****

Tainted resources are those resources in terraform which are forced to be destroyed and they are asked to be recreated on the next apply command. When you mar a resource as tainted, nothing changes on infrastructure but state file is updated with this information. After marking a resource as tainted, terraform plan out will show that resource will get destroyed and recreated, and when the next apply happens the changes will get implemented.

### ****Q19. How do you test a terraform module?****

There are multiple ways to do that, the most popular way is using Terratest. Terratest has been developed by Gruntworks. It has been built with the purpose of testing the terraform module (code) by using a unit testing framework. This framework has been built in the Go programming language. To run Terratest, you just need to provide the terraform file (module file) and the Go test. And run the command go test.

### ****Q20. Tell about a few Terraform best practices.****

* Follow a proper directory structure of the terraform workspace. The projects on production can get very complex if they are not well-structured.
* Use naming conventions to make the cluster structure understandable.
* Always use the latest stable terraform version, they have new features and a lot of security patches.
* You official terraform modules, don’t waste time in creating similar modules that are already available in the terraform registry.
* Always backup the terraform state files.
* Use official terraform docker containers in your CICD pipeline jobs.
* Lock the state files to avoid any conflict between teams or team members.

## Final Thoughts

Terraform is a vast topic and there is plenty to learn. Do not just mug up these answers, understand the technicalities of these answers also. Terraform documentation is the best place to get in technical depth. But these top 20 terraform interview questions will give you a kickstart to your future terraform interviews. So, prepare well and all the best!

-------------------------------------------------------------------------------------------------------------------

### 1) What is Terraform?

**Terraform** is an open-source infrastructure as code software tool created by HashiCorp. It enables users to define and provision a datacenter infrastructure using a high-level configuration language known as Hashicorp Configuration Language, or optionally JSON.

### 2) Enlist major features of Terraform?

The key features of Terraform are

* Infrastructure as Code (IAC)
* Execution Plans
* Change Automation
* Resource Graph

### 3) Why Terraform is preferred for DevOps?

There are a bunch of reasons for giving preference to **Terraform** to be used as one of the significant tools of **DevOps**. The main motive of DevOps is to bring efficiency and quality in software delivery. For this, some tools are required for making the delivery smoother, faster and efficient. Here, terraform comes to the limelight where it aids organizations in automating and also aids with infrastructure as code. Terraform is helpful with its availability of implementing every type of coding principle. The extraordinary feature of terraforming includes its quickness and the operations performed by it. These are some of the important reasons for which Terraform is gaining popularity and attention in the organizations.

### 4) What is terraform in aws?

**Terraform by HashiCorp**, an AWS Partner Network (APN) **Advanced Technology Partner** and member of the AWS DevOps Competency, is an "infrastructure as code" tool similar to **AWS** CloudFormation that allows you to create, update, and version your Amazon Web Services (AWS) infrastructure.

### 5) What does HCL stand for?

In **Terraform HCL** stands for **HashiCorp Configuration Language**. It is a configuration language built by HashiCorp that is used to build a structured configuration language that is both human and machine-friendly for use with command-line tools but specifically targeted towards DevOps tools, servers, etc.

**Download Free :**[Terraform Interview Questions PDF](https://www.interviewmocks.com/pdf/terraform-interview-questions.pdf)

### 6) Explain the architecture of terraform?

The architecture of terraform consists of following components:

* CLI (Command Line interface)
* Backends for executing operations,storing state, and storing workspace-defined variables
* Configuration Loader
* State Manager
* Graph Builder
* Graph Walk
* Vertex Evaluation
* Expression Evaluation
* Sub-graphs

### 7) List some major competitors of terraform?

**Ansible**, **Kubernetes**, **Packer**, **Cloud Foundry**, and **Pulumi** are the most popular alternatives and competitors to **Terraform**.

### 8) In which programming language Terraform is written?

**Terraform** is an open-source tool created by **HashiCorp** and written in the **Go programming language**.

### 9) What is use of Terraform CLI? Enlist few major command.

The **terraform CLI** is a well-behaved command-line application. It is used to run commands for terraforming.

Below are few common Terraform commands:

* **Apply:** builds or changes infrastructure.
* **Console:** Interactive console for Terraform interpolations.
* **destroy:** Destroy Terraform-managed infrastructure.
* **env:** Workspace management
* **fmt:** Rewrites config files to canonical format
* **get:** Download and install modules for the configuration
* **graph:** Build a visible graph of Terraform resources Import: existing infrastructure into Terraform
* **Init:** Initialize a Terraform working directory
* **output:** Read output from a state file plan: Generate and show an execution plan validate: Validates the Terraform files
* **version:** Prints the Terraform version
* **Workspace:** Workspace management

### 10) What is a provider in Terraform? Enlist some Terraform Providers.

**Terraform provider** is responsible for understanding API interactions and exposing resources. Providers generally are an IaaS (e.g. Alibaba Cloud, AWS, GCP, Microsoft Azure, OpenStack), PaaS (e.g. Heroku), or SaaS services (e.g. Terraform Cloud, DNSimple, Cloudflare).

**Some Terraform providers are:**

* **IaaS services:** Examples are Alibaba Cloud, AWS, GCP, Microsoft Azure, OpenStack.
* **PaaS services:** Examples are Heroku.
* **SaaS services:** examples are Terraform Cloud, Cloudflare, DNSimple.

### 11) For what provisioners are used in Terraform?

**Provisioners in Terraform** can be used to prepare servers or other infrastructure objects for service. It models specific actions on the local machine as well as on remote machines.

### 12) Enlist some Built-in Provisioners available in Terraform?

**Below is the list of some Built-in provisioners in Terraform:**

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

### 13) What are Modules in Terraform?

A**module in Terraform** is a container for several resources that are used together. The root module is compulsory for every terraform that includes resources mentioned in the .tf files.

### 14) What is Terraform cloud? Enlist few features provided by it?

**Terraform Cloud** is an application that let the teams use the Terraform together. It runs in a reliable environment and includes easy access to shared state and secret data. It provides access controls for adopting changes to infrastructure and a private registry for sharing terraform modules.

**Features of the Terraform cloud are:**

* Audit logging
* SAML single sign-on.

### 15) How to check installed version of Terraform?

To check the installed version of Terraform, use the below command:

**Syntax**

terraform -version

### 16) What do you mean by IAC in Terraform?

**IAC** is an abbreviation to the term **"Infrastructure as Code"**. IaC refers to a system whereby developers can manage and provision the computer data centers automatically instead of getting into a manual process. Terraform, for instance, is an example tool of IaC.

### 17) Can terraform be used for on-prem infrastructure?

Yes, Terraform can be used for on-prem infrastructure. There are many providers that are available. you can choose any one of them which suits you best. Many also build customer terraform providers for themselves, all needed is just an API.

### 18) What is Oracle Cloud Infrastructure?

**Oracle Cloud** is a cloud computing service offered by Oracle Corporation providing servers, storage, network, applications and services through a global network of Oracle Corporation managed data centers. The company allows these services to be provisioned on-demand over the Internet.

### 19) How to create dependency between modules in terraform?

In Terraform dependencies are normally created automatically by references, rather than explicitly using depends\_on. In terraform, there is no way to use depends\_on variables.

### 20) How do you deal with versioning in Terraform?

**Semantic Versioning** of Models should be used. Apply should only be done with Versioned modules/tags.

### 21) What is used of terraform apply command?

The **Terraform apply command** is used to apply the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a terraform plan execution plan.

### 22) What is the different between Platform.sh and Terraform?

**Platform.sh** and **Terraform** are similar in that they both subscribe to the idea of "infrastructure as code".

**Platform.sh** is a second-generation Platform-as-a-Service built especially for continuous deployment. It allows you to host web applications on the cloud while making your development and testing workflows more productive whereas **Terraform** is an open-source infrastructure as code software tool that enables users to define and provision a data center infrastructure using a high-level configuration language known as Hashicorp Configuration Language, or optionally JSON

### 23) What is Software Defined Networking?

**Software-defined networking (SDN)** technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring making it more like cloud computing than traditional network management

### 24) Which command is used to destroy Terraform-managed infrastructure?

Terraform **destroy command** is used to destroy the Terraform-managed infrastructure.

**Syntax**

**terraform** **destroy** [options] [dir]

### 25) What is use of fmt command in Terraform?

The **terraform fmt command** is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability.

**Syntax**

**terraform** **fmt** [options] [DIR]

### 26) What does the providers command do?

The **terraform providers command** prints information about the providers used in the current configuration.

**Syntax**

terraform providers [config-path]

### 27) What is Terragrunt?

**Terragrunt** is a thin wrapper for Terraform that implements the practices advocated by the Terraform: Up and Running book. We've found Terragrunt helpful as it encourages versioned modules and reusability for different environments with some handy features, including recursive code execution in subdirectories.

### 28) What is terraform state?

**Terraform state** is a command that is used for advanced state management. The state is a necessary requirement for Terraform to function. Terraform must store state about your managed infrastructure and configuration. This state is used by Terraform to map real-world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures. Syntax

**terraform** **state** [options] [args]

### 29) What is terraform init?

The **terraform init** is a command that is used to initialize a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It is safe to run this command multiple times.

**Syntax**

**terraform** **init** [options] [DIR]

### 30) What is terraform backend?

A **"backend"** in Terraform determines how the state is loaded and how an operation such as apply is executed. This abstraction enables non-local file state storage, remote execution, etc.

By default, Terraform uses the "local" backend, which is the normal behavior of Terraform you're used to

-----------------------------------------------------------------------------------------------

<https://www.edureka.co/community/cloud-computing/terraform>

-------------------------------------------------------------------------------------------------

[Home](https://www.cyberithub.com/)

[DevOps](https://www.cyberithub.com/category/devops/)

[Terraform](https://www.cyberithub.com/category/devops/terraform/)

13 Best Terraform Interview Questions and Answers

# 13 Best Terraform Interview Questions and Answers

[cyberithub](https://www.cyberithub.com/author/bholu/)  December 12, 2019  [DevOps](https://www.cyberithub.com/category/devops/), [Terraform](https://www.cyberithub.com/category/devops/terraform/)  [No Comments](https://www.cyberithub.com/terraform-interview-questions/#respond)

Table of Contents

* [Terraform Interview Questions](https://www.cyberithub.com/terraform-interview-questions/#Terraform_Interview_Questions)
  + [1. What is Terraform?](https://www.cyberithub.com/terraform-interview-questions/#1_What_is_Terraform)
  + [2. What do you mean by IaC?](https://www.cyberithub.com/terraform-interview-questions/#2_What_do_you_mean_by_IaC)
  + [3. Is the management of Azure Availability Zones alike the management done by distinct cloud providers available?](https://www.cyberithub.com/terraform-interview-questions/#3_Is_the_management_of_Azure_Availability_Zones_alike_the_management_done_by_distinct_cloud_providers_available)
  + [4. Can terraform be used for on-prem infrastructure?](https://www.cyberithub.com/terraform-interview-questions/#4_Can_terraform_be_used_for_onprem_infrastructure)
  + [5. On Azure, Can callbacks be used with terraforming?](https://www.cyberithub.com/terraform-interview-questions/#5_On_Azure,_Can_callbacks_be_used_with_terraforming)
  + [6. Can the Terraform Module Versions be locked anyhow?](https://www.cyberithub.com/terraform-interview-questions/#6_Can_the_Terraform_Module_Versions_be_locked_anyhow)
  + [7. Mention the version controls that are supported besides GitHub?](https://www.cyberithub.com/terraform-interview-questions/#7_Mention_the_version_controls_that_are_supported_besides_GitHub)
  + [8. Give a configuration of for creating a single E2C instance in Amazon Web Services ( AWS ).](https://www.cyberithub.com/terraform-interview-questions/#8_Give_a_configuration_of_for_creating_a_single_E2C_instance_in_Amazon_Web_Services_(_AWS_))
  + [9. Are themes supported by terraforming?](https://www.cyberithub.com/terraform-interview-questions/#9_Are_themes_supported_by_terraforming)
  + [10. What are the features of Terraform?](https://www.cyberithub.com/terraform-interview-questions/#10_What_are_the_features_of_Terraform)
  + [11. Can you shortly explain the best practices for cloning an infrastructure with Terraform?](https://www.cyberithub.com/terraform-interview-questions/#11_Can_you_shortly_explain_the_best_practices_for_cloning_an_infrastructure_with_Terraform)
  + [12. If I was interested in the on-prem Terraform product, how could I use Sentinel as well? Is it a separately licensed module of the Terraform server?](https://www.cyberithub.com/terraform-interview-questions/#12_If_I_was_interested_in_the_onprem_Terraform_product,_how_could_I_use_Sentinel_as_well_Is_it_a_separately_licensed_module_of_the_Terraform_server)
  + [13. Is there a way to bulk import the state of current cloud subscription into Terraform state?](https://www.cyberithub.com/terraform-interview-questions/#13_Is_there_a_way_to_bulk_import_the_state_of_current_cloud_subscription_into_Terraform_state)

In this article, i will take you through 13 Best Terraform Interview Questions and Answers. Terraform is an open-source infrastructure as code software tool created by HashiCorp. It enables users to define and provision a datacenter infrastructure using a high-level configuration language known as Hashicorp Configuration Language, or optionally JSON.



## Terraform Interview Questions

Below are the few Best terraform interview questions that usually get asked in DevOps Interview.

### 1. What is Terraform?

Ans. Terraform is an open source tool for securely and efficiently provisioning and managing cloud infrastructure.It is a highly reliable tool that supports multiple infrastructure through providers. You can always use Terraform to create, modify, or delete multiple resources, such as ECS, VPC, RDS, and SLB.

It writes the infrastructure, for ex: virtual machines, storage accounts, and network interfaces in the configuration file that describes the cloud resource topology. The CLI of Terraform provides a simple mechanism, which is used for deploying and versioning configuration files on AWS Cloud or any other supported cloud.

### 2. What do you mean by IaC?

Ans. IaC is known as “Infrastructure as Code”. It refers to a system through which developers can maintain and provision the computer data centers automatically instead of getting into a manual process. Terraform, for instance, is an example tool of IaC.

### 3. Is the management of Azure Availability Zones alike the management done by distinct cloud providers available?

Ans.These zones are situated within an area and each zone has its separate source of power and network. Any region where availability zone is enabled has at least three such availability zones. The AzureRM Terraform provider have no resource accessible to manage the Azure Availability Zones at present.

### 4. Can terraform be used for on-prem infrastructure?

Ans. Yes, Terraform can be used for an on-prem infrastructure. There are many providers which are available that can be chosen as per your requirement. Many also build customer terraform providers for themselves, all needed is just an API.

Advertisements

### 5. On Azure, Can callbacks be used with terraforming?

Ans.Yes, callbacks can be used on Azure with terraforming with the help of the Azure Event Hubs. The AzureRM provider of Terraform is today making this functionality easily available. Now the terraform actually comes already pre-installed itself in the portal of Azure Cloud Shell of Microsoft.

### 6. Can the Terraform Module Versions be locked anyhow?

Ans.Yes, there is a way to lock it. if terraform module registry is being used as a source, then the ‘version’ attribute can be used in the module in a configuration file of Terraform. If as a source, a repository of Github is being used then in the query string, branch, and versions to be specified by’? ref ‘.

### 7. Mention the version controls that are supported besides GitHub?

Ans. This is simple terraform interview questions yet should be taken into note significantly. GitLab CE and GitLab EE are supported. Bucket cloud is also supported.

### 8. ****Give a configuration of for creating a single E2C instance in Amazon Web Services ( AWS ).****

provider "aws" {

region = "ap-south-1"

}

resource "aws\_instance"

"example" {

ami = "ami-4fc58420"

instance\_type = "t2.micro"

tags {

Name = "terraform-example"

}

}

### 9. Are themes supported by terraforming?

Ans. Yes. Gtk themes are efficiently supported by the 0.3.1 version of terraforming. If your system contains a gtk theme, then it can be enabled by performing this: cp/usr/wherever/THEMENAME/gtk/gtkrc $HOME/.gtkrc If it shows error on opening the theme files and theme files fail to open, then .gtkrc should be edited and after that attach a line like the below one at the starting of the file. pixmap\_path”/usr/wherever/THEMENAME/gtk” Now the theme should be able to load on startup.

### 10. What are the features of Terraform?

Ans. This is very important for Terraform Interview Questions. Main features include:

**Graphing** – Its features of graphing that are built-in are helpful in visualizing the infrastructure.  
**Custom Syntax** – It’s custom syntax is very friendly which aids in enhancing efficiency.  
**Resource Relationships** – A very beneficial feature of terraforming is that it is able to understand resource relationships.  
**Updates** – The updates and features are added by the Open Source Project. It does so with a group of lots of contributors.  
**Improved Maintenance** – It is capable of breaking down the configuration into small parts or chunks for improving the organization and the maintenance.

### 11. Can you shortly explain the best practices for cloning an infrastructure with Terraform?

Ans. If you’re referring to promotion, you could have dev and prod branches of a repository with Terraform code and then point dev and prod workspaces against those branches and promote changes from dev to prod to clone, but set different values of Terraform variables to account for differences between dev and prod. If by “clone” you mean bring existing infrastructure not provisioned by Terraform under Terraform management, see the below questions about to terraform import. If you just want to duplicating an existing infrastructure with Terraform, you would need to map all of your existing resources into a Terraform template.

### 12. If I was interested in the on-prem Terraform product, how could I use Sentinel as well? Is it a separately licensed module of the Terraform server?

Ans. Absolutely. Sentinel is available with Terraform Enterprise Premium for both the SaaS and Private versions. There is no separate charge for Sentinel. It is not licensed separately.

### 13. Is there a way to bulk import the state of current cloud subscription into Terraform state?

Ans. You can use the terraform import command to import individual resources into your Terraform state, but there is not currently a bulk import tool.

---------------------------------------------------------------------------------------------------------

**External data sources must return information in what format?**

* A. XML
* B. HTML
* C. JSON
* D. YAML

Ans: C. JSON

**Which of the following is an example of Source Control Management software?**

* A. Jenkins
* B. Visual Studio Code
* C. Docker
* D. Git

Ans: D. Git

**Why would you use a partial configuration for a backend?**

* A. So that you can store your AWS credentials in the configuration
* B. To make it simpler to change the backend storage type to a new service
* C. So that the remainder of configuration can be defined the variables
* D. To allow backend configuration data to be defined at runtime

Ans: D. To allow backend configuration data to be defined at runtime

**What environment variable lets Terraform know it’s running in an automation context?**

* A. TF\_AUTO\_YES
* B. TF\_IN\_AUTOMATION
* C. TF\_AUTOMATION
* D. TF\_AUTOMATION\_TRUE

Ans: B. TF\_IN\_AUTOMATION

**What is special about the default workspace?**

* A. It is removed when you create your first workspace.
* B. It cannot be deleted.
* C. You cannot select it.
* D. It is named after the main configuration file.

Ans: B. It cannot be deleted.

**What are the four common components of Configuration Management systems?**

* A. Get, Test, Set, Put
* B. Authorization, Accounting, Auditing, and Access
* C. Normalization, Comparison, Accounting, and Validation
* D. Identification, Control, Accounting, and Verification

Ans: D. Identification, Control, Accounting, and Verification

**Which Ansible command executes a playbook?**

* A. ansible-run-playbook
* B. ansible-run
* C. ansible-execute
* D. ansible-playbook

Ans: D. ansible-playbook

**What is the purpose of running the terraform init command?**

* A. Init will download the required plugins and modules for the configuration.
* B. Init is an optional command to configure aliases.
* C. Init will remove any existing resources from previous deployments.
* D. Init will create new configuration files for you to populate.

Ans: A. Init will download the required plugins and modules for the configuration.

**What does the import command do?**

* A. Imports a text file containing values to be used in the configuration
* B. Imports an existing, unmanaged resource into the Terraform configuration
* C. Creates a new Terraform configuration for unmanaged resources
* D. Imports the settings from a previous Terraform installation

Ans: B. Imports an existing, unmanaged resource into the Terraform configuration

**What type of request does the HTTP data source use?**

* A. PUT
* B. POST
* C. LIST
* D. GET

Ans: D. GET

**What command is being deprecated in favor of workspaces?**

* A. Production
* B. Environment
* C. Posture
* D. Staging

Ans: B. Environment

**Which option tells Terraform to use a local directory for plugins?**

* A. -plugin-dir=
* B. -plugin-local=
* C. -plugin-source=
* D. -init-dir=

Ans: A. -plugin-dir=

**What format does a rendered template take?**

* A. Map
* B. String
* C. JSON
* D. List

Ans: B. String

**Why would you use the option ‘-input=false’ when running terraform init?**

* A. It disables the use of variable files for input.
* B. It ensures that Terraform only generates output.
* C. It prevents Terraform from import files.
* D. It prevents Terraform from prompting the user for input.

Ans: D. It prevents Terraform from prompting the user for input.

**What format are ansible files written in?**

* A. JSON
* B. HTML
* C. YAML
* D. XML

Ans: C. YAML

**What is one benefit of using an immutable deployment?**

* A. Instances are never patched or upgraded in-place.
* B. Instances and infrastructure are never redeployed.
* C. It offloads all work to configuration management software.
* D. Images are generalized and configured after the instance is created.

Ans: A. Instances are never patched or upgraded in-place.

**Where can you find official modules to use with Terraform?**

* A. You must create your own
* B. On the provider’s website
* C. You have to request them from Hashicorp
* D. On the Terraform module repository

Ans: D. On the Terraform module repository

**In the import command, what do the ADDR and ID fields represent?**

* A. ADDR is the Terraform identifier; ID is the provider identifier.
* B. ADDR is the name of the key; ID is the value of the key.
* C. ADDR is the loopback address; ID is the static IP address.
* D. ID is the Terraform identifier; ADDR is the provider identifier.

Ans: A. ADDR is the Terraform identifier; ID is the provider identifier.

**How is the source for a template defined?**

* A. On the Terraform website
* B. Using a map generated by Terraform
* C. By pulling from GitHub
* D. Either in-line or from a file

Ans: D. Either in-line or from a file

**What command must be run to configure the backend in Terraform?**

* A. terraform backend
* B. terraform fmt
* C. terraform plan
* D. terraform init

------------------------------------------------------------------------------------------------------