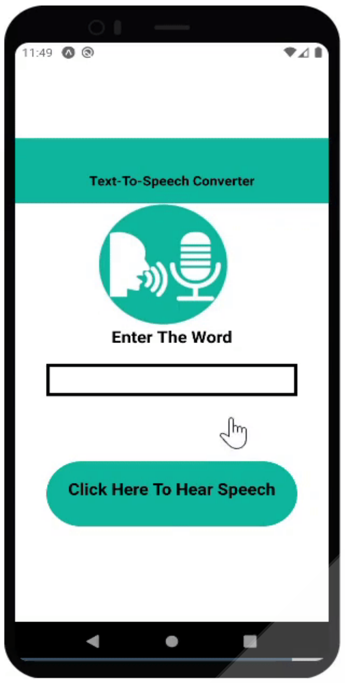
Topic	Git and making open source contributions	
Cl ass Description	Students learn to install git and practically use advanced features of git versioning tools - staging, branching, merging, comparing, resetting etc. - through writing a story. Students also learn to contribute to open source projects on github using git.	
Class	C67	
Class time	45 mins	
Goal	● Install git. ● Implement advanced git commands on our project git repository. ● Modify certain features of Monkey-Chunky App. ● Fork the Monkey Chunky Repository and make an open source contribution to the app.	
Resources Required	● Teacher Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Android/iOS Smartphone with Expo App installed ● Student Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Android/iOS Smartphone with Expo App installed	
Class structure	Warm Up Teacher-led Activity Student-led Activity Wrap up	5 mins 20 min 15 min 5 min

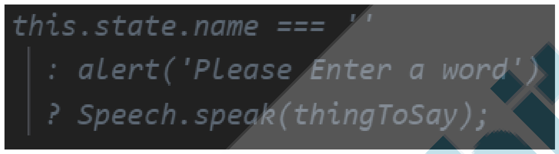
WARM-UP SESSION - 5 mins

CONTEXT

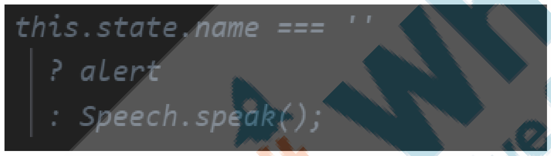
- **Review the git commands students learned earlier and how they are used by developers to collaboratively work on a project.**

Teacher starts slideshow from slides 1 to 11 Refer to speaker notes and follow the instructions on each slide.	
Activity details	**Solution/Guidelines**
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Run the presentation from slide 1 to slide 4** **Following are the WARM-UP session deliverables:** ● **Greet the student.** ● **Revision of previous class activities.** ● **Quizzes**	**ESR**: Hi, thanks, Yes I am excited about it! Click on the slide show tab and present the slides
QnA Session	
Question	**Answer**
Identify the correct option to import the Speech component. A. `import * as Speech from 'expo-speech';` B. `import * as from 'expo-speech';` C. `import * as Speech ;` D. `import * as Speech 'expo-speech';`	**A**
From the following options select the correct option to display an alert box when textInput is kept empty and the button is pressed.	**D**

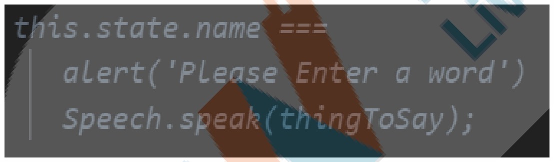


A. 

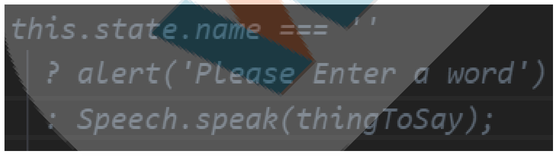
```
this.state.name === ''
  : alert('Please Enter a word')
  ? Speech.speak(thingToSay);
```

B. 

```
this.state.name === ''
  ? alert
  : Speech.speak();
```

C. 

```
this.state.name ===
  alert('Please Enter a word')
  Speech.speak(thingToSay);
```

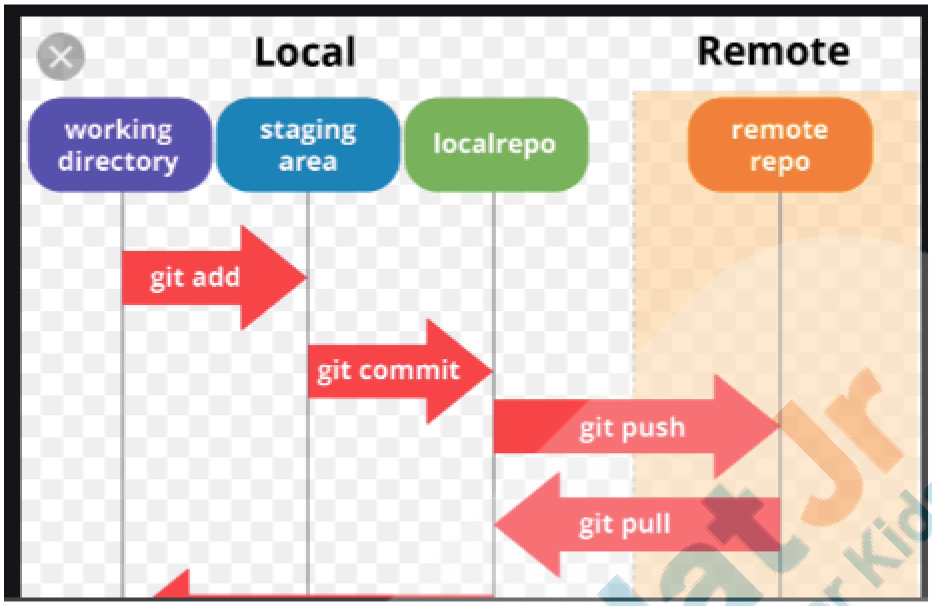
D. 

```
this.state.name === ''
  ? alert('Please Enter a word')
  : Speech.speak(thingToSay);
```

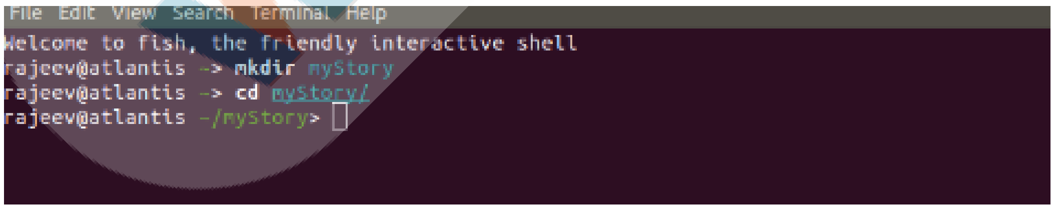
Continue the WARM-UP session	
Activity details	**Solution/Guidelines**
Run the presentation from slide 5 to slide 11 to set the problem statement.	Narrate the story by using hand gestures and voice modulation methods to

Following are the WARM-UP session deliverables: ● Appreciate the student. ● Recall the learnings of React Native.		bring in more interest in students.
Teacher ends slideshow		
TEACHER-LED ACTIVITY - 20 mins		
Teacher Initiates Screen Share		
CHALLENGE ● **Install git** ● **Write a story using advanced git commands.**		
Step 2: Teacher-led Activity (20 min)	Let's start by installing the git on our systems Teacher activity 1 *<Teacher follows the steps from the doc to install git on her system and guides the student to install alongside>* **The following are some of the basic Git commands.** **git clone <path of repository>** - This command helps to clone a git repository **git status** - This command gives the current status of the working repository **git add** - This command associates the working directory on our git bash shell to the remote repository. **git commit** - This command helps to commit the changes to the remote repository. **git push** - This command helps to push	*<Student installs git on his/her system>*

	the current changes to the github repository.	
	If you remember, git divides our work area into 3 stages: 1. **Working directory:** Where we are creating and making edits to our files. 2. **Staging area:** When the files are ready to be committed, we have to stage them first. 3. **Local GIt Repository:** After the files are committed, they become a part of commit history in a local repository. You can compare this with Get, Set, Go! First we GET the files we have modified, then we SET them up in the staging area and then we commit (GO) in our local repository. Then there is also a remote repository where we can push all our work so that anyone else (a team member / another developer) can access our work. We are quickly going to see this through an example and then things will get even more clear. Any questions so far?	The student asks questions and get clarifications.



	Alright then. Just like we code, let's try to write a story using git and see how git helps in keeping control of all the versions of our story (similar to different versions of code). Open a terminal (git bash). Let's make a new directory called "MyStory" using '**mkdir**' command and change into the directory (using cd command).	The student observes.

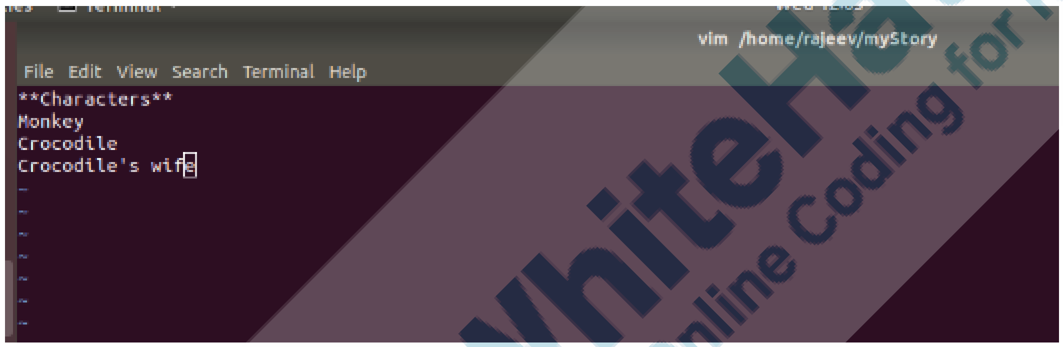


```
Welcome to fish, the friendly interactive shell
rajeev@atlantis ~> mkdir myStory
rajeev@atlantis ~> cd myStory/
rajeev@atlantis ~/myStory>
```

	Let's navigate to the said folder and create two empty files called "**Characters**" and "**Story**".	The student gives inputs on the story and the characters.

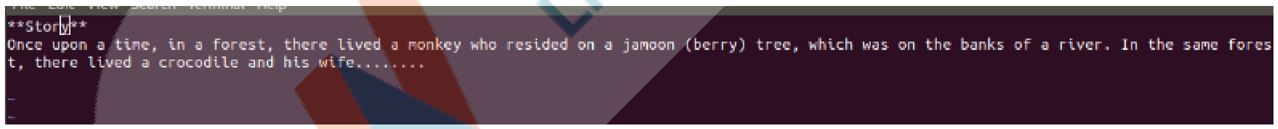
	You can open these files in any editor and start writing. *Note: You can be interactive with the student and create a new story with different characters based on student inputs. Let's initialize git using '**git init**' so that git can start tracking changes in our files.	

Characters File:



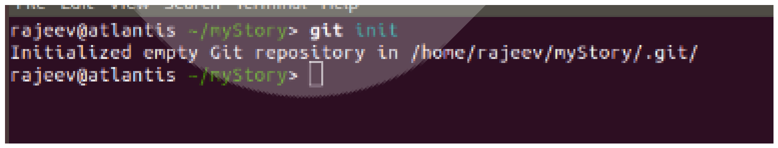
```
**Characters**
Monkey
Crocodile
Crocodile's wife
~
~
~
~
~
~
~
```

Story File:



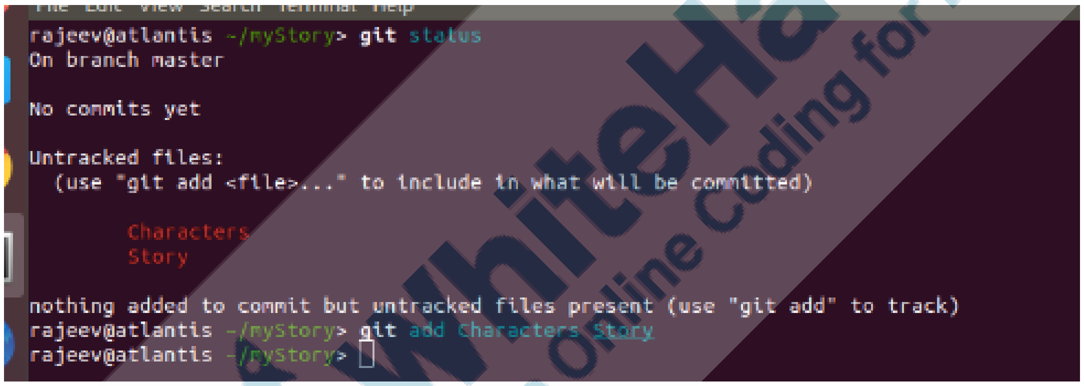
```
**Story**
Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same fores
t, there lived a crocodile and his wife........
~
~
```

Git Init:



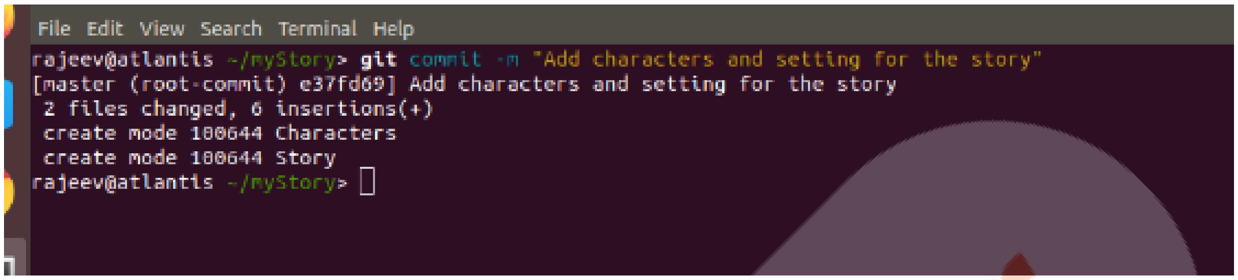
```
rajeev@atlantis ~/myStory> git init
Initialized empty Git repository in /home/rajeev/myStory/.git/
rajeev@atlantis ~/myStory>
```

	The changes we are making in the files are in the working directory. We can make as many changes as we want here.	The student learns how to use git add to add files to the staging area.

	If we want to commit these changes to the history in our local repo, we need to stage these changes. We can simply type git status to check the status of our git working directory. You can see that there are two files which need to be added to the staging area. We can do that using **git add [filename1] [filename2]**	
		
	We can continue to make more changes in the file. For every change in a file, we will have to change the stages again. Otherwise git will have the snapshot of the older staged file in its memory. At one point of time, we will want to commit the changes to our local repository's history. We can do that using **git commit-m "a message"**	The student learns how to use git add to add files to the staging area.

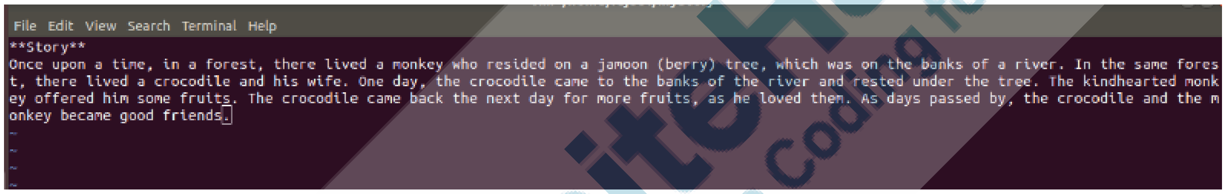
```
rajeev@atlantis ~/myStory> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Characters
        Story

nothing added to commit but untracked files present (use "git add" to track)
rajeev@atlantis ~/myStory> git add Characters Story
rajeev@atlantis ~/myStory> 
```

	Teacher shows how to commit a message.	

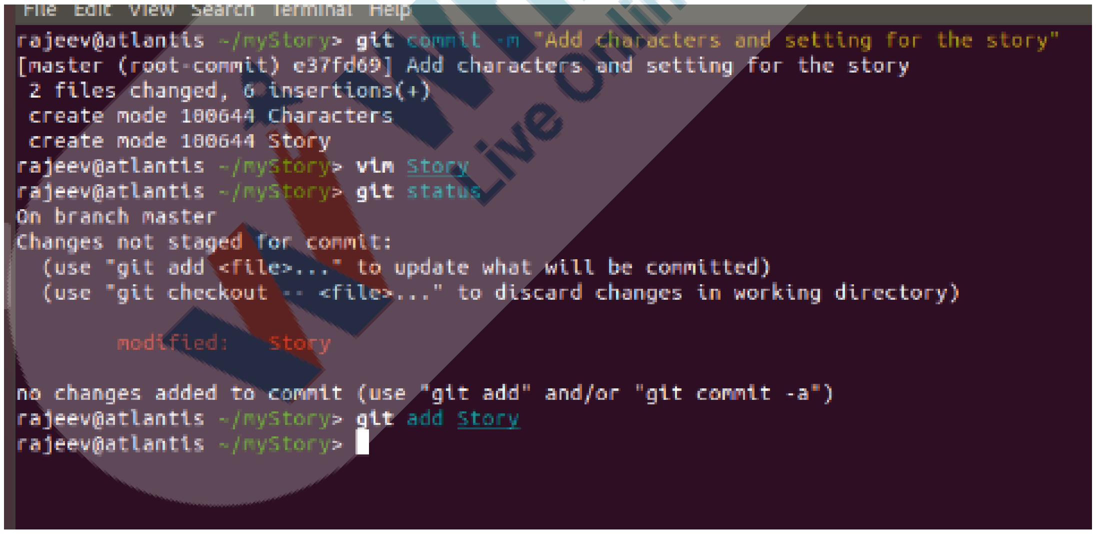


```
rajeev@atlantis ~/myStory> git commit -m "Add characters and setting for the story"
[master (root-commit) e37fd69] Add characters and setting for the story
 2 files changed, 6 insertions(+)
 create mode 100644 Characters
 create mode 100644 Story
rajeev@atlantis ~/myStory>
```

	Let's make a few more changes in our story and stage the changes again. Can you help me do that?	The student helps the teacher in adding to the story and then staging the change again.



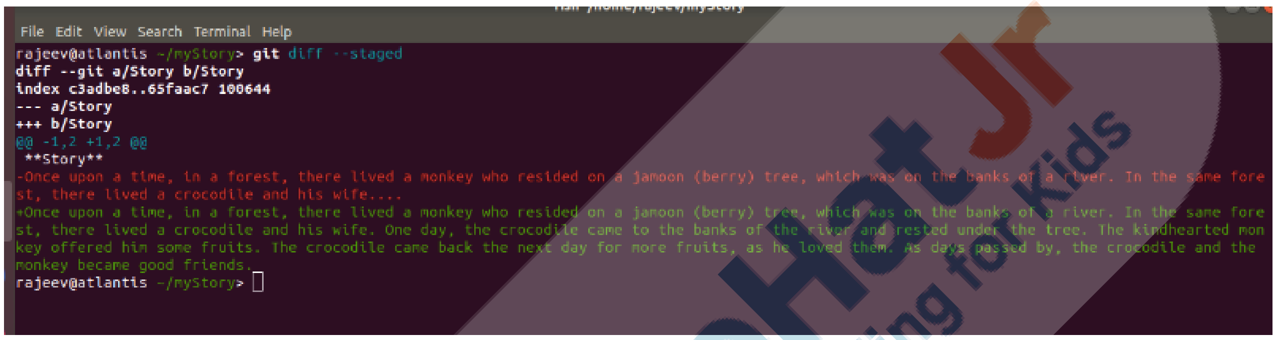
```
**Story**
Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same fores
t, there lived a crocodile and his wife. One day, the crocodile came to the banks of the river and rested under the tree. The kindhearted monk
ey offered him some fruits. The crocodile came back the next day for more fruits, as he loved them. As days passed by, the crocodile and the m
onkey became good friends.
~
~
~
~
```



```
rajeev@atlantis ~/myStory> git commit -m "Add characters and setting for the story"
[master (root-commit) e37fd69] Add characters and setting for the story
 2 files changed, 6 insertions(+)
 create mode 100644 Characters
 create mode 100644 Story
rajeev@atlantis ~/myStory> vim Story
rajeev@atlantis ~/myStory> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Story

no changes added to commit (use "git add" and/or "git commit -a")
rajeev@atlantis ~/myStory> git add Story
rajeev@atlantis ~/myStory>
```

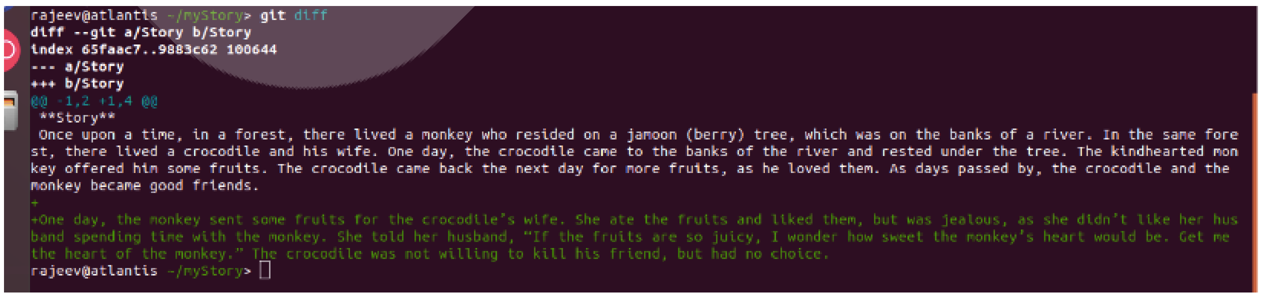
	There is a beautiful command using which you can see the difference between the two files since the last commit and the current staged file.	Student observes and asks questions

	You can do this using **'git diff --staged'**. This will show you the difference between the last committed file and the currently staged file. Teacher writes the command to show how git diff works.	



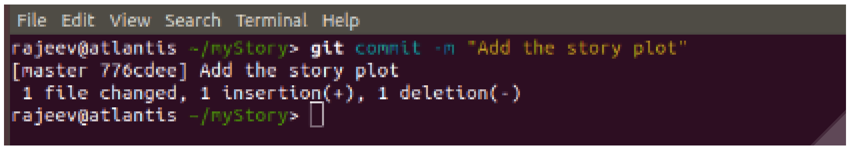
```
rajeev@atlantis ~/myStory> git diff --staged
diff --git a/Story b/Story
index c3adbe8..65faac7 100644
--- a/Story
+++ b/Story
@@ -1,2 +1,2 @@
 **Story**
-Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same fore
st, there lived a crocodile and his wife....
+Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same fore
st, there lived a crocodile and his wife. One day, the crocodile came to the banks of the river and rested under the tree. The kindhearted mon
key offered him some fruits. The crocodile came back the next day for more fruits, as he loved them. As days passed by, the crocodile and the
monkey became good friends.
rajeev@atlantis ~/myStory> 
```

	You can also make more changes in the working directory and see the difference between your working directory and the currently staged file. We do that using **"git diff"** git diff is very helpful if a developer is looking at changes made by someone else in their code since they last worked on it. Teacher shows how to do this.	The student learns how to use git diff.

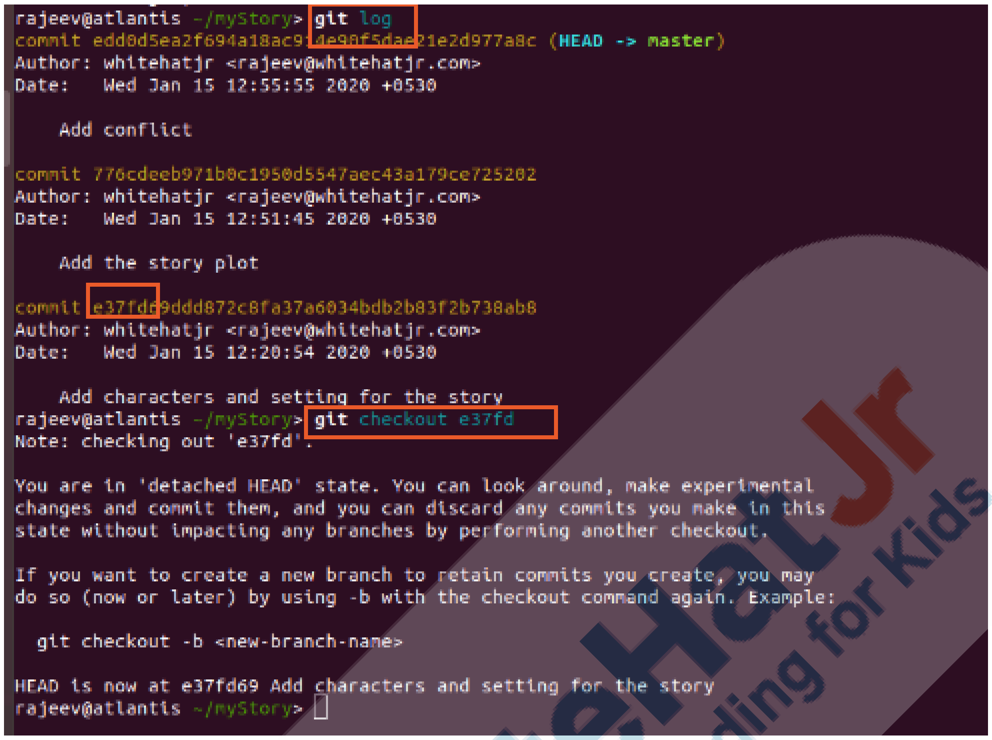


```
rajeev@atlantis ~/myStory> git diff
diff --git a/Story b/Story
index 65faac7..9883c62 100644
--- a/Story
+++ b/Story
@@ -1,2 +1,4 @@
 **Story**
 Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same fore
st, there lived a crocodile and his wife. One day, the crocodile came to the banks of the river and rested under the tree. The kindhearted mon
key offered him some fruits. The crocodile came back the next day for more fruits, as he loved them. As days passed by, the crocodile and the
monkey became good friends.
+
+One day, the monkey sent some fruits for the crocodile's wife. She ate the fruits and liked them, but was jealous, as she didn't like her hus
band spending time with the monkey. She told her husband, "If the fruits are so juicy, I wonder how sweet the monkey's heart would be. Get me
the heart of the monkey." The crocodile was not willing to kill his friend, but had no choice.
rajeev@atlantis ~/myStory> 
```

	We can commit the file to the local repo again with a commit message. Can you help me? How do I do this?	The student guides the teacher on how to commit to the local repo.

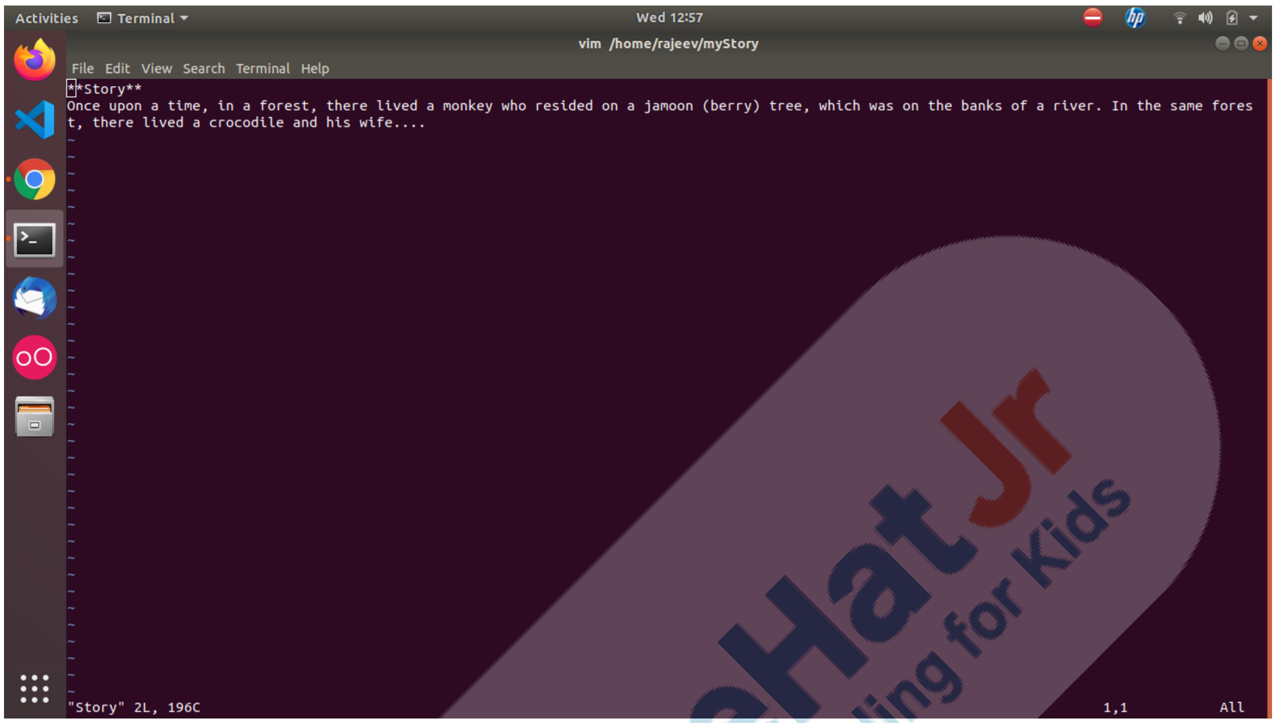

`rajeev@atlantis ~/myStory> git commit -m "Add the story plot"`
`[master 776cdee] Add the story plot`
` 1 file changed, 1 insertion(+), 1 deletion(-)`
`rajeev@atlantis ~/myStory>`		

	Let's say you don't like the current story plot which you have written and want to move back to the older commit. This should be possible right? You can see all your commits using the **git log** command. You can also see the time of the commit, the author and the commit message. There is also a **commit id** given to each commit. You can use the id or even the first 5 characters of the id to move back to that commit using **git checkout <commit id>** Teacher shows how to use the **git checkout**.	ESR: Yes!



```
rajeev@atlantis ~/myStory> git log
commit edd0d5ea2f694a18ac914e90f5dae21e2d977a8c (HEAD -> master)
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:55:55 2020 +0530

    Add conflict

commit 776cdeeb971b0c1950d5547aec43a179ce725202
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:51:45 2020 +0530

    Add the story plot

commit e37fd69ddd872c8fa37a6034bdb2b83f2b738ab8
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:20:54 2020 +0530

    Add characters and setting for the story
rajeev@atlantis ~/myStory> git checkout e37fd
Note: checking out 'e37fd'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at e37fd69 Add characters and setting for the story
rajeev@atlantis ~/myStory> 
```

	You can open the Story file and see that it contains the text you had written in your first commit! Isn't that amazing! This way you can move around to any commit history you have made. If you are properly committing your work for every story line (or every feature while writing code), you can fearlessly experiment with your work without having the fear of losing your work!!	The student asks any questions he/she has on their minds.

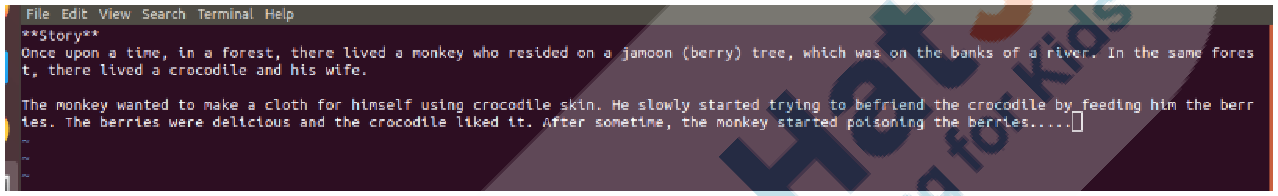


```
*Story**
Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same fores
t, there lived a crocodile and his wife....
"Story" 2L, 196C                                                                                                   1,1           All
```

	Now, there is a possibility that you might want to explore another plot in your story or add new characters. You can do this by creating a new branch here. Your current branch was called 'master' by default. We all know this story where the monkey is the hero and crocodile's wife is the villain. Let's flip the story in our new branch. Let's make the crocodile's wife a hero and monkey the villain. You can create a new branch using **git branch <branchName>** You can also see all the branches in your local repo using the **'git branch'** command. Your current branch will be shown with *	The student understands about git branching and git checkout to a new branch.

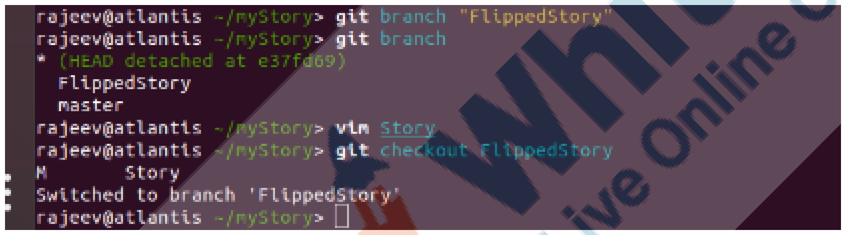
	. After creating the branch, you will also have to checkout to that branch using git checkout command. This will move you to the FlippedStory branch. Let's write our flipped story here, stage the file and write a new commit.	

Flipped Story:

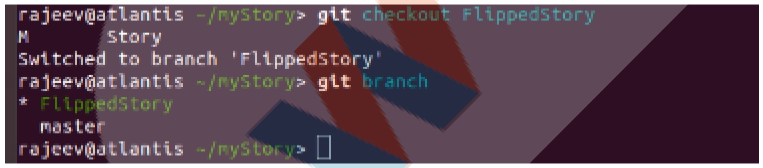


```
**Story**
Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same fores
t, there lived a crocodile and his wife.

The monkey wanted to make a cloth for himself using crocodile skin. He slowly started trying to befriend the crocodile by feeding him the berr
ies. The berries were delicious and the crocodile liked it. After sometime, the monkey started poisoning the berries.....
~
~
~
```

Creating Git branch:

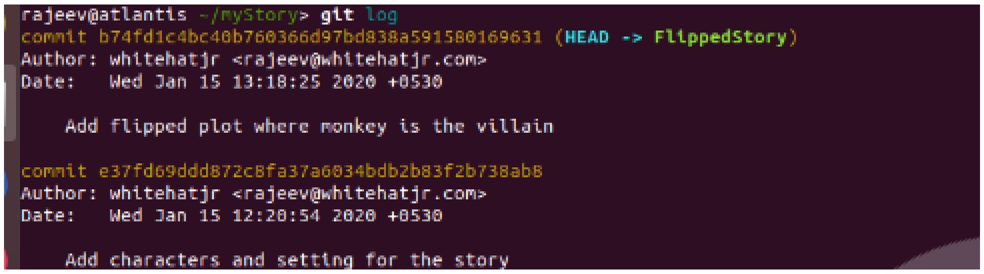


```
rajeev@atlantis ~/myStory> git branch "FlippedStory"
rajeev@atlantis ~/myStory> git branch
* (HEAD detached at e37fd69)
  FlippedStory
  master
rajeev@atlantis ~/myStory> vim Story
rajeev@atlantis ~/myStory> git checkout FlippedStory
M       Story
Switched to branch 'FlippedStory'
rajeev@atlantis ~/myStory> 
```



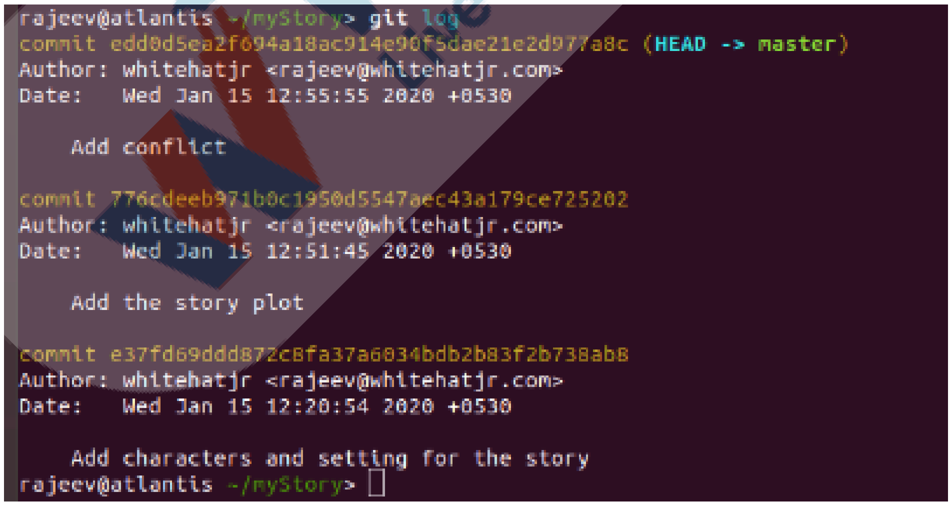
```
rajeev@atlantis ~/myStory> git checkout FlippedStory
M       Story
Switched to branch 'FlippedStory'
rajeev@atlantis ~/myStory> git branch
* FlippedStory
  master
rajeev@atlantis ~/myStory> 
```

	If you currently press **git log**, it will show your the commits log for only the current branch. Teacher does ggit lo and shows the output.	



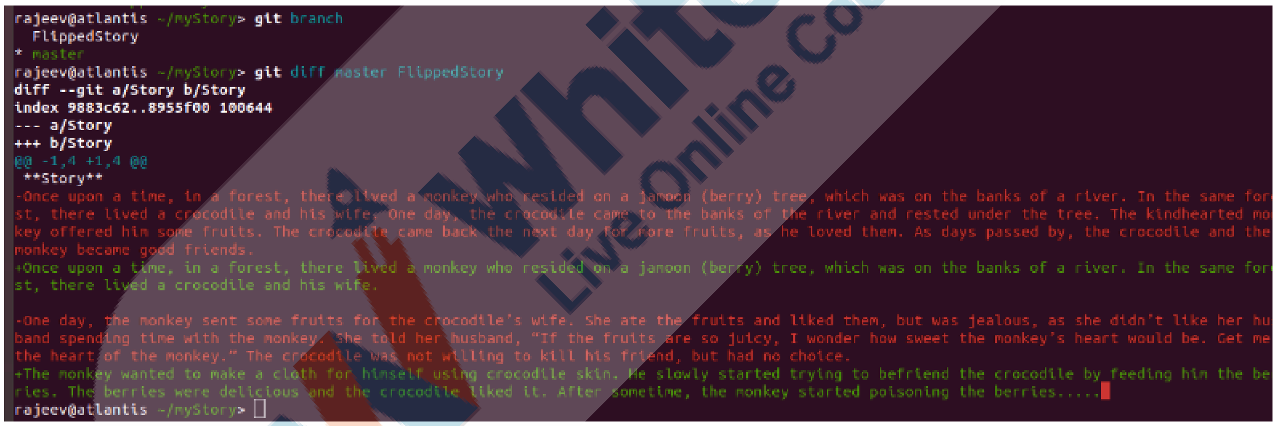
```
rajeev@atlantis ~/myStory> git log
commit b74fd1c4bc40b760366d97bd838a591580169631 (HEAD -> FlippedStory)
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 13:18:25 2020 +0530

    Add flipped plot where monkey is the villain

commit e37fd69ddd872c8fa37a6034bdb2b83f2b738ab8
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:20:54 2020 +0530

    Add characters and setting for the story
```

	At any point of time, you can abandon this branch and move to the master branch using '**git checkout master**'. It will take you to the last commit you had made in the master branch. Teacher shows how to checkout to a master branch. Later you can switch to FlippedBranch anytime and continue writing the story wherever you have left. Think how this would be useful when we are coding?	The student thinks about the use of branching in writing code. ● When we are working on a new feature in our code. ● When we are re-thinking the idea of what our app does and want to test different features etc.



```
rajeev@atlantis ~/myStory> git log
commit edd0d5ea2f694a18ac914e90f5dae21e2d977a8c (HEAD -> master)
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:55:55 2020 +0530

    Add conflict

commit 776cdeeb971b0c1950d5547aec43a179ce725202
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:51:45 2020 +0530

    Add the story plot

commit e37fd69ddd872c8fa37a6034bdb2b83f2b738ab8
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:20:54 2020 +0530

    Add characters and setting for the story
rajeev@atlantis ~/myStory> 
```

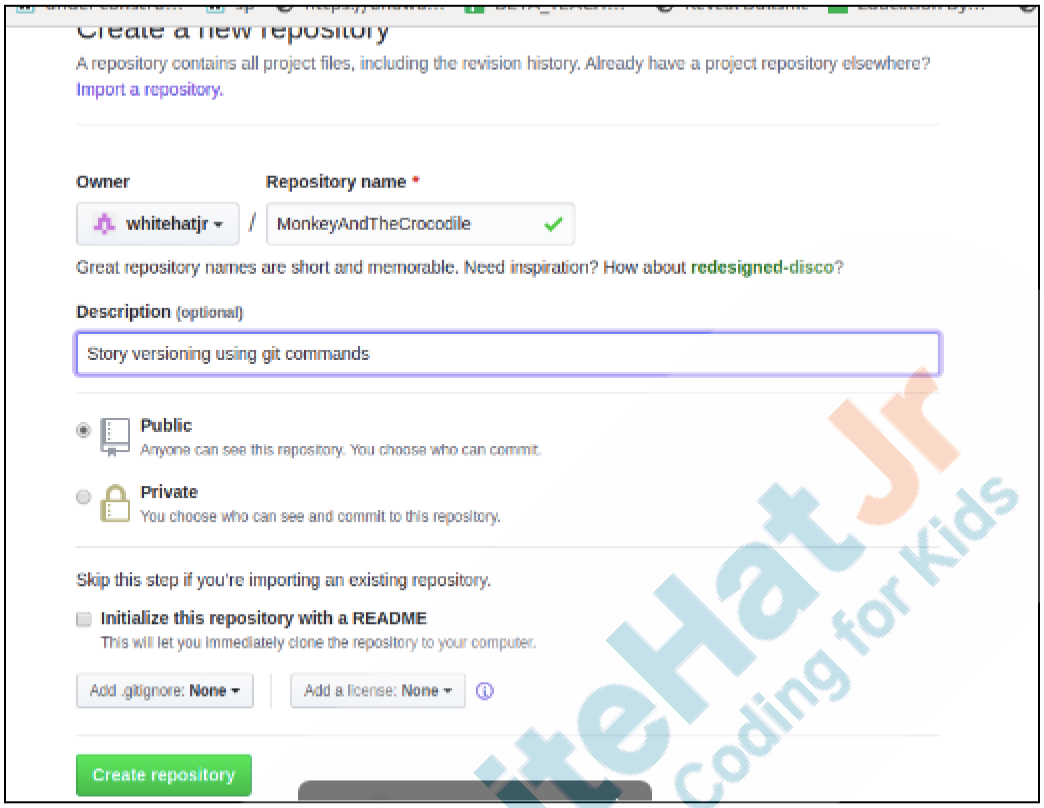
	You can check the difference between the lines in the HEAD of the two branches using **git diff branchA branchB** HEAD here refers to the latest commit of the two branches. Teacher shows how git diff works with branches. Sometimes two branches can work towards the same feature and then later they can be merged using **git merge** command. But here the files are too different. We can explore git merge sometime later.	



```
rajeev@atlantis ~/myStory> git branch
  FlippedStory
* master
rajeev@atlantis ~/myStory> git diff master FlippedStory
diff --git a/Story b/Story
index 9883c62..8955f00 100644
--- a/Story
+++ b/Story
@@ -1,4 +1,4 @@
 **Story**
-Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same fore
st, there lived a crocodile and his wife. One day, the crocodile came to the banks of the river and rested under the tree. The kindhearted mon
key offered him some fruits. The crocodile came back the next day for more fruits, as he loved them. As days passed by, the crocodile and the
monkey became good friends.
+Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same fore
st, there lived a crocodile and his wife.

-One day, the monkey sent some fruits for the crocodile’s wife. She ate the fruits and liked them, but was jealous, as she didn’t like her hus
band spending time with the monkey. She told her husband, “If the fruits are so juicy, I wonder how sweet the monkey’s heart would be. Get me
the heart of the monkey.” The crocodile was not willing to kill his friend, but had no choice.
+The monkey wanted to make a cloth for himself using crocodile skin. He slowly started trying to befriend the crocodile by feeding him the ber
ries. The berries were delicious and the crocodile liked it. After sometime, the monkey started poisoning the berries.....
rajeev@atlantis ~/myStory> 
```

	Now, let's upload all our commits to a remote repository (also called upstream repository). We will first need to create an empty Github repository. You already know how to do that and you can guide me.	The student guides the teacher to create a new empty git repository.

WhiteHat Jr
Live online Coding for Kids



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: whitehatjr / Repository name *: MonkeyAndTheCrocodile ✓

Great repository names are short and memorable. Need inspiration? How about **redesigned-disco**?

Description (optional): Story versioning using git commands

● **Public** — Anyone can see this repository. You choose who can commit.

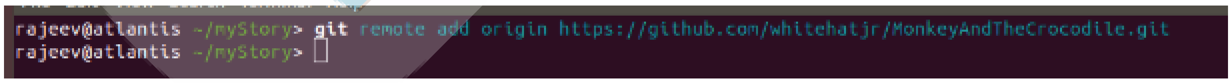
○ **Private** — You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README** — This will let you immediately clone the repository to your computer.

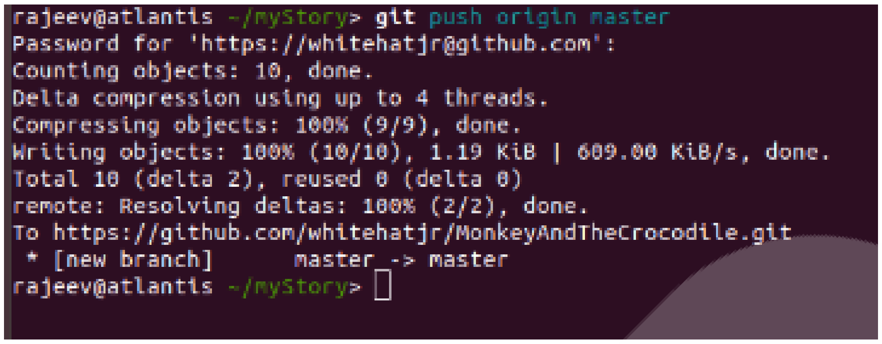
Add .gitignore: **None** | Add a license: **None**

	Awesome! Now we need to set the remote (upstream) repository for our local repo. You have already done that before. Can you guide me on how to do that? Teacher sets the upstream repository for the local repo.	We can do that using **git remote add [remoteName] [remoteURL]** We can give any name to remote.



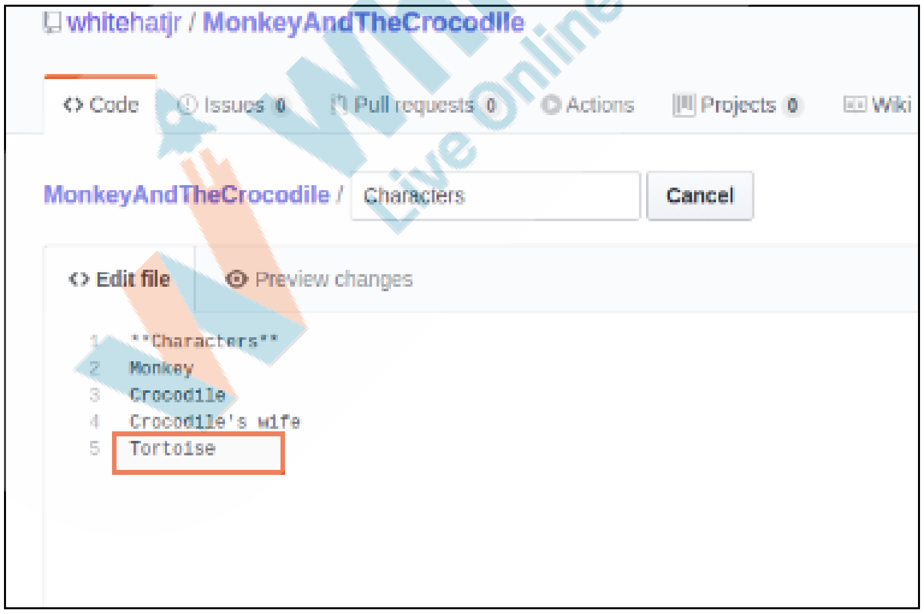
```
rajeev@atlantis ~/myStory> git remote add origin https://github.com/whitehatjr/MonkeyAndTheCrocodile.git
rajeev@atlantis ~/myStory> 
```

	You know how to push the current code to the remote repo using git push.	The student guides the teacher on how to push the local git repo to the upstream repo.

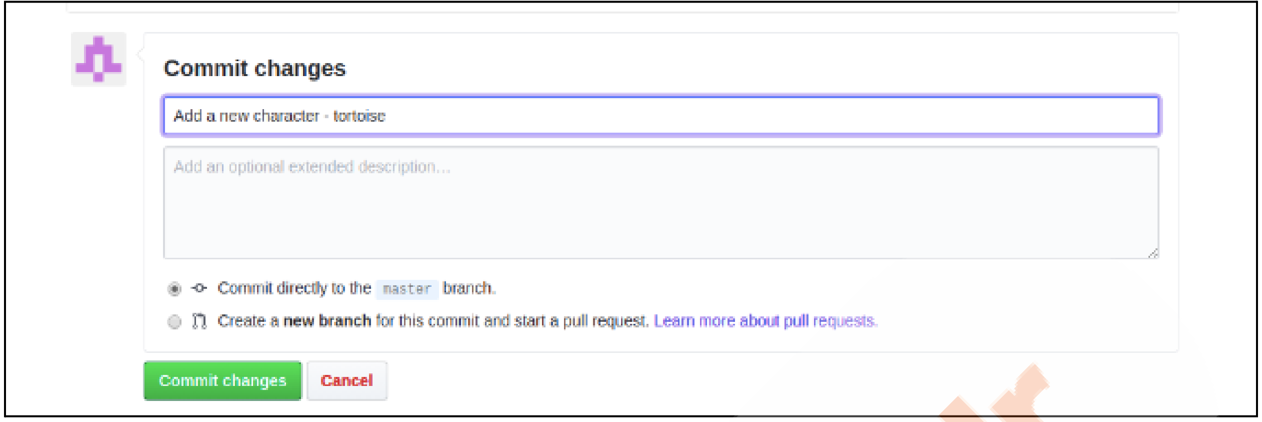


```
rajeev@atlantis ~/myStory> git push origin master
Password for 'https://whitehatjr@github.com':
Counting objects: 10, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 1.19 KiB | 609.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/whitehatjr/MonkeyAndTheCrocodile.git
 * [new branch]      master -> master
rajeev@atlantis ~/myStory>
```

	Now, what if some other developer made some changes in the remote repo? Your local repo will be out of sync then. Let's try to do this. Let's manually change something on the remote repo. Teacher changes a file on the remote repo and makes a commit.	The student observes.



```
Characters
**Characters**
Monkey
Crocodile
Crocodile's wife
Tortoise
```


Commit changes

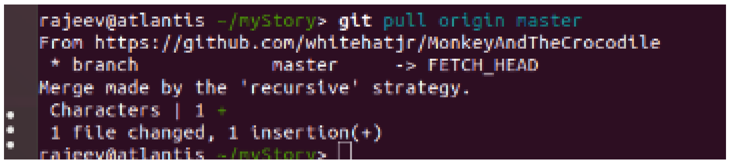
Add a new character - tortoise

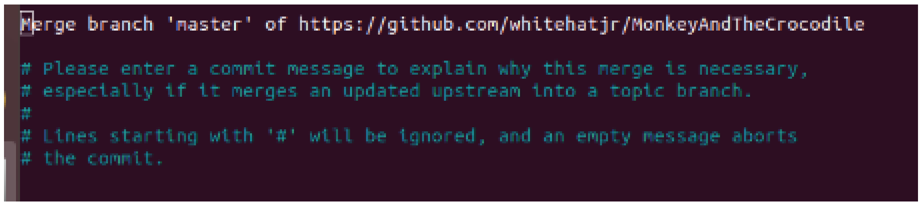
Add an optional extended description...

● Commit directly to the `master` branch.

○ Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

	Now, we will not be able to make any new pushes to the remote repo since our local repo is out of sync. We can use **git pull** to update our repo with the upstream repo. Teacher shows how to use **git pull [remoteName] [currentBranchName]**. *Note: If there are too many conflicts in the current working directory vs the remote repo, you will be asked to make a new commit. Make a new commit and then apply the git push. Now you can open the working directory to see that the current repo is updated with the new name of the character. The teacher can open the characters page and show the student that the character names have been updated.	The student observes and asks questions.


```
rajeev@atlantis ~/myStory> git pull origin master
From https://github.com/whitehatjr/MonkeyAndTheCrocodile
 * branch            master     -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 Characters | 1 +
 1 file changed, 1 insertion(+)
rajeev@atlantis ~/myStory> 
```



```
Merge branch 'master' of https://github.com/whitehatjr/MonkeyAndTheCrocodile

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
```

	This is a lot to learn about git in a single day. As you practice using git more, you will become more comfortable with how git works. We also have a great visual tutorial for most of the git commands which is very helpful for visualizing how git commands work. Why don't you try this?	

Teacher Stops Screen Share

STUDENT-LED ACTIVITY - 25 mins

- **Ask Student to press ESC key to come back to panel**
- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

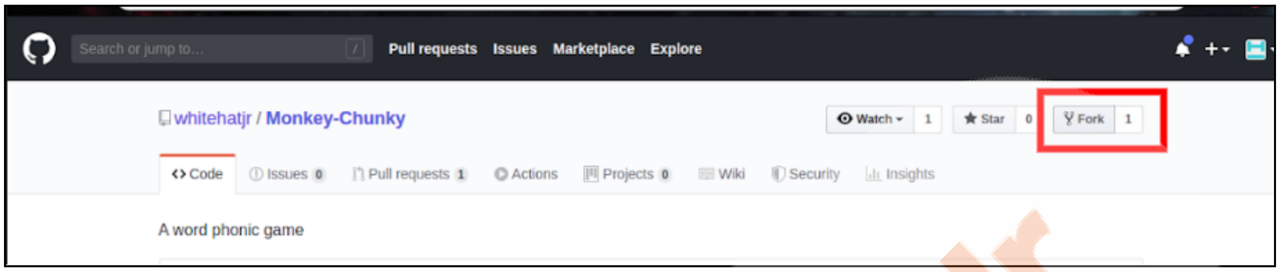
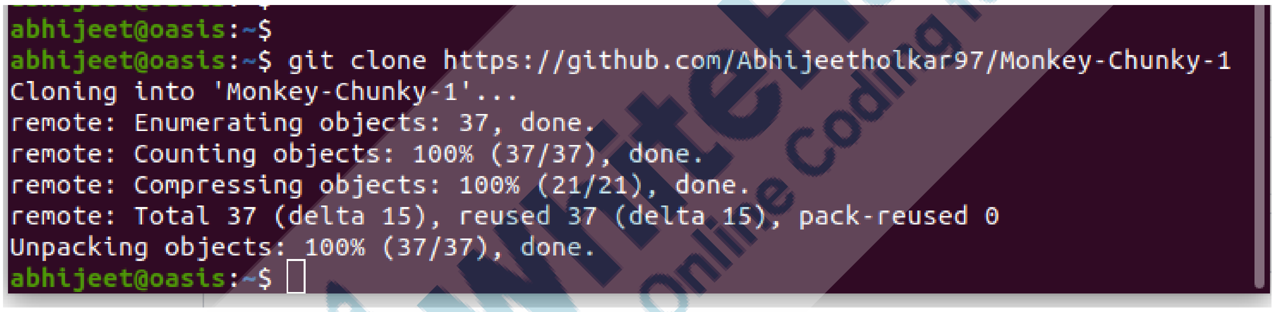
ACTIVITY

- **Fork the Monkey Chunky Repository and make an open source contribution to the project.**

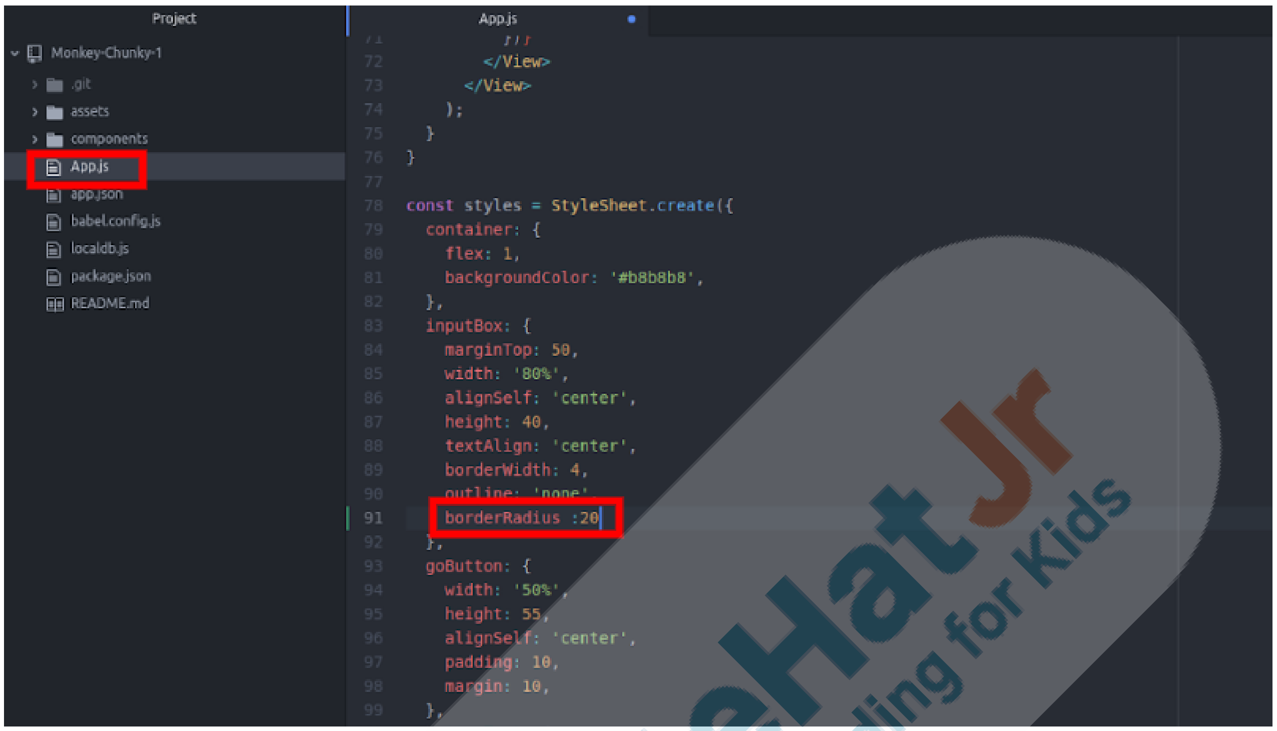
Teacher starts slideshow :Slide 12 to 13
Refer to speaker notes and follow the instructions on each slide.

	Now it's your turn. Please share your screen with me.	

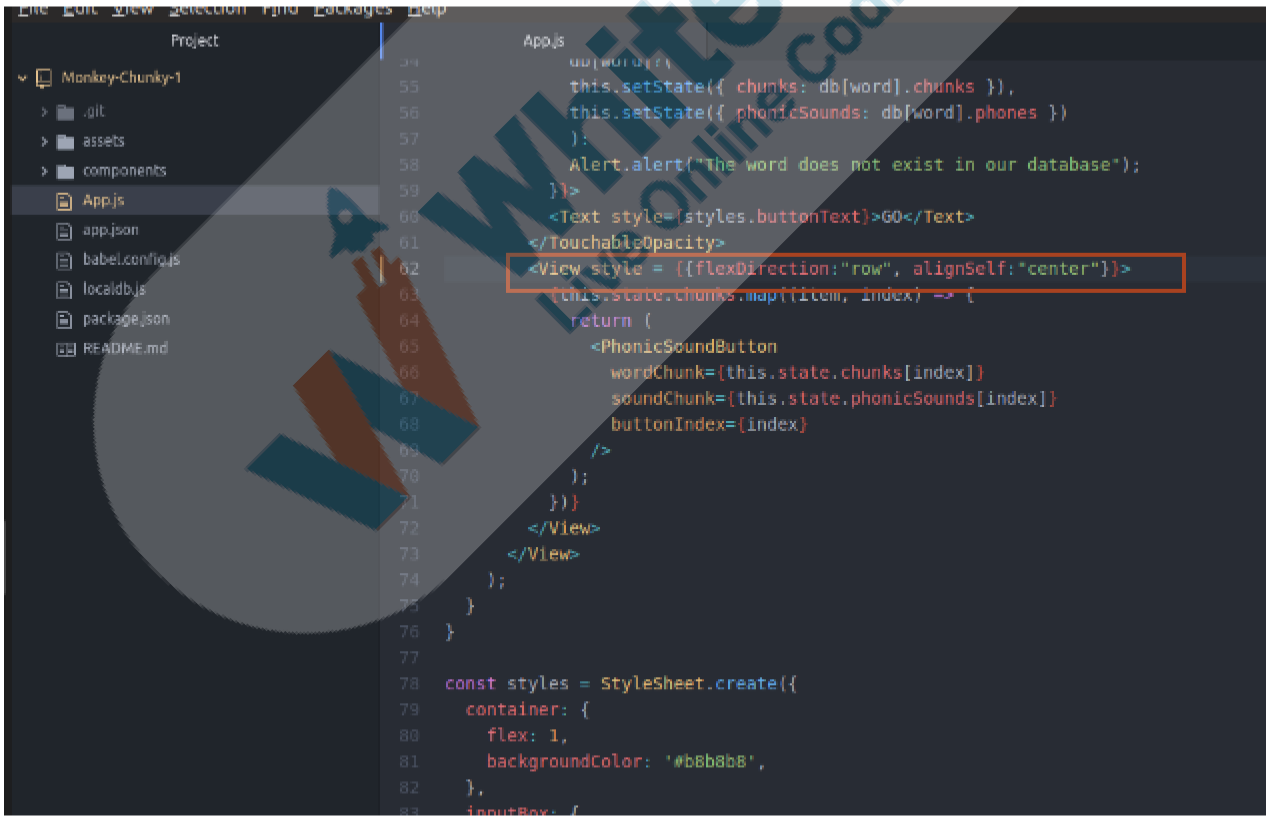
Teacher ends slideshow		
Step 3: Student-Led Activity (15 min)	Teacher helps the student go through the instruction and tutorials for **Student Activity1**	The student spends some time going through each of the git commands, practicing and visualizing the commits. The student asks questions where needed.
	Awesome! Did this exercise help in bringing more clarity and understanding about Monkey Chunky App?	ESR: Yes!
	Alright. Now that we know how to use git, we are going to learn how to make contributions to open source projects. Look at Student Activity 2. It is the Monkey Chunky App which is released as an open source project.	Student looks at **Student Activity 2**
	This repository contains all the commits made while the developer worked on the Monkey Chunky App. To make a contribution to the app, we must first fork the project. Forking creates a duplicate of the repository in your own account where you can work and modify the contents of the project. Teacher guides the student to fork the project.	The student forks the project.

	*Note: Forking the project can take a little time.	
 Search or jump to... Pull requests Issues Marketplace Explore whitehatjr / Monkey-Chunky Watch 1 Star 0 Fork 1 Code Issues 0 Pull requests 1 Actions Projects 0 Wiki Security Insights A word phonic game		
	This is the forked repository. You can clone this repository into your system.	The student uses git clone to clone the repository into his/her system.
		
	Now cd into your forked repo. Let's make some changes into this project. Some example changes you can make are: 1. You can make the input box rounded. 2. You can make the phonic chunk buttons in a horizontal row instead of column by changing the flex-direction.	The student makes edits to the file to make these changes.

```
abhijeet@oasis:~$
abhijeet@oasis:~$ git clone https://github.com/Abhijeetholkar97/Monkey-Chunky-1
Cloning into 'Monkey-Chunky-1'...
remote: Enumerating objects: 37, done.
remote: Counting objects: 100% (37/37), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 37 (delta 15), reused 37 (delta 15), pack-reused 0
Unpacking objects: 100% (37/37), done.
abhijeet@oasis:~$
```

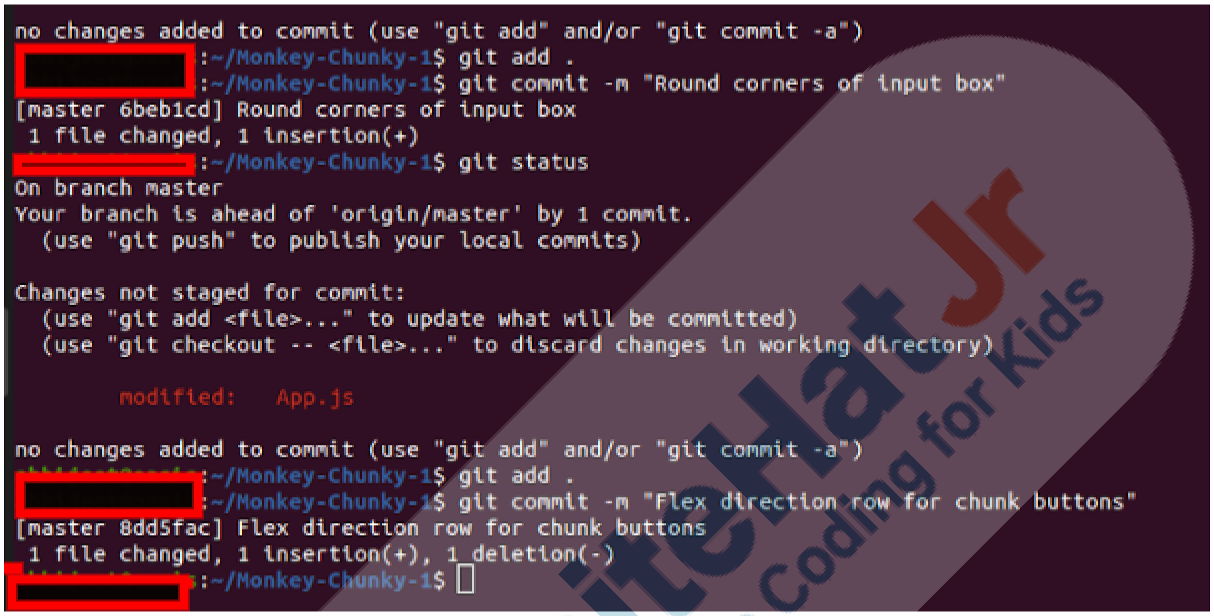


```
      </View>
    </View>
  );
 }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
  },
  inputBox: {
    marginTop: 50,
    width: '80%',
    alignSelf: 'center',
    height: 40,
    textAlign: 'center',
    borderWidth: 4,
    outline: 'none',
    borderRadius :20
  },
  goButton: {
    width: '50%',
    height: 55,
    alignSelf: 'center',
    padding: 10,
    margin: 10,
  },
```



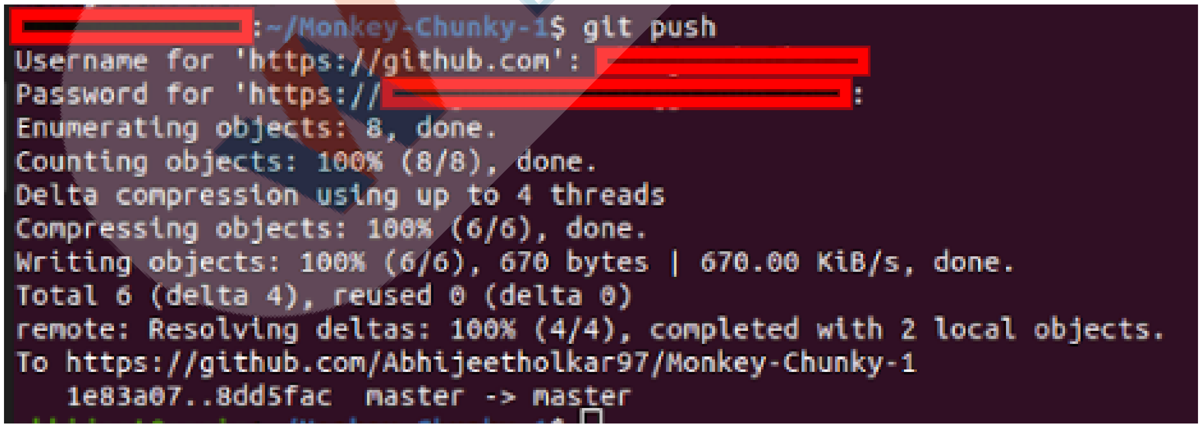
```
            this.setState({ chunks: db[word].chunks }),
            this.setState({ phonicSounds: db[word].phones })
          ):
          Alert.alert("The word does not exist in our database");
        }}>
        <Text style={styles.buttonText}>GO</Text>
      </TouchableOpacity>
      <View style = {{flexDirection:"row", alignSelf:"center"}}>
        {this.state.chunks.map((item, index) => {
          return (
            <PhonicSoundButton
              wordChunk={this.state.chunks[index]}
              soundChunk={this.state.phonicSounds[index]}
              buttonIndex={index}
            />
          );
        })}
      </View>
    </View>
  );
 }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
  },
```

	Finally, you can stage and commit these changes.	The student stages the files and commits the changes with a commit message.



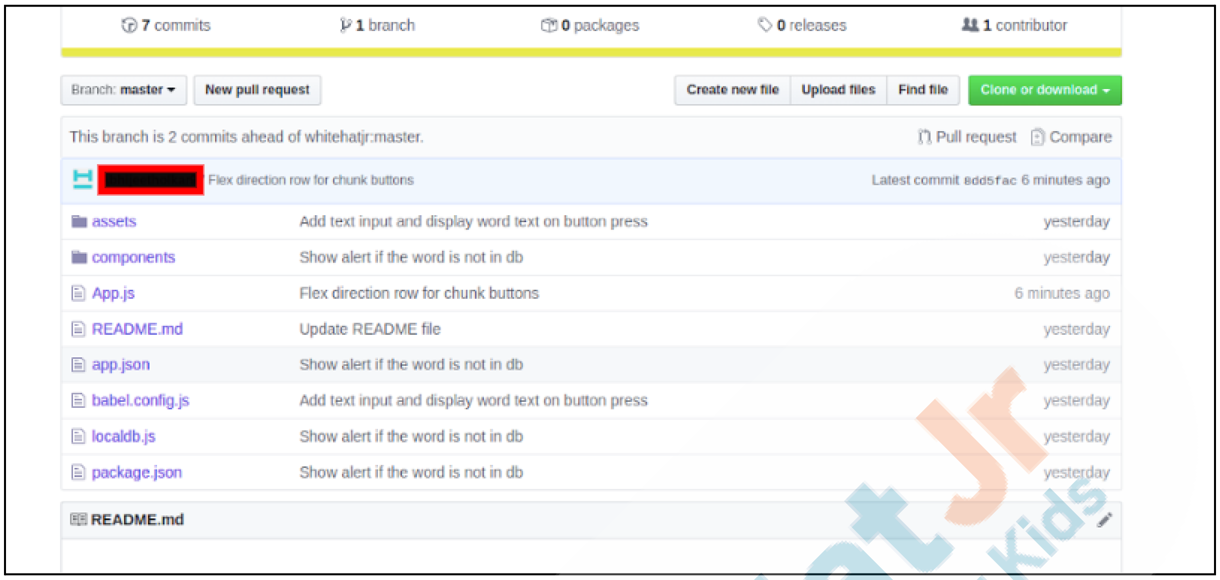
```
no changes added to commit (use "git add" and/or "git commit -a")
[redacted]:~/Monkey-Chunky-1$ git add .
[redacted]:~/Monkey-Chunky-1$ git commit -m "Round corners of input box"
[master 6beb1cd] Round corners of input box
 1 file changed, 1 insertion(+)
[redacted]:~/Monkey-Chunky-1$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   App.js

no changes added to commit (use "git add" and/or "git commit -a")
[redacted]:~/Monkey-Chunky-1$ git add .
[redacted]:~/Monkey-Chunky-1$ git commit -m "Flex direction row for chunk buttons"
[master 8dd5fac] Flex direction row for chunk buttons
 1 file changed, 1 insertion(+), 1 deletion(-)
[redacted]:~/Monkey-Chunky-1$ 
```

	Push the new commits to your forked repo.	The student cd into the forked local repo and pushes the commits to the remote repo.



```
[redacted]:~/Monkey-Chunky-1$ git push
Username for 'https://github.com': [redacted]
Password for 'https://[redacted]:
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 670 bytes | 670.00 KiB/s, done.
Total 6 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To https://github.com/Abhijeetholkar97/Monkey-Chunky-1
   1e83a07..8dd5fac  master -> master
```

	Awesome. let's see if the new commits are added in the remote forked repo.	The student sees the new commits added to the remote forked repo.



7 commits | 1 branch | 0 packages | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

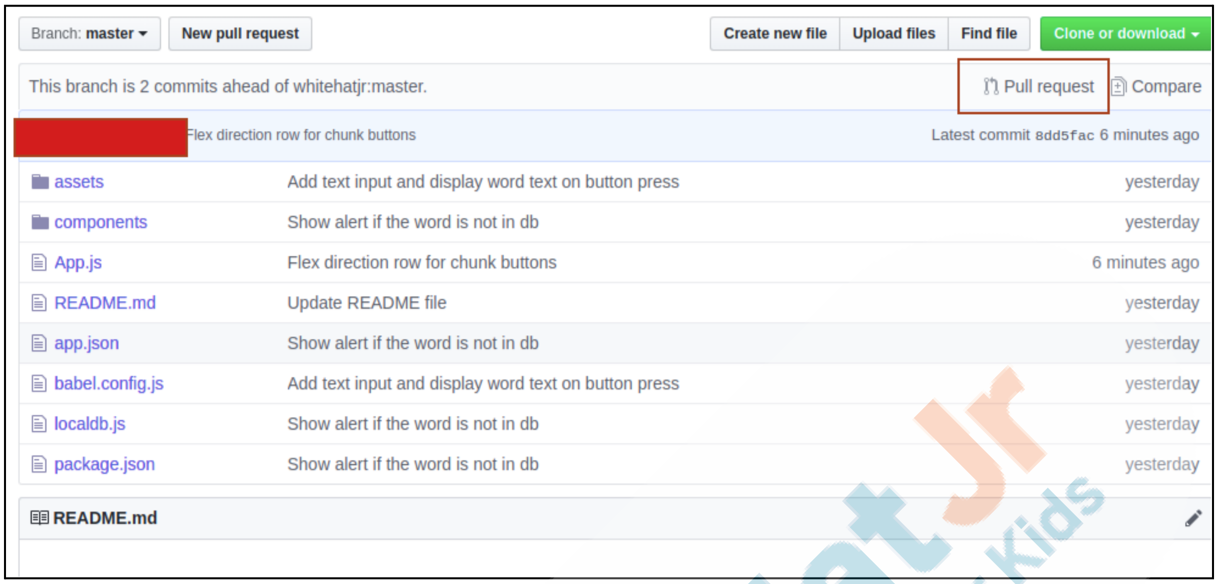
This branch is 2 commits ahead of whitehatjr:master. | Pull request | Compare

[redacted] Flex direction row for chunk buttons | Latest commit 8dd5fac 6 minutes ago

assets	Add text input and display word text on button press	yesterday
components	Show alert if the word is not in db	yesterday
App.js	Flex direction row for chunk buttons	6 minutes ago
README.md	Update README file	yesterday
app.json	Show alert if the word is not in db	yesterday
babel.config.js	Add text input and display word text on button press	yesterday
localdb.js	Show alert if the word is not in db	yesterday
package.json	Show alert if the word is not in db	yesterday

README.md

	Now your forked repo is different from the original repo. You can make a pull request on the original repo. This will send a request to the original developer of the project to pull the changes you have made into their current project. You will have to write a message to tell the developer what change you made. The developer can accept the pull request and merge the changes into their current project. The pull request will remain open till it is merged with the original project or closed by the developer. Once merged, you will then be added as a contributor to the project. Guide the student to make a pull request.	The student makes a pull request.



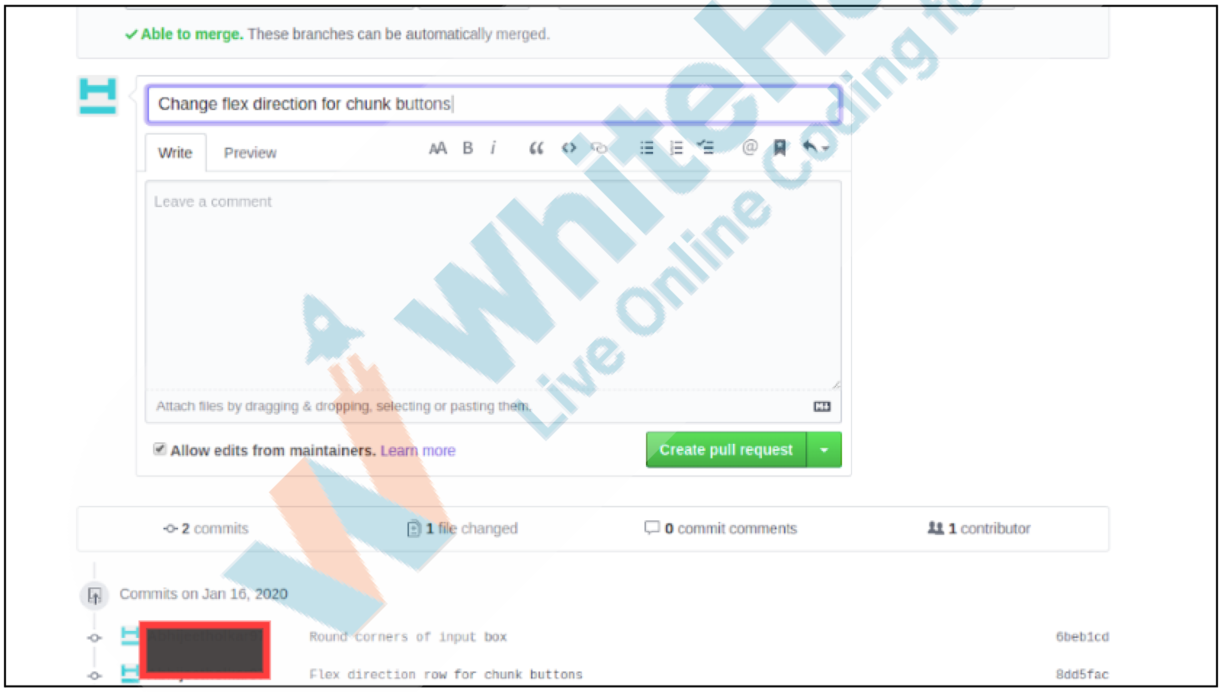
Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

This branch is 2 commits ahead of whitehatjr:master. | Pull request | Compare

[redacted] Flex direction row for chunk buttons — Latest commit 8dd5fac 6 minutes ago

assets	Add text input and display word text on button press	yesterday
components	Show alert if the word is not in db	yesterday
App.js	Flex direction row for chunk buttons	6 minutes ago
README.md	Update README file	yesterday
app.json	Show alert if the word is not in db	yesterday
babel.config.js	Add text input and display word text on button press	yesterday
localdb.js	Show alert if the word is not in db	yesterday
package.json	Show alert if the word is not in db	yesterday

README.md



✓ Able to merge. These branches can be automatically merged.

Change flex direction for chunk buttons

Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Allow edits from maintainers. Learn more
Create pull request

2 commits | 1 file changed | 0 commit comments | 1 contributor

Commits on Jan 16, 2020

[redacted] Round corners of input box 6beb1cd

Flex direction row for chunk buttons 8dd5fac

	Fantastic! You have learned how to make open source contributions!	

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 Mins

Teacher starts slideshow from slide 14 to slide 25	
Activity details	**Solution/Guidelines**
Run the presentation from slide 14 to slide 25 **Following are the WRAP-UP session deliverables:** **Following are the wrap-up session deliverables:** ● **Explain the facts and trivias** ● **Next class challenge** ● **Project for the day** ● **Additional Activity**	Discuss with the student the current class activities and Student will ask doubts related to the activities.
Quiz time - Click on in-class quiz	
Question	**Answer**
Which git command is used to initialize the repository? A. git init B. git add C. git commit D. git push	**A**
When it comes to the git stages what is a working directory? A. it is the stage where the files are ready to be committed, we have to stage them first B. it is the stage where we are creating and making edits to our files C. it is the stage after the files are committed, they become a part of commit history in a local repository D. it is the stage where the local repository is created	**B**
Which of the following commands is used to see all your commits and also see the time of the commit, the author and the commit message?	**D**

A. git commit B. git checkout C. git push D. git log		
End the quiz panel		
FEEDBACK ● **Get students to explore different open source projects in React Native.**		
	You get a "hats off". Till next class then. See you. Bye!	Make sure you have given at least 2 Hats Off during the class for: Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
	In the next class, we will be starting on a new project and a new problem statement.	
Project Pointers and Cues (5 min)	CONTRIBUTE TO TEAM VOTING APP **Goal of the Project:** In Class 67 you learned about advanced features of git versioning tools - staging, branching, merging, comparing, resetting and how to contribute to open source projects on github using git.	**Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.**

	In this project you have to contribute to a project app called “Team Voting App” using advanced git commands. **Story:** Every Saturday, your teacher organizes a fun game activity in which she divides students into two teams in your class. When it comes to voting for the winning teams, it becomes very complex for the teacher to count votes. Your teacher knows that you are learning how to code. She has asked you to create a Team Voting App and help her in resolving her issue. You need to explore the existing Team Voting app and make meaningful contributions to the functioning of this app. Are you ready? I am very excited to see your project solution and I know you both will do really well. Bye Bye!	
Teacher ends slideshow		
Teacher Clicks ✖ End Class		
Additional Activities	Encourage the student to write reflection notes in their reflection journal using markdown.	The student uses the markdown editor to write

	Use these as guiding questions: ● What happened today? ○ Describe what happened ○ Code I wrote ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? ● What did I find difficult?	her/his reflection as a reflection journal.

Activity	Activity Name	Links
Teacher Activtiy 1	Git installation guide	https://s3-whjr-curriculum-uploads.whjr.online/22782b72-ed36-4497-8a8f-2689290acb78.pdf
Student Activity 1	Visual Git Tutorial	https://onlywei.github.io/explain-git-with-d3/
Student Activity 2	Monkey Chunky Open Repo	https://github.com/whitehatjr/Monkey-Chunky
Project Solution Link	CONTRIBUTE TO TEAM VOTING APP	Solution depends on students' submission.
Teacher Reference visual aid link	Visual aid link	https://s3-whjr-curriculum-uploads.whjr.online/9a0052d6-b684-4b8b-be21-004515ef6641.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/54182c5b-ed48-4a71-918

		2-125c9885422c.pdf