

基于 Amazon AWS EC2 部署 Ghost 博客

一、实验简介

本次课程的主要内容就是介绍如何通过**在AWS EC2 上部署 Ghost v0.9.0** 来创建个人博客。

补充：Ghost 是基于 Node.js 的开源博客平台，由前 WordPress UI 部门主管 John O’Nolan 和 WordPress 高级工程师 Hannah Wolfe 创立，目的是为了给用户提供一种更加纯粹的内容写作与发布平台。

参考：

[Ghost 中文网](#)

1.1 知识点

使用 Ghost 搭建博客主要涉及的都是一些环境配置操作，本次课程将涉及以下知识点：

- 常见的 Linux shell 操作
- 编译安装 Node.js
- 配置安装 Ghost 服务
- 使用 nginx 反向代理

1.2 操作流程

以下是本次课程的主要操作流程：

1. 注册 AWS 并创建 EC2 服务器
2. 在 EC2 中配置 Node.js
3. 在 EC2 中配置安装 Ghost 服务
4. 在 EC2 中配置 nginx 和 pm2
5. 编写第一篇 Ghost 博客

1.3 操作环境

以下是本次课程的操作环境以及将用到的软件的版本

- 操作系统：Ubuntu 14.04 64bit
- Node.js：v4.2.2
- Ghost：v0.9.0
- nginx：v1.4.6

1.4 效果截图

这是部署成功之后，使用默认主题样式的博客首页。

up up day
Thoughts, stories and ideas.

我的第一篇 Ghost 文章

我的第一篇 Ghost 文章 这是我的第一篇 Ghost 文章，写于实验楼。 >>

 25 AUGUST 2016

Welcome to Ghost

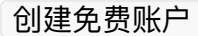
You're live! Nice. We've put together a little post to introduce you to the Ghost editor and get you started. You can manage your content by >>

 on Getting Started | 24 AUGUST 2016



二、创建并使用 AWS EC2

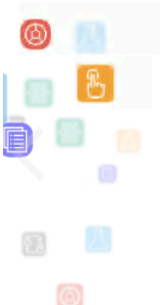
2.1 注册 AWS

进入 AWS 官网(<https://aws.amazon.com/cn/>)进行账户注册，点击  进行注册。已经有 AWS 账号的同学可跳过此步，直接登陆。

菜单



中文 (简体) | 我的账户 | 注册



几分钟内即可开始使用 AWS

了解更多 »

点击注册 AWS 账户

开始免费使用 AWS

创建免费账户

Amazon DynamoDB
25 GB 的存储和每月多达 2 千万个请求

查看 AWS 免费套餐详细信息 »



入门
了解如何在几分钟内开始使用 AWS



AWS 定价
优化可变或稳定工作负载的支出



AWS 免费套餐
获得免费上手体验 AWS 12 个月的机会



AWS 文档
帮助您快速入门的分步说明



补充：自 2006 年初起，亚马逊 AWS 开始在云中为各种规模的公司提供技术服务平台。利用亚马逊 AWS，软件开发人员可以轻松购买计算、存储、数据库和其他基于 Internet 的服务来支持其应用程序。开发人员能够灵活选择任何开发平台或编程环境，以便于其尝试解决问题。由于开发人员只需按使用量付费，无需前期资本支出，亚马逊 AWS 是向最终用户交付计算资源、保存的数据和其他应用程序的一种最经济划算的方式。

参考：

[AWS 中国](#)

在注册页面中，需要输入邮箱，姓名及密码：



创建您的账户

请填写以下信息来创建用于 AWS 以及 Amazon.com 的账户

请输入姓名：

请输入邮箱地址：

请再次输入邮件地址：

注意：我们将使用这个电子邮件地址与您取得联系

请输入密码：

请再次输入密码：

创建帐户

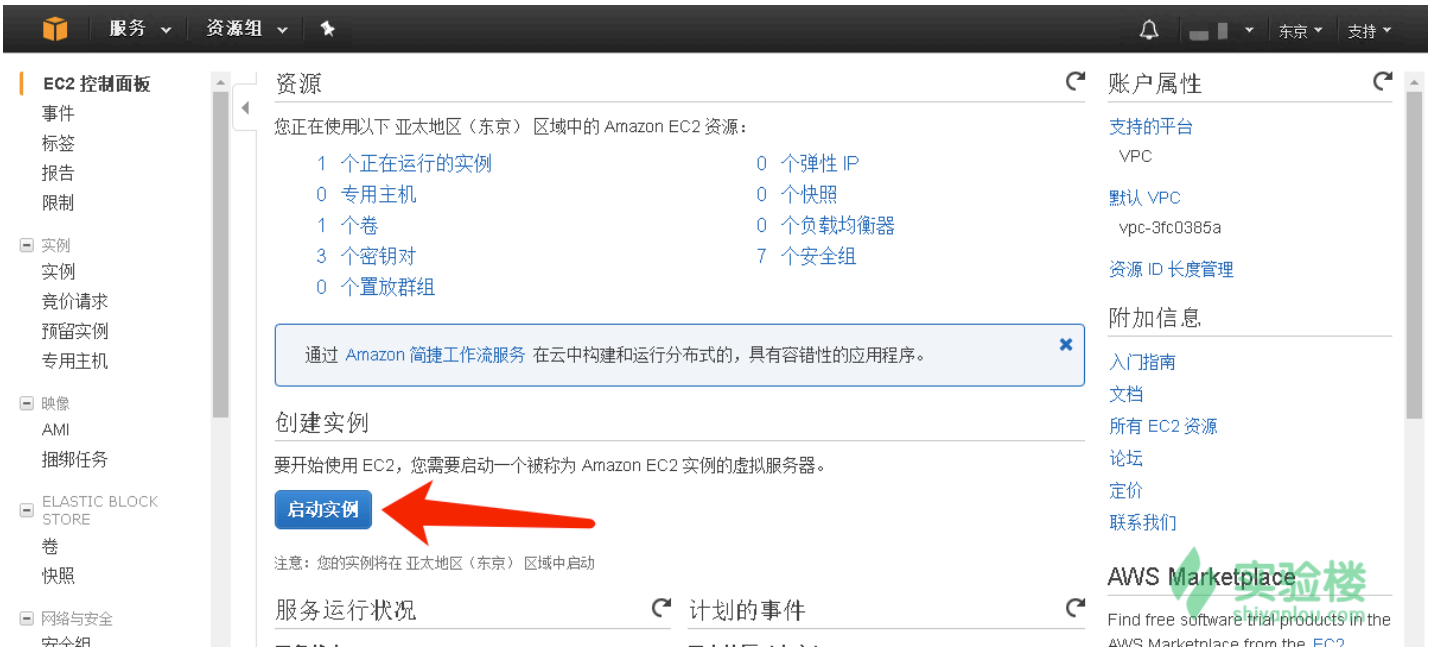


2.2 创建 AWS EC2

注册之后，登陆 AWS，登陆后会直接进入控制台界面，点击右上角选择创建 EC2 服务器所在的区域，可以选择世界各地的数据中心。



然后点击左上角的 服务 ，选择其中的 EC2 进入到 EC2 控制面板，在控制面板点击 启动实例 开始创建 EC2。



启动实例的创建过程需要多个步骤，每个步骤都增加新的配置内容，最后通过审核确认后创建。

步骤一选择一个 Amazon 系统映像，我们选择 Ubuntu 14.04 LTS 64 位 HVM SSD 这个 AMI。

服务▼资源组▼★

1. 选择 AMI2. 选择实例类型3. 配置实例4. 添加存储5. Add Tags6. 配置安全组7. 审核

步骤 1: 选择一个Amazon 系统映像(AMI)

取消并退出

AMI tools preinstalled, Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.9.1 available.

根设备类型: ebs 虚拟化类型: paravirtual

Microsoft Windows Server 2003 R2 Base - ami-11c3b176 (64 位) / ami-3ac3b15d (32 位)

选择

Windows

Microsoft Windows 2003 R2 SP2 Datacenter edition. [English]

符合条件的免费套餐

根设备类型: ebs 虚拟化类型: hvm

64 位 32 位

Amazon Linux AMI 2016.09.1 (PV) - ami-bdd2abda

选择

Amazon Linux

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

符合条件的免费套餐

根设备类型: ebs 虚拟化类型: paravirtual

64 位

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-8fccbe08

选择

Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

符合条件的免费套餐

根设备类型: ebs 虚拟化类型: hvm

64 位

实验楼shiyianlou.com

实例类型可以根据自己预估的访问量选择配置：

服务▼资源组▼★

1. 选择 AMI2. 选择实例类型3. 配置实例4. 添加存储5. Add Tags6. 配置安全组7. 审核

步骤 2: 选择一个实例类型

Amazon EC2 提供多种经过优化，适用于不同使用案例的实例类型以供选择。实例就是可以运行应用程序的虚拟服务器。它们由 CPU、内存、存储和网络容量组成不同的组合，可让您灵活地为您的应用程序选择适当的资源组合。[了解更多](#) 有关实例类型以及这些类型如何满足您的计算需求的信息。

筛选条件: 所有实例类型▼最新一代▼显示/隐藏列

当前选择的实例类型: t2.micro (变量 ECU, 1 vCPU, 2.5 GHz, Intel Xeon Family, 1 GiB 内存, 仅限于 EBS)

	系列	类型	vCPU	内存 (GiB)	实例存储 (GB)	可用的优化 EBS	网络性能	IPv6 Support
☐	通用型	t2.nano	1	0.5	仅限于 EBS	-	低到中等	是
☒	通用型	t2.micro 符合条件的免费套餐	1	1	仅限于 EBS	-	低到中等	是
☐	通用型	t2.small	1	2	仅限于 EBS	-	低到中等	是
☐	通用型	t2.medium	2	4	仅限于 EBS	-	低到中等	是
☐	通用型	t2.large	2	8	仅限于 EBS	-	低到中等	是

取消

上一步

审核和启动

下一步: 配置实例详细信息

实验楼shiyianlou.com

后续步骤都可以使用默认的选项：

步骤 3: 配置实例详细信息

配置实例以便满足您的需求。您可以从同一 AMI 上启动多个实例，请求竞价型实例以利用其低价优势，向实例分配访问管理角色等等。

实例的数量	1	启动至 Auto Scaling 组
购买选项	☐ 请求竞价型实例	
网络	vpc-3fc0385a (default)	新建 VPC
子网	无首选项(任何可用区的默认子网)	新建子网
自动分配公有 IP	使用子网设置 (启用)	
IAM 角色	无	创建新的 IAM 角色
关闭操作	停止	
启用终止保护	☐ 防止意外终止	
监控	☐ 启用 CloudWatch 详细监控 将收取额外费用。	
租赁	共享 - 运行共享硬件实例	

[取消](#) [上一步](#) [审核和启动](#) [下一步: 添加存储](#)

步骤 4: 添加存储

您的实例将使用以下存储设备设置启动。您可以将其他 EBS 卷和实例存储卷连接到您的实例，或编辑根卷的设置。您还可以在启动实例后连接其他 EBS 卷而非实例存储卷。[了解更多](#) 关于 Amazon EC2 中的存储选项。

卷类型	设备	快照	大小 (GiB)	卷类型	IOPS	吞吐量 (MB/s)	终止时删除	加密
根目录	/dev/sda1	snap-031da7705fc51659e	30	General Purpose SSD (GP2)	100 / 3000	不适用	☒	未加密

[添加新卷](#)

有资格使用免费套餐的客户最多可获得 30 GB 的 EBS 通用型 (SSD) 或磁存储空间。[了解更多](#) 有关免费使用套餐资格和使用限制的信息。

步骤 5: Add Tags

标签包含一个区分大小写的键值对。例如，您可以定义一个键为"Name"且值为"Webserver"的标签。[了解更多](#) 有关标记 Amazon EC2 资源的信息。

密钥 (最多 127 个字符)	值 (最多 255 个字符)
Name	

[Add another tag](#) (最多 50 个标签)

安全组配置位置需要打开 Ghost 博客将使用的端口以及 SSH 使用的 22 端口：

步骤 6: 配置安全组

安全组是一组防火墙规则，用于控制针对您的实例的流量。在此页面上，您可以添加规则来允许到达您的实例的特定流量。例如，如果您希望设置一个 Web 服务器，并允许 Internet 流量到达您的实例，请添加相应的规则来允许不受限制地访问 HTTP 和 HTTPS 端口。您可以创建一个新的安全组或从下面选择一个现有的安全组。[了解更多](#) 有关 Amazon EC2 安全组的信息。

分配安全组: ☒ 创建一个新安全组

☐ 选择一个现有的安全组

安全组名称: launch-wizard-2

描述: launch-wizard-2 created 2017-02-04T14:46:10.483+08:00

类型	协议	端口范围	来源
SSH	TCP	22	自定义 0.0.0.0/0
HTTP	TCP	80	自定义 0.0.0.0/0, :::/0
自定义 TCP 规则	TCP	2368	自定义 0.0.0.0/0

添加规则



警告

设置为 0.0.0.0/0 的源规则允许所有 IP 地址访问您的接口。我们建议将安全组规则设置为仅允许从已知的 IP 地址进行访问。



取消

上一步

审核和启动

最后审核并启动 EC2。

2.3 SSH 登陆 AWS EC2

EC2 可以使用 SSH 密钥登陆。拥有一台 EC2 服务器之后，通过 AWS 控制台下载证书，然后使用常用的 SSH 客户端，例如 putty 登陆 EC2 服务器。

连接到您的实例

我希望连接到 ☒ 一个独立的 SSH 客户端 ☐ 直接从我的浏览器连接的 Java SSH 客户端 (需要安装 Java)

要访问您的实例:

1. 打开 SSH 客户端。(了解如何 [使用 PuTTY 连接](#))
2. 查找您的私有密钥文件 (ss.pem)。向导会自动检测您用于启动实例的密钥。
3. 您的密钥必须不公开可见，SSH 才能工作。如果需要，请使用此命令：

```
chmod 400 ss.pem
```
4. 通过其公有 DNS 连接到您的实例:

```
ec2-99-240-60-ap-northeast-1.amazonaws.com
```

示例:

```
ssh -i "ss.pem" ubuntu@ec2-99-240-60-ap-northeast-1.amazonaws.com
```

请注意，在大多数情况下，上述用户名都是正确的，但请确保您已阅读了 AMI 使用说明以确定 AMI 所有人没有更改默认的 AMI 用户名。

如果您在连接到您的实例方面需要任何帮助，请参阅我们的 [链接文档](#)。

关闭

后续的实验操作都是在 SSH 登陆 EC2 服务器后的命令行中执行的。

三、配置 Node.js

3.1 下载 Node.js

从 Ghost 0.9.0 开始，Ghost 官方开始推荐使用 `Node.js v4.2x`，同时还支持 `Node.js v0.10.x` 和 `v0.12.x`，所以在安装 `Node.js` 的时候注意版本的选择，这里我们采用 `v4.2.2` 的版本。

这里我们采用下载源码包自己进行编译的方式进行安装，`v4.2.2` 版本的源码地址为：

```
https://nodejs.org/dist/v4.2.2/node-v4.2.2.tar.gz
```

使用 `wget` 指令下载 `Node.js`。

```
$ wget https://nodejs.org/dist/v4.2.2/node-v4.2.2.tar.gz
```

如果提示没有安装 `wget`，可以通过 `sudo apt-get install wget` 的方式安装 `wget`。

另外，我们上方提供的是 Node 官方源下载地址，由于国内网络原因可能导致下载较慢或者下载失败，那么你可以从下方这个镜像进行下载。

```
https://npm.taobao.org/mirrors/node/v4.2.2/node-v4.2.2.tar.gz
```

补充：`wget` 是 Linux 系统下的一个下载文件的工具，它支持 HTTP, HTTPS 和 FTP 协议，具有支持递归下载、支持代理服务器、自动下载等特性。可以使用它下载软件或者从远程服务器恢复备份。

参考：

[Wget-维基百科](#) [Ghost 0.9.0 官方文档](#)

[Node.js 官网](#)

3.2 安装 Node.js

执行以下命令解压缩压缩包 `node-v4.2.2.tar.gz`。

```
$ tar -xvf node-v4.2.2.tar.gz
```

执行命令 `$ cd node-v4.2.2` 进入文件夹 `node-v4.2.2/`，使用 `$ tree -L 1` 命令查看当前目录下深度为 1 的文件列表树。


```
root@kali:~# cd node-v4.2.2/  
root@kali:~/node-v4.2.2# tree -L 1
```

```
.  
├── android-configure  
├── AUTHORS  
├── benchmark  
├── BSDmakefile  
├── CHANGELOG.md  
├── COLLABORATOR_GUIDE.md  
├── common.gypi  
├── config.gypi  
├── config.mk  
├── configure  
├── CONTRIBUTING.md  
├── deps  
├── doc  
├── GOVERNANCE.md  
├── icu_config.gypi  
├── lib  
├── LICENSE  
├── Makefile  
├── Makefile.build  
├── node -> out/Release/node  
├── node.gyp  
├── out  
├── README.md  
├── ROADMAP.md  
├── src  
├── test  
├── tools  
├── vcbuild.bat  
└── WORKING_GROUPS.md
```

```
8 directories, 21 files
```



我们发现这里边有许多文件和文件夹，或许你不清楚这些文件都是干什么用的，但是没关系，接下来的操作只涉及里边的很少一部分文件，所以我们也不需要知道每个文件的具体作用。

补充：Linux 的 tree 指令可用于以树状图列出目录的内容。

参考：

[Linux tree](#)

接下来再执行 `$ sudo ./configure` 命令，这条指令将会检测系统内核和环境，生成所需的中间目录以及解析相关的参数，再利用这些参数生成 C 源代码文件和 Makefile 等文件。

```
root@...:~/node-v4.2.2# ./configure
creating ./icu_config.gypi
{ 'target_defaults': { 'cflags': [],
                        'default_configuration': 'Release',
                        'defines': [],
                        'include_dirs': [],
                        'libraries': []},
  'variables': { 'asan': 0,
                  'gas_version': '2.24',
                  'host_arch': 'x64',
                  'icu_small': 'false',
                  'node_byteorder': 'little',
                  'node_install_npm': 'true',
                  'node_prefix': '/usr/local',
                  'node_release_urlbase': '',
                  'node_shared_http_parser': 'false',
                  'node_shared_libuv': 'false',
                  'node_shared_openssl': 'false',
                  'node_shared_zlib': 'false',
                  'node_tag': '',
                  'node_use_dtrace': 'false',
                  'node_use_etw': 'false',
                  'node_use_lttng': 'false',
                  'node_use_openssl': 'true',
                  'node_use_perfctr': 'false',
                  'openssl_fips': '',
                  'openssl_no_asm': 0,
                  'python': '/usr/bin/python',
                  'target_arch': 'x64',
                  'uv_parent_path': '/deps/uv/',
                  'uv_use_dtrace': 'false',
                  'v8_enable_gdbjit': 0,
                  'v8_enable_il8n_support': 0,
                  'v8_no_strict_aliasing': 1,
                  'v8_optimized_debug': 0,
                  'v8_random_seed': 0,
                  'v8_use_snapshot': 1,
                  'want_separate_host_toolset': 0}}
creating ./config.gypi
creating ./config.mk
```

然后再执行 `$ sudo make` 命令，该命令会利用 Makefile 文件提供的信息对项目文件进行编译最终生成可执行文件。

温馨提示：进行 `make` 的过程会有点慢，需要耐心等待。

`make` 指令执行完毕之后再执行 `$ sudo make install`，这条指令将会根据 `configure` 执行时的参数将 `Node.js` 部署到指定的安装目录，包括相关目录的建立和配置文件的复制等。

至此我们的 `Node.js` 安装就算完成了，如果要查看 `Node.js` 是否安装成功，可以执行 `node -v` 进行查看。如果收到类似下图的版本号回复，则说明成功安装。

```
root@...:~/node-v4.2.2# node -v
v4.2.2
```

四、配置安装 Ghost

创建并进入文件夹 `shiyancelou_blog/` 。

```
$ mkdir shiyancelou_blog  
  
$ cd shiyancelou_blog
```

下载 `Ghost` 。

```
$ wget https://ghost.org/zip/ghost-latest.zip
```

```
root@shiyancelou_blog:~/shiyancelou_blog# wget https://ghost.org/zip/ghost-latest.zip  
--2016-08-24 16:05:59-- https://ghost.org/zip/ghost-latest.zip  
Resolving ghost.org (ghost.org)... 104.16.85.186, 104.16.83.186, 104.16.81.186, ...  
Connecting to ghost.org (ghost.org)|104.16.85.186|:443... connected.  
HTTP request sent, awaiting response... 302 Found  
Location: http://d36u7lo2kegjl.cloudfront.net/archives/ghost-0.9.0.zip [following]  
--2016-08-24 16:06:00-- http://d36u7lo2kegjl.cloudfront.net/archives/ghost-0.9.0.zip  
Resolving d36u7lo2kegjl.cloudfront.net (d36u7lo2kegjl.cloudfront.net)... 52.84.212.218, 52.84.212.244, 52.84.212.14, ...  
Connecting to d36u7lo2kegjl.cloudfront.net (d36u7lo2kegjl.cloudfront.net)|52.84.212.218|:80... failed: Connection timed out.  
Connecting to d36u7lo2kegjl.cloudfront.net (d36u7lo2kegjl.cloudfront.net)|52.84.212.244|:80... connected.  
HTTP request sent, awaiting response... 301 Moved Permanently  
Location: https://ghost.org/archives/ghost-0.9.0.zip [following]  
--2016-08-24 16:08:08-- https://ghost.org/archives/ghost-0.9.0.zip  
Connecting to ghost.org (ghost.org)|104.16.85.186|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 3884366 (3.7M) [application/zip]  
Saving to: 'ghost-latest.zip'  
  
100%[=====>] 3,884,366 82.3KB/s in 34s  
  
2016-08-24 16:08:43 (111 KB/s) - 'ghost-latest.zip' saved [3884366/3884366]  
  
root@shiyancelou_blog:~/shiyancelou_blog#
```



下载并安装 `zip` 的解压缩软件 `unzip` 。

```
$ sudo apt-get install unzip
```

对刚下载的 `ghost-latest.zip` 进行解压。

```
$ unzip -uo ghost-latest.zip -d ghost
```

进入刚刚解压缩所得的 `ghost` 文件夹。

```
$ cd ghost
```

使用 `npm` 安装 `ghost`，注意 `production` 前面是两横线 `--` 。

```
$ npm install --production
```

温馨提示：`npm install` 这一步操作需要下载安装一些依赖包，而由于国内网络原因可能导致这个过程安装失败，那么可以试一下安装这个版本 [Ghost v0.7.4 中文版](#)。这个版本虽然要比实验的 v0.9.0

来的低一些，但是不影响后续操作，另外这个已经进行汉化，很适合那些不是很喜欢看英文的同学，此外最重要的是这个版本已经集成了依赖包，从而免去了 `npm install` 的过程，下载下来解压缩之后即可用。

获取方式：

```
$ wget http://dl.ghostchina.com/Ghost-0.7.4-zh-full.zip
```

之后执行 `$ npm start` 。

看到后台输出 `Listening on 127.0.0.1:2368`，说明 Ghost 已经正常安装和运行了。

```
Ghost is running in development...
Listening on 127.0.0.1:2368
Url configured as: http://localhost:2368
Ctrl+C to shut down
```

补充：这里可能有人会奇怪为什么 Ghost 默认在 `development` 模式运行而执行 `npm install --production` 却是安装的 `production` 版本。如果在安装 Ghost 时不包含 `--production`，确实不会产生什么问题，但它会安装额外的仅仅是对想要开发 Ghost 核心的人有用的软件包。这就需要我们有一个特定的包，可以使用 `npm install -g grunt-cli` 命令安装在全局中，但是如果只是想作为一个博客运行 Ghost 的话，它是不是必需的。

参考：

[Ghost 使用指南](#)

接下来修改 `ghost` 目录下的 `config.js`，`config.js` 文件中的 `config` 配置的前两项分别为 `production` 和 `development`，它们分别设定了 Ghost 运行的两大模式，其中 `production` 是生产模式，而 `development` 是开发模式，也是默认的运行模式。两种模式并没有存在太大差别，只是这样环境可以允许我们在可能运行的不同的模式中创建不同的配置，通常意义上来讲 `development` 用于 Ghost 调试和开发，而当我们公开运行 Ghost 的时候选择 `production` 模式。以下我们只要修改 `production` 的内容即可。

注意：`config.js` 只有在第一次运行 Ghost 之后才会生成，这个文件里边的信息指定了 Ghost 运行的端口号，采用的数据库等信息。如果你尚未第一次运行 Ghost，那么目录底下将不会有 `config.js` 这个文件，但是目录底下已经有了文件 `config.example.js`，这个文件内容和 `config.js` 一样，所以可以通过运行指令 `$ cp config.example.js config.js` 自行创建，再修改。

运行命令 `$ vim config.js` 打开文件，修改的部分内容如下：

```

production: {
  url: '输入你的域名或者ip(格式: http://xxx.xxx.xxx.xxx)',
  mail: {}, //这一项可配置可不配置
  database: {
    client: 'sqlite3', // 默认使用自带的 sqlite3 数据库
    connection: {
      filename: path.join(__dirname, '/content/data/ghost.db')
    },
    debug: false
  },

  server: {
    host: '0.0.0.0', # 将此处修改为 0.0.0.0 这样程序才能在公网监听
    port: '2368'
  }
},

```

如果不修改数据库的话，只要修改 `url` 和 `host` 便行了。

补充：如果你想要使用 mysql 的话，`database` 那一项的配置方法如下：

```

database: {
  client: 'mysql',
  connection: {
    host: 'localhost',
    user: 'xxx', // 用户名
    password: 'xxxxxx', // 密码
    database: 'xxx', // 数据库名
    charset: 'utf8'
  },
  debug: false
},

```

然后使用 `npm` 命令来试运行，这里使用了 `NODE_ENV=production` 指定在 `production` 模式下运行 Ghost。

```
$ NODE_ENV=production npm start index.js
```

```

root@xxx.xxx.xxx:~/shiyancelou_blog/ghost# NODE_ENV=production npm start index.js

> ghost@0.9.0 start /root/shiyancelou_blog/ghost
> node index "index.js"

Migrations: Creating tables...
Migrations: Creating table: posts
Migrations: Creating table: users
Migrations: Creating table: roles
Migrations: Creating table: roles_users
Migrations: Creating table: permissions
Migrations: Creating table: permissions_users
Migrations: Creating table: permissions_roles
Migrations: Creating table: permissions_apps
Migrations: Creating table: settings
Migrations: Creating table: tags
Migrations: Creating table: posts_tags
Migrations: Creating table: apps
Migrations: Creating table: app_settings
Migrations: Creating table: app_fields
Migrations: Creating table: clients
Migrations: Creating table: client_trusted_domains
Migrations: Creating table: accesstokens
Migrations: Creating table: refreshtokens
Migrations: Creating table: subscribers
Migrations: Running fixture populations
Migrations: Creating owner
Ghost is running in production...
Your blog is now available on http://xxx.xxx.xxx:2368
Ctrl+C to shut down
127.0.0.1 - - [24/Aug/2016:03:17:34 +0000] "GET / HTTP/1.0" 200 - "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36"
127.0.0.1 - - [24/Aug/2016:03:17:35 +0000] "GET /assets/css/screen.css?v=49e212992a HTTP/1.0" 200 - "http://xxx.xxx.xxx:2368/" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36"
127.0.0.1 - - [24/Aug/2016:03:17:35 +0000] "GET /assets/js/jquery.fitvids.js?v=49e212992a HTTP/1.0" 200 - "http://xxx.xxx.xxx:2368/" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36"
127.0.0.1 - - [24/Aug/2016:03:17:35 +0000] "GET /assets/js/index.js?v=49e212992a HTTP/1.0" 200 - "http://xxx.xxx.xxx:2368/" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36"
127.0.0.1 - - [24/Aug/2016:03:17:37 +0000] "GET /favicon.ico HTTP/1.0" 200 - "http://xxx.xxx.xxx:2368/" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36"

```

在浏览器地址栏中输入 `http://xxx.xxx.xxx:2368` 进行访问，如果出现如下界面则说明配置成功。

温馨提示：访问地址中 `xxx` 部分对应你的云服务器的 ip 地址。

Ghost

Just a blogging platform.

Welcome to Ghost

You're live! Nice. We've put together a little post to introduce you to the Ghost editor and get you started. You can manage your content by »

Ghost Owner on Getting Started | 24 AUGUST 2016

Page 1 of 1

Ghost © 2016

Proudly published with Ghost
 实验楼
shiyancelou.com

五、配置 nginx 和 pm2

通过以上的几步操作我们已经可以通过 ip 地址加端口号的形式来访问 Ghost 页面了，但是还存在点问题。

1. 我们现在是通过 SSH 终端来启动云服务器中的 Ghost 进程，但是一旦把 SSH 终端断开，Ghost 进程也将被终止，从而无法进行访问。
2. Ghost 服务一旦挂掉了就无法自动重启。
3. 另外目前这样无法实现通过域名直接对博客进行访问。

所以接下来我们还要请出以下的两个工具来帮助我们解决这些问题。

5.1 使用 pm2 让 Ghost 一直在线

其实要让 Ghost 程序能在 SSH 终端关闭之后还能继续运行的方法挺多的，如下。

1. `forever`：一个 `node.js` 的进程管理模块，以后台任务运行 Ghost。
2. `pm2`：与 `forever` 类似，是一个进程管理模块，也能保护 Ghost 后台进程。
3. Supervisor：进程控制系统，允许系统在启动的时候无需初始化脚本就能运行 Ghost。

这里我们选用 `pm2` 进行进程保护，配置方法如下。

首先使用 `cd` 指令进入到之前 `ghost` 的安装目录下。然后执行以下指令安装 `pm2`。

```
$ npm install pm2 -g
```

```
root@i:~# npm install pm2 -g
npm ERR! git clone --template=/root/.npm/_git-remotes/_templates --mirror http://ikt.pm2.io/ikt.git /root/.npm/_git-remotes/http-ikt-pm2-i
o-ikt-git-4d23bf0d: undefined
npm ERR! git clone --template=/root/.npm/_git-remotes/_templates --mirror http://ikt.pm2.io/ikt.git /root/.npm/_git-remotes/http-ikt-pm2-i
o-ikt-git-4d23bf0d: undefined
npm WARN optional dep failed, continuing ikt@git+http://ikt.pm2.io/ikt.git#master
npm WARN optional dep failed, continuing fsevents@1.0.14
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
/usr/local/bin/pm2 -> /usr/local/lib/node_modules/pm2/bin/pm2
/usr/local/bin/pm2-dev -> /usr/local/lib/node_modules/pm2/bin/pm2-dev
pm2@1.1.3 /usr/local/lib/node_modules/pm2
├── eventemitter2@0.4.14
├── async@1.5.2
├── pidusage@1.0.4
├── semver@5.1.1
├── vizion@0.2.13
├── coffee-script@1.10.0
├── shelljs@0.6.0
├── commander@2.9.0 (graceful-readlink@1.0.1)
├── pmx@0.6.3 (json-stringify-safe@5.0.1)
├── cli-table@0.3.1 (colors@1.0.3)
├── nssocket@0.6.0 (lazy@1.0.11)
├── debug@2.2.0 (ms@0.7.1)
├── moment@2.14.1
├── mkdirp@0.5.1 (minimist@0.0.8)
├── chalk@1.1.1 (escape-string-regexp@1.0.5, supports-color@2.0.0, ansi-styles@2.2.1, has-ansi@2.0.0, strip-ansi@3.0.1)
├── pm2-multimeter@0.1.2 (charm@0.1.2)
├── pm2-deploy@0.2.1 (async@1.4.2, tv4@1.0.18)
├── pm2-axon@2.0.11 (amp-message@0.1.2, escape-regexp@0.0.1, amp@0.3.1, configurable@0.0.1)
├── pm2-axon-rpc@0.3.6 (json-stringify-safe@5.0.1, commander@1.0.5)
├── source-map-support@0.4.0 (source-map@0.1.32)
├── cron@1.1.0 (moment-timezone@0.3.1)
├── yamls@0.2.7 (argparse@0.1.16, glob@4.5.3)
├── chokidar@1.4.3 (path-is-absolute@1.0.0, inherits@2.0.1, glob-parent@2.0.0, async-each@1.0.0, is-glob@2.0.1, is-binary-path@1.0.1, read
dirp@2.1.0, anymatch@1.3.0)
```

使用 `pm2` 守护进程，以 `production` 模式运行 `ghost`，并且将此进程命名为 `shiyancelou` 方便之后管理。

```
NODE_ENV=production pm2 start index.js --name "shiyancelou"
```

```
root@i:~/shiyancelou_blog/ghost_new# NODE_ENV=production pm2 start index.js --name "shiyancelou"
[PM2] Starting index.js in fork_mode (1 instance)
[PM2] Done.
```

App name	id	mode	pid	status	restart	uptime	memory	watching
ghost	0	fork	0	stopped	0	0	0 B	disabled
shiyancelou	1	fork	2101	online	0	0s	6.883 MB	disabled

Use `pm2 show <id|name>` to get more details about an app

设定为开机自动启动。

```
$ pm2 startup ubuntu
```

```
$ pm2 save
```

温馨提示：如果你是 CentOS 的系统，可以写成 `pm2 startup centos`。


```
root@i[REDACTED]:~/shiyancelou_blog/ghost# pm2 startup ubuntu
[PM2] Generating system init script in /etc/init.d/pm2-init.sh
[PM2] Making script booting at startup...
[PM2] -ubuntu- Using the command:
    su -c "chmod +x /etc/init.d/pm2-init.sh && update-rc.d pm2-init.sh defaults"

Adding system startup for /etc/init.d/pm2-init.sh ...
/etc/rc0.d/K20pm2-init.sh -> ../init.d/pm2-init.sh
/etc/rc1.d/K20pm2-init.sh -> ../init.d/pm2-init.sh
/etc/rc6.d/K20pm2-init.sh -> ../init.d/pm2-init.sh
/etc/rc2.d/S20pm2-init.sh -> ../init.d/pm2-init.sh
/etc/rc3.d/S20pm2-init.sh -> ../init.d/pm2-init.sh
/etc/rc4.d/S20pm2-init.sh -> ../init.d/pm2-init.sh
/etc/rc5.d/S20pm2-init.sh -> ../init.d/pm2-init.sh
[PM2] Done.
```



```
root@i[REDACTED]:~/shiyancelou_blog/ghost# pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /root/.pm2/dump.pm2
root@i[REDACTED]:~/shiyancelou_blog/ghost#
```

到此为止我们的 Ghost 服务程序就能在云服务器启动的时候也自动加载了，接下来将解决通过域名进行访问的问题。

5.2 使用 nginx 进行反向代理

事实上到目前为止，如果想通过域名或者 ip 直接访问 Ghost 博客是无法访问的，这时候就需要配置 nginx 进行反向代理，将对域名或者 ip 的请求转发给 Ghost 服务，nginx 安装配置流程如下。

安装 `nginx` 。

```
$ sudo apt-get install nginx
```

然后配置站点信息。

在目录 `/etc/nginx/sites-available` 下创建 `shiyancelou_blog.conf` 文件。

```
$ vim /etc/nginx/sites-available/shiyancelou_blog.conf
```

再把以下配置信息复制进 `shiyancelou_blog.conf` 文件中。

```
server {  
    listen 80; // 监听80端口  
    server_name http://xxx.xxx.xxx.xxx; // 输入你的域名或者IP  
  
    location / {  
        proxy_set_header    X-Real-IP $remote_addr;  
        proxy_set_header    Host      $http_host;  
        proxy_pass            http://127.0.0.1:2368;  
    }  
}
```

以上内容，只需要修改 `server_name` 一项便可。

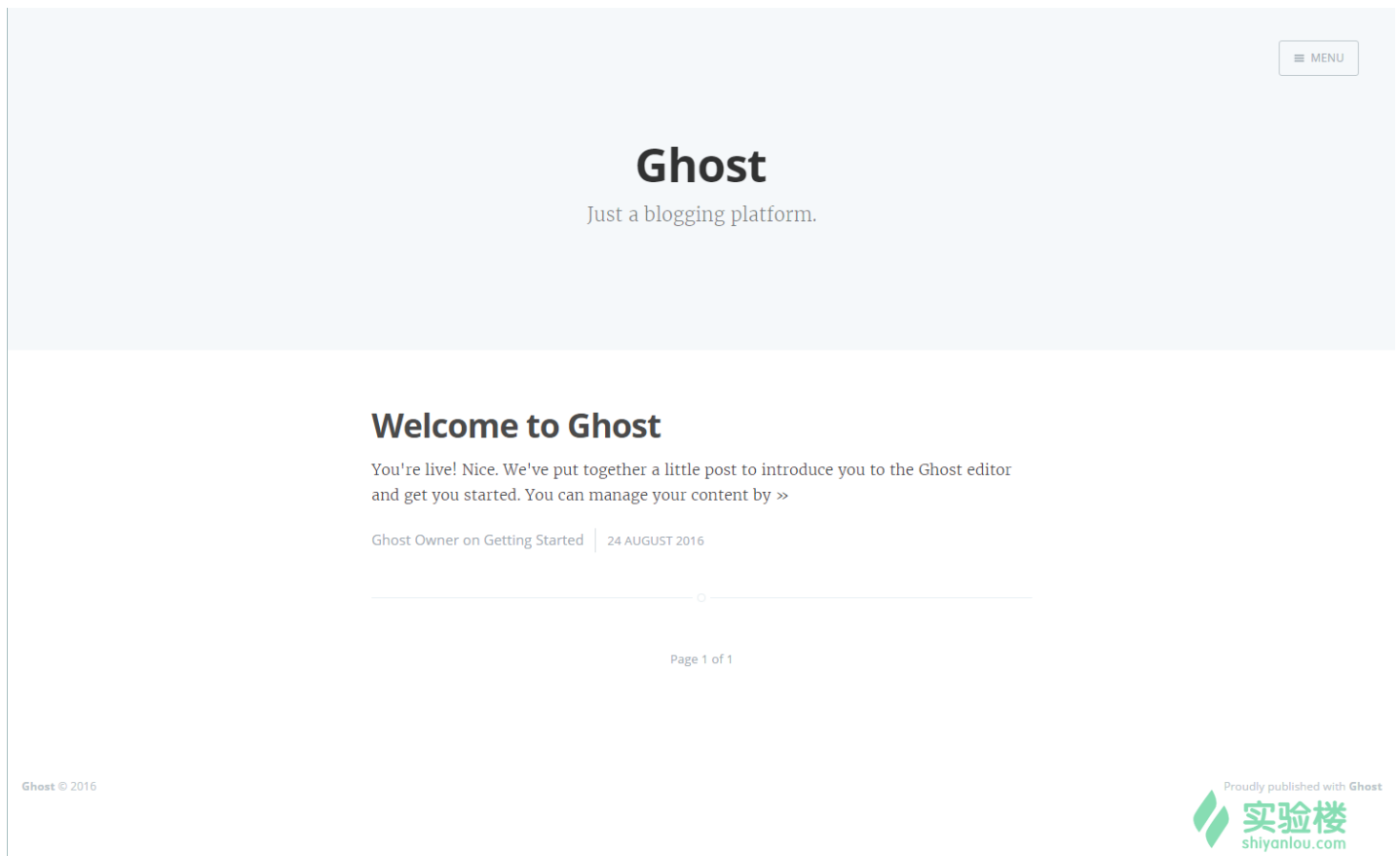
再将 `shiyanolou_blog.conf` 的配置文件软链接到 `sites-enabled` 文件夹下。

```
$ sudo ln -s /etc/nginx/sites-available/shiyanolou_blog.conf /etc/nginx/sites-enabled/shiyanolou_blog.conf
```

开启 `nginx` 。

```
$ sudo service nginx start
```

然后就可以在浏览器中输入你的域名或者 ip 进行访问，如果显示了如下页面则说明配置成功。



六、编写我的第一篇 Ghost 博客

在首次登录后台的时候首先需要注册，访问 `<your URL>/ghost/signup` 进行相关注册。之后就可以登录后台，编写博客了。

✓

2

3

Create your account



Email address



Full name



Password



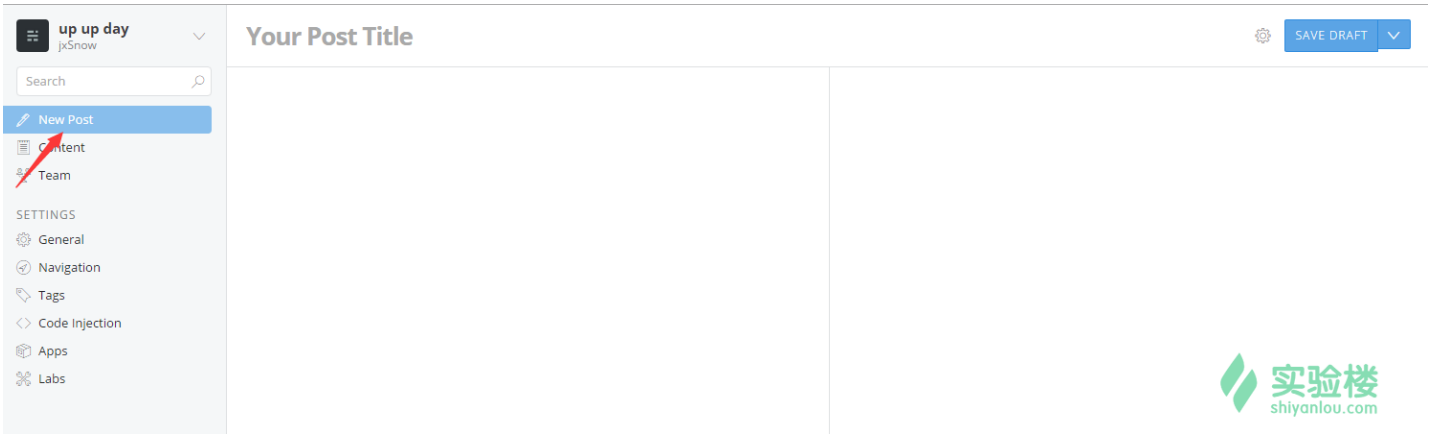
Blog title



LAST STEP: INVITE YOUR TEAM

实验楼
shiyancelou.com

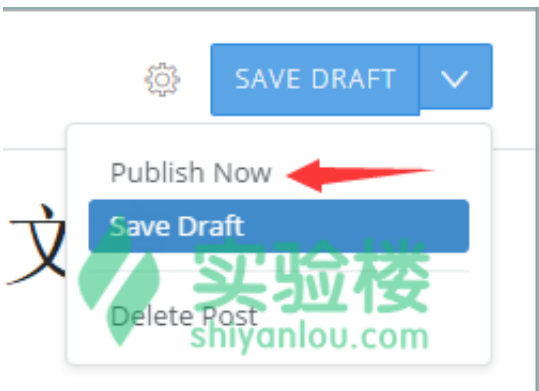
进入后台，点击「New Post」，进入文章编辑。



Ghost 博客编写支持 Markdown 语法，随便写点内容，然后点击「SAVE DRAFT」进行保存。



如果要发布文章或者删除文章可以点击「SAVE DRAFT」旁边的箭头，从下拉选项中选择所需的操作。



然后访问主页你自己的域名或者 ip 就可以看到博客中最新发布文章了。

up up day

Thoughts, stories and ideas.

我的第一篇 Ghost 文章

我的第一篇 Ghost 文章 这是我的第一篇 Ghost 文章，写于实验楼。 >>

25 AUGUST 2016

Welcome to Ghost

You're live! Nice. We've put together a little post to introduce you to the Ghost editor and get you started. You can manage your content by >>

on Getting Started | 24 AUGUST 2016



七、实验总结

本次课程我们主要学习了如何在AWS EC2 上部署 Ghost 博客，这门课程内容比较简单，不过希望同学们在学习这门课程之后能创建一个自己的个人博客并好好经营，将平时学习的收获或者生活的感言都记录在上。

温馨提示：在实验过程中有不懂的问题欢迎提出进行交流。另外想要获得更多的关于 Ghost 主题设置与 Ghost 博客管理相关的内容可以参考 [Ghost 中文网](#) 的文档。

参考资料

- [Ghost 中文网](#)

- [Ghost 快捷手册](#)
- [AWS官网](#)
- [Node.js 官网](#)