Learning Connections

school.learningconnections.it
info@learningconnections.it

Introduzione ai linguaggi e agli strumenti per la network automation

Workbook

Panoramica

Learning Connections organizza una serie di webinar ed eventi formativi gratuiti su temi legati alla network automation e al software defined networking.

Questo workbook contiene gli esercizi illustrati dagli istruttori nel corso dell'erogazione dei webinar di livello basic/associate erogati allo scopo di accompagnare gli studenti dei nostri corsi CCNA "tradizionali" nei primi passi verso il mondo della Network Automation.

Repository: https://github.com/Learning-Connections/netdevops-intro

Ultimo aggiornamento: 13 Novembre 2021

Introduzione

Immaginate di dover accedere ad un dispositivo di rete (ad esempio un Cisco IOS-XE) per verificarne lo stato. Come procediamo ? Le risposte più comuni sarebbero “accedere in SSH”, oppure "visualizzare la GUI tramite browser”; alcuni inoltre potrebbero suggerire l’utilizzo di SNMP.

Da oggi proveremo ad accedere ai dati di configurazione, di stato, agli eventi e alle operazioni (RPC) del dispositivo in conformità con il protocollo RESTCONF/NETCONF.

Accedendo in VPN al nostro laboratorio, potrai eseguire interrogazioni RESTCONF utilizzando curl (oppure Postman o altro ambiente di esecuzione in grado di confezionare richieste HTTP) come ad esempio:

curl -k https:// [IP-Addr] /restconf/data/Cisco-IOS-XE-native:native/hostname

Il repository Github nel quale trovare i sorgenti Yang di vari vendor è disponibile al seguente link:

https://github.com/YangModels/yang

Una buona guida per esercitarsi su RESTCONF - YANG è disponibile a questo link

E’ questo un ambito applicativo dove si fa utilizzo ad un gran numero di tecnologie ben note agli sviluppatori software ma, quasi sicuramente, meno a chi opera nel settore networking da diversi anni e opera su architetture e utilizza strumenti “legacy”.

Per questo motivo, proponiamo una serie di esercizi guidati di livello base per accompagnare gli studenti nello studio con il consueto approccio “learning by doing”.

Buon divertimento !

Exercise #1

durata: 30 min.

Obiettivi

- ☐ **Codifica Base64**.
- ☐ **Codifica UTF-8.**

Introduzione

La codifica base64 è ampiamente utilizzata per rappresentare con caratteri stampabili ASCII sequenze arbitrarie di byte.

La codifica UTF-8 è lo standard per i linguaggi JSON, YAML, XML, ecc.. E' necessario comprenderne le caratteristiche di base.

Attività:

- ❖ base64

  Convertire in base64 le sequenze binarie { 0xF5, 0xA401, 0x010203 }

  La codifica base64 rappresentante una sequenza di byte può avere una lunghezza arbitraria di cifre ? Ad esempio, è possibile rappresentare una sequenza di byte tramite i codici “abcde”, “abcde=”, “abcde==” ?

- ❖ UTF-8

  Deriviamo la codifica binaria del carattere Unicode ’✉’, corrispondente al code-point U+2709 esadecimale, 9993 in decimale.

Utilizzare il seguente servizio online per verificarne la correttezza:

https://www.cogsci.ed.ac.uk/~richard/utf-8.cgi.

Utilizzare il sito https://hexed.it/ per visualizzare/editare file contenenti caratteri UTF-8 e verificarne la corrispondente rappresentazione esadecimale dei byte.

Exercise #2

durata: 90 min.

Obiettivi

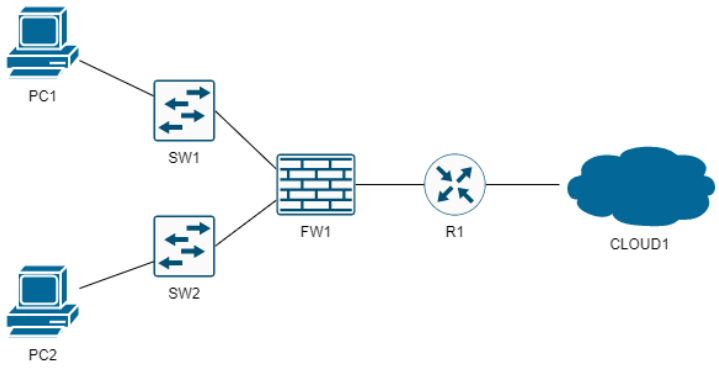
- [ ] **XML**
- [ ] **JSON**
- [ ] **YAML.**

Introduzione

XML, JSON e YAML sono i linguaggi utilizzati per “serializzare” oggetti, ovvero per rappresentarli con una sequenza di caratteri spesso definito “stream”. Differiscono per livello di leggibilità e predisposizione al “parsing” .

Attività:

Si propone la seguente rappresentazione grafica di una topologia di rete LAN:



- rappresentazione XML, JSON e YAML

  Con riferimento alla topologia in figura, produrre tre file di testo nei linguaggi XML, JSON e YAML. Si scelga di rappresentare le informazioni minime.

- Validatori e “Linter”

  Convalidare la **sintassi** dei documenti prodotti utilizzando i seguenti strumenti online:

  https://www.w3schools.com/xml/xml_validator.asp

  https://jsonlint.com/

  http://www.yamllint.com/

- da YAML a JSON

  convertire in JSON gli esempi pubblicati in https://netplan.io/examples/

  verificare la correttezza degli elaborati tramite il sito https://www.json2yaml.com/

Exercise #3

durata: 30 min.

Obiettivi

- ☐ **Git basics**

Introduzione

Git è lo strumento per il controllo di versione più diffuso nella comunità degli sviluppatori. Con esso è possibile gestire repository in forma collaborativa. Questa scheda operativa si propone di mostrare le operazioni di base eseguite da linea di comando.

Attività:

- ❖ Primi passi con Git

  Avviare Git Bash

  configurare lo username e password:

  git config --global user.name "<nome>"

  git config --global user.email "<email>"

  inizializzare un repository a partire da una directory

  git init

  verificare lo stato del repository tramite il comando

  git status

  creare il file vuoto README.md tramite il comando

touch README.md

ripetere il comando ***git status***

aggiungere il file appena creato alla 'staging area'

git add README.md

eseguire il primo commit:

git commit -m "create README.md"

eseguire il comando

git log

ripristinare la versione iniziale del repository

git checkout

annullare quest'ultima operazione tramite :

git checkout -

mostrare l'elenco dei branch: si osservi la presenza dell'HEAD detached

git branch

ripristinare la versione "master"

git checkout master

creare un nuovo branch dal nome "Day1" e mostrare il log

git branch Day1

eseguire delle modifiche ai file ed eseguire il commit

fondere nel branch master le modifiche del Day1

git checkout master

git merge Day1

- Per approfondire le tecniche di branching
  https://learngitbranching.js.org/?locale=it_IT

Exercise #4

durata: 60 min.

Obiettivi

- ☐ **Curl e REST API**

Introduzione

Le API Restful sono ampiamente utilizzate per abilitare una comunicazione “stateless” tra client e server. In questa scheda operativa eseguiremo i primi passi con **curl**, il client HTTP a riga di comando

Attività:

- Primi passi con curl

  Visualizzare la pagina https://reqres.in/ . Questo sito espone una API Restful utilizzabile a scopi didattici.

  Utilizzando **curl** dalla **Git Bash**, si eseguano delle chiamate API descritte nella pagina.

  Si utilizzi l’autorizzazione Basic (non richiesta dalla API) aggiungendo all’header il parametro seguente:

  ```
  -H "Authorization: Basic TG9MOnN1cGVyc2VjcmV0"
  ```

  Utilizzando il comando **for** della shell Bash, mostrare i primi 10 utenti.

- Primi passi con RESTCONF

  Come riportato in seconda pagina di questo workbook, il repository github di riferimento per reperire la documentazione su RESTCONF e YANG è il seguente:

https://github.com/YangModels/yang

All'interno dello stesso repository è pubblicato il riferimento ad una guida Cisco per le operazioni RESTCONF - YANG. Essa è disponibile tramite questo link

Interagiamo con il nostro router IOS-XE provando le seguenti richieste API:

- [ ] curl -k https://10.0.99.193/restconf/ -u "ciscouser:cisco"
- [ ] curl -k https://10.0.99.193/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=1?depth=1 -u "ciscouser:cisco"
- [ ] curl -k https://10.0.99.193/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=1?depth=1 -u "ciscouser:cisco" -v -H "Accept: application/yang-data+json"
- [ ] curl -k https://10.0.99.193/restconf/data/Cisco-IOS-XE-native:native -u "ciscouser:cisco" -v -X "OPTIONS"

Exercise #5

durata: 60 min.

Obiettivi

- ☐ **Guestshell IOS-XE : setup iniziale**

Introduzione

La guest shell di IOS-XE è un ambiente Linux disponibile in forma di Linux Container (LXC) e dotato di un interprete python che per questo viene definito **'on-box'**.

L'utilizzo principale della guest shell è l'automazione della prima installazione del dispositivo di rete (router o switch con IOS-XE), abilitando così lo Zero-touch Provisioning.

Attività:

- ❖ Setup IOS-XE, abilitazione di IOx e della guestshell

  Una configurazione funzionante è disponibile all'indirizzo:
  https://github.com/Learning-Connections/netdevops-intro/blob/main/guestshell/router-config

  Occore eseguire la configurazione iniziale del router CSR1000v abilitando i seguenti servizi:

  - ☐ AAA
  - ☐ HTTP server
  - ☐ accesso SSH
  - ☐ interfaccia VirtualPortGroup
  - ☐ app-hosting
  - ☐ NAT

❖ Installazioni utilità Linux guest shell

Eseguiamo la customizzazione del nostro ambiente guestshell installando una seleziond di strumenti di utilità in ambito linux:

- ☐ sudo yum install openssh
  (per poter generare le chiavi per accesso ssh alla guestshell)
- ☐ sudo yum install nano
- ☐ sudo yum install iperf3
  (nella topologia GNS3, l'appliance "ipterm" dispone anch'esso di iperf3)
- ☐ sudo yum install bind-utils
  (come si può fare a meno di **dig** !)

Può essere necessario installare delle rotte IP nel nostro host Windows e negli appliance di GNS3, ad esempio, nella nostra topologia:

Windows-Privileged-CMD>route add 10.0.100.2 mask 255.255.255.255 10.0.99.193

su Linux

route add 10.0.100.2 gw 10.0.99.193

Exercise #6

durata: 60 min.

Obiettivi

- ☐ **Eseguire uno script Python dalla guest shell**

Introduzione

Avendo installato Git, possiamo clonare un repository contenente i nostri script di esempio.

Eseguiremo il nostro primo script python per eseguire dei comandi della shell IOS

Attività:

- ❖ Esecuzione di uno script Python tramite guestshell

  Entriamo nella guest shell del router ed eseguiamo il seguente comando:

  git clone https://github.com/Learning-Connections/netdevops-intro/

  qualora fosse già presente una vecchia versione del repository, è possibile rimuoverlo tramite il comando **sudo rm netdevops-intro -Rf**

  Visualizzare lo script netdevops-intro/guestshell/script1.py

  Eseguire lo script tramite il comando

  [guestshell@guestshell ~]$ **python netdevops-intro/guestshell/script1.py loop55**

  Uscire dall'ambiente guest shell per rientrare nella CLI di IOS-XE.

  Eseguire nuovamente lo script tramite il comando:

  Cisco-IOS-XE# **guestshell run python netdevops-intro/guestshell/script1.py loop55**

Exercise #7

durata: 60 min.

Obiettivi

- ☐ **Aggiungere uno script Python ad un'applet EEM**

Introduzione

Embedded Event Manager è un tool ben conosciuto dalla comunità dei network engineer perché consente di eseguire script TCL in risposta a vari eventi, quale una notifica cron, un messaggio syslog, una esecuzione manuale ecc..

Oltre al linguaggio TCL, grazie alla guestshell, è adesso possibile eseguire script in Python

Attività:

- ❖ Setup IOS-XE, abilitazione di IOx e della guestshell

Si vuole eseguire il backup della configurazione del router su un server TFTP ogni qualvolta un utente esegue il login al dispositivo:

Configuriamo la seguente applet EEM:

```
event manager applet myTestApplet
event syslog pattern "Login Success"
action 0.0 cli command "enable"
action 1.0 cli  command "guestshell run python /bootflash/myTest.py"
```

Accedendo alla guestshell, copiare rinominandolo il file netdevops-intro/guestshell/script2.py in /bootflash/myTest.py

Il file myTest.py va editato in modo che punti al server TFTP corretto.

Eseguire il login in SSH al router da un dispositivo remoto e verificare l'avvenuta esecuzione dello script tramite il comando:

show event manager history events

Exercise #8

durata: 90 min.

Obiettivi

- ☐ **Cisco Yang-Suite**

Introduzione

Cisco YANG Suite è un'applicazione gratuita realizzata da Cisco per il testing, lo studio di interfacce NETCONF.

Essa viene distribuita in forma di Docker container o di modulo Python da installare in un ambiente virtualizzato.

Attività:

❖ Setup Python Virtual Environment e installazione di YANG Suite

La versione attualmente disponibile dell'applicazione nel repository PIP richiede la versione 3.8 di Pyhton disponibile a questo link.

- ☐ Installare Python 3.8
- ☐ Creare il virtual environment da linea di comando utilizzando il comando seguente:
  C:\Users\<username>\AppData\Local\Programs\Python\Python38\python.exe -m venv yang-suite
- ☐ Eseguire il file **activate.bat** presente all'interno della directory **yang-suite\Scripts**
- ☐ Installare il modulo xmltodict digitando il comando:
  pip install xmltodict
- ☐ ***pip install yangsuite[core]***
- ☐ Avviare il primo setup di YANG Suite eseguendo il file ***yangsuite.exe***
- ☐ Collegarsi alla applicazione in locale tramite browser e navigare all'interno della GUI

❖ Collegamento alla sandbox di Cisco DevNet

Cisco DevNet mette a disposizione l'accesso NETCONF ad un dispositivo IOS-XE utilizzando i seguenti parametri di connessione:

host: **sandbox-iosxe-latest-1.cisco.com**

user: **developer**

password: **C1sco12345**

- ☐ Verificare la raggiungibilità dell'host tramite SSH su porta TCP 830: si dovrebbe ricevere la messaggistica RESTCONF codificata in XML. Interrompere la connessione premendo CTRL+C.
- ☐ Accedere alla GUI di YANG Suite e aggiungere il device sopra descritto cliccando su ***Setup -> Device profiles***
- ☐ Importare i modelli YANG dal dispositivo configurato cliccando su ***Setup -> YANG files and repositories***
- ☐ definire un nuovo YANG set e aggiungere il modulo "Cisco-IOS-XE-native" e le sue dipendenze
- ☐ eseguire una deep validation cliccando su "Validate YANG modules in greater depth"
- ☐ cliccando su Protocols, interrogare (get-config) il dispositivo Cisco Sandbox utilizzando il modulo "ietf-interfaces"
- ☐ - eseguire una RPC per modificare la descrizione della interfaccia "GigabitEthernet2" impostandola a "webinar" . Verificare eseguendo un'ulteriore RPC get-config

Exercise #9

durata: 60 min.

Obiettivi

- ☐ **Primi passi con la libreria Python ncclient**

Introduzione

In questa esercitazione eseguiremo i primi passi con la libreria ncclient in ambiente Python.

Il repository GitHub contiene all'interno della directory NETCONF lo script Python che verrà costruito passo-passo in questa esercitazione.

Per eseguirne il download digitare da GitBash:

> **git clone https://github.com/learning-connections/netdevops-intro**
>
> In caso di errore perché presente una precedente versione del repository in locale, eseguire il comando seguente per rimuovere la vecchia versione e ripetere l'operazione:
>
> **rm netdevops-intro/ -Rf**

ncclient è una libreria Python open source con licenza 'Apache License' . Per installare occorre raggiungere il percorso di esecuzione di **pip** (se non incluso nella variabile di ambiente PATH) ed eseguire il comando:

c:\Users\ < username > \AppData\Local\Programs\Python\Python38\Scripts\pip install ncclient

Utilizzeremo tra le altre la libreria xmltodict che va anch'essa installata tramite pip:

c:\Users\ < username > \AppData\Local\Programs\Python\Python38\Scripts\pip install xmltodict

- ☐ avviare IDLE Shell ed eseguire passo passo i seguenti comandi per la predisposizione dell'ambiente di esecuzione:

```
from ncclient import manager
from pprint import pprint
import xml.dom.minidom
import xmltodict
```

- ☐ Collegarsi alla sandbox di Cisco DevNet utilizzando il comando seguente:

```
my_manager = manager.connect(host="sandbox-iosxe-latest-1.cisco.com",
                             username="developer",
                              password="C1sco12345",
                             hostkey_verify=False)
```

- ☐ E' possibile visualizzare la documentazione della classe manager digitando:

```
help(manager)
```

- ☐ E' pure possibile investigare i contenuti della classe manager digitando:

```
dir(manager)
```

- ☐ Accediamo alla configurazione del dispositivo sandbox digitando:

```
result = my_manager.get_config(source='running')
```

- [ ] Investigare il tipo di dato ricevuto digitando:

  type(result)
  help(result)

- [ ] Visualizzare il contenuto dell'oggetto result in formato XML:

  print(result.xml)
  pprint(result.xml)

- [ ] Utilizzando la libreria xml.dom.minidom, convertiamo la stringa result.xml in un oggetto Document, ispezionandone le funzionalità:
  result_pretty=xml.dom.minidom.parseString(result.xml)

  help(result_pretty)

- [ ] Mostrare il documento XML utilizzando il metodo toprettyxml():

  print(result_pretty.toprettyxml())

- [ ] estraiamo le informazioni relative agli username:

  users =
  result_pretty.getElementsByTagNameNS("http://cisco.com/ns/yang/Cisco-IOS-XE-native
  ",'username')

  type(users)

  user = users[0]

  type(user)

- [ ] Mostriamo a video gli username:

```
for user in users:
    user_dict = xmltodict.parse( user.toxml() )
    print(user_dict['username']['name'])
```

Exercise #10

durata: 60 min.

Obiettivi

- ☐ **Approfondimenti sul tema REST API utilizzando la libreria Python "Requests"**

Introduzione

In questa esercitazione eseguiremo i primi passi con la libreria "***Requests***" (sito web) per Python.

Come per l'esercitazione #9, utilizzeremo l'ambiente IDLE per analizzare le funzionalità della libreria passo-passo.

Nel corso dell'esercitazione utilizzeremo diverse API RESTful pubblicamente su Internet per scopi didattici. Tra queste, il sito " httpbin.org " espone un set completo di servizi utili per studiare in dettaglio le intestazioni HTTP.

Primi passi con requests

- ☐ avviare l'ambiente IDLE
- ☐ digitare "**import requests**". Se si riceve il codice di errore "ModuleNotFound Error" allora vuol dire che la libreria non è installata nel sistema. In questo caso, verificare la versione del runtime Python e procedere all'installazione del modulo come segue:
  - ☐ Avviare il terminale di Windows e raggiungere la directory dove risiede il packet manager PIP:
    cd C:\Users\<USER-NAME>\AppData\Local\Programs\Python\Python38\Scripts

pip install requests

- ☐ Da IDLE, creare un nuovo file cliccando su “File -> new File” e salvarlo con il nome “Esercitazione”

  Qualora fosse necessario accedere a servizi RESTful API tramite HTTPS di sistemi che non dispongono di un certificato TLS valido, occorrerebbe disabilitare i warning sui certificati. Digitare le seguenti istruzioni :

```
import requests

HOST = 'httpbin.org'
ARGS = 'args1=abcd&args2=1234'
URL = 'https://{H}/get?{A}'.format(H=HOST, A=ARGS)
myHeaders = {'Content-Type': 'application/json', 'Accept': 'application/json'}

response = requests.get(URL,headers = myHeaders)

myJSON = response.json()
print(myJSON)
```

- ☐ Eseguire il codice appena creato selezionando dal menu “Run - Run Module” (oppure si prema il tasto funzione F5). IDLE mostrerà l’output dell’esecuzione. Si osservi che la API ha restituito il contenuto della intestazione delle richiesta GET inviata dal nostro programma python tramite la libreria requests.
- ☐ Digitare in IDLE i comandi per:
  - ☐ elencare i valori degli argomenti

```
for arg_value in myJSON['args'].values():
    print(arg_value)
```

  - ☐ mostrare l’origine della richiesta:

```
print(myJSON[‘origin’]
```

  - ☐ mostrare l’intestazione della risposta

```
print(response.headers)
```

- [ ] Creare un nuovo File e digitare il codice seguente:

```
import requests
HOST = 'httpbin.org'
URL = 'https://{H}/post'.format(H=HOST)
myHeaders = {'Content-Type': 'application/json', 'Accept': 'application/json'}
myData = {'FirstName':'Chuck', 'LastName': 'Norris'}

response = requests.post(URL,headers = myHeaders, data=myData)

myJSON = response.json()
print(myJSON)
```

- [ ] Analizzare l'output e indicare cosa c'è di **non corretto** nella formulazione della richiesta. A cosa servono i MIME TYPE **application/x-www-form-urlencoded** e **multipart/form-data ?**

PeeringDB API

E' disponibile una API pubblica per accedere ad un database di peering BGP:

https://www.peeringdb.com/apidocs/

Analizzare la documentazione della API ed eseguire delle richieste di prova. Ad esempio, è possibile eseguire il codice seguente per mostrare gli IX italiani (file peeringDB.py, è necessario importare prettytable)

```
import requests
from prettytable import PrettyTable
myParams = {'country':'IT'}
response = requests.get('https://www.peeringdb.com/api/ix', params=myParams)
myJSONData=response.json()['data']

myTable= PrettyTable()
myTable.field_names=['Name', 'City', 'Website']
for result in myJSONData:
        myTable.add_row([result['name'], result['city'], result['website']])

print(myTable)
```

Autorizzazione accessi API

L'accesso alle API RESTful è di solito protetto da schemi di autorizzazione basati sul metodo "**Basic**" nativo di HTTP (di solito solo se protetti da TLS) e sul metodo "**Bearer**".

Autorizzazione Basic

- [ ] Creare un nuovo File e digitare il codice seguente:

```
import requests

HOST = 'httpbin.org'
URL = 'https://{H}/basic-auth/Chuck%20Norris/guessme'.format(H=HOST)

myHeaders = {'Content-Type': 'application/json',
        'Accept': 'application/json'
        }

print("Sending request without authorization Basic data:\n")

try:
        response = requests.get(URL, headers=myHeaders)
        print(response.headers)

except:
        print('Errore nella richiesta')


myHeaders = {'Content-Type': 'application/json',
        'Accept': 'application/json',
        'Authorization': 'Basic
{base64Encoded}'.format(base64Encoded='Q2h1Y2sgTm9ycmlzOmd1ZXNzbWU=')
        }

print("\nSending request with authorization Basic data:\n")

try:
        response = requests.get(URL,headers = myHeaders)
        myJSON = response.json()
        print(myJSON)
```

```
        print(response.headers)
except:
        print('Errore nella richiesta')
```

Exercise #11

durata: 60 min.

Obiettivi

- [ ] **JWT - JSON Web Token**

Introduzione

In questa esercitazione analizzeremo l'impiego di un particolare tipo di Bearer Token denominato "JWT - JSON Web Token" (RFC 7519)

Si farà uso di una API realizzata in nodeJS e disponibile all'interno del repository.

- [ ] Generare un JWT eseguendo il codice seguente (JWT.py):

```
import requests
import hashlib
import random

# host nel quale è in esecuzione la API demo di Learning Connections
HOST = '10.0.99.130'
PORT = 8000
URL = 'http://{H}:{P}/getToken'.format(H=HOST, P=PORT)

myHeaders = {'Content-Type': 'application/x-www-form-urlencoded',
        'Accept': 'application/json'
        }

"""

        l'utente invia tramite POST in formato x-www-form-urlencoded
        username, seed, hash = sha(seed+sha(pwd))

"""
```

```
mySeed = random.randint(100000,999999)
mySha = hashlib.sha1(b'Learning Connections').hexdigest()
myShaAgain = hashlib.sha1((str(mySeed)+str(mySha)).encode('utf-8')).hexdigest()
print('password hash: '+mySha+' sentHash '+myShaAgain)

myData = {'USERNAME':'Learning Connections',
        'SEED':mySeed,
        'HASH':  myShaAgain
        }
print("Sending POST request for getting JWT token:\n")

try:
        response = requests.post(URL, headers=myHeaders, data=myData)
        myToken = response.json()['TOKEN']
        print('Received token : \n' + myToken)

        print('Now sending a /list request for retrieving private data using token')

        URL = 'http://{H}:{P}/list'.format(H=HOST, P=PORT)
        myParams = {'token': myToken}
        response = requests.get(URL, params=myParams)
        print ('received data: ')
        print (response.json())

except:
        print('Error ')
        print(response)
```

- [ ] Analizzare il token ricevuto tramite il sito ***jwt.io***.
  Si verifichi la firma del token utilizzando la chiave privata: 'xyz123'