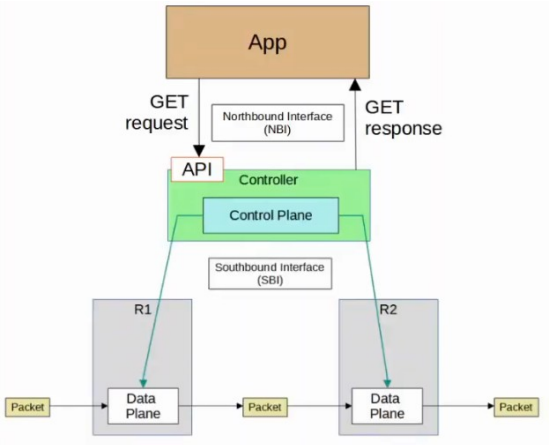
Cours 61 : REST APIs

Dans ce cours nous verrons REST APIs. Les APIs sont utilisés pour permettre aux programmes d'échanger des informations entre eux, ils sont important pour l'automatisation du réseau car ils permettent aux programmes de facilement interagir avec les appareils et le contrôleur.
Les REST APIs sont spécifiquement utilisé pour le Northbound Interface dans une architecture SDN. Nous ferons d'abord un aperçu des API. Les opérations CRUD et les verbes HTTP.
Nous verrons ce qu'est un REST API et ses caractéristiques et en dernier temps nous utiliserons un Cisco Devnet.

Une API (Application Programming Interface) est une interface logiciel qui permet à deux applications de communiquer entre elles. Les APIs sont essentiels pas seulement pour l'automatisation mais pour tout type d'applications.

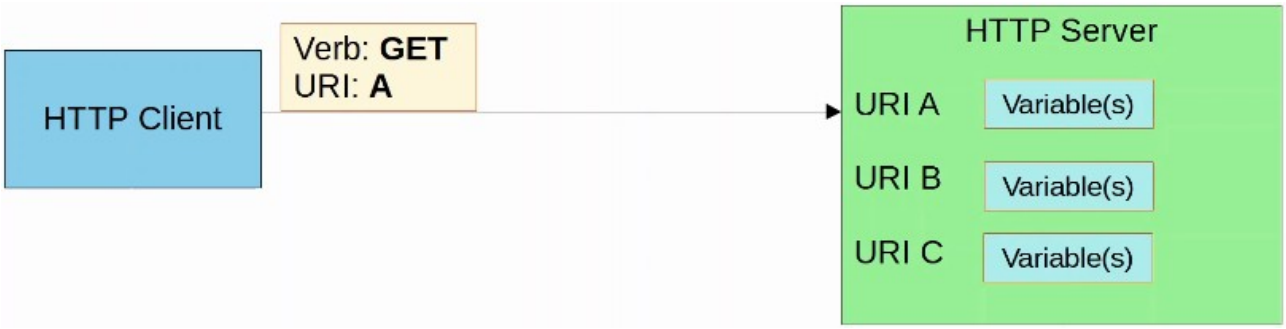


Dans une architecture SDN, les APIs sont utilisés pour communiquer entre les applications et le contrôleur SDN (Par le NBI), et entre le contrôleur SDN et les appareils du réseau (par le SBI).
Le NBI utilise les REST APIs. NETCONF et RESTCONF sont des APIs populaire southbound.
CRUD (Create, Read, Update, Delete) se réfère aux opérations qui fonctionnent en utilisant REST APIs.
- Les opérations Create sont utilisés pour créer de nouvelles variables et afin de placer leur valeur initial. Par exemple créer une variable « ip_address » et placer la valeur à « 10.1.1.1 »
- Les opération Read sont utilisés pour obtenir la valeur de la variable. Par exemple quelle est la valeur de la variable « ip_address » ?
- Les opérations Update sont utilisés pour changer la valeur d'une variable. Par exemple changer la valeur de la variable « ip_address » pour « 10.2.3.4 »
- Les opérations Delete sont utilise pour supprimer des variables. Par exemple supprimer la variable « ip_address »

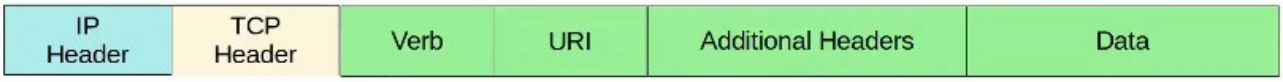
HTTP utilise des verbes ou méthodes pour cartographier les opérations de CRUD.
REST APIs utilise HTTP. Voici un tableau qui permet de résumer les verbes HTTP :

Intérêt	Opération CRUD	Verbe HTTP
Créer une nouvelle variable	Create	POST
Obtenir la valeur d'une variable	Read	GET
Changer la valeur d'une variable	Update	PUT, PATCH
Supprimer une variable	Delete	DELETE

Voyons plus en détail comment ces verbes sont utilisés. Lorsqu'un client HTTP envoie une requête vers un serveur HTTP, l'entête HTTP inclus des informations comme :
- Le verbe HTTP (par exemple GET)
- Une URI (Uniform Resource Identifier) qui indique la ressource auquel il essaie d'accéder.

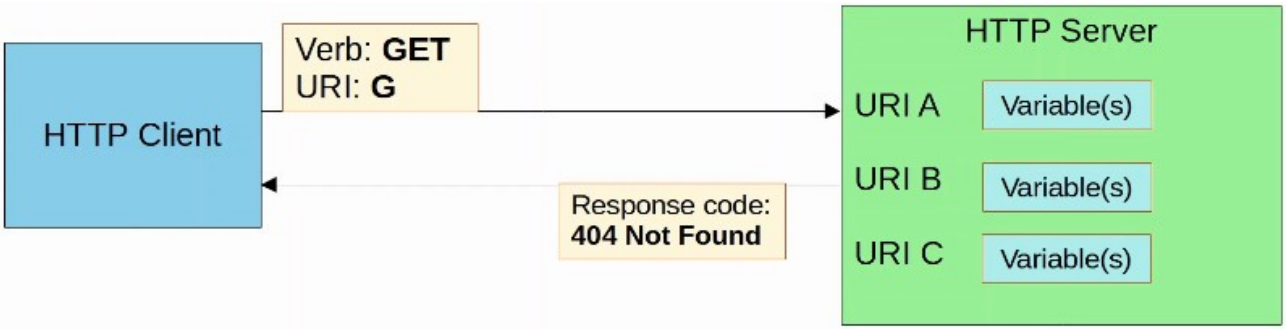


Par exemple le client envoie un message GET afin de demander au serveur de récupérer l'information de la variable contenu dans URI A.
La requête HTTP peut inclure des entêtes supplémentaires qui passent des informations supplémentaire au serveur.



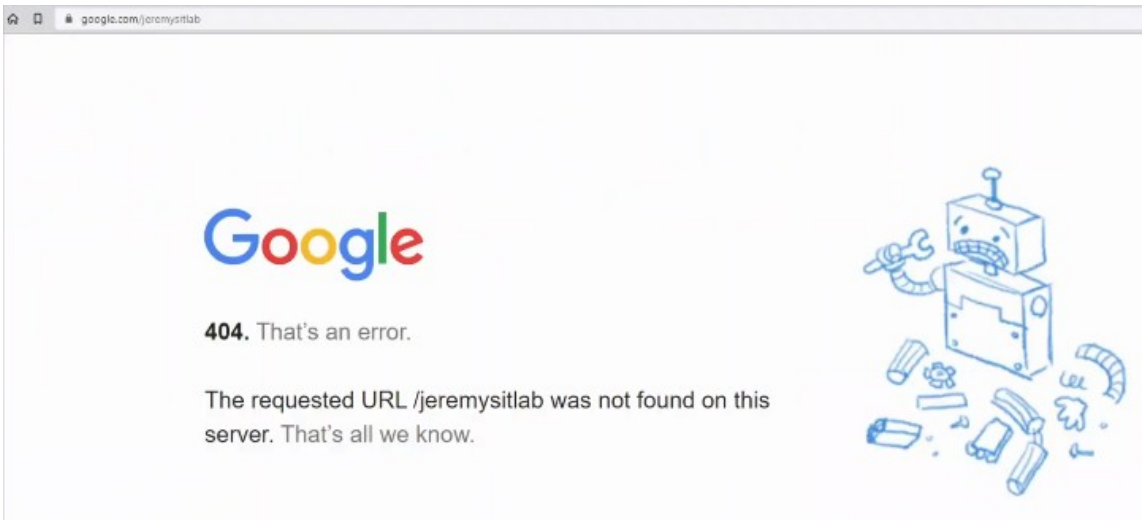
Un exemple pourrait être une entête Accept qui informe le serveur à propos du type de donnée qui peut être envoyé en retour au client.
Par exemple le lien peut accepter des applications json ou applications XML avec cette écriture :
`Accept: application/json` ou `Accept: application/xml`

Lorsque le client REST fait un appelle API (requête) vers un serveur REST, il envoie une requête HTTP tout comme l'écriture précédente.
Les REST APIs n'ont pas à utiliser HTTP pour la communication, HTTP est le choix le plus commun.
La réponse serveur inclus un code de statut qui indique si la requête à réussi ou a échoué, avec d'autres détails.
Le premier chiffre indique la classe de la réponse :
- 1xx informationnel – la requête est reçu, et continue de progresser.
- 2xx succès – la requête a été réussi et bien reçu, comprise et accepté.
- 3xx redirection – une action est nécessaire afin de compléter la requête
- 4xx erreur client – la requête contient une mauvaise syntaxe ou ne peut pas être satisfaite
- 5xx erreur serveur – le serveur n'a pas réussi à répondre pour terminer une requête valide



Voici un exemple dans lequel le client HTTP envoie une requête GET et où le serveur HTTP rpond avec un code d'erreur 404 signifiant qu'il y a erreur de syntaxe.

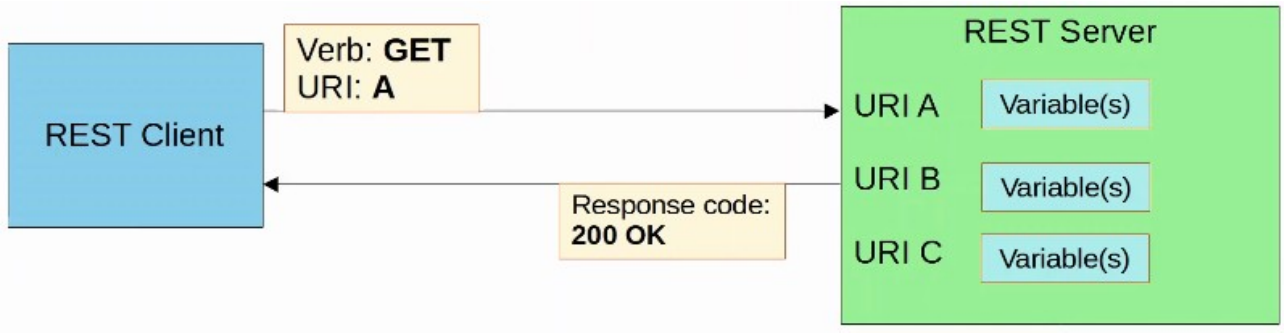
Un exemple de page 404 peut être affiché sur Google lorsque une erreur dans la syntaxe du site :



Voici quelques exemple de chaque classe de réponse HTTP :
- 1xx informationnel
102 Processing : indique que le serveur a reçu la requête et entrain de la traiter, mais la réponse n'est pour le moment pas disponible.
- 2xx Succès
200 OK : indique que la requête a réussi
201 Created : indique que la requête a réussi et que la nouvelle ressource a été crée (en réponse au POST)
- 3xx Redirection
- 301 Moved Permanently indique que la ressource de la requête a été déplacé, et que le serveur indique sa nouvelle localisation.
- 4xx erreur client
403 unauthorized signifie que le client doit s'authentifier pour avoir une réponse.
404 Not Found signifie que la ressource de la requête n'a pas été trouvé.
- 5xx erreur Serveur
500 Erreur serveur Interne signifie que le serveur à rencontrer quelque chose d'imprévu qui ne sais pas comment gérer

REST est l'acronyme de Representational State Transfer. REST APIs sont aussi connu comme REST-based APIs ou RESTful APIs. REST n'est pas spécifique à une API. Il peut décrire plusieurs règles à propos de comment l'API devrait fonctionner.
Il y a 6 contraintes des architecture RESTful :
- Uniform Interface
- Client server
- Stateless
- Cacheable ou non-cacheable
- Layered system
- Code on demand (optionnel)

Nous allons en voir quelques une en détail :
- Client-Server :
Les REST APIs utilisent des architectures client-server
Le client utilise des appelles API (requêtes HTTP) pour accéder aux ressources sur le serveur.
La séparation entre le client et le serveur signifie qu'ils peuvent tout deux changer et évoluer indépendamment l'un de l'autre. Lorsque l'application client change ou que le serveur de l'application change, l'interface entre eux ne doit pas être cassé.
Voici un exemple de client-serveur :



- Stateless :
Les échanges REST APIs neutres.
Cela signifie que chaque échange API est un évènement séparé, indépendant de tout les échanges passés entre le client et le serveur.
Le serveur ne stocke pas d'information à propos des requêtes depuis le client pour déterminer comment il devrait répondre aux nouvelles requêtes.
Si une authentification est requise cela signifie que le client doit s'authentifier avec le serveur pour chaque requête faite. TCP est un exemple de protocole stateful. UDP est un exemple de protocole Stateless. Bien que REST API utilise HTTP, qui utilise TCP (stateful) avec la couche 4 du protocole, HTTP et REST APIs eux même ne sont pas stateful.
Les fonctions de chaque couche sont séparés.

- Cacheable ou non-cacheable
REST API doit supporter le chache de données
Le cache se réfère au stockage de données pour une utilisation future.
Par exemple, l'ordinateur peut cacher plusieurs éléments d'une page web donc il n'a pas à récupérer la page entière chaque fois que l'on la visite.
Cela permet d'améliorer les performance du client et de réduire le chargement du serveur.
Pas toutes les ressources doivent être en cache, mais les ressources cachés doivent être déclarés comme cachés.

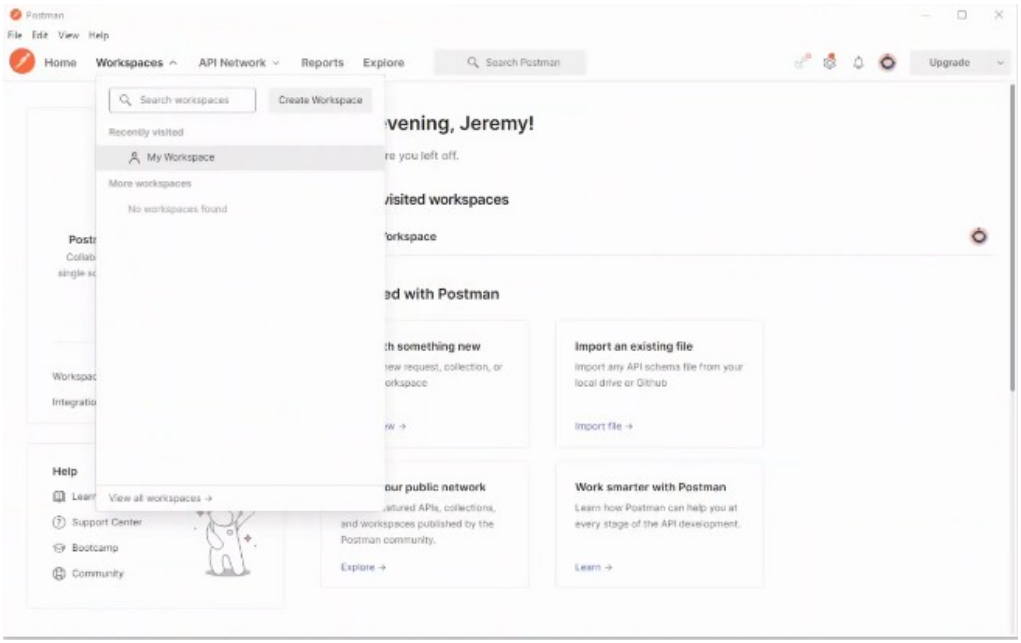
Pour plus d'informations sur les autres type d'architecture RESTful il est possible de visiter ce site : https://restfulapi.net/rest-architectural-constraints/

Pour que les applications puissent communiquer à travers le réseau, les protocoles réseau doivent être facilement utilisable pour faciliter leurs communications.
Pour REST APIs, HTTP(S) est le choix le plus commun.

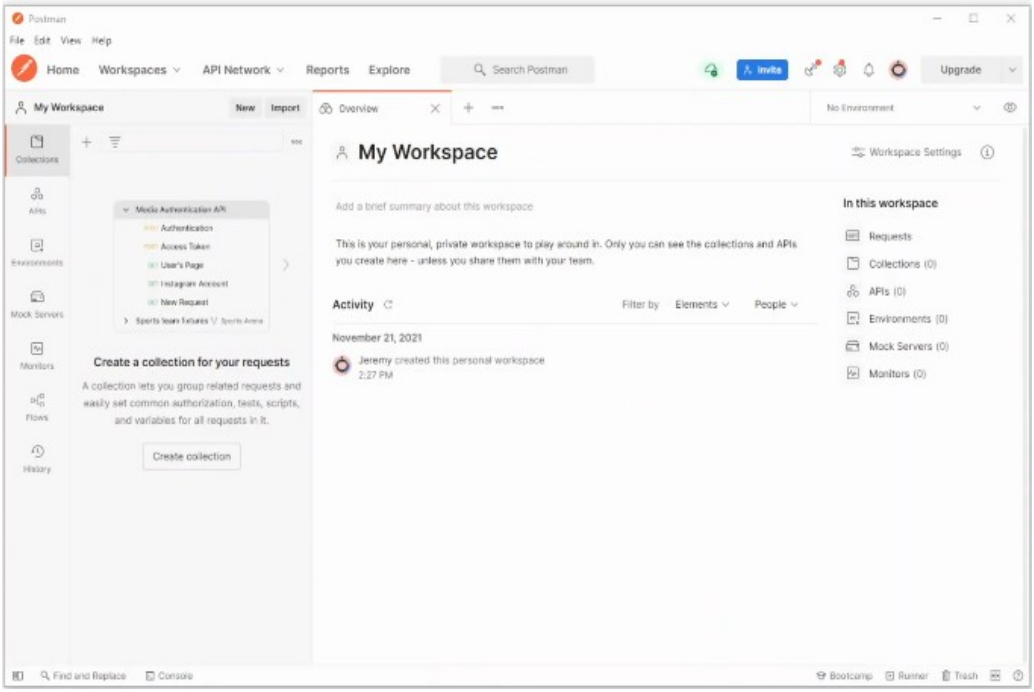
Cisco Devnet est un programme développé par Cisco pour aider les développeurs et professionnels en IT qui veulent écrire leurs applications et développer leur intégration avec des produits Cisco, des plateformes et des APIs.

DevNet offre un tas de ressources gratuites comme des cours, des tutoriels, labs, sandboxes, documentation, etc. pour apprendre à propos de l'automatisation et développer ses compétences. Il existe aussi des certifications DevNet que l'on peut suivre si l'on est intéressé par l'automatisation. Nous utiliserons Cisco DNA Center Sandbox pour envoyer un appelle REST API en utilisant Postman. Un centre DNA est l'un des contrôleur Cisco SDN.
Postman est une plateforme pour construire et utiliser des APIs.

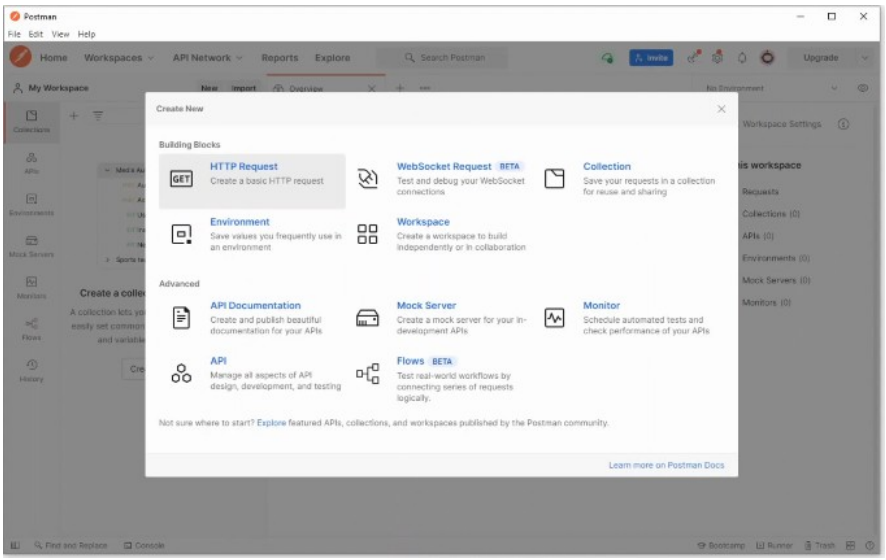
Voici l'interface :



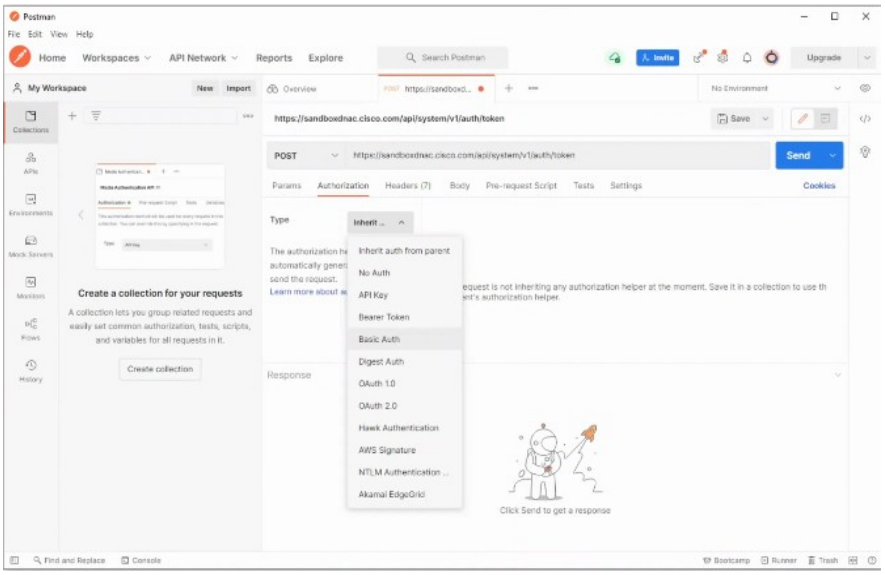
Postman se présente comme suit :



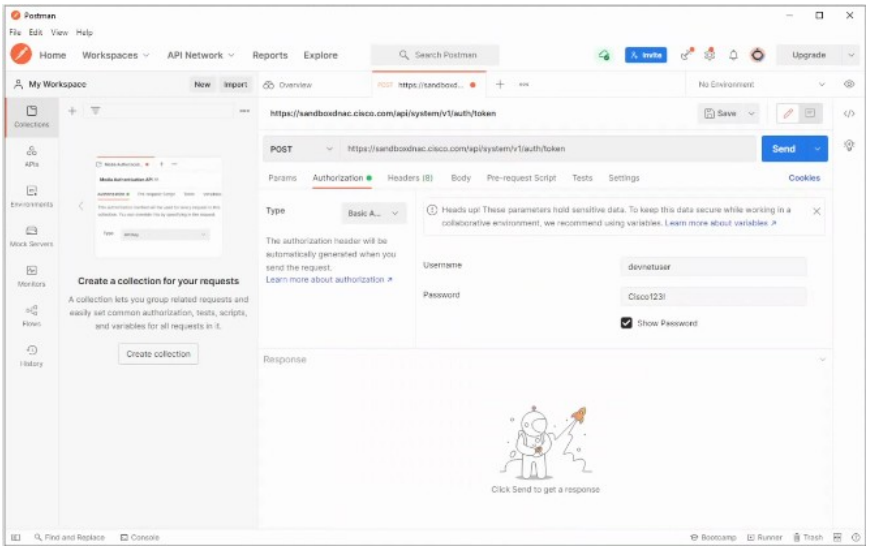
On peut créer différents éléments :



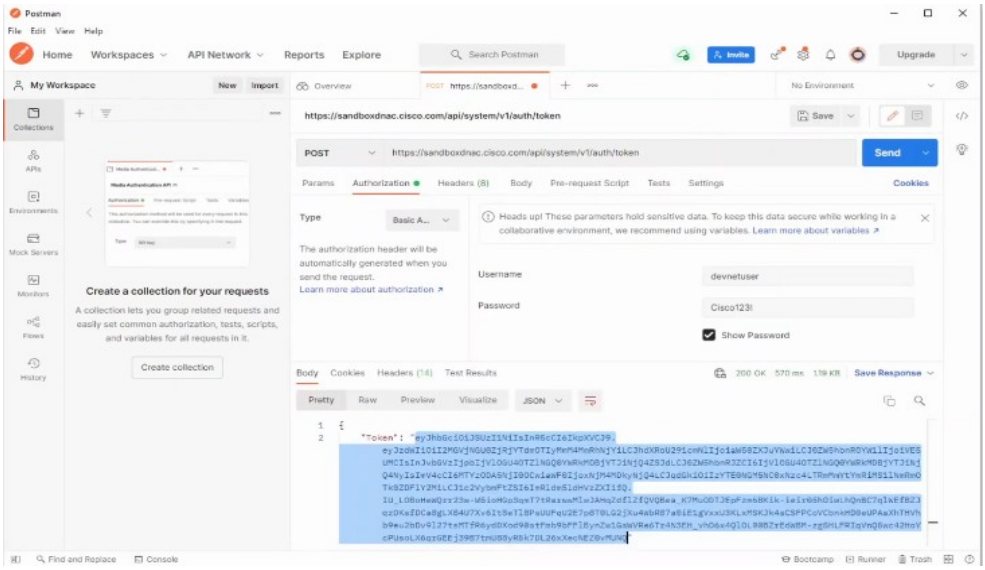
On crée une requête HTTP :



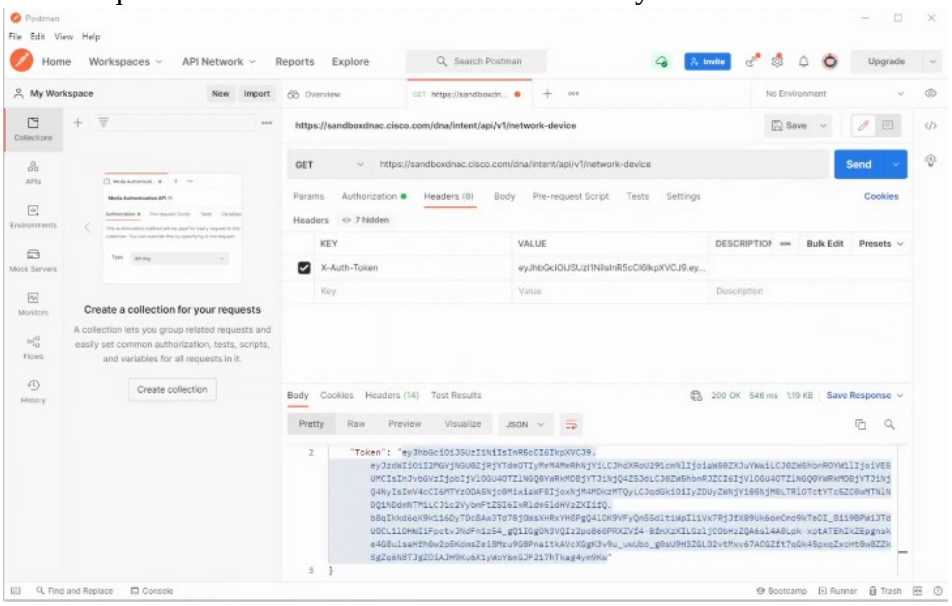
On crée le nom d'utilisateur et le mot de passe et on voie la requête avec « Send »:



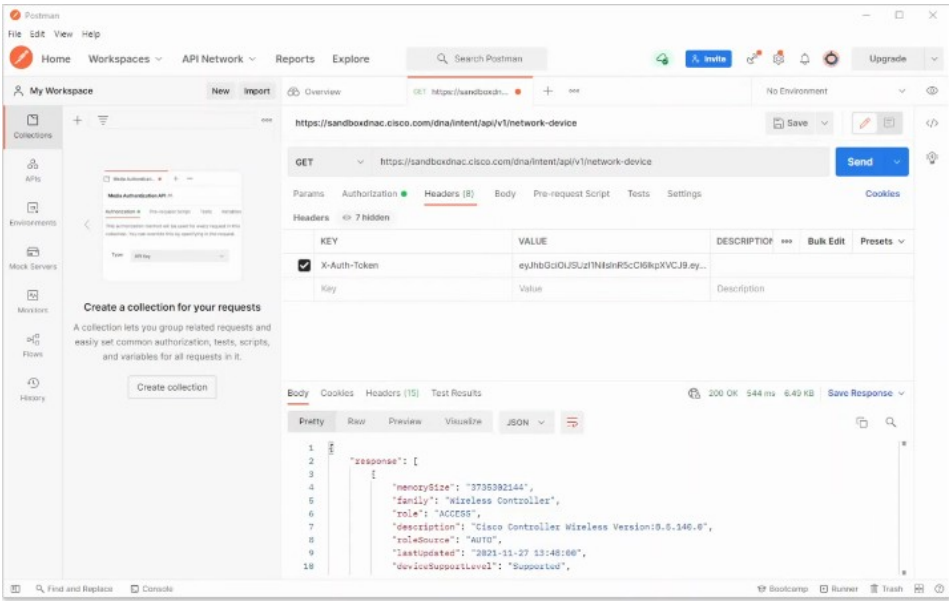
La requête a ici réussi puisqu'il y a un code de réponse « OK » :



Pour envoyer une requête GET on modifie la bannière à envoyer :



La requête a réussi puisqu'il y a un code de réponse OK :



Voici la réponse plus détaillé:


```
{
  "memorySize": "NA",
  "family": "Switches and Hubs",
  "role": "ACCESS",
  "roleSource": "AUTO",
  "lastUpdated": "2021-11-27 13:50:20",
  "deviceSupportLevel": "Supported",
  "softwareType": "IOS-XE",
  "softwareVersion": "17.3.3",
  "macAddress": "84:8a:8d:05:76:00",
  "collectionInterval": "Global Default",
  "inventoryStatusDetail": "<status><general
code=\"SUCCESS\"/></status>",
  "serialNumber": "FCW2220G09V",
  "lastUpdateTime": 1638021020343,
  "hostname": "leaf1.abc.inc",
  "tagCount": "0",
  "tunnelUdpPort": null,
  "uptimeSeconds": 2648950,
  "waasDeviceMode": null,
  "apManagerInterfaceIp": "",
  "bootDateTime": "2021-10-28 18:10:20",
  "collectionStatus": "Managed",
  "locationName": null,
  "managementIpAddress": "10.10.20.81",
  "platformId": "C9300-24U",
  "reachabilityFailureReason": "",
  "reachabilityStatus": "Reachable",
  "series": "Cisco Catalyst 9300 Series Switches",
  "snmpContact": "",
  "snmpLocation": "",
  "upTime": "29 days, 19:40:48.91",
  "apEthernetMacAddress": null,
  "associatedWlcIp": "",
  "errorCode": null,
  "errorDescription": null,
  "interfaceCount": "0",
  "lineCardCount": "0",
  "lineCardId": "",
  "managedAtleastOnce": true,
  "location": null,
  "type": "Cisco Catalyst 9300 Switch",
  "managementState": "Managed",
  "instanceUuid": "aa0a5258-3e6f-422f-9c4e-9c196db115ae",
  "instanceTenantId": "5e8e896e4d4add00ca2b6487",
  "id": "aa0a5258-3e6f-422f-9c4e-9c196db115ae"
}
```


On peut voir différentes informations comme la « famille » d’appareil est Switch et Hubs.
Le nom d’hôte est « leaf1.abc.inc »
Le nom de modèle est avec platformID : « C9300-24U »