# Docker遇到问题与解决方案

1. Docker安装版本

选择docker-ce版本进行安装

Docker分为社区版本与企业版本，Docker-ce版本支持更多平台，企业版基本支持Linux

ce版本功能更多

ce版本开源免费

1. Docker选择指定龙芯源

在run命令后添加cr.loongson.cn

1. Docker报错拒绝当前用户访问Docker引擎下的unix socket

sudo gpasswd -a $USER docker//将当前用户添加至docker组中

newgrp docker

重新执行命令

1. Docker配置适用于龙芯的镜像源

sudo vim /etc/docker/daemon.json

添加语句：

{

“registry-mirror”:[“https://cr.loongnix.cn”]

}

退出并保存

systemctl daemon-reload//加载daemon配置文件

systemctl restart docker//重启docker

1. 修改daemon文件后重启docker报错

cd /etc/docker/daemon.json

mv daemon.json daemon.conf

systemctl restart docker

systemctl status docker

1. Docker查询及删除镜像及容器命令

查询镜像 docker images -a

查询容器 docker ps -a

删除镜像 docker rmi -f ID

删除容器 docker rm -f ID

批量删除全部非运行容器 docker rm $(docker ps -a -q) || docker container prune

批量删除全部非运行镜像 docker rmi $(docker images -a -q)

1. 若配置文件未配置龙芯源，使用源中镜像需要先从源中
   1. docker run “源地址”/.../...
   2. docker pull “源地址”/.../...
2. CMD和ENTRYPOINT的区别（均只能执行一条最后的命令）
   1. CMD作为默认执行的第一个命令，不能在启动容器后添加参数
   2. ENTRYPOINT作为程序的入口点，可以启动容器后添加参数
   3. CMD支持第三种格式 CMD [“arg1”,”arg2”]
3. /go/pkg/mod/golang.org/x/sys@v0.0.0-20191120155948-bd437916bb0e/unix/affinity\_linux.go:14:20: undefined: \_CPU\_SETSIZE

问题：缺少外部依赖

解决方案：更新外部依赖

1. go get -u golang.org/x/...缺少的依赖
2. go mod tidy
3. 重新操作
4. cr 镜像提交流程 push与pull 不要用sudo命令！！！
5. docker images | head -n 10 查看要提交的镜像
6. docker tag 原镜像名:版本 当前要提交的镜像名:版本
7. docker login -u -p
8. docker push 当前要提交的镜像名:版本

shanjiantao dota6.78b

docker login -u -p

docker tag

docker push

1. error: RPC failed; curl 56 GnuTLS recv error (-110): Error decoding the receiv

类似问题解决方案：

git config --global http.postBuffer 524288000

git config --global https.postBuffer 524288000

git config --global corepression -1

1. make段错误

一般是架构不兼容 找到Makefile中对应配置环境变量的shell脚本或文件，进行添加

如LOCAL\_ARCH = loongarch64 TARGET\_ARCH = loong64

1. 注意在Makefile及其引用文件中的BUILD\_WITH\_CONTAINER := 0而不是 ?=0
2. go 出现build等命令未找到错误时 需要重新source或查看环境变量是否配置，重启bash
3. set -eux中的u慎用 u代表未定义变量也当作错误
4. 出现batch response:XXX......proxyconnect tcp: EOF时说明代理影响了下载

取消代理命令

unset http\_proxy

unset https\_proxy

设置代理命令

export http\_proxy=ip:port

export https\_proxy=ip:port

1. 在make过程 中出现了需要go1.18以下版本的依赖，因此需要先删除go1.19 安装go1.17
2. 当前目录下有多个版本更换时，需要删除usr/bin下的go和gofmt软连接，然后添加软连接

安装步骤：

1. 卸载go1.19 删除环境变量

删除go目录(whereis go 寻找目录) sudo rm -rf 路径

删除软链接 sudo rm -rf /usr/bin/go

env 查看当前环境变量，删除GOROOT、GOPROXY、GOPATH等

1. 安装1.17之前的版本，apt-get install golang命令安装
2. 查看已安装的1.15路径 locate go 在usr/lib下
3. 找到lib/go-1.15.6 文件夹 sudo cp go-1.15.6 /usr/local/go-1.15.6 -rp
4. 在/usr/local解压1.17 进入sudo su模式
5. 设置export GOROOT\_BOOTSTRAP=/usr/local/go1.15.6
6. 进入src文件夹 ./make.bash 执行 得到编译好的go
7. 执行结束后将当前版本文件夹修改为go
8. 添加软连接和环境变量

sudo ln -s /usr/local/go/bin/\* /usr/bin/

创建工作空间

mkdir -p $HOME/go-workspace/src

mkdir -p $HOME/go-workspace/pkg

mkdir -p $HOME/go-workspace/bin

配置环境变量

sudo vim $HOME/.bashrc

export GOROOT=/usr/local/go #设置为go安装的路径，有些安装包会自动设置默认的goroot

export GOPATH=$HOME/go-workspace #默认安装包的路径

export PATH=$PATH:$GOROOT/bin:$GOPATH/bin

source $HOME/.bashrc　　#注：这里不要用sudo执行，sudo无该命令

1. 冒泡测试流程

在本机运行镜像查看是否可用 —— 在x86机器下载dockerhub中的镜像并运行查看是否相同并查看部分依赖可执行文件的version方法 —— 在本机查看当前依赖的可执行文件是否相同

本机：docker run -it --rm IMAGEID||IMAGEL:TAGS

x86：[ssh loongson@10.2.10.204](mailto:ssh@10.2.10.204) password:loongson

docker push 地址

docker run -it --rm IMAGE:TAGS -v

docker run -it --rm --entrypoint /bin/bash IMAGE:TAGS

进入容器后ls查看对应依赖二进制文件名 执行 文件名 --help

本机：docker run -it --rm --entrypoint /bin/bash IMAGE:TAGS

进入容器后ls查看对应依赖二进制文件名 执行 文件名 --help

对比x86查看是否相同

1. 若想让make过程中的指令执行过程展示，需要修改的部分

Makefile文件 /bin/bash ——》 /bin/bash -x

.mk文件 -set -e ——》 -set -ex

.sh文件 -set -e ——》 -set -ex

1. 遇到git 443端口拒绝访问时修改git配置

git config --global http.proxy http://10.130.20:7890

git config --global https.proxy http://10.130.20:7890

取消

git config --global --unset http.proxy

git config --global --unset https.proxy

查看

git config --list

1. 遇到OPENSSL 或 提交失败时，重新设置系统全局http代理

export http\_proxy=”http://10.130.20:7890”

export https\_proxy=”http://10.130.20:7890”

打印全局变量 echo $变量名

取消代理 unset 代理名

1. 出现cp /usr/local/bin 中不存在某个文件(file do not exist)

查看项目的Makefile文档，可能是当前需要的某个二进制文件通过Makefile中的命令进行下载时出现了问题，如网络问题，代理问题或者下载链接不对

适当查看文档，修改部分代码，包括Makefile文档、执行命令对应的shell脚本等。

1. 遇到gitClone报错：GnuTLS recv error (-110): The TLS connection was non-properly terminated.的解决方法
   1. 尝试取消重设git 代理
   2. clone时设置git clone -b x.x.x githttp --depth=1
2. linux环境下配置java环境变量

检查当前JAVA\_HOME echo $JAVA\_HOME

寻找本机java位置 find / -name java

配置环境变量

sudo vim ～/.bashrc

export JAVA\_HOME=/usr/bin/java

export JRE\_HOME=$JAVA\_HOME/jre

export CLASSPATH=$JAVA\_HOME/lib:$JRE\_HOME/lib:$CLASSPATH

export PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$PATH

source ~/.bashrc

1. ./autogen.sh:行47: autoreconf: 未找到命令
   1. yum -y install autoconf automake libtool

这里同时安装了三个命令，不安的话后续都会报错

1. yum的第三方源有部分是默认禁用状态，可以下载两个命令，都可以解决
   1. yum install dnf-plugins-core
   2. yum install yum-utils

接着 yum config-manager --set-enabled powertools(一定是源的小名字而非.repo后缀)

1. 缺少libstdc++.so
   1. ld -verbose //查找当前ld依赖库
   2. find / -name libstdc++.\* | grep ltdc++ //搜索所有libstdc++相关依赖
   3. 尝试添加软链接，并在添加链接后构建项目查看是否报错

ln -s 原地址 /usr/.../....so

* 1. 不可用则使用rm命令删除
  2. yum install glibc -static libstdc++ -static -y 安装
  3. 再次尝试

1. 存在error while loading shared libraries: libprotobuf.so.20: cannot open shared错误

说明此时系统找不到对应的lib库，可以ldconfig刷新缓存再试，不可以的话有如下两种方法，前提是找到当前需要的lib文件存储的位置，本问题存储于/usr/lib/下

1. 修改配置文件

修改～/.bashrc文件 添加 export LD\_LIBRARY\_PATH=路径:$LD\_LIBRARY\_PATH

刷新 source ~/.bashrc

1. 修改配置文件2
2. 构建成功后jar包不一定在根目录下的target中，可能是多个子目录的target中包含不同的jar包
3. 构建过程中configure: error: cannot guess build type; you must specify one

说明当前构建工程无法识别本地系统，需要将config.guess和config.sub复制到当前工程的根目录或build目录下，搜索系统中两个文件的命令(find /usr -name config.guess/sub)

1. 在容器中使用systemctl失败
   1. 将当前容器commit到镜像 docker commit 容器ID 新镜像名称
   2. 运行新的镜像 docker run -itd --privileged=true --name 新容器名 新镜像名 /usr/sbin/init #使用镜像“my-centos”以“特权”后台交互式运行名为“my\_centos”的容器执行“/usr/sbin/init”命令，并返回该容器的ID
   3. 进入 docker exec -it 容器ID||容器名 /bin/bash
   4. 可以执行
2. 容器内启动docker服务失败，先看systemctl能否使用，然后再看报错信息
3. 将容器文件复制到本地
   1. docker cp ID:路径 本地路径
   2. m2 和 gradle
4. 如果java中某个命令不可执行，在ftp中找到合适java版本，wget到本地，解压到usr/local下，配置环境变量到profile中，source，在查看是否可执行