White paper

Microsoft Azure

Azure Proof of Concept Guide for Developers

Azure Proof of Concept Guide for Developers

Introduction

With the advent of new technologies, many organisations are embarking on proof of concept projects to learn and explore new capabilities, and conduct feasibility assessments of proposed concepts. A proof of concept is an important first step in cultivating business innovations.

In a recent survey (see Figure 1: How tools/platforms are evaluated and adopted by developers) conducted by Packt Publishing, building a proof of concept project is the most popular choice for developers when it comes to evaluating and adopting new tools and platforms.

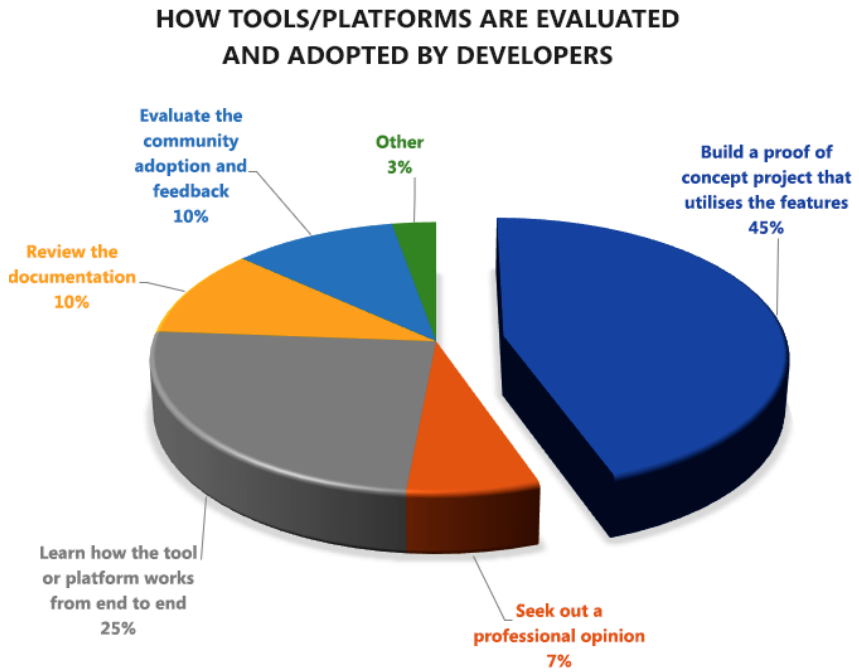


Figure 1: How tools/platforms are evaluated and adopted by developers

What is a proof of concept?

A proof of concept is a scoped, time-bound exercise with specific, measurable goals and metrics of success. Ideally, it should have some basis in business reality so that the results are meaningful.

Benefits of proof of concept projects

A proof of concept project can be a valuable tool to evaluate whether a potential technology or concept can be used to fulfil the requirements of a business solution. It can help identify any potential technical and logistical issues before the service is implemented in a mainstream project. In addition, it provides timely insights on the technology while mitigating risks by allowing key decisions to be made in the early stages of the development process.

There are several benefits that proof of concept projects provide:

Experimenting with new technologies

As technologies evolve, proof of concept projects enable organisations to discover, learn and experiment with groundbreaking technologies that can potentially be used in upcoming projects. Under the guidelines of a well-scoped and time-framed execution, the development team can quickly ramp up on new technology without posing risks to the company's mainstream large-scale projects. Best of all, the success of the proof of concept project might be adapted in a mainstream project in the future. As a result, proof of concept projects encourage innovation.

Minimising risks

Prior to embarking on a high-risk, large-scale and potentially expensive project, it is a good practice to minimise risks and costs by performing a quick validation of the parts of the project that are considered risky. For example, a project team could identify and isolate a particular technology used in a large-scale project, and flag it as risky due to the development team's inexperience with such technology. The project team can perform a quick feasibility experimentation and assessment of said technology by scoping out a small sub-project with a fixed budget and timeline. The risks are mitigated as proof of concept projects are typically executed in a controlled and sandboxed environment.

About this guide

The purpose of the Azure Proof of Concept Guide for Developers is to present the reader with guidance on how to plan a successful proof of concept project. It will also provide directions on how to start developing in the Microsoft Azure cloud platform. This guide is designed for developers and architects who are embarking on their first journey into Azure.

The methodology outlined in Chapter 1: Proof of concept guide builds the core foundation necessary for planning and executing a successful proof of concept project.

In Chapter 2: Sample project – implementing a web app using Azure Static Web Apps and Chapter 3: Sample project – building an intelligent chat bot, we will showcase two practical projects that might inspire you to come up with your own proof of concept projects.

In Chapter 4: An overview of Azure for developers, we introduce you to the Azure platform. For those who are new to Azure, you will learn how to obtain an Azure free account to begin your proof of concept journey on Azure. You will also learn how to pick the appropriate cloud model and services to start developing your project for Azure.

We will also provide you with some useful learning materials in Chapter 5: Further learning and resources.

What is Microsoft Azure?

The focus of this guide will be on Microsoft Azure. Azure is a cloud platform that enables you to host your existing applications with minimum changes, as well as develop new cloud-native applications. There is a vast number of ready-to-use Azure services that you can integrate with your applications to instantly take advantage of new cloud capabilities, while minimising the need to develop those services yourself.

By hosting your applications in Azure, you can build a Minimum Viable Product and then scale your application as your customer demand grows. Azure offers the reliability that is needed for high-availability applications, even including failover between different regions. The Azure portal lets you easily manage all your Azure services. You can also manage your services programmatically by using service-specific APIs and templates.

We will learn more about Azure in Chapter 4: An overview of Azure for developers.

With so much to cover, let's get started with the proof of concept guide.

Chapter 1: Proof of concept guide

Step 1: Defining your goal and success criteria

Most proof of concept projects are results driven. Set the goal on what you want to accomplish from your proof of concept project and determine what would be considered a successful execution of your project by stating the success criteria.

We want to prove whether a concept works or not without investing a huge amount of time and resources. If the proof of concept project succeeds and achieves the expected results, the project team can give the green light to proceed with the next steps. However, should the experimentation fail, it would have failed within a fixed budget, timeline, scope and set of resources, which would be minimal costs in the grand scheme of things. Therefore, the benefit of doing a proof of concept project is that it allows the project team to quickly determine whether to continue to pursue the concept, abandon it or make alternate choices. Bear in mind when scoping out your project (refer to Step 3: Scoping the proof of concept project) that the goal of your proof of concept is to succeed quickly (and conversely, fail quickly) so that subsequent decisions can be made in a timely manner.

Step 2: Setting your timeline and budget

Once the goal and success criteria for the proof of concept project have been defined, you will set the timeline and budget.

In our experience, proof of concept projects have the best outcome when they are timeboxed between two to four weeks. This provides enough time for the work to be completed without the burden of too many use cases and complex test matrices.

Here are some tips:

- Make realistic estimates of the time that will be required to complete the tasks in your proof of concept.
- If you find that your proof of concept is estimated to run longer than four weeks, consider reducing its scope to focus on the highest priority goals.
- Get buy-ins from all key resources and sponsors for the timeline and budget before proceeding.

Now that you have determined your timeline and budget, let's move on to scoping the proof of concept.

Step 3: Scoping the proof of concept project

It is important to define the scope for your proof of concept project before you start the implementation to prevent scope creep. Knowing in advance that the resources will not stay around indefinitely sets appropriate expectations for the stakeholders of the proof of concept.

Tip: Avoid turning a proof of concept project into a production project by defining your scope before starting the project. Then, throughout the course of the project, ensure that everyone involved adheres to the scope.

The following key questions will help you determine the scope of your proof of concept project:

- What do you want to learn or achieve from your proof of concept?
- What are the success criteria?
- What workloads or scenarios will be covered?

Tip: To ensure that your proof of concept project can be scoped and completed in a timely manner, keep the workloads and scenarios as small as possible.

- What resources must be available?
- Who are the users and teams to validate the outcome?
- What is the duration of the project?

Tip: Choose a timeline that aligns well with the planned scope, such as two weeks or four weeks.

- After the proof of concept is complete, what will happen with the resources that were allocated during the project? Do you plan to discard those resources?

With the scope defined, we can create a high-level architecture for the proof of concept.

Step 4: Creating a high-level architecture

Keep in mind that you are developing a proof of concept to validate whether a concept is feasible. Therefore, in order to keep the goal of your project achievable within your well-scoped timeline, when you create your high-level architecture, you must decide which essential components will be part of the proof of concept and which non-essential components should be excluded from it.

As a rule of thumb, the more components you include in your proof of concept, the more complicated your project will become and the longer it will be for you to achieve your success criteria.

As part of your high-level architecture, you will also need to choose the appropriate cloud model to host your proof of concept application. You will learn 'How to choose the appropriate cloud model for your Azure proof of concept project' in Chapter 4: An overview of Azure for developers.

As a best practice, your proof of concept projects should always reside in a separate Azure subscription from production. Ideally, a Dev/Test subscription should be used to keep the cost low. You will learn more about Azure subscription in the Getting started with Microsoft Azure section in Chapter 4: An overview of Azure for developers.

Tip: There is no need to be perfect. It is tempting to try and architect a perfect proof of concept, which will mirror the eventual product if it is rolled out to production. However, for a proof of concept project, this would be counterproductive. The closer you attempt to get to perfection, the more time and effort you will have to exert from start to finish. That would be undesirable. As you recall, the purpose of the proof of concept is to quickly prove a certain well-scoped concept. It is to allow you to make the correct decisions in a timely manner. Therefore, the focus of the should always be on selecting the smallest essential dependencies and associated workloads that meet specific measurable goals, to help guarantee a swift victory.

With the architectural plan in place, you are ready to assemble your team for the proof of concept project.

Step 5: Assembling your team

For a small one-off, simple, non-critical experimental proof of concept, a one-person team might suffice. However, for most typical proof of concept projects where the results are critical and could influence the decision of a bigger project, you should identify the mandatory team members needed and the commitment required to support your proof of concept. The team that you are assembling must reflect the scope of your project.

Step 6: Implementation and testing

With the goal, timeline, budget and scope defined and your team assembled, you can begin implementing your proof of concept project based on the high-level architecture. To maximise execution success, follow modern DevOps processes with iterative development and testing throughout your implementation.

Step 7: When your proof of concept is complete

Once your proof of concept is complete, evaluate whether you have met the success criteria which you defined in Step 1: Defining your goal and success criteria.

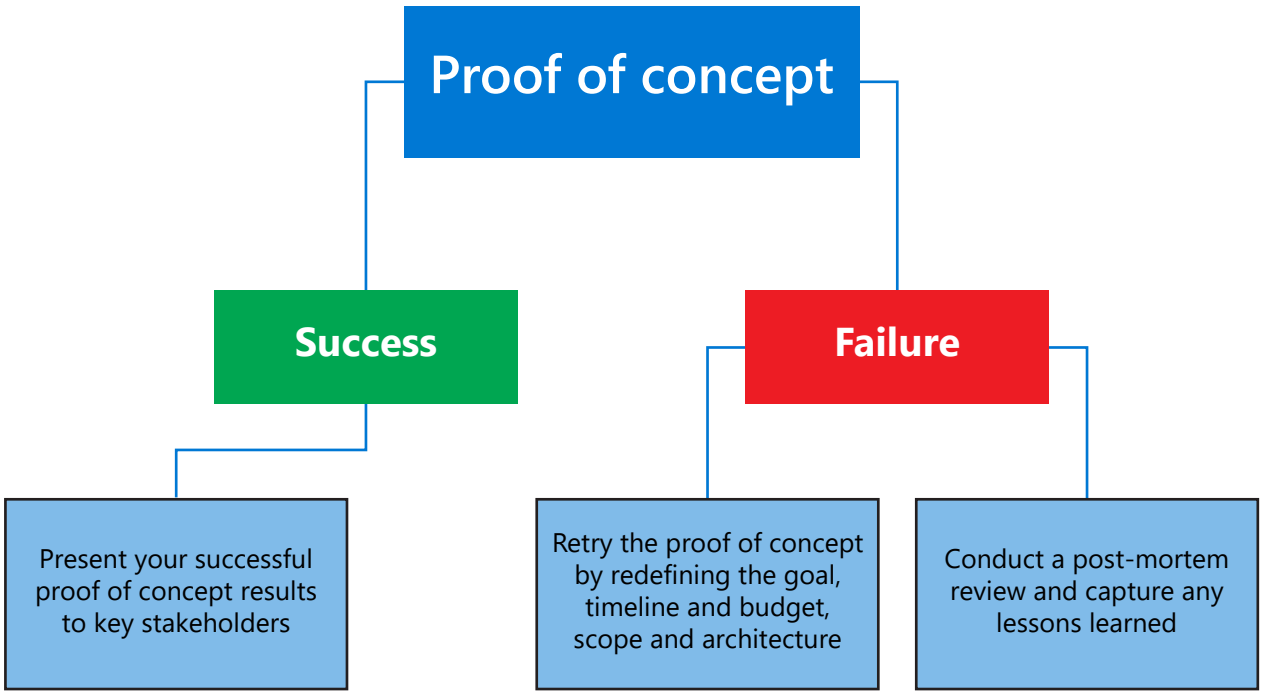


Figure 2: Evaluate your proof of concept

If your proof of concept is successful

- Present your successful results to key stakeholders.

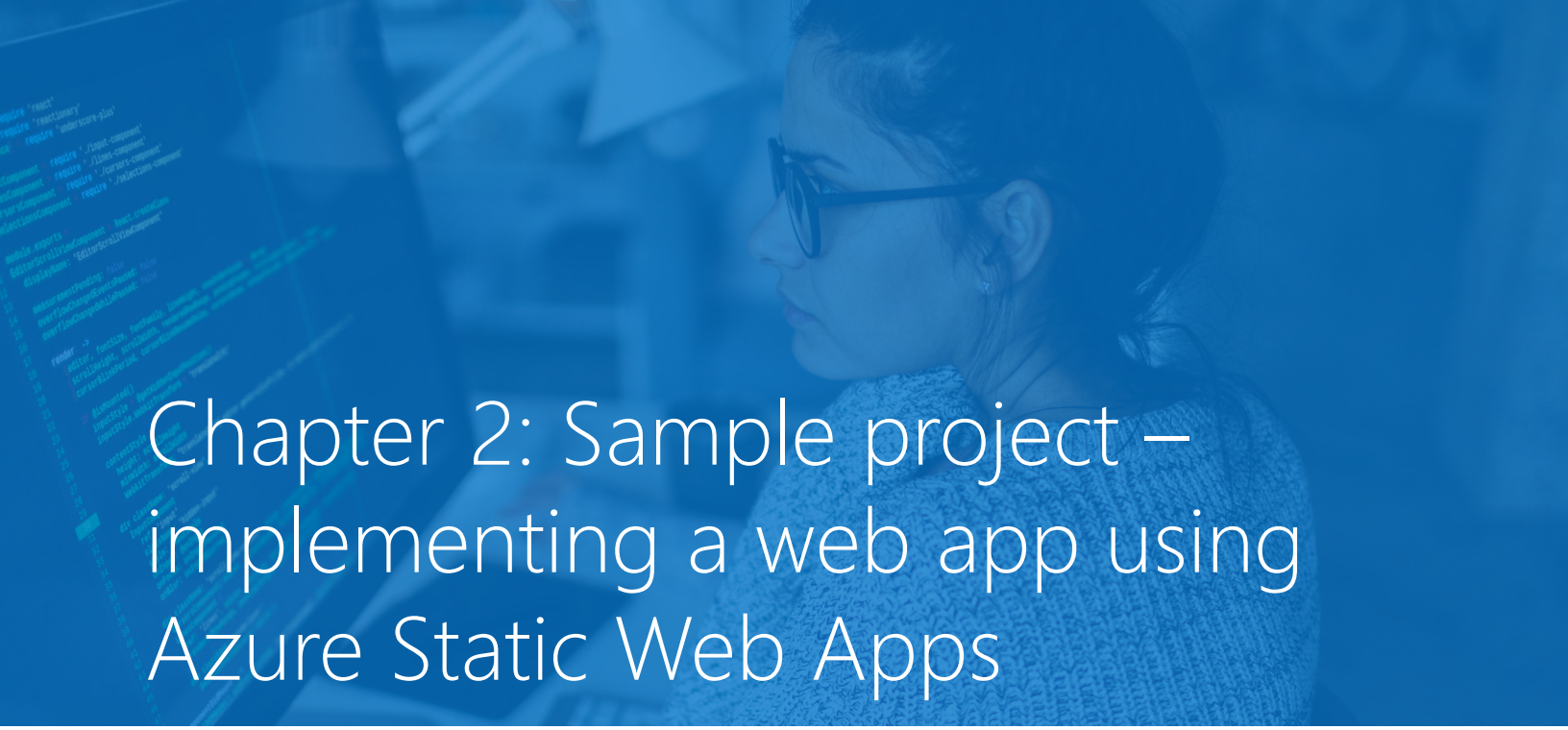
  Tip: When presenting to key decision makers about your successful proof of concept, try to extrapolate financial success to business return on investment. Here are some examples:

  - Our proof of concept demonstrated that we can save USD X per month in Azure spending due to the optimisation in the new implementation. We recommend implementing this concept in production.
  - Customers have been asking for this new functionality, which has proven to be successful in the proof of concept. If we implement this concept in Azure, we estimate that revenue will grow by X% per quarter.

If your proof of concept failed

- You have two decisions:
  - Retry the proof of concept by redefining the goal, timeline and budget, scope and architecture.
  - Conduct a post-mortem review on the failure to see if there is any insights or lessons learned during the proof of concept.

In this section, you learned how to plan and execute the proof of concept. In the following section, we will showcase two practical projects, complete with step-by-step instructions. We hope that they will inspire you to come up with your own proof of concept projects.



Chapter 2: Sample project – implementing a web app using Azure Static Web Apps

In the first sample project, you will discover how to implement a web app using Azure Static Web Apps. You will learn about Azure App Service in Chapter 4: An overview of Azure for developers, but this project will cover one of the expanded hosting options of App Service: Static Web Apps. Developers can use Static Web Apps to render static contents (such as HTML, CSS and JavaScript) while delivering the necessary dynamic logic by developing serverless APIs with Azure Functions.

The Static Web Apps workflow (see Figure 3: Azure Static Web Apps workflow) closely resembles the daily workflow of a developer. Static Web Apps provide a managed Continuous Integration and Continuous Delivery (CI/CD) pipeline that automatically builds and deploys full stack web apps from a GitHub repository to Azure. This is made possible by GitHub Actions (to learn more about GitHub Actions, visit this documentation).

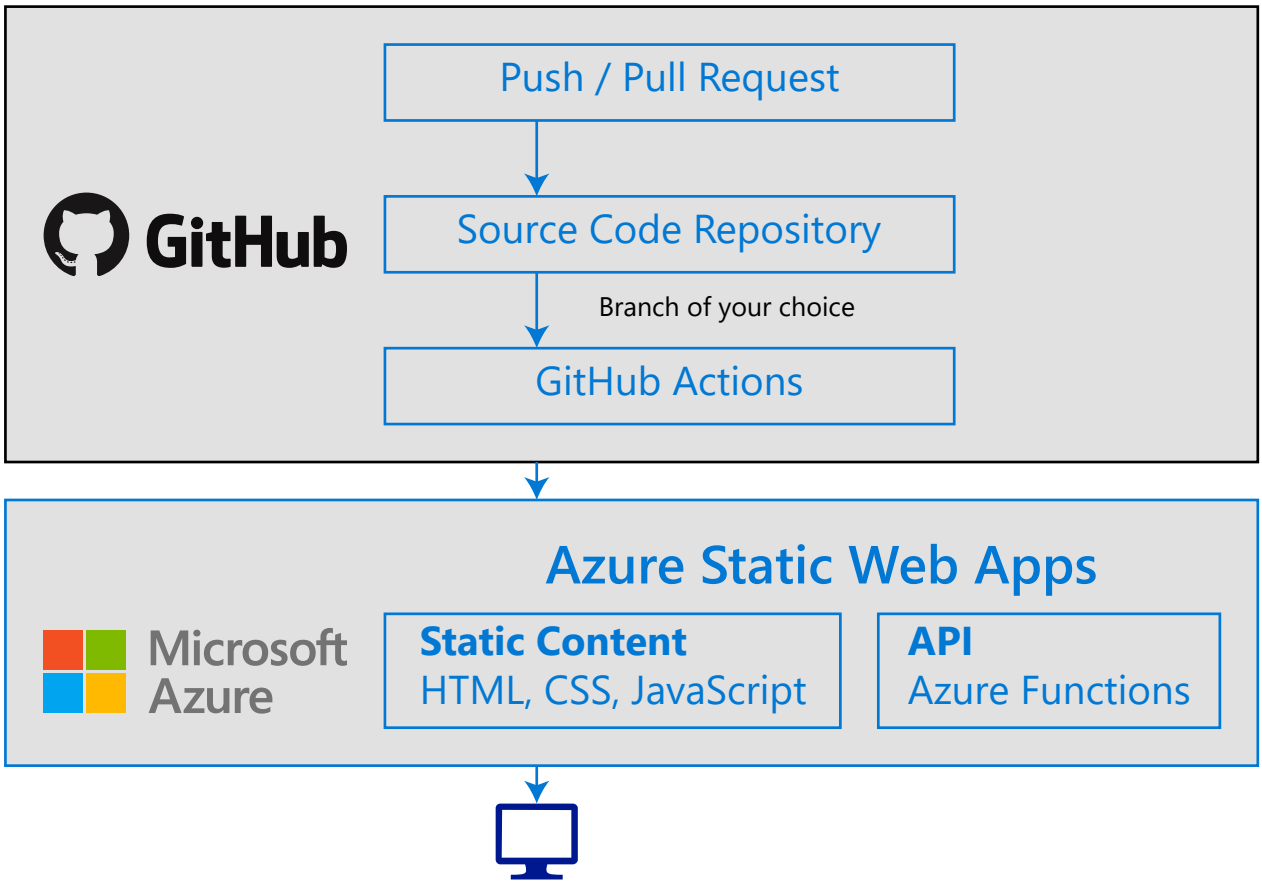


Figure 3: Azure Static Web Apps workflow

When you create a Static Web Apps resource, Azure sets up a GitHub Actions workflow in the app's source code repository, which monitors a branch of your choice. Each time you push commits or accept pull requests into the watched branch, GitHub Actions automatically builds and deploys your app and its API to Azure.

Azure Static Web Apps are commonly built using libraries and frameworks such as Angular, React, Svelte or Vue. These apps include HTML, CSS, JavaScript and image assets that make up the application.

Azure Static Web Apps versus traditional web server

With a traditional web server, assets are served from a single server alongside any required API endpoints.

With Static Web Apps, static assets are separated from a traditional web server and are instead served from points geographically distributed around the world. This distribution makes serving files much faster, as files are physically closer to end users.

In addition, API endpoints are hosted using a serverless architecture, which reduces the necessity of a full backend server.

Key features of Azure Static Web Apps

- Web hosting for static content such as HTML, CSS, JavaScript and images
- Integrated API support provided by Azure Functions
- First-party GitHub integration where repository changes trigger builds and deployments
- Automatically create staging environments to test code updates before rolling them out to production
- Globally distributed static content, putting content closer to your users
- Free SSL certificates, which are automatically renewed
- Custom domains to provide branded customisations to your app
- Seamless security model with a reverse-proxy when calling APIs, which requires no CORS configuration
- Authentication provider integrations with Azure Active Directory, Facebook, Google, GitHub and Twitter
- Customisable authorisation role definition and assignments
- Backend routing rules enabling full control over the content and routes you serve

Use case of Azure Static Web Apps

- Build single-page applications with frameworks and libraries such as Angular, React, Svelte or Vue with an Azure Functions backend
- Publish static sites with frameworks such as Gatsby, Hugo and VuePress
- Deploy web applications with frameworks such as Next.js and Nuxt.js

Putting it into practice

Note: As of the time of writing, Azure Static Web App is in preview and it is free of charge. This is subject to change once it is out of preview.

1. Sign in to your GitHub account and go to this URL to create a new repository.
2. Give your repository a name, such as my-first-static-web-app. Then click the **Create repository from template** button:

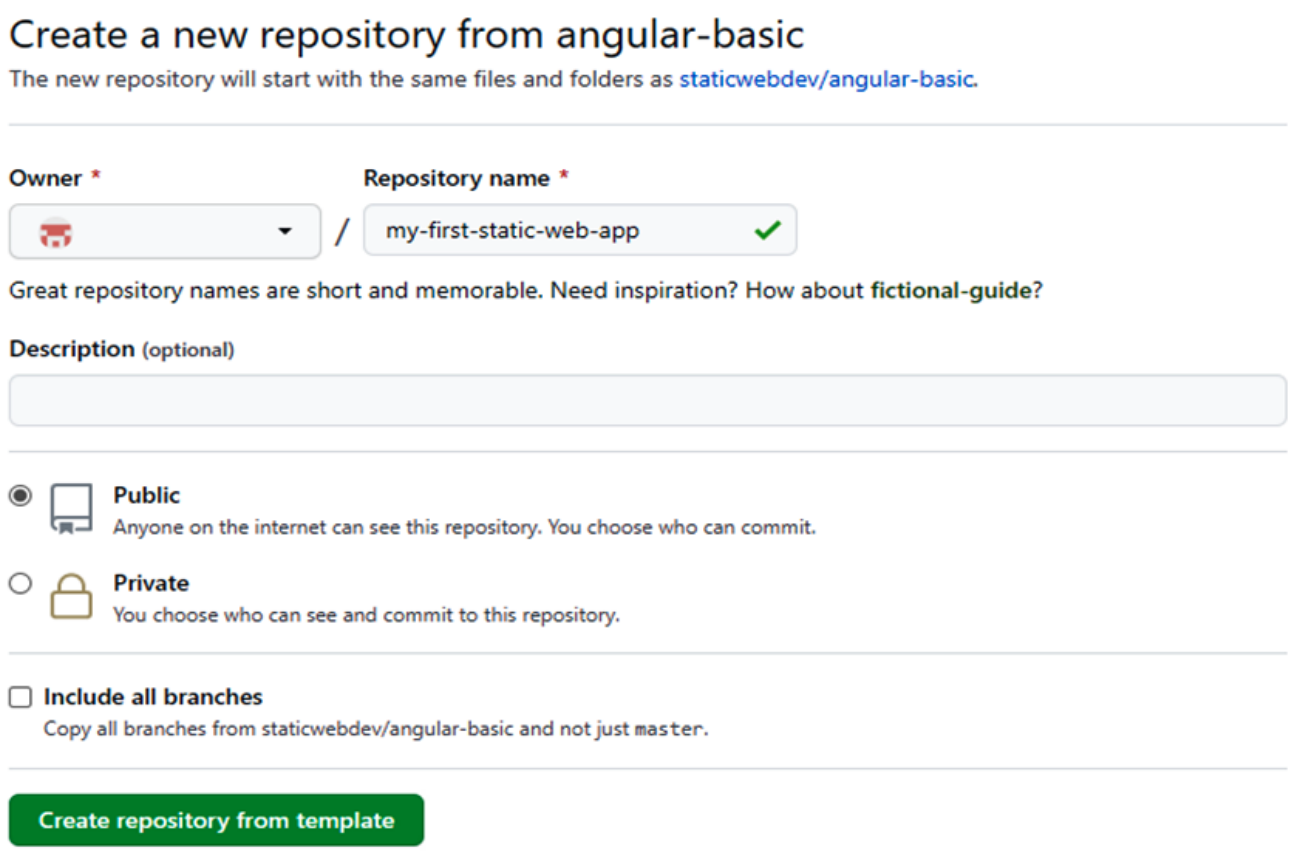


Figure 4: Create your GitHub repository from template

3. Next, go to the Azure portal.
4. Go to the Azure Marketplace by clicking the **Create a resource** button. Type *Static Web App* in the search box.

5. Click **Static Web App (preview)**, then click the **Create** button:

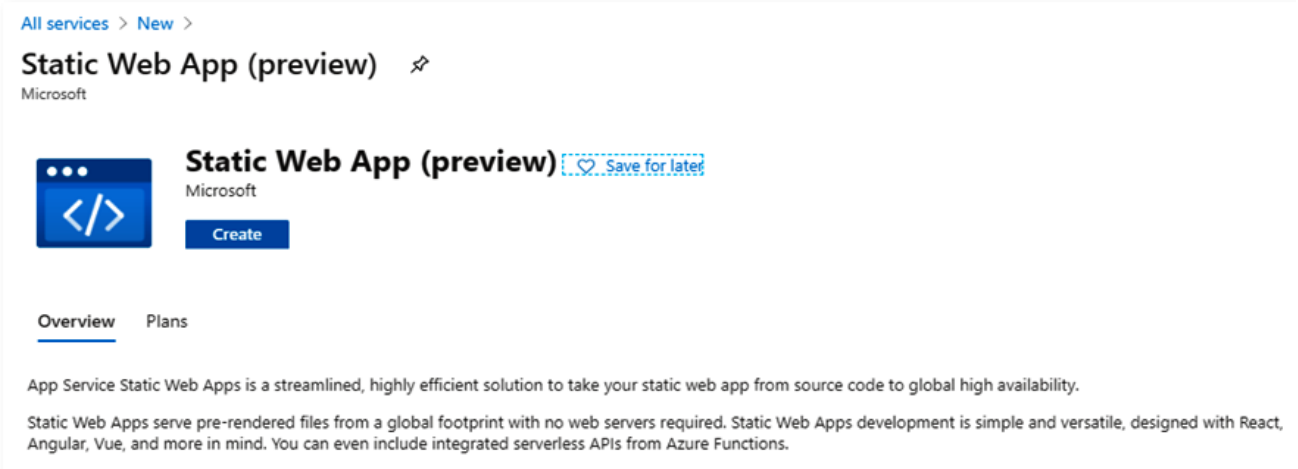


Figure 5: Create Static Web App

6. Fill in the form:

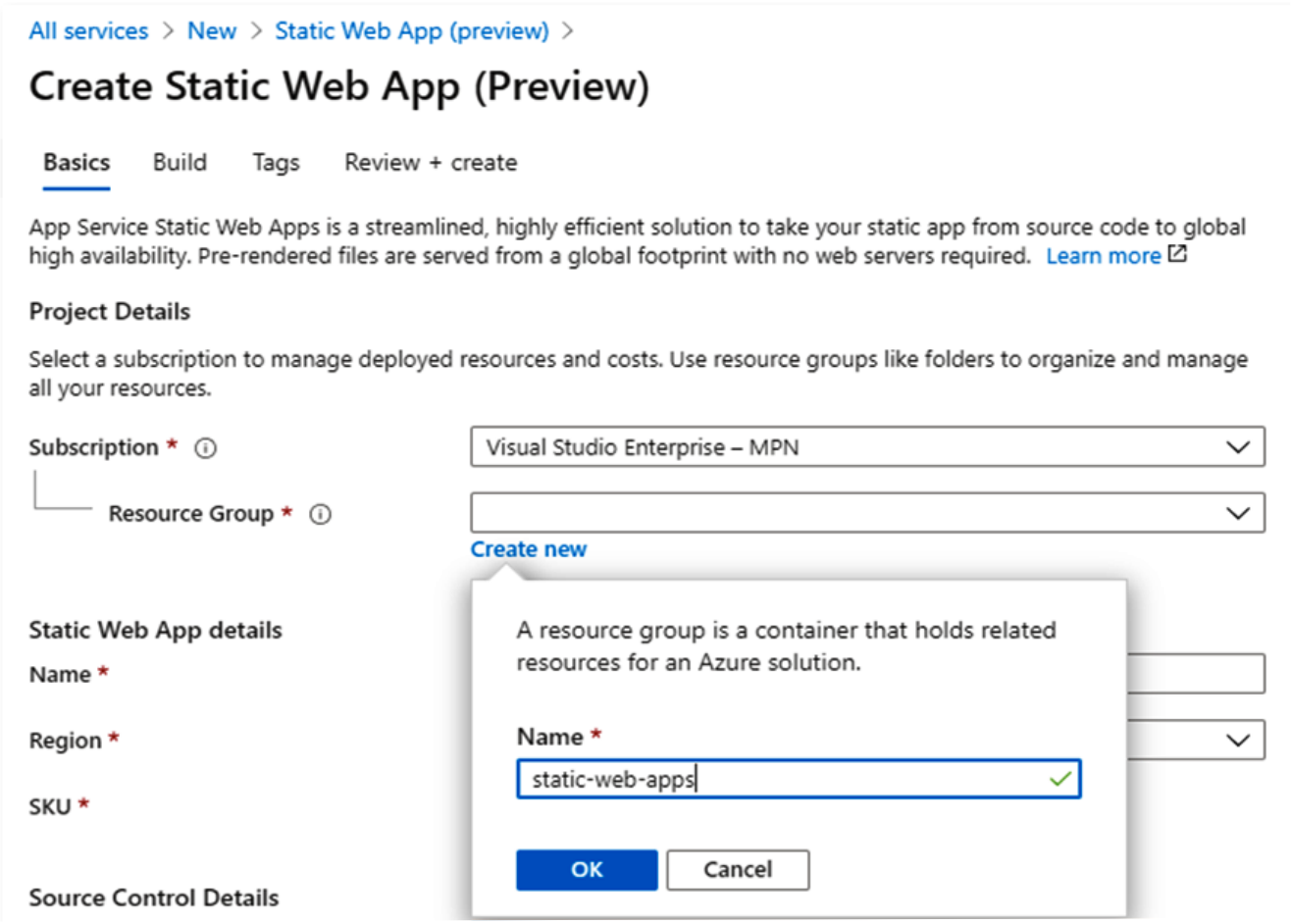


Figure 6: Create a new Resource Group

7. Pick a region where you want your Azure Static Web App to be hosted:

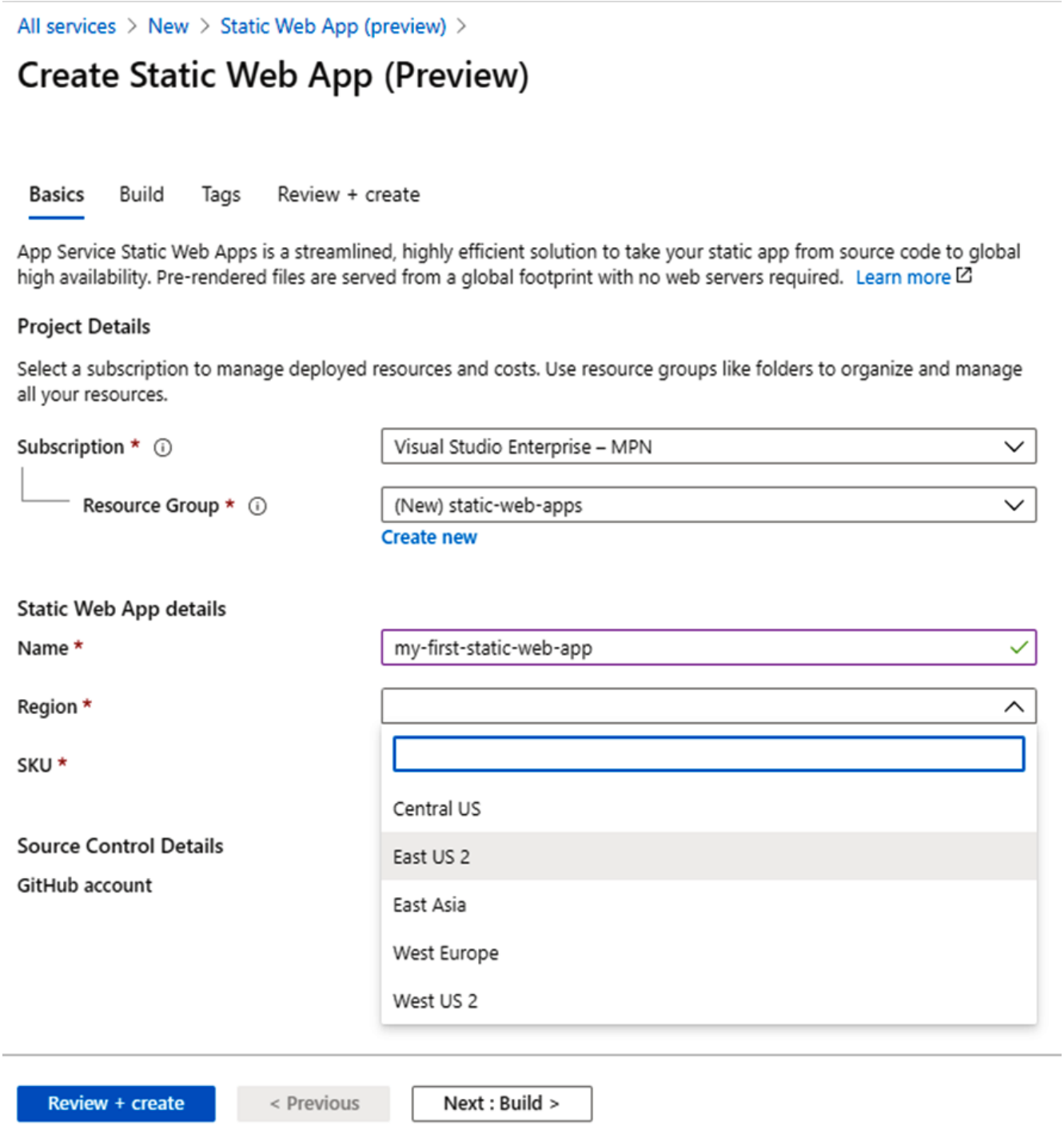


Figure 7: Pick a region

8. Connect to your GitHub account:

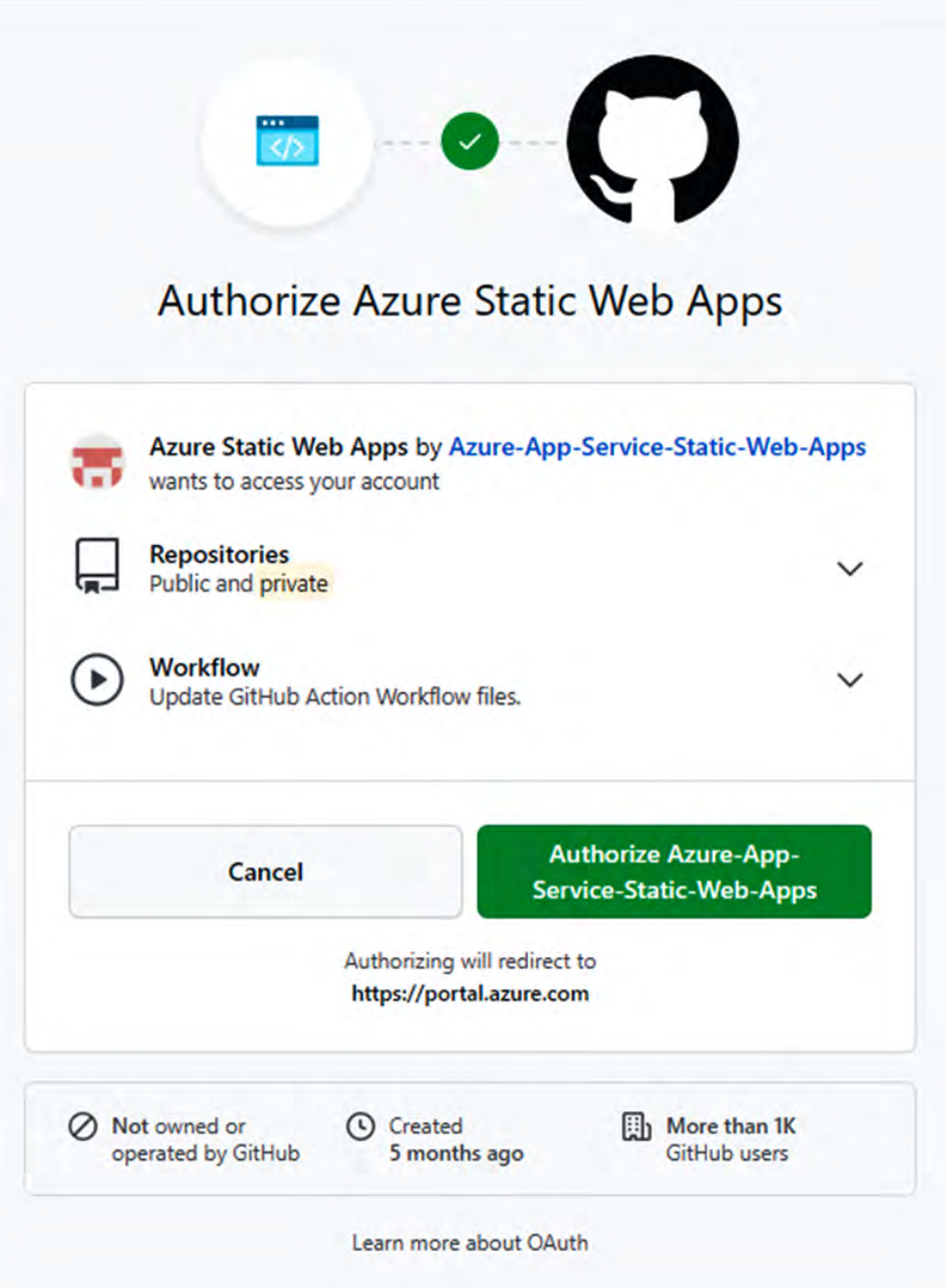


Figure 8: Connect to your GitHub account

9. In the GitHub account section, supply the **Organisation**, **Repository** and **Branch** information as shown:

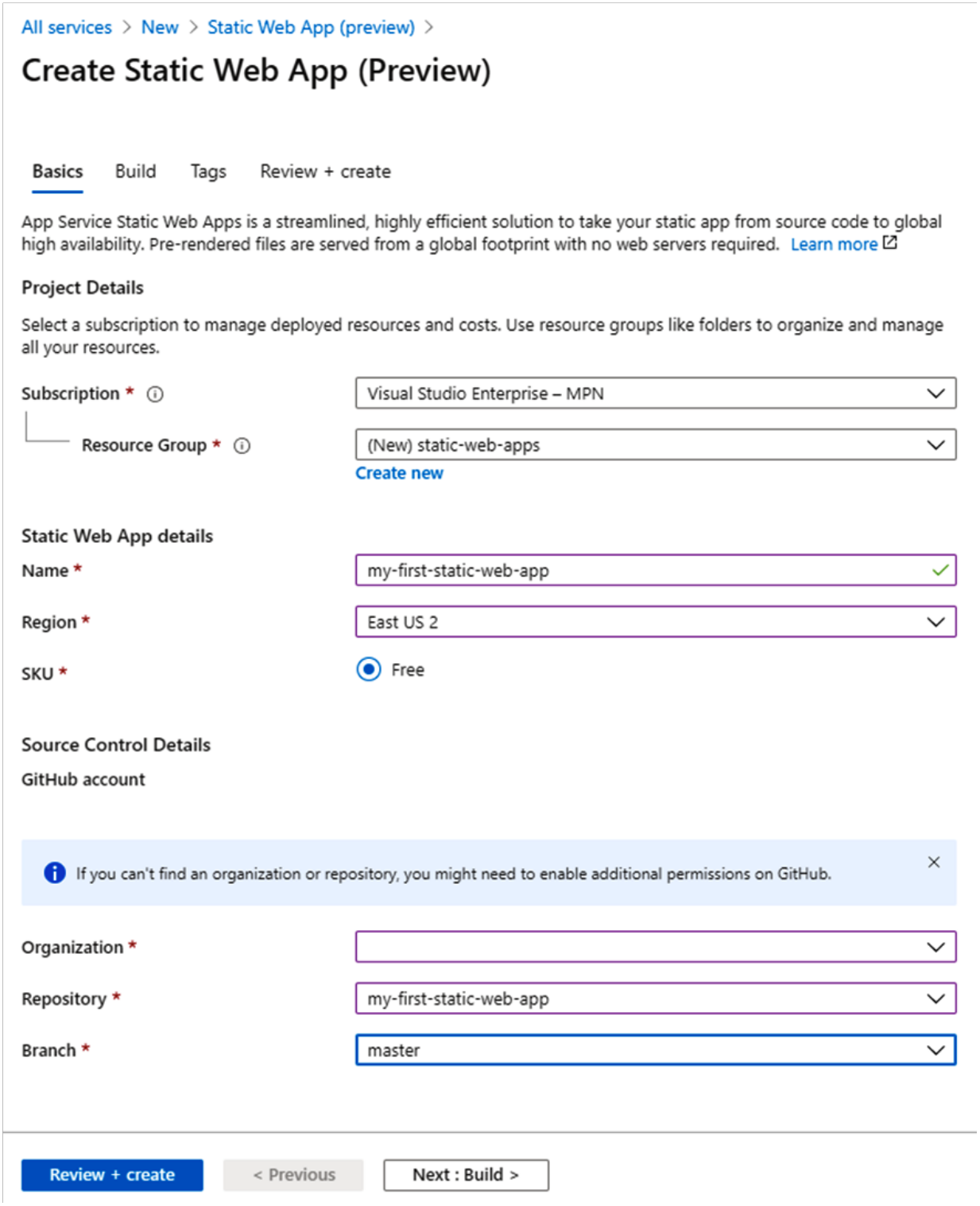


Figure 9: Supply the Organisation, Repository and Branch information

10. Provide the initial build variables as shown:

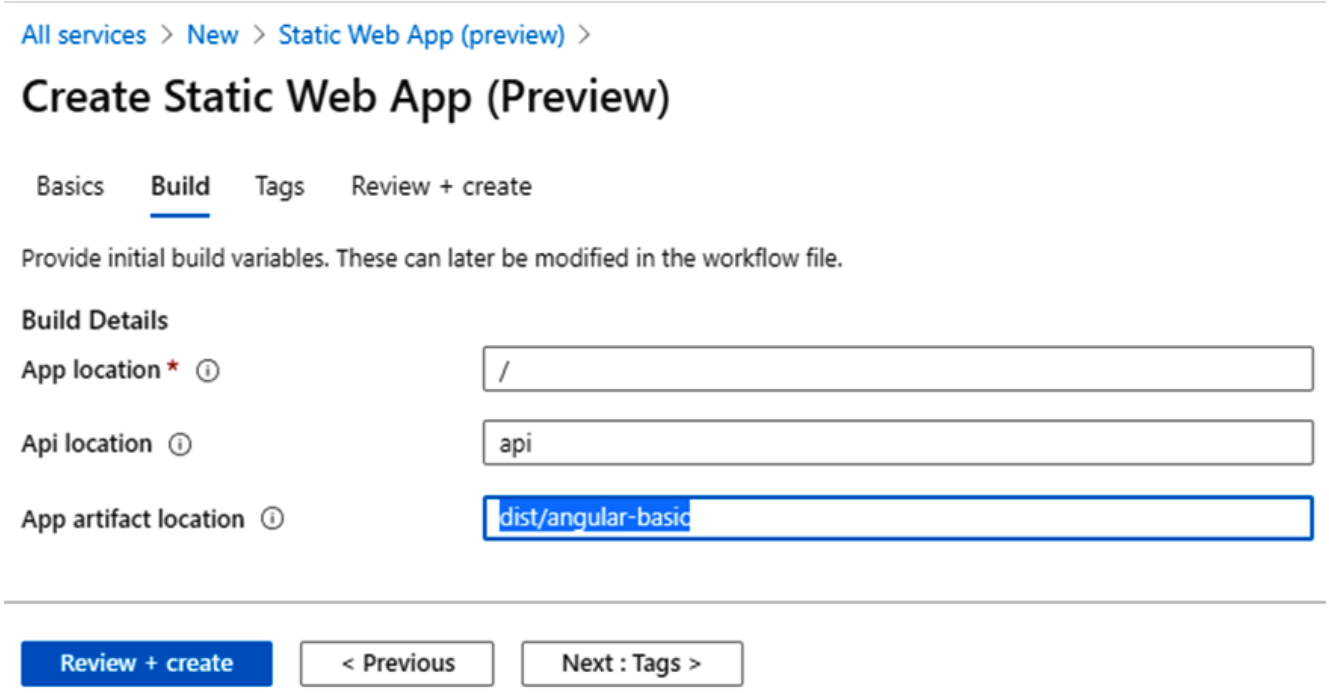


Figure 10: Provide the initial build variables

11. Navigate to the **Review + create** tab and click the **Create** button:

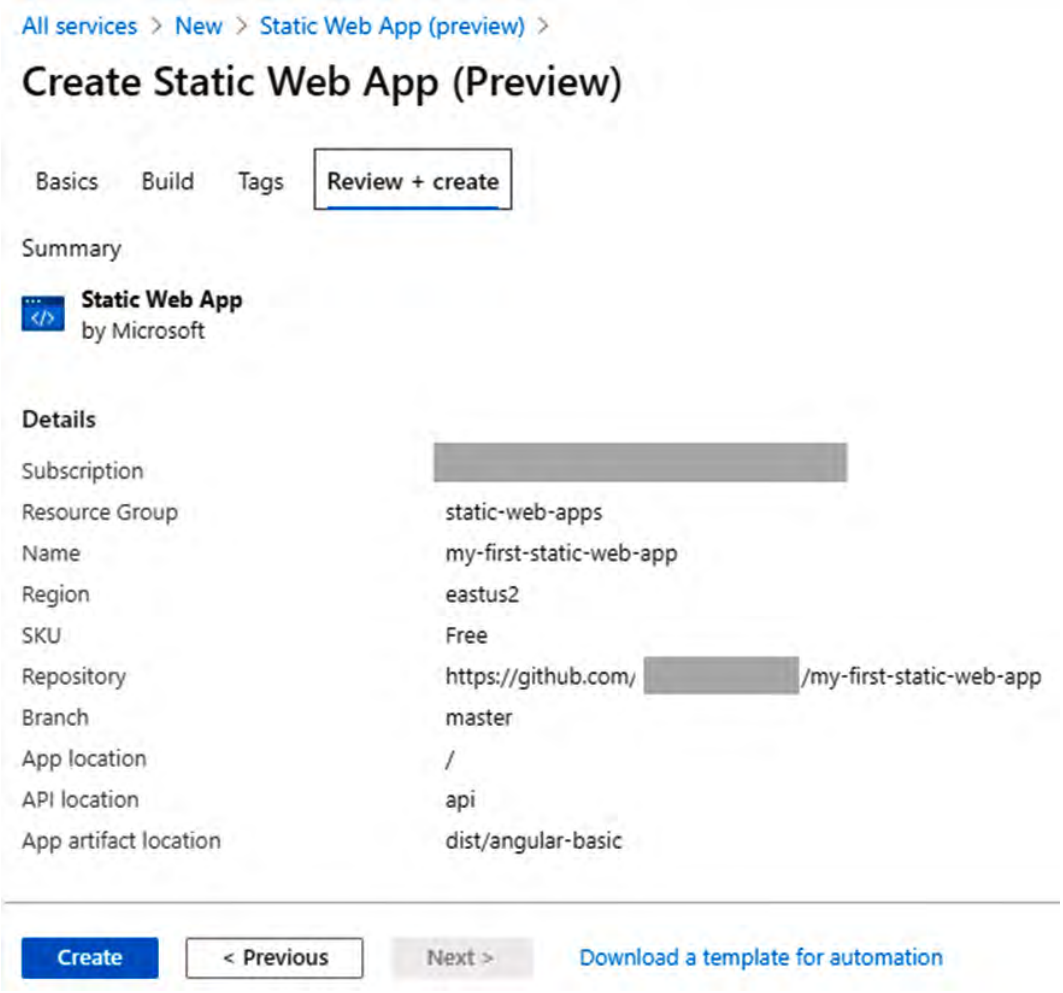


Figure 11: Review + create

12. Once your deployment is complete, click the **Go to resources** button to find the new URL that has been generated for your static web app:

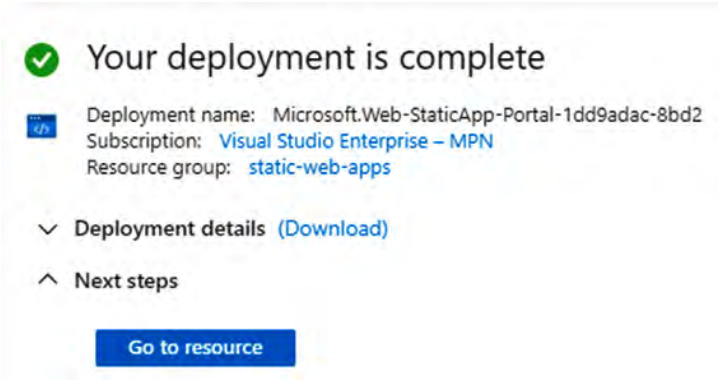


Figure 12: Deployment is complete

13. Find the new URL that has been generated for your static web app:

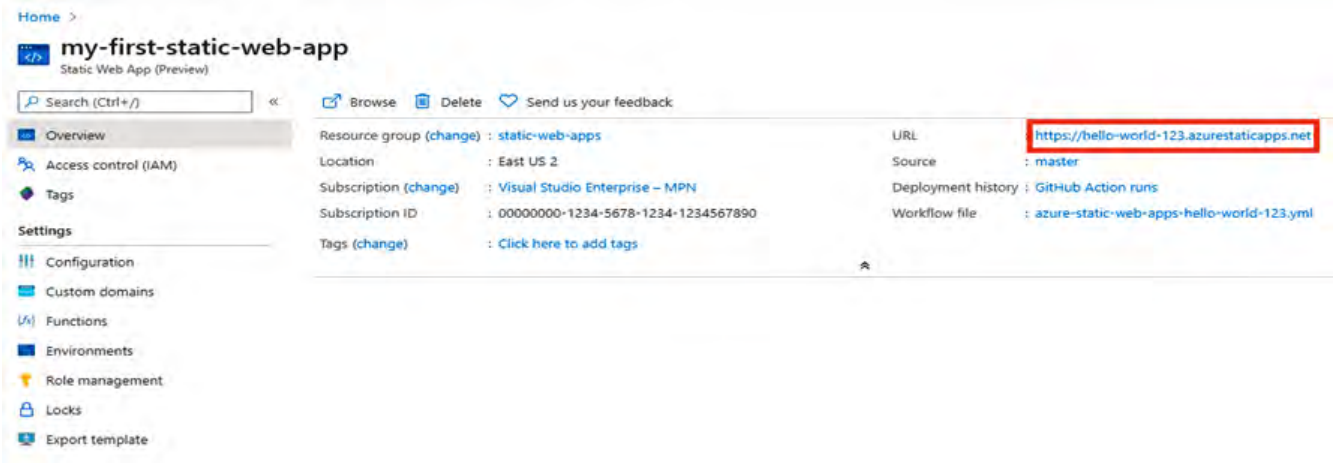


Figure 13: Static Web App URL

14. Open your browser and visit the generated URL to see your static web app in action:

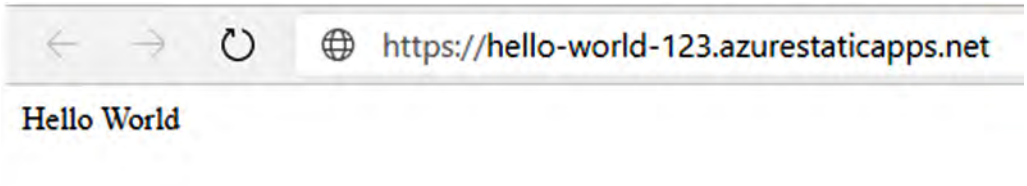


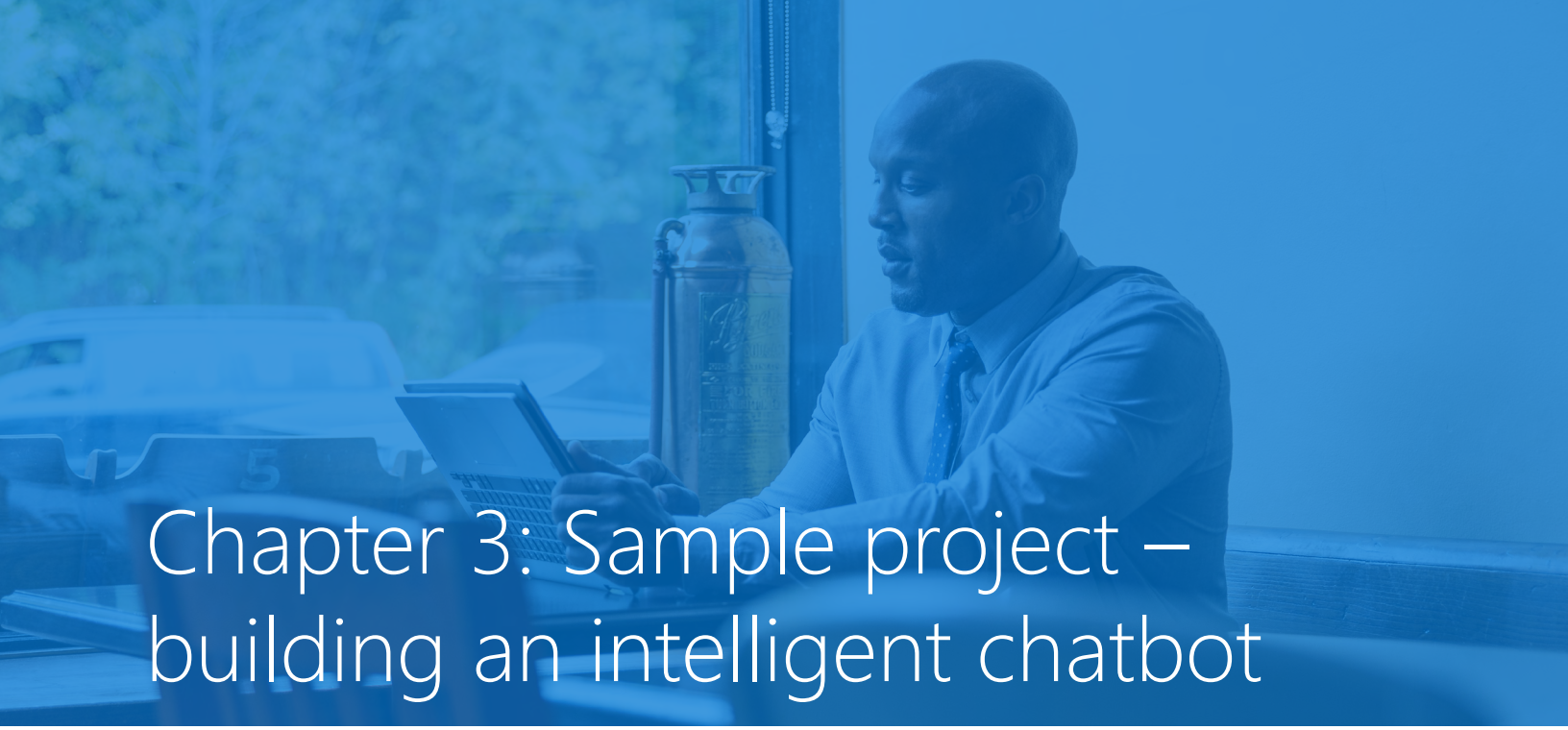
Figure 14: Static Web App in action

Congratulations, you have successfully deployed your first Azure Static Web App.

To learn more

- GitHub Actions official documentation
- Review pull requests in pre-production environments in Azure Static Web Apps

Next, we will explore a second sample project, building an intelligent chatbot.



Chapter 3: Sample project – building an intelligent chatbot

In this sample project, you will explore how to build an intelligent chatbot that uses artificial intelligence (AI). As AI continues to dominate in mainstream technology, the time is now for developers like you to harness the power of AI in your applications.

Today, many of us use a variety of technologies to communicate. For example:

- Phone calls
- Messaging services
- Online chat applications
- Email
- Social media platforms
- Collaborative tools

We have become accustomed to ubiquitous connectivity, and we expect the organisations we deal with to be easily contactable and immediately responsive through the channels we already use. Additionally, we expect these organisations to engage with us individually, and be able to answer complex questions at a personal level.

Key features of chatbots

While many organisations publish support information and answers to frequently asked questions (FAQs) that can be accessed through a web browser or dedicated app, answers to specific questions are difficult to find. These organisations frequently find their support staff being overburdened with requests for help through various channels, including phone calls, email, text messages and social media.

Many businesses are progressively turning to AI solutions that make use of AI agents (commonly known as chatbots) to provide a first line of automated support through the full range of channels that we use to communicate. Bots are designed to interact with users in a conversational manner, as shown in Figure 15: An example of a chatbot user interface:

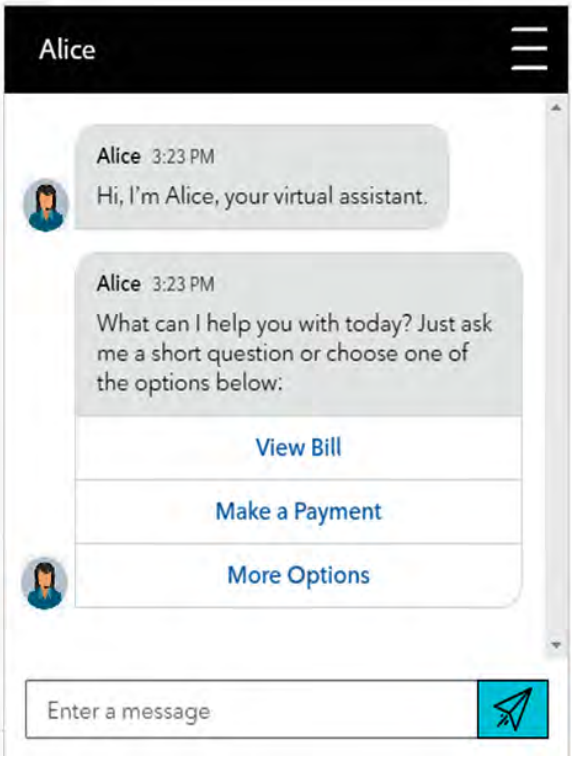


Figure 15: An example of a chatbot user interface

The example shown here is a chatbot interface like ones that you find on retail store websites. However, bots can be designed to work across multiple channels, including email, social media platforms and even voice calls. Regardless of the channel used, chatbots typically manage conversation flows using a combination of natural language and constrained option responses that guide the user to a resolution.

Use case for chatbots

Figure 16: Azure QnA Maker and Azure Bot Service, illustrates the two key components in creating an intelligent chatbot. The first component is a knowledge base of questions and answers. On Azure, this is QnA Maker. The second component is a bot service that provides an interface to the knowledge base. On Azure, this is Azure Bot Service. Using QnA Maker and Azure Bot Service, you can build a chatbot that provides users with answers to FAQs. The interface of the bot can be a chat section on your website.

Typically, conversations take the form of messages exchanged in turns. One of the most common kinds of conversational exchange is a question followed by an answer. This pattern forms the basis for many user support bots and can often be based on existing FAQ documentation.

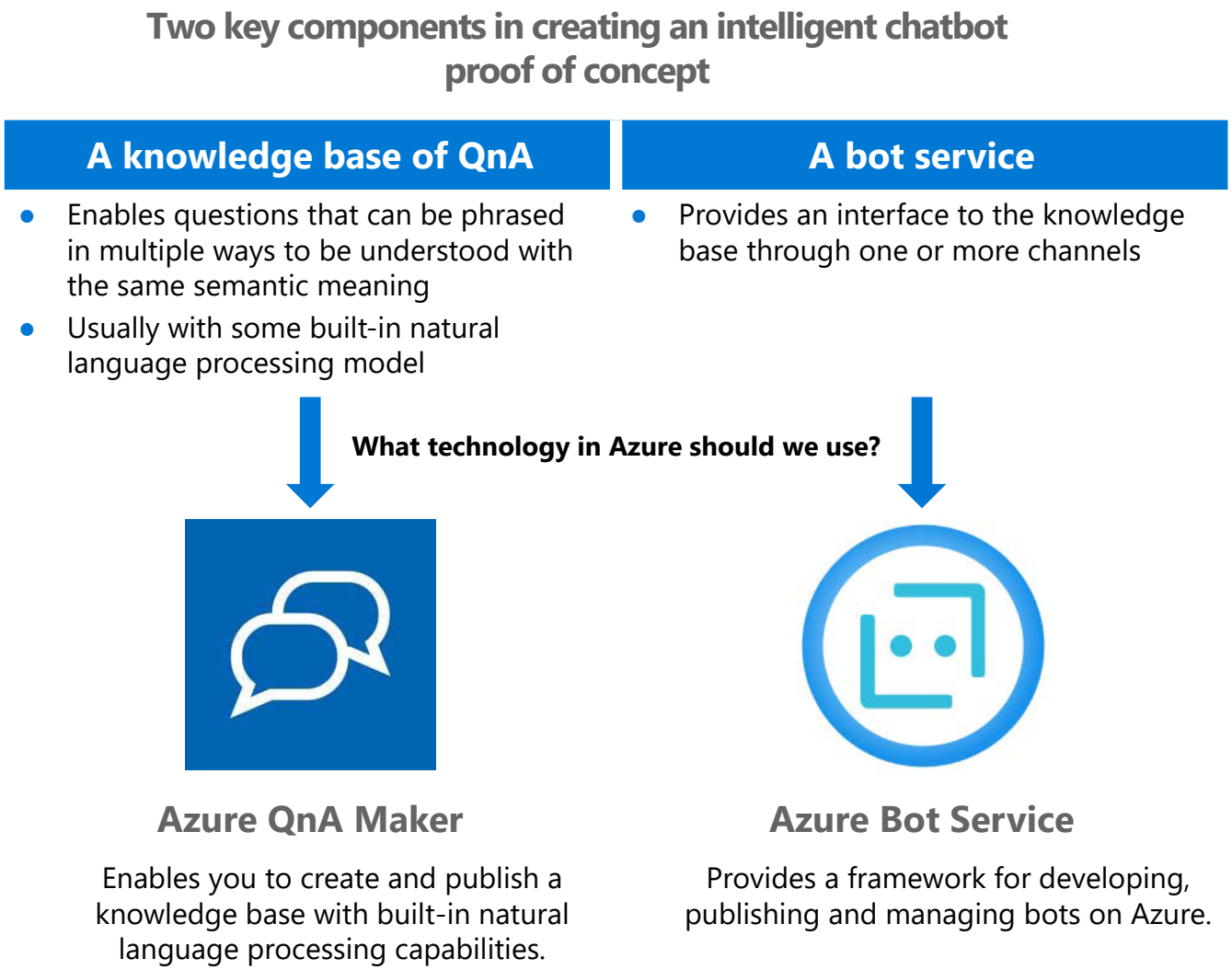


Figure 16: Azure QnA Maker and Azure Bot Service

Putting it into practice

1. Sign in to the QnA Maker portal using your Azure credentials (if you don't have an Azure subscription yet, review the Azure subscription section in Chapter 4: An overview of Azure for developers).
2. Click **Create a knowledge base** and then the **Create a QnA service** button to create a new QnA Maker resource:

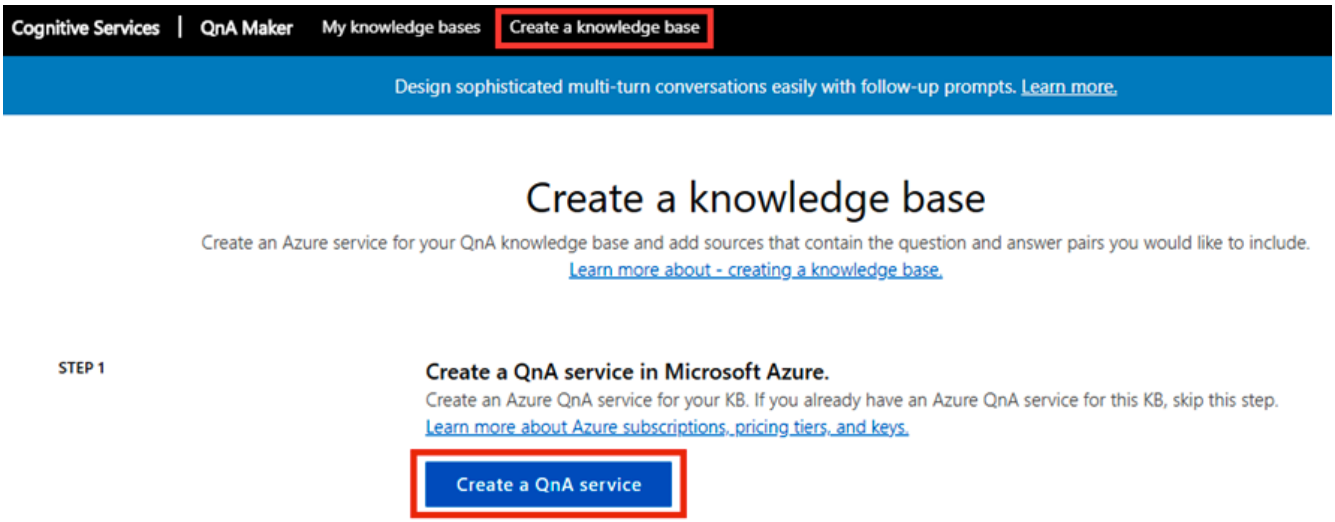


Figure 17: Create a new QnA Maker resource

3. Once the Azure portal is launched, populate the form as follows, then click the **Review + create** tab. The entry in the **Name** field must be globally unique. If you receive a name conflict error, just try again with another unique name:

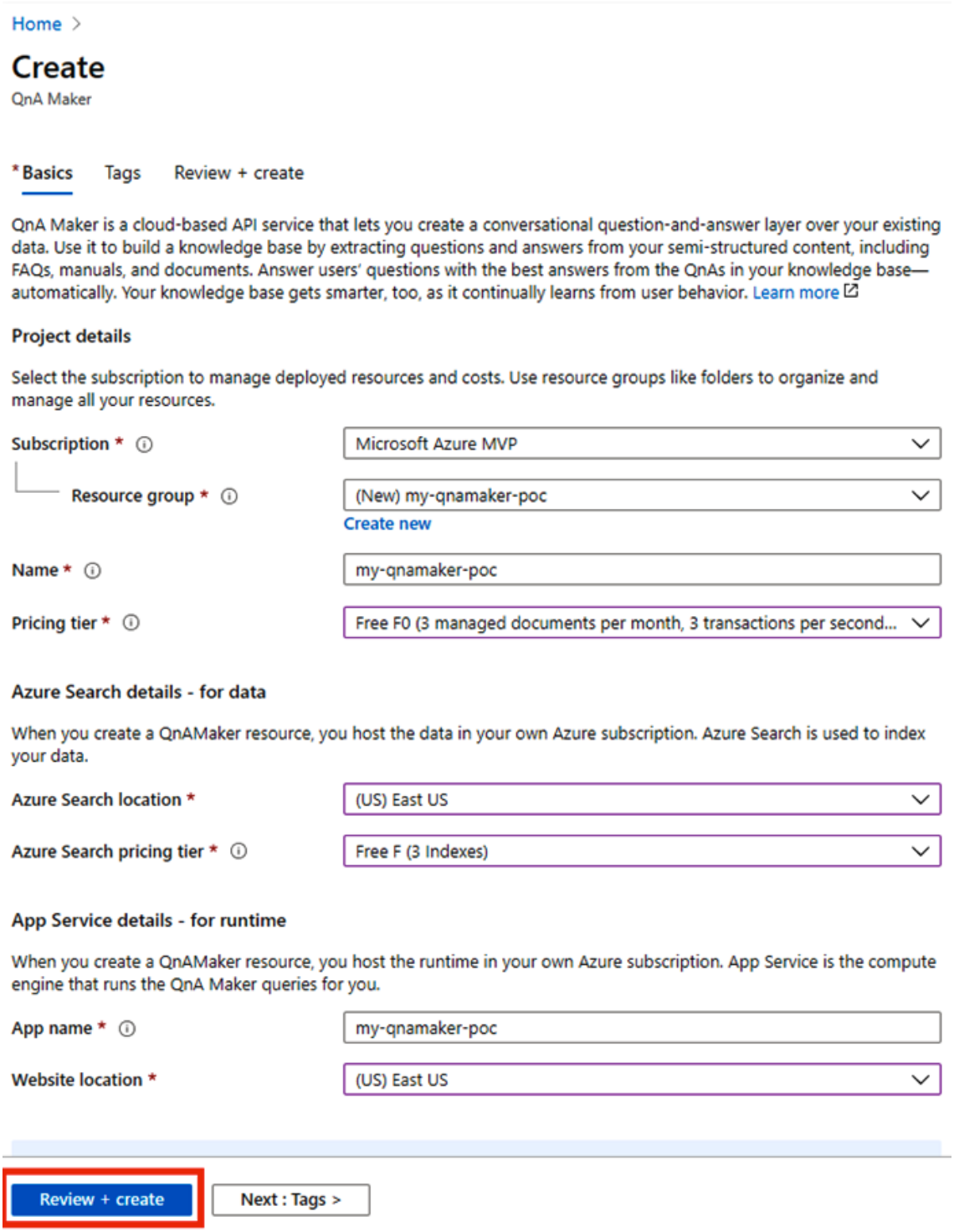


Figure 18: Add details in the Basics tab

4. After your QnA Maker resource has been provisioned, go back to Create a knowledge base, refresh the page and continue to Step 2:

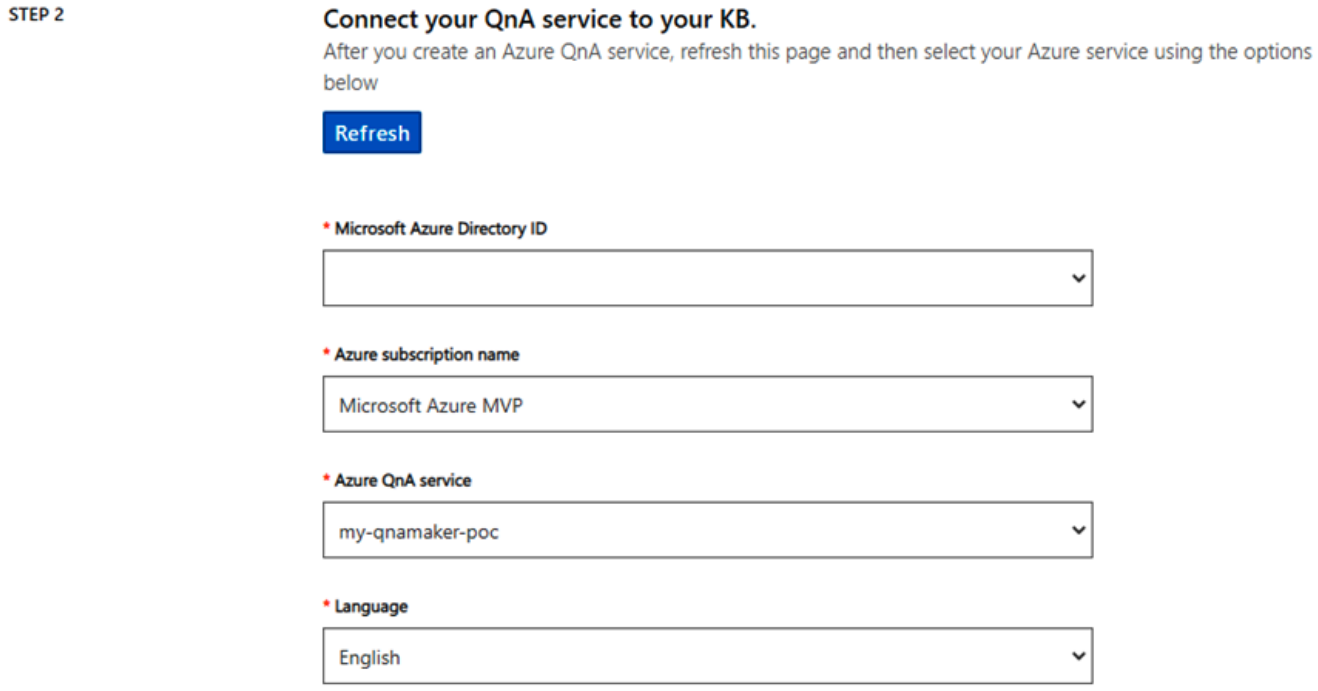


Figure 19: Step 2 of QnA Maker – Connecting QnA service to your KB

5. In Step 3, name your knowledge base:

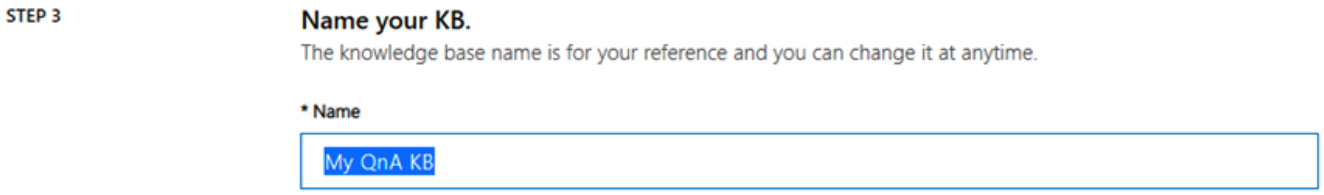


Figure 20: Step 3 of QnA Maker – Name your KB

6. Populate Step 4 as follows:

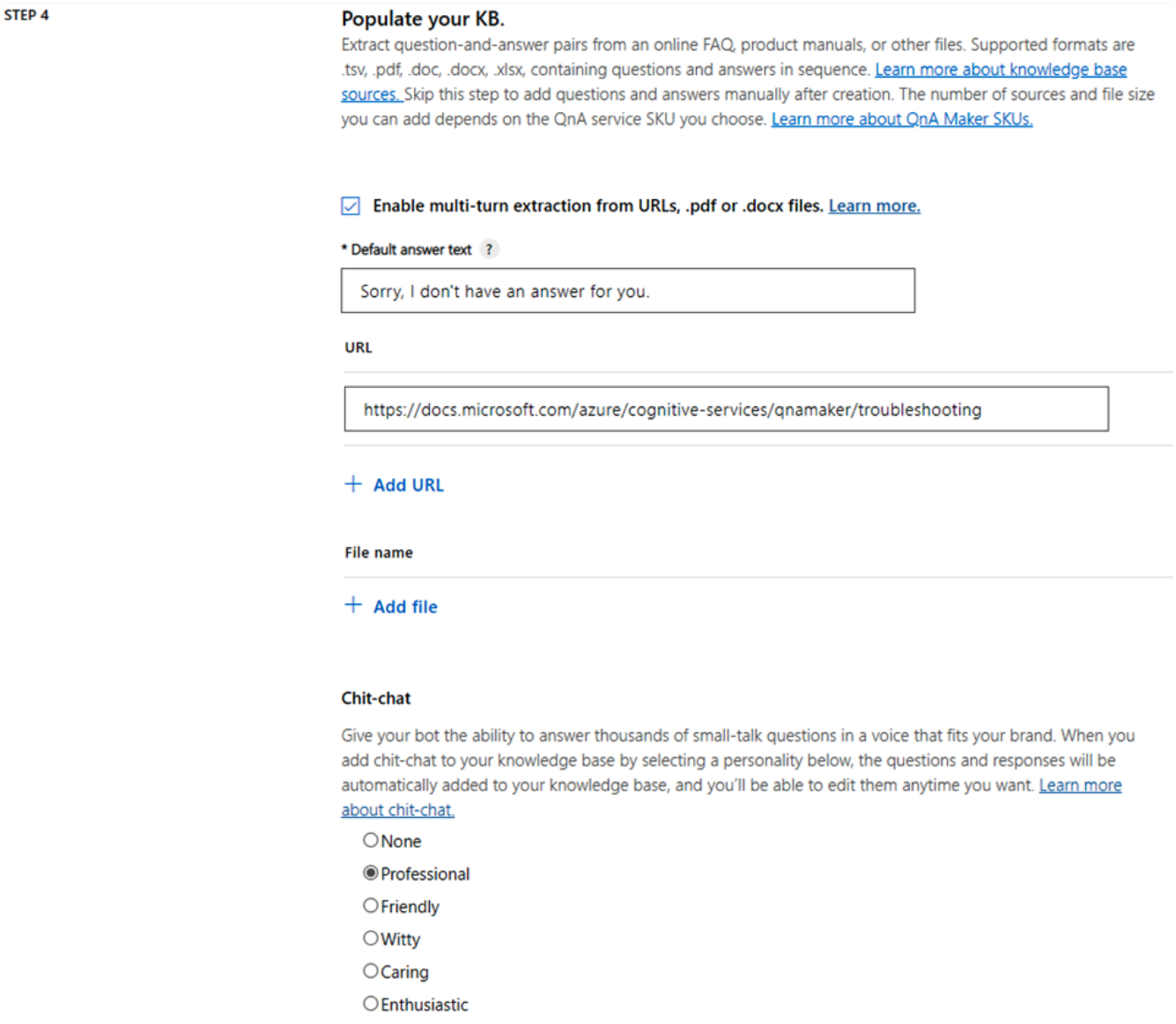


Figure 21: Step 4 of QnA Maker – Populate your KB

7. Click the **Create your KB** button in Step 5:

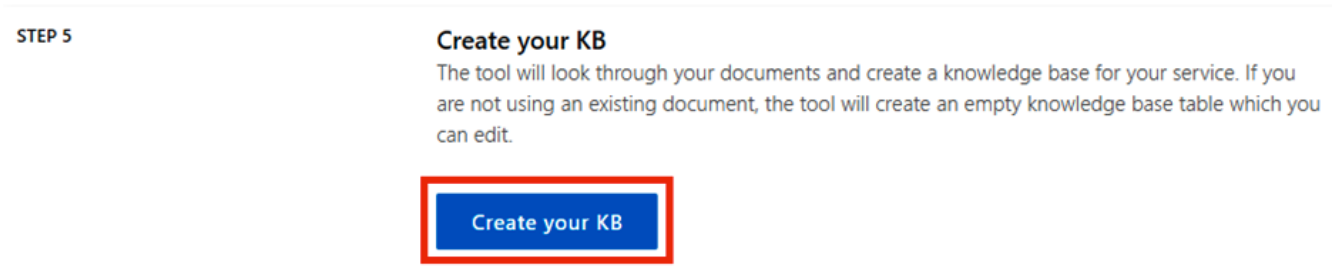


Figure 22: Step 5 of QnA Maker – Create your KB

8. In the QnA Maker portal, on the **Edit** page, select **+ Add QnA pair** from the toolbar:

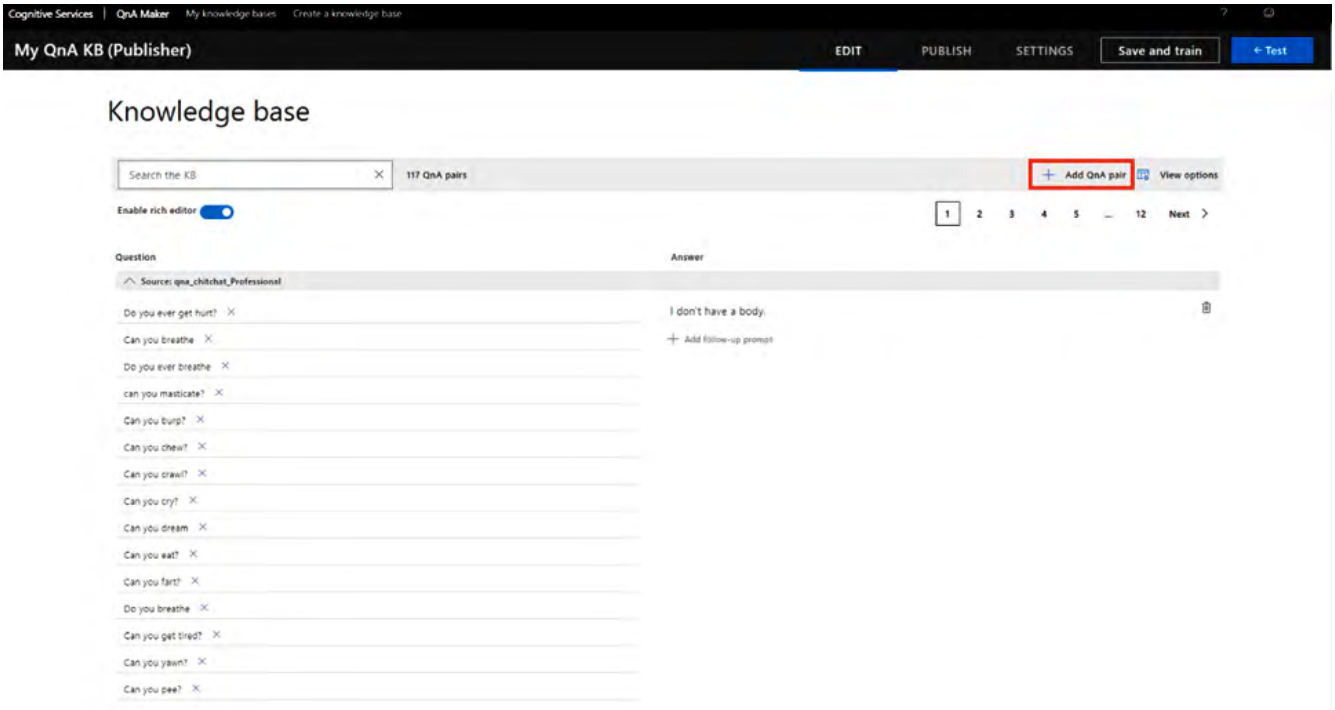


Figure 23: Add a QnA pair

9. Add the question and answer. Then click the **Save and train** button:

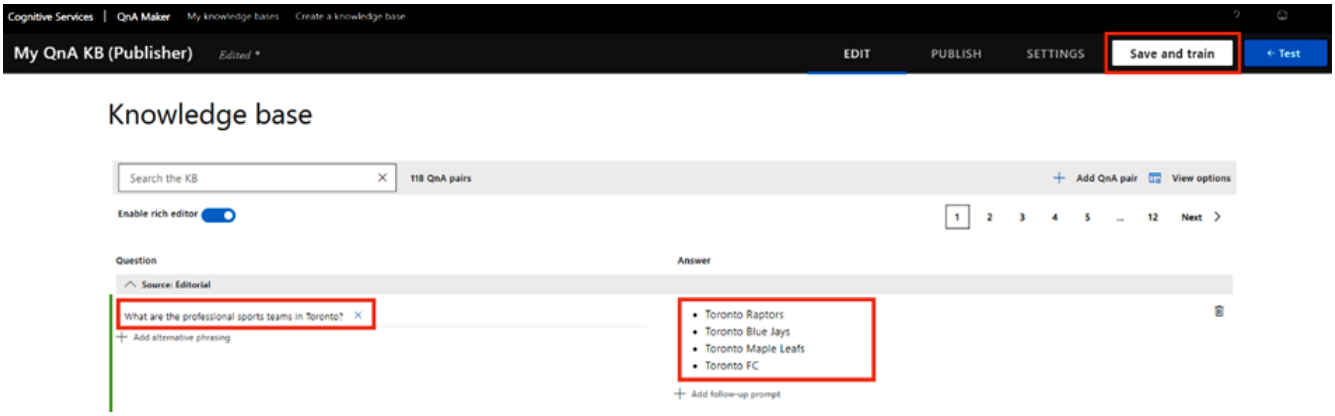


Figure 24: Add questions and answers

10. You can test your knowledge base right away by pressing the **Test** button. Then, enter a question such as, "*What are the professional sports teams in Toronto?*":

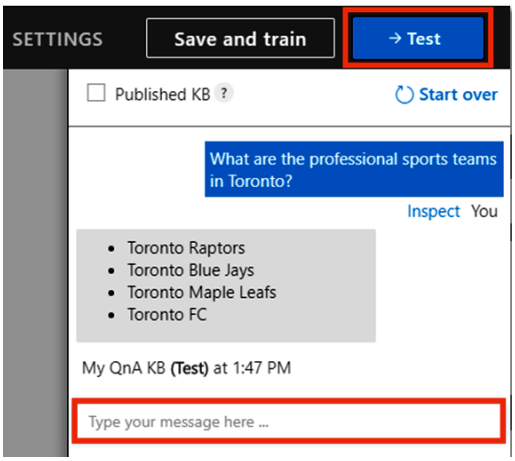


Figure 25: Test your knowledge base

11. Press **Inspect** to examine the details of the conversation:

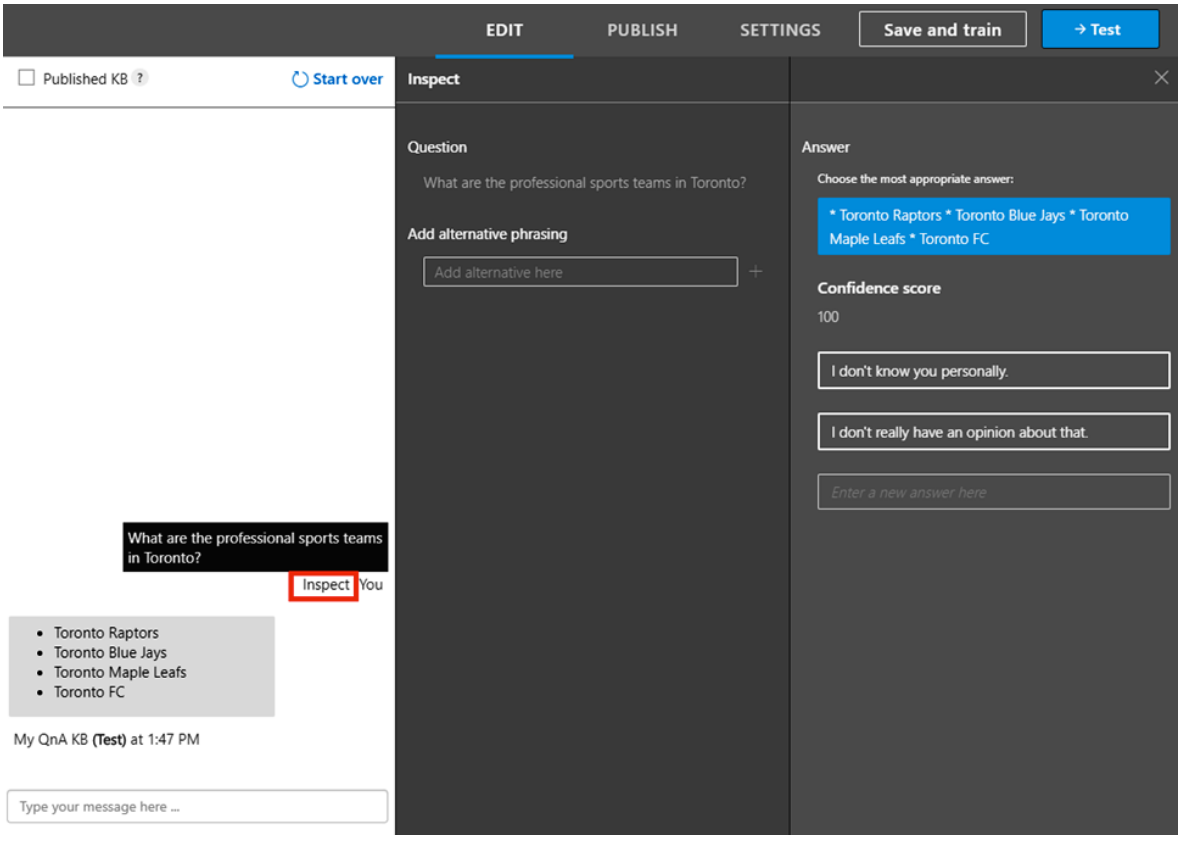


Figure 26: Examine the details of the conversation

12. Click the **Test** button to close the test panel.

Publishing your QnA knowledge base

When you publish your knowledge base, its contents move from the test index to a production index in Azure Search.

In the QnA Maker portal, click the **Publish** button:

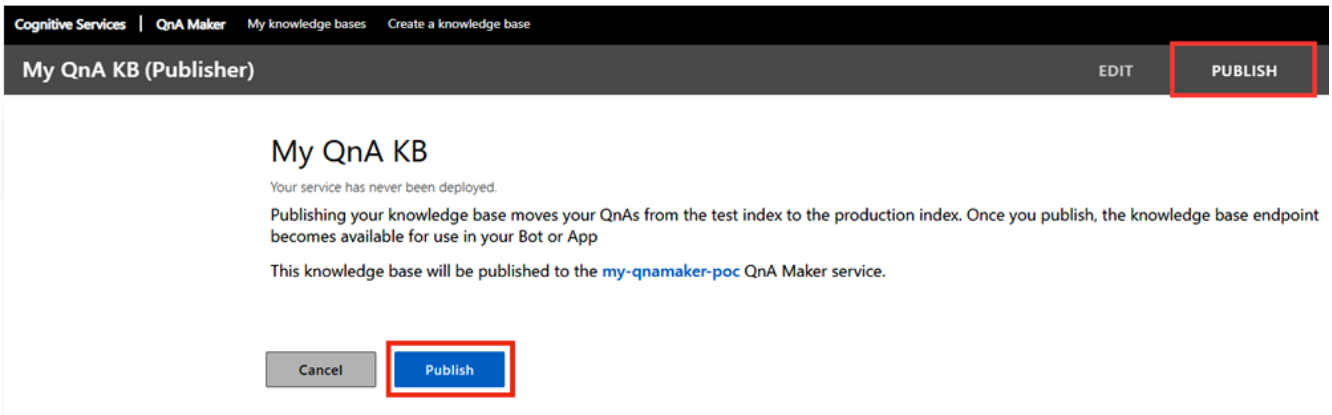


Figure 27: Publish your QnA KB

Create the bot in Azure Bot Service

Next, you will create a bot in Azure Bot Service to bind to the knowledge base you created in the previous steps:

1. After your knowledge base has been successfully deployed, click the **Create Bot** button to launch the Azure Bot Service creation page in the Azure portal:

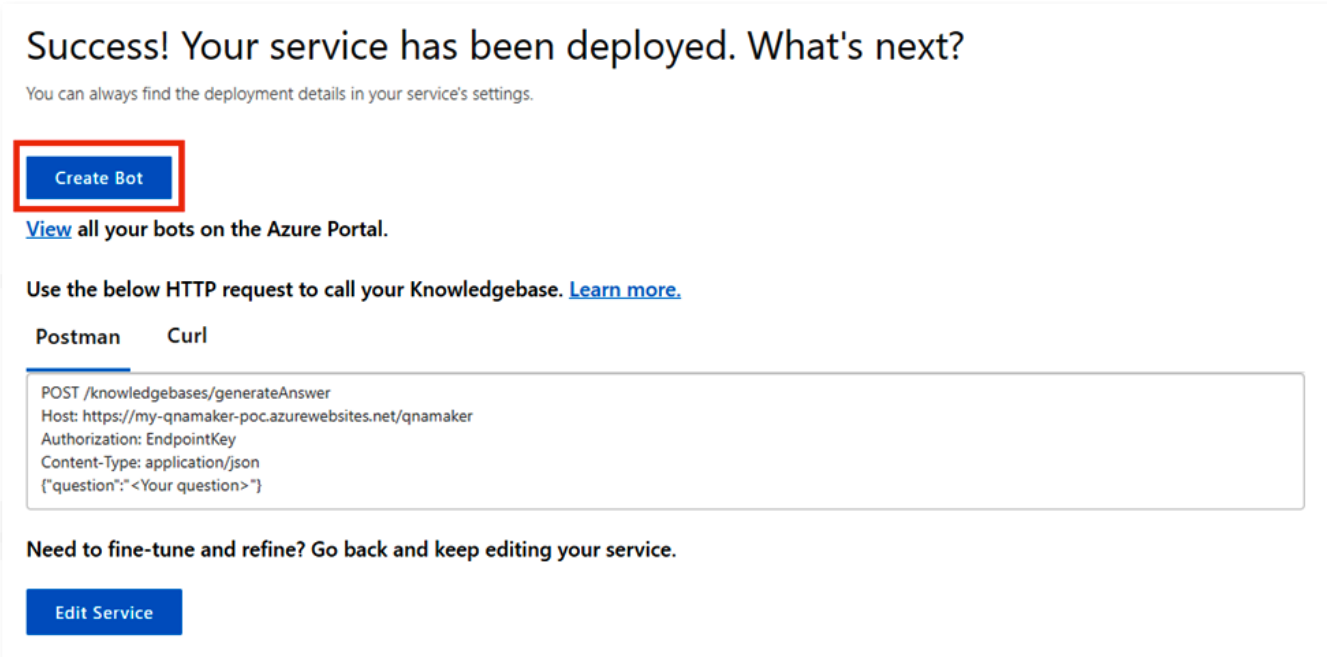


Figure 28: Launch the Azure Bot Service creation page

2. On the Azure Bot Service creation page, populate the form as follows:

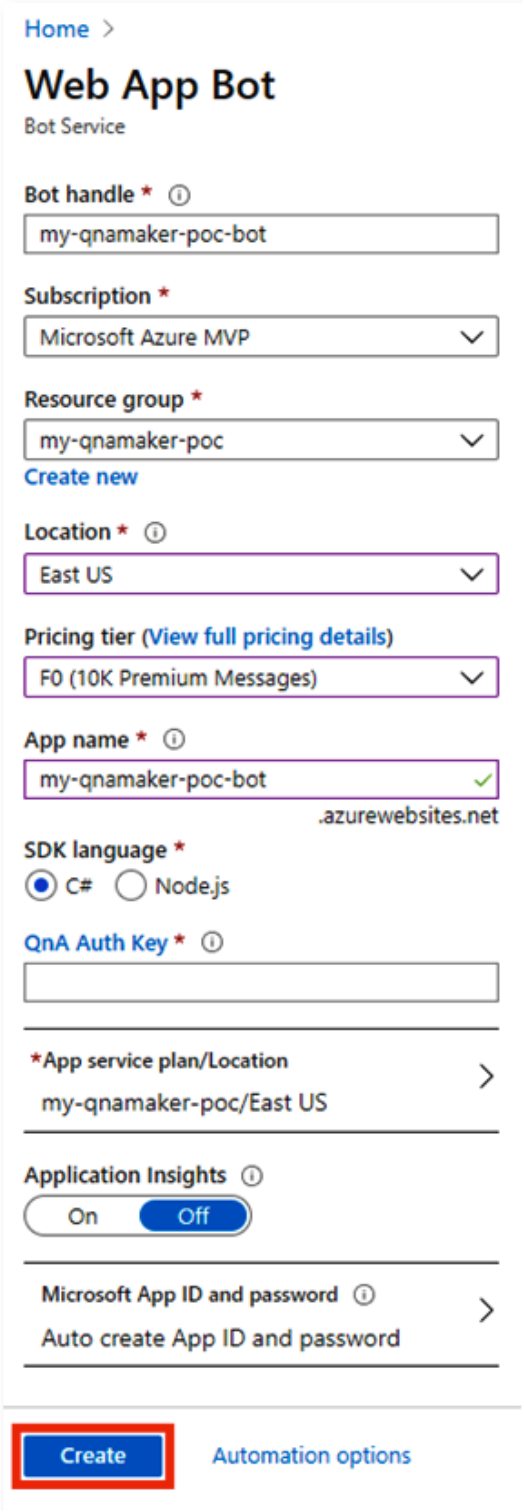


Figure 29: Add Web App Bot details

3. Once the bot has been provisioned, open it from Bot Services:

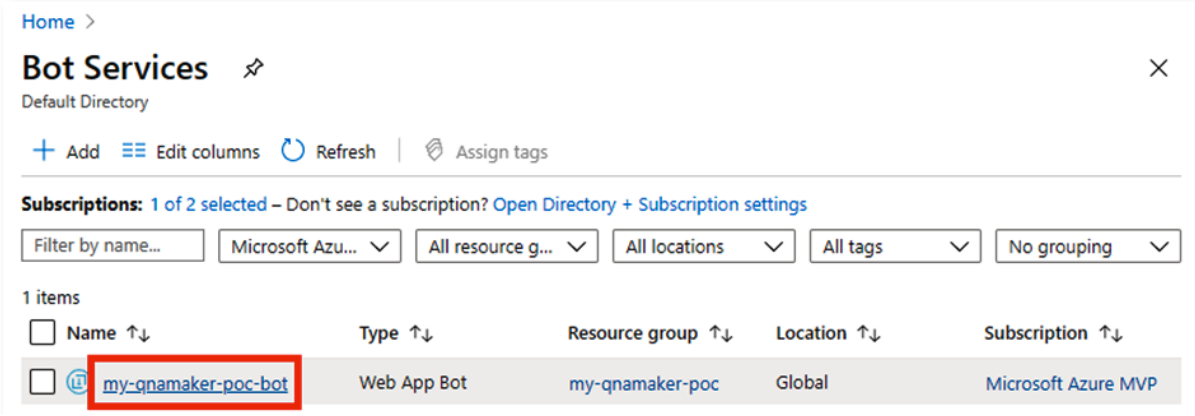


Figure 30: Open bot from Bot Services

4. Under **Bot management**, select **Test in Web Chat**:

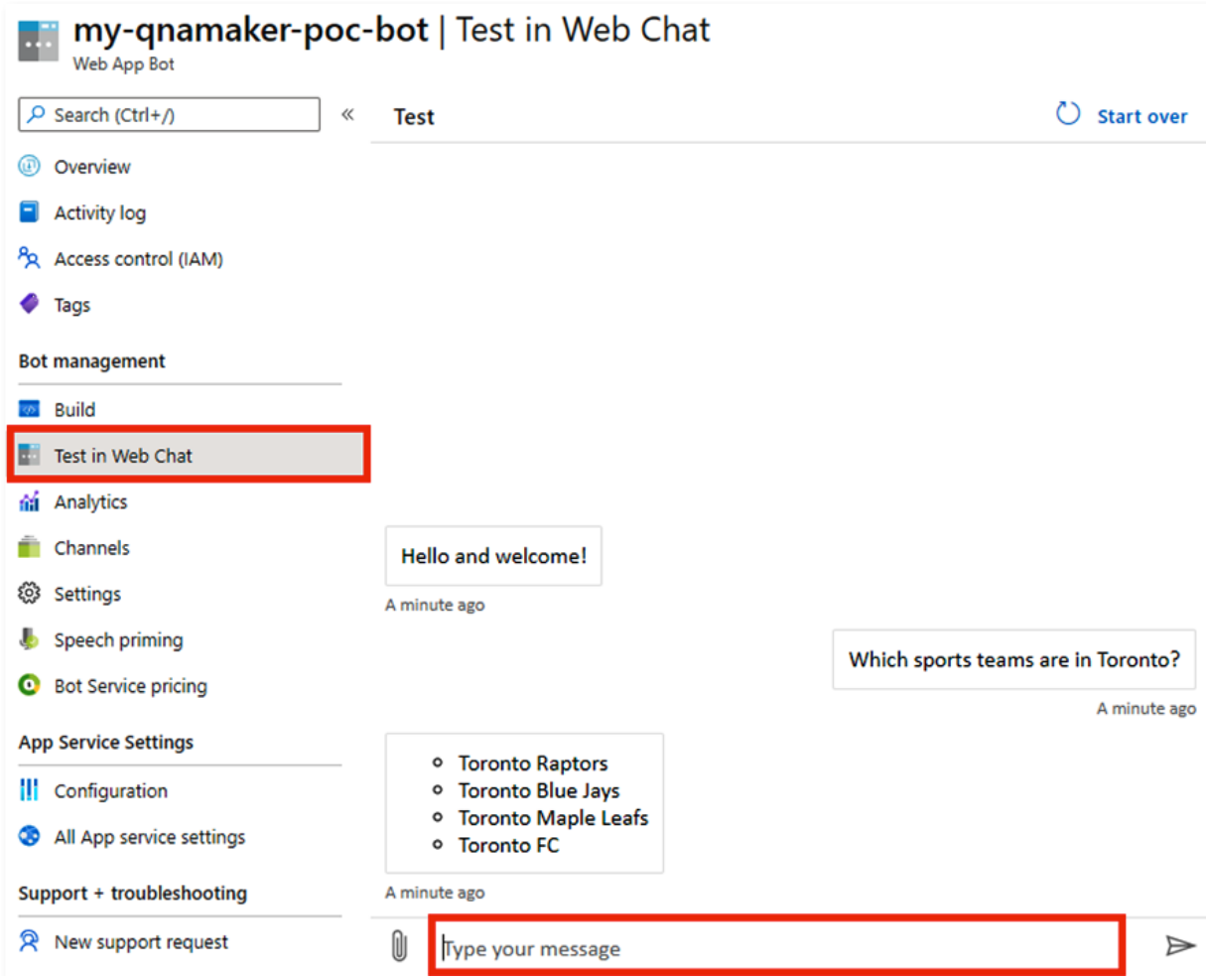


Figure 31: Select Test in Web Chat

In summary, you used the QnA Maker to create a new knowledge base in Azure. You added a public URL to the knowledge base. Then, you added your own QnA pair, trained and tested. After publishing your knowledge base, you created a Web App Bot in Azure Bot Services. Finally, you tested the bot running in Azure. Bots can help reduce support costs by providing automated support through multiple communication channels. We have shown you how to use QnA Maker and Azure Bot Services to create a chatbot that answers user support questions.

This concludes our tour of two practical projects. We hope they inspire you to come up with your own proof of concept projects. In the next section, we will explore an overview of Azure for developers, and learn how to pick the appropriate cloud model for your Azure proof of concept projects.

Chapter 4: An overview of Azure for developers

Getting started with Microsoft Azure

As discussed in the introduction, Azure is a cloud platform which empowers you to:

- Host your existing applications with minimum changes
- Develop new cloud-native applications

There is a huge number of ready-to-use Azure services that you can integrate with your applications to instantly take advantage of new cloud capabilities while minimising the need to develop those services yourself. In this section, we will discuss:

- The benefits of Azure
- How to get started with Azure
- Tools you need to develop your application for Azure
- How to choose the appropriate cloud model for your Azure proof of concept project

The benefits of Azure

The following are some key benefits of hosting your applications on Azure:

Lower costs

- By developing applications on Azure, your company will save on capital expenditures in the long run thanks to Azure's pay-as-you-go pricing model.
- You pay only for the services that you use.
- There's no need to carry capital expenses by purchasing on-premises servers.

Less maintenance

- With managed services, you can focus on developing your application.
- Azure will look after infrastructure maintenance for you.

Scalability

- Azure services are highly scalable to meet the growing demands of your business.

High availability and reliability

- You can configure your Azure services to ensure your applications are highly available and reliable.

Azure subscription

To start building applications to run on Azure, you will need an Azure subscription. If your company provides you with Azure credits, you are all set. Otherwise, simply sign up for an Azure free account.

What does the Azure free account include?

The Azure free account allows you to get started with 12 months of free services and USD $200 credit to explore Azure for 30 days. These offers may change over time. For the most up-to-date details on what is included in the Azure free account, please visit the Azure free account FAQ.

If you are new to Azure, there are plenty of free resources to help you get started with ease. You can find these resources in Chapter 5: Further learning and resources.

In the next section, you will look at tools that you need to develop your proof of concept project for Azure.

Tools that you need to develop your proof of concept project for Azure

The following is a list of tools that are essential for the examples shown in this guide:

- Azure subscription
- Visual Studio Code
- Azure Functions Core Tools
- GitHub account
- Microsoft Edge (Chromium-based) browser
- Node.js

In the next section, you will learn how to choose the appropriate cloud model for your Azure proof of concept project.

How to choose the appropriate cloud model for your Azure proof of concept project

In Chapter 1: Proof of concept guide, you learned about the proof of concept. Once you have your proof of concept planning in place, you can start your implementation and testing. In this section, we will show you the four cloud models (see Figure 32: Azure cloud models for Application Development and Hosting) that are available for you to choose for your Azure proof of concept project. We will explain the use case for each of these cloud models so that you can make an informed choice.

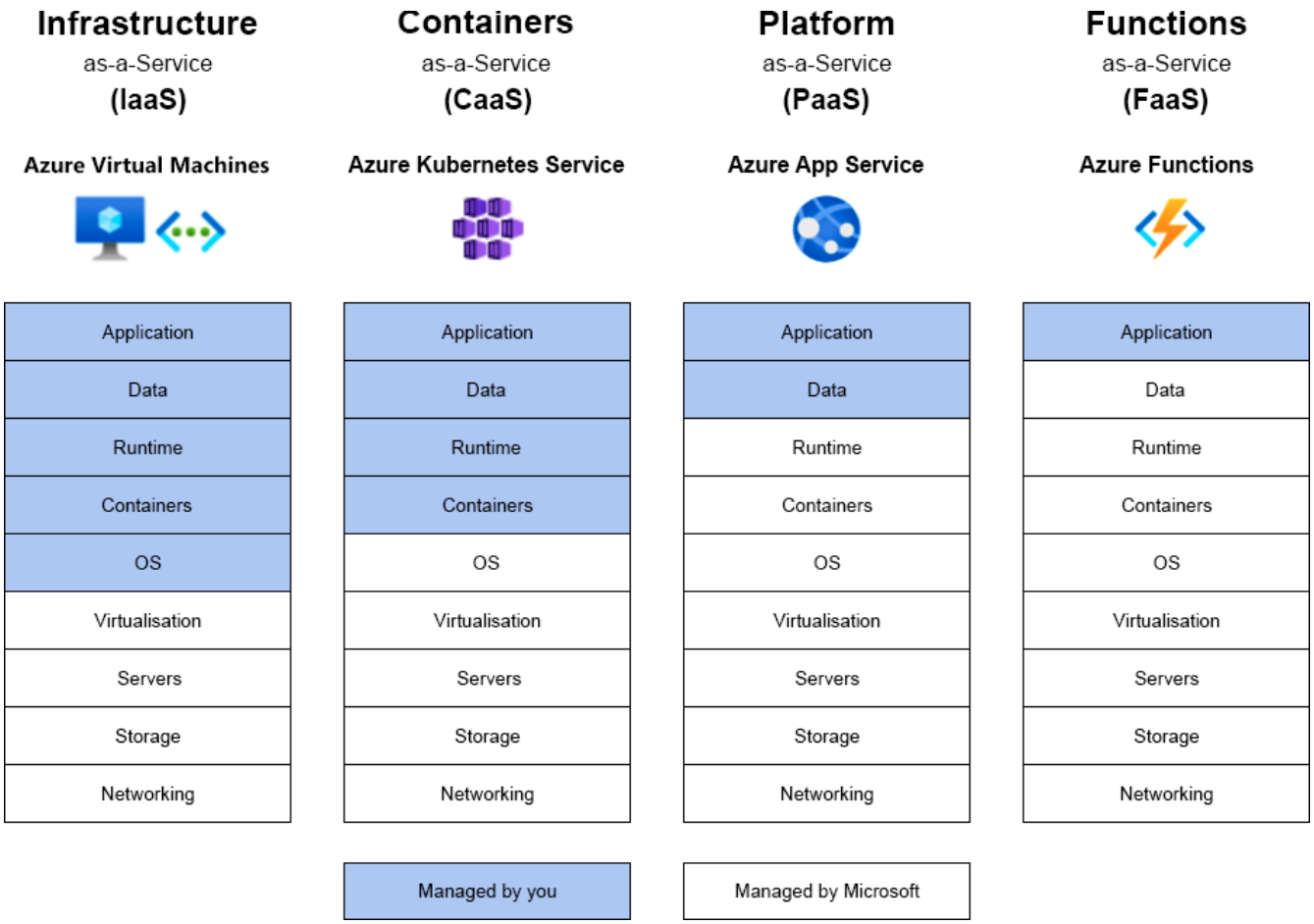


Figure 32: Azure cloud models for Application Development and Hosting

Infrastructure-as-a-Service (IaaS)

In the IaaS model, Azure gives you full control to manage your own application hosting environment.

Introduction to Azure Virtual Machines

In the IaaS model, Azure enables you to deploy or migrate your application to either Windows or Linux virtual machines. You will have full control over the machine configuration. However, in the IaaS model, you are responsible for all operating system upgrades, server software installation, configuration and maintenance.

Key Features of Azure Virtual Machines

- Azure Virtual Machines allows you to have complete control of the operating system.
- You can fine-tune the CPU/memory balance, the machine family (SKU) and the disk layouts.

Use case for Azure Virtual Machines

- Azure Virtual Machines would be a good choice if you wish to have full control over your application infrastructure or to migrate on-premises application workloads to Azure with little to no changes.

To learn more

- Windows Virtual Machines documentation
- Linux Virtual Machines documentation

Container-as-a-Service (CaaS)

A container, by definition, is a standard unit of software that bundles an entire runtime environment (an application, configuration files required to run it, and all its dependencies, libraries and other binaries) into a single package. By containerising the application platform and its dependencies, variances in OS environments and underlying infrastructure are abstracted from it.

Introduction to Azure Kubernetes Service

In the CaaS model, once you run more than one container, you need an orchestrator. The managed container orchestrator in Azure is called the Azure Kubernetes Services (AKS).

AKS manages your hosted Kubernetes environment and makes it simple to deploy a managed Kubernetes cluster in Azure. You can create an AKS cluster in the Azure portal, with the Azure command-line interface or template-driven deployment options (such as Azure Resource Manager templates and Terraform). With AKS, you can easily deploy and manage containerised applications. You do not need to be an expert in container orchestration to use AKS.

The Kubernetes master and worker nodes are deployed and configured for you when you deploy an AKS cluster. During deployment, other features (such as Azure Active Directory integration, monitoring and advanced networking) can also be configured. AKS supports Windows Server containers as well.

Key features of AKS

- Reduces the complexity and operational overhead of managing Kubernetes by offloading much of that responsibility to Azure.
- Azure handles critical tasks like health monitoring and maintenance for you.
- Azure manages the Kubernetes masters for you.
- Kubernetes masters are free of charge.
- You only manage and maintain the agent nodes.
- You only pay for the agent nodes within your clusters.

Use case for AKS

- AKS is a good choice if you want to simplify the deployment and management of applications based on microservices. You can also use AKS to migrate existing applications to containers and run them.

To learn more

- Azure Kubernetes Service

Platform-as-a-Service (PaaS)

In the PaaS model, Azure provides you with a fully managed hosted environment to run your application on. You will not have to worry about the underlying infrastructure details.

Introduction to Azure App Service

Azure App Service is a PaaS that gives you the fastest way to publish your web-based projects. With App Service, you can easily extend your web apps to support your mobile clients and publish REST APIs.

With App Service, you can create the following types of applications:

- Web apps
- APIs
- Mobile app backends

Since all these application types share the App Service runtime, you can literally host a website, support mobile clients and expose your APIs in Azure from a single project or solution.

App Service is designed with DevOps in mind. It supports various tools for publishing and CI/CD, such as:

- Azure DevOps
- GitHub
- Bitbucket
- Docker Hub
- Azure Container Registry

Key features of Azure App Service

- Authentication using social media providers (such as Microsoft Account, Facebook, Twitter, Google)
- Traffic-based auto-scaling
- Testing in production
- Continuous and container-based deployments

Use case for Azure App Service

- Azure App Service supports a wide range of web technologies
- Great way to create proof of concepts to almost any web service or web frontend (such as .NET, Java, Python or PHP, just to name a few)
- Azure App Service would be a good choice when:
  - You are migrating an existing web application to Azure.
  - You need a fully managed hosting platform for your web apps.
  - You need to deploy and run a containerised web app.
  - You need to expose REST APIs with your app.

Migrating to Azure App Service

- The Migrate to Azure App Service tool assists you with the migration of existing .NET and PHP apps to Azure App Service.

To learn more

- Azure App Service overview

Putting it into practice

To put App Service in action, go to Try Azure App Service.

This allows you to provision a short-term proof of concept app and try the platform in a sandbox environment without requiring an Azure subscription. It is free of charge and there's no commitment.

1. From the main page, you can select **Web App** or **Web App for Containers**. Let's pick **Web App**:

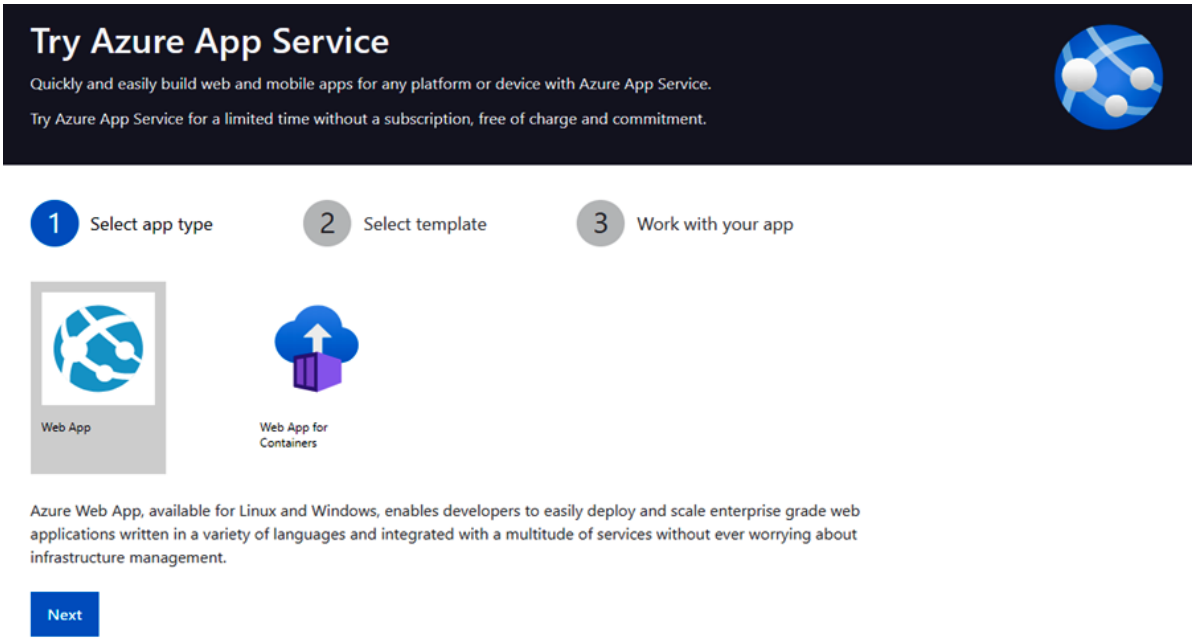


Figure 33: Pick Web App or Web App for Containers

2. Next, pick a language to select a template to create your Web App. Let's pick C# and ASP.NET Core. Then click the **Create** button:

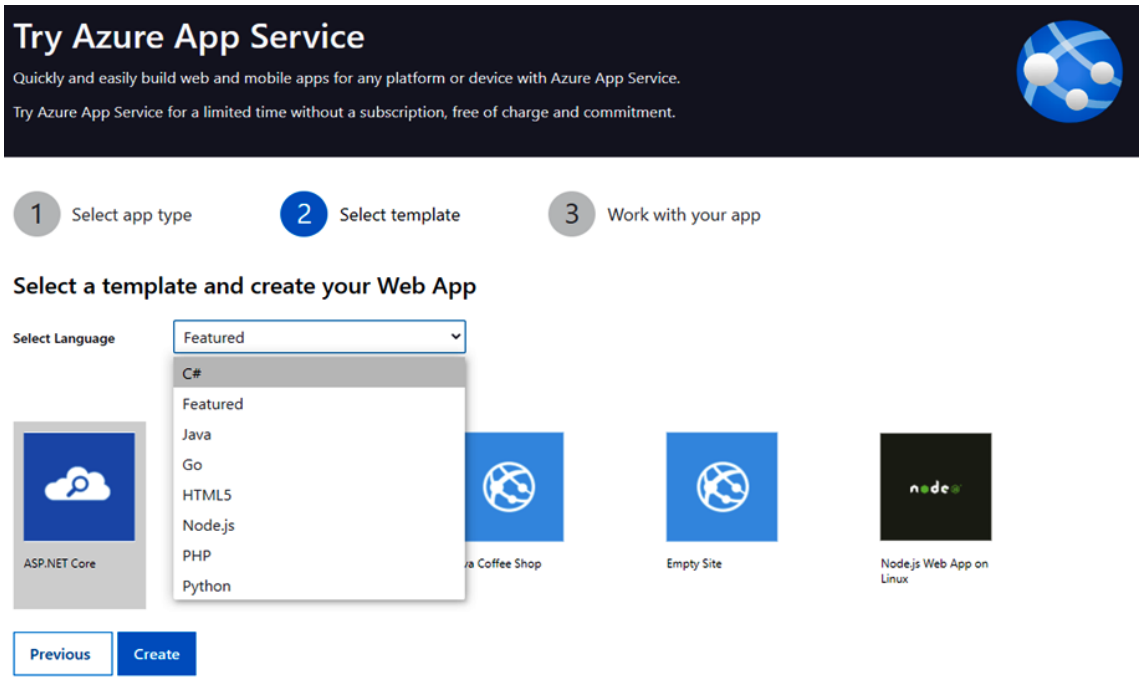


Figure 34: Pick a language

3. In the next step, you would need to sign in to create your Web App using any one of these four options:

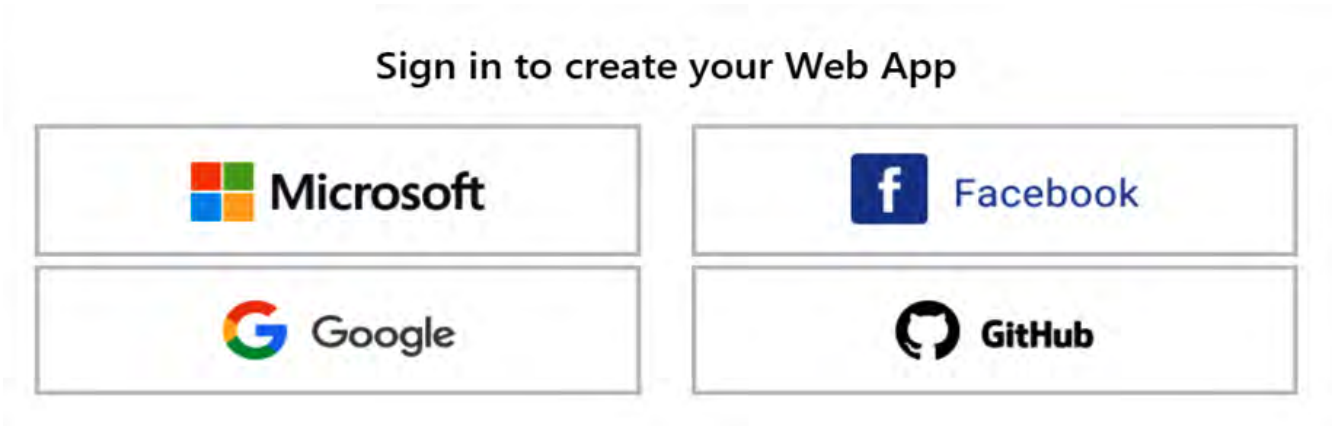


Figure 35: Sign in to create your Web App

4. Once this is successfully deployed, you can make changes to the content and experiment with the changed results:

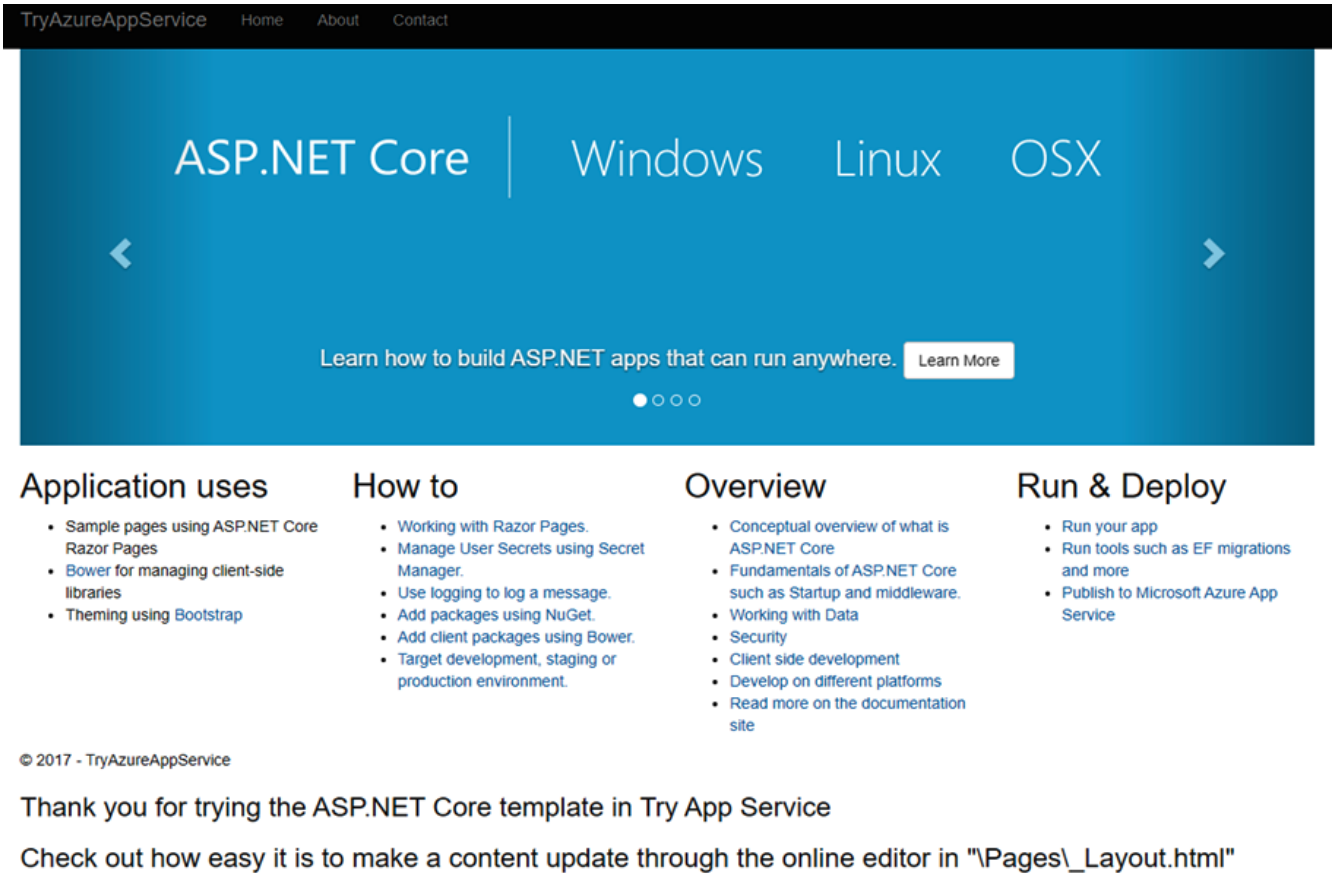


Figure 36: Successfully deployed website

5. When you are done with this proof of concept app, go back to the Try Azure App Service site and experiment with another template.

Next, let's consider the serverless offering known as Azure Functions.

Function-as-a-Service (FaaS)

With the FaaS model, Azure provides you with a serverless environment. All you need to do is develop your code. The underlying infrastructure details are all handled by Azure.

Introduction to Azure Functions

Azure Functions enables you to run serverless code without needing to provision your own infrastructure. An Azure function is a unit of code logic that can be triggered by HTTP requests, an event in another Azure service or on a schedule.

Azure Functions is serverless because you can focus on writing your code without having to worry about a server that executes the code. You are only billed when the endpoint is called. When the endpoints are not being used, there is no charge. With consumption-based billing, you only pay for the time that your code executes, and Azure will scale as needed. This makes Azure Functions an ideal choice for APIs.

Key features of Azure Functions

Features	Benefits
No need to manage any infrastructure	Allows you to focus on adding value. Scaling can be automated and flexible.
Support for many popular languages	You can write your code in: - C# - JavaScript - F# - Java - PowerShell - Python - TypeScript For details on supported languages in Azure Functions, see this documentation.

Features	Benefits
Full development experience	Integrated tools and built-in DevOps capabilities, which allow you to build and debug as well as deploy and monitor.
Simplified integration	Easily integrate with Azure services and Software-as-a-Service (SaaS) offerings.
Pay-per-use pricing	With the Consumption hosting plan, you are only charged when your code runs.

Using Azure Functions, you can build small pieces of functionality quickly, and host them in an elastic environment that automatically manages scaling.

Use case for Azure Functions

- Azure Functions would be a good choice when you have code that is triggered by other Azure services, by web-based events or on a schedule. You can also use Azure Functions when you have no need for the overhead of a complete hosted project, or when you only want to pay for the time that your code runs.
- To further exemplify this, suppose you want to automate the image resizing process whenever a new image file is uploaded to Azure Blob storage. You can create an Azure Function that is triggered every time a new image file is uploaded to Azure Blob storage. The function then resizes the image and writes it back to the Blob storage account. There is no need to write the plumbing for connecting to Blob storage; you just configure it.

To learn more

- Azure Functions documentation
- Azure Functions triggers and bindings concepts

Putting it into practice

Like Azure App Service, you can try Azure Functions for free in a sandbox environment, without an Azure subscription:

1. Navigate to this URL and create your first Azure function.
2. Select the **'I'm not a robot'** checkbox and click the **Create a sample function app** button:

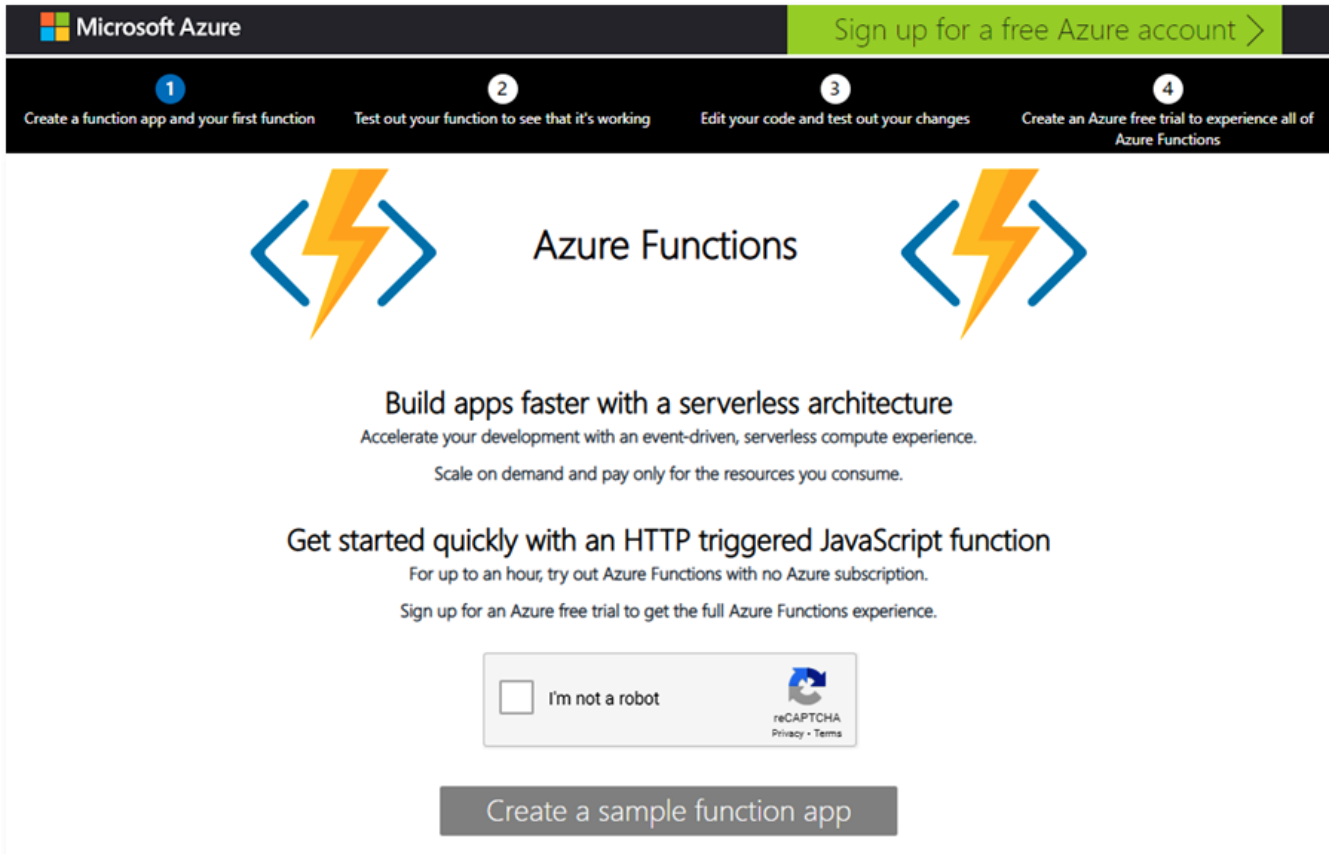


Figure 37: Create a sample function app

3. A sample HTTP trigger function app is pre-populated in the editor pane for you. This function is triggered by an HTTP GET or POST request and sends as output an HTTP response based on the user code provided. Click the **Run** button to see it in action:

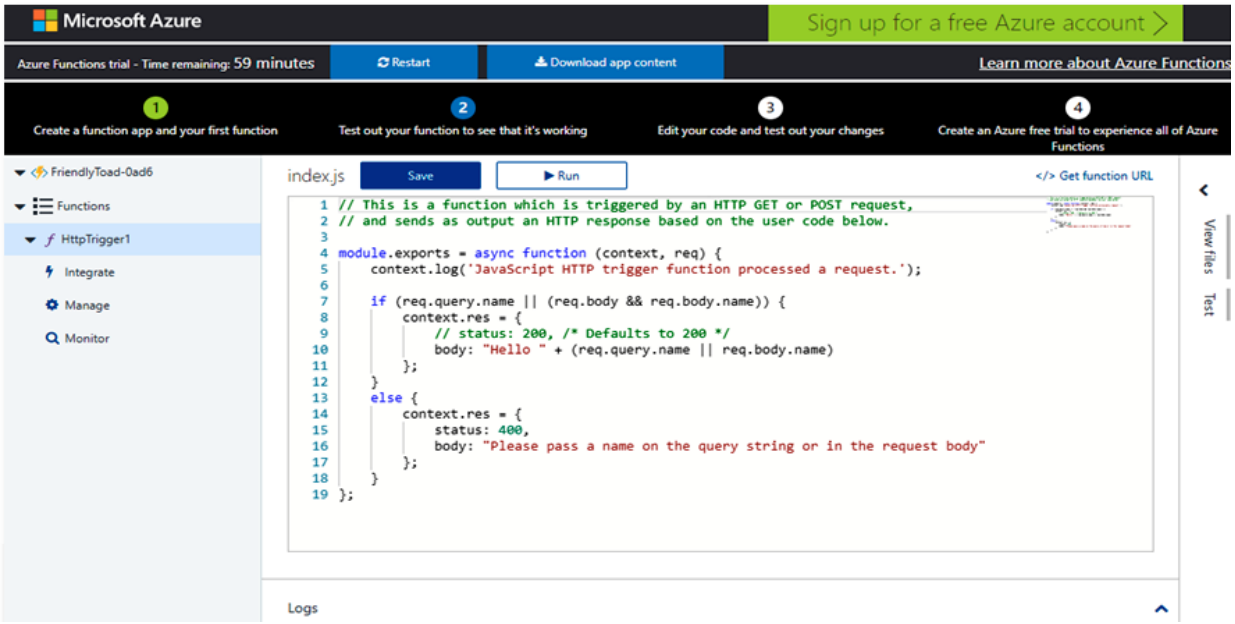


Figure 38: Click Run

4. If the execution is successful, you will see the following output:

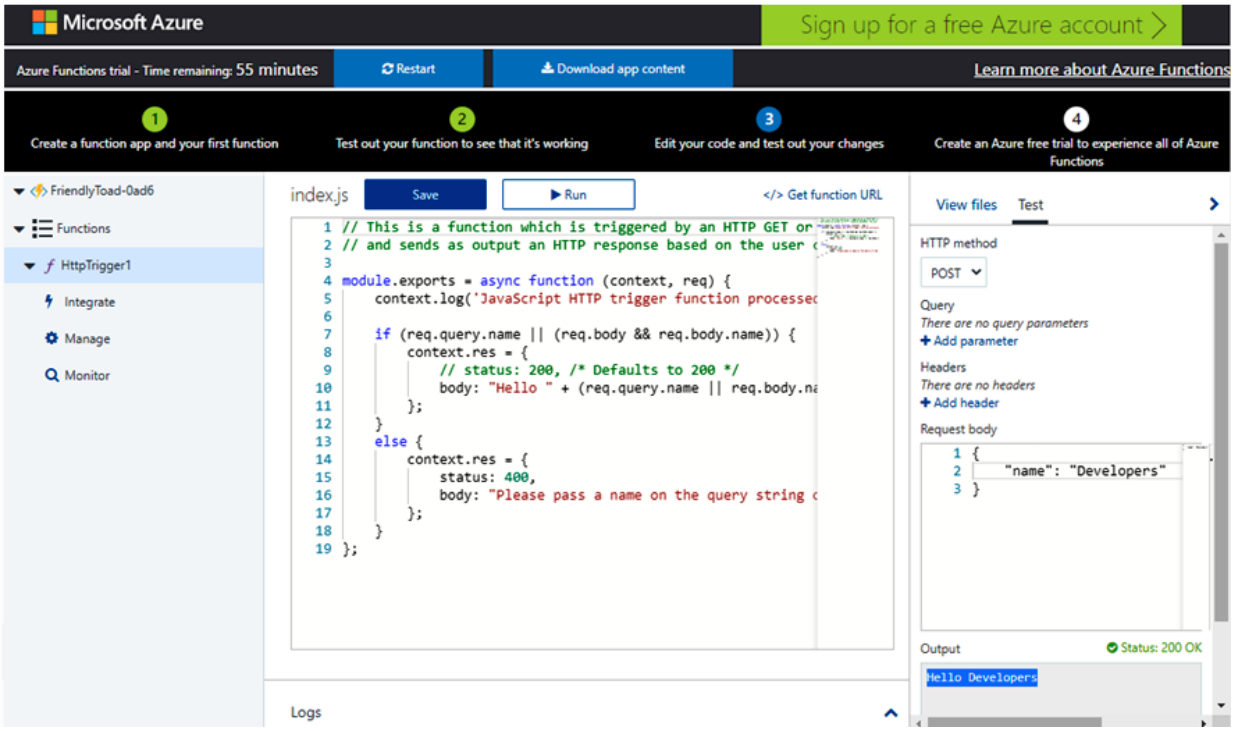


Figure 39: Check the output after execution

5. Now, change the **HTTP method** to **GET** and provide the query name and variable as follows. Observe the output after you click the **Run** button:

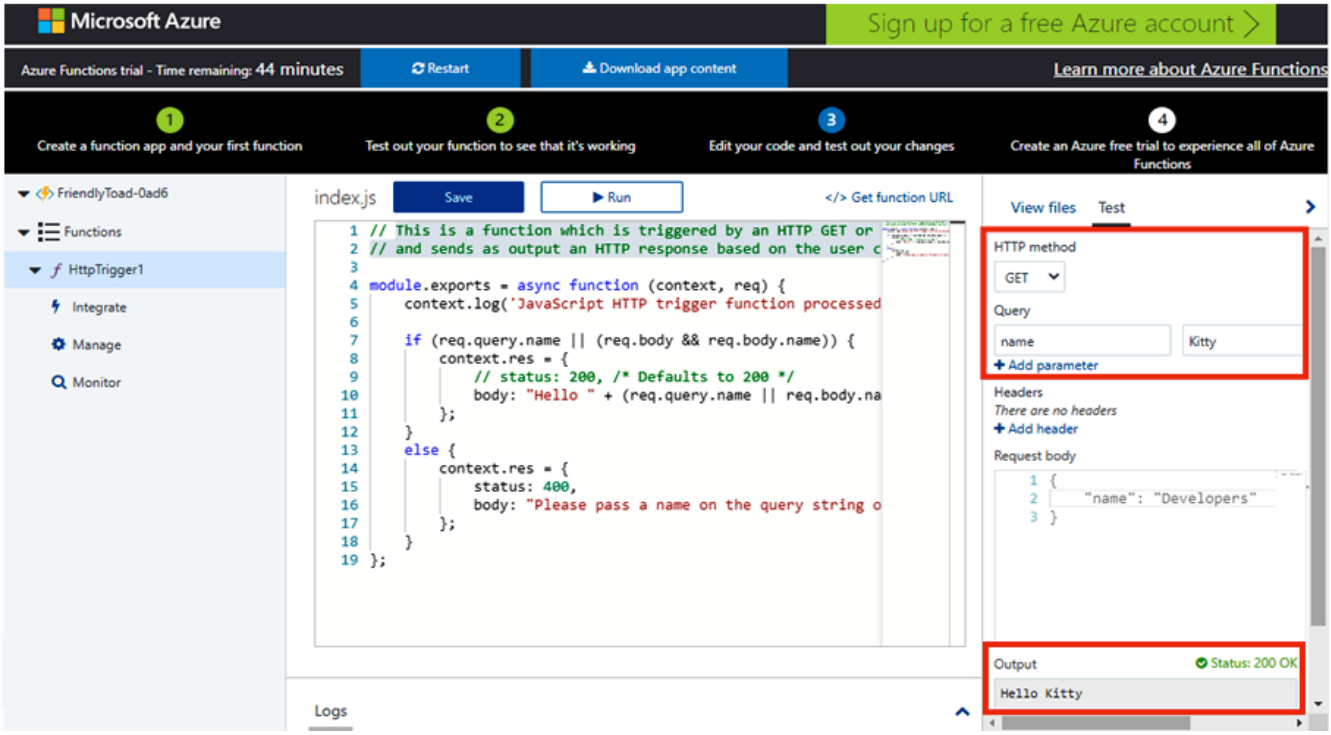


Figure 40: HTTP Get

6. You can build another simple Azure Functions proof of concept app in this free interface. Give it a try and experiment with it!

Find out more about [choosing an Azure compute service for your application]. In the next section, we will provide you with some useful learning materials and resources.

Chapter 5: Further learning and resources

Learning the Azure fundamentals

If you are new to Azure, we recommend the following interactive learning paths:

- Azure Fundamentals
- Explore Microsoft Azure cloud concepts
- Distinguish Microsoft Azure Core Services
- Examine Microsoft Azure security, privacy, compliance and trust
- Review Microsoft Azure pricing, service level agreements and lifecycles
- Microsoft Learn

Once you have become proficient in Azure, you might want to consider taking the Microsoft Azure Fundamental AZ-900 exam to get certified. For more information, please visit this exam guideline.

Tools that you need for developing your proof of concept project for Azure

The following is a list of tools that are essential for the examples shown in this guide:

- Azure subscription
- Visual Studio Code
- Azure Functions Core Tools
- GitHub account
- Microsoft Edge (Chromium-based) browser
- Node.js

Other useful resources

Introduction to Azure Virtual Machines

- Windows Virtual Machines documentation
- Linux Virtual Machines documentation

Introduction to Azure Kubernetes Service

- Azure Kubernetes Service

Introduction to Azure App Service

- App Service Migration tool
- Azure App Service overview
- Try Azure App Service

Introduction to Azure Functions

- Azure Functions documentation
- About Azure Functions triggers and bindings concepts
- Try Azure Functions

Chapter 2: Sample project – implementing a web app using Azure Static Web Apps

- GitHub Actions official documentation
- Review pull requests in pre-production environments in Azure Static Web Apps

Chapter 3: Sample project – building an intelligent chatbot

- Microsoft Bot Framework SDK
- Bot Builder Samples
- QnA Maker

General resources

- Choose an Azure compute service for your application
- Azure free account FAQ

Conclusion

In this guide, we discussed how a proof of concept project can be a valuable tool to evaluate whether a potential technology or concept can be used to fulfil the requirements of a business solution. It can help you to identify any potential technical and logistical issues before it is implemented in a mainstream project. In addition, it provides timely insights on the technology while mitigating risks by allowing key decisions to be made in the early stages of the development process.

In **Chapter 1:** Proof of concept guide, you learned the core foundations necessary for planning and executing a successful proof of concept project.

In **Chapter 2:** Sample project – implementing a web app using Azure Static Web Apps and **Chapter 3:** Sample project – building an intelligent chatbot, we showcased two practical projects that might inspire you to come up with your own proof of concept projects.

In **Chapter 4:** An overview of Azure for developers, you were introduced to the Azure platform. You learned how to obtain an Azure free account to get started. You also learned how to pick the appropriate cloud model and services to start developing your proof of concept project for Azure.

We also provided you with some useful learning materials in **Chapter 5:** Further learning and resources.

You are now ready to begin your proof of concept project. Good luck!

Get started today

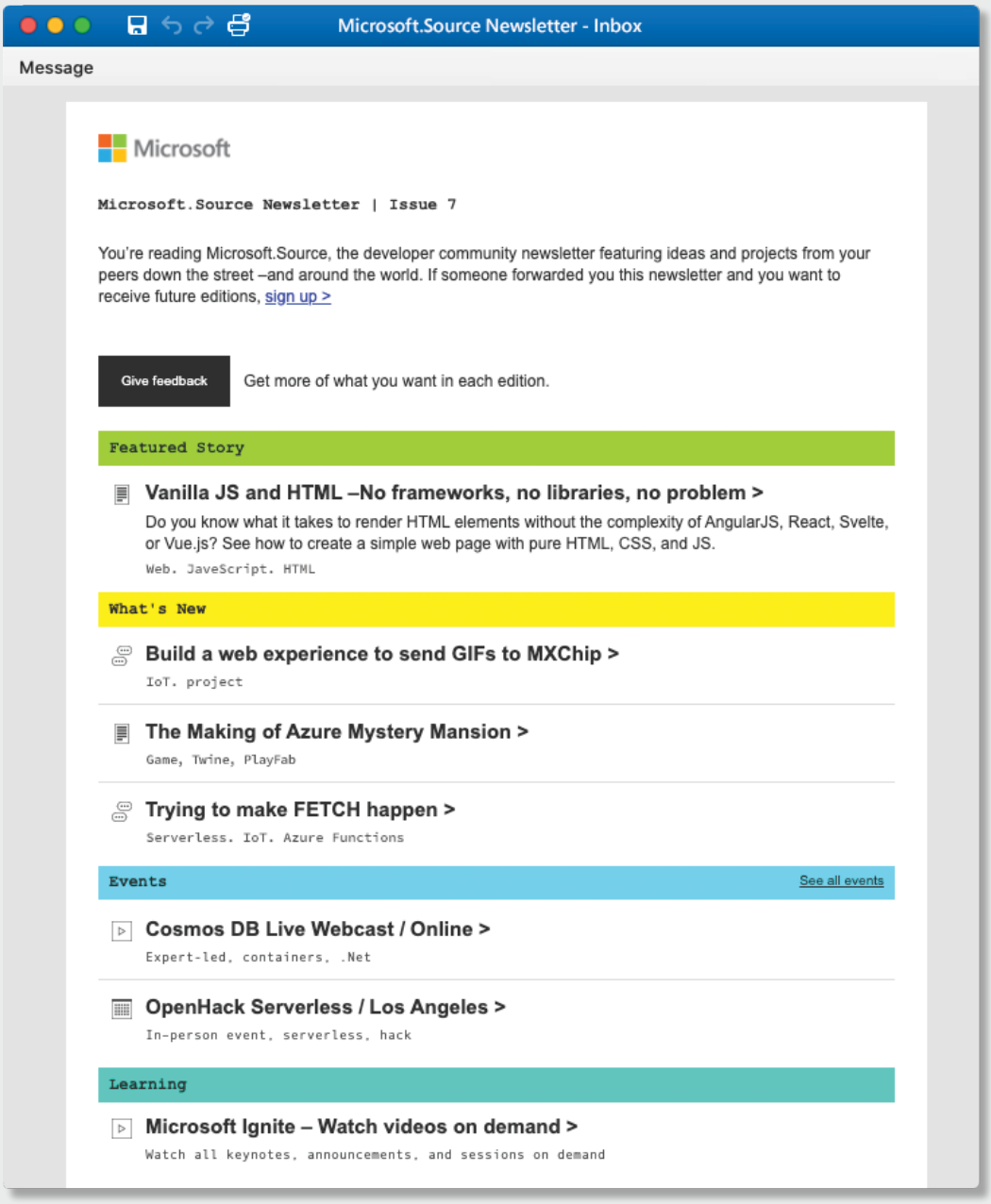
Sign up for an Azure free account

Learn more about Azure solutions

Speak to a sales specialist for help with pricing, best practices and implementing a proof of concept

Microsoft

Microsoft



By developers, for developers

Microsoft.Source newsletter

Get technical articles, sample code and information on upcoming events in Microsoft.Source, the curated monthly developer community newsletter.

- Keep up on the latest technologies
- Connect with your peers at community events
- Learn with hands-on resources

Sign up