Initial setup

Before you perform any of the instructions discussed in this page, make sure you've followed the steps described in Initial setup (common.html#chisubmit-common).

Registering for an assignment

Before you can make a submission for an assignment, you will need to register for that assignment. Furthermore, in some classes, your git repository may not be created until you register for your krst assignment (on the other hands, some classes may pre-create individual repositories for each student).

To register for an assignment, your instructor will provide an *assignment identifier*, which we will refer to as `ASSIGNMENT_ID`. You can also see the list of upcoming assignments (and their identikers) by running this:

```
chisubmit student assignment list
```

If you are registering individually for an assignment, you just need to run this:

```
chisubmit student assignment register ASSIGNMENT_ID
```

If you are registering as a team, you will need to use the `--partner` option. For example, if you are `studentA` and your project partner is `studentB`, you (`studentA`) must krst run this command:

```
chisubmit student assignment register ASSIGNMENT_ID --partner studentB
```

Your partner (`studentB`) needs to conkrm that they want to be in that same team. So, they (not you) must run this command:

```
chisubmit student assignment register ASSIGNMENT_ID --partner studentA
```

If you are registering a team with three or more students, then you just need to repeat the `--partner` option for each partner. So, assuming we have three students (`studentA` , `studentB` , and `studentC`), each student would run the following commands, respectively:

```
chisubmit student assignment register ASSIGNMENT_ID --partner studentB --partner studentC

chisubmit student assignment register ASSIGNMENT_ID --partner studentA --partner studentC

chisubmit student assignment register ASSIGNMENT_ID --partner studentA --partner studentB
```

chisubmit will assign you a team name that is composed of your student identikers (e.g., in the two-student example, the team name would be `studentA-studentB`). You can see the list of teams you are in by running this command:

```
chisubmit student team list
```

Take into account that, when registering as a team, your registration will not be complete until **all** members of the team have run `chisubmit student assignment register` . If you have run the command, but are not sure whether all your partners have also registered, you can use this command to check the status of your team (and, in particular, whether your partner(s) have completed their part of the registration):

```
chisubmit student team show TEAMNAME
```

(where `TEAMNAME` should be replaced by your team name)

Initializing your repository

Once you have registered for your krst assignment, a git repository will be created for you (either on GitHub or on a GitLab server, depending on the setup of your course). Take into account that if you register for one assignment individually, an individual repository will be created for you. If you then register for a different assignment as part of a team, a *separate* team repository will be created for you (where all your team members will have access).

Please take into account that the repository creation is not instantaneous. There can be a lag of 10-30 minutes between completing your registration and having your repository created.

Note

Once your repository is created, you will be able to access it on GitHub or GitLab. However, before you can pull or push from/to GitHub or GitLab, you have to make sure you add your SSH key to Github or GitLab.

Instructions for adding your SSH key on GitHub can be found here (https://help.github.com/articles/generating-ssh-keys/).

Instructions for adding your SSH key on a GitLab server will depend on the setup of your server, but general instructions can be found here (https://about.gitlab.com/2014/03/04/add-ssh-key-screencast/).

If your course is using GitHub, you will receive an "invitation e-mail" from GitHub asking you to join a group. Make sure you accept this invitation; you will not be able to access your repository until you do.

Once your repository has been created, you can verify that chisubmit can access it by running this command:

```
chisubmit student team repo-check TEAM_NAME
```

Where `TEAM_NAME` is your team name. If you signed up for the assignment individually, this will just be your university username.

If your course is set up to use GitHub, you should see something like this:

```
Your repository exists and you have access to it.
Repository website: https://github.com/GIT_ORGANIZATION/COURSE_ID-TEAM_NAME
Repository URL: git@github.com:GIT_ORGANIZATION/COURSE_ID-TEAM_NAME.git
```

Where `GIT_ORGANIZATION` will be the GitHub Organization used by your course and `COURSE_ID` is the course identiker.

If your course is set up to use GitLab, you should see something like this:

```
Your repository exists and you have access to it.
Repository website: https://git-server.example.edu/COURSE_ID/TEAM_NAME
Repository URL: git@git-server.example.edu:COURSE_ID/TEAM_NAME.git
```

Where, instead of `git-server.example.edu`, you will see your course's GitLab server.

Note

If your GitLab server is `git-server.example.edu`, then it's likely that the URL to add your SSH key to GitLab will be `https://git-server.example.edu/profile/keys`. Unless your instructor tells you otherwise, the username and password for the GitLab server will likely be your university username and password.

In the following instructions, we will be using the `Repository URL` value, which we will refer to as `GIT_URL`.

IMPORTANT: If you have a team repository (not an individual repository) the repository only has to be initialized by one of the team members.

To initialize your repository, the krst thing you need to do is create an empty local repository. In an empty directory, run the following:

```
git init
git remote add -f origin GIT_URL
```

Where `GIT_URL` should be replaced with the `Repository URL` printed by `chisubmit student team repo-check`.

Next, create a `README` kle and enter the names of all the team members. Add, commit, and push this kle to your repository:

```
git add README
git commit -m "Added README"
git push -u origin master
```

Cloning your repository

If a repository has already been initialized as described above, and you want to create a clone elsewhere, just run the following:

```
git clone GIT_URL
```

Where `GIT_URL` should be replaced with the `Repository URL` printed by `chisubmit student team repo-check`.

Setting up an upstream repository

Some assignments involve starting from some initial seed code provided by the instructors. The preferred method of adding this seed code to your repository is by having the instructor upload the code to a separate repository (which we will refer to as the *upstream* repository), which you will then pull into your repository, making it easy to then pull any future changes that happen in the upstream repository.

Do not follow these instructions unless told to by your instructor. There are many other ways of supplying seed code, and your instructor may provide alternate instructions.

Method 1: A single upstream repository

If your course has a single upstream repository, it is typically enough to run the following:

```
git remote add -f upstream UPSTREAM_URL
```

Where `UPSTREAM_URL` is the URL of the upstream repository.

To pull changes from the upstream repository, just run this:

```
git pull upstream master
```

Method 2: Multiple upstream repositories

Some classes may use multiple upstream repositories (e.g., one per project). In this case, we will use Git *subtrees* to pull kles from the upstream repositories. Each upstream repository will have an `UPSTREAM_URL`, and a prekx supplied by your instructor, which we will refer to as `PREFIX`.

To connect your repository to one of the upstream repositories, you need to run the following:

```
git remote add -f PREFIX-upstream UPSTREAM_URL
git subtree add --prefix PREFIX PREFIX-upstream master --squash
```

The code from this upstream repository will be located in a directory with the same name as the prekx provided by your instructor. However, at this point, you have only added the code to your local repository. To push it to your git repository, run the following:

```
git push -u origin master
```

If your instructor makes any changes to the upstream repository, and you want to merge them into your repository, you will need to run the following command:

```
git subtree pull --prefix PREFIX PREFIX-upstream master --squash
```

Submitting an assignment

When you are ready to submit an assignment, make sure you have pushed all your commits to your course's git server (either GitHub or a GitLab server). If your code hasn't been pushed, then chisubmit will not see it.

Making the submission

To submit an assignment, simply run the following **BEFORE THE DEADLINE**:

```
chisubmit student assignment submit ASSIGNMENT_ID
```

Where `ASSIGNMENT_ID` is the assignment identiker. Your instructor will tell you what identiker to use, but you can also see the list of possible assignment ids by running `chisubmit student assignment list`.

By default, chisubmit will create a submission using the latest commit in your repository. You should see something like this:

```
SUBMISSION FOR ASSIGNMENT p1a (chirc: Part 1)
--------------------------------------------

This is a TEAM submission for team amr-borja with the following students:
 - Anne Rogers
 - Borja Sotomayor

The latest commit in your repository is the following:

      Commit: c53b947521b3d741ee8c5562e4552e3bc06ddc6e
        Date: 2017-01-09 18:05:04
     Message: p1a done!
      Author: Borja Sotomayor <borja@cs.uchicago.edu>

PLEASE VERIFY THIS IS THE EXACT COMMIT YOU WANT TO SUBMIT

Your team currently has 4 extensions

You are going to use 0 extensions on this submission.

You will have 4 extensions left after this submission.

Are you sure you want to continue? (y/n):
```

Before you type `y`, **take a moment to ensure that this commit is the one you actually want to submit**. In particular, if you follow the instructions in the assignment, but there is an error when committing or pushing (e.g., if you encounter a git merge conflict), you may be submitting an older commit (because the last one you tried to do actually failed). If the commit message and date shown by chisubmit don't look right, then you should double-check whether you were able to successfully commit and push your code (a very common mistake is to simply forget to run `git push`)

After you type `y`, you should see the following message:

```
Your submission has been completed.
```

If you want to submit an earlier commit, use the `--commit-sha` option when running chisubmit:

```
chisubmit student assignment submit p1a --commit-sha 2e5969ce281b88bcb3743dc81539623124e63f41
```

Verifying that your submission was made correctly

If you see `Your submission has been completed`, that means you are all set. If you want to be extra sure, the following command will show you the assignments you are registered for, and their submission status:

```
chisubmit student team show TEAMNAME
```

(where `TEAMNAME` should be replaced by your team name)

If you have not submitted an assignment, you will see something like this:

```
ASSIGNMENTS
-----------
ID: p1a
Name: chirc: Part 1
Deadline: 2017-01-09 20:00:00-06:00
NOT SUBMITTED
```

If you have submitted the assignment correctly, you will see something like this:

```
ASSIGNMENTS
-----------
ID: p1a
Name: chirc: Part 1
Deadline: 2017-01-09 20:00:00-06:00
Last submitted at: 2017-01-09 18:11:47-06:00
Commit SHA: 2e5969ce281b88bcb3743dc81539623124e63f41
Extensions used: 0
```

If you want to be super extra double sure, you can log into the GitLab server (https://mit.cs.uchicago.edu (https://mit.cs.uchicago.edu)), then click on your repository on the right side of the page, then click on "Commits" on the left side of the page. You will see the list of commits that are on the server (this is what the graders will see). On the right side of the page, you will see the krst eight characters of each commit's SHA; knd the one that was shown by `chisubmit student team show` command, and verify that it is, indeed the version of the code that you want the graders to grade. To do this, simply click on "Browse kles" to see the state of the repository at that commit

Please note that, if you click on the commit itself (either the title or the identiker), you will see the changes that were included just in that commit. *Don't be alarmed if you don't see all your files in your last commit!* A Git commit only stores the information of what changed in your repository since the previous commit; to see the complete state of your repository (up to and including a given commit), just click on "Browse kles" for that commit in the list of commits.

Re-submitting

Re-submitting is possible, but can sometimes be tricky because, in some cases, the graders may have already started grading your previous submission! So, if you think you may want to resubmit an assignment, you have to remember this important rule:

> **If the deadline for an assignment passes, and you have made a submission for that assignment before the deadline, the graders will be able to start grading it!**

What happens internally is that, once the deadline passes, chisubmit looks at all the submissions that have been already made and flags them as "ready for grading". So, when a grader checks whether there is any grading assigned to them, your submission will show up on their end.

Don't worry: if you have extensions to use, there are ways of ensuring that you can re-submit even after the deadline passes, but it requires being careful about what steps you take to do so. If you knd yourself in that situation, make sure you read the following sections *very carefully*.

For now, let's assume the simplest (and most common) scenario: resubmitting when you have no intention of using any extensions. In this case, things become very simple: you can resubmit as many times as you want *before the deadline*. Then, once the deadline passes, your last submission before the deadline will be the one that the graders will see.

To re-submit before the deadline, just run the submission command like before:

```
chisubmit student assignment submit ASSIGNMENT_ID
```

chisubmit will know that you already made a submission, and will ask you to conkrm that you want to create a new submission:

```
SUBMISSION FOR ASSIGNMENT p1a (chirc: Part 1)
-----------------------------------------

This is a TEAM submission for team amr-borja with the following students:
 - Anne Rogers
 - Borja Sotomayor

You have already submitted assignment p1a

You submitted the following commit on 2017-01-09 18:11:47-06:00:

      Commit: c53b947521b3d741ee8c5562e4552e3bc06ddc6e
        Date: 2017-01-09 18:05:04
     Message: p1a done!
      Author:  Borja  Sotomayor  <borja@cs.uchicago.edu>

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
IF YOU CONTINUE, THE ABOVE SUBMISSION FOR p1a (chirc: Part 1) WILL BE CANCELLED.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

If you continue, your submission will instead point to the following commit:

      Commit: 5b485c80fad76d02df675044b6b4c6c0d32b4ae8
        Date: 2017-01-09 19:17:56
     Message: p1a done! For real! This time!
      Author:  Borja  Sotomayor  <borja@cs.uchicago.edu>

PLEASE VERIFY THIS IS THE EXACT COMMIT YOU WANT TO SUBMIT

Your team currently has 4 extensions

You used 0 extensions in your previous submission of this assignment.
and you are going to use 0 additional extensions now.

You will have 4 extensions left after this submission.

Are you sure you want to continue? (y/n):  y

Your submission has been completed.
```

The safety submission strategy

Since you can submit as many times as you want before the deadline, a good strategy is to always make a **safety submission** well ahead of the deadline. For example, if an assignment has kve tasks, and you have completed four of those tasks but expect to be working on the kfth one right up until the deadline, you should make a submission before you start working on the kfth task. That way, if you end up missing the deadline, there is already a submission in the system with most of your work on it (which may not be as good as a submission with partial or complete work for that kfth task, but still better than not submitting anything at all).

Safety submissions are specially important if you have exhausted your extensions. If the deadline passes and you have not made any submissions, and you are out of extensions, that means an automatic zero on that assignment.

We also recommend that you plan to make your absolute knal submission at least an hour before the deadline, in case there are any issues when you try to submit. If an issue does come up, and you alert an instructor or TA about it with an hour to go, it is very likely that someone will be able to assist you before the deadline. If you wait until three minutes before the deadline to submit, and run into issues, that limits how much assistance they can provide.

Using extensions

If your course allows extensions on assignments, and you want to use an extensions, you do not need to ask an instructor for permission and you do not need to notify them of you intention to do so. All you need to do is wait until *after* the deadline to make your submission, and chisubmit will automatically determine how many extensions you need to use.

If you made a submission before the deadline, and then realize you want to use an extension, make sure you read "Re-submitting after the deadline" below.

If you made no submissions before the deadline, and then submit less than 24 hours after the deadline (meaning you only need to use one extension), chisubmit will include something like this when you run the submission command:

```
Your team currently has 4 extensions

You are going to use 1 extensions on this submission.

You will have 3 extensions left after this submission.
```

To check how many extensions you have left, just run the following:

```
chisubmit student course show-extensions
```

The deadline grace period

Your course may be conkgured so that there is a grace period after the deadline. During this grace period, a submission will not consume an extension. The intent of this grace period is to give some breathing room to students who run into last-minute issues when trying to submit their code (e.g., issues with git). However, you should always aim to submit your code *before* the deadline, for the following reasons:

- The length of the grace period is not disclosed.
- chisubmit informs the instructors of which students have submitted during the grace period. If an instructor sees a team that submits during the grace period for multiple projects, they may begin applying point penalties to any future submissions you make during a grace period.

When you submit during the grace period, chisubmit will print the following before you conkrm your submission:

```
NOTE: You are submitting after the deadline, but the instructor has
allowed some extra time after the deadline for students to submit
without having to consume an extension.
```

And the following once you conkrm your submission:

```
Your submission has been completed.

Your submission was made during the deadline's grace period. This means
that, although your submission was technically made *after* the
deadline, we are counting it as if it had been made before the deadline.

In the future, you should not rely on the presence of this grace period!
Your instructor may choose not to use one in future assignments, or may
use a shorter grace period. Your instructor is also aware of which
submissions are made during the grace period; if you repeatedly submit
during the grace period, your instructor bring this to your attention.
```

Re-submitting after the deadline

As we said earlier, chisubmit flags your submission as being ready for grading once the deadline passes, which means the graders might start looking at your submission right away. This is why re-submitting after the deadline is a bit trickier: the graders may have already started grading your code.

If you made a submission before the deadline and realize (before the deadline) that you want to use an extension after all, then you need to cancel your submission. That way, chisubmit will not flag it as ready for grading when the deadline passes. Simply run this command:

```
chisubmit student assignment cancel-submit p1a
```

You should see something like this:

```
This is your existing submission for assignment p1a:

      Commit: 5b485c80fad76d02df675044b6b4c6c0d32b4ae8
        Date: 2017-01-09 19:17:56
     Message: p1a done! For real! This time!
      Author: Borja Sotomayor <borja@cs.uchicago.edu>

Are you sure you want to cancel this submission? (y/n):  y

Your submission has been cancelled.
```

Once you've done that, just re-submit *after* the deadline, and chisubmit will apply the necessary extensions.

The above command may even work *after* the deadline: if the graders haven't actually started grading your code, chisubmit will still allow you to cancel your submission, even if the deadline has passed. However, if you plan to resubmit after the deadline, you should always aim to cancel your submission before the deadline.

If you try to cancel it after the deadline, and the graders have started grading your code, chisubmit will not allow you to cancel your submission. You will need to ask an instructor to cancel your submission manually, which may involve having to tell the graders to discard any grading they have already done on your submission. This is very inconvenient to the graders, so please try to avoid getting into this situation.

Other useful chisubmit commands

chisubmit student assignment list

Shows the list of assignments, including their deadlines:

```
$ chisubmit student assignment list
p1a  2015-01-12 20:00:00-06:00  chirc: Part 1
p1b  2015-01-22 20:00:00-06:00  chirc: Part 2
p1c  2015-02-02 20:00:00-06:00  chirc: Part 3
p2a  2015-02-18 20:00:00-06:00  chitcp: Part 1
p2b  2015-02-25 20:00:00-06:00  chitcp: Part 2
p3   2015-03-11 20:00:00-05:00  Simple Router
```

chisubmit student assignment show-deadline ASSIGNMENT_ID

Provides more details about the deadline of an assignment:

```
$ chisubmit student assignment show-deadline p1a
chirc: Part 1

      Now:  2015-01-10  20:19:29-06:00
 Deadline:  2015-01-12  20:00:00-06:00

The deadline has not yet passed
You have 1 days, 23 hours, 40 minutes, 31 seconds left
```

If the deadline has passed, it will tell you how many extensions you need:

```
$ chisubmit student assignment show-deadline pa1
Programming Assignment 1

      Now:  2015-01-10  20:21:12-06:00
 Deadline:  2015-01-10  17:00:00-06:00

The deadline passed 0 days, 3 hours, 21 minutes, 12 seconds ago
If you submit your assignment now, you will need to use 1 extensions
```

chisubmit student team show TEAM_ID

Will show you information about the team, including the number of extensions remaining, assignments you are registered for, and extensions used in previous assignments:

```
$ chisubmit student team show the-reticent-reallocs
Team name: the-reticent-reallocs

Extensions available: 3

STUDENTS
--------
jmalloc: Mallock, John  (CONFIRMED)
sprintf: Printeffe, Sarah  (CONFIRMED)

ASSIGNMENTS
-----------
ID: pa1
Name: Programming Assignment 1
Deadline:  2015-01-10  20:00:00-06:00
Last submitted at: 2015-01-10 20:28:39-06:00
Extensions used: 1

ID: pa2
Name: Programming Assignment 2
Deadline:  2015-01-11  20:00:00-06:00
Last submitted at: 2015-01-10 20:28:40-06:00
Extensions used: 0

ID: pa3
Name: Programming Assignment 3
Deadline:  2015-01-12  20:00:00-06:00
NOT SUBMITTED
```