

Associated Engineering Team Discovery Channel

Deploying a React Client to Azure

Saturday, February 17, 2018
Version 1.0

Document Information

Revision History

Date	Version	Status	Prepared by	Comments
2018/02/17	1.0	Submitted	Ryan Koon	This documentation is a post project status action item

Stakeholders

Name	Role	Contact Details
Shawn Goulet	Sponsor (Main Contact)	(reachable via TAs)
Steve Robinson	Sponsor	(reachable via TAs)
Julin Song	TA	(reachable on Piazza)
Iris Ren	Supporting TA	(reachable on Piazza)
Anna Scholtz	Supporting TA	(reachable on Piazza)
Ryan Koon	Project Manager / Front-end Developer	ryankoon@alumni.ubc.ca / 604 349 8247
Andrew Chang	Front-end Lead / Developer	
Shih-Chun (Joyce) Huang	Back-end Lead / Developer	
Nicky Chan	Front-end Developer	(reachable via Front-end Lead)
Qiushan Zhao	Back-end Developer	(reachable via Back-end Lead)
Sebastian Wels-Lopez	Back-end Developer	(reachable via Back-end Lead)
Alan Wong	Back-end Developer	(reachable via Back-end Lead)

Table of Contents

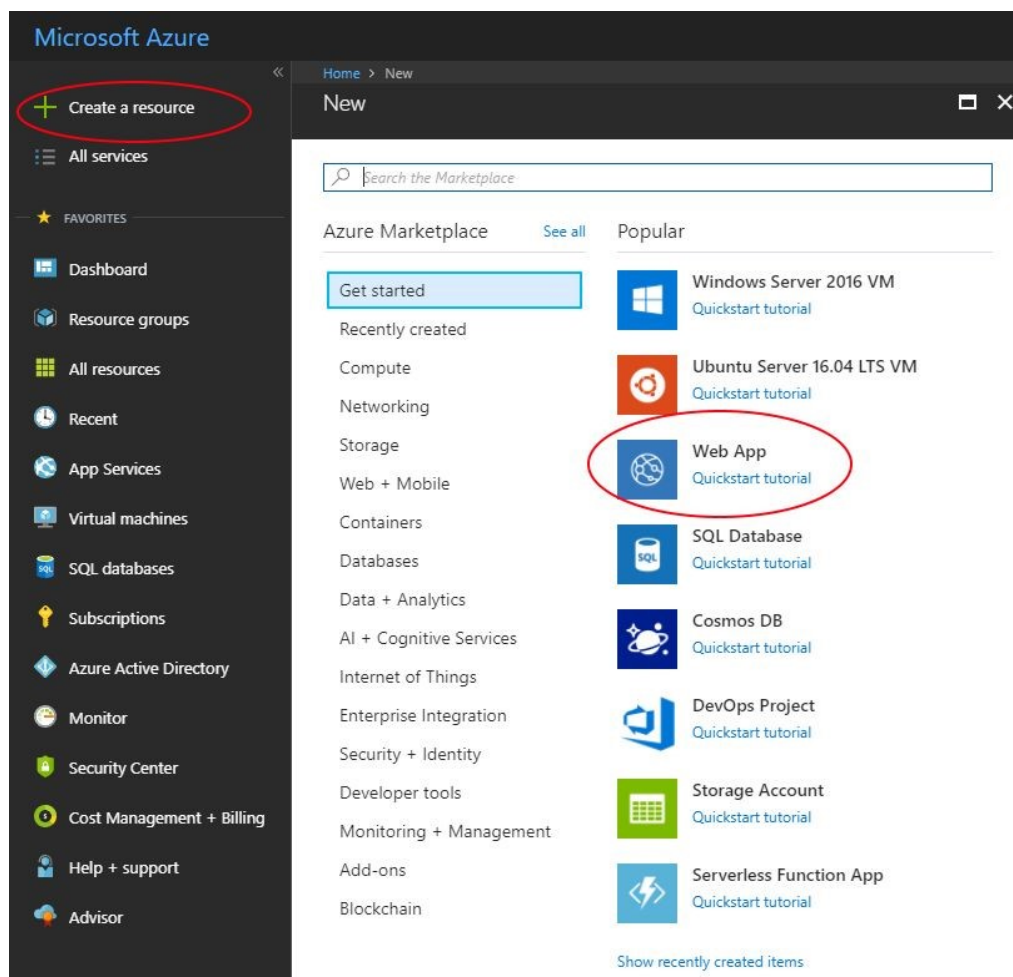
Document Information	2
Revision History	2
Stakeholders	2
Table of Contents	3
Create and run React application	4
Create Web App on Azure	4
Assign to an existing app service plan or create a new one	5
Download publish profile	6
Create a Web.config file	7
Build React app	8
Transfer files via ftp	8
Deploying from GitLab (optional)	10
Other Resources	11

1. Create and run React application

```
npx create-react-app my-app  
cd my-app  
npm start
```

More details can be found here: <https://github.com/facebook/create-react-app#quick-overview>

2. Create Web App on Azure



3. Assign to an existing app service plan or create a new one

Microsoft Azure

Search resources, services and docs

Home > New > Web App > App Service plan > New App Service Plan

Create a resource

All services

FAVORITES

Dashboard

Resource groups

All resources

Recent

App Services

Virtual machines

SQL databases

Subscriptions

Azure Active Directory

Monitor

Security Center

Cost Management + Billing

Help + support

Advisor

Web App

Create

* App name
createazurewebapp ✓
azurewebsites.net

* Subscription
DreamSpark

* Resource Group
☒ Create new ☐ Use existing
createazurewebapp ✓

* App Service plan/Location
azureserviceplan(West US) >

☐ Pin to dashboard

Create Automation options

App Service plan

Select a plan for the web app

An App Service plan is the container for your app. The App Service plan settings will determine the location, features, cost and compute resources associated with your app.

Create new

New App Service Plan

Create a plan for the web app

* App Service plan
serviceplan ✓

* Location
West US

* Pricing tier
F1 Free >

OK

4. Download publish profile

From the navigation bar or search:

App Services > createazurewebapp > Overview

The screenshot shows the Azure App Service Overview page for an application named 'createazurewebapp'. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and a DEPLOYMENT section with links to Quickstart, Deployment credentials, Deployment slots, Deployment options, and Continuous Delivery (Preview). The top toolbar includes actions like Browse, Stop, Swap, Restart, Delete, and 'Get publish profile' (which is circled in red), along with 'Reset publish profile'. A purple banner below the toolbar provides a link to the Quickstart guide. The main content area displays app details in two columns: Resource group (createazurewebapp), Status (Running), Location (West US), Subscription (DreamSpark), and Subscription ID (adb22bee-489c-4467-b9a0-9110e9fe2610) on the left; and URL (https://createazurewebapp.azurewebsites.net), App Service plan/pricing tier (Free: 0 Small), FTP/deployment username (createazurewebapp/ryankoon), FTP hostname (ftp://waws-prod-bay-091.ft.azurewebsites.windows.net), FTPS hostname (ftps://waws-prod-bay-091.ft.azurewebsites.windows.net), OS name (Windows Server 2016), and OS version (10.0.14393) on the right. At the bottom, there are three performance metrics: Http 5xx (100, 80, 60, 40), Data In (100B, 80B, 60B, 40B), and Data Out (100B, 80B, 60B, 40B).

Http 5xx		Data In		Data Out	
100	100B	100B	100B		
80	80B	80B	80B		
60	60B	60B	60B		
40	40B	40B	40B		

This file contains the credentials and ftp site for uploading files at a later step.

5. Create a Web.config file

We will need to rewrite urls, otherwise the server on Azure attempts to look for files based on the url instead of loading the correct React components through the React router (if it is set up)

Here is an example:

```
<configuration>
  <system.webServer>
    <handlers>
      <!-- indicates that the app.js file is a node.js application to be
handled by the iisnode module -->
      <add name="iisnode" path="server.js" verb="*" modules="iisnode" />
    </handlers>
    <rewrite>
      <rules>
        <!-- Don't interfere with requests for logs -->
        <rule name="LogFile" patternSyntax="ECMAScript"
stopProcessing="true">
          <match url="^[a-zA-Z0-9_\-]+\\.js\.logs\/\d+\\.txt$" />
        </rule>
        <!-- Don't interfere with requests for node-inspector debugging
-->
        <rule name="NodeInspector" patternSyntax="ECMAScript"
stopProcessing="true">
          <match url="^server.js\/debug[\/]?" />
        </rule>

        <rule name="React Routes" stopProcessing="true">
          <match url=".*" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_FILENAME}" matchType="IsFile"
negate="true" />
            <add input="{REQUEST_FILENAME}" matchType="IsDirectory"
negate="true" />
            <add input="{REQUEST_URI}" pattern="^/(api)"
negate="true" />
          </conditions>
          <action type="Rewrite" url="/" />
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```

If you are deploying a static web app without a back-end, you should not need the handlers or rules for server.js above.

More details can be found here:

https://medium.com/@to_pe/deploying-create-react-app-on-microsoft-azure-c0f6686a4321

In addition to the application files, we will need to transfer this file to Azure.

6. Build React app

Create a production build of your app.

```
npm run build
```

More details can be found here:

<https://github.com/facebook/create-react-app/blob/master/packages/react-scripts/template/README.md#deployment>

7. Transfer files via ftp

Sample Publish Profile:

```
createazurewebapp.PublishSettings  X
1 <publishData><publishProfile profileName="createazurewebapp - Web Deploy"
2   publishMethod="MSDeploy"
3   publishUrl="createazurewebapp.scm.azurewebsites.net:443"
4   msdeploySite="createazurewebapp" userName="$createazurewebapp"
5   userPWD="YourUserPWD"
6   destinationAppUrl="http://createazurewebapp.azurewebsites.net"
7   SQLServerDBConnectionString="" mySQLDBConnectionString=""
8   hostingProviderForumLink="" controlPanelLink="http://windows.azure.com"
9 </publishProfile><publishProfile
10  profileName="createazurewebapp - FTP" publishMethod="FTP"
11  publishUrl="ftp://waws-prod-bay-091.ftp.azurewebsites.windows.net/site/wwwroot"
12  ftpPassiveMode="True" userName="createazurewebapp\createazurewebapp"
13  userPWD="YourUserPWD"
14  destinationAppUrl="http://createazurewebapp.azurewebsites.net"
15  SQLServerDBConnectionString="" mySQLDBConnectionString=""
16  hostingProviderForumLink="" controlPanelLink="http://windows.azure.com"
17 </publishProfile></publishData>
18
19
```

We will use lftp to transfer files to Azure.

The three values we will need from the publish profile are:

- publishUrl
- userName
- userPWD

lftp command:

```
lftp -u $FTP_USER,$FTP_PASSWORD $FTP_URL -e "mirror -R -p -x=$SKIP_FILES  
$SOURCE_FOLDER $TARGET_FOLDER"
```

Using the sample publish profile:

```
lftp -u createazurewebapp\\\$createazurewebapp,YourUserPWD  
ftp://waws-prod-bay-091.ftp.azurewebsites.windows.net/site/wwwroot -e  
"mirror -R -p -x=node_modules build/. ./"
```

mirror - replaces all the files in the target folder to match the source
-R - transfers files from \$SOURCE_FOLDER to \$TARGET_FOLDER
-p - do not set file permissions (needed when deploying from GitLab)
-x=... - skip matching files
build/. - mirror the contents of the build folder
./ - target folder (wwwroot)

The lftp documentation can be found here:

<https://lftp.yar.ru/lftp-man.html>

Note: The username has escape characters for the backslash and dollar sign.

If you are using windows, you can install the “Windows Subsystem for Linux” to run these commands:

<https://docs.microsoft.com/en-us/windows/wsl/install-win10>

Do not forget to copy the web.config file.

You can copy the file into the build folder prior to running lftp.

In summary, these are the commands for uploading via lftp:

```
cp web.config build  
apt-get update  
apt-get install lftp  
lftp -u createazurewebapp\\\$createazurewebapp,YourUserPWD  
ftp://waws-prod-bay-091.ftp.azurewebsites.windows.net/site/wwwroot -e  
"mirror -R -p -x=node_modules build/. ./"
```

8. Deploying from GitLab (optional)

Sample `.gitlab-ci.yml` (to be placed in project root):

```
image: node:latest

cache:
  paths:
    - node_modules/

stages:
  - deploy

deploy_production:
  stage: deploy
  script:
    - yarn install
    - yarn run build
    - cp web.config build
    - apt-get update
    - apt-get install lftp
    - lftp -u $FTP_USER,$FTP_PASSWORD $FTP_URL -e "mirror -R -p -x=node_modules
build/. ./"
  environment:
    name: Production
    url: https://createazurewebapp.azurewebsites.net
  only:
    - master
```

<i>image</i>	- run in a docker image with the latest node version
<i>cache</i>	- cache node_modules folder to speed up updating dependencies
<i>stages</i>	- defines the stages of the pipeline
<i>deploy_production</i>	- user defined name of a pipeline job
<i>script</i>	- commands to run for the job
<i>environment</i>	- associate the deployment job with an environment to be able to track deployed commits in different environments on GitLab
<i>only</i>	- the job will only run on the specified branch

The above sample yaml uses secret variables (e.g. \$FTP_PASSWORD) that are configured on GitLab for each project (Settings > CI/CD > Secret Variables).

The \$ is used by GitLab to indicate a secret variable.

To escape this character for the Azure Username, use two dollar signs.

Example:

Secret Variable: FTP_USER

Value: createazurewebapp\\\$\$createazurewebapp

Other Resources

Facebook's documentation on deploying React apps to various targets (e.g. Azure, AWS, Firebase, etc.):

<https://github.com/facebook/create-react-app/blob/master/packages/react-scripts/template/README.md#deployment>

Kudu Services for your Azure web app:

<https://<azureappname>.scm.azurewebsites.net/>

Retrieve public key of via Kudu (to pull from GitLab):

<https://<azureappname>.scm.azurewebsites.net/api/sshkey?ensurePublicKey=1>

Add to GitLab via Settings > Repository > Deploy Keys

GitLab YAML configuration:

<https://docs.gitlab.com/ce/ci/yaml/README.html#gitlab-ci-yml>