How to Set up a Git Repo for Your Team

Learning, understanding, and consistently implementing good DevOps practices saves teams time, reduces defects, and increases overall team efficiency.

In this kata/exercise, each Apprentice is expected to complete the setup of a new team repo for a team development project, and have their work verified by an A100 Team Member. Immediately afterwards, one member of each Apprentice team will set up an official repo for the team to actually use on their Venture project and will set the remote to a private repo in the Apprentice100 organization on Github.

<u>Recommended Badge:</u> *You'll earn your DevOps I Badge (required for baseline MVS score at Week 4) for completion of this exercise individually. Prerequisites for this badge include git I and git II training and confirmed completion of related exercises.*

Prerequisites:

- Knowledge of command line and access to the tool on your machine
- An internet connection
- Access to Meteor.com documentation
- Access to Git documentation (at git-scm.com)

Each team member must also set up their own local Dev environment including:

- Install Trello & Slack (for communication)
- git (for SCM)
- GitHub (for authorized access to team repo)
- Sublime Text (as your IDE)
- Chrome (for Chrome developer tools)

Quick Outline: Every Apprentice will complete Steps 0-2. After this is done one person from each team, whoever is designated ScrumMaster, will complete step 3. When step 3 is complete every Apprentice will complete steps 4-5.

0: Install Meteor locally, add appropriate packages, set up the project structure, and put it under version control locally using git.
1: Create Remote repo and creates team, inviting other team members to write.
2: Link your local repo to the remote and push.
3: Steps 1 & 2, Revisited, for your official Team Repo. (ScrumMaster only)
4: Clone the official Team Repo down.
5: Use this in your development, using a dev-master-production workflow.

Step Zero.

A designated member of your team (this should be one of the first Tasks created on your team's Taskboard) creates a Meteor project in their own local machine using the latest version of Meteor, named for your project (e.g., project-xavier or julio-and-friends). The installation should include:

- Bootstrap version 3.8.0
- jQuery version 1.9.2
- Iron Router version 1.0.7
- accounts-UI 1.2.0
- accounts-password 1.1.1
- email 1.0.6

If you need documentation on any package, check out the Atmosphere site: https://atmospherejs.com/

Drill-down:

To download Meteor, visit meteor.com. For OSX and Linux users, use the following command in Terminal:

```
$ curl https://install.meteor.com/ | sh
```

To check your Meteor version (you should have installed v. 1.1.0.2), use the following command in Terminal:

```
$ meteor --version
```

To create the new project, navigate to your working directory (e.g., `$ cd Desktop/project-workspace`), and run the command:

```
$ meteor create [your project name]
```

Next, go into the new project directory:

```
$ cd [your project name]
```

Type this to add the package "Iron Router" for routing:

```
$ meteor add iron:router
```

Add the following packages (Bootstrap, JQuery, Accounts-UI, Accounts-Password, and Email):

```
$ meteor add twbs:bootstrap
$ meteor add jquery
$ meteor add accounts-ui
$ meteor add accounts-password
$ meteor add email
```

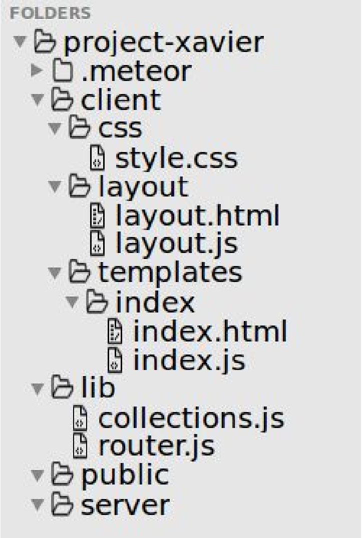
To check your package versions by using the following command *inside your new Meteor project directory* in Terminal:

```
$ meteor list
```

or look in the `.meteor/versions` folder.

Set up basic directory structure:

- delete .css, .js, .html files in root folder
- set up client, server, lib, public folders, and create empty files that go inside to exactly mirror the folder structure shown in this image (next page).



```
FOLDERS
▼ project-xavier
  ▶ .meteor
  ▼ client
    ▼ css
        style.css
    ▼ layout
        layout.html
        layout.js
    ▼ templates
      ▼ index
          index.html
          index.js
  ▼ lib
      collections.js
      router.js
  ▼ public
  ▼ server
```

We'll explore what all these files mean later, but for now, we're focused on setting up the right dev environment, not on Meteor application development. Once you've got all the files above, and your project directory contains everything the project-xavier directory contains, you're done with the first chunk, and ready to make your "Initial commit." (`dev` and `production` branches must also be created, following our normal protocol).

Drill-down:

To put your project under SCM using git, type the following command in the project directory after meteor is installed:

```
$ git init
```

```
$ git status
```

// confirms that your project exists in the directory, but has not been added or committed.

To commit your baseline Meteor code, first add all the files, then commit them:

```
$ git add -A
$ git commit -m "Initial commit of Meteor platform with modified
directory structure."
```

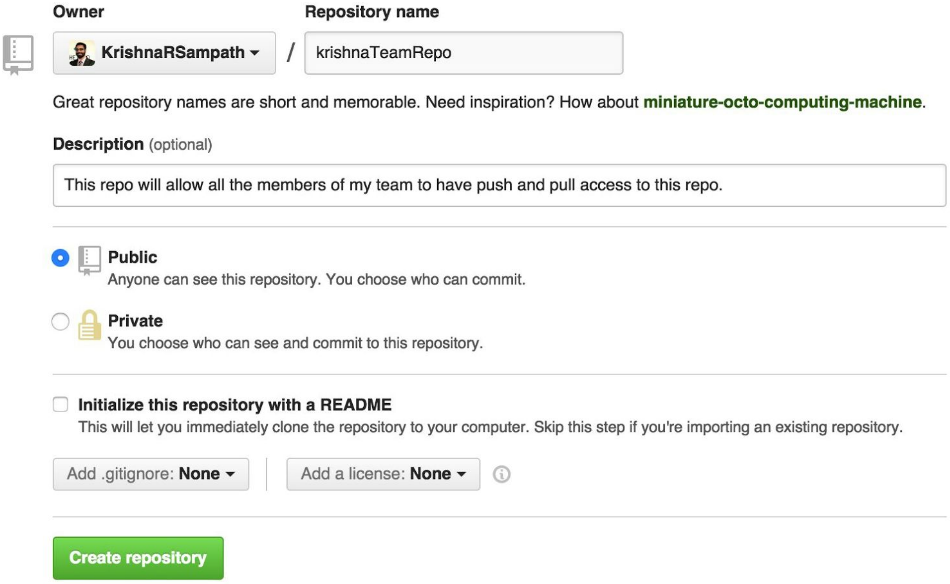
To create `dev` and `production` branches from the baseline so your team can begin work:

```
$ git branch production
$ git branch dev
$ git checkout dev
```

(Remember, your code for project work MUST not be shared publicly, and so cannot be shared on any public Github account. You'll use a PRIVATE Github repo in the A100 Github organization to serve as a central repo for backup, and to transfer to your Product Owner upon approval. This is why we have you create a `production` branch and not a `gh-pages` branch.)

Step One.

Create a new public repo in Github, named in this format: `firstnameTeamRepo`



Owner: KrishnaRSampath / Repository name: krishnaTeamRepo

Great repository names are short and memorable. Need inspiration? How about **miniature-octo-computing-machine**.

Description (optional)

This repo will allow all the members of my team to have push and pull access to this repo.

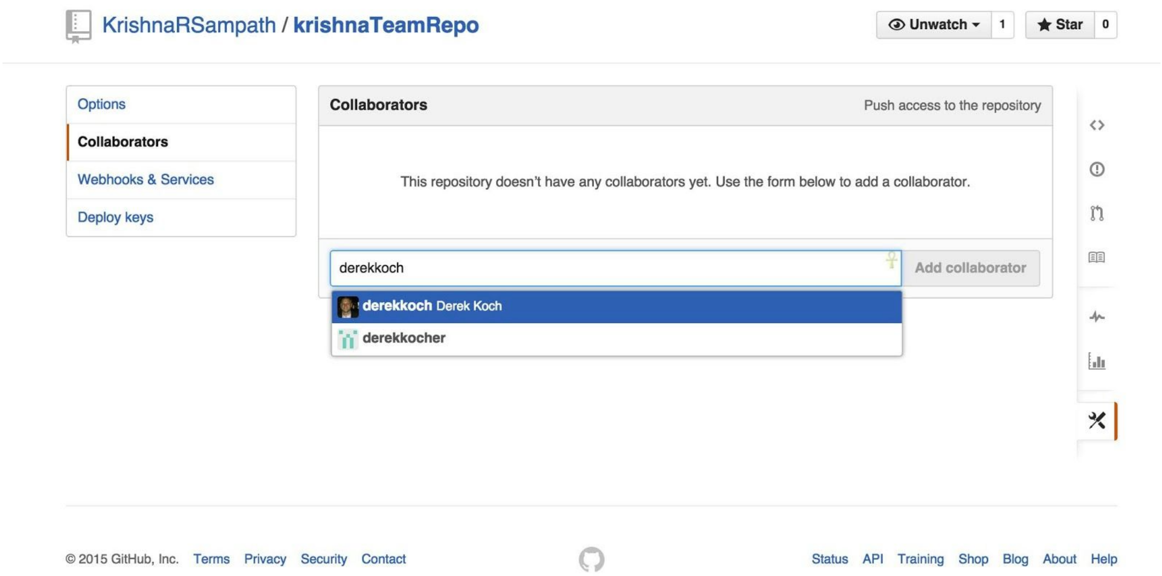
● **Public**
Anyone can see this repository. You choose who can commit.

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

□ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

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

Don't initialize this repo with a README, as you'll be linking your local repo up to it. Once you create it, go to the repo's Settings page (on the right side), and add your teammates as Collaborators (you'll need to enter your password to get to this screen).


KrishnaRSampath / krishnaTeamRepo

Collaborators

Push access to the repository

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

derekkoch

derekkoch Derek Koch

derekkocher

Here, Krishna adds Derek as a collaborator, since they'll be working together on a team project.

Step Two.

Next, push your local repo (all branches) up to your new Github repo. Use the following command to set up the Github project as the remote repo:

```
$ git remote set-url origin https://github.com/[username]/[project-name].git
```

// confirm that you correctly set up your remote repo.

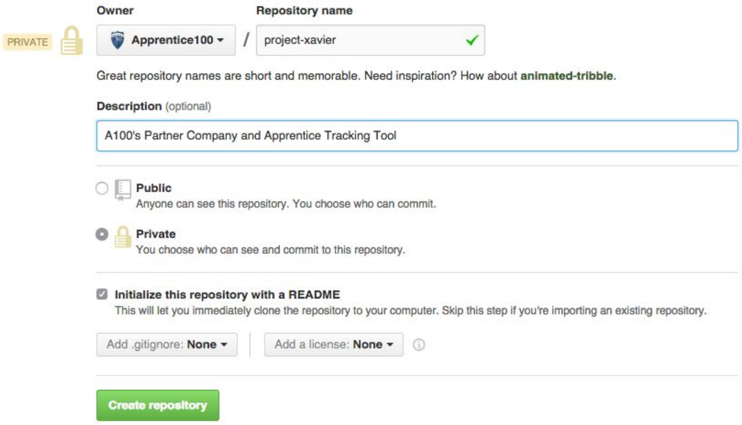
```
$ git remote -v
```

// pushes all three branches up to your remote repo (`git push origin [branch]` only pushes the specified branch).

```
$ git push -u origin --all
```

Step Three.

Your ScrumMaster then requests that the Site Manager or a Senior Apprentice create a new repo for them in A100's Github organization and create a team with just that team member included with Admin privileges (in the repository's Settings) to control access to this version of the repo.



PRIVATE

Owner: Apprentice100 / Repository name: project-xavier ✓

Great repository names are short and memorable. Need inspiration? How about **animated-tribble**.

Description (optional)

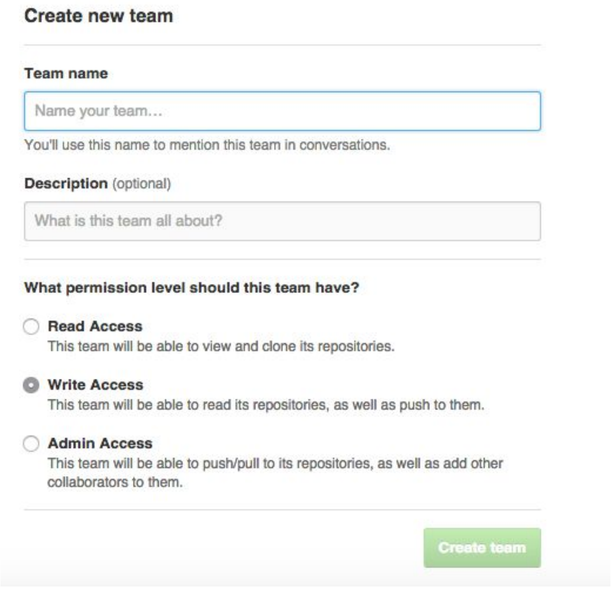
A100's Partner Company and Apprentice Tracking Tool

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

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

☑ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

New team members should be added with Write access by the designated team member (see next page for a picture).



Create new team

Team name

Name your team...

You'll use this name to mention this team in conversations.

Description (optional)

What is this team all about?

What permission level should this team have?

○ **Read Access**
This team will be able to view and clone its repositories.

● **Write Access**
This team will be able to read its repositories, as well as push to them.

○ **Admin Access**
This team will be able to push/pull to its repositories, as well as add other collaborators to them.

Then, the designated team member's local repo (all branches) is pushed to the Github repo as the start point. Use the following command to set the private repo on Github as the new remote repo:

```
$ git remote set-url origin https://github.com/Apprentice100/[team-name].git
```

// confirm that you correctly set up your remote repo.

```
$ git remote -v
```

```
$ git push -u origin --all
```

Step Four.

Each team member clones the baseline repo to their local machine in their A100 workspace (`a100-dev`) folder, which creates a new project folder (e.g., `a100-dev/project-xavier`).

Use the following command:

```
$ git clone https://github.com/Apprentice100/[project-name].git
```

Step Five.

Each team member will begin to make updates according to the Taskboard and team discussions. All code should follow the A100 coding guidelines, and team members should follow these practices:

- Locally commit work against logical units of work (from the Taskboard) after each coding session. Use the A100 Coding Guidelines to ensure your code passes code review.
- Push successfully tested units to the `dev` branch after code review by another teammate.
- Move versions to `master` for release (and before each weekly demo).
- A designated Senior Apprentice & the Site Manager will take turns reviewing your code and noting issues (code review events should be initiated as Tasks on the Taskboard).