

云服务器

一. 购买云服务器

1.1. 注册阿里云的账号

云服务器我们可以有很多的选择：阿里云、腾讯云、华为云。

- 目前在公司使用比较多的是阿里云；
- 我自己之前也一直使用阿里云，也在使用腾讯云；
- 之前华为云也有找我帮忙推广他们的活动；

但是在我们的课程中，我选择目前使用更加广泛的阿里云来讲解：

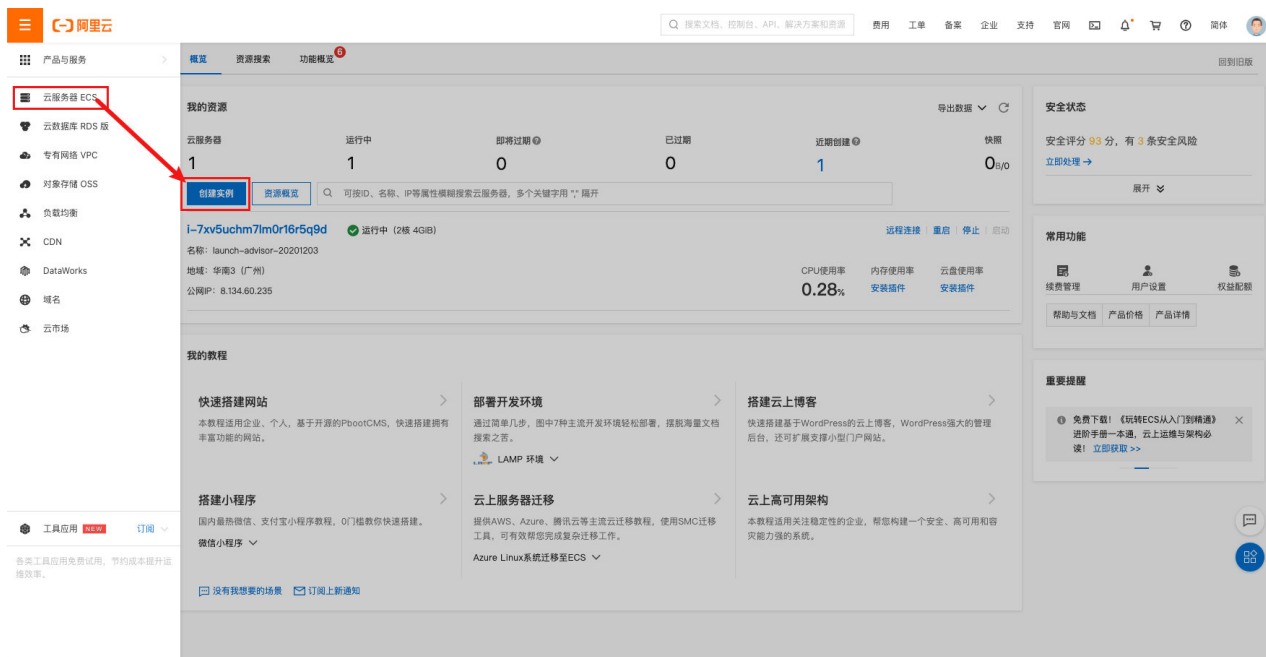
我们需要注册阿里云账号

- <https://aliyun.com/>
- 注册即可，非常简单

1.2. 购买云服务器

购买云服务器其实是购买一个实例。

1.来到控制台：



2.创建实例，选择类型和配置

付费模式

包年包月

按量付费

抢占式实例

按量付费 ECS 支持停机后部分资源不收费功能，可以有效降低成本，了解相关限制和触发条件 [查看详情>](#)
节省计划可以 **大幅降低** 按量付费账单，且计划内支持多种产品灵活使用，不受地域/升级/变更规则族限制，[前往介绍页了解更多>](#)

地域及可用区

华南 3（广州）

随机分配

可用区 A（1）

可用区 B

如何选择地域

不同地域的实例之间内网互不相通；选择靠近您客户的地域，可降低网络时延、提高您客户的访问速度。

实例

分类选型

场景化选型

当前代

所有代

2 vCPU 4 GiB

筛选

2 vCPU

4 GiB

指定规格名称，如：ecs.g5.large

I/O 优化实例

最高支持 IPv6

规格族	实例规格	vCPU	内存	平均基准 CPU 计算性能	处理器主频/睿频	内网带宽	内网收发包	存储 IOPS	IPv6	参考价格	处理器型号
<input checked="" type="radio"/> 高主频计算型 hfc7	ecs.hfc7.large	2 vCPU	4 GiB	-	3.3/3.8 GHz	最高 10 Gbps	90 万 PPS	2 万	是	¥ 0.471 /时	Intel Xeon Platinum (Cooper Lake) 8369
<input type="radio"/> 计算平衡增强型 c6e	ecs.c6e.large	2 vCPU	4 GiB	-	2.5 GHz/3.2 GHz	最高 10 Gbps	90 万 PPS	2.1 万	是	¥ 0.41 /时	Intel Xeon (Cascade Lake) Platinum 8269CY
<input type="radio"/> 安全增强型 c6t	ecs.c6t.large	2 vCPU	4 GiB	-	2.5 GHz/3.2 GHz	最高 10 Gbps	90 万 PPS	2.1 万	是	¥ 0.423 /时	Intel Xeon (Cascade Lake) Platinum 8269CY
<input type="radio"/> 高主频计算型 hfc6	ecs.hfc6.large	2 vCPU	4 GiB	-	3.1 GHz/3.5 GHz	最高 3 Gbps	30 万 PPS	1.05 万	是	¥ 0.448 /时	Intel Xeon (Cascade Lake) Platinum 8269
<input type="radio"/> 计算平衡增强型 c6e	ecs.c6e.large	2 vCPU	4 GiB	-	2.5 GHz/3.2 GHz	最高 3 Gbps	30 万 PPS	1.05 万	是	¥ 0.395 /时	Intel Xeon (Cascade Lake) Platinum 8269CY

当前选择实例

ecs.hfc7.large（2 vCPU 4 GiB，高主频计算型 hfc7）

购买实例数量

-

1

+

当前所选实例规格在 广州 可用区 A 的总配额最多，已开通 0 台，最多还可开通 128 台。如需更多配额，您可 [前往控制台提升>](#)

镜像

公共镜像

自定义镜像

共享镜像

镜像市场

CentOS 7.6 64位

安全加固

系统盘

配置费用：¥ 0.513 /时

下一步：网络和安全组

确认订单

3.配置网络安全组

基础配置

2 网络和安全组

3 系统配置（选项）

4 分组设置（选项）

5 确认订单

网络

专有网络

如何选择网络

默认专有网络

如需创建新的专有网络，您可 [前往控制台创建>](#)

默认交换机

交换机所在可用区：随机分配 交换机网段：-

您可 [前往控制台创建>](#)

公网 IP

☒ 分配公网 IPv4 地址

系统会分配公网 IP，也可采用更加灵活的弹性公网 IP 方案，了解 [如何配置并绑定弹性公网 IP 地址>](#)

公网带宽计费

按使用流量

按固定带宽

后付费模式，按使用流量（单位为GB）计费，每小时扣费，请保证余额充足。

带宽峰值

1M 25M 50M 75M 100M

5 Mbps

阿里云免费提供最高 5Gbps 的恶意流量攻击防护。[了解更多 | 提升防护能力](#)

安全组

重新选择安全组

安全组类似防火墙功能，用于设置网络访问控制。您也可以到管理控制台 [新建安全组>](#) [安全FAQ>](#)

所选安全组 1)，默认安全组（自定义端口）

请确保所选安全组开放包含 22（Linux）或者 3389（Windows）端口，否则无法远程登录ECS，您可以进入ECS控制台设置。[前往设置>](#)

请勾选要开通的IPv4的协议/端口：☒ HTTP 80 端口 ☒ HTTPS 443 端口 ☒ 22 端口 ☐ 3389 端口 ☒ ICMP 协议

弹性网卡

选择专有网络交换机方可配置弹性网卡

通过弹性网卡，您可以实现高可用集群搭建、低成本故障转移和精细化的网络管理。[了解更多>](#)

公网带宽：5Mbps 按使用流量

配置费用：¥ 0.513 /时

公网流量费用：¥ 0.800 /GB

上一步：基础配置

下一步：系统配置

确认订单

4.创建实例

云服务器 ECS

一键购买

自定义购买

购买历史

产品价格

购买云盘

产品控制台

基础配置

网络和安全组

系统配置 (选项)

分组设置 (选项)

确认订单

所选配置

基础配置

付费模式: 按量付费

地域及可用区: 华南 3 (广州) / 随机分配

实例: 高主频计算型 hfc7 / ecs.hfc7.large (2vCPU 4GiB)

购买数量: 1 台

镜像: CentOS 7.6 64位 (安全加固)

系统盘: ESSD云盘 40GiB, 随实例释放, PLO (单盘IOPS性能上限1万)

网络和安全组

网络: 专有网络

VPC: 默认专有网络

交换机: 默认交换机

公网带宽: 按使用流量 5Mbps

安全组: 1), 默认安全组 (自定义端口)

系统配置

登录凭证: 自定义密码

实例名称: launch-advisor-20201203

保存为启动模板

生成Open API最佳实践脚本

保存当前购买配置为ROS模板

使用期限

☐ 设置自动释放服务时间

ECS实例将在您预约的时间点进行释放, 实例释放后数据及IP地址不会被保留且无法找回, 请谨慎操作。

服务协议

☒ 《云服务器 ECS 服务条款》

购买须知

订单对应的发票信息, 请在 管理控制台-费用中心-发票管理 中查看。

云产品默认禁用 TCP 25 端口和基于此端口的邮箱服务, 特殊情况需报备审核后使用, 查看详情>

公网带宽: 5Mbps 按使用流量

配置费用: ¥0.513 /时

公网流量费用: ¥0.800 /GB

上一步: 分组设置

创建实例

二. 配置云服务器

2.1. 连接云服务器

通常情况下, 我们会通过ssh连接云服务器:

- Windows电脑上我推荐直接使用git bash ssh工具;
- Mac OS电脑上我们可以直接通过终端来使用ssh工具;

```
coderwhy — root@coderwhy:~ — ssh root@8.134.60.235 — 80x24
coderwhy@why ~ % ssh root@8.134.60.235
root@8.134.60.235's password:

Welcome to Alibaba Cloud Elastic Compute Service !

Activate the web console with: systemctl enable --now cockpit.socket

Last failed login: Thu Dec 3 23:33:05 CST 2020 from 116.23.114.222 on ssh:notty
There were 3 failed login attempts since the last successful login.
Last login: Thu Dec 3 15:52:28 2020 from 112.94.190.104
[root@coderwhy ~]#
```

知识点补充: 如果在计算机中想要更改主机名

- 修改之后需要重启服务器

```
1 | hostnamectl --static set-hostname coderwhy
```

2.2. 安装Node.js

我们安装软件使用工具：dnf

- **DNF**，全称**Dandified**（时髦的、华丽的）**Yum**；
- 是Yum的下一个版本，也被称之为Yum的替代品；
- 如果是centos7的版本，我记得是没有自带dnf的，需要通过yum进行安装（这个自行安装一下）；
- 刚才在选择云服务器时，我选择的是centos8，所以是自带dnf的；

检查dnf是否可用：

```
1 | dnf --help
```

如果我们希望安装一个软件包，可以进行如下的操作：

```
1 | # 搜索软件包
2 | dnf search nodejs
3 |
4 | # 查看软件包信息：nodejs的版本是10.21.0
5 | dnf info nodejs
6 |
7 | # 安装nodejs
8 | dnf install nodejs
```

我们会发现版本其实是10.21.0：

- 我们其实希望使用更高的版本，比如最新的LTS或者Current版本；
- 这个时候我们可以使用之前讲过的一个工具：n；

```
1 | # 安装n
2 | npm install n -g
3 |
4 | # 通过n安装最新的lts和current
5 | n install lts
6 | n install latest
7 |
8 | # 通过n切换版本
9 | n
```

如果发现切换之后终端没有反应，可以进行重启：

- 方式一：重新通过ssh建立连接；
- 方式二：重启ssh `service sshd restart`

2.3. 安装MySQL

2.3.1. 安装MySQL

我们依然使用dnf来安装MySQL：

```
1 # 查找MySQL
2 dnf search mysql-server
3
4 # 查看MySQL，这里的版本是8.0.21
5 dnf info mysql-server
6
7 # 安装MySQL，这里加-y的意思是依赖的内容也安装
8 dnf install mysql-server -y
```

启动mysql-server：

```
1 # 开启MySQL后台服务
2 systemctl start mysqld
3
4 # 查看MySQL服务：active (running)表示启动成功
5 systemctl status mysql
6
7 # 随着系统一起启动
8 systemctl enable mysqld
```

2.3.2. 配置MySQL

我们之前在Mac或者Windows上安装MySQL时会有一些配置：

- 比如账号密码；
- 在centos中通过dnf安装之后，我们如何配置账号密码呢？

配置MySQL账号和密码：

```
1 mysql_secure_installation
2
3 # 接下来有一些选项，比如密码强度等等一些
4 # MySQL8开始通常设置密码强度较强，选择2
5 # 其他的选项可以自行选择
```

现在，我们就可以直接在服务器中操作MySQL了：

```
[root@coderwhy ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 200
Server version: 8.0.21 Source distribution

Copyright (c) 2000, 2020, 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>
```

但是如果我们希望在自己的电脑上直接连接MySQL呢？

- 也就是和MySQL建立远程连接；
- 比如直接Navicat工具中连接MySQL；

这个时候必须要配置root可以远程连接：

```
1 # 使用mysql数据库
2 use mysql;
3 # 查看user表中，连接权限，默认看到root是localhost
4 select host, user from user;
5 # 修改权限
6 update user set host = '%' where user = 'root';
```

但是呢，阿里云默认有在安全组中禁止掉3306端的连接的：

- 所以我们需要配置3306的安全组

sg-7xvh21rj2r2e9igau8m... / vpc-7xv18ufjncqyikj17dt1								教我设置	返回	添加安全组规则	快速创建规则
入方向	出方向										导入 导出
<input type="checkbox"/>	授权策略	协议类型	端口范围	授权类型 (全部)	授权对象	描述	优先级	创建时间	操作		
<input type="checkbox"/>	允许	自定义 TCP	3306/3306	IPv4地址段访问	0.0.0.0/0	-	1	2020年12月3日 11:36	修改	克隆	删除
<input type="checkbox"/>	允许	自定义 TCP	443/443	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改	克隆	删除
<input type="checkbox"/>	允许	自定义 TCP	80/80	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改	克隆	删除
<input type="checkbox"/>	允许	自定义 TCP	22/22	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改	克隆	删除
<input type="checkbox"/>	允许	全部 ICMPv4	-1/-1	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改	克隆	删除
<input type="checkbox"/>	删除										

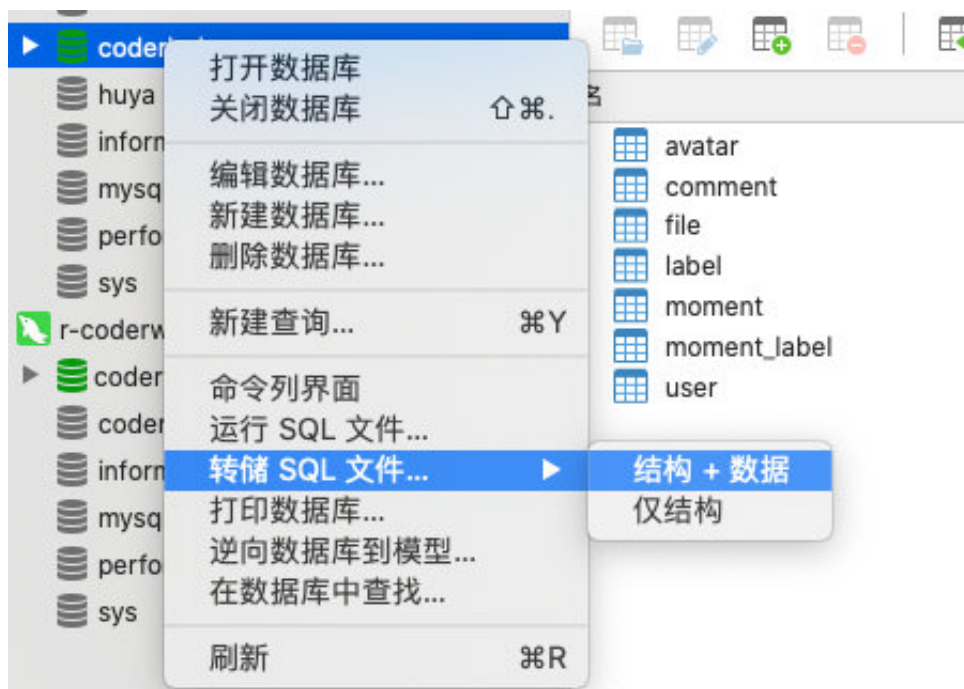
2.3.3. 数据库迁移

我们需要将之前项目中（本地）MySQL的数据库迁移到服务器中的MySQL数据库中。

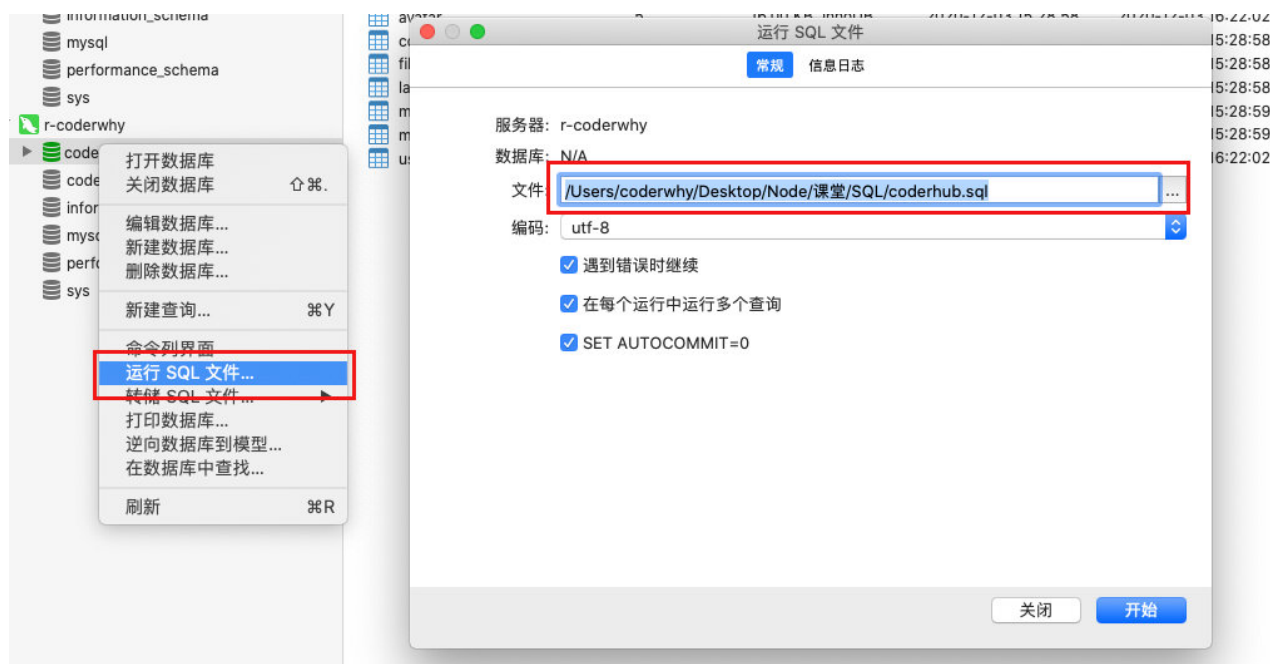
第一步：在服务器MySQL中创建coderhub的数据库：

- 可以通过命令，也可以通过Navicat直接创建

第二步：在Navicat工具中本地MySQL直接导出数据库



第三步：在Navicat工具中服务器MySQL中执行MySQL



三. 部署Node项目

3.1. 手动部署

3.1.1. 代码托管到Git仓库

在GitHub中创建仓库，并且将当前代码放到GitHub中

添加.gitignore文件时，忽略一些文件：

- 忽略uploads
- 忽略.env

这里不再给出详细的步骤，就是把代码托管到GitHub上。

3.1.2. 代码clone到服务器

服务器我们使用Git来clone代码：

- centos8服务器中默认是没有安装Git的；
- 我们可以通过dnf来安装；

```
1 dnf search git;
2 dnf info git;
3 # 当然你也可以直接安装（上面两个只是让大家看一下Git的详情）
4 dnf install git;
```

我们可以在根目录下创建一个自己的文件夹，比如why

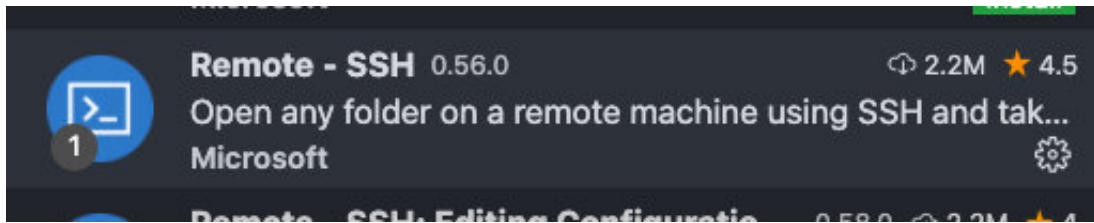
```
1 cd /
2 mkdir why
3 cd why/
```

clone项目到why中：

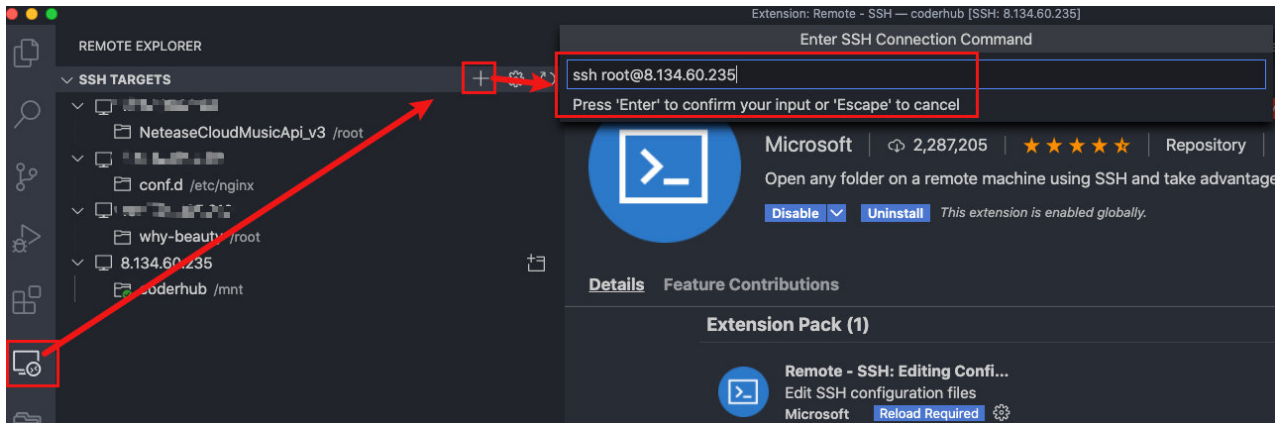
```
1 git clone https://github.com/coderwhy/coderhub.git
```

3.1.3. VSCode中打开代码

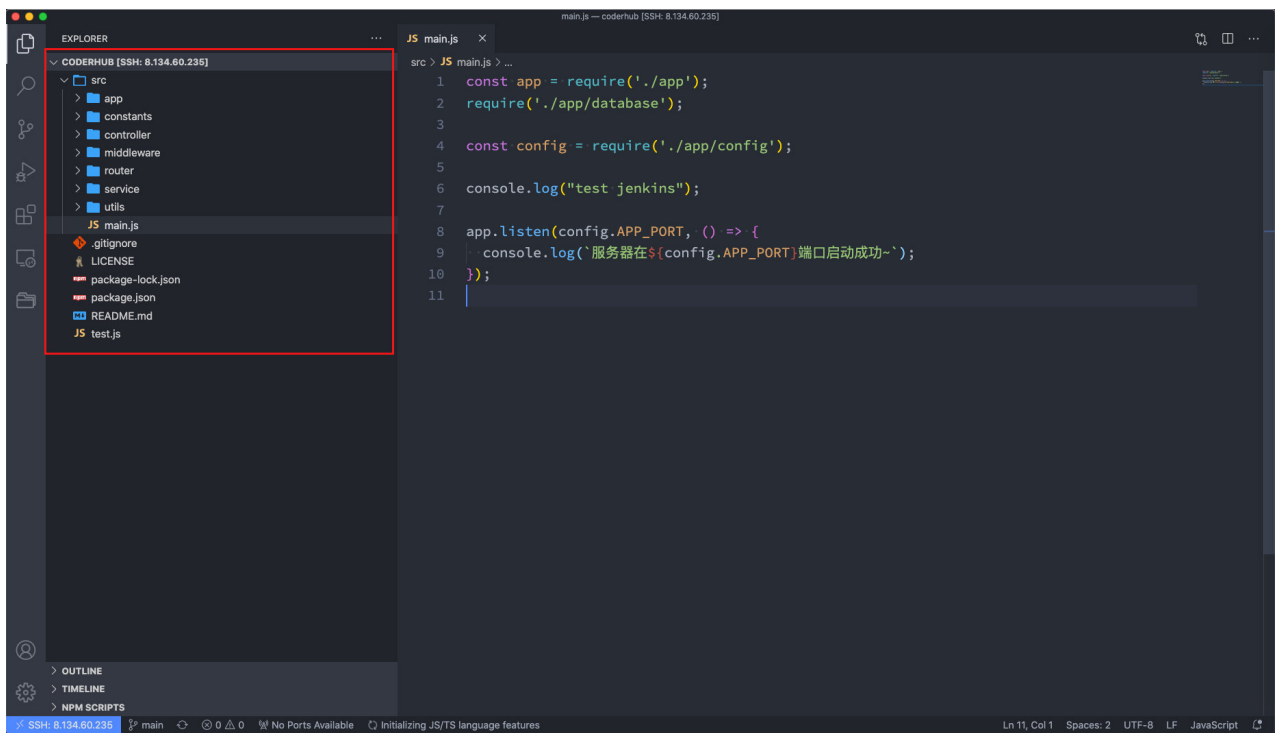
如果我们希望在本地的VSCode中可以直接编辑远程的代码，可以使用一个VSCode的插件：remote-ssh



连接远程服务器：



查看远程的项目代码：



安装项目的依赖：

```
1 | npm install
```

配置.env文件

```
1 APP_HOST=http://8.134.60.235
2 APP_PORT=8001
3
4 MYSQL_HOST=localhost
5 MYSQL_PORT=3306
6 MYSQL_DATABASE=coderhub
7 MYSQL_USER=root
8 MYSQL_PASSWORD=Coderwhy888.
```

注意：加入8001端口到安全组中

入方向	出方向								导入	导出
<input type="checkbox"/> 授权策略	协议类型	端口范围	授权类型 (全部)	授权对象	描述	优先级	创建时间	操作		
<input type="checkbox"/> 允许	自定义 TCP	8001/8001	IPv4地址段访问	0.0.0.0/0	-	1	2020年12月4日 16:57	修改 克隆 删除		
<input type="checkbox"/> 允许	自定义 TCP	8080/8080	IPv4地址段访问	0.0.0.0/0	-	1	2020年12月3日 17:24	修改 克隆 删除		
<input type="checkbox"/> 允许	自定义 TCP	8000/8000	IPv4地址段访问	0.0.0.0/0	-	1	2020年12月3日 16:50	修改 克隆 删除		
<input type="checkbox"/> 允许	自定义 TCP	3306/3306	IPv4地址段访问	0.0.0.0/0	-	1	2020年12月3日 11:36	修改 克隆 删除		
<input type="checkbox"/> 允许	自定义 TCP	443/443	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改 克隆 删除		
<input type="checkbox"/> 允许	自定义 TCP	80/80	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改 克隆 删除		
<input type="checkbox"/> 允许	自定义 TCP	22/22	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改 克隆 删除		
<input type="checkbox"/> 允许	全部 ICMP(IPv4)	-1/-1	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改 克隆 删除		

测试请求的结果：

The screenshot shows the Postman interface with a GET request to `{{baseURL}}/moment/1`. The response is a JSON object with the following structure:

```
{
  "id": 1,
  "content": "我说错了，C语言才是最好的语言~",
  "createTime": "2020-11-23T14:05:23.000Z",
  "updateTime": "2020-11-27T13:35:42.000Z",
  "author": {
    "id": 4,
    "name": "lucy",
    "avatarUrl": "http://8.134.60.235:8000/users/4/avatar"
  },
  "labels": [
    {
      "id": 3,
      "name": "爱情"
    }
  ],
  {
    "id": 5,
```

3.1.4. pm2启动node程序

刚才我们是通过终端启动的node程序，那么如果终端被关闭掉了呢？

- 那么这个时候相当于启动的Node进程会被关闭掉；
- 我们将无法继续访问服务器；

在真实的部署过程中，我们会使用一个工具pm2来管理Node的进程：

- PM2是一个Node的进程管理器；
- 我们可以使用它来管理Node的后台进程；
- 这样在关闭终端时，Node进程会继续执行，那么服务器就可以继续为前端提供服务了；

安装pm2：

```
1 | npm install pm2 -g
```

pm2常用的命令：

```
1  # 命名进程
2  pm2 start app.js --name my-api
3  # 显示所有进程状态
4  pm2 list
5  # 停止指定的进程
6  pm2 stop 0
7  # 停止所有进程
8  pm2 stop all
9  # 重启所有进程
10 pm2 restart all
11 # 重启指定的进程
12 pm2 restart 0
13
14 # 杀死指定的进程
15 pm2 delete 0
16 # 杀死全部进程
17 pm2 delete all
18
19 #后台运行pm2，启动4个app.js，实现负载均衡
20 pm2 start app.js -i 4
```

3.2. jenkins自动化部署

3.2.1. 安装Java环境

Jenkins本身是依赖Java的，所以我们需要先安装Java环境：

- 这里我安装了Java1.8的环境

```
1 dnf search java1.8
2 dnf install java-1.8.0-openjdk.x86_64
```

3.2.2. 安装Jenkins

因为Jenkins本身是没有在dnf的软件仓库包中的，所以我们需要连接Jenkins仓库

```
1 wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-
  stable/jenkins.repo
2
3 # 导入GPG密钥以确保您的软件合法
4 rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
```

编辑一下文件/etc/yum.repos.d/jenkins.repo

- 可以通过vim编辑

```
1 [jenkins]
2
3 name=Jenkins-stable
4
5 baseurl=http://pkg.jenkins.io/redhat
6
7 gpgcheck=1
```

安装Jenkins

```
1 dnf install jenkins
```

启动Jenkins的服务：

```
1 systemctl start jenkins
2 systemctl status jenkins
3 systemctl enable jenkins
```

Jenkins默认使用8080端口提供服务，所以需要加入到安全组中：

<input type="checkbox"/> 授权策略	协议类型	端口范围	授权类型 (全部)	授权对象	描述	优先级	创建时间	操作
<input type="checkbox"/> 允许	自定义 TCP	8001/8001	IPv4地址段访问	0.0.0.0/0	-	1	2020年12月4日 16:57	修改 克隆 删除
<input type="checkbox"/> 允许	自定义 TCP	8080/8080	IPv4地址段访问	0.0.0.0/0	-	1	2020年12月3日 17:24	修改 克隆 删除
<input type="checkbox"/> 允许	自定义 TCP	8000/8000	IPv4地址段访问	0.0.0.0/0	-	1	2020年12月3日 16:50	修改 克隆 删除
<input type="checkbox"/> 允许	自定义 TCP	3306/3306	IPv4地址段访问	0.0.0.0/0	-	1	2020年12月3日 11:36	修改 克隆 删除
<input type="checkbox"/> 允许	自定义 TCP	443/443	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改 克隆 删除
<input type="checkbox"/> 允许	自定义 TCP	80/80	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改 克隆 删除
<input type="checkbox"/> 允许	自定义 TCP	22/22	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改 克隆 删除
<input type="checkbox"/> 允许	全部 ICMP(IPv4)	-1/-1	IPv4地址段访问	0.0.0.0/0	System created rule.	100	2020年12月3日 10:43	修改 克隆 删除

3.2.3. Jenkins配置

打开浏览器，输入：<http://8.134.60.235:8080/>

- 注意：你输入自己的IP地址

获取输入管理员密码：

- 在下面的地址中 `cat /var/lib/jenkins/secrets/initialAdminPassword`

入门

解锁 Jenkins

为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里？](#)）该文件在服务器上：

```
/var/lib/jenkins/secrets/initialAdminPassword
```

请从本地复制密码并粘贴到下面。

管理员密码



继续

可以安装推荐的插件：

Recommended Plugins

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.


Select plugins to install


Select and install plugins most suitable for your needs.


3.2.4. Jenkins任务


新建任务：


Dashboard ▾ ▸


 新建任务


 用户列表


 构建历史


 项目关系

 检查文件指纹

 系统管理

 我的视图



 Lockable Resources

 新建视图

构建队列 ^

队列中没有构建任务

所有 +

S	W	名称 ↓	上次成功
		coderhub	20 秒 - #51

图标: 小 中 大

输入一个任务名称

该字段不能为空，请输入一个合法的名称



构建一个自由风格的软件项目

这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统.



流水线

精心地组织一个可以长期运行在多个节点上的任务.适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。



构建一个多配置项目

适用于多配置项目,例如多环境测试,平台指定构建,等等.



文件夹

创建一个可以嵌套存储的容器。利用它可以进行分组。视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间，因此你可以有多个相同名称的内容，只要它们在不同的文件 夹里即可。



GitHub 组织

扫描一个 GitHub 组织（或者个人账户）的所有仓库来匹配已定义的标记。



多分支流水线

根据一个或多个仓库来创建分支，并自动创建、更新分支。

配置构建的保留策略：

General

源码管理

构建触发器

构建环境

构建

构建后操作

描述

自动化构建 coderhub project

[纯文本] 预览

☐ GitHub 项目

☐ Permission to Copy Artifact

☐ This build requires lockable resources

☐ Throttle builds

☒ 丢弃旧的构建

策略

Log Rotation

保持构建的天数

10

如果非空，构建记录将保存此天数

保持构建的最大个数

8

如果非空，最多此数目的构建记录将被保存

高级...

☐ 参数化构建过程

☐ 关闭构建

☐ 在必要的时候并发构建

源码管理：

General

源码管理

构建触发器

构建环境

构建

构建后操作

源码管理

无

Git

Repositories

Repository URL

https://github.com/coderwhy/coderhub.git

Credentials

coderwhy/*****

添加

高级...

Add Repository

Branches to build

指定分支 (为空时代表any)

*/main

增加分支

源码库浏览器

(自动)

构建触发器：

General

源码管理

构建触发器

构建环境

构建

构建后操作

构建触发器

触发远程构建 (例如,使用脚本)

其他工程构建后触发

定时构建

GitHub hook trigger for GITScm polling

轮询 SCM

日程表

HH * * *

上次运行的时间 Thursday, December 3, 2020 7:19:32 PM CST; 下次运行的时间 Friday, December 4, 2020 7:19:32 PM CST.

忽略钩子 post-commit

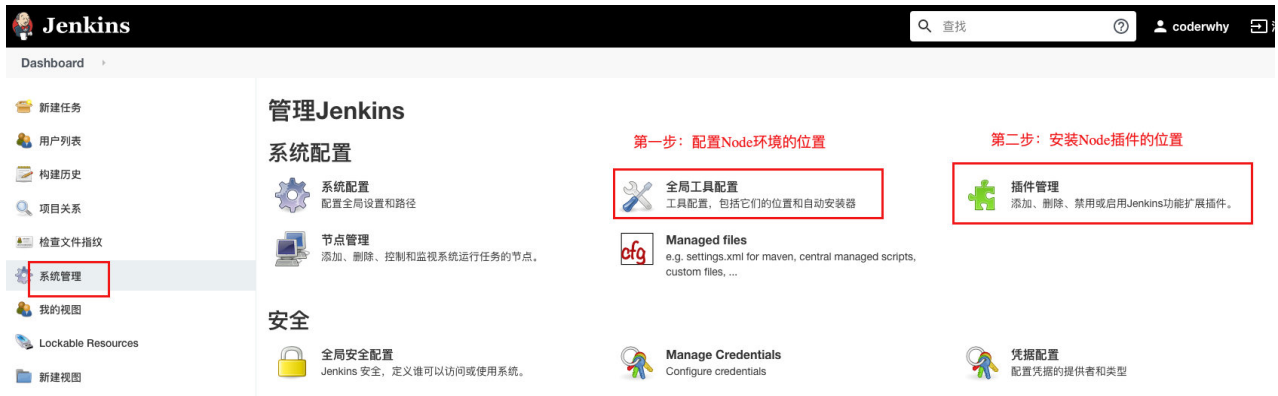
构建环境

构建环境：

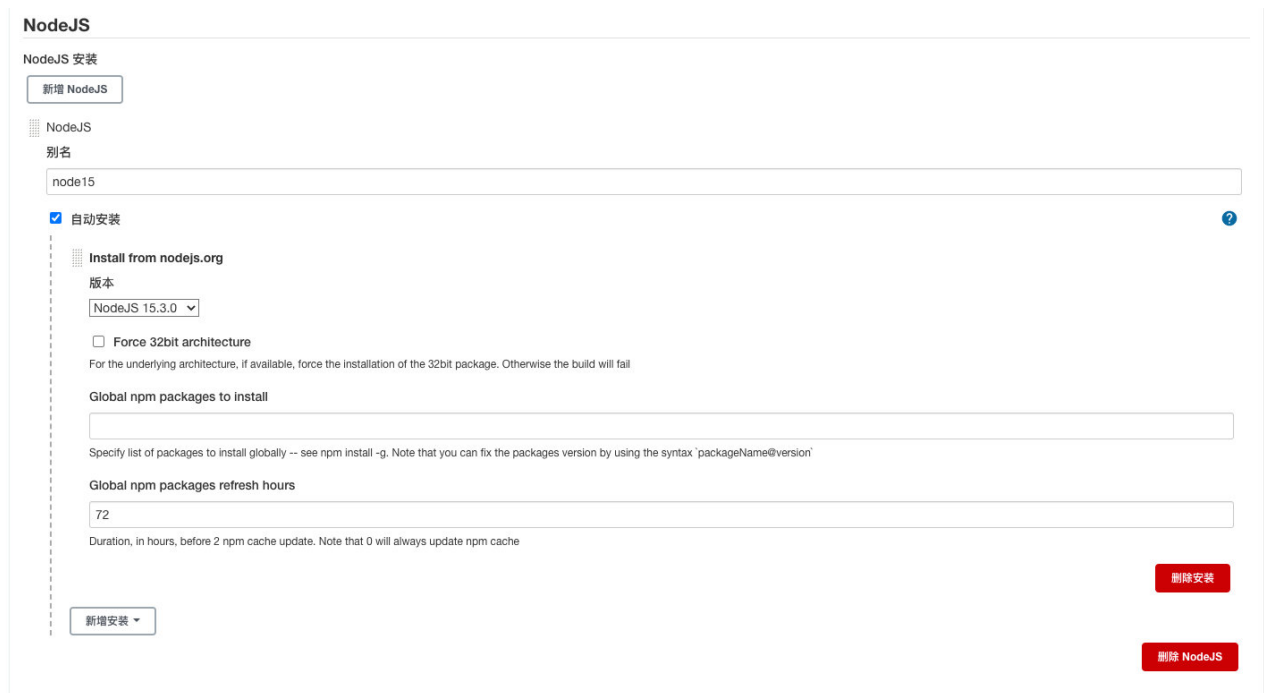
注意：我们需要搭建Node的环境

- 第一步：配置Node的环境；

- 第二步：安装Node的插件；



第一步：配置Node的环境



第二步：安装Node的插件

- 这里因为我已经安装过了，所以没有搜索到；



构建执行的任务：

- 查看Node的版本等是否有问题；
- 执行 `npm install` 安装项目的依赖；

- 移除 `/mnt/coderhub` 中的文件，除了

```
1  ls
2
3  node -v
4  npm -v
5
6  npm install
7
8  cd /mnt/coderhub/
9  ls
10 shopt -s extglob
11 rm -rf /mnt/coderhub/* !(".env"|"."|"..")
12
13 cd /var/lib/jenkins/workspace/coderhub
14 ls
15 pwd
16 cp -rf * /mnt/coderhub/
```

构建

执行 shell

命令

```
ls
node -v
npm -v
npm install
cd /mnt/coderhub/
ls
shopt -s extglob
rm -rf /mnt/coderhub/* !(".env"|"."|"..")
cd /var/lib/jenkins/workspace/coderhub
ls
pwd
cp -rf * /mnt/coderhub/
```

查看 [可用的环境变量列表](#)

高级...

增加构建步骤 ▾

ecosystem.config.js文件:

```
1 module.exports = {  
2   apps: [  
3     {  
4       name: "coderhub",  
5       script: "./src/main.js",  
6       watch: true  
7     }  
8   ]  
9 }
```