

# Continuous Delivery

# Conditions and Terms of Use

## Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet website references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

## Copyright and Trademarks

© 2016 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

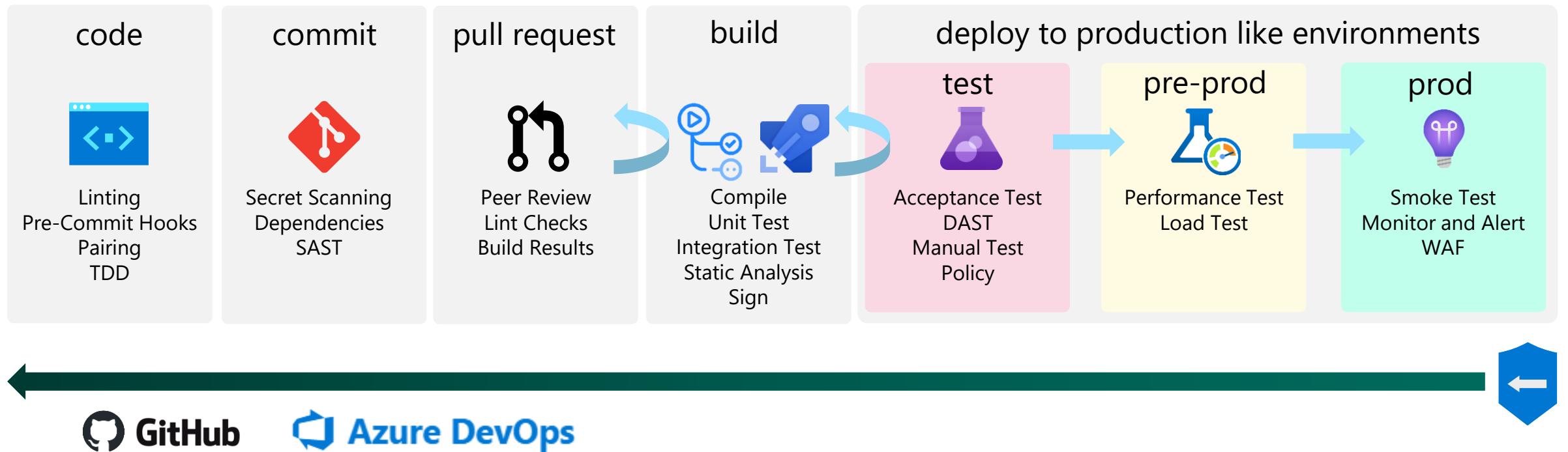
For more information, see **Use of Microsoft Copyrighted Content** at  
<https://www.microsoft.com/en-us/legal/intellectualproperty/permissions/default.aspx>

Microsoft®, Internet Explorer®, Outlook®, SkyDrive®, Windows Vista®, Zune®, Xbox 360®, DirectX®, Windows Server® and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

# Shift Left

Policy and Static Analysis as part of Continuous Delivery

# Shift Left with DevSecOps



# 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



CLI

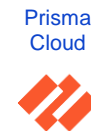
```
> terraform fmt  
> terraform validate  
> terraform plan
```



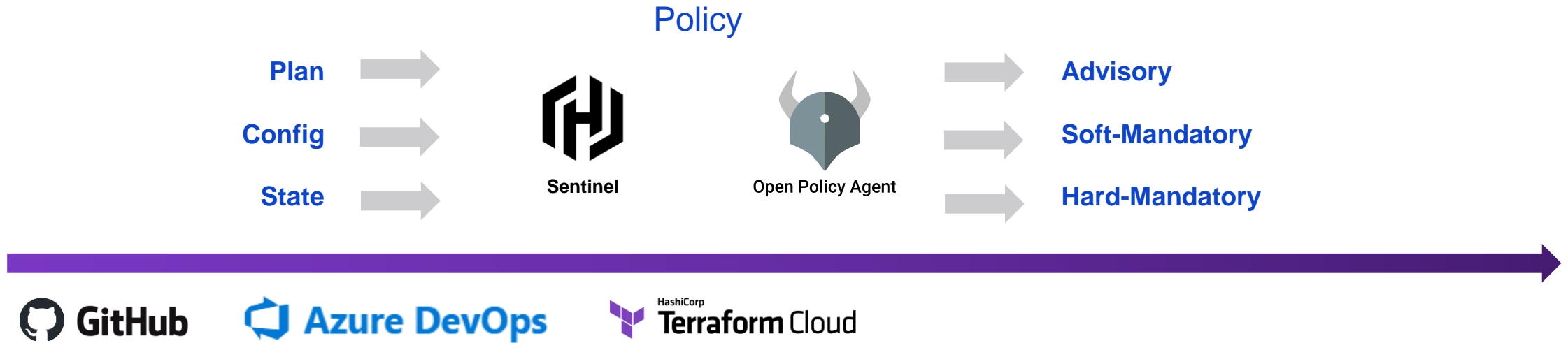
Microsoft Defender for DevOps



Third Party Integrations / Run Tasks



# 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:
  - Snyc
  - 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

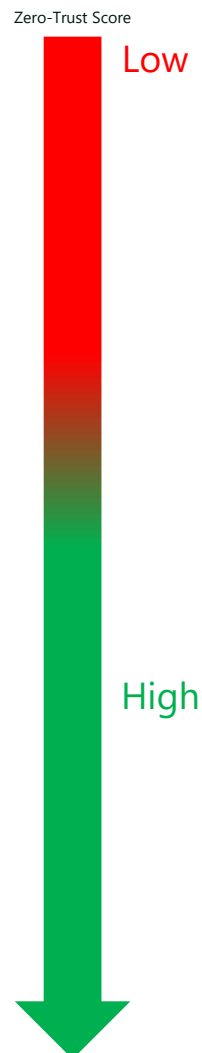
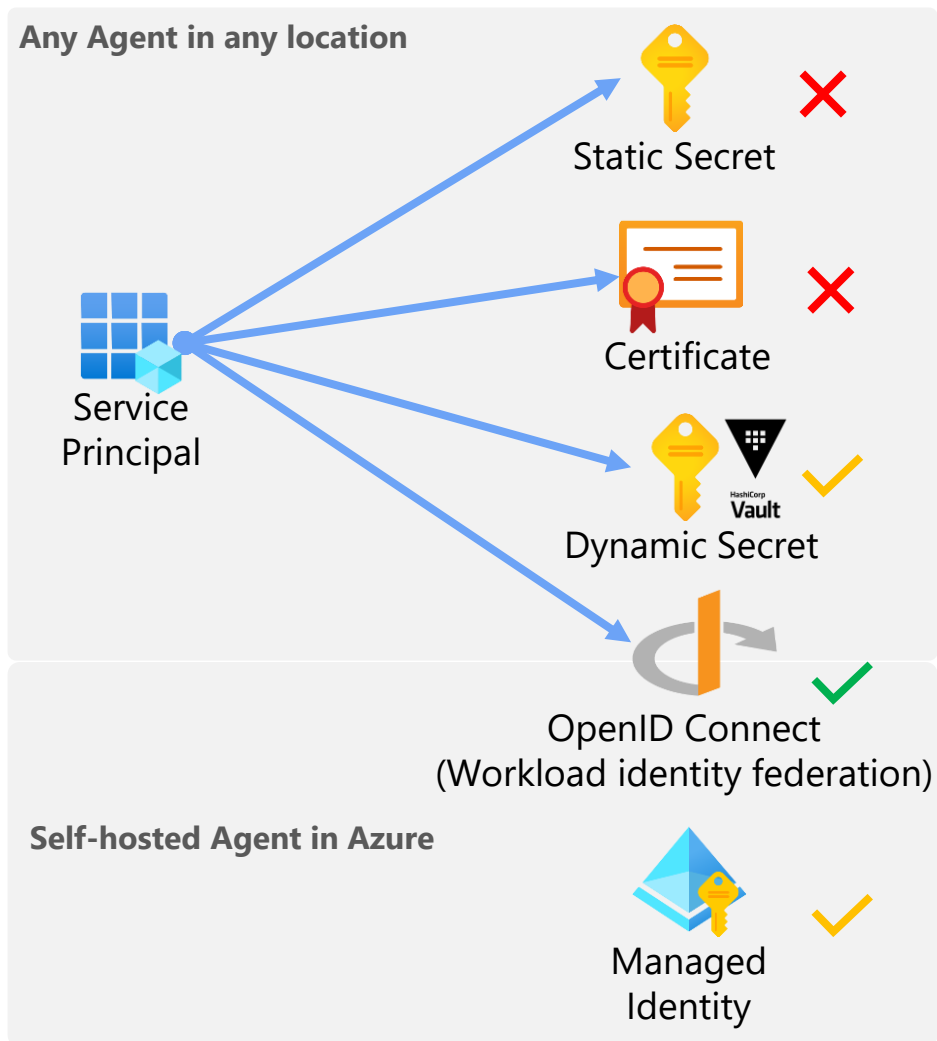
- Secrets used to connect to cloud provider or other providers used by Terraform
- E.g. Azure Service Principal for azuread or azurerm providers

## Run-time Secret

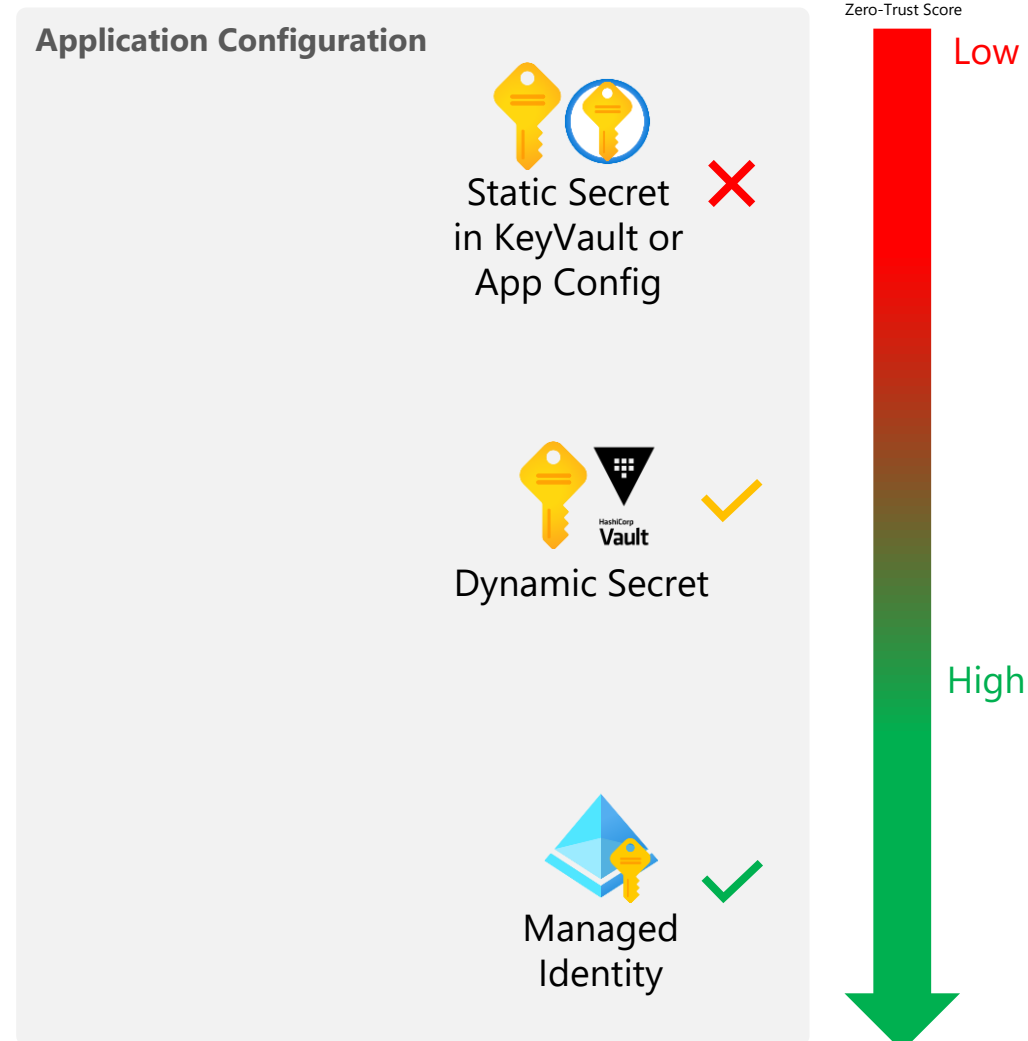
- Secrets used by applications deployed by Terraform.
- E.g. SQL connection string for an ASP.NET app deployed to an App Service to connect back to a database

# Deploy-time and Run-time Secrets

## Deploy-time Secrets



## 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!



<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





Thank you