Version of docker-compose not specified on top of yaml file.

- No way the nodeapp is even trying to talk to database, need environmental variables.
- Networking – one is backend and one is frontend, they are isolated
- Nodeapp should depend on DB ideally – do we need it ?
- Db ports should be exposed, nodeports should be published as Ports – as this is the one that gets accessed from outside of the services (on the host). Note- in recnt version expose doesn't have any impact, mostly readblity
- MySQL volumes not specified correct format (- data-volume:/var/lib/mysql)
- nodeApp cannot contain capslock (name)

Firstly, We are told that the nodeApp is not yet developed, checking the server.js structure, we see that no connections are being made from the nodeapp to the DB. We are going to assume that this is going to be fixed by the developers in the future, and we will continue with the rest of the setup for now.

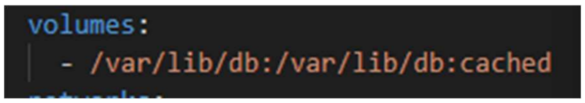
Firstly, observing the errors in the .yaml file –

1. The version of docker-compose to be used is not mentioned on the top. We would need to specify the version, depending on which some of the syntaxes can vary.

2. Next, the service name "nodeApp" cannot contain a UPPERCASE character, rename it to "nodeapp"

3. Since the nodeapp on startup will be trying to talk to a DB, we will have to make sure DB container is started first. We can do this by adding "depends_on" for the nodeapp container.

4. Now the SQL server persistent volume mount/bind – It should just be an array format with name:value pairs.



```
volumes:
  - /var/lib/db:/var/lib/db:cached
```

Note - This would make a bind mount, with the actual database files stored in the host, mapped onto the containers – with caching enabled on the container for better performance.

5. Assuming that the server.js is all setup and working correctly, we need to enable the request from nodeapp container to DB container to go through. Lets deal with networking-

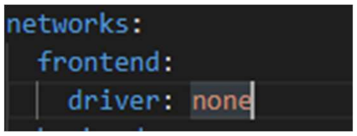
a. setup environmental variables in nodeApp container to access the DB. Example – DB_HOST, DB_USER, DB_PASSWORD, etc.

b. next, both our containers are put in different networks – isolating them. We can fix this multiple ways – Simplest one being to put them on the same network. (A little more complex solution would be to make a new *container that can access both the networks and relay information between the networks). For this example, Ill put both services on the "frontend" network. Further improvements in next step.*

c. The DB ports need not be exposed outside of the network to the host generally as this is a security risk. Instead, the nodeapp listening port(8080) must be exposed with the keyword "ports".

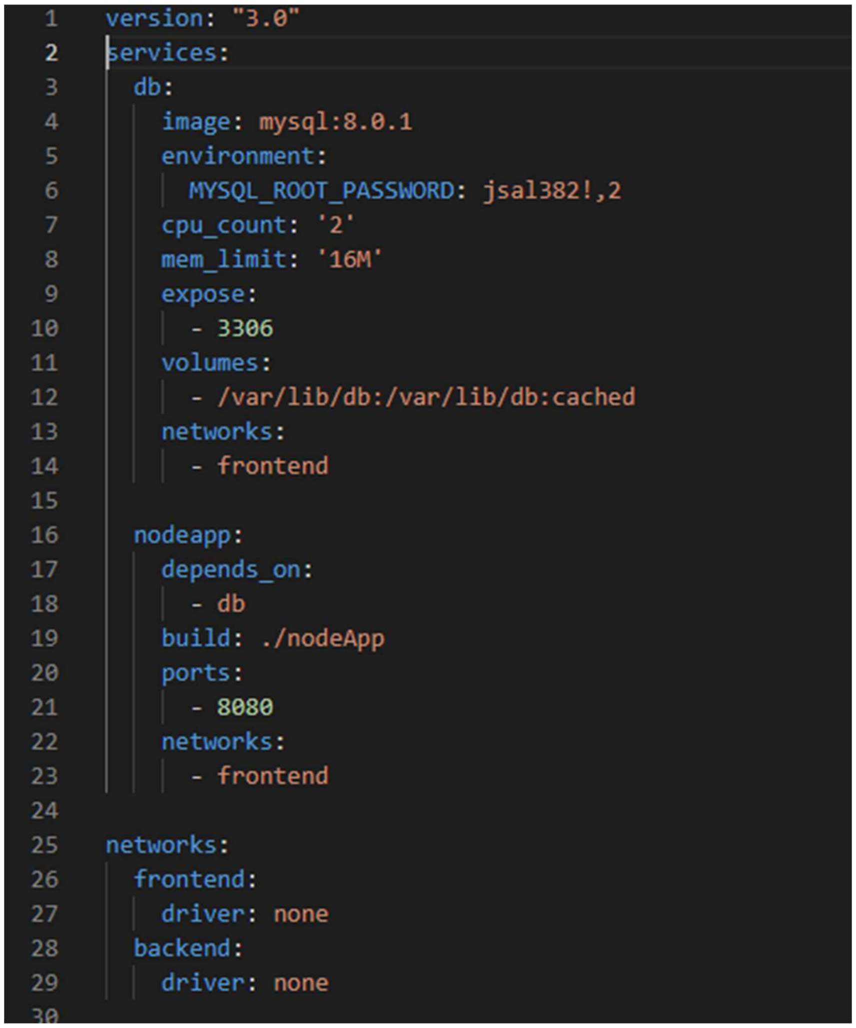
(Note- in the newer version of compose – "EXPOSE" is not effective, as all container ports are by default accessible from other container ports in the same node.

c. The networks section has wrong syntax – The hierarchy should be just name -> driver



```
networks:
  frontend:
    driver: none
```

The above steps should essentially have our setup ready for the developers to begin testing their node app. My final version of .yaml file looks like this –



```
version: "3.0"
services:
  db:
    image: mysql:8.0.1
    environment:
      MYSQL_ROOT_PASSWORD: jsal382!,2
    cpu_count: '2'
    mem_limit: '16M'
    expose:
      - 3306
    volumes:
      - /var/lib/db:/var/lib/db:cached
    networks:
      - frontend

  nodeapp:
    depends_on:
      - db
    build: ./nodeApp
    ports:
      - 8080
    networks:
      - frontend

networks:
  frontend:
    driver: none
  backend:
    driver: none
```