

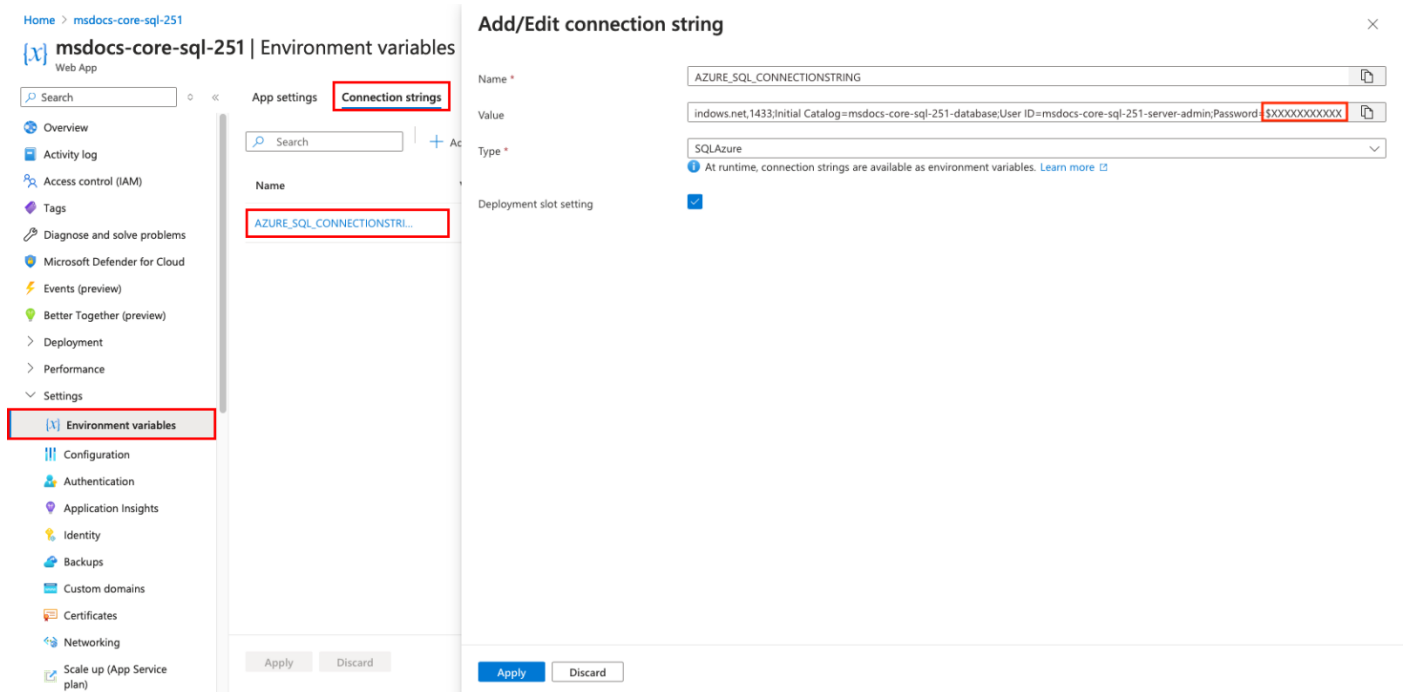
DEPLOY AN ASP.NET CORE AND AZURE SQL DATABASE APP TO AZURE APP SERVICE -2

3. SECURE CONNECTION SECRETS

The creation wizard generated the connectivity variable for you already as [.NET connection strings](#) and [app settings](#). However, the security best practice is to keep secrets out of App Service completely. You'll move your secrets to a key vault and change your app setting to [Key Vault references](#) with the help of Service Connectors.

STEP 1: RETRIEVE THE EXISTING CONNECTION STRING

1. In the left menu of the App Service page, select **Settings > Environment variables > Connection strings**.
2. Select **AZURE_SQL_CONNECTIONSTRING**.
3. In **Add/Edit connection string**, in the **Value** field, find the *Password=* part at the end of the string.
4. Copy the password string after *Password=* for use later. This connection string lets you connect to the SQL database secured behind a private endpoint. However, the secrets are saved directly in the App Service app, which isn't the best. Likewise, the Redis cache connection string in the **App settings** tab contains a secret. You'll change this.



STEP 2: CREATE A KEY VAULT FOR SECURE MANAGEMENT OF SECRETS

1. In the top search bar, type "key vault", then select **Marketplace > Key Vault**.
2. In **Resource Group**, select **msdocs-core-sql-XYZ_group**.
3. In **Key vault name**, type a name that consists of only letters and numbers.
4. In **Region**, set it to the same location as the resource group.

Azure Key Vault is a cloud service used to manage keys, secrets, and certificates. Key Vault eliminates the need for developers to store security information in their code. It allows you to centralize the storage of your application secrets which greatly reduces the chances that secrets may be leaked. Key Vault also allows you to securely store secrets and keys backed by Hardware Security Modules or HSMs. The HSMs used are Federal Information Processing Standards (FIPS) 140-2 Level 2 validated. In addition, key vault provides logs of all access and usage attempts of your secrets so you have a complete audit trail for compliance.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group * [Create new](#)

Instance details

Key vault name * ✓

Region *

Pricing tier *

Recovery options

[Give feedback](#)

STEP 3: SECURE THE KEY VAULT WITH A PRIVATE ENDPOINT

1. Select the **Networking** tab.
2. Unselect **Enable public access**.
3. Select **Create a private endpoint**.
4. In **Resource group**, select **msdocs-core-sql-XYZ_group**.
5. In the dialog, in **Location**, select the same location as your App Service app.
6. In **Name**, type **msdocs-core-sql-XYZVvaultEndpoint**.
7. In **Location**, select the same location as your App Service app.
8. In **Virtual network**, select the virtual network in the **msdocs-core-sql-XYZ_group** group.
9. In **Subnet**, select the available compatible subnet. The Web App wizard created it for your convenience.
10. Select **OK**.
11. Select **Review + create**, then select **Create**. Wait for the key vault deployment to finish. You should see "Your deployment is complete."

Home >

Create a key vault

Basics

Access configuration

Networking

Tags

You can connect to this key vault either publicly, via public IP address, or via a private endpoint.

Enable public access ☐

Private endpoint

Create a private endpoint to allow a private connection to this key vault.

[+ Create a private endpoint](#)

Name	Subscription	Resource group
Click on add button to add private endpoint		

Create private endpoint

Subscription *

Resource group *

Location *

Name *

Target sub-resource *

msdocs-core-sql-235_group

Canada Central

msdocs-core-sql-235VaultEndpoint

Vault

Networking

To deploy the private endpoint, select a virtual network subnet. [Learn more about private endpoint networking](#)

Virtual network *

Subnet *

vnet-zgiehxxw (msdocs-core-sql-235_group)

subnet-qgswdyqjib64

Private DNS integration

To connect privately with your private endpoint, you need a DNS record. We recommend that you integrate your private endpoint with a private DNS zone. You can also utilize your own DNS servers or create DNS records using the host files on your virtual machines. [Learn more about private DNS integration](#)

Integrate with private DNS zone *

Private DNS Zone *

Yes No

(New) privatelink.vaultcore.azure.net

OK

Discard

STEP 4: SQL DATABASE CONNECTOR

- In the top search bar, type *msdocs-core-sql*, then the App Service resource called **msdocs-core-sql-XYZ**.
- In the App Service page, in the left menu, select **Settings > Service Connector**. There are already two connectors, which the app creation wizard created for you.
- Select checkbox next to the SQL Database connector, then select **Edit**.
- Select the **Authentication** tab.
- In **Password**, paste the password you copied earlier.
- Select **Store Secret in Key Vault**.
- Under **Key Vault Connection**, select **Create new**. A **Create connection** dialog is opened on top of the edit dialog.

Home > msdocs-core-sql-251

msdocs-core-sql-251 | Service Connector ☆ ...

Web App

Search

Events (preview)

Better Together (preview)

> Deployment

> Performance

> Settings

> Environment variables

> Configuration

> Authentication

> Application Insights

> Identity

> Backups

> Custom domains

> Certificates

> Networking

> Scale up (App Service plan)

> Scale out (App Service plan)

> WebJobs

Service Connector

> Locks

> App Service plan

> Development Tools

> API

Using Service Connector will register resource pro

+ Create Edit Refresh Validate

Use Service Connector to connect your Azure Comp
environment variables set by Service Connector. Lea

☐ Service type

☒ SQL Database

☐ Cache for Redis

defaultConnector

Basics Authentication Networking

Select the authentication type you'd like to use between your compute service and target service. [Learn more](#)

☐ System assigned managed identity(Supported via Azure CLI. [Learn more](#)) ⓘ

☐ User assigned managed identity(Supported via Azure CLI. [Learn more](#)) ⓘ

☒ Connection string ⓘ

☐ Service principal(Supported via Azure CLI. [Learn more](#)) ⓘ

Continue with...

Database credentials

Key Vault

Username *
msdocs-core-sql-251-server-admin

Password *
.....

[Forgot password?](#)

☒ Store Secret In Key Vault ⓘ

Key Vault Connection * ⓘ
No item available

Create new

☐ Store Configuration in App Configuration ⓘ

Next : Networking Previous Cancel

STEP 5: ESTABLISH THE KEY VAULT CONNECTION

1. In the **Create connection** dialog for the Key Vault connection, in **Key Vault**, select the key vault you created earlier.
2. Select **Review + Create**.
3. When validation completes, select **Create**.

Create connection



Basics

Networking

Review + Create

Select the service instance and client type.

Service type * ⓘ

Key Vault

Connection name * ⓘ

keyvault_ca5fa

Subscription * ⓘ

Visual Studio Enterprise Subscription

Key vault * ⓘ

vault03940324

[Create new](#)

Client type * ⓘ

.NET

Next : Networking

Cancel

[Report an issue.](#)

STEP 6: FINALIZE THE SQL DATABASE CONNECTOR SETTINGS

1. You're back in the edit dialog for **defaultConnector**. In the **Authentication** tab, wait for the key vault connector to be created. When it's finished, the **Key Vault Connection** dropdown automatically selects it.
2. Select **Next: Networking**.
3. Select **Configure firewall rules to enable access to target service**. The app creation wizard already secured the SQL database with a private endpoint.
4. Select **Save**. Wait until the **Update succeeded** notification appears.

defaultConnector



Basics

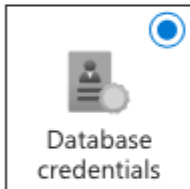
Authentication

Networking

Select the authentication type you'd like to use between your compute service and target service. [Learn more](#)

- ☐ System assigned managed identity(Supported via Azure CLI. [Learn more](#)) ⓘ
- ☐ User assigned managed identity(Supported via Azure CLI. [Learn more](#)) ⓘ
- ☒ Connection string ⓘ
- ☐ Service principal(Supported via Azure CLI. [Learn more](#)) ⓘ

Continue with...



Username *

msdocs-core-sql-251-server-admin

Password *

.....

[Forgot password?](#)

☒ Store Secret In Key Vault ⓘ

Key Vault Connection * ⓘ

vault03940324 (keyvault_ca5fa)

[Create new](#)

☐ Store Configuration in App Configuration ⓘ

Next : Networking

Previous

Cancel

STEP 7: CONFIGURE THE REDIS CONNECTOR TO USE KEY VAULT SECRETS

1. In the Service Connector page, select the checkbox next to the Cache for Redis connector, then select **Edit**.
2. Select the **Authentication** tab.
3. Select **Store Secret in Key Vault**.

- Under **Key Vault Connection**, select the key vault you created.
- Select **Next: Networking**.
- Select **Configure firewall rules to enable access to target service**.
- Select **Save**. Wait until the **Update succeeded** notification appears.

The screenshot shows the Azure portal interface for the 'msdocs-core-sql-251' Service Connector. On the left, the 'Service Connector' menu item is highlighted with a red box. The main panel shows the 'Authentication' tab, also highlighted with a red box. Under 'Service type', 'Cache for Redis' is selected with a red box. The 'Key Vault Connection' dropdown is set to 'vault03940324 (keyvault_ca5fa)' and is also highlighted with a red box. At the bottom, the 'Next : Networking' button is highlighted with a red box. Other tabs like 'Basics' and 'Networking' are visible at the top of the right panel.

STEP 8: VERIFY THE KEY VAULT INTEGRATION

- From the left menu, select **Settings > Environment variables > Connection strings** again.
- Next to **AZURE_SQL_CONNECTIONSTRING**, select **Show value**. The value should be `@Microsoft.KeyVault(...)`, which means that it's a [key vault reference](#) because the secret is now managed in the key vault.
- To verify the Redis connection string, select the **App setting** tab. Next to **AZURE_REDIS_CONNECTIONSTRING**, select **Show value**. The value should be `@Microsoft.KeyVault(...)` too.

The screenshot shows the Azure App Service settings interface. On the left, the 'Environment variables' option is selected in the sidebar. The main content area is titled 'App settings' and 'Connection strings'. A table lists connection strings with columns for Name, Value, and Deploy. One entry is visible: 'AZURE_SQL_CONNECTI...' with a 'Show value' button in the Value column. The 'Show value' button and the 'Connection strings' header are highlighted with red boxes. At the bottom, there are 'Apply' and 'Discard' buttons, and a 'Send us your feedback' link.

Name	Value	Deploy
AZURE_SQL_CONNECTI...	Show value	✓

To summarize, the process for securing your connection secrets involved:

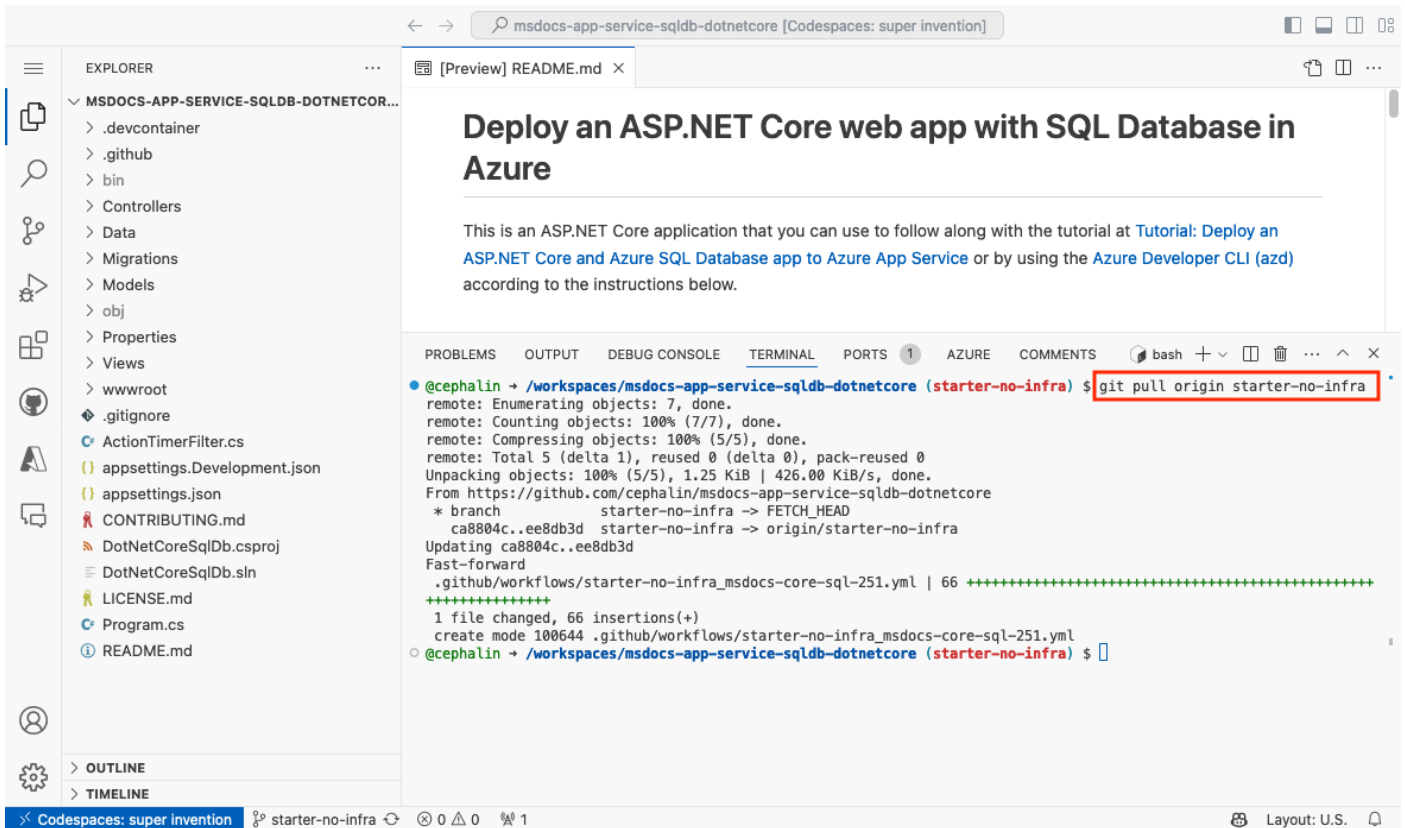
- Retrieving the connection secrets from the App Service app's environment variables.
- Creating a key vault.
- Creating a Key Vault connection with the system-assigned managed identity.
- Updating the service connectors to store the secrets in the key vault.

4. DEPLOY SAMPLE CODE

In this step, you configure GitHub deployment using GitHub Actions. It's just one of many ways to deploy to App Service, but also a great way to have continuous integration in your deployment process. By default, every git push to your GitHub repository kicks off the build and deploy action.

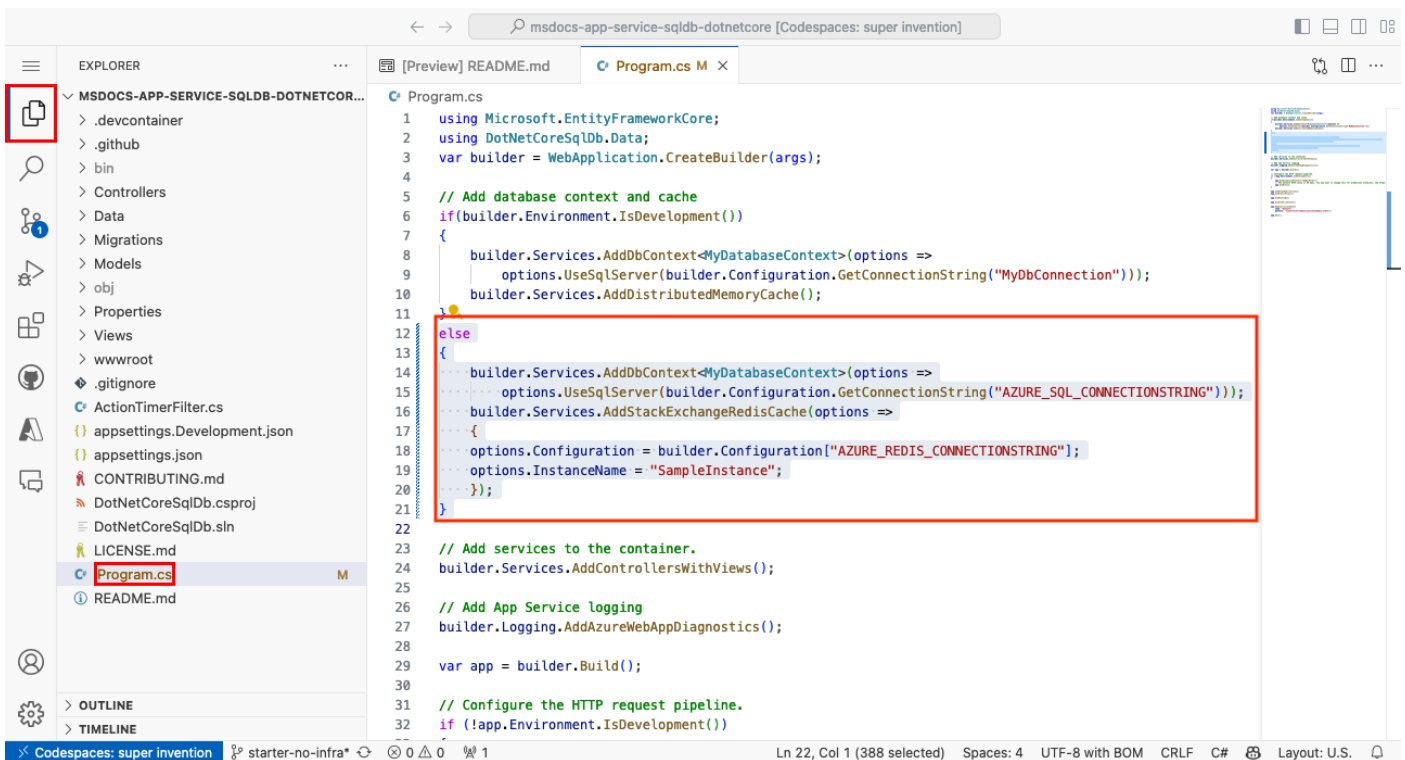
STEP 1:

Back in the GitHub codespace of your sample fork, run `git pull origin starter-no-infra`. This pulls the newly committed workflow file into your codespace.



STEP 2:

1. Open *Program.cs* in the explorer.
2. Find the commented code (lines 12-21) and uncomment it. This code connects to the database by using `AZURE_SQL_CONNECTIONSTRING` and connects to the Redis cache by using the app setting `AZURE_REDIS_CONNECTIONSTRING`.




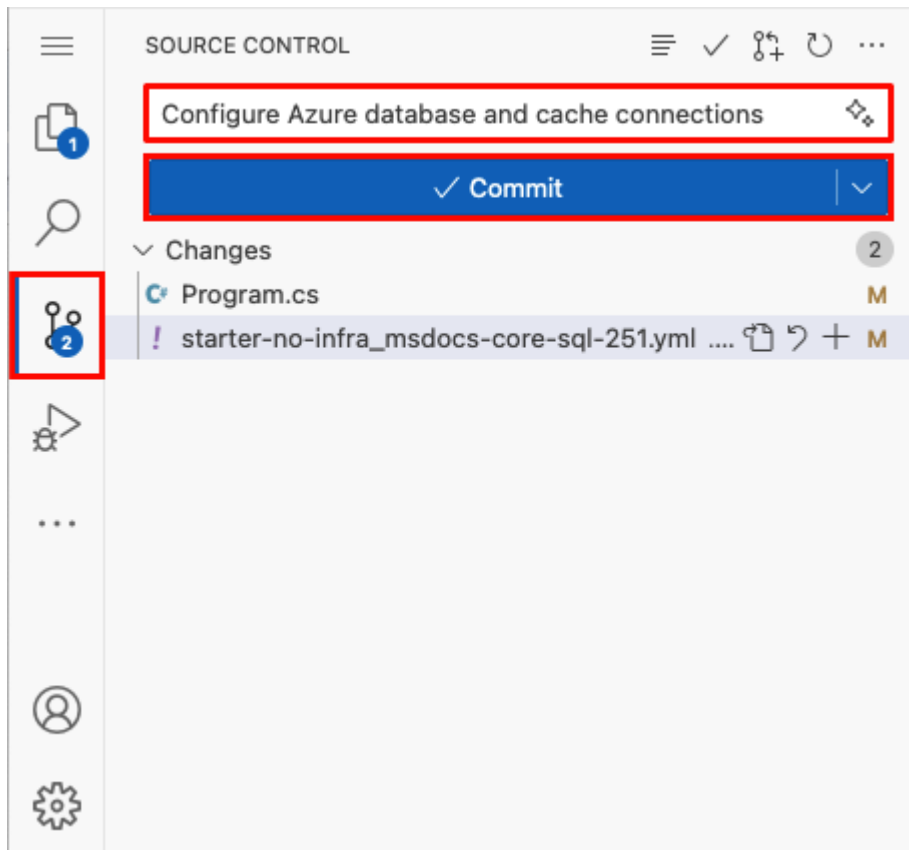
STEP 3:

1. Open `.github/workflows/starter-no-infra_msdocs-core-sql-XYZ` in the explorer. This file was created by the App Service create wizard.
2. Under the dotnet publish step, add a step to install the [Entity Framework Core tool](#) with the command `dotnet tool install -g dotnet-ef --version 8.*`.
3. Under the new step, add another step to generate a database [migration bundle](#) in the deployment package: `dotnet ef migrations bundle --runtime linux-x64 -o ${env.DOTNET_ROOT}/myapp/migrationsbundle`. The migration bundle is a self-contained executable that you can run in the production environment without needing the .NET SDK. The App Service linux container only has the .NET runtime and not the .NET SDK.

```
1 # Docs for the Azure Web Apps Deploy action: https://github.com/Azure/webapps-deploy
2 # More GitHub Actions for Azure: https://github.com/Azure/actions
3
4 name: Build and deploy ASP.Net Core app to Azure Web App - msdocs-core-sql-251
5
6 on:
7   push:
8     branches:
9       - starter-no-infra
10  workflow_dispatch:
11
12 jobs:
13   build:
14     runs-on: ubuntu-latest
15
16     steps:
17       - uses: actions/checkout@v4
18
19       - name: Set up .NET Core
20         uses: actions/setup-dotnet@v1
21         with:
22           dotnet-version: '8.x'
23           include-prerelease: true
24
25       - name: Build with dotnet
26         run: dotnet build --configuration Release
27
28       - name: dotnet publish
29         run: dotnet publish -c Release -o ${env.DOTNET_ROOT}/myapp
30
31       - name: Install dotnet ef
32         run: dotnet tool install --global dotnet-ef --version 8.*
33
34       - name: Create migrations bundle
35         run: dotnet ef migrations bundle --runtime linux-x64 -o ${env.DOTNET_ROOT}/myapp/migrationsbundle
36
37       - name: Upload artifact for deployment job
38         uses: actions/upload-artifact@v3
39         with:
```

STEP 4:

1. Select the **Source Control** extension.
2. In the textbox, type a commit message like Configure Azure database and cache connections. Or, select  and let GitHub Copilot generate a commit message for you.
3. Select **Commit**, then confirm with **Yes**.
4. Select **Sync changes 1**, then confirm with **OK**.



STEP 5:

Back in the Deployment Center page in the Azure portal:

1. Select the **Logs** tab, then select **Refresh** to see the new deployment run.
2. In the log item for the deployment run, select the **Build/Deploy Logs** entry with the latest timestamp.

Save

Discard

Browse

Manage publish profile

Sync

Leave Feedback

Settings

Logs

FTPS credentials

Refresh

Delete

Time	Com...	Logs	Commit Author	Status	Message
Friday, June 28, 2024 (3)					
06/28 2024, 5:44:...	016986b	Build/Deploy Lo...		In Progress...	Configure Azure database and cache connections
06/28 2024, 5:25:...	a5858da	App Logs	N/A	Success (Active)	Add or update the Azure App Service build and deployment workflow config
06/28 2024, 5:23:...	ee8db3d	Build/Deploy Lo...		Success	Add or update the Azure App Service build and deployment workflow config

STEP 6:

You're taken to your GitHub repository and see that the GitHub action is running. The workflow file defines two separate stages, build and deploy. Wait for the GitHub run to show a status of **Success**. It takes about 5 minutes.

The screenshot shows the GitHub Actions interface for the repository `<github-alias> / msdocs-app-service-sqlldb-dotnetcore`. The workflow is titled `Build and deploy ASP.Net Core app to Azure Web App - msdocs-core-sql-251`. The current job is `Configure Azure database and cache connections #2`, which has a status of **Success**. The job was triggered via a push 6 minutes ago by `<github-alias> pushed` to the `starter-no-infra` branch. The total duration of the job is `3m 17s`, and it has produced `1` artifact. The workflow file is `starter-no-infra_msdocs-core-sql-251.yml`, which is triggered on push. The workflow consists of two jobs: `build` (duration `1m 55s`) and `deploy` (duration `1m 0s`). The `deploy` job is linked to the `build` job. The `deploy` job is configured to deploy to `msdocs-core-sql-251.azurewebsites.net`. The interface includes a sidebar with links to `Summary`, `Jobs`, `Run details`, `Usage`, and `Workflow file`. The `Jobs` section shows a list of jobs: `build` and `deploy`, both with a status of **Success**. The `Run details` section shows the workflow file and the job details.

← Build and deploy ASP.Net Core app to Azure Web App - msdocs-core-sql-251

✓ **Configure Azure database and cache connections #2** Re-run all jobs ...

Summary

Jobs

- ✓ build
- ✓ deploy

Run details

- Usage
- Workflow file

Triggered via push 6 minutes ago

`<github-alias> pushed` -o- 9ab82de `starter-no-infra` **Success**

Total duration: **3m 17s** | Artifacts: **1**

starter-no-infra_msdocs-core-sql-251.yml
on: push

✓ build (1m 55s) → ✓ deploy (1m 0s)
`msdocs-core-sql-251.azurewebsites.net`