暗痛



**etcd集群部署与遇到的坑**

在k8s集群中使用了etcd作为数据中心，在实际操作中遇到了一些坑。今天记录一下，为了以后更好操作。

# ETCD参数说明

- —data-dir 指定节点的数据存储目录，这些数据包括节点ID，集群ID，集群初始化配置，Snapshot文件，若未指定—wal-dir，还会存储WAL文件；
- —wal-dir 指定节点的was文件的存储目录，若指定了该参数，wal文件会和其他数据文件分开存储。
- —name 节点名称
- —initial-advertise-peer-urls 告知集群其他节点url.
- — listen-peer-urls 监听URL，用于与其他节点通讯
- — advertise-client-urls 告知客户端url，也就是服务的url
- — initial-cluster-token 集群的ID
- — initial-cluster 集群中所有节点

# 节点迁移

在生产环境中，不可避免遇到机器硬件故障。当遇到硬件故障发生的时候，我们需要快速恢复节点。ETCD集群可以做到在不丢失数据的，并且不改变节点ID的情况下，迁移节点。
具体办法是：

- 1）停止待迁移节点上的etc进程；
- 2）将数据目录打包复制到新的节点；
- 3）更新该节点对应集群中peer url，让其指向新的节点；
- 4）使用相同的配置，在新的节点上启动etcd进程
-
-
-

# etcd配置

## node1

编辑etcd启动脚本/usr/local/etcd/start.sh

```
/usr/local/etcd/etcd -name niub1 -debug \
-initial-advertise-peer-urls http://niub-etcd-1:2380 \
-listen-peer-urls http://niub-etcd-1:2380 \
-listen-client-urls http://niub-etcd-1:2379,http://127.0.0.1:2379 \
-advertise-client-urls http://niub-etcd-1:2379 \
-initial-cluster-token etcd-cluster-1 \
-initial-cluster niub1=http://niub-etcd-1:2380,niub2=http://niub-etcd-2:2380,niub3=http://niub-etcd-3:2380 \
-initial-cluster-state new  >> /niub/etcd_log/etcd.log 2>&1 &
```

## node2

编辑etcd启动脚本/usr/local/etcd/start.sh

```
/usr/local/etcd/etcd -name niub2 -debug \
-initial-advertise-peer-urls http://niub-etcd-2:2380 \
-listen-peer-urls http://niub-etcd-2:2380 \
-listen-client-urls http://niub-etcd-2:2379,http://127.0.0.1:2379 \
-advertise-client-urls http://niub-etcd-2:2379 \
-initial-cluster-token etcd-cluster-1 \
-initial-cluster niub1=http://niub-etcd-1:2380,niub2=http://niub-etcd-2:2380,niub3=http://niub-etcd-3:2380 \
-initial-cluster-state new  >> /niub/etcd_log/etcd.log 2>&1 &
```

## node3

编辑etcd启动脚本/usr/local/etcd/start.sh

```
/usr/local/etcd/etcd -name niub3 -debug \
-initial-advertise-peer-urls http://niub-etcd-3:2380 \
-listen-peer-urls http://niub-etcd-3:2380 \
-listen-client-urls http://niub-etcd-3:2379,http://127.0.0.1:2379 \
-advertise-client-urls http://niub-etcd-3:2379 \
-initial-cluster-token etcd-cluster-1 \
```

2013年4月 (1)
2012年3月 (7)
2012年1月 (10)
2011年12月 (22)

文章分类

k8s源码分析

最新评论

1. Re:hive on spark
这个应该是文件错误，java 报的错误是file找不到。你可以从这个方向看看
--暗痛
2. Re:hive on spark
博主，spark连接hive出现这问题是怎么回事，搞了几天都没解决，博主遇到过没17/09/18 15:57:24 INFO BlockManagerMaster: Registered BlockM.....
--错误犯过一次，就不要在犯第二次
3. Re:Kubernetes集群安全配置案例
还是表示感谢的，省了拼写的时间
--天生我柴
4. Re:Kubernetes集群安全配置案例
配置kubelet进程时，kubeconfig错误拼写client-certificats，改为client-certificate
--天生我柴
5. Re:hive on spark
@dafeng007不可以 是你在编译的时候没有吧 hive编译进入 编译的hive的版本低...
--回眸,境界

阅读排行榜

1. etcd集群部署与遇到的坑(50045)
2. etcd 命令行(18249)
3. hive on spark(14764)
4. Kubernetes集群安全配置案例(13008)
5. flume监控(12474)

评论排行榜

1. hive on spark(8)
2. flume监控(6)
3. Kubernetes集群安全配置案例(3)
4. JavaScript的作用域与闭包(2)
5. Mongodb的安装与启动(1)

推荐排行榜

1. Kubernetes集群安全配置案例(3)
2. flume监控(2)
3. hive on spark(2)
4. Avro总结(RPC/序列化)(1)
5. 解剖Nginx·自动脚本篇（3）源码相关变量脚本 auto/sources(1)

```
-initial-cluster niub1=http://niub-etcd-1:2380,niub2=http://niub-etcd-2:2380,niub3=http://niub-etcd-3:2380 \
-initial-cluster-state new  >> /niub/etcd_log/etcd.log 2>&1 &
```

## 防火墙

在这3台node服务器开放2379、2380端口，命令：

```
iptables -A INPUT -p tcp -m state --state NEW -m tcp --dport 2379 -j ACCEPT
iptables -A INPUT -p tcp -m state --state NEW -m tcp --dport 2380 -j ACCEPT
```

## haproxy配置

haproxy配置过程略 编辑/etc/haproxy/haproxy.cfg文件，增加：

```
frontend etcd
    bind 10.10.0.14:2379
    mode tcp
    option tcplog
    default_backend etcd
    log 127.0.0.1 local3
    backend etcd
    balance roundrobin
    fullconn 1024
    server etcd1 10.10.0.11:2379 check port 2379 inter 300 fall 3
    server etcd2 10.10.0.12:2379 check port 2379 inter 300 fall 3
    server etcd3 10.10.0.13:2379 check port 2379 inter 300 fall 3
```

# 检查etcd服务运行状态

使用curl访问：

```
curl http://10.10.0.14:2379/v2/members
```

返回以下结果为正常（3个节点）：

```
{
  "members": [
    {
      "id": "1f890e0c67371d24",
      "name": "niub1",
      "peerURLs": [
        "http://niub-etcd-1:2380"
      ],
      "clientURLs": [
        "http://niub-etcd-1:2379"
      ]
    },
    {
      "id": "b952ccccefdd8a93",
      "name": "niub3",
      "peerURLs": [
        "http://niub-etcd-3:2380"
      ],
      "clientURLs": [
        "http://niub-etcd-3:2379"
      ]
    },
    {
      "id": "d6dbdb24d5bfc20f",
      "name": "niub2",
      "peerURLs": [
        "http://niub-etcd-2:2380"
      ],
      "clientURLs": [
        "http://niub-etcd-2:2379"
      ]
    }
  ]
}
```

## etcd备份

使用etcd自带命令etcdctl进行etc备份，脚本如下：

```
#!/bin/bash

date_time=`date +%Y%m%d`
etcdctl backup --data-dir /usr/local/etcd/niub3.etcd/ --backup-dir /niub/etcd_backup/${date_time}

find /niub/etcd_backup/ -ctime +7 -exec rm -r {} \;
```

# etcdctl操作

### 更新一个节点

如果你想更新一个节点的 IP(peerURLS)，首先你需要知道那个节点的 ID。你可以列出所有节点，找出对应节点的 ID。

```
$ etcdctl member list
6e3bd23ae5f1eae0: name=node2 peerURLs=http://localhost:23802 clientURLs=http://127.0.0.1:23792
924e2e83e93f2560: name=node3 peerURLs=http://localhost:23803 clientURLs=http://127.0.0.1:23793
a8266ecf031671f3: name=node1 peerURLs=http://localhost:23801 clientURLs=http://127.0.0.1:23791
```

在本例中，我们假设要更新 ID 为 a8266ecf031671f3 的节点的 peerURLs 为：http://10.0.1.10:2380

```
$ etcdctl member update a8266ecf031671f3 http://10.0.1.10:2380
Updated member with ID a8266ecf031671f3 in cluster
```

### 删除一个节点

假设我们要删除 ID 为 a8266ecf031671f3 的节点

```
$ etcdctl member remove a8266ecf031671f3
Removed member a8266ecf031671f3 from cluster
```

执行完后，目标节点会自动停止服务，并且打印一行日志：

```
etcd:  this member has been permanently removed from the cluster. Exiting.
```

如果删除的是 leader 节点，则需要耗费额外的时间重新选举 leader。

### 增加一个新的节点

增加一个新的节点分为两步：

- 通过 etcdctl 或对应的 API 注册新节点
- 使用恰当的参数启动新节点

先看第一步，假设我们要新加的节点取名为 infra3, peerURLs 是 http://10.0.1.13:2380

```
$ etcdctl member add infra3 http://10.0.1.13:2380
added member 9bf1b35fc7761a23 to cluster

ETCD_NAME="infra3"
ETCD_INITIAL_CLUSTER="infra0=http://10.0.1.10:2380,infra1=http://10.0.1.11:2380,infra2=http://10.0.1.12:2380,infra3=http://10.0.1.13:2380"
ETCD_INITIAL_CLUSTER_STATE=existing
```

etcdctl 在注册完新节点后，会返回一段提示，包含3个环境变量。然后在第二部启动新节点的时候，带上这3个环境变量即可。

```
$ export ETCD_NAME="infra3"
$ export
ETCD_INITIAL_CLUSTER="infra0=http://10.0.1.10:2380,infra1=http://10.0.1.11:2380,infra2=http://10.0.1.12:2380,infra3=http://10.0.1.13:2380"
$ export ETCD_INITIAL_CLUSTER_STATE=existing
$ etcd -listen-client-urls http://10.0.1.13:2379 -advertise-client-urls http://10.0.1.13:2379  -listen-peer-urls http://10.0.1.13:2380 -initial-advertise-peer-urls http://10.0.1.13:2380 -data-dir %data_dir%
```

这样，新节点就会运行起来并且加入到已有的集群中了。

值得注意的是，如果原先的集群只有1个节点，在新节点成功启动之前，新集群并不能正确的形成。因为原先的单节点集群无法完成leader的选举。
直到新节点启动完，和原先的节点建立连接以后，新集群才能正确形成。

### 服务故障恢复

在使用etcd集群的过程中，有时会出现少量主机故障，这时我们需要对集群进行维护。然而，在现实情况下，还可能遇到由于严重的设备或网络的故障，导致超过半数的节点无法正常工作。

在etcd集群无法提供正常的服务，我们需要用到一些备份和数据恢复的手段。etcd背后的raft，保证了集群的数据的一致性与稳定性。所以我们对etcd的恢复，更多的是恢复etcd的节点服务，并还原用户数据。

首先，从剩余的正常节点中选择一个正常的成员节点， 使用 etcdctl backup 命令备份etcd数据。

```
$ ./etcdctl backup --data-dir /var/lib/etcd -backup-dir /tmp/etcd_backup
$ tar -zcxf backup.etcd.tar.gz /tmp/etcd_backup
```

这个命令会将节点中的用户数据全部写入到指定的备份目录中，但是节点ID,集群ID等信息将会丢失， 并在恢复到目的节点时被重新。这样主要是防止原先的节点意外重新加入新的节点集群而导致数据混乱。

然后将Etcd数据恢复到新的集群的任意一个节点上， 使用 --force-new-cluster 参数启动Etcd服务。这个参数会重置集群ID和集群的所有成员信息，其中节点的监听地址会被重置为localhost:2379, 表示集群中只有一个节点。

```
$ tar -zxvf backup.etcd.tar.gz -C /var/lib/etcd
$ etcd --data-dir=/var/lib/etcd --force-new-cluster ...
```

启动完成单节点的etcd,可以先对数据的完整性进行验证， 确认无误后再通过Etcd API修改节点的监听地址，让它监听节点的外部IP地址，为增加其他节点做准备。例如：

用etcd命令找到当前节点的ID。

```
$ etcdctl member list

98f0c6bf64240842: name=cd-2 peerURLs=http://127.0.0.1:2580 clientURLs=http://127.0.0.1:2579
```

由于etcdctl不具备修改成员节点参数的功能， 下面的操作要使用API来完成。

```
$ curl http://127.0.0.1:2579/v2/members/98f0c6bf64240842 -XPUT \
 -H "Content-Type:application/json" -d '{"peerURLs":["http://127.0.0.1:2580"]}'
```

注意，在Etcd文档中， 建议首先将集群恢复到一个临时的目录中，从临时目录启动etcd，验证新的数据正确完整后，停止etcd，在将数据恢复到正常的目录中。

最后，在完成第一个成员节点的启动后，可以通过集群扩展的方法使用 etcdctl member add 命令添加其他成员节点进来。

# 扩展etcd集群

在集群中的任何一台etcd节点上执行命令，将新节点注册到集群：

```
curl http://127.0.0.1:2379/v2/members -XPOST -H "Content-Type: application/json" -d '{"peerURLs": ["http:/
```

在新节点上启动etcd容器，注意-initial-cluster-state参数为existing

```
/usr/local/etcd/etcd \
-name etcd03 \
-advertise-client-urls http://192.168.73.150:2379,http://192.168.73.150:4001 \
-listen-client-urls http://0.0.0.0:2379 \
-initial-advertise-peer-urls http://192.168.73.150:2380 \
-listen-peer-urls http://0.0.0.0:2380 \
-initial-cluster-token etcd-cluster \
-initial-cluster "etcd01=http://192.168.73.140:2380,etcd02=http://192.168.73.137:2380,etcd03=http://192.1
-initial-cluster-state existing


```

任意节点执行健康检查：

```
[root@docker01 ~]# etcdctl cluster-health
member 2bd5fcc327f74dd5 is healthy: got healthy result from http://192.168.73.140:2379
member c8a9cac165026b12 is healthy: got healthy result from http://192.168.73.137:2379
cluster is healthy
```

# Etcd数据迁移

## 数据迁移

在 gzns-inf-platform53.gzns.baidu.com 机器上运行着一个 etcd 服务器，其 data-dir 为 /var/lib/etcd/。我们要以 /var/lib/etcd 中的数据为基础，搭建一个包含三个节点的高可用的 etcd 集群，三个节点的主机名分别为：

```
gzns-inf-platform53.gzns.baidu.com gzns-inf-platform56.gzns.baidu.com gzns-inf-platform60.gzns.baidu.com
```

### 初始化一个新的集群

我们先分别在上述三个节点上创建 /home/work/etcd/data-dir/ 文件夹当作 etcd 集群每个节点的数据存放目录。然后以 gzns-inf-platform60.gzns.baidu.com 节点为起点创建一个单节点的 etcd 集群，启动脚本 force-start-etcd.sh 如下：

```
#!/bin/bash

# Don't start it unless etcd cluster has a heavily crash !

../bin/etcd --name etcd2 --data-dir /home/work/etcd/data-dir --advertise-client-urls http://gzns-inf-platform60.gzns.baidu.com:2379,http://gzns-inf-platform60.gzns.baidu.com:4001 --listen-client-urls http://0.0.0.0:2379,http://0.0.0.0:4001 --initial-advertise-peer-urls http://gzns-inf-platform60.gzns.baidu.com:2380 --listen-peer-urls http://0.0.0.0:2380 --initial-cluster-token etcd-cluster-1 --initial-cluster etcd2=http://gzns-inf-platform60.gzns.baidu.com:2380 --force-new-cluster > ./log/etcd.log 2>&1
```

这一步的 **--force-new-cluster** 很重要，可能是为了抹除旧 etcd 的一些属性信息，从而能成功的创建一个单节点 etcd 的集群。

这时候通过

```
etcdctl member list
```

查看 peerURLs 指向的是不是 http://gzns-inf-platform60.gzns.baidu.com:2380 ？如果不是，需要更新这个 etcd 的 peerURLs 的指向，否则这样在加入新的节点时会失败的。

我们手动更新这个 etcd 的 peerURLs 指向

```
etcdctl member update ce2a822cea30bfca http://gzns-inf-platform60.gzns.baidu.com:2380
```

### 添加**etcd1**成员

然后添加 gzns-inf-platform56.gzns.baidu.com 节点上的 etcd1 成员

```
etcdctl member add etcd1 http://gzns-inf-platform56.gzns.baidu.com:2380
```

注意要先添加 etcd1 成员后，再在 gzns-inf-platform56.gzns 机器上启动这个 etcd1 成员

这时候我们登陆上 gzns-inf-platform56.gzns.baidu.com 机器上启动这个 etcd1 实例，启动脚本 force-start-etcd.sh 如下：

```
#!/bin/bash

# Don't start it unless etcd cluster has a heavily crash !

../bin/etcd --name etcd1 --data-dir /home/work/etcd/data-dir --advertise-client-urls http://gzns-inf-platform56.gzns.baidu.com:2379,http://gzns-inf-platform56.gzns.baidu.com:4001 --listen-client-urls http://0.0.0.0:2379,http://0.0.0.0:4001 --initial-advertise-peer-urls http://gzns-inf-platform56.gzns.baidu.com:2380 --listen-peer-urls http://0.0.0.0:2380 --initial-cluster-token etcd-cluster-1 --initial-cluster etcd2=http://gzns-inf-platform60.gzns.baidu.com:2380,etcd1=http://gzns-inf-platform56.gzns.baidu.com:2380 --initial-cluster-state existing > ./log/etcd.log 2>&1
```

注意在这个节点上我们先把 data-dir 文件夹中的数据删除（如果有内容的情况下），然后设置 --initial-cluster和 --initial-cluster-state。

### 添加 **etcd0** 成员

这时候我们可以通过

```
etcdctl member list
```

观察到我们新加入的节点了，然后我们再以类似的步骤添加第三个节点 gzns-inf-platform53.gzns.baidu.com上 的 etcd0 实例

```
etcdctl member add etcd0 http://gzns-inf-platform53.gzns.baidu.com:2380
```

然后登陆到 gzns-inf-platform53.gzns.baidu.com 机器上启动 etcd0 这个实例，启动脚本 force-start-etcd.sh 如下：

```
#!/bin/bash

# Don't start it unless etcd cluster has a heavily crash !

../bin/etcd --name etcd0 --data-dir /home/work/etcd/data-dir --advertise-client-urls http://gzns-inf-platform53.gzns.baidu.com:2379,http://gzns-inf-platform53.gzns.baidu.com:4001 --listen-client-urls http://0.0.0.0:2379,http://0.0.0.0:4001 --initial-advertise-peer-urls http://gzns-inf-platform53.gzns.baidu.com:2380 --listen-peer-urls http://0.0.0.0:2380 --initial-cluster-token etcd-cluster-1 --initial-cluster etcd2=http://gzns-inf-
```

```
platform60.gzns.baidu.com:2380,etcd1=http://gzns-inf-platform56.gzns.baidu.com:2380,etcd0=http://gzns-inf-platform53.gzns.baidu.com:2380 --initial-cluster-state existing > ./log/etcd.log 2>&1
```

过程同加入 etcd1 的过程相似，这样我们就可以把单节点的 etcd 数据迁移到一个包含三个 etcd 实例组成的集群上了。

#### 大体思路

先通过 --force-new-cluster 强行拉起一个 etcd 集群，抹除了原有 data-dir 中原有集群的属性信息（内部猜测），然后通过加入新成员的方式扩展这个集群到指定的数目。

### 高可用**etcd**集群方式（可选择）

上面数据迁移的过程一般是在紧急的状态下才会进行的操作，这时候可能 etcd 已经停掉了，或者节点不可用了。在一般情况下如何搭建一个高可用的 etcd 集群呢，目前采用的方法是用 supervise 来监控每个节点的 etcd 进程。

在数据迁移的过程中，我们已经搭建好了一个包含三个节点的 etcd 集群了，这时候我们对其做一些改变，使用supervise 重新拉起这些进程。

首先登陆到 gzns-inf-platform60.gzns.baidu.com 节点上，kill 掉 etcd 进程，编写 etcd 的启动脚本 start-etcd.sh，其中 start-etcd.sh 的内容如下：

```
#!/bin/bash
../bin/etcd --name etcd2 --data-dir /home/work/etcd/data-dir --advertise-client-urls http://gzns-inf-platform60.gzns.baidu.com:2379,http://gzns-inf-platform60.gzns.baidu.com:4001 --listen-client-urls http://0.0.0.0:2379,http://0.0.0.0:4001 --initial-advertise-peer-urls http://gzns-inf-platform60.gzns.baidu.com:2380 --listen-peer-urls http://0.0.0.0:2380 --initial-cluster-token etcd-cluster-1 --initial-cluster etcd2=http://gzns-inf-platform60.gzns.baidu.com:2380,etcd1=http://gzns-inf-platform56.gzns.baidu.com:2380,etcd0=http://gzns-inf-platform53.gzns.baidu.com:2380 --initial-cluster-state existing > ./log/etcd.log 2>&1
```

然后使用 supervise 执行 start-etcd.sh 这个脚本，使用 supervise 启动 start-etcd.sh 的启动脚本 etcd_control 如下：

```
#!/bin/sh

if [ $# -ne 1 ]; then
    echo "$0: start|stop"
fi


work_path=`dirname $0`
cd ${work_path}
work_path=`pwd`

supervise=${work_path}/supervise/bin/supervise64.etcd
mkdir -p ${work_path}/supervise/status/etcd


case "$1" in
start)
    killall etcd supervise64.etcd
    ${supervise} -f "sh ./start-etcd.sh" \
        -F ${work_path}/supervise/conf/supervise.conf  \
        -p  ${work_path}/supervise/status/etcd
    echo "START etcd daemon ok!"
;;
stop)
    killall etcd supervise64.etcd
    if [ $? -ne 0 ]
    then
        echo "STOP etcd daemon failed!"
        exit 1
    fi
    echo "STOP etcd daemon ok!"
```

这里为什么不直接用 supervise 执行 etcd 这个命令呢，反而以一个 start-etcd.sh 脚本的形式启动这个 etcd 呢？原因在于我们需要将 etcd 的输出信息重定向到文件中，

如果直接在 supervise 的 command 进行重定向，将发生错误。

分别登陆到以下两台机器

- gzns-inf-platform56.gzns.baidu.com
- gzns-inf-platform53.gzns.baidu.com

上进行同样的操作，注意要针对每个节点的不同修改对应的etcd name 和 peerURLs 等。

## 常见问题

**1、etcd** 读取已有的 **data-dir** 数据而启动失败，常常表现为**cluster id not match**什么的

可能原因是新启动的 etcd 属性与之前的不同，可以尝 --force-new-cluster 选项的形式启动一个新的集群

**2、etcd** 集群搭建完成后，通过 **kubectl get pods** 等一些操作发生错误的情况

目前解决办法是重启一下 apiserver 进程

**3、还是 etcd**启动失败的错误，大多数情况下都是与**data-dir** 有关系，**data-dir** 中记录的信息与 **etcd**启动的选项所标识的信息不太匹配造成的

如果能通过修改启动参数解决这类错误就最好不过的了，非常情况下的解决办法：

- 一种解决办法是删除data-dir文件
- 一种方法是复制其他节点的data-dir中的内容，以此为基础上以 --force-new-cluster 的形式强行拉起一个，然后以添加新成员的方式恢复这个集群，这是目前的几种解决办法

分类: 分布式系统



暗痛
关注 - 0
粉丝 - 19
+加关注

1 0

« 上一篇：flume监控
» 下一篇：etcd 命令行

posted @ 2016-08-02 10:11 暗痛 阅读(50046) 评论(0) 编辑 收藏



相关博文：
· etcd集群部署与遇到的坑(转)
· 转发:etcd集群部署与遇到的坑
· Kubernetes容器集群部署Etcd(三)
· etcd3.0集群安装
· etcd集群部署