CA APM Plugin

Plugin Information
View CA-APM on the plugin site for more information.

Allows users to collect CA APM performance data and publish it in Jenkins and allows the users to publish build number, status etc into CA-APM APM Team Center(ATC)

- CA APM Plugin 2.x and later
  - Features
  - Requirements
    - Jenkins
  - CA APM Jenkins plug-in Packaging
  - Setup
  - Install the Jenkins plug-in
  - Using the Jenkins Pipeline Syntax to Run the CA APM Jenkins plug-in
    - Blazemeter
    - Jmeter
  - Plugin Configuration
  - Default Comparison Strategies and Output Handler Available with the plug-in
    - List of Available Comparison-Strategies
    - Mean Latency Comparison Strategy
    - GC Heap Comparison Strategy
    - CPU Utilization Comparison Strategy
    - Static Threshold Strategy
    - List of Available Output-Handlers
      - Plain Text Email Report Output Handler
      - JSON File Output Handler
      - Chart Output Handler
      - Comparison Strategy to Output-Handler Mapping
  - Jenkins Attributes in APM Team Center
  - Extending the plugin
  - Troubleshoot the plug-in
    - Comparator Plugin Execution Completed with failures
    - Multiple Error Messages for similar Symptom
      - Message printed on Jenkins Console Output
    - Comparison Strategy handler for <strategy_name> is not defined
    - No Agent Specifier(s) defined for <strategy_name>
    - No metric specifier defined for <strategy_name>
    - Warning: No output handler(s) mapped to <strategy_name>
    - No output-handler(s) defined in the configuration, hence exiting Exiting means will stop running the plugin further
    - class is not present in the classpath
    - Error in executing comparison strategy <strategy_name>
    - Error in executing Output strategy <strategy_name>
  - Version history
    - Version 2.0-beta-0 (January 25, 2019)

> For ca-apm 1.x please refer to CA APM Plugin 1.x

CA APM Plugin 2.x and later

> Work in progress..

Jenkins is a tool to automate continuous delivery pipelines. A most common use of Jenkins is to run tests against the new builds of an application. The CA APM Jenkins plug-in extension automates comparing the performance of builds. The plug-in provides abstraction to run the plug-in after any LoadGenerator, define multiple comparison strategies and define multiple output handlers.

- **LoadGenerator** :You can use Jmeter and CA Balzemeter as the load generators
- **Comparison Strategies:** CA Jenkins plugin supports the following comparison strategies:
  - **Latency(ART):** Compares the average value of Average Response Time metric of a transaction with its corresponding value of benchmark build.
  - **GCHeap:** Compares the average value of Bytes in Use metric of the system with its corresponding value during benchmark build run time frame.
  - **CPU Utilization:** Compares the average value of CPU Utilization percentage metric value with its corresponding value during benchmark build run time frame.

    - **Static Threshold:** Compares the value of the metric with the corresponding threshold value defined in the configuration.
- **Output Handlers:** Email HTML Report, JSON, and Chart Output

More Information:

- CA APM Plugin 1.x
- Configure LoadGenerator When Using JMeter
- Configure the CA APM Jenkins Plugin Properties
- Extending the CA APM Jenkins Plugin

Features

- Ability to fetch multiple metrics using regex as KPI for tracking
- Ability to configure multiple fail conditions
    - Metric A less than or greater than a constant value OR
    - Metric A in comparison to Metric B
- Ability to send notification via Email (if smtp server is configured)
- Ability to view KPI's per build or cross build dashboard in Jenkins
- Ability to decorate APM Team Center 10.1 nodes using custom attributes like build status, number and date
- Ability to just notify via email: If "justEmail" is checked the build will NOT be marked as fail even if the conditions are met but only a notification will be sent
- Ability Jump to Webview in context (for both metric and time range) from build or x-build dashboard

Requirements

Jenkins

Jenkins version 2.60.1 or newer

CA APM Jenkins plug-in Packaging

The CA APM Jenkins plug-in package contains the following folder and files:

- Properties
    - `performance-comparator.properties`
- Repo
    - `ca-apm-jenkins.hpi (HPI file)`
    - `ca-apm-api.jar`
    - `ca-apm-core.jar`
    - `ca-apm-default-strategy.jar`
- javadoc directory - Contains java documentation about comparison-strategy, output-handler, and available helper utilities

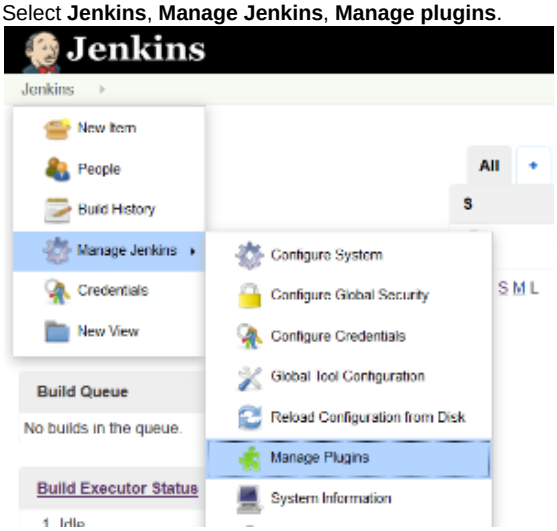
Setup

Install the Jenkins plug-in

This section describes the steps to install the plug-in. You can install the Jenkins plug-in from the Jenkin's Manage Plug-ins page.

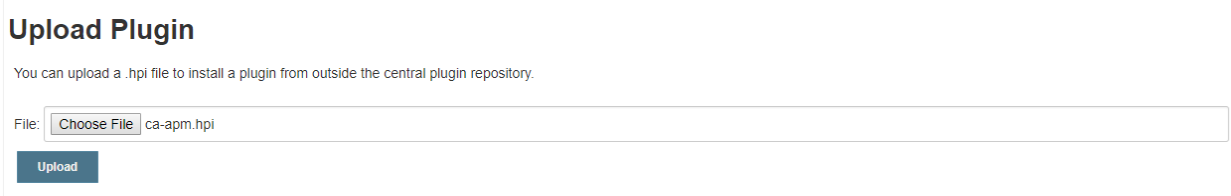
Follow these steps:

1. Select **Jenkins**, **Manage Jenkins**, **Manage plugins**.



   The Manage Plug-in page opens.
2. Select the **Advanced** tab.

   The HTTP Proxy Configuration page opens.
3. Browse and upload the performance-comparator-jenkins.hpi file available in the Repo directory.



   Wait for the installation to complete.

   > **Note:** Ensure that no jobs are running

4. Restart the Jenkins when installation is complete.

Using the Jenkins Pipeline Syntax to Run the CA APM Jenkins plug-in

1. ## Blazemeter

   To use BlazeMeter in pipeline syntax you need to have BlazeMeter Plugin

```
pipeline {
 agent any
 stages {
```

```
    stage('Blazemeter'){
        steps{
            blazeMeterTest credentialsId:'<credentials_id>',
            serverUrl:'https://blazemeter.ca.com',
            testId:'<test_id>',
            notes:'',
            sessionProperties:'',
            jtlPath:'',
            junitPath:'',
            getJtl:false,
            getJunit:false
        }
    }
    stage('CAAPMPerformanceComparator'){
        steps{
            caapmplugin "${env.WORKSPACE}/properties/performance-
comparator.properties"
        }
    }
 }
}
```

2. **Jmeter**

Sample pipeline scripts with jmeter. Please refer the Configure LoadGenerator When Using JMeter

```
pipeline {
   agent any
   stages {
        stage('JMeter'){
                steps{
                        bat "${JMETER_HOME}/bin/jmeter.bat -n -t
${JMX_FILE_PATH}/<jmx_file_name>.jmx -l ${env.WORKSPACE}
/${BUILD_NUMBER}/jmeterOutput.csv"

        }
    }
    stage('CAAPMPerformanceComparator'){
        steps{
            caapmplugin "${env.WORKSPACE}/properties/performance-
comparator.properties"
        }
    }
  }
}
```

Follow these steps:

1. Add the predefined stage in your existing pipeline job.
2. Run the Jenkins job.

Important: The Jenkins plug-in and the properties files must be in the same machine.

Plugin Configuration

Configure properties in performance-comparator.properties file.

Download the performance-comparator.properties file from http://repo.jenkins-ci.org/releases/org/jenkins-ci/plugins/ca-apm/ca-apm-dist/<version>/ca-apm-dist-<version>.zip

Please refer the Configure the CA APM Jenkins Plugin Properties

Default Comparison Strategies and Output Handler Available with the plug-in

The plug-in has the following built-in Comparison Strategies and Output Handlers.

List of Available Comparison-Strategies

By default the following strategies are executed:

- MeanLatencyStrategy (ART)
- gcheapstrategy
- cpuutilizationstrategy
- staticThresholdStrategy

If you do not want to execute the default strategy, remove the corresponding strategy name from `comparisonstrategies.list` property value list.

Note:

- Do not change the value of the comparator property for these strategies.
- Do not change the metric name from the metric specifier regular expression provided.

For example, for meanLatencyStrategy, it can be any RegEx like *Frontend.*Health: Average Response Time \\(ms\\). Do not remove Health: Average Response Time \\(ms\\), else it looses its logical meaning.

By default, the following comparison-strategies are available with the plug-in:

Mean Latency Comparison Strategy

The strategy compares the average of Average Response Time metric between current build and benchmark build with threshold of five percentage. You can set a custom value for the `meanLatencyStrategy.threshold` property. The following code shows the sample configuration:

```
strategy.comparisonstrategy.name=meanLatencyStrategy
meanLatencyStrategy.threshold=5
meanLatencyStrategy.agentspecifier=.*
meanLatencyStrategy.comparator=com.ca.apm.jenkins.
performancecomparatorplugin.comparisonstrategy.
MeanLatencyComparisonStrategy
meanLatencyStrategy.metricspecifier=.*Business Segment.*Health:Average
Response Time \\(ms\\)
meanLatencyStrategy.outputhandlers=plaintextemail,jsonfilestore,
chartoutputhtml
```

GC Heap Comparison Strategy

The strategy compares the average value of GC Heap: Bytes in Use metrics with the threshold value of two percent. You can set a custom value for the `gcheapstrategy.threshold` property. The following code shows the sample configuration:

```
strategy.comparisonstrategy.name=gcheapstrategy
gcheapstrategy.comparator=com.ca.apm.jenkins.
performancecomparatorplugin.comparisonstrategy.GCHeapComparisonStrategy
gcheapstrategy.threshold=2
gcheapstrategy.agentspecifier=.*
gcheapstrategy.metricspecifier=.*GC Heap:Bytes In Use
```

```
gcheapstrategy.outputhandlers=plaintextemail, jsonfilestore,
chartoutputhtml
```

CPU Utilization Comparison Strategy

The strategy compares the average value of CPU utilization, with the threshold value of two percent. You can set a custom value for the `cpu utilizationstrategy.threshold` property. The following code shows the sample configuration:

```
strategy.comparisonstrategy.name=cpuutilizationstrategy
cpuutilizationstrategy.comparator=com.ca.apm.jenkins.
performancecomparatorplugin.comparisonstrategy.
CPUUtilizationComparisonStrategy
cpuutilizationstrategy.threshold=2
cpuutilizationstrategy.agentspecifier=.*
cpuutilizationstrategy.metricspecifier=.*CPU.*Processor 0:Utilization %
\\(aggregate\\)
cpuutilizationstrategy.outputhandlers=plaintextemail,jsonfilestore,
chartoutputhtml
```

Static Threshold Strategy

The strategy compares your Concurrent Invocation metric of transactions of the current build with the given threshold value (2).

```
strategy.comparisonstrategy.name=staticthresholdstrategy
staticThresholdStrategy.threshold=2
staticThresholdStrategy.agentspecifier=.*
staticThresholdStrategy.comparator=com.ca.apm.jenkins.
performancecomparatorplugin.comparisonstrategy.
staticthresholdcomparisonstrategy
staticThresholdStrategy.metricspecifier=.*Business Segment.*Health:
Concurrent Invocations
staticThresholdStrategy.outputhandlers=plaintextemail,jsonfilestore,
chartoutputhtml
```

List of Available Output-Handlers

The plug-in provides the following default output handlers that are mapped to all the following default comparison-strategies in the outputhandlers property. You can remove any or all of the output handlers. The output-handlers have the following syntax:

Plain Text Email Report Output Handler

The handler generates a plain text email comparison report. The report contains the metrics for all the strategies that are mapped that are in the properties file.

```
plaintextemail.outputhandler=com.ca.apm.jenkins.
performancecomparatorplugin.outputhandler.plaintextemailoutputhandler
```

JSON File Output Handler

The handler generates a JSON report of the average value of the metrics comparison between current build and benchmark build as a percentage.

```
change jsonfilestore.outputhandler=com.ca.apm.jenkins.
performancecomparatorplugin.outputhandler.JSONFileStoreoutputhandler
```

Chart Output Handler

The handler generates a Chart Output of comparison of performance metrics between benchmark build and current build.

```
chartoutputhtml.outputhandler=com.ca.apm.jenkins.
performancecomparatorplugin.outputhandler.chartoutputhandler
```

The output is stored in `<current_build>/workspace/chartOutput` folder. The output html files are in the `chartOutput/output` folder. The folder contains one file for each comparison-strategy. For Example, If one comparison strategy has six metric paths that are identified, then the output file contains six charts.

Comparison Strategy to Output-Handler Mapping

The mapping lets you receive the output from only those `comparison-strategies` which are mapped to output-handler using the `outputHandlers` property. The communication between a comparison-strategy and output-handler happens using the Data Structure T within `StrategyResult Entity`.Map the comparison-strategy and output-handler using the same data format. If the mapping is incorrect, it is not processed by Output-Handler and plug-in raises a warning.

The default implementation of T is DefaultStrategyResult provided with the plug-in. All the default strategies and output-handlers communicate using the `StrategyResult<DefaultStrategyResult>` Entity.

Jenkins Attributes in APM Team Center

With the help of Jenkins plug-in, you can view the following Jenkins attributes as Custom attributes to each application vertex:

- currentBuildNumber
- benchmarkBuildNumber
- buildStatus
- loadGeneratorStartTime
- loadGeneratorEndTime
- loadGeneratorName

Extending the plugin

Extending the plug-in lets you add new sub commands to the existing plug-in to perform new tasks. Please refer the Extending the CA APM Jenkins Plugin

Troubleshoot the plug-in

This section discusses the error message that you see on your screen, symptom, and solution to resolve the error.

Comparator Plugin Execution Completed with failures

Symptom:

The jenkins job fails and detailed specific error is printed on the Jenkins build console.

Solution:

Review the comparison-runner.log file that is generated in `<current_build>/workspace` directory.

Multiple Error Messages for similar Symptom

- APM authentication configuration file does not exist
- LoadGenerator Configuration file does not exist

- Strategies Configuration file does not exist, file path is <file_path>
- System Configuration file does not exist, file path is <file_path>

Message printed on Jenkins Console Output

Input Properties file defined in parameters does not exist, please check

Symptom:

Jenkins plug-in run fails because the specific configuration file does not exist.

Solution:

Place the file in the specified location as printed in the error message and run the plug-in.

Comparison Strategy handler for <strategy_name> is not defined

Symptom:

There is no value against the property omparator which is the comparison strategy handler.

Solution:

Provide the complete qualified class name of the comparison strategy handler.

No Agent Specifier(s) defined for <strategy_name>

Symptom:

Values are not specified against agentspecifier for a given comparison strategy.

Solution:

Provide at least one agent-specifier regular expression, at least .*

No metric specifier defined for <strategy_name>

Symptom:

Value is not specified against metricspecifier for a given comparison strategy.

Solution:

Provide a qualified correct regular expression for metric specifier using the following syntax:

```
Business Segment\|thieves\|[^|]+\|Health:Average Response Time \(ms\)
```

Warning: No output handler(s) mapped to <strategy_name>

Symptom:

The given comparison strategy is unmapped to any output-handler, so its output is not used anywhere.

Solution:

Either comment this comparison-strategy or assign a relevant output-handler to this comparison-strategy.

No output-handler(s) defined in the configuration, hence exiting Exiting means will stop running the plugin further

Symptom:

No output handlers are mentioned against the outputhandlers list.

Solution:

Provide at least one output-handler, which is mapped to at least one comparison-strategy.

class is not present in the classpath

Symptom:

The specified class is not present inside a jar in the extensions directory that is defined in the `performance-comparator.properties` file.

Solution:

Check if the specified class is present in the configured extensions directory.

Error in executing comparison strategy <strategy_name>

Symptom:

Detailed reason is present in comparison-runner.log file present in workspace current build directory.

Solution:

Check for detailed error and act appropriately.

Error in executing Output strategy <strategy_name>

Symptom:

Detailed reason is present in comparison-runner.log file present in the `<current_build>/workspace` directory.

Solution:

Check for detailed error and act appropriately.

Version history

Version 2.0-beta-0 (January 25, 2019)

- Initial release for 2.x