WIKI.JS SETUP INCLUDING POSTGRESQL AND GITEA

Wiki.js installation include postgresql

Follow walkthrough (https://docs.requarks.io/install/ubuntu)

Update Machine

```
# Fetch latest updates
sudo apt -qqy update

# Install all updates automatically
sudo DEBIAN_FRONTEND=noninteractive apt-get -qqy -o
Dpkg::Options::='--force-confdef' -o
Dpkg::Options::='--force-confold' dist-upgrade
```

Docker Installation

```
# Install dependencies to install Docker
sudo apt -qqy -o Dpkg::Options::='--force-confdef' -o
Dpkg::Options::='--force-confold' install ca-certificates curl gnupg
lsb-release

# Register Docker package registry
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Refresh package udpates and install Docker
sudo apt -qqy update
sudo apt -qqy -o Dpkg::Options::='--force-confdef' -o
Dpkg::Options::='--force-confold' install docker-ce docker-ce-cli
containerd.io docker-compose-plugin
```

Setup Containers

```
# Create installation directory for Wiki.js
mkdir -p /etc/wiki

# Generate DB secret
```

```
openssl rand -base64 32 > /etc/wiki/.db-secret

# Create internal docker network
docker network create wikinet

# Create data volume for PostgreSQL
docker volume create pgdata

# Create the containers
docker create --name=db -e POSTGRES_DB=wiki -e POSTGRES_USER=wiki -e
POSTGRES_PASSWORD_FILE=/etc/wiki/.db-secret -v
/etc/wiki/.db-secret:/etc/wiki/.db-secret:ro -v
pgdata:/var/lib/postgresql/data --restart=unless-stopped -h db
--network=wikinet postgres:11
docker create --name=wiki -e DB_TYPE=postgres -e DB_HOST=db -e
DB_PORT=5432 -e DB_PASS_FILE=/etc/wiki/.db-secret -v
/etc/wiki/.db-secret:/etc/wiki/.db-secret:ro -e DB_USER=wiki -e
DB_NAME=wiki -e UPGRADE_COMPANION=1 --restart=unless-stopped -h wiki
--network=wikinet -p 80:3000 -p 443:3443 ghcr.io/requarks/wiki:2
docker create --name=wiki-update-companion -v
/var/run/docker.sock:/var/run/docker.sock:ro --restart=unless-stopped
-h wiki-update-companion --network=wikinet
ghcr.io/requarks/wiki-update-companion:latest
```

Start Container

```
docker start db
docker start wiki
docker start wiki-update-companion
```

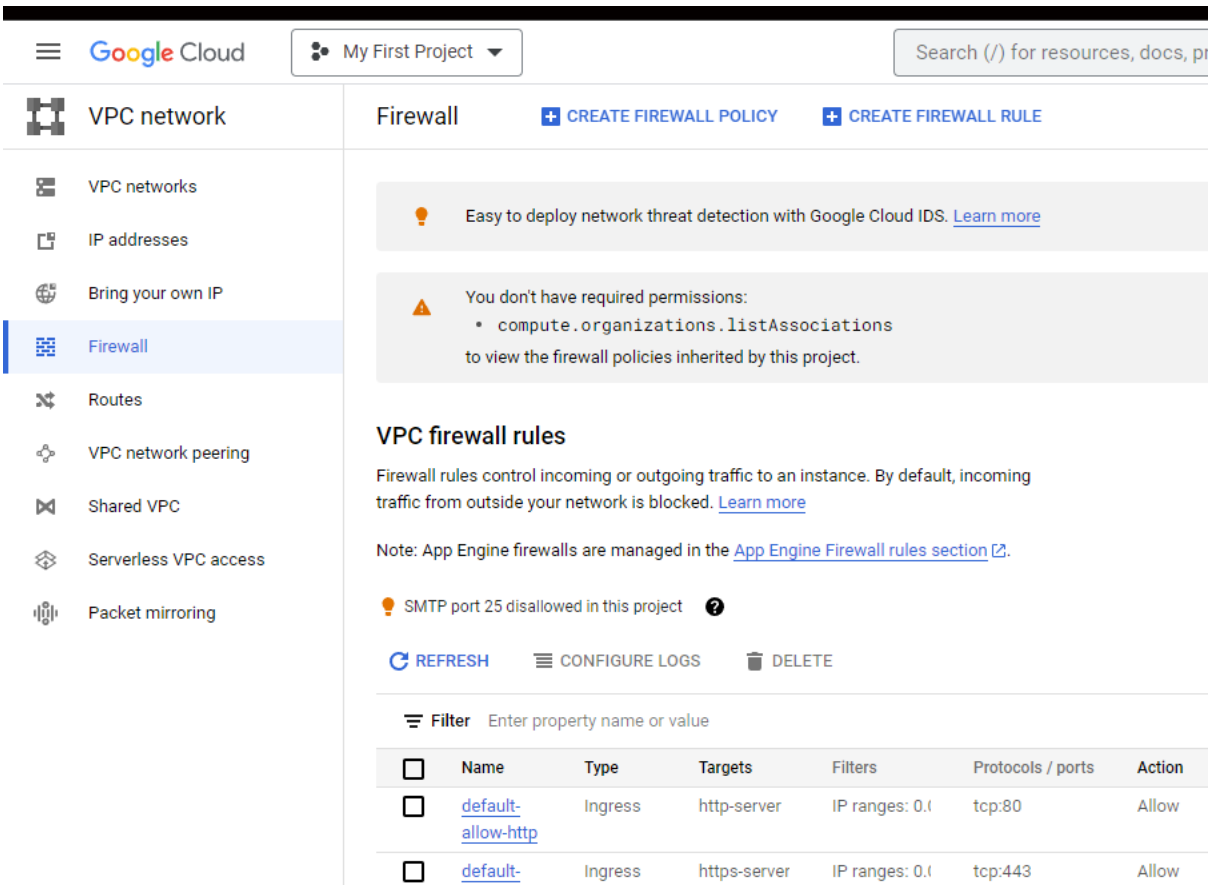
Gitea Setup

Install portainer to make it easier

```
docker volume create portainer_data

docker run -d -p 8000:8000 -p 9443:9443 --name portainer \
    --restart=always \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v portainer_data:/data \
    portainer/portainer-ce:2.11.0
```

Enable port 9443 in firewall

VPC -> Firewall -> Create Firewall Rule



Add port 9443 and your cloud network tags, for our case its http-server and https-server

Allow

Targets
Specified target tags

Target tags *
http-server https-server

Source filter
IPv4 ranges

Source IPv4 ranges *
0.0.0.0/0

Second source filter
None

Protocols and ports

- ○ Allow all
- ◉ Specified protocols and ports

☑ TCP

Ports
9443

E.g. 20, 50-60

☐ UDP

Ports

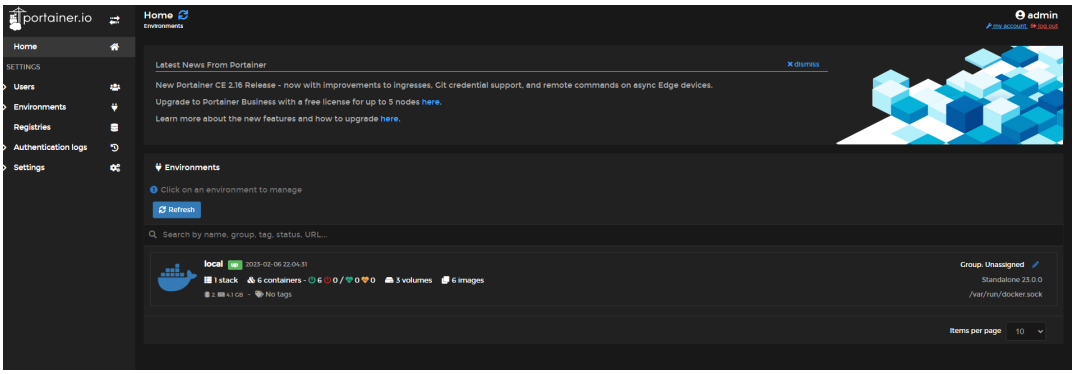
E.g. all

☐ Other

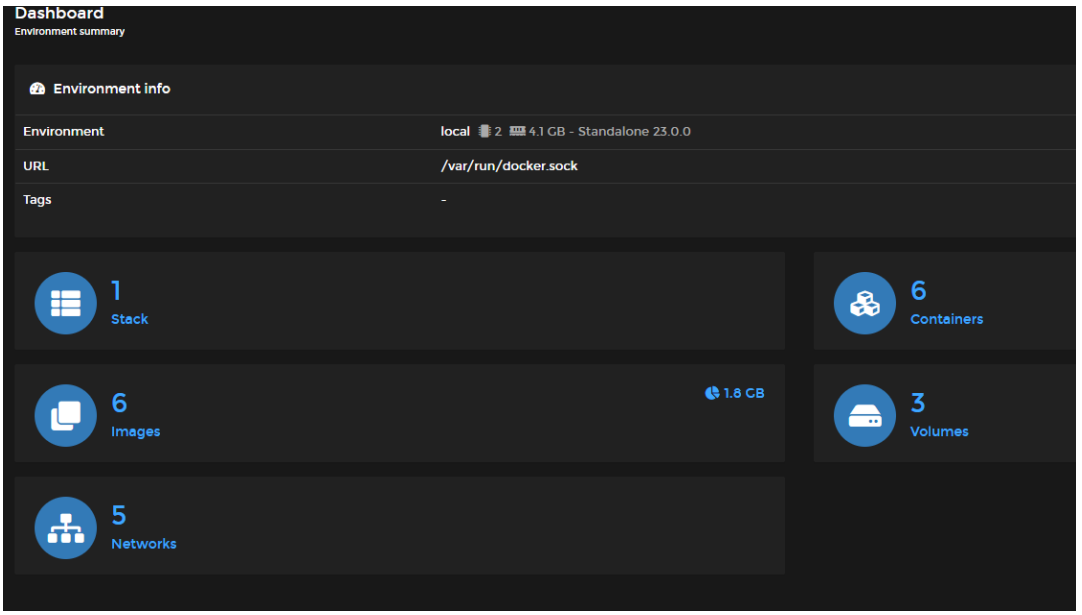
Protocols

Separate multiple protocols by commas, e.g. ah, sctp

Open portainer using ur browser key in the url
https://<machine_ip>:9443
Portainer will ask to create user, create one



This is the home page go into local environment



Click on stack then add stack

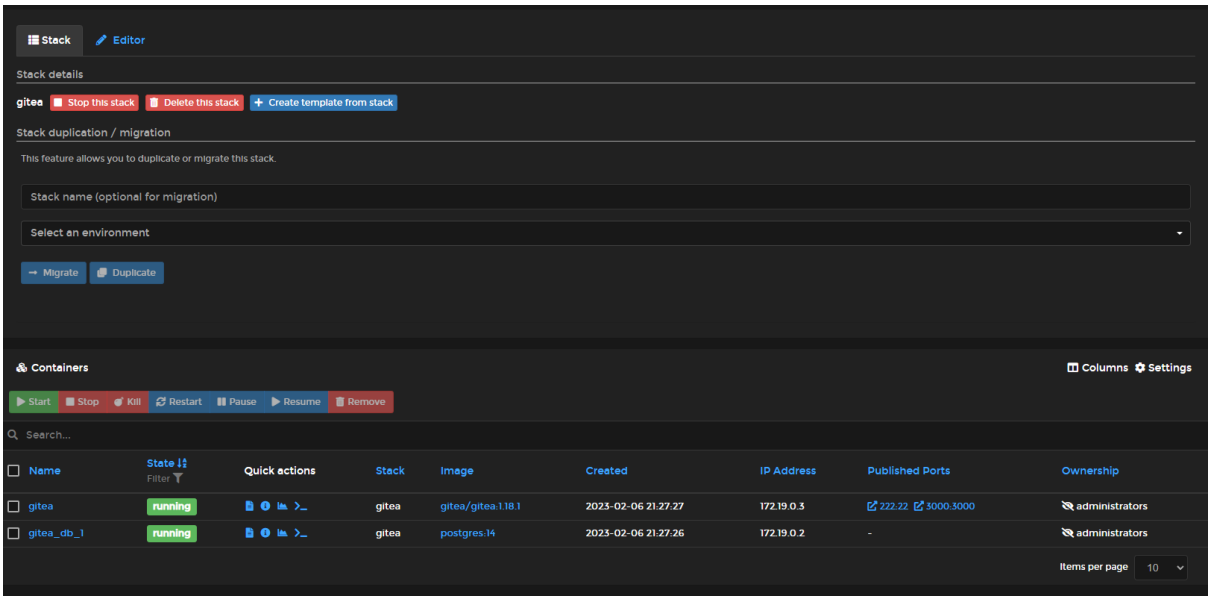
Put in this yaml code

```
-----------------------------------------------------------------
version: "3"

networks:
  gitea:
    external: false

services:
  server:
    image: gitea/gitea:1.18.1
    container_name: gitea
    environment:
      - USER_UID=<user uid>
      - USER_GID=<user gid>
      - GITEA__database__DB_TYPE=postgres
      - GITEA__database__HOST=db:5432
      - GITEA__database__NAME=gitea
      - GITEA__database__USER=gitea
      - GITEA__database__PASSWD=gitea
    restart: always
    networks:
      - wikinet
    volumes:
      - ./gitea:/data
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "3000:3000"
```

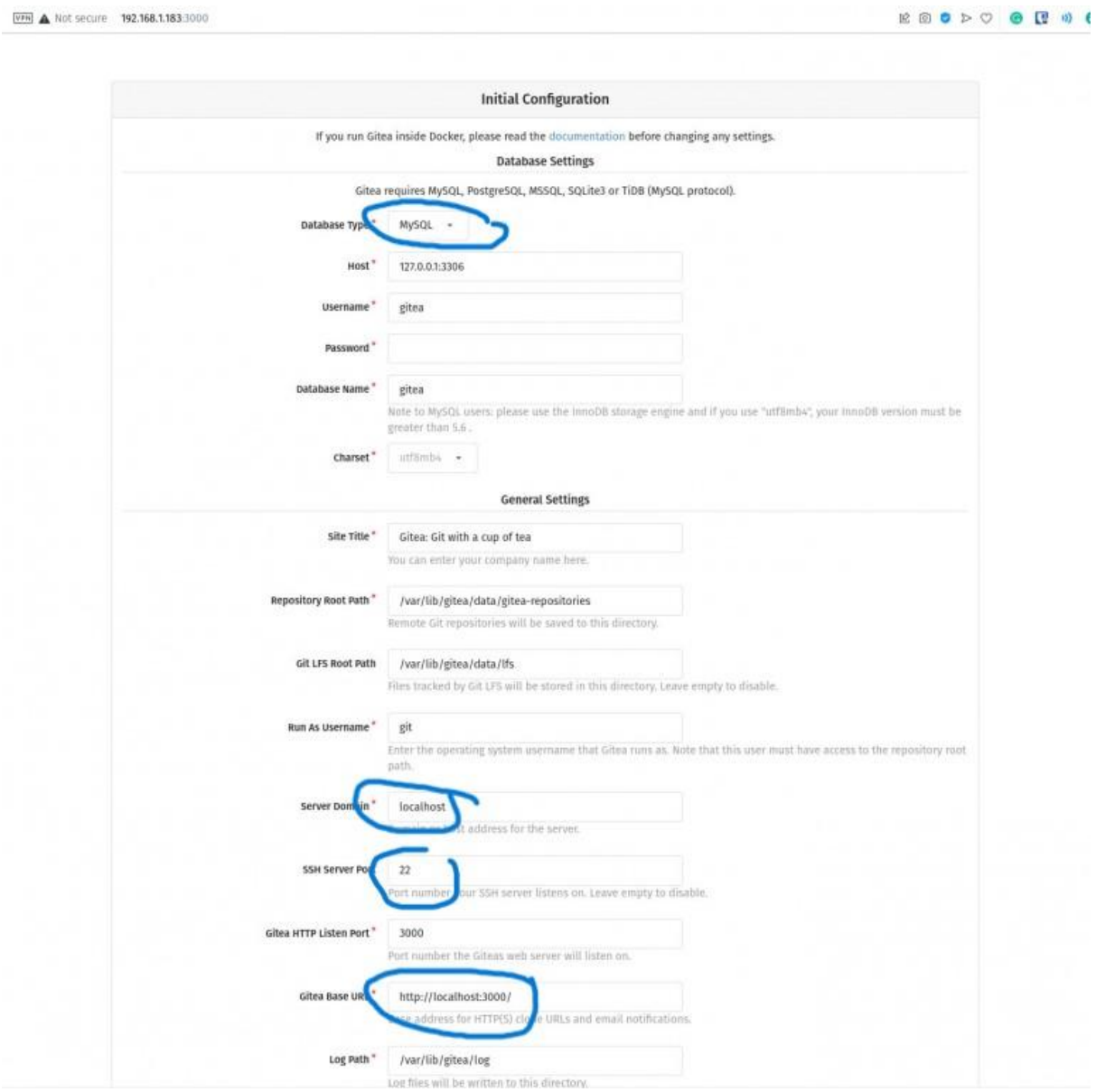
```
      - "222:22"
    depends_on:
      - db

  db:
    image: postgres:14
    restart: always
    environment:
      - POSTGRES_USER=gitea
      - POSTGRES_PASSWORD=gitea
      - POSTGRES_DB=gitea
    networks:
      - gitea
    volumes:
      - ./postgres:/var/lib/postgresql/data
-------------------------------------------------------------------------
----
```

Click deploy stack (it may take few minutes

You will get this



Then we need to enable port 3000 in firewall rule to open up gitea (follow previous steps)



Change the circle above

MySQL -> postgres

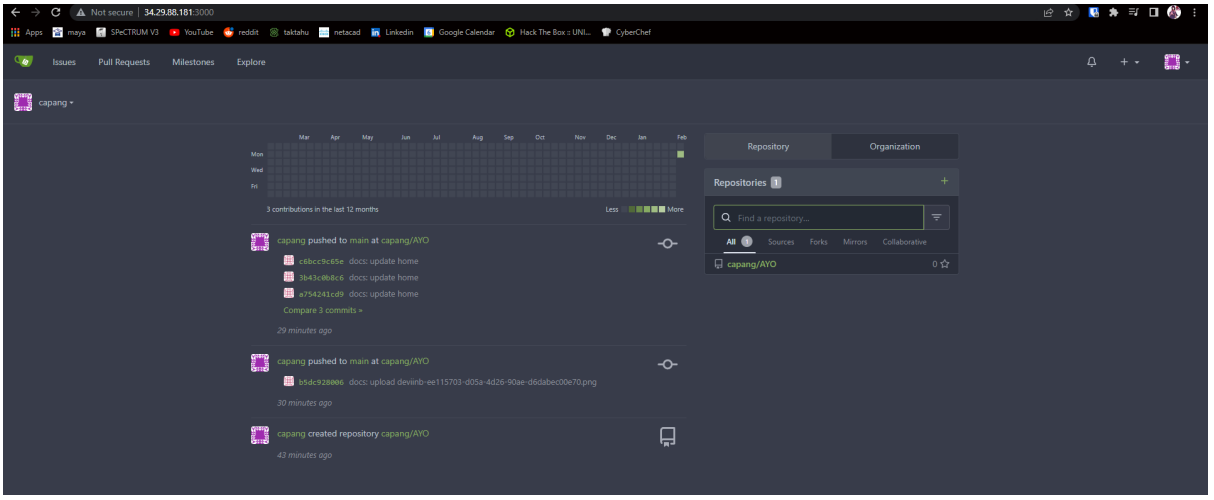
Localhost -> <machine_ip>

22 -> 222

http://localhost:3000/ -> http://<machine_ip>:3000/

Below can configure administrator user, go ahead create one then click install gitea

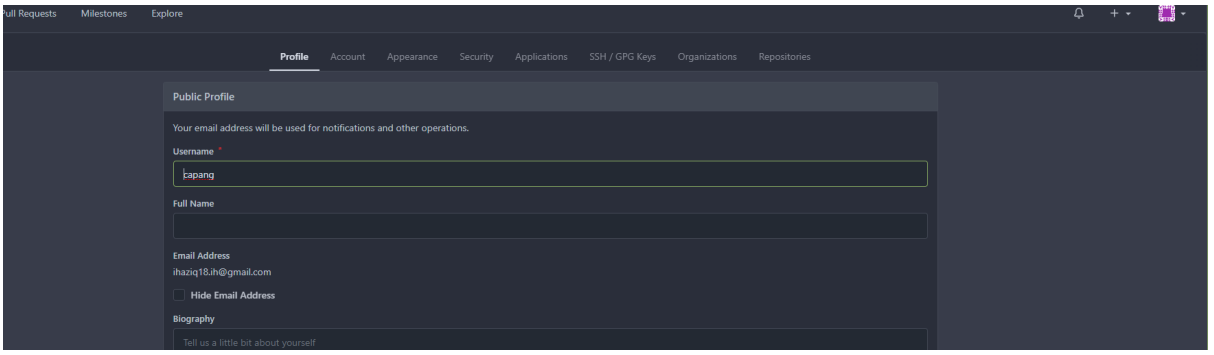
Then we got into gitea!



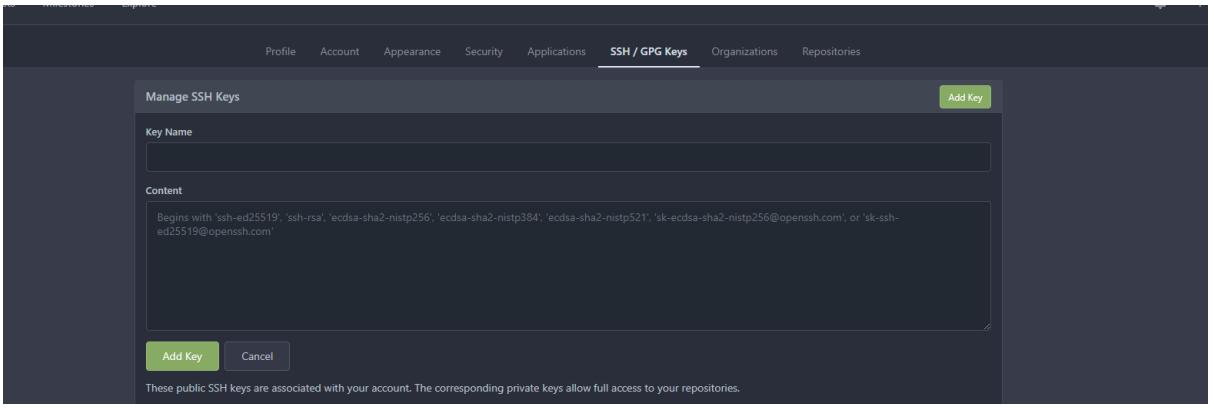
Now we going to enable git sync with wiki.js
First we need to add firewall rule for port 222 (follow previous steps)
Next we create a repository
Then run this command in your machine terminal

```
ssh-keygen -t rsa -b 4096
```

Browse to the where it is stored
Put in our public key in gitea
Settings -> SSH/GPG Key

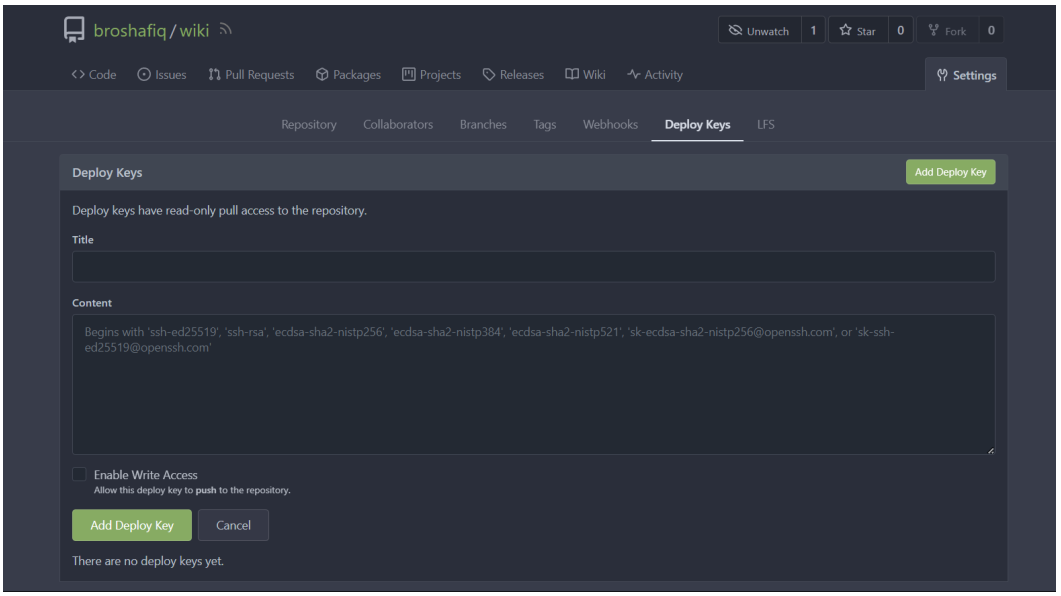


Click add key then paste in our public ssh key recently generated

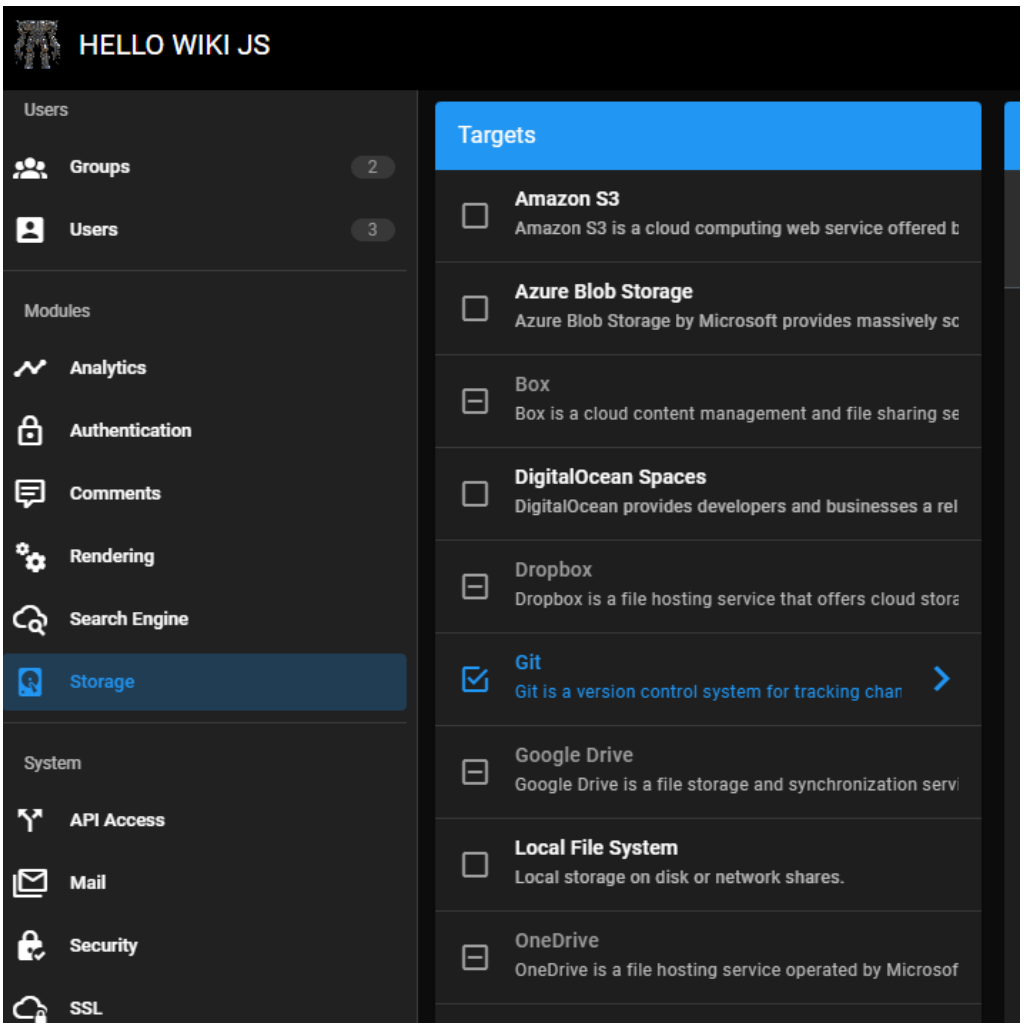


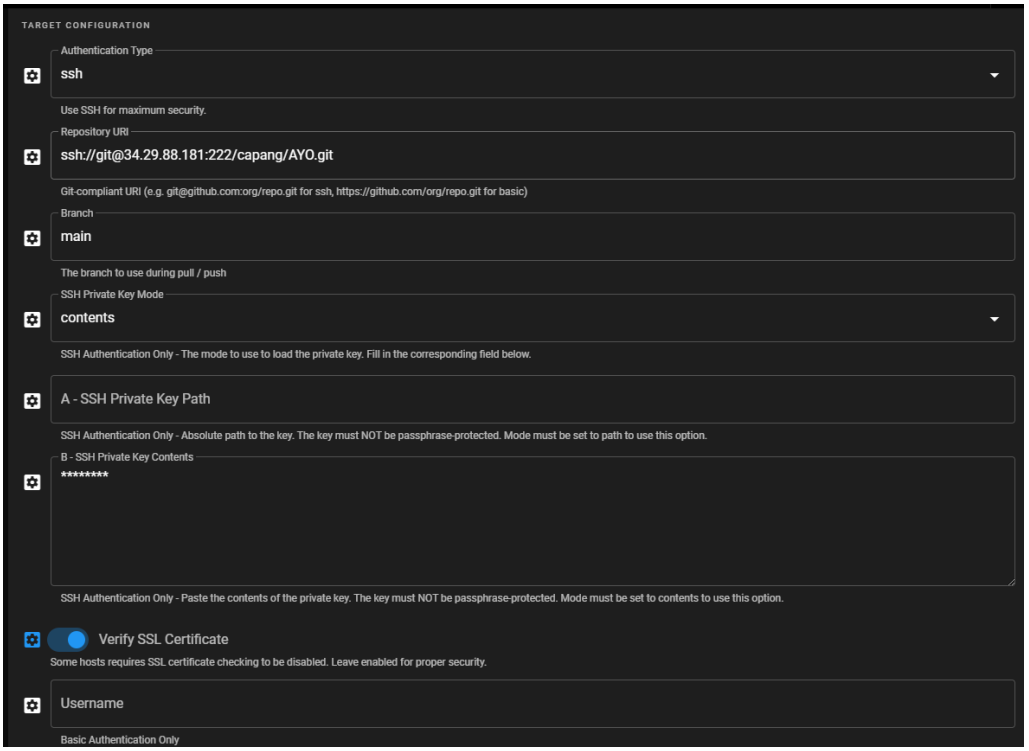
Alternatively, can go to repository that was created > Settings

Then deploy keys,
Click Add Deploy Key then copy pase the SSH key generated



In our wiki.js administrator page/settings page
Setup git sync
Browse to storage -> git

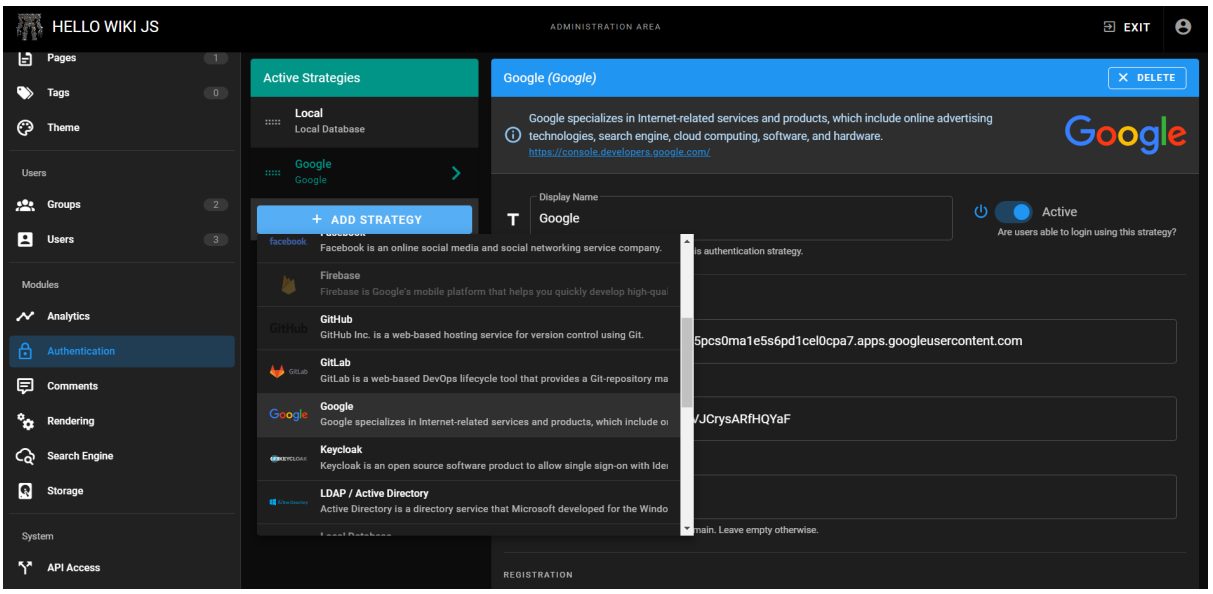

HELLO WIKI JS
Users
Groups 2
Users 3
Modules
Analytics
Authentication
Comments
Rendering
Search Engine
Storage
System
API Access
Mail
Security
SSL
Targets
Amazon S3
Amazon S3 is a cloud computing web service offered b
Azure Blob Storage
Azure Blob Storage by Microsoft provides massively sc
Box
Box is a cloud content management and file sharing se
DigitalOcean Spaces
DigitalOcean provides developers and businesses a rel
Dropbox
Dropbox is a file hosting service that offers cloud stora
Git
Git is a version control system for tracking chan
Google Drive
Google Drive is a file storage and synchronization servi
Local File System
Local storage on disk or network shares.
OneDrive
OneDrive is a file hosting service operated by Microsof


TARGET CONFIGURATION
Authentication Type
ssh
Use SSH for maximum security.
Repository URI
ssh://git@34.29.88.181:222/capang/AYO.git
Git-compliant URI (e.g. git@github.com:org/repo.git for ssh, https://github.com/org/repo.git for basic)
Branch
main
The branch to use during pull / push
SSH Private Key Mode
contents
SSH Authentication Only - The mode to use to load the private key. Fill in the corresponding field below.
A - SSH Private Key Path
SSH Authentication Only - Absolute path to the key. The key must NOT be passphrase-protected. Mode must be set to path to use this option.
B - SSH Private Key Contents
****
SSH Authentication Only - Paste the contents of the private key. The key must NOT be passphrase-protected. Mode must be set to contents to use this option.
Verify SSL Certificate
Some hosts requires SSL certificate checking to be disabled. Leave enabled for proper security.
Username
Basic Authentication Only

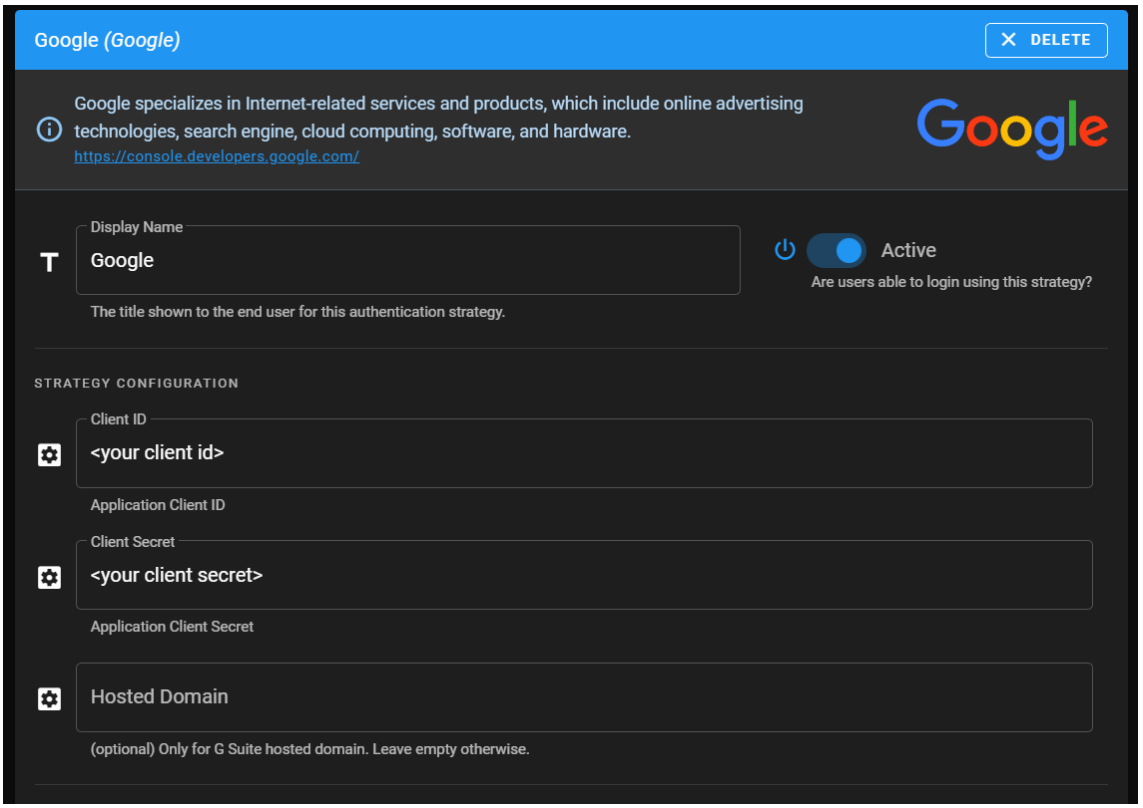
Select authentication type to ssh
Put in repo url we copy from recently created repo
Select ssh private key mode into contents
SSH private key contents we copy from previously created
Then click apply
If no error then we good

Wiki.js Setup

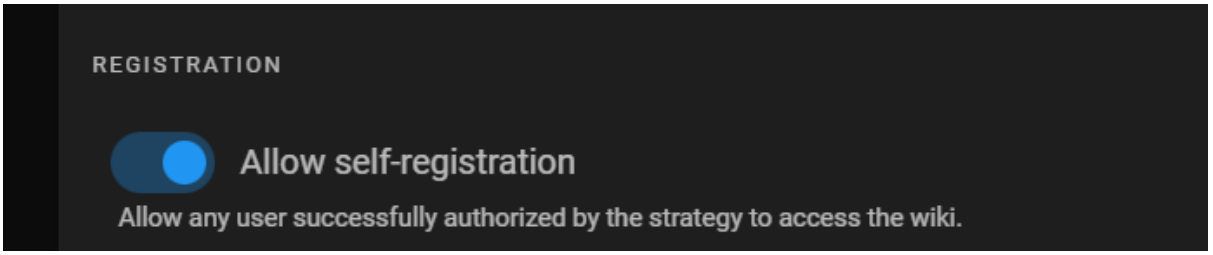
1. **Authentication (Follow walkthrough https://docs.requarks.io/auth/google)**
    a. **Creating a Google Cloud Project**
        i. **Launch the <u>Google Cloud Console</u> and create a new project *(if not already the case)*.**
        ii. **From the left sidebar, click on APIs & Services.**
        iii. **At the top, click on Enable APIs and Services**
        iv. **Click on the Google+ API tile and enable it.**
        v. **From the left sidebar, mouse over APIs & Services and choose Credentials in the sub-menu.**
        vi. **Click on the blue Create credentials button and choose OAuth client ID.**
        vii. **Select Web application as the application type.**
        viii. **Give a proper name *(e.g. Wiki.js)***
        ix. **Leave the other fields empty for now, we'll fill them later.**
        x. **Click Create to be presented with the Client ID and Client Secret. Copy these 2 keys. We'll need them later.**
    b. **Enable google strategy in Wiki js**
        i. **In the Administration Area of your wiki, click on Authentication in the left navigation.**
        ii. **Click on Google.**



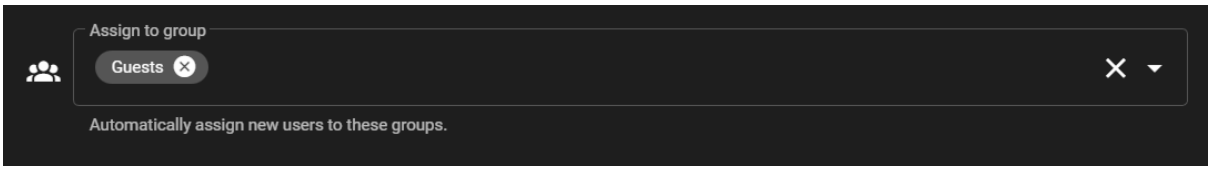
iii. **Enter the Client ID and Client Secret values copied earlier.**



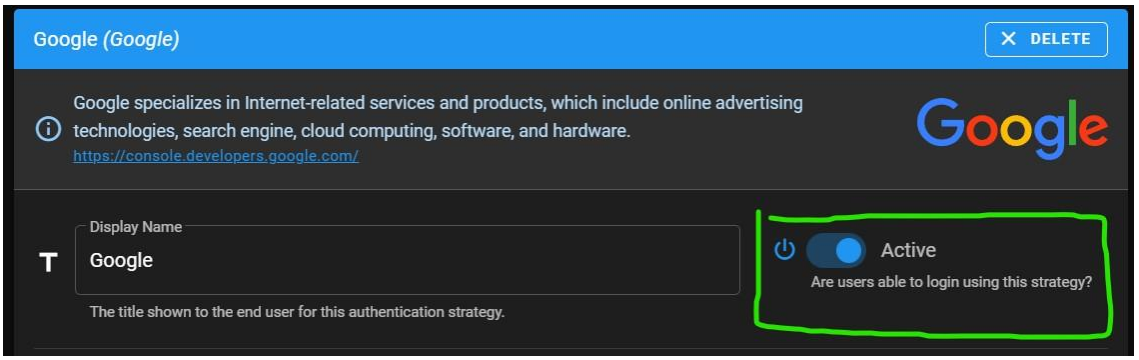
iv. **Enable the Self-registration option *(unless you plan on authorizing users manually)*.**



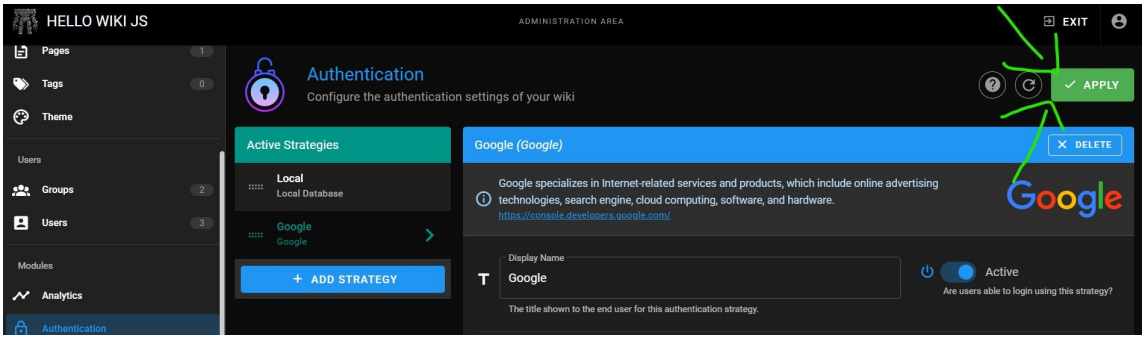
v. **Select the group new users should be assigned to when they login for the first time.**



vi. **Make sure the checkbox next to Google in the list of strategies is checked. The text should now say that the strategy is active.**



vii. **Click Apply on the upper right of the page to save and apply the configuration.**



c. **Enter allowed endpoints on Google**
   i. **Going back to the Credentials page on the Google Cloud Console, click the edit icon next to the OAuth client you created earlier.**
   ii. **Enter the wiki's domain for Authorized JavaScript origins.**
   iii. **Enter the redirect URI (found under Configuration Reference displayed below the settings of the Google strategy in Wiki.js) in Authorized redirect URIs**
   iv. **Click Save.**

d. **Set the OAuth consent screen**
   i. **From the same Credentials page, click on the oAuth consent screen tab.**

ii. **Fill in the name, logos, emails, etc as needed.**
iii. **In the scopes for Google APIs, make sure the following scopes are listed**
    1. **Email**
    2. **Profile**
    3. **Openid**
iv. **In the authorized domains, make sure the domain of your wiki is listed.**

-Shafiqps
-broCapang