Microsoft

IaC Options

Azure Database for MySQL – Flexible Server

June 2022

Hosts: Avnish Rastogi / Jim Toland / Brian Hitney

Azure Data Academy: https://aka.ms/ada
Feedback: https://aka.ms/ada-feedback

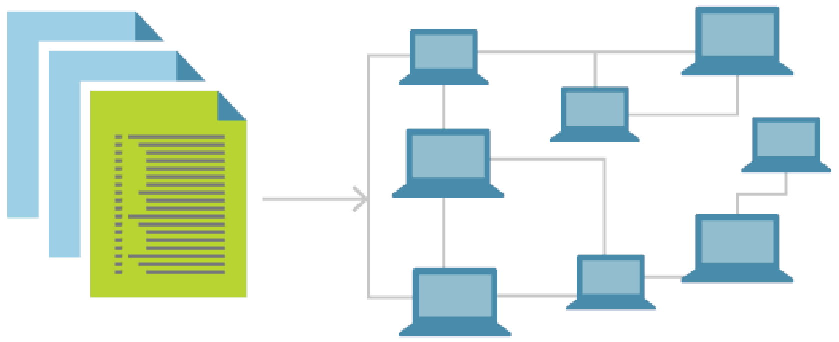
Azure Database for MySQL
Live Roadmap and Q&A Session

- Join us for a LIVE Roadmap and Q&A session on July 14, 2022 1pm ET
- Get an inside look at the Azure Database for MySQL Roadmap and schedule
- Ask questions
- Find other sessions in the series: https://aka.ms/ada
- Register at https://aka.ms/mysqlroadmap

This session is intended for Microsoft Partners under NDA. When registering, be sure to use your company e-mail address. This session will not be recorded.

What is Infrastructure as Code (IaC)?

- IaC is the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning as a DevOps team uses for source code.
- An IaC model generates the same environment every time it is applied.
- IaC is a key DevOps practice that is used in conjunction with continuous delivery.



- Azure provides native support for IaC via Azure Resource Manager, PowerShell, the Azure CLI, and Bicep.
- Managing automated infrastructure in Azure is also supported by third-party solutions such as Terraform, Ansible, Chef, Pulumi, etc.
- Different third-party solutions support different, and often multiple, file formats, such as YAML, JSON, XML, HCL, etc.

IaC options in Azure

Option	Description
Azure PowerShell	• An extension of Windows PowerShell based on the .NET Standard; works with PowerShell 7.0.6 LTS and PowerShell 7.1.3 or higher on all platforms (Windows, macOS, and Linux). • Contains cmdlets for performing both control plane and data plane operations in Azure by making REST API calls to the Azure API.
Azure CLI	• A cross-platform command-line program supporting Windows, macOS, and Linux. • Uses shell on Windows, or bash on macOS and Linux.
ARM	• Native solution for Azure IaC. • ARM templates define resource details, including names, locations, availability zones, security settings, and networks. • Use Visual Studio, Visual Studio Code, or Azure Portal • ARM templates are in JSON
Bicep	• In a way, a revision to ARM, using the same core functionality and runtime. • Bicep files compile to JSON, and then JSON is sent to Azure for deployment.

1st Party vs 3rd Party IaC solution

Azure Resource Manager vs Terraform

- If your organization is pursuing a multi-cloud strategy, a cloud agnostic tool such as Terraform might be a good fit.
- Azure Resource Manager (ARM), an Azure native product, works with the latest Azure features as soon as Microsoft releases them.
- Terraform, however, is open source and supports 100+ providers.
- ARM templates and Terraform provide different methods for variables, conditions, and internal logic.
- It takes a significant amount of work to write the ARM infrastructure in a different provider and achieve identical results.

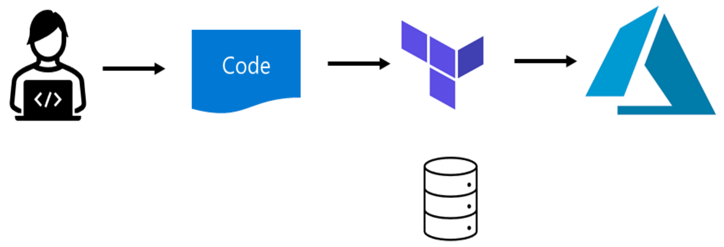
ARM	Terraform
JSON	HCL*
Parameters	Variables
Variables	Local Variables
Resources	Resources
Functions	Functions
Nested Templates	Modules
Explicit Dependency	Automatic Dependency
Refer by reference or resourceid	Refer by resource or data source

*HashiCorp Configuration Language; easy to add comments, more human readable and forgiving syntax

[Terraform on Azure documentation - Articles, samples, references, and resources - Terraform]

What is Terraform?

- An open-source infrastructure as code (IaC) software that provides a consistent CLI workflow to manage hundreds of cloud services.
- Terraform can manage infrastructure on multiple cloud platforms including Azure, AWS, GCP, etc.

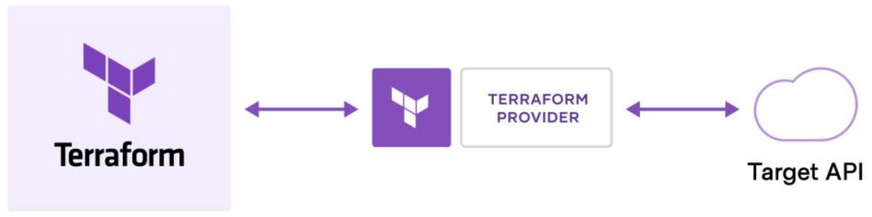


- A provider is responsible to provide APIs for their resources (VMs, DBs, etc.)
- Terraform code, written in HCL (Hashicorp Configuration Language), automatically identify dependencies between resources to create or destroy them in the correct order.
- A terraform module is a standard interface for creating resources by providing input and returning outputs.
- Terraform modules can call each other which greatly simplifies configurations.
- Terraform works by building a graph database that provides operators with insight into resource dependencies.

Using Terraform with Azure

How does Terraform Work?

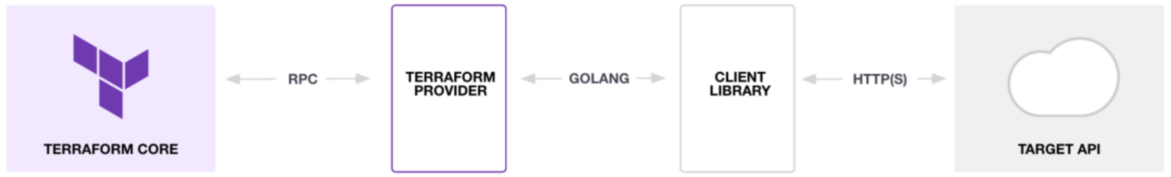
- **Write:** You define resources, which may be across multiple cloud providers and services. For example, you might create a configuration to deploy an application on virtual machines in a Virtual Private Cloud (VPC) network with security groups and a load balancer.



- **Init:** Initialize working directory containing Terraform configuration files.
- **Plan:** Terraform creates an execution plan describing the infrastructure it will create, update, or destroy based on the existing infrastructure and your configuration.
- **Apply:** On approval, Terraform performs the proposed operations in the correct order, respecting any resource dependencies. For example, if you update the properties of a virtual network and change the number of virtual machines, Terraform will recreate the virtual network before scaling the virtual machines.

Terraform Providers

- Terraform has two important components: Terraform Core and Terraform Plugins.
- Terraform relies on plugins ("providers") to interact with cloud providers and other APIs



- Terraform Registry is the main directory of publicly available Terraform providers.
- A provider documentation in the registry is versioned.
- Resources from a given provider requires below information in the configuration file…

Provider Requirements

```
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 2.65"
    }
  }
  required_version = ">= 1.1.0"
}
```

Provider Configuration

```
provider "azurerm" {
  features {}

  subscription_id = "5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30"
}
```

- Dependency Lock File automatically created each time you run *terraform init*

Terraform Variables, Outputs and Locals

- Terraform language include Input, Output and Locals block.
- Each input variable is declared using the "*variable*" block.
- Variable name can be any valid identifier except *source, version, providers, count, for_each, lifecycle, depends_on*, and *locals*

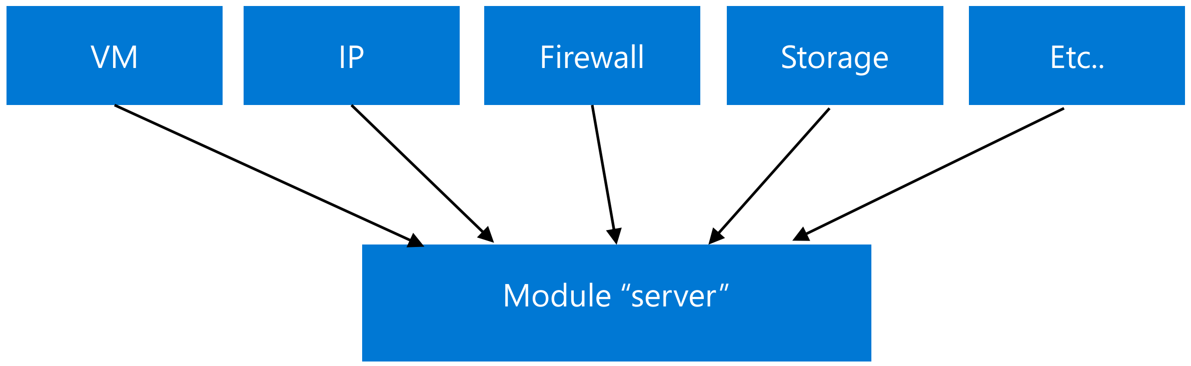
```
variable "resource_group_v" {
    description = "Azure Resource Group"
    type = string
    default = "avrastog-terra-rg"
    nullable = false
    sensitive = false
    validation {
        condition     = length(var.resource_group_v) > 4 && substr(var.resource_group_v, 0, 8) == "avrastog"
        error_message = "The resource group value must start with \"avrastog\"."
    }
}
```

- Variables are referenced as an attribute on an object named "*var*"
- Set variables individually with the -*var* command line option or variable definition "*.tfvars*" file(s)
  - *terraform apply -var "var1=<value>", -var "var2=<value>"*
- Another option to set a variable is using TF_VAR_<variable_name> environment variable.
- Each output variable must be declared using an *output* block.
- A set of related values can be declared in a single "*locals*" block.

Terraform Module

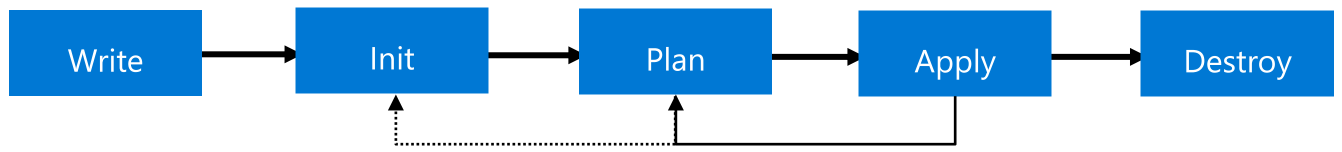
- Modules are the primary way to package and reuse resource configurations with Terraform.
- A module may include a set of resources.
- A set of Terraform configuration files in a single directory. Simple configurations may include:
  - *.tf - contains set of configuration file(s).
  - variables.tf - contains variables definition.
  - outputs.tf - contains output definition.
- "To call a module" means to use it in the configuration file.

```
module "server" {
    count          = 5
    source         = "./module_server"
    region         = var.region_v
}
```



- Other files to be aware of:
  - terraform.tfstate and terraform.tfstate.backup: Contain Terraform state, and how Terraform keeps track of the relationship between the configuration and the infrastructure provisioned.
  - .tfstate file won't appear until you run a *terraform plan* command.

Terraform Workflow



- *terraform init* to initialize working directory to download and install the *plugins* for each provider.
- A core Terraform workflow has 3 steps:
  - Write – Author Infrastructure as Code (IaC)
  - Plan – Preview changes before applying
    - *terraform plan*
    - *terraform plan –out=<planfile>*
  - Apply – Provision reproducible infrastructure
    - *terraform apply*
    - *terraform apply <planfile> -var "var1=<value>", -var "var2=<value>"*
- Terraform plan is essentially a dry run on the configuration to provide detailed information on what the deployment will look like.
  - A newly created resource will have + while a Destroyed resource will have -
- Terraform apply will deploy the specified resources and create a *state file* ".tfstate"

Terraform CLI

- Command line interface to Terraform via the *terraform* command, which accepts a variety of main commands such as …
  - *init*
  - *validate*
  - *plan*
  - *apply*
  - *destroy*
- Other terraform commands includes …
  - *fmt*
  - *graph*
  - *output*
  - *show*
  - *state*
  - *taint – mark a resource instance as not fully functional*

Terraform Azure DB for MySQL

Single Server

Resource name	Manages a…
azurerm_mysql_server	MySQL server
azurerm_mysql_database	MySQL database within a MySQL server
azurerm_mysql_server_key	Customer Managed Key for a MySQL server
azurerm_mysql_virtual_network_rule	MySQL virtual vetwork rule
azurerm_mysql_firewall_rule	Firewall rule for a MySQL server

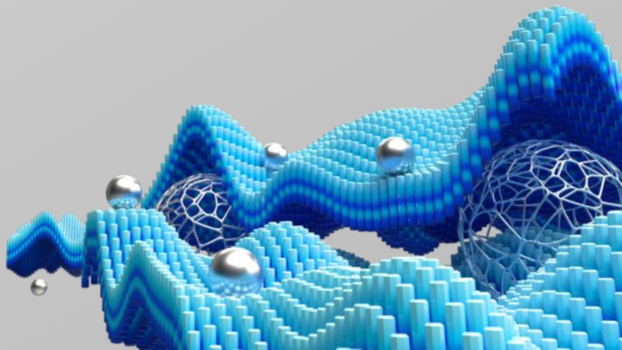
Note: azurerm_mysql_server is a data source with details about an existing MySQL server.

Flexible Server

Resource name	Manages a…
azurerm_mysql_flexible_server	MySQL flexible server
azurerm_mysql_database	MySQL database within a MySQL server
azurerm_mysql_flexible_server_firewall_rule	Firewall rule for a MySQL flexible server

Microsoft

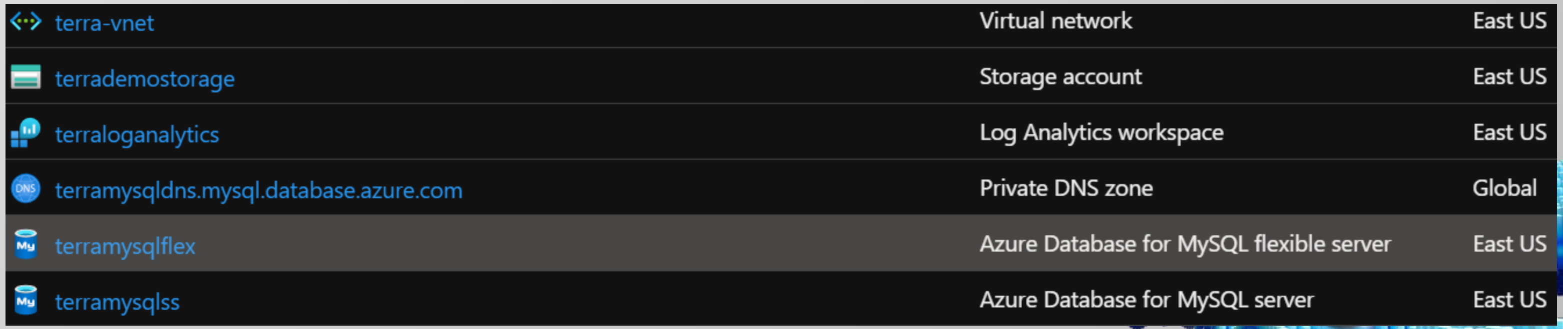
Terraform Demo



Azure MySQL Office Hours Terraform Demo

Terraform Azure DB for MySQL Demo

- Define a provider (provider.tf)
- Input and Output Variables (variables.tf and outputs.tf)
- Provision Azure Resource Group (rg.tf)
- Provision Azure VNet, Delegated Subnet, Azure DNS, and Azure Virtual Network Link (network.tf)
- Provision Azure Storage Account and Azure Log Analytics Workspace (storage.tf)
- Provision Azure DB for MySQL Single Server, configure firewall and provision a database (mysqlss.tf)
- Provision Azure DB for MySQL Flexible Server (HA/Non HA, RR, DB, Public and Private, etc.) (mysqlflex.tf)



terra-vnet	Virtual network	East US
terrademostorage	Storage account	East US
terraloganalytics	Log Analytics workspace	East US
terramysqldns.mysql.database.azure.com	Private DNS zone	Global
terramysqlflex	Azure Database for MySQL flexible server	East US
terramysqlss	Azure Database for MySQL server	East US

Microsoft

Azure Database for MySQL
Live Roadmap and Q&A Session

- Join us for a LIVE Roadmap and Q&A session on July 14, 2022 1pm ET
- Get an inside look at the Azure Database for MySQL Roadmap and schedule
- Ask questions
- Find other sessions in the series: https://aka.ms/ada
- Register at https://aka.ms/mysqlroadmap

This session is intended for Microsoft Partners under NDA. When registering, be sure to use your company e-mail address. This session will not be recorded.