这节课说说怎么触发（trigger）robot去执行jobs，主要学习使用Jobs和Schedules的功能。

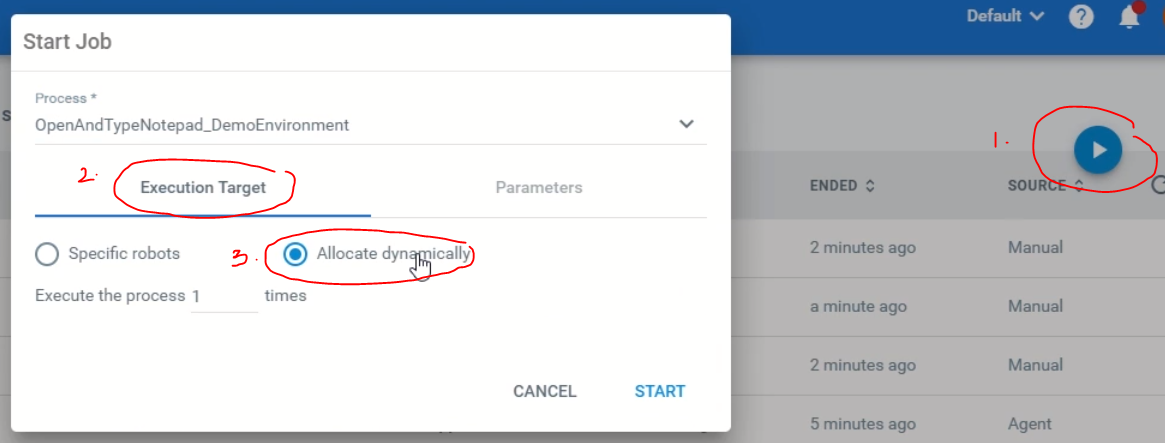
**1. robot的动态分配（dynamic allocation）**

在创建job的时候，可以选择动态分配robot。确切地说，如果我们想要某个process执行n次，在选择动态分配robot的情况下，process所在的环境（environment）里的所有robot只要空闲就会去执行该process，那些没闲着的robot一旦闲下来了也会马上去执行process，直到达到了要求的次数。

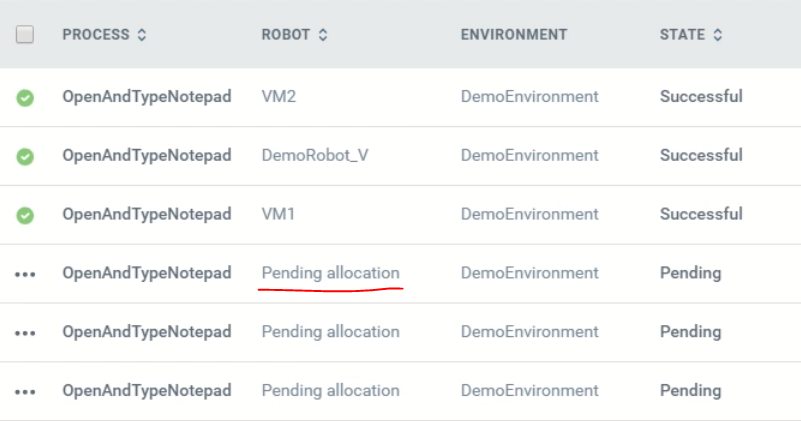
1) 在Orchestrator的Jobs页面创建一个新job。

2) 在Execution Target标签页下面选择Allocate Dynamically。

3) 输入process要执行的次数。



* 如果指定的次数超过了可用的robot数量，剩下的次数会处于等待分配（Pending Allocation）的状态，等待新的可用的robot，或者刚才的robot执行完一次process。

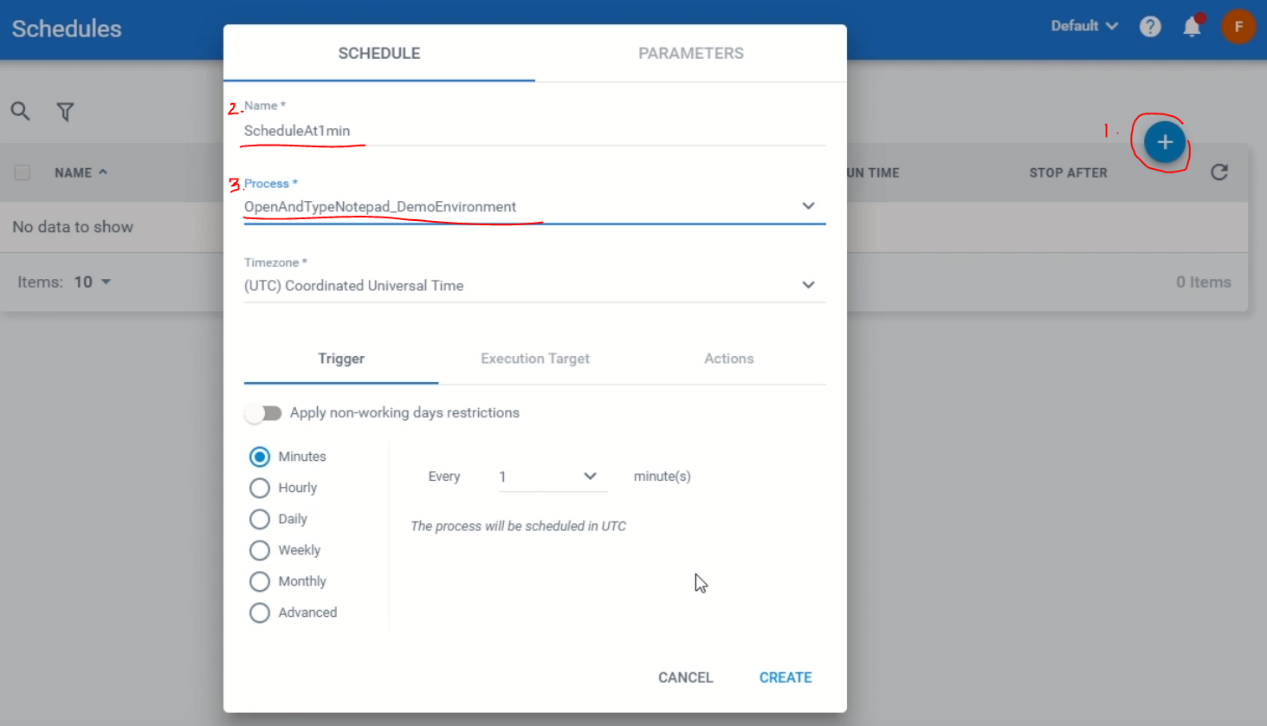


**2. 计划触发（trigger）一个job**

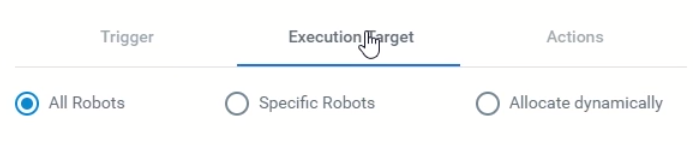
除了手动触发job，即创建一个新job，Part 1详细讲过，我们还可以计划（schedule）一个job。

1) 下面我们计划一个job，它将触发指定环境里的所有robot，每隔1分钟就执行某个process。

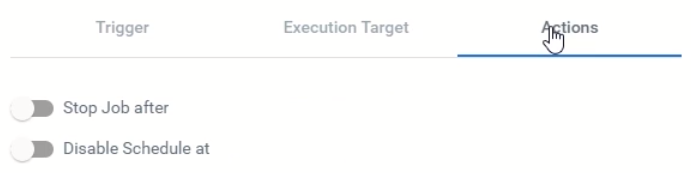
* 在Orchestrator的Schedules页面，点+号创建一个新的Job Schedule。
* 给Schedule起个名字，选择要安排执行的process，Timezone是选择时区的。
* 在Trigger标签页下面设定计划的时间。



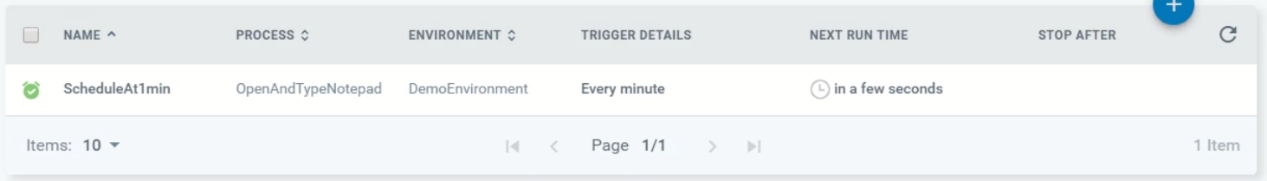
* 切换到Execution Target标签页，可以设置安排哪些robot，按照要求我们选择All Robots。



* Actions标签页下面也有一些功能。
  + Stop Job after可以设置在一段时间后停止计划。
  + Disable Schedule at可以设置在某个日期停止计划。



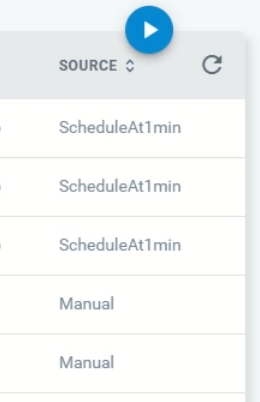
* 设置好了点Create，安排上了！



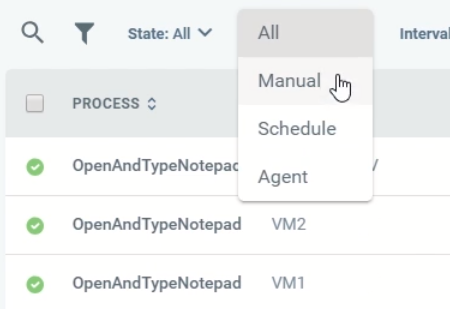
2) 查看触发的jobs。

现在我们回到Jobs页面，可以看到环境里的3个robots已经分别执行process了。一分钟后，三个robots会再次被触发，去执行process。

* 通过Source列可以看出某个job是通过什么方式触发的。

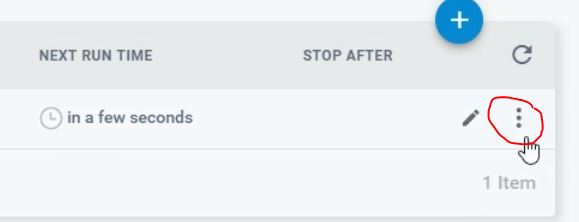


* 也可以使用筛选功能专门查看某种方式触发的jobs。
  + Manual：只显示手动触发的job，即在Jobs页面创建的jobs。
  + Schedule：只显示在Schedule页面安排的jobs。
  + Agent：只显示通过UiPath Robot直接触发的jobs。

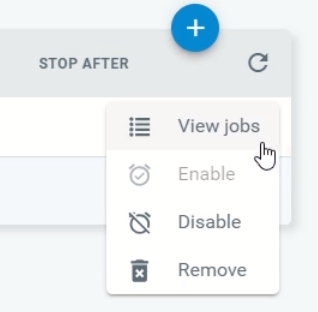


3) 停止计划或删除计划。

鼠标移动到创建的那条计划上，点右侧的三个点，会弹出操作选项。

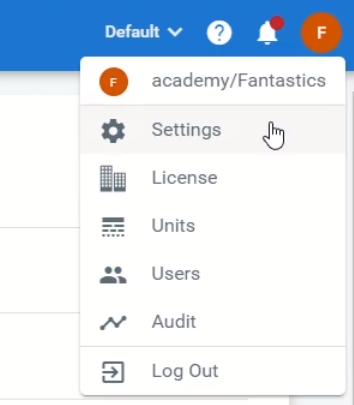


* Disable可以停止计划。如果要重新启动计划，可以选择Enable。
* Remove会直接删除计划，计划将从Schedule页面移除。
* View Jobs可以查看与这个计划相关jobs。

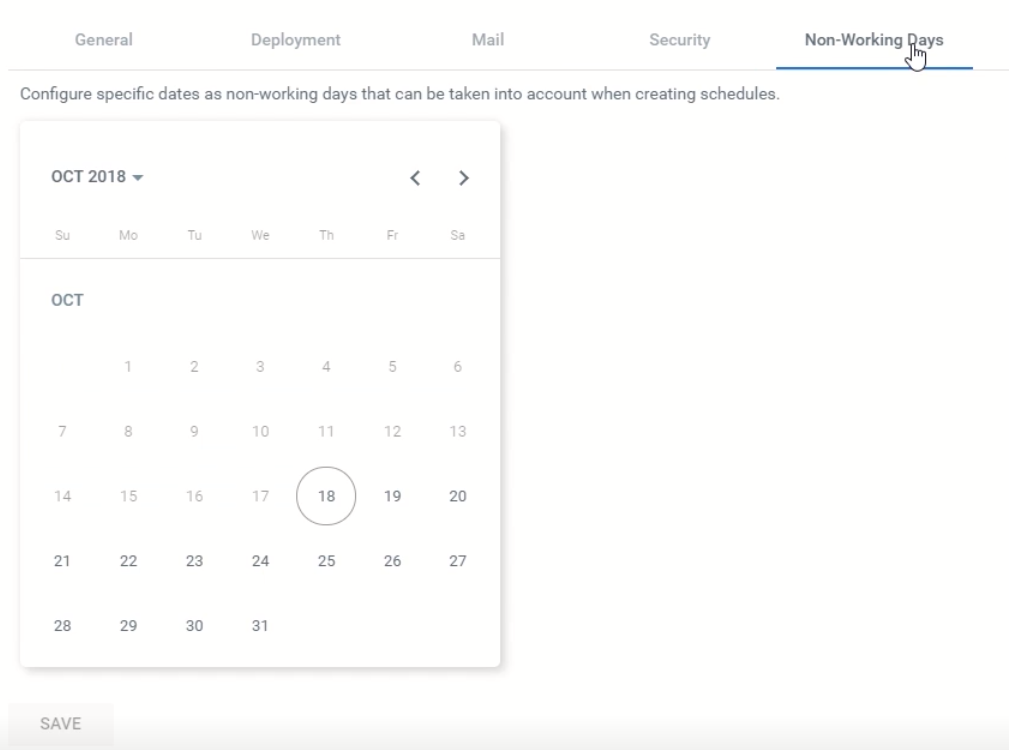


 4) 设置仅在工作日执行计划。

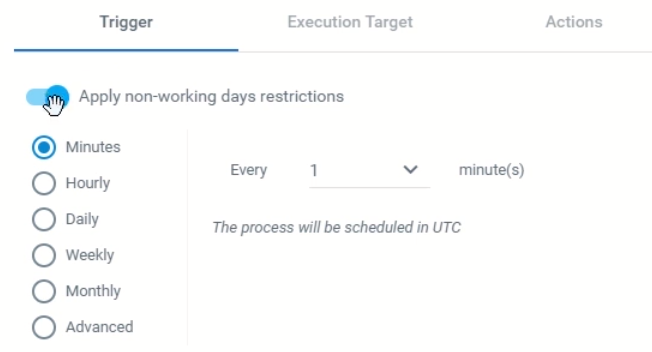
* 右上角点头像，选择Settings。



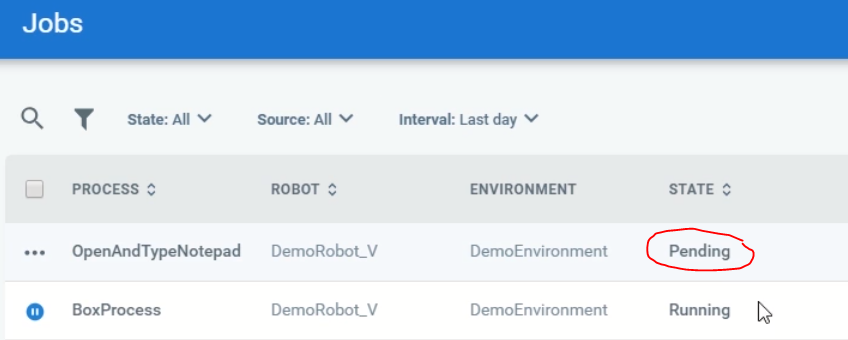
* 选择Non-Working Days标签页，在出现的日历里就可以点选非工作日。



* 在创建新计划的时候，在trigger下面选上Apply non-working days restrictions。

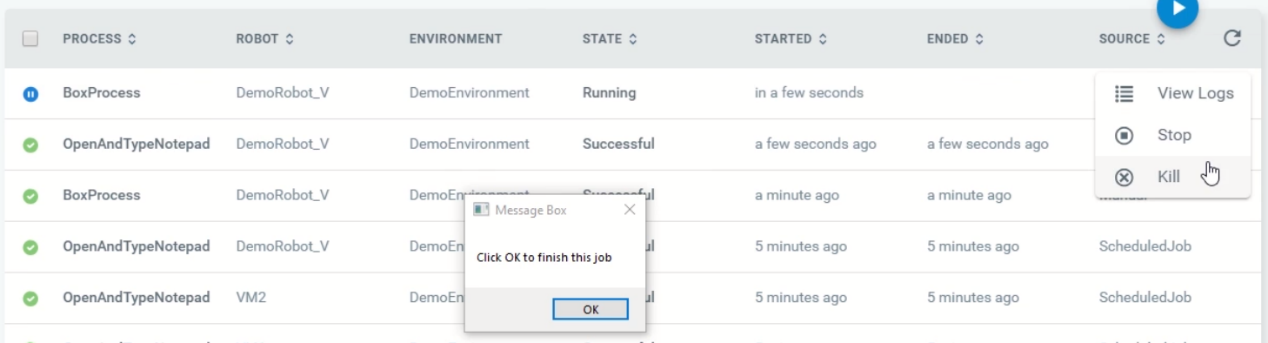


 5) 如果我们计划一个job，在计划触发job时，指定的robot正在执行其他process，那么计划的job会处于等待（pending）状态，排队等着robot空闲了再执行。



* 如果有很多个pending的job，它们会按照创建的时间顺序依次执行。

**3. 停止正在执行的job**



1) Kill会立即强行停止当前的job，忽略正在执行的步骤，停止后，该job的状态为Stopped。



2) Stop也可以停止job，但是必须配合Sudio里的**Should Stop活动**使用。

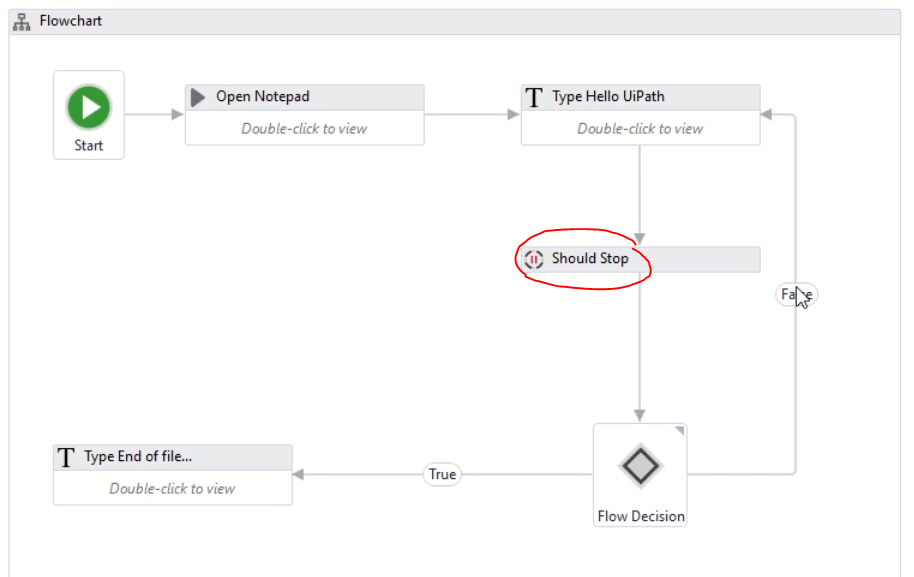
点击Stop后，job不会立即停止，robot会继续执行process，直到运行到Should Stop活动。

Stop和Should Stop活动使我们能安全地停止某个正在执行的process，我们应该确保在每个关键步骤前或后加上Should Stop活动。

使用Stop停止的job，其状态为Successful。

下面是一个示例流程：

* 创建一个循环，robot会不断在记事本写入Hello UiPath Orchestrator。
* 添加一个Should Stop活动，如果在Orchestrator里点了Stop则退出循环。
  + Should Stop活动返回一个布尔值（True/False），当我们没有在Orchestrator里选择Stop时返回False，当在Orchestrator里选择了Stop时返回True。
* 根据Should Stop活动的返回结果，robot继续写入文本，或退出循环并写入End of file。



分类: [Uipath Level II](https://www.cnblogs.com/TeresaMu/category/1428871.html)