# FastDFS客户端的链码调用

## 0 概述

本文档介绍如何将已经搭建好的FastDfs集群，实现对fabric链码搭的调用。

## 1 前提条件

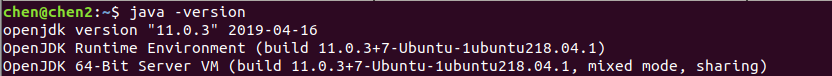
1. 完成FastDfs集群的搭建，并实现用java调用，详细过程见我的另两篇文章《搭建分布式FastDFS集群》和《FastDFS-JAVA-Client使用示例》。
2. 在你的虚拟机中完成docker的安装，以保证链码能够正常的运行。
3. 完成fabric链码的搭建

在搭建fabric链码之前需要完成的工作：

1. 在ubuntu系统中完成java的安装，并配好相应的路径。

<https://www.cnblogs.com/freescience/p/7272070.html>

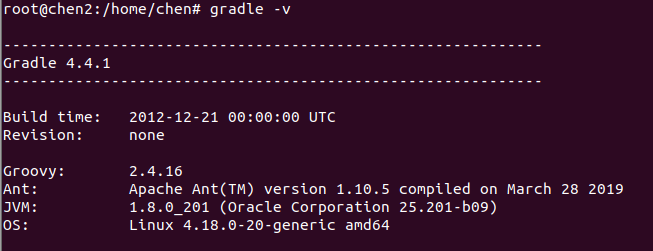
安装完成之后执行，java -version命令看看到对应的安装版本。



1. 在ubuntu系统中安装gradle，用于编译并构建链码。

注：此处gradle的安装只能安装4.4.x版本，其他版本的gradle在编译时会发生报错。

在ubuntu系统中执行 apt install gradle ,系统默认安装的是4.4.1版本。安装完成后执行gradle -v命令查看当前安装版本。（此处安装建议切换到root用户下进行安装）



1. 下载对应的链码。

wget <https://github.com/Chenkeyu1126/TestnetFabricDmeo>

## 2 安装链码

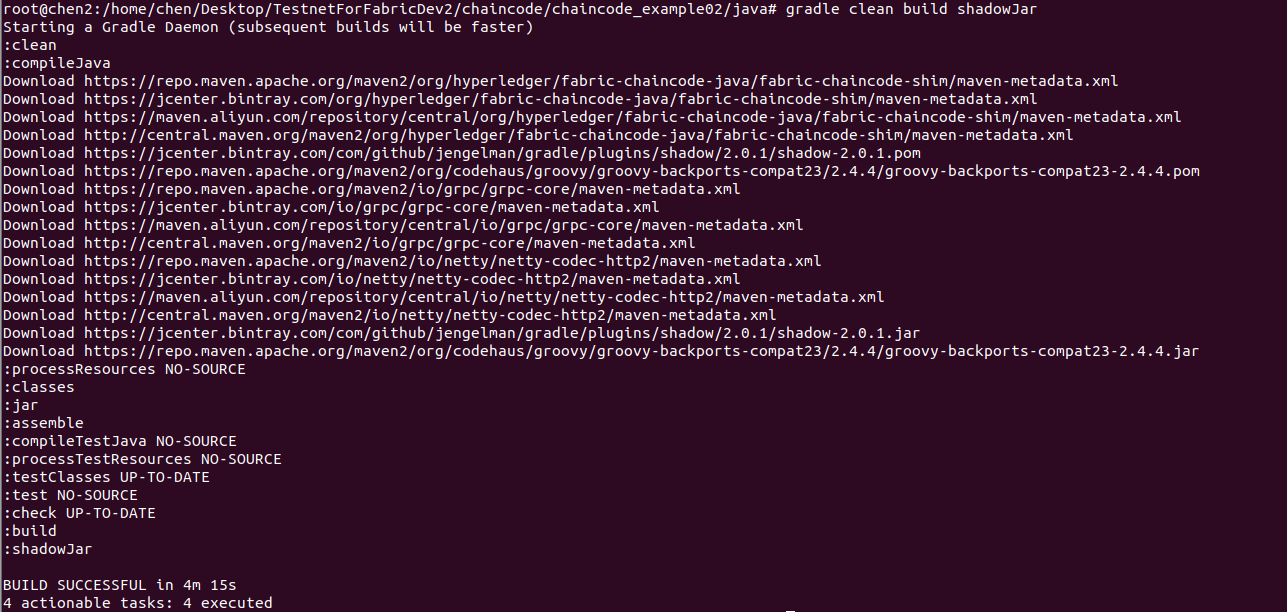
1)下载完成后，解压文件，进入如下目录中，并执行如下命令：

# tar -xvf TestnetForFabricDev.tar.gz

# cd ./TestnetForFabricDev2/chaincode/chaincode\_example02/java

# gradle clean build shadowJar

执行完成后可以看到如下界面：



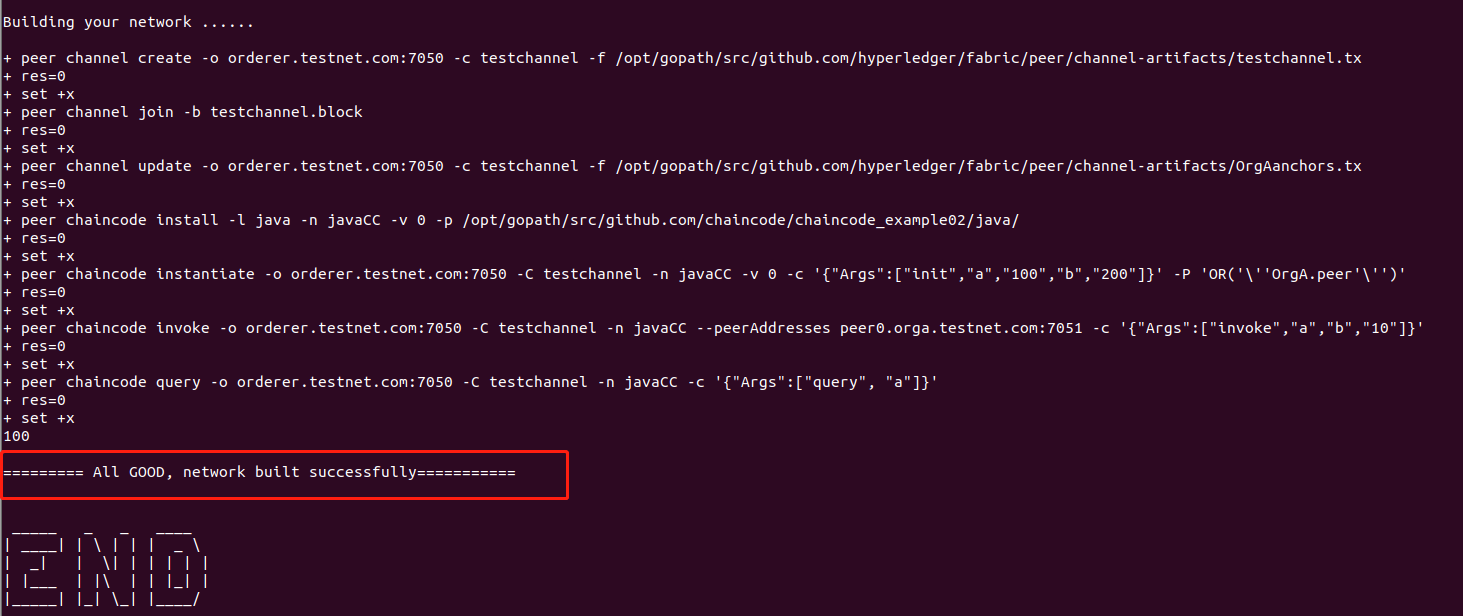
如果编译失败，就必须根据提示的错误信息，修改源码，直到编译成功为止。

2) 进入到如下目录，执行命令：

# cd ./TestnetForFabricDev2/

# ./testnet up

此步操作可以自动的完成链码的构建，执行完成之后，如图所示：



看到这个画面说明你的链码已经构建成功。

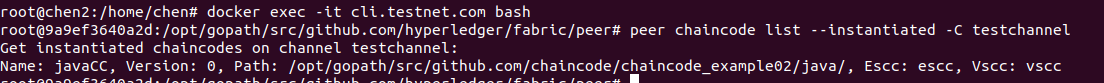
1. 进入cli.testnet.com容器中

# docker exec -it cli.testnet.com bash

1. 查看当前安装的链码

$ peer chaincode list --instantiated -C testchannel

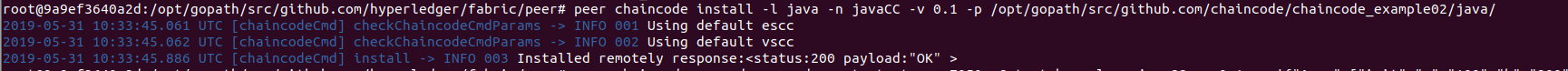
你将看到类似如下输出：



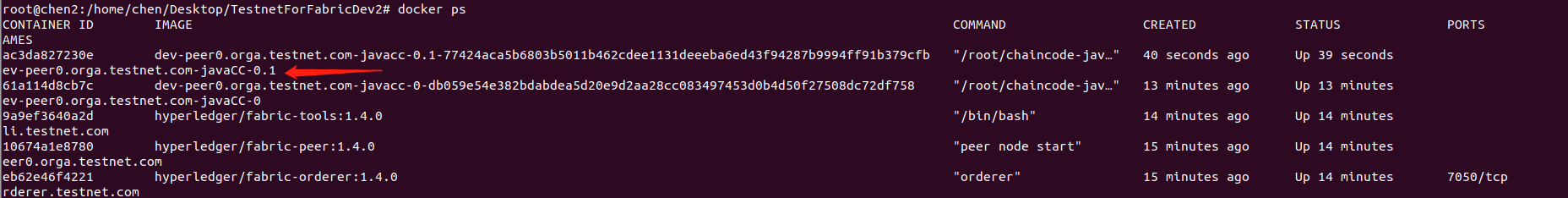
 该输出指明了，当前节点上安装的链码，名称为javaCC，版本号为0，源码路径为/opt/gopath/src/github.com/chaincode/chaincode\_example02/java/。

1. 安装升级链码

$ peer chaincode install -l java -n javaCC -v 0.1 -p /opt/gopath/src/github.com/chaincode/chaincode\_example02/java/         # 安装链码



 $ peer chaincode upgrade -o orderer.testnet.com:7050 -C testchannel -n javaCC -v 0.1 -c '{"Args":["init","a","100","b","200"]}' -P 'OR('\''OrgA.peer'\'')'        # 升级链码



执行完之后可以看见版本号为0.1的peer容器已经生成了。

注： 该过程中，链码的版本号，即 -v 所指定的参数，必须和之前链码的不一样，之前链码的版本号为 0 ，则新安装的链码版本号必须为 0.1 或者 1、2等。并且所有新安装的版本号不能和之前的所有安装过的版本号重复，比如升级过多次链码，之前的版本号分别为，0.1、0.2、1.1，则这些版本号以后都不可以再次重复使用。

升级链码的操作，即 peer chaincode upgrade 执行的时候，其后边要跟链码调用的参数，即 -c '{"Args":["init","a","100","b","200"]}' 部分，该部分的参数定义，其实就是链码中 init 方法所定义的内容，也就是说，执行 peer chaincode upgrade 的操作，其实调用了链码中的 init 方法。如果修改了 init 方法的定义，则执行升级操作的时候，-c 所跟的参数需要根据链码中 init 方法所定义的内容进行修改。

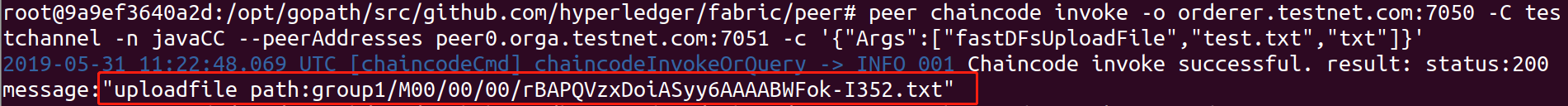
## 3 调用链码

示例链码中提供了三个功能：invoke，query，delete分别为链码执行功能、查询功能、删除功能。另外在链码中新增加了三个方法fastUploadFile、fastDownloadFile、fastDFsDeleteFile、fastDFsBuildFile，分别可以实现文件的上传、下载、删除、创建新文件的操作。

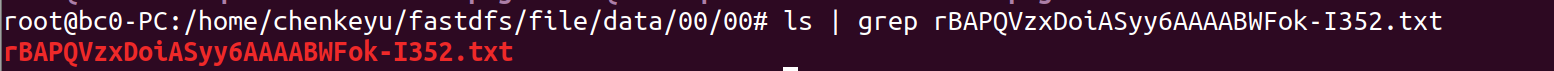
1. 上传文件

$ peer chaincode invoke -o orderer.testnet.com:7050 -C testchannel -n javaCC --peerAddresses peer0.orga.testnet.com:7051 -c '{"Args":["fastDFsUploadFile","test.txt","txt"]}'

利用链码原有的invoke函数实现对方法fastDFsUploadFile的调用，其中test.txt和txt分别为上传文件的路径和文件的类型。



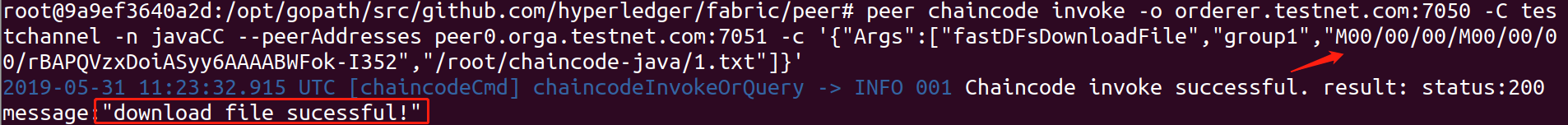
此时进入我们之前搭建的FastDFs的storage中可以查看到我们的文件已经上传成功，且文件名和上图返回值一样。



1. 下载文件

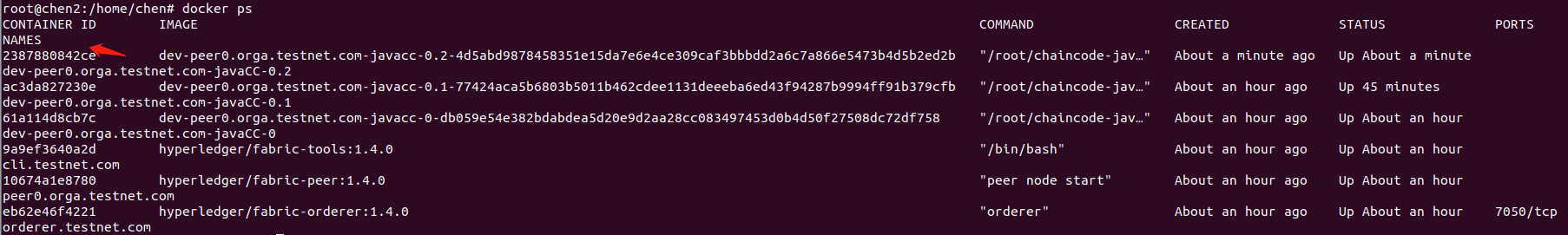
$ peer chaincode invoke -o orderer.testnet.com:7050 -C testchannel -n javaCC --peerAddresses peer0.orga.testnet.com:7051 -c '{"Args":["fastDFsDownloadFile","group1","M00/00/00/M00/00/00/rBAPQVzxDoiASyy6AAAABWFok-I352.txt","/root/chaincode-java/1.txt"]}'

利用链码原有的invoke函数实现对方法fastDFsDownloadFile的调用，其中group1、M00/00/00/M00/00/00/rBAPQVzxDoiASyy6AAAABWFok-I352和/root/chaincode-java/1.txt分别为下载文件的组名、路径和下载文件的存放路径。



此时可以进入order容器中查看该文件是否下载成功

$ docker ps 查看正在运行的容器



找到peer所对应的容器id。

$ docker exec -it container.id bash 进入容器中

$ ls 查看当前容器路径下的文件



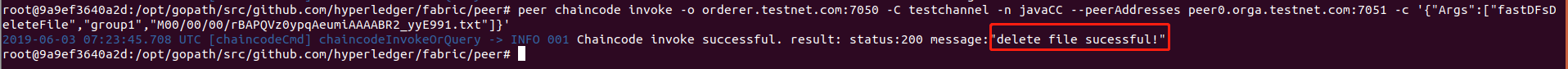
可以看到我们下载的文件1.txt，已经成功下载到了本地。

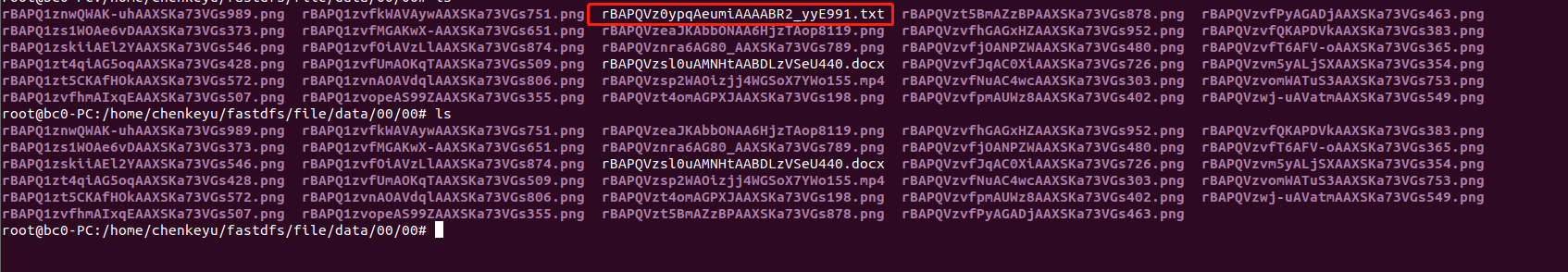
1. 删除文件

$ peer chaincode invoke -o orderer.testnet.com:7050 -C testchannel -n javaCC --peerAddresses peer0.orga.testnet.com:7051 -c '{"Args":["fastDFsDeleteFile","group1","M00/00/00/rBAPQVz0ypqAeumiAAAABR2\_yyE991.txt"]}'

利用链码原有的invoke函数实现对方法fastDFsDeleteFile的调用，其中group1和M00/00/00/rBAPQVz0ypqAeumiAAAABR2\_yyE991.txt分别为下载文件的组名、路径。

执行以上命令可以在搭建好的fastDFS集群发现文件已经被删除。

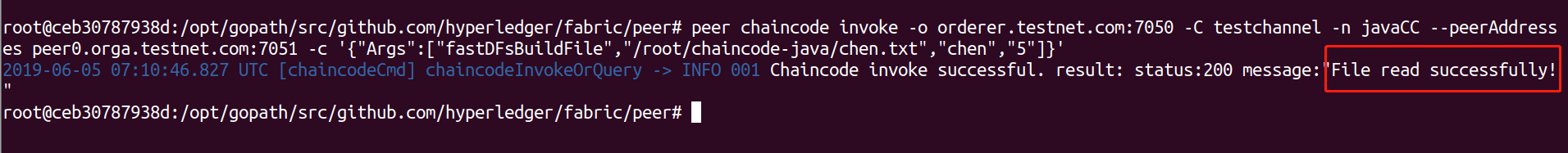




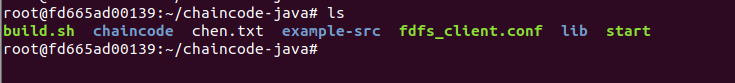
1. 创建新文件并指定大小

peer chaincode invoke -o orderer.testnet.com:7050 -C testchannel -n javaCC --peerAddresses peer0.orga.testnet.com:7051 -c '{"Args":["fastDFsBuildFile","/root/chaincode-java/chen1.txt","chen","5"]}'

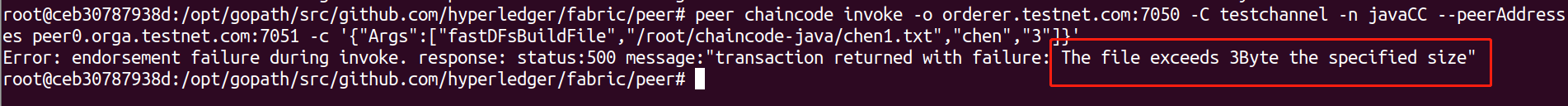
利用链码原有的invoke函数实现对方法fastDFsBuildFile的调用，其中/root/chaincode-java/chen.txt、chen、5分别为创建文件指定的路径、文件插入的内容以及文件大小限制（单位：字节）。



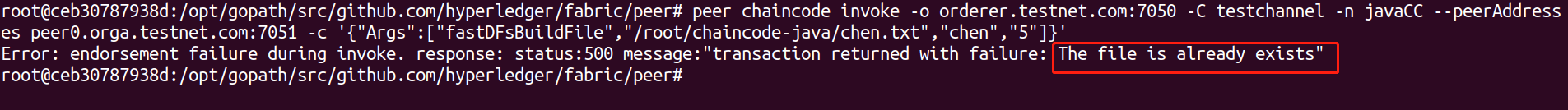
可以看到文件创建成功，进入peer容器查看：



如果执行过程中输入的内容超过了设定的大小就会发生报错。



如果创建的文件已存在，也会报错。



## 4 关闭测试网络

# cd TestnetForFabricDev2

# ./testnet.sh down