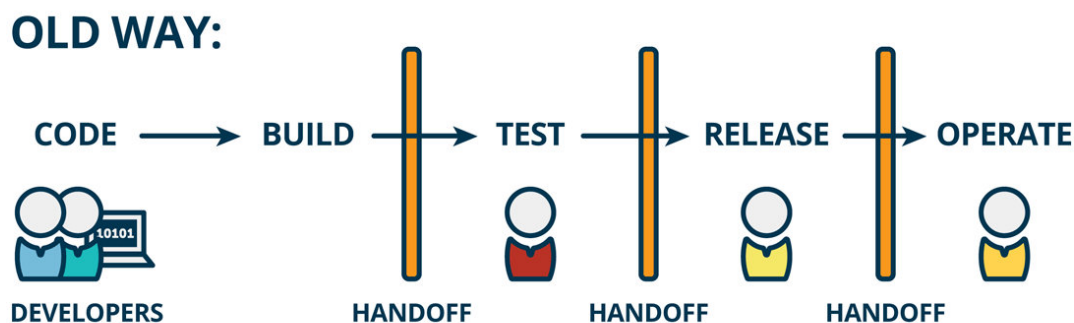


项目打包和自动化部署

一. 项目部署和DevOps

1.1. 传统的开发模式

在传统的开发模式中，开发的整个过程是按部就班就行：

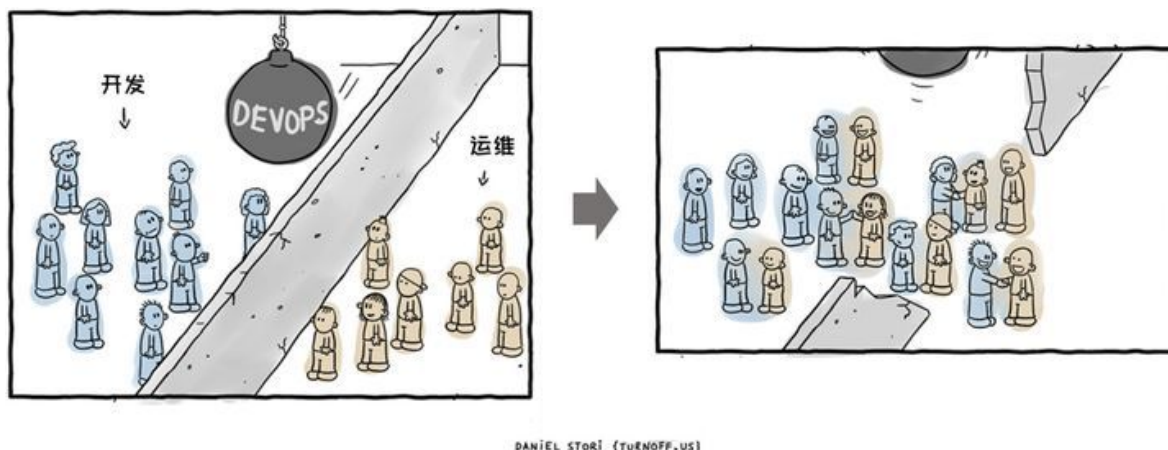


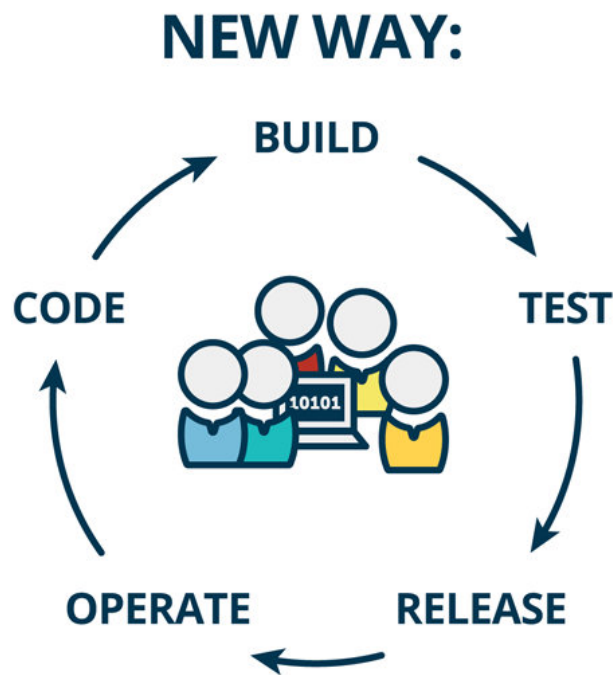
但是这种模式存在很大的弊端：

- 工作的不协调：开发人员在开发阶段，测试和运维人员其实是处于等待的状态。等到测试阶段，开发人员等待测试反馈bug，也会处于等待状态。
- 线上bug的隐患：项目准备交付时，突然出现了bug，所有人员需要加班、等待问题的处理；

1.2. DevOps开发模式

DevOps是Development和Operations两个词的结合，将开发和运维结合起来的模式：



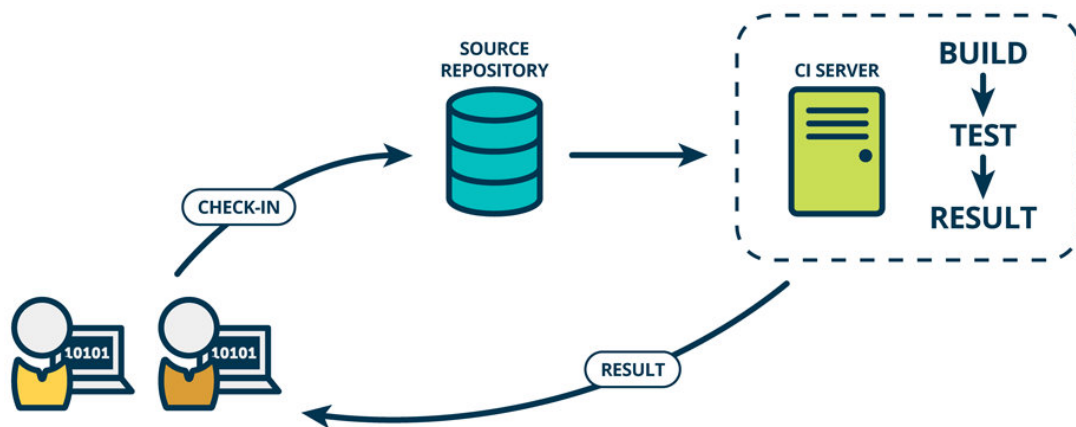


1.3. 持续集成和持续交付

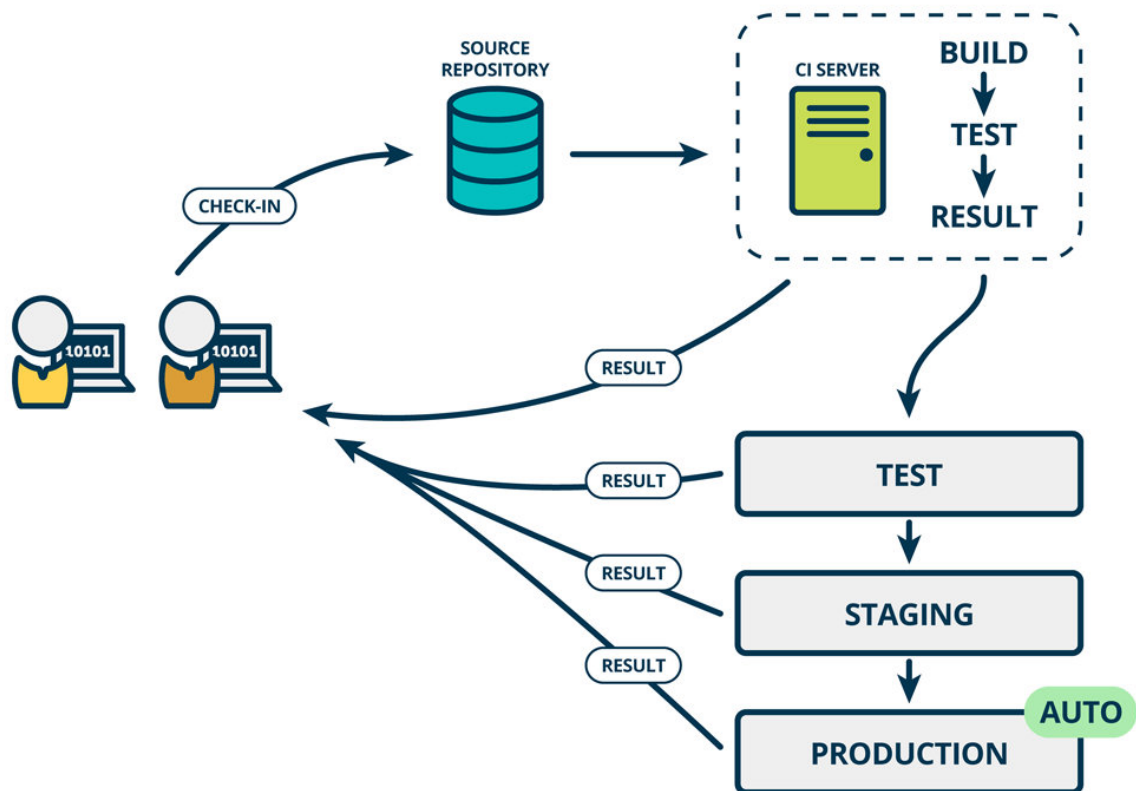
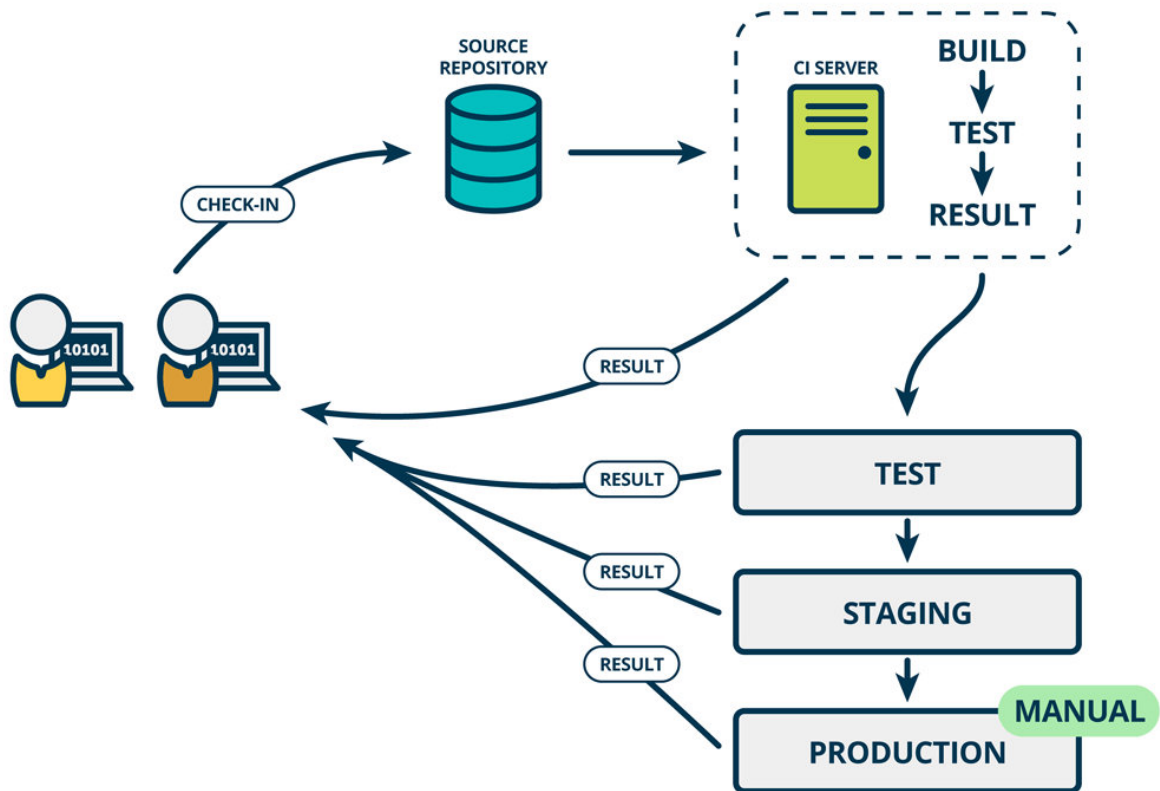
伴随着DevOps一起出现的两个词就是持续集成和持续交付(部署):

- CI是Continuous Integration (持续集成) ;
- CD是两种翻译: Continuous Delivery (持续交付) 或Continuous Deployment (持续部署) ;

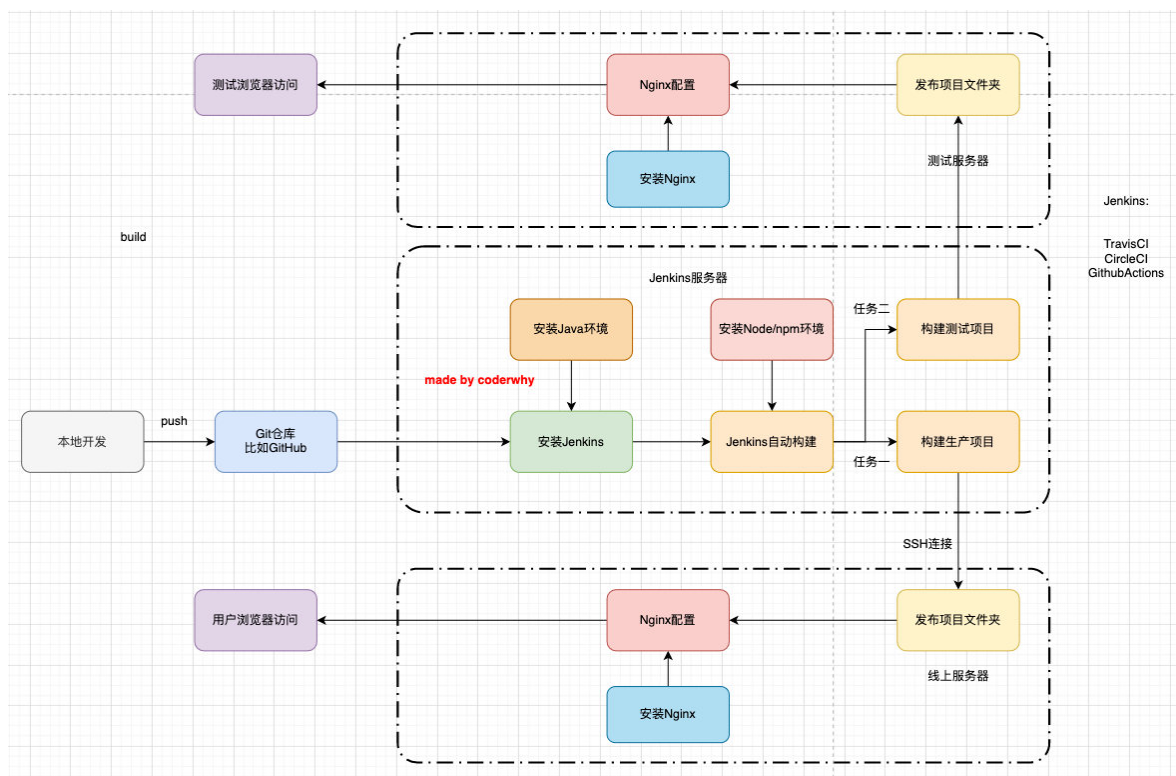
持续集成CI:



持续交付和持续部署:



1.4. 自动化部署流程



二. 购买云服务器

2.1. 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：

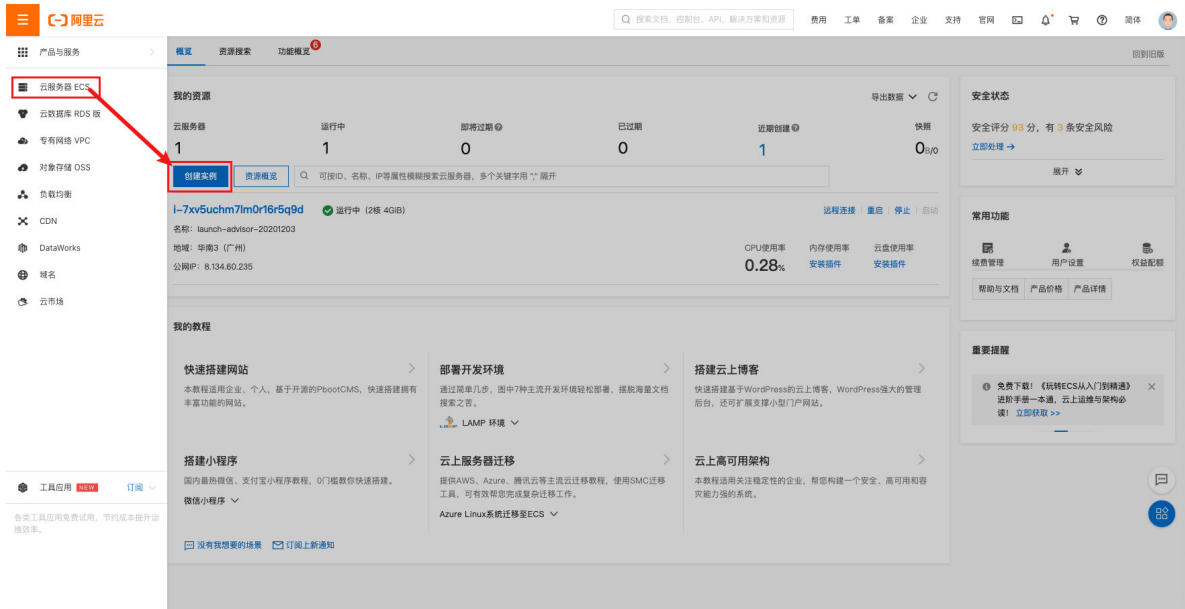
我们需要注册阿里云账号

- <https://aliyun.com/>
- 注册即可，非常简单

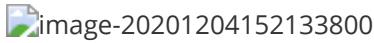
2.2. 购买云服务器

购买云服务器其实是购买一个实例。

1.来到控制台：



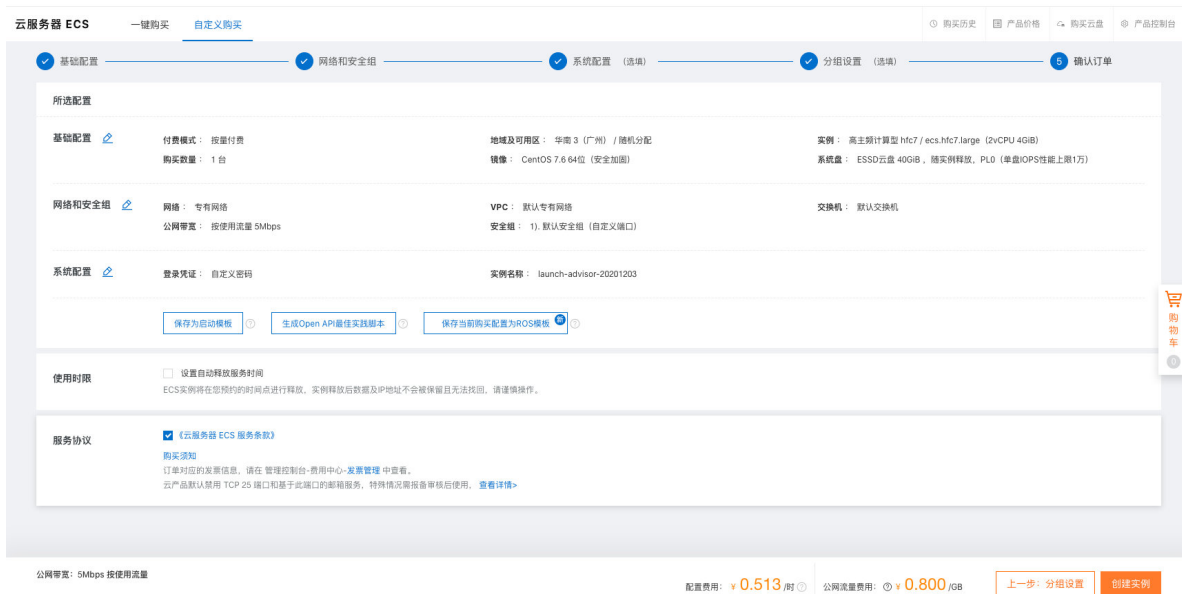
2.创建实例，选择类型和配置



3.配置网络安全组



4.创建实例



三. 搭建服务器环境

3.1. jenkins自动化部署

3.1.1. 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

```
dnf search java-1.8
dnf install java-1.8.0-openjdk.x86_64
```

3.1.2. 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：

- wget是Linux中下载文件的一个工具，-O表示输出到某个文件夹并且命名为什么文件；
- rpm：全称为**The RPM Package Manage**，是Linux下一个软件包管理器；

```
wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo

# 导入GPG密钥以确保您的软件合法
rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
# 或者
rpm --import http://pkg.jenkins-ci.org/redhat/jenkins-ci.org.key
```

编辑一下文件/etc/yum.repos.d/jenkins.repo

- 可以通过vim编辑

```
[jenkins]
```

```
name=Jenkins-stable
```

```
baseurl=http://pkg.jenkins.io/redhat
```

```
gpgcheck=1
```

安装Jenkins

```
dnf install jenkins # --nogpgcheck(可以不加)
```

启动Jenkins的服务：

```
systemctl start jenkins
systemctl status jenkins
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

3.1.3. Jenkins用户

我们后面会访问centos中的某些文件夹，默认Jenkins使用的用户是 `jenkins`，可能会没有访问权限，所以我们需要修改一下它的用户：

修改文件的路径：`/etc/sysconfig/jenkins`

```
Welcome  jenkins x
etc > sysconfig > jenkins
20
21  ## Type: .....string
22  ## Default: ..... "jenkins"
23  ## ServiceRestart: jenkins
24  #
25  # Unix user account that runs the Jenkins daemon
26  # Be careful when you change this, as you need to update
27  # permissions of $JENKINS_HOME and /var/log/jenkins.
28  #
29  JENKINS_USER="root"
```


之后需要重启一下Jenkins:

```
# 也可以将Jenkins添加到root组中
sudo usermod -a -G root jenkins

systemctl restart jenkins
```

3.1.4. Jenkins配置

打开浏览器, 输入: <http://8.134.60.235:8080/>

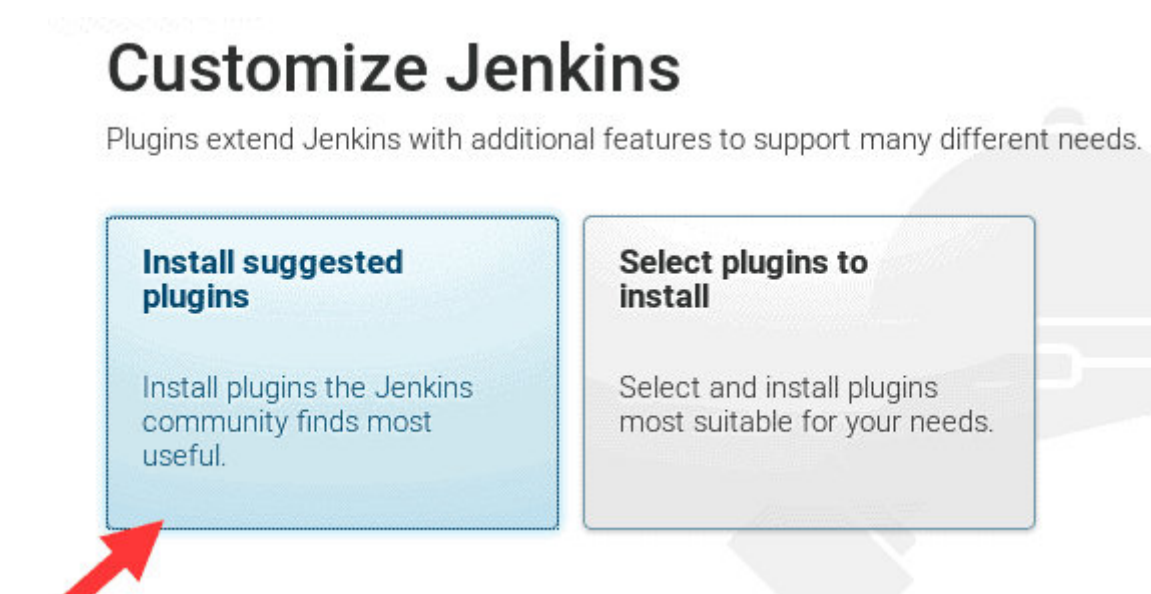
- 注意: 你输入自己的IP地址

获取输入管理员密码:

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

 image-20201203173047824

可以安装推荐的插件:



3.1.5. 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SCM仓库来检测并分支创建，并构建流水线。

配置项目和保留策略：

General 源码管理 构建触发器 构建环境 构建 构建后操作

描述

xue3_mall cms

[纯文本] 预览

☐ GitHub 项目

☐ This build requires lockable resources

☐ Throttle builds

☐ 丢弃旧的构建

☐ 参数化构建过程

☐ 关闭构建

☐ 在必要的时候开发构建

高级...

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/ny-cms-vue3-ts

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

构建触发器：

这里的触发器规则是这样的：

- 定时字符串从左往右分别是：分 时 日 月 周

#每半小时构建一次OR每半小时检查一次远程代码分支，有更新则构建

H/30 * * * *

#每两小时构建一次OR每两小时检查一次远程代码分支，有更新则构建

H H/2 * * *

#每天凌晨两点定时构建

H 2 * * *

#每月15号执行构建

H H 15 * *

#工作日，上午9点整执行

H 9 * * 1-5

#每周1,3,5，从8:30开始，截止19:30，每4小时30分构建一次

H/30 8-20/4 * * 1,3,5

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

日程表

H H * * *

上次运行的时间 Wednesday, August 25, 2021 2:30:29 AM CST; 下次运行的时间 Thursday, August 26, 2021 2:30:29 AM CST.

GitHub hook trigger for GITScm polling

轮询 SCM

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

构建环境

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Provide Configuration files
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published Gradle build scans
- ☒ Provide Node & npm bin/ folder to PATH

NodeJS installation

node14

Specify needed nodejs installation where npm installed packages will be provided to the PATH

npmrc file

- use system default -

Cache location

Default (~/.npm or %APP_DATA%\npm-cache)

☐ With Ant

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；

Search:

可更新 可选插件 已安装 高级

安装 ↑	名称	版本	Released
------	----	----	----------

使用上面的搜索框来检索可用的插件。

直接安装 下载待重启后安装 1 天 1 小时 之前获取了更新信息 立即获取

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；
- 移除原来mall_cms文件的所有内容；
- 将打包的dist文件夹内容移动到mall_cms文件夹；

```
pwd
node -v
npm -v

npm install
npm run build

pwd

echo '构建成功'
```

```
ls
```

```
# 删除/root/mall_cms文件夹里所有的内容
```

```
rm -rf /root/mall_cms/*
```

```
cp -rf ./dist/* /root/mall_cms/
```

构建

执行 shell

命令

```
pwd
node -v
npm -v

npm install
npm run build

pwd
echo '构建成功'

ls

# 删除/root/mall_cms文件夹里所有的内容
rm /root/mall_cms/* -rf

cp -rf ./dist/* /root/mall_cms/
```

查看 可用的环境变量列表

高级...

3.2. nginx安装和配置

3.2.1. 安装nginx

后续我们部署会使用nginx，所以需要先安装一下nginx：

```
dnf install nginx
```

启动nginx：

```
systemctl start nginx
systemctl status nginx
systemctl enable nginx
```

3.2.2. 配置nginx

我们这里主要配置nginx的用户和默认访问目录：

```
/etc/nginx/nginx.conf
```

配置用户：

```
5 user root;
6 worker_processes auto;
7 error_log /var/log/nginx/error.log;
8 pid /run/nginx.pid;
```

通过Linux命令创建文件夹和文件：

```
mkdir /root/mall_cms
cd /root/mall_cms
touch index.html

vi index.html
```

配置访问目录：

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    #root /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
        root /root/mall_cms;
        index index.html;
    }
}
```