HIT400 Application Deployment Guide for Ubuntu

This guide walks through the complete setup of a Rails application on Ubuntu with PostgreSQL (including pgvector extension), Supervisor for process management, Nginx as a web server, and Let's Encrypt SSL certificates generated with Certbot.

Table of Contents

System Requirements

- Ubuntu 22.04 LTS or newer
- At least 2GB RAM (4GB recommended)
- At least 20GB of disk space
- A domain name pointing to your server's IP address

Initial Server Setup

1. Update your system:

```
sudo apt update
sudo apt upgrade -y
```

2. Create a deployment user (optional but recommended):

```
sudo adduser deploy
sudo usermod -aG sudo deploy
```

3. Set up SSH keys for secure access (optional):

```
ssh-copy-id deploy@your_server_ip
```

4. Configure the firewall:

```
sudo apt install -y ufw
sudo ufw allow OpenSSH
sudo ufw allow 'Nginx Full'
sudo ufw enable
```

Installing Dependencies

1. Install essential packages:

```
sudo apt install -y curl git build-essential libssl-dev libreadline-dev
zlib1g-dev \
                    libpq-dev libffi-dev nodejs npm imagemagick
```

2. Install Yarn:

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
sudo apt update && sudo apt install -y yarn
```

PostgreSQL Setup

1. Install PostgreSQL:

```
sudo apt install -y postgresql postgresql-contrib
```

2. Install pgvector extension:

```
sudo apt install -y postgresql-server-dev-14  # Replace 14 with your
PostgreSQL version
git clone https://github.com/pgvector/pgvector.git
cd pgvector
make
sudo make install
cd ..
rm -rf pgvector
```

3. Create a database user and database:

```
sudo -u postgres psql -c "CREATE USER your_app_name WITH PASSWORD
'your_password';"
sudo -u postgres psql -c "CREATE DATABASE your_app_name_production OWNER
your_app_name;"
sudo -u postgres psql -c "ALTER USER your_app_name WITH SUPERUSER;"  #
Needed temporarily for pgvector
```

4. Enable pgvector extension:

```
sudo -u postgres psql -d your_app_name_production -c "CREATE EXTENSION
vector;"
```

5. Remove superuser privileges (after extension is created):

```
sudo -u postgres psql -c "ALTER USER your_app_name WITH NOSUPERUSER;"
```

Ruby and Rails Installation

1. Install rbenv:

```
git clone https://github.com/rbenv/rbenv.git ~/.rbenv
cd ~/.rbenv && src/configure && make -C src
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
source ~/.bashrc
```

2. Install ruby-build:

```
git clone https://github.com/rbenv/ruby-build.git "$(rbenv
root)"/plugins/ruby-build
```

3. Install Ruby:

```
rbenv install 3.2.6  # Replace with your Ruby version
rbenv global 3.2.6
```

4. Install Bundler and Rails:

```
gem install bundler
gem install rails
```

Application Deployment

1. Clone your application:

```
git clone https://github.com/SlimGee/machine.git /home/deploy/machine
cd /home/deploy/your_app_name
```

2. Install dependencies:

```
bundle install --deployment --without development test
yarn install --production
```

3. Configure database:
4. Edit the database.yml file:

```
vim config/database.yml
```

Update the production section:

```
VISUAL=vim rails credentials:edit -e production
```

Change the database password to your postgresql db username and password

5. Compile assets:

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

6. Run migrations:

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

Environment Variables and Secrets

8. Set up RAILS_ENV in .bash_profile:

```
echo 'export RAILS_ENV=production' >> ~/.bash_profile
source ~/.bash_profile
```

Nginx Configuration

9. Install Nginx:

```
sudo apt install -y nginx
```

10. Create Nginx server block:

```
sudo vim /etc/nginx/sites-available/your_app_name
```

3. Add the following configuration:

```
upstream puma {
  server unix:///home/deploy/machine/shared/tmp/sockets/puma.sock;
}

server {
  listen 80;
  server_name your-domain.com www.your-domain.com;
  root /home/deploy/machine/public;

  # Allow uploads up to 100MB in size
  client_max_body_size 100m;

  location ^~ /assets/ {
    gzip_static on;
    expires max;
    add_header Cache-Control public;
  }

  try_files $uri/index.html $uri @puma;

  location @puma {
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_redirect off;
    proxy_pass http://puma;
  }

  error_page 500 502 503 504 /500.html;
  error_page 404 /404.html;
}
```

4. Enable the server block:

```
sudo ln -s /etc/nginx/sites-available/your_app_name /etc/nginx/sites-
enabled/
sudo nginx -t  # Test the configuration
sudo systemctl restart nginx
```

SSL Configuration with Let's Encrypt

1. Install Certbot:

```
sudo apt install -y certbot python3-certbot-nginx
```

2. Generate SSL certificates:

```
sudo certbot --nginx -d your-domain.com -d www.your-domain.com
```

3. Follow the prompts and select the option to redirect HTTP traffic to HTTPS.
4. Set up auto-renewal:

```
sudo systemctl status certbot.timer  # Verify the timer is active
```

Supervisor Setup

1. Install Supervisor:

```
sudo apt install -y supervisor
```

2. Create a configuration file for your Rails application:

```
sudo vim /etc/supervisor/conf.d/machine.conf
```

3. Add the following configuration:

```
[program:machine]
command=/home/deploy/.rbenv/shims/bundle exec puma -C config/puma.rb -
production
directory=/home/deploy/your_app_name
user=deploy
autostart=true
autorestart=true
stdout_logfile=/home/deploy/your_app_name/log/supervisor.log
stderr_logfile=/home/deploy/your_app_name/log/supervisor.error.log
```

4. Create directories for sockets and pids:

```
mkdir -p /home/deploy/your_app_name/shared/tmp/sockets
mkdir -p /home/deploy/your_app_name/shared/tmp/pids
```

5. Create or update config/puma.rb:

```
# config/puma.rb
directory '/home/deploy/machine'
environment ENV.fetch('RAILS_ENV') { 'production' }
threads_count = ENV.fetch('RAILS_MAX_THREADS') { 5 }
threads threads_count, threads_count

bind 'unix:///home/deploy/your_app_name/shared/tmp/sockets/puma.sock'
pidfile '/home/deploy/your_app_name/shared/tmp/pids/puma.pid'
state_path '/home/deploy/your_app_name/shared/tmp/pids/puma.state'

workers ENV.fetch('WEB_CONCURRENCY') { 2 }
preload_app!

plugin :tmp_restart

plugin :solid_queue
```

6. Reload and start Supervisor:

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl start machine
```

Final Steps

1. Verify that your application is running:

```
sudo supervisorctl status machine
```

2. Check the Nginx status:

```
sudo systemctl status nginx
```

3. Visit your domain in a browser to make sure everything is working.
4. Set up log rotation (optional but recommended):

```
sudo vim /etc/logrotate.d/your_app_name
```

Add:

```
/home/deploy/your_app_name/log/*.log {
  daily
```

```
  missingok
  rotate 7
  compress
  delaycompress
  notifempty
  copytruncate
}
```

Troubleshooting

If the application won't start:

1. Check Supervisor logs:

```
sudo supervisorctl tail -f your_app_name
```

2. Check Rails logs:

```
tail -f /home/deploy/your_app_name/log/production.log
```

3. Check Nginx logs:

```
sudo tail -f /var/log/nginx/error.log
```

If you're having database connection issues:

```
sudo -u postgres psql -d your_app_name_production -c "GRANT ALL PRIVILEGES
ON DATABASE your_app_name_production TO your_app_name;"
```

If pgvector extension is causing issues:

Make sure the extension is properly installed:

```
sudo -u postgres psql -d your_app_name_production -c "\dx"
```

If Certbot fails:

Check if port 80 is open and not blocked by a firewall:

```
sudo ufw status
```

And ensure Nginx is properly configured and running:

```
sudo nginx -t
sudo systemctl restart nginx
```

Restart everything after major changes:

```
sudo supervisorctl restart your_app_name
sudo systemctl restart nginx
```

Remember to replace placeholders like `your_app_name`, `your-domain.com`, `your_password`, and `your_master_key` with your actual values throughout this guide.

Adding Intial Data

At this point the system is empty without any data. To start importing data start the rails console:

```
rails c
```

Then run the following command to import the data:

```
ImportAssetsJob.perform_later
```

```
FeedIngestionJob.perform_later
```

```
FetchMaliciousDomainsJob.perform_later
```

```
AnalsysJob.perform_later
```

After this the system will start populating the database, analysis information maybe be available in a couple of days