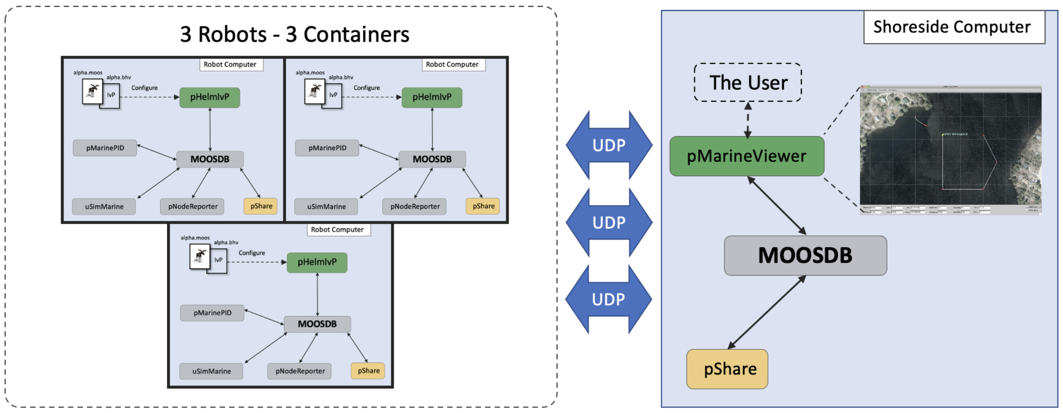
# Docker Lab 5 -
# Using Docker Compose to Launch Groups of Containers



**Jun 23rd, 2022**

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

**1 Overview and Objectives** **3**

**2 A Simple Dockerfile Compose File to Launch a Vehicle** **3**
2.1 Launching Shoreside and Vehicle Locally (review) . . . . . . . . 3
2.2 Launching Shoreside Locally, Vehicle with Docker Run (review) . . . . . . . . 3
2.3 Launching Shoreside Locally, Vehicle with Docker Compose (new) . . . . . . . . 4

**3 Launching Multiple Vehicles with Docker Compose** **4**
3.1 A First Version of a Multi-Vehicle Docker Compose . . . . . . . . 5
3.2 A More Flexible Version of a Multi-Vehicle Docker Compose . . . . . . . . 7

## 1 Overview and Objectives

In this lab we use Docker Compose to begin to automate the launching of docker containers. In the previous lab, only one container was launched in the uField Alpha mission, so it was easy to just launch the vehicle with the `docker run` command. When multiple `docker run` commands are involved in starting a multi-vehicle or multi-container mission, then hand-typing multiple `docker run` commands become cumbersome. Normally this can be fixed by writing a shell script to run a sequence of `docker run` commands. However, the Docker Compose utility offers some advantage over shell scripts for this task. In this lab we will get comfortable with Docker Compose, remaining in the context of the uField Alpha mission to launch multiple vehicles, each slightly different in configuration.

The goals of this lab is:

- Introduction to Docker Compose
- Use simple Docker Compose file to launch a single vehicle
- Use a more complex Docker Compose file to launch muliple vehicles

## 2 A Simple Dockerfile Compose File to Launch a Vehicle

In the previous lab, we launched the alpha mission by (a) launching a shoreside community, and (b) launching a vehicle community. In this section we progress through launching the uField Alpha mission:

- Launch the shoreside locally (native OS), and the vehicle locally.
- Launch the shoreside locally, the vehicle launched with `docker run`
- Launch the shoreside locally, the vehicle launched with `docker compose`

### 2.1 Launching Shoreside and Vehicle Locally (review)

If both were run natively, it would look like:

```
# In Terminal 1:
$ ./launch_shoreside.sh
```

```
# In Terminal 2:
$ ./launch_vehicle.sh
```

In this case both the vehicle and shoreside have a ”localhost” IP address. All parameters take on default values and no command line arguments are needed.

### 2.2 Launching Shoreside Locally, Vehicle with Docker Run (review)

If launching the vehicle using our Docker image, the vehicle is launched instead with:

```
# In Terminal 1:
$ ./launch_shoreside.sh
```

```
# In Terminal 2:
$ docker run -it -p 9201:9201/udp --rm --env SHORE_IP=192.168.7.5 mikerbenj/moos-ivp-ualpha:latest
```

In the Docker version, the vehicle running in the container cannot reach the shoreside community as ”localhost”. The container is instead given the IP address of the local machine on the local router, as the SHORE_IP location.

### 2.3 Launching Shoreside Locally, Vehicle with Docker Compose (new)

In this step, the vehicle is again launched with a container, but we use docker compose instead:

```
# In Terminal 1:
$ ./launch_shoreside.sh
```

```
# In Terminal 2:
$ docker-compose up
```

The above docker-compose command presumes there is a docker-compose.yml file in the folder where docker-compose was run. The contents of this file look like:

```
# docker-compose.yml file
version: '3.7'
services:
  abe:
    image: "mikerbenj/moos-ivp-ualpha:latest"
    ports:
      - "9201:9201/udp"
    environment:
      - SHORE_IP=192.168.1.224
      - TIME_WARP=2
      - CONFIRM=no
```

Much of the information passed previously to the docker run command is now set in the docker-compose.yml file. When launching with docker compose, we really don’t want to prompted for confirmation before continuing with the launch, so we set CONFIRM=no.

## 3 Launching Multiple Vehicles with Docker Compose

The uField Alpha mission is normally used for testing a wide variety of single-vehicle-plus-shoreside missions. It does also support connecting any number of additional vehicles. Normally subsequent vehicles are given different unique names, starting positions, ip addresses, and pShare listen ports. These are normally passed in as command line arguments to the launch_vehicle.sh script. When only launching one vehicle as we have above, the default arguments suffice. Here we describe how to launch three vehicles in the uField Alpha mission, conveyed in Figure 1

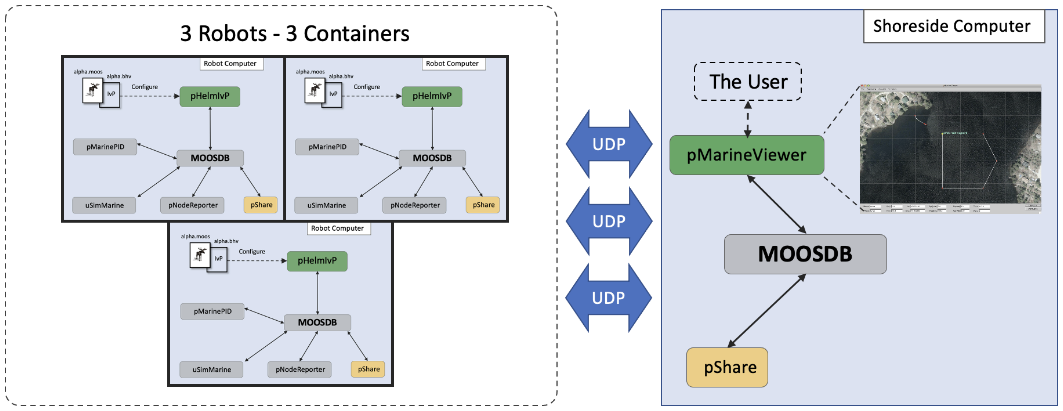


Figure 1: The uField Alpha mission is run with four simulated vehicle each running in their own container, launched with `docker-compose`.

## 3.1 A First Version of a Multi-Vehicle Docker Compose

Below is the `docker-compose.yml` file for bringing up all three vehicles in three separate containers.

```
version: '3.7'
services:
  abe:
    image: "mikerbenj/moos-ivp-ualpha:latest"
    ports:
      - "9201:9201/udp"
    environment:
      - SHORE_IP=192.168.1.224
      - TIME_WARP=2
      - CONFIRM=no
  ben:
    image: "mikerbenj/moos-ivp-ualpha:latest"
    ports:
      - "9202:9202/udp"
    environment:
      - SHORE_IP=192.168.1.224
      - TIME_WARP=2
      - VNAME=ben
      - CONFIRM=no
      - INDEX=2
      - START_POS=20,0
  cal:
    image: "mikerbenj/moos-ivp-ualpha:latest"
    ports:
      - "9203:9203/udp"
    environment:
      - SHORE_IP=192.168.1.224
      - TIME_WARP=2
      - VNAME=cal
      - CONFIRM=no
      - INDEX=3
      - START_POS=40,0
```

Just as a reminder, each of these containers, one for vehicles abe, ben, and cal, are launched from the same image `mikerbenj/moos-ivp-ualpha:latest`, with an `entrypoint.sh` script:

```
#!/bin/bash
cd /home/moos/moos-ivp/ivp/missions/ufld_alpha
./launch_vehicle.sh
```

If vehicle `cal` were launched on the command line in our local operating system, the `launch_vehicle.sh` script would probably look like:

```
$ ./launch_vehicle.sh --shore=192.168.1.224 --vname=cal --noconfirm --index=3 \
--startpos=40,0 2
```

These command line arguments are replaced by setting environment variables. Again, leaving Docker aside for a moment, the above launch could also be accomplished with:

```
$ export SHORE_IP=192.168.1.224
$ export TIME_WARP=2
$ export VNAME=cal
$ export CONFIRM=no
$ export INDEX=3
$ export START_POS=40,0
$ ./launch_vehicle.sh
```

Or, if we don't like setting environment variables with persistent scope that are only needed for a one-time use, we could do this instead:

```
$ SHORE_IP=192.168.1.224 TIME_WARP=2 VNAME=cal CONFIRM=no INDEX=3 \
START_POS=40,0 ./launch_vehicle.sh
```

And, final review point, these environment variables are handled in the `launch_vehicle.sh` script because they are initialized from the incoming value of the environment variable if it is set. For example, inside the `launch_vehicle.sh` script one will find:

```
START_POS=${START_POS:-0,0}
```

The `START_POS` variable will default to `(0,0)` if the process that called `launch_vehicle.sh` did not have the environment variable `START_POS` set. If it *is* set in the calling process, it will be used instead as the default value for `START_POS`. If the caller also invokes the `--startpos=x,y` command line argument, the command line argument overwrites any other value.

## 3.2 A More Flexible Version of a Multi-Vehicle Docker Compose

In the `docker-compose.yml` file above, there are a couple parameters for each vehicle that may change often between runs. The time warp, and the IP address of the shoreside computer. When or if these values change, they must be edited for each service/container/vehicle block. We handle this by setting them from an environment variable passed in when launching `docker-compose`. Our new file looks like:

```
version: '3.7'
services:
  abe:
    image: "mikerbenj/moos-ivp-ualpha:latest"
    ports:
      - "9201:9201/udp"
    environment:
      - SHORE_IP=${SHORE_IP:-192.168.1.224}     <--change
      - TIME_WARP=${TIME_WARP:-2}               <--change
      - CONFIRM=no
  ben:
    image: "mikerbenj/moos-ivp-ualpha:latest"
    ports:
      - "9202:9202/udp"
    environment:
      - SHORE_IP=${SHORE_IP:-192.168.1.224}     <--change
      - TIME_WARP=${TIME_WARP:-2}               <--change
      - VNAME=ben
      - CONFIRM=no
      - INDEX=2
      - START_POS=20,0
  cal:
    image: "mikerbenj/moos-ivp-ualpha:latest"
    ports:
      - "9203:9203/udp"
    environment:
      - SHORE_IP=${SHORE_IP:-192.168.1.224}     <--change
      - TIME_WARP=${TIME_WARP:-2}               <--change
      - VNAME=cal
      - CONFIRM=no
      - INDEX=3
      - START_POS=40,0
```

When running `docker-compose` on the command line, the values are passed in like this:

```
$ SHORE_IP=192.168.7.5 TIME_WARP=8 docker-compose up
```

## URL References

[1] Mouat, Adrian. Using Docker: Developing and Deploying Software with Containers (Kindle Locations 488-490). O'Reilly Media. Kindle Edition.