

Microsoft Cloud Adoption Framework for Azure

Optimize your organization with DevOps and
Terraform landing zones

Arnaud Lheureux
Lead Engineer – CAF Terraform
Strategic Partnerships
Microsoft

Agenda

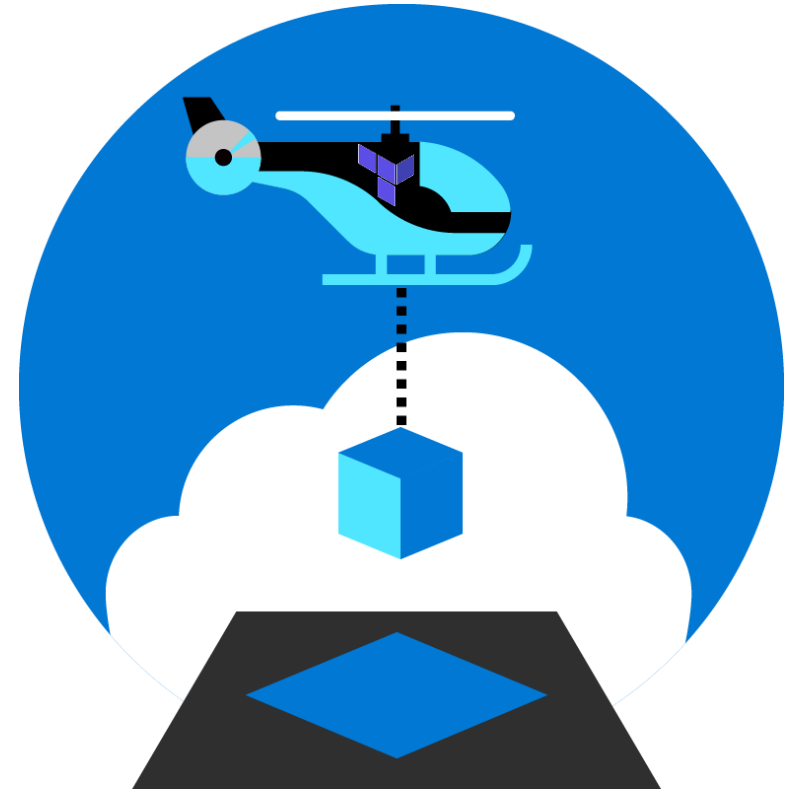
- Quick Introduction to Azure landing zones and architecture blueprint
- Introduction to CAF Terraform landing zones concepts and tools

Azure landing zones

Help customers **set up their Azure environment**—
for **scale, security, governance, networking, and identity**

Azure landing zones:

- Enable **migrations** and **net new apps**
- Consider **all platform resources**
- **Don't differentiate** between **IaaS** or **PaaS**



Azure landing zones

Design areas



Azure billing &
Active Directory tenant

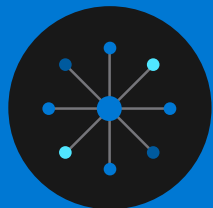


Identity & access
management

ENVIRONMENT



Resource organization



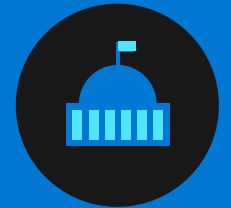
Network topology
& connectivity

COMPLIANCE

Security



Governance



Management

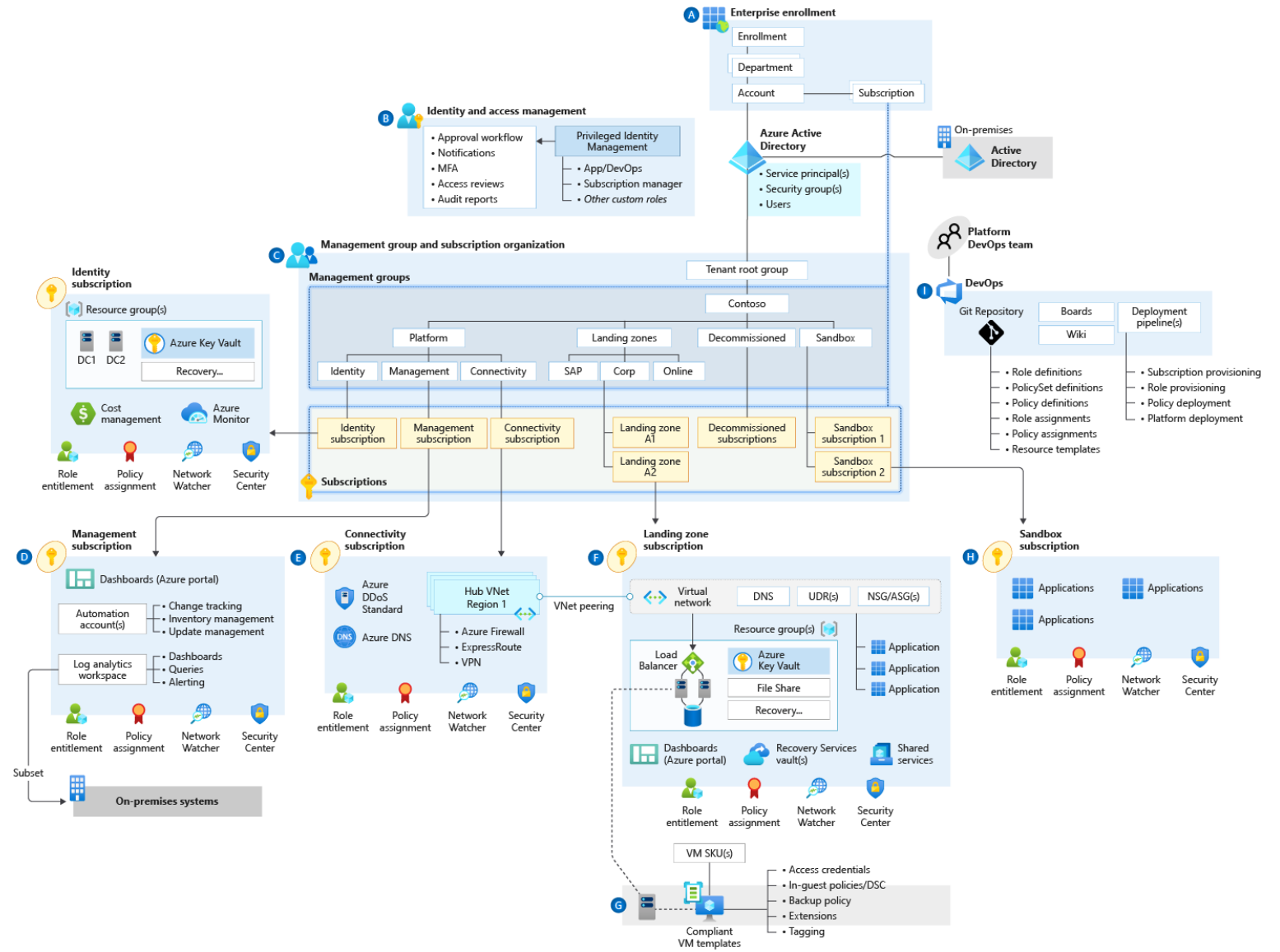


Platform automation
& DevOps



Azure landing zone—conceptual architecture

- Target end-state for the majority of organizations
- Scaled-out, mature environment
- Represents broad range of Microsoft best practices for Azure environment design
- Provides strong foundation for organizations to establish on-going management, governance and security processes



Everything-as-code



Stand up environments in the fastest possible way



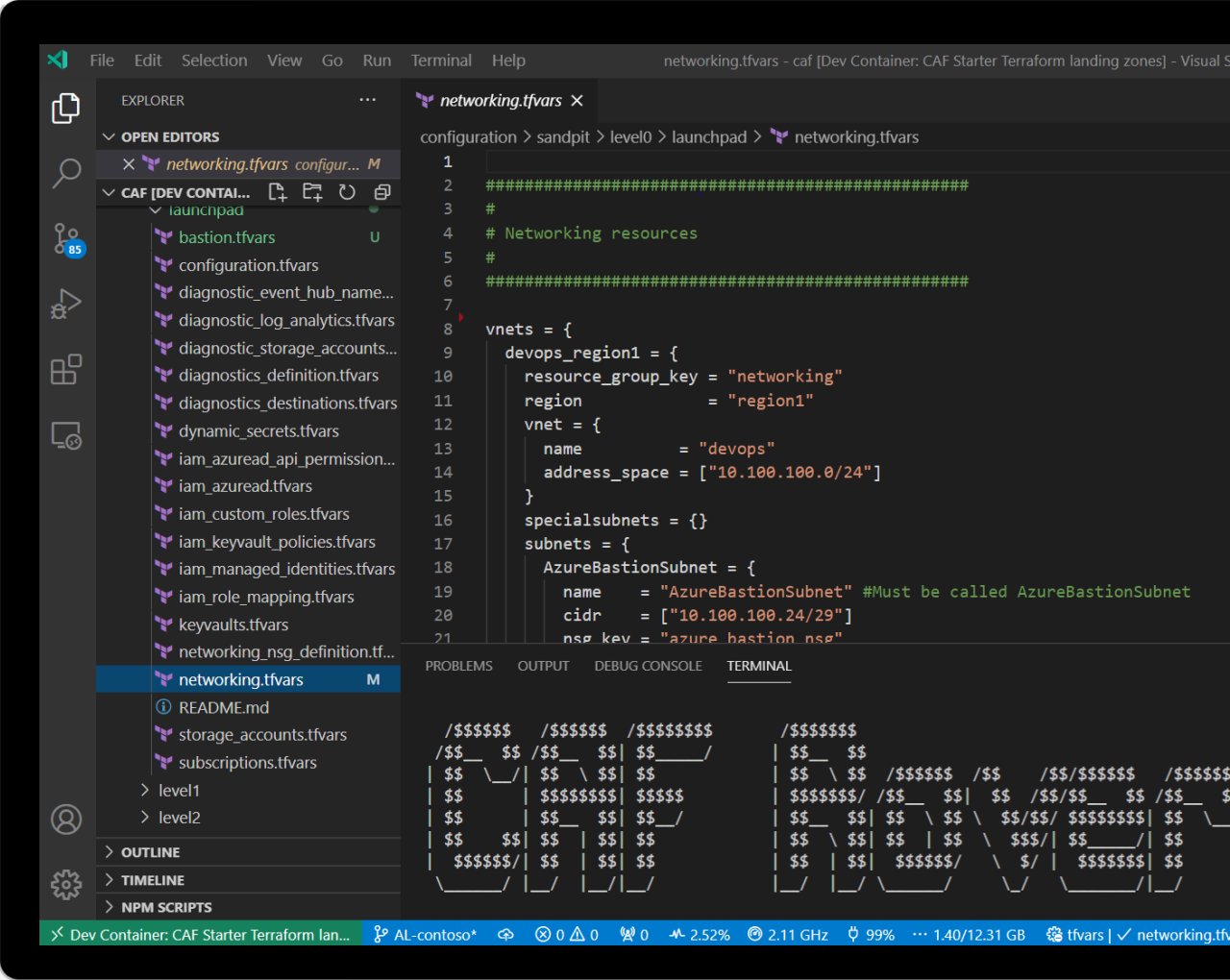
Remove the human element and reliably and repeatable deploy every time



Improve environment visibility and improve developer efficiency



Store your configuration definitions alongside your application code



Why Terraform for Azure landing zones?

- Declarative code: Collaboration enabler among team members
- Providers and skills ecosystem
- State management and operations predictability
- Version control



Why CAF Terraform landing zones

Why do I need a framework when I can just Terraform.exe?

Not everyone is a developer

It's hard to ensure: consistency, readability, maintainability, reusability

Immutable infrastructure requires: centralizing knowledge, experience, features into Terraform

Deliver value to your customers, not modules!

State management and delegation in complex organizations

```
1 #####
2 # These resources will create an additional subnet for user connectivity
3 # and a Linux Server to use with the Bastion Service.
4 | 1 #####
5 | 2 # These resources will create an additional subnet for user connectivity
6 | 3 # and a Linux Server to use with the Bastion Service.
7 | 4 | 1 #####
8 | 5 | 2 # These resources will create an additional subnet for user connectivity
9 | 6 | 3 # and a Linux Server to use with the Bastion Service.
10 | 7 | 4 #####
11 | 8 | 5 | 1 #####
12 | 9 | 6 # Dev 2 # These resources will create an additional subnet for user connectivity
13 | 10 | 7 # (Add 3 # and a Linux Server to use with the Bastion Service.
14 | 11 | 8 resour 4 #####
15 | 12 | 9 name 5
16 | 13 | 10 reso 6 # Dev Subnet
17 | 14 | 11 virt 7 # (Additional subnet for Developer Jumpbox)
18 | 15 | 12 addr resource "azurerm_subnet" "dev" {
19 | 16 | 13 enfo 9 name = "devSubnet"
20 | 17 | 14 10 resource_group_name = azurerm_resource_group.rg.name
21 | 18 | 15 } 11 virtual_network_name = azurerm_virtual_network.vnet.name
22 | 19 | 16 12 address_prefixes = ["10.0.4.0/24"]
23 | 20 | 17 resour 13 enforce_private_link_endpoint_network_policies = false
24 | 21 | 18 name 14
25 | 22 | 19 reso 15 }
26 | 23 | 20 loca 16
27 | 24 | 21 17 resource "azurerm_network_security_group" "dev-nsg" {
28 | 25 | 22 18 name = "${azurerm_virtual_network.vnet.name}-${azurerm_subnet.dev.name}"
29 | 26 | 23 19 resource_group_name = azurerm_resource_group.rg.name
30 | 27 | 24 20 location = azurerm_resource_group.rg.location
31 | 28 | 25 21
32 | 29 | 26 22 security_rule {
33 | 30 | 27 23 name = "SSH"
34 | 31 | 28 24 priority = 1001
35 | 32 | 29 25 direction = "Inbound"
36 | 33 | 30 26 access = "Allow"
37 | 34 | 31 27 protocol = "Tcp"
38 | 35 | 32 28 source_port_range = "*"
39 | 36 | 33 29 destination_port_range = "22"
40 | 37 | 34 30 source_address_prefix = var.source_address_prefix
41 | 38 | 35 31 destination_address_prefix = "*"
42 | 39 | 36 32 }
```

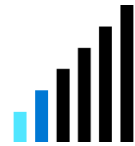
Core principles

Fundamental building blocks



Enterprise as Configuration

Whatever you need, don't write code, just configuration files.



Transparent composition

Read or Write Terraform states easily between landing zones.



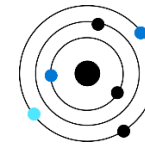
Hierarchy and delegation

Proposed hierarchical model of Terraform state files allow enterprise composition and delegates innovation to business units.



Easy state management

Just code, let rover put the state at the right place and ensure its safety and resiliency.



Developer productivity

Run your code on your laptop or in your pipelines – just the same way.

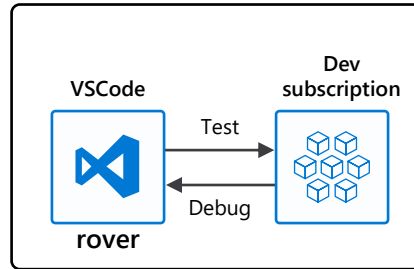


DevOps Ubiquity

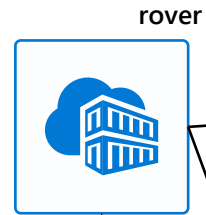
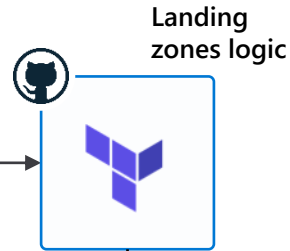
Run on any pipeline and CI/CD.

Seamless development experience

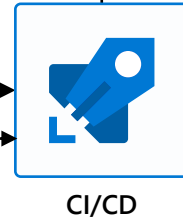
Ubiquitous Inner feedback loop



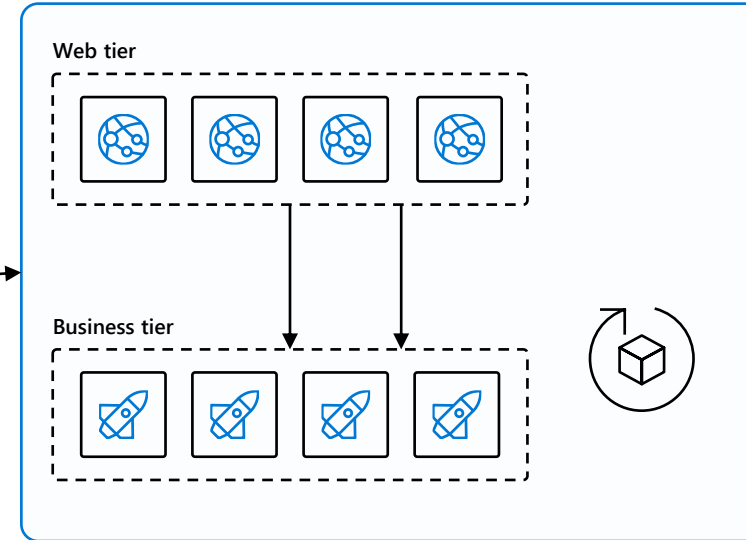
Windows
Linux
Mac
GitHub Codespaces



Azure Pipelines,
GitHub Actions,
etc.



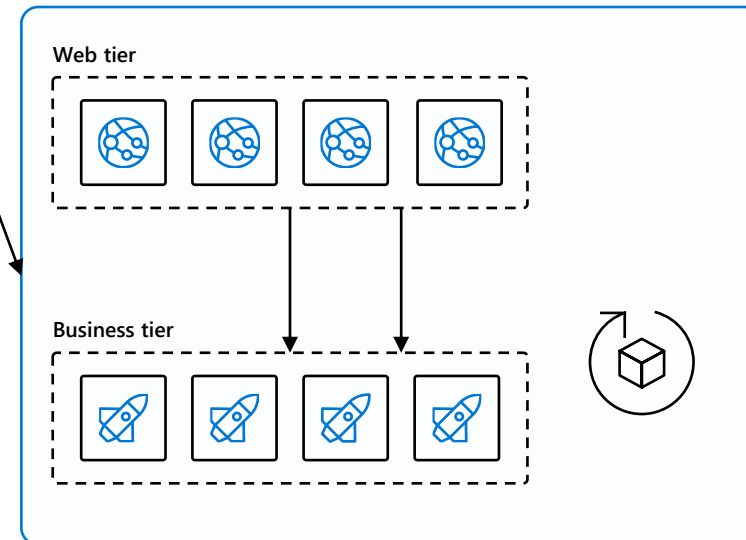
production environment



Azure
Monitor



dev, qa, etc. environment



Any CI/CD



CAF SRE Stack on Azure

Application landing zones

Project custom code (business application)
Infrastructure IaC project specific

Databricks, Data factory, ML workspace,
Application landing zones (platform ops solution accelerators) (AKS, App Service, Data analytics, etc.)

Azure Subscription Vending Machine

Landing zone factory, bridge to the platform,
solutions accelerators

Virtual network, RBAC mapping, backup store and policies, delegated identities for pipelines

Platform control plane

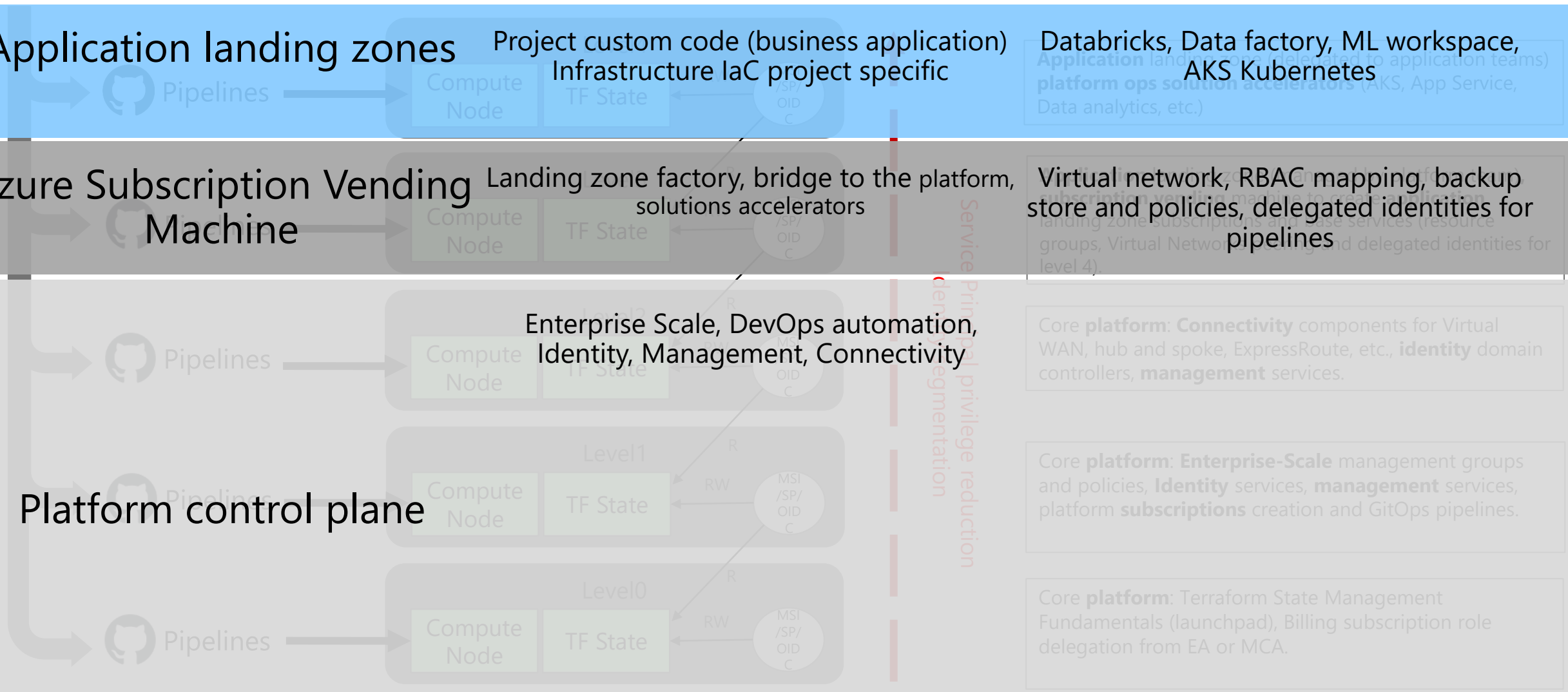
Enterprise Scale, DevOps automation,
Identity, Management, Connectivity

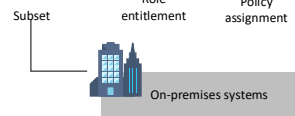
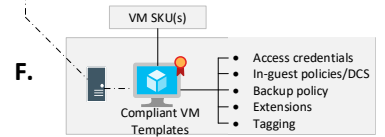
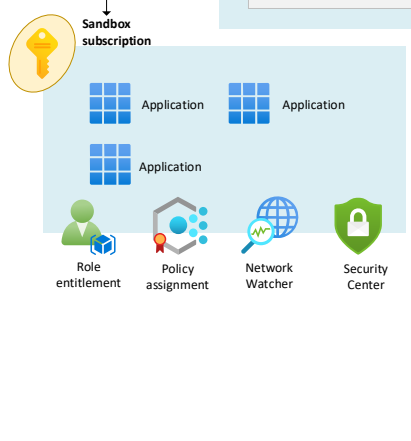
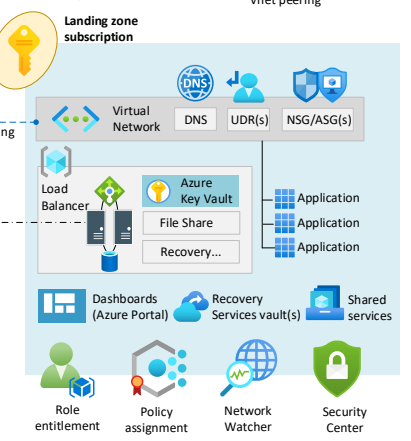
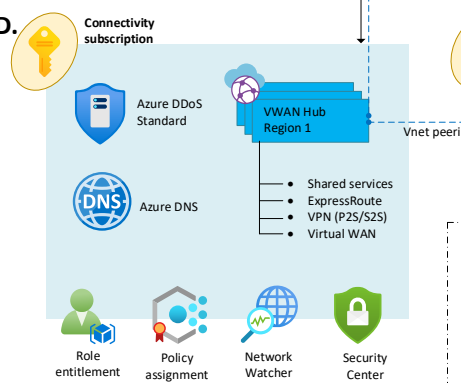
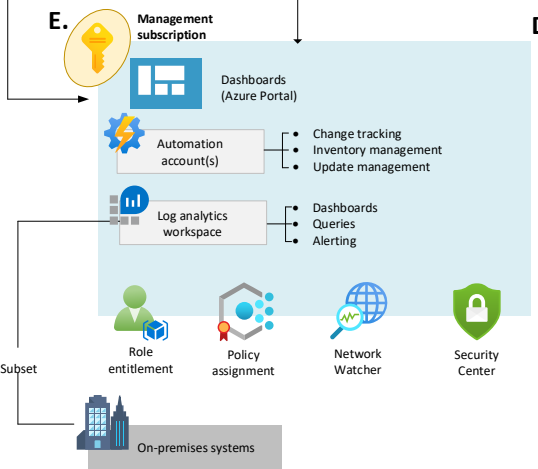
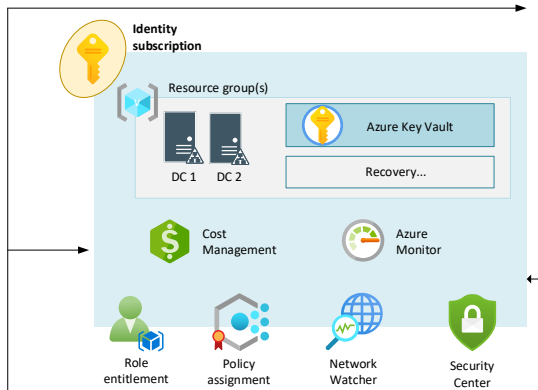
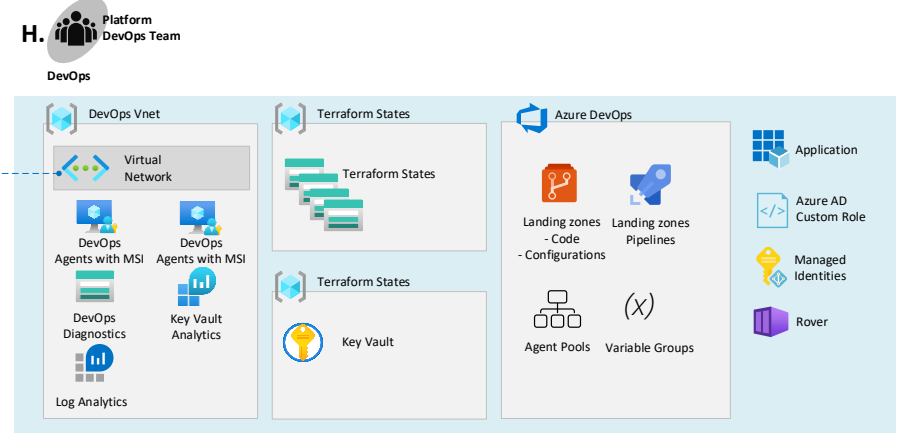
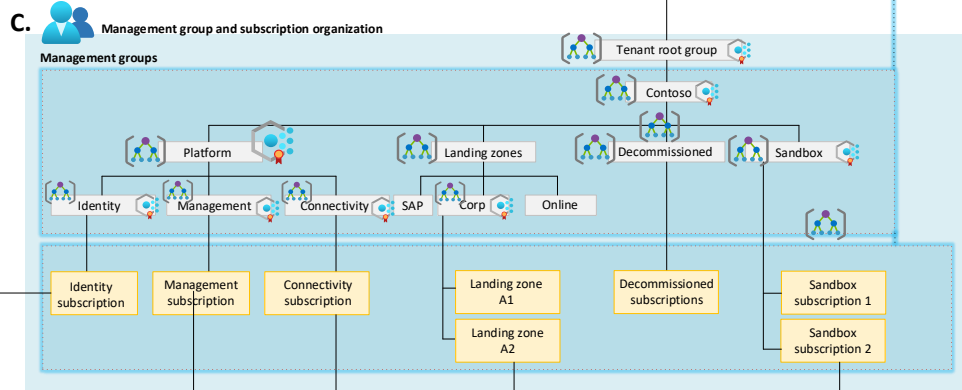
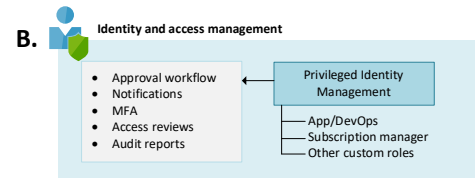
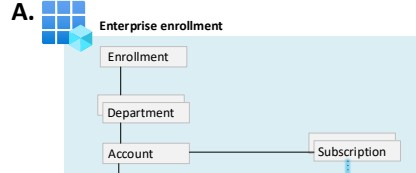
Core platform: **Connectivity** components for Virtual WAN, hub and spoke, ExpressRoute, etc., **identity** domain controllers, **management** services.

Core platform: **Enterprise-Scale** management groups and policies, **Identity** services, **management** services, platform **subscriptions** creation and GitOps pipelines.

Core platform: Terraform State Management Fundamentals (launchpad), Billing subscription role delegation from EA or MCA.

Service Principal privilege reduction
Identity Segmentation





Overall process

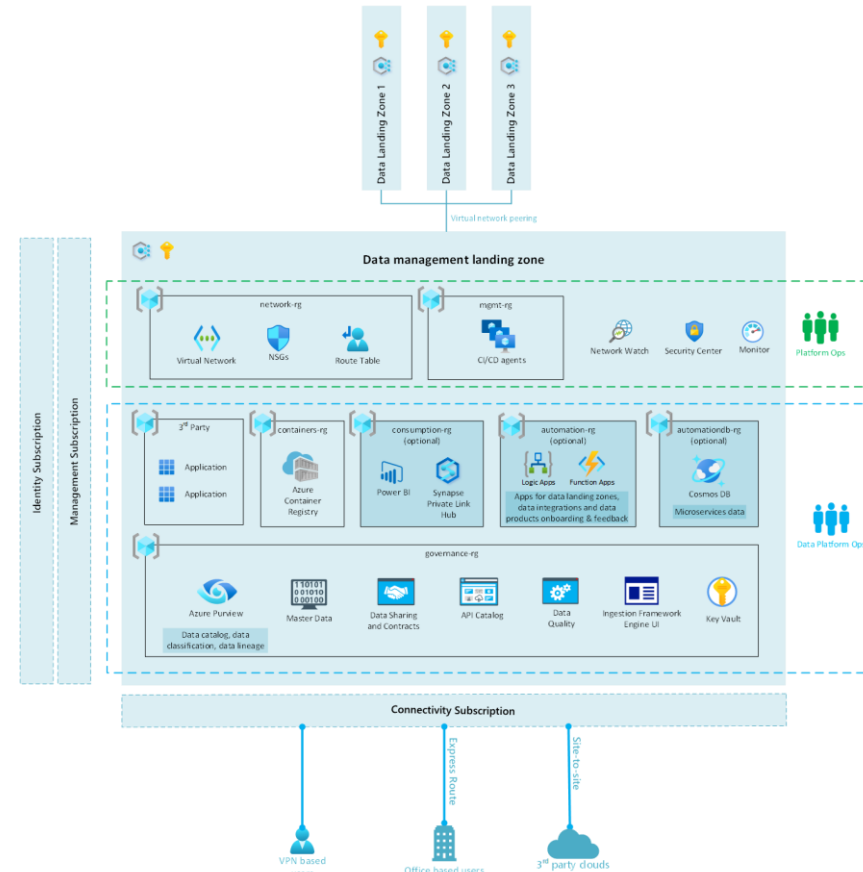
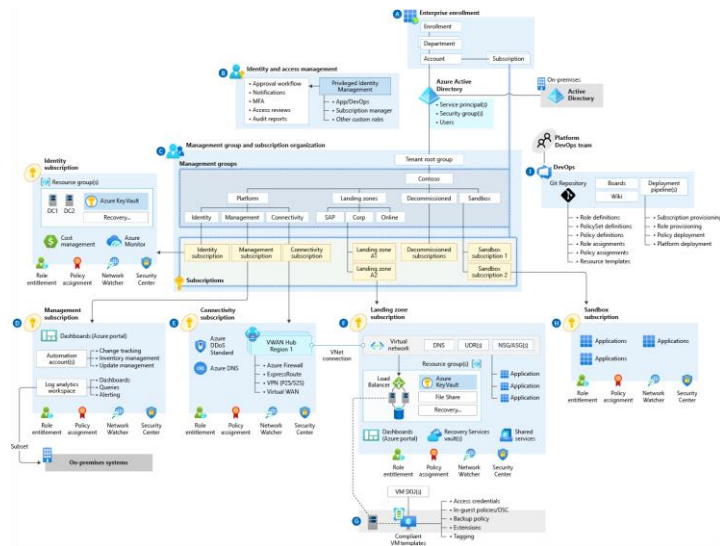
Build the Azure Platform Enterprise Scale



Create the landing zones

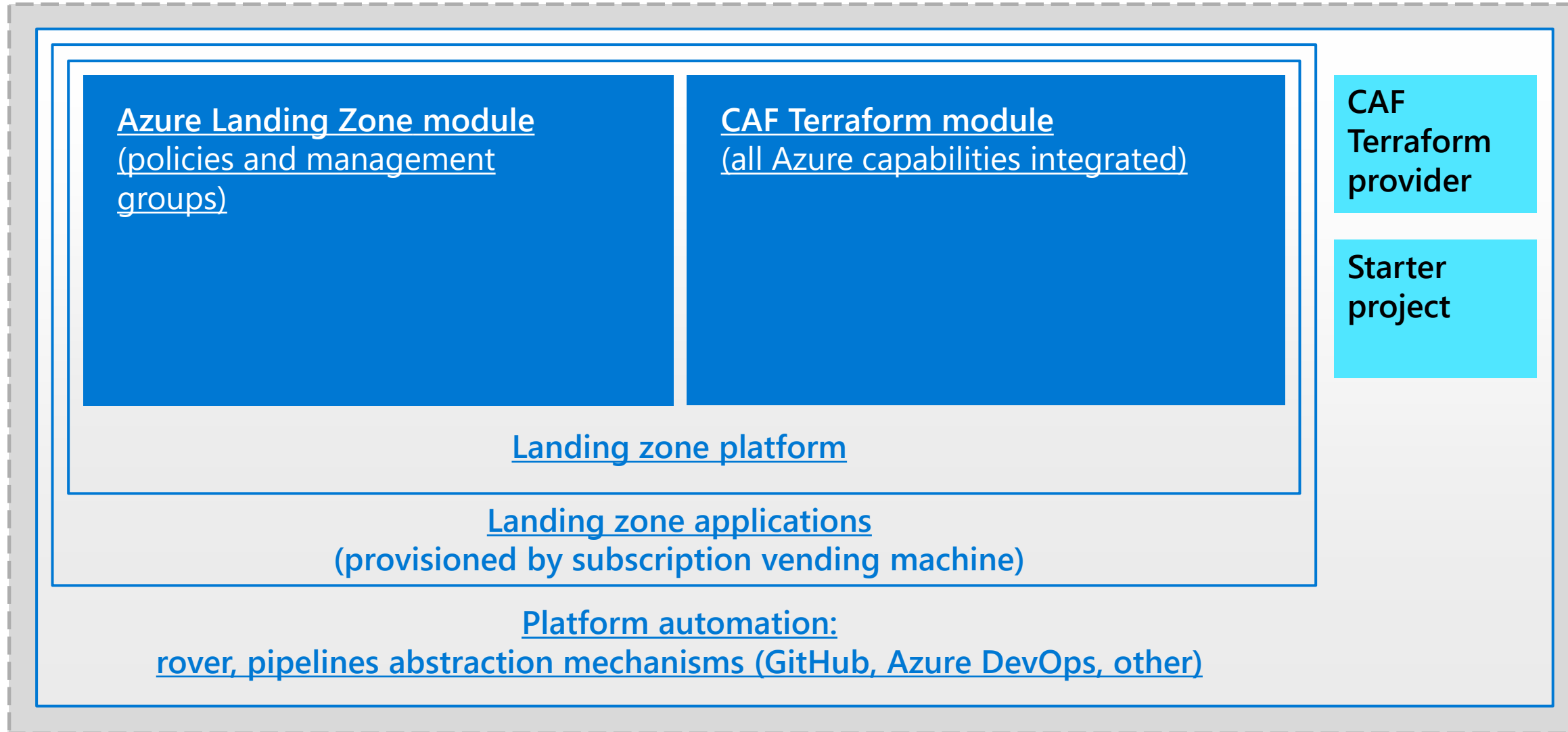


Deploy Solution landing zone Accelerators

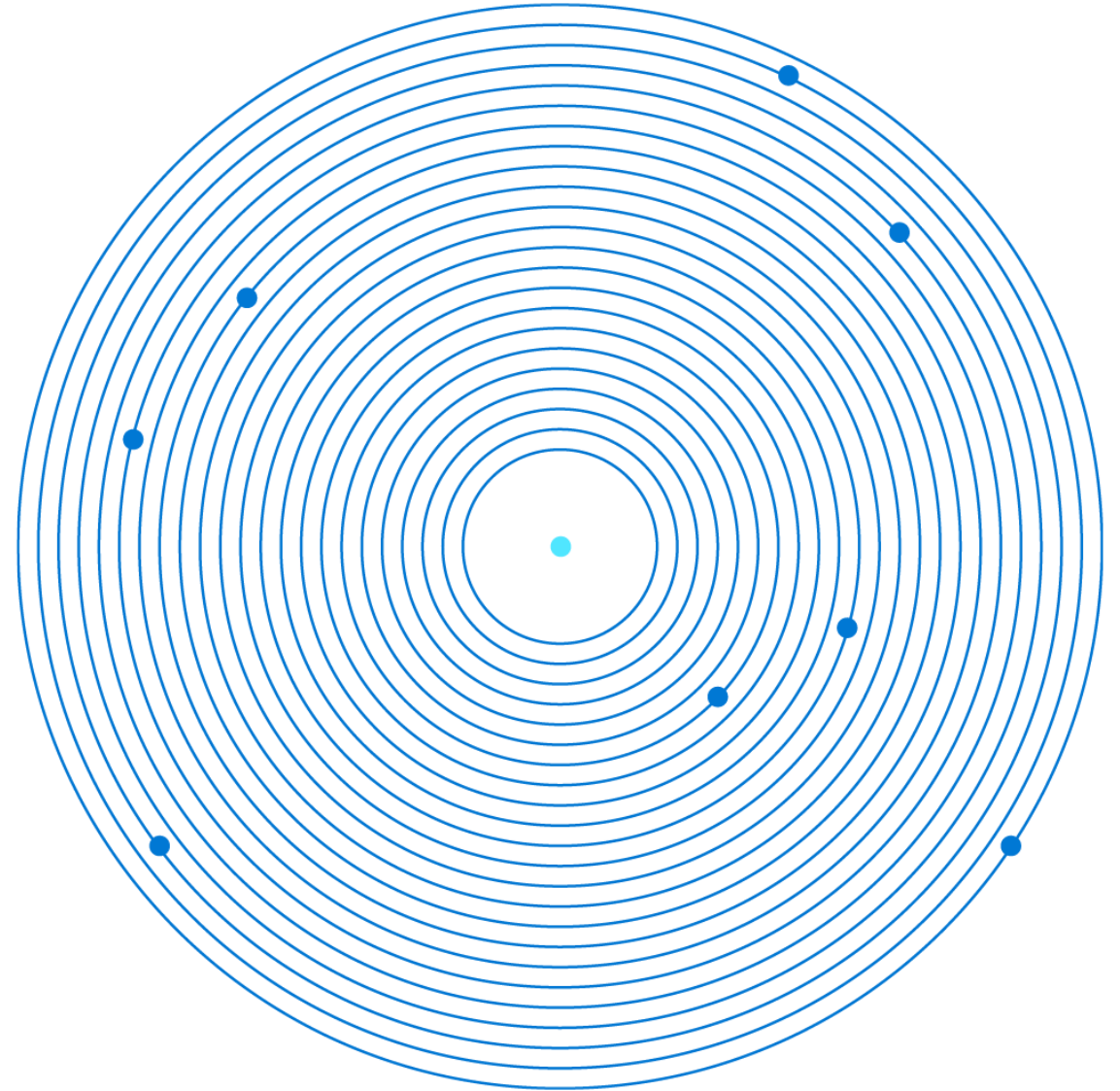


Your SRE toolbox

CAF Terraform: Site reliability engineering components



Zoom on CAF module



Why using CAF module?

Bespoke vs Standard Terraform

Everyone can do a Terraform module

Difficult part #1: the integration work

Difficult part #2: adding capabilities, maintenance over time, and testing

IaC: Infrastructure-as-Configuration

Not everyone wants to write code, so -> configuration is the contract

Focus on deploying new features with preserved configuration

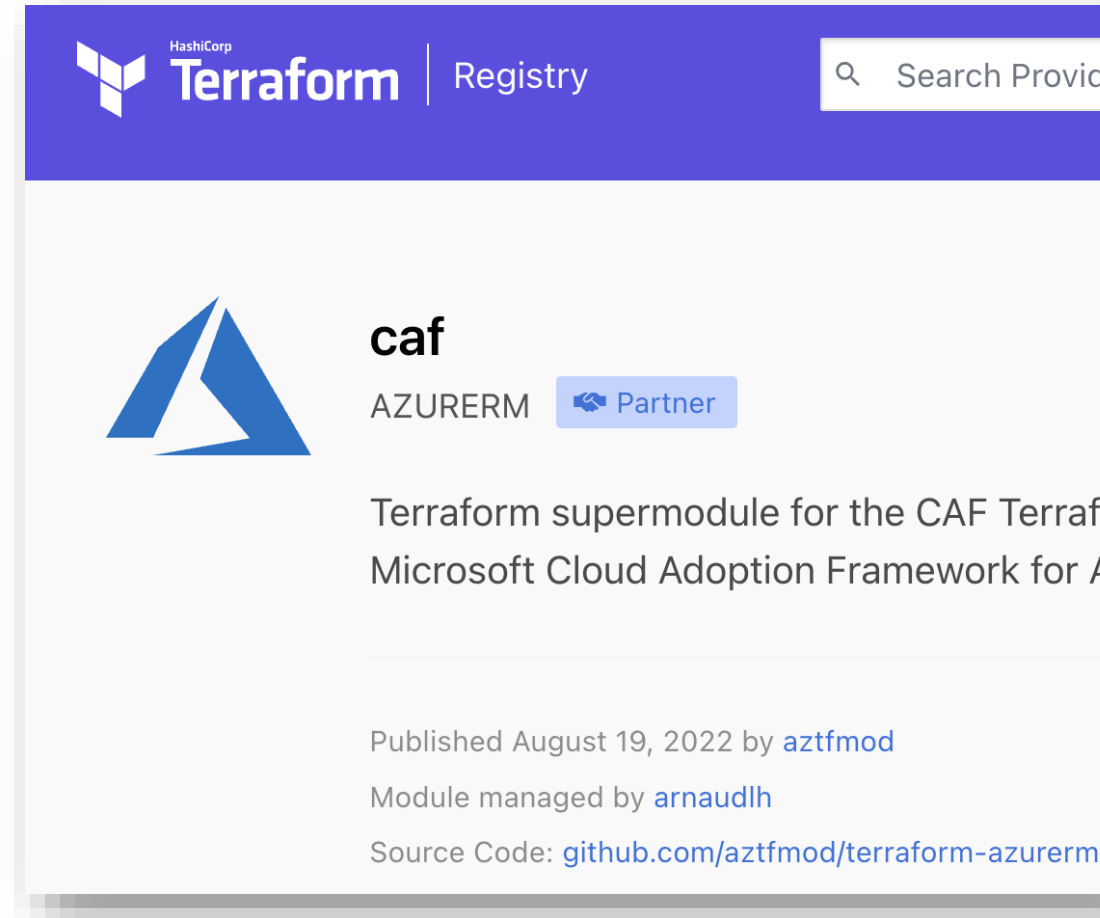
Tested and validated against regressions

Fully Declarative and Iterative

Just declare variable, we iterate and compose for you

CAF Terraform module capabilities

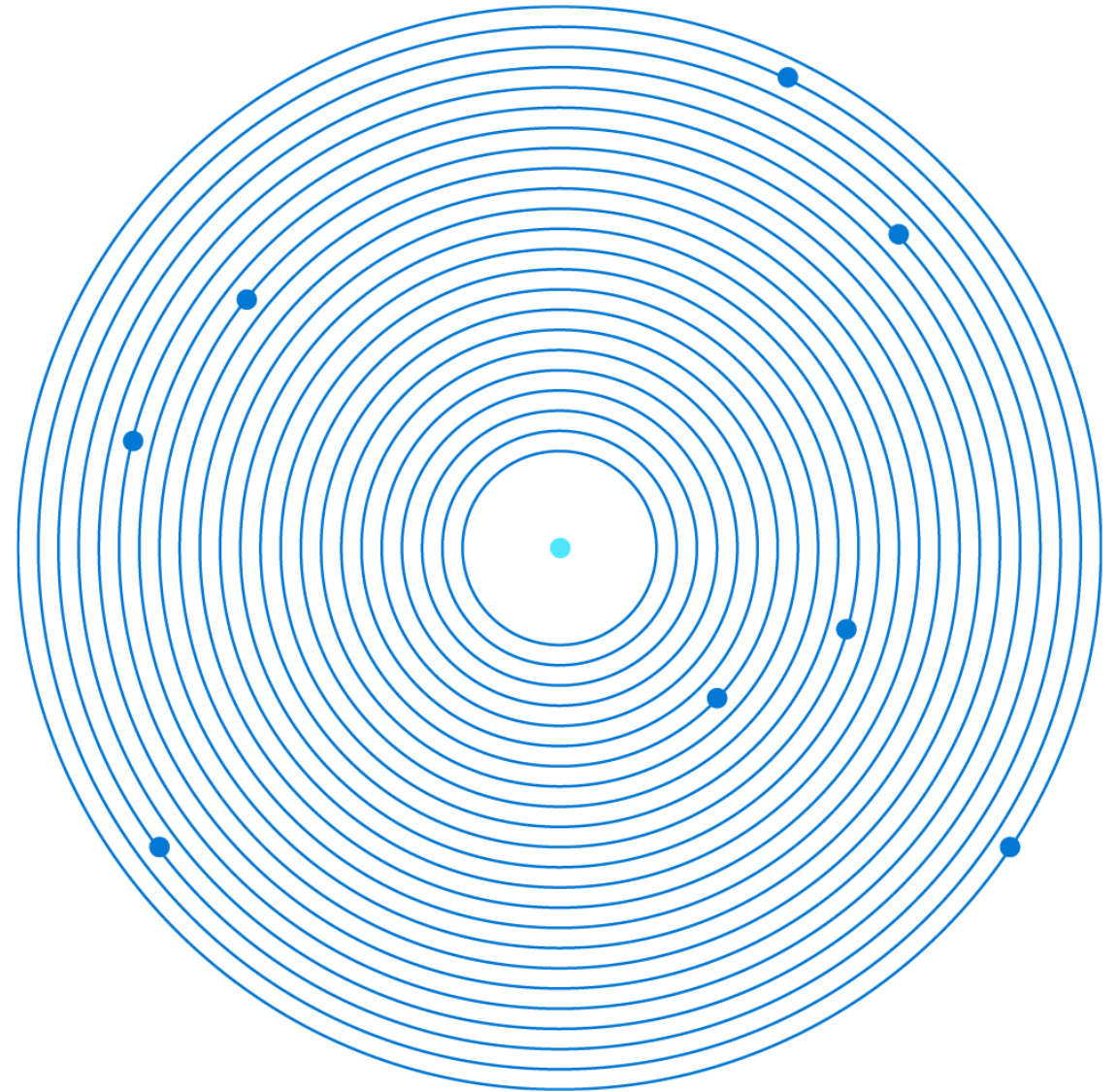
- #1 VERIFIED module for AzureRM and AzAPI with 1 M provisions
- Fully Open Source
- 80+ contributors from Microsoft engineers, partners and customers
- Iterative by design, works on all version of Terraform starting 0.14 (up to current 1.1.3)
- Leveraging key-association pattern for easy composition within all* Azure capabilities
- Useable with or without rover



The screenshot shows the Terraform Registry interface. At the top, there is a blue header with the HashiCorp Terraform logo and the word 'Registry'. A search bar is located on the right side of the header. Below the header, the main content area features a blue triangle icon on the left. To the right of the icon, the module name 'caf' is displayed in a large, bold font. Below the name, the provider 'AZURERM' is listed, followed by a blue 'Partner' badge. A descriptive paragraph follows, stating 'Terraform supermodule for the CAF Terraform Microsoft Cloud Adoption Framework for Azure'. Below this, the publication date 'Published August 19, 2022 by aztfmod' and the manager 'Module managed by arnaudlh' are shown. At the bottom, the source code link 'Source Code: github.com/aztfmod/terraform-azurerm' is provided.

[aztfmod/caf/azurerm | Terraform Registry](https://registry.terraform.io/modules/aztfmod/caf/azurerm)

Zoom on starter repository



Starter repository

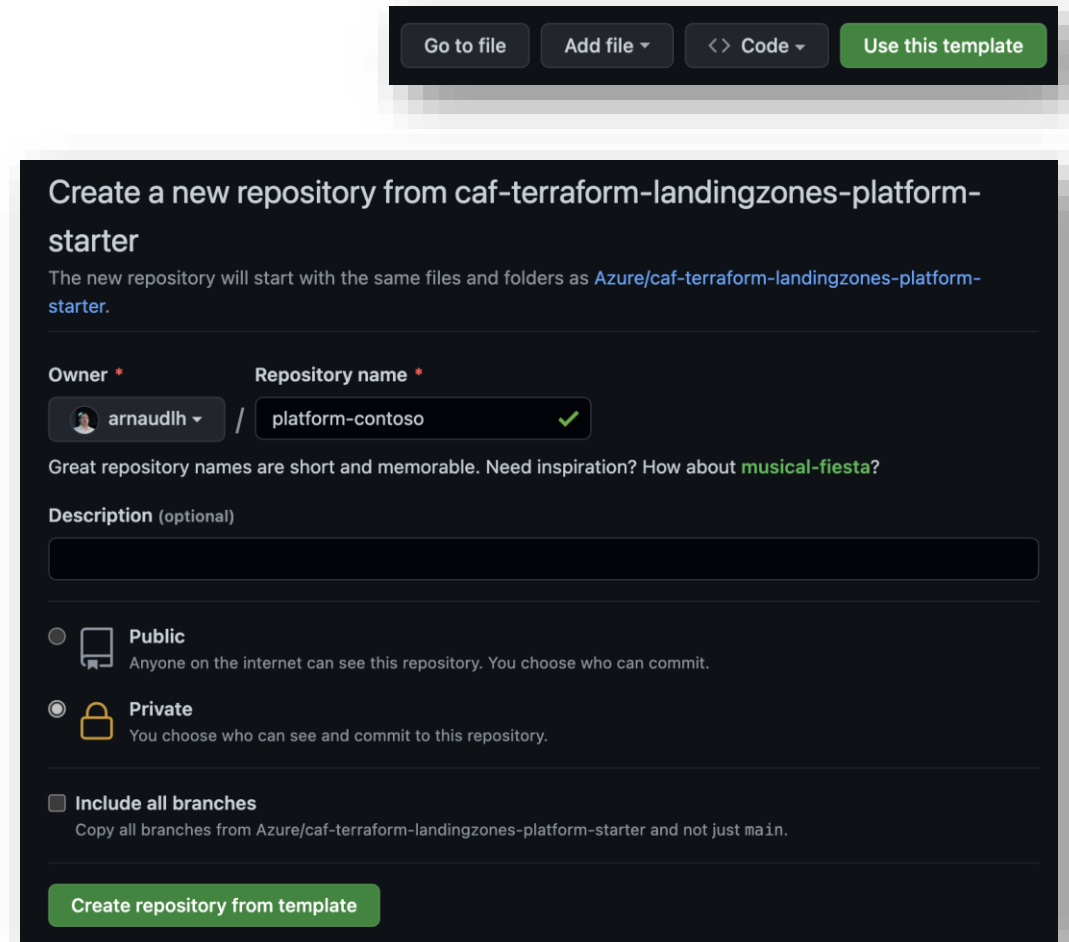
Sample of configuration repository
Ready to be cloned and started
Typically in organizations: 1 repo for platform, many repos for applications

Contains:

1. Rover version
2. Templates

[Azure/caf-terraform-landingzones-platform-starter: CAF Terraform landing zone - platform configuration starter kit \(github.com\)](https://github.com/Azure/caf-terraform-landingzones-platform-starter)

[Azure/caf-terraform-landingzones-starter: Starter project for Cloud Adoption Framework for Azure landing zones on Terraform \(github.com\)](https://github.com/Azure/caf-terraform-landingzones-starter)



The screenshot shows the GitHub interface for creating a new repository from a template. At the top, there are navigation buttons: 'Go to file', 'Add file', '<> Code', and 'Use this template'. The main heading is 'Create a new repository from caf-terraform-landingzones-platform-starter'. Below this, a note states: 'The new repository will start with the same files and folders as Azure/caf-terraform-landingzones-platform-starter.' The form includes fields for 'Owner' (set to 'arnaudlh') and 'Repository name' (set to 'platform-contoso'). A suggestion for a repository name, 'musical-fiesta?', is provided. There is an optional 'Description' field. The visibility is set to 'Private'. The 'Include all branches' checkbox is checked. A green button at the bottom says 'Create repository from template'.

Demo: getting started(d|r)



CAF Terraform module

Example of key-association patterns

```
resource_groups = {
  vnet_region1 ← { ← key
    name = "vnet-hub-region1"
  }
  vnet_region2 = {
    name = "vnet-hub-region2"
  }
}

vnets = {
  hub_sg = {
    resource_group_key = "vnet_region1" ← key reference
    vnet = {
      name = "hub"
      address_space = ["10.10.100.0/24"]
    }
    subnets = {
      jumphost = {
        name = "jumphost"
        cidr = ["10.10.100.0/25"]
        nsg_key = "jumphost"
      }
      spoke1 = {}
      spoke2 = {}
    }
  }
}
```

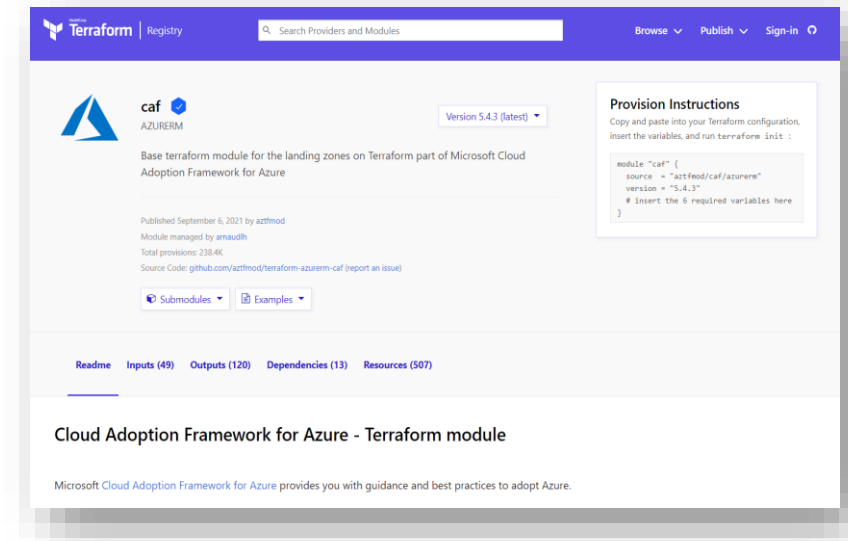
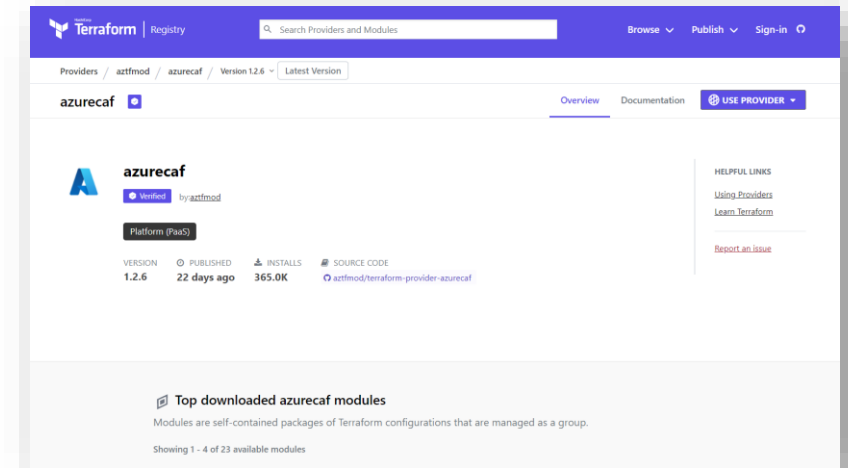
```
module "networking" {
  source = "./modules/networking/virtual_network"
  for_each = try(var.networking.vnets, {})

  location = lookup(each.value, "region", null) == null ? module.resource_g
  resource_group_name = module.resource_groups[each.value.resource_group_key].name
  settings = each.value
  network_security_group_definition = local.networking.network_security_group_definition
  route_tables = module.route_tables
  tags = try(each.value.tags, null)
  diagnostics = local.diagnostics
  global_settings = local.global_settings
  ddos_id = try(azurerem_network_ddos_protection_plan.ddos_protection_plan
}
```

```
# Establish a peering with the devops vnet
hub_rg1-T0-launchpad_devops = {
  name = "hub_rg1-T0-devops_region1"
  from = {
    vnet_key = "hub_rg1"
  }
  to = {
    tfstate_key = "foundations"
    lz_key = "launchpad"
    output_key = "vnets"
    vnet_key = "devops_region1"
  }
  allow_virtual_network_access = true
  allow_forwarded_traffic = false
  allow_gateway_transit = false
  use_remote_gateways = false
}
```


Getting started with CAF Terraform landing zones

- Explore the Azure landing zone section in CAF – <https://aka.ms/adopt/landingzones>
- CAF Terraform landing zones documentation – <https://aka.ms/caf/terraform>
- Explore the CAF Terraform modules in the Terraform registry – <https://aka.ms/terraformio>
- Hashicorp: Industrialized Workflows - Using Microsoft CAF Patterns and Terraform- [Industrialized Workflows - Using Microsoft CAF Patterns and Terraform \(hashicorp.com\)](https://aka.ms/industrialized-workflows-using-microsoft-caf-patterns-and-terraform)



Case study

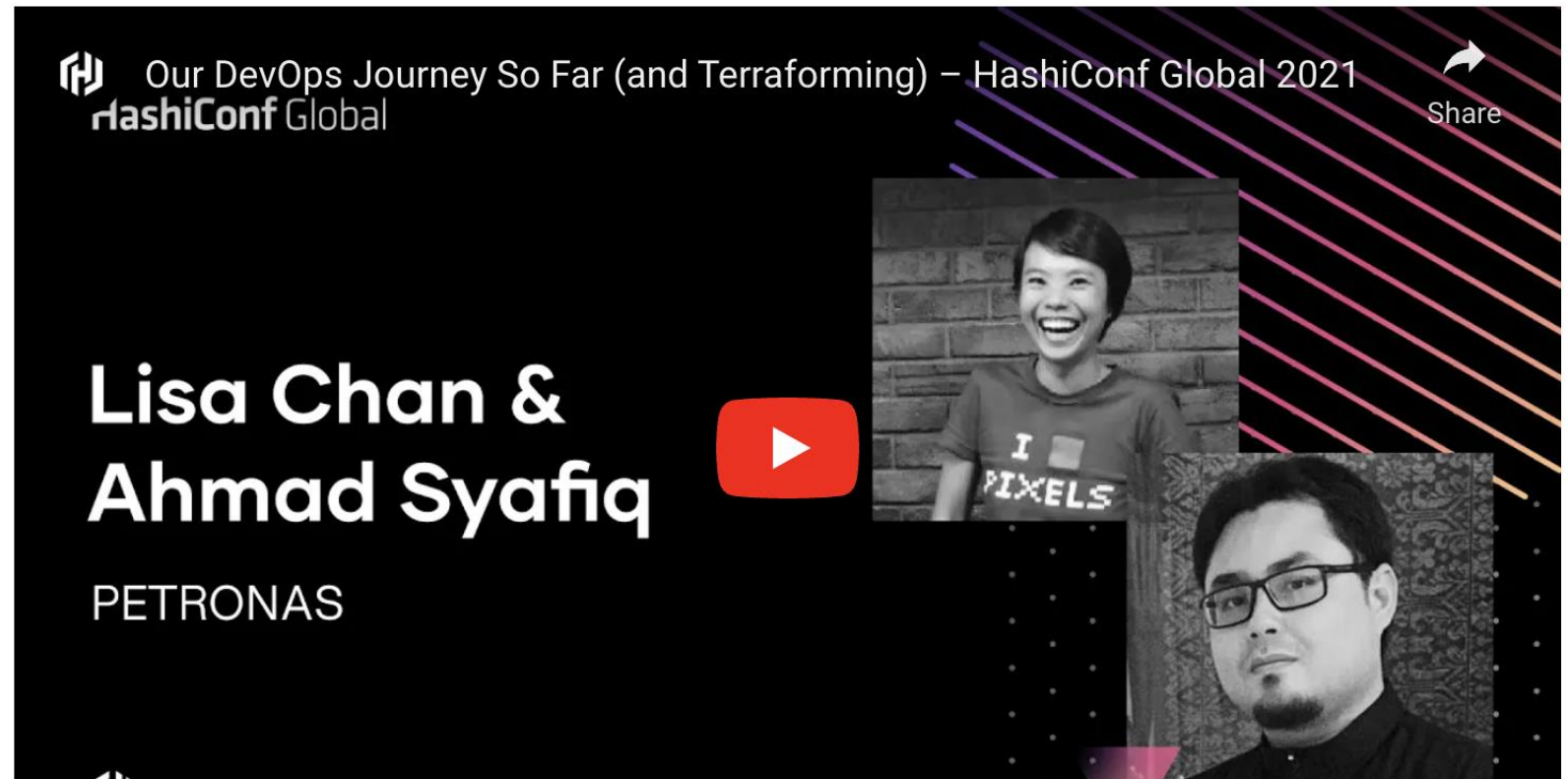
[Multi-Cloud DevOps at PETRONAS with Terraform \(hashicorp.com\)](https://www.hashicorp.com/blog/multi-cloud-devops-at-petronas-with-terraform)

CASE STUDY

Multi-Cloud DevOps at PETRONAS with Terraform

Published 4:15 PM GMT+8 Dec 10, 2021

Learn about a Malaysian energy company's DevOps journey while operating infrastructure as code in both AWS and Azure using HashiCorp Terraform.





Cloud Adoption Framework for Terraform Landing zones

Everything-as-code. Opinionated.

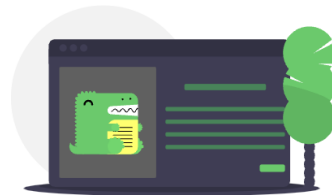
<https://aka.ms/caf/terraform>

Let's start deploying!



Configuration-driven IaC

Spend your time deploying what you need on Azure, not writing IaC modules.



Empowering the SRE on Azure

We equip the Site Reliability Engineering on Azure providing community-driven and built artifacts.



DevOps by nature

Whichever DevOps tools you are using, we have you covered.