

Multi Modal Intelligent Traffic Signal System

Docker SIL Setup

University of Arizona

Version 1.0

TABLE OF CONTENTS

Introduction	3
Docker	4
Setup Docker	4
Docker Image	4
Configuring Network.....	6
MMITSS SILS Package.....	7
applications	7
intersections.....	7
Scripts.....	8
run.sh	8
stop.sh.....	9
build.sh	9
Dockerfile	9
prepare_apps.sh	9

INTRODUCTION

The Software in the Loop (SIL) Simulation of MMITSS applications uses a Containerization Technology provided by a software called Docker. The purpose is to run MMITSS applications inside the Docker containers, as if they were running on individual devices like Savari's StreetWave™ or regular Linux PCs. Many such containers can be run on a single Linux PC with their own userspaces and network configuration.

This document will present the instructions required to run MMITSS applications in a Docker Environment on an Ubuntu PC by using Docker containers to emulate an RSE.

DOCKER

Docker is an open-source project that automates the deployment of applications inside software containers, by providing operating-system-level virtualization on Linux. This allows independent "containers" to run within a single Linux instance, avoiding the overhead of starting virtual machines. Docker is different from Virtual Machines in that it uses the Linux kernel's features to provide resource isolation instead of a hypervisor which completely emulates guest hardware. This reduces the start-up time and resource utilization of containers compared to virtual machines while still providing similar features.

Setup Docker

We can install Docker in Ubuntu with the following command:

```
sudo apt-get install docker.io
```

Running Docker requires the user to be an administrator or a part of the Docker group. So, add the current user to the docker group so that we can run Docker without sudo.

```
sudo usermod -a -G docker <username>
```

This setup uses LXC as its back-end instead of the default Docker back-end. This can be configured by adding Docker Options to the daemon at the end in the file `/etc/default/docker.io`

```
# Docker Upstart and SysVinit configuration file
# bind docker to this interface while starting
DOCKER_OPTS="-e lxc"
```

After this step, the Docker daemon needs to be restarted for the options to take effect. Please use the below command:

```
sudo service restart docker.io
```

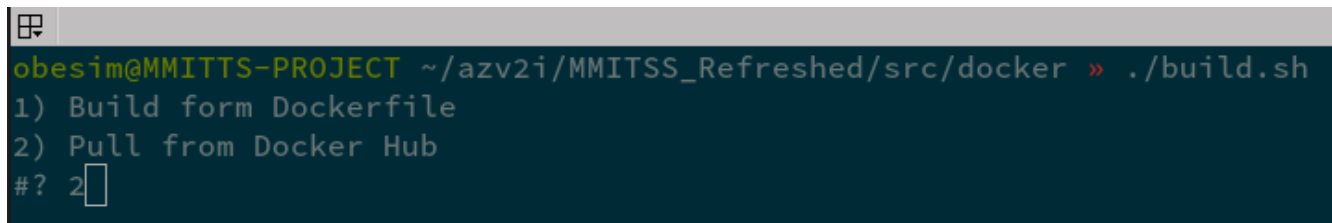
Docker Image

Docker containers which run on Linux are instances of a Docker image which contains all the software resources such as applications and libraries. If a Docker container is analogous to a process (a program in execution), then a Docker image is analogous to the program. The image should first be obtained before we can run the containers.

The image required to run MMITSS applications can either be built locally or pulled from Docker Hub. Docker Hub is a repository of images hosted by the company behind Docker. The image required by the MMITSS SILS package is already pushed to Docker Hub. Pulling from docker hub is just like downloading the image via Docker. The image is about 650MB in size.

To build locally, a recipe for the image is present in a file called Dockerfile. This file contains a sequence of commands to Docker to build the image locally. This method will pull all the packages required from their respective locations and compile some libraries from sources. The first time it is run, it takes about 15 minutes (depending on network bandwidth) to download the sources, compile the intermediate packages, install them and build the final image.

Both these options can be used from the script called `build.sh`. A screenshot is as follows:

A terminal window screenshot showing a user at the prompt 'obesim@MMITTS-PROJECT' in the directory '~/azv2i/MMITSS_Refreshed/src/docker'. The user has executed './build.sh', which displays a menu with two options: '1) Build form Dockerfile' and '2) Pull from Docker Hub'. The user has entered '#? 2' and the cursor is positioned after the number 2.

```
obesim@MMITTS-PROJECT ~/azv2i/MMITSS_Refreshed/src/docker » ./build.sh
1) Build form Dockerfile
2) Pull from Docker Hub
#? 2
```

The recommended way is to pull from Docker Hub as it avoids possibilities of compilation failures in third party applications and libraries. Also, the image is downloaded from a single location instead of depending on multiple sites.

CONFIGURING NETWORK

Docker creates Bridged Interfaces to provide network connectivity to its containers. These virtual interfaces are “attached” to a physical interface that is available on the PC. We will not use the default bridged interface created by Docker and hence need to create one.

A virtual interface bridged to a physical interface should be configured with a static IP that is in the same sub-net as the Windows PC with VISSIM running on it. To do this in Ubuntu for an Ethernet Interface, the file `/etc/network/interfaces` has to be modified as shown below.

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 10.254.56.8    # Static IP of LAN (eth0)
    netmask 255.255.255.0
    gateway 10.254.56.1

auto docker999          # Virtual Interface for docker
iface docker999 inet static
    address 10.254.56.8    # Same as IP of eth0
    netmask 255.255.255.0
    gateway 10.254.56.1
    bridge_ports eth0
    bridge_stp off
    bridge_fd 0
    bridge_maxwait 0
```

The first section is a loopback interface. The second section is where we configure a static IP to the Ethernet port (eth0) on the Linux PC. The third section is where we create a bridged interface to eth0 on the Linux PC. To make these changes take effect, run the following commands:

```
sudo ifdown eth0
sudo ifup eth0
sudo ifup docker999
ifconfig
```

If any of the command returned errors or if the last command did not show the interfaces docker999 and eth0 as configured, please reboot your PC.

MMITSS SILS PACKAGE

The MMITSS SILS package consists of the Docker setup directory structure as shown in the figure below. Each of the top level contents shall be explained.

```
obesim@MMITSS-PROJECT ~/azv2i/MMITSS_Refreshed/src/docker » tree -L 1
.
├── applications
├── build.sh
├── Dockerfile
├── intersections
├── libj2735-linux.a
├── prepare_apps.sh
├── README.md
├── run.sh
└── stop.sh
```

applications

This is the directory containing all the applications that are run inside a container. The file `applications/bootstrap.sh` is a script that runs all these applications in desired order when a Docker container starts.

```
applications
├── bootstrap.sh
├── mehdi_mprsolver_ack
├── MMITSS_MRP_EquippedVehicleTrajectoryAware_ASN
├── MMITSS_MRP_MAP_SPAT_Broadcast_ASN
├── MMITSS_MRP_PriorityRequestServer_ackasn
├── MMITSS_MRP_TrafficControllerInterface
├── MMITSS_OBE_MAP_SPAT_Receiver_ASN
├── MMITSS_OBE_PriorityRequestGenerator_ackasn
├── MMITSS_rsu_PerformanceObserver_TT
└── MMITSS_rsu_Signal_Control_visual
```

intersections

This directory contains different sub-directories as shown below.

```
intersections
├── test
├── rse205_Anthem
├── rse204_Memorial
├── rse203_Hastings
├── rse202_Meridian
└── rse201_Dedication
```

Each of them represents an intersection's configuration store. An example directory structure of `intersections/rse200_Gavilan` is as shown below:

```
intersections/rse200_Gavilan
├── config
├── nojournal
│   └── bin
│       ├── ActiveMAP.txt
│       ├── ActiveRNDf.txt
│       ├── ConfigInfo_GavilanPeak.txt
│       ├── ConfigInfo.txt
│       ├── Daysi_Gav_Reduced.nmap
│       ├── DSRC_Range.txt
│       ├── InLane_OutLane_Phase_Mapping.txt
│       ├── Intersection_maps.txt
│       ├── IPInfo.txt
│       ├── Lane_Movement_Mapping.txt
│       ├── Lane_No.txt
│       ├── Lane_Phase_Mapping.txt
│       ├── log
│       └── nmap_name.txt
```

The `config` file contains the IP that is to be assigned to the Docker Container. Make sure that this IP is in the same subnet as the host PC. Inside the container, the directory `/nojournal` is obtained by mounts the folder `intersections/<dir_name>/nojournal`. Applications running inside the container will read configurations and write logs to this directory. This directory can be accessed outside the container and is useful to check the logs of the running applications.

Scripts

run.sh

This script starts a container by reading the static IP from the file `intersections/<dir_name>/config`.

- This script takes one or more arguments.
- Each argument must be the path of the directory containing the config file and the `nojournal` directory.
- These paths must be below the current working directory where you execute the `run.sh`
- One container will be spawned for every directory argument given that they all have unique IP addresses.

The below screenshot shows the directories available and the containers spawned by the command.


```
obesim@MMITTS-PROJECT ~/azv2i/MMITSS_Refreshed/src/docker » ./run.sh intersect
ions/rse100 intersections/rse101 intersections/rse102 intersections/rse103
Creating a new Container with IP = 10.254.56.100 : de3c0bd6e5157c4e06602596bec
c33bee0d65e08df7cf45f11051581da37d9ee
Creating a new Container with IP = 10.254.56.101 : d2c25812a7430552efdef9faa0f
ec3b1fc0d81183533dcb0f8afe2a648561a35
Creating a new Container with IP = 10.254.56.102 : 9b02a4a10d043503880ae47844d
3b274d79fa498fec7809a3671ad82249c349a
Creating a new Container with IP = 10.254.56.103 : 34167b1cd745b506c8bbd1bce5a
830258c30719ed29b6680a0055222ae378b2e
```

stop.sh

This script stops a container. As with `run.sh`, it takes the directory paths as arguments and stops the containers that correspond to those paths.

build.sh

Script file to prepare the Docker image as [described earlier](#).

Dockerfile

Recipe file to prepare the Docker image as [described earlier](#).

prepare_apps.sh

This script builds applications from the MMITSS source repo and populates the applications directory.