1) What Terraform?
2) What Cloud Providers Terraform Supports?
3) What Do You Need To Learn Before Start Using Terraform?
4) Terraform Basic Commands
5) Demo References

Terraform Fundamentals & Demo

By Sandip Das

What is Terraform?

Terraform 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.

Configuration files describe to Terraform the components needed to run a single application or your entire datacenter. Terraform generates an execution plan describing 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.

The infrastructure Terraform can manage includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc.

The key features of Terraform are:
Infrastructure as Code:
Infrastructure is described using a high-level configuration syntax. This allows a blueprint of your datacenter to be versioned and treated as you would any other code. Additionally, infrastructure can be shared and re-used.

Execution Plans:
Terraform has a "planning" step where it generates an *execution plan*. The execution plan shows what Terraform will do when you call apply. This lets you avoid any surprises when Terraform manipulates infrastructure.
Resource Graph: Terraform builds a graph of all your resources, and parallelism 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.
Change Automation: Complex changesets can be applied to your infrastructure with minimal human interaction. With the previously mentioned execution plan and resource graph, you know exactly what Terraform will change and in what order, avoiding many possible human errors.
-- From Terraform Official Site

Note: Don’t worry, I will explain in details what all above actually means in the upcoming video and there going to be a whole series on it, make sure subscribe my YouTube Channel and check for contents time to time: https://www.youtube.com/SandipDas-official

What Cloud Providers Terraform Supports?

Of Course It support: Alibaba Cloud, AWS, GCP, Microsoft Azure, here these are treated as A Provider.

As per Terraform, **Provider** is responsible for understanding API interactions and exposing resources.

Here is the full list of Providers:
ACME, Akamai, Alibaba Cloud, Archive, Arukas, Avi Vantage, Aviatrix, AWS, Azure, Azure Active Directory, Azure Stack, A10, Networks, Bitbucket, Brightbox, CenturyLinkCloud, Chef, CherryServers, Circonus, Cisco ASA, Cisco ACI, Cloudflare,CloudScale.ch, CloudStack, Cobbler, Consul, Datadog, DigitalOcean, DNS, DNSimple, DNSMadeEasy, Docker, , ,Dome9 ,Dyn,Exoscale ,External,F5 BIG-IP,Fastly,FlexibleEngine, FortiOS, Genymotion, GitHub, GitLab, Google Cloud Platform, Grafana, Gridscale, Hedvig, Helm, Heroku, Hetzner Cloud, HTTP, HuaweiCloud, HuaweiCloudStack, Icinga2, Ignition, InfluxDB, JDCloud, Kubernetes, LaunchDarkly, Librato, Linode, Local, Logentries, LogicMonitor, Mailgun, MongoDB Atlas, MySQL, Naver Cloud, Netlify, New Relic, Nomad, NS1, Null, Nutanix, 1&1, OpenNebula, OpenStack, OpenTelekomCloud, OpsGenie, Oracle Cloud Infrastructure, Oracle Cloud Platform, Oracle Public Cloud, OVH, Packet, PagerDuty, Palo Alto Networks, PostgreSQL, PowerDNS, ProfitBricks, Pureport, RabbitMQ, RancherRancher2Random, , RightScale, Rundeck, RunScope, Scaleway, Selectel, SignalFx, Skytap, SoftLayer, Spotinst, StatusCake, TelefonicaOpenCloud, Template, TencentCloud, Terraform, Terraform Cloud, TLS, Triton, UCloud, UltraDNS, Vault , Venafi, VMware NSX-T, VMware vCloud Director, VMware vRA7, VMware vSphere, Vultr, Yandex

You Don’t have to learn about all prodivers, just focus on what needed, but it’s good to know what all providers are supported in Terraform, so check full list just for reference.

What You Need To Learn Before Start Using Terraform

Terraform is often referred as Infrastructure as Code, so past coding experience will definately be an advantage for you although it’s just High level Syntax i.e. mainly configuring but you have to think and implement logically, use terraform variable and learn Syntax used in Terraform.

Now this Syntax Highly depend on Providers e.g. AWS , GCP , Aure, or any provider of your choice (as you might already seen in full provider list), so having good knowledge over particular provider is absolutely necessary.

Other than that, Linux skills and Bash scripting skills definitely will be useful time to time.

Terraform Basic Commands

After Installing Terraform CLI, if you type "terraform" in command line, you will see like below:

```
$ terraform
Usage: terraform [-version] [-help] <command> [args]
Common 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              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a Terraform working directory
    output             Read an output from a state file
    plan               Generate and show an execution plan
    providers            Prints a tree of the providers used in the configuration
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version
    workspace            Workspace management
```

```
All other commands:
    0.12upgrade        Rewrites pre-0.12 module source code for
v0.12
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    push               Obsolete command for Terraform Enterprise
legacy (v1)
    state              Advanced state management
```

Among all these commands, we will be mostly using commands like:
validate, fmt, plan, apply , graph
validate: In simple terms, it's check all the terraform files and confirm all files are proper , executable/deployable by terraform apply command
fmt: Just beautify the codes
plan: This will tell you what resources going to get affected and how.
apply: It will make all changes required to match the desired state of resources, but always run plan command before apply command
graph: It visually show all the resource relations

Demo Reference

In Terraform writing codes depends highly depends on intended providers, so for the demo I will show using AWS as provider, and I will below use repository in demo:

https://github.com/terraform-providers/terraform-provider-aws/tree/master/examples/two-tier

I will thoroughly explain what used where and why, so that it can help you to run your own first Terraform application.

Prerequisite:

Install terraform : https://learn.hashicorp.com/terraform/getting-started/install.html e.g. for mac run: brew install terraform

Generate ssh keys, e.g. for mac: ssh-keygen -t rsa (similarly look in google how to generate for your os, it’s really simple ,name the key as terraform) and add identity to keychain: ssh-add -K ~/.ssh/terraform_rsa

For this demo I am using AWS, but if you are lookings for GCP, you can refer to this:
https://github.com/terraform-providers/terraform-provider-google/tree/master/examples

Similarly for Azure:

https://github.com/terraform-providers/terraform-provider-azurerm

All the provider codes are here and check providers repo and check examples folder to find example of your preferred provider and test:
https://github.com/terraform-providers or https://www.terraform.io/docs/providers/index.html

After you are done with your tests, don’t forget to clean up resources, just run: “terraform destroy” , it will remove resources so you will be no more get billed for anything.

This Video Link: https://youtu.be/nzUF-u6Ij5M

If you have not subscribed yet, subscribe now to get more videos like this and also make sure to click on Bell icon to get notification: **https://www.youtube.com/sandipdas-official?sub_confirmation=1**

Good luck!

I hope you'll use this knowledge and build awesome solutions.

If any issue contact me in Linkedin: https://www.linkedin.com/in/sandip-das-developer/