

# 统一实施教程

杭州麟云科技有限公司

2018 年 09 月 26 日

## 目录

第 1 章 统一架构.....	7
1.1 产品组服务器配置.....	7
1.2 政务云服务器配置.....	7
1.3 阿里云服务器配置.....	8
1.4 私有云服务器配置.....	8
1.5 部署脚本.....	9
第 2 章 系统初始化.....	10
2.1 确认服务器信息.....	10
2.1.1 验证账户和权限.....	10
2.1.2 查看系统版本.....	10
2.1.3 检查逻辑 cpu 个数.....	10
2.1.4 检查内存和交换分区.....	10
2.1.5 磁盘分区并挂载.....	11
2.1.6 检查系统防火墙.....	11
2.1.7 检查网络和 DNS.....	12
2.1.8 配置 yum 源.....	13
2.1.9 时间同步服务.....	15
2.2 执行系统初始化脚本.....	15
2.2.1 系统初始化脚本简介.....	15
2.2.2 执行系统初始化脚本.....	16
2.2.3 手动配置内网 yum 源.....	16
第 3 章 部署 Nginx.....	17
3.1 Nginx 反向代理.....	17
3.2 注意事项.....	17
3.3 Nginx 配置文件.....	17
3.4 Nginx 服务管理.....	19
3.5 防火墙.....	19
3.6 Nginx 反向代理配置实例.....	19
第 4 章 部署 tomcat.....	20
4.1 环境要求.....	20
4.2 服务管理和开机启动.....	20
4.2.1 Centos6.....	20

4.2.2 Centos7.....	21
4.3 用户身份.....	21
4.4 修改文件需谨慎.....	21
4.5 防火墙.....	22
4.6 JVM 堆大小.....	22
4.7 安装步骤.....	22
4.7.1 安装界面概述.....	22
4.7.2 添加 <code>sudo</code> 用户.....	23
4.7.3 安装 <code>tomcat</code> .....	24
4.7.4 可选择安装: <code>catalina-jmx-remote.jar</code> (监控 <code>tomcat</code> 实例) .....	25
4.7.5 删除 <code>tomcat</code> 实例.....	27
4.7.6 其他选项.....	28
4.7.7 自定义 <code>tomcat</code> 版本.....	28
第 5 章 部署 <code>jar</code> .....	29
5.1 环境要求.....	29
5.2 脚本功能.....	29
5.3 服务管理和开机启动.....	29
5.4 用户身份.....	29
5.5 防火墙.....	29
5.6 安装步骤.....	30
5.6.1 安装界面概述.....	30
5.6.2 部署新项目.....	31
5.6.3 已存在项目注册服务.....	32
5.6.4 重新注册服务.....	33
第 6 章 部署 <code>Oracle</code> .....	33
6.1 运行环境.....	33
6.2 部署 <code>Oracle</code> 过程.....	34
6.2.1 数据库版本.....	34
6.2.2 数据盘挂载.....	34
6.2.3 参数修改.....	34
6.2.4 安装前操作.....	36
6.2.5 安装 <code>Oracle</code> .....	38
6.2.6 设置开机启动.....	44
6.2.7 初始化.....	47

6.2.8 日志清理脚本.....	48
6.3 防火墙.....	49
第 7 章 数据备份方案.....	50
7.1 MySQL.....	50
7.2 Oracle.....	51
第 8 章 部署 Hbase.....	52
8.1 运行环境.....	52
8.2 准备工作.....	53
8.3 部署 zookeeper 过程.....	55
8.4 部署 hdfs 过程.....	58
8.5 部署 hbase 过程.....	69
第 9 章 部署 xxl-job-admin.....	75
9.1 运行环境.....	75
9.2 准备工作.....	75
9.3 安装 xxl-job-admin.....	75
9.4 安装 xxl-job-executor（可选）.....	77
9.5 防火墙.....	78
第 10 章 部署 kettle 集群.....	78
10.1 运行环境.....	78
10.2 准备工作.....	79
10.3 部署 master 节点.....	80
10.4 部署 slave 节点.....	82
10.5 配置资源库并创建 carte 集群规范.....	86
10.6 创建 job 规范.....	93
10.7 配置 job 日志规范.....	93
10.8 配置 xxl-job 定期执行 job 实例.....	94
10.9 防火墙.....	96
第 11 章 附录.....	96
11.1 防火墙配置实例.....	96
11.2 创建 swap 分区并挂载.....	97
11.2.1 使用磁盘.....	97
11.2.2 使用文件.....	99
11.3 磁盘分区并挂载.....	99
11.4 搭建基础源服务器.....	106

11.5 部署目录规范.....	106
11.6 文件上传下载规范.....	107
11.6.1 上传.....	107
11.6.2 下载.....	107
11.7 服务管理规范.....	107

## 版本修订

日期	版本	作者	行为	说明
2018-09-26	1.0	王涛	创建	新建此文档
2019-05-31	1.1	王涛	增加	完善磁盘分区和 nginx 操作
2019-06-06	1.2	王涛	修改	tomcat 部署脚本迭代
2019-06-11	1.3	王涛	修改	调整顺序，完善实施内容
2019-06-14	1.4	王涛	修改	根据脚本修改 yum 源配置方式
2019-06-21	1.5	王涛	修改	根据脚本修改磁盘分区挂载方式
2019-07-02	1.6	王涛	增加	增加创建并挂载 swap 分区内容
2020-04-21	1.7	王涛	增加	增加 oracle，数据备份和 hbase
2020-04-29	1.8	王涛	增加	增加 xxl-job，kettle 部署教程

## 第 1 章 统一架构

在实施部署规范的基础上，形成一系列自动化部署脚本。统一公司的所有产品的系统运行环境与项目部署路径等。

### 1.1 产品组服务器配置

产品组	系统	CPU和内存最低配置	数据磁盘	数量
基础服务(sso、orm)	CentOS 7.7	4核8G	60G	1台
数据中台	CentOS 7.7	8核32G	100G	1台
综合执法 应用端	CentOS 7.7	8核32G	100G	1台
综合执法 分析端	CentOS 7.7	8核32G	100G	1台
运行监测 应用端	CentOS 7.7	8核32G	100G	1台
运行监测 分析端	CentOS 7.7	16核32G	100G	1台
品质出行 公网10Mbps	CentOS 7.7	4核16G	200G	3台
安全风控	CentOS 7.7	8核32G	100G	1台

### 1.2 政务云服务器配置

备注：n 大于等于 0.5。根据项目实际情况估算

应用和版本	系统	CPU和内存最低配置	数据磁盘	数量
ORACLE 11.2.0.4	CentOS 7.7	8核16G	nT（推荐SSD）	1台
ORACLE备份服务器	CentOS 7.7	2核4G	nT	1台
MySQL 5.7高可用	云产品	8核16G	nT（推荐SSD）	1台
Hbase 2 master	云产品	4核8G	100G*2	1台
Hbase 2 slave	云产品	4核16G	nT*6	3台
OSS 文件存储	云产品	标准型(ZRS) 存储包	nT	1台

Redis 5	云产品	2G主从版		1台
RocketMQ 4.6	CentOS 7.7	4核8G	50G	1台
Nacos 1.2 Xxl-job-admin 2.2	CentOS7.7	4核8G	100G	1台
Nginx 1.16	CentOS 7.7	2核2G	50G	1台
kettle 9 master	Windows Server 2008R2	8核16G	100G	1台
kettle 9 slave	CentOS 7.7	8核16G	数据盘100G	2台

## 1.3 阿里云服务器配置

备注：n 大于等于 0.5。根据项目实际情况估算

应用和版本	系统	CPU和内存最低配置	数据磁盘	数量
ORACLE 11.2.0.4	CentOS 7.7	8核16G	nT（推荐SSD）	1台
MySQL 5.7高可用	云产品	8核16G	nT（推荐SSD）	1台
ORACLE备份服务器	CentOS 7.7	2核4G	nT	1台
Hbase 2 master	云产品	4核8G	100G*2	1台
Hbase 2 slave	云产品	4核16G	nT*6	3台
OSS 文件存储	云产品	标准型 (ZRS) 存储包	nT	1台
Redis 5	云产品	2G主从版		1台
RocketMQ 4.6	CentOS 7.7	4核8G	50G	1台
Nacos 1.2 Xxl-job-admin 2.2	CentOS7.7	4核8G	100G	1台
Nginx 1.16	CentOS 7.7	2核2G	50G	1台
kettle 9 master	Windows Server 2008R2	8核16G	100G	1台
kettle 9 slave	CentOS 7.7	8核16G	数据盘100G	2台

## 1.4 私有云服务器配置

备注：n 大于等于 0.5。根据项目实际情况估算

应用和版本	系统	CPU和内存最低配置	数据磁盘	数量
-------	----	------------	------	----



ORACLE 11.2.0.4	CentOS 7.7	8核16G	nT（推荐SSD）	1台
MySQL 5.7	CentOS 7.7	8核16G	nT*2（推荐SSD）+ nT	1台
数据库备份服务器	CentOS 7.7	2核4G	nT*2（mysql + oracle）	1台
Hbase 2 master	CentOS 7.7	4核8G	100G*2	1台
Hbase 2 slave	CentOS 7.7	4核16G	nT*6	3台
SeaweedFS文件存储	CentOS 7.7	4核4G	nT	1台
RocketMQ 4.6	CentOS 7.7	4核8G	50G	1台
Nacos 1.2 Xxl-job-admin 2.2	CentOS 7.7	4核8G	100G	1台
Nginx 1.16	CentOS 7.7	2核2G	50G	1台
kettle 9 master	Windows Server 2008R2	8核16G	100G	1台
kettle 9 slave	CentOS 7.7	8核16G	数据盘100G	1台

## 1.5 部署脚本

类型	脚本名称	脚本路径
源服务器搭建	Yum源	<a href="http://192.168.90.117/yum.tar.gz">http://192.168.90.117/yum.tar.gz</a>
	时间服务器	<a href="http://192.168.90.117/time-sync.tar.gz">http://192.168.90.117/time-sync.tar.gz</a>
系统初始化	磁盘分区挂载	<a href="http://192.168.90.117/diskpart.tar.gz">http://192.168.90.117/diskpart.tar.gz</a>
	系统初始化	<a href="http://192.168.90.117/system.tar.gz">http://192.168.90.117/system.tar.gz</a>
中间件部署	Nginx1.16-centos	<a href="http://192.168.90.117/nginx.tar.gz">http://192.168.90.117/nginx.tar.gz</a>
	Nginx1.16-windows	<a href="http://192.168.90.117/nginx_windows.tar.gz">http://192.168.90.117/nginx_windows.tar.gz</a>
	Tomcat8.5-centos	<a href="http://192.168.90.117/tomcat.tar.gz">http://192.168.90.117/tomcat.tar.gz</a>
	Tomcat8.5-windows	<a href="http://192.168.90.117/tomcat_windows.tar.gz">http://192.168.90.117/tomcat_windows.tar.gz</a>
	RocketMQ4.6	<a href="http://192.168.90.117/rocketmq.tar.gz">http://192.168.90.117/rocketmq.tar.gz</a>
数据库部署	MySQL5.7	<a href="http://192.168.90.117/mysql.tar.gz">http://192.168.90.117/mysql.tar.gz</a>
	Oracle11.2.0.4	<a href="http://192.168.90.117/centos_install_oracle.pdf">http://192.168.90.117/centos_install_oracle.pdf</a>
	Redis5	<a href="http://192.168.90.117/redis.tar.gz">http://192.168.90.117/redis.tar.gz</a>
产品部署	War	脚本部署tomcat之后，根据产品规范，把war包放到对应的tomcat实例中
	Jar-centos	<a href="http://192.168.90.117/jar.tar.gz">http://192.168.90.117/jar.tar.gz</a>
	Jar-windows	<a href="http://192.168.90.117/jar_windows.tar.gz">http://192.168.90.117/jar_windows.tar.gz</a>
第三方软件部署	Fastdfs5.11	<a href="http://192.168.90.117/fastdfs.tar.gz">http://192.168.90.117/fastdfs.tar.gz</a>
	SeaweedFs1.7	<a href="http://192.168.90.117/seaweedfs.tar.gz">http://192.168.90.117/seaweedfs.tar.gz</a>
	Nacos-1.2	<a href="http://192.168.90.117/nacos.tar.gz">http://192.168.90.117/nacos.tar.gz</a>
	Xxl-job-admin 2.2	<a href="http://192.168.90.117/software/xxl-job/xxl-job-">http://192.168.90.117/software/xxl-job/xxl-job-</a>

工具类脚本		admin-2.2.0.tar.gz
	Xxl-job-executor 2.2	http://192.168.90.117/software/xxl-job/xxl-job-executor-2.2.0.tar.gz
	清理文件-centos	http://192.168.90.117/rotateLogs-centos.tar.gz
	清理文件-window	http://192.168.90.117/rotateLogs-windows.zip

## 第 2 章 系统初始化

CentOS 服务器在执行系统初始化之前必须先按照 2.1 的步骤确认服务器信息

### 2.1 确认服务器信息

#### 2.1.1 验证账户和权限

检查客户提供的服务器清单，并登录验证用户名密码是否正确，不正确的信息及时沟通更正，不要把错误信息发布给开发人员。

执行初始化脚本需要 root 或 sudo 用户。注意 sudo 用户需要配置所有权限，如果客户环境不允许开放 sudo 用户的所有权限，需要项目经理和客户协商。推荐的解决方案：先临时开启 sudo 用户的所有权限或者给 root 用户的密码，等初始化完成后再收回 sudo 用户的权限或修改 root 密码。

#### 2.1.2 查看系统版本

目前使用的 linux 服务器，主要是 centos7 系统，其次是 centos6 系统。

```
cat /etc/centos-release
```

#### 2.1.3 检查逻辑 cpu 个数

```
lscpu | grep "CPU(s)" | sed -n '1p'
```

#### 2.1.4 检查内存和交换分区

free -m    ##单位为 M，注意查看总大小和剩余大小。

或者

free -g    ##单位为 G，注意查看总大小和剩余大小。

**注意：**如果部署 oracle 数据库，还要确保有 swap 分区，且空间大于 128M。通常云主机都没有 swap 分区，此时要手动创建 swap 分区，参考[附录](#)

```
[root@localhost ~]# free -m
```

	total	used	free	shared	buff/cache	available
Mem:	7822	134	7332	8	355	7430
Swap:	2047	0	2047			

## 2.1.5 磁盘分区并挂载

1) 查看当前服务器的磁盘：sudo lsblk

**注意：**有些新添加的磁盘，需要重启服务器才会显示

磁盘名称通常是：sd\*（物理磁盘），vd\*（虚拟磁盘），hd\*三种格式。如下：

sdb 和 vdb 就是新磁盘。完整路径（脚本要用到）是/dev/sdb，/dev/vdb

```
[deploy@localhost ~]$ lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	20G	0	disk	
└─sda1	8:1	0	500M	0	part	/boot
└─sda2	8:2	0	19.5G	0	part	
└─┬─cl-root	253:0	0	17.5G	0	lvm	/
└─┬─cl-swap	253:1	0	2G	0	lvm	[SWAP]
sdb	8:16	0	20G	0	disk	

```
[deploy@iZwz93g36wzfbgu4n8yc0mZ ~]$ lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
vda	253:0	0	40G	0	disk	
└─vda1	253:1	0	40G	0	part	/
vdb	253:16	0	200G	0	disk	

2) 下载磁盘分区脚本（参考第一章），对数据磁盘进行分区，并挂载到/linkcld 目录。

如果是 oracle 服务器，请挂载数据磁盘到/home 目录

3) 如果脚本挂载出现问题，请参考[附录](#)

## 2.1.6 检查系统防火墙

### 1、centos7 系统

执行命令：sudo firewall-cmd --list-all

结果分析：

1) FirewallD is not running # 说明防火墙已经关闭，没有启用

2) 结果如下图所示则说明防火墙没有配置规则，需要向客户确认是否需要启用系统防火墙，根据客户反馈结果，在执行系统初始化脚本时选择开启或关闭防火墙

```
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: dhcpv6-client ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

## 2、centos6 系统

执行命令：sudo iptables -nL

结果分析：

1) iptables: Firewall is not running. # 说明防火墙已经关闭，没有启用

2) 结果如下图所示则说明防火墙没有配置规则，需要向客户确认是否需要启用系统防火墙，根据客户反馈结果，在执行系统初始化脚本时选择开启或关闭防火墙

```
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

### 2.1.7 检查网络和 DNS

生产环境中，无论是 yum 源还是时间同步服务，都需要用到外网。因此需要自己检查并和客户确认服务器的网络状态。

1) 检查外网访问：首先根据 2.1.6 查看系统防火墙是否开启，如果已经配置了防火墙规则，而且此时 `ping 114.114.114.114` 不通，可以先配置系统防火墙把 `114.114.114.114` 加入白名单，配置方法参考附录。如果能 `ping` 通 `114.114.114.114`，则说明可以访问外网，否则说明无法访问外网。

2) 根据第一步的结果，如果可以访问外网，如果开启了防火墙则先把 `122.224.168.146` 加入白名单。然后 `curl demo.linkld.com:666`，如果返回的结果是 `Nginx` 页面，表示服务器配置了 `DNS`；如果返回的结果是 `Could not resolve host`，说明没有配置 `DNS`。

3) 根据以上两步的结果：

3.1) 如果无法访问外网，那就意味着无法使用 `yum` 源和时间同步服务或者使用内网的 `yum` 源代理和时间同步服务器。

3.2) 如果可以访问外网且配置好了 `DNS`，则不需要手动操作，使用系统初始化脚本即可。

3.3) 如果可以访问外网，但是因为防火墙等因素导致无法配置 `DNS`，此时只能依靠自己手动配置域名和 `IP`，操作如下：

首先找一台能够访问外网且可以正常访问 `baidu` 的电脑或服务器。执行以下操作：

3.3.1) `ping ntp1.aliyun.com` #记录 `IP`

3.3.2) `ping time1.aliyun.com` #记录 `IP`

3.3.3) `ping mirrors.aliyun.com` #记录 `IP`

3.3.4) 如果系统防火墙开启了，则把 `ping` 获得的三个 `IP` 都加入白名单。否则跳过这一步，直接看下一步。

3.3.5) `sudo vim /etc/hosts` #编辑此文件，把 3.3.1-3.3.3 中的 `IP` 和域名解析添加进去，比如：

`120.25.115.20 ntp1.aliyun.com`

`203.107.6.88 time1.aliyun.com`

`122.228.4.225 mirrors.aliyun.com`

## 2.1.8 配置 `yum` 源

**注意：**如果服务器没有 `yum` 源，则无法正常执行系统初始化脚本，而且后期很多应用

难以部署，甚至无法正常部署。既然 yum 源这么重要，那么如何配置呢？

- 1) 如果该服务器可以访问外网，则不需要手动配置 yum 源。
- 2) 如果该服务器无法访问外网，则必须手动配置 yum 源。常见的情况分为 3 种：

2.1) 有内网 yum 源：参考 2.3.8.2) 中的操作

2.2) 没有内网 yum 源，但是有前置机可以访问外网。此时可以在前置机部署 Nginx，为内网服务器代理 yum 源。操作如下：

2.2.1) 前置机部署 Nginx (centos 系统可以先初始化，然后部署 Nginx，参考 [Nginx 部署](#))，Nginx 配置如下 (监听端口 9999 可以自定义，但是服务器配置要一致)：

```
server {  
  
    listen 9999 default_server;  
  
    server_name _;  
  
    location ~* /(centos|epel) {  
        proxy_read_timeout 300;  
        proxy_connect_timeout 300;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_pass http://mirrors.aliyun.com;  
    }  
}
```

2.2.2) 配置其他服务器使用代理的 yum 源：其他服务器在执行系统初始化脚本，配置局域网 yum 源时，类型输入“nginx”，然后输入前置机部署 Nginx 的“ip:端口号”，端口号是 80 时，只需要输入 ip 即可。

2.3) 没有内网 yum 源，前置机也不能访问外网时，需要手动下载 iso 镜像文件，然后上传到该服务器中：

2.3.1) 下载 iso 文件：请选择对应的系统版本：

备注：以下 centos6 和 centos7 镜像文件，公司 ftp 上有：<ftp://192.168.90.117>，然后打开：架构/软件部署/系统镜像

#### 2.3.1.1) centos7 镜像公网地址:

[http://mirrors.aliyun.com/centos/7/isos/x86\\_64/CentOS-7-x86\\_64-Everything-1810.iso](http://mirrors.aliyun.com/centos/7/isos/x86_64/CentOS-7-x86_64-Everything-1810.iso)

#### 2.3.1.2) centos6 镜像公网地址:

[http://mirrors.aliyun.com/centos/6.10/isos/x86\\_64/CentOS-6.10-x86\\_64-bin-DVD1.iso](http://mirrors.aliyun.com/centos/6.10/isos/x86_64/CentOS-6.10-x86_64-bin-DVD1.iso)

2.3.2) 把下载的 iso 镜像文件，上传到服务器。执行部署 YUM 源服务器脚本  
脚本下载参考第一章

## 2.1.9 时间同步服务

强烈建议服务器都开启时间同步服务，如果是搭建分布式集群，比如 Hbase 集群，一定要开启时间同步服务。

现在客户提供的服务器通常配置了软硬件防火墙，而且服务器无法访问外网，此时需要的解决方案如下：

**防火墙问题：**如果需要配置硬件防火墙，请联系客户配置；如果只需要配置服务器系统防火墙，可以直接配置 IP 白名单，请参考附录。

1) 如果客户提供的环境有时间同步服务器(在沟通服务器环境时必须和客户确认)，则在执行系统初始化时直接使用。

2) 如果客户提供的环境没有时间同步服务器，此时需要一台能访问外网的服务器，使用时间服务器部署脚本部署成时间同步服务器。然后环境中的其他内网服务器在执行系统初始化时都和这台时间同步服务器同步时间。

## 2.2 执行系统初始化脚本

CentOS 服务器在执行系统初始化之前必须先按照 2.1 的步骤确认服务器信息

### 2.2.1 系统初始化脚本简介

针对全新的 centos6 或 7 服务器，初始化系统环境，主要包括以下方面：

1) 优化 sshd 配置



- 2) 判断是否能访问外网, 根据情况增加 DNS: 1.2.4.8
- 3) 添加 deploy 为 sudo 用户并优化权限等
- 4) 创建/linkld 以及各项子目录并更改属主和属组
- 5) 配置默认时区为亚洲/上海, 并同步到硬件时区
- 6) 交互式修改 hostname, 并解析 ip 和主机名
- 7) 安装 jdk-8u251 的 rpm 包, 并配置系统 JAVA\_HOME 和 CLASSPATH 系统环境变量
- 8) 交互式配置 firewall 是否关闭
- 9) 交互式配置 yum 源
- 10) 安装一些基础软件包: vim, rsync, wget, telnet, iftop, traceroute, lsof, gdisk, unzip, rz, ntpdate 等
- 11) 优化系统参数: tcp 连接, swap 策略, 文件描述符等
- 12) 交互式配置时间同步服务

## 2.2.2 执行系统初始化脚本

脚本下载参考第一章

```
[root@fw116 system_init]# [sh main.sh] ▶ 执行脚本
Installing software/jdk-8u181-linux-x64.rpm.....
Success: install java version=1.8.0_181
Please Wait.....Checking Yum Repository
Please Wait.....install softwares
Input NTP_SERVER ip or domain name <example: 1.1.1.1;2.2.2.2> <e|E(pass)>: ntp1.aliyun.com;time1.aliyun.com
Success: valid ip or domain name is:
ntp1.aliyun.com
time1.aliyun.com
sure or input again ? Input (y|Y = sure)|(n|N = input again): y
Success: NTP configure successfully
```

安装jdk8  
如果系统已经安装jdk8的rpm包, 则会跳过安装  
如果部署脚本中的software目录中没有jdk包, 则会跳过安装  
如果服务器能访问公网, 则这两步不需要配置, 会自动完成  
如果服务器不能连接外网, 则需要手动配置ip (内网yum源服务器的ip)  
如果既不能访问公网, 内网也没有可用的yum源服务器, 则可以跳过这一步

对输入的ip或域名进行检测, 并把可用的ip和域名列出  
然后下一步需要输入y或n选择

手动输入ntp服务器的ip, 多个ip或域名用: 分隔  
如果可访问公网, 可以输入图中的两个地址  
如果不能访问公网, 可以输入局域网ntp服务器的ip  
ntp服务器由项目部署区域自行管理维护

ntp地址信息确认  
y或Y是确定  
n或N是重新输入地址

## 2.2.3 手动配置内网 yum 源

```
Please Wait.....Checking Yum Repository
configure yum repository ? Input <y|Y|n|N>: y
nginx proxy_pass yum or iso yum ? input --> nginx|iso or e|E = pass): iso
Please input yum repository server ip_address=192.168.90.188
yum repository server ip_address=192.168.90.188
sure ? Input <y|Y|n|N e|E(return the upper level)>: y
Please Wait.....Checking Yum Repository
Please Wait.....install softwares
```

nginx表示nginx代理到公网  
iso表示下载iso镜像部署的yum

nginx代理yum源服务器的ip, 非80端口时, 要写ip:端口号

y=配置yum源, n=跳过  
y=输入无误  
n=重新选择

表示yum源可用



## 第 3 章 部署 Nginx

脚本下载参考第一章

### 3.1 Nginx 反向代理

在生产环境中，Nginx 通常会被配置为既是静态 Web 服务器也是反向代理服务器，反向代理功能可以实现 web 服务代理和端口代理。Nginx 反向代理功能和场景如下：

#### 1) 反向代理 web 服务

1.1) 功能：减少端口映射；打通两个网络

1.2) 实例：代理 Tomcat、jar 和 Nginx 等 web 服务；内网服务器借助前置机访问 yum 源；

#### 2) 反向代理端口

1.1) 功能：打通两个网络

### 3.2 注意事项

注意事项：

- 1) 如果服务器不能访问公网，则需要交互式输入局域网 yum 源服务器的 ip，请参考“[第 2 章 2.2 节](#)”
- 2) 如果服务器既不能访问公网，也没有局域网 yum 源。则安装的 nginx 不是完整版，只有基础模块功能。

### 3.3 Nginx 配置文件

#### 1) 目录结构：重点关注红色字体内容

/etc/nginx/

├── fastcgi\_params

├── koi-utf

├── koi-win

├── mime.types

└── modules -> ../../usr/lib64/nginx/modules



- 2) 全局配置文件：通常不需要更改

/etc/nginx/nginx.conf

- 3) 配置模板目录：/etc/nginx/sites-available

该目录的配置文件并不生效

- 4) 启用的配置目录：/etc/nginx/sites-enabled

nginx 当前使用的配置文件都在该目录中

- 5) 实例：使用 sites-available 和 sites-enabled 新增配置

以下操作内容：nginx 新增加一个 81 端口的 server

- 5.1) sudo vim /etc/nginx/sites-available/81.cfg

```

server {

    listen 81

    server_name _;

    location / {

        root /usr/share/nginx/html;

        index index.html index.htm;

    }

}

```

5.2) `sudo ln -sf /etc/nginx/sites-available/81.cfg /etc/nginx/sites-enabled/`

5.3) `sudo nginx -t`

如果配置文件检查有问题，则根据提示修改配置文件。确保配置文件检查无误后再重启 **nginx** 服务

5.4) `sudo systemctl restart nginx`

## 3.4 Nginx 服务管理

1) 每次更改配置文件以后，都要检测配置是否正确：

`sudo nginx -t`

2) centos7:

`sudo systemctl start nginx`

`sudo systemctl stop nginx`

`sudo systemctl restart nginx`

3) centos6:

`sudo service nginx start`

`sudo service nginx stop`

`sudo service nginx restart`

## 3.5 防火墙

参考附录

## 3.6 Nginx 反向代理配置实例

```
server {  
    listen 80;  
    server_name _;  
  
    location ~^/tse {  
        proxy_read_timeout 300;  
    }  
}
```

```
proxy_connect_timeout 300;

proxy_redirect      off;

proxy_set_header Host          $http_host;
proxy_set_header X-Real-IP     $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header X-Frame-Options  SAMEORIGIN;

proxy_pass http://192.168.90.167:9000;

}

}
```

## 第 4 章 部署 tomcat

脚本下载参考第一章

### 4.1 环境要求

- 1) 确保服务器的操作系统是 CentOS 6 或 7，查看系统版本的方式：

```
cat /etc/centos-release
```

- 2) 确保系统剩余内存至少 1G 以上，因为每个 tomcat 实例的 jdk 运行内存是 1G+

### 4.2 服务管理和开机启动

#### 4.2.1 Centos6

**备注：**centos6 中使用 `etc/init.d/tomcat-产品名-端口号` 文件启动 tomcat 服务使用的是 tomcat 用户

- 1) Tomcat 服务的启动文件是 `/etc/init.d/tomcat-产品名-端口号`
- 2) 启动服务方式一：`sudo /etc/init.d/tomcat-产品名-端口号 start`  
关闭服务方式一：`sudo /etc/init.d/tomcat-产品名-端口号 stop`  
重启服务方式一：`sudo /etc/init.d/tomcat-产品名-端口号 restart`

启动服务方式二： `sudo service tomcat-产品名-端口号 start`

关闭服务方式二： `sudo service tomcat-产品名-端口号 stop`

重启服务方式二： `sudo service tomcat-产品名-端口号 restart`

## 2) 服务开机启动配置：

开机启动：

`sudo chkconfig --add tomcat-产品名-端口号`

`sudo chkconfig tomcat-产品名-端口号 on`

开机不启动：

`sudo chkconfig tomcat-产品名-端口号 off`

PS：推荐使用方式一启动、关闭或重启 tomcat 服务，因为方式二无法 table 补齐

**注意：**服务端口小于 1024 时会以 root 身份启动 tomcat 服务，否则端口无法启动

## 4.2.2 Centos7

**备注：**centos7 中 systemctl 启动 tomcat 服务使用的是 tomcat 用户

如果是 deploy 用户，执行以下操作时，记得在命令前边加上 sudo

1) Tomcat 服务的启动文件在 `/lib/systemd/system/tomcat-产品名-端口号.service`

2) 启动服务： `sudo systemctl start tomcat-产品名-端口号.service`

关闭服务： `sudo systemctl stop tomcat-产品名-端口号.service`

**注意：**服务端口小于 1024 时会以 root 身份启动 tomcat 服务，否则端口无法启动

## 4.3 用户身份

确保部署用户的身份是 root 或者 deploy 等 sudo 用户，因为脚本需要 root 的权限

## 4.4 修改文件需谨慎

在不熟悉脚本文件参数的情况下，不要贸然对任何文件进行修改；如果对文件进行了修改，需要先在虚拟环境中测试运行效果

## 4.5 防火墙

参考附录

## 4.6 JVM 堆大小

如下图：部署 tomcat 时，需要选择 JVM 堆大小。默认值是 2G，输入 1 表示 1G，2 表示 2G，4 表示 4G……；也可以小于 1G，比如输入 0.25，表示 0.25G。

注意：JVM 不能小于 0.2G，而且服务器的剩余内存不能小于 128M

```
=====
      脚本部署tomcat实例
=====

1  初始化熵池值      <输入 1 >
2  初始化sudo用户    <输入 2 >
3  安装JDK           <输入 3 >
4  安装tomcat实例     <输入 4 >
5  安装jmx远程监控   <输入 5 >
6  删除tomcat实例     <输入 d >
7  exit              <输入 0|e|q >

请输入您的选择：1|2|3|4|5|0|d >>: 4
tomcat instances service_group_name <a-z,A-Z,0-9,-> <default:SS0> Input: ORM
tomcat instances service_port <default=9000> Input:
tomcat instances shutdown_port <1024-65535|-1(close shutdown port)> <default=9001> Input:
JVM Xmx <0.2G+;1G;2G;4G;6G;8G> USAGE:< 1|2(default)|4|6|8|0.2+|0 exit> Input: 0.25
```

## 4.7 安装步骤

### 4.7.1 安装界面概述

#### 1) 中文版界面

执行命令：sh main.sh

```
[deploy@localhost centos-tomcat]$ sh main.sh
```

```
+-----+
|                                     |
|          =====                 |
|          脚本部署tomcat实例       |
|          =====                 |
|                                     |
|      1  初始化sudo用户           <输入 1 > |
|      2  安装tomcat实例           <输入 2 > |
|      3  安装jmx远程监控          <输入 3 > |
|      4  删除tomcat实例           <输入 d > |
|      5  exit                     <输入 0|e|q > |
|                                     |
+-----+
```

请输入您的选择：1|2|3|4|5|0|d >>:

2) 英文版界面（**当中文界面乱码时使用**）

执行命令：sh main\_en.sh

```
[deploy@localhost centos-tomcat]$ sh main_en.sh
```

```
+-----+
|                                     |
|          =====                 |
|          install tomcat          |
|          =====                 |
|                                     |
|      add user deploy             <input  1 > |
|      install tomcat              <input  2 > |
|      config jmx                  <input  3 > |
|      remove tomcat               <input  d > |
|      exit                        <input 0|e|q > |
|                                     |
+-----+
```

input:1|2|3|4|5|0|d >>:

## 4.7.2 添加 sudo 用户

- 1) 作用：该用户用来以后维护服务器。主要是为了控制过度使用 root 权限
- 2) 执行方式：输入 1 即可执行；
- 3) 执行结果：返回执行成功的提示，或者报错信息

## 4.7.3 安装 tomcat

### 1. 作用：部署 tomcat 单实例或多实例

1.1) 单实例：选项 2 执行成功一次即可

1.2) 多实例：多次执行选项 2 则会部署多个 tomcat 实例

### 2. 执行方式：输入 2 即可执行；下图红色圈中的部分：

2.1) service\_name 输入产品名（不能包含端口号，因为部署脚本会自动组合 tomcat-服务名-端口号），该值关系到 tomcat 的安装目录名；

**注意：**产品名只能是大小写字母-组成；比如 ORM-REST

2.2) service\_port 是 tomcat 提供 HTTP 服务的端口；如果直接回车，默认：9000

2.3) remote\_port 是 tomcat 的远程管理端口；如果直接回车，默认：服务端口+1

```
[deploy@localhost centos-tomcat]$ sh main.sh
+-----+
|                                     |
|          =====                  |
|          脚本部署tomcat实例        |
|          =====                  |
|                                     |
|      1  初始化sudo用户      <输入 1 > |
|      2  安装tomcat实例      <输入 2 > |
|      3  安装jmx远程监控      <输入 3 > |
|      4  删除tomcat实例      <输入 d > |
|      5  exit                  <输入 0|e|q > |
|                                     |
+-----+

请输入您的选择: 1|2|3|4|5|0|d >>: 2
tomcat instances service_group_name <a-z,A-Z,0-9,-> <default:SSO> Input: ORM 产品名
tomcat instances service_port <default=9000> Input: 9010 tomcat服务端口号
tomcat instances shutdown_port <1024-65535|-1(close shutdown port)> <default=9011> Input: 9011
JVM Xmx <0.2G+;1G;2G;4G;6G;8G> USAGE:< 1|2(default)|4|6|8|0.2+|(0 exit)> Input: 1

service_group_name=ORM
service_port=9010
shutdown_port=9011
JVM xmx=1G
```

tomcat的 shutdown 端口号

tomcat的jvm堆大小 单位GB 输入范围0.2-8

请确认以上输入的参数是否正确： (1 确定)|(2 不确定) 1 以上配置参数检查无误，输入1开始安装  
以上配置参数有错误，则输入2重新输入所有参数

### 3. 执行过程中的提示信息：

3.1) 端口冲突，原因有如下两种：

3.1.1) 端口和已经正在使用中的应用端口冲突：

解决方法：要么更换端口，如果该端口的服务没有意义，可以停止该端口的服务，注意：



线上服务器需要谨慎处理

```
请输入您的选择: 1|2|3|4|5|0|d >>: 2
tomcat instances service_group_name <a-z,A-Z,0-9,-> <default:SS0> Input: ORM
tomcat instances service_port <default=9000> Input: 9010
Error: 9010 is using
脚本执行失败
```

3.2) 如果已经存在同名的 tomcat 实例目录，报错如下：

```
请输入您的选择: 1|2|3|4|5|0|d >>: 2
tomcat instances service_group_name <a-z,A-Z,0-9,-> <default:SS0> Input: ORM
tomcat instances service_port <default=9000> Input: 9010
Error: /lib/systemd/system/tomcat-ORM-9010.service is existed;you must check it or remove it
脚本执行失败
```

注意：subdir=\${dirname}-\${service\_port}；即：实例目录名=产品名-端口号；如下图中的两个实例名，subdir 分别是 ptss-8080 和 ptss-8081

```
[root@free3 ~]# ls /linkcld/tomcat8/
ptss-8080  ptss-8081
```

3.3) tomcat 实例的启动文件已存在

3.3.1) centos6 中：

**/etc/init.d/tomcat-组名-端口号** 文件已存在，这是启动 tomcat 实例服务的文件，如果报错，需要检查该文件是否有效，确认不需要的情况下，可以删除，重新部署

3.3.2) centos7 中：

**/lib/systemd/system/tomcat-组名-端口号.service** 文件已存在，这是启动 tomcat 实例服务的文件，如果报错，需要检查该文件是否有效，确认不需要的情况下，可以删除，重新部署

#### 4.7.4 可选择安装：catalina-jmx-remote.jar（监控 tomcat 实例）

1. 作用：提供 web 界面远程监测 tomcat 实例的运行状态

2. 执行方式：输入 3 即可执行；

3. 执行过程：

3.1) 执行成功

3.1.1) 首先输入 **已经部署的 tomcat 的产品名**

3.1.2) 然后输入 service\_port 表示 **已经部署的 tomcat 的服务端口**

3.1.3) **start\_port1** 表示远程监控连接的端口，默认：服务端口+2

3.1.4) **start\_port2** 表示 rmiServerPortPlatform 端口，直接回车，默认：start\_port1+1

3.1.5) 对输入的内容进行确认，无误后选择 1 确定，如果有问题，选择 2 即可退出

```
[deploy@localhost centos-tomcat]$ sh main.sh
+-----+
|                                     |
|          =====                  |
|          脚本部署tomcat实例        |
|          =====                  |
|                                     |
|      1  初始化sudo用户      <输入 1 > |
|      2  安装tomcat实例      <输入 2 > |
|      3  安装jmx远程监控    <输入 3 > |
|      4  删除tomcat实例      <输入 d > |
|      5  exit                <输入 0|e|q > |
|                                     |
+-----+
请输入您的选择: 1|2|3|4|5|0|d >>: 3
tomcat instances service_group_name <a-z,A-Z,0-9,-> <default:ORM> : ORM
tomcat instances service_port <default=9010 Enter> : 9010
tomcat instances start_port1 <default=9012 Enter> : 9012
tomcat instances start_port2 <default=9013 Enter> : 9013

service_group_name=ORM
service_port=9010
rmiRegistryPortPlatform=9012
rmiServerPortPlatform=9013

请确认以上输入的参数是否正确: (1 确定)|(2 不确定) 1
```

已部署tomcat的产品名

已部署tomcat的服务端口号

这两个都是远程监控端口，其中9012是远程调用，9013是内部通信

参数无误，则输入1安装  
参数有误，则输入2重来

#### 4. 执行过程中的提示信息:

4.1) 首先输入已经部署 tomcat 实例的产品名和服务端口号，如果检测到已部署的实例目录不存在，则会报错。

4.2) rmiRegistryPortPlatform 端口或 rmiServerPortPlatform 端口与现有使用中的端口冲突

#### 5. 验证远程监控:

5.1) 找一台安装了 JDK 的主机，而且该主机有图形化界面

5.2) 在 windows 系统的 cmd 命令行，或者 linux 系统命令行执行 jconsole 命令，出现如下界面后，输入连接 ip:port，点击连接，即可打开界面查看详情



9012 是上图安装过程中设置的 rmiRegistryPortPlatform 端口号

#### 4.7.5 删除 tomcat 实例

```
[deploy@localhost centos-tomcat]$ sh main.sh
```

```

=====
      脚本部署tomcat实例
=====

1  初始化sudo用户      <输入 1 >
2  安装tomcat实例      <输入 2 >
3  安装jmx远程监控     <输入 3 >
4  删除tomcat实例      <输入 d >
5  exit                <输入 0|e|q >

```

请输入您的选择: 1|2|3|4|5|0|d >>: **d**

tomcat instances service\_group\_name <a-z,A-Z,0-9,-> : **ORM**

tomcat instances service\_port <default=9000 Enter> : **9010**

service\_group\_name=**ORM**

service\_port=**9010**

已部署tomcat  
的产品名

已部署tomcat的  
服务端口号

请确认以上输入的参数是否正确: (1 确定)|(2 不确定) **1**

## 4.7.6 其他选项

1) 执行方式：输入 0 或者 e、E、q、Q 即可退出安装程序

2) 操作过程中的提示信息：

2.1) 如果输入的选项不在程序提供的范围内，会报错

请输入您的选择：1|2|3|d|0|e|q >>: 7

您的输入有误，请重新输入！

2.2) 如果输入为空，直接回车，提示如下；如果超时 20 秒没有选择，也会出现该提示

请输入您的选择：1|2|3|d|0|e|q >>:

您的输入为空，请重新输入！

## 4.7.7 自定义 tomcat 版本

### 1. 操作方法：

在 centos-tomcat8-vxxx 部署脚本中，有一个 software 目录：如下图

```
[deploy@fw116 ~]$ cd /linkcld/software/centos-tomcat8-v20180925/  
[deploy@fw116 centos-tomcat8-v20180925]$ ls software/  
apache-tomcat-8.5.33.tar.gz catalina-jmx-remote.jar rng-tools.rpm
```

当前 tomcat 版本是 8.5.33；可以删除 apache-tomcat-8.5.33.tar.gz，自己从官网下载 linux 版本的 tomcat 拷贝到 software 目录中。

比如：下载 apache-tomcat-8.5.15.tar.gz，拷贝到 centos-tomcat8-v20180925/software 目录中，不需要其他操作，然后执行 sh main.sh 即可。

### 2. 注意事项：

tomcat 安装包必须是 **apache-tomcat-xxx.tar.gz**，前缀和后缀是固定的。

## 第 5 章 部署 jar

脚本下载参考第一章

### 5.1 环境要求

- 1) 确保服务器的操作系统是 CentOS 6 或 7，查看系统版本的方式：  
`cat /etc/centos-release`
- 2) 确保系统剩余内存至少 1G 以上，因为每个 tomcat 实例的 jdk 运行内存是 1G+

### 5.2 脚本功能

- 1) 新部署 jar 项目并注册服务，设置开机自启动
- 2) 已存在的 jar 项目注册服务，设置开机启动

### 5.3 服务管理和开机启动

#### 1) 服务管理

**注意：**服务端口小于 1024 时会无法启动 jar 服务

- 1.1) cd 到 jar 项目目录中
- 1.2) 

<code>./startup.sh</code>	启动
<code>./shutdown.sh</code>	关闭
<code>./restart.sh</code>	重启
<code>./status.sh</code>	查看是否在运行
- 2) 开机启动配置：请参考“附录”

### 5.4 用户身份

确保部署用户的身份是 root 或者 deploy 等 sudo 用户，因为脚本需要 root 的权限

### 5.5 防火墙

参考附录



## 5.6 安装步骤

### 5.6.1 安装界面概述

#### 3) 中文版界面

执行命令: sh main.sh

```
[root@fwl12 jar_install20180921]# sh main.sh
+-----+
|                                     |
|          =====                  |
|          脚本部署jar    第一步      |
|          =====                  |
|                                     |
|          1 新部署jar项目    <输入 1 安装 > |
|          2 已存在jar项目    <输入 2 安装 > |
|          0 exit              <输入 0|e|E > |
|                                     |
+-----+
+-----+
|                                     |
|          =====                  |
|          脚本部署jar    第二步      |
|          =====                  |
|                                     |
|          1 jar文件可直接启动服务  <输入 1 > |
|          2 java -jar xxx启动服务  <输入 2 > |
|          0 返回上一级菜单          <输入 0|e|E > |
|                                     |
+-----+
```

#### 4) 英文版界面（**当中文界面乱码时使用**）

执行命令: sh main\_en.sh

```

=====
install jar    step1
=====

1 new jar project      <input 1>
2 existed jar project  <input 2>
0 exit                 <input 0|e|E >

=====
install jar    step2
=====

1 ./xxx.jar can start  <input 1>
2 java -jar xxx.jar start <input 2>
0 return up level      <input 0|e|E >

```

## 5.6.2 部署新项目

1) 第一步: sh main.sh 然后输入 1

```

=====
脚本部署jar    第一步
=====

1 新部署jar项目    <输入 1 安装 >
2 已存在jar项目    <输入 2 安装 >
0 exit              <输入 0|e|E >

请输入您的选择: 1|2|0|e|E >>: 1

```

2) 第二步: 根据 jar 文件类型, 输入 1 或 2:

2.1) jar 项目 build 以后, 通过 jar 文件可以直接启动服务, 比如执行 ./agent.jar 即可直

接启动服务；则输入 1

2.2) jar 包需要借助 java -jar agent.jar .....启动服务；则输入 2

详细操作步骤如下：

**备注：jar 的服务名可以包含端口号**

```

=====
脚本部署jar  第二步
=====

1 jar文件可直接启动服务 <输入 1 >
2 java -jar xxx启动服务 <输入 2 >
0 返回上一级菜单 <输入 0|e|E >

请输入您的选择：1|2|0|e|E >>: 1 → jar项目build之后, 可以 ./xxx.jar启动jar项目, 则输入1, 否则输入2
Please Input: e|E=(return upper level)|service_name(product|project)=application-agent → 注册的服务名, 同时也是目录名
Please Input: e|E=(return upper level)|jar_file_path(zip|tar|jar)=application-agent.zip → jar项目文件路径, 绝对路径或相对路径都可以
Please Input JAVA_OPTS(heap_size 16-31744MB) <(e|E=return upper level)|[(default=512)|example 1024|2048|4096>: 32
Configure jmxremote.port? Input: (1024-65535|n|N <default= N >)=10001
basedir="/linkcd/jarapps"
service_name="application-agent"
jar_packet="application-agent.zip"
JAVA_HOME="/usr/java/default"
JAVA_OPTS="-Xms32m -Xmx32m -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=10001 -Dcom.sun.management.
thenticate=false -Dcom.sun.management.jmxremote.ssl=false"
check variables is right? (y|Y sure)|(e|E return upper level): y → 配置正确时, 输入y或Y, 开始安装
配置有误时, 输入n或N, 重新配置
  
```

远程监控端口号  
直接按回车键则跳过

jdk堆大小  
单位是M  
范围16-31744

## 5.6.3 已存在项目注册服务

1) 第一步：sh main.sh 然后输入 2

```

=====
脚本部署jar  第一步
=====

1 新部署jar项目 <输入 1 安装 >
2 已存在jar项目 <输入 2 安装 >
0 exit <输入 0|e|E >

请输入您的选择：1|2|0|e|E >>: 2
  
```

2) 第二步：根据 jar 文件类型，输入 1 或 2：

2.1) jar 项目 build 以后，通过 jar 文件可以直接启动服务，比如执行 ./agent.jar 即可直接启动服务；则输入 1

2.2) jar 包需要借助 java -jar agent.jar .....启动服务；则输入 2



```

=====
脚本部署jar  第二步
=====

1 jar文件可直接启动服务 <输入 1 >
2 java -jar xxx启动服务 <输入 2 >
0 返回上一级菜单 <输入 0|e|E >

请输入您的选择: 1|2|0|e|E >>: 2
Please Input: e|E=(return upper level)|/linkcd/jarapps/$project_directory_name=lbs-receive3.0
Please Input JAVA_OPTS(heap_size 16-31744MB) <(e|E=return upper level)|(default=512)|example 1024|2048|4096>: 2048
Configure jmxremote.port ? Input: (1024-65535|n|N <default= N >)=10002
basedir="/linkcd/jarapps"
service_name="lbs-receive3.0"
project_path="/linkcd/jarapps/lbs-receive3.0"
JAVA_HOME="/usr/java/default"
JAVA_OPTS="-Xms2048m -Xmx2048m -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=10002 -Dcom.sun.
e.authenticate=false -Dcom.sun.management.jmxremote.ssl=false"
check variables is right ? (y|Y sure)|(e|E return upper level): y

```

必须是/linkcd/jarapps/目录中已存在的jar项目  
输入该jar项目的目录名称即可  
注册的服务名等于该jar项目目录名称(无法更改)

远程监控端口号  
直接回车或输入n或N则跳过

jvm堆大小  
单位M  
范围16-31744

确认配置信息正确: 输入y或Y开始安装  
配置信息有误: 输入n或N重新配置

## 5.6.4 重新注册服务

### 1. 停止 jar 服务:

- 1) cd 到 jar 项目目录中
- 2) ./shutdown.sh

### 2. 删除配置和数据:

- 1) sudo rm -f /lib/systemd/system/注册的服务名 或者 sudo rm -f /etc/init.d/注册的服务名。慎重!!! 不要删错文件
- 2) sudo rm -f /linkcd/jarapps/jar 项目目录/\*.sh

### 3. 按重新注册服务:

按照上面已存在项目注册服务, 重新部署

## 第 6 章 部署 Oracle

### 6.1 运行环境

CPU: 8 核及以上

内存: 16G 或以上

数据盘: nT(建议  $n \geq 0.5$ , 根据实际需求)

操作系统: CentOS 7.7



## 6.2 部署 Oracle 过程

### 6.2.1 数据库版本

统一用 11.2.0.4 企业版本的 oracle，版本名称：Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production。

安装包下载地址：<http://192.168.90.117/software/oracle/linux/>

安装包文件名称如下：

 p13390677\_112040\_Linux-x86-64\_1of7.zip  
 p13390677\_112040\_Linux-x86-64\_2of7.zip

### 6.2.2 数据盘挂载

生产环境中，首先对磁盘进行分区。需要在创建 oracle 用户之前，挂载数据盘到 /home/oracle 目录，使用磁盘分区脚本操作，不要手动创建 /home/oracle 目录。

### 6.2.3 参数修改

#### 1、limits.conf 文件修改

修改用户的 SHELL 的限制，修改 /etc/security/limits.conf 文件

输入命令 `vi /etc/security/limits.conf` i 键进入编辑模式增加以下内容

```
oracle soft nproc 2047
oracle hard nproc 16384
oracle soft nofile 1024
oracle hard nofile 65536
```

Esc 退出编辑，“:wq”回车存盘退出

#### 2、login 文件修改

修改 /etc/pam.d/login 文件

输入命令：`vi /etc/pam.d/login`，i 键进入编辑模式，将下列内容加入该文件末尾。

```
session required /lib64/security/pam_limits.so  
session required pam_limits.so
```

Esc 退出编辑，“:wq”回车存盘退出

### 3、sysctl.conf 文件修改

输入命令：vi /etc/sysctl.conf，i 键进入编辑模式，将下列内容加入该文件：

```
fs.file-max = 6815744  
fs.aio-max-nr = 1048576  
kernel.shmall = 2097152  
kernel.shmmax = 2147483648  
kernel.shmmni = 4096  
kernel.sem = 250 32000 100 128  
net.ipv4.ip_local_port_range = 9000 65500  
net.core.rmem_default = 4194304  
net.core.rmem_max = 4194304  
net.core.wmem_default = 262144  
net.core.wmem_max = 1048576
```

Esc 退出编辑，“:wq”回车存盘退出

刷新输入：sysctl -p

### 4、/etc/profile 文件修改

输入命令：vi /etc/profile，i 键进入编辑模式，将下列内容加入该文件。

```
if [ $USER = "oracle" ]; then  
    if [ $SHELL = "/bin/ksh" ]; then  
        ulimit -p 16384  
        ulimit -n 65536  
    else  
        ulimit -u 16384 -n 65536  
    fi  
fi
```

```
fi
```

Esc 退出编辑，“:wq”回车存盘退出

## 6.2.4 安装前操作

### 1、安装依赖包

执行下面的语句，查看软件包的版本是不是大于等于需求：

#### 1.1) 离线安装

离线安装脚本：

<http://192.168.90.117/software/oracle/oracle-rpms.tar.gz>

#### 1.2) yum 源安装：和离线安装 2 选 1 就行了

```
sudo yum -y install epel-release
```

执行下面的语句更新软件包：

```
sudo yum -y install binutils compat-libstdc++-33 elfutils-libelf \
elfutils-libelf-devel elfutils-libelf-devel-static gcc gcc-c++ \
glibc glibc-common glibc-devel glibc-headers kernel-headers \
ksh libaio libaio-devel libgcc libgomp libstdc++ libstdc++-devel \
make sysstat unixODBC unixODBC-devel
```

#### 1.3) 检查依赖包是否全部安装成功

```
sudo rpm -q binutils compat-libstdc++-33 elfutils-libelf \
elfutils-libelf-devel elfutils-libelf-devel-static gcc gcc-c++ \
glibc glibc-common glibc-devel glibc-headers kernel-headers \
ksh libaio libaio-devel libgcc libgomp libstdc++ libstdc++-devel \
make sysstat unixODBC unixODBC-devel
```

### 2、创建用户和组

创建相关用户和组，作为软件安装和支持组的拥有者。

创建 Oracle 用户和密码, 输入如下命令:

```
sudo groupadd dba
sudo groupadd oinstall
sudo useradd -d /home/oracle -g dba -G oinstall oracle
sudo cp /etc/skel/.bash* /home/oracle/
sudo chown -R oracle.dba /home/oracle/
sudo chmod 700 /home/oracle/
sudo passwd oracle
```

然后会让你输入密码(linkld123456)，密码任意输入 2 次，但必须保持一致，回车确认。

### 3、创建数据库软件目录

创建数据库软件目录和数据文件存放目录，目录的位置，根据自己的情况来定，注意磁盘空间即可，输入如下命令:

```
mkdir -p /home/oracle/app/{oracle,oraInventory,oradata}
```

### 4、更改目录属主

更改目录属主为 Oracle 用户所有，输入命令:

```
chown -R oracle:oinstall /home/oracle/app
```

### 5、配置环境变量

配置 oracle 用户的环境变量，首先，切换到新创建的 oracle 用户下，

输入: su - oracle ，然后直接再输入 : vi .bash\_profile i 键进入编辑模式，将下列内容加入该文件末尾

```
export ORACLE_BASE=/home/oracle/app/oracle
export ORACLE_HOME=$ORACLE_BASE/product/11.2.0/dbhome_1
export ORACLE_SID=orcl
export PATH=$PATH:$HOME/bin:$ORACLE_HOME/bin
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/lib
```


编辑完成后按 Esc 键，输入 “:wq” 存盘退出

## 6.2.5 安装 Oracle

### 1、解压安装程序

首先将下载的 Oracle 安装包复制到 linux 中，压缩包如下：

 p13390677\_112040\_Linux-x86-64\_1of7.zip

 p13390677\_112040\_Linux-x86-64\_2of7.zip

用 SSH 或者其他 ftp 工具拷贝，拷贝完成后，运行解压命令如下：

```
sudo yum -y install unzip --安装解压包（可忽略）  
unzip p13390677_112040_Linux-x86-64_1of7.zip  
unzip p13390677_112040_Linux-x86-64_2of7.zip
```

解压完成后 cd 进入其解压后的目录 database

输入命令如下：

```
cd database
```

使用 ls 命令也可以查看解压后 database 所包含的文件

### 2、复制响应文件模板

Oracle 用户登入，su - oracle，输入如下命令：

```
mkdir etc  
cp /home/oracle/database/response/* /home/oracle/etc/
```

### 3、设置响应文件权限

切换到 root 用户，su - root，输入如下命令：

```
chown -R oracle.dbg /home/oracle/etc  
chmod 700 /home/oracle/etc/*.rsp
```

### 4、静默安装 oracle 软件

切换回 oracle 用户，su - oracle

修改安装 Oracle 软件的响应文件 /home/oracle/etc/db\_install.rsp

/home/oracle/app 可以改成自定义的数据目录，输入如下命令：

```
vi /home/oracle/etc/db_install.rsp
```

i 键进入编辑模式，要修改的参数如下：

```
oracle.install.option=INSTALL_DB_SWONLY //29 行 安装类型
ORACLE_HOSTNAME=localhost //37 行 主机名称，通过 hostname 命令获得
UNIX_GROUP_NAME=oinstall //42 行 安装组
INVENTORY_LOCATION=/home/oracle/app/oraInventory //47 行 INVENTORY 目录
SELECTED_LANGUAGES=en, zh_CN, zh_TW //78 行 选择语言
ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1 //83 行
oracle_home 和上面配置环境变量里的目录保持一致
ORACLE_BASE=/home/oracle/app/oracle //88 行 oracle_base, 和 INVENTORY 目录要同级
oracle.install.db.InstallEdition=EE //99 行 oracle 版本，EE 为企业版
oracle.install.db.DBA_GROUP=dba //142 行 dba 用户组
oracle.install.db.OPER_GROUP=oinstall //147 行 oper 用户组
oracle.install.db.config.starterdb.type=GENERAL_PURPOSE //160 行 数据库类型
oracle.install.db.config.starterdb.globalDBName=orcl //165 行
globalDBName
oracle.install.db.config.starterdb.SID=orcl //170 行 SID
oracle.install.db.config.starterdb.characterSet=ZHS16GBK //指定字符集
oracle.install.db.config.starterdb.installExampleSchemas=false //是否
载入模板示例
oracle.install.db.config.starterdb.enableSecuritySettings=true //是否
启用安全设置
oracle.install.db.config.starterdb.memoryLimit=512 //192/200 行 自动管理内存的最小内存(M)
```

```
oracle.install.db.config.starterdb.password.ALL=LINKCLD_ORCL //23 行设定所有数据库用户使用同一个密码
```

```
DECLINE_SECURITY_UPDATES=true //385 行 设置安全更新
```

改完之后，Esc 退出编辑，“:wq” 回车存盘退出

在 oracle 用户下，su - oracle

切换到 database 目录下 cd database，输入如下命令：

```
./runInstaller -silent -force -responseFile /home/oracle/etc/db_install.rsp
```

开始安装，安装中，如果提示[WARNING]不必理会，此时安装程序仍在进行，如果出现[FATAL]，则安装程序已经停止了。执行直到出现如下页面：

```
The following configuration scripts need to be executed as the "root" user.
```

```
#!/bin/sh
```

```
#Root scripts to run
```

```
/home/oracle/app/oraInventory/orainstRoot.sh
```

```
/home/oracle/app/product/11.2.0/db_1/root.sh
```

```
To execute the configuration scripts:
```

1. Open a terminal window
2. Log in as "root"
3. Run the scripts
4. Return to this window and hit "Enter" key to continue

```
Successfully Setup Software.
```

## 5、执行 root.sh

新开一个窗口，切换到 root 用户，su - root

然后执行以上两个脚本，命令如下：

```
sh /home/oracle/app/oraInventory/orainstRoot.sh
```



```
sh /home/oracle/app/product/11.2.0/db_1/root.sh
```

脚本执行完成后，回到以上界面，按回车键

## 6、增加 oracle 用户的环境变量

在 oracle 用户下，su - oracle，编辑文档，输入如下命令：

```
vi ~/.bash_profile
```

键 i 进入编辑模式，在最后直接增加以下内容：

```
export TNS_ADMIN=$ORACLE_HOME/network/admin
export PATH=.:${PATH}:%HOME/bin:%ORACLE_HOME/bin
export PATH=${PATH}:/usr/bin:/bin:/usr/bin/X11:/usr/local/bin
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:%ORACLE_HOME/lib
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:%ORACLE_HOME/oracm/lib
export
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/lib:/usr/lib:/usr/local/lib
export CLASSPATH=${CLASSPATH}:%ORACLE_HOME/JRE
export CLASSPATH=${CLASSPATH}:%ORACLE_HOME/JRE/lib
export CLASSPATH=${CLASSPATH}:%ORACLE_HOME/jlib
export CLASSPATH=${CLASSPATH}:%ORACLE_HOME/rdbms/jlib
export CLASSPATH=${CLASSPATH}:%ORACLE_HOME/network/jlib
export LIBPATH=${CLASSPATH}:%ORACLE_HOME/lib:%ORACLE_HOME/ctx/lib
export ORACLE_OWNER=oracle
export SPFILE_PATH=%ORACLE_HOME/dbs
export ORA_NLS10=%ORACLE_HOME/nls/data
```

加完后，Esc 退出编辑，“:wq”回车存盘退出

输入如下命令使以上的设置生效：

```
source /home/oracle/.bash_profile
```

## 7、配置静默网络

Oracle 用户下，su - oracle，输入如下命令：

```
$ORACLE_HOME/bin/netca /silent /responseFile  
/home/oracle/etc/netca.rsp
```

如果报如下错:

```
UnsatisfiedLinkError exception loading native library: njni11  
java.lang.UnsatisfiedLinkError:  
/home/oracle/app/oracle/product/11.2.0/db_1/lib/libnjni11.so:  
libclntsh.so.11.1: cannle: No such file or directory
```

解决方法:

```
cp  
/home/oracle/app/oracle/product/11.2.0/db_1/inventory/Scripts/ext/lib/libcl  
ntsh.so.11.1 /home/oracle/app/oracle/product/11.2.0/db_1/lib
```

再次执行如下命令:

```
netca /silent /responseFile /home/oracle/etc/netca.rsp
```

提示如下:

```
Listener started successfully.  
Listener configuration complete.  
Oracle Net Services configuration successful. The exit code is 0
```

输入如下命令查看:

```
ss -antp|grep 1521
```

## 8、修改静默安装配置

修改仅安装数据库的响应文件, 输入如下命令:

```
vi /home/oracle/etc/dbca.rsp
```

i 键进入编辑模式, 修改下列参数:

```
GDBNAME="orcl.localdomain " //78 行 全局数据库的名字=SID+主机域名  
SID="orcl" //149 行 SID  
CHARACTERSET="ZHS16GBK" //415 行 编码  
NATIONALCHARACTERSET="AL16UTF16" //425 行 编码
```

```
DATAFILEDESTINATION = /home/oracle/app/oradata //356 行
```

## 9、进行静默安装数据库

Oracle 用户下，su - oracle 输入如下命令回车：

```
$ORACLE_HOME/bin/dbca -silent -responseFile /home/oracle/etc/dbca.rsp
```

执行完以上命令之后，屏幕会变成白板，其实此时是输入 oracle 的 sys 和 system 的密码，输入两次，然后会显示执行的百分比进度

同时会提示：oracle 的日志文件是 /u01/app/oracle/cfgtoollogs/dbca/orcl/orcl.log

然后检查并修改 oracle 命令的权限(6751)：

如果权限不对，输入如下命令修改：

```
chmod 6751 $ORACLE_HOME/bin/oracle
```

## 10、建库后实例检查

启动监听端口，输入如下命令：

```
lsnrctl start
```

建库后实例检查，输入如下命令：

```
ps -ef | grep ora | grep -v grep | wc -l
ps -ef | grep ora | grep -v grep
ps -ef|grep oracle|egrep -v "bash|su|grep|ps"
```

检查到 sqlplus 进程才算正常，可以配置下面的服务文件

```
oracle      3695    3413    0 14:46 pts/1    00:00:00 sqlplus
oracle      11082          1    0 15:19 ?              00:00:00
/u01/app/oracle/product/11.2.0/db_1/bin/tnslsnr LISTENER -inherit
```

建库后监听状态检查，输入如下命令：

```
lsnrctl status
```

如果出现以下错误：

```
lsnrctl:
error          while          loading          shared          libraries:
```

```
/u01/app/oracle/product/11.2.0/db_1/lib/libclntsh.so.11.1: cannot restore  
segment prot after reloc: Permission denied
```

解决办法，切换到 root 用户，su - root，输入以下命令：

```
setenforce 0
```

## 6.2.6 设置开机启动

### 1、配置修改

切换到 root 用户，su - root，输入如下命令：

```
vi /home/oracle/app/oracle/product/11.2.0/dbhome_1/bin/dbstart
```

i 键进入编辑模式，将下列内容加入该文件：

```
ORACLE_HOME_LISTNER=$ORACLE_HOME           //在 65 行添加
```

Esc 退出编辑，“:wq”回车存盘退出，输入如下命令：

```
vi /home/oracle/app/oracle/product/11.2.0/dbhome_1/bin/dbshut
```

i 键进入编辑模式，将下列内容加入该文件：

```
ORACLE_HOME_LISTNER=$ORACLE_HOME           //在 39 行添加
```

Esc 退出编辑，“:wq”回车存盘退出，输入如下命令：

```
vi /etc/oratab
```

i 键进入编辑模式，把末尾的 N 改成 Y，如下：

```
orcl:/home/oracle/app/oracle/product/11.2.0/dbhome_1:Y
```

Esc 退出编辑，“:wq”回车存盘退出

### 2、创建启动文件

修改文件配置，切换到 root 用户，su - root，输入如下命令：

```
vi /etc/rc.d/init.d/oracled
```

i 键进入编辑模式把，以下内容粘贴到新文件中，ORACLE\_HOME=自己安装的目录

```
#!/bin/bash
```

```
# chkconfig: 345 90 10

# description: The Oracle Database is an Object-Relational Database
Management System.

#

# processname: oracle

. /etc/rc.d/init.d/functions

LOCKFILE=/var/lock/subsys/oracle

ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1

ORACLE_USER=oracle

case "$1" in
'start')

    if [ -f $LOCKFILE ]; then

        echo $0 already running.

        exit 1

    fi

    echo -n $"Starting Oracle Database:"

    su - $ORACLE_USER -c "$ORACLE_HOME/bin/lsnrctl start"

    su - $ORACLE_USER -c "$ORACLE_HOME/bin/dbstart $ORACLE_HOME"

    ##su - $ORACLE_USER -c "$ORACLE_HOME/bin/emctl start dbconsole"

    touch $LOCKFILE

    ;;

'stop')

    if [ ! -f $LOCKFILE ]; then

        echo $0 already stopping.

        exit 1

    fi

    echo -n $"Stopping Oracle Database:"

    su - $ORACLE_USER -c "$ORACLE_HOME/bin/lsnrctl stop"
```

```
su - $ORACLE_USER -c "$ORACLE_HOME/bin/dbshut"
##su - $ORACLE_USER -c "$ORACLE_HOME/bin/emctl stop dbconsole"
rm -f $LOCKFILE

;;
'restart')
    $0 stop
    $0 start
    ;;
'status')
    if [ -f $LOCKFILE ]; then
        echo $0 started.
    else
        echo $0 stopped.
    fi
    ;;
*)
    echo "Usage: $0 [start|stop|status]"
    exit 1
esac
exit 0
```

Esc 退出编辑，“:wq”回车存盘退出

修改权限配置，同样是 root 用户下，su - root，输入如下命令：

```
chown oracle.dba /etc/init.d/oracled
chmod a+x /etc/init.d/oracled
chkconfig --add oracled
chkconfig oracled on    //添加 oracled 服务开机启动
```

设置 oracle 用户的 sudo 权限，使用 systemctl 命令管理 oracled 服务，输入如下命令：

```
chattr -i /etc/sudoers
visudo
```

i 键进入编辑模式，在该文件的 95 行左右添加一行内容如下：

1) centos7:

```
oracle          ALL=(ALL)          NOPASSWD:
/etc/init.d/oracled,/usr/bin/systemctl * oracled*
```

centos7: oracle 服务的启动关闭重启命令如下：

```
su - oracle //su - root 也可以
sudo systemctl start oracled
sudo systemctl stop oracled
sudo systemctl restart oracled
```

2) centos6:

```
oracle  ALL=(ALL)          NOPASSWD: /etc/init.d/oracled,/sbin/service
oracled *
```

centos6: oracle 服务的启动关闭重启命令如下：

```
su - oracle //su - root 也可以
sudo service oracled start
sudo service oracled stop
sudo service oracled restart
```

## 6.2.7 初始化

- 修改数据库最大连接数为 1500 或者以上。登录 sys 用户，操作如下：
 

```
alter system set processes = 1500 scope = spfile;
```
- 创建用户后，修改用户密码过期改为无限期（默认 180 天）。登录 sys 用户，操作如下：
 

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_LIFE_TIME UNLIMITED;
```
- 修改完成后重启 oracle 服务，配置才会生效。

## 6.2.8 日志清理脚本

### 1、创建日志脚本目录

```
su - root
sudo mkdir -p /linkcld/exec/oracle
```

### 2、编辑清理脚本

```
sudo vim /linkcld/exec/oracle/deloraclelogs.sh
```

添加以下内容（按照该部署教程部署-则无需修改以下内容）：

字体缩小避免错行，直接拷贝内容即可

```
#!/bin/bash

#

source /home/oracle/.bash_profile

if [ ! -d /home/oracle/app/oracle/product/11.2.0/dbhome_1 ]; then

    echo "Failed: /home/oracle/app/oracle/product/11.2.0/dbhome_1 is not existed"

    exit 1

fi

if [ ! -x /home/oracle/app/oracle/product/11.2.0/dbhome_1/bin/lsnrctl ]; then

    echo "Failed: /home/oracle/app/oracle/product/11.2.0/dbhome_1/bin/lsnrctl is not existed"

    exit 1

fi

/home/oracle/app/oracle/product/11.2.0/dbhome_1/bin/lsnrctl set log_status off >/dev/null 2>&1

/home/oracle/app/oracle/product/11.2.0/dbhome_1/bin/lsnrctl trace off >/dev/null 2>&1

find /home/oracle/app/oracle/diag/rdbms/orcl/orcl/trace/* -maxdepth 0 -mtime +3 -exec rm -fr {} \; 2>/dev/null

find /home/oracle/app/oracle/diag/rdbms/orcl/orcl/alert/* -maxdepth 0 -mtime +3 -exec rm -fr {} \; 2>/dev/null

datetime=`date +%Y%m%d%H%M`

cp /home/oracle/app/oracle/diag/rdbms/orcl/orcl/trace/alert_orcl.log /home/oracle/app/oracle/diag/rdbms/orcl/orcl/trace/alert_orcl.log_${datetime}

echo -n "" | tee /home/oracle/app/oracle/diag/rdbms/orcl/orcl/trace/alert_orcl.log

cp /home/oracle/app/oracle/diag/rdbms/orcl/orcl/alert/log.xml /home/oracle/app/oracle/diag/rdbms/orcl/orcl/alert/log.xml_${datetime}
```



```

echo -n "" | tee /home/oracle/app/oracle/diag/rdbms/orcl/orcl/alert/log.xml

find /home/oracle/app/oracle/diag/tnlsnr/$HOSTNAME/listener/trace/* -maxdepth 0 -mtime +3 -exec rm -fr {} \; 2>/dev/null

find /home/oracle/app/oracle/diag/tnlsnr/$HOSTNAME/listener/alert/* -maxdepth 0 -mtime +3 -exec rm -fr {} \; 2>/dev/null

cp /home/oracle/app/oracle/diag/tnlsnr/$HOSTNAME/listener/trace/listener.log /home/oracle/app/oracle/diag/tnlsnr/$HOSTNAME/listener/trace/listener.log_${datetime}

echo -n "" | tee /home/oracle/app/oracle/diag/tnlsnr/$HOSTNAME/listener/trace/listener.log

cp /home/oracle/app/oracle/diag/tnlsnr/$HOSTNAME/listener/alert/log.xml /home/oracle/app/oracle/diag/tnlsnr/$HOSTNAME/listener/alert/log.xml_${datetime}

echo -n "" | tee /home/oracle/app/oracle/diag/tnlsnr/$HOSTNAME/listener/alert/log.xml

/home/oracle/app/oracle/product/11.2.0/dbhome_1/bin/lsnrctl set log_status on >/dev/null 2>&1

/home/oracle/app/oracle/product/11.2.0/dbhome_1/bin/lsnrctl trace 12 >/dev/null 2>&1

if [ $? -eq 0 ]; then

    echo "$(date +%F %T)" oracle log rotation success"

else

    echo "$(date +%F %T)" oracle log rotation failure"

fi

```

保存退出

### 3、授权

```
sudo chown -R oracle:oinstall /linkcd/exec/oracle
```

### 4、创建计划任务

```
sudo crontab -e -u oracle
```

添加以下内容：字体缩小避免错行，直接拷贝内容即可

```
30 2 * * * /bin/sh /linkcd/exec/oracle/deloraclelogs.sh >/linkcd/exec/oracle/exec.log 2>&1
```

保存退出

## 6.3 防火墙

参考附录

## 第 7 章 数据备份方案

### 7.1 MySQL

#### 1、备份方案

全量备份+增量备份

#### 2、备份策略

每周六凌晨 02:30 通过 xtrabackup 工具实现在线全量备份，压缩并上传备份数据到备份服务器，一、三、五、七的凌晨 02:30 通过 xtrabackup 实现在线增量备份，每天上传 binlog 文件到备份服务器实现二次增量备份。

#### 3、安装 xtrabackup

```
sudo tee /etc/yum.repos.d/percona_pxc.repo <<EOF
[percona]
name=percona_repo
baseurl=https://mirrors.tuna.tsinghua.edu.cn/percona/centos/7Server/RPMS/x86_64/
enabled=1
gpgcheck=0
EOF
sudo yum clean all && sudo yum makecache fast
sudo yum install percona-xtrabackup-24 -y --nogpgcheck
```

#### 4、配置 xtrabackup

登录 MySQL 执行以下命令

```
CREATE USER 'bkpuser'@'localhost' IDENTIFIED BY 'Linkld456.';
GRANT RELOAD,LOCK TABLES,PROCESS,REPLICATION CLIENT ON *.* TO 'bkpuser'@'localhost';
FLUSH PRIVILEGES;
```

#### 5、xtrabackup 备份

首先准备一块数据盘用来存储备份数据，使用磁盘分区脚本挂载到/mysql\_bak 目录

假如周日 2020-04-19 开始备份，一个完整的备份周期操作如下：

### 周六全量备份:

```
sudo xtrabackup --defaults-file=/etc/my.cnf --no-timestamp --user=bkpuser --password=Linkcld456.
--socket=/var/lib/mysql/mysql.sock --backup --target-dir=/mysql_bak/xtra_base_20200418
```

### 周一增量备份:

```
xtrabackup --defaults-file=/etc/my.cnf --no-timestamp --user=bkpuser --password=Linkcld456. --socket=/var/lib/mysql/mysql.sock --backup
--target-dir=/mysql_bak/xtra_inc_20200420 --incremental-basedir=/mysql_bak/xtra_base_20200418
```

### 周三增量备份:

```
xtrabackup --defaults-file=/etc/my.cnf --no-timestamp --user=bkpuser --password=Linkcld456. --socket=/var/lib/mysql/mysql.sock --backup
--target-dir=/mysql_bak/xtra_inc_20200422 --incremental-basedir=/mysql_bak/xtra_base_20200420
```

### 周五增量备份:

```
xtrabackup --defaults-file=/etc/my.cnf --no-timestamp --user=bkpuser --password=Linkcld456. --socket=/var/lib/mysql/mysql.sock --backup
--target-dir=/mysql_bak/xtra_inc_20200424 --incremental-basedir=/mysql_bak/xtra_base_20200422
```

## 6、xtrabackup 还原

```
xtrabackup --prepare --apply-log-only --target-dir=/mysql_bak/xtra_base_20200418 --incremental-dir=/mysql_bak/xtra_inc_20200424
xtrabackup --prepare --apply-log-only --target-dir=/mysql_bak/xtra_base_20200418 --incremental-dir=/mysql_bak/xtra_inc_20200422
xtrabackup --prepare --target-dir=/mysql_bak/xtra_base_20200418 --incremental-dir=/mysql_bak/xtra_inc_20200420
xtrabackup --defaults-file=/etc/my.cnf --copy-back --target-dir=/mysql_bak/xtra_base_20200418
```

## 7.2 Oracle

### 1、备份方案

每天对重要数据全量备份

### 2、备份策略

每天凌晨 02:30 通过 expdp 实现在线全量备份，压缩并上传备份数据到备份服务器。

### 3、expdp 使用方法

优点	1. 可导出空表，clob 字段等; 2. 导出效率高
缺点	1. 必须在数据库服务端执行; 2. 数据库必须是 10g 以上版本;
基本语法	

## 1. 导出一个完整数据库[结构+数据]

```
expdp system/manager DIRECTORY=dpdata1 DUMPFILE=full.dmp LOGFILE=full.log FULL=y;
```

## 2. 只导出数据库的结构而不导出里面的数据

```
expdp system/manager DIRECTORY=dpdata1 DUMPFILE=full.dmp CONTENT=metadata_only
LOGFILE=full.log FULL=y;
```

## 3. 导出用户 admin 的所有数据

```
expdp system/manager DIRECTORY=dpdata1 DUMPFILE=admin.dmp LOGFILE=admin.log
schemas=admin
```

## 4. 同时导出 admin 和 visitor 两个用户的所有对象

```
expdp system/manager DIRECTORY=dpdata1 DUMPFILE=admin.dmp LOGFILE=admin.log
schemas=admin,visitor
```

## 5. 只导出一个 Table, 如只导出 admin 用户下面的 testtable 表

```
expdp system/manager DIRECTORY=dpdata1 DUMPFILE=admin.dmp LOGFILE=admin.log
TABLES=admin.testtable
```

## 6. 如果需要导出多个表

```
expdp system/manager DIRECTORY=dpdata1 DUMPFILE=admin.dmp LOGFILE=admin.log
TABLES=admin.table1,admin.table2
```

## 4、impdp 使用方法

```
IMPDP SYS/PSSWD PARFILE=PATH/admin.dmp
```

## 第 8 章 部署 Hbase

### 8.1 运行环境

备注: nT(建议  $n \geq 0.5$ , 根据实际需求)

应用和版本	系统	CPU 和内存最低配置	数据磁盘	数量
Hbase 2 master	CentOS 7.7	4 核 8G	100G*2	1 台
Hbase 2 slave	CentOS 7.7	4 核 16G	nT*6	3 台

## 8.2 准备工作

每个节点都必须执行一遍系统初始化脚本（包括安装 jdk8 包）

### 1、主机名解析（所有节点）：

```
sudo vim /etc/hosts
```

添加以下内容（以下 IP 只是举个例子，根据项目实际情况）：

```
192.168.80.5 master
192.168.80.6 slave1
192.168.80.7 slave2
192.168.80.8 slave3
```

### 2、添加用户（所有节点）：

```
sudo groupadd hadoop
sudo useradd -M -s /sbin/nologin -g hadoop hdfs
sudo useradd -M -s /sbin/nologin -g hadoop zookeeper
sudo useradd -M -s /sbin/nologin -g hadoop yarn
sudo useradd -g hadoop hbase
sudo passwd hbase
```

输入 hbase 密码（linkclld123456），输入两遍

### 3、配置文件描述符（所有节点）：

```
vim /etc/security/limits.conf
```

添加以下内容

```
zookeeper soft nfile 65535
zookeeper hard nfile 65535
zookeeper soft nproc 131072
zookeeper hard nproc 131072
hdfs soft nfile 128000
hdfs hard nfile 128000
hdfs soft nproc 131072
hdfs hard nproc 131072
yarn soft nfile 65535
yarn hard nfile 65535
yarn soft nproc 131072
yarn hard nproc 131072
hbase soft nfile 32000
hbase hard nfile 32000
```

```
hbase soft nproc 16000
hbase hard nproc 16000
```

#### 4、挂载数据盘

执行磁盘分区脚本，挂载第 1 块数据盘到/data1

执行磁盘分区脚本，挂载第 2 块数据盘到/data2

执行磁盘分区脚本，挂载第 3 块数据盘到/data3

.....

注意：这一步会影响 hdfs 的配置!!!

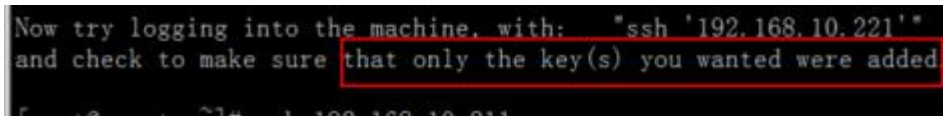
#### 5、配置节点的 root 用户之间免密登录

以下操作都在 master 节点的 root 用户下执行

```
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
ssh-copy-id slave1
```

输入 slave1 的 root 密码

出现 “wanted were added” 说明添加成功了



同理 slave1、slave2 和 slave3

```
ssh-copy-id slave2
```

输入 slave2 的 root 密码

```
ssh-copy-id slave3
```

输入 slave3 的 root 密码

```
scp ~/.ssh/id_rsa slave1:~/.ssh/id_rsa
scp ~/.ssh/id_rsa slave2:~/.ssh/id_rsa
scp ~/.ssh/id_rsa slave3:~/.ssh/id_rsa
```

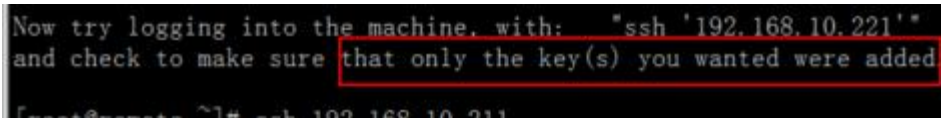
#### 6、配置节点的 hbase 用户之间免密登录（参考上一步）

以下操作都在 master 节点的 hbase 用户下执行

```
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
ssh-copy-id slave1
```

输入 slave1 的 hbase 密码

出现 “wanted were added” 说明添加成功了



同理 slave1、slave2 和 slave3

```
ssh-copy-id slave2
```

输入 slave2 的 hbase 密码

```
ssh-copy-id slave3
```

输入 slave3 的 hbase 密码

```
scp ~/.ssh/id_rsa slave1:~/.ssh/id_rsa
scp ~/.ssh/id_rsa slave2:~/.ssh/id_rsa
scp ~/.ssh/id_rsa slave3:~/.ssh/id_rsa
```

## 8.3 部署 zookeeper 过程

以下操作都在 slave1 节点的 root 用户下执行

### 1、下载 zookeeper 安装包

<http://192.168.90.117/software/bigdata/zookeeper-3.4.14.tar.gz>

### 2、安装 zookeeper

```
tar xf zookeeper-3.4.14.tar.gz -C /linkcd/install
ln -s /linkcd/install/zookeeper-3.4.14/ /linkcd/install/zookeeper
vim /etc/profile.d/zk.sh
```

添加以下内容：

```
export ZK_HOME=/linkcd/install/zookeeper
export PATH=$ZK_HOME/bin:$PATH
```

加载变量：

```
source /etc/profile.d/zk.sh
```

同步变量:

```
scp /etc/profile.d/zk.sh slave2:/etc/profile.d/zk.sh
scp /etc/profile.d/zk.sh slave3:/etc/profile.d/zk.sh
```

### 配置 zookeeper

```
cd /linkcd/install/zookeeper/conf/
cp zoo_sample.cfg zoo.cfg
vim zoo.cfg
```

修改以下内容:

```
tickTime=3000
maxClientCnxns=0
dataDir=/linkcd/install/zookeeper/data
```

添加以下内容:

```
dataLogDir=/linkcd/install/zookeeper/logs
server.1=slave1:2888:3888
server.2=slave2:2888:3888
server.3=slave3:2888:3888
```

### 3、创建目录并同步给其他节点

```
mkdir -p /linkcd/install/zookeeper/{data,logs}
echo 1 > /linkcd/install/zookeeper/data/myid
chown -R zookeeper:hadoop /linkcd/install/zookeeper
scp -r /linkcd/install/zookeeper slave2:/linkcd/install/
scp -r /linkcd/install/zookeeper slave3:/linkcd/install/
```

### 4、slave2 节点配置 zookeeper

以下操作在 slave2 的 root 下执行

```
echo 2 > /linkcd/install/zookeeper/data/myid
source /etc/profile.d/zk.sh
chown -R zookeeper:hadoop /linkcd/install/zookeeper
```

### 5、slave3 节点配置 zookeeper

以下操作在 slave2 的 root 下执行



```
echo 3 > /linkcd/install/zookeeper/data/myid  
source /etc/profile.d/zk.sh  
chown -R zookeeper:hadoop /linkcd/install/zookeeper
```

## 6、配置 zookeeper 的服务文件

以下操作在 **slave1** 的 **root** 下执行

```
vim /usr/lib/systemd/system/zookeeper.service
```

添加以下内容：

```
[Unit]  
Description=zookeeper service  
After=syslog.target network.target  
  
[Service]  
Type=forking  
  
Environment=JAVA_HOME=/usr/java/default  
Environment=ZOO_LOG_DIR=/linkcd/install/zookeeper/logs  
Environment=ZOOCFG=/linkcd/install/zookeeper/conf/zoo.cfg  
ExecStartPre=/usr/bin/cd /linkcd/install/zookeeper  
ExecStart=/usr/bin/nohup /linkcd/install/zookeeper/bin/zkServer.sh start  
$ZOOCFG >/dev/null 2>&1 &  
ExecReload=/bin/kill -s HUP $MAINPID  
ExecStop=/bin/kill -s QUIT $MAINPID  
  
User=zookeeper  
Group=hadoop  
  
RestartSec=20  
Restart=always
```

```
[Install]
```

```
WantedBy=multi-user.target
```

同步到其他节点：

```
scp /usr/lib/systemd/system/zookeeper.service slave2:/usr/lib/systemd/system/zookeeper.service
scp /usr/lib/systemd/system/zookeeper.service slave3:/usr/lib/systemd/system/zookeeper.service
```

## 7、启动 zookeeper 集群

以下操作分别在 **slave1**、**slave2**、**slave3** 的 **root** 下执行

```
systemctl daemon-reload
systemctl enable zookeeper.service
systemctl start zookeeper.service
```

## 8、检查集群状态

集群启动 3 分钟后，分别在 **slave1**、**slave2**、**slave3** 的 **root** 下执行

```
zkServer.sh status
ss -antlp | grep 2181
ss -antlp | grep 2888
ss -antlp | grep 3888
```

注意：2181 和 3888 在所有节点都有监听，2888 只有 **leader** 节点有监听

## 8.4 部署 hdfs 过程

以下操作都在 **master** 节点的 **root** 用户下执行

### 1、下载 hdfs 安装包

<http://192.168.90.117/software/bigdata/hadoop-3.2.1.tar.gz>

### 2、解压 hdfs

```
tar xf hadoop-3.2.1.tar.gz -C /linkcd/install/
ln -s /linkcd/install/hadoop-3.2.1 /linkcd/install/hadoop
```

### 3、配置环境变量

```
sudo tee /etc/profile.d/hdfs.sh <<EOF
export HADOOP_HOME=/linkcd/install/hadoop
```

```
export PATH=\$PATH:\$HADOOP_HOME/bin:\$HADOOP_HOME/sbin
export HADOOP_CONF_DIR=\$HADOOP_HOME/etc/hadoop
EOF
source /etc/profile.d/hdfs.sh
scp /etc/profile.d/hdfs.sh slavel:/etc/profile.d/hdfs.sh
scp /etc/profile.d/hdfs.sh slave2:/etc/profile.d/hdfs.sh
scp /etc/profile.d/hdfs.sh slave3:/etc/profile.d/hdfs.sh
```

#### 4、配置 hadoop-env.sh

```
vi /linkcd/install/hadoop/etc/hadoop/hadoop-env.sh
```

从 46 行开始添加

```
export JAVA_HOME=/usr/java/default
export HADOOP_HOME=/linkcd/install/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=${HADOOP_HOME}/lib/native"
export HADOOP_PID_DIR=$HADOOP_HOME/tmp
export HADOOP_SECURE_PID_DIR=${HADOOP_PID_DIR}
```

#### 5、配置 core-site.xml

```
vi /linkcd/install/hadoop/etc/hadoop/core-site.xml
```

编辑文件内容如下（多块数据盘时，只需要用/data1 即可）：

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
  <property>
    <name>io.file.buffer.size</name>
    <value>131072</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>file:/data1/hdfs/tmp</value>
  </property>
```

```
<property>
  <name>hadoop.security.authorization</name>
  <value>>false</value>
</property>
<property>
  <name>hadoop.proxyuser.root.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.root.groups</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.hadoop.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.hadoop.groups</name>
  <value>*</value>
</property>
</configuration>
```

## 6、配置 hdfs-site.xml

```
vi /linkcld/install/hadoop/etc/hadoop/hdfs-site.xml
```

编辑文件内容如下：

多块数据盘时配置：

```
<value>file:/data1/hdfs/datanode,file:/data2/hdfs/datanode</value>
```

以上是两块盘，多块盘依次类推 file:/data3/hdfs/datanode……

```
<configuration>
```

```
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/data1/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/data1/hdfs/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>2</value>
</property>
<property>
  <name>dfs.blocksize</name>
  <value>128m</value>
</property>
<property>
  <name>dfs.namenode.rpc-address</name>
  <value>master:9000</value>
</property>
<property>
  <name>dfs.namenode.http-address</name>
  <value>master:50070</value>
  <final>true</final>
</property>
<property>
  <name>dfs.namenode.https-address</name>
  <value>master:50470</value>
```

```
</property>
<property>
  <name>dfs.namenode.secondary.http-address</name>
  <value>slave1:50090</value>
</property>
<property>
  <name>dfs.datanode.address</name>
  <value>0.0.0.0:50010</value>
</property>
<property>
  <name>dfs.datanode.ipc.address</name>
  <value>0.0.0.0:50020</value>
</property>
<property>
  <name>dfs.datanode.http.address</name>
  <value>0.0.0.0:50075</value>
</property>
<property>
  <name>dfs.datanode.https.address</name>
  <value>0.0.0.0:50475</value>
</property>
<property>
  <name>dfs.https.port</name>
  <value>50470</value>
</property>
<property>
  <name>dfs.datanode.max.transfer.threads</name>
  <value>16384</value>
```

```
</property>
<property>
  <name>dfs.datanode.balance.bandwidthPerSec</name>
  <value>52428800</value>
</property>
<property>
  <name>dfs.datanode.du.reserved</name>
  <value>21474836480</value>
</property>
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
</property>
<property>
  <name>dfs.permissions</name>
  <value>false</value>
</property>
<property>
  <name>dfs.hosts.exclude</name>
  <value>/linkcd/install/hadoop/etc/hadoop/exclude</value>
  <final>true</final>
</property>
<property>
  <name>dfs.permissions</name>
  <value>false</value>
</property>
</configuration>
```

## 7、配置 mapred-site.xml

```
vi /linkcd/install/hadoop/etc/hadoop/mapred-site.xml
```

编辑文件内容如下：

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
    <final>true</final>
  </property>
  <property>
    <name>mapreduce.jobtracker.http.address</name>
    <value>master:50030</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>master:10020</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>master:19888</value>
  </property>
  <property>
    <name>mapred.job.tracker</name>
    <value>http://master:9001</value>
  </property>
</configuration>
```

## 8、配置 yarn-site.xml

```
vi /linkcd/install/hadoop/etc/hadoop/yarn-site.xml
```

编辑文件内容如下：



```
<configuration>
<!-- Site specific YARN configuration properties -->

  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>master</value>
  </property>

  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>

  <property>
    <name>yarn.resourcemanager.address</name>
    <value>master:8032</value>
  </property>

  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>master:8030</value>
  </property>

  <property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>master:8031</value>
  </property>

  <property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>master:8033</value>
  </property>

  <property>
    <name>yarn.resourcemanager.webapp.address</name>
```

```
<value>master:8088</value>
</property>
<property>
  <name>yarn.resourcemanager.nodes.exclude-path</name>
  <value>/linkcd/install/hadoop/etc/hadoop/exclude</value>
</property>
<property>
  <name>yarn.resourcemanager.am.max-attempts</name>
  <value>4</value>
</property>
<property>
  <name>yarn.nodemanager.resource.memory-mb</name>
  <value>6144</value>
</property>
<property>
  <name>yarn.nodemanager.resource.cpu-vcores</name>
  <value>6</value>
</property>
<property>
  <name>yarn.scheduler.minimum-allocation-mb</name>
  <value>2048</value>
</property>
</configuration>
```

## 9、创建 master 节点目录

以下在 master 节点用 root 执行

注意：多个数据目录时，mkdir -p /data2/hdfs/namenode 依次类推

```
mkdir -p $HADOOP_HOME/{tmp,logs}
```

```
mkdir -p /data1/hdfs/namenode
```

## 10、创建所有节点目录（包括 master）

以下在所有节点用 root 执行

注意：多个数据目录时：

```
mkdir -p /data2/hdfs/datanode 依次类推
```

```
sudo chown -R hdfs:hadoop /data2/hdfs 依次类推
```

```
mkdir -p /data1/hdfs/{tmp,datanode}
```

```
sudo chown -R hdfs:hadoop /data1/hdfs/
```

## 11、配置 workers

```
vi /linkcd/install/hadoop/etc/hadoop/workers
```

编辑文件内容如下：

注意：如果 master 的数据盘和 slave 一模一样（即 master 也保存数据），则把 master 也写入到以下内容中

```
slave1
```

```
slave2
```

```
slave3
```

## 12、配置 yarn-env.sh

```
vi /linkcd/install/hadoop/etc/hadoop/yarn-env.sh
```

在 16 行下面添加以下内容：

```
YARN_RESOURCEMANAGER_USER=root
```

```
HADOOP_SECURE_DN_USER=yarn
```

```
YARN_NODEMANAGER_USER=root
```

## 13、配置 start-dfs.sh

```
vi /linkcd/install/hadoop/sbin/start-dfs.sh
```

在 17 行下面添加以下内容：

```
HDFS_DATANODE_USER=root
```

```
HDFS_DATANODE_SECURE_USER=hdfs
```

```
HDFS_NAMENODE_USER=root
HDFS_SECONDARYNAMENODE_USER=root
```

#### 14、配置 stop-dfs.sh

```
vi /linkcd/install/hadoop/sbin/stop-dfs.sh
```

在最后一行下面添加以下内容：

```
HDFS_DATANODE_USER=root
HDFS_DATANODE_SECURE_USER=hdfs
HDFS_NAMENODE_USER=root
HDFS_SECONDARYNAMENODE_USER=root
```

#### 15、配置堆内存

```
vi /linkcd/install/hadoop/etc/hadoop/hadoop-env.sh
```

在 55 行下面添加以下内容：

```
export HDFS_NAMENODE_OPTS="-Xms2048m -Xmx2048m $HDFS_NAMENODE_OPTS"
export HDFS_DATANODE_OPTS="-Xms2048m -Xmx2048m $HDFS_DATANODE_OPTS"
export HDFS_SECONDARYNAMENODE_OPTS="-Xms2048m -Xmx2048m $HDFS_SECONDARYNAMENODE_OPTS"
export HADOOP_CLIENT_OPTS="-Xmx4096m $HADOOP_CLIENT_OPTS"
```

#### 16、同步到其他节点：

在 **slave** 节点的 **root** 用户下操作

注意：目录名中的版本号根据版本实际情况

```
sudo mkdir -p /linkcd/install/hadoop-3.2.1/
sudo ln -s /linkcd/install/hadoop-3.2.1 /linkcd/install/hadoop
sudo chown -R hdfs:hadoop /linkcd/install/hadoop/
```

在 **master** 节点的 **root** 用户下操作

```
sudo chown -R hdfs:hadoop /linkcd/install/hadoop/
sudo scp -r /linkcd/install/hadoop/* slave1:/linkcd/install/hadoop/
sudo scp -r /linkcd/install/hadoop/* slave2:/linkcd/install/hadoop/
sudo scp -r /linkcd/install/hadoop/* slave3:/linkcd/install/hadoop/
```

在 **slave** 节点的 **root** 用户下操作

```
source /etc/profile.d/hdfs.sh  
sudo chown -R hdfs:hadoop /linkcd/install/hadoop/
```

## 17、启动 hdfs

在 **master** 节点的 **root** 用户下操作

```
cd /linkcd/install/hadoop/bin  
./hdfs --workers --daemon start journalnode  
./hdfs namenode -format  
./hdfs datanode -format  
./hdfs zkfc -formatZK  
./hdfs --workers --daemon stop journalnode  
cd /linkcd/install/hadoop/sbin  
. /start-all.sh
```

在最后一行下面添加以下内容：

```
YARN_RESOURCEMANAGER_USER=root  
HADOOP_SECURE_DN_USER=yarn  
YARN_NODEMANAGER_USER=root
```

## 18、验证

如果启动异常，一定要仔细看/linkcd/install/hadoop/logs 中的日志

在 master 节点用 root 执行：

```
hdfs dfsadmin -report
```

通过以下地址访问查看 hdfs 的状态

http://master 的 ip:50070

# 8.5 部署 hbase 过程

## 1、下载 hbase 安装包

<http://192.168.90.117/software/bigdata/hbase-2.2.4-bin.tar.gz>

## 2、解压 tar 包

以下操作都在 **master** 节点的 **root** 用户下执行

```
sudo tar xf hbase-2.2.4-bin.tar.gz -C /linkcd/install
sudo ln -s /linkcd/install/hbase-2.2.4 /linkcd/install/hbase
sudo chown -R hbase:hadoop /linkcd/install/hbase/
```

### 3、配置环境变量

以下操作都在 **master** 节点的 **hbase** 用户下执行

```
tee -a ~/.bash_profile <<EOF
export HBASE_HOME=/linkcd/install/hbase
export PATH=$PATH:$HBASE_HOME/bin
EOF
source ~/.bash_profile
scp ~/.bash_profile slave1:~
scp ~/.bash_profile slave2:~
scp ~/.bash_profile slave3:~
```

### 4、配置/linkcd/install/hbase/conf/hbase-env.sh

以下操作都在 **master** 节点的 **hbase** 用户下执行

```
vim /linkcd/install/hbase/conf/hbase-env.sh
```

在 22 行下面添加以下内容

**注意：** **HBASE\_MASTER\_OPTS**， **HBASE\_REGIONSERVER\_OPTS=HBASE\_HEAPSIZE** 的堆内存是基于 16G 物理服务器的配置，当物理服务器内存较小时，相应调小对应的堆内存大小（比如 32G 时，分别设置为 4g，20g 和 20g）

```
export JAVA_HOME=/usr/java/default
export HBASE_PID_DIR=$HBASE_HOME/tmp
export HBASE_CLASSPATH=$HADOOP_HOME/etc/hadoop
export HBASE_MANAGES_ZK=false
export HBASE_MASTER_OPTS="-Xms2g -Xmx2g $HBASE_MASTER_OPTS"
export HBASE_REGIONSERVER_OPTS="-Xms8g -Xmx8g $HBASE_REGIONSERVER_OPTS"
export HBASE_HEAPSIZE=8G
```

## 5、配置/linkcd/install/hbase/conf/hbase-site.xml

以下操作都在 **master** 节点的 **hbase** 用户下执行

```
vi /linkcd/install/hbase/conf/hbase-site.xml
```

编辑文件内容如下：

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://master:9000/hbase</value>
  </property>
  <property>
    <name>hbase.unsafe.stream.capability.enforce</name>
    <value>false</value>
  </property>
  <property>
    <name>hbase.master.info.bindAddress</name>
    <value>0.0.0.0</value>
  </property>
  <property>
    <name>hbase.master.info.port</name>
    <value>16010</value>
  </property>
  <property>
    <name>hbase.master.namespace.init.timeout</name>
    <value>2400000</value>
  </property>
  <property>
    <name>hbase.master.port</name>
    <value>16000</value>
  </property>

```

```
</property>
<property>
  <name>hbase.master.ui.readonly</name>
  <value>false</value>
</property>
<property>
  <name>hbase.master.wait.on.regionserver.timeout</name>
  <value>30000</value>
</property>
<property>
  <name>hbase.regionserver.handler.count</name>
  <value>100</value>
</property>
<property>
  <name>hbase.regionserver.info.port</name>
  <value>16030</value>
</property>
<property>
  <name>hbase.regionserver.port</name>
  <value>16020</value>
</property>
<property>
  <name>hbase.regionserver.thread.compaction.small</name>
  <value>3</value>
</property>
<property>
  <name>zookeeper.recovery.retry</name>
  <value>6</value>
```



```
</property>
<property>
  <name>zookeeper.session.timeout</name>
  <value>90000</value>
</property>
<property>
  <name>zookeeper.znode.parent</name>
  <value>/hbase</value>
</property>
<property>
  <name>hbase.zookeeper.useMulti</name>
  <value>true</value>
</property>
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>slave1:2181,slave2:2181,slave3:2181</value>
</property>
<property>
  <name>hbase.tmp.dir</name>
  <value>/linkcd/install/hbase/tmp</value>
</property>
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
</configuration>
```

## 6、配置/linkcd/install/hbase/conf/regionserver

以下操作都在 **master** 节点的 **hbase** 用户下执行

```
vi /linkcd/install/hadoop/etc/hadoop/hadoop-env.sh
```

编辑文件内容如下：

注意：如果 **master** 的数据盘和 **slave** 一模一样（即 **master** 也保存数据），则把 **master** 也写入到以下内容中

```
slave1  
slave2  
slave3
```

## 7、同步到其他节点：

在 **slave** 节点的 **root** 用户下操作

注意：目录名中的版本号根据版本实际情况

```
mkdir -p /linkcd/install/hbase-2.2.4  
ln -s /linkcd/install/hbase-2.2.4 /linkcd/install/hbase  
chown hbase:hadoop /linkcd/install/hbase/
```

在 **master** 节点的 **hbase** 用户下操作

```
mkdir -p /linkcd/install/hbase/tmp  
rm -f /linkcd/install/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.25.jar  
scp -r /linkcd/install/hbase/* slave1:/linkcd/install/hbase/  
scp -r /linkcd/install/hbase/* slave2:/linkcd/install/hbase/  
scp -r /linkcd/install/hbase/* slave3:/linkcd/install/hbase/
```

在 **slave** 节点的 **root** 用户下操作

```
chown -R hbase:hadoop /linkcd/install/hbase/
```

## 8、启动 hdfs

在 **master** 节点的 **hbase** 用户下操作

```
cd /linkcd/install/hbase/bin/  
./start-hbase.sh
```

## 9、验证

如果启动异常，一定要仔细看/linkcd/install/hadoop/logs 中的日志

通过以下地址访问查看 hbase 的状态  
http://master 的 ip:16010

## 第 9 章 部署 xxl-job-admin

### 9.1 运行环境

标准运行环境中，Nacos，xxl-job-admin 和应用程序使用的 Mysql（非生产数据库）共用一台服务器

应用和版本	系统	CPU 和内存最低配置	数据磁盘	数量
Nacos 1.2 Xxl-job-admin 2.2	CentOS 7.7	4 核 8G	100G	1 台

### 9.2 准备工作

- 1、系统初始化并安装 jdk8
- 2、安装 Mysql 数据库
- 3、下载 jar 部署脚本和 xxl-job-admin 软件包（参考第一章地址）

### 9.3 安装 xxl-job-admin

- 1、执行 jar 部署脚本安装 xxl-job-admin

**注意：**输入 1 选择新项目，然后第二步输入 1 选择"./xxx.jar can start"

service\_name 使用"xxl-job-admin"

- 2、部署 xxl-job-admin 以后，cd 到 xxl-job-admin 安装目录的 sql 目录中，有一个 tables\_xxl\_job.sql 文件，需要自己登录到 Mysql 中，source 该文件，然后创建 xxl 用户

**注意：**tables\_xxl\_job.sql 文件的路径和 xxl 用户名密码可以根据项目实际情况修改

```
source /linkcd/jarapps/xxl-job-admin-2.2.0/sql/ tables_xxl_job.sql
create user xxl@'%' identified by 'xxljob123';
grant all privileges on `xxl_job`.* to xxl@'%';
flush privileges;
```

### 3、修改 xxl-job-admin 的配置文件

```
cd xxl-job-admin 安装目录中  
vim config/application.properties
```

根据项目实际情况修改以下内容并保存：

```
server.port=8080 # xxl-job-admin 服务的监听端口号  
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/xxl_job?useUnicode=true&characterEnc  
oding=UTF-8&autoReconnect=true&serverTimezone=Asia/Shanghai  
spring.datasource.username=xxl # 上一步在 mysql 中创建的 xxl 用户名  
spring.datasource.password=xxljob123 # 上一步在 mysql 中创建的 xxl 用户密码
```

### 4、启动 xxl-job-admin

```
cd xxl-job-admin 安装目录中  
./startup.sh
```

### 5、登录 xxl-job-admin 控制台

注意：IP:Port 根据项目实际情况（Port 是 xxl-job-admin 服务的监听端口号）

```
http://IP:Port/xxl-job-admin  
控制台初始账户：admin/123456
```

### 6、修改 admin 的密码

点击右上角"欢迎 admin"的下拉三角，选择"修改密码"  
输入新密码为：linkcld123456

### 7、用户管理

在使用调度中心时：添加不同的执行器，授权给不同的用户，不要用 admin 执行调度  
使用方法：登录 admin 用户，首先添加执行器，然后再添加用户同时选择对应的执行器，  
如下图所示：

## 新增用户

账号*	<input type="text" value="kettle"/>
密码*	<input type="password" value="linkcld123456"/>
角色*	<input checked="" type="radio"/> 普通用户 <input type="radio"/> 管理员
权限*	<input checked="" type="checkbox"/> carte(carte)

## 9.4 安装 xxl-job-executor（可选）

部署并启动 xxl-job-admin 以后，可以根据项目实际需求部署 xxl-job-executor

1、下载 jar 部署脚本和 xxl-job-executor 软件包（参考第一章地址）

2、执行 jar 脚本部署 xxl-job-executor

**注意：**输入 1 选择新项目，然后第二步输入 1 选择"./xxx.jar can start"

service\_name 使用"xxl-job-executor-端口号"

3、配置 xxl-job-executor

修改 xxl-job-executor 的配置文件

```
cd xxl-job-executor 安装目录中
vim config/application.properties
```

根据项目实际情况修改以下红色参数并保存：

```
server.port=8090 # xxl-job-executor 服务的 web 监听端口号
xxl.job.admin.addresses=http://127.0.0.1:8080/xxl-job-admin # xxl-job-admin 的地址
xxl.job.executor.appname=xxl-kettle # xxl.job.executor 执行器的标签，在 xxl-job-admin 中
注册执行器时需要用到，请保持一致
```

```
xxl.job.executor.ip=192.168.80.31 # xxl.job.executor 的 IP, 多网卡时必须配置!
xxl.job.executor.port=9999 # xxl.job.executor 的端口号, xxl-job-admin 通过该端口调用执行器
```

#### 4、启动 xxl-job-admin

```
cd xxl-job-executor 安装目录中
./startup.sh
```

#### 5、登录 xxl-job-admin 添加 xxl.job.executor

根据项目实际情况修改以下红色参数:

点击左侧的菜单栏: "执行器管理"

然后点击右上角"新增"按钮:

**AppName\***: **xxl-kettle** # 必须和第 3 步中配置的 xxl.job.executor.appname 保持一致

**名称\***: **xxl-kettle** # 执行器在 xxl-job-admin 控制台的显示名称。根据项目情况自定义

**注册方式\***: **手动录入**

**http:// IP:Port**

**注意**: 多个执行器用"逗号"分隔: 比如

http:// IP1:Port1, http:// IP2:Port2, http:// IP3:Port3

## 9.5 防火墙

参考附录

# 第 10 章 部署 kettle 集群

## 10.1 运行环境

应用和版本	系统	CPU 和内存最低配置	数据磁盘	数量
Kettle 9 master	Windows Server 2008R2	8 核 16G	100G	1 台
Kettle 9 slave	CentOS 7.7	8 核 16G	100G	2 台

## 10.2 准备工作

### 1、master 节点安装 jdk8 并配置 java 环境

下载 jdk8: <http://192.168.90.117/software/jdk/jdk-8u181-windows-x64.exe>

安装 jdk8，然后配置环境变量：

a).打开我的电脑--属性--高级--环境变量

b).新建系统变量 JAVA\_HOME 和 CLASSPATH

变量名：JAVA\_HOME

变量值：C:\Program Files\Java\jdk1.8.0\_181 [具体路径以自己本机安装目录为准]

变量名：CLASSPATH

变量值：.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar;

c). 选择“系统变量”中变量名为“Path”的环境变量，双击该变量，把 JDK 安装路径中 bin 目录的绝对路径，添加到 Path 变量的值中，并使用半角的分号和已有的路径进行分隔。

;%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;

2、slave 节点挂载数据盘到/linkcd 目录，然后执行系统初始化脚本（包括安装 jdk）

3、下载 kettle 9 安装包到每个节点

<http://192.168.90.117/software/etl/pdi-ce-9.0.0.0-423.zip>

4、master 在数据盘创建 kettle 安装目录

master 节点是 window server 2008R2 系统。kettle 的安装目录创建在数据盘。比如：

D:\linkcd\install\kettle9，以下安装过程都是基于数据盘是 D 盘的情况，请根据项目实际情况修改。

5、slave 节点使用部署脚本安装依赖包

依赖包部署脚本：<http://192.168.90.117/software/etl/webkitgtk-centos7.tar.gz>

## 6、slave 节点创建用户和 kettle 安装目录

```
sudo useradd -s /sbin/nologin kettle
sudo mkdir -p /linkcd/install/kettle9
sudo chown kettle:kettle /linkcd/install/kettle9
```

## 7、master 和 slave 节点关闭防火墙或者设置允许互相通信

# 10.3 部署 master 节点

1、解压 `pdi-ce-9.0.0.0-423.zip` 到 `D:\linkcd\install\kettle9` 目录中

2、`cd` 到 `D:\linkcd\install\kettle9\data-integration\pwd` 中

编辑 master 的配置文件：`carte-config-master.xml`，修改以下参数值：

IP 和端口号请根据项目实际情况修改

```
<slaveserver>
  <name>kettle-master</name>
  <hostname>192.168.80.33</hostname>
  <port>19090</port>
  <master>Y</master>
  <username>kettle</username>
  <password>linkcd123456</password>
</slaveserver>
```

## 3、配置 Spoon.bat

`cd D:\linkcd\install\kettle9\data-integration`

修改以下参数值，内存大小根据项目实际情况修改：以实例是 16G 内存情况

```
if "%PENTAHO_DI_JAVA_OPTIONS%"==" " set PENTAHO_DI_JAVA_OPTIONS="-Xms10240m"
"-Xmx10240m" "-Dfile.encoding=UTF-8"
```

## 4、master 注册 carte 为服务

1) 下载 winsw:

<http://192.168.90.117/software/window%E6%B3%A8%E5%86%8C%E6%9C%8D%E5%8A%A1/winsw2.6.2.zip>



2) 在 D:\linkcd\install\kettle9\data-integration 目录中创建 service 目录

3) 解压 winsw2.6.2.zip, 拷贝 WinSW.NET2.exe 到 D:\linkcd\install\kettle9\data-integration\service 目录中

4) 重命名 WinSW.NET2.exe 为 kettle.exe, 并新建 kettle.xml 文件, 编辑内容如下:

```
<service>
  <id>kettle</id>
  <name>kettle9 carte Service</name>
  <description>kettle9 carte Service</description>
  <logpath>D:\linkcd\install\kettle9\data-integration\service\logs</logpath>
  <log mode="roll-by-size">
    <sizeThreshold>8192</sizeThreshold>
    <keepFiles>3</keepFiles>
  </log>
  <env name="JAVA_HOME" value="C:\Program Files\Java\jdk1.8.0_181" />
  <workingdirectory>D:\linkcd\install\kettle9\data-integration</workingdirectory>
  <executable>D:\linkcd\install\kettle9\data-integration\Carte.bat</executable>
  <stopexecutable>D:\linkcd\install\kettle9\data-integration\Carte.bat</stopexecutable>
  <stoparguments> -s</stoparguments>
</service>
```

5) 编辑 D:\linkcd\install\kettle9\data-integration\Carte.bat 文件, 修改倒数第二行:

原内容: call Spoon.bat -main org.pentaho.di.www.Carte %\*

新内容: call Spoon.bat -main org.pentaho.di.www.Carte . /pwd/carte-config-master.xml %\*

6) 注册 kettle 服务:

```
cd D:\linkcd\install\kettle9\data-integration\service
```

管理员身份运行 cmd

执行: `./kettle.exe install`

7) 然后打开"控制面板\所有控制面板项\管理工具\服务", 找到 **kettle**, 确认启动类型为"自动", 修改 kettle 服务的登录用户为"administrator", 如下图



8) 启动 kettle 服务

## 10.4 部署 slave 节点

以下操作在 **root** 或 **sudo** 用户下完成

1、两个 slave 节点解压 **pdi-ce-9.0.0.0-423.zip** 到 **/linkcd/install/kettle9** 目录中

```
unzip pdi-ce-9.0.0.0-423.zip -d /linkcd/install/kettle/
```

```
cd /linkcd/install/kettle9/data-integration/pwd
```

## 2、编辑 **slave1** 的配置文件：carte-config-slave1.xml，修改以下参数值：

IP 和端口号请根据项目实际情况修改

```
<masters>

  <slaveserver>

    <name>kettle-master</name>

    <hostname>192.168.80.33</hostname>

    <port>19090</port>

    <username>kettle</username>

    <password>linkcld123456</password>

    <master>Y</master>

  </slaveserver>

</masters>

<report_to_masters>Y</report_to_masters>

<slaveserver>

  <name>kettle-slave1</name>

  <hostname>192.168.80.34</hostname>

  <port>19090</port>

  <username>kettle</username>

  <password>linkcld123456</password>

  <master>N</master>

</slaveserver>
```

## 3、编辑 **slave2** 的配置文件：carte-config-slave1.xml，修改以下参数值：

IP 和端口号请根据项目实际情况修改

```
<masters>
```

```
<slaveserver>  
  <name>kettle-master</name>  
  <hostname>192.168.80.33</hostname>  
  <port>19090</port>  
  <username>kettle</username>  
  <password>linkcld123456</password>  
  <master>Y</master>  
</slaveserver>
```

```
</masters>
```

```
<report_to_masters>Y</report_to_masters>
```

```
<slaveserver>  
  <name>kettle-slave2</name>  
  <hostname>192.168.80.35</hostname>  
  <port>19090</port>  
  <username>kettle</username>  
  <password>linkcld123456</password>  
  <master>N</master>  
</slaveserver>
```

#### 4、两个 slave 节点配置 spoon.sh

```
cd D:\linkcld\install\kettle9\data-integration  
vim spoon.sh
```

修改以下参数值，内存大小根据项目实际情况修改

```
if [ -z "$PENTAHO_DI_JAVA_OPTIONS" ]; then  
  PENTAHO_DI_JAVA_OPTIONS="-Xms12g -Xmx12g -Dfile.encoding=UTF-8"
```

fi

## 5、配置 **slave1** 节点的服务文件

**vim /usr/lib/systemd/system/kettle.service**

**slave1** 的服务文件内容如下：

```
# Systemd unit file for kettle carte slave1

[Unit]

Description=kettle carte slave1 service

After=syslog.target network.target


[Service]

Environment=JAVA_HOME=/usr/java/default

ExecStartPre=/usr/bin/cd /linkcd/install/kettle9/data-integration

ExecStart=/usr/bin/nohup /bin/bash /linkcd/install/kettle9/data-integration/carte.sh /linkcd/install/kettle9/data-integration/pwd/carte-config-slave1.xml

ExecStop=/usr/bin/nohup /bin/bash /linkcd/install/kettle9/data-integration/carte.sh -s /linkcd/install/kettle9/data-integration/pwd/carte-config-slave1.xml


User=kettle

Group=kettle

UMask=0007

Restart=always

RestartSec=20


[Install]

WantedBy=multi-user.target
```

## 6、配置 **slave2** 节点的服务文件

**slave2** 的服务文件内容如下：

```
# Systemd unit file for kettle carte slave2

[Unit]

Description=kettle carte slave2 service
```

```
After=syslog.target network.target

[Service]

Environment=JAVA_HOME=/usr/java/default

ExecStartPre=/usr/bin/cd /linkcd/install/kettle9/data-integration

ExecStart=/usr/bin/nohup /bin/bash /linkcd/install/kettle9/data-integration/carte.sh /linkcd/install/kettle9/data-integration/pwd/carte-config-slave2.xml

ExecStop=/usr/bin/nohup /bin/bash /linkcd/install/kettle9/data-integration/carte.sh -s /linkcd/install/kettle9/data-integration/pwd/carte-config-slave2.xml

User=kettle

Group=kettle

UMask=0007

Restart=always

RestartSec=20

[Install]

WantedBy=multi-user.target
```

## 7、启动两个 slave 节点的服务

```
sudo chown -R kettle:kettle /linkcd/install/kettle9

sudo systemctl daemon-reload

sudo systemctl start kettle

sudo systemctl enable kettle
```

## 10.5 配置资源库并创建 carte 集群规范

### 1、通过浏览器访问可以检查节点状态：

控制台账户：kettle/linkcd123456

master 节点：http://192.168.80.33:19090

slave1 节点：http://192.168.80.34:19090

slave2 节点：http://192.168.80.35:19090

## 2、在 master 节点启动 spoon

```
cd D:\linkcld\install\kettle9\data-integration
```

双击 Spoon.bat

## 3、配置资源库

**配置资源库：**数据库的 APP\_KETTLE 库中

1、需要一个 oracle 或 mysql 数据库（自行安装）

2、在 mysql 或 oracle 数据库中创建一个 **app\_kettle** 库和 **app\_kettle** 用户

**mysql 语句：**

**注意：** mysql 全小写

```
create database `app_kettle`;
```

```
create user 'app_kettle'@'%' identified by 'app_kettle';
```

```
grant all privileges on `app_kettle`.* to 'app_kettle'@'%';
```

```
flush privileges;
```

**oracle 语句：**

**注意：** oracle 全大小，根据项目实际情况替换数据文件的路径

```
CREATE TABLESPACE TBS_APP_KETTLE
```

```
DATAFILE
```

```
    '/home/oracle/app/oradata/orcl/TBS_APP_KETTLE01.DBF' SIZE 1G AUTOEXTEND ON
```

```
NEXT 32M MAXSIZE 8G
```

```
EXTENT MANAGEMENT LOCAL
```

```
SEGMENT SPACE MANAGEMENT AUTO;
```

```
COMMIT;
```

```
CREATE USER APP_KETTLE IDENTIFIED BY APP_KETTLE
```

```
    DEFAULT TABLESPACE TBS_APP_KETTLE
```

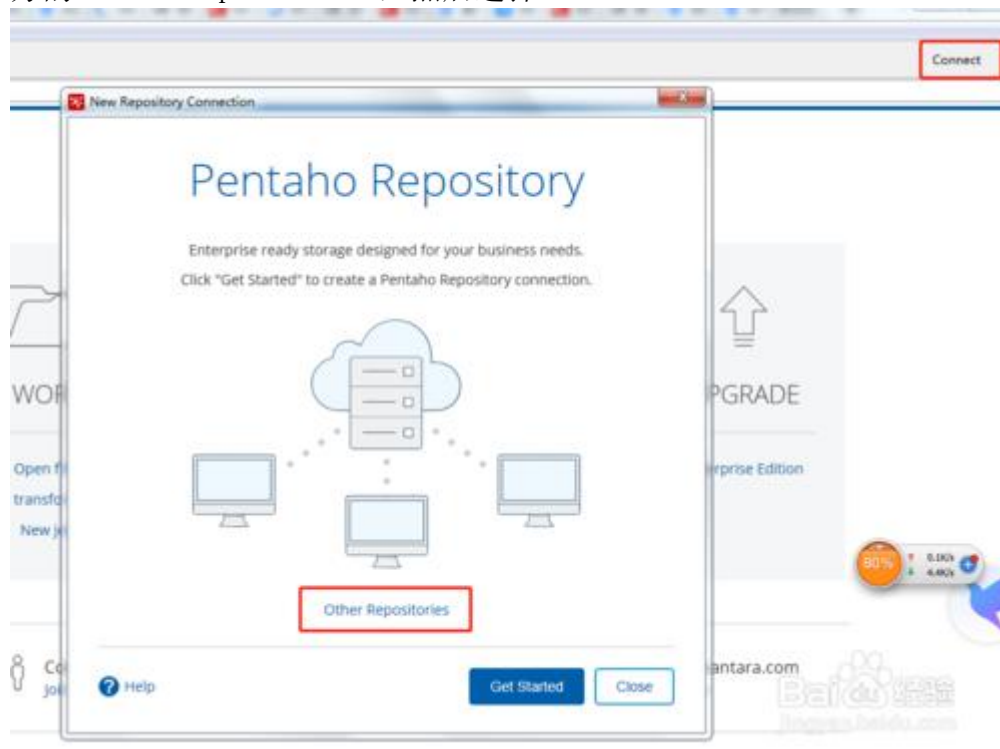
QUOTA UNLIMITED ON TBS\_APP\_KETTLE

TEMPORARY TABLESPACE TBS\_TEMP\_GROUP;

GRANT CONNECT, RESOURCE TO APP\_KETTLE;

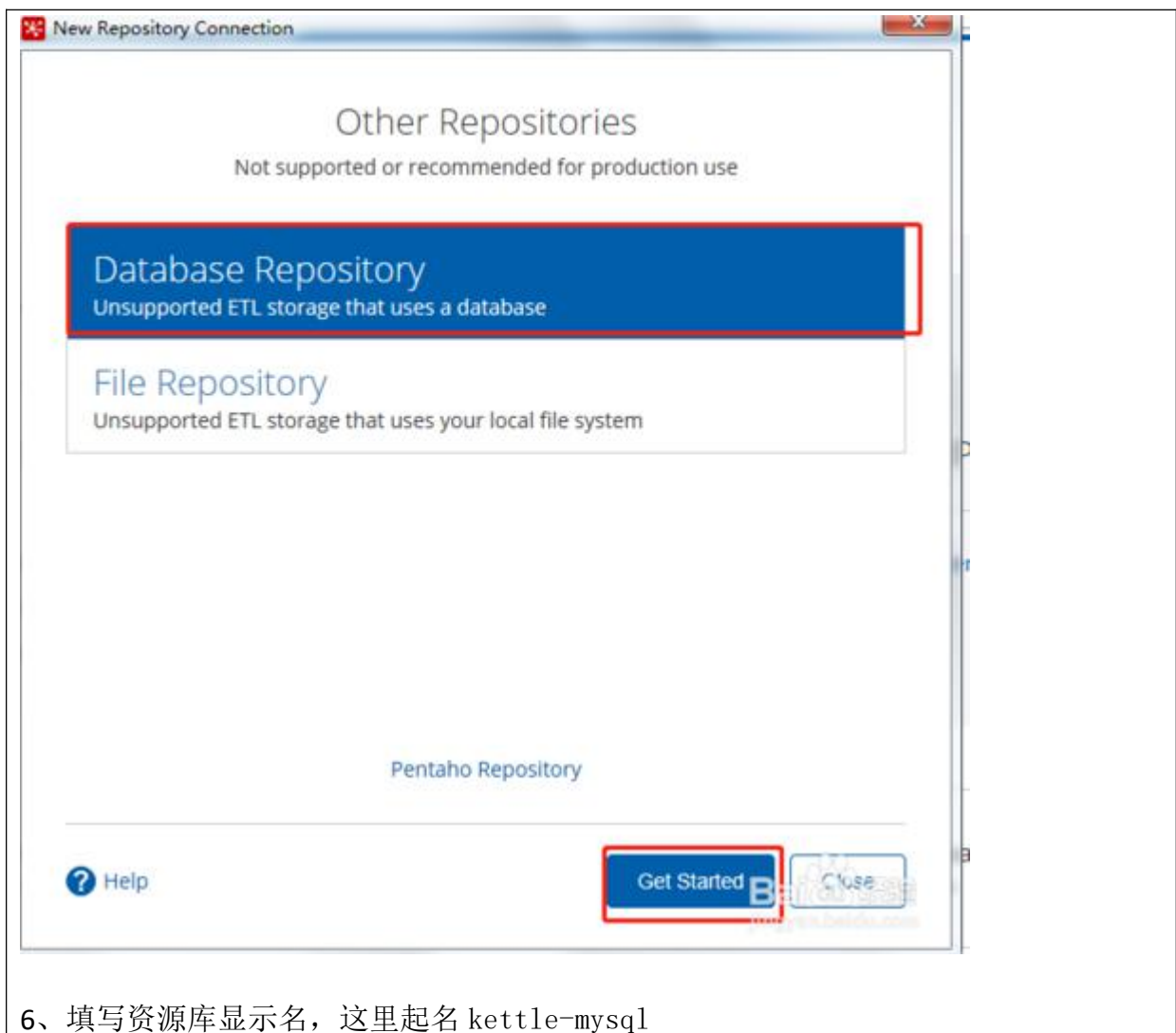
COMMIT;

4、打开 kettle 工具，开始配置资源库，点击右上角 connect，弹出如图界面，点击下方的 Other Repositories，然后选择



5、选择 Database Repositories，点击 Get Started





6、填写资源库显示名，这里起名 kettle-mysql

## Connection Details

Display Name

kettle-mysql

Database Connection

kettle-mysql >

Description

Database repository

☒ Launch connection on startup

Back

Finish

7、点击 database connection 框，在弹出的界面中点击 Create New Connection ，弹出数据库连接界面。填写连接名称（自己定义）、选择连接类型 mysql 、选择连接方式 JDBC、填写上面配置的资源库主机名称（一般为 ip, 根据项目实际情况填写）、填写资源库（app\_kettle）、填写端口号（数据库端口号）、填写用户名（app\_kettle）、填写密码（app\_kettle），点击测试，成功后点击确定。

连接名称:  
kettle-mysql

连接类型:  
MySQL  
Native Mondrian  
Neoview  
Netezza  
Oracle  
Oracle RDB  
Palo MOLAP Server  
Pentaho Data Services  
PostgreSQL  
Redshift  
Remedy Action Request System  
SAP ERP System  
SQLite  
Snowflake  
SparkSQL  
Sybase  
SybaseIQ  
Teradata

连接方式:  
Native (JDBC)  
ODBC  
JNDI

设置  
主机名称:  
192.168.80.189  
数据库名称:  
app\_kettle  
端口号:  
3306  
用户名:  
app\_kettle  
密码:  
●●●●●●●●●●  
☒ Use Result Streaming Cursor

8、经过上一步骤数据库连接已经创建好，选中之后点击 Back，点击 Finish 完成，等待创建完成，完成后点击 Finish。

9、登录 mysql 数据库检查资源库 **app\_kettle** 库中表是否创建成功

```
mysql> use `app_kettle`;
```

```
mysql> show tables;
```

显示 46 张表则表示成功

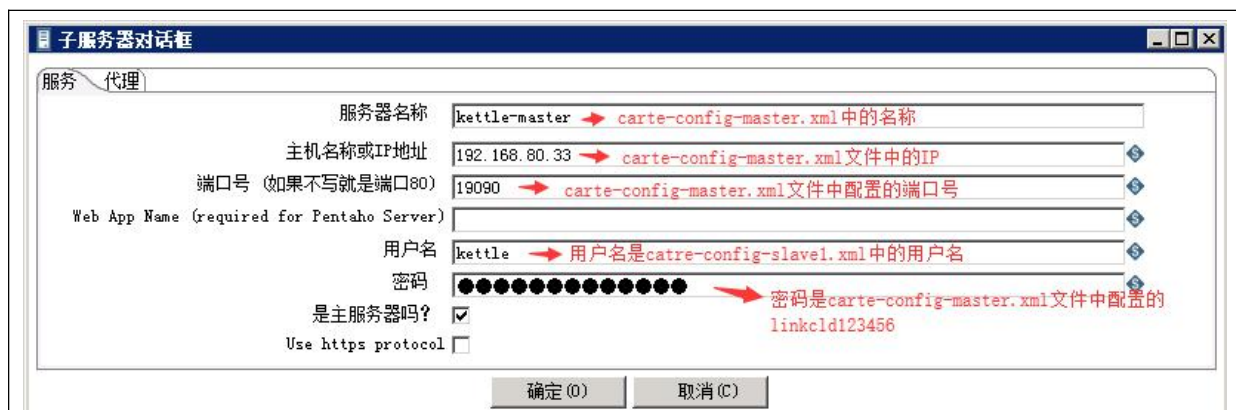
10、登录到资源库

点击左上角的"connect"，再点击"kettle-mysql"，输入用户名密码：admin/admin

#### 4、创建 carte 集群

1、点击左上角的"主对象树"，双击"转换"，再点击左上角的"主对象树"，双击"子服务器"，分 3 次创建 master、slave1 和 slave2。操作如下：

2、master 节点信息参考 D:\linkcd\install\data-integration\pwd\carte-config-master.xml



### 3、slave1 节点信息参考:

/linkcld/install/kettle9/data-integration/pwd/karte-config-slave1.xml

注意: 是主服务器吗? 不要勾选

### 4、slave2 节点信息参考:

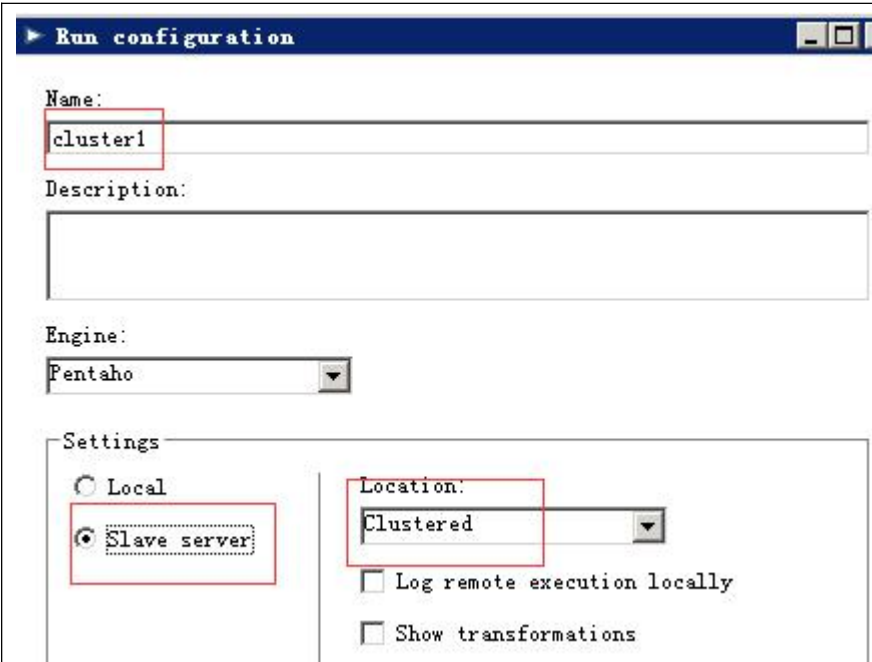
/linkcld/install/kettle9/data-integration/pwd/karte-config-slave2.xml

注意: 是主服务器吗? 不要勾选

### 5、双击" kettle 集群 schemas", 选择子服务器时, 按住 ctrl 可以同时选择多个



### 6、右击"Run configurations", 然后点击"New", 配置信息如下图:



Run configuration

Name:  
cluster1

Description:

Engine:  
Pentaho

Settings

☐ Local  
☒ Slave server

Location:  
Clustered

☐ Log remote execution locally  
☐ Show transformations

7、然后保存配置到配置库

## 10.6 创建 job 规范

单个 job 建议只配置单个转换，否则一旦 job 执行失败，监控无法精确定位到其中哪个转换发生了异常，只能通过查看日志确认。

## 10.7 配置 job 日志规范

1、点击左上角的"主对象树"，右击你创建的 job，点击"设置"，点击"日志"，点击"转换"。

日志保存到资源库 app\_kettle 库的 e\_job\_log 表中

日志数据库连接地址根据项目实际情况配置，库名和表名必须严格遵守规范



作业 命名参数 设置 日志

作业日志表  
作业环境日志表  
日志通道日志表

日志数据库连接: oracle178

日志模式

日志表 e\_job\_log

日志记录时间间隔 (秒) 1

日志行超时时间 (天) 1

在内存中保存的日志行数 1

## 2、配置完成后保存 job 到资源库中

# 10.8 配置 xxl-job 定期执行 job 实例

作用：配置 xxl-job-admin 调度中心定期执行 kettle 的 job，并通过"调度日志"监控执行状态

1、参考"第九章 部署 xxl-job-admin"部署 xxl-job-admin，以及在两个 slave 节点上部署执行器"carte"，并设置执行器以"自动注册"方式添加到 xxl-job-admin。



2、配置转换的表输入的数据源和表输出的保存库，然后右击"表输出"，选择"集群"--选择上一步创建的"cluster1"，保存转换为"trans1"。

3、配置 job，创建"start"，"转换"，"成功"。配置转换，选择上一步创建的"trans1"，单个 job 建议只配置单个转换，否则一旦 job 执行失败，监控无法精确定位到其中哪个转换发生了异常，只能通过查看日志确认。Run configuration：选择创建 carte 集群时配置的"cluster1"。保存 job 为"job1"

转换

作业项名称:

job1

Transformation:

/trans1

浏览...

Options

设置日志

Arguments

命名参数

Run configuration:

cluster1

Execution

☐ 执行每一个输入行?

☐ 在执行前清除结果行列表?

☐ 在执行前清除结果文件列表?

☒ 等待远程转换执行结束

☐ 本地转换终止时远程转换也通知终止

4、"任务管理", 选择"新增", 选择在两个 slave 节点创建的执行器"carte", 运行模式使用下图配置:

执行器*	carte	任务描述*	carte
路由策略*	轮询	Cron*	0/10 * * * * ?
运行模式*	GLUE(Shell)	JobHandler*	请输入JobHandler
阻塞处理策略*	单机串行	子任务ID*	请输入子任务的任务ID,如存在多个则逗
任务超时时间*	0	失败重试次数*	0
负责人*	xxx	报警邮件*	请输入报警邮件, 多个邮件地址则逗号
任务参数*	请输入任务参数		

5、编辑 GLUE IDE, 修改并保存, 内容如下:

红色部分地址为 **carte master** 节点的地址, 其他参数根据 **kettle** 的实际资源库和 **job** 参数调整。

```
#!/bin/bash
```

```
curl -u 'kettle:linkcd123456' "http://192.168.80.33:19090/kettle/executeJob/?rep=kettle-mysql&user=admin&pass=admin&job=/job/job1&level=INFO"
```

6、点击"操作", 选择"启动"执行任务。点击左侧菜单"调度日志"查看执行情况

## 10.9 防火墙

参考附录

# 第 11 章 附录

## 11.1 防火墙配置实例

考虑到生产中各个服务器的防火墙设置不统一, 或开启, 或关闭; 当防火墙开启时, 需要自己手动配置防火墙放行服务端口。

放行命令: 举个例子, tomcat 服务端口是 28080 时:

备注: 手动配置防火墙时, 请把 28080 替换成自己部署的服务端口

### 1、centos7

备注: 如果 centos 系统没有运行 firewalld, 而是使用 iptables 防火墙, 请参考 centos6

#### 1.1) 放行端口

```
sudo firewall-cmd --permanent --add-port=28080/tcp
```

```
sudo firewall-cmd --reload
```

#### 1.2) IP 白名单 (比如 192.168.80.183)

```
sudo firewall-cmd --permanent --add-source=192.168.80.183 --zone=trusted
```

```
sudo firewall-cmd --reload
```

#### 1.3) 网段白名单 (比如 192.168.80.0/24)

```
sudo firewall-cmd --permanent --add-source=192.168.80.0/24 --zone=trusted
```

```
sudo firewall-cmd --reload
```

### 2、centos6



### 2.1) 放行端口

```
sudo iptables -I INPUT -p tcp --dport 28080 -j ACCEPT
sudo iptables -I OUTPUT -p tcp --sport 28080 -j ACCEPT
sudo service iptables save
```

### 2.1) IP 白名单（比如 192.168.80.183）

```
sudo iptables -I INPUT -s 192.168.80.183 -j ACCEPT
sudo iptables -I OUTPUT -d 192.168.80.183 -j ACCEPT
sudo service iptables save
```

### 2.3) 网段白名单（比如 192.168.80.0/24）

```
sudo iptables -I INPUT -s 192.168.80.180/24 -j ACCEPT
sudo iptables -I OUTPUT -d 192.168.80.180/24 -j ACCEPT
sudo service iptables save
```

## 11.2 创建 swap 分区并挂载

云服务器通常没有 swap 分区，此时需要自己创建。情况分为两种：

- 1) 如果服务器有空闲磁盘，可以直接使用磁盘分区创建 swap。
- 2) 服务器所有磁盘空间都已经创建分区挂载了。此时没有磁盘可以用于创建 swap，只能使用文件来创建 swap。

### 11.2.1 使用磁盘

以下例子仅供参考：以 root 身份，磁盘是/dev/sdb。实际操作请根据服务器环境。

- 1) 创建分区

```
[root@node188 ~]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x3aabf49f.

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p):
Using default response p
Partition number (1-4, default 1):
First sector (2048-104857599, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-104857599, default 104857599): +2G
Partition 1 of type Linux and of size 2 GiB is set

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): L

   0  Empty           24  NEC DOS               81  Minix / old Lin  bf  Solaris
   1  FAT12           27  Hidden NTFS Win  82  Linux swap / So c1  DRDOS/sec (FAT-
   2  XENIX root     39  Plan 9           83  Linux             c4  DRDOS/sec (FAT-

Hex code (type L to list all codes): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

## 2) 查看分区:

```
[root@node188 ~]# fdisk -l /dev/sdb

Disk /dev/sdb: 53.7 GB, 53687091200 bytes, 104857600 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x3aabf49f
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		2048	4196351	2097152	82	Linux swap / Solaris

## 3) 格式化 swap 分区:

```
[root@node188 ~]# mkswap /dev/sdb1
Setting up swapspace version 1, size = 2097148 KiB
no label, UUID=dae469f1-19a9-488a-9529-bf714d8fd4a1
```

4) 挂载: 编辑/etc/fstab 添加一行

```
vim /etc/fstab
```

```
/dev/sdb1          swap swap          defaults          0 0
```

5) 激活 swap:

```
[root@localhost ~]# swapon /dev/sdb1
[root@localhost ~]# free -m
```

	total	used	free	shared	buff/cache	available
Mem:	7822	134	7332	8	356	7430
Swap:	2047	0	2047			

## 11.2.2 使用文件

注意事项: 如果/目录空间足够大, 文件尽量建立在/目录中。避免建立在多层目录中, 以免开机无法自动加载。以下实例在/目录生成文件, 实际操作请根据服务器环境

1) 生成一个 2G 的文件 (所以 swap 分区大小也是 2G):

```
sudo dd if=/dev/zero of=/swapfile count=4194304
```

2) `sudo mkswap /swapfile`

3) `sudo chmod 0600 /swapfile`

4) `sudo swapon /swapfile`

5) 编辑/etc/fstab 文件, 添加一行内容:

```
sudo vim /etc/fstab
```

```
/swapfile swap swap defaults 0 0
```

## 11.3 磁盘分区并挂载

新服务器除了系统盘已挂载使用, 其他磁盘通常没有分区挂载。

1) 如果服务器新加磁盘, 需要使用以下两个命令对比分析:

1.1) `lsblk` //查看有几块磁盘, 以及磁盘分区情况

注意: 有些新添加的磁盘, 需要重启服务器才会显示

磁盘名称通常是: `sd*` (物理磁盘), `vd*` (虚拟磁盘), `hd*` 三种格式。如下:

```
[deploy@localhost ~]$ lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                  8:0    0   20G  0 disk
├─sda1               8:1    0   500M  0 part /boot
├─sda2               8:2    0  19.5G  0 part
│   └─cl-root        253:0    0  17.5G  0 lvm  /
│       └─cl-swap     253:1    0    2G    0 lvm  [SWAP]
sdb                  8:16    0   20G  0 disk

[deploy@iZwz93g36wzfbgu4n8yc0mZ ~]$ lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                 253:0    0   40G  0 disk
├─vda1              253:1    0   40G  0 part /
vdb                 253:16    0  200G  0 disk
```

- 1.2) `sudo fdisk -l /dev/sdb` //查看磁盘分区情况，结果仅供参考，查看结果没有分区时，并不代表 100%就没有分区，**要结合 1.1,1.2,1.3,1.4 四步的查询结果进行综合判断**。大于 2TB 的磁盘要把 `fdisk` 命令换成 `gdisk` 命令。

1.2.1) 比如：可以看见 `/dev/sda` 磁盘明显已经有分区了，1.1 中也可以看出来

```
[deploy@localhost ~]$ sudo fdisk -l /dev/sda
```

```
Disk /dev/sda: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000b5673
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	2048	1026047	512000	83	Linux
/dev/sda2		1026048	41943039	20458496	8e	Linux LVM

1.2.2) 如下看不出 `/dev/sdb` 有分区，还要继续 1.3 和 1.4 步骤

```
[deploy@localhost ~]$ sudo fdisk -l /dev/sdb
```

```
Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
[deploy@localhost ~]$ █
```

- 1.3) 查看分区格式化情况，命令如下：



```
sudo blkid
```

实例：如下可以看出/dev/sda 和 sdb 都已经格式化了，这种情况已经说明 sdb 已经在用了。没有如果发现 sdb 的信息，还要继续 1.4 步骤

```
[deploy@localhost ~]$ sudo blkid
/dev/sda1: UUID="44a98461-a94a-4bd8-996f-80471e17292b" TYPE="xfs"
/dev/sda2: UUID="U3N4Fy-mYoP-QUtv-A1cX-4S0B-37b3-1DILLm" TYPE="LVM2_member"
/dev/mapper/cl-root: UUID="893d1815-c6f2-499d-93f5-9b3fdf7e3245" TYPE="xfs"
/dev/mapper/cl-swap: UUID="5c32dd85-4925-4b5d-9f0f-2f091ad6d33b" TYPE="swap"
/dev/sdb: UUID="c5c44fd7-861c-4da3-a4c3-20dbb1c4cfdc" TYPE="xfs"
```

1.4) 查看磁盘分区和挂载情况，和 1.1) 中的结果对比，查看哪个分区没有挂载：

```
sudo df -h
```

实例：可以看见/dev/sda1 分区挂载到/boot 目录了。并没有发现 sdb 挂载

```
[deploy@localhost ~]$ sudo df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/cl-root 18G  1.9G   16G  11% /
devtmpfs        1.9G     0   1.9G   0% /dev
tmpfs           1.9G     0   1.9G   0% /dev/shm
tmpfs           1.9G  8.6M   1.9G   1% /run
tmpfs           1.9G     0   1.9G   0% /sys/fs/cgroup
/dev/sda1       497M  132M  366M  27% /boot
tmpfs           380M     0   380M   0% /run/user/1000
```

注意：结合 1.1,1.2,1.3,1.4 步骤可以判断一块磁盘是否已经分区或格式化

2) 新磁盘成功显示后，下一步开始分区，首先看看分区命令：

2.1) 小于 2TB 的磁盘使用 fdisk 命令（系统自带该命令）；

2.2) 大于 2TB 的磁盘使用 gdisk（大部分情况都需要手动安装）：

```
sudo yum -y install gdisk
```

2.3) fdisk 和 gdisk 使用方法基本相同，以下操作以 fdisk 命令举例

3) 分区命令参数：

fdisk 或者 gdisk 命令使用方法：

一、在 console 上输入 fdisk -l /dev/sda，观察硬盘之实体使用情形。

二、在 console 上输入 fdisk /dev/sda，可进入磁盘分区模式。

1. 输入 m 显示所有命令列示。

2. 输入 p 显示硬盘分区情形。

3. 输入 n 设定新的硬盘分区。

- 3.1. 输入 **e** 硬盘为[扩展]分区(extend)。
- 3.2. 输入 **p** 硬盘为[主要]分区(primary)。
4. 输入 **t** 改变硬盘分割区属性。(制作交换分区时会用到)
5. 输入 **d** 删除硬盘分割区属性 //慎重，单个分区时会直接删除；多分区时需要指定分区编号
6. 输入 **q** 结束不保存
7. 输入 **w** 结束并保存

#### 4) 分区操作:

```
[deploy@localhost ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda          8:0    0   20G  0 disk
├─sda1       8:1    0   500M  0 part /boot
├─sda2       8:2    0  19.5G  0 part
│   └─cl-root 253:0    0  17.5G  0 lvm  /
│       └─cl-swap 253:1    0    2G  0 lvm  [SWAP]
└─sdb        8:16   0   20G  0 disk
```

在分区之前，应该先查看该磁盘的分区情况，参考 1.2)；或者进入 fdisk 分区模式查看：

```
[deploy@localhost ~]$ sudo fdisk /dev/sda 进入磁盘分区模式
Welcome to fdisk (util-linux 2.23.2).
```

Changes will remain in memory only, until you decide to write them.  
Be careful before using the write command.

Command (m for help): **p** 查看磁盘当前的分区情况

```
Disk /dev/sda: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000b5673
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	2048	1026047	512000	83	Linux
/dev/sda2		1026048	41943039	20458496	8e	Linux LVM

##### 4.1) 整个磁盘只做一个分区:

```
[deploy@localhost ~]$ sudo fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).
```

Changes will remain in memory only, until you decide to write them.  
Be careful before using the write command.

Command (m for help): **n** 新建分区

Partition type:

p primary (0 primary, 0 extended, 4 free)  
e extended

Select (default p): **p** 分区格式为主分区 (分区数量小于4个时, 都选择这个)

Partition number (1-4, default 1): **1**

First sector (2048-41943039, default 2048): **回车键**

Using default value 2048

Last sector, +sectors or +size{K,M,G} (2048-41943039, default 41943039): **回车**

Using default value 41943039

Partition 1 of type Linux and of size 20 GiB is set

Command (m for help): **w** 保存

The partition table has been altered!

Calling ioctl() to re-read partition table.  
Syncing disks.

#### 4.2) 整个磁盘做多个分区, 指定分区大小:

```
[deploy@localhost ~]$ sudo fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).
```

Changes will remain in memory only, until you decide to write them.  
Be careful before using the write command.

Command (m for help): **n**

Partition type:

p primary (0 primary, 0 extended, 4 free)  
e extended

Select (default p): **p**

Partition number (1-4, default 1): **1**

First sector (2048-41943039, default 2048): 这一步任何时候都直接回车即可

Using default value 2048

Last sector, +sectors or +size{K,M,G} (2048-41943039, default 41943039): **+10G**

Partition 1 of type Linux and of size 10 GiB is set

新建的分区大小共10G

Command (m for help): **w**

The partition table has been altered!

Calling ioctl() to re-read partition table.  
Syncing disks.

#### 5) 格式化分区:

**注意事项 1:** 分区越大, 格式化越慢, 格式化过程中千万不要中断或取消, 一定要



等命令执行完毕

**注意事项 2：**如果执行格式化时，出现如下警告，则说明该磁盘曾经分区或者格式化，这时候要慎重，联系管理人员询问该磁盘是否有数据

```
[deploy@localhost ~]$ sudo mkfs.xfs /dev/sdb
mkfs.xfs: /dev/sdb appears to contain an existing filesystem (xfs).
mkfs.xfs: Use the -f option to force overwrite.
```

实例：继续上面的步骤，此时 lsblk 可以查看到刚才新建的分区

```
[deploy@localhost ~]$ lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                  8:0    0   20G  0 disk
├─sda1               8:1    0   500M  0 part /boot
└─sda2               8:2    0  19.5G  0 part
   ├─cl-root         253:0    0  17.5G  0 lvm  /
   └─cl-swap         253:1    0    2G    0 lvm  [SWAP]
sdb                  8:16    0   20G  0 disk
└─sdb1               8:17    0   10G  0 part
sr0                  11:0    1 1024M  0 rom
```

### 5.1) centos7:

```
[deploy@localhost ~]$ sudo mkfs.xfs /dev/sdb1
meta-data=/dev/sdb1             isize=512    agcount=4, agsize=655360 blks
      =                       sectsz=512    attr=2, projid32bit=1
      =                       crc=1        finobt=0, sparse=0
data      =                       bsize=4096   blocks=2621440, imaxpct=25
      =                       sunit=0      swidth=0 blks
naming    =version 2           bsize=4096   ascii-ci=0 ftype=1
log       =internal log        bsize=4096   blocks=2560, version=2
      =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none                extsz=4096   blocks=0, rtextents=0
```

### 5.2) centos6:



```
[deploy@localhost ~]$ sudo mkfs.ext4 /dev/sdb1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
655360 inodes, 2621440 blocks
131072 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2151677952
80 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

## 6) 挂载分区配置:

### 6.1) 查看新建分区的 uuid

```
[deploy@localhost ~]$ sudo blkid
/dev/sda1: UUID="44a98461-a94a-4bd8-996f-80471e17292b" TYPE="xfs"
/dev/sda2: UUID="U3N4Fy-mYoP-QUtv-A1cX-4S0B-37b3-1DILLm" TYPE="LVM2_member"
/dev/mapper/cl-root: UUID="893d1815-c6f2-499d-93f5-9b3fdf7e3245" TYPE="xfs"
/dev/mapper/cl-swap: UUID="5c32dd85-4925-4b5d-9f0f-2f091ad6d33b" TYPE="swap"
/dev/sdb1: UUID="1e874869-85c9-4248-bc54-e765ce8acdc0" TYPE="xfs"
```

### 6.2) 使用 uuid 进行挂载

**注意事项:** `/etc/fstab` 不允许有任何错误, 否则系统无法启动

#### 6.2.1) 打开配置文件: `sudo vim /etc/fstab`

#### 6.2.2) 编辑内容并保存

格式:

UUID	挂载目录	分区格式	defaults	0 0
------	------	------	----------	-----

实例:

/dev/mapper/cl-root	/	xfs	defaults	0 0
UUID=44a98461-a94a-4bd8-996f-80471e17292b	/boot		xfs	defaults 0 0
/dev/mapper/cl-swap	swap	swap	defaults	0 0
UUID="1e874869-85c9-4248-bc54-e765ce8acdc0"	/linkcd	xfs	defaults	0 0

UUID 后面的双引号加不加都可以

### 6.3) 挂载分区

6.3.1) 挂载目录必须提前创建: `sudo mkdir /linkcld`

6.3.2) 挂载: `sudo mount /dev/sdb1`

如下图所示, 表示挂载成功:

```
[deploy@localhost ~]$ sudo df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/mapper/cl-root	18G	1.9G	16G	11%	/
devtmpfs	1.9G	0	1.9G	0%	/dev
tmpfs	1.9G	0	1.9G	0%	/dev/shm
tmpfs	1.9G	8.6M	1.9G	1%	/run
tmpfs	1.9G	0	1.9G	0%	/sys/fs/cgroup
/dev/sda1	497M	132M	366M	27%	/boot
tmpfs	380M	0	380M	0%	/run/user/0
/dev/sdb1	10G	33M	10G	1%	/linkcld

## 11.4 搭建基础源服务器

1) 以省或市为单元, 每个单元首先搭建一台 yum 源服务器和一台时钟服务器(可以部署在同一台服务器, 该服务器最好能连接公网)。分别使用 `yum_server` 和 `ntp_server` 部署脚本自动化部署。

**目的: 为局域网内无法连接公网的服务器提供 yum 源和时间同步服务。**

举例: 浙江省局的 yum 源服务器: 10.100.32.77; 时间服务器: 10.100.32.41

2) 能连接公网的服务器可以使用默认的公网 yum 源和阿里云的时间服务器 (ntp1.aliyun.com 和 time1.aliyun.com)。

## 11.5 部署目录规范

程序部署在非系统盘, 日志, 文件等存放路径设置在非系统盘。

目录结构规范参照:

/linkcld

--bak 存放备份文件

--exec 存放自定义脚本。比如自己写的计划任务脚本等

--jarapps 部署 jar 项目

--software 程序包临时存放目录  
--tomcat8 tomcat 应用存放目录  
--install 其他程序存放目录  
--www 存放 web 服务的静态数据

## 11.6 文件上传下载规范

### 11.6.1 上传

安装包和部署包放在/linkcd/software 目录中，不要堆放在 deploy 家目录。

- 1) 通常先把文件上传到 deploy 的家目录: /home/deploy
- 2) `sudo mv 文件 目的路径`

### 11.6.2 下载

- 1) 通常先把文件拷贝到 deploy 的家目录: /home/deploy  
`sudo cp -rf 文件或目录 /home/deploy`  
`sudo chown -R deploy.deploy /home/deploy/文件或目录`
- 2) 通过 xftp, crt 等工具或者 sz 命令下载即可

## 11.7 服务管理规范

### 1) centos7

服务启动: `sudo systemctl start 服务名`  
服务停止: `sudo systemctl stop 服务名`  
服务重启: `sudo systemctl restart 服务名`  
服务开机启动: `sudo systemctl enable 服务名`  
取消开机启动: `sudo systemctl disable 服务名`

### 2) centos6

服务启动: `sudo service 服务名 start`  
`sudo /etc/init.d/服务名 start`  
服务停止: `sudo service 服务名 stop`

`sudo /etc/init.d/服务名 stop`

服务重启: `sudo service 服务名 restart`

`sudo /etc/init.d/服务名 restart`

服务开机启动: `sudo chkconfig --add 服务名`

取消开机启动: `sudo chkconfig --del 服务名`