

CS 423
Introduction To DevOps
Assignment # 2



Name: Syed Muhammad Ashhar Shah
Reg No: 2020478
Faculty: B(CS)

PROJECT INTRODUCTION

In this project we are going to automate the deployment process in which we have developer's who are working on developing the application and we have a QA-Engineer who is making sure that there are no bugs or issues in the underlying application.

For the deployment process we will be creating two separate servers, one will be for testing the application (for the QA and the Developers) and the other will be for staging the application (for the Client to view changes).

To automate this the Developers will first modify the code-base to add or remove features as per the requirements of the Client. Once done they will trigger a workflow which will deploy the changes to the testing server which can be viewed by the QA and the developer to see if there are any issues with the code-base and all the features work properly.

If there are no issues, the QA-Engineer will then initiate another workflow which will deploy all the changes to the staging server after which he can prompt the client to view said changes and provide his feedback.

This process will continue whenever there is a new feature being developed on the code-base being modified.

TASK COMPLETION

Task 1: Selection and deployment on localhost of any simple open-source application.

1. Clone the following simple open-source application from GitHub built using ReactJs and NodeJs.
 - o ReactNodeTesting app
 - o You may also clone any simple open-source application built using any of the following technologies but must discuss with me before proceeding:
 - Front-end: React or Angular
 - Back-end: NodeJs, Python, Java, PHP

To clone the repository we will use the command:

- *git clone <https://github.com/eljamaki01/ReactNodeTesting.git>*

The screenshot shows a dual-monitor setup. The left monitor displays a GitHub repository page for 'eljamaki01 / ReactNodeTesting'. The page includes details like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', and 'Insights'. It lists files such as 'vscode', 'public', 'src', '.ignore', 'LICENSE', 'README.md', 'api_endpoint.test.js', 'carts.js', 'index.js', and 'package.json'. The right monitor shows a terminal window with the command 'ls -la' executed, displaying a file list with permissions and timestamps. The terminal session is on a Linux system named 'ashhar@ashhar-Lenovo-B590'.

```
ashhar@ashhar-Lenovo-B590:~/Desktop/DevOps_Assignment_2/ReactNodeTesting$ ls -la
total 13
drwxr-x 6 ashhar ashhar 4096 Nov 26 10:39 .
drwxrwxr-x 3 ashhar ashhar 4096 Nov 26 10:39 ..
-rw-rw-r-- 1 ashhar ashhar 2292 Nov 26 10:39 api_endpoint.test.js
-rw-rw-r-- 1 ashhar ashhar 1361 Nov 26 10:39 carts.js
drwxrwxr-x 8 ashhar ashhar 4096 Nov 26 10:39 .git
-rw-rw-r-- 1 ashhar ashhar 1084 Nov 26 10:39 index.js
-rw-rw-r-- 1 ashhar ashhar 2103 Nov 26 10:39 README.md
-rw-rw-r-- 1 ashhar ashhar 1065 Nov 26 10:39 LICENSE
-rw-rw-r-- 1 ashhar ashhar 1065 Nov 26 10:39 package.json
drwxrwxr-x 2 ashhar ashhar 4096 Nov 26 10:39 .vscode
-rw-rw-r-- 1 ashhar ashhar 272 Nov 26 10:39 .gitignore
drwxrwxr-x 2 ashhar ashhar 4096 Nov 26 10:39 .gitkeep
drwxrwxr-x 2 ashhar ashhar 4096 Nov 26 10:39 .vscode
ashhar@ashhar-Lenovo-B590:~/Desktop/DevOps_Assignment_2/ReactNodeTesting$
```

This will clone the git repository onto our local system.

2. Clone the following simple open-source application from GitHub built using ReactJs and NodeJs.

To deploy the application on our system we will first install the node dependencies using the command:

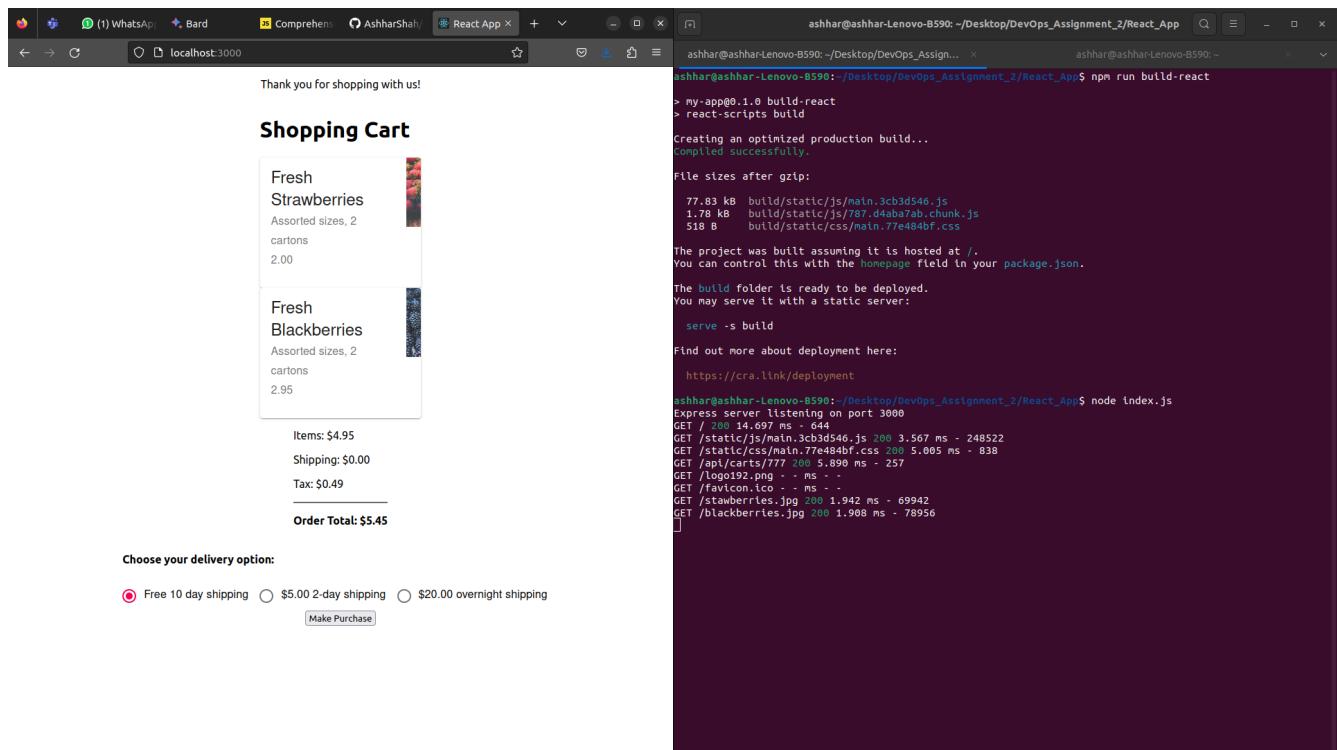
- *npm install*

Once we have installed all the required dependencies we will host our application using the command:

- *node index.js*

We can also use nodemon via the command:

- *npx nodemon index.js*



3. As you and your team will be making further changes to this application so you must initialize this cloned repository code to your new repository and push it to your GitHub repository. Add a second team member as a collaborator to your GitHub project.

Since I am working on this as a solo project, I will not be adding another collaborator to the repository. I will only add project to my own GitHub repository by creating a repository with the command:

- *gh repo create*

Then I will push the local repository to the GitHub servers using the commands:

- *git add .*
- *git commit -m "Adding Files"*
- *git push -u origin master*

The screenshot shows a GitHub repository page for 'DevOps_Assignment_2'. The commit history lists numerous 'create mode' entries for various files, mostly in the 'React_App' directory. The terminal window at the bottom shows the execution of a git push command, which fails because there is no upstream branch. It then shows the command to set the current branch as the upstream.

```
ashhar@ashhar-Lenovo-B590:~/Desktop/DevOps_Assignment_2$ git push
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master

ashhar@ashhar-Lenovo-B590:~/Desktop/DevOps_Assignment_2$ git push -u origin master
Enumerating objects: 33675, done.
Counting objects: 100% (33675/33675), done.
Delta compression using up to 4 threads
Compressing objects: 100% (32432/32432), done.
Writing objects: 100% (33675/33675), 40.48 MiB | 2.68 MiB/s, done.
Total 33675 (delta 6737), reused 0 (delta 0), pack-reused 0
remote: Compressing deltas using up to 4 threads...
remote: To https://github.com/AshharShah/DevOps_Assignment_2.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
ashhar@ashhar-Lenovo-B590:~/Desktop/DevOps_Assignment_2$
```

4. You must make your repository public and add me as a collaborator to that repository as well.

When we initialize the repository in part 3, we already specify that the repository is going to be public.

Link for verification:

https://github.com/AshharShah/DevOps_Assignment_2

The screenshot shows a GitHub repository page for 'DevOps_Assignment_2'. The repository is public and owned by 'AshharShah'. The main interface includes a code editor, issue and pull request tabs, and navigation links for Projects, Wiki, Security, Insights, and Settings. The repository has 1 branch (master) and 0 tags. A recent commit by 'AshharShah' is listed, showing the addition of a Next Workflow. The commit message is 'Added Next Workflow'. The commit was pushed yesterday at 8766ef2. There are 30 commits in total. The repository has 0 stars, 1 watching, and 0 forks. It also has no releases published. A 'README' button is available to help people understand the project. The Languages section shows a horizontal bar with small segments of different colors, indicating the programming languages used in the repository.

5. Update your repository settings such that no collaborator should be able to commit directly to the main branch.

To restrict any developer from pushing directly to the main branch. We will define a branch rule in the repository setting which disables the option to push to the master branch directly without a pull request.

The screenshot shows a browser window with multiple tabs open. The active tab is 'Settings - Branches - Branch protection rules' for the repository 'AshharShah / DevOps_Assignment_2'. The left sidebar has sections for General, Access, Collaborators, Moderation options, Code and automation (with 'Branches' selected), Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages, Security (Code security analysis, Deploy keys, Secrets and variables), and Integrations (GitHub Apps, Email notifications). The main content area is titled 'Branch protection rule' and shows a configuration for the 'master' branch. It includes a 'Branch name pattern' input field containing 'master', a section for 'Applies to 1 branch' (set to 'master'), and a 'Protect matching branches' section. Under 'Protect matching branches', the 'Require status checks to pass before merging' checkbox is checked, while others like 'Require a pull request before merging', 'Require branches to be up to date before merging', 'Require conversation resolution before merging', 'Require signed commits', and 'Require linear history' are unchecked. A note states 'No status checks found' with a message: 'Sorry, we couldn't find any status checks in the last week for this repository.' and a link 'Learn more about status checks'.

Task 2: Create two EC2 instances on AWS for Testing and Staging environment.

1. Give an appropriate name such that it represents environment.

The instances will have the names:

- *Testing Server – DevOps*
- *Staging Server – DevOps*

The image shows two separate browser windows side-by-side, both displaying the 'Instance details' page for AWS EC2 instances. The left window shows the 'Testing Server - DevOps' instance (ID: i-08b756cdf294e107d), and the right window shows the 'Staging Server - DevOps' instance (ID: i-07d0dabc050c22a6a). Both instances are currently stopped. The details listed include their respective Public IPv4 addresses (54.210.17.12 and 3.232.126.178), Private IPv4 addresses (172.31.23.33 and 172.31.16.210), and Public IPv4 DNS names (ec2-54-210-17-12.compute-1.amazonaws.com and ec2-3-232-126-178.compute-1.amazonaws.com). Other fields like VPC ID, IAM Role, and Auto Scaling Group name are also visible.

2. Each environment Amazon Machine Image (AMI) should be Ubuntu Server 22.04 LTS (HVM)

Both the servers are of t2.micro instances both running the Ubuntu operating system of version 22.04 LTS which can be seen in the screenshot below.

The image displays two separate screenshots of the AWS EC2 Instance Details page, side-by-side. Both instances are of the t2.micro type and are running the Ubuntu 22.04 LTS (HVM) operating system. The left instance has an AMI ID of ami-0fc5d935ebf8bc3bc and was launched on Wednesday, November 29, 2023, at 14:41:38 GMT+0500 (Pakistan Standard Time). The right instance has an AMI ID of ami-0fc5d935ebf8bc3bc and was also launched on the same date and time. Both instances have the same configuration, including the same AMI name (ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230919), launch time, and other parameters like Stop protection, Instance auto-recovery, and Credit specification.

Parameter	Left Instance Value	Right Instance Value
Platform	Ubuntu (Inferred)	Ubuntu (Inferred)
Platform details	Linux/UNIX	Linux/UNIX
Stop protection	Disabled	Disabled
Instance auto-recovery	Default	Default
AMI Launch index	0	0
Credit specification	standard	standard
Usage operation	RunInstances	RunInstances
Enclaves Support	-	-
Allow tags in instance metadata	Use RBN as guest OS hostname	Use RBN as guest OS hostname
AMI ID	ami-0fc5d935ebf8bc3bc	ami-0fc5d935ebf8bc3bc
AMI name	ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230919	ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230919
Launch time	Wed Nov 29 2023 14:41:38 GMT+0500 (Pakistan Standard Time) (1 day)	Wed Nov 29 2023 14:27:51 GMT+0500 (Pakistan Standard Time) (1 day)
AMI location	amazon/ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230919	amazon/ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230919
Lifecycle	normal	normal
Stop-hibernate behavior	Disabled	Disabled
State transition reason	User initiated (2023-11-29 13:46:24 GMT)	User initiated (2023-11-29 13:46:24 GMT)
Kernel ID	-	-
State transition message	Client.UserInitiatedShutdown: User initiated shutdown	Client.UserInitiatedShutdown: User initiated shutdown
Owner	875109026501	875109026501
RAM disk ID	-	-
Current instance boot mode	legacy-bios	legacy-bios
Answer RBN DNS hostname IPv4		
Monitoring	disabled	disabled
Termination protection	Disabled	Disabled
AMI location	amazon/ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230919	amazon/ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20230919
Launch time	Wed Nov 29 2023 14:27:51 GMT+0500 (Pakistan Standard Time) (1 day)	Wed Nov 29 2023 14:27:51 GMT+0500 (Pakistan Standard Time) (1 day)
Lifecycle	normal	normal
Stop-hibernate behavior	Disabled	Disabled
State transition reason	User initiated (2023-11-29 13:46:24 GMT)	User initiated (2023-11-29 13:46:24 GMT)
Kernel ID	-	-
State transition message	Client.UserInitiatedShutdown: User initiated shutdown	Client.UserInitiatedShutdown: User initiated shutdown
Owner	875109026501	875109026501
RAM disk ID	-	-
Current instance boot mode	legacy-bios	legacy-bios
Answer RBN DNS hostname IPv4	Enabled	Enabled

3. All instances must share one common security group such that if you add/remove any rule, it should affect all instances.

Both the instances have the same security group:

- sg-0b31ee315977fd7c2

The security group has the following rules:

Rule ID	Port	Protocol	Source
a) sgr-0268e78ec9804cf3f	22	TCP	0.0.0.0/0 (SSH)
b) sgr-0558b8e1bb6108e2a	80	TCP	0.0.0.0/0 (HTTP)

The image displays two separate AWS EC2 instance detail pages side-by-side. Both pages are for instances in the 'us-east-1' region and show the 'Security' tab selected. In the 'Security groups' section, both instances are listed under the same security group: 'sg-0b31ee315977fd7c2 (default)'. The 'Inbound rules' table for both instances shows the following rules:

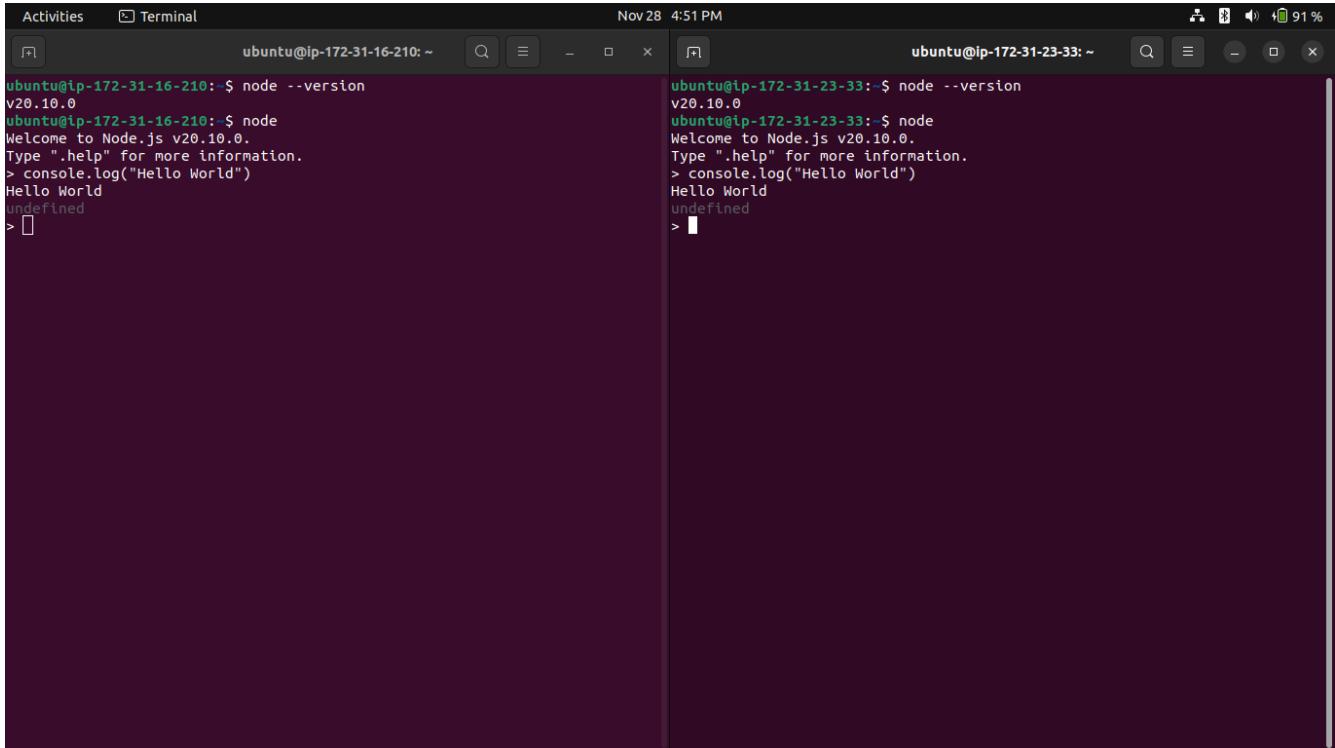
Security group rule ID	Port range	Protocol	Source
sgr-0268e78ec9804cf3f	22	TCP	0.0.0.0/0
sgr-0558b8e1bb6108e2a	80	TCP	0.0.0.0/0

4. Configure your environments such that it includes required dependencies for your React and Nodejs project.

To configure our environment we will need to install NodeJs along with Node Package Manager (npm) on our AWS instances.

To do this we will simply SSH into the instances and run the following set of commands:

- *sudo apt-get update*
- *sudo apt-get install -y ca-certificates curl gnupg*
- *sudo mkdir -p /etc/apt/keyrings*
- *curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key | sudo gpg --dearmor -o /etc/apt/keyrings/nodesource.gpg*
- *NODE_MAJOR=20*
- *echo "deb [signed-by=/etc/apt/keyrings/nodesource.gpg] https://deb.nodesource.com/node_\$NODE_MAJOR.x nodistro main" | sudo tee /etc/apt/sources.list.d/nodesource.list*
- *sudo apt-get update*
- *sudo apt-get install nodejs -y*



The screenshot shows two terminal windows side-by-side. Both windows are titled 'Terminal' and show the command 'node --version' being run. The left window is for instance 'ip-172-31-16-210' and shows the output: 'v20.10.0'. The right window is for instance 'ip-172-31-23-33' and also shows 'v20.10.0'. Both terminals also show a Node.js REPL session where 'Hello World' is logged to the console.

```
Activities Terminal Nov 28 4:51 PM ubuntu@ip-172-31-16-210: ~
ubuntu@ip-172-31-16-210: $ node --version
v20.10.0
ubuntu@ip-172-31-16-210: $ node
Welcome to Node.js v20.10.0.
Type ".help" for more information.
> console.log("Hello World")
Hello World
undefined
> 

ubuntu@ip-172-31-23-33: ~
ubuntu@ip-172-31-23-33: ~$ node --version
v20.10.0
ubuntu@ip-172-31-23-33: ~$ node
Welcome to Node.js v20.10.0.
Type ".help" for more information.
> console.log("Hello World")
Hello World
undefined
> 
```

5. Add or remove rules in your security group where required.

Since we only want to SSH into the web server and host an application. We will only enable the SHH and HTTP ports in the security group settings.

The screenshot shows the AWS EC2 console with the URL <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules>. The page is titled "Edit inbound rules". It displays two security group rules:

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
sgr-0268e78ec9804cf3f	SSH	TCP	22	Custom	0.0.0.0/0
sgr-0558b8e1bb6108e2a	HTTP	TCP	80	Custom	0.0.0.0/0

A warning message at the bottom states: "⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." There are "Cancel", "Preview changes", and "Save rules" buttons at the bottom right.

6. As you will be working in a group you may create first AWS instance in one account and second in other account.

Since this is a solo project, I will be creating both EC2 instances on the same account which is my personal AWS account.

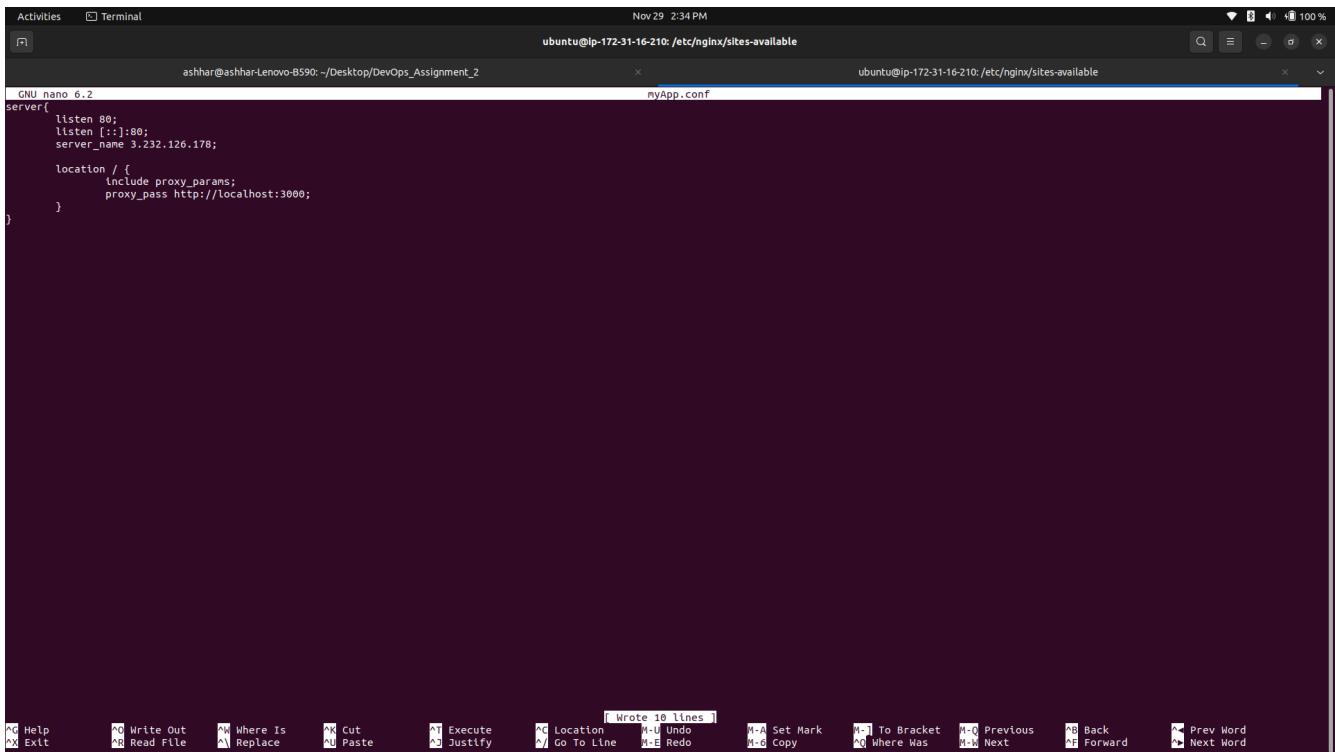
Also to host the Node/React application I will be using a Nginx HTTP server.

To install nginx on the instances I will use the command:

- *sudo apt install nginx*

Then I will configure the nginx server to point toward by NodeJs application that runs on the localhost:3000. To do this we will first create a config file in the sites-available folder and link it to the folder sites-enabled. To do this we will use the following commands:

- *sudo touch /etc/nginx/sites-available/myApp.conf*
- *sudo ln /etc/nginx/sites-available/myApp.conf /etc/nginx/sites-enabled*
- *sudo nginx -t*
- *sudo systemctl restart nginx.service*



The screenshot shows a terminal window with two tabs open. The left tab is titled 'myApp.conf' and contains the following Nginx configuration code:

```
server{
    listen 80;
    listen [::]:80;
    server_name 3.232.126.178;

    location / {
        include proxy_params;
        proxy_pass http://localhost:3000;
    }
}
```

The right tab is titled 'ubuntu@ip-172-31-16-210:/etc/nginx/sites-available' and shows the command being run: 'sudo nano 6.2'. A status bar at the bottom indicates 'Wrote 10 lines.'

Then to start the application we will simply navigate into the react application's folder then use the command:

- *node index.js*

OR

- *npx nodemon index.js*

The screenshot shows a Firefox browser window with two tabs. The active tab displays a shopping cart page for a fruit store. The page includes a header 'Thank you for shopping with us!', a section titled 'Shopping Cart' showing two items: 'Yummy Strawberries' and 'Yummy Blackberries', and a summary at the bottom with 'Order Total: \$5.45'. Below the cart, there is a delivery option section with three radio buttons: 'Free 10 day shipping' (selected), '\$5.00 2-day shipping', and '\$20.00 overnight shipping'. A 'Make Purchase' button is at the bottom. The second tab in the browser shows a terminal session on an Ubuntu system (ip-172-31-16-210). The terminal output details the deployment process, starting with cloning the repository from GitHub, navigating to the React app directory, creating a new Nginx configuration file, linking it, and finally restarting the Nginx service. The terminal also shows the execution of 'npx nodemon index.js' to start the React application.

```
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ git clone https://github.com/AshharShah/DevOps_Assignment_2.git
Cloning into 'DevOps_Assignment_2'...
remote: Enumerating objects: 34027, done.
remote: Counting objects: 100% (91/91), done.
remote: Compressing objects: 100% (60/60), done.
remote: Total 34027 (delta 22), reused 76 (delta 18), pack-reused 33936
Receiving objects: 100% (34027/34027), 43.10 MB | 16.71 MB/s, done.
Resolving deltas: 100% (6804/6804), done.
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ cd /etc/nginx/sites-available/
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ nano myApp.conf
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ ls
default myApp.conf
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ sudo ln /etc/nginx/sites-available/ /etc/nginx/sites-available/myApp.conf
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ sudo ln -s /etc/nginx/sites-available/myApp.conf /etc/nginx/sites-available/default
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ sudo systemctl restart nginx
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ cd ~/DevOps_Assignment_2/React_App/
ashhar@ashhar-Lenovo-B590:~/DevOps_Assignment_2/React_App$ npx nodemon index.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Express server listening on port 3000
GET / 200 15.419 ms - 644
GET /static/js/main.3ch3d546.js 200 7.146 ms - 248522
GET /static/css/main.77e48abf.css 200 3.340 ms - 838
GET /logo192.png 200 1.699 ms - 5347
GET /api/carts/777 200 3.474 ms - 257
GET /strawberries.jpg 200 2.056 ms - 69942
GET /blackberries.jpg 200 1.932 ms - 78956
```

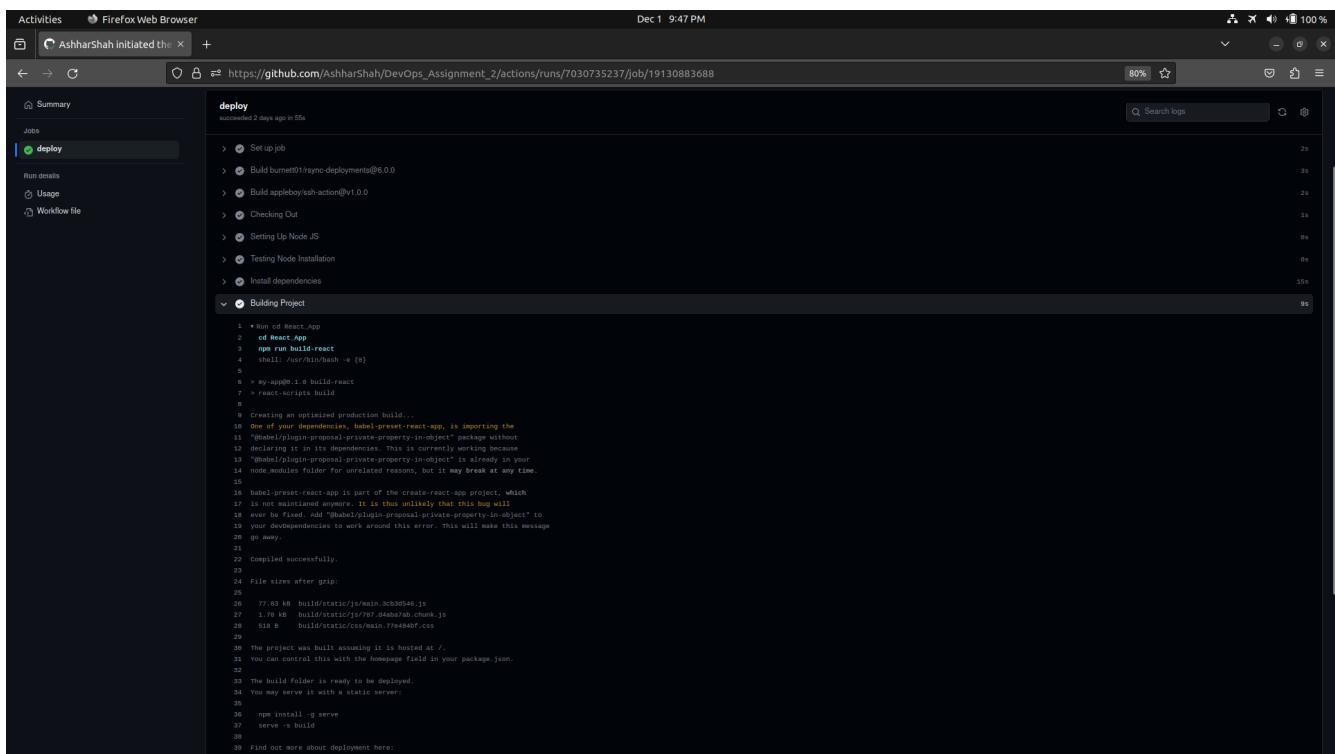
Task 3: Automating application from development to deployment.

1. You or your team members will always create features or fix branches from the main branch to add something new or update anything existing in future. Create a pull request, once done with the changes on the feature or fix branch. This pull request must trigger a workflow in your GitHub actions which will deploy changes for QA to test on AWS testing server. There should also be a button on your workflow to trigger this workflow without an pull requests if needed. QA should be able to access the deployed project using the link: http://<Instance_IP>. Your workflow must do the following:

- (1) Build your project: Compile the source code and build the application or software. This step ensures that the code can be successfully compiled.

To build the project we will simply navigate into the directory and run the command:

- *npm run build-react*

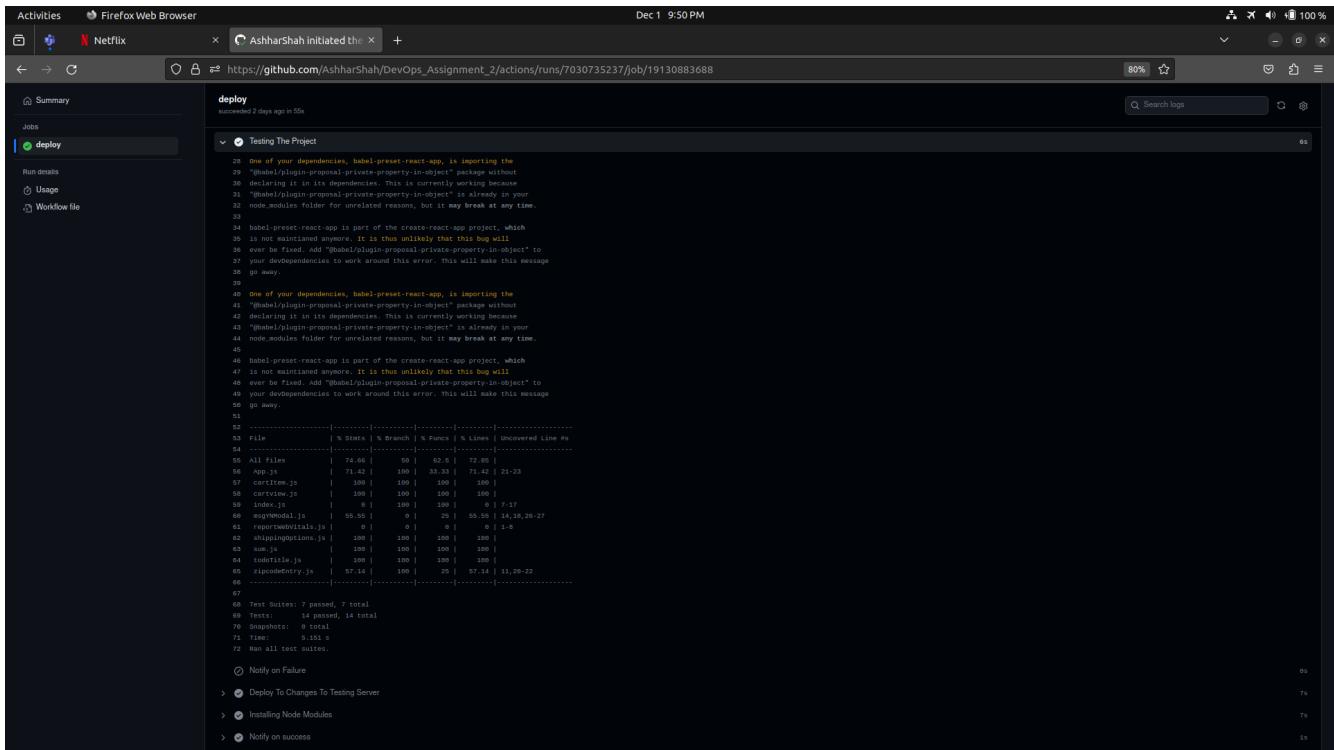


```
Activity Firefox Web Browser
AshharShah initiated the workflow
https://github.com/AshharShah/DevOps_Assignment_2/actions/runs/7030735237/jobs/19130883688
Summary
Jobs
deploy succeeded 2 days ago in 55s
  Set up job
  Build burnet01/sync-deployments@6.0.0
  Build appleboy/ssh-action@v1.0.0
  Checking Out
  Setting Up Node JS
  Testing Node Installation
  Install dependencies
    Building Project
      1 * Detected React App
      2 cd React_App
      3 npm run build-react
      4 shell: /usr/bin/bash --executed
      5
      6 × npx -y react-app -t & npx build-react
      7 > react-scripts build
      8
      9 Creating an optimized production build...
      10 One of the dependencies, babel-plugin-react-app, is importing the
      11 "babel/plugin-proposal-private-property-in-object" package without
      12 declaring it in its dependencies. This is currently working because
      13 "babel/plugin-proposal-private-property-in-object" is already in your
      14 node_modules folder for unrelated reasons, but it may break at any time.
      15
      16 babel-plugin-react-app is part of the create-react-app project,
      17 which is no longer maintained. It is thus unlikely that this bug will
      18 ever be fixed. Add "babel/plugin-proposal-private-property-in-object" to
      19 your dependencies to work around this error. This will make this message
      20 go away.
      21
      22 Compiled successfully.
      23
      24 File sizes after gzip:
      25
      26 77.83 kB build/static/js/main.3cb3d846.js
      27 1.78 kB build/static/js/787d4ab9ab.chunk.js
      28 518 B build/static/css/main.7f84d4bf.css
      29
      30 The project was built assuming it is hosted at /.
      31 You can control this with the homepage field in your package.json.
      32
      33 The build folder is ready to be deployed.
      34 You may serve it with a static server.
      35
      36 npm install -g serve
      37 serve -s build
      38
      39 Find out more about deployment here:
```

(2) Unit Testing: Run unit tests to verify that individual components or functions of the code work as expected. Unit tests are typically fast and focused on specific functionalities. (you must choose a project which has a few unit test cases, or you must create your own).

To run the test cases on our react we will use the following command in the workflow:

- *npm run test-react*



The screenshot shows a Firefox browser window with the URL https://github.com/AshharShah/DevOps_Assignment_2/actions/runs/7030735237/job/19130883688. The page displays the results of a GitHub Actions run titled 'deploy' which succeeded 2 days ago in 5s. The 'Testing The Project' section shows the output of the 'test-react' command. The log includes several warning messages from 'babel-preset-react-app' about deprecated 'private-property-in-object' syntax. It also provides detailed coverage reports for various files like App.js, CartItem.js, CartView.js, index.js, AsyncModule.js, reportWorkflow.js, and ZipcodeEntry.js, showing coverage percentages for statements, branches, and lines. The final summary indicates 7 tests passed, 14 total, and 0 snapshots.

```
One of your dependencies, babel-preset-react-app, is importing the "babel/plugin-proposal-private-property-in-object" package without declaring it as an its dependencies. This is currently working because "babel/plugin-proposal-private-property-in-object" is already in your node_modules folder for unrelated reasons, but it may break at any time.
...
One of your dependencies, babel-preset-react-app, is importing the "babel/plugin-proposal-private-property-in-object" package without declaring it as an its dependencies. This is currently working because "babel/plugin-proposal-private-property-in-object" is already in your node_modules folder for unrelated reasons, but it may break at any time.
...
babel-preset-react-app is part of the create-react-app project, which is not maintained anymore. It is thus unlikely that this bug will ever be fixed. Add "babel/plugin-proposal-private-property-in-object" to your dependencies to work around this error. This will make this message go away.
...
babel-preset-react-app is part of the create-react-app project, which is not maintained anymore. It is thus unlikely that this bug will ever be fixed. Add "babel/plugin-proposal-private-property-in-object" to your dependencies to work around this error. This will make this message go away.
...
File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----
All files | 74.66 | 60 | 62.5 | 77.06 |
App.js | 71.43 | 100 | 33.33 | 71.43 | 21-23
CartItem.js | 100 | 100 | 100 | 100 |
CartView.js | 100 | 100 | 100 | 100 |
index.js | 0 | 100 | 100 | 0 | 7-17
AsyncModule.js | 55.55 | 0 | 25 | 55.55 | 14,16,26-27
reportWorkflow.js | 0 | 0 | 0 | 0 | 1-9
AsyncModuleData.js | 100 | 100 | 100 | 100 |
Vue.js | 100 | 100 | 100 | 100 |
TodoTitle.js | 100 | 100 | 100 | 100 |
ZipcodeEntry.js | 57.14 | 100 | 25 | 57.14 | 11,29-29
...
Test Suites: 1 passed, 7 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        1:16.5
Ran all test suites.
```

(3) Code Analysis/Linting: Perform static code analysis or linting to check for code quality, adherence to coding standards, and potential issues. This step helps maintain a consistent and high-quality code-base.

In the application, there is no linting setup so we will configure eslint in our react application. To do so I have used the following [article](#).

Activities Firefox Web Browser Dec 1 10:21:PM

Netflix https://github.com/AshharShah/DevOps_Assignment_2/actions/runs/7063108006/job/19228368886 80%     

Summary

deploy (Started 1m 38s ago) 166 78 5s

Jobs

deploy

Run details

Usage

Workflow file

deploy

Install dependencies

Building Project

Testing The Project

PASS src/cartview.test.js
PASS src/czipcodeentry.test.js
PASS src/app.mock.test.js
PASS src/dam.test.js
PASS src/appmock.test.js
PASS src/app.test.js
PASS src/todotitle.test.js
One of your dependencies, babel-preset-react-app, is importing the "babel/plugin-proposal-private-property-in-object" package without specifying a version. This can lead to bugs when someone else uses "babel/plugin-proposal-private-property-in-object" is already in your node_modules folder for unrelated reasons, but it may break at any time. babel-preset-react-app is part of the create-react-app project, which is not maintained anymore. It is the unlikely this bug will be fixed. Add "babel/plugin-proposal-private-property-in-object" to your dependencies to work around this error. This will make this message go away.
27
A worker process has failed to exit gracefully and has been force exited. This is likely caused by tests leaking due to improper teardown. Try running with --detectOpenHandles to find leaks. Active timers can also cause this, ensure that .unref() was called on them.
File % Status | N Branch | N Funcs | N Lines | Uncovered Line #s

All files | 74.66 | 50 | 62.5 | 72.85 |
App.js | 71.42 | 100 | 53.33 | 71.42 | 21-23
CartView.js | 100 | 100 | 100 | 100 |
Cartview.js | 100 | 100 | 100 | 100 |
index.js | 0 | 100 | 100 | 0 | 7-17
asymmod.js | 55.55 | 0 | 25 | 55.55 | 14,18,26-27
reporterxitals.js | 0 | 0 | 0 | 0 | 1-8
asymmodoptions.js | 100 | 100 | 100 | 100 |
view.js | 100 | 100 | 100 | 100 |
todoTitle.js | 100 | 100 | 100 | 100 |
ticodeentry.js | 67.14 | 100 | 25 | 67.14 | 11,20-22

Test Suites: 14 passed, 0 total
Tests: 14 passed, 0 total
S_snapshots: 0 total
Time: 5.000 s
Ran all test suites.

Linking In The Project

Run on React_App

Warning: React version not specified in eslint-plugin-react settings. See <https://github.com/rx-eslint/eslint-plugin-react#configuration>.

Notify on Failure

Deploy To Changes To Testing Server

(4) Notification: Your workflow must send an email to me and the developer who created a pull request if the workflow fails. Moreover, it must send an email to the QA (to me) if it successfully deploys your change on the instance with all details how the QA can access the deployed app feature for testing.

Failure:

The screenshot displays two browser windows side-by-side. The left window is a GitHub Actions logs page for a workflow named 'Deploy To Staging Server (EC2)'. It shows a single job named 'deploy' that failed 7 minutes ago. The logs detail the workflow steps: Set up job, Build, Check out, Setting Up Node JS, Testing Node Installation, Install dependencies, Building Project, Testing The Project, Linting In The Project, Deploy To Changes To Testing Server, Installing Node Modules, Notify on success, and Notify on Failure. The 'Notify on Failure' step shows the command `run dwdiddo/action-send-mail@v3` with its configuration. The right window is a Gmail inbox showing an email from 'ashhar.shah786@gmail.com' titled 'Github Action Failure'. The email body contains the message: 'Workflow of AshharShah/DevOps_Assignment_2 initiated by AshharShah failed, kindly check the workflow logs!'. Below the email are standard Gmail controls for reply and forward.

Success:

The left window displays the GitHub Actions logs for a 'deploy' job. The logs show the following steps and their durations:

- Set up job: 2s
- Build burnettt/sync-deployments@6.0.0: 2s
- Build appleboy/ssh-action@v1.0.0: 2s
- Checking Out: 1s
- Setting Up Node JS: 3s
- Testing Node Installation: 6s
- Install dependencies: 16s
- Building Project: 7s
- Testing The Project: 6s
- Lining In The Project: 1s
- Deploy To Changes To Testing Server: 54s
- Installing Node Modules: 18s

The total duration is 1 minute 49s. A 'Notify on success' checkbox is checked at the bottom.

The right window shows an email in the Gmail inbox with the subject 'Github Action Sucess'. The email content is:

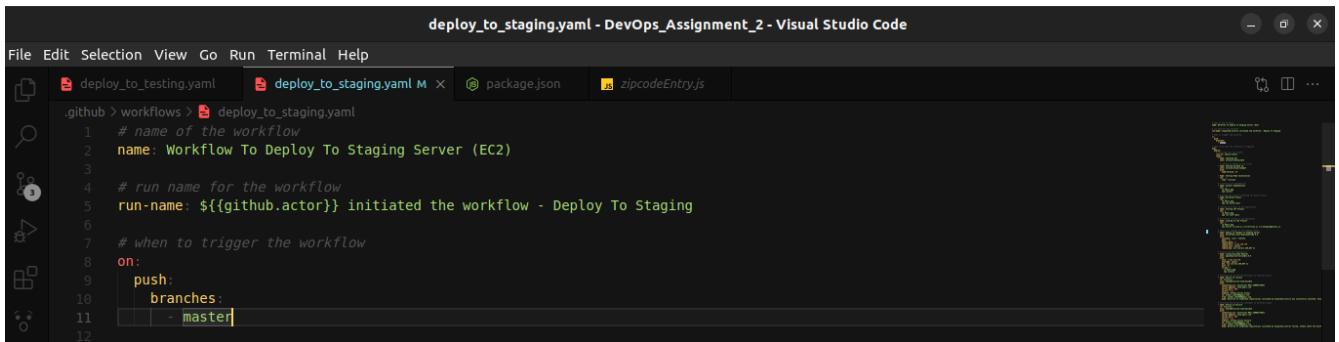
Workflow of AshharShah/DevOps_Assignment_2 initiated by AshharShah was successfully executed. View the changes on the server <http://3.232.126.178/>

Buttons for 'Reply' and 'Forward' are visible below the email.

2. Assume that the QA validates all the changes made in a Testing environment and everything looks fine. Now let's say QA confirms the pull request and merges the changes into master/main branch. This push must trigger a workflow in your GitHub actions which will deploy changes for client and team members to an AWS staging server. There should also be a button on your workflow to trigger this workflow without any push event if needed. All users should be able to access the deployed project using the link: http://<Instance_IP>. Your workflow must do all the tasks you did in your previous workflow.

For this we will create another workflow by which implements the same changes but works only when we perform a push to the main branch which is initiated by a pull-request's validation.

The rest of the code will be the same, we will simply change the server to and the “on” statement in the workflow.



The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Tab Bar:** deploy_to_testing.yaml, deploy_to_staging.yaml (highlighted), package.json, zipcodeEntry.js
- Code Editor:** The deploy_to_staging.yaml file contains the following YAML code:

```
github > workflows > deploy_to_staging.yaml
1  # name of the workflow
2  name: Workflow To Deploy To Staging Server (EC2)
3
4  # run name for the workflow
5  run-name: ${github.actor} initiated the workflow - Deploy To Staging
6
7  # when to trigger the workflow
8  on:
9    push:
10      branches:
11        - master
```
- Right Panel:** A tree view showing the repository structure, including .gitignore, .github, .vscode, README.md, requirements.txt, and setup.py.

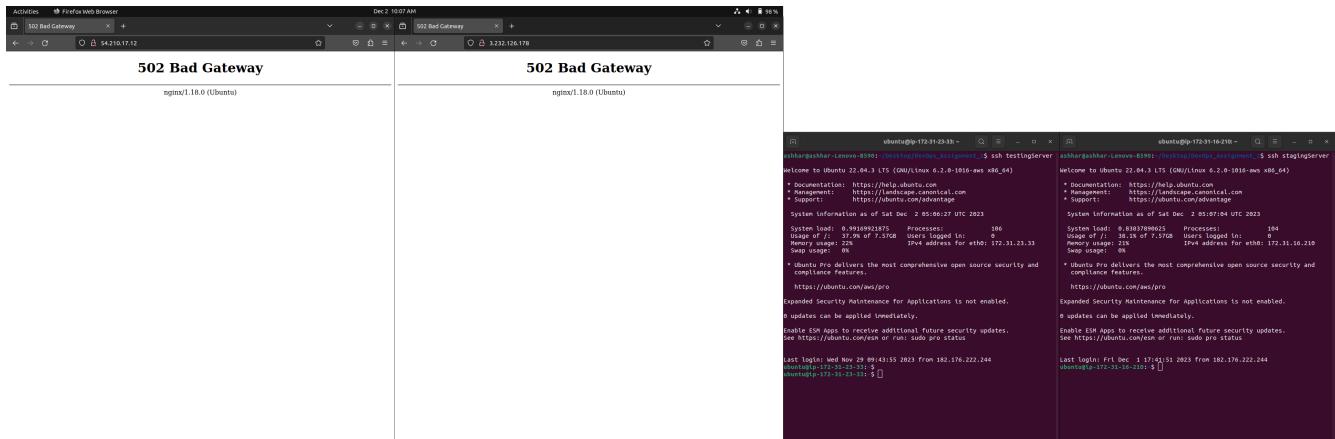
WORKFLOW DEMONSTRATION

Now that we have looked at how to do the tasks that were mentioned in the assignment, lets take a look at a step by step demonstration that shows how the workflow works to automate the deployment process.

NOTE: The browser window on the Left represents the Testing Server while the browser window on the Right represents the Staging Server (SAME FOR THE CLI WINDOWS)

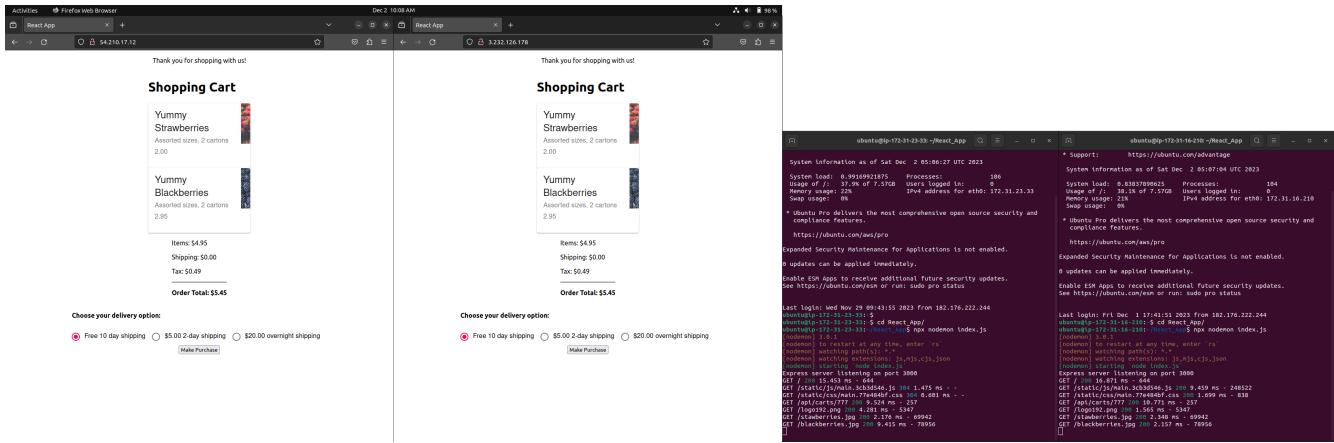
1. Starting The Server

- Our first step will be to start the EC2 instance's so that we can connect to them using our public IP's.
- Initially we would get a error 502 which represents Bad Gateway, this is because Nginx is configured to point to localhost:3000. However there is no application running on that specified port at the moment.



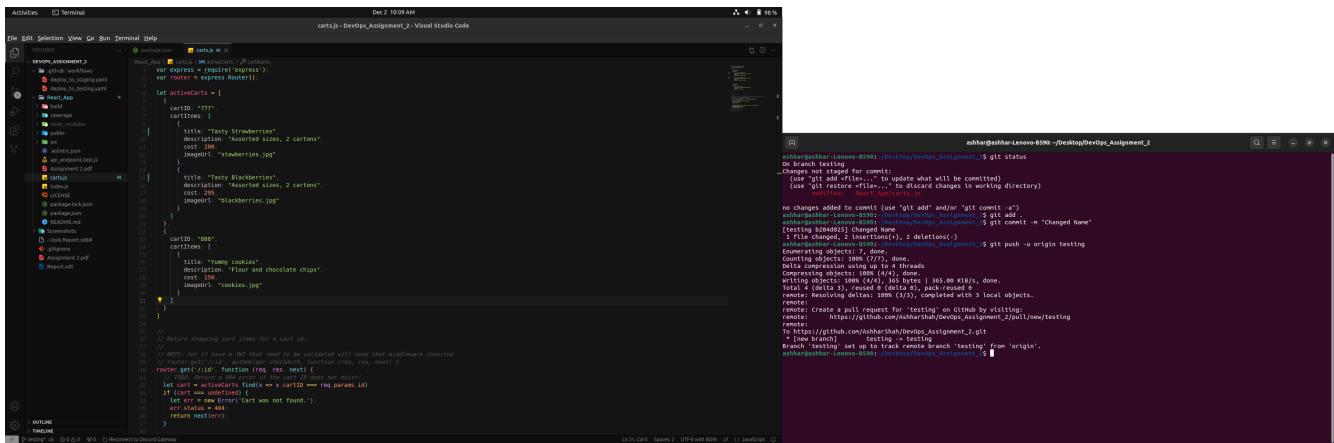
2. Starting NodeJs Application

- The next step will be to start the Node/React app so we can see it when we connect to the server via HTTP.
- Once we start the application on both the server we will see that we can view the application when we refresh the web browser.



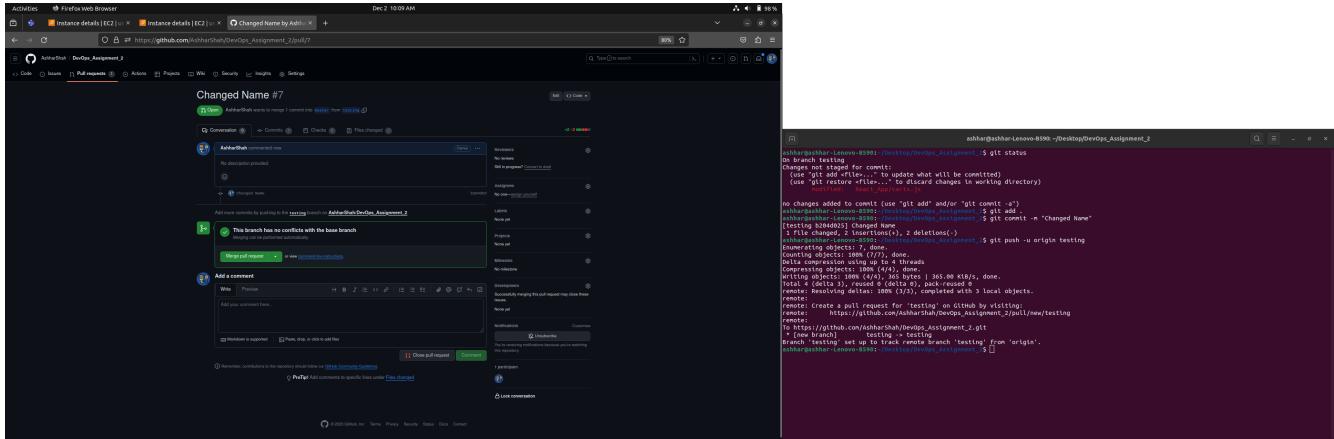
3. Modify The Code & Push To Origin

- Next we will create a new branch, in the example I have created the branch “testing” in which I will modify the underlying code by changing the name of the items from “Yummy” to “Tasty”.
- After this I will push all the changes that I have made to the “testing” branch on the “origin” server.

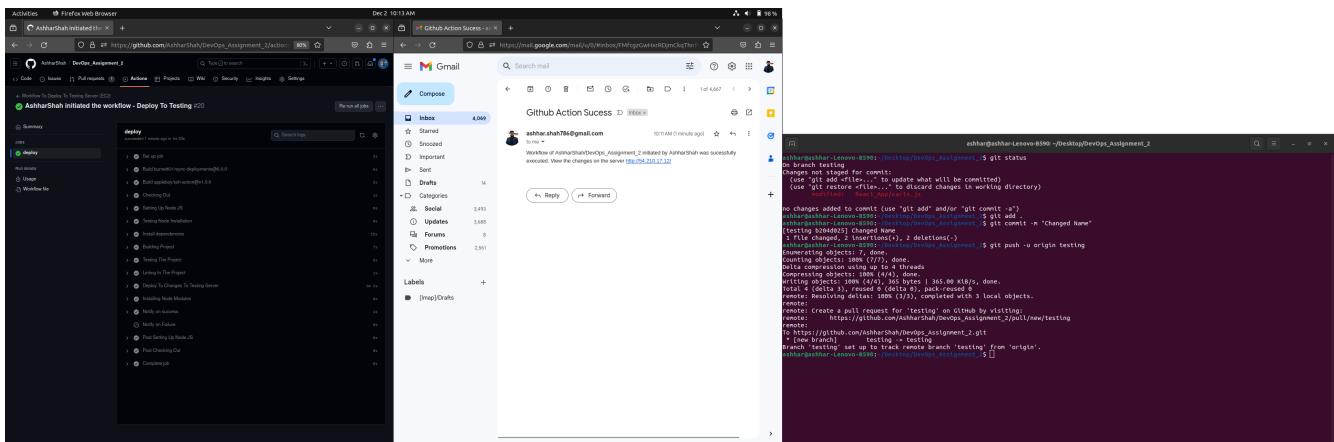


4. Create A Pull-Request

- Now that I have pushed the changes to the “origin” server. I will get an option to generate a pull-request as the “testing” branch is ahead of the “master” branch.

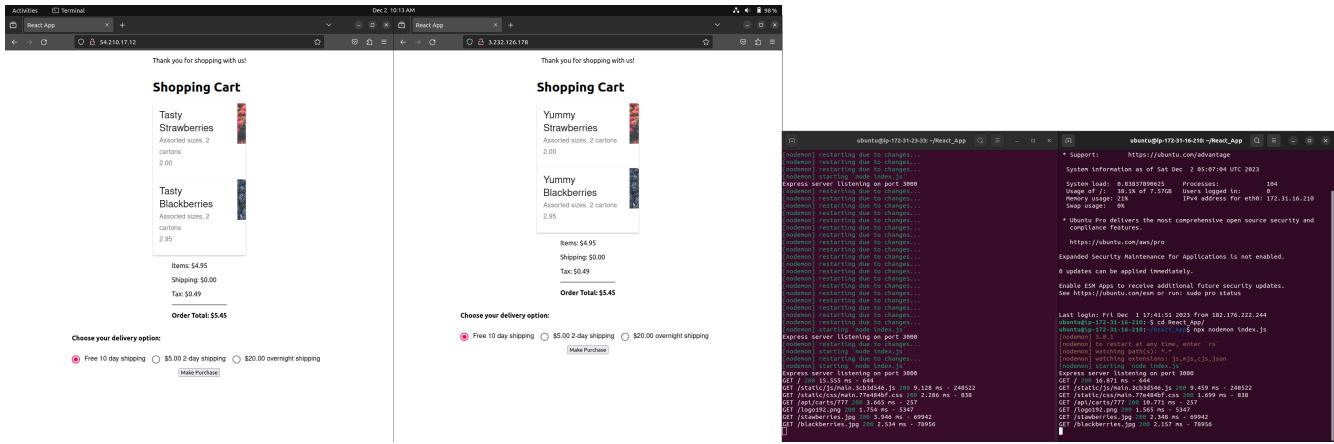


- The pull-request when generated will trigger the workflow “Deploy To Testing” which we can view in the Actions tab of our repository.
- This workflow when successfully executed, will send an email to the developer and the QA (me) which will prompt us that there were no errors. (If there was any failure it would also send an email saying that there was an error during the workflow’s execution).



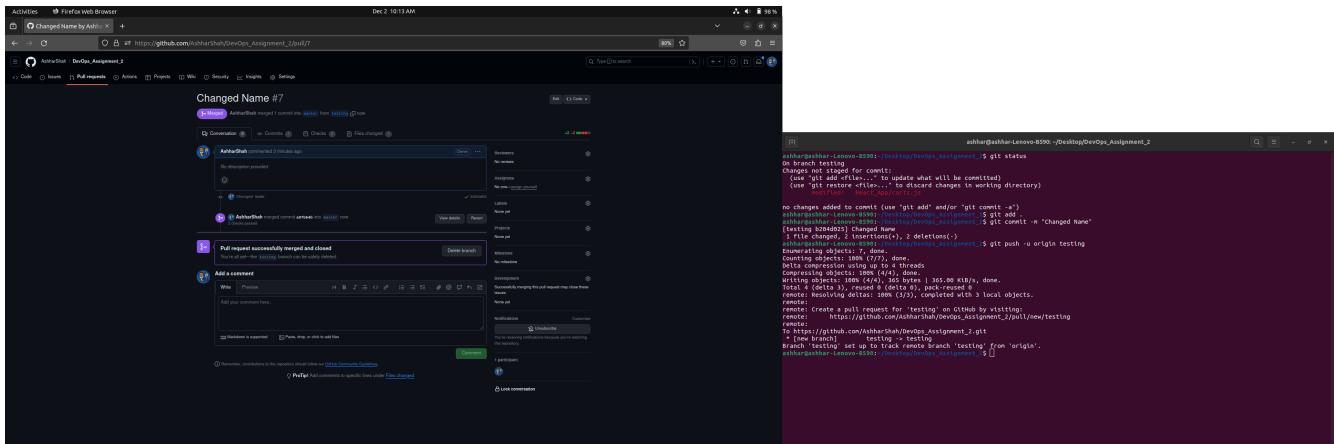
5. Verify Testing Changes

- Now the changes will be deployed to the testing server which can be viewed by the QA and the developer for any errors or bugs.
- We can see in the screenshot below that both the “testing” and the “staging” server’s are running different application versions when we view the name of the cart items.



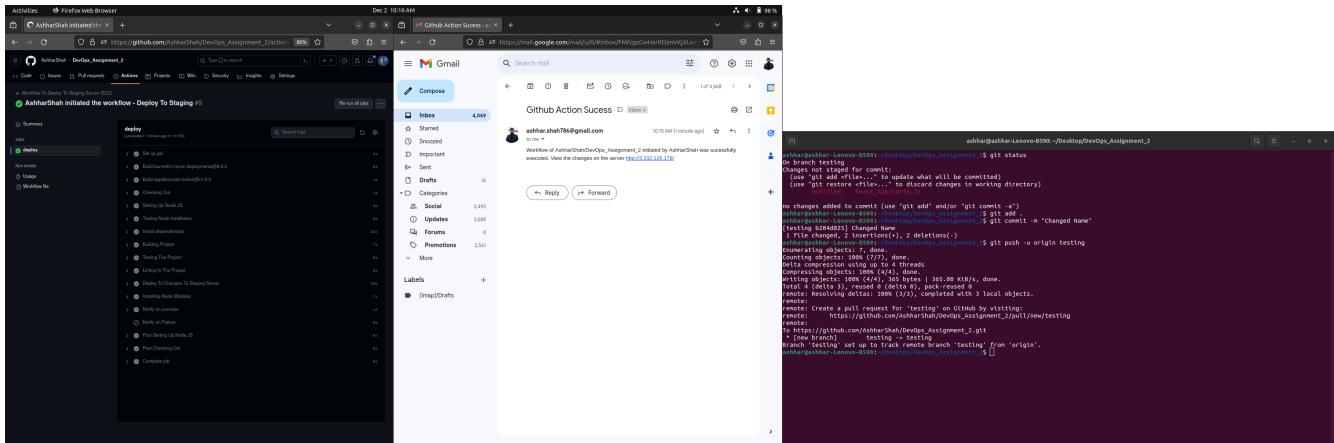
6. Merge Changes

- If the QA decides that there are no errors in the application and the application is ready to be deployed on the staging server so the client can view the changes.



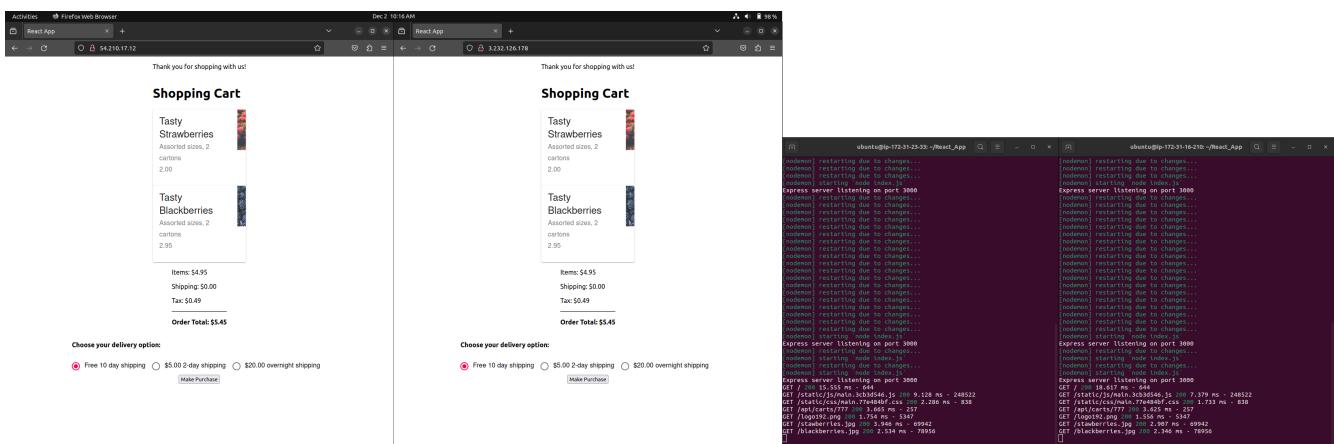
7. Deploy To Staging

- The QA will merge the “testing” branch with the “master” branch via the generated pull-request. This will then run another workflow “Deploy to Staging” which will deploy all the changes to the staging server and send an email to the QA (me).



8. Client Verification

- Now that all the application has been deployed onto the staging server, the client can view the application by simply opening his web browser and entering the Public IP of the staging server.
- At this point the application running on the “testing” and “staging” server will be identical.



CHALLENGES FACED

During the project I faced a few challenges which are listed in the points below.

1. **Setting up Nginx:** As this was my first time working with Nginx, I had difficulty in setting up the HTTP Nginx server to serve the Node/React application and make it connect to the localhost:3000.
2. **Linting In Application:** In the application there was no linting steps configured by default so I had to do research on how to use eslint to perform linting in React/Node applications to check if the syntax is up-to the required standards.
3. **Application Startup Automation:** I also had difficulty in modifying the server's setting such that when the server starts up. The React/Node application runs by default and for the changes we don't have to keep shutting down the application and restating so the changes are viewed. For this I had used nodemon which tracks the changes in the Node/React application and automatically restarts the server if there are any changes made to the application.
4. **AWS Resource Limit:** Another challenge was that I was using free-tier AWS service and in the project I had reserved 2 Elastic IP's for both of the server. I was only given 1 hour of Elastic IP service which I could use and had to utilize them efficiently. The issue with dynamic IP was that during each workflow run, I would have to change the IP's provided in the workflows.

FLOW DIAGRAM

WORKFLOW DIAGRAM

