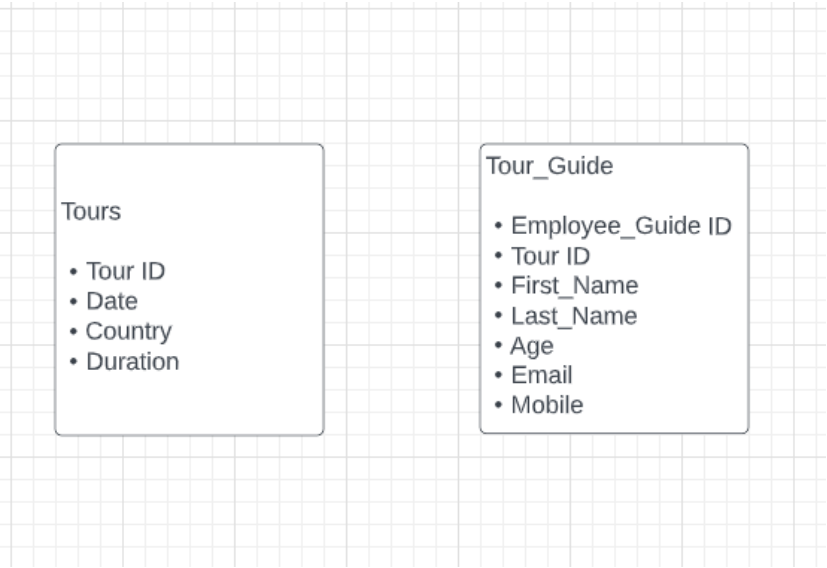
DFE Final Project

Created a web application based database via a linux based virtual machine. The create a pipeline using Jenkins and Github.

ER Diagram:



The ER disgrace shows a one-to- many relation as there is one tour guide per tour however a tour guide can have many tours allocated to them.

DataBase:

tour_app.py | SQLTools Settings | mydata.db

```python
from flask import Flask,render_template, redirect, url_for, request
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///mydata.db'
db = SQLAlchemy(app)

class TourGuide(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    first_name = db.Column(db.String)
    last_name = db.Column(db.String)
    age = db.Column(db.Integer)
    email = db.Column(db.String)
    mobile = db.Column(db.String)
    tour = db.relationship('Tour', backref='all_tour')


class Tour(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    date = db.Column(db.String)
    duration = db.Column(db.String)
    city = db.Column(db.String)
    country = db.Column(db.String)
    tourguide = db.Column(db.ForeignKey('tour_guide.tourguide_id'))


db.create_all()
g1 = TourGuide(id=1, first_name='Jesscia', last_name= 'Sanchez', age= 25, email= 'jessica@eurotours.com', mobile='07963784932')
g2 = TourGuide(id=2, first_name='Abby', last_name= 'Longhart', age= 30, email= 'abby@eurotours.com', mobile='07986536374')
g3 = TourGuide(id=3, first_name='Jamie', last_name= 'Bennett', age= 34, email= 'jamie@eurotours.com', mobile='07738564783')
g4 = TourGuide(id=4, first_name='Tomas', last_name= 'Smith', age= 28, email= 'tomas@eurotours.com', mobile='07456819282')
db.session.add(g1)
db.session.add(g2)
db.session.add(g3)
db.session.add(g4)
db.session.commit()
t1 = Tour(id=1, date='12-10-2022', duration='5 days', city='Spain', country='Barcelona', tourguide= 1)
t2 = Tour(id=2, date='01-12-2022', duration='7 days', city='Budapest', country='Hungary', tourguide= 2)
t3 = Tour(id=3, date='24-03-2023', duration='14 days', city='Geneva', country='Switzerland', tourguide= 3)
t4 = Tour(id=4, date='20-05-2023', duration='3 days', city='Berlin', country='Germany', tourguide= 4)
t5 = Tour(id=5, date='20-05-2023', duration='2 days', city='Paris', country='France', tourguide= 5)
t6 = Tour(id=6, date='15-07-2023', duration='8 days', city='Venice', country='Italy', tourguide= 6)
db.session.add(t1)
db.session.add(t2)
db.session.add(t3)
db.session.add(t4)
db.session.add(t5)
db.session.add(t6)
db.session.commit()
```

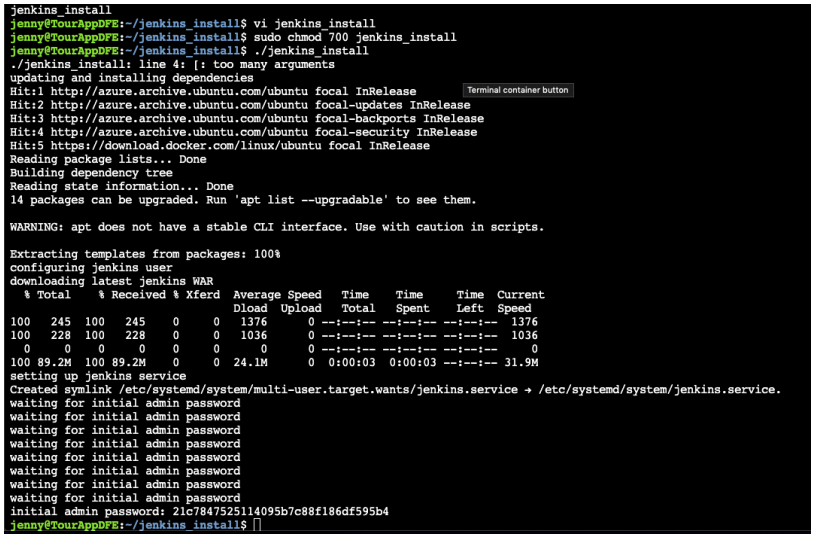
I created a database in visual code studio using my sqlite. I unfortunately wasn't able to connect the database to my virtual machine.

This database is suppose to produce two tables one for the tours where the foreign key is the tour-guide and another table for the tour guide. I have also created two separate classes with variables that link them together.

I then created a simple database using the the code I have presented a print screen of on the left.

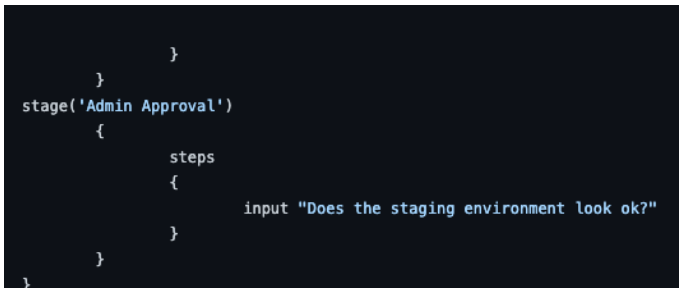
Pipeline:

I created a Jenkins pipeline via a Linux virtual machine. Below is a screen shot of the installation process.

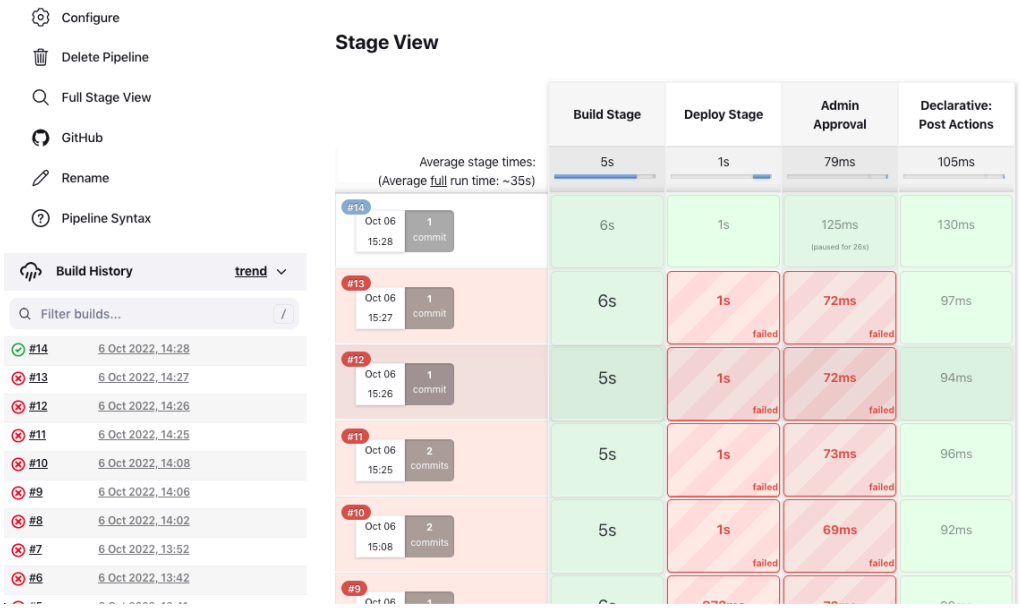


In this code we have established that this is the build stage of the pipeline. The echo command simply prints out ‘This is Build part’ when building in Jenkins. The sh ‘chmod 777 build.sh’ allows full access permissions to the file built.sh and the last command sh’./build.sh’ runs the file. The build.sh file contains a docker build command which builds Docker images from a Dockerfile which contains the App.py file.

Very Similar to the build stage we have the Deploy stage of the pipeline. The commands are also very similar and executed the same way except this time we have a run.sh file instead and this contains a docker run command dictating which port number the app will run on.



The final step of the pipeline is simply approving the completion of the pipeline.



This screen shot illustrates a working Jenkins Pipeline