Migrating from Bamboo with GitHub Actions Importer

In this article

Learn how to use GitHub Actions Importer to automate the migration of your Bamboo pipelines to GitHub Actions.

Legal notice

About migrating from Bamboo with GitHub Actions Importer

The instructions below will guide you through configuring your environment to use GitHub Actions Importer to migrate Bamboo pipelines to GitHub Actions.

Prerequisites

- A Bamboo account or organization with projects and pipelines that you want to convert to GitHub Actions workflows.
- Bamboo version of 7.1.1 or greater.
- Access to create a Bamboo personal access token for your account or organization.
- An environment where you can run Linux-based containers, and can install the necessary tools.
  - Docker is installed and running.
  - GitHub CLI is installed.

> **Note**: The GitHub Actions Importer container and CLI do not need to be installed on the same server as your CI platform.

Limitations

There are some limitations when migrating from Bamboo to GitHub Actions with GitHub Actions Importer:

- GitHub Actions Importer relies on the YAML specification generated by the Bamboo Server to perform migrations. When Bamboo does not support exporting something to YAML, the missing information is not migrated.
- Trigger conditions are unsupported. When GitHub Actions Importer encounters a trigger with a condition, the condition is surfaced as a comment and the trigger is transformed without it.
- Bamboo Plans with customized settings for storing artifacts are not transformed. Instead, artifacts are stored and retrieved using the `upload-artifact` and `download-artifact` actions.
- Disabled plans must be disabled manually in the GitHub UI. For more information, see "Disabling and enabling a workflow."
- Disabled jobs are transformed with a `if: false` condition which prevents it from running. You must remove this to re-enable the job.
- Disabled tasks are not transformed because they are not included in the exported plan when using the Bamboo API.
- Bamboo provides options to clean up build workspaces after a build is complete. These are not transformed because it is assumed GitHub-hosted runners or ephemeral self-hosted runners will automatically handle this.
- The hanging build detection options are not transformed because there is no equivalent in GitHub Actions. The closest option is `timeout-minutes` on a job, which can be used to set the maximum number of minutes to let a job run. For more information, see "Workflow syntax for GitHub Actions."
- Pattern match labeling is not transformed because there is no equivalent in GitHub Actions.
- All artifacts are transformed into an `actions/upload-artifact`, regardless of whether they are `shared` or not, so they can be downloaded from any job in the workflow.
- Permissions are not transformed because there is no suitable equivalent in GitHub Actions.
- If the Bamboo version is between 7.1.1 and 8.1.1, project and plan variables will not be migrated.

Manual tasks

Certain Bamboo constructs must be migrated manually. These include:

- Masked variables
- Artifact expiry settings

Installing the GitHub Actions Importer CLI extension

1. Install the GitHub Actions Importer CLI extension:

```
Bash
gh extension install github/gh-actions-importer
```

2. Verify that the extension is installed:

```
$ gh actions-importer -h
Options:
  -?, -h, --help  Show help and usage information

Commands:
  update     Update to the latest version of GitHub Actions Importer.
```

```
  version    Display the version of GitHub Actions Importer.
  configure  Start an interactive prompt to configure credentials used to
authenticate with your CI server(s).
  audit      Plan your CI/CD migration by analyzing your current CI/CD
footprint.
  forecast   Forecast GitHub Actions usage from historical pipeline
utilization.
  dry-run    Convert a pipeline to a GitHub Actions workflow and output its
yaml file.
  migrate    Convert a pipeline to a GitHub Actions workflow and open a pull
request with the changes.
```

Configuring credentials

The `configure` CLI command is used to set required credentials and options for GitHub Actions Importer when working with Bamboo and GitHub.

1. Create a GitHub personal access token (classic). For more information, see "Managing your personal access tokens."

   Your token must have the `workflow` scope.

   After creating the token, copy it and save it in a safe location for later use.

2. Create a Bamboo personal access token. For more information, see Personal Access Tokens in the Bamboo documentation.

   Your token must have the following permissions, depending on which resources will be transformed.

Resource Type	View	View Configuration	Edit
Build Plan	✓	✓	✓
Deployment Project	✓	✓	×
Deployment Environment	✓	×	×

   After creating the token, copy it and save it in a safe location for later use.

3. In your terminal, run the GitHub Actions Importer `configure` CLI command:

   ```
   gh actions-importer configure
   ```

   The `configure` command will prompt you for the following information:

   - For "Which CI providers are you configuring?", use the arrow keys to select `Bamboo`, press `Space` to select it, then press `Enter`.
   - For "Personal access token for GitHub", enter the value of the personal access token (classic) that you created earlier, and press `Enter`.
   - For "Base url of the GitHub instance", enter the URL for your GitHub Enterprise Server instance, and press `Enter`.
   - For "Personal access token for Bamboo", enter the value for the Bamboo personal access token that you created earlier, and press `Enter`.
   - For "Base url of the Bamboo instance", enter the URL for your Bamboo Server or Bamboo Data Center instance, and press `Enter`.

   An example of the `configure` command is shown below:

```
$ gh actions-importer configure
✔ Which CI providers are you configuring?: Bamboo
Enter the following values (leave empty to omit):
✔ Personal access token for GitHub: ***************
✔ Base url of the GitHub instance: https://github.com
✔ Personal access token for Bamboo: ********************
✔ Base url of the Bamboo instance: https://bamboo.example.com
Environment variables successfully updated.
```

4. In your terminal, run the GitHub Actions Importer `update` CLI command to connect to GitHub Packages Container registry and ensure that the container image is updated to the latest version:

   ```
   gh actions-importer update
   ```

   The output of the command should be similar to below:

   ```
   Updating ghcr.io/actions-importer/cli:latest...
   ghcr.io/actions-importer/cli:latest up-to-date
   ```

Perform an audit of Bamboo

You can use the `audit` command to get a high-level view of all projects in a Bamboo organization.

The `audit` command performs the following steps:

1. Fetches all of the projects defined in a Bamboo organization.
2. Converts each pipeline to its equivalent GitHub Actions workflow.
3. Generates a report that summarizes how complete and complex of a migration is possible with GitHub Actions Importer.

Running the audit command

To perform an audit of a Bamboo instance, run the following command in your terminal:

```
gh actions-importer audit bamboo --output-dir tmp/audit
```

Inspecting the audit results

The files in the specified output directory contain the results of the audit. See the `audit_summary.md` file for a summary of the audit results.

The audit summary has the following sections.

Pipelines

The "Pipelines" section contains a high-level statistics regarding the conversion rate done by GitHub Actions Importer.

Listed below are some key terms that can appear in the "Pipelines" section:

- **Successful** pipelines had 100% of the pipeline constructs and individual items

converted automatically to their GitHub Actions equivalent.

- **Partially successful** pipelines had all of the pipeline constructs converted, however, there were some individual items that were not converted automatically to their GitHub Actions equivalent.
- **Unsupported** pipelines are definition types that are not supported by GitHub Actions Importer.
- **Failed** pipelines encountered a fatal error when being converted. This can occur for one of three reasons:
  - The pipeline was misconfigured and not valid in Bamboo.
  - GitHub Actions Importer encountered an internal error when converting it.
  - There was an unsuccessful network response that caused the pipeline to be inaccessible, which is often due to invalid credentials.

Build steps

The "Build steps" section contains an overview of individual build steps that are used across all pipelines, and how many were automatically converted by GitHub Actions Importer.

Listed below are some key terms that can appear in the "Build steps" section:

- A **known** build step is a step that was automatically converted to an equivalent action.
- An **unknown** build step is a step that was not automatically converted to an equivalent action.
- An **unsupported** build step is a step that is either:
  - Fundamentally not supported by GitHub Actions.
  - Configured in a way that is incompatible with GitHub Actions.
- An **action** is a list of the actions that were used in the converted workflows. This can be important for:
  - If you use GitHub Enterprise Server, gathering the list of actions to sync to your instance.
  - Defining an organization-level allowlist of actions that are used. This list of actions is a comprehensive list of actions that your security or compliance teams may need to review.

Manual tasks

The "Manual tasks" section contains an overview of tasks that GitHub Actions Importer is not able to complete automatically, and that you must complete manually.

Listed below are some key terms that can appear in the "Manual tasks" section:

- A **secret** is a repository or organization-level secret that is used in the converted pipelines. These secrets must be created manually in GitHub Actions for these pipelines to function properly. For more information, see "Using secrets in GitHub Actions."
- A **self-hosted runner** refers to a label of a runner that is referenced in a converted pipeline that is not a GitHub-hosted runner. You will need to manually define these runners for these pipelines to function properly.

Files

The final section of the audit report provides a manifest of all the files that were written to disk during the audit.

Each pipeline file has a variety of files included in the audit, including:

- The original pipeline as it was defined in GitHub.
- Any network responses used to convert the pipeline.
- The converted workflow file.
- Stack traces that can be used to troubleshoot a failed pipeline conversion.

Additionally, the `workflow_usage.csv` file contains a comma-separated list of all actions, secrets, and runners that are used by each successfully converted pipeline. This can be useful for determining which workflows use which actions, secrets, or runners, and can be useful for performing security reviews.

Forecasting usage

You can use the `forecast` command to forecast potential GitHub Actions usage by computing metrics from completed pipeline runs in your Bamboo instance.

Running the forecast command

To perform a forecast of potential GitHub Actions usage, run the following command in your terminal. By default, GitHub Actions Importer includes the previous seven days in the forecast report.

```
gh actions-importer forecast bamboo --output-dir tmp/forecast_reports
```

Forecasting a project

To limit the forecast to the plans and deployments environments associated with a project, you can use the `--project` option, where the value is set to a build project key.

For example:

```
gh actions-importer forecast bamboo --project PAN --output-dir
tmp/forecast_reports
```

Inspecting the forecast report

The `forecast_report.md` file in the specified output directory contains the results of the forecast.

Listed below are some key terms that can appear in the forecast report:

- The **job count** is the total number of completed jobs.
- The **pipeline count** is the number of unique pipelines used.
- **Execution time** describes the amount of time a runner spent on a job. This metric can be used to help plan for the cost of GitHub-hosted runners.
    - This metric is correlated to how much you should expect to spend in GitHub Actions. This will vary depending on the hardware used for these minutes. You can use the GitHub Actions pricing calculator to estimate the costs.
- **Queue time** metrics describe the amount of time a job spent waiting for a runner to be available to execute it.
- **Concurrent jobs** metrics describe the amount of jobs running at any given time. This metric can be used to

Perform a dry-run migration of a Bamboo pipeline

You can use the `dry-run` command to convert a Bamboo pipeline to an equivalent GitHub Actions workflow. A dry-run creates the output files in a specified directory, but does not open a pull request to migrate the pipeline.

Running a dry-run migration for a build plan

To perform a dry run of migrating your Bamboo build plan to GitHub Actions, run the following command in your terminal, replacing `:my_plan_slug` with the plan's project and plan key in the format `<projectKey>-<planKey>` (for example: `PAN-SCRIP`).

```
gh actions-importer dry-run bamboo build --plan-slug :my_plan_slug --output-dir
tmp/dry-run
```

Running a dry-run migration for a deployment project

To perform a dry run of migrating your Bamboo deployment project to GitHub Actions, run the following command in your terminal, replacing `:my_deployment_project_id` with the ID of the deployment project you are converting.

```
gh actions-importer dry-run bamboo deployment --deployment-project-id
:my_deployment_project_id --output-dir tmp/dry-run
```

You can view the logs of the dry run and the converted workflow files in the specified output directory.

If there is anything that GitHub Actions Importer was not able to convert automatically, such as unknown build steps or a partially successful pipeline, you might want to create custom transformers to further customize the conversion process. For more information, see "Extending GitHub Actions Importer with custom transformers."

Perform a production migration of a Bamboo pipeline

You can use the `migrate` command to convert a Bamboo pipeline and open a pull request with the equivalent GitHub Actions workflow.

Running the migrate command for a build plan

To migrate a Bamboo build plan to GitHub Actions, run the following command in your terminal, replacing the `target-url` value with the URL for your GitHub repository, and `:my_plan_slug` with the plan's project and plan key in the format `<projectKey>-<planKey>`.

```
gh actions-importer migrate bamboo build --plan-slug :my_plan_slug --target-url
:target_url --output-dir tmp/migrate
```

The command's output includes the URL to the pull request that adds the converted workflow to your repository. An example of a successful output is similar to the following:

```
$ gh actions-importer migrate bamboo build --plan-slug :PROJECTKEY-PLANKEY --
target-url https://github.com/octo-org/octo-repo --output-dir tmp/migrate
[2022-08-20 22:08:20] Logs: 'tmp/migrate/log/actions-importer-20220916-
014033.log'
```

```
[2022-08-20 22:08:20] Pull request: 'https://github.com/octo-org/octo-
repo/pull/1'
```

Running the migrate command for a deployment project

To migrate a Bamboo deployment project to GitHub Actions, run the following command in your terminal, replacing the `target-url` value with the URL for your GitHub repository, and `:my_deployment_project_id` with the ID of the deployment project you are converting.

```
gh actions-importer migrate bamboo deployment --deployment-project-id
:my_deployment_project_id --target-url :target_url --output-dir tmp/migrate
```

The command's output includes the URL to the pull request that adds the converted workflow to your repository. An example of a successful output is similar to the following:

```
$ gh actions-importer migrate bamboo deployment --deployment-project-id 123 --
target-url https://github.com/octo-org/octo-repo --output-dir tmp/migrate
[2023-04-20 22:08:20] Logs: 'tmp/migrate/log/actions-importer-20230420-
014033.log'
[2023-04-20 22:08:20] Pull request: 'https://github.com/octo-org/octo-
repo/pull/1'
```

Inspecting the pull request

The output from a successful run of the `migrate` command contains a link to the new pull request that adds the converted workflow to your repository.

Some important elements of the pull request include:

- In the pull request description, a section called **Manual steps**, which lists steps that you must manually complete before you can finish migrating your pipelines to GitHub Actions. For example, this section might tell you to create any secrets used in your workflows.
- The converted workflows file. Select the **Files changed** tab in the pull request to view the workflow file that will be added to your GitHub Enterprise Server repository.

When you are finished inspecting the pull request, you can merge it to add the workflow to your GitHub Enterprise Server repository.

Reference

This section contains reference information on environment variables, optional arguments, and supported syntax when using GitHub Actions Importer to migrate from Bamboo.

Using environment variables

GitHub Actions Importer uses environment variables for its authentication configuration. These variables are set when following the configuration process using the `configure` command. For more information, see the "Configure credentials for GitHub Actions Importer" section.

GitHub Actions Importer uses the following environment variables to connect to your Bamboo instance:

- `GITHUB_ACCESS_TOKEN`: The personal access token (classic) used to create pull

requests with a converted workflow (requires `repo` and `workflow` scopes).
- `GITHUB_INSTANCE_URL`: The URL to the target GitHub instance (for example, `https://github.com`).
- `BAMBOO_ACCESS_TOKEN`: The Bamboo personal access token used to authenticate with your Bamboo instance.
- `BAMBOO_INSTANCE_URL`: The URL to the Bamboo instance (for example, `https://bamboo.example.com`).

These environment variables can be specified in a `.env.local` file that is loaded by GitHub Actions Importer when it is run.

Optional arguments

There are optional arguments you can use with the GitHub Actions Importer subcommands to customize your migration.

`--source-file-path`

You can use the `--source-file-path` argument with the `dry-run` or `migrate` subcommands.

By default, GitHub Actions Importer fetches pipeline contents from the Bamboo instance. The `--source-file-path` argument tells GitHub Actions Importer to use the specified source file path instead.

For example:

```
gh actions-importer dry-run bamboo build --plan-slug IN-COM -o tmp/bamboo --
source-file-path ./path/to/my/bamboo/file.yml
```

`--config-file-path`

You can use the `--config-file-path` argument with the `audit`, `dry-run`, and `migrate` subcommands.

By default, GitHub Actions Importer fetches pipeline contents from the Bamboo instance. The `--config-file-path` argument tells GitHub Actions Importer to use the specified source files instead.

Audit example

In this example, GitHub Actions Importer uses the specified YAML configuration file to perform an audit.

```
gh actions-importer audit bamboo -o tmp/bamboo --config-file-path
"./path/to/my/bamboo/config.yml"
```

To audit a Bamboo instance using a config file, the config file must be in the following format, and each `repository_slug` must be unique:

```
source_files:
  - repository_slug: IN/COM
    path: path/to/one/source/file.yml
  - repository_slug: IN/JOB
    path: path/to/another/source/file.yml
```

Dry run example

In this example, GitHub Actions Importer uses the specified YAML configuration file as the source file to perform a dry run.

The repository slug is built using the `--plan-slug` option. The source file path is matched and pulled from the specified source file.

```
gh actions-importer dry-run bamboo build --plan-slug IN-COM -o tmp/bamboo --
config-file-path "./path/to/my/bamboo/config.yml"
```

Supported syntax for Bamboo pipelines

The following table shows the type of properties that GitHub Actions Importer is currently able to convert.

Bamboo	GitHub Actions	Status
`environments`	`jobs`	Supported
`environments.<environment_id>`	`jobs.<job_id>`	Supported
`<job_id>.artifacts`	`jobs.<job_id>.steps.actions/upload-artifact`	Supported
`<job_id>.artifact-subscriptions`	`jobs.<job_id>.steps.actions/download-artifact`	Supported
`<job_id>.docker`	`jobs.<job_id>.container`	Supported
`<job_id>.final-tasks`	`jobs.<job_id>.steps.if`	Supported
`<job_id>.requirements`	`jobs.<job_id>.runs-on`	Supported
`<job_id>.tasks`	`jobs.<job_id>.steps`	Supported
`<job_id>.variables`	`jobs.<job_id>.env`	Supported
`stages`	`jobs.<job_id>.needs`	Supported
`stages.<stage_id>.final`	`jobs.<job_id>.if`	Supported
`stages.<stage_id>.jobs`	`jobs`	Supported
`stages.<stage_id>.jobs.<job_id>`	`jobs.<job_id>`	Supported
`stages.<stage_id>.manual`	`jobs.<job_id>.environment`	Supported
`triggers`	`on`	Supported
`dependencies`	`jobs.<job_id>.steps.<gh cli step>`	Partially Supported
`branches`	Not applicable	Unsupported
`deployment.deployment-permissions`	Not applicable	Unsupported
`environment-permissions`	Not applicable	Unsupported

`notifications`	Not applicable	Unsupported
`plan-permissions`	Not applicable	Unsupported
`release-naming`	Not applicable	Unsupported
`repositories`	Not applicable	Unsupported

For more information about supported Bamboo concept and plugin mappings, see the `github/gh-actions-importer` repository.

Environment variable mapping

GitHub Actions Importer uses the mapping in the table below to convert default Bamboo environment variables to the closest equivalent in GitHub Actions.

Bamboo	GitHub Actions
`bamboo.agentId`	`${{ github.runner_name }}`
`bamboo.agentWorkingDirectory`	`${{ github.workspace }}`
`bamboo.buildKey`	`${{ github.workflow }}-${{ github.job }}`
`bamboo.buildNumber`	`${{ github.run_id }}`
`bamboo.buildPlanName`	`${{ github.repository }}-${{ github.workflow }}-${{ github.job }`
`bamboo.buildResultKey`	`${{ github.workflow }}-${{ github.job }}-${{ github.run_id }}`
`bamboo.buildResultsUrl`	`${{ github.server_url }}/${{ github.repository }}/actions/runs/${{ github.run_id }}`
`bamboo.build.working.directory`	`${{ github.workspace }}`
`bamboo.deploy.project`	`${{ github.repository }}`
`bamboo.ManualBuildTriggerReason.userName`	`${{ github.actor }}`
`bamboo.planKey`	`${{ github.workflow }}`
`bamboo.planName`	`${{ github.repository }}-${{ github.workflow }}`
`bamboo.planRepository.branchDisplayName`	`${{ github.ref }}`
`bamboo.planRepository.<position>.branch`	`${{ github.ref }}`
`bamboo.planRepository.<position>.branchName`	`${{ github.ref }}`
`bamboo.planRepository.<position>.name`	`${{ github.repository }}`
`bamboo.planRepository.<position>.repositoryUrl`	`${{ github.server }}/${{ github.repository }}`
`bamboo.planRepository.<position>.revision`	`${{ github.sha }}`
`bamboo.planRepository.<position>.username`	`${{ github.actor}}`

`bamboo.repository.branch.name`	`${{ github.ref }}`
`bamboo.repository.git.branch`	`${{ github.ref }}`
`bamboo.repository.git.repositoryUrl`	`${{ github.server }}/${{ github.repository }}`
`bamboo.repository.pr.key`	`${{ github.event.pull_request.number }}`
`bamboo.repository.pr.sourceBranch`	`${{ github.event.pull_request.head.ref }}`
`bamboo.repository.pr.targetBranch`	`${{ github.event.pull_request.base.ref }}`
`bamboo.resultsUrl`	`${{ github.server_url }}/${{ github.repository }}/actions/runs/${{ github.run_id }}`
`bamboo.shortJobKey`	`${{ github.job }}`
`bamboo.shortJobName`	`${{ github.job }}`
`bamboo.shortPlanKey`	`${{ github.workflow }}`
`bamboo.shortPlanName`	`${{ github.workflow }}`

Note: Unknown variables are transformed to `${{ env.<variableName> }}` and must be replaced or added under `env` for proper operation. For example, `${bamboo.jira.baseUrl}` will become `${{ env.jira_baseUrl }}`.

System Variables

System variables used in tasks are transformed to the equivalent bash shell variable and are assumed to be available. For example, `${system.<variable.name>}` will be transformed to `$variable_name`. We recommend you verify this to ensure proper operation of the workflow.

Legal notice

Portions have been adapted from https://github.com/github/gh-actions-importer/ under the MIT license: