How to Install Gitea Code Hosting Platform with HTTPS on Debian 10

Gitea is a code hosting web application written in Go and forked from Gogs. As its name suggests, it is designed to be used with the popular source control program Git, similarly to Gitlab and Github. This guide will explain how to install Gitea on Debian 10 behind an HTTPS reverse proxy (Nginx).

Requirements

- A Debian 10 system on which you have root privileges.
- A registered domain name pointing to your server.
- The $EDITOR environment variable should be set.
- Access to an SMTP server for email notifications (optional).

Step 1: Preparing the system

Start by updating your package index and install any available updates:

```
apt update
apt upgrade -y
reboot
```

For this setup, several software packages are required:

- Git, a dependency of Gitea.
- PostgreSQL, as Gitea requires a database.
- Nginx, which will be used as a reverse proxy.
- Certbot, a utility for obtaining Let's Encrypt SSL certificates.
- Sudo, to run commands as the postgres system user.

Install them as follows:

```
apt install -y git nginx certbot postgresql sudo
```

Next, create a user to run Gitea:

```
adduser --system --shell /bin/bash --home /home/gitea gitea
```

Then create the directory structure for Gitea:

```
mkdir -p /var/lib/gitea/{data,log} /etc/gitea /run/gitea
```

And set ownerships and permissions as follows:

```
chown -R gitea:gitea /var/lib/gitea
chown -R gitea:gitea /run/gitea
chown -R root:gitea /etc/gitea
chmod -R 750 /var/lib/gitea
chmod 770 /etc/gitea
```

The permissions on /etc/gitea are temporary and will be tightened after running the web installer.

Step 2: Database Setup

Make sure Postgres is enabled and running:

```
systemctl enable --now postgresql@11-main.service
```

Then create a user role and database to be used by Gitea:

```
sudo -u postgres psql
postgres=# CREATE ROLE gitea LOGIN ENCRYPTED PASSWORD 'your_password';
postgres=# CREATE DATABASE gitea;
postgres=# GRANT ALL PRIVILEGES ON DATABASE gitea TO gitea;
postgres=# exit;
```

Step 3: Installing Gitea

Download the latest linux-amd64 binary from Gitea's download page. For example:

```
wget https://dl.gitea.io/gitea/master/gitea-master-linux-amd64 -O
/usr/local/bin/gitea
chmod 755 /usr/local/bin/gitea
```

Next, create a systemd unit file for Gitea:

```
$EDITOR /etc/systemd/system/gitea.service
```

And enter the following:

```
[Unit]
Description=Gitea (Git with a cup of tea)
After=syslog.target
After=network.target
Requires=postgresql.service
[Service]
Type=simple
User=gitea
Group=gitea
WorkingDirectory=/var/lib/gitea/
```

```
RuntimeDirectory=gitea
ExecStart=/usr/local/bin/gitea web -c /etc/gitea/app.ini
Restart=always
Environment=USER=gitea HOME=/home/gitea GITEA_WORK_DIR=/var/lib/gitea
[Install]
WantedBy=multi-user.target
```

Make sure the new unit is loaded:

```
systemctl daemon-reload
```

Then instruct systemd to start Gitea at system startup:

```
systemctl enable gitea.service
```

Step 4: Configuring Gitea

For the initial configuration, we'll use the included web install script. First, start Gitea:

```
systemctl start gitea.service
```

Then navigate to http://your_domain:3000/install and fill in the required parameters as follows:

- Database Type: PostgreSQL
- Host: 127.0.0.1:5432
- Username: gitea
- Password: Enter the password you chose during Postgres role creation.
- Database Name: gitea
- SSL: Disable
- Site Title: Title of your choice.
- Repository Root Path: /var/lib/gitea/data/repositories
- Git LFS Root Path: /var/lib/gitea/data/lfs
- Run As Username: gitea
- SSH Server Domain: your_domain
- SSH Server Port: 22
- Gitea HTTP Listen Post: 3000
- Gitea Base URL: https://your_domain/
- Log Path: /var/lib/gitea/log

Configure email and the remaining settings as deemed fit, then click „Install Gitea“. You will be redirected to a faulty URL. This is normal, as we haven't configured Nginx or HTTPS yet. For performance reasons, we will now configure Gitea to listen on a unix socket instead of the default TCP port.

Stop Gitea before proceeding:

```
systemctl stop gitea.service
```

Tighten permissions on /etc/gitea as shown below. This prevents anyone not in the gitea group from

reading app.ini, which contains sensitive information, including database credentials.

```
chmod 750 /etc/gitea
chown root:gitea /etc/gitea/app.ini
chmod 640 /etc/gitea/app.ini
```

Open its configuration file:

```
$EDITOR /etc/gitea/app.ini
```

Remove the following line from the [server] section:

```
HTTP_PORT = 3000
```

And add the following lines to the [server] section:

```
HTTP_ADDR        = /run/gitea/gitea.sock
PROTOCOL         = unix
UNIX_SOCKET_PERMISSION = 666
```

Step 5: Setting Up the Reverse Proxy

Stop Nginx if it is running, as certbot will need to bind to port 80:

```
systemctl stop nginx.service
```

Use the following command to obtain a certificate for your domain:

```
certbot certonly --standalone --agree-tos -m your_email@example.com -d
your_domain
```

Let's Encrypt will verify domain ownership before issuing the certificate. Your certificate, chain, and private key will be stored in /etc/letsencrypt/live/your_domain/.

We can now configure Nginx. Create a new configuration file:

```
$EDITOR /etc/nginx/sites-available/gitea
```

And enter the following configuration:

```
server {
        listen 80;
        listen [::]:80;
        server_name your_domain;
        return 301 https://$server_name$request_uri;
    access_log /var/log/nginx/gitea-proxy_access.log;
    error_log /var/log/nginx/gitea-proxy_error.log;
}
```

```
server {
        listen 443 ssl;
        listen [::]:443 ssl;
        server_name your_domain;
        ssl on;
        ssl_certificate /etc/letsencrypt/live/your_domain/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/your_domain/privkey.pem;
        location / {
                proxy_pass http://unix:/var/run/gitea/gitea.sock;
    }
    access_log /var/log/nginx/gitea-proxy_access.log;
    error_log /var/log/nginx/gitea-proxy_error.log;
}
```

The first server block simply serves to redirect all HTTP requests to HTTPS. The second block listens for HTTPS connections and proxies them to the unix socket on which we configured Gitea to listen.

Once you've saved the above configuration, run the following to enable it:

```
ln -s /etc/nginx/sites-available/gitea /etc/nginx/sites-enabled
```

Check for any syntax errors with and edit your configuration accordingly:

```
nginx -t
```

Finally, start Nginx and Gitea:

```
systemctl start nginx.service gitea.service
```

Your Gitea instance should now be running successfully. If you did not create an administrator account using the initial web installer, the first user to sign up will be given the administrator role.

Optional Steps

Logging Configuration

By default, Gitea logs messages of severity level Info and above. You will most likely want to change that to Warn or Error. To do so, open /etc/gitea/app.ini and change the LEVEL parameter in the [log] section to one of: trace, debug, info, warn, error, critical, fatal, none. For example, to log messages of severity Warn and above, use:

```
[log]
MODE = file
LEVEL = warn
ROOT_PATH = /var/lib/gitea/log
```

Restart Gitea for the changes to take effect:

```
systemctl restart gitea.service
```

Separate SSH server

Gitea can alternatively use its own SSH server. To enable it, add the following line to the [server] configuration section:

```
START_SSH_SERVER = true
```

And change the SSH port to any number above 1000, for instance:

```
SSH_PORT = 2222
```

Then restart Gitea to apply the changes.