**Postman Newman**

* Newman is a command line Collection Runner for Postman. It allows you to run and test a Postman Collection directly from the command line. It is built with extensibility in mind so that you can easily integrate it with your continuous integration servers and build systems.
* Newman maintains feature parity with Postman and allows you to run collections the way they are executed inside the collection runner in the Postman app.
* Newman resides in the [NPM registry](https://www.npmjs.org/package/newman) and on [GitHub](https://github.com/postmanlabs/newman).

**Install Newman**

* Newman is built on Node.js. To run Newman, make sure you have Node.js installed.
* You can [download and install](http://nodejs.org/download/) Node.js on Linux, Windows, and Mac OSX.
* After you install Node.js, Newman is just a command away. Install Newman from npm globally on your system, which allows you to run it from anywhere.
* $ npm install -g newman

**Run collection via Newman**

* The easiest way to run Newman is to run it with a collection. You can run any collection file from your file system.
* To learn how to export collections to share as a file
* $ newman run mycollection.json
* You can also pass a collection as a URL.
* Your collection probably uses environment variables. To provide an accompanying set of environment variables, [export the template](https://learning.getpostman.com/docs/postman/environments_and_globals/manage_environments/) from Postman and run them with the -e flag.

$ $ newman run https://www.getpostman.com/collections/cb208e7e64056f5294e5 -e dev\_environment.json

* Use the -n option to set the number of iterations to run the collection.

$ newman run mycollection.json -n 10 # runs the collection 10 times

* To provide a different set of data, such as variables for each iteration, you can use the -d to specify a JSON or CSV file.

For example, a data file such as the one shown below runs 2 iterations, with each iteration using a set of variables.

[{

"url": "http://127.0.0.1:5000",

"user\_id": "1",

"id": "1",

"token\_id": "123123",

},

{

"url": "http://postman-echo.com",

"user\_id": "2",

"id": "2",

"token\_id": "899899",

}]

**$ newman run mycollection.json -d data.json**

Here's an example of the CSV file for the above set of variables:

url, user\_id, id, token\_id

http://127.0.0.1:5000, 1, 1, 123123123

http://postman-echo.com, 2, 2, 899899

Newman, by default, exits with a status code of 0 if everything runs well, such as without any exceptions.

Continuous integration tools respond to these exit codes and correspondingly pass or fail a build.

You can use the --bail flag to tell Newman to halt on a test case error with a status code of 1, which can then be picked up by a CI tool or build system.

$ newman run PostmanCollection.json -e environment.json --bail newman

**Importing test results**

The results of all tests and requests can be exported into a file and later imported into Postman for further analysis. Use the JSON reporter and a file name to save the runner output into a file.

$ newman run mycollection.json --reporters cli,json --reporter-json-export outputfile.json

**Note:** Newman allows you to use all [libraries and objects](https://learning.getpostman.com/docs/postman/scripts/postman_sandbox/) that Postman supports to run tests and pre-request scripts.

Library

Newman has been built as a library from the ground up. It can be extended and used in various ways. You can use it as follows in your Node.js code:

var newman = require('newman'); // require newman in your project

// call newman.run to pass `options` object and wait for callback

newman.run({

collection: require('./sample-collection.json'),

reporters: 'cli'

}, function (err) {

if (err) { throw err; }

console.log('collection run complete!');

});

**Integration with Jenkins**

Postman contains a full-featured [testing sandbox](https://learning.getpostman.com/docs/postman/scripts/postman_sandbox/) that lets you write and execute JavaScript based tests for your API. You can then hook up Postman with your build system using [Newman](https://learning.getpostman.com/docs/postman/collection_runs/command_line_integration_with_newman/), the command line collection runner for Postman.

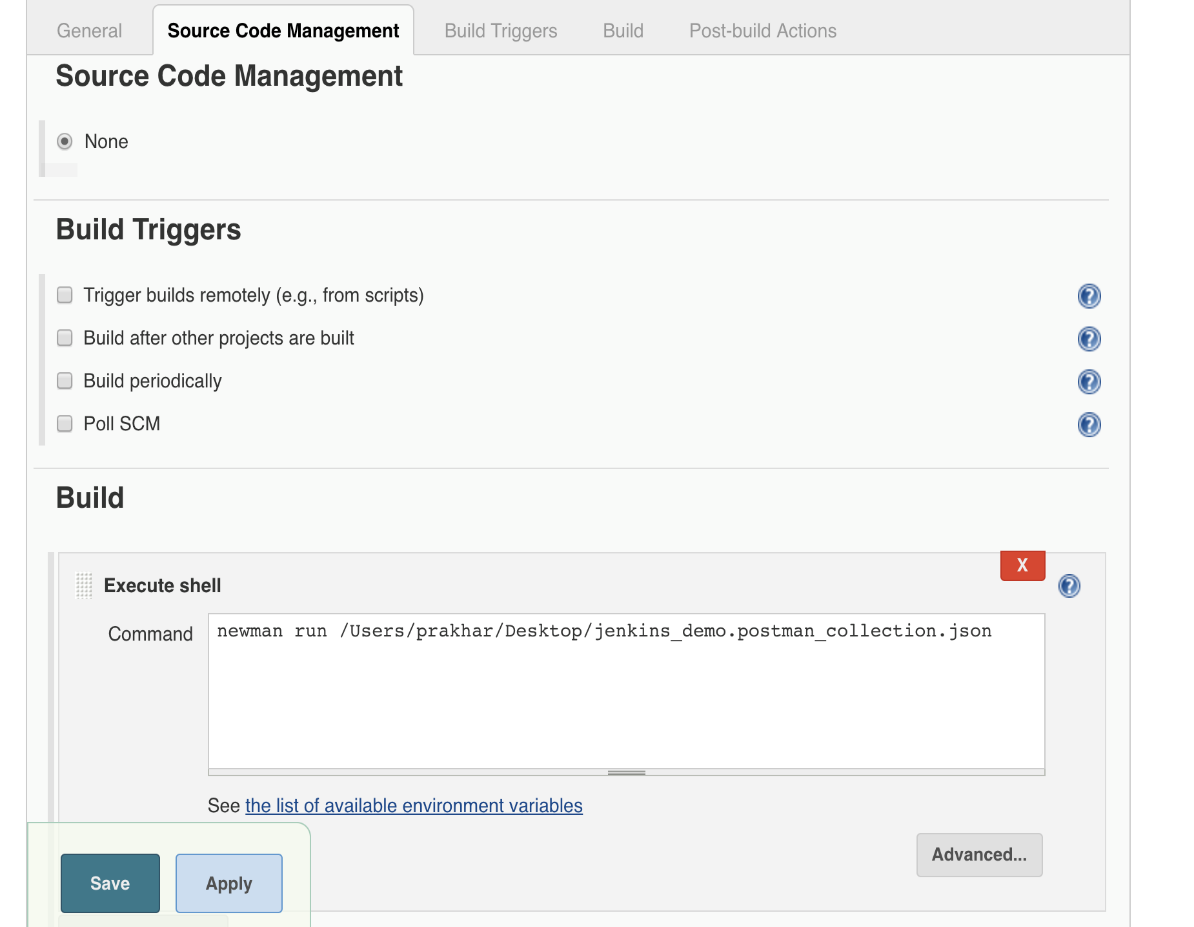
Newman allows you to run and test a Postman Collection. Newman and Jenkins are a perfect match. Let's review these topics to set up this operation.

1. Create a new job by clicking on the “New Item” link on the left sidebar > Select a “Freestyle Project” from the options > Name your project.
2. Add a build step in the project. The build step executes a Shell command.

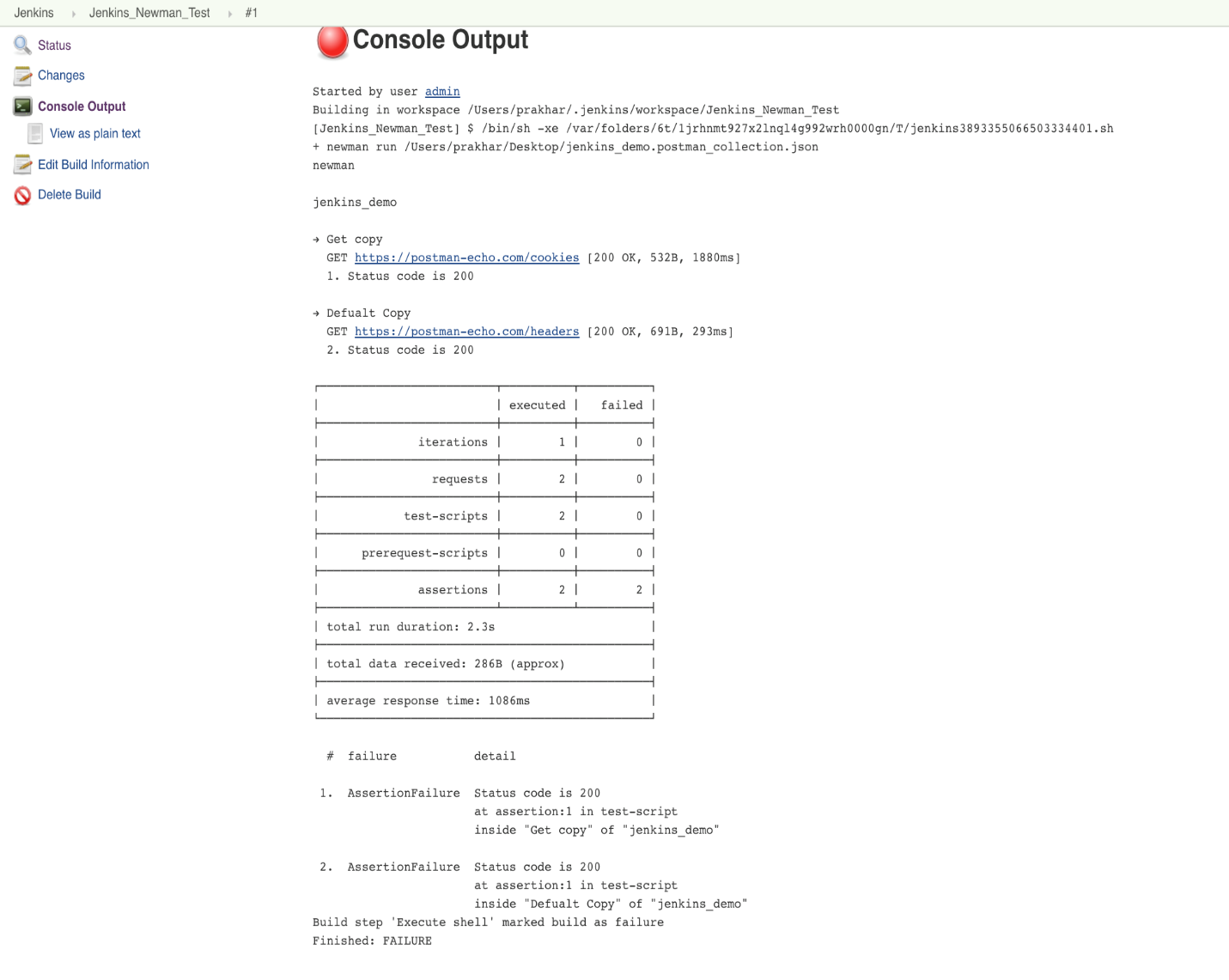
$ newman jenkins\_demo.postman\_collection --exitCode 1

Note here that we are using the Newman command parameter “exitCode” with the value 1. This denotes that Newman is going to exit with this code that will tell Jenkins that everything did not go well.

1. Click the **Save** button to finish creating the project.



1. Run this build test manually by clicking on the “Build Now” link in the sidebar.
2. Jenkins indicates that the build has failed with a red dot in the title. We can check why with the console output from Newman.
3. Click the “Console Output” link in the sidebar to see what Newman returned.



To set up the frequency with which Jenkins runs Newman, click on “Configure project” in the main project window and then scroll down.=. The syntax for setting the frequency is H/(30) \* \* \* \*.

**Note:** 30 can be replaced with another number.

Jenkins will now run Newman at your desired frequency and will tell you whether the build failed or succeeded. In a bigger setup, Newman will be part of your build process and probably not the entire process. You can set up notifications and customize Jenkins as per your needs.