# Zabbix & Activiti

## 简介：

开源监控系统：有Zabbix，nagios，ganglia，Cacti等。选用Zabbix的主要原因是这个简单。安装，使用简单，功能全，all in one的设计。

开源工作流：工作流科学工作流和商业工作流，在【工作流综述】中我已经详细的讲了这两个工作流的区别。

## 环境：

基础运行环境：Ubuntu或任何linux发行版，mysql，php（目前用的是7.0）

目前的运行版本是Ubuntu16.10.内核版本是4.8.0-59-generic，gcc version 6.2.0。

Zabbix可以从官网上获得中文版的说明文档，Ubuntu可以通过加载源并用apt install 来快速安装。

Activity是基于java运行的。官方文档上给出的需求是JDK6以上。因为是在Ubuntu中，所以我使用apt install openjdk来安装。目前版本：

openjdk version "1.8.0\_162"

OpenJDK Runtime Environment (build 1.8.0\_162-8u162-b12-0ubuntu0.16.04.2-b12)

OpenJDK 64-Bit Server VM (build 25.162-b12, mixed mode)

Activiti的版本是6.0。可以从官网下载压缩包，解压之后有三个war包，我是直接用的这些war包，如果使用Activiti做二次开发的话，需要解压这些war包并利用里面的jar包。这三个war包可以部署在tomcat中，而且我也是这么用的，所以还有一个tomcat的版本，我安装了两个版本的tomcat，9.0.6和9.0.7，我发现好像只能在9.0.6中运行，原因未知。

Tomcat中部Activiti官网下载的三个war包，这个很简单。仅仅是拖拽进去就行了。然后修改其中的配置文件。在【Activiti使用文档】中有，几个文件的属性修改，并重启tomcat或在tomcat中进行reload。

## 目标：

我实现的目标是：在Zabbix中发现问题后，可以启动工作流，在工作流中对Zabbix进行控制，启动脚本或者变更监控项等操作。增加Zabbix的智能化，目标是我们可以根据有限的处理方法，让Zabbix 自己去分析问题，并自己去修正问题。在这个过程中主要涉及了Zabbix对Activiti的通信和Activiti对Zabbix的通信。

具体来讲：当一个监控节点出现了问题，我们会根据问题启动一个工作流，在启动工作流之后，会根据工作流的设计启动更加详细的监控项，利用更加详细的监控项来确定具体的问题，并且利用Zabbix的触发器自动化处理问题。所以在这个过程中，需要Zabbix启动工作流，Activiti对Zabbix中进行开关监控项的操作，在进一步确认中，Zabbix利用触发器配合响应动作对正在进行的工作流中完成进一步的确认，从而进一步推动工作流进行。

## 重要的资料：

Zabbix 官方文档：https://www.Zabbix.com/documentation/3.4/start

Activiti 官方文档：[https://www.Activiti.org/userguide/](https://www.activiti.org/userguide/)

相信我，这两个文档解决了90%以上的问题。

最好对于Activiti工作流有个大概的概念，学习资料：[https://github.com/jennyzhang8800/Activiti-learn](https://github.com/jennyzhang8800/activiti-learn)

需要了解工作流是怎么工作的，知道设计，部署，启动这几个步骤。Activiti的环境简单了解一下就行。

## 重要的数据表：

Zabbix中重要的数据表：

* Hosts：该表格记录了所有的监控节点。每个节点有一个hostid，name等数据。Hostid是唯一的。
* Items：记录了所有的监控项，每个监控项对应一个hostid，在这个表格中，itemid是惟一的。
* History系列表格：history，history\_log等表格，history后面的单词代表在这个表格中存放的数据类型。Log就是记录Zabbix中相关的日志信息。

Activiti中重要的数据表：

* ACT\_ID\_USER:这个表格里记录了Activiti的所有用户。
* ACT\_RE\_DEPLOYMENT：这个表格里记录了所有的工作流部署。
* ACT\_RE\_PROCDEF:这个表格里记录了所有的部署，包括同一个工作流的各个版本，启动工作流时用的用的ID和KEY是这个表格里的ID和KEY。
* ACT\_RU\_TASK:这个表格里记录了所有的正在进行的工作流中正要进行的任务。
* ACT\_RU\_VARIABLE:这个表格里记录的是工作流中的变量。

## Zabbix对Activiti的通信：

Activiti官方文档中有相关利用rest接口启动和控制工作流的相关配置。在官方文档中的第13章。主要是构造一个json数据，并发送到一个地址中来完成控制。不过，根据你功能的需求，会有不同的url地址，而且有的地址是动态生成的，例如一个工作流实例的相关参数就是在一个动态的url中才能获得。另外，所有的REST资源都需要一个有效的Activiti用户在默认情况下进行身份验证。使用基本的HTTP访问身份验证即可。

另外，为了使Zabbix与Activiti能够通信，我写了两个两个程序。一个用来启动工作流和另一个用来对工作流中的某一任务进行确认。Zabbix在设计的时候就是正常的设置监控项，触发器以及相应的动作。在动作的设置的时候，要用Zabbix的宏，这样就可以根据每个监控节点有不同的启动参数。另外，在进行任务确认的时候要想办法区分各个不同的工作流实例。我目前的方法是在启动的时候加入全局变量，并在进行确认的时候核对该变量从而找到唯一的任务。

## Activiti对Zabbix的通信：

Zabbix提供了自己的API接口，并且也是使用json数据格式进行通讯，地址是http://company.com/Zabbix/api\_jsonrpc.php，同样，Zabbix中也提供了一系列的方法，这些方法同样也可以控制Zabbix，详情见官方文档。在用户身份的认证上有所不同，Zabbix使用的是token序列进行认证，获取token的方式同样是api中login方法，构造相应的json数据就可以了。Zabbix中可以设置一些脚本，这些脚本同样可以被API调用，并且根据已有的设置运行。

在Zabbix agent的配置文件中有个两个设置非常重要。一个是allowroot，这个决定了是否可以以root身份运行指令。还有一个remotecommand,这个配置决定了节点能否执行远程命令。这两个默认都是关闭的状态，需要配置开启。

对于构造json这个事情，各个语言都有自己的构造方法，构造的格式在官方文档中都有了。所以这部分很简单。

## Activiti工作流的设置：

### 工作流的设计

正经来讲，Activiti的工作流是在一个JAVA项目中，对Activiti的API进行调用，从而完成工作流的，这种情况常见于公司的OA系统。对于我的这个项目而言，我仅仅是用了Activiti的自带的web配置界面就完成了。这两者的差异主要是在工作流的一些配置上，具体的我也不太清楚，反正我自己用API配置的工作流引擎是不支持超时的，但是他们自带的工作流引擎就可以，我没有深探究。

我在设计、部署工作流的时候主要用的是Activiti-app这个war包。在部署这个war包之后，访问：[http://127.0.0.1:8080/Activiti-app](http://127.0.0.1:8080/activiti-app)登录。如图所示，有三个默认的应用，主要有用的是第一个APP以及工作流的设计，第三个是身份管理，可能偶尔会需要。

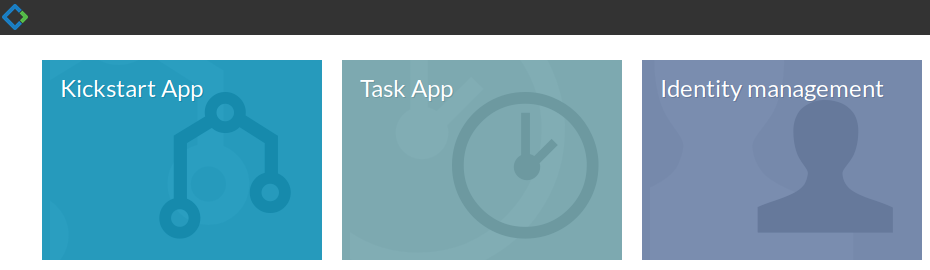


图 1

主要介绍一下第一个应用：

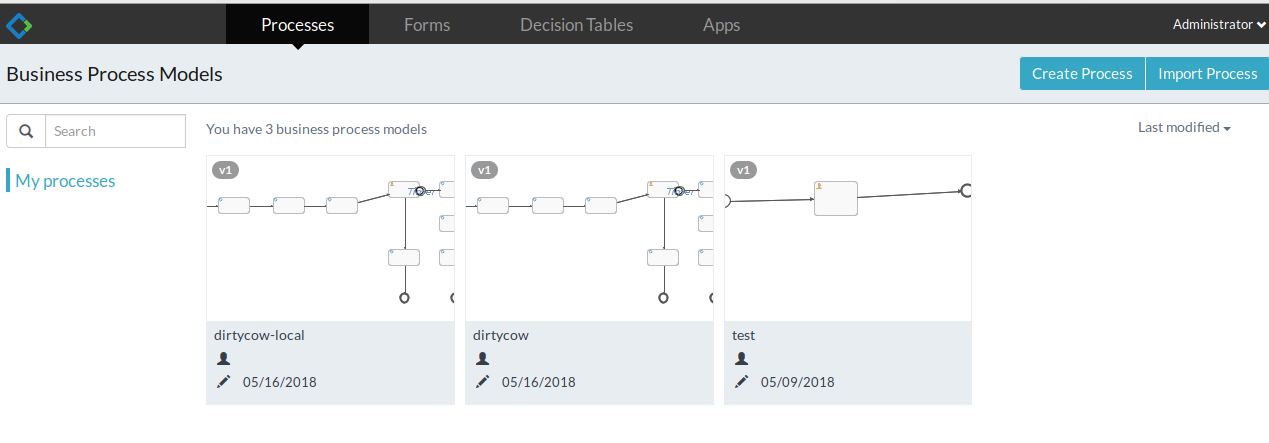


图 2

点进去之后可以看到目前已经设计好的工作流，也可以导入工作流，在导入工作流的时候用的是bpmn文件。

在创建工作流见面中，可以通过拖拽建立工作流。我常用的三个控件：

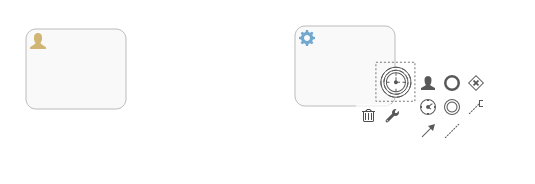


图 3

* 第一个，有人物标志的那个是。USER TASK，就是把这个任务分配给一个用户，这个用户可以在Idnetity management中设置。我使用的场景是这样的，在启动一个工作流后，设置这么一个USER TASK，目的是等待Zabbix端的进一步响应。在Zabbix 端启动一个脚本来对这个USER Task进行确认，从而推动工作流。Assignments这个属性用来设置分配到的用户。

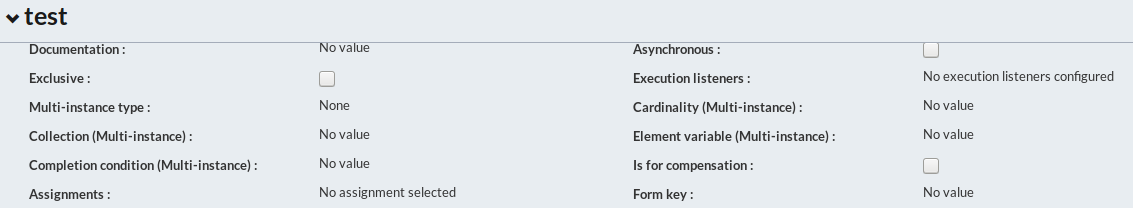


图 4

* 第二个齿轮状的是server task，可以让Activiti执行一段java程序，这段程序可以是自己写的，并且在程序中也可以获得整个流程的变量以及设置新的变量。这段程序可以在任何IDE中开发，在开发完成后，导出为jar包即可。把该jar包存入到/tomcat/webapp/activity-rest/WEB-INF/lib后，Activiti就可以在rest接口中正常使用这个jar包了，如果你想在Activiti-app中也使用这个的话就把jar包放入/tomcat/webapp/activity-app/WEB-INF/lib。我并没有测试这个玩意是不是异步的，但我感觉应该是同步的。Class属性是你写java时创建的packet以及函数。在Class fields中可以设置你启动函数的参数。



图 5

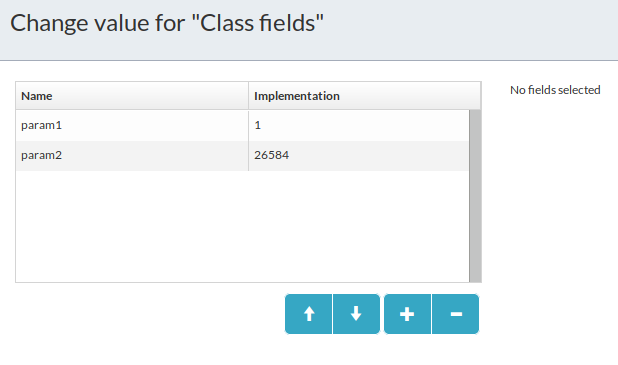


图 6

在这里我设置了两个参数。Param1和param2

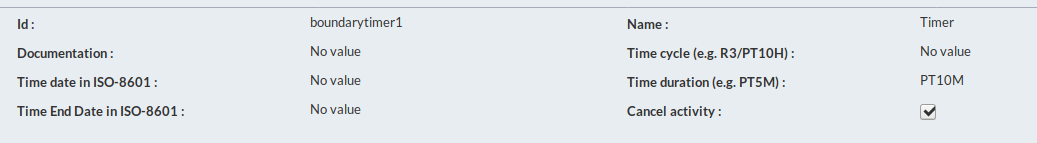
* 第三个，那个小钟表，那个是时间边界事件。这个有三个功能，轮询启动，定时启动，超时启动。我用的是超时启动，就是我设定一个时间，超过这个时间这个任务还没有结束则启动时间边界事件。重要的是Cancel activity，打钩的话就是代表一旦启动时间边界事件，则取消现在进行的活动，如果没有勾的话代表只是启动时间边界事件，继续当前的事件。关于时间的写法参见（其实还是在Activiti的使用手册中。）：[https://www.Activiti.org/userguide/#timerEventDefinitions](https://www.activiti.org/userguide/#timerEventDefinitions)

图 7

* 最后，server task其实还可以直接执行shell命令，但其实在Activiti-app中是无法设置的，所以不提了。如果有需求的话需要改以xml的形式打开bpmn文件去修改server task中的属性。

### 工作流的部署与启动：

在设计完工作流之后，可以按图二的Apps，出现以下界面：按create app，并加载刚刚设计的工作流，这就算是工作流的部署了，下一步就可以启动工作流了。

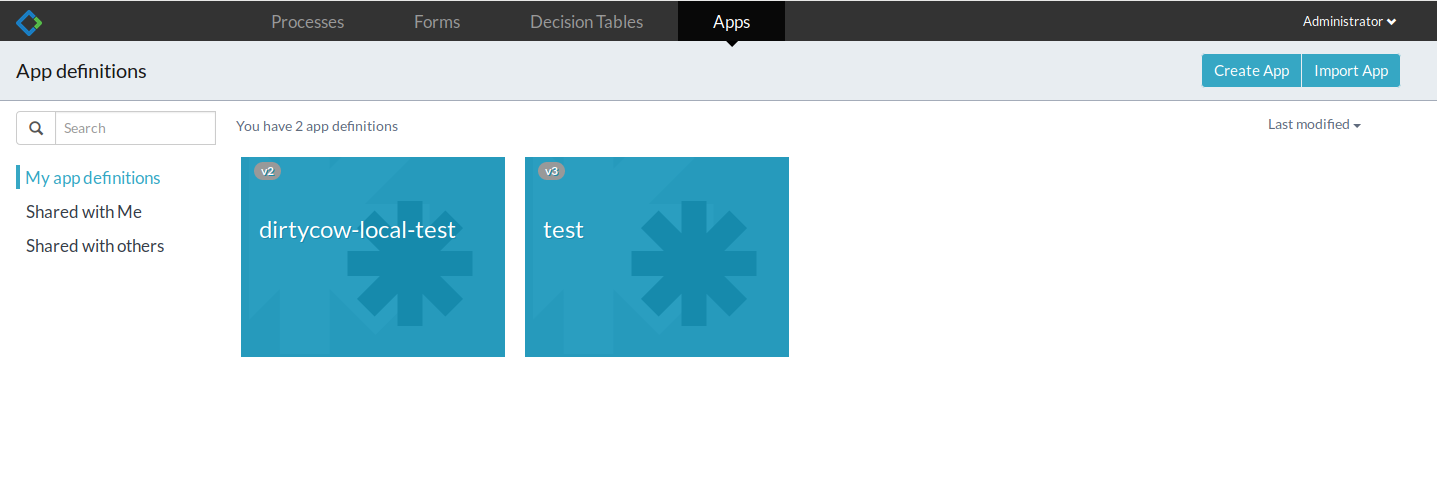


图 8

在Activiti-app启动工作流：返回图一可以看见刚刚创建的APP，点进去之后按Processes，然后就可以start a process。

在Activiti-rest中启动工作流：构造一个json，json中有工作流的key和ID，利用这个就可以启动工作流了。Key和ID可以从mysql数据库中直接查到。真的，官方文档贼有用。

https://www.Activiti.org/userguide/#\_start\_a\_process\_instance

