Quickstart: Deploy Bicep files by using GitHub Actions

Article • 01/22/2024

GitHub Actions is a suite of features in GitHub to automate your software development workflows. In this quickstart, you use the GitHub Actions for Azure Resource Manager deployment to automate deploying a Bicep file to Azure.

It provides a short introduction to GitHub actions and Bicep files. If you want more detailed steps on setting up the GitHub actions and project, see Deploy Azure resources by using Bicep and GitHub Actions.

Prerequisites

- An Azure account with an active subscription. Create an account for free.
- A GitHub account. If you don't have one, sign up for free.
- A GitHub repository to store your Bicep files and your workflow files. To create one, see Creating a new repository.

Create resource group

Create a resource group. Later in this quickstart, you'll deploy your Bicep file to this resource group.

CLI

Azure CLI

```
az group create -n exampleRG -l westus
```

Generate deployment credentials

Service principal

Your GitHub Actions run under an identity. Use the az ad sp create-for-rbac command to create a service principal for the identity. Grant the service principal the contributor role for the resource group created in the previous session so that the GitHub action with the identity can create resources in this resource group. It is recommended that you grant minimum required access.

Azure CLI

```
az ad sp create-for-rbac --name {app-name} --role contributor --scopes /subscriptions/{subscription-id}/resourceGroups/exampleRG --json-auth
```

Replace the placeholder `{app-name}` with the name of your application. Replace `{subscription-id}` with your subscription ID.

The output is a JSON object with the role assignment credentials that provide access to your App Service app similar to below.

Output

```
  {
    "clientId": "<GUID>",
    "clientSecret": "<GUID>",
    "subscriptionId": "<GUID>",
    "tenantId": "<GUID>",
    ...
  }
```

Copy this JSON object for later. You'll only need the sections with the `clientId`, `clientSecret`, `subscriptionId`, and `tenantId` values. Make sure you don't have an extra comma at the end of the last line, for example, the `tenantId` line in the preceding example, or else it will result in an invalid JSON file. You will get an error during the deployment saying "Login failed with Error: Content is not a valid JSON object. Double check if the 'auth-type' is correct."

Configure the GitHub secrets

Service principal

Create secrets for your Azure credentials, resource group, and subscriptions. You will use these secrets in the Create workflow section.

1. In GitHub, navigate to your repository.
2. Select **Settings** > **Secrets and variables** > **Actions** > **New repository secret**.
3. Paste the entire JSON output from the Azure CLI command into the secret's value field. Name the secret `AZURE_CREDENTIALS`.
4. Create another secret named `AZURE_RG`. Add the name of your resource group to the secret's value field (`exampleRG`).
5. Create another secret named `AZURE_SUBSCRIPTION`. Add your subscription ID to the secret's value field (example: `90fd3f9d-4c61-432d-99ba-1273f236afa2`).

Add a Bicep file

Add a Bicep file to your GitHub repository. The following Bicep file creates a storage account:

Bicep

```
@minLength(3)
@maxLength(11)
param storagePrefix string

@allowed([
  'Standard_LRS'
  'Standard_GRS'
  'Standard_RAGRS'
  'Standard_ZRS'
  'Premium_LRS'
  'Premium_ZRS'
  'Standard_GZRS'
  'Standard_RAGZRS'
])
param storageSKU string = 'Standard_LRS'

param location string = resourceGroup().location

var uniqueStorageName = '${storagePrefix}${uniqueString(resourceGroup().id)}'

resource stg 'Microsoft.Storage/storageAccounts@2023-04-01' = {
  name: uniqueStorageName
  location: location
  sku: {
    name: storageSKU
  }
  kind: 'StorageV2'
  properties: {
    supportsHttpsTrafficOnly: true
  }
}

output storageEndpoint object = stg.properties.primaryEndpoints
```

The Bicep file requires one parameter called **storagePrefix** with 3 to 11 characters.

You can put the file anywhere in the repository. The workflow sample in the next section assumes the Bicep file is named **main.bicep**, and it's stored at the root of your repository.

Create workflow

A workflow defines the steps to execute when triggered. It's a YAML (.yml) file in the **.github/workflows/** path of your repository. The workflow file extension can be either **.yml** or **.yaml**.

To create a workflow, take the following steps:

1. From your GitHub repository, select **Actions** from the top menu.
2. Select **New workflow**.
3. Select **set up a workflow yourself**.
4. Rename the workflow file if you prefer a different name other than **main.yml**. For example: **deployBicepFile.yml**.
5. Replace the content of the yml file with the following code:

   Service principal

   yml

   ```yaml
   name: Deploy Bicep file
   on: [push]
   jobs:
     build-and-deploy:
       runs-on: ubuntu-latest
   ```

```yaml
    steps:

    - name: Checkout code
      uses: actions/checkout@main

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

    - name: Deploy Bicep file
      uses: azure/arm-deploy@v1
      with:
        subscriptionId: ${{ secrets.AZURE_SUBSCRIPTION }}
        resourceGroupName: ${{ secrets.AZURE_RG }}
        template: ./main.bicep
        parameters: 'storagePrefix=mystore storageSKU=Standard_LRS'
        failOnStdErr: false
```

Replace `mystore` with your own storage account name prefix.

Note

You can specify a JSON format parameters file instead in the ARM Deploy action (example: `.azuredeploy.parameters.json`).

The first section of the workflow file includes:

- **name**: The name of the workflow.
- **on**: The name of the GitHub events that triggers the workflow. The workflow is triggered when there's a push event on the main branch.

6. Select **Commit changes**.

7. Select **Commit directly to the main branch**.

8. Select **Commit new file** (or **Commit changes**).

Updating either the workflow file or Bicep file triggers the workflow. The workflow starts right after you commit the changes.

Check workflow status

1. Select the **Actions** tab. You'll see a **Create deployBicepFile.yml** workflow listed. It takes 1-2 minutes to run the workflow.
2. Select the workflow to open it, and verify the `Status` is `Success`.

Clean up resources

When your resource group and repository are no longer needed, clean up the resources you deployed by deleting the resource group and your GitHub repository.

CLI

Azure CLI

```
az group delete --name exampleRG
```

Next steps

Bicep file structure and syntax

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A