Kiuwan Life Cycle Doc

This section will introduce you to the Kiuwan Life Cycle module.

Contents:

To see an explanation of the dashboard, please go to Deliveries Management

Introduction

Kiuwan provides support for the **Application Life Cycle**, allowing you to run an analysis based on the application's **Baselines** and **Deliveries**.

> **Baseline**
>
> An application's **Baseline** is a specific version of an application that is relevant enough to be considered as a reference to track further changes to it.

This version might be a production or a development version of the application, depending on its life cycle stage:

- If the application is running in production, you might consider the production version as the baseline, so you can track the differences of any further changes.
- If the application is under development, you might consider some important development milestones (e.g. the Minimum Viable Product release in Agile methodology) as your baseline.

Whatever the reason to consider a concrete version as the baseline, that version will be considered as the reference to track (and compare) any new distribution of the application.

> **Deliveries**
>
> A new distribution of the application contains changes to the baseline. The new distribution will contain a set of source files that modify the source code of the application baseline.
> In Kiuwan terminology, those changes to the application baseline are considered as a **Delivery**.
>
> Depending on the content, the delivery can be **Partial** (only a subset of application files is delivered) or **Complete** (the complete application source code is delivered).

Kiuwan's functionality on **Baseline** and **Deliveries** allows you to fully manage the complete application's life cycle:

- Establish a baseline version of the application, with the known state in terms of quality indicators, defects and associated action plan
- Track modifications to the application baseline and relate those modifications to your Life Cycle model (Feature Requests, Branches, etc)
- Trace the progress of a new version against a defined action plan
- Compare those new versions of the application against the baseline
- Define "validation" conditions (an Audit in Kiuwan terms) and let Kiuwan automatically check if those conditions are met by the delivery, providing a full report of validation status, reasons and what to do in case the audit fails.
- Evolve the baseline, either by creating a new baseline or by promoting deliveries
  - In case of complete delivery, update the whole baseline with the delivery results
  - In case of partial delivery, update incrementally the baseline with the delivery results

Some common use cases

Kiuwan Life Cycle is commonly used tightly-coupled with existing Continuous Integration, Deployment and Development systems.

Although variety can be infinite, there are two typical scenarios of using Kiuwan Local Analyzer together with existing systems.

These two scenarios are based on subjacent technology :

- Three-Tier applications (J2EE, .NET, client-server, etc)
- Mainframe applications (Cobol)

Three-Tier applications (J2EE, .NET, client-server, etc)

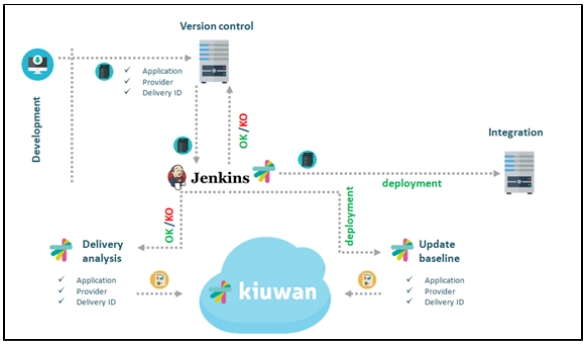
In the case of Three-Tier applications, with a **continuous integration** development process in place, where developers are continuously working on code and generating builds, Kiuwan's flexibility is ideal.

First, developers can analyze their code during development, while coding in Eclipse (for example) giving them a chance to fix the flaws in their code before uploading it to the CI tool (Jenkins, TFS or Bamboo for example, in this case, we'll assume that it's Jenkins).

Once the code reaches Jenkins, after having been inspected already, another analysis can be run, automatically deciding whether the code is accepted for the next step in the development process (moving it onto a deployment stage for example).

This decision can be made automatically using **Kiuwan's audits**, which are tests done to the source code, based on checkpoints, to decide whether the code passes or fails (OK or KO) the audit, based on the results of the analysis. The type of analysis run at this point depends on the development process of each company and can be either a delivery analysis or a baseline analysis. Usually, a delivery analysis is run when the code is not ready yet to become a new baseline, and Kiuwan offers different types of delivery analyses, each with different properties, to adapt to each company's particularities.

Since Kiuwan Local Analyzer is fast at performing analyses (especially when compared to other static analysis tools), it can be added as another step in the CI process, without adding the hassle of having to wait long periods of time for the results of the analysis to continue to the next step of the CI process.



Mainframe applications (COBOL)

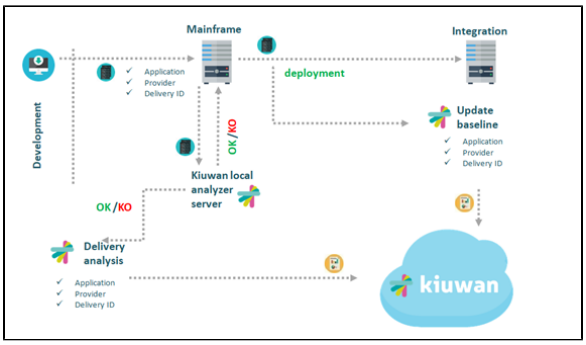
In the case of **mainframe** applications, where the deployment method is usually different, Kiuwan offers the possibility to **analyze applications incrementally**, meaning that not all of an application's source code has to be analyzed on each analysis so that only the modified files have to be analyzed whenever changes are made.

This is achieved by performing **partial delivery analyses** when changes are made, and only occasionally performing complete delivery or baseline analyses, **reducing analysis times** considerably, and not needing access to all of the code when an analysis is to be made, but only the modified files.

Delivery analyses can also be **promoted** to baseline level, meaning that whenever we are sure that our delivery code is ready to become a baseline, we can simply indicate this to Kiuwan, and there is **no need to re-analyze** all of our code as a baseline analysis.

Therefore, the source code management tool being used does not need to provide Kiuwan Local Analyzer all of the code every time, but only the code to be analyzed.

Again, promotion between stages (pre-production, production…) can easily be automatically managed using Kiuwan's **audits**, which will return an OK or KO status for each analysis.



More on Life Cycle