# The QualiMaster infrastructure – Installation guide

The following steps are required to install the QualiMaster infrastructure in the cluster. The QualiMaster infrastructure consists of several components:

* The Apache zookeeper service representing a distributed central memory in terms of a virtual directory structure. Apache Storm and the QualiMaster infrastructure use this for storing and signalling important information. Typically it is recommended to have an odd number of zookeepers, at minimum 3 as they run a voting/quorum protocol.
* The Apache Storm stream processor consisting of
  + The central nimbus node as a command processor frontend. Typically, the QualiMaster infrastructure services are installed on the same machine.
  + The Storm services on the worker nodes, i.e., the supervisor service for controlling the work horses spawned automatically at runtime.
  + In addition, the logviewer on the worker machines can provide convenient access to the distributed execution logs and the central user interface provides access to statistics and to starting and stopping topologies. The latter functions shall be used only in emergency cases as the QualiMaster infrastructure must be informed about such activities through its own commands (relying on the underlying Storm comments).
* The QualiMaster infrastructure adds six components:
  + The libraries needed to run QualiMaster pipelines as Storm topologies. These libraries must be distributed over all nodes. We suggest using a distributed file system for this purpose.
  + The (optional) node monitor providing resource allocation information for (unused) cluster nodes.
  + The pipeline elements repository (an accessible Maven repository) storing algorithms and pipelines.
  + The central infrastructure services for controlling compute resources, monitoring them and adapting them
  + The command line interface for issuing commands to the infrastructure.
  + QM-IConf as graphical frontend for configuring the infrastructure and the pipelines before runtime as well as for viewing monitoring information and issuing infrastructure commands for individual pipelines.

For installing Apache zookeeper and Apache Storm, we recommend reading the respective documentations.

## Preparation

Create a distributed file system under /var/nfs

Be sure the following environment variables are set:

* QM\_HOME is set to /var/nfs/qm (since 0.5.0)
* QM\_LIBS is set to /var/nfs/libs/qm-libs
* PROVIDED\_LIBS is set to /var/nfs/libs/provided-libs
* STORM\_HOME is set to storm installation directory.

Further (JVM) environment settings considered by the infrastructure are:

* java.io.tempdir: default artifact location of the QM Infrastructure settings
* qm.base.dir: used by Jenkins to set the actual folder location with the test data
* qm.home.dir: home directory of qm installation containing qm.cfg and qm.infrastructure.cfg (since 0.5.0)
* qm.reasoning [true/false] (temporary): enable or disable reasoning for debugging (default: true)
* qm.adaptation.reasoning: [true/false] (temporary): enable or disable reasoning in rt-VIL / adaptation for debugging (default: true)
* qm.adaptation.debug: Write the actual frozen system states to files adaptation\_ in the folder specified by the configuration setting monitoring.logInfra.location
* qm.adaptation.debug: Write the actual frozen system states to files monitoring\_ in the folder specified by the configuration setting monitoring.logInfra.location
* user.home: by the Infrastructure CLI to find the qm.cfg file
* qm.log.dir: Internally set by the Infrastructure services (main.sh) to communicate the desired log directory to logback, i.e., qm.log.dir can be used in a logback configuration for the infrastructure.
* logback.configurationFile: For the infrastructure services (main.sh), the official logback variable for setting the location of the logback configuration file (if the locations provided by the infrastructure services are not sufficient).

**2. Binaries**

* Download the latest version of all QM infrastructure libraries from [FullInfrastructure](http://jenkins.sse.uni-hildesheim.de/job/FullInfrastructure/ws/FullInfrastructure/) or execute the build script in the related project. Basically, infra-….zip is required. If you need the jars with the timestamp as a part of the jar name for indicating the detailed version, please download infra-…-stamped.zip. Those zip files are structured to package two folders mentioned in the needed environment variables, i.e., the qm-libs and provided-libs. Both the qm-libs and the provided-libs are needed for the QM infrastructure while the provided-libs are used only for the pipeline running on Storm.
* Store it in the directory /var/nfs/libs.
* Unzip it.

**3. Modifications to Storm**

* Modify Storm script. In order to include the provided jars of the pipeline into the Storm classpath, you need to modify the Storm script by adding the directory of /var/nfs/libs/provided-libs to the Storm classpth. Basically, the Storm script is located in /storm installation path/bin. Please find the get\_classpath method in the storm script and add the code ret.extend(get\_jars\_full(“/var/nfs/libs/provided-libs”)) before the method returns.
* After modified Storm script as shown above, please do not forget to restart the entire cluster. To avoid manually restarting the cluster machine by machine, the Storm Restart Script in the adaptation layer might help.

**4. Infrastructure scripts**

* Download the recent infrastructure scripts from AdaptationLayer-scripts and store them in the directory /var/nfs/qm. Alternatively, the files may be stored on a Unix system under /etc/qualiMaster and /var/log/qualiMaster is considered as default logging directory (if it exists).

**5. Setup**

* Zookeepers:

Storm nodes as well as QualiMaster pipeline elements required connections to the Zookeepers. However, to prevent DNS attacks, Zookeeper by default limits the number of connections per machine. Depending on the physical layout of the pipelines on the cluster, this may cause individual parts to fail of the pipelines, on different clusters different parts, e.g., depending on the respective pipeline elements that don’t start, it may appear that some pipelines don’t start completely (which prevents the infrastructure from connecting the data sources), on pipeline elements considered as “optional” don’t start and “just” a reduction of performance may happen. For this reason, please check the zookeeper setting maxClientCnxns (default 10) and either set it to 0 to disable the limitation or in a cluster with local network setup or a number that is higher than (2+2\*task) per node in the tightest possible layout (depends on the number of pipeline elements, their lowest parallelization, the intended number of workers).

* Storm workers

XXX

* Storm supervisors

XXX

* Storm nimbus

XXX

* QualiMaster infrastructure: The setup of the QualiMaster infrastructure is defined by two files, qm.cfg (configuration of the infrastructure command line tool cli.sh) and qm.infrastructure.cfg (configuration of the infrastructure service and the node monitor) based on your cluster information. The infrastructure reads only qm.infrastructure.cfg while cli.sh reads first qm.infrastructure.cfg and then qm.cfg so that general settings are taken over but can be overridden, in particular, where the QualiMaster event bus is located allowing cli.sh to be executed on any machine with network access to nimbus, zookeepers and the event bus.
* Both files are by default located in the same directory as cli.sh and main.sh (the startup script for the infrastructure services), but they also can be located in the users home directory (for local installation in particular of cli.sh) or for a more Linux compliant installation in /etc/qm. To avoid confusion, it may be helpful to create symbolic links in the respective directories to one unified location, in particular for /var/nfs/qm/qm.infrastructure.cfg, but feel free to do that also for the qm.cfg:
  + For cli.sh this can be done by: ln -s /var/nfs/qm/qm.infrastructure.cfg ~/qm.infrastructure.cfg
  + The NodeMonitors searches first in /etc/qualiMaster, which does not exist and would need a local copy on each worker. Than it searches for configuration files in the home of qm, which is specified via the qm.home.dir property at startup. Currently, it is set to /var/nfs/storm/qm. The simplest solution is the creation of this folder and to link the configuration file there to the original (alternatively someone could fix the wring property). For instance:
    - mkdir /var/nfs/storm
    - mkdir /var/nfs/storm/qm
    - ln -s /var/nfs/qm/qm.infrastructure.cfg /var/nfs/storm/qm/qm.infrastructure.cfg
* In our installations, we run the infrastructure on the same machine than the nimbus, but this is not required. Therefore, the zookeeper and nimbus settings are repeated in qm.infrastructure.cfg rather than read from the storm configuration.
* What to modify:
  + set eventBus.host to the machine running the infrastructure (localhost only for qm.cfg or for local cluster tests)
  + set nimbus.host to the hostname/ip of the nimbus machine
  + set zookeeper.host to the hostname/ip of the zookeeper machine (multiple hosts can be given separated through commas)
  + Configure pipelines.ports to define the range of IP ports across the cluster reserved for communication among pipeline parts. Default value is "63000-64000".
  + optional: set monitoring.log.location to a local writable directory where the Monitoring Layer shall store monitoring logs
  + optional: set storm.shutdownEvents.time if the time needed for an explicit shutdown preparation of the executors is not sufficient (default 300ms, defers the final topology kill command).
  + optional: set adaptation.port (default 7012) or event.port (default 9998) in case of conflicts with your local port assignment
  + optional: set eventBus.disableLogging (default empty) to the comma-separated list of event classes that shall not be logged, applies to the infrastructure as well as to pipelines
  + set pipelines.settings to the folder where the pipeline settings / setup files shall be copied to on infrastructure startup (disabled by default)
* Adjust the logback.xml according to your needs.
* If you want to run the QM infrastructure as a service, e.g., via Linux supervisord, we need
  + supervisor.main.sh (which makes /etc/profile available to the supervisord process)
  + supervisor.qm.conf in /etc/supervisord/conf.d (then call sudo supervisorctl reread and sudo supervisorctl update)
  + Please note that the exec commands in the script are in place in order to pass the process id from Java to supervisord.

**6. Running the Infrastructure**

* Start the Zookeepers
* Start the Storm supervisors
* Start Storm Nimbus
* Start the QM Infrastructure
* How to use infrastructure commands(scripts):
  + Run main.sh under the same user as Storm is running or run the infrastructure as service using the same user as Storm..
  + We take the QualiMaster priority pipeline as an example to show the specific commands. Assumed that the name of the priority pipeline is 'PriorityPip' and one of its family elements called 'fSentimentAnalysis' has two alternative algorithms 'SentimentAnaylsisSentiWordNetTopology' and 'SentimentAnaylsisSVMTopology'. Further, an integer parameter 'timeSeriesGranularity' belongs to this family.
  + **Start Infrastructure.** In order to run the pipeline through the QualiMaster infrastructure, first it is required to start the infrastructure. In the infrastructure scripts directory of storm master node, launch the command: ./main.sh. This is not needed if the QM infrastructure is running as a service as described above.
  + **Start Pipeline.** When the infrastructure is running, you can start your pipeline with: ./cli.sh start <pipeline>. Note that the pipeline should be stored in the /var/nfs/qm/pipelines and the name of the pipeline jar should be the same as the topology name in the pipeline(the configured pipeline name). Therefore, to start the priority pipeline, the command is ./cli.sh start PriorityPip.
  + **Switch Algorithm.** When you want to switch the current algorithm to another one, you can run the command: ./cli.sh changeAlgo <pipeline> <pipelineElement> <algorithm>. Whereby, the pipeline is the configured name of the pipeline, the pipelineElement is the configured name of the family element where you want to change the algorithm and the algorithm is the configured name of the algorithm(not the actual implementation name!!). In the case of the priority pipeline, if the family element 'SentimentAnalysis' wants to change the algorithm to 'SentimentAnaylsisSVMTopology', the switching command will be: ./cli.sh changeAlgo PriorityPip SentimentAnalysis SentimentAnaylsisSVMTopology.
  + **Send Parameter.** When you want to send a parameter to your algorithm, you can run the command: ./cli.sh setParam <pipeline> <pipelineElement> <param> <value>. The pipelineElement is the same as in the switch command. The param is the name of the parameter and the value is the actual value of the parameter. For the priority pipeline, if you want to send an integer value 10 to the parameter 'timeSeriesGranularity' for the sentiment analysis family, the command will be: ./cli.sh setParam PriorityPip SentimentAnalysis timeSeriesGranularity 10.
  + **Stop Pipeline.** The command for stopping the pipeline: ./cli.sh stop <pipeline>. For the priority pipeline, it will be: ./cli.sh stop PriorityPip.
  + Here is the full help output of cli.sh (31/05/16):

QM infrastructure commandline (event client mode)

using:

- zookeeper: localhost @ 2181

- nimbus: localhost @ 6627

- eventServer: localhost @ 9998

- monitoring log location:

commands:

- start <pipeline>

starts the given pipeline

- stop <pipeline>

stops the given pipeline

- changeAlgo <pipeline> <pipelineElement> <algorithm> (param value)\*

changes an algorithm at runtime within the given pipeline

- setParam <pipeline> <pipelineElement> <param> <value>

param: WARMUP\_DELAY as java.lang.Integer, COPROCESSOR\_HOST as java.lang.String, CONTROL\_RESPONSE\_PORT as java.lang.Integer, CONTROL\_REQUEST\_PORT as java.lang.Integer, INPUT\_PORT as java.lang.Integer, OUTPUT\_PORT as java.lang.Integer, IMPLEMENTING\_ARTIFACT as java.lang.String

- rebalance <pipeline> <#workers> (<pipelineElement> <tasks>)\*

performs a Storm rebalance operation

- replay <pipeline> <sink> <ticket> <boolean> (<from> <to> <speed> <query>)?

starts/stops replaying data (boolean); if start, the further parameters shall be given

from/to as data longs, speed as int, query as string

- shed <pipeline> <pipelineElement> <shedder> (<paramName> <value>)\*

starts load shedding on the given element using the given shedder (class name or id)

with shedder specific parameters

- tracing <boolean>

enables/disables algorithm tracing for new pipelines, requires monitoring.log.location

- profile <family> <algorithm>

runs, monitors and profiles the given algorithm

- update

updates the infrastructure model (experimental)

- shutdown

shuts down the infrastructure (may consider Storm depending on the settings)