Adding a Slack Bot to a Concourse Pipeline

Contents

This topic explains how to create a Slack notification bot for the build status of a Concourse pipeline.

Overview

The Cloud Foundry Community organization provides a `slack-notification` Concourse resource type for sending notifications to Slack. To enable Slack notifications, you must add the `slack-notification` resource type and a corresponding Slack resource to your Concourse `pipeline.yml` and `plan.yml` files. Because these YAML files are auto-generated when you update a pipeline, you must configure `lib/pub_tools/scheme.rb` and `Rakefile` to add the resource type and resource.

> **⚠ Warning**
>
> Do not directly modify Concourse YAML files. If you add a resource type or resource to a YAML file, Concourse overwrites your changes when you update the pipeline.

For more information about the `slack-notification` resource type, see Slack notification sending resource on GitHub. For more information about resource types and resources, see Resource Types and Resources in the Concourse documentation.

> **💡 Hint**
>
> You can use this procedure as a template for adding other custom resource types to a Concourse pipeline.

Set up Slack WebHook Integration

Slack provides an Incoming WebHook integration that can post messages from external sources to Slack. You can add this integration to a Concourse pipeline to send messages to Slack about the pipeline's build status.

To set up the Slack WebHook:

1. In a web browser, go to the Slack integration page for Incoming WebHooks.
2. Select the Slack channel that you want the bot to post to in **Integration Settings**.
3. Record the provided **Webhook URL**.
4. Customize the **Name** and **Icon**.
5. Click **Save Settings**.

6. Open `workspace/docs-concourse-creds/credentials.yml` in a text editor.
7. Add the following line to `credentials.yml`:

```
slack-webhook: YOUR-WEBHOOK-URL
```

   Where `YOUR-WEBHOOK-URL` is the **Webhook URL** you recorded in the previous step. When you update your pipeline, Concourse uses this webhook to connect to the Slack bot.

Add the Slack Resource Type

To enable Slack notifications, you must add the `slack-notification` resource type to `pipeline.yml`. You can do this by editing the `Rakefile` file that auto-generates `pipeline.yml`.

To add the `slack-notification` resource type to `pipeline.yml`:

1. Open `workspace/concourse-scripts-docs/Rakefile` in a text editor.
2. Locate the following line:

```
File.write(File.join(pipeline_name, 'pipeline.yml'), "# Generated
file...\n" + yaml)
```

3. Edit the above line to add the `slack-notification` resource type as follows:

```
File.write(File.join(pipeline_name, 'pipeline.yml'), "# Generated
file...\n" + yaml +
"resource_types:
- name: slack-notification
  type: docker-image
  source:
    repository: cfcommunity/slack-notification-resource)
    tag: latest"
```

Add the Slack Resource

After you add the `slack-notification` resource type to your pipeline, you must add a corresponding Slack resource to `pipeline.yml`. You can do this by editing the `scheme.rb` file that auto-generates `pipeline.yml`.

To add the Slack resource to `pipeline.yml`:

1. Open `workspace/concourse-scripts-docs/lib/pub_tools/scheme.rb` in a text editor.
2. Edit the `deploy_resources` definition to add the Slack resource as follows:

```
def deploy_resources
      [
        {
          'name' => 'concourse-scripts-bundle',
          'type' => 's3',
          'source' => {
            'bucket' => 'concourse-interim-steps',
            'versioned_file' => 'concourse-scripts-bundle.tar.gz',
            'private' => true,
            'access_key_id' => '{{aws-access-key}}',
            'secret_access_key' => '{{aws-secret-key}}'
          }
        },

        {
          'name' => 'notify',
          'type' => 'slack-notification',
          'source' => {
           'url' => '((slack-webhook))'
            },
        }

      ]
    end
```

Add Step Hooks

Concourse provides a `on_failure` step hook that runs a parent step fails. You can add a `on_failure` step hook to a bind or deploy task in `plan.yaml`. When the parent task fails, Concourse triggers the `notify` resource and sends a message to a Slack channel. You can add the step hook by editing the `scheme.rb` file that auto-generates `plan.yml`.

For more information about the `on_failure` step hook, see **`on_failure`** Step Hook in the Concourse documentation.

> **Hint**
>
> If you want to post a message to Slack when a task succeeds, use the `on_success` step hook instead. For more information, see **`on_success`** Step Hook in the Concourse documentation.

To add the `on_failure` step hook to `plan.yml`:

1. Open `workspace/concourse-scripts-docs/lib/pub_tools/scheme.rb` in a text editor.
2. Edit the `bind_plan` definition to add the `on_failure` step hook as follows:

```
def bind_plan
      aggregate = config.all_repos.map { |repo|
        { 'get' => repo.friendly_name, 'resource' => repo.full_name,
'trigger' => repo.trigger, 'params' => { 'submodules' => 'none' } }
      }

      aggregate << { 'get' => 'bookbinder-release', 'resource' =>
'bookbinder-release-complete', 'trigger' => true }

      [
        { 'aggregate' => aggregate },
        { 'task' => "#{name}-bind", 'file' => "concourse-scripts/#
{pipeline}/#{group}/#{name}-bind/task.yml",
        'on_failure' => {
          'put' => 'notify',
          'params' =>
             {'text' => 'BOT-MESSAGE'}
        },
        { 'aggregate' =>
          [
            { 'put' => s3_resource, 'params' => { 'file' =>
'bind_output/final_app.tar.gz'} },
          ]
        }
      ]
    end
```

Where `BOT-MESSAGE` is the message that you want to post to Slack when documentation fails to build.

> **Hint**
>
> For a descriptive message, you can configure the bot message with Concourse environment variables. For more information, see Metadata in the Concourse documentation.

3. Edit the `deploy_plan` definition to add the `on_failure` step hook as follows:

```
 def deploy_plan(environment, stream_id, dependency, trigger)
      previous_job = "#{name}-#{dependency}"
      environment = environment.gsub('-', '_')

      [
        {
          'aggregate' => [
            { 'get' => 'concourse-scripts', 'resource' => 'concourse-
scripts-docs-master', 'passed' => [previous_job]},
            { 'get' => 'concourse-scripts-bundle' },
            { 'get' => 'site-source', 'resource' => s3_resource, 'passed'
=> [previous_job], 'trigger' => trigger }
          ]
        },
        { 'task' => 'deploy', 'file' => 'concourse-
scripts/deploy_task.yml', 'params' => {
          'DEPLOY_DETAILS' => "concourse-scripts/#{pipeline}/#
{group}/config.yml",
          'DEPLOY_ENV' => environment,
          'BOOK_ID' => stream_id,
          'USERNAME' => "{{cloud-foundry-username}}",
          'PASSWORD' => "{{cloud-foundry-password}}"
        },
        'on_failure' => {
          'put' => 'notify',
          'params' =>
              {
                'text' => 'BOT-MESSAGE'}
        }
      }
      ]
    end
```

Where `BOT-MESSAGE` is the message that you want to post to Slack when documentation fails to deploy.

Update the Pipeline

To apply your changes to `pipeline.yml` and `plan.yml`, you must update your Concourse pipeline.

To update your pipeline:

1. Run the following command:

   ```
   rake scheme:update[PIPELINE/PIPELINE-GROUP]
   ```

   Where:
   - `PIPELINE` is the Concourse pipeline you are updating.
   - `PIPELINE-GROUP` is the Concourse pipeline group you are updating.

   Alternatively, to update all of the pipeline groups in a pipeline, run the following command:

   ```
   rake scheme:update_all[PIPELINE]
   ```

2. Run the following command:

```
rake fly:set_pipeline[PIPELINE]
```

Where `PIPELINE` is the Concourse pipeline from the previous step.

3. When prompted, confirm that the new configuration for the pipeline is correct.
4. Add, commit, and push your changes to GitHub.
