

# **Robotic Applications and Design Fall 2021**

## **Robotics Software Tools (Intro)**

**Ioannis REKLEITIS, Ibrahim SALMAN**



UNIVERSITY OF  
**South Carolina**

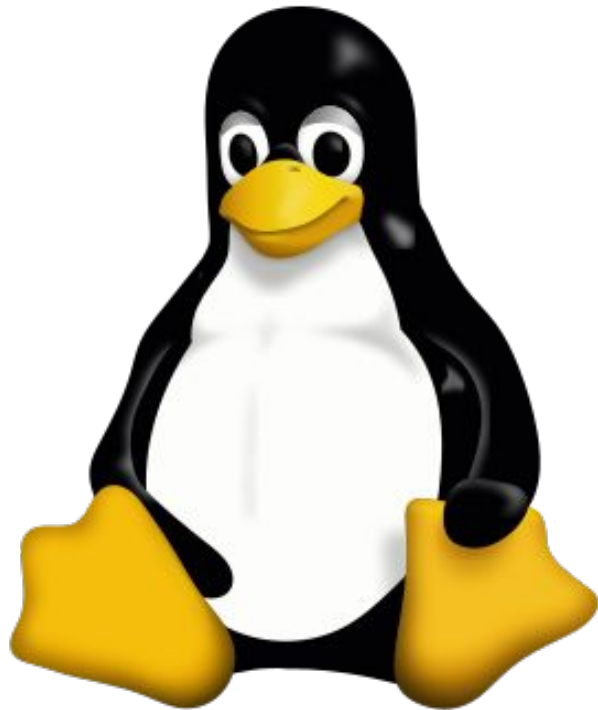
---

College of Engineering  
and Computing

# Lecture Outline

- Linux Overview
- ROS Overview
- Git Overview
- Docker Overview

# Linux Overview



# Linux Overview Outline

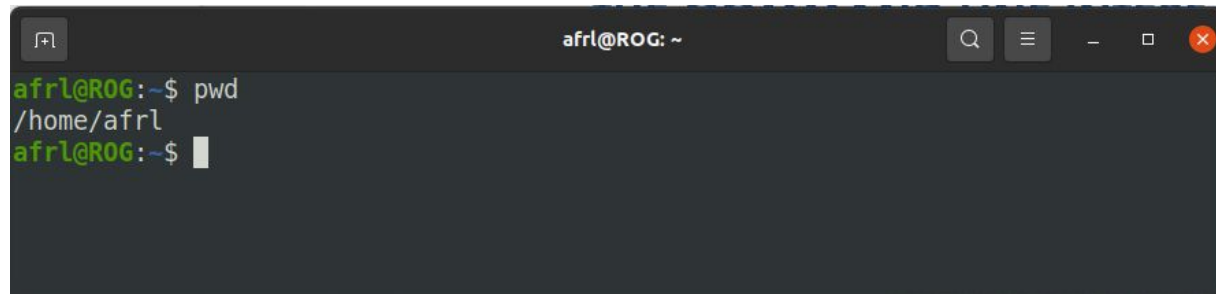
- Why Linux?
- Linux Command Line Interface (CLI)
- Common Topics:
  - CLI manuals
  - Common Helpful commands and Directory Structures
  - File Permissions
  - Shell Scripts
  - ssh & scp
- Notes:
  - **Ubuntu 20.04** is the official supported distro this semester
  - Natively booting linux is recommended (**single** or **dual** boot)

# Why Linux

- Open source
- Security
- Privacy
- Cost (FREE)
- Portability
- Hardware Support
- Support

# Command Line Interface

- CLI vs GUI
- Different command line shells:
  - bash (most common)
  - zsh
  - fish
- Secret tip to master terminal (Hint: practice)

A terminal window with a dark background and light text. The window title is 'afri@ROG: ~'. The prompt is 'afri@ROG:~\$'. The user has entered 'pwd' and the output is '/home/afri'. The prompt is now 'afri@ROG:~\$' with a cursor.

```
afri@ROG: ~  
afri@ROG:~$ pwd  
/home/afri  
afri@ROG:~$
```

# CLI Manuals and Help

- Manual Pages
  - Display a Man Page
  - Search Specific Sections
  - **Tip:** you can also search the manual on the web
- Help
  - help option for most programs: -h and --help
- Will go over piping output from man and help and using grep to search for specific options later in the slides

# Helpful commands & Directory Structures

- Basics
  - ls
  - cd
  - **Directory Structures**
  - mkdir
  - rm
  - pwd
  - mv
  - cp
  - echo
  - cat
  - history
  - alias
  - **source (!important)**
- pipe
- grep, find
- text editors in terminal: (vi, vim, nano, etc..)



# File Permissions

- Permission groups
- Permission Types (bit codes):
  - **read** -  $r = 4$
  - **write** -  $w = 2$
  - **execute** -  $x = 1$
- chmod command (owner/group/public)
- Making files executable

# Shell Scripts

- Why scripts
- Types of scripts
  - `bash`
  - `sh`
  - etc..
- writing scripts
- making a script executable
- Executing scripts

# Secure Shell (SSH)

- What is ssh?
- Why ssh?
  - Remote CLI
  - Remote Execution
  - **SCP** (Secure file transfer over **ssh**)
- ssh options
  - auth
  - ports

# Questions about Linux?

# Robot Operating System

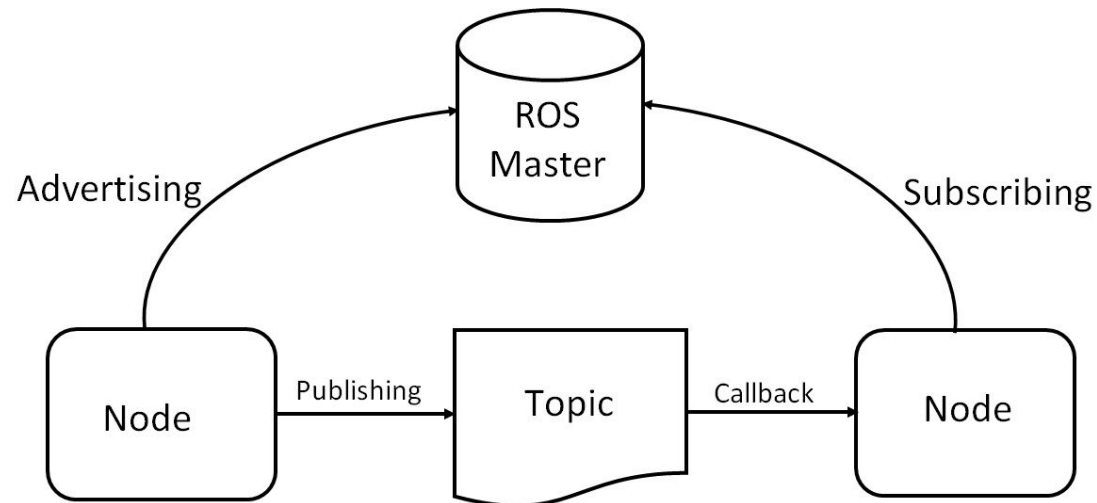
 ROS

# ROS Overview Outline

- How to pronounce
- What is ROS?
- Why ROS?
- ROS & Linux?
- ROS basics

# What is ROS?

- ROS is an open-source, meta-operating system for your robot.
- General Concept
  - Multiple nodes
  - communicate with other nodes using a publish/subscribe messaging model



Credit: [www.designnews.com](http://www.designnews.com)

# Why ROS?

- ROS is lightweight
- a lot of support and packages (**open-source**)
- Programming languages support (C++/Python/JAVA)
- package management



# ROS & LINUX

- ROS Dir structure
  - /home/**\$USER**/catkin\_ws
    - build
    - devel
    - src
      - package1
        - CMakeLists.txt
        - launch
        - package.xml
        - src
      - package 2
- `catkin_create_pkg` is used to init a ros package.
- ROS uses shell variables, so we source `~/catkin_ws/devel/setup.bash` to ensure all variables defined in setup.bash are added to the current shell.
- Remember: always source setup.bash after running `catkin_make`

# ROS basics

- node
  - programs that run in ros and can communicate with other nodes through topics
- topic
  - messages with defined type. Can use standard messages or define your own
  - a node can publish and subscribe to multiple topics or to none
- service
  - get information from node upon request rather than continuous broadcast

# ROS basics

- `roslaunch`
  - use to run and inspect individual nodes
- `rostopic`
  - interact with topics: list, get info, display data, publish to topic
- `rostopic`
  - inspect nodes
- `rostopic`
  - find potential bugs
- `roslaunch`

# Questions about ROS?

# Git



# Git Overview Outline

- What is git?
- Why git?
- Git basics

# What is git

- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Source: <https://git-scm.com/>
- Github

**Note:** a significant part of the course material will be on github.

# Why git

- easy to learn
- lightweight
- fast
- collaboration

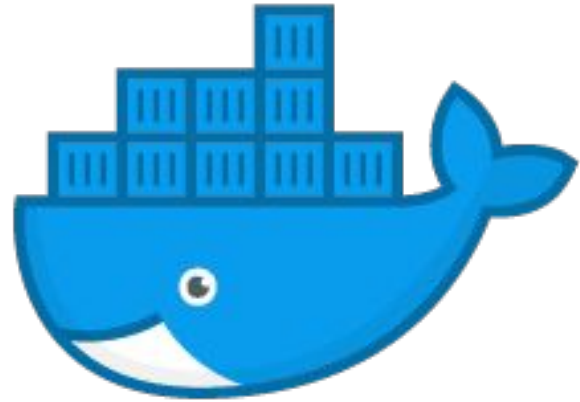


# Git basics

- Repository
- clone
- add
- commit
- push
- pull
- branch
- issue
- collaboration

# Questions about Git?

# Docker



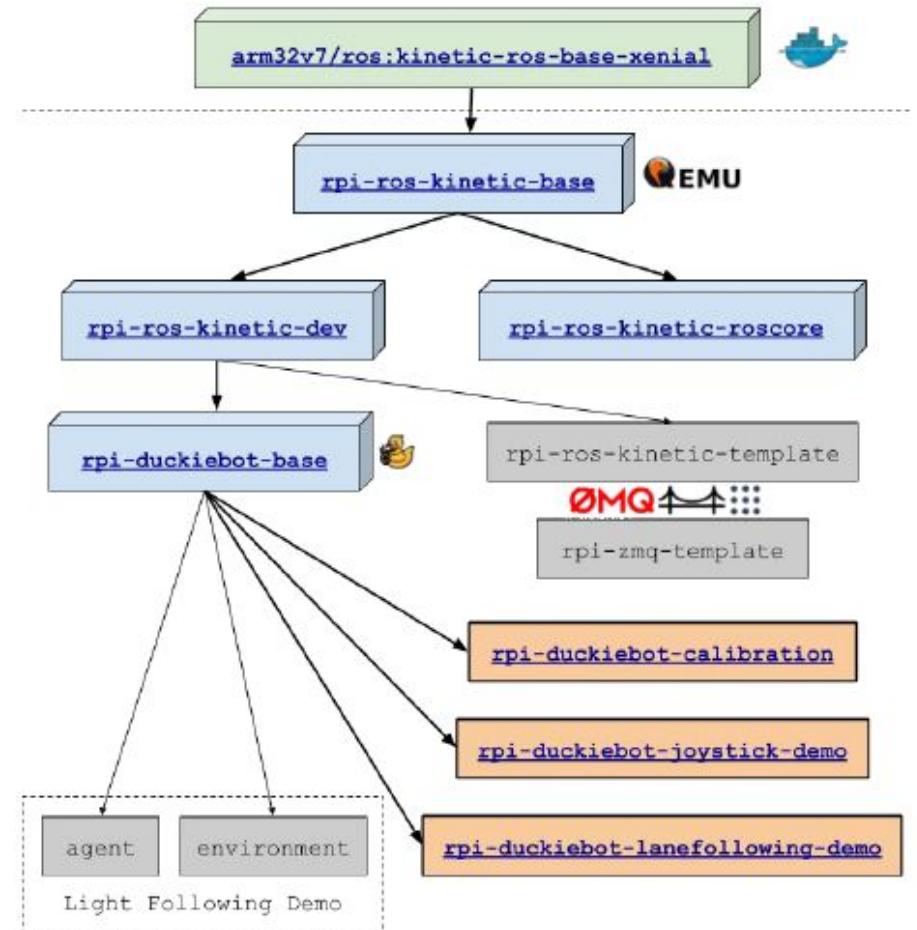
docker

# What is Docker

- Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. Source: IBM

# DOCKER

- Containers stored online: Docker Hub
- Docker images: builds of a configuration
- Images can be overlaid on each other
- Docker container: runtime of an image



# DOCKER USAGE

- Pull images from docker's servers using  
`$ docker pull <image name>`
- To start a container from an image use:  
`$ docker run -it <image name>`
  - This will also pull any required images
  - Note the “-it” flag: this makes it interactive allowing you to type in the terminal
- To share a folder on your computer with this container use the `-v` flag:  
`$ docker run -it -v <host path>:<container path> <image name>`
- To find what containers are running and their names use:  
`$ docker ps`
- To start a second process/terminal in a single docker container run:  
`$ docker exec -it <container name> bash`

# DOCKER BUILD

- It is often useful to build your own docker images
- These are usually based on other images
- Some have only minor modifications
- Build the image with:  
`$ docker build -t <name>:<version> <path-to-dockerfile>`
- Only needs to be run when dockerfile is changed (rarely)

# CONCEPT OF OPERATIONS

- Run assignments in docker images
- Keeps environment consistent for student and instructor
- Make sure to turn in whatever you use
- Easy transition to programming Duckiebots in a few weeks



# Questions about Docker?

# Questions ?

Email: [YIANNISR@cse.sc.edu](mailto:YIANNISR@cse.sc.edu)  
Email: [IJSALMAN@email.sc.edu](mailto:IJSALMAN@email.sc.edu)