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	[redacted]
Shih-Chun (Joyce) Huang	Back-end Lead / Developer	[redacted]
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)

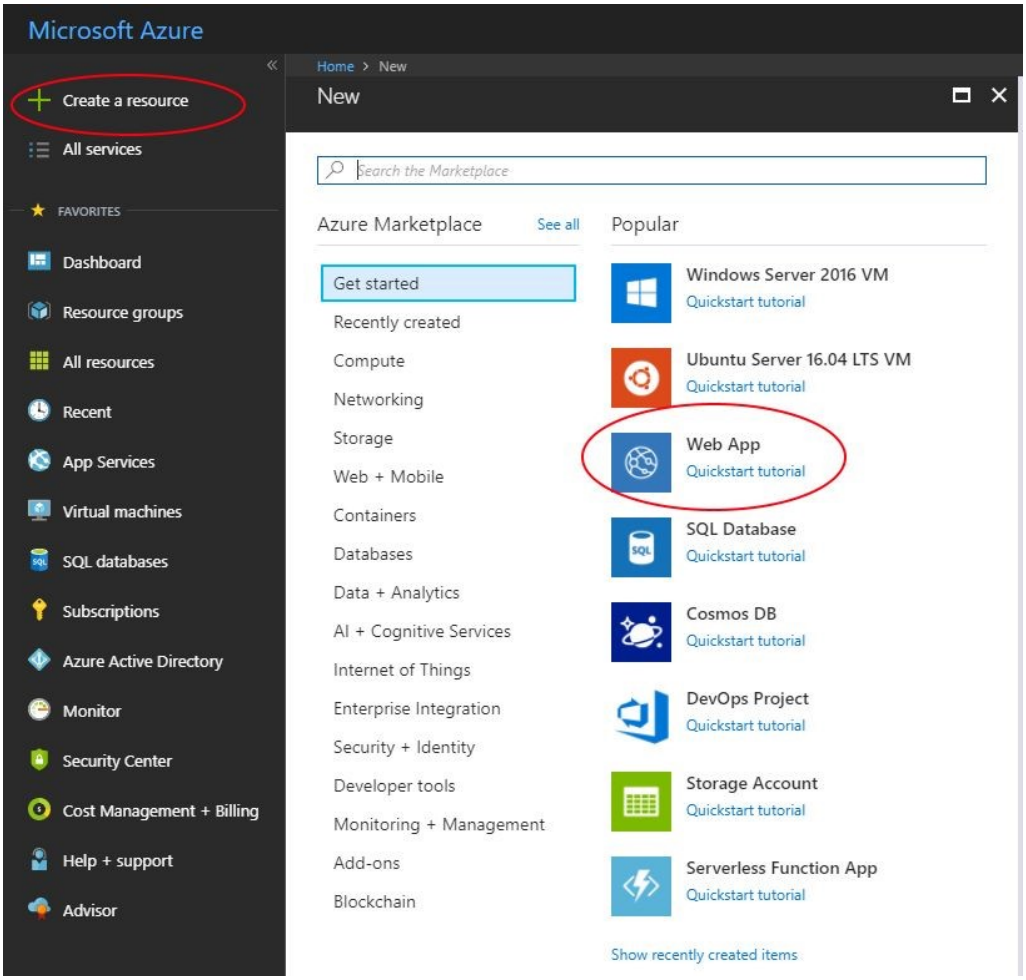
Table of Contents

1. Create and run React application

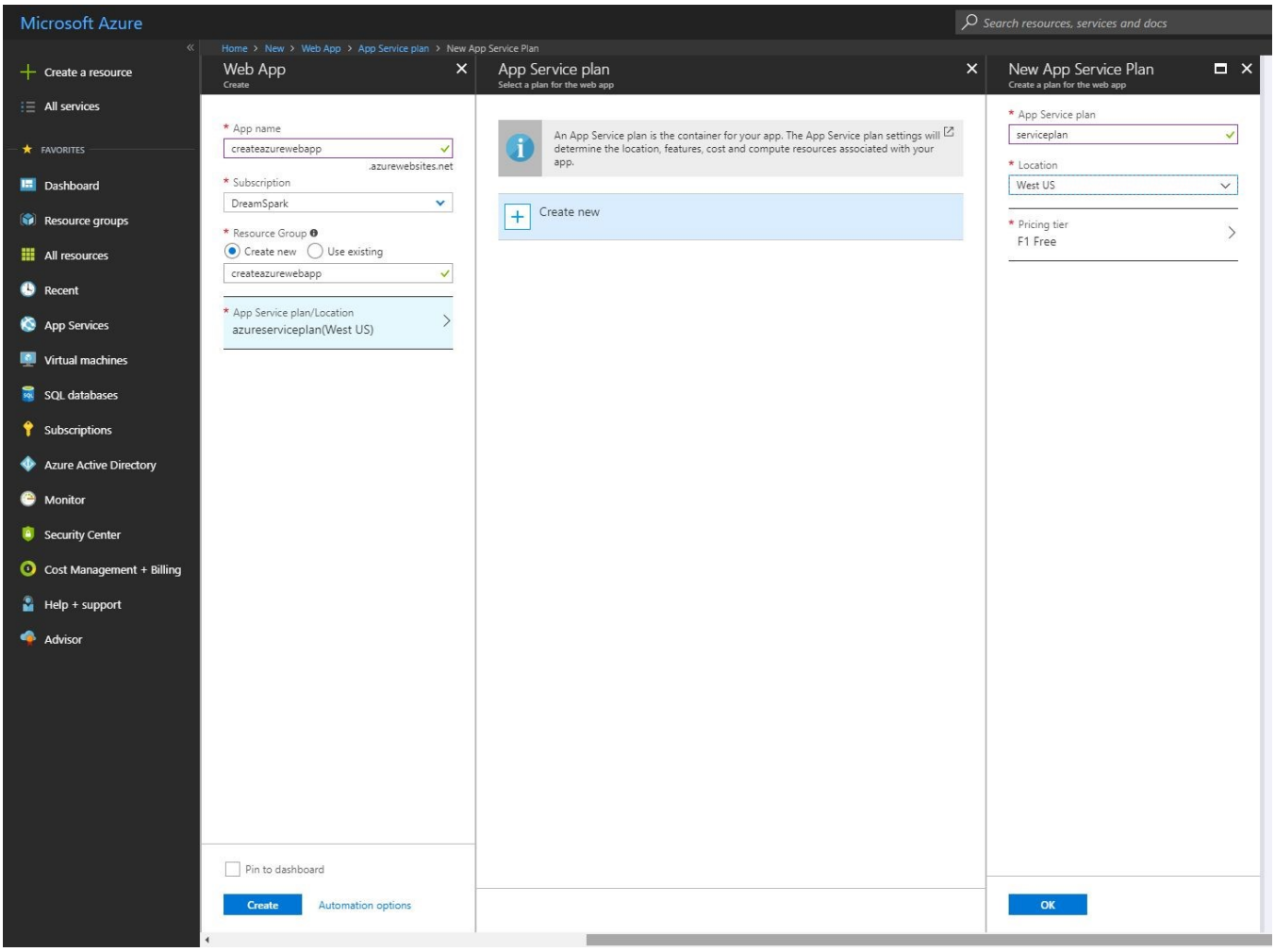
```
npx create-react-app my-app
cd my-app
npm start
```

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

2. Create Web App on Azure

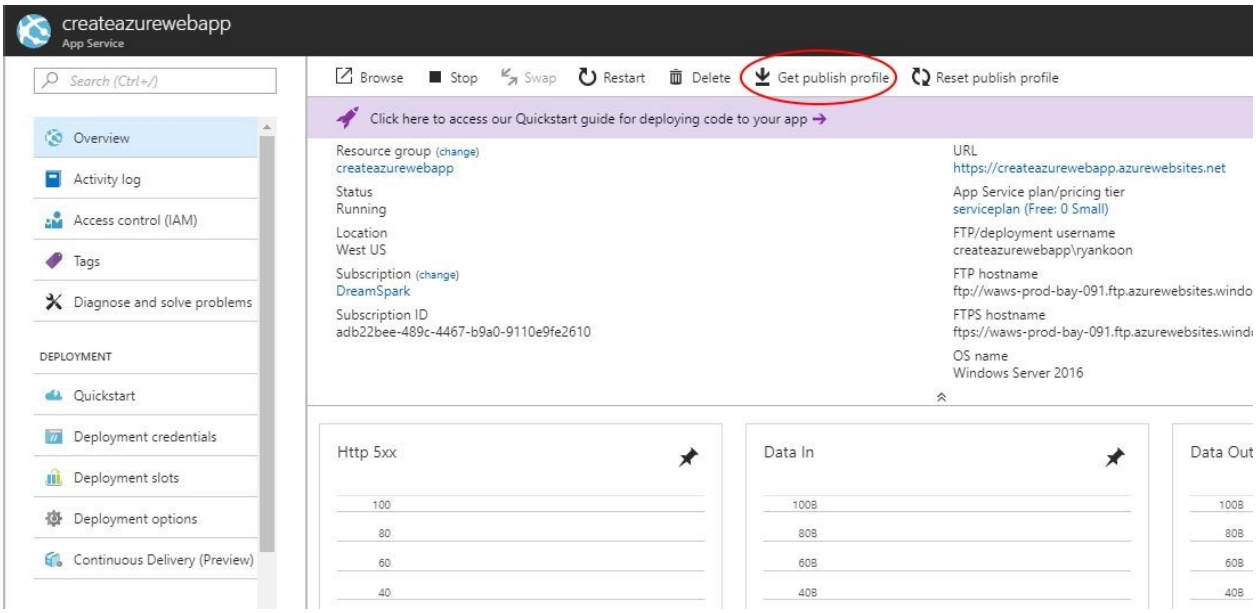


3. Assign to an existing app service plan or create a new one



4. Download publish profile

From the navigation bar or search:
App Services > createazurewebapp > Overview



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 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

```
<publishData><publishProfile profileName="createazurewebapp - Web Deploy"
publishMethod="MSDeploy"
publishUrl="createazurewebapp.scm.azurewebsites.net:443"
msdeploySite="createazurewebapp" userName="$createazurewebapp"
userPWD="YourUserPWD"
destinationAppUrl="http://createazurewebapp.azurewebsites.net"
SQLServerDBConnectionString="" mySQLDBConnectionString=""
hostingProviderForumLink="" controlPanelLink="http://windows.azure.com"
webSystem="WebSites"><databases /></publishProfile><publishProfile
profileName="createazurewebapp - FTP" publishMethod="FTP"
publishUrl="ftp://waws-prod-bay-091.ftp.azurewebsites.windows.net/site/wwwroot"
ftpPassiveMode="True" userName="createazurewebapp\$createazurewebapp"
userPWD="YourUserPWD"
destinationAppUrl="http://createazurewebapp.azurewebsites.net"
SQLServerDBConnectionString="" mySQLDBConnectionString=""
hostingProviderForumLink="" controlPanelLink="http://windows.azure.com"
webSystem="WebSites"><databases /></publishProfile></publishData>
```

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
```

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