11 【实操篇-定时任务 软件安装 克隆虚拟机】

1.crontab 系统定时任务

在实际工作中，系统的定时任务一般是需要重复执行的，这就需要使用 crontab 命令来执行循环定时任务。

每个用户都可以实现自己的 crontab 定时任务，只需使用这个用户身份执行“crontab -e”命令即可。当然，这个用户不能写入 /etc/cron.deny 文件。

crontab 命令的基本格式如下：

```
[root@localhost ~]# crontab [选项] [file]
```

注意，这里的 file 指的是命令文件的名字，表示将 file 作为 crontab 的任务列表文件并载入 crontab，若在命令行中未指定文件名，则此命令将接受标准输入（键盘）上键入的命令，并将它们键入 crontab。

选项	功能
-u user	用来设定某个用户的 crontab 服务，例如 "-u demo" 表示设备 demo 用户的 crontab 服务，此选项一般有 root 用户来运行。
-e	编辑某个用户的 crontab 文件内容。如果不指定用户，则表示编辑当前用户的 crontab 文件。
-l	显示某用户的 crontab 文件内容，如果不指定用户，则表示显示当前用户的 crontab 文件内容。
-r	从 /var/spool/cron 删除某用户的 crontab 文件，如果不指定用户，则默认删除当前用户的 crontab 文件。
-i	在删除用户的 crontab 文件时，给确认提示。

其实 crontab 定时任务非常简单，只需执行“crontab -e”命令，然后输入想要定时执行的任务即可。不过，当我们执行“crontab -e”命令时，打开的是一个空文件，而且操作方法和 Vim 是一致的。那么，这个文件的格式才是我们真正需要学习的内容。文件格式如下：

```
[root@localhost ！]# crontab -e
#进入 crontab 编辑界面。会打开vim编辑你的任务
* * * * * 执行的任务
```

这个文件中是通过 5 个“”*来确定命令或任务的执行时间的，这 5 个*“。

项目	含义	范围
第一个"*"	一小时当中的第几分钟（minute）	0~59
第二个"*"	一天当中的第几小时（hour）	0~23
第三个"*"	一个月当中的第几天（day）	1~31
第四个"*"	一年当中的第几个月（month）	1~12
第五个"*"	一周当中的星期几（week）	0~7（0和7都代表星期日）

在时间表示中，还有一些特殊符号需要学习。

特殊符号	含义
*（星号）	代表任何时间。比如第一个"*"就代表一小时种每分钟都执行一次的意思。
,（逗号）	代表不连续的时间。比如"0 8，12，16***命令"就代表在每天的 8 点 0 分、12 点 0 分、16 点 0 分都执行一次命令。
-（中杠）	代表连续的时间范围。比如"0 5 ** 1-6命令"，代表在周一到周六的凌晨 5 点 0 分执行命令。
*/n	代表每隔多久执行一次。比如"*/10命令"，代表每隔 10 分钟就执行一次命令。

当“crontab -e”编辑完成之后，一旦保存退出，那么这个定时任务实际就会写入 /var/spool/cron/ 目录中，每个用户的定时任务用自己的用户名进行区分。而且 crontab 命令只要保存就会生效，只要 crond 服务是启动的。知道了这 5 个时间字段的含义，我们多举几个时间的例子来熟悉一下时间字段。

时间	含义
45 22 ***命令	在 22 点 45 分执行命令
0 17 ** 1命令	在每周一的 17 点 0 分执行命令
0 5 1，15**命令	在每月 1 日和 15 日的凌晨 5 点 0 分执行命令
40 4 ** 1-5命令	在每周一到周五的凌晨 4 点 40 分执行命令
*/10 4 ***命令	在每天的凌晨 4 点，每隔 10 分钟执行一次命令
0 0 1，15 * 1命令	在每月 1 日和 15 日，每周一个 0 点 0 分都会执行命令，注意：星期几和几日最好不要同时出现，因为它们定义的都是天，非常容易让管理员混淆

现在我们已经对这 5 个时间字段非常熟悉了，可是在“执行的任务”字段中都可以写什么呢？既可以定时执行系统命令，也可以定时执行某个 Shell 脚本，这里举几个实际的例子。

【例 1】让系统每隔 5 分钟就向 /tmp/test 文件中写入一行“11”，验证一下系统定时任务是否会执行。

```
[root@localhost ~]# crontab -e
#进入编辑界面
*/5 * * * * /bin/echo "11" >> /tmp/test
```

这个任务在时间工作中没有任何意义，但是可以很简单地验证我们的定时任务是否可以正常执行。如果觉得每隔 5 分钟太长，那就换成“*”，让它每分钟执行一次。而且和 at 命令一样，如果我们定时执行的是系统命令，那么最好使用绝对路径。

【例 2】让系统在每周二的凌晨 5 点 05 分重启一次。

```
[root@localhost ~]# crontab -e
5.5 * * 2 /sbin/shutdown -r now
```

如果服务器的负载压力比较大，则建议每周重启一次，让系统状态归零。比如绝大多数游戏服务器每周维护一次，维护时最主要的工作就是重启，让系统状态归零。这时可以让我们的服务器自动来定时执行。

【例 3】在每月 1 日、10 日、15 日的凌晨 3 点 30 分都定时执行日志备份脚本 autobak.sh。

```
[root@localhost ~]# crontab -e
30.3 1，10，15 * * /root/sh/autobak.sh
```

这些定时任务保存之后，就可以在指定的时间执行了。我们可以使用命令来查看和删除定时任务，命令如下：

```
[root@localhost ~]# crontab -l
#查看root用户的crontab任务
*/5 * * * * /bin/echo "11" >> /tmp/test
5.5 * * 2 /sbin/shutdown -r now
30.3 1，10，15 * * /root/sh/autobak.sh
[root@localhost ~]# crontab -r
#删除root用户所有的定时任务。如果只想删除某个定时任务，则可以执行“crontab -e”命令进入
#编辑模式手工删除
[root@localhost ~]# crontab -l
no crontab for root
#删除后，再查询就没有root用户的定时任务了
```

在书写 crontab 定时任务时，需要注意以下几个事项：

- 6 个选项都不能为空，必须填写。如果不确定，则使用“*”代表任意时间。
- crontab 定时任务的最小有效时间是分钟，最大有效时间是月。像 2018 年某时执行、3 点 30 分 30 秒这样的时间都不能被识别。
- 在定义时间时，日期和星期最好不要在一条定时任务中出现，因为它们都以天为单位，非常容易让管理员混淆。
- 在定时任务中，不管是直接写命令，还是在脚本中写命令，最好都使用绝对路径。有时使用相对路径的命令会报错。

2.软件包管理

2.1 源码包和二进制包

Linux下的软件包众多，且几乎都是经 GPL 授权、免费开源（无偿公开源代码）的。这意味着如果你具备修改软件源代码的能力，只要你愿意，可以随意修改。

> GPL，全称 General Public License，中文名称“通用性公开许可证”，简单理解 GPL 就是一个保护软件自由的一个协议，经 GPL 协议授权的软件必须开源，请猛击《开源协议》了解更多信息。

Linux下的软件包可细分为两种，分别是源码包和二进制包。

2.1.1 源码包

实际上，源码包就是一大堆源代码程序，是由程序员按照特定的格式和语法编写出来的。

我们都知道，计算机只能识别机器语言，也就是二进制语言，所以源码包的安装需要一名“翻译官”将“abcd”翻译成二进制语言，这名“翻译官”通常被称为编译器。

“编译”指的是从源代码到直接被计算机（或虚拟机）执行的目标代码的翻译过程，编译器的功能就是把源代码翻译为二进制代码，让计算机识别并运行。

虽然源码包免费开源，但用户不会编程怎么办？一大堆源代码程序不会使用怎么办？源码包容易安装吗？等等这些都是使用源码包安装方式无法解答的问题。

另外，由于源码包的安装需要把源代码编译为二进制代码，因此安装时间较长。比如，大家应该都在Windows下安装过 QQ，QQ 功能较多，程序相对较大（有 70 MB左右），但由于其并非是以源码包的形式发布，而是编译后才发布的，因此只需几分钟（经过简单的配置）即可安装成功。但如果我们以源码包安装的方式在 Linux 中安装一个 MySQL 数据库，即便此软件的压缩包仅有 23 MB左右，也需要 30 分钟左右的时间（根据硬件配置不同，略有差异）。

通过对比你会发现，源码包的编译是很费时间的，况且绝多大数用户并不熟悉程序语言，在安装过程中我们只能祈祷程序不要报错，否则初学者很难解决。

为了解决使用源码包安装方式的这些问题，Linux 软件包的安装出现了使用二进制包的安装方式。

2.1.2 二进制包

二进制包，也就是源码包经过成功编译之后产生的包。由于二进制包在发布之前就已经完成了编译的工作，因此用户安装软件的速度较快（同 Windows下安装软件速度相当），且安装过程报错几率大大减小。

二进制包是 Linux 下默认的软件安装包，因此二进制包又被称为默认安装软件包。目前主要有以下 2 大主流的二进制包管理系统：

- RPM 包管理系统：功能强大，安装、升级、查询和卸载非常简单方便，因此很多 Linux 发行版都默认使用此机制作为软件安装的管理方式，例如 Fedora、CentOS、SuSE 等。
- DPKG 包管理系统：由 Debian Linux 所开发的包管理机制，通过 DPKG 包，Debian Linux 就可以进行软件包管理，主要应用在 Debian 和 Ubuntu 中。

> RPM 包管理系统和 DPKG 管理系统的原理和形式大同小异，可以触类旁通。由于本教程使用的是 CentOS 6.x 版本，因此本节主要讲解 RPM 二进制包。

2.1.3 源码包 VS RPM二进制包

源码包一般包含多个文件，为了方便发布，通常会将源码包做打包压缩处理，Linux 中最常用的打包压缩格式为“tar.gz”，因此源码包又被称为 Tarball。

Tarball 是 Linux 系统的一款打包工具，可以对源码包进行打包压缩处理，人们习惯上将最终得到的打包压缩文件称为 Tarball 文件。

源码包需要我们自己去软件官方网站进行下载，包中通常包含以下内容：

- 源代码文件。
- 配置和检测程序（如 configure 或 config 等）。
- 软件安装说明和软件说明（如 INSTALL 或 README）。

总的来说，使用源码包安装软件具有以下几点好处：

- 开源。如果你有足够的能力，则可以修改源代码。
- 可以自由选择所需的功能。
- 因为软件是编译安装的，所以更加适合自己的系统，更加稳定，效率也更高。
- 卸载方便。

但同时，使用源码包安装软件也有几点不足：

- 安装过程步骤较多，尤其是在安装较大的软件集合时（如 LAMP 环境搭建），容易出现拼写错误。
- 编译时间较长，所以安装时间比二进制安装要长。
- 因为软件是编译安装的，所以在安装过程中一旦报错，新手很难解决。

相比源码包，二进制包是在软件发布时已经进行过编译的软件包，所以安装速度比源码包快得多（和 Windows 下软件安装速度相当）。也正是因为已经进行通译，大家无法看到软件的源代码。

使用 RMP 包安装软件具有以下 2 点好处：

1. 包管理系统简单，只通过几个命令就可以实现包的安装、升级、查询和卸载。
2. 安装速度比源码包安装快得多。

与此同时，使用 RMP 包安装软件有如下不足：

- 经过编译，不能在看到源代码。
- 功能选择不如源码包灵活。
- 依赖性。有时我们会发现，在安装软件包 a 时需要先安装 b 和 c，而在安装 b 时需要先安装 d 和 e。这就需要先安装 d 和 e，再安装 b 和 c，最后才能安装 a。比如，我买了一个漂亮的灯具，打算安装在客厅里，可是在安装灯具之前，客厅需要有顶棚，并且顶棚需要刷好油漆。安装软件和装修及其类似，需要有一定的顺序，但是有时依赖性会非常强。

2.1.4 如何选择

通过源码包和 RMP 二进制包的对比，在 Linux 进行软件安装时，我们应该使用哪种软件包呢?

为了更好的区别两种软件包，这里举个例子。假设我们想做一套家具，源码包就像所有的家具完全由自己动手手工打造（手工编译），想要什么样的板材、油漆、颜色和样式都由自己决定（功能自定义，甚至可以修改源代码）。想想就觉得爽，完全不用被黑心的厂商所左右，而且不用担心质量问题（软件更适合自己的系统，效率更高，更加稳定）。但是，所花费的时间大大超过了买一套家具的时间（编译浪费时间），而且自己真的有做木工这个能力吗（需要对源代码非常了解）？就算请别人定制好的家具，再由自己组装，万一哪个部件不匹配（报错很难解决），怎么办?

那么二进制包呢？也是我们需要一套家具，去商场买了一套（安装简单），家具都是现成的，不会有哪个部件不匹配，除非因为自身问题没有量好尺寸而导致放不下（报错很少）。但是我们完全不知道这套家具用的是什么材料、油漆是否合格，而且家具的样式不能随意选择（软件基本不能自定义功能）。

2.2 RPM包统一命名规则

RPM 二进制包的命名需遵守统一的命名规则，用户通过名称就可以直接获取这类包的版本、适用平台等信息。

RPM 二进制包命名的一般格式如下：

> 包名-版本号-发布次数-发行商-Linux平台-适合的硬件平台-包扩展名

例如，RPM 包的名称是`httpd-2.2.15-15.el6.centos.1.i686.rpm`，其中：

- httped：软件包名。这里需要注意，httped 是包名，而 httpd-2.2.15-15.el6.centos.1.i686.rpm 通常称为包全名，包名和包全名是不同的，在某些 Linux 命令中，有些命令（如包的安装和升级）使用的是包全名，而有些命令（包的查询和卸载）使用的是包名，一不小心就会弄错。
- 2.2.15：包的版本号，版本号的格式通常为`主版本号.次版本号.修正号`。
- 15：二进制包发布的次数，表示此 RPM 包是第几次编程生成的。
- el*：软件发行商，el6 表示此包是由 Red Hat 公司发布，适合在 RHEL 6.x (Red Hat Enterprise Unux) 和 CentOS 6.x 上使用。
- centos：表示此包适用于 CentOS 系统。
- i686：表示此包使用的硬件平台，目前的 RPM 包支持的平台如表所示：

平台名称	适用平台信息
i386	386 以上的计算机都可以安装
i586	686 以上的计算机都可以安装
i686	奔腾 II 以上的计算机都可以安装，目前所有的 CPU 是奔腾 II 以上的，所以这个软件版本居多
x86_64	64 位 CPU 可以安装
noarch	没有硬件限制

- rpm：RPM 包的扩展名，表明这是编译好的二进制包，可以使用 rpm 命令直接安装。此外，还有以 src.rpm 作为扩展名的 RPM 包，这表明是源代码包，需要安装生成源码，然后对其编译并生成 rpm 格式的包，最后才能使用 rpm 命令进行安装。

有人可能会问，Linux 系统不靠扩展名分区文件类型，那为什么包全名中要包含 .rpm 扩展名呢？其实，这里的扩展名是为系统管理员准备的，如果我们不对 RPM 包标注扩展名，管理员很难知道这是一个 RPM 包，当然也就无法正确使用。

2.3 RPM包安装、卸载和升级

2.3.1 RPM包默认安装路径

通常情况下，RPM 包采用系统默认的安装路径，所有安装文件会按照类别分散安装到表所示的目录中。

安装路径	含 义
/etc/	配置文件安装目录
/usr/bin/	可执行的命令安装目录
/usr/lib/	程序所使用的函数库保存位置
/usr/share/doc/	基本的软件使用手册保存位置
/usr/share/man/	帮助文件保存位置

RPM 包的默认安装路径是可以通过命令查询的。

除此之外，RPM 包也支持手动指定安装路径，但此方式并不推荐。因为一旦手动指定安装路径，所有的安装文件会集中安装到指定位置，且系统中用来查询安装路径的命令也无法使用（需要进行手工配置才能被系统识别），得不偿失。

与 RPM 包不同，源码包的安装通常采用手动指定安装路径（习惯安装到 /usr/local/ 中）的方式。既然安装路径不同，同一 apache 程序的源码包和 RPM 包就可以安装到一台 Linux 服务器上（但同一时间只能开启一个，因为它们需要占用同一个 80 端口）。

2.3.2 RPM 包的安装

安装 RPM 的命令格式为：

`[root@localhost ~]# rpm -ivh 包全名`

注意一定是包全名。涉及到包全名的命令，一定要注意路径，可能软件包在光盘中，因此需提前做好设备的挂载工作。

此命令中各选项参数的含义为：

- -i：安装（install）；
- -v：显示更详细的信息（verbose）；
- -h：打印 #，显示安装进度（hash）；

例如，使用此命令安装 apache 软件包，如下所示：

```
[root@localhost ~]# rpm -ivh \
/mnt/cdrom/Packages/httpd-2.2.15-15.el6.centos.1.i686.rpm
Preparing...
####################
[100%]
1:httpd
####################
[100%]
```

注意，直到出现两个 100% 才是真正的安装成功，第一个 100% 仅表示完成了安装准备工作。

此命令还可以一次性安装多个软件包，仅需将包全名用空格分开即可，如下所示：

`[root@localhost ~]# rpm -ivh a.rpm b.rpm c.rpm`

如果还有其他安装要求（比如强制安装某软件而不管它是否有依赖性），可以通过以下选项进行调整：

- -nodeps：不检测依赖性安装。软件安装时会检测依赖性，确定所需的底层软件是否安装，如果没有安装则会报错。如果不管依赖性，想强制安装，则可以使用这个选项。注意，这样不检测依赖性安装的软件基本上是不能使用的，所以不建议这样做。
- -replacefiles：替换文件安装。如果要安装软件包，但是包中的部分文件已经存在，那么在正常安装时会报"某个文件已经存在"的错误，从而导致软件无法安装。使用这个选项可以忽略这个报错而覆盖安装。
- -replacepkgs：替换软件包安装。如果软件包已经安装，那么此选项可以把软件包重复安装一遍。
- -force：强制安装。不管是否已经安装，都重新安装。也就是 -replacefiles 和 -replacepkgs 的综合。
- -test：测试安装。不会实际安装，只是检测一下依赖性。
- -prefix：**指定安装路径。为安装软件指定安装路径，而不使用默认安装路径。**

apache 服务安装完成后，可以尝试启动：

```
[root@localhost ~]# systemctl start|stop|restart|status 服务名
```

各参数含义：

- start：启动服务；
- stop：停止服务；
- restart：重启服务；
- status: 查看服务状态；

例如：

`[root@localhost ~]# systemctl start httpd #启动apache服务`

服务启动后，可以查看端口号 80 是否出现。命令如下：

```
[root@localhost ~]# netstat -tlun | grep 80
tcp 0 0 :::80:::* LISTEN
```

2.3.3 RPM包的升级

使用如下命令即可实现 RPM 包的升级：

`[root@localhost ~]# rpm -Uvh 包全名`

-U（大写）选项的含义是：如果该软件没安装过则直接安装；若没安装则升级至最新版本。

`[root@localhost ~]# rpm -Fvh 包全名`

-F（大写）选项的含义是：如果该软件没有安装，则不会安装，必须安装有较低版本才能升级。

2.4.4 RPM包的卸载

RPM 软件包的卸载要考虑包之间的依赖性。例如，我们先安装的 httpd 软件包，后安装 httpd 的功能模块 mod_ssl 包，那么在卸载时，就必须先卸载 mod_ssl，然后卸载 httpd，否则会报错。

软件包卸载和拆除大楼是一样的，本来先盖的 2 楼，后盖的 3 楼，那么拆楼时一定要先拆除 3 楼。

如果卸载 RPM 软件不考虑依赖性，执行卸载命令会包依赖性错误，例如：

```
[root@localhost ~]# rpm -e httpd
error: Failed dependencies:
httpd-mmn = 20051115 is needed by (installed) mod_wsgi-3.2-1.el6.i686
httpd-mmn = 20051115 is needed by (installed) php-5.3.3-3.el6_2.8.i686
httpd-mmn = 20051115 is needed by (installed) mod_ssl-1:2.2.15-15.el6.
centos.1.i686
httpd-mmn = 20051115 is needed by (installed) mod_perl-2.0.4-10.el6.i686
httpd = 2.2.15-15.el6.centos.1 is needed by (installed) httpd-manual-2.2.
15-15.el6.centos.1 .noarch
httpd is needed by (installed) webalizer-2.21_02-3.3.el6.i686
httpd is needed by (installed) mod_ssl-1:2.2.15-15.el6.centos.1.i686
httpd=0:2.2.15-15.el6.centos.1 is needed by(installed)mod_ssl-1:2.2.15-
15.el6.centos.1.i686
```

2.4 rpm命令查询软件包

rpm 命令还可用来对 RPM 软件包做查询操作，具体包括：

- 查询软件包是否已安装；
- 查询系统中所有已安装的软件包；
- 查看软件包的详细信息；
- 查询软件包的文件列表；
- 查询某系统文件具体属于哪个 RPM 包。

使用 rpm 做查询命令的格式如下：

`[root@localhost ~]# rpm 选项 查询对象`

2.4.1 rpm -q：查询软件包是否安装

用 rpm 查询软件包是否安装的命令格式为：

`[root@localhost ~]# rpm -q 包名`

- -q 表示查询，是 query 的首字母。

例如，查看 Linux 系统中是否安装 apache，rpm 查询命令应写成：

```
[root@localhost ~]# rpm -q httpd
httpd-2.2.15-15.el6.centos.1.i686
```

注意这里使用的是包名，而不是包全名。因为已安装的软件包只需给出包名，系统就可以成功识别（使用包全名反而无法识别）。

2.4.2 rpm -qa：查询系统中所有安装的软件包

使用 rpm 查询 Linux 系统中所有已安装软件包的命令为：

```
[root@localhost ~]# rpm -qa
libsamplerate-0.1.7-2.1.el6.i686
startup-notification-0.10-2.1.el6.i686
gnome-themes-2.28.1-6.el6.noarch
fontpackages-filesystem-1.41-1.1.el6.noarch
gdm-libs-2.30.4-33.el6_2.i686
gstreamer-0.10.29-1.el6.i686
redhat-lsb-graphics-4.0-3.el6.centos.i686
…省略部分输出…
```

此外，这里还可以使用管道符查找出需要的内容，比如：

```
[root@localhost ~]# rpm -qa | grep httpd
httpd-devel-2.2.15-15.el6.centos.1.i686
httpd-tools-2.2.15-15.el6.centos.1.i686
httpd-manual-2.2.15-15.el6.centos.1.noarch
httpd-2.2.15-15.el6.centos.1.i686
```

相比`rpm -q 包名`命令，采用这种方式可以找到含有包名的所有软件包。

2.4.3 rpm -qi：查询软件包的详细信息

通过 rpm 命令可以查询软件包的详细信息，命令格式如下：

`[root@localhost ~]# rpm -qi 包名`

- -i 选项表示查询软件信息，是 information 的首字母。

例如，想查看 apache 包的详细信息，可以使用如下命令：

```
[root@localhost ~]# rpm -qi httpd
Name : httpd Relocations:(not relocatable)
#包名
Version : 2.2.15 Vendor:CentOS
#版本和厂商
Release : 15.el6.centos.1 Build Date: 2012年02月14日星期二 06时27分1秒
#发行版本和建立时间
Install Date: 2013年01月07日星期一19时22分43秒
Build Host:
c6b18n2.bsys.dev.centos.org
#安装时间
Group : System Environment/Daemons Source RPM:
httpd-2.2.15-15.el6.centos.1.src.rpm
#组和源RPM包文件名
Size : 2896132 License: ASL 2.0
#软件包大小和许可协议
Signature :RSA/SHA1,2012年02月14日星期二 19时11分00秒，Key ID
0946fca2c105b9de
#数字签名
Packager：CentOS BuildSystem http://bugs.centos.org
URL : http://httpd.apache.org/
#厂商网址
Summary : Apache HTTP Server
#软件包说明
Description:
The Apache HTTP Server is a powerful, efficient, and extensible web server.
#描述
```

2.4.4 rpm -ql：命令查询软件包的文件列表

通过前面的学习我们知道，rpm 软件包通常采用默认路径安装，各安装文件会分门别类安放在适当的目录文件下。使用 rpm 命令可以查询到已安装软件包中包含的所有文件及各自安装路径，命令格式为：

`[root@localhost ~]# rpm -ql 包名`

-l 选项表示列出软件包所有文件的安装目录。

例如，查看 apache 软件包中所有文件以及各自的安装位置，可使用如下命令：

```
[root@localhost ~]# rpm -ql httpd
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/README
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
...省略部分输出...
```

同时，rpm 命令还可以查询未安装软件包中包含的所有文件以及打算安装的路径，命令格式如下：

`[root@localhost ~]# rpm -qlp 包全名`

- -p 选项表示查询未安装的软件包信息，是 package 的首字母。

注意，由于软件包还未安装，因此需要使用“绝对路径+包全名”的方式才能确定包。

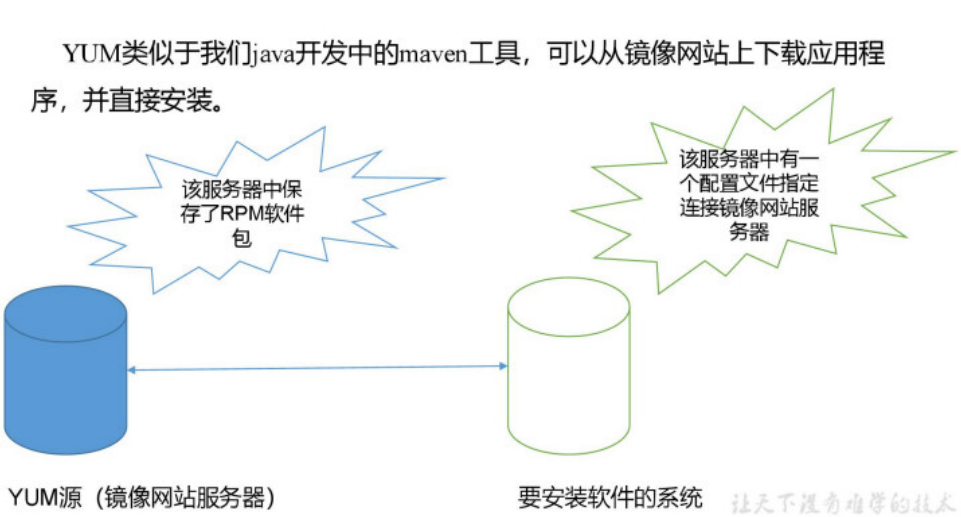
比如，我们想查看 bing 软件包（未安装，绝对路径为：/mnt/cdrom/Packages/bind-9.8.2-0.10.rc1.el6.i686.rpm）中的所有文件及各自打算安装的位置，可以执行如下命令：

```
[root@localhost ~]# rpm -qlp /mnt/cdrom/Packages/bind-9.8.2-
0.10.rc1.el6.i686.rpm
/etc/NetworkManager/dispatcher.d/13-named
/etc/logrotate.d/named
/etc/named
/etc/named.conf
/etc/named.iscdlv.key
/etc/named.rfc1912.zones
...省略部分输出...
```

2.5 yum是什么

本节介绍一种可自动安装软件包（自动解决包之间依赖关系）的安装方式。

yum，全称“Yellow dog Updater, Modified”，是一个专门为了解决包的依赖关系而存在的软件包管理器。就好像 Windows 系统上可以通过 360 软件管家实现软件的一键安装、升级和卸载，Linux 系统也提供有这样的工具，就是 yum。



可以这么说，yum 是改进型的 RPM 软件管理器，它很好的解决了 RPM 所面临的软件包依赖问题。yum 在服务器端存有所有的 RPM 包，并将各个包之间的依赖关系记录在文件中，当管理员使用 yum 安装 RPM 包时，yum 会先从服务器端下载包的依赖性文件，通过分析此文件从服务器端一次性下载所有相关的 RPM 包并进行安装。

yum 软件可以用 rpm 命令安装，安装之前可以通过如下命令查看 yum 是否已安装：

```
[root@localhost ~]# rpm -qa | grep yum
yum-metadata-parser-1.1.2-16.el6.i686
yum-3.2.29-30.el6.centos.noarch
yum-utils-1.1.30-14.el6.noarch
yum-plugin-fastestmirror-1.1.30-14.el6.noarch
yum-plugin-security-1.1.30-14.el6.noarch
```

可以看到，系统上已经安装了 yum。

使用 rpm 命令安装 yum 的具体方式可查看《Linux怎么安装yum》一节。

使用 yum 安装软件包之前，需指定好 yum 下载 RPM 包的位置，此位置称为 yum 源。换句话说，yum 源指的就是软件安装包的来源。

使用 yum 安装软件时至少需要一个 yum 源。yum 源既可以使用网络 yum 源，也可以将本地光盘作为 yum 源。接下来就给大家介绍这两种 yum 源的搭建方式。

2.6 yum命令（查询、安装、升级和卸载软件包）

2.6.1 yum查询命令

使用 yum 对软件包执行查询操作，常用命令可分为以下几种：

- yum list：查询所有已安装和可安装的软件包。例如：

```
[root@localhost yum.repos.d]# yum list
#查询所有可用软件包列表
Installed Packages
#已经安装的软件包
ConsdeKit.i686 0.4.1-3.el6
@anaconda-CentOS-201207051201 J386/6.3
ConsdeKit-libs.i686 0.4.1-3.el6 @anaconda-CentOS-201207051201 J386/6.3
…省略部分输出…
Available Packages
#还可以安装的软件包
389-ds-base.i686 1.2.10.2-15.el6 c6-media
389-ds-base-devel.i686 1.2.10.2-15.el6 c6-media
#软件名 版本 所在位置（光盘）
…省略部分输出…
```

- yum list 包名：查询执行软件包的安装情况。例如：

```
[root@localhost yum.repos.d]# yum list samba
Available Packages samba.i686 3.5.10-125.el6 c6-media
#查询 samba 软件包的安装情况
```

- yum search 关键字：从 yum 源服务器上查找与关键字相关的所有软件包。例如：

```
[root@localhost yum.repos.d]# yum search samba
#搜索服务器上所有和samba相关的软件包
======N/S Matched:
samba =======
samba-client.i686：Samba client programs
samba-common.i686：Files used by both Samba servers and clients
samba-doc.i686: Documentation for the Samba suite
…省略部分输出…
Name and summary matches only, use"search all" for everything.
```

- yum info 包名：查询执行软件包的详细信息。例如：

```
[root@localhost yum.repos.d]# yum info samba
#查询samba软件包的信息
Available Packages <-没有安装
Name : samba <-包名
Arch : i686 <-适合的硬件平台
Version : 3.5.10 <-版本
Release : 125.el6 <-发布版本
Size : 4.9M <-大小
Repo : c6-media <-在光盘上
…省略部分输出…
```

2.6.2 yum安装命令

yum 安装软件包的命令基本格式为：

`[root@localhost yum.repos.d]# yum -y install 包名`

其中：

- install：表示安装软件包。
- -y：自动回答 yes。如果不加 -y，那么每个安装的软件都需要手工回答 yes；
  例如使用此 yum 命令安装 gcc：

```
[root@localhost yum jepos.d]#yum -y install gcc
#使用yum自动安装gcc
```

gcc 是 C 语言的编译器，鉴于该软件包涉及到的依赖包较多，建议使用 yum 命令安装。

2.6.3 yum 升级命令

使用 yum 升级软件包，需确保 yum 源服务器中软件包的版本比本机安装的软件包版本高。

yum 升级软件包常用命令如下：

- `yum -y update`：升级所有软件包。不过考虑到服务器强调稳定性，因此该命令并不常用。
- `yum -y update 包名`：升级特定的软件包。

2.6.4 yum 卸载命令

使用 yum 卸载软件包时，会同时卸载所有与该包有依赖关系的其他软件包，即便有依赖包属于系统运行必备文件，也会被 yum 无情卸载，带来的直接后果就是使系统崩溃。

除非你能确定卸载此包以及它的所有依赖包不会对系统产生影响，否则不要使用 yum 卸载软件包。

yum 卸载命令的基本格式如下：

```
[root@localhost yum.repos.d]# yum remove 包名
#卸载指定的软件包
```

例如，使用 yum 卸载 samba 软件包的命令如下：

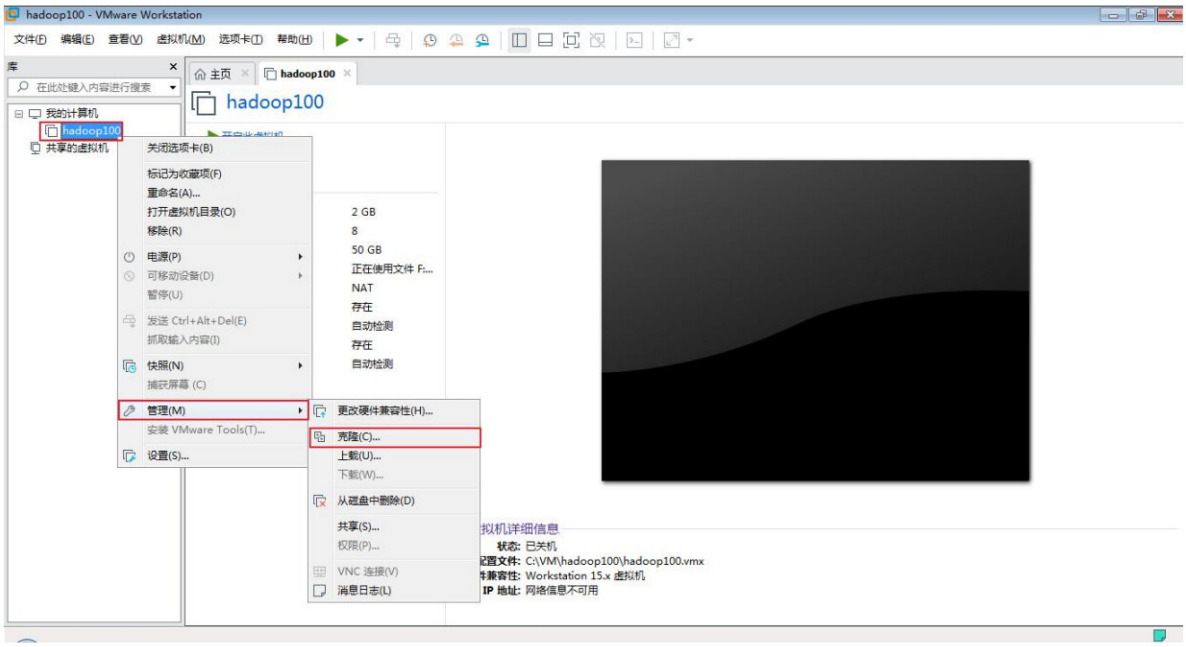
```
[root@localhost yum.repos.d]# yum remove samba
#卸载samba软件包
```

2.7 yum命令补充

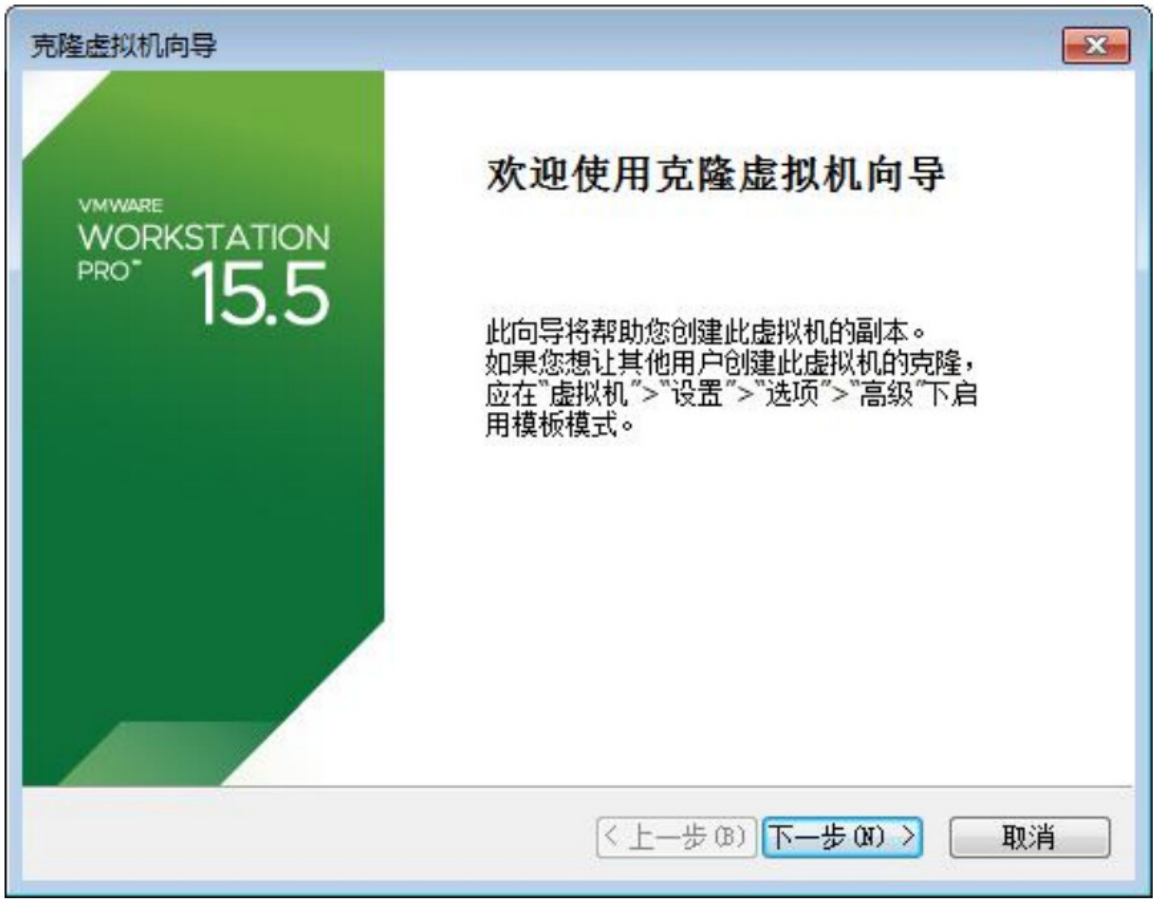
- check-update 检查是否有可用的更新 rpm 软件包
- clean 清理 yum 过期的缓存
- deplist 显示 yum 软件包的所有依赖关系

3.克隆虚拟机

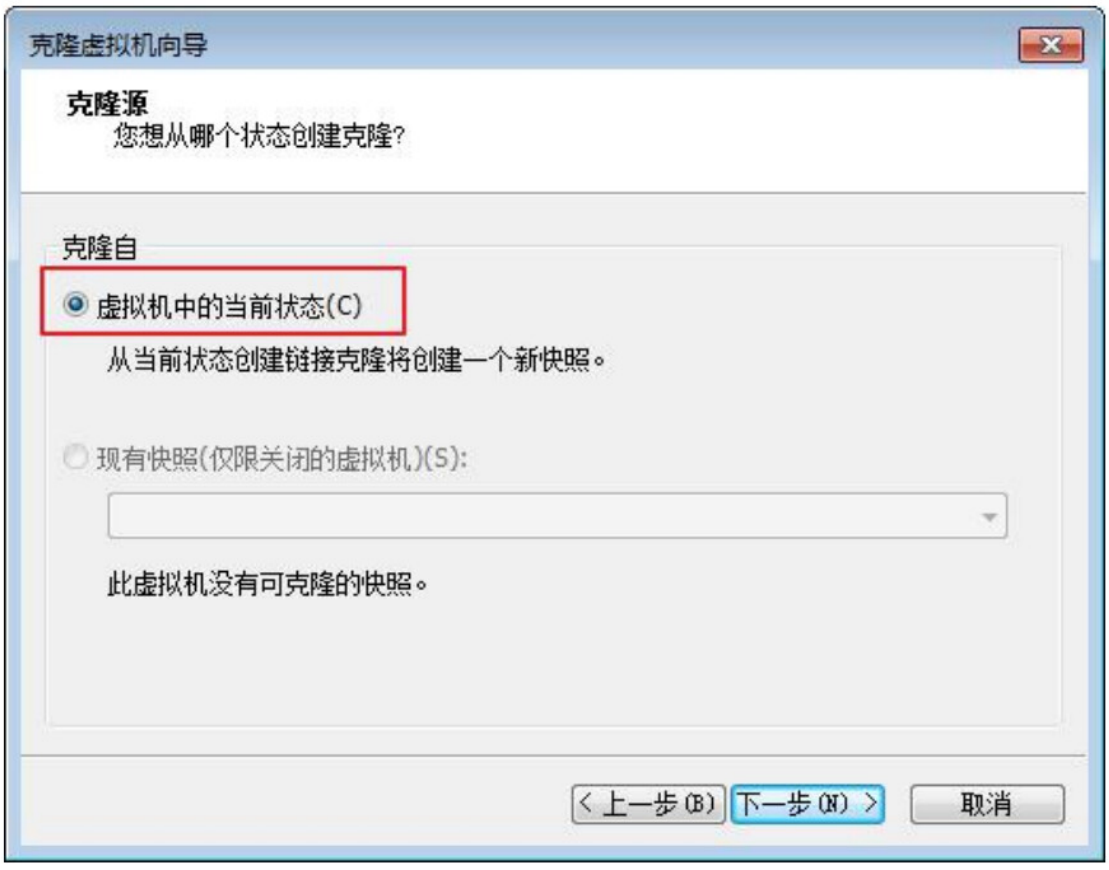
1）从现有虚拟机(关机状态)克隆出新虚拟机，右键选择管理=>克隆



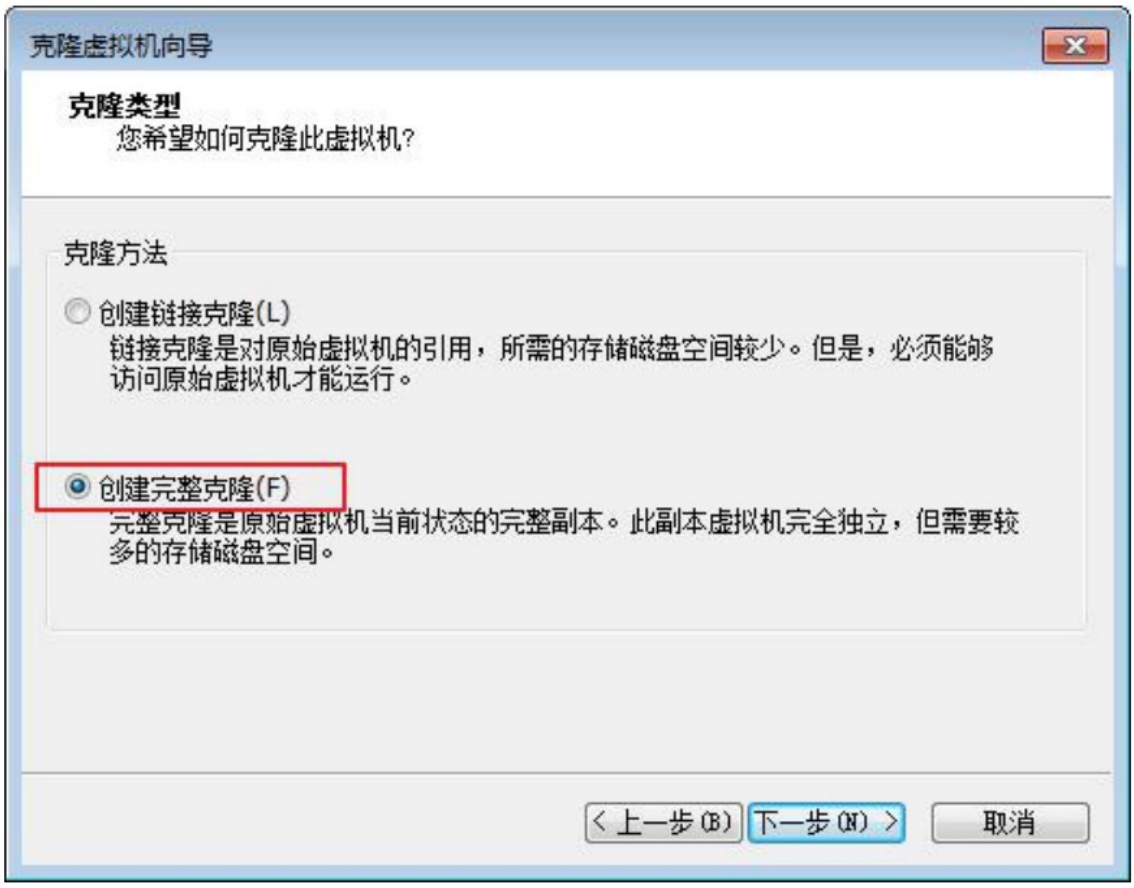
2）点击下一步



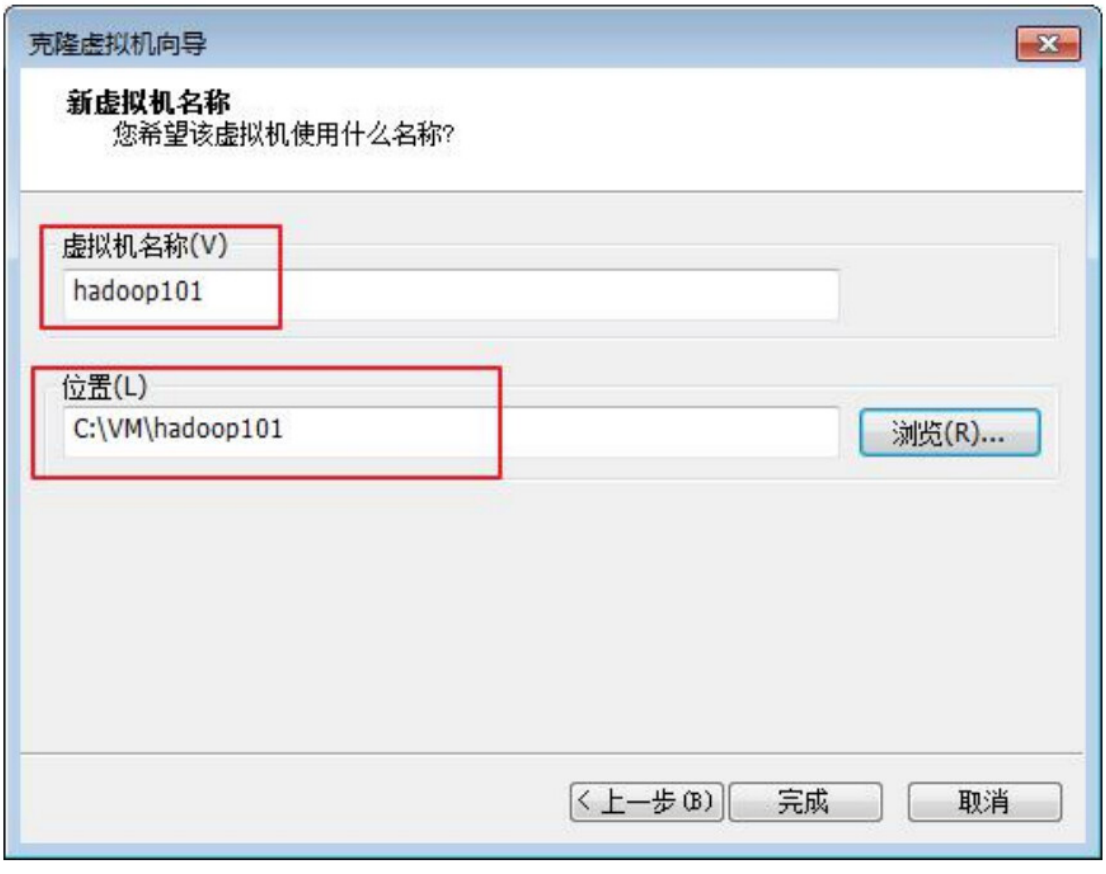
3）选择虚拟机中的当前状态



4）选择创建完整克隆



5）设置虚拟机名称及存储位置



6）等等等.....等待克隆完成