# Task 1 - Minimalist Application Development / Docker / Kubernetes: -

### 🔧 Tech Stack

* Language: **Python (Flask)**
* Container: **Docker (Containerization)**
* Registry: **DockerHub (public)**
* Repository: **GitHub** (public)
* **App.py**
* from flask import Flask, request, jsonify
* from datetime import datetime
* app = Flask(\_\_name\_\_)
* @app.route('/', methods=['GET'])
* def index():
* return jsonify({
* "timestamp": datetime.utcnow().isoformat() + "Z",
* "ip": request.remote\_addr,
* "Partcle41\_Assessment":"Tiny App Development\_SimpleTimeService"
* })
* if \_\_name\_\_ == '\_\_main\_\_':
* app.run(host='0.0.0.0', port=5000)

# -> After creating python(app.py) application, we need to install docker on the server.

# -> sudo apt-get update && sudo apt-get install docker.io -y

# -> Create Dockerfile to containerize the application.

* **Dockerfile**

FROM python:3.12-slim

RUN useradd -m appuser

WORKDIR /app

COPY app.py requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt

RUN chown -R appuser:appuser /app

USER appuser

EXPOSE 5000

CMD ["python", "app.py"]

**-> docker build -t simpletimeservice .**

**-> docker ps**

**-> docker run -it -d -p 5000:5000 simpletimeservice:latest**

**-> docker images**

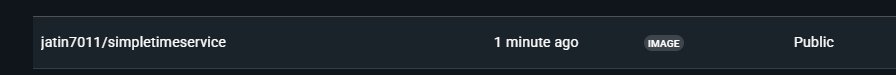
****

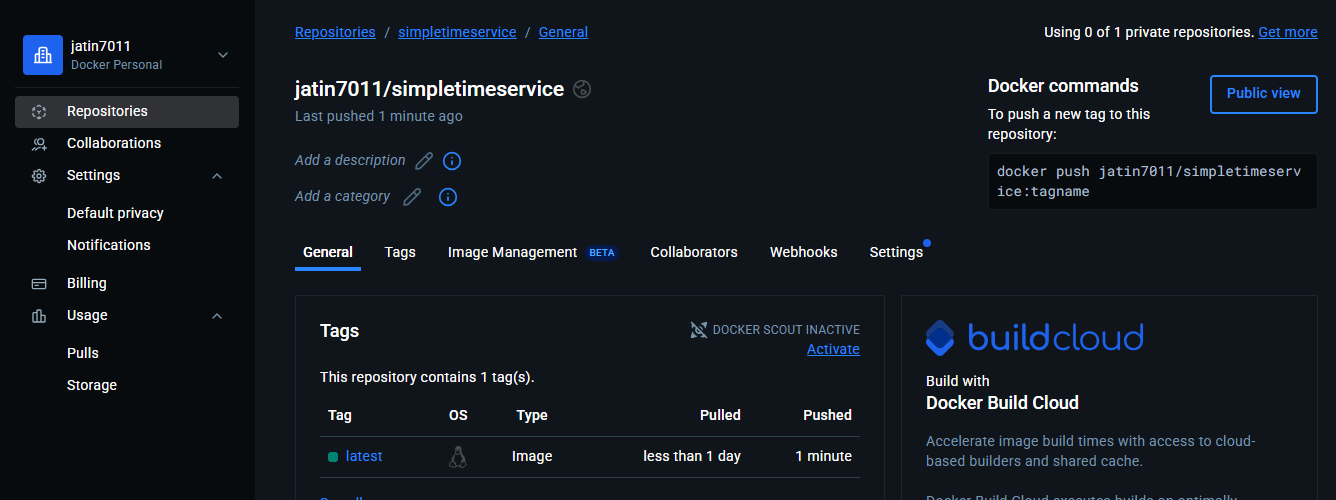
**To push into DockerHub, we need to login into DockerHub Account:**

**-> docker login**

**-> docker tag simpletimeservice:latest jatin7011/simpletimeservice:latest**

**-> docker push jatin7011/simpletimeservice:latest**





**-> After that, we need to initialize the git repo to push into public repo.**

**-> Create Public Repo inside GitHub to push the code.**

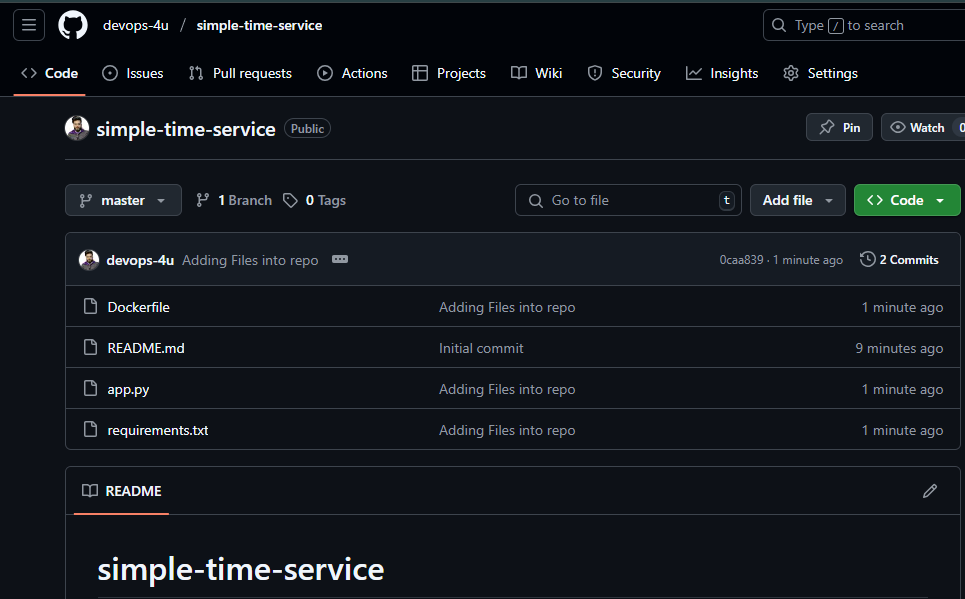
**-> git init**

**-> git status**

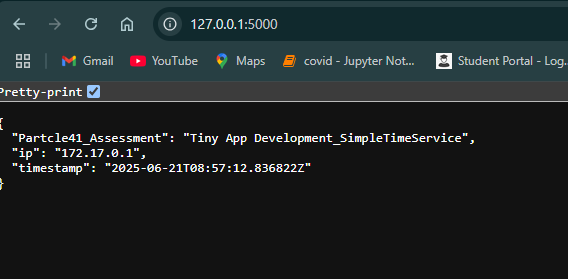
**-> git add .**

**-> git commit -m “Initial commit - SimpleTimeService”**

**-> git push origin master**



**OUTPUT**:-



# Task 2 - Terraform and Cloud: create the infrastructure to host your container.

* **main.tf**

terraform {

  required\_version = ">= 1.3.0"

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = ">= 5.43.0, < 6.0.0"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

module "vpc" {

  source  = "terraform-aws-modules/vpc/aws"

  version = "5.1.0"

  name = "eks-vpc"

  cidr = "10.0.0.0/16"

  azs  = ["${var.aws\_region}a", "${var.aws\_region}b"]

  public\_subnets  = ["10.0.1.0/24", "10.0.2.0/24"]

  private\_subnets = ["10.0.3.0/24", "10.0.4.0/24"]

  enable\_dns\_support   = true

  enable\_dns\_hostnames = true

  enable\_nat\_gateway   = true

  single\_nat\_gateway   = true

}

module "eks" {

  source  = "terraform-aws-modules/eks/aws"

  version = "20.8.4"

  cluster\_name    = var.cluster\_name

  cluster\_version = "1.29"

  vpc\_id          = module.vpc.vpc\_id

  subnet\_ids      = module.vpc.private\_subnets

  enable\_irsa = true

  eks\_managed\_node\_groups = {

    default = {

      desired\_capacity = 2

      min\_capacity     = 1

      max\_capacity     = 3

      instance\_types = ["t3.small"]

      subnet\_ids     = module.vpc.private\_subnets

    }

  }

}

* **Outputs.tf**

output "cluster\_name" {

  value = module.eks.cluster\_name

}

output "cluster\_endpoint" {

  value = module.eks.cluster\_endpoint

}

* **Variable.tf**

variable "aws\_region" {

  default = "ap-south-1"

}

variable "cluster\_name" {

  default = "simpletimeservice-eks"

}

* **Terraform.tfvars**

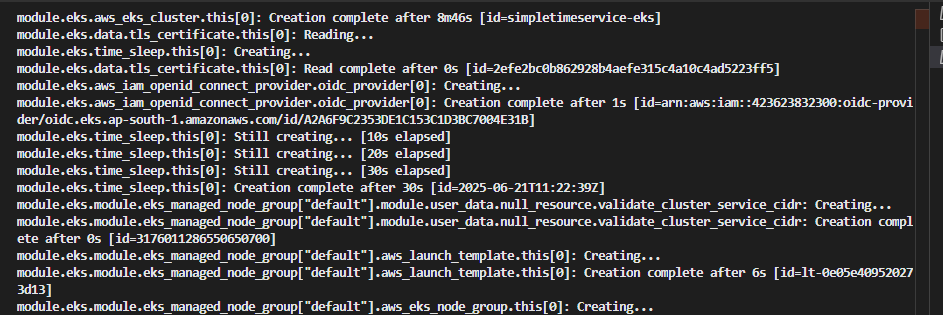
aws\_region    = "ap-south-1"

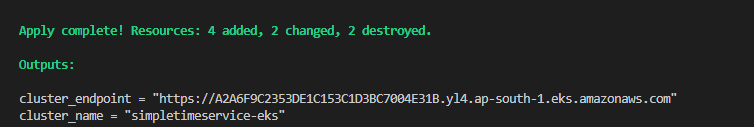
cluster\_name  = "simpletimeservice-eks"

**-> terraform init**

**-> terraform plan**

**-> terraform apply –auto-approve**





**Now, our infra is ready.. Now we have to create bastion host to make communication with our eks cluster because our eks cluster is in private subnet.**

**-> Create EC2 instance within same VPC as EKS with public subnet.**

**-> Attach IAM Role to Bastion {** AmazonEKSClusterPolicy, AmazonEKSWorkerNodePolicy, AmazonEC2ContainerRegistryReadOnly**}**

**-> Install CLI on new instance.**

**-> Connect to EKS with below command:-**

**aws eks update-kubeconfig --region ap-south-1 --name simpletimeservice-eks**

**-> Add an inbound rule for port 443 on EKS control plane from our bastion host**

**-> Now we are able to reach the EKS cluster endpoint but we are** **not authenticated.**

**-> To authenitacate the bastion host cli user we need to provide below policy under access:**

**eks-cluster-name -> access -> create access [CLI USER] -> Add both policy for user.**

**AmazonEKSAdminPolicy, AmazonEKSClusterAdminPolicy**

**-> aws eks update-kubeconfig --region ap-south-1 --name simpletimeservice-eks**

**-> kubectl get nodes**

**->** **sudo vi deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: simpletimeservice

spec:

  replicas: 2

  selector:

    matchLabels:

      app: simpletimeservice

  template:

    metadata:

      labels:

        app: simpletimeservice

    spec:

      containers:

      - name: simpletimeservice

        image: jatin7011/simpletimeservice:latest

        ports:

        - containerPort: 5000

-**> sudo vi service.yaml**

apiVersion: v1

kind: Service

metadata:

  name: simpletimeservice

spec:

  selector:

    app: simpletimeservice

  ports:

    - protocol: TCP

      port: 80

      targetPort: 5000

  type: LoadBalancer

**->kubectl apply -f deployment.yaml**

**-> kubectl apply -f service.yaml**

**-> kubectl get deployment, svc**

**-> nohup kubectl port-forward deployment/simpletimeservice 5000:5000 > /dev/null 2>&1 & -> curl http://localhost:5000**