

默认bash

shell脚本执行

1. 手工
2. 脚本 /bin/bash test.sh

eg:#!/bin/bash

#shell脚本

echo “hello”

单行注释 #

多行注释 :<<! ... !

:<<字符 ... 字符

脚本执行方式:

1. /bin/bash test.sh

或bash /home/Desktop/test.sh

1. chmod u+x test.sh

/home/Desktop/test.sh

1. source test.sh

. test.sh

变量

1. 本地变量:当前系统某个环境下才生效
2. 普通变量 (echo $+变量名)

变量名=值 必须是整体,不能有特殊字符

变量名=’值’ 有什么输出什么

变量名=”值” 先解析其中变量,再输出

1. 命令变量

变量名=`命令` 先执行``中的命令,再赋值

变量名=$(命令) 先执行$()中的命令,再赋值

1. 全局变量:当前系统所有环境都生效

查看:env

定义:

1. 变量名=值

export 变量名

1. export 变量=值 (推荐)

变量查看

1.$变量名

2.”$变量名”

3.${变量名}

4. “${变量名}” 标准写法

取消变量

unset 变量名

内置变量

1. 脚本文件相关

$0: 获取当前脚本名称

$n: 执行当前脚本的第n个参数,n=1-9,n>9后写{10}

eg: bash testsh a b

#!/bin/bash

echo $1 ->a

$#: 获取参数的数量

$?: 获取执行上一条命令的状态返回值(0成功,非0失败)

字符串截取

格式: ${变量名:起始位置:截取长度} (倒数0-n)

eg:${str::5}

${str:0-3:3}

${str:5}

默认值

格式:${变量名:-默认值}

强制默认值

格式: ${变量名+默认值}

测试语句

1. test条件表达式
2. [ 条件表达式 ] 注意空格(常用)

条件成立返回0,不成立返回1

逻辑表达式

&& 同真则真

|| 有真则真

文件表达式

-f 判断输入内容是否是一个文件

# [ -f weizhi.sh ] && echo "是一个文件"

# [ -f weizhi.sddh ] || echo "不是一个文件"

-d 判断输入内容是否是一个目录

# [ -d weizhi.sddh ] || echo "不是一个目录"

# [ -d nihao ] && echo "是一个目录"

-x 判断输入内容是否可执行

# [ -x age.sh ] || echo "文件没有执行权限"

# [ -x test.sh ] && echo "文件有执行权限"

数值操作

n1 -eq n2 相等

n1 -gt n2 大于

n1 -lt n2 小于

n1 -ne n2 不等于

字符串比较

str1 == str2 str1和str2字符串内容一致

str1 != str2 str1和str2字符串内容不一致

计算表达式

1.$(( 计算表达式 )) (可以有空格,常用)

eg:$((1+1))

2.let 计算表达式(必须是整体)

eg:let a+1

Linux常见符号

重定向 > >>

>:左覆盖右

>>:左追加到右末行下一行

管道 |

命令1 | 命令2 :左结果给右使用

后台展示符 &

把一个命令从前台移到后台执行

其他符号

1 表示正确输出的信息

2 表示错误输出的信息

2>&1 代表所有输出的信息

bash test.sh 1>> all 2>>all

bash test.sh >> all 2>&1

linux系统垃圾桶

/dev/null 是linux下的一个设备文件,容量无限大

grep

grep [参数] [关键字] <文件名>

-c：只输出匹配行的计数。

-n：显示匹配行及行号。

-v：显示不包含匹配文本的所有行。

精确定位错误代码

grep -nr [错误关键字] \*

sed行文本编辑

格式：sed [参数] '<匹配条件> [动作]' [文件名]

参数为空 表示sed的操作效果，实际上不对文件进行编辑

-i 表示对文件进行编辑

注意：mac版本的bash中使用 -i参数，必须在后面单独加个东西： -i ''

'/关键字/'

注意：隔离符号 / 可以更换成 @、#、！等符号

替换：sed -i [替换格式] [文件名]

替换格式：'(行号)s#原内容#替换后内容#(列号)'

行号不写，所有行

列号不写，第一个

列号写g，所有列(所有个数)

增加: 指定行下一行: sed -i '行号**a**\增加的内容' 文件名

eg:1,3(指1-3行)

指定行前一行: sed -i '行号i\增加的内容' 文件名

删除: 删除指定行: sed -i '行号**d**' 文件名

eg:n,m(指n-m行)

awk文档编辑

格式: awk [参数] '[ 动作]' [文件名]

常见内置变量

FILENAME 当前输入文件的文件名，该变量是只读的

NR 指定显示行的行号

NF 输出 最后一列的内容

OFS 输出格式的列分隔符，缺省是空格

FS 输入文件的列分融符，缺省是连续的空格和Tab

print 显示内容

$0 显示当前行所有内容

$n 显示当前行的第n列内容，如果存在多个$n，它们之间使用逗号(,)隔开

eg: awk '{print $1}' awk.txt 打印第一列

awk 'NR==1 {print $1,$3}' awk.txt 打印第1行第1,3列

awk -F ':' '{print $1,$7}' linshi.txt 以:为分隔符打印第1,7列

awk 'BEGIN{OFS=":"} {print NR,$0}' awk.txt 设置分隔符为:打印一行并显示行号

find命令

find [路径] [参数] [关键字]

**参数详解**

-name 按照文件名查找文件。

-perm 按照文件权限来查找文件。

-user 按照文件属主来查找文件。

-group 按照文件所属的组来查找文件。

-type 查找某一类型的文件，

诸如：

b - 块设备文件 d - 目录 c - 字符设备文件

p - 管道文件 l - 符号链接文件 f - 普通文件。

-size n：[c] 查找文件长度为n块的文件，带有c时表示文件长度以字节计。

-depth：在查找文件时，首先查找当前目录中的文件，然后再在其子目录中查找。

-mindepth n：在查找文件时，查找当前目录中的第n层目录的文件，然后再在其子目录中查找。

! : 表示取反

eg: find /home/ -name "awk.txt" -type f

流程控制

单分支if

if [ 条件 ]

then

指令

fi

双分支

if [ 条件 ]

then

指令1

else

指令2

fi

多分支

if [ 条件 ]

then

指令1

elif [ 条件2 ]

then

指令2

else

指令3

fi

case语句

case 变量名 in

值1)

指令1

;;

值2)

指令2

;;

值3)

指令3

;;

\*)

指令4

;;

esac

for循环语句

for 值 in 列表

do

执行语句

done

while循环语句

while 条件

do

执行语句

done

until循环语句

until 条件

do

执行语句

done

函数定义

函数名(){

函数体

}

调用函数：

函数名

传参函数

函数名(){

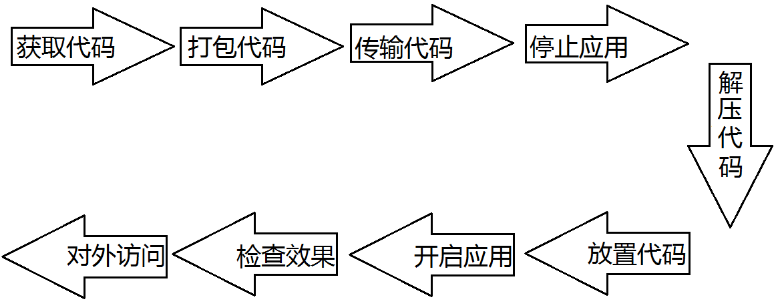
函数体 $n

}

调用函数:

函数名 参数

代码发布



文件的压缩

压缩格式：tar zcvf 压缩后的文件名 将要压缩的文件

文件的解压

解压格式：tar xf 压缩后的文件名

命令参数详解

z 指定压缩文件的格式为 tar.gz

c 表示压缩

v 显示详细过程

f 指定压缩文件

x 解压

查看压缩文件内容

zcat 压缩文件

文件传输

scp传输工具：

命令格式：

scp 要传输的文件 要放置的位置

将本地文件推送到远程主机

scp python.tar.gz root@192.168.8.15:/root/

将远程主机的文件拉取到本地

scp root@192.168.8.15:/root/python.tar.gz ./

文件的备份

date命令详解：

命令格式：date [option]

常见参数：

%F 显示当前日期格式，%Y-%m-%d

%T 显示当前时间格式，%H:%M:%S

eg: 年月日时分秒：date +%Y%m%d%H%M%S

备份格式

方式一：复制备份

cp file file-$(date +%Y%m%d%H%M%S)

方式二：移动备份(采用)

mv file file-$(date +%Y%m%d%H%M%S)

环境部署

批量创建基本目录,-p创建所有目录(父目录)

mkdir /data/{server,logs,backup,softs,virtual,scripts,codes} -p

ssh密钥认证

1、本机生成秘钥对

ssh-keygen -t rsa

-t:指定秘钥的类型 rsa:秘钥类型

秘钥目录：/root/.ssh/

私钥 id\_rsa 钥匙

公钥 id\_rsa.pub 锁

2、远端编辑认证文件

admin-1@ubuntu:~/.ssh# vim /root/.ssh/authorized\_keys

复制公钥文件内容 ssh-rsa AAAAB3N...PVp admin-1@ubuntu

3、远端编辑ssh配置文件

admin-1@ubuntu:~/.ssh# vim /etc/ssh/sshd\_config

...

AuthorizedKeysFile %h/.ssh/authorized\_keys

注意：直接取消该行的注释即可

4、远端配置文件生效

重启ssh服务

/etc/init.d/ssh restart

5、本机验证连接

ssh root@192.168.8.15

需求分析:

2、python环境 ---> 3、python虚拟环境

1、django环境部署

4、django软件安装

5、项目基本操作

6、应用基本操作

7、view和url配置

8、问题：只有本机能访问

9、方案代理---- 10、nginx

11、nginx实现代理

13、pcre软件安装

12、nginx软件安装

14、nginx基本操作

15、nginx代理的配置

16、目录结构

17、查看配置文件

18、找到对应的代理配置项

19、启动django

20、启动nginx

21、整个项目调试

环境部署方案

一、django环境部署

1.1 python虚拟环境

1.2 django环境部署

1.2.1 django软件安装

1.2.2 项目基本操作

1.2.3 应用基本操作

1.2.4 view和url配置

二、nginx代理django

2.1 nginx软件安装

2.1.1 pcre软件安装

2.1.2 nginx软件安装

2.1.3 nginx基本操作

2.2 nginx代理配置

2.2.1 目录结构查看

2.2.2 配置文件查看

2.2.3 编辑代理配置项

三、项目调试

3.1 启动软件

3.1.1 启动django

3.1.2 启动nginx

3.2 整个项目调试

项目部署:

1.安装虚拟环境软件

apt-get install python-virtualenv -y

创建(当前目录下)

virtualenv -p /usr/bin/python3.5 venv

进入

source venv/bin/activate

退出

deactivate

删除

rm -rf venv

(之前创建django虚拟环境mkvirtualenv django\_py3 -p python3)

2.django软件安装

先进入虚拟环境

解压

cd /data/soft

tar xf Django-1.10.7.tar.gz

查看

cd Django-1.10.7

cat INSTALL or README

安装

python setup.py install

拓展知识点：

python类型软件的安装流程

普通：

解压 安装

特殊：

解压 编译 安装

编译：python setup.py build

3.创建项目

创建项目

cd /data/server

django-admin startproject itcast

创建应用

cd /data/server/itcast

python manage.py startapp test1

注册应用

# vim itcast/settings.py

INSTALL\_APP = [

。。。

'test1',

]

view 配置文件生效

admin-1@ubuntu:/data/soft# cat /data/server/itcast/test1/views.py

from django.shortcuts import render

from django.http import HttpResponse

def hello(resquest):

return HttpResponse("itcast V1.0")

url文件配置

admin-1@ubuntu:/data/soft# cat /data/server/itcast/itcast/urls.py

...

from test1.views import \*

urlpatterns = [

url(r'^admin/', admin.site.urls),

url(r'^hello/$', hello),

]

启动django：

cd /data/server/itcast

python manage.py runserver

4.nginx环境

**pcre软件安装**

解压

cd /data/soft/

tar xf pcre-8.39.tar.gz

查看帮助

cd pcre-8.39

INSTALL 或者 README

配置

./configure

编译

make

安装

make install

**拓展知识点：**

linux中软件安装的一般流程

解压

tar

解压文件，获取真正的配置文件

配置

configure (–-prefix安装目录)

根据默认的配置项或者更改配置项，生成编译配置文件(Makefile)

编译

make

根据 Makefile 内容，编译生成指定的软件所需要的所有文件

安装

make install

将编译生成的所有文件，转移到软件指定安装的目录下面

nginx安装

解压

cd /data/soft/

tar xf nginx-1.10.2.tar.gz

配置

cd nginx-1.10.2/

./configure --prefix=/data/server/nginx --without-http\_gzip\_module

编译

make

安装

make install

nginx操作

检查

/data/server/nginx/sbin/nginx -t

开启

/data/server/nginx/sbin/nginx

关闭

/data/server/nginx/sbin/nginx -s stop

重载

/data/server/nginx/sbin/nginx -s reload

ldd 运行文件 查找运行程序所需模块

ngnix配置文件

全局配置段

http配置段

server配置段 项目或者应用

location配置段 url配置

配置实现

配置内容

62: location /hello/ {

63: proxy\_pass http://127.0.0.1:8000;

64: }

配置文件生效

/data/server/nginx/sbin/nginx -t

/data/server/nginx/sbin/nginx -s reload

5.手动代码发布

…

关闭django应用

根据端口查看进程号，

lsof -Pti :8000

杀死进程号

kill 56502

一条命令搞定它：

kill $(lsof -Pti :8000)

6.脚本发布代码

远程执行

ssh root@ip “执行命令” (所有路径使用绝对路径)

**6.1脚本框架:**

#!/bin/bash

# 功能：打包代码

# 脚本名：deploy.sh

# 作者：itcast

# 版本：V 0.1

# 联系方式：www.itcast.cn

# 获取代码

get\_code(){

echo "获取代码"

}

# 打包代码

tar\_code(){

echo "打包代码"

}

# 传输代码

scp\_code(){

echo "传输代码"

}

# 关闭应用

stop\_serv(){

echo "关闭应用"

echo "关闭nginx应用"

echo "关闭django应用"

}

# 解压代码

untar\_code(){

echo "解压代码"

}

# 放置代码

fangzhi\_code(){

echo "放置代码"

echo "备份老文件"

echo "放置新文件"

}

# 开启应用

start\_serv(){

echo "开启应用"

echo "开启django应用"

echo "开启nginx应用"

}

# 检查

check(){

echo "检查项目"

}

# 部署函数

deploy\_pro(){

get\_code

tar\_code

scp\_code

stop\_serv

untar\_code

fangzhi\_code

start\_serv

check

}

# 主函数

main(){

deploy\_pro

}

# 执行主函数

main

**6.2命令填充**

#!/bin/bash

# 功能：打包代码

# 脚本名：deploy.sh

# 作者：itcast

# 版本：V 0.2

# 联系方式：www.itcast.cn

# 获取代码

get\_code(){

echo "获取代码"

}

# 打包代码

tar\_code(){

echo "打包代码"

ssh root@192.168.8.15 "/bin/bash /data/scripts/tar\_code.sh"

}

# 传输代码

scp\_code(){

echo "传输代码"

cd /data/codes

[ -f django.tar.gz ] && rm -f django.tar.gz

scp root@192.168.8.15:/data/codes/django.tar.gz ./

}

# 关闭应用

stop\_serv(){

echo "关闭应用"

echo "关闭nginx应用"

/data/server/nginx/sbin/nginx -s stop

echo "关闭django应用"

kill $(lsof -Pti :8000)

}

# 解压代码

untar\_code(){

echo "解压代码"

cd /data/codes

tar xf django.tar.gz

}

# 放置代码

fangzhi\_code(){

echo "放置代码"

echo "备份老文件"

mv /data/server/itcast/test1/views.py /data/backup/views.py-$(date +%Y%m%d%H%M%S)

echo "放置新文件"

mv /data/codes/django/views.py /data/server/itcast/test1/

}

# 开启应用

start\_serv(){

echo "开启应用"

echo "开启django应用"

source /data/virtual/venv/bin/activate

cd /data/server/itcast/

python manage.py runserver >> /dev/null 2>&1 &

deactivate

echo "开启nginx应用"

/data/server/nginx/sbin/nginx

}

# 检查

check(){

echo "检查项目"

netstat -tnulp | grep ':80'

}

**6.3添加日志**

LOG\_FILE='/data/logs/deploy.log'

# 增加日志功能

write\_log(){

DATE=$(date +%F)

TIME=$(date +%T)

buzhou="$1"

echo "${DATE} ${TIME} $0 : ${buzhou}" >> "${LOG\_FILE}"

}

# 获取代码

get\_code(){

...

write\_log "获取代码"

}

# 打包代码

tar\_code(){

...

write\_log "打包代码"

}

# 传输代码

scp\_code(){

...

write\_log "传输代码"

}

# 关闭应用

stop\_serv(){

...

write\_log "关闭应用"

...

write\_log "关闭nginx应用"

...

write\_log "关闭django应用"

}

# 解压代码

untar\_code(){

...

write\_log "解压代码"

}

# 放置代码

fangzhi\_code(){

...

write\_log "放置代码"

...

write\_log "备份老文件"

...

write\_log "放置新文件"

}

# 开启应用

start\_serv(){

...

write\_log "开启应用"

...

write\_log "开启django应用"

...

write\_log "开启nginx应用"

}

# 检查

check(){

...

write\_log "检查项目"

}

**6.4增加锁文件**

#!/bin/bash

...

PID\_FILE='/tmp/deploy.pid'

...

# 增加锁文件功能

add\_lock(){

echo "增加锁文件"

touch "${PID\_FILE}"

write\_log "增加锁文件"

}

# 删除锁文件功能

del\_lock(){

echo "删除锁文件"

rm -f "${PID\_FILE}"

write\_log "删除锁文件"

}

# 部署函数

deploy\_pro(){

add\_lock

...

del\_lock

}

# 脚本报错信息

err\_msg(){

echo "脚本 $0 正在运行，请稍候..."

}

# 主函数

main(){

if [ -f "${PID\_FILE}" ]

then

err\_msg

else

deploy\_pro

fi

}

# 执行主函数

main

6.5

脚本调试功能

-n 检查脚本中的语法错误(常用)

-v 先显示脚本所有内容，然后执行脚本，结果输出，如果执行遇到错误，将错误输出。

-x 将执行的每一条命令和执行结果都打印出来(常用)