Gitea

This page doesn't intend to replace or improve existing documentation, but rather provides me with material to look back on in the future if I ever have to revisit this setup again. The information here may or may not help setting up your gitea, but in either case I strongly recommend checking out the following links to learn more about Gitea.

Gitea official GitHub repository

Gitea official documentation

While learning Unreal Engine 5, I began exploring options for using Git LFS without breaking the bank or sacrificing content in my game to save space. I quickly realized this was going to become an issue, when GitHub alerted me that I hit the 1GB limit for LFS repositories. Honestly I'm not entirely sure why GitHub even offers LFS, if you're reaching for LFS chances are you'll hit that 1GB limit in a very short timeframe, if you hadn't already exceeded the limit before initializing LFS.

GitLab was a better option for a while, they offer 10GB free storage for LFS repositories, which was plenty to get started with UE5. Eventually I hit the 10GB mark, and explored pricing options for increasing storage.

GitLab charges $60 annually for 10GB, and I just can't reason with that price for a simple UE5 project with no need for CI / DevOps or any of the other features GitLab probably factored into their pricing.

So I realized I had to host my own Git service, and that's where Gitea comes in. The service outlined in this page offers me 25GB of LFS storage for $6 a month from Digital Ocean, and if I choose to upgrade that storage I pay $10/month for an additional 100GB. By doing this, I avoid running into a problem again in a few months, since I can easily upgrade this storage for a fraction of the cost when compared to LFS on GitLab.

With this setup, I pay $60 per year for 25GB, or $120 per year for 125GB of LFS storage. I like those numbers much better. At GitLab's price, this same setup would cost almost $800 per year to run with 125GB of LFS storage. I don't anticipate ever being able to fill this 125GB storage capacity, so this is the wost-case scenario, and I think $120 annually (10/mo) is a reasonable price to pay for this convenience

I'll admit I did not look at GitHub's pricing because I was initially frustrated with the 1GB limit for LFS repositories. It's unusable right from the start, and I'm not interested in shelling out hundreds of dollars to risk another road block like this in the future.

Server Setup

To start this process I purchased the cheapest VPS possible from DigitalOcean which costs only $6 a month. The server came with no configurations or services installed - I always select the latest barebones LTS Ubuntu version offered by DigitalOcean.

Once the droplet is created, login as the root user and create a user to run the service.

```
sudo adduser docker-gitea
sudo usermod -aG sudo docker-gitea
```

Now we go to the home directory of the user and set up the service within a subdirectory

```
sudo su docker-gitea
cd
mkdir docker-gitea
cd docker-gitea
vim docker-compose.yml
```

That's it! The next section will cover the contents of this new `docker-compose.yml` file, and how to start and stop the service.

Docker

To configure the service, I chose to use a docker container defined by a `docker-compose.yml` with the contents below. See the official [Gitea Configuration Cheatsheet](#) for details on what these options mean, if you're curious.

```yaml
version: "3"

networks:
  gitea:
    external: false

services:
  server:
    image: gitea/gitea:latest
    container_name: gitea
    environment:
      - USER_UID=1000
      - USER_GID=1000
      - GITEA__database__DB_TYPE=mysql
      - GITEA__database__HOST=db:3306
      - GITEA__database__NAME=gitea
```

```yaml
      - GITEA__database__USER=gitea
      - GITEA__database__PASSWD=somepassword
    restart: always
    networks:
      - gitea
    volumes:
      - ./gitea:/data
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "2000:3000"
      - "222:22"
    depends_on:
      - db

  db:
    image: mysql:8
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=someotherpasswordforrootuser
      - MYSQL_USER=gitea
      - MYSQL_PASSWORD=somepassword
      - MYSQL_DATABASE=gitea
    networks:
      - gitea
    volumes:
      - ./mysql:/var/lib/mysql
```

Once you've looked over these configurations carefully, you can run the following commands to start the service from within the `/home/docker-gitea/docker-gitea/` directory.

> If you get Permission Denied errors, make sure to add your user to the `docker` group by running `sudo usermod -aG docker docker-gitea` where `docker-gitea` should be replaced with the username you selected for running your service

```
docker-compose up -d
```

You can now access your gitea service by visiting your server's IP and manually providing the port you specified in the `docker-compose.yml`. For the configuration above, we use `2000:3000` to route HTTP traffic, so we can visit `<YOUR_SERVER_IP>:2000` and we are greeted with the gitea landing page!

To stop the service you can run this command, but the following sections will expect that the docker container is running and the service is up, so make sure it is before continuing.

```
docker-compose down
```

NGINX Configuration+

First, head over to **Knoats - NGINX** to see how to generate your SSL certificate, and maybe skim through some of the notes there on NGINX if needed. Don't let SSL setup intimidate you, it is not a difficult process and only takes a few extra minutes to setup. SSL is a very important security feature!

Once you have generated your SSL certificates, the only thing left to do on the back-end is route traffic and configure Gitea. For the `docker-compose.yml` outlined above, we should note the following lines that contain ports we will need.

```
  ports:
    - "2000:3000"
    - "222:22"
```

With this information in mind, we can run `sudoedit /etc/nginx/nginx.conf` and configure your NGINX server to route traffic to your docker container.

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;

events { }

http {
  include mime.types;

  # Basic Server Configuration
  server {
    # NGINX listens on port 80
    listen 80;
    server_name git.shaunreed.com;
    return 301 https://$host$request_uri;
  }

  # Terminate SSL and route traffic
  server {
```

```
    server_name localhost;
    server_tokens off;


    # SSL Settings
    listen 443 ssl;
    # NOTE: Full path to your SSL certificates
    ssl_certificate /etc/letsencrypt/live/git.shaunreed.com/fullchain.pem;

    ssl_certificate_key /etc/letsencrypt/live/git.shaunreed.com/privkey.pem;

    include /etc/letsencrypt/options-ssl-nginx.conf;

    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;


    location / {
      include proxy_params;
      # Pass traffic to our docker container listening on port 2000
      # NOTE: Here is port 2000 from docker-compose.yml
      proxy_pass http://0.0.0.0:2000/;
    }
  }
}
```

Now run the following command to test NGINX configuration -

```
sudo nginx -t
```

If this test fails, the output should point you in the right direction, and google is your friend :)

Once the test passes, you're ready to restart the NGINX service and configure Gitea.

```
sudo systemctl restart nginx.service
```

Gitea Configuration

This section covers the various useful settings I found in Gitea for my use case. There may be other options that suit your use better, so I will again refer you to the Gitea Configuration Cheatsheet

By default when running gitea from a docker container as outline in the previous steps, your configurations will be mounted to `/home/docker-gitea/docker-gitea/gitea/gitea/conf/app.ini`

Here's some copypasta for the settings I found useful in the below configuration. The format is the same as the Configuration Cheatsheet, `SETTING_NAME: default_value: <DESCRIPTION>`

DOMAIN: localhost: Domain name of this server.

SSH_DOMAIN: %(DOMAIN)s: Domain name of this server, used for displayed clone URL.

LANDING_PAGE: home: Landing page for unauthenticated users [home, explore, organizations, login].

REGISTER_EMAIL_CONFIRM: false: Enable this to ask for mail confirmation of registration. Requires Mailer to be enabled.

REGISTER_MANUAL_CONFIRM: false: Enable this to manually confirm new registrations. Requires REGISTER_EMAIL_CONFIRM to be disabled. (I didn't use it, but it's neat)

DISABLE_REGISTRATION: false: Disable registration, after which only admin can create accounts for users.

ENABLE_NOTIFY_MAIL: false: Enable this to send e-mail to watchers of a repository when something happens, like creating issues. Requires Mailer to be enabled.

ENABLE_CAPTCHA: false: Enable this to use captcha validation for registration.

Here's the entire `app.ini` configuration for my gitea instance, with the private bits replaced with XX_PRIVATE_XX

```
APP_NAME = Gitea: Shaun Reed
RUN_MODE = prod
RUN_USER = git


[repository]
ROOT = /data/git/repositories


[repository.local]
LOCAL_COPY_PATH = /data/gitea/tmp/local-repo


[repository.upload]
TEMP_PATH = /data/gitea/uploads


[server]
APP_DATA_PATH    = /data/gitea
DOMAIN           = git.shaunreed.com
SSH_DOMAIN       = git.shaunreed.com
HTTP_PORT        = 3000
ROOT_URL         = https://git.shaunreed.com
DISABLE_SSH      = false
SSH_PORT         = 22
SSH_LISTEN_PORT  = 22
LFS_START_SERVER = true
LFS_CONTENT_PATH = /data/git/lfs
LFS_JWT_SECRET   = XX_PRIVATE_XX
```

```ini
LFS_HTTP_AUTH_EXPIRY = 40
OFFLINE_MODE    = false
LANDING_PAGE    = explore


[database]
PATH    = /data/gitea/gitea.db
DB_TYPE  = mysql
HOST    = db:3306
NAME    = gitea
USER    = gitea
PASSWD   = XX_PRIVATE_XX
LOG_SQL  = false
SCHEMA   =
SSL_MODE = disable
CHARSET  = utf8mb4


[indexer]
ISSUE_INDEXER_PATH = /data/gitea/indexers/issues.bleve


[session]
PROVIDER_CONFIG = /data/gitea/sessions
PROVIDER        = file


[picture]
AVATAR_UPLOAD_PATH            = /data/gitea/avatars
REPOSITORY_AVATAR_UPLOAD_PATH = /data/gitea/repo-avatars
DISABLE_GRAVATAR             = false
ENABLE_FEDERATED_AVATAR       = true


[attachment]
PATH = /data/gitea/attachments


[log]
MODE     = console
LEVEL    = info
ROUTER   = console
ROOT_PATH = /data/gitea/log


[security]
INSTALL_LOCK             = true
```

```
SECRET_KEY              =
REVERSE_PROXY_LIMIT         = 1
REVERSE_PROXY_TRUSTED_PROXIES = *
INTERNAL_TOKEN           = XX_PRIVATE_XX
PASSWORD_HASH_ALGO          = XX_PRIVATE_XX


[service]
DISABLE_REGISTRATION          = false
REQUIRE_SIGNIN_VIEW           = false
REGISTER_EMAIL_CONFIRM         = true
ENABLE_NOTIFY_MAIL           = true
ALLOW_ONLY_EXTERNAL_REGISTRATION  = false
ENABLE_CAPTCHA             = true
REQUIRE_EXTERNAL_REGISTRATION_CAPTCHA = true
DEFAULT_KEEP_EMAIL_PRIVATE       = false
DEFAULT_ALLOW_CREATE_ORGANIZATION = true
DEFAULT_ENABLE_TIMETRACKING      = true
NO_REPLY_ADDRESS            = noreply.private
REGISTER_MANUAL_CONFIRM = true


[mailer]
ENABLED      = true
FROM         = gitea@shaunreed.com
MAILER_TYPE   = smtp
HOST         = smtp.gmail.com:587
IS_TLS_ENABLED = false
USER         = mailedknoats@gmail.com
PASSWD        = `XX_PRIVATE_XX`



[openid]
ENABLE_OPENID_SIGNIN = true
ENABLE_OPENID_SIGNUP = true
```

Hopefully this page either helped you configure gitea, or helped you determine whether or not you want to deploy gitea on your server. For me it has been great, it's light weight, I like the interface, and it saves me a lot of money while offering peace-of-mind that my repositories have plenty of storage space. Good luck!

LFS Push Errors

If you're experiencing errors when pushing to an LFS project like I was, these links might be helpful.

[Gitea logging configurations](#)

[Stackoverflow gitea HTTP 413 question](#)

[HTTP 413 LFS errors (Issue #2930)](#)

[Make HTTP auth period configurable (PR #4035)](#)

This is what did it for me: [NGINX client_max_body_size](#)

Adding the `client_max_body_size` setting to our NGINX configuration would look like this

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;

events { }

http {
  include mime.types;

# Basic Server Configuration
  server {
    # NGINX listens on port 80
    listen 80;
    server_name git.shaunreed.com;
    return 301 https://$host$request_uri;
  }

  # Terminate SSL and route traffic
  server {
    server_name localhost;
    server_tokens off;

    # SSL Settings
    listen 443 ssl;
    # NOTE: Full path to your SSL certificates
    ssl_certificate /etc/letsencrypt/live/git.shaunreed.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/git.shaunreed.com/privkey.pem;
```

```
      include /etc/letsencrypt/options-ssl-nginx.conf;
      ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;


      location / {
        include proxy_params;
        # Pass traffic to our docker container listening on port 2000
        # NOTE: Here is port 2000 from docker-compose.yml
        proxy_pass http://0.0.0.0:2000/;
      }
    }
    # NOTE: In my case, adding this line fixed HTTP 413 errors when pushing to LFS projects
    client_max_body_size 3000m;
  }
```

Additionally, you should check that the `ROOT_URL` setting within your `app.ini` for gitea specifies `https://` and not `http://`

Here's my `ROOT_URL` setting and all other LFS settings within my `app.ini`. With this setup I have pushed an LFS project >10GB in a single push with no issues.

```
[server]
APP_DATA_PATH    = /data/gitea
DOMAIN           = git.shaunreed.com
SSH_DOMAIN       = git.shaunreed.com
HTTP_PORT        = 3000
ROOT_URL         = https://git.shaunreed.com
DISABLE_SSH      = false
SSH_PORT         = 22
SSH_LISTEN_PORT  = 22
LFS_START_SERVER = true
LFS_CONTENT_PATH = /data/git/lfs
LFS_JWT_SECRET   = XX_PRIVATE_XX
LFS_HTTP_AUTH_EXPIRY = 40
OFFLINE_MODE     = false
LANDING_PAGE     = explore
```

You might also consider adjusting `LFS_HTTP_AUTH_EXPIRY`, if you think you're push will take longer than 40 minutes.

If you're still struggling, you could try adding these settings to your `app.ini` to enable more verbose logging

```
[log]
MODE      = file
LEVEL     = debug
ROUTER    = ,
ROOT_PATH = /data/gitea/log
```

And then watch the logs or check them after a push. I ran the following command to watch the logs during a push that I knew would fail

```
sudo tail -f /home/docker-gittea/docker-gittea/gitea/gitea/log/gitea.log
```

SSH Configuration

To clone via ssh with gitea listening on a custom port, place the following in your `~/.ssh/config`, after editing path and url to match your instance

```
# Gitea
Host git.shaunreed.com
  HostName git.shaunreed.com
  Port <PORT_NUMBER>
  IdentityFile /home/kapper/.ssh/<PRIVATE_KEY_VAME>
```

Revision #5
Created 25 January 2022 18:56:00 by Shaun Reed
Updated 22 May 2022 13:48:24 by Shaun Reed