



Microsoft **Imagine**

Using Microsoft Azure for Students

Dive into **Azure** through [Microsoft Imagine](#)'s free new offer and learn how to develop and deploy to the cloud, at no cost!

To take advantage of Microsoft's cloud development platform, you'll need to create a [DreamSpark student account and verify your student status](#), and then [install Visual Studio Community 2013](#) (or a FTP client) to publish your work to Azure. After you've set up those, you'll need to [register a Microsoft Azure account via DreamSpark](#).

Once you've set up the foundation of your development environment, select how you'd like to publish your work – we'll walk you through the basic process of getting your project on Azure and seeing it live!

There are three main ways of publishing your code to Azure:

- [Publish a Web App to Azure using Visual Studio](#)
- [Enabling Continuous Integration with Azure Web Apps & GitHub](#)
- [Publish a Web App to Azure using FTP](#)

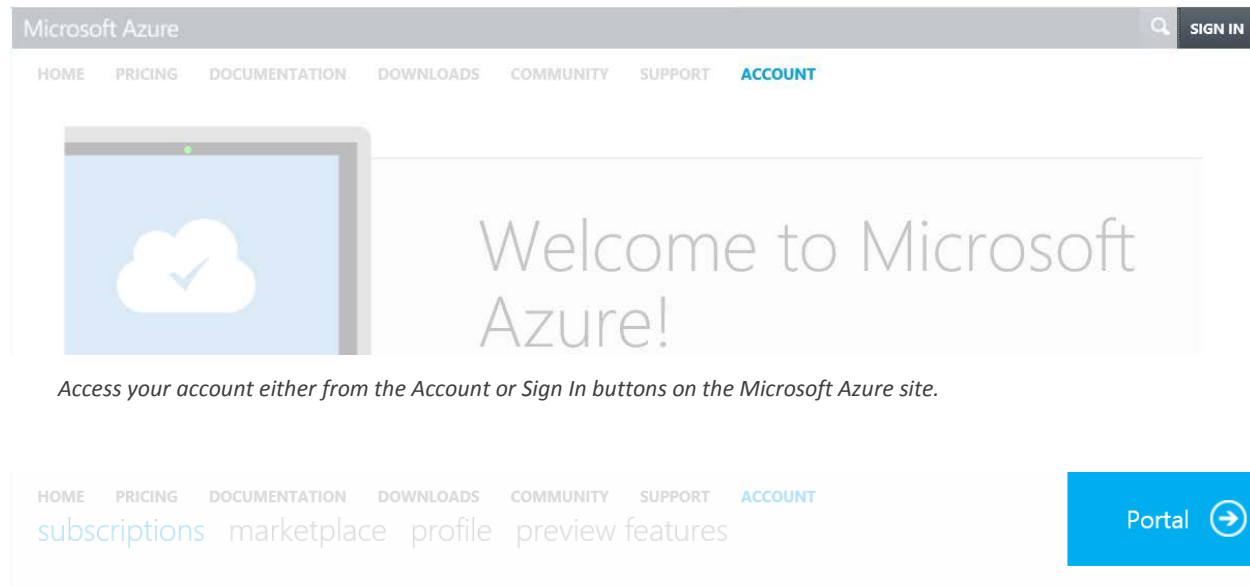
Publish a Web App to Azure using Visual Studio

When publishing a **Web App** (like an ASP.NET project) to **Azure**, here are the basic steps that we'll cover:

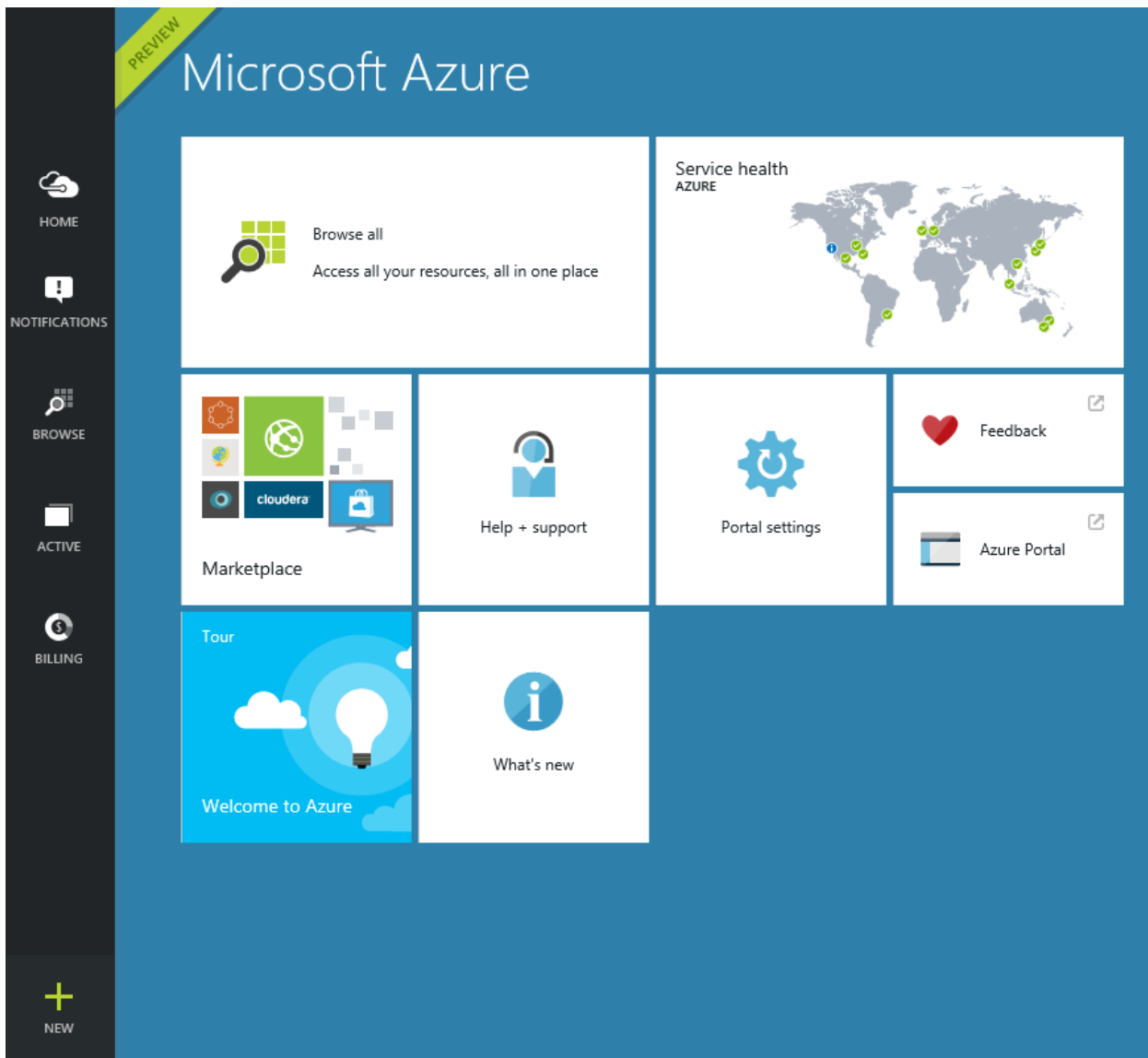
1. [Creating the web app](#)
2. [Downloading a publish settings file](#)
3. [Using the publish settings file to publish the project](#)

Step 1: Creating the web app

First, [log in to your DreamSpark Azure account](#) from the main site, and then open the **Azure Portal**.



Access the Azure Portal.

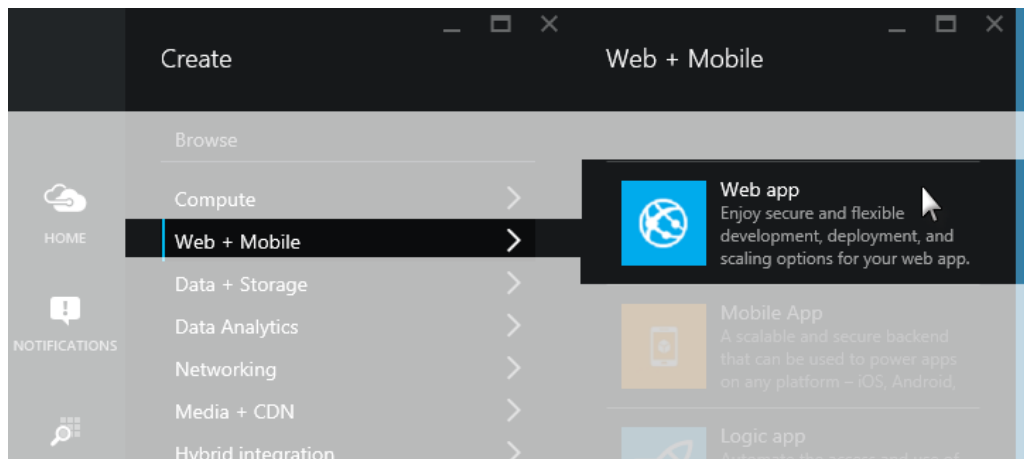


The main Azure Portal, a convenient dashboard for checking your Azure project status.



From the main **Azure Portal**, in the lower right-hand corner, click on **NEW** to start creating a new web app.

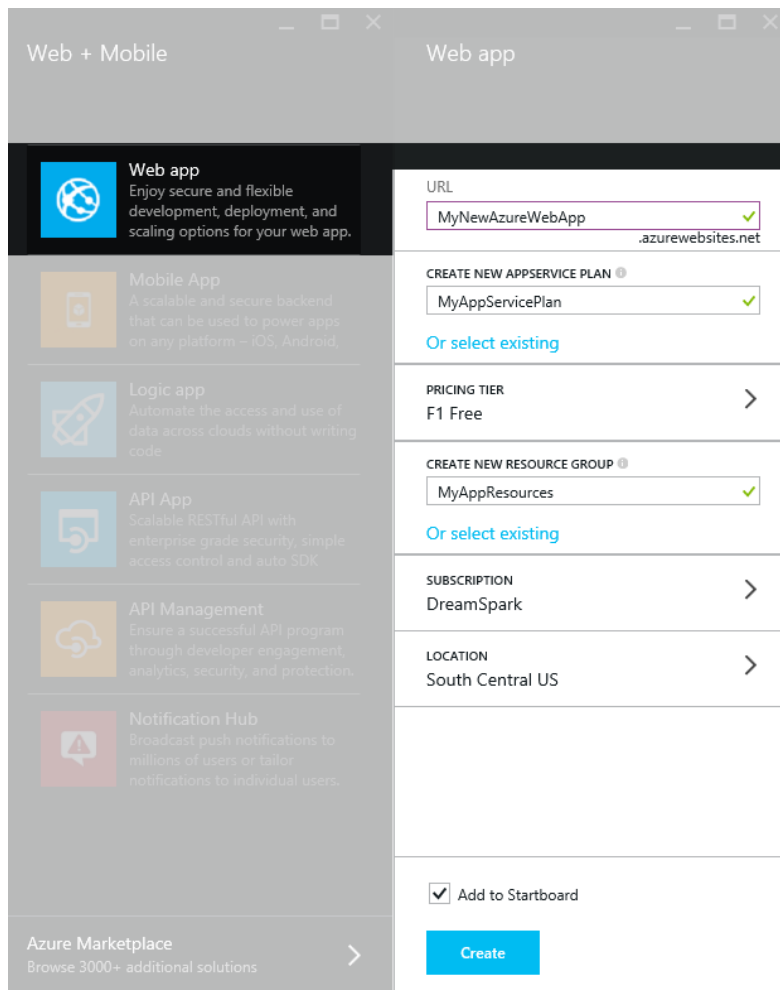
After selecting the **NEW** button, you'll be offered a menu full of different things you can create on Azure, but for now, focus on **Web+Mobile**, and then from the **Web+Mobile menu**, choose **Web app**.



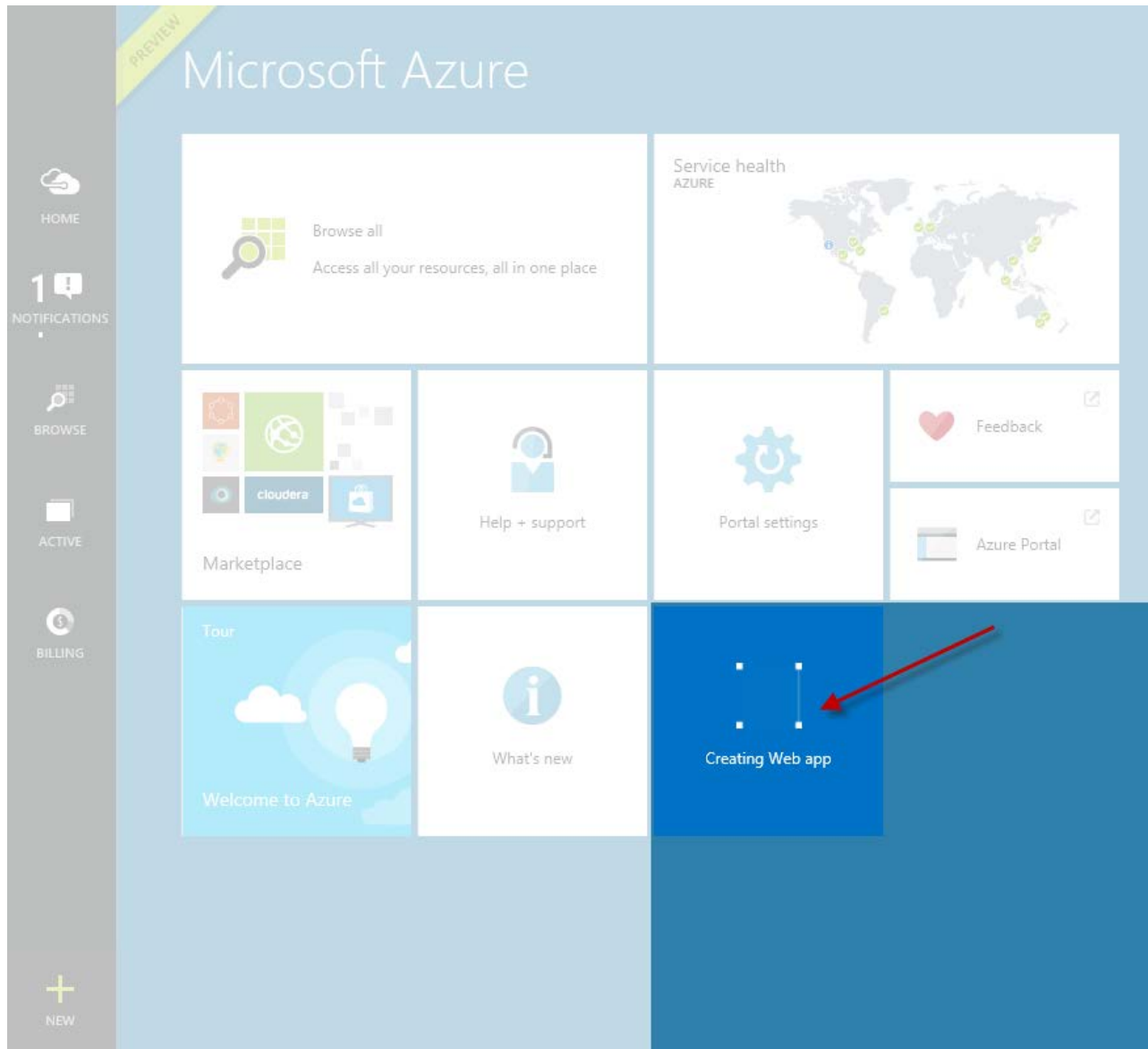
Menus for selecting and creating a new Web app.

Once you select **Web app**, a new menu will open up to the right side. You'll want to enter a **URL** for your project, and choose to name and create a **New AppService Plan**. (All you do is type in any name you want to use. You might try your URL plus "ASPlan" to keep it simple.) Keep the pricing on **F1 Free** for now, and create a **New Resource Group**. (Again, just type in any name you want to use, such as your URL plus "RG".) Azure will let you know if names are taken, so keep adjusting your naming if it turns out a name you'd like is unavailable. Keep the **location** used for your cloud development on South Central US for now, though in future projects you'll be able to customize this if you'd like. Leave the checkbox for **Add to Startboard** checked, so that it's easier to access this work in the near future.

Once everything has been entered to your liking, hit the **Create** button.



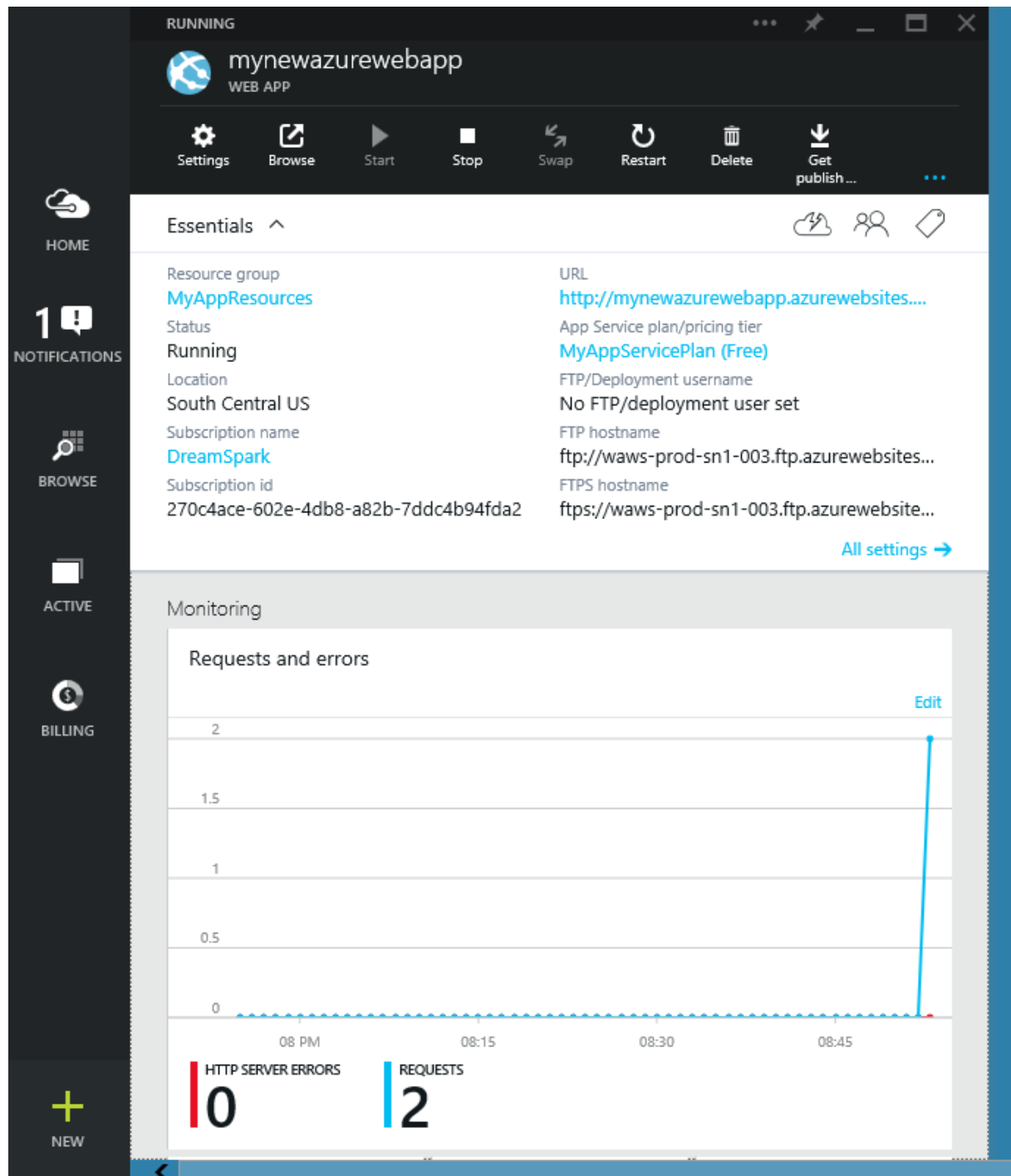
Completing the details for the new Web app.



New tile depicting Web app creation in-process.

Microsoft Azure will work on setting everything up and initial creation of your Web app, and then the tile will appear on your **Startboard** once it's ready.

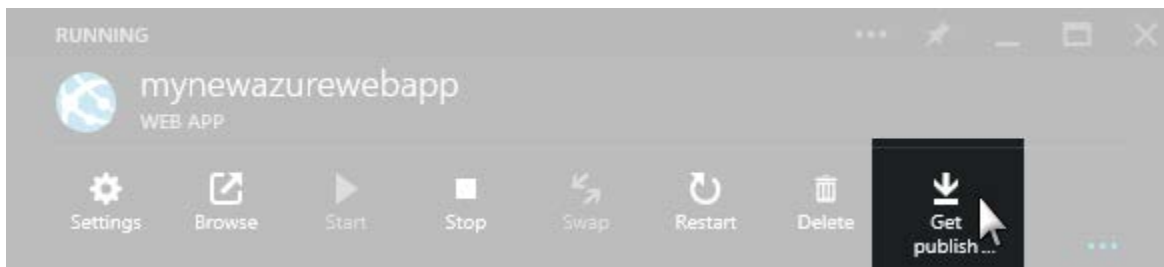
Click on your web app's tile to open up the related display, which gives you information about your web app's live performance.



Web app information blade.

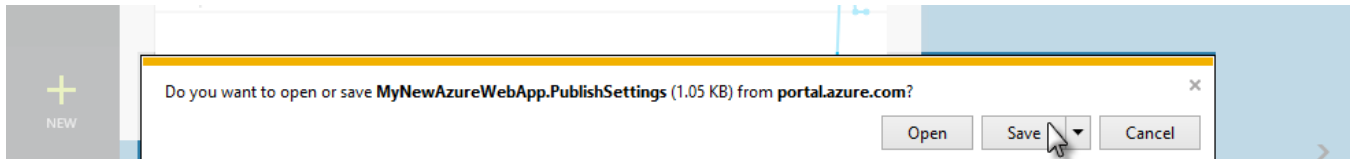
Step 2: Downloading a publish settings file

At the top of this display, you'll want to click on the **Get Publish Settings File** to get the necessary publish settings for this particular Web app.



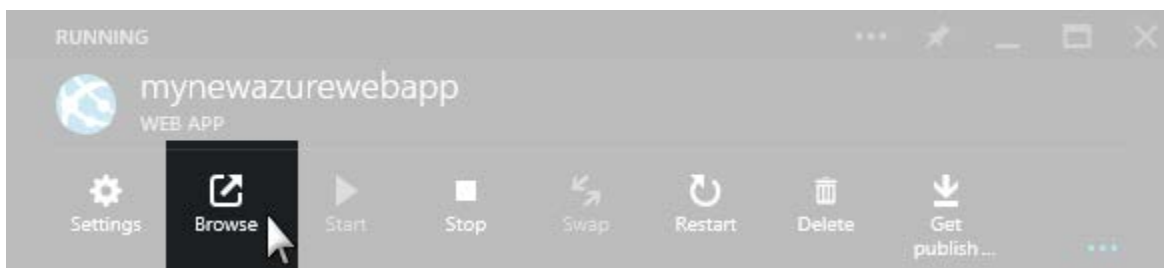
Get Publish Settings File button of interest.

Now you'll need to save this file to your **hard drive**, since it's useful when working in **Visual Studio** (more on this shortly).



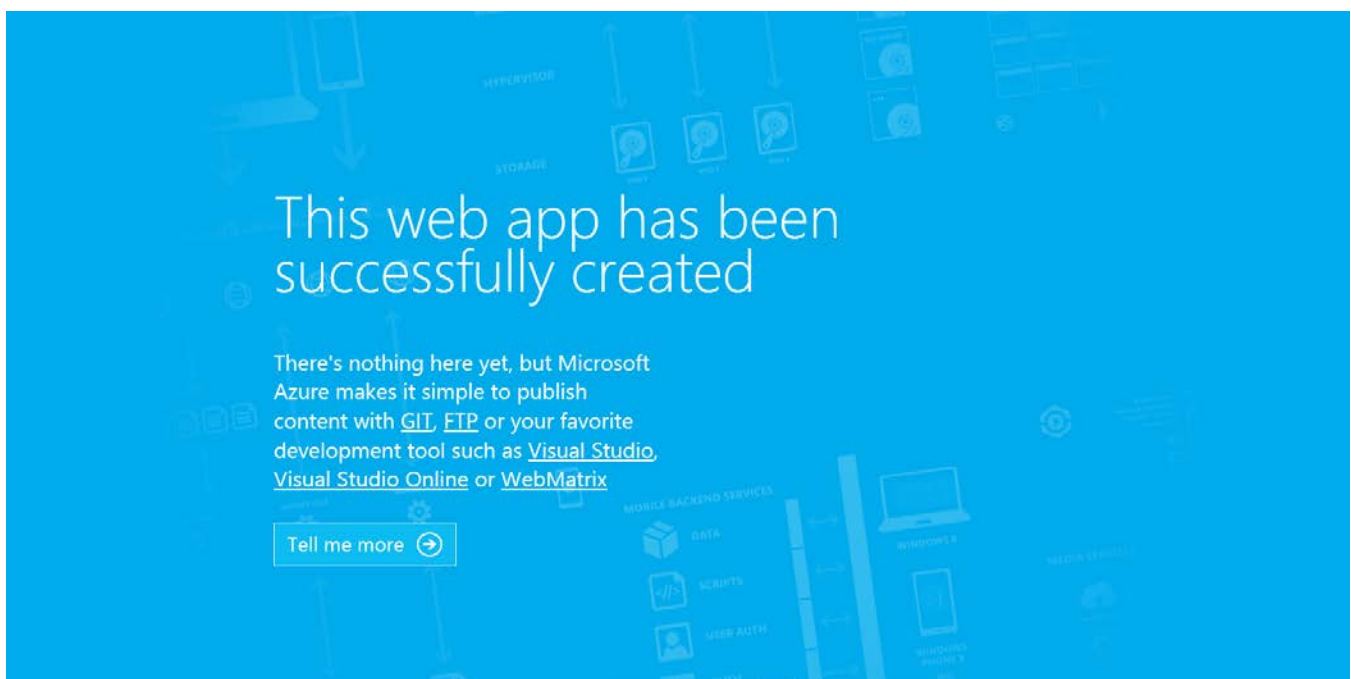
Save the publish settings file somewhere easily located later on.

Now we can head back over to the Web app's blade display and select **Browse**, which will let us load the app in a browser.



Browse to see the app in browser.

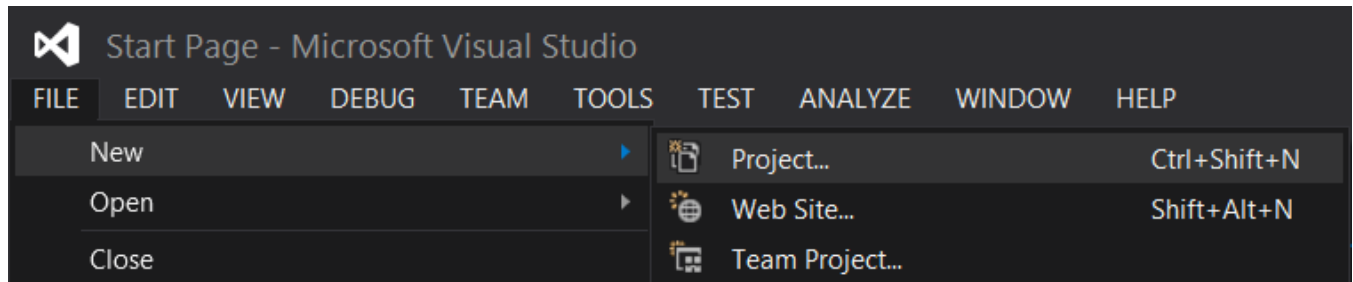
The default **Web app hosting start** page will appear, so now we're ready to start working on the main content of the project in **Microsoft Visual Studio Community 2013**.



Default starting page for Web apps in Microsoft Azure.

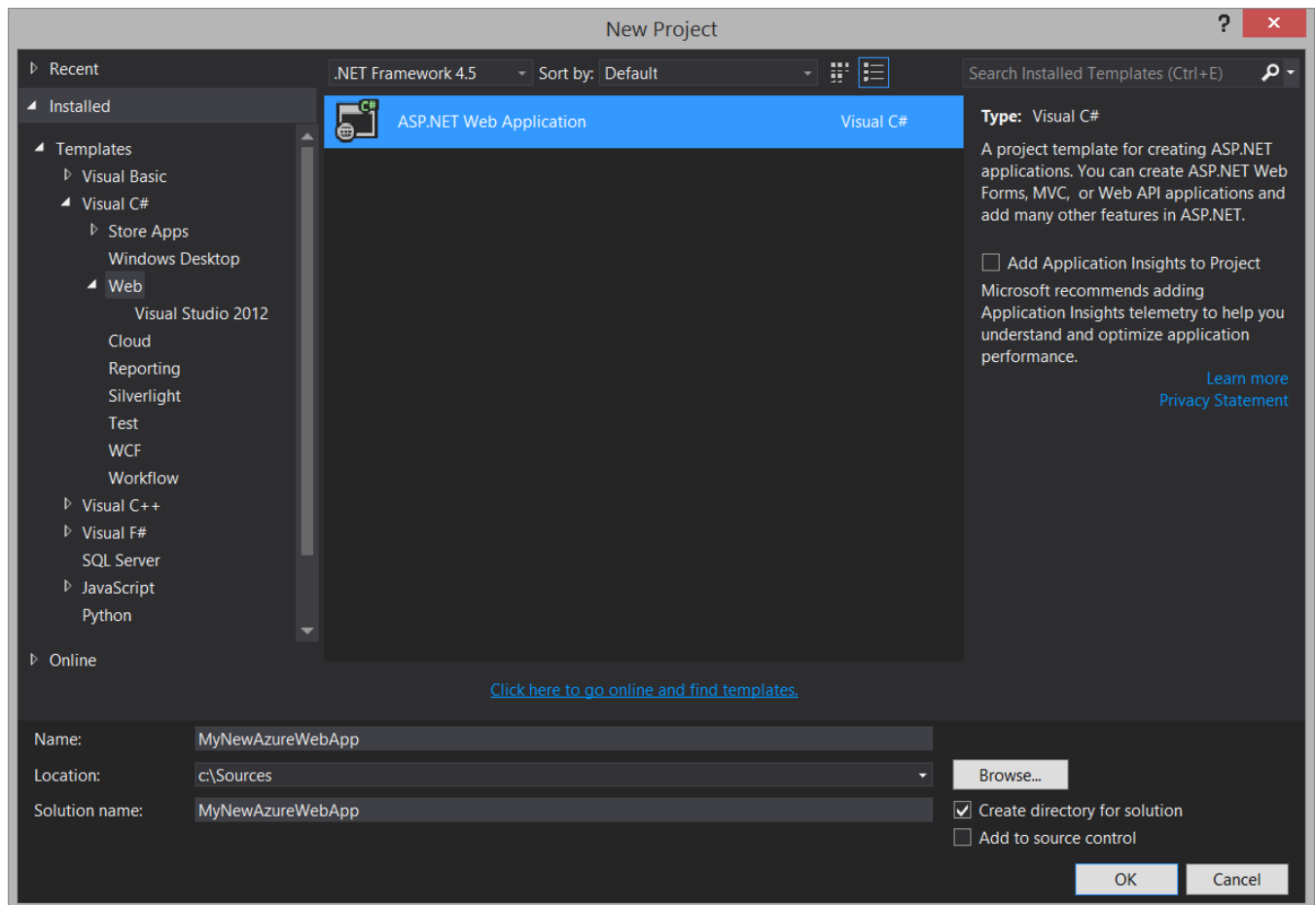
Step 3: Using the publish settings file to publish the project

Open **Microsoft Visual Studio Community 2013** and then **File > New > Project...**



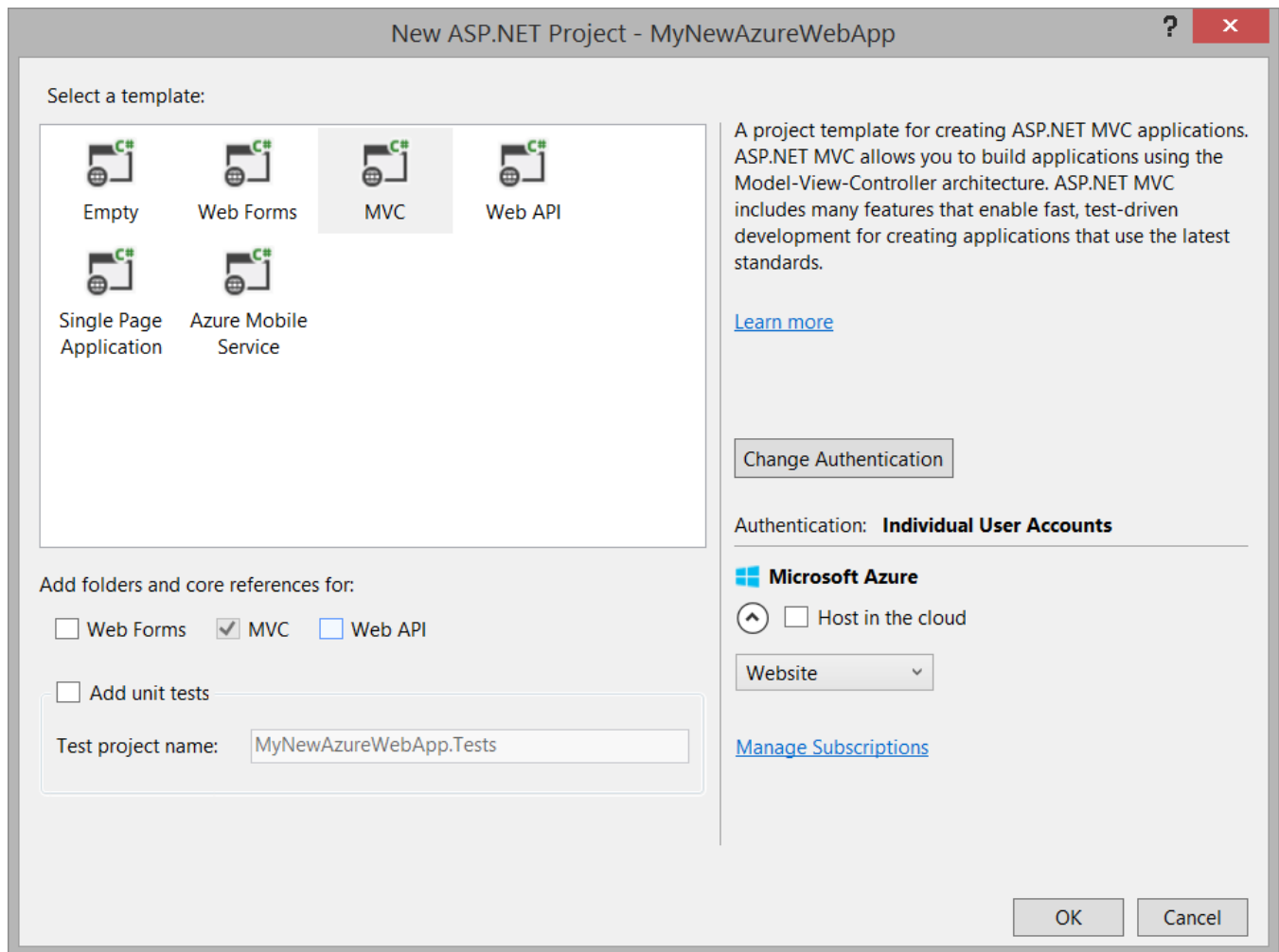
In Visual Studio Community 2013, start a New Project...

In the lower part of the **New Project** window, create a **Name**, choose a good **Location**, and overall **Solution name** for this project. Once those are in order, on the left-hand side, select **Templates > Visual C# > Web**, select **ASP.NET Web Application**, and then click **OK**.

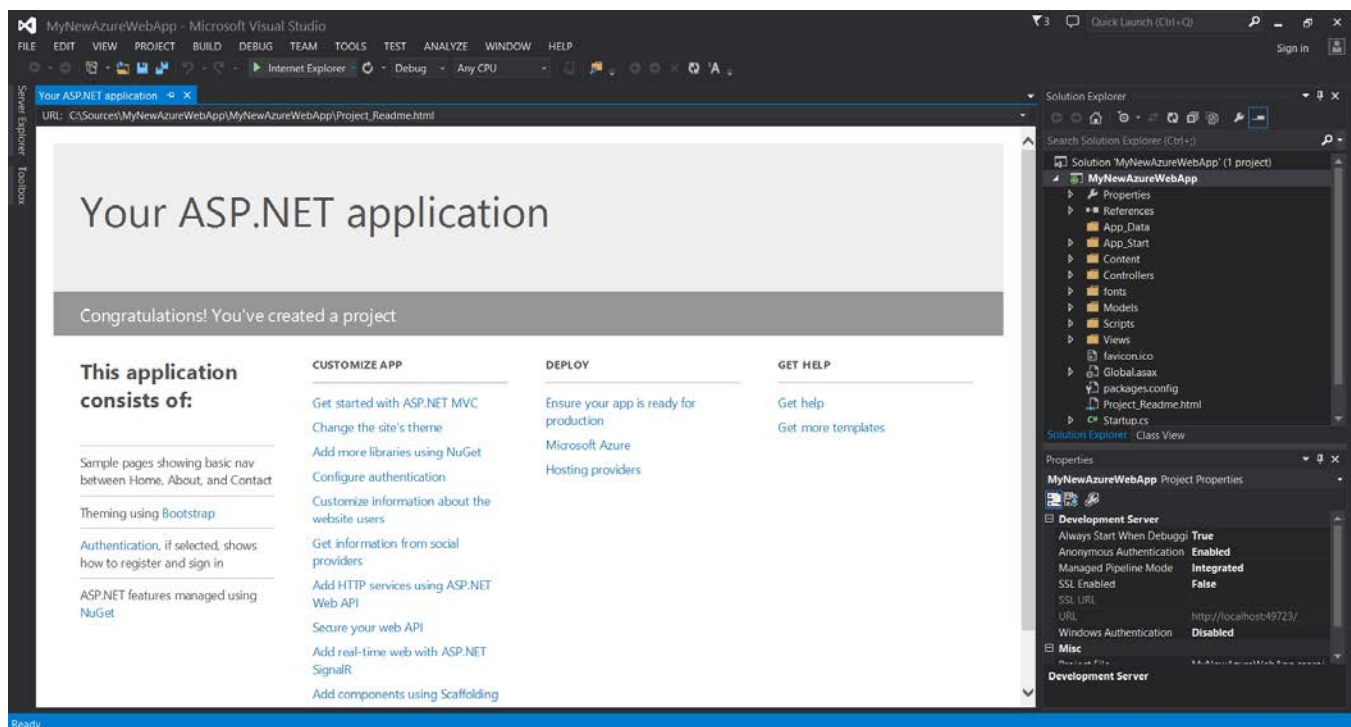


Creating the new Web app project in Visual Studio Community 2013.

This will open up the **New ASP.NET Project** window, and from here, select any template – we will demonstrate using the **MVC** (model-view-controller) architecture template. Uncheck **Microsoft Azure > Host in the Cloud** for now so we can demonstrate another method. Click on **OK**, and the project will open up in Visual Studio.

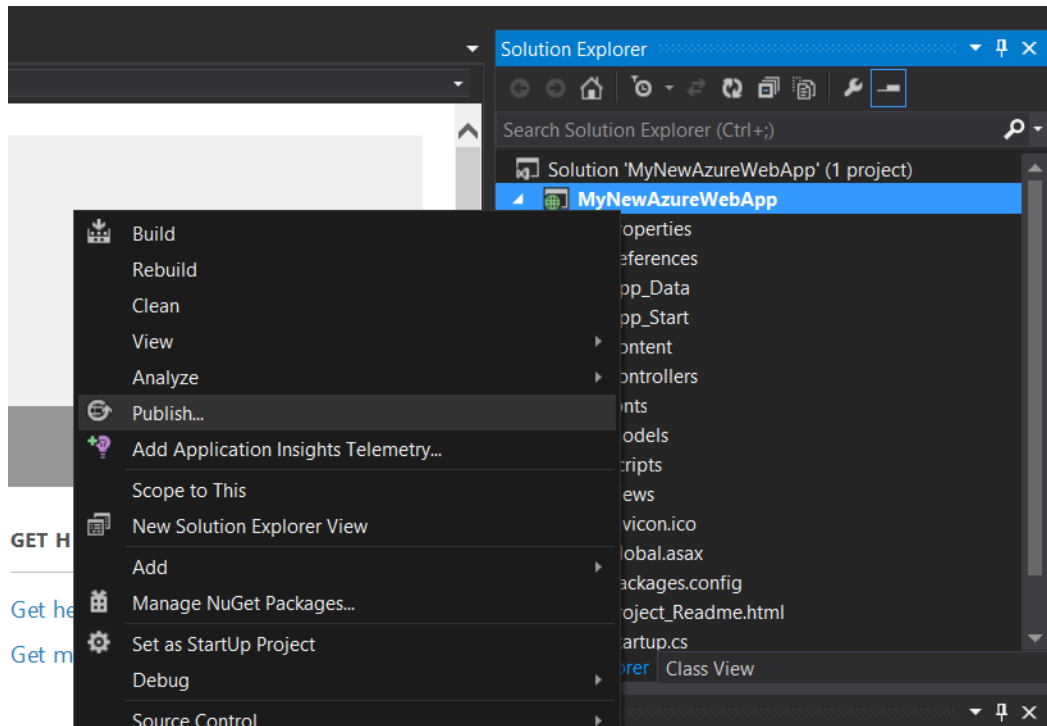


MVC new ASP.NET project window.



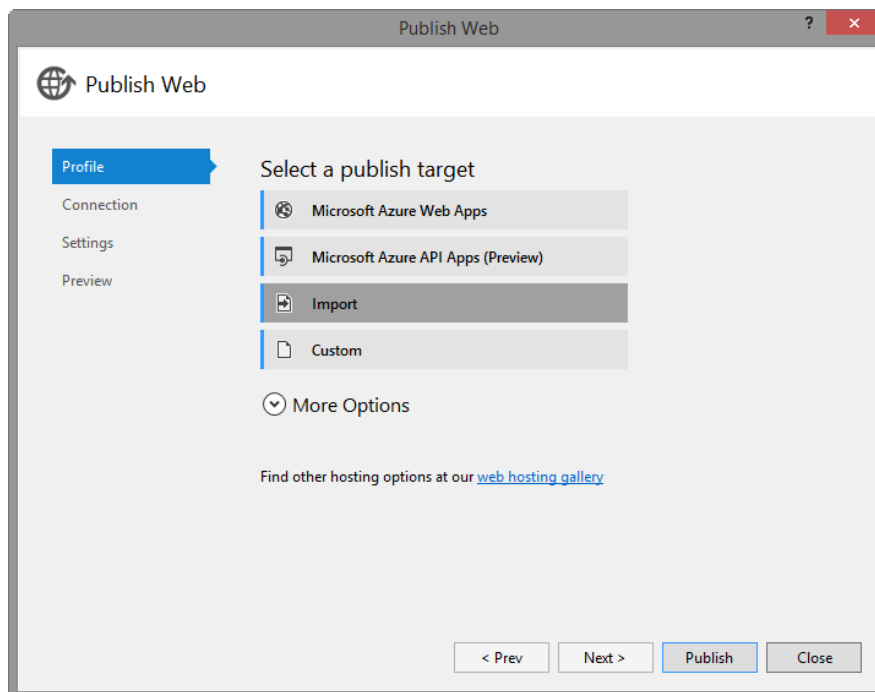
Main view of the new Web app project in Visual Studio.

In the **Solution Explorer**, go to the root node of the project and right-click on it (in our examples this is **MyNewAzureWebApp**) and then select **Publish** from the menu.



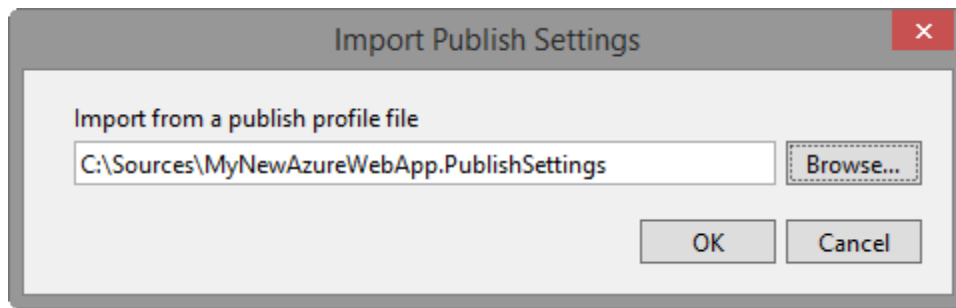
Right-click menu on the root node of the project.

The **Publish Web** dialog will open up, and under **Select a publish target**, choose **Import**.



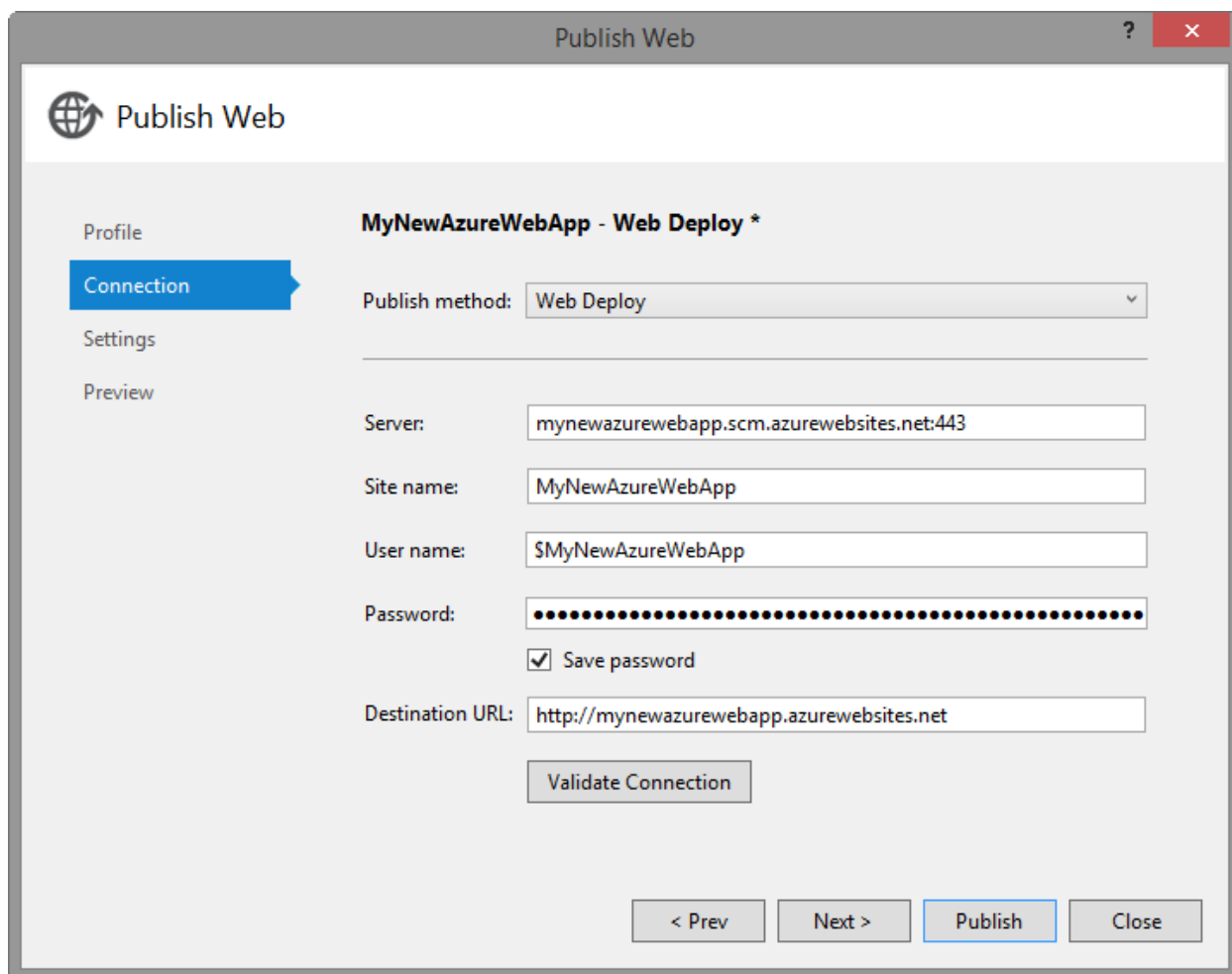
Publish web dialog.

Here's where we'll take advantage of the **PublishSettings** file we downloaded from the Azure Portal. When we Import, we'll enter the location of this file in the dialog, and then click OK.



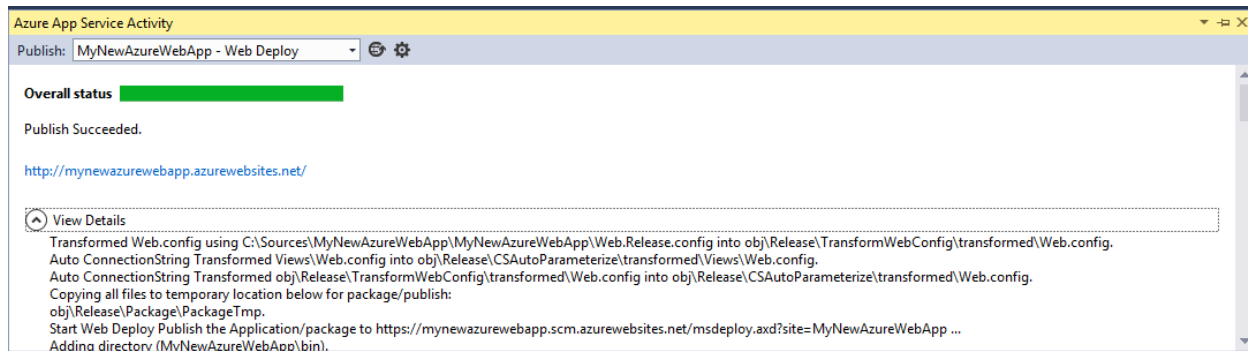
Import Publish Settings dialog – browse to wherever you saved the downloaded Azure Portal file.

Once that file is imported, the destination settings will appear in the **Web Publish** dialog.



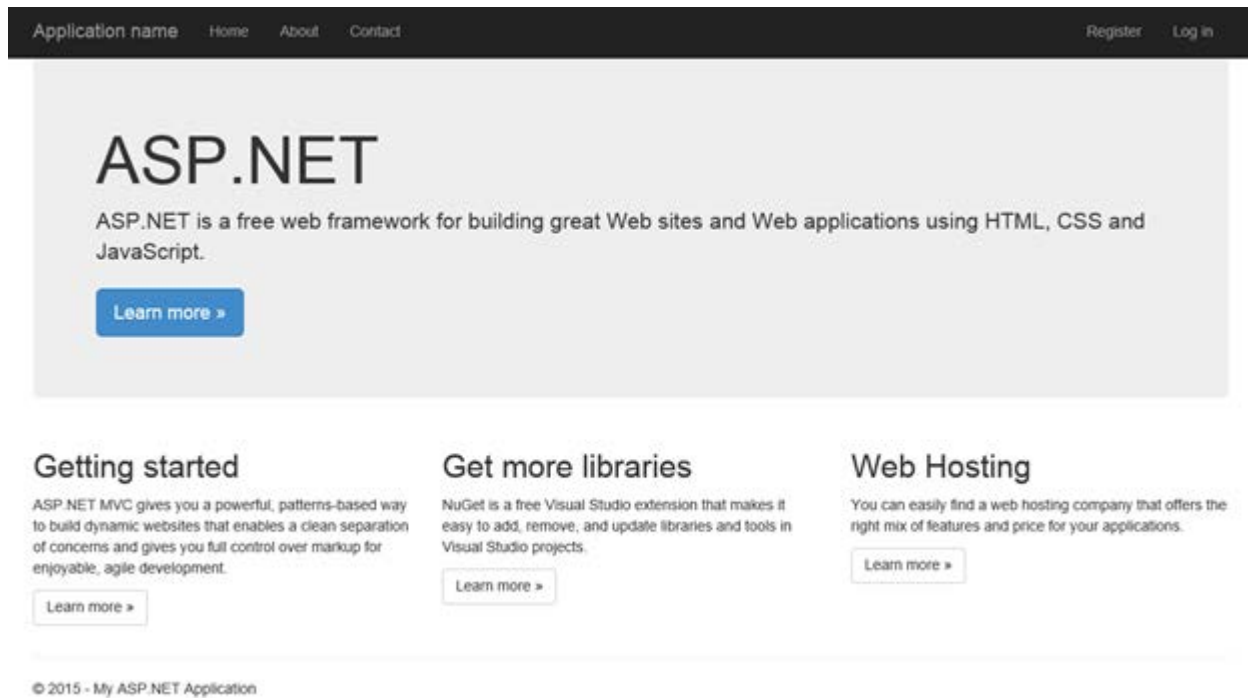
Imported publish settings are now present in the Publish Web dialog.

Now you can click on the **Publish** button to publish your ASP.NET Web App to Azure. The **Azure App Service Activity** window will provide status throughout the deployment.



Information about deployment activity via the Azure App Service Activity window.

After the code is deployed into the Azure cloud platform, the site will open up in your web browser.



Deployed web app visible in-browser.

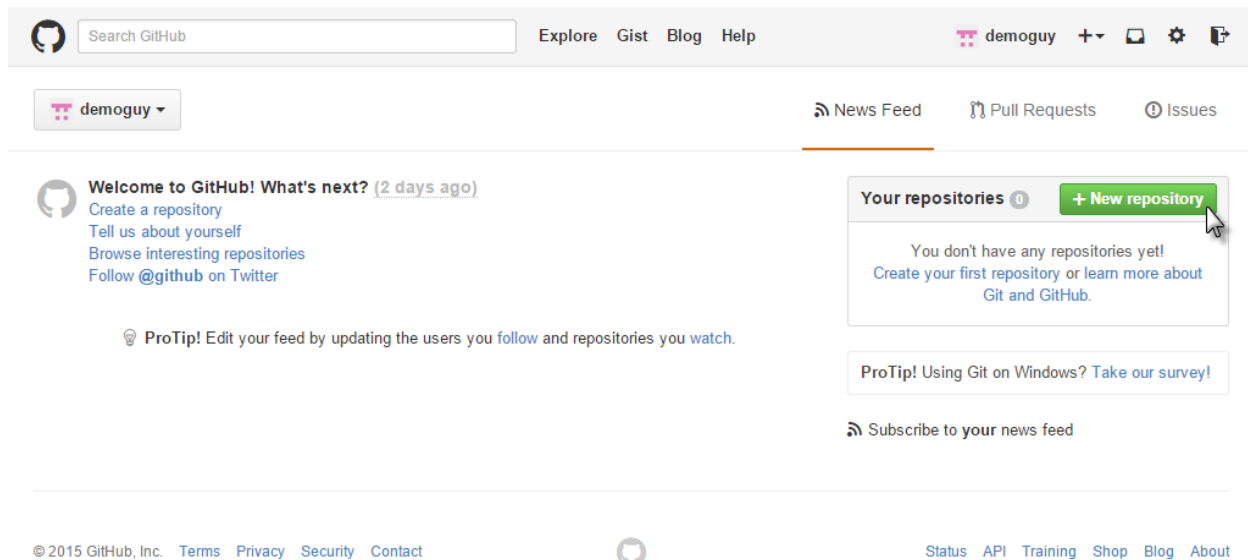
Enabling Continuous Integration with Azure Web Apps & GitHub

To keep your source code backed up and constantly up-to-date with your latest changes, you can take advantage of the **Azure** platform's continuous deployment with **GitHub**. Whenever you commit code to your GitHub repo, that code can be automatically synced up with your Azure Web app, without any extra effort. Here are the basic steps that we'll cover when first setting up continuous integration:

1. [Setting up a GitHub repository](#)
2. [Creating a new Web app in Azure](#)
3. [Setting up continuous integration](#)
4. [Updating and committing new code](#)

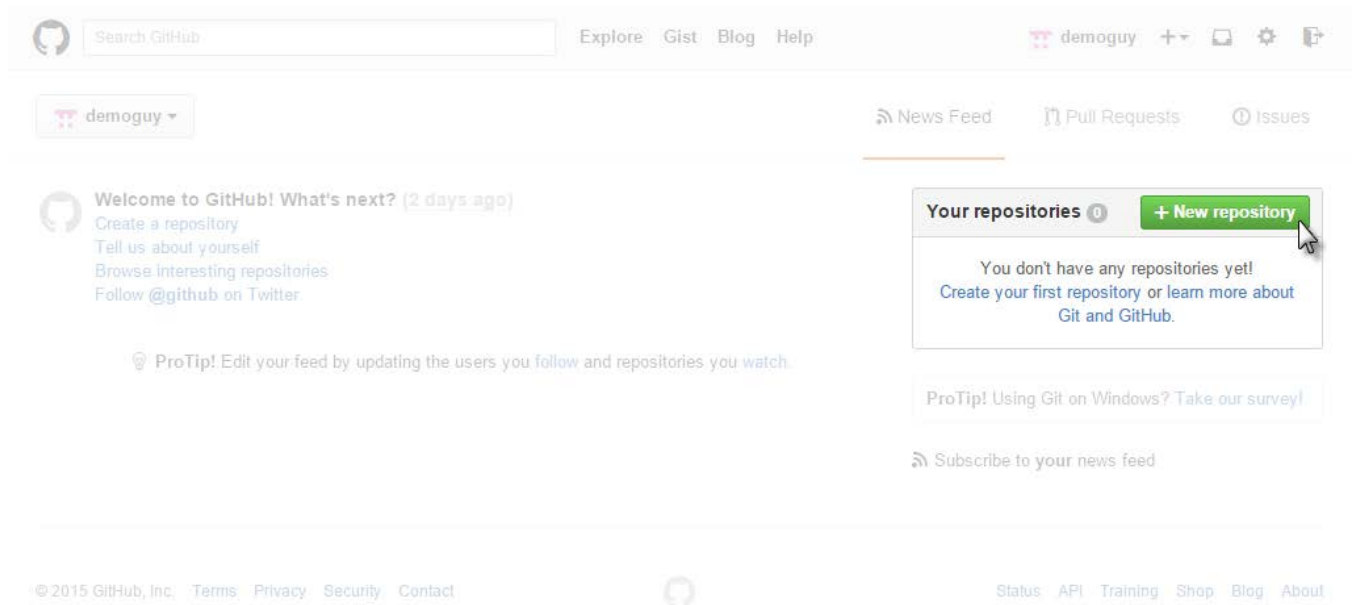
Step 1: Setting up a GitHub repository

If you don't currently have a GitHub account, you'll want to set one up and then log into it to follow the rest of this walkthrough. If you have a brand-new account, or one without any repositories, your browser will look like this:



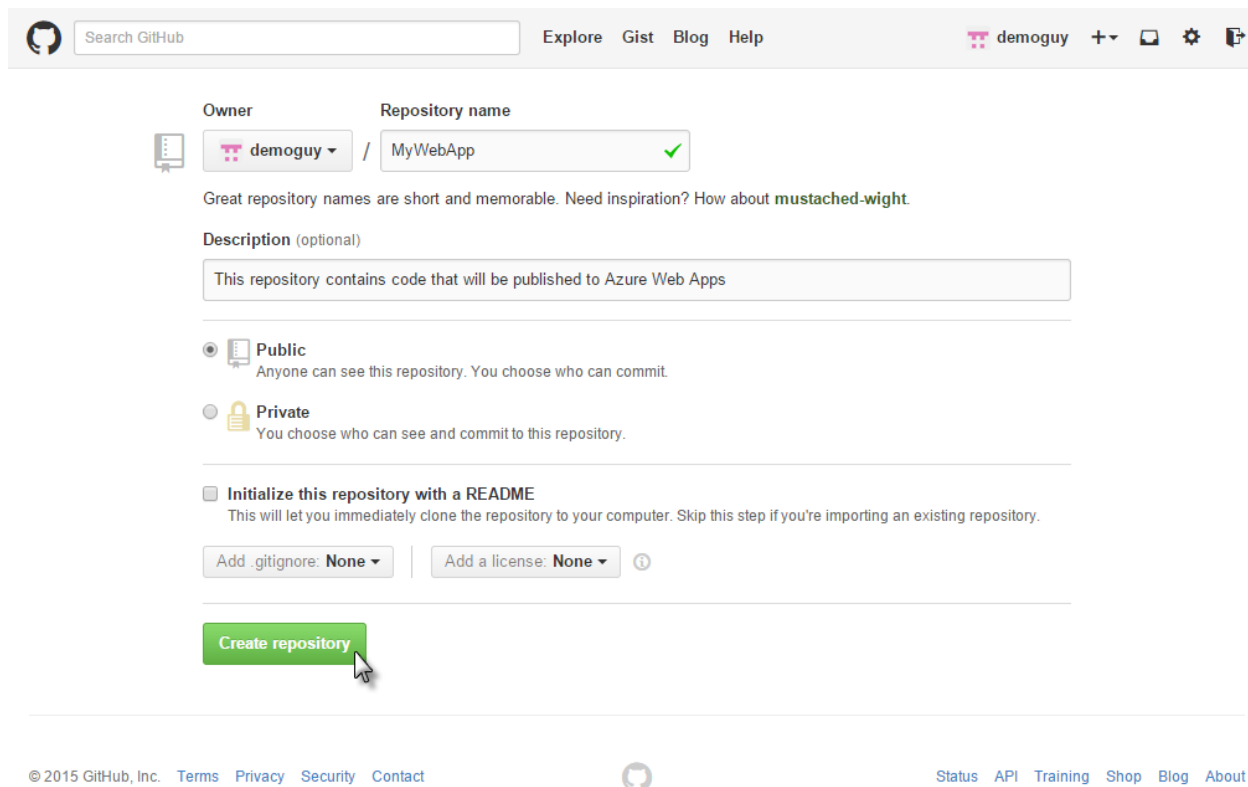
Initial screen for new GitHub accounts.

On this screen, you'll want to click on the **+ New Repository** button, which will create a new repository for your code in your GitHub account.



Green + New Repository button for creating new repos.

Once you've clicked the **+ New Repository** button, you'll see a new screen that allows you to make a **new Repository name**, a **description**, select either **public** or **private** settings, and whether or not to **initialize with a README** (we'll leave this **unchecked** for our walkthrough). Once you're satisfied with your repo's unique name and descriptive text, click on the **Create repository** button to create this new repo with no code in it.



GitHub repository setup screen and options.

The screenshot shows the GitHub interface for a newly created repository named 'MyWebApp' under the user 'demoguy'. The repository is empty, with 0 stars and 1 watch. The main content area provides instructions for setting up the repository. It includes a 'Quick setup' section with a button to 'Set up in Desktop' and a link to the repository URL 'https://github.com/demoguy/MyWebApp.git'. Below this, it offers instructions for creating a new repository on the command line, showing a series of git commands: 'git init', 'git add README.md', 'git commit -m "first commit"', 'git remote add origin https://github.com/demoguy/MyWebApp.git', and 'git push -u origin master'. It also provides instructions for pushing an existing repository from the command line and for importing code from another repository. On the right side, there is a sidebar with links to 'Code', 'Issues', 'Pull requests', 'Wiki', 'Pulse', 'Graphs', and 'Settings'.

Brand-new GitHub repository waiting for your code.

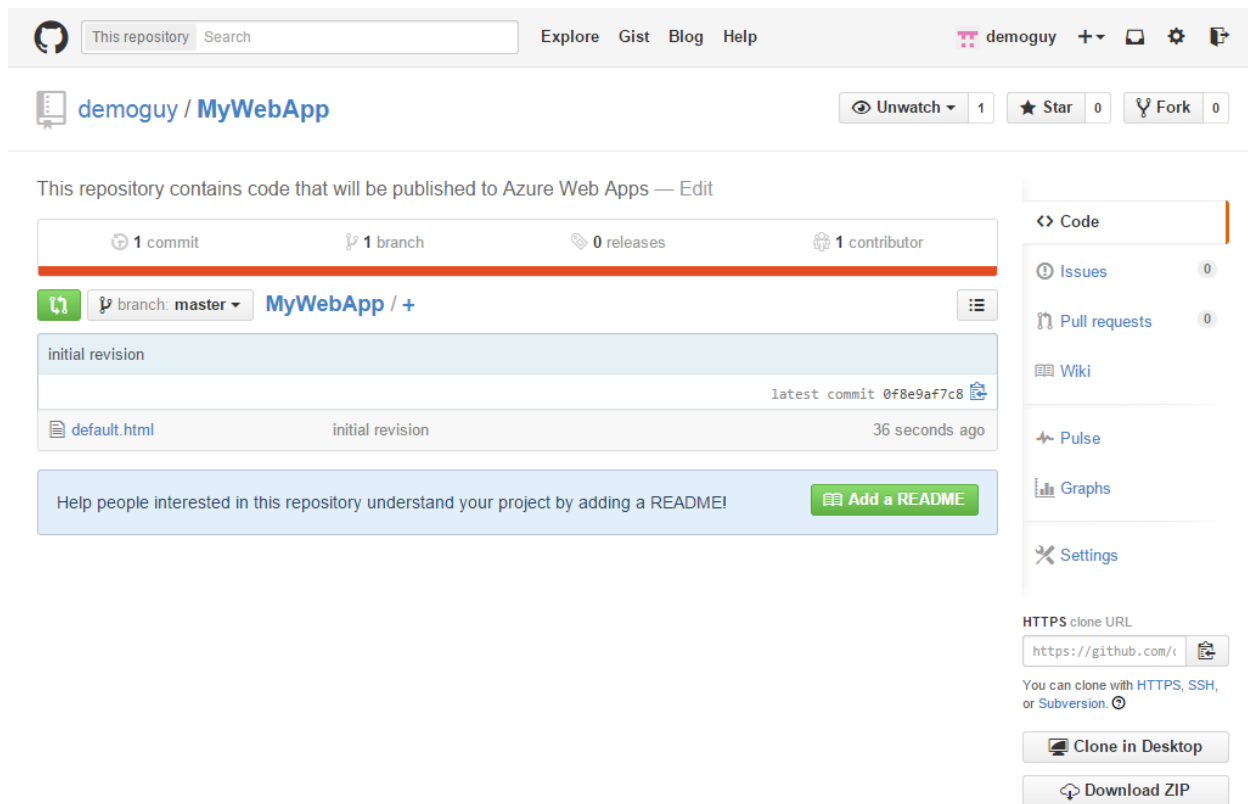
We want to make sure that this repository works, so you'll want to use your favorite **Git client** to commit code back to the repo you've created. There are many different Git clients available, such as [GitHub for Windows](#), **posh-git**, or **git bash**. Create or use any file that you'd like (and don't mind sharing publically) and then use your Git client to commit and push your file into the repository.

```

posh~git ~ MyWebApp [master]
C:\Sources\MyWebApp [master]> explorer .
C:\Sources\MyWebApp [master]> git add .
C:\Sources\MyWebApp [master +1 -0 -0]> git commit -m 'initial revision'
[master (root-commit) 0f8e9af] initial revision
 1 file changed, 14 insertions(+)
 create mode 100644 default.html
C:\Sources\MyWebApp [master]> git push origin master
Username for 'https://github.com': demoguy
Password for 'https://demoguy@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 437 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/demoguy/MyWebApp.git
 * [new branch]      master -> master
C:\Sources\MyWebApp [master]>
  
```

Example of using posh-git client command line to add a file.

Back in your web browser, refresh to see the newly-added file in your repository.

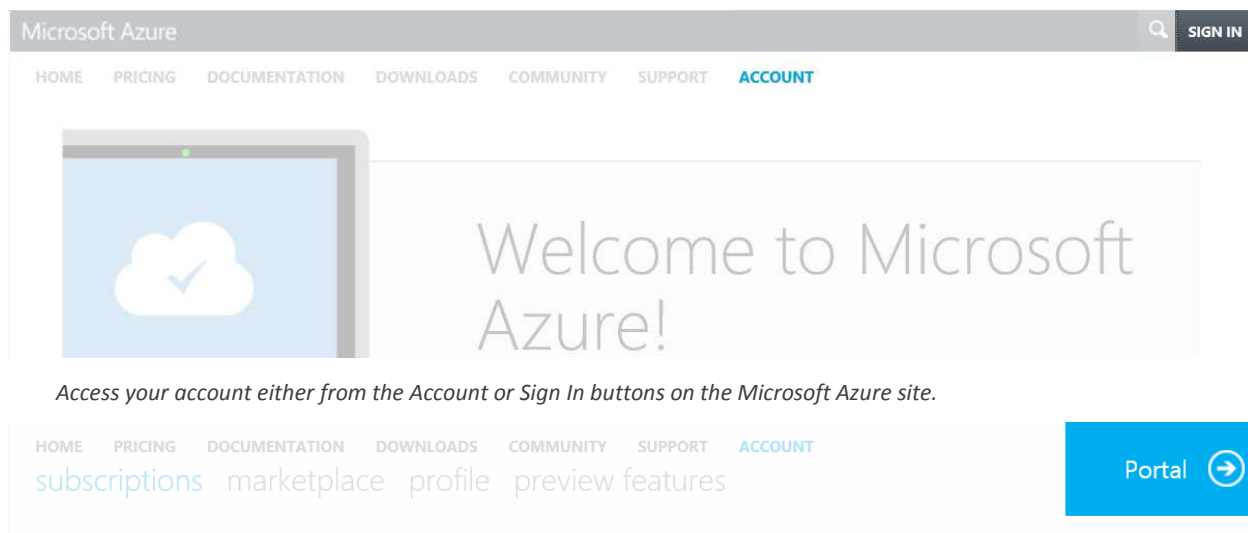


The `default.html` file added via the Git client is now visible in the repository.

Step 2: Creating a new Web app in Azure

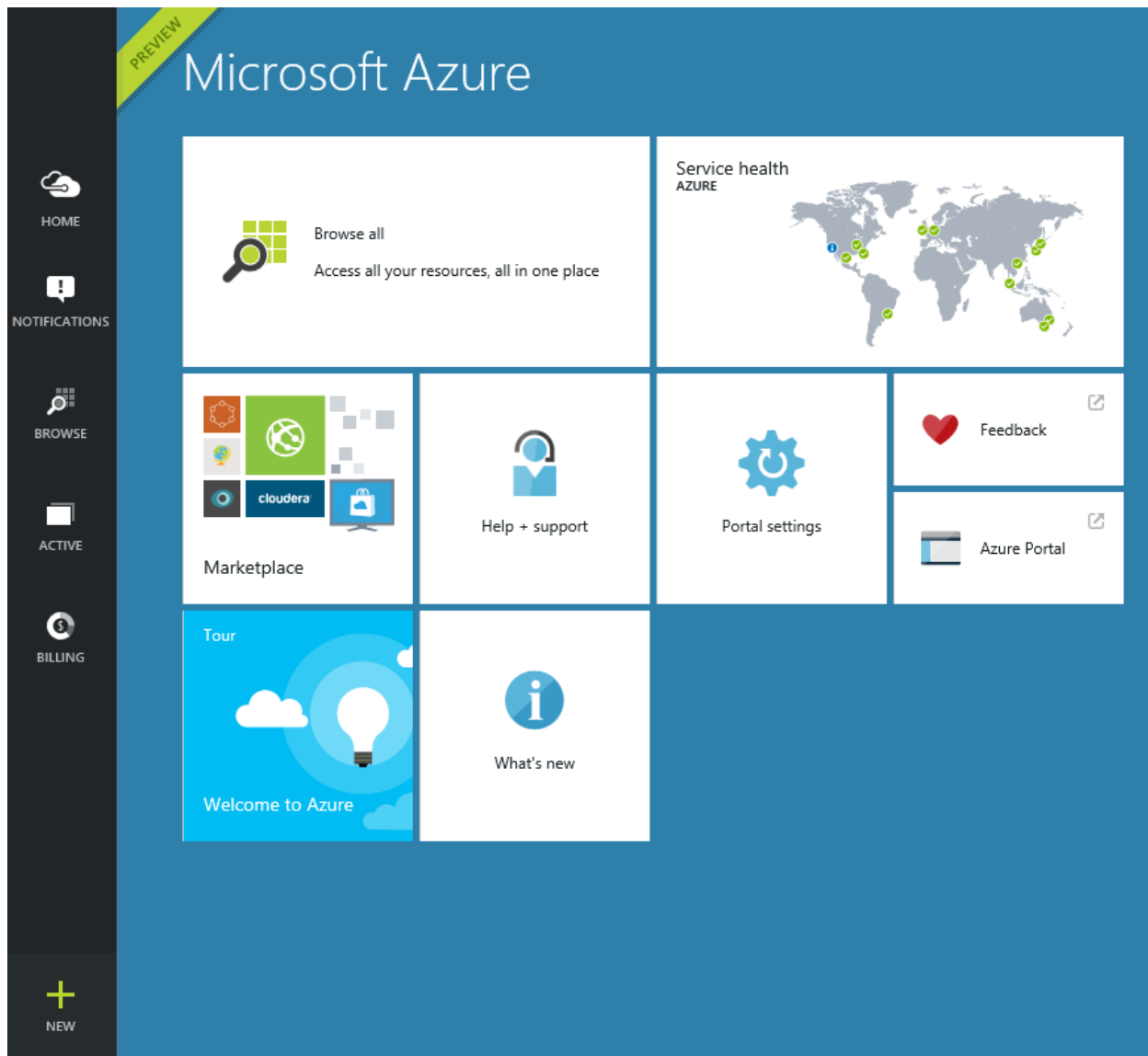
We've confirmed that our GitHub repository is working, since committing and pushing changes is confirmed, so now we work on creating our **Web app** in **Microsoft Azure**.

In your favorite browser, [open the Azure website](#), log in, and then open the Azure Portal.



Access your account either from the Account or Sign In buttons on the Microsoft Azure site.

Access the Azure Portal.

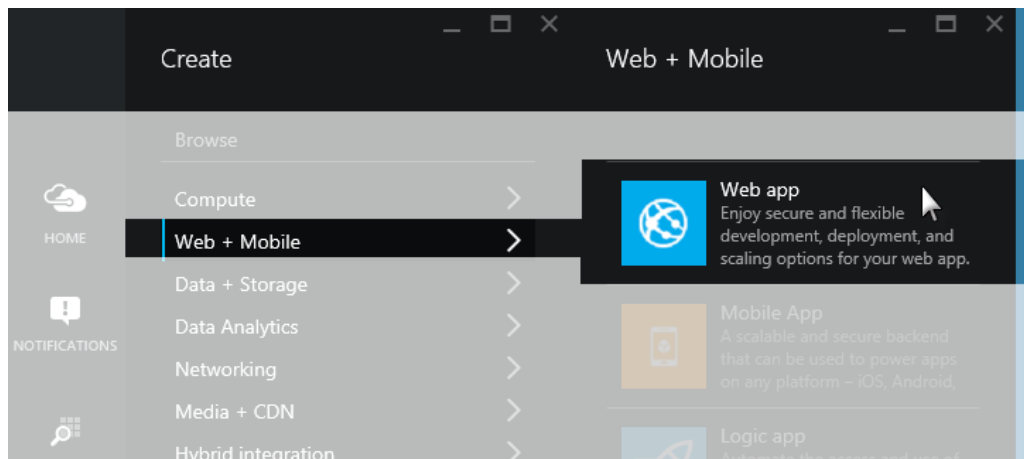


The main Azure Portal, a convenient dashboard for checking your Azure project status.



From the main **Azure Portal**, in the lower right-hand corner, click on **NEW** to start creating a new web app.

After selecting the **NEW** button, you'll be offered a menu full of different things you can create on Azure, but for now, focus on **Web+Mobile**, and then from the **Web+Mobile menu**, choose **Web app**.

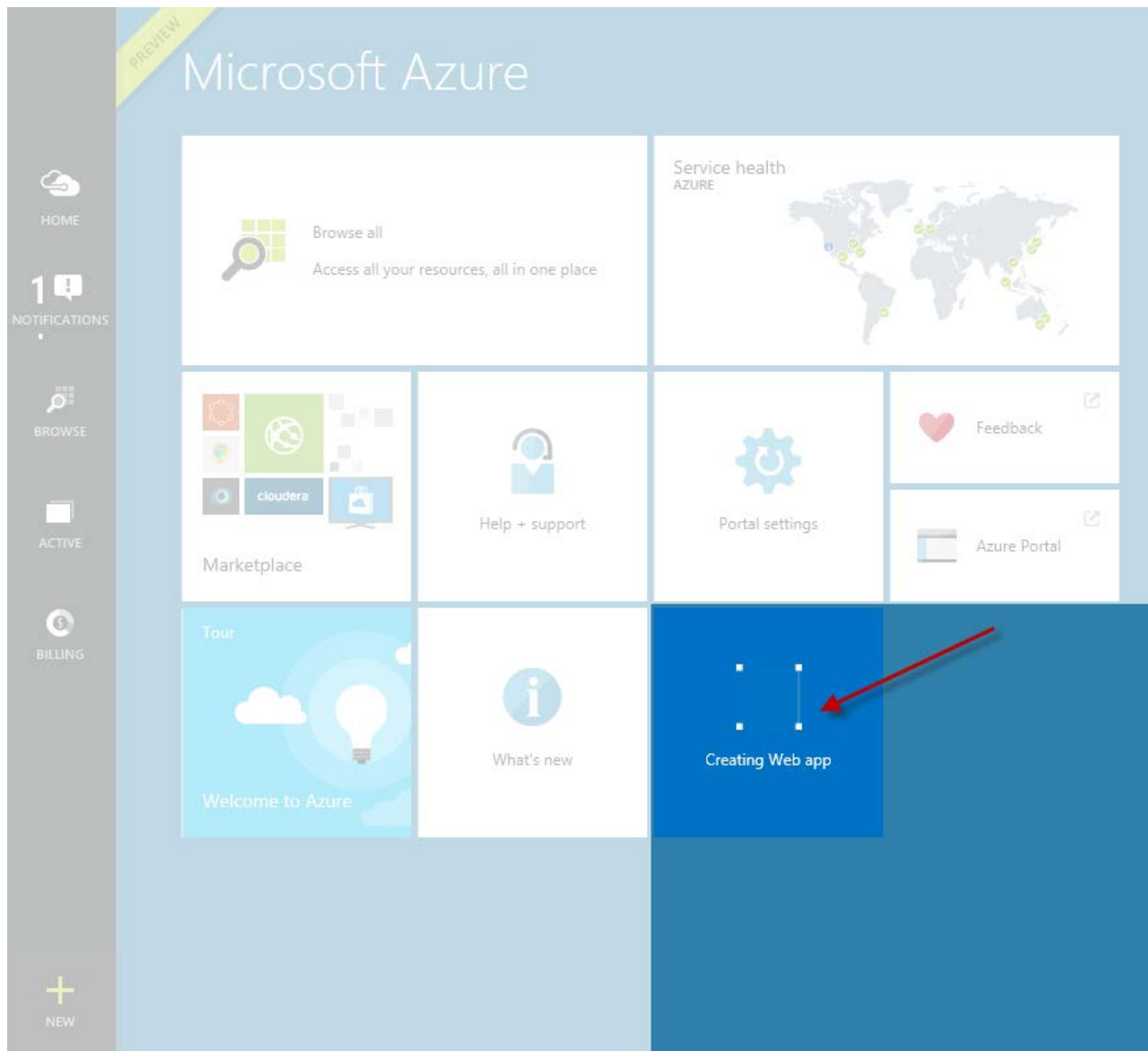


Menus for selecting and creating a new Web app.

Once you select **Web app**, a new menu will open up to the right side. You'll want to enter a **URL** for your project, and choose to name and create a **New AppService Plan**. Keep the pricing on **F1 Free** for now, and create a **New Resource Group**. Azure will let you know if names are taken, so keep adjusting your naming if it turns out a name you'd like is unavailable. Keep the **location** used for your cloud development on South Central US for now, though in future projects you'll be able to customize this if you'd like. Leave the checkbox for **Add to Startboard** checked, so that it's easier to access this work in the near future.

Once everything has been entered to your liking, hit the **Create** button.

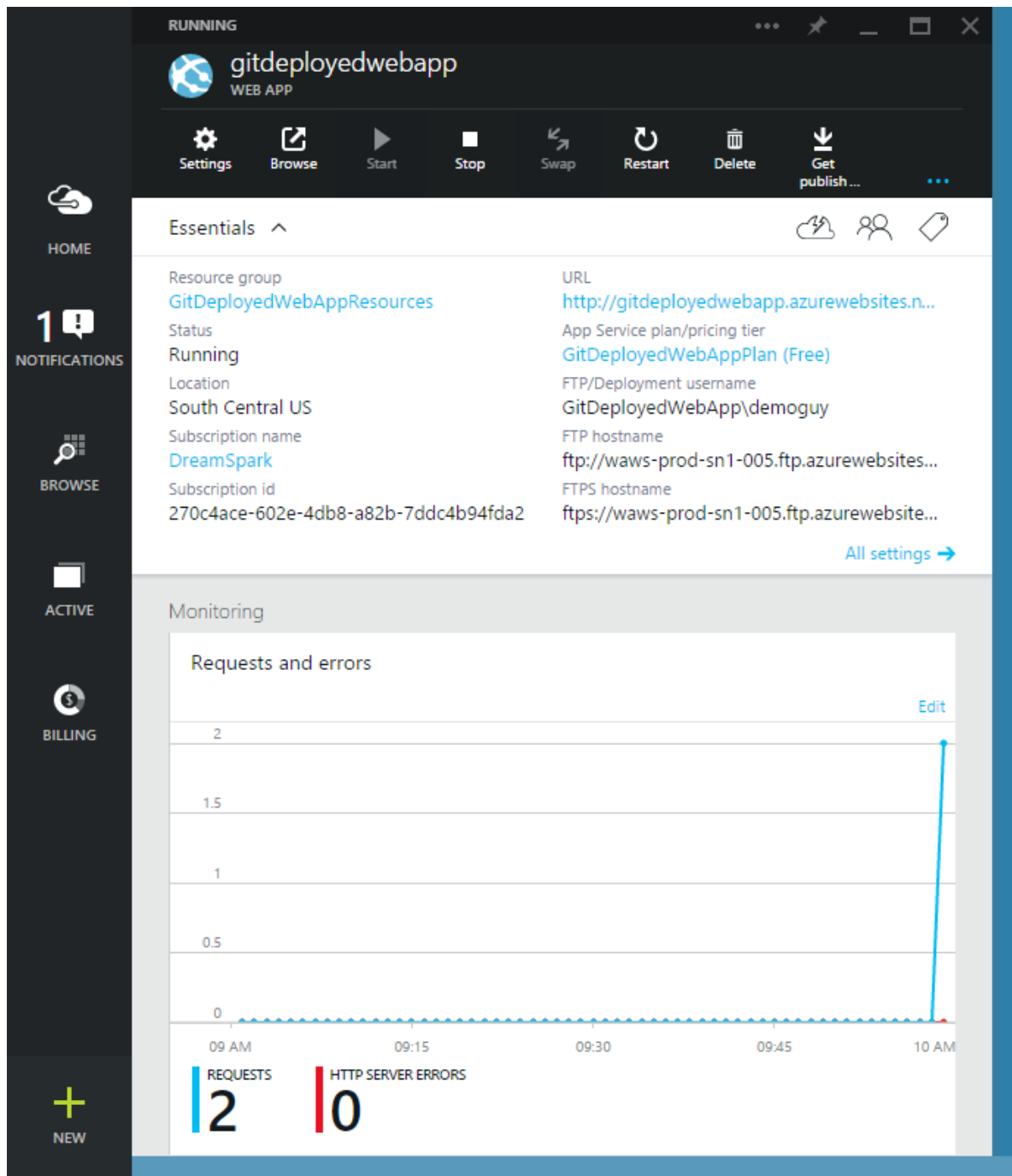
Completing the details for the new Web app.



New tile depicting Web app creation in-process.

Microsoft Azure will work on setting everything up and initial creation of your Web app, and then the tile will appear on your **Startboard** once it's ready.

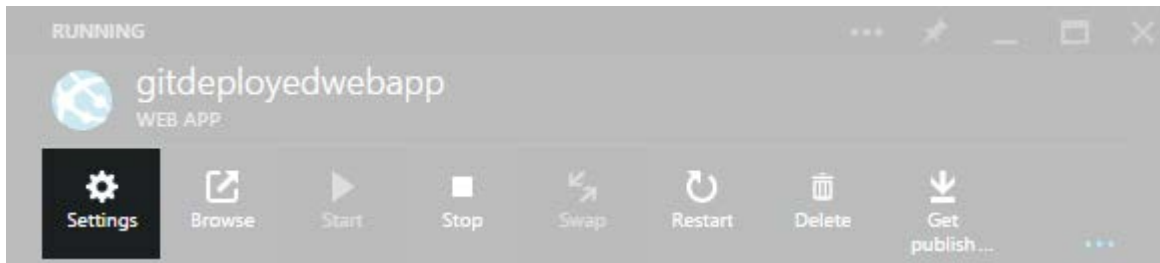
Click on your web app's tile to open up the related display, which gives you information about your web app's live performance.



Web app information blade.

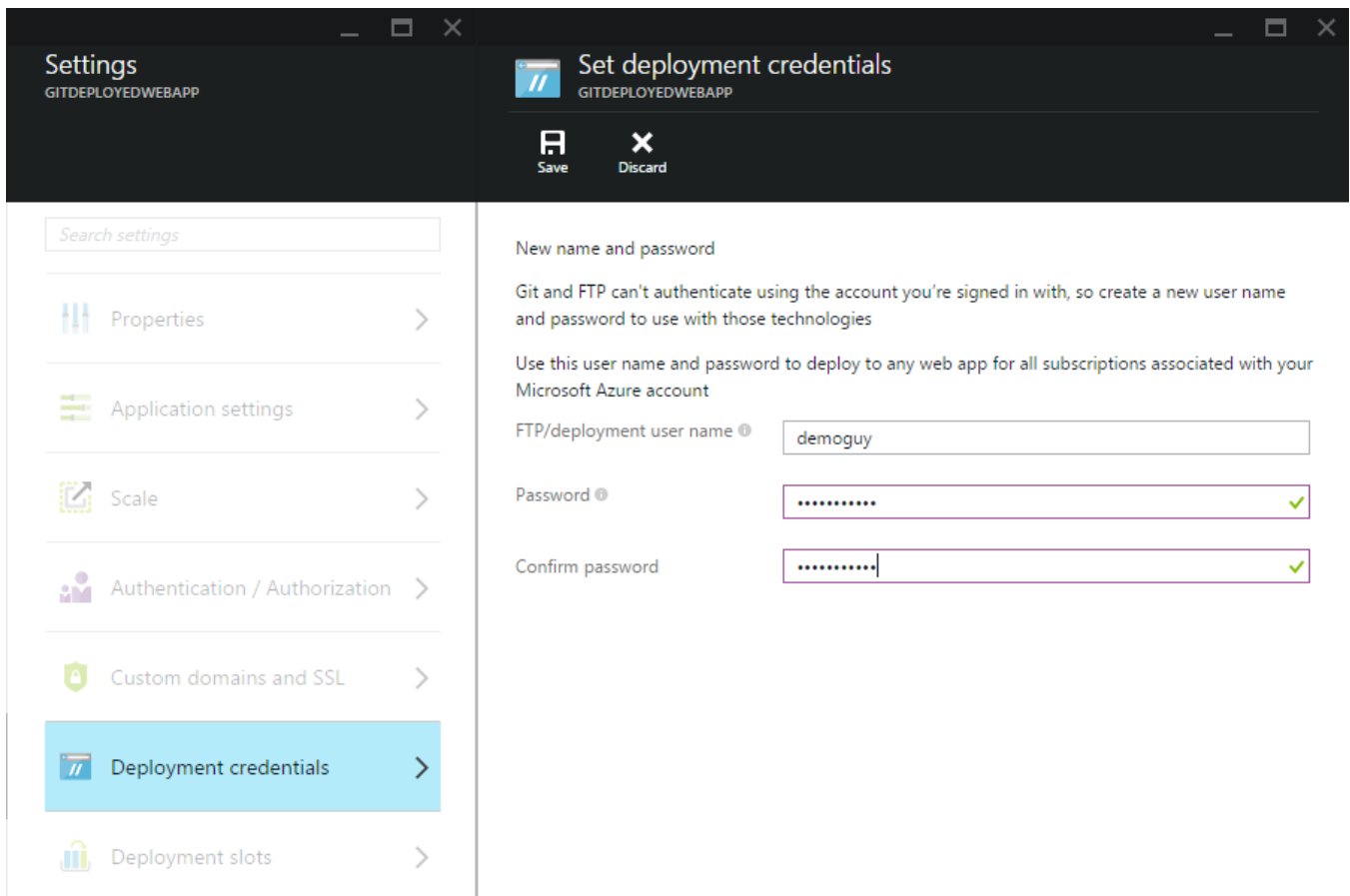
Step 3: Setting up a continuous integration

Now that the app is ready for your work, you'll want to connect it with your GitHub repository so that changes are synced across both. First thing is to go into **Settings**.



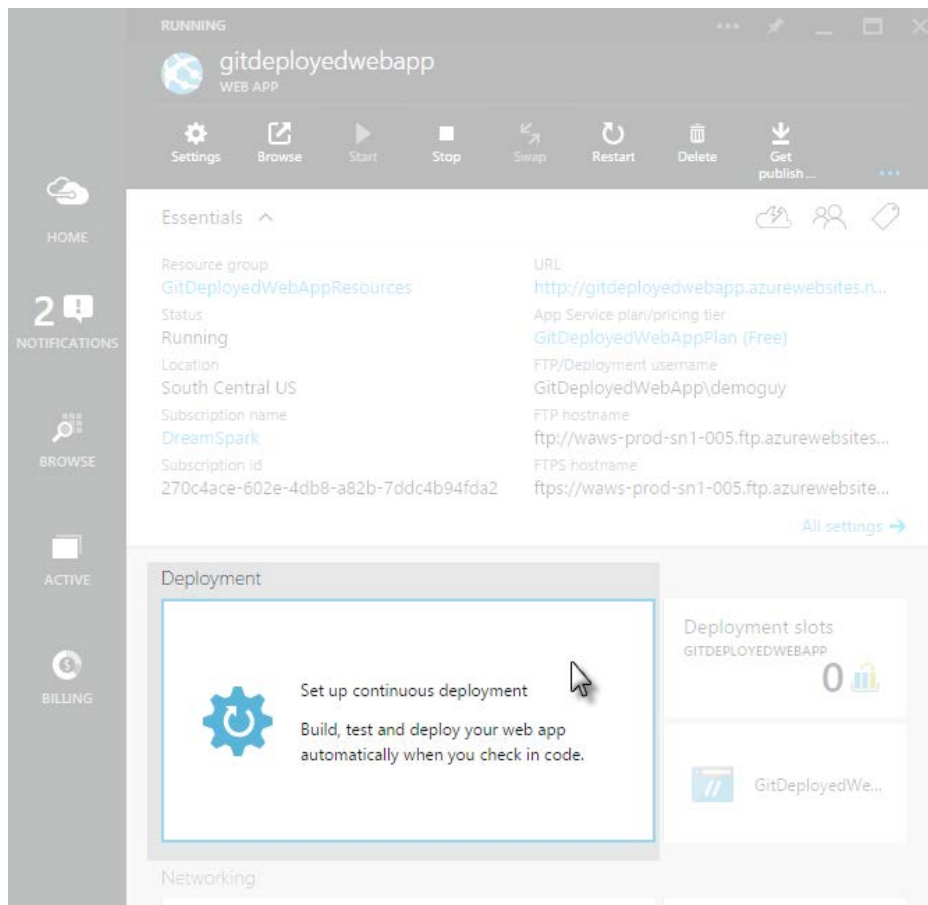
Settings menu icon available from the Web app's main page.

This opens up a new blade with all of the settings available, and we want to select **Deployment credentials**. This will open up another new blade next to the Settings blade, called **Set deployment credentials**. Fill in the appropriate **username** and **password** for your deployments to use.



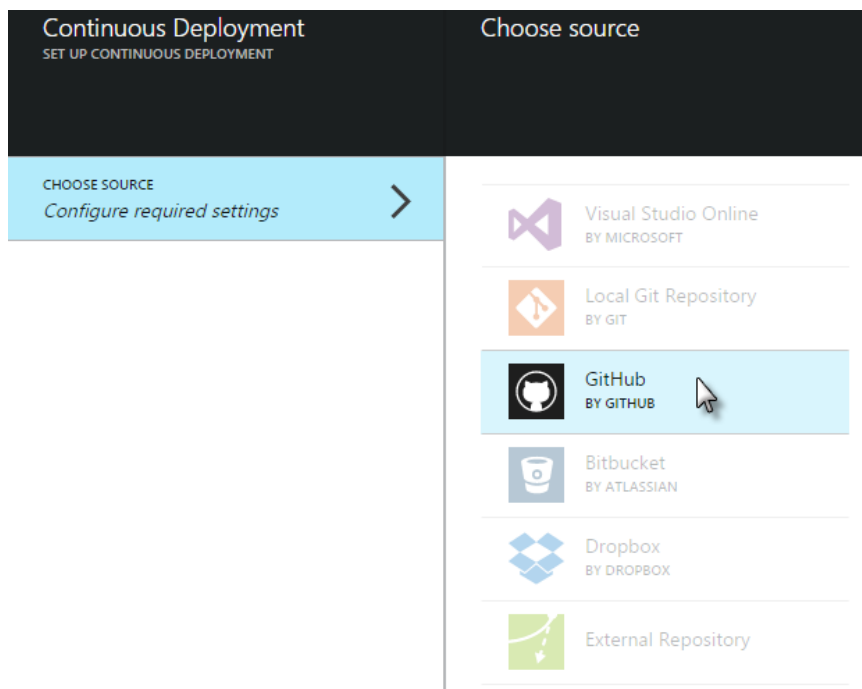
Blades for Deployment credentials-related settings.

Now that you've set that up, go back to the main blade for your web app and scroll down, far below the Monitoring graph. Towards the bottom, you'll find a section for **Deployment**, and a tile for **Set up continuous deployment**. Go ahead and click on that box and we'll work on the main setup for continuous integration with GitHub.



Click the *Set up continuous deployment* tile to proceed.

The **Continuous Deployment** menu blade will open up, so **Choose Source**, and then from that blade, select **GitHub** from the list of providers.

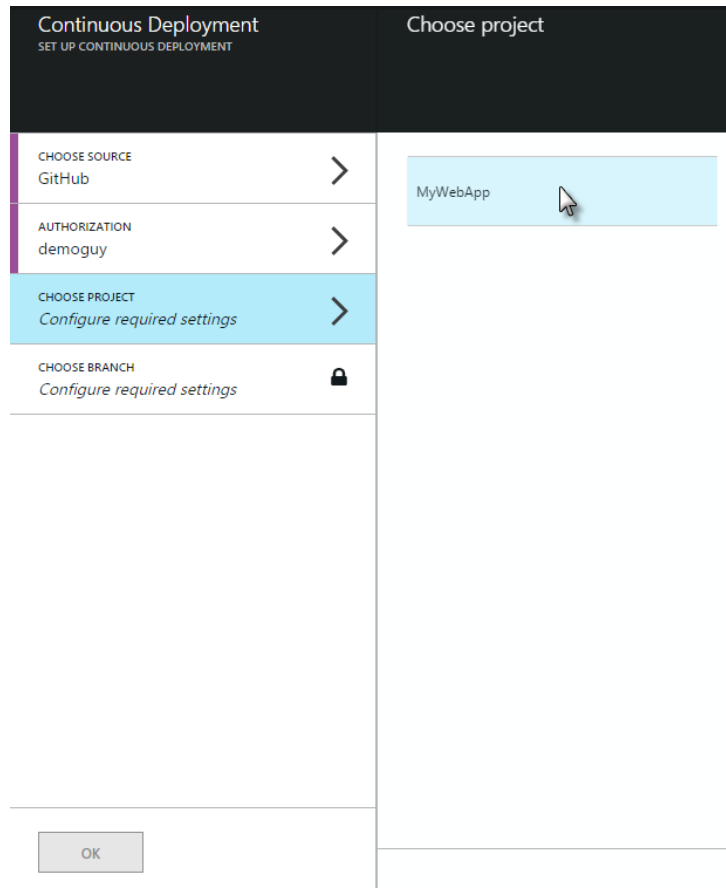


Select *GitHub* as your source for this Web app.

After you've selected GitHub as your source, you may be prompted to **authorize your GitHub account to enable the connection between GitHub and your Azure web app**. Log in with your GitHub credentials (same as those used to create your repository), and then once you've authorized the connection between Azure and GitHub, you'll be able to **select the repository** you want to integrate.

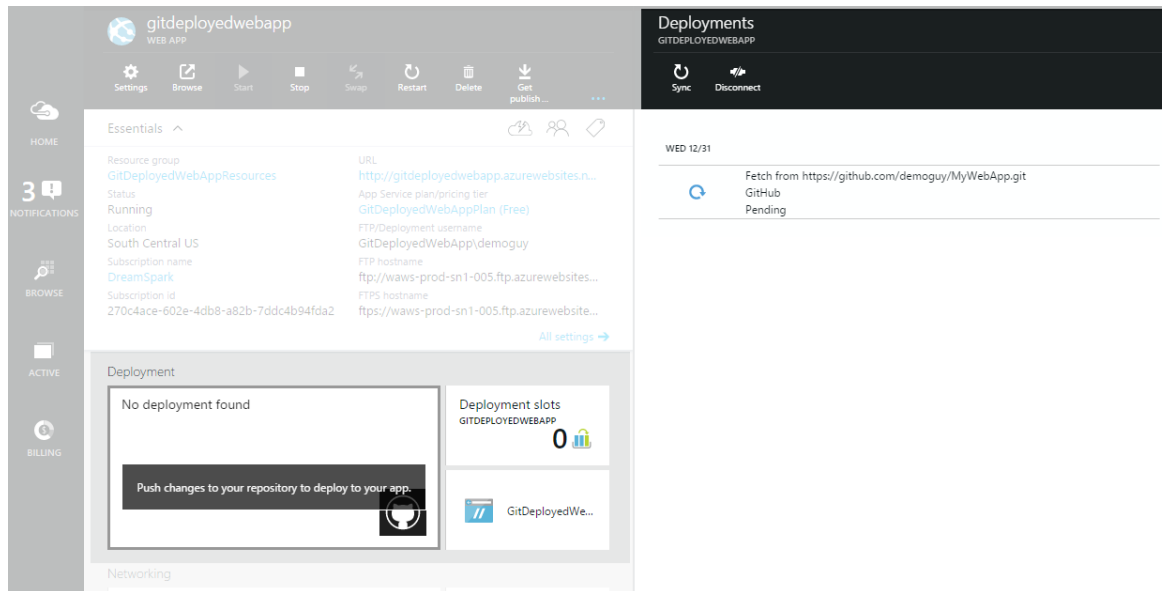
Your **Continuous Deployment** blade will update with the chosen **source** and **authorization** name, and then you will need to **choose the GitHub repository** to integrate with under **Choose Project**.

Once you've selected your repo, click the **OK** button in the **Continuous Deployment** blade.



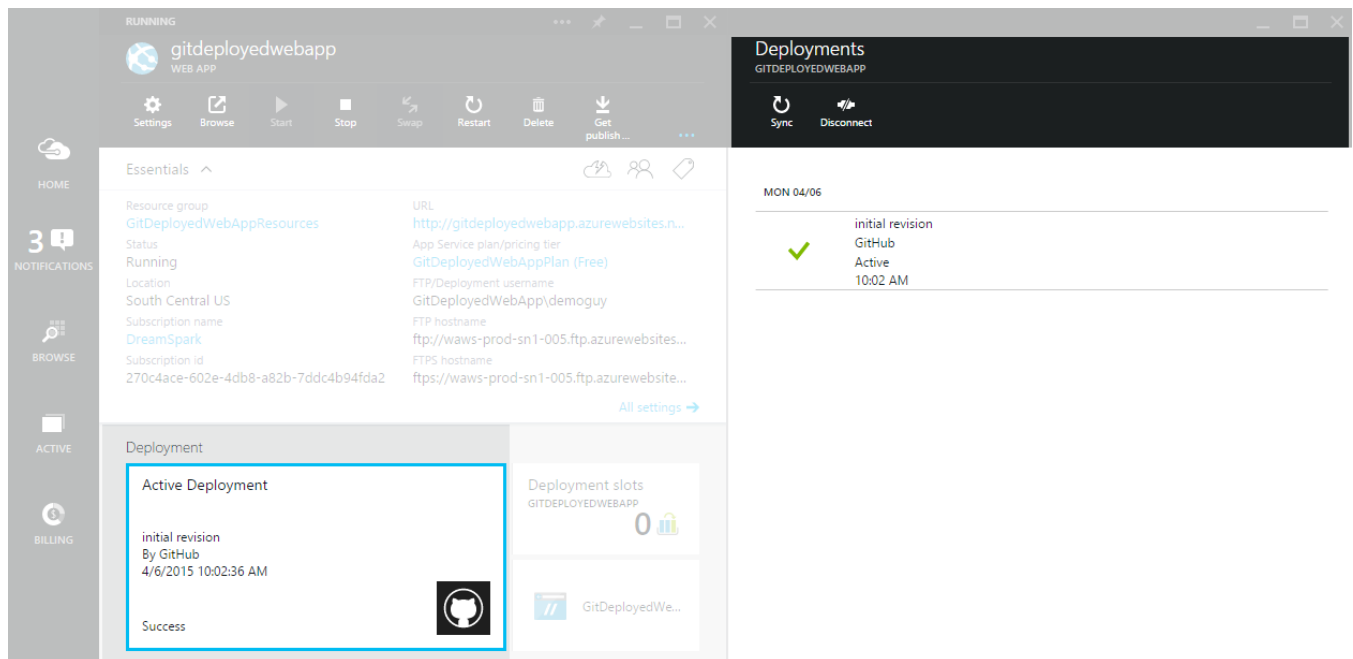
Select your repository from the Choose Project blade, click OK in Continuous Deployment.

After you click OK, your code is fetched from the specified branch in your GitHub repository and deployed to Azure automatically.



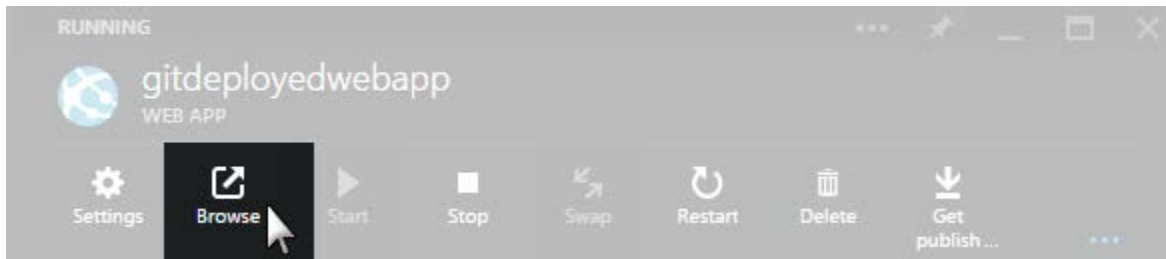
Fetching code in progress.

The deployment status will update on the Azure portal's **Deployments** blade as soon as the deployment completes, and then you can see the **Active Deployment** under the portal blade.

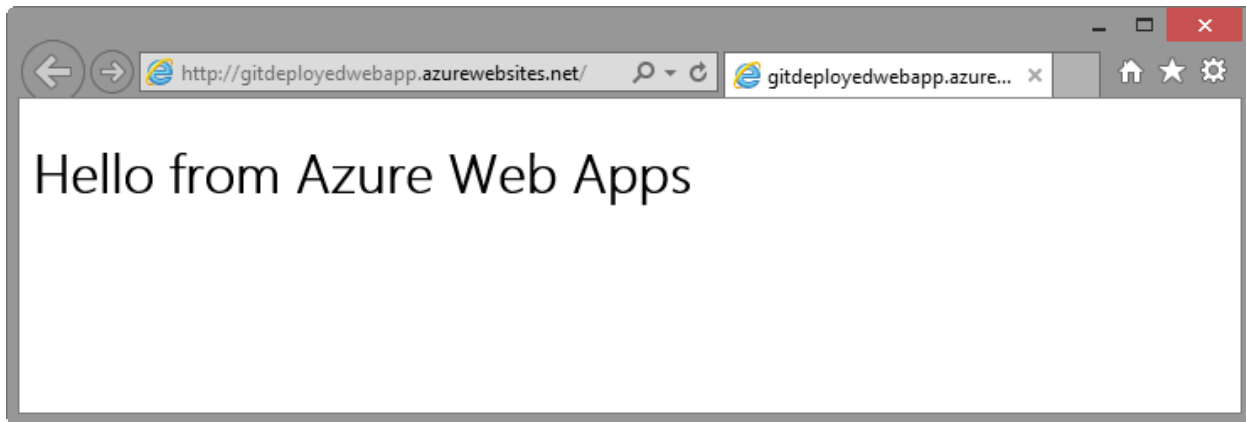


Successful code deployment, and updated Azure information.

Now click on the Browse button in the Web app blade's toolbar to see your code running in your Azure Web app, in your browser.



Click Browse to see your code in action.



Updated site visible in-browser.

Step 4: Updating and committing new code

Now that we've confirmed successful deployment of your code, we can work on updating it and confirming that those changes are pushed forward.

Open the GitHub repository where you committed your code, and then open one of those committed files using the **Edit this file** toolbar icon.



In the upper-right corner, click on Edit this file to start making changes on GitHub.

In our example, we'll add a new paragraph via GitHub, below our header.



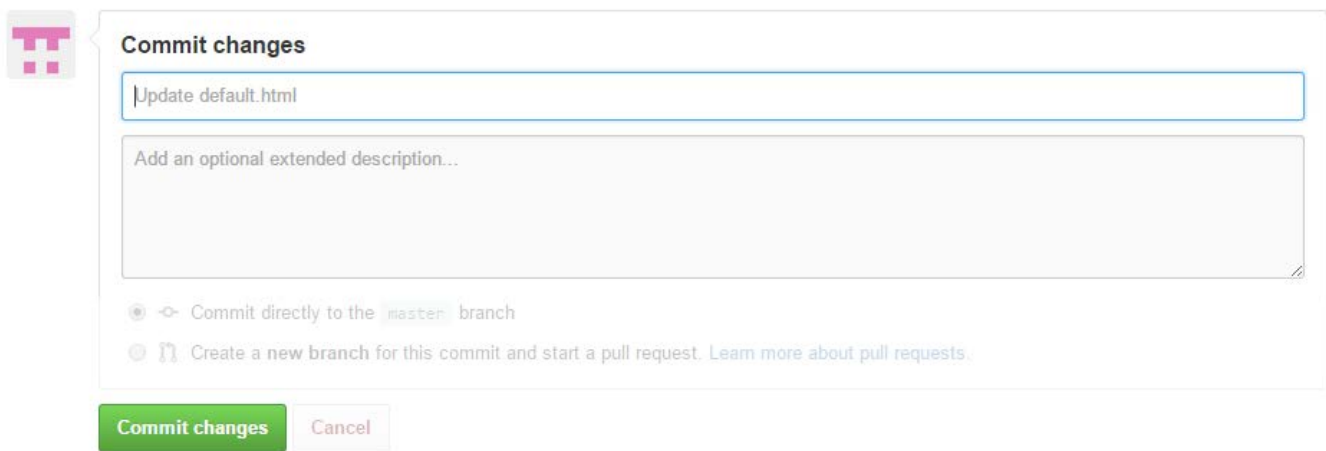
The screenshot shows a code editor with two tabs: 'Edit file' and 'Preview changes'. The 'Edit file' tab is active, displaying an HTML document. The document structure is as follows:

```
1 <!DOCTYPE html>
2
3 <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5   <meta charset="utf-8" />
6   <title></title>
7   <style type="text/css">
8     body { font-family: "Segoe UI Light";
9   </style>
10 </head>
11 <body>
12   <h1>Hello from Azure Web Apps</h1>
13   <p>This is a paragraph of text I added live in GitHub.</p>
14 </body>
15 </html>
```

The new paragraph on line 13 is highlighted with a grey background.

New paragraph added to the file on GitHub.

Once you're satisfied with your code updates, you'll want to consider adding a **commit message** explaining the changes. Below the editor area, there's a **Commit changes** section where coders can either take the default message, or provide a more detailed explanation to help give context to the changes. When you finish updating the commit message, go ahead and click on the **Commit changes button** to save your work to the repository.



The screenshot shows the 'Commit changes' section on GitHub. It features a text input field with the default message 'Update default.html'. Below this is a larger text area for an optional extended description. At the bottom, there are two radio buttons: 'Commit directly to the master branch' (selected) and 'Create a new branch for this commit and start a pull request'. A green 'Commit changes' button and a grey 'Cancel' button are at the bottom.

Main commit changes section on GitHub.

Once your code is committed to your repo, **continuous integration** kicks in and the code on GitHub will be pulled into your Azure Web app and **redeployed automatically**. If you open up and watch the Azure Portal again after your GitHub commit, the change is quickly pulled in and **deployed to your live site**.

The screenshot shows the Azure portal interface for a web app named 'gitdeployedwebapp'. The left sidebar contains navigation links: HOME, NOTIFICATIONS (3), BROWSE, ACTIVE, and BILLING. The main content area is divided into two sections: 'Essentials' and 'Deployments'.

Essentials:

- Resource group: GitDeployedWebAppResources
- Status: Running
- Location: South Central US
- Subscription name: DreamSpark
- Subscription id: 270c4ace-602e-4db8-a82b-7ddc4b94da2
- URL: http://gitdeployedwebapp.azurewebsites.net/
- App Service plan/pricing tier: GitDeployedWebAppPlan (Free)
- FTP/Deployment username: GitDeployedWebApp\demoguy
- FTP hostname: ftp://waws-prod-snl-005.ftp.azurewebsites.net
- FTPS hostname: ftps://waws-prod-snl-005.ftp.azurewebsites.net

Deployments:

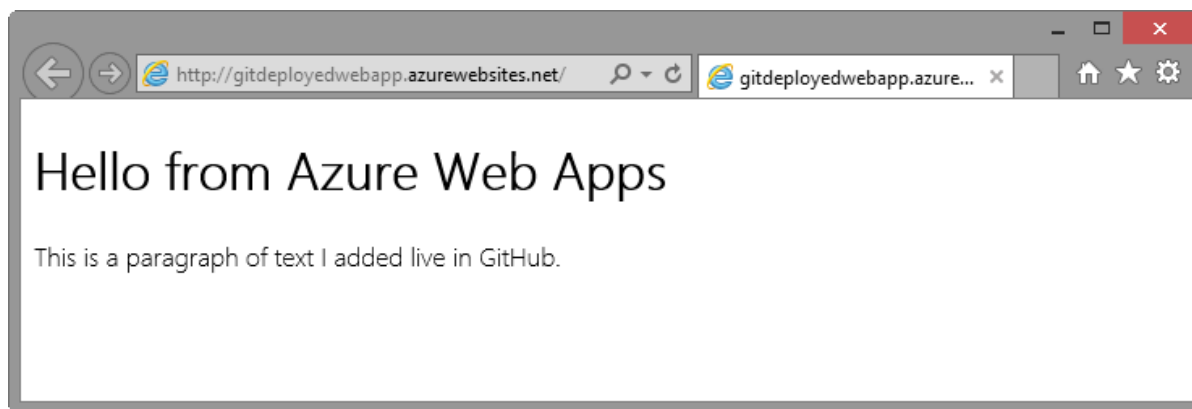
MON 04/06

Deployment	Provider	Status	Time
Update default.html	GitHub	Active	10:06 AM
Initial revision	GitHub	Inactive	10:02 AM

The 'Active Deployment' section shows a successful deployment of 'Update default.html' by GitHub at 10:06:22 AM on 4/6/2015. The deployment slots section shows 0 slots available.

Latest changes appear in Azure after GitHub repository's code is updated.

Now you can refresh your browser to see your changes live on your Azure web app.



Updated version of the Web app in-browser.

Publish a Web App to Azure using FTP

You can also **publish to Azure via FTP client**, so these instructions assume you've already prepared your Web app to your liking and have downloaded your Publish Settings from Azure. (See **Publish a Web App to Azure using Visual Studio, Downloading a publish settings file** for more information.)

We'll take a moment to use a text editor to open up the Publish Settings file and look for key information. In this walkthrough, we'll use a simulated publish settings file to demonstrate what to look for as a guide for when you look over your unique publish settings file.

```
<publishData><publishProfile profileName="A_NAME - Web Deploy" publishMethod="MSDeploy" publishUrl="A_NAME.scm.azurewebsites.net:443"
msdeploySite="A_NAME" userName="$A_NAME" userPWD="A_PASSWORD" destinationAppUrl="http://A_NAME.azurewebsites.net"
SQLServerDBConnectionString="" mySQLDBConnectionString="" hostingProviderForumLink="" controlPanelLink="http://windows.azure.com"
webSystem="WebSites"><databases /></publishProfile><publishProfile profileName="A_NAME - FTP" publishMethod="FTP"
publishUrl="ftp://A_SITE.net/site/wwwroot" ftpPassiveMode="True" userName="A_NAME\A_NAME" userPWD="A_PASSWORD"
destinationAppUrl="http://A_NAME.azurewebsites.net" SQLServerDBConnectionString="" mySQLDBConnectionString=""
hostingProviderForumLink="" controlPanelLink="http://windows.azure.com" webSystem="WebSites"><databases /></publishProfile></publishData>
```

Simulated publish settings file, as an example.

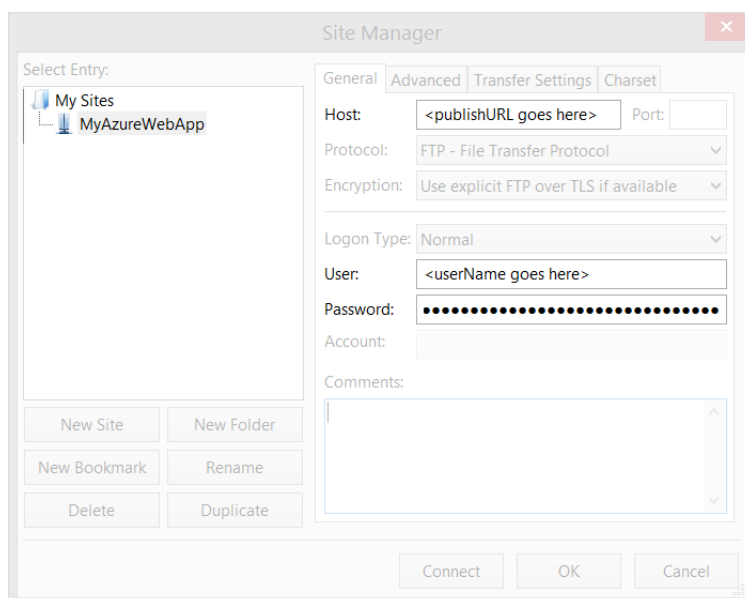
Looking over the text present, we want to focus on the **publish profile** that features **publishMethod="FTP"**. There are three key pieces of information you'll want to pull out:

1. **publishUrl**
2. **username**
3. **userPWD**

```
<publishData><publishProfile profileName="A_NAME - Web Deploy" publishMethod="MSDeploy" publishUrl="A_NAME.scm.azurewebsites.net:443"
msdeploySite="A_NAME" userName="$A_NAME" userPWD="A_PASSWORD" destinationAppUrl="http://A_NAME.azurewebsites.net"
SQLServerDBConnectionString="" mySQLDBConnectionString="" hostingProviderForumLink="" controlPanelLink="http://windows.azure.com"
webSystem="WebSites"><databases /></publishProfile><publishProfile profileName="A_NAME - FTP" publishMethod="FTP"
publishUrl="ftp://A_SITE.net/site/wwwroot" ftpPassiveMode="True" userName="A_NAME\A_NAME" userPWD="A_PASSWORD"
destinationAppUrl="http://A_NAME.azurewebsites.net" SQLServerDBConnectionString="" mySQLDBConnectionString=""
hostingProviderForumLink="" controlPanelLink="http://windows.azure.com" webSystem="WebSites"><databases /></publishProfile></publishData>
```

Key information for FTP in publish settings file, profileName and publishMethod highlighted for recognizability as well.

Now we can use this information in a FTP client to establish that connection to the Azure platform. In a FTP client such as FileZilla (for example, any will work), you'll simply copy-paste these three pieces of information into their respective fields.



FileZilla FTP client, highlighting the key areas to place publish settings information.