# Git Auto-creation

This is a simple process that does some heavy lifting for your course and otherwise stays out of the way.

## How it Works

To get started, read The Introduction, and the rest of this document. Email `techstaff@cs.uchicago.edu` and indicate which course you would like to use it with. We can start creating project structures immediately.

For each course that is enabled for automation, by default:

- A dedicated Namespace is created in Gitlab, using course numbers by default.
- Repo and Project resources are refreshed from official rosters each night. Any customizations you make are refreshed and merged with the upstream data sources once per hour.
- Each student is given an individual Project beneath the main project (git repository).
- Nominated Graders get appropriate access roles to Projects and the enclosing namespace.
- Subsequent customization is ignored or respected by the automation (add, drop).
- An administrative project is established for Instructors and TAs to optionally continue to modify the provisioner (refresh rate is once per hour).
- The tool is flexible and can be modified to suit different situations.

## Accessing Repositories

The Gitlab server emails each individual at the time they are granted access to a resource. If you have not received an email with repository details and think that you should have, please write Techstaff and let us know.

Hints for finding a repository:

- The current server is named Proj , and you can login and look around
- The entire namespace path is identical to the Course identifier, e.g., `https://proj.cs.uchicago.edu/mpcs-53001-aut-20`
- Use Gitlab built-in search if necessary

# Controlling The Rosters

The overall automation tool is controlled by Techstaff. It must be enabled **before** you can modify the provisioner's behavior. However, online customization of repository and project details are available to instructors and other course staff after Techstaff has set the tool to work for you.

# Repo-based Customizations

Before the automations creates the first repository, it can be configured to read data that is published in **your own** secure Gitlab repository. Such a repository should be created by you at https://proj.cs.uchicago.edu. This project should exist before your course starts, in its own namespace with limited access. It will probably exist after the course ends, and you can use it to capture the list of TAs as it changes over time, along with other metadata, and course artifacts. It is also possible to control multiple simultaneous course instances through this single "master" repository.

Every hour, the automations bot will checkout the head of your `main`, or other nominated branch, and scan a directory for YAML files. It expects to find course and enrollment data in a particular format. Configuration found there is merged with upstream sources to form the execution plan against the Gitlab API. The tool will use your configuration data and the UC official roster data to build project and repository structures for one, or multiple courses.

Because of the aforementioned Roster Config Merging, you should expect to merely *augment* the Registrar's enrollment data. You are able to add and drop students, and elevate or modify other roles. As there is no other datasource at this time, you *must* add TAs to this file, for example. See Advanced Usage.

It is also possible to specify partial information, or split data across multiple files in your repository. All configuration files are merged with others having the same course identifier. For a [contrived] example, one may wish to modify parameters of the repositories on a per-student basis. This structure is easier to maintain if one file is created for each student, instead of placing all data into one YAML file.

# File Format

Files must end in a YAML file-extension and be in a pre-determined directory. If you have created your own repository, as suggested above, just tell us where to fetch the files from. If we created it for you, the directory is named `autorevcon.d` at the root of your `Admin` repository, which is contained in the main Course Project.

All structures must begin with a key that uniquely and globally identifies the course and offering that is currently being taught. The format of the key is opaque to the provisioner and is not parsed, and you are also not restricted to choosing from known keys. Therefore, you are able to create repositories that have no upstream datasource, for testing, future use, and other devices. The example below uses a familiar key structure intended for real courses.

```
CMSC-3456-aut-2020:
  display_name: Optimal Data Structures 2020
  memberships:
    student:
      - tdobes
    grader:
      - chudler
      - ctopper
```

When your configuration is split across multiple files, merging takes place on the basis of the key. So, be sure to use the same key in different YAML files if your intention is to augment, and use *different keys* if you wish to have entirely different top-level projects created for you.

An example to create another course without any associated Registrar data is the same as above, but includes more memberships

```
# sample config to be checked into a file rosters.d/sample.yaml
#
my-globaly-unique-id-1234-aut-2020:
  display_name: my-special Custom Course Taking Place in 2020
  memberships:
    instructor:
      - rdb
    student:
      - kauffman
      - tdobes
    ta:
      - chudler
    grader:
      - ctopper
```

As the example shows, the automations will consume this structure and map the groups of people ("tas", "students") into roles appropriate for use within Gitlab. Only these four groups are recognized at this time, and these are CNetIDs.

#### Configuration Merging

Techstaff will augment any configuration you provide with roster data from the University Registrar. The union of memberships is considered, and scalar values are overridden by your values.

More Roster Configuration Examples