



Course: IT485

Project: HawkinCloud

Members: Aliyev Omar, Haider Zaheer, Michael Hernandez, Nephtho Pierre.

REPORT #5

Management Guidance:

Our consultation management team creates project pipelines and roadmaps that are nearly 80% complete. We apply a structured approach to weekly work assignments, which we have successfully implemented since the first project. Each team member provides guidance and shares key points, enabling us to set up point-to-point integration and complete other tasks before moving on to the next stage of the blackboard model.

We maintain strong communication and collaboration through our Discord model, ensuring that any miscommunications are avoided. Clear documentation and models also help team members improve their knowledge and share it with others in the group. We're looking for someone who can help us solve a problem with managing the duration and contribution of team members. With this, we hope to make our management process even more efficient.

Team Member participant:

Each profile in the team members continuously attends and sees role model/responsibility as the time stand of the table. Each stands deploying customized to be done programming and use services in the project. Because some of the way to back we are highly looking to enable it and push programming developing different level running OS systems and user types. That assures us to integrate the Flask project to be capable of its maximum side of the availability of the work and expand features in the AWS services and container process to request less log activity and a high scalability process. Continuous Integration and Continuous development applying team members' content issues and testing analysis development to help it, any programmatic level various and unlimited further it and integrate much more models once demanding on the field.

Output Deliverables:**Project Concept**

Project building period and components add-ons each feature can be applying develop beginning to ending period. We are obtain based on the resources finish up Python environment to AWS and installing pip dependencies whenever we need it. This option also updates requirements.txt file to be migrate microservices compose up requesting exactly real time app configuration. Our concept moves into the above microservices and pipeline progress assets mini cluster.

Microservices

The process of creating a Dockerfile and releasing a repository is essential for the smooth operation of software projects. The Dockerfile and repository can be easily updated by customers, with limited modifications allowed using downloader credentials. This flexibility allows customers to customize the software to meet their specific needs while maintaining the integrity of the original software design.

In addition, a pre-installation process is required to maintain the progress of installing AWS CLI into a mini-cluster. This process enables customers to use programmatic advantages to create their own minikube clusters, such as Kubernetes clusters. The AWS CLI is a command-line interface tool that enables users to interact with various AWS services, including Amazon S3, Amazon EC2, and Amazon CloudWatch. By pre-installing the AWS CLI, customers can streamline the process of deploying and managing their software projects in the cloud. Another important aspect of software development is the inclusion of SQLAlchemy. When building an image for a project, it is essential to include a fresh SQLAlchemy database to ensure smooth operation of the software.

```

RUN pip install -r requirements.txt
COPY src src
COPY src database.db
COPY src/app.py /src/
RUN mkdir -p /src/templates
RUN mkdir -p /src/static
RUN mkdir -p /src/static/pics
COPY src/static/pics/logo__4-removebg.png /src/static/pics
COPY src/static/pics/logo__5-removebg.png /src/static/pics

COPY src/static/dashboard.css /src/static/
COPY src/static/signin.css /src/static/
COPY src/static/starter-template.css /src/static/

COPY src/templates/charts.html /src/templates/
COPY src/templates/dashboard.html /src/templates/
COPY src/templates/ec2.html /src/templates/
COPY src/templates/events_chart.html /src/templates/
COPY src/templates/index.html /src/templates/
COPY src/templates/kms_graph.html /src/templates/
COPY src/templates/login.html /src/templates/
COPY src/templates/sg_graph.html /src/templates/
COPY src/templates/signup.html /src/templates/

EXPOSE 4000
ENTRYPOINT ["python", "app.py"]

RUN echo "from app import db; db.create_all(); exit()" | flask shell
CMD ["sqlite3", "database.db", "select * from user; .exit"]

RUN apk add --no-cache curl \
&& curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip" \
&& unzip awscliv2.zip \
&& ./aws/install \
&& rm awscliv2.zip \
&& apk del curl

ENV AWS_ACCESS_KEY_ID=<your-access-key-id>
ENV AWS_SECRET_ACCESS_KEY=<your-secret-access-key>
ENV AWS_DEFAULT_REGION=<your-aws-region>

```











































o.alivev001@umb.edu

m.zaheer001@umb.edu

michael.hernandez003@umb.edu

nephthro.pierre001@umb.edu

Once the image is built, it can be signed up in the customer's environment, and REQUEST event files can be connected to the database to service the software and dashboard. This integration allows customers to manage their software projects with ease and efficiency. During the testing and response phase, microservices images are handled through the use of POST requests. The Dockerfile is used to handle the POST request and ensure the software is responding actively and using maintainable features. This phase is critical in ensuring that the software is functioning correctly and meeting the customer's needs. The creation of a Dockerfile and repository is essential for any software project. It provides customers with the flexibility to customize the software to meet their specific needs while maintaining the integrity of the original software design. The pre-installation process of the AWS CLI ensures that customers can create their own minikube clusters, which can streamline the process of deploying and managing software projects in the cloud. Finally, the inclusion of SQLAlchemy databases and the testing and response phase are critical in ensuring that the software is functioning correctly and meeting the customer's needs.

<input type="checkbox"/>	 vibrant_mcclintock 8595d2db8449 	-	Exited	4000:4000 	1 day ago
<input type="checkbox"/>	 keen_antonelli 24fae7d00d9f 	-	Exited	4000:4000 	1 day ago
<input type="checkbox"/>	 clever_bose 1d4dda0c8240 	-	Exited	4000:4000 	1 day ago
<input type="checkbox"/>	 elastic_zhukovsky f5821f16d056 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 quizzical_swartz 791f970759ea 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 strange_williamson8 d72ad9e2b9fb 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 hopeful_lamarr 8d58b3199459 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 objective_tesla 3098169a0565 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 suspicious_chandrasekhar 1906f74b73f3 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 magical_einstein 377de47d5624 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 peaceful_haibt e96f989c317b 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 suspicious_kalam 1099de5406d8 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 exciting_haibt 8483ead0233d 	-	Exited	4000:4000 	2 days ago
<input type="checkbox"/>	 charming_thompson f1f9b15bf9e1 	-	Exited	4000:4000 	2 days ago

CI/CD Pipeline

The purpose of implementing a CI/CD pipeline to AWS Elastic Beanstalk is to automate our build and deployment process using GitHub. This is a critical part of software development that enables faster response times and smoother delivery of Docker images to the production environment. Elastic Beanstalk is a stable platform as a service that deploys, manages, and scales applications, making it highly recommended.

Our pipeline method divides each CI pipeline run when code is pushed to the master branch of the GitHub repository. It then performs a Git clone of the repository and generates deployment packages before building. We configure the pipeline on the test pipeline in the GitHub environment as the tokens listed in the .yml file. Whenever a pipeline is pushed, it prints an error message or completion feedback in real-time.

The CD pipeline runs after every event type, and before the CI pipeline, it looks for a new Elastic Beanstalk application version in the environment variables. These two packages are included as the main contents of the package declaration in the steps.

```

65 lines (58 sloc) | 2.51 KB
1
2 name: CI-CD-Pipeline-to-AWS-ElasticBeanstalk
3 env:
4   EB_PACKAGE_S3_BUCKET_NAME : "hawkinccloud.flask-app"
5   EB_APPLICATION_NAME       : "Hawkinccloud"
6   EB_ENVIRONMENT_NAME       : "Hawkinccloud-env"
7   DEPLOY_PACKAGE_NAME       : "hawkinccloud-app-$(github.sha).zip"
8   AWS_REGION_NAME          : "us-east-1"
9
10 on:
11   push:
12     branches:
13       - master
14   jobs:
15     my_ci_pipeline:
16       runs-on: ubuntu-latest
17
18     steps:
19       - name: Git clone our repository
20         uses: actions/checkout@v1
21
22       - name: Create ZIP deployment package
23         run: zip -r $(env.DEPLOY_PACKAGE_NAME) ./. -x *.git*
24
25       - name: Configure my AWS Credentials
26         uses: aws-actions/configure-aws-credentials@v1
27         with:
28           aws-access-key-id : $( secrets.MY_AWS_ACCESS_KEY )
29           aws-secret-access-key: $( secrets.MY_AWS_SECRET_KEY )
30           aws-region       : $( env.AWS_REGION_NAME )
31
32       - name: Copy our Deployment package to S3 bucket
33         run: aws s3 cp $( env.DEPLOY_PACKAGE_NAME ) s3://$( env.EB_PACKAGE_S3_BUCKET_NAME )/
34
35       - name: Print nice message on completion of CI Pipeline
36         run: echo "CI Pipeline part finished successfully"
37
38     my_cd_pipeline:
39       runs-on: ubuntu-latest
40       needs : [my_ci_pipeline]
41
42     steps:
43       - name: Configure my AWS Credentials
44         uses: aws-actions/configure-aws-credentials@v1
45         with:
46           aws-access-key-id : $( secrets.MY_AWS_ACCESS_KEY )
47           aws-secret-access-key: $( secrets.MY_AWS_SECRET_KEY )
48           aws-region       : $( env.AWS_REGION_NAME )
49
50       - name: Create new ElasticBeanstalk Application Version
51         run: |
52           aws elasticbeanstalk create-application-version \
53             --application-name $( env.EB_APPLICATION_NAME ) \
54             --source-bundle S3Bucket=$( env.EB_PACKAGE_S3_BUCKET_NAME ),S3Key=$( env.DEPLOY_PACKAGE_NAME ) \
55             --version-label "Ver-$( github.sha )" \
56             --description "CommitSHA=$( github.sha )"
57
58       - name: Deploy our new Application Version
59         run: aws elasticbeanstalk update-environment --environment-name $( env.EB_ENVIRONMENT_NAME ) --version-label "Ver-$( github.sha )"
60
61       - name: Print nice message on completion of CD Pipeline
62         run: echo "CD Pipeline part finished successfully"
63
64
65

```

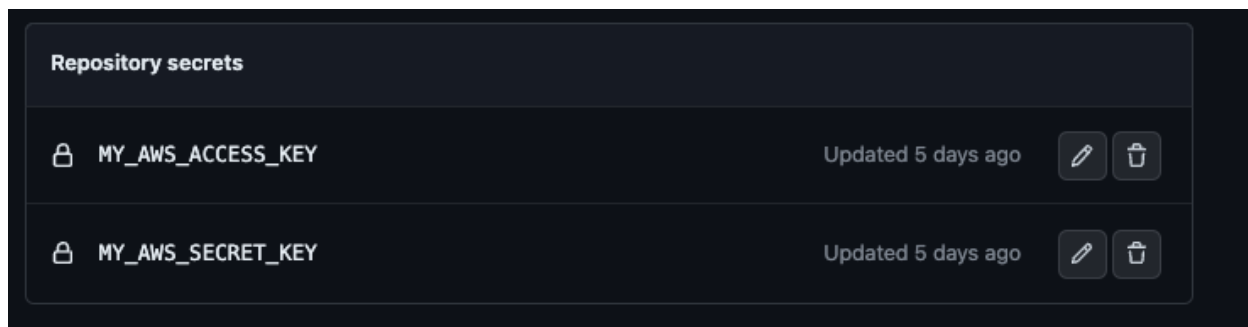
EB_PACKAGE_S3_BUCKET_NAME:	S3 bucket where the deployment package is stored.
EB_APPLICATION_NAME:	ElasticBeanstalk application.
EB_ENVIRONMENT_NAME:	ElasticBeanstalk environment.
DEPLOY_PACKAGE_NAME:	Deployment package.
AWS_REGION_NAME:	AWS region where the ElasticBeanstalk application is deployed.

GitHub Actions

Automating workflows is an essential step towards ensuring efficient and reliable software development processes. By automating tasks and defining dependencies using a YAML file, it is possible to build and test applications in a matter of minutes. This approach can also help to minimize errors that may arise during the build process, by automating the entire process.

Updating dependencies is a crucial aspect of software development that must be handled with care. Failure to do so could lead to compatibility issues and other problems that may impact the overall performance of the application. As such, it is important to trigger the workflow and test the building of docker images whenever new features are added to the system.

```
aws-access-key-id      : ${ secrets.MY_AWS_ACCESS_KEY }  
aws-secret-access-key: ${ secrets.MY_AWS_SECRET_KEY }  
aws-region             : ${ env.AWS_REGION_NAME }
```



Pushing the image into Github action is another essential step in the development process, as it may reveal any new optimization opportunities for our FLASK application. By continually optimizing the application, we can ensure that it performs at its best, delivering faster response times and increased efficiency. Moreover, automating workflows can also help to free up developer time, allowing them to focus on more critical tasks such as designing new features or fixing complex bugs. Automating workflows and understanding the benefits of this approach can help organizations to optimize their software development processes and deliver more efficient, reliable, and robust applications. By using tools like YAML files, Github actions, and Docker images, developers can build and test applications quickly and efficiently, while minimizing the risk of errors and other issues that may arise during the

build process.

