Azure ARM Template - Automation

Sample ARM Template

name: Deploy ARM Template to Azure

on:

Trigger on push to main branch

push:
branches: [main]
paths:
- 'infrastructure/**'
- '.github/workflows/deploy-infrastructure.yml'

Trigger on pull request to main branch

pull_request:
branches: [main]
paths:
- 'infrastructure/**'

Allow manual trigger

workflow_dispatch:
inputs:
environment:
description: 'Environment to deploy to'
required: true

```
default: 'dev'
type: choice
options:
- dev
- test
- prod

env:
AZURE_RESOURCE_GROUP: 'rg-student-demo'
AZURE_LOCATION: 'East US'

jobs:
validate:
name: Validate ARM Template
runs-on: ubuntu-latest

  steps:
  - name: Checkout code
    uses: actions/checkout@v4

  - name: Azure Login
    uses: azure/login@v1
    with:
      creds: ${{ secrets.AZURE_CREDENTIALS }}

  - name: Validate ARM Template
    uses: azure/arm-deploy@v1
    with:
      subscriptionId: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
      resourceGroupName: ${{ env.AZURE_RESOURCE_GROUP }}
      template: ./infrastructure/azuredeploy.json
      parameters: ./infrastructure/azuredeploy.parameters.dev.json
      deploymentMode: Validate

deploy-dev:
name: Deploy to Development
runs-on: ubuntu-latest
```

needs: validate
if: github.ref == 'refs/heads/main' || github.event_name == 'workflow_dispatch'
environment: development

```
steps:
- name: Checkout code
  uses: actions/checkout@v4

- name: Azure Login
  uses: azure/login@v1
  with:
    creds: ${{ secrets.AZURE_CREDENTIALS }}

- name: Create Resource Group
  run: |
    az group create \\
      --name ${{ env.AZURE_RESOURCE_GROUP }}-dev \\
      --location "${{ env.AZURE_LOCATION }}"

- name: Deploy ARM Template to Dev
  uses: azure/arm-deploy@v1
  id: deploy
  with:
    subscriptionId: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
    resourceGroupName: ${{ env.AZURE_RESOURCE_GROUP }}-dev
    template: ./infrastructure/azuredeploy.json
    parameters: ./infrastructure/azuredeploy.parameters.dev.json
    deploymentName: 'github-actions-${{ github.run_number }}'

- name: Output deployment results
  run: |
    echo "Storage Account Name: ${{ steps.deploy.outputs.storageAccountNa
me }}"
    echo "Storage Account ID: ${{ steps.deploy.outputs.storageAccountId }}"
```

deploy-test:
name: Deploy to Test
runs-on: ubuntu-latest
needs: deploy-dev
if: github.ref == 'refs/heads/main'
environment: test

```
steps:
- name: Checkout code
  uses: actions/checkout@v4

- name: Azure Login
  uses: azure/login@v1
  with:
    creds: ${{ secrets.AZURE_CREDENTIALS }}

- name: Create Resource Group
  run: |
    az group create \\
      --name ${{ env.AZURE_RESOURCE_GROUP }}-test \\
      --location "${{ env.AZURE_LOCATION }}"

- name: Deploy ARM Template to Test
  uses: azure/arm-deploy@v1
  with:
    subscriptionId: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
    resourceGroupName: ${{ env.AZURE_RESOURCE_GROUP }}-test
    template: ./infrastructure/azuredeploy.json
    parameters: ./infrastructure/azuredeploy.parameters.test.json
    deploymentName: 'github-actions-${{ github.run_number }}'
```

deploy-prod:
name: Deploy to Production
runs-on: ubuntu-latest
needs: deploy-test
if: github.ref == 'refs/heads/main' && (github.event_name == 'workflow_dispatch'

&& github.event.inputs.environment == 'prod')
environment: production

```
steps:
- name: Checkout code
  uses: actions/checkout@v4

- name: Azure Login
  uses: azure/login@v1
  with:
    creds: ${{ secrets.AZURE_CREDENTIALS }}

- name: Create Resource Group
  run: |
    az group create \\
      --name ${{ env.AZURE_RESOURCE_GROUP }}-prod \\
      --location "${{ env.AZURE_LOCATION }}"

- name: Deploy ARM Template to Production
  uses: azure/arm-deploy@v1
  with:
    subscriptionId: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
    resourceGroupName: ${{ env.AZURE_RESOURCE_GROUP }}-prod
    template: ./infrastructure/azuredeploy.json
    parameters: ./infrastructure/azuredeploy.parameters.prod.json
    deploymentName: 'github-actions-${{ github.run_number }}'

- name: Send deployment notification
  run: |
    echo "Production deployment completed successfully!"
    # Add notification logic here (Slack, Teams, etc.)
```

Parameters

```
{
"$schema": "
https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
"contentVersion": "1.0.0.0",
"parameters": {
"storageAccountName": {
"value": "mystudentprod001"
},
"storageAccountType": {
"value": "Standard_RAGRS"
},
"location": {
"value": "East US"
},
"vnetName": {
"value": "student-prod-vnet"
},
"vnetAddressPrefix": {
"value": "10.3.0.0/16"
},
"subnetName": {
"value": "prod-subnet"
},
"subnetAddressPrefix": {
"value": "10.3.1.0/24"
},
"environment": {
"value": "prod"
},
"enableHttpsTrafficOnly": {
"value": true
}
}
}
```

GitHub Actions Automation Setup Guide

This guide shows how to set up automated ARM template deployment using GitHub Actions.

Repository Structure

```
your-repo/
├── .github/
│   └── workflows/
│       └── deploy-infrastructure.yml
├── infrastructure/
│   ├── azuredeploy.json
│   ├── azuredeploy.parameters.dev.json
│   ├── azuredeploy.parameters.test.json
│   └── azuredeploy.parameters.prod.json
└── README.md
```

Step 1: Set up Azure Service Principal

Create a service principal that GitHub Actions will use to deploy to Azure:

```
# Login to Azure
az login

# Create service principal
az ad sp create-for-rbac \
  --name "github-actions-arm-deployment" \
  --role contributor \
  --scopes /subscriptions/{subscription-id} \
  --json-auth

# This will output JSON like:
{
  "clientId": "...",
```

```
  "clientSecret": "...",
  "subscriptionId": "...",
  "tenantId": "...",
  "activeDirectoryEndpointUrl": "...",
  "resourceManagerEndpointUrl": "...",
  "activeDirectoryGraphResourceId": "...",
  "sqlManagementEndpointUrl": "...",
  "galleryEndpointUrl": "...",
  "managementEndpointUrl": "..."
}
```

Step 2: Configure GitHub Secrets

In your GitHub repository, go to **Settings > Secrets and variables > Actions** and add:

Repository Secrets:

- `AZURE_CREDENTIALS`: The entire JSON output from step 1
- `AZURE_SUBSCRIPTION_ID`: Your Azure subscription ID

Environment Secrets:

Create environments for each stage:

1. Go to **Settings > Environments**
2. Create environments: `development`, `test`, `production`
3. For `production` environment, add protection rules:
   - ✅ Required reviewers
   - ✅ Wait timer (optional)

Step 3: Workflow Triggers

The GitHub Actions workflow triggers on:

1. **Push to main branch** - Automatically deploys to dev and test

2. **Pull request to main** - Validates the template only
3. **Manual dispatch** - Allows deployment to any environment including production

Step 4: Deployment Process

Automatic Flow:

```
Code Push → Validate → Deploy to Dev → Deploy to Test
```

Manual Production Deployment:

1. Go to **Actions** tab in GitHub
2. Select "Deploy ARM Template to Azure" workflow
3. Click "Run workflow"
4. Choose "prod" environment
5. Approve deployment (if protection rules are enabled)

Step 5: Environment-Specific Configuration

Each environment has different:

- **Storage account names** (must be globally unique)
- **Storage replication types** (LRS for dev, GRS for test, RAGRS for prod)
- **Network address spaces** (to avoid conflicts)
- **Resource group naming** (includes environment suffix)

Step 6: Monitoring and Notifications

View Deployment Status:

- **Actions tab**: See workflow runs and logs
- **Azure Portal**: View resource deployments and status

- **Azure CLI**: Check deployment history

Add Notifications:

```
- name: Notify Slack on Success
  if: success()
  uses: 8398a7/action-slack@v3
  with:
    status: success
    webhook_url: ${{ secrets.SLACK_WEBHOOK }}

- name: Notify on Failure
  if: failure()
  uses: 8398a7/action-slack@v3
  with:
    status: failure
    webhook_url: ${{ secrets.SLACK_WEBHOOK }}
```

Security Best Practices

1. **Least Privilege**: Service principal has only necessary permissions
2. **Environment Protection**: Production requires manual approval
3. **Secret Management**: Sensitive data stored in GitHub secrets
4. **Branch Protection**: Main branch requires PR reviews
5. **Audit Trail**: All deployments logged and traceable

Advanced Features

Template Validation:

```
- name: ARM Template Toolkit (TTK) Test
  uses: aliencube/arm-ttk-actions@v0.3.0
```

```
  with:
    path: ./infrastructure
```

What-If Deployment:

```
- name: What-If Analysis
  run: |
    az deployment group what-if \
      --resource-group ${{ env.AZURE_RESOURCE_GROUP }}-dev \
      --template-file ./infrastructure/azuredeploy.json \
      --parameters ./infrastructure/azuredeploy.parameters.dev.json
```

Rollback Strategy:

```
- name: Rollback on Failure
  if: failure()
  run: |
    az deployment group create \
      --resource-group ${{ env.AZURE_RESOURCE_GROUP }}-prod \
      --template-file ./infrastructure/previous-version.json \
      --parameters ./infrastructure/azuredeploy.parameters.prod.json
```

Teaching Points

1. **Infrastructure as Code**: ARM templates in version control
2. **CI/CD Pipeline**: Automated testing and deployment
3. **Environment Promotion**: Dev → Test → Prod progression
4. **Security**: Service principals and secrets management
5. **Collaboration**: PR reviews and approval gates
6. **Monitoring**: Deployment tracking and notifications
7. **Rollback**: Quick recovery from failed deployments

This automation approach demonstrates modern DevOps practices for cloud infrastructure management.