# 3. Random movement

This lesson takes MoveIT simulation as an example. If you need to set up the real machine and simulation to be synchronized, please see the lesson [02, MoveIt Precautions and Controlling the Real Machine]. ! ! ! be safe! ! !

The effect demonstration is a virtual machine and other main control running conditions (related to the main control performance, depending on the actual situation).

## 3.1. Start

```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
```

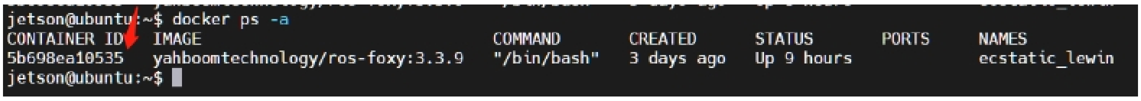
Start MoveIT

```
roslaunch arm_moveit_demo x3plus_moveit_demo.launch sim:=true
```

**<PI5 needs to open another terminal to enter the same docker container**

1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

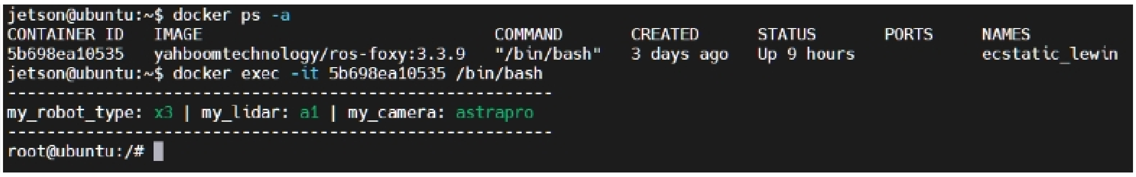
```
docker ps -a
```



```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND       CREATED      STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9    "/bin/bash"   3 days ago   Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$
```

2. Now enter the docker container in the newly opened terminal:

```
docker exec -it 5b698ea10535 /bin/bash
```



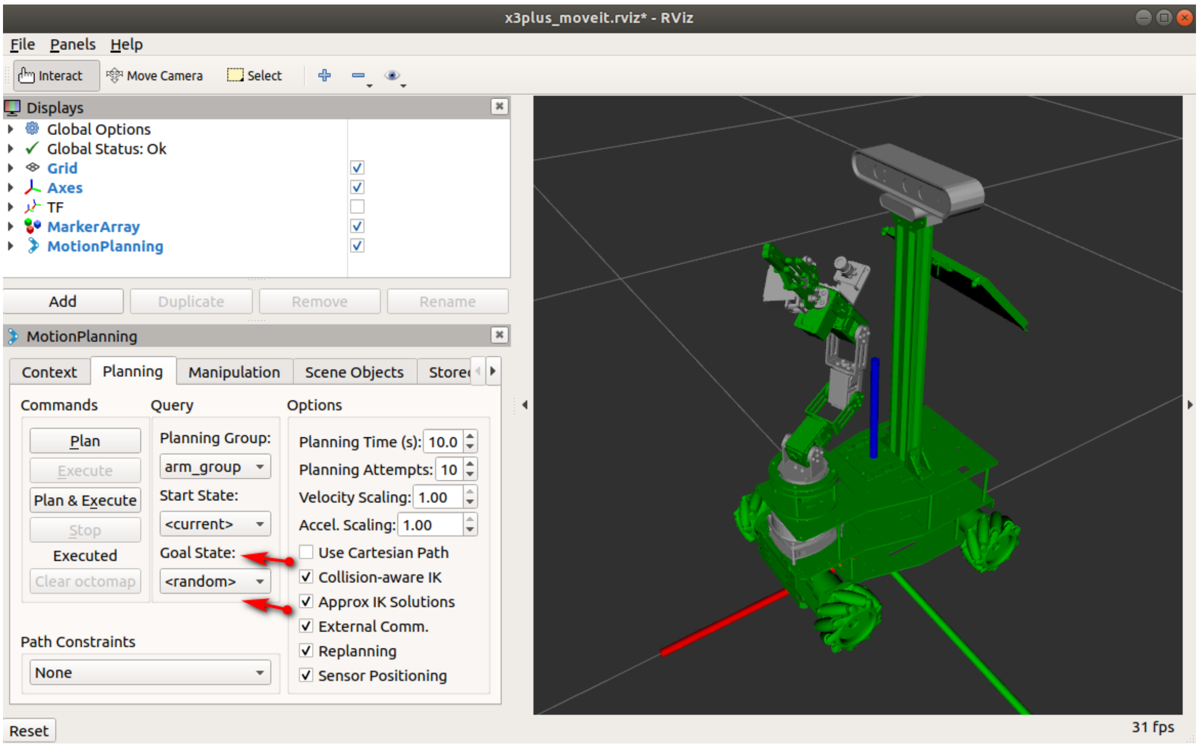
```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND       CREATED      STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9    "/bin/bash"   3 days ago   Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$ docker exec -it 5b698ea10535 /bin/bash
-------------------------------------------------------
my_robot_type: x3 | my_lidar: a1 | my_camera: astrapro
-------------------------------------------------------
root@ubuntu:/#
```

After successfully entering the container, you can open countless terminals to enter the container.

Start random motion node (choose one of two)

```
rosrun arm_moveit_demo 01_random_move.py # python
rosrun arm_moveit_demo 01_random_move # C++
```

The renderings are as follows



- Global Options
- Global Status: Ok
- Grid ☑
- Axes ☑
- TF □
- MarkerArray ☑
- MotionPlanning ☑

Executed

Planning Group: arm_group

Start State: <current>

Goal State: <random>

Path Constraints: None

Planning Time (s): 10.0

Planning Attempts: 10

Velocity Scaling: 1.00

Accel. Scaling: 1.00

- □ Use Cartesian Path
- ☑ Collision-aware IK
- ☑ Approx IK Solutions
- ☑ External Comm.
- ☑ Replanning
- ☑ Sensor Positioning

The source code implementation effect is the same as the MoveIT interface [MotionPlanning--->Planning--->Goal State (random)].

# 3.2, python source code

Import header file

```
import rospy
from time import sleep
from moveit_commander.move_group import MoveGroupCommander
```

Initialize nodes and create planning group instances

```
#Initialize node
     rospy.init_node("yahboomcar_random_move")
     # Initialize the robot arm planning group
     yahboomcar = MoveGroupCommander("arm_group")
```

Set planning parameters

```
# When motion planning fails, re-planning is allowed
     yahboomcar.allow_replanning(True)
     # Set planning time
     yahboomcar.set_planning_time(5)
     # Number of attempts to plan
     yahboomcar.set_num_planning_attempts(10)
     #Set the allowed target position error
     yahboomcar.set_goal_position_tolerance(0.01)
     #Set the allowed target attitude error
     yahboomcar.set_goal_orientation_tolerance(0.01)
     # Set the allowable target error
     yahboomcar.set_goal_tolerance(0.01)
```

```
    # Set maximum speed
    yahboomcar.set_max_velocity_scaling_factor(1.0)
    # Set maximum acceleration
    yahboomcar.set_max_acceleration_scaling_factor(1.0)
```

Set random target points in a loop

```
while not rospy.is_shutdown():
        #Set random target point
        yahboomcar.set_random_target()
        # Start exercising
        yahboomcar.go()
        sleep(0.5)
```

## 3.3, C++ source code

Import header file

```
#include <iostream>
#include "ros/ros.h"
#include <moveit/move_group_interface/move_group_interface.h>
```

Create nodes and planning groups

```
ros::init(argc, argv, "yahboomcar_random_move_cpp");
    ros::NodeHandle n;
ros::AsyncSpinner spinner(1);
spinner.start();
    moveit::planning_interface::MoveGroupInterface yahboomcar("arm_group");
```

Set planning parameters and initial position

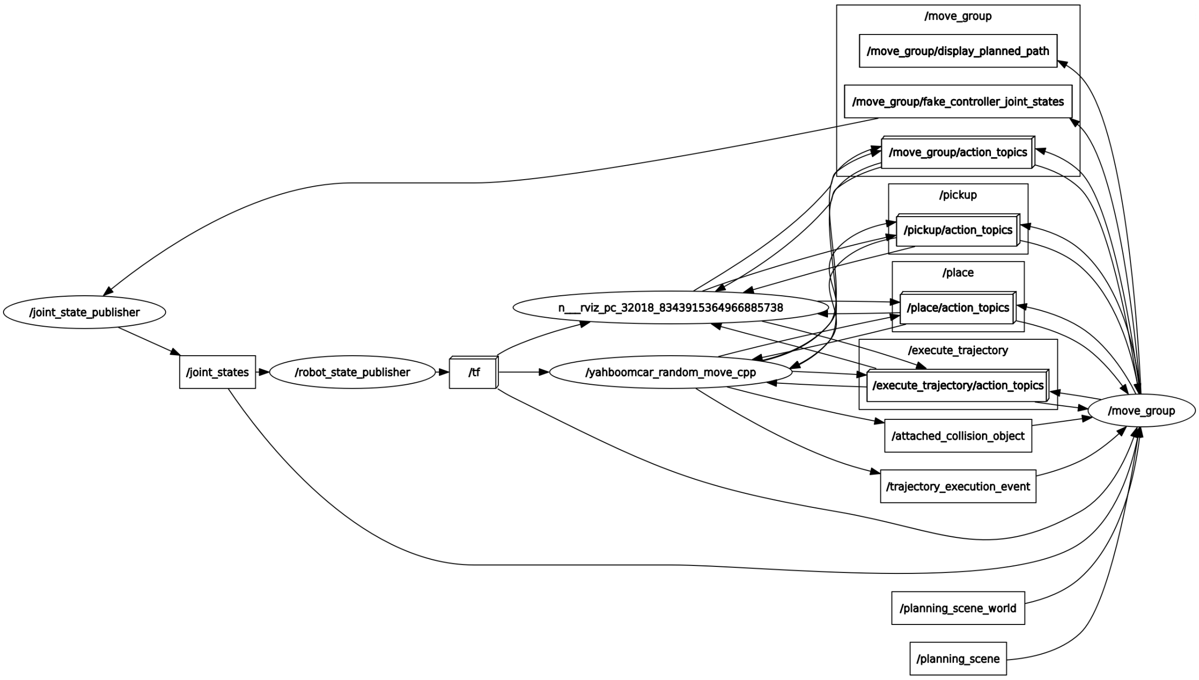
```
    //Set the maximum speed
    yahboomcar.setMaxVelocityScalingFactor(1.0);
    //Set the maximum acceleration
    yahboomcar.setMaxAccelerationScalingFactor(1.0);
    //Set target point
    yahboomcar.setNamedTarget("down");
    //Start moving
    yahboomcar.move();
    sleep(0.1);
```

Set random target points in a loop

```
    while (!ros::isShuttingDown()){
    //Set random target point
    yahboomcar.setRandomTarget();
    yahboomcar.move();
    sleep(0.5);
    }
```

# 3.4, Node graph

Take C++ node as an example


/move_group
/move_group/display_planned_path
/move_group/fake_controller_joint_states
/move_group/action_topics
/pickup
/pickup/action_topics
/place
/place/action_topics
/execute_trajectory
/execute_trajectory/action_topics
/joint_state_publisher
n___rviz_pc_32018_8343915364966885738
/joint_states
/robot_state_publisher
/tf
/yahboomcar_random_move_cpp
/move_group
/attached_collision_object
/trajectory_execution_event
/planning_scene_world
/planning_scene