Remote File System and Kernel Booting for Embedded Systems: A Dockerized Approach

Table of contents

1. Introduction

- Overview
- Technologies Used
- Intended Audience
- Objectives

2. Kernel booting Sequence

- General Boot Process
- Benefits of Remote Booting
- Remote Booting

3. NFS and TFTP and u-boot configuration

- Introduction to NFS and TFTP and picocom
- Configuration of NFS and TFTP Servers
- configuration of u-boot using picocom

4. Docker

- What is Docker?
- Benefits of Docker
- Why Docker was used over vm
- How Docker Was Used in This Project

5. Project Setup

- Requirements for the Project
- Project Startup

6. Live Demonstration

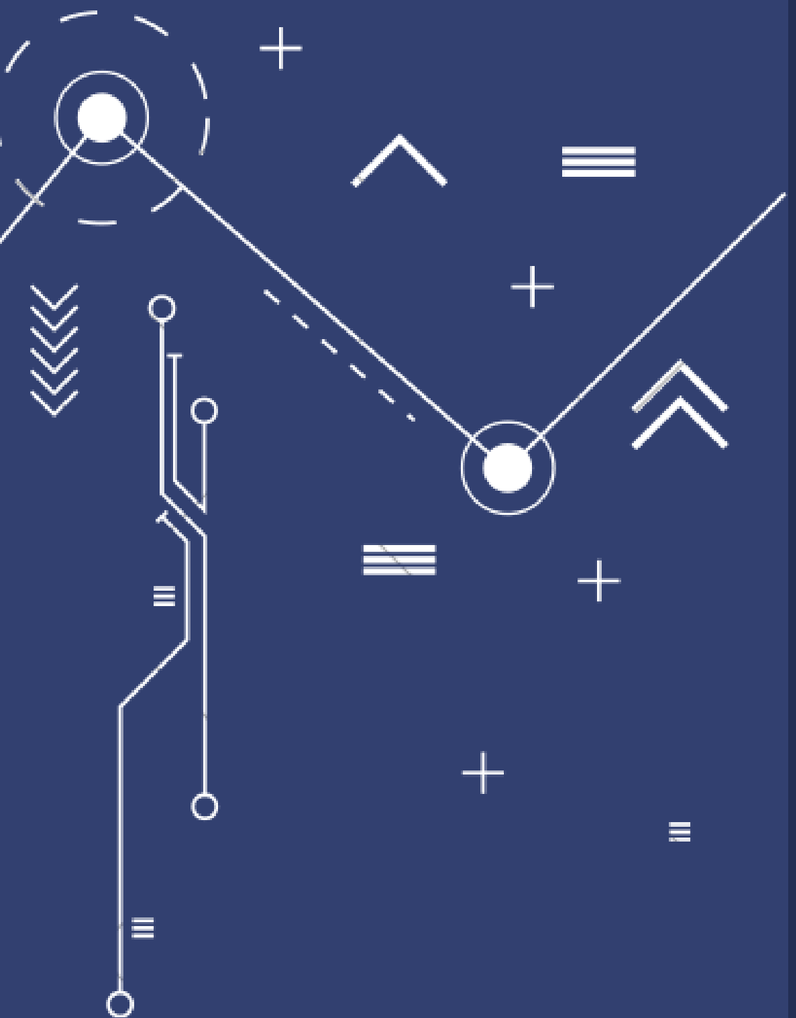
- System in Action

7. Expanding the usage of dockerised servers

- Potential Additional Use Cases
- Integration with Online Repositories for Automated Updates

8. Q&A Section

INTRODUCTION



OVERVIEW

Technology Used

TFTP

NFS

picocom
Empowering Wireless

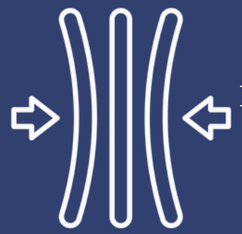
docker

Targeted Audience

Objectives

Simplify Remote Managment

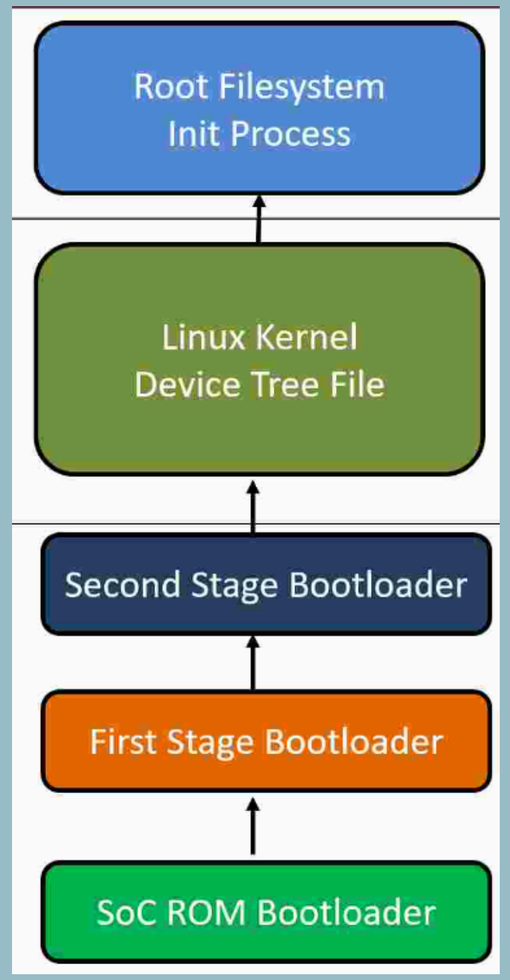
Dockerized Service managment



Provide a flexiable Development and Deployment

KERNEL BOOTING SEQUENCE

Kernel Normal Booting Sequence



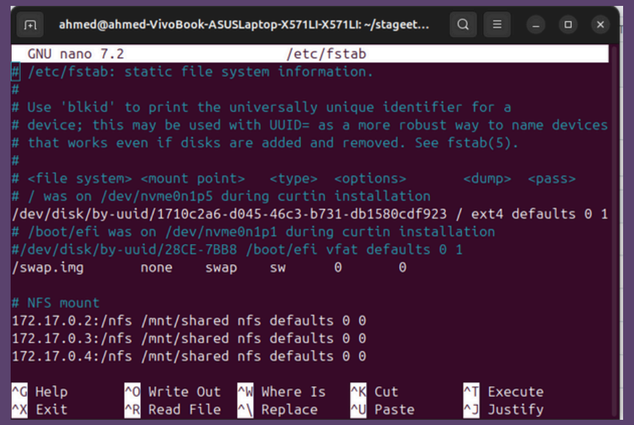
Benefits of Remote Booting

Remote Booting without Docker

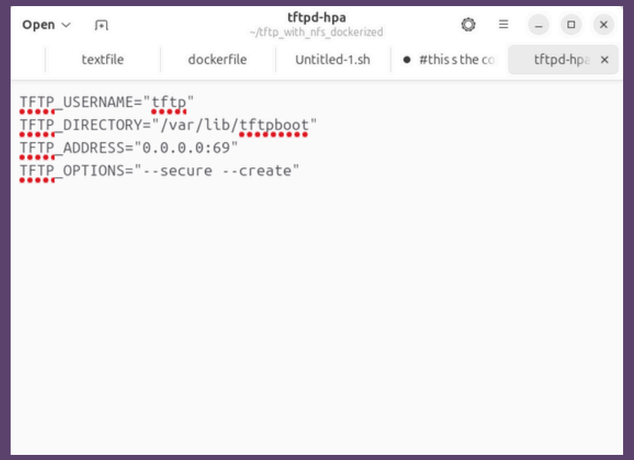
NFS AND TFTP AND PICOCOM CONFIGURATION

Introduction To NFS ,TFTP,PICOCOM

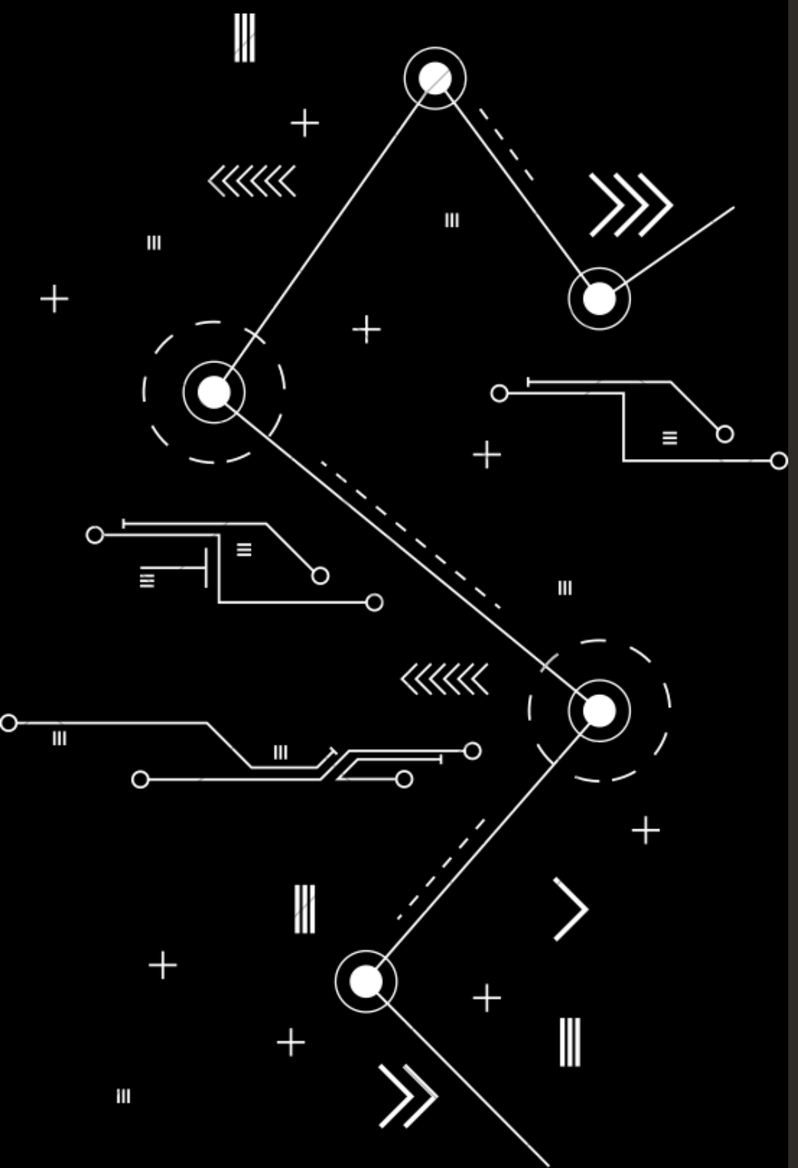
Configuration NFS



Configuration TFTP



Docker



WHAT S DOCKER ?

Benefits

Why we choose Docker Over VM?

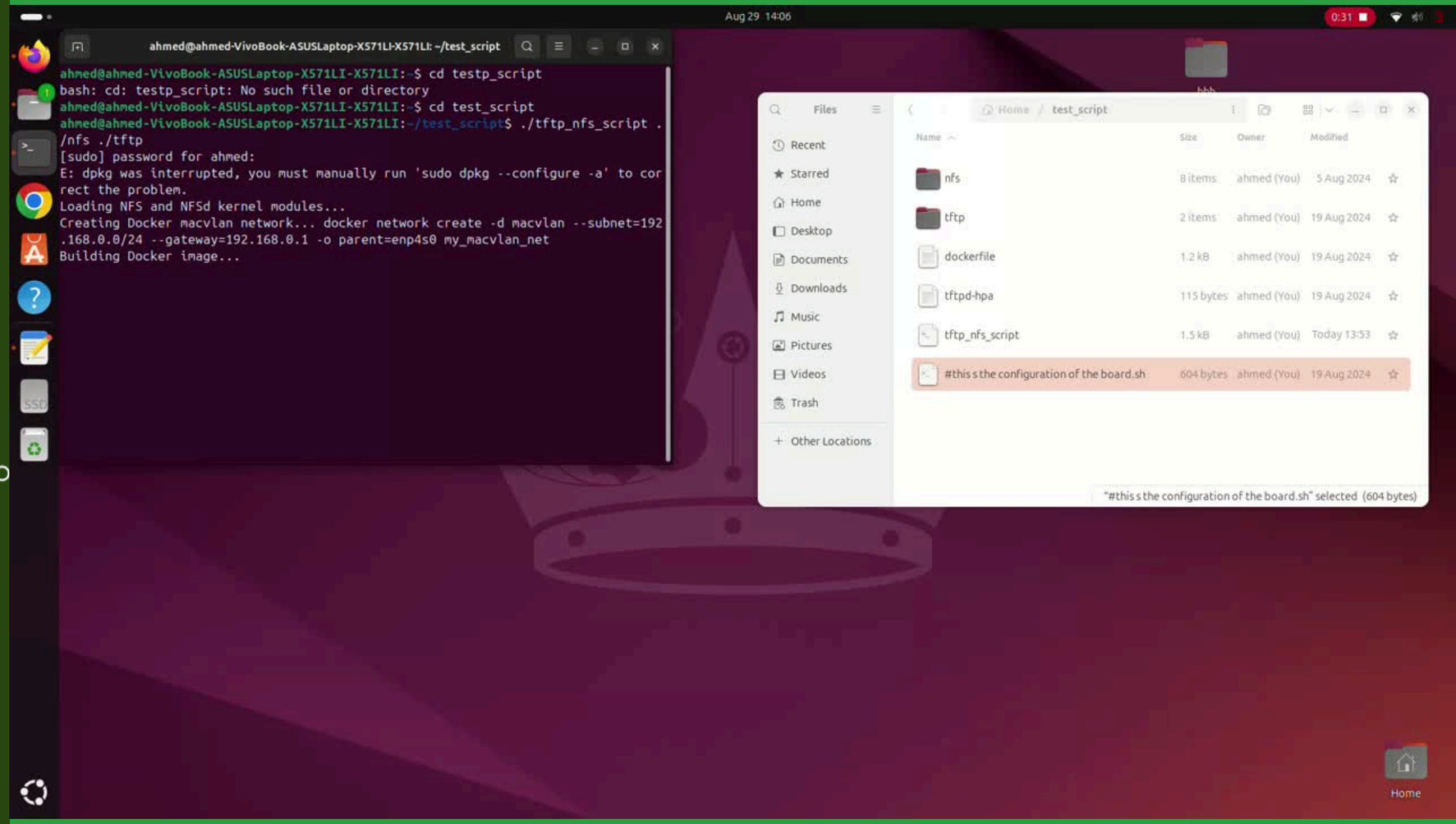
How we Used Docker?

Project Setup

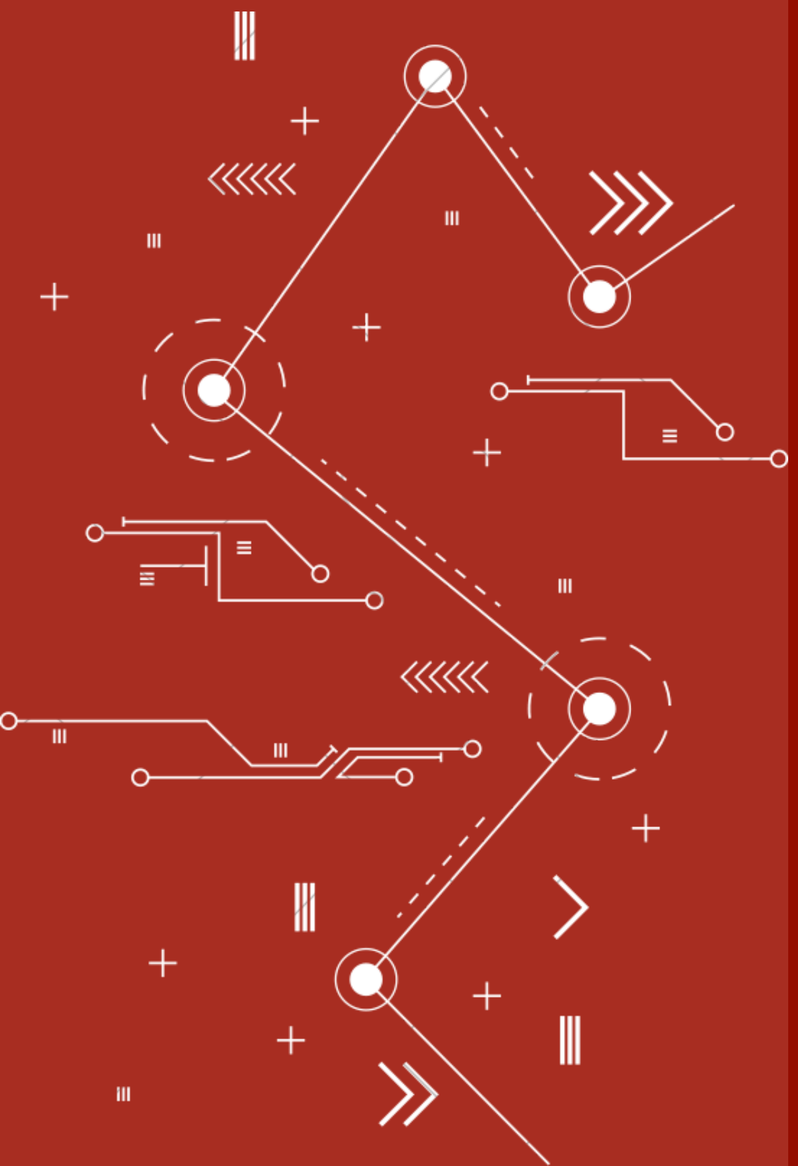
Requirements for the Project

Network Configuration

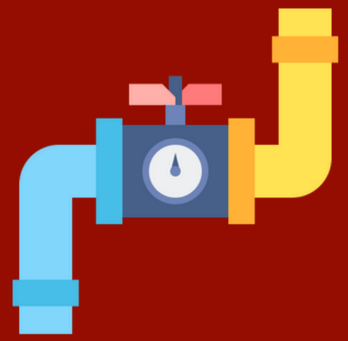
Live Demonstration



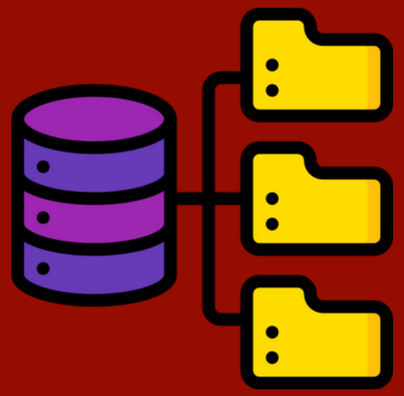
Expanding the use of Dockerized servers



PIPELINES



Data bases



Future applications

Q and A

ASK AWAY!

THANK YOU!

GNU nano 7.2 tftp_nfs_script

```bash
#!/bin/bash
sudo apt-get install -y nfs-kernel-server

# Load NFS and NFSd kernel modules
echo "Loading NFS and NFSd kernel modules..."
sudo modprobe nfs
sudo modprobe nfsd

# Get directories or repository URLs from command-line arguments
NFS_INPUT="$1"
TFTP_INPUT="$2"

# Handle NFS directory or GitHub repository URL
if [[ "$NFS_INPUT" == *"git.com"* ]]; then
    if [ ! -d "./nfs" ]; then
        echo "Creating directory ./nfs..."
        mkdir -p "./nfs"
    fi
    echo "Cloning repository from $NFS_INPUT into ./nfs..."
    git clone "$NFS_INPUT" "./nfs"
    NFS_DIR="./nfs"
else
    NFS_DIR="$NFS_INPUT"
fi

# Handle TFTP directory or GitHub repository URL
if [[ "$TFTP_INPUT" == *"git.com"* ]]; then
    if [ ! -d "./tftp" ]; then
        echo "Creating directory ./tftp..."
        mkdir -p "./tftp"
    fi
    echo "Cloning repository from $TFTP_INPUT into ./tftp..."
    git clone "$TFTP_INPUT" "./tftp"
    TFTP_DIR="./tftp"
else
    TFTP_DIR="$TFTP_INPUT"
fi

# Create Docker macvlan network
echo "Creating Docker macvlan network..." docker network create -d macvlan --subnet=192.168.0.0/24 --gateway=192.168.0.1   -o parent=enp4s0   my_macvlan_net

# Build Docker image
echo "Building Docker image..."
docker build -t tftpnfs .

# Run Docker container
echo "Running Docker container..."
docker run -d --name tftp-nfs-server  --network my_macvlan_net   --ip 192.168.0.9  --privileged   -v $NFS_DIR:/nfs  -v $TFTP_DIR:/var/lib/tftpboot  -p 2080:2049   -p 69:69/udp   tftpnfs

echo "Docker setup complete."
```

```dockerfile
# Use an Ubuntu base image l'image 24.04 et 22.04 sont les plus stable
FROM ubuntu:24.04

# Install necessary packages, including TFTP-HPA, NFS, and UFW portmapper(rpcbind needed for nfs)
RUN apt-get update && \
    apt-get install -y tftpd-hpa nfs-kernel-server nfs-common rpcbind ufw && \
    apt-get clean

# Create directories for TFTP and NFS root ou je met les volume
RUN mkdir -p /var/lib/tftpboot /nfs && \
    chown nobody:nogroup /nfs

# Copy the TFTP configuration file into the container file exists in the folder u can configure it however u want
COPY tftpd-hpa /etc/default/tftpd-hpa

# Configure NFS exports /nfs path de folder que je l'utilise * pour tous l'adresse rights rw readwrite sync syncing of folders inside and outside the countainer no-subtree-check no permissions
RUN echo "/nfs *(rw,sync,no_subtree_check)" > /etc/exports

# Expose TFTP (UDP 69) and NFS (2049) ports
EXPOSE 69/udp 2049

# Start TFTP and NFS servers, disable UFW, and keep the container running
CMD ["bash", "-c", "ufw disable && mkdir -p /n/rurpcbind && rpcbind && rpc.statd && rpc.mountd && rpc.nfsd 8 && exportfs -rav && in.tftpd --foreground --secure /var/lib/tftpboot && tail -f /dev/null"]
```