INFRASTRUCTURE-AS-CODE (IAC)

USING TERRAFORM
(BEGINNER EDITION)

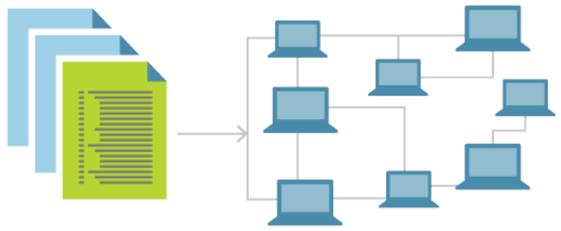
Omaha Azure User Group

Adin Ermie
Cloud Solution Architect
(Azure Apps & Infra)
Microsoft

AGENDA

- Benefits of / Tools for Infrastructure-as-Code (IaC)
- What is Terraform and why do people love it?
- Terraform basics
  - Commands, workflow, resource creation, file structure
- Azure Resource Manager (ARM) vs Terraform
- Best practices
- Resources for learning and certification

BENEFITS OF INFRASTRUCTURE-AS-CODE (IAC)

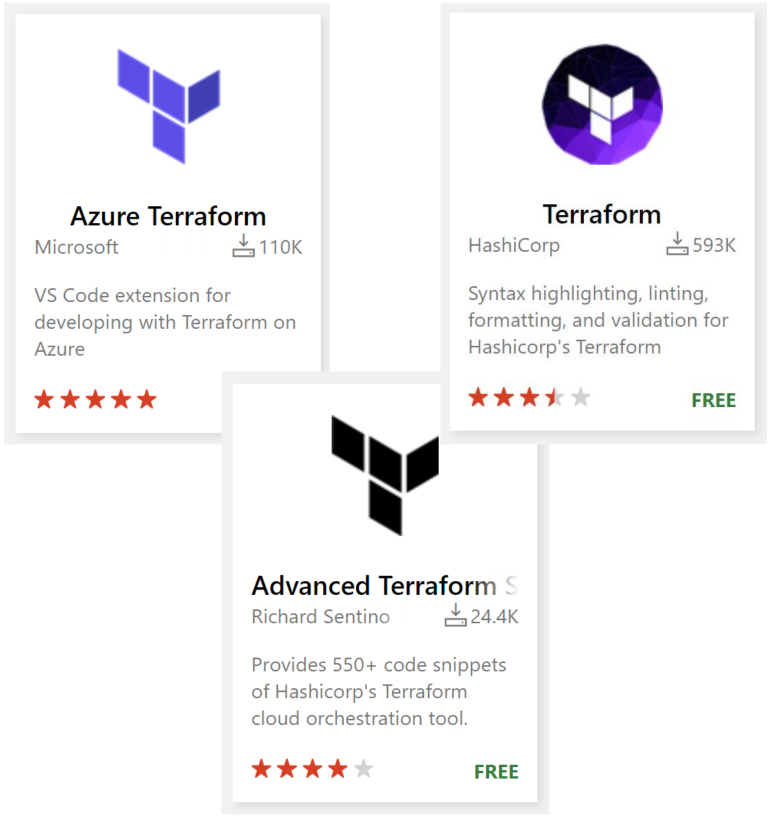


- ✓ Reproducible environments
- ✓ Automation – CI/CD
- ✓ Trackable – GitHub
- ✓ Language - HCL
- ✓ Workflow
- ✓ Providers

- ✗ Apply same configuration across clouds

TOOLS FOR INFRASTRUCTURE-AS-CODE (IAC)

- Most popular code editor is Visual Studio Code (aka VS Code)
  - Download at code.visualstudio.com
- Visual Studio Code Extensions
  - Lets you add languages, debuggers, and tools to your installation to support your development workflow
  - Recommended:
    - Azure Terraform by Microsoft
    - Terraform by Mikael Olenfalk (now owned by HashiCorp)
    - Advanced Terraform Snippets Generator by Richard Sentino

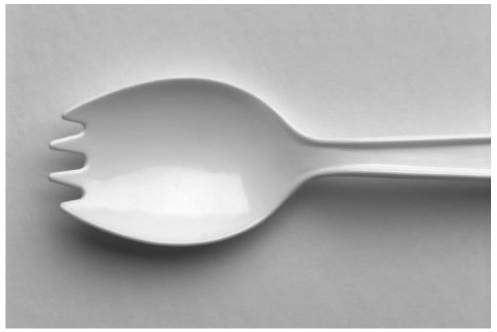

Azure Terraform
Microsoft 110K
VS Code extension for developing with Terraform on Azure
★★★★★

Terraform
HashiCorp 593K
Syntax highlighting, linting, formatting, and validation for Hashicorp's Terraform
★★★½☆ FREE

Advanced Terraform S
Richard Sentino 24.4K
Provides 550+ code snippets of Hashicorp's Terraform cloud orchestration tool.
★★★★☆ FREE

WHAT IS TERRAFORM?

- A templating language created by HashiCorp called HashiCorp Configuration Language (HCL)
  - Written in Go Lang
- A tool that can be used to orchestrate the provisioning of:
  - Public clouds (Azure, AWS, GCP, Oracle, Alibaba)
  - On-premises (VMware)
  - Other (Cisco, GitHub, GitLab, New Relic, Okta, Rabbit MQ)
- Uses State files (more on this later)
- Is NOT used for configuration
  - PowerShell Desired State Configuration (DSC), Chef, Puppet, Ansible



WHY PEOPLE LOVE TERRAFORM?

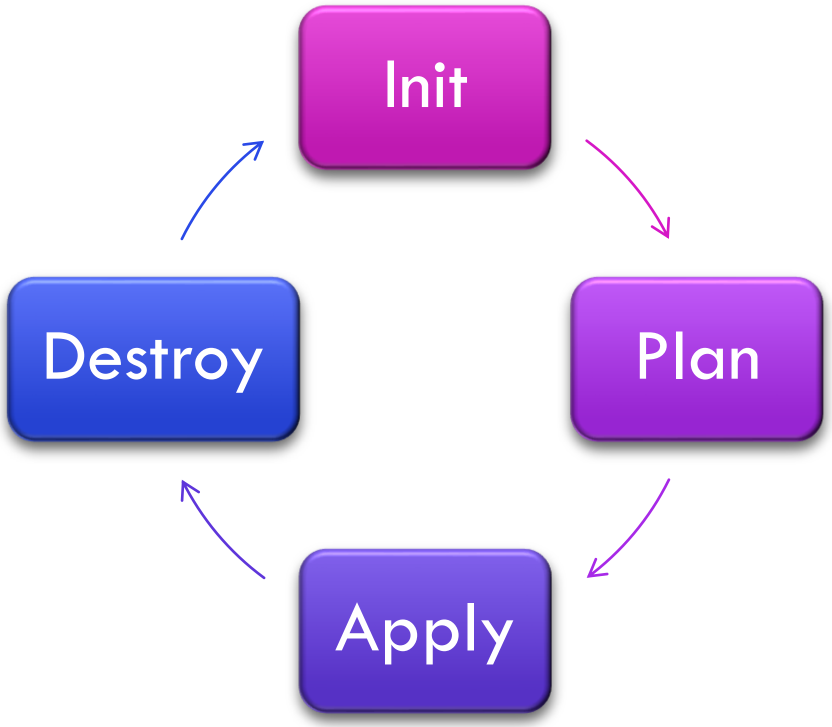
- Clean and easy code to write and maintain
- Fully declarative configuration
- Version control on infrastructure
- Implicit dependencies management – explicit can be forced
- Ecosystem of providers and skilled personnel

TERRAFORM BASICS

SO, HOW DOES THIS WORK?

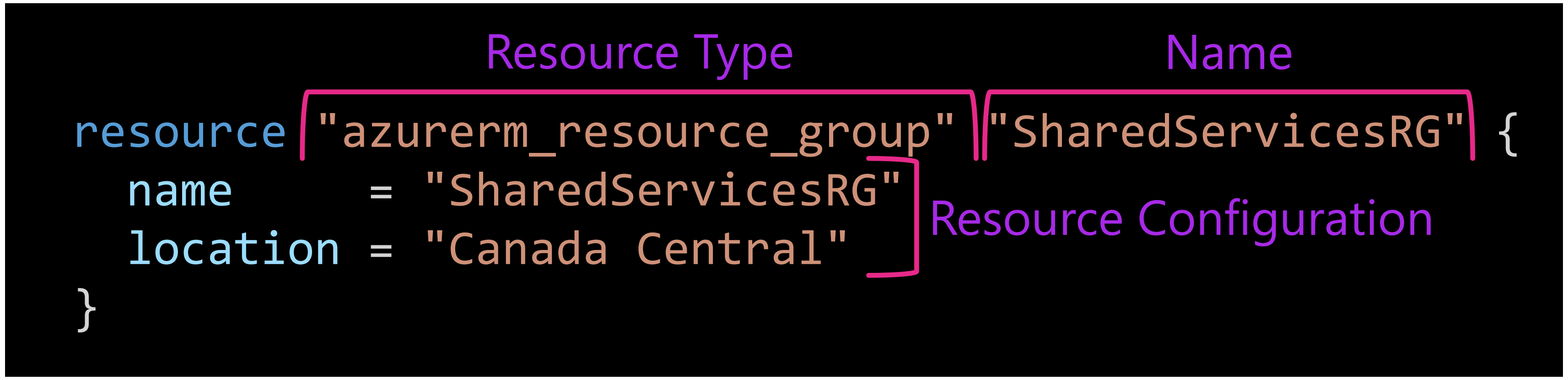
TERRAFORM COMMANDS/WORKFLOW

- Terraform **init**
  - Initializes the current working directory
- Terraform **plan**
  - Execution plan to validate against existing environment
- Terraform **apply**
  - Deploys and updates resources
- Terraform **destroy**
  - Removes all resources defined in a configuration



BASIC RESOURCE CREATION

- Resource Type
  - Required provider
- Name
  - Internal (to Terraform) name
- Configuration
  - Deployment details



```
resource "azurerm_resource_group" "SharedServicesRG" {
  name     = "SharedServicesRG"
  location = "Canada Central"
}
```

TERRAFORM FILE STRUCTURE

```
terraform {
  required_version = ">=0.12.0"
  backend "azurerm" {
    resource_group_name  = "tstate"
    storage_account_name = "tstate123"
    container_name       = "tstate"
    key                  = "terraform.tfstate"
  }
}

provider "azurerm" {
  version = ">=2.0.0"
  subscription_id = "<<REMOVED>>"
  client_id       = "<<REMOVED>>"
  client_secret   = "<<REMOVED>>"
  tenant_id       = "<<REMOVED>>"
}
```

```
resource "azurerm_resource_group" "example" {
  name = var.resource_group_name
  location = var.location
}

resource "azurerm_storage_account" "example" {
  name = "storageaccountname"
  resource_group_name = azurerm_resource_group.example.name
  location = azurerm_resource_group.example.location
  account_tier = "Standard"
  account_replication_type = "GRS"
  tags = {
    environment = "staging"
  }
}
```

BACKENDS

- Determines how state is loaded and how an operation such as apply is executed
- By default, Terraform uses the "local" backend
- Benefits of backends:
  - Working in a team
    - Store state remotely and protect state with locks to prevent corruption
  - Keeping sensitive information off disk
    - Retrieved on demand and only stored in memory
  - Remote operations
    - Remote execution

```
terraform {
  required_version = ">=0.12.0"
  backend "azurerm" {
    resource_group_name  = "tstate"
    storage_account_name = "tstate123"
    container_name       = "tstate"
    key                  = "terraform.tfstate"
  }
}
```

STATE

- State keeps track of the all managed resources and their associated properties with current values.
- Essential for managing changes to infrastructure over time
- Preserve the state file for the entire life cycle of the resources
  - You can create a separate state file per resource group, application, shared service (ie. core networking)
  - Terraform Workspaces should also be used to separate application and environment boundaries
- Recommended to use a remote backend to save state in centralized, secure storage
  - Example: Storage account, Terraform Cloud, Terraform Enterprise, Artifactory, Consul
- You must initialize the Terraform State
  - This is what terraform init does

IMPORTANT!
Secrets (like usernames/passwords, access keys/tokens, etc.) can be written to your state file!

PROVIDERS

- The provider block is used to configure the named provider
- Is responsible for creating and managing resources, and for all other interactions including authentication
- The version argument is optional, but recommended

```
provider "azurerm" {
  version = ">=2.0.0"
  subscription_id = "<<REMOVED>>"
  client_id       = "<<REMOVED>>"
  client_secret   = "<<REMOVED>>"
  tenant_id       = "<<REMOVED>>"
}
```

VARIABLES

- Parameterize the configurations
- If no value is assigned to a variable and the variable has a default key in its declaration, that value will be used for the variable
- Can be provided…
  - Within the Terraform template
  - Within its own Terraform template file
  - Within a .TFVARS files
  - Through command-line
  - Through Environment variables

```
variable "location" {
  type        = string
  default     = "westus"
  description = "Specify a location see: az acc
ount list-locations -o table"
}


resource "azurerm_resource_group" "example" {

  name = var.resource_group_name

  location = var.location

}
```

NOTE!
Variables have precedence
1. Command-line
2. From a file
3. Environment variables
4. UI Input

DEPENDENCIES

- *Implicit* dependencies, which Terraform and the provider determine automatically based on the configuration
- *Explicit* dependencies, which you define using the depends_on meta-argument

BONUS!
Terraform v0.13.0 beta
Modules will support…
count, for_each, and depends_on

```
resource "azurerm_resource_group" "example" {
  name = var.resource_group_name
  location = var.location
}

resource "azurerm_storage_account" "example" {
  name = "storageaccountname"
  resource_group_name = azurerm_resource_group.example.name
  location = azurerm_resource_group.example.location
  account_tier = "Standard"
  account_replication_type = "GRS"
  tags = {
    environment = "staging"
  }
}
```

OUTPUTS

- Used to organize data to be easily queried and shown back to the Terraform user
- Data is outputted when apply is called
- Outputs can be queried after a run using the terraform output <<output name>> command

```
output "SharedServices-RGName" {
  value = azurerm_virtual_network.SharedServicesVNET.*.resource_group_name
}

output "SharedServices-VNet-Name" {
  value = azurerm_virtual_network.SharedServicesVNET.*.name
}

output "SharedServices-VNet-ID" {
  value = azurerm_virtual_network.SharedServicesVNET.*.id
}
```

Demo

ARM VS TERRAFORM

PEANUT BUTTER / JELLY TIME!

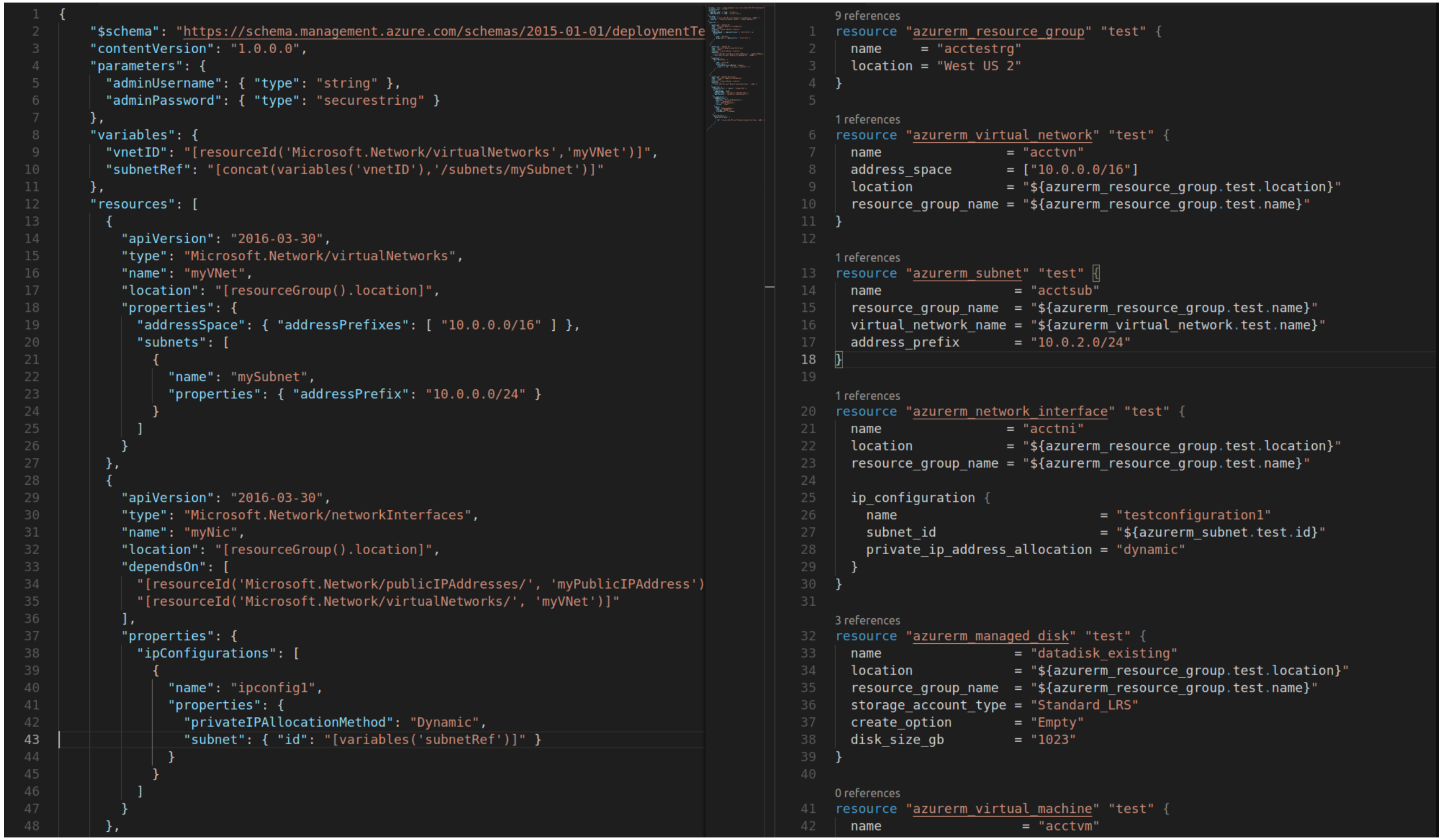
FEATURE COMPARISON

Feature	**ARM**	**Terraform**
Infrastructure as Code (IaC)	Yes	Yes
Readability	JavaScript Object Notation (JSON)	HashiCorp Configuration Language (HCL)
Execution plans	Yes (‘What-If’ Preview)	Yes
Dependencies	Yes (Explicit)	Yes (Implied)
Multi-Cloud	No	**Yes**
Configuration	Inline ‘DeploymentScript’ (PowerShell) Preview	Provisioners / other Providers
Rollback State	Yes – deploy prior template / rollback	Yes – maintains state
Azure Preview features	Yes	Yes – inline ARM snippets
KeyVault support	Yes	Yes (also integrates with HashiCorp Vault)
Corrupted State	**State not needed**	Can be an issue
Supports DevOps	Yes	Yes
Cost / Support	Free, uses Azure support	Free / Paid (purchase support)
Parallel deployments	Yes	Yes

FEATURE COMPARISON (CONTINUED)

Feature	**ARM**	**Terraform**
Runs “Locally”	ARM template is uploaded / deployed in Azure	Terraform uses REST calls via a client machine
Delete resource in portal and not worry about state	**Yes**	No
Support Comments	Via an Attribute	**Yes** including block comments
Speed	Can take a while	Can be fast since it can deploy just a single item based upon its plan
Math Functions	Yes	Yes
Count / Loops	Yes	Yes
Sub-Templates/Modules	Yes – Linked Templates	Yes – Modules
Deploy to multiple resource groups	Requires many template	Can be done in one template
Reference existing resources	Variable with resource ID path	“data” resource type
Reverse Engineer resources	Export and Visual Studio	Object by Object through importing

SYNTAX DIFFERENCES



```json
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTe
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUsername": { "type": "string" },
    "adminPassword": { "type": "securestring" }
  },
  "variables": {
    "vnetID": "[resourceId('Microsoft.Network/virtualNetworks','myVNet')]",
    "subnetRef": "[concat(variables('vnetID'),'/subnets/mySubnet')]"
  },
  "resources": [
    {
      "apiVersion": "2016-03-30",
      "type": "Microsoft.Network/virtualNetworks",
      "name": "myVNet",
      "location": "[resourceGroup().location]",
      "properties": {
        "addressSpace": { "addressPrefixes": [ "10.0.0.0/16" ] },
        "subnets": [
          {
            "name": "mySubnet",
            "properties": { "addressPrefix": "10.0.0.0/24" }
          }
        ]
      }
    },
    {
      "apiVersion": "2016-03-30",
      "type": "Microsoft.Network/networkInterfaces",
      "name": "myNic",
      "location": "[resourceGroup().location]",
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses/', 'myPublicIPAddress')
        "[resourceId('Microsoft.Network/virtualNetworks/', 'myVNet')]"
      ],
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": { "id": "[variables('subnetRef')]" }
            }
          }
        ]
      }
    },
```

```hcl
9 references
resource "azurerm_resource_group" "test" {
  name     = "acctestrg"
  location = "West US 2"
}

1 references
resource "azurerm_virtual_network" "test" {
  name                = "acctvn"
  address_space       = ["10.0.0.0/16"]
  location            = "${azurerm_resource_group.test.location}"
  resource_group_name = "${azurerm_resource_group.test.name}"
}

1 references
resource "azurerm_subnet" "test" {
  name                 = "acctsub"
  resource_group_name  = "${azurerm_resource_group.test.name}"
  virtual_network_name = "${azurerm_virtual_network.test.name}"
  address_prefix       = "10.0.2.0/24"
}

1 references
resource "azurerm_network_interface" "test" {
  name                = "acctni"
  location            = "${azurerm_resource_group.test.location}"
  resource_group_name = "${azurerm_resource_group.test.name}"

  ip_configuration {
    name                          = "testconfiguration1"
    subnet_id                     = "${azurerm_subnet.test.id}"
    private_ip_address_allocation = "dynamic"
  }
}

3 references
resource "azurerm_managed_disk" "test" {
  name                 = "datadisk_existing"
  location             = "${azurerm_resource_group.test.location}"
  resource_group_name  = "${azurerm_resource_group.test.name}"
  storage_account_type = "Standard_LRS"
  create_option        = "Empty"
  disk_size_gb         = "1023"
}

0 references
resource "azurerm_virtual_machine" "test" {
  name                  = "acctvm"
```

BEST PRACTICES

SO YOU DO IT RIGHT, THE FIRST TIME!

BEST PRACTICES

- Use remote backends
- Manage Terraform, providers and modules versions
- Use implicit dependencies
- Use modules (custom or from the HashiCorp public registry https://registry.terraform.io)
- Use ARM templates only if you don’t have another choice

RESOURCES

FOR LEARNIN’ STUFF

RESOURCES

aka.ms/AA8J4ON

- Why we use Terraform and not Chef, Puppet, Ansible, SaltStack, or CloudFormation - Yevgeniy Brikman
- Best practices lab (by James Dumont Le Douarec)
- Built-in Terraform functions
- Terraform on Azure documentation (aka.ms/TFHub)
- Microsoft is investing deeply in Terraform on Azure
- Introducing the Azure Terraform Resource Provider
- Top questions about Terraform and Azure
- Adin's personal curated list of Terraform learning resources

Don't forget about these Visual Studio Code (VS Code) extensions:

- Azure Terraform (by Microsoft)
- Terraform (by Mikael Olenfalk)
  - Now owned by HashiCorp!
- Advanced Terraform Snippets Generator by Richard Sentino

CERTIFICATION RESOURCES

- HashiCorp Terraform Certified Associate Preparation Guide (co-authored by Adin)
- Study Guide - Terraform Associate Certification (HashiCorp official)
- Exam Review - Terraform Associate Certification (HashiCorp official)
- Sample Questions - Terraform Associate Certification (HashiCorp official)

HashiCorp Terraform
Certified Associate Preparation Guide

Ned Bellavance
Adin Ermie

THIS IS ME

ADIN ERMIE

- Cloud Solution Architect – Azure Apps & Infra @ Microsoft
  - Azure Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS)
  - Cloud Management & Security
    - Azure Monitor, Azure Log Analytics and Azure Security Center (ASC)
  - Cloud Governance
    - Azure Policy, Blueprints, Management Groups, and Azure Cost Management (ACM)
  - Business Continuity and Disaster Recovery (BCDR)
    - Azure Site Recovery (ASR) / Azure Migrate, and Azure Backup
  - Infrastructure-as-Code (IaC)
    - Azure Resource Manager (ARM), and Terraform
- 5x MVP - Cloud and Datacenter Management (CDM)
- 1x HCA – HashiCorp Ambassador

Adin Ermie
Cloud Solution Architect (Apps & Infra)

Adin.Ermie@outlook.com
@AdinErmie
linkedin.com/in/adinermie
https://AdinErmie.com