GoBuild

Overview

The **GoBuild** native step performs a build from Go (GoLang) source.

Page Contents

YAML Schema

The YAML schema for GoBuild native step is as follows:

GoBuild

```
pipelines:
  - name: <string>
    steps:
      - name: <string>
        type: GoBuild
        configuration:
          #inherits all the tags from bash; https://www.jfrog.com/confluence/display/JFROG/Bash
          sourceLocation:   <string>  # optional
          outputLocation:   <string>  # optional
          outputFile:       <string>  # optional
          resolverRepo:     <string>  # optional
          repository:       <string>  # optional
          goCommand:        <string>  # optional

          integrations:
            - name:         <artifactory integration>  # may be required
          inputResources:
            - name:         <GitRepo resource>         # required
            - name:         <FileSpec resource>        # optional

        execution:
          onStart:
            - echo "Preparing for work..."
          onSuccess:
            - echo "Job well done!"
          onFailure:
            - echo "uh oh, something went wrong"
          onComplete:                                  #always
            - echo "Cleaning up some stuff"
```

Tags

name

An alphanumeric string (underscores are permitted) that identifies the step.

type

Must be `GoBuild` for this step type.

configuration

Specifies all configuration selections for the step's execution environment. This step inherits the Bash/PowerShell step configuration tags, including these pertinent tags:

Tag	Description of usage	Required /Optional
`integrations`	Specifies an Artifactory Integration where modules will be published. If a FileSpec resource is specified in `inputResources` then this is optional. Otherwise, it is required.	May be required
`inputResources`	Must specify a GitRepo resource that has Go source files in `sourceLocation`. Also may specify an optional FileSpec resource that specifies what files to copy to `sourceLocation` to build.	Required Optional

In addition, these tags can be defined to support the step's native operation:

> **Tags derived from Bash**
>
> All native steps derive from the Bash step. This means that all steps share the same base set of tags from Bash, while native steps have their own additional tags as well that support the step's particular function. So it's important to be familiar with the Bash step definition, since it's the core of the definition of all other steps.

Tag	Description of usage	Required /Optional
`sourceLocation`	Location where the Go source files are available, relative to the root of the GitRepo repository. If not specified, the default is the root of the GitRepo repository.	Optional
`outputLocation`	Location where the built Go modules should be published.	Optional
`outputFile`	File that has the output of the Go command. The default filename is the name of the step.	Optional
`resolverRepo`	Name of the Artifactory repository to be used to resolve dependencies.	Optional
`repository`	Alternative to `resolverRepo`. Only one of these options may be specified.	Optional
`goCommand`	Specifies a command line string of options to use with the Go client. **Default**: `build -o $outputLocation/$outputFile`	Optional

execution

Declares collections of shell command sequences to perform for pre- and post-execution phases:

Tag	Description of usage	Required/Optional
`onStart`	Commands to execute in advance of the native operation	Optional
`onSuccess`	Commands to execute on successful completion	Optional
`onFailure`	Commands to execute on failed completion	Optional
`onComplete`	Commands to execute on any completion	Optional

The actions performed for the `onExecute` phase are inherent to this step type and may not be overridden.

Examples

The following examples show how to configure a GoBuild step.

Full Pipeline Example

- This example requires an Artifactory Integration and a GitHub Integration.
- The Pipelines DSL for this example is available in this repository in the JFrog GitHub account.
- For a full tutorial, see Pipeline Example: Go Build.

```
# This config file is templatized so that it can be easily customized. Values can be provided with a values.
yml file. For more information, see the 'Pipeline Example: Go Build' quickstart.
template: true   # required for local templates
valuesFilePath: ./values.yml

resources:
  # Sample Go app in a GitRepo
  - name: go_repo
    type: GitRepo
    configuration:
      path: {{ .Values.repoPath }}
      branches:
        include: main
      gitProvider: {{ .Values.gitProvider }}

  # Build info for the published Go app
  - name: go_buildinfo
    type: BuildInfo
    configuration:
      sourceArtifactory: {{ .Values.artifactory }}


pipelines:
  - name: go_build_pipeline_example
    steps:
      # Build the Go sample app from the GitRepo. Docs at https://www.jfrog.com/confluence/display/JFROG
/GoBuild
      - name: build_go
        type: GoBuild
        configuration:
          sourceLocation: .
          resolverRepo: go-virtual
          noRegistry: true
          inputResources:
            - name: go_repo
          integrations:
            - name: {{ .Values.artifactory }}

              # Publish the Go sample app binary to Artifactory. Docs at https://www.jfrog.com/confluence
/display/JFROG/GoPublishBinary
      - name: publish_go_binary
        type: GoPublishBinary
        configuration:
          inputSteps:
            - name: build_go
          targetRepository: go-local
          integrations:
            - name: {{ .Values.artifactory }}

      # Publish the Go sample app build info. Docs at https://www.jfrog.com/confluence/display/JFROG
/PublishBuildInfo
      - name: publish_build
        type: PublishBuildInfo
        configuration:
          inputSteps:
            - name: publish_go_binary
          outputResources:
            - name: go_buildinfo
```

Default Locations

A GoBuild step using default locations.

GoBuild

```
pipelines:
  - name: goBuildPipeline
    steps:
      - name: goBuildStep
        type: GoBuild
        configuration:
          inputResources:
            - name: gitRepoResource
          integrations:
            - name: artifactory_integration
```

FileSpec Input

A GoBuild step with a FileSpec input providing files for the build that are not in the GitRepo and resolverRepo specifying an Artifactory repository to use when resolving dependencies.

GoBuild

```
pipelines:
  - name: goBuildPipeline
    steps:
      - name: goBuildStep
        type: GoBuild
        configuration:
          resolverRepo: repo
          inputResources:
            - name: gitRepoResource
            - name: fileSpec
```

Alternative Source Location in Git Repo

A GoBuild step with an alternative source location in the GitRepo and an alternative Go command for the build.

GoBuild

```
pipelines:
  - name: goBuildPipeline
    steps:
      - name: goBuildStep
        type: GoBuild
        configuration:
          sourceLocation: "app/go"
          goCommand: "build -insecure -o output/outputFile"
          inputResources:
            - name: gitRepoResource
          integrations:
            - name: artifactory_integration
```

How it Works

When you use the **GoBuild** native step in a pipeline, it performs the following functions in the background:

- jfrog rt config (to configure the JFrog CLI with the Artifactory credentials in the input FileSpec if there is no input integration)
- jfrog rt use (to set the current default Artifactory configuration)
- cp (if there is an input FileSpec, copy the files to the root of the cloned GitRepo)
- jfrog rt go-config (configure the repository to resolve dependencies)
- jfrog rt go (build)
- add_run_variables (save information about this step for future steps)

- add_run_files (save the output and the build information in the run state for later publish steps)
- jfrog rt build-collect-env (collect environment variables)

Related Topics

Go Build Quickstart