Overview

The WeatherApp is a Nest.js application that generates random weather information.

Purpose of Documentation

This documentation provides step-by-step instructions for setting up, running, and customizing the WeatherApp.

It is intended for developers and users who want to understand and use the application effectively.

Github Repository

```
https://github.com/imtiaz2564/weather-app
```

Accessing Weather Information

You can access random weather information by making a GET request to the following endpoint:

`http://104.248.101.152:3000/weather`

This endpoint will return a JSON response with random weather data.

Development steps:

Step 1: DigitalOcean Kubernetes Cluster

Cluster name: k8s-1-27-4-do-0-fra1-1694645877578
nodes in the cluster: pool-8bnjtg28s-y6nzf, pool-8bnjtg28s-y6nzy
Load Balancer: a8b7e8007b6754bcfa0168d0096203aa

Step 2: Dockerized Nest.js Backend

Registry Name: `registry.digitalocean.com/weatherregistery`
Image: `registry.digitalocean.com/weatherregistery/weatherapp:2174714`

Step 3: Set Up GitHub Action

```
https://github.com/imtiaz2564/weather-app/tree/main/.github/workflows
```

```
name: Docker Build and Push

on:
  push:
    branches:
      - main  # Replace with your main branch name

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Install doctl
        uses: digitalocean/action-doctl@v2
        with:
          token: ${{ secrets.DIGITALOCEAN_ACCESS_TOKEN }}

      # Build a Docker image of your application in your registry and tag the
image with the $GITHUB_SHA.
      - name: Build container image
        run: docker build -t
```

```
registry.digitalocean.com/weatherregistery/weatherapp:$(echo $GITHUB_SHA | head
-c7) .

      - name: Log in to DigitalOcean Container Registry with short-lived
credentials
        run: doctl registry login --expiry-seconds 1200

      - name: Push image to DigitalOcean Container Registry
        run: docker push
registry.digitalocean.com/weatherregistery/weatherapp:$(echo $GITHUB_SHA | head
-c7)
```

Step 4: Deploy to Kubernetes

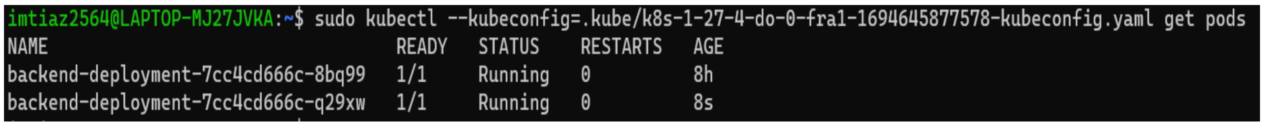
Backend Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "19"
  creationTimestamp: "2023-09-14T16:43:39Z"
  generation: 28
  labels:
    app: backend-deployment
  name: backend-deployment
  namespace: default
  resourceVersion: "1427228"
  uid: 6d33566d-0126-4b1d-902f-2b4d8d9b43f9
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: backend-deployment
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: backend-deployment
    spec:
      containers:
      - image:
registry.digitalocean.com/weatherregistery/weatherapp:2174714
        imagePullPolicy: IfNotPresent
        name: weatherapp
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
status:
  availableReplicas: 1
  conditions:
  - lastTransitionTime: "2023-09-18T14:57:56Z"
    lastUpdateTime: "2023-09-18T14:57:56Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2023-09-14T16:43:39Z"
    lastUpdateTime: "2023-09-18T15:18:27Z"
    message: ReplicaSet "backend-deployment-7cc4cd666c" has successfully
progressed.
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 28
  readyReplicas: 1
  replicas: 1
  updatedReplicas: 1
```

Backend service:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"we
atherapp-service","namespace":"default"},"spec":{"ports":[{"name":"http","p
ort":3000,"protocol":"TCP","targetPort":3000}],"selector":{"app":"backend-d
eployment"},"type":"LoadBalancer"}}
    kubernetes.digitalocean.com/load-balancer-id:
2bffc01c-fb12-4077-88a6-9d9bfa1a6140
    service.beta.kubernetes.io/do-loadbalancer-protocol: tcp
  creationTimestamp: "2023-09-16T13:00:34Z"
  finalizers:
  - service.kubernetes.io/load-balancer-cleanup
  name: weatherapp-service
  namespace: default
  resourceVersion: "865201"
  uid: 8b7e8007-b675-4bcf-a016-8d0096203aab
spec:
  allocateLoadBalancerNodePorts: true
  clusterIP: 10.245.8.210
  clusterIPs:
  - 10.245.8.210
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http
    nodePort: 32102
    port: 3000
    protocol: TCP
    targetPort: 3000
  selector:
    app: backend-deployment
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 104.248.101.152
```



```
imtiaz2564@LAPTOP-MJ27JVKA:~$ sudo kubectl --kubeconfig=.kube/k8s-1-27-4-do-0-fra1-1694645877578-kubeconfig.yaml get pods
NAME                                  READY   STATUS    RESTARTS   AGE
backend-deployment-7cc4cd666c-8bq99   1/1     Running   0          8h
backend-deployment-7cc4cd666c-q29xw   1/1     Running   0          8s
```

Application link: http://104.248.101.152:3000/

Future Work:

Create a Dockerized Next.js Frontend
Create Kubernetes deployment configuration and service configurations