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

In this tutorial, you learn how to deploy a data-driven ASP.NET Core app to Azure App Service and connect to an Azure SQL Database. You'll also deploy an Azure Cache for Redis to enable the caching code in your application. Azure App Service is a highly scalable, self-patching, web-hosting service that can easily deploy apps on Windows or Linux. Although this tutorial uses an ASP.NET Core 8.0 app, the process is the same for other versions of ASP.NET Core.

IN THIS TUTORIAL, YOU LEARN HOW TO:

- Create a secure-by-default App Service, SQL Database, and Redis cache architecture.
- Secure connection secrets using a managed identity and Key Vault references.
- Deploy a sample ASP.NET Core app to App Service from a GitHub repository.
- Access App Service connection strings and app settings in the application code.
- Make updates and redeploy the application code.
- Generate database schema by uploading a migrations bundle.
- Stream diagnostic logs from Azure.
- Manage the app in the Azure portal.
- Provision the same architecture and deploy by using Azure Developer CLI.
- Optimize your development workflow with GitHub Codespaces and GitHub Copilot.

PREREQUISITES

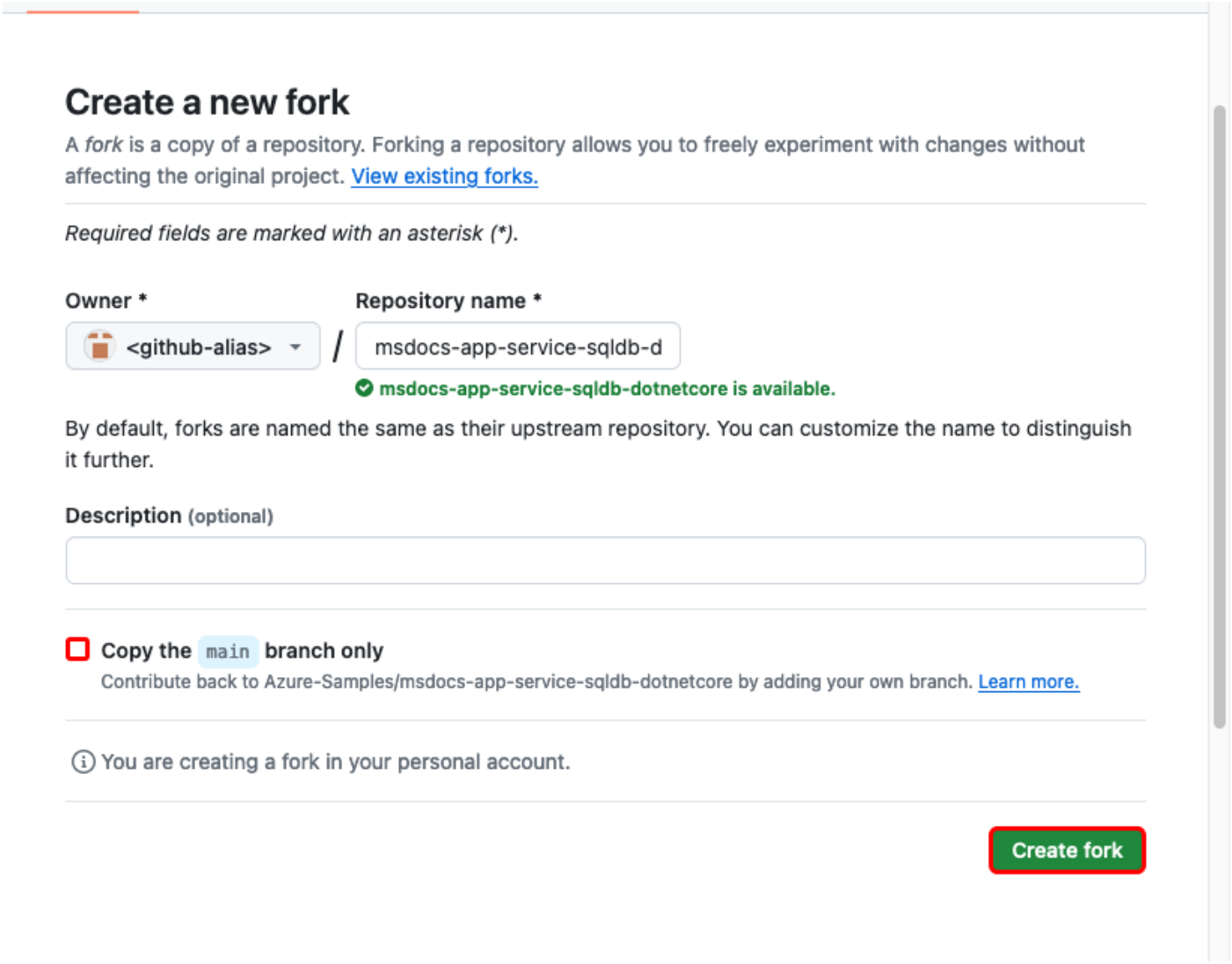
- An Azure account with an active subscription. If you don't have an Azure account, you can create one for free.
- A GitHub account. You can also get one for free.
- Knowledge of ASP.NET Core development.
- **(Optional)** To try GitHub Copilot, a GitHub Copilot account. A 30-day free trial is available.

1. RUN THE SAMPLE

First, you set up a sample data-driven app as a starting point. For your convenience, the sample repository, includes a dev container configuration. The dev container has everything you need to develop an application, including the database, cache, and all environment variables needed by the sample application. The dev container can run in a GitHub codespace, which means you can run the sample on any computer with a web browser.

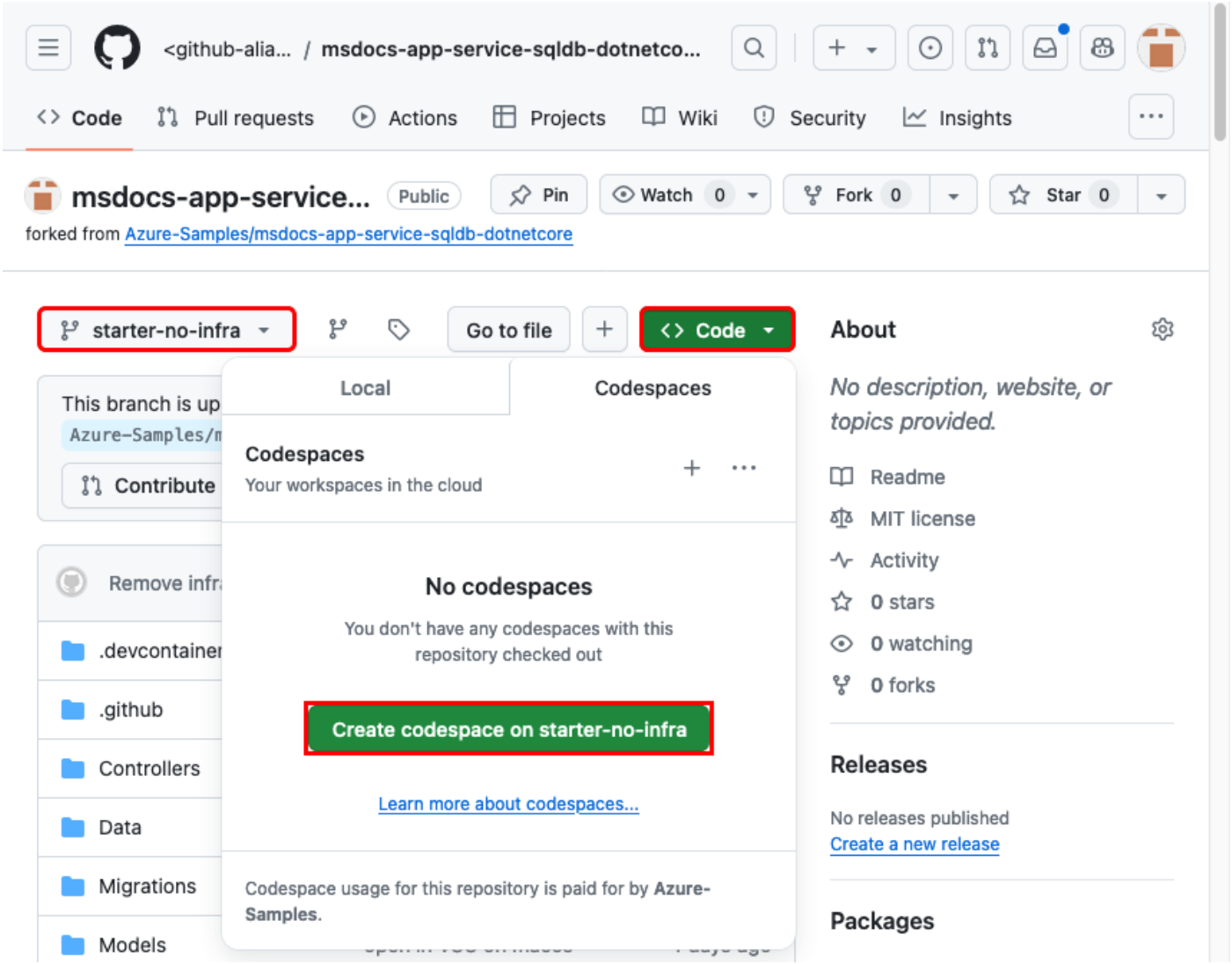
Step 1: In a new browser window:

1. Sign in to your GitHub account.
2. Navigate to https://github.com/Azure-Samples/msdocs-app-service-sqldb-dotnetcore/fork.
3. Unselect **Copy the main branch only**. You want all the branches.
4. Select **Create fork**.



Step 2: In the GitHub fork:

1. Select **main** > **starter-no-infra** for the starter branch. This branch contains just the sample project and no Azure-related files or configuration.
2. Select **Code** > **Codespaces** > **Create codespace on starter-no-infra**. The codespace takes a few minutes to set up.



Step 3: In the codespace terminal:

1. Run database migrations with dotnet ef database update.
2. Run the app with dotnet run.
3. When you see the notification Your application running on port 5093 is available., select **Open in Browser**. You should see the sample application in a new browser tab. To stop the application, type Ctrl+C.

2. CREATE APP SERVICE, DATABASE, AND CACHE

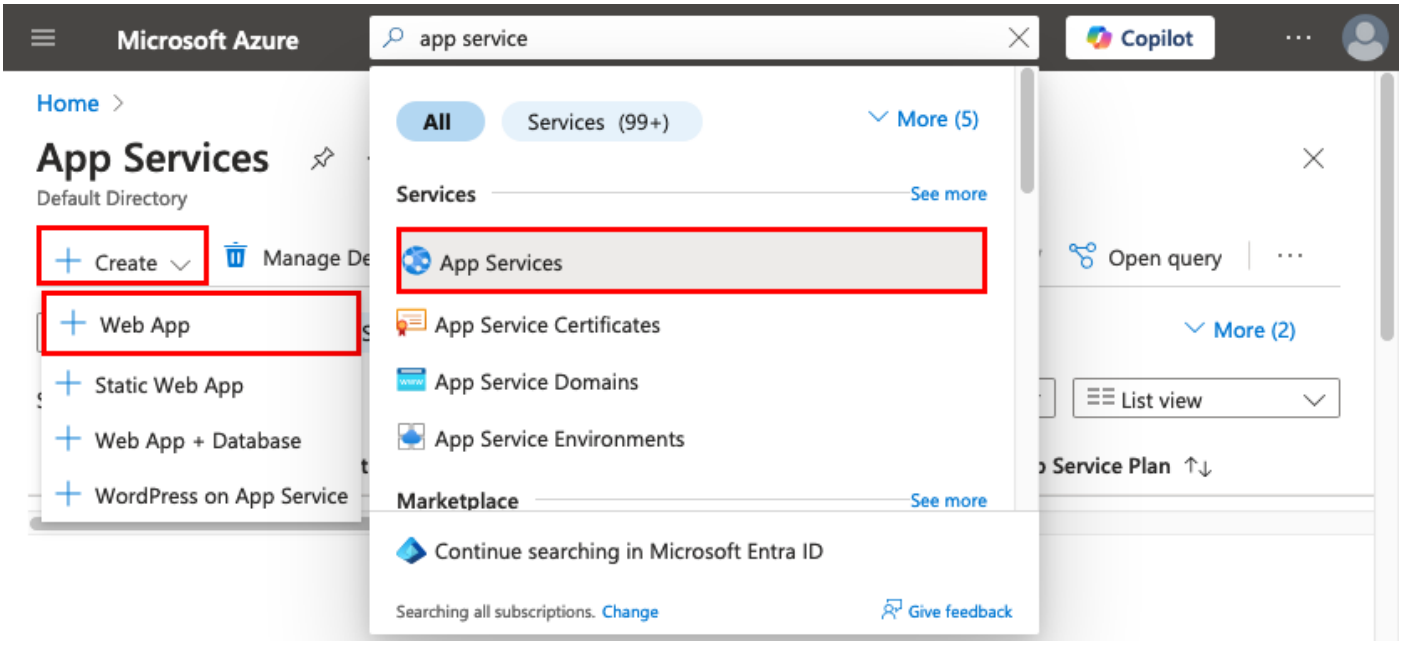
In this step, you create the Azure resources. The steps used in this tutorial create a set of secure-by-default resources that include App Service, Azure SQL Database, and Azure Cache. For the creation process, you'll specify:

- The **Name** for the web app. It's used as part of the DNS name for your app.
- The **Region** to run the app physically in the world. It's also used as part of the DNS name for your app.
- The **Runtime stack** for the app. It's where you select the .NET version to use for your app.
- The **Hosting plan** for the app. It's the pricing tier that includes the set of features and scaling capacity for your app.
- The **Resource Group** for the app. A resource group lets you group (in a logical container) all the Azure resources needed for the application.

Sign in to the Azure portal and follow these steps to create your Azure App Service resources.

Step 1: In the Azure portal:

1. In the top search bar, type *app service*.
2. Select the item labeled **App Service** under the **Services** heading.
3. Select **Create** > **Web App**. You can also navigate to the creation wizard directly.



Step 2: In the **Create Web App** page, fill out the form as follows.

1. *Name*: **msdocs-core-sql-XYZ**. A resource group named **msdocs-core-sql-XYZ_group** will be generated for you.
2. *Runtime stack*: **.NET 8 (LTS)**.
3. *Operating System*: **Linux**.
4. *Region*: your preferred region.
5. *Linux Plan*: **Create new** and use the name **msdocs-core-sql-XYZ**.
6. *Pricing plan*: **Basic B1**. When you're ready, you can scale up to a different pricing tier.

Home > App Services >

Create Web App

Basics | Database | Deployment | Networking | Monitor + secure | Tags | Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. Learn more

Project Details

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

Subscription *

Resource Group * (New) msdocs-core-sql_group

Create new

Instance Details

Name: msdocs-core-sql .azurewebsites.net

Secure unique default hostname on. More about this update

Publish * Code Container

Runtime stack * .NET 8 (LTS)

Operating System * Linux Windows

Region * Canada Central

Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. Learn more

Linux Plan (Canada Central) * (New) msdocs-core-sql

Create new

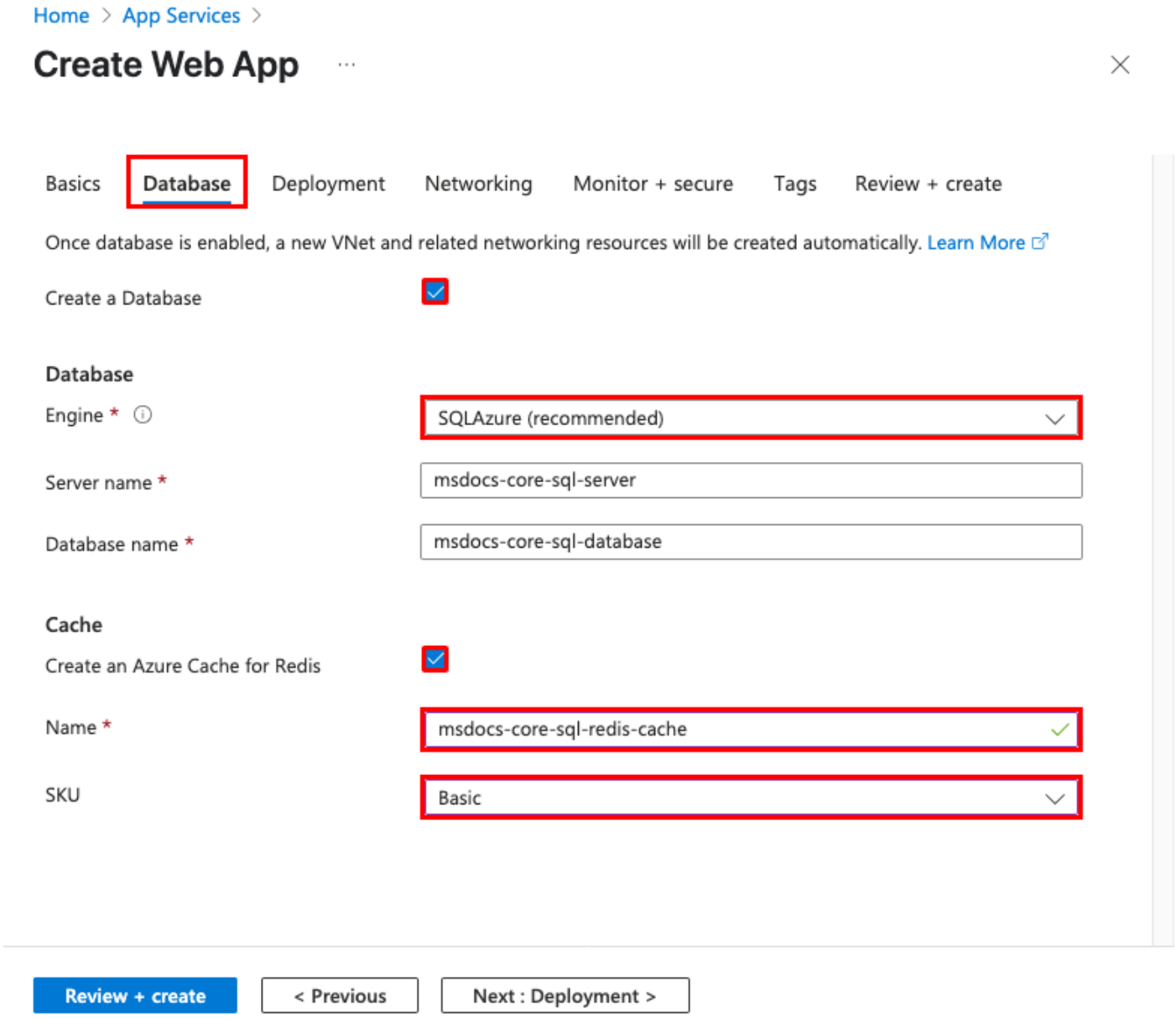
Pricing plan Basic B1 (100 total ACU, 1.75 GB memory, 1 vCPU)

Explore pricing plans

Review + create | < Previous | Next : Database >

Step 3:

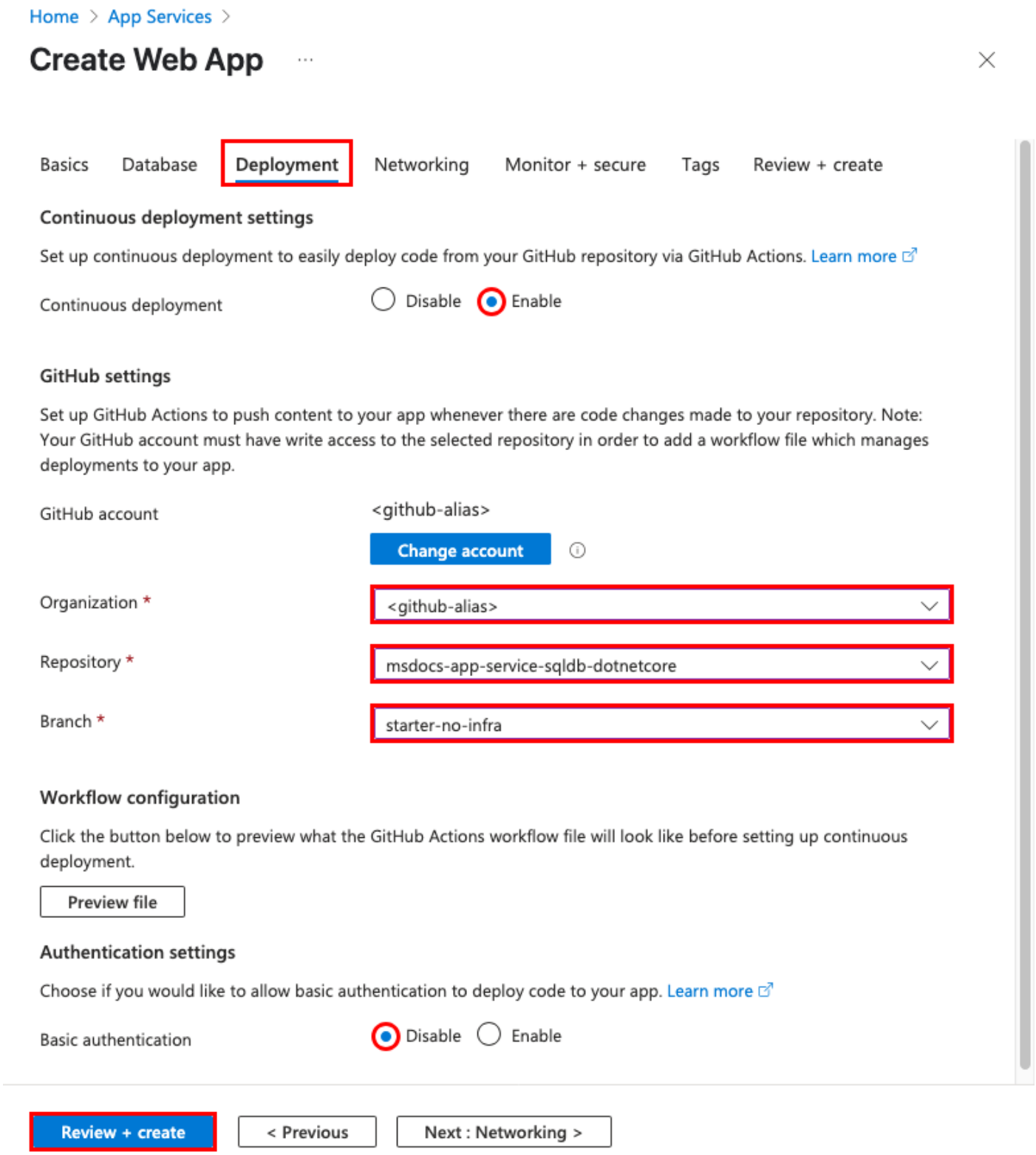
1. Select the **Database** tab.
2. Select **Create a Database**.
3. In **Engine**, select **SQLAzure**.
4. Select **Create an Azure Cache for Redis**.
5. In **Name** (under Cache), enter a name for the cache.
6. In **SKU**, select **Basic**.



Step 4:

1. Select the **Deployment** tab.
2. Enable **Continuous deployment**.
3. In **Organization**, select your GitHub alias.
4. In **Repository**, select **msdocs-app-service-sqldb-dotnetcore**.
5. In **Branch**, select **starter-no-infra**.

6. Make sure **Basic authentication** is disabled.
7. Select **Review + create**.
8. After validation completes, select **Create**.



Step 5: The deployment takes a few minutes to complete. Once deployment completes, select the **Go to resource** button. You're taken directly to the App Service app, but the following resources are created:

- **Resource group**: The container for all the created resources.
- **App Service plan**: Defines the compute resources for App Service. A Linux plan in the *Basic* tier is created.

- **App Service**: Represents your app and runs in the App Service plan.
- **Virtual network**: Integrated with the App Service app and isolates back-end network traffic.
- **Private endpoints**: Access endpoints for the key vault, the database server, and the Redis cache in the virtual network.
- **Network interfaces**: Represents private IP addresses, one for each of the private endpoints.
- **Azure SQL Database server**: Accessible only from behind its private endpoint.
- **Azure SQL Database**: A database and a user are created for you on the server.
- **Azure Cache for Redis**: Accessible only from behind its private endpoint.
- **Key vault**: Accessible only from behind its private endpoint. Used to manage secrets for the App Service app.
- **Private DNS zones**: Enable DNS resolution of the key vault, the database server, and the Redis cache in the virtual network.