Using variables

Variables allow you to store and reuse values in your requests and scripts. By storing a value in a variable, you can reference it throughout your collections, environments, and requests—and if you need to update the value, you only have to change it in one place. Using variables increases your ability to work efficiently and minimizes the likelihood of error.

Variables quick start

To try out a variable, use the following steps:

- Click the **Environment quick look** (eye button) in the top right of Postman and click **Edit** next to **Globals**.
- Add a variable named `my_variable` and give it an initial value of `Hello`—click **Save** and close the environment modal.
- Open a new request tab and enter `https://postman-echo.com/get?var={{my_variable}}` as the URL. Hover over the variable name and you'll see the value.
- **Send** the request. In the response, you'll see that Postman sent the variable value to the API. *Try changing the value in the Environment quick look and sending the request again*.

Read on for more detail on how you can use variables in Postman.

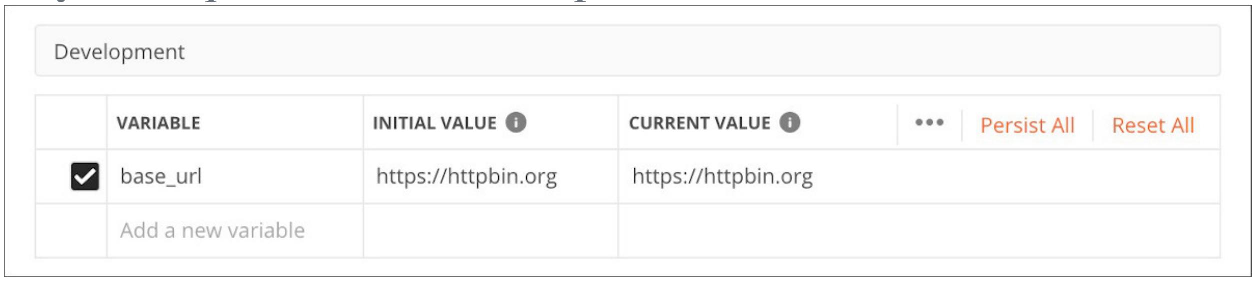
Contents

-
-

Understanding variables and environments

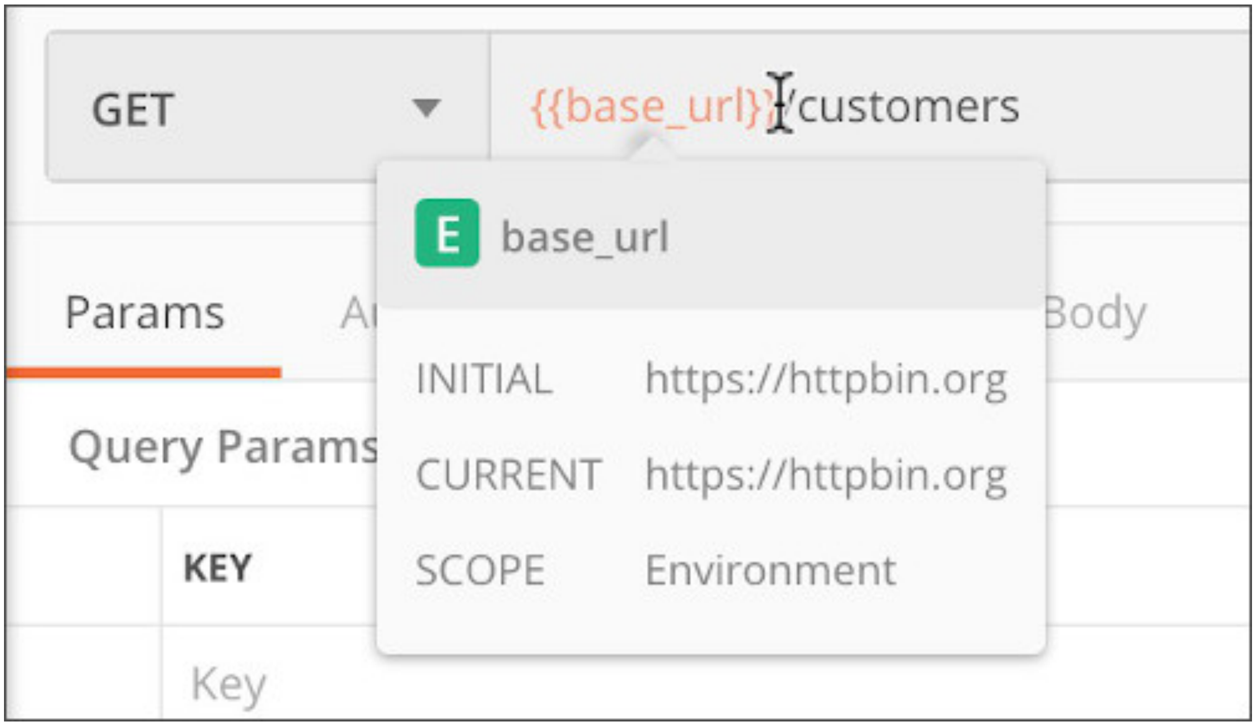
A variable is a symbolic representation of data that allows you to access a value without having to enter it manually wherever you need it. This can be useful especially if you are using the same values in multiple places. Variables make your requests more flexible and readable, by abstracting some of the detail involved.

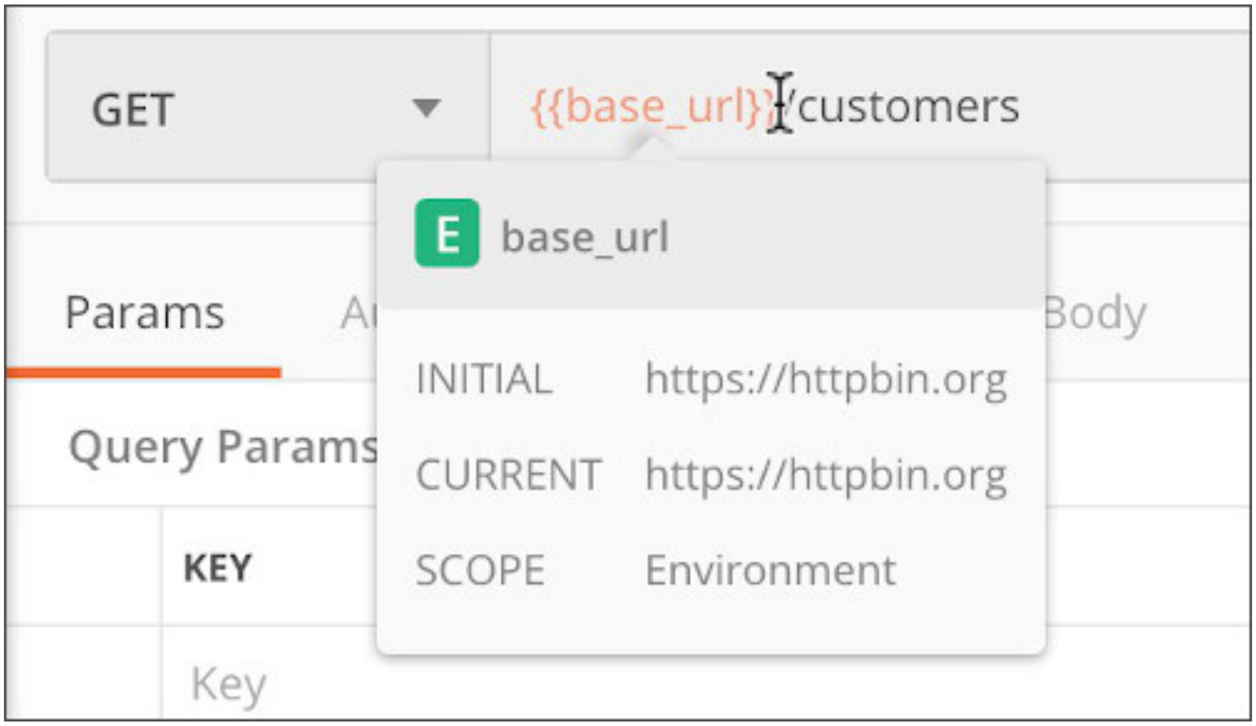
For example, if you have the same URL in multiple requests—but the URL might change—you can store it in a variable. If the URL changes, you only need to change the variable value and it will be reflected throughout your collection, wherever you've used the variable name. The same principle applies to any part of your request code that is repeated.



Development

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	base_url	https://httpbin.org	https://httpbin.org
	Add a new variable		





Postman supports variables at different scopes, allowing you to tailor your processing to a variety of development, testing, and collaboration tasks. Scopes in Postman relate to the different contexts that your requests run in—within the Postman app, in collections, in environments, and in Newman / the Collection Runner.

Postman will store environment and global variables as strings. If you’re storing objects or arrays, remember to `JSON.stringify()` them before storing, and `JSON.parse()` them when you retrieve them.

Environments in Postman

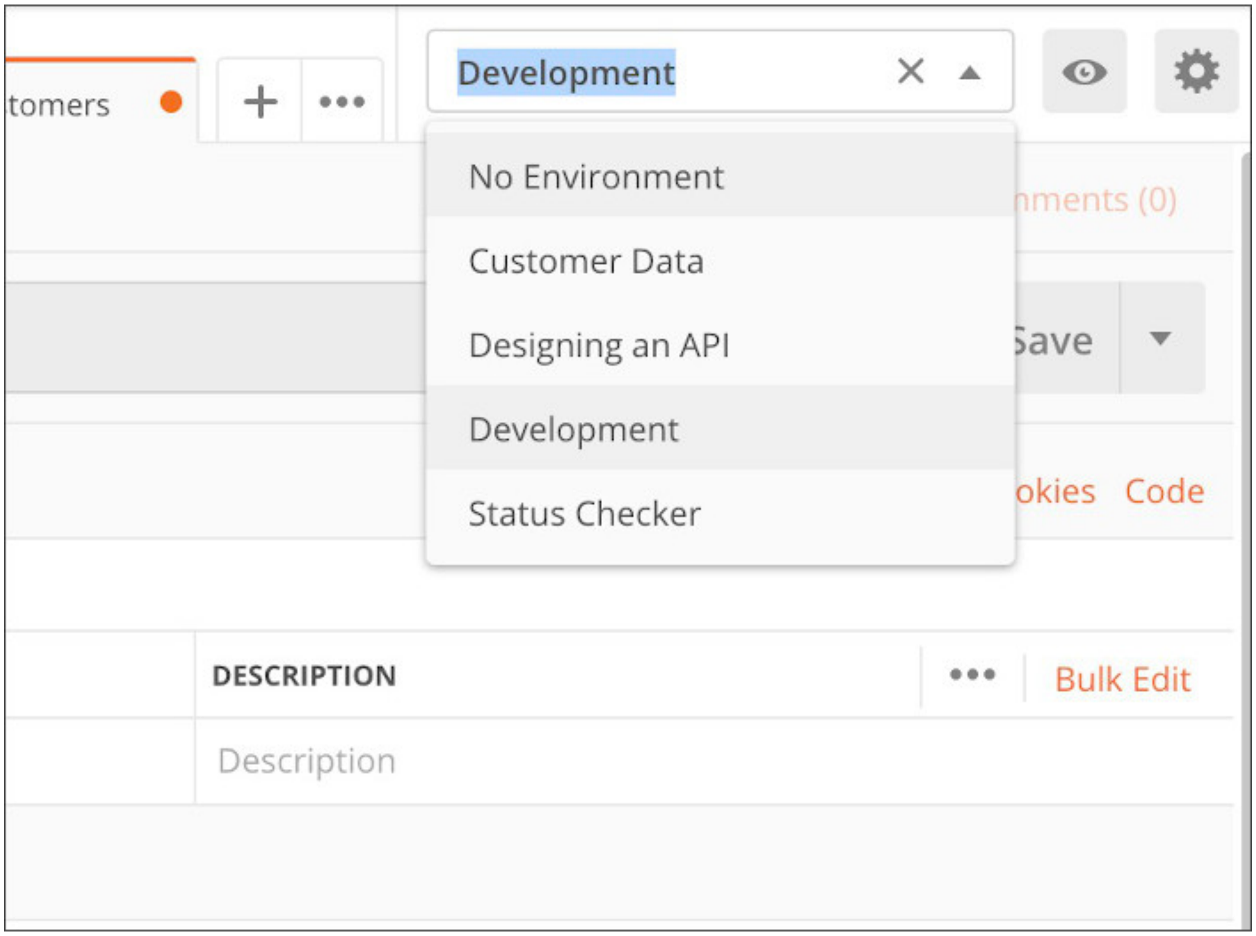
Environments allow you to run requests and collections against different data sets. For example, you could have an environment for development, one for testing, and another for production. You can use variables to pass data between requests and tests, for example if you are chaining requests using a collection.

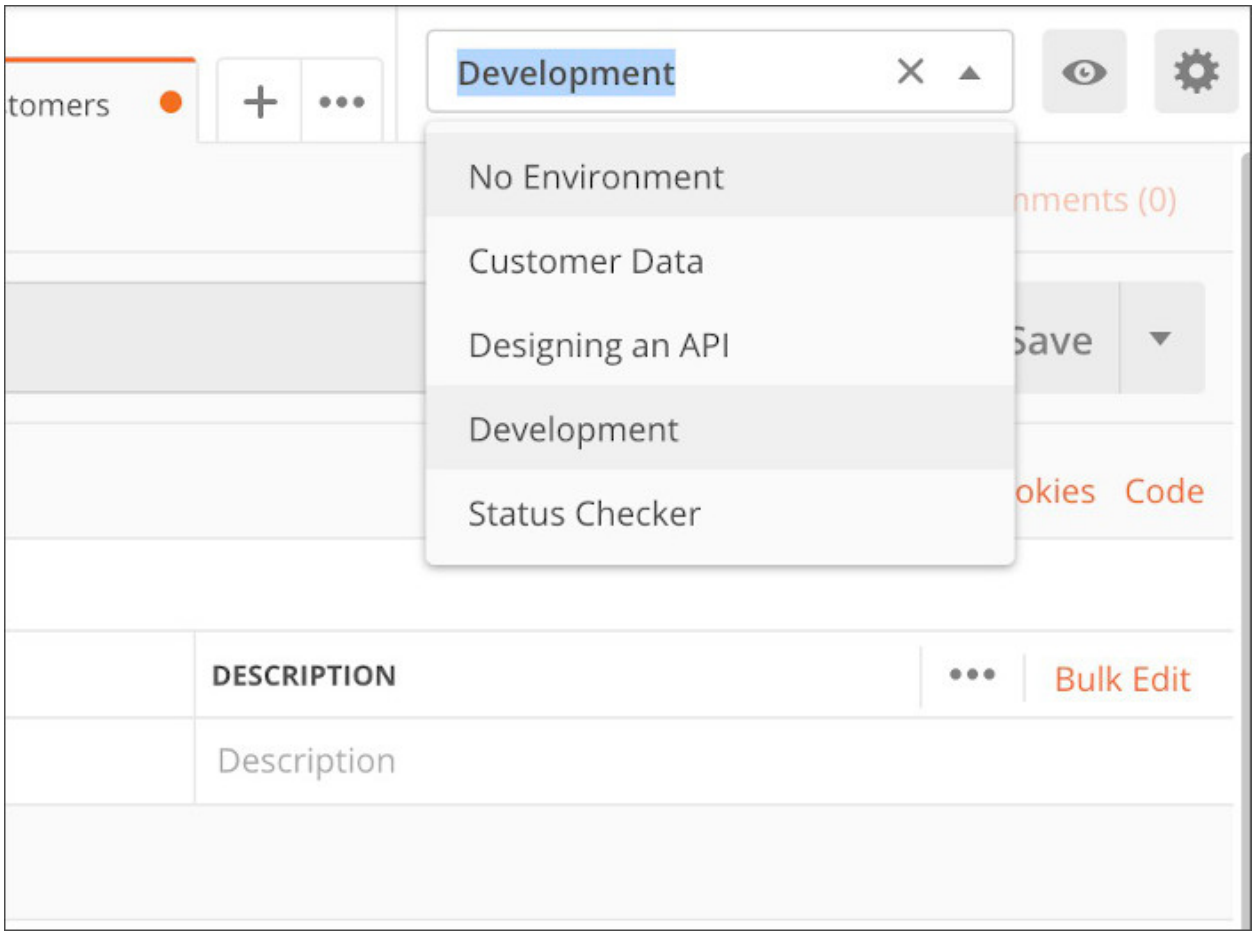
Environments in Postman are key-value pairs of variables. Each variable name represents its key, so referencing the variable name allows you to access its value.

For example, if you have a base URL for requests stored in a variable named `base_url`, you can reference it in your requests using `{{base_url}}`. Whatever value is stored in the variable will be included wherever you've referenced the variable when your requests run. If the base URL value is `https://httpbin.org`, and is listed as part of the request URL using `{{base_url}}/get?customers=new`, Postman will send the request to `https://httpbin.org/get?customers=new`.

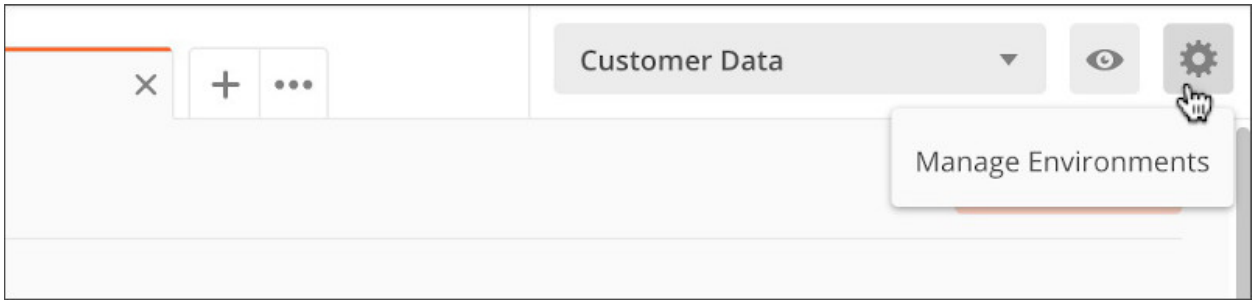
To create an environment, use **New > Environment**, or the **Manage environments** button in the top right of Postman, and click **Add**. Give your environment a name , such as "Testing"—you can add variables at creation or later, by editing the environment.

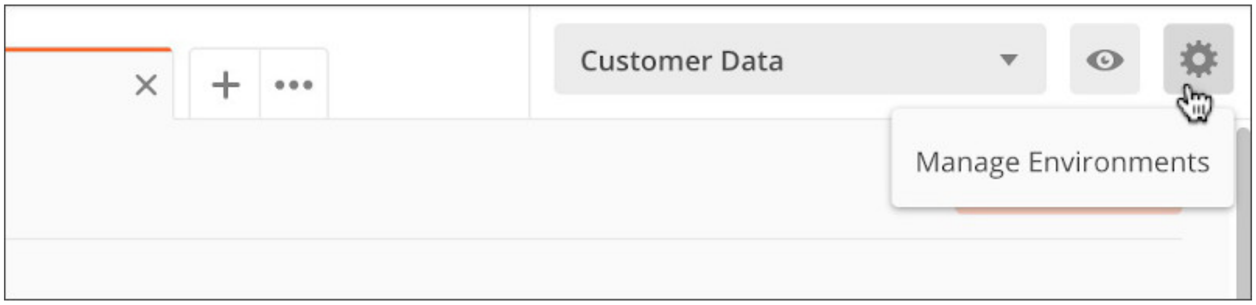
To select the environment to run your requests within, use the drop-down at the top right of the Postman app.

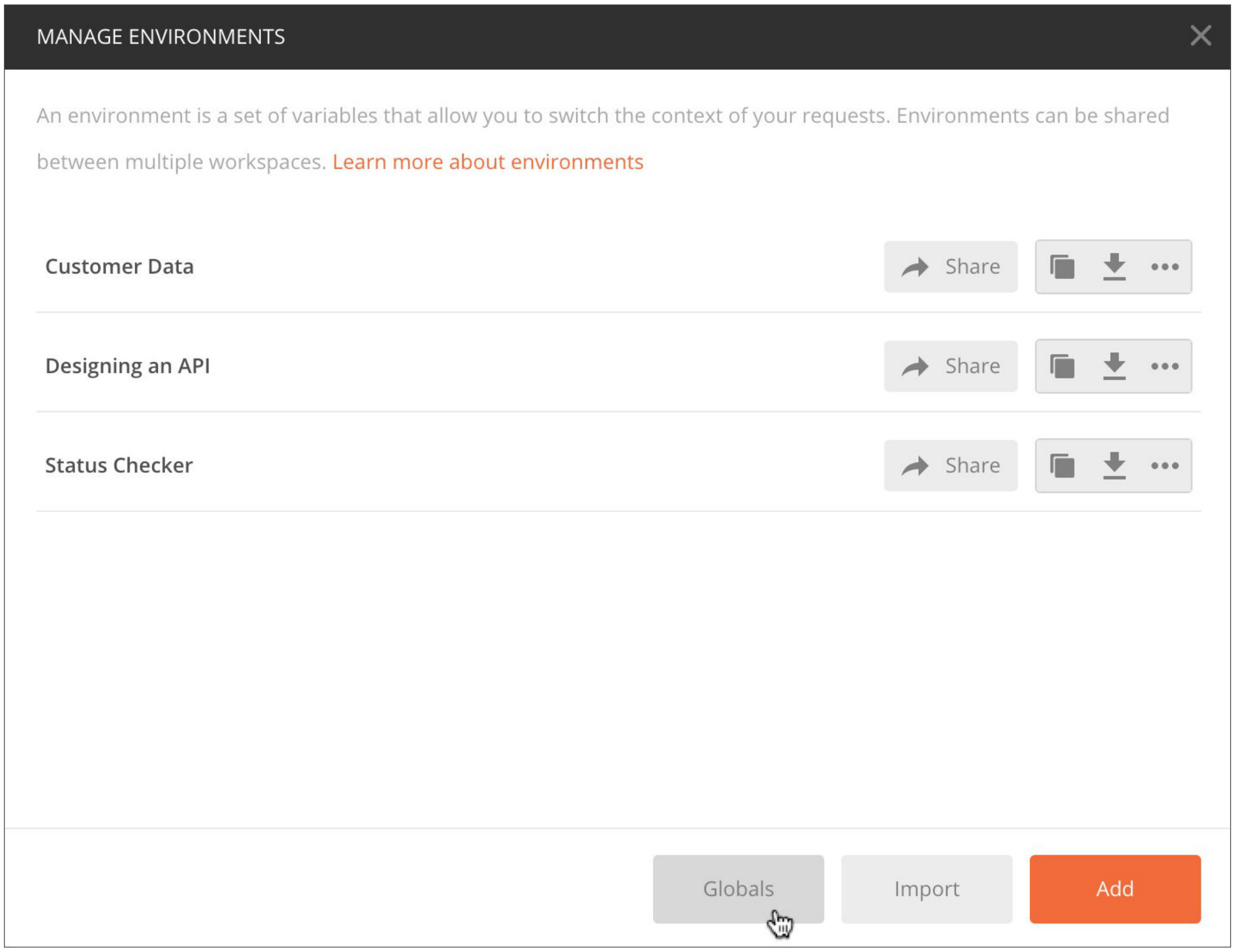

tomers
Development
No Environment
Customer Data
Designing an API
Development
Status Checker
nments (0)
Save
okies
Code
DESCRIPTION
Bulk Edit
Description



Your requests will run with the data variables listed in the environment you have selected.
You can duplicate, delete, download / import environment JSON, and share environments with collaborators in **Manage Environments**. Sharing environments allows you to let other people run your requests against the same data sets.






MANAGE ENVIRONMENTS

An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. Learn more about environments

Customer Data

Designing an API

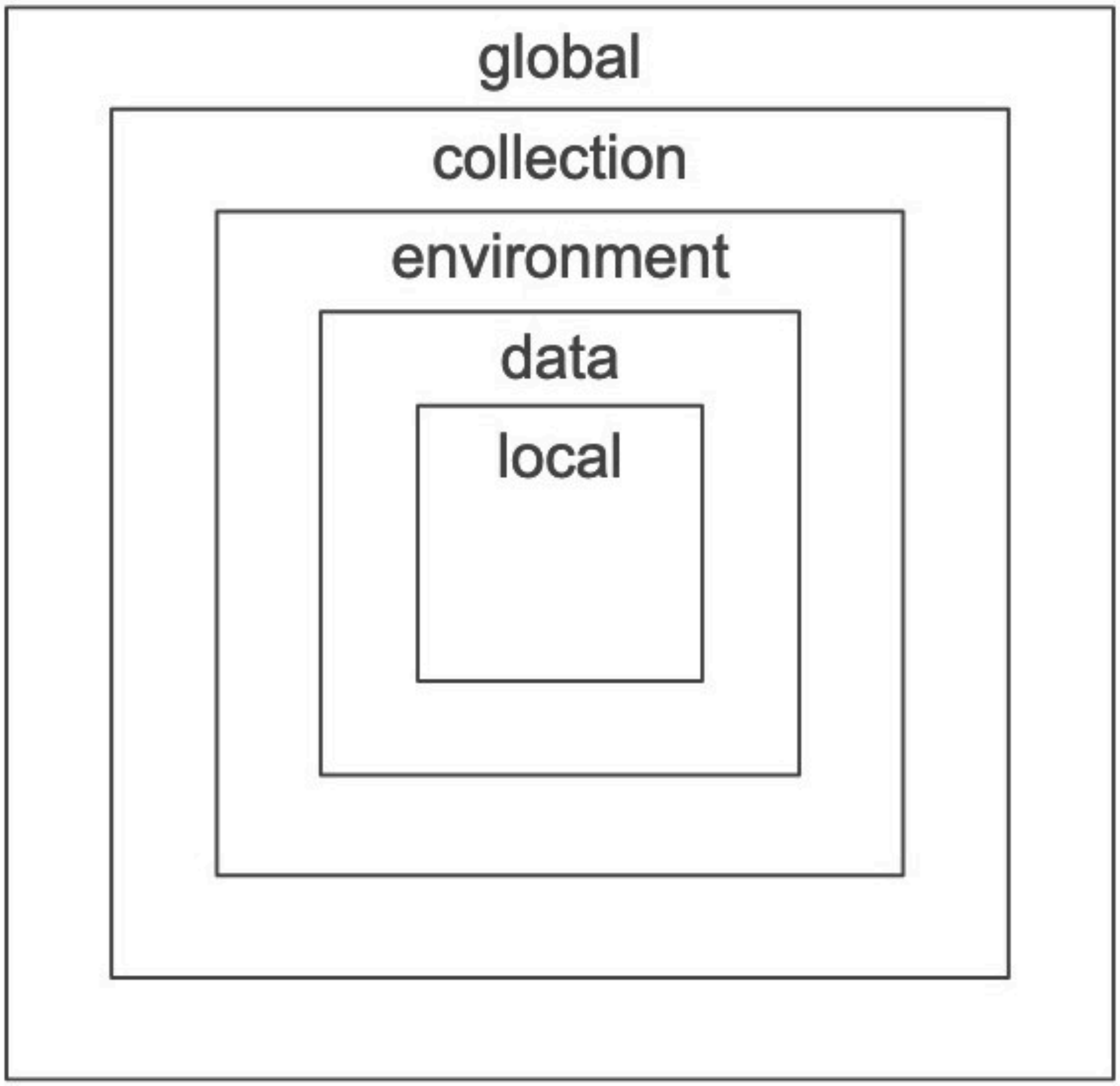
Status Checker

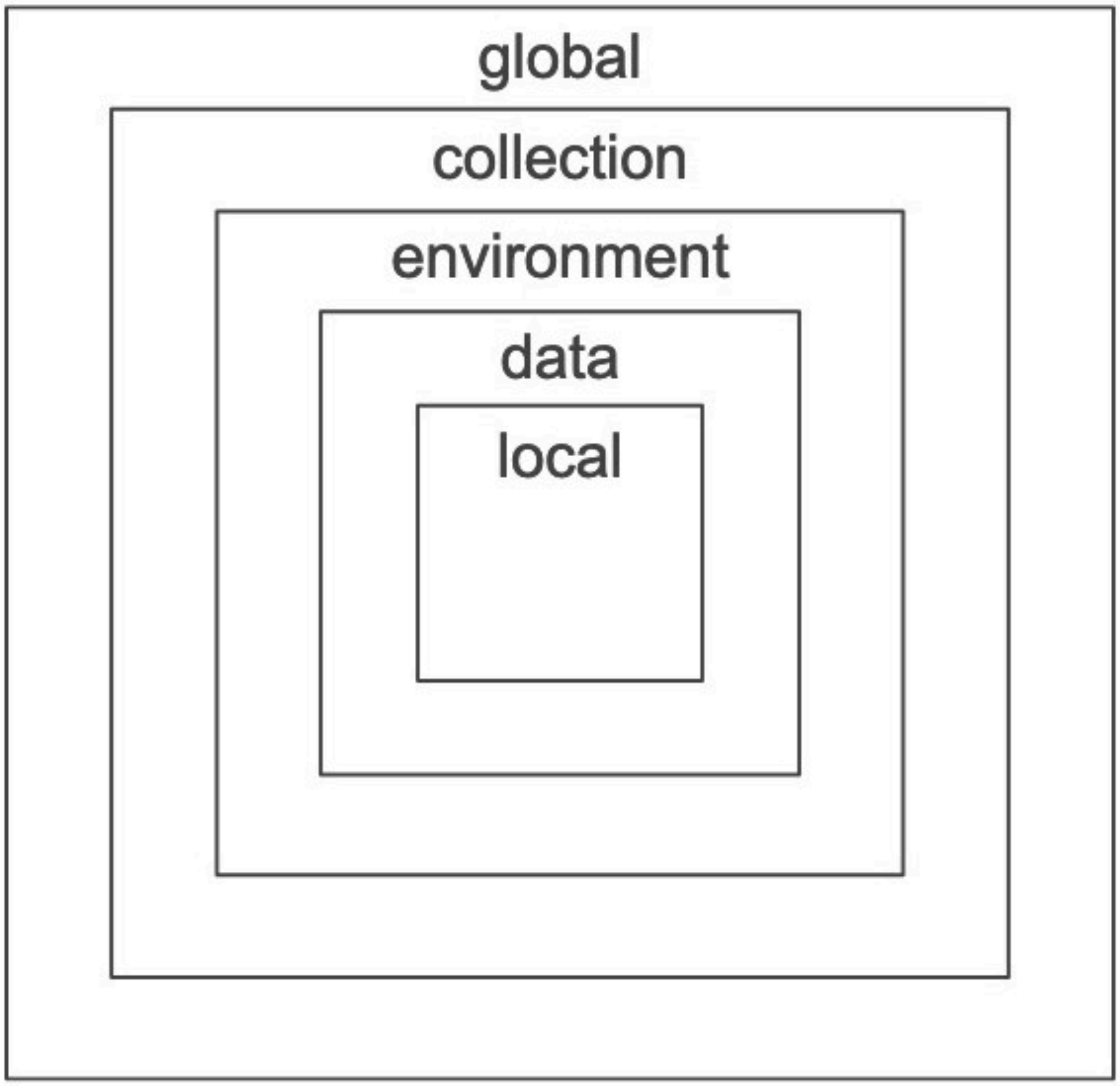
To share an environment safely, create a duplicate and remove any sensitive data such as auth values first. When your collaborator imports the environment they can enter their own credentials. Any changes you make to a shared environment will be reflected throughout the team with access to it.

Variable scopes

Postman supports the following variable scopes:

- Global
- Collection
- Environment
- Data
- Local


global
collection
environment
data
local



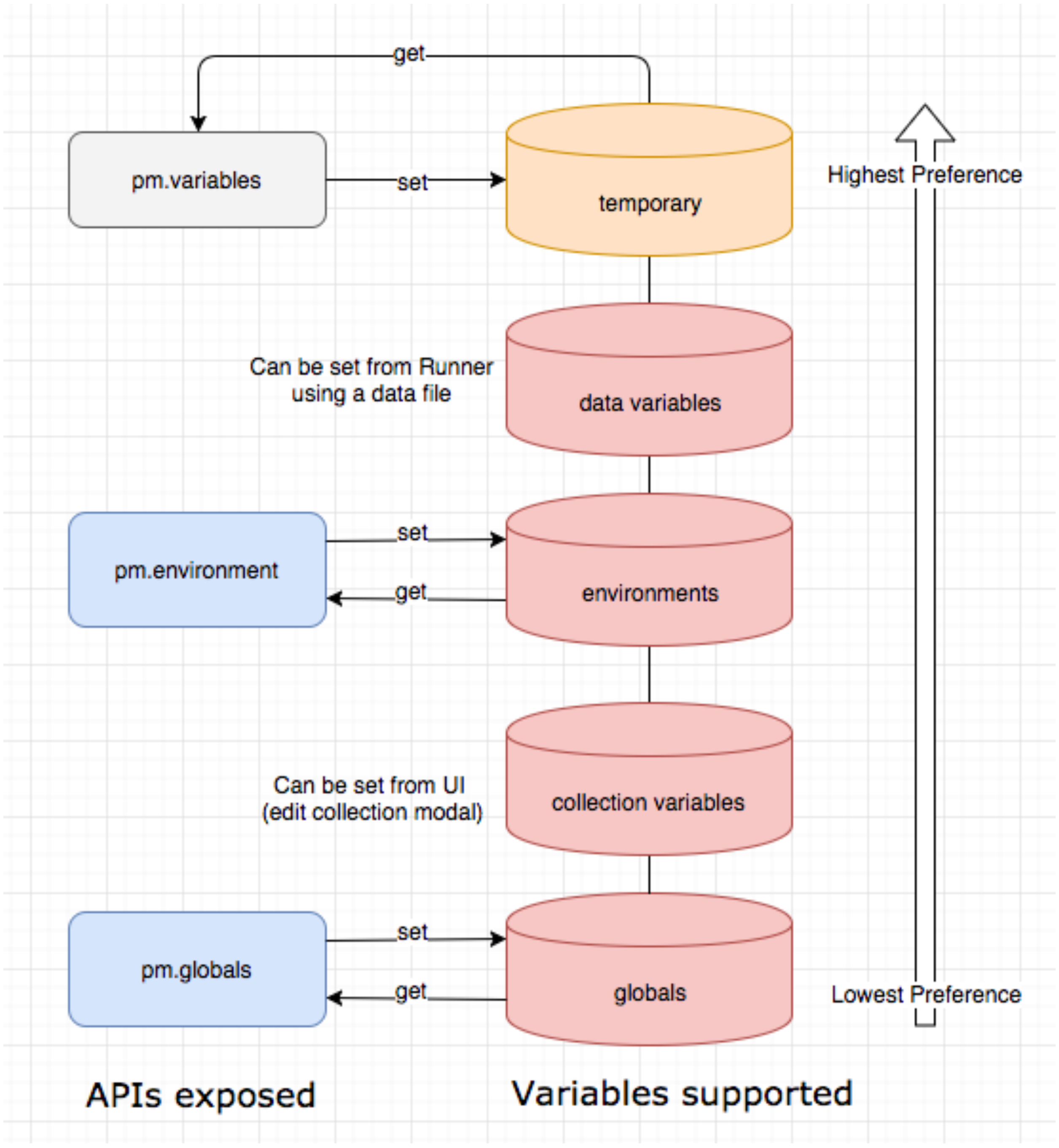
If a variable with the same name is declared in two different scopes, the value stored in the variable with narrowest scope will be used—for example if there is a global and a local variable both named `username`, the local value will be used when the request runs.

Choosing variables

Variable scopes are suited to different tasks in Postman:

- **Global variables** allow you to access data between collections, requests, test scripts, and environments.
    - *Since global variables can create confusion, you should only use them sparingly—for example to quickly test something or when your project is at a very early prototyping stage.*
- **Collection variables** are available throughout the requests in a collection and are independent of environments, so do not change based on the selected environment.
    - *Collection variables are suitable if you are only using a single environment, for example for auth / URL details.*
- **Environment variables** allow you to tailor your processing to different environments, for example local development vs testing or production. Only one environment can be active at a time.
    - *If you only have one environment, using collection variables is more efficient.*
- **Local variables** are temporary, and only accessible in your request scripts. Local variable values are scoped to a single request or collection run, and are no longer available when the run is complete.

    - *Local variables are suitable if you need a value to override all other variable scopes but do not want the value to persist once execution has ended.*
- **Data variables** come from external CSV and JSON files to define data sets you can use when running collections via Newman or the Collection Runner.



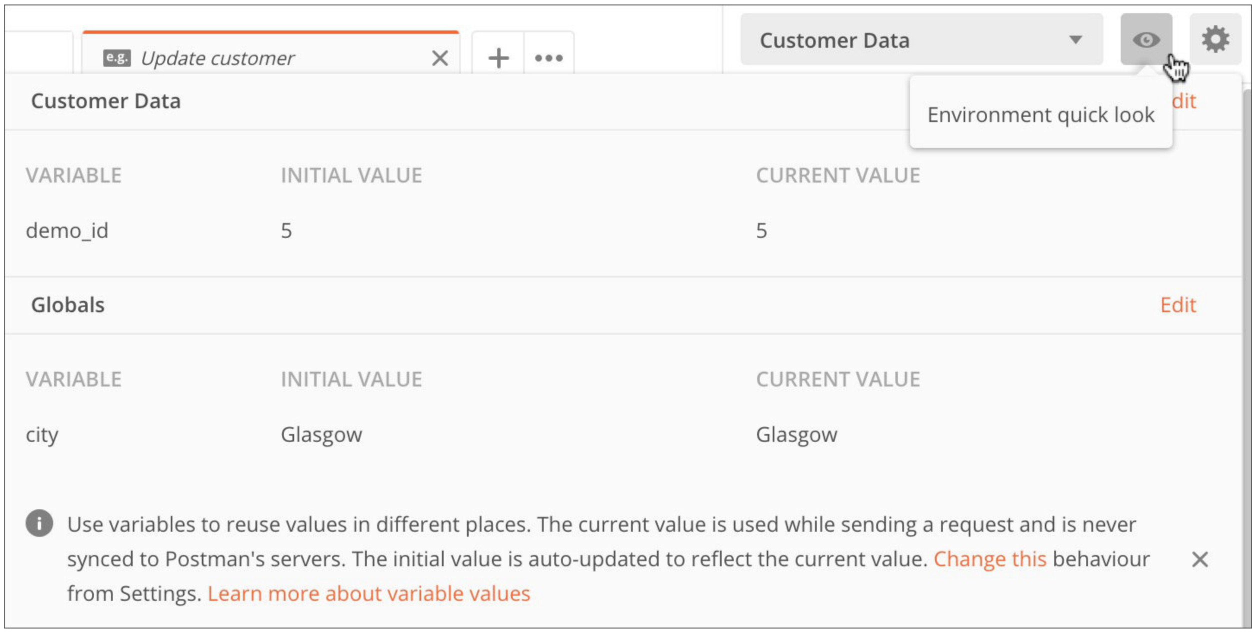
Defining variables

You can define variables in a variety of ways, depending on whether you need , or scope.

Remember to delete variables you are no longer using.

Defining global and environment variables

To create or edit a variable at global or environment scope, use the **Environment quick Look** and click **Edit** next to the environment or global variable.



Customer Data

VARIABLE	INITIAL VALUE	CURRENT VALUE
demo_id	5	5

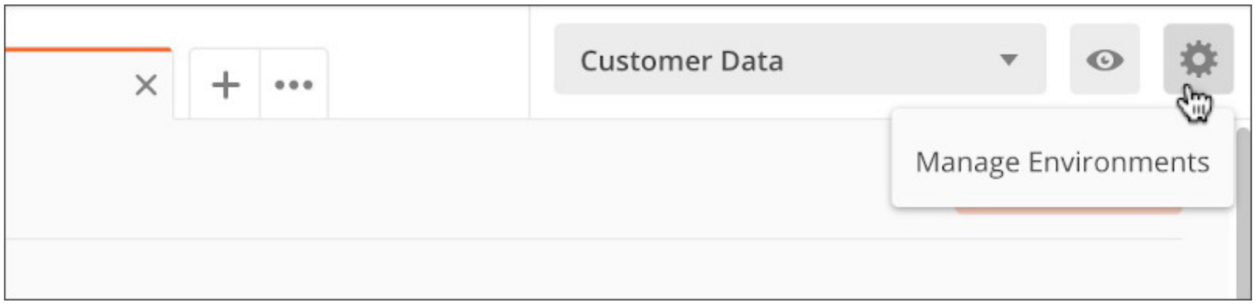
Globals

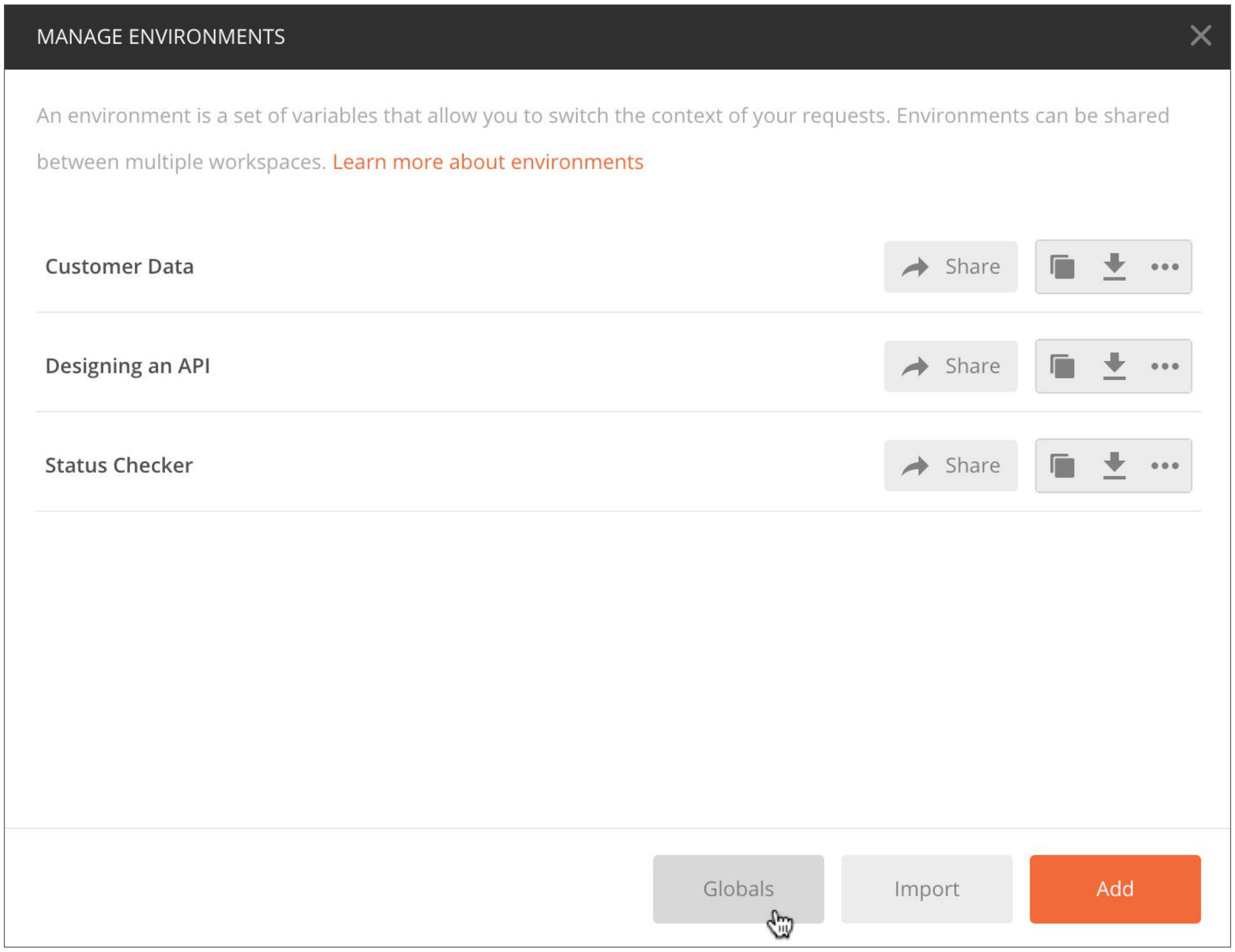
VARIABLE	INITIAL VALUE	CURRENT VALUE
city	Glasgow	Glasgow

Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. Change this behaviour from Settings. Learn more about variable values

The quick look view provides a reference to the current state of any global or environment variables you're working with—you can also access the quick look using the keyboard shortcut `CMD/CTRL + ALT + E`.

Alternatively, click **Manage environments** and select the relevant environment, or click **Globals**.




MANAGE ENVIRONMENTS

An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. Learn more about environments

Customer Data

Designing an API

Status Checker

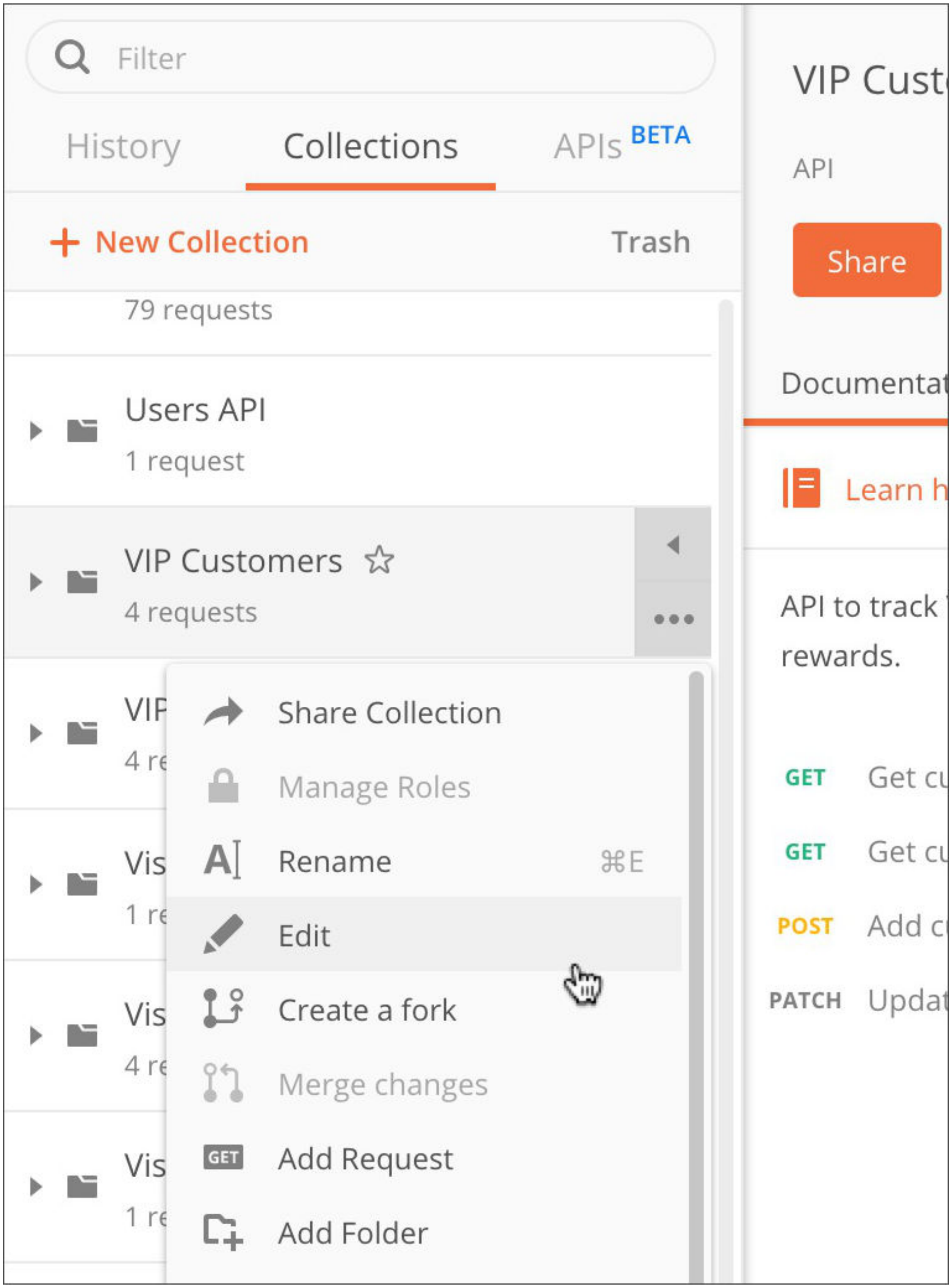
You can   if you do not already have one.

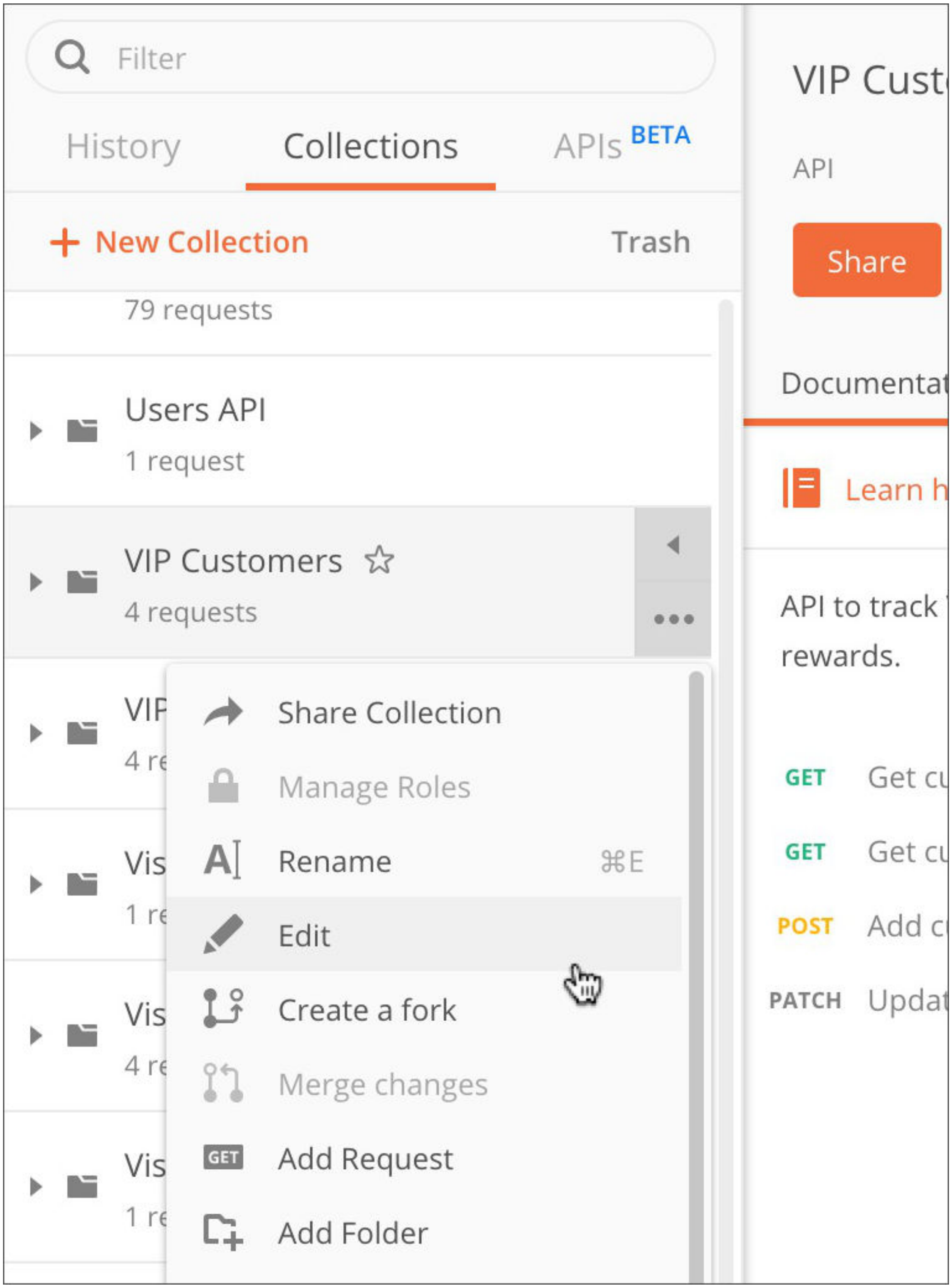
Once you have your scope selected, you can
.

You can also ______________________________________.

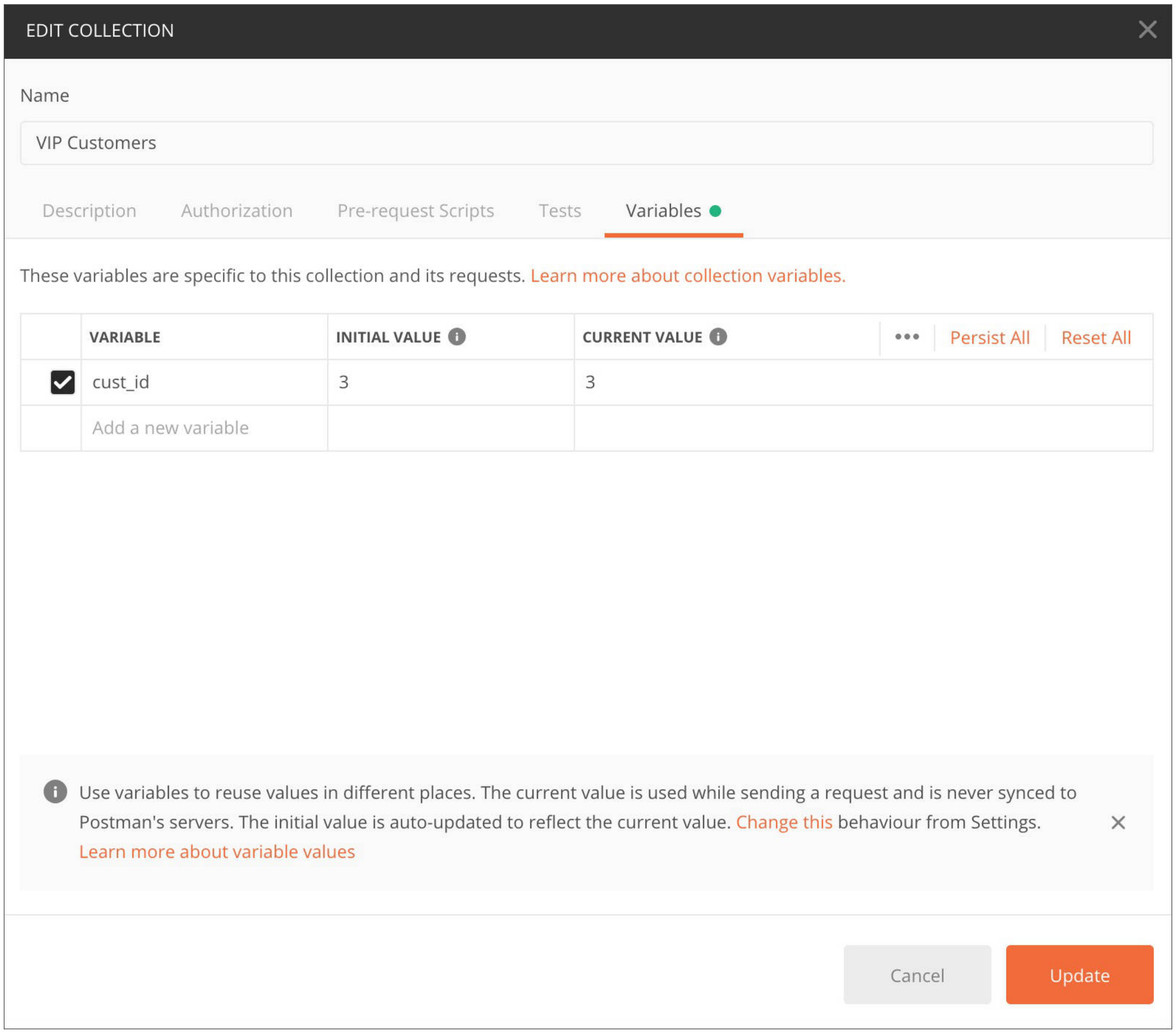
Defining collection variables

To create or edit a variable for an existing collection, select the collection in **Collections** on the left of the Postman app, open the **View more actions** (**...**) menu, and click **Edit**.


Filter
History
Collections
APIs BETA
New Collection
Trash
79 requests
Users API
1 request
VIP Customers
4 requests
VIP
4 re
Vis
1 re
Vis
4 re
Vis
1 re
Share Collection
Manage Roles
Rename ⌘E
Edit
Create a fork
Merge changes
GET Add Request
Add Folder
VIP Cust
API
Share
Documentat
Learn h
API to track
rewards.
GET Get cu
GET Get cu
POST Add c
PATCH Updat


Filter
History
Collections
APIs BETA
+ New Collection
Trash
79 requests
Users API
1 request
VIP Customers
4 requests
VIP
4 re
Vis
1 re
Vis
4 re
Vis
1 re
Share Collection
Manage Roles
Rename ⌘E
Edit
Create a fork
Merge changes
Add Request
Add Folder
VIP Cust
API
Share
Documentat
Learn h
API to track
rewards.
GET Get cu
GET Get cu
POST Add c
PATCH Updat

Choose the **Variables** tab to edit or add to your collection variables.



EDIT COLLECTION

Name

VIP Customers

These variables are specific to this collection and its requests. Learn more about collection variables.

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	cust_id	3	3
	Add a new variable		

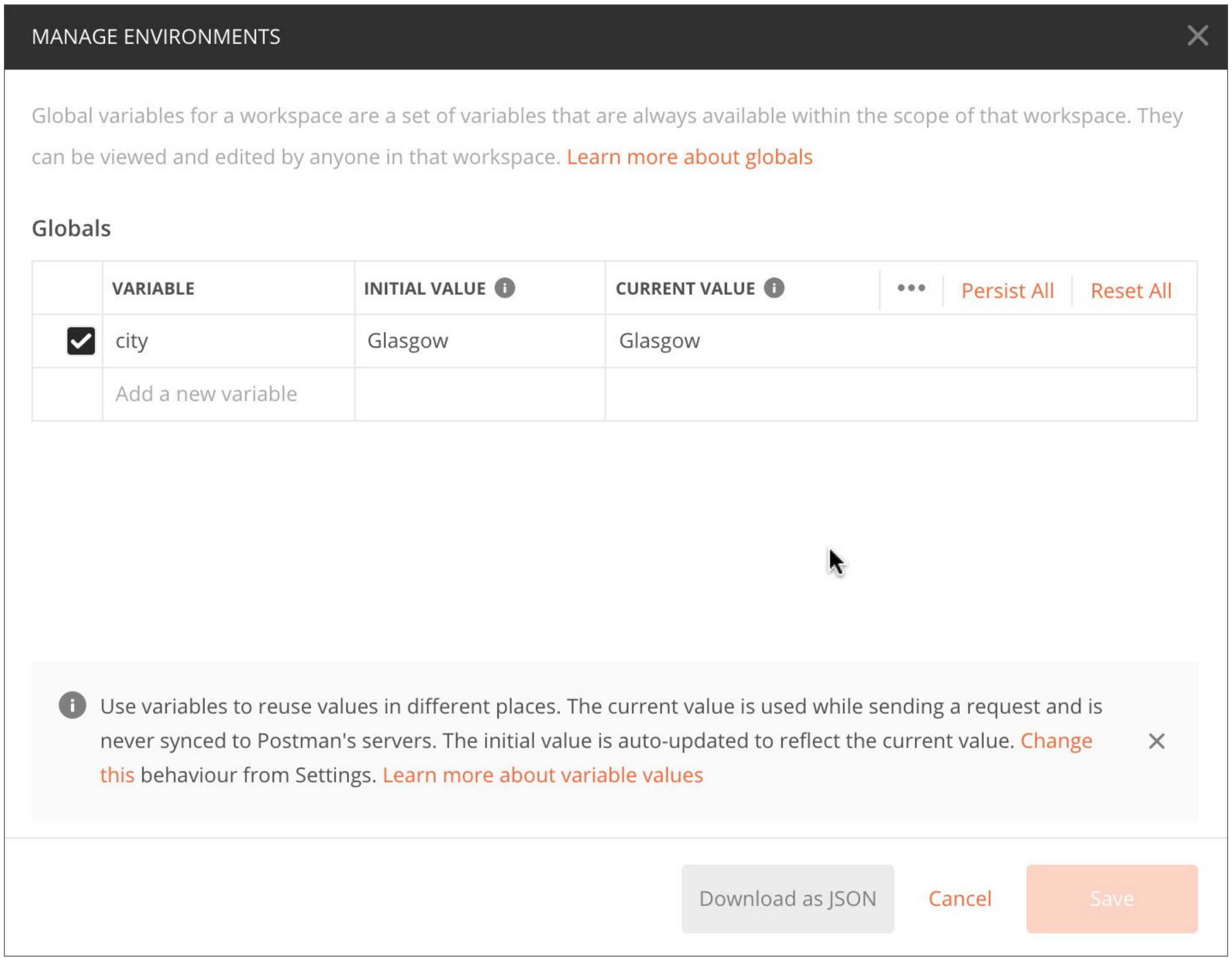
Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. Change this behaviour from Settings. Learn more about variable values

You can add collection variables when you create a new collection.

You can also ________________________________.

Specifying variable detail

You can add and edit variables at any time. All you need to include for a new variable is a name—you can choose to supply an initial value but can alternatively set it later, including from scripts. Use the checkbox to enable or disable a variable.



Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. Learn more about globals

Globals

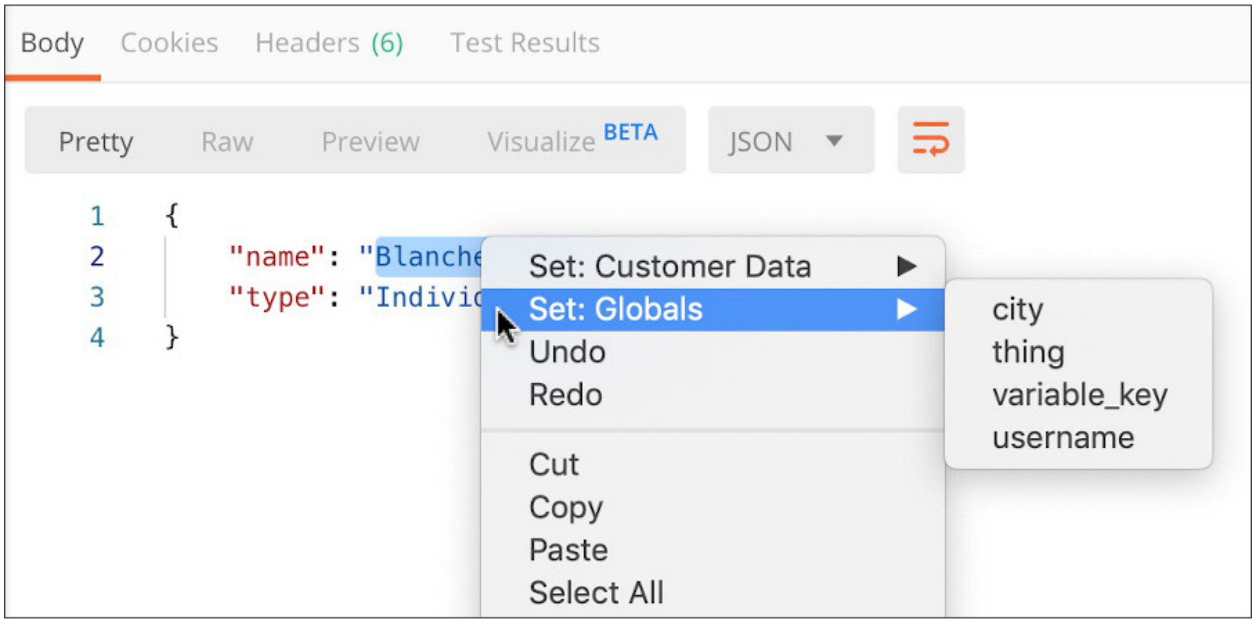
	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	city	Glasgow	Glasgow
	Add a new variable		

Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. Change this behaviour from Settings. Learn more about variable values

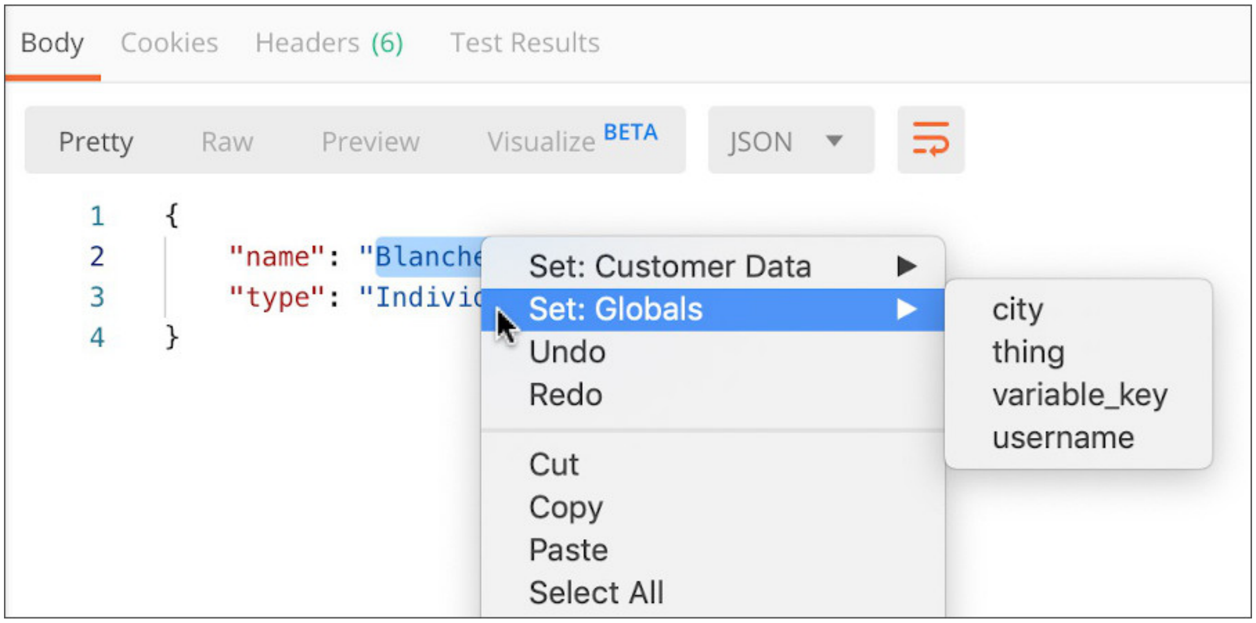
Initial values are shared when you share a collection or environment. Current values are local and not synced or shared. See for more on local vs synced variables.

You can download global variables as JSON from **Manage Environments**.

You can set response body values to variables by selecting text, right-clicking / CTRL + clicking, and choosing the relevant variable by name.



```
{
    "name": "Blanche
    "type": "Indivic
}
```



```
{
    "name": "Blanche
    "type": "Indivic
}
```

Defining variables in scripts

You can set variables programmatically in your request scripts.

Use to define a global variable:

```
pm.globals.set("variable_key", "variable_value");
```

Use   to define a collection variable:

```
pm.collectionVariables.set("variable_key",
"variable_value");
```

Use   to define an environment variable (in the currently selected environment):

```
pm.environment.set("variable_key", "variable_value");
```

Check out the   for more on scripting with variables.

Defining local variables

Local variables are temporary values you set in your request scripts using the following syntax:

```
pm.variables.set("variable_key", "variable_value");
```

Local variables do not persist between sessions, but allow you to override all other scopes temporarily, during the execution of a request or collection / monitor run. For example, if you need to process a temporary test value for a single request or collection run locally, and don't want the value to sync with your team or remain available when the request / collection has finished running, you can use a local variable.

Accessing variables

You can use double curly braces to reference variables throughout the Postman app user interface. For example, to reference a variable named "username" in your request auth settings, you could use the following syntax with double curly braces around the name:

```
{{username}}
```

When you run a request, Postman will resolve the variable and replace it with it's current value.

For example, you could have a request URL referencing a variable as follows:

```
http://pricey-trilby.glitch.me/customer?id={{cust_id}}
```

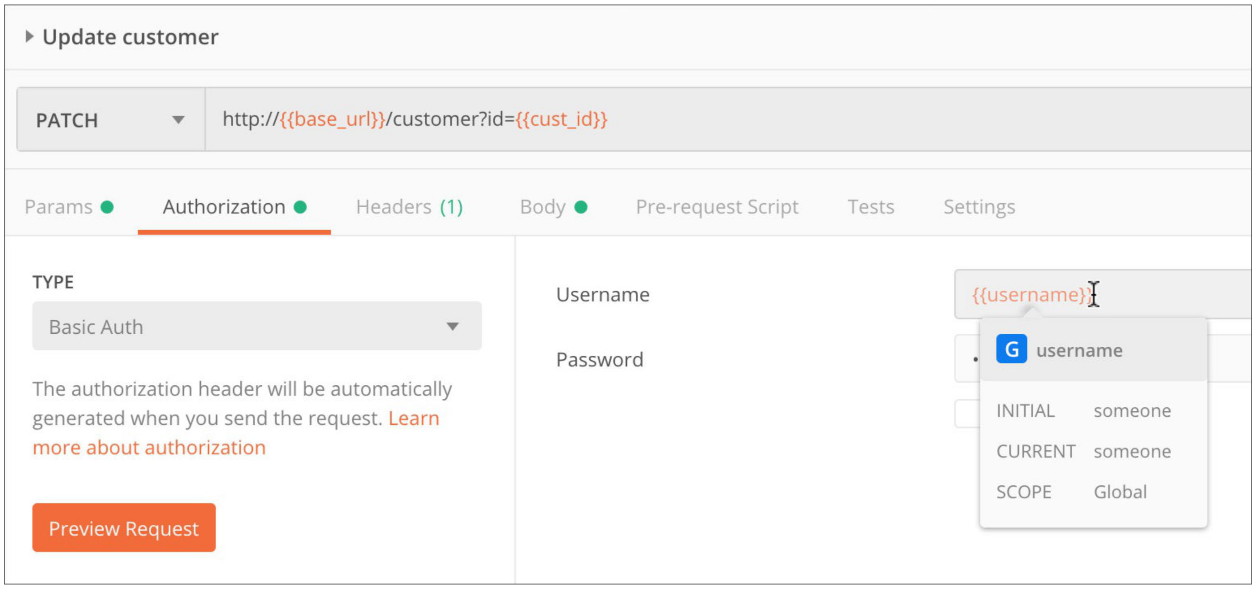
Postman will send whatever value you currently have stored for the `cust_id` variable when the request runs. If `cust_id` is currently `3`, the request will be sent to the following URL including query parameter:

```
http://pricey-trilby.glitch.me/customer?id=3
```

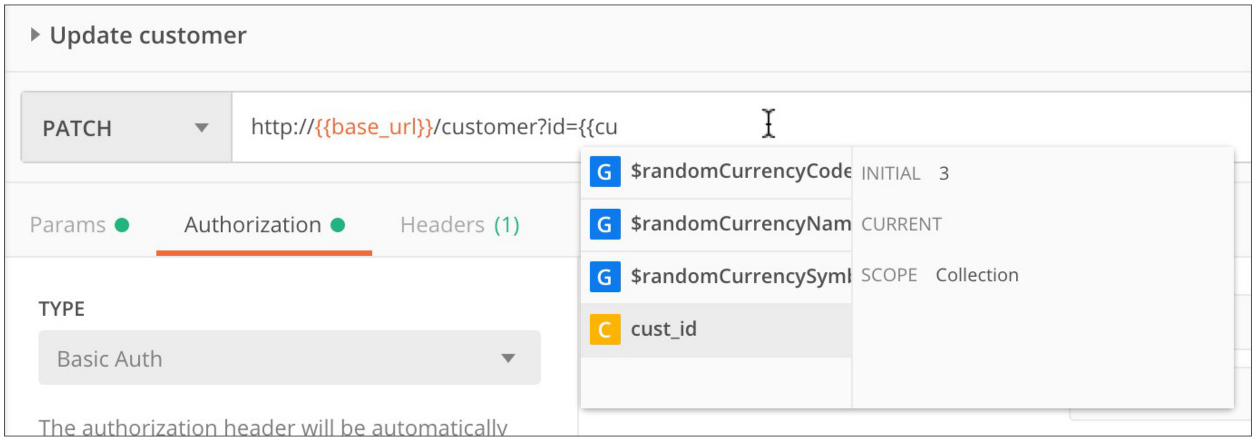
Alternatively, you could have a request body that accesses a variable by wrapping its reference in double-qoutes:

```
{ "customer_id" : "{{cust_id}}" }
```

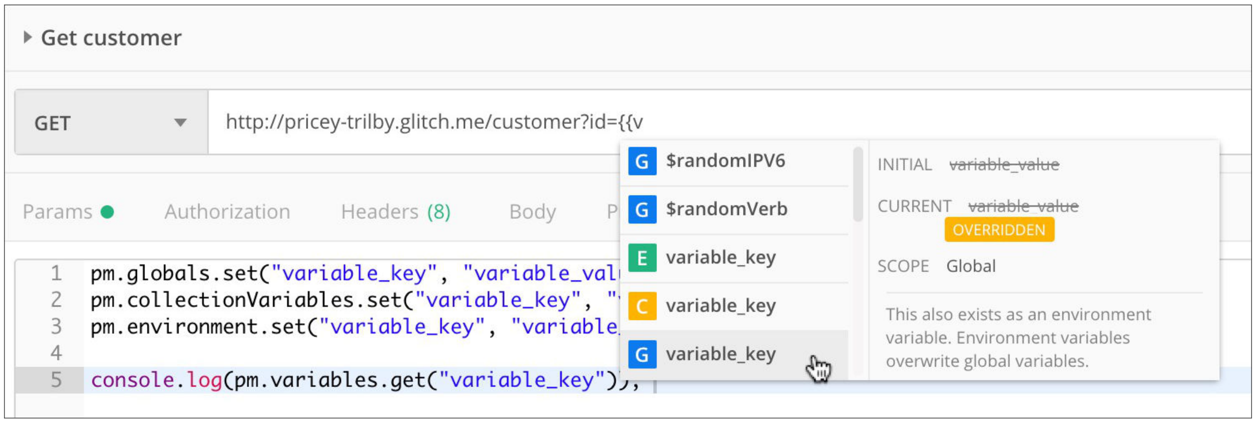
You can use variables in request URLs, parameters, headers, authorization, body, and header presets.



When you hover over a variable you can see an overview of its current status. As you type variables into your requests, Postman will prompt you with any that are currently defined.



The prompt will indicate current value, scope (highlighted by color), and overridden status where relevant.


Get customer

GET http://pricey-trilby.glitch.me/customer?id={{v

Params ● Authorization Headers (8) Body P

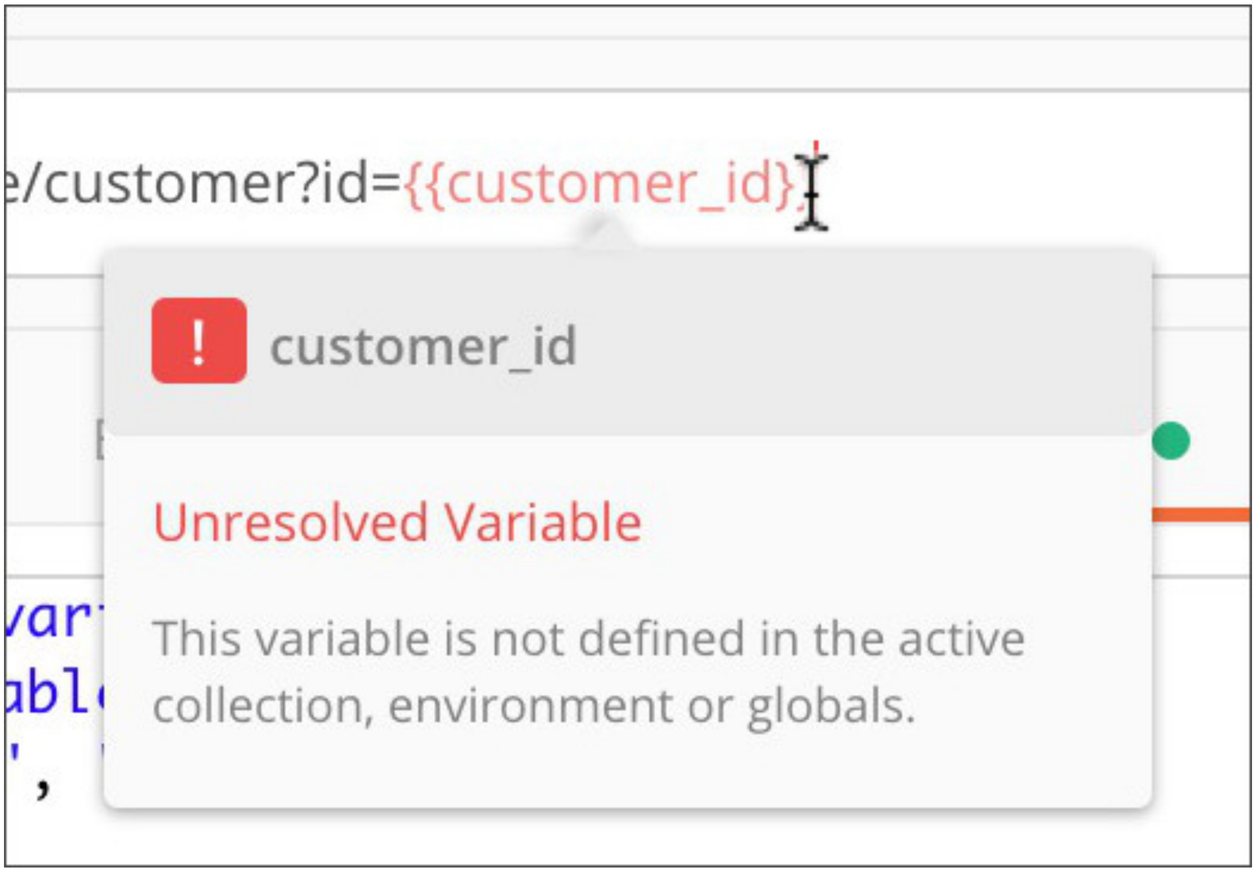
```
pm.globals.set("variable_key", "variable_val
pm.collectionVariables.set("variable_key", "
pm.environment.set("variable_key", "variable

console.log(pm.variables.get("variable_key"));
```

INITIAL ~~variable_value~~
CURRENT ~~variable_value~~ OVERRIDDEN
SCOPE Global

This also exists as an environment variable. Environment variables overwrite global variables.

If a variable is unresolved, Postman will highlight it in red.

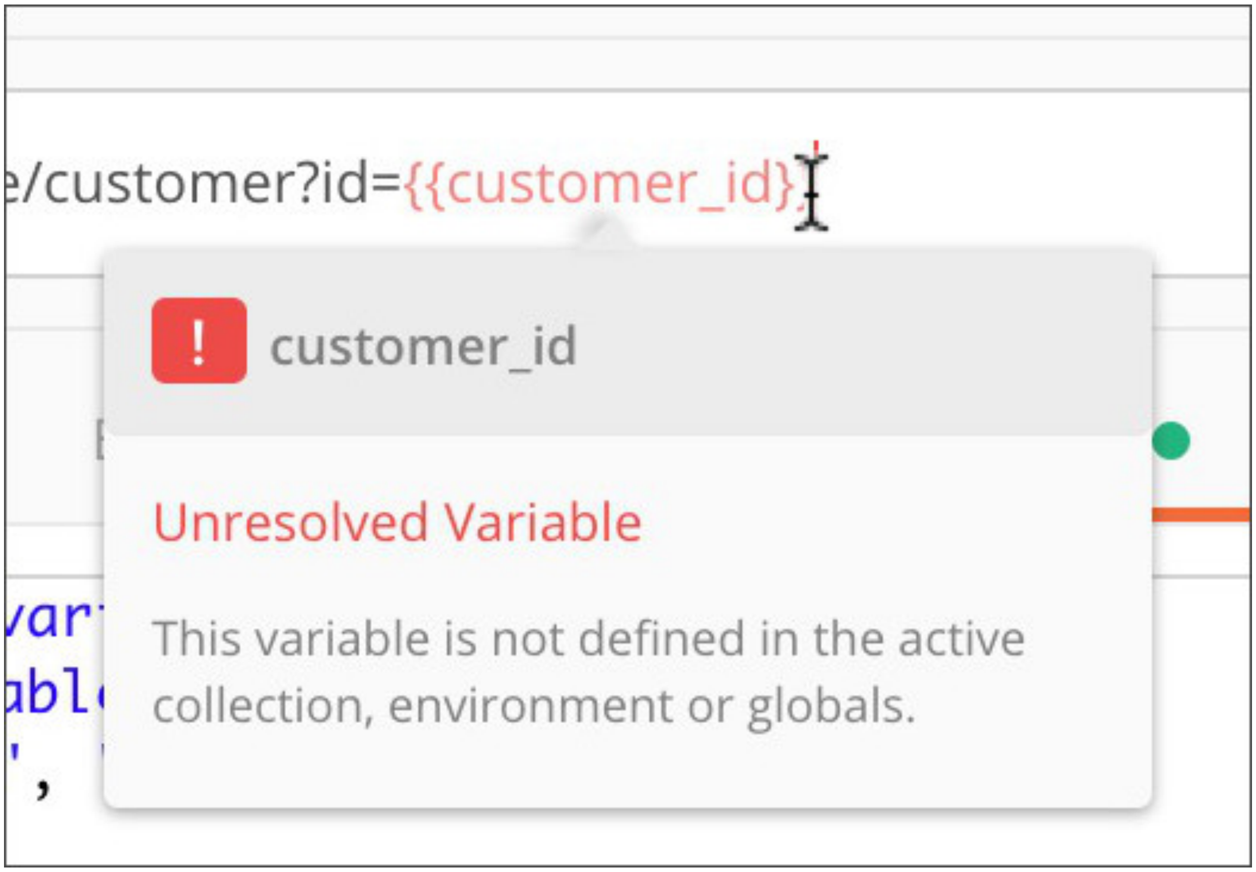


e/customer?id={{customer_id}}

customer_id

Unresolved Variable

This variable is not defined in the active collection, environment or globals.


e/customer?id={{customer_id}}

customer_id

Unresolved Variable

This variable is not defined in the active collection, environment or globals.

Using variables in scripts

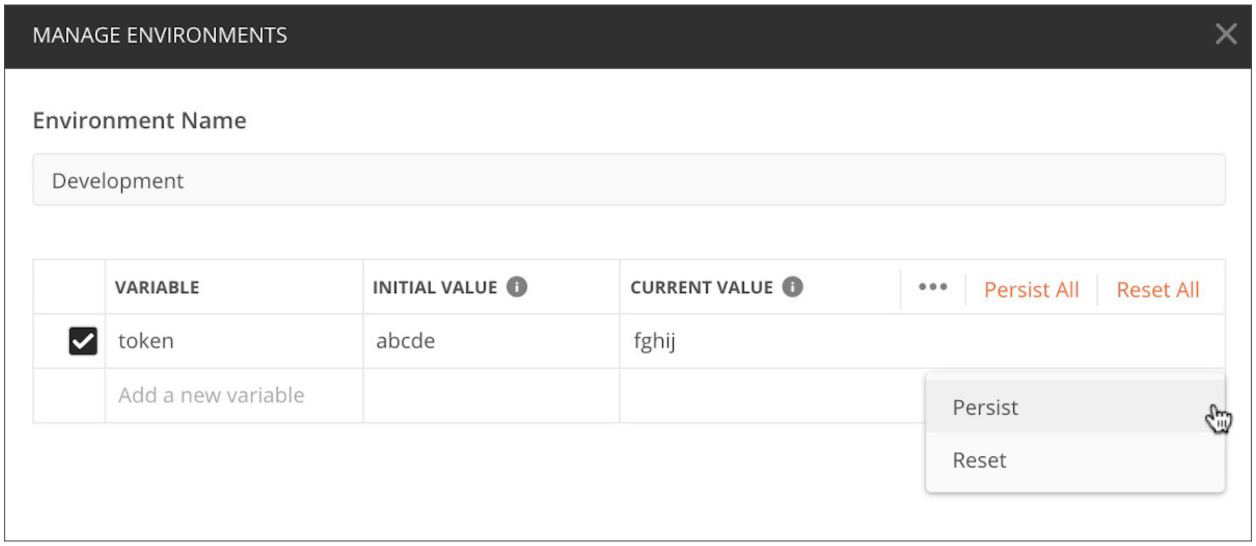
You can retrieve the current value of a variable in your scripts using the object representing the scope level and the `.get` method:

```
//access a variable at any scope including local
pm.variables.get("variable_key");
//access a global variable
pm.globals.get("variable_key");
//access a collection variable
pm.collectionVariables.get("variable_key");
//access an environment variable
pm.environment.get("variable_key");
```

Using `pm.variables.get()` to access variables in your scripts gives you the option to change variable scope without affecting your script functionality. This method will return whatever variable currently has highest precedence (or narrowest scope).

Sessions in Postman

When you edit global, collection, and environment variables in Postman, you will see **Current Value**, **Persist**, and **Reset** options for individual variables and for all variables. These allow you to control what happens within your local instance of Postman, independently of how the data is synced with anyone you're sharing requests, collections, and environments with.


MANAGE ENVIRONMENTS

Environment Name

Development

	VARIABLE	INITIAL VALUE	CURRENT VALUE	••• Persist All Reset All
✓	token	abcde	fghij	
	Add a new variable			

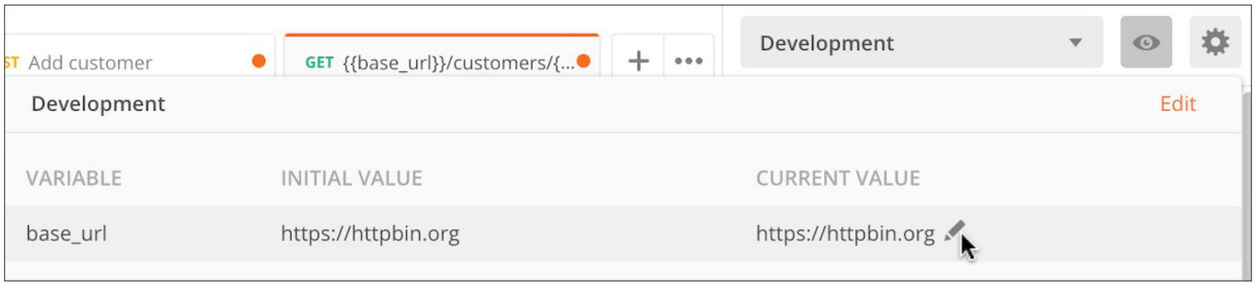
Your local session in Postman can use values that are transient and only visible to you. This lets you develop and test using private credentials or experimental values, without risk of exposing these details or affecting others on your team.

For example, your team could have a shared API key and individual API keys—you could do more experimental development work locally using your personal key, but use the shared key for team collaboration. Similarly, you could have a variable that represents exploratory work you're doing locally but are not ready to share with the team—you can later choose to persist the local data so that others on your team can also access it.

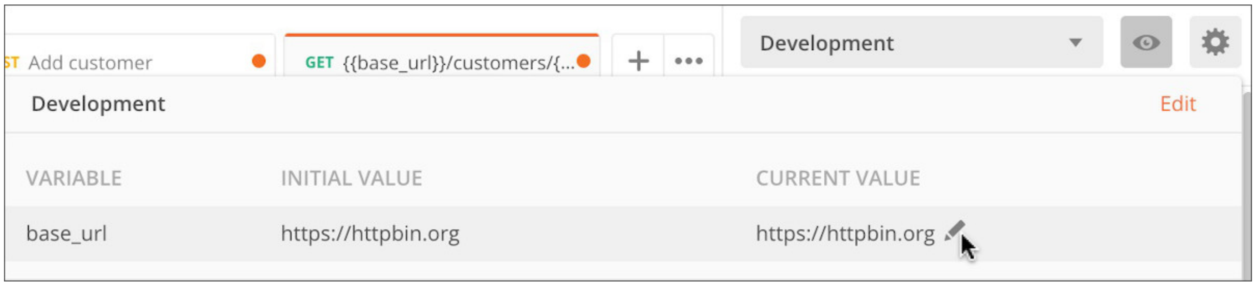
When you create or edit a variable, you can enter both an initial and a current value. You can choose to leave the current value empty, in which case it will default to the initial value. If you specify a current value, it will be local only to your instance—the **Persist** option lets you push your current value to the shared data, updating the initial value to match the current value.

Using **Persist** will make your current value sync with Postman's servers and be reflected for anyone sharing your collection or environment. To reset your current local values to reflect the initial (shared) values, use **Reset**.

You can edit a current value inline from the Environment quick look:



VARIABLE	INITIAL VALUE	CURRENT VALUE
base_url	https://httpbin.org	https://httpbin.org



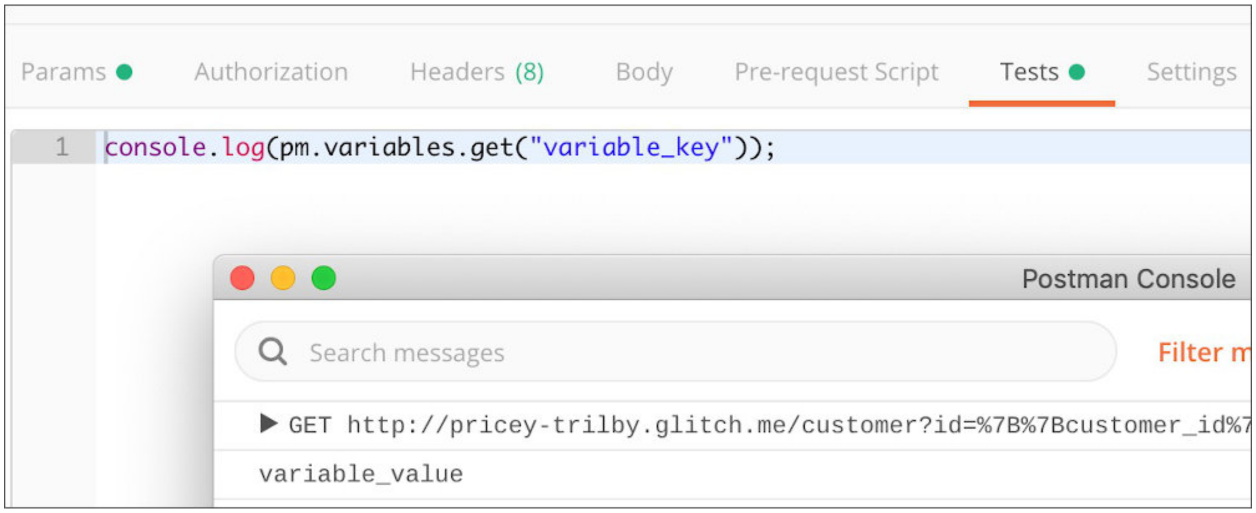
VARIABLE	INITIAL VALUE	CURRENT VALUE
base_url	https://httpbin.org	https://httpbin.org

Local and data variables only have current values, which do not persist beyond request or collection runs.

Logging variables

You can log variable values to the   while your requests run. Open the console from the button on the bottom left of Postman, or from the **View** menu. To log the value of a variable, use the following syntax in your script:

```
console.log(pm.variables.get("variable_key"));
```



```
console.log(pm.variables.get("variable_key"));
```

```
▶ GET http://pricey-trilby.glitch.me/customer?id=%7B%7Bcustomer_id%7
variable_value
```

```
console.log(pm.variables.get("variable_key"));
```

```
▶ GET http://pricey-trilby.glitch.me/customer?id=%7B%7Bcustomer_id%7
variable_value
```

Using data variables

The Collection Runner lets you import a CSV or a JSON file, and use the values from the data file inside requests and scripts. You cannot set a data variable inside Postman because it is pulled from the data file, but you can access data variables inside scripts, for example using `pm.iterationData.get("variable_name")`.

See ______________________ and the ______________________ for more.

Using dynamic variables

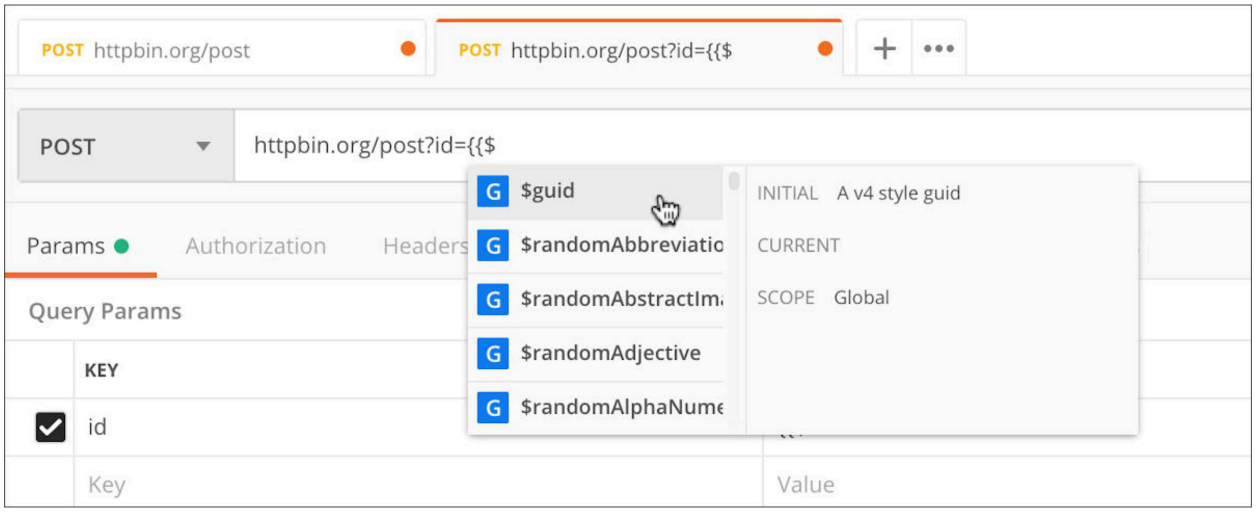
Postman provides dynamic variables that you can use in your requests.

Examples of dynamic variables are as follows:

- `{{$guid}}` : A v4 style guid
- `{{$timestamp}}`: The current timestamp (Unix timestamp in seconds)
- `{{$randomInt}}`: A random integer between 0 and 1000

See the section for a full list.

To use dynamic variables in pre-request or test scripts, you need to use `pm.variables.replaceIn()`, e.g. `pm.variables.replaceIn('{{$randomFirstName}}')`.



Next steps

Check out for more on using variables in your request scripting, and for more on how you can use data between requests.

Prerequisites

Additional Resources

VIDEOS

RELATED BLOG POSTS

Next Steps

PRODUCT

RESOURCES

USE CASES

PRICING

SUPPORT

[redacted]

COMPANY

[redacted]

[redacted]

[redacted]

Using variables

Variables allow you to store and reuse values in your requests and scripts. By storing a value in a variable, you can reference it throughout your collections, environments, and requests—and if you need to update the value, you only have to change it in one place. Using variables increases your ability to work efficiently and minimizes the likelihood of error.

Variables quick start

To try out a variable, use the following steps:

- Click the **Environment quick look** (eye button) in the top right of Postman and click **Edit** next to **Globals**.
- Add a variable named `my_variable` and give it an initial value of `Hello`—click **Save** and close the environment modal.

- Open a new request tab and enter `https://postman-echo.com/get?var={{my_variable}}` as the URL. Hover over the variable name and you'll see the value.
- **Send** the request. In the response, you'll see that Postman sent the variable value to the API. *Try changing the value in the Environment quick look and sending the request again*.

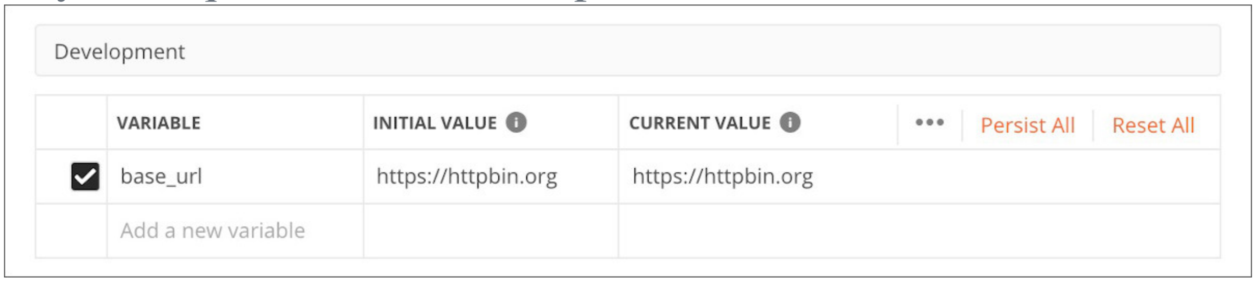
Read on for more detail on how you can use variables in Postman.

Contents

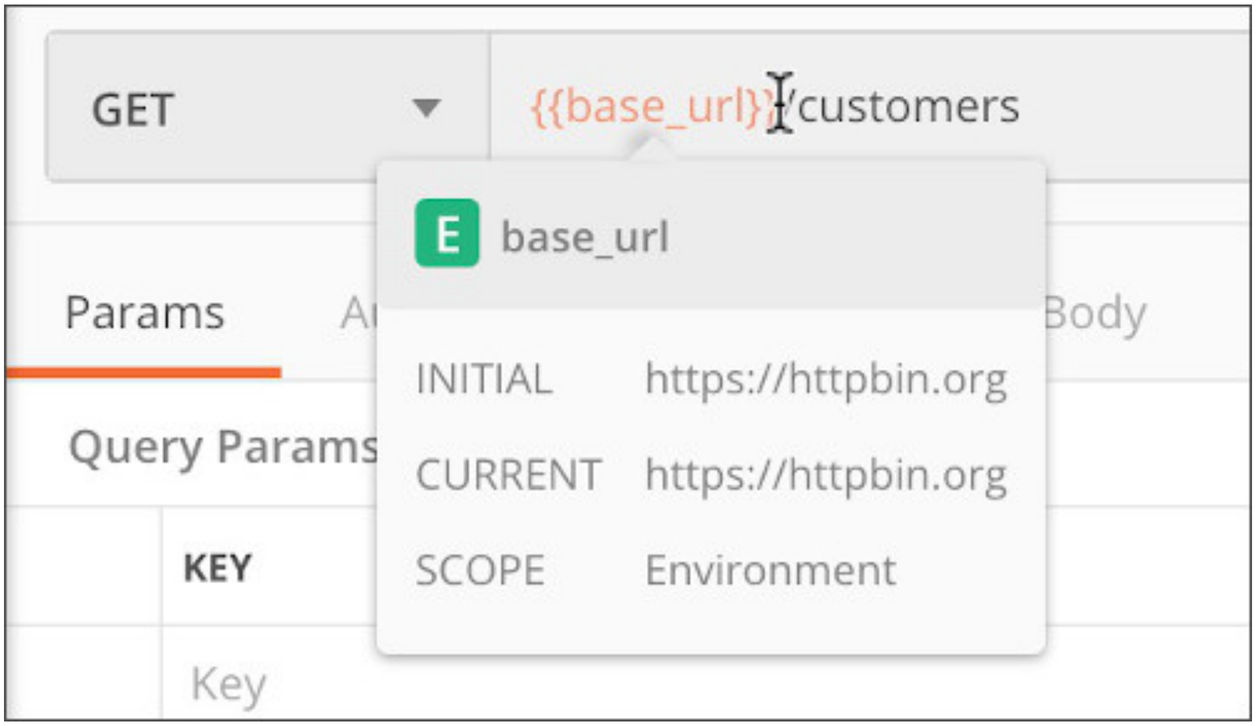
Understanding variables and environments

A variable is a symbolic representation of data that allows you to access a value without having to enter it manually wherever you need it. This can be useful especially if you are using the same values in multiple places. Variables make your requests more flexible and readable, by abstracting some of the detail involved.

For example, if you have the same URL in multiple requests—but the URL might change—you can store it in a variable. If the URL changes, you only need to change the variable value and it will be reflected throughout your collection, wherever you've used the variable name. The same principle applies to any part of your request code that is repeated.

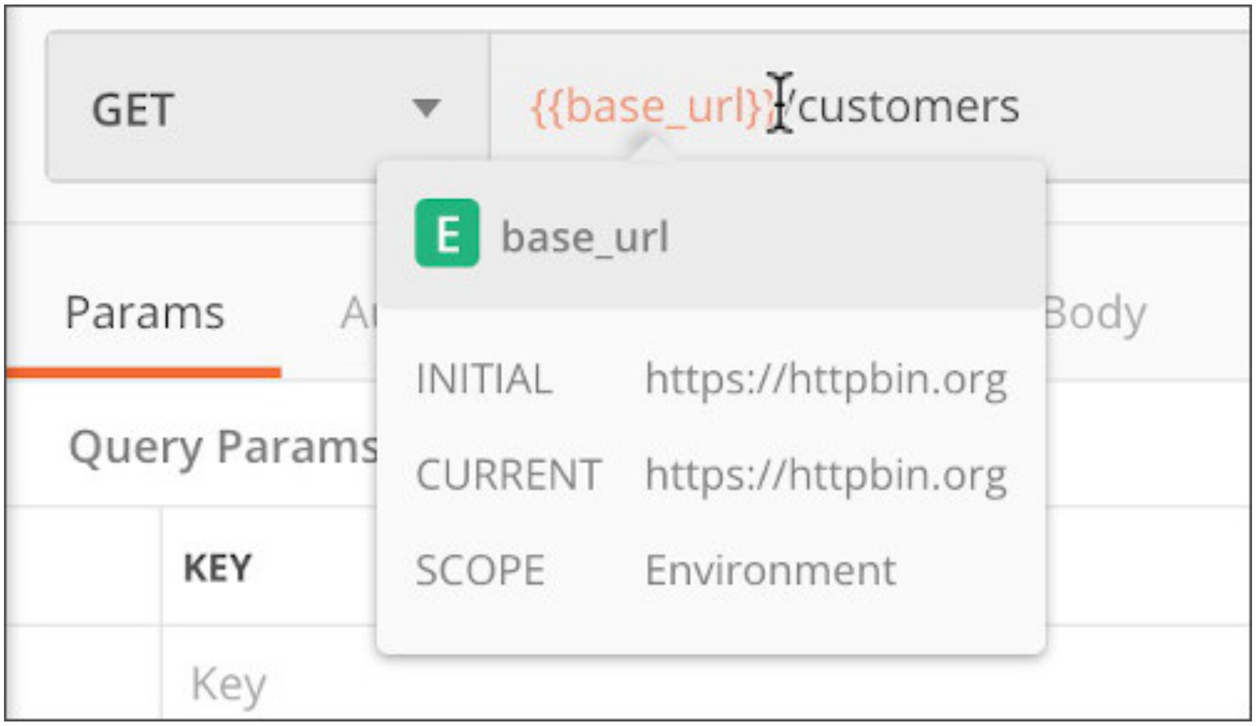

Development

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	base_url	https://httpbin.org	https://httpbin.org
	Add a new variable		


GET {{base_url}}/customers

base_url

INITIAL	https://httpbin.org
CURRENT	https://httpbin.org
SCOPE	Environment


GET {{base_url}}/customers

base_url

INITIAL https://httpbin.org
CURRENT https://httpbin.org
SCOPE Environment

Postman supports variables at different scopes, allowing you to tailor your processing to a variety of development, testing, and collaboration tasks. Scopes in Postman relate to the different contexts that your requests run in—within the Postman app, in

collections, in environments, and in Newman / the Collection Runner.
Postman will store environment and global variables as strings. If you’re storing objects or arrays, remember to `JSON.stringify()` them before storing, and `JSON.parse()` them when you retrieve them.

Environments in Postman

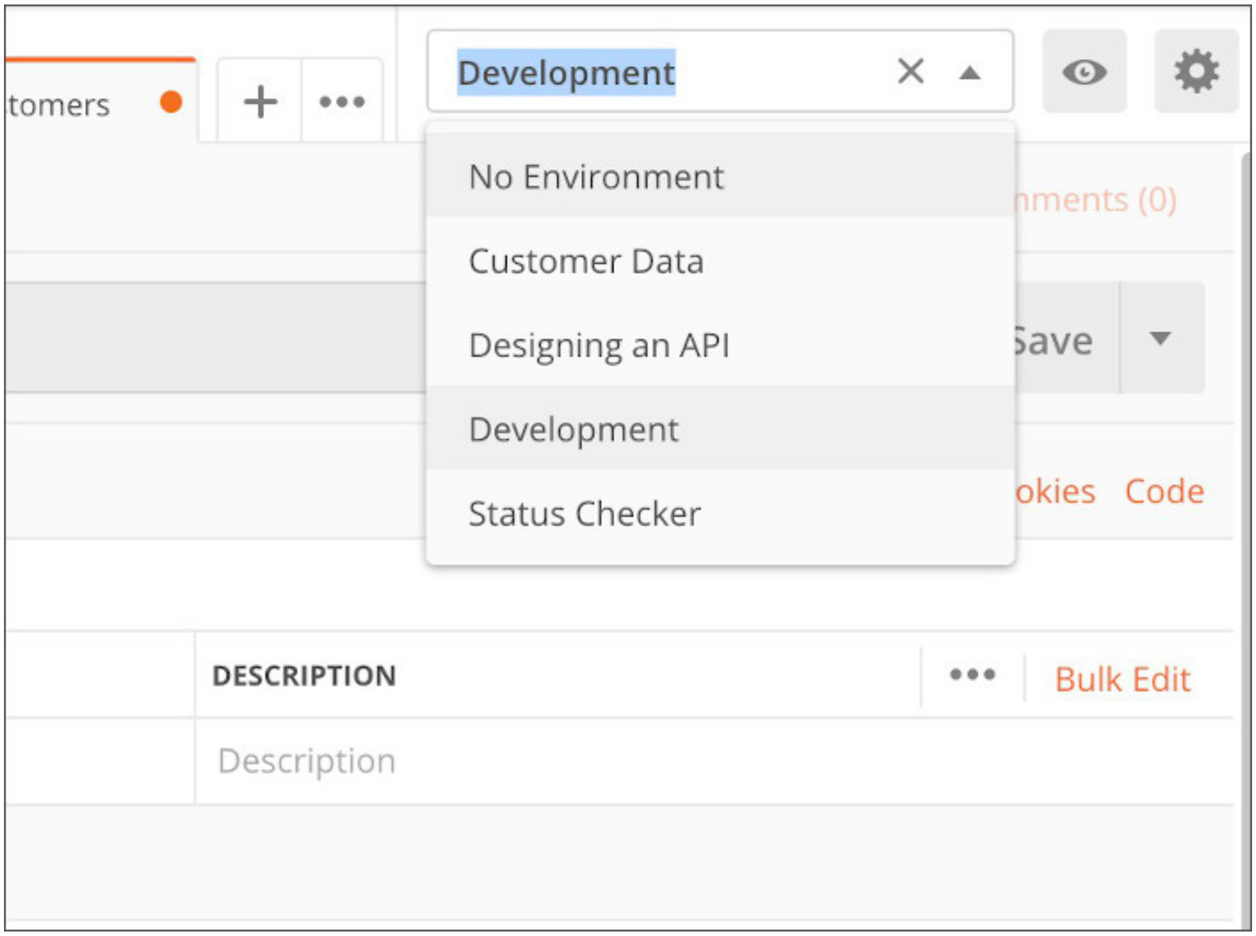
Environments allow you to run requests and collections against different data sets. For example, you could have an environment for development, one for testing, and another for production. You can use variables to pass data between requests and tests, for example if you are chaining requests using a collection.

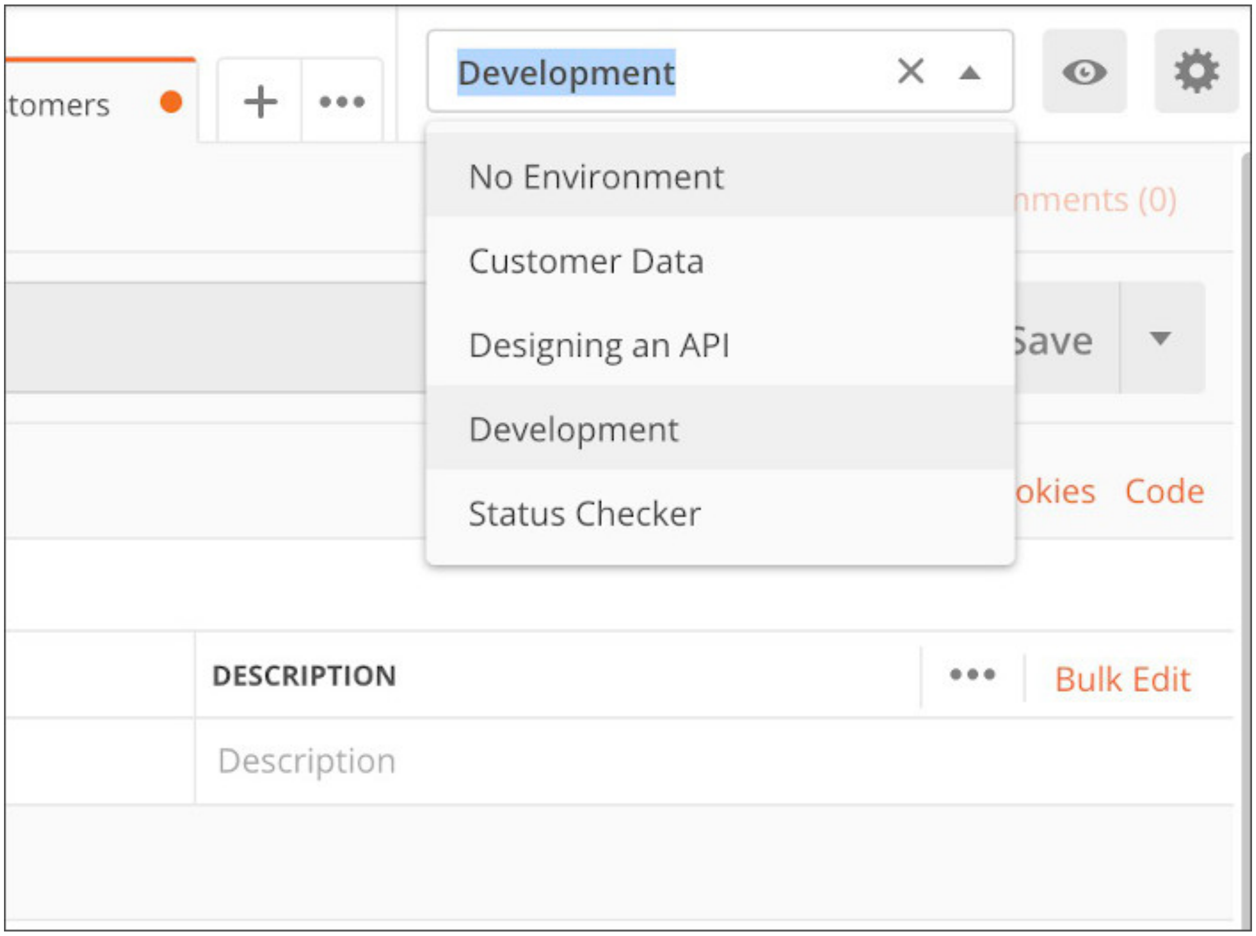
Environments in Postman are key-value pairs of variables. Each variable name represents its key, so referencing the variable name allows you to access its value.

For example, if you have a base URL for requests stored in a variable named `base_url`, you can reference it in your requests using `{{base_url}}`. Whatever value is stored in the variable will be included wherever you've referenced the variable when your requests run. If the base URL value is `https://httpbin.org`, and is listed as part of the request URL using `{{base_url}}/get?customers=new`, Postman will send the request to `https://httpbin.org/get?customers=new`.

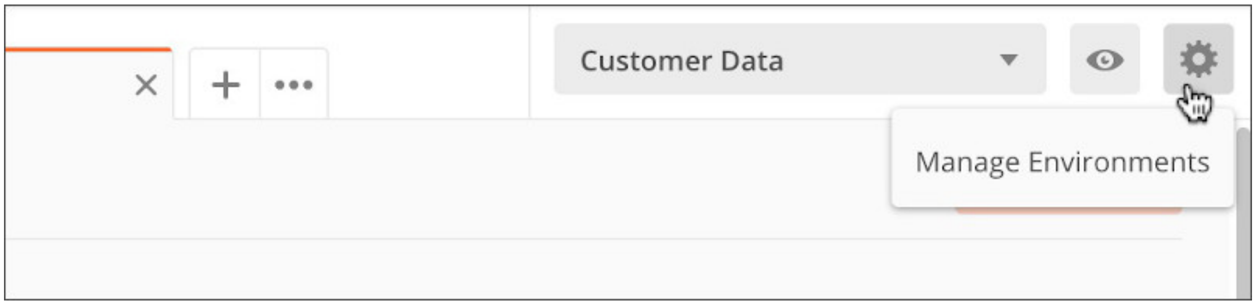
To create an environment, use **New > Environment**, or the **Manage environments** button in the top right of Postman, and click **Add**. Give your environment a name , such as "Testing"—you can add variables at creation or later, by editing the environment.

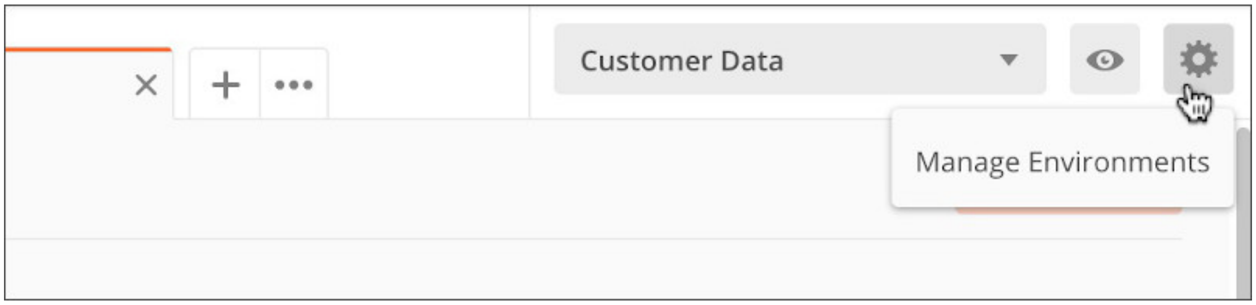
To select the environment to run your requests within, use the drop-down at the top right of the Postman app.

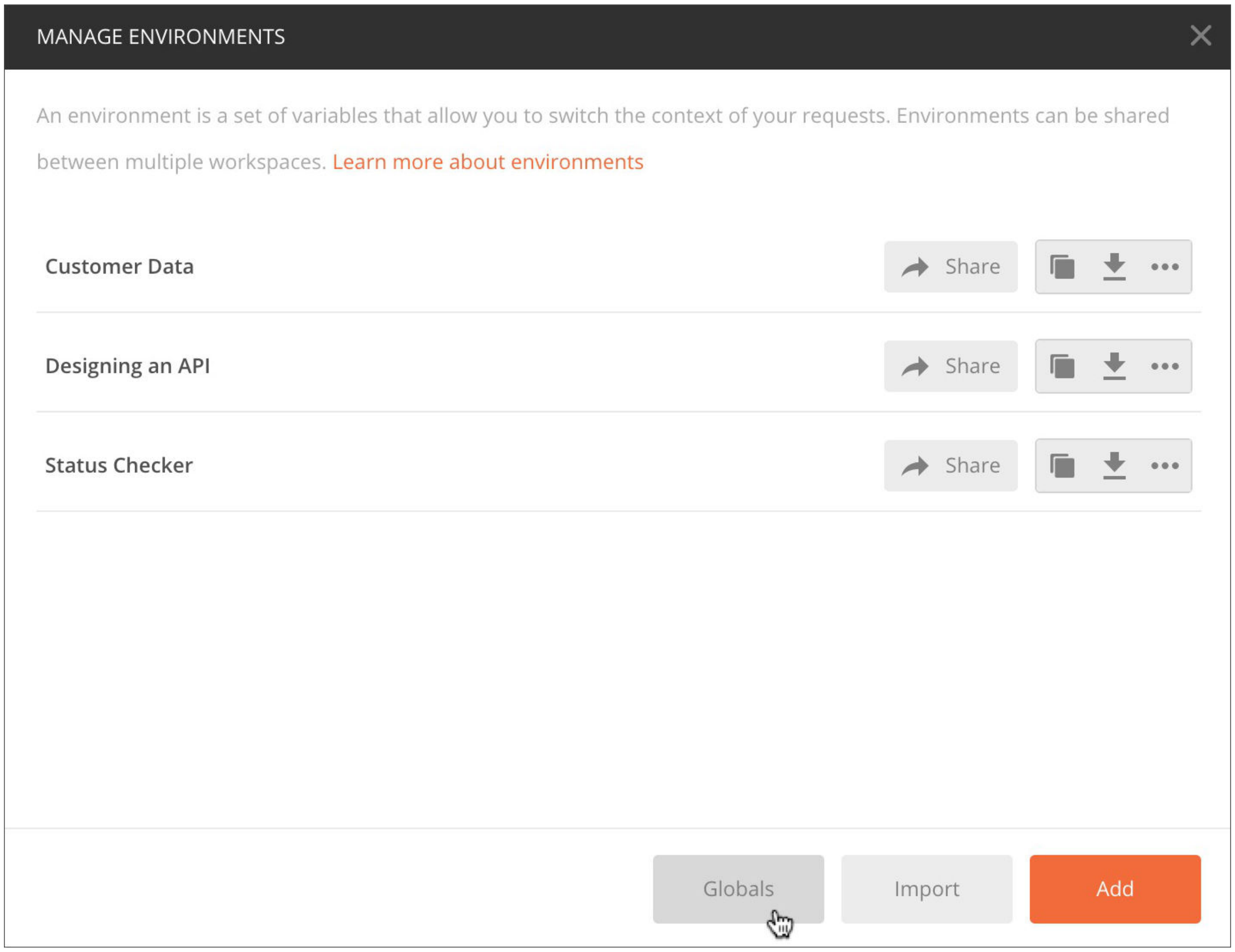

tomers
Development
No Environment
Customer Data
Designing an API
Development
Status Checker
nments (0)
Save
okies
Code
DESCRIPTION
Bulk Edit
Description



Your requests will run with the data variables listed in the environment you have selected.
You can duplicate, delete, download / import environment JSON, and share environments with collaborators in **Manage Environments**. Sharing environments allows you to let other people run your requests against the same data sets.






MANAGE ENVIRONMENTS

An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. Learn more about environments

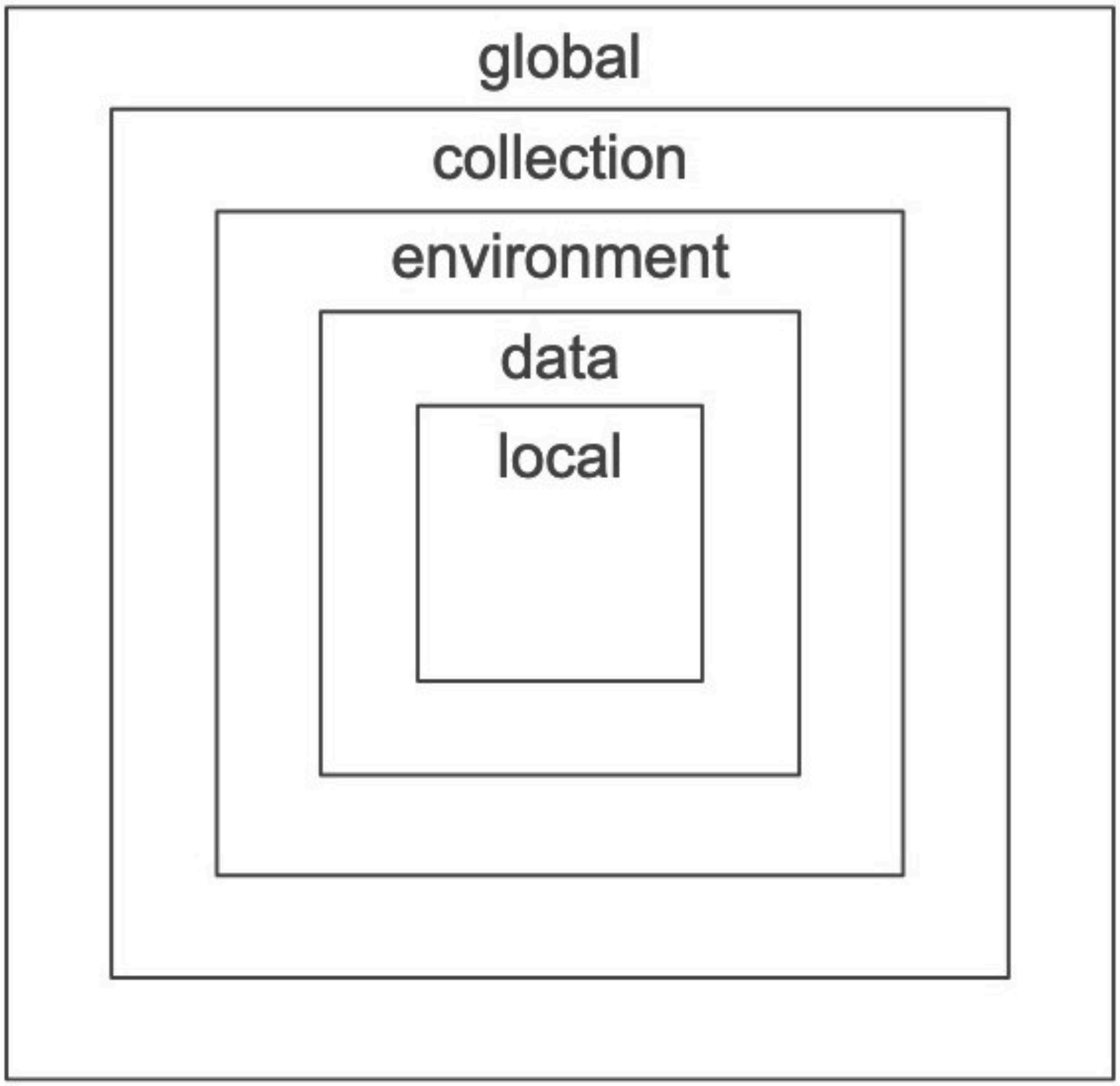
Customer Data	Share
Designing an API	Share
Status Checker	Share

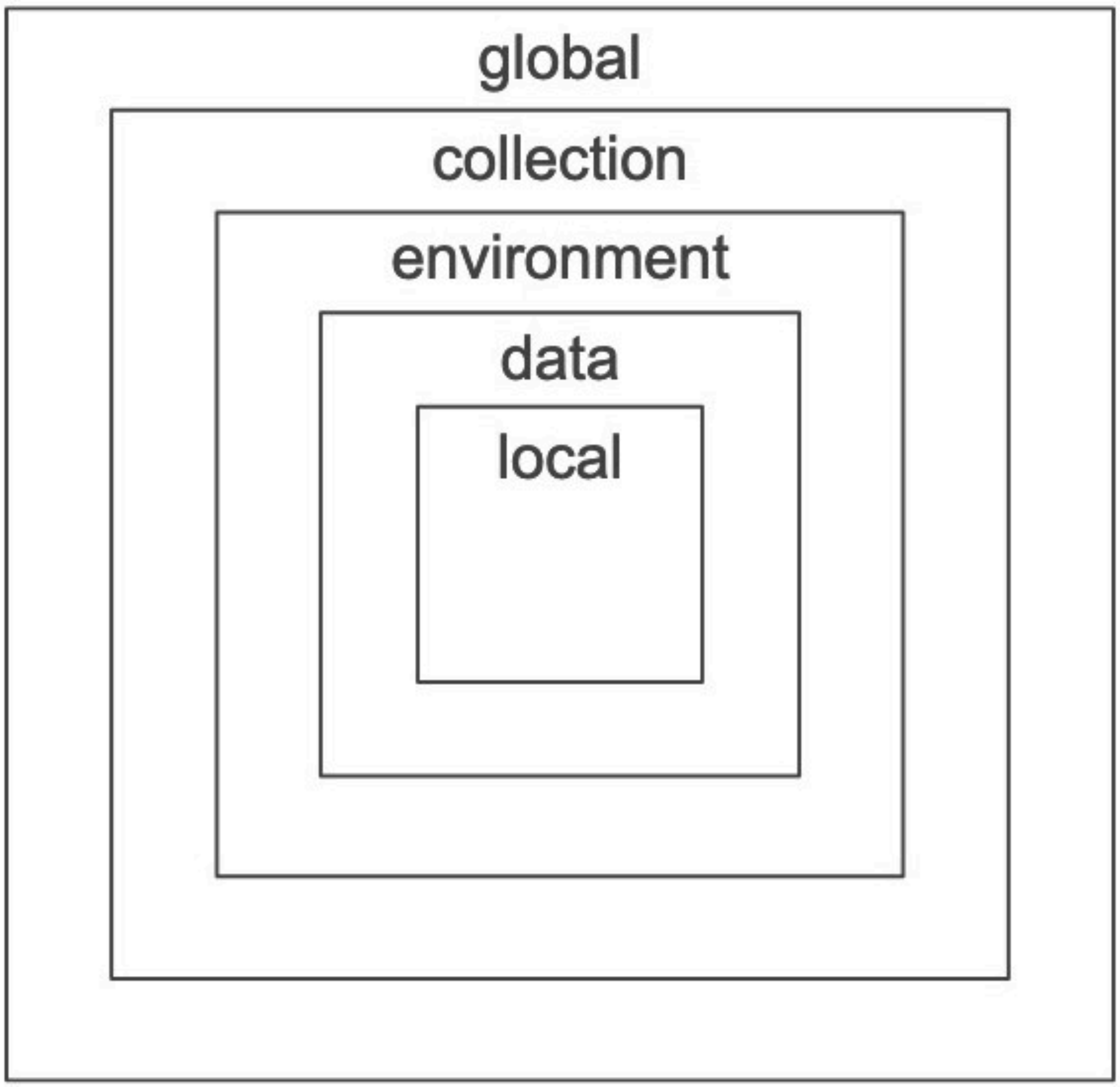
To share an environment safely, create a duplicate and remove any sensitive data such as auth values first. When your collaborator imports the environment they can enter their own credentials. Any changes you make to a shared environment will be reflected throughout the team with access to it.

Variable scopes

Postman supports the following variable scopes:

- Global
- Collection
- Environment
- Data
- Local


global
collection
environment
data
local



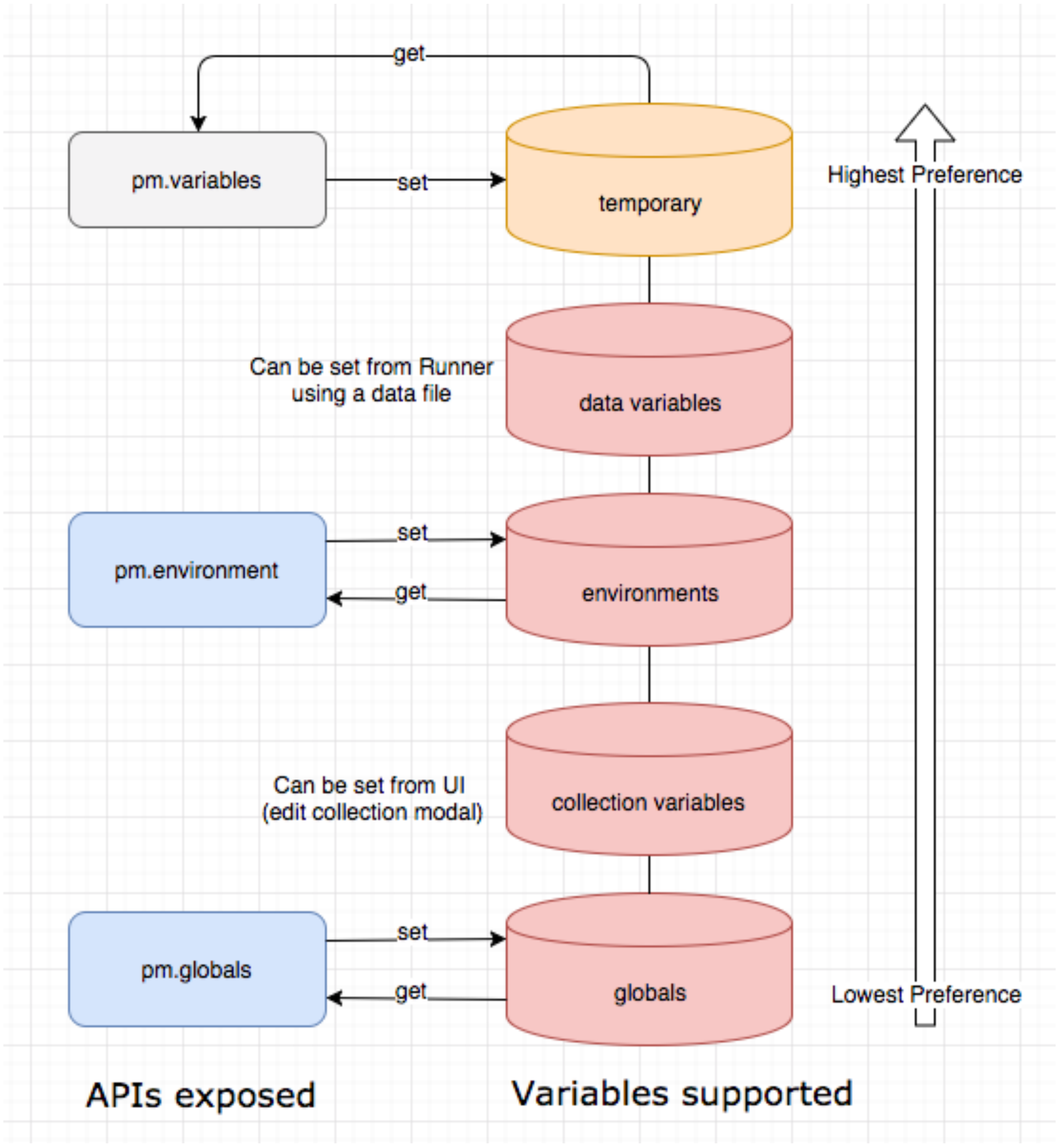
If a variable with the same name is declared in two different scopes, the value stored in the variable with narrowest scope will be used—for example if there is a global and a local variable both named `username`, the local value will be used when the request runs.

Choosing variables

Variable scopes are suited to different tasks in Postman:

- **Global variables** allow you to access data between collections, requests, test scripts, and environments.
    - *Since global variables can create confusion, you should only use them sparingly—for example to quickly test something or when your project is at a very early prototyping stage.*
- **Collection variables** are available throughout the requests in a collection and are independent of environments, so do not change based on the selected environment.
    - *Collection variables are suitable if you are only using a single environment, for example for auth / URL details.*
- **Environment variables** allow you to tailor your processing to different environments, for example local development vs testing or production. Only one environment can be active at a time.
    - *If you only have one environment, using collection variables is more efficient.*
- **Local variables** are temporary, and only accessible in your request scripts. Local variable values are scoped to a single request or collection run, and are no longer available when the run is complete.

    - *Local variables are suitable if you need a value to override all other variable scopes but do not want the value to persist once execution has ended.*
- **Data variables** come from external CSV and JSON files to define data sets you can use when running collections via Newman or the Collection Runner.



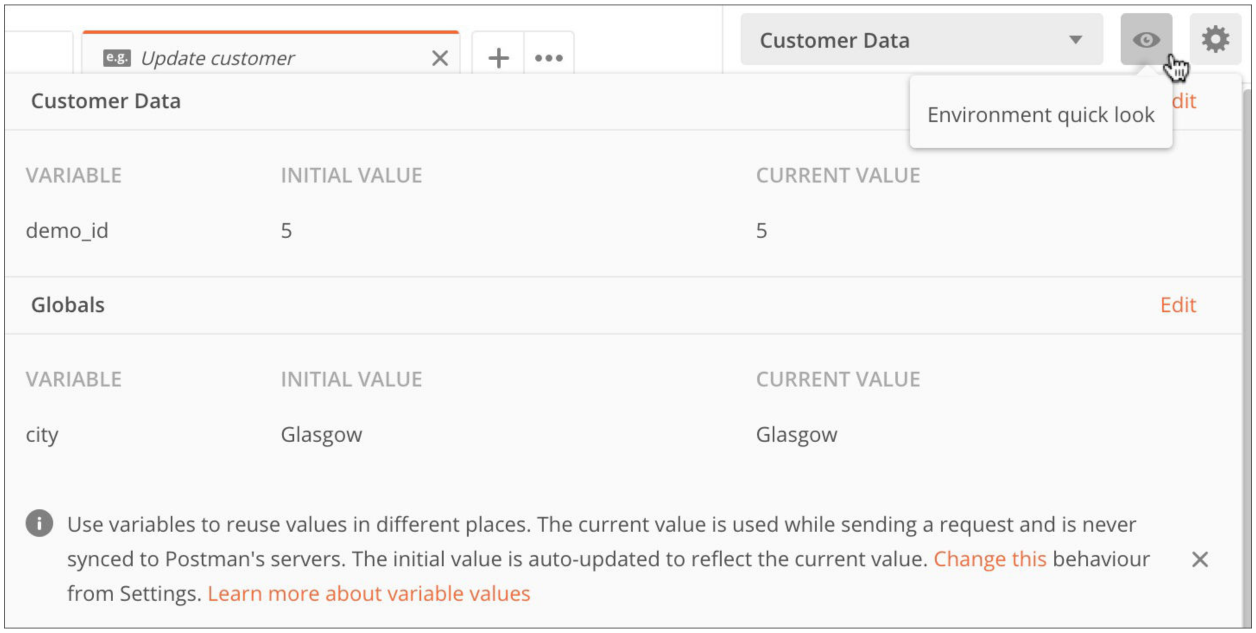
Defining variables

You can define variables in a variety of ways, depending on whether you need , or scope.

Remember to delete variables you are no longer using.

Defining global and environment variables

To create or edit a variable at global or environment scope, use the **Environment quick Look** and click **Edit** next to the environment or global variable.



Customer Data

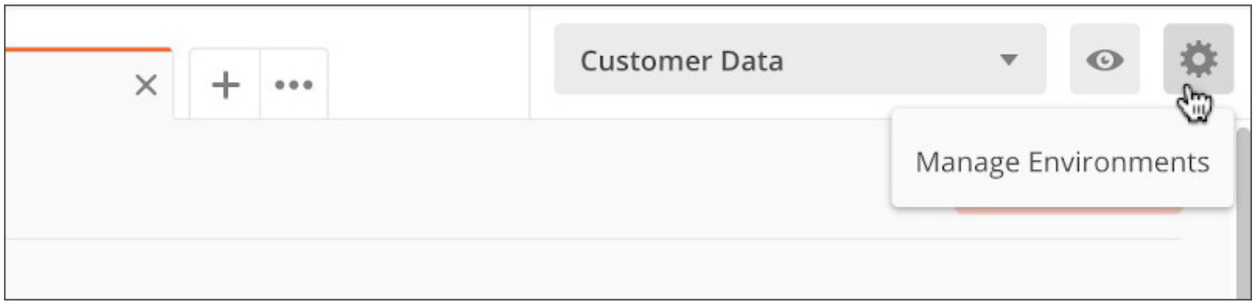
VARIABLE	INITIAL VALUE	CURRENT VALUE
demo_id	5	5

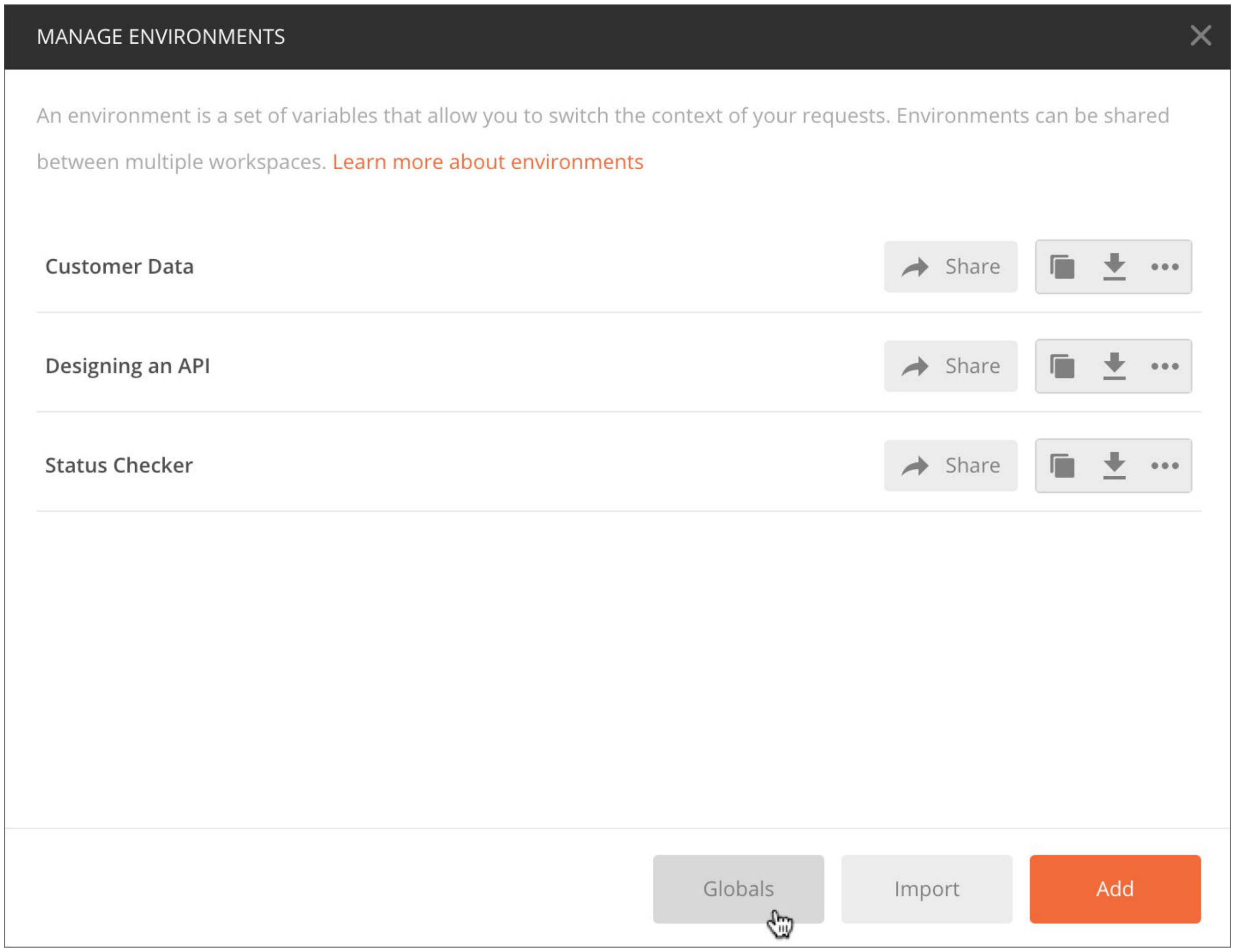
Globals

VARIABLE	INITIAL VALUE	CURRENT VALUE
city	Glasgow	Glasgow

Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. Change this behaviour from Settings. Learn more about variable values

The quick look view provides a reference to the current state of any global or environment variables you're working with—you can also access the quick look using the keyboard shortcut `CMD/CTRL + ALT + E`.
Alternatively, click **Manage environments** and select the relevant environment, or click **Globals**.




MANAGE ENVIRONMENTS

An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. Learn more about environments

Customer Data

Designing an API

Status Checker

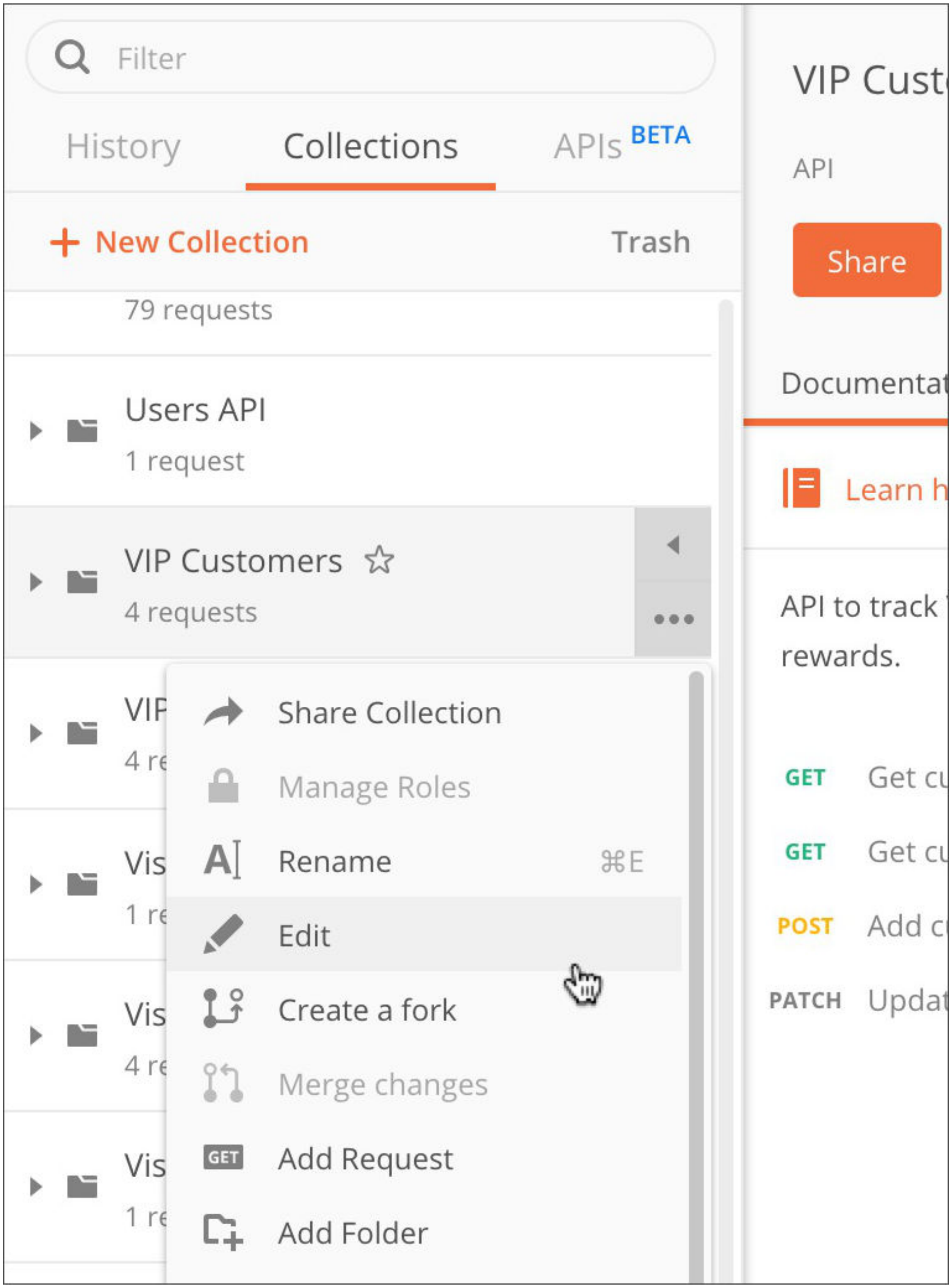
You can if you do not already have one.

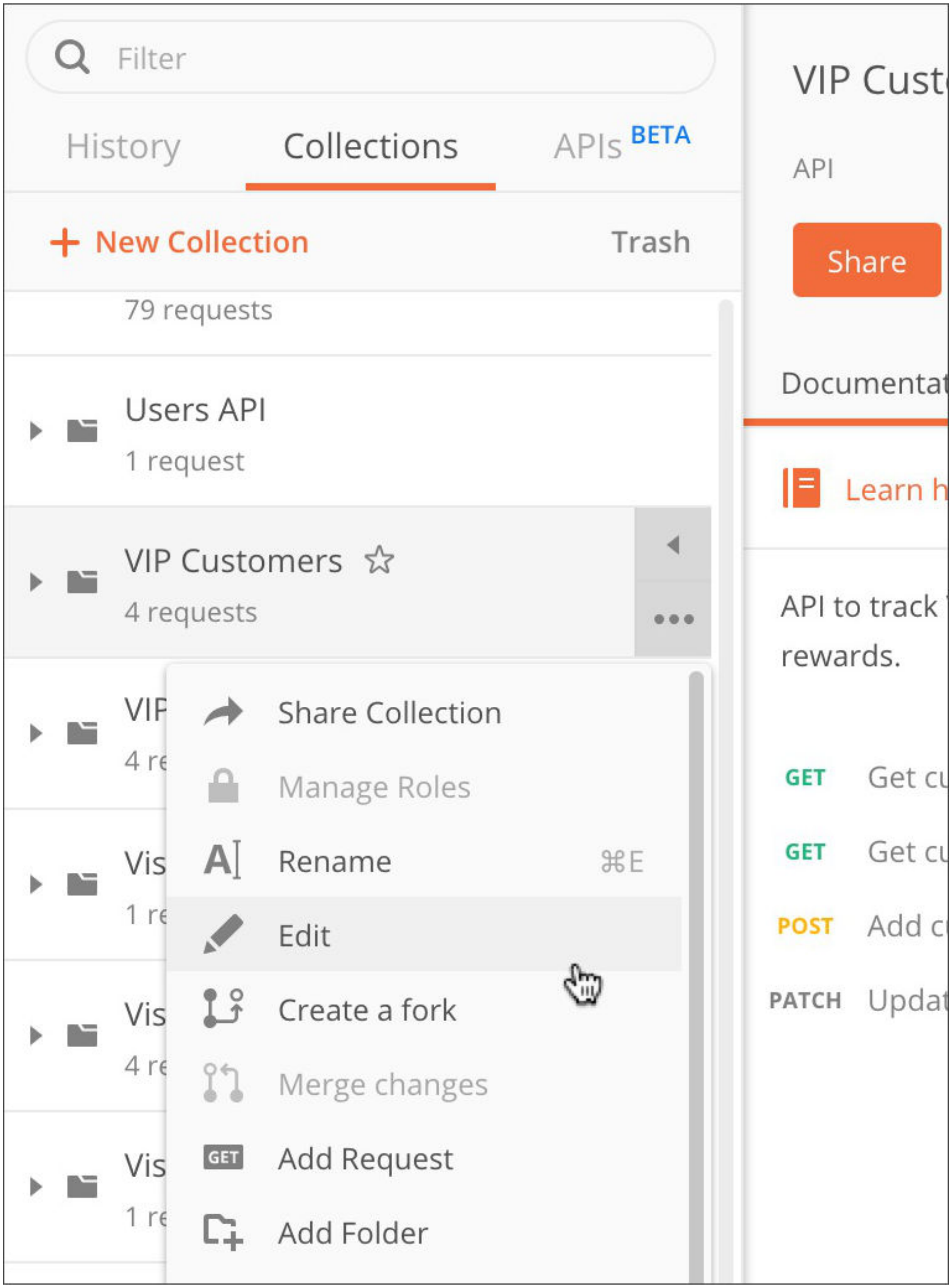
Once you have your scope selected, you can
.

You can also ____.

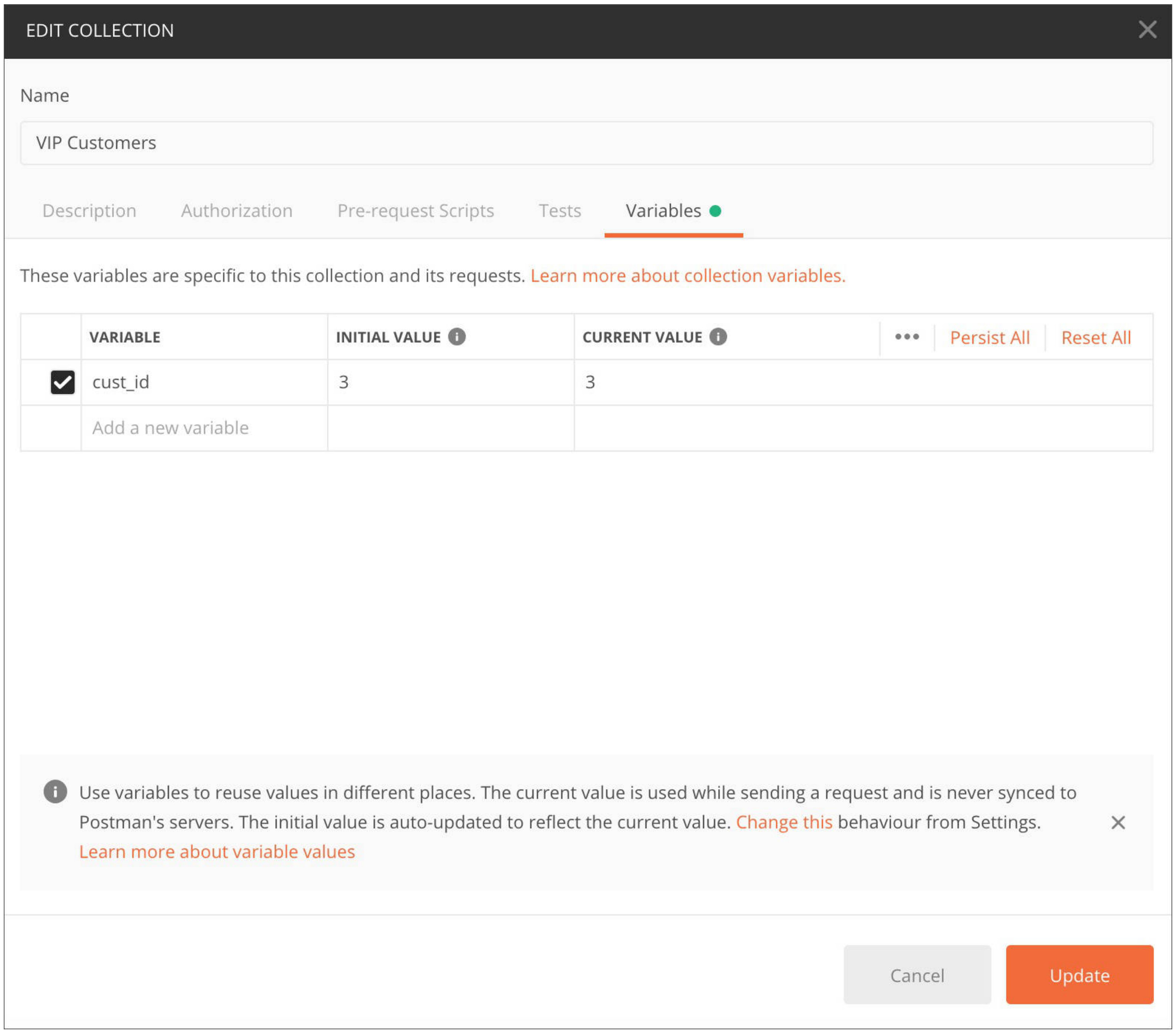
Defining collection variables

To create or edit a variable for an existing collection, select the collection in **Collections** on the left of the Postman app, open the **View more actions** (**...**) menu, and click **Edit**.


Filter
History
Collections
APIs BETA
+ New Collection
Trash
79 requests
Users API
1 request
VIP Customers
4 requests
VIP
4 re
Vis
1 re
Vis
4 re
Vis
1 re
Share Collection
Manage Roles
Rename ⌘E
Edit
Create a fork
Merge changes
Add Request
Add Folder
VIP Cust
API
Share
Documentat
Learn h
API to track
rewards.
GET Get cu
GET Get cu
POST Add c
PATCH Updat


Filter
History
Collections
APIs BETA
+ New Collection
Trash
79 requests
Users API
1 request
VIP Customers
4 requests
VIP
4 re
Vis
1 re
Vis
4 re
Vis
1 re
Share Collection
Manage Roles
Rename ⌘E
Edit
Create a fork
Merge changes
GET Add Request
Add Folder
VIP Cust
API
Share
Documentat
Learn h
API to track
rewards.
GET Get cu
GET Get cu
POST Add c
PATCH Updat

Choose the **Variables** tab to edit or add to your collection variables.


EDIT COLLECTION

Name

VIP Customers

These variables are specific to this collection and its requests. Learn more about collection variables.

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	cust_id	3	3
	Add a new variable		

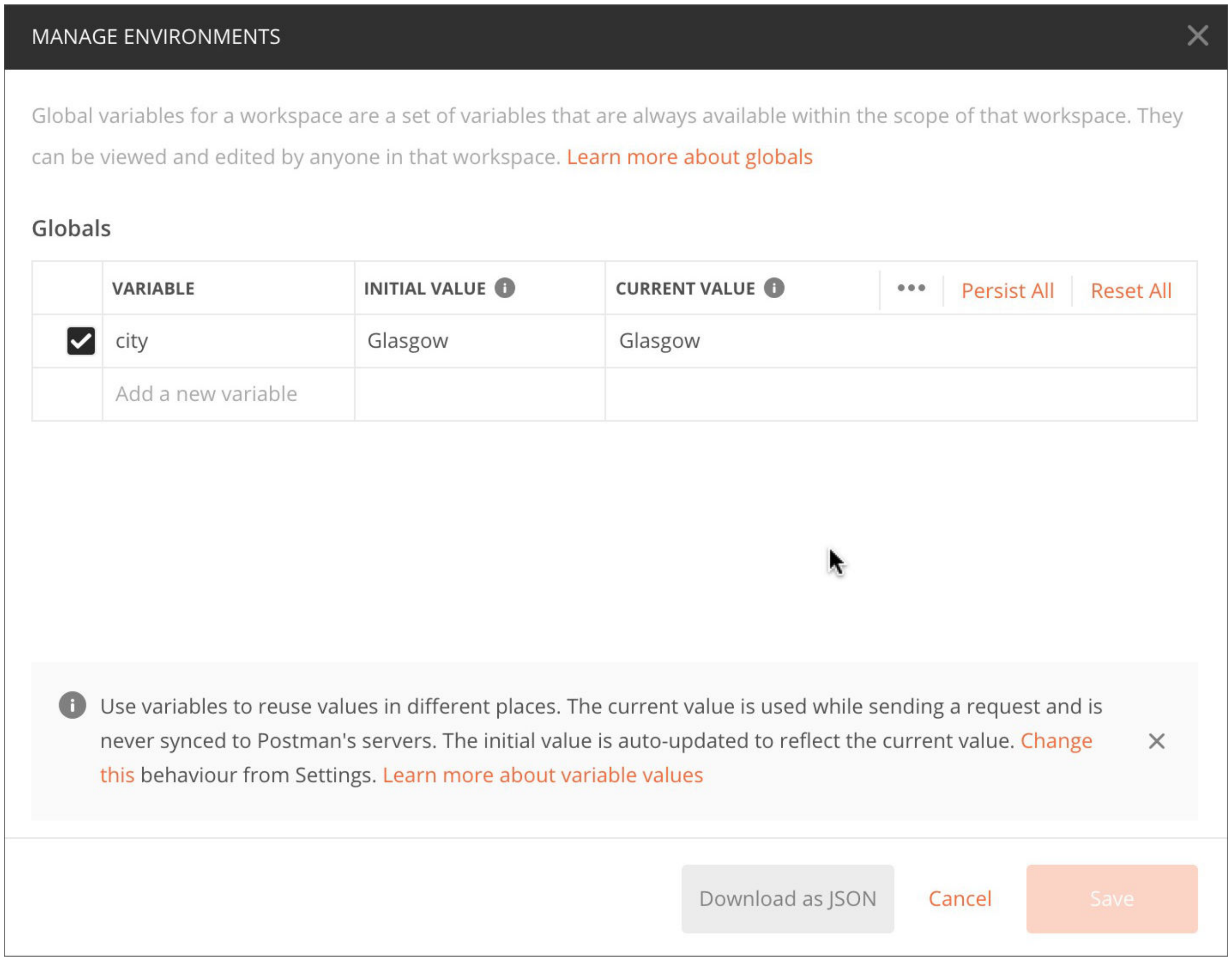
Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. Change this behaviour from Settings. Learn more about variable values

You can add collection variables when you create a new collection.

You can also ________________________________.

Specifying variable detail

You can add and edit variables at any time. All you need to include for a new variable is a name—you can choose to supply an initial value but can alternatively set it later, including from scripts. Use the checkbox to enable or disable a variable.


MANAGE ENVIRONMENTS

Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. Learn more about globals

Globals

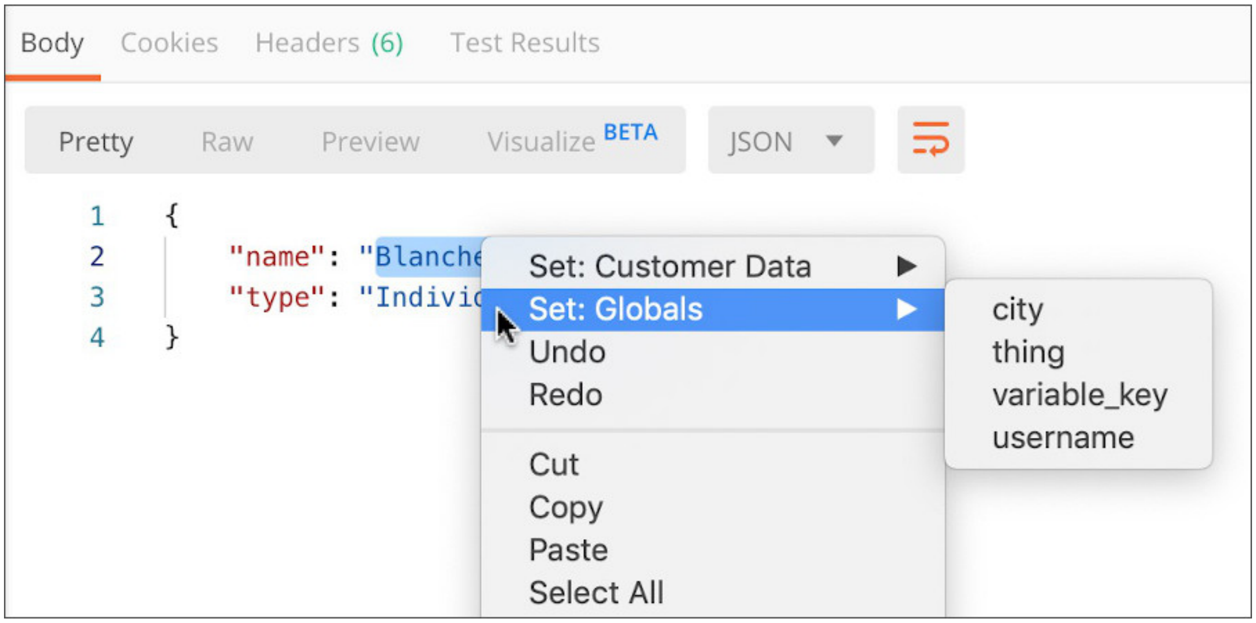
	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	city	Glasgow	Glasgow
	Add a new variable		

Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. Change this behaviour from Settings. Learn more about variable values

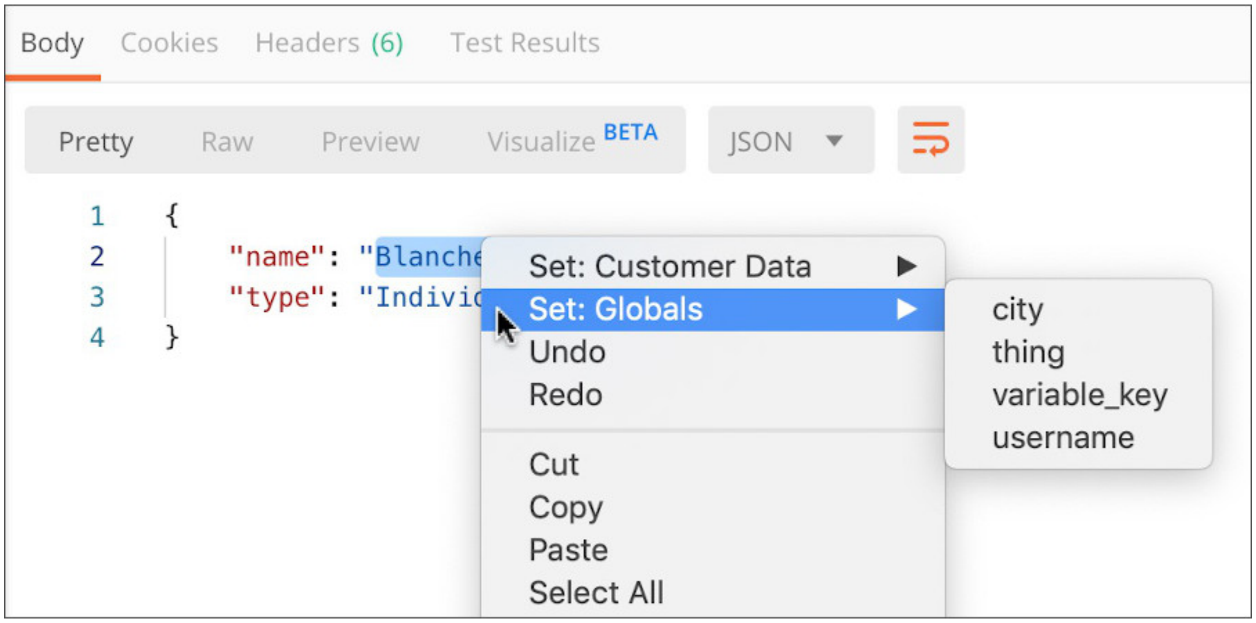
Initial values are shared when you share a collection or environment. Current values are local and not synced or shared. See for more on local vs synced variables.

You can download global variables as JSON from **Manage Environments**.

You can set response body values to variables by selecting text, right-clicking / CTRL + clicking, and choosing the relevant variable by name.



```
{
    "name": "Blanche
    "type": "Indivic
}
```



```
{
    "name": "Blanche
    "type": "Indivic
}
```

Defining variables in scripts

You can set variables programmatically in your request scripts.

Use   to define a global variable:

```
pm.globals.set("variable_key", "variable_value");
```

Use to define a collection variable:

```
pm.collectionVariables.set("variable_key",
"variable_value");
```

Use to define an environment variable (in the currently selected environment):

```
pm.environment.set("variable_key", "variable_value");
```

Check out the for more on scripting with variables.

Defining local variables

Local variables are temporary values you set in your request scripts using the following syntax:

```
pm.variables.set("variable_key", "variable_value");
```

Local variables do not persist between sessions, but allow you to override all other scopes temporarily, during the execution of a request or collection / monitor run. For example, if you need to process a temporary test value for a single request or collection run locally, and don't want the value to sync with your team or remain available when the request / collection has finished running, you can use a local variable.

Accessing variables

You can use double curly braces to reference variables throughout the Postman app user interface. For example, to reference a variable named "username" in your request auth settings, you could use the following syntax with double curly braces around the name:

```
{{username}}
```

When you run a request, Postman will resolve the variable and replace it with it's current value.

For example, you could have a request URL referencing a variable as follows:

```
http://pricey-trilby.glitch.me/customer?id={{cust_id}}
```

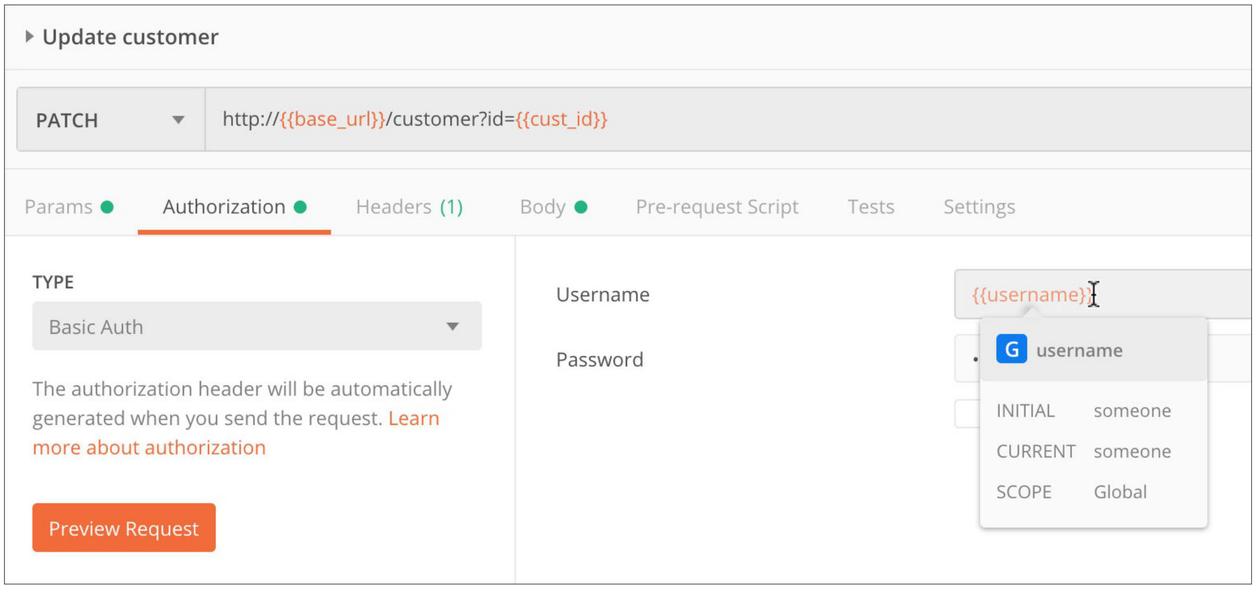
Postman will send whatever value you currently have stored for the `cust_id` variable when the request runs. If `cust_id` is currently `3`, the request will be sent to the following URL including query parameter:

```
http://pricey-trilby.glitch.me/customer?id=3
```

Alternatively, you could have a request body that accesses a variable by wrapping its reference in double-qoutes:

```
{ "customer_id" : "{{cust_id}}" }
```

You can use variables in request URLs, parameters, headers, authorization, body, and header presets.



PATCH http://{{base_url}}/customer?id={{cust_id}}

TYPE
Basic Auth

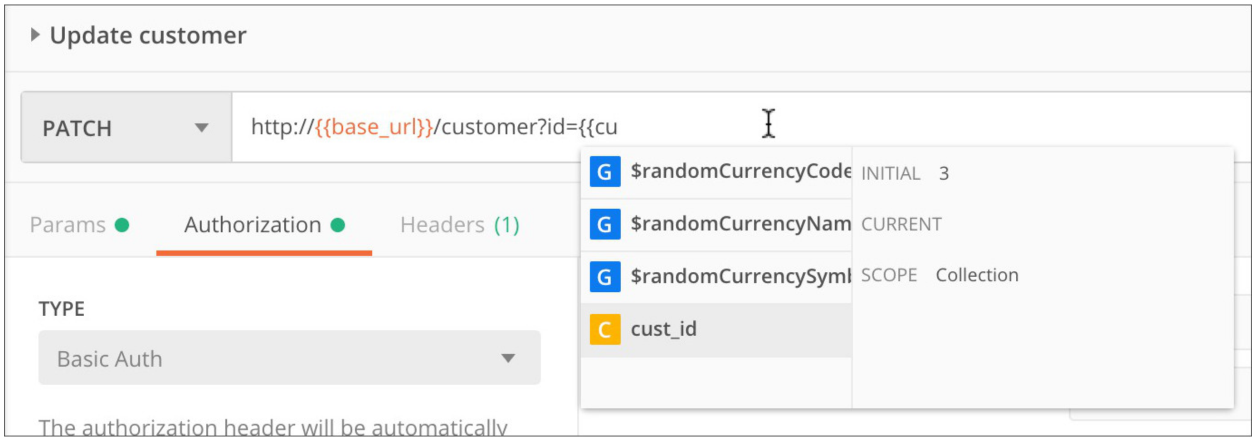
The authorization header will be automatically generated when you send the request. Learn more about authorization

Username {{username}}
Password

G username

INITIAL	someone
CURRENT	someone
SCOPE	Global

When you hover over a variable you can see an overview of its current status. As you type variables into your requests, Postman will prompt you with any that are currently defined.



PATCH http://{{base_url}}/customer?id={{cu

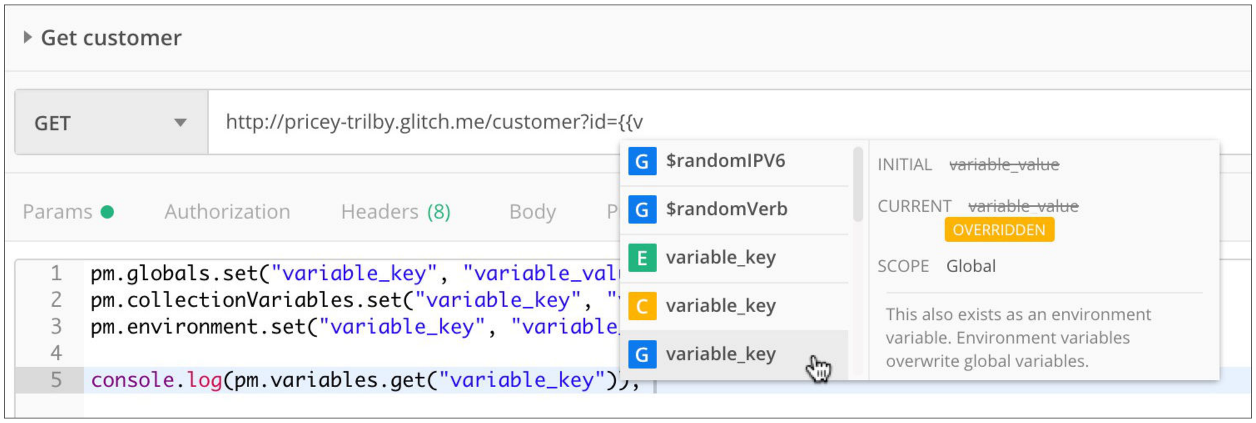
G $randomCurrencyCode
G $randomCurrencyNam
G $randomCurrencySymb
C cust_id

INITIAL	3
CURRENT	
SCOPE	Collection

TYPE
Basic Auth

The authorization header will be automatically

The prompt will indicate current value, scope (highlighted by color), and overridden status where relevant.



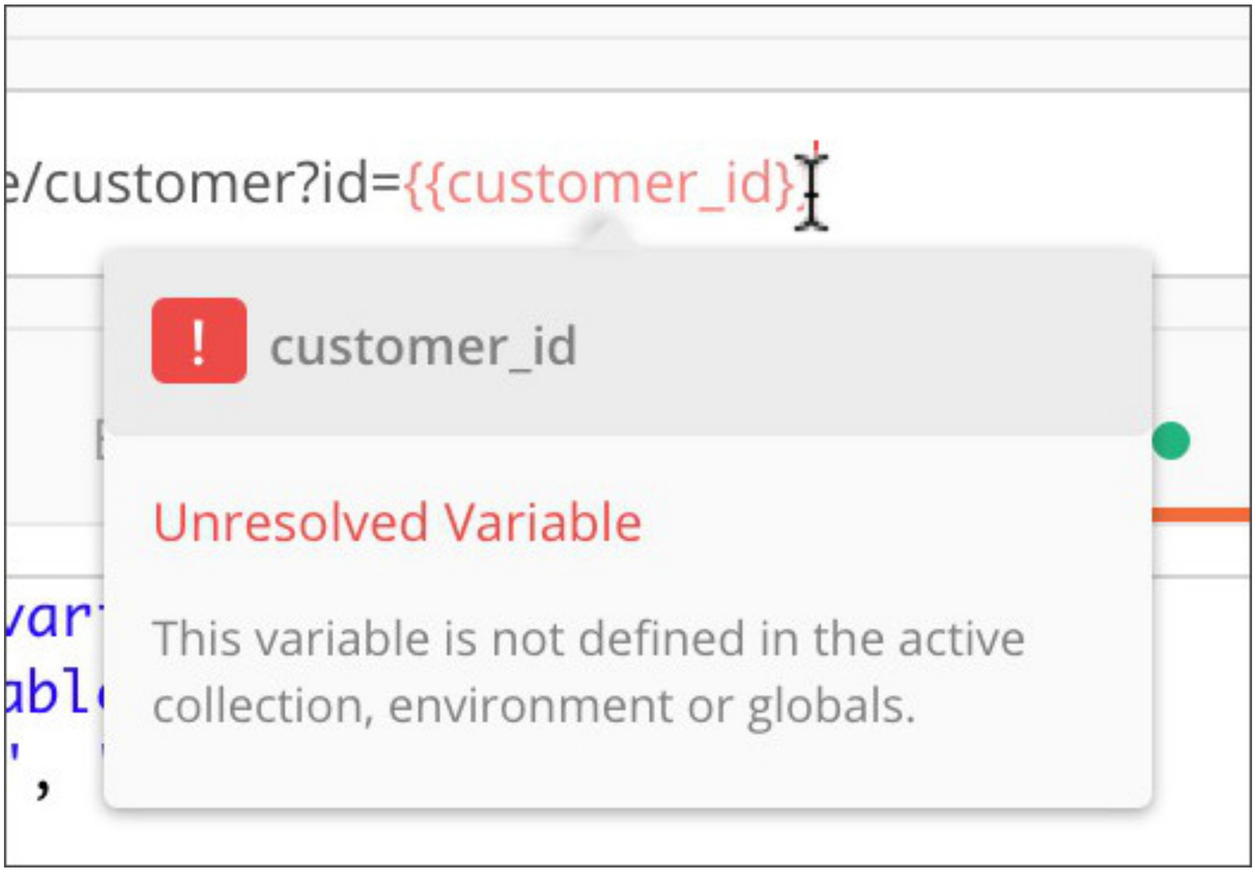
http://pricey-trilby.glitch.me/customer?id={{v

```
pm.globals.set("variable_key", "variable_val
pm.collectionVariables.set("variable_key", "
pm.environment.set("variable_key", "variable

console.log(pm.variables.get("variable_key"));
```

This also exists as an environment variable. Environment variables overwrite global variables.

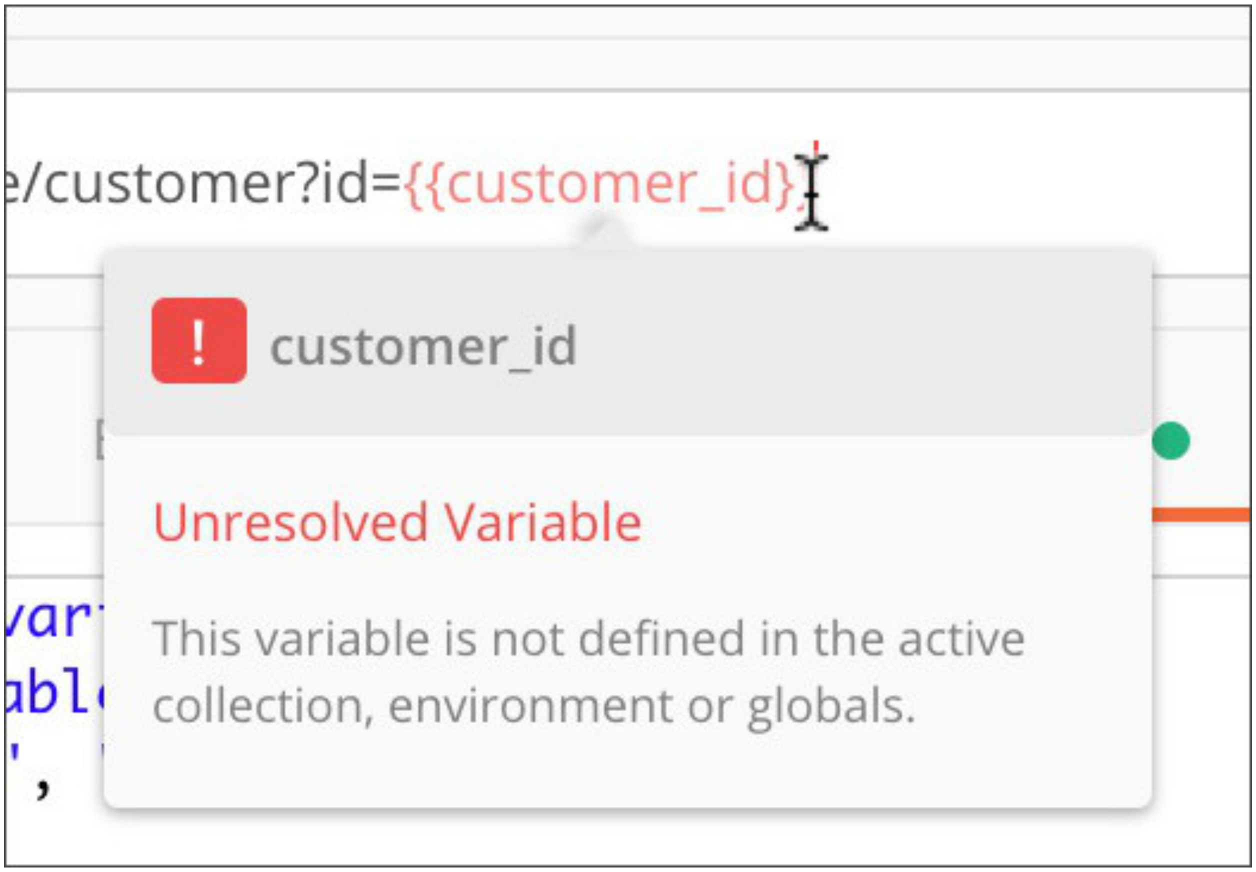
If a variable is unresolved, Postman will highlight it in red.


/customer?id={{customer_id}}

customer_id

Unresolved Variable

This variable is not defined in the active collection, environment or globals.



This variable is not defined in the active collection, environment or globals.

Using variables in scripts

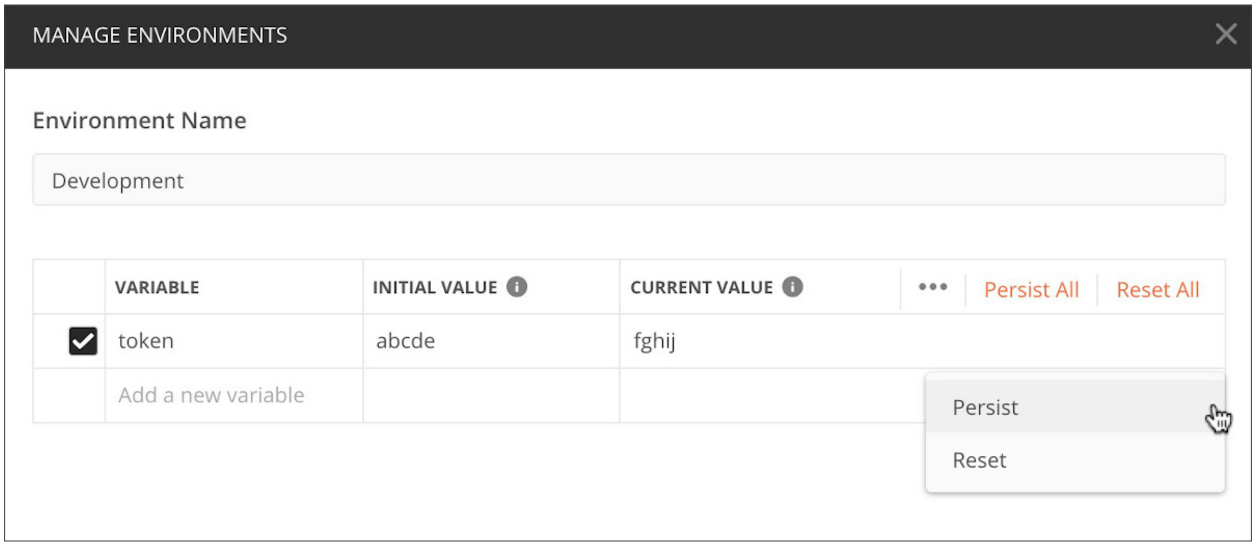
You can retrieve the current value of a variable in your scripts using the object representing the scope level and the `.get` method:

```
//access a variable at any scope including local
pm.variables.get("variable_key");
//access a global variable
pm.globals.get("variable_key");
//access a collection variable
pm.collectionVariables.get("variable_key");
//access an environment variable
pm.environment.get("variable_key");
```

Using `pm.variables.get()` to access variables in your scripts gives you the option to change variable scope without affecting your script functionality. This method will return whatever variable currently has highest precedence (or narrowest scope).

Sessions in Postman

When you edit global, collection, and environment variables in Postman, you will see **Current Value**, **Persist**, and **Reset** options for individual variables and for all variables. These allow you to control what happens within your local instance of Postman, independently of how the data is synced with anyone you're sharing requests, collections, and environments with.



MANAGE ENVIRONMENTS

Environment Name

Development

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	token	abcde	fghij
	Add a new variable		

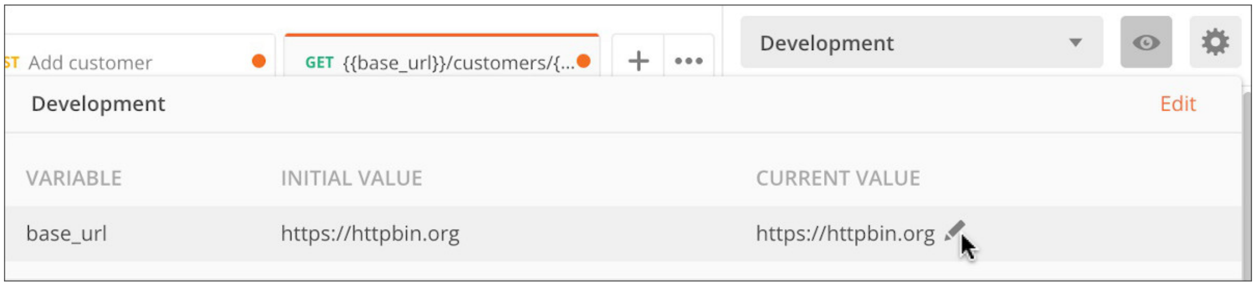
Your local session in Postman can use values that are transient and only visible to you. This lets you develop and test using private credentials or experimental values, without risk of exposing these details or affecting others on your team.

For example, your team could have a shared API key and individual API keys—you could do more experimental development work locally using your personal key, but use the shared key for team collaboration. Similarly, you could have a variable that represents exploratory work you're doing locally but are not ready to share with the team—you can later choose to persist the local data so that others on your team can also access it.

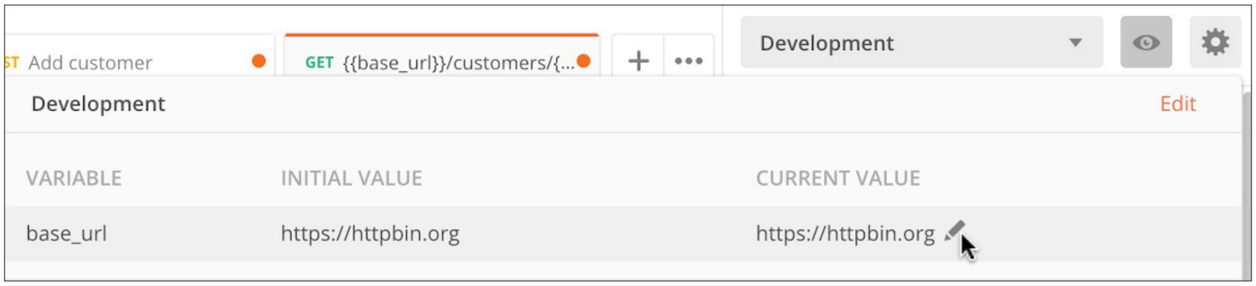
When you create or edit a variable, you can enter both an initial and a current value. You can choose to leave the current value empty, in which case it will default to the initial value. If you specify a current value, it will be local only to your instance—the **Persist** option lets you push your current value to the shared data, updating the initial value to match the current value.

Using **Persist** will make your current value sync with Postman's servers and be reflected for anyone sharing your collection or environment. To reset your current local values to reflect the initial (shared) values, use **Reset**.

You can edit a current value inline from the Environment quick look:



VARIABLE	INITIAL VALUE	CURRENT VALUE
base_url	https://httpbin.org	https://httpbin.org



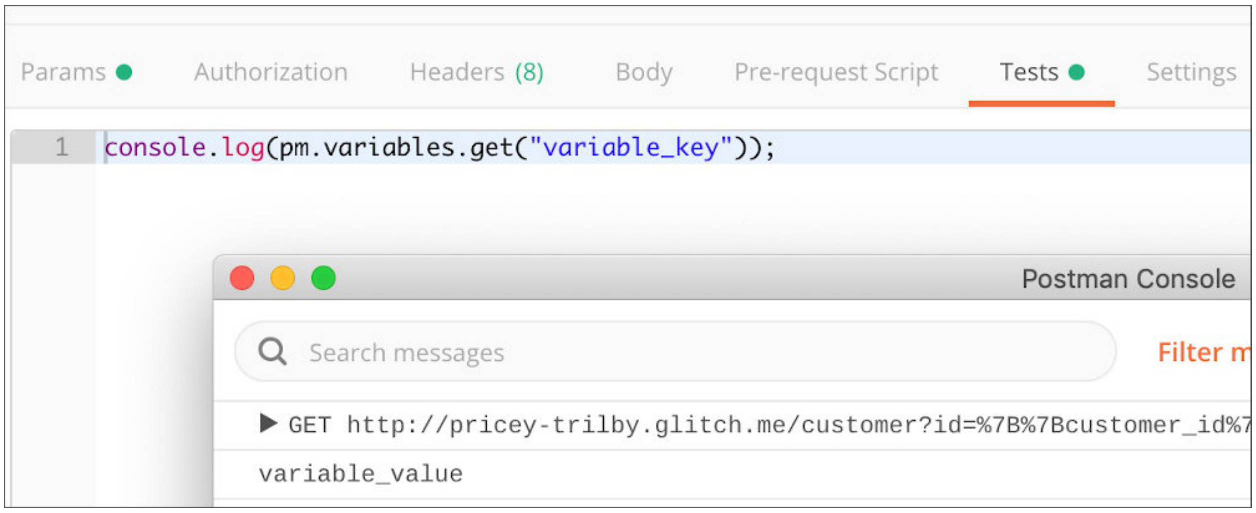
VARIABLE	INITIAL VALUE	CURRENT VALUE
base_url	https://httpbin.org	https://httpbin.org

Local and data variables only have current values, which do not persist beyond request or collection runs.

Logging variables

You can log variable values to the while your requests run. Open the console from the button on the bottom left of Postman, or from the **View** menu. To log the value of a variable, use the following syntax in your script:

```
console.log(pm.variables.get("variable_key"));
```



```
console.log(pm.variables.get("variable_key"));
```

```
▶ GET http://pricey-trilby.glitch.me/customer?id=%7B%7Bcustomer_id%7
variable_value
```

```
console.log(pm.variables.get("variable_key"));
```

```
▶ GET http://pricey-trilby.glitch.me/customer?id=%7B%7Bcustomer_id%7
variable_value
```

Using data variables

The Collection Runner lets you import a CSV or a JSON file, and use the values from the data file inside requests and scripts. You cannot set a data variable inside Postman because it is pulled from the data file, but you can access data variables inside scripts, for example using `pm.iterationData.get("variable_name")`.

See ________________________ and the ________________________ for more.

Using dynamic variables

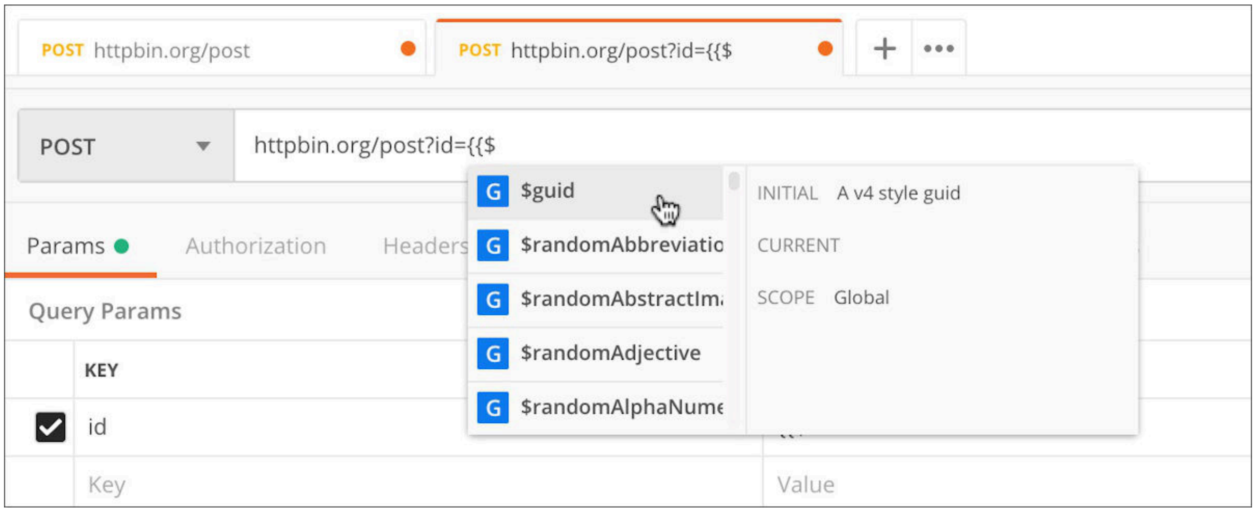
Postman provides dynamic variables that you can use in your requests.

Examples of dynamic variables are as follows:

- `{{$guid}}` : A v4 style guid
- `{{$timestamp}}`: The current timestamp (Unix timestamp in seconds)
- `{{$randomInt}}`: A random integer between 0 and 1000

See the section for a full list.

To use dynamic variables in pre-request or test scripts, you need to use `pm.variables.replaceIn()`, e.g. `pm.variables.replaceIn('{{$randomFirstName}}')`.



Next steps

Check out for more on using variables in your request scripting, and for more on how you can use data between requests.

Prerequisites

Additional Resources

VIDEOS

RELATED BLOG POSTS

Next Steps

PRODUCT

RESOURCES

USE CASES

PRICING

SUPPORT

[redacted]

COMPANY

[redacted]

[redacted]