

Installation

For more details on how to install GraderFinder, including on systems like macOS and Ubuntu 16.04, please see the two guides provided by Instructure.

- [Quick Start](#)
- [Production Start](#)

The following installation instructions are based on what can be found in the above two links. They will assume that you are using Ubuntu 18.04, which is the preferred OS for running GraderFinder.

Firstly, create a local directory containing all of GraderFinder's files. This will be the root of the application, and most of the installation will take place from within this folder. In fact, you can perform the entire installation from within this folder.

External Dependencies

Firstly, we will install most of the external dependencies using the following commands:

```
sudo apt-get update
sudo apt install ruby ruby-dev postgresql zlib1g-dev libxml2-dev libsqlite3-dev libpq-dev libxmlsec1-dev
```

In the second command, you will be asked to confirm the installation, type Y for yes.

Node.js

Now we need to install Node.js and Yarn

Node.js needs to be installed from a source that provides a newer version than Ubuntu 18.04 has by default:

```
$ curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
$ sudo apt-get install -y nodejs
```

Yarn

Use the following commands to install Yarn

```
$ curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
$ echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
$ sudo apt-get update && sudo apt-get install yarn=1.10.1-1
```

Postgres User Configuration

If postgres is not already running, which is probably the case if you are using WSL, start it with the command:

```
$ sudo service postgresql start
```

Next run the following two commands to create a postgres user:

```
sudo -u postgres createuser $USER
sudo -u postgres psql -c "alter user $USER with superuser" postgres
```

Bundler

Bundler is used to manage Ruby gems. GraderFinder is built on a version of Canvas that uses Bundler version 2.1.4. This was the latest version at the start of development, but Bundler 2.2.0 was released on December 10, 2020. Therefore, the version must be specified when installing Bundler. This is not yet reflected in Instructure's guides.

```
sudo gem install bundler -v '>= 1.13.3, <= 2.1.4'
```

GraderFinder Dependencies

If you are not already in your GraderFinder directory, please go there and stay there for the rest of the installation.

Run the following commands to install GraderFinder's gems and other dependencies:

```
$ cd <graderfinder_dir> #wherever your graderfinder directory is
$ bundle install
$ yarn install --pure-lockfile
```

Yarn install may need to be run twice if any errors occur in the first run.

Data Setup

The following commands sets up Rails configuration files with the default configuration files:

```
$ for config in amazon_s3 delayed_jobs domain file_store outgoing_mail security external_migration; \
do cp -v config/$config.yml.example config/$config.yml; done

$ cp config/dynamic_settings.yml.example config/dynamic_settings.yml
```

File Generation/Compiling Assets

The following command compiles GraderFinder's assets.

```
bundle exec rails canvas:compile_assets
```

If you change files such as javascript or css files, then you will have to recompile assets. Generally, if you change something and the change does not take place immediately, you will have to recompile assets. The command **bundle exec rails canvas:compile_assets_dev** can be used to do this and is faster because it only compiles files that are relevant to development.

Database Configuration

The following will setup graderfinder with the default database configuration files, make sure that postgres is running:

```
$ cp config/database.yml.example config/database.yml
$ createdb canvas_development
$ bundle exec rails db:initial_setup
```

After running the last command here, you will be asked to enter an email, password, and account name. These will be used to access the admin account in GraderFinder.

Test database configuration

The following commands will setup a test database to use with Canvas. I don't know why this is part of the instructions in the quickstart guide, we never used the test database during development, but its here for the sake of completeness and also to explain how to get around a common error when setting this up.

The commands to setup a test database are:

```
psql -c 'CREATE USER canvas' -d postgres
psql -c 'ALTER USER canvas CREATEDB' -d postgres
createdb -U canvas canvas_test
psql -c 'GRANT ALL PRIVILEGES ON DATABASE canvas_test TO canvas' -d canvas_test
psql -c 'GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO canvas' -d canvas_test
psql -c 'GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO canvas' -d canvas_test
RAILS_ENV=test bundle exec rails db:test:reset
```

You will probably run into an error in the third command **createdb -U canvas canvas_test** telling you about failed peer authentication. To fix this, enter the following command to enter postgres console:

```
psql postgres
```

In the postgres console, type:

```
show hba_file
```

This will give you the location of the configuration file we will need to edit.

Editing pg_hba.conf

Open this file and you will see something like the following at the bottom of the file:

```
# Database administrative login by Unix domain socket
local  all                postgres                    peer

# TYPE  DATABASE      USER      ADDRESS      METHOD

# "local" is for Unix domain socket connections only
local  all          all                trust
# IPv4 local connections:
host   all          all            127.0.0.1/32    trust
# IPv6 local connections:
host   all          all            ::1/128         md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local  replication  all                peer
host   replication  all            127.0.0.1/32    md5
host   replication  all            ::1/128         md5
```

In your file, change each METHOD field to trust where it is set in the example above. You may want to find a safer way to do this if you are releasing the application for production.

More information about this issue can be found here: * <https://gist.github.com/AtulKsol/4470d377b448e56468baef85af7fd614>

Apache Installation

This part of the guide comes from the production start guide. At the moment, you would be able to run Canvas with all of its assets with the following command:

```
bundle exec rails s
```

but you would not be able to download files or zip submissions. To do that, first we will install apache and all of it's dependencies. Remember that this part of the guide also takes place withing your GraderFinder directory. Run the following commands:

```
$ sudo apt-get install passenger libapache2-mod-passenger apache2
$ sudo a2enmod rewrite
$ sudo service apache2 restart

$ sudo a2enmod passenger
$ sudo service apache2 restart

$ sudo a2enmod ssl
$ sudo service apache2 restart
```

Apache Configuration

We will now need to configure apache. This is probably the most complex part of the guide.

First create the configuration file:

```
sudo nano /etc/apache2/sites-available/canvas.conf
```

And paste the following text:

```

<VirtualHost *:80>
  ServerName canvas.example.com
  ServerAlias canvasfiles.example.com
  ServerAdmin youremail@example.com
  DocumentRoot /var/canvas/public
  RewriteEngine On
  RewriteCond %{HTTP:X-Forwarded-Proto} !=https
  RewriteCond %{REQUEST_URI} !^/health_check
  RewriteRule (.*?) https://%{HTTP_HOST}%{REQUEST_URI} [L]
  ErrorLog /var/log/apache2/canvas_errors.log
  LogLevel warn
  CustomLog /var/log/apache2/canvas_access.log combined
  SetEnv RAILS_ENV production
  <Directory /var/canvas/public>
    Allow from all
    Options -MultiViews
  </Directory>
</VirtualHost>
<VirtualHost *:443>
  ServerName canvas.example.com
  ServerAlias canvasfiles.example.com
  ServerAdmin youremail@example.com
  DocumentRoot /var/canvas/public
  ErrorLog /var/log/apache2/canvas_errors.log
  LogLevel warn
  CustomLog /var/log/apache2/canvas_ssl_access.log combined
  SSLEngine on
  BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
  # the following ssl certificate files are generated for you from the ssl-cert package.
  SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
  SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
  SetEnv RAILS_ENV production
  <Directory /var/canvas/public>
    Allow from all
    Options -MultiViews
  </Directory>
</VirtualHost>

```

The most important things to change here are:

- SetEnv RAILS_ENV production to SetEnv RAILS_ENV development (2 instances)
- Directory to your GraderFinder directory (2 instances)
- DocumentRoot to the public folder in your GraderFinder directory (2 instances)

You may also want to change the following:

- ServerName
- ServerAdmin
- SSLCertificateField
- SSLCertificateKeyFile

The latter two are for when the program is going to be released and already has an SSL certificate.

You will also want to change the options within the directory segment to the following:

```

<Directory /var/canvas/public>
  Options All
  AllowOverride All
  Require all granted
</Directory>

```

Of course, /var/canvas should have already been set to your directory.

Optimizing File Downloads

To optimize file downloads you will want to install xsendfile with the following commands:

```
sudo apt-get install libapache2-mod-xsendfile
sudo apachectl -M | sort
```

Now reopen the config file **/etc/apache2/sites-available/canvas.conf** and add the following two lines under each virtual host:

```
XSendFile On
XSendFilePath /var/canvas
```

Remember to replace the file path in the second command with your GraderFinder directory.

Background Jobs

Configure background jobs with the following commands:

```
$ sudo ln -s <your_graderfinder_dir>/script/canvas_init /etc/init.d/canvas_init
$ sudo update-rc.d canvas_init defaults
$ sudo /etc/init.d/canvas_init start
```

Replace `your_graderfinder_dir` with your graderfinder directory. You may want to check the file **/etc/init.d/canvas_init** with a text editor and make sure that it is configured for development instead of production.

Starting Canvas

Every time that you want to start graderfinder, you will want to run the following commands to start all of the servers that graderfinder needs. If you are doing this right after installation, they will all probably already be running.

Commands:

```
$ sudo service postgresql start
$ sudo service apache2 start
$ sudo /etc/init.d/canvas_init start
$ bundle exec rails s
```

The final command starts graderfinder, you will be able to access it in localhost:3000.

If background jobs are not working, you may want to open another terminal and use the command **sudo /etc/init.d/canvas_init run**, which allows them to run in the foreground. Remember to close the instance already running beforehand.

GraderFinder In-Program Setup

After starting GraderFinder, you will want to do the following.

Enable Self Registration

- Log in with the email and password you entered during database initialization
- Click on the admin button on the sidebar
- Click on the account you named in database initialization
- Click on the authentication tab in the sidebar of the resulting page
- Scroll down and select All Account Types
- Click on save settings in the bottom right corner

Enable Teacher Course Creation

- From the same account click on the settings tab in the sidebar
- Scroll down to the bottom of the page and select Teachers
- Click on update Settings

For more information check Canvas's guides.