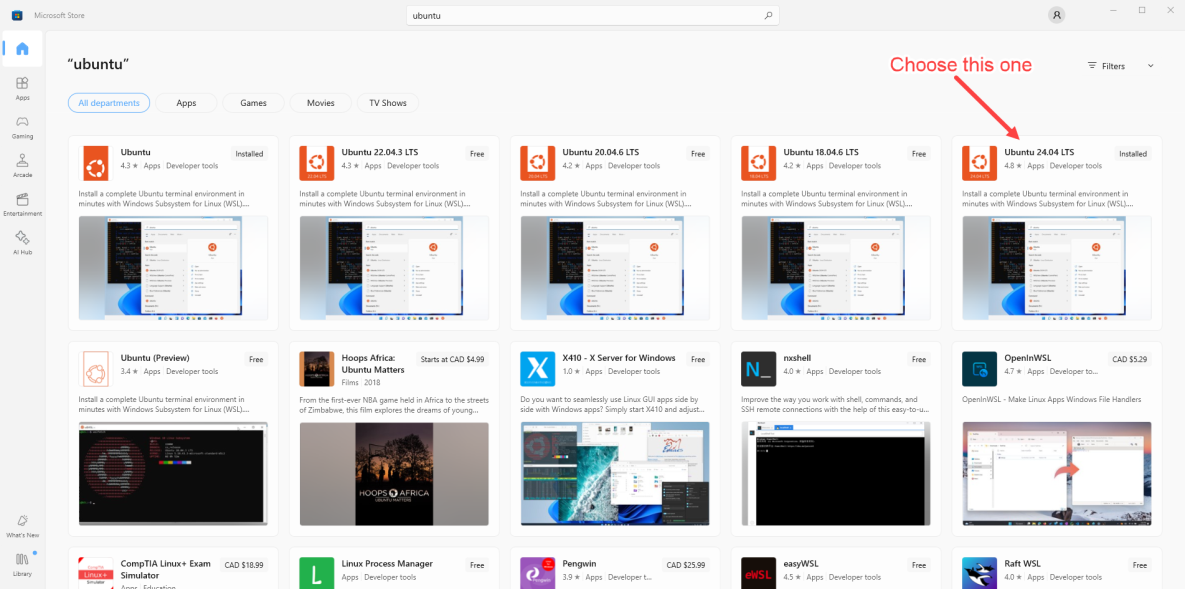
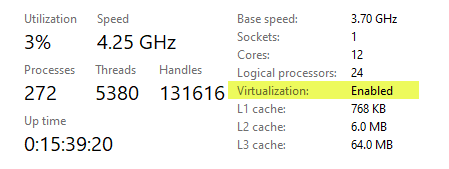
1. Developing on Windows OS, we first need to install a full version of the Linux OS, we need this to create and build Docker images and containers. Windows store comes with a flavor of Ubuntu LTS linux. We can run what is called Windows Subsystem Linux (WSL) OS on windows and keep our development on Windows
   1. Go to Windows Store, search for “ubuntu” and choose WSL Ubuntu 24.04 LTS, this is the latest long term support version at the time of this writing.

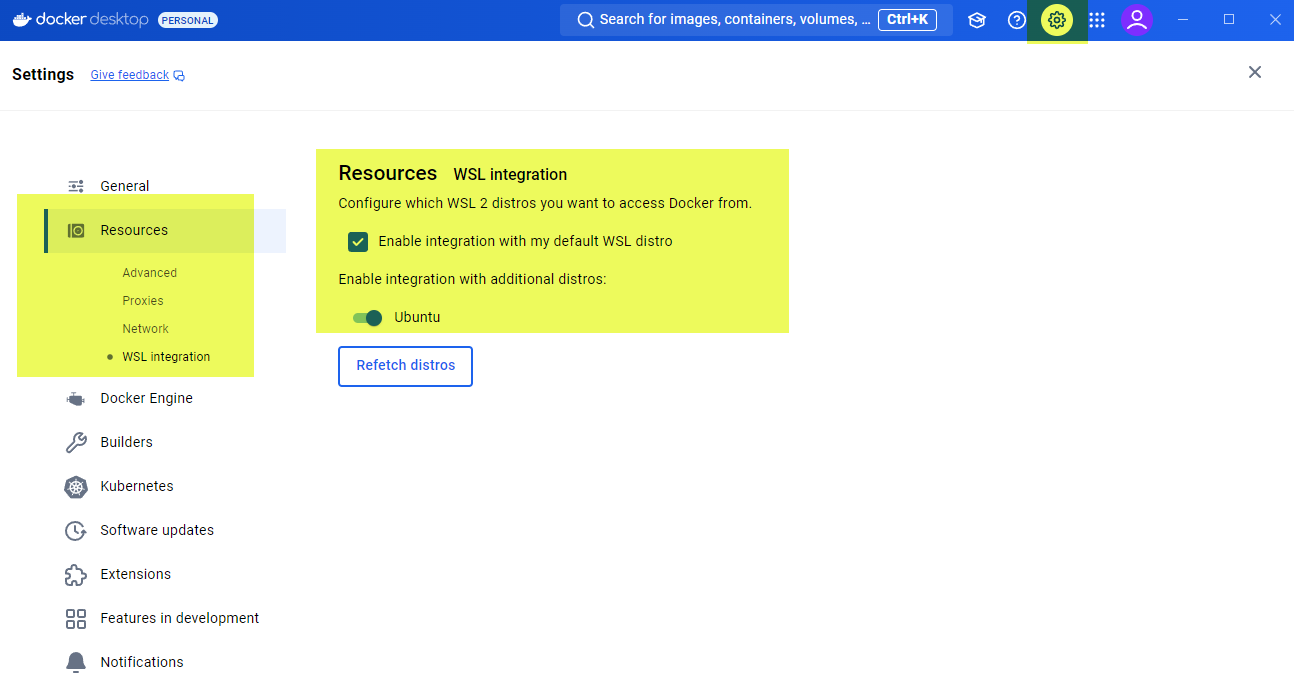


* 1. Install the Ubuntu WSL from the Windows Store

1. Download Docker Desktop for Windows OS version here -> [Docker Desktop: The #1 Containerization Tool for Developers | Docker](https://www.docker.com/products/docker-desktop/)
   1. Make sure you set your Windows machine to enable Virtualization, to verify, open Task Manager, under Performance tab, you should see the following:



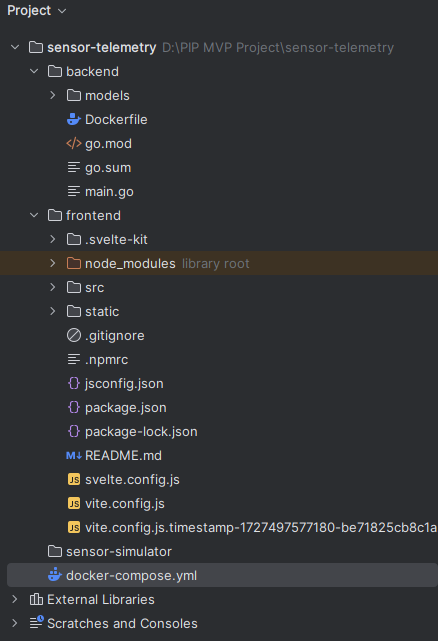
* 1. Install Docker Desktop, when ask if you want to use Hyper V or WSL, choose WSL because we are using Ubuntu WSL, this will give us a development environment that is similar to our production environment.
  2. Run Docker Desktop
  3. Go to Settings->Resources->WSL Integration, and enable integration for WSL distro, in our case it’s Ubuntu as shown below:



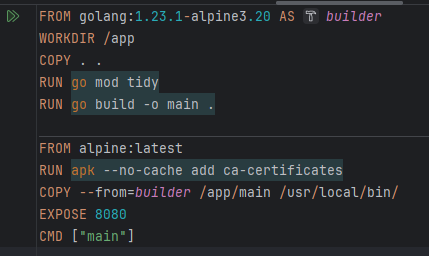
1. Install development IDE of your choice, VS Code is good.
2. Create your project boiler plate and add docker-compose.yml and Docker files to proper place
3. Keep Docker Desktop open and make sure WSL is running by restarting your Windows
4. Docker development environment is ready to go!
5. Preparing the **docker-compose.yml** file to include all the components for the VPC projects (a backend folder utilizing Golang, a frontend folder utilizing Svelte & Svelte Kit, Apache Kafka, Quest DB, and all their dependencies). Please refer to the following working **docker-compose.yml** file

services:  
 go-dev:  
 image: golang:1.23.1-alpine3.20  
 volumes:  
 - ./backend:/app  
 working\_dir: /app  
  
 backend:  
 build: ./backend  
 ports:  
 - "8080:8080"  
 depends\_on:  
 - kafka  
 - questdb  
 environment:  
 KAFKA\_BROKERS: kafka:9092  
 QUESTDB\_URL: questdb:9000  
  
 frontend:  
 image: node:alpine  
 ports:  
 - "3000:3000"  
 working\_dir: /app  
 depends\_on:  
 - backend  
 volumes:  
 - ./frontend:/app  
 entrypoint: ['npm', 'run', 'dev']  
  
 zookeeper:  
 image: confluentinc/cp-zookeeper:7.4.0  
 environment:  
 ZOOKEEPER\_CLIENT\_PORT: 2181  
 ZOOKEEPER\_TICK\_TIME: 2000  
 ports:  
 - "2181:2181"  
 volumes:  
 - zookeeper\_data:/var/lib/zookeeper *#persist zookeeper data* kafka:  
 image: confluentinc/cp-kafka:7.4.0  
 depends\_on:  
 - zookeeper  
 ports:  
 - "9092:9092"  
 environment:  
 KAFKA\_BROKER\_ID: 1  
 KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181  
 KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT\_INTERNAL://kafka:29092  
 KAFKA\_LISTENERS: PLAINTEXT://0.0.0.0:9092,PLAINTEXT\_INTERNAL://0.0.0.0:29092  
 KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT\_INTERNAL:PLAINTEXT  
 KAFKA\_INTER\_BROKER\_LISTENER\_NAME: PLAINTEXT\_INTERNAL  
 KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1  
 KAFKA\_LOG\_RETENTION\_HOURS: -1 *# Infinite retention* KAFKA\_LOG\_RETENTION\_BYTES: -1 *# No size limit for retention* volumes:  
 - kafka\_data:/var/lib/kafka  
  
 kafka-rest-proxy:  
 image: confluentinc/cp-kafka-rest:7.4.0  
 depends\_on:  
 - kafka  
 ports:  
 - "8082:8082"  
 environment:  
 KAFKA\_REST\_BOOTSTRAP\_SERVERS: "PLAINTEXT://kafka:29092"  
 KAFKA\_REST\_LISTENERS: "http://0.0.0.0:8082"  
  
 questdb:  
 image: questdb/questdb:latest  
 ports:  
 - "9000:9000"  
  
volumes:  
 zookeeper\_data:  
 kafka\_data:

1. Structure the project folder as follow:

****

1. Prepare the Dockerfile in the backend folder as follow in order for Docker to build the Go application and map the proper working folder.

****

1. Open the terminal in the ID of your choice, VS Code or Pycharm and run the “docker compose up –build” to build your docker image.