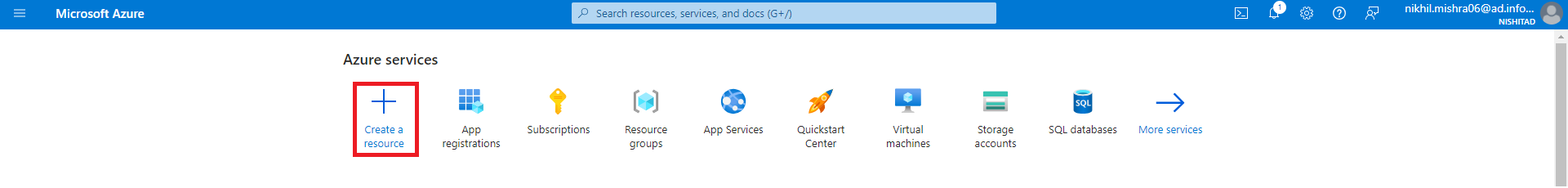
*Angular CI/CD Build and Release pipeline.*

Note: Upzip the AngularApp.zip folder provided with this document and use as template angular app for hosting.

**Section 1: Create Web Application on Azure.**

Before creating pipelines to deploy our code, we need to create a web app on azure. When you serve your angular project locally, a localhost port is made available to host angular files. We will create a similar server to host our angular application on the cloud with Azure web App.

1. On the Azure Portal, click on “Create a resource” or search for “Web App” in the search bar.



A screenshot of a computer

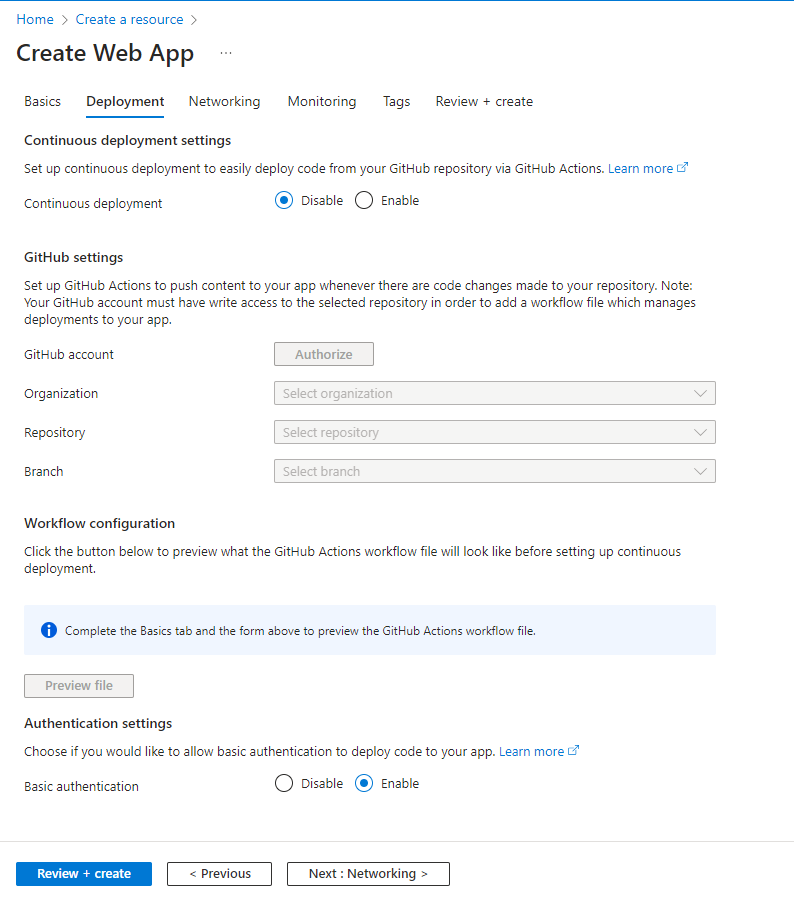
Description automatically generated

1. Fill in web app details:
   1. Subscription: - Name of subscription
   2. Resource Group – You can either create a resource group or select an existing one.
   3. Name: unique name for your webapp. The name of webapp will appear in the web app Url where your angular application is hosted.
   4. Publish: Select “code”
   5. Runtime stack: Select “Node LTS”.
   6. Operating System: Select “Windows”
   7. Region: Select your preferred location
   8. Under Pricing plans: Select existing plan or create a new plan and select appropriate tier.
   9. Under the deployment tab, enable “Basic Authentication”.

After Filling all the details, select “Review+create” and wait for deployment to complete.

A screenshot of a computer

Description automatically generated

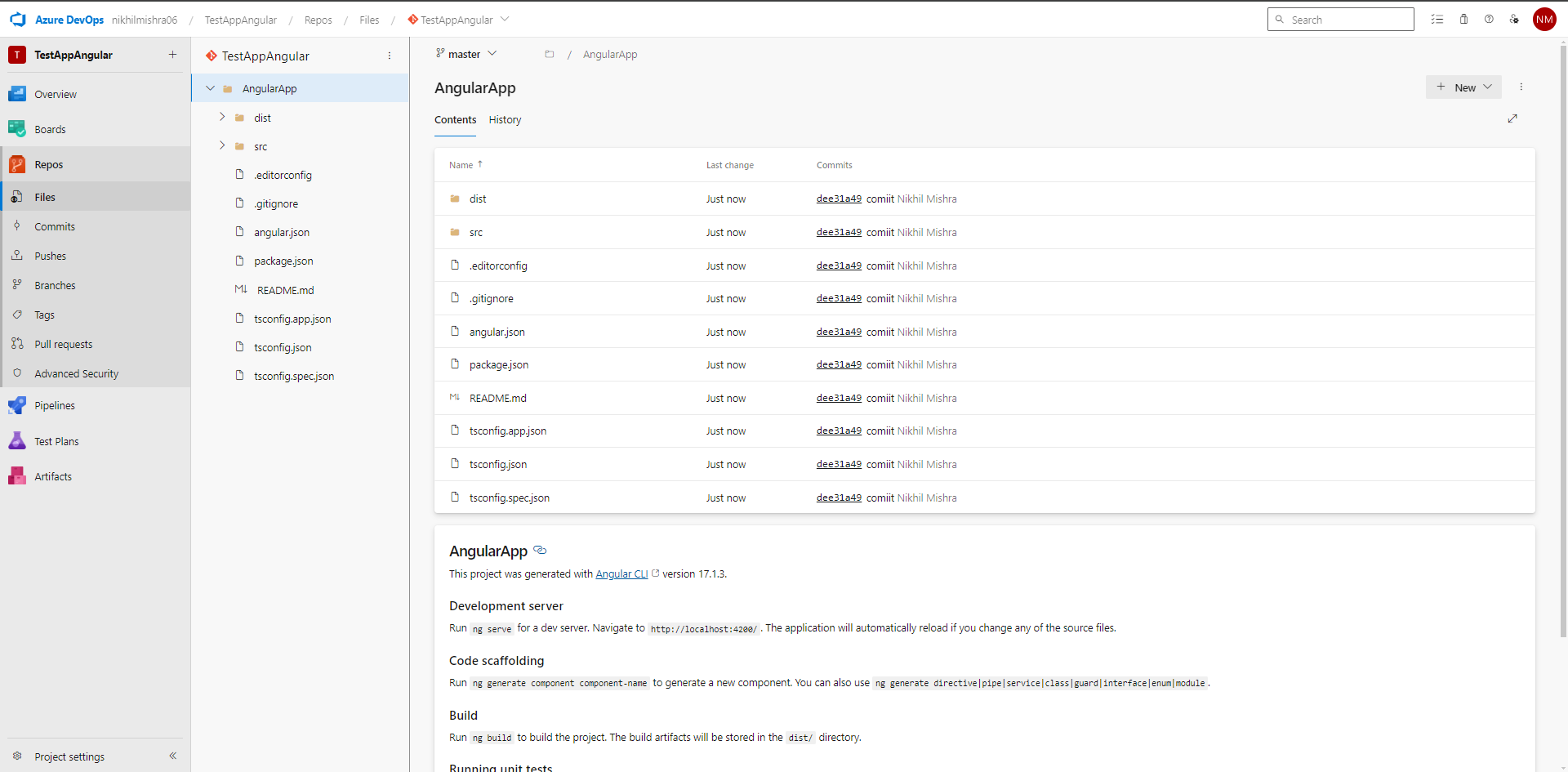


A screenshot of a computer

Description automatically generated

**Section 2: Establish repository.**

1. We require setting up a repository on devops that will store your code. When we establish a pipeline , Azure Devops performs certain tasks (e.g. build, install packages etc.) on the code present in repository.
   1. Later you’ll see that we will create a build and release pipeline , build - for installing packages, and building the application, release – for deploying the repository code to web app URL.
   2. Sign in to dev.azure.com. Create a new project or select any existing project and setup a repository for your angular application.
2. After establishing the repository, clone it using Visual Studio, use the “AngularApp.zip” folder, extract it and place the AngularApp (the folder which consists of package.json) directly under your repository.



**Section 3 : Create Build pipeline.**

1. Now in the same project, hover on Pipelines option under Repos and select “Pipelines”.

A screenshot of a video game

Description automatically generated

1. Click on ‘Create pipeline’. You will see a section to select the source of your operation. Click on “Use classic editor” .

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Configure following details:
2. Select a source – set “Azure Repos Git”.
3. Team project – select project from the dropdown.
4. Repository -- select repository from the dropdown.
5. Default branch – select “master” from dropdown.

Click on “Continue”.

A screenshot of a computer

Description automatically generated

1. In the next section, you can see all the project templates available. When you select a template, few pre-defined tasks are added to the pipeline. Since we don’t have a pre-defined template for angular , Select “Empty Job”.

A screenshot of a computer

Description automatically generated

1. Configure pipeline details.
2. Name – name of the pipeline.
3. Agent pool -- select “Azure pipelines” from the dropdown.
4. Agent Specification – select “windows-2019” from the dropdown.

A screenshot of a computer

Description automatically generated

1. On left hand side, click on “Agent Job 1” Configure “Display name” for the Agent Job. Click on the plus icon next to agent job display name.

A screenshot of a computer

Description automatically generated

1. In the “Add Tasks” section, search for “npm”. Click on “Add” button.

A screenshot of a computer

Description automatically generated

1. New Task is added to the job. Select the task and configure the following details:
2. Display name – enter the name of your task.
3. Command- mention the command you want to run. This command will be run on the repository code once you run the pipeline. ‘install’ is the default command. This task will install the npm packages required to build angular code.
4. Working folder that contains package.json – set according to your repository. Make sure you select folder where package.json on application is present.

A screenshot of a computer

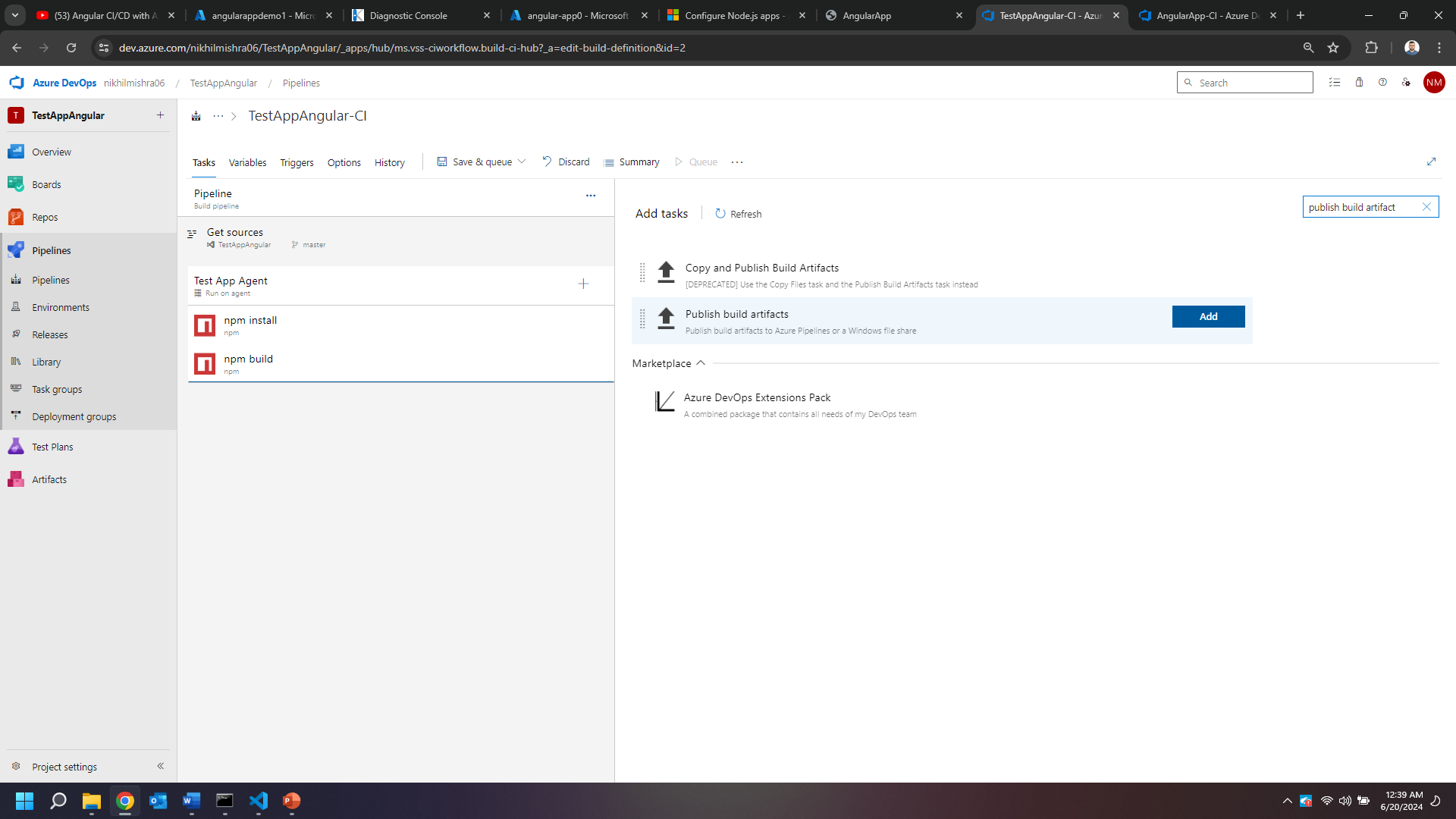
Description automatically generated

1. Add new task by repeating step 11. Configure the task:
2. Display name – enter the name of your task.
3. Command- select custom from dropdown.
4. Working folder that contains package.json – set according to your repository. Make sure you select folder where package.json on application is present.
5. Command and arguments – you will run the build command specified in package.json. type “run build-prod”.

A screenshot of a computer

Description automatically generated

1. Next, we will add another task to publish artifacts created after building the app. Click on “plus” icon and search for “Publish Build Artifacts. Click on “Add” button.



1. Configure the publish artifact task:
2. Display Name -- set display name for task.
3. Path to publish -- select “dist” folder.
4. Artifact name – set name of artifact.

A screenshot of a computer

Description automatically generated

1. In the triggers tab, select enable continuous integration. Select the master branch. Then Click on “Save and queue”.

A screenshot of a computer

Description automatically generated

1. Fill save comments and click on “Save and run”. Wait for your pipeline to complete. Check Logs in case of errors.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Hence, we have created a build pipeline. Next, lets create a release pipeline.

**Section 4: Create Release pipeline.**

1. Hover over pipelines, select releases. Click on ”New Pipeline”.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

1. Select “Azure App Service deployment” and click on “Apply”

A screenshot of a computer

Description automatically generated

1. Click on add artifact and add the artifact generated by CI pipeline. Configure following details:

A screenshot of a computer

Description automatically generated

1. Source type :- Build
2. Project :- Select appropriate project name.
3. Source(build pipeline) :- Select the build pipeline created previously.
4. Default version :- Select latest or any of the options according to your requirements.
5. Source alias :- alias name for source.
6. Click on “Add”. Click on “ ” icon and enable “Continuous deployment trigger”. This will create a release as soon as a new build is generated. Configure build branch filters as required.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Click on Stage 1 created in “stages” dialog. Set appropriate name for the Stage (e.g. dev, prod, etc.) and select stage owner. Click on the hyperlink “ 1 job, 1 task” to edit tasks mentioned in release.

A screenshot of a computer

Description automatically generated

Configure following details for stage :-

1. Azure subscription :- select available azure subscription(Verify/Authorize if required).
2. App type :- Select “Web App on Windows”.
3. App service name :- Select the web app created previously.

A screenshot of a computer

Description automatically generated

1. We will upload our built artifacts to the deployed web app. Configure the following details of the agent :-
2. Display name :- set name for the agent.
3. Under Agent selection, Agent Pool :- select “Azure Pipelines”.
4. Agent Specification :- select “windows-2019”.

A screenshot of a computer

Description automatically generated

1. Click on Azure App service Deploy and configure :
2. Display Name :- Set appropriate name .
3. Connection Type :- Select “Azure Resource Manager”.
4. Azure subscription :- Select the service connection that you created earlier.
5. App Service type :- Select “Web App on Windows”.
6. App service name :- Select app service name, refer to the web app name created in azure.
7. Check the “Deploy to Slot or App Service environment” checkbox.
8. Select the Resource Group
9. Set “Slot” to production.
10. Set “package or folder”to - “$(System.DefaultWorkingDirectory)/\_TestAppAngular-CI/app/angular-app” or browse and select the AppName folder under build artifact [ confirm that “browser” folder is present inside AppName folder]

A screenshot of a computer

Description automatically generated

1. Under “File transforms & Variable Substitution Options”, click on “” and click on “ok “ on the next dialog box that appears.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Under “ Additional Deployment method” , configure the parameters according to the image below.

A person holding a pen

Description automatically generated with medium confidence

1. Click on “save” and then on “ok”.

A screenshot of a computer

Description automatically generated

1. Click on “Create Release”.

A screenshot of a computer

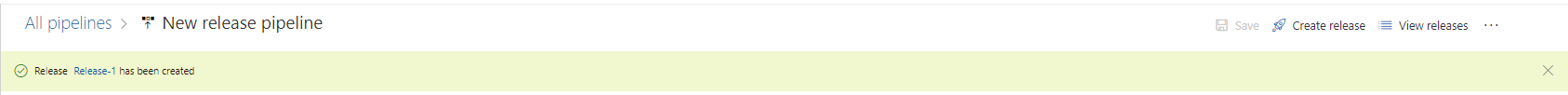
Description automatically generated

1. Click on “Create”.

A screenshot of a computer

Description automatically generated

1. You get a notification of release creation.



1. You can click on “release” hyperlink and then on “Logs” to check completion.

A white rectangular object with a dotted line

Description automatically generated

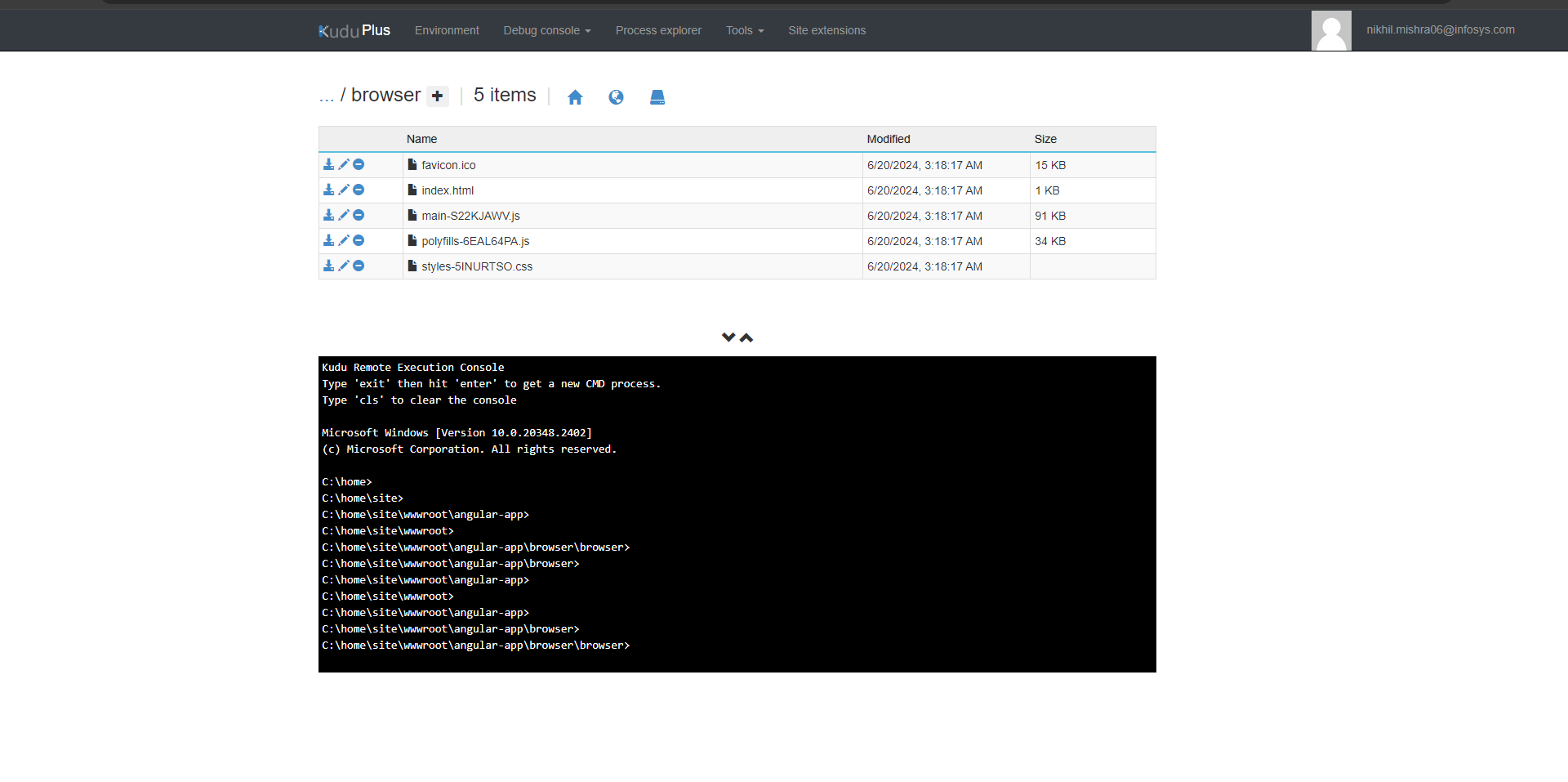
A screenshot of a computer

Description automatically generated

Hence, we have created a build and release pipeline.

**Section 5: Configure path for app access.**

1. Now we will see the built app. For this we have to configure the Web App that we created in Section 1. Go to Azure Portal. Open the web app created under your resource group.
2. In the overview tab, copy default domain. Update it by adding “.scm” after app name. E.g “angularappdemo1.scm.azurewebsites.net” and open it in a browser.
3. Click on “Debug Console>CMD” from the tabs that appear after previous link has loaded.



1. Browse through - site\wwwroot\angular-app\browser\browser
2. Copy the path after wwwroot. E.g – for given example copy – “\angular-app\browser\browser”.
3. Open the Web App on portal, and go to Setting>Configurations

A screenshot of a computer

Description automatically generated

1. Under the General setting tab make change to following properties:
2. Set the platform to “64 Bit”.
3. Change FTP state to “All Allowed”.
4. Go to the “Default Documents” tab and remove all documents except “index.html”.

A screenshot of a computer

Description automatically generated

1. Under the “Path Mapping” tab, under “Virtual Application and directories” , edit the physical path and add the path copied after the site\wwwroot.

A screenshot of a computer

Description automatically generated

1. Click on “Save”. Wait for operation to complete. Now Access the default Domain of app. You will see the following output.

A white background with black dots

Description automatically generated