Docker :

Some important Topics to explore :

WSL (windows subsystem for Linux)

<https://docs.microsoft.com/en-us/windows/wsl/about>

The Windows Subsystem for Linux lets developers run a GNU/Linux environment -- including most command-line tools, utilities, and applications -- directly on Windows, unmodified, without the overhead of a traditional virtual machine or dualboot setup

WSL2 :

WSL 2 is a new version of the Windows Subsystem for Linux architecture that powers the Windows Subsystem for Linux to run ELF64 Linux binaries on Windows. Its primary goals are to **increase file system performance**, as well as adding **full system call compatibility**.

This new architecture changes how these Linux binaries interact with Windows and your computer's hardware, but still provides the same user experience as in WSL 1 (the current widely available version).

<https://docs.docker.com/desktop/windows/wsl/>

Windows Subsystem for Linux (WSL) 2 introduces a significant architectural change as it is a full Linux kernel built by Microsoft, allowing Linux containers to run natively without emulation. With Docker Desktop running on WSL 2, users can leverage Linux workspaces and avoid having to maintain both Linux and Windows build scripts. In addition, WSL 2 provides improvements to file system sharing, boot time, and allows access to some cool new features for Docker Desktop users.

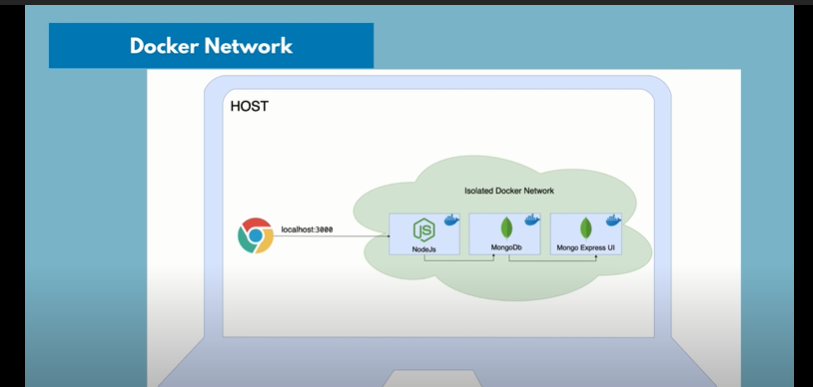
Docker Desktop uses the dynamic memory allocation feature in WSL 2 to greatly improve the resource consumption. This means, Docker Desktop only uses the required amount of CPU and memory resources it needs, while enabling CPU and memory-intensive tasks such as building a container to run much faster.

Additionally, with WSL 2, the time required to start a Docker daemon after a cold start is significantly faster. It takes less than 10 seconds to start the Docker daemon when compared to almost a minute in the previous version of Docker Desktop.

Source : <https://gitlab.com/nanuchi/techworld-js-docker-demo-app>

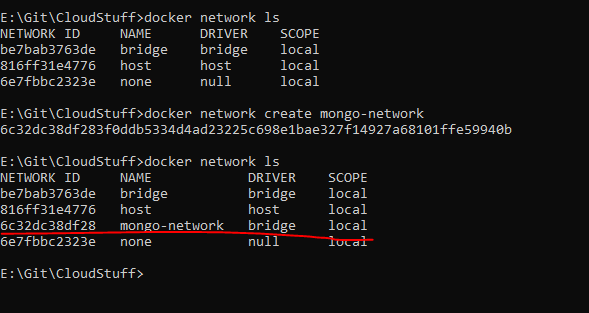
Some Point need to Explore

**Docker Network** : Internal communication between containers.



Source : <https://www.youtube.com/watch?v=3c-iBn73dDE&t=4866>

1. Default Network and new network created for application



1. Run Mongo Containers

docker run -d -p 27017:27017 -e MONGO\_INITDB\_ROOT\_USERNAME=admin -e MONGO\_INITDB\_ROOT\_PASSWORD=password --name mongodb --net mongo-network mongo

1. Run Mongo Express

docker run -d -p 8081:8081 -e ME\_CONFIG\_MONGODB\_ADMINUSERNAME=admin -e ME\_CONFIG\_MONGODB\_ADMINPASSWORD=password --net mongo-network --name mongo-express -e ME\_CONFIG\_MONGODB\_SERVER=mongodb mongo-express

1. Access Data base express : <http://localhost:8081>
2. Create New Database: **user-account**
3. Start your nodejs application locally - go to app directory of project

npm install

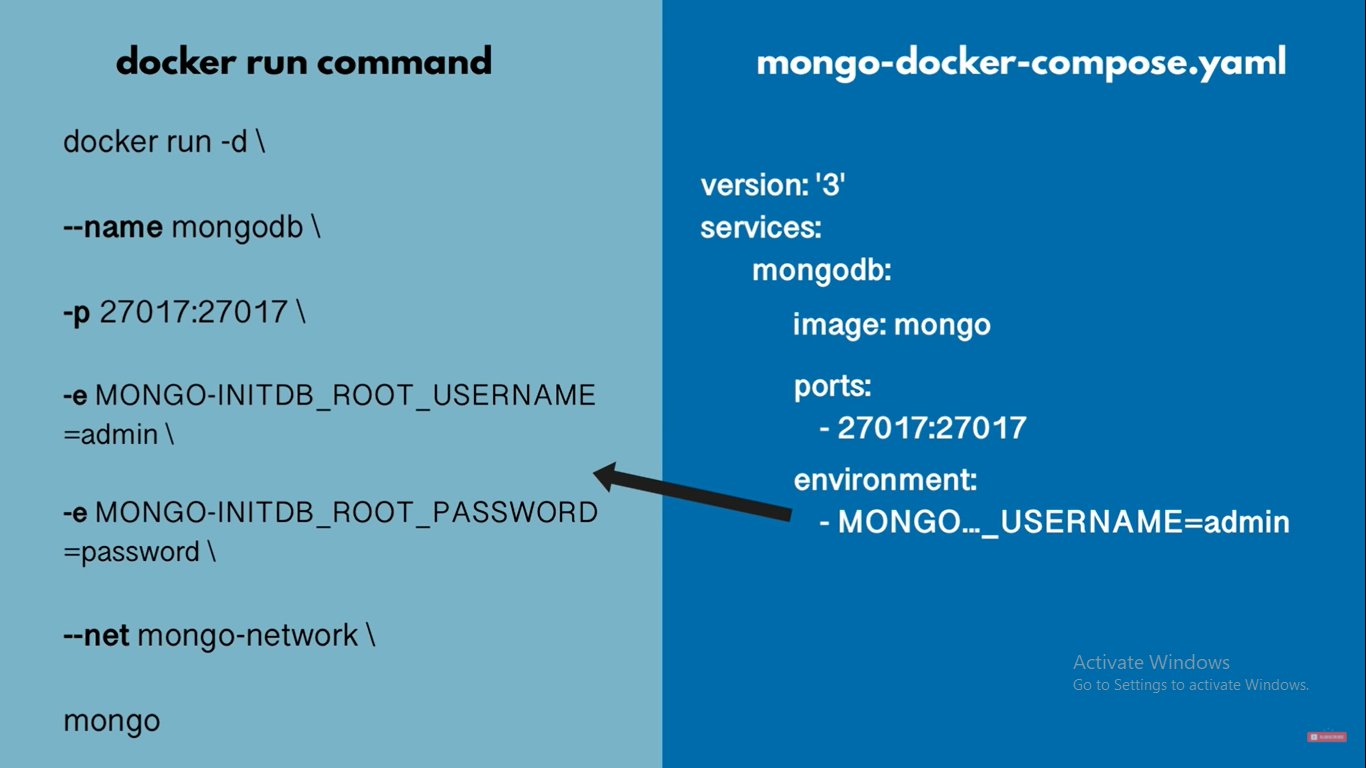
node server.js

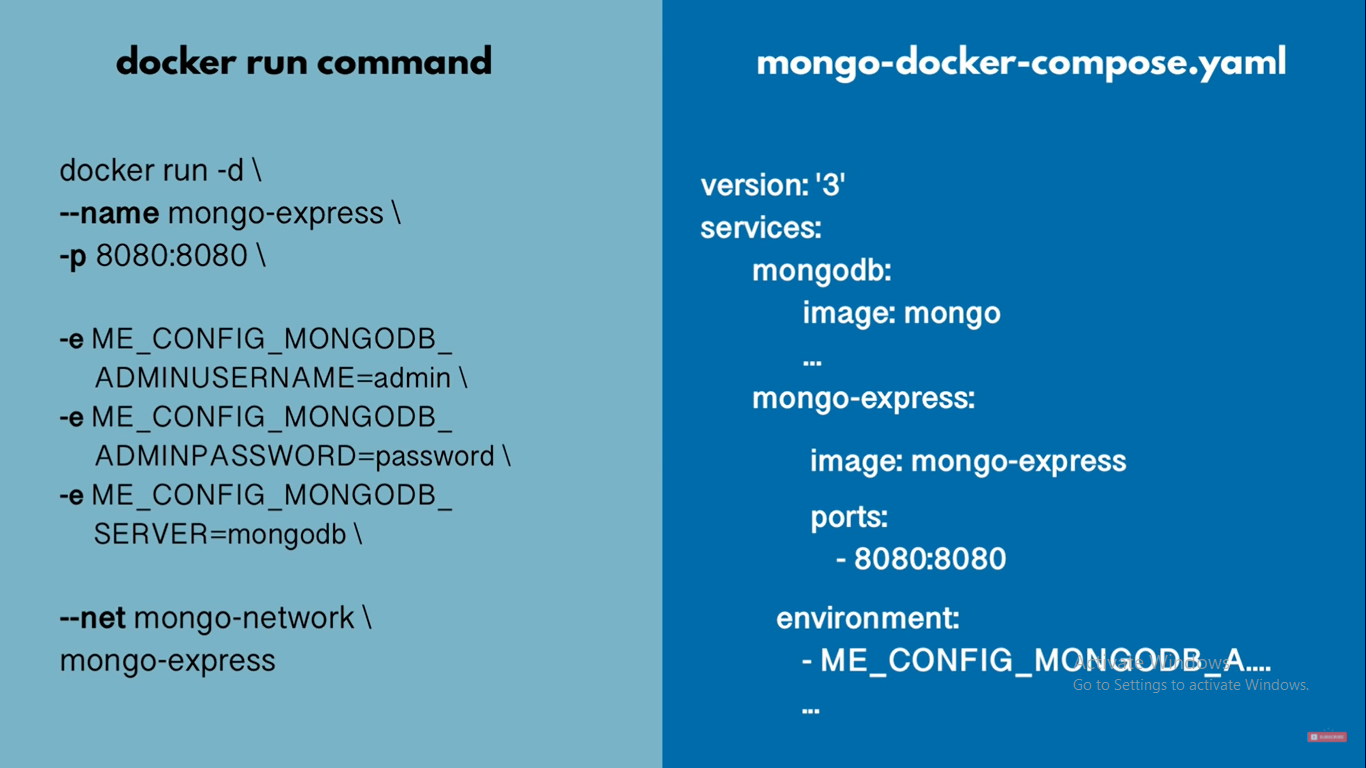
Finally, able to update the record in MongoDB

Docker Compose

<https://docs.docker.com/compose/#features>

1. Compose is a tool for defining and running multi-container Docker applications
2. use a YAML file to configure your application’s services
3. single command, can create and start all the services from your configuration





Source : <https://www.youtube.com/watch?v=3c-iBn73dDE&t=4866>

**So Where is Docker-Network or Network configuration in Docker Compose file ?**

Docker compose will take care of creating a common network

Step 1: start mongodb and mongo-express

docker-compose -f docker-compose.yaml up