# 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.

## 3. GitHub Actions Workflow

Below is the GitHub workflow YAML file for Azure Policy deployment.

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."