

.NET Conf

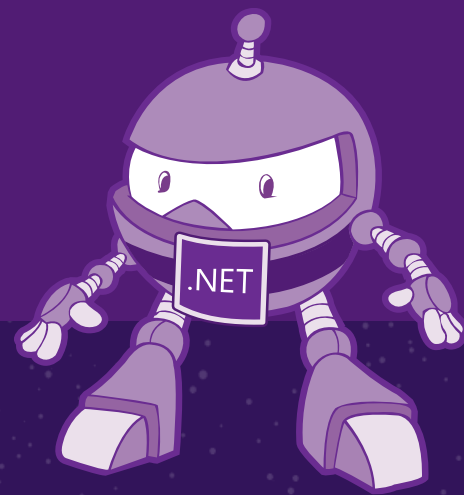
探索 .NET 新世界



Host by
STUDY4

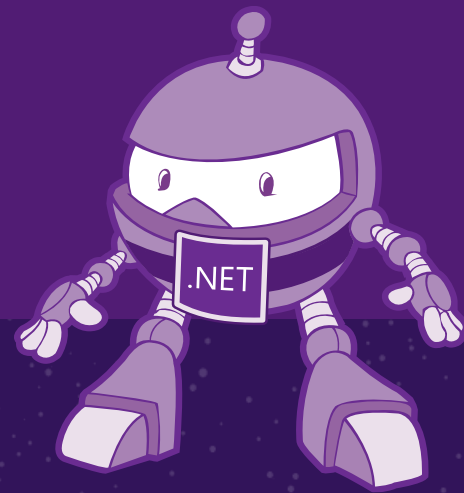
利用 HashiCorp 降低自己的肝指數

Perl Tsai
雲端邊緣肥宅



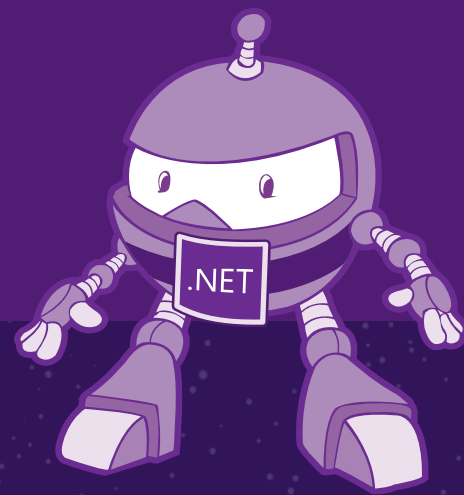
利用 HashiCorp 降低自己的肝指數

Perl Tsai
雲端邊緣肥宅

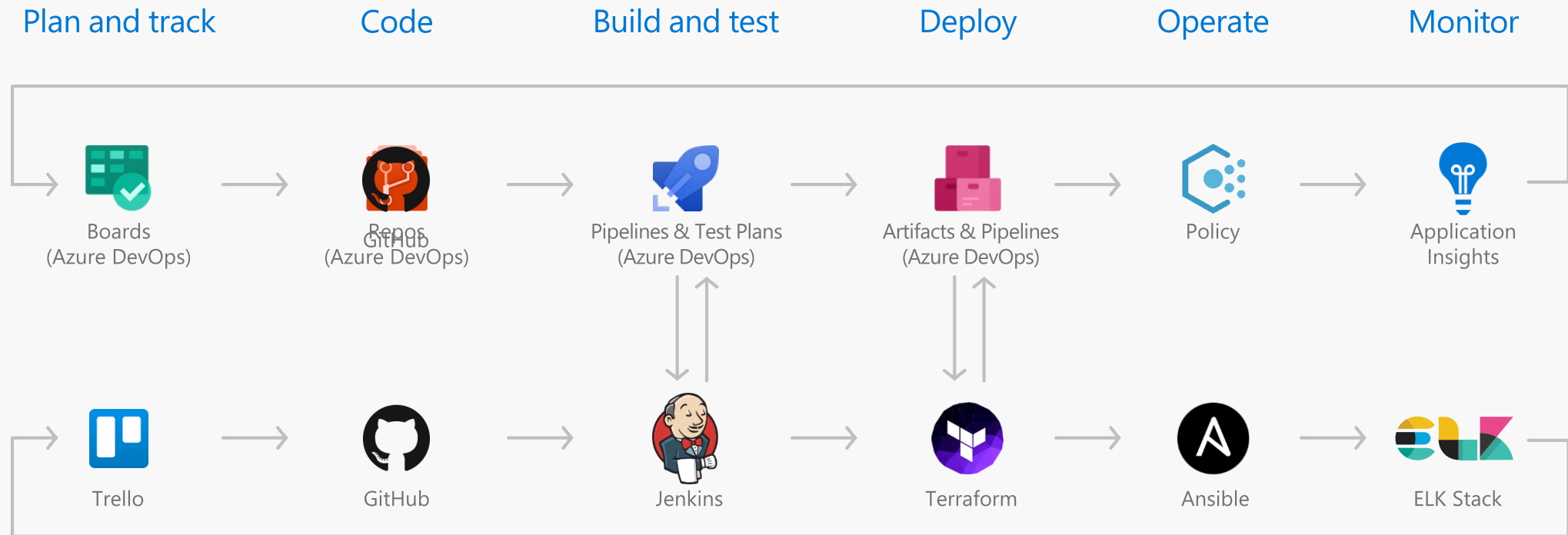


教各位如何 強化自己的肝硬度

Perl Tsai
雲端邊緣肥宅



DevOps on Azure framework



DevOps Tool Integrations

Your favorite toolchain, seamlessly integrated with Azure



The Best Development Experience

Whether your developing on Azure, on-premise, or another cloud, Visual Studio Code extensions from Microsoft and the community help accelerate development across Linux, macOS and Windows



First-class Integration

Microsoft collaborates directly in open source projects with our partners, and the community, to bring native Azure integration. Many of these tools are also directly available in Azure Cloud Shell – try them out!



Accelerated Customer Success

You can get clear guidance for integrating your favorite tools with Azure, with dedicated documentation hubs and example solution architecture. Get started, fast.

<http://azure.microsoft.com/solutions/devops>



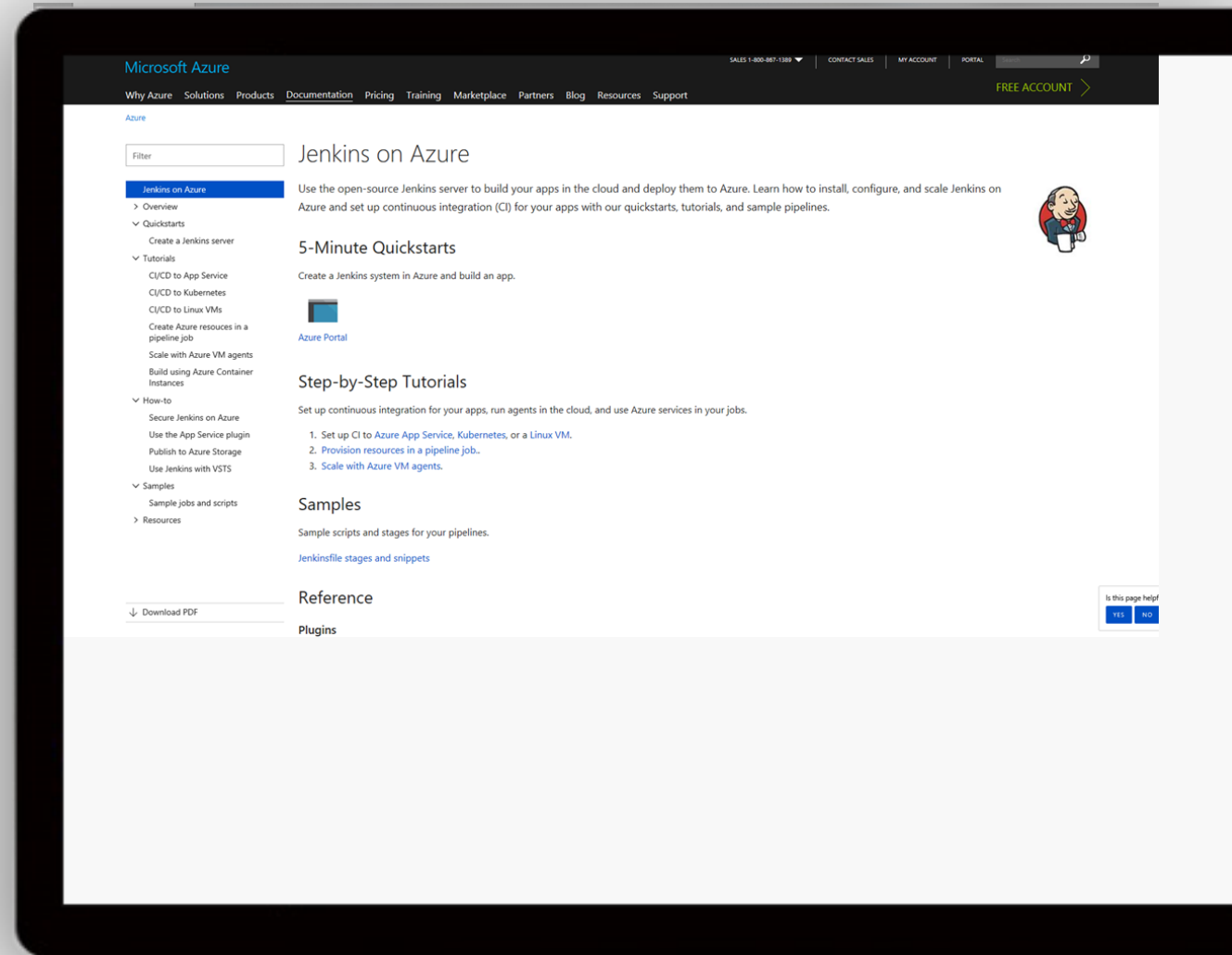
ANSIBLE



HashiCorp
Packer



HashiCorp
Terraform



Terraform

Azure DevOps Tool Integrations

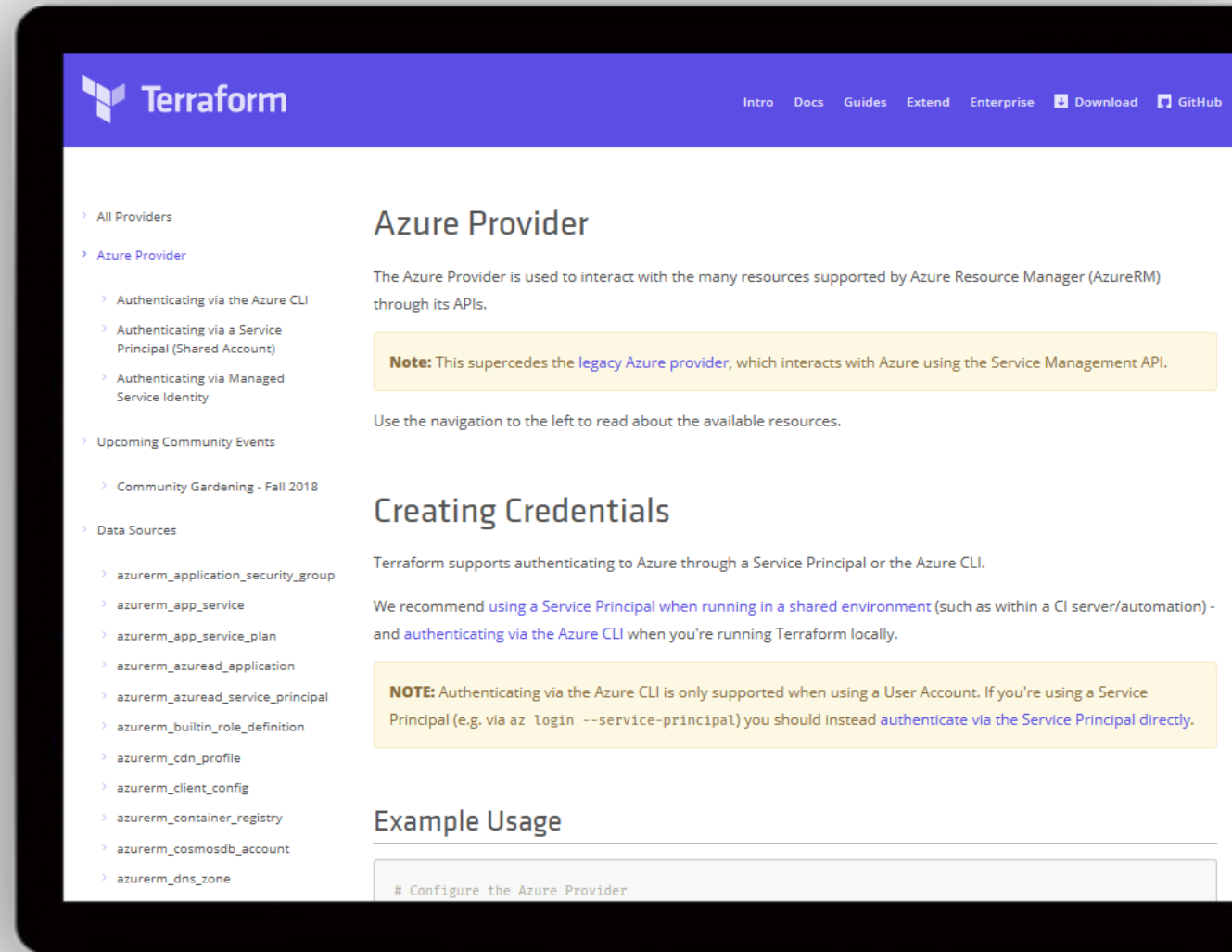


Bringing native Azure support for customers using Terraform

- [Documentation Hub for Terraform](#)
- [Terraform in Azure Cloud Shell](#)
- [Azure Resource Provider](#)
- [Azure Module Registry](#)
- [Azure Cloud Shell Integration](#)

 docs.microsoft.com/azure/terraform

© Microsoft Corporation



Azure

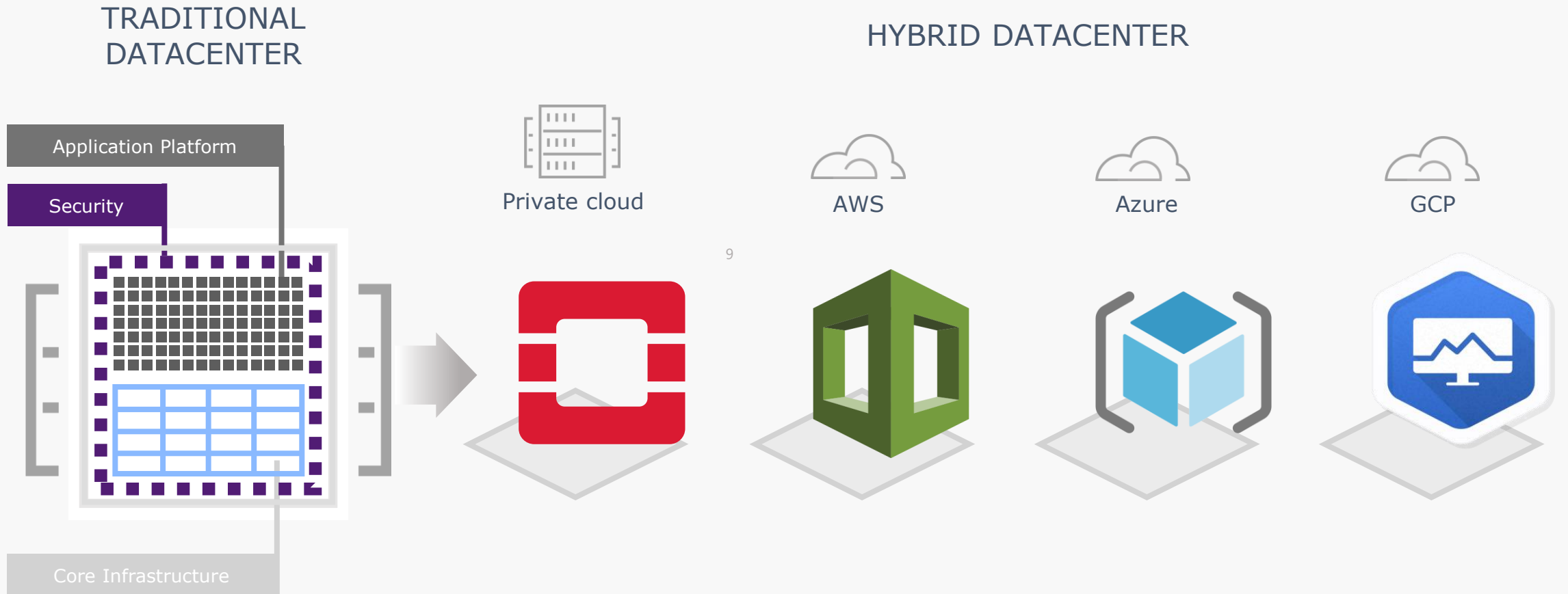
Open Source

According to Forrester, over **forty percent** of CIOs view adoption of open source technologies as critical for them in the next year – primarily because of low cost, avoidance of vendor lock-in and agility.

And that same North Bridge study saw use of OSS increased **sixty-five percent** over same companies surveyed from previous year.

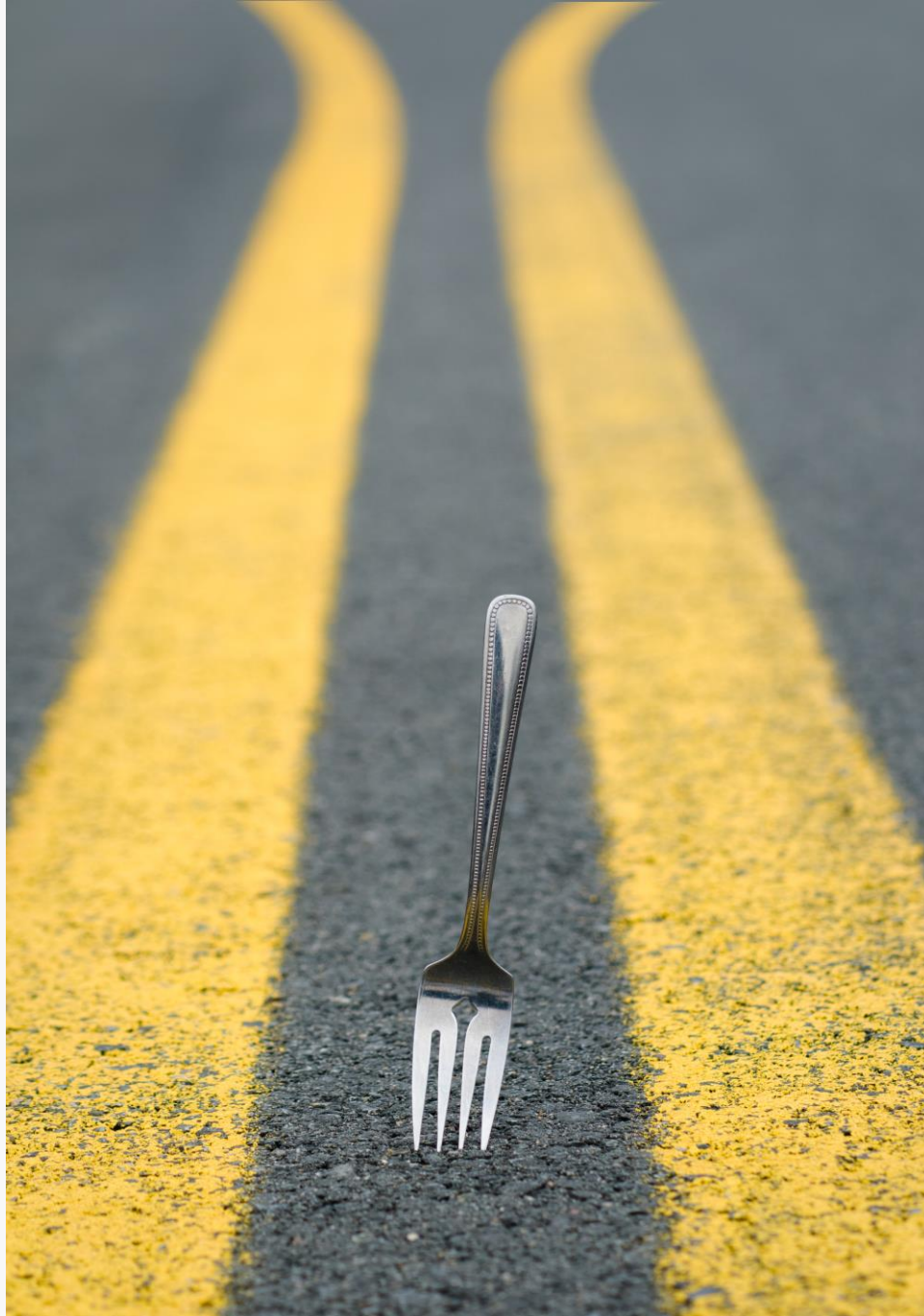
[Mark Russinovich on Microsoft Blog](#)

Multi-cloud infrastructure transition



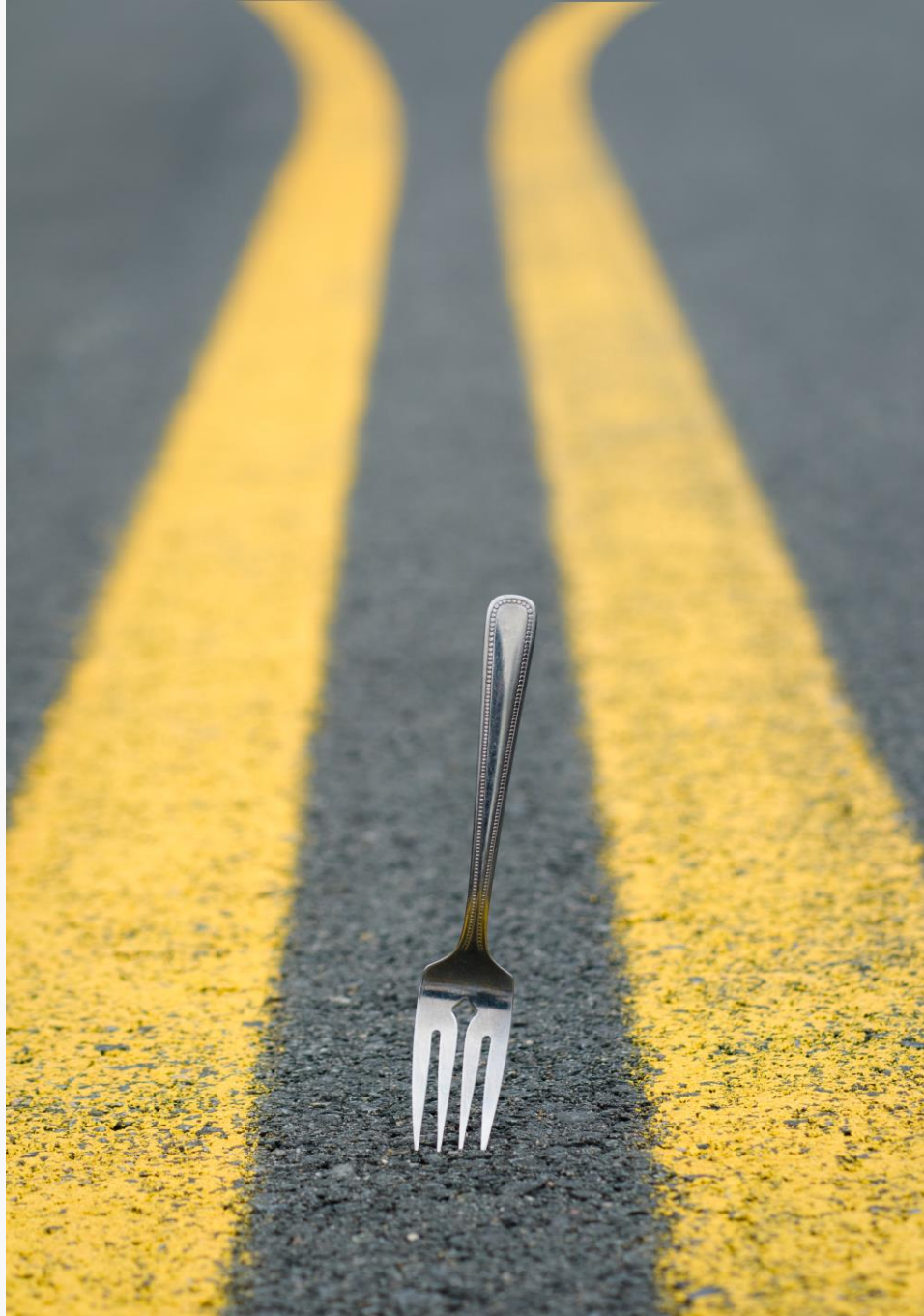


ARM and az CLI





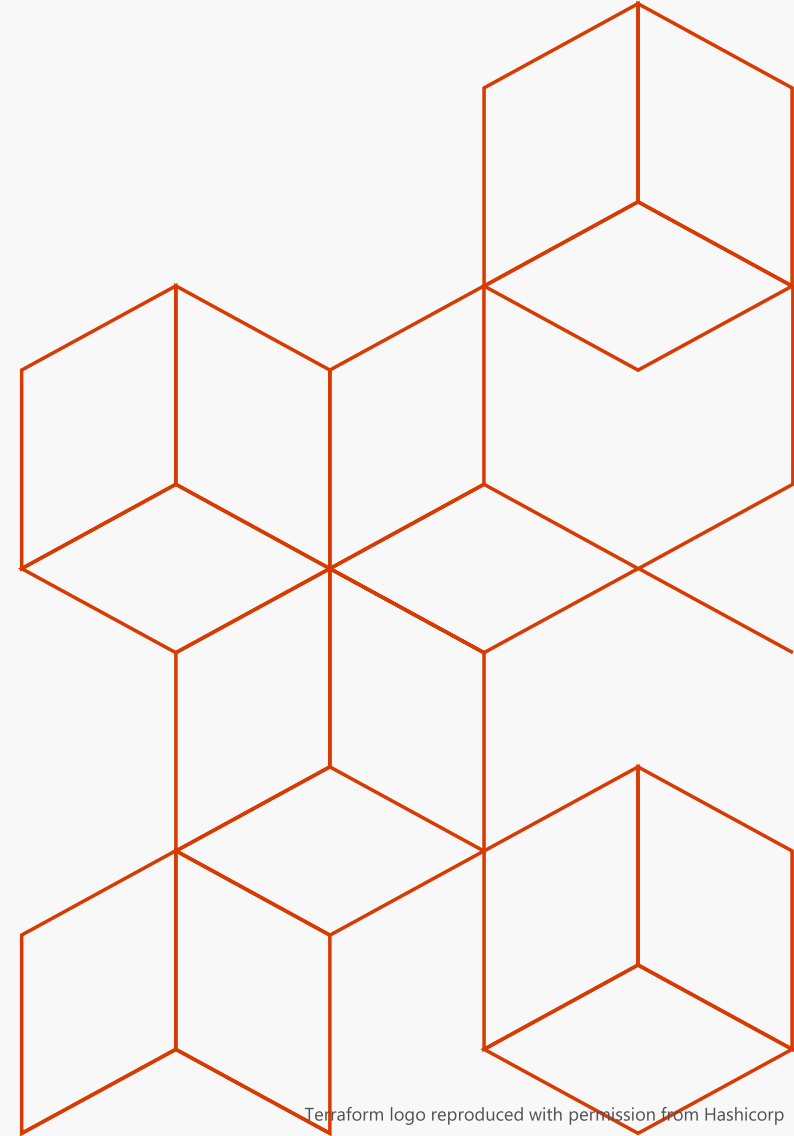
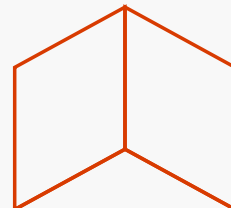
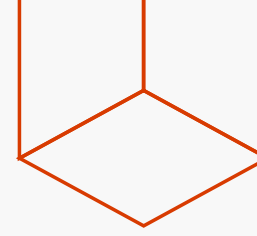
ARM and az CLI



Fully OSS
Multi-cloud
Hybrid

Terraform vs Chef/Puppet/Ansible

Terraform	Chef/Puppet/Ansible
Infra management tool at data center level	Config management tools to deploy apps
Integrates with config management tools	Traditional config as code app deployments
Integrates with Packer, Consul and Vault for immutable app deployments	Not recommended for immutable app deployments (as leads to config drift)





Infrastructure as Code

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

✗ Apply same config across clouds

The HashiCorp Configuration Language

(HCL) The HashiCorp Configuration Language (HCL) is a small domain specific language which is based on JSON.

```
resource "azurerm_redis_cache" "sample" {  
  name          = "tf-redis-basic"  
  location      = "${azurerm_resource_group.test.location}"  
  resource_group_name = "${azurerm_resource_group.test.name}"  
  capacity      = 0  
  family        = "C"  
  sku_name      = "Basic"  
  enable_non_ssl_port = "${var.redis_enable_non_ssl}"  
  tags          = "${local.all_tags}"  
}
```


Assembly

Cloud Shell

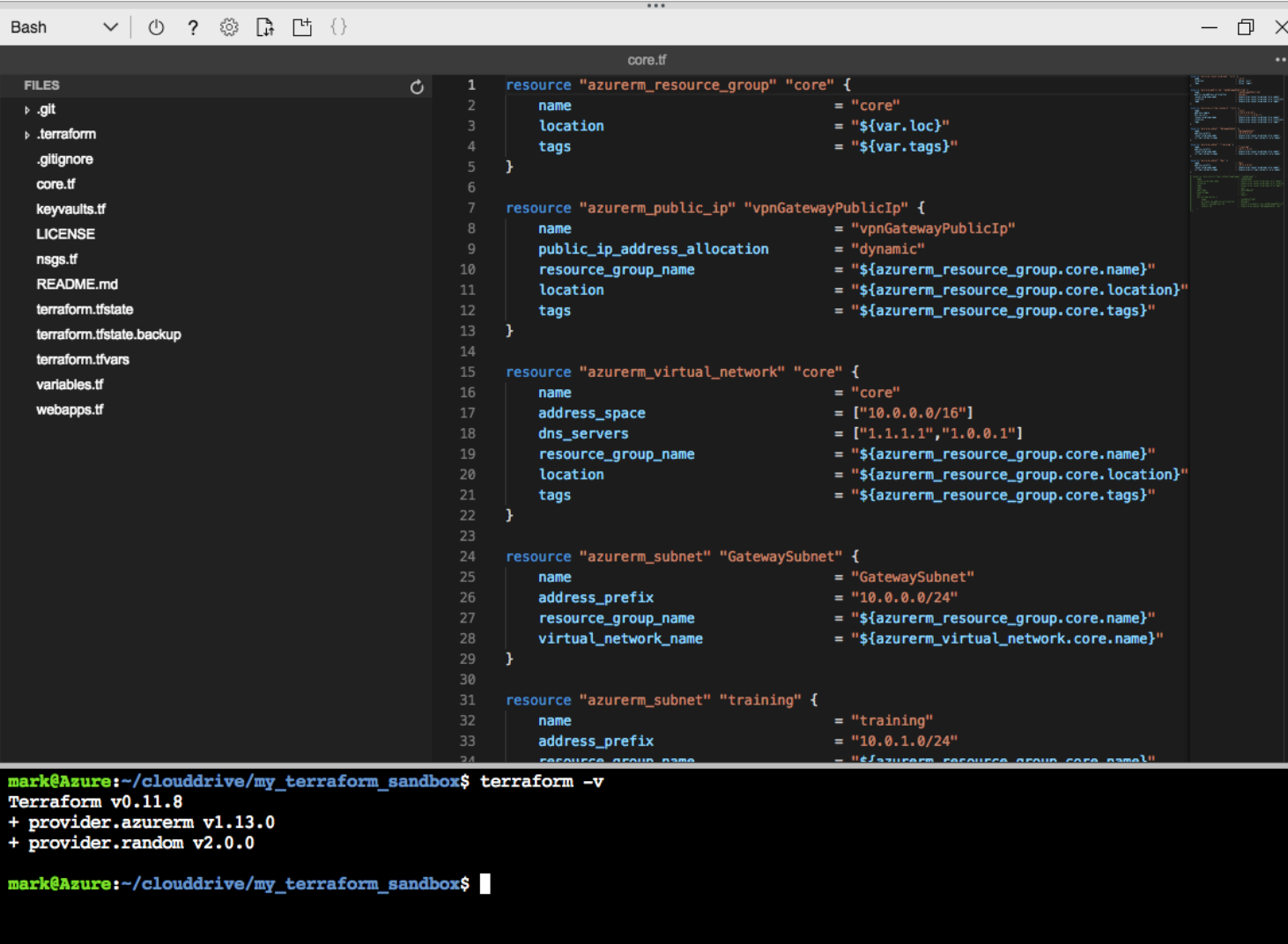
Terraform integration

Editor with Terraform Syntax Highlight

Visual Studio Code

Terraform extension

Azure Terraform extension



The screenshot shows a Visual Studio Code editor window with a dark theme. The left sidebar displays a file explorer with a list of files: `.git`, `.terraform`, `.gitignore`, `core.tf`, `keyvaults.tf`, `LICENSE`, `nsgs.tf`, `README.md`, `terraform.tfstate`, `terraform.tfstate.backup`, `terraform.tfvars`, `variables.tf`, and `webapps.tf`. The main editor area shows the content of `core.tf` with Terraform syntax highlighting. The code defines three resources: `azurerm_resource_group` (named "core"), `azurerm_public_ip` (named "vpnGatewayPublicIp"), and `azurerm_virtual_network` (named "core"). It also defines two subnets: `azurerm_subnet` (named "GatewaySubnet") and `azurerm_subnet` (named "training"). The bottom terminal window shows the output of the `terraform -v` command, indicating the installed versions of Terraform and its providers.

```
core.tf
1 resource "azurerm_resource_group" "core" {
2   name           = "core"
3   location       = "${var.loc}"
4   tags           = "${var.tags}"
5 }
6
7 resource "azurerm_public_ip" "vpnGatewayPublicIp" {
8   name           = "vpnGatewayPublicIp"
9   public_ip_address_allocation = "dynamic"
10  resource_group_name = "${azurerm_resource_group.core.name}"
11  location         = "${azurerm_resource_group.core.location}"
12  tags             = "${azurerm_resource_group.core.tags}"
13 }
14
15 resource "azurerm_virtual_network" "core" {
16   name           = "core"
17   address_space  = ["10.0.0.0/16"]
18   dns_servers    = ["1.1.1.1", "1.0.0.1"]
19   resource_group_name = "${azurerm_resource_group.core.name}"
20   location       = "${azurerm_resource_group.core.location}"
21   tags           = "${azurerm_resource_group.core.tags}"
22 }
23
24 resource "azurerm_subnet" "GatewaySubnet" {
25   name           = "GatewaySubnet"
26   address_prefix = "10.0.0.0/24"
27   resource_group_name = "${azurerm_resource_group.core.name}"
28   virtual_network_name = "${azurerm_virtual_network.core.name}"
29 }
30
31 resource "azurerm_subnet" "training" {
32   name           = "training"
33   address_prefix = "10.0.1.0/24"
34   resource_group_name = "${azurerm_resource_group.core.name}"
35 }

mark@Azure:~/clouddrive/my_terraform_sandbox$ terraform -v
Terraform v0.11.8
+ provider.azurearm v1.13.0
+ provider.random v2.0.0

mark@Azure:~/clouddrive/my_terraform_sandbox$
```

Azure Resource Manager Template Example

```
{
  "$schema": "https://schema.management.azure.com/..json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [{
    "type": "Microsoft.Resources/resourceGroups",
    "apiVersion": "2018-05-01",
    "location": "eastus",
    "name": "demo-storage",
    "properties": {}
  },
  {
    "type": "Microsoft.Storage/storageAccounts",
    "name": "demo-storage",
    "apiVersion": "2018-02-01",
    "location": "eastus",
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "properties": {}
  }
]
```



Resource Group

Storage Account

Terraform Example

```
resource "azurerm_resource_group" "testrg" {  
  name = "resourceGroupName"  
  location = "westus"  
}
```



Resource Group

```
resource "azurerm_storage_account" "testsa" {  
  name = "storageaccountname"  
  resource_group_name = "testrg"  
  location = "westus"  
  account_tier = "Standard"  
  account_replication_type = "GRS"  
}
```



Storage Account

Demo

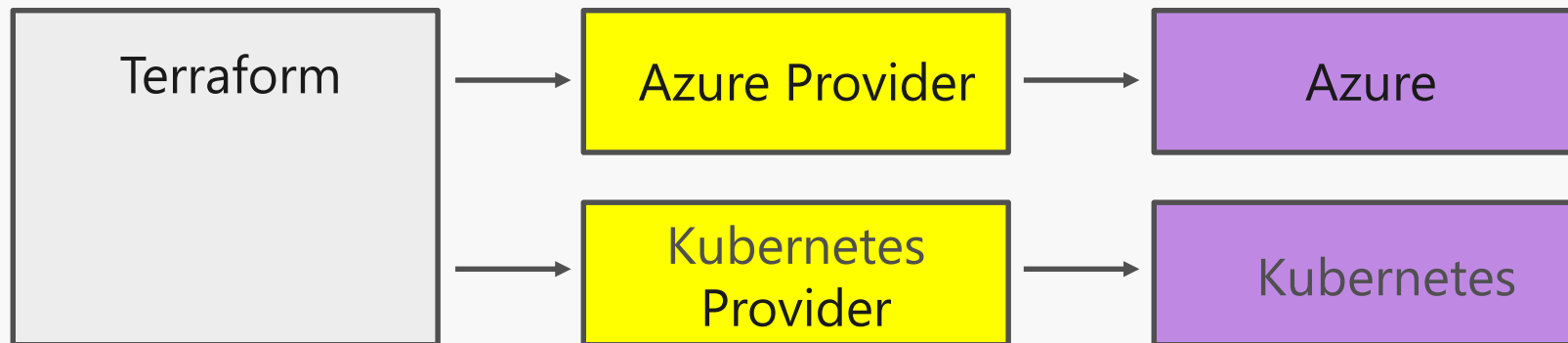
Terraform Authoring



Providers

What is a Terraform provider?

- Terraform 'extensions' for deploying resources
- Manages cloud / endpoint specific API interactions
- Available for major clouds and other platforms
- Hand authored (azurerm)



Providers

Defines how
Terraform will
interact with:

Cloud

Azure

AWS

Google

AliCloud

OpenStack

Infrastructur
e

Kubernete
s

Docker

Rancher

Network

DNS

Cloudflare

HTTP

Version
Control

GitHub

GitLab

Bitbucket

Monitoring

DataDog

Grafana

PagerDuty

Database

InfluxDB

MySQL

PostgreSQL

Etc.

Basic resource creation

Deployment foundations.

- Resource Type: required provider
- Name: internal name
- Configuration: deployment details

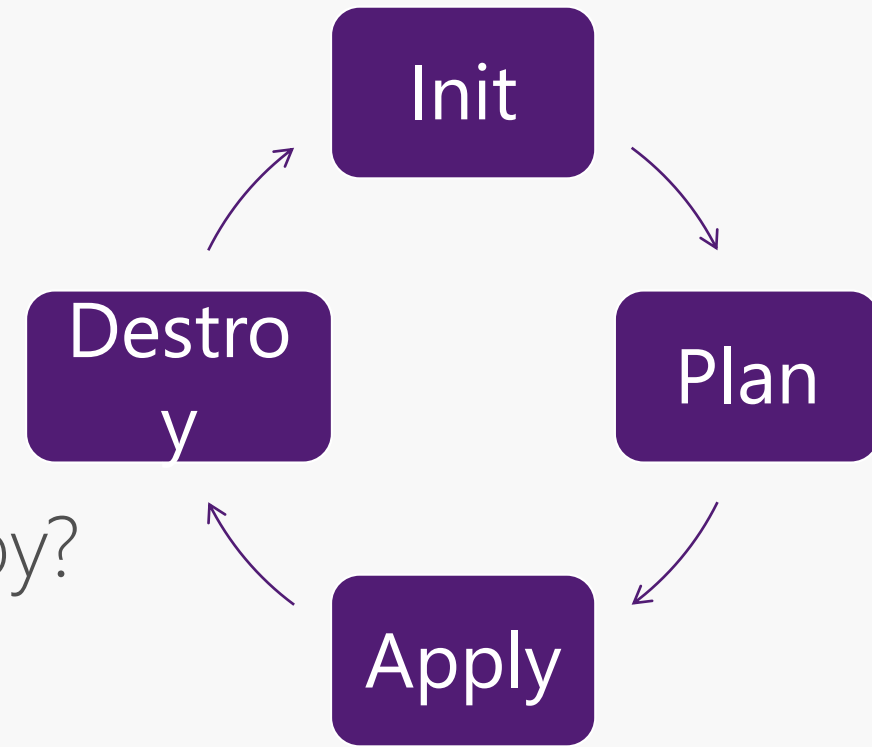
```
resource "azurerm_resource_group" "demo-rg" {  
  name = "demo-rg"  
  location = "westus"  
}
```

Resource Type

Name

Resource Configuration

Basic Terraform commands



Once we have authored, how do we deploy?

- Terraform init –初始化工作目錄
- Terraform plan – 執行前驗證
- Terraform apply –部署和更新資源
- Terraform destroy –刪除配置中定義的所有資源

Variables and output

Variables and output

- Input variables: parameters for Terraform modules
- Environment variables: TF_VAR_azureclientid

```
$ TF_VAR_azureclientid = "00000000-0000-0000-0000-000000000000"  
  
variable "azureclientid" {}
```

String Interpolation

Interpolation: the insertion of something of a different nature into something else.

- Variables
- Other resources
- Functions: `${count.index + 1}`
- Others ([Docs](#))

```
resource "azurerm_container_group" "demo-aci" {  
  name = "demo-aci"  
  location = "${azurerm_resource_group.demo-rg.location}"  
}
```

from resource

Dependencies

How are resource dependencies managed?

- Implicit – derived from interpolation
- Explicit – hard coded / explicit dependency

```
resource "azurerm_container_group" "demo-aci" {  
    name = "demo-aci"  
  
    depends_on = ["azure_cosmosdb_account.vote-db"]  
}
```

Demo

Creating Azure Services



State / Backend

State / Backend

What is Terraform state and why store it remotely?

Issues with local state:

- No collaboration
- Easy to delete / loose
- State files include secrets

Alternative:

- Store state in a backend (Azure Storage)

Azurerm backend

Standard Backend with state locking & consistency checking

Azure Storage (Blob)

remote-state.tf

```
terraform {  
  backend "azurerm" {}  
}
```

.backend.tfvars


```
storage_account_name = "tfbackend4mcg"  
container_name = "tfstate"  
key = "sandbox.terraform.tfstate"  
access_key = "
```


Data Sources

What is a Terraform data source?

- 用於Terraform配置的外部來源數據
- 就像在資源創建中一樣使用提供程序

```
data "terraform_remote_state" "azurerm" {  
    <configuration goes here>  
}
```



```
"${data.terraform_remote_state.azurerm.resource-group}"
```

Terraform Module

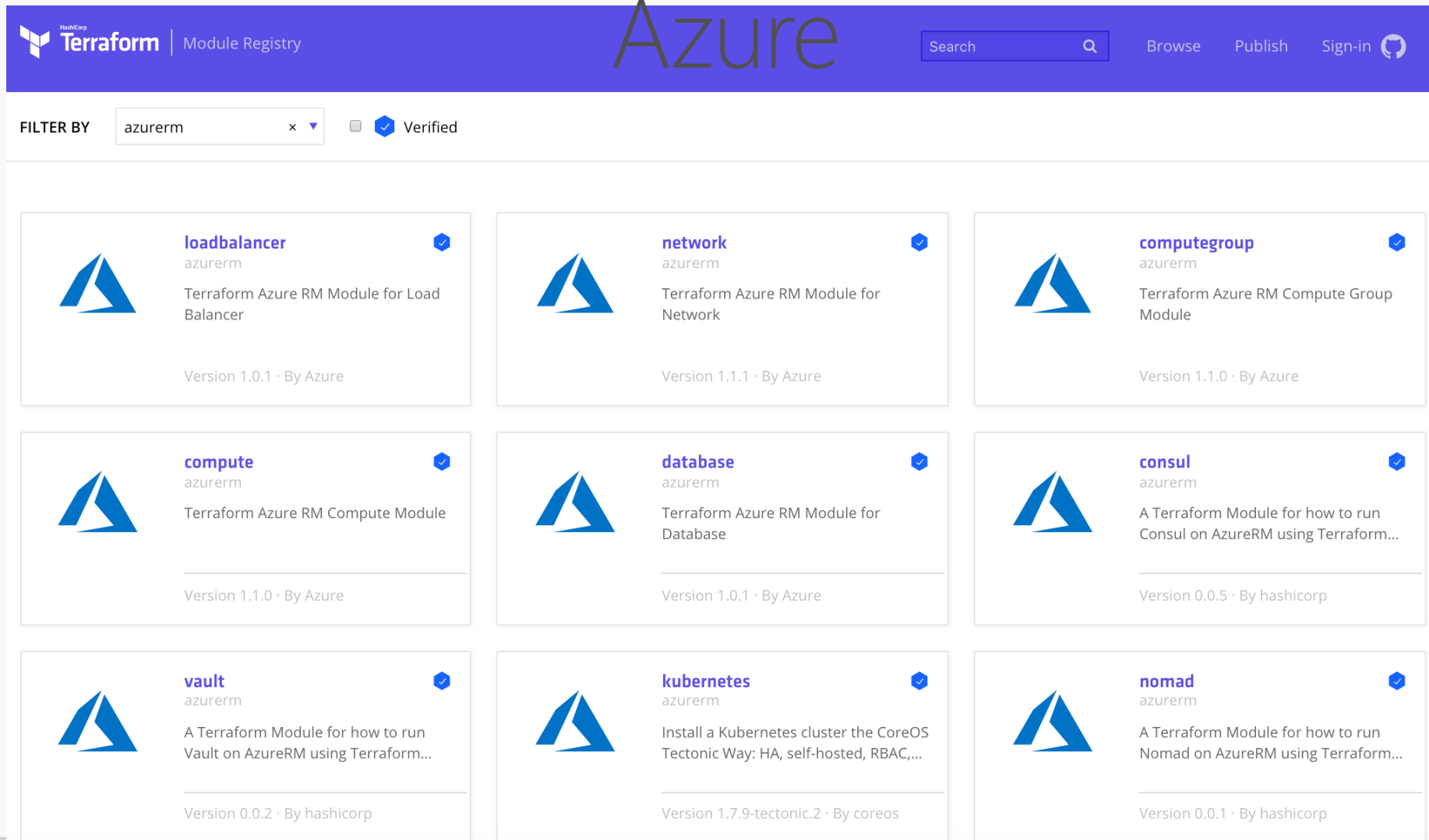


Modules

```
module "MyApp-Network-AWS" {  
  source      = "terraform-aws-modules/vpc/aws"  
  
  name        = "network1"  
  
  tags = {  
    environment = "dev"  
    costcenter  = "it"  
  }  
}
```

```
resource "module" "MyApp-Network-Azure" {  
  name      = "MyApp-Network-Azure"  
  source    = "Azure/network/azurerm"  
  
  vnet_name = "network1"  
  location  = "westus"  
  
  tags = {  
    environment = "dev"  
    costcenter  = "it"  
  }  
}
```

Terraform Module Registry and Microsoft Azure



The screenshot shows the Terraform Module Registry interface for Azure. The header is purple with the Terraform logo, 'Module Registry' text, a search bar, and links for 'Browse', 'Publish', and 'Sign-in'. Below the header, a filter bar shows 'FILTER BY' with a dropdown set to 'azurerm' and a 'Verified' checkbox. The main content area displays a grid of nine module cards, each featuring the Terraform logo, a title, a description, and version information.

Module Name	Provider	Description	Version	Author
loadbalancer	azurerm	Terraform Azure RM Module for Load Balancer	Version 1.0.1	By Azure
network	azurerm	Terraform Azure RM Module for Network	Version 1.1.1	By Azure
computegroup	azurerm	Terraform Azure RM Compute Group Module	Version 1.1.0	By Azure
compute	azurerm	Terraform Azure RM Compute Module	Version 1.1.0	By Azure
database	azurerm	Terraform Azure RM Module for Database	Version 1.0.1	By Azure
consul	azurerm	A Terraform Module for how to run Consul on AzureRM using Terraform...	Version 0.0.5	By hashicorp
vault	azurerm	A Terraform Module for how to run Vault on AzureRM using Terraform...	Version 0.0.2	By hashicorp
kubernetes	azurerm	Install a Kubernetes cluster the CoreOS Tectonic Way: HA, self-hosted, RBAC,...	Version 1.7.9-tectonic.2	By coreos
nomad	azurerm	A Terraform Module for how to run Nomad on AzureRM using Terraform...	Version 0.0.1	By hashicorp

Demo

Terraform Module for Azure



Demo

AKS + KafKa

01010101

01010101

01010101



Terraform

Azure DevOps Tool Integrations

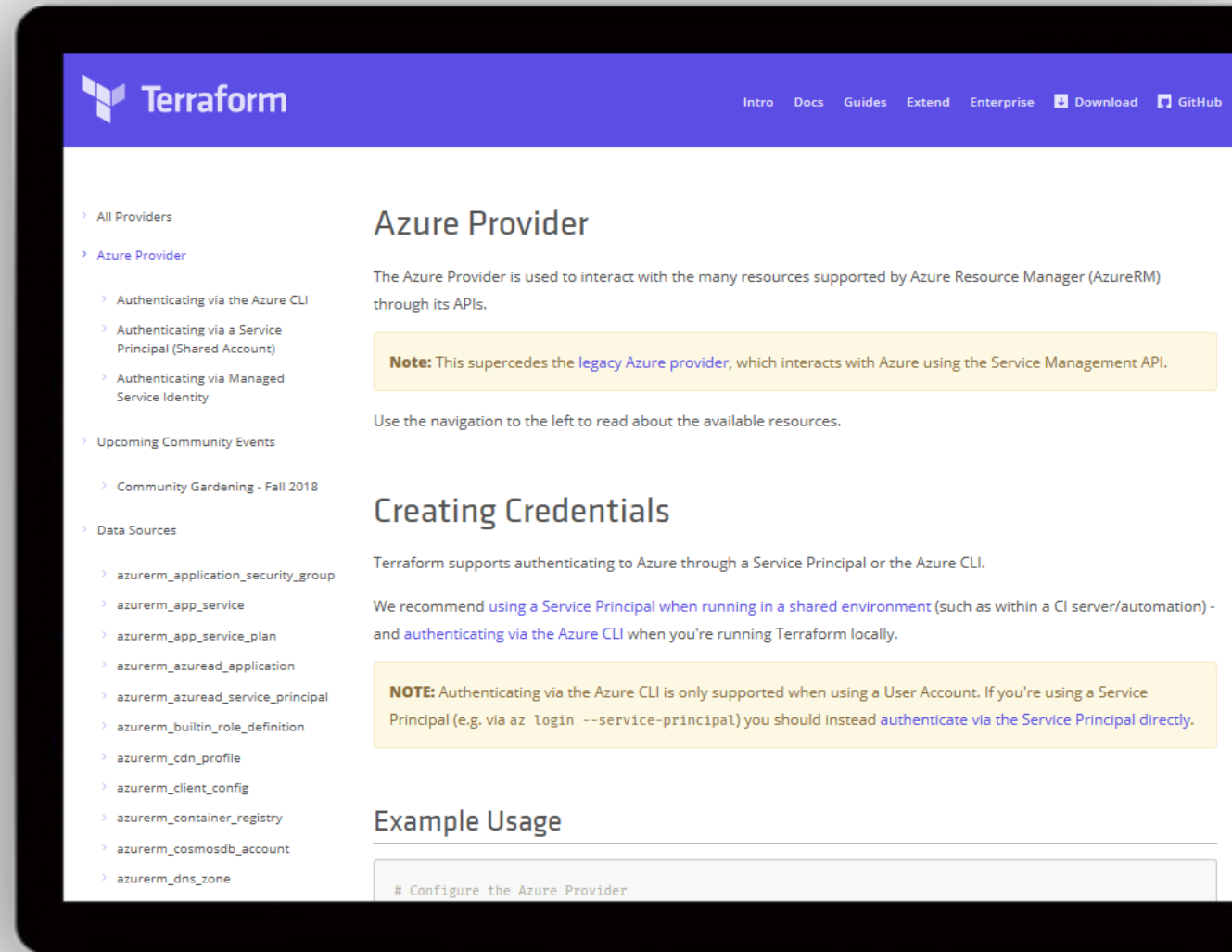


Bringing native Azure support for customers using Terraform

- [Documentation Hub for Terraform](#)
- [Terraform in Azure Cloud Shell](#)
- [Azure Resource Provider](#)
- [Azure Module Registry](#)
- [Azure Cloud Shell Integration](#)

 docs.microsoft.com/azure/terraform

© Microsoft Corporation



Azure

Session resources

Docs – <http://aka.ms/terraform>

Code – <http://aka.ms/tfgit>

Articles – <http://aka.ms/tfhub>

Solution – <http://aka.ms/azdotf>

Advisors – email me (perltsai223@gmail.com)

特別感謝



R-Ladies Taipei



多奇·數位創意



以及各位參與活動的你們

