

# 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

# 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>

The screenshot shows a REST client interface with tabs for Params, Authorization, Headers (9), Body, Pre-request Script, Tests, and Settings. The Authorization tab is selected, showing a dropdown for 'Basic Auth' and a warning message: 'Heads up! These parameters hold sensitive data. To keep this data secure using variables. [Learn more about variables](#)'. Below this, the 'Username' field contains 'devnetuser' and the 'Password' field is masked with dots. The Headers tab is also visible, showing a table of headers.

	KEY	VALUE
<input checked="" type="checkbox"/>	Authorization ⓘ	Basic ZGV2bmV0dXNlcjE6Q2lzY28xMjMh
<input checked="" type="checkbox"/>	Cache-Control ⓘ	no-cache

# 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:

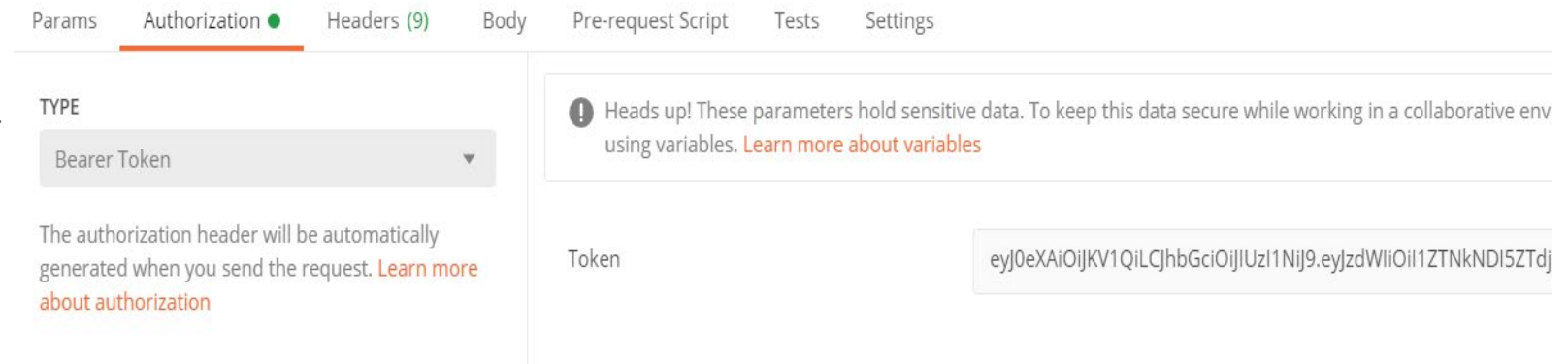
The screenshot shows the 'Authorization' tab in a REST client. The 'TYPE' dropdown is set to 'API Key'. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)'. The 'Key' field is 'X-Auth-Key', the 'Value' field contains a long alphanumeric string, and the 'Add to' dropdown is set to 'Header'. Below the fields, a note says: 'The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)'.

The screenshot shows the 'Headers' tab in a REST client. It displays a table of headers with columns for KEY, VALUE, and DESCRIPTION. The first header is 'X-Auth-Key' with the same long alphanumeric value as in the previous screenshot. The second header is 'Cache-Control' with the value 'no-cache'. Both headers have a checkmark in the first column and an information icon.

	KEY	VALUE	DESCRIPTION
✓	X-Auth-Key ⓘ	zsdwwef123sdfs4567893fedsfetc5f0q5000bfo0c38d90bbeb	
✓	Cache-Control ⓘ	no-cache	

# 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



The screenshot shows the 'Authorization' tab in Postman. The 'TYPE' dropdown is set to 'Bearer Token'. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative env using variables. [Learn more about variables](#)'. The 'Token' field contains the value: 'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWUiOiI1ZTNkNDI5ZTdj'.

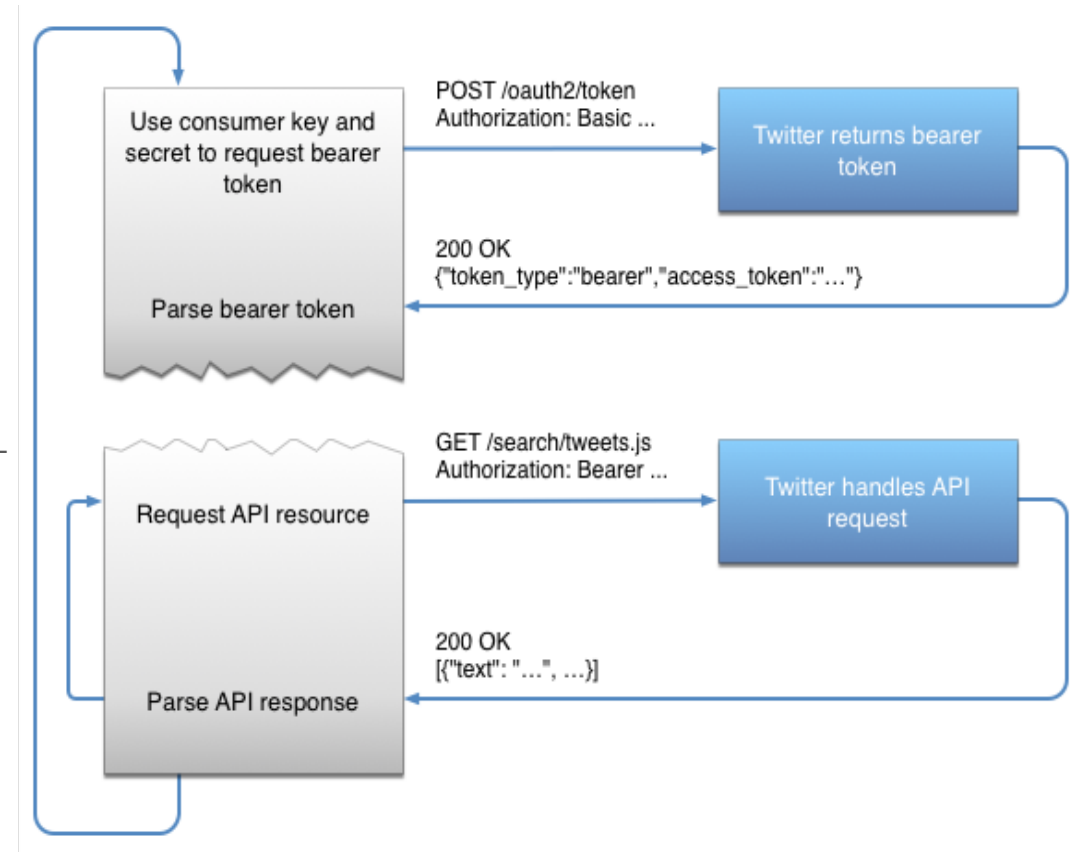
- Like Basic, sent in Auth header
- Value is **Bearer**<space><token>

	Params	Authorization ●	Headers (9)	Body	Pre-request Script	Tests	Settings
	KEY		VALUE				
<input checked="" type="checkbox"/>	Authorization ⓘ		Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWUiOiI1ZTNkNDI5ZTdj				
<input checked="" type="checkbox"/>	Cache-Control ⓘ		no-cache				

# 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

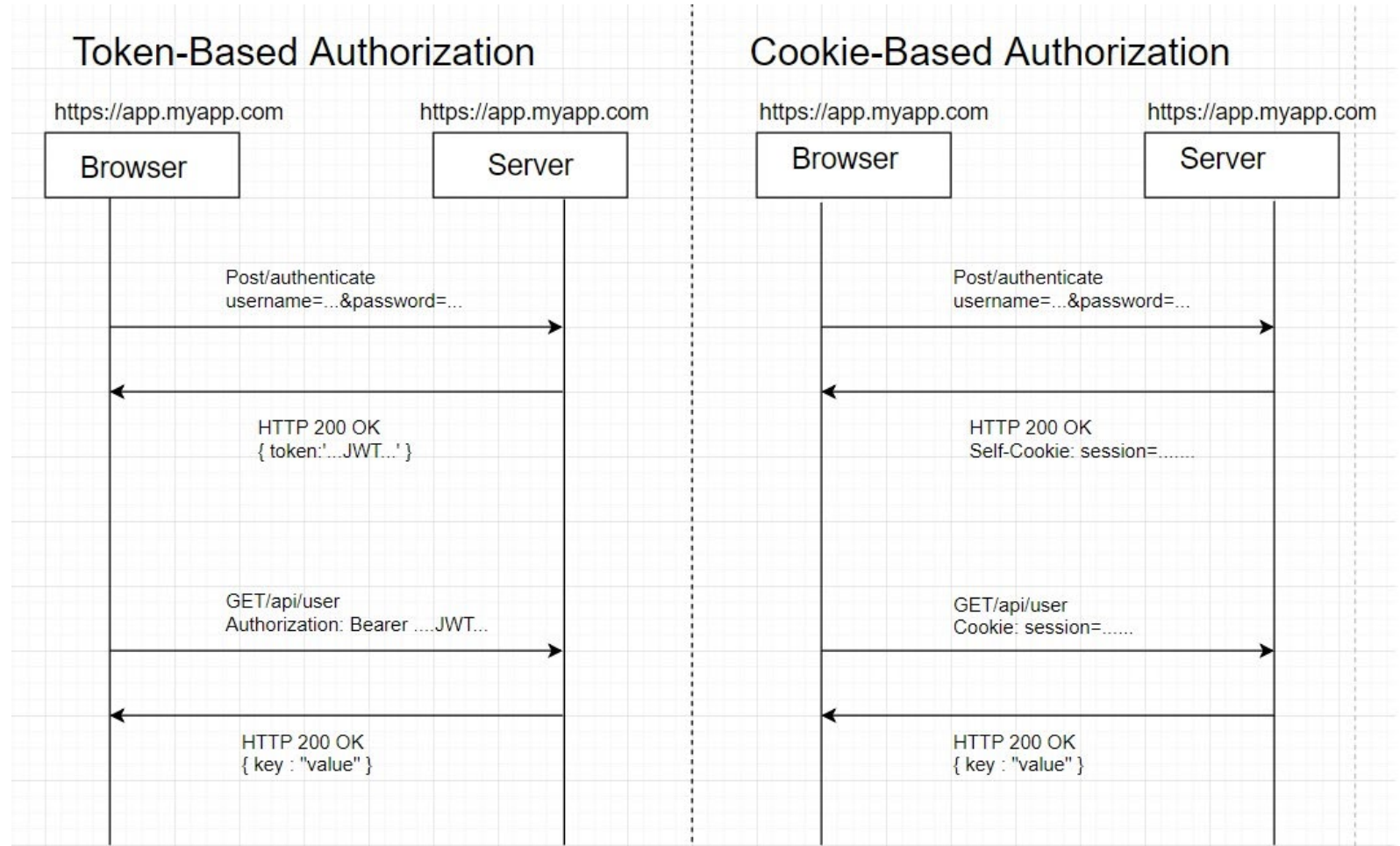


Considered the most secure and scalable method, used by all public cloud providers and most of SaaS providers

# 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





# 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

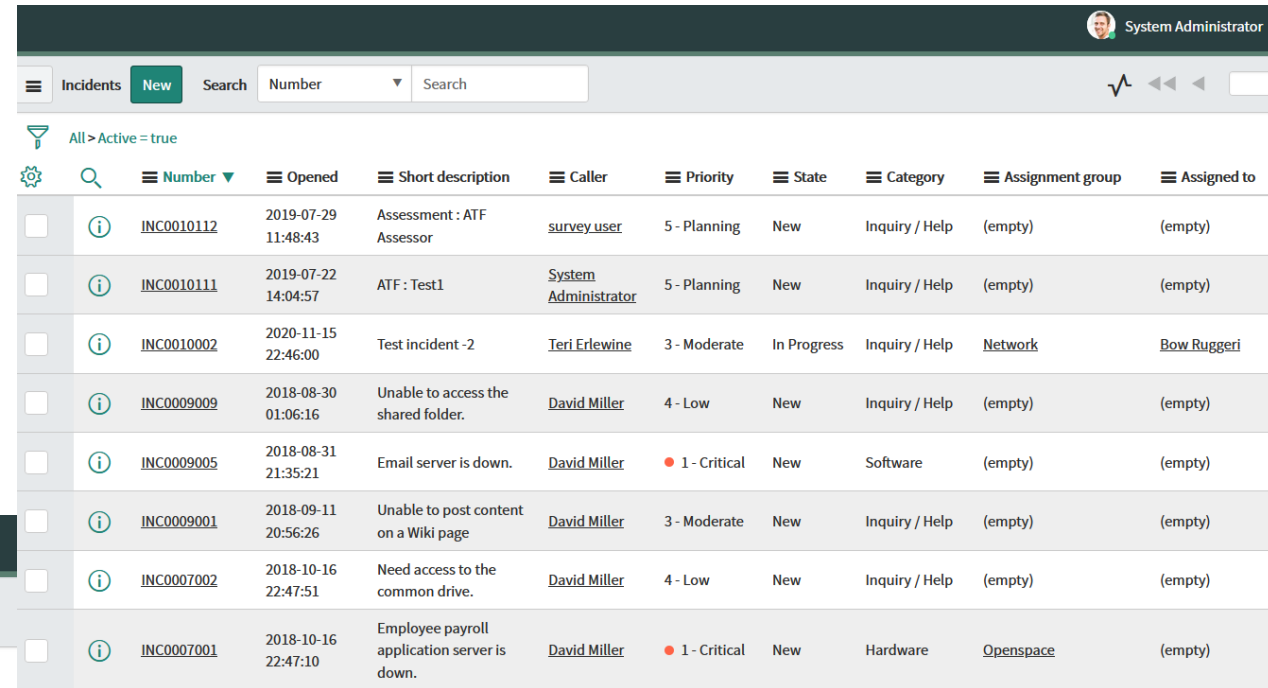
# ServiceNow

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>

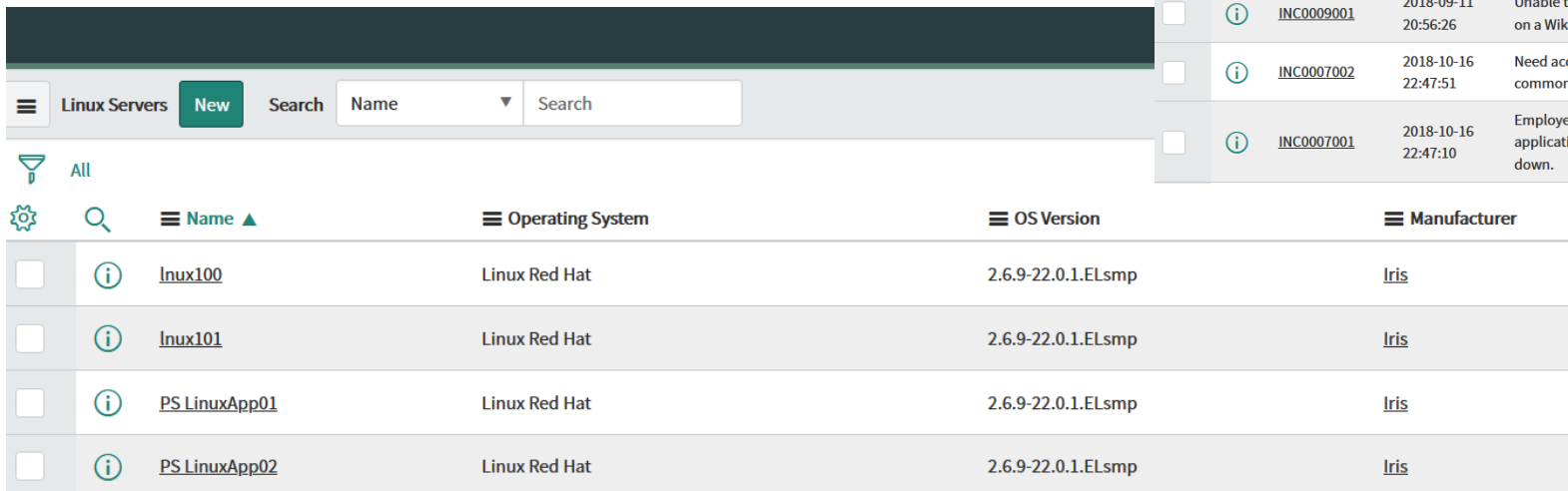
We'll use Incidents and Linux Servers tables

ServiceNow supports Basic Auth and OAuth,  
we will use Basic



This screenshot shows the 'Incidents' table in the ServiceNow interface. The table lists various incidents with columns for Number, Opened, Short description, Caller, Priority, State, Category, Assignment group, and Assigned to. The user 'System Administrator' is logged in.

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



This screenshot shows the 'Linux Servers' table in the ServiceNow interface. The table lists various Linux servers with columns for Name, Operating System, OS Version, and Manufacturer. The user 'System Administrator' is logged in.

		Name	Operating System	OS Version	Manufacturer
<input type="checkbox"/>	<a href="#">i</a>	<a href="#">lnux100</a>	Linux Red Hat	2.6.9-22.0.1.ELsmp	<a href="#">Iris</a>
<input type="checkbox"/>	<a href="#">i</a>	<a href="#">lnux101</a>	Linux Red Hat	2.6.9-22.0.1.ELsmp	<a href="#">Iris</a>
<input type="checkbox"/>	<a href="#">i</a>	<a href="#">PS LinuxApp01</a>	Linux Red Hat	2.6.9-22.0.1.ELsmp	<a href="#">Iris</a>
<input type="checkbox"/>	<a href="#">i</a>	<a href="#">PS LinuxApp02</a>	Linux Red Hat	2.6.9-22.0.1.ELsmp	<a href="#">Iris</a>

# ServiceNow API Doc

Search for API

You can try real API calls

The screenshot displays the ServiceNow REST API Explorer interface. On the left, a sidebar shows a search bar with 'api' entered and a list of categories: System Web Services, REST (expanded), REST API Explorer, Scripted Web Services, Scripted REST APIs, and REST & SOAP API Analytics. The main panel is titled 'REST API Explorer' and contains filters for Namespace (now), API Name (Table API), and API Version (latest). Below these filters, a link 'Retrieve records from a table (GET)' is visible. On the right, the 'Table API' section explains that it allows CRUD operations and provides a sample GET request: 'GET https://dev78092.service-now.com/api/now/table/{tableName}'. Below this, there is a 'Prepare request' section with a table for parameters.

Parameter	Value
tableName	

Below the API Explorer, a 'Scripted REST Service Application Service' section is visible. It includes 'Related Links' (Enable versioning, Explore REST API, API analytics) and a table of resources. The table has tabs for 'Resources (6)', 'Request Headers', and 'Query Parameters (2)'. The 'Resources' tab is active, showing a list of resources with columns for Name, HTTP method, Relative path, and Resource path.

Name	HTTP method	Relative path	Resource path
CreateOrUpdate	POST	/app_service/create	/api/now/cmdm/app_service/create
Find Service	GET	/csdm/app_service/find_service	/api/now/cmdm/csdm/app_service/find_service

# 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>

The screenshot displays the Cisco DNA Center web interface. The top navigation bar includes tabs for DESIGN, POLICY, PROVISION (selected), and ASSURANCE. Below this, there are sub-tabs for Devices, Fabric, and Services. The left sidebar shows a hierarchy starting with 'Global' and listing various locations: Unassigned Devices, Baltimore, Bruxelles, Luxembourg, Owings Mills, San Jose, and Washington DC. The main content area is titled 'DEVICES (2)' and 'FOCUS: Inventory'. It features a 'DEVICE TYPE' filter with options: All, Routers, Switches, APs, and WLCs. Below the filter, there is a table of devices with columns: Device Name, IP Address, Support Type, and Device Family. The table lists two devices: 'cat\_9k\_1' (c3850, Switch\_9300) and 'cat\_9k\_2' (c3850, Switch\_9300). At the bottom right, a sidebar menu is visible with options: About, LEARN, API Reference, Developer Resources, and Help.

Device Name	IP Address	Support Type	Device Family
cat_9k_1 c3850, Switch_9300	10.10.22.66	Supported	Switches and Hubs
cat_9k_2 c3850, Switch_9300	10.10.22.70	Supported	Switches and Hubs

# Practice Basic Auth with Service Now

---

Basic Auth - see [https://dev78092.service-now.com/\\$restapi.do#/code\\_modal](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"`

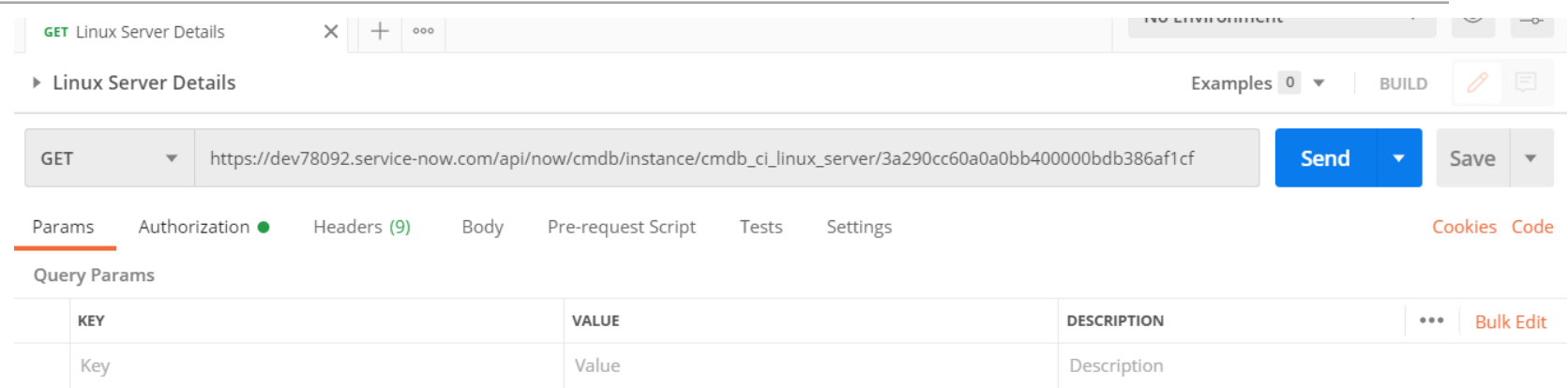


# Postman

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



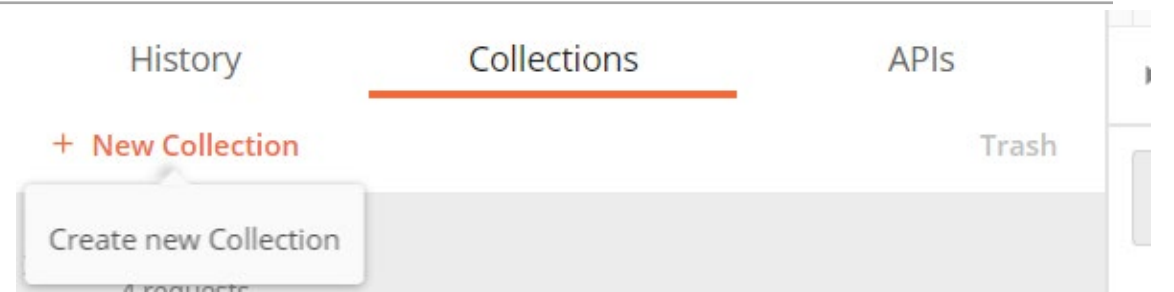
Response



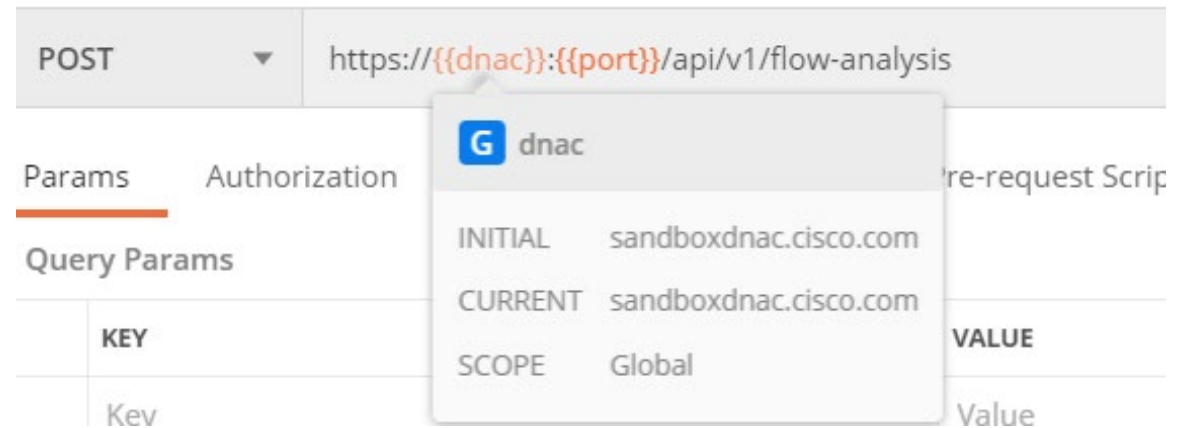
<https://learning.postman.com/docs/sending-requests/requests/>  
<https://www.guru99.com/postman-tutorial.html>

# 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

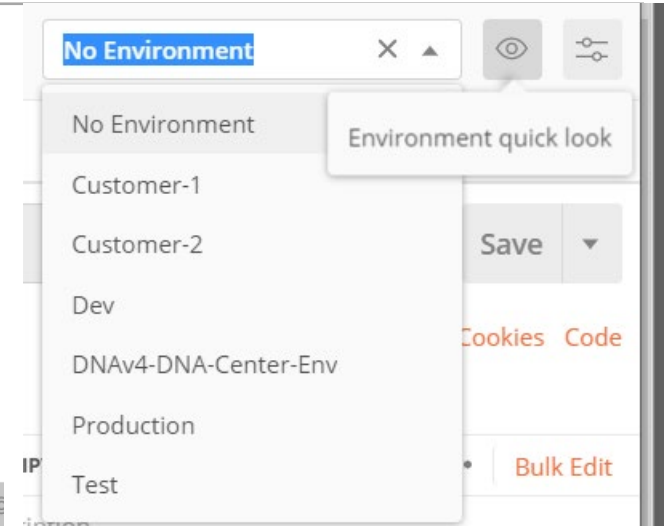
## Two methods to set values to variables

**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

Environment			Add
No active Environment			
An environment is a set of variables that allow you to switch the context of your requests.			
<a href="#">Learn more about environments</a>			
Globals			Edit
VARIABLE	INITIAL VALUE	CURRENT VALUE	
dnac	sandboxdnac.cisco.com	sandboxdnac.cisco.com	
port	443	443	
cisco-dnac-user	devnetuser	devnetuser	



# 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

GET <https://dev78092.service-now.com/api/now/table/incident> Send

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

TYPE  
Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment we recommend using variables. [Learn more about variables](#)

Username: api\_user

Password: .....

☐ Show Password

- Go to Headers Tab

Note Auth Header is populated with Base64 value

Click Send

GET <https://dev78092.service-now.com/api/now/table/incident> Send

Params Auth Headers (9) Body Pre-req. Tests Settings

Headers Hide auto-generated headers

	KEY	VALUE	D	...	Bulk Edit
<input checked="" type="checkbox"/>	Authorization ⓘ	Basic YXBpX3VzZXI6dXNlcnN1cC...			

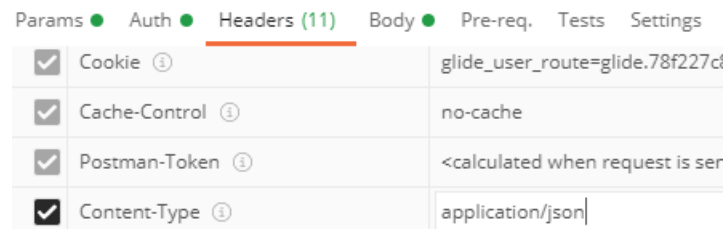
Check the response code and Body content

# Practice with ServiceNow – Postman - POST

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

- Go to Headers Tab

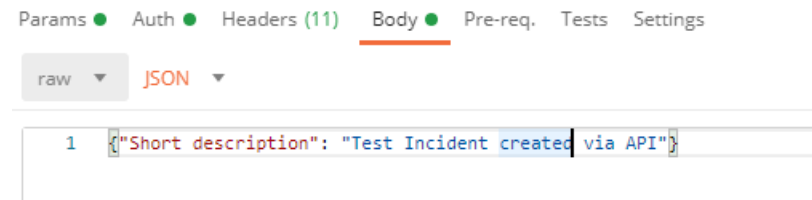
Create Key Content-Type    application/json



2. Go to Body Tab

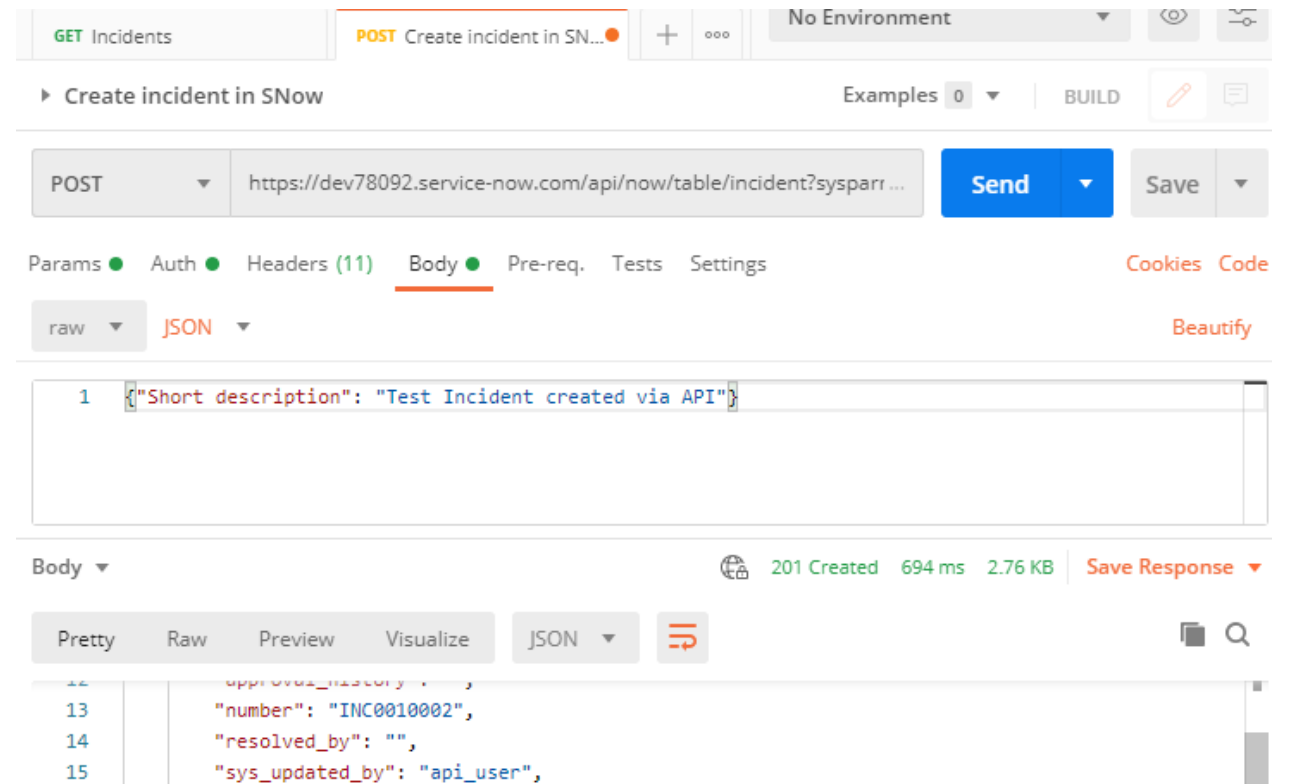
Change type to JSON and enter

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



3. Click Send

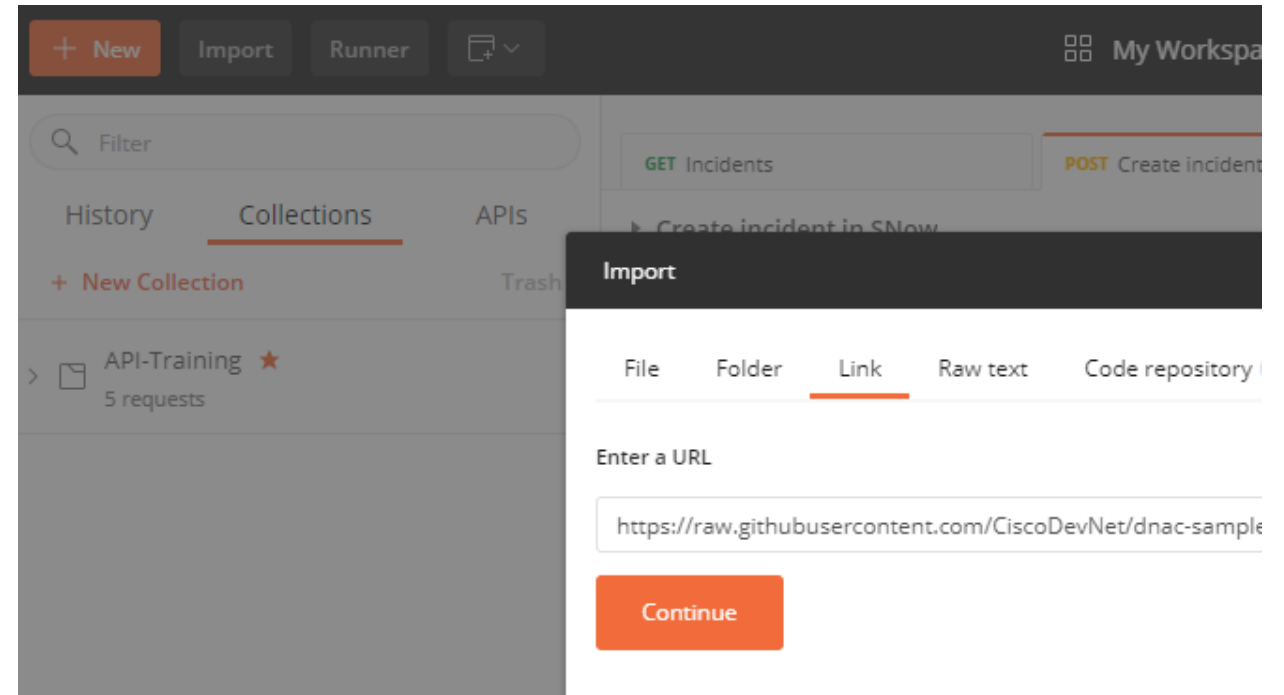
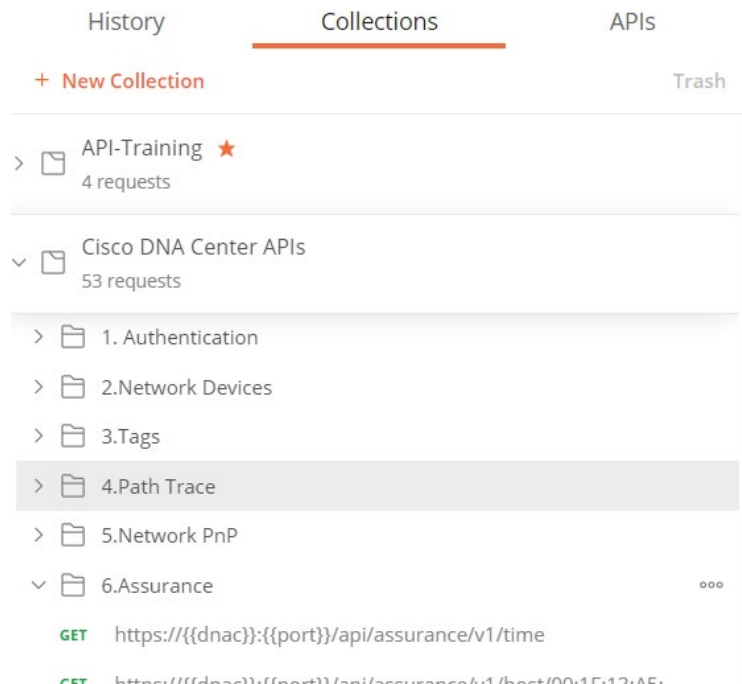
Note you get response code 201 and INC number in Body



# Practice with DNA Centre Sandbox - Postman

Top left corner - Import -> Link

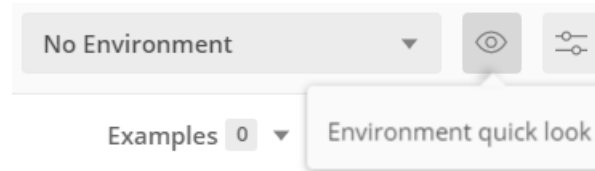
[https://raw.githubusercontent.com/CiscoDevNet/dnac-samples-aradford/master/tools/postman/01-DNAC-Sandbox.postman\\_collection.json](https://raw.githubusercontent.com/CiscoDevNet/dnac-samples-aradford/master/tools/postman/01-DNAC-Sandbox.postman_collection.json)



<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



## MANAGE ENVIRONMENTS

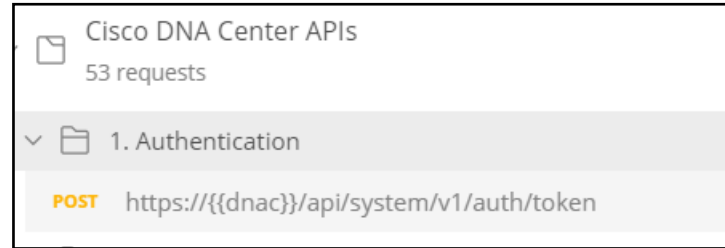
Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. [Learn more about globals](#)

### Globals

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	dnac	sandboxdnac.cisco.com	sandboxdnac.cisco.com			
<input checked="" type="checkbox"/>	port	443	443			
<input checked="" type="checkbox"/>	cisco-dnac-user ⓘ	devnetuser ⓘ	devnetuser ⓘ			
<input checked="" type="checkbox"/>	cisco-dnac-password	Cisco123! ⓘ	Cisco123! ⓘ			

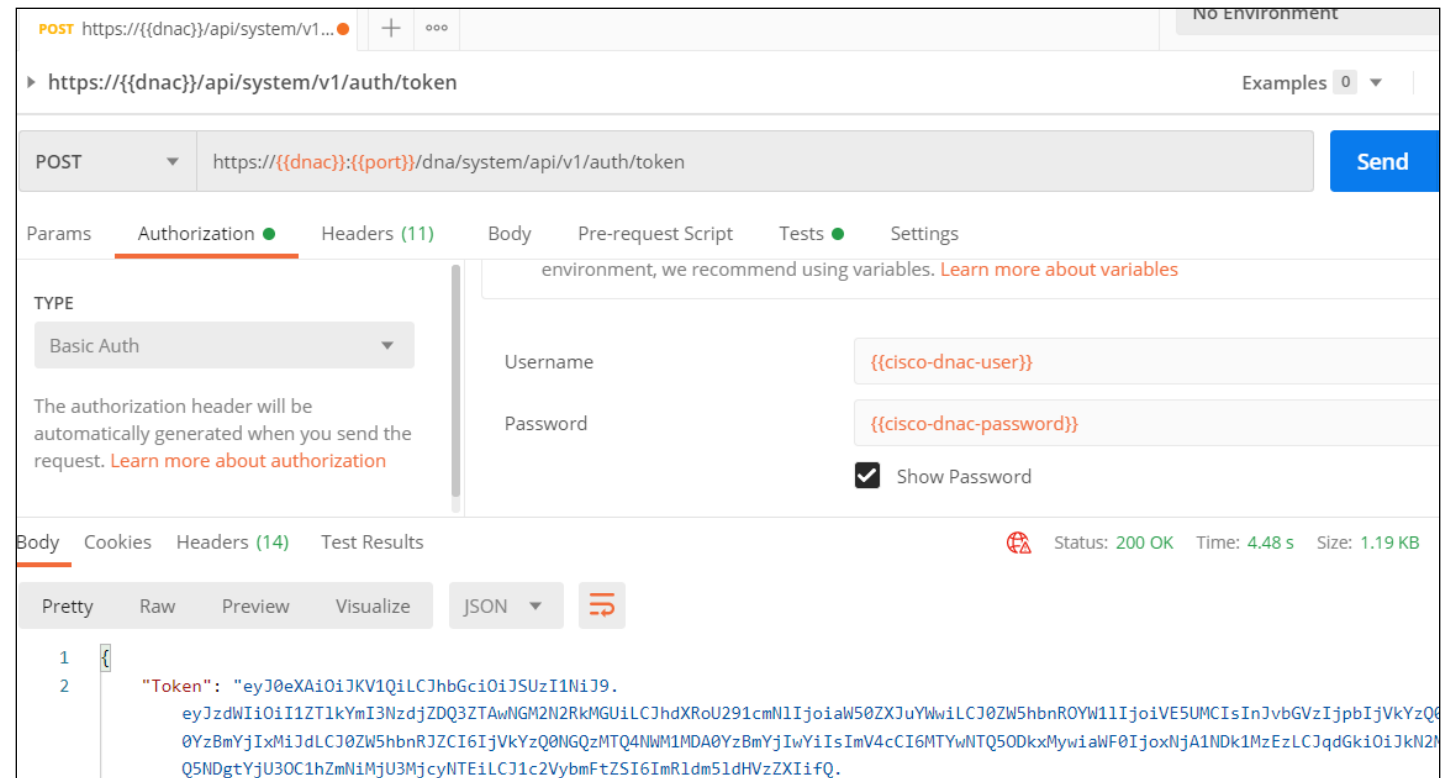
# DNA Centre Sandbox

- 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

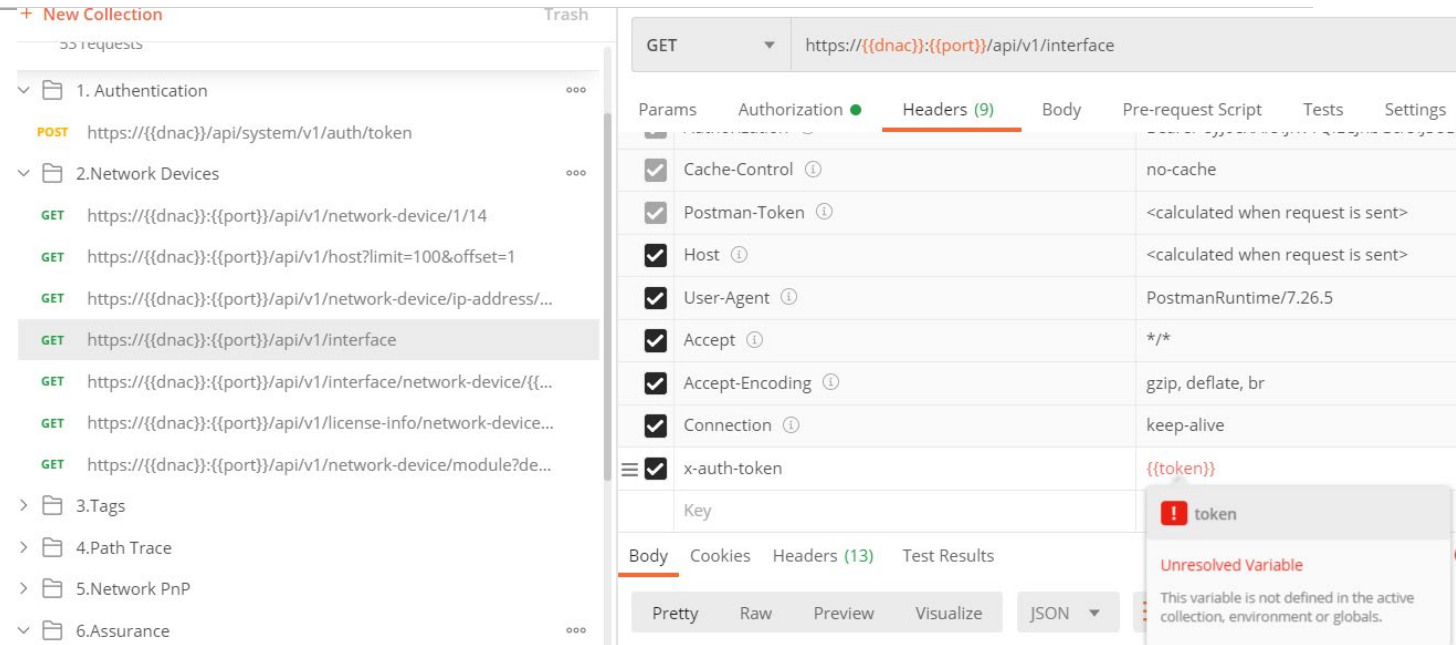
- Once you get a token at the previous step
- Choose any Get request
- Switch to Headers tab
- Note there is no token defined >>>>
- Define the token as global variable

## MANAGE ENVIRONMENTS

Global variables for a workspace are a set of variables that are shared across all collections. They can be viewed and edited by anyone in that workspace.

### Globals

	VARIABLE	INITIAL VALUE ⓘ	CUF
✓	dnac	sandboxdnac.cisco.com	sar
✓	port	443	443
✓	cisco-dnac-user	devnetuser	de
✓	cisco-dnac-password	Cisco123! # ...	Cis
✓	token	eyJ0eXAiOiJKV1QiLCJ... eyJ	
	Add a new 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

The screenshot displays a REST client interface with a request list at the top. The selected request is a GET to `https://{{dnac}}:{{port}}/api/v1/interface`. The 'Headers' tab is active, showing the following headers:

Key	Value	Description
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
<input checked="" type="checkbox"/> x-auth-token	{{token}}	

The 'Body' tab is also active, showing the JSON response in 'Pretty' format:

```
8  {"status": "up",
9   "adminStatus": "UP",
10  "deviceId": "21335daf-f5a1-4e97-970f-ce4eac339f6",
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",
--  }
```

The status bar at the bottom right shows: Status: 200 OK, Time: 1402 ms, Size: 68.86 KB.

# 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/>