Postman and Crimes of Testing

RESTful Testing and Gating with Postman

VALUE FOR YOUR TIME

- Make, facilitate communication and testing
- Measurement of coverage
- Reduce quantity of bad code getting out
  - Gated Checkins
  - Sonar

NEEDS BEING SOLVED BY POSTMAN

Easy

- GUI based
- Content assists throughout

NEEDS BEING SOLVED BY POSTMAN

Documentation

- APIs defined, both in English and inputs and outputs fully documented
- Can define both happy and sad paths
- Grouping of services into collections
- Sharable

NEEDS BEING SOLVED BY POSTMAN

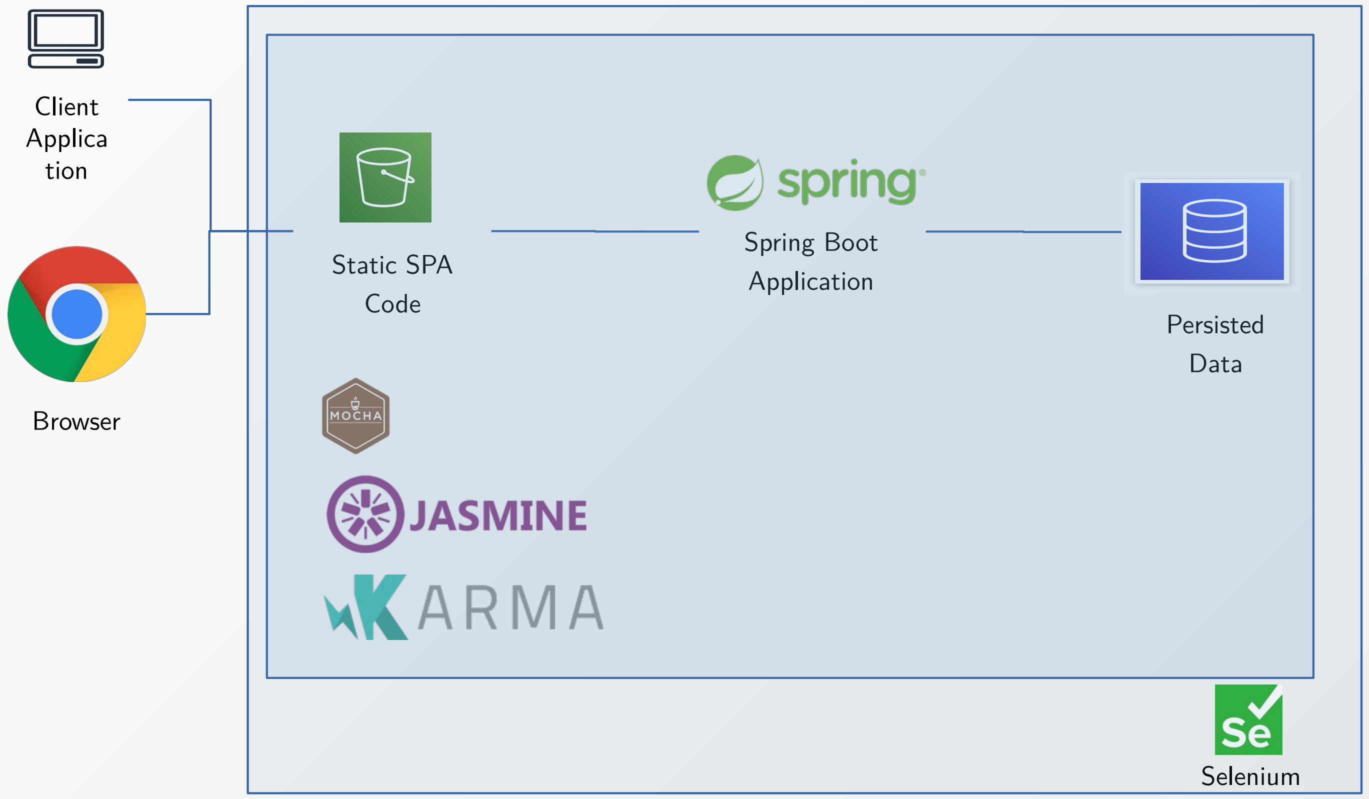
Execution

- Can do quick one-offs, or run collections. For example, add all the APIs for your project to have full, repeatable, regression testing.
- Run the same service in any environment at the change of a drop-down.
- Easily see full parameters, authorization, headers, body and the full response body, cookies and headers. Easily “lint” the results into indented JSON. Save the response to separate file.
- Set up a mock server for UI development.
- Regularly monitor results.

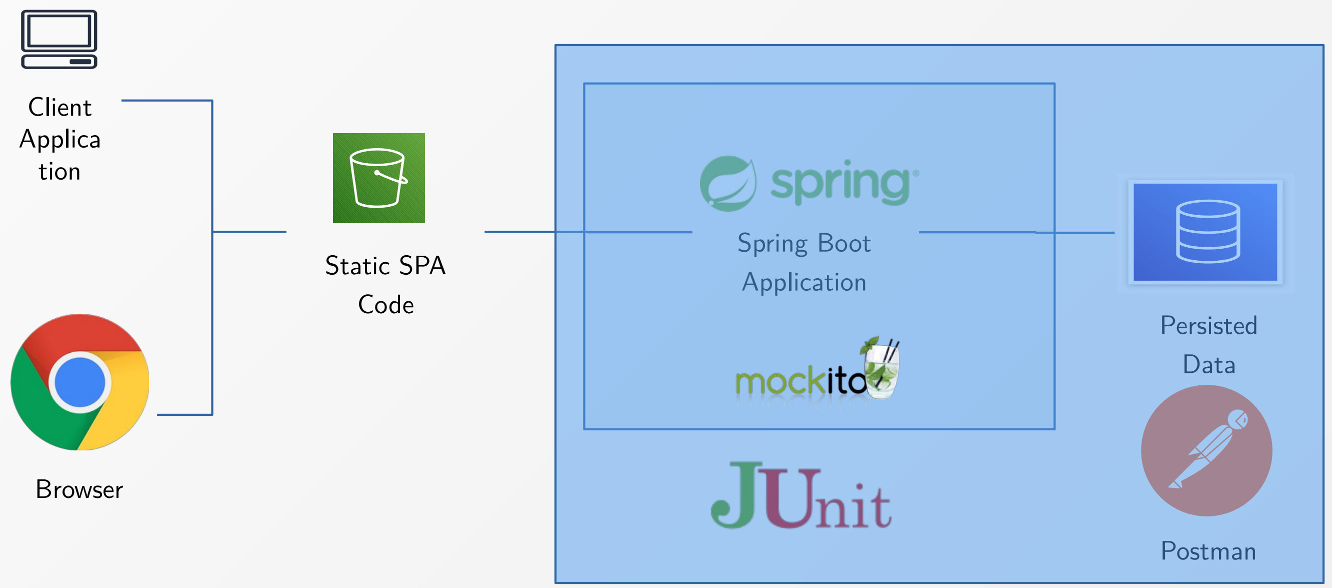
Testing Tool Landscape

RURAL SOURCING

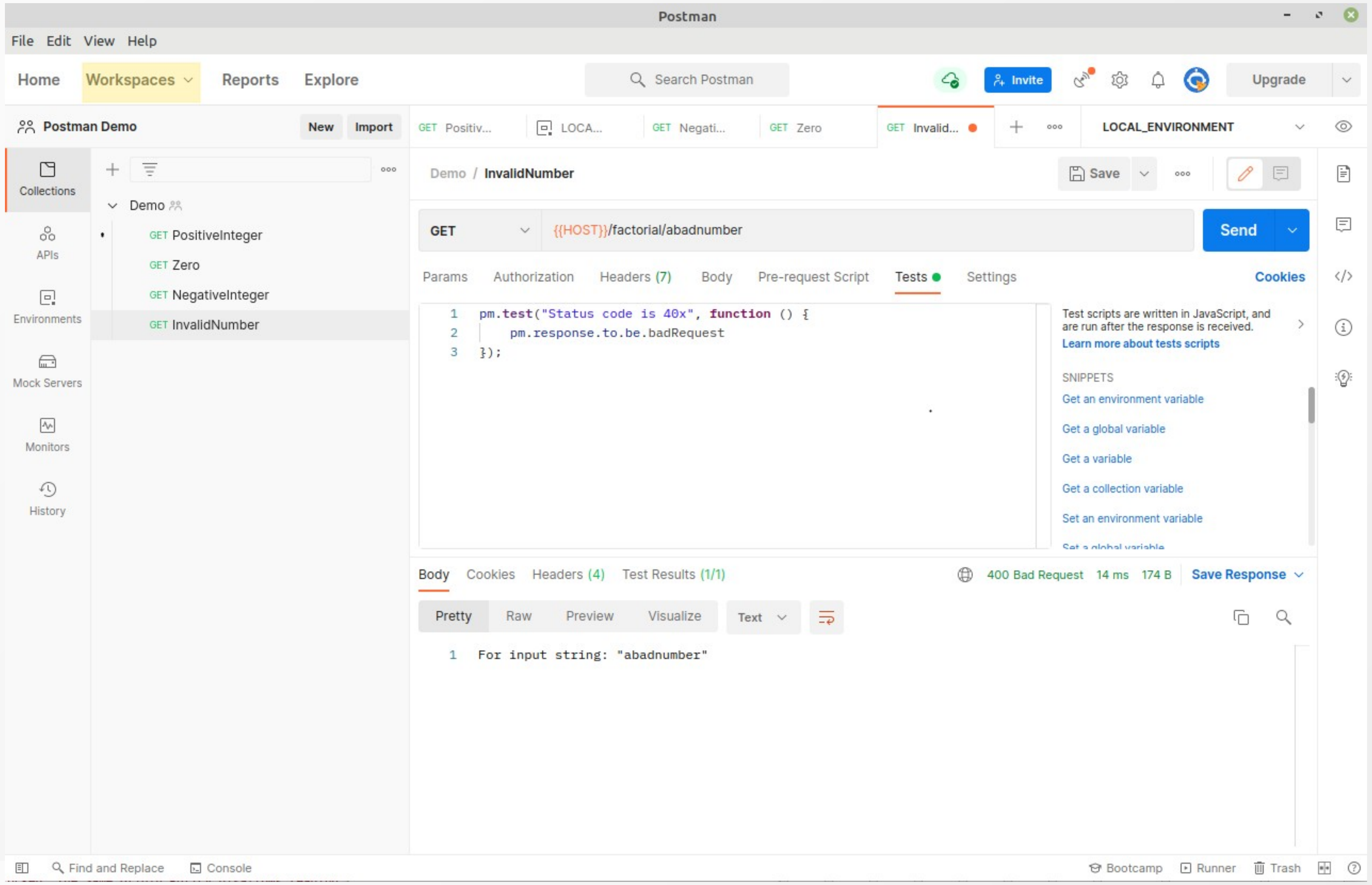
Testing Tool Landscape



Testing Tool Landscape



POSTMAN LAYOUT



GET {{HOST}}/factorial/abadnumber

```
pm.test("Status code is 40x", function () {
    pm.response.to.be.badRequest
});
```

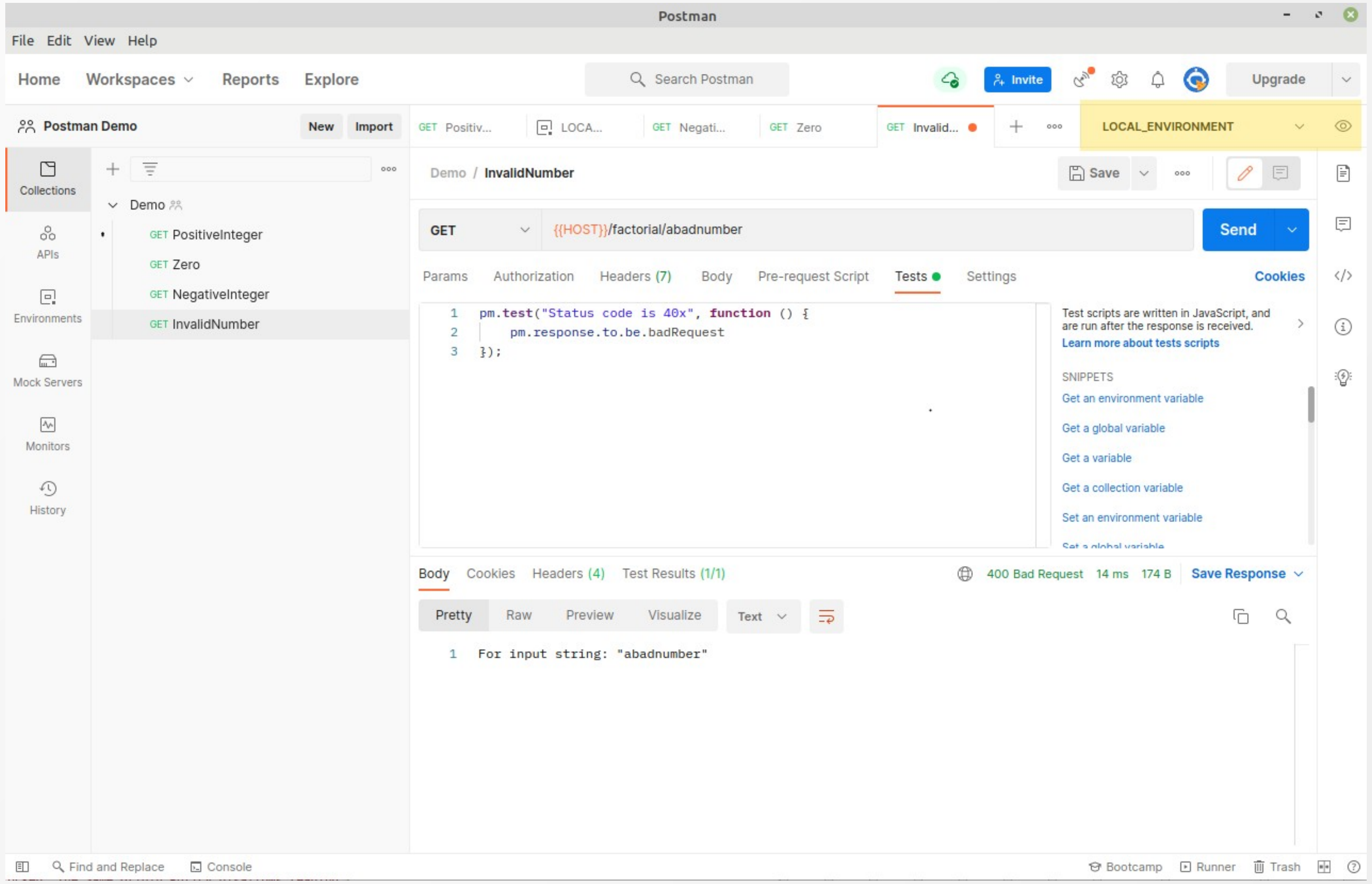
Test scripts are written in JavaScript, and are run after the response is received.
Learn more about tests scripts

400 Bad Request 14 ms 174 B

```
For input string: "abadnumber"
```

RURAL SOURCING

POSTMAN LAYOUT



GET {{HOST}}/factorial/abadnumber

```
pm.test("Status code is 40x", function () {
    pm.response.to.be.badRequest
});
```

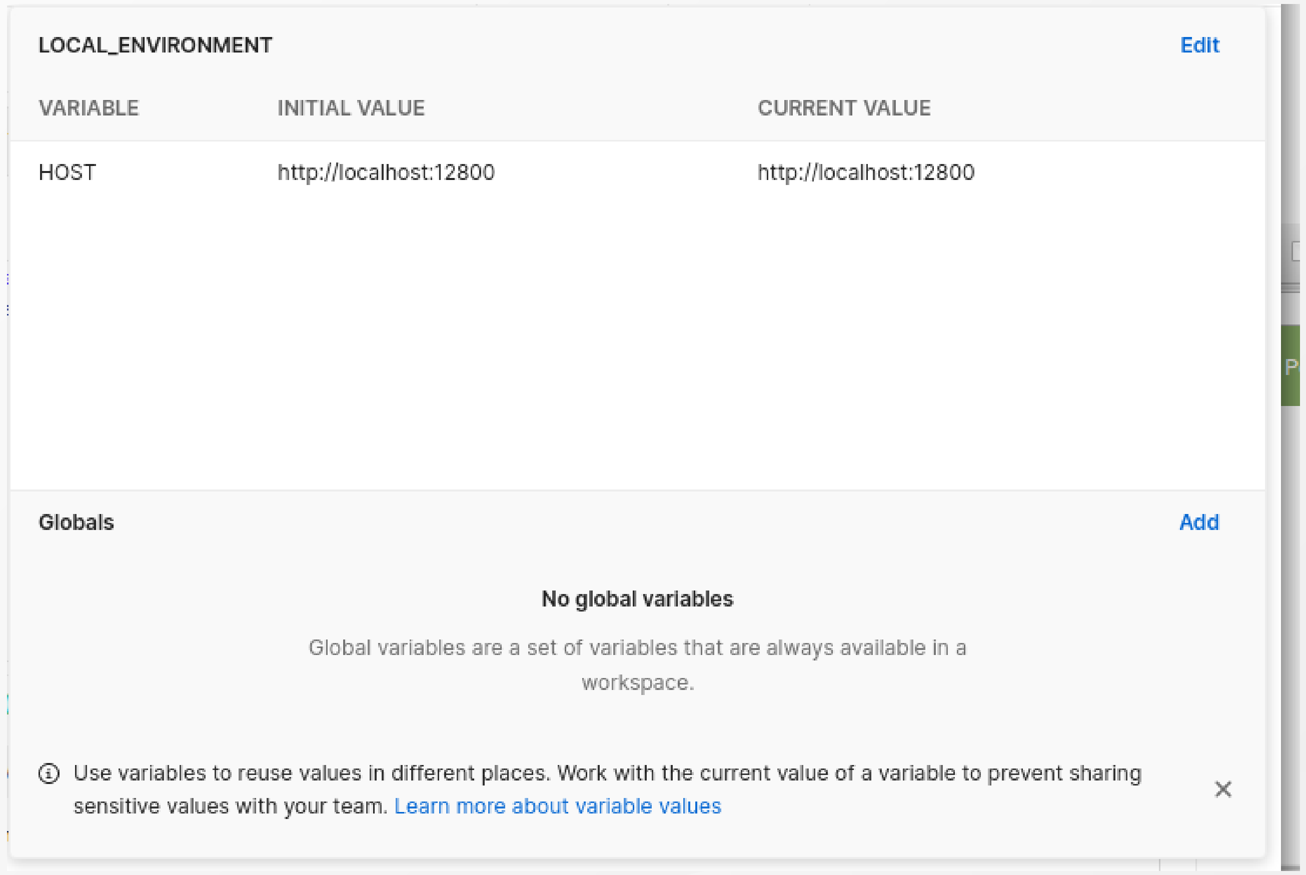
Test scripts are written in JavaScript, and are run after the response is received.
Learn more about tests scripts

400 Bad Request 14 ms 174 B

```
For input string: "abadnumber"
```

RURAL SOURCING

POSTMAN LAYOUT



LOCAL_ENVIRONMENT

VARIABLE	INITIAL VALUE	CURRENT VALUE
HOST	http://localhost:12800	http://localhost:12800

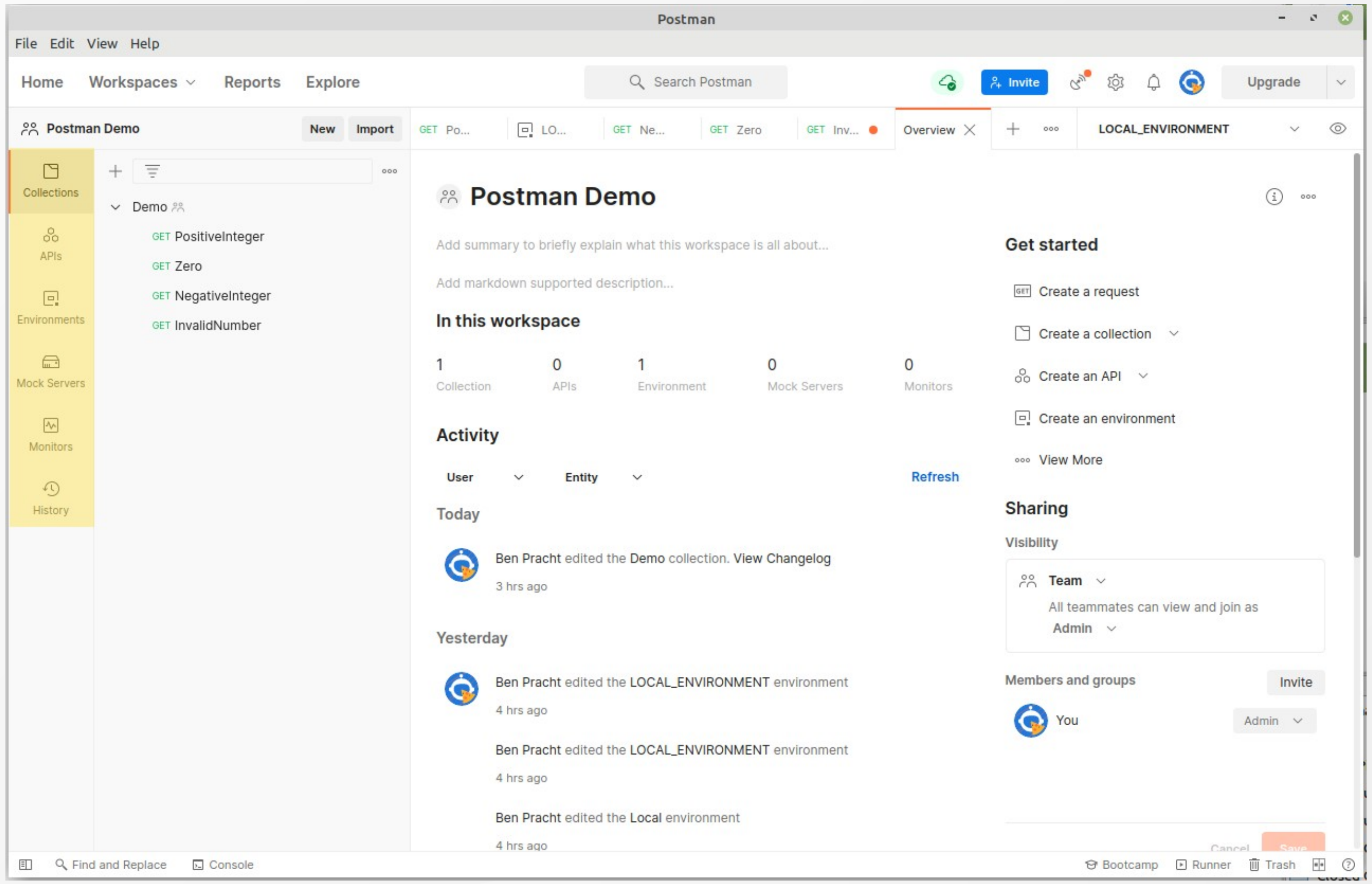
Globals

No global variables

Global variables are a set of variables that are always available in a workspace.

Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values

POSTMAN LAYOUT



Postman Demo

Add summary to briefly explain what this workspace is all about...

Add markdown supported description...

In this workspace

1	0	1	0	0
Collection	APIs	Environment	Mock Servers	Monitors

Activity

User Entity Refresh

Today

Ben Pracht edited the Demo collection. View Changelog
3 hrs ago

Yesterday

Ben Pracht edited the LOCAL_ENVIRONMENT environment
4 hrs ago

Ben Pracht edited the LOCAL_ENVIRONMENT environment
4 hrs ago

Ben Pracht edited the Local environment
4 hrs ago

Get started

- Create a request
- Create a collection
- Create an API
- Create an environment
- View More

Sharing

Visibility

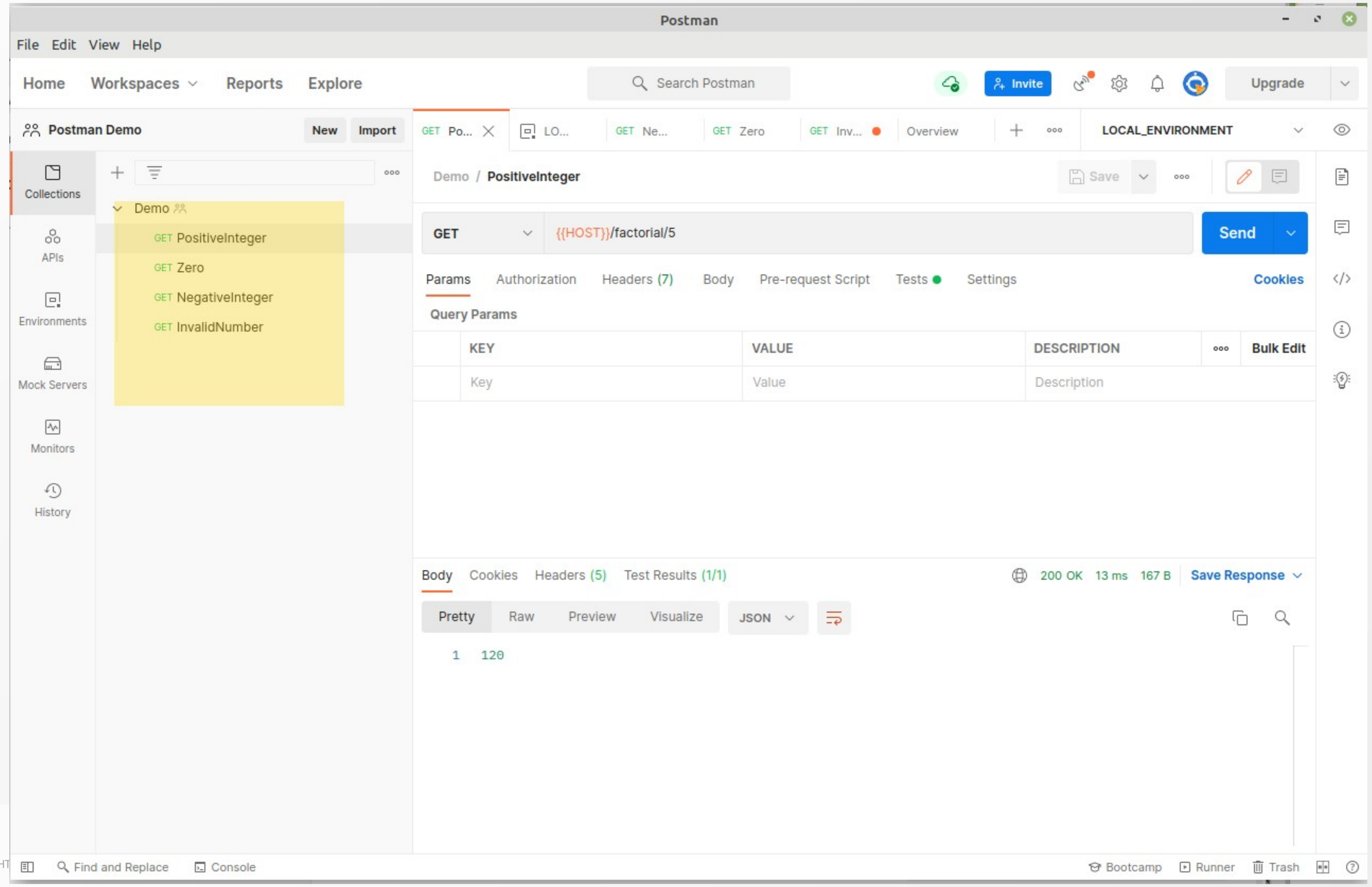
Team
All teammates can view and join as Admin

Members and groups Invite

You Admin

RURAL SOURCING

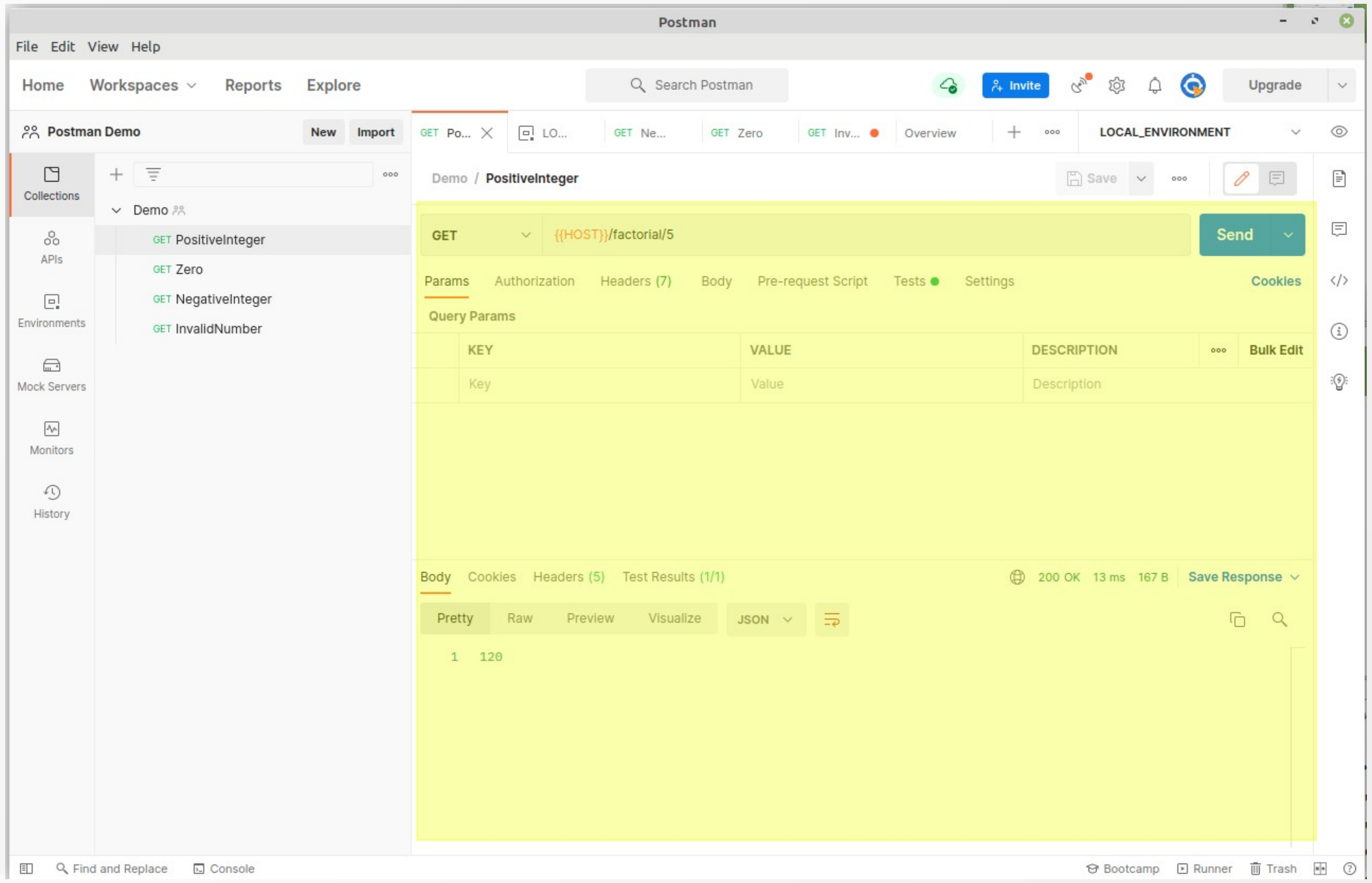
POSTMAN LAYOUT



```
{{HOST}}/factorial/5
```

```
1  120
```

POSTMAN LAYOUT



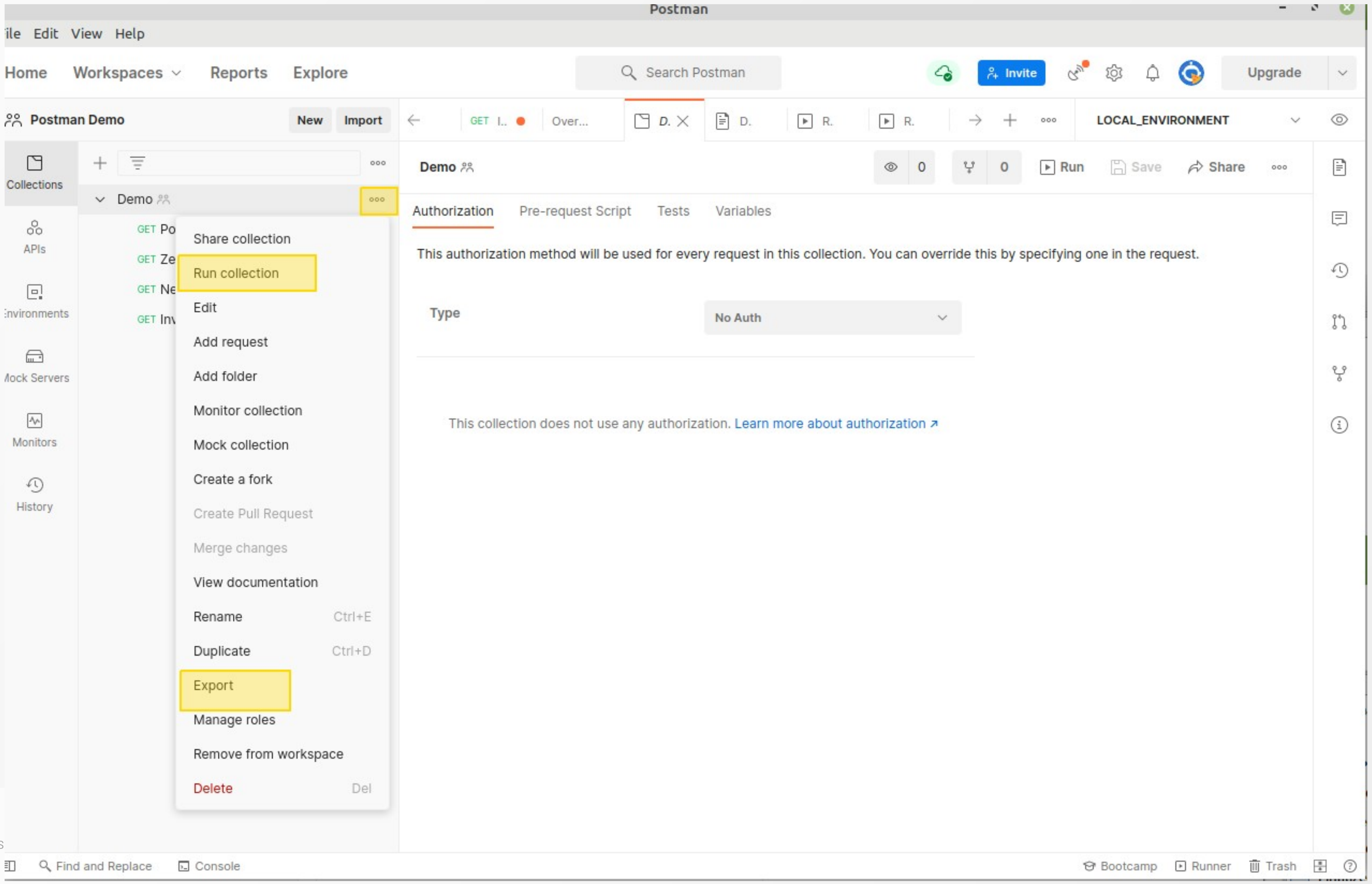
GET {{HOST}}/factorial/5

KEY	VALUE	DESCRIPTION
Key	Value	Description

200 OK 13 ms 167 B

```
120
```

POSTMAN LAYOUT

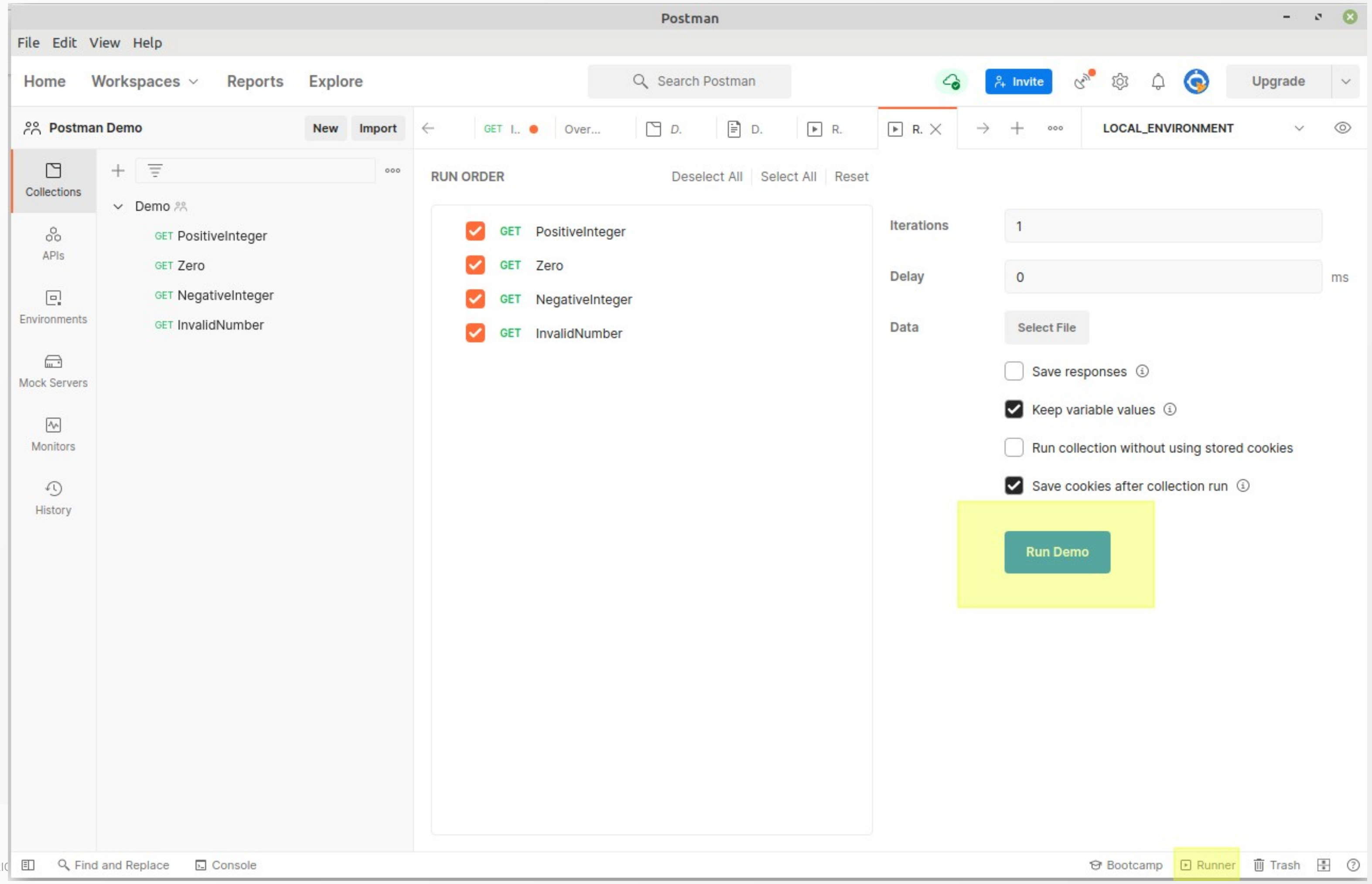


This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

Type: No Auth

This collection does not use any authorization. Learn more about authorization ↗

POSTMAN LAYOUT



RUN ORDER — Deselect All | Select All | Reset

- [x] GET PositiveInteger
- [x] GET Zero
- [x] GET NegativeInteger
- [x] GET InvalidNumber

Iterations: 1

Delay: 0 ms

Data: Select File

- [ ] Save responses
- [x] Keep variable values
- [ ] Run collection without using stored cookies
- [x] Save cookies after collection run

Run Demo

RURAL SOURCING

POSTMAN DO’S AND DONT’S

Do define every service in your collection. At least define the expected normal path.

Do use variables to “genericize” the service. Don’t make one request for dev and another for test. Instead, shift the differences to an environment variable.

Do not assume the service follows standard RESTful API conventions, especially regarding returned status codes. Many shops do not, while some mostly follow them but deviate slightly.

Do document the purpose of the service.

Do persist the collection. Postman offers Git sync capability. Please see https://blog.postman.com/better-practices-for-git-version-control-in-postman/
for more information.

BRIEF INCANTATION FOR QUALITY GATING

Using open source Github repository. You’ll likely have a different Git host, but the ideas will be similar. This was easier and quicker than standing up a Jenkins server, a SonarCube server and a git repository.

1 Created a repository with my Java code

2 Add Github workflow actions to automate my Maven build and Sonar Scan

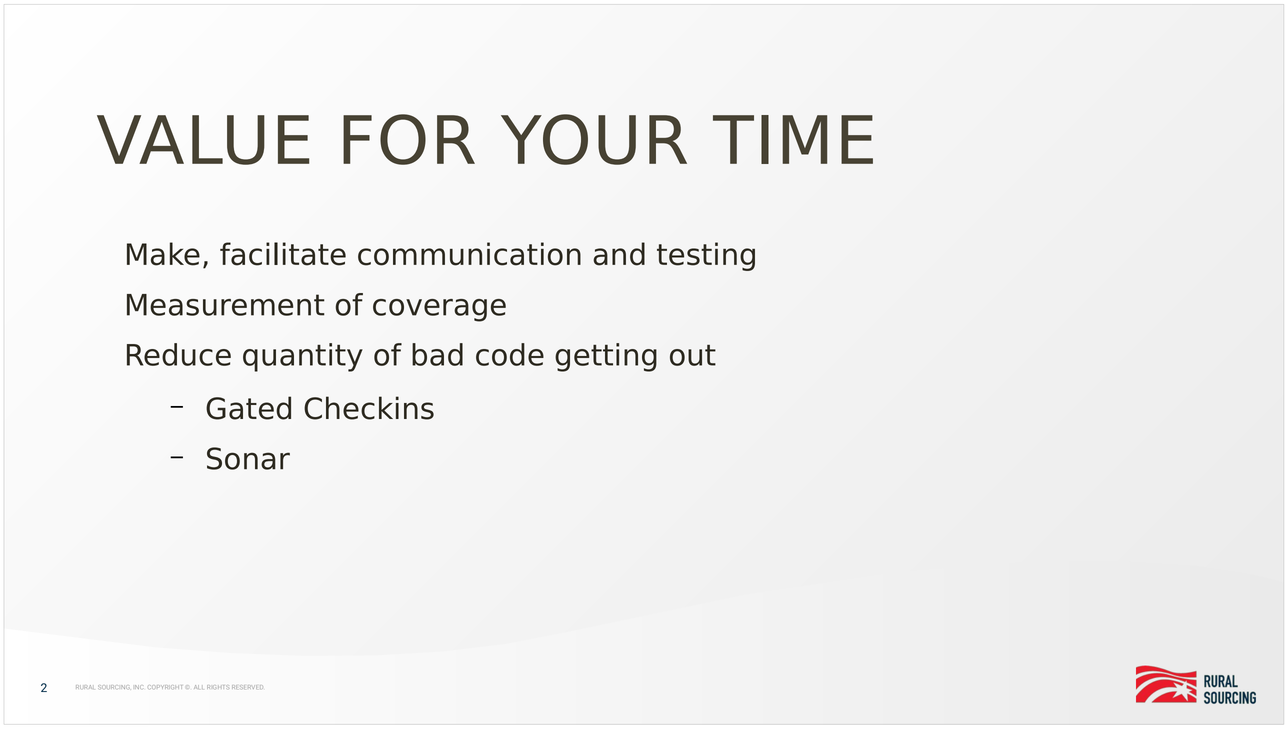
3 Block bad test results from merge

RURAL SOURCING

HAPPY TESTING!

Postman and Crimes of Testing

RESTful Testing and Gating with Postman



VALUE FOR YOUR TIME

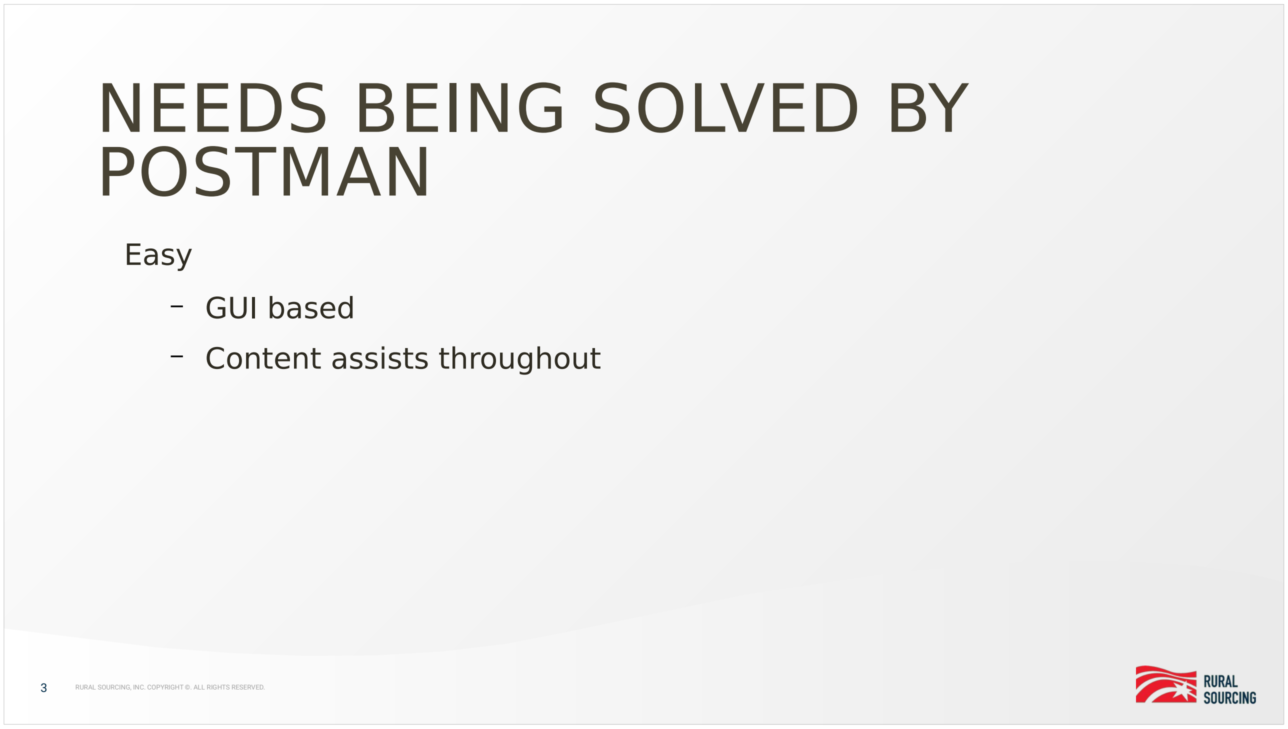
Make, facilitate communication and testing

Measurement of coverage

Reduce quantity of bad code getting out

- Gated Checkins
- Sonar

Justify the use of your time listening to me ramble by sharing benefits of specific usage of a specific tool and on preventing bad code from getting out into the wild



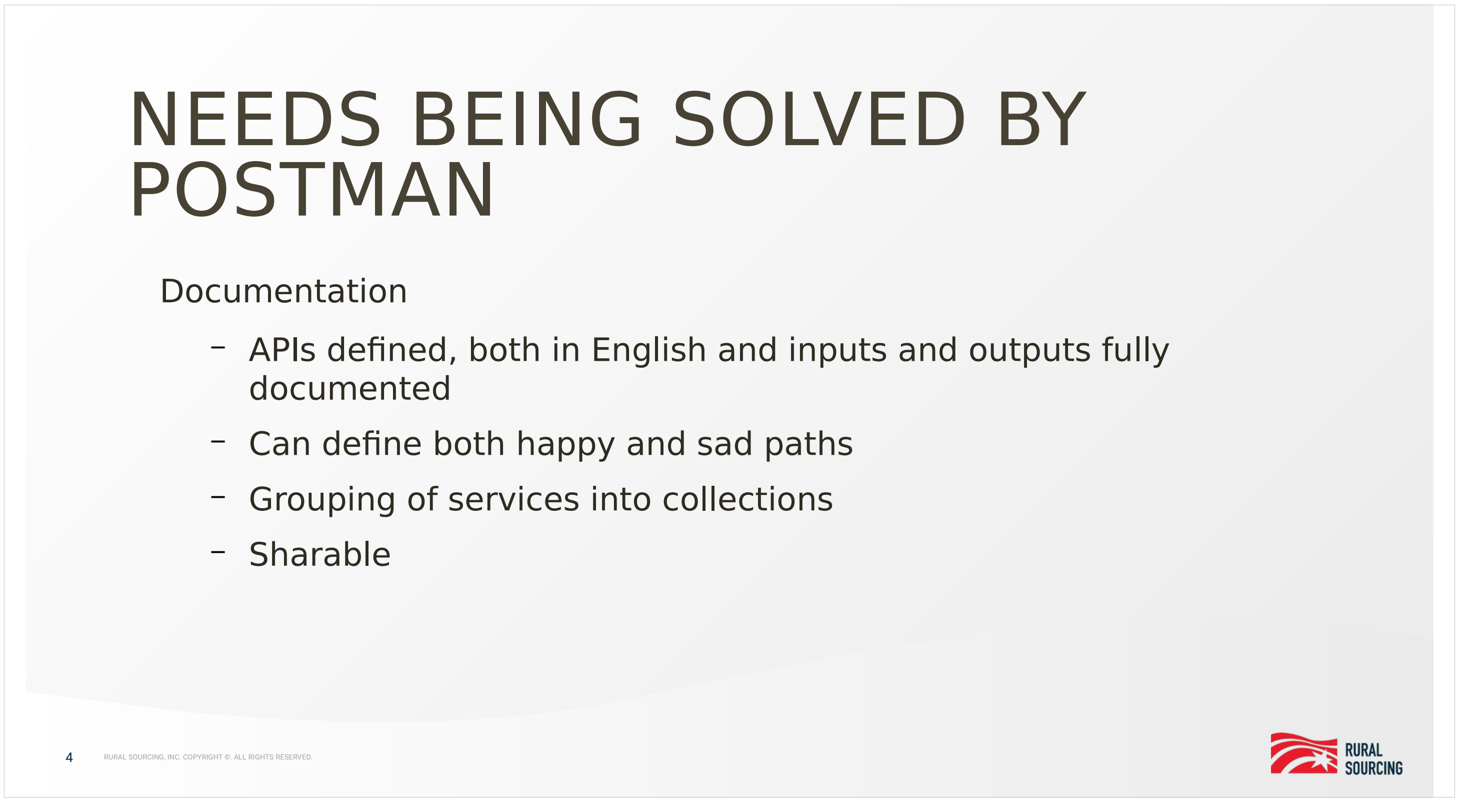
NEEDS BEING SOLVED BY POSTMAN

Easy

- GUI based
- Content assists throughout

Postman is one of many GUI based web service clients. While there are other options, this client has a significant marketshare, making the file format somewhat portable among other utilities.

Postman goes through appreciable links to be friendly. Content assist features are present throughout and done intelligently. For example, Content-Type headers have appropriate selections. There's code assist for testing the validity of a response.



NEEDS BEING SOLVED BY POSTMAN

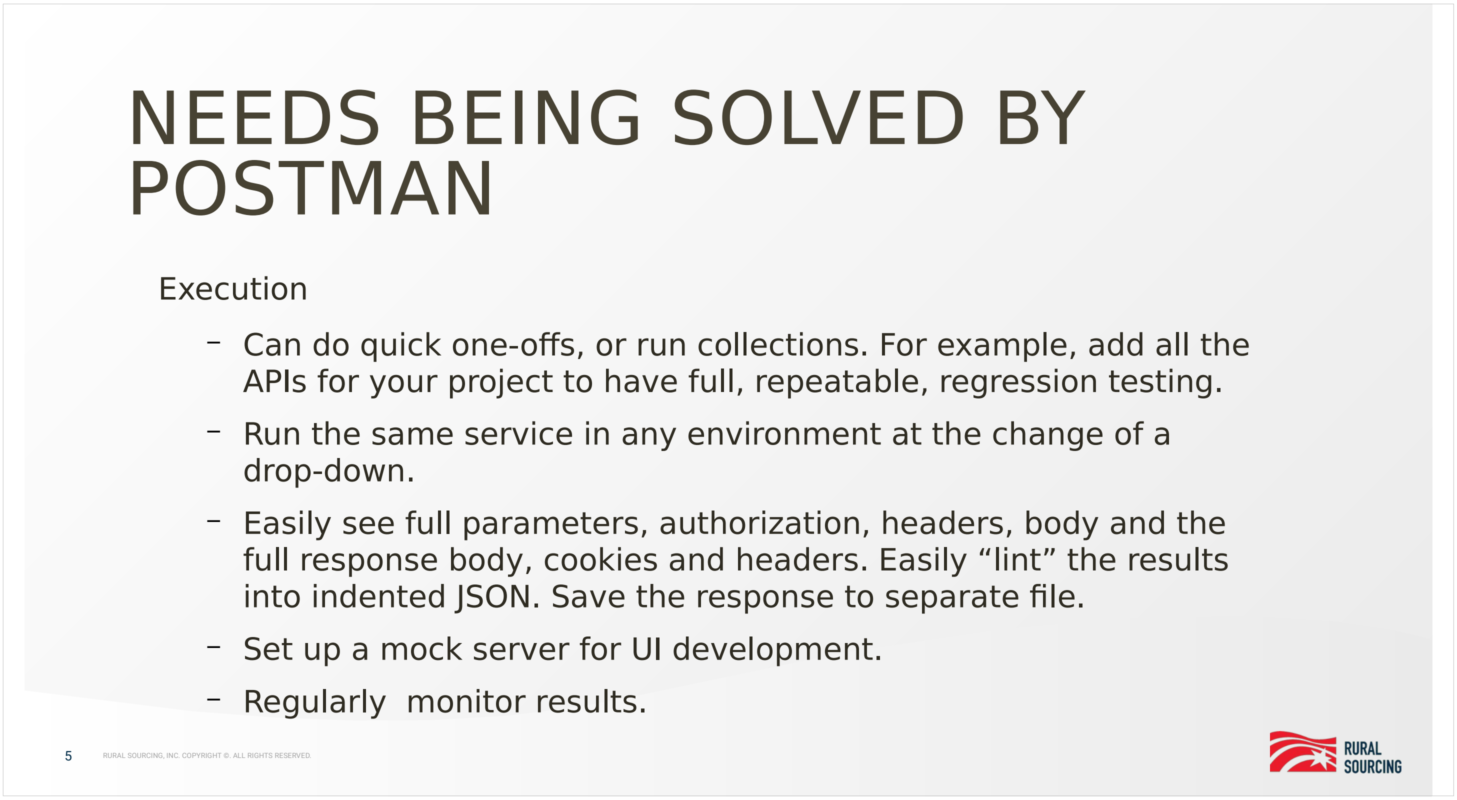
Documentation

- APIs defined, both in English and inputs and outputs fully documented
- Can define both happy and sad paths
- Grouping of services into collections
- Sharable

In any project good communication is key. Postman allows an always active, always current documentation of the backend interface. Changes to the Postman files can be immediately uploaded to a workspace and coded against. They can also be saved in a Git repository and later imported upon checkout, making it versioned.

Postman in particular, and for a fee, provides a team interface. People on your team can be invited to join a workspace and get the latest version, sparing people a download from a browser, Git synchronization, email, etc.

The expected behavior is actively defined, especially the happy paths. The project errors, as best as they're tested, are also defined here. Little things like HTTP status codes get clearly deliniated—an important thing because not every project is consistent.



NEEDS BEING SOLVED BY POSTMAN

Execution

- Can do quick one-offs, or run collections. For example, add all the APIs for your project to have full, repeatable, regression testing.
- Run the same service in any environment at the change of a drop-down.
- Easily see full parameters, authorization, headers, body and the full response body, cookies and headers. Easily “lint” the results into indented JSON. Save the response to separate file.
- Set up a mock server for UI development.
- Regularly monitor results.

5

With only a few clicks, a request can be created and tested and see the full results. It’s possible to define a request via curl command (or export a request to curl from Postman).

More commonly, a single request can be executed, or a full collection can be executed. This allows a manual regression test, or it could be inserted into your CI/CD pipeline.

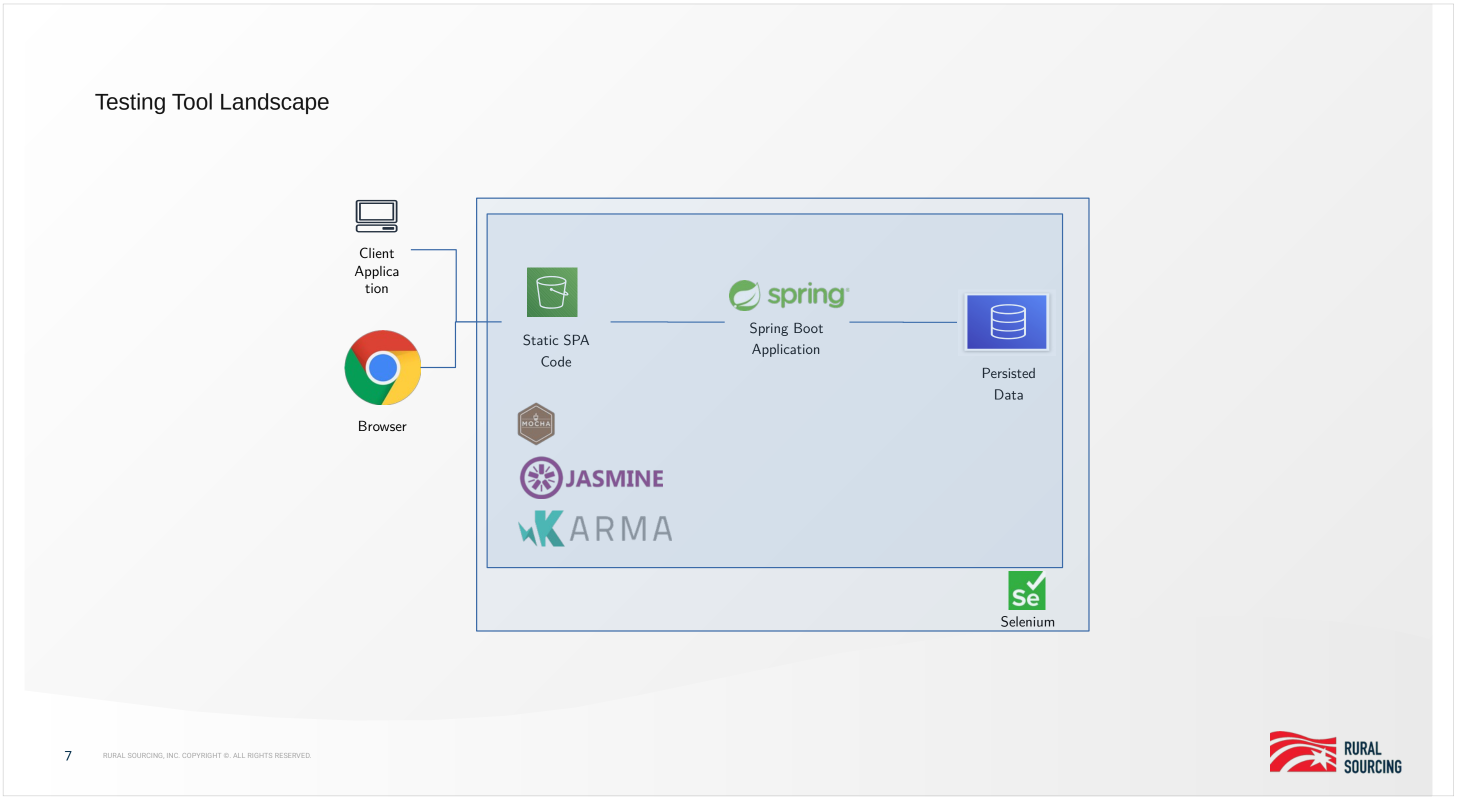
Postman is aware of your environments. Each environments can be defined as needed, such as local, test/qa, production, etc, and environment specific variables can be included in your request. For example, define HOST as https://test.somecompany.com for your TEST environment. Authorization can be set up to be fetched, either through a standard OATH2 sequence or via a user’s browser capture. So after a deployment to test, you could select the TEST environment, then run the services for a TEST regression.

Additionally, a mock server can be defined so that the UI people have a server to target while the back-end people are still working on the implementation. Finally, it can be set up for routine monitoring.

Testing Tool Landscape

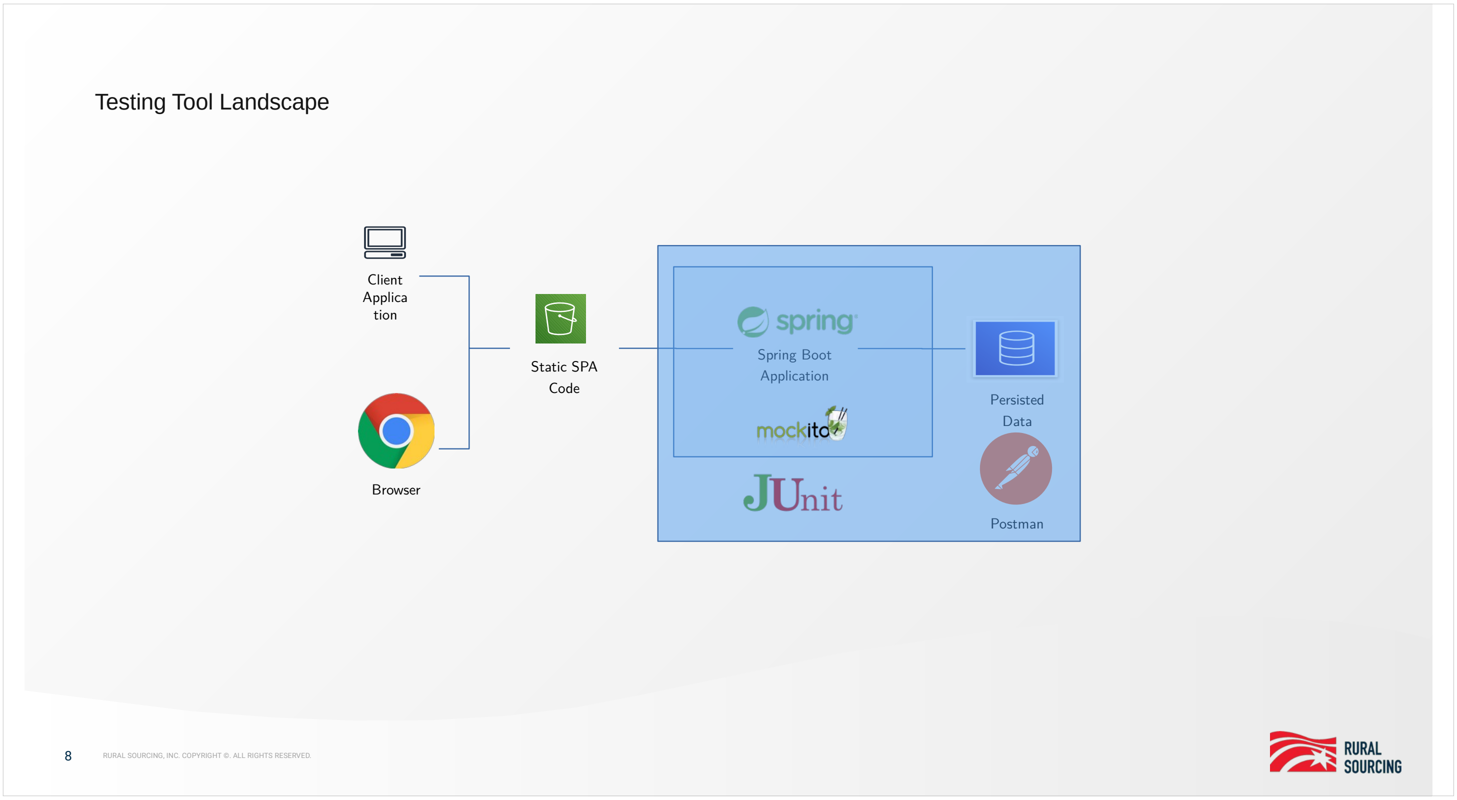
RURAL SOURCING

Before we get too deep, I wanted to clarify the scope of the testing. Here I'm showing a heavily and drastically simplified corporate stack that's somewhat commonly found. Reading from left to right, we start with whatever the user sees, be it an actual person or some other client application, and whatever form the UI takes. If there is a visual UI such as a React, Angular or other SPA application, One or more testing frameworks could be used to assure quality at the UI level. Focusing on the Java side, we're gladly avoiding

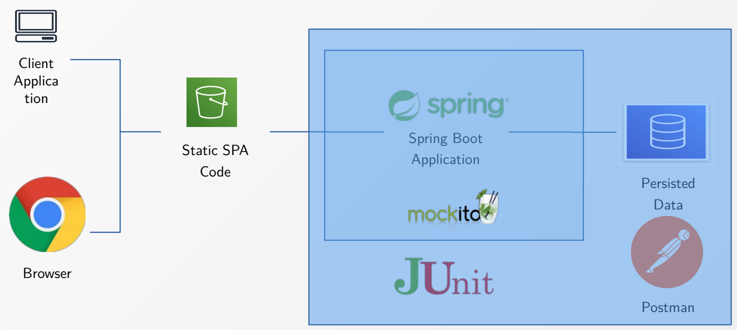


Testing Tool Landscape

Before we get too deep, I wanted to clarify the scope of the testing. Here I'm showing a heavily and drastically simplified corporate stack that's somewhat commonly found. Reading from left to right, we start with whatever the user sees, be it an actual person or some other client application, and whatever form the UI takes. If there is a visual UI such as a React, Angular or other SPA application, One or more testing frameworks could be used to assure quality at the UI level. Focusing on the Java side, we're gladly avoiding



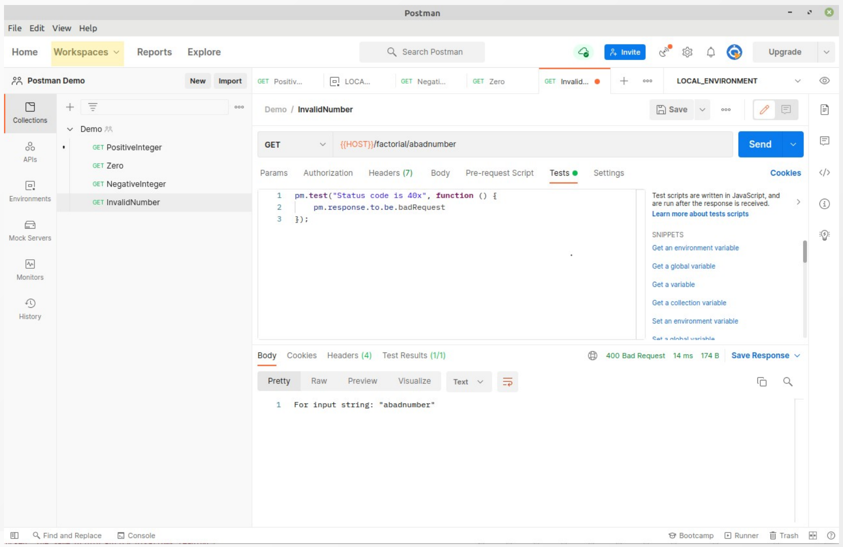
Testing Tool Landscape



RURAL SOURCING

Again from left to right, many of us Java people will be more focused on the backend. As such, we typically focus on quality unit testing as done by Junit. To help isolate the testing, we'll also be mocking things not in test using a mocking framework such as Mockito or maybe PowerMock. However, much of our RESTful testing can begin with Postman.

POSTMAN LAYOUT



GET {{HOST}}/factorial/abadnumber

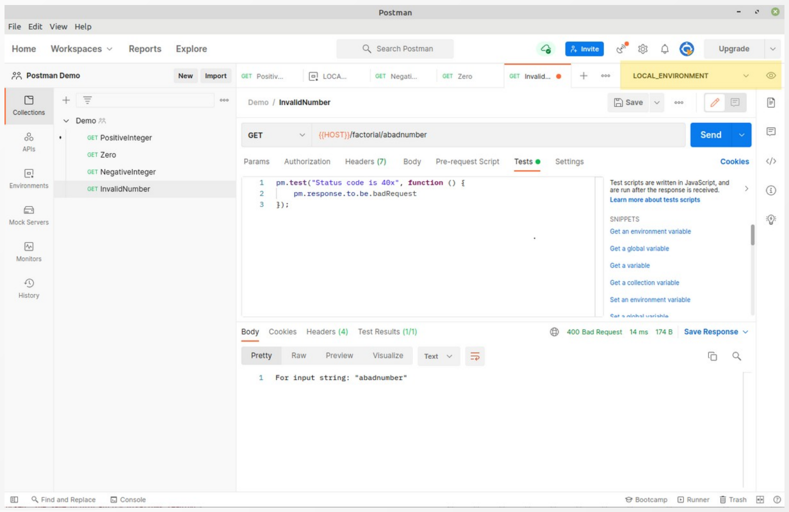
```
pm.test("Status code is 40x", function () {
    pm.response.to.be.badRequest
});
```

Test scripts are written in JavaScript, and are run after the response is received.
Learn more about tests scripts

```
For input string: "abadnumber"
```

A workspace is akin to a project in an IDE like Eclipse or IntelliJ.

POSTMAN LAYOUT



GET {{HOST}}/factorial/abadnumber

```
pm.test("Status code is 40x", function () {
    pm.response.to.be.badRequest
});
```

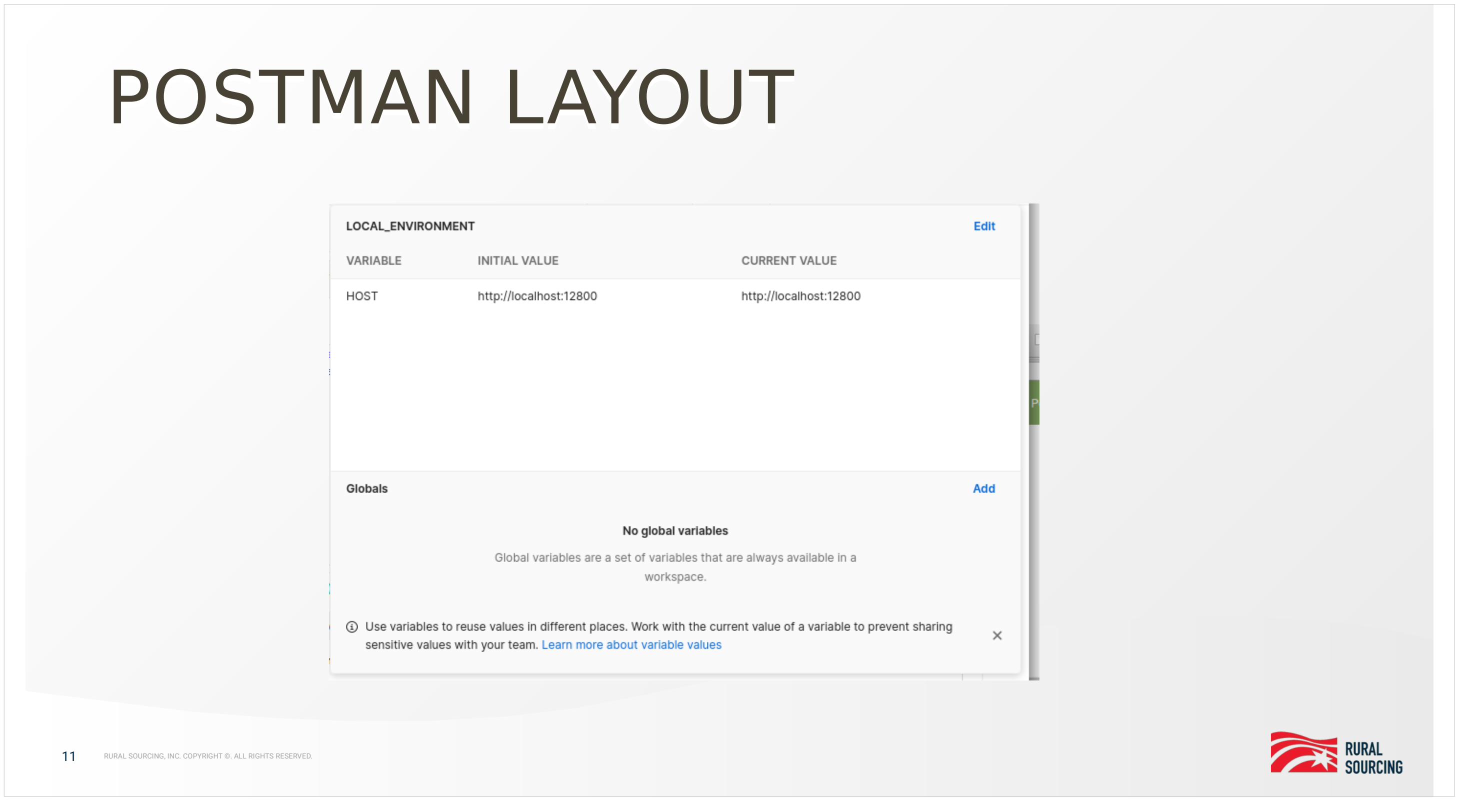
Test scripts are written in JavaScript, and are run after the response is received.
Learn more about tests scripts

400 Bad Request 14 ms 174 B

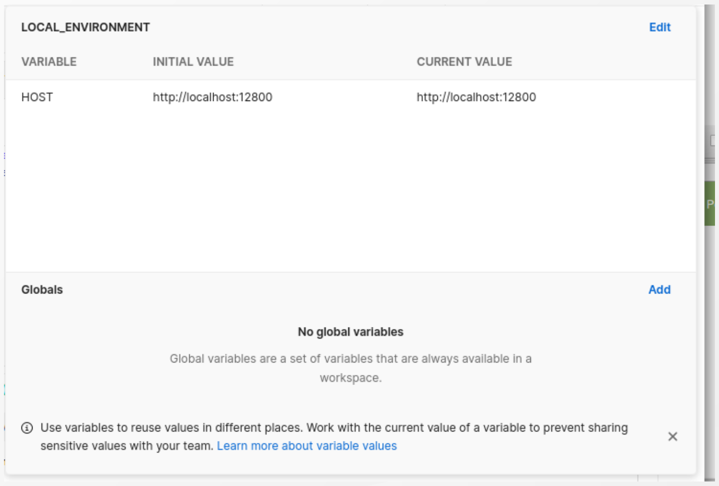
```
For input string: "abadnumber"
```

RURAL SOURCING

This is the environment selector drop down and the place to change specific environment settings.



POSTMAN LAYOUT



LOCAL_ENVIRONMENT — Edit

VARIABLE	INITIAL VALUE	CURRENT VALUE
HOST	http://localhost:12800	http://localhost:12800

Globals — Add

No global variables

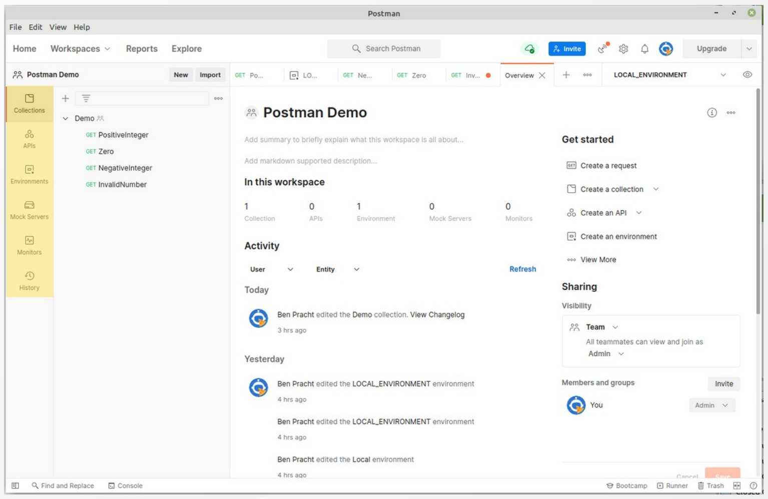
Global variables are a set of variables that are always available in a workspace.

Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values

RURAL SOURCING

This is the environment selector drop down and the place to change specific environment settings. Surround the variable with {{ and }}, eg. use {{VARIABLE_NAME}} anywhere in your request. Global and system variables are usable as well.

POSTMAN LAYOUT



Postman Demo

Add summary to briefly explain what this workspace is all about...

Add markdown supported description...

In this workspace

1	0	1	0	0
Collection	APIs	Environment	Mock Servers	Monitors

Activity

User Entity Refresh

Today

Ben Pracht edited the Demo collection. View Changelog
3 hrs ago

Yesterday

Ben Pracht edited the LOCAL_ENVIRONMENT environment
4 hrs ago

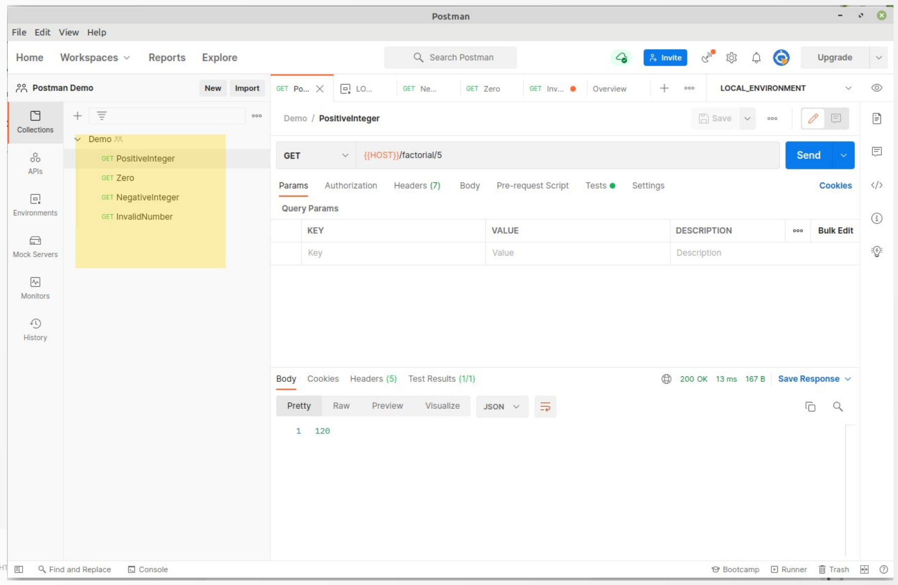
Ben Pracht edited the LOCAL_ENVIRONMENT environment
4 hrs ago

Ben Pracht edited the Local environment
4 hrs ago

All teammates can view and join as Admin

This is the environment selector drop down and the place to change specific environment settings. Surround the variable with {{ and }}, eg. use {{VARIABLE_NAME}} anywhere in your request. Global and system variables are usable as well.

POSTMAN LAYOUT



LOCAL_ENVIRONMENT

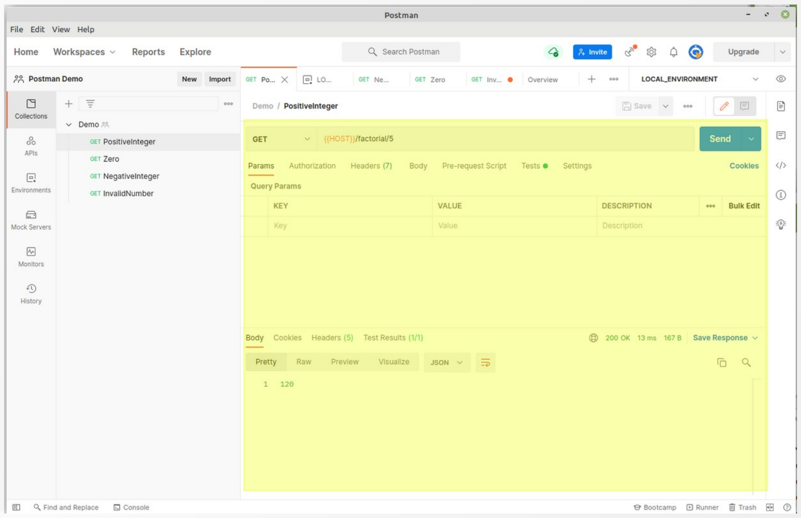
GET {{HOST}}/factorial/5

200 OK 13 ms 167 B

```
1  120
```

This is the environment selector drop down and the place to change specific environment settings. Surround the variable with {{ and }}, eg. use {{VARIABLE_NAME}} anywhere in your request. Global and system variables are usable as well.

POSTMAN LAYOUT



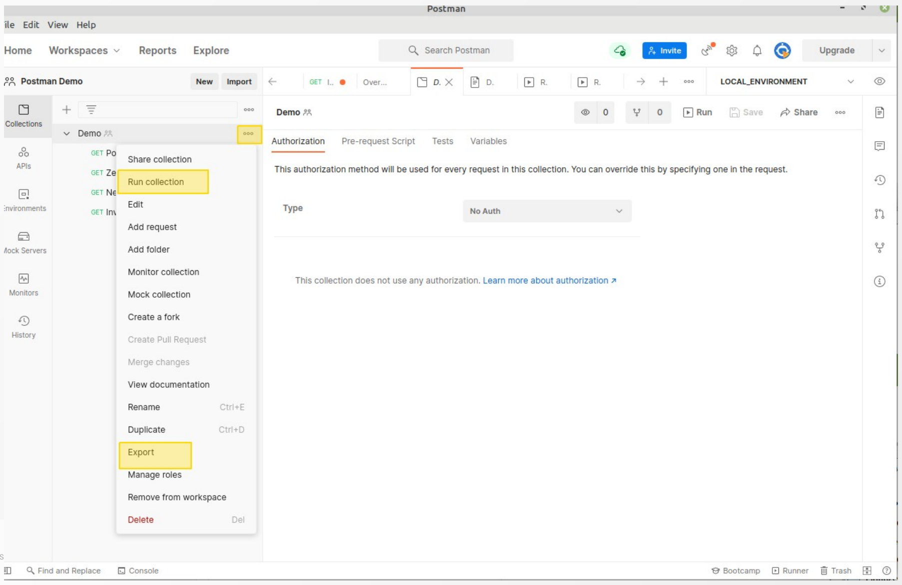
GET {{HOST}}/factorial/5

200 OK 13 ms 167 B

```
1  120
```

This is the environment selector drop down and the place to change specific environment settings. Surround the variable with {{ and }}, eg. use {{VARIABLE_NAME}} anywhere in your request. Global and system variables are usable as well.

POSTMAN LAYOUT

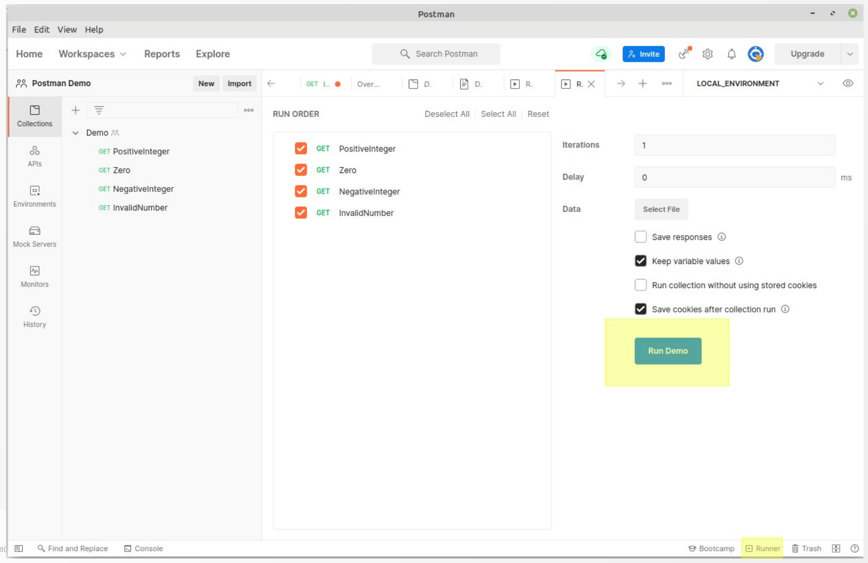


This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

This collection does not use any authorization. Learn more about authorization ↗

This is the environment selector drop down and the place to change specific environment settings. Surround the variable with {{ and }}, eg. use {{VARIABLE_NAME}} anywhere in your request. Global and system variables are usable as well.

POSTMAN LAYOUT



RUN ORDER Deselect All | Select All | Reset

- [✓] GET PositiveInteger
- [✓] GET Zero
- [✓] GET NegativeInteger
- [✓] GET InvalidNumber

Iterations	1
Delay	0 ms
Data	Select File

- [ ] Save responses
- [✓] Keep variable values
- [ ] Run collection without using stored cookies
- [✓] Save cookies after collection run

This is the environment selector drop down and the place to change specific environment settings. Surround the variable with {{ and }}, eg. use {{VARIABLE_NAME}} anywhere in your request. Global and system variables are usable as well.

POSTMAN DO’S AND DONT’S

Do define every service in your collection. At least define the expected normal path.

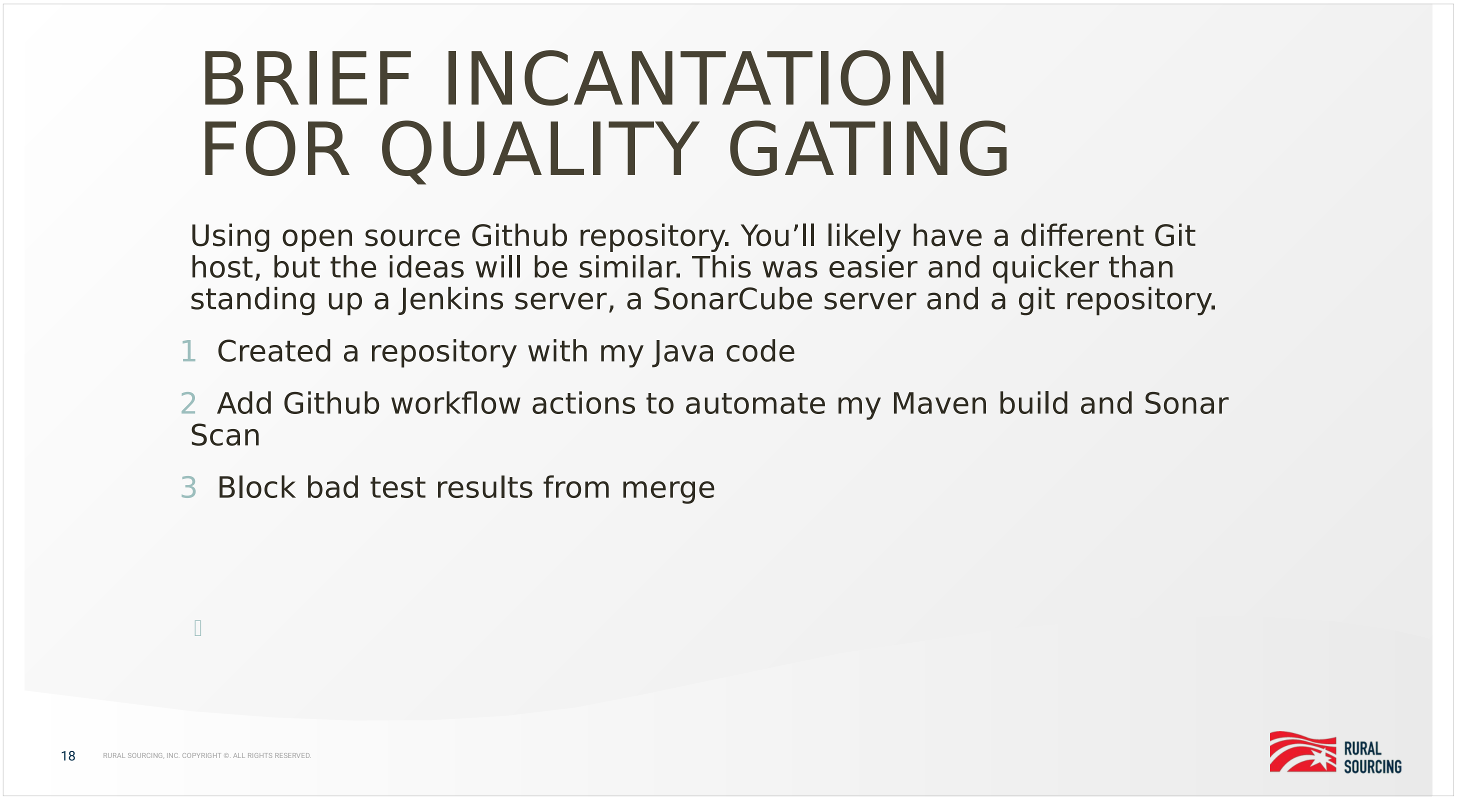
Do use variables to “genericize” the service. Don’t make one request for dev and another for test. Instead, shift the differences to an environment variable.

Do not assume the service follows standard RESTful API conventions, especially regarding returned status codes. Many shops do not, while some mostly follow them but deviate slightly.

Do document the purpose of the service.

Do persist the collection. Postman offers Git sync capability. Please see https://blog.postman.com/better-practices-for-git-version-control-in-postman/
for more information.

RURAL SOURCING



BRIEF INCANTATION FOR QUALITY GATING

Using open source Github repository. You'll likely have a different Git host, but the ideas will be similar. This was easier and quicker than standing up a Jenkins server, a SonarCube server and a git repository.

1. Created a repository with my Java code
2. Add Github workflow actions to automate my Maven build and Sonar Scan
3. Block bad test results from merge

Here, I created a demo project which provides a factorial service as a RESTful endpoint. It's built using Maven .

However for ease and convenience, I put this on a private Github repository and set up GitHub workflow actions. If you're using GHE or Bitbucket, there are parallel workflow capabilities you could use.

HAPPY TESTING!