Sending Build Status to Source Control

Overview

A pipeline can send information about its current build status (*processing*, *success*, or *failure*) to a source repository for logging and display.

Most source control providers feature the ability to receive and log build status results from an external CI automation tool such as Pipelines. Source control users can then view those status logs through the source control's UI, or retrieve them through the source control manager's API.

Pipelines provides the update_commit_status utility function to send build status information to a GitRepo resource, associated with the sha of the GitRepo's latest commit.

You may wish to use this function to:

- Identify whether a pull request can successfully build prior to merge
- View the results of your build in your source provider UI
- Make build results visible to those who do not have accounts on your organization's JFrog Platform Deployment, or do not have permissions to view the pipeline in Pipelines
- Provide build status results to other automation tools through the source control API

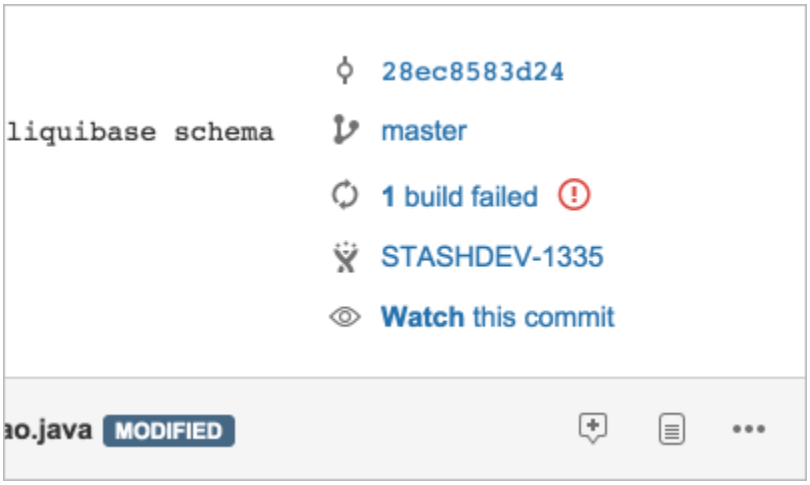
Page Contents

Source Control Build Status

An ability to log status information with your commits is a feature provided by many source control repository managers. Its availability and operation will differ according to your provider.

For example, the status checks feature of GitHub will mark each commit listed in the GitHub UI with the build status received, and the details of the status log can be viewed by clicking the mark. The build status log data can also be retrieved through the GitHub REST APIs

Similarly, the build status feature in Bitbucket Server displays the build status for the commit in the Bitbucket UI, and makes that information available through REST resources.



Sending Build Status from Pipelines

Where many CI automation servers require a plugin or custom integration to send build status to a source control repository, Pipelines provides a built-in utility function that can be used in any step.

To send build status from a pipeline step to a commit in a source repository, you must:

1. Declare the GitRepo in the step's `inputresources` .
2. Use the `update_commit_status` utility function in any of the step's `execution` blocks: `onStart`, `onExecute`, `onFailure`, or `onSuccess`.

The format of the `update_commit_status` function is:

```
update_commit_status <gitRepo resource name> --status <status> --message <message> --context <context>
```

You must specify the GitRepo resource in the `update_commit_status` function. The remaining parameters are all optional. By default, the function will infer the status from the `execution` block where it is invoked. Default message and context strings are constructed from the step and pipeline name.

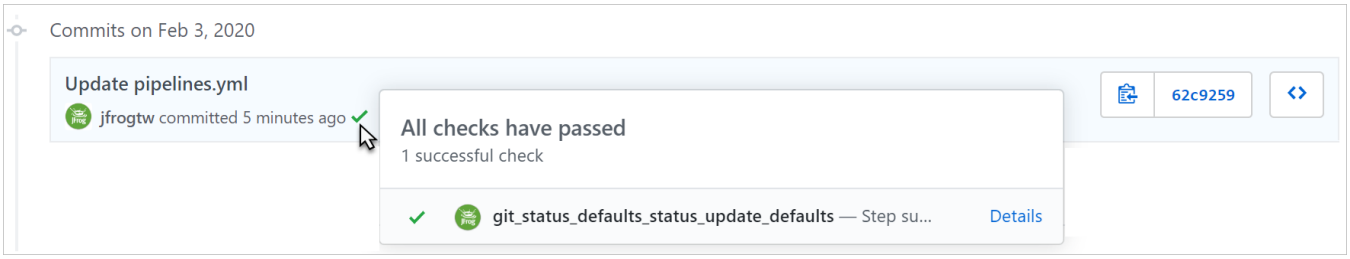
For example, this step will update the commit's build status for each execution phase using the defaults:

```
pipelines:
  - name: git_status_defaults
    steps:
      - name: status_update_defaults
        type: Bash
        configuration:
          inputResources:
            - name: myGitRepo
        execution:
          onStart:
            - update_commit_status myGitRepo               # Status: "processing"
          onExecute:
            - update_commit_status myGitRepo               # Status: "processing"
            - echo "Hello World!"
          onFailure:
            - update_commit_status myGitRepo               # Status: "failure"
          onSuccess:
            - update_commit_status myGitRepo               # Status: "success"
```

When the above example pipeline is run:

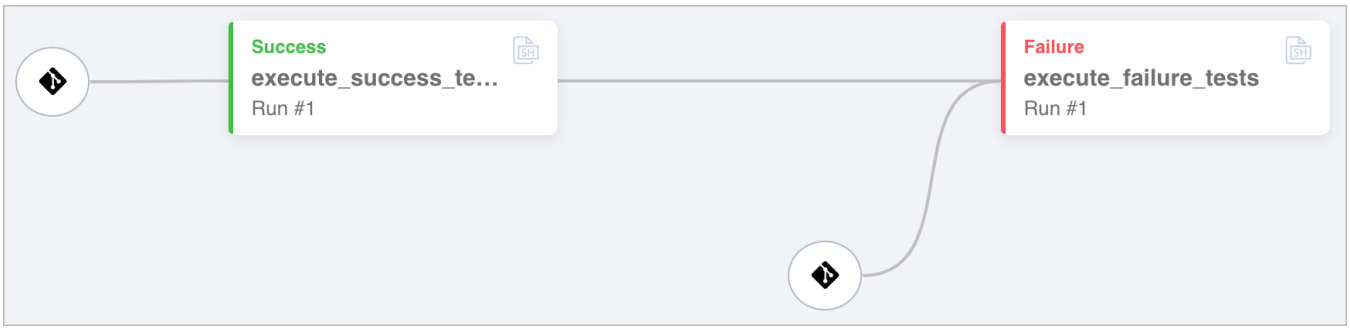
- The receiving source repository will create a status log entry for the step.
- Each call to `update_commit_status` will overwrite the status and message values of that log entry.

The resulting build status log for the commit can be viewed in the source repository's UI (on GitHub):



Example Pipeline

The following example pipeline demonstrates sending build status messages to the source control repository that will create a status log entry for each executed step.



The `resources` for the pipeline declares the GitRepo of the source control repository.

pipelines.resources.yml

```
resources:
  - name: myGitRepo
    type: GitRepo
    configuration:
      gitProvider: myGitHub
      path: myaccount/myproject
```

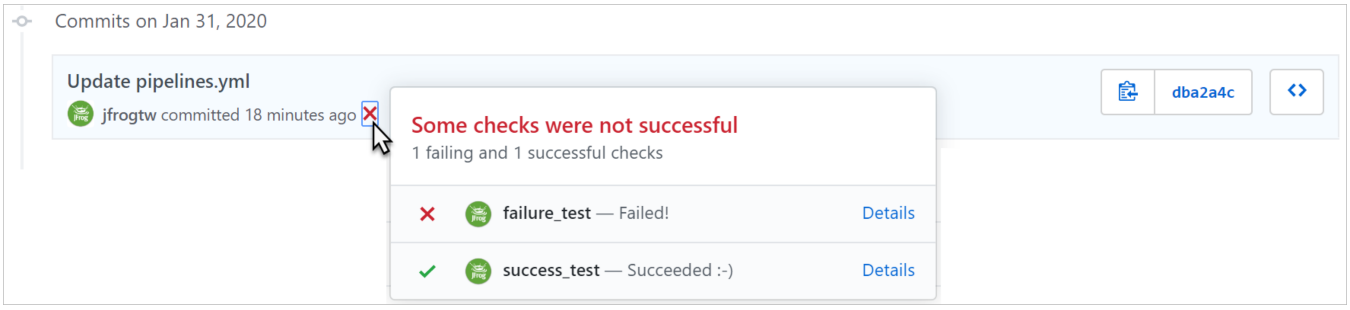
The pipeline consists of two dummy steps, one that will always execute successfully and one that will always fail to execute. The `update_commit_status` function will send build status messages to the source repository for each step.

pipelines.steps.yml

```
pipelines:
  - name: git_status_test
    steps:
      ##
      ## Step 1: Execute a simple, always successful test
      ##
      - name: success_test
        type: Bash
        configuration:
          inputResources:
            - name: myGitRepo
        execution:
          onStart:
            - update_commit_status myGitRepo --message "starting..." --context "$step_name"
          onExecute:
            - update_commit_status myGitRepo --message "running..." --context "$step_name"
            - echo "Hello World!"
          onFailure:
            - update_commit_status myGitRepo --message "Failed!" --context "$step_name"
          onSuccess:
            - update_commit_status myGitRepo --message "Succeeded :-)" --context "$step_name"

      ##
      ## Step 2: Execute a simple, always failing test
      ##
      - name: failure_test
        type: Bash
        configuration:
          inputResources:
            - name: myGitRepo
              trigger: false
          inputSteps:
            - name: success_test
        execution:
          onStart:
            - update_commit_status myGitRepo --message "starting..." --context "$step_name"
          onExecute:
            - update_commit_status myGitRepo --message "running..." --context "$step_name"
            - cd fail               # No such directory -- guaranteed to fail
          onFailure:
            - update_commit_status myGitRepo --message "Failed!" --context "$step_name"
          onSuccess:
            - update_commit_status myGitRepo --message "Succeeded :-)" --context "$step_name"
```

When viewed in the source repository's UI (on GitHub), the build status log for the commit shows both a successful and a failed execution of each step:



Since the context option is always set to the `$step_name` environment variable in the example, the receiving source control repository will create a build status log entry for each step.

If you wanted to log only a single build status for the entire pipeline, you might set the context option to the `$pipeline_name` environment variable instead. To log a build status for each run of the pipeline, you might combine it with the `$run_number` environment variable:

```
update_commit_status myGitRepo --context "$pipeline_name:$run_number"
```