杂记

同步系统时间

```
mv /etc/localtime /etc/localtime.bak
ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
rm -f /etc/localtime.bak
```

设置DNS

```
mv /etc/resolv.conf /etc/resolv.conf.bak
echo "nameserver 10.0.14.14" >> /etc/resolv.conf
echo "nameserver 10.0.14.15" >> /etc/resolv.conf
```

安装jdk脚本

```
if grep -qc "JAVA_HOME" /etc/profile;then
   echo yes;
else
   wget http://172.16.1.188:88/base/jdk-7u55-linux-x64.tar.gz && tar -xvf jdk-7u55-linux-x64.tar.gz  -C /u01/
   echo 'export JAVA_HOME=/u01/jdk1.7.0_55' >> /etc/profile
   echo 'export PATH=$JAVA_HOME/bin:$PATH' >> /etc/profile
   echo 'export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar' >> /etc/profile
   source /etc/profile
   rm -f jdk-7u55-linux-x64.tar.gz
fi;
```

设置yum代理脚本

```
if grep -qc "proxy" /etc/yum.conf;then
   echo yes;
else
   echo "proxy=http://172.16.1.188:3128" >> /etc/yum.conf
   #Disable Base
   /usr/bin/yum-config-manager --disable base
   wget -O /etc/yum.repos.d/CentOS-Base.repo http://172.16.1.188:88/base/Centos-7.repo
   mkdir /u01/docker
   chmod 755 /u01/docker
fi;
```

安装docker脚本

更改docker目录不再使用sed的方式，改为使用ln -s软连接的方式 ln -s /home/docker/ /var/lib/docker

```
wget -O /etc/yum.repos.d/docker-main.repo http://172.16.1.188:88/base/docker-main.repo
```

yum -y install docker-engine device-mapper-event-libs(与lvm2冲突时 yum install libdevmapper* -y，然后安装docker，lvm2是虚拟机需要的，如果卸掉，重启虚拟机起不来)

```
#yum -y install docker device-mapper-event-libs

if [ $? = 0 ]; then
   ## Setting Docker Path
   sed -i "9c ExecStart=/usr/bin/docker daemon -g /u01/docker -H fd://" /usr/lib/systemd/system/docker.service
   #sed -i "4c OPTIONS=--selinux-enabled -g='/u01/docker'" /etc/sysconfig/docker


   ## Close Firewall and postfix
   /bin/systemctl stop firewalld
   /bin/systemctl stop iptables
   /bin/systemctl stop postfix
   /bin/systemctl disable firewalld
   /bin/systemctl disable iptables
   /bin/systemctl disable postfix


   ## Setting Start on system start
```

```
    chkconfig docker on
    /bin/systemctl start docker


    ## Load Images
    #rm -f *.tar
    #wget http://172.16.1.188:88/base/tomcat.tar
    #docker load < tomcat.tar
    #rm -f tomcat.tar
else
    echo 'docker not install'
fi
```

统计行数

```
grep -c "" filename
```

ping

```
cscript tping.vbs 172.26.1.78 -t > 78.log
nohup ping smtp.changan.com.cn | awk '{ print $0"\t" strftime("%H:%M:%S",systime()) } '>ping1.log &
```

系统资源监控

```
vmstat 2 | awk  '{print $1"\t"$(NF-6)"\t"$(NF-5)"\t"(100-$(NF-2))}'
```

查看redis版本

```
redis-cli -h 127.0.0.1 info
```

anywhere-user

1, 进入 web 项目, 下面有package.json bower.json gulpfile.js三个文件, package.json管理nodejs 的包(各种第三方组件), bower.json 管理我们项目所需要的第三方资源库(jquery, bootstrap等), gulpfile.js 基于nodejs语法, 编写任务, 静态资源下载, 提取, 编译, 打包, 压缩, 加密 等任务
2, 在web项目下 执行 npm install --save-dev
3, 在web项目下 执行 gulp build

logback

Since logging is initialized **before** the `ApplicationContext` is created, it isn't possible to control logging from `@PropertySources` in Spring`@Configuration` files. System properties and the conventional Spring Boot external configuration files work just fine.)

查会员人数

```
SELECT COUNT(*) FROM account_info a WHERE a.cyc_id IS NOT NULL OR a.syc_id IS NOT NULL
```

linux查询进程

```
ps -ef | grep tomcat
```

开发需关闭防火墙和一切杀毒软件
电脑管家等会拦截DUBBO的tcp 20880端口

使用DUBBO controller 调用service时一定要传可序列化的参数

DUBBO @service 有可能和aop冲突

webservice调用方式
1、通过wsdl文件生成客户端代码

```
ServiceLocator service = new ServiceLocator();

java.net.URL url = new new java.net.URL(wsdl地址);

ServiceStub stub = new ServiceStub(url, service);
```

2、直接调

```
JaxWsDynamicClientFactory dcf = JaxWsDynamicClientFactory.newInstance();
Client client = dcf.createClient("http://127.0.0.1:8080/services/wsservice?wsdl");
Object[] objects = client.invoke("getName", "123456789lll");
//输出调用结果
System.out.println(objects[0].getClass());
System.out.println(objects[0].toString());
```

apply plugin "spring-boot" 会去每个项目中找main函数，所以只需要在有main函数的项目里加就行了的

gradle 发布到私服 gradle -q publish

maven 上传到私服 mvn deploy 安装到本地仓库 mvn install

select2 如果使用input标签，默认不会选中，如果使用select标签会默认选中第一项

wsdl2java用法：

wsdl2java -p com -d src -all aa.wsdl

-p 指定其wsdl的命名空间，也就是要生成代码的包名:

-d 指定要产生代码所在目录

-client 生成客户端测试web service的代码

-server 生成服务器启动web service的代码

-impl 生成web service的实现代码

-ant 生成build.xml文件

-all 生成所有开始端点代码：types,service proxy,,service interface, server mainline, client mainline, implementation object, and an Ant build.xml file.

ex: wsdl2java -p com.changan.test -d D:\ -client http://183.230.8.209:8091/services/SmsChannel?wsdl

nginx -t 检测语法

nginx -s reload 不停服，重启nginx服务

linux 查找文件

find / -name aaa.aaa

cat smsandemail-log-2016-07-26.log | grep f965ebc0cbc04cfda3375cef125ba6f2 | awk -F 'mobile:' '{print $2}' | awk -F ' send' '{print $1}' >> mobile.txt

sort mobile.txt | uniq -c | awk '$1>=30 {print $0}' >> mobile-result.txt

sort -n mobile-result.txt >> mobile-result-n.txt

ehcahe timetoidleseconds(最后一次访问后闲置失效时间) timetoliveseconds(从缓存实例创建时开始的生命周期)

linux 定时任务

crontab -e 在最后添加 * * * * * .....
service crond restart

linux 磁盘使用情况

du查看目录大小，df查看磁盘使用情况。

我常使用的命令（必要时，sudo使用root权限），

1.查看某个目录的大小：du -hs /home/master/documents

查看目录下所有目录的大小并按大小降序排列：sudo du -sm /etc/* | sort -nr | less

2.查看磁盘使用情况（文件系统的使用情况）：sudo df -h

df --block-size=GB

-h是使输出结果更易于人类阅读；du -s只展示目录的使用总量（不分别展示各个子目录情况），-m是以MB为单位展示目录的大小（当然-k/-g就是KB/GB了）