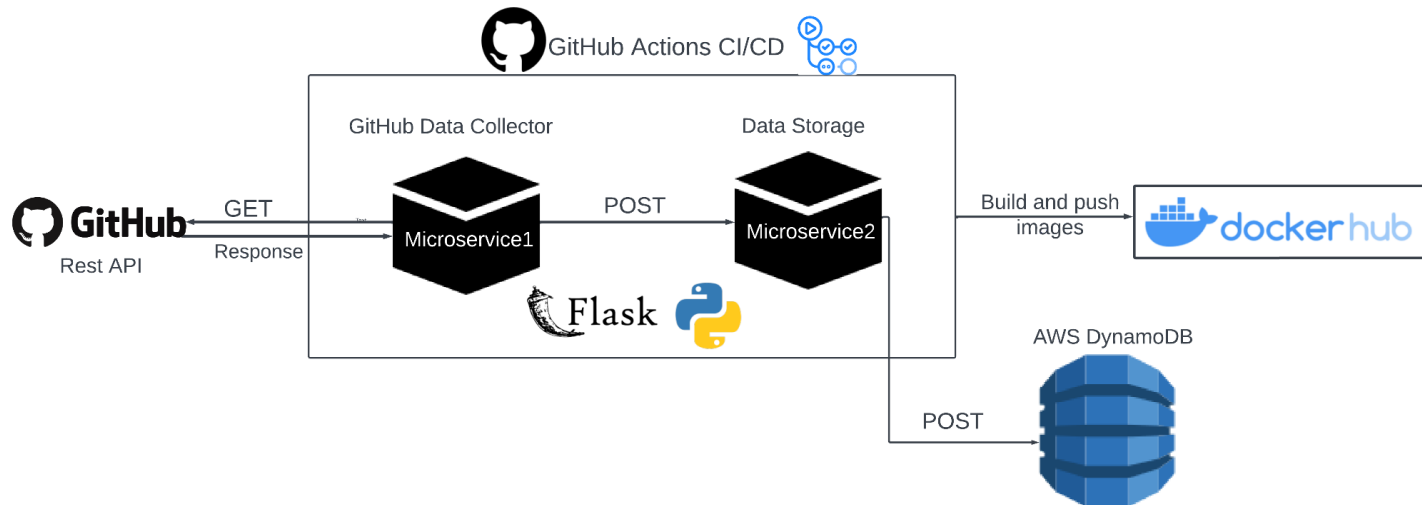


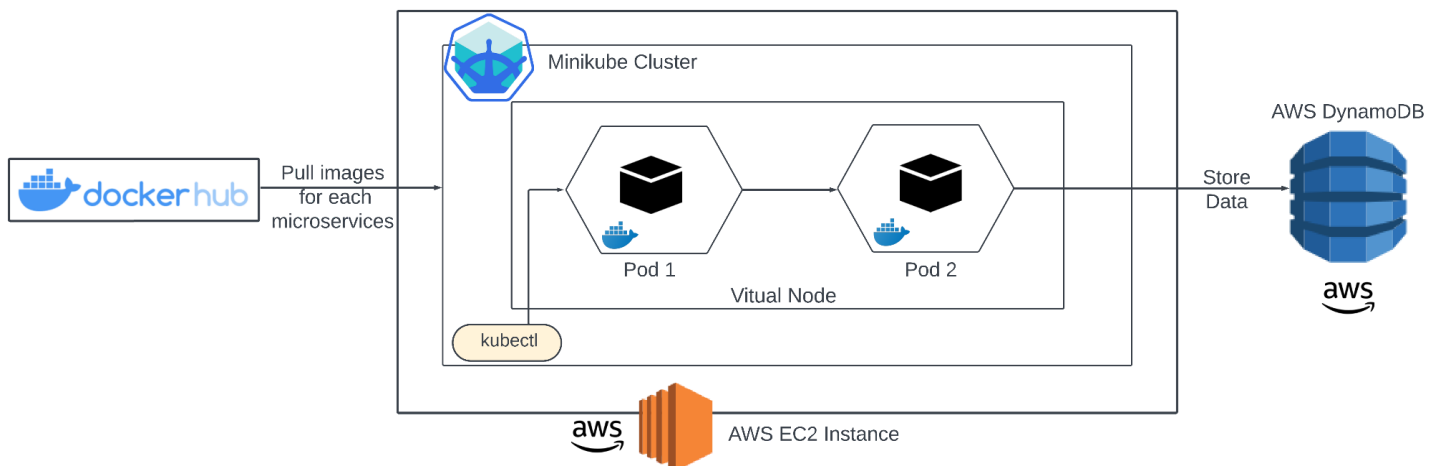
# Implementation of DeveloperIQ Tracking App for Developers.

## 1.1 Solution Architecture Diagram



- ❖ Developed 2 microservices such as GitHub\_Data\_Collector Microservice, and Data\_Storage microservice using Python.
- ❖ Used Python Flask as the web framework.
- ❖ GitHub\_Data\_Collector microservice is using GitHub Rest API to get data.
- ❖ Created “Developer-Tracking-Data” DynamoDB table. Developer\_Username is the partition key as it is unique.

## 1.2 Deployment Architectural Diagram



- ❖ Refer to 2 images for each of the microservices from Docker Hub and create separate pods for each microservice in the Minikube cluster.

- ❖ Replicas are mentioned respectively for each microservice in the deployment YAML files.
- ❖ Both github\_data\_collector microservice and data\_storage have 2 replicas. The DeveloperIQ services will run periodically only when triggered.

## 2. Security and Ethic Challenges Faced during Cloud Implementation - DeveloperIQ

### Security Challenges:

- ❖ Storing and transmitting data
  - DOCKER\_USERNAME, DOCKER\_PASSWORD, AWS\_ACCESS\_KEY\_ID, and AWS\_SECRET\_ACCESS\_KEY are stored in GitHub secrets.
  - Credentials of that IAM role such as AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY, and AWS\_REGION are stored in GitHub secrets
- ❖ Access Control
  - Created an IAM role with full read-write access to AWS DynamoDB to access the DynamoDB table.

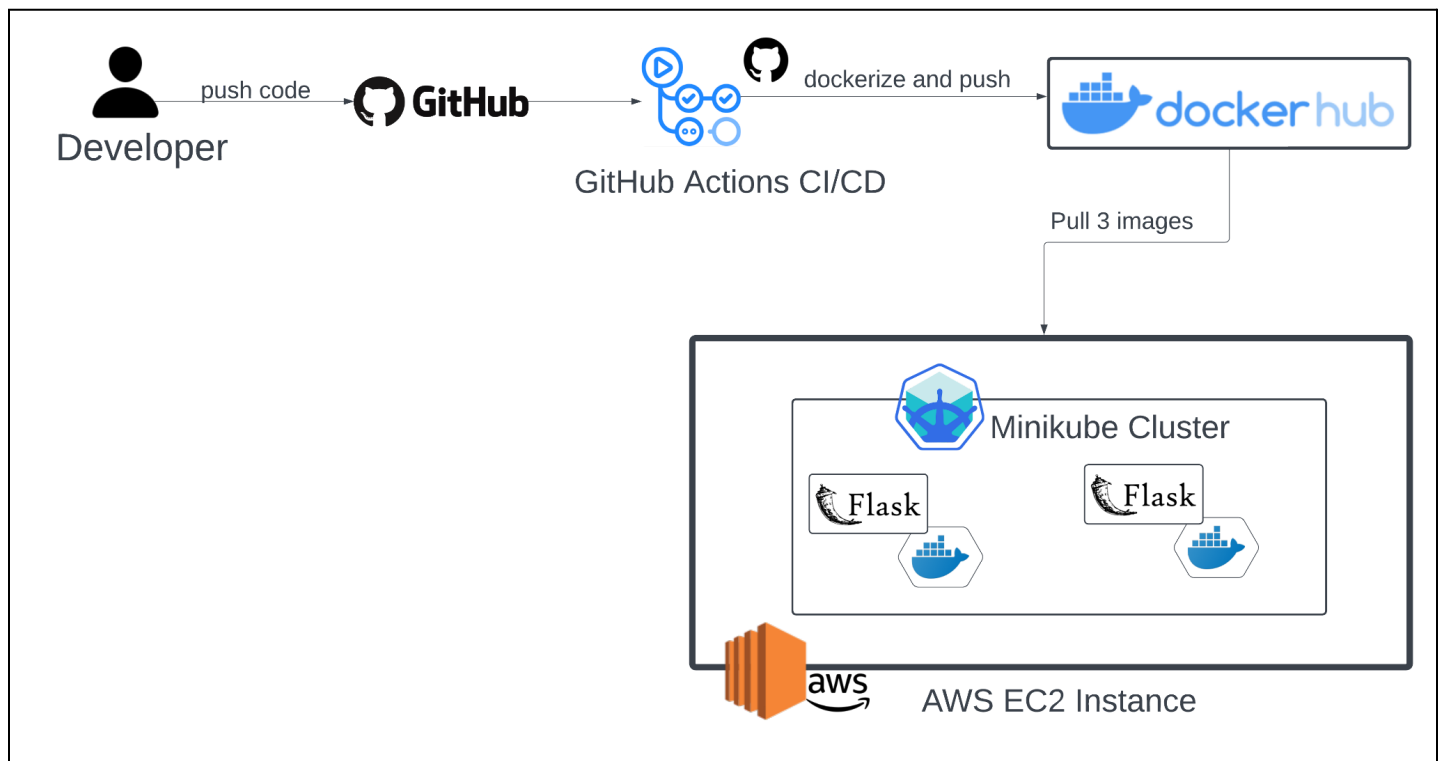
### Ethical Challenges :

- ❖ Bias in testing
  - Without using the same repo for development and testing using some other GitHub public repository.
- ❖ Transparency and Accountability:
  - Maintain clear documentation, communicate openly about testing procedures, and establish accountability for the development, testing, and deployment phases.

```
env:  
  aws_access_key_id: ${ secrets.AWS_ACCESS_KEY_ID }  
  aws_secret_access_key: ${ secrets.AWS_SECRET_ACCESS_KEY }
```

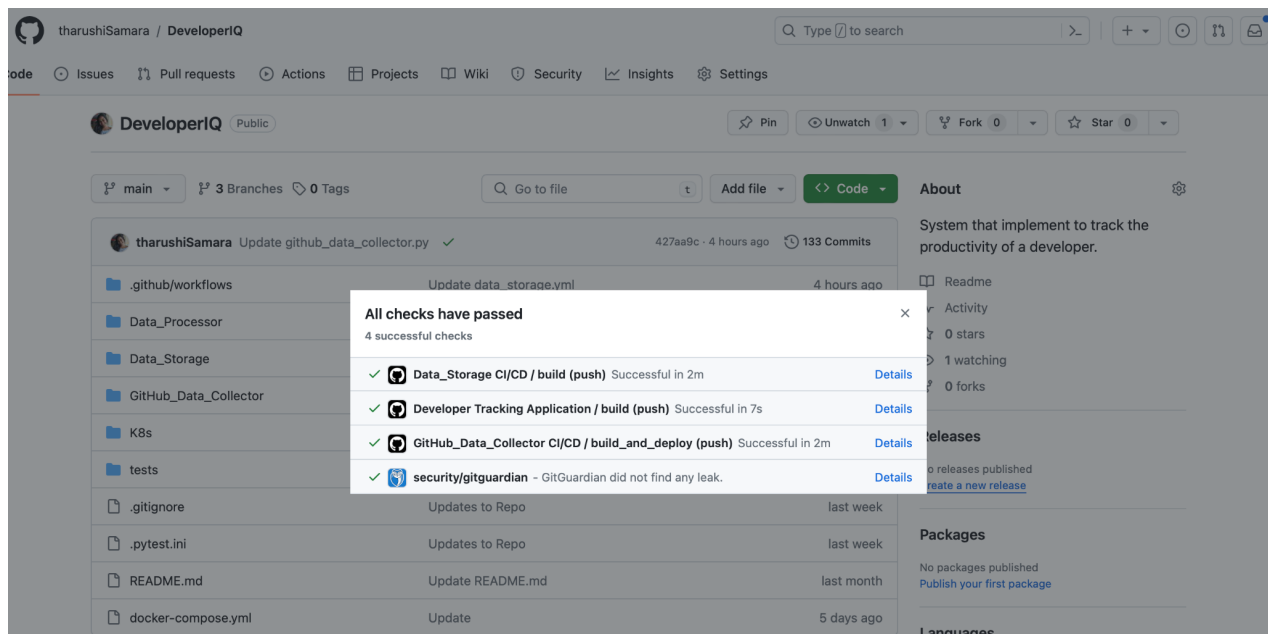
### 3. Implementation of CI/CD Pipeline using GitHub Actions

#### ❖ Design Diagram



#### ❖ Process Descriptions

- After every push the CI/CD pipelines will automatically start and files in `.github/workflow` scripts.
- CI/CD pipelines will run for each microservice.



## 4. CI/CD Scripts

### ❖ github\_data\_collector.yml

```
name: GitHub_Data_Collector CI/CD

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build_and_deploy:
    runs-on: ubuntu-latest

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

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v1

      - name: Log in to Docker Hub
        uses: docker/login-action@v1
        with:
          username: ${ secrets.DOCKER_USERNAME }
          password: ${ secrets.DOCKER_PASSWORD }

      - name: Build and push Docker image
        run: |
          docker buildx build -t tharushisamara/github-data-collector:latest --file
          ./GitHub_Data_Collector/Dockerfile --platform linux/amd64,linux/arm64 --push .

      - name: Connect with EC2
        uses: appleboy/ssh-action@master
        with:
          host: ${ secrets.EC2_HOST }
          username: ${ secrets.EC2_USERNAME }
          key: ${ secrets.EC2_SSH_KEY }
          script: |
            docker pull tharushisamara/github-data-collector:latest
```

```

- name: Copy deployment file to EC2
  uses: appleboy/scp-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    source: "./K8s/github_data_collector_deployment.yml"
    target: "/home/${ secrets.EC2_USERNAME }/"

- name: Delete Kubernetes Service if exists
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    script: |
      kubectl delete -f ./K8s/github_data_collector_deployment.yml || true

      kubectl wait --for=delete service/github-data-collector --timeout=300s
--ignore-not-found || echo "Service not found or deletion completed"

- name: Deploy to Minikube
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    script: |
      kubectl apply -f ./K8s/github_data_collector_deployment.yml
      kubectl get pods

      kubectl wait --for=condition=available --timeout=150s
deployment/github-data-collector
# run: |
#   whoami
# kubectl apply -f ./K8s/github_data_collector_deployment.yml
# kubectl wait --for=condition=available --timeout=150s
deployment/github-data-collector

```

```
- name: Getting Minikube IP
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    script: |
      minikube ip
      kubectl get service github-data-collector-service
-o=jsonpath='{.spec.ports[0].nodePort}'

- name: Delete Kubernetes Service if exists
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    script: |
      kubectl delete service github-data-collector --ignore-not-found

- name: Expose deployment
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    script: |
      kubectl get pods
      kubectl expose deployment github-data-collector --type=LoadBalancer
--port=8080

- name: Expose Minikube service externally (for demo purposes)
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    script: |
      minikube service github-data-collector-service --url
      kubectl get svc
```

## ❖ Data\_storage.yml

```
name: Data_Storage CI/CD

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

env:
  aws_access_key_id: ${ secrets.AWS_ACCESS_KEY_ID }
  aws_secret_access_key: ${ secrets.AWS_SECRET_ACCESS_KEY }

jobs:
  build:
    runs-on: ubuntu-latest

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

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v1

      - name: Set up Docker CLI
        uses: docker/login-action@v1
        with:
          username: ${ secrets.DOCKER_USERNAME }
          password: ${ secrets.DOCKER_PASSWORD }

      - name: Build and push Docker image
        run: |
          docker buildx build -t tharushisamara/data-storage:latest --file
          ./Data_Storage/Dockerfile --platform linux/amd64,linux/arm64 --push .

      - name: Deploy to Minikube on EC2
        uses: appleboy/ssh-action@master
        with:
          host: ${ secrets.EC2_HOST }
          username: ${ secrets.EC2_USERNAME }
          key: ${ secrets.EC2_SSH_KEY }
          script: |
```

```

    docker pull tharushisamara/data-storage:latest

- name: Copy deployment file to EC2
  uses: appleboy/scp-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    source: "./K8s/data_storage_deployment.yml"
    target: "/home/${ secrets.EC2_USERNAME }/"

- name: Set Environment Variables on EC2
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    script: |
      echo "export aws_access_key_id=${ secrets.AWS_ACCESS_KEY_ID }" >>
~/ .bashrc
      echo "export aws_secret_access_key=${ secrets.AWS_SECRET_ACCESS_KEY }" >>
~/ .bashrc
      source ~/ .bashrc

- name: Deploy to Minikube
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }
    key: ${ secrets.EC2_SSH_KEY }
    script: |
      kubectl apply -f ./K8s/data_storage_deployment.yml
      kubectl get pods

      kubectl wait --for=condition=available --timeout=150s deployment/data-storage

- name: Getting Minikube IP
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }
    username: ${ secrets.EC2_USERNAME }

```



```

    key: ${ secrets.EC2_SSH_KEY }}
    script: |
        minikube ip
        kubectl get service data-storage-service
-o=jsonpath='{.spec.ports[0].nodePort}'

- name: Check if Service Exists
  id: check_service
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }}
    username: ${ secrets.EC2_USERNAME }}
    key: ${ secrets.EC2_SSH_KEY }}
    script: |
        SERVICE_NAME="data-storage-service"
        NAMESPACE="default"
        if kubectl get service "$SERVICE_NAME" -n "$NAMESPACE" &> /dev/null; then
            echo "::set-output name=service_exists::true"
        else
            echo "::set-output name=service_exists::false"
        fi

- name: Expose deployment
  if: steps.check_service.outputs.service_exists == 'false'
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }}
    username: ${ secrets.EC2_USERNAME }}
    key: ${ secrets.EC2_SSH_KEY }}
    script: |
        kubectl get pods
        kubectl expose deployment data-storage --type=LoadBalancer --port=8081

- name: Expose Minikube service externally (for demo purposes)
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.EC2_HOST }}
    username: ${ secrets.EC2_USERNAME }}
    key: ${ secrets.EC2_SSH_KEY }}
    script: |
        minikube service data-storage --url

```

```
kubectl get svc
```

## ❖ Deployment and service scripts

➤ github\_data\_collector.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: github-data-collector
spec:
  replicas: 2
  selector:
    matchLabels:
      app: github-data-collector
  template:
    metadata:
      labels:
        app: github-data-collector
    spec:
      containers:
        - name: github-data-collector
          image: tharushisamara/github-data-collector:latest
          imagePullPolicy: Always
          envFrom:
            - configMapRef:
                name: github-config
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: github-data-collector-service
spec:
  selector:
    app: github-data-collector
  ports:
    - protocol: "TCP"
      port: 8080
      targetPort: 8080
  type: LoadBalancer
```

## ❖ data\_storage\_deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: data-storage
spec:
  replicas: 2
  selector:
    matchLabels:
      app: data-storage
  template:
    metadata:
      labels:
        app: data-storage
    spec:
      containers:
        - name: data-storage
          image: tharushisamara/data-storage:latest
          imagePullPolicy: Always
          envFrom:
            - configMapRef:
                name: dynamo-config
          ports:
            - containerPort: 8081
---
apiVersion: v1
kind: Service
metadata:
  name: data-storage-service
spec:
  selector:
    app: data-storage
  ports:
    - protocol: "TCP"
      port: 8081
      targetPort: 8081
  type: LoadBalancer
```

## 5. Observability Cluster

### ❖ Tunneling

```
ssh -i node-js.pem -L 6565:10.105.112.250:8081 ec2-user@13.234.239.76
```

### ❖ Using Minikube Dashboard

Workloads > Pods									
Pods									
	Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑
	data-storage-5bcd95747-4n85x	tharushisamara/data-storage:latest	app: data-storage pod-template-hash: 5bcd95747	minikube	Running	0	-	-	.15 minutes ago
	data-storage-5bcd95747-nq8l4	tharushisamara/data-storage:latest	app: data-storage pod-template-hash: 5bcd95747	minikube	Running	0	-	-	.15 minutes ago
	github-data-collector-6c6df97b8-dqygh	tharushisamara/github-data-collector:latest	app: github-data-collector pod-template-hash: 6c6df97b8	minikube	Running	0	-	-	.15 minutes ago
	github-data-collector-6c6df97b8-zd2pk	tharushisamara/github-data-collector:latest	app: github-data-collector pod-template-hash: 6c6df97b8	minikube	Running	0	-	-	.15 minutes ago

### ❖ Daemon Sets - Deployments - Pods - Replica Sets

kubernetes

All namespaces

Search

Workloads

Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Service

Ingresses

Ingress Classes

Services

Config and Storage

Config Maps

Persistent Volume Claims

Secrets

Storage Classes

Cluster

Cluster Role Bindings

Cluster Roles

Events

Namespaces

Workload Status

Running: 1

Running: 5

Running: 13

Running: 5

Daemon Sets

Deployments

Pods

Replica Sets


Daemon Sets

Name	Namespace	Images	Labels	Pods	Created
kube-proxy	kube-system	registry.k8s.io/kube-proxy:v1.28.3	k8s-app: kube-proxy	1 / 1	2 days ago


Deployments



Name	Namespace	Images	Labels	Pods	Created
github-data-collector	default	tharushisamara/github-data-collector:latest	-	2 / 2	17 minutes ago
data-storage	default	tharushisamara/data-storage:latest	-	2 / 2	17 minutes ago
dashboard-metrics-scraper	kubernetes-dashboard	docker.io/kubernetes/metrics-scraper:v1.0.8@sha256:76049887f07a0476dc93efc2d3569b952	addonmanager.kubernetes.io/mode: Reconcile k8s-app: dashboard-metrics-scraper	1 / 1	8 days ago

### ❖ Replica Sets

kubernetes

All namespaces ▾

 Search



Workloads > Replica Sets

Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Service






Ingresses

Ingress Classes

Services

Config and Storage

Replica Sets

Name	Namespace	Images	Labels	Pods	Created ↑
 <a href="#">data-storage-5bcd95747</a>	default	tharushisamara/data-storage:latest	app: data-storage pod-template-hash: 5bcd95747	2 / 2	18 minutes ago
 <a href="#">github-data-collector-6c6df97b8</a>	default	tharushisamara/github-data-collector:latest	app: github-data-collector pod-template-hash: 6c6df97b8	2 / 2	18 minutes ago
 <a href="#">dashboard-metrics-scraper-7fd5cb4ddc</a>	kubernetes-dashboard	docker.io/kubernetes/metrics-scraper:v1.0.8@sha256:76049887f0780476dc93efc2d3569b9529bf082b22d29f356092ce206e98765c	k8s-app: dashboard-metrics-scraper pod-template-hash: 7fd5cb4ddc	1 / 1	a day ago
 <a href="#">kubernetes-dashboard-8694d4445c</a>	kubernetes-dashboard	docker.io/kubernetes/dashboard:v2.7.0@sha256:2e500d29e9d54a086b908eb8dfe7ecac57d2ab09d65b24f588b1d449841ef93	gcp-auth-skip-secret: true k8s-app: kubernetes-dashboard pod-template-hash: 8694d4445c	1 / 1	a day ago
 <a href="#">coredns-5dd5756b68</a>	kube-system	registry.k8s.io/coredns/coredns:v1.10.1	k8s-app: kube-dns pod-template-hash: 5dd5756b68	1 / 1	2 days ago

## ❖ Services

Services							
Name	Namespace	Labels	Type	Cluster IP	Internal Endpoints	External Endpoints	Created
data-storage	default	-	NodePort	10.102.25.187	data-storage:5003 TCP data-storage:30619 TCP	-	19 minutes ago
github-data-collector	default	-	NodePort	10.96.78.64	github-data-collector:5000 TCP github-data-collector:30496 TCP	-	19 minutes ago
github-data-collector-service	default	-	NodePort	10.101.22.92	github-data-collector-service:5000 TCP github-data-collector-service:31237 TCP	-	19 minutes ago
data-storage-service	default	-	NodePort	10.108.114.226	data-storage-service:5003 TCP data-storage-service:30999 TCP	-	19 minutes ago
data-processor	default	-	NodePort	10.102.114.253	data-processor:5002 TCP data-processor:30934 TCP	-	4 hours ago
data-processor-service	default	-	NodePort	10.110.209.102	data-processor-service:5002 TCP data-processor-service:30720 TCP	-	4 hours ago

## ❖ Running Pods with the respective replicas of each Pod

```
[ec2-user@ip-172-31-33-151 ~]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
data-storage-5bcd95747-4n85x        1/1     Running   0           3m19s
data-storage-5bcd95747-nq814        1/1     Running   0           3m19s
github-data-collector-6c6df97b8-dqvgh 1/1     Running   0           3m19s
github-data-collector-6c6df97b8-zd2pk 1/1     Running   0           3m19s
[ec2-user@ip-172-31-33-151 ~]$ minikube dashboard
[ec2-user@ip-172-31-33-151 ~]$ kubectl get deployments
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
data-storage                        2/2     2             2           22m
github-data-collector               2/2     2             2           22m
[ec2-user@ip-172-31-33-151 ~]$
```

## ❖ Running Nodes

Cluster > Nodes

Ingresses

Ingress Classes

Services

Config and Storage

Config Maps

Persistent Volume Claims

Nodes

minikube

beta.kubernetes.io/arch: amd64

beta.kubernetes.io/os: linux

kubernetes.io/arch: amd64

Show all

Ready

True

CPU requests (cores)

750.00m (37.50%)

CPU limits (cores)

0.00m (0.00%)

CPU capacity (cores)

2.00

Memory requests (bytes)

170.00Mi (8.94%)

Memory limits (bytes)

170.00Mi (8.94%)

Memory capacity (bytes)

1.86Gi

Pods

13 (11.82%)

Created

2 days ago

## ❖ API requests visibility using Postman

The screenshot displays the Postman application interface. At the top, the URL bar shows `http://127.0.0.1:6560/collect-developer-metrics`. The request method is set to **GET**. The **Body** tab is selected, showing a JSON payload:

```
1 {
2   "username": "MohamedSabthar",
3   "repo": "Smart-VAT"
4 }
```

Below the request, the **Body** tab of the response is selected, showing a JSON response with developer metrics:

```
13   "resolved_issues_count": 23
14 },
15   "PunsaraUCSC": {
16     "commit_count": 26,
17     "developer_username": "PunsaraUCSC",
18     "pull_requests_count": 30,
19     "resolved_issues_count": 13
20   },
21   "tharushiSamara": {
22     "commit_count": 30,
23     "developer_username": "tharushiSamara",
24     "pull_requests_count": 30,
25     "resolved_issues_count": 30
26   }
27 },
28 "repository": "Smart-VAT"
29 }
```

The status bar at the bottom indicates a **200 OK** response, a time of **9.73 s**, and a size of **725 B**. The **Save as example** button is visible.

## ❖ Storing in AWS DynamoDB table - Developer-Tracking-Data

[DynamoDB](#) > [Explore items: Developer-Tracking-Data](#) > [Edit item](#)

### Edit item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn](#)

#### Attributes

☒ View DynamoDB JSON

```
1 {
2   "developer_username": {
3     "S": "Smart-VAT-developer_metrics_2023-12-11 07:51:49.571763"
4   },
5   "variables": {
6     "M": {
7       "ImalshaRathnaweera": {
8         "M": {
9           "developer_username": {
10             "S": "ImalshaRathnaweera"
11           },
12           "commit_count": {
13             "N": "30"
14           },
15           "pull_requests_count": {
16             "N": "30"
17           },
18           "resolved_issues_count": {
19             "N": "30"
20           }
21         }
22       },
23       "MohamedSabthar": {
24         "M": {
```




JSON

Ln 1, Col 1

✖ Errors: 0

⚠ Warnings: 0

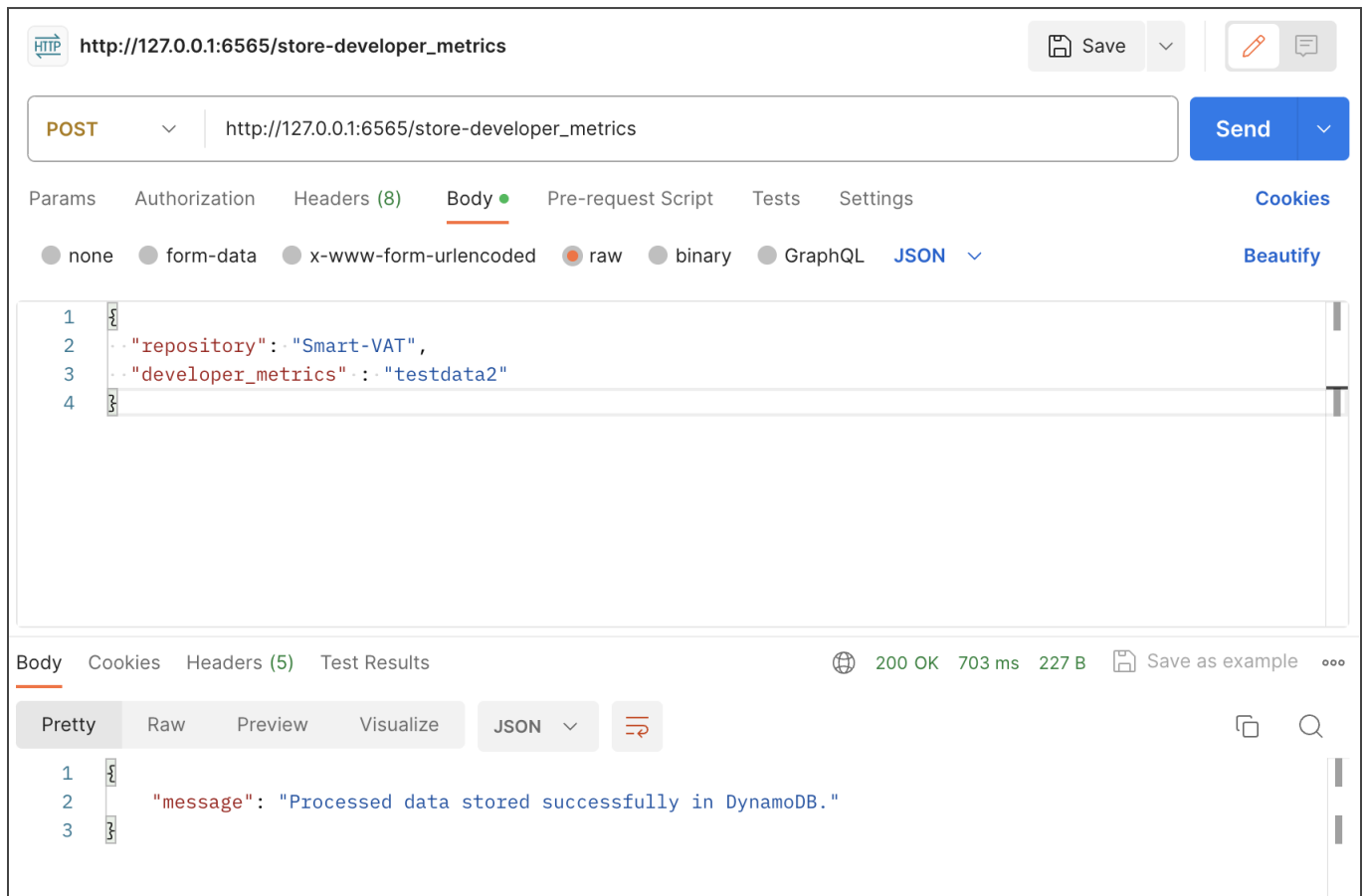
❖ In every run, the data is stored with the current date on the DynamoDB table

Items returned (16)			Actions ▾	Create item
		< 1 >  		
<input type="checkbox"/>	developer_username (String) ▾	variables ▾		
<input type="checkbox"/>	<a href="#">Smart-VAT-developer_metrics_2023-12-11 07:51:49.571763</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-developer_metrics_2023-12-11 03:53:58.071069</a>	testdata		
<input type="checkbox"/>	<a href="#">Smart-VAT-processed-data_2023-12-08 04:52:30.787555</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-developer_metrics_2023-12-11 04:28:01.959037</a>	testdata2		
<input type="checkbox"/>	<a href="#">Smart-VAT-processed-data_2023-12-07 16:10:45.152037</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-developer_metrics_2023-12-11 05:34:16.315201</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-processed-data_2023-12-07</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-developer_metrics_2023-12-11 09:05:13.092042</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-developer_metrics_2023-12-11 08:12:09.016096</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-processed-data</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">test</a>			
<input type="checkbox"/>	<a href="#">Smart-VAT-processed-data_2023-12-08 05:02:01.045552</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-processed-data_2023-12-07 16:55:13.931992</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-developer_metrics_2023-12-11 04:56:29.830921</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-processed-data_2023-12-07 16:13:30.815228</a>	{ "MohamedSabthar" : { "M" : { "commit_...		
<input type="checkbox"/>	<a href="#">Smart-VAT-developer_metrics_2023-12-11 04:35:14.885591</a>	testdata2		



## 6. Testing

Testing the data-storage microservice which is connected to the initial microservice This is tested via Postman to show that the 2nd microservice is working



# RunBook - DeveloperIQ

# Deploying DeveloperIQ App to Kubernetes(Minikube) on AWS EC2

## DeveloperIQ App Overview

DeveloperIQ App is designed using 2 microservices such as GitHub\_Data\_Collector and Data\_Storage and developed using Python Flask.

### Pre-requisites:

- ❖ GitHub Repository Setup:
  - Ensure your microservices code is hosted on a GitHub repository.
  - Set up GitHub Actions workflows for CI/CD (eg: github\_data\_collector.yml and data\_store.yml).
- ❖ Create Docker Hub Account:
  - Build Docker images for your microservices and push them to a container registry (e.g., Docker Hub).
  - Need to create separate Dockerfiles for every microservice. Since this app contains relatively small microservices, use one repo and create separate folders for each microservice.
- ❖ Create an AWS Account:
  - Create a separate IAM user in addition to the root user which had DynamoDB access, and EC2 access.
  - Store AWS\_ACCESS\_KEY\_ID and AWS\_SECRET\_ACCESS\_KEY of the IAM role on GitHub Secrets

Below is the file structure:

```
DeveloperIQ/
|--.github/
|  |--workflows/
|  |  |--python-app.yml
|  |  |--github_data_collector.yml
|  |  |--data_storage.yml
|-- GitHub_Data_Collector/
|  |-- github_data_collector.py
|  |-- requirements.txt
|  |-- Dockerfile
|-- Data_Storage/
|  |-- data_storage.py
```

```
| |-- requirements.txt  
| |-- Dockerfile  
|--K8s/  
| |--github_data_collector_deployment.yml  
| |--data_storage_deployment.yml  
|-- docker-compose.yml
```

## Deployment Steps:

### ❖ AWS EC2:

- Launch AWS EC2 instances while having at least minimum requirements
  - ❖ 2 CPUs or more
  - ❖ 2GB of free memory (t3 small)
  - ❖ 20GB of free disk space
- Store EC2 information such as EC2\_HOST, EC2\_SSH\_KEY, EC2\_USERNAME on GitHub Secrets
- Connect with the EC2 instance

```
ssh -i <PEM file> ec2-user@<Public_IP>
```

- Install minikube

```
curl -LO  
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd  
64  
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

- Install Docker

```
sudo yum install docker  
sudo usermod -aG docker ec2-user  
sudo service docker start
```

```
minikube start --driver=docker
```

### ❖ Install kubectl

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

sudo mv kubectl /usr/local/bin/

sudo chmod 755 /usr/local/bin/kubectl
```

❖ Test the kubectl installation

```
kubectl get namespaces
kubectl get pods -n kube-system
```

- ❖ Configure security groups to allow necessary traffic.
- ❖ Create a DynamoDB table
  - Store all the credentials such as DYNAMODB\_REGION on GitHub secrets.
- ❖ Update Code on GitHub:
  - Make changes to your microservices code and push it to the GitHub repository.
- ❖ GitHub Actions CI/CD:
  - GitHub Actions will trigger automatically upon code push.
  - CI workflow should include steps for building and testing microservices. Refer to the GitHub repository [here](#). The steps for each deployment are under **.github/workflows**
  - CD workflow should include steps for building Docker images and pushing them to the container registry.

Note: The below docker image building pushing it to Docker Hub and Deploy microservices to conduct the CI/CD pipeline. The below commands need to be run inside the workflow files separately for each microservice.

❖ Build and Push Docker Images to Minikube:

- Build Docker images locally using Minikube's Docker daemon:

```
docker build -t <your-image-name>:<tag> .
```

- Tag the image for minikube

```
docker tag <your-image-name>:<tag>  
$(minikubeip):5000/<your-image-name>:<tag>
```

➤ Push the image to Minikube's registry:

```
docker push $(minikube ip):5000/<your-image-name>:<tag>
```

❖ Deploy microservices to Minikube

```
kubectl apply -f path/to/deployment.yaml  
kubectl apply -f path/to/service.yaml
```

❖ Prepare Kubernetes Manifests:

- Create Kubernetes manifest files for your microservices (e.g., `github_data_collector_deployment.yaml`, `data_collector_deployment.yaml`).
- Note:- These `_deployments.yaml` files can be created separately as `deployment.yaml` and `services.yaml`
- Customize these files with appropriate configurations.

To verify the deployments need to connect to the EC2 instance locally. Type the below command in cmd/PowerShell.

```
ssh -i <EC2 secret key file>.pem ec2-user@<EC2 host ID>
```

❖ Verify Deployments:

```
kubectl get pods  
kubectl get svc
```

❖ Tunneling to generate External IP for each service

```
ssh -i node-js.pem -L <local random port>:<External IP>:<port given  
to service> ec2-user@13.234.239.76
```

Example: `ssh -i node-js.pem -L 6560:10.102.190.88:8080 ec2-user@13.234.239.76`

❖ Generating Observability cluster

- For this case use Minikube Dashboard as the observer.

```
minikube dashboard
```

- Create a tunnel using ssh as above
- Open the URL in a browser that uses the Minikube dashboard.