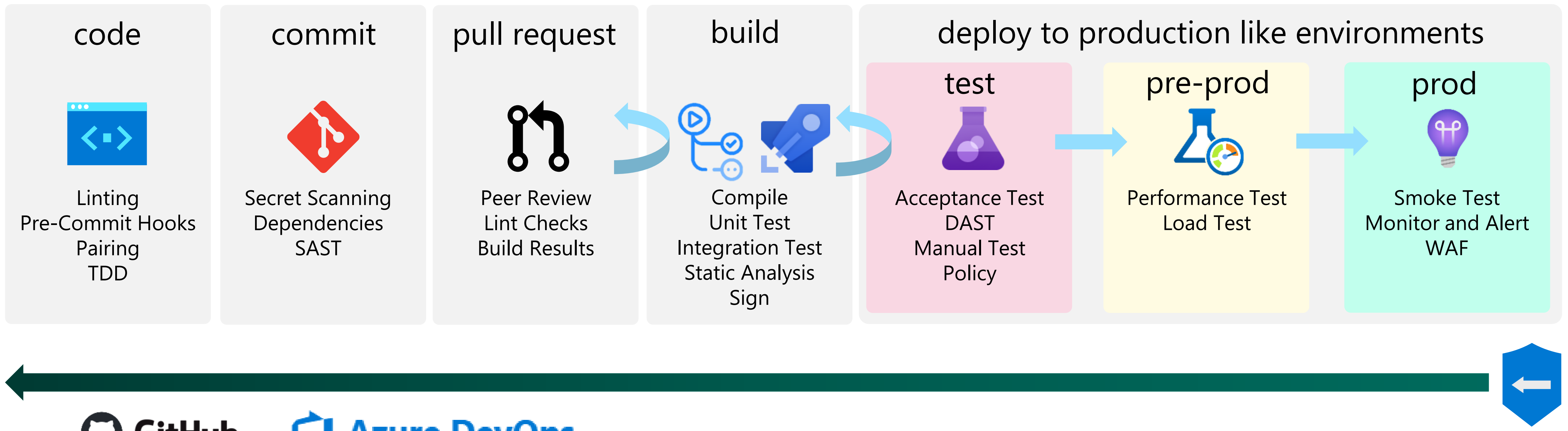
Microsoft

Continuous Delivery

Shift Left

Policy and Static Analysis as part of Continuous Delivery

Shift Left with DevSecOps



GitHub Azure DevOps

What can Shift Left?

Module testing

- Modules can be integration tested
- No unexpected surprises when moving to a new version

Static analysis

- Scan for vulnerabilities (SAST)
- Format checks
- Custom checks on code
  - E.g. Check for version constraints
- Custom checks on plan

Policy as code

- OPA or Sentinel
- Check for standards before apply
- Fail fast and avoid partial deployments
- Examples:
  - Only allow certain regions
  - Only allow certain SKUs
  - Only allow pattern modules and disallow direct resource references

Cost Estimation

- Ability see increase or decrease in cost prior to deploying or updating
- This can feed into policies to stop / approve large uplifts

Shift Left with DevSecOps and Terraform

HashiCorp Terraform

CLI

```
> terraform fmt
> terraform validate
> terraform plan
```

Microsoft Defender for DevOps

terrascan by tenable

Third Party Integrations / Run Tasks

Prisma Cloud

Snyk

Bridge Crew

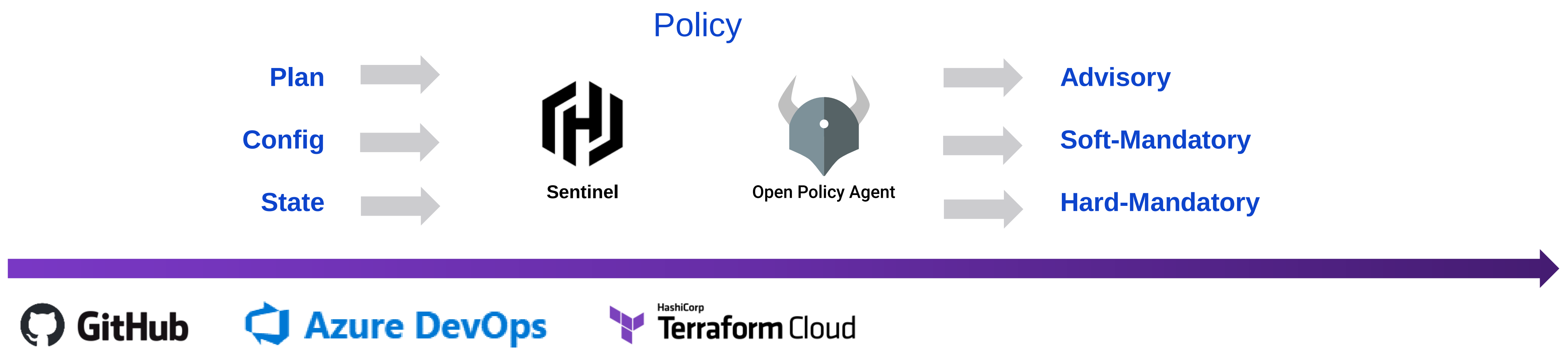
HCP Packer

GitHub

Azure DevOps

HashiCorp Terraform Cloud

Shift Left with DevSecOps and Terraform



Automating Terraform

Terraform CLI Automation Overview

- Your agent needs the Terraform CLI.
- Must use Remote State!
- Use Environment Variables for credentials.
- Use Parameters for Remote State settings.
- Use the same commands as you do locally to **`init`**, **`plan`** and **`apply`**.

- Questions to consider:
  - How will I access my modules?
  - Do I want to have an approval between plan and apply?
  - Do I want to run any static analysis?

Getting the Terraform CLI

Options:

- Custom runner with Terraform baked in to the VM or Container.
- Use a pre-defined step to download the version you need.
  - HashiCorp - Setup Terraform · Actions · GitHub Marketplace

```
steps:
- uses: hashicorp/setup-terraform@v2
  with:
    terraform_version: 1.9
```

- Use **`curl`** or similar to download the version you require.

Static Analysis

Terraform Format and Validate

```
steps:
- uses: hashicorp/setup-terraform@v2
  with:
    terraform_version: 1.9

- name: Clone repo
  uses: actions/checkout@master

- name: Terraform fmt
  id: fmt
  run: terraform fmt -check

- name: Terraform Init
  id: init
  run: terraform init

- name: Terraform Validate
  id: validate
  run: terraform validate -no-color
```

Security Static Analysis

- Third party plugins:
  - Snyk
  - Bridgecrew
  - Tfsec
  - Tfscan
  - Etc…
- All can be integrated into your pipeline.

```
steps:
- name: Clone repo
  uses: actions/checkout@master
- name: tfsec
  uses: aquasecurity/tfsec-pr-commenter-action@v1.2.0
  with:
    tfsec_args: --soft-fail
    github_token: ${{ github.token }}
```

Approvals

Approval between plan and apply

- Warning: There can be a long wait for a human to review!
- Options:
  - Run `plan` then run another plan during `apply` stage.
  - Output the `plan` and consume that in the `apply` stage.

```
steps:
…
- name: Terraform plan
  run: terraform plan -out plan.tfplan
# Upload plan file to an artefact

# Job has an Environment with an approval
steps:
…
# Download the plan file artefact
- name: Terraform apply
  run: terraform apply -auto-approve plan.tfplan
```

Authentication

Secret Fundamentals

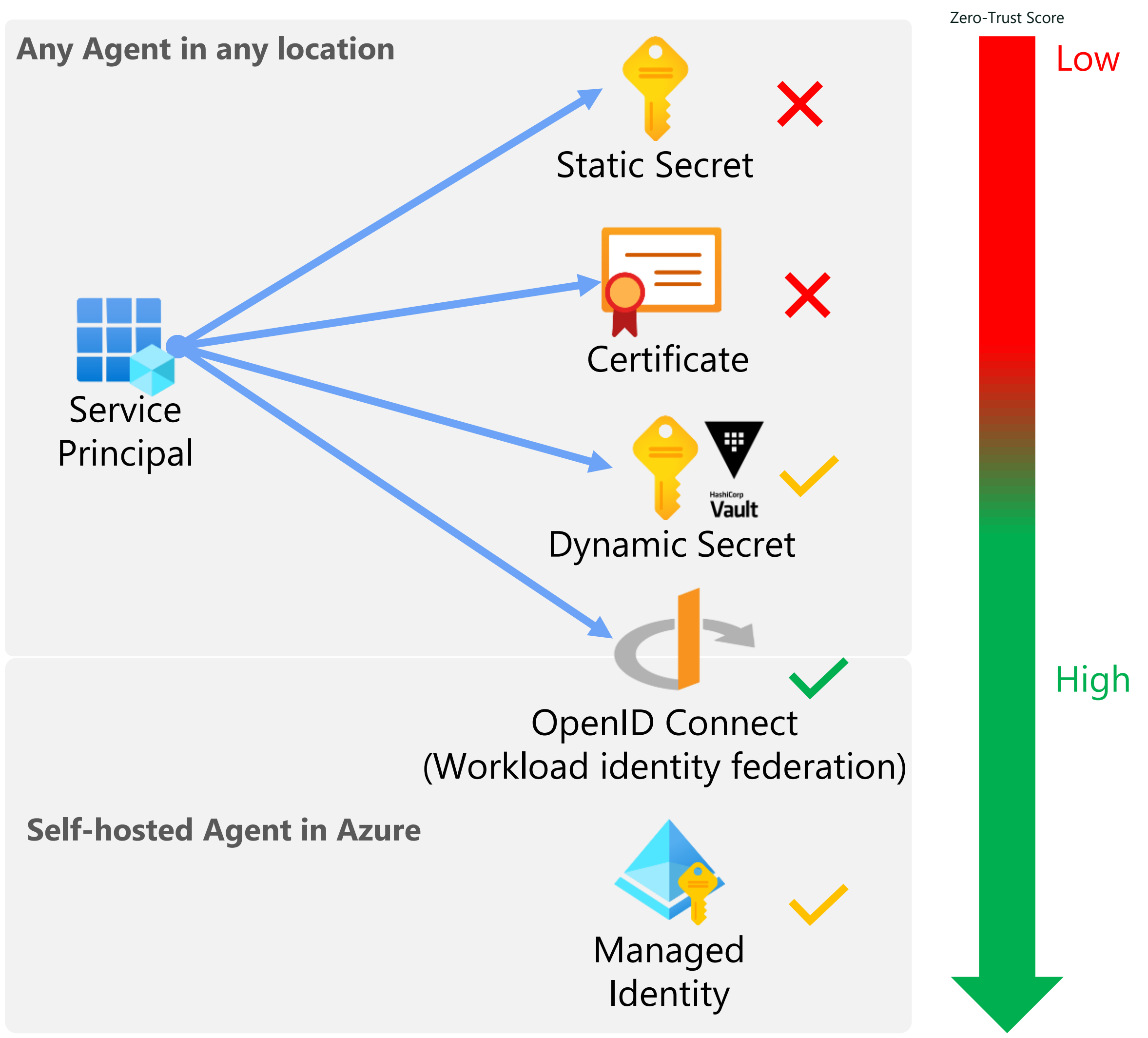
Deploy-time Secret

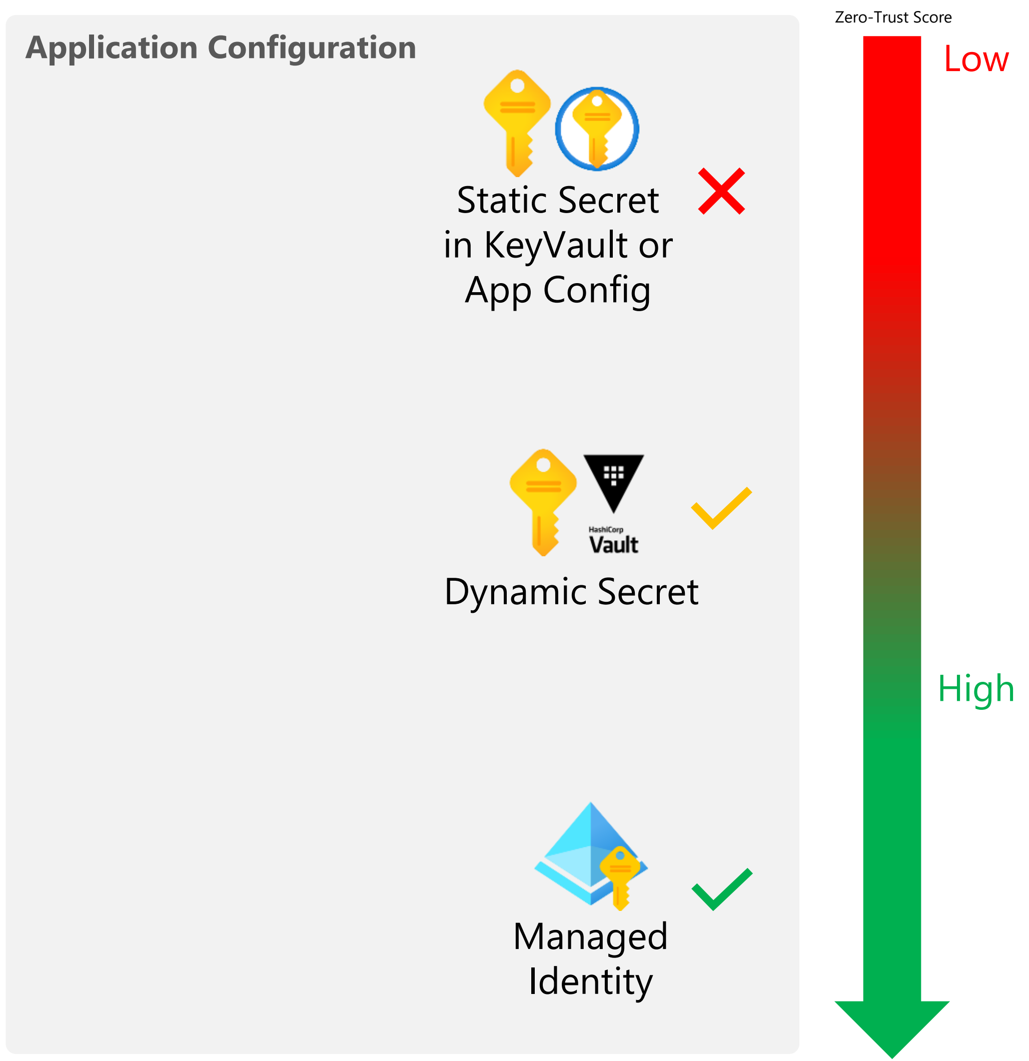
- Secrets used to connect to cloud provider or other providers used by Terraform
- E.g. Azure Service Principal for azuread or azurerm providers

Run-time Secret

- Secrets used by applications deployed by Terraform.
- E.g. SQL connection string for an ASP.NET app deployed to an App Service to connect back to a database

Deploy-time and Run-time Secrets





Deploy-time Secrets for Azure (AD and ARM)

Service Principal Secret

- Most basic and least secure.
- Requires you store, manage and rotate the secret.

Service Principal Certificate

- Usually applied to an agent.
- Similar level of security to a Secret.
- Requires you store, manage and rotate.

Managed Service Identity

- Can only be used on a self-hosted Agent or Terraform Enterprise server.
- No credentials need to be stored.
- User Managed Identity can be used to scale independent of agents, but this does not offer robust security silo.
- Scaling and keeping robust delineation can be hard.

OpenID Connect Federation (Workload identity federation)

- Supports self-hosted and provider-hosted agents.
- Supported by GitHub, Azure DevOps (preview) and Terraform Cloud / Enterprise.
- Delineation within a single GitHub Actions pipeline can be difficult to control. Based on environments.

<ALL OF THESE OPTIONS CAN AND SHOULD BE AUTOMATED!/>

Run-time Secrets for Azure

Secret / Password

- Use as a last resort if the API / service you are calling does not support Managed Identity.
- Generate and store as part of your Terraform.
- Do not involve a human!

ℹ <ALL OF THESE OPTIONS CAN AND SHOULD BE AUTOMATED!/>

Password Vault / Ephemeral Secrets

- Next best option if Managed Identity is not supported.
- Use a service that can automatically rotate your secrets.
- Terraform does not need to know the secret.
- Use Terraform to configure the service.
- Do not involve a human!

Managed Service Identity

- Most resource types support Managed Identity.
- Remember to apply granular permissions using principle of least privilege.

azurerm Service Principal with Secret

DO NOT USE!

- Steps:
  - Create App Registration (Service Principal) in Azure.
  - Generate a Secret for the Service Principal.
  - Assign Permissions on the Subscription or Resource Group for the Service Principal.

```
jobs:
  deploy_to_dev:
    …
    env:
     ARM_CLIENT_ID: ${{ secrets.ARM_CLIENT_ID }}
     ARM_CLIENT_SECRET: ${{ secrets.ARM_CLIENT_SECRET }}
     ARM_SUBSCRIPTION_ID: ${{ secrets.ARM_SUBSCRIPTION_ID }}
     ARM_TENANT_ID: ${{ secrets.ARM_TENANT_ID }}

    steps:
    …
     - name: Terraform Apply
       run: terraform apply -auto-approve
```

azurerm Service Principal with Secret

DO NOT USE!

- Required Environment Variables:
  - ARM_CLIENT_ID: Service Principal Application ID
  - ARM_CLIENT_SECRET: Service Principal Secret
  - ARM_SUBSCRIPTION_ID: The Azure Subscription ID
    - NOTE: This is a restriction of the azurerm provider, you can use `alias`, but it is not dynamic.
  - ARM_TENANT_ID: The Azure AD Tenant ID

```
jobs:
  deploy_to_dev:
    …
    env:
     ARM_CLIENT_ID: ${{ secrets.ARM_CLIENT_ID }}
     ARM_CLIENT_SECRET: ${{ secrets.ARM_CLIENT_SECRET }}
     ARM_SUBSCRIPTION_ID: ${{ secrets.ARM_SUBSCRIPTION_ID }}
     ARM_TENANT_ID: ${{ secrets.ARM_TENANT_ID }}

    steps:
    …
     - name: Terraform Apply
       run: terraform apply -auto-approve
```

azurerm Managed Identity

BE CONCIOUS OF THE BLAST RADIUS WITH SHARED AGENTS

- Steps:
  - Deploy GitHub Runner to Virtual Machine, Container Instance, etc.
  - Create a Machine Assigned or User Assigned Managed Identity for the Compute.
  - Assign Permissions on the Subscription or Resource Group for the Managed Identity.

```
jobs:
  deploy_to_dev:
    …
    env:
     ARM_USE_MSI: true
     ARM_MSI_ENDPOINT: ${{ env.MSI_ENDPOINT }}
     ARM_CLIENT_ID: ${{ secrets.ARM_CLIENT_ID }} #Only for User Assigned
     ARM_SUBSCRIPTION_ID: ${{ secrets.ARM_SUBSCRIPTION_ID }}
     ARM_TENANT_ID: ${{ secrets.ARM_TENANT_ID }}

    steps:
    …
     - name: Terraform Apply
       run: terraform apply -auto-approve
```

azurerm Managed Identity

BE CONCIOUS OF THE BLAST RADIUS WITH SHARED AGENTS

- Required Environment Variables:
  - ARM_USE_MSI: Must be set to true
  - ARM_MSI_ENDPOINT: Some Azure services have a different endpoint
    - Can set it to the MSI_ENDPOINT environment variable.
  - ARM_CLIENT_ID: Only required for User Assigned Managed Identity
  - ARM_SUBSCRIPTION_ID: The Azure Subscription ID
  - ARM_TENANT_ID: The Azure AD Tenant ID

```
jobs:
  deploy_to_dev:
    …
    env:
     ARM_USE_MSI: true
     ARM_MSI_ENDPOINT: ${{ env.MSI_ENDPOINT }}
     ARM_CLIENT_ID: ${{ secrets.ARM_CLIENT_ID }} #Only for User Assigned
     ARM_SUBSCRIPTION_ID: ${{ secrets.ARM_SUBSCRIPTION_ID }}
     ARM_TENANT_ID: ${{ secrets.ARM_TENANT_ID }}

    steps:
    …
     - name: Terraform Apply
       run: terraform apply -auto-approve
```

azurerm Service Principal and OpenID Connect
(Workload identity federation)

- Steps:
  - Create a User Assigned Managed Identity or App Registration (Service Principal) in Azure.
  - Add a Federated Credential for GitHub
  - Scope to the GitHub Repository
    - Optionally scope to Environment, Branch, Tag or Pull Request
    - E.g. subject = repo:my_github_org/my_github_repo:environment:dev
  - Assign Permissions on the Subscription or Resource Group for the Service Principal.

```
jobs:
  deploy_to_dev:
    …
    environment: dev
    env:
     ARM_USE_OIDC: true
     ARM_CLIENT_ID: ${{ secrets.ARM_CLIENT_ID }}
     ARM_SUBSCRIPTION_ID: ${{ secrets.ARM_SUBSCRIPTION_ID }}
     ARM_TENANT_ID: ${{ secrets.ARM_TENANT_ID }}

    steps:
    …
     - name: Terraform Apply
       run: terraform apply -auto-approve
```

azurerm Service Principal and OpenID Connect (Workload identity federation)

- Required Environment Variables:
  - ARM_USE_OIDC: Must be set to true
  - ARM_CLIENT_ID: Required to tell it which Service Principal to use
  - ARM_SUBSCRIPTION_ID: The Azure Subscription ID
  - ARM_TENANT_ID: The Azure AD Tenant ID

```
jobs:
  deploy_to_dev:
    …
    environment: dev
    env:
     ARM_USE_OIDC: true
     ARM_CLIENT_ID: ${{ secrets.ARM_CLIENT_ID }}
     ARM_SUBSCRIPTION_ID: ${{ secrets.ARM_SUBSCRIPTION_ID }}
     ARM_TENANT_ID: ${{ secrets.ARM_TENANT_ID }}

    steps:
    …
     - name: Terraform Apply
       run: terraform apply -auto-approve
```

Lab 8: Continuous Delivery

Microsoft

Thank you