Method 2: Creating a web app bot using the CLI

In addition to the prerequisites I described above, for this method you need to have the Azure [command-line](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest) installed and configured.

At the time of writing, bot service is in preview for the command line, so it doesn’t come with the defaults for the CLI—you need to enable the bot service extension in the CLI. To do so, let’s run the following command:

az extension add -n botservice

Let’s create the resource group for the web app bot with the following command:

az group create --name cmpizzabot --location centralus

To create the web app bot, run the following command:

az bot create --resource-group "cmpizzabot" --name "cmpizzabot" --kind "webapp" --description "A Pizza Bot"

Let’s explore the parameters and values we entered here:

* **–resource-group**is the name of the resource group we just created.
* **–name**is the name of the web app bot that needs to be unique.
* **–kind** is the hosting type for the bot.
* **–description** is the description of what the bot does.

By default, Azure CLI will choose a C# template and other types of settings like the virtual machine size, location, and pricing tier that we reviewed in the previous section.

OK, so let’s deploy our pizza bot.

Deploying a Pizza Bot

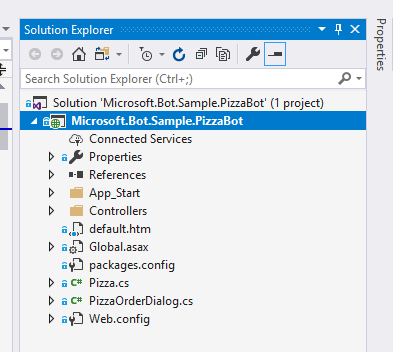
We are all tired of the simple “hello world” examples, so let’s use one from the existing list of examples: [a Pizza Bot](https://github.com/christianhxc/azure-pizza-bot) (based on [this example](https://github.com/Microsoft/BotBuilder/tree/master/CSharp/Samples/PizzaBot)).

Run Bot locally

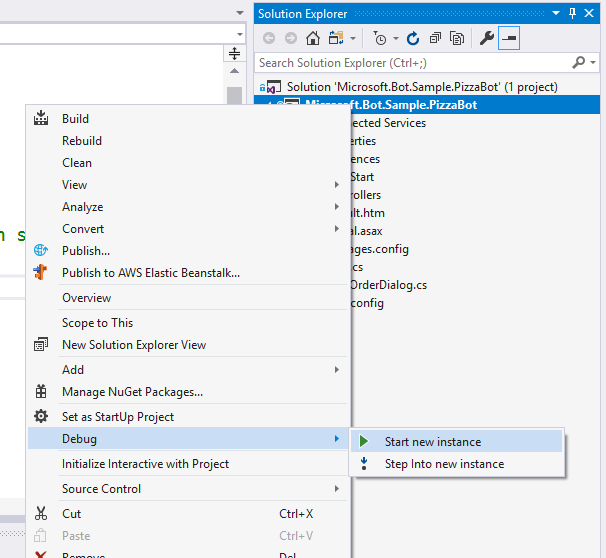
Let’s start by cloning the GitHub repo by running the following command:

git clone https://github.com/christianhxc/azure-pizza-bot.git

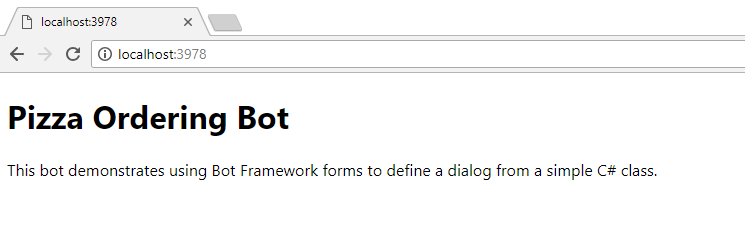
Open the solution with Visual Studio:



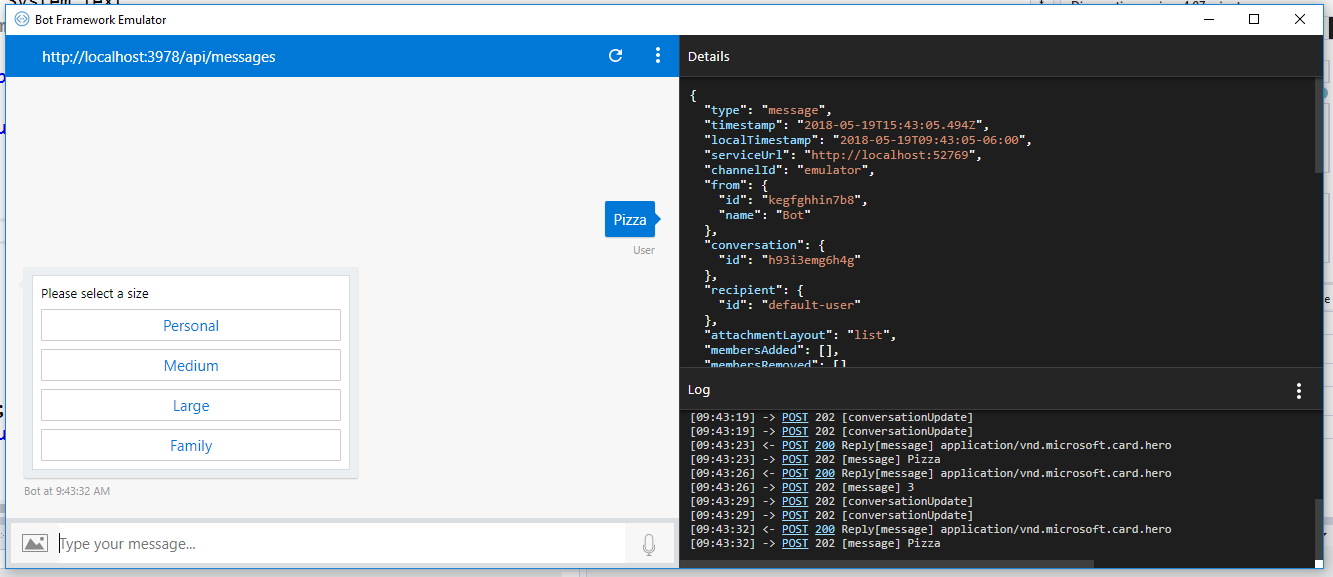
Let’s run the app locally. Right click on the project, then choose “Debug,” and then “Start new instance.”



It will start running and you’ll see this:



Now, open the bot emulator you installed before when you were setting the prerequisites. Enter the local URL of the bot. In my case, it’s **http://localhost:3978**, as you saw in the above image. In order for the emulator to work, you need to enter the URL that Visual Studio created as well as the message’s path. As seen in the screen below, I put [**http://localhost:3978/api/messages**](http://localhost:3978/api/messages) to interact with the bot.

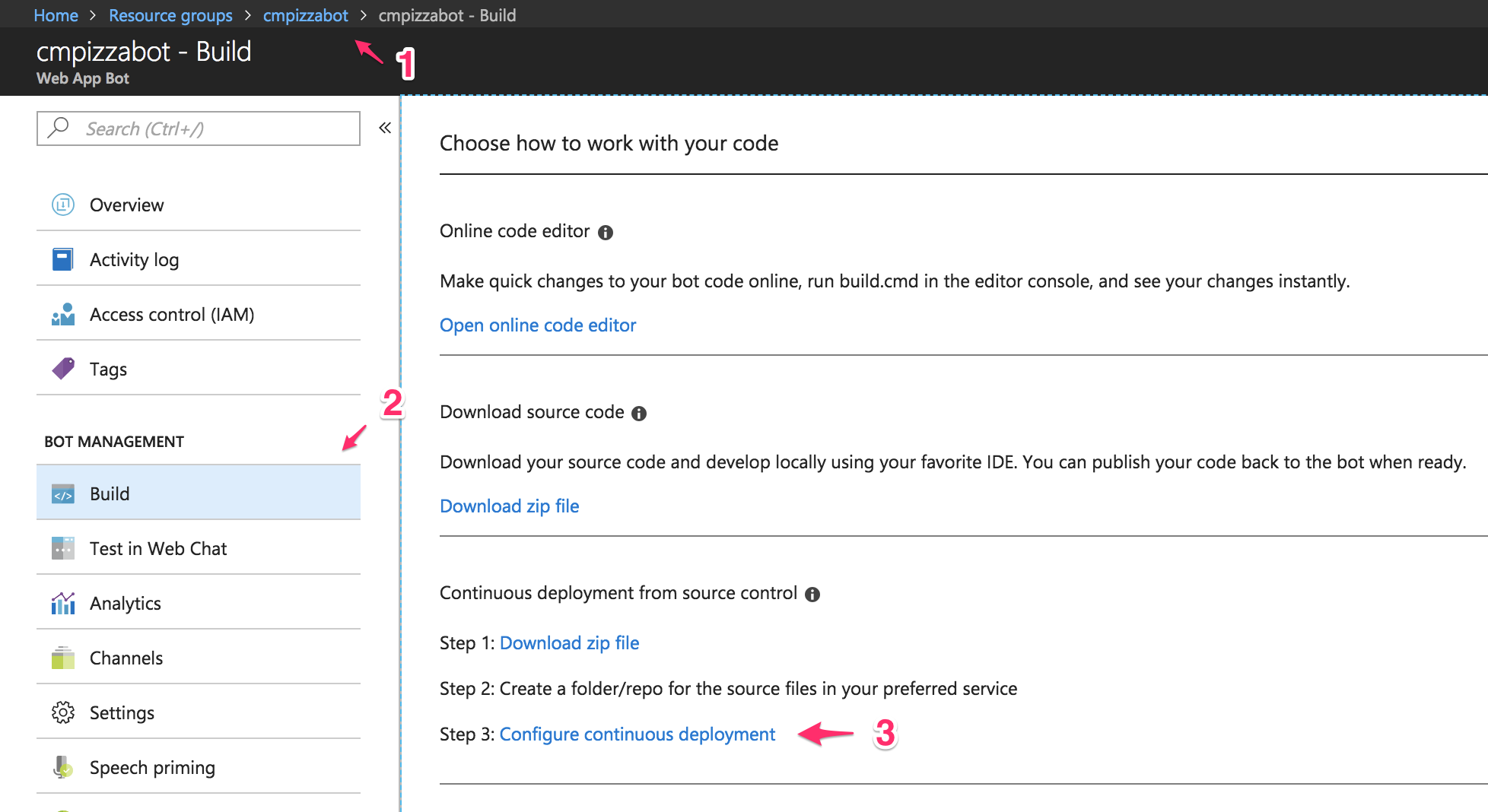


Just enter “pizza” and the bot will reply back with the options configured to select a pizza size.

Set up continuous deployment

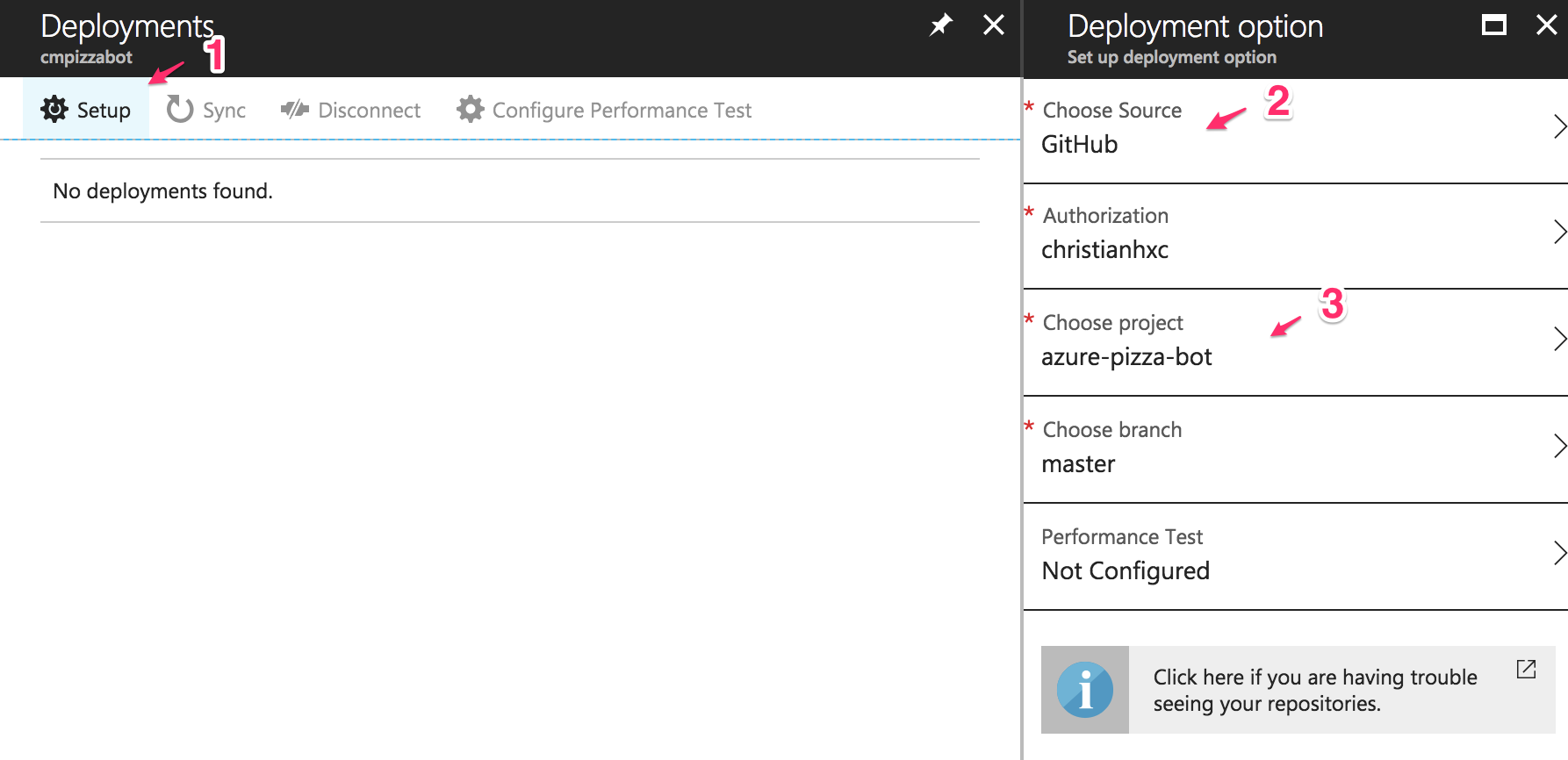
So how do we deploy this bot to Azure? Well, you could do it using Visual Studio, but that only works if you’re the only one developing. Also, it involves a set of steps that you’ll need to run manually. But what if you could “forget” about that and deploy the changes you just did after pushing changes to the repo? That’s way more helpful, so let’s learn how to do it.

Go back to the Azure portal to the web app bot page (1) where we were testing the bot in the portal. Click on the “Build” link (2) and then click on the “Configure continuous deployment” link (3) as seen in the image below:

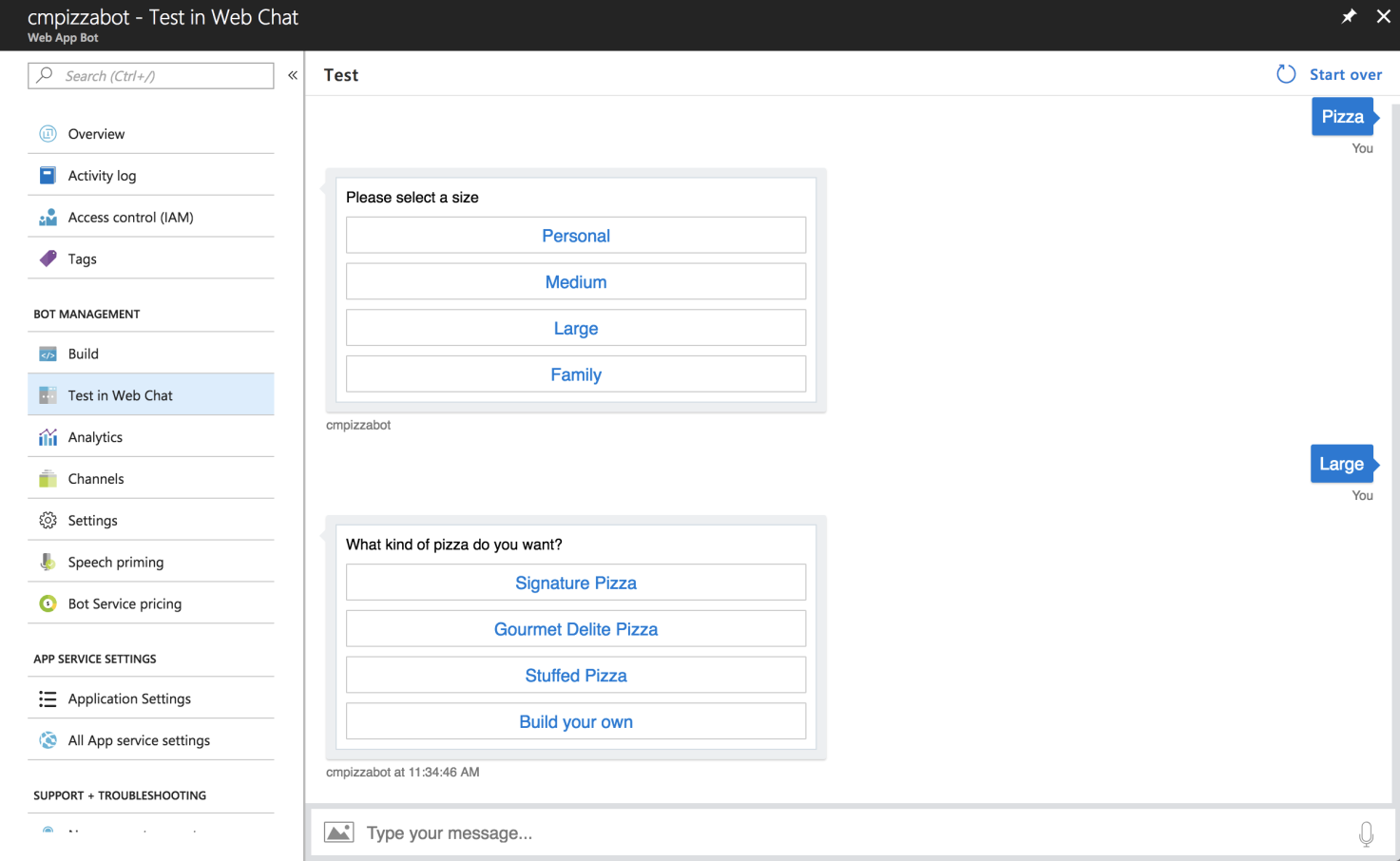


We’re going to configure the continuous deployment (CD) process with GitHub. So you’ll need to push the pizza bot to a repository in your GitHub account. In Azure, you’ll need to create a connection with GitHub—if you don’t, this guide won’t work.

The below screen should appear. Click on the “Setup” link (1) and then choose GitHub (2). You’ll have to configure access by logging in to GitHub and creating the authorization. Choose the repository you just created for the pizza bot code (3).



Click on the blue “OK” button and then wait. It’s a nice opportunity to grab another slice of pizza. A new deployment will be triggered, and you’ll see the new bot running when testing it in the portal:



**NOTE:**If for some reason after deploying the new source code **the bot stops working**, make the initial [deployment using Visual Studio,](https://docs.microsoft.com/en-us/azure/bot-service/bot-service-continuous-deployment?view=azure-bot-service-3.0) and make sure you’ve removed any additional files in the server before doing the deployment.

Next time you do a change in the project and push it to GitHub, Azure will automatically update the web app bot with the latest changes.

Bots that scale as you grow

And that’s all for now! You can easily create bots using the Microsoft Bot Framework and then host it using Azure Bot Service. You’ll be able to scale the app as necessary to support traffic increases in case the bot gets famous. Azure Bot Service will continue to evolve, so I invite you to keep up to date by reading the official docs.