Here is a **GitHub Actions workflow YAML** for **Azure Policy as Code deployment**. This workflow:

* **Validates JSON syntax** for policies.
* **Runs policy compliance tests**.
* **Deploys policies** to Sandbox first, then Production upon approval.

GitHub Workflow YAML (.github/workflows/deploy-policy.yml)

name: Deploy Azure Policies

on:

push:

branches:

- main

- sandbox

pull\_request:

branches:

- main

- sandbox

jobs:

validate-policy:

name: Validate JSON and Policy Definitions

runs-on: ubuntu-latest

steps:

- name: Checkout Repository

uses: actions/checkout@v4

- name: Validate JSON Syntax

run: |

find ./policies -name '\*.json' -exec jq empty {} \;

- name: Login to Azure

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

- name: Validate Azure Policies

run: |

az policy definition list --query "[].{name:name}" -o table

deploy-policy:

name: Deploy Policies to Azure

runs-on: ubuntu-latest

needs: validate-policy

steps:

- name: Checkout Repository

uses: actions/checkout@v4

- name: Login to Azure

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

- name: Set Target Environment

id: env

run: |

if [[ "${{ github.ref }}" == "refs/heads/main" ]]; then

echo "ENVIRONMENT=Production" >> $GITHUB\_ENV

elif [[ "${{ github.ref }}" == "refs/heads/sandbox" ]]; then

echo "ENVIRONMENT=Sandbox" >> $GITHUB\_ENV

fi

- name: Deploy Policy Definitions

run: |

for policy in $(ls policies/\*.json); do

policyName=$(basename $policy .json)

az policy definition create --name $policyName --rules $policy --mode All || az policy definition update --name $policyName --rules $policy

done

- name: Assign Policies to Subscription

run: |

for assignment in $(ls assignments/\*.json); do

assignmentName=$(basename $assignment .json)

az policy assignment create --name $assignmentName --policy $assignment || az policy assignment update --name $assignmentName --policy $assignment

done

- name: Display Deployment Summary

run: |

echo "Policies deployed successfully to ${{ env.ENVIRONMENT }} environment."

**How It Works**

✅ **Validates JSON** using jq.  
✅ **Checks Azure Policy Definitions** for syntax errors.  
✅ **Deploys policies to Sandbox or Production**, depending on the branch.  
✅ **Ensures assignments are applied to Azure subscriptions**.  
✅ **Uses GitHub Secrets (AZURE\_CREDENTIALS)** for authentication.

**Next Steps**

1. **Store Azure Credentials in GitHub Secrets**:
   * Generate a **service principal** in Azure with Contributor and Policy Administrator roles.
   * Save the output JSON in **GitHub Secrets** as AZURE\_CREDENTIALS.
2. **Define Your Policy Files**:
   * Store policy definitions under policies/\*.json.
   * Store assignments under assignments/\*.json.
3. **Test the Workflow**:
   * Push a policy update to the **sandbox** branch.
   * If successful, merge into **main** to deploy to production.

**Azure Policy as Code - Requirements & GitHub Actions Workflow**

**1. Overview**

This document outlines the requirements and implementation details for managing **Azure Policy as Code** using **GitHub** and **GitHub Actions**. It includes the necessary branching strategy, role-based access control (RBAC), and automated deployment workflow.

**2. Requirements**

**2.1 Version Control & Policy Management**

* Use **GitHub** as the central repository for managing Azure Policy as Code.
* Policies and initiatives should be defined using **JSON** with version control enabled.

**2.2 File Structure for Policies & Initiatives**

* Each **policy** and **initiative** should have dedicated files, including:
  + **Definition**: Parameters, rules, and compliance states.
  + **Assignment & Exemptions**: Configuration for applying policies to resources.

**2.3 Environments & Deployment Strategy**

* The following **Azure environments** should be supported:
  + **Sandbox** – for testing policy changes.
  + **Production** – for enforcing policies across actual workloads.

**2.4 Branching Strategy**

* Implement **Git branching model** for controlled policy deployment:
  + **Main Branch** → Production Environment.
  + **Sandbox Branch** → Sandbox Environment.
  + **Feature Branches** → Used for development and testing before merging into sandbox/main.

**2.5 Access Control & Security**

* **Role-based Access Control (RBAC)** must be enforced:
  + Policies with **Deny, Modify, DeployIfNotExists, Mutate, or Append** effects → **Only CloudOps team** can deploy.
  + Policies with **Audit & AuditIfNotExists** → **CloudOps and GCI teams** can deploy.

**2.6 GitHub Workflow Automation**

* **GitHub Actions & Workflows** should be configured to:
  + Automatically **validate JSON syntax** and test policies.
  + Deploy policies to the **Sandbox** first.
  + Deploy policies to **Production** only after passing validation.

**2.7 Issue Management & Change Process**

* **GitHub Issues & Templates** should be used for:
  + **New policy requests**.
  + **Updates to existing policies**.
  + **Tracking exemption requests**.

**2.8 Automated Policy Testing & Compliance**

* Define a **test suite** to ensure:
  + Policies deploy without conflicts.
  + Assignments are applied correctly.
  + Policies remain compliant with Azure Security & Governance standards.

**2.9 CI/CD Pipeline for Policy Deployment**

* Implement a **GitHub Actions CI/CD pipeline** with:
  + **Pre-deployment checks** (syntax validation, policy simulation).
  + **Automated Testing** before merging to main.
  + **Scheduled compliance scans** for drift detection.

**3. GitHub Actions Workflow**

**GitHub Workflow YAML (.github/workflows/deploy-policy.yml)**

name: Deploy Azure Policies

on:

push:

branches:

- main

- sandbox

pull\_request:

branches:

- main

- sandbox

jobs:

validate-policy:

name: Validate JSON and Policy Definitions

runs-on: ubuntu-latest

steps:

- name: Checkout Repository

uses: actions/checkout@v4

- name: Validate JSON Syntax

run: |

find ./policies -name '\*.json' -exec jq empty {} \;

- name: Login to Azure

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

- name: Validate Azure Policies

run: |

az policy definition list --query "[].{name:name}" -o table

deploy-policy:

name: Deploy Policies to Azure

runs-on: ubuntu-latest

needs: validate-policy

steps:

- name: Checkout Repository

uses: actions/checkout@v4

- name: Login to Azure

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

- name: Set Target Environment

id: env

run: |

if [[ "${{ github.ref }}" == "refs/heads/main" ]]; then

echo "ENVIRONMENT=Production" >> $GITHUB\_ENV

elif [[ "${{ github.ref }}" == "refs/heads/sandbox" ]]; then

echo "ENVIRONMENT=Sandbox" >> $GITHUB\_ENV

fi

- name: Deploy Policy Definitions

run: |

for policy in $(ls policies/\*.json); do

policyName=$(basename $policy .json)

az policy definition create --name $policyName --rules $policy --mode All || az policy definition update --name $policyName --rules $policy

done

- name: Assign Policies to Subscription

run: |

for assignment in $(ls assignments/\*.json); do

assignmentName=$(basename $assignment .json)

az policy assignment create --name $assignmentName --policy $assignment || az policy assignment update --name $assignmentName --policy $assignment

done

- name: Display Deployment Summary

run: |

echo "Policies deployed successfully to ${{ env.ENVIRONMENT }} environment."

**4. Next Steps**

1. **Store Azure Credentials in GitHub Secrets**:
   * Generate a **service principal** in Azure with Contributor and Policy Administrator roles.
   * Save the output JSON in **GitHub Secrets** as AZURE\_CREDENTIALS.
2. **Define Your Policy Files**:
   * Store policy definitions under policies/\*.json.
   * Store assignments under assignments/\*.json.
3. **Test the Workflow**:
   * Push a policy update to the **sandbox** branch.
   * If successful, merge into **main** to deploy to production.

**5. Additional Enhancements**

1. **Use Azure Policy Remediation Tasks**
   * Ensure that **DeployIfNotExists** and **Modify** effects trigger automated remediation for non-compliant resources.
2. **Logging & Monitoring**
   * Use **Azure Monitor, Log Analytics, and Policy Insights** for tracking compliance.
3. **Define Exemption Process**
   * Clearly document the process to request and approve **policy exemptions**.

**End of Document**