

# Version Control with Git

Chu-Pan Wong

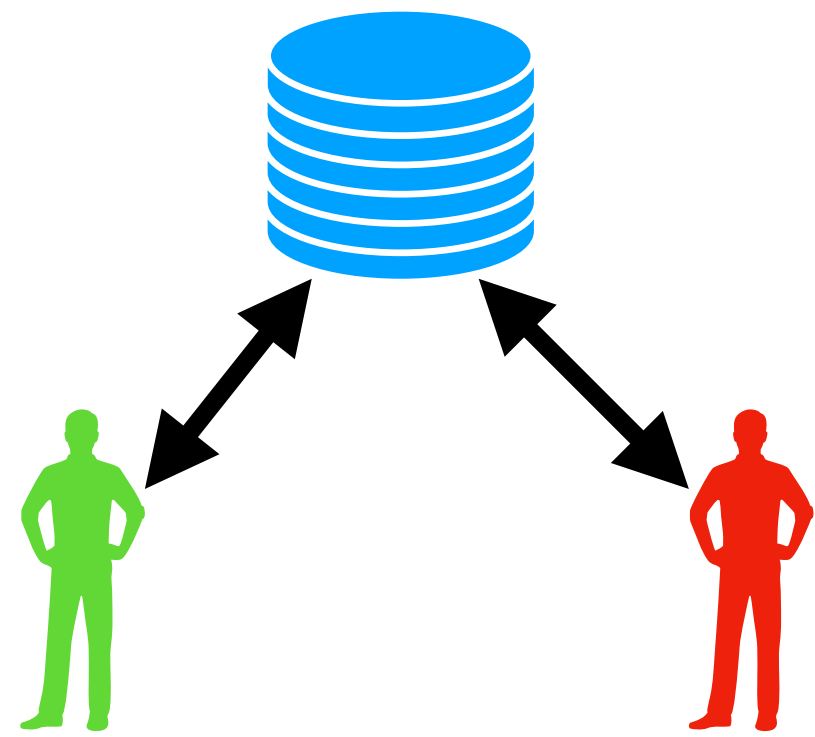
Sep 6, 2019

“I run a small hosting provider with more or less 1535 customers and I use Ansible to automate some operations to be run on all servers,”

“Last night I accidentally ran, on all servers, a Bash script with a  
*rm -rf {foo}/{bar}*  
with those variables undefined due to a bug in the code above this  
line”

*–Marco Marsala*

# Git



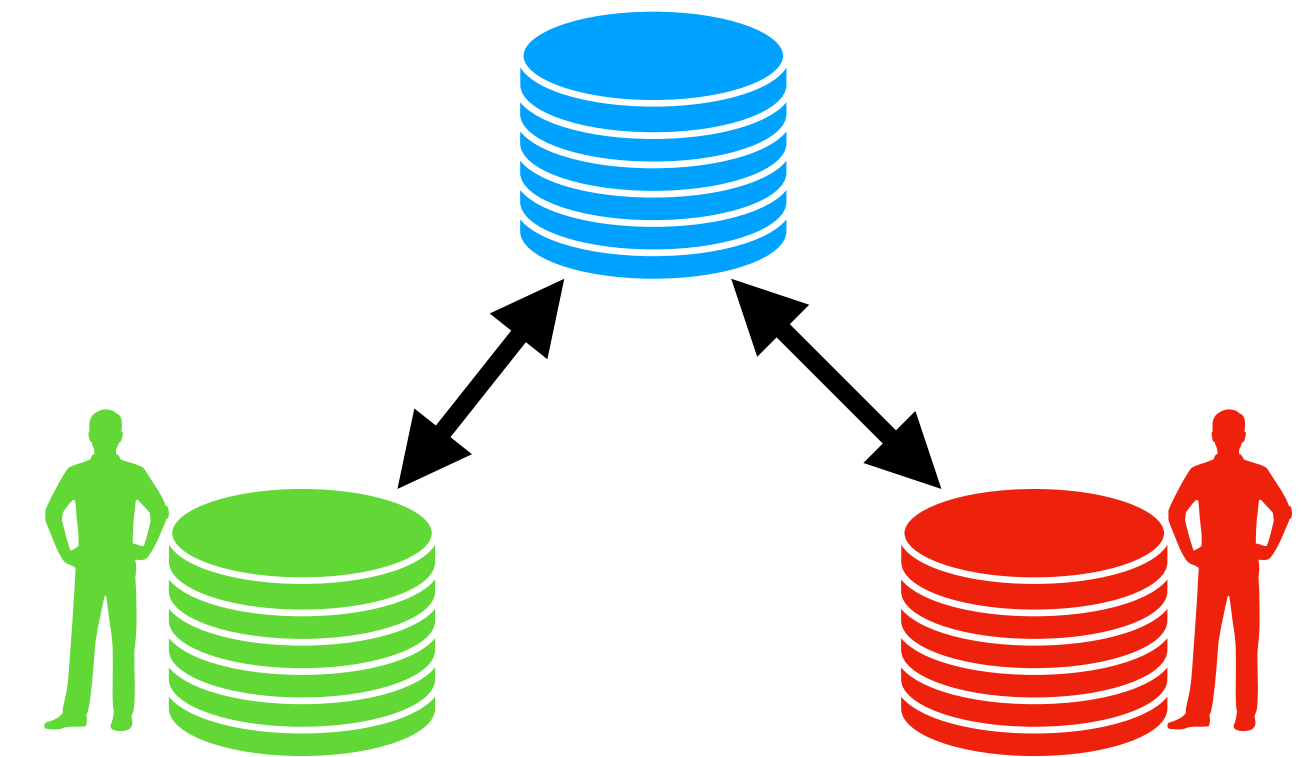
**Disconnected Development**

**Fast Performance**

**Ease of Use**

**Strong Support for Branching**

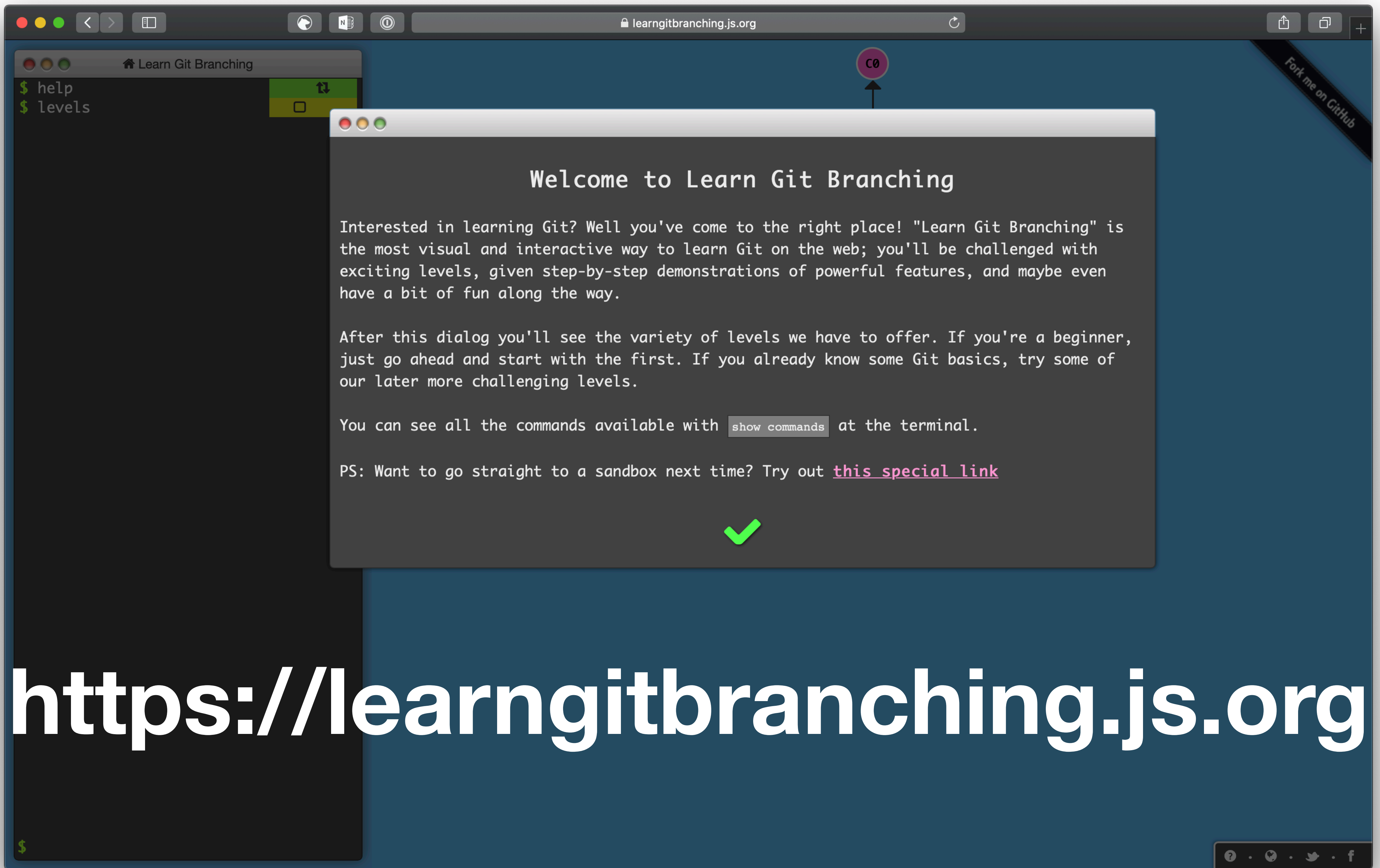
**One Working Area, Multiple Branches**



# Common Git Commands

- git init
- git add
- git status
- git log
- git commit
- git reset / git revert
- git tag
- git branch
- git checkout
- git merge / git rebase
- git cherry-pick
- git clone
- git pull
- git fetch
- git push

**man git-<sub-command>**



<https://learngitbranching.js.org>

# Best Practices

**Commit Related Changes**

**Commit Often**

**Test Before You Commit**

**Write Good Commit Message**

**Exclude Unnecessary Files**

**Use Branches**

**Stick to a Workflow**



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

- Separate subject from body with a blank line
- Limit the subject line to 50 (or so) characters
- Wrap body with 70 (or so) characters
- Use the body to explain what, why, how

Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72 characters or so. In some contexts, the first line is treated as the subject of the commit and the rest of the text as the body. The blank line separating the summary from the body is critical (unless you omit the body entirely); various tools like `log`, `shortlog` and `rebase` can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you are making this change as opposed to how (the code explains that). Are there side effects or other unintuitive consequences of this change? Here's the place to explain them.

Further paragraphs come after blank lines.

- Bullet points are okay, too
- Typically a hyphen or asterisk is used for the bullet, preceded by a single space, with blank lines in between, but conventions vary here

If you use an issue tracker, put references to them at the bottom, like this:

Resolves: #123

See also: #456, #789



# Merge Conflict

```
14 <tr>
15 <td>Harvey</td>
16 <td>Jennings</td>
17 <td>Leaving Soon</td>
18 <td>hjennings@atlassian.com</td>
19 <==== HEAD
20 <td>2</td>
21 <td>B</td>
22 ===== B
23 <td>2-B</td>
24 </tr>
25 <tr>
26 <td>Ryan</td>
27 <td>Lee</td>
28 <td>New Hire</td>
29 <td>rlee@atlassian.com</td>
30 <td>5-E</td>
31 >>>>>> 2a04e55405e527fb7924888b3ee0336a24849cf1
32 </tr>
33 <tr>
34 <td>Alana</td>
35 <td>Grant</td>
36 <td>Current</td>
37 <td>agrانت@atlassian.com</td>
```

team\_contact\_info UNREGISTERED

Line 26, Column 18 Spaces: 2 Plain Text



Github GUI



**TOWER**

 Sublime Merge



# Task 1: Maven to Gradle

- Install Gradle
  - brew install gradle
  - sudo apt install gradle
- git clone <https://github.com/chupanw/kafka-stream.git>
- gradle init

- Open build.gradle
- Change the 'maven-publish' to 'application'
- Add the following line after the plugins section:

```
mainClassName = "myapps.LineSplit"
```

- Delete the entire 'publishing' section

- Edit .gitignore: <https://www.gitignore.io>
- `git rm pom.xml`
- `git status`
- `git add *`
- `git add .gitignore`
- `git commit`
- Sample solution: `git checkout gradle`

On branch gradle

Changes not staged for commit:

(use "git add <file>..." to update w

(use "git restore <file>..." to disc

modified: .gitignore

Untracked files:

(use "git add <file>..." to include

build.gradle

gradle/

gradlew

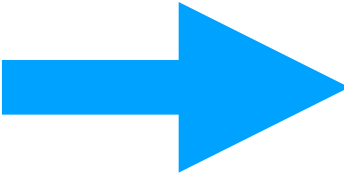
gradlew.bat

settings.gradle



**org.apache.kafka.streams.errors.StreamsException: Input record ConsumerRecord(topic = movielog, partition = 0, leaderEpoch = null, offset = 7996437, CreateTime = -1, serialized key size = -1, serialized value size = 63, headers = RecordHeaders(headers = [], isReadOnly = false), key = null, value = 2019-05-29T22:14:25,82547,GET /data/m/widows%27+peak+1994/7.mpg) has invalid (negative) timestamp. Possibly because a pre-0.10 producer client was used to write this record to Kafka without embedding a timestamp, or because the input topic was created before upgrading the Kafka cluster to 0.10+. Use a different TimestampExtractor to process this data.**

at org.apache.kafka.streams.processor.FailOnInvalidTimestamp.onInvalidTimestamp (FailOnInvalidTimestamp.java:73)  
at org.apache.kafka.streams.processor.ExtractRecordMetadataTimestamp.extract (ExtractRecordMetadataTimestamp.java:61)  
at org.apache.kafka.streams.processor.FailOnInvalidTimestamp.extract (FailOnInvalidTimestamp.java:48)  
at org.apache.kafka.streams.processor.internals.RecordQueue.updateHead (RecordQueue.java:167)  
at org.apache.kafka.streams.processor.internals.RecordQueue.addRowRecords (RecordQueue.java:100)  
at org.apache.kafka.streams.processor.internals.PartitionGroup.addRowRecords (PartitionGroup.java:136)  
at org.apache.kafka.streams.processor.internals.StreamTask.addRecords (StreamTask.java:746)  
at org.apache.kafka.streams.processor.internals.StreamThread.addRecordsToTasks (StreamThread.java:1023)  
at org.apache.kafka.streams.processor.internals.StreamThread.runOnce (StreamThread.java:861)  
at org.apache.kafka.streams.processor.internals.StreamThread.runLoop (StreamThread.java:805)  
at org.apache.kafka.streams.processor.internals.StreamThread.run (StreamThread.java:774)  
.....



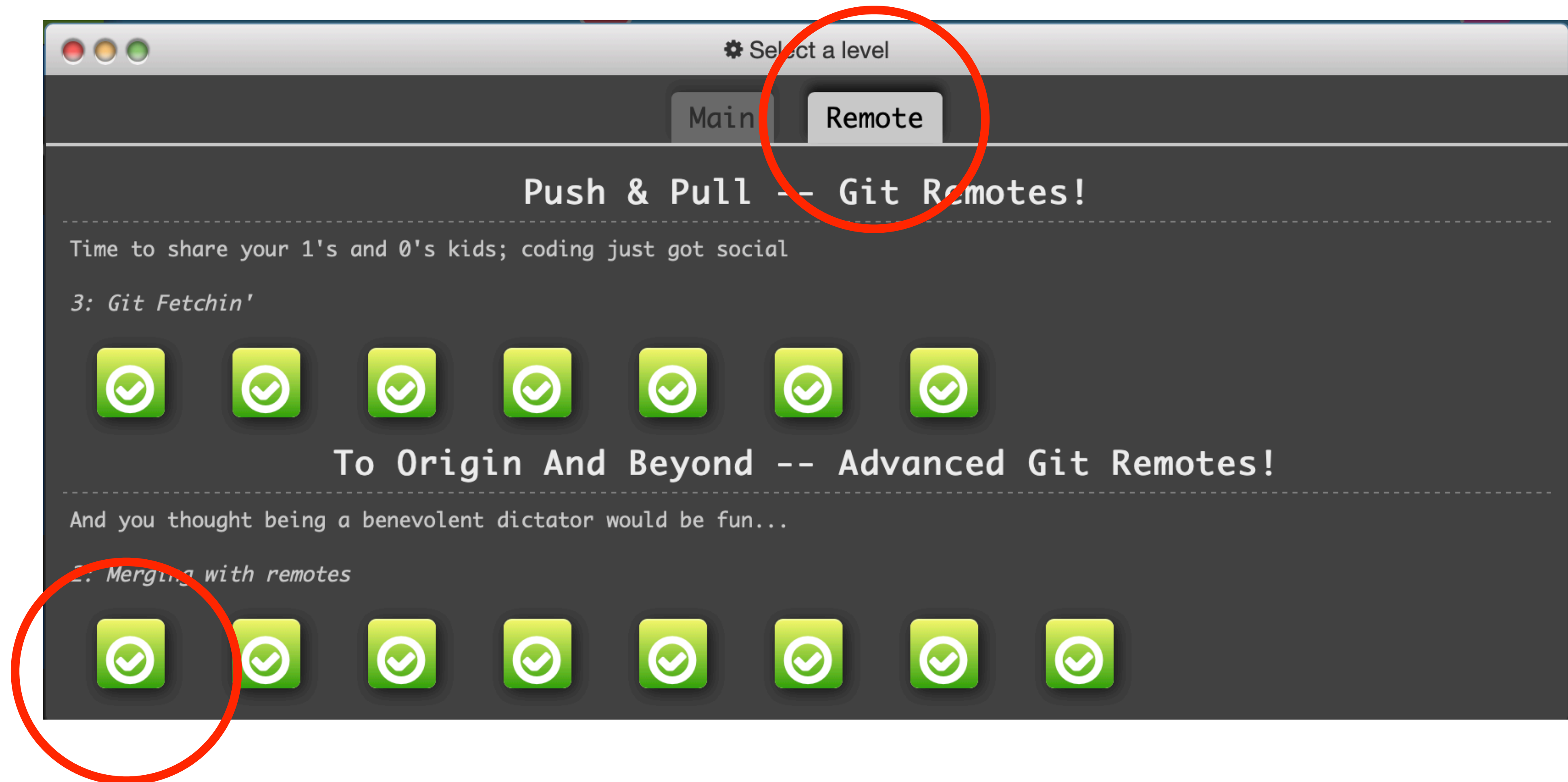
```
public static void main(String[] args) throws Exception {  
    Properties props = new Properties();  
    props.put(StreamsConfig.APPLICATION_ID_CONFIG, "test");  
    props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");  
    props.put(StreamsConfig.DEFAULT_TIMESTAMP_EXTRACTOR_CLASS_CONFIG, WallclockTimestampExtractor.class);  
    props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());  
    props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());  
}
```

# Git flow

# **Task 2:**

# **Rebase Feature Branches**

# <https://learngitbranching.js.org>





# **Task 3:**

# **Pull Request**

- Fork our course repo on Github: <https://github.com/ckaestne/seai.git>
- Clone your forked repo on your laptop
- Open README.md and find some typos to fix
- Commit, push to your forked repo, and send us a pull request
- You're contributing to our course!

# More Resources

- <https://git-scm.com/doc>
- <https://learngitbranching.js.org>
- <https://try.github.io>