ROS
(Robot Operating System)

2017-03-14

Table of Contents

What is ROS?

ROS is an open source, meta-operating system for your robot.

ROS provides the services you would expect from an operating system such as :

- Hardware abstraction
- Low-level device control
- Implementation of commonly-used functionality
- Message-passing between processes
- Package management.

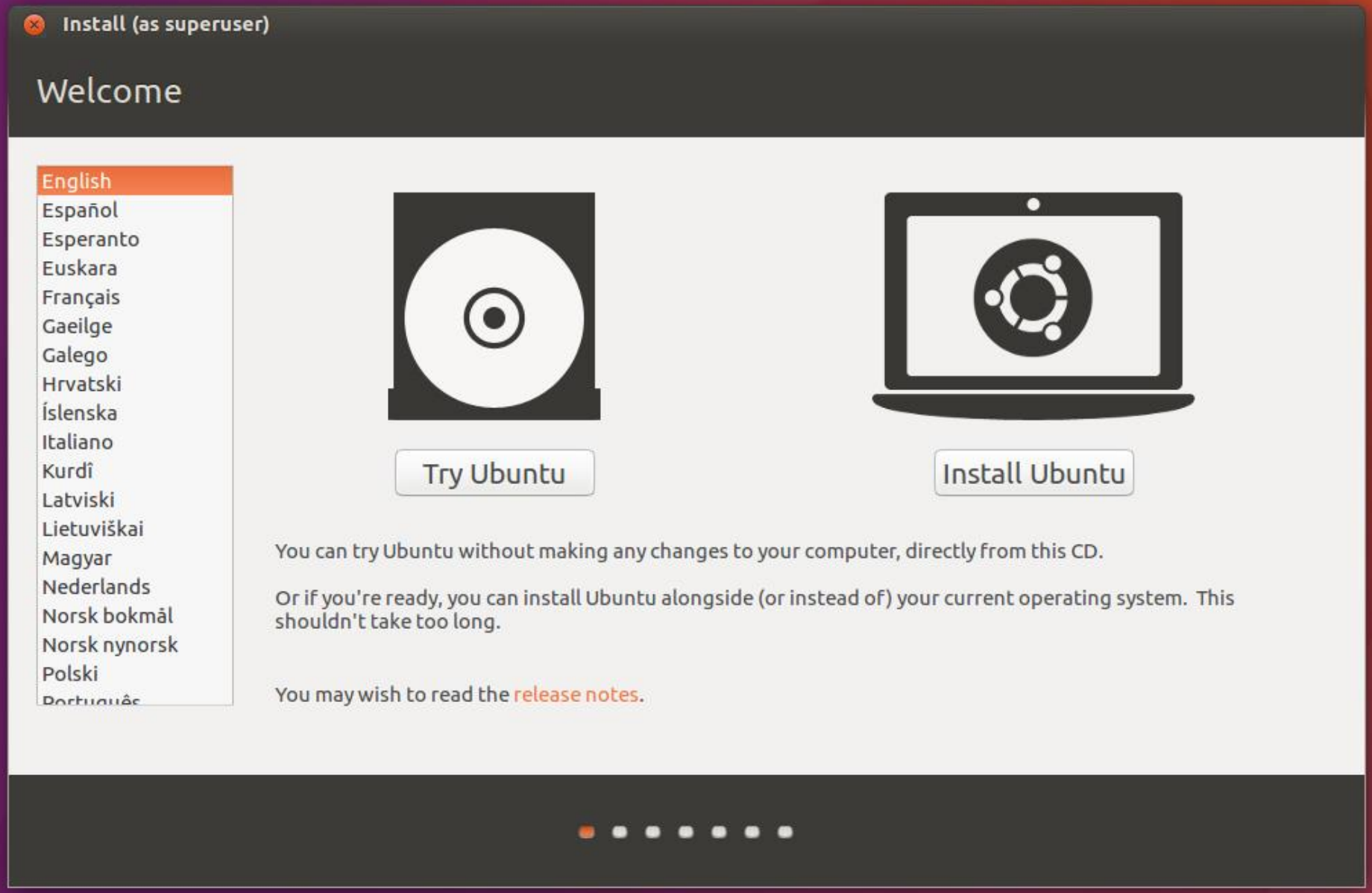
It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

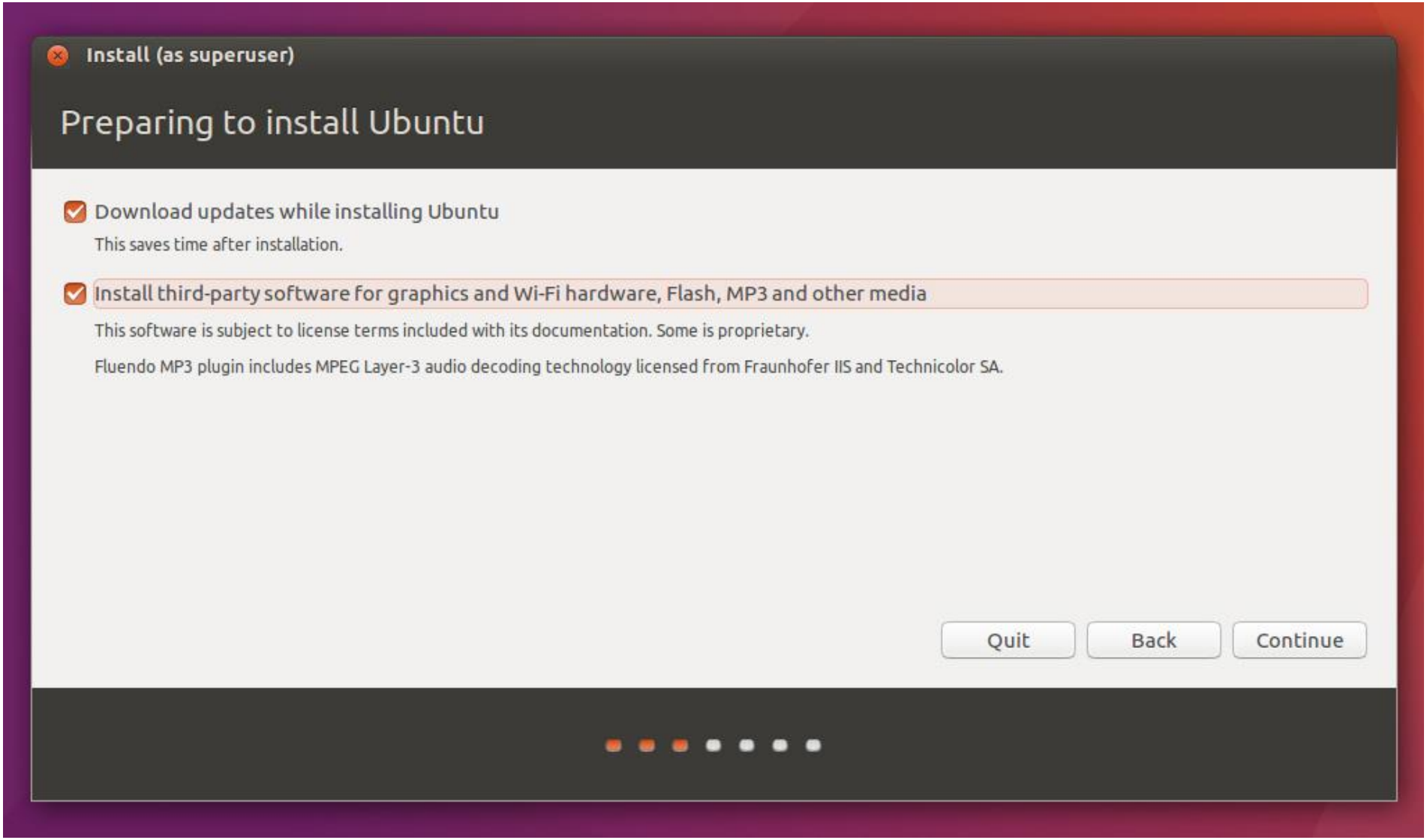
Installing Ubuntu

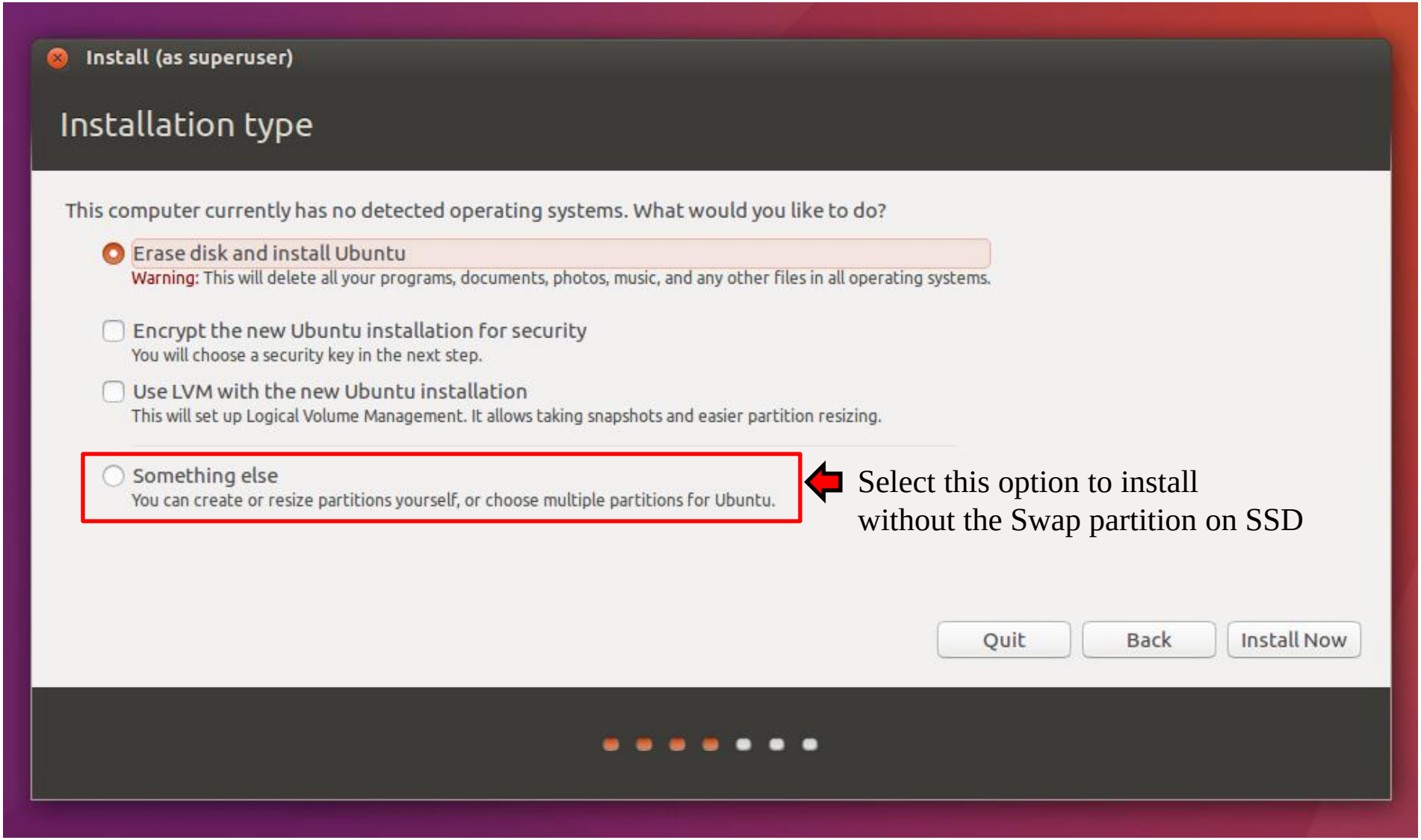
- Preparing Installation
  - USB flash device (>2GB)
  - Download Ubuntu 16.04.2(Download Link)
  - Create a bootable USB(Windows)
  - Create a bootable USB(Ubuntu)

Installing Ubuntu

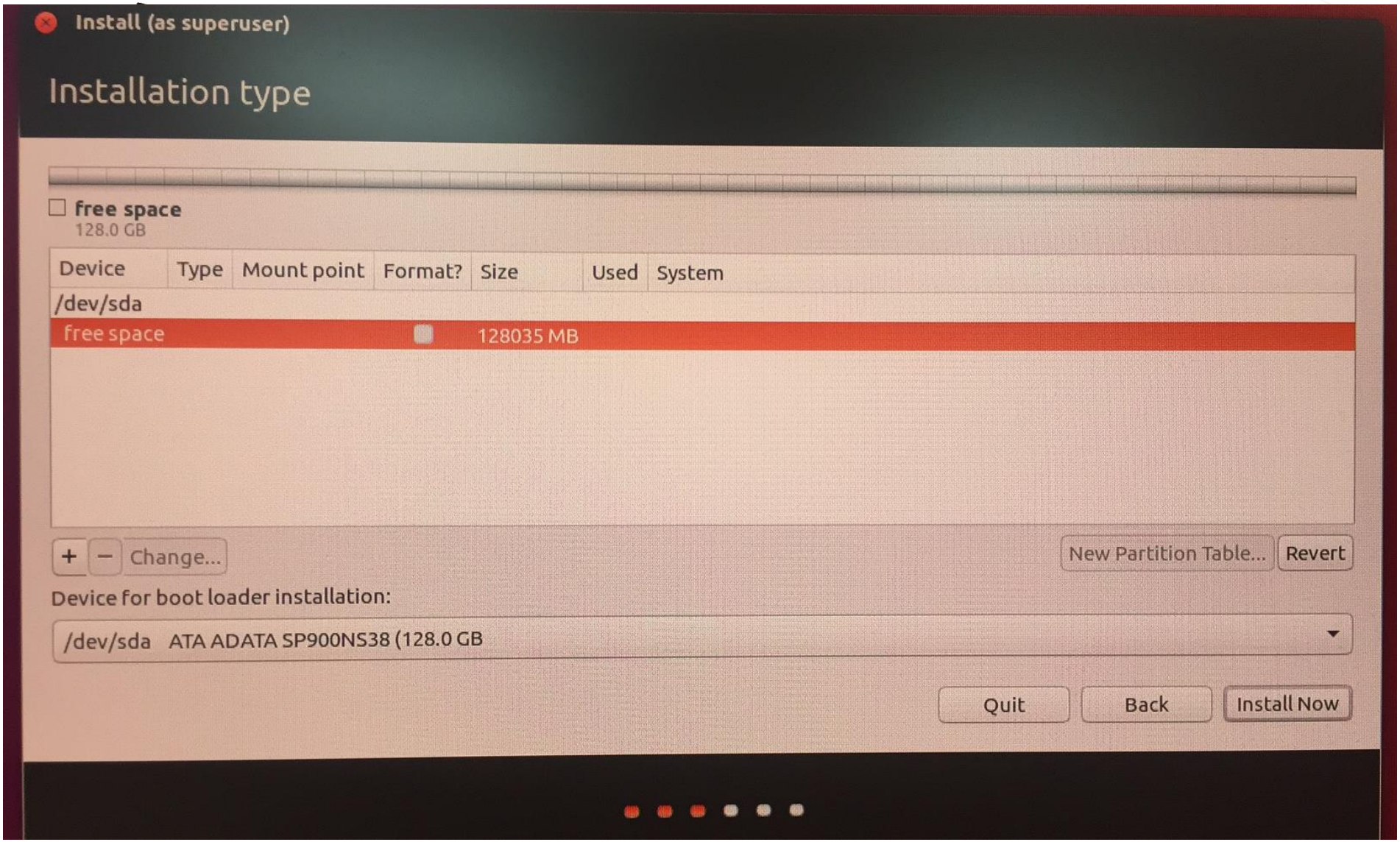
- Plug in the bootable USB device and turn on the PC.
- If the PC doesn’t automatically boot from USB, enter the BIOS setting to enable USB boot option.
- Check UEFI, AHCI option from BIOS setting

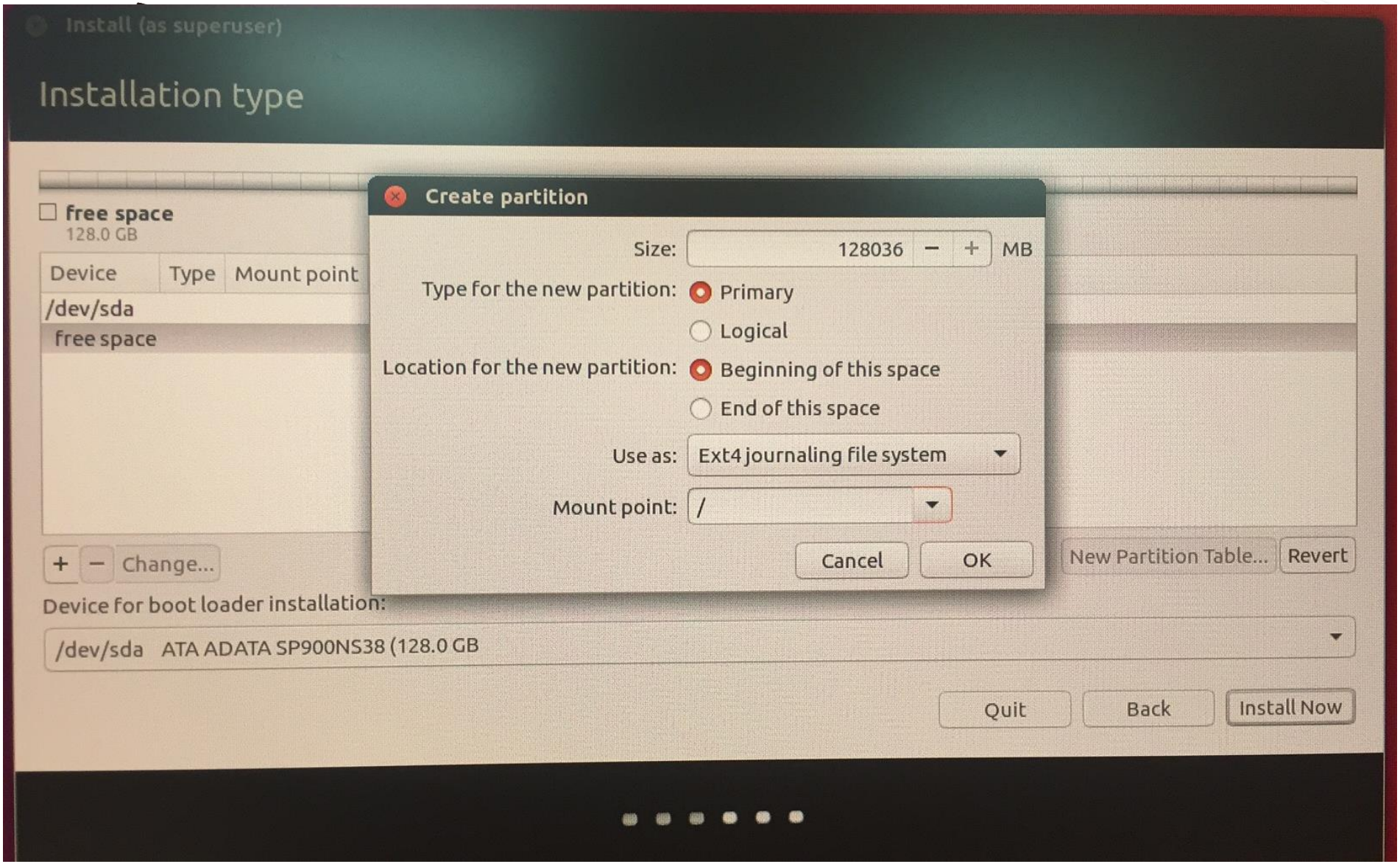

Install (as superuser)
Welcome
English
Español
Esperanto
Euskara
Français
Gaeilge
Galego
Hrvatski
Íslenska
Italiano
Kurdî
Latviski
Lietuviškai
Magyar
Nederlands
Norsk bokmål
Norsk nynorsk
Polski
Português
Try Ubuntu
Install Ubuntu
You can try Ubuntu without making any changes to your computer, directly from this CD.
Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.
You may wish to read the release notes.

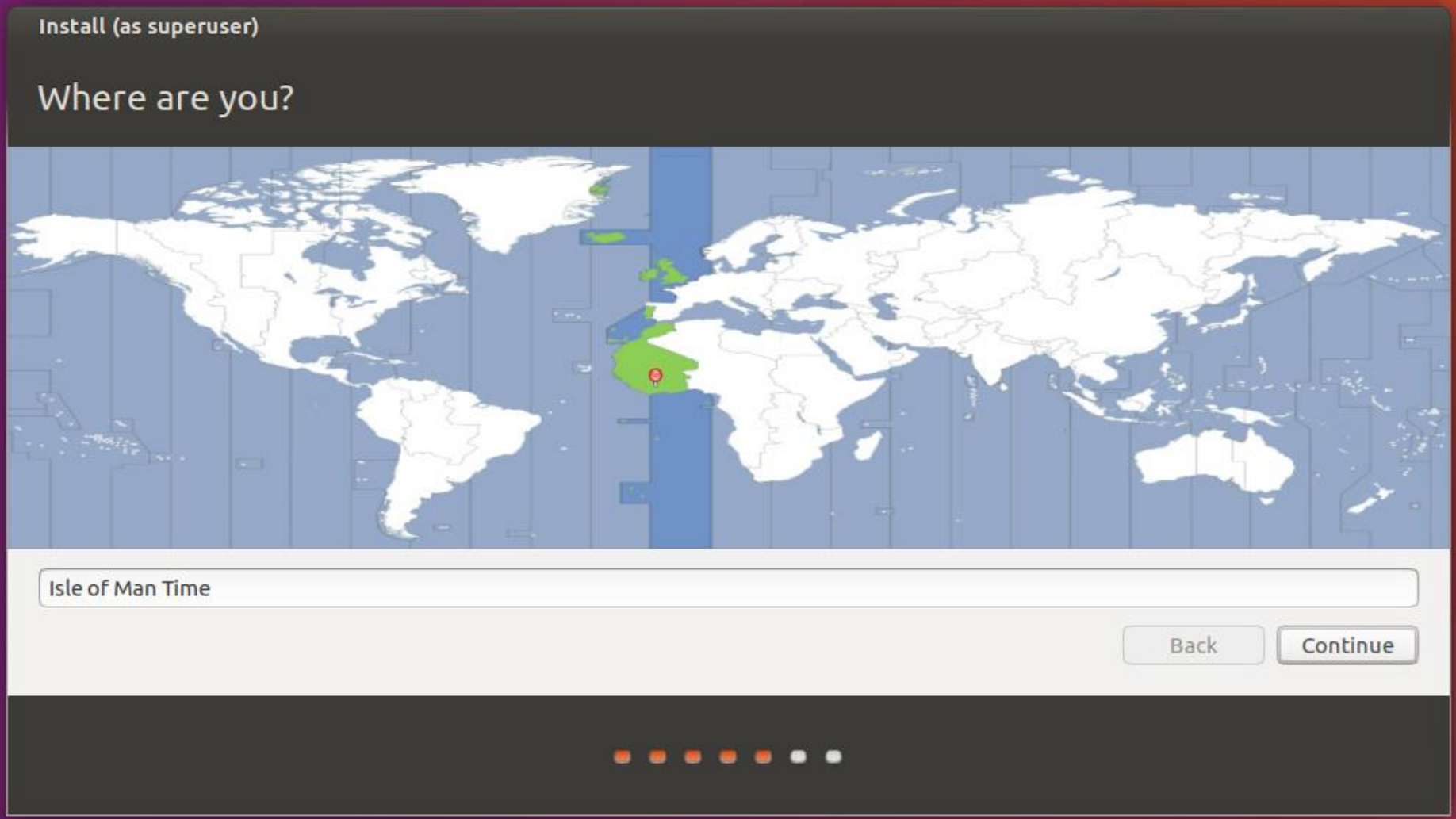

Install (as superuser)
Preparing to install Ubuntu
Download updates while installing Ubuntu
This saves time after installation.
Install third-party software for graphics and Wi-Fi hardware, Flash, MP3 and other media
This software is subject to license terms included with its documentation. Some is proprietary.
Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.
Quit
Back
Continue



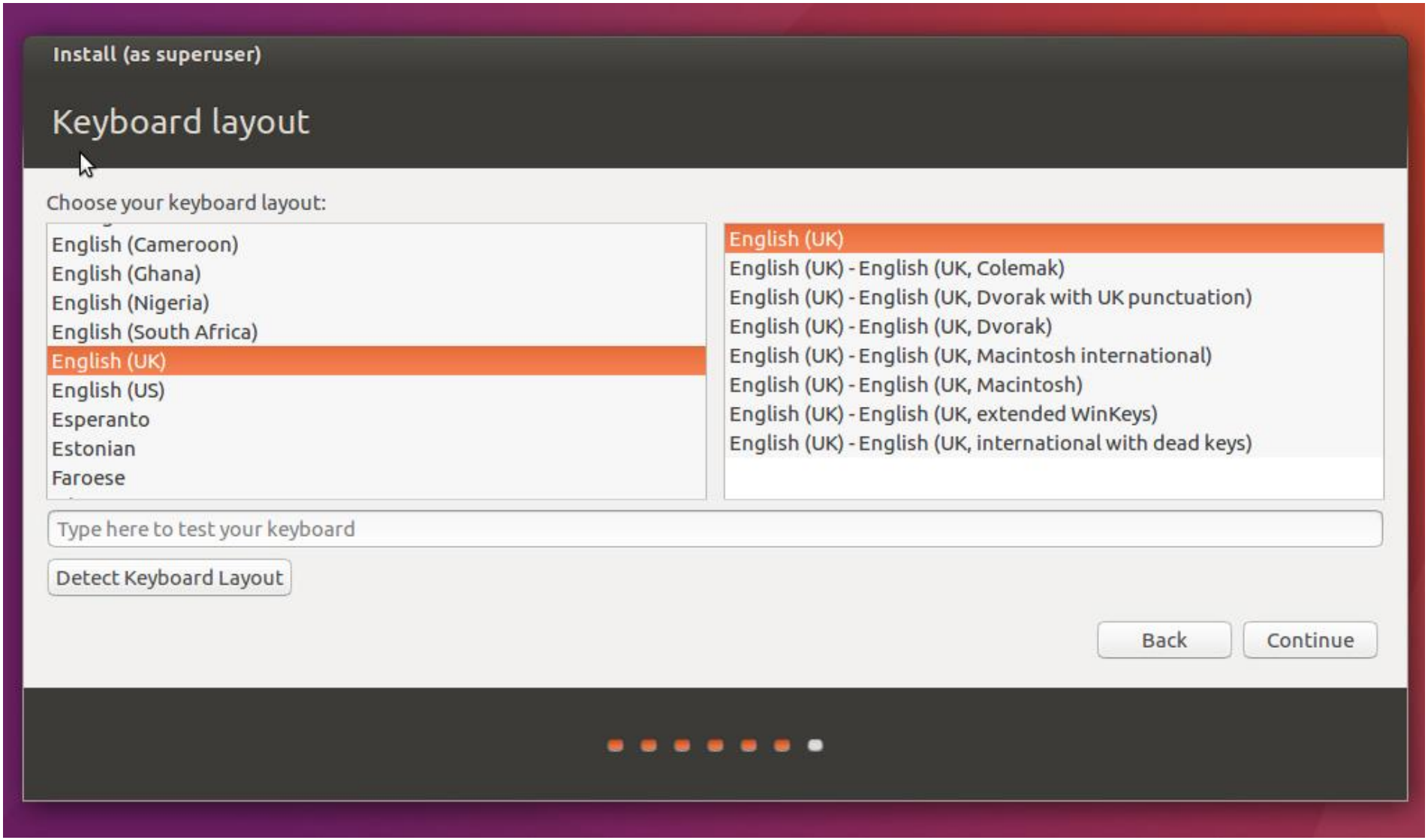
Select this option to install without the Swap partition on SSD

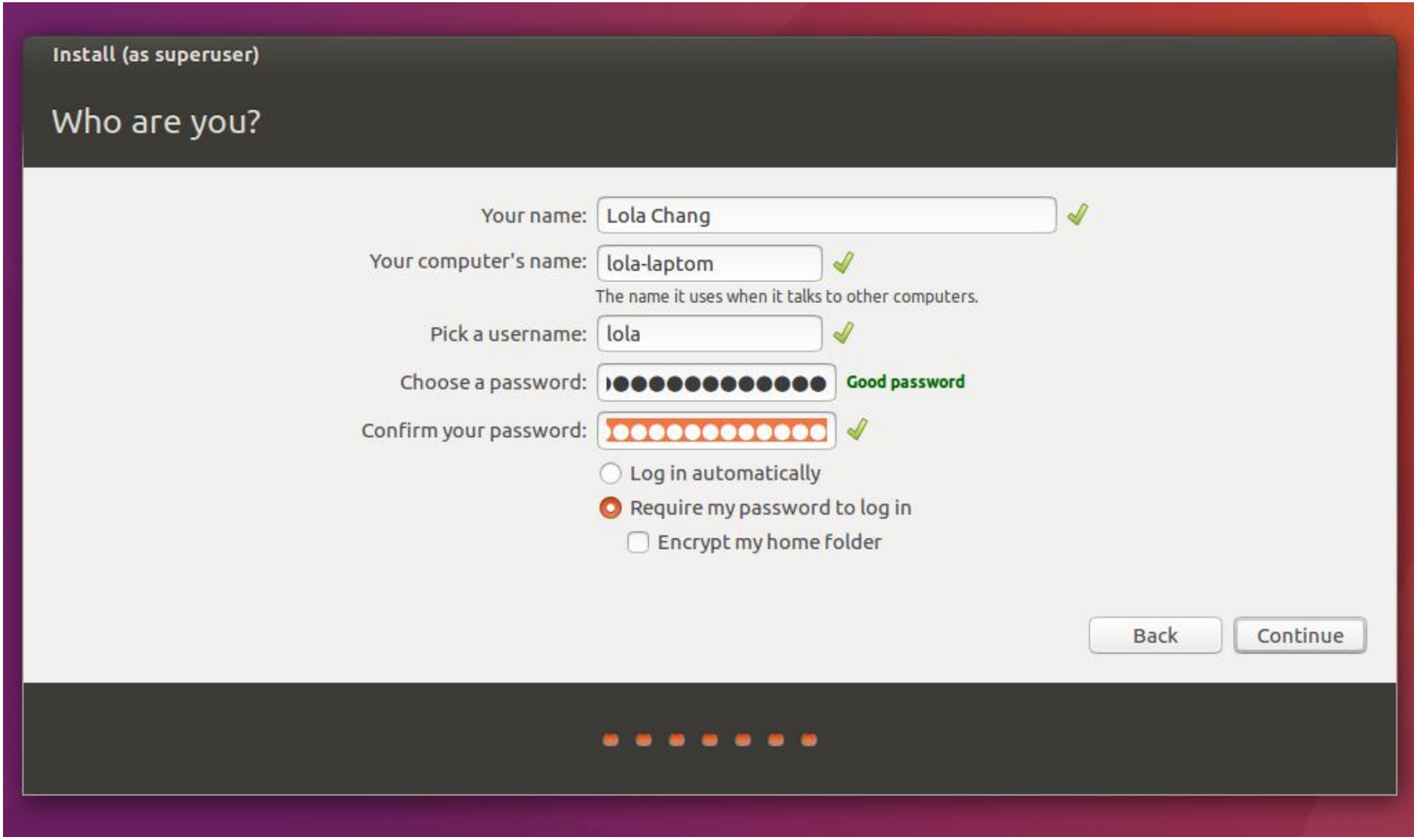

Install (as superuser)
Installation type
free space
128.0 GB
Device
Type
Mount point
Format?
Size
Used
System
/dev/sda
free space
128035 MB
+
−
Change...
New Partition Table...
Revert
Device for boot loader installation:
/dev/sda ATA ADATA SP900NS38 (128.0 GB
Quit
Back
Install Now


Install (as superuser)
Installation type
free space
128.0 GB
Device
Type
Mount point
/dev/sda
free space
+
−
Change...
Create partition
Size:
128036
MB
Type for the new partition:
Primary
Logical
Location for the new partition:
Beginning of this space
End of this space
Use as:
Ext4 journaling file system
Mount point:
/
Cancel
OK
New Partition Table...
Revert
Device for boot loader installation:
/dev/sda ATA ADATA SP900NS38 (128.0 GB
Quit
Back
Install Now


Install (as superuser)
Where are you?
Isle of Man Time
Back
Continue

ROBOTIS


Install (as superuser)
Keyboard layout
Choose your keyboard layout:
English (Cameroon)
English (Ghana)
English (Nigeria)
English (South Africa)
English (UK)
English (US)
Esperanto
Estonian
Faroese
English (UK)
English (UK) - English (UK, Colemak)
English (UK) - English (UK, Dvorak with UK punctuation)
English (UK) - English (UK, Dvorak)
English (UK) - English (UK, Macintosh international)
English (UK) - English (UK, Macintosh)
English (UK) - English (UK, extended WinKeys)
English (UK) - English (UK, international with dead keys)
Type here to test your keyboard
Detect Keyboard Layout
Back
Continue


Install (as superuser)
Who are you?
Your name:
Lola Chang
Your computer's name:
lola-laptom
The name it uses when it talks to other computers.
Pick a username:
lola
Choose a password:
Good password
Confirm your password:
Log in automatically
Require my password to log in
Encrypt my home folder
Back
Continue

Install (as superuser)

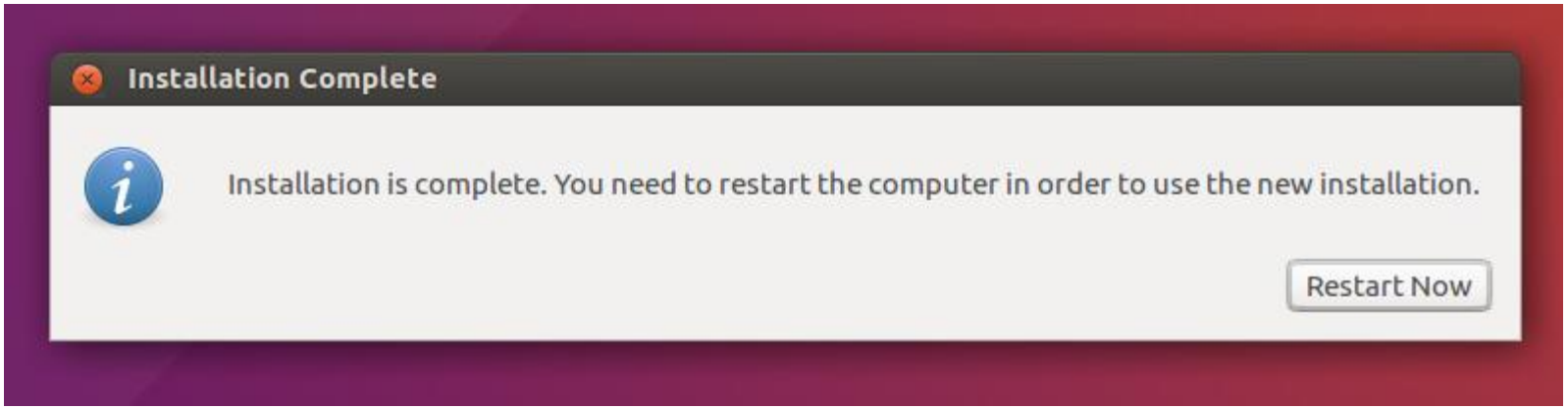
Welcome to Ubuntu

Fast and full of new features, the latest version of Ubuntu makes computing easier than ever. Here are just a few cool new things to look out for…

Almost finished copying files...

Skip

Installing Ubuntu



Installing ROS(Kinect Kame)

1. Setting up sources.list

   Type below command on the terminal.

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release
 -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Setting up the key.

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80
--recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

Installing ROS(Kinect Kame)

3. Debian package index update

```
$ sudo apt-get update
```

4. Desktop-Full Install(Recommended)

```
$ sudo apt-get install ros-kinectic-desktop-full
```

5. Initialize rosdep(Install system dependencies)

```
$ sudo rosdep init
$ rosdep update
```

ROS Environment Setup

1. Create a ROS Workspace

Load ROS environment setting

```
$ source /opt/ros/kinetic/setup.bash
```

Create workspace directory

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

Build

```
$ cd ~/catkin_ws/
$ catkin_make
```

ROS Environment Setup

2. Network Environment Setup

Add ROS environment variables to bash session

```
$ gedit ~/.bashrc
```

Append below code at the end of the .bashrc file

```
# Set ROS Kinetic
source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash

#### Set ROS Network ###
export
ROS_MASTER_URI=http://localhost:11311

# local ROS IP
export ROS_HOSTNAME=localhost
```

ROS Tutorial

1. ROS Filesystem

Packages : Software organization unit of ROS code. Each package can contain libraries, executables, scripts, or other artifacts.

Manifests(package.xml) : Description of a package. Defines dependencies between packages and captures meta information about the package.

ROS Filesystem Command-line Tools

```
$ rospack
$ roscd [package name]
$ roscd log
$ rosls [package name]
```

ROS Tutorial

2. Creating a ROS Package

ROS Package must contain catkin compliant package.xml and CMakeLists.txt files.

Only one package is allowed in each folder.

```
$ cd ~/catkin_ws/src
```

catkin_create_pkg <package name> [depend1] [depend2]...

```
$ catkin_create_pkg tutorial std_msgs roscpp
```

ROS Tutorial

3. Building a ROS Package

A package should be built in the catkin workspace.

```
$ cd ~/catkin_ws
$ catkin_make
```

To add the workspace to the ROS environment, you need to source the generated setup file.

```
$ . ~/catkin_ws/devel/setup.bash
```

ROS Tutorial

4. Understanding ROS Nodes
   A. A node is a process that performs computation.
      A robot system comprise many nodes and usually each function has its own single node.

rosnode command line tool

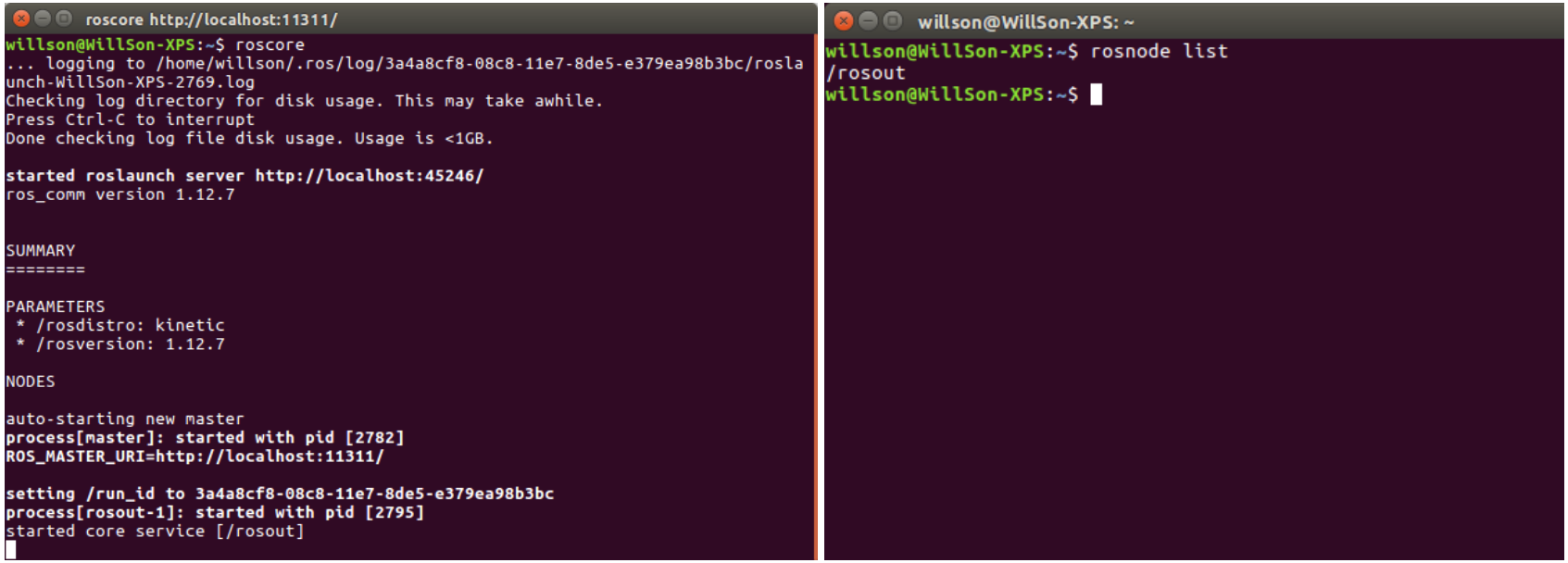
```
$ rosnode info [node name]
$ rosnode kill [node name]
$ rosnode list
$ rosnode machine [PC name or IP address]
$ rosnode ping [node name]
$ rosnode cleanup
```

ROS Tutorial

4. Understanding ROS Nodes(continue)

B. Turtlesim Example

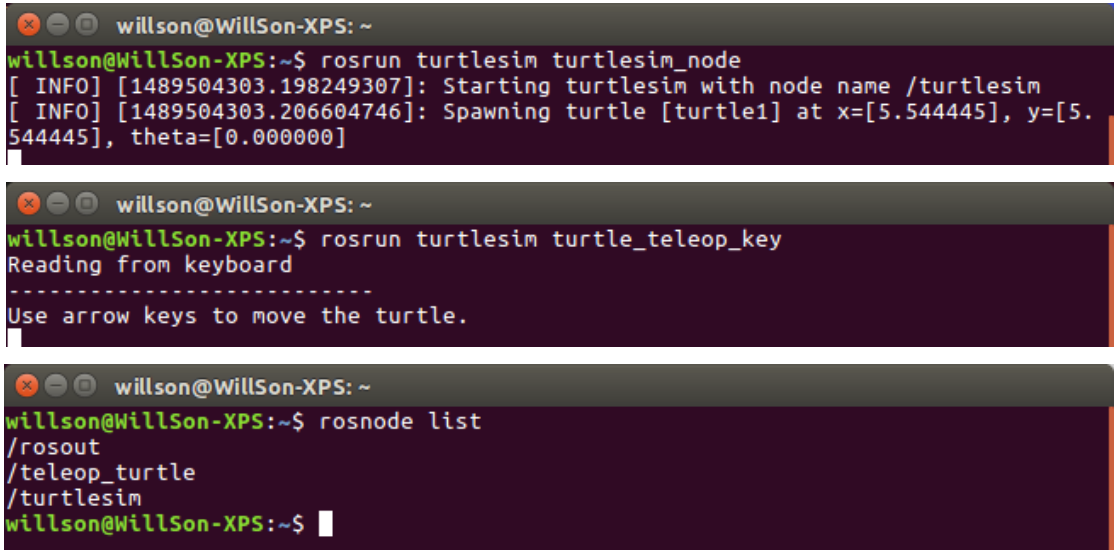
```
$ roscore
$ rosnode list    (on a new terminal)
```

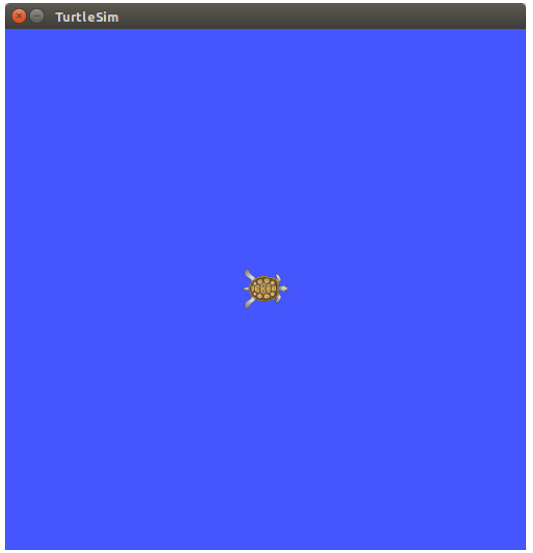


4. Understanding ROS Nodes(continue)

B. Turtlesim Example

```
$ rosrun turtlesim turtlesim_node        (on a new terminal)
$ rosrun turtlesim turtle_teleop_key     (on a new terminal)
$ rosnode list                           (on a new terminal)
```





ROS Tutorial

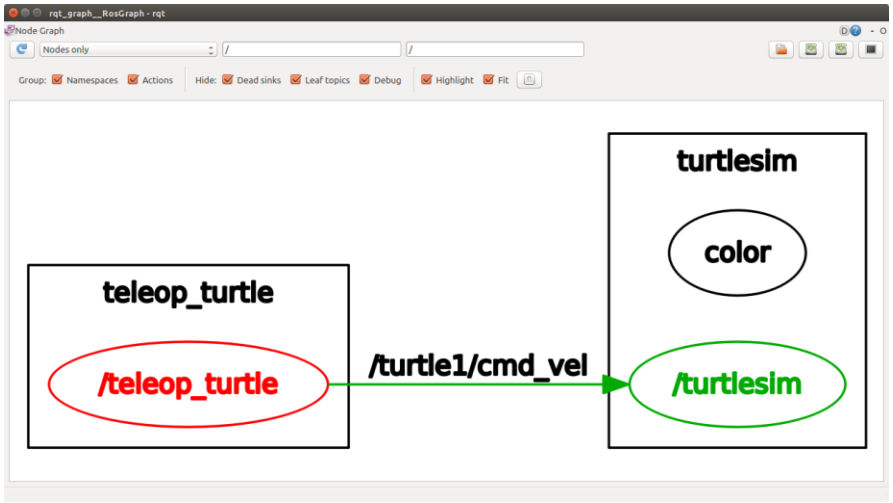
5. Understanding ROS Topics
   A. Topics are named buses over which nodes exchange messages. Nodes that generate data publish to the relevant topic.

rostopic command line tool

```
$ rostopic bw
$ rostopic echo
$ rostopic hz
$ rostopic list
$ rostopic type
$ rostopic delay
$ rostopic find
$ rostopic info
$ rostopic pub
```

5. Understanding ROS Topics(continue)
   B. From the previous node example, turtlesim nodes and topics can be graphically visualized with a graph tool. Open a new terminal window and type below command to launch graph tool.

```
$ rosrun rqt_graph rqt_graph
```



ROS Tutorial

5. Understanding ROS Topics(continue)
   B. Similar to the graph tool, you can observe various data published on topics over the time with the plot tool. Open a new terminal window and type below command to launch the plot tool. (or `rqt` will launch the tool)

```
$ rosrun rqt_plot rqt_plot
```

If `rqt` commands fail to launch the tool, you'll have to install rqt packages first.

```
$ sudo apt-get install ros-kinetic-rqt
$ sudo apt-get install ros-kinetic-rqt-common-plugins
```

5. Understanding ROS Topics(continue)

   C. rosbag records topic data from a running ROS system.

      Recorded topic data can be played later.

```
$ mkdir ~/bagfiles
$ cd ~/bagfiles
$ rosbag record -a
$ rosbag play
```

6. Understanding a ROS msg

A message contains various data with predefined types and it can be transferred via topics or services.

.msg file consists of fields types and field names.

rosmsg command-line tool

```
$ rosmsg show          $ rosmsg list
$ rosmsg package       $ rosmsg packages
```

ROS Tutorial

7. Understanding ROS Services & Parameters

Service is a lot like Topic except service requires a response.

When a service node receives a request from a client node, it replies to the client node.

rosservice command-line tool

```
$ rosservice call          $ rosservice find
$ rosservice info          $ rosservice list
$ rosservice type          $ rosservice uri
```

ROS Tutorial

7. Understanding ROS Services & Parameters

   The parameter server can store various types of data in the parameter server and can be accessed with rosparam commands.

   rosparam command-line tool

```
$ rosparam set            $ rosparam get
$ rosparam load           $ rosparam dump
$ rosparam delete         $ rosparam list
```

ROS Tutorial

8. Creating a ROS msg

Let's use the tutorial package we created earlier.

```
$ roscd tutorial
$ mkdir msg
$ echo "int64 num" > msg/Num.msg
```

In order to build msg file as a source code for C++,

uncomment the following lines from package.xml

```
<build_depend>message_generation</build_depend>
<run_depend>message_runtime</run_depened>
```

8. Creating a ROS msg(continue)

Add or uncomment below codes in the CMakeLists.txt.

```
find_package(catkin REQUIRED COMPONENTS
…
message_generation
...
)
catkin_package(
…
CATKIN_DEPENDS message_runtime
…
)
```

8. Creating a ROS msg(continue)

Add or uncomment below codes in the CMakeLists.txt.

```
add_message_files(
    FILES
    Num.msg
)
generate_messages(
    DEPENDENCIES
    std_msgs
)
```

ROS Tutorial

9. Creating a ROS srv

Let's use the tutorial package we created earlier.

```
$ roscd tutorial
$ mkdir srv
$ echo -e "int64 A\nint64 B\n---\nint64 SUM" > srv/Add.srv
```

or

```
$ roscp rospy_tutorials AddTwoInts.srv srv/Add.srv
```

In order to build srv file as a source code for C++, uncomment the following lines from package.xml

```
<build_depend>message_generation</build_depend>
<run_depend>message_runtime</run_depened>
```

9. Creating a ROS srv(continue)

Add or uncomment below codes in the CMakeLists.txt.

```
find_package(catkin REQUIRED COMPONENTS
…
message_generation
...
)
catkin_package(
…
CATKIN_DEPENDS message_runtime
…
)
```

9. Creating a ROS srv(continue)

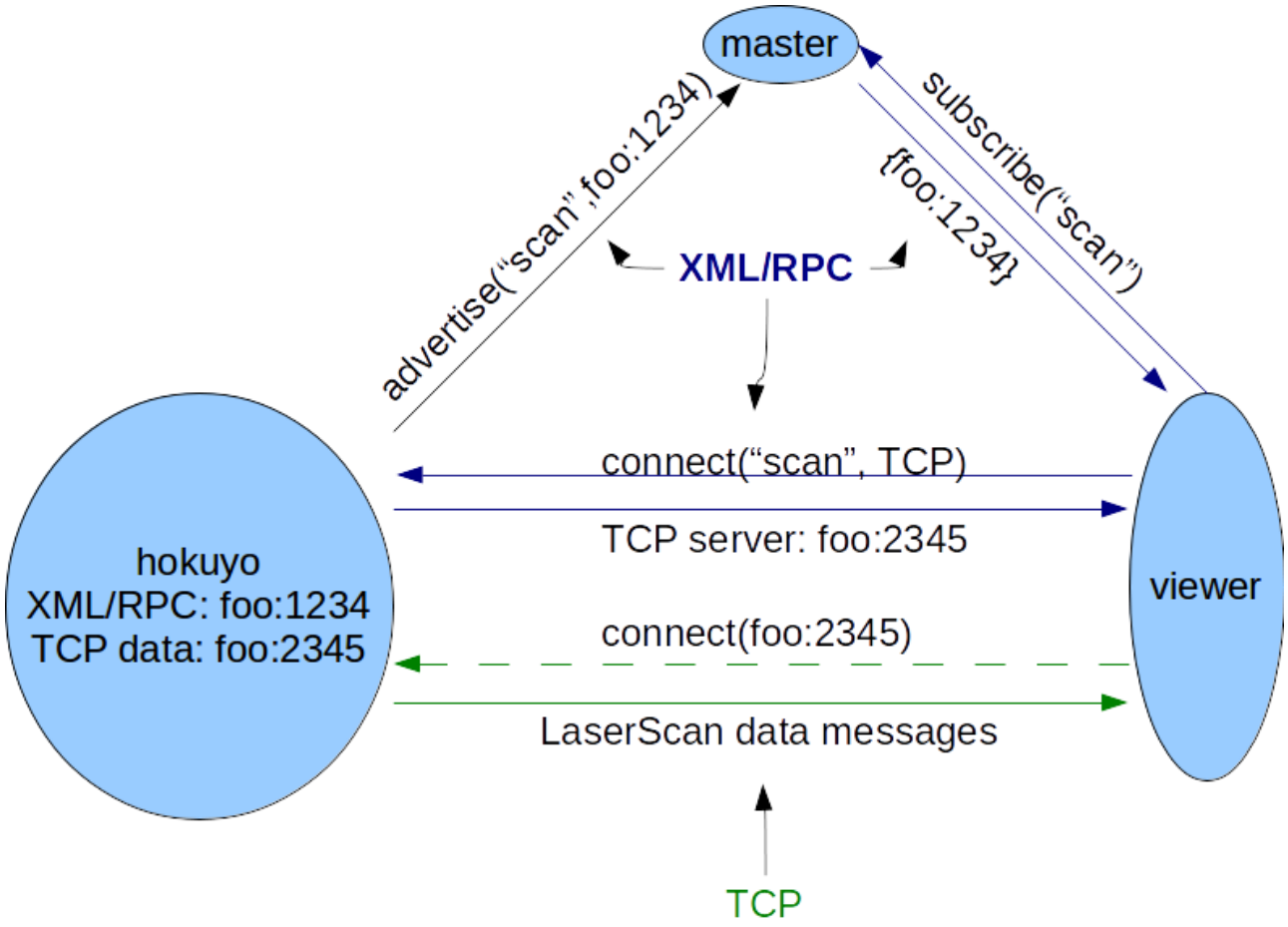
Add or uncomment below codes in the CMakeLists.txt.

```
add_message_files(
    FILES
    Add.srv
)
generate_messages(
    DEPENDENCIES
    std_msgs
)
```

10. Build ROS msg & srv

```
$ cd ~/catkin_ws
$ catkin_make install
```

ROS Tutorial



ROS Tutorial

11. Writing a simple Publisher

Create the src/talker.cpp file in the tutorial package and write the following code in the file.

```
#include “ros/ros.h”
#include “std_msgs/String.h”
#include <sstream>
int main(int argc, char **argv)
{
    ros::init(argc, argv, “talker”);
    ros::NodeHandle n;
    ros::Publisher chatter_pub = n.advertise<std_msgs::String>(“chatter”, 1000);
    ros::Rate loop_rate(10);
```

11. Writing a simple Publisher(continue)

```
int count = 0;
while (ros::ok())
{
    std_msgs::String msg;
    std::stringstream ss;
    ss << “hello world “ << count;
    msg.data = ss.str();

    ROS_INFO(“%s”, msg.data.c_str());
        chatter_pub.publish(msg);
        ros::spinOnce();
        loop_rate.sleep();
        ++count;
    }
    return 0;
}
```

11. Writing a simple Subscriber

Create the src/listener.cpp file in the tutorial package and write the following code in the file.

```
#include “ros/ros.h”
#include “std_msgs/String.h”

void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
    ROS_INFO(“I heard: [%s]”, msg->data.c_str());
}
```

11. Writing a simple Subscriber(continue)

```
int main(int argc, char **argv)
{
    ros::init(argc, argv, “listener”);
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe(“chatter”, 1000, chatterCallback);
    ros::spin();
    return 0;
}
```

12. Build Publisher & Subscriber

In order to build the package, add the following at the end of CMakeLists.txt and *make* the project.

```
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(talker src/talker.cpp)
target_link_libraries(talker ${catkin_LIBRARIES})
add_dependencies(talker tutorial_generate_messages_cpp)

add_executable(listener src/listener.cpp)
target_link_libraries(listener ${catkin_LIBRARIES})
add_dependencies(listener tutorial_generate_message_cpp)
```

12. Build Publisher & Subscriber(continue)

```
$ cd ~/catkin_ws
$ catkin_make
```

How to examine Publisher & Subscriber

```
$ roscore
$ rosrun tutorial talker      (on a new terminal)
$ rosrun tutorial listener    (on a new terminal)
```

ROS Tutorial

13. Writing a simple Service node

Create the src/add_server.cpp file in the tutorial package.

```
#include “ros/ros.h”
#include “tutorial/Add.h”

bool add(tutorial::Add::Request &req, tutorial::Add::Response &res)
{
    res.SUM = req.A + req.B;
    ROS_INFO(“request: x=%ld, y=%ld”, (long int)req.A, (long int)req.B);
    ROS_INFO(“sending back response: [%ld]”, (long int)res.SUM);
    return true;
}
```

13. Writing a simple Service node(continue)

```
int main(int argc, char **argv)
{
    ros::init(argc, argv, “add_two_ints_server”);
    ros::NodeHandle n;
    ros::ServiceServer service = n.advertiseService(“add_two_ints”, add);
    ROS_INFO(“Ready to add two ints.”);
    ros::spin();
    return 0;
}
```

ROS Tutorial

14. Writing a simple Client node

Create the src/add_client.cpp file in the tutorial package.

```
#include “ros/ros.h”
#include “tutorial/Add.h”
#include <cstdlib>
int main(int argc, char **argv)
{
    ros::init(argc, argv, “add_two_ints_client”);
    if (argc != 3)
    {
        ROS_INFO(“usage: add_two_ints_client X Y”);
         return 1;
    }
    ros::NodeHandle n;
    ros::ServiceClient client = n.serviceClient<tutorial::Add>(“add_two_ints”);
    tutorial::Add srv;
```

14. Writing a simple Client node(continue)

```
    srv.request.A = atoll(argv[1]);
    srv.request.B = atoll(argv[2]);
    if (client.call(srv))
    {
          ROS_INFO("Sum: %ld", (long int)srv.response.SUM);
    }
    else
    {
          ROS_ERROR("Failed to call service add_two_ints");
          return 1;
    }
    return 0;
}
```

ROS Tutorial

15. Build Service & Client node

```
$ cd ~/catkin_ws
$ catkin_make
```

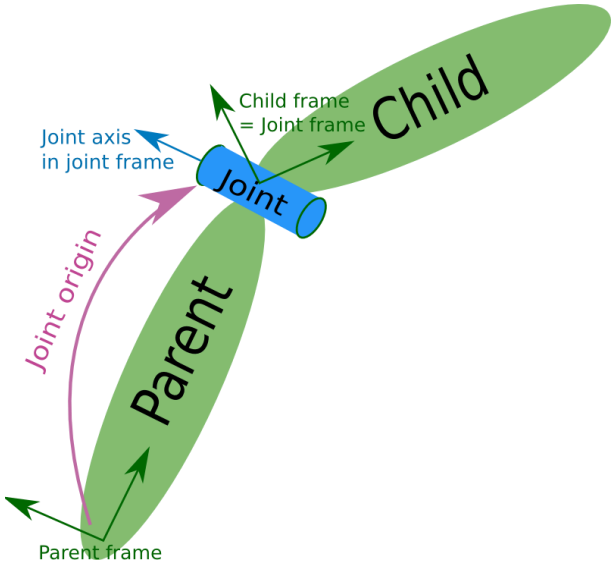
How to examine Service & Client

```
$ roscore
$ rosrun tutorial add_server        (on a new terminal)
$ rosrun tutorial add_client 3 6    (on a new terminal)
```

16. URDF(Universal Robotic Description Format)

URDF is an XML format for representing a robot model such as robot's joint, link, inertia information. Robot's TF(transformation) information can be created by calculating URDF information.

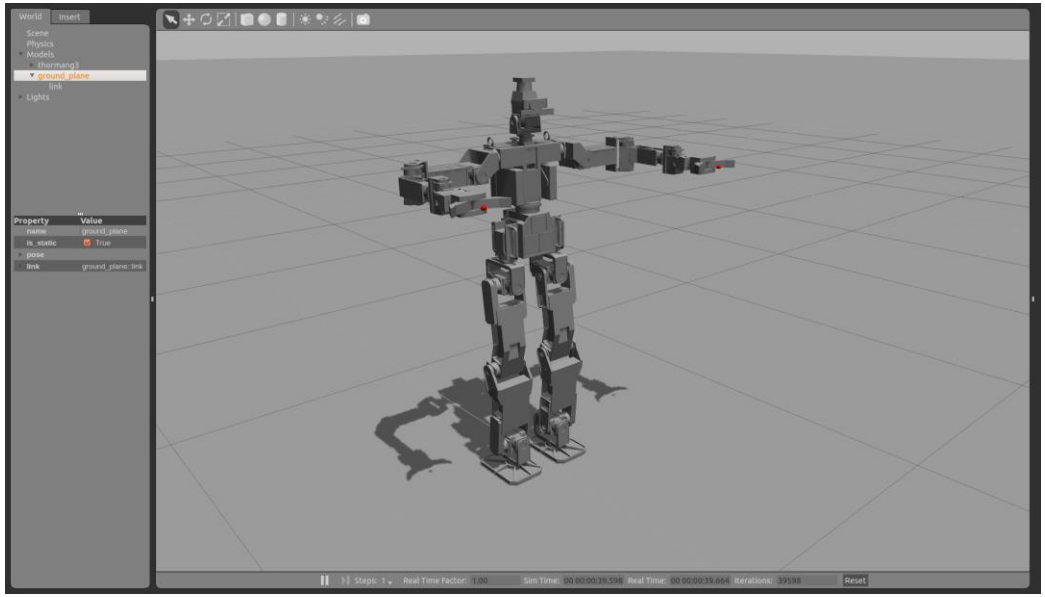
Thormang3 URDF model is included in the thormang3_description package.



17. Gazebo

Gazebo is a 3D dynamic simulator based on physics engine to simulate populations of robots in complex environments.

GUI interface supports 3D model editor that users can create and define joint link hierarchy of the robot.



18. Rviz

Rviz is a 3D visualization tool for ROS.

An interactive marker can be created on the workspace to calculate the pose of the robot or the endeffector before actually operate it.