目录

[1. 常用文件管理命令 2](#_Toc100767078)

[1. 常用命令介绍 2](#_Toc100767079)

[2. tmux和vim 2](#_Toc100767080)

[1. tmux教程 2](#_Toc100767081)

[2. vim教程 3](#_Toc100767082)

[3. shell语法 5](#_Toc100767083)

[1.概论： 5](#_Toc100767084)

[2.注释： 6](#_Toc100767085)

[3.变量： 7](#_Toc100767086)

[4.默认变量： 8](#_Toc100767087)

[5.数组： 9](#_Toc100767088)

[6.expr命令： 10](#_Toc100767089)

[7.read命令 12](#_Toc100767090)

[8.echo命令 12](#_Toc100767091)

[9.printf命令 14](#_Toc100767092)

[10. test命令与判断符号[] 15](#_Toc100767093)

[11.判断语句 17](#_Toc100767094)

[12.循环语句： 21](#_Toc100767095)

[13.函数 23](#_Toc100767096)

[14.exit命令 25](#_Toc100767097)

[15.文件重定向 26](#_Toc100767098)

[16.引入外部脚本 27](#_Toc100767099)

[4. ssh 27](#_Toc100767100)

[ssh——ssh登录 28](#_Toc100767101)

[ssh——scp传文件 30](#_Toc100767102)

[5 .git 30](#_Toc100767103)

[1.git基本概念 31](#_Toc100767104)

[2.git常用命令 31](#_Toc100767105)

[6. thrift 32](#_Toc100767106)

[官网地址 32](#_Toc100767107)

[7. 管道、环境变量与常用命令 32](#_Toc100767108)

[1.管道 32](#_Toc100767109)

[2.环境变量 33](#_Toc100767110)

[3.常用命令 34](#_Toc100767111)

[8. 租云服务器及配docker环境 36](#_Toc100767112)

[阿里云 37](#_Toc100767113)

[docker教程 37](#_Toc100767114)

1. 常用文件管理命令

### 1. 常用命令介绍

(1) ctrl c: 取消命令，并且换行

(2) ctrl u: 清空本行命令

(3) tab键：可以补全命令和文件名，如果补全不了快速按两下tab键，可以显示备选选项

(4) ls: 列出当前目录下所有文件，蓝色的是文件夹，白色的是普通文件，绿色的是可执行文件

(5) pwd: 显示当前路径

(6) cd XXX: 进入XXX目录下, cd .. 返回上层目录

(7) cp XXX YYY: 将XXX文件复制成YYY，XXX和YYY可以是一个路径，比如../dir\_c/a.txt，表示上层目录下的dir\_c文件夹下的文件a.txt

(8) mkdir XXX: 创建目录XXX

(9) rm XXX: 删除普通文件; rm XXX -r: 删除文件夹

(10) mv XXX YYY: 将XXX文件移动到YYY，和cp命令一样，XXX和YYY可以是一个路径；重命名也是用这个命令

(11) touch XXX: 创建一个文件

(12) cat XXX: 展示文件XXX中的内容

(13) 复制文本

windows/Linux下：Ctrl + insert，Mac下：command + c

(14) 粘贴文本

windows/Linux下：Shift + insert，Mac下：command + v

2. tmux和vim

### 1. tmux教程

功能：

(1) 分屏。

(2) 允许断开Terminal连接后，继续运行进程。

结构：

一个tmux可以包含多个session，一个session可以包含多个window，一个window可以包含多个pane。

实例：

tmux:

session 0:

window 0:

pane 0

pane 1

pane 2

...

window 1

window 2

...

session 1

session 2

...

操作：

(1) tmux：新建一个session，其中包含一个window，window中包含一个pane，pane里打开了一个shell对话框。

(2) 按下Ctrl + a后手指松开，然后按%：将当前pane左右平分成两个pane。

(3) 按下Ctrl + a后手指松开，然后按"（注意是双引号"）：将当前pane上下平分成两个pane。

(4) Ctrl + d：关闭当前pane；如果当前window的所有pane均已关闭，则自动关闭window；如果当前session的所有window均已关闭，则自动关闭session。

(5) 鼠标点击可以选pane。

(6) 按下ctrl + a后手指松开，然后按方向键：选择相邻的pane。

(7) 鼠标拖动pane之间的分割线，可以调整分割线的位置。

(8) 按住ctrl + a的同时按方向键，可以调整pane之间分割线的位置。

(9) 按下ctrl + a后手指松开，然后按z：将当前pane全屏/取消全屏。

(10) 按下ctrl + a后手指松开，然后按d：挂起当前session。

(11) tmux a：打开之前挂起的session。

(12) 按下ctrl + a后手指松开，然后按s：选择其它session。

方向键 —— 上：选择上一项 session/window/pane

方向键 —— 下：选择下一项 session/window/pane

方向键 —— 右：展开当前项 session/window

方向键 —— 左：闭合当前项 session/window

(13) 按下Ctrl + a后手指松开，然后按c：在当前session中创建一个新的window。

(14) 按下Ctrl + a后手指松开，然后按w：选择其他window，操作方法与(12)完全相同。

(15) 按下Ctrl + a后手指松开，然后按PageUp：翻阅当前pane内的内容。

(16) 鼠标滚轮：翻阅当前pane内的内容。

(17) 在tmux中选中文本时，需要按住shift键。（仅支持Windows和Linux，不支持Mac，不过该操作并不是必须的，因此影响不大）

(18) tmux中复制/粘贴文本的通用方式：

(1) 按下Ctrl + a后松开手指，然后按[

(2) 用鼠标选中文本，被选中的文本会被自动复制到tmux的剪贴板

(3) 按下Ctrl + a后松开手指，然后按]，会将剪贴板中的内容粘贴到光标处

### 2. vim教程

功能：

(1) 命令行模式下的文本编辑器。

(2) 根据文件扩展名自动判别编程语言。支持代码缩进、代码高亮等功能。

(3) 使用方式：vim filename

如果已有该文件，则打开它。

如果没有该文件，则打开个一个新的文件，并命名为filename

模式：

(1) 一般命令模式

默认模式。命令输入方式：类似于打游戏放技能，按不同字符，即可进行不同操作。可以复制、粘贴、删除文本等。

(2) 编辑模式

在一般命令模式里按下i，会进入编辑模式。

按下ESC会退出编辑模式，返回到一般命令模式。

(3) 命令行模式

在一般命令模式里按下:/?三个字母中的任意一个，会进入命令行模式。命令行在最下面。

可以查找、替换、保存、退出、配置编辑器等。

操作：

(1) i：进入编辑模式

(2) ESC：进入一般命令模式

(3) h 或 左箭头键：光标向左移动一个字符

(4) j 或 向下箭头：光标向下移动一个字符

(5) k 或 向上箭头：光标向上移动一个字符

(6) l 或 向右箭头：光标向右移动一个字符

(7) n<Space>：n表示数字，按下数字后再按空格，光标会向右移动这一行的n个字符

(8) 0 或 功能键[Home]：光标移动到本行开头

(9) $ 或 功能键[End]：光标移动到本行末尾

(10) G：光标移动到最后一行

(11) :n 或 nG：n为数字，光标移动到第n行

(12) gg：光标移动到第一行，相当于1G

(13) n<Enter>：n为数字，光标向下移动n行

(14) /word：向光标之下寻找第一个值为word的字符串。

(15) ?word：向光标之上寻找第一个值为word的字符串。

(16) n：重复前一个查找操作

(17) N：反向重复前一个查找操作

(18) :n1,n2s/word1/word2/g：n1与n2为数字，在第n1行与n2行之间寻找word1这个字符串，并将该字符串替换为word2

(19) :1,$s/word1/word2/g：将全文的word1替换为word2

(20) :1,$s/word1/word2/gc：将全文的word1替换为word2，且在替换前要求用户确认。

(21) v：选中文本

(22) d：删除选中的文本

(23) dd: 删除当前行

(24) y：复制选中的文本

(25) yy: 复制当前行

(26) p: 将复制的数据在光标的下一行/下一个位置粘贴

(27) u：撤销

(28) Ctrl + r：取消撤销

(29) 大于号 >：将选中的文本整体向右缩进一次

(30) 小于号 <：将选中的文本整体向左缩进一次

(31) :w 保存

(32) :w! 强制保存

(33) :q 退出

(34) :q! 强制退出

(35) :wq 保存并退出

(36) :set paste 设置成粘贴模式，取消代码自动缩进

(37) :set nopaste 取消粘贴模式，开启代码自动缩进

(38) :set nu 显示行号

(39) :set nonu 隐藏行号

(40) gg=G：将全文代码格式化

(41) :noh 关闭查找关键词高亮

(42) Ctrl + q：当vim卡死时，可以取消当前正在执行的命令

异常处理：

每次用vim编辑文件时，会自动创建一个.filename.swp的临时文件。

如果打开某个文件时，该文件的swp文件已存在，则会报错。此时解决办法有两种：

(1) 找到正在打开该文件的程序，并退出

(2) 直接删掉该swp文件即可

3. shell语法

### 1.概论：

Linux系统中一般默认使用bash，所以接下来讲解bash中的语法。

文件开头需要写#! /bin/bash，指明bash为脚本解释器。



### 2.注释：

****

### 3.变量：

定义变量

定义变量，不需要加$符号，例如：

name1='yxc' # 单引号定义字符串

name2="yxc" # 双引号定义字符串

name3=yxc # 也可以不加引号，同样表示字符串

使用变量

使用变量，需要加上$符号，或者${}符号。花括号是可选的，主要为了帮助解释器识别变量边界。

name=yxc

echo $name # 输出yxc

echo ${name} # 输出yxc

echo ${name}acwing # 输出yxcacwing

只读变量

使用readonly或者declare可以将变量变为只读。

name=yxc

readonly name

declare -r name # 两种写法均可

name=abc # 会报错，因为此时name只读

删除变量

unset可以删除变量。

name=yxc

unset name

echo $name # 输出空行

变量类型

自定义变量（局部变量）

子进程不能访问的变量

环境变量（全局变量）

子进程可以访问的变量

自定义变量改成环境变量：

acs@9e0ebfcd82d7:~$ name=yxc # 定义变量

acs@9e0ebfcd82d7:~$ export name # 第一种方法

acs@9e0ebfcd82d7:~$ declare -x name # 第二种方法

环境变量改为自定义变量：

acs@9e0ebfcd82d7:~$ export name=yxc # 定义环境变量

acs@9e0ebfcd82d7:~$ declare +x name # 改为自定义变量

字符串

字符串可以用单引号，也可以用双引号，也可以不用引号。

单引号与双引号的区别：

单引号中的内容会原样输出，不会执行、不会取变量；

双引号中的内容可以执行、可以取变量；

name=yxc # 不用引号

echo 'hello, $name \"hh\"' # 单引号字符串，输出 hello, $name \"hh\"

echo "hello, $name \"hh\"" # 双引号字符串，输出 hello, yxc "hh"

获取字符串长度

name="yxc"

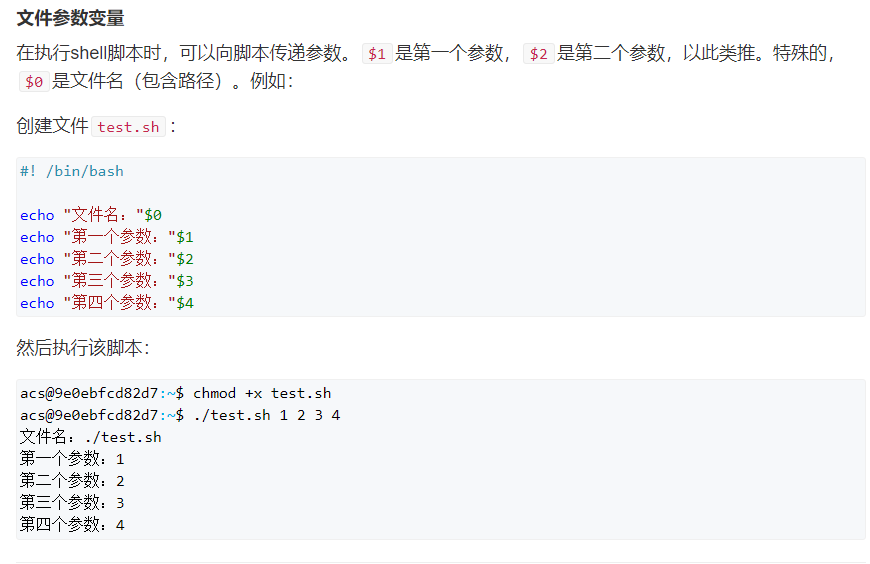
echo ${#name} # 输出3

提取子串

name="hello, yxc"

echo ${name:0:5} # 提取从0开始的5个字符

### 4.默认变量：

****

****

### 5.数组：

数组中可以存放多个不同类型的值，只支持一维数组，初始化时不需要指明数组大小。

数组下标从0开始。

定义

数组用小括号表示，元素之间用空格隔开。例如：

array=(1 abc "def" yxc)

也可以直接定义数组中某个元素的值：

array[0]=1

array[1]=abc

array[2]="def"

array[3]=yxc

读取数组中某个元素的值

格式：

${array[index]}

例如：

array=(1 abc "def" yxc)

echo ${array[0]}

echo ${array[1]}

echo ${array[2]}

echo ${array[3]}

读取整个数组

格式：

${array[@]} # 第一种写法

${array[\*]} # 第二种写法

例如：

array=(1 abc "def" yxc)

echo ${array[@]} # 第一种写法

echo ${array[\*]} # 第二种写法

数组长度

类似于字符串

${#array[@]} # 第一种写法

${#array[\*]} # 第二种写法

例如：

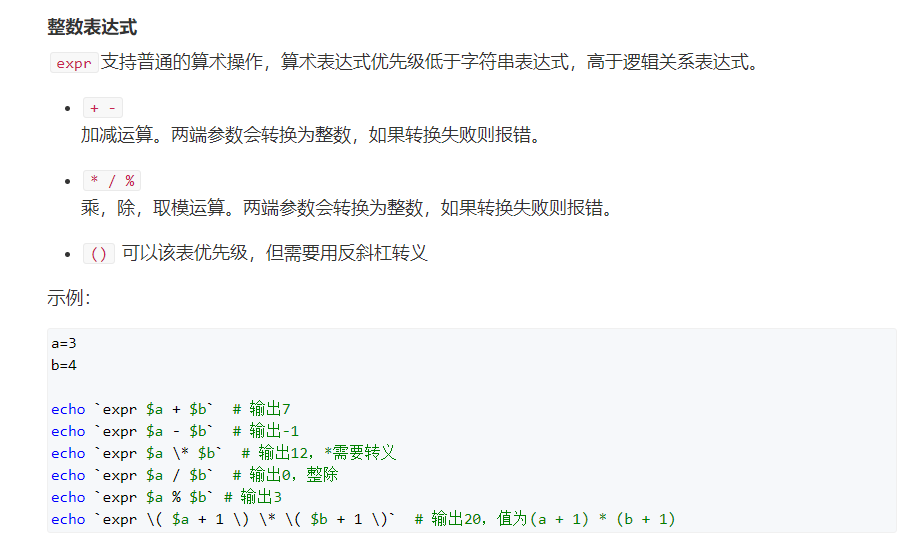
array=(1 abc "def" yxc)

echo ${#array[@]} # 第一种写法

echo ${#array[\*]} # 第二种写法

### 6.expr命令：

****

****

****

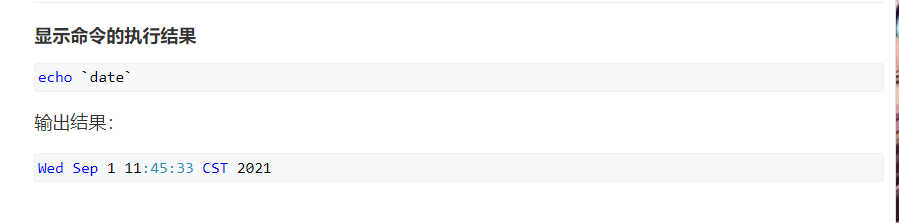
### 7.read命令

****

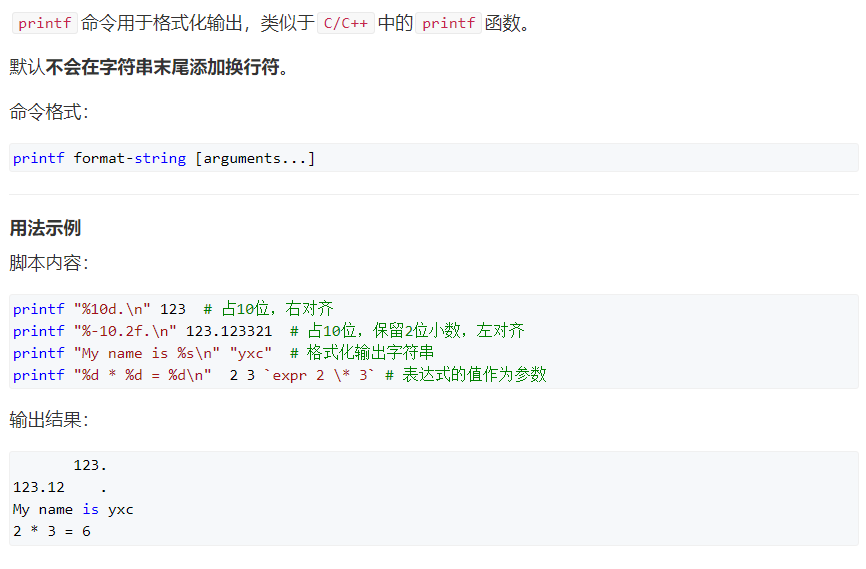
### 8.echo命令

****

****

****

### 9.printf命令

****

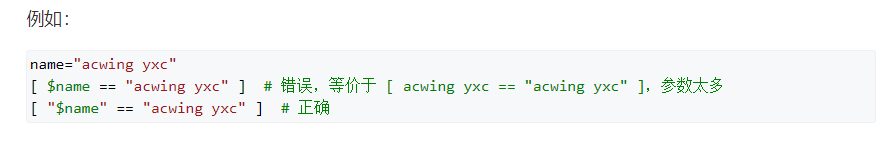
### 10. test命令与判断符号[]

****

****

****

****

****

### 11.判断语句

if…then形式

类似于C/C++中的if-else语句。

单层if

命令格式：

if condition

then

语句1

语句2

...

fi

示例：

a=3

b=4

if [ "$a" -lt "$b" ] && [ "$a" -gt 2 ]

then

echo ${a}在范围内

fi

输出结果：

3在范围内

单层if-else

命令格式

if condition

then

语句1

语句2

...

else

语句1

语句2

...

fi

示例：

a=3

b=4

if ! [ "$a" -lt "$b" ]

then

echo ${a}不小于${b}

else

echo ${a}小于${b}

fi

输出结果：

3小于4

多层if-elif-elif-else

命令格式

if condition

then

语句1

语句2

...

elif condition

then

语句1

语句2

...

elif condition

then

语句1

语句2

else

语句1

语句2

...

fi

示例：

a=4

if [ $a -eq 1 ]

then

echo ${a}等于1

elif [ $a -eq 2 ]

then

echo ${a}等于2

elif [ $a -eq 3 ]

then

echo ${a}等于3

else

echo 其他

fi

输出结果：

其他

case…esac形式

类似于C/C++中的switch语句。

命令格式

case $变量名称 in

值1)

语句1

语句2

...

;; # 类似于C/C++中的break

值2)

语句1

语句2

...

;;

\*) # 类似于C/C++中的default

语句1

语句2

...

;;

esac

示例：

a=4

case $a in

1)

echo ${a}等于1

;;

2)

echo ${a}等于2

;;

3)

echo ${a}等于3

;;

\*)

echo 其他

;;

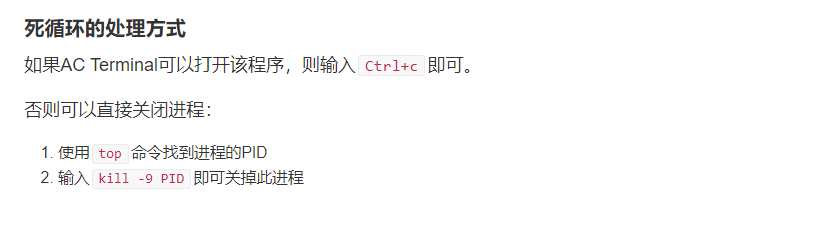
esac

### 12.循环语句：

****

****

****

****

### 13.函数

****

****

****

****

### 14.exit命令

****

### 15.文件重定向

****

****

### 16.引入外部脚本



4. ssh

### ssh——ssh登录







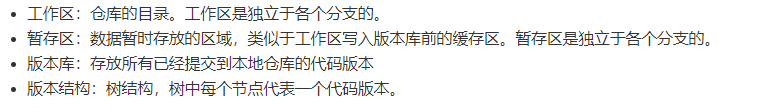
### ssh——scp传文件





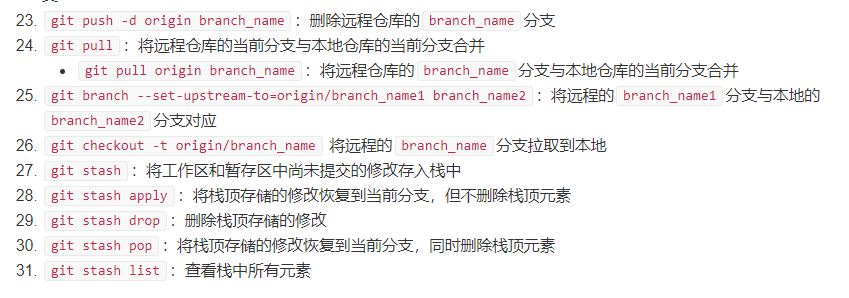
1. .git

### 1.git基本概念



### 2.git常用命令





6. thrift

### 官网地址

<https://thrift.apache.org/>

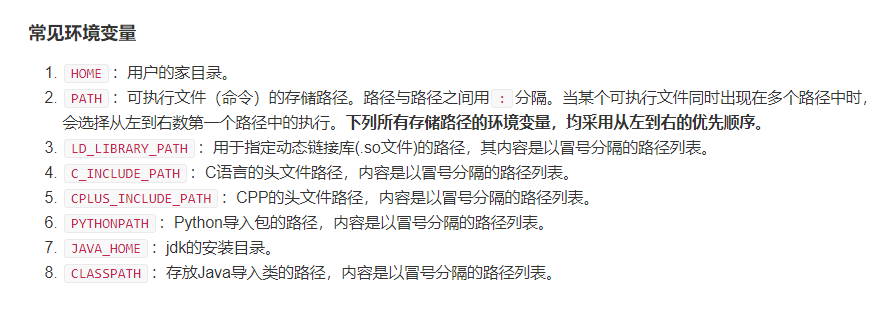
7. 管道、环境变量与常用命令

### 1.管道



### 2.环境变量

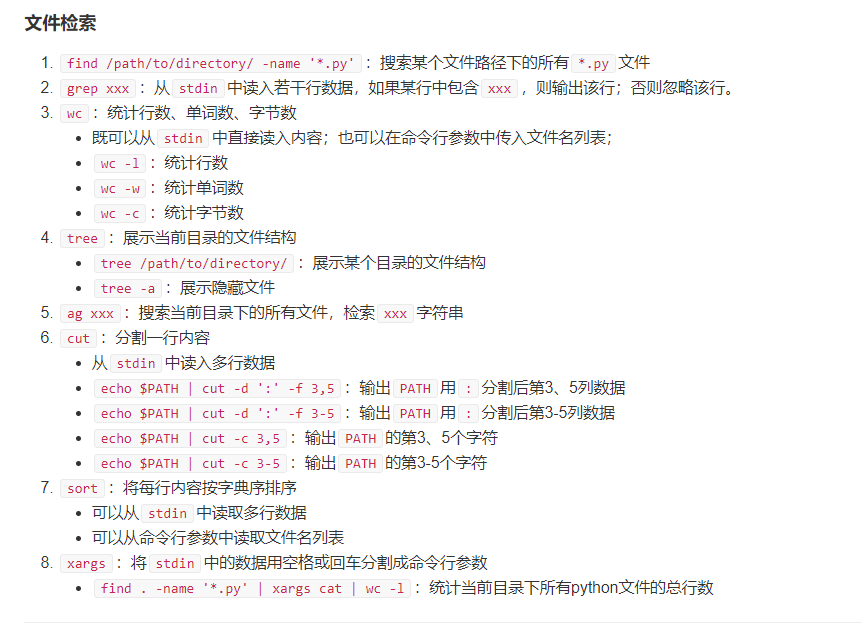




### 3.常用命令





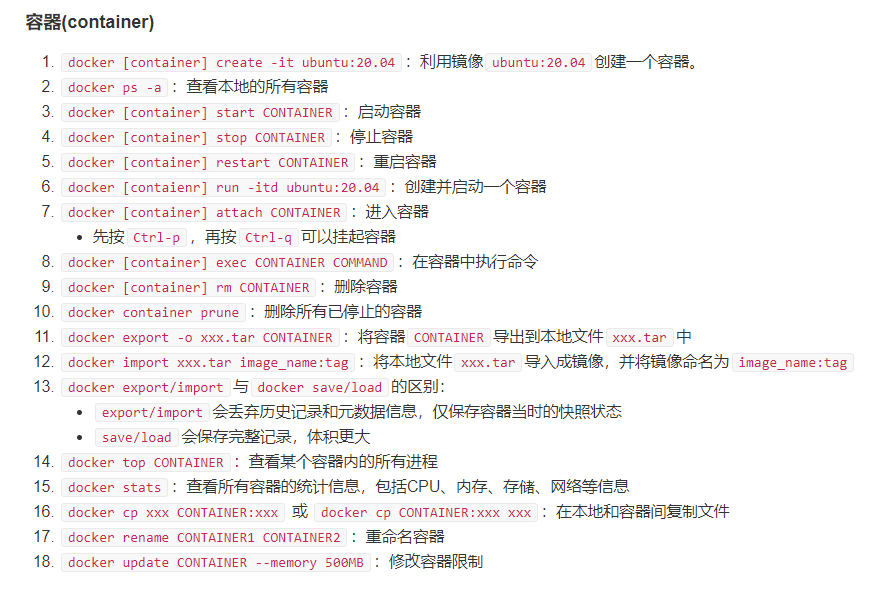


8. 租云服务器及配docker环境

### 阿里云



### docker教程



Docker官方文档：https://docs.docker.com/engine/install/linux-postinstall/