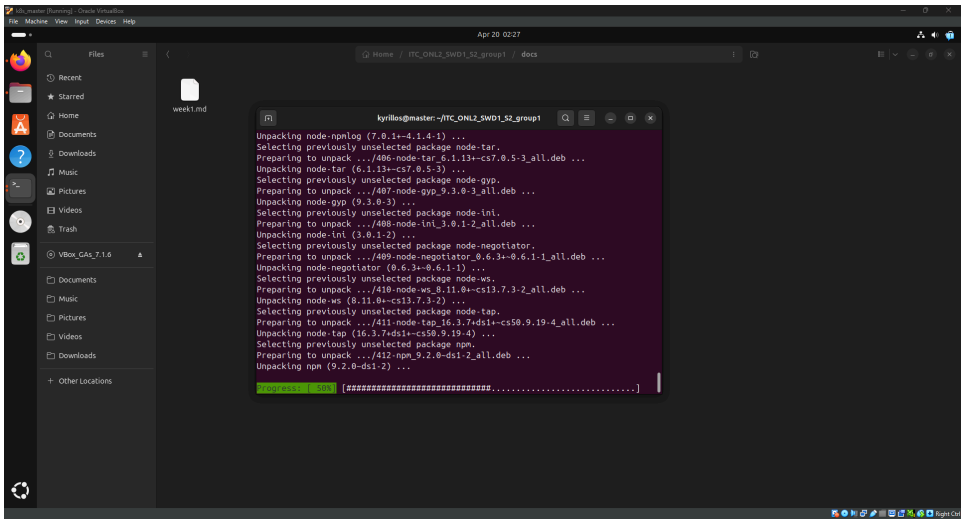
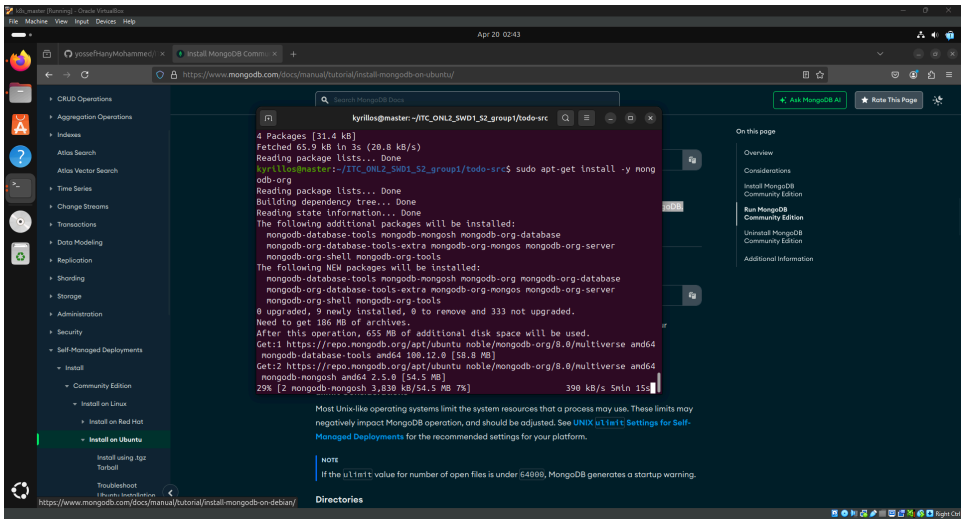
Week 1: Environment Setup, App Initialization & Dockerization

1. Set up the local development environment: Install and configure necessary DevOps tools (manual)
2. Choose and prepare a simple project to work with
3. Clone the project repo locally
4. Set up GitHub and classic token
5. Reload the local packages using `sudo apt update`
6. Install project requirements and dependencies
7. Install npm on machine



8. `cd todo-src`
9. `npm install`
10. Try `sudo apt install -y mongodb` but encountered an error (Package 'mongodb' 8. has no installation candidate)
11. Head to MongoDB official documentation for installation guide: https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/



From a terminal, install `gnupg` and `curl` if they are not already available:

```
sudo apt-get install gnupg curl
```

To import the MongoDB public GPG key, run the following command:

```
curl -fsSL https://www.mongodb.org/static/pgp/server-8.0.asc | \   sudo gpg -o /usr/share/keyrings/mongodb-server-8.0.gpg \   --dearmor
```

Create the list file for Ubuntu 24.04 (Noble):

```
echo"deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-8.0.gpg ] https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 multiverse" | sudotee /etc/apt/sources.list.d/mongodb-org-8.0.list
```

You can install the latest stable version of MongoDB **Community Server:**

```
sudo apt-get install -y mongodb-org
```

12. Install production dependencies (where `package.json` lives):

```
npm install bcrypt chalk cors dotenv express mongoose \
  connect-mongodb-session express-session validator ejs --save

# Tidy up low-risk issues:
npm audit fix
```

13. Install development tools:

```
npm install nodemon eslint prettier eslint-config-prettier --save-dev
eslint --init
```

14. Add a dev-script to your `package.json`:

```
"dev": "nodemon -L server.js",
"start": "node server.js",
```

15. **Create a** `.env` (in the same folder as `src/index.js`):

```
MONGO_URI=mongodb://todo-database:27017/todo-app

SESSION_SECRET=my-super-secret

PORT=3000
```

16. Run MongoDB (already installed it):

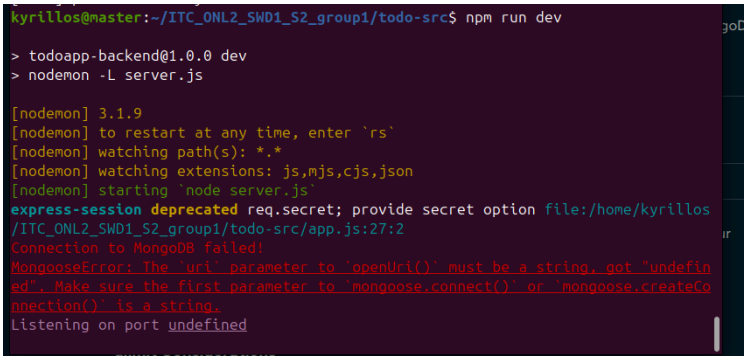
```
sudo systemctl start mongod
```

17. Launch app in dev mode:

```
npm run dev
```

18. **Open** http://localhost:3000 in browser.

19. Encountered this error:

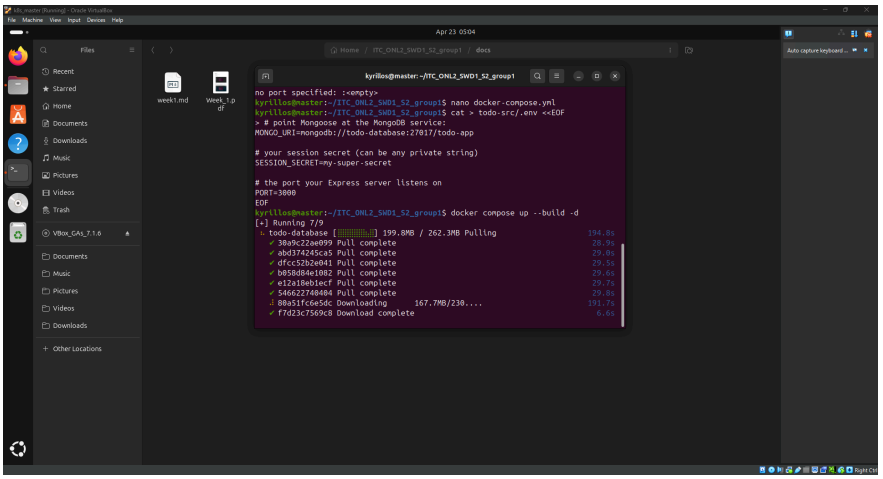


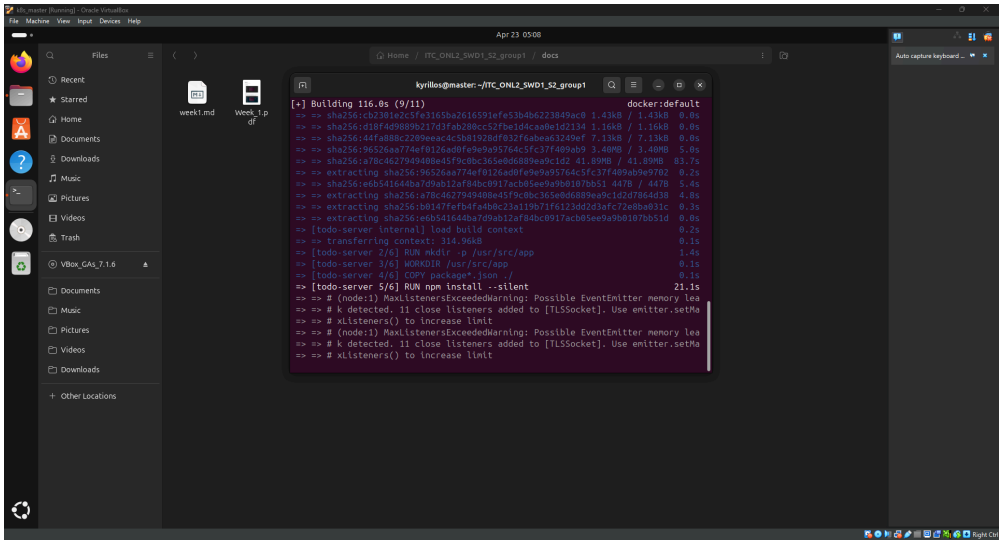
20. Try installing mongo using Docker Compose instead
21. First: Dockerfile for backend

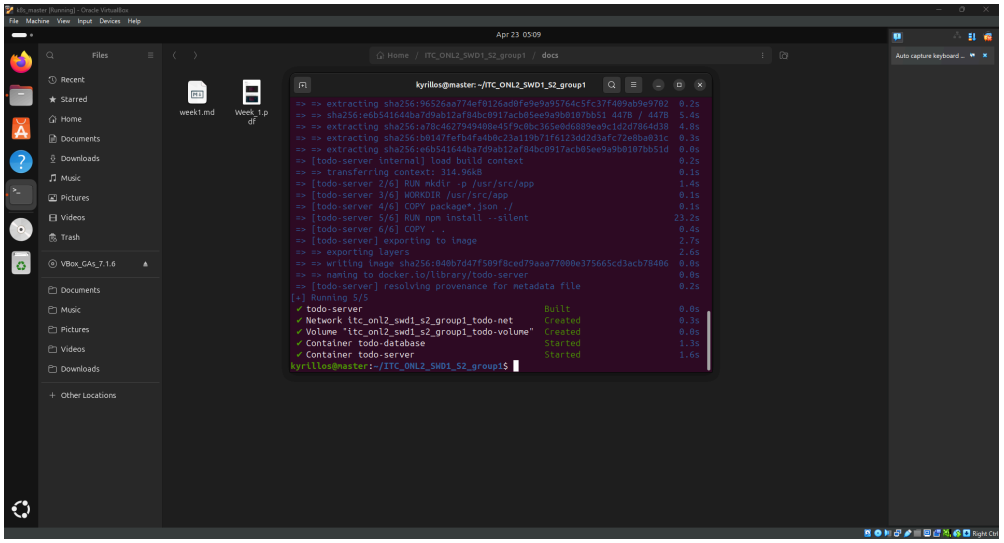
```
FROM node:20.9-alpine

# 1st: Create app directory
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

# 2nd: Copy package.json and package-lock.json to the working directory
COPY package*.json ./

# 3rd: Install npm
RUN npm install --silent

# 4th: Copy source code
COPY . .

# 5th: Exports
EXPOSE 8082

# 6th: Start node server
CMD ["npm","start"]
```

22. Then: docker-compose file

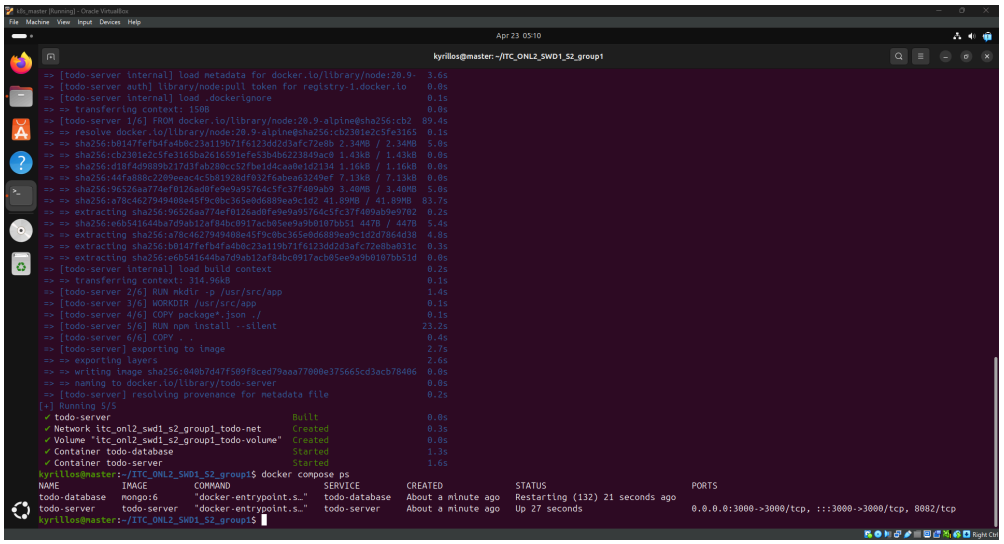
```
version: '3.9'
services:
 todo-database:
   image: mongo
   container_name: todo-database
   restart: always
   ports:
     - "27017:27017"
   volumes:
     - todo-volume:/data/db
   networks:
     - todo-net

 todo-server:
   build:
     context: ./todo-src
     dockerfile: Dockerfile
   image: todo-server
   container_name: todo-server
   restart: on-failure
   ports:
     - "3000:3000"
   env_file:
     - ./todo-src/.env
   depends_on:
     - todo-database
   networks:
     - todo-net

volumes:
 todo-volume:

networks:
 todo-net:
   driver: bridge
```



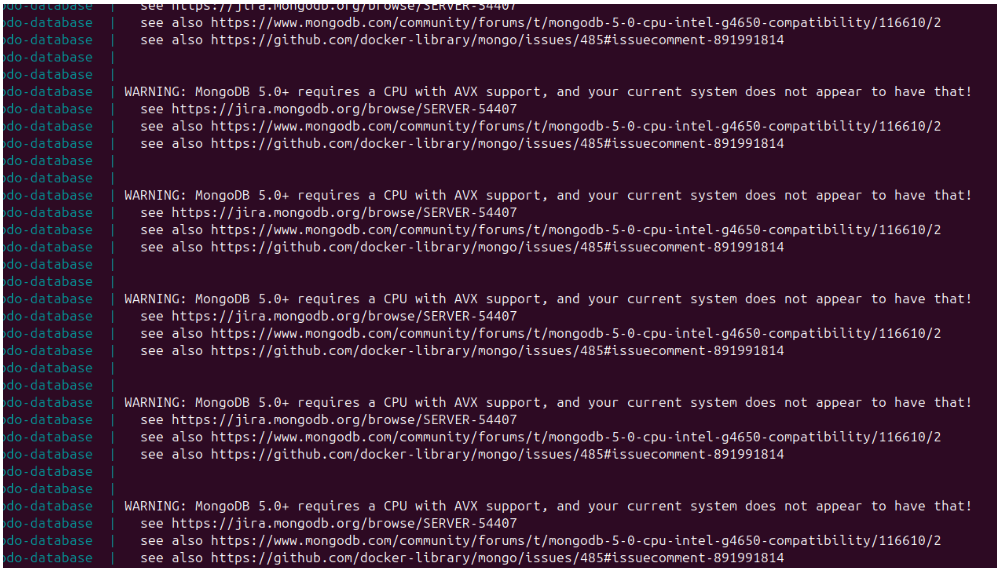




23. Verify the two containers have started using `docker compose ps`



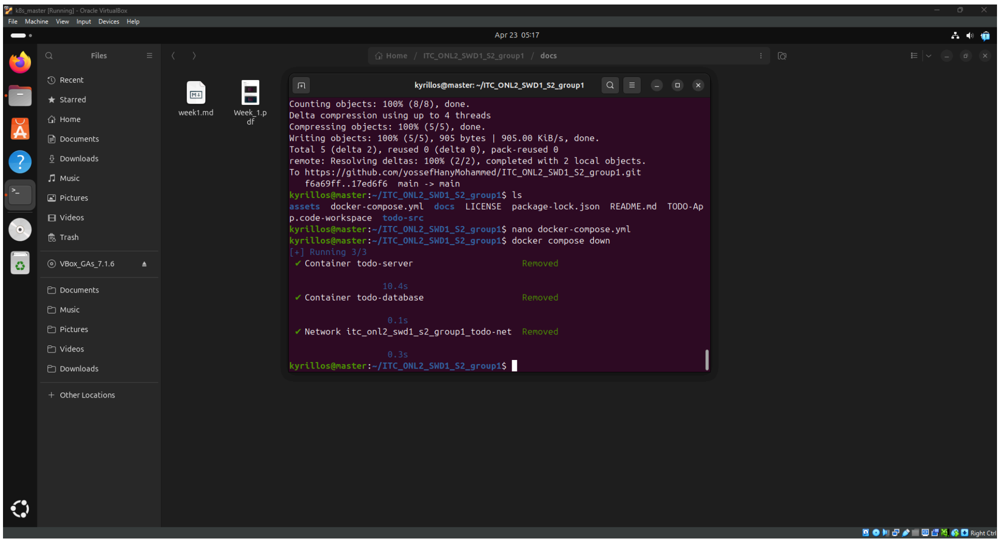
24. Problem database container exiting and restarting

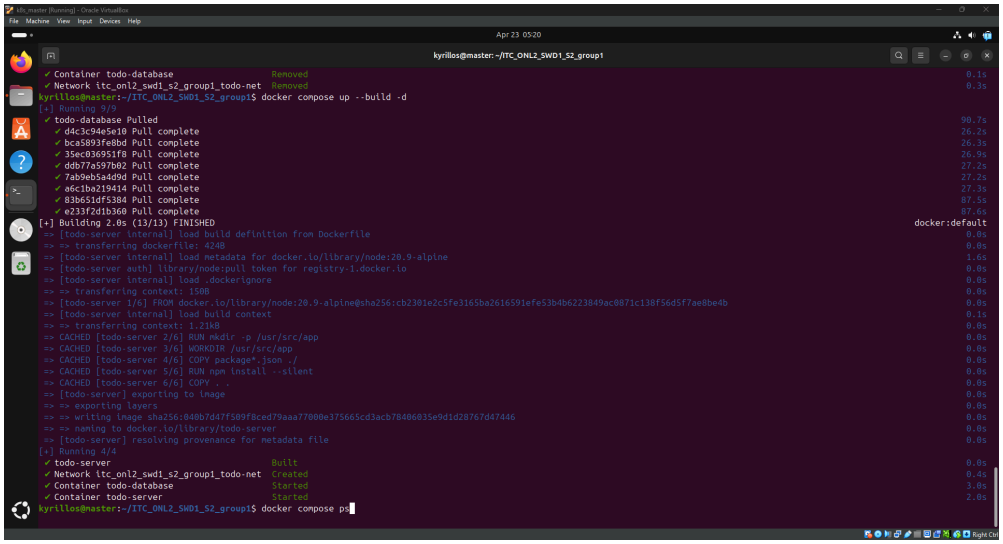


25. "Restarting (132)" paired with that AVX warning means CPU (or VM) doesn't support the AVX instructions that Mongo 5+ now requires, so `mongod` is crashing immediately. The quickest fix is to fall back to a 4.x Mongo image that doesn't need AVX.

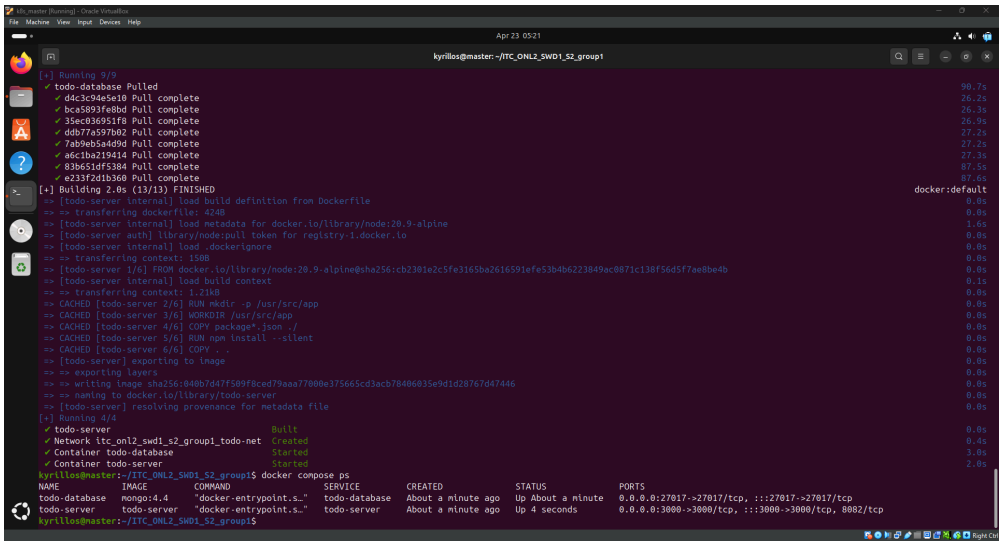
26. EDIT docker compose, first `docker compose down` then `docker compose up --build -d`

```
image: mongo:4.4
```

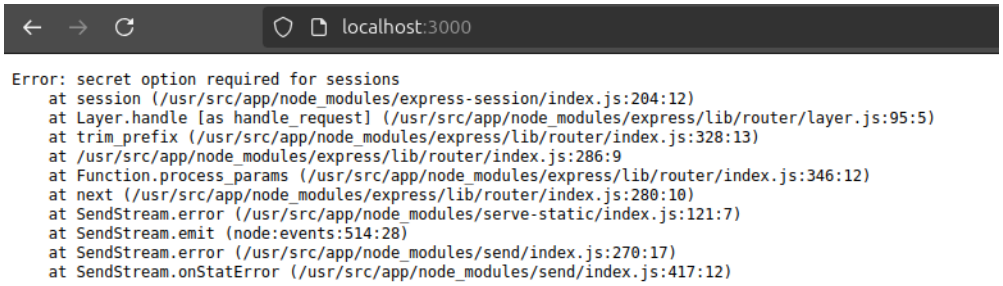




27. `docker compose ps` and now both are running



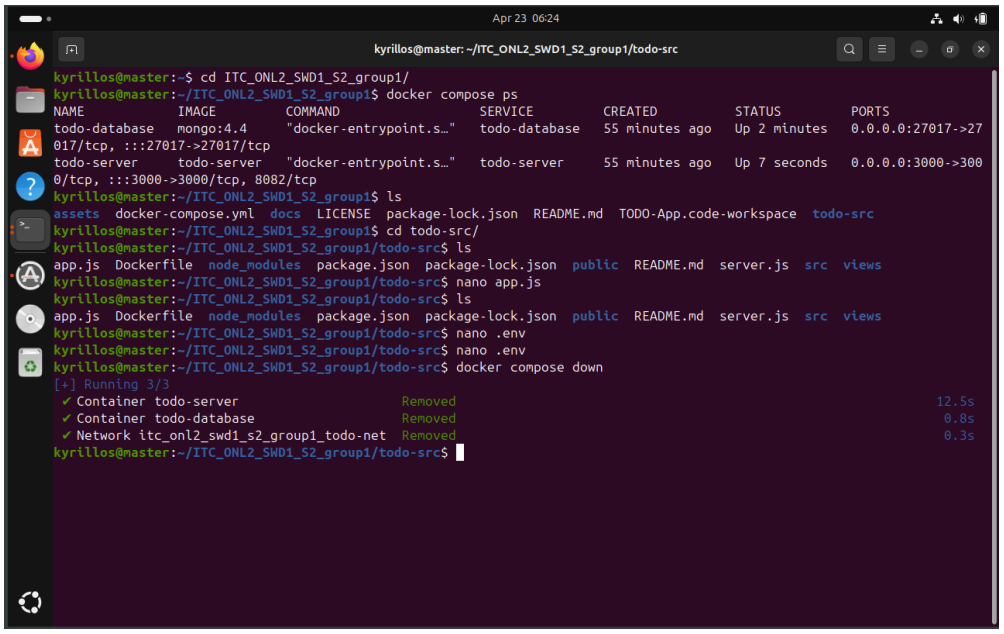
28. SECRET .env problem:

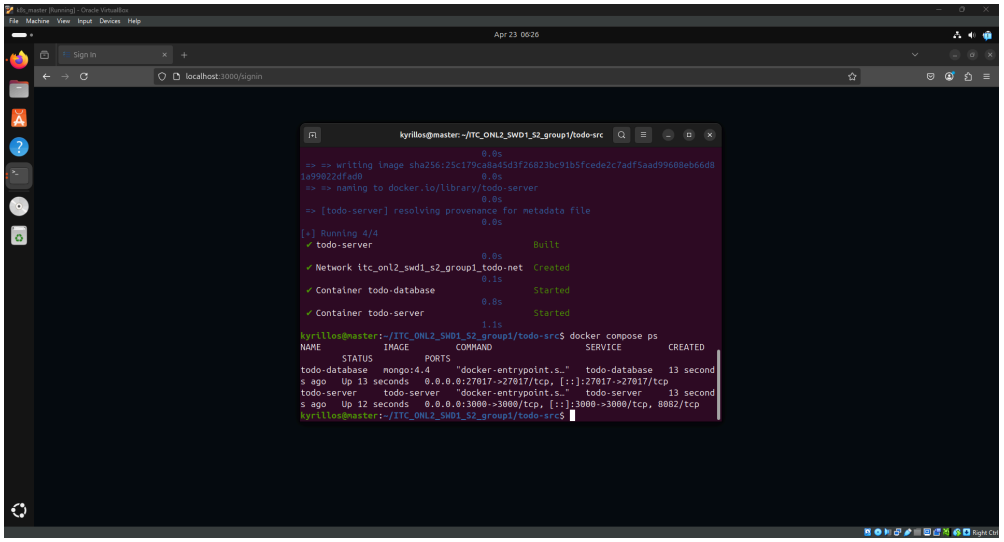


`process.env.SESSION_KEY` but we set `SESSION_SECRET` in `.env`. Since `SESSION_KEY` is undefined, express-session throws that “secret option required” error.

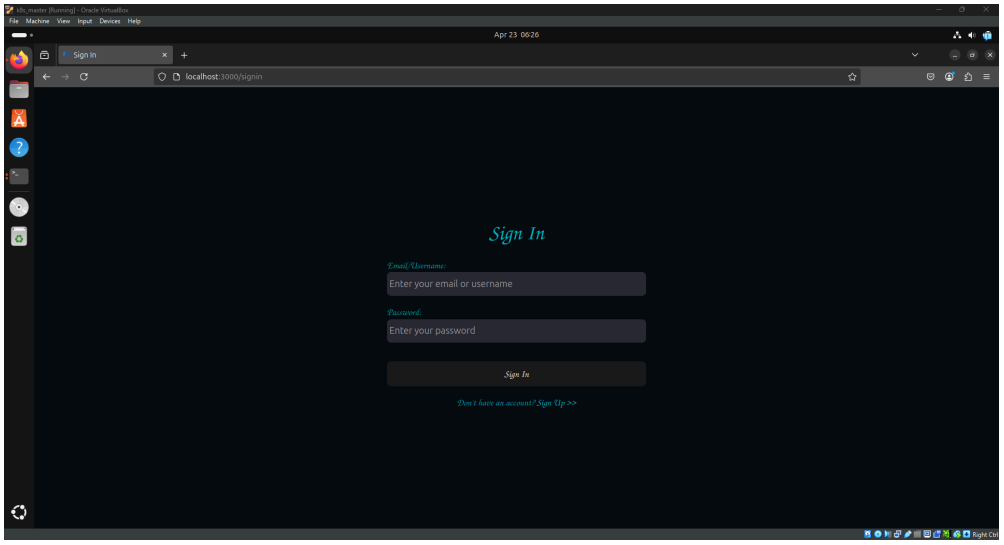
29. Edit the .env file:

```
MONGO_URI=mongodb://todo-database:27017/todo-app

SESSION_KEY=my-super-secret

PORT=3000
```





30. The app is dockerized and runninggg



31. Jenkins CI/CD Pipeline plan draft

Stage	Trigger	Tasks	Tooling	Deliverable
Build	Push to `main` or `dev`	• Checkout repo (`git clone ...`).• Build Docker image from `todo-src/Dockerfile` .• Validate & lint Dockerfile.	Docker, Docker Compose	`todo-server:ci-<SHA>` image
Test	On successful build	• Run `npm run lint` on `todo-src` .• (If you add tests) run `npm test` .• Run `npm audit` for vulns.	ESLint, Prettier, npm	Pass/Fail report in CI console
Publish	Tests pass	• Tag image with CI build number (`ci-<BUILD_ID>`).• Push images (`mongo` comes from off-the-shelf).	Docker CLI, Docker Hub	Images in central registry
Deploy	Image push (or manual)	• Pull new image on target host.• Run `docker compose pull && docker compose up -d` .• Smoke-test HTTP.	Docker Compose, curl	Live app at http://<host>:3000
Notify	Deploy success/failure	• Send Slack/Email alert.• On failure, trigger rollback playbook (e.g. re-deploy previous tag).	CI Notifications plugin	Channel/Email updated with pipeline status