

# GoToSocial setup guide

*Note: This guide is currently a work in progress. These instructions may not be complete, and are subject to change.*

Welcome to this guide for putting together a Website League node! Hosting a webserver on the public Internet can be a bit daunting, but hopefully this guide will help guide you through what's necessary to do so. There may be parts of the guide which are hard to understand, but try to take it slow, and if you need help, there's folks on the Discord who are willing to give assistance (the author of this guide included.) It's in this guide's author's opinion that everyone deserves to be able to have their own slice of the Internet, and knowledge should not be a barrier to that.

There's a few parts to setting up any webserver that uses ActivityPub - you have to obtain a **domain name** and a **server**, **link the domain** to the server, and then **set up and configure your server** to run your software of choice.

- [Planning a Node](#)
- [Setting Up your Domain Name](#)
- [Preparing Your VPS](#)
- [Installing GoToSocial](#)
- [Quick Reference for common admin tasks](#)

# Planning a Node

## Decisions

The two big things you *can't* change about your node once you've started it is your **domain name**, and your **software**. There are currently no known migration options for migrating from one federating software to another, and ActivityPub is **extremely** reliant on the domain name.

It's effectively impossible to change either without effectively starting over with your instance, and re-using the same domain can have complications (although, you could use a (different) subdomain without problems.) You will want to make sure you are comfortable moving forward with the two things you choose in this regard.

## Domain Name

As far as domain name goes, you'll need to purchase it from a domain registrar - there are several options available to you in this regard. Choose one that's trustworthy, and one that you feel would have your back in the event of a security issue (i.e. your account gets hijacked.) The author of this document uses [Porkbun](#), but there are other options out there; a few that have been mentioned are [Namecheap](#), [Glauca](#), and [Cloudflare](#).

Keep in mind that name registrars will often offer a deeply discounted price for their domain names, and a much steeper price for subsequent years. Assuming you will be running this node for at least over a year, be mindful of what the normal renewal price will be, so you don't get stuck with a \$40 renewal for what you thought was a \$2 domain!

Once you've picked out your registrar of choice, go looking for a domain name! Keep in mind this domain will be at the end of your (and your users, if this is a multi-user node) username, so pick something that's both available and that you'd like at the end of your username! (For example, if you chose the domain sickos.social and your username was critter, your full username would be @critter@sickos.social.) You don't have to necessarily be elegant with this, but it can be fun coming up with a good domain!

Once you've come up with (and purchased) a good domain name for your instance, it's time to think about the software you'll use.

## Fediverse Software

As far as software goes, you have a few options. The two known/adapted to work for the Website League as of writing this are [GoToSocial](#) and [Akkoma](#). This guide will focus on installing **GoToSocial**, but you may want to compare and contrast both to see which one you would prefer to

run.

GoToSocial is lighter and simpler to spin up, but doesn't come with its own UI/front-end for posting; you'd need to use (or install) an external client, such as [Phanpy](#) or [Tusky](#). Akkoma, on the other hand, is more complicated, but it also offers several front-ends which can then be re-skinned/re-coloured to your liking, among other bits of flexibility.

There are other federating software choices out there - such as Sharkey or the well-known Mastodon, but these are not included here as they either lack features or have flaws which disqualify them for use with the Website League. You may be able to patch the software to make it work, but this is out of the scope of this guide.


## Server

Now that you've picked what software you'd like to use, and the domain name you'd like, the last thing you need to choose is where to put your instance! There are, again, multiple options for this - [Vultr](#), [Hivelocity](#), [Linode](#), [DigitalOcean](#), and [OVH](#) are option ([Oracle](#) is *also* an option, and you *may* be able to get it for free, with the caveat that it's Oracle. However, Oracle and Hetzner, two major hosting options, disallow adult content on their servers, which mostly takes them out of the running for use on a social network that permits adult content.) For this guide, we will be using Vultr.

For this guide, we're going to get a **regular cloud compute shared CPU** VM with **1 vCPU, 1 GB RAM, and 25 GB SSD** for **\$5/mo**. If you start to run into resource constraints, you can always upgrade your VM. For the operating system, choose **Rocky Linux 9**.

### Choose Plan

AMD High Performance <sup>?</sup> Intel High Performance <sup>?</sup> High Frequency <sup>?</sup> Regular Cloud Compute <sup>?</sup>

Name	Cores	Memory	Storage	Bandwidth	Price
 <b>25 GB SSD</b>	1 vCPU	1 GB	25 GB SSD	1 TB	<b>\$5/month</b> \$0.007/hour

These size constraints make sense for the software we are running - GoToSocial is pretty lightweight. If you're running other software, you may want to consider something with more RAM.

Give your VM a hostname and hit Deploy Now. After a moment, it'll start up and give you an IP address. You're now the proud owner of a virtual machine! Click on your VM name, then take note of the **IP address** and **IPv6 address** it gives you as we move on to the next section - configuring your domain name!

IP Address: 10.13.12.25 

IPv6 Address: fd42:694:2069:acab:5400:05ff:fe1c:587b 

If you aren't using Vultr, note that some hosts may not have IPv6 addresses enabled by default. For example, Digital Ocean allows you to [automatically enable it during VM creation](#) or [manually afterwards](#).

# Setting Up your Domain Name

This page assumes you have, at a *minimum*, the following:

- A domain name you wish to use
- A VPS and its corresponding IP address as well as IPv6 address

If you don't have either, consider checking out the previous page, [Introduction and Planning a Node](#), then come back here once you have those.

It's now time to configure your domain name so that it links to your VPS, which will eventually hold your Website League node. For this guide, we will be using Porkbun, but whatever domain registrar you're using likely has its own interface for doing this.

Be careful if you're using Cloudflare to manage your domain - Cloudflare has been known before to cause federation issues by blocking server requests from time to time. You'll likely need to set up exceptions so it doesn't block federated traffic.

To link your domain to your VPS, you will create two records for the domain - an A record and an AAAA record. Computers don't inherently understand what a domain means, but they use records like this to translate domain names into IP addresses - which they do understand.

This complete guide will only cover setting up an instance on the same domain you will actually be hosting it from. It is possible to make where you host the instance differ from the instance name (the bit that goes at the end of your username), but this is a more advanced topic that can cause federation issues if done improperly. We will not be covering this in this guide.

Fill in the records with the following:

Record Type	Host	Answer
A	Your domain name	Your VPS's IP address
AAAA	Your domain name	Your VPS's IPv6 address

You can set the TTL to whatever you'd like. I like to set it to 300 to start for quick updates, then increase it down the line when it's less likely to change. Below is an example of what an A record might look like on Porkbun, if your IP address was **10.13.12.25** and your desired domain was **website.420131269.xyz**:

Type

A - Address record

Host

Leave blank to create a record for the root domain. Use \* to create a wildcard. Please note that ALIAS records do not support wildcards.

website

.420131269.xyz

Answer

10.13.12.25

TTL

600

Priority

Notes

This is for your own use and does not affect DNS.

my website league website

Cancel

Add

Note: On Porkbun, if I wished to just host it on 420131269.xyz with no subdomain, I would just leave that box blank. This may be different depending on how your registrar is set up. Feel free to ask in the Discord if you're uncertain!

Once you're done with that, it's time to start setting up the server!

# Preparing Your VPS

This page makes the following assumptions:

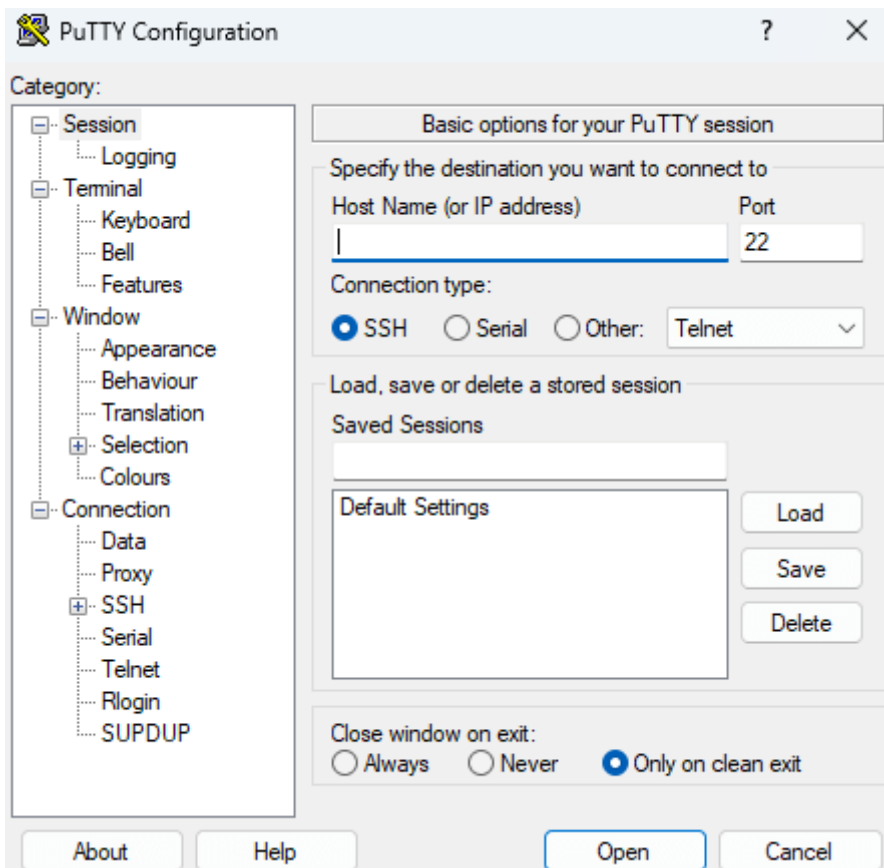
- You have a running VPS.
- You have a domain name that's pointed to your VPS.

If this is not the case, go through the first two pages, [Introduction and Planning a Node](#) and [Setting Up your Domain Name](#), then come back here.

## Logging Into Your VPS

Your host should have given you a username and password - the user is likely root. If the user is not root, it's likely it has "sudo" permissions - meaning you can use it to become root once logged in (and you can disregard the "Creating a User Account" steps later in this guide.) You likely also have a console you can log in through on your host - and this is useful, especially if you goof something up and can't log in any other way - but it's more ideal to be able to log in through something called SSH. This basically gives you a command prompt directly into your server, and, among other things, it's likely much easier to copy-paste info to and from it.

Assuming you're running Windows, one program you can use for logging into SSH is called [PuTTY](#). You'll want to download the MSI "Windows Installer" - likely 64-bit x86, or 32-bit x86 if your PC is *really* old. All the defaults should be fine. Once you start up PuTTY, you'll be presented with a window like this:



You'll put your VPS's IP address under "Host Name" and click Open, then press Accept. It'll then ask you for your username and password. Type in the details that your hosting provider gave you. Note that it won't show you typing any characters when entering your password - don't panic, that's by design. Once you do that, you should be in!

```
login as: root
root@      's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last failed login: Thu Sep 26 01:50:19 UTC 2024 from 167.114.39.217 on ssh:notty
There were 110 failed login attempts since the last successful login.
[root@vultr ~]#
```

Before you continue, change the root password real quick by running this command:

```
passwd
```

It will ask you for a password. Type in whichever password you'd like, hit enter, and then type it in again. It should say "**passwd: all authentication tokens updated successfully.**" If not, try again.

Let's also make sure nano is installed. This is a console text editor that's probably one of the friendlier options out there. (If you know you'd prefer vim/emacs, you can go ahead and use that instead!) Run this command:

```
yum install -y nano
```



It should either install nano, or basically say "is already installed, nothing to do." Either way, you're good!

# Securing Your VPS

At this point, you only have a root account with a password. While this *technically* works, this is not a good idea. The root account is one that has ultimate power over the server, and it's not one you typically want to be able to log directly into; instead, it's best to have a user account that you can then promote yourself to root through.

## Creating a New User with Sudo Permissions

We're going to disable logging into the system as root over the internet. Instead, you're going to log in as a new user you're about to create, that can then be promoted to root using a command called "sudo". Run the following commands, replacing USERNAME\_HERE with your desired username:

```
adduser USERNAME_HERE  
passwd USERNAME_HERE
```

After running the passwd command, it will once again ask you for a password, this time for your new user. Type in whichever password you'd like, hit enter, and then type it in again. It should say **"passwd: all authentication tokens updated successfully."** If not, try again.

Next, run this command to add them to the wheel group, which is a special group that can be "promoted" to root using sudo:

```
usermod -aG wheel USERNAME_HERE
```

This works on Rocky Linux 9, and may also work on other Red Hat-based distros. If this doesn't work, try replacing "wheel" with "sudo". If that doesn't work either, you can simply give your specific user sudo permissions by running the command **EDITOR=nano visudo**, then adding the following line:

```
USERNAME_HERE ALL=(ALL:ALL) ALL
```

At this point, try logging out and logging back in as your new user. Then, run the following command to promote yourself to sudo:

```
sudo su -
```

It'll ask for your password - type it in. It should take you to a command prompt as root. If it says something along the lines of "This user is not in the sudoers file. This incident will be reported", then you'll need to log back in as root and check the sudoers file. If you need help with this, there are folks in the Discord!

If it did work, though, it's time to lock out remote root login.

## Disabling Root Login

Before proceeding, make absolutely certain you can both log in as your new user **and** you can run **sudo su** - successfully. If you proceed, and you find you are unable to do either of these things, you will be locked out of SSH access to root altogether, and will need to use your webhost's console to re-enable root login.

Run the following command:

```
sudo nano /etc/ssh/sshd_config
```

In this file, look for a line that says "PermitRootLogin yes". You will want to change this line to instead say

```
PermitRootLogin no
```

To use Nano, you can move the text cursor with the arrow keys, and type as you normally would. When you're finished, press Ctrl+X. If you've made any changes, you'll be asked whether you want to save them or not - hit Y for Yes, N for No, then Enter to write the changes to the file you were writing (or change the name if you wish to write it to a new/different file.)

Save the file, and then restart SSH:

```
sudo systemctl restart sshd
```

From now on, you will only log into your server with the username and password you gave it. Trying to log in with the root credentials over SSH will not work. They will still work when logging in via your webhost's console, however, so keep your root password somewhere safe just in case you need it.

While not included here, you may also want to consider setting up login using a private key. This is more secure than a password. Steps to do this will be included at a later date, but it should be something you can find online or ask for help with.

# Setting Up a Firewall

We're now going to set up the firewall so it allows you to log in to SSH and allows it to act as a webserver, but doesn't allow anyone else to connect to it otherwise.

These instructions should work on Rocky Linux 9 and any other recent RHEL-based Linux distro. If you're using a different distro, instructions to set up the firewall may be different.

First, make sure firewalld is installed and running:

```
sudo yum install -y firewalld
sudo systemctl enable --now firewalld
```

Then, check to see what your default zone is, then list all the rules in your current zone:

```
sudo firewall-cmd --get-default-zone
```

```
# Replace public in all following commands with whatever zone your default is
sudo firewall-cmd --zone=public --list-all
```

If ssh isn't in services, add it:

```
sudo firewall-cmd --zone=public --add-service=ssh
```

Then, add HTTP and HTTPS:

```
sudo firewall-cmd --zone=public --add-service=http
sudo firewall-cmd --zone=public --add-service=https
```

Finally, run this command to make this configuration permanent:

```
sudo firewall-cmd --runtime-to-permanent
```

Hopefully, running these commands did not result in you being disconnected from the server. In the event this happens, however, don't panic - use your webhost's console to log in and add SSH to the currently running firewall config.

## Bonus - Setting Up Your Webhost's Firewall

Some (not all) webhosts offer a configurable firewall in front of your VPS. If you are using one such webhost, you can configure it to only permit SSH and HTTP(S) as well. The benefit of this is that it blocks this traffic from even getting to your VPS in the first place. This guide will go over setting up

such a firewall on Vultr.

In Vultr, go to Network > Firewall, then click "Add Firewall Group". Give it a name, and then under Inbound IPv4 Rules, add a few rules so that the list looks like this:

#### Inbound IPv4 Rules

Action	Protocol	Port (or range) <sup>?</sup>	Source	Notes	Action
accept	SSH	22	Anywhere	0.0.0.0/0	<a href="#">Add note</a> +
accept	SSH	22	0.0.0.0/0		
accept	TCP (HTTP)	80	0.0.0.0/0		
accept	TCP (HTTPS)	443	0.0.0.0/0		
drop	any	0 - 65535	0.0.0.0/0		(default)

If you want to be extra secure, you can set up the SSH line to only accept your IP address - this will stop the waves of attempts from bots aimlessly trying to log in with default credentials. You'll need to keep it updated with your IP, though, or you'll lose access!

Do the same under IPv6 Rules. Finally, under Linked Instances, add your current VPS.

## Update

Now that we've got those bits out of the way, let's make sure your OS is up to date. Run the following command:

```
sudo yum update -y
```

It should automatically download and update all your OS packages. You may want to restart afterwards, just to make sure all the updates are in effect:

```
sudo shutdown -r now
```

Relatively easy!

These are only a few of the things you can do to increase the security of your system - this is only scratching the surface. You can write an entire book with all the ways you can further secure a system - such as installing and configuring something like fail2ban to monitor failed logins, auditing applications to monitor operating system changes, tweaking kernel settings - although, each security improvement also comes with its own caveats and associated difficulties. Consider what makes sense for your system and your risk level.

Once you've got all that out of the way, it's time for the moment we've all been waiting for - installing GoToSocial!

At this point, if you're planning on installing Akkoma instead, you may want to refer to something such as the Akkoma documentation for continuing. A guide for Akkoma will be made in the future.

# Installing GoToSocial

This page makes the following assumptions:

- You have a running VPS, prepared according to the instructions in [Preparing Your VPS](#).
- You have a domain name that's pointed to your VPS.

If either of these aren't true, please read through the last three pages.

We are now going to download, install, and configure GoToSocial on your system. This is what we've been working towards!

For the ease of simplicity, this guide will go through the simplest possible install of GoToSocial - this means having GoToSocial handle web traffic, using SQLite for your database, and not installing a frontend yourself. Alternate versions of this guide going through more complicated installs may be created in the future - for now, check out the GoToSocial documentation for instructions on how to set these up.

This guide is written based off, and uses a lot of the same commands, as the upstream documentation: [GoToSocial Installation - Bare metal](#)

You can also view [a video setup guide](#). It was recorded in early september, and may not agree with this written guide on some specifics. Double check the "Configuring GoToSocial" section for up to date configuration recommendations. Also see "Making SELinux Happy" if you are on Rocky Linux.

Make some directories for GoToSocial to live in:

```
sudo mkdir -p /gotosocial/storage/certs
cd /gotosocial
```

Next, download and extract the Website League GoToSocial fork:

Check [the releases page](#) for the latest version. The link below may point to an older release!

```
sudo wget https://gitlab.com/website-league/gotosocial/-/releases/v0.17.0-wl0/downloads/gotosocial_0.17.0-wl0_linux_amd64.tar.gz
sudo tar -xzf ./gotosocial_0.17.0-wl0_linux_amd64.tar.gz
```

Then, make a copy of the example config.yaml and open it up. It's time to configure your new instance!

```
sudo cp example/config.yaml ./
sudo nano config.yaml
```

## Configuring GoToSocial

There's going to be a lot of settings! You're probably going to want to look through them all, but below is some of them you're probably going to want to change/adjust from default:

```
# host: Set this to your domain name.
host: "my-awesome-website.instance"

# Running on allowlist is required for Website League nodes.
instance-federation-mode: "allowlist"

# Make clients that demand mastodon happy
instance-inject-mastodon-version: true

# instance-languages: Set this to you(r users') preferred languages!
# https://en.wikipedia.org/wiki/IETF_language_tag#List_of_common_primary_language_subtags
instance-languages: ["en"]

# If this is a single user instance, you can change this to your user
# and it'll go to your profile when browsing to your instance domain.
landing-page-user: ""

# Enable this if you're making a multi-user node and you're ready to accept registrations.
accounts-registration-open: false

# If you're from Cohost, you probably want this. Not as good as federated CSS crimes,
# but hey, you can at least have custom CSS on your profile!
accounts-allow-custom-css: true

# Adjust that character limit, make it something a little less cramped.
# (50k characters appeared to cause no problems in testing.)
```

```
statuses-max-chars: 50000
```

```
# Set up GoToSocial to run as the webserver  
# and set up SQLite as our database of choice.
```

```
port: 443
```

```
db-type: "sqlite"
```

```
db-address: "sqlite.db"
```

```
# This should be correct, but change it if you're using a different directory.
```

```
storage-local-base-path: "/gotosocial/storage"
```

```
# Set up GoToSocial to get a free SSL cert from Let's Encrypt
```

```
letsencrypt-enabled: true
```

```
letsencrypt-cert-dir: "/gotosocial/storage/certs"
```

```
letsencrypt-email-address: "your@awesome-email.here"
```

It's probably best to take some time to look through the entire configuration file, though, and see if there's anything you want to change.

Configurations that have different ``host`` and ``account-domain`` configuration values can introduce problems with federation later that can be tricky to diagnose, so it's easiest to keep them the exact same. (or leave ``account-domain`` blank)

## Smoke Test

Let's see if it works!

Run the following command:

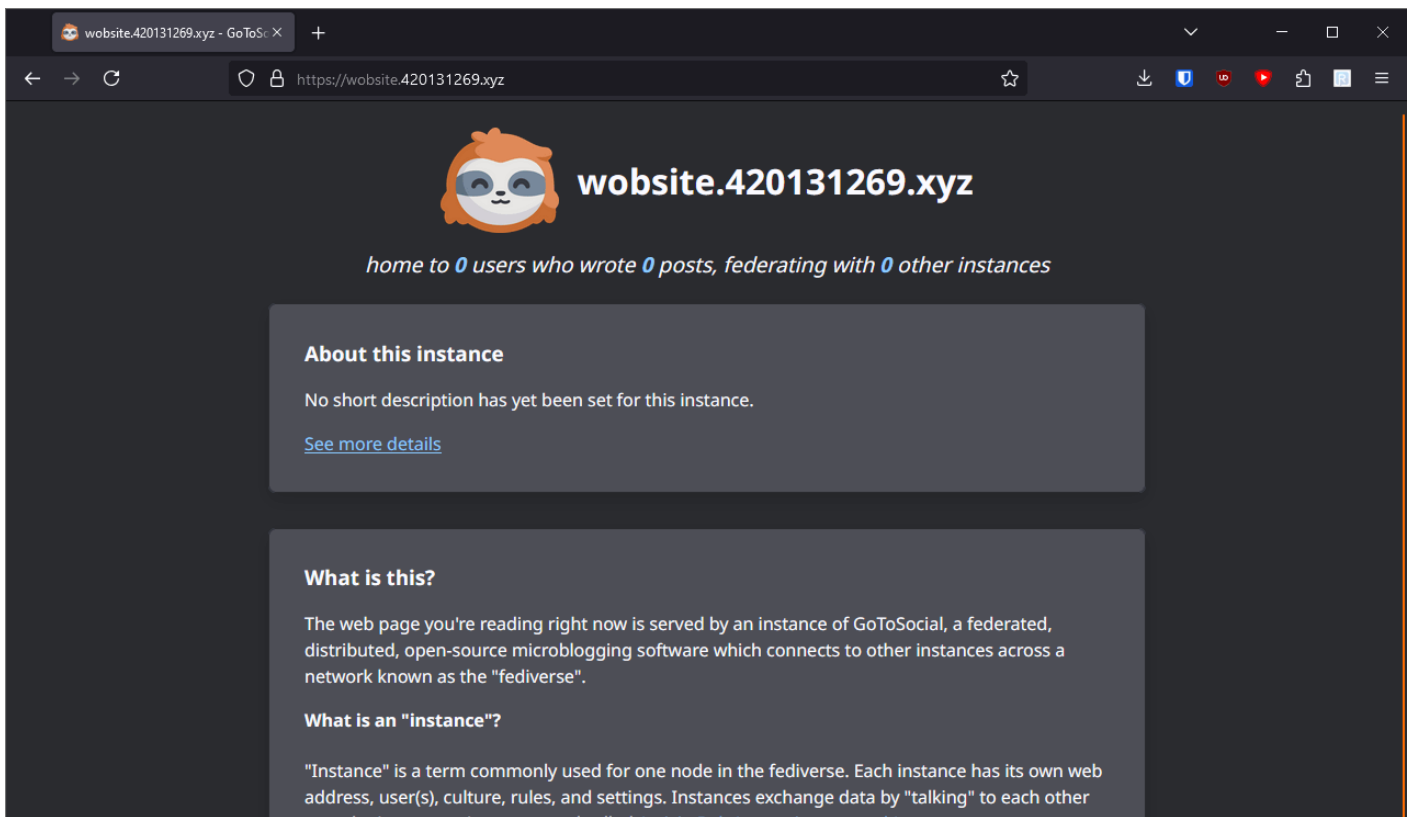
```
sudo ./gotosocial --config-path ./config.yaml server start
```

Your screen should fill with lines, then you should see something like the following near or at the bottom:

```
timestamp="26/09/2024 04:39:03.389" func=router.(*Router).letsEncryptTLS.func2 level=INFO  
msg="letsencrypt listening on 0.0.0.0:80"  
timestamp="26/09/2024 04:39:03.390" func=router.(*Router).Start.func1 level=INFO msg="listening on  
0.0.0.0:443"
```

Go to your browser, type in your instance domain, and see if it loads!





If it loads, congratulations - you've got yourself a brand new fediverse instance! If not, there's a variety of things you can check. It may need just a moment to get itself an SSL certificate, for one. You may also want to double-check your DNS configuration, as well as the rest of your GoToSocial configuration. Remember - if you need help, there are folks that're willing and able to provide.

## Run GoToSocial as a Service

Now that we know it can run, it's time to run it in a way that *doesn't* require you to be logged in to your console forever. Press Ctrl+C in the console to stop it running for now (it'll be back very soon!)

First, let's create a user like we did earlier, but for GoToSocial, then make it own the /gotosocial folder we made:

```
sudo useradd -r gotosocial
sudo groupadd gotosocial
sudo usermod -a -G gotosocial gotosocial
sudo chown -R gotosocial:gotosocial /gotosocial
```

Don't worry about creating a password for it - this is a system account. Nobody should be logging in as this user.

Now, copy the systemd service to the correct folder, and open it up in a text editor:

```
sudo cp /gotosocial/example/gotosocial.service /etc/systemd/system/  
sudo nano /etc/systemd/system/gotosocial.service
```

If you followed the instructions as given here, you'll only need to uncomment a single line:

```
# You might need this if you are running as non-root on a privileged port (below 1024)  
AmbientCapabilities=CAP_NET_BIND_SERVICE
```

It should look just like that - no # in front of it.

## Making SELinux Happy

If you try to run the service right now:

```
sudo systemctl enable --now gotosocial.service
```

you'll notice it fails right away with an odd error.

```
Sep 26 05:22:34 vultr.guest systemd[1]: Starting GoToSocial Server...  
Sep 26 05:22:34 vultr.guest systemd[4941]: gotosocial.service: Failed to locate executable  
/gotosocial/gotosocial: Permission denied  
Sep 26 05:22:34 vultr.guest systemd[4941]: gotosocial.service: Failed at step EXEC spawning  
/gotosocial/gotosocial: Permission denied  
Sep 26 05:22:34 vultr.guest systemd[1]: gotosocial.service: Main process exited, code=exited, status=203/EXEC  
Sep 26 05:22:34 vultr.guest systemd[1]: gotosocial.service: Failed with result 'exit-code'.  
Sep 26 05:22:34 vultr.guest systemd[1]: Failed to start GoToSocial Server.
```

This is because the permissions don't quite fit with SELinux, and so SELinux has decided to deny GoToSocial from starting.

SELinux is something that runs in Rocky Linux and other Linux distros like it. Its purpose is to put limits on things like what resources applications are allowed to access. If its permissions aren't properly set, it can cause issues like the ones we're seeing here.

We can solve this in one of two ways: the easy way, or the secure way. (You're probably perfectly fine just doing it the easy way, but I prepared the secure way in case you want it!) Follow the instructions according to which path you're comfortable taking:

## The Easy Way

We set SELinux into permissive mode - effectively, make it not really matter what SELinux thinks.

```
sudo setenforce 0
```

This sets it temporarily into permissive mode, so it takes effect right away. We then edit a line in `/etc/selinux/config` to make this permanent so it continues to persist after restarts:

```
# change
SELINUX=enforcing
# to
SELINUX=permissive
```

You can also set it to "disabled" in this file if you want to completely disable SELinux. Either way, it should not hinder starting GoToSocial anymore. Skip "The Secure Way" and continue onwards to "Creating a User".

## The Secure Way

This might be a bit of a headache, but it'll be more secure - we basically import an SELinux policy for GoToSocial, and patch it over time if GoToSocial ever needs new permissions. Doing this has the benefit of not having to disable SELinux.

Use git to clone a fork of [this selinux policy](https://github.com/tenna/gotosocial-selinux) modified to work with our setup:

```
git clone https://git.tenna.zip/tenna/gotosocial-selinux
```

Compile it and run the included script to re-label some files:

```
cd gotosocial-selinux
sudo make -f /usr/share/selinux/devel/Makefile load
sudo ./gotosocial-selinux-relabel
```

Run gotosocial again, and see what happens!

```
sudo systemctl start gotosocial
```

If it works, great! If SELinux doesn't allow it to do something for some reason, get SELinux to give you a config that would fix that problem:

```
sudo sepolgen-ifgen
sudo audit2allow -larR
```

It'll spit out some code - add this to the selinux files where relevant. Re-compile it and restart GoToSocial.

If you're having trouble with this, feel free to ask about it in the [Discord](#). There is, also, of course, the option of just disabling SELinux if worst comes to worst, but it's good to keep it enabled if at all possible!

## Creating a User

You're now running GoToSocial as a bonafide service! ...Except you don't have an account. Let's fix that.

Run this command, then enter the password you want and hit Enter again:

```
read -r THEPASSWORD
```

Then, run the following commands, replacing `your_username_here` and [your@awesome.email](#) with your username and E-mail (don't replace THEPASSWORD - this will get substituted automatically with the password you just entered):

```
cd /gotosocial/  
sudo -u gotosocial ./gotosocial --config-path ./config.yaml \  
  admin account create \  
  --username your_username_here \  
  --email your@awesome.email \  
  --password ${THEPASSWORD}  
sudo -u gotosocial ./gotosocial --config-path ./config.yaml \  
  admin account promote --username your_username_here
```

Finally, restart GoToSocial. You've got your own user now! Now you can use a client like [Semaphore](#) to log in and make posts!

...To nobody, yet. You've gotta add the other Website League nodes to your allowlist. This is fairly simple, though!

## Managing the Allowlist

As of the writing of this guide, there is currently no automated way to manage the allowlist for the Website League. One is in the works - once it is finished, this guide will be updated to include the installation of such automation. In the interim, this will go over how to manually add nodes to your allowlist.

Go to the following URL, replacing "your-cool.instance" with your instance domain:

```
https://your-cool.instance/settings
```

Log in to the account you just made. Then, go to Domain Permissions > Allows. Here, you can then type the name of an instance, then hit Allow, then Allow again.

Alternatively, if you have a list of domains, you can go to Domain Permissions > Import/Export, paste the list, tick the "Domain allows" box, then import.

You're done! If you've made it all the way to the end here - congrats on your new instance, and thanks for reading! Hope it helped. If you have any questions or need any help, there's folks in the Discord who're willing to help, including the author of this guide, Tenna Lotor.

# Quick Reference for common admin tasks

Some administrative actions currently [require using the gotosocial CLI](#) and aren't available in the settings ui.

To use the gotosocial CLI, you need to somehow pass your config to it. These examples use `--config-path`.

## Confirming a user's account

You may need to do this to activate a user if they weren't able to receive an email from the node. There doesn't seem to be an obvious way to re-send the email.

```
cd /gotosocial/  
sudo -u gotosocial ./gotosocial --config-path ./config.yaml \  
  admin account confirm \  
  --username username_to_confirm
```