Install Discourse on Windows for development

- Installing Discourse
  - Install Discourse Dependencies
  - Clone Discourse
  - Setup Database
  - Bootstrap Discourse
  - Creating a Command to Start Discourse
  - Notes About Windows Environment
  - Invalid Redis connection to a specific port / connection refusal
  - Invalid connection to psql port
  - The Result
  - Invalid Redis connection to a specific port / connection refusal
  - Invalid connection to psql port
  - The Result

:information_source: This tutorial has been tested on Windows 10 and 11.

To set up a development environment for Discourse on Windows, you can do it using Windows Subsystem for Linux feature.

This setup requires the WSL 2 installation. It is only available in Windows 10 builds 18917 or higher.

We'll assume that you already installed Windows Subsystem for Linux 2 (Ubuntu) on your Windows 10 system. **WARNING:** Install Ubuntu 18.04, and not 20.04 since some installations will fail on 20.04. For more information see June 30th, 2020 notes at the bottom of this post.

Let's begin!

Installing Discourse

1. **Initially follow the steps from the topic Beginners Guide to Install Discourse on Ubuntu for Development** until the step **Clone Discourse**.

[quote="Arpit Jalan, post:1, topic:14727, username:techAPJ"]

Install Discourse Dependencies

You will need the following packages on your system:

- Git
- rbenv or asdf
- ruby-build
- Ruby (latest stable)
- Rails
- PostgreSQL
- SQLite
- Redis
- Node.js
- pnpm
- libvips
- MailHog**
- ImageMagick**

** *optional*

Now that we have installed Discourse dependencies, let's move on to install Discourse itself.
[/quote]

2. Before setting up the database you have to start PostgreSQL service & Redis server manually using following commands

```
sudo service postgresql start
redis-server --daemonize yes
```

3. **Then go through all the remaining steps of the Ubuntu guide**.

[quote="Arpit Jalan, post:1, topic:14727, username:techAPJ"]

Clone Discourse

Clone the Discourse repository in `~/discourse` folder:

```
git clone https://github.com/discourse/discourse.git ~/discourse
```

`~` *indicates home folder, so Discourse source code will be available in your home folder.*

Setup Database

Create role **with the same name as your ubuntu system username**:

```
sudo -u postgres createuser -s "$USER"
```

Bootstrap Discourse

Switch to your Discourse folder:

```
cd ~/discourse
```

Install the needed gems

```
source ~/.bashrc
bundle install
```

Now that you have successfully installed gems, run these commands:

```
bundle exec rake db:create
bundle exec rake db:migrate
RAILS_ENV=test bundle exec rake db:create db:migrate
```

Try running the specs:

```
bundle exec rake autospec
```

All the tests should pass.

Start rails server:

```
bundle exec rails server
```

You should now be able to connect with your Discourse app on http://localhost:3000 - try it out!

In a separate terminal instance, navigate to your discourse folder (`cd ~/discourse`) and run:

```
bin/dev --only ember
```

This starts the frontend bundler; the app remains served from http://localhost:3000. [/quote]

Creating a Command to Start Discourse

Now your development environment is almost ready. The only problem is every time when you open Ubuntu on Windows you have to start the PostgreSQL service & Redis server manually. Don't worry we can have a workaround for this by creating a custom command :wink:

```
cd ~
```

Create a new file using the command `nano start-discourse` and paste the content below then save and exit.

```
#!/bin/bash

# to start PostgreSQL
sudo service postgresql start

# to start Redis server
redis-server --daemonize yes
```

Now modify the CHMOD using below command

```
chmod +x start-discourse
```

And copy the file to your bin folder

```
sudo cp start-discourse /usr/bin/
```

It's done. Now, whenever you open the Ubuntu bash just run the command below and start developing :+1:

```
start-discourse
```

Alternatively, if you are using Windows 10 enterprise, pro, or education edition then you can create a Linux virtual machine in hyper-v to set up the Discourse dev environment.

Notes About Windows Environment

[details=As of June 30, 2020:] [quote="Andrea Habib, post:56, topic:75149, username:AndreaHabib"] With the introduction of Windows 10 build 2004 and WSL 2 and many of you might run into some issues since Discourse now requires WSL 2 to actually run. **For starters:** If you haven’t already, you would want to update your system to Windows 2004 (19041) **Note: Don’t install using Windows Update in the system settings as you might get errors mid-installation. Also make sure you are on Windows 1903 (18362) or 1909 (18363), you can search winver for that.**

- Go to: Download Windows 10
- Click the Update Now button under Windows 10 May 2020 Update to download the update assistant and let it install and finish
- If you get any errors, then download the media creation tool, run it, and create a bootable flash drive (at least 8GB flash) and use it to install Windows.

When you are done with the Windows installation, you will need to manually install WSL 2 But first, we need to make sure that 2 features are enabled:

- Virtual Machine Platform which can be enabled directly from your Windows by going to Programs and Features and then enabling Optional Features in **Turn Windows features on or off**
- The other is Virtualization which can be only enabled from your BIOS. However this option can vary from one motherboard to the other. eg: on AMD it’s called SVM. So please look up your specific motherboard.

Now that you have both features enabled:

- Install Ubuntu 18.04, and not 20.04 since some installations will fail on 20.04.
- Go here and update Linux Kernel: Link in replies
- Go to this link and follow the steps: Install Windows Subsystem for Linux (WSL) on Windows 10 | Microsoft Docs
- Go to powershell (run as administrator) and run: `wsl --set-version Ubuntu-18.04 2` or `wsl --set-version Ubuntu 2`, whichever distro shows up in `wsl -l -v`
- Conversion takes time, a few minutes
- Run `wsl -l -v` to make sure that Ubuntu is running on WSL 2

Now you can run Ubuntu-18.04 or Windows Terminal with Ubuntu and start installing the dependencies and tools

Note: If you are running into an error installing one of the gems during `bundle install`: Make sure you are on Ubuntu-18.04 and not 20.04

However, if you already have postgreSQL 12 and running on port 5432, you might run into some issues, since the dependencies install postgreSQL 10, which might increment the port to 5433. To fix that:

- Open your folder in a text editor (VS Code for example)
- Go to `config/database.yml`
- Under development and under adapter, add a new line and then add: **port: 5433**
- Make sure you do the same for test and profile.

If the above doesn't work, make sure to check if PostgreSQL is running and at which port via `pg_isready`. Then follow the same instructions above, setting the port equal to the one the database is running on.

FOR THE LINUX KERNEL: Updating the WSL 2 Linux kernel | Microsoft Docs [/quote]

[quote="Ricky Chon, post:61, topic:75149, username:RickyC0626"] A solution that does not require explicit port configuration in `database.yml`

Some errors during database creation and migration:

Invalid Redis connection to a specific port / connection refusal

```
bundle exec rake db:create
```

```
rake aborted!
Redis::CannotConnectError: Error connecting to Redis on localhost:6379 (Errno::ECONNREFUSED)
...

Caused by:
Errno::ECONNREFUSED: Connection refused - connect(2) for 127.0.0.1:6379
...

Caused by:
IO::EINPROGRESSWaitWritable: Operation now in progress - connect(2) would block
...

Tasks: TOP => db:create => db:load_config => environment
(See full trace by running task with --trace)
```

From this error we can see that the port Redis uses is `6379`, and we can fix this by starting the Redis server via that port.

By default, `redis-server --daemonize yes` should work, but if not, use:

```
redis-server --daemonize yes --port 6379
```

Check status of Redis instance:

```
redis-cli

127.0.0.1:6379> ping
PONG
```

Invalid connection to psql port

```
bundle exec rake db:migrate
```

```
PG::ConnectionBad: could not connect to server: No such file or directory
       Is the server running locally and accepting
       connections on Unix domain socket "/var/run/postgresql/.s.PGSQL.5432"?
```

From this error we can see that PostgreSQL is having trouble connecting to port `5432`. Usually starting the service should work

```
sudo service postgresql start
```

If this still doesn’t work and the error persists, try checking to see which port the service is listening from, and if necessary, change the port to the one you want in postgresql.conf:

```
pg_isready
```

```
/var/run/postgresql:5432 - no response
or
/var/run/postgresql:5432 - accepting connections
```

The Result

Once we have done the above or executed the `start-discourse` command, the two instances should run on their default/specified ports. To check their status via the default Windows `cmd` terminal, we can run:

```
netstat -anop tcp
```

which will show something like this:

```
Proto    Local Address        Foreign Address        State           PID
TCP      127.0.0.1:5432       0.0.0.0:0              LISTENING       17768
TCP      127.0.0.1:6379       0.0.0.0:0              LISTENING       17768
```

We have now confirmed that both our postgresql and redis-server instances are running.

We can also check the status of the instances on Ubuntu or WSL, with the following commands:

```
lsof -i
```

```
COMMAND     PID        USER  FD  TYPE  DEVICE  SIZE/OFF  NODE  NAME
mailcatch  3244  rickyc0626  7u  IPv4   36127       0t0   TCP  localhost:1025 (LISTEN)
mailcatch  3244  rickyc0626  8u  IPv4   36128       0t0   TCP  *:socks (LISTEN)
redis-ser  3287  rickyc0626  6u  IPv6   29352       0t0   TCP  *:6379 (LISTEN)
redis-ser  3287  rickyc0626  7u  IPv6   29353       0t0   TCP  *:6379 (LISTEN)
```

```
sudo ss -plunt | grep postgres
```

```
tcp  LISTEN  0  128  127.0.0.1:5432  0.0.0.0:*  users:(("postgres",pid=3070,fd=7))
```

```
pg_lsclusters
```

```
Ver Cluster Port Status Owner    Data directory              Log file
10  main    5432 online postgres /var/lib/postgresql/10/main /var/log/postgresql/postgresql-
10-main.log
```

From here, these commands should work with no major issues, without needing to modify the `database.yml` file:

```
bundle exec rake db:create
bundle exec rake db:migrate
RAILS_ENV=test bundle exec rake db:create db:migrate
```

Any further issues that show up can be addressed in the future. [/quote]

[/details]

[details=As of July 1, 2020]

[quote="Ricky Chon, post:61, topic:75149, username:RickyC0626"] A solution that does not require explicit port configuration in `database.yml`

Some errors during database creation and migration:

Invalid Redis connection to a specific port / connection refusal

```
bundle exec rake db:create
```

```
rake aborted!
Redis::CannotConnectError: Error connecting to Redis on localhost:6379 (Errno::ECONNREFUSED)
...

Caused by:
Errno::ECONNREFUSED: Connection refused - connect(2) for 127.0.0.1:6379
```

```
...

Caused by:
IO::EINPROGRESSWaitWritable: Operation now in progress - connect(2) would block
...

Tasks: TOP => db:create => db:load_config => environment
(See full trace by running task with --trace)
```

From this error we can see that the port Redis uses is `6379`, and we can fix this by starting the Redis server via that port.

By default, `redis-server --daemonize yes` should work, but if not, use:

```
redis-server --daemonize yes --port 6379
```

Check status of Redis instance:

```
redis-cli

127.0.0.1:6379> ping
PONG
```

Invalid connection to psql port

```
bundle exec rake db:migrate
```

```
PG::ConnectionBad: could not connect to server: No such file or directory
       Is the server running locally and accepting
       connections on Unix domain socket "/var/run/postgresql/.s.PGSQL.5432"?
```

From this error we can see that PostgreSQL is having trouble connecting to port `5432`. Usually starting the service should work

```
sudo service postgresql start
```

If this still doesn't work and the error persists, try checking to see which port the service is listening from, and if necessary, change the port to the one you want in postgresql.conf:

```
pg_isready
```

```
/var/run/postgresql:5432 - no response
or
/var/run/postgresql:5432 - accepting connections
```

The Result

Once we have done the above or executed the `start-discourse` command, the two instances should run on their default/specified ports. To check their status via the default Windows `cmd` terminal, we can run:

```
netstat -anop tcp
```

which will show something like this:

```
Proto    Local Address        Foreign Address        State           PID
TCP      127.0.0.1:5432       0.0.0.0:0              LISTENING       17768
TCP      127.0.0.1:6379       0.0.0.0:0              LISTENING       17768
```

We have now confirmed that both our postgresql and redis-server instances are running.

We can also check the status of the instances on Ubuntu or WSL, with the following commands:

```
lsof -i
```

```
COMMAND     PID        USER  FD  TYPE  DEVICE  SIZE/OFF  NODE  NAME
mailcatch  3244  rickyc0626  7u  IPv4   36127       0t0   TCP  localhost:1025 (LISTEN)
mailcatch  3244  rickyc0626  8u  IPv4   36128       0t0   TCP  *:socks (LISTEN)
redis-ser  3287  rickyc0626  6u  IPv6   29352       0t0   TCP  *:6379 (LISTEN)
redis-ser  3287  rickyc0626  7u  IPv6   29353       0t0   TCP  *:6379 (LISTEN)
```

```
sudo ss -plunt | grep postgres
```

```
tcp  LISTEN  0  128  127.0.0.1:5432  0.0.0.0:*  users:(("postgres",pid=3070,fd=7))
```

```
pg_lsclusters
```

```
Ver Cluster Port Status Owner    Data directory              Log file
10  main    5432 online postgres /var/lib/postgresql/10/main /var/log/postgresql/postgresql-
10-main.log
```

From here, these commands should work with no major issues, without needing to modify the `database.yml` file:

```
bundle exec rake db:create
bundle exec rake db:migrate
RAILS_ENV=test bundle exec rake db:create db:migrate
```

Any further issues that show up can be addressed in the future. [/quote]

[/details]

Last Reviewed by @SaraDev on [date=2022-06-15 time=19:00:00 timezone="America/Los_Angeles"]

□□□ #2
□ □□□ □□□ 2026-10-31 13:27:34 UTC
□ □□□ □□□ 2026-10-31 13:27:35 UTC