```
yum  update
```

yum是一种在Red Hat、CentOS和Fedora等Linux发行版中常用的包管理器，yum update是yum命令的一个选项，用于更新系统中的所有已安装的软件包到最新版本。

当执行yum update命令时，yum会先检查可用的软件包，确定哪些软件包需要更新，并将它们的最新版本下载到系统中。这个过程中，yum会自动检查所有软件包的依赖关系，并在必要时同时更新依赖关系。更新完成后，yum还会重新配置系统中的软件包，以确保它们都能够正常工作。

阿里云的最好不*执*行，*执*行各种*报错*

yum更换源

参考：https://developer.aliyun.com/mirror/centos?spm=a2c6h.13651102.0.0.3e221b114JXWlf

执行下面命令

```
mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup
wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-vault-8.5.2111.
yum clean all && yum makecache
```

安装jdk17，git

备注：之前通过yum安装jdk，maven，有一个问题，maven的版本很低，而且自带jdk1.8.导致装完maven后，服务器同时存在jdk17和jdk1.8。这里建议自己下载安装。

查看已安装的java

```
rpm -qa | grep java
```

安装jdk17，yum有jdk17，所以可以直接使用yum安装。git版本低一点没关系。

```
# yum -y install maven  //安装的这个版本自带jdk1.8，有问题，建议自己下载安装
# yum -y remove java-1.8.0-openjdk*   //删除所有1.8jdk
yum -y install java-17-openjdk
yum -y install git
# yum -y install nodejs //yum安装的版本太低，只有10.几，建议自己下载安装
```

安装maven

参考：https://blog.csdn.net/qq_40300509/article/details/127951375

创建目录

```
mkdir -p /apps/maven
```

到这个目录

```
cd /apps/maven
```

下载最新的maven

```
wget https://dlcdn.apache.org/maven/maven-3/3.9.3/binaries/apache-maven-3.9.3-bin.zip
```

解压缩

```
unzip  apache-maven-3.9.3-bin.zip
```

打开环境变量

```
vim /etc/profile
```

增加下面配置

```
# maven
MAVEN_HOME=/apps/maven/apache-maven-3.9.3
# 修改path, NodeJs也是改这里
export PATH=${MAVEN_HOME}/bin:${PATH}
```

重新编译配置文件

```
source  /etc/profile
```

安装nodejs

参考：https://blog.csdn.net/itScholar001/article/details/130950350

创建目录

```
mkdir -p /apps/nodejs
```

到这个目录

```
cd /apps/nodejs
```

下载最新的node，官方的太慢了，建议使用国内镜像

```
wget https://registry.npmmirror.com/-/binary/node/v18.16.1/node-v18.16.1-linux-x64.tar.gz
```

解压缩

```
tar -zxvf node-v18.16.1-linux-x64.tar.gz
```

打开环境变量

```
vim /etc/profile
```

增加下面配置

```
#node,node-v18.16.1
NODE_HOME=/apps/nodejs/node-v18.16.1-linux-x64
export PATH=${NODE_HOME}/bin:${PATH}
```

重新编译配置文件

```
source  /etc/profile
```

执行pnpm安装

```
npm install -g pnpm
```

查看安装情况

```
java -version
mvn -v
node -v
git --version
npm -v
```

可以看到都是最新的了。

配置Apache服务

参照：https://help.aliyun.com/document_detail/253434.html?spm=a2c4g.223746.0.0.3154637fD0lg1C

运行以下命令，安装Apache服务。

```
yum install -y httpd
```

依次运行以下命令，启动Apache服务，并将服务设置为开机自启动。

```
systemctl start httpd
```

将Apache服务设置为开机自启动。

```
systemctl enable httpd
```

运行以下命令，查看Apache服务的运行状态。

```
systemctl status httpd
```

安装docker

参照：https://docs.docker.com/engine/install/centos/

Install the yum-utils package (which provides the yum-config-manager utility) and set up the repository.

```
yum install -y yum-utils
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

如果第二句报错

Invalid configuration value: failovermethod=priority in /etc/yum.repos.d/CentOS-epel.repo; 配置：ID 为 “failovermethod” 的 OptionBinding 不存在

执行下面命令，注释 *failovermethod=priority* ， :wq! 强制保存即可

```
vim /etc/yum.repos.d/CentOS-epel.repo
```

Install Docker Engine, containerd, and Docker Compose

```
yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugir
```

常见命令：

开启

```
systemctl start docker
```

重启

```
systemctl restart docker
```

停止

```
systemctl stop docker
```

开机启动

```
systemctl enable docker
```

日志

```
docker logs 容器名称
```

删除

```
docker rm -f 容器名称
```

语法参照：https://www.runoob.com/docker/docker-run-command.html

生成密码，需Httpd

创建账号密码：redminepostgres的加密后字符串。Nginx做Basic认证可以导入一个文件。

```
htpasswd -nbm redmine redminepostgres
redmine:$apr1$hrI62q/P$dJgHJ5WyMQtv4xSVjmwUE.
```

安装数据库Postgres

参照： https://developer.aliyun.com/article/1240243

查询

```
docker search postgres
```

下载

```
docker pull postgres
```

安装

```
rm -rf /apps/postgresql
mkdir -p /apps/postgresql
```

创建容器

```
   docker run -id --restart=always --name=postgresql -e TZ="Asia/Shanghai" -e LANG=C.UTF-8 -p 543
```

另外一种创建文件夹方式：

```
docker volume create postgre-data
```

volumes(卷挂载)：存储在主机文件系统的一部分中，该文件系统由Docker管理 （在Linux上是“/var/lib/docker/volumes /”）。 非Docker进程不应修改文件系统的这一部分。 卷是在Docker中持久存储数据的最佳方法。

docker自动在外部创建文件夹自动挂载容器内部指定的文件夹内容(Dockerfile VOLUME指令的作用)

为redmine创建db和账号

注意owner redmine必须有，官方的文档没有 `owner redmine` ，在最新的pg报错。猜测是文档当时的PG是没错的，但最新的PG有错。
参照：https://github.com/sameersbn/docker-redmine

```
CREATE ROLE redmine with LOGIN CREATEDB PASSWORD 'PASSWORD';
CREATE DATABASE redmine_production owner redmine;
GRANT ALL PRIVILEGES ON DATABASE redmine_production to redmine;
```

安装redmine

下载

```
# docker pull sameersbn/postgresql //官方github的都有问题，最后还是选用链接自定义Db
docker pull sameersbn/redmine
```

创建挂载文件夹

```
# rm -rf /apps/redmine/postgresql  使用自定义Db，所以注释
rm -rf /apps/redmine/redmine
rm -rf /apps/redmine/redmine-logs
# mkdir -p /apps/redmine/postgresql    使用自定义Db，所以注释
mkdir -p /apps/redmine/redmine
mkdir -p /apps/redmine/redmine-logs
```

-p 确保目录名称存在，不存在的就建一个。

创建容器

```
docker run --name=redmine -id  --restart=always \
  --env='DB_ADAPTER=postgresql' \
  --env='DB_HOST=47.93.14.110'    --env='DB_PORT=5432' --env='DB_NAME=redmine_production' \
  --env='DB_USER=redmine' --env='DB_PASS=PASSWORD' \
  --publish=3000:80 --env='REDMINE_PORT=3000'     --env='SMTP_DOMAIN=www.163.com'    --env='SMTF
  --volume=/apps/redmine/redmine:/home/redmine/data \
  --volume=/apps/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine
```

- 之前测试的时候，共用一个映射文件夹，会报错,用之前最好执行rm 删除文件夹重新建：
  FATAL: password authentication failed for user "redmine",
- 下面这个容器创建没问题，也可以访问，但是没有权限建表
  docker run --name=postgresql-redmine -d --publish=5434:5432 -e TZ="Asia/Shanghai" --restart=always
  --env='DB_NAME=redmine_production'
  --env='DB_USER=redmine' --env='DB_PASS=password'

--volume=/apps/redmine/postgresql:/var/lib/postgresql
sameersbn/postgresql

- redmine容器创建，链接上面的数据库，访问的时候会报错：
  PG::InsufficientPrivilege: ERROR: permission denied for schema public
  docker run --name=redmine -it --rm --link=postgresql-redmine:postgresql
  --volume=/apps/redmine/redmine:/home/redmine/data
  --volume=/apps/redmine/redmine-logs:/var/log/redmine/
  sameersbn/redmine

最后放弃了，还是使用官方的postgresql，和其他web公用一个db。

测试:http://47.93.14.110:3000/

redis安装

拉取镜像

```
docker pull redis
```

启动容器

```
docker run  -d --restart=always --name redis   -p 6379:6379  redis
```

Nginx

参考：
https://blog.csdn.net/baidu_21349635/article/details/102738972
https://blog.csdn.net/BThinker/article/details/123507820

创建本地文件夹

```
rm -rf /apps/nginx
mkdir -p /apps/nginx/
```

创建临时容器

```
docker run --name tmp-nginx-container -d nginx
```

拷贝容器内配置文件到本地

```
docker cp tmp-nginx-container:/etc/nginx/nginx.conf /apps/nginx/nginx.conf
docker cp -a tmp-nginx-container:/usr/share/nginx/html /apps/nginx/html
docker cp -a tmp-nginx-container:/var/log/nginx /apps/nginx/logs
```

删除临时容器

```
docker rm -f tmp-nginx-container
```

创建容器

```
docker run -d --restart=always -p 80:80 --name nginx -e TZ="Asia/Shanghai" -v /apps/nginx/html:/
```

如果产生错误：

Error starting userland proxy: listen tcp4 0.0.0.0:80: bind: address already in use.

查看端口占用情况

```
netstat -tanlp
```

假如端口被30016占用

```
kill 30016
```

打开配置文件

```
vim /apps/nginx/nginx.conf
```

代理redmine

```
    server{
                    listen 80;
                    server_name 47.93.14.110;
                    charset utf-8;
                    location /{
                            root /usr/share/nginx/html;
                            index index.html index.htm;
                    }
                    location /redmine {
                            proxy_pass http://47.93.14.110:3000/;
                            sub_filter '"/' '"/redmine/';
                            sub_filter '/favicon.ico' '/redmine/favicon.ico';
                            sub_filter_once off;
                            # proxy_redirect default;
                    }
            }
```

测试:http://47.93.14.110/redmine