API and Python training

Session 2

This session agenda

- REST API Authentication
- Basic authentication
- Bearer authentication
- API Keys
- OAuth
- Cookies
- Custom headers
- Combination of methods
- Practice – API authentication with cURL
- Postman interface
- Main features
- Postman practice

API Authentication

API authentication and security is a major and broad topic, we'll cover most common auth methods

First of all, API requests/responses should use HTTPS transport and valid server certificates.

Recall from last session, in REST API **the client** is responsible for providing all necessary information to the server in **each** request, so each request carries some sort of client and/or session identification.
Special headers in the requests, **special header - Authorization or custom headers**

Auth methods

- No authentication – open public services, we used them in previous session
- Basic authentication - username/password
- API keys
- Bearer token
- OAuth
- Custom headers or cookies
- Combination of the above methods

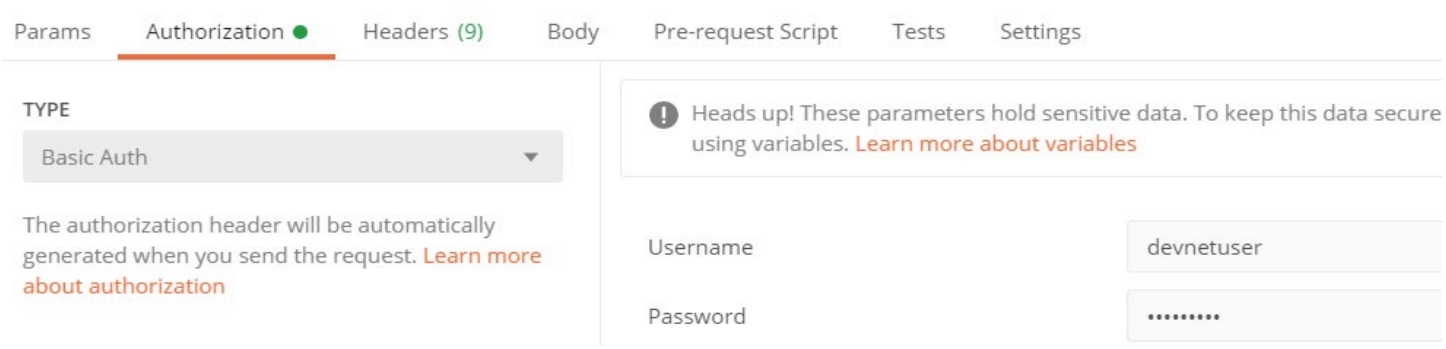
Read the documentation for the API you're going to use
Auth methods should be described there

https://blog.restcase.com/restful-api-authentication-basics/

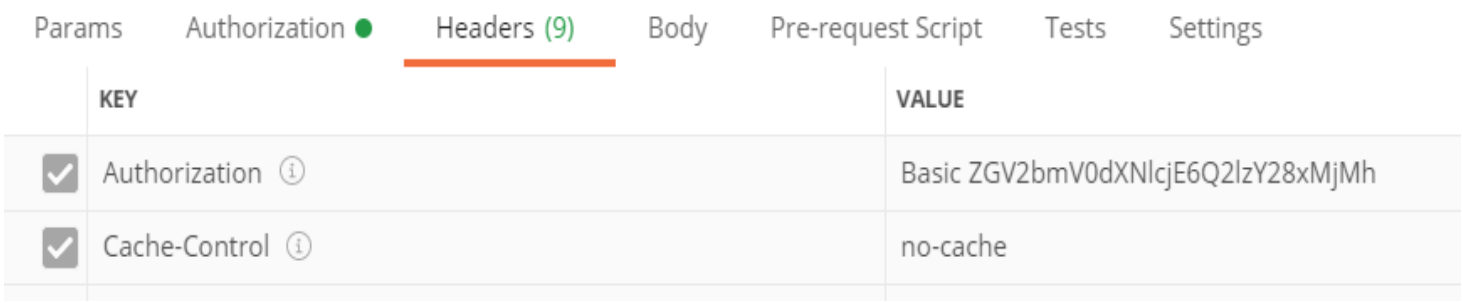
Basic authentication

Username and password

- Easy to implement
- Static username/password
- Difficult to rotate/change
- Considered not very secure
- Not used in prod. Internet-facing apps
- Used in most API-enabled network devices
- OK for PoC, test apps in internal network



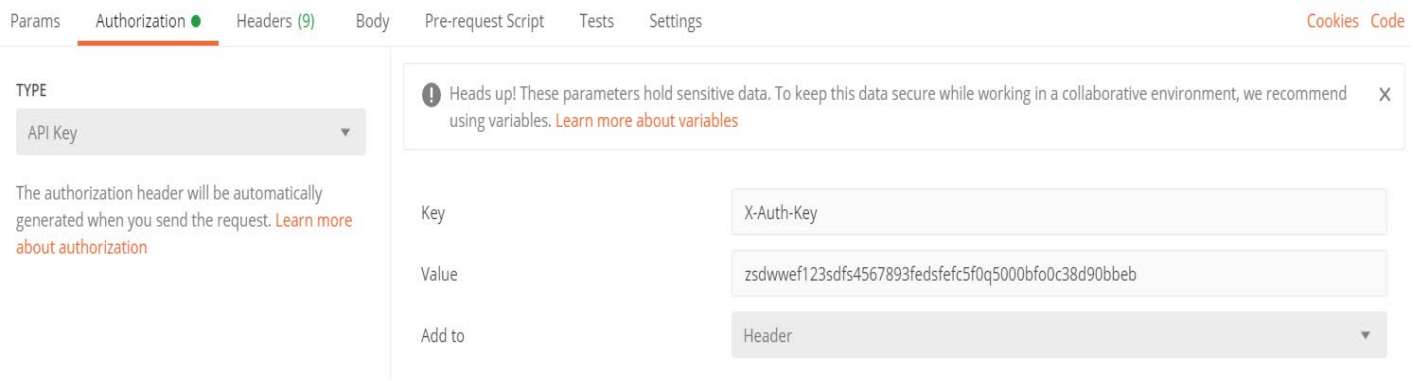
Converted into Base64 format and sent in Auth header



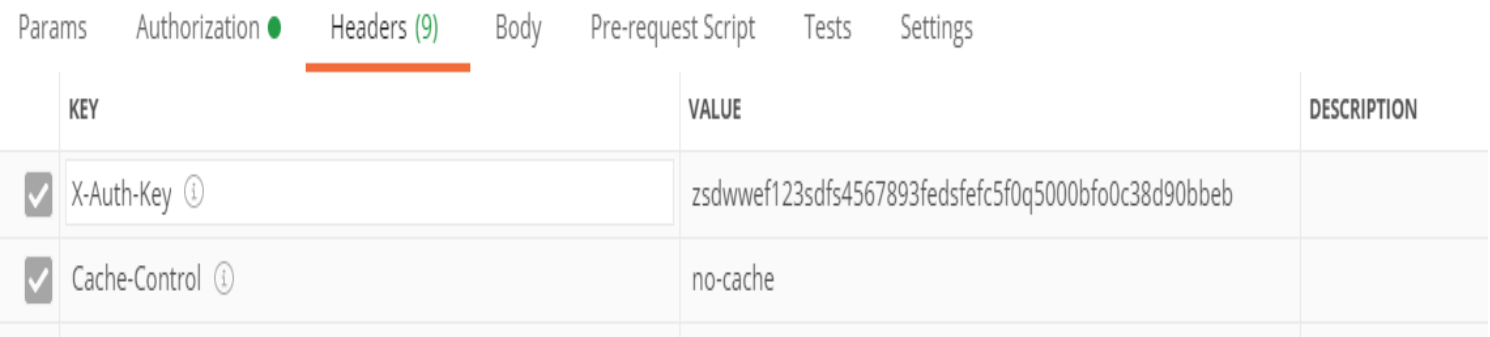
Value is **Basic**<space><token>

API Keys

- API Keys are similar to Basic
- Created per application, not user
- Don't expose real user/password
- Disallow access using other methods such as SSH, console
- Flexible, can be configured with very granular access to specific resources
- Easier to revoke
- Also not the most secure method
- Often used as first step in OAuth

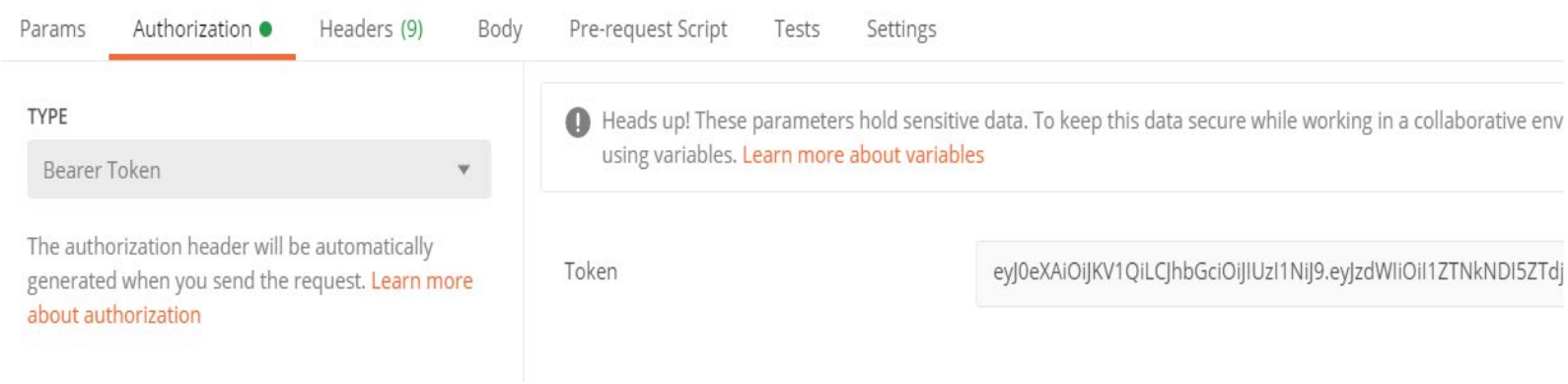


- Sent in a custom header like this:

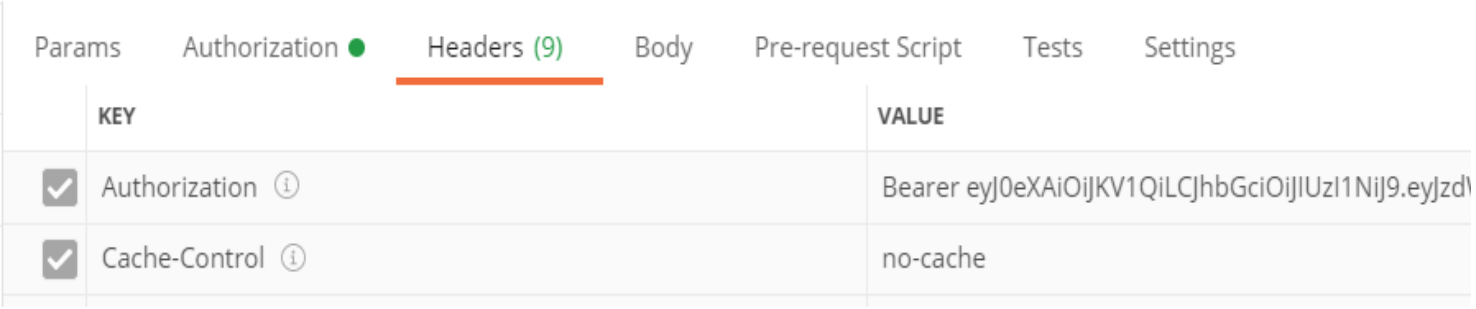


Bearer (Token) Authentication

- Uses Token
- Usually short-lived (hours)
- In most cases returned by the server in response to API call – login request
- Used in OAuth2, we'll discuss it later



- Like Basic, sent in Auth header
- Value is **Bearer**<space><token>



OAuth

Two-steps procedure

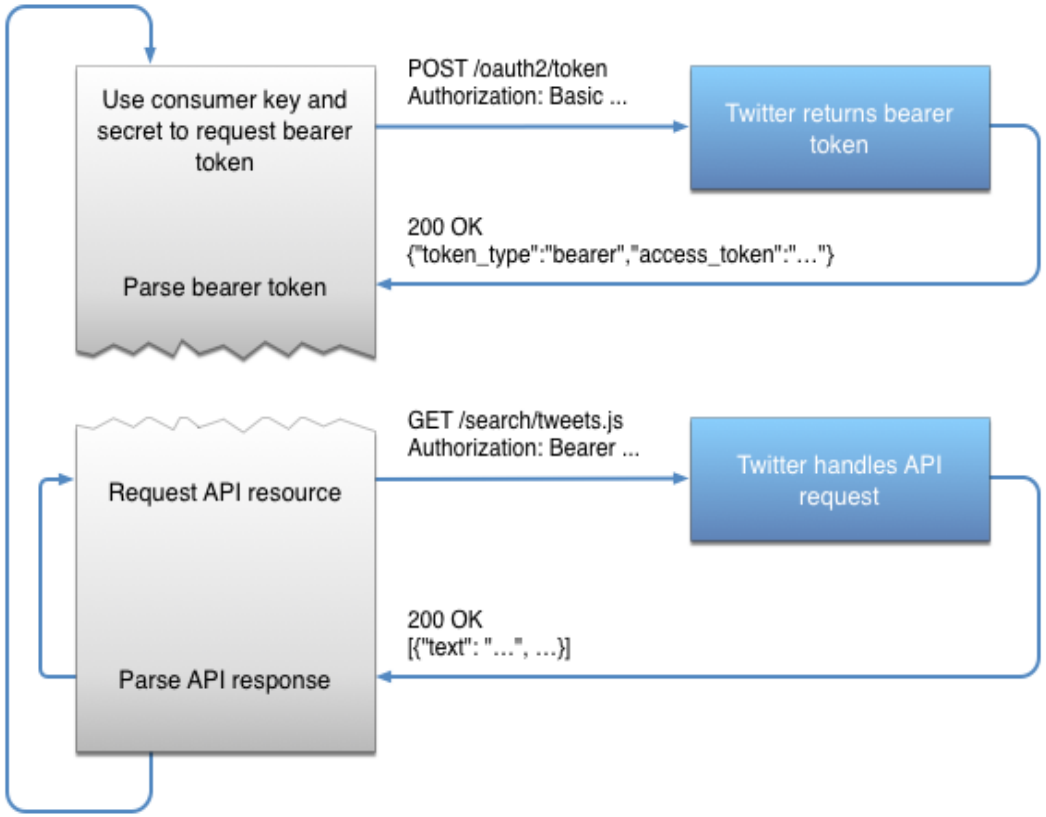
- **First request** to obtain a token to a special Auth service
- Basic Auth is used at this step, normally not username/password, but API keys – client ID and client secret
- OAuth server can be the same organization or third-party
- Possible to use MFA

- **Second request** to the actual API endpoint with

Bearer values as Token received in the first step

Token has limited time, usually hours

No API keys or usernames are exposed, only this temporary token



Considered the most secure and scalable method, used by all public cloud providers and most of SaaS providers

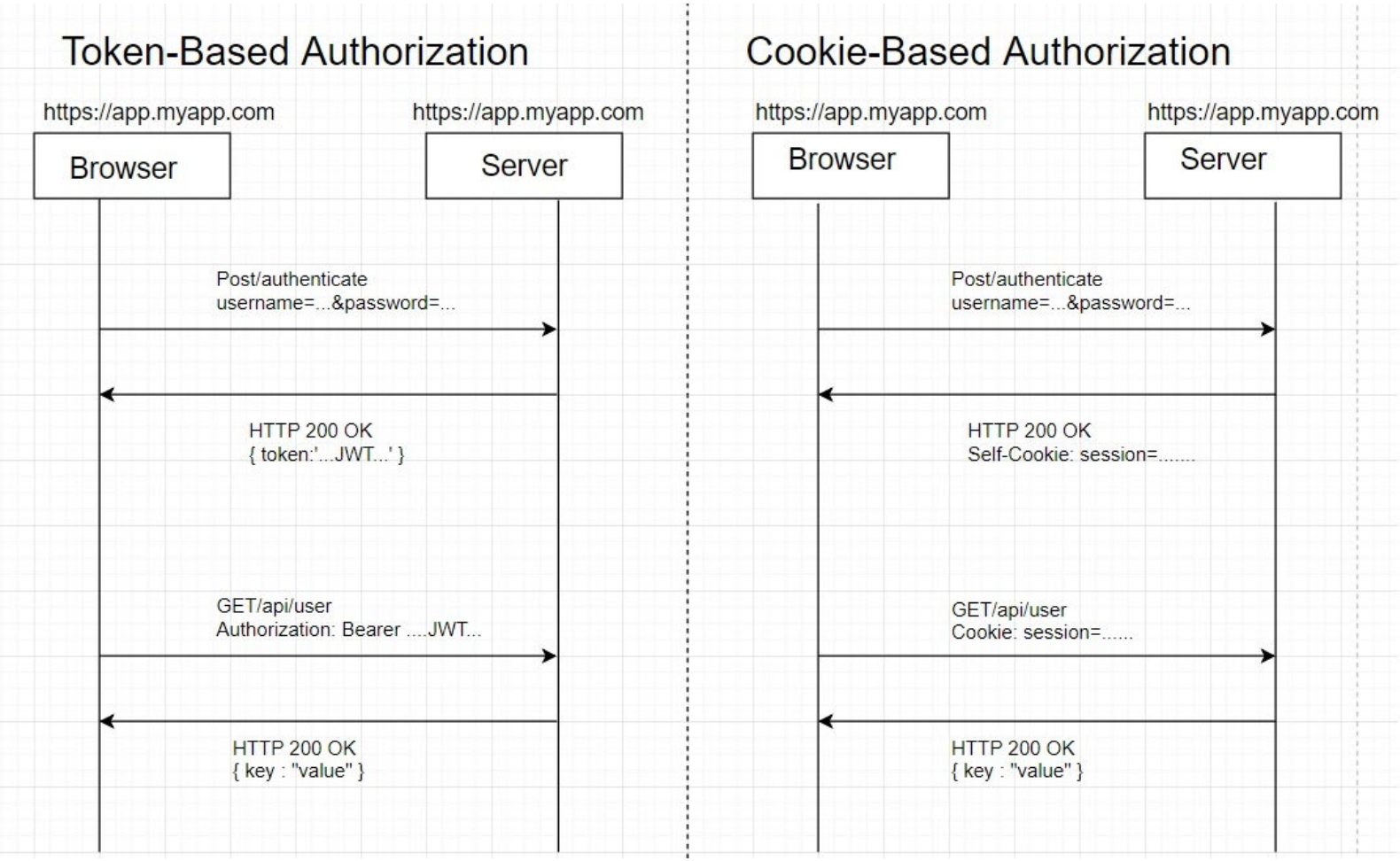
https://developer.twitter.com/ja/docs/basics/authentication/overview/application-only

Cookies

Cookies

- Similar to token-based auth
- Also two steps: first to receive secure string (cookie) and then use this sting as authenticator
- Considered not very scalable, as difficult to handle cross-domain or cross-app comms

- Used in vManage and vCentre, as the **auth server and the app are in fact the same**



https://labs.tadigital.com/index.php/2019/05/16/web-security-authentication-cookies-vs-tokens-vs-html5-web-storage/

Combination of several methods

- Multiple Headers:

```
curl https://api.cloudflare.com/client/v4/zones/cd7d0123e301230df9514d \
   -H "X-Auth-Key: 1234567893feefc5f0q5000bfo0c38d90bbeb" \        <<<<<  API Key
   -H "X-Auth-Email: example@example.com"                          <<<<<  plus additional Header
```

- Combination:

```
 curl -X POST   "https://172.21.36.35:8443/dataservice/device/action/software"  \
-d '{"platformFamily":"c1100"}'  -b cookies.txt  -H "X-XSRF-TOKEN: $TOKEN"  -H "Content-Type: application/json"
                                    ^^^^^            ^^^^^^^^^^^^
                                 Cookies file  AND    Custom Header
```

Summary - Auth methods depend on the application, no standard ways, check the documentation for the application

Practice Time

We'll use **Cisco DNA Centre Sandbox** and **ServiceNow Dev** instance

cURL and then Postman

ServiceNow

We'll be using demo instance I created for testing, but you can create your own: https://developer.servicenow.com/

https://dev78092.service-now.com/navpage.do

We'll use **Incidents** and **Linux Servers** tables

ServiceNow supports Basic Auth and OAuth,

we will use Basic

System Administrator

Incidents | New | Search | Number | Search

All > Active = true

Number	Opened	Short description	Caller	Priority	State	Category	Assignment group	Assigned to
INC0010112	2019-07-29 11:48:43	Assessment : ATF Assessor	survey user	5 - Planning	New	Inquiry / Help	(empty)	(empty)
INC0010111	2019-07-22 14:04:57	ATF : Test1	System Administrator	5 - Planning	New	Inquiry / Help	(empty)	(empty)
INC0010002	2020-11-15 22:46:00	Test incident -2	Teri Erlewine	3 - Moderate	In Progress	Inquiry / Help	Network	Bow Ruggeri
INC0009009	2018-08-30 01:06:16	Unable to access the shared folder.	David Miller	4 - Low	New	Inquiry / Help	(empty)	(empty)
INC0009005	2018-08-31 21:35:21	Email server is down.	David Miller	1 - Critical	New	Software	(empty)	(empty)
INC0009001	2018-09-11 20:56:26	Unable to post content on a Wiki page	David Miller	3 - Moderate	New	Inquiry / Help	(empty)	(empty)
INC0007002	2018-10-16 22:47:51	Need access to the common drive.	David Miller	4 - Low	New	Inquiry / Help	(empty)	(empty)
INC0007001	2018-10-16 22:47:10	Employee payroll application server is down.	David Miller	1 - Critical	New	Hardware	Openspace	(empty)

Linux Servers | New | Search | Name | Search

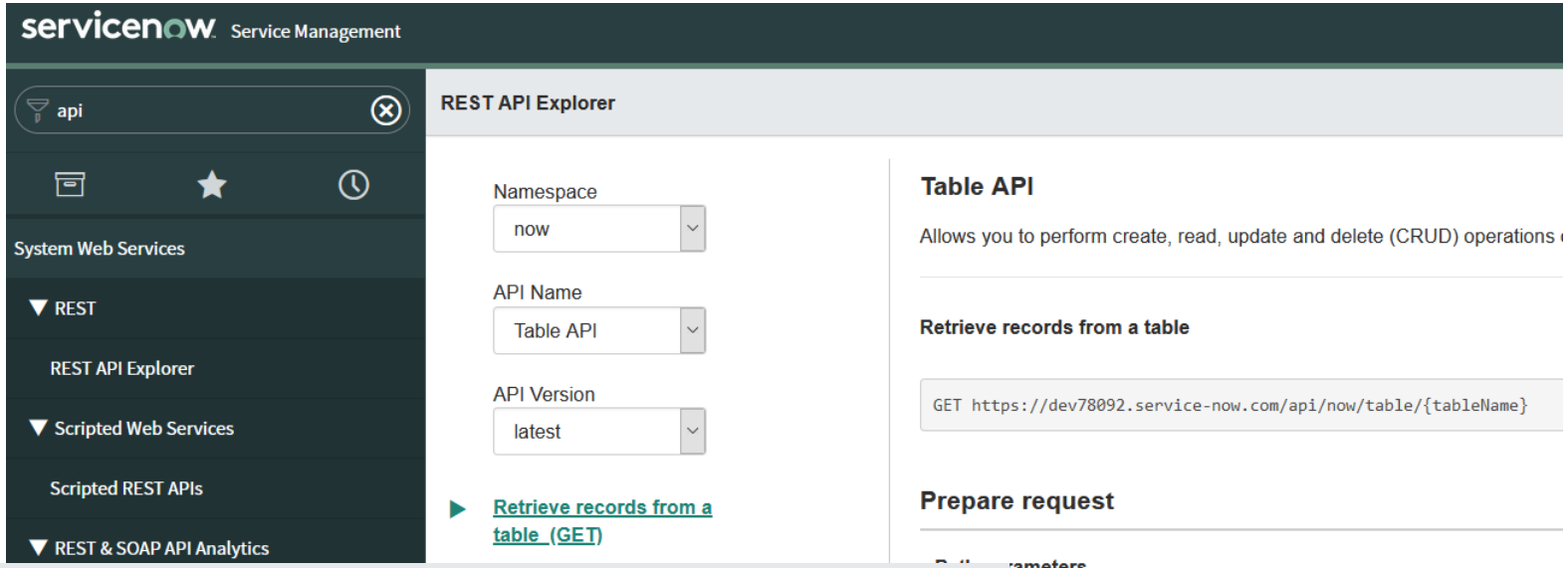
All

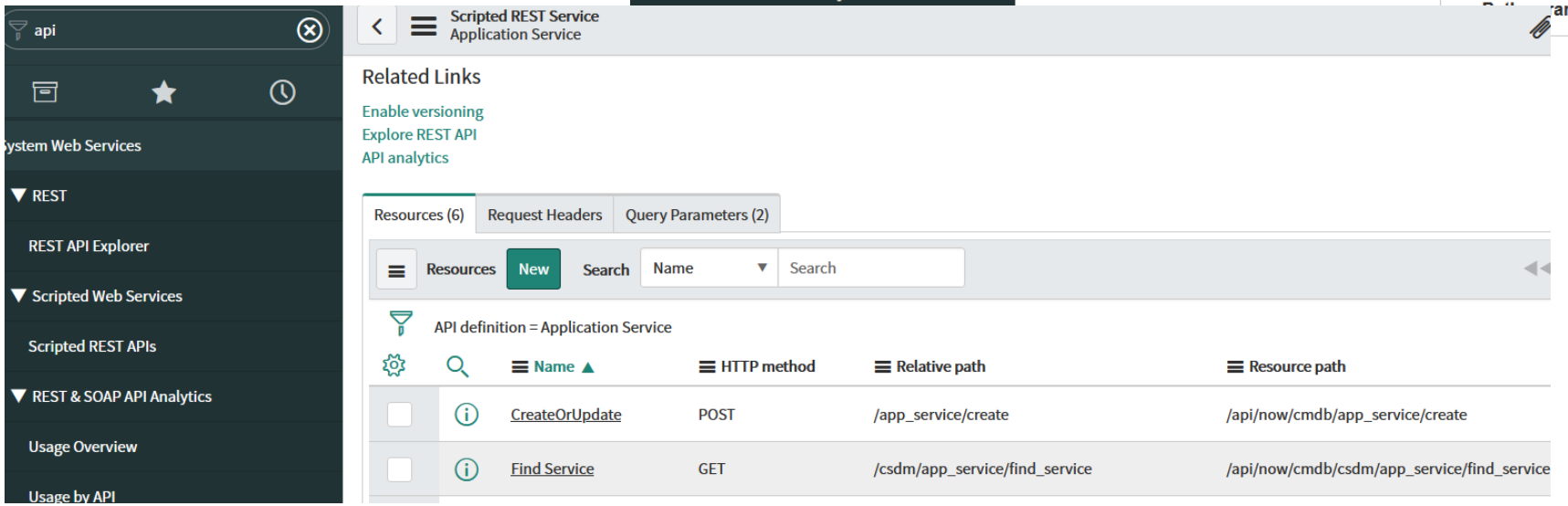
Name	Operating System	OS Version	Manufacturer
lnux100	Linux Red Hat	2.6.9-22.0.1.ELsmp	Iris
lnux101	Linux Red Hat	2.6.9-22.0.1.ELsmp	Iris
PS LinuxApp01	Linux Red Hat	2.6.9-22.0.1.ELsmp	Iris
PS LinuxApp02	Linux Red Hat	2.6.9-22.0.1.ELsmp	Iris

ServiceNow API Doc

Search for API

You can try real API calls





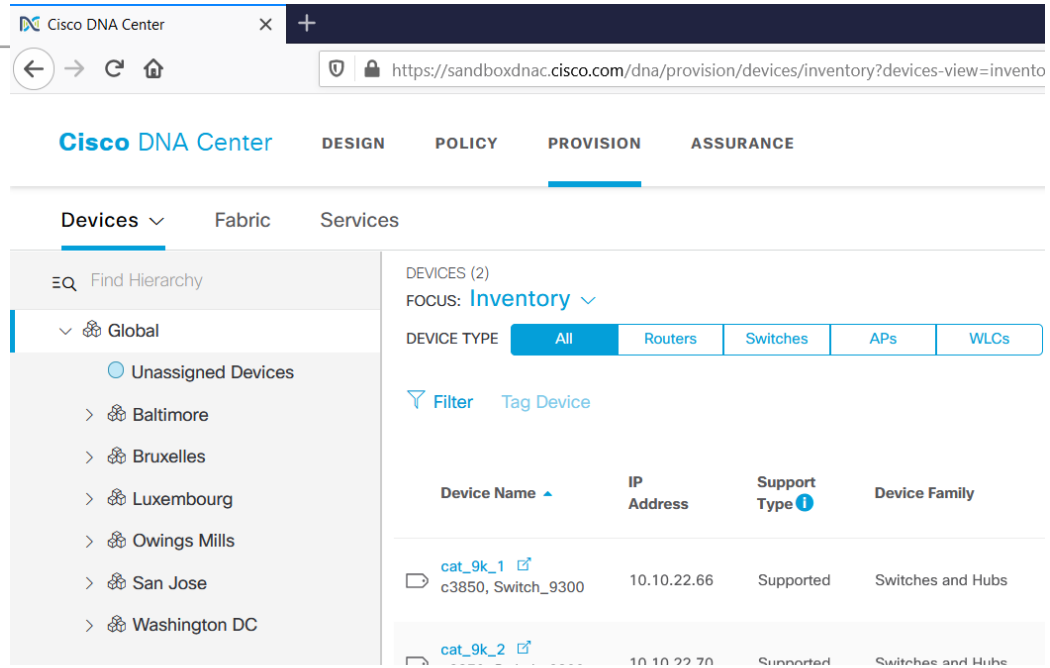
Cisco DNA Center

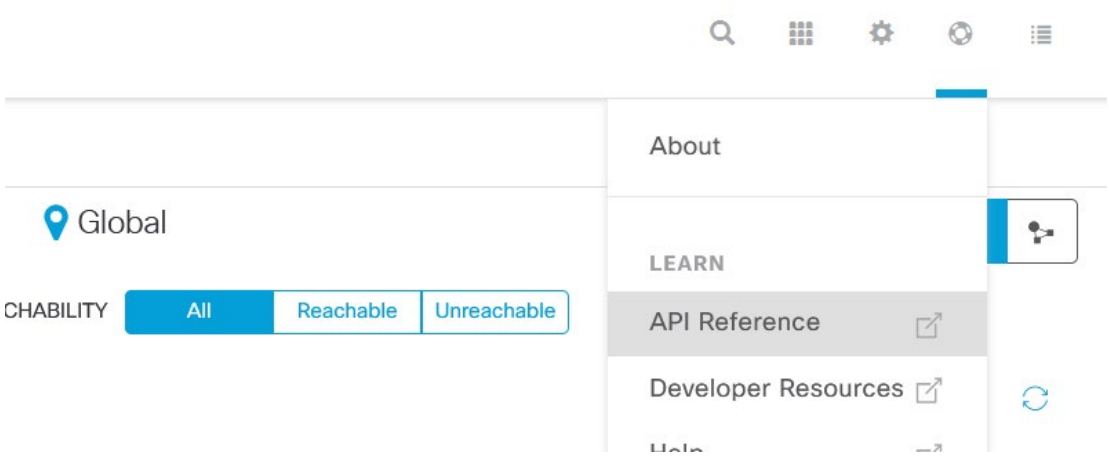
Cisco DNA Centre Sandbox
https://sandboxdnac.cisco.com

- Live instance
- Only GET requests
- Login – devnetuser
- Password – Cisco123!
- Supports OAuth

API documentation:

https://developer.cisco.com/docs/dna-center/#!cisco-dna-center-platform-overview/intent-api-northbound





Practice Basic Auth with Service Now

Basic Auth - see https://dev78092.service-now.com/$restapi.do#/code_modal

- curl –v "https://dev78092.service-now.com/api/now/table/incident?sysparm_limit=1" \

--request GET \

--header "Accept:application/json" \

--user 'api_user':'secret-here'

Practice custom Auth Header with Cisco DNAC

https://sandboxdnac.cisco.com

Username: devnetuser

Pass: Cisco123!

Step 1 - get Token from Auth API endpoint :

```
curl  -k --request POST --url https://sandboxdnac.cisco.com/dna/system/api/v1/auth/token --user 'devnetuser':'Cisco123!'
```

OR

```
TOKEN=$(curl --insecure --request POST --header "Authorization: Basic $(echo -n devnetuser:Cisco123! | base64)" \
https://sandboxdnac.cisco.com/dna/system/api/v1/auth/token -v \
| python -c "import sys, json; print json.load(sys.stdin)['Token']")
echo $TOKEN
```

Step 2 – make request using this token in header x-auth-token:

```
curl -k  --location --request GET 'https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device' --header "x-auth-token: $TOKEN" -v | python -m json.tool
```

Requests to try for vCentre and vManage

vCentre – Non Prod:

- read -s PASS
- curl -k -u nonp-username@nonp.nttict.com.au:$PASS -X POST https://10.7.142.15/rest/com/vmware/cis/session -c cookie.txt
- curl -k -b cookie.txt https://10.7.142.15/rest/vcenter/vm | jq

OR curl -k -b cookie.txt https://10.7.142.15/rest/vcenter/vm | python -m json.tool

vManage - Demo:

- read -s j_password
- curl -v --request POST -k --url https://172.21.36.35/j_security_check --data "j_username=username&j_password=$j_password" -c cookies.txt
- curl -k -b cookies.txt --url https://172.21.36.35/dataservice/server/info

for POST requests to vManage use additional token:

- TOKEN=$(curl -k -b 'cookies.txt' --url https://172.21.36.35/dataservice/client/token)
- echo $TOKEN
- curl -k -vvv -X POST "https://172.21.36.35:8443/dataservice/device/action/software" \

-d '{"platformFamily":"c1100","controllerVersionName": "20.1.x", "versionName":"17.2.2", "versionURL":"http://t.blob.core.windows.net/images/c1100-universalk9.17.02.02.SPA.bin"}' \ -b cookies.txt --insecure -H "X-XSRF-TOKEN: $TOKEN" -H "Content-Type: application/json"

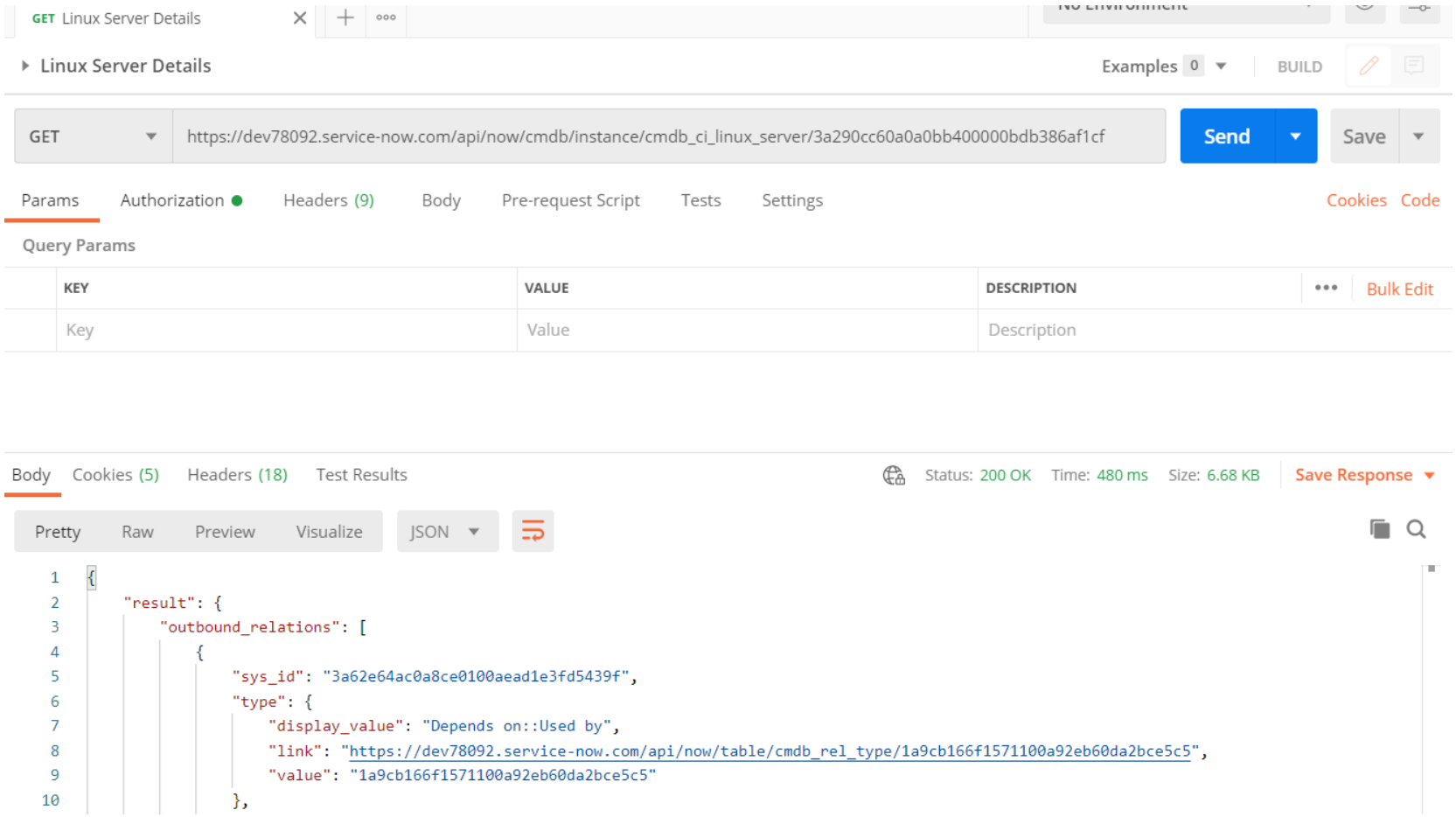
Postman

We tried using API in CLI, let's use a UI client which is Postman

Postman is very popular API querying and testing tool.

Request

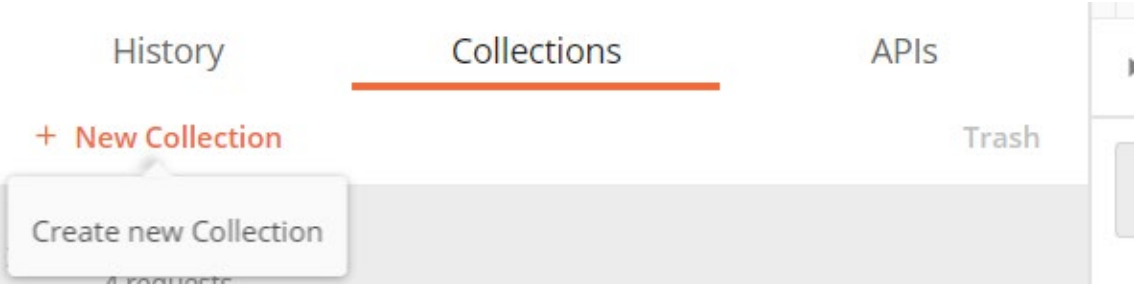
Response



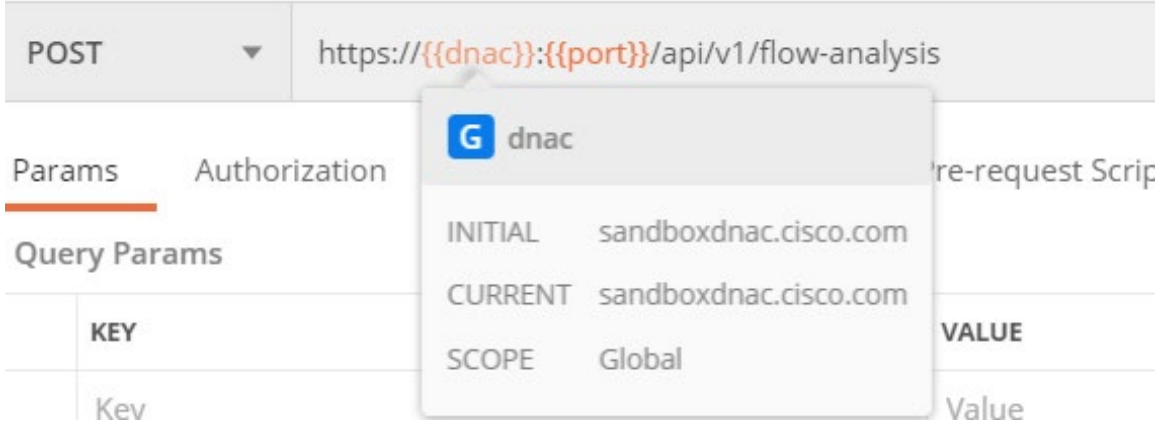
https://learning.postman.com/docs/sending-requests/requests/
https://www.guru99.com/postman-tutorial.html

Postman – main features

Organise Requests to **collections** – like folders



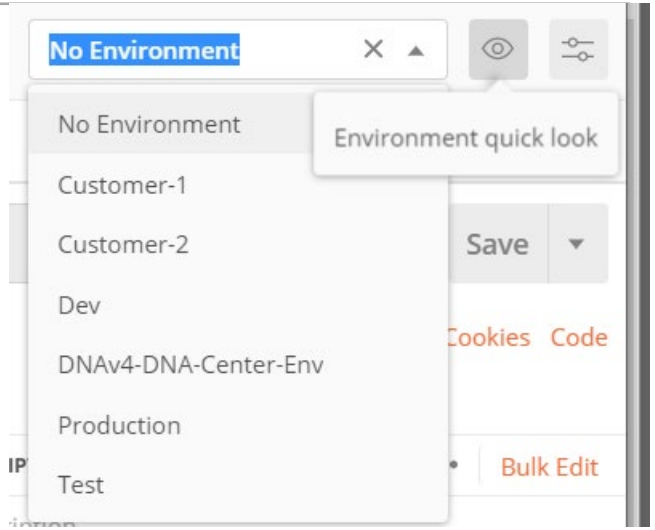
- **Variables** – often used in URLs, especially where you use the same requests to different systems
- For example, we have multiple vManage controllers, and need to make the same requests, easier not to hardcode URL/port, but use variables
- Format is {{variable}}



Postman – Environments and Global

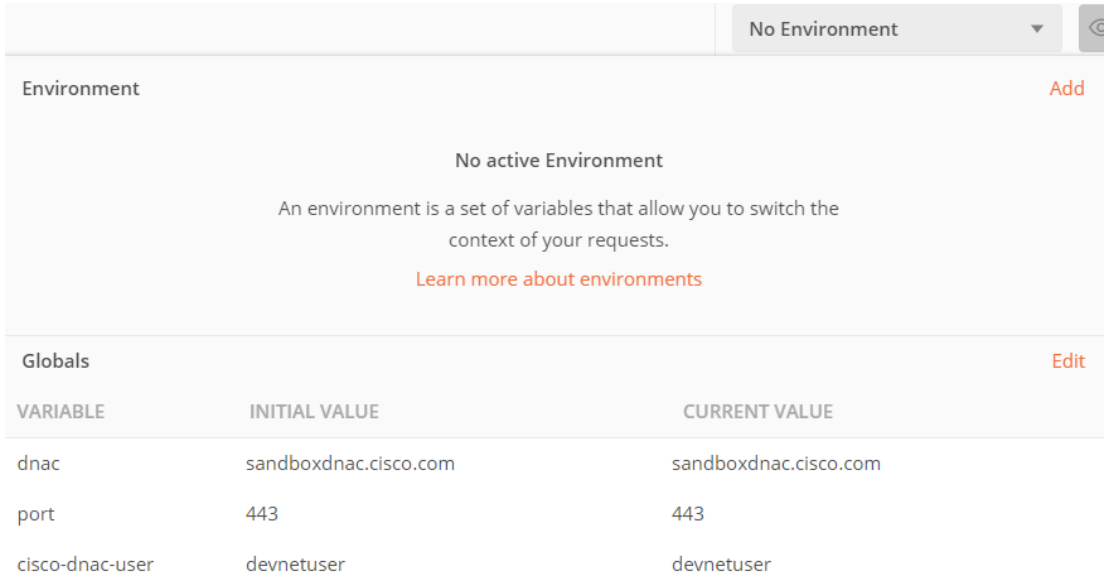
Two methods to set values to variables

Environments – define variables for each env, so you can switch between them and variables will be assigned different values



Globals – fixed

Not dependent on env



Practice with ServiceNow – Postman - GET

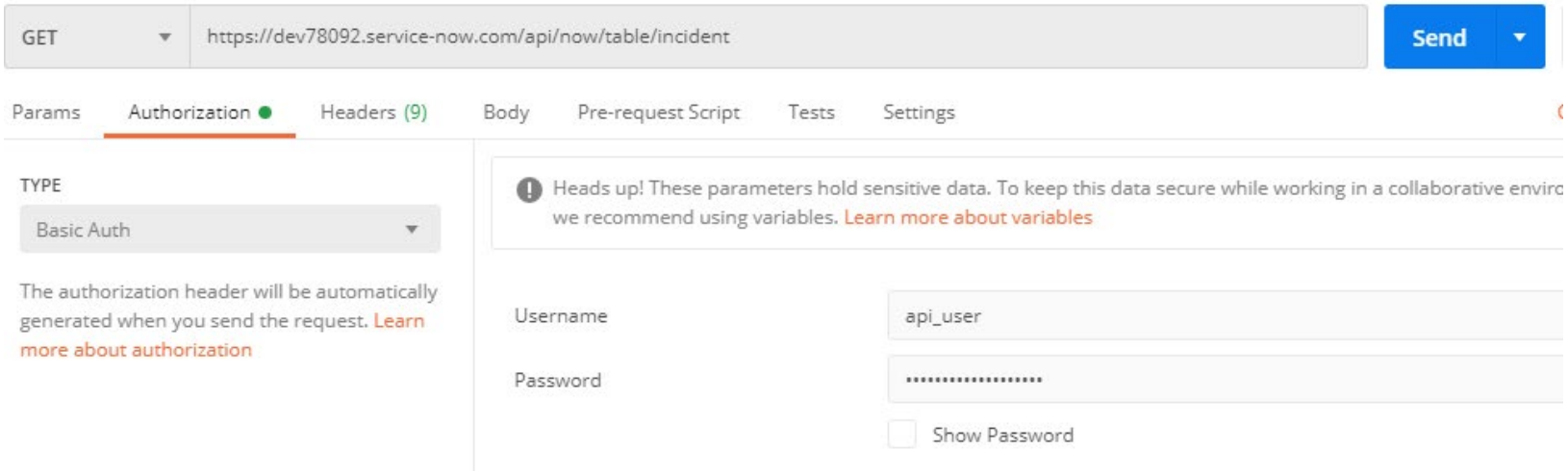
Enter URL https://dev78092.service-now.com/api/now/table/incident

- Go to Auth Tab

Auth Type - Basic

Username –**api_user**

Password – in Teams

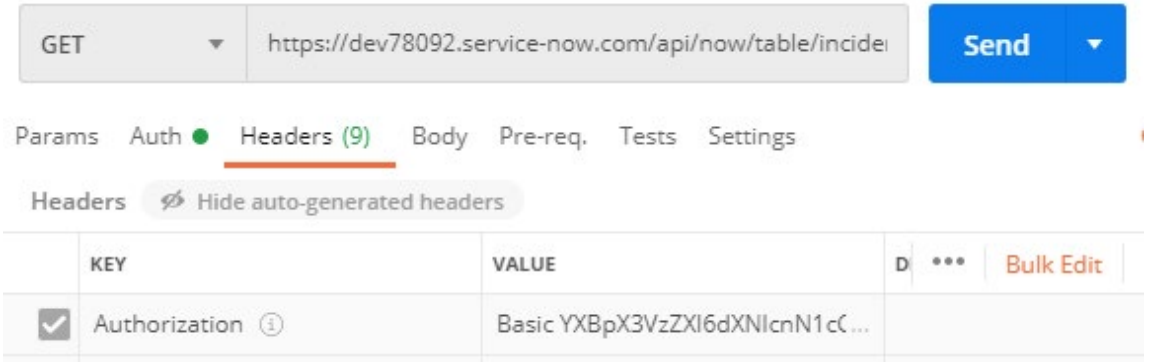


- Go to Headers Tab

Note Auth Header is populated with Base64 value

Click Send

Check the response code and Body content

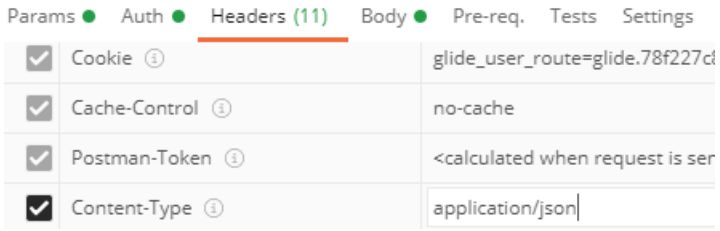


Practice with ServiceNow – Postman - POST

1. Go to Auth Tab – the same as previous request

- Go to Headers Tab

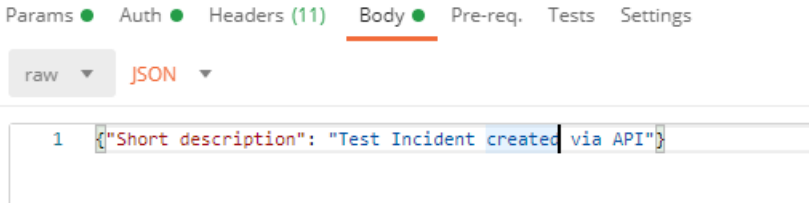
Create Key Content-Type application/json



2. Go to Body Tab

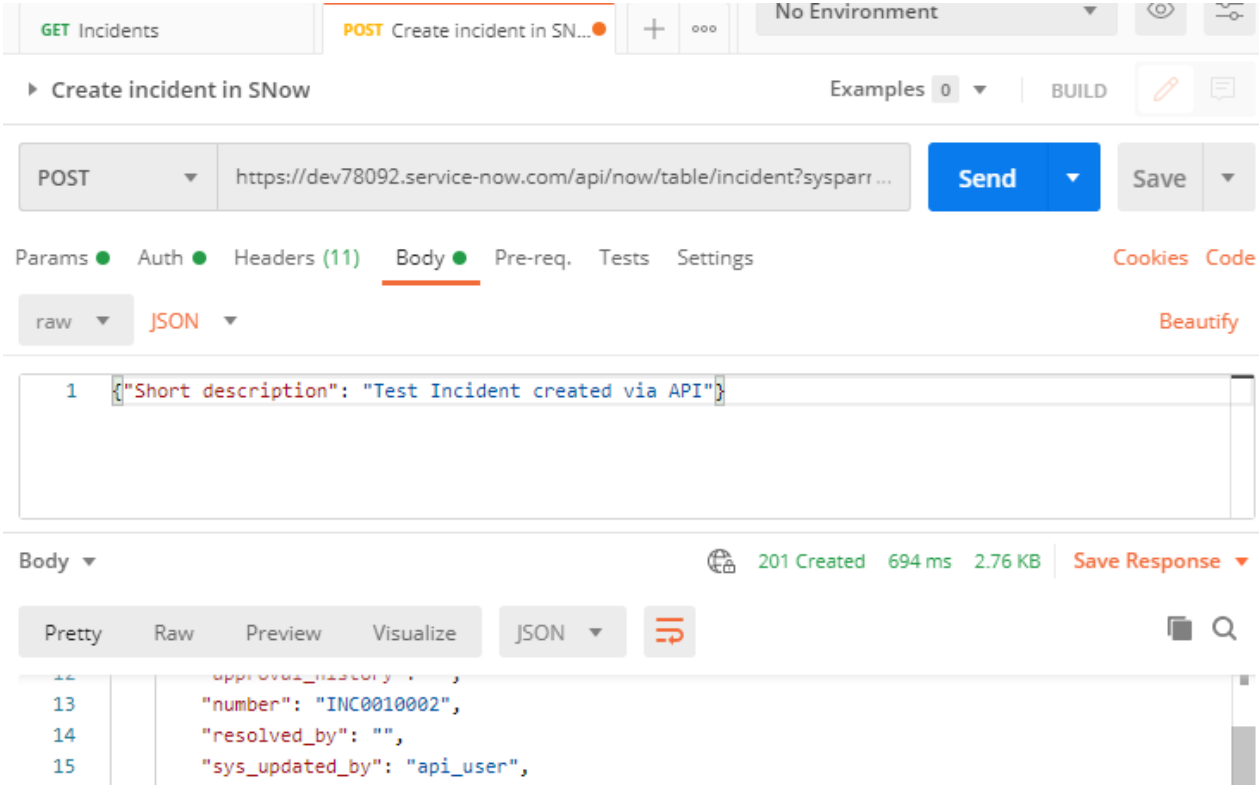
Change type to JSON and enter

```
{"Short description": "Test Incident created via API"}
```



3. Click Send

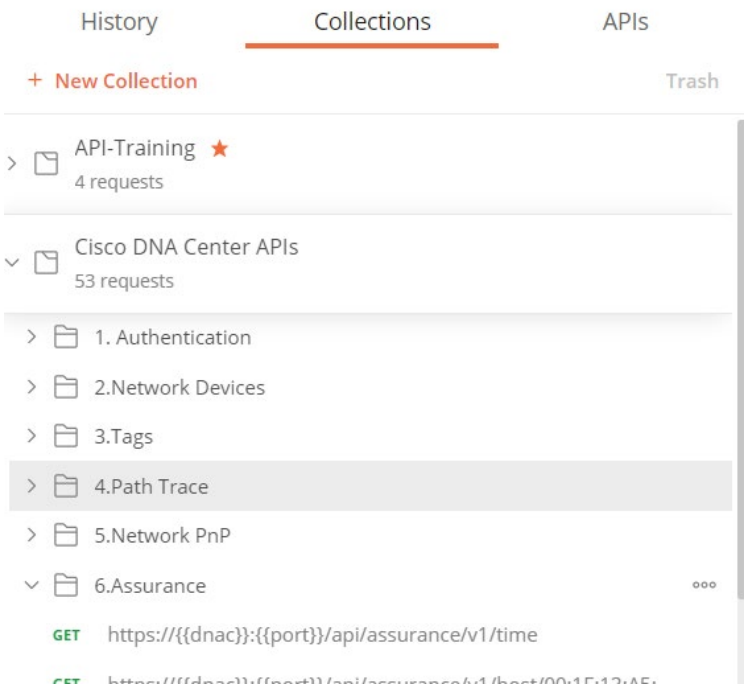
Note you get response code 201 and INC number in Body

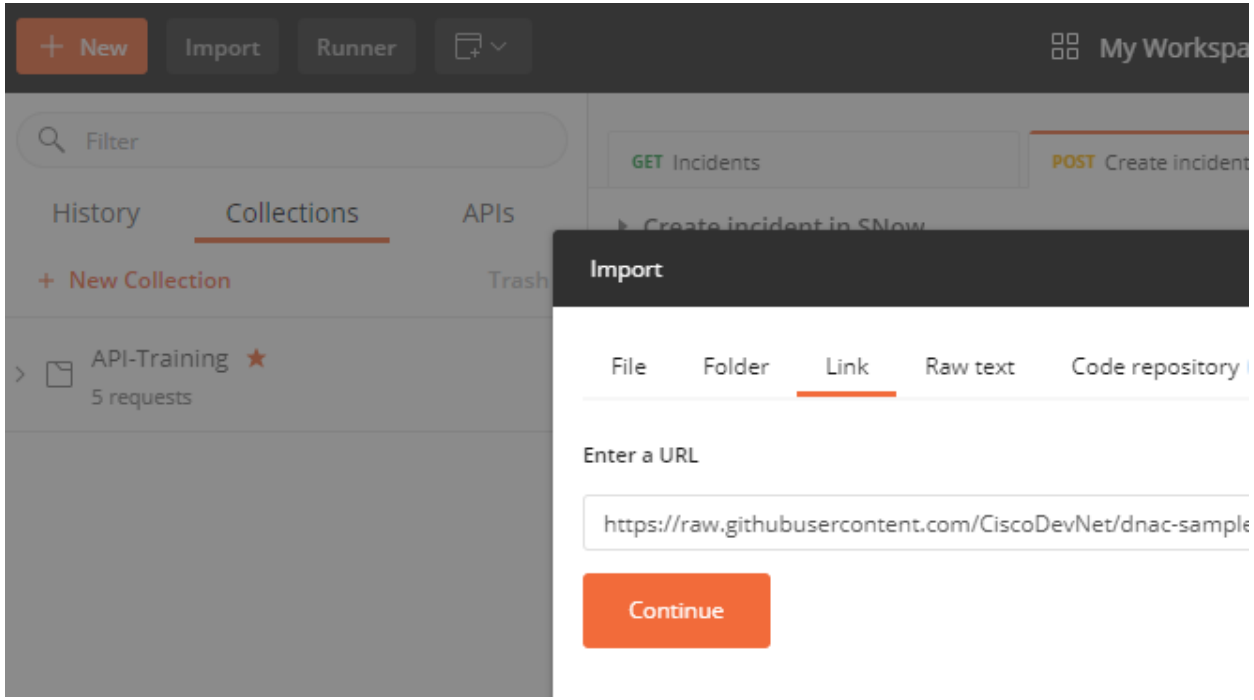


Practice with DNA Centre Sandbox - Postman

Top left corner - Import –> Link

https://raw.githubusercontent.com/CiscoDevNet/dnac-samples-aradford/master/tools/postman/01-DNAC-Sandbox.postman_collection.json

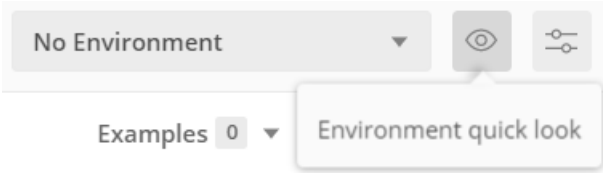


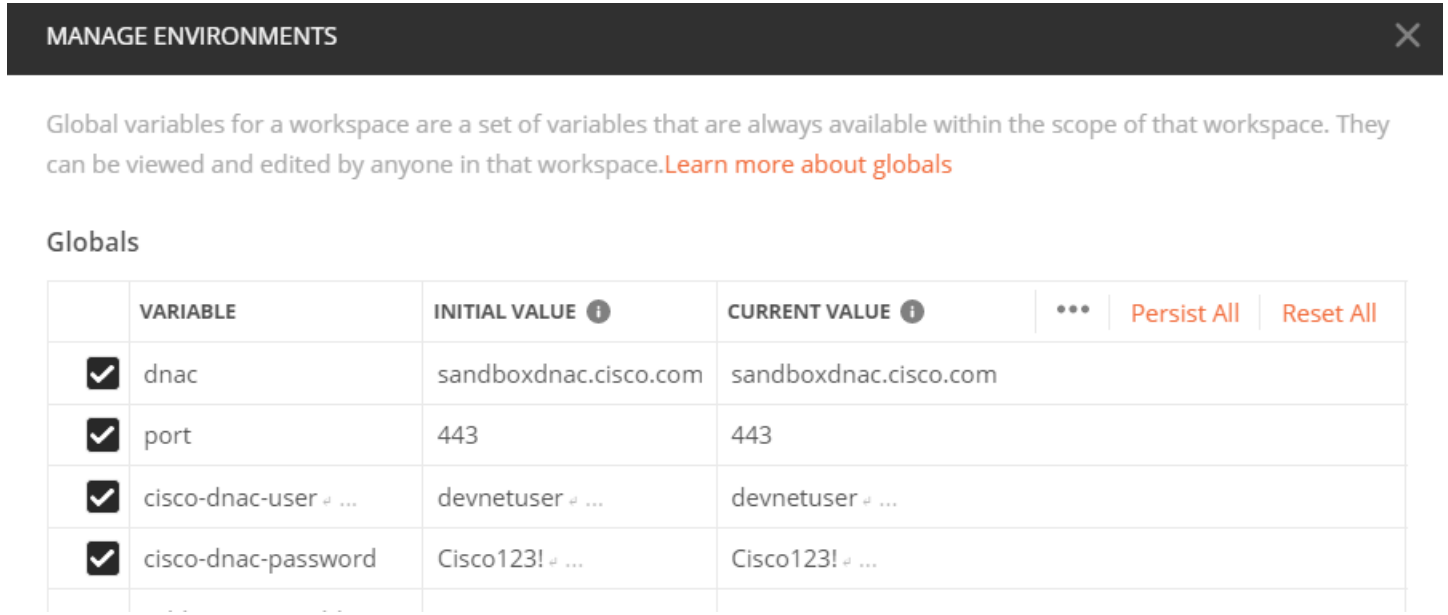


https://github.com/CiscoDevNet/dnac-samples-aradford/
https://developer.cisco.com/site/dnac-101/

Practice with DNA Centre Sandbox - Postman

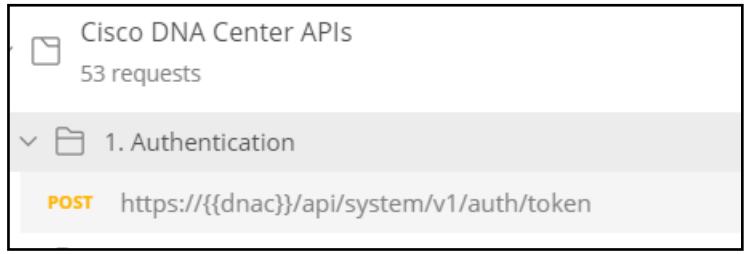
- Click the eye icon and edit Globals



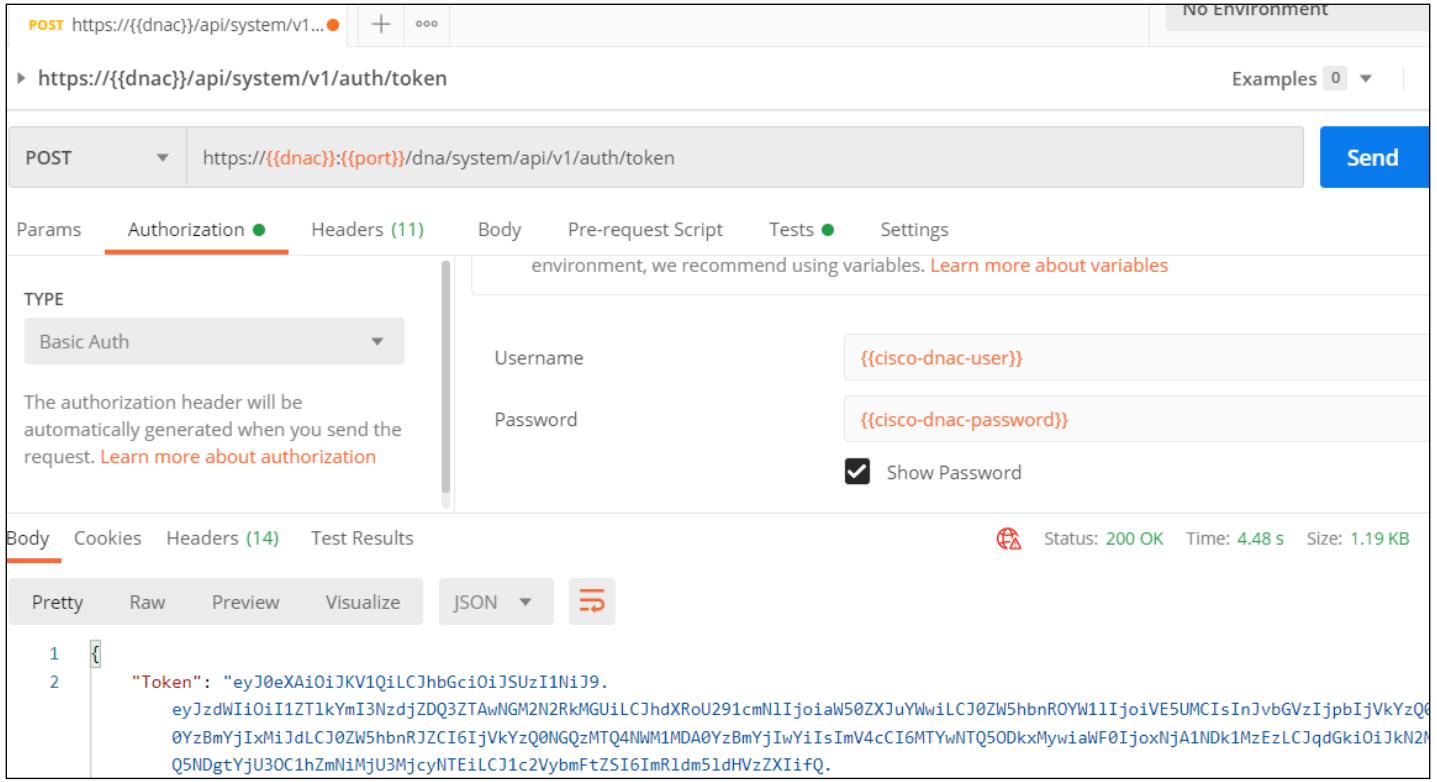


DNA Centre Sandbox

- In Postman collections choose DNA Center and Authentication



- Find here four variables ☺
- Hover over the vars to see if they have correct values
- Click Send
- See if you received status 200 OK

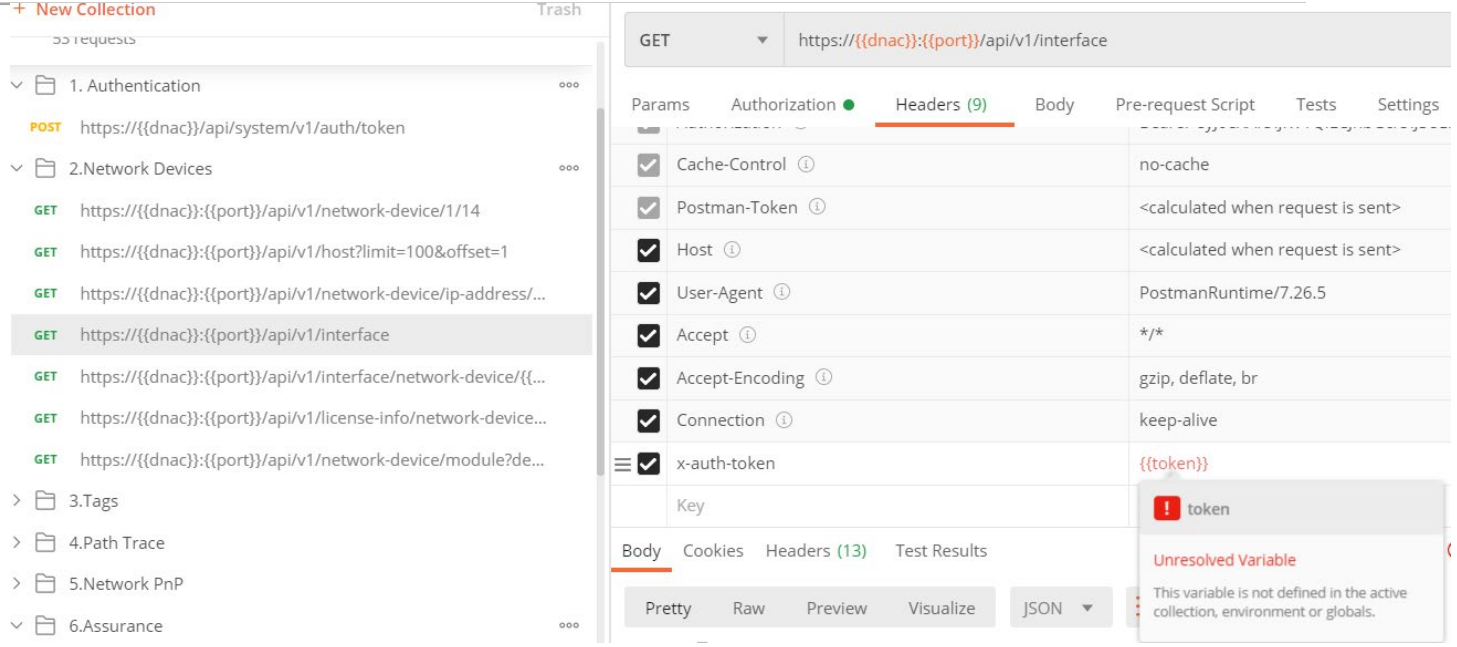


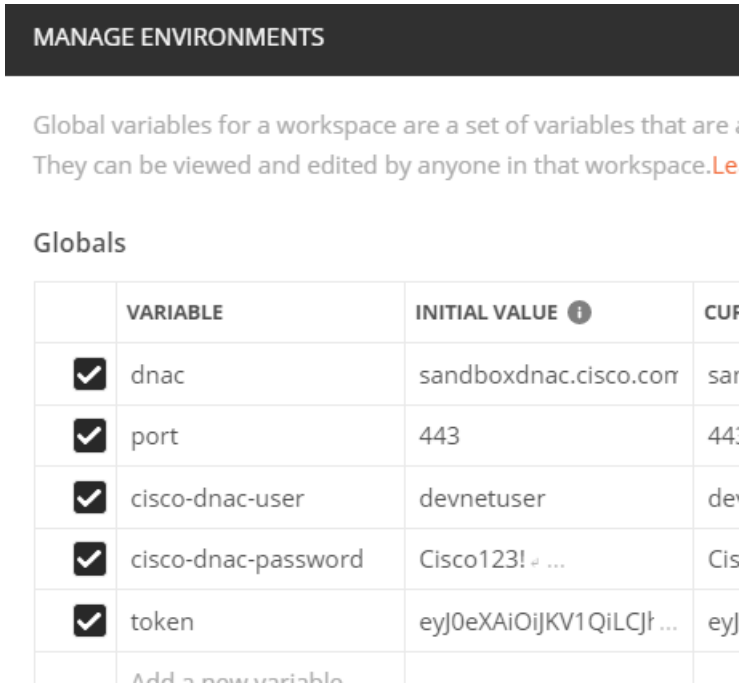
- Token will be in response body
- Copy it – it will be used in the next step

DNA Centre Sandbox

- Once you get a token at the previous step
- Choose any Get request
- Switch to Headers tab
- Note there is no token defined >>>>

- Define the token as global variable





Final step

- Click Send and see that you get Response code 200 OK and JSON data in body
- If it's not 200, but 401 or 403, check token value
- Try other requests, check what headers you use in Requests and receive in responses

POST https://{{dnac}}/api... | GET https://{{dnac}}:{{por... | GET https://{{dnac}}:{{po... | GET https://{{dnac}}:{{po...

No Environment

https://{{dnac}}:{{port}}/api/v1/interface

Examples 0 | BUILD

GET | https://{{dnac}}:{{port}}/api/v1/interface | Send

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings

Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
x-auth-token	{{token}}	
Key	Value	Description

Body | Cookies | Headers (15) | Test Results

Status: 200 OK | Time: 1402 ms | Size: 68.86 KB | Sav

Pretty | Raw | Preview | Visualize | JSON


```
8    "status": "up",
9    "adminStatus": "UP",
10   "deviceId": "21335daf-f5a1-4e97-970f-ce4eaec339f6",
11   "portName": "Vlan1",
12   "mediaType": null,
13   "speed": "1000000",
14   "duplex": null,
15   "interfaceType": "Virtual",
16   "ipv4Address": "10.10.22.97",
17   "ipv4Mask": "255.255.255.240",
18   "isisSupport": "false",
19   "mappedPhysicalInterfaceId": null,
20   "mappedPhysicalInterfaceName": null,
21   "nativeVlanId": null,
22   "ospfSupport": "true",
```

Summary and next steps

- Today we covered API authentication and practiced Postman
- In your free time practice with Postman, build a new collection and create the requests we tried in last session
- Next session we'll switch to Python and start with basics –virtual environments, data types and operations, and we'll use IDE Pycharm

Please install python3

https://www.python.org/downloads/

and Pycharm Community Edition

https://www.jetbrains.com/pycharm/download/