Linux部署项目

1. Linux部署方式

Linux部署只要分为两种方式：一种是将项目部署在服务器上，包括本地的硬件服务器、云端的虚拟服务器（阿里云、腾讯云、百度云等），此类部署均需要购置服务器或者网络付费购买；另一种则是基于自己计算机的虚拟机，可以通过VMvare、VirtualBox等虚拟化工具创建，也可以通过Vagrant搭配VirtualBox的方式搭建虚拟机来进行部署。

此次也是通过Vagrant搭配VirtualBox的方式搭建虚拟机来进行部署，通过Vagrant来管理虚拟机方便的地方就在于可以通过配置、命令的方式管理虚拟机。

2. Vagrant搭建服务器

关于Vagrant以及VirtualBox的具体安装及使用方法，可以查看另一篇文章Django开发总结-（一）Vagrant虚拟环境搭建。需要注意的地方就是Vagrant与VirtualBox搭配使用时软件版本是有依赖的，如果版本不搭配可能会造成无法使用。此处使用的版本分别是：

- Vagrant：2.2.3
- VirtualBox：5.2.22

2.1. box文件添加

此次使用的Linux发行版是CentOS7.2，通过`vagrant box add`命令在本地仓库添加镜像文件。

```
D:\VirtualMachine\Vagrant\project\microblog
$ vagrant box add superwong/centos7_x86_64 ..\..\boxes\centos7_1902.01_64.box
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'superwong/centos7_x86_64' (v0) for provider:
    box: Unpacking necessary files from:
file://D:/VirtualMachine/Vagrant/boxes/centos7_1902.01_64.box
    box:
==> box: Successfully added box 'superwong/centos7_x86_64' (v0) for 'virtualbox'!

D:\VirtualMachine\Vagrant\project\microblog
$ vagrant box list
superwong/centos7_x86_64 (virtualbox, 0)
```

创建一个虚拟机工作目录，通过`vagrant init`命令初始化，获取到Vagrant的初始化配置文件`Vagrantfile`。

```
D:\VirtualMachine\Vagrant\project\microblog
$ vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.

D:\VirtualMachine\Vagrant\project\microblog
$ ls
Vagrantfile
```

2.2. 配置并启动虚拟机

修改`Vagrantfile`，设置虚拟机配置信息。

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  # 基础box镜像名称
  config.vm.box = "superwong/centos7_x86_64"
  # 设置虚拟的HOSTNAME
  config.vm.hostname = "microblog"

  # 端口映射，目前不开启
  # config.vm.network "forwarded_port", guest: 80, host: 8080
  # config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"

  # 私有网络，相当于常说的host-only模式，设置的IP只用于宿主机对虚拟机的访问
  config.vm.network "private_network", ip: "192.168.33.10"

  # 公有网络，相当于常说的birdge模式
  config.vm.network "public_network"

  # 共享目录映射，便于宿主机向虚拟机共享文件
  # 默认存在"." => "/vagrant"映射关系，即配置文件所在目录与虚拟机/vagrant目录的映射。
```

```
  # config.vm.synced_folder "../data", "/vagrant_data"

  # 虚拟机相关配置
  config.vm.provider "virtualbox" do |vb|
    # 是否开启图形化界面，此处关闭
    vb.gui = false
    # 虚拟机内存大小
    vb.memory = "1024"
    # 设置虚拟机名称
    vb.name = "microblog"
    # 设置虚拟机的CPU数
    vb.cpus = 2
  end
end
```

通过`vagrant up`命令，启动虚拟机。

```
D:\VirtualMachine\Vagrant\project\microblog
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'superwong/centos7_x86_64'...
==> default: Matching MAC address for NAT networking...
==> default: Setting the name of the VM: microblog
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
    default: Adapter 3: bridged
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
[default] No installation found.
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: centos.ustc.edu.cn
 * extras: centos.ustc.edu.cn
 * updates: centos.ustc.edu.cn
Package binutils-2.27-34.base.el7.x86_64 already installed and latest version
Package 1:make-3.82-23.el7.x86_64 already installed and latest version
Package bzip2-1.0.6-13.el7.x86_64 already installed and latest version
```

```
Resolving Dependencies
--> Running transaction check
......
Dependencies Resolved

================================================================================
 Package                   Arch      Version                   Repository  Size
================================================================================
Installing:
......
Installing for dependencies:
......
Updating for dependencies:
......

Transaction Summary
================================================================================
Install  4 Packages (+32 Dependent packages)
Upgrade             (  4 Dependent packages)

Total download size: 92 M
Downloading packages:
Delta RPMs reduced 3.9 M of updates to 816 k (79% saved)
Public key for glibc-headers-2.17-260.el7_6.4.x86_64.rpm is not installed
warning: /var/cache/yum/x86_64/7/updates/packages/glibc-headers-2.17-
260.el7_6.4.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID f4a80eb5: NOKEY
Public key for libmpc-1.0.1-3.el7.x86_64.rpm is not installed
--------------------------------------------------------------------------------
Total                                              433 kB/s |  89 MB  03:31
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Importing GPG key 0xF4A80EB5:
 Userid     : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"
 Fingerprint: 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5
 Package    : centos-release-7-6.1810.2.el7.centos.x86_64 (@anaconda)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
......
Installed:
......
Dependency Installed:
......
Dependency Updated:
......

Complete!
Copy iso file C:\Program Files\Oracle\VirtualBox\VBoxGuestAdditions.iso into the box
/tmp/VBoxGuestAdditions.iso
Mounting Virtualbox Guest Additions ISO to: /mnt
mount: /dev/loop0 is write-protected, mounting read-only
Installing Virtualbox Guest Additions 5.2.22 - guest version is unknown
```

```
Verifying archive integrity... All good.
Uncompressing VirtualBox 5.2.22 Guest Additions for Linux........
VirtualBox Guest Additions installer
Copying additional installer modules ...
Installing additional modules ...
VirtualBox Guest Additions: Building the VirtualBox Guest Additions kernel modules.  This
may take a while.

VirtualBox Guest Additions: Look at /var/log/vboxadd-setup.log to find out what went wrong
VirtualBox Guest Additions: Starting.
Redirecting to /bin/systemctl start vboxadd.service
Redirecting to /bin/systemctl start vboxadd-service.service
Unmounting Virtualbox Guest Additions ISO from: /mnt
==> default: Checking for guest additions in VM...
==> default: Setting hostname...
==> default: Configuring and enabling network interfaces...
==> default: Rsyncing folder: /cygdrive/d/VirtualMachine/Vagrant/project/microblog/ =>
/vagrant
```

通过`vagrant ssh`免密登录虚拟机。

```
D:\VirtualMachine\Vagrant\project\microblog
$ vagrant ssh
[vagrant@microblog ~]$ pwd
/home/vagrant
```

2.3. python3环境搭建

python36环境搭建方法，文章Django开发总结-（一）Vagrant虚拟环境搭建中也有涉及。

2.4. 安装其他软件

对于数据库服务器，将从SQLite切换到MySQL。 Postfix包是一个邮件传输代理，我将用它来发送电子邮件。 Supervisor工具将监视Flask服务器进程，并在其崩溃时自动重启，并当Supervisor服务重启后自动启动其监视的服务。 Nginx服务器将接受来自外部世界的所有请求，并将它们转发给应用程序。最后，我将使用git来从git仓库下载应用程序。

对于此部署，选择不安装Elasticsearch。 这项服务需要大量的RAM，所以只有拥有超过2GB内存的大型服务器时才可以考虑。 为了避免服务器内存不足的问题，将停用搜索功能。 如果有高配的服务器，可以从Elasticsearch站点下载官方的软件包，并按照其安装说明将其添加到服务器。

默认安装的postfix可能不足以在生产环境中发送电子邮件。 为了避免垃圾邮件和恶意邮件，很多服务器都要求发件人服务器通过安全扩展标识自己，这意味着至少必须拥有与服务器相关联的域名。

安装mysql：

```
[root@microblog software]# cd /usr/local/software/
[root@microblog software]# wget http://repo.mysql.com/mysql-community-release-el7-
5.noarch.rpm
[root@microblog software]# yum install mysql-server
```

安装supervisor：

```
[root@microblog software]# yum install -y epel-release
[root@microblog software]# yum install -y supervisor
```

安装nginx：

```
[root@microblog software]# yum install -y gcc gcc-c++ zlib zlib-devel pcre-devel openssl
openssl-devel
[root@microblog software]# wget http://nginx.org/download/nginx-1.15.8.tar.gz
[root@microblog software]# tar -zxf nginx-1.15.8.tar.gz
[root@microblog software]# cd nginx-1.15.8
[root@microblog software]# ./configure --with-http_ssl_module --with-
http_gzip_static_module --with-http_flv_module --with-http_stub_status_module
[root@microblog software]# make install
```

安装postfix、git：

```
yum -y install postfix git
```

3. 安装应用

3.1. 克隆项目代码

通过`git clone`命令从远程git服务器克隆项目代码。

```
[vagrant@microblog ~]$ mkdir projects
[vagrant@microblog ~]$ cd projects/
[vagrant@microblog projects]$ git clone https://gitee.com/superwong/flask-mega-tutorial.git
Cloning into 'flask-mega-tutorial'...
remote: Enumerating objects: 362, done.
remote: Counting objects: 100% (362/362), done.
remote: Compressing objects: 100% (338/338), done.
remote: Total 362 (delta 181), reused 0 (delta 0)
Receiving objects: 100% (362/362), 11.79 MiB | 1.01 MiB/s, done.
Resolving deltas: 100% (181/181), done.
[vagrant@microblog projects]$ ls
flask-mega-tutorial
```

3.2. 创建虚拟环境

通过`python3 -m venv venv`命令创建虚拟环境。

```
[vagrant@microblog projects]$ cd flask-mega-tutorial/
[vagrant@microblog flask-mega-tutorial]$ python3 -m venv venv
[vagrant@microblog flask-mega-tutorial]$ ls
app  babel.cfg  docs  LICENSE  migrations  README.en.md  README.md  requirements.txt
setup.cfg  tests  venv
```

通过`source venv/bin/activate`命令进入虚拟环境安装项目环境依赖包。

```
[vagrant@microblog flask-mega-tutorial]$ source venv/bin/activate
(venv) [vagrant@microblog flask-mega-tutorial]$ pip install -r requirements.txt
```

除了`requirements.txt`中的包之外，还将使用此生产部署指定的两个包，因此它们不包含在`requirements.txt`文件中。 `gunicorn`软件包是Python应用程序的生产Web服务器。 `pymysql`软件包包含MySQL驱动程序，它使SQLAlchemy能够与MySQL数据库一起工作：

此处选择`gunicorn`作为web服务器的原因是，目前比较主流的python web服务器主要就是`gunicorn`和`uwsgi`，两者的区别基本就是`uwsgi`相比来说功能较为丰富，可配置型强，`gunicorn`相比来说更为稳定，单笔性能更强。

```
pip install gunicorn pymysql
```

3.3. 项目相关环境变量配置

创建`.env`文件，其中包含项目启动需要的环境变量

```
SECRET_KEY=661af64169764eac900a46e32dab0451
MAIL_SERVER=localhost
MAIL_PORT=25
DATABASE_URL=mysql+pymysql://microblog:<db-password>@localhost:3306/microblog
```

其中`SECRET_KEY`参数最好使用随机字符串，可以通过以下命令获取：

```
python3 -c "import uuid; print(uuid.uuid4().hex)"
```

若想flask解析.env文件，需要安装`python-dotenv`三方库。

```
pip install python-dotenv
```

同时修改`app/config.py`脚本，增加`.env`文件的加载。

```
from dotenv import load_dotenv

BASE_DIR = os.path.abspath(os.path.dirname(__file__))
load_dotenv(os.path.join(BASE_DIR, '.env'))
```

`.env`文件可以用于所有配置变量，但是不能用于Flask命令行的`FLASK_APP`和`FLASK_DEBUG`环境变量，因为它们在应用启动的早期（应用实例和配置对象存在之前）就被使用了，因此需要在`~/.bash_profile`配置文件中手动设置。

```
(venv) [vagrant@microblog ~]$ echo "export FLASK_APP=app" >> ~/.bash_profile
(venv) [vagrant@microblog ~]$ source ~/.bash_profile
```

通过`flask --help`可以查看`FLASK_APP`是否配置生效。如果帮助信息显示应用程序已添加的`translate`命令，那么你就知道应用程序已被找到。

```
(venv) [vagrant@microblog flask-mega-tutorial]$ flask --help
```

```
[2019-05-09 07:34:37,787] INFO in logger: 博客已启动
Usage: flask [OPTIONS] COMMAND [ARGS]...

  A general utility script for Flask applications.

  Provides commands from Flask, extensions, and the application. Loads the
  application defined in the FLASK_APP environment variable, or from a
  wsgi.py file. Setting the FLASK_ENV environment variable to 'development'
  will enable debug mode.

    $ export FLASK_APP=hello.py
    $ export FLASK_ENV=development
    $ flask run

Options:
  --version  Show the flask version
  --help     Show this message and exit.

Commands:
  db         Perform database migrations.
  routes     Show the routes for the app.
  run        Runs a development server.
  shell      Runs a shell in the app context.
  translate  翻译命令组
```

执行`flask translate compile`命令，编译语言翻译。

```
(venv) [vagrant@microblog flask-mega-tutorial]$ flask translate compile
[2019-05-09 07:38:18,499] INFO in logger: 博客已启动
compiling catalog app/translations/en/LC_MESSAGES/messages.po to
app/translations/en/LC_MESSAGES/messages.mo
```

3.4. 配置MySQL数据库

在开发过程中使用过的sqlite数据库非常适合简单的应用程序，但是当部署可能需要一次处理多个请求的健壮Web服务器时，最好使用更强大的数据库。 出于这个原因，需要建立一个名为`microblog`的MySQL数据库。

要管理数据库服务器，需要使用`mysql`命令：

```
[vagrant@microblog ~]$ systemctl start mysqld
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to manage system services or units.
Authenticating as: root
Password:
==== AUTHENTICATION COMPLETE ===
[vagrant@microblog ~]$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.44 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

创建名为`microblog`的数据库、用户，并设置用户的密码与权限。`microblog`用户的密码需要与你包含在*.env*文件中的`DATABASE_URL`变量中的密码相匹配。

```
mysql> create database microblog character set utf8 collate utf8_bin;
Query OK, 1 row affected (0.00 sec)

mysql> create user 'microblog'@'localhost' identified by '<db-password>';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges on microblog.* to 'microblog'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> quit;
Bye
```

运行数据库迁移。

```
[vagrant@microblog flask-mega-tutorial]$ source venv/bin/activate
(venv) [vagrant@microblog flask-mega-tutorial]$ flask db upgrade
[2019-05-09 09:43:59,115] INFO in logger: 博客已启动
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 7c2bb80cfe7b, empty message
INFO  [alembic.runtime.migration] Running upgrade 7c2bb80cfe7b -> 7bddf580d26a, empty
message
INFO  [alembic.runtime.migration] Running upgrade 7bddf580d26a -> 99a18f68c13a, '新增
about_me、last_seen字段'
INFO  [alembic.runtime.migration] Running upgrade 99a18f68c13a -> 251e4854ef4c, 'followers'
```

3.5. 设置Gunicorn

新增`microblog.py`文件，用于作为主应用模块，用于`gunicorn`启动服务。

```
from app import create_app, db
from app.models import User, Post

app = create_app()

@app.shell_context_processor
def make_shell_context():
    return {'db': db, 'User': User, 'Post':Post}
```

通过gunicorn下启动Microblog项目服务。

- `-b`选项告诉gunicorn在哪里监听请求，`localhost:8000`表示在8000端口上监听了内部网络接口；
- `-w`选项配置gunicorn将运行多少*worker*。 拥有四个进程可以让应用程序同时处理多达四个客户端，这对于Web应用程序通常足以处理大量客户端请求，因为并非所有客户端都在不断请求内容。 根据服务器的RAM大小，可能需要调整worker数量，以免内存不足；
- `microblog:app`参数告诉gunicorn如何加载应用程序实例。 冒号前的名称是包含应用程序的模块，冒号后面的名称是此应用程序的名称。

```
(venv) [vagrant@microblog flask-mega-tutorial]$ gunicorn -b localhost:8000 -w 4
microblog:app
[2019-05-09 10:00:21 +0000] [5173] [INFO] Starting gunicorn 19.9.0
[2019-05-09 10:00:21 +0000] [5173] [INFO] Listening at: http://127.0.0.1:8000 (5173)
[2019-05-09 10:00:21 +0000] [5173] [INFO] Using worker: sync
[2019-05-09 10:00:21 +0000] [5176] [INFO] Booting worker with pid: 5176
[2019-05-09 10:00:21 +0000] [5177] [INFO] Booting worker with pid: 5177
[2019-05-09 10:00:21 +0000] [5179] [INFO] Booting worker with pid: 5179
[2019-05-09 10:00:21 +0000] [5181] [INFO] Booting worker with pid: 5181
[2019-05-09 10:00:23,271] INFO in logger: 博客已启动
[2019-05-09 10:00:23,358] INFO in logger: 博客已启动
[2019-05-09 10:00:23,402] INFO in logger: 博客已启动
[2019-05-09 10:00:23,491] INFO in logger: 博客已启动
```

3.6. 设置Supervisor

Supervisor使用配置文件定义它要监视什么程序以及如何在必要时重新启动它们。 配置文件必须存储在`/etc`目录中。新建`/etc/supervisord.d/microblog.ini`用于设置Microblog的配置。

`command`，`directory`和`user`设置告诉supervisor如何运行应用程序。 如果计算机启动或崩溃，`autostart`和`autorestart`设置会使microblog自动重新启动。 `stopasgroup`和`killasgroup`选项确保当supervisor需要停止应用程序来重新启动它时，它仍然会调度成顶级gunicorn进程的子进程。

```
[program:microblog]
command=/home/vagrant/projects/flask-mega-tutorial/venv/bin/gunicorn -b localhost:8000 -w 4
microblog:app
directory=/home/vagrant/projects/flask-mega-tutorial
user=vagrant
autostart=true
autorestart=true
stopasgroup=true
killasgroup=true
```

设置Supervisor开机自启，并启动服务。

```
(venv) [vagrant@microblog flask-mega-tutorial]$ systemctl enable supervisord
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-unit-files ===
Authentication is required to manage system service or unit files.
Authenticating as: root
Password:
==== AUTHENTICATION COMPLETE ===
==== AUTHENTICATING FOR org.freedesktop.systemd1.reload-daemon ===
Authentication is required to reload the systemd state.
Authenticating as: root
Password:
==== AUTHENTICATION COMPLETE ===

(venv) [vagrant@microblog flask-mega-tutorial]$ systemctl start supervisord
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to manage system services or units.
Authenticating as: root
Password:
==== AUTHENTICATION COMPLETE ===

(venv) [vagrant@microblog flask-mega-tutorial]$ systemctl status supervisord
● supervisord.service - Process Monitoring and Control Daemon
   Loaded: loaded (/usr/lib/systemd/system/supervisord.service; enabled; vendor preset:
disabled)
   Active: active (running) since Thu 2019-05-09 10:18:08 UTC; 58s ago
  Process: 5372 ExecStart=/usr/bin/supervisord -c /etc/supervisord.conf (code=exited,
status=0/SUCCESS)
 Main PID: 5373 (supervisord)
   CGroup: /system.slice/supervisord.service
           └─5373 /usr/bin/python /usr/bin/supervisord -c /etc/supervisord.conf

(venv) [vagrant@microblog flask-mega-tutorial]$ ps -ef|grep supervisord
root      5373     1  0 10:18 ?        00:00:00 /usr/bin/python /usr/bin/supervisord -c
/etc/supervisord.conf
vagrant   5376  4648  0 10:19 pts/0    00:00:00 grep --color=auto supervisord
```

编写配置文件后，必须重载supervisor服务的配置才能导入它，重启命令如下：

```
$ sudo supervisorctl reload
```

像这样，这个gunicorn web服务器就已经启动和运行，并处于监控之中：

```
(venv) [vagrant@microblog flask-mega-tutorial]$ ps -ef|grep gunicorn
vagrant   5412  5411  6 10:26 ?        00:00:00 /home/vagrant/projects/flask-mega-
tutorial/venv/bin/python3 /home/vagrant/projects/flask-mega-tutorial/venv/bin/gunicorn -b
localhost:8000 -w 4 microblog:app
vagrant   5415  5412 19 10:26 ?        00:00:00 /home/vagrant/projects/flask-mega-
tutorial/venv/bin/python3 /home/vagrant/projects/flask-mega-tutorial/venv/bin/gunicorn -b
localhost:8000 -w 4 microblog:app
vagrant   5416  5412 20 10:26 ?        00:00:00 /home/vagrant/projects/flask-mega-
tutorial/venv/bin/python3 /home/vagrant/projects/flask-mega-tutorial/venv/bin/gunicorn -b
localhost:8000 -w 4 microblog:app
vagrant   5418  5412 20 10:26 ?        00:00:00 /home/vagrant/projects/flask-mega-
tutorial/venv/bin/python3 /home/vagrant/projects/flask-mega-tutorial/venv/bin/gunicorn -b
localhost:8000 -w 4 microblog:app
vagrant   5420  5412 26 10:26 ?        00:00:00 /home/vagrant/projects/flask-mega-
tutorial/venv/bin/python3 /home/vagrant/projects/flask-mega-tutorial/venv/bin/gunicorn -b
localhost:8000 -w 4 microblog:app
vagrant   5424  4648  0 10:26 pts/0    00:00:00 grep --color=auto gunicorn
```

3.7. 启动nginx服务

创建`certs`目录，并在该目录下生创建一个*自签名SSL证书*，以此实现https服务启动。

```
(venv) [vagrant@microblog flask-mega-tutorial]$ mkdir certs
(venv) [vagrant@microblog flask-mega-tutorial]$ openssl req -new -newkey rsa:4096 -days 365
-nodes -x509 -keyout certs/key.pem -out certs/cert.pem
Generating a 4096 bit RSA private key
.....................................................++
.....................................................++
writing new private key to 'certs/key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:
```

修改nginx配置文件`/usr/local/nginx/conf/nginx.conf`。

```
user  root;
worker_processes  4;

#error_log  logs/error.log;
```

```
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid        logs/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       mime.types;
    default_type  application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    # 将access和error日志写入logs目录
    access_log  /home/vagrant/logs/microblog_access.log;
    error_log   /home/vagrant/logs/microblog_error.log;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

    server {
        # 监听80端口
        listen       80;
        server_name  localhost;
        location / {
            # 所有http请求均转为https请求
            return 301 https://$host$request_uri;
        }
    }

    # HTTPS server

    server {
        # 监听https的443端口
        listen       443 ssl;
        server_name  localhost;

        # 证书
        ssl_certificate      /home/vagrant/projects/flask-mega-tutorial/certs/cert.pem;
        ssl_certificate_key  /home/vagrant/projects/flask-mega-tutorial/certs/key.pem;

        ssl_session_cache    shared:SSL:1m;
```

```
        ssl_session_timeout  5m;

        ssl_ciphers  HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers  on;

        location / {
            # 将请求转发值gunicorn服务
            proxy_pass http://localhost:8000;
            proxy_redirect off;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        location /static {
            # 静态文件转发
            alias /home/vagrant/projects/flask-mega-tutorial/app/static;
            expires 30d;
        }
    }

}
```

创建`nginx`命令软连接。

```
sudo ln -s /usr/local/nginx/sbin/nginx /usr/bin/nginx
```

启动nginx服务命令：

```
[vagrant@microblog conf]$ sudo nginx
[vagrant@microblog conf]$ ps -ef|grep nginx
root      8279     1  0 05:27 ?        00:00:00 nginx: master process nginx
root      8479  8279  0 05:50 ?        00:00:00 nginx: worker process
root      8480  8279  0 05:50 ?        00:00:00 nginx: worker process
root      8481  8279  0 05:50 ?        00:00:00 nginx: worker process
root      8482  8279  0 05:50 ?        00:00:00 nginx: worker process
vagrant   8488  5399  0 05:54 pts/0    00:00:00 grep --color=auto nginx
```

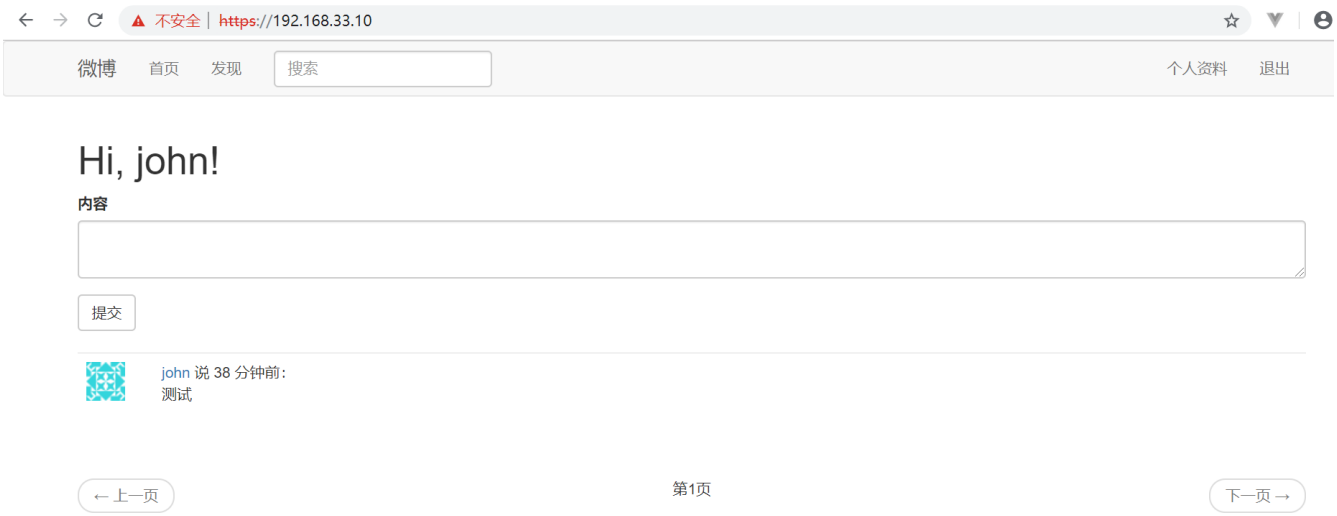
停止nginx服务命令：

```
sudo nginx -s stop
```

重启nginx命令：

```
sudo nginx -s reload
```

由于使用的是自签名证书，因此将收到来自Web浏览器的警告，你必须解除该警告，才能继续访问，访问结果如下：



4. 生产部署设置

生产部署，为了安全起见，可以通过一些配置，增加系统的安全性，设置如下：

- 免密登录

- 禁止root用户登录
- 禁止密码登录，只能通过免密方式登录
- 设置防火墙
- MySQL设置root用户密码（默认密码为空）

5. 部署应用更新操作

进行应用升级通常比重新启动服务器更为复杂。 可能需要应用数据库迁移或编译新的语言翻译，因此实际上，执行升级的过程通常涉及以下命令：

- 拉取最新代码

```
git pull
```

- 停止当前应用服务

```
sudo supervisorctl stop microblog
```

- 数据库迁移

```
flask db upgrade
```

- 编译语言翻译

```
flask translate compile
```

- 启动当前应用服务

```
sudo supervisorctl start microblog
```