

# Deploying Java to Azure App Service

## In this article

- Introduction
- Prerequisites
- Creating the workflow
- Additional resources

You can deploy your Java project to Azure App Service as part of your continuous deployment (CD) workflows.

**Note:** GitHub-hosted runners are not currently supported on GitHub Enterprise Server. You can see more information about planned future support on the [GitHub public roadmap](#).

## Introduction

This guide explains how to use GitHub Actions to build and deploy a Java project to [Azure App Service](#).

**Note:** If your GitHub Actions workflows need to access resources from a cloud provider that supports OpenID Connect (OIDC), you can configure your workflows to authenticate directly to the cloud provider. This will let you stop storing these credentials as long-lived secrets and provide other security benefits. For more information, see "[About security hardening with OpenID Connect](#)" and "[Configuring OpenID Connect in Azure](#)."

## Prerequisites

Before creating your GitHub Actions workflow, you will first need to complete the following setup steps:

- 1 Create an Azure App Service plan.

For example, you can use the Azure CLI to create a new App Service plan:

Bash



```
az appservice plan create \
  --resource-group MY_RESOURCE_GROUP \
  --name MY_APP_SERVICE_PLAN \
  --is-linux
```

In the command above, replace `MY_RESOURCE_GROUP` with your pre-existing Azure Resource Group, and `MY_APP_SERVICE_PLAN` with a new name for the App Service plan.

See the Azure documentation for more information on using the [Azure CLI](#):

- For authentication, see "[Sign in with Azure CLI](#)."
- If you need to create a new resource group, see "[az group](#)."

## 2 Create a web app.

For example, you can use the Azure CLI to create an Azure App Service web app with a Java runtime:

Bash

```
az webapp create \  
  --name MY_WEBAPP_NAME \  
  --plan MY_APP_SERVICE_PLAN \  
  --resource-group MY_RESOURCE_GROUP \  
  --runtime "JAVA|11-java11"
```

In the command above, replace the parameters with your own values, where `MY_WEBAPP_NAME` is a new name for the web app.

## 3 Configure an Azure publish profile and create an `AZURE_WEBAPP_PUBLISH_PROFILE` secret.

Generate your Azure deployment credentials using a publish profile. For more information, see "[Generate deployment credentials](#)" in the Azure documentation.

In your GitHub repository, create a secret named `AZURE_WEBAPP_PUBLISH_PROFILE` that contains the contents of the publish profile. For more information on creating secrets, see "[Using secrets in GitHub Actions](#)."

## 4 Optionally, configure a deployment environment. Environments are used to describe a general deployment target like `production`, `staging`, or `development`. When a GitHub Actions workflow deploys to an environment, the environment is displayed on the main page of the repository. You can use environments to require approval for a job to proceed, restrict which branches can trigger a workflow, gate deployments with custom deployment protection rules, or limit access to secrets. For more information about creating environments, see "[Using environments for deployment](#)."

# Creating the workflow

Once you've completed the prerequisites, you can proceed with creating the workflow.

The following example workflow demonstrates how to build and deploy a Java project to Azure App Service when there is a push to the `main` branch.

Ensure that you set `AZURE_WEBAPP_NAME` in the workflow `env` key to the name of the web app you created. If you want to use a Java version other than `11`, change `JAVA_VERSION`.

If you configured a deployment environment, change the value of `environment` to be the name of your environment. If you did not configure an environment, delete the `environment` key.

YAML

```
# This workflow uses actions that are not certified by GitHub.  
# They are provided by a third-party and are governed by  
# separate terms of service, privacy policy, and support  
# documentation.
```

```
# GitHub recommends pinning actions to a commit SHA.
# To get a newer version, you will need to update the SHA.
# You can also reference a tag or branch, but the action may change without
warning.
```

```
name: Build and deploy JAR app to Azure Web App
```

```
env:
  AZURE_WEBAPP_NAME: MY_WEBAPP_NAME    # set this to your application's name
  JAVA_VERSION: '11'                  # set this to the Java version to use
```

```
on:
  push:
    branches:
      - main
```

```
jobs:
  build:
    runs-on: ubuntu-latest
```

```
  steps:
    - uses: actions/checkout@v4

    - name: Set up Java version
      uses: actions/setup-java@v3
      with:
        java-version: ${ env.JAVA_VERSION }
        cache: 'maven'

    - name: Build with Maven
      run: mvn clean install

    - name: Upload artifact for deployment job
      uses: actions/upload-artifact@v3
      with:
        name: java-app
        path: '${ github.workspace }/target/*.jar'
```

```
  deploy:
    runs-on: ubuntu-latest
    needs: build
    environment:
      name: 'production'
      url: ${ steps.deploy-to-webapp.outputs.webapp-url }
```

```
  steps:
    - name: Download artifact from build job
      uses: actions/download-artifact@v3
      with:
        name: java-app

    - name: Deploy to Azure Web App
      id: deploy-to-webapp
      uses: azure/webapps-deploy@85270a1854658d167ab239bce43949edb336fa7c
      with:
        app-name: ${ env.AZURE_WEBAPP_NAME }
        publish-profile: ${ secrets.AZURE_WEBAPP_PUBLISH_PROFILE }
        package: '*.jar'
```

## Additional resources [↗](#)

The following resources may also be useful:

- For the original starter workflow, see [azure-webapps-java-jar.yml](#) in the GitHub Actions [starter-workflows](#) repository.
- The action used to deploy the web app is the official Azure [Azure/webapps-deploy](#) action.

- For more examples of GitHub Action workflows that deploy to Azure, see the [actions-workflow-samples](#) repository.

## Legal

© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Status](#) [Pricing](#) [Expert services](#) [Blog](#)