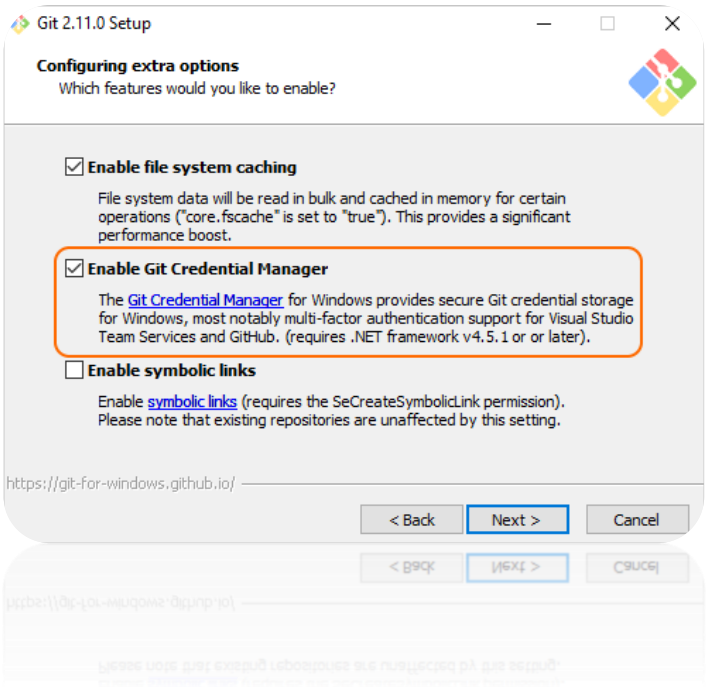
1.Pre-requisites

1. This lab assumes you have the Google Chrome browser installed and available for debugging. If you do not have Chrome installed, go to https://www.google.com/chrome/browser/

Download and run the latest Git for Windows installer, which includes the Git Credential Manager for Windows. Make sure to leave the Git Credential Manager installation option enabled when prompted.2



Note: When you connect to a VSTS Git repository from your Git client for the first time, the credential manager prompts for your Microsoft Account or Azure Active Directory credentials. If your account has multi-factor authentication enabled, you are prompted to go through that experience as well.

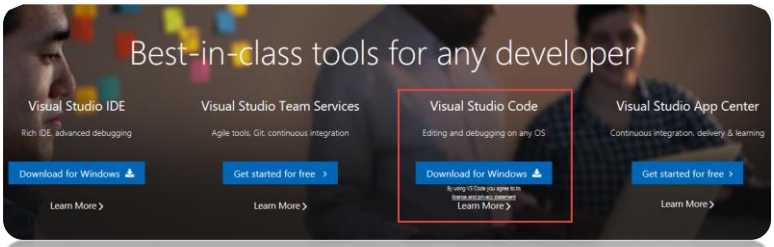
Download Visual Studio Code from http://visualstudio.com

Style Definition: Code: Space Before: 6 pt, After: 6 pt, Pattern: 10% (Auto Foreground, White Background)

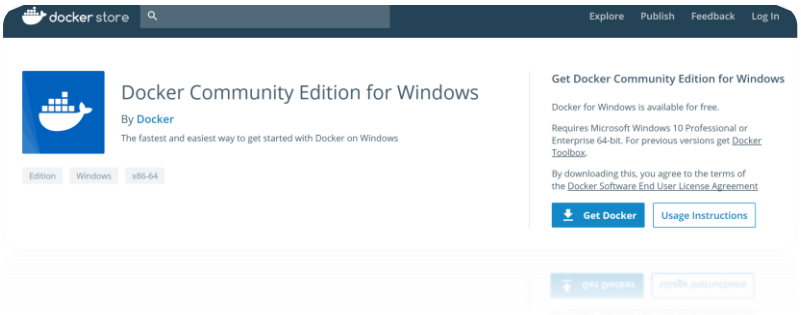
Formatted: Font: (Default) Segoe UI, Font color: Black

Formatted: If-text-block, Indent: Left: 0.5", Space Before: 12 pt, No bullets or numbering

Formatted: Font: (Default) Segoe UI, Font color: Black

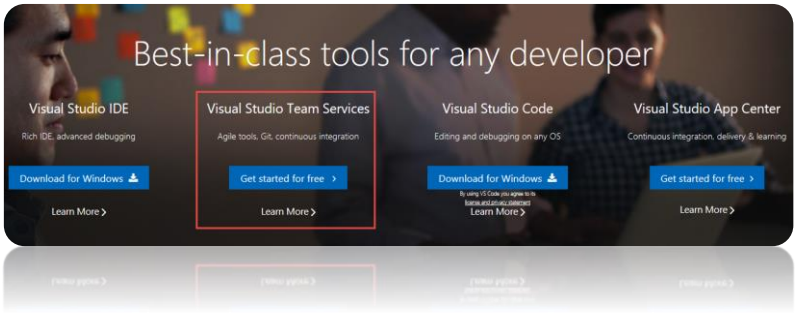


Install Docker from https://docs.docker.com/install/



2.Create a Project in VSTS

1. Create a new instance of Visual Studio Team Services by navigating to http://visualstudio.com



2. Click on "**New Project**" in VSTS.

3. Enter Project Name, Description, Version control, and Work item process and click **Create**.

Create new project

Projects contain your source code, work items, automated builds and more.

Project name *

FirstApp

Description

First application with VSTS

Version control

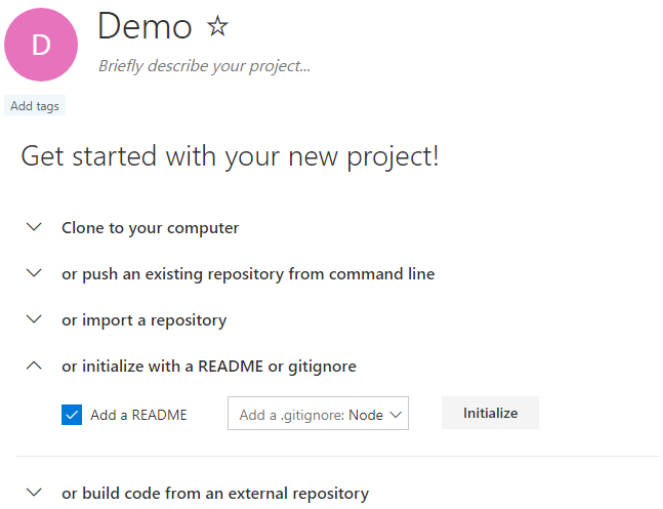
Git

Work item process

Agile

Create Cancel

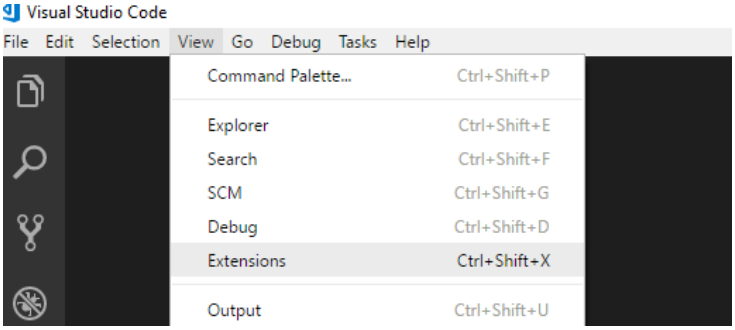
4. Select “or initialize with a readme or gitignore”.
5. Add a .gitignore file by selecting “Node”,
6. Click Initialize.



Readme file is used to give a brief introduction of the project and gitignore file is used to ignore tracking of files such as temp files and build results.

3. Open Visual Studio Code

1. Install Extensions by Selecting View→ Extensions and typing "javascript"



Recommended extensions to install:

Angular, TypeScript and HTML Snippets for VS Code

VSCode Angular TypeScript & Html Snippets

ESLint

JavaScript (ES6) code snippets

npm IntelliSense

Debugger for Chrome

Visual Studio Team Services

Docker

Docker Explorer

Nginx.Conf

Nginx.Conf Hint

Apache conf

Apache Conf Snippets for VS Code

Commented [JB1]: There is not an exact match for this. There are two close matches – not sure which one I should use?

Commented [JB2]: Again, no exact match so left wondering which one I should choose

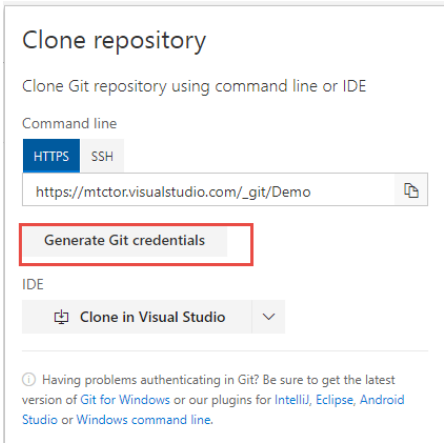
Commented [JB3]: There is only an "Apache Conf Snippets" extension… did not see one ending with VS Code

2. Launch Git Bash or use Windows Command line to execute the following commands to create our repository directory:

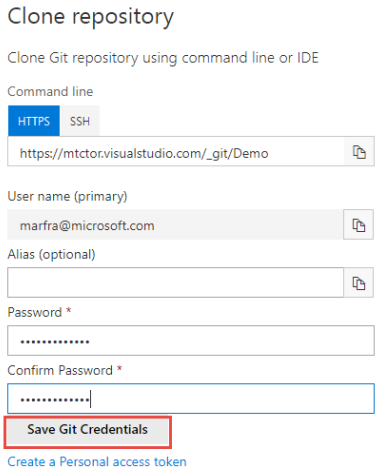
MINGW64:/c/shoppingcartdemo

```
codec@DESKTOP-GFGMI69 MINGW64 /c
$ cd /c

codec@DESKTOP-GFGMI69 MINGW64 /c
$ mkdir shoppingcartdemo

codec@DESKTOP-GFGMI69 MINGW64 /c
$ cd shoppingcartdemo/

codec@DESKTOP-GFGMI69 MINGW64 /c/shoppingcartdemo
$
```

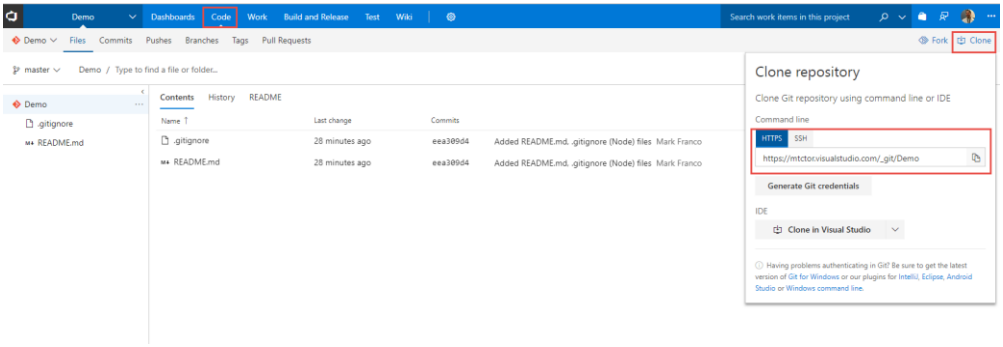
3. Open your VSTS project in your browser
~~3.~~4. Click on Clone in the upper right-hand corner
~~4.~~5. Generate Git Credentials:



~~5.~~6. Then ~~Enter~~ enter a new password and click Save Git Credentials:



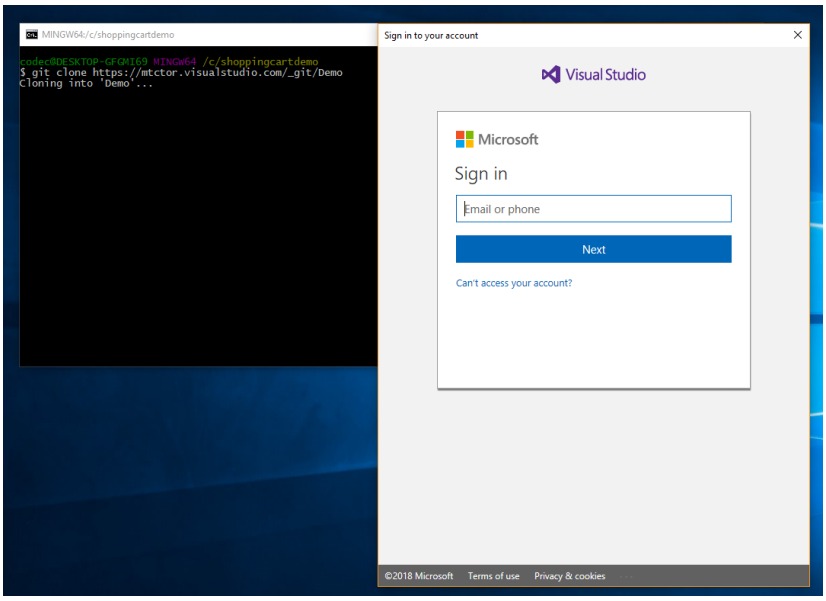
~~6.~~7. copy the git repository url as follows:



~~7.~~8. Clone the repository from the bash shell you opened earlier as follows:

~~8.~~ Git clone <git Repository you copied in previous step>

9. Enter your credentials you setup in previous steps



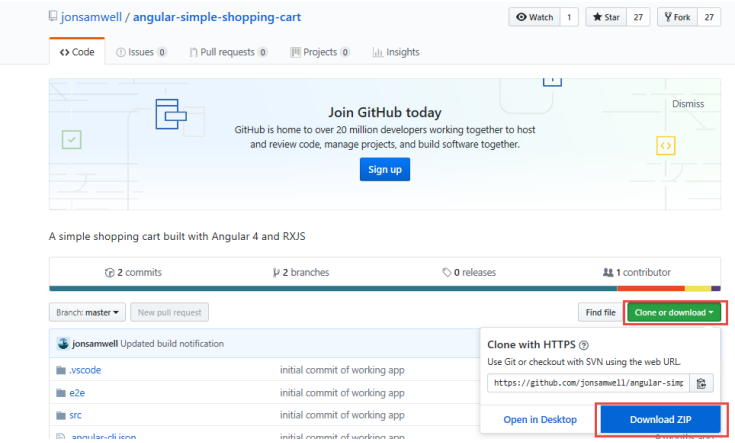
After successful login you should see:

```
MINGW64:/c/shoppingcartdemo

codec@DESKTOP-GFGMI69 MINGW64 /c/shoppingcartdemo
$ git clone https://mtctor.visualstudio.com/_git/Demo
Cloning into 'Demo'...
remote:
remote:                    vSTs
remote:                  vSTSVSTSv
remote:                vSTSVSTSVST
remote: VSTS         vSTSVSTSVSTSV
remote: VSTSVS     vSTSVSTSV STSVS
remote: VSTSVSTSvsTSVSTSVS   TSVST
remote: VS  tSVSTSVSTSv      STSVS
remote: VS   tSVSTSVST       SVSTS
remote: VS tSVSTSVSTSVSts    VSTSV
remote: VSTSVST    SVSTSVSTs VSTSV
remote: VSTSv        STSVSTSVSTSVS
remote:                VSTSVSTSVST
remote:                  VSTSVSTs
remote:                    VSTs    (TM)
remote:
remote:  Microsoft (R) Visual Studio (R) Team Services
remote:
remote: Found 4 objects to send. (86 ms)
Unpacking objects: 100% (4/4), done.

codec@DESKTOP-GFGMI69 MINGW64 /c/shoppingcartdemo
$
```

4. Write code

(not quite, we are just going to use an existing code base from GitHub and download the latest copy of the source to update our local repo).

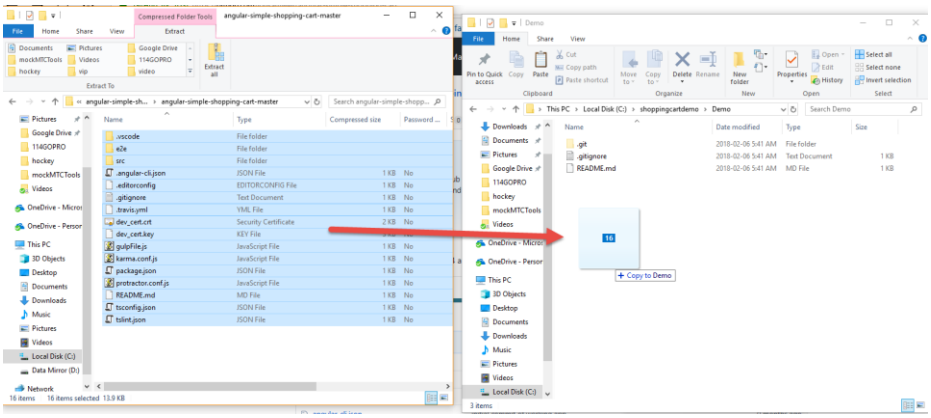
1. Open the browser and navigate to https://github.com/jonsamwell/angular-simple-shopping-cart
2. Download code as follows:



3. Extract the contents of the "angular-simple-shopping-cart-master" folder within the zip file to c:\shoppingcartdemo\demo

   Note: answer "replace" when duplicate files found.

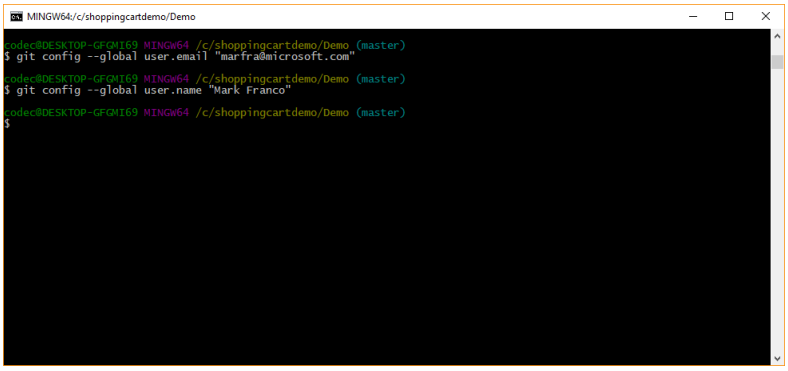
Commented [JB4]: The folder based on earlier steps should be FirstApp not Demo



4. Now we are going to add untracked files and commit our changes to our local repository, but before we can do that we have to tell Git who we are by issuing the two following commands:

```
git config --global user.email "you@outlook.com"
```

```
git config --global user.name "Your Name"
```



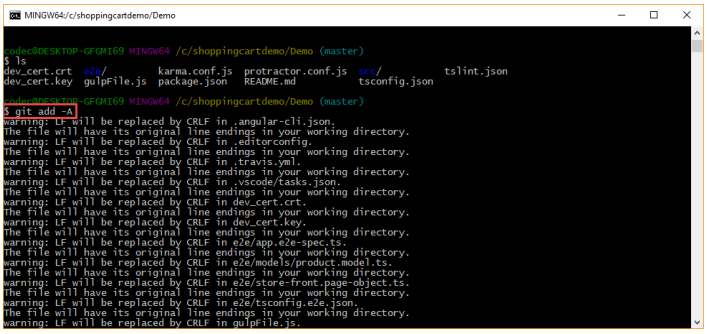
5. Add untracked files as follows:

Cd \FirstApp

~~5.~~ git add -A

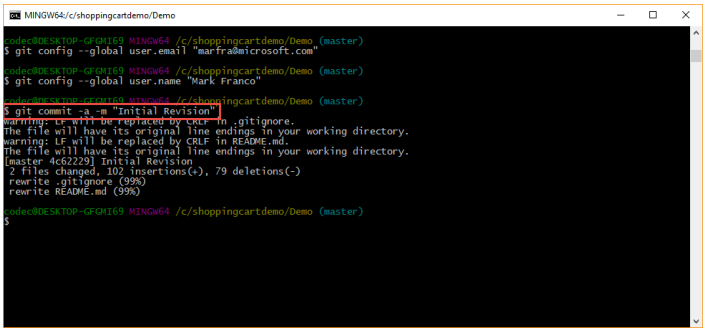
Commented [JB5]: I needed to move to the repo folder for FirstApp for git add -A to work

Formatted: Code, Space Before: 0 pt, After: 0 pt, No bullets or numbering, Pattern: Clear



6. Commit Changes:

~~6.~~ git commit -a -m "Initial Revision"

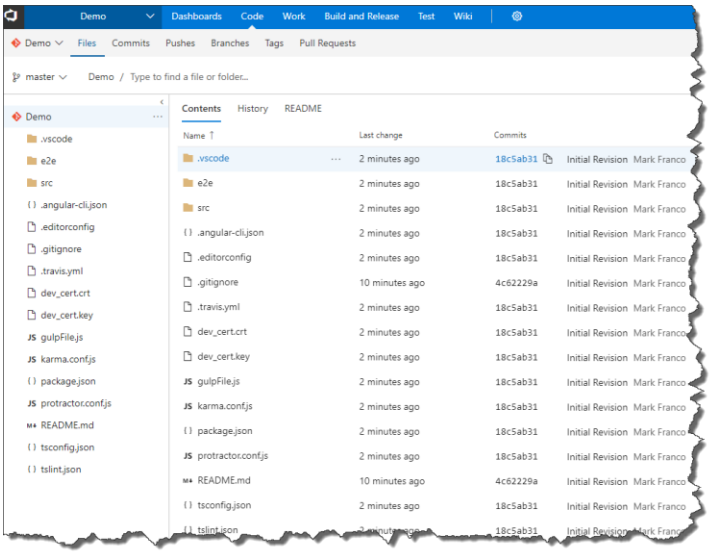


7. Push repository to VSTS into Master branch by executing the following command (no Screenshot):

Git push –repo <VSTS Git Repository url from previous steps>

i.e. git push –repo https://mtctor.visualstudio.com/_git/Demo

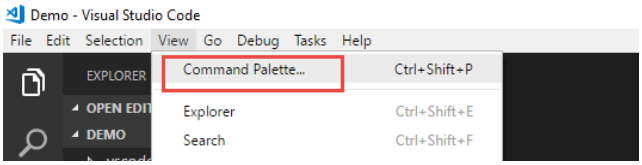
8. And Voila! You can now see your repository pushed up into VSTS:



5. Launch VSCode and setup VSTS integration using the new experience ~~again,~~

~~9. Open our project and setup VSTS integration using the new experience:~~

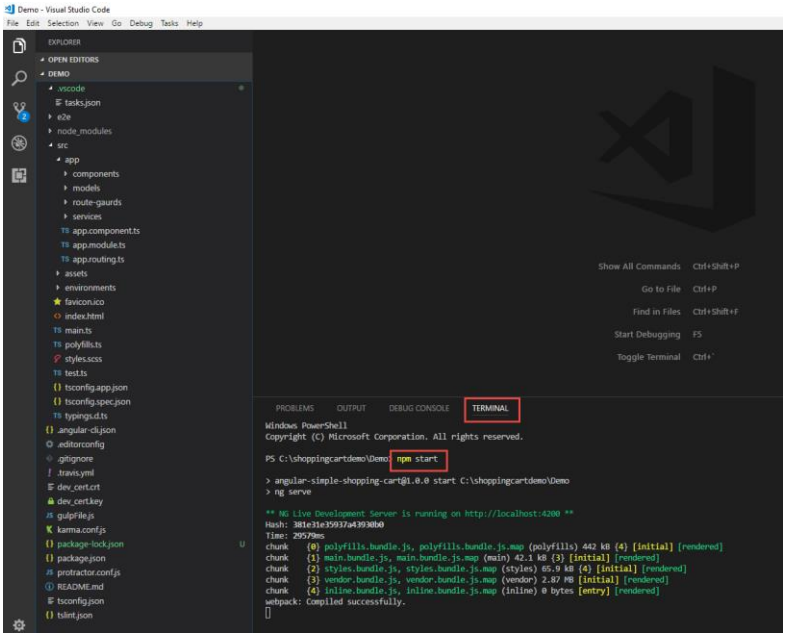
~~11.~~2. Watch this step by step video on how to setup the new experience. Note: to open the Command Pallet as shown in the video, use the menu as follows:



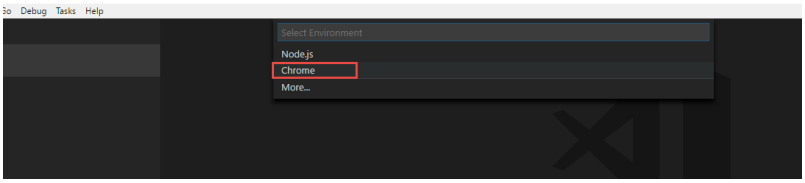
https://youtu.be/HnDNdm1WClo?t=2m55s

3. Ensure all dependencies are current by running "npm install" in the VS Code terminal window

~~12.~~4. Run a local instance of the app to see how it runs by running "npm start" in the vscode terminal:



Your app is compiled and running under a node web server, but we need to add a launch file so we can launch a debugger window using Chrome. We do so by creating a new configuration file by selecting the "Debug→Add Configuration" menu item and selecting "Chrome" from the drop down.



We need to ensure the new launch.json file is pointing to the correct url. Node will automatically assign a random port on your computer to host your angular application on and you can get this url from the previous step where you ran "NPM Start":

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\shoppingcartdemo\Demo> npm start

> angular-simple-shopping-cart@1.0.0 start C:\shoppingcartdemo\Demo
> ng serve

** NG Live Development Server is running on http://localhost:4200 **
Hash: 381e31e35937a43930b0
Time: 29579ms
chunk    {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 442 kB {4} [initial] [rendered]
chunk    {1} main.bundle.js, main.bundle.js.map (main) 42.1 kB {3} [initial] [rendered]
chunk    {2} styles.bundle.js, styles.bundle.js.map (styles) 65.9 kB {4} [initial] [rendered]
chunk    {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.87 MB [initial] [rendered]
chunk    {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry] [rendered]
webpack: Compiled successfully.
```

With this url , you are going to update the launch.json file and specifically update the “url” property of the Chrome configuration as such:

```
launch.json ×
1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5      "version": "0.2.0",
6      "configurations": [
7
8          {
9              "type": "chrome",
10             "request": "launch",
11             "name": "Launch Chrome against localhost",
12             "url": "http://localhost:4200",
13             "webRoot": "${workspaceFolder}"
14         }
15     ]
16 }
```

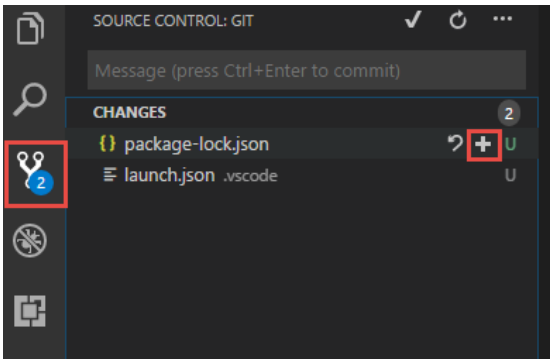
Now click on Debug→Start debugging

Final house keeping bits:

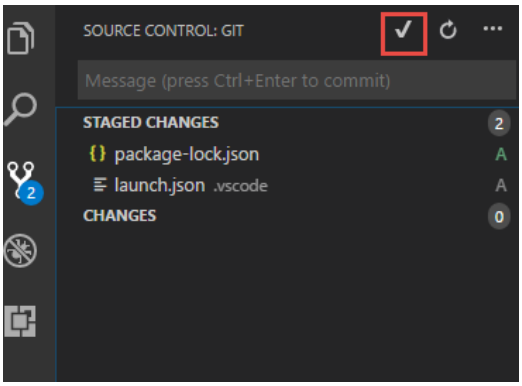
Check in your additional file “Launch.json” using the VSCODE IDE now:

Add Files to local repository (Stage)

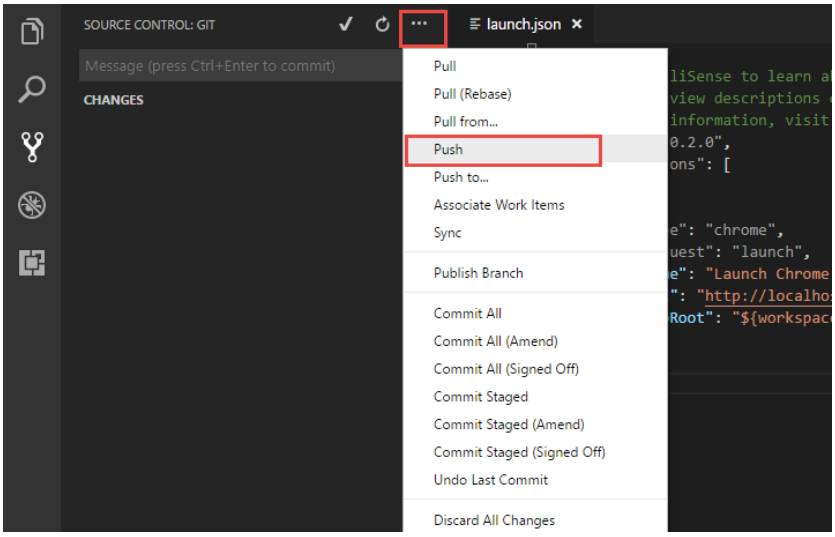
Commented [JB6]: Do you want to highlight the + button on the Changes header instead of the one on a single file?



Commit Changes to local Repository



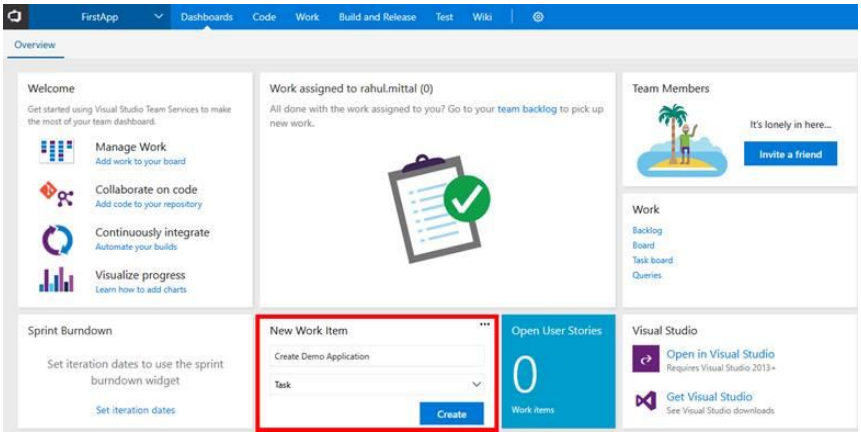
Push Changes from local repository to VSTS



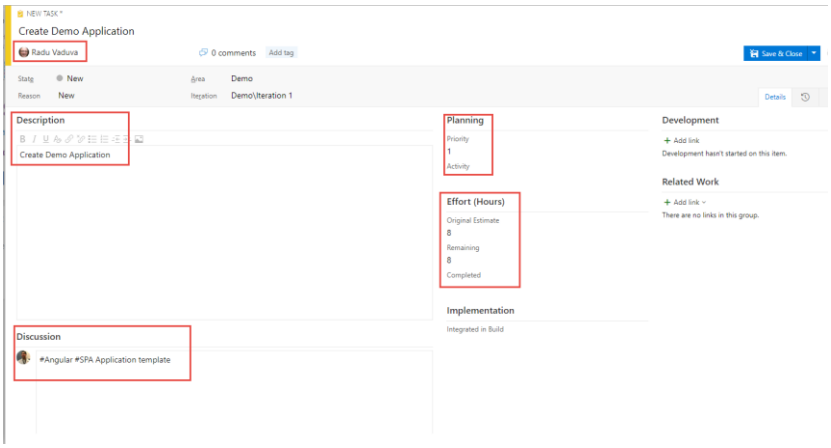
Development Complete...

6 Setting Up Work Item Check-in and Build Configuration

Go to VSTS ~~dashboard, and~~dashboard and create a task. We will associate this task with check-in.



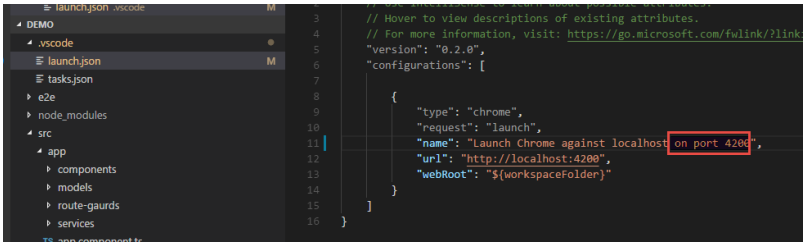
Assign a task to a resource (`Yourself` in this case), enter description, set priority, and specify effort. Click Save and Close.



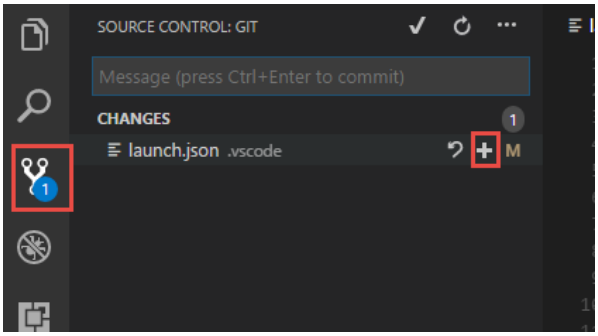
When user saves, unique task number is assigned to each task.

Go back to VSCODE, make changes to the launch.json file and associate the work item while committing the code.

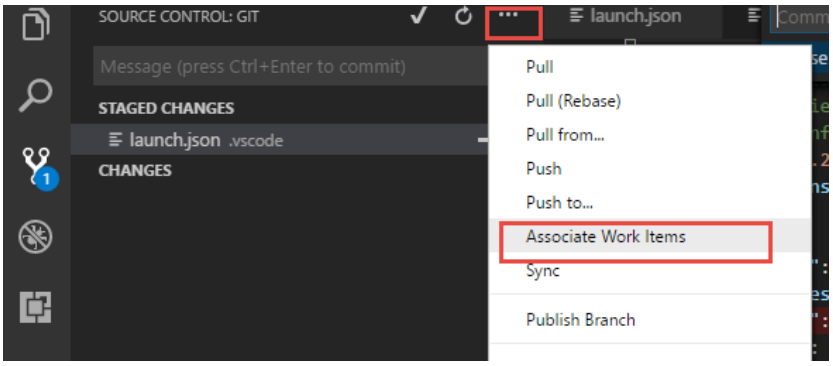
Make the code change by appending “on port 4200” as shown below:



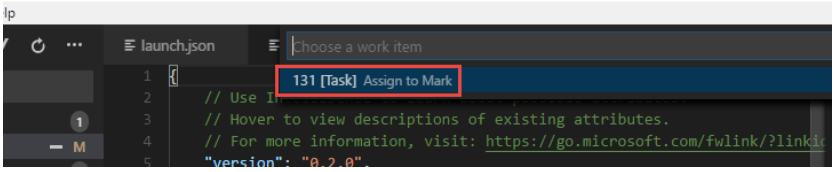
Add Change (Stage)



Commit Change by Associating work item:



Select Work Item task:



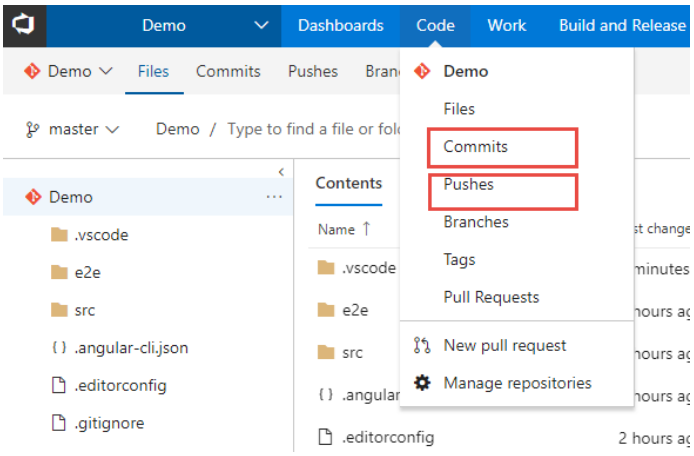
Commit Change with a message “Added port ”

Push Change to VSTS.

Done.

When we again go to task board in VSTS, we can see development history associated with this item.

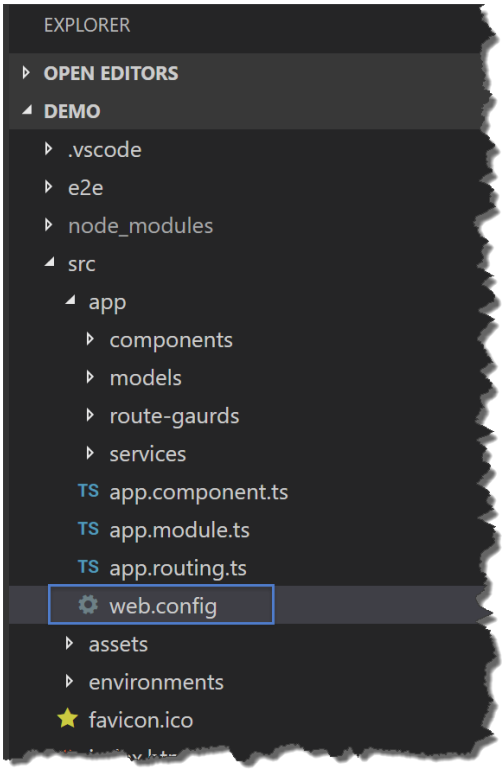
Check out here:



Deploy to Azure App Services

We will need to add a web.config file to instruct our underlying web server on Azure to rewrite all incoming request to serve our *index.html* file.

Create a new file named web.config in src\app\ by right-clicking on src\app folder and selecting 'New File'~~web.config:~~



Add the following contents to the web.config:

```
<configuration>

    <system.webServer>
        <staticContent>
            <mimeMap fileExtension=".json" mimeType="application/json" />
        </staticContent>

        <rewrite>
        <rules>
        <clear />

        <!-- ignore static files -->
        <rule name="AngularJS Conditions" stopProcessing="true">
        <match url="(assets/.*|.js|.css)" />
        <conditions logicalGrouping="MatchAll" trackAllCaptures="false" />
        <action type="None" />
        </rule>

        <!-- check if its root url and navigate to default page -->
        <rule name="Index Request" enabled="true" stopProcessing="true">
        <match url="^$" />
        <action type="Redirect" url="/home" logRewrittenUrl="true" />
        </rule>

        <!--remaining all other url's point to index.html file -->
        <rule name="AngularJS Wildcard" enabled="true">
        <match url="(.*)" />
        <conditions logicalGrouping="MatchAll" trackAllCaptures="false" />
        <action type="Rewrite" url="index.html" />
        </rule>

        </rules>
        </rewrite>
    </system.webServer>
</configuration>
```

Commented [JB7]: You need to include the text of the web.config as real text and not an image so people can copy paste. This is as far as I got today

Modify the /gulpfile.js as follows to remove the code that modifies the index.html `<base href="/">` element. The developer added this code but it is no longer needed as you can leverage angular CLI to modify this directly. Also we have added a copy process to deploy the web.config to the distribution folder:

```
var gulp = require('gulp');
```

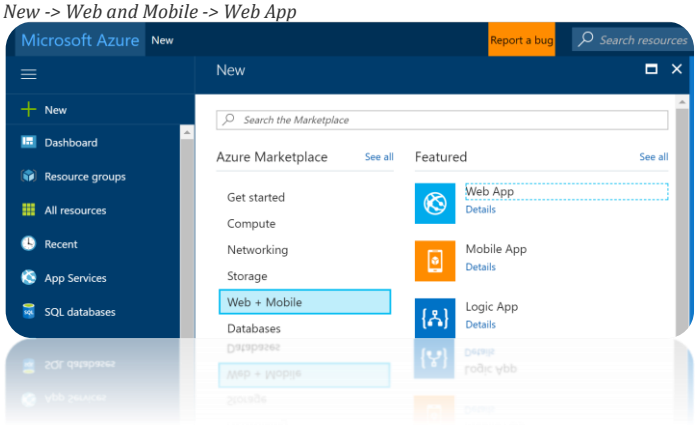
```
var replace = require('gulp-replace');
var htmlmin = require('gulp-htmlmin');

gulp.task('js:minify', function () {
  gulp.src(["./dist/main.*.js", "./dist/polyfills.*.js",
"./dist/inline.*.js"])
    .pipe(replace(/\/\*([\s\S]*?)\*\/[\s\S]?/g, ""))
    .pipe(gulp.dest("./dist"));
});

gulp.task('web:config', function () {
  gulp.src(["./src/app/web.config"])
    .pipe(gulp.dest("./dist"));
});
```

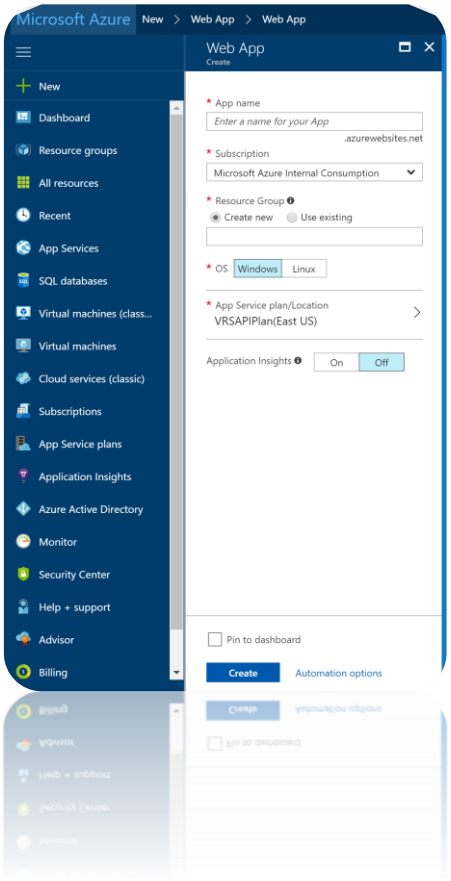
```
gulp.task("html:minify", function () {
  return gulp.src('dist/*.html')
    .pipe(htmlmin({ collapseWhitespace: true }))
    .pipe(gulp.dest('./dist'));
});

gulp.task("default", ["js:minify", "html:minify", "web:config"]);
```

Create the Azure App Service

The next step is to create an Azure Web App which will host our Angular application. You can sign up for a free or paid account and log in the Azure portal.

1. *New -> Web and Mobile -> Web App*



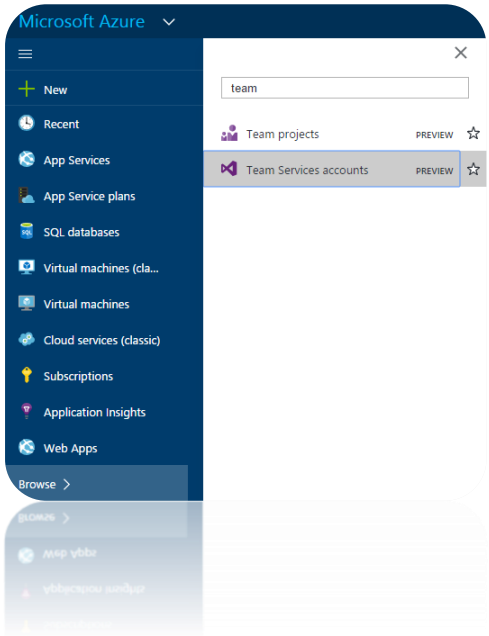


2. Fill out the required fields:
3. After pressing "Create" you should now have an Azure Web App created.

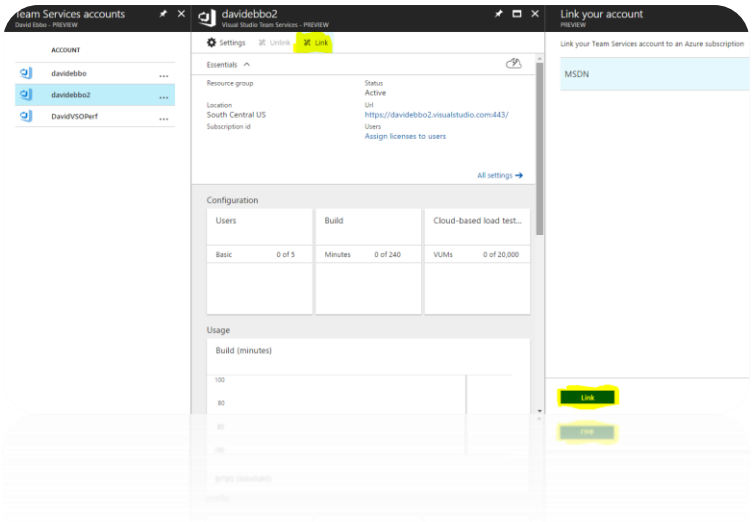
Linking your VSTS account to your Azure subscription

The last step is that you need to link your VSTS account to your Azure subscription (see also this post on this topic).

To do this, go to the Azure Portal, click More Services (image says 'Browse' but that was the old name) and search for 'Team':



Now select the relevant Team Services account, click Link button at the top, and then the Link button in the other blade:

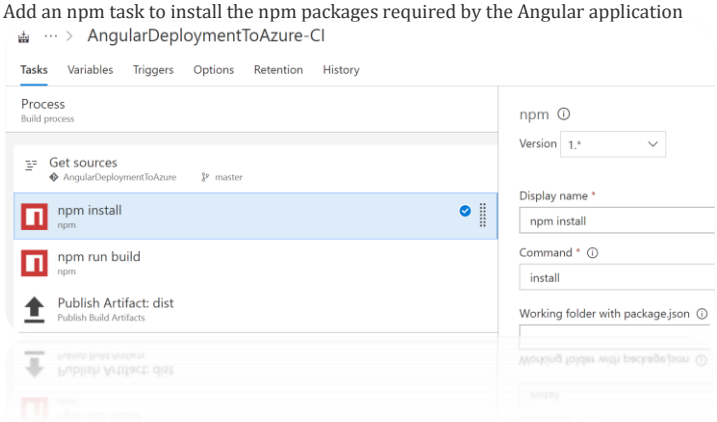


And you're done! You will now be able to set up continuous deploying to your git repos hosted in VSTS.

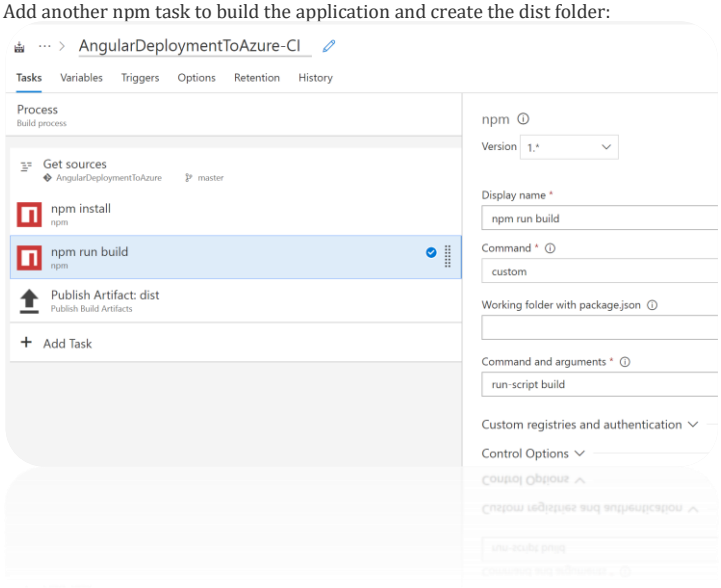
Setting Up CI Pipeline With VSTS

In the next steps we will set up our VSTS CI/CD pipeline to push the Angular application to the newly created Azure Web App. Start by creating a new build definition under VSTS:

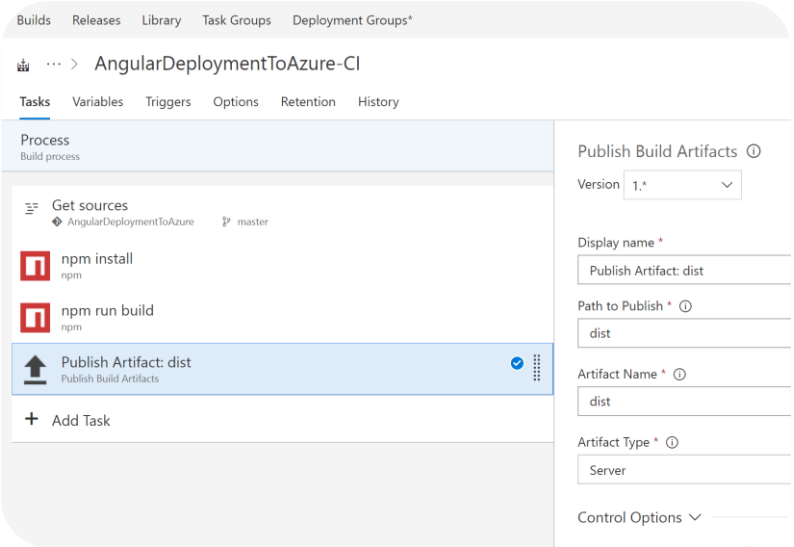
1. *Build and Release -> Builds -> New*
2. Add an npm task to install the npm packages required by the Angular application



3. Add another npm task to build the application and create the dist folder:

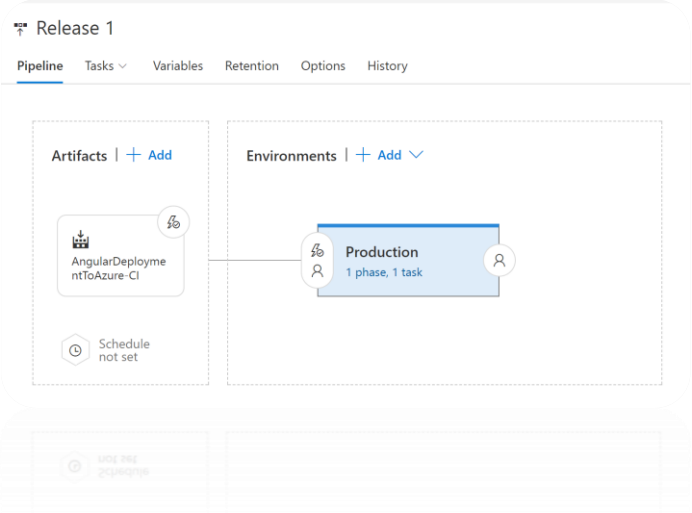


4. Add a publish artifact task that generates the dist artifact which will be provided later on as an input to our release definition:

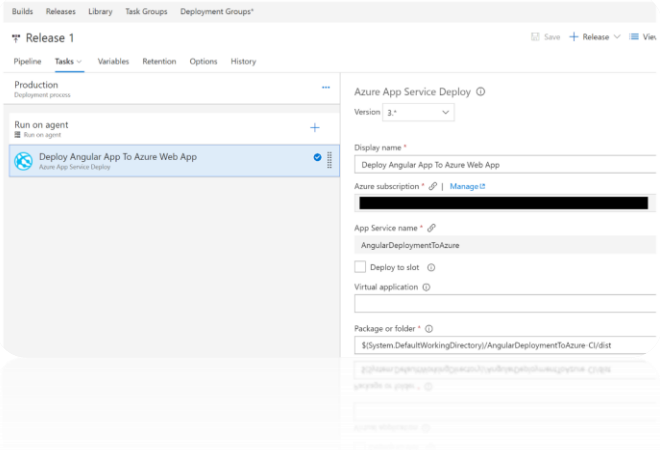


Setting Up CD Pipeline With VSTS

The last step is to add a CD pipeline which will deploy the artifacts created by the build to the Azure Web App. In this demo I am keeping the release pipeline simple by deploying the artifacts directly to production. In a real life application you will probably create multiple environments before releasing to production (Development, QA, Staging, etc.):



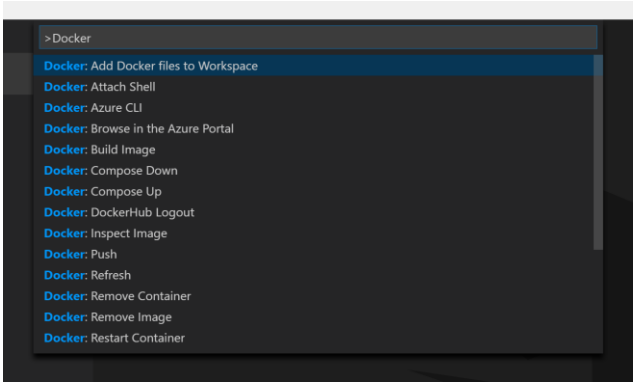
The production environment includes a single task that deploys the Angular application to an Azure Web App:



That's it! You now have a fully functional CI/CD pipeline that will deploy your Angular application to an Azure Web App the next time you check in your code.

Bonus: Deploy your SPA to a Linux VM

https://blogs.msdn.microsoft.com/wael-kdouh/2018/01/02/deploying-your-dockerized-angular-application-to-azure-using-vsts/



When Prompted Select: node.js and then set the Port to 4200