Setting up the L&G test env on Azure

The url of the grid manager is the following:

|  |
| --- |
| proactive-scheduler.uksouth.cloudapp.azure.com |

The username and password are the usual:

* Username: tibcodev
* Password: gr1dDu5er

## Deploy a new release

Connect to the grid manager with:

|  |
| --- |
| ssh tibcodev@proactive-scheduler.uksouth.cloudapp.azure.com |

You need to make sure there is no running scheduler instance before removing it:

|  |
| --- |
| /tibco/proactive/tools/start-stop.sh stop  sudo chmod +w -R /tibco/proactive  Rm -Rf /tibco/proactive\* |

Then let’s clean the previous logs and release binaries (be careful not to remove the Miniconda and PyYAML files in the ProActive-pkgs folder):

|  |
| --- |
| lng\_clean\_batchlogs  rm /algodata/software/ProActive-pkgs/\*.zip |

To be able to download a release from jenkins into the grid manager, we need to get 3 things:

* Your jenkins username
* Your jenkins API Token
* The url of the release you want to use

The USER\_NAME and the API\_TOKEN can be found in your Jenkins user page:

1. Go to <http://jenkins.activeeon.com>
2. Log in
3. Click on your name in the top right corner
4. Click on the “Configure” item in the left menu
5. Search for the “Show API Token...” button
6. Reveal the credentials by clicking on this button



You can get the URL of the release you want by going to the release building job of your choice and right clicking on the artifact you want and select “Copy link” (or something similar)

At this point, we have everything to download the release with the following command:

|  |
| --- |
| wget --auth-no-challenge --http-user=USER\_NAME --http-password=API\_TOKEN --secure-protocol=TLSv1 URL |

Once the release has finished downloading, you need to move it to the /algodata/software/ProActive-Pkgs/ folder and remember its full path for later on. For example:

|  |
| --- |
| /algodata/software/ProActive-pkgs/Proactive-SNAPSHOT.zip |

If we need to get the latest code from the legalandgeneral repository, we can do so by calling the following bash convenience function:

|  |
| --- |
| lng\_copy\_from\_git <optionnel: nom de la branche> |

Optionally, this function can have a parameter which is the branch you want to use instead of the default master branch.

This command will git clone the legalandgeneral repo, modify some environment specific stuff (have a look at the function in the ~/.bashrc file if you want to see for yourself).

Then, we need to modify the build.sh script so it references our release:

|  |
| --- |
| chmod +w ~/THE\_BUILDING\_FOLDER/build.sh  vi ~/THE\_BUILDING\_FOLDER/build.sh |

You’ll need to modify the RELEASE\_NAME variable to use your release, so with my example, the RELEASE\_NAME will become:

|  |
| --- |
| RELEASE\_NAME=Proactive-SNAPSHOT |

**Notice that I removed the extension of the zip file !**

Then we can run the build.sh script:

|  |
| --- |
| cd ~/THE\_BUILDING\_FOLDER  ./build.sh |

If no error occurs, you should have a new file: proactive.tar.gz

If an error occurred, it means the build.sh file has changed on L&G side, we need to impact the environment -> SHOUT for help

Next, move that file to the /tibco folder:

|  |
| --- |
| mv ~/THE\_BUILDING\_FOLDER/proactive.tar.gz /tibco  cd /tibco  sudo chmod +w -R proactive  rm -Rf proactive  tar xvfz proactive.tar.gz |

Before running the scheduler, make sure there is no running instances of the Scheduler.

Then, run the install.sh script:

|  |
| --- |
| cd proactive  ./install.sh |

If everything goes well, you should have the grid-manager up and running in a few minutes. If something goes wrong -> SHOUT for help

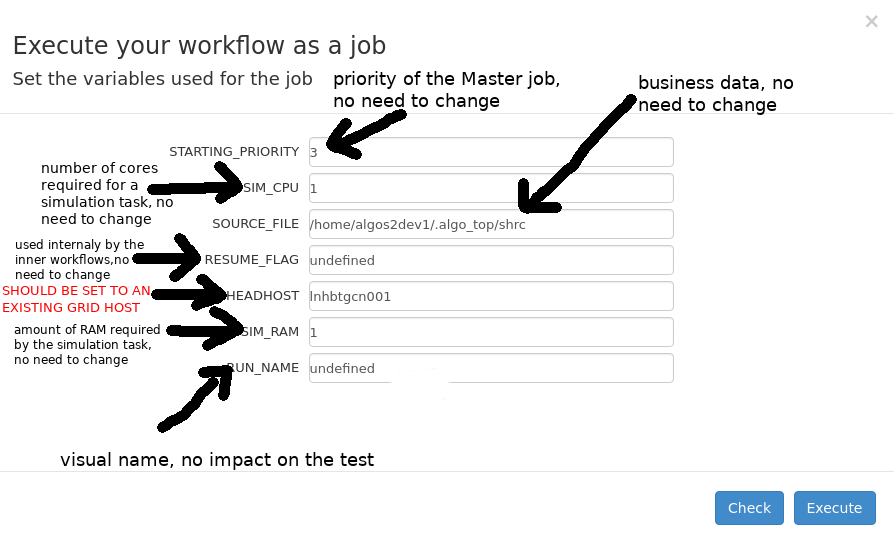
Finally, you need to restore the L&G workflows:

|  |
| --- |
| lng\_restore\_workflows |

Before submitting the Master workflow, switch to the small batch (3 simulations):

|  |
| --- |
| lng\_use\_small\_batch |

Then connect to the studio with algos2dev1/alg0Dev1 and submit the ${RUN\_NAME}.MASTER workflow with the following parameters:



The HEADHOST parameter should be set to one of the existing grid hosts name, for example: **lnggridhostuksouth**

If either the SIM\_CPU, SOURCE\_FILE, HEADHOST or SIM\_RAM are not set properly, the batch will hang forever.

Once that batch succeed, you can switch to the big batch (+15 simulations):

|  |
| --- |
| lng\_use\_big\_batch |

**Don’t forget to use algos2dev1/alg0Dev1 to submit jobs !**

## Deploy the workflows

Sometimes the client will update the workflows on his side, and send them to us so we can test our changes + his.

After testing the previous workflows, you should backup them by using the following command:

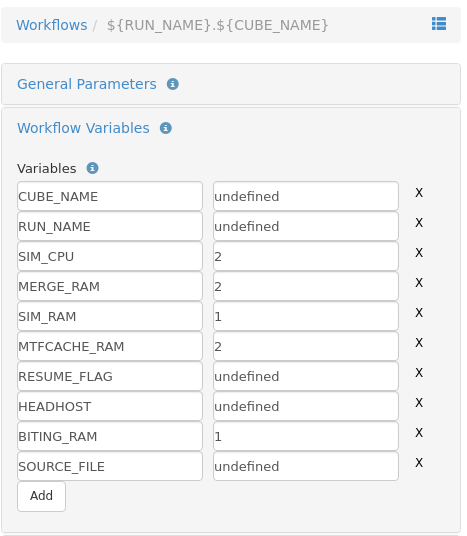
|  |
| --- |
| lng\_backup\_workflows |

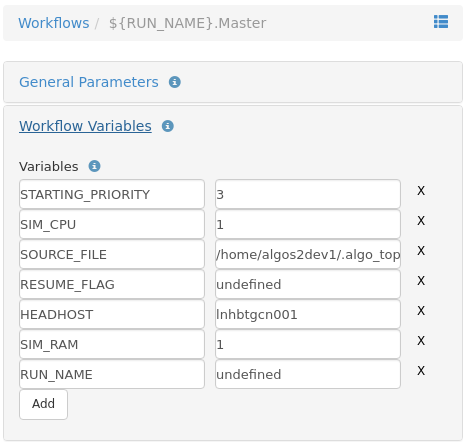
This command will copy algos2dev1’s workflows into:

|  |
| --- |
| /home/tibcodev/BACKUP\_WFS |

Connect to the studio using algos2dev1,

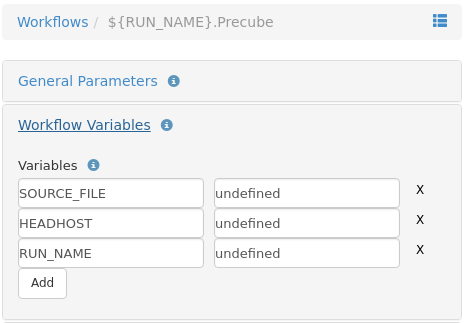
Then, because we still have some slight differences between their environment and ours, we need to adjust some workflows parameters. So for all our workflows, we’ll keep the workflows parameters around:

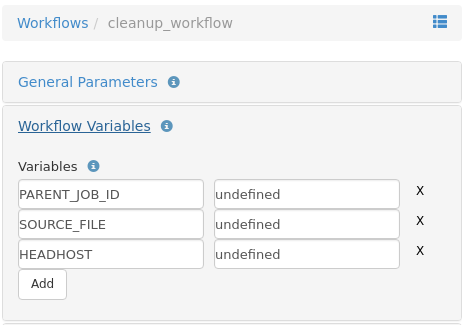




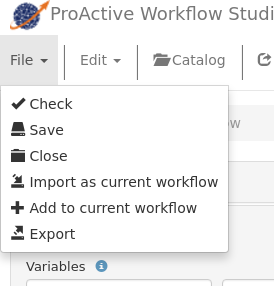
Complete line for SOURCE\_FILE:

|  |
| --- |
| /home/algos2dev1/.algo\_top/shrc |





Then on each of those workflows, we’ll replace the content with the new versions of the client by using the File -> Import as current workflow sub menu:



Finally, replace those new workflows parameters with the ones from the screen captures.

Once this is done, resubmit the Master workflow.



## Setting up the metrics and log collectors (Optional)

This has already been done and is here only to explain how the collectors have been started.

### Get\_jvm\_ram\_usage (for the SchedulerStarter or RMNodeStarter processes)

|  |
| --- |
| ./bin/get\_jvm\_ram\_usage.sh |

|  |
| --- |
| influxdb\_url='http://monitoring.ap.activeeon.com:8086/write?db=telegraf' influxdb\_username='admin' influxdb\_password='pouet123' host='lng-aws-grid-manager' while [ 1 -eq 1 ]; do  schedulerStarter=$(top -b -n 1 -c -w 10000|\  grep SchedulerStarter|\  grep -v grep|\  grep -v awk|\  sed 's/,/./g'|\  awk '{print $6}')  echo \[raw\] schedulerStarter=$schedulerStarter  if [[ $schedulerStarter == \*"g"\* ]]; then  schedulerStarter=$(echo $schedulerStarter | sed 's/g//g' | awk '{print $schedulerStarter\*1000}')  elif [[ $schedulerStarter == \*"m"\* ]]; then  schedulerStarter=$(echo $schedulerStarter | sed 's/\..\*m//g')  else  schedulerStarter=$(awk "BEGIN {print $schedulerStarter/1000}" | sed 's/\..\*//g')  fi  echo \[in Megabytes\] schedulerStarter=$schedulerStarter  schedulerStarter=$(echo $schedulerStarter | awk -v telegraf\_host=$host '{print "SchedulerStarter,host="telegraf\_host" jvm\_ram\_resident="$1}')  echo \[to POST\] $schedulerStarter  echo $schedulerStarter |curl -i -XPOST $influxdb\_url -u $influxdb\_username:$influxdb\_password --data-binary @-;  # rmnodeStarter=$(top -b -p $(jps|grep RMNodeStarter|awk '{print $1}') -n 1 -c -w 10000|\ # grep java|\ # grep -v grep|\ # grep -v awk|\ # sed 's/,/./g'|\ # awk '{print $6}') # echo \[raw\] rmnodeStarter=$rmnodeStarter # if [[ $rmnodeStarter == \*"g"\* ]]; then # rmnodeStarter=$(echo $rmnodeStarter | sed 's/g//g'| awk '{print $rmnodeStarter\*1000}') # elif [[ $rmnodeStarter == \*"m"\* ]]; then # rmnodeStarter=$(echo $rmnodeStarter | sed 's/\..\*m//g') # else # rmnodeStarter=$(awk "BEGIN {print $rmnodeStarter/1000}" | sed 's/\..\*//g') # fi # echo \[in Megabytes\] rmnodeStarter=$rmnodeStarter # rmnodeStarter=$(echo $rmnodeStarter|awk '{print "RMNodeStarter,host=$host jvm\_ram\_resident="$1}') # echo \[To POST\] $rmnodeStarter # echo $rmnodeStarter |curl -i -XPOST $influxdb\_url -u $influxdb\_username:$influxdb\_password --data-binary @-; #  sleep 5; done |

### Telegraf (for the metrics collection)

|  |
| --- |
| docker run -v /home/tibcodev/monitoring\_conf/telegraf.conf:/etc/telegraf/telegraf.conf:ro -v /tibco/proactive/logs:/telegraf/logs telegraf |

|  |
| --- |
| [agent]  hostname = "lng-aws-grid-manager"  interval = "5s"  [[inputs.logparser]]  ## file(s) to tail:  files = ["/telegraf/logs/Scheduler.log"]  from\_beginning = false  name\_override = "scheduler"  ## For parsing logstash-style "grok" patterns:  [inputs.logparser.grok]  patterns = ["%{TABLE\_MON\_LOG}","%{JOBS\_MON\_LOG}","%{HIB\_MON\_LOG}","%{CLEANING\_TIME\_MON\_LOG}"]  #patterns = ["%{CUSTOM\_LOG}"]  custom\_patterns = '''  TABLE\_MON\_LOG %{TIMESTAMP\_ISO8601:logdate} %{WORD:threadname}\] %{LOGLEVEL:loglevel} %{DATA:classname}\] JobData \(All\)\: %{NUMBER:job\_data\_all:int}, JobData \(Finished\)\: %{NUMBER:job\_data\_finished:int,JobContent\: %{NUMBER:job\_c\  ontent:int}, TaskData\: %{NUMBER:task\_data:int}, TaskDataNotFinished\: %{NUMBER:task\_data\_not\_finished:int}, SelectorData\: %{NUMBER:selector\_data:int}, EnvironmentModifierData\: %{NUMBER:environment\_modifier\_data:int}, ScriptData\: %{NU\  MBER:script\_data:int}, SelectionScriptData\: %{NUMBER:selection\_script\_data:int}, TaskDataVariable\: %{NUMBER:task\_data\_variable:int}, TaskResultData\: %{NUMBER:task\_result\_data:int}, ThirdPartyCredentialData\: %{NUMBER:third\_party\_cred\_\  data:int}  JOBS\_MON\_LOG %{TIMESTAMP\_ISO8601:logdate} %{WORD:threadname}\] %{LOGLEVEL:loglevel} %{DATA:classname}\] \[SchedulerState\] pendingJobs\: %{NUMBER:pending\_jobs:int}, runningJobs\: %{NUMBER:running\_jobs:int}, finishedJobs\: %{NUMBER:f\  inished\_jobs:int}, allJobsActual\: %{NUMBER:all\_jobs\_actual:int}, allJobsComputed\: %{NUMBER:all\_jobs\_computed:int}  HIB\_MON\_LOG %{TIMESTAMP\_ISO8601:logdate} %{WORD:threadname}\] %{LOGLEVEL:loglevel} %{DATA:classname}\] \[HibernateStats\] deleteCount\: %{NUMBER:delete\_count:int}, updateCount\: %{NUMBER:update\_count:int}, insertCount\: %{NUMBER:ins\  ert\_count:int}, fetchCount\: %{NUMBER:fetch\_count:int}, loadCount\: %{NUMBER:load\_count:int}, flushCount\: %{NUMBER:flush\_count:int}  CLEANING\_TIME\_MON\_LOG %{TIMESTAMP\_ISO8601:logdate} %{WORD:threadname}\] %{LOGLEVEL:loglevel} %{DATA:classname}\] HOUSEKEEPING of jobs \[%{DATA:jobs\_id\_list}\] performed \(Hibernate context removal took %{NUMBER:removal\_from\_mem\_sec:\  int} ms and db removal took %{NUMBER:removal\_from\_db\_sec:int} ms\)  '''  # Read metrics about CPU usage  [[inputs.cpu]]  percpu = true  totalcpu = true  fieldpass = [ "usage\*" ]  # Read metrics about disk usage  [[inputs.disk]]  fielddrop = [ "inodes\*" ]  mount\_points=["/"]  # Read metrics about diskio usage  [[inputs.diskio]]  devices = ["sda1"]  skip\_serial\_number = true  # Read metrics about memory usage  [[inputs.mem]]  # no configuration  # Read metrics about swap memory usage  [[inputs.swap]]  # no configuration  # Read metrics about system load & uptime  [[inputs.system]]  # no configuration  #[[outputs.file]]  # Files to write to, "stdout" is a specially handled file.  #files = ["stdout"]  [[outputs.influxdb]]  # The full HTTP or UDP endpoint URL for your InfluxDB instance.  urls = ["http://monitoring.ap.activeeon.com:8086"] # required  username = "admin"  password = "pouet123"  # The target database for metrics (telegraf will create it if not exists).  database = "telegraf" # required  # Write timeout (for the InfluxDB client), formatted as a string.  timeout = "5s" |

### Fluentd (for the logs collection)

|  |
| --- |
| docker run -v /tibco/proactive/logs:/fluentd/logs -v /home/tibcodev/monitoring\_conf:/fluentd/etc -e FLUENTD\_CONF=fluentd-logs.ap.activeeon.com.conf paraita/fluentd-with-es-plugin |

|  |
| --- |
| <system>  log\_level debug  </system>  <source>  type tail  path /fluentd/logs/Scheduler.log  # exclude\_path <LOG\_FILES\_TO\_EXCLUDE\_1>  # exclude\_path <LOG\_FILES\_TO\_EXCLUDE\_N>  pos\_file /tmp/fluentd.pos  # time\_format %Y-%m-%dT%H:%M:%S.%NZ  format multiline  format1 /^(?<time>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2},\d{3})[ ]+(?<level>[^ ]\*)[ ]+\[(?<thread>[^ ]\*)\][ ]+(?<message>.\*)$/  time\_format %Y-%m-%d %H:%M:%S,%L  tag scheduler.\*  # format json  read\_from\_head true  </source>  <match scheduler.\*\*>  #@type elasticsearch\_dynamic  @type elasticsearch  hosts http://fluentd:pouet123@logs.ap.activeeon.com:9200  # hosts http://admin:pouet123@logs.ap.activeeon.com:9200  include\_tag\_key true  logstash\_format true  logstash\_prefix scheduler-lng  flush\_interval 5s  max\_retry\_wait 30  retry\_wait 10s  disable\_retry\_limit  num\_threads 1  slow\_flush\_log\_threshold 30.0  </match> |