Integration with Atlassian Bamboo

This guide will show you how to integrate Kiuwan with Atlassian Bamboo.

Contents:

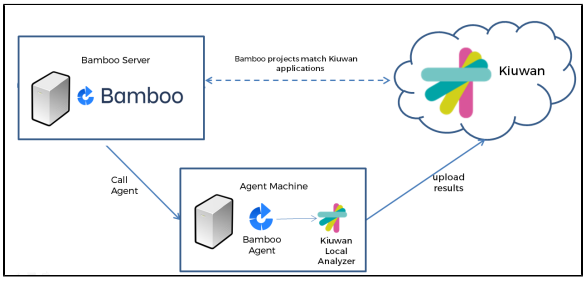
Introduction to the integration with Atlassian Bamboo

Kiuwan analyses can be integrated into the continuous integration process with Atlassian Bamboo. The analyses can run completely automated to enable continuous code security. It is even possible to automatically enforce your security policies.

In continuous integration and continuous delivery (CI/CD) environments, it is very common (and recommended) to ensure the security and quality of the software under development with an automated solution.

Kiuwan allows you to do baseline or delivery analysis as a step in the build plan defined in Bamboo.

A Kiuwan analysis will be executed by a Bamboo agent. This could be a remote agent or the default agent running on the Bamboo server.



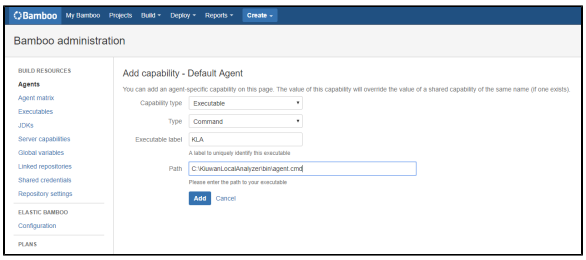
Configure the Bamboo agents

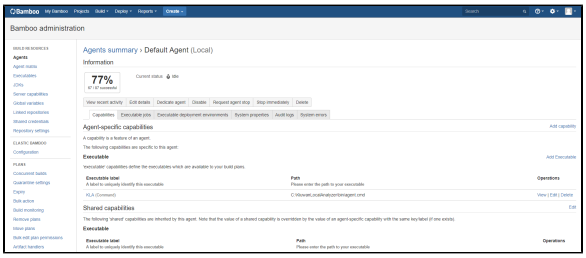
Before you start:

1. Make sure that any agent which can potentially run a Kiuwan analysis as part of a build plan has the Kiuwan Local Analyzer (KLA) installed on the same machine .
2. Configure a Kiuwan agent-specific capability in the agent.

For example, if you install your KLA in *C:\KiuwanLocalAnalyzer*, you can configure a new agent-specific capability as follows:

1. Go to the **Bamboo agents admin page** and select the agent you want to configure
2. Click **Add capability**
3. Select **Executable** from the Capability type dropdown
4. Select **Command** from the Type dropdown
5. Give the new capability a **unique label** to identify it (*KLA* for example)
6. Specify the complete absolute path to the KLA command (*C:\KiuwanLocalAnalyzer\bin\agent.cmd* for example)





Repeat this operation for all agents that will run Kiuwan analyses.

The more agents you have with the KLA capability, the more analyses you can run in parallel. This will depend on the number of applications you have under continuous development and your build/analysis strategy.

Once you have all your agents configured, you can create (or modify) build plans to run Kiuwan analyses.

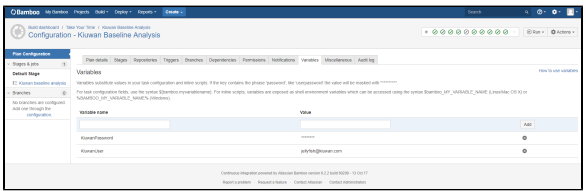
Create Bamboo builds for projects to analyze

Build plans in Bamboo are created for a specific project. Create one, or select an existing one to configure it.

Configure Kiuwan credentials

First, configure the Kiuwan credentials variables.

Go to the **Variables** tab and create two variables called **KiuwanUser** and **KiuwanPassword.** For example:

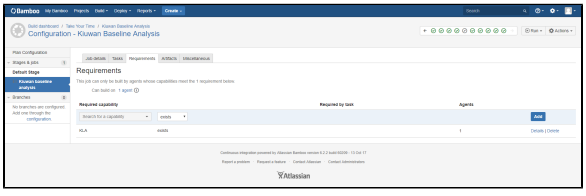


Alternatively, you could create a global variable for your credentials. In that case, the same credentials will be used across your build plans.

Create an analysis stage

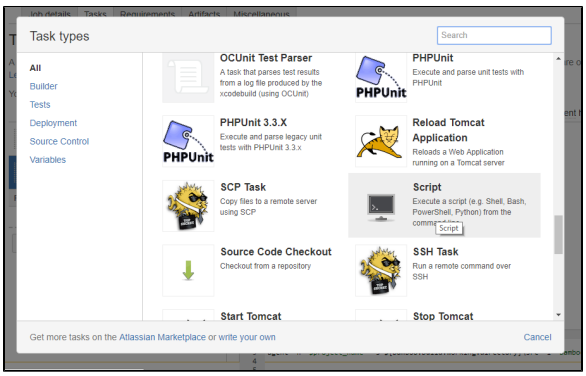
Create a stage to add a Kiuwan analysis job to it or add a job in an existing stage (as we have done here in the default stage).

In **Job definition,** search for the KLA capability and specify that it must ‘exist’. This configuration ensures that the job is sent to an agent that has the KLA installed.



Then go to the **Tasks** tab to configure the Kiuwan analysis task. Kiuwan needs the source code to analyze it, so make sure that there is a 'Source Code Checkout' task before anything else.

Finally, add a Script task to run the Kiuwan analysis. Configure more tasks (like build) in this stage or just use it only for the analysis.



1. Select the new **Script** task and add a description (in our case Kiuwan baseline analysis, because that is what it is going to do)
2. Select the **Windows PowerShell,** if your agents are Windows-based. For Linux-based agents, you have to specify a different agent capability and here select Shell.
3. Next, select **Inline** in script location (you could have a file in your repository with the script to run).
4. Finally, enter the following script code in the Script body text area:

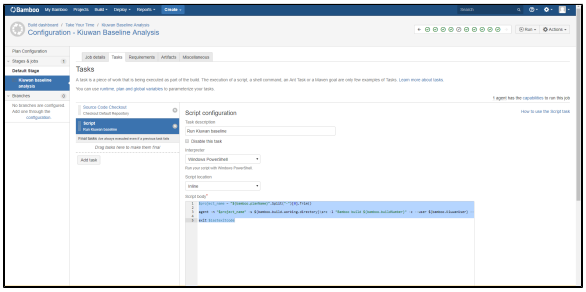
This is a PowerShell example, using some PowerShell commandlets. You can create a similar script for Linux shells mimicking the functionality.

Kiuwan baseline analysis script

```
$project_name = "${bamboo.planName}".Split("-")[0].Trim()

agent -n "$project_name" -s ${bamboo.build.working.directory}\src -l
"Bamboo build ${bamboo.buildNumber}" -c --user ${bamboo.KiuwanUser} --pass
${bamboo.KiuwanPassword} -wr

exit $lastexitcode
```



Alternative Kiuwan analysis: Delivery

Kiuwan can run two different types of analysis: baseline and delivery. In the previous example, the script runs a baseline analysis. If you want to run a delivery analysis your script will look something like the following:

Kiuwan Delivery analysis script

```
$project_name = "${bamboo.planName}".Split("-")[0].Trim()

$change_request = "New CR"

agent -n $project_name -s ${bamboo.build.working.directory}\src -l
"${bamboo.shortPlanKey}${bamboo.shortJobKey}-${bamboo.buildNumber}" -as
completeDelivery -crs resolved -cr  ${bamboo.planRepository.branch} -bn
${bamboo.planRepository.branch} --user ${bamboo.KiuwanUser} --pass
${bamboo.KiuwanPassword} -wr

exit $lastexitcode
```

When do you run a baseline or a delivery analysis? That is up to you and your development process. The difference between the examples here is that the delivery analysis will return a non-zero (10) exit code if the Kiuwan audit fails; so the stage will fail, and depending on how you configure the plan, it could fail in turn. Check the Kiuwan life cycle documentation for more information on deliveries and audits that implement your security policies for changes.

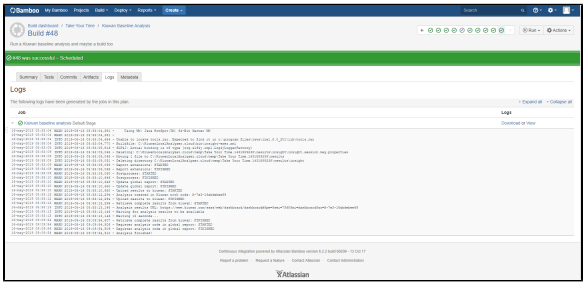
A good use case could be to run a Delivery analysis for every pull request, so the policy is applied before the merge. The merge can be blocked if the pull request doesn't pass the Kiuwan audit.

Run build plans

You can run the build plan manually or define different triggers and strategies to run them automatically.

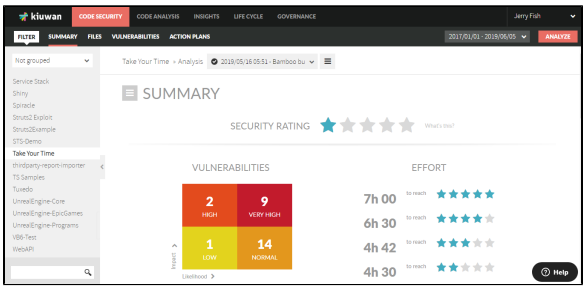
What you do will depend on your development process.

You can follow the Kiuwan task execution on-line or check the logs after execution:

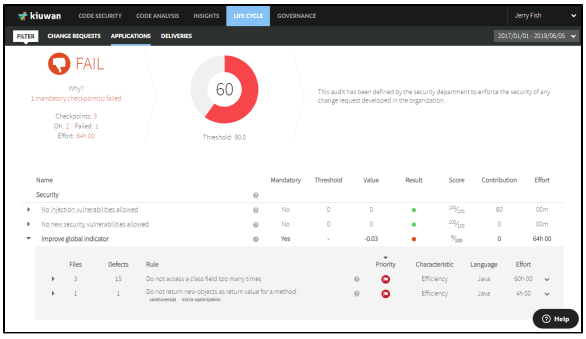


Check Kiuwan analysis results

Finally, go to your Kiuwan account and check the results. If the analysis was baseline, go to Code Security:



Or check the Kiuwan audit results for delivery analyses:


kiuwan
CODE SECURITY
CODE ANALYSIS
INSIGHTS
LIFE CYCLE
GOVERNANCE
Jerry Fish
FILTER
CHANGE REQUESTS
APPLICATIONS
DELIVERIES
2017/01/01 - 2019/06/05
FAIL
Why?
1 mandatory checkpoint(s) failed
Checkpoints: 3
OK: 2 Failed: 1
Effort: 64h 00
60
Threshold: 90.0
This audit has been defined by the security department to enforce the security of any change request developed in the organization.
Name
Mandatory
Threshold
Value
Result
Score
Contribution
Effort
Security
No injection vulnerabilities allowed
No
0
0
100/100
60
00m
No new security vulnerabilities allowed
No
0
0
100/100
0
00m
Improve global indicator
Yes
-
-0.03
0/100
0
64h 00
Files
Defects
Rule
Priority
Characteristic
Language
Effort
3
15
Do not access a class field too many times
Efficiency
Java
60h 00
1
1
Do not return new objects as return value for a method
controversial
micro-optimization
Efficiency
Java
4h 00
Help