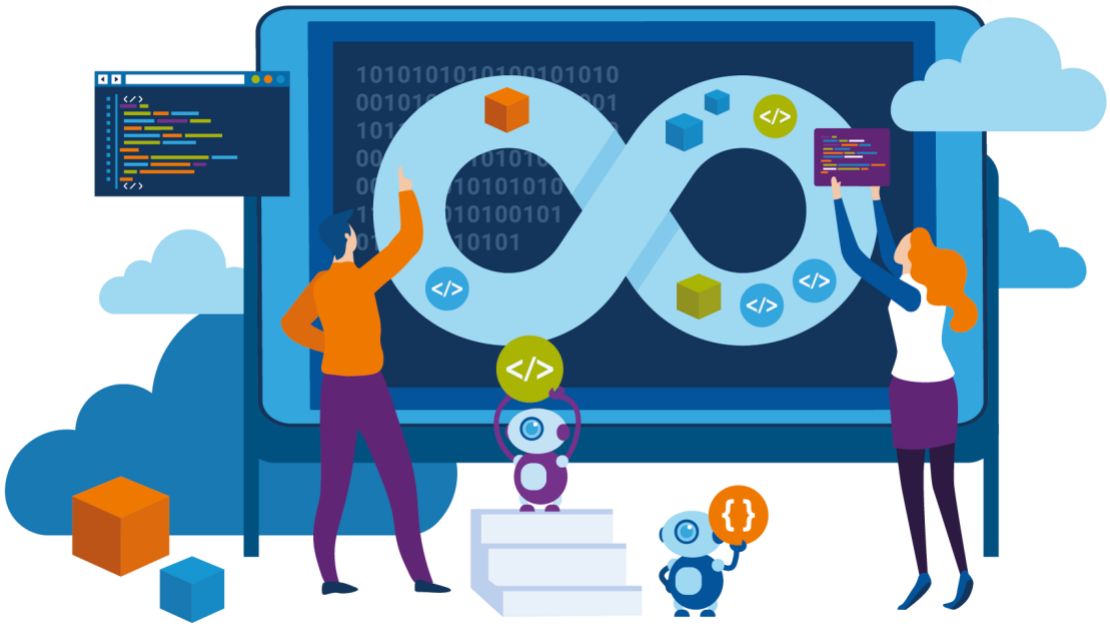
CI / CD Pipeline for UiPath Robots



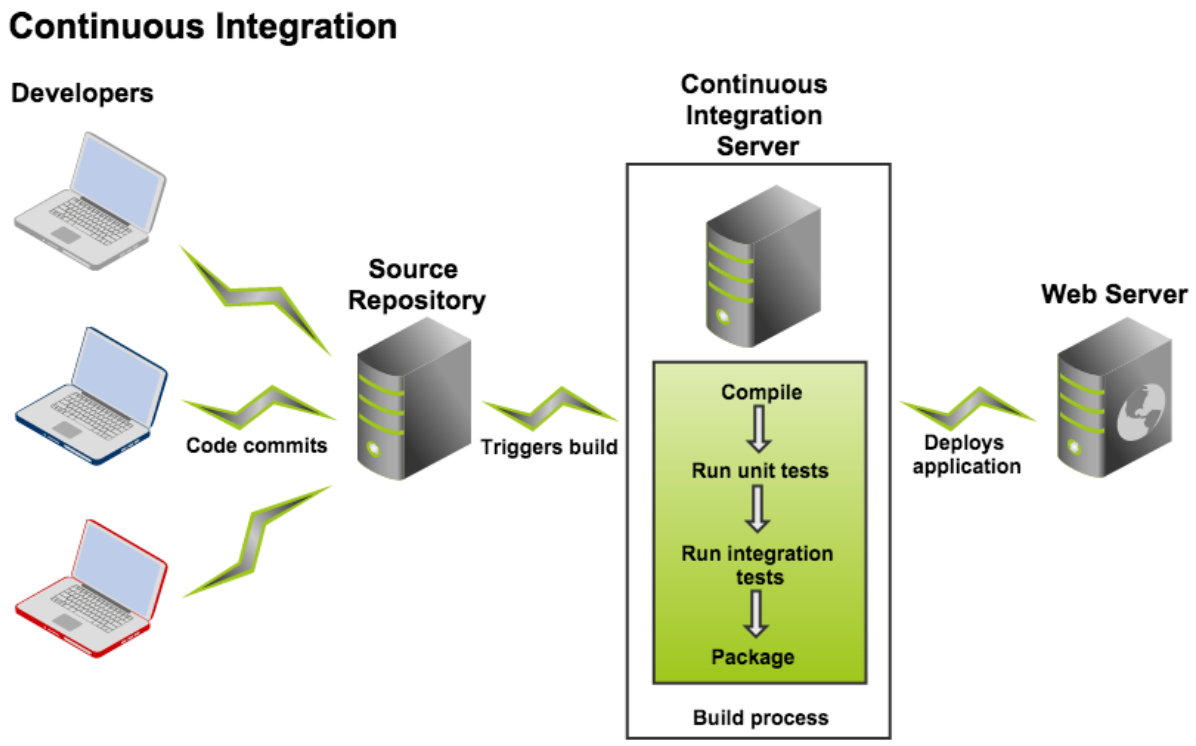
Although most of the organisations have started implementing Robotic Process Automation in their business processes, only a few are in full control of RPA robots. Maintaining high performing virtual robots goes beyond having skilled developers, they have to align to best practices and standards to benefit at the fullest.

A CI/CD (Continuous Integration, Continuous Delivery) pipeline allows you to automate robot delivery process. The pipeline will initiate the build from version control process, ensuring code merges and after completing all the steps, the package will reside in respective Orchestrator (Stage / Prod) automatically. It will help to take care of robot delivery process steps such as compiling & creating the package, uploading it to Orchestrator etc., in a seamless manner.

What is CI/CD?

Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. It is one of the best practices allowing developers to frequently merge code changes into a central repository where builds and tests then run. Automated tools are used to assert the new code's correctness before integration.

A source code version control system is the crux of the CI process. The version control system is also supplemented with other checks like automated code quality tests, syntax style review tools, and more.



Without CI, developers must manually coordinate and communicate when they are contributing code to the end product. This coordination extends beyond the development teams to operations and the rest of the organisation. Product teams must coordinate when to sequentially launch features and fixes and which team members will be responsible.

Continuous Delivery (CD) is the natural extension of Continuous Integration: an approach in which teams ensure that every change to the system is releasable, and that we can release any version at the push of a button.

Why RPA needs CI/CD?

Although many organisations have started implementing a Digital Workforce into their business operations, only a few are fully in control of their RPA bots. Maintaining high performing virtual workers goes beyond having skilled developers, they've to align to quality standards and practices to benefit from RPA to the fullest.

A CI/CD (Continuous Integration, Continuous Delivery) pipeline allows you to automate your robot delivery process. The pipeline will initiate the build from your version control system and after completing every step the package will reside within your Orchestrator, ready to be executed. It may sound like overhead, but it certainly is not – it will help you to always cover the steps required to successfully deliver a robot. These steps could be as simple as compiling your package and uploading it to the Orchestrator – but can also include code reviews, approvals and other steps you've defined building your Center of Excellence.

Benefits of CI/CD

Continuous integration is an essential aspect of Software Development Lifecycle. Yet CI benefits are not limited to the engineering team but greatly benefit the overall organisation. CI enables better transparency and insight into the process of software development and delivery. These benefits enable the rest of the organisation to better plan and execute go to market strategies. The following are some of the overall organisational benefits of CI. Below are few key benefits of CI/CD

- Always have the latest code available in your version control system

Often, we see that clients are struggling with their version control. With a pipeline, the only way to upload a package to the orchestrator is through committing your final version to the version control system.

- Remain compliant and in control of your deployments

You can implement controls such as an approval flow or code reviews

- Practice what you preach – reduce time spent on cumbersome tasks such as a deployment

Deployments can take up hours of costly development time, eliminate cumbersome tasks from your CoE

What is Bamboo?

Atlassian Bamboo is a continuous integration (CI) and deployment server. Bamboo assists software development teams by providing automated building and testing of software source-code status, updates on successful and failed builds and reporting tools for statistical analysis.

- Bamboo is the central management server which schedules and coordinates all work.
- Bamboo itself has interfaces and plugins for lots of types of work.
- Bamboo first gets your source from a source repository (lots of plugins here for a variety of systems).
- Then Bamboo starts the build - that can be done by calling something like MSBuild to build your Visual Studio solution, or Maven to call your compiler and linker - whatever you use.
- Once your solution or project is built, you have "artefacts" (build results, for example, an executable app, config files, etc.).
- You can do additional things with the build artefacts:
    - zip them up into a ZIP file and copy them somewhere.
    - run an install builder on them and create an MSI.
    - install them on a test server to make sure everything installs just fine.
- Bamboo provides a web front-end for configuration and for reporting the status of builds.

Understanding the Bamboo CI Server

Bamboo is a continuous integration (CI) server that can be used to automate the release management for a software application, creating a continuous delivery pipeline.

CI is a software development methodology in which a build, unit tests and integration tests are performed, or triggered, whenever code is committed to the repository, to ensure that new changes integrate well into the existing code base. Integration builds provide early 'fail fast' feedback on the quality of new changes.

Release management describes the steps that are typically performed to release a software application, including building and functional testing, tagging releases, assigning versions, and deploying and activating the new version in production

What problem does Bamboo solve?

If you are a solo developer, then using Bamboo gives you:

- an automated, and therefore reliable, build and test process, leaving you free to code more.
- a way to manage builds that have different requirements or targets.
- automatic deployment to a server, such as the App Store or Google Play.

If you work in a team, then as well as the above advantages, using Bamboo also means that:

- your build and test process is not dependent on a specific local environment.
- builds and integration tests are triggered automatically as soon as a developer commits code (continuous integration).

If you work on a large, complex application, then, in addition to all the above advantages, using Bamboo means that:

- you can optimise build performance through parallelism.
- you can leverage elastic resources.
- you can deploy continuously and implement release management

Bamboo workflows

Bamboo uses the concept of a 'plan' with 'jobs' and 'tasks' to configure and order the actions in the workflow.

Project

- Has none, one, or more, plans.
- Provides reporting (using the wallboard, for example) across all plans in the project.
- Provides links to other applications.
- Allows setting up permissions for all the plans it contains

Plan

- Has a single stage, by default, but can be used to group jobs into multiple stages.
- Processes a series of one or more stages that are run **sequentially** using the **same** repository.
- Specifies the default repository.
- Specifies how the build is triggered, and the triggering dependencies between the plan and other plans in the project.
- Specifies notifications of build results.
- Specifies who has permission to view and configure the plan and its jobs.
- Provides for the definition of plan variables.

Stage

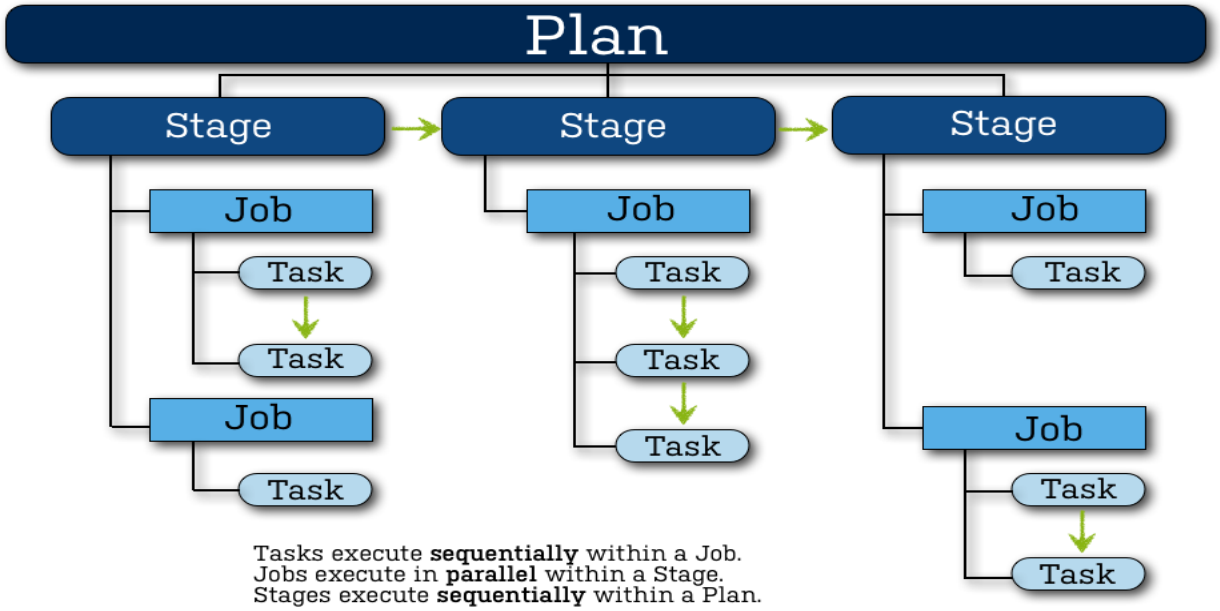
- By default has a single job but can be used to group multiple jobs.
- Processes its jobs in **parallel**, on **multiple** agents (where available).
- Must successfully complete all its jobs before the next stage in the plan can be processed.
- May produce artefacts that can be made available for use by a subsequent stage.

Job

- Processes a series of one or more tasks that are run **sequentially** on the **same** agent.
- Controls the order in which tasks are performed.
- Collects the requirements of individual tasks in the job, so that these requirements can be matched with agent capabilities.
- Defines the artefacts that the build will produce.
- Can only use artefacts produced in a previous stage.
- Specifies any labels with which the build result or build artefacts will be tagged.

Task

- Is a small discrete unit of work, such as source code checkout, executing a Maven goal, running a script, or parsing test results.
- Is run sequentially within a job on a Bamboo working directory.



Bamboo Configuration Options

Bamboo provides native support for a number of different test, build, and deployment tasks. If the capability your looking for isn't included in Bamboo out-of-the-box then you have a couple of different options:

- Third-party plugin:

  Atlassian Marketplace offers a variety of custom build plugins that can used to add additional features to your instance.

- Custom Script:

  Bamboo has the ability to run custom scripts using its script task. Anything that can be run via the command line can be triggered by Bamboo.

- Build a custom plugin:

  Bamboo provides an extensive REST API that can be used to build custom plugins and add-ons.

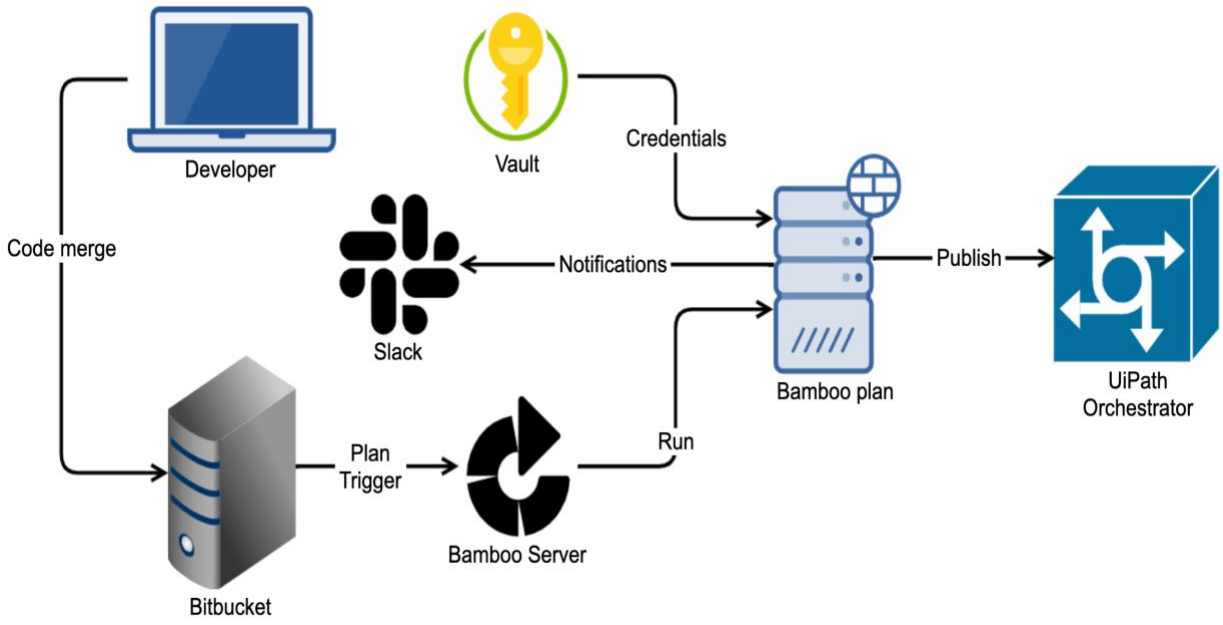
Prerequisites for Bamboo

Bamboo schedules and coordinates the work involved in building and testing your application. Therefore, to use Bamboo, you will need to already have the following set up:

- a code repository that contains the complete source code for the project.
- build scripts
- test suites

It is generally assumed that the person who commits a change to the code is responsible for fixing any resulting build errors immediately.

Implementation of CD with UiPath using Bamboo

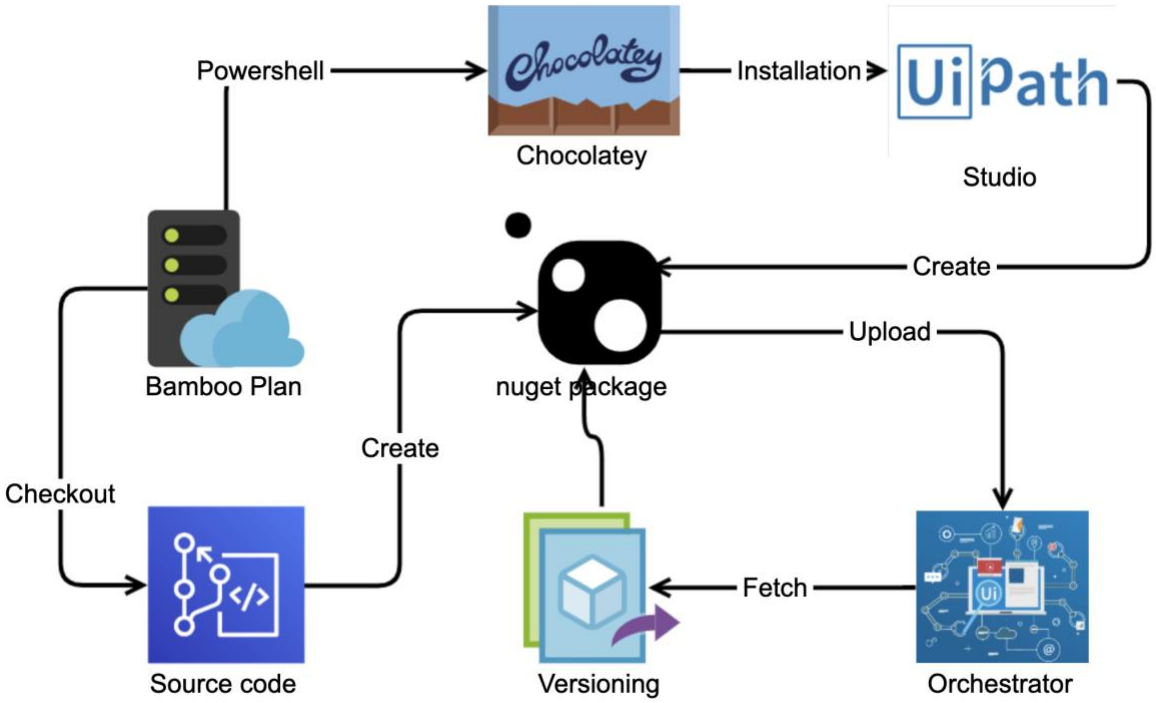


When an UiPath developer pushed the code to Git / Bitbucket server, Bamboo plan will be triggered as per configuration and start the packaging & publishing process to Orchestrator.

Vault secrets can be used to store UiPath Orchestrator credentials which can be passed to Bamboo plans.

Bamboo agents runs PowerShell scripts built in Bamboo plan and creates .nuget package, publish the same to UiPath orchestrator.

Bamboo Plan - Deep dive



Trigger: Commit to branch in Bitbucket repository

Chocolatey: Installs UiPath Studio in Bamboo agent using PowerShell scripts

UiPath Studio: Creates .nuget package and keep it ready for deployment

Orchestrator: Fetches existing version and upload created .nuget package

Bamboo plan: Orchestrates all the above steps and deploy the package seamlessly into orchestrator

Slack: Receives notification post deployment

References:

https://confluence.atlassian.com/alldoc/bamboo-documentation-directory-23855144.html

https://docs.uipath.com/studio/docs/mass-update-command-line-parameters

https://www.uipath.com/hubfs/Documentation/OrchestratorAPIGuide_2016.2/UiPathOrchestratorAPIGuide_2016.2.html