

Optimize your Local Machine for the Kiuwan Local Analyzer

How to optimize your local machine for the Kiuwan Local Analyzer (KLA)

Here are some things you should consider to properly configure your analyses.

Optimize the Java Virtual Machine

By default, KLA comes with pre-configured default values:

- max memory to use during every single step
- max duration time (timeout) of every single step

These parameters are configured through KLA configuration mechanism, please do not modify KLA scripts to include JVM flags such as -Xmx, use the mechanisms Kiuwan provides.



For Linux/Unix users

Please check /dev/random configuration of JVM. This may produce severe performance problems (java.urandom.Entropy).

Here is how to fix it: [Analyses are very Slow in Unix Linux, or Halt when Uploading Results to Kiuwan](#)

Single and parallel execution of analyses

Every step in an analysis with the KLA is executed sequentially, following the order below:

1. For each technology
 - a. rule analysis
 - b. metrics analysis
 - c. clone detection
2. Report generation, encryption and uploading to Kiuwan cloud

Every step is executed by a “new” JVM (Java 8 or above required) and the Kiuwan configuration applies to all of those JVM instances. If your source code contains more than one technology, it will execute each step for each technology.

If the machine running KLA analyses contains more available CPUs, you can run parallel analyses by executing additional instances of KLA.

In a parallel KLA execution scenario, every running analysis is completely independent from each other, so you can execute multiple analysis provided your machine has enough CPUs.

Memory configuration

By default, the KLA comes preconfigured with the following memory default values (analyzer.properties):

```
# Starting size for heap memory (128m = 128 Megabytes)
memory.start=128m
# Maximum size for heap memory (1024m = 1 Gigabyte)
memory.max=1024m
# Stack memory, per thread (1024k = 1 Megabyte)
stack.size=2048k
```

If your local analysis ends with an Out of Memory (OOM) error, you need to increase the max memory allocated to the JVM (by default, 1GB).

The troubleshooting links below can help you identify OOM errors:

- [Java Returned 1](#)
- [Out of Memory](#)

You can configure Kiuwan to increase memory limits either for the whole installation or per application.

- If you are using Kiuwan GUI, see [Kiuwan Local Analyzer GUI - Graphical User Interface Deprecated#GraphicalUserInterface-AnalysisConfigurationtab](#)
- If you are using CLI, see [Kiuwan Local Analyzer CLI - Command Line Interface Deprecated#CommandLineInterface-Mostcommonlyconfiguredparameters](#)



Attention

Depending on your available physical memory, OS and JVM version, if you increase the max memory the JVM might not start (please see <https://www.kiuwan.com/docs/display/K5/Not+enough+Memory>)

In these cases, stopping unneeded processes (or restarting the machine) can free unneeded allocated memory. Nevertheless, sometimes this does not free memory so you need to test with lower memory values.

If you notice the process is performing a high activity of JVM garbage collection, even without an OOM, this situation may indicate your analysis needs more memory and the performance is suffering due to GC activity. In this case, try to increase the max memory, most probably the analysis performance will be faster.



Do not increase the memory indefinitely.

If your analysis needs more than 2GB to finish, it might be a clue of an existing memory leak or some other issue.

If this happens, contact Kiuwan Technical Support and report this situation.

Timeout configuration

By default, every step of a local analysis is configured to a default max execution time (60 minutes) (*analyzer.properties*):

```
# Timeout to use for max execution time of each analysis step (in msec)
timeout=3600000
```

The default value is often enough for most analyses, but depending on several circumstances (code size, memory, ruleset, etc) could not be enough and a timeout error occurs ([Timeout Killed the Subprocess](#)).

If this happens, increase the default value.