Mastodon Setup

Howdy, stranger! This document is the other half of this video, in which I set up a single-server instance of Mastodon. This was assembled on 9 April 2017, and there's a good chance that some of the specifics here will change over time. I'll keep an updated version up on wogan.blog.

0. Pre-Prerequisites

At a bare minimum, you're going to need:

- A domain name, with the ability to add an A record yourself
- A free mailgun.com account, with the account verified and your sandbox enabled to send to you
- A 1GB RAM machine with decent network access. This document uses a DigitalOcean VM.

This setup procedure skips a few things that you may want to do on a "productionized" or "community" instance of Mastodon, such as configuring S3 file storage, or using a non-sandbox email send account. You may also want a beefier machine than just 1GB RAM.

For reference, the OS version in use is `Ubuntu 16.04.2 LTS` and all the commands are being run from the `root` user unless explicitly specified.

1. Getting started!

The first couple steps:

- Create the VM
- Point your domain to it immediately, by setting the A record to the public IP
- Log into the VM
- Set your root password
- Create a new Mastodon user: `adduser mastodon`
- Update the apt cache: `apt-get update`

2. Install Prerequisites

Now we'll grab all the prerequisite software packages in one go:

```
apt-get install imagemagick ffmpeg libpq-dev libxml2-dev
libxslt1-dev nodejs file git curl redis-server redis-tool
s postgresql postgresql-contrib autoconf bison build-esse
ntial libssl-dev libyaml-dev libreadline6-dev zlib1g-dev
libncurses5-dev libffi-dev libgdbm3 libgdbm-dev git-core
letsencrypt nginx
```

That'll take a little while to run. When it's done, you'll need Node (version 4) and yarn:

```
curl -sL https://deb.nodesource.com/setup_4.x | bash -
```

```
apt-get install nodejs
npm install -g yarn
```

You'll also want to be sure that redis is running, so do:

```
service redis-server start
```

3. Configure Database

With Postgres installed, you need to create a new user. Drop into the postgres user and create a mastodon account:

```
su - postgres
psql
CREATE USER mastodon CREATEDB;
\q
exit
```

Later on we'll configure mastodon to use that.

4. Generate SSL certificate

Before configuring nginx, we can generate the files we'll need to support SSL. First, kill nginx:

```
service nginx stop
```

Now proceed through the LetsEncrypt process:

- Run `letsencrypt certonly`
- Enter your email address
- Read and acknowledge the terms
- Enter the domain name you chose

If the domain name has propagated (which is why it's important to do this early), LetsEncrypt will find your server and issue the certificate in one go. If this step fails, you may need to wait a while longer for your domain to propagate so that LetsEncrypt can see it.

5. Configure nginx

With the SSL cert done, time to configure nginx!

```
cd /etc/nginx/sites-available
nano mastodon
```

Simply substitute your domain name where it says `example.com` in this snippet (lines 9, 15, 23, 24), then paste the entire thing into the file and save it.

```
map $http_upgrade $connection_upgrade {
  default upgrade;
  ''      close;
}
```

```
server {
  listen 80;
  listen [::]:80;
  server_name example.com;
  return 301 https://$host$request_uri;
}

server {
  listen 443 ssl;
  server_name example.com;

  ssl_protocols TLSv1.2;
  ssl_ciphers EECDH+AESGCM:EECDH+AES;
  ssl_ecdh_curve prime256v1;
  ssl_prefer_server_ciphers on;
  ssl_session_cache shared:SSL:10m;

  ssl_certificate     /etc/letsencrypt/live/example.com/f
ullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/example.com/p
rivkey.pem;

  keepalive_timeout    70;
  sendfile             on;
  client_max_body_size 0;
  gzip off;
```

```
  root /home/mastodon/live/public;

  add_header Strict-Transport-Security "max-age=31536000;
 includeSubDomains";

  location / {
    try_files $uri @proxy;
  }

  location @proxy {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forward
ed_for;
    proxy_set_header X-Forwarded-Proto https;

    proxy_pass_header Server;

    proxy_pass http://localhost:3000;
    proxy_buffering off;
    proxy_redirect off;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;

    tcp_nodelay on;
  }
```

```
  location /api/v1/streaming {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forward
ed_for;
    proxy_set_header X-Forwarded-Proto https;

    proxy_pass http://localhost:4000;
    proxy_buffering off;
    proxy_redirect off;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;

    tcp_nodelay on;
  }

  error_page 500 501 502 503 504 /500.html;
}
```

Once you've saved and closed the file, enable it by creating a symlink:

```
ln -s /etc/nginx/sites-available/mastodon /etc/nginx/site
s-enabled/mastodon
```

Then test that the file is OK by running `nginx -t`. If it reports any errors, you'll want to fix them before moving on. If the file comes back

OK, fire it up!

```
service nginx start
```

Open a browser tab and navigate to your domain. You should get a 502 Gateway Error, secured with your LetsEncrypt cert. If not, go back and make sure you've followed every preceding step correctly.

6. Configure Systemd

Mastodon consists of 3 services (web, sidekiq and streaming), and we need to create config files for each. You can use the code straight from this page, as-is.

```
cd /etc/systemd/system/
```

The first file is called `mastodon-web.service` and consists of the following:

```
[Unit]
Description=mastodon-web
After=network.target

[Service]
Type=simple
User=mastodon
WorkingDirectory=/home/mastodon/live
```

```
Environment="RAILS_ENV=production"
Environment="PORT=3000"
ExecStart=/home/mastodon/.rbenv/shims/bundle exec puma -C
 config/puma.rb
TimeoutSec=15
Restart=always

[Install]
WantedBy=multi-user.target
```

The next file is called `mastodon-sidekiq.service` and consists of the following:

```
[Unit]
Description=mastodon-sidekiq
After=network.target

[Service]
Type=simple
User=mastodon
WorkingDirectory=/home/mastodon/live
Environment="RAILS_ENV=production"
Environment="DB_POOL=5"
ExecStart=/home/mastodon/.rbenv/shims/bundle exec sidekiq
 -c 5 -q default -q mailers -q pull -q push
TimeoutSec=15
Restart=always
```

```
[Install]
WantedBy=multi-user.target
```

The final file is called `mastodon-streaming.service` and consists of the following:

```
[Unit]
Description=mastodon-streaming
After=network.target

[Service]
Type=simple
User=mastodon
WorkingDirectory=/home/mastodon/live
Environment="NODE_ENV=production"
Environment="PORT=4000"
ExecStart=/usr/bin/npm run start
TimeoutSec=15
Restart=always

[Install]
WantedBy=multi-user.target
```

Once all those are saved, we've done all we can with the root user for now.

7. Switch to the Mastodon user

If you haven't yet logged into the server as `mastodon`, do so now in a second SSH window. We're going to set up ruby and pull down the actual Mastodon code here.

8. Install rbenv, rbenv-build and Ruby

As the `mastodon` user, clone the rbenv repo into your home folder:

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

When that's done, link the bin folder to your PATH:

```
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bash_pr
ofile
```

Then add the init script to your profile:

```
echo 'eval "$(rbenv init -)"' >> ~/.bash_profile
```

That line is valid for the OS we're on (Ubuntu 16.04 LTS) but it may differ slightly for you. You can run `~/.rbenv/bin/rbenv init` to check what line you need to use.

Once you've saved that, log out of the mastodon user, then log back in to complete the rest of this section.

Install the ruby-build plugin like so:

```
git clone https://github.com/rbenv/ruby-build.git ~/.rbenv/plugins/ruby-build
```

Then install Ruby proper:

```
rbenv install 2.3.1
```

This could take up to 15 minutes to run!

When it's done, change to your home folder and clone the Mastodon source:

```
cd ~
git clone https://github.com/tootsuite/mastodon.git live
cd live
```

Next up, dependencies! Always more dependencies - we'll install bundler, then use that to install everything else:

```
gem install bundler
bundle install --deployment --without development test
```

```
yarn install
```

If all of those succeeded, we're ready to configure!

9. Configure Mastodon

Before diving into the configuration file, generate 3 secret strings by running this command 3 times:

```
bundle exec rake secret
```

Copy those out to a text file - you'll paste them back in later. Create the config file by copying the template, then editing it with nano:

```
cp .env.production.sample .env.production
nano .env.production
```

Inside this file we're going to make several quick changes.

```
REDIS_HOST=localhost
DB_HOST=/var/run/postgresql
DB_USER=mastodon
DB_NAME=mastodon_production
```

To enable federation, you need to set your domain name here:

```
LOCAL_DOMAIN=example.com
```

Then, for these 3, paste in each key you generated earlier:

```
PAPERCLIP_SECRET=
SECRET_KEY_BASE=
OTP_SECRET=
```

Finally, configure your SMTP details:

```
SMTP_LOGIN= (whatever your mailgun is)
SMTP_PASSWORD= (whatever your mailgun is)
```

Save and close the file.

10. Run installer

If you've done everything correctly, this command will install the database:

```
RAILS_ENV=production bundle exec rails db:setup
```

If that passes successfully (it'll echo out every command it runs), you can then precompile the site assets, which may take a few minutes:

```
RAILS_ENV=production bundle exec rails assets:precompile
```

At this point, we're almost ready to go!

11. Configure cronjob

This is technically optional, but highly recommended to keep your instance in good order. As the `mastodon` user, start by determining where your bundle command lives:

```
which bundle
```

That path will be substituted for `$bundle`. Now, edit your own crontab:

```
crontab -e
```

Select nano (2) if you're prompted. Paste in the following lines, making sure to substitute `$bundle` for the path you got with `which bundle`:

```
5 0 * * * RAILS_ENV=production $bundle exec rake mastodon
:media:clear
10 0 * * * RAILS_ENV=production $bundle exec rake mastodo
n:push:refresh
15 0 * * * RAILS_ENV=production $bundle exec rake mastodo
n:feeds:clear
```

Those commands will run at 00:05, 00:10 and 00:15 respecively. Save and close the crontab.

12. Log out and return to root

We're done with the `mastodon` account. Log out and return to your `root` shell.

13. Start Mastodon

The moment of truth! Enable the Mastodon services (so that they start on boot):

```
systemctl enable /etc/systemd/system/mastodon-*.service
```

Then fire up Mastodon itself:

```
systemctl start mastodon-web.service mastodon-sidekiq.ser
vice mastodon-streaming.service
```

Open up a browser tab on your domain. Mastodon can take up to 30 seconds to warm up, so if you see an error page, don't fret. Only fret if it's there for longer than a minute - that requires troubleshooting, which is outside the scope of this document.

You should eventually get a signup page. Congratulations! Register an

account for yourself, receive the confirmation email, and activate it.

14. Securing Mastodon

This is by no means a comprehensive guide to server security, but there are two quick things you can change while the root shell is open. Start by editing the passwd file:

```
nano /etc/passwd
```

Find the `mastodon` entry (it'll be near the bottom) and replace `/bin/bash` with `/usr/sbin/nologin`. Save and quit. This will prevent anyone from logging in as the mastodon user.

Next, configure ufw. First check if it's disabled:

```
ufw status
```

It should be off, since this is a brand new VM. Configure it to allow SSH (port 22) and HTTPS (port 443), then turn it on:

```
ufw allow 22
ufw allow 443
ufw enable
y
```

That will prevent any connection attempts on other ports.

15. Enjoy!

If you enjoyed this guide, I'd appreciate a follow! You can find me by searching `wogan@wogan.im` in your Mastodon web UI. Give me a shout if you were able to get an instance set up with these instructions, or if you ran into any problems.

Sources

A lot of this guide was sourced from the official Production guide on the Mastodon Github page. I reorded it into a logical sequence after running through it for a few tries.