

```

core.tf > ...
1  # Resource Groups
2  resource "azurerm_resource_group" "rg-ide" {
3      name      = "rg-baselabv2-${var.region1code}-identity-01"
4      location  = var.region1
5      tags = {
6          Environment = var.environment_tag
7          Function    = "BaseLabv2-identity"
8      }
9  }
10 resource "azurerm_resource_group" "rg-con" {
11     name      = "rg-baselabv2-${var.region1code}-connectivity-01"
12     location  = var.region1
13     tags = {
14         Environment = var.environment_tag
15         Function    = "BaseLabv2-connectivity"
16     }
17 }
18 resource "azurerm_resource_group" "rg-sec" {
19     name      = "rg-baselabv2-${var.region1code}-security-01"
20     location  = var.region1
21     tags = {
22         Environment = var.environment_tag
23         Function    = "BaseLabv2-security"
24     }
25 }
26 # Key Vault
27 resource "random_id" "kv-name" {
28     byte_length = 6
29     prefix      = "kv"
30 }
31 data "azurerm_client_config" "current" {}
32 resource "azurerm_key_vault" "kv1" {
33     name                = random_id.kv-name.hex
34     location            = var.region1
35     resource_group_name = azurerm_resource_group.rg-sec.name
36     enabled_for_disk_encryption = true
37     tenant_id           = data.azurerm_client_config.current.tenant_id
38     soft_delete_retention_days = 7
39     purge_protection_enabled = false
40
41     sku_name = "standard"
42 }

```

An Introduction to Azure Terraform

Festive Tech Calendar

2023



Hello!



Jake Walsh

Senior Solution Architect @ CDW UK

@jakewalsh90
jakewalsh.co.uk

Please note – the views/opinions in this presentation are entirely my own. If in any doubt, please check latest documentation etc!

Please note – the views/opinions in this presentation are entirely my own. This presentation will not be kept updated after the Festive Tech Calendar 2023 (December 2023) – so may be outdated if downloaded afterwards.

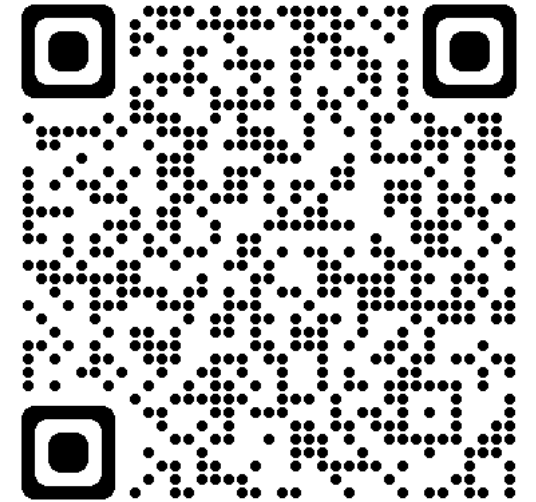
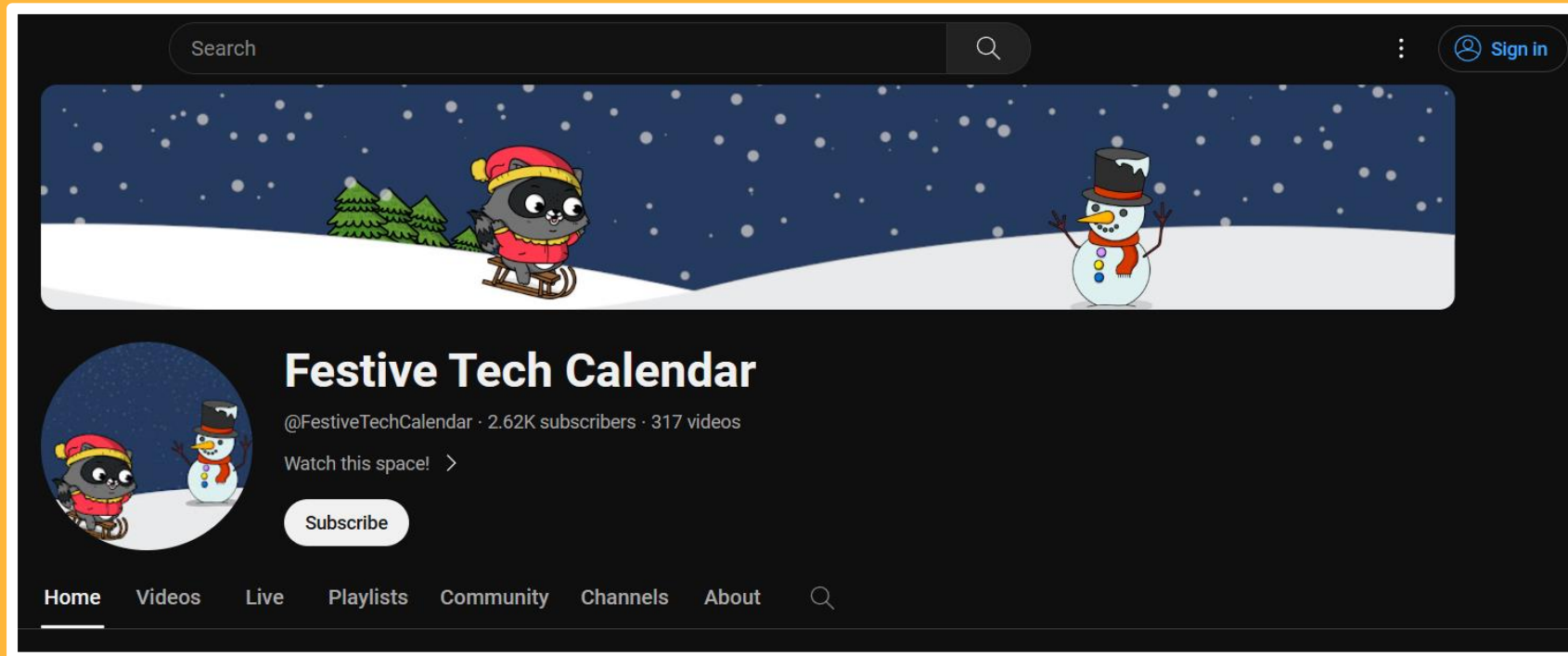
If in any doubt, please check latest documentation and MS Links for updated info!

Festive Tech Calendar

2023

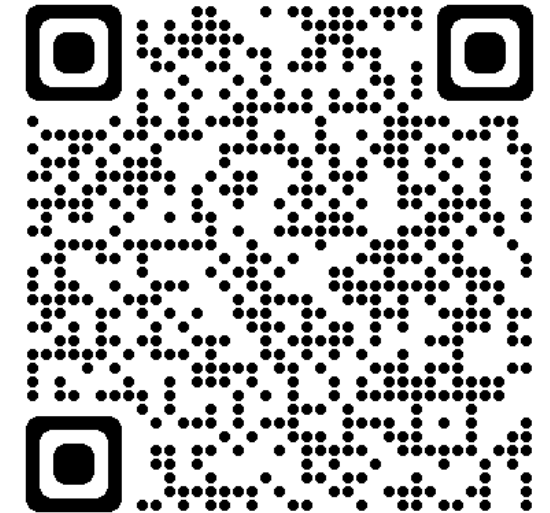


Festive Tech Calendar - YouTube




<https://www.youtube.com/channel/UCJL9wCcmeMBbah4J0uOWIPg>

Raspberry Pi Foundation



<https://www.justgiving.com/page/festive-tech-calendar-2023>

Session Goals

- **What** is Infrastructure as Code – and **why** use it?
- What is Azure **Terraform**? 
- How Terraform Works & Why it's relevant
- **Getting Started** – Installation
- Code Samples / Demo



What is Infrastructure as Code (IAC)?

- ✓ A method of managing and provisioning infrastructure resources via code.
- ✓ In most cases either uses **imperative** or **declarative** code.

↓ ↓
“do this” “build this”

- ✓ Often integrated into version control systems – e.g. Git.
- ✓ Can be edited and managed in most common tools and platforms – e.g. GitHub, Visual Studio Code, Azure DevOps etc.
- ✓ Usually adopted as part of a wider DevOps Strategy.
- ✓ Allows a move away from ClickOps and provides options to version control infrastructure resources.

Imperative

- Defines a task to be carried out
- In this example repeated executions would error – as the VM already exists after 1 run

Azure CLI

```
vmname="myVM"  
username="azureuser"  
az vm create \  
    --resource-group $resourcegroup \  
    --name $vmname \  
    --image Win2022AzureEditionCore \  
    --public-ip-sku Standard \  
    --admin-username $username
```



```

1  # Resource Groups
2  resource "azurerm_resource_group" "rg-ide" {
3      name      = "rg-baselabv2-${var.region1code}-identity-01"
4      location = var.region1
5      tags = {
6          Environment = var.environment_tag
7          Function    = "BaseLabv2-identity"
8      }
9  }
10 resource "azurerm_resource_group" "rg-con" {
11     name      = "rg-baselabv2-${var.region1code}-connectivity-01"
12     location = var.region1
13     tags = {
14         Environment = var.environment_tag
15         Function    = "BaseLabv2-connectivity"
16     }
17 }
18 resource "azurerm_resource_group" "rg-sec" {
19     name      = "rg-baselabv2-${var.region1code}-security-01"
20     location = var.region1
21     tags = {
22         Environment = var.environment_tag
23         Function    = "BaseLabv2-security"
24     }
25 }

```

Declarative

- Defines infrastructure components to be created
- In this example repeated executions would result in a message informing us that “no changes” are required (as the VM is already built).

Why use Infrastructure as Code?



Cost – enables more rapid deployment, changes, test environments etc.



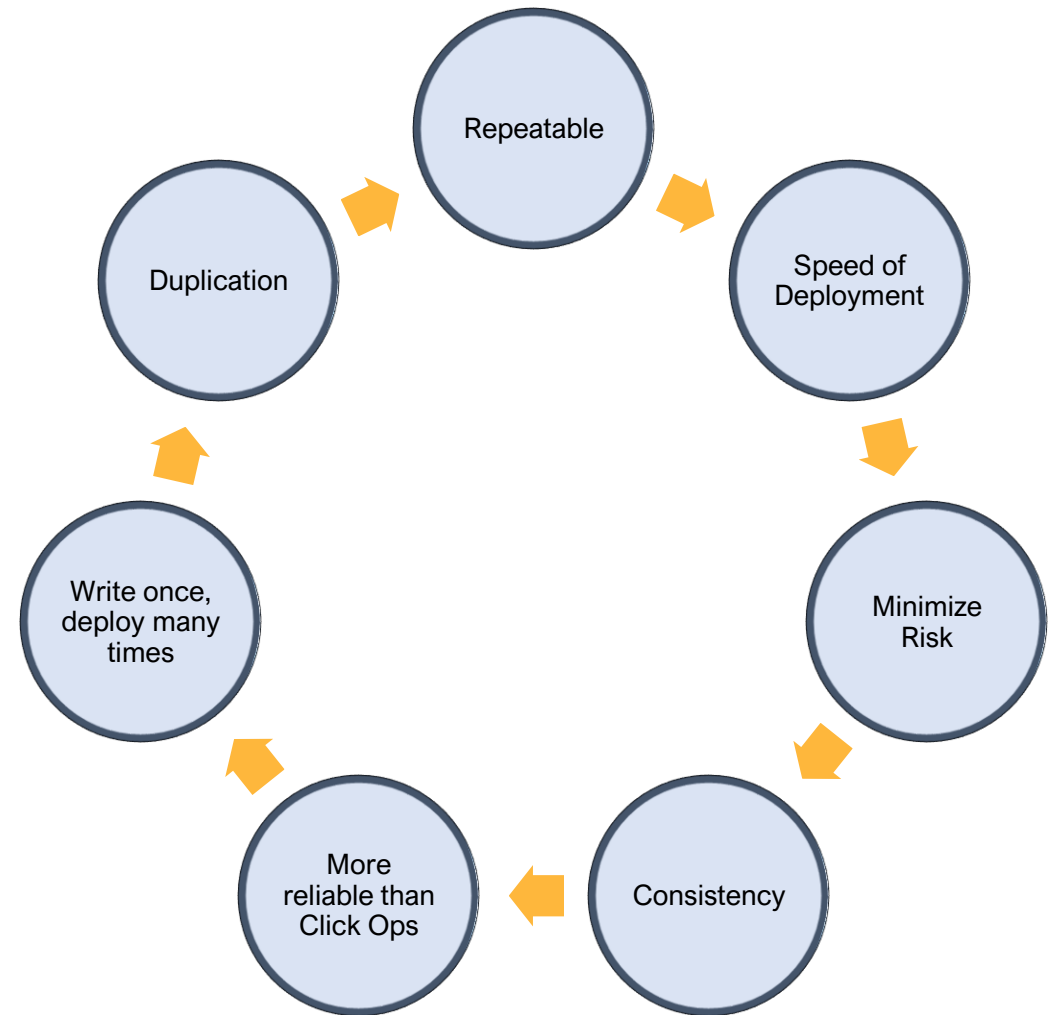
Speed – faster deployment due to less manual intervention (no ClickOps), easy testing, less human error etc. Enables DevOps methods/practices.



Risk – reduced through testing, consistency of deployments, version control etc.

Why use Infrastructure as Code?

Benefits Cycle



Other platforms are available...

Tool	Released by	Method	Approach	Written in	Comments
Chef	Chef (2009)	Pull	Declarative and imperative	Ruby	-
Otter	Inedo (2015)	Push	Declarative and imperative	-	Windows-oriented
Puppet	Puppet (2005)	Push and Pull	Declarative and imperative	C++ & Clojure since 4.0, Ruby	-
SaltStack	SaltStack (2011)	Push and Pull	Declarative and imperative	Python	-
CFEngine	Northern.tech	Pull	Declarative	C	-
Terraform	HashiCorp (2014)	Push	Declarative and imperative	Go	-
Ansible / Ansible Tower	Red Hat (2012)	Push	Declarative and imperative	Python	-

https://en.wikipedia.org/wiki/Infrastructure_as_code

What is Terraform?



- Terraform is an **Infrastructure as Code** Software tool, that can interact with a wide range of Platforms and Environments, using Providers.



- Can be used in both Cloud and On-Premises environments. Can be used to combine on-premises and Cloud, or Cloud and Cloud for example.
- Terraform comes in 3 main varieties:
 - Community Edition – I will be using this to demo today!
 - Terraform Cloud
 - Terraform Enterprise

What is Terraform?



- Terraform comes in 3 main varieties:
 - Community Edition – I will be using this to demo today!
 - Terraform Cloud
 - Terraform Enterprise

Community

Self-managed | Always free

Download >

Terraform Cloud

Managed Terraform

Compare plans >

Enterprise

Self-managed custom
deployments

Learn more >

<https://www.terraform.io/>

What is Terraform?



- Terraform Cloud
- Terraform Enterprise

Terraform pricing

Free	Standard	Plus	Enterprise
UP TO 500 resources per month	STARTING AT \$0.00014 per hour per resource	Custom	Custom
Cloud	Cloud	Cloud	Self-managed
Get started with all capabilities needed for infrastructure as code provisioning.	For professional individuals or teams adopting infrastructure as code provisioning.	For enterprises standardizing and managing infrastructure automation and lifecycle, with scalable runs.	For enterprises with special security, compliance, and additional operational requirements.
No credit card required	Enterprise support included	Enterprise support included	Enterprise support included
Get started	Get started	Contact sales	Contact sales
	First 500 resources per month are free Learn more		

Authentication

- **Demo / Lab Environments**

Usually authenticate at the CLI or use a Service Principal

- **Production Environments**

Service Principal or a Managed Service Identity

<https://learn.microsoft.com/en-us/azure/developer/terraform/authenticate-to-azure?tabs=bash>

Authentication

Providers / hashicorp / azurerm / Version 3.79.0 ▾ Latest Version

azurerm ⓘ Overview Documentation **USE PROVIDER ▾**

AZURERM DOCUMENTATION

Filter

azurerm provider

▾ Guides

- Azure Provider: Authenticating via a Service Principal and a Client Certificate
- Azure Provider: Authenticating via a Service Principal and a Client Secret
- Azure Provider: Authenticating via a Service Principal and OpenID Connect
- Azure Provider: Authenticating via Managed Identity
- Azure Provider: Authenticating via the Azure CLI
- Azure Provider: Migrating from Deprecated Resources Guide
- Azure Resource Manager: 3.0 Upgrade Guide
- Azure Resource Manager: Continuous

Azure Provider: Authenticating using a Service Principal with a Client Certificate

Terraform supports a number of different methods for authenticating to Azure:

- [Authenticating to Azure using the Azure CLI](#)
- [Authenticating to Azure using Managed Service Identity](#)
- Authenticating to Azure using a Service Principal and a Client Certificate (which is covered in this guide)
- [Authenticating to Azure using a Service Principal and a Client Secret](#)
- [Authenticating to Azure using a Service Principal and OpenID Connect](#)

We recommend using either a Service Principal or Managed Service Identity when running Terraform non-interactively (such as when running Terraform in a CI server) - and authenticating using the Azure CLI when running Terraform locally.

ON THIS PAGE

- Setting up an Application and Service Principal
- Generating a Client Certificate
- Creating the Application and Service Principal
- Configuring Terraform to use the Client Certificate

Report an issue ↗

<https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs>

Providers

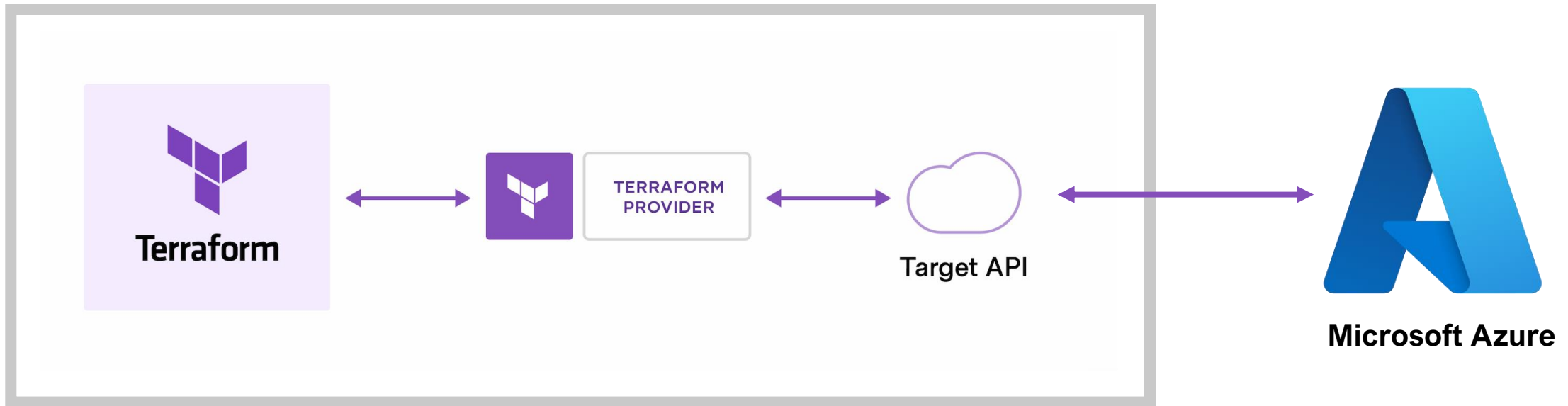
- Before we can run Terraform, we need to add a “Provider” to our Code. Providers are plugins for Terraform that allow Terraform to interact with an external API.

<https://registry.terraform.io/browse/providers>

- In simple Terms – providers enable communication with platforms or services outside of Terraform
- For example – with Microsoft Azure, we would need to add the AzureRM Provider to Terraform before we can interact with Azure.

<https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs>

AzureRM Provider



<https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs>

AZURERM DOCUMENTATION

azurerm provider

- > Guides
- > AAD B2C
- > API Management
- > Active Directory Domain Services
- > Advisor
- > Analysis Services
- > App Configuration
- > App Service (Web Apps)
- > Application Insights
- > Arc Resource Bridge

Azure Provider

The Azure Provider can be used to configure infrastructure in [Microsoft Azure](#) using the Azure Resource Manager API's. Documentation regarding the [Data Sources](#) and [Resources](#) supported by the Azure Provider can be found in the navigation to the left.

To learn the basics of Terraform using this provider, follow the hands-on [get started tutorials](#).

Interested in the provider's latest features, or want to make sure you're up to date? Check out the [changelog](#) for version information and release notes.

Authenticating to Azure

Terraform supports a number of different methods for authenticating to Azure:

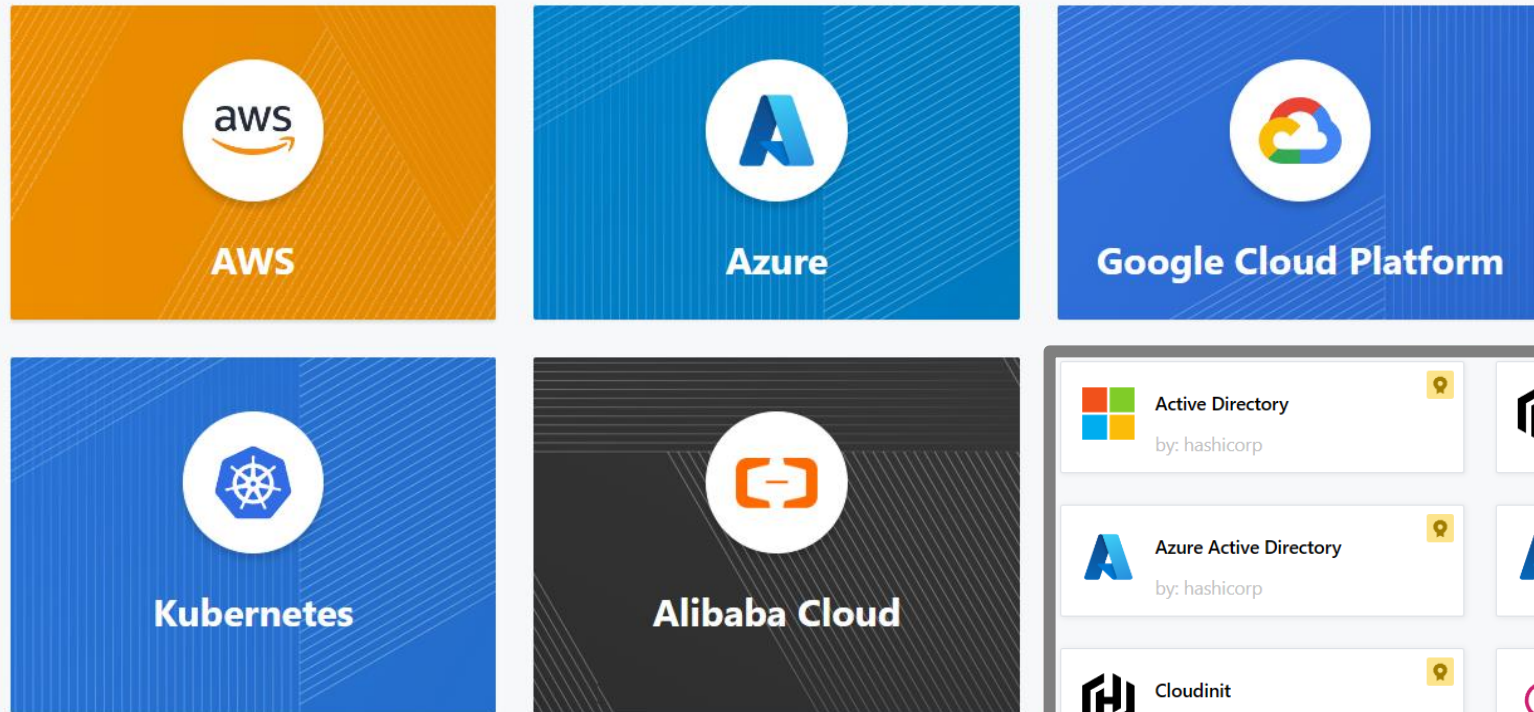
ON THIS PAGE

- [Authenticating to Azure](#)
- [Example Usage](#)
- [Bugs and Feature Requests](#)
- [Argument Reference](#)
- [Features](#)

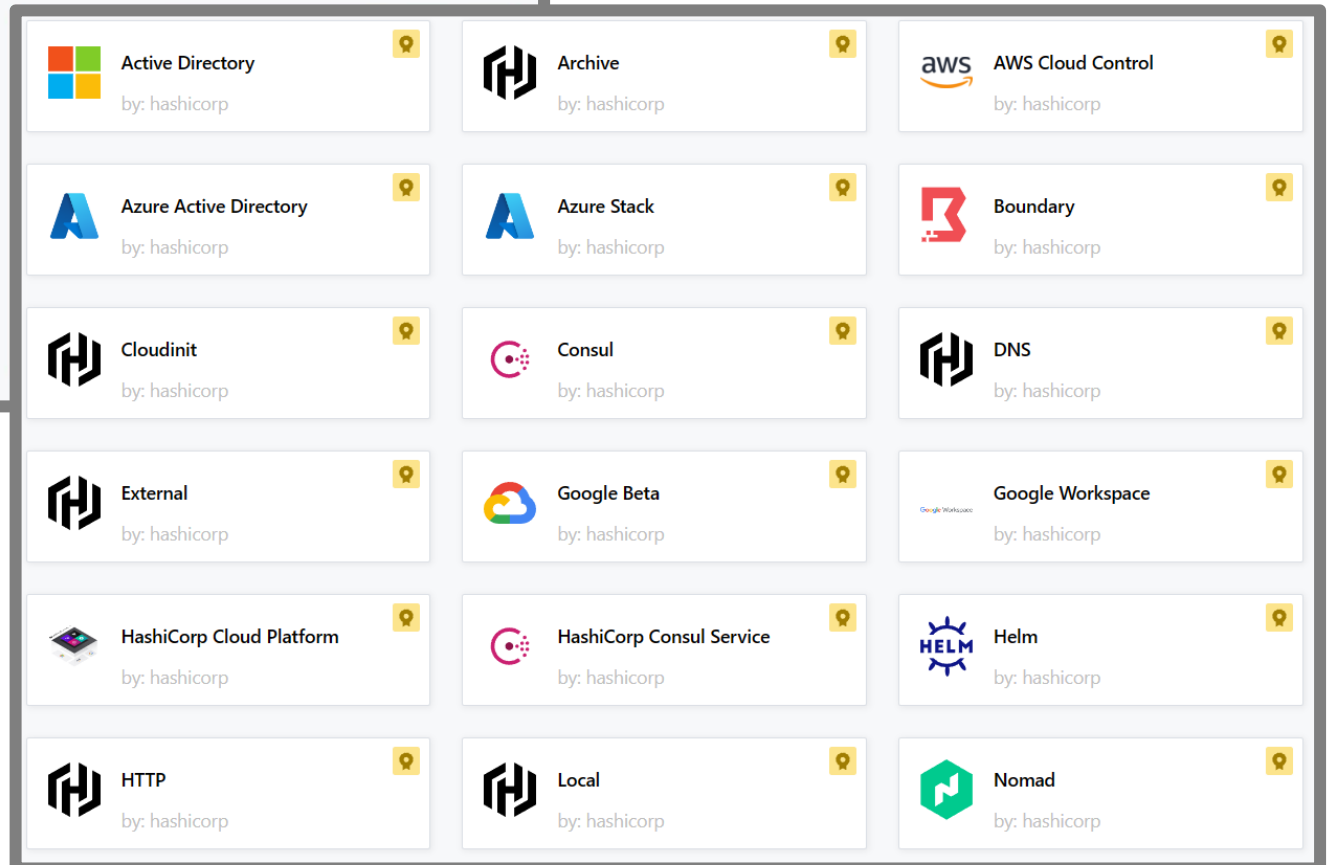
[Report an issue](#)

Providers

Providers are a logical abstraction of an upstream API. They are responsible for understanding API interactions and exposing resources.



What other Providers are available?



<https://registry.terraform.io/browse/providers>

AWS DOCUMENTATION

[aws provider](#)

- > Guides
- > ACM (Certificate Manager)
- > ACM PCA (Certificate Manager Private Certificate Authority)
- > AMP (Managed Prometheus)
- > API Gateway
- > API Gateway V2
- > Account Management
- > Amplify
- > App Mesh
- > App Runner
- > AppConf

AWS Provider

Use the Amazon Web Services (AWS) provider to interact with the many resources supported by AWS. You must configure the provider with the proper credentials before you can use it.

Use the navigation to the left to read about the available resources. There are currently 1271 resources and 520 data sources available in the provider.

To learn the basics of Terraform using this provider, follow the hands-on [get started tutorials](#). Interact with AWS services, including Lambda, RDS, and IAM by following the [AWS services tutorials](#).

Example Usage

Terraform 0.13 and later:

```
terraform {
  provider "aws" {
    # Configuration for the AWS provider
  }
}
```

New Multi-language provider docs

Terraform ▾



The Registry now supports multi-language docs powered by CDK for Terraform. [Learn more ↗](#)

ON THIS PAGE

- [Example Usage](#)
 - Authentication and Configuration
 - AWS Configuration Reference
 - Custom User-Agent Information
 - Argument Reference
 - Getting the Account ID





[Report an issue ↗](#)

Process

1. Terraform code is typically arranged across a number of files: “tf files”. (Because they have the extension TF)
- 
2. These files define the infrastructure and its configuration (or changes!) that we want Terraform to apply.
- 
3. At the time of running Terraform, these files are analysed by Terraform, and turned into an execution plan to apply our changes.

Terraform Stages

- Running Terraform involves a number of stages of deployment:


Terraform init		This stage initialises the Terraform binaries, and downloads the required providers, based on what we have defined.
Terraform plan		This stage examines our TF files and provides an overview of the infrastructure changes – by providing an execution plan.
Terraform apply		This stage carries out the execution plan, and implements the changes. Note: this also runs plan.
Terraform destroy		This stage destroys the created infrastructure – use carefully!

The State File

Terraform must store information about your infrastructure within a file known as the “State File”.

This is so that Terraform can work out changes required to the infrastructure based on your code or configuration changes.

The State File can be stored locally, or remotely, depending on the deployment type and needs.

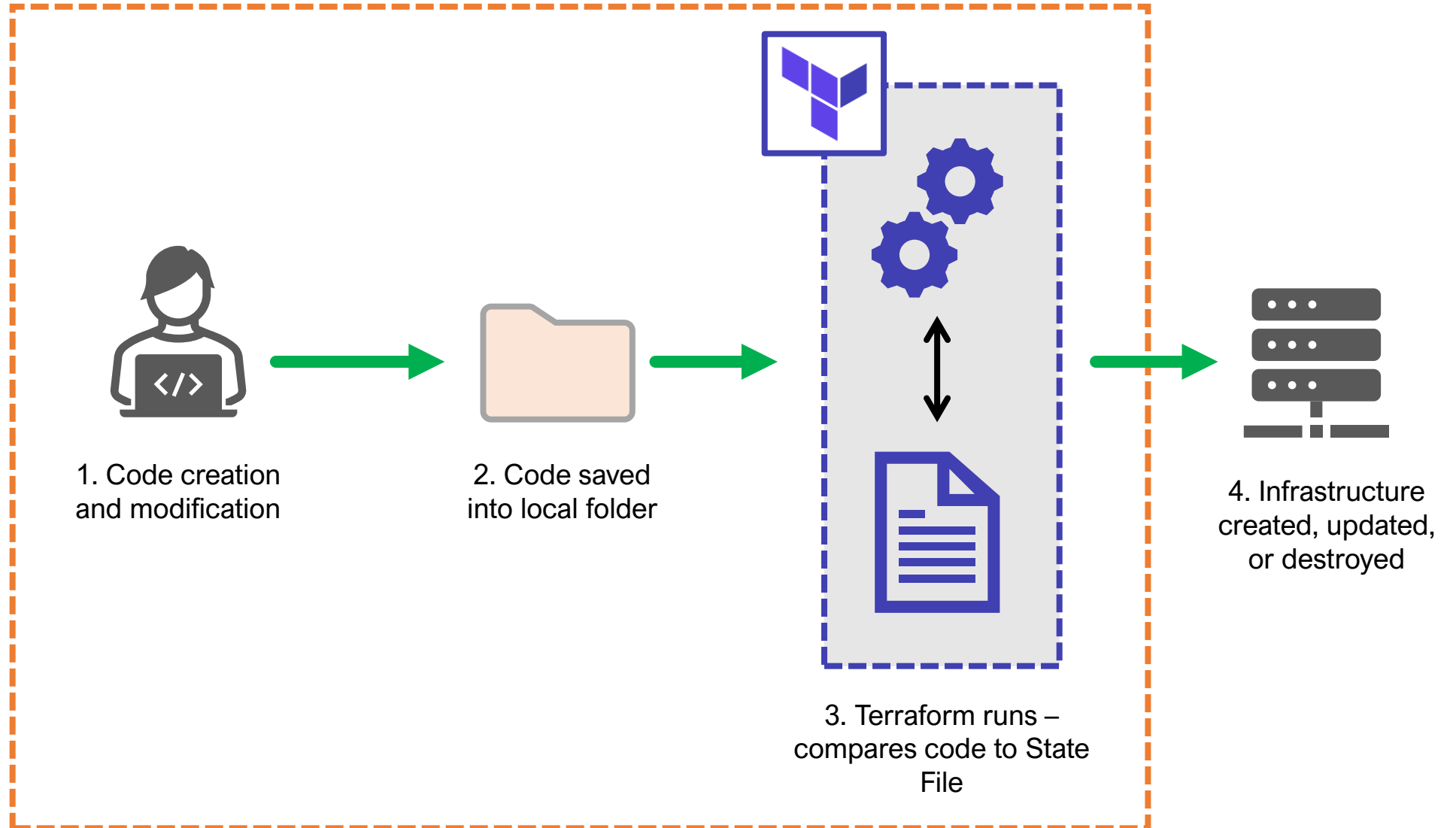
- Local State – learning/testing/labs/development 
- Remote State – using DevOps tooling or collaborating on code

<https://developer.hashicorp.com/terraform/language/state>

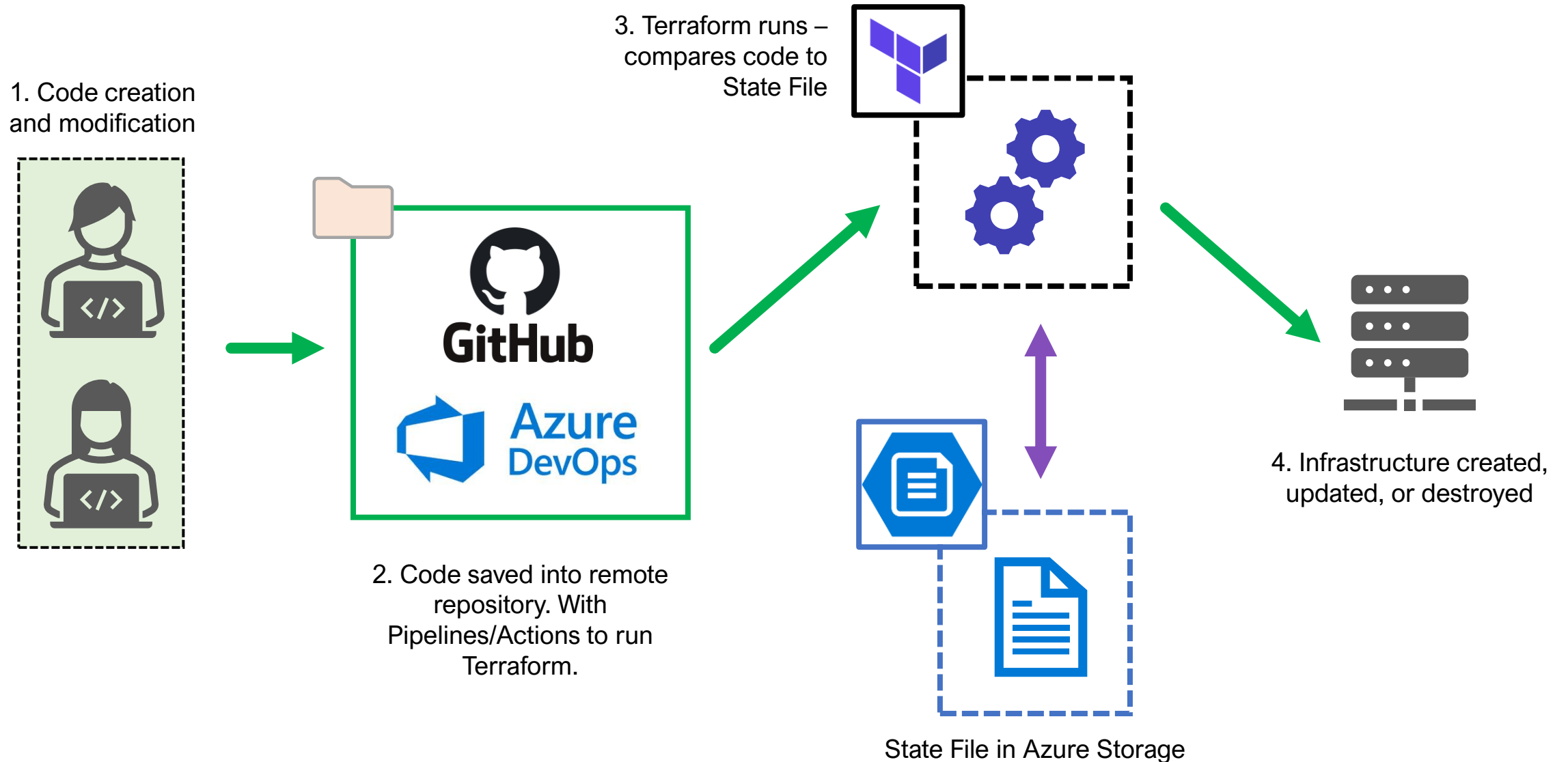
Ways of Working – Local Example



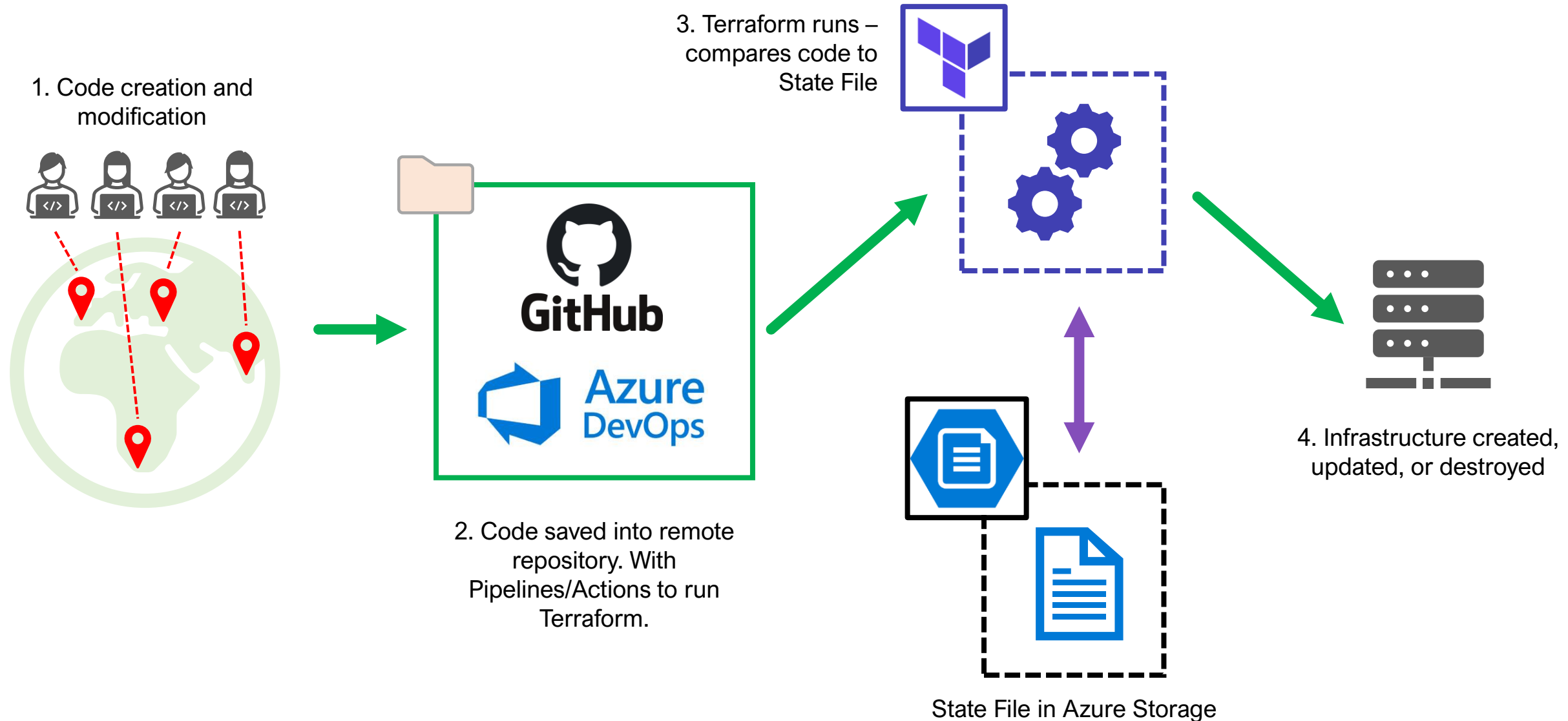
Local Example:
All work done
on a single
machine.



Ways of Working – A Remote Example



Ways of Working – An even more Remote Example

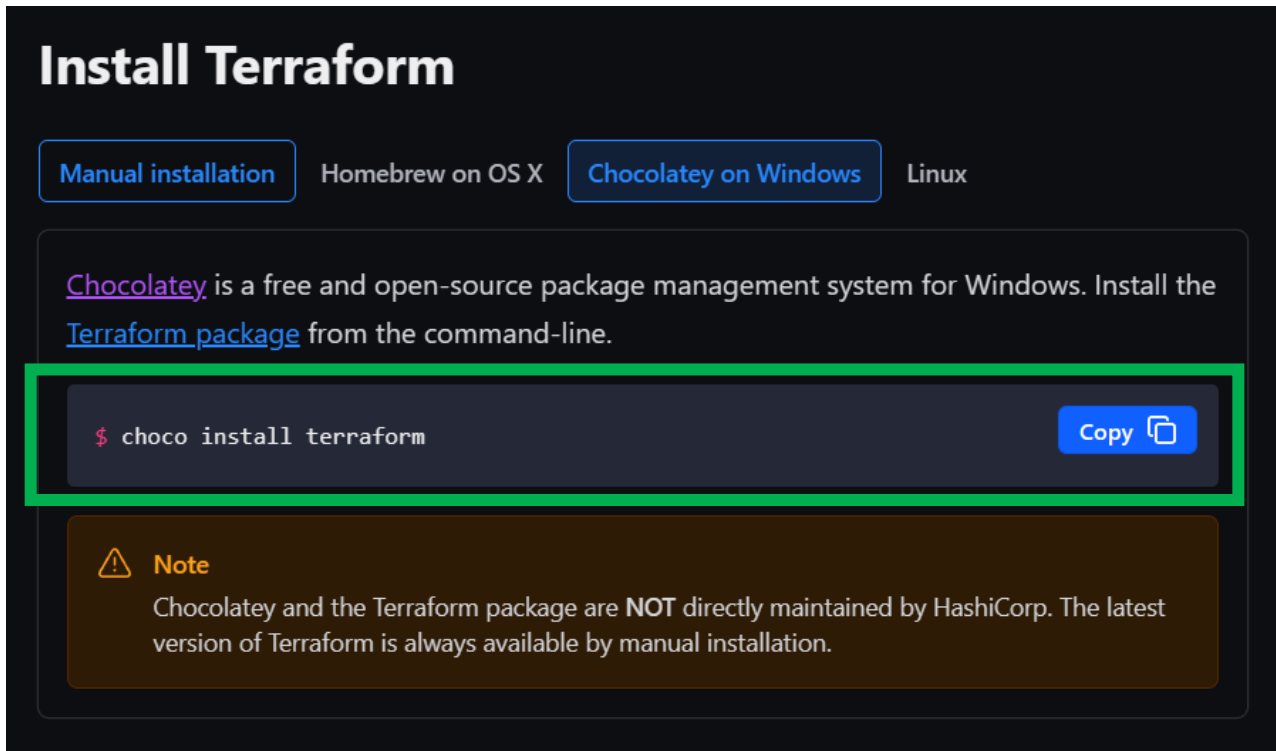


Why use Terraform for Azure Deployment?

Just a few key reasons!

- Create infrastructure easily, repeatably, in different locations/platforms/regions.
- Enables Rapid Development / Testing
- Cost Effective Test Environments – create on demand, destroy once used.
- Scale up/down/in/out as required
- Expansion – use functions like count and variable methods like maps
- Enables Version control of Infrastructure
- Work safely and in a standardised way across distributed teams.
- Write once, deploy many times.

Installing Terraform



The screenshot shows the 'Install Terraform' page with tabs for 'Manual installation', 'Homebrew on OS X', 'Chocolatey on Windows', and 'Linux'. The 'Chocolatey on Windows' tab is selected. Below the tabs, a text block explains that Chocolatey is a free and open-source package management system for Windows and instructs to install the Terraform package from the command-line. A terminal snippet is highlighted with a green border, showing the command `$ choco install terraform` and a 'Copy' button. Below this, a 'Note' box states that Chocolatey and the Terraform package are NOT directly maintained by HashiCorp and that the latest version of Terraform is always available by manual installation.

Install Terraform

[Manual installation](#) Homebrew on OS X [Chocolatey on Windows](#) Linux

[Chocolatey](#) is a free and open-source package management system for Windows. Install the [Terraform package](#) from the command-line.

```
$ choco install terraform
```

Note
Chocolatey and the Terraform package are **NOT** directly maintained by HashiCorp. The latest version of Terraform is always available by manual installation.

Recommended minimum software:

- Terraform
- Visual Studio Code + Extensions

<https://community.chocolatey.org/>



```
choco install terraform -y  
choco install azure-cli -y  
choco install vscode -y
```

<https://developer.hashicorp.com/terraform/tutorials/azure-get-started/install-cli>

Installing Terraform – VSCode Plugin



HashiCorp Terraform

HashiCorp  hashicorp.com |  3,292,248 installs | ★★☆☆☆ (186) |

Syntax highlighting and autocompletion for Terraform

Install

[Trouble Installing?](#)

[Overview](#)

[Version History](#)

[Q & A](#)

[Rating & Review](#)

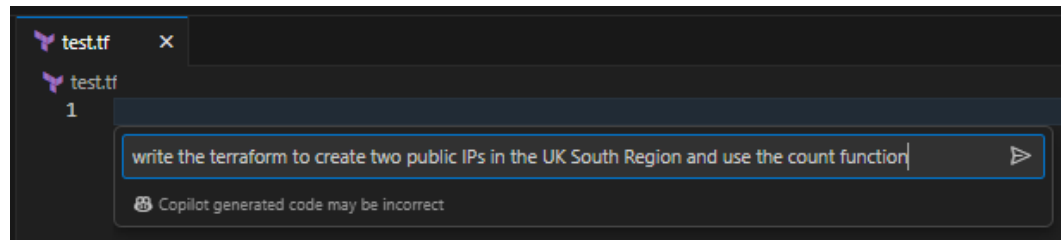
Terraform Extension for Visual Studio Code

The HashiCorp [Terraform Extension for Visual Studio Code \(VS Code\)](#) with the [Terraform Language Server](#) adds editing features for [Terraform](#) files such as syntax highlighting, IntelliSense, code navigation, code formatting, module explorer and much more!

Features:

- Intellisense
- Syntax Validation and Highlighting
- Code Navigation
- Code Formatting
- Code Snippets
- Terraform Cloud Integration

<https://marketplace.visualstudio.com/items?itemName=HashiCorp.terraform>



```
test.tf
test.tf > ...
1 provider "azurerm" {
2   features {}
3 }
4
5 resource "azurerm_public_ip" "example" {
6   count = 2
7
8   name          = "example-${count.index}"
9   location      = "uksouth"
10  resource_group_name = azurerm_resource_group.example.name
11
12  allocation_method = "Static"
13
14  tags = {
15    environment = "dev"
16  }
17 }
18
19 resource "azurerm_resource_group" "example" {
20   name     = "example-resources"
21   location = "uksouth"
22 }
23
```

A helping hand... Try GitHub Copilot!

Demo Time!

```
core.tf > ...
1 # Resource Groups
2 resource "azurerm_resource_group" "rg-ide" {
3   name     = "rg-baselabv2-${var.region1code}-identity-01"
4   location = var.region1
5   tags = {
6     Environment = var.environment_tag
7     Function    = "BaseLabv2-identity"
8   }
9 }
10 resource "azurerm_resource_group" "rg-con" {
11   name     = "rg-baselabv2-${var.region1code}-connectivity-01"
12   location = var.region1
13   tags = {
14     Environment = var.environment_tag
15     Function    = "BaseLabv2-connectivity"
16   }
17 }
18 resource "azurerm_resource_group" "rg-sec" {
19   name     = "rg-baselabv2-${var.region1code}-security-01"
20   location = var.region1
21   tags = {
22     Environment = var.environment_tag
23     Function    = "BaseLabv2-security"
24   }
25 }
26 # Key Vault
27 resource "random_id" "kv-name" {
28   byte_length = 6
29   prefix      = "kv"
30 }
31 data "azurerm_client_config" "current" {}
32 resource "azurerm_key_vault" "kv1" {
33   name                = random_id.kv-name.hex
34   location             = var.region1
35   resource_group_name = azurerm_resource_group.rg-sec.name
36   enabled_for_disk_encryption = true
37   tenant_id           = data.azurerm_client_config.current.tenant_id
38   soft_delete_retention_days = 7
39   purge_protection_enabled = false
40
41   sku_name = "standard"
42 }
```

- Everything I am using today is available in my Terraform-Azure repo: <https://github.com/jakewalsh90/Terraform-Azure>
- All you need is Terraform, VSCode, Azure CLI, and an Azure Subscription.
- We will explore & demo the following:
 - Deployment of a Lab Environment
 - Run through code files & deployment
 - Changes
 - Destruction

Next Steps...

Links and Resources:

Azure / Terraform Blog Posts:

- <https://jakewalsh.co.uk/category/terraform/>
- <https://jakewalsh.co.uk/category/azure/>

HashiCorp Learn – Azure Tutorial:

- <https://developer.hashicorp.com/terraform/tutorials/azure-get-started>

Try Some Sample Environments:

- <https://github.com/jakewalsh90/Terraform-Azure>


```
core.tf > ...
1 # Resource Groups
2 resource "azurerm_resource_group" "rg-ide" {
3   name      = "rg-baselabv2-${var.region1code}-identity-01"
4   location  = var.region1
5   tags = {
6     Environment = var.environment_tag
7     Function    = "BaseLabv2-identity"
8   }
9 }
10 resource "azurerm_resource_group" "rg-con" {
11   name      = "rg-baselabv2-${var.region1code}-connectivity-01"
12   location  = var.region1
13   tags = {
14     Environment = var.environment_tag
15     Function    = "BaseLabv2-connectivity"
16   }
17 }
18 resource "azurerm_resource_group" "rg-sec" {
19   name      = "rg-baselabv2-${var.region1code}-security-01"
20   location  = var.region1
21   tags = {
22     Environment = var.environment_tag
23     Function    = "BaseLabv2-security"
24   }
25 }
26 # Key Vault
27 resource "random_id" "kv-name" {
28   byte_length = 6
29   prefix      = "kv"
30 }
31 data "azurerm_client_config" "current" {}
32 resource "azurerm_key_vault" "kv1" {
33   name                = random_id.kv-name.hex
34   location            = var.region1
35   resource_group_name = azurerm_resource_group.rg-sec.name
36   enabled_for_disk_encryption = true
37   tenant_id           = data.azurerm_client_config.current.tenant_id
38   soft_delete_retention_days = 7
39   purge_protection_enabled = false
40
41   sku_name = "standard"
42 }
```

Thank You!

Please note – the views/opinions in this presentation are entirely my own. This presentation will not be kept updated after the Festive Tech Calendar 2023 (December 2023) – so may be outdated if downloaded afterwards.

If in any doubt, please check latest documentation and MS Links for updated info!

Festive Tech Calendar

.....
2023

