Weston Robot

RSE2107A – Lecture 3

Introduction to common development tools & foundations of mobile robots

Weston Robot

Agenda

01
Git Basics

02
Introduction to Ground Robots

03
Project Management Fundamentals

Weston Robot

Agenda

04
LIMO Fix

05
Kickstarter

06
Project Update

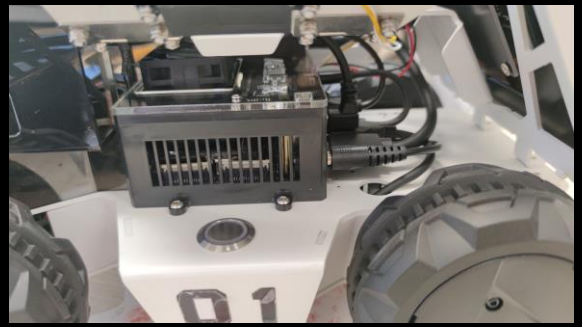
LIMO Diagnosis

- What happened:
  - Plug the 12V charger to the 5V port of NANO
  - The Nano board cannot start
  - Producing 3-beeps

- Read manual completely before operation
- For electrical & electronics: shape matching doesn't mean compatible
- Don’t debug hardware without tools
- Try not to debug system in front of “customers” (except common steps)
- NEVER Point Fingers

LIMO Diagnosis

- Step 1: Isolate the issues
- Step 2: Reproduce the result (be careful of the cost, avoid further damage)
- Step 3: Identify the root cause (TVSD)
- Step 4: Apply the fixes (total cost: SGD7,860 to fix the two LIMOs)



All prices are in SGD

#	Product Details	Quantity	Availability	Unit Price	Extended Price
1	SMAJ5.0ALFCT-ND SMAJ5.0A TVS DIODE 5VWM 9.2VC DO214AC	20	Immediate	0.43900	$8.78
2	SSM3K333RLFCT-ND SSM3K333R,LF MOSFET N CH 30V 6A 2-3Z1A	20	Immediate	0.54000	$10.80
3	1727-1306-1-ND PESD1CAN-UX TVS DIODE 24VWM 50VC SOT323	50	Immediate	0.49800	$24.90
4	505-ADT7420UCPZ-RL7CT-ND ADT7420UCPZ-RL7 SENSOR DGTL -40C-150C 16LFCSP	2	Immediate	13.68000	$27.36

https://www.digikey.sg/en/products/detail/littelfuse-inc/SMAJ5-0A/762250

KICKSTARTER

About Kickstarter (e.g., Robots)

- Cool ideas
- Solving a real pain point
- Confident for delivery

Luba: An Intelligent, Perimeter Wire Free Robot Lawn Mower

No Perimeter Cable | Multi-zone Management | Mow up to 5,000㎡ | Obstacle Avoidance | App Control | 75% Slope (AWD) | Auto-recharging

S$ 1,616,678
pledged of S$ 8,789 goal

883
backers

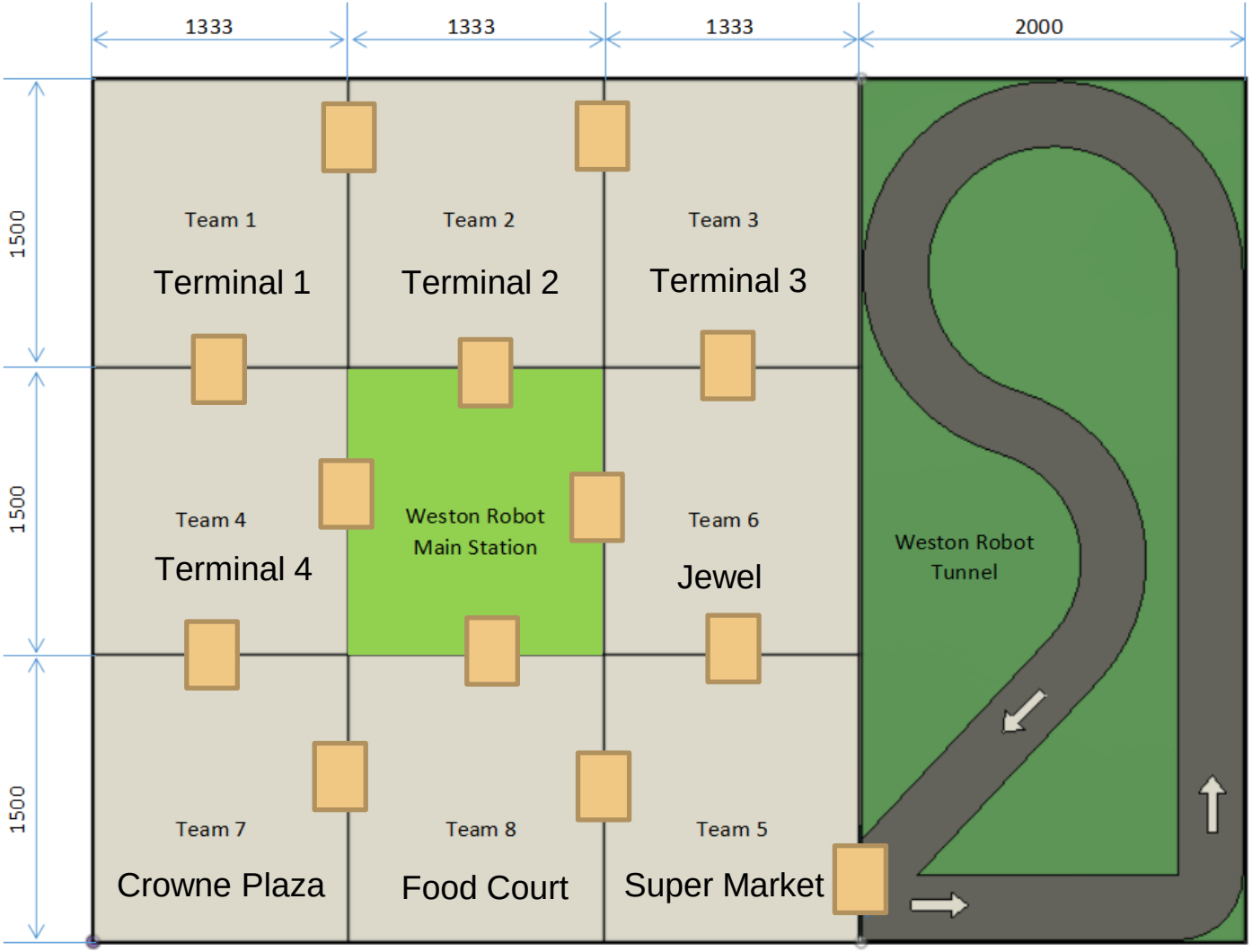
32
days to go

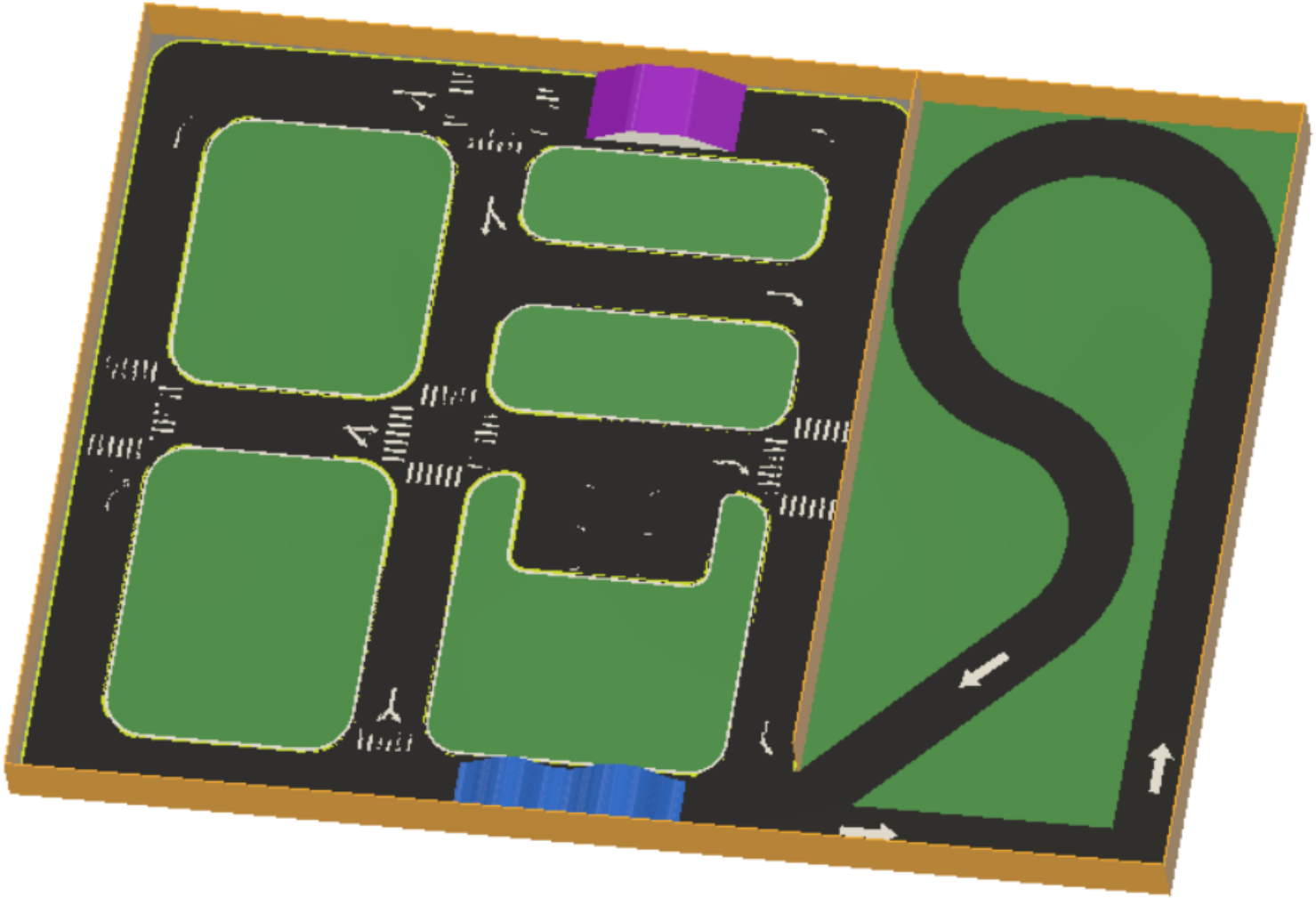
Back this project

Remind me

All or nothing. This project will only be funded if it reaches its goal by Thu, June 23 2022 10:58 PM AWST.

Project We Love | Robots | Hong Kong, Hong Kong


1333
1333
1333
2000
1500
1500
1500
Team 1
Terminal 1
Team 2
Terminal 2
Team 3
Terminal 3
Team 4
Terminal 4
Weston Robot
Main Station
Team 6
Jewel
Weston Robot
Tunnel
Team 7
Crowne Plaza
Team 8
Food Court
Team 5
Super Market



Schedule

- First draft by 5/6
- Feedback to you by 8/6
- Second Iteration by 12/6
- Feedback to you by 15/6
- Design freezes by 19/6
- Mat received before 15/7
  - Make sure your decoration items are completed within this month

Remarks

- Work as a big team (daily communication)
- Please use fusion 360 (free for students)
- Please make two copies for each of your decoration
- Please mind your budget
- Be cool

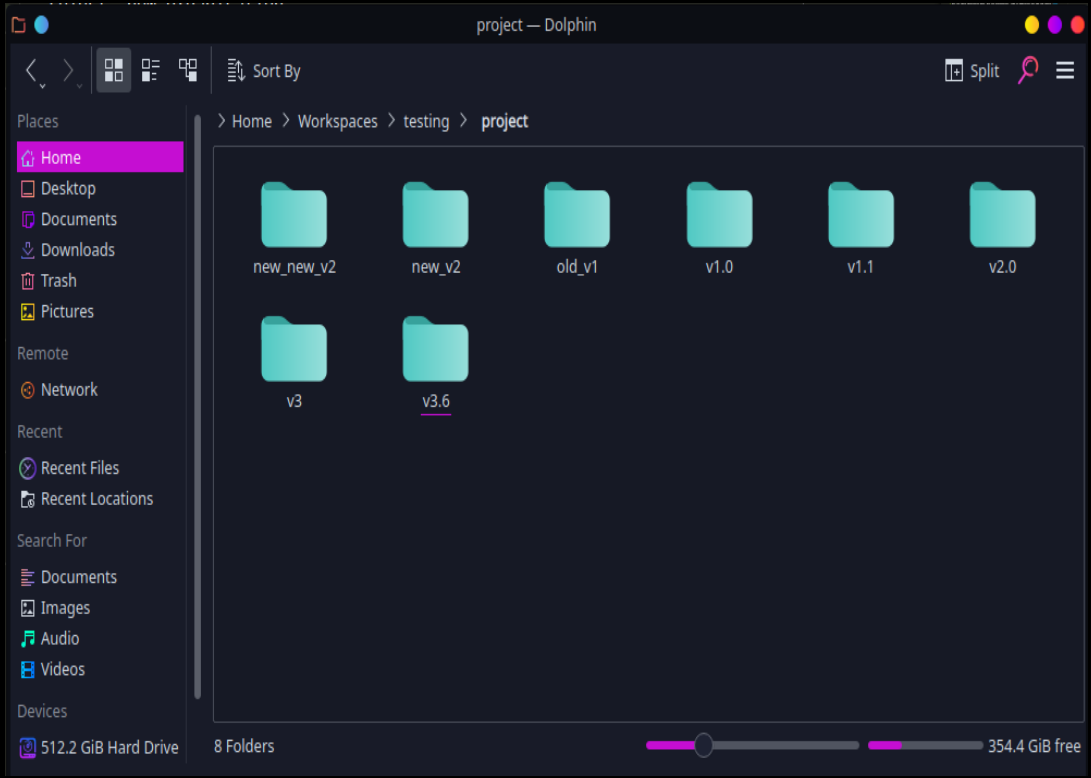
Version Control

chaos vs controlled chaos

Weston Robot

What is version control?

- Version control (aka source control), is the practice of tracking and managing changes to a codebase.
- Easy to understand, difficult to practice.



Version Control Systems

- Version control systems are tools that aims to simplify the process by taking over the task of maintaining a complete history of changes to a codebase.

- Git is one such system, a free and open source distributed version control system originally developed in 2005 and one of the most commonly used systems today.

Git? GitHub/Gitlab?

- GitHub is an online service to which developers who use Git can connect and upload or download resources.

GitHub

Git(hub) Basics

Git Projects & Repositories

- A Git project is a folder of software and other resources that is tracked by Git.

- A Git repository tracks and maintains the history of all changes made to the files within a Git project and saves this data into a “.git” folder (aka the repository folder).

Weston Robot

Git History

- Git stores the history of a project by keeping a snapshot (aka a commit) throughout the project's life, each building on another earlier commit.

- These commits (identified by a unique id (aka hash)), together form a complete change history (or graph) for the given project.

```
commit 434de730f5abe43c8f6f8e32247f2e04d31635f6 (HEAD -> newversion, origin/master, origir
Author: fyodor <fyodor@e0a8ed71-7df4-0310-8962-fdc924857419>
Date:   Sun Dec 9 02:00:55 2018 +0000

    Update copyright year for Ncat and Ncat Guide

commit 6d420e82b2c55b6c7723c07e33771c52ad193b5e
Author: dmiller <dmiller@e0a8ed71-7df4-0310-8962-fdc924857419>
Date:   Sun Dec 2 05:54:58 2018 +0000

    Changelog for #1227

commit 1ba01193725f4c83bf9e4b4cd589dbc9fc626152
Author: dmiller <dmiller@e0a8ed71-7df4-0310-8962-fdc924857419>
Date:   Sun Dec 2 05:48:27 2018 +0000

    Add a length check for certificate parsing. Fixes #1399

commit b1efd742499b00eef970feeef84dc64f301db61f
Author: dmiller <dmiller@e0a8ed71-7df4-0310-8962-fdc924857419>
Date:   Thu Nov 29 20:27:05 2018 +0000

    Warn for raw scan options without needed privileges

commit b642dc129c4d349a849fb0eb055cde263d9d3eb6
Author: dmiller <dmiller@e0a8ed71-7df4-0310-8962-fdc924857419>
Date:   Thu Nov 29 17:42:09 2018 +0000

    Fix a bug in the fix. https://github.com/nmap/nmap/commit/ebf083cb0bfc239a000aea7764cc

commit 350bbe0597d37ad67abe5fef8fba984707b4e9ad
Author: dmiller <dmiller@e0a8ed71-7df4-0310-8962-fdc924857419>
Date:   Thu Nov 29 17:42:09 2018 +0000

    Avoid a crash (double-free) when SSH connection fails
```

A file’s Lifecycle

- From creation to deletion, a file in any git project will go through multiple stages throughout the lifespan of the project.

- These stages are what Git uses to track, control and maintain its change history and a user can control which stage a file/change is in.

Stage - *Untracked*

- Files that did not exist in the previous commit and has not been added to the git repository yet.
  - Typically newly created files
  - Files that have been explicitly removed or ignored from the history.
- Git won’t start tracking the file (again) unless told explicitly to.

Stage - *Unmodified*

- Files that have been previously committed and have not been changed since the last commit.

Stage - *Modified*

- Files that have been previously committed and have been changed since the last commit.

Stage - *Staged*

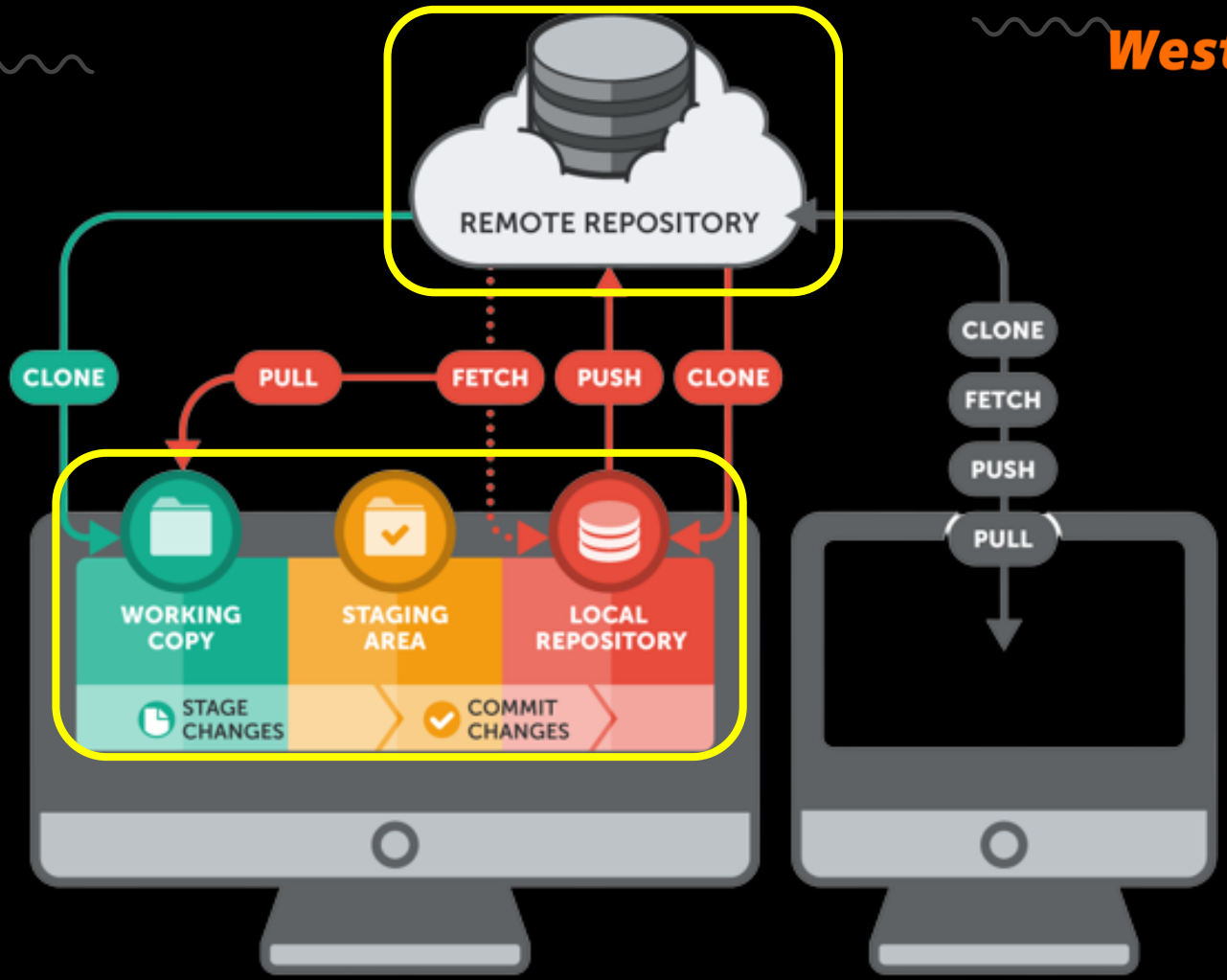
- Files that are being prepared to be committed
  - Only *Untracked & Modified* files can be staged.
  - These files will be compared to their state in the previous commit and there changes tracked.

- Once these files have been committed to the history, they return to being *Unmodified* files.

Local vs Remote Repositories

- Local
  - Repositories that resides on a developer’s computer.
  - Often the one being worked on.
  - Almost certainly necessary in any Git project
- Remote
  - Repositories resides on a remote computer/server (like GitHub’s) .
  - Often used to share/collaborate on a codebase.
  - Not strictly necessary

Weston Robot



Developer A

Developer B

Common Git commands

git init

- Turns a directory (and all its sub-directories) into an empty Git repository

```
# change directory to codebase
$ cd /Users/computer-name/Documents/website

# make directory a git repository
$ git init
Initialized empty Git repository in /Users/computer-name/Documents/website/.git/
```

git add

- Adds files to staging area

```
# To add all files not staged:
$ git add .

# To stage a specific file:
$ git add index.html

# To stage an entire directory:
$ git add css
```

git commit

- Record changes made to staged files into a commit in the local repository

```
$ git commit -m "My first commit message"
[SecretTesting 0254c3d] My first commit message
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 homepage/index.html
```

git status

- Returns the current state of the repository

```
# Message when files have not been staged (git add)
$ git status
On branch SecretTesting
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        homepage/index.html

# Message when files have been not been committed (git commit)
$ git status
On branch SecretTesting
Your branch is up-to-date with 'origin/SecretTesting'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   homepage/index.html

# Message when all files have been staged and committed
$ git status
On branch SecretTesting
nothing to commit, working directory clean
```

git config

- Assigning/Removing/Changing Git configurations and settings.

```
# Running git config globally
$ git config --global user.email "my@emailaddress.com"
$ git config --global user.name "Brian Kerr"

# Running git config on the current repository settings
$ git config user.email "my@emailaddress.com"
$ git config user.name "Brian Kerr"
```

Weston Robot

git branch

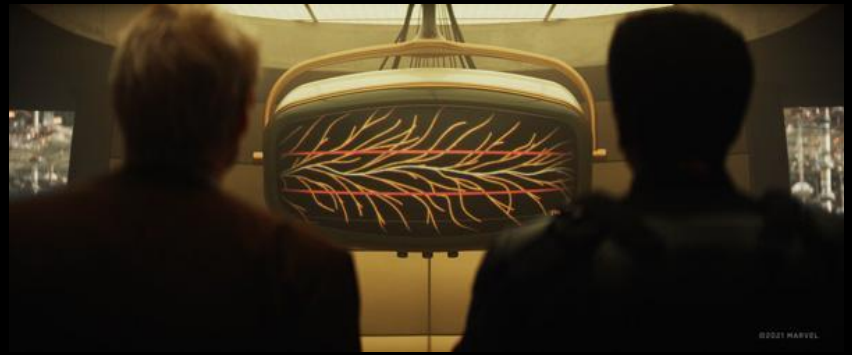
- Determine what branch local repository is on, adding, viewing and deleting branches

```
# Create a new branch
$ git branch new_feature

# List branches
$ git branch -a
* SecretTesting
  new_feature
  remotes/origin/stable
  remotes/origin/staging
  remotes/origin/master -> origin/SecretTesting

# Delete a branch
$ git branch -d new_feature
Deleted branch new_feature (was 0254c3d).
```

Branching

- If we view the git history as a timeline, branching like the name suggests creates an “alternate” history where we can make changes to the code without affecting the main history.

- This “alternate” timelines can be merged back into the main history, maintained separately or even pruned.

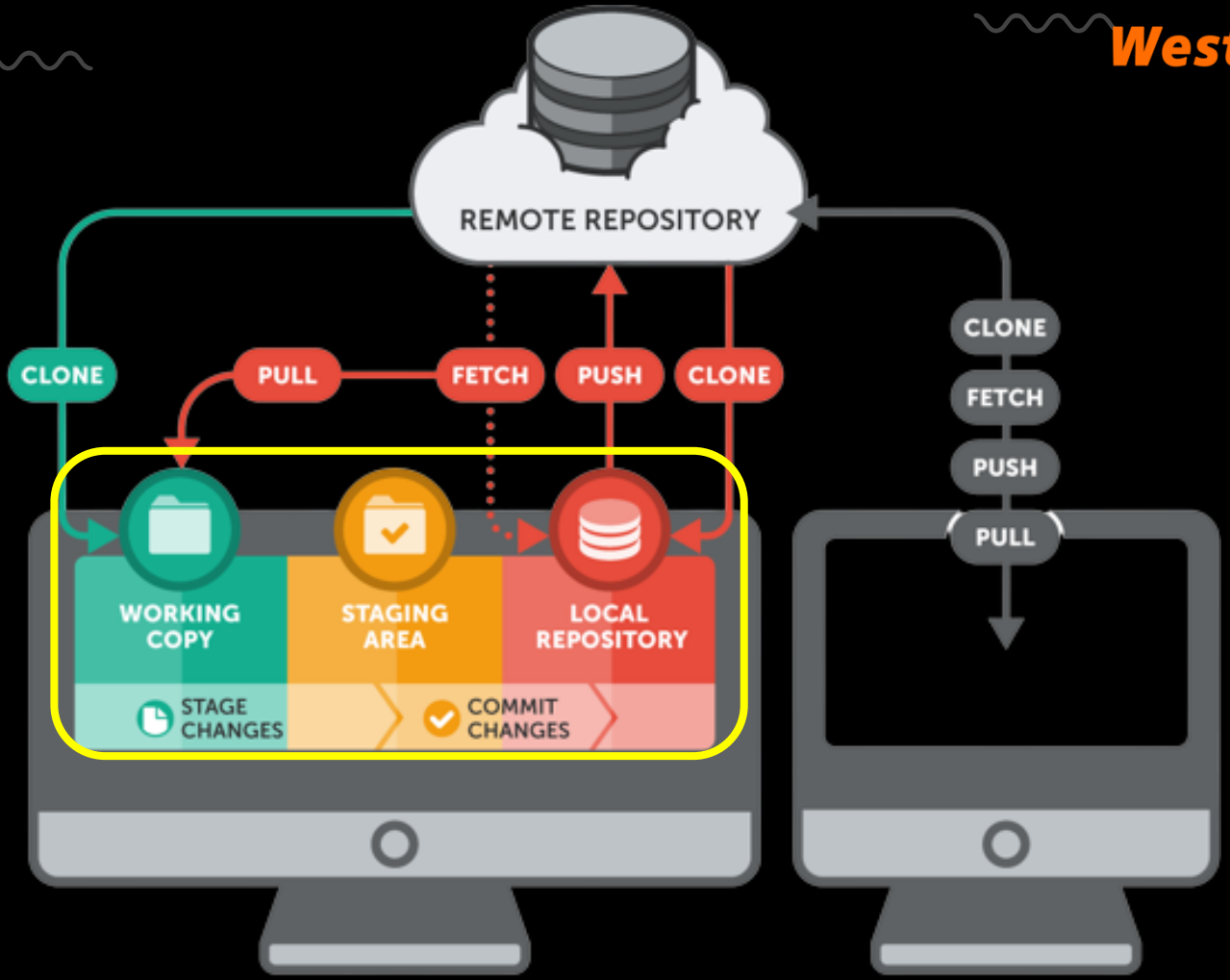


git checkout

- Switch to work in a different branch (for switching between branches)

```
# Switching to branch 'new_feature'
$ git checkout new_feature
Switched to branch 'new_feature'

# Creating and switching to branch 'staging'
$ git checkout -b staging
Switched to a new branch 'staging'
```

Weston Robot



Developer A

Developer B

git clone

- Create a local working copy from an existing remote repository

```
$ git clone git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git
Cloning into 'repository_name'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (5/5), 3.08 KiB | 0 bytes/s, done.
Checking connectivity... done.
```

git fetch

- Checks for any new commits from a remote repository (but does not get these changes).

```
kb@phoenixNAP:~/project$ git fetch origin
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (13/13), done.
Unpacking objects: 100% (24/24), 3.89 KiB | 1.95 MiB/s, done.
remote: Total 24 (delta 0), reused 9 (delta 0), pack-reused 0
From github.com:phoenixNAP-KB/test
 * [new branch]      global      -> origin/global
 * [new branch]      main        -> origin/main
 * [new branch]      test        -> origin/test
 * [new branch]      test_alias  -> origin/test_alias
 * [new branch]      test_branch -> origin/test_branch
 * [new tag]         v1          -> v1
 * [new tag]         v1.1        -> v1.1
kb@phoenixNAP:~/project$
```

git pull

- Getting latest branch version/changes/history from a remote repository branch

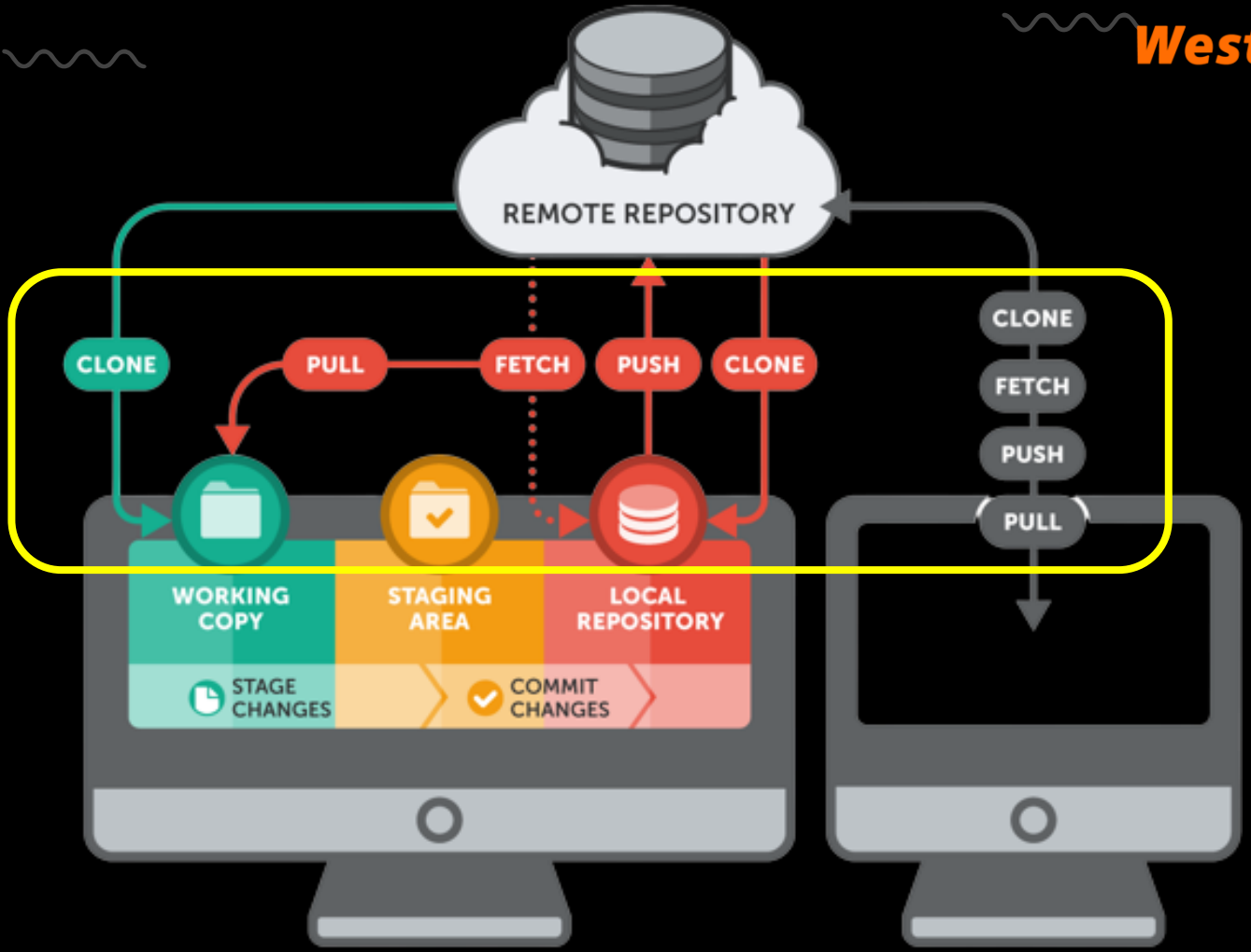
```
# Pull from named remote
$ git pull origin staging
From account_name.git.beanstalkapp.com:/account_name/repository_name
 * branch            staging    -> FETCH_HEAD
 * [new branch]      staging    -> origin/staging
Already up-to-date.

# Pull from URL (not frequently used)
$ git pull git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git staging
From account_name.git.beanstalkapp.com:/account_name/repository_name
 * branch            staging    -> FETCH_HEAD
 * [new branch]      staging    -> origin/staging
Already up-to-date.
```

git push

- Sends local commits to the remote repository (updating remote repository with all commits done)

```
# Push a specific branch to a remote with named remote
$ git push origin staging
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 734 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
To git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git
   ad189cb..0254c3d  SecretTesting -> SecretTesting
```

Weston Robot



Developer A

Developer B

Managing conflicts

When dealing with multiple merge requests from different collaborators within the same repository (typically a shared remote one), you are likely to encounter merge conflicts.

- e.g. Member A and B both made their own changes to the same file.
  - Git needs to know which code/change to use, which results in a merge conflict
  - Conflict will need to be resolved before being able to merge successfully
    - Needs to be resolved by whoever is doing the merging

Important tools for GitHub

- Markdown
  - Lightweight markup language used to add or format elements to plain-text markup language
  - Portable across different platforms (Windows, Mac, Linux, …)
  - Supported on GitHub
- Link to cheatsheet for Markdown
  - https://www.markdownguide.org/cheat-sheet

Weston Robot

Important tools for GitHub

Markdown syntax

How it will look like

README.md

limo_gazebo_sim > README.md > # limo_gazebo_sim > ## Usage > ### 1. Run the star file of limo model and visualize t

hanskw4267, last week | 1 author (hanskw4267)

```
# limo_gazebo_sim

## About
This package contains the gazebo simulation worlds and launch configuration files needed to run a limo robot within a gazebo simulation.

## Environment
On top of the ROS installation, we will need additional packages to run the simulations successfully.

### Download and install required function package

   Download and install ros-control function package, ros-control is the robot control middleware provided by ROS

```
sudo apt install ros-melodic-ros-control
```

   Download and install ros-controllers function package, ros-controllers are the kinematics plug-in of common models provided by ROS

```
sudo apt install ros-melodic-ros-controllers
```

   Download and install gazebo-ros function package, gazebo-ros is the communication interface between gazebo and ROS, and connect the ROS and Gazebo

```
sudo apt install ros-melodic-gazebo-ros
```

   Download and install gazebo-ros-control function package, gazebo-ros-control is the communication standard controller between ROS and Gazebo

```
sudo apt install ros-melodic-gazebo-ros-control
```

   Download and install joint-state-publisher-gui package.This package is used to visualize the joint control.
```

Preview README.md

limo_gazebo_sim

About

This package contains the gazebo simulation worlds and launch configuration files needed to run a limo robot within a gazebo simulation.

Environment

On top of the ROS installation, we will need additional packages to run the simulations successfully.

Download and install required function package

Download and install ros-control function package, ros-control is the robot control middleware provided by ROS

```
sudo apt install ros-melodic-ros-control
```

Download and install ros-controllers function package, ros-controllers are the kinematics plug-in of common models provided by ROS

```
sudo apt install ros-melodic-ros-controllers
```

Download and install gazebo-ros function package, gazebo-ros is the communication interface between gazebo and ROS, and connect the ROS and Gazebo

```
sudo apt install ros-melodic-gazebo-ros
```

Download and install gazebo-ros-control function package, gazebo-ros-control is the communication standard controller between ROS and Gazebo

```
sudo apt install ros-melodic-gazebo-ros-control
```

Introduction to Ground Robot

Weston Robot

Robot Locomotion

- Type of motions
  - Ackerman
  - Differential
  - Tracked
  - Omni Drive

Weston Robot

Types of mode

Ackermann Mode

A geometry designed to solve the problem of wheels on the inside and outside of a turn needing to trace out circles of different radii in the steering of vehicles.

Four-Wheel Differential Mode

Four-wheel drive, which can realize on the spot auto-rotation, but it will cause serious tire wear; please do not auto-rotate on the spot for a long time

Weston Robot

Types of mode

Track Mode

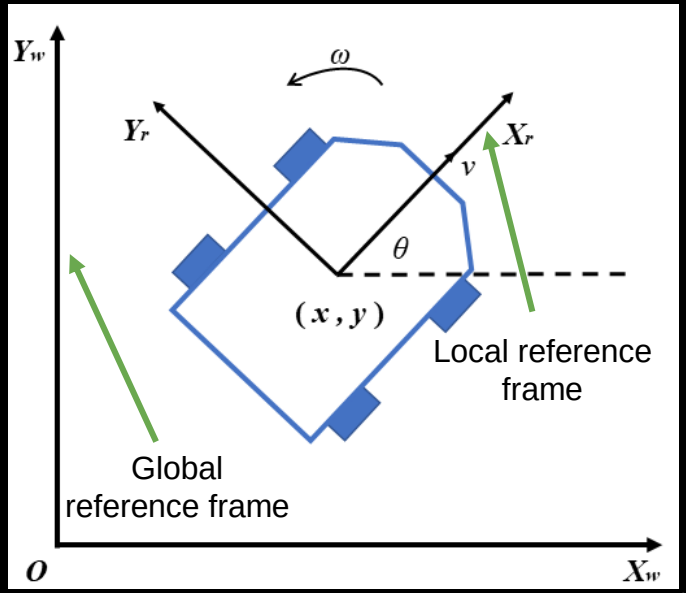
It has good off-road performance and can climb 40° slopes and small steps

Mecanum Wheel Mode

The omni-directional motion equipment based on Mecanum wheel technology can realize forward, lateral, oblique, rotation and combinations of motion modes.

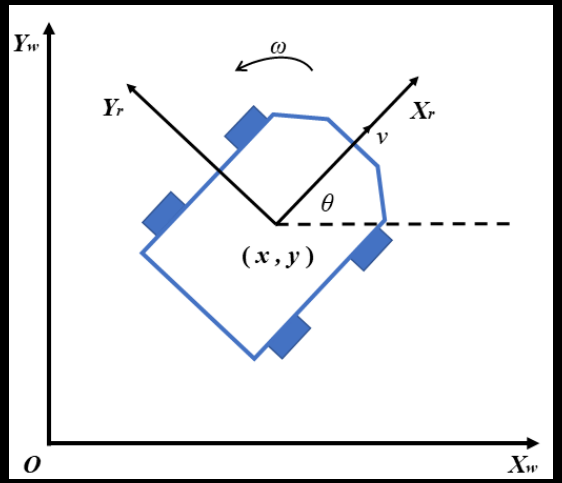
Representing robot position

- Reference frames are with respect to a body that can be used to describe the position of points with respect to the body's coordinate system(frame).
  - Local reference frame refers to the coordinate system of the robot (body).
  - Global reference frame refers to the coordinate system of the area that the robot is functioning within.



Representing robot position

- Orthogonal rotation matrix, R(θ)
  - Used to map vectors expressed in global frame (X_W , Y_W , Z_W) to that of local reference frame (X_r , Y_r , Z_r).
  - Given position of robot in global reference frame: $\xi_W = [x,y,\theta]^T$
  - Position of robot expressed in local frame: $\xi_W = R(\theta) \times \xi_r$.



$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Project Management

Weston Robot

Company Wide (not Execution)

- Start with pre-LTS (long-term strategical plan), LTS
  - Often conducted by industrial experts with lots of market study
    - Why?/What?
    - Resources
    - Roadmap
- For Execution:
  - Pre-launch (Dev. Lead, PM, QE, Supply Chain Management)
    - Beta users
    - 4P: Product/Price/Place/People
  - Launch (Direct launch? Kickstarter? KOL?)
  - Post-launch
    - Feedback
    - Roadmap adjustment (minor)

Weston Robot

Team-Level Execution (Development)

- Keep team members below 9 ppl (5-9 are most effective)
  - A team leader (help get the resources)
  - A technical leader
- Roadmap + Backlog + Sprint
  - Roadmap: critical milestones to achieve
    - Milestones with tasks and subtasks
    - Quantifiable and trackable
  - Backlog
    - Tasks popped up: Technical support/Bug fixes/Enhancement (new features)
  - Sprint
    - A time interval during which tasks in the roadmap and backlog must be completed
    - Resources must be allocated to complete the tasks
    - A consensus (no excuses to fail)

Weston Robot

Team-Level Execution: Tools

Related Project Management Picks: Best Online Collaboration Software | Best Business Messaging Apps | Best Time Tracking Software

OUR 10 TOP PICKS

Best for Beginners

PC EDITORS CHOICE 4.5 Outstanding

GanttPro

Available at GanttPRO — Check Price

GanttPro is a top project management app for small teams because it's easy to use and has plentiful features. Don't expect customizable reports or dashboards, however.

Read Our GanttPro Review

Best for Client Work

PC EDITORS CHOICE 4.5 Outstanding

Teamwork

Available at Teamwork — Check Price

With an extensive set of features and intuitive interface, Teamwork is one of the best services for managing projects. With billing and invoicing included, it's especially suited to teams that handle client work.

Read Our Teamwork Review

Best for Small and Growing Teams

PC EDITORS CHOICE 4.5 Outstanding

Zoho Projects

Available at Zoho Projects — Check Price

Because Zoho Projects is a low-cost project management app with an array of helpful features, it's an attractive option for small and growing businesses.

Read Our Zoho Projects Review

Best for Value

4.0 Excellent

Celoxis

Available at Celoxis — Check Price

For medium or large organizations, Celoxis packs a lot of value into a project management app, especially for decision-makers and business owners.

Read Our Celoxis Review

Best for A

LiquidPla

Availa
at Liquid Pla

LiquidPlan
managing
more, autc
scheduling
factors cha

Read Our

PROS & CONS | COMPARE SPECS

Team-Level Execution (Development)

Example (at Weston Robot):

1. Development Project Management: Jira
2. Internal Communication: Slack
3. Internal File Storage: OneDrive
4. Internal Document Writing: Yuque
5. Code Hosting: GitHub (public) and Gitlab (private)
6. Sales/Marketing Management: SalesMate
7. Sales/Marketing Task Management: Teambition

Lab 3

- Prelab and lab notes will be released today.

END