一：mesos环境搭建

1. 准备工作
2. 五台机器Ubuntu16.04 、JDK1.8系统(其中dev\_mesosagent01是master，dev\_mesosagent02是slave)。
3. 10.1.2.62 dev\_mesosagent01

10.1.2.63 dev\_mesosagent02

3）10.1.2.59 **dev\_zookeeper\_srv01**

10.1.2.60 **dev\_zookeeper\_srv02**

10.1.2.61 **dev\_zookeeper\_srv03**

1. 编译安装Mesos

**在10.1.2.62机器上安装mesos-master**

1），Download the latest stable release from Apache (Recommended)

进去目录cd/usr/lib 下载并解压。

$ wget http://www.apache.org/dist/mesos/1.6.0/mesos-1.6.0.tar.gz

$ tar -zxf mesos-1.6.0.tar.gz

2），在Ubuntu 16.04下进行update，进入以下命

# Update the packages.

$ sudo apt-get update

# Install a few utility tools.

$ sudo apt-get install -y tar wget git

# Install the latest OpenJDK.

$ sudo apt-get install -y openjdk-8-jdk

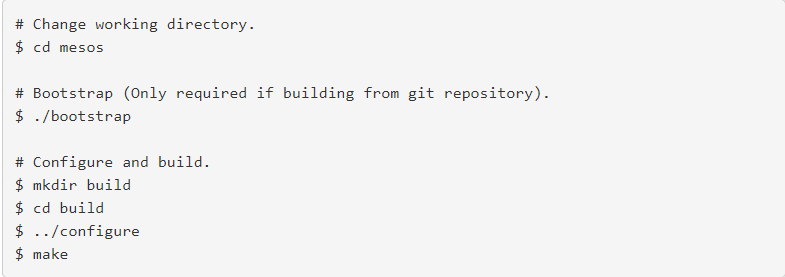
# Install autotools (Only necessary if building from git repository).

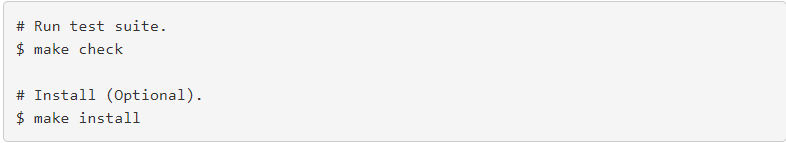
$ sudo apt-get install -y autoconf libtool

# Install other Mesos dependencies.

$ sudo apt-get -y install build-essential python-dev python-six python-virtualenv libcurl4-nss-dev libsasl2-dev libsasl2-modules maven libapr1-dev libsvn-dev zlib1g-dev iputils-ping

3），Building Mesos



如果编译error,就使用make clear命令，再重新make

4），ExamplesTest

进入build目录来启动mesos-master

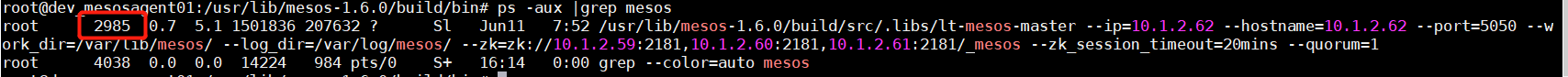
进入以下命令：

**./bin/mesos-master.sh --ip=10.1.2.62 --port=5050 --work\_dir=/var/lib/mesos --hostname=10.1.2.62**

注：其中--work\_dir 存放一些持久化的文件、以及一些framwork的日志文件，ip和hostname存放本台机器mesos-master的地址

查看此mesos-master是否启动成功？？

用命令ps -aux|grep mesos来查看



**在10.1.2.63机器上安装mesos-slave**

1），安装步骤，流程和mesos-master一致。

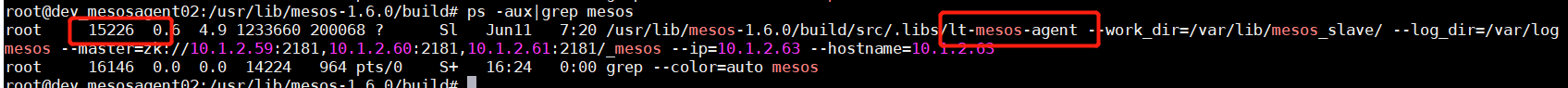
2），启动mesos-slave

进入build目录，

**./bin/mesos-agent.sh --master=10.1.2.62:5050 --work\_dir=/var/lib/mesos**

注：--master代表本台机器mesos-slave是安装在mesos-master下面。

继续查看进程:ps -aux|grep mesos



测试：

并且在浏览器上输入：10.1.2.62:5050看webUI页面

**以上mesos-master和mesos-slave环境搭建和测试完毕。**

注：以上只是针对单台机器，如果出现单台机器宕机，资源就没办法继续利用了为了实现mesos的高可资源最大化利用，我们采用了zookeeper解决方案：

1. Zookeeper使用场景和结构

现在很多时候我们的服务需要7\*24小时工作，假如一台机器挂了，我们希望能有其它机器顶替它继续工作。此类问题现在多采用master-salve模式，也就是常说的主从模式，正常情况下主机提供服务，备机负责监听主机状态，当主机异常时，可以自动切换到备机继续提供服务(这里有点儿类似于数据库主库跟备库，备机正常情况下只监听，不工作)，这个切换过程中选出下一个主机的过程就是master选举。

对于以上提到的场景，传统的解决方式是采用一个备用节点，这个备用节点定期给当前主节点发送ping包，主节点收到ping包后会向备用节点发送应答ack，当备用节点收到应答，就认为主节点还活着，让它继续提供服务，否则就认为主节点挂掉了，自己将开始行使主节点职责。

1. zookeeper的master选举机制

在单个Mesos master节点的集群中，如果Mesos master节点故障，或者服务不可用，虽然在每一个Slave节点上的任务可以继续运行，但是集群中新的资源将无法再分配给上层Framework，上层Framework将无法再利用已经收到的offer提交新任务，并且无法收到正在运行任务的状态更新。为了解决这个问题，提高Mesos集群的高可用性，减少Mesos master节点故障所带来的影响，Mesos集群采用了传统的主备冗余模式（Active－Standby）来支持一个Mesos集群中部署多个Mesos master节点，**借助于ZooKeeper进行Leader的选举**。选举出的Leader负责将集群的资源以契约（offer）的形式发送给上层的每一个Framework，并处理集群管理员与上层Framework的请求，另外几个Mesos master节点将作为Follower一直处于备用状态，并监控当前的状态，当Mesos master节点宕机，或服务中断之后，新Leader将会很快从Follower中选出来接管所有的服务，减少了Mesos集群服务的宕机时间，大大提高了集群的可用性。

那么基于以上的场景 我们接下来就要利用zookeeper来实现mesos的高可利用度来满足服务器宕机的问题。

1. 搭建zookeeper。

1），在机器10.1.2.59上搭建zookeeper环境

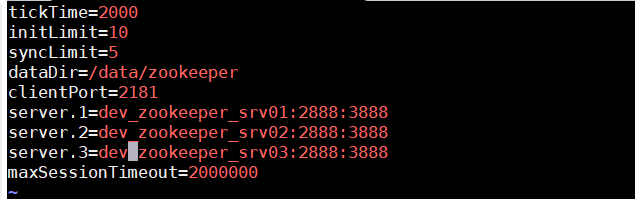
进入/usr/lib目录下：下载并解压

# wget http://mirrors.shu.edu.cn/apache/zookeeper/zookeeper-3.4.12/zookeeper-3.4.12.tar.gz

# tar -zxf zookeeper-3.4.12.tar.gz

并重命名文件cp zoo\_example.cfg zoo.cfg

并进入VM编辑器，编写zookeeper的配置信息



dataDir: 是一个类似于snapshot.{16进制串}的内存数据结构目录，便于快速恢复，里面有一个myid文件，里面就一个数字，用来唯一标识这个服务。这个id是很重要的，一定要保证整个集群中唯一。zookeeper会根据这个id来取出server.x上的配置。比如当前id为1，则对应着zoo.cfg里的server.1的配置。

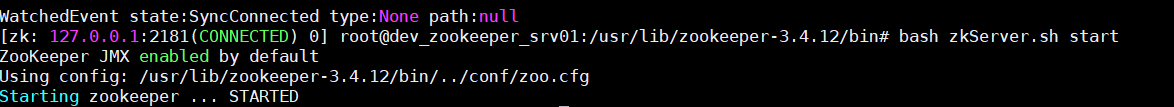
Server.1=dev\_zookeeper\_srv01:2888:3888

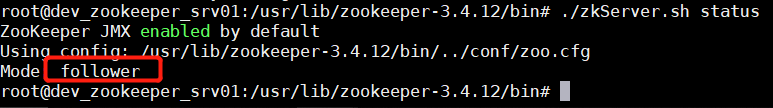
#server.A=B:C:D 其中A是一个数字，代表这是第几号服务器；B是服务器的IP地址；C表示服务器与群集中的“领导者”交换信息的端口；当领导者失效后，D表示用来执行选举时服务器相互通信的端口。

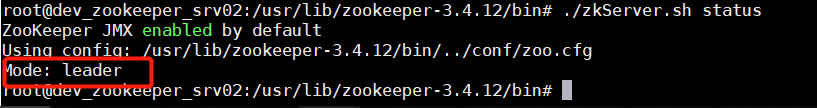
2），上面就是配置的讲解了，现在我们可以启动zookeeper集群了。进入到bin目录，执行 ./zkServer.sh start即可

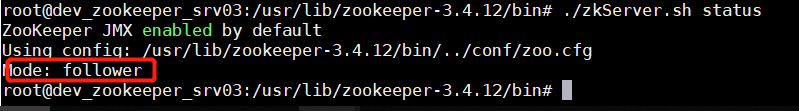
root@dev\_zookeeper\_srv01:/usr/lib/zookeeper-3.4.12/bin# **./zkCli.sh -timeout 5000 -server 127.0.0.1:2181**

**测试启动是否成功？有以下信息表示启动成功**









说明dev\_zookeeper\_srv01服务器此时处于follower模式，同样切换至dev\_zookeeper\_srv01/bin目录下执行相同的命令，说明dev\_zookeeper\_srv02服务器被选举为leader

4，重新启动mesos-master和mesos-agent（脚本上加上zookeeper的IP和port）

1），在/usr/lib/mesos-1.6.0/build目录下创建start-mesos-master.sh脚本文件

#!/bin/bash

ip=`ip a |grep 'global eth0'|awk '{print $2}'|awk -F '/' '{print $1}'`

/usr/lib/mesos-1.6.0/build/bin/mesos-master.sh --ip=$ip --hostname=$ip --port=5050 --work\_dir=/var/lib/mesos/ --log\_dir=/var/log/mesos/ --zk=zk://dev\_zookeeper\_srv01:2181,dev\_zookeeper\_srv02:2181,dev\_zookeeper\_srv03:2181/$1\_mesos --zk\_session\_timeout=20mins --quorum=1 &

log\_dir：用来存放mesos-master日志文件，

--mesos：用来 本台机器注册到三台zookeeper机器上对应的节点信息

2），同理得到mesos-agent的zookeeper的配置信息脚本文件start-mesos-agent.sh

#!/bin/bash

ip=`ip a |grep 'global eth0'|awk '{print $2}'|awk -F '/' '{print $1}'`

/usr/lib/mesos-1.6.0/build/bin/mesos-slave.sh --work\_dir=/var/lib/mesos\_slave/ --log\_dir=/var/log/mesos --master=zk://dev\_zookeeper\_srv01:2181,dev\_zookeeper\_srv02:2181,dev\_zookeeper\_srv03:2181/$1\_mesos --ip=$ip --hostname=$ip &

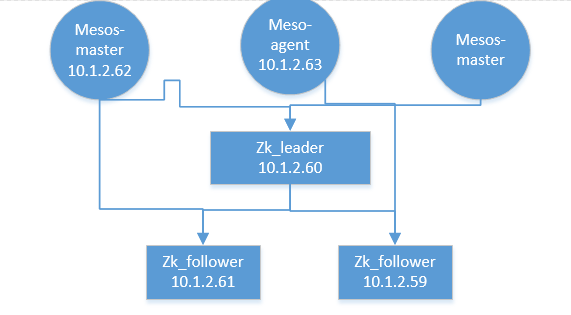
**Mesos-master，mesos-agent，zookeeper三者之间的交互联系**

#!/bin/bash

ip=`ip a |grep 'global eth0'|awk '{print $2}'|awk -F '/' '{print $1}'`

/usr/lib/mesos-1.6.0/build/bin/mesos-master.sh --ip=$ip --hostname=$ip --port=5050 --work\_dir=/var/lib/mesos/ --log\_dir=/var/log/mesos/ --zk=zk://dev\_zookeeper\_srv01:2181,dev\_zookeeper\_srv02:2181,dev\_zookeeper\_srv03:2181/$1\_mesos --zk\_session\_timeout=20mins --quorum=1 &

1. 先启动zookeeper，再通过start-mesos-master.sh启动Mesos-master，

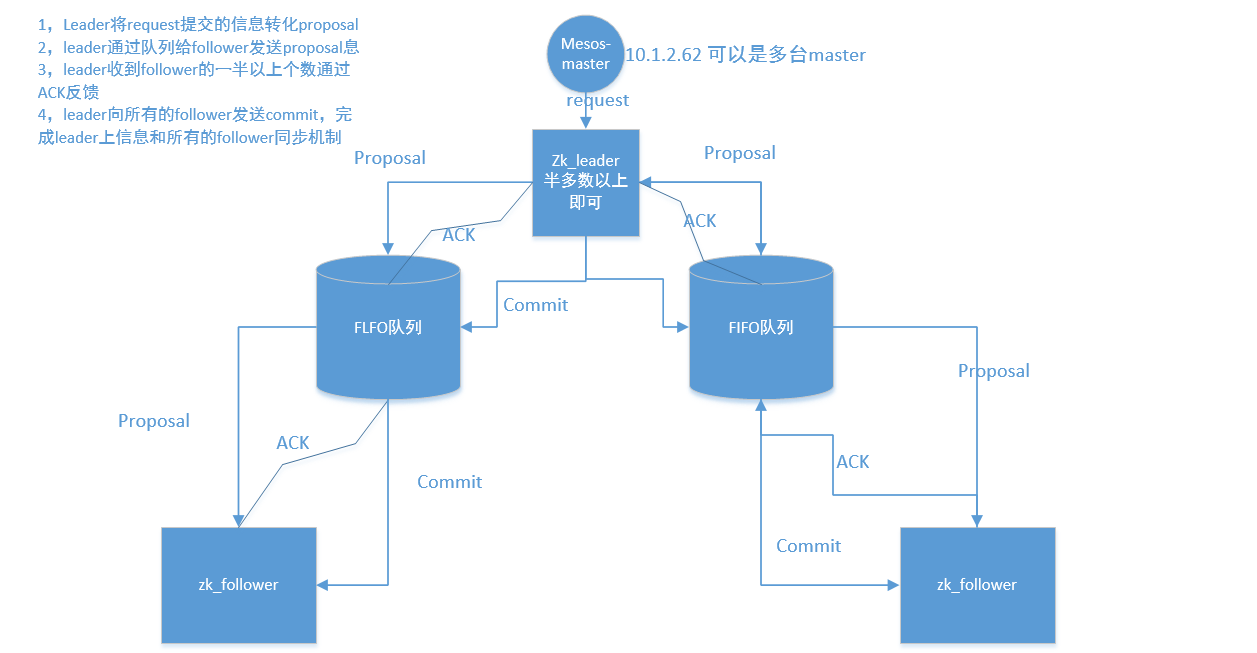


1. 会通过zk=zk://dev\_zookeeper\_srv01:2181,dev\_zookeeper\_srv02:2181,dev\_zookeeper\_srv03:2181/$1\_mesos把master相关的IP和port注册到zookeeper上，通过节点\_mesos信息可以get获取
2. 此时只有一台10.1.2.60这台机器leader负责处理外部客户端mesos-master,mesos-agent的事物请求(或写操作)，然后leader服务器将客户端的写操作数据同步到所有的follower节点,
3. 通过netstat -ano|grep 10.1.2.59(zk\_follower) <------->10.1.2.63(agent)

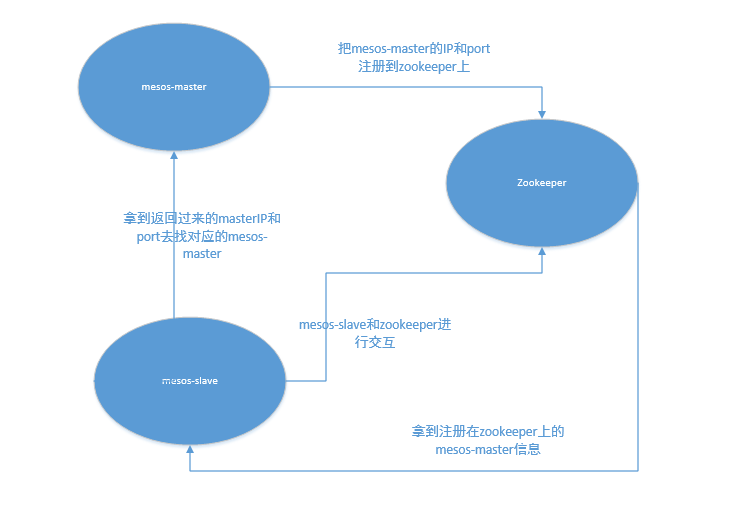
10.1.2.60(zk\_leader)<----->10.1.2.62(master)

10.1.2.61(zk\_follower)<------>10.1.2.62(master)

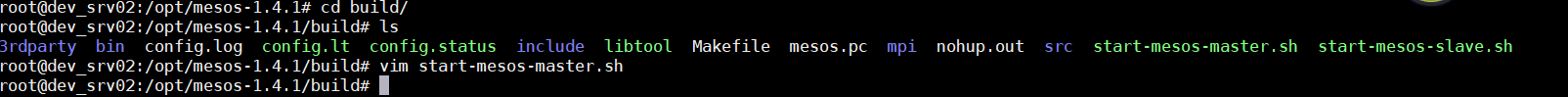
消息广播机制，数据怎么同步？

  
4.1. 客户端mesos-master10.1.2.62发起一个请求request   
4.2. zk\_Leader服务器10.1.2.60将master的request请求转化为事物proposal，同时为每个proposal分配一个全局唯一的ID，即ZXID。   
4.3. zk\_leader服务器10.1.2.62与每个zk\_follower10.1.2.59,10.1.2.61之间都有一个队列，zk\_leader将消息发送到该队列   
4.4. zk\_follower机器从队列中取出消息处理完(写入本地事物日志中)毕后，向zk\_leader服务器发送ACK确认。   
4.5.zk\_ leader服务器收到半数以上的zk\_follower的ACK后，即认为可以发送commit   
4.6. zk\_leader向所有的zk\_follower服务器发送commit消息。

以上流程图体现了几个zookeeper节点信息数据是同步的  
      （1）当Leader出现问题，则会选举出新的Leader服务器。当选举出新的Leader服务器，同时集群中已经有过半的机器与该Leader服务器完成状态同步（数据同步），退出崩溃恢复模式。进入消息广播模式。



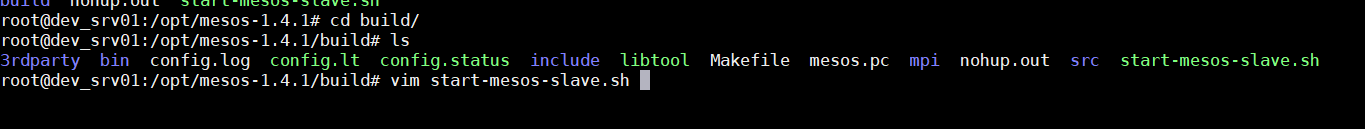
Master



ip=`ip a |grep 'global eth0'|awk '{print $2}'|awk -F '/' '{print $1}'`

/opt/mesos-1.4.1/build/bin/mesos-master.sh --ip=$ip --hostname=$ip --port=5050 --hostname\_lookup=false --work\_dir=/var/lib/mesos/ --log\_dir=/var/log/mesos/ --zk=zk://dev\_zookeeper\_srv01:2181,dev\_zookeeper\_srv02:2181,dev\_zookeeper\_srv03:2181/$1\_mesos\_dev --zk\_session\_timeout=20mins --quorum=1 &

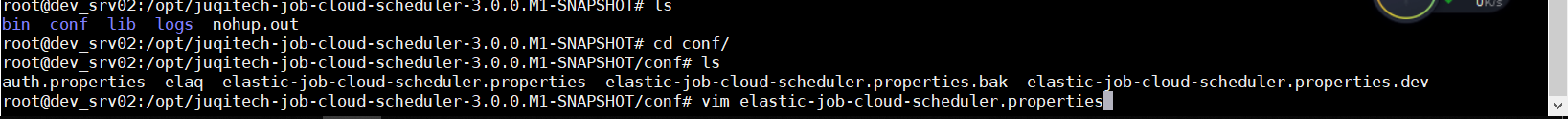
Agent



#!/bin/bash

ip=`ip a |grep 'global eth0'|awk '{print $2}'|awk -F '/' '{print $1}'`

/opt/mesos-1.4.1/build/bin/mesos-slave.sh --work\_dir=/var/lib/mesos\_slave/ --log\_dir=/var/log/mesos --master=zk://dev\_zookeeper\_srv01:2181,dev\_zookeeper\_srv02:2181,dev\_zookeeper\_srv03:2181/$1\_mesos\_dev --ip=$ip --hostname=$ip &



# Routable IP address

hostname=10.1.2.26

# Username for mesos framework

user=

# Mesos zookeeper address

mesos\_url=zk://dev\_zookeeper\_srv01:2181,dev\_zookeeper\_srv02:2181,dev\_zookeeper\_srv03:2181/\_mesos\_dev

# Role for mesos framework

#mesos\_role=

# Elastic-Job-Cloud's zookeeper address

zk\_servers=dev\_zookeeper\_srv01:2181,dev\_zookeeper\_srv02:2181,dev\_zookeeper\_srv03:2181

# Elastic-Job-Cloud's zookeeper namespace

zk\_namespace=elastic-job-cloud-DEV

# Elastic-Job-Cloud's zookeeper digest

zk\_digest=

# Job rest API port

http\_port=8899

# Max size of job accumulated

job\_state\_queue\_size=10000

mater

ect/hosts

127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts

::1 localhost ip6-localhost ip6-loopback

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

10.1.2.26 dev\_srv02

10.1.2.59 dev\_zookeeper\_srv01

10.1.2.60 dev\_zookeeper\_srv02

10.1.2.61 dev\_zookeeper\_srv03

Agent

127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts

::1 localhost ip6-localhost ip6-loopback

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

10.1.2.28 dev\_srv01

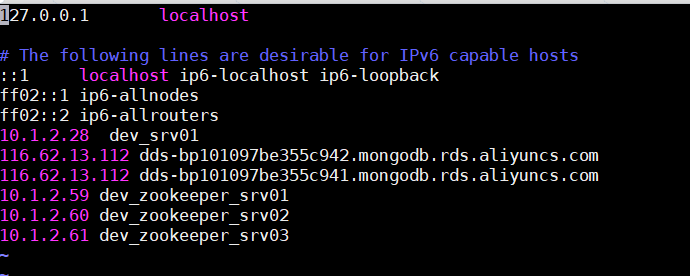
116.62.13.112 dds-bp101097be355c942.mongodb.rds.aliyuncs.com

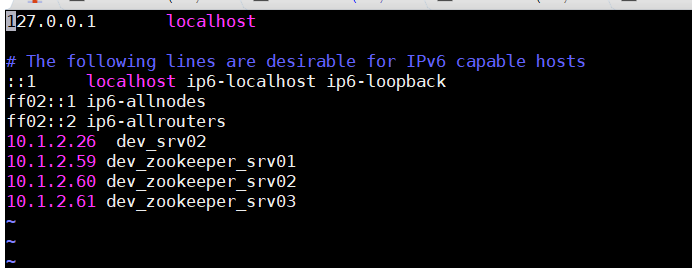
116.62.13.112 dds-bp101097be355c941.mongodb.rds.aliyuncs.com

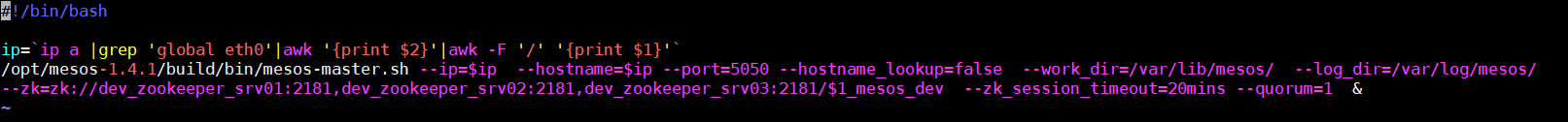
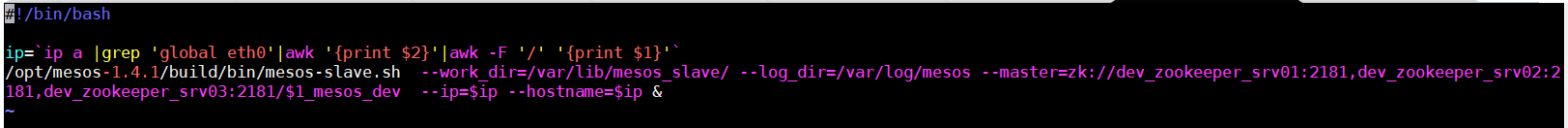
10.1.2.59 dev\_zookeeper\_srv01

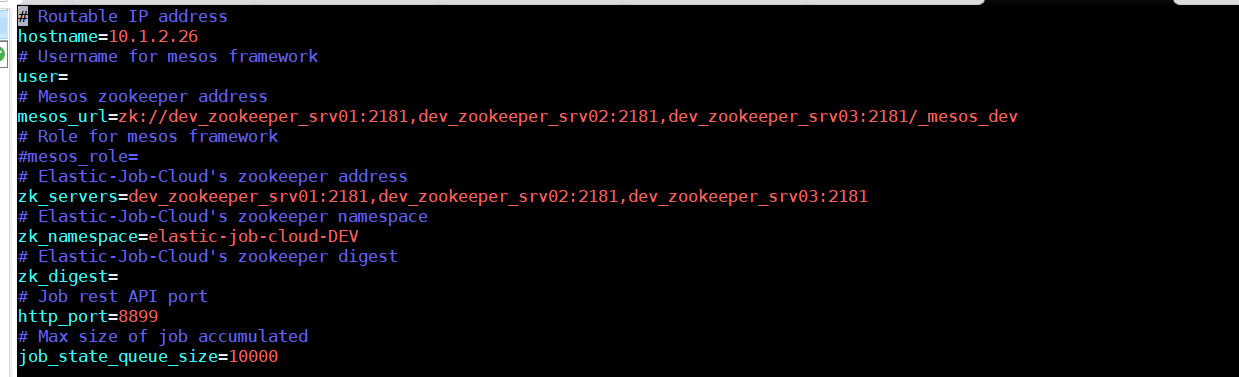
10.1.2.60 dev\_zookeeper\_srv02

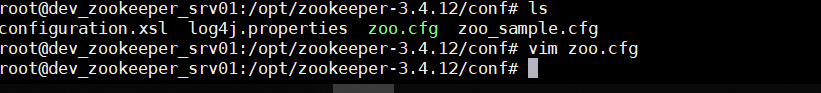
10.1.2.61 dev\_zookeeper\_srv03











tickTime=2000

initLimit=10

syncLimit=5

dataDir=/tmp/zookeeper

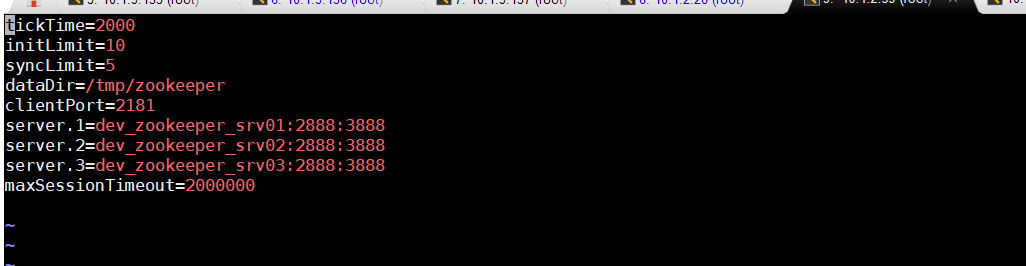
clientPort=2181

server.1=dev\_zookeeper\_srv01:2888:3888

server.2=dev\_zookeeper\_srv02:2888:3888

server.3=dev\_zookeeper\_srv03:2888:3888

maxSessionTimeout=2000000



Zookeeper

# default hosts

127.0.0.1 localhost

127.0.1.1 localhost.localdomain localhost

# The following lines are desirable for IPv6 capable hosts

::1 localhost ip6-localhost ip6-loopback

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

# Custom hosts

10.1.3.217 iZbp1fi3cw7xrdn9302r9yZ

10.1.2.59 dev\_zookeeper\_srv01

10.1.2.60 dev\_zookeeper\_srv02

10.1.2.61 dev\_zookeeper\_srv03

