关于ulimit

1）修改当前交互终端的limit值

    查询当前终端的文件句柄数： ulimit -n 回车，一般的系统默认的1024.

    修改文件句柄数为65535，ulimit -n 65535.此时系统的文件句柄数为65535.

    2）将ulimit 值添加到/etc/profile文件中（适用于有root权限登录的系统）

    为了每次系统重新启动时，都可以获取更大的ulimit值，将ulimit 加入到/etc/profile 文件底部。

    echo ulimit -n 65535 >>/etc/profile

    source /etc/profile    #加载修改后的profile

    ulimit -n     #显示65535，修改完毕！

到此步只是提高某一次执行脚本的文件句柄打开数，当用户再次登录进来的时候，ulimit的值还是1024。原因在于登录的用户身份，是不是root用户，由于服务器的root用户权限很大，一般是不能用来登录的，都是通过自己本人的登录权限进行登录，并通过sudo方式切换到root用户下进行工作。 用户登录的时候执行sh脚本的顺序：

    /etc/profile.d/file

    /etc/profile

    /etc/bashrc

    /mingjie/.bashrc

    /mingjie/.bash\_profile

由于ulimit -n的脚本命令加载在第二部分，用户登录时由于权限原因在第二步还不能完成ulimit的修改，所以ulimit的值还是系统默认的1024。

解决办法：

    修改linux的软硬件限制文件/etc/security/limits.conf.

    在文件尾部添加如下代码：

    \* soft nofile 65535

    \* hard nofile 65535

    保存并推出，命令说明：

    domino type item value

    domino是以符号@开头的用户名或组名，\*表示所有用户，type设置为hard or soft。item指定想限制的资源。如cpu,core nofile nproc or maxlogins。value是相应的限制值。

     保存后，只需要重新登录一次，ulimit的值就生效了，如果还不行就 reboot（重新启动服务器）

安装rabbitmq

1.进入目录:

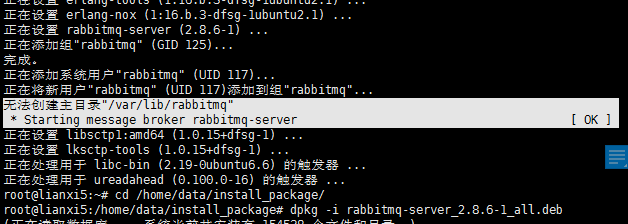
cd /home/data/install\_package

执行命令:

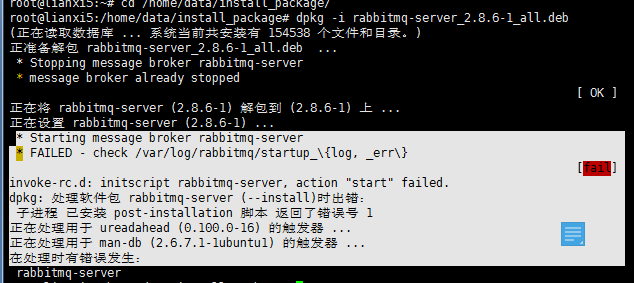
dpkg -i rabbitmq-server\_2.8.6-1\_all.deb时，包错误未安装软件包 erlang-nox



返回再次执行apt-get -f install，出现以下错误



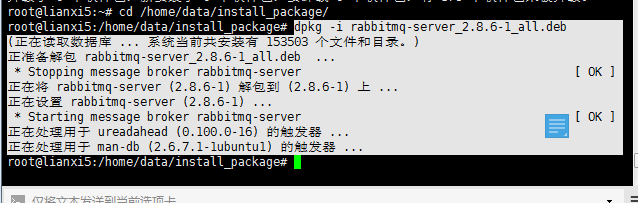
未加理会，第二次执行dpkg -i rabbitmq-server\_2.8.6-1\_all.deb，报错



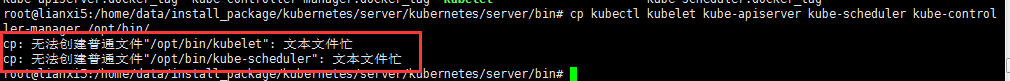
使用**sudo apt-get autoremove --purge 卸载了**libncurses5-dev和python-pip，然后从第一步从头执行，成功了

**sudo apt-get autoremove --purge** libncurses5-dev

**sudo apt-get autoremove --purge** python-pip



问题：

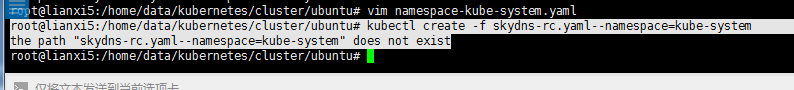


解决办法：

修改util文件，把所有cp改为cp -f，强行覆盖

cp -f kubectl kubelet kube-apiserver kube-scheduler kube-controller-manager /opt/bin/

问题：



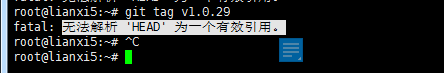
解决办法

问题：



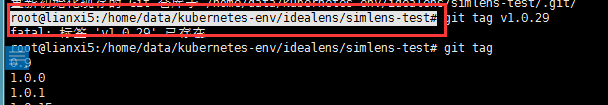
原因：前面第3/7/12/16编辑配置文件错误，根据配置文档修改之即可

问题：



解决办法：

执行目录错误，在如图目录下执行目录



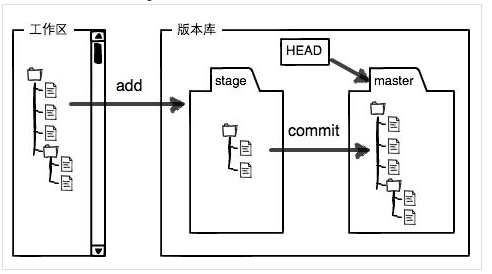
资料查询：

首先了解到 HEAD是个什么东西，有什么作用

**工作区**

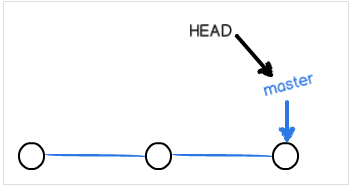
就是你在电脑里能看到的目录  
**版本库**

工作区有一个隐藏目录.git,就是Git的版本库



Git 分支：每次提交，Git都把它们串成一条时间线，这条时间线就是一个分支，所以分支就是一条commi时间线。

HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是**当前分支。**



然后了解到 git tag 标签管理

一个版本库保存一个分支master，各自的版本库保存着各自的分支，各自的主分支名称都为master。

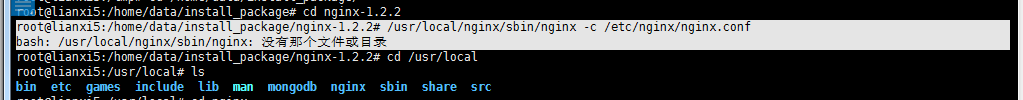
发布一个版本时，我们通常先在版本库中打一个标签，这样，就唯一确定了打标签时刻的版本。将来无论什么时候，取某个标签的版本，就是把那个打标签的时刻的历史版本取出来。所以，标签也是版本库的一个快照

标签是指向某个commit的指针  
**打标签**:git tag <name> //**默认**标签是打在最新提交的commit上



**查看标签**：git tag  
**查看标签信息**：git show <tagname>  
**删除标签**：git tag -d <tagname>//(本地)

问题：



原因：

前面未执行第10步，执行命令make install

解决办法：

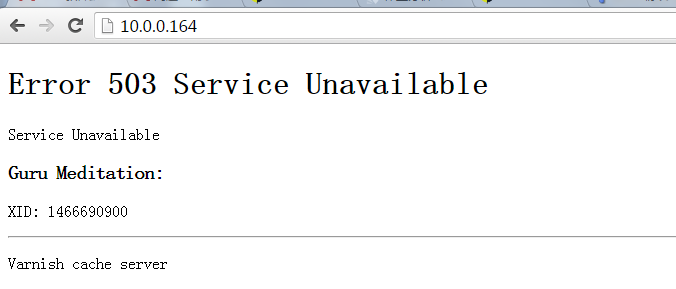
从开头重新执行步骤

PS:

经检查，在/usr/local目录下无nginx目录，此处为添加包含子目录的目录 [译注：递归地创建目录]，故需要使用 -p 参数

问题：

访问503



原因：经检查，kubernetes未启动成功，重启Kubernetes也仅启动一般服务，故重新执行kubernetes的安装后解决此问题。