
# Automation Testing

# Environments

# A Day in the Life of an Automation Testing Engineer

John is now assigned a task to create an Environment and validate the REST requests in Postman.

After completing this session, John will be able to create a different Environment to be used across requests and perform REST request validations in Postman.

# Learning Objectives

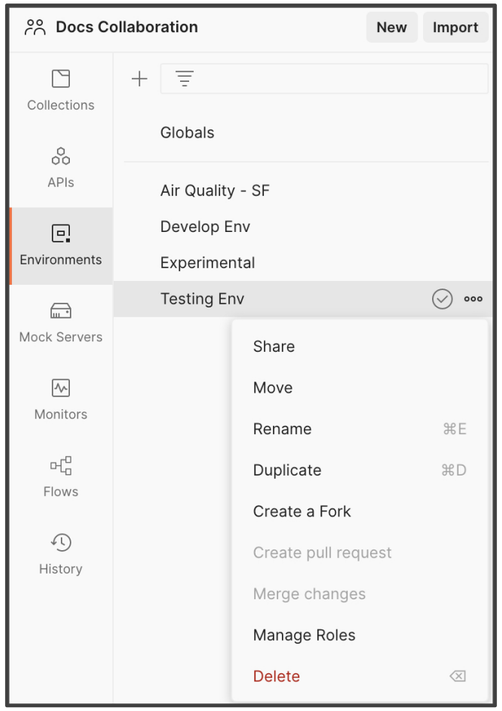
By the end of this lesson, you will be able to:

- Identify an Environment in Postman
- State the usage of an Environment
- Explain the advance validation on response in REST Assured
- List the steps to perform end-to-end case automation

# What Is an Environment?

# What Is an Environment?

An Environment is a set of variables users can use in their Postman requests. Users can use an Environment to group related sets of values.




They can also manage access to the shared Postman data.

# Features of an Environment

Environments allow users to control the visibility of sensitive data, such as API secrets, passwords, tokens, and keys and work on shared resources.

Any global variables they add or modify in a shared workspace will be accessible to everyone else in the workspace.

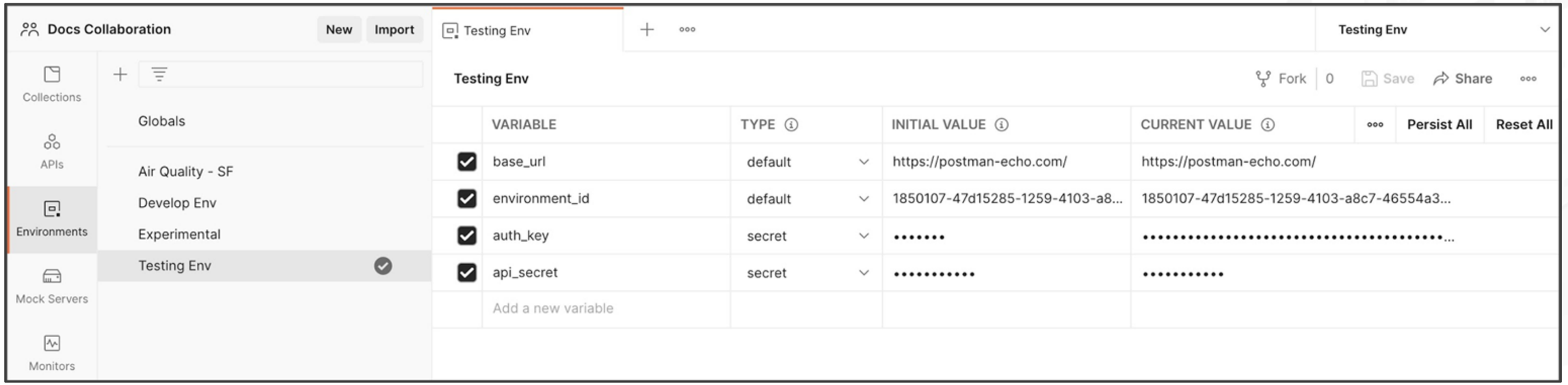
Variables can be defined as secret types to hide the initial and current values for every workspace member and avoid unintended disclosure of sensitive information.

Users can move Environments into shared workspaces to work with the team.

# Usage of Environments

# Usage of an Environment

Users can use an Environment to collaborate on shared resources and configure the visibility of sensitive data, including API secrets, passwords, tokens, and keys.

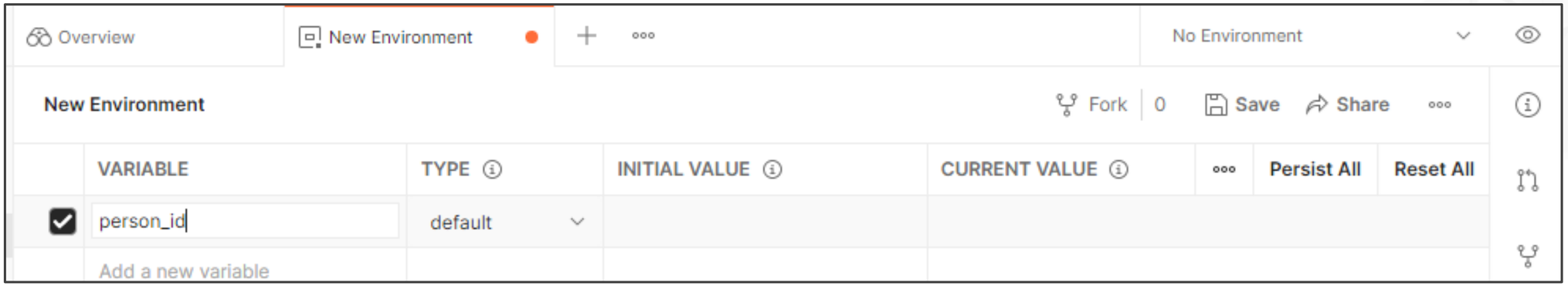



Source: learning.postman.com

# Managing Environment Variables

Creating an Environment is simple. Here are the steps to create an Environment:

- Click on the **New** button
- Select the **Environment** option
- Enter user ID in the **VARIABLE** column's first field
- Set **INITIAL VALUE** to **1**
- Click on **Save** to save the Environment
- Exit the dialog box



# Managing Environment Variables


- Click the drop-down menu on the upper right corner of the screen
- Select **Environment_01** from the drop-down list
- Send a **Get** request

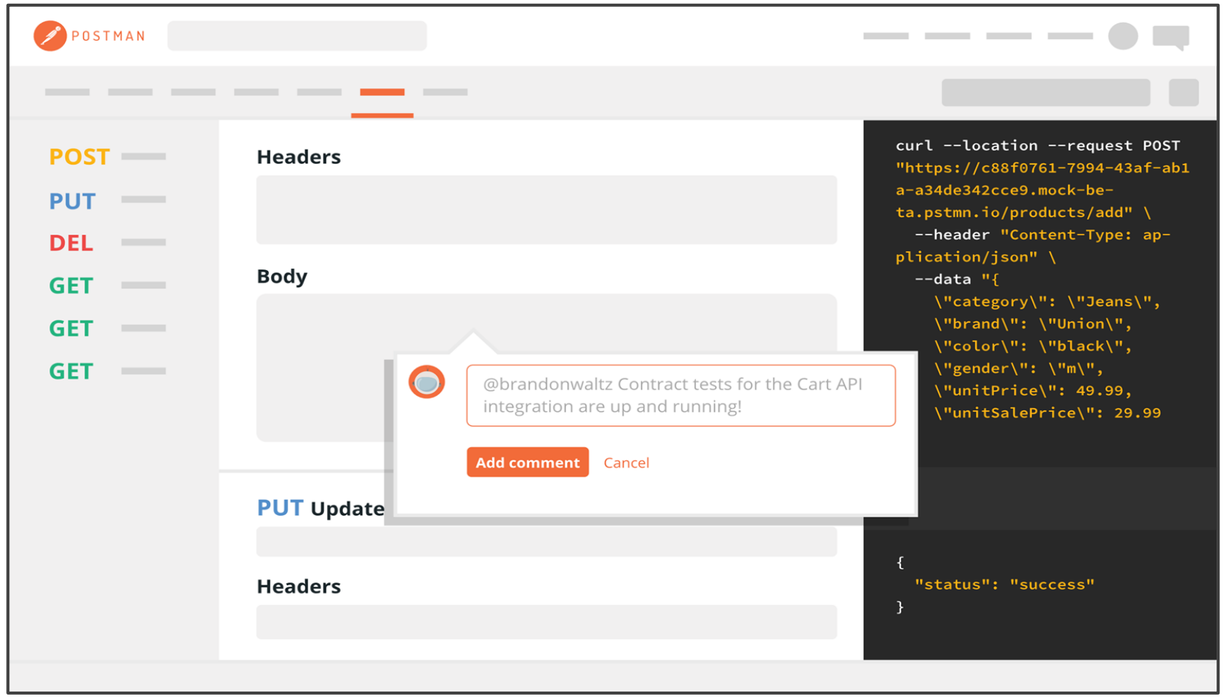
Users can send **Get** requests as the Environment is already active.

# End-to-End Case Automation

# End-to-End Case Automation

Users need to integrate automated testing in the CI/CD pipeline to ensure the changes in the code will not break the API during production.

# End-to-End Case Automation




1. Understand the API
2. Build the test suite
3. Work and verify with team
4. Parameterize requests
5. Run the tests and debug
6. Integrate with DevOps lifecycle

# Automation Suite Creation with Postman

Here are some terms and their meanings used in Postman that users need to know before constructing an automation suite in the tool:

### Collection

It is a bunch of saved requests that users can organize into folders. It aids in the creation of comprehensible request hierarchies.

### Global/Environmental Variables

A set of key-value pairs is called an Environment. It allows users to customize requests with variables, enabling them to move between different setups without changing the requests quickly.

# Automation Suite Creation with Postman

## Authentication Options

Authorization is used by APIs to verify if the client requests have secure access to data. Postman supports a variety of authorization techniques as well.

## Pre-Request

Pre-request scripts are pieces of code executed before a Collection request is issued. These are commonly used to generate values and inject them into requests through Environment variables.

# Automation Suite Creation with Postman

### Tests

Tests are JavaScript-based scripts that are run after a response is received.
Tests can be executed as part of a single request or a set of requests.

### In-built JS Snippets

Postman allows users to develop a JavaScript code that asserts responses and checks them automatically.
Assertions can be written using the **Snippets** feature of the **Tests** tab.

# Advance Validation on Response in Rest Assured

# Advance Validation on Response in Rest Assured

There are advanced assertions used in Postman to validate a response in different complex ways. Advanced validations are of two types:

## Response body validation against Schema

```
var jsonResponse = pm.response.json();
pm.test('Schema is valid', function()
{pm.expect(tv4.validate(jsonResponse, schema)) .to.be.
true;});
```

# Advance Validation on Response in Rest Assured

There are advanced assertions used in Postman to validate a response in different complex ways. Advanced validations are of two types:

## Assertion of a value in the particular field on JSON

```
pm.test("value of per page field is 6", function () {
var jsonData = pm.response.json();
pm.expect(jsonData.per_page).to.eql(6);});
```

# How to Validate a Get Response?

# Validation in Postman

Here are some features of validating documents, tests, monitors, and more in Postman:

Users can check elements, such as documents, tests, fake servers, monitors, and so on, against their API schema in Postman.

It maintains the clarity of their API and guarantees that the items are consistent with the schema. Postman displays a list of issues discovered along with the remedies, if there is a validation error on the schema, or if elements do not fit.

The API elements can then be fixed and revalidated.

# Validation in Postman

## Validating schemas

Postman highlights the validation problems that arise as users change their API schema on the **Definition** tab on the API version page.
These errors include missing mandatory fields, improper field names, inappropriate data types, incorrect nesting, and other API schema validation problems.

## Validating elements

Users may assess whether the elements of their API schemas need to be changed to stay in sync. The assessment is done by comparing the elements to the documentation, mock servers, tests, and associated monitors.

## Validating requests

When a request is sent, Postman verifies it. If there are problems with the request, Postman displays a warning message with the request's name, and the number of problems adjacent to it.

# Key Takeaways

- Users can use an Environment to collaborate on shared resources and configure the visibility of sensitive data, including API secrets, passwords, tokens, and keys.
- An Environment can be used to group related sets of values and manage access to shared Postman data.
- There are advanced assertions used in Postman to validate a response in different complex ways.
- End-to-end test cases can be automated using the Postman API suite.