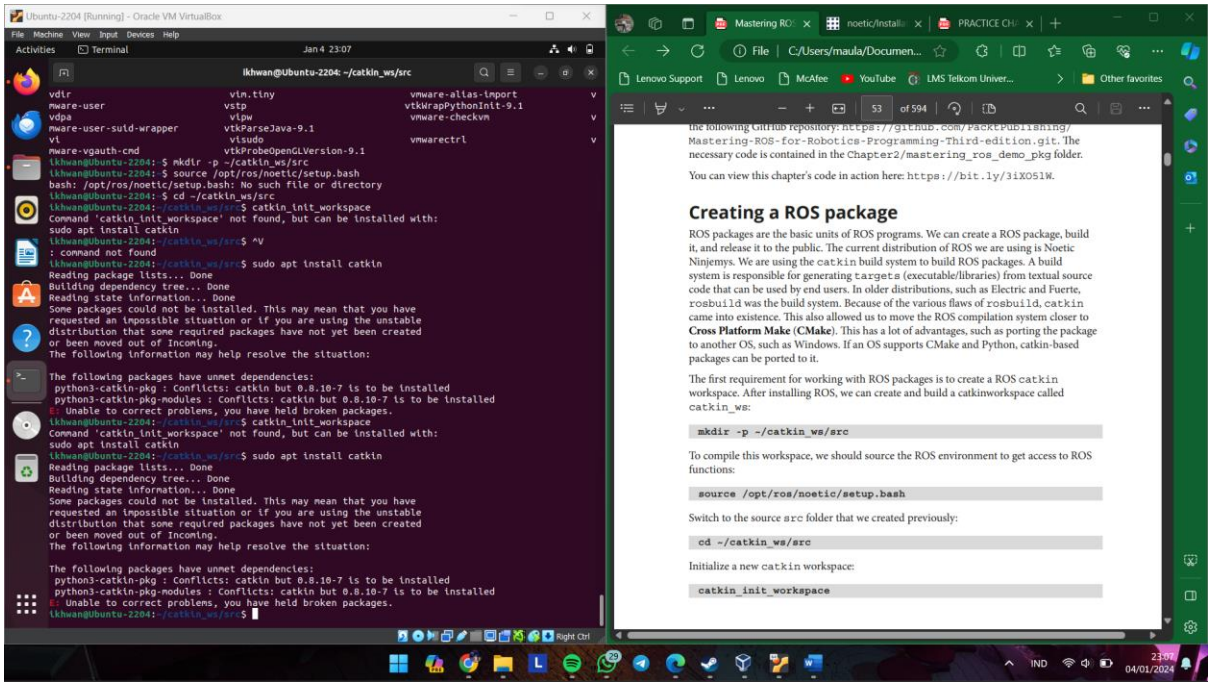
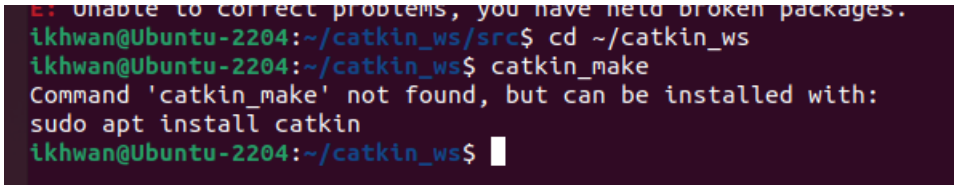
MUHAMMAD IKHWAN MAULANA

1103203219

PRACTICE CHAPTER 2

1. Creating a ROS package





Terjadi error pada saat ingin install catkin sehingga program selanjutnya tidak bisa dijalankan

2. Creating ROS nodes

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

int main(int argc, char **argv) {
    ros::init(argc, argv,"demo_topic_publisher");
    ros::NodeHandle node_obj;
    ros::Publisher number_publisher =  node_obj.advertise<std_
msgs::Int32>("/numbers", 10);
    ros::Rate loop_rate(10);
    int number_count = 0;
    while ( ros::ok() ) {
        std_msgs::Int32 msg;
        msg.data = number_count;
        ROS_INFO("%d",msg.data);
        number_publisher.publish(msg);
        loop_rate.sleep();
        ++number_count;
    }
    return 0;
}
```

Here is the definition of the subscriber node:

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

void number_callback(const std_msgs::Int32::ConstPtr& msg) {
   ROS_INFO("Received [%d]",msg->data);
}

int main(int argc, char **argv) {
   ros::init(argc, argv,"demo_topic_subscriber");
   ros::NodeHandle node_obj;
   ros::Subscriber number_subscriber = node_obj.subscribe("/
numbers",10,number_callback);
   ros::spin();
   return 0;
}
```

3. Building the nodes

```
include_directories(
   include
   ${catkin_INCLUDE_DIRS}
)
#This will create executables of the nodes
add_executable(demo_topic_publisher src/demo_topic_publisher.
cpp)
add_executable(demo_topic_subscriber src/demo_topic_subscriber.
cpp)

#This will link executables to the appropriate libraries
target_link_libraries(demo_topic_publisher ${catkin_LIBRARIES})
target_link_libraries(demo_topic_subscriber ${catkin_
LIBRARIES})
```

4. Adding custom .msg and .srv files

Edit the current `CMakeLists.txt` file and add the `message_generation` line, as follows:

```
find_package(catkin REQUIRED COMPONENTS
 roscpp
 rospy
 message_generation
)
```

Uncomment the following line and add the custom message file:

```
add_message_files(
   FILES
   demo_msg.msg
)
## Generate added messages and services with any dependencies
listed here
generate_messages(


)
```

We can test the message by adding the following lines of code to `CMakeLists.txt`:

```
add_executable(demo_msg_publisher src/demo_msg_publisher.cpp)
add_executable(demo_msg_subscriber src/demo_msg_subscriber.cpp)

add_dependencies(demo_msg_publisher mastering_ros_demo_pkg_
generate_messages_cpp)
add_dependencies(demo_msg_subscriber mastering_ros_demo_pkg_
generate_messages_cpp)

target_link_libraries(demo_msg_publisher ${catkin_LIBRARIES})
target_link_libraries(demo_msg_subscriber ${catkin_LIBRARIES})
```

5. Working with ROS services

`demo_service_server.cpp` is the server, and its definition is as follows:

```
#include "ros/ros.h"
#include "mastering_ros_demo_pkg/demo_srv.h"
#include <iostream>
#include <sstream>
using namespace std;

bool demo_service_callback(mastering_ros_demo_pkg::demo_
srv::Request &req,
     mastering_ros_demo_pkg::demo_srv::Response &res) {
    ss << "Received Here";
    ROS_INFO("From Client [%s], Server says [%s]",req.in.c_
str(),res.out.c_str());
    return true;
}

int main(int argc, char **argv) {
    ros::init(argc, argv, "demo_service_server");
    ros::NodeHandle n;
    ros::ServiceServer service = n.advertiseService("demo_
service", demo_service_callback);
    ROS_INFO("Ready to receive from client.");
    ros::spin();
    return 0;
}
```

Next, let's see how `demo_service_client.cpp` works. Here is the definition of this code:

```
#include "ros/ros.h"
#include <iostream>
#include "mastering_ros_demo_pkg/demo_srv.h"
#include <iostream>
#include <sstream>
using namespace std;

int main(int argc, char **argv) {
    ros::init(argc, argv, "demo_service_client");
    ros::NodeHandle n;
    ros::Rate loop_rate(10);
    ros::ServiceClient client = n.serviceClient<mastering_ros_
demo_pkg::demo_srv>("demo_service");
    while (ros::ok()) {
        mastering_ros_demo_pkg::demo_srv srv;
        ss << "Sending from Here";
        srv.request.in = ss.str();
        if (client.call(srv)) {
            ROS_INFO("From Client [%s], Server says [%s]",srv.
request.in.c_str(),srv.response.out.c_str());
        } else {
            ROS_ERROR("Failed to call service");
            return 1;
        }
        ros::spinOnce();
        loop_rate.sleep();
    }
    return 0;
}
```

6. Building the ROS action server and client

We must include the `Boost` library in `CMakeLists.txt` to build these nodes. Also, we must add the action files that we wrote for this example. We should pass `actionlib`, `actionlib_msgs`, and `message_generation` in `find_package()`:

```
find_package(catkin REQUIRED COMPONENTS
 roscpp
 std_msgs
 actionlib
 actionlib_msgs
 message_generation
)
```

We should also add `Boost` as a system dependency:

```
## System dependencies are found with CMake's conventions
find_package(Boost REQUIRED COMPONENTS system)
## Generate actions in the 'action' folder
 add_action_files(
  FILES
  Demo_action.action
 )
```

Then, we must add `actionlib_msgs` to `generate_messages()`:

```
## Generate added messages and services with any dependencies
listed here
 generate_messages(
  DEPENDENCIES
```

```
  std_msgs
  actionlib_msgs
 )
catkin_package(
 CATKIN_DEPENDS roscpp rospy std_msgs actionlib actionlib_msgs
message_runtime
)

include_directories(
 include
 ${catkin_INCLUDE_DIRS}
 ${Boost_INCLUDE_DIRS}
)
```

Finally, we can define the executable that's generated after the compilation of this node, along with its dependencies and linked libraries:

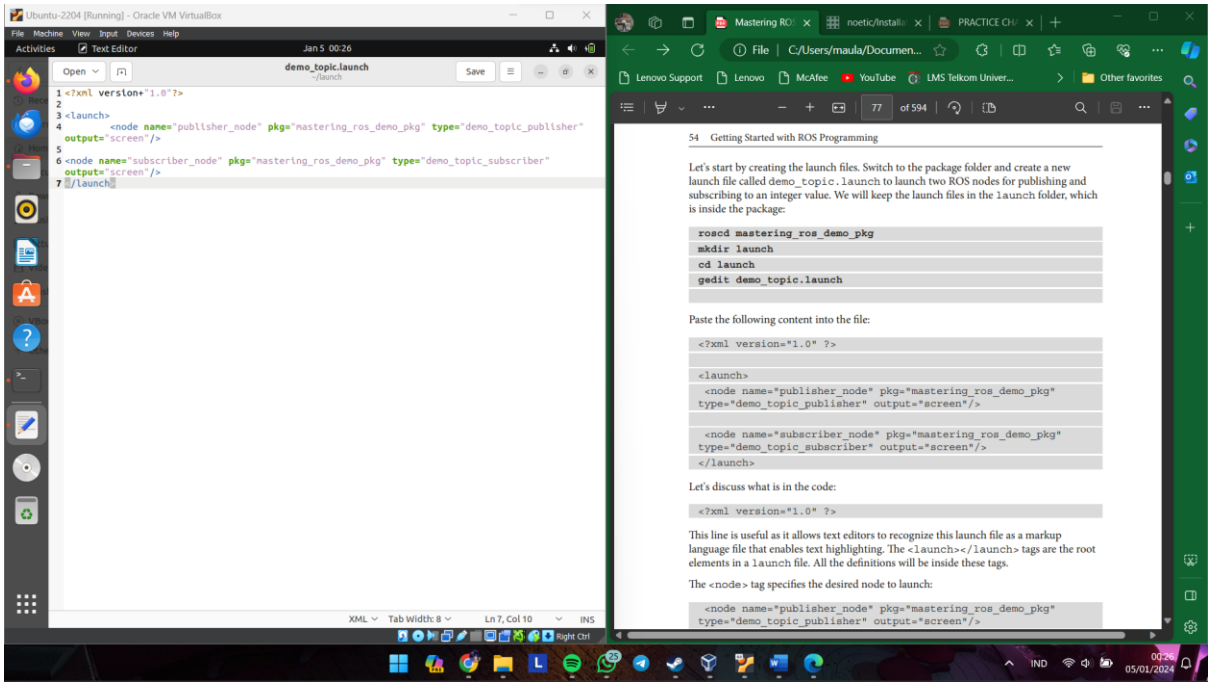
```
##Building action server and action client

add_executable(demo_action_server src/demo_action_server.cpp)
add_executable(demo_action_client src/demo_action_client.cpp)

add_dependencies(demo_action_server mastering_ros_demo_pkg_
generate_messages_cpp)
add_dependencies(demo_action_client mastering_ros_demo_pkg_
generate_messages_cpp)

target_link_libraries(demo_action_server ${catkin_LIBRARIES} )
target_link_libraries(demo_action_client ${catkin_LIBRARIES})
```

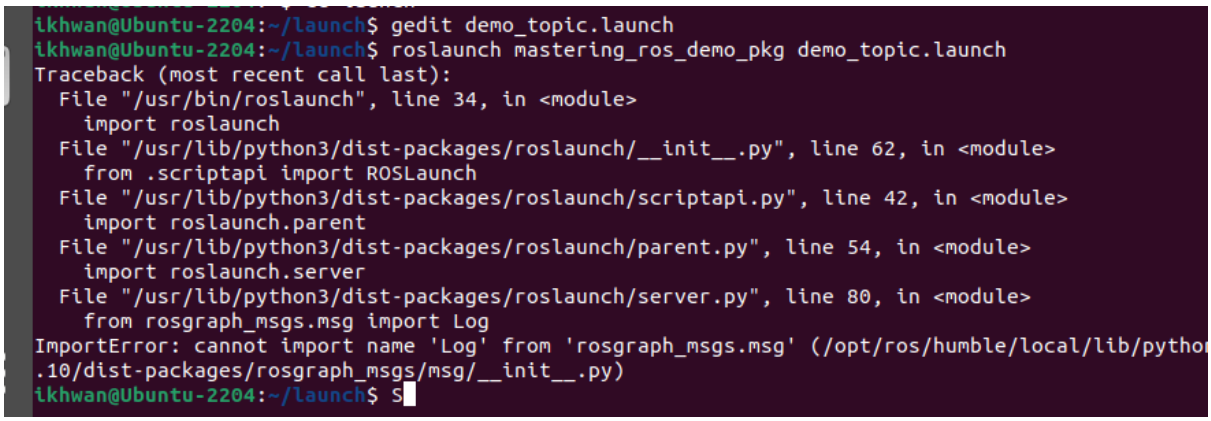
```
ikhwan@Ubuntu-2204:~$ rosrun mastering_ros_demo_pkg demo_action_server
Command 'rosrun' not found, but can be installed with:
sudo apt install rosbash
ikhwan@Ubuntu-2204:~$ sudo apt install rosbash
[sudo] password for ikhwan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help resolve the situation:

The following packages have unmet dependencies:
 python3-catkin-pkg : Conflicts: catkin but 0.8.10-7 is to be installed
 python3-catkin-pkg-modules : Conflicts: catkin but 0.8.10-7 is to be installed
E: Unable to correct problems, you have held broken packages.
ikhwan@Ubuntu-2204:~$ S
```

Saat mencoba perintah, tidak bisa

7. Creating launch files

```
roscd mastering_ros_demo_pkg
mkdir launch
cd launch
gedit demo_topic.launch
```





Perintah tidak keluar