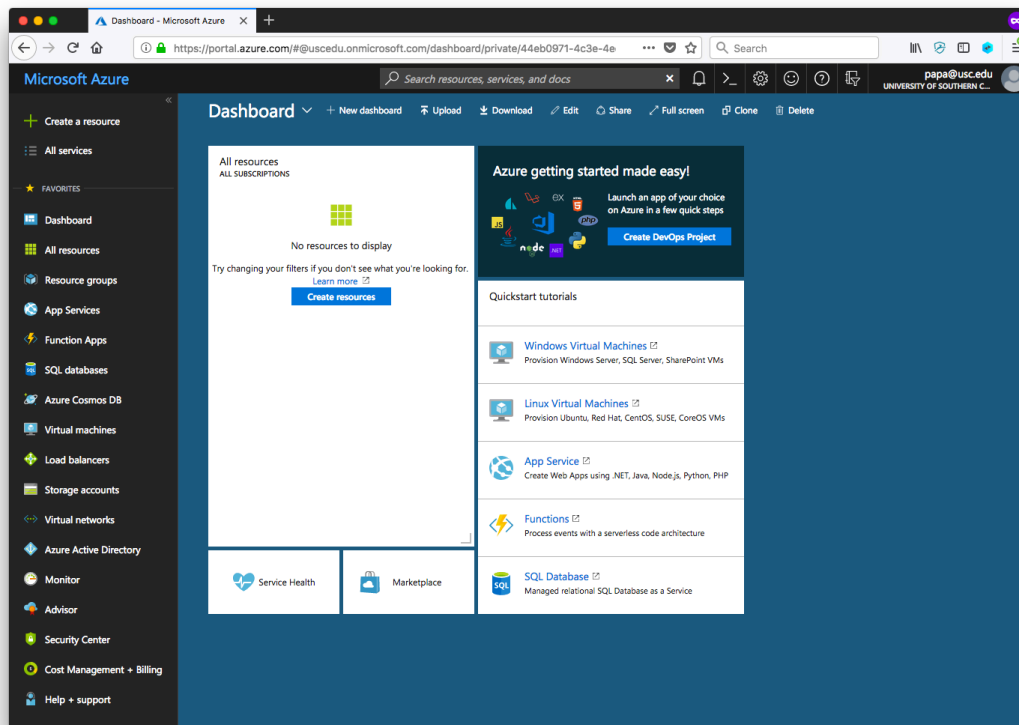


Homework #5 Microsoft Azure with Node.js

Using the instructions below one can establish a Node.js Web app using Microsoft Azure App Service developed for Assignment #8 and have it executed there.

1. Sign ups

This section assumes that you have performed the installation for homework #5, including (1) Sign up for **Microsoft Azure for Students** free trial, (2) obtained \$100 credit, and (3) activated your user account using your @usc.edu email address., and (4) logged in to the **Microsoft Azure Dashboard**.



2. Create a Node.js App in Azure App Service on Linux

App Service on Linux provides a highly scalable, self-patching web hosting service using the Linux operating system. Microsoft Azure provides a **QuickStart tutorial** that shows how to deploy a Node.js app to **Azure App Service on Linux**. The tutorial is available at:

<https://docs.microsoft.com/en-us/azure/app-service/containers/quickstart-nodejs>

You can follow the steps in the tutorial using a Mac, Windows, or Linux machine.

The Quickstart shows how to deploy a Node.js app to App Service on Linux using the Cloud Shell.

Note: in the step labeled “**Create a web app**”, this is the sequence of commands to execute to the **az CLI**:

```
cd quickstart                                (←-----NEW!!!!)
cd nodejs-docs-hello-world
az webapp up -n <app_name>
```

The output may look like this.

```
$az webapp up -n csci571nodejs
Resource group 'appsvc_rg_Linux_CentralUS' already exists.
$ az webapp up -n csci571nodejs
Creating Resource group 'appsvc_rg_Linux_CentralUS' ...
Resource group creation complete
Creating App service plan 'appsvc_asp_Linux_CentralUS' ...
Operation failed with status: 'Too Many Requests'. Details: 429 Client
Error: Too Many Requests for url:
https://management.azure.com/subscriptions/ad1c0ffe-1c5f-4ba7-b6
4d-849c8e0aed24/resourceGroups/appsvc_rg_Linux_CentralUS/provide
rs/Microsoft.Web/serverfarms/appsvc_asp_Linux_CentralUS?api-vers
ion=2016-09-01

$ az webapp up -n csci571nodejs
Resource group 'appsvc_rg_Linux_CentralUS' already exists.
App service plan 'appsvc_asp_Linux_CentralUS' already exists.
Creating app 'csci571nodejs' ....
```

LONG WAIT HERE

```
Webapp creation complete
Updating app settings to enable build after deployment
Creating zip with contents of dir
/home/marco/quickstart/nodejs-docs-hello-world ...
Preparing to deploy and build contents to app.
Fetching changes.
Preparing deployment for commit id '97c72e1c5d'.
Generating deployment script.
Generating deployment script.
Running deployment command...
Running deployment command...
Running deployment command...
```

```

Running deployment command...
Running deployment command...
Running deployment command...
Running post deployment command(s)...
All done.
{
  "app_url": "https://csci571nodejs.azurewebsites.net",
  "location": "Central US",
  "name": "csci571nodejs",
  "os": "Linux",
  "resourcegroup": "appsvc_rg_Linux_CentralUS ",
  "serverfarm": "appsvc_asp_Linux_CentralUS",
  "sku": "STANDARD",
  "src_path": "/home/marco/quickstart/nodejs-docs-hello-world ",
  "version_detected": "6.9",
  "version_to_create": "node|6.9"
}
$

```

The full URL for your application is listed as the value of the **app_url** key in the JSON :

<http://yourapp.azurewebsites.net/>

Also notice, if you get an error, like above, just re-run the “az” command again.

3. Editing your Nodejs web app

In the quickstart tutorial, you edit the **index.js** file using a remote editor like **nano**. After edits, you re-deploy using the az command:

```
az webapp up -n <app_name>
```

This is example output of a re-deploy:

```

$az webapp up -n csci571nodejs
Resource group 'appsvc_rg_Linux_CentralUS' already exists.
App service plan 'appsvc_asp_Linux_CentralUS' already exists.
App 'csci571nodejs' already exists
Updating app settings to enable build after deployment
Creating zip with contents of dir
/home/marco/quickstart/nodejs-docs-hello-world ...
Preparing to deploy and build contents to app.
Fetching changes.
Preparing deployment for commit id 'bd6337d99f'.
Running deployment command...
Running deployment command...

```

Running deployment command...

Running post deployment command(s)...

All done.

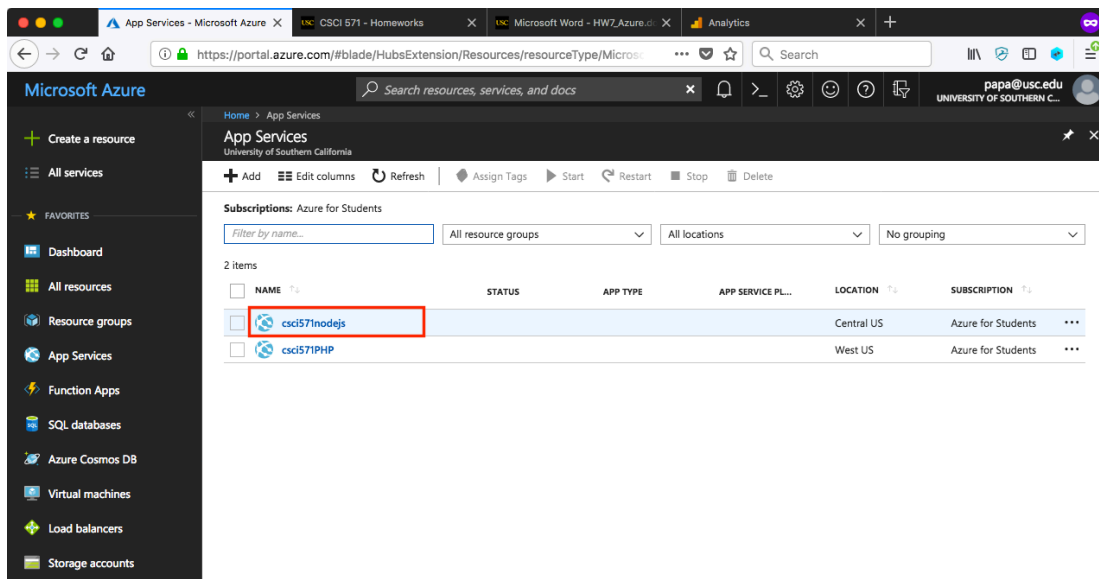
```
{
  "app_url": "https://csci571nodejs.azurewebsites.net",
  "location": "Central US",
  "name": "csci571nodejs",
  "os": "Linux",
  "resourcegroup": "appsvc_rg_Linux_CentralUS ",
  "serverfarm": "appsvc_asp_Linux_CentralUS",
  "sku": "STANDARD",
  "src_path": "/home/marco/quickstart/nodejs-docs-hello-world ",
  "version_detected": "6.9",
  "version_to_create": "node|6.9"
}
```

4. Manage your Azure web app

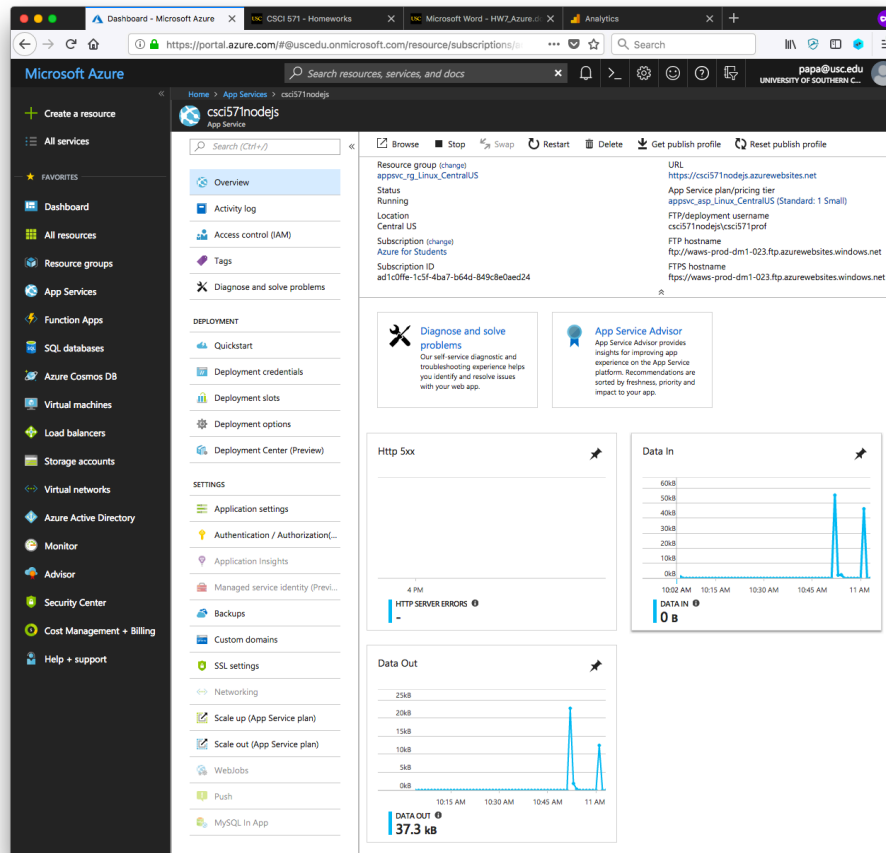
Go to the **Azure portal** to manage the web app you created.

<https://portal.azure.com/>

From the left menu, click App Services, and then click the name of your Azure web app.



Click the **app NAME**. You see your web app's **Overview** page. Here, you can perform basic management tasks like browse, stop, start, restart, and delete.



The left menu provides different pages for configuring your app.

5. Connecting using SSH

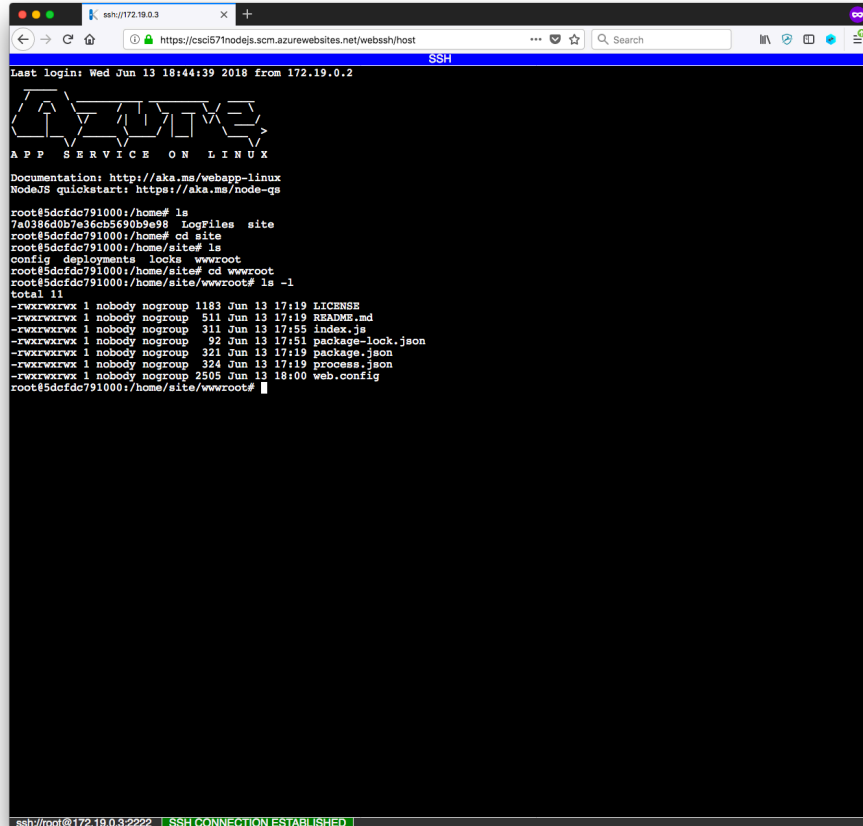
Follow the steps “**Open SSH session in browser**” in the article titled ‘**SSH support for Azure App Service on Linux,**’ available at:

<https://docs.microsoft.com/en-us/azure/app-service/containers/app-service-linux-ssh-support>

For example, I can connect with a browser to my Nodejs instance:

<https://csci571nodejs.scm.azurewebsites.net/webssh/host>

You will need to be logged in to Azure to be authenticated though SSH. You will see output as in the next image.

A screenshot of a web browser window displaying an SSH terminal session. The browser's address bar shows the URL 'https://csci571nodejs.scm.azurewebsites.net/webssh/host'. The terminal window has a title bar 'SSH' and a status bar at the bottom that reads 'ssh://root@172.19.0.3:2222 SSH CONNECTION ESTABLISHED'. The terminal output shows the last login time as 'Wed Jun 13 18:44:39 2018 from 172.19.0.2'. It displays the 'APP SERVICE ON LINUX' logo and links to documentation and NodeJS quickstart. The user 'root' is at the prompt 'root@5dcfcd791000:/home#'. They run 'ls' showing files like 'Logfiles' and 'site'. Then they run 'cd site' and 'ls' again, showing files like 'config', 'deployments', 'locks', and 'wwwroot'. Finally, they run 'cd wwwroot' and 'ls -l', which shows a long listing of files including 'LICENSE', 'README.md', 'index.js', 'package-lock.json', 'package.json', 'process.json', and 'web.config'. The prompt then changes to 'root@5dcfcd791000:/home/site/wwwroot#'.

Notice that the root of the NodeJS server is:

/home/site/wwwroot

6. Connecting using FTP

Follow the steps in the article titled “**Configure deployment credentials for Azure App Service**,” at:

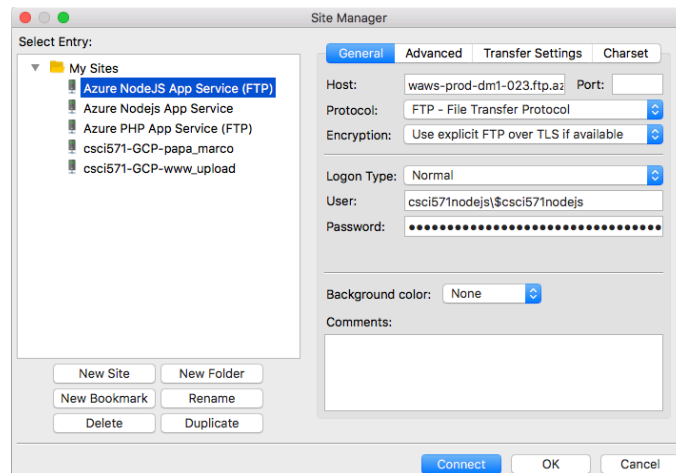
<https://docs.microsoft.com/en-us/azure/app-service/app-service-deployment-credentials>

In the article, scroll to the section titled “**Get and reset app-level credentials**”. As described, download the Publish profile, by clicking on the **Get publish profile** link. Open the profile (an XML file) on your local machine, and extract these values:

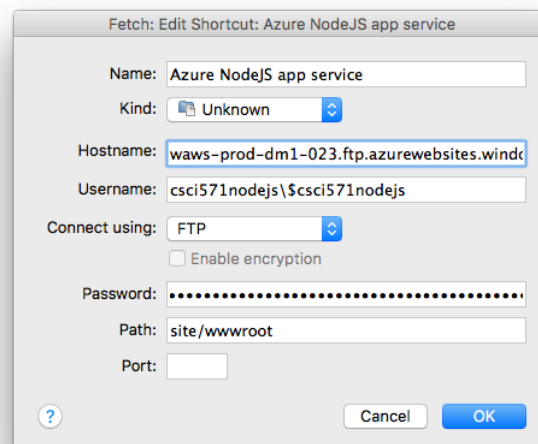
```
publishUrl="ftp://waws-prod-dm1-023.ftp.azurewebsites.windows.net/site/wwwroot"
userName="csci571nodejs\csci571nodejs"
userPWD="1D6JgGkTAmjCoNm7tHj23xkSGRBws4tfztyNQub970xY2n46dauLcvijwoBn"
```

Remove “ftp://” and “/site/wwwroot” from the publishUrl, to obtain the hostname (waws-prod-dm1-023.ftp.azurewebsites.windows.net). The default remote directory or path is /site/wwwroot.

Enter these values in an FTP client such as **Fetch** or **Filezilla**. Please note that we were unable to use SFTP with the above clients. Use FTP protocol.



Filezilla FTP client Setting page



Fetch FTP client Shortcut

Have fun exploring Azure Web Services!!