Coworking Space Service Extension

The Coworking Space Service is a set of APIs that enables users to request one-time tokens and administrators to authorize access to a coworking space. This service follows a microservice pattern and the APIs are split into distinct services that can be deployed and managed independently of one another.

For this project, you are a DevOps engineer who will be collaborating with a team that is building an API for business analysts. The API provides business analysts basic analytics data on user activity in the service. The application they provide you functions as expected locally and you are expected to help build a pipeline to deploy it in Kubernetes.

Getting Started

Dependencies

Local Environment

1. Python Environment - run Python 3.6+ applications and install Python dependencies via `pip`
2. Docker CLI - build and run Docker images locally
3. `kubectl` - run commands against a Kubernetes cluster
4. `helm` - apply Helm Charts to a Kubernetes cluster

Remote Resources

1. AWS CodeBuild - build Docker images remotely
2. AWS ECR - host Docker images
3. Kubernetes Environment with AWS EKS - run applications in k8s
4. AWS CloudWatch - monitor activity and logs in EKS
5. GitHub - pull and clone code

Setup

1. Configure a Database

Set up a Postgres database using a Helm Chart.

1. Set up Bitnami Repo

```
helm repo add <REPO_NAME> https://charts.bitnami.com/bitnami
```

2. Install PostgreSQL Helm Chart

```
helm install <SERVICE_NAME> <REPO_NAME>/postgresql
```

This should set up a Postgre deployment at `<SERVICE_NAME>-postgresql.default.svc.cluster.local` in your Kubernetes cluster. You can verify it by running `kubectl svc`

By default, it will create a username `postgres`. The password can be retrieved with the following command:

```
export POSTGRES_PASSWORD=$(kubectl get secret --namespace default <SERVICE_NAME>-
postgresql -o jsonpath="{.data.postgres-password}" | base64 -d)

echo $POSTGRES_PASSWORD
```

* The instructions are adapted from Bitnami's PostgreSQL Helm Chart.

3. Test Database Connection The database is accessible within the cluster. This means that when you will have some issues connecting to it via your local environment. You can either connect to a pod that has access to the cluster *or* connect remotely via `Port Forwarding`

- Connecting Via Port Forwarding

```
kubectl port-forward --namespace default svc/<SERVICE_NAME>-postgresql 5432:5432 &
    PGPASSWORD="$POSTGRES_PASSWORD" psql --host 127.0.0.1 -U postgres -d postgres -p 5432
```

- Connecting Via a Pod

```
kubectl exec -it <POD_NAME> bash
PGPASSWORD="<PASSWORD HERE>" psql postgres://postgres@<SERVICE_NAME>:5432/postgres -c
<COMMAND_HERE>
```

4. Run Seed Files We will need to run the seed files in `db/` in order to create the tables and populate them with data.

```
kubectl port-forward --namespace default svc/<SERVICE_NAME>-postgresql 5432:5432 &
    PGPASSWORD="$POSTGRES_PASSWORD" psql --host 127.0.0.1 -U postgres -d postgres -p 5432
< <FILE_NAME.sql>
```

2. Running the Analytics Application Locally

In the `analytics/` directory:

1. Install dependencies

```
apt update -y
apt install -y build-essential libpq-dev

# Dependencies are installed during build time in the container itself so we don't have OS
mismatch
pip install --upgrade pip setuptools wheel --trusted-host pypi.org --trusted-host
pypi.python.org --trusted-host=files.pythonhosted.org
pip install -r requirements.txt --trusted-host pypi.org --trusted-host pypi.python.org --
trusted-host=files.pythonhosted.org
```

2. Run the application (see below regarding environment variables)

```
<ENV_VARS> python app.py
```

There are multiple ways to set environment variables in a command. They can be set per session by running `export KEY=VAL` in the command line or they can be prepended into your command.

- `DB_USERNAME`
- `DB_PASSWORD`
- `DB_HOST` (defaults to `127.0.0.1`)
- `DB_PORT` (defaults to `5432`)
- `DB_NAME` (defaults to `postgres`)

If we set the environment variables by prepending them, it would look like the following:

```
DB_USERNAME=username_here DB_PASSWORD=password_here python app.py
```

The benefit here is that it's explicitly set. However, note that the `DB_PASSWORD` value is now recorded in the session's history in plaintext. There are several ways to work around this including setting environment variables in a file and sourcing them in a terminal session.

3. Verifying The Application

- Generate report for check-ins grouped by dates `curl <BASE_URL>/api/reports/daily_usage`
- Expected output should look like

```
{"2023-02-07":40,"2023-02-08":202,"2023-02-09":179,"2023-02-10":158,"2023-02-
11":146,"2023-02-12":176,"2023-02-13":196,"2023-02-14":142}
```

- Generate report for check-ins grouped by users `curl <BASE_URL>/api/reports/user_visits`
- Expected output should look like

```
{"1":{"joined_at":"2023-01-20 03:23:39.757813","visits":6},"2":{"joined_at":"2023-02-02
16:23:39.757830","visits":5},"3":{"joined_at":"2023-01-31
10:23:39.757836","visits":5},"4":{"joined_at":"2023-02-13
05:23:39.757840","visits":2},"5":{"joined_at":"2023-02-11
22:23:39.757844","visits":7},"6":{"joined_at":"2023-02-07 18:23:39.757848","visits":3}}
```

3. Building the Docker Image

The build process is automated using AWS CodeBuild. The `buildspec.yml` file contains the build instructions for the Docker image. The build process is triggered by any push to the `main` branch in a separated GitHub repository - `https://github.com/rednag/udacity-cde-ng-p3-app`.

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging into ECR
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --username
AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
  build:
    commands:
      - echo Starting build at `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$CODEBUILD_BUILD_NUMBER .
      - docker tag $IMAGE_REPO_NAME:$CODEBUILD_BUILD_NUMBER
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$CODEBUILD_BUIL
D_NUMBER
  post_build:
    commands:
      - echo Completed build at `date`
      - echo Pushing the Docker image...
      - docker push
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$CODEBUILD_BUIL
D_NUMBER
```

4. Deploying the Application

The application is deployed using Kubernetes. The deployment configuration is stored in the `deployment/` directory. The deployment configuration is split into

- configmap.yaml - contains the environment variables for the application
- coworking-service.yaml - contains the service configuration
- coworking.yaml - contains the deployment configuration
- postgres-service.yaml - contains the service configuration for the Postgres database
- postgres.yaml - contains the deployment configuration for the Postgres database
- pv.yaml - contains the persistent volume configuration
- pvc.yaml - contains the persistent volume claim configuration

The deployment can be applied using the following command:

```
kubectl apply -f deployment/
```

5. Monitoring the Application

The application logs can be monitored using AWS CloudWatch. The logs are stored in the `/aws/containerinsights/my-cluster/application` log group. The log group can be found in the AWS Console under CloudWatch.

6. Troubleshooting

If you encounter any issues, please refer to the following resources:

- Kubernetes Documentation
- Docker Documentation
- AWS Documentation

If you are unable to resolve the issue, please reach out to the DevOps team for assistance.

7. Additional Commands

In this section all relevant commands for the deployment are listed:

```
kubectl apply -f <FILE_NAME>
kubectl delete -f <FILE_NAME>
kubectl logs <POD_NAME>
kubectl describe <deployment|service> <POD_NAME>
kubectl get pods
kubectl get svc
kubectl cp <source_file> <destination_file>
kubectl port-forward svc/<SERVICE_NAME>-postgresql 5432:5432 &
kubectl exec --stdin --tty <POD_NAME> -- bash
kubectl get secret db-secret -o jsonpath="{.data.DB_PASSWORD}" | base64 -d
```

Project Submission

Deliverables

1. `Dockerfile`

```
FROM python:3.10-slim-buster
```

```
USER root

WORKDIR /src

COPY ./requirements.txt requirements.txt

# Dependencies required for psycopg2 (used for Postgres client)
RUN apt update -y && apt install -y build-essential libpq-dev

# Dependencies are installed during build time in the container itself so we don't have OS
mismatch
RUN pip install --upgrade pip setuptools wheel --trusted-host pypi.org --trusted-host
pypi.python.org --trusted-host=files.pythonhosted.org
RUN pip install -r requirements.txt --trusted-host pypi.org --trusted-host pypi.python.org
--trusted-host=files.pythonhosted.org

COPY . .

CMD python app.py
```

2. Screenshot of AWS CodeBuild pipeline *See section screenshots or directory `screenshots/` for the screenshots.*
3. Screenshot of AWS ECR repository for the application's repository *See section screenshots or directory `screenshots/` for the screenshots.*
4. Screenshot of `kubectl get svc` *See section screenshots or directory `screenshots/` for the screenshots.*
5. Screenshot of `kubectl get pods` *See section screenshots or directory `screenshots/` for the screenshots.*
6. Screenshot of `kubectl describe svc <DATABASE_SERVICE_NAME>` *See section screenshots or directory `screenshots/` for the screenshots.*
7. Screenshot of `kubectl describe deployment <SERVICE_NAME>` *See section screenshots or directory `screenshots/` for the screenshots.*
8. All Kubernetes config files used for deployment (ie YAML files)

`configmap.yaml` `coworking-service.yaml` `coworking.yaml` `postgres-service.yaml` `postgres.yaml` `pv.yaml` `pvc.yaml`

9. Screenshot of AWS CloudWatch logs for the application *See section screenshots or directory `screenshots/` for the screenshots.*

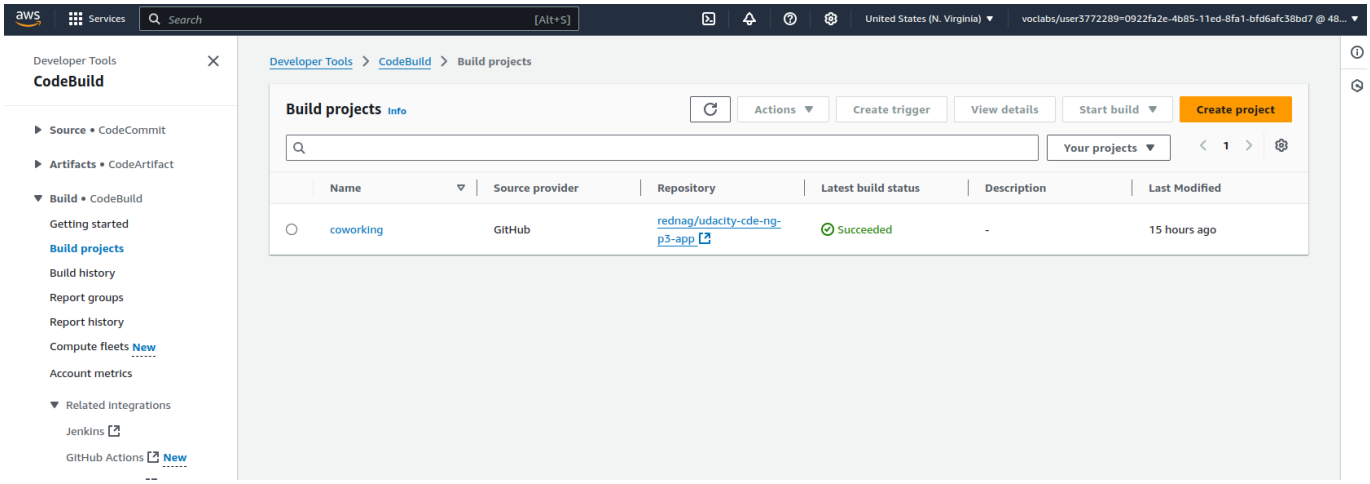
Stand Out Suggestions

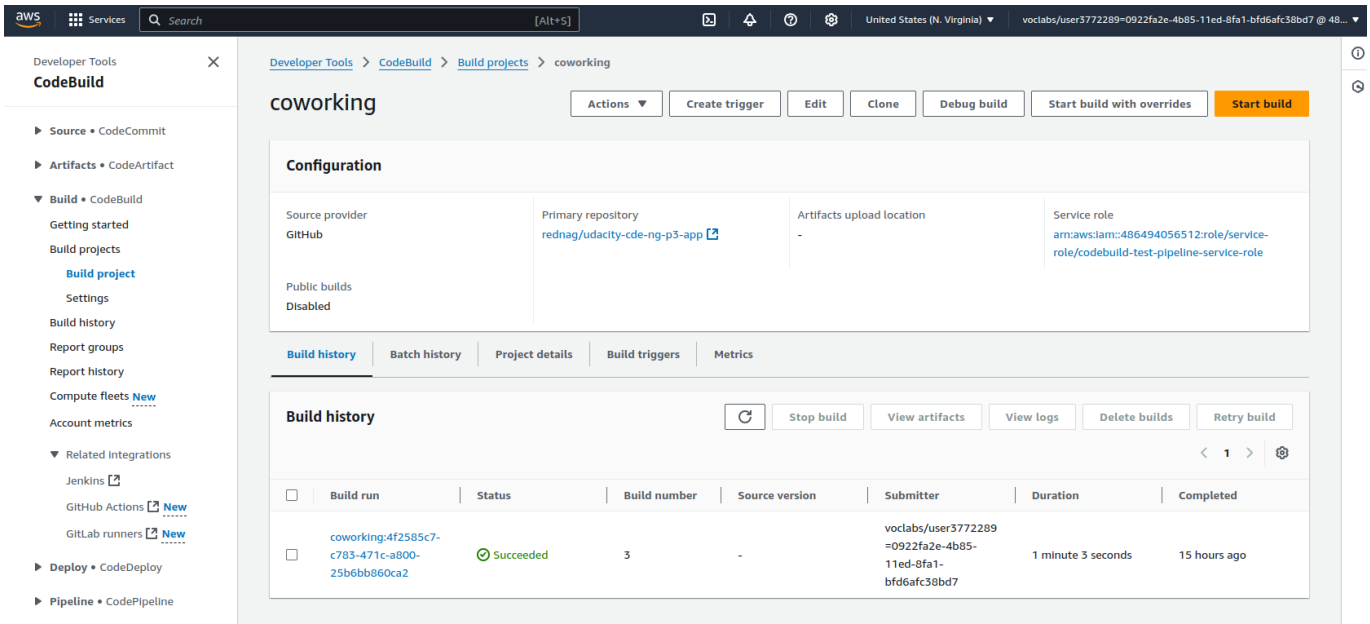
Please provide up to 3 sentences for each suggestion. Additional content in your submission from the standout suggestions do *not* impact the length of your total submission.

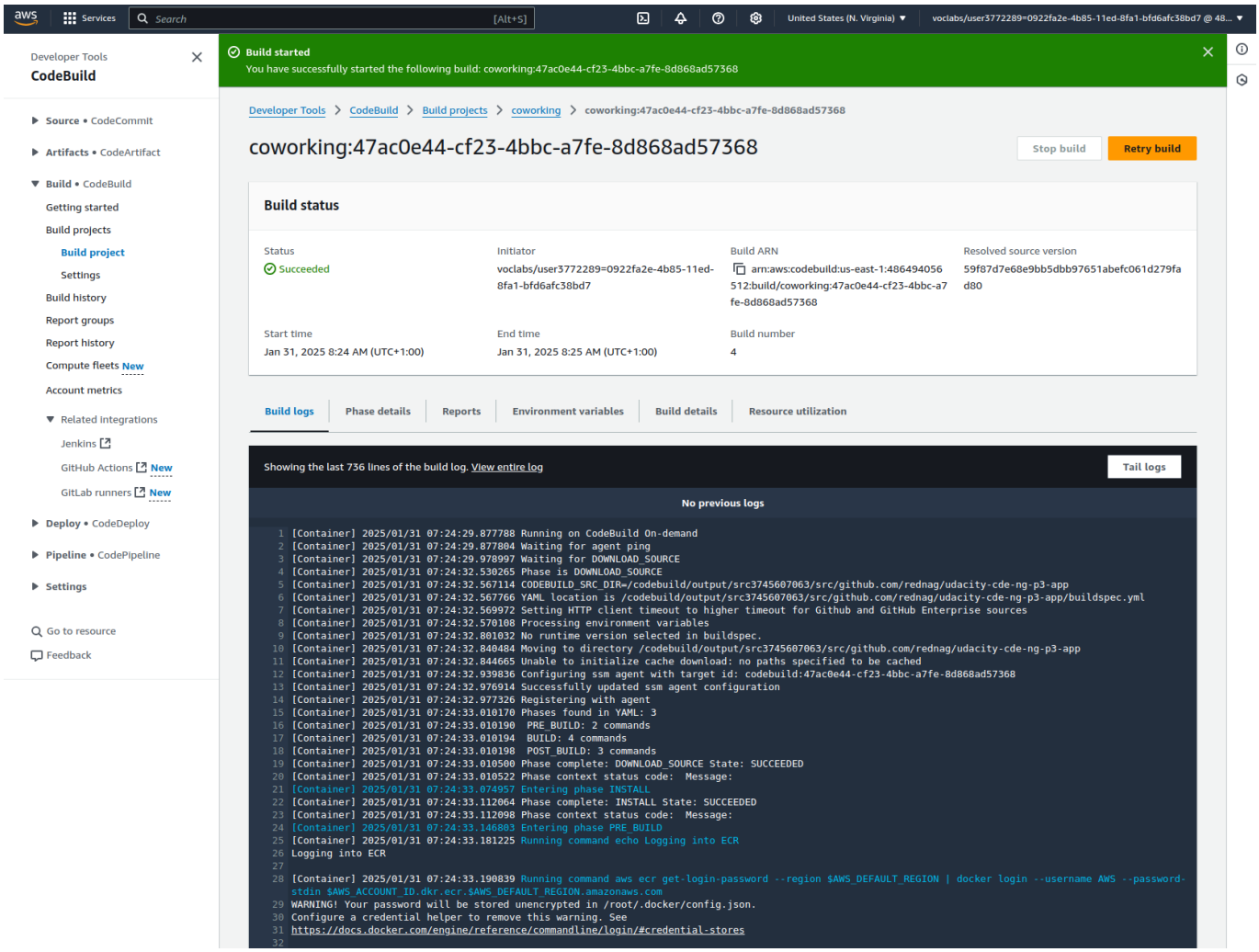
1. Specify reasonable Memory and CPU allocation in the Kubernetes deployment configuration *Since we are talking about a simple application with a few endpoints and no expected heavy load I'd run it on a pretty small resource and if needed it can be easily changed in the deployment file, same could be done for the datebase.*
2. In your README, specify what AWS instance type would be best used for the application? Why? *For the EKS I've chosen t3.small for such an application this is a sufficient instance type, it has 2 vCPUs and 2GB of memory, which is more than enough for the application and the database. If needed it can be easily scaled up.*
3. In your README, provide your thoughts on how we can save on costs? *Here we have already chosen a pretty small instance type, so the costs are already pretty low, but if we want to save even more we could use spot instances for the EKS cluster, which would save addtional costs. We could also think of choosing more or less nodes to be deployed in the EKS or maybe directly choose Fargate*

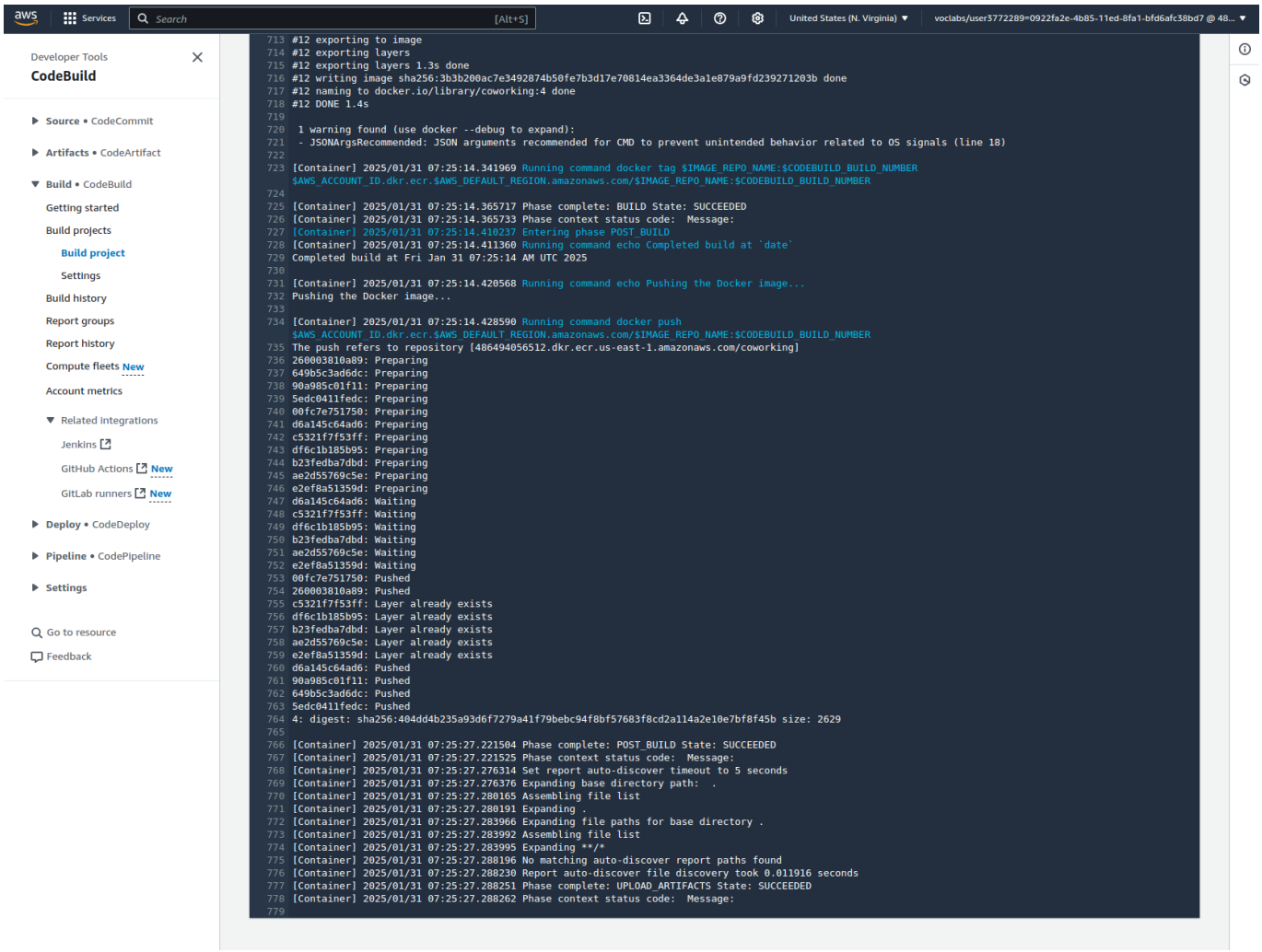
Screenshots

Screenshot of AWS CodeBuild pipeline

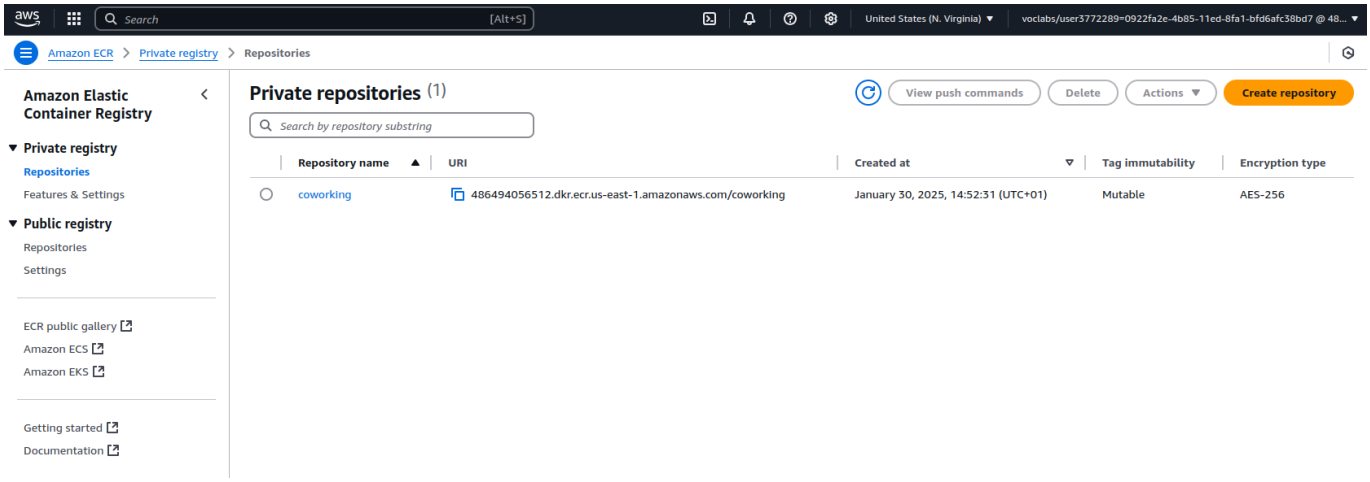


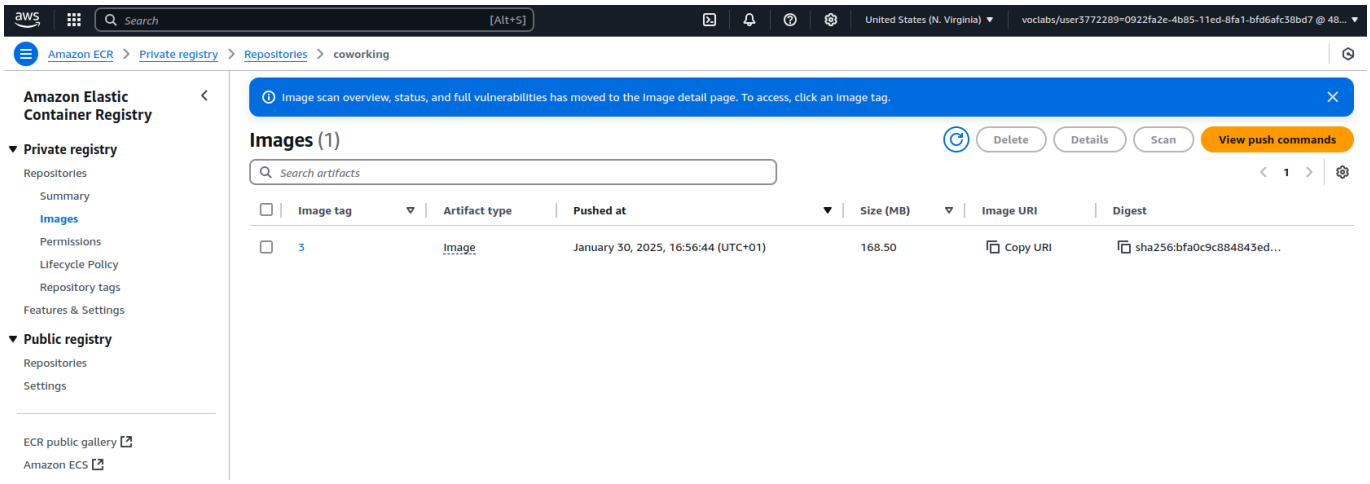



Services
Search
[Alt+S]
United States (N. Virginia)
Developer Tools
CodeBuild
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Getting started
Build projects
Build project
Settings
Build history
Report groups
Report history
Compute fleets New
Account metrics
Related integrations
Jenkins
GitHub Actions New
GitLab runners New
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Build started
You have successfully started the following build: coworking:47ac0e44-cf23-4bbc-a7fe-8d868ad57368
Developer Tools > CodeBuild > Build projects > coworking > coworking:47ac0e44-cf23-4bbc-a7fe-8d868ad57368
coworking:47ac0e44-cf23-4bbc-a7fe-8d868ad57368
Stop build
Retry build
Build status
Status
Succeeded
Initiator
voclabs/user3772289=0922fa2e-4b85-11ed-8fa1-bfd6afc38bd7
Build ARN
arn:aws:codebuild:us-east-1:486494056512:build/coworking:47ac0e44-cf23-4bbc-a7fe-8d868ad57368
Resolved source version
59f87d7e68e9bb5dbb97651abefc061d279fad80
Start time
Jan 31, 2025 8:24 AM (UTC+1:00)
End time
Jan 31, 2025 8:25 AM (UTC+1:00)
Build number
4
Build logs
Phase details
Reports
Environment variables
Build details
Resource utilization
Showing the last 736 lines of the build log. View entire log
Tail logs
No previous logs
[Container] 2025/01/31 07:24:29.877788 Running on CodeBuild On-demand
[Container] 2025/01/31 07:24:29.877804 Waiting for agent ping
[Container] 2025/01/31 07:24:29.978997 Waiting for DOWNLOAD_SOURCE
[Container] 2025/01/31 07:24:32.530265 Phase is DOWNLOAD_SOURCE
[Container] 2025/01/31 07:24:32.567114 CODEBUILD_SRC_DIR=/codebuild/output/src3745607063/src/github.com/rednag/udacity-cde-ng-p3-app
[Container] 2025/01/31 07:24:32.567766 YAML location is /codebuild/output/src3745607063/src/github.com/rednag/udacity-cde-ng-p3-app/buildspec.yml
[Container] 2025/01/31 07:24:32.569972 Setting HTTP client timeout to higher timeout for Github and GitHub Enterprise sources
[Container] 2025/01/31 07:24:32.570108 Processing environment variables
[Container] 2025/01/31 07:24:32.801032 No runtime version selected in buildspec.
[Container] 2025/01/31 07:24:32.840484 Moving to directory /codebuild/output/src3745607063/src/github.com/rednag/udacity-cde-ng-p3-app
[Container] 2025/01/31 07:24:32.844665 Unable to initialize cache download: no paths specified to be cached
[Container] 2025/01/31 07:24:32.939836 Configuring ssm agent with target id: codebuild:47ac0e44-cf23-4bbc-a7fe-8d868ad57368
[Container] 2025/01/31 07:24:32.976914 Successfully updated ssm agent configuration
[Container] 2025/01/31 07:24:32.977326 Registering with agent
[Container] 2025/01/31 07:24:33.010170 Phases found in YAML: 3
[Container] 2025/01/31 07:24:33.010190 PRE_BUILD: 2 commands
[Container] 2025/01/31 07:24:33.010194 BUILD: 4 commands
[Container] 2025/01/31 07:24:33.010198 POST_BUILD: 3 commands
[Container] 2025/01/31 07:24:33.010500 Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED
[Container] 2025/01/31 07:24:33.010522 Phase context status code: Message:
[Container] 2025/01/31 07:24:33.074957 Entering phase INSTALL
[Container] 2025/01/31 07:24:33.112064 Phase complete: INSTALL State: SUCCEEDED
[Container] 2025/01/31 07:24:33.112098 Phase context status code: Message:
[Container] 2025/01/31 07:24:33.146803 Entering phase PRE_BUILD
[Container] 2025/01/31 07:24:33.181225 Running command echo Logging into ECR
Logging into ECR
[Container] 2025/01/31 07:24:33.190839 Running command aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores



Screenshot of AWS ECR repository for the application's repository





Screenshot of `kubectl get svc`

```
→  deployment git:(main) ✗ kubectl get svc                                                                                    <aws:udacity3>
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP                                                                  PORT(S)          AGE
coworking           LoadBalancer   10.100.13.130   k8s-default-coworkin-c778bbf6b9-ace555c0c5fbdcc6.elb.us-east-1.amazonaws.com   5153:31751/TCP   14m
kubernetes          ClusterIP      10.100.0.1      <none>                                                                       443/TCP          11h
postgresql-service  NodePort       10.100.67.135   <none>                                                                       5432:30753/TCP   41m
→  deployment git:(main) ✗
```

Screenshot of `kubectl get pods`

```
→  deployment git:(main) ✗ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
coworking-57b8757b84-hz5vd    1/1     Running   0          4m6s
postgresql-5b7cbbcd8c-nwsrw   1/1     Running   0          51m
→  deployment git:(main) ✗
```

Screenshot of `kubectl describe svc <DATABASE_SERVICE_NAME>`

```
→  deployment git:(main) ✗ kubectl describe svc postgresql-service
Name:                     postgresql-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=postgresql
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.100.67.135
IPs:                      10.100.67.135
Port:                     <unset>  5432/TCP
TargetPort:               5432/TCP
NodePort:                 <unset>  30753/TCP
Endpoints:                192.168.70.237:5432
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
→  deployment git:(main) ✗
```

Screenshot of `kubectl describe deployment <SERVICE_NAME>`

```
➜  deployment git:(main) ✗ kubectl describe deployment postgresql
Name:                   postgresql
Namespace:              default
CreationTimestamp:      Fri, 31 Jan 2025 07:46:08 +0100
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=postgresql
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=postgresql
  Containers:
   postgresql:
    Image:      postgres:latest
    Port:       5432/TCP
    Host Port:  0/TCP
    Environment:
      POSTGRES_DB:        postgres
      POSTGRES_USER:      postgres
      POSTGRES_PASSWORD:  test123
    Mounts:
      /var/lib/postgresql/data from postgresql-storage (rw)
  Volumes:
   postgresql-storage:
    Type:          PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:     postgresql-pvc
    ReadOnly:      false
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   postgresql-5b7cbbcd8c (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  51m   deployment-controller  Scaled up replica set postgresql-5b7cbbcd8c from 0 to 1
➜  deployment git:(main) ✗
```

```
➜  deployment git:(main) ✗ kubectl describe deployment coworking
Name:                   coworking
Namespace:              default
CreationTimestamp:      Fri, 31 Jan 2025 08:33:18 +0100
Labels:                 name=coworking
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               service=coworking
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  service=coworking
  Containers:
   coworking:
    Image:      486494056512.dkr.ecr.us-east-1.amazonaws.com/coworking:3
    Port:       <none>
    Host Port:  <none>
    Liveness:   http-get http://:5153/health_check delay=5s timeout=2s period=10s #success=1 #failure=3
    Readiness:  http-get http://:5153/readiness_check delay=5s timeout=5s period=10s #success=1 #failure=3
    Environment Variables from:
      db-env  ConfigMap  Optional: false
    Environment:
      DB_USERNAME:  <set to the key 'DB_USER' of config map 'db-env'>     Optional: false
      DB_PASSWORD:  <set to the key 'DB_PASSWORD' in secret 'db-secret'>  Optional: false
    Mounts:         <none>
  Volumes:          <none>
  Node-Selectors:   <none>
  Tolerations:      <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   coworking-57b8757b84 (1/1 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  4m59s  deployment-controller  Scaled up replica set coworking-57b8757b84 from 0 to 1
➜  deployment git:(main) ✗
```

Screenshot of AWS CloudWatch logs for the application