Employee Information App

Project Brief

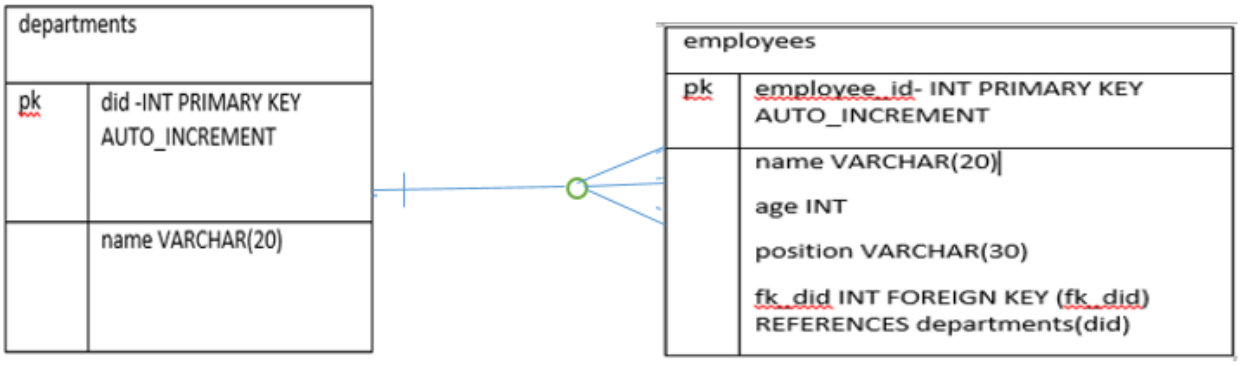
For this project I was given the brief to design, produce, and deploy a flask web app of my choosing.

The app must include integration with a MySQL database, and CRUD functionality. It must contain at least two tables sharing a one-to-many relationship. The structure of this app is shown in the diagram below.

The app must be hosted and deployed using containers, and a CI/CD pipeline must be used to automatically test, build, and deploy the application. (Webhook)

App Design

I have built a python flask app for use in an employee information. Current departments can be added in to the database which allows users to CRUD employee id, name, age and position can be added to the database. Both departments and employees can be edited (update functionality), and removed from the database (delete functionality).



Pipeline

This project requires the integration of a CI pipeline, including version control, virtual environment, containers, and a build server.

Git was used as version control, with the project repository hosted on Github. This allows changes to be made and committed, while keeping the commit history for other versions. I began by committing a working flask app to the main branch. Changes used for tests and containerization were implemented

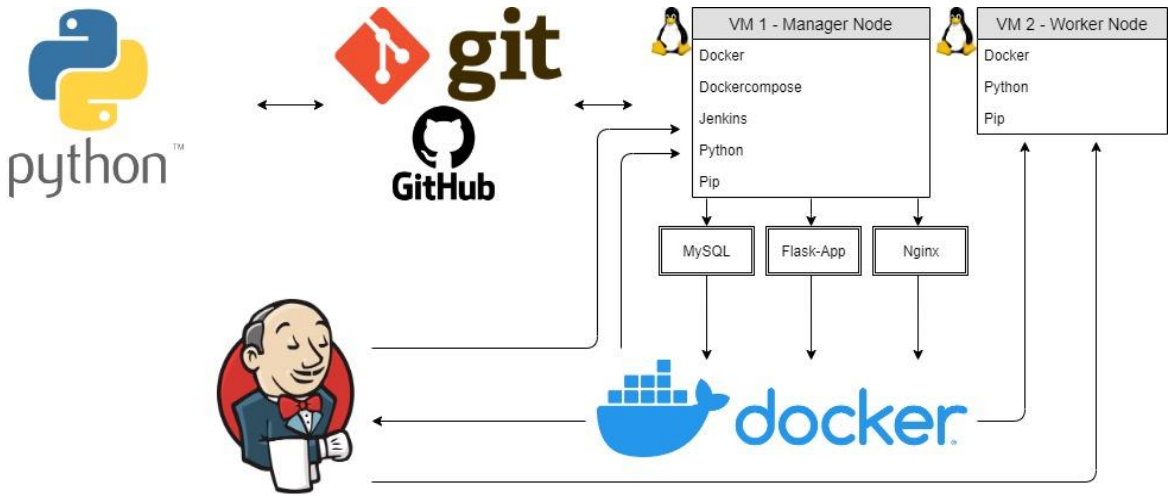
on separate branches. A .gitignore file is used to prevent the upload of the created database, and unnecessary information.

The application was originally created in Visual Studio Code, using a virtual environment. The develop environment was created using a python3 virtual environment hosted on a virtual machine using Ubuntu 20.04. This allows pip installs, and ensures that running the app will not conflict with any other pip installs on the same machine.

Docker containers were used in this project to create 3 packages: the flask application, the MySQL database, and Nginx. Deploying these 3 containers hosts the application on the local public IP, connects it to the database, and allows access via a reverse proxy. These were uploaded to Dockerhub for use with the Dockercompose and Dockerswarm. A docker compose file was used to define and run these containers using a single command. This was further expanded by using docker swarm deployment. Using two virtual machines (master and worker nodes), I am able to run the containers as a service, and deploy them across two machines. I can access my app via the public IP addresses of the machines.

Jenkins was used as a CI server. It was connected to the git repository, and automatically tested, built, and deployed the application using the docker compose file. It connected to a webhook, which triggered an automatic Jenkins build when a change is made to the code.

This pipeline is shown in the diagram below.



Requirements

For this project, I used two virtual machines, one Master, and one Worker. The Master requires: Docker, Dockercompose, Jenkins, Python, Pip. The Worker requires: Docker, Python, Pip. Port 80 must be open on both machines to access the application.

Description	Evaluation	Likelihood	Impact	Responsibility	Response	Control Measures
Webhook not triggering - 403 response	403 - permissions issue	2	1	Developer	Added Jenkins API to Github webhook	Add API to new Jenkins builds
Bug in code uploaded to main	Broken route, webpage did not load	3	4	Developer	Pull previous build, work on develop branch	Do not upload code to main branch without testing
App not displaying on Worker	Port 80 closed	4	2	Developer	Open port 80	Ensure it is in the documentation
SQL injection	Data breach	2	5	Developer	Follow data protection laws	Prepared statements, stored procedures
Jenkins VM goes down	CI functionality lost	2	3	Developer	Pull build and requirements onto new VM, connect to Workers	Keep repos up to date, list of requirements
Worker VM goes down	Swarm functionality lost	2	2	Developer	Pull build and requirements onto new VM, connect to Master	Keep repos up to date, list of requirements
GitHub goes down	Can't update code	1	3	GitHub	Use competitor service, eg BitBucket, update Jenkins	Keep local copy of up to date code
DockerHub goes down	Can't update containers	1	4	DockerHub	Run the app locally	Keep local copy of containers
Jenkins goes down	Can't deploy pipeline	1	3	Jenkins	Run the app locally	Keep local copy of up to date code
MySQL instance goes down	Can't interact with database	1	4	MySQL	Back up database regularly	Use new database

Likelihood		Impact	
1	Very unlikely	1	Minimal
2	Unlikely	2	Low
3	Moderate	3	Moderate
4	Likely	4	High
5	Very Likely	5	Catastrophic

Testing

Testing is required for the project. In this case, unit testing using pycharm was implemented. This tests the functionality of the app. Unit tests were written for the create data, read, update, and delete functionality. These tests feed in a test department, and a test employees. Using this information, we are testing that the routes of my app return a 200 response.

The tests I have written currently have 54% coverage, as they encounter TypeError . This has been added to my list of work for the future. The coverage is shown below, with test_.py only reaching 59% coverage. The tests have not been integrated into the Jenkins pipeline.

```
---------- coverage: platform linux, python 3.8.10-final-0 -----------
Name                                                             Stmts   Miss  Cover
------------------------------------------------------------------------------
/usr/lib/python3/dist-packages/blinker/__init__.py                   3      0   100%
/usr/lib/python3/dist-packages/blinker/_saferef.py                  83     39    53%
/usr/lib/python3/dist-packages/blinker/_utilities.py               100     53    47%
/usr/lib/python3/dist-packages/blinker/base.py                     162     71    56%
application/__init__.py                                              8      0   100%
application/forms.py                                                12      0   100%
application/models.py                                               11      0   100%
application/routes.py                                               70     46    34%
application/tests/test_app.py                                       76     31    59%
------------------------------------------------------------------------------
TOTAL                                                              525    240    54%
```

Future Work

My first priority is creating functioning tests. This will ensure that my application is functional before deploying it from Jenkins.

Currently, I am using two virtual machine. The master node is also hosting Jenkins. In the future I will run Jenkins on a third separate machine.

I will implement environment variables to ensure that sensitive information is not uploaded to GitHub. In future I would like to implement Login credentials and Role name and description