1. Create a new directory with name testlabs, Initialize as git repo

* 1. use git config parameters username: testlab and email: testmail@ow.com
  2. create a new file test.sh, perform git add and commit
  3. create a new repo on github, named mynewlab and map to testlabs
  4. push your code to github.
  5. Ans: Refer lab0 and lab 1
  6. 2.
  7. Above created lab modify the repo from github and add a new file test1.sh Modify the file test.sh from your local git repos (testlabs)
  8. Perform git add and commit.
  9. Try to push the code, it will give rebase error
  10. Fix the issue using git rebase command (Don’t use the git clone command)
  11. <https://git-scm.com/docs/git-rebase>
  12. 3. <https://git-scm.com/docs/git-stash>
      1. git add .
      2. git status
      3. git stash save "add work"
      4. git status
      5. git stash list
      6. git stash apply 0
      7. git status
      8. 4.
      10. Create Jenkins cicd pipeline job git@github.com:amitopenwriteup/java11-examples.git Stage: checkout
      11. Stage: maven package buiding
      12. Stage: arhcieve artifacts
      13. <https://www.openwriteup.com/?page_id=1135>
  13. 5)dockerfile creation
  14. - create repo
  15. - Create docker file
  16. <https://github.com/amitopenwriteup/cicd/blob/master/dockerfile>
  17. Write the Jenkins file
  18. <https://github.com/amitopenwriteup/cicd/blob/master/jenkinsfile>
  19. 6)Modify the above cicd pipeline and add one more stage to create deployment in kubernetes cluster
  20. <https://github.com/amitopenwriteup/cicd/blob/master/jenkinfilekub>
  21. 7)
  22. Perform this labs and deploy the eks cluster
  23. <https://developer.hashicorp.com/terraform/tutorials/kubernetes/eks>

1. 8) Create a dynamic website using docker
2. To create a dynamic website using Docker, you’ll want to follow a setup that includes:
3. 1. \*\*Backend server\*\* (for processing logic, e.g., Node.js, Python with Flask/Django, PHP, etc.)
4. 2. \*\*Database\*\* (e.g., MySQL, PostgreSQL)
5. 3. \*\*Frontend\*\* (HTML/CSS/JavaScript or a framework like React, Angular, or Vue)
6. Here's an example of how to create a basic dynamic website using \*\*Node.js\*\* for the backend and \*\*MySQL\*\* for the database, containerized with Docker.
7. ### 1. Project Structure
8. Create a directory structure like this:
9. ```plaintext
10. dynamic-website/
11. ├── backend/
12. │ ├── Dockerfile
13. │ ├── app.js
14. │ └── package.json
15. ├── frontend/
16. │ ├── Dockerfile
17. │ └── index.html
18. └── docker-compose.yml
19. ```
20. ### 2. Backend Setup (Node.js)
21. #### backend/package.json
22. ```json
23. {
24. "name": "backend",
25. "version": "1.0.0",
26. "main": "app.js",
27. "dependencies": {
28. "express": "^4.17.1",
29. "mysql": "^2.18.1"
30. }
31. }
32. ```
33. #### backend/app.js
34. ```javascript
35. const express = require('express');
36. const mysql = require('mysql');
37. const app = express();
38. const PORT = 3000;
39. // MySQL Connection
40. const db = mysql.createConnection({
41. host: 'mysql-db',
42. user: 'root',
43. password: 'example',
44. database: 'website\_db'
45. });
46. db.connect(err => {
47. if (err) throw err;
48. console.log('Connected to MySQL database');
49. });
50. // Routes
51. app.get('/', (req, res) => {
52. res.send('Welcome to the Dynamic Website!');
53. });
54. app.get('/data', (req, res) => {
55. db.query('SELECT \* FROM users', (err, results) => {
56. if (err) throw err;
57. res.json(results);
58. });
59. });
60. app.listen(PORT, () => {
61. console.log(`Server is running on port ${PORT}`);
62. });
63. ```
64. #### backend/Dockerfile
65. ```Dockerfile
66. # Use Node.js LTS image
67. FROM node:18
68. # Create app directory
69. WORKDIR /usr/src/app
70. # Install app dependencies
71. COPY package.json ./
72. RUN npm install
73. # Bundle app source
74. COPY . .
75. # Expose the port and start the app
76. EXPOSE 3000
77. CMD ["node", "app.js"]
78. ```
79. ### 3. Frontend Setup
80. #### frontend/index.html
81. ```html
82. <!DOCTYPE html>
83. <html lang="en">
84. <head>
85. <meta charset="UTF-8">
86. <title>Dynamic Website</title>
87. </head>
88. <body>
89. <h1>Dynamic Website with Docker</h1>
90. <div id="data"></div>
91. <script>
92. fetch('http://localhost:3000/data')
93. .then(response => response.json())
94. .then(data => {
95. const dataDiv = document.getElementById('data');
96. dataDiv.innerHTML = '<h2>User Data:</h2>' + JSON.stringify(data);
97. });
98. </script>
99. </body>
100. </html>
101. ```
102. #### frontend/Dockerfile
103. ```Dockerfile
104. # Use Nginx to serve static files
105. FROM nginx:alpine
106. # Copy static files to Nginx server directory
107. COPY index.html /usr/share/nginx/html/index.html
108. ```
109. ### 4. Docker Compose Setup
110. Create a `docker-compose.yml` file to define the multi-container setup.
111. #### docker-compose.yml
112. ```yaml
113. version: '3'
114. services:
115. frontend:
116. build: ./frontend
117. ports:
118. - "80:80" # Map port 80 to frontend
119. backend:
120. build: ./backend
121. ports:
122. - "3000:3000" # Map port 3000 to backend
123. depends\_on:
124. - mysql-db
125. mysql-db:
126. image: mysql:5.7
127. environment:
128. MYSQL\_ROOT\_PASSWORD: example
129. MYSQL\_DATABASE: website\_db
130. ports:
131. - "3306:3306" # Map port for MySQL
132. volumes:
133. - mysql-data:/var/lib/mysql
134. volumes:
135. mysql-data:
136. ```
137. ### 5. Running the Project
138. To start the website:
139. 1. Navigate to your project directory (`dynamic-website`).
140. 2. Run Docker Compose:
141. ```bash
142. docker-compose up --build
143. ```
144. This command will build and start the services:
145. - \*\*Frontend\*\* will be accessible at `http://localhost`.
146. - \*\*Backend\*\* will be accessible at `http://localhost:3000`.
147. - \*\*MySQL database\*\* will be available at `localhost:3306` (only accessible to containers).
148. ### 6. Testing the Setup
149. 1. Access `http://localhost` to view the static HTML page.
150. 2. Visit `http://localhost:3000/data` to see data fetched from the MySQL database.
151. This is a basic setup. You can extend it by adding more routes, implementing a frontend framework, or adding more complex data operations.