**Interview Questions**

**What is Terraform?**

Terraform is an open source “**Infrastructure as Code**” tool, created by **HashiCorp**. It is a declarative coding tool, Terraform enables developers to use a high-level configuration language called HCL (**HashiCorp Configuration Language**) to describe the desired “**end-state**” cloud or on-premises infrastructure for running an application. It then generates a plan for reaching that end-state and executes the plan to provision the infrastructure.

**What is use of Terraform CLI?**

**CLI in terraform** is a well mannered command line application. In erroneous cases, a non-zero exit status will be returned. CLI also responds to **-h** and**— help** as you’d most likely expect. To get help for any particular command, pass the -h flag to the relevant sub-command.

**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.

**What are the use cases of Terraform?**

Heroku App Setup  
Multi-Tier Applications  
Self-Service Clusters  
Software Demos  
Disposable Environments  
Software Defined Networking  
Resource Schedulers  
Multi-Cloud Deployment

**What are the advantages of Terraform?**

Platform Agnostic  
State Management  
Operator Confidence

**How can Terraform build infrastructure so efficiently?**

Terraform builds a graph of all your resources, and parallelizes the creation and modification of any non-dependent resources. Because of this, Terraform builds infrastructure as efficiently as possible, and operators get insight into dependencies in their infrastructure.

**What are Modules in Terraform?**

Terraform modules provide an easy way to abstract common blocks of configuration into reusable infrastructure elements. if you wan to write a module, you have to apply the same concepts that you would for any configuration. Modules are collections of **.tf files** containing resources, input variables, and outputs, which exist outside the root folder of your configuration.

**Explain 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. We can see who is accessing your state file, control who accesses the state file, and more.

**How we can check installed version of Terraform?**

To verify your installation and check the version, launch Windows PowerShell and enter: **terraform -version**.

**What are terraform Provisioners?**

Provisioners are used for **executing scripts** or **shell commands** on a local or remote machine as part of resource **creation/deletion**. They are similar to “**EC2 instance user data**” scripts that only run once on the creation and if it fails terraform marks it tainted.

**How can we upgrade plugins on Terraform?**

We can modify terraform init ‘s plugin behavior with the following options: **-upgrade — Update** all previously installed plugins to the newest version that complies with the configuration’s version constraints

**What are the primary responsibilities of Terraform Core?**

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

**What is the Terraform Plugins?**

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.

**What is the Terraform configuration for creating a single EC2 instance on AWS?**

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"

}

}

**What is Infrastructure as Code in Terraform?**

Infrastructure is described using a high-level configuration syntax. It allows a blueprint of the datacenter to be versioned and treated as any other code. Additionally, infrastructure can be shared and re-used.

**What is Execution Plans in Terraform?**

Terraform has a 'planning' step where it generates an execution plan. The execution plan shows what Terraform will do when the call apply. This will avoid any surprises when Terraform manipulates infrastructure.

**What is Change Automation in Terraform?**

Complex changesets can be applied to the infrastructure with minimal human interaction. With the previously mentioned execution plan and resource graph, it can be known exactly what Terraform will change and in what order, avoiding many possible human errors.

**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.

**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.

**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.

**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.

**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.

**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.

**What is cloud-agnostic in terms of provisioning tools?**

cloud-agnostic and allows a single configuration to be used to manage multiple providers, and to even handle cross-cloud dependencies.

**What are the meta-arguments that are defined by Terraform itself and available for all provider blocks?**

**version:** Constraining the allowed provider versions  
**alias:**using the same provider with different configurations for different resources

**If different teams are working on the same configuration. How do you make files to have syntactically valid and internally consistent?**

terraform validate  
This command will check and report errors within modules, attribute names, and value types.Validate your configuration. If your configuration is valid, Terraform will return a success message.

**How do you inspect the current state of the infrastructure applied?**

terraform show  
When you applied your configuration, Terraform wrote data into a file called terraform.tfstate. This file now contains the IDs and properties of the resources Terraform created so that it can manage or destroy those resources going forward.

**What is plug-in based architecture?**

Defining additional features as plugins to your core platform or core application. This provides extensibility, flexibility and isolation

**What is the command to create infrastructure?**

terraform apply

**What is the command to show the execution plan and not apply?**

terraform plan

**If your state file is too big and you want to list the resources from your state. What is the command?**

terraform state list  
<https://learn.hashicorp.com/terraform/getting-started/build#manually-managing-state>

**How do you define provisioners?**

resource "aws\_instance" "example" {  
  ami           = "ami-b374d5a5"  
  instance\_type = "t2.micro"

  provisioner "local-exec" {  
    command = "echo hello > hello.txt"  
  }  
}Provisioner block within the resource block. Multiple provisioner blocks can be added to define multiple provisioning steps. Terraform supports [multiple provisioners](https://www.terraform.io/docs/provisioners/index.html)<https://learn.hashicorp.com/terraform/getting-started/provision>

**What is a local-exec provisioner and when do we use it?**

The local-exec provisioner executing a command locally on your machine running Terraform.We use this when we need to do something on our local machine without needing any external URL

**What is a remote-exec provisioner and when do we use it?**

Another useful provisioner is remote-exec which invokes a script on a remote resource after it is created. This can be used to run a configuration management tool, bootstrap into a cluster, etc.

**Are provisioners runs only when the resource is created or destroyed?**

Provisioners are only run when a resource is created or destroyed. Provisioners that are run while destroying are Destroy provisioners.They are not a replacement for configuration management and changing the software of an already-running server, and are instead just meant as a way to bootstrap a server.

**What do we need to use a remote-exec?**

In order to use a remote-exec provisioner, you must choose an **ssh** or **winrm** connection in the form of a connection block within the provisioner.Here is an example  
provider "aws" {  
  profile = "default"  
  region = "us-west-2"  
}  
resource "aws\_key\_pair" "example" {  
  key\_name   = "examplekey"  
  public\_key = file("~/.ssh/terraform.pub")  
}  
resource "aws\_instance" "example" {  
  key\_name      = aws\_key\_pair.example.key\_name  
  ami           = "ami-04590e7389a6e577c"  
  instance\_type = "t2.micro"  
  connection {  
    type        = "ssh"  
    user        = "ec2-user"  
    private\_key = file("~/.ssh/terraform")  
    host        = self.public\_ip  
  }  
  provisioner "remote-exec" {  
    inline = [  
      "sudo amazon-linux-extras enable nginx1.12",  
      "sudo yum -y install nginx",  
      "sudo systemctl start nginx"  
    ]  
  }  
}

**When terraform mark the resources are tainted?**

If a resource successfully creates but fails during provisioning, Terraform will error and mark the resource as "tainted".A resource that is tainted has been physically created, but can't be considered safe to use since provisioning failed.

**You applied the infrastructure with terraform apply and you have some tainted resources. You run an execution plan now what happens to those tainted resources?**

When you generate your next execution plan, Terraform will not attempt to restart provisioning on the same resource because it isn't guaranteed to be safe.Instead, Terraform will remove any tainted resources and create new resources, attempting to provision them again after creation.<https://learn.hashicorp.com/terraform/getting-started/provision>

**Terraform also does not automatically roll back and destroy the resource during the apply when the failure happens. Why?**

Terraform also does not automatically roll back and destroy the resource during the apply when the failure happens, because that would go against the execution plan: the execution plan would've said a resource will be created, but does not say it will ever be deleted. If you create an execution plan with a tainted resource, however, the plan will clearly state that the resource will be destroyed because it is tainted.<https://learn.hashicorp.com/terraform/getting-started/provision>

**How do you manually taint a resource?**

terraform taint resource.id

**Does the taint command modify the infrastructure?**

terraform taint resource.id  
This command will not modify infrastructure, but does modify the state file in order to mark a resource as tainted. Once a resource is marked as tainted, the next plan will show that the resource will be destroyed and recreated and the next apply will implement this change.

**By default, provisioners that fail will also cause the Terraform apply itself to fail. Is this true?**

True

**By default, provisioners that fail will also cause the Terraform apply itself to fail. How do you change this?**

The **on\_failure** setting can be used to change this. The allowed values are:**continue:** Ignore the error and continue with creation or destruction.**fial:** Raise an error and stop applying (the default behavior). If this is a creation provisioner, taint the resource.//  
Example  
resource "aws\_instance" "web" {  
  # ...

  provisioner "local-exec" {  
    command **=** "echo The server's IP address is ${self.private\_ip}"  
    on\_failure **=** "continue"  
  }  
}

**When does the terraform does not recommend using provisions?**

Passing data into virtual machines and other compute resources  
<https://www.terraform.io/docs/provisioners/#passing-data-into-virtual-machines-and-other-compute-resources>  
Running configuration management software  
<https://www.terraform.io/docs/provisioners/#running-configuration-management-software>