MySQL课程介绍

- 该阶段课程时间为6天
  - Day1主要讲MySQL的安装部署及启动排错
  - Day2主要讲MySQL的体系结构及基本的SQL语句
  - Day3主要讲MySQL的备份恢复
  - Day4主要讲MySQL的Replication
  - Day5主要讲MySQL的高可用架构及读写分离
  - Day6项目实战
- 该阶段的培养目标
  - 能够在任何环境任何平台下以任何方式对MySQL进行部署安装
  - 能够对数据库进行基本的日常管理维护
  - 能够理解MySQL的复制原理以及各种高可用架构
  - 能够搭建基本的MySQL的主从架构以及使用中间件进行读写分离

环境准备：

- 掌握centos7系统的安装
- 掌握centos7系统的基本使用
  - 网络配置
  - 主机名配置
  - yum源配置
  - centos7中的服务管理

一、centos7系统的安装

任务需求：

1. 最小化安装一个centos7操作系统，版本为7.3
2. 安装基础库和开发工具包
3. 该服务器日后用于安装mysql数据库
4. 磁盘大小20g，内存至少1G，逻辑卷管理

Table 8.3. Recommended System Swap Space

Amount of RAM in the system	Recommended swap space	Recommended swap space if allowing for hibernation
less than 2 GB	2 times the amount of RAM	3 times the amount of RAM
2 GB - 8 GB	Equal to the amount of RAM	2 times the amount of RAM
8 GB - 64 GB	4GB to 0.5 times the amount of RAM	1.5 times the amount of RAM
more than 64 GB	workload dependent (at least 4GB)	hibernation not recommended

表 11.2. 推荐的系统 swap 空间

系统中的 RAM 量	推荐的 swap 空间	允许休眠的建议 swap 空间大小
低于 2 GB	RAM 量的倍数	RAM 容量的三倍
2 GB - 8 GB	等于 RAM 量	RAM 量的倍数
8 GB - 64 GB	4 GB 到 RAM 容量的 0.5 倍	RAM 容量的 1.5 倍
超过 64 GB	独立负载（至少 4GB）	不建议使用休眠功能

二、系统基本配置

1. 基本网络配置

1.1 NetworkManager简介

在Red Hat Enterprise Linux 7中，默认的网络服务由NetworkManager(网络管理器)提供,这是一个动态的网络控制和配置守护进程，它试图在可用的时候保持网络设备和连接的活跃和活跃。

1.2 NetworkManager安装启动

```
1. 查看NetworkManager是否安装（默认已安装）
[root@localhost ~]# rpm -q NetworkManager
NetworkManager-1.4.0-12.el7.x86_64
注意：如果没有安装请安装它

2. 查看NetoworkManager状态（确保是running）
[root@localhost ~]# systemctl status NetworkManager
● NetworkManager.service - Network Manager
   Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service; enabled; vendor preset:
enabled)
   Active: active (running) since Sat 2018-06-16 10:23:54 CST; 24h ago
     Docs: man:NetworkManager(8)
```

注意：如果没有启动请启动它，命令如下：

```
[root@localhost ~]# systemctl start NetworkManager //启动
[root@localhost ~]# systemctl stop NetworkManager  //停止
[root@localhost ~]# systemctl reload NetworkManager   //重新加载配置文件
[root@localhost ~]# systemctl start NetworkManager //重启服务
```

查看是否开机自启动：

```
centos6：
# chkconfig --list
# chkconfig --list|grep NetworkManager
centos7：
# systemctl list-unit-files
# systemctl list-unit-files|grep NetworkManager
```

设定服务开机自启动|自关闭：

```
[root@localhost ~]# systemctl disable NetworkManager.service   //开机不自动启动
[root@localhost ~]# systemctl enable NetworkManager.service    //开机自启动
```

2. 配置静态IP地址

环境准备：

1. 添加一张虚拟网卡，网络模式为仅主机模式
2. 修改仅主机模式下的子网网段
3. 重启NetworkManager服务让新添加的网卡自动获取IP地址

2.1 文本工具配置静态IP

```
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:4c:a3:04 brd ff:ff:ff:ff:ff:ff
    inet 192.168.159.133/24 brd 192.168.159.255 scope global dynamic ens33
       valid_lft 1649sec preferred_lft 1649sec
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:4c:a3:0e brd ff:ff:ff:ff:ff:ff
    inet 10.1.1.1/32 brd 10.1.1.1 scope global ens37
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe4c:a30e/64 scope link
       valid_lft forever preferred_lft forever
```



```
[root@localhost ~]# nmtui
```

2.2 命令工具nmcli配置静态IP

- 使用 nmcli 工具启动和停止任意网络接口，其中包括主接口。例如：


```
[root@localhost ~]# nmcli dev status      //查看当前可用设备
DEVICE  TYPE      STATE         CONNECTION
ens33   ethernet  connected     ens33
ens37   ethernet  disconnected  --
lo      loopback  unmanaged     --

[root@localhost ~]# nmcli con show        //查看当前可用的网络连接
NAME   UUID                                  TYPE            DEVICE
ens33  c96bc909-188e-ec64-3a96-6a90982b08ad  802-3-ethernet  ens33
ens37  a22469d0-14c0-34fc-ae56-b7fa7b73a2df  802-3-ethernet  --

[root@localhost ~]# nmcli con up id ens37 //启动指定的网卡
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/22)
[root@localhost ~]# nmcli con show
NAME   UUID                                  TYPE            DEVICE
ens33  c96bc909-188e-ec64-3a96-6a90982b08ad  802-3-ethernet  ens33
ens37  a22469d0-14c0-34fc-ae56-b7fa7b73a2df  802-3-ethernet  ens37

[root@localhost ~]# nmcli dev disconnect ens37  //断开指定网卡连接
Device 'ens37' successfully disconnected.
```

- 使用nmcli添加静态以太网连接

```
创建名为my-home的静态连接配置文件:
[root@localhost ~]# nmcli con add type ethernet con-name my-home ifname ens37 ip4 10.1.1.2/24
gw4 10.1.1.254

说明:
NetworkManager会将其内部参数ipv4.method设定为manual，将connection.autoconnect设定为yes。
NetworkManager还会将设置写入 /etc/sysconfig/network-scripts/ifcfg-my-home文件，其中会将对应
BOOTPROTO设定为 none，并将ONBOOT设定为yes。

配置ipv4 DNS服务器:
//指定dns服务器，会替换之前的dns服务器
[root@localhost ~]# nmcli con mod my-home ipv4.dns "114.114.114.114"

//+ipv4.dns代表追加dns服务器
[root@localhost ~]# nmcli con mod my-home +ipv4.dns "8.8.8.8"
```

```
[root@localhost ~]# nmcli con mod my-home -ipv4.dns "8.8.8.8"     //删除dns服务器

[root@localhost ~]# nmcli con up my-home     //激活连接
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/29)

[root@localhost ~]# nmcli con delete my-home //删除
Connection 'my-home' (3b6c8c75-ad73-43a1-ae85-a02518c112fa) successfully deleted.
```

3. 配置主机名

- 使用nmtui工具配置主机名

```
[root@localhost ~]# nmtui     //立刻马上生效
```

- 使用 hostnamectl 配置主机名

```
[root@localhost ~]# hostnamectl status //查看所有主机名
[root@localhost ~]# hostnamectl set-hostname mysql01.Misshou.com  //设置所有主机名
[root@localhost ~]# hostname -f     //查看完全规范的主机名(FQDN)
mysql01.misshou.com
```

补充了解:

hostnamectl 可用于查询与修改系统主机名以及其他相关设置。 所谓"主机名"，其实有三种不同的含义：

"pretty"主机名，仅供人类阅读，可以包含各种特殊字符，且无长度限制。 例如"Lennart's Laptop"(必须是UTF-8编码)。 pretty(易读主机名)、chassis(设备类型)、icon(图标名称)存储在 `/etc/machine-info` 文件中，详见 machine-info(5) 手册。

"static"主机名，用于在系统启动时设置内核的主机名，仅可包含 "-","a-z","0-9" 字符，且最大不超过64个字符长度。 例如"lennarts-laptop"。此种主机名就是通常所说的"主机名"，存储在 `/etc/hostname` 文件中， 详见 hostname(5) 手册。

"transient"主机名，是从网络配置中获得的 fallback 主机名， 同样仅可包含 "-","a-z","0-9" 字符，且最大不超过64个字符长度。 如果存在"static"主机名且不等于"localhost"， 那么将忽略"transient"主机名。

4. 配置本地yum源

```
[root@mysql01 ~]# mount -o ro /dev/sr0 /mnt
[root@mysql01 ~]# cd /etc/yum.repos.d/
[root@mysql01 yum.repos.d]# cat local.repo
[local]
name=xxx
baseurl=file:///mnt
enabled=1
gpgcheck=0

为了能够使用tab键补全，需要安装bash-completion软件包
[root@mysql01 ~]# yum -y install bash-completion
```

5. 关闭防火墙和selinux

- 关闭防火墙

```
[root@mysql01 ~]# systemctl list-unit-files --type service --all|grep firewall
firewalld.service                           enabled
[root@mysql01 ~]# systemctl stop firewalld
[root@mysql01 ~]# systemctl disable firewalld
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
Removed symlink /etc/systemd/system/basic.target.wants/firewalld.service.
[root@mysql01 ~]# systemctl list-unit-files --type service --all|grep firewall
firewalld.service                           disabled
[root@mysql01 ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

说明：

在Centos7/Redhat7上如果想使用iptables防火墙的话，还需要安装iptables-services包；否则没有/etc/sysconfig/iptables配置文件。

```
[root@mysql01 ~]# yum -y install iptables-services
```

- 关闭selinux

```
[root@mysql01 ~]# getenforce
Enforcing
[root@mysql01 ~]# setenforce 0
[root@mysql01 ~]# getenforce
Permissive
[root@mysql01 ~]# vim /etc/sysconfig/selinux
...
SELINUX=disabled
...
```

三、Centos6和Centos7主要区别

1. 服务管理程序的区别

1.1 Centos6使用init管理

- 服务的启动、关闭、状态等方式

所有的服务启动脚本通通放置于/etc/init.d/下，基本上都是使用bash shell script所写的脚本，需要启动、关闭、重新启动、观察状态时，可以通过如下命令方式来处理：

```
启动：/etc/init.d/xxx start   或者 service xxx start
关闭：/etc/init.d/xxx stop
重新启动：/etc/init.d/xxx restart
查看状态：/etc/init.d/xxx status
```

- 服务的分类
  - 独立服务

  有自己独立的启动脚本，服务常驻于内存中，响应速度快等特点。
  - 依赖服务

  依赖于xinetd或inetd服务，也称为“super daemon”，该超级守护程序通常管理一些轻量级的服务，通过提供对应的[1]来进行管理。当有用户访问时，xinetd程序会唤醒相应的服务。优点是不占用系统资源，缺点是唤醒服务需要一定的延迟。
- 服务的依赖

  服务与服务之间有可能存在相互依赖关系，比如文件共享服务nfs，要启动nfs，必须首先启动rpcbind服务，此时需要管理员手动先去启动rpcbind，再去启动nfs。也就是说init管理下的服务是没有办法协助直接启动依赖服务的。
- 根据系统运行级别进行启动

  centos6中，硬件初始化完毕后，kernel主动呼叫/sbin/init进程，然后init进程根据用户定义的运行级别(/etc/inittab)来唤醒相应目录下不同的服务[2]
- 开机是否自动启动

  如果想要让一个服务开机自动启动，那么需要将上面的/etc/rc.d/rcX/目录下以K开头的服务变成S开头，只需要如下命令就可以完成：

  ```
  开机自动启动：  chkconfig xxx on
  开机不自动启动：  chkconfig xxx off
  查看是否开机自启动：  chkconfig --list xxx
  ```

1.2 Centos7使用systemd管理

systemd 使用unit分类进行管理！好处如下：

- 并行处理 所有服务，提高开机速度

- 灵活的启动和管理方式

  systemd全部都是由systemd服务搭配systemctl指令来处理，无须其他额外的指令来配合。不像system V启动流程中，还要init, chkconfig, service...等指令。此外， systemd由于常驻内存，因此任何要求都可以按需立即处理。

- 依赖服务的自我检查

  由于systemd可以定义服务依赖性的检查，因此如果B服务是架构在A服务上面启动的，那当你在没有启动A服务的情况下仅手动启动B服务时， systemd会自动帮你启动A服务。

- 根据服务或者程序功能分类
  - systemd管理的服务非常多，为了理清所有服务的功能，首先systemd先定义所有的服务为一个服务单位(unit)，并将该unit归类到不同的服务类型(type)去。
  - systemd将服务单位(unit)区分为service, socket, target, path, snapshot, timer等多种不同的类型(type)，方便管理员分类与记忆。
- 将多个daemon集合成为一个群组

  如同system V的init中有个runlevel的特色，systemd亦将许多的功能集合成为一个所谓的target项目，这个项目主要在设计操作系统的运行级别，所以是集合了许多的daemon，也就是执行某个target就是执行好多个daemon的意思！

注意：虽然如此，systemd有些地方没有完全取代init，如下：

- 在runlevel 的对应上，大概仅有runlevel 1, 3, 5 有对应到systemd 的某些target 类型而已，没有全部对应；
- systemd 都用systemctl 这个命令俩管理，而systemctl支持的语法有限制，不像/etc/init.d/xxx就是纯脚本可以自订参数，systemctl 不可自订参数。

2. systemd管理的文件目录

基本上， systemd 将过去所谓的服务执行脚本通通称为一个服务单位(unit)，而每种服务单位依据功能来区分时，就分类为不同的类型(type)。类型多，如何设置，在哪设置，需要我们掌握。

目录	说明
/usr/lib/systemd/system/	每个服务最主要的启动脚本设定，类似以前的/etc/init.d底下的文件,rpm包安装后单元文件默认存放位置
/run/systemd/system/	系统执行过程中所产生的服务脚本，这里脚本的优先级要比/usr/lib/systemd/system/高
/etc/systemd/system/	管理员根据需求所建立的执行脚本，这个目录有点像以前/etc/rc.d/rc5.d/Sxx之类的功能，执行优先序又比/run/systemd/system/

注意：

1. 系统开机会不会执行某些服务其实是看/etc/systemd/system/ 底下的设定，所以该目录底下就是一大堆软连接文件，链接到/usr/lib/systemd/system/下。
2. 想要对某个服务进行启动配置修改，应到/usr/lib/systemd/system/下找到相应服务配置文件修改。
3. 参照官网：**Systemd Unit Files Locations**

3. systemd的unit类型说明

Available systemd Unit Types

文件名	主要功能
*.service	服务单元(service unit)：主要是系统服务，包括服务器本身所需要的本机服务以及网路服务，经常被使用到的服务大多是这种类型。所以，这也是最常见的类型了。
*.socket	socket单元(socket unit):主要用于进程之间通讯，当有透过此socket文件发送信息要连结服务时，就依据当时的状态将该用户的请求传送到对应的daemon， 若daemon 尚未启动，则启动该daemon 后再传送用户的要求。一般用于本机服务较多，类似xinetd服务。
*.target	目标单元(target unit):是一群unit 的集合，比如multi-user.target 其实就是一堆服务的集合，也就是说， 选择执行multi-user.target 就是执行一堆*.service 或*.socket 之类的服务.
.mount和.automount	挂载单元(mount /automount unit)：文件系统挂载相关服务，例如来自网路的自动挂载、NFS网络系统挂载等。
*.path	路径单元(path unti):某些服务需要检测特定的目录来提供服务，比如最常见的打印服务，就是通过检测打印队列目录来启动打印功能。

4. systemctl命令管理服务

- **Centos6/Redhat6中使用service、chkconfig命令来实现服务的启动配置**

服务程序管理主要是依赖于/etc/rc.d/init.d/目录下的所有的脚本文件，所以可以使用service调用。

```
service命令：
Usage：
service  servicename  {start|stop|status|restart|reload}
chkconfig servername {on|off}
chkconfig --list  servicename       //查看服务是否开机自启动
```

- **Centos7/RHEL7中使用systemctl命令实现服务的启动配置**

服务程序管理主要是依赖于 /usr/lib/systemd/system/目录下的以.service结尾的unit file文件来管理，我们使用systemctl来调用。

```
systemctl命令：
Usage: systemctl { start | restart | stop | status | enable | disable }  servername
systemctl list-unit-files                              //查看所有单元文件的状态
systemctl list-unit-file --type=target      //列出系统target单元
systemctl is-enabled NAME.service       //查看服务是否开机自启动
systemctl is-active name.service            //查看服务是否激活

说明：
systemd使用unit file文件进行控制程序的启动和关闭，需要注意以下几点：

1. 所有的unit file文件都保存在/usr/lib/systemd/system/目录下面，而/etc/systemd/system下的是unit file文件的软连接

2. 使用yum安装的程序会自带一个unit file文件，保存在/usr/lib/systemd/system/目录下，比如：/usr/lib/systemd/system/httpd.service
```

```
3. 常见的unit file文件类型
   Service unit : NAME.service          //用于定义系统服务
   Target unit: NAME.target             //用于模拟实现"运行级别"
   Device unit: NAME.device              //用于定义内核识别的设备
   Mount unit: NAME.mount               //用于定义文件系统的挂载点
   Socket unit: NAME.socket             //用于表示进程间通信用到的socket文件
   Snapshot: NAME.snapshot              //用于管理系统快照
   Swap unit: NAME.swap                 //用于管理Swap设备
   Automount unit: NAME.automount       //用于文件系统自动挂载点设备
   Path unit: NAME.path                 //用于定义文件系统中的某一文件或目录
```

总结:

1. systemctl对于服务的启动、关闭等管理命令是固定不变的。
2. 如果不是yum安装的程序，程序的启动、关闭无法被systemd管控，因此就无法使用systemctl来启动，但是我们可以自己编写一个 unit file，实现让systemctl来管理非yum安装的程序。

5. 运行级别

systemctl使用目标target取代了运行级别的概念.

Centos6/RedHat6	Centos7/RedHat7
init管理	systemd管理
init 0	systemctl poweroff
init 1	systemctl isolate rescue.target
init 3	systemctl isolate multi-user.target
init 5	systemctl isolate
init 6	systemctl reboot

Centos7中运行级别的设置:

```
[root@mysql01 ~]# cat /etc/inittab
# inittab is no longer used when using systemd.
#
# ADDING CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
#
# Ctrl-Alt-Delete is handled by /usr/lib/systemd/system/ctrl-alt-del.target
#
# systemd uses 'targets' instead of runlevels. By default, there are two main targets:
#
# multi-user.target: analogous to runlevel 3    //文本模式
# graphical.target: analogous to runlevel 5     //图形模式
#
# To view current default target, run:
# systemctl get-default                     //查看当前的默认运行级别
#
```

```
# To set a default target, run:
# systemctl set-default TARGET.target   //设置当前运行级别
```

总结： 1. 不管是centos6/rhel6还是centos7/rhel7，如果要使用service或者systemctl命令来启动、停止服务，那么/etc/init.d/或者/usr/lib/systemd/system/下需要有服务相对应的脚本文件或者unit file文件，否则需要自己创建它。 2. 详细信息请看man文档：# man systemd

补充：

RHEL7 关机、重启、挂起、创建快照

```
关机：systemctl halt | systemctl poweroff
重启：systemctl reboot
挂起：systemctl suspend
快照：systemctl hibernate
快照并挂起：systemctl hybrid-sleep
```

课后参考：

systemd介绍

---

1. Socket又称"套接字"，应用程序通常通过"套接字"向网络发出请求或者应答网络请求。用于描述IP地址和端口，可以用来实现不同计算机之间的通信。↩

2. 其中，0-6代表运行级别；S代表开机启动；K代表开机不启动；xxx为数字代表启动顺序↩