

8byte Intern Assignment – DevOps Deployment Project

Public Application Access

The deployed Node.js application is accessible at:
<http://98.92.148.150:3000/>

Git repo: [vommidapuchinni/8bytes](https://github.com/vommidapuchinni/8bytes)

Prerequisites

Before running this project locally or provisioning infrastructure, ensure you have:

- Node.js & npm
- Docker & Docker Desktop
- Terraform
- AWS Account with IAM user credentials (Access Key ID & Secret Access Key)

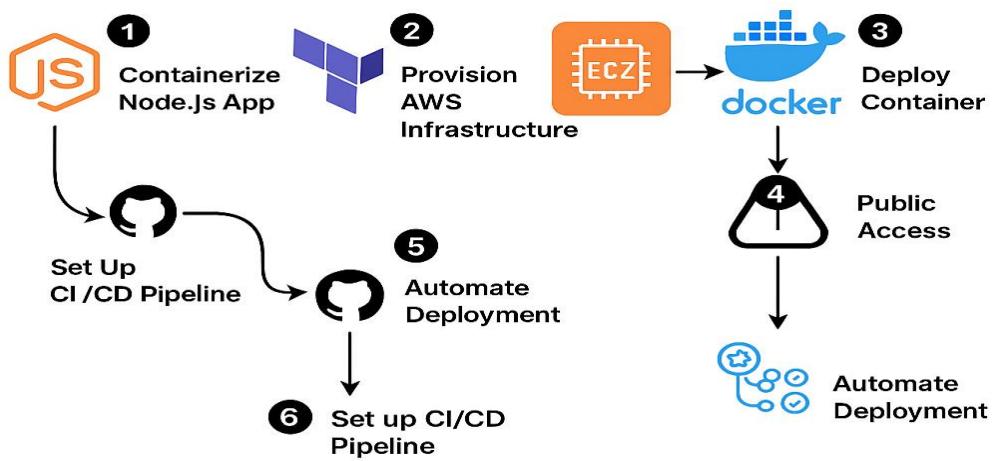
Objectives

This project demonstrates hands-on experience in:

1. Infrastructure as Code using Terraform
2. AWS Networking (VPC architecture design)
3. Containerization using Docker
4. Continuous Integration using GitHub Actions
5. Application deployment on EC2 with public access
6. Infrastructure lifecycle and cost management

Architecture Overview

Deploy a Containerized Node.js Application on AWS using Terraform and GitHub Actions



Component	Purpose
VPC	Custom isolated cloud network
Public Subnet	Hosts the EC2 instance
Internet Gateway	Enables internet connectivity
Route Table	Routes outbound internet traffic
Security Group	Allows SSH (22) & App (3000)
EC2 Instance	Hosts the Dockerized application
Docker	Runs the Node.js application
GitHub Actions	Automates Docker build (CI/CD)

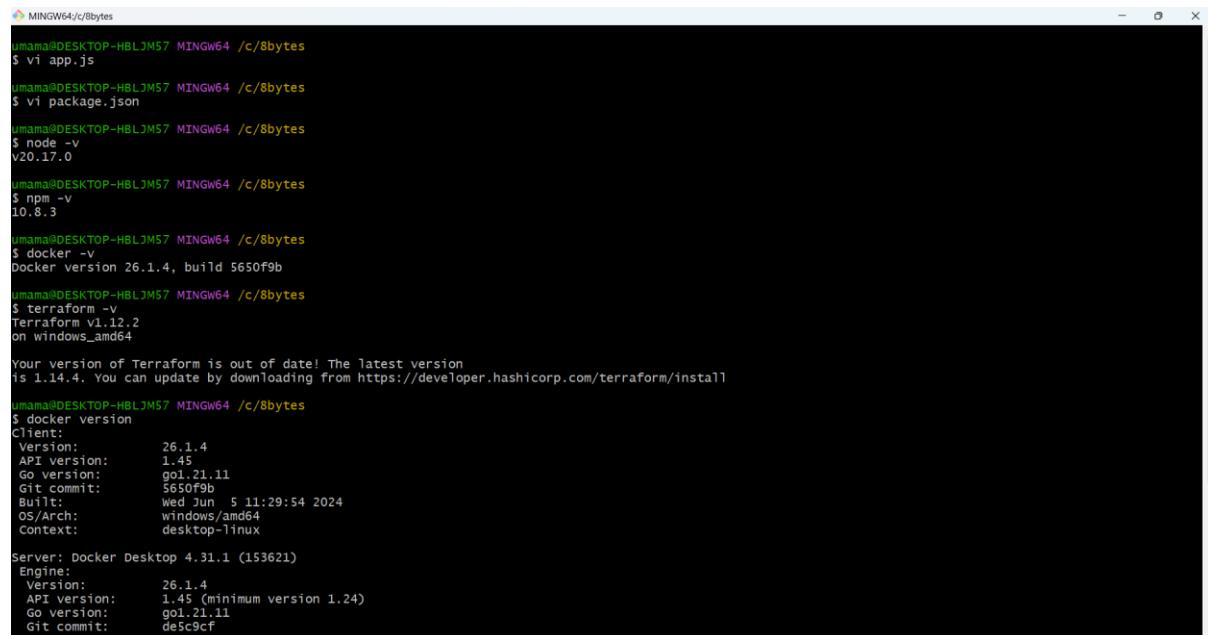
Run Application Locally

Before deploying to AWS, you can test and run the Node.js application on your local machine to ensure it works as expected.

Install dependencies and run the application:

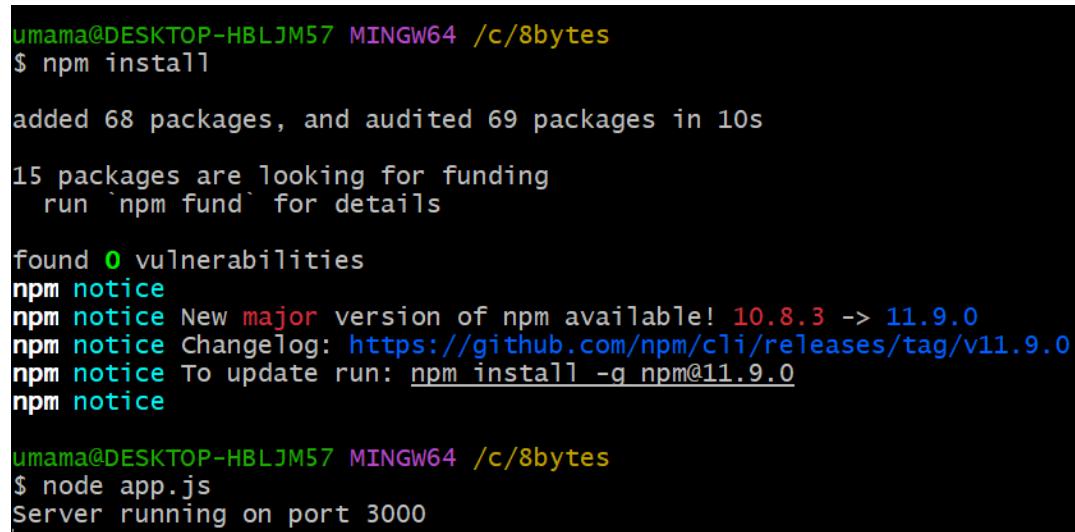
```
npm install && node app.js
```

Installations



```
MINGW64/c/8bytes
$ vi app.js
MINGW64 /c/8bytes
$ vi package.json
MINGW64 /c/8bytes
$ node -v
v20.17.0
MINGW64 /c/8bytes
$ npm -v
10.8.3
MINGW64 /c/8bytes
$ docker -v
docker version 26.1.4, build 5650f9b
MINGW64 /c/8bytes
$ terraform -v
Terraform v1.12.2
on windows_amd64
your version of Terraform is out of date! The latest version
is 1.14.4. You can update by downloading from https://developer.hashicorp.com/terraform/install
MINGW64 /c/8bytes
$ docker version
Client:
  Version:          26.1.4
  API version:      1.45
  Go version:       go1.21.11
  Git commit:       5650f9b
  Built:            Wed Jun  5 11:29:54 2024
  OS/Arch:          windows/amd64
  Context:          desktop-linux
Server: Docker Desktop 4.31.1 (153621)
  Engine:
    Version:          26.1.4
    API version:      1.45 (minimum version 1.24)
    Go version:       go1.21.11
    Git commit:       de5c9cf
```

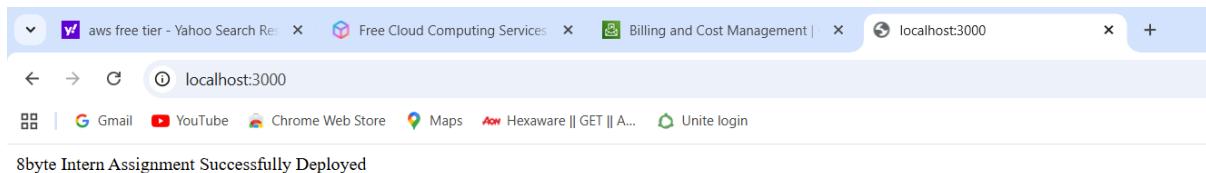
```
npm install && node app.js
```



```
MINGW64 /c/8bytes
$ npm install
added 68 packages, and audited 69 packages in 10s
15 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 10.8.3 -> 11.9.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.9.0
npm notice To update run: npm install -g npm@11.9.0
npm notice

MINGW64 /c/8bytes
$ node app.js
Server running on port 3000
```

Open in browser: <http://localhost:3000>



Expected Output: 8byte Intern Assignment Successfully Deployed

Dockerizing the Application

We will now containerize the Node.js app using Docker to prepare it for deployment on EC2.

```
FROM node:18
```

```
WORKDIR /app
```

```
COPY . .
```

```
RUN npm install
```

```
EXPOSE 3000
```

```
CMD ["node", "app.js"]
```

Build & Run

```
docker build -t 8byte-intern-app . # build docker file
```

```
docker run -p 3000:3000 8byte-intern-app
```

Docker build locally

```

MINGW64/c/8bytes
uama@DESKTOP-HBLJM57 MINGW64 /c/8bytes
$ docker build -t 8byte-intern-app .
Building 339.4s (11/11) FINISHED
= [internal] load build definition from Dockerfile
=> transfering dockerfile: 144B
= [internal] load metadata for docker.io/library/node:18
=> [auth] library/node:pull token for registry-1.docker.io
= [internal] load .dockerignore
=> transferring context: 2B
[1/5] FROM docker.io/library/node:18@sha256:c6ae79e4898325dbd7193d391e6ec1d224d96c693aa8a4d943498556716d3783
=> resolve docker.io/library/node@sha256:c6ae79e4898325dbd7193d391e6ec1d224d96c693aa8a4d943498556716d3783
=> sha256:b50082bc3c670d0396b2d90e4b0e5b1b0265ba0f0ee16bf40ff9a50f704ee563 6.39kB / 6.39kB
=> sha256:7f2393631e97889faadca878d4abc6f7fd399adbb13b7ed3c1f1badee69 2.49kB / 2.49kB
=> sha256:c6ae79e3848832dc667193d391e6ec1d224d96c693aa8a4d943498556716d3783 6.41kB / 6.41kB
=> sha256:79b947ad444365269b5cc095ed64808a0d01a1c062108ad0054a0 94964 48.49MB / 48.49MB
=> sha256:79b947ad444365269b5cc095ed64808a0d01a1c062108ad0054a0 94964 48.49MB / 48.49MB
=> sha256:c677977ed1608b72799e60fbfe45700268ea915799119e050b4-d8d24 24.02MB / 24.02MB
=> sha256:c677977ed1608b72799e60fbfe45700268ea915799119e050b4-d8d24 24.02MB / 24.02MB
=> sha256:c6f099911d692f62851fc4911e97294788a115f5cd20014180e4d5684a4b 211.36MB / 211.36MB
=> sha256:c6f099911d692f62851fc4911e97294788a115f5cd20014180e4d5684a4b 211.36MB / 211.36MB
=> sha256:c6b303f1696651af10ac00521f60355b1fd4645a1c2b010385e7e83c67 45.68MB / 45.68MB
=> sha256:c6b303f1696651af10ac00521f60355b1fd4645a1c2b010385e7e83c67 45.68MB / 45.68MB
=> sha256:48177477f740d3437d6a19584d16772dd0d77f577c501df41a3754d 446B / 446B
=> extracting sha256:3e9b9a511d19f12623a48a59d1d10a1c7b82108adcf0605a200294984
=> extracting sha256:379f27ed01b1b2608b72799e688b16f458268e9a3a937b9210e050b4-d8d4
=> extracting sha256:79b947ad444365269b5cc81a95ed64808a0d01a1c062108ad0054a0
=> extracting sha256:c677977ed1608b72799e60fbfe45700268ea915799119e050b4-d8d4
=> extracting sha256:c6f099911d692f62851fc4911e97294788a115f5cd20014180e4d3684d343b
=> extracting sha256:cda7f44f2bdccc4b7154474074b3f73705de00dd635533be5ac7808e8b7125
=> extracting sha256:c6b303f1696651af10ac00521f60355b1fd4645a1c2b010385e7e83c67
=> extracting sha256:369b09d0c98bb0d71fd4637e1d3491d00e7bf93a7377688e76d82309b3c5a145
=> extracting sha256:461077372ff7e40d34a37d6a1958c4d16772dd0d77f572ec501fd41a3754d
[internal] load build context
=> transferring context: 2.33MB
[2/5] WORKDIR /app
[3/5] COPY package.json .
[4/5] RUN npm install
[5/5] COPY
=> exporting to image
=> exporting layers
=> writing image sha256:23d33af155083ec40bcfa02a1006f5f685ac6738953dfba8f9e84a6d0f23fdc
=> naming to docker.io/8byte-intern-app

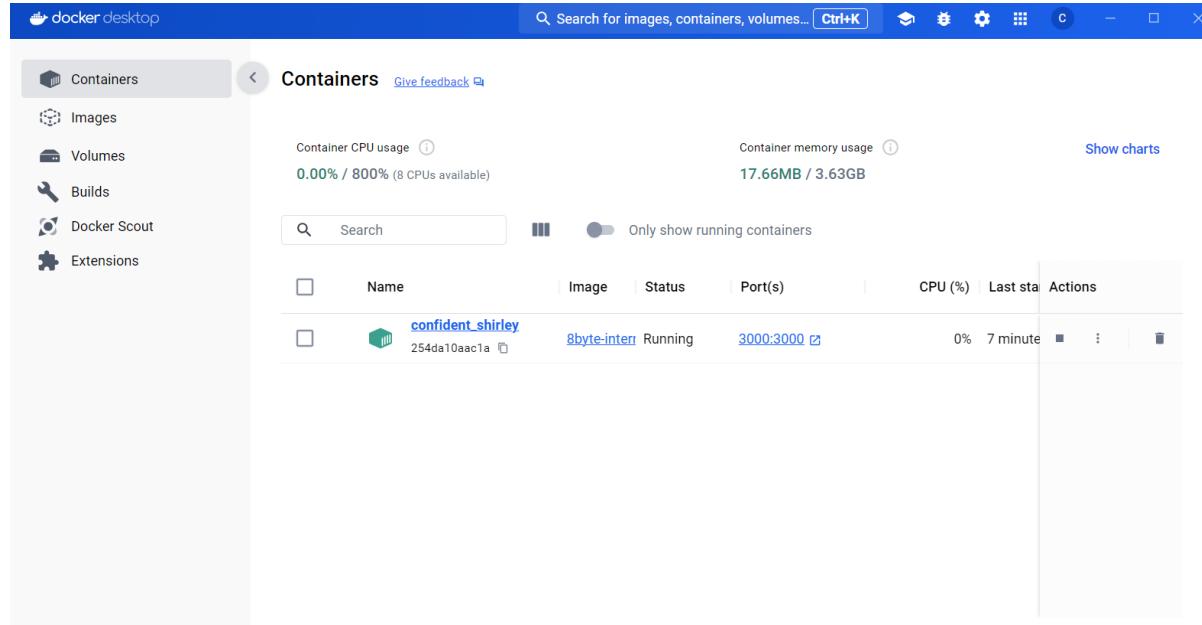
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/b812heg46h1pxw2ul0d0mfax

What's next:
```

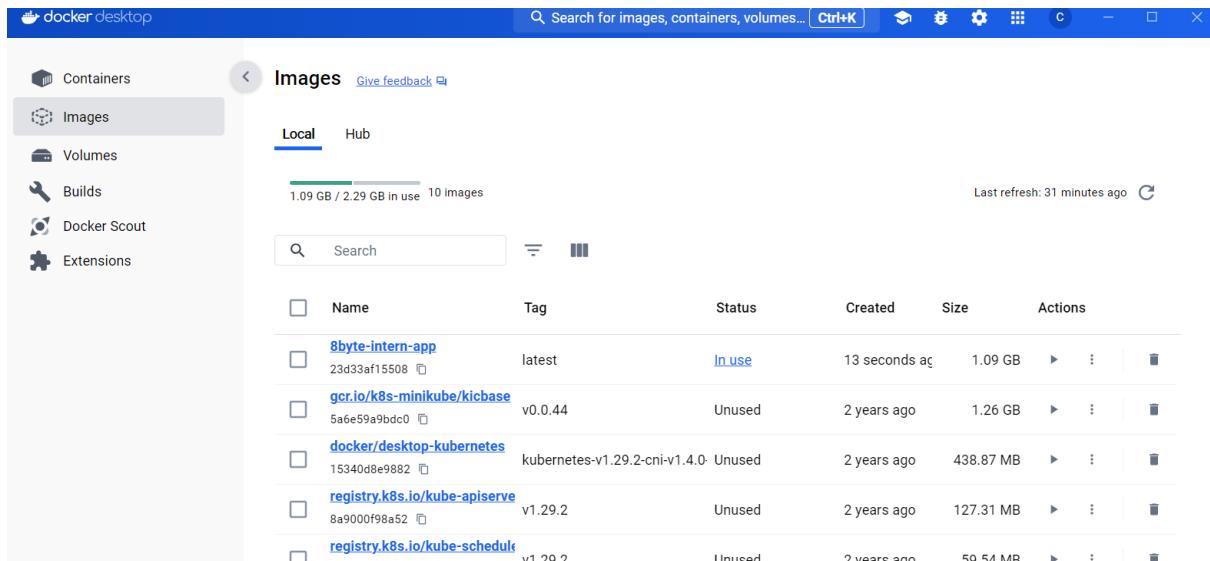
Docker run locally

```
umama@DESKTOP-HBLJM57 MINGW64 /c/8bytes
$ docker run -p 3000:3000 8byte-intern-app
Server running on port 3000
```

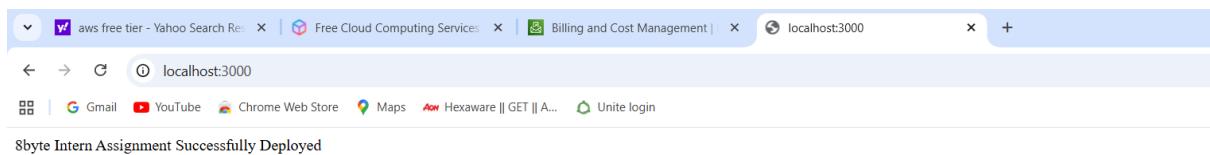
Docker container



Docker image



Access APP



Visit: <http://localhost:3000>

Infrastructure Provisioning with Terraform

Terraform allows us to define and provision AWS infrastructure in code, making deployments repeatable and manageable.

Directory Structure `terraform/ provider.tf, variables.tf, terraform.tfvars, main.tf, outputs.tf`

Purpose of Each File

File	Purpose
provider.tf	Configures AWS provider (region: us-east-1)
variables.tf	Defines reusable input variables
Internet Gateway	Stores environment-specific values
main.tf Contains	infrastructure resources
outputs.tf	Displays useful outputs (EC2 Public IP)

To create resources on AWS locally using Terraform, you must provide AWS credentials. These credentials are obtained by creating an IAM user with programmatic access and generating an Access Key ID and Secret Access Key.

Terraform uses these keys to authenticate and provision AWS resources.

IAM User

The screenshot shows the AWS IAM User details page for a user named 'chinni'. The page is divided into several sections:

- Identity and Access Management (IAM)**: A sidebar with navigation links like Dashboard, Access Management (Users, Roles, Policies), Identity providers, Account settings, Root access management, Temporary delegation requests, and Access reports.
- Summary**: Displays the ARN (arn:aws:iam::615948015803:user/chinni), creation date (February 08, 2026, 15:16 UTC+05:30), and console access status (Enabled without MFA). It also shows the last console sign-in (Never).
- Permissions**: A tabbed section showing the user's permissions. It lists two policies: 'AdministratorAccess' and 'IAMUserChangePassword'. Both are AWS managed - job function type and attached directly.
- Groups**, **Tags**, **Security credentials**, and **Last Accessed**: Other tabs in the 'Permissions' section.
- Actions**: Buttons for Delete, Remove, and Add permissions.
- Footer**: Includes links to CloudShell, Feedback, and Console Mobile App, along with copyright information: © 2026, Amazon Web Services, Inc. or its affiliates.

Secret keys creation for credentials

The screenshot shows the AWS IAM 'Create access key' interface. The user is on Step 1, titled 'Access key best practices & alternatives'. A note says to avoid long-term credentials like access keys to improve security. Below are five options for 'Use case':

- Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.
- Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- Application running outside AWS
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

Access and secret keys

The screenshot shows the AWS IAM 'Create access key' interface. The user is on Step 2, titled 'Retrieve access keys'. A note states that this is the only time the secret access key can be viewed or downloaded. It cannot be recovered later. A 'Secret access key' field contains 'AKIAY62K43C5XPUAV346' with a 'Show' link. Below is a section on 'Access key best practices':

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

At the bottom are 'Download .csv file' and 'Done' buttons.

Terraform Commands

terraform init

```
umama@DESKTOP-HBLJM57 MINGW64 /c/8bytes/terraform
$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.31.0...
- Installed hashicorp/aws v6.31.0 (signed by Hashicorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

terraform plan

```
MINGW64/c/8bytes/terraform
umama@DESKTOP-HBLJM57 MINGW64 /c/8bytes/terraform
$ terraform plan
data.aws_ami.ubuntu: Reading...
data.aws_ami.ubuntu: Read complete after 2s [id=ami-0030e4319cbf4dbf2]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
+ create

Terraform will perform the following actions:

# aws_instance.app will be created
+ resource "aws_instance" "app" {
  + ami
  + arn
  + associate_public_ip_address
  + availability_zone
  + disable_api_stop
  + disable_api_termination
  + ebs_optimized
  + enable_primary_ipv6
  + force_destroy
  + get_password_data
  + host_id
  + host_resource_group_arn
  + iam_instance_profile
  + id
  + instance_initiated_shutdown_behavior
  + instance_lifecycle
  + instance_state
  + instance_type
  + ipv6_address_count
  + ipv6_addresses
  + key_name
  + monitoring
  + outpost_arn
  + password_data
  + placement_group
  + root_device_name
  + security_group_ids
  + subnet_id
  + tags
  + tenancy
  + type
  = "ami-0030e4319cbf4dbf2"
  = (Known after apply)
  = true
  = (Known after apply)
  = false
  = false
  = (Known after apply)
  = "t2.micro"
  = (Known after apply)
  = (Known after apply)
  = "8byte-key"
  = (Known after apply)
  = (Known after apply)
  = (Known after apply)
  = (Known after apply)
  = (Known after apply)
```

terraform apply

```
MinGW64/c/8bytes/terraform
$ terraform apply
data.aws_ami.ubuntu: Reading...
aws_vpc.main: Refreshing state... [id=vpc-058c06fbf476e3fef]
data.aws_ami.ubuntu: Read complete after 2s [id=ami-0030e4319cbf4dbf2]
aws_internet_gateway.gw: Refreshing state... [id=igw-06c0f1267650d5a3d]
aws_subnet.public: Refreshing state... [id=subnet-046e488860d33780]
aws_route_table.rt: Refreshing state... [id=rtb-053c8a5519a471e67]
aws_route_table_association.a: Refreshing state... [id=rtbassoc-0d0ec0aed0d725e91]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
+ create

Terraform will perform the following actions:

# aws_instance.app will be created
+ resource "aws_instance" "app" {
    + ami
    + arn
    + associate_public_ip_address
    + availability_zone
    + display_name
    + display_status
    + disable_api_termination
    + ebs_optimized
    + enable_primary_ipv6
    + force_destroy
    + get_password_data
    + host_id
    + host_resource_group_arn
    + iam_instance_profile
    + id
    + instance_initiated_shutdown_behavior
    + instance_lifecycle
    + instance_state
    + instance_type
    + ipv6_address_count
    + ipv6_addresses
        = "ami-0030e4319cbf4dbf2"
        = (Known after apply)
        = true
        = (Known after apply)
        = false
        = false
        = (Known after apply)
        = t3.micro
        = (Known after apply)
        = (Known after apply)
```

outputs

Outputs:

```
public_ip = "98.92.167.154"
ssh_command = "ssh -i 8byte-key.pem ubuntu@98.92.167.154"

umama@DESKTOP-HBLJM57 MINGW64 /c/8bytes/terraform
$
```

Resources Created: VPC, Public Subnet, Internet Gateway, Route Table & Association, Security Group (ports 22 and 3000), EC2 Instance (Ubuntu with Docker installed via user_data)

Resources:

EC2 running

The screenshot shows the AWS EC2 Instances details page for instance `i-025349514124a7d20`. The instance is listed as `Running` with a `t3.micro` instance type. It has a `Private IP DNS name (IPv4 only)` of `ip-10-0-1-105.ec2.internal` and a `VPC ID` of `vpc-04fde247c35eb3dbc (8byte-vpc)`. The `Auto-assigned IP address` is `98.92.148.150 [Public IP]`. The `Subnet ID` is `subnet-09fd66c0fb9c8cef8 (8byte-public-subnet)` and the `Instance ARN` is `arn:aws:ec2:us-east-1:615948015803:instance/i-025349514124a7d20`.

VPC

The screenshot shows the AWS VPC dashboard for VPC `vpc-058c06fb4f476e3fef`. The VPC is in the `Available` state. `Block Public Access` is set to `Off`. The `DNS hostnames` are `Disabled`. The `Main route table` is `rtb-01a38191da64ee7c4`. There is one subnet and two route tables associated with this VPC.

Subnet

The screenshot shows the AWS VPC Subnet details page. The subnet ID is subnet-046e488860d133780. Key details include:

- Subnet ARN:** arn:aws:ec2:us-east-1:615948015803:subnet/subnet-046e488860d133780
- IPv4 CIDR:** 10.0.1.0/24
- Available IPv4 addresses:** 250
- Network border group:** us-east-1
- Default subnet:** No
- Customer-owned IPv4 address:** No
- IPv6 CIDR:** -
- VPC:** vpc-058c06fbf476e3fef | 8byte-vpc
- Auto-assign public IPv4 address:** Yes
- Outpost ID:** -
- Hostname type:** IP name
- Owner:** 615948015803
- Block Public Access:** Off
- IPv6 CIDR association ID:** -
- Route table:** rtb-053c8a5519a471e67 | 8byte-rt
- Auto-assign IPv6 address:** No
- IPv4 CIDR reservations:** -
- Resource name DNS AAAA record:** Disabled

Internet gateway

The screenshot shows the AWS VPC Internet gateway details page. The internet gateway ID is igw-06c0f1267650d5a3d. Key details include:

- Internet gateway ID:** igw-06c0f1267650d5a3d
- State:** Attached
- VPC ID:** vpc-058c06fbf476e3fef | 8byte-vpc
- Owner:** 615948015803

Tags (1):

Key	Value
Name	8byte-igw

Security groups

EC2 > Security Groups > sg-0b8ba60e90967b419 - 8byte-sg

Details

Security group name sg-8byte-sg	Security group ID sg-0b8ba60e90967b419	Description Managed by Terraform	VPC ID vpc-058c06fbf476e3fef
Owner 615948015803	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-0eee652b7608ca92b	IPv4	Custom TCP	TCP	3000
-	sgr-0a24e264e8765ecd4	IPv4	SSH	TCP	22

Route table and subnet association

VPC dashboard <

Route tables > rtb-06f27a57a96015927 / 8byte-rt

Details Info

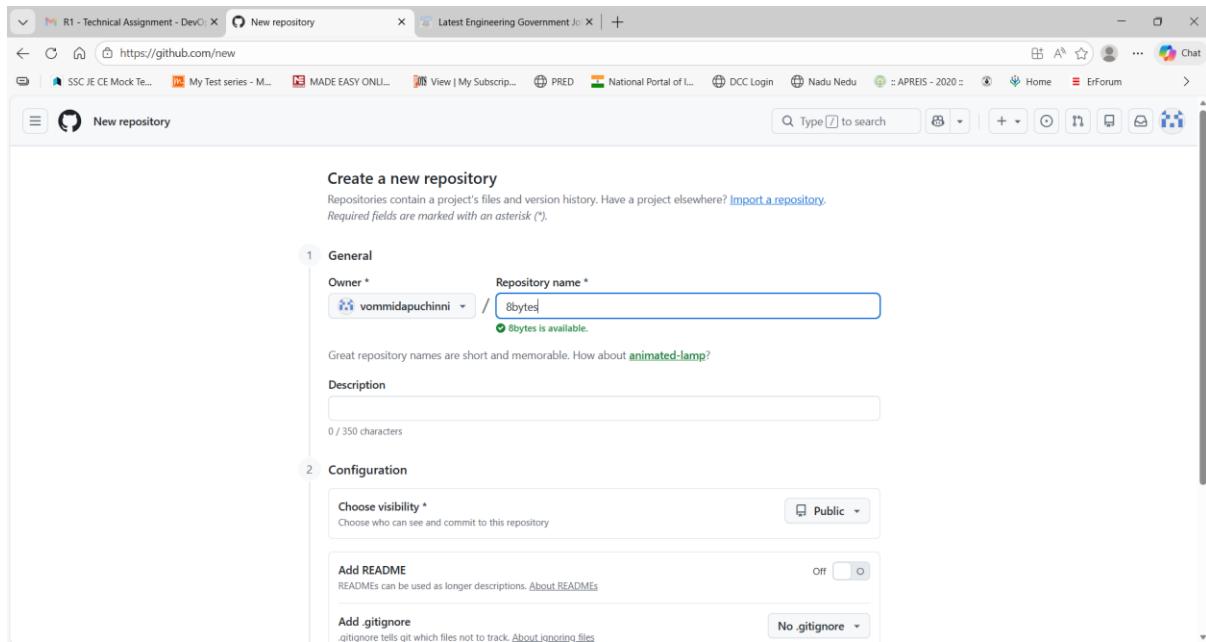
Route table ID rtb-06f27a57a96015927	Main No	Explicit subnet associations subnet-09fd66c0fb9c8cef8 / 8byte-public-subnet
VPC vpc-04fde247c35eb3dbc 8byte-vpc	Owner ID 615948015803	Edge associations -

Routes (2)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-0e0b394c0cbe0c1f7	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table

Push the code to github

create new git repository



Push code to github

```

umama@DESKTOP-HBLJM57 MINGW64 /c/8bytes
$ git init
Initialized empty Git repository in C:/8bytes/.git/
umama@DESKTOP-HBLJM57 MINGW64 /c/8bytes (main)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Dockerfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'terraform/terraform.lock.hcl', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'terraform/main.tf', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'terraform/outputs.tf', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'terraform/terraform.tfvars', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'terraform/variables.tf', LF will be replaced by CRLF the next time Git touches it
umama@DESKTOP-HBLJM57 MINGW64 /c/8bytes (main)
$ git commit -m "1st commit"
[main (root-commit) 69fcdf] 1st commit
 11 files changed, 1009 insertions(+)
create mode 100644 .gitignore
create mode 100644 Dockerfile
create mode 100644 app.js
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 terraform/.terraform.lock.hcl
create mode 100644 terraform/main.tf
create mode 100644 terraform/outputs.tf
create mode 100644 terraform/provider.tf
create mode 100644 terraform/terraform.tfvars
create mode 100644 terraform/variables.tf
umama@DESKTOP-HBLJM57 MINGW64 /c/8bytes (main)
$ git remote add origin https://github.com/vommidaipuchinni/8bytes.git
umama@DESKTOP-HBLJM57 MINGW64 /c/8bytes (main)
$ git push origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (14/14), 10.43 KiB | 534.00 KiB/s, done.

```

Deploy Application on EC2

Once the infrastructure is ready, we deploy the Dockerized Node.js application to the EC2 instance and verify it works in the cloud environment.

ssh -i key.pem ubuntu@EC2-PUBLIC-IP

SSH into EC2

```

ubuntu@ip-10-0-1-243:~$ ls
2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '24.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Feb  8 10:25:07 2026 from 157.50.95.82
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-1-243:~$ docker version
Client:
  Version:            28.2.2
  API version:        1.50
  Go version:         go1.23.1
  Git commit:         28.2.2-0ubuntu1~22.04.1
  Built:              Wed Sep 10 14:50:16 2025
  OS/Arch:            linux/amd64
  Context:             default

Server:
  Engine:
    Version:          28.2.2
    API version:      1.56 (minimum version 1.24)
    Go version:       go1.23.1
    Git commit:       28.2.2-0ubuntu1~22.04.1
    Built:            Wed Sep 10 14:50:16 2025
    OS/Arch:          linux/amd64
    Experimental:     false
  containerd:
    Version:          1.7.28
    GitCommit:        06b94a2518
  runc:
    Version:          1.3.3-0ubuntu1~22.04.3
    GitCommit:        9efc0a7d05
  docker-init:
    Version:          0.19.0
    GitCommit:        9efc0a7d05

```

check docker version and status by using the commands
docker version && systemctl status docker

Docker status

```

ubuntu@ip-10-0-1-243:~$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2026-02-08 10:01:39 UTC; 31min ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
 Main PID: 2023 (dockerd)
   Tasks: 9
    Memory: 31.9M
      CPU: 536ms
     CGroup: /system.slice/docker.service
             └─2023 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Feb 08 10:01:38 ip-10-0-1-243 dockerd[2023]: time="2026-02-08T10:01:38.954613193Z" level=info msg="Loading containers: start."
Feb 08 10:01:39 ip-10-0-1-243 dockerd[2023]: time="2026-02-08T10:01:39.250251548Z" level=info msg="Loading containers: done."
Feb 08 10:01:39 ip-10-0-1-243 dockerd[2023]: time="2026-02-08T10:01:39.277502695Z" level=info msg="Docker daemon" commit="28.2.2-0ubuntu1~22.04.1" container="00000000000000000000000000000000"
Feb 08 10:01:39 ip-10-0-1-243 dockerd[2023]: time="2026-02-08T10:01:39.277618286Z" level=info msg="Initializing buildkit"
Feb 08 10:01:39 ip-10-0-1-243 dockerd[2023]: time="2026-02-08T10:01:39.287128637Z" level=warning msg="CDI setup error: /var/run/cdi: failed to monitor for changes"
Feb 08 10:01:39 ip-10-0-1-243 dockerd[2023]: time="2026-02-08T10:01:39.287156981Z" level=warning msg="CDI setup error: /etc/cdi: failed to monitor for changes"
Feb 08 10:01:39 ip-10-0-1-243 dockerd[2023]: time="2026-02-08T10:01:39.320655535Z" level=info msg="Completed buildkit initialization"
Feb 08 10:01:39 ip-10-0-1-243 dockerd[2023]: time="2026-02-08T10:01:39.328472831Z" level=info msg="Daemon has completed initialization"
Feb 08 10:01:39 ip-10-0-1-243 dockerd[2023]: time="2026-02-08T10:01:39.328723178Z" level=info msg="API listen on /run/docker.sock"
Feb 08 10:01:39 ip-10-0-1-243 systemd[1]: Started Docker Application Container Engine.

ubuntu@ip-10-0-1-243:~$ 

```

clone repo to EC2 from github

```

ubuntu@ip-10-0-1-243:~$ git clone https://github.com/vommidapuchinni/8bytes.git
Cloning into '8bytes'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 14 (delta 0), reused 14 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (14/14), 10.43 KiB | 5.22 MiB/s, done.
ubuntu@ip-10-0-1-243:~$ ls
8bytes
ubuntu@ip-10-0-1-243:~$ cd 8bytes/
ubuntu@ip-10-0-1-243:~/8bytes$ ls
Dockerfile app.js package-lock.json package.json terraform

```

docker build -t app .

Docker build on EC2

```
ubuntu@ip-10-0-1-243:~/8bytes$ docker build -t 8byte-app .
DEPRECATION: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 112.6kB
Step 1/7 : FROM node:18
18: Pulling from library/node
3e6b9d1a9511: Pulling fs layer
37927ed901b1: Pulling fs layer
79b2f47ad444: Pulling fs layer
e23f099911d6: Pulling fs layer
cd47f44f2bdd: Pulling fs layer
c6b30c3f1696: Pulling fs layer
3697be50c98b: Pulling fs layer
461077a72fb7: Pulling fs layer
cd47f44f2bdd: Waiting
c6b30c3f1696: Waiting
3697be50c98b: Waiting
461077a72fb7: Waiting
e23f099911d6: Waiting
37927ed901b1: Verifying Checksum
37927ed901b1: Download complete
3e6b9d1a9511: Verifying Checksum
3e6b9d1a9511: Download complete
79b2f47ad444: Verifying Checksum
79b2f47ad444: Download complete
cd47f44f2bdd: Verifying Checksum
cd47f44f2bdd: Download complete
3697be50c98b: Verifying Checksum
3697be50c98b: Download complete
461077a72fb7: Verifying Checksum
461077a72fb7: Download complete
c6b30c3f1696: Verifying Checksum
c6b30c3f1696: Download complete
e23f099911d6: Verifying Checksum
e23f099911d6: Download complete
3e6b9d1a9511: Pull complete
37927ed901b1: Pull complete
79b2f47ad444: Pull complete
e23f099911d6: Pull complete
```

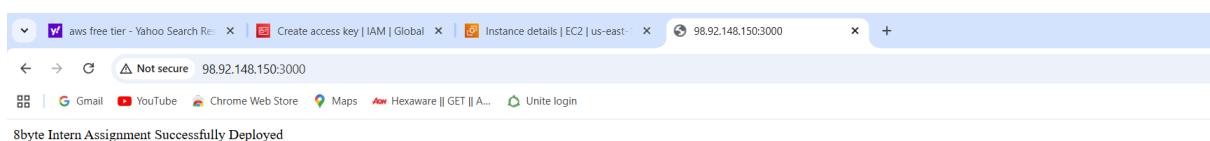
docker run -d -p 3000:3000 app

Docker run on EC2

```
ubuntu@ip-10-0-1-243:~/8bytes$ docker run -d -p 3000:3000 --name app 8byte-app
99f3b6a6855a03f9089abe7a04bda8feb2afdeed5d4296f72a096210b1646aa
ubuntu@ip-10-0-1-243:~/8bytes$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
99f3b6a6855a        8byte-app          "docker-entrypoint.s..."   37 seconds ago    Up 36 seconds   0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp   app
ubuntu@ip-10-0-1-243:~/8bytes$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
8byte-app           latest              d5e1bdaf9c0        43 seconds ago   1.1GB
node                18                 b50082bc3670       10 months ago    1.09GB
ubuntu@ip-10-0-1-243:~/8bytes$
```

Access in browser: <http://:3000>

Access app



GitHub Actions CI/CD

We will automate building, testing, and optionally pushing the Docker image using GitHub Actions to implement a full CI/CD pipeline.

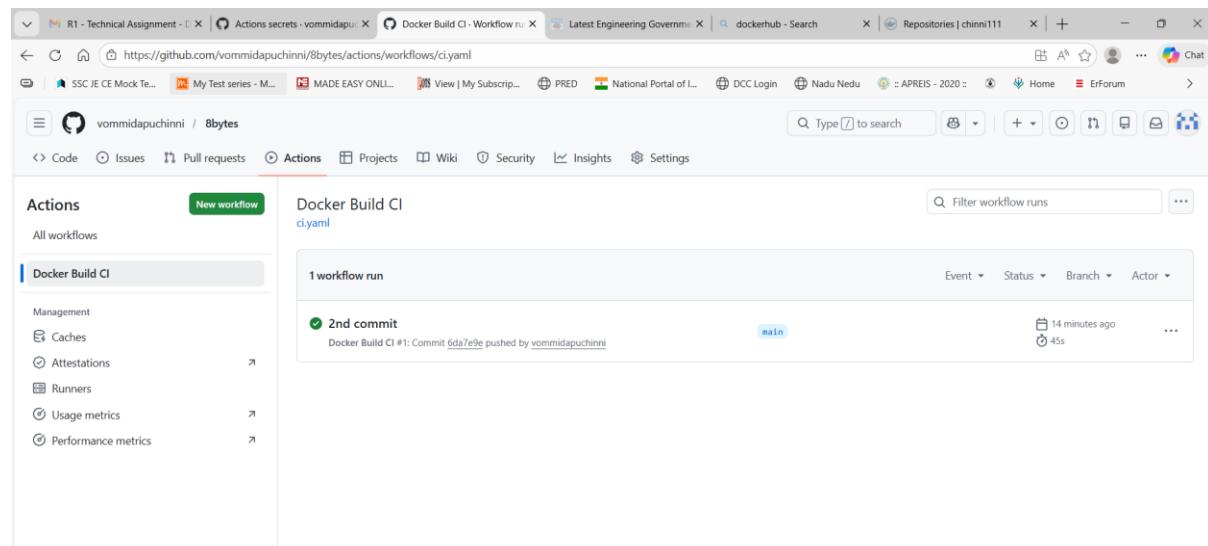
Workflow Location: .github/workflows/ci.yml

Creation of workflow files and push to github

```
MINGW64:/c/8bytes/.github/workflows
bash: cd: .github: No such file or directory
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes (main)
$ mkdir .github
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes (main)
$ cd .github
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes/.github (main)
$ mkdir workflows
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes/.github (main)
$ cd workflows/
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes/.github/workflows (main)
$ ls
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes/.github/workflows (main)
$ vi ci.yaml
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes/.github/workflows (main)
$ git add .
warning: in the working copy of '.github/workflows/ci.yaml', LF will be replaced by CRLF the next time Git touches it
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes/.github/workflows (main)
$ git commit -m "2nd commit"
[main 6da7e9e] 2nd commit
 1 file changed, 30 insertions(+)
 create mode 100644 .github/workflows/ci.yaml
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes/.github/workflows (main)
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 652 bytes | 163.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/vommidapuchinni/8bytes.git
 69fcdf..6da7e9e main -> main
umama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes/.github/workflows (main)
$
```

Pipeline Behavior

Triggered on push to the main branch.



repo secrets to login to Docker Hub

The screenshot shows the 'Actions secrets and variables' page in GitHub. The left sidebar has sections like General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Models, Webhooks, Copilot, Environments, Codespaces, Pages), Security (Advanced Security, Deploy keys), Secrets and variables, and Actions. The main area shows 'Environment secrets' (empty) and 'Repository secrets' with two entries: 'DOCKER_PASSWORD' (updated 17 minutes ago) and 'DOCKER_USERNAME' (updated 18 minutes ago). A 'New repository secret' button is visible.

Pipeline Steps:

1. Checkout repository
2. Set up Docker
3. Build Docker image
4. Verify successful build
5. Push image to Docker Hub

Workflow Successfully ran

The screenshot shows the 'Actions' tab for a GitHub repository. It displays a successful build named '2nd commit #1'. The build summary shows it succeeded 13 minutes ago. The workflow steps listed are: Set up job, Checkout Repository, Set up Docker Buildx, Log in to Docker Hub, Build Docker Image, Push Docker Image to Docker Hub, Post Log in to Docker Hub, Post Set up Docker Buildx, Post Checkout Repository, and Complete job. Each step is accompanied by a timestamp (e.g., 1s, 5s, 20s, 8s, 0s, 0s, 0s, 0s).

Docker Hub

The screenshot shows the Docker Hub interface. On the left, there's a sidebar for the user 'chinni111' with options like Repositories, Hardened Images, Collaborations, Settings, Default privacy, Notifications, Billing, Usage, Pulls, and Storage. The main area is titled 'Repositories' and shows a single repository: 'chinni111/8byte-intern-app'. It has a status of '15 minutes ago', is 'IMAGE' type, 'Public', and 'Inactive'. There's a 'Create a repository' button at the top right of the table.

Image pushed to docker hub

This screenshot shows the detailed view of the repository 'chinni111/8byte-intern-app'. The sidebar remains the same. The main content includes a 'General' tab showing the repository was last pushed about 1 hour ago, has 0 stars, and 19 forks. It also shows options to add a description or category. Below this is a 'Tags' section with a table:

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	about 1 hour

There's also a 'Repository overview' section labeled 'INCOMPLETE'.

This ensures every code change is automatically validated through CI.

The screenshot shows the GitHub Actions interface for a repository named '8bytes'. The 'Actions' tab is selected, displaying a workflow named 'Docker Build CI'. The workflow file is 'ci.yaml'. On the right, there is a list of '3 workflow runs' for the 'main' branch. The runs are ordered by event time: 4th commit (36 minutes ago), 3rd commit (43 minutes ago), and 2nd commit (52 minutes ago). Each run is shown with a green checkmark icon, indicating success. The runs were triggered by commits pushed by the user 'vommidapuchinni'.

Infrastructure Lifecycle & Cost Management

After successfully deploying and verifying the application on AWS, it is important to manage resources efficiently and avoid unnecessary costs.

The AWS infrastructure was provisioned using Terraform and the application deployment was successfully verified.

To follow responsible cloud usage practices and minimize charges, the infrastructure was destroyed after validation using: terraform destroy

```

MINGW64:/c/8bytes/terraform
$ umama@DESKTOP-HBLJM57 MINGW64 ~/8bytes (main)
$ cd terraform/
$ ls
main.tf outputs.tf provider.tf terraform.tfstate terraform.tfstate.backup terraform.tfvars variables.tf
$ terraform destroy
data.aws_ami.ubuntu: Reading...
aws_vpc.main: Refreshing state... [id=vpc-058c06fbf476e3fef]
data.aws_ami.ubuntu: Read complete after 2s [id=ami-0030e4319cbf4dbf2]
aws_internet_gateway.gw: Refreshing state... [id=igw-06c0f1267650d5a3d]
aws_subnet.main: Refreshing state... [id=subnet-046e488860d13780]
aws_security_group.sg: Refreshing state... [id=sg-0b8ba60e90967b419]
aws_route_table.rt: Refreshing state... [id=rtb-053c8a5519a471e67]
aws_route_table_association.a: Refreshing state... [id=rtaassoc-0dec0aed0d725e91]
aws_instance.app: Refreshing state... [id=i-0e40a9c74b1fe4b61]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_instance.app will be destroyed
- resource "aws_instance" "app" {
    - ami
    - arn
    - associate_public_ip_address
    - availability_zone
    - disable_api_stop
    - disable_api_termination
    - ebs_optimized
    - force_destroy
    - get_password_data
    - hibernation
    - id
    - instance_initiated_shutdown_behavior = "stop" -> null
    - instance_state = "running" -> null
    - instance_type = "t3.micro" -> null
    - ipv6_address_count = 0 -> null
}

```

```

MINGW64/c/8bytes/terraform
  - "Name" = "8byte-vpc"
  } -> null
  - tags_all = {
    - "Name" = "8byte-vpc"
    } -> null
    # (4 unchanged attributes hidden)
}

Plan: 0 to add, 0 to change, 7 to destroy.

Changes to Outputs:
- public_ip   = "98.92.167.154" -> null
- ssh_command = "ssh -i 8byte-key.pem ubuntu@98.92.167.154" -> null

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_route_table_association.a: Destroying... [id=rtbassoc-0d0ec0aed0d725e91]
aws_instance.app: Destroying... [id=i-0e40a9c74b1fe4b61]
aws_route_table_association.a: Destruction complete after 2s
aws_route_table.rt: Destroying... [id=rb-053cb8a5519a471e67]
aws_route_table.rt: Destruction complete after 2s
aws_internet_gateway.gw: Destroying... [id=igw-06c0f1267650d5a3d]
aws_instance.app: Still destroying... [id=i-0e40a9c74b1fe4b61, 00m10s elapsed]
aws_internet_gateway.gw: Still destroying... [id=igw-06c0f1267650d5a3d, 00m10s elapsed]
aws_instance.app: Still destroying... [id=i-0e40a9c74b1fe4b61, 00m20s elapsed]
aws_internet_gateway.gw: Still destroying... [id=igw-06c0f1267650d5a3d, 00m20s elapsed]
aws_internet_gateway.gw: Destruction complete after 2s
aws_instance.app: Still destroying... [id=i-0e40a9c74b1fe4b61, 00m30s elapsed]
aws_instance.app: Destruction complete after 35s
aws_subnet.public: Destroying... [id=subnet-046e488860d133780]
aws_security_group.sg: Destroying... [id=sq-0bb8a60e90967b419]
aws_subnet.public: Destruction complete after 2s
aws_security_group.sg: Destruction complete after 2s
aws_vpc.main: Destroying... [id=vpc-058c067bf476e3ref]
aws_vpc.main: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.

jumama@DESKTOP-HBLJMS7 MINGW64 /c/8bytes/terraform (main)
$ 

```

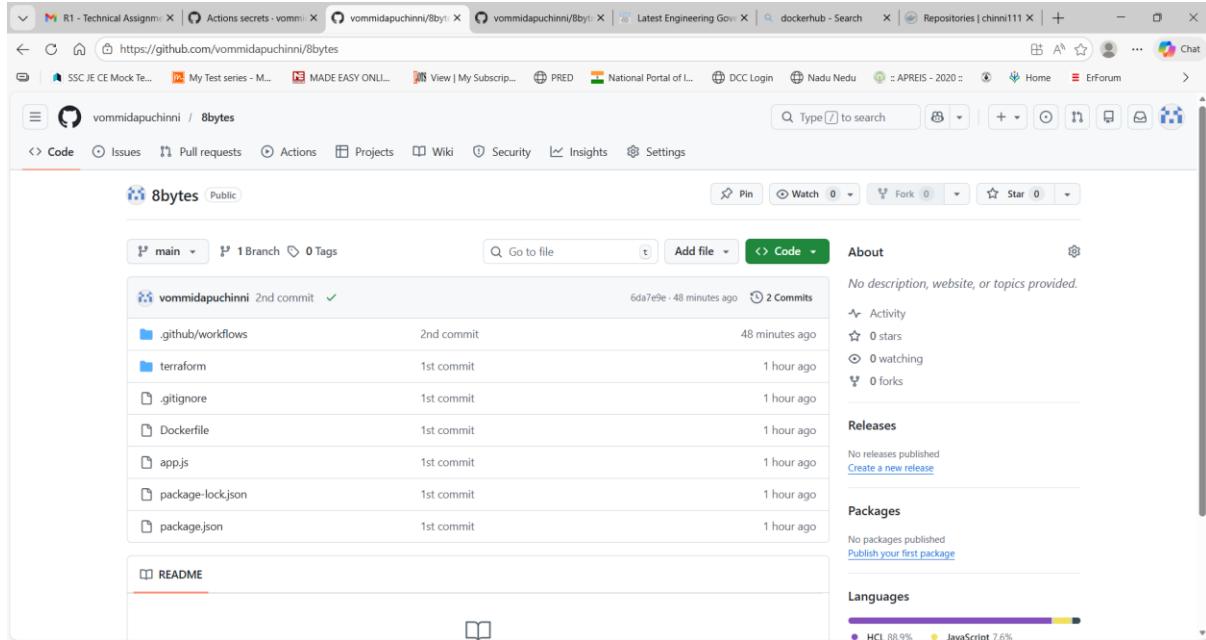
Since the setup is fully automated with Infrastructure as Code, the environment can be recreated at any time. I am prepared to redeploy the complete infrastructure live during evaluation if required.

Final Outcome

This project demonstrates a production-style DevOps workflow that covers the full lifecycle from setup to automation, including:

1. Cloud Infrastructure Provisioning
2. Secure AWS Networking
3. Dockerized Application Deployment
4. CI/CD Automation
5. Reproducible Infrastructure

FINAL Full repo



Conclusion

This project demonstrates a complete DevOps workflow by containerizing a Node.js application, provisioning AWS infrastructure with Terraform, and automating builds using GitHub Actions. It showcases practical skills in cloud deployment, Infrastructure as Code, CI/CD automation, and cost-efficient infrastructure management.