May13th, 2023

#GlobalAzureAthens

#GlobalAzureAthens

Deploy resources on Azure using IaC (Azure Terraform)

#GlobalAzureAthens

Who Am I

George – Chrysovalantis Grammatikos

Solutions Architect - Microsoft Most Valuable Professional Azure

Microsoft TechNet | Microsoft | Tech Community | Microsoft Developer Blogs

CloudOpsZone | Microsoft | Learn Q&A | Experts Exchange

MVP Microsoft Most Valuable Professional

Microsoft Certified: DevOps Engineer Expert | Azure Security Engineer Associate | Identity and Access Administrator Associate | Microsoft 365 Certified: Teams Administrator Associate

Microsoft Certified: Azure Administrator Associate | Azure Solutions Architect Expert | Cybersecurity Architect Expert | Microsoft Certification Exam MB-200

#GlobalAzure

Dear Global Azure Athens 2023 sponsors,
your support made all the difference — **thank you!**

kaizen GAMING

uni systems

Microsoft

Info Quest TECHNOLOGIES

Office line
envision . empower . evolve

CUBE

CANDI
ADVANCED BUSINESS AND DIGITAL SOLUTIONS

BlueStream SOLUTIONS

INFOLAB
Enterprise Training

Code.Hub

SIGNAL

#GlobalAzureAthens

Agenda

- Infrastructure as Code (IaC)
  - What is Infrastructure as Code (IaC)
  - Infrastructure as Code (IaC) Tools
- Terraform
  - What is Terraform?
  - Terraform Deployment Tools
  - Terraform Workflow
  - Main Terraform Commands
  - Terraform Benefits
  - Terraform Terminology
  - Architecture diagram
- How the Terraform code looks
- Demo

Infrastructure as Code (IaC)

What is Infrastructure as Code (IaC)

" With infrastructure as code (IaC), infrastructure, such as networks, virtual machines, load balancers, and connection topologies, is defined and deployed using DevOps methodologies and versioning. "

Infrastructure as Code (IaC) Tools

Pulumi

AWS CloudFormation

puppet

HashiCorp Terraform

Terraform

What is Terraform?

Terraform creates and manages resources on cloud platforms and other services through their application programming Interfaces (APIs).

Providers enable Terraform to work with virtually any platform or service with an accessible API.

Terraform Deployment Tools

Azure PowerShell

Windows Terminal

Windows CMD

Bash

Azure DevOps pipelines

Visual Studio Code

Azure Cloud Shell

Terraform Workflow

Write

Terraform configuration files define the infrastructure resources to be created or managed with Terraform.

Plan

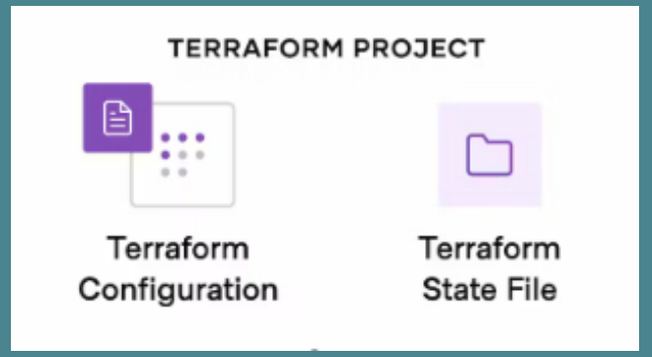
The "terraform plan" command lets you see how Terraform will modify your infrastructure after it is initialized.

Apply

With "terraform apply", you can apply the changes to your infrastructure once the plan has been reviewed and approved.

Write

Define infrastructure in configuration files



Plan

Review the changes Terraform will make to your infrastructure

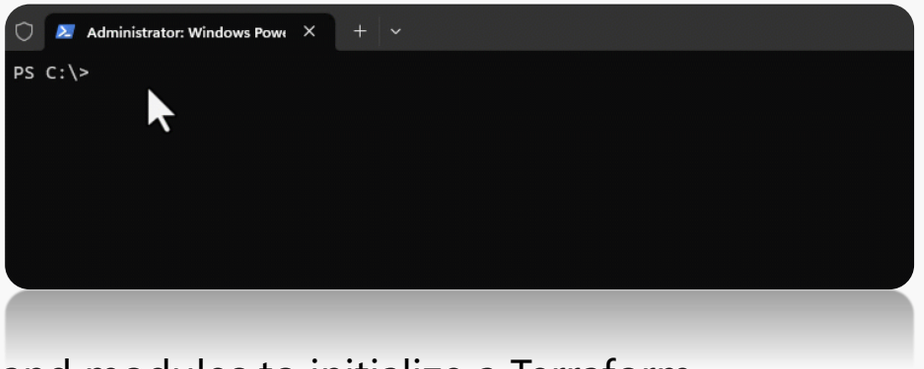
```
$ terraform plan

...

Terraform will perform
the following actions
```

Apply

Terraform provisions your infrastructure and updates the state files

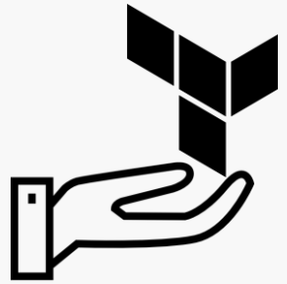
DATADOG aws 1000+ PROVIDERS

Main Terraform Commands



> terraform init	It downloads and installs any necessary plugins and modules to initialize a Terraform working directory.
> terraform plan	This command, generates an execution plan showing what will happen when you apply your configuration. In other words, it is a preview of the changes to your infrastructure without having to make them.
> terraform apply	It applies the changes to an infrastructure. Depending on the desired state, resources are created, updated, or deleted.
> terraform validate	It validates the terraform configuration files to check for any syntax errors.
> terraform destroy	This command, destroys the infrastructure resources managed by terraform.

Terraform Benefits



- Cloud platform agnostic

  Being agnostic cloud platform terraform means it can be used to manage infrastructure resources across multiple cloud providers, such as Microsoft Azure, Google Cloud, AWS, and others.

- Agentless

  It doesn’t require additional software. Nothing is needed to install. The agentless approach in Terraform simplifies the infrastructure management process, increases flexibility, improves security, and reduces costs.

- Ease deployment

  Organizations can deploy infrastructure resources faster, more efficiently, and consistently while reducing errors and automating infrastructure deployments.

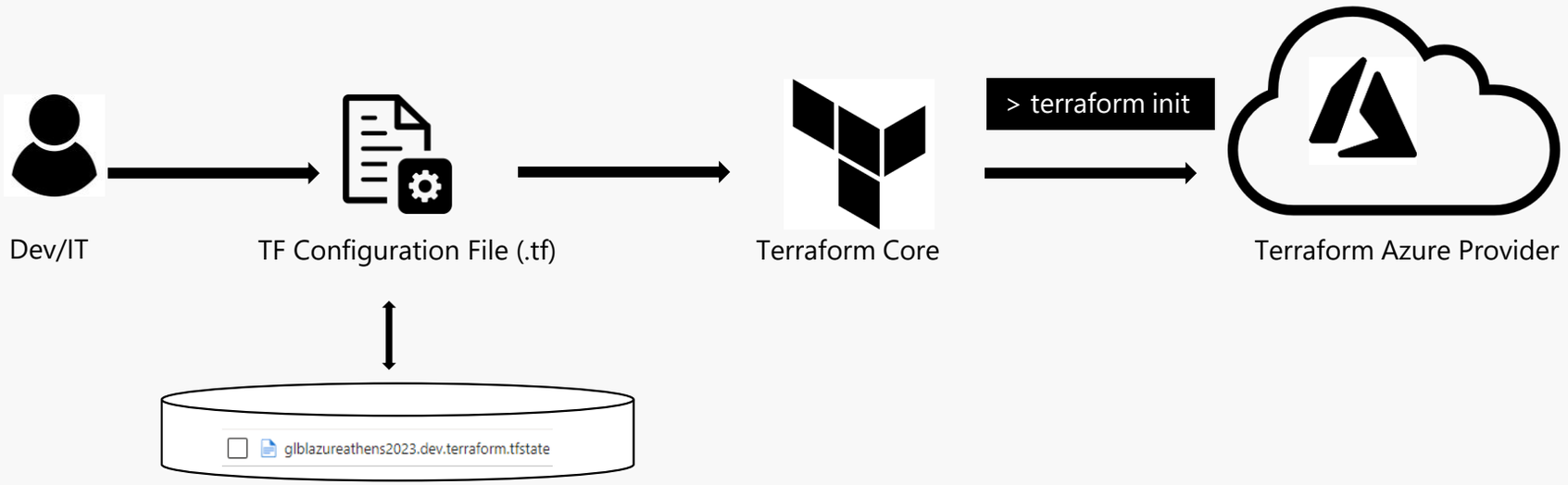
- Cost Estimation

  The cost estimation benefits of Terraform can help organizations better manage their infrastructure spending, create more accurate budgets, optimize resource usage, and choose the most cost-effective cloud provider for their needs.

Terraform Terminology

Resource:	Resources are the most important element in the Terraform language. Each resource block describes one or more infrastructure objects, such as virtual networks, compute instances, or higher-level components such as DNS records.
Provider:	A plugin that defines the APIs and resources available for a specific cloud platform or service. For Azure, the provider is the "azurerm" provider.
Module:	Modules are containers for multiple resources that are used together. A module consists of a collection of .tf and/or .tf.json files kept together in a directory.
Input variable:	Input variables let you customize aspects of Terraform modules without altering the module's own source code.
Output variable:	Output values make information about your infrastructure available on the command line and can expose information for other Terraform configurations to use. Output values are similar to return values in programming languages.
Data Source:	Data sources allow Terraform to use information defined outside of Terraform, defined by another separate Terraform configuration, or modified by functions.
State:	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. This state is stored by default in a local file named "terraform.

Architecture Diagram



How the terraform code looks?

```
terraform{
            required_providers {
                        azurerm = {
                        source = "hashicorp/azurerm"
                                    }
                                }
         }

provider "azurerm" {
           features {}
                   }
resource "azurerm_resource_group" "{resource group}" {
            name="{resource group}"
            location = "westeurope"
}

resource "azurerm_storage_account" "{storage account}" {
            name = "{storage account}"
            resource_group_name = azurerm_resource_group.{resource group}.name
            location = azurerm_resource_group.{resource group}.location
            account_tier = "Standard"
            account_replication_type = "LRS"
}

resource "azurerm_storage_container" "images" {
            name = "images"
            storage_account_name = azurerm_storage_account.{storage account}.name
            container_access_type = "private"
}
```

Demo

Useful links

- https://learn.microsoft.com/en-us/azure/developer/terraform/
- https://github.com/hashicorp/terraform-provider-azurerm
- https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs
- https://developer.hashicorp.com/terraform/cloud-docs/cost-estimation/azure
- https://github.com/topics/terraform-cost-estimation

#GlobalAzureAthens

Thank you

Q & A

Call to Action

- Azure Batch Shipyard @ https://github.com/Azure/batch-shipyard
- Cognitive toolkit @ https://cntk.ai
- Learn more about Azure N-Series on Channel 9
- Re-visit Connect on Channel 9.

Please evaluate !

A big **thank you** to our sponsors!

https://bit.ly/GA23Evaluation

#GlobalAzureAthens