目录 (五十八期·上)

Jenkins 系列(4)之数据迁移和备份

◆ 作者：合肥人真帅

无论从事什么行业，大家都希望工作的时候，工作环境安全、稳定，工作时可以高效的完成任务。但是，意外总是无法避免的。对于 IT 行业来说，停电，服务器宕机，断网，人为的误操作等，都会造成数据的丢失。

这些风险都是我们可以想象到的，但是却无法避免的。虽然无法避免，但也不能任人宰割。我们可以定期的备份数据，如果服务器坏了，还可以迁移数据到新的服务器上。

前三篇介绍了自动化部署、自动化接口测试、自动化 UI 测试

Jenkins 系列(1)之部署

Jenkins 系列(2)之 UI 测试

Jenkins 系列(3)之接口测试

里面安装了很多插件，创建了很多用户和项目，那么这些东西都是保存在哪里的？要如何去备份和迁移呢？

一、系统迁移

1、查看主目录

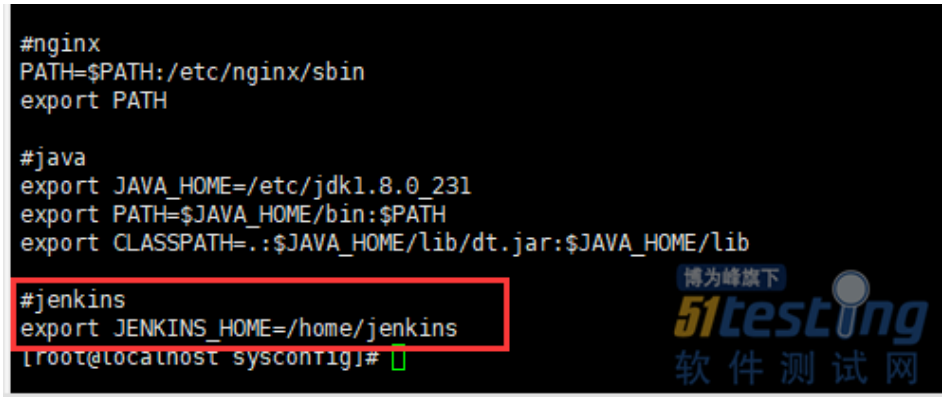
第一种：通过网站查看，登录 jenkins 网站，在系统管理->系统配置页面，可以查看到主目录地址

第二种：linux 服务器通过命令查看，可以使用 cat /etc/sysconfig/jenkins 命令查看主目录

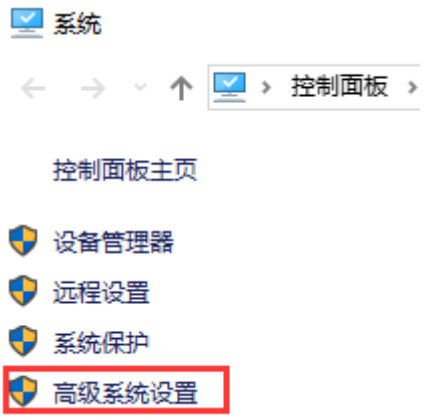
```
[root@localhost sysconfig]# cat /etc/sysconfig/jenkins
## Path:        Development/Jenkins
## Description: Jenkins Automation Server
## Type:        string
## Default:     "/var/lib/jenkins"
## ServiceRestart: jenkins
#
# Directory where Jenkins store its configuration and working
# files (checkouts, build reports, artifacts, ...).
#
JENKINS_HOME="/home/jenkins"

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
```

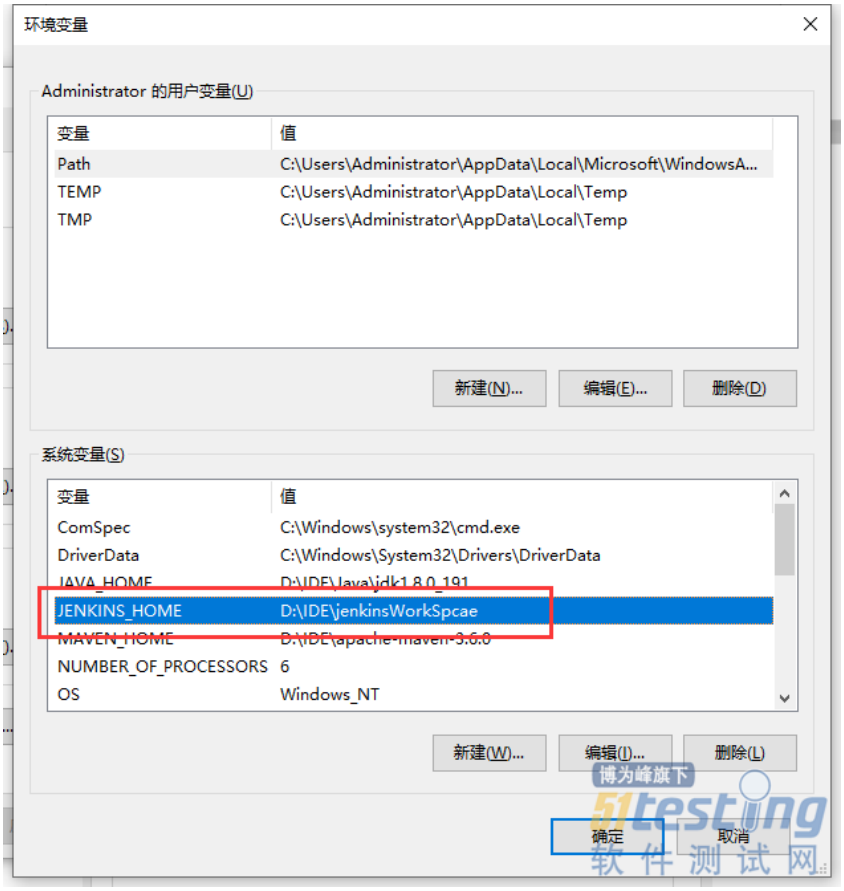
或者使用 cat /etc/profile 命令查看主目录



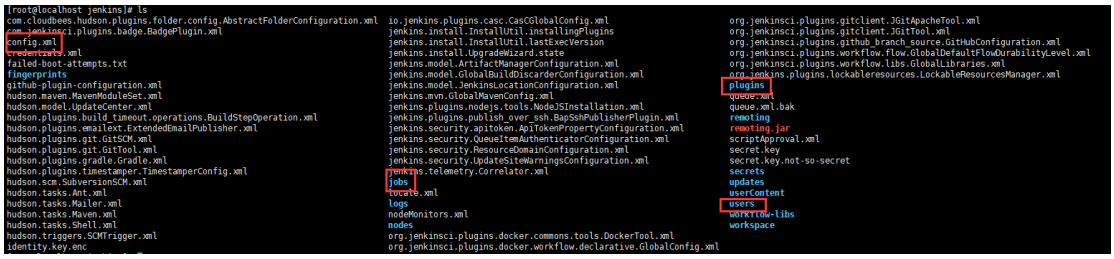
第三种：windows 服务器，此电脑属性->高级系统设置->环境变量->系统变量，找到 JENKINS_HOME，查看主目录




系统属性
计算机名
硬件
高级
系统保护
远程
要进行大多数更改，你必须作为管理员登录。
性能
视觉效果，处理器计划，内存使用，以及虚拟内存
设置(S)...
用户配置文件
与登录帐户相关的桌面设置
设置(E)...
启动和故障恢复
系统启动、系统故障和调试信息
设置(T)...
环境变量(N)...
确定
取消


环境变量
Administrator 的用户变量(U)
变量
值
Path
C:\Users\Administrator\AppData\Local\Microsoft\WindowsA...
TEMP
C:\Users\Administrator\AppData\Local\Temp
TMP
C:\Users\Administrator\AppData\Local\Temp
新建(N)...
编辑(E)...
删除(D)
系统变量(S)
变量
值
ComSpec
C:\Windows\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
JENKINS_HOME
D:\IDE\jenkinsWorkSpcae
NUMBER_OF_PROCESSORS
6
OS
Windows_NT
新建(W)...
编辑(I)...
删除(L)
确定
取消

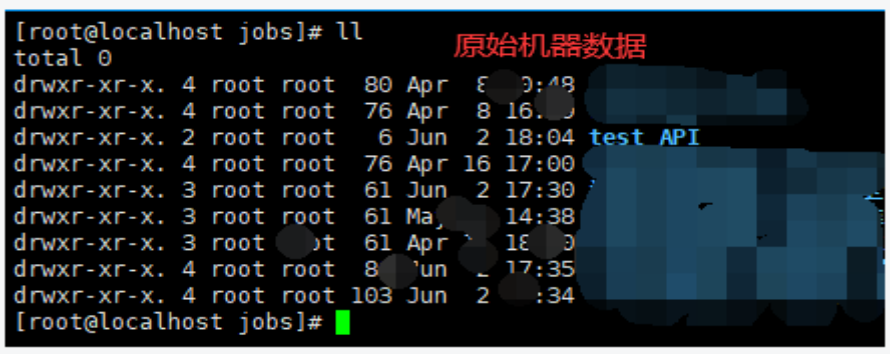
2、打包需要迁移的数据

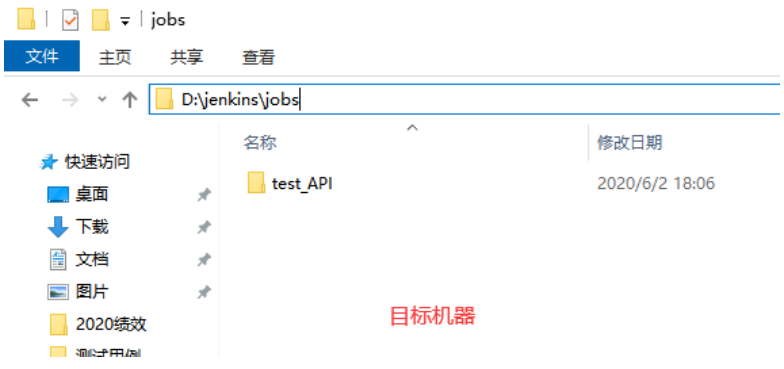


进入到主目录下，可以看到很多文件，是否需要全部打包呢？其实我们只需要打包 4 个文件就可以了，如上图所示分别是 config.xml 文件，jobs 文件夹，users 文件夹和 plugins 文件夹。从名称上就可以看出来各自的作用，config.xml 是存放配置信息的，jobs 是存放创建的工程项目的，users 是存放用户账信息的，plugins 是存放插件的。

3、将数据迁移到目标机器上

将这 4 个文件夹打包，拷贝到需要迁移的目标机器上，放到 jenkins 的主目录下。建议在打包和拷贝的时候，两台机器的 jenkins 都停止服务，防止打包不完整或拷贝不完整。如果迁移的目标机器没有安装 jenkins，那么只要在安装完 jenkins 后直接覆盖这 4 个文件夹就行了。如果目标机器已经安装 jenkins，且安装了部分插件，创建了用户和项目，那么有些文件就不能直接拷贝进去。例如需要迁移的 job 文件夹中有一个项目和目标机器中 job 文件夹里的项目同名，那么你就需要取舍了。





如果两个同名的项目都想保留。我建议从 web 端修改目标机器的项目名称，如果有其他 job 引用了这个项目或地址，也要一起改。然后再把备份文件放进目标机器的 job 中。其余几个文件夹也是同样的操作。操作完成后，启动目标机器 jenkins 服务就能读取到迁移过来的数据了。如果目标机器当前有重要的 job 在构建，不能停止服务。但是迁移过来的数据很重要，也想在最短的时间内恢复迁移项目的运行。也可以按照上述步骤进行操作，在当前 job 构建完成后，需要在 web 端的系统管理菜单下，点击读取设置即可。

二、数据备份

数据备份有两种办法，一种是手动的，一种是自动的。

1、手动备份

比较简单就像上述迁移步骤那样，把原始机器上的数据打包。打包后有两种选择，第一种是在原始机器上，其他路径下创建一个文件夹，把数据丢进去。例如原始机器上的数据是存储在/home/jenkins，我们打包后可以放到/home/backups，这样做的好处是如果误删了 jenkins，我们可以到 backups 下找回原始数据；第二种是将打包的文件拷贝到另外一台物理机上，这样做的好处是如果原始机器宕机了，我们可以在另外一台机器上找到备份文件，在最短的时间内恢复工作。

2、自动备份

相对于手动备份，我们也可以编写脚本实现自动备份，例如 linux 系统中，使用

shell 脚本，如下命令可以实现备份：

cp -r /home/jenkins/需要备份的文件夹名 /home/backups/目标文件夹名

注意：命令中的两个路径之间是有空格的。

同理 windows 下可以使用批处理：

xcopy D:\jenkins\需要备份的文件夹名*.* D:\backups\目标文件夹名\ /s /e

注意：*.*后面有一个空格，目标文件夹名后面有个斜杠不能丢（你若皮，可以丢一下试试看会有怎么样的效果）

那么跨平台备份文件应该怎么办呢？Linux 的文件传到 windows 上，windows 的文件传到 linux 上。这里我们需要借助一个工具 pscp，安装好以后就可以直接用了。

Windwos 传到 linux 命令如下：

pscp D:\jenkins root@192.168.0.15:/mnt/backups

其中 D:\jenkins 是 windows 文件夹，root 是用户名，@后面是 ip，冒号后面是 linux 的文件夹

从 linux 传到 windows 把命令倒过来就行了，如下

pscp root@192.168.0.15:/mnt/backups D:\jenkins

三、总结

备份这项工作，在项目正常运转的情况下会被人嫌弃，原因有费时间、占用存储空间等等。既然有种种缺点，备份这个工作还要不要做？我认为具体事情，具体对待。但要把握一个原则：在没有备份的情况下恢复生产所需要的时间远远大于有备份的情况下恢复生产所需要的时间，就需要备份。

小提示：时间久远的备份数据，记得要及时清理释放存储空间哦。

基于 BullseyeCoverage 工具的代码覆盖率研究

◆作者：党珊珊

摘要：代码覆盖率是反映测试用例对被测软件覆盖程度的重要指标，既可以用于单元测试，也可以用于黑盒测试。代码覆盖率并不能表明覆盖到的代码不包含缺陷，不能完全用来衡量代码质量。但是，它可以帮我们定位到没有被测试覆盖的代码，发现测试用例的薄弱部分，进而改善测试用例。本文将通过介绍 C/C++代码覆盖率测试工具 BullseyeCoverage 的实践，来说明一下代码覆盖率统计在测试工作中的作用。

软件公司对软件项目的期望是在预计的时间，合理的预算下，提交一个可以交付的产品。可以交付/发布的产品并不是没有错误的产品，而是要把错误控制在一个合理的范围之内。软件测试是为了寻找软件的错误与缺陷，评估与提高软件质量。软件测试只能证明软件存在错误，而不能证明软件没有错误。想要进行完全的测试，在有限的时间和资源条件下，找出所有的软件缺陷和错误，使软件趋于完美，是不现实的。一个适度规模的程序，其路径组合近似天文数字，对于每一种可能的路径都执行一次的穷举测试是不可能的。要根据测试错误的概率以及软件可靠性要求，确定最佳停止测试时间，我们不能无限地测试下去。

软件规模越来越大，功能越来越复杂，如何进行充分而有效的测试成为难题。在对一个软件产品进行了繁多的测试之后，我们能不能就此对软件的质量产生一定的信心呢？测试的代码覆盖率可以作为度量方式之一。虽然代码覆盖率并不能表明覆盖到的代码不包含缺陷。但是，如果测试仅覆盖了代码的一小部分，那么不管我们写了多少测试用例 ，我们也不能相信软件质量是有保证的。相反，如果测试覆盖到了软件的绝大部分代码，我们就能对软件的质量有一个合理的信心。我们可以通过代码覆盖率工具定位到未被测试覆盖的代码，发现测试的薄弱部分，补充测试用例，使我们的测试更加充分。

一、理论基础

代码覆盖（Code Coverage）是软件测试中的一种度量，描述程序中源代码被测试的比例和程度，所得比例称为代码覆盖率。衡量代码覆盖程度的指标有很多，常用的有函数覆盖，语句覆盖，判断覆盖，条件覆盖和路径覆盖。**下面简单介绍一下各覆盖率的含义：**

函数覆盖(Function Coverage)：执行到程序中的每一个函数。

语句覆盖(Statement Coverage)：又称行覆盖，就是度量被测代码中每个可执行语句是否被执行到了。语句覆盖常被人指责为“最弱的覆盖”，它只管覆盖代码中的执行语句，却不考虑各种分支的组合等。

判断覆盖(Decision Coverage)：又称分支覆盖(Branch Coverage)，基本路径覆盖（Basic Path Coverage）。它度量程序中每一个判定的分支是否都被测试到了。

条件覆盖(Condition Coverage)：它度量判定中的每个子表达式结果 true 和 false 是否被测试到了。条件覆盖不是将判定中的每个条件表达式的结果进行排列组合，而是只要每个条件表达式的结果 true 和 false 测试到了就好了。因此，我们可以推出：完全的条件覆盖并不能保证完全的判断覆盖。

路径覆盖(Path Coverage)：又称断言覆盖(Predicate Coverage)。它度量了是否函数的每一个分支都被执行了。有多个分支嵌套时，需要对多个分支进行排列组合。被认为是“最强的覆盖”

二、工具介绍

BullseyeCoverage 是 Bullseye 公司提供的一款 C/C++代码覆盖率测试工具。除了支持各种 Unix 下的编译器之外，在 Windows 下支持 VC, Borland C++, Gnu C++, Inter C++。提供的代码覆盖率是条件/分支覆盖率。

BullseyeCoverage 采用的是先对代码进行“插桩”，然后收集覆盖数据，最后分析覆盖率的技术，其工作原理是：针对不同的编译器，设计一个和真实编译器同名的拦截器（拦截器文件存放在 BullseyeCoverage 的 bin 目录下），当覆盖编译开关打开时，C/C++源文件在编译过程中将首先被这些拦截器拦截，由拦截器将一系列探针代码插入到源文件中，然后源文件再通过真实的编译器生成能够被 Bullseye Coverage 分析的二进制文件，当程序被执行时，代码覆盖的情况会被自动记录到指定的 cov 文件中去，通过一些命令能够从该文件中获取代码和条件/分支覆盖的情况。

代码覆盖率是一个白盒概念，那么是不是只有白盒测试才能统计代码覆盖率呢？使用 BullseyeCoverage 工具，即使没有源代码，也能统计出代码覆盖率。也就是说，功能测试或者黑盒测试同样能够统计代码覆盖率。但是如果要查看某个函数中具体条件/分支的执行情况，就必须在有代码的机器上进行分析。

BullseyeCoverage 的安装很简单，大部分地方只需根据默认进行安装。但要注意到 Coverage File Path 这一步时，根据需要更改 cov 文件的存放路径。

BullseyeCoverage 与 Visual Studio 的集成比较好。工具安装后，在 Visual Studio 工具菜单中将会增加 Enable /Disable BullseyeCoverage Build 选项。用 Visual Studio 编译程序前，勾选 Enable BullseyeCoverage Build 选项。编译完成后，运行测试程序，覆盖率就会自动记录到 cov 文件中，打开工具加载 cov 文件，就可以直观看到统计数据。覆盖率统计的结果还可以保存成 html 格式，在没安装工具的电脑上，也很方便查看。

如果要把编译完的程序，拷到其它测试机来运行和统计覆盖率，那么测试机也需要安装 BullseyeCoverage 工具，将编译时生成的 cov 文件拷到测试机安装 BullseyeCoverage 时指定的 cov 存放目录就好了。

对于比较复杂的项目，需要多人同时完成测试时，可以利用该工具的合并功能将每个人测得的 cov 文件进行合并，得到一个总的覆盖率文件。具体方法为：将各 cov 文件都放到 cov 存放目录，打开 cmd，输入 covmerge -c -f total.cov test1.cov test2.cov 命令，得到的 total.cov 就可以用来查看总的覆盖率。

三、实践应用

本文以视频车检器的配置软件测试为例，简单介绍一下 BullseyeCoverage 工具在测试 C++程序时的应用。

执行视频车检器的配置软件的测试之后，打开 BullseyeCoverage 工具，点击刷新图标，工具就会对测试的代码覆盖情况进行显示，其中蓝色部分代表已经覆盖到的代码，红色部分代表未覆盖的代码。代码覆盖率分两种级别，函数覆盖率和条件/分支覆盖率，可以分别对模块，文件进行统计。

本次实践中，对模块的覆盖率统计：

Name	Function cover...	Uncovered func...	Condition/decision...	Uncovered conditions/d...
HSClient	84%	46	63%	654

各文件的代码覆盖率统计：

Name	Function cover...	Uncovered func...	Condition/decision...	Uncovered conditions/d...
Service.cpp	0%	9	0%	36
APP_Send.cpp	0%	5	0%	6
BroadcastDlg.cpp	0%	4		0
Globe.h	0%	2		0
HistoryEdit.h	0%	2		0
CustomListCtrl.cpp	66%	4	55%	23
ModifyCal.cpp	80%	1	83%	1
SetNumDlg.cpp	83%	1	64%	5
SerialPortSetDlg.cpp	85%	1	75%	5
SearchDevice.cpp	87%	1	66%	2
RebootPatameter.cpp	87%	1	71%	4
CRecvFromThread.cpp	89%	5	57%	160
ClientDlg.cpp	90%	10	71%	231
Client.cpp	100%	0	50%	2
configure.cpp	100%	0	57%	151
QColiPropertyDlg.cpp	100%	0	62%	9
PropertyPresetBitDlg.cpp	100%	0	66%	2
Coil.cpp	100%	0	66%	1
RemoteIPList.cpp	100%	0	68%	7
QCoil.cpp	100%	0	70%	3
HistoryEdit.cpp	100%	0	75%	2
BlockInfo.cpp	100%	0	75%	2
PropertyDlg.cpp	100%	0	87%	2
CLogThread.cpp	100%	0		0
CollectCycleDlg.cpp	100%	0		0
LicenseKey.cpp	100%	0		0
PresettingBit.cpp	100%	0		0

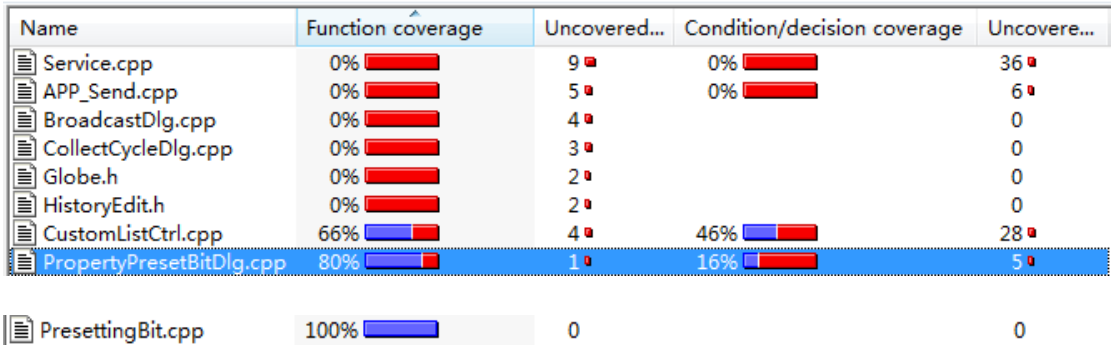
得到覆盖率后，如何用它来完善测试用例呢?

举一个简单的例子说明这个过程。

做完一轮测试后，发现如下几个文件没有执行到：

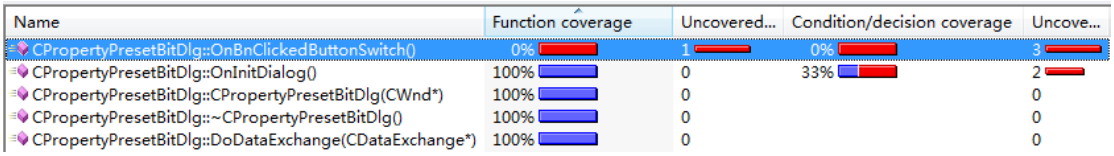
Name	Function coverage	Uncovered...	Condition/decision coverage	Uncovere...
Service.cpp	0%	9	0%	36
APP_Send.cpp	0%	5	0%	6
PropertyPresetBitDlg.cpp	0%	5	0%	6
BroadcastDlg.cpp	0%	4		0
CollectCycleDlg.cpp	0%	3		0
PresettingBit.cpp	0%	3		0
Globe.h	0%	2		0
HistoryEdit.h	0%	2		0

根据文件的命名和对产品功能的基本了解，由 PropertyPresetBitDlg.cpp 和 PresettingBit.cpp 两个文件，推测出是车检器的预置位漏测了。于是设置了预置位来进行验证。设置后的统计结果：



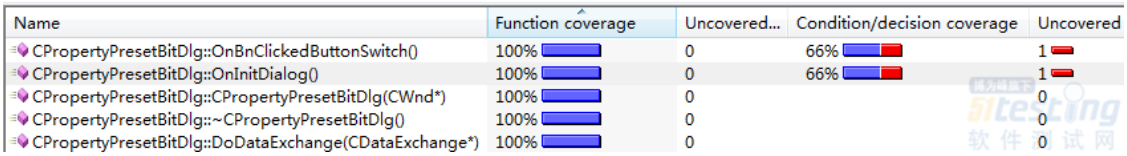
Name	Function coverage	Uncovered...	Condition/decision coverage	Uncovere...
Service.cpp	0%	9	0%	36
APP_Send.cpp	0%	5	0%	6
BroadcastDlg.cpp	0%	4		0
CollectCycleDlg.cpp	0%	3		0
Globe.h	0%	2		0
HistoryEdit.h	0%	2		0
CustomListCtrl.cpp	66%	4	46%	28
PropertyPresetBitDlg.cpp	80%	1	16%	5
PresettingBit.cpp	100%	0		0

可见 PresettingBit.cpp 已经达到了 100%，而 PropertyPresetBitDlg.cpp 函数覆盖率达到了 80%，条件/分支覆盖率只有 16%。我们可以打开文件分析下具体什么函数没被执行到。



Name	Function coverage	Uncovered...	Condition/decision coverage	Uncove...
CPropertyPresetBitDlg::OnBnClickedButtonSwitch()	0%	1	0%	3
CPropertyPresetBitDlg::OnInitDialog()	100%	0	33%	2
CPropertyPresetBitDlg::CPropertyPresetBitDlg(CWnd*)	100%	0		0
CPropertyPresetBitDlg::~CPropertyPresetBitDlg()	100%	0		0
CPropertyPresetBitDlg::DoDataExchange(CDataExchange*)	100%	0		0

由上图可见 OnBnClickedButtonSwitch()没有执行，推测是一个切换功能的按钮没有点击。补充切换按钮的测试后：



Name	Function coverage	Uncovered...	Condition/decision coverage	Uncovered
CPropertyPresetBitDlg::OnBnClickedButtonSwitch()	100%	0	66%	1
CPropertyPresetBitDlg::OnInitDialog()	100%	0	66%	1
CPropertyPresetBitDlg::CPropertyPresetBitDlg(CWnd*)	100%	0		0
CPropertyPresetBitDlg::~CPropertyPresetBitDlg()	100%	0		0
CPropertyPresetBitDlg::DoDataExchange(CDataExchange*)	100%	0		0

以上是文件和函数级别的覆盖率提高，就算没有代码，也可以做到。如果有代码的话，还可以看到函数中哪个条件/分支没有执行。**点击具体函数，查看函数实现，工具中有具体图标指示执行情况，在代码行标左边标出，图标含义为：**

√ 表示已执行过该函数；

→表示该函数（判定分支/表达式）未被执行。

→T 表示只执行过该判定分支为真的情况；

→F 表示只执行过该判定分支为假的情况；

TF 表示该判定分支的真假两种情况均被执行过；

→t 表示只执行过该表达式为真的情况；

→f 表示只执行过该表达式为假的情况；

tf 表示该表达式的真假两种情况均被执行过；

举例说明：

```
32 void DDV_MyMinMaxInt(CDataExchange* pDX,const int &Value,const int &minval,const int &maxval,const CString &ctrlname)
33 {
34     ASSERT(minval<=maxval);
35     if (pDX->m_bSaveAndValidate)
36     {
37a         if (
37b             Value<minval||
37c                         Value>maxval)
38         {
39             CString mesg;
40             mesg.Format(_T("%s%h%d%dQ%b"),ctrlname,minval,maxval);
41             ::AfxMessageBox(mesg);
42             pDX->Fail();
43         }
44     }
45 }
```

由上图可以看出，小于最小值或大于最大值的条件分支还没有执行过，即给定数值范围的无效值还没有测试过。补充该项测试后，该函数左边的图标发生改变，表明该分支已经被执行。

```
32 void DDV_MyMinMaxInt(CDataExchange* pDX,const int &Value,const int &minval,const int &maxval,const CString &ctrlname)
33 {
34     ASSERT(minval<=maxval);
35     if (pDX->m_bSaveAndValidate)
36     {
37a         if (
37b             Value<minval||
37c                         Value>maxval)
38         {
39             CString mesg;
40             mesg.Format(_T("%s%h%d%dQ%b"),ctrlname,minval,maxval);
41             ::AfxMessageBox(mesg);
42             pDX->Fail();
43         }
44     }
45 }
```

测试人员通过以上步骤对文件，函数，条件/分支的分析，对用例进行完善后，可能发现还是有一些文件没执行到，这时可以和研发讨论，这些文件是否有用，有用的话什么情况下会被执行。当然还可以进一步请教已执行文件中条件/分支覆盖率低的原因。本例中经过与研发确认，未被执行的文件中，除了 HistoryEdit.h，其他 4 个文件都没有用。对于与研发确认过，在工程中没用到又没有进行删除的文件，可以不计入统计。方法：选中文件，右键选择 Exclude. 被 Exclude 的文件上会有个“/”标记，想重新包含时，右键选择 include 即可恢复。

最终，本例中代码覆盖率的统计结果为：

Name	Function cover...	Uncovered func...	Condition/decision...	Uncovered conditions/d...
HSClient	91%	24	65%	612

四、作用及建议

1、得到代码覆盖率之后，对使用它的步骤进行归纳：

1） 根据测试人员对产品已有的了解，以及文件命名，函数命名来推测，未被执行

的文件和函数是什么功能相关的，然后执行这些功能。

2） 如果对代码有一定了解的话，可以打开具体某个函数，看一下条件/分支覆盖率是否还能再提高。

3） 由前面步骤提高一部分覆盖率后，如果还有未被执行的文件，函数的话，可以请教研发人员这些文件，函数是否确实有用，有用的话，在什么情况下会执行到，然后测试这些情况，使覆盖率进一步提高。对于确实没用的代码，建议研发进行删除。

通过对代码覆盖率的使用，分析出它的作用:

1） 把测试覆盖率作为质量目标的意义并不大，我们应该把它作为一种发现未被测试覆盖的代码的手段，进而完善测试用例。

2） 能检测出程序中的废代码，可以逆向反推在代码设计中的思维混乱点，提醒设计/开发人员理清代码逻辑关系，提升代码质量。

2、代码覆盖率统计的建议:

如果要统计完整的代码覆盖率，就需要完整的执行一遍测试用例。所以，代码覆盖率的统计是比较耗资源的一项工作，尤其是测试的自动化程度不高的话，代价会很高。既然代码覆盖率的意义主要是为了完善测试用例，那么就不需要在每次测试的时候都统计。在回归测试的时候统计的意义不大，在系统测试的初始阶段，和增加新功能的版本测试时，最好来统计一下，这样就可以尽早地定位到我们测试用例的薄弱环节，进而完善测试用例。

五、结束语

本文通过对测试覆盖率的分析及 BullseyeCoverage 工具的应用实例，阐述了代码覆盖率的统计方法及意义。白盒测试和黑盒测试都能统计代码覆盖率，区别是白盒能看到具体哪些条件/分支没被执行到。我们追求的不是 100%的代码覆盖率，因为代码覆盖率统计不能完全用来衡量代码质量，代码覆盖率统计是用来发现没有被测试覆盖的代码，通过分析代码覆盖率的统计情况，来发现测试中遗漏的部分，进而补充和完善测试用例，使测试更加充分，对交付的产品更加放心。

参考资料:

软件评测师教程 柳纯录主编 清华大学出版社
代码覆盖率工具 BullseyeCoverage 的应用研究 李雨江

Oracle 透明数据加密技术研究与实践

◆ 作者：颜廷义 刘丛伟 邢琳

近年来，随着信息技术的快速发展和互联网应用的普及，在给人带来便利的同时，也给个人信息安全带来了严重威胁。电信诈骗事件相继爆出，短信、电话骚扰等让人深恶痛绝，如何在确保数据安全的前提下最大化的使用数据带给我们的便利也被越来越多的提出。针对数据安全与保护问题，本文对 Oracle 透明数据加密技术（Oracle Transparent Data Encryption，以下简称 TDE）的原理进行研究，并将其应用于工作实践，希望对数据服务、数据安全、研发测试等人员有所帮助。

一、Oracle 透明数据加密技术（TDE）介绍

1、安装简易性

Oracle TDE 是 Oracle 自带的数据加密工具，无需安装任何补丁和插件即可使用，简单方便;

2、加密透明性

Oracle TDE 对数据的加解密过程完全透明，用户无感知;

3、算法多样性

支持 AES128、AES192、AES256、3DES168 等多种加密算法，默认使用 AES192 对数据进行加密。

4、功能多样性

在 Oracle 10G 版本中，主要支持对数据列加密，在 Oracle11G 后，增加了对表空间数据加密功能，功能更强大。

二、Oracle 透明数据加密技术（TDE）原理说明

Oracle TDE 加密主要涉及三个概念:

1、wallet（钱包）。需自行建立并设置钱包密码，主要用于存放 TDE Master Key。

2、TDE Master Key。存放于 wallet 中，主要用于对数据加密秘钥进行加解密;

3、数据加密秘钥。用于对具体数据进行加解密，TDE Master Key 对其进行加密后存储在数据字典（对于加密表空间，存放在对应表空间）中。

数据插入时，首先通过密码打开 wallet 钱包，获取到 TDE Master Key，之后使用 TDE Master Key 对存放在数据字典（或加密表空间）中已加密的数据加密秘钥进行解密，进而得到数据加密秘钥，进而对数据进行加密，并将加密后数据存储在数据文件中;

数据读取时，同样通过密码打开 wallet 钱包，获取到 TDE Master Key，之后使用 TDE Master Key 对存放在数据字典（或加密表空间）中已加密的数据加密秘钥进行解密，进而得到数据加密秘钥，之后对从磁盘读取的加密数据进行解密，反馈给用户。具体流程图如下:

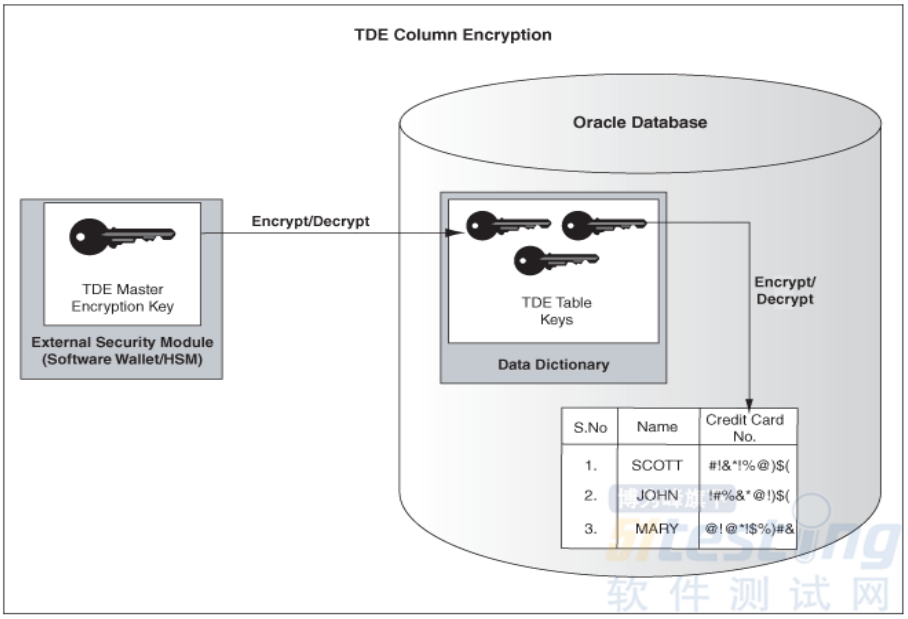


图 1　TDE 列加密

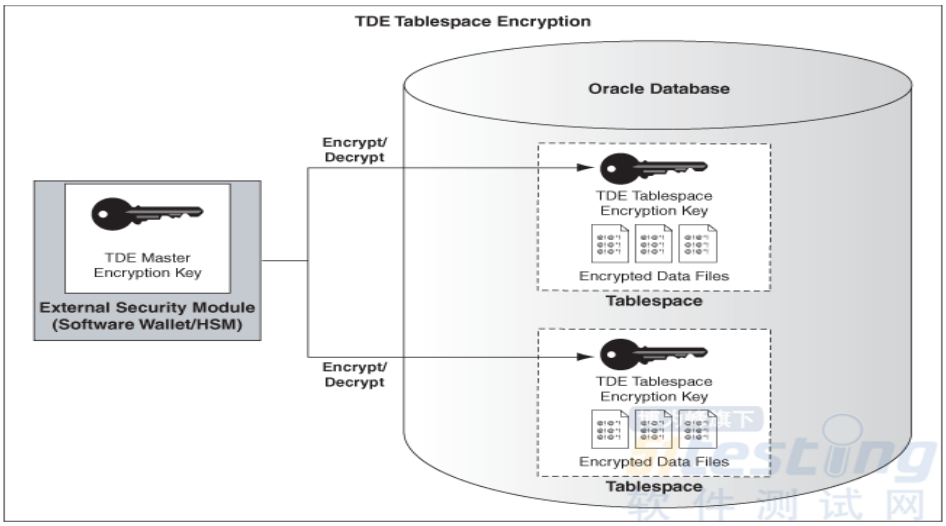


图 2　TDE 表空间加密

从分析可以看到:

1、数据的加密与解密过程对于用户完全透明，用户需要做的只是使用 wallet 密码打开钱包，之后数据的加解密会自动完成。

2、通过 TDE 加密，即使保存在磁盘上的数据或备份被盗，由于 Master key 并没有被盗，在没有 Master Key 的情况下，数据无法被获取。即使“钱包（wallet）”被盗，如果没有钱包密码，Master Key 密钥还是无法获取。因此，即使磁盘或者数据文件被未经授权的拷贝和泄露，也不会造成数据的泄露。

三、Oracle 透明数据加密技术（TDE）技术实践

1、环境准备阶段

1.1 测试环境说明:

本次测试使用 Suse Linux 11 SP4+ Oracle 12.0.1.2 环境。

1.2 创建 wallet(钱包)目录以及指定位置

Oracle 数据库中 wallet 目录默认存放在：$ORACLE_BASE/admin/$ORACLE_SID/下（若在对应目录下没有 wallet 目录，可以自行建立）。如果想修改 wallet 默认目录，可以在位于$ORACLE_HOME/network/admin 目录下的 sqlnet.ora（如果没有则创建）文件中以指定的方式选择一个不同的目录。例如：如果要把 wallet 放在/data 目录下，在

sqlnet.ora 文件中写入如下内容:

```
ENCRYPTION_wallet_LOCATION =
(SOURCE=
(METHOD=file)
(METHOD_DATA=
(DIRECTORY=/data)))
```

1.3 设置 wallet（钱包）密码

使用具有 alter system 权限的用户执行如下语句:

SQL>alter system set encryption key identified by "password";----password 即为 wallet 密码。

执行完成后将在相应的 wallet 目录下生成密码文件。

1.4 wallet（钱包）操作介绍

SQL>alter system set wallet open identified by "password";打开钱包。
SQL>alter system set wallet close identified by "password";关闭钱包。

2、测试验证阶段

2.1 Oracle TDE 列加密

Oracle 支持对以下列进行 TDE 加密:

BINARY_DOUBLE • BINARY_FLOAT •CHAR • DATE • INTERVAL DAY TO SECOND • INTERVAL YEAR TO MONTH • LOBs (Internal LOBs and SECUREFILE LOBs Only) • NCHAR • NUMBER • NVARCHAR2 • RAW • TIMESTAMP (includes TIMESTAMP WITH TIME ZONE and TIMESTAMP WITH LOCAL TIME ZONE) • VARCHAR2

2.1.1 打开 wallet 钱包:

SQL>alter system set wallet open identified by "password";

2.1.2 建表并插入语句:

```
SQL>CREATE TABLE employee (
        name VARCHAR2(128) ENCRYPT,     name 为加密列
        empID NUMBER,
        salary NUMBER(6)
);
SQL> insert into employee values ('jack',1,8000);
SQL> commit;
```

2.1.3 查询结果:

```
SQL>select * from employee;
NAME                 EMPID          SALARY
jack                 1              8000
```

关闭 wallet 再次查询

```
SQL>alter system set wallet close identified by "password";
SQL>select * from employee;
ORA-28365:wallet is not open
```

尝试查询未加密列

```
SQL>select empid，salary from employee;
EMPID          SALARY
1              8000
```

钱包关闭情况下插入数据:

```
SQL> insert into employee values ('jack',1,8000);
ORA-28365:wallet is not open
```

可以看到，只有在 wallet 钱包开启的情况下才可以对加密列进行数据的插入与查看；没有开启时，对加密列的操作都将会得到“wallet is not open”的错误；另外，对未加密字段的操作不受钱包是否开启的影响。

2.1.4 列加密修改方式说明

1、表已存在情况下增加加密列:

```
SQL> ALTER TABLE employee ADD (address VARCHAR2(11) ENCRYPT);
```

2、将表中未加密列修改为加密列（表中有数据情况下需打开钱包）

```
SQL> ALTER TABLE employee MODIFY (name ENCRYPT);
```

将表中加密列修改为未加密列（表中有数据情况下需打开钱包）

```
SQL> ALTER TABLE employee MODIFY (name DECRYPT);
```

修改列加密算法（一个表的各个字段中只支持一种加密算法）

```
SQL>alter table employee MODIFY (name ENCRYPT using 'AES128');
```

2.2 表空间加密

Oracle 11G 版本中增加了表空间数据加密功能，在加密表空间内创建的表都将具备加密特性。

2.2.1 创建加密表空间

```
SQL> CREATE TABLESPACE test
DATAFILE '/u01/app/oracle/oracledata/test.dbf'
SIZE 50M
ENCRYPTION
DEFAULT STORAGE(ENCRYPT);
```

2.2.2 在加密表空间中创建表

```
SQL>CREATE TABLE emp (
      id              NUMBER(10),
      name            VARCHAR2(128),
      salary          NUMBER(6) )
TABLESPACE test;
```

2.2.3 开启 wallet 钱包并插入数据

```
SQL>alter system set wallet open identified by "password";
SQL> insert into emp values (1,'tom',8888);
SQL> commit;
```

2.2.4 查询数据

```
SQL> select * from emp;
ID                    NAME              SALARY
1                      tom               8888
```

关闭 wallet 再次查询

```
SQL>alter system set wallet close identified by "password";
SQL> select * from emp;
ORA-28365:wallet is not open
```

由此可见，在加密表空间内创建的表已经被加密，必须显式打开 wallet 查询语句才能查询结果。

2.3 Oracle TDE 数据泵导入导出测试

数据泵是 Oracle 数据库中常用的数据操作工具，主要用于数据的应用级备份与迁移等，以下就 TDE 技术在数据泵中的测试说明如下：

2.3.1 wallet 钱包开启情况下导出、导入数据

我们尝试对 2.2 节中建立的 test 加密表空间下的 emp 表进行测试（数据泵的具体操作在此不在详细说明）:

首先打开 wallet 钱包

SQL>alter system set wallet open identified by "password";

导出 emp 表中的数据

$expdp test/****** directory=test dumpfile=test.dmp tables=emp

之后将 test.dmp 文件拷贝到其他 Oracle 环境中并进行导入

impdp test/****** directory=test dumpfile=test.dmp tables=emp

通过查询可以发现，数据被正常导入。

2.3.2 wallet 钱包关闭情况下导出、导入数据

首先关闭 wallet 钱包

SQL>alter system set wallet close identified by "password";

导出 emp 表中的数据

$expdp test/****** directory=test dumpfile=test.dmp tables=emp

导入到其他数据库中

$impdp test/****** directory=test dumpfile=test.dmp tables=emp

发现数据没有被正常导入。这也说明在 wallet 钱包未打开情况下，即使数据被导出，由于没有 wallet 密码进而没有数据加密秘钥，数据也不会泄露。

2.3.3 DUMP 文件添加密码参数

Oralce 数据泵还支持密码参数，如果需要对 dump 文件加密，只需要添加 ENCRYPTION_PASSWORD 参数即可。

$expdp test/****** directory=test dumpfile=test.dmp ENCRYPTION_PASSWORD=password

在进行数据导入时也需要加上相应参数

$impdp test/****** directory=test dumpfile=test.dmp ENCRYPTION_PASSWORD=password

如果不加该参数，将报 ORA39174:Encryption password must be supplyed 错误。

该方法对于 DUMP 文件的保护非常有用，可用性更高。即使源库没有配置 TDE 加密，也可以在数据导出时使用 ENCRYPTION_PASSWORD 对 DUMP 文件进行保护，这

样即使 DUMP 数据在传输或存储时被窃取，由于没有对应的密码，数据也不会导入并泄露，这对于数据服务人员加强交付数据的安全性起到很好的效果。

关于验证 TDE 是否对数据文件进行加密存储，读者可使用 LogMinner 日志挖掘工具进行验证，在此不在演示。

四、总结

Oracle TDE 作为 Oracle 数据库自带的数据加密技术，对数据加密后存储在数据或备份文件中，这样即使数据文件或磁盘被窃取，也不会造成数据的泄露，对数据起到了很好的保护作用。当然，任何技术都有其局限性，由于存在加密存储、解密显示等过程，数据库的效率、索引等性能问题也是需要考虑的因素（作者对包含 40 万行数据表的某一字段进行加密测试，加密时间在 8 秒左右），这就需要根据具体需求与特点进行综合考虑，在安全性与性能之间找到平衡；对于数据支持人员，在进行数据交付时可以通过对 DUMP 文件添加密码参数，进而对数据进行保护，防止数据被未经授权人员的窃取与使用，给数据加上一把锁，大家都放心！

参考文献

1、Database Advanced Security Administrator’ s Guide
2、Oracle 透明数据加密常见问题解答

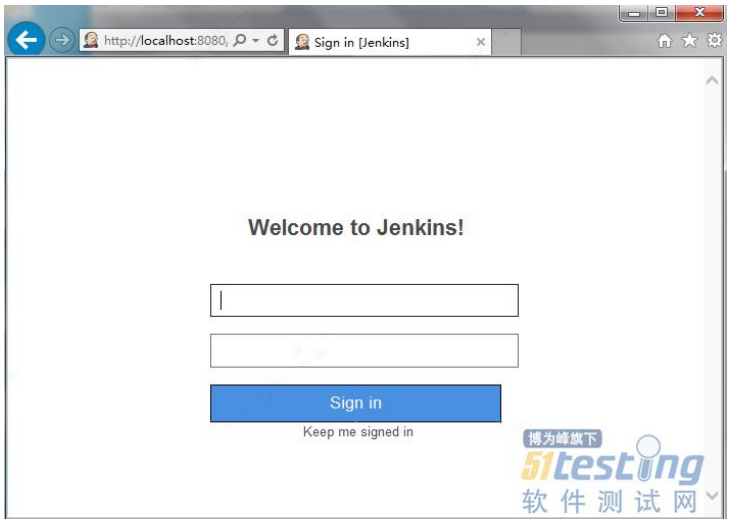
想偷懒就自己做个 Jenkins Job

◆作者：张淼

从 09 年开始接触 Jenkins，从最基础的做 job，到做 view，到做大型项目 job 的并行，串联，出报告，功能之强大真的是叹为观止，就算不用插件，也能满足日常测试需要了。最近被同事安利各种好用的插件，更新很快嘛，这么多好用的东西不拿来用真是浪费。正好前段时间自己做实验装了个 Jenkins，打算试一把，主要目的是访问 Linux Server，然后在上面执行一些东西，想象中是个简单的活儿，做起来简直到处都是坑~

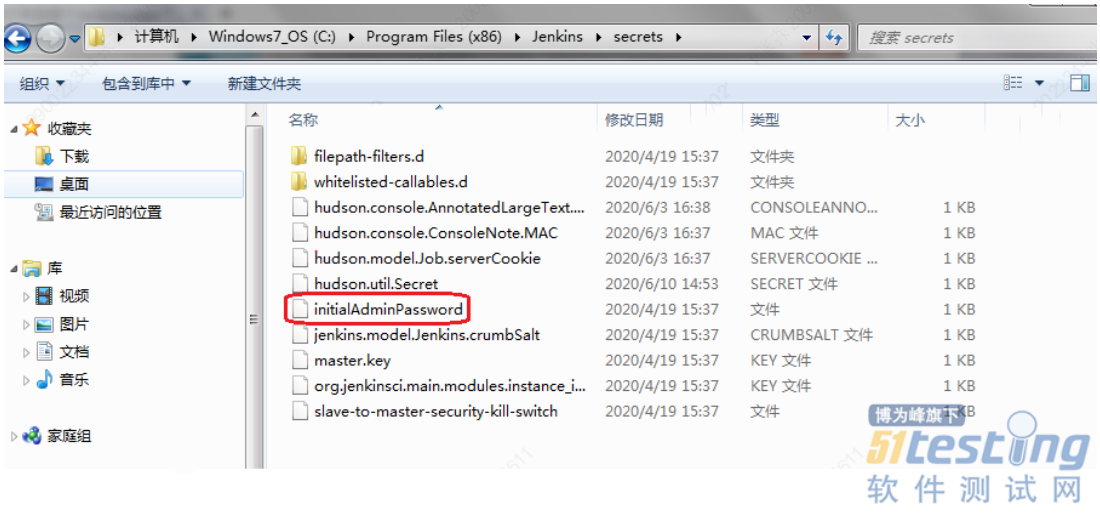
最开始想的是这个 Job 做成功之后，比如需要去某个 server 上面查 log 就再也不用每天 Putty -> Login -> 一堆 command 过去 -> vi -> 使劲翻篇找内容，只要做一个 job，每次要看的时候点一下，就搞定啦！这还没算上每次都要从小本本上找 server 的用户名密码浪费时间，按进去的隐藏密码还可能会出错，没有大显示器 vi 完了翻个篇都头晕，还要记住各个环境的名字，尤其是同时做五六个项目，很有可能会记乱了测试 Server，等等等等。赶紧动手来干吧！

首先，我们在本地装 Jenkins，网上的安装教程一大堆，随便找一个，一路 Next 也就装上了：



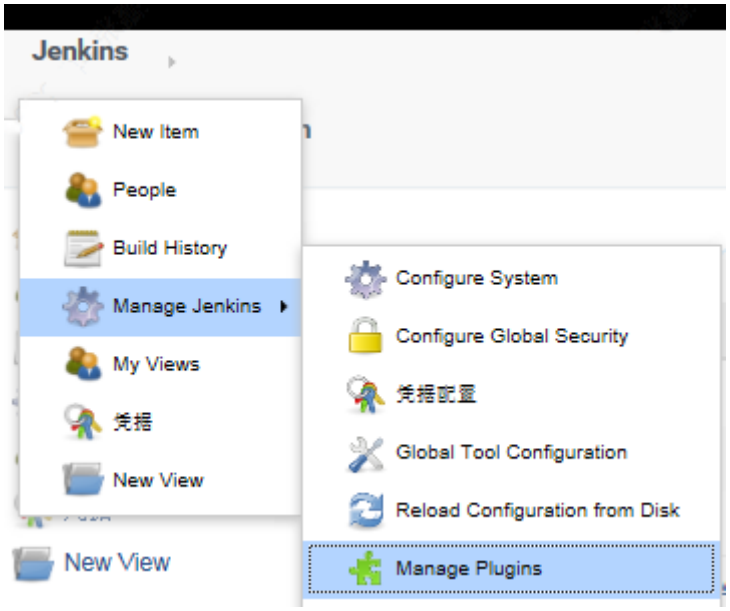
但是谁说装上就没有坑的！安装之后第一次登陆是需要初始密码的，然后我第二天到公司突然发现把密码忘了，顺便把初始密码路径也忘了……但是隐约记得刚装好的时候是有提示的：

傻乎乎地重装了一次，终于找到了，赶紧找个小本本记上，初始密码路径是 C:\Program Files (x86)\Jenkins\secrets\:

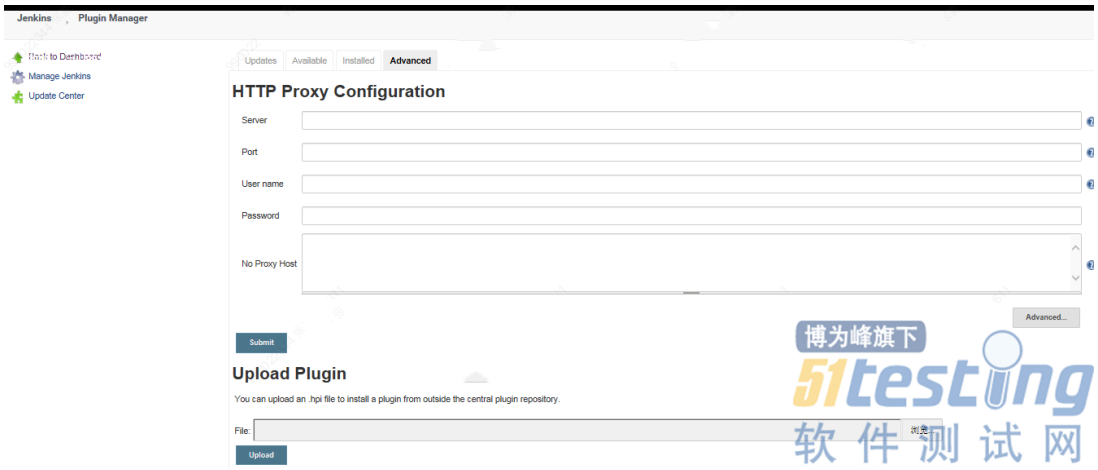


装好之后自定义 Jenkins，可以选择安装推荐的插件，和选择插件来安装，看了看“选择插件来安装”里面的插件列表，包括汉化包，Git，Email Extension 之类的，感觉很多用不到的东西，装上之后反而会让 Jenkins 很臃肿，于是打算只选择自己用的到的 Publish Over SSH，然而坑来了，电脑没有网……

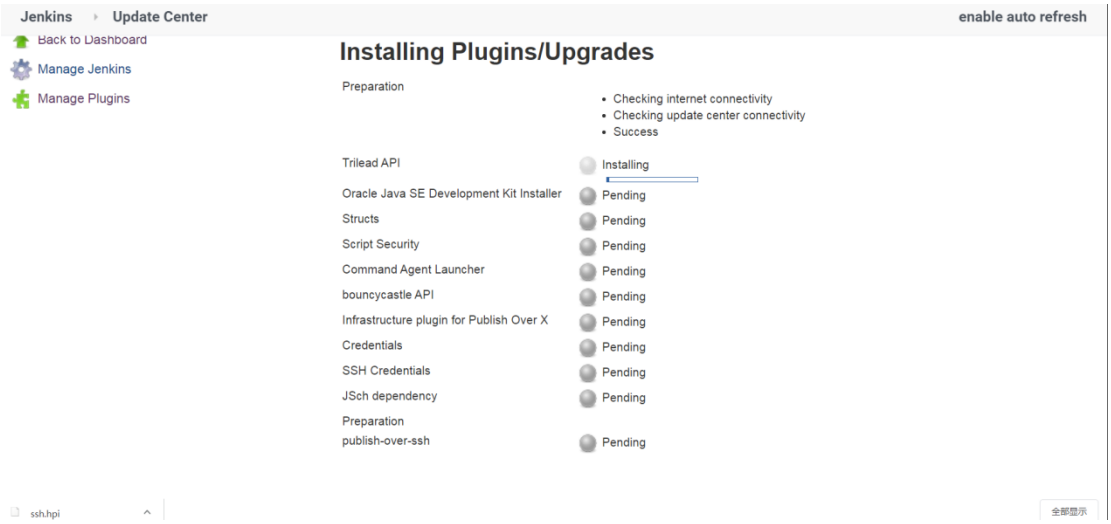
如果不安装任何插件，后续也可以去 plugins.jenkins.io 再下载，然后离线安装。



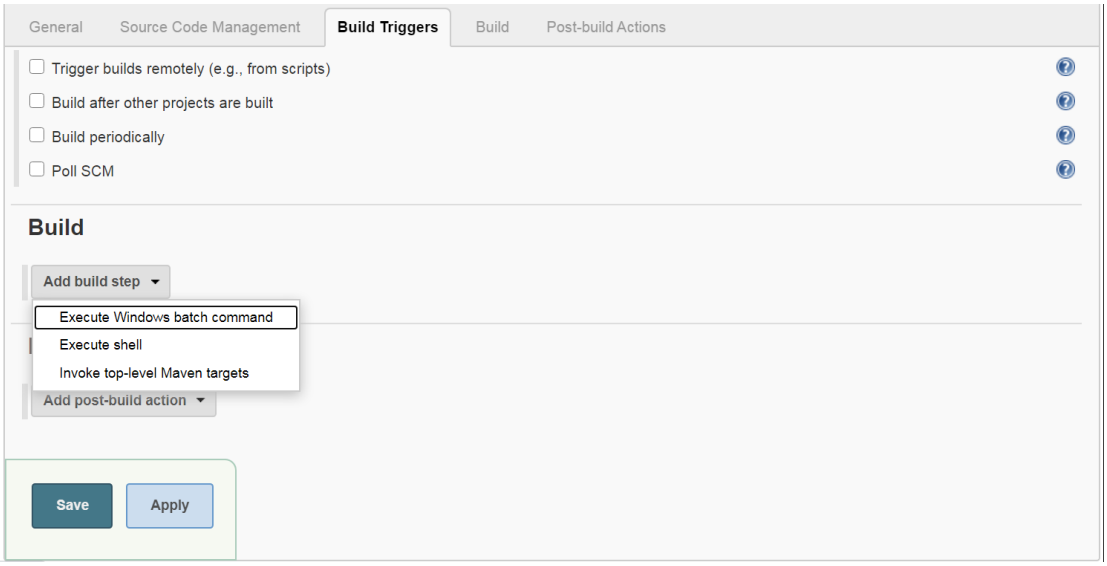
支持离线安装就好，下载完之后，在 Jenkins -> Manage Jenkins -> Manage Plugins -> Advance Tab -> Upload Plugin:



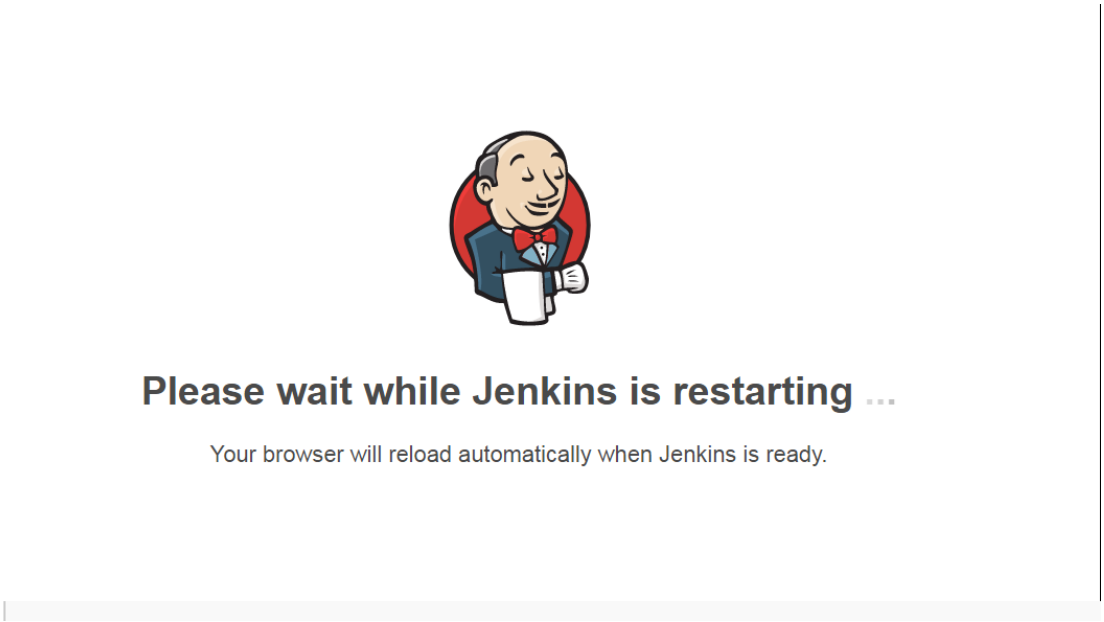
天真地以为 upload 之后会出现一个 Success，结果却是缺！依！赖！包！如果没有网的机器，请务必下好依赖包，按顺序安装，且一定要注意依赖包的版本，有的包太新导致不能兼容旧版 Jenkins:



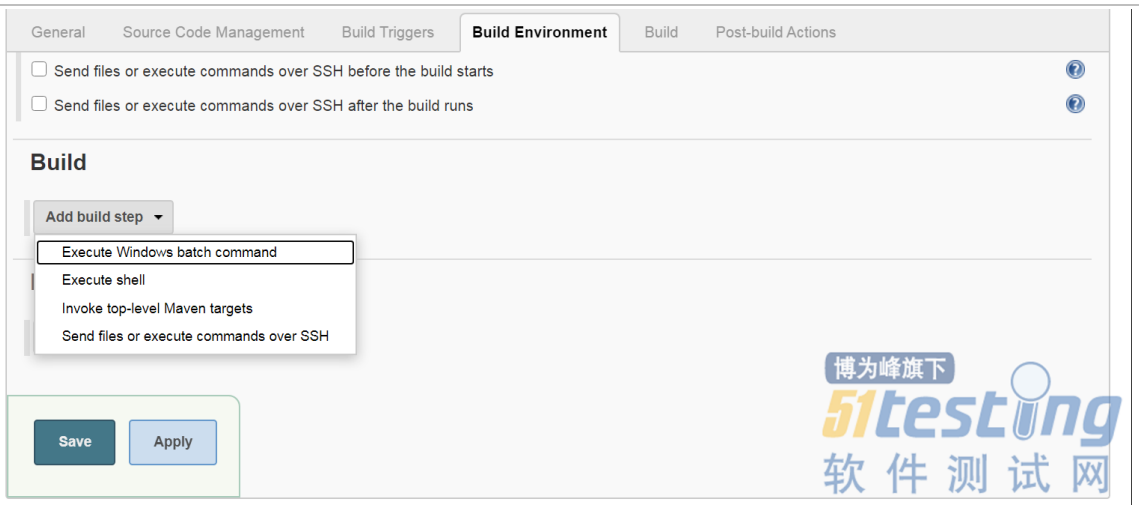
趟过了这几个坑咱们继续往后进行日常操作：New Item -> FreeStyle Project -> OK，在安装 Publish Over SSH 插件之前，Configure 里面的 Build 只有 3 种模式:



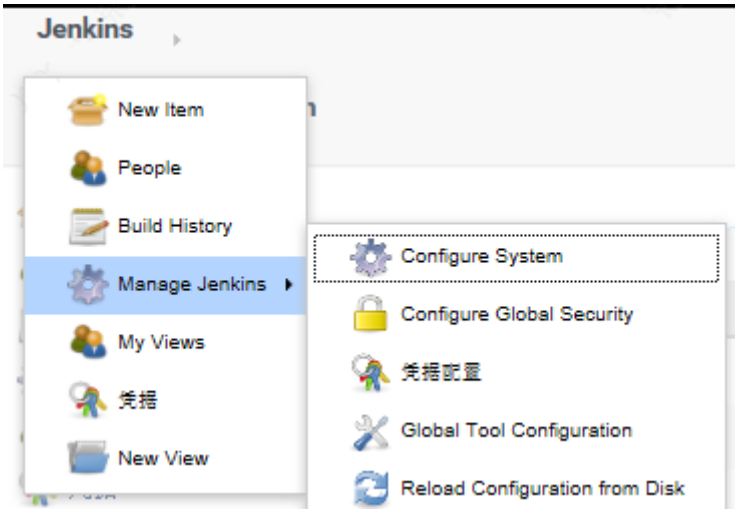
装好插件之后，重启 Jenkins:



然后回到 Job 里面我们会发现，多了个选项叫“Send files or execute commands over SSH”:

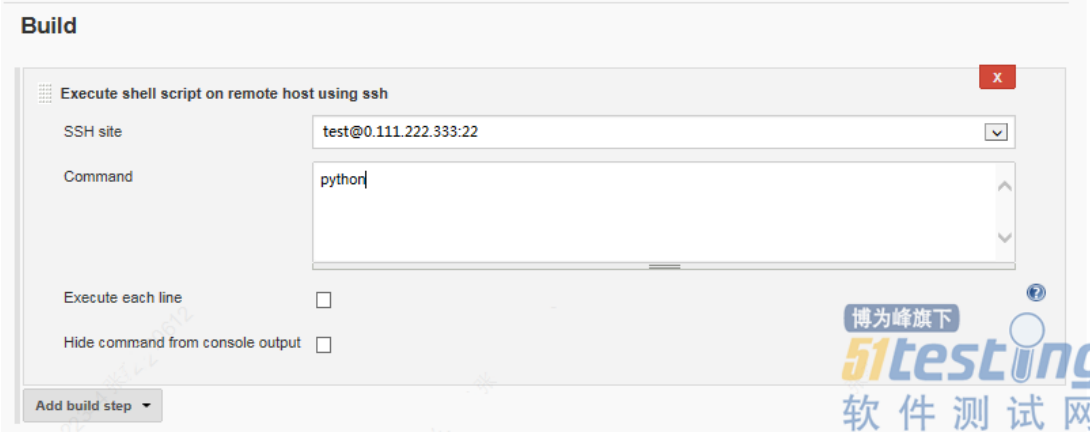


下一步就是开始配这个 Publish Over SSH，网上教程一大堆，咱们就按官网来吧，先去到 Manage Jenkins -> Configure System:



找到 Publish over SSH 配置的模块，点 SSH Servers 后面的 Add，填 Name，Hostname，Username，然后点 Advanced，勾选 Use password authentication, or use a different key，填上 Passphrase/Password，最后点 Test Configuration，如果显示 Success 就表示连上了。这个地方的坑在于可能会报各种 jenkins.plugins.publish.over.BapPublisherException，见招拆招，提示还是很明确的，基本都能解决。

配置完这块回到 Job 的 Configure 里面，点击 Build 下面 Add build step，选择 Execute shell script on remote host using ssh，SSH Site 选择刚才配置的，然后 Command 里面直接输入你想执行的命令就行了，比如 python:



马上要大功告成了？竟然没有坑？我果然又天真了……报错如下：

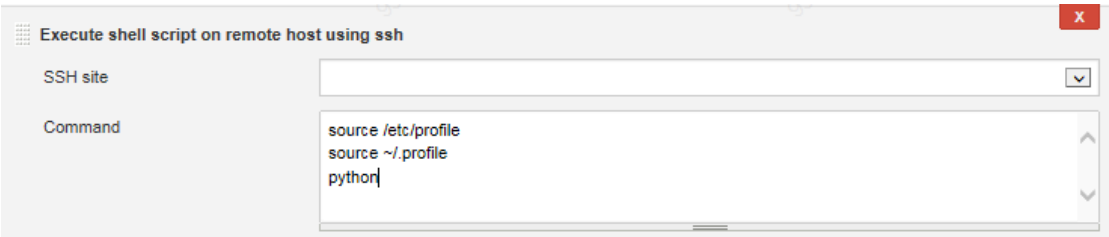
Which: no python in ($PATH)

大概的意思是 Path 里找不到 python，这个很明显是环境变量没配对嘛，于是果断 putty 到机器上，echo $PATH 了一下，然后就惊呆了，python 明明就在那里啊!!! Debug 的过程大概就是回到 Job 里面，在 Command 里填了 echo $PATH，再跑一次，发现果然

没有 Python。这……Path 还能被 Jenkins 吃了的？看了无数帖子发现，对的，Jenkin 会吃 path 的（大概意思就是 publish over ssh 会把 Jenkins server 自身的环境变量带过去），把网上找到的各种解决方案都试了一遍，甚至连 bashrc 文件都做了一个还是不行，最终发现唯一好用的方法就是在 Command 里面加上下面两行，根据系统不同，这里的 profile 有可能是 bash_profile:

source /etc/profile

source ~/.profile



然后再执行，哇！果然成功了！蓝色的小点点在向我微笑:

测试女巫紧跟时代脉搏之大数据分析系列（3)

◆ 作者：王平平

很快，三个月又过去了，口罩已经成了生活的必需品，“后疫情时代”已经把我们变得已经对于每天疫情情况麻木了，我们也许真的不得不习惯与“新冠肺炎”长期共处。时间总是像奔腾不息的江水一样，汹涌地奔腾向前，真的不管外界的环境是怎样，我们真的需要时常问自己：被这些奔腾的江水裹挟着的我们，在这些汹涌的江水里能沉淀些什么？女巫这三个月把如何使用 docker swarm 的 stack 进行分布式发布，以及如何利用 redis cache db 来提高代码的性能搞定，也是浓墨重彩的一笔，好吧，闲话不多说，我们进入此次的学习吧~

此次学习的方向是有关模组的学习：即企业数据可视化解决方案 Plotly。上一期我们已经学习了统计学的"描述统计学"又叫"叙述统计"，以及数据清理，数据整理。我们已经具备了大数据分析的最基础的知识，这一期我们将继续学习用形象化的方法即画图的方式来分析大数据。

Plotly 介绍

也可以选择使用 pyechats, matplotlib 都可以画图；选择一个 module 进行 study 即可~

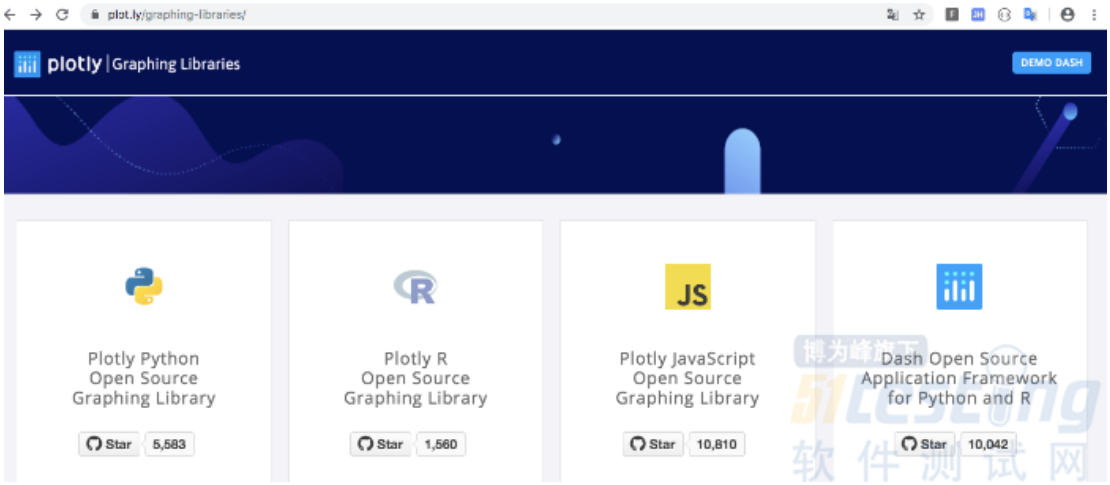
目标

Bring data science out of the lab and into the business

Open Source

plotly 本身是基于 javascript 开发的，但是提供了大量与其他主流数据分析语言的 API，可以参考 https://plot.ly/api/

来自官网的解释：Plotly's team maintains the fastest growing open-source visualization libraries for R, Python, and JavaScript.如下图

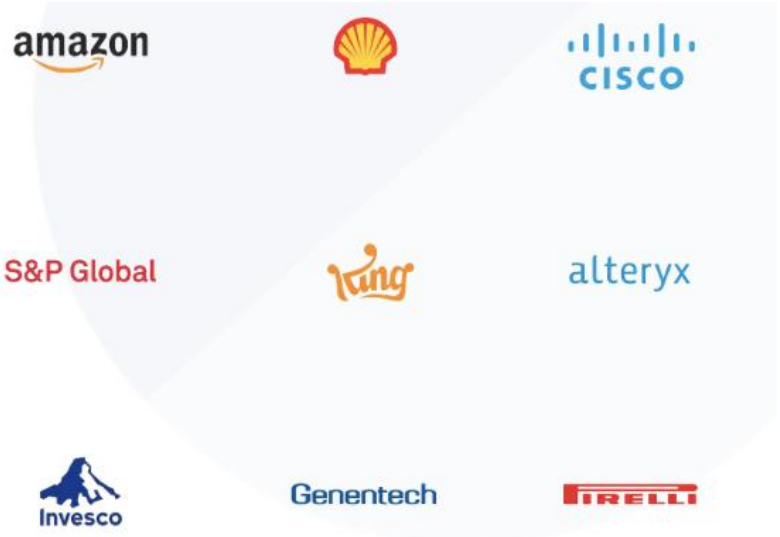


注意 dash open source is the front-end to your analytical Python backend，它只有大概 40 行的代码，提供了一个用于键入 UI 控件的界面，包含滑块，下拉列表和带代码的图形等等；因为我们已经有 angular 所以它对我们意义不是很大。

但是很受大家欢迎，从 star 就可以看出来。

应用广泛

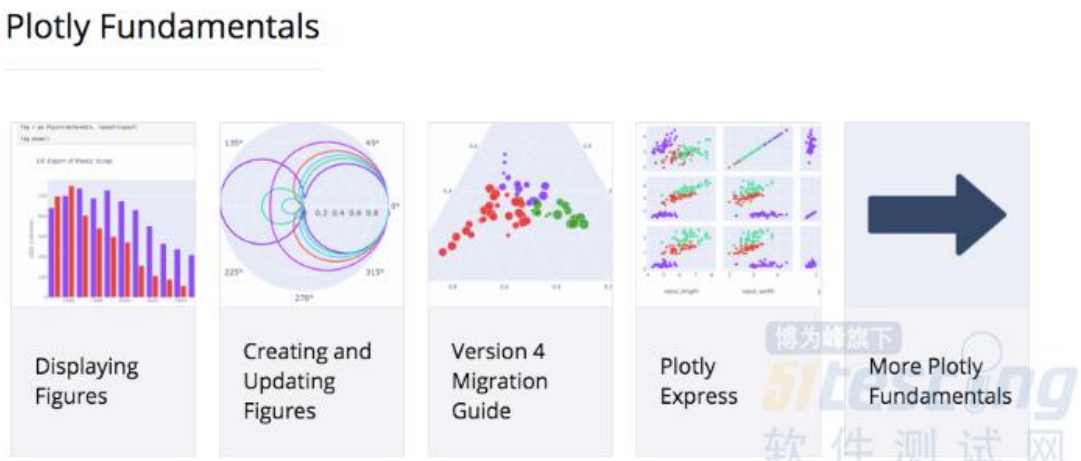
Our software is embedded in mission critical applications across the Fortune 500，如下图



Plotly_Python

Poltly Fundamentals

如下图



我们这次介绍的主要是 plotly.graph_objects 以及 Plotly Express

https://plot.ly/python/creating-and-updating-figures/#constructor

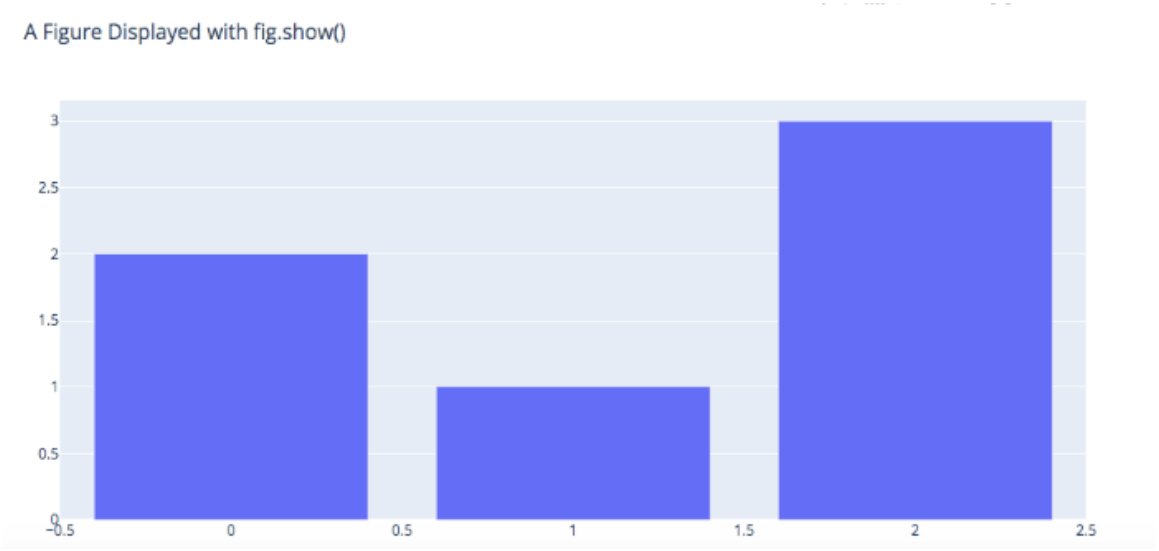
Create Figures

Figures as graph objects:

使用 renderers framework 显示 figures，即导入的 graph_objects 这个 module，调用其中的函数 Figure，并返回实例，并将实例 show 出来

```
    import plotly.graph_objects as go
fig = go.Figure(
    data=[go.Bar(y=[2, 1, 3])],
    layout_title_text="A Figure Displayed with fig.show()"
)
fig.show()
fig #这种方式与 fig.show()作用一样，都是将 figure 显示出来
```

产生的结果如下图



调整 title 为居中的参数

```
import plotly.graph_objects as go
fig = go.Figure(
    data = [go.Bar(x =[1,2,3],y = [1,2,3])],
    layout = go.Layout(
        title = go.layout.Title(text="我是柱状图",x = 0.5) # x=0.5 指的是 title 需要居中显示
    )
)
fig.show()
```

JSON data structure 的方式描述 figure 的属性

```
fig ={
    "data":[
        {
            "type" : "bar",
            "x":[1,2,3],
            "y":[1,2,3]
        }
    ],
    "layout":{"title":{"text":"我是柱状图"}}
}

# To display the figure defined by this dict, use the low-level plotly.io.show function
import plotly.io as pio
pio.show(fig)
```

当然也可以用 graph object 的方式实现 JSON data structure

```
import plotly.graph_objects as go
fig = go.Figure({
```

```
        "data": [{"type": "bar",
                  "x": [1, 2, 3],
                  "y": [1, 3, 2]}],
        "layout": {"title": {"text": "A Bar Chart"}}
    })
    fig.show()
```

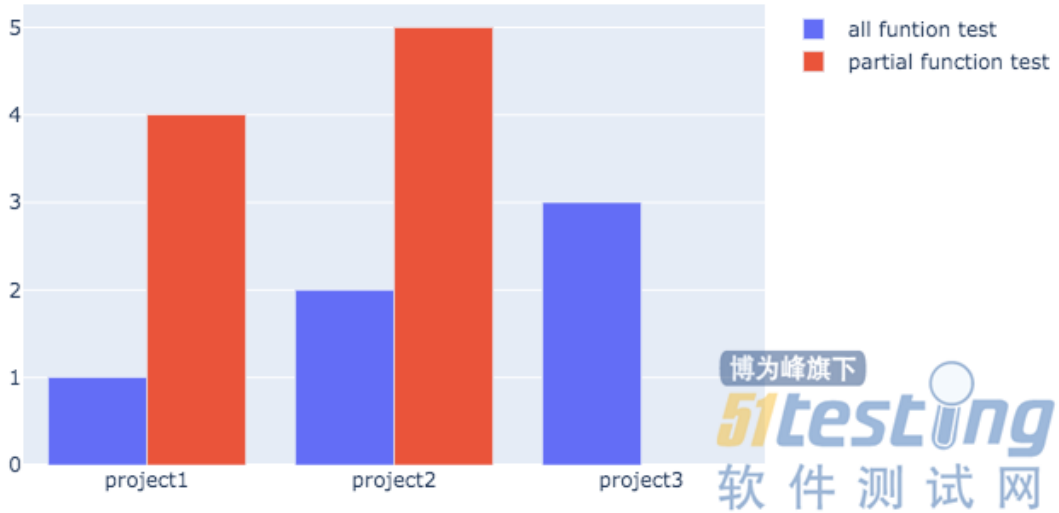
Update Layout

update_layout 是一种更新图形布局(可以自动以 group chart)，也可以更新图形标题的文本和字体大小

更新图形布局

```
import plotly.graph_objects as go
fig = go.Figure(data =[ go.Bar(name ='all funtion test',x = ['project1','project2','project3'],y = [1,2,3]),
                        go.Bar(name = 'partial function test',x =['project1','project2'],y = [4,5,6])])
fig.update_layout(barmode='group')
fig.show()
```

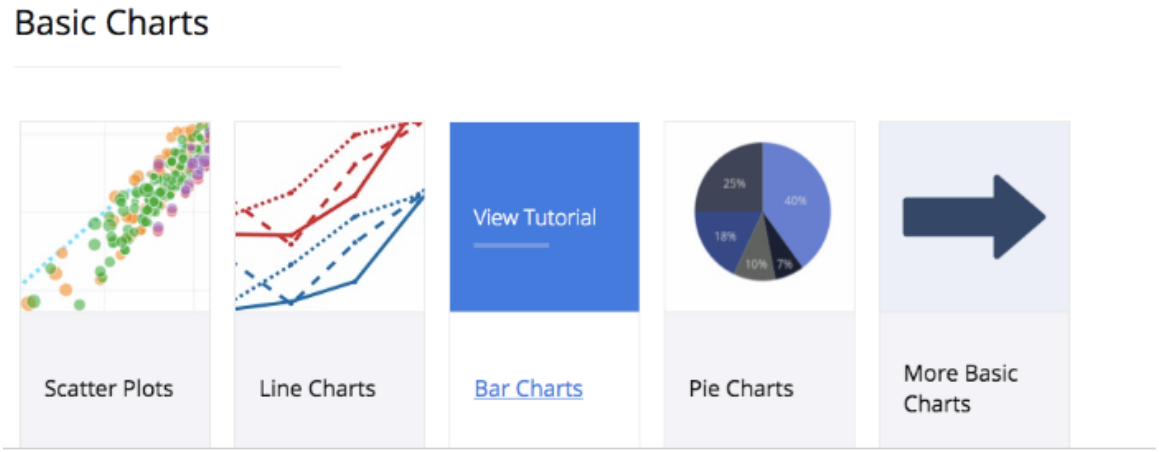
运行结果如下图



各个图形的细节的属性

例如图形的悬浮文字，颜色等可以根据图形的样式查看 Basic Chart，例如你如果使用的是 bar chart 就点击 bar chart 其中就可以找到你想要的细节属性设置

如下图 6



Plotly Express

它是一个简洁的包装器，它之前是一个独立的软件包，安装它需要 pip plotly-experss，现在都打包到 plotly 这个整体包之中，它已经融入 plotly 其中；相对于 plotly.graphobjects 而言它是一个比较高级的包装器，plotly.graphobjects 主要是用来快速数据探索和图形生成。

它其中还包含一些科学数据，供你画图使用

我们查看有关花的一些集成到 plotly_express 的科学数据包

```
import plotly.express as px
# 这个 express 中，包含了很多让你实验的数据
iris = px.data.iris()
print(px.data.iris.__doc__)
px.data.iris().head()
```

运行结果如下:

使用 iris.doc 查看这个数据的说明文档，我们也要养成写""" """ 包裹的某个 class function 设置 package 的说明文档的习惯。同时将这组数据的 head 也打印出来，大概了解这组数据是 pandas 的 dataframe，它由 150 行，6 列，注意会自动生成 index:即 0-4，因为对于 pandas 的 Dataframe，如果没有指定现有的列为 index，它会自动生成一列 index。

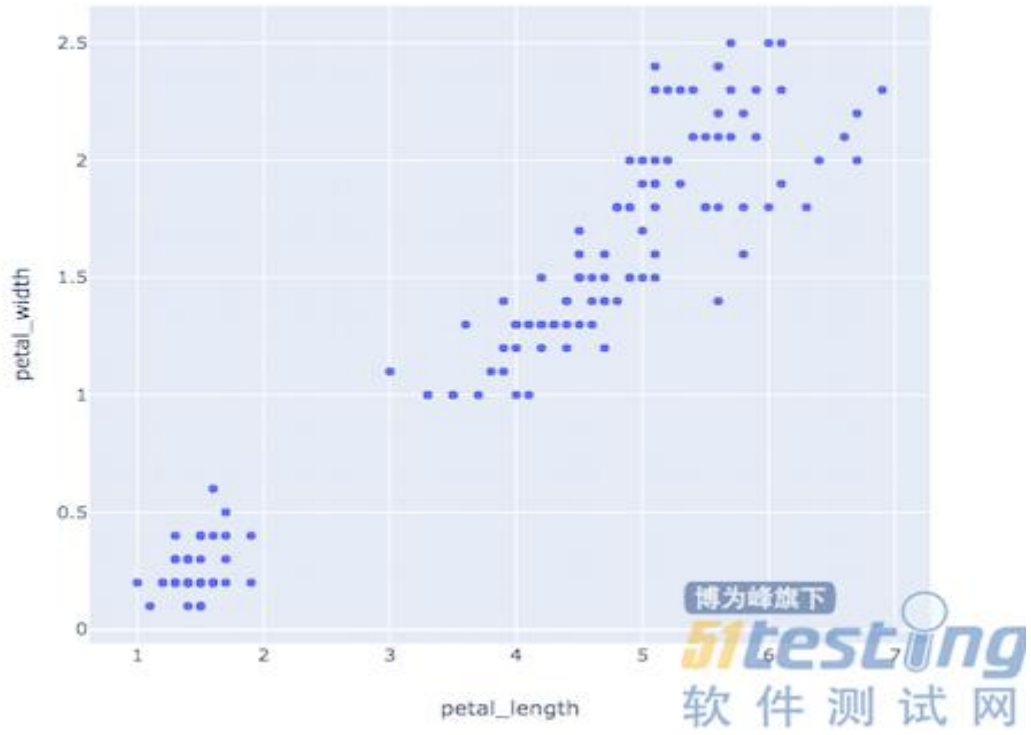
```
Each row represents a flower.
    https://en.wikipedia.org/wiki/Iris_flower_data_set
    Returns:
        A `pandas.DataFrame` with 150 rows and the following columns: `['sepal_length',
'sepal_width', 'petal_length', 'petal_width', 'species',
```

'species_id']`.

	sepal_length	sepal_width	petal_length	petal_width	species	species_id
0	5.1	3.5	1.4	0.2	setosa	1
1	4.9	3.0	1.4	0.2	setosa	1
2	4.7	3.2	1.3	0.2	setosa	1
3	4.6	3.1	1.5	0.2	setosa	1
4	5.0	3.6	1.4	0.2	setosa	1

```
import plotly.express as px
# 这个 express 中，包含了很多让你实验的数据
iris = px.data.iris()
fig = px.scatter(iris,x='petal_length',y='petal_width')
fig.show()
```

运行结果如下图 7

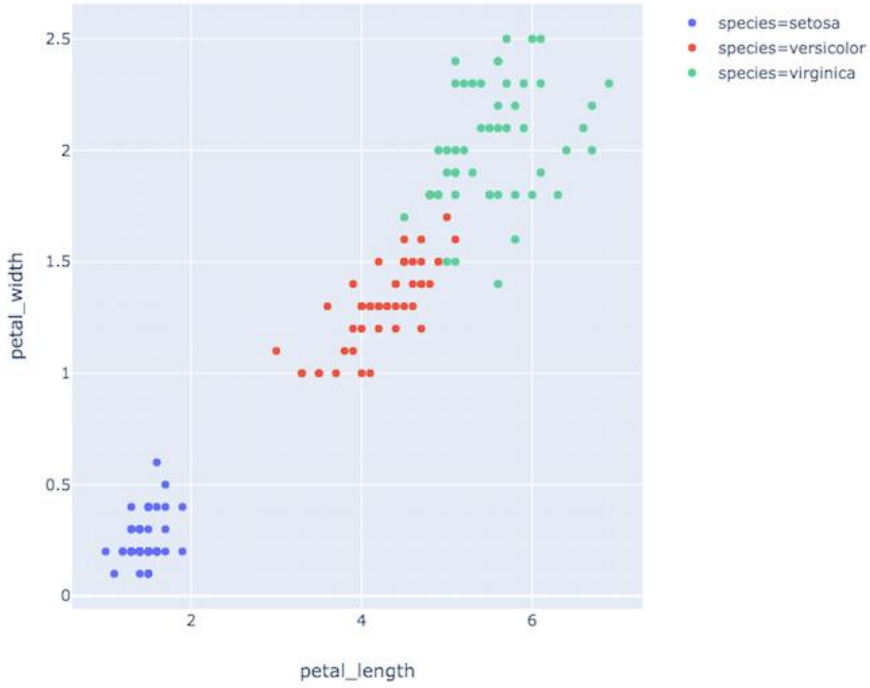


如果使用三列，即将 species 加上去，按照颜色进行分类，可以这样呈现

```
import plotly.express as px
# 这个 express 中，包含了很多让你实验的数据
iris = px.data.iris()
fig = px.scatter(iris,x='petal_length',y='petal_width',color="species")
```

```
fig.show()
```

运行结果如下图 8



各个图形的细节属性

https://plot.ly/python/plotly-express/#plotly-express

Mixed Subplots in Python

作用

在一张图中混合显示不同形式的图

https://plot.ly/python/mixed-subplots/#mixed-subplot

范例

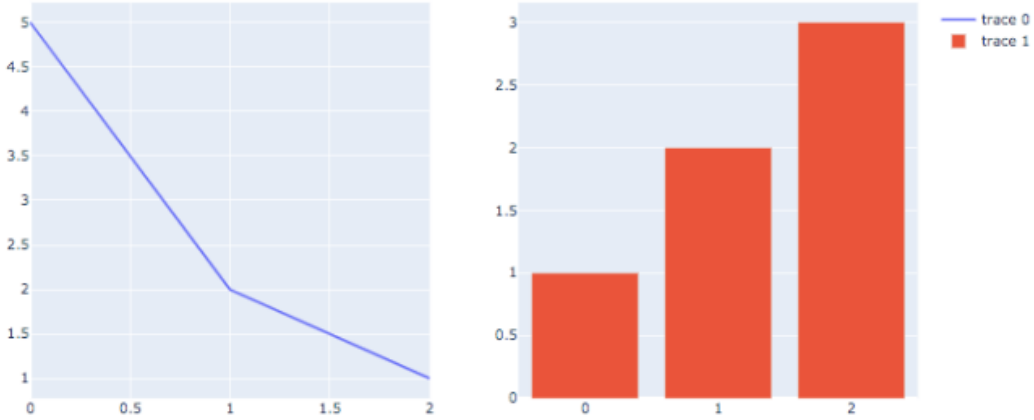
使用 make_subplots 开确定图形显示几行几列，如下是 1 行 2 列

使用 add_trace 来将各个图形的参数定义清楚

```
#创建子图
#看源代码的方法：在 jupyter 中： shift+tab
from plotly.subplots import make_subplots
fig = make_subplots(rows=1,cols=2)
fig.add_trace(go.Scatter(y=[5,2,1],mode = 'lines'),row=1,col=1)
fig.add_trace(go.Bar(y=[1,2,3]),row =1,col =2)
```

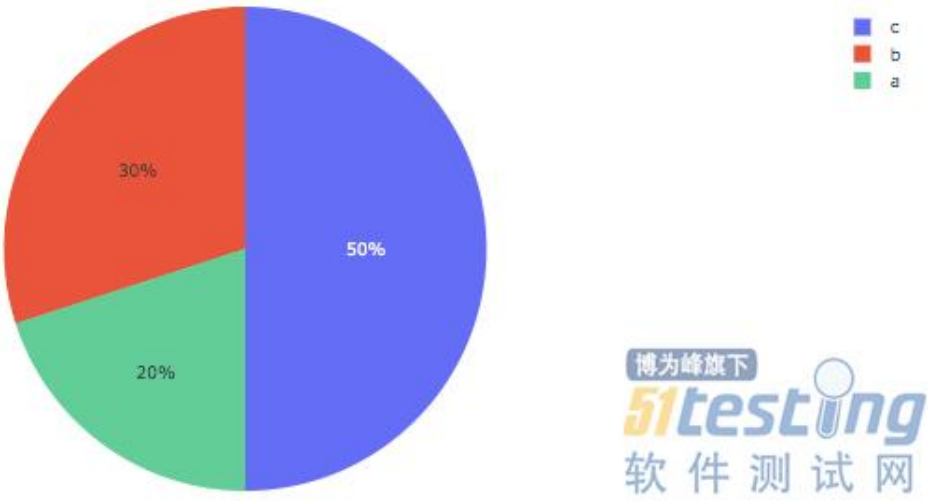
```
fig.show()
```

运行结果如下图 9




```
#饼图
import plotly.graph_objects as go
import plotly.express as px
fig = go.Figure(data=[
    go.Pie(labels=['a','b','c'],values=[20,30,50])
])
fig.show()
‘’ ‘
```

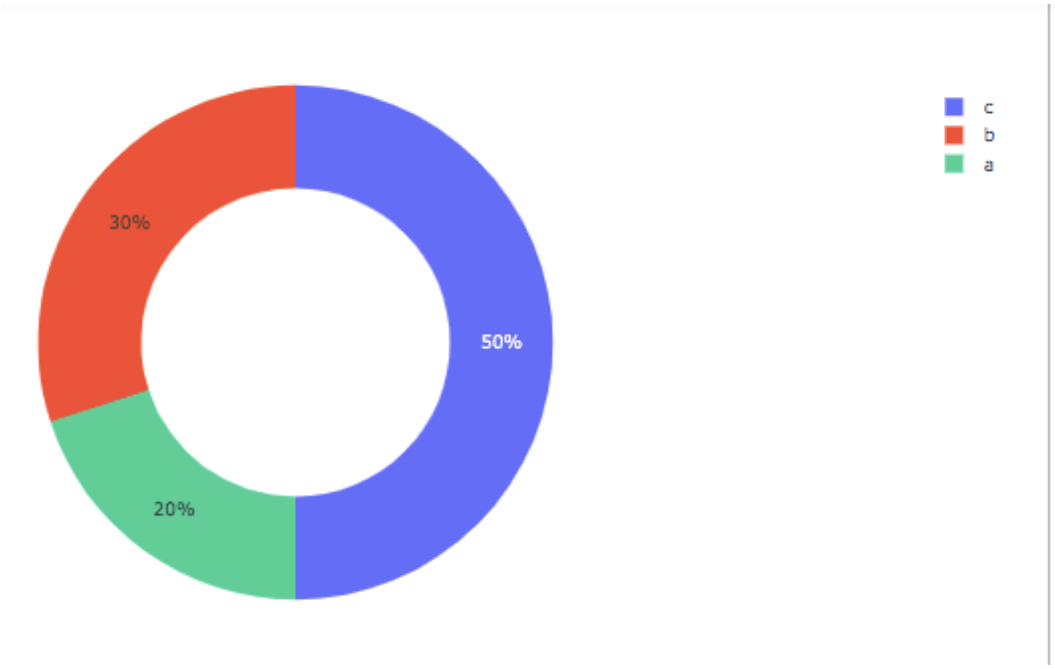
运行结果如下图



```
#环图

import plotly.graph_objects as go
import plotly.express as px
fig = go.Figure(data=[
    go.Pie(labels=['a','b','c'],values=[20,30,50],hole = 0.6)
])
fig.show()
```

运行结果如下图



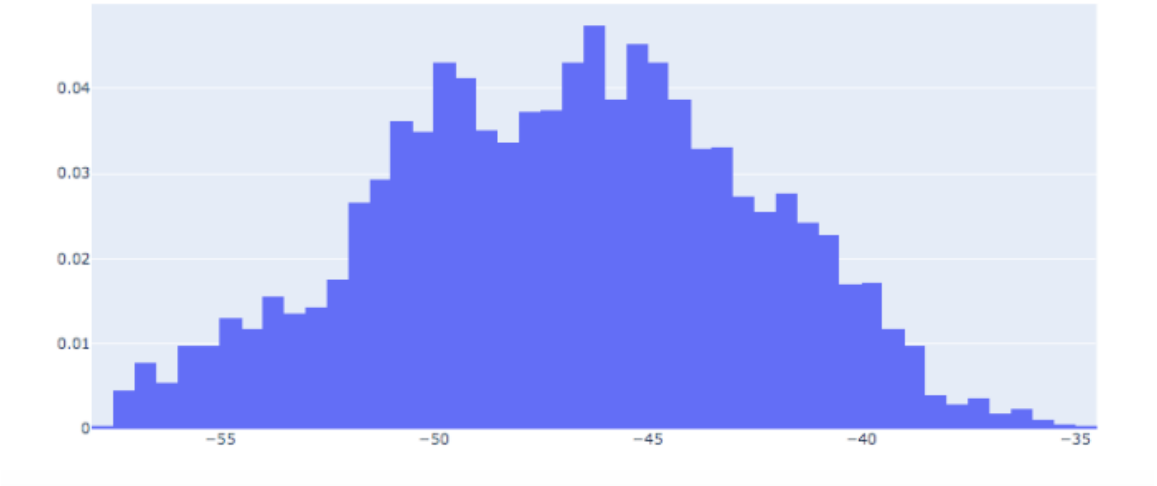
数据分布直方图

即画出描述数据分布的直方图，从直方图就可以看出数据分布，虽然不是很完美但是基本上就是正态分布；大家一定要记住一个概念，看数据的分布以及进行数据的分析，不能只看某一个分析的结果，一定要结合数据的各种 chart 一起确认，一定要从不同的角度去论证，以保证你的结果的正确性。

```
import numpy as np
import pandas as pd
from scipy import stats
import plotly.graph_objs as go
excel_data = pd.read_csv('11adtoolsingle.csv')
excel_data_txpower = excel_data['Value']
trace=go.Histogram(
    x=excel_data_txpower,
```

```
        histnorm = 'probability'
)
fig = go.Figure(data = [trace])
fig.show()
```

运行结果如下:



这一期我们学习了 plotly 这个第三方模组，重点学习了 plotly.graph_objects 以及 Plotly Express 中的柱形图，分布图，圆饼图，环形图等的画法；其实希望大家能领悟的是如何学习利用 open source 的这个第三方模组画出我们需要的图，下一期我们将根据这三期学到的基础知识，对于我们工作中遇到的问题，进行实战分析；这个实战分析结束后，将继续进入大数据分析的“深水区”即利用现有的大数据如何“预测未来”，并将其应用到我们的工作中，怎么样？是不是很期待？加油！让我们在这个多事之秋，快速提高自己的 coding 以及分析问题的能力，为新的局面提前做好准备！加油！

简述产品测试方法

◆作者：祝德慧

摘要： 当软件成为一个成功的产品后，拥有的用户量大，迭代频繁，测试的周期短，重复性强。面对紧张复杂的产品测试工作，应怎样完成这一系列的测试工作呢？下面根据自身经验介绍下产品测试方法。

一、根据软件产品特点，形成产品测试的知识体系

从产品的角度来看，公共组件的功能测试是每次测试的重中之重，下面来看下功能测试框架:

序号	测试点	分散
1	安装	首次安装 覆盖安装 补丁包安装 不同操作系统安装
2	卸载	使用中卸载 控制面板卸载 安装包卸载
3	更新	相邻版本更新 跨版本更新 在线更新 离线更新
4	权限	加密锁授权 网络版授权 其它方式授权 离线授权 在线授权
5	正向测试	UI 界面 按钮功能 正向业务流程 数据展示 快捷键使用
6	反向测试	输入错误数据 快速多次点击同一按钮

		快速多次执行同一功能 反向业务流程
7	性能测试	不同负载情况下
8	安全测试	防病毒软件（360 软件等） 防火墙
9	文档测试	需求文档 概要设计文档 开发进度计划 用户手册 帮助文档
11	兼容性测试	不同浏览器 不同分辨率 不同操作系统

二、根据产品功能，开发自动化测试脚本

由于产品的迭代频繁，UI 功能稳定，每次发版前的测试工作量大，重复性工作多，此时引进自动化测试可以提升回归效率，节约人力，更好地利用资源，提高软件测试结果的准确性，增加软件信任度。**开展测试工作的步骤如下:**

1、选择自动化测试工具。一般对于 Web 界面测试的常用工具有 Selenium，使用的开发语言为 Python；对于 delphi 语言或其他语言编写的客户端软件，测试工具一般由公司内部开发，通用的开源测试工具不能读取这类软件的控件和控件数据。

2、确定自动化测试的范围。产品第 1 个版本稳定后，可考虑将正向业务流程、核心功能、UI 界面稳定的功能这些功能编写自动化测试脚本。

3、开发自动化测试脚本。针对不同的测试工具，开发模式不同。

（1）使用 Selenium 工具，建议使用 PO 模式。先封装一个 BasePage 类，每个 Page 都继承 BasePage，通过 driver 来管理本 page 中元素，将 page 中的操作封装成一个个的方法；TestCase 继承 unittest.Testcase 类，并且依赖 page 类，从而实现相应的测试步骤。

BasePage 代码如下:

```
from selenium.webdriver.support.wait import WebDriverWait
from selenium import  webdriver
from selenium.webdriver.support import expected_conditions as EC
class BasePage(object): #BasePage封装所有页面的公有方法
def _init_(self,selenium_driver,base_url):
    self.driver = selenium_driver
    self.base_url = base_url

def on_page(self, pagetitle):
    return pagetitle in self.driver.title

def _open(self, url):
    self.driver.get(url)
    self.driver.maximize_window()

def open(self):
    self._open(self.base_url, self.pagetitle)

def find_element(self, *loc):  # *loc任意数量的位置参数（带单个星号参数）
    try:
        WebDriverWait(self.driver, 10).until(EC.visibility_of_element_located(loc))
        return self.driver.find_element(*loc)
    except:
        print("%s 页面未能找到 %s 元素" % (self, loc))

def script(self, src):
    self.driver.excute_script(src)

def send_keys(self, loc, vaule, clear_first=True, click_first=True):
    try:
        loc = getattr(self, "_%s" % loc)  # getattr相当于实现self.loc
        if click_first:
            self.find_element(*loc).click()
```



```
        if clear_first:
            self.find_element(*loc).clear()
            self.find_element(*loc).send_keys(vaule)
    except AttributeError:
        print("%s 页面中未能找到 %s 元素" % (self, loc))
```

Page 代码如下:

```python
from selenium.webdriver.common.by import By
from PODemo.BasePage import BasePage  #假设Page.py、BasePage.py均在PODemo.BasePage目录下
from selenium import webdriver

class SearchPage(BasePage):
    # 定位元素
    search_loc = (By.ID,"kw")  #通过ID来定位搜索框
    btn_loc = (By.NAME,"go")   #通过NAME来定位搜索按钮

    def open(self):
        self._open(self.base_url)

    def search_content(self,content):
        Content1 = self.find_element(*self.search_loc)
        Content1.send_keys(content)

    def btn_click(self):
        Btn1 = self.find_element(*self.btn_loc)
        Btn1.click()
```

TestCase 代码如下:

```python
from unittest import TestCase
import unittest
from selenium import webdriver
from time import sleep
from SeleniumProject.PO.Search import SearchPage
class CaseRun(TestCase):
    def setup(self):
        self.driver = webdriver.Chrome()
        sleep(3)
        self.url = "https://www.taobao.com/"
        sleep(3)
        self.content = "炒锅"
    # 测试步骤
    def test_search(self):
        bing_page = SearchPage(self.driver,self.url)
        bing_page.open()
        sleep(3)
        bing_page.search_content(self.content)
        sleep(3)
        bing_page.btn_click()
        sleep(3)

    def tearDown(self):
        self.driver.quit()

if _name_ == "_main_":
    unittest.main()
```

（2）使用其他测试工具，建议根据软件特性，建立公共函数库，确定测试脚本的

配置文件和参数内容。TestCase 引用公共函数库，公共函数文件的位置需与测试工具的执行程序在同一目录，从而实现相应的测试步骤。

公共函数的内容如下:

```
function StartProgram(path)
  root := ExtractFilePath(path)
  WinExecute(path, "", root)
  Sleep(500)
  hwnd := WaitWindowPath(2000, "|@用户登录|TPanel,2|TBitBtn,2")
  ClickBtn(hwnd)
  Sleep(9000)
  hwnd1 := WaitWindowPath(1000, "|@警告|TBitBtn")
  if hwnd1 >0 then
    ClickBtn(hwnd1)
    Sleep(250)
  endif
return 0

function OpenFile(fileName,Ystarea)
  hwnd := WaitWindowPath(2000, "|@打开|ComboBoxEx32|ComboBox|Edit")
  setfocus(hwnd)
  Sleep(500)
  SetEditText(hwnd, fileName)
  Sleep(1000)
  hwnd := WaitWindowPath(5000, "|@打开|Button")
  ClickBtn(hwnd)
  Sleep(1000)

return 0

function SaveFile(fileName,Ystarea)
  hwnd := WaitWindowPath(2000, "|@保存为...|DUIViewWndClassName|DirectUIHWND|FloatNotifySink|ComboBox|Edit")
  SetEditText(hwnd, fileName)
  sleep(500)
  hwnd := WaitWindowPath(4000, "|@保存为...|Button")
  ClickBtn(hwnd)
  Sleep(250)
return 0
```

TestCase 的内容如下:

```
uses YstPublic
uses CLB
uses QTXMQD


iniopen("config.ini")
DBSJFILE:=iniread("system","DBSJFILE","")
exepath  :=  iniread("system","exepath","")
Ystarea :=  iniread("system","Ystarea","")
Report :=  iniread("system","Report","")
inipath := iniread("system","inipath","")
Report := Report + Sdate()  + ".xls"

;iniopen(inipath)
;iniwrite("system","UIAutomation",1)
;iniclose()

Obj1 := CreateObj("ExcelIO.ExcelFile")
DoObjMethod(Obj1, "Open", DBSJFILE)
DoObjMethod(Obj1, "SelectSheet", "Sheet1")
Obj2 := CreateObj("ExcelIO.ExcelFile")
DoObjMethod(Obj2, "Open", Report)
DoObjMethod(Obj2, "SelectSheet", "Sheet1")

;StartProgram(exepath)
hMain := WaitWindowPath(3000, "|@"+Ystarea+"%")
setfocus(hMain)


SETREPORT(obj1,obj2,"分部分项清单")
LRDESJ(Obj1, Ystarea,"分部分项清单",40)    ;录入分部分项清单的定额数据
BCDE(Ystarea)          ;增加补充定额
ADDBCCL(Ystarea)       ;增加补充材料
ADDCL(Ystarea)         ;增加材料
ADDZC(Ystarea)     ;对补1设置主材仪表
ADDSB(Ystarea)       ;;对1-0122定额增加设备
CLZCP(Ystarea)         ;对1-0405定额执行材料由半成品转换成成品
```


4、执行自动化测试脚本。确定软件变更需求后，根据需求维护已有的测试脚本。在开发提交第一轮测试时，运行自动化测试脚本，发现 BUG 后提交至 BUG 管理工具并进行回归测试。待开发修正所有的 BUG 后，需再次运行测试脚本，验证所有测试脚本的测试结果完全正确时才能发布。

5、维护自动化测试脚本。软件发布后，软件新增的功能需增加新的测试脚本加入测试库，用于下一次版本迭代的验证测试。

经过这几年的产品测试的实战经验，手工测试和自动化测试相结合，才能有效利用资源和时间。因为对于某些测试，手工测试方法只需要花费很短时间；但是如果使用自动化测试，却需要花费几个小时甚至几天的时间编写测试脚本。然而，自动化测试可以执行一些手工测试困难或不可能进行的测试，具有可重复性，回归测试更方便。因此，产品测试时将需求变动不频繁的核心功能用自动化测试来替代，其他功能手工完成，可以获得良好的投资回报率。

企业测试自动化：突破顶级障碍的 4 种方法

◆ 作者：枫叶译

摘要：具有复杂系统的成熟公司如何才能达到现代交付计划和流程所要求的测试自动化水平？有四种策略已帮助许多组织最终突破了测试自动化的障碍：简化整个技术体系的自动化，结束测试维护的噩梦，转向 API 测试，并选择适合您需求的工具。

尽管在软件测试会议、网络研讨会和出版物上不乏测试自动化成功案例，但它们主要针对开发人员和技术测试人员，这些人员包括：1）专注于测试简单的 Web UI；以及 2）拥有构建其应用程序的极大体验，以及在过去的几年中从头开始测试流程。他们的故事引人入胜——但与具有数十年缓慢发展的异构体系结构，合规性要求和质量流程的典型公司并不完全相关。

具有复杂系统的成熟公司如何才能达到现代交付计划和流程所要求的测试自动化水平？快速的答案是：这不一定。

让我们看看在多年的尝试之后，已经帮助许多组织最终突破了测试自动化障碍的四大策略：

简化整个技术体系的自动化

- 结束测试维护的噩梦
- 在可行的情况下转向 API 测试
- 选择适合您需求的工具

在阅读它们时，至关重要的是要认识到没有适合每个组织中每个部门的单一正确方法。对于每种高级策略，我都会指出一些可能影响它在你的组织中的重要性的关键考虑因素。

简化整个技术体系的自动化

测试自动化的传统方法依赖于基于脚本的技术。在开始自动化之前，必须先开发一个测试自动化框架。一旦最终实现，测试和调试了框架，就可以添加测试脚本以利用该框架。随着应用程序的发展，还需要检查，可能更新和调试这些测试脚本以及测试自动化框架本身。

通常，仅使用一项技术（例如 Web UI 或移动界面）就需要大量资源来进行测试自动化。这可能包括对现有测试人员进行关于您选择的特定脚本编制方法的培训，将开发资源重新分配给测试，或雇用已经掌握了该特定方法的基于脚本的测试自动化的新资源。

即使是精通脚本的测试人员也发现，构建、扩展和维护测试自动化是一项繁琐且耗时的任务。这通常会分散测试人员的核心能力：利用他们的领域专业知识来确定会损害用户体验并带来业务风险的问题。

如果您具有要测试的异构应用程序堆栈（例如打包的应用程序，例如 SAP、Salesforce、ServiceNow 或 Oracle EBS 以及 API、ESB、大型机、数据库以及 Web 和移动前端），则需要学习多个框架，构建和链接以自动化端到端测试用例。Selenium 是迄今为止所有现代测试自动化框架中最受欢迎的框架，它专门专注于自动化 Web UI。对于移动用户界面，您需要类似的框架 Appium。还测试 API、数据、打包的应用程序等吗？这意味着需要获取、配置、学习和链接更多的工具和框架。

现在，让我们退后一步，记住自动化的最终目标：加快测试速度，以便可以根据需要快速而频繁地执行测试。为此，你需要一种测试自动化方法，使您的测试团队能够为您的应用程序快速构建端到端测试自动化。

如果您的测试团队由脚本专家组成，并且你的应用程序是一个简单的 Web 应用程序，那么 Selenium 或基于 Selenium 的免费工具可能更适合你。如果你的团队由业务领域专家主导，并且你的应用程序依赖于广泛的技术组合，那么你可能需要一种测试自动化方法，该方法可以简化测试企业应用程序的复杂性，并使典型的企业用户能够通过最小的学习曲线。

你可能会发现组织的不同部门喜欢不同的方法（例如面向客户界面（如移动应用程序）的团队可能不希望与后端处理系统的团队使用相同的测试方法。只需确保所有方法

和技术以促进协作和重用的方式连接，同时提供集中的可见性即可。

重要注意事项

这对于在涉及多种技术的复杂企业环境中进行测试最为重要，例如打包的应用程序以及 API、ESB、Web 和移动设备。你要测试的接口越多，你应该对它们进行优先级排序。如果你是测试单个界面的小型团队，那么这可能对你来说不是问题。

结束测试维护噩梦

如果你的测试难以维护，则测试自动化计划将失败。如果你真正致力于检查脆弱的脚本，那么你将在测试维护中投入大量时间和资源——侵蚀了测试自动化所承诺的节省时间，并使测试（再次）成为过程瓶颈。

如果你不是 100％致力于维护测试，则测试结果将被误报（和误报）困扰，以至于测试结果不再受信任。

维护问题源于两个核心问题:

- 测试不稳定
- 难以更新的测试

解决不稳定问题的关键是找到一种表达测试的更可靠的方法。如果在未更改应用程序的情况下自动化测试开始失败，则说明你遇到了稳定性问题。

有很多技术解决方案可以解决此问题（例如使用更稳定的标识符）。这些策略对于掌握至关重要。但是，从测试自动化计划的一开始就考虑测试稳定性也很重要。在评估测试自动化解决方案时，请密切注意该工具如何响应可接受和预期的变化以及需要多少工作才能使该工具与不断发展的应用程序保持同步。此外，请注意，即使是最稳定的测试，如果它们使用不合适的测试数据或在不稳定或不完整的测试环境中运行，也会遇到问题。

为了解决更新问题，模块化和重用是关键。开发团队每次改进或扩展现有功能时（现在可以每天、每小时甚至更频繁地进行），你都负担不起更新每个受影响的测试的费用。为了使测试与开发保持同步所需的效率和“精简性”，应从易于更新的模块构建测试，这些模块可在整个测试套件中重复使用。当业务流程更改时，你希望能够更新单个模块并自动同步受影响的测试。

重要注意事项

对于希望实现高度自动化水平的团队和积极开发应用程序的团队而言，此策略至关重要。如果要针对相对静态的应用程序自动化一些基本测试，则可能有足够的时间和资源来解决所需的维护。但是，构建的测试自动化程度越高或应用程序更改的频率越高，越早进行测试维护将成为一个噩梦。

此外，快速增长和高周转的团队更容易受到“测试膨胀”的影响：大量的冗余测试在风险覆盖率方面没有任何价值，但仍需要执行，检查和更新的资源。专注于重用和应用良好的测试设计策略将使膨胀降至最低。

转向 API 测试

如今，UI 测试占据了功能测试自动化的绝大多数，只有一小部分的测试是在 API 级别上进行的。但是，对持续测试 Rainbow 的第二次观察表明，我们需要达到一种实质上相反的状态：

为什么？API 测试被广泛认为更适合现代开发流程，因为：

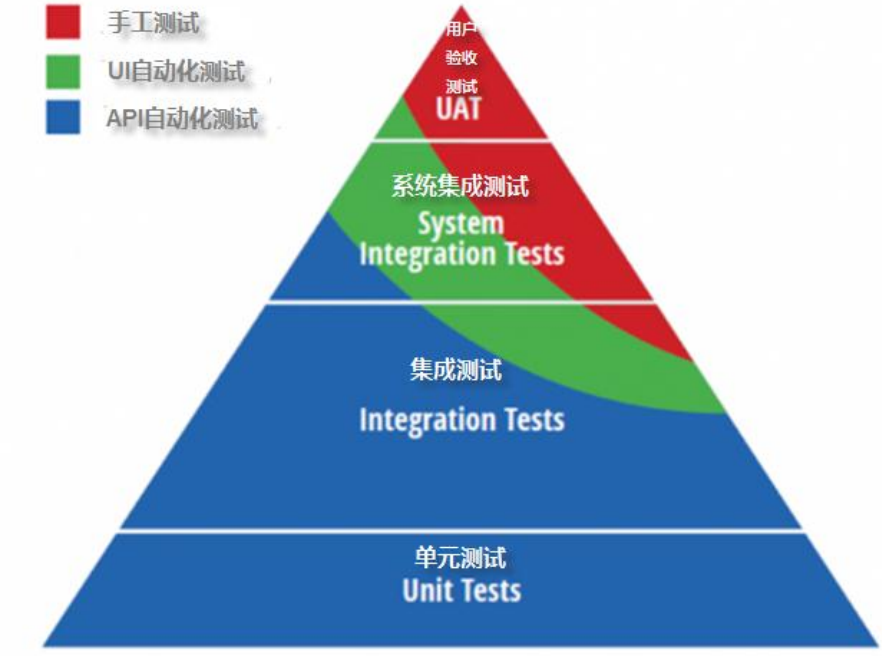
- 由于 API（“交易层”）被认为是与被测系统最稳定的接口，因此与 UI 测试相比，API 测试不那么脆弱并且更易于维护
- API 测试可以在每个 sprint 中比 UI 测试更早地实现和执行（此外，通过服务虚拟化模拟尚未完成的 API，你可以使用 TDD 方法向左移动测试）
- API 测试通常可以验证 UI 测试范围之外的详细“幕后”功能

- API 测试执行起来要快得多，因此适合检查每个新版本是否会影响现有的用户体验

实际上，Tricentis 的最新研究已经量化了使用 API 测试相对于 UI 测试自动化的一些关键优势：

任务	UI测试自动化	API测试	因素
设置	100%	25%	4x
维护	100%	16%	6x
运行时	100%	<1%	100+ x
定时	递减	递增	

这使我建议对测试金字塔采取以下措施：



金字塔的红色尖端表明了手动测试（通常通过探索性测试）最适合在现代开发过程中发挥的作用。绿线表示我们已被视为 UI 测试自动化的“最佳地点”。API 测试涵盖了三角形的绝大部分，它基于开发级单元测试。

顺便提一下，重要的是要认识到，随着时间的流逝，测试金字塔实际上会腐蚀成钻石。底部掉出，使金字塔变得不稳定，但是你可以采取一些措施来防止这种情况的发生。

从实际的角度来看，你如何确定应该在 API 层进行哪些测试以及应该在 UI 层进行哪些测试？一般的经验法则是，你希望尽可能接近业务逻辑。如果业务逻辑是通过 API 公开的，请使用 API 测试来验证该逻辑。然后，在需要验证可能会在设备，浏览器等之间变化的 UI 元素或功能的存在和位置的情况下，保留 UI 测试。同时，开发人员应在单元级别测试 API 的基础代码以暴露一旦引入实施错误。

重要注意事项

显然，如果你承担的测试功能没有通过 API 公开，那么这对你来说不是可行的策略。例如，如果要测试未利用 API 的 SAP 应用程序，则根本无法选择 API 测试。你需要以其他方式确保测试的可重复性和稳定性。

选择适合你需求的工具

市场上不乏开源和免费测试自动化工具。如果你要将测试自动化引入一个测试单个 Web 或移动界面或隔离的 API 的小型团队中，则可能会找到一个免费工具，该工具将帮助你入门并获得可观的测试自动化收益。

另一方面，如果你是一家大型企业，负责测试通过 SAP、API、大型机、Web、移动设备等进行的业务交易，则需要一个测试自动化工具，该工具可以简化所有这些技术的测试——使团队成员能够有效地重用并基于彼此的工作。

但是，在专注于选择工具之前，请考虑以下问题：组织在测试自动化计划中犯下的最大错误是，认为获取测试自动化工具是采用测试自动化的最重要步骤。不幸的是，这并不容易。无论选择哪种工具，都必须将其视为涉及流程、人员和技术的更广泛的转换的一个组成部分，这一点至关重要。

重要注意事项

不可否认，成本是每个工具采购决策中的一个因素。确保考虑总成本，包括培训和增加现有资源（或雇用其他资源），构建测试框架以及构建和维护测试所需的资源。

另外，请认识到让不同的团队使用不同的工具是完全可行的（并且通常很有价

值)。一个为年度公司活动创建移动应用程序的小型团队不需要使用与测试频繁升级如何影响基于 SAP 的业务关键交易的团队相同的工具。“单一窗口”报告提供集中的可见性，同时允许每个团队和部门选择适合其需求的最佳工具。

Perfmon 性能测试监控操作实践

◆ 作者：桃 子

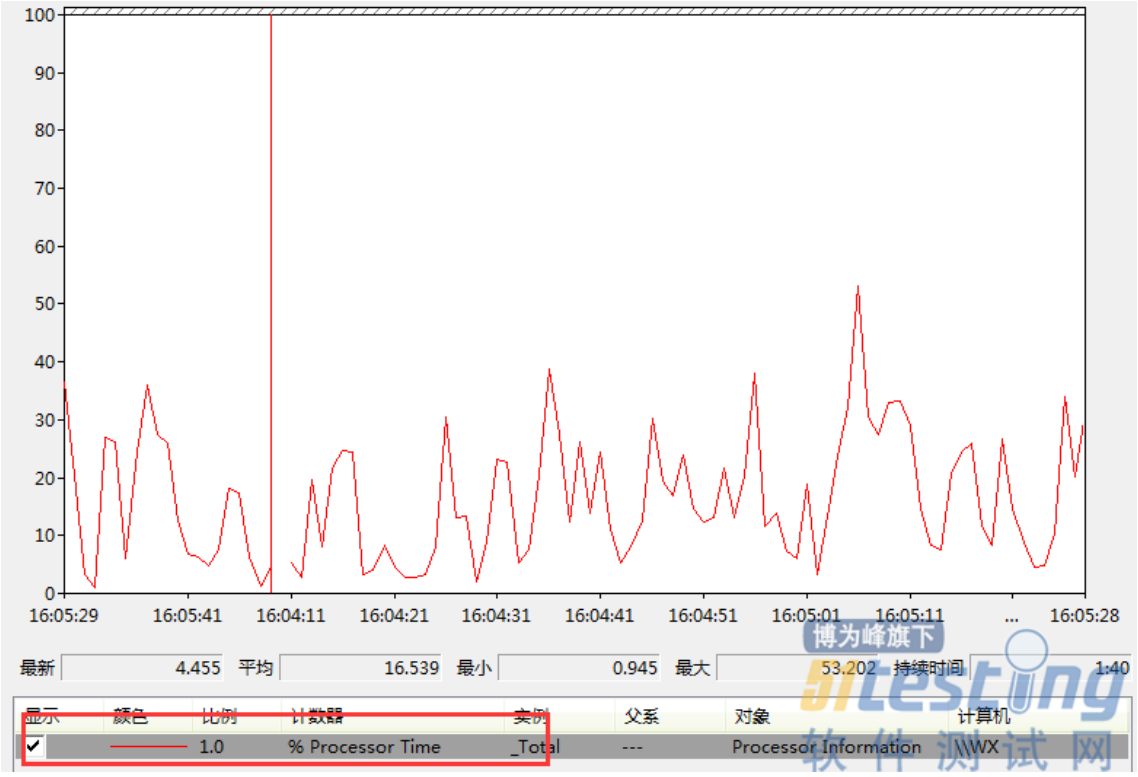
Perfmon 是 windows 自带的系统检测工具，可以用于监视 CPU 使用率、内存使用率、硬盘读写速度、网络速度等。

本文描述 Perfmon 工具的使用，性能数据的分析指标两部分

一 、操作步骤

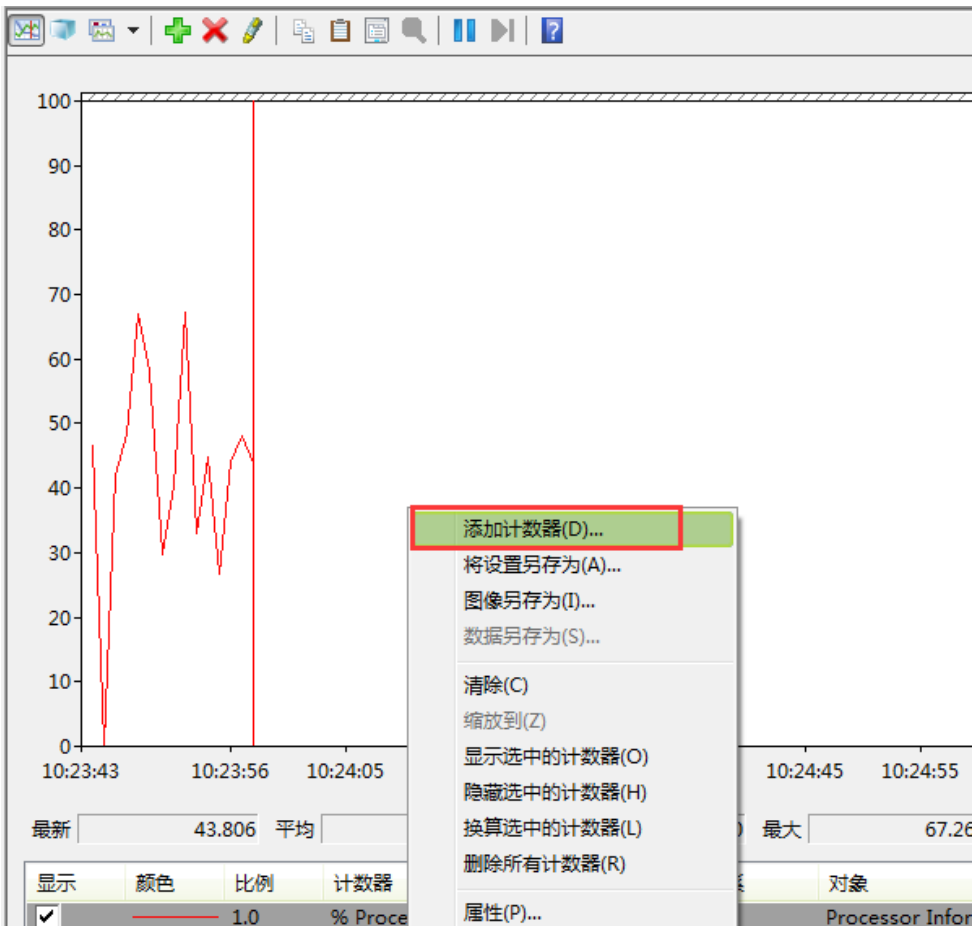
1、打开 cmd 输入 Perfmon 回车

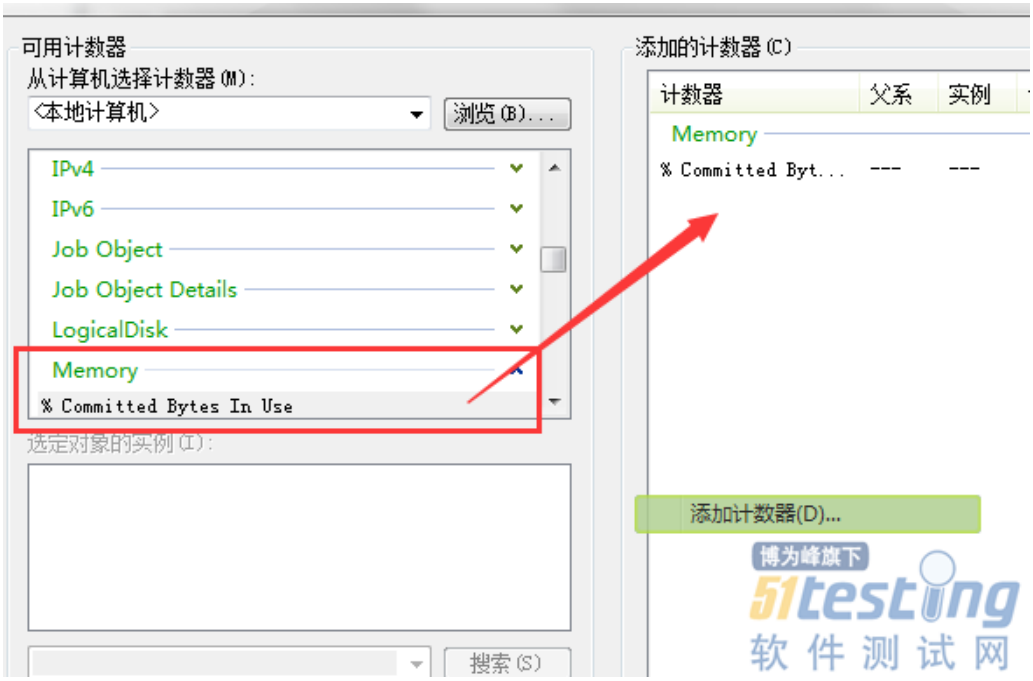
2、点击性能监视器，运行一段时间默认显示 cpu 占用率



3、添加内存百分比计数器

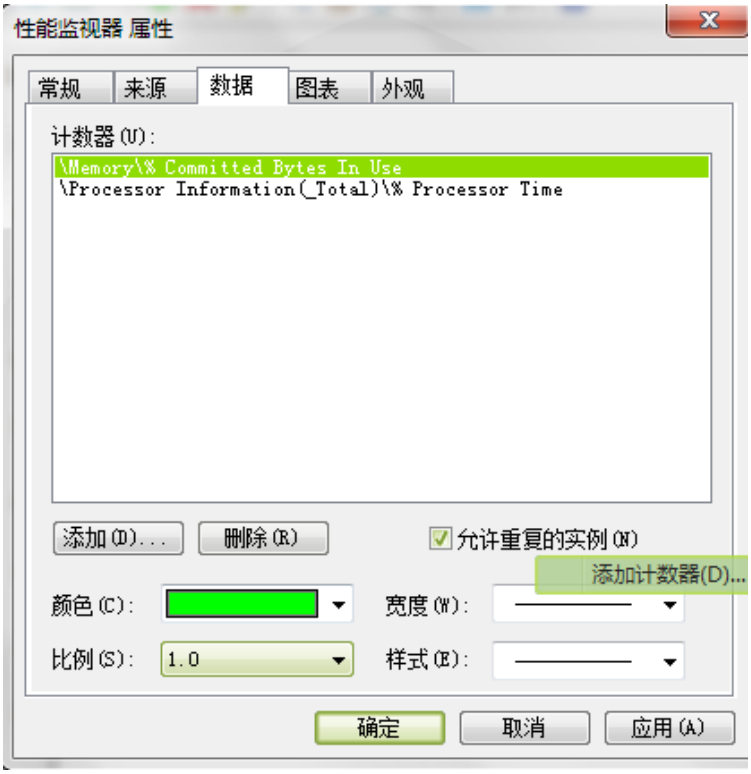
右键->添加计数器->选择 memory 下的%commited use in bytes

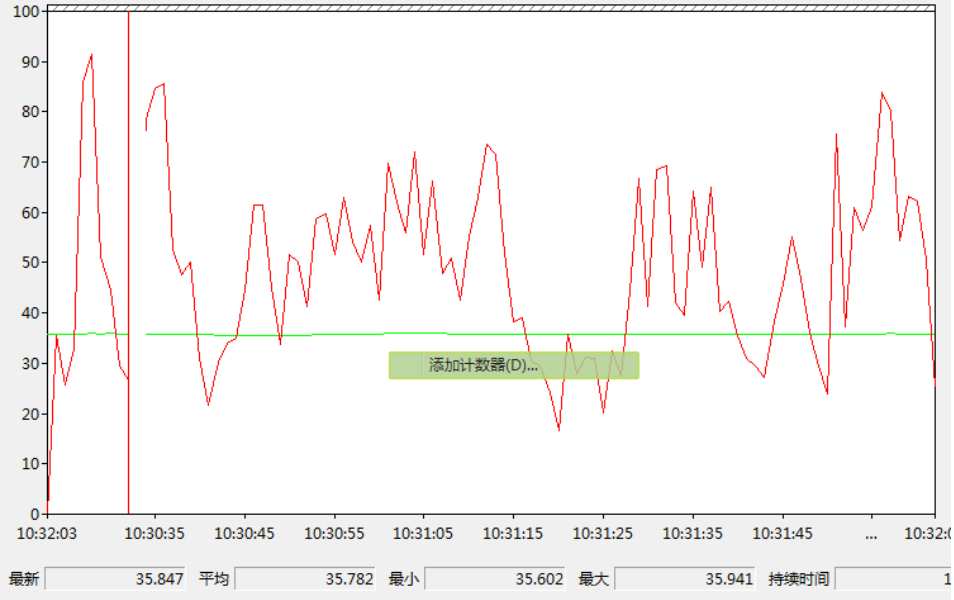




修改显示颜色

双击显示记录，属性框选择其他颜色后确定





4、添加网络计数器

同步骤 3（network interface-bytes total/sec）

5、添加磁盘驱动器

同步骤 3(pysical disk-%busy time)

二、分析数据

1、内存法

如果 memory 下的%commited use in bytes 数值比较大，说明内存可能存在问题，需要进一步查看下面几项:

page/sec　持续高于几百可能有内存问题

pages fault/ sec　每秒发生页面失效次数　次数越多，内存读取数越多需要查看 pages read/ sec

pages read/ sec　值超过 5 说明内存方面存在问题

2、处理器分析法

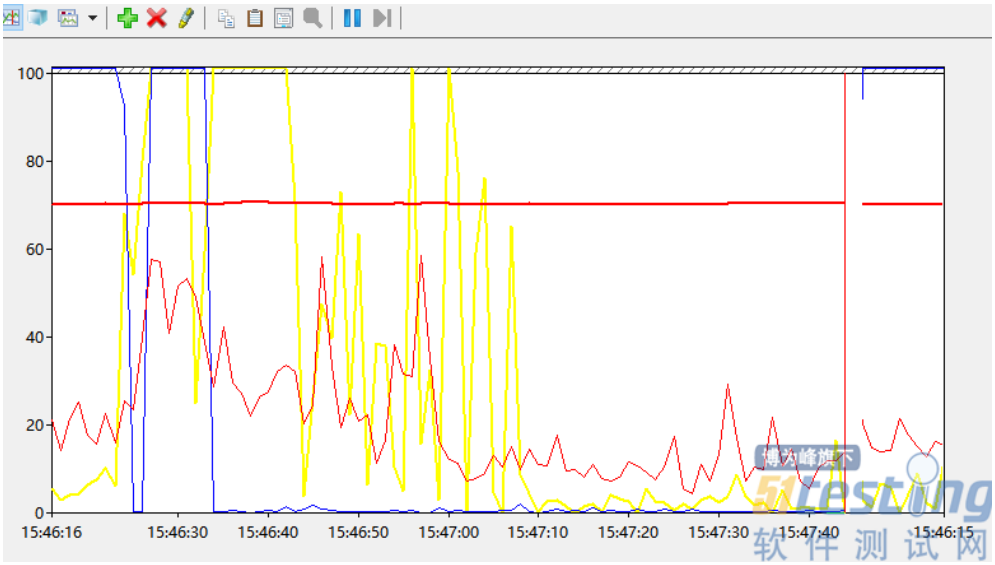
查看% processor time　持续超过 90%，说明面临性能方面瓶颈

3、磁盘 I/0 分析方法

如果.磁盘 I/0 数超过磁盘 I/0 处理能力，则存在磁盘系统瓶颈

4、网络分析法

network interface-bytes total/sec 数和网络带宽对比 ，如果较大说明面临性能方面瓶颈



举个简单的例子，从网络上下载文件如 qq 文件，下载期间查看各项指标参数

图中一条平直的红线是内存，没有什么变化，表明正常

上图蓝色代表网路计数器，黄色代表磁盘。从图中可以看出下载文件期间，网路计数器、磁盘值较大，下载完成后各项指标正常，只说明下载文件这段期间性能有问题

具体指标还要结合实际项目分析，希望对大家有所帮助！

《51 测试天地》(五十八) 下篇 精彩预览

- Selenium 自动化中的常见挑战
- Yapi 接口自动化平台技术调研
- 详细解读 Cypress 的测试报告
- GT 性能测试工具使用实践
- 基于软件性能效率的第三方测试需求分析思路总结
- 使用 Nexus 搭建 Maven 私有库
- 我是如何实现工资翻倍成长的？
- 疫情下的寒冬，何时才会消退

马上阅读