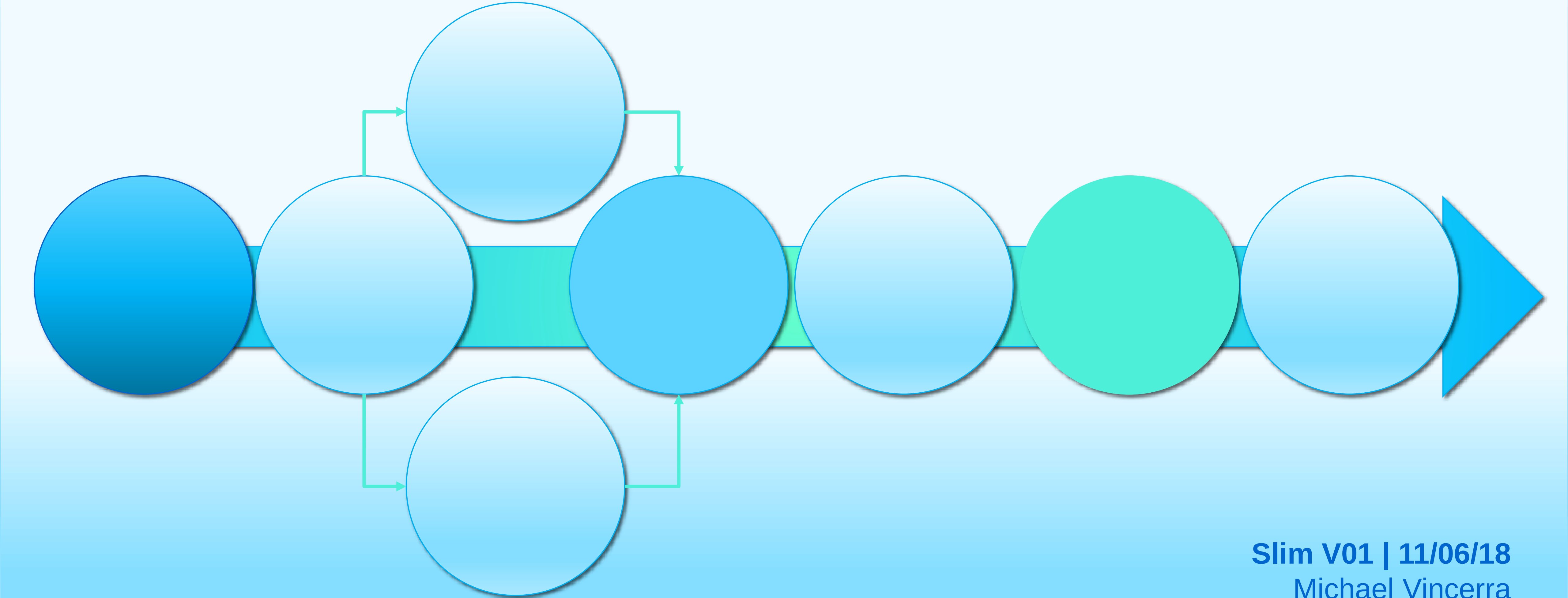
Git Workflows

Click this link to see instructions on Practice Repo .



Slim V01 | 11/06/18

Michael Vincerra

Cloud Software Engineer, Intel OTC

overview

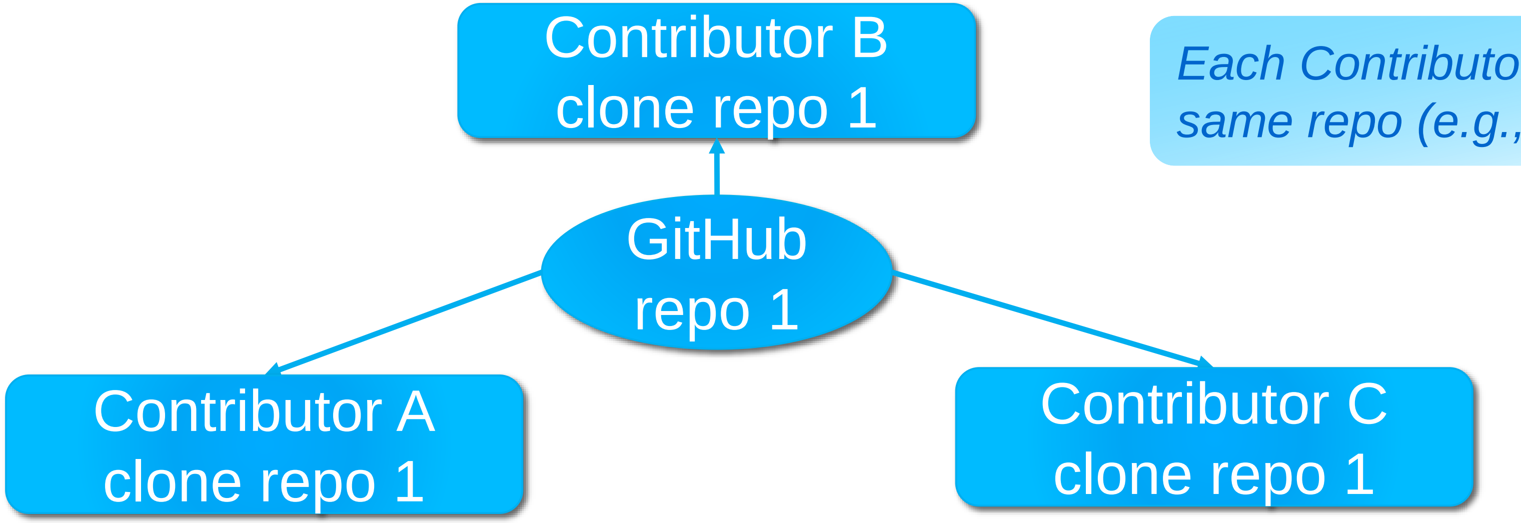
How to use a CLI and GitHub to contribute to Clear Linux* documentation:
https://github.com/clearlinux/clear-linux-documentation

table of contents

set up: fork, clone, & add upstream
synchronize
option 1: create a branch
option 2: add a remote user & create a branch
prepare for PR: add, commit, push
create a pull request (PR)
manage merge conflicts
commands, terms, roles

what is GitHub?

“**GitHub** Flow is a lightweight, branch-based workflow that supports teams and projects where deployments are made regularly[...]” https://guides.github.com/introduction/flow/



Each Contributor clones the same repo (e.g., repo 1)

- You create a local copy of the repo on your computer when you **clone**
- Your local copy is independent of the **upstream** repo
- Your local copy can have many **remotes**

why branch?

“When you're working on a project, you're going to have a bunch of different features or ideas in progress at any given time – some of which are ready to go, and others which are not. ***Branching*** *exists to help you manage this workflow. […]*

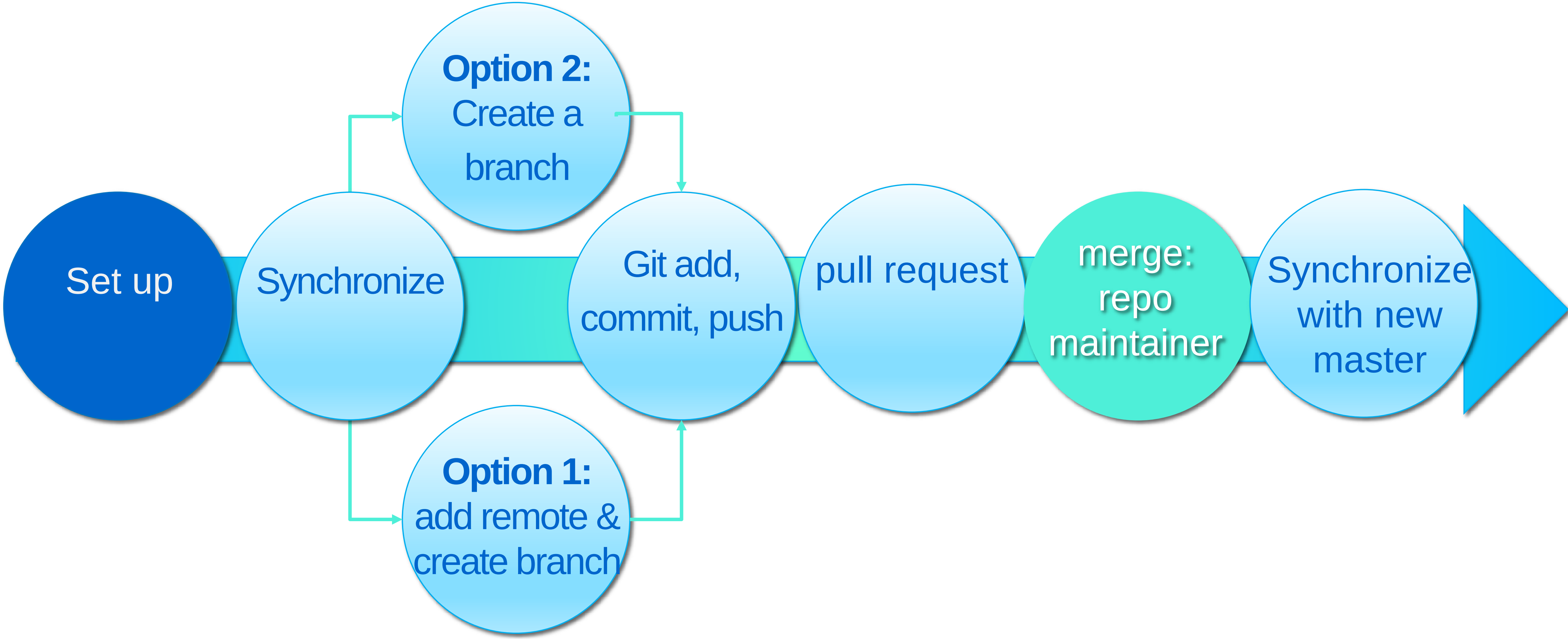
“Branching is a core concept in **Git.** [...] ‘**master**’ branch is always deployable. Because of this**, it's extremely important that your new branch is created off of master when working on a feature or a fix […]”**

Always create a branch and make your edits on it. Never work directly on master.

Source: https://guides.github.com/introduction/flow/

set up:
fork, clone, & add upstream

Note: This training assumes that you already have a GitHub account. To sign up, visit: http://www.github.com


Set up
Synchronize
Option 2: Create a branch
Option 1: add remote & create branch
Git add, commit, push
pull request
merge: repo maintainer
Synchronize with new master

LEGEND

● = You are here

configure SSH file & Create SSH key on GitHub

OTC recommends using a SSH public key for GitHub for security reasons.

Follow the GitHub instructions under “Connecting to GitHub with SSH” found here:

https://help.github.com/articles/connecting-to-github-with-ssh/

NOTE: This tutorial assumes that you’re using SSH on a Linux environment.

Complete these sections at minimum:

- Checking for existing SSH keys
- Generating a new SSH key and adding it to the ssh-agent
- Adding a new SSH key to your GitHub account

Note: You do not need to use a passphrase when setting up SSH.

For Windows OS, we recommend that you install and use Cygwin.

clone documentation repo; add remote upstream

1. In the CLI, navigate to your home directory.
2. On GitHub, go to: https://github.com/clearlinux/clear-linux-documentation
3. **Select the Fork** button in the far upper right.
4. Now in *your own GitHub account*, navigate to YOUR FORKED repo.
5. Press **Clone or download**.
6. In the dialogue box, select “Use SSH” . Then copy the text in the dialogue box.
7. In CLI, enter *git clone* + the copied text, and assure [yourusername] appears.

```
git clone git@github.com:[yourusername]/clear-linux-documentation.git
```

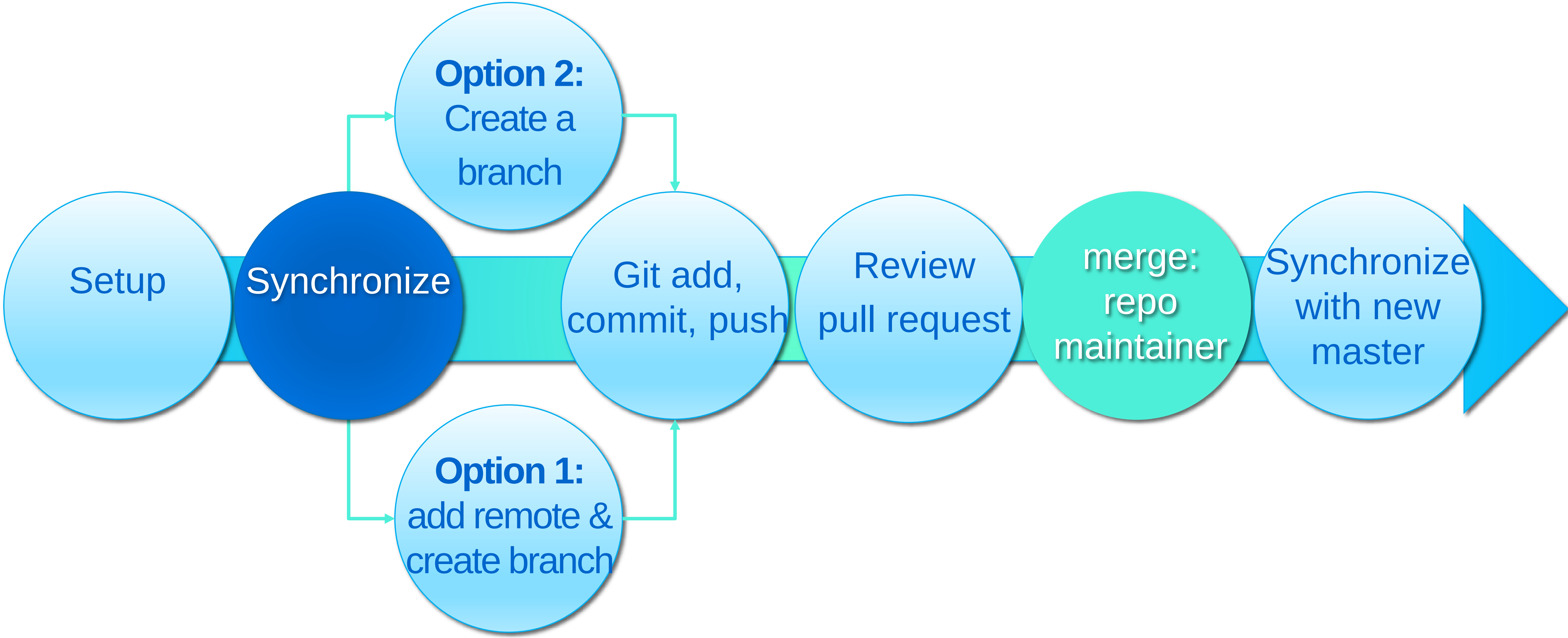
8. Return to the CLI and change directory into the forked repo:

```
$ cd clear-linux-documentation
```

9. Repeat Step 2. Repeat Step 5 and copy upstream repo’s URL. Enter command:

```
git remote add upstream git@github.com:clearlinux/clear-linux-documentation.git
```

synchronize



LEGEND

● = You are here

synchronize your fork to upstream repo

1. On the CLI, enter:

```
git checkout master
```

2. Next, enter:

```
git pull --rebase upstream master
```

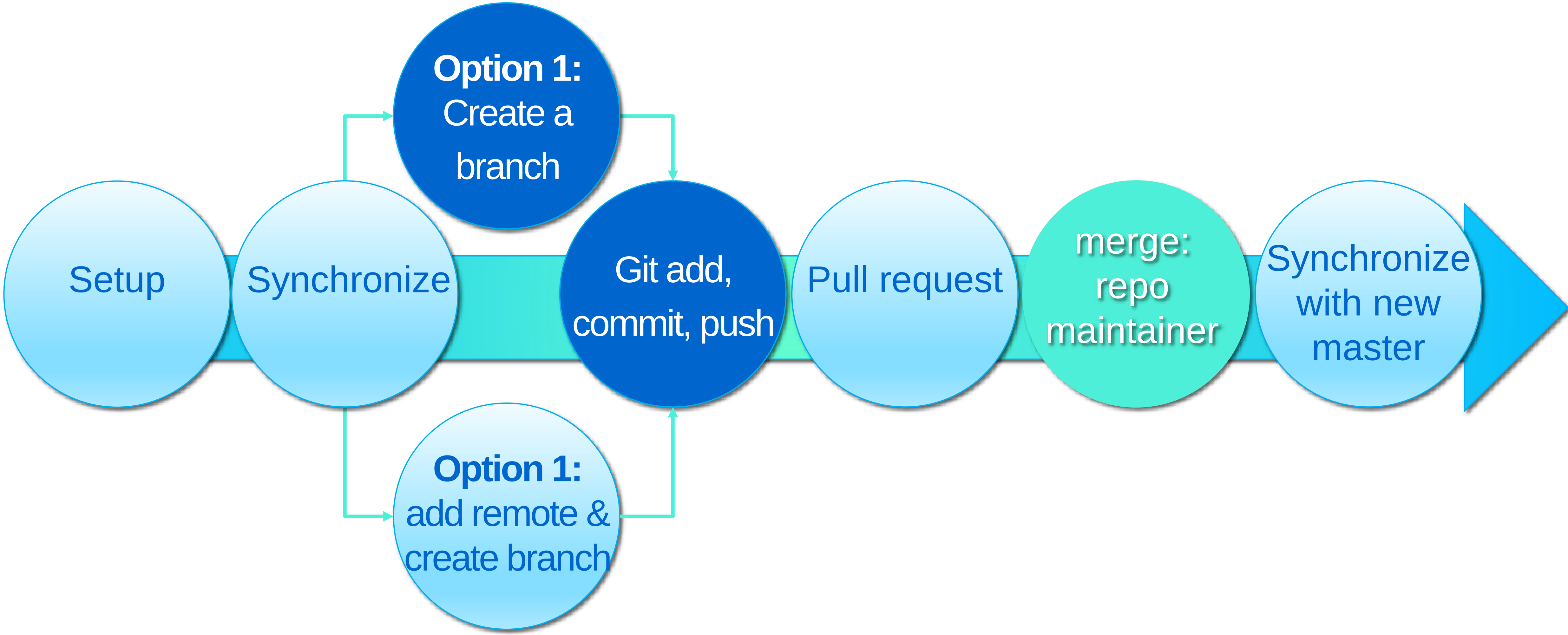
NOTE: We recommend using ‘--rebase’ in this command for a cleaner commit history.

3. Push the downloaded changes to your local **origin**—to assure that your fork is up to date.

```
git push origin master
```

Congratulations! Your forked repo is now up to date. Make a habit of doing this regularly—every time that you start to work. Also, do this before you start a new branch.

option 1: create a branch



LEGEND

● = You are here

create a branch: add, commit, and push.

1. Assure you complete steps in Synchronize before proceeding.
2. While on the master branch in the CLI, create a new branch. Enter:

```
git checkout -b [xx]-[filename]
```

NOTE: Name branch with your initials [xx], hyphen [filename] (no spaces).

3. While on new branch, make edits to the document in the Editor (e.g., Sublime).
4. In the CLI, enter command to **add** the revised [filename] and commit your changes.

```
git add [relative/path/filename]**
```

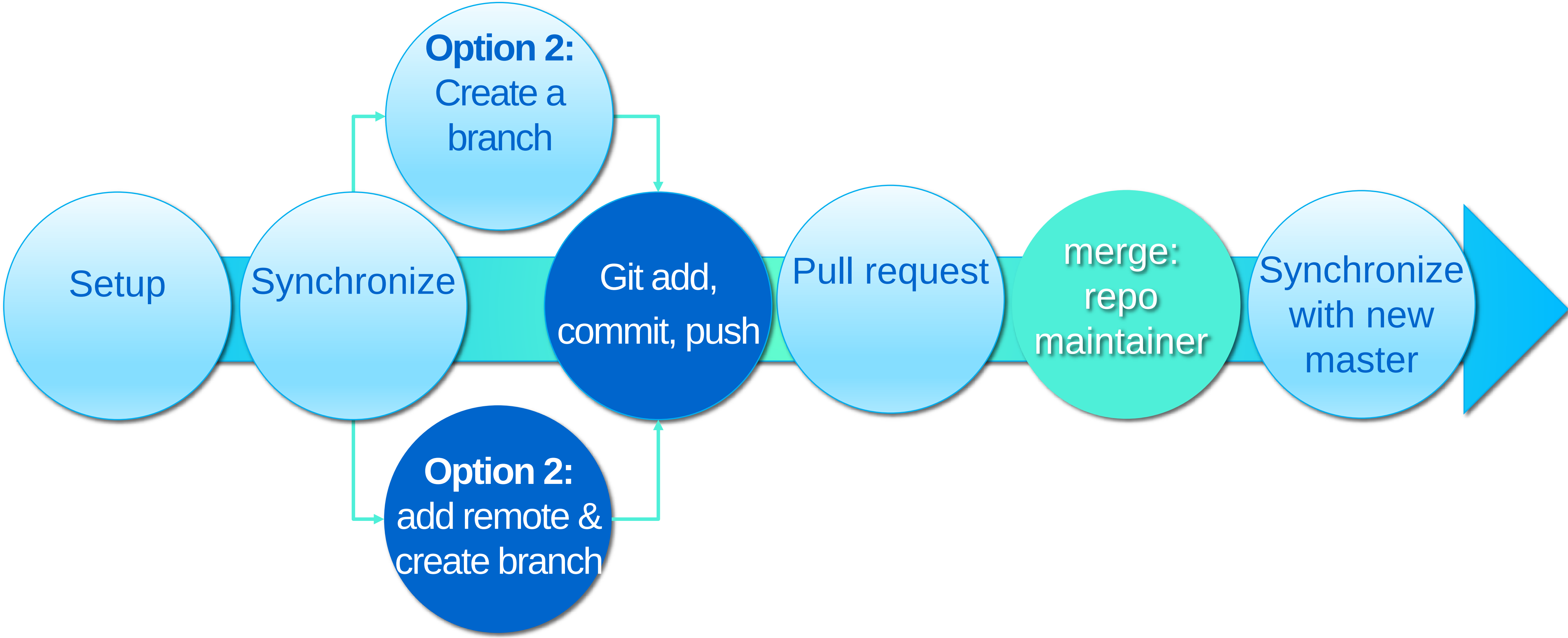
5. Enter command. In the editor write a descriptive commit message. Save and exit.

```
git commit –s
```

6. Push your new branch. Then complete a Pull Request (see “create a pull request”)

```
git push origin [xx]-[filename]
```

option 2: add a remote user & create a branch



LEGEND

● = You are here

add a user’s forked repo as a remote

1. To collaborate with another user/contributor, add a user’s fork as a **remote**:

```
git remote add [username] [**https:github.com/username/reponame.git]
```

*NOTE: ** Visit contributor’s GitHub repo and copy URL or press `Clone or Download`.*

2. Fetch their forked copy. Enter:

```
git fetch [username]
```

3. Checkout and create a new branch:

```
git checkout -b userbranch username/userbranch
```

Note: 1) You must request a contributor to assign you permission to push to his/her branch.
*2) OTC’s convention is to use the **same name for your branch** as that of the user.*
You create a new branch that you later push to the contributor/user’s fork.

prepare for PR: add, commit, push

add, commit, and push

While on a new branch, make edits to the documentation and save. Follow these steps.

1. In the CLI, enter `git status` to verify file edited. Next enter:

```
git status
```

```
git add [filename]
```

Note: [filename] is the file that you just edited.

2. Next, enter commit message:

```
git commit -s
```

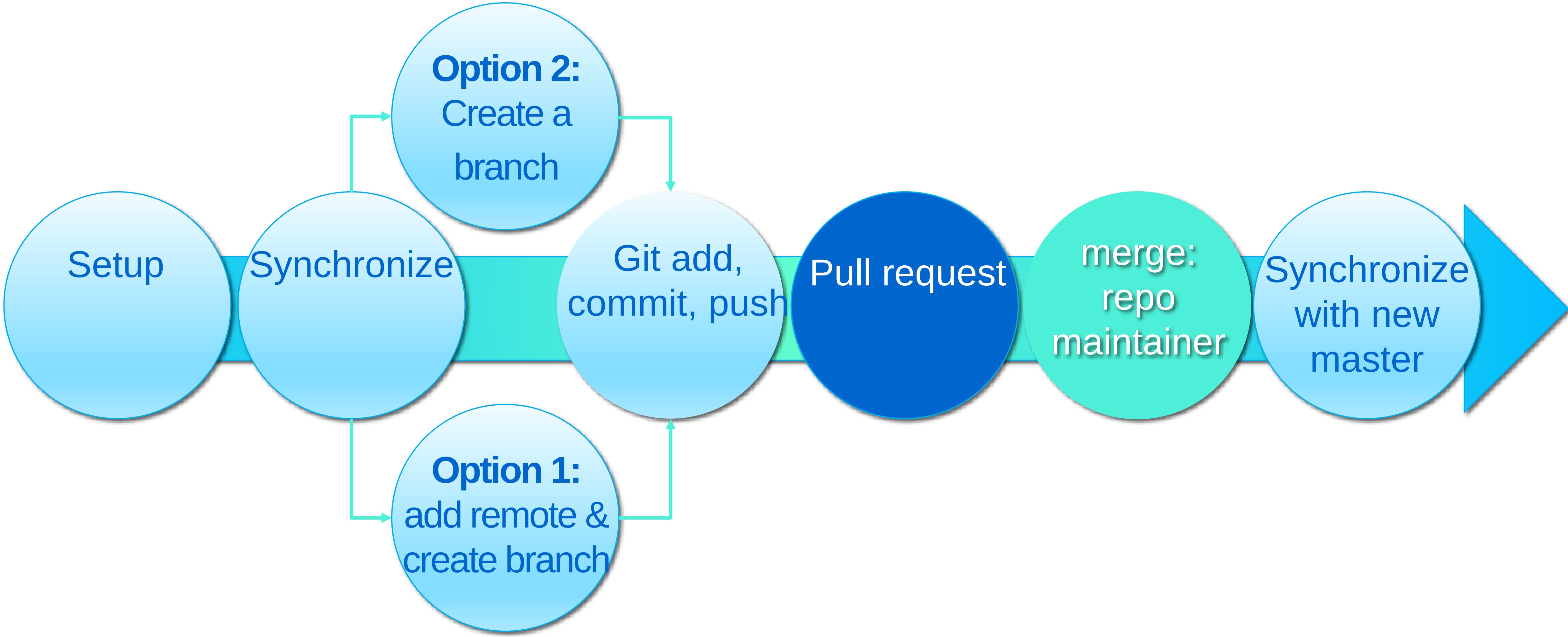
Note: Using "-s" allows a 'sign-off' on a commit; it automatically opens your editor. Write a detailed message addressing the why and what of your proposed change.

3. Enter:

```
git push [username] userbranch
```

4. Finally, follow the instructions in "Create a pull request."

create a pull request (PR)



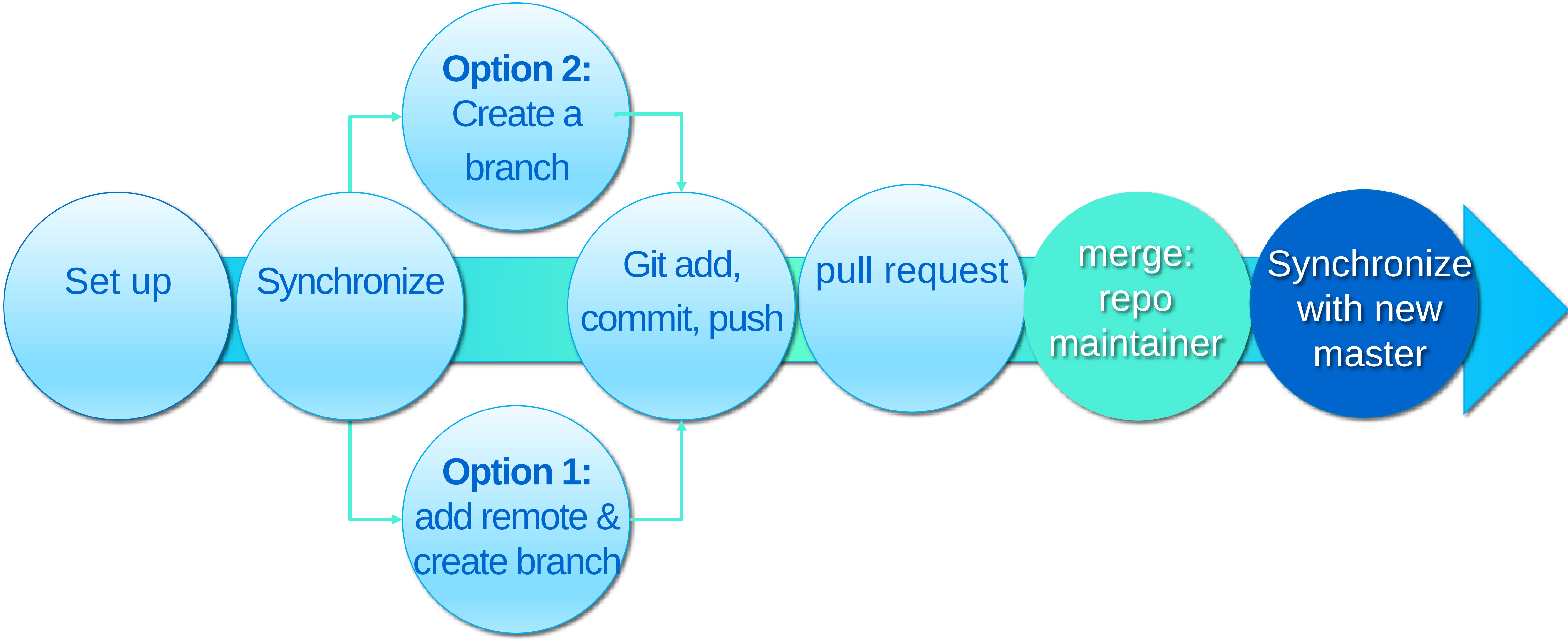
LEGEND

● = You are here

create a pull request

1. Visit https://github.com/clearlinux/clear-linux-documentation, or the ‘upstream’ CL repo.
2. “*Your recently pushed branches* :” will appear as the last branch you pushed.
3. Select the button “Compare & pull request.”
4. “Open a pull request” appears.
5. Find the **gear icon** beside Reviewers: Select appropriate reviewers (Repo Maintainer).
   Note: If you added a remote user and created a new branch, add that user as a Reviewer.
6. Review the pull down menus below “Open a pull request.”
7. Assure that *compare* *(right)* shows your branch and *base fork* (left) shows upstream.
8. Select the button “Create pull request.”
9. You will receive email notifications on the status of your pull request.
10. The Repo Maintainer will merge your pull request.

Note: Your pull request will be either merged or others may request additional changes. For the latter, you can make more edits. Then follow the instructions to “add, commit, and push” in “option 2: create a branch”) Continue pushing to your same branch if it has not been merged.



Set up

Synchronize

Option 2: Create a branch

Option 1: add remote & create branch

Git add, commit, push

pull request

merge: repo maintainer

Synchronize with new master

LEGEND

● = You are here

synchronize with upstream master

After your pull request is merged, you need to synchronize your local fork with upstream ‘master.’

In short, you must assure that your local ‘master’ reflects the same status as the upstream ‘master’—before you start any new branches. Return to the section Synchronize.

Congratulations! You’ve come full circle.

manage merge conflicts

manage merge conflicts

Branches become outdated. When a GitHub branch shows that “this branch has conflicts…”, you need to merge master while on your own branch (below) to incorporate any previous merges made to upstream master that may conflict with your branch. Run the command below as a separate commit and push it to your branch. If you're unsure, ask the repo maintainer.

1. Check out your branch. Enter:

   ```
   git checkout [branchname]
   ```

2. Your branch needs to pull in any other changes merged with upstream master. Since the time you checked out your branch, others have pushed and had PRs merged that may conflict with your changes. Note: This action results in a "merge commit" on your branch.

   ```
   git merge master
   ```

3. Return to “prepare for PR” and follow instructions.

Practice Repo

For the purpose of this training, you will use files found at this repo in order to practice : https://github.com/mvincerx/git-workflows-training

Read the README.md at this URL for instructions on training.

The Practice Repo is intended to give you a place where you can learn Git Workflows. The Repo is provided as a safe environment where you're free to attempt git commands, and occasionally make mistakes, without any major consequences.

The instructor will guide you through using this repo during training, instead of using the official Clear Linux* documentation.

commands, terms, roles

commands

git command	purpose
git **checkout** master	Assures that your CLI is pointed to your local master branch.
git **pull** --**rebase** upstream master	Downloads all changes from (shared) upstream master.
git **push** origin master*	Pushes the above downloaded changes to your local master. *Optional yet recommended.
git **status**	Shows the status of the working tree

commands

git command	purpose
git **clone** […]	Creates a copy of the master branch on your localhost; creates origin as a **remote**
git push **origin** master*	Synchronizes your local copy with origin on the master branch. **origin** points to YOUR FORK of upstream. *Command is optional yet recommended.
git **remote add upstream** […]	Creates a remote on your local machine that points to upstream fork (the original).
*NOTE: You receive ‘**origin**’ for free when you ‘**git clone**’.*	

commands

git command	purpose
git **checkout -b** [branchname]	Creates and switches to a new branch (-b). Option: git **branch** [branchname] also works
git **checkout** [branchname]	Switches to a new branch
git **add [filename]**	Prepares and stages files to be committed and pushed
git **commit** -s [-m][“message”]	Commits staged files and: -s: Opens text editor to write commit message -m “Write commit message here”
git **push** origin [branchname]	Pushes your branch to origin

terms

add: Stages a file to be committed and pushed. If necessary, you can add multiple files to a commit by repeating `git add [filename]`. In general, OTC prefers one file per branch per PR. However, there are always exceptions. For example, if you create a tutorial with multiple images, you'll need to add all of them in one PR. In general, always run 'git status' to view the files that you have changed.

commit: Write a short concise message that describes the nature of revisions or the purpose of adding a new file. Note: The commit message becomes the title of the PR, so be sure to accurately describe your proposed changes.

push: git push origin [branchname]. This command literally pushes your branch to your local remote called origin. Your push will appear in the main repo as "*Your recently pushed branches.*" See "Create a pull request."

terms

CLI: Command line interface. This tutorial uses BASH, or Bourne-again Shell, and thus, BASH commands. Several other CLIs exist.

clone: Your own copy of the upstream repository, or repo. Your copy is independent of the upstream; you must update regularly your local copy to upstream ‘master’.

local: First, your local copy of the repo can be referred to as ‘local’ or sometimes as your ‘fork.’ When you create branches, you create a local branch that only you can see at first; when you push your branch to origin and create a PR, others see your branch.

remote: A remote branch is a branch on a remote location (see also ‘origin’). You push your local branches to ‘origin.’ When you use ‘git clone’ on a repo, ‘origin’ is automatically added as a ‘remote.’ You can add others’ branches as remotes.

terms

master: The local ‘master’ branch on your local forked copy of the repo. This is independent of the upstream ‘master’ (see below).

upstream: The official version of upstream ‘master’ which is regularly deployed. Contributors create branches that are proposed to be merged with upstream ‘master.’ The process by which these branches are merged with upstream ‘master’ requires a pull request (PR).

roles: who does what?

Repo Maintainer: A repo maintainer manages Pull Requests (PR) , reviews and approves them, and merges them. A repo maintainer may manage PRs for several branches from several contributors.

Contributor: A contributor can be anyone who starts a new document or creates a new branch. This may be a TME, Developer, a Technical Writer, or an external contributor.

Reviewer: A reviewer is internal to the Clear Linux* documentation team. A reviewer may be a Technical Writer, or anyone else who reviews another person's PR on Clear Linux documentation. *Note: Don't confuse 'reviewer' as used here with the "Reviewers" on the PR.*