# **Exploring AWS ElasticBeanstalk, AWS DynamoDB, AWS S3 and AWS CloudFront Services**

**Overview**

In this workshop you will explore four related Cloud Computing services with Amazon AWS: S3, CloudFront, Beanstalk and AWS DynamoDB. These services are often used together to create cloud-based web application services using AWS.

**Part 1** will show you how to clone the repository locally from the github.

**Part 2** will show you how to create the DynamoDB (NoSQL) database

**Part 3** will cover how to deploy and publish a dynamic web application (in our case Node Js CRUD API server) via AWS Elastic Beanstalk and

**Part 4** will show you how to upload the frontend static contents to the AWS S3 and

**Part 5** Serving the contents via AWS CloudFront.

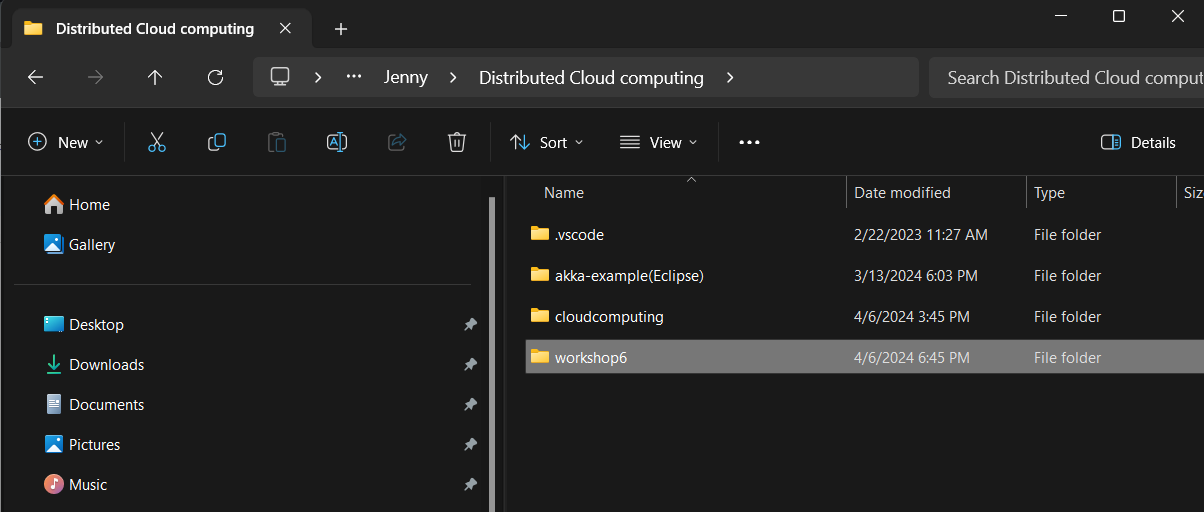
**Part 6** Updating the API base URL in the React app

You will need to use the Google Chrome web browser for this workshop.

**Cleaning up your AWS Resources**

## Part 1: Cloning Github repository

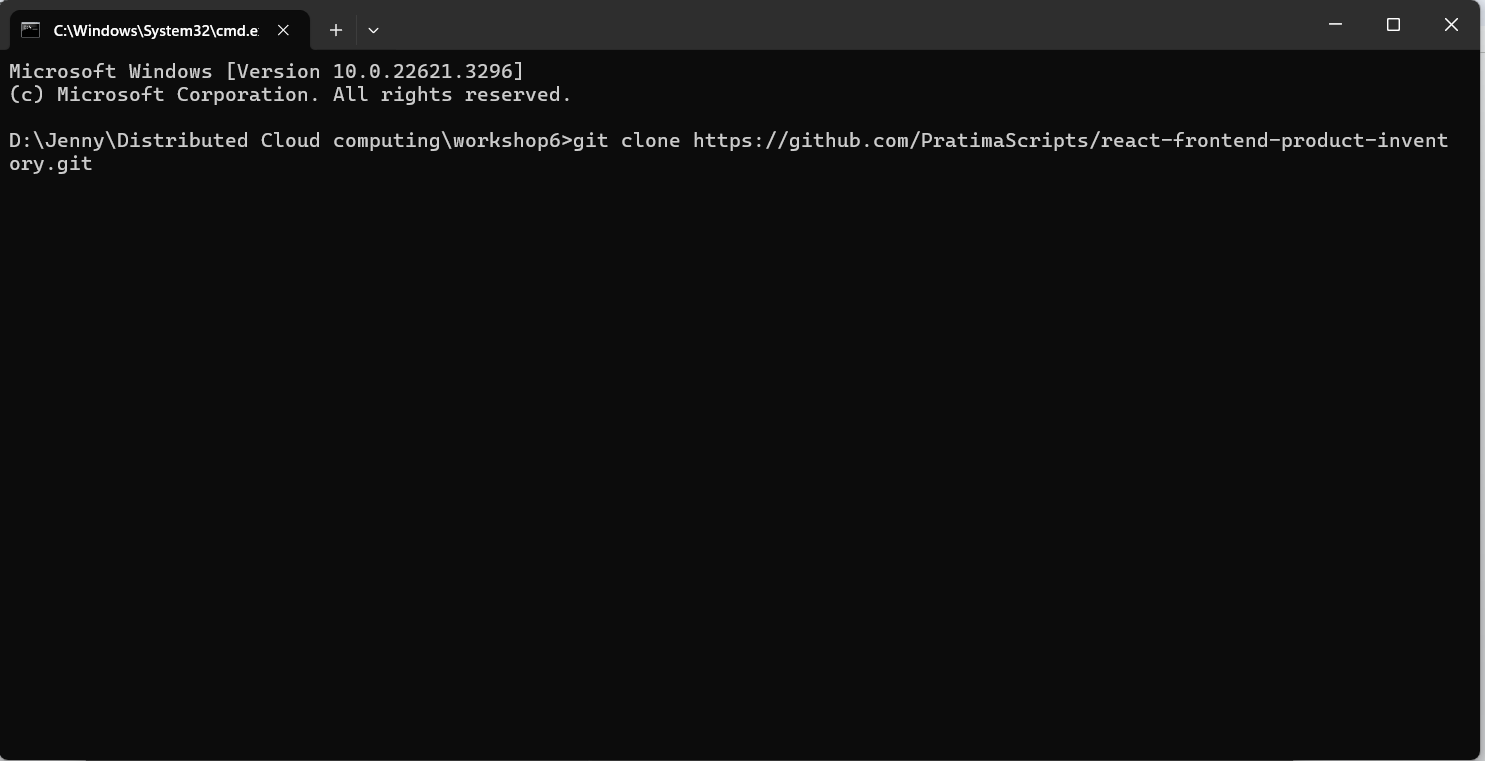
Please make a folder and name it as workshop6 as shown in the screenshot below



Open command prompt on that folder and type

**Frontend Repository:**

git clone <https://github.com/PratimaScripts/react-frontend-product-inventory.git>

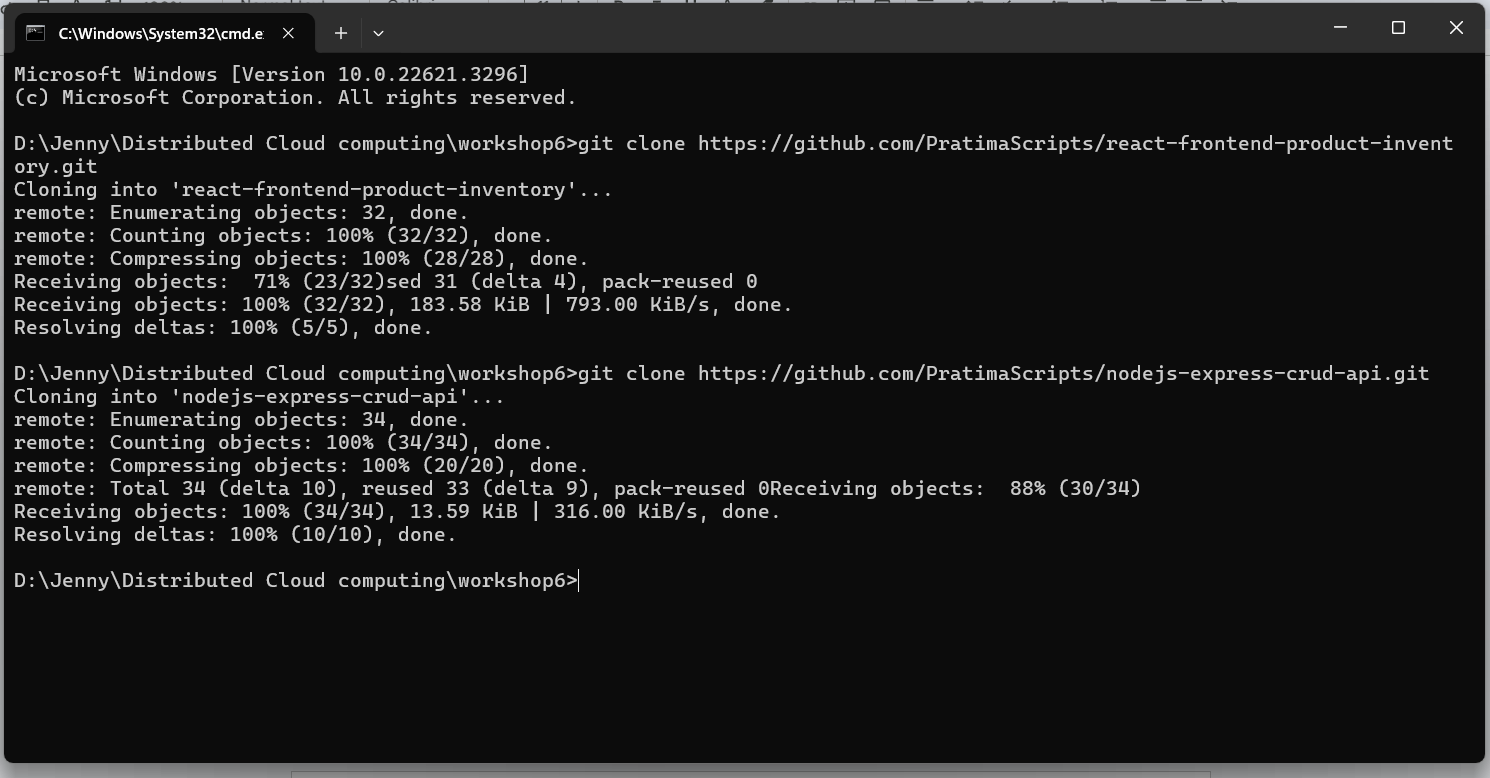


**Backend (Node.js Express CRUD API)**

git clone <https://github.com/PratimaScripts/nodejs-express-crud-api.git>

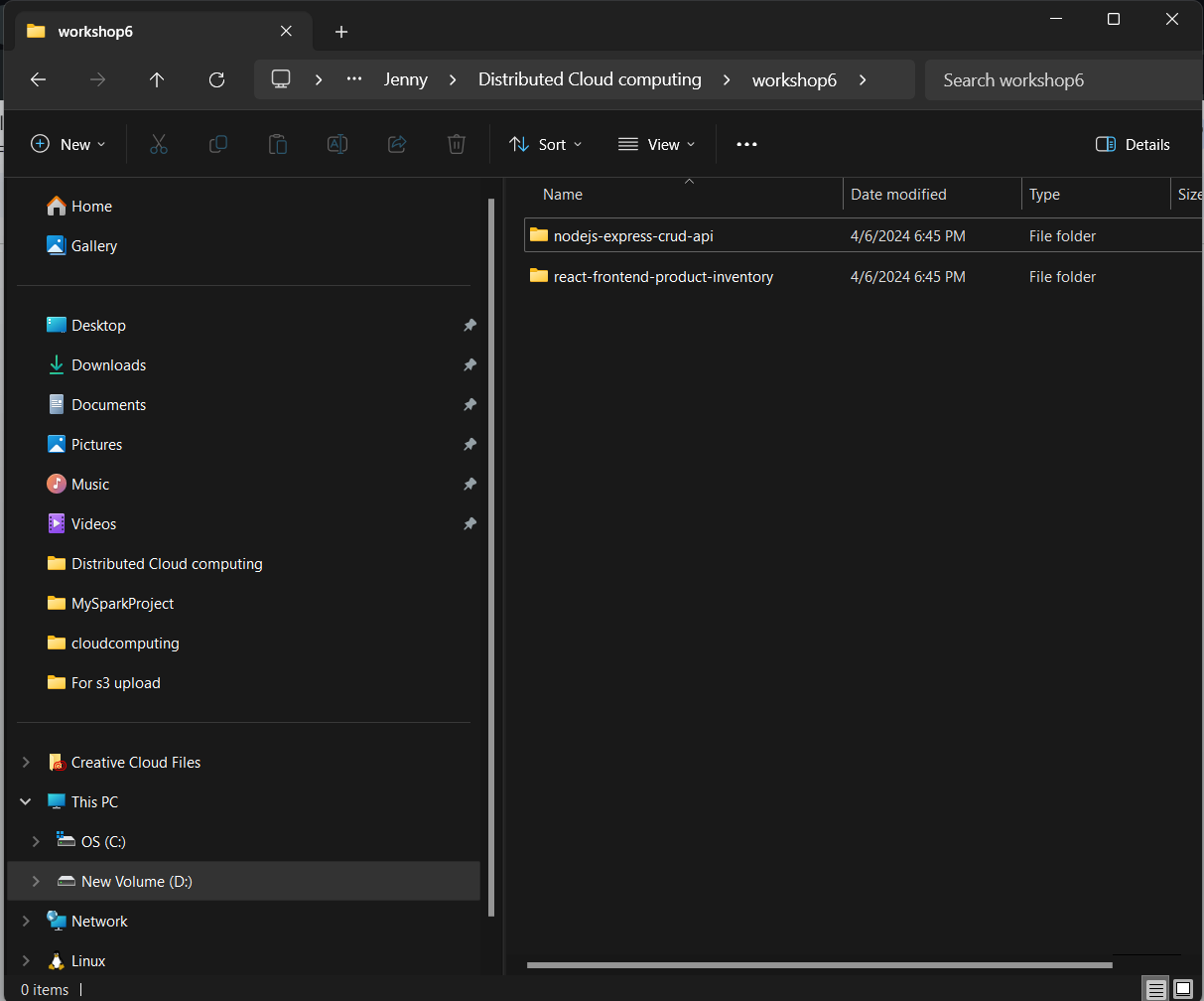
Please wait for the respective clone to complete one by one.

Please refer to the screenshot below if you have any confusion.



Note: You are cloning this repository for completing the task for the workshop, we need these repositories for deploying the client-side and server-side application on AWS Cloud.

After cloning the repository successfully, you can observe that two folders with necessary files have been uploaded to your local directory.



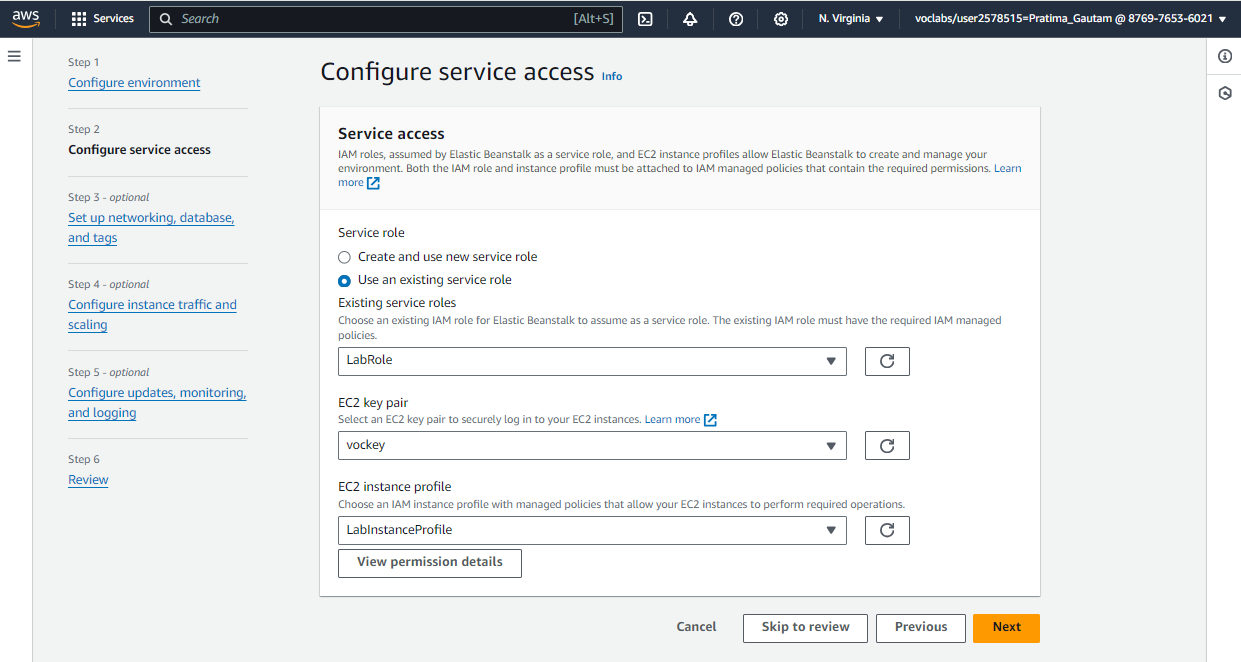
## Part 2: DynamoDB Table

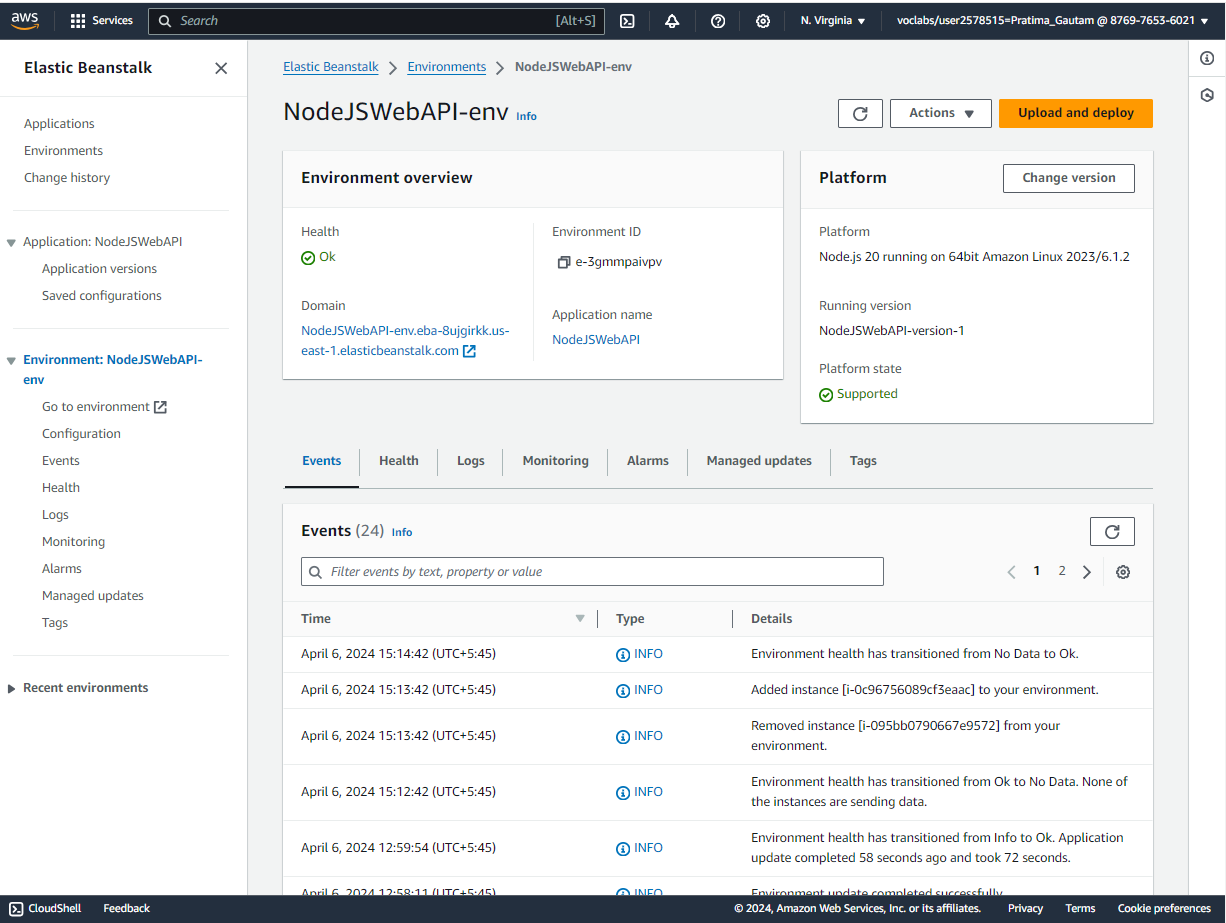
Before deploying the API, create a DynamoDB table named **‘product-inventory’** in the same AWS region (i.e. us-east-01) where you plan to deploy your application.

1. Go to the DynamoDB and click on the **Create table.**
2. Enter the Table name as **product-inventory** and Partition key as **productId.** Leave the Sort key blank.
3. Scroll down to the bottom and click on **Create Table**

## Part 3: Deploying Web API Server to AWS Elastic Beanstalk

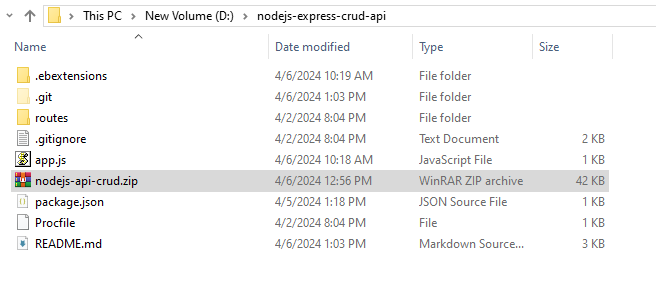
To deploy the Node.js CRUD API server to the AWS Elastic Beanstalk.

1. Go to the main AWS Console
2. Choose the Services menu, locate the Compute services, and choose Elastic Beanstalk.
3. Choose **Create Application**.
4. For Application name, enter a name for your application; for example, JeffreyWebApp
5. For Platform, select **Node.js**
6. For Application code, select Sample application and click on **Next** at the end of the page.
7. On the **Configure service access** tab, select the following options:
8. Click on **Skip to review** and scroll to the bottom and click on **Submit.**
9. Wait for the console as Elastic Beanstalk creates and runs the necessary resources to run the application. It takes a few minutes for the process to complete.

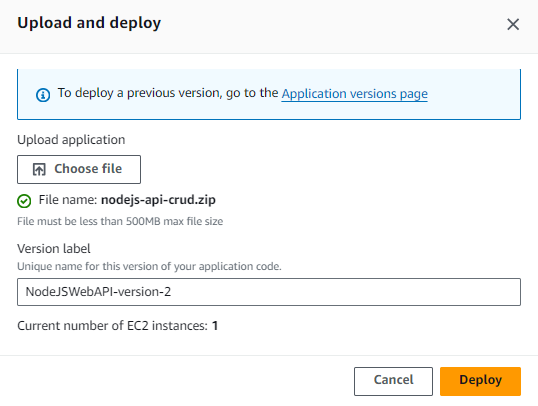
****

Elastic Beanstalk creates an Amazon Simple Storage Service (Amazon S3) storage bucket and a security group, launches an Amazon Elastic Compute Cloud (Amazon EC2) instance, and runs the code.

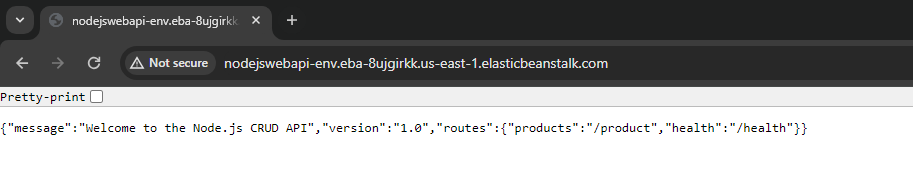
1. When complete, the screen changes to show the newly created environment. It is ready for you to upload a Node.js application.
2. Create the zip of the backend code you cloned on Part **1.**

****

1. Return to the Elastic Beanstalk console tab.
2. Choose Upload and deploy.
3. Choose Choose file, navigate to and select the zip file, and choose Open.
4. Choose Deploy.



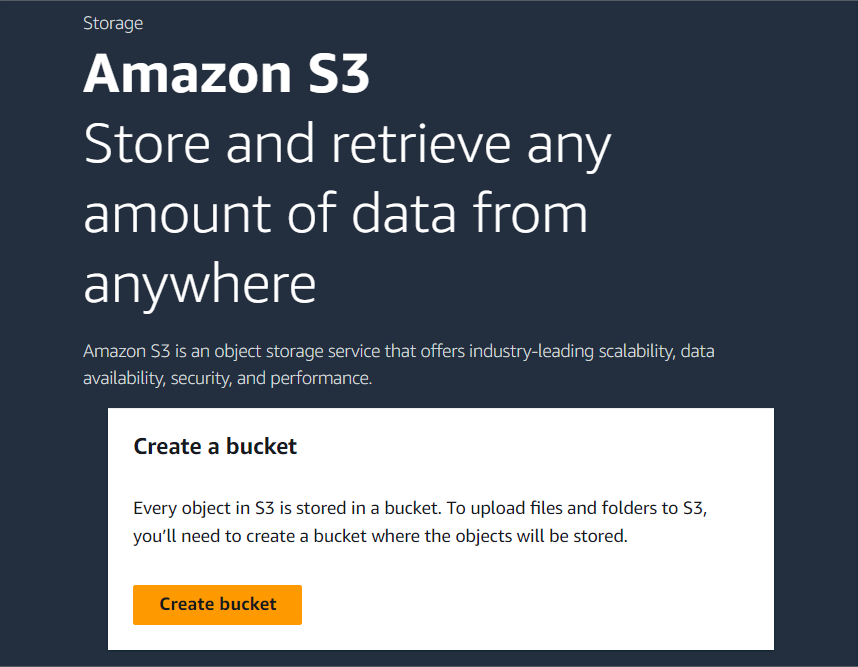
1. The application deploys to the environment using all of the cloud resources Elastic Beanstalk provisioned.



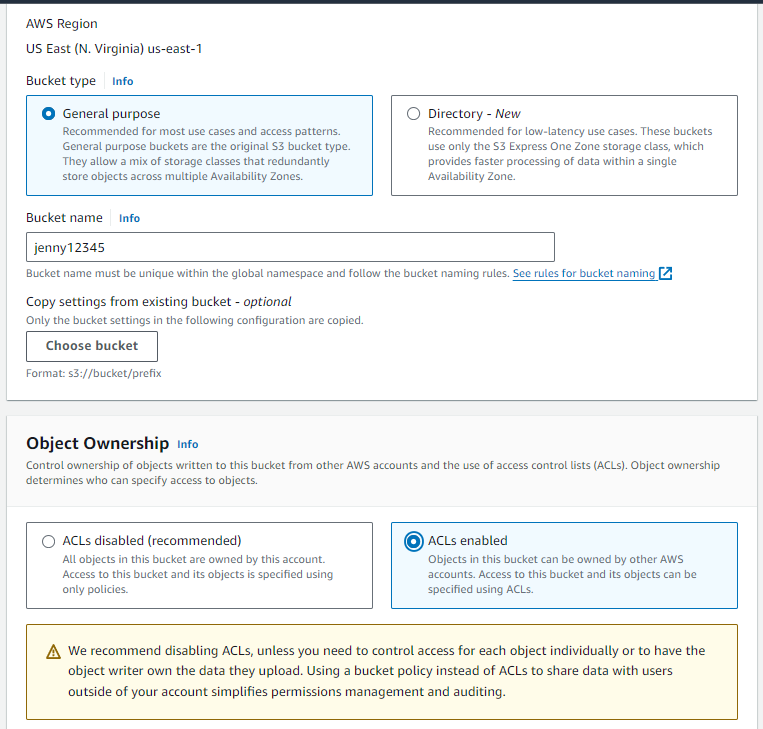
## Part 4: Uploading the Frontend to the AWS S3

To upload the frontend react static codes to the AWS S3 follow these steps:

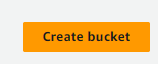
1. Install dependencies using **npm install** for the frontend code you cloned on **Part 1**
2. Build the React app using **npm run build**
3. Go to the AWS Management Console and search for AWS S3.
4. Select Create bucket



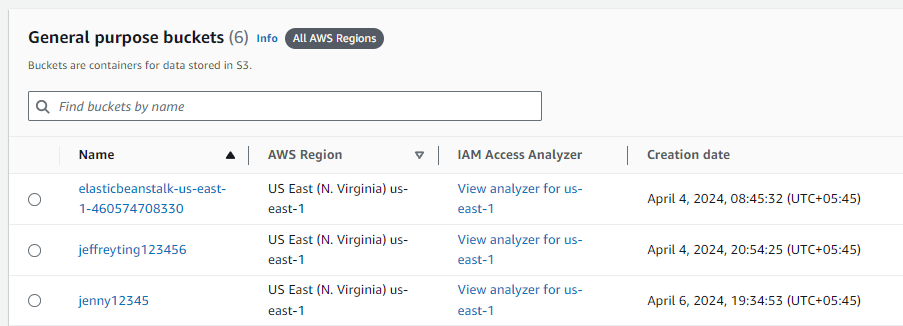
1. For bucket type leave it as it is **“General purpose”**
2. For AWS Region, leave it as the selection **(US East)**.
3. For bucketname, enter a unique name system (DNS)-complaint name. For example, **“jeffreything12345”** Note: The name must only contain lowercase characters.
4. For Object Ownership, select **ACLs enabled**

****

1. **Block all public access,** keep it as it is.
2. Scroll to the bottom of the page and select **Create bucket**

****

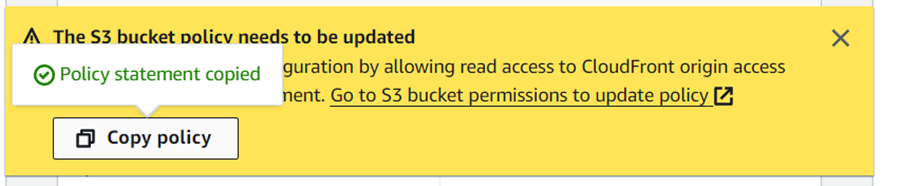
Your new bucket appears in the **Buckets** list



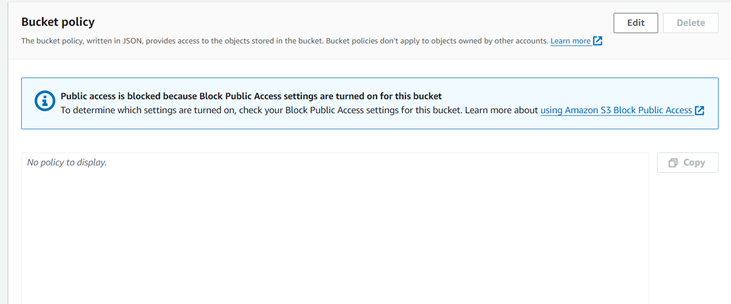
1. Upload the **build** files to an **S3 bucket**.

## Part 5: Serving the contents via AWS CloudFront

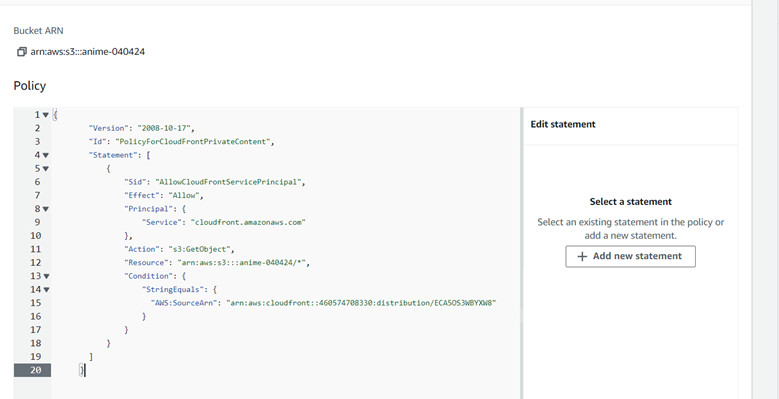
* Configure a CloudFront distribution to serve the files from the S3 bucket.
* Copy the S3 Bucket Policy from CloudFront, go to your bucket permissions and update the policy.



* It will then redirect you to a page where you can change the policy for S3 bucket please navigate to Bucket policy and click on the edit button



* In the policy session please paste the policy which you have copied on the last steps and click on save changes button just below the page



## Part 6: Update the API base URL in the React app

* Update the API base URL in the React app to point to your Node.js Express API hosted on Elastic Beanstalk.
* Create a .env file in the root directory of the frontend project.
* Add the following line to the .env file and replace **<elasticbeanstalk-domain-url>** with your Elastic Beanstalk domain URL:

**REACT\_APP\_API\_BASE\_URL=http://<elasticbeanstalk-domain-url>**

**DON’T ADD TRAILING SLASH (/) AT THE END OF THE URL**

* Access the deployed application using the CloudFront distribution URL.

# 

# **Cleaning up your AWS Resources**

## Deleting Lambda Functions

1. Navigate to the Lambda console: [AWS Management Console > Lambda](https://console.aws.amazon.com/lambda/).
2. Select the Lambda function you want to delete from the list.
3. Once you've selected the function, click on the "Actions" dropdown menu.
4. From the dropdown menu, choose "Delete function".
5. You'll be prompted to confirm the deletion. Click on "Delete" to confirm.

## Deleting DynamoDB Table

1. Navigate to the DynamoDB console: [AWS Management Console > DynamoDB](https://console.aws.amazon.com/dynamodb/).
2. Select the table you want to delete.
3. Click on the "Delete table" button.
4. Keep the other setting as it is:
   1. **(Ticked)** Delete all CloudWatch alarms for product-inventory.
   2. **(Unticked)** onCreate an on-demand backup of product-inventory before deletion.
5. Confirm the deletion by typing confirm and clicking "Delete".

Note: Deleting a DynamoDB table will permanently delete all data stored in the table.

## Deleting Elastic Beanstalk Environment

1. Navigate to the Elastic Beanstalk console: [AWS Management Console > Elastic Beanstalk](https://console.aws.amazon.com/elasticbeanstalk/).
2. Select the environment you want to delete.
3. Click on the "Actions" dropdown menu.
4. Choose "Terminate environment".
5. Confirm the termination by entering the name of your environment in the text field and terminate.
6. Go to "Applications" on the right side and select the application.
7. Click on the "Actions" dropdown menu.
8. Choose "Delete Application", enter your application name and confirm delete by clicking on "Delete" button.

Note: Deleting an Elastic Beanstalk environment does not automatically delete associated resources such as S3 buckets. You'll need to delete these resources separately.

## Deleting S3 Buckets created by Elastic Beanstalk

1. Navigate to the S3 console: [AWS Management Console > S3](https://s3.console.aws.amazon.com/s3/).
2. Find the bucket associated with your Elastic Beanstalk environment. It usually has a name like elasticbeanstalk-<region>-<account-id>
3. Click on the bucket name.
4. Select the "Permissions" tab.
5. Under the "Bucket Policy" section, you'll see the policy attached to the bucket.
6. Click on the "Delete" button next to the policy to remove it. Confirm the deletion by clicking "Delete".
7. Click on the "Empty bucket" button to delete all objects within the bucket. To confirm deletion, type *permanently delete* in the text input field and click on Empty.
8. Once the bucket is empty, click on the "Delete" button, type your bucket name and Delete it.

## Deleting CloudFront Distribution

1. Navigate to the CloudFront console: [AWS Management Console > CloudFront](https://console.aws.amazon.com/cloudfront/).
2. Select the distribution you want to delete and click on the "Disable" button to disable the distribution.
3. Once disabled, click on the "Actions" dropdown menu.
4. Choose "Delete".
5. Confirm the deletion by clicking "Yes, Delete".

Note: Deleting a CloudFront distribution may take some time to complete. Once deleted, the distribution and associated resources will be permanently removed.