# MultiCloud, DevOps & AI Challenge - Day 2

## Part 1 – Docker

### Step 1: Install Docker on EC2

* Execute the following commands:

sudo yum update -y

sudo yum install docker -y

sudo systemctl start docker

sudo docker run hello-world

sudo systemctl enable docker

docker --version

sudo usermod -a -G docker $(whoami)

newgrp docker

* + **sudo yum install docker -y** -> sudo → Runs the command as the superuser (administrator).
  + yum → The package manager used in RHEL-based Linux distributions.
  + update → Updates all system packages to their latest versions.
  + -y → Automatically confirms "yes" to all prompts.
  + **sudo yum install docker** - Installs Docker from the official package repository and the -y flag confirms installation without prompting the user.
  + **sudo systemctl start docker** - systemctl → A command to control system services.
  + start docker → Starts the Docker service.
  + **sudo docker run hello-world** -> docker run hello-world - Runs a simple test container to verify that Docker is working.
  + This pulls a small test image (hello-world) from Docker Hub and runs it.
  + If Docker is working, it will print a welcome message.
  + **sudo systemctl enable docker -** Ensures Docker starts automatically when the system boots up.
  + **docker –version -** Displays the installed Docker version.
  + **sudo usermod -a -G docker $(whoami) ->** usermod -a -G docker $(whoami) - usermod - Modify a user account.
  + -a - Append the user to a group.
  + -G docker - Add the user to the docker group.
  + $(whoami) - Gets the currently logged-in user.
  + This allows the user to run Docker commands without sudo.
  + **newgrp docker** -> Activates the new group membership without logging out.
  + If you skip this step, you might need to log out and log back in for the changes to take effect.

### Step 2: Create Docker image for CloudMart

**Backend**

* Create folder and download source code:
  + mkdir -p challenge-day2/backend && cd challenge-day2/backend - Creates a folder & Navigates into the backend directory.
  + wget <https://tcb-public-events.s3.amazonaws.com/mdac/resources/day2/cloudmart-backend.zip> - Command-line tool to download files from the internet.
  + unzip cloudmart-backend.zip - Extracts the contents of cloudmart-backend.zip into the current directory (backend/).
* Create .env file:
  + vi .env
  + Contents of .env:

PORT=5000

AWS\_REGION=us-east-1

BEDROCK\_AGENT\_ID=<your-bedrock-agent-id> BEDROCK\_AGENT\_ALIAS\_ID=<your-bedrock-agent-alias-id> OPENAI\_API\_KEY=<your-openai-api-key> OPENAI\_ASSISTANT\_ID=<your-openai-assistant-id>

* Create Dockerfile:
  + vi Dockerfile
  + Content of Dockerfile:

FROM node:18

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["npm", "start"]

**Frontend**

* Create folder and download source code:

cd ..

mkdir frontend && cd frontend

wget <https://tcb-public-events.s3.amazonaws.com/mdac/resources/day2/cloudmart-frontend.zip> unzip cloudmart-frontend.zip

* Create Dockerfile:
  + vi Dockerfile
  + Content of Dockerfile:

FROM node:16-alpine as build

WORKDIR /app

COPY package\*.json ./

RUN npm ci

COPY . .

RUN npm run build

FROM node:16-alpine

WORKDIR /app

RUN npm install -g serve

COPY --from=build /app/dist /app

ENV PORT=5001

ENV NODE\_ENV=production

EXPOSE 5001

CMD ["serve", "-s", ".", "-l", "5001"]

## Part 2 – Kubernetes

Attention: AWS Kubernetes service is not free, so when executing the hands-on below, you will be charged a few cents on your AWS account according to EKS pricing on AWS.

Remember to delete the cluster to avoid unwanted charges. Use the removal section at the end of the doc.

### Cluster Setup on AWS Elastic Kubernetes Services (EKS)

1. Create a user named eksuser with Admin privileges and authenticate with it

aws configure

1. Install the CLI tool eksctl

curl --silent --location [https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz](https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname%20-s)_amd64.tar.gz) | tar xz -C /tmp

sudo cp /tmp/eksctl /usr/bin

eksctl version

1. Install the CLI tool kubectl

curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl

chmod +x ./kubectl

mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin

echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc

kubectl version --short --client

1. Create an EKS Cluster

eksctl create cluster \

--name cloudmart \

--region us-east-1 \

--nodegroup-name standard-workers \

--node-type t3.medium \

--nodes 1 \

--with-oidc \

--managed

1. Connect to the EKS cluster using the kubectl configuration

aws eks update-kubeconfig --name cloudmart

1. Verify Cluster Connectivity

kubectl get svc

kubectl get nodes

1. Create a Role & Service Account to provide pods access to services used by the application (DynamoDB, Bedrock, etc).

eksctl create iamserviceaccount \

--cluster=cloudmart \

--name=cloudmart-pod-execution-role \

--role-name CloudMartPodExecutionRole \

--attach-policy-arn=arn:aws:iam::aws:policy/AdministratorAccess\

--region us-east-1 \

--approve

NOTE: In the example above, Admin privileges were used to facilitate educational purposes. Always remember to follow the principle of least privilege in production environments

### Backend Deployment on Kubernetes

* Create an ECR Repository for the Backend and upload the Docker image to it

Repository name: cloudmart-backend (do this step in AWS)

* Switch to backend folder

cd ../..

cd challenge-day2/backend

* Follow the ECR steps to build your Docker image( As shown in the ECR Repository)
* Create a Kubernetes deployment file (YAML) for the Backend

vi cloudmart-backend.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: cloudmart-backend-app

spec:

replicas: 1

selector:

matchLabels:

app: cloudmart-backend-app

template:

metadata:

labels:

app: cloudmart-backend-app

spec:

serviceAccountName: cloudmart-pod-execution-role

containers:

- name: cloudmart-backend-app

image: public.ecr.aws/l4c0j8h9/cloudmart-backend:latest

env:

- name: PORT

value: "5000"

- name: AWS\_REGION

value: "us-east-1"

- name: BEDROCK\_AGENT\_ID

value: "xxxxxx"

- name: BEDROCK\_AGENT\_ALIAS\_ID

value: "xxxx"

- name: OPENAI\_API\_KEY

value: "xxxxxx"

- name: OPENAI\_ASSISTANT\_ID

value: "xxxx"

---

apiVersion: v1

kind: Service

metadata:

name: cloudmart-backend-app-service

spec:

type: LoadBalancer

selector:

app: cloudmart-backend-app

ports:

- protocol: TCP

port: 5000

targetPort: 5000

* Deploy the Backend on Kubernetes

kubectl apply -f cloudmart-backend.yaml

* Monitor the status of objects being created and obtain the public IP generated for the API

kubectl get pods

kubectl get deployment

kubectl get service

### Frontend Deployment on Kubernetes

Preparation

* Change the Frontend's .env file to point to the API URL created within Kubernetes obtained by the kubectl get service command

cd ../challenge-day2/frontend

nano .env

Content of .env:

VITE\_API\_BASE\_URL=http://<your\_url\_kubernetes\_api>:5000/api

* Create an ECR Repository for the Frontend and upload the Docker image to it
  + Repository name: cloudmart-frontend (Do this step in AWS).
* Follow the ECR steps to build your Docker image ( As shown in the AWS ECR).
* Create a Kubernetes deployment file (YAML) for the Frontend

nano cloudmart-frontend.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: cloudmart-frontend-app

spec:

replicas: 1

selector:

matchLabels:

app: cloudmart-frontend-app

template:

metadata:

labels:

app: cloudmart-frontend-app

spec:

serviceAccountName: cloudmart-pod-execution-role

containers:

- name: cloudmart-frontend-app

image: public.ecr.aws/l4c0j8h9/cloudmart-frontend:latest

---

apiVersion: v1

kind: Service

metadata:

name: cloudmart-frontend-app-service

spec:

type: LoadBalancer

selector:

app: cloudmart-frontend-app

ports:

- protocol: TCP

port: 5001

targetPort: 5001

* Deploy the Frontend on Kubernetes

kubectl apply -f cloudmart-frontend.yaml

* Monitor the status of objects being created and obtain the public IP generated for the API

kubectl get pods

kubectl get deployment

kubectl get service

### Removal

* At the end of the hands-on, delete all resources:
* If you delete the cluster at the end of the exercise, you'll have to recreate it for the next days. So decide what makes more sense for you: delete the cluster and recreate it every day or keep it and pay for the time it's running. However, don't forget to delete it permanently at the end of the Challenge.

kubectl delete service cloudmart-frontend-app-service

kubectl delete deployment cloudmart-frontend-app

kubectl delete service cloudmart-backend-app-service

kubectl delete deployment cloudmart-backend-app

eksctl delete cluster --name cloudmart --region us-east-1