Rockefeller Documentation

Release 0.0.6

Michael Nguyen

Jan 14, 2019

Getting Started

Rockefeller is a tool to easily create AWS CodePipelines, including support for Handel deployments.

CHAPTER 1

Introduction

Rockefeller is a command-line library that helps you easily create Continuous Delivery pipelines in the AWS CodePipeline service.

Included in this library is the support for doing deployments using the Handel deployment library.

1.1 How does this library work?

You specify a file called *rockefeller.yml* in your code repository. This file contains a YAML specification of how the library should configure your pipeline.

Once you've defined your *rockefeller.yml* file, you can run the library. It will prompt you for further pieces of information, after which it will create the pipeline.

CHAPTER 2

Installation

Rockefeller is a CLI tool written in Node.js. In order to install it, you will first need Node.js installed on your machine.

2.1 Installing Node.js

The easiest way to install Node.js is to download the compiled binaries from the Node.js website. Rockefeller requires Node.js *version 6.x or greater* in order to run.

Once you have completed the installation on your machine, you can verify it by running these commands:

```
node --version
npm --version
```

The above commands should show you the versions of Node and NPM, respectively.

2.2 Installing Rockefeller

Once you have Node.js installed, you can use the NPM package manager that is bundled with Node.js to install Rockefeller:

```
npm install -g rockefeller
```

When the above commands complete successfully, you should be able to run the Rockefeller CLI to deploy your application.

2.3 Next Steps

See the *Tutorial* section for a tutorial on deploying a simple Node.js application to AWS using Rockefeller.

CHAPTER 3

Tutorial

This page contains a tutorial showing how to use Rockefeller to set up a pipeline using Handel for deployments.

Important: Before going through this tutorial, make sure you have installed Rockefeller on your machine as shown in the *Installation* section.

This tutorial also assumes you already have an application with a valid Handel file configured.

3.1 Tutorial

This tutorial contains the following steps:

1. *Write the Rockefeller File*
2. *Write the CodeBuild BuildSpec File*
3. *Deploy the Pipeline*

Follow along with each of these steps in the sections below in order to complete the tutorial.

Note: This tutorial assumes you are deploying a Node.js application. You may need to modify some further things in this tutorial if you are using another platform.

3.1.1 Write the Rockefeller File

We're going to create a single pipeline with three phases:

1. Pull code from a GitHub branch.
2. Build the project using CodeBuild.
3. Deploy the project using Handel.

Create a file named *rockefeller.yml* in the root of your repository with the following contents:

```
version: 1

name: <your-app-name> # Replace with your own app name

pipelines:
  dev:
    phases:
    - type: github
      name: Source
      owner: <your-github-username> # Replace with your own GitHub username
      repo: <your-github-repo> # Replace with your own GitHub repository name
      branch: master
    - type: codebuild
      name: Build
      build_image: aws/codebuild/nodejs:6.3.1
    - type: handel
      name: Deploy
      environments_to_deploy:
      - dev
```

Important: Remember to replace the noted sections in the above file with your own information.

3.1.2 Write the CodeBuild BuildSpec File

Our second phase uses the AWS CodeBuild service to perform any build steps required. This service requires that you put a file called *buildspec.yml* at the root of the repository. This file contains instructions about the commands CodeBuild should run.

Create a file called *buildspec.yml* at the root of your repository with the following contents:

```
version: 0.2

phases:
  build:
    commands:
    - npm install

artifacts:
  files:
  - ./**/*
```

You will likely need to modify this file to run different commands for your application build process. See the CodeBuild documentation for more information on the *buildspec.yml* file.

3.1.3 Deploy the Pipeline

Important: Before running Rockefeller, you must be logged into your AWS account on the command line. You can do this by setting your AWS access keys using the AWS CLI.

See Configuring the AWS CLI for help on doing this once you've installed the AWS CLI.

If you work for an organization that uses federated logins through something like ADFS, then you'll have a different process for logging in on the command-line. In this case, ask your organization how they login to AWS on the command-line.

Now that you have your *rockefeller.yml* and *buildspec.yml* files, you can deploy the pipeline:

```
rockefeller deploy
```

The pipeline will ask a series of questions with additional information and secrets it needs:

```
info:    Welcome to the Rockfeller setup wizard
? Please enter the name of the pipeline from your rockefeller.yml file that you would␣
↪like to deploy
? Please enter the name of the account where your pipeline will be deployed
? Please enter the path to the directory containing the Handel account configuration␣
↪files
? 'GitHub' phase - Please enter your GitHub access token
```

Once you've provided all required information, the pipeline will be created with output something like the following:

```
info:    Creating source phase 'GitHub'
info:    Creating build phase CodeBuild project my-pipeline-dev-Build
info:    Creating CodePipeline for the pipeline 'my-pipeline-dev'
info:    Finished creating pipeline in 111111111111
```

3.2 Next Steps

Now that you've deployed a simple pipeline, where do you go next?

3.2.1 Learn more about Rockefeller

Read through the following documents in the *Rockefeller Basics* section:

- *Using Rockefeller*
- *Rockefeller File*

3.2.2 Learn about the different phase types

Once you understand Rockefellers's basic configuration, see the *Supported Pipeline Phase Types* section, which contains information about the different phase types supported in Rockefeller

CHAPTER 4

CLI Reference

The Rockefeller command-line interface should be run in a directory with a *rockefeller.yml* file.

It defines four commands: *check*, *deploy*, *delete* and *list-required-secrets*

4.1 *rockefeller check*

Validates that a given Rockefeller configuration is valid.

4.1.1 Parameters

rockefeller check does not accept parameters.

4.2 *rockefeller deploy*

Validates and deploys the resources in a given environment.

4.2.1 Parameters

Parameter	Type	Required	Default	Description
–pipeline <value>	string	Yes		The pipeline from your rockefeller.yml file that you wish to deploy.
–account_name <value>	string	Yes		The account you are deploying into.
–secrets <value>	*Secrets*	yes		The base64 encoded JSON string of the deploy secrets. See *Secrets*

4.2.2 Secrets

A base64 encoded array of secrets objects. Note that the required secrets can be obtained with *rockefeller list-required-secrets*.

```
[
  {
      "phaseName": "Github", // The phase the secret is associated with.
      "name": "githubAccessToken", // The name of the secret
      "message": "'Github' phase - Please enter your GitHub access token", // This is␣
↪not necessary, but will be present if the original object was obtained from␣
↪rockefeller list-required-secrets.
      "value": "ABCDEFGHIJKLMNOPQRSTUVWXYZ" // The secret's value
  }
]
```

4.3 *rockefeller delete*

Deletes the AWS CodePipeline.

4.3.1 Parameters

Parameter	Type	Required	Default	Description
–pipeline <value>	string	Yes		The pipeline from your rockefeller.yml file that you wish to delete.
–account_name <value>	string	Yes		The account you are deploying into.

4.4 *rockefeller list-required-secrets*

Returns a JSON string with all of the secrets required for the pipeline.

4.4.1 Parameters

Parameter	Type	Required	Default	Description
–pipeline <value>	string	Yes		The pipeline from your rockefeller.yml file that you want to retreive required secrets from.

4.4.2 Example Response

```
[
  {
    "phaseName": "Github",
    "name": "githubAccessToken",
```

(continues on next page)

(continued from previous page)

```
    "message": "'Github' phase - Please enter your GitHub access token"
  },
  {
    "phaseName": "npmDeploy",
    "name": "npmToken",
    "message": "npmDeploy' phase - Please enter your NPM Token"
  },
  {
    "phaseName": "pypiDeploy",
    "name": "pypiUsername",
    "message": "'pypiDeploy' phase - Please enter your PyPi username"
  },
  {
    "phaseName": "pypiDeploy",
    "name": "pypiPassword",
    "message": "'pypiDeploy' phase - Please enter your PyPi password"
  },
  {
    "phaseName": "RunscopeTests",
    "name": "runscopeTriggerUrl",
    "message": "'RunscopeTests' phase - Please enter your Runscope Trigger URL"
  },
  {
    "phaseName": "RunscopeTests",
    "name": "runscopeAccessToken",
    "message": "'RunscopeTests' phase - Please enter your Runscope Access Token"
  },
  {
    "phaseName": "Notify",
    "name": "slackUrl",
    "message": "'Notify' phase - Please enter the URL for Slack Notifications"
  }
]
```

CHAPTER 5

Using Rockefeller

Rockefeller is a command-line utility that you can use to facilitate creation of CodePipelines that use the Handel library for deployment. This page details how to use this library.

5.1 AWS Permissions

When you run Rockefeller to deploy a new pipeline, you must run it with a set of AWS IAM credentials that have administrator privileges. This is because Rockefeller creates roles for the deploy phase of the pipeline that have administrator privileges.

Once the pipeline is deployed, it will only use the created role for deployments, so you won't need to keep the user around with administrator privileges. Since human users are recommended to have non-administrative permissions, it is recommended you use a temporary user with admin permissions to create the pipeline, then delete that user once the pipeline is created.

5.2 Creating New Pipelines

To deploy a new pipeline, do the following:

1. Create a new *Rockefeller File* in your repository.
2. Install Rockefeller:

```
npm install -g rockefeller
```

3. Ensure you have your AWS credentials configured on the command line.

```
# This command will prompt you for your AWS Access Key ID and Secret␣
↪Access Keys
aws configure
```

> **Note:** If you specified a profile when running *aws configure* above, you'll need to make Rockefeller aware of which profile to use by setting the AWS_PROFILE environment variable.
>
> For example, if you configured your credentials in a profile named *my-account*, you'll run `export AWS_PROFILE=my-account` on Mac/Linux to set the environment variable that tells Rockefeller which profile to use.

4. Run Rockefeller:

```
rockefeller deploy
```

5. Rockefeller will walk you through a series of questions, asking you to provide further input:

```
Welcome to the Rockefeller setup wizard
? Please enter the name of the pipeline from your rockefeller.yml file␣
↪that you would like to deploy prd
? Please enter the name of the account where your pipeline will be␣
↪deployed my-account
? Please enter the path to the directory containing the Handel account␣
↪configuration files /path/to/account/config/files
? Please enter a valid GitHub access token (CodePipeline will use this to␣
↪pull your repo) SOMEFAKETOKEN
```

After you provide the appropriate input, Rockefeller will deploy the pipeline with the specified phases.

CHAPTER 6

Rockefeller File

Rockefeller requires you to specify a pipeline specification file, which contains information on how your pipeline should be configured. This specification file must be named *rockefeller.yml*. It doesn't contain any secrets, so it may be committed to your repository alongside your Handel file.

6.1 Rockefeller File Specification

The Rockefeller file is a YAML file that has the following schema:

```
version: 1

name: <app_name>

pipelines:
  <pipeline_name>:
    phases:
    - type: <phase_type>
      name: <phase_name>
      <phase_params>
```

The above file schema shows that you can specify one or more pipelines, giving them a unique <pipeline_name>. In each pipeline, you specify an ordered series of phases. Each phase has a <type> and a <name>. The type field is defined by Rockefeller, and the name field is one that you specify.

In addition, you must specify a top-level *name* field, which is a string you choose for the overall name of your application.

Each phase then has additional parameters that are specific to the phase type. See the *Supported Pipeline Phase Types* section for information on each phase type.

Important: The first two phases are required to be of a certain type. The first phase must be a source code action type such as *github*. The second phase must be a build action type such as *codebuild*.

CHAPTER 7

Approval

The *Approval* phase type configures a pipeline phase to require manual approval before proceeding with the rest of the pipeline.

7.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	approval	This must always be *approval* for the Approval phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.

7.2 Secrets

This phase type doesn't prompt for any secrets when creating the pipeline.

7.3 Example Phase Configuration

This snippet of a *rockefeller.yml* file shows the GitHub phase being configured:

```
version: 1

pipelines:
  dev:
    ...
    phases:
```

(continues on next page)

(continued from previous page)

```
  - type: approval
    name: ManualApproval
  ...
```

CHAPTER 8

CloudFormation

The *CloudFormation* phase type configures a pipeline phase to deploy a CloudFormation template

8.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	cloudformation	This must always be *cloudformation* for the CloudFormation phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
template_path	string	Yes		The path in your repository to your CloudFormation template.
deploy_role	string	Yes		The role CloudFormation will use to create your role. This role must already exist in your account and must be assumable by CloudFormation.

8.2 Secrets

This phase type doesn't prompt for any secrets when creating the pipeline.

8.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the CloudFormation phase being configured:

```
version: 1

pipelines:
  dev:
    phases:
    ...
    - type: cloudformation
      name: Deploy
      template_path: cf-stack.yml
      deploy_role: myservicerole
    ...
```

CHAPTER 9

CodeBuild

The *CodeBuild* phase type configures a pipeline phase to build the source code pulled from the repository. The second phase of every pipeline created with Rockefeller must be a build code phase such as this CodeBuild type.

9.1 Build Configuration

You can specify any arbitrary build process in this phase using the buildspec.yml file. You must have this *buildspec.yml* file in the root of your repository or the CodeBuild phase will fail.

9.2 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	codebuild	This must always be *codebuild* for the CodeBuild phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
build_image	string	Yes		The name of the CodeBuild image to use when building your code. See the CodeBuild documentation for a list of images.
environment_variables	map	No	{}	A set of key/value pairs that will be injected into the running CodeBuild jobs.
cache	string	No	*no-cache*	Whether to enable a build cache for this phase. Valid values are *no-cache* and *s3*.
build_role	string	No	Handel-created role	The role that will be assigned to the CodeBuild project. This role must already exist in your account and must be assumable by CodeBuild.

Note: You can use a custom build image in your account's EC2 Container Registry by prefixing the build_image parameter with *<account>/*. For example, *<account>/IMAGE:TAG* will resolve at run-time to

AWS_ACCOUNT_ID.dkr.ecr.AWS_REGION.amazonaws.com/IMAGE:TAG.

Using a custom build image also configures the CodeBuild image in privileged mode, which allows you to run Docker inside your image if needed.

9.3 Secrets

This phase type doesn't prompt for any secrets when creating the pipeline.

9.4 Example Phase Configuration

This snippet of a rockefeller.yml file shows the CodeBuild phase being configured:

```
version: 1

pipelines:
  dev:
    phases:
    ...
    - type: codebuild
      name: Build
      build_image: aws/codebuild/docker:1.12.1
      environment_Variables:
        MY_CUSTOM_ENV: my_custom_value
    ...
```

CHAPTER 10

CodeCommit

The *CodeCommit* phase type configures a pipeline phase to pull source code from CodeCommit. The pipeline is launched when code is pushed to CodeCommit on the specified branch. The first phase of every pipeline created with Rockefeller must be a source code phase such as this CodeCommit type.

10.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	codecommit	This must always be *codecommit* for the CodeCommit phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
repo	string	Yes		The name of the CodeCommit repository containing the source code that will build and deploy in the pipeline.
branch	string	Yes	master	The name of the Git branch in the repository from which the pipeline will be invoked.

10.2 Secrets

This phase type doesn't prompt for any secrets when creating the pipeline.

10.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the CodeCommit phase being configured:

```
version: 1

pipelines:
  dev:
    phases:
    - type: codecommit
      name: Source
      owner: byu-oit-appdev
      repo: aws-credential-detector
      branch: master
    ...
```

CHAPTER 11

GitHub

The *GitHub* phase type configures a pipeline phase to pull source code from GitHub. The pipeline is launched when code is pushed to GitHub on the specified branch. The first phase of every pipeline created with Rockefeller must be a source code phase such as this GitHub type.

11.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	github	This must always be *github* for the GitHub phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
owner	string	Yes		The GitHub username or organization where the repository lives.
repo	string	Yes		The name of the GitHub repository containing the source code that will build and deploy in the pipeline.
branch	string	No	master	The name of the Git branch in the repository from which the pipeline will be invoked.

11.2 Secrets

In addition to the parameters specified in your rockefeller.yml file, this phase will prompt you for the following secret information when creating your pipeline:

- GitHub personal access token.

This is not saved in your rockefeller.yml file because by having the token others can interact with GitHub on your behalf.

11.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the GitHub phase being configured:

```
version: 1

pipelines:
  dev:
    phases:
    - type: github
      name: GitHub
      owner: byu-oit-appdev
      repo: aws-credential-detector
      branch: master
    ...
```

CHAPTER 12

Handel

The *Handel* phase type configures a pipeline phase to deploy one or more of your application environments using the Handel library. You may configure multiple phases of this type if you wish to deploy your application environments across different phases.

12.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	handel	This must always be *handel* for the Handel phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
environments_to_deploy	list<string>	Yes		A list of one or more environment names from your Handel file that you wish to deploy in this phase.

12.2 Secrets

This phase type doesn't prompt for any secrets when creating the pipeline.

12.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the Handel phase being configured:

```
version: 1

pipelines:
```

(continues on next page)

(continued from previous page)

```
  dev:
    phases:
    - type: handel
      name: DevDeploy
      environments_to_deploy:
      - dev
    ...
```

CHAPTER 13

Handel Delete

The *Handel Delete* phase type configures a pipeline phase to delete one or more of your Handel application environments that was previously deployed. This phase is useful if you want to spin up an ephemeral environment, run tests against it, and delete the environment after the tests.

Warning: This environment will DELETE all resources in an environment, including data resources such as RDS, ElastiCache, and DynamoDB!

The data from these will likely be unrecoverable once deleted. You should only use this phase type against ephemeral environments that don't need to persist data.

Use this phase at your own risk. It is highly recommended you double-check which environments are being deleted before adding this phase to a pipeline.

13.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	handel_delete	This must always be *handel_delete* for the Handel Delete phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
environments_to_delete	list<string>	Yes		A list of one or more environment names from your Handel file that you wish to delete in this phase.

13.2 Secrets

This phase type doesn't prompt for any secrets when creating the pipeline.

13.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the Handel phase being configured:

```
version: 1

pipelines:
  dev:
    phases:
    - type: handel_delete
      name: Teardown
      environments_to_delete:
      - dev
    ...
```

CHAPTER 14

Invoke Lambda

The *Invoke Lambda* phase type configures a pipeline phase to execute an arbitrary Lambda function in your account.

14.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	invoke_lambda	This must always be *invoke_lambda* for the Invoke Lambda phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
function_name	string	Yes		The name of the Lambda function you wish to invoke in this phase.
function_parameters	map<string, string>	No		An object of parameter values to pass into the Lambda function.

14.2 Secrets

This phase type doesn't prompt for any secrets when creating the pipeline.

14.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the GitHub phase being configured:

```
version: 1

pipelines:
```

(continues on next page)

(continued from previous page)

```
dev:
  ...
  phases:
  - type: invoke_lambda
    name: InvokeMyFunction
    function_name: my_function_name_to_invoke
    function_parameters:
      myParam1: hello
      myParam2: world
  ...
```

CHAPTER 15

NPM

The *NPM* phase type configures a pipeline phase to deploy one or more of your application npmjs.

15.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	npm	This must always be *npm* for the NPM phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
build_image	string	No	aws/codebuild/nodejs:6.3.1	The code build image needed to deploy project to npm. See here for more info AWS Codebuild Docs

15.2 Secrets

In addition to the parameters specified in your rockefeller.yml file, this phase will prompt you for the following secret information when creating your pipeline:

- NPM Token

For Security reasons these are not saved in your rockefeller.yml file. The NPM token can be found in your .npmrc file see here for more information.

15.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the NPM phase being configured:

```
version: 1

pipelines:
  dev:
    phases:
    ...
    - type: npm
      name: npmDeploy
    ...
```

CHAPTER 16

Pypi

The *Pypi* phase type configures a pipeline phase to deploy one or more of your application environments using the Pypi library.

16.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	pypi	This must always be *pypi* for the Pypi phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
server	string	No	pypi	The full url for the pypi repo ie: https://test.pypi.org/legacy/
build_image	string	No	aws/codebuild/python:3.6.5	The code build image needed to deploy project to pypi. See here for more info AWS Codebuild Docs

16.2 Secrets

In addition to the parameters specified in your rockefeller.yml file, this phase will prompt you for the following secret information when creating your pipeline:

- Pypi Username.
- Pypi Password.

For Security reasons these are not saved in your rockefeller.yml file.

16.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the Pypi phase being configured:

```
version: 1

pipelines:
  dev:
    phases:
    ...
    - type: pypi
      name: pypiDeploy
      server: https://testpypi.python.org/pypi
    ...
```

CHAPTER 17

Runscope

The *Runscope* phase type configures a pipeline phase to execute tests from a Runscope bucket.

17.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	runscope	This must always be *runscope* for the Runscope phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.

17.2 Secrets

This phase will prompt you for the following secret information when creating your pipeline:

- Runscope Trigger URL
- Runscope API Access Token

These secrets are not saved in your rockefeller.yml file because they allow others to invoke your tests and make API calls to Runscope on your behalf.

17.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the GitHub phase being configured:

```
version: 1

pipelines:
  dev:
    ...
    phases:
    - type: runscope
      name: RunscopeTests
    ...
```

CHAPTER 18

Slack Notify

The *Slack Notify* phase type configures a pipeline phase to send a notification to a Slack channel.

18.1 Parameters

Parameter	Type	Required	Default	Description
type	string	Yes	slack_notify	This must always be *slack_notify* for the Slack Notify phase type.
name	string	Yes		The value you want to show up in the CodePipeline UI as your phase name.
message	string	Yes		The message to send to the Slack channel when this phase executes.
channel	string	Yes		The Slack channel you wish to send to. This can either be a username, such as "@dsw88", or a channel, such as "#mydeploys".

Important: In the *channel* parameter above, make sure that you put your channel names in quotes, since YAML treats the # character as a comment and will cause your Rockefeller file to be invalid.

18.2 Secrets

In addition to the parameters specified in your rockefeller.yml file, this phase will prompt you for the following secret information when creating your pipeline:

- Slack notify URL

This is not saved in your rockefeller.yml file because by having this URL others can also post to your Slack instance.

18.3 Example Phase Configuration

This snippet of a rockefeller.yml file shows the GitHub phase being configured:

```
version: 1

pipelines:
  dev:
    ...
    phases:
    - type: slack_notify
      name: Notify
      channel: "#mydeployschannel"
      message: Successfully deployed the app!
    ...
```