OBJECTIVES OF LABS

* Part 1: Build the Network and Verify Connectivity
* Part 2: Use a NETCONF Session to Gather Information
* Part 3: Use ncclient to Connect to NETCONF
* Part 4: Use ncclient to Retrieve Running Configuration (get-config)
* Part 5: Use ncclient to Edit Configuration (edit-config)

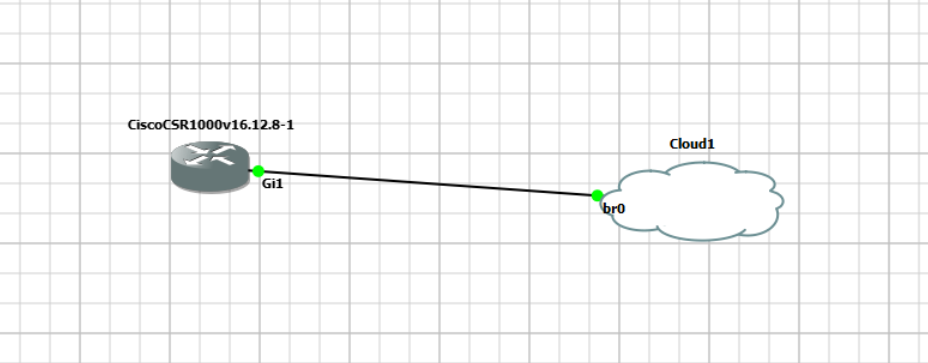
# Netconf

Netconf is a network monitoring protocol which configure network devices like routers, switches programmatically. It removes configuration complexity, vendor specific commands and have better efficiency which is generally found in SNMP.

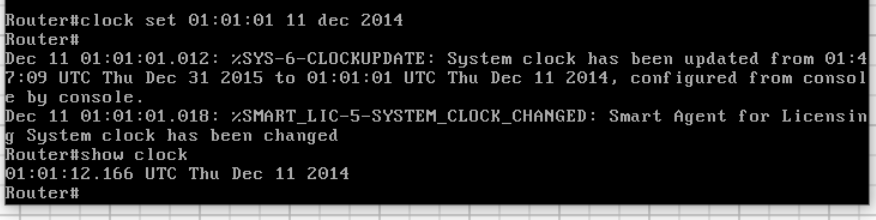
* **Data representation: XML**
* **Protocol used: SSH**
* **Alternative: RestConf**
* **Features:**
  + Device configuration management (CRUD of devices)
  + Transactional changes (support commit and rollback)
  + Standardized and structured with Yang model
* **Benefits:**
  + Automation-Ready (SDN and automation tool like ansible etc)
  + Standardized (Yang models)
  + Error Handling (validation and rollback)
  + Security (due to SSH)
* **Disadvantages:**
  + Complexity (XML and other learning curve)
  + Performance overhead (XML And SSH compare to json and Rest features)
  + Scalability (separate SSH connection for each SSH connection, which limits how many simultaneous sessions a server can manage.)
* **Useful Commands:**
  + **get** - Retrieve operational data e.g., interface status.
  + **get-config** - Retrieve configuration data.
  + **edit-config** - Modify the device configuration.
  + **commit** - Apply candidate configuration to the running state.
  + **discard-changes** - Discard uncommitted changes.
  + **lock and unlock** - Prevent concurrent edits during a session.
  + **delete-config** - Delete a specified data store (e.g., candidate).
  + **close-session**
* **What it solves:**
  + Automated Configuration
  + Transaction Integrity
  + Scalability
  + Efficiency
  + Example: In a large network, manually updating IP addresses on hundreds of routers is inefficient. NETCONF allows a centralized script to update all routers automatically using programs.
* **Tools Required:**
  + **GNS3**
  + **Router with capability to integrate with python (e.g., CSR 1000v)**
  + **Python with ncclient**
  + **GitHub**

# Part 1: Build the Network and Verify Connectivity:

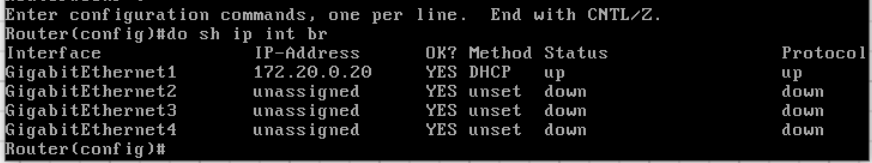
* **Topology:**
  + **Cisco csr 1000v connected to cloud with Giga Ethernet.**
  + **Cloud side is configured with br0.**



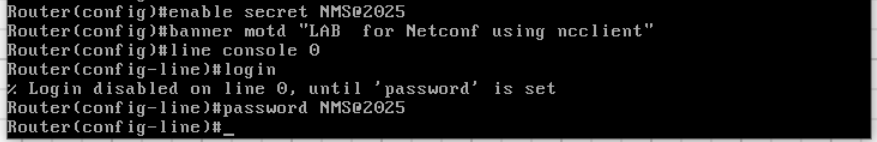
* **Setting up clock**



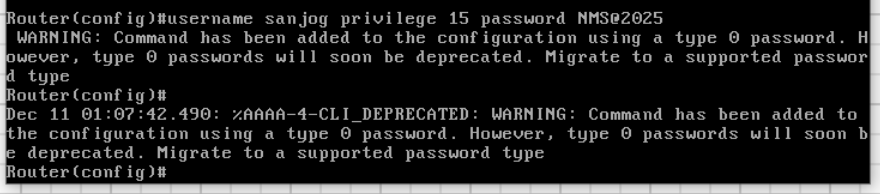
* **Interfaces:** 
  + DHCP automatically assign Ip address to connected interface(gi1)



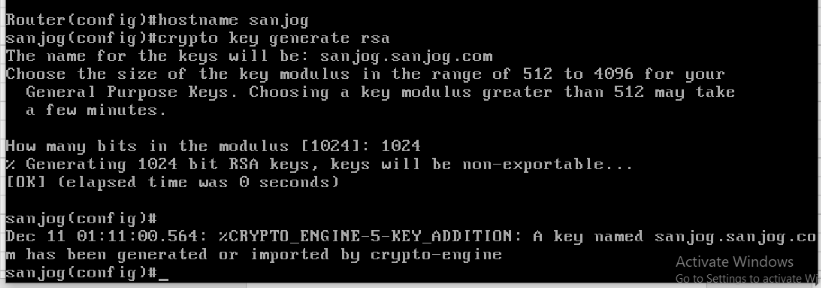
* **Configure SSH with cisco csr 1000v:**
  + Basic Configuration for password and banners



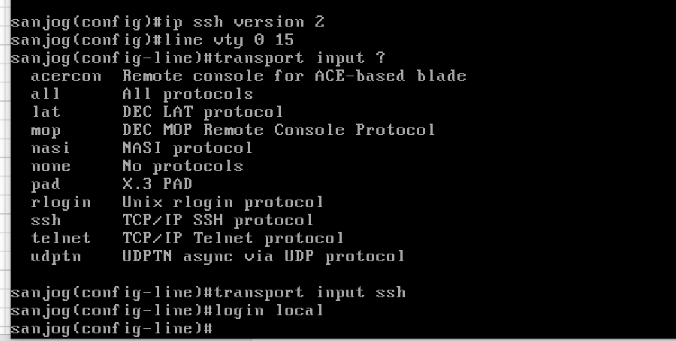
* + Creating a username and provide privileges to user to all tasks.



* + Now generate a RSA (Rivest-Shamir-Adleman) key pairs, which is essential for enabling secure communication protocols such as SSH (Secure Shell) and HTTPS.



* + Configure “ssh version 2” and then enter into virtual terminal.
  + VTY lines control **remote access** (Telnet/SSH) to the Cisco device.
  + “Transport input ssh” Restricts the transport protocol to **SSH** only.
  + login local defines that Configures the router to **authenticate** remote users using **local credentials**

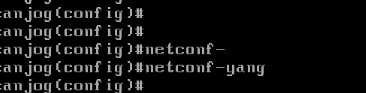


* + Verify User can access router remotely with SSH credentials.

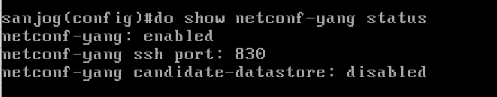
# 

# Part 2: Use a NETCONF Session to Gather Information

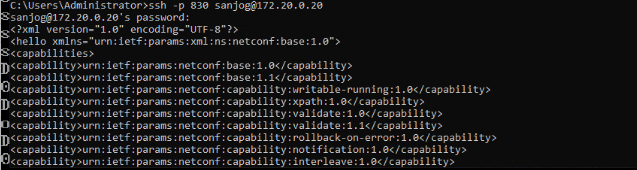
* + Enable Netconf in router:



* + Verify router is enabled for Netconf

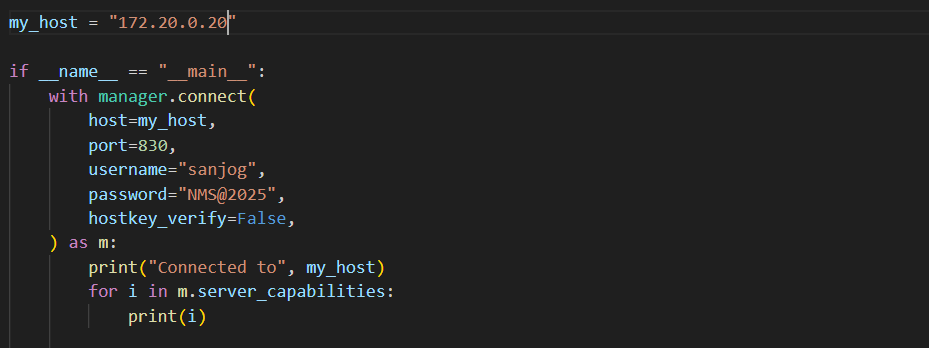


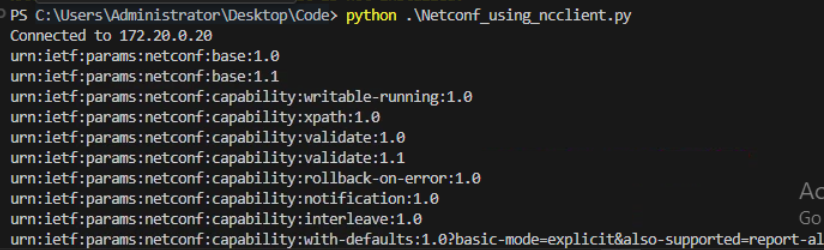
* + Verify Netconf connection



# Part 3: Use ncclient to Connect to NETCONF

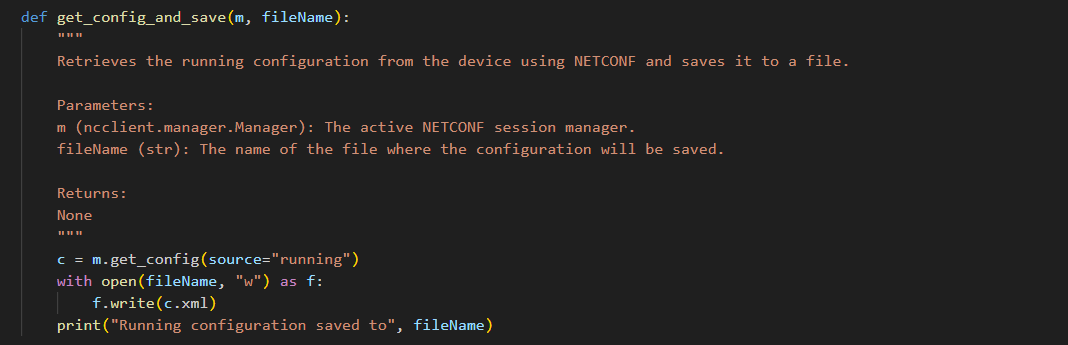
* Using manager from ncclient library to connect with Router and verify with server capability.



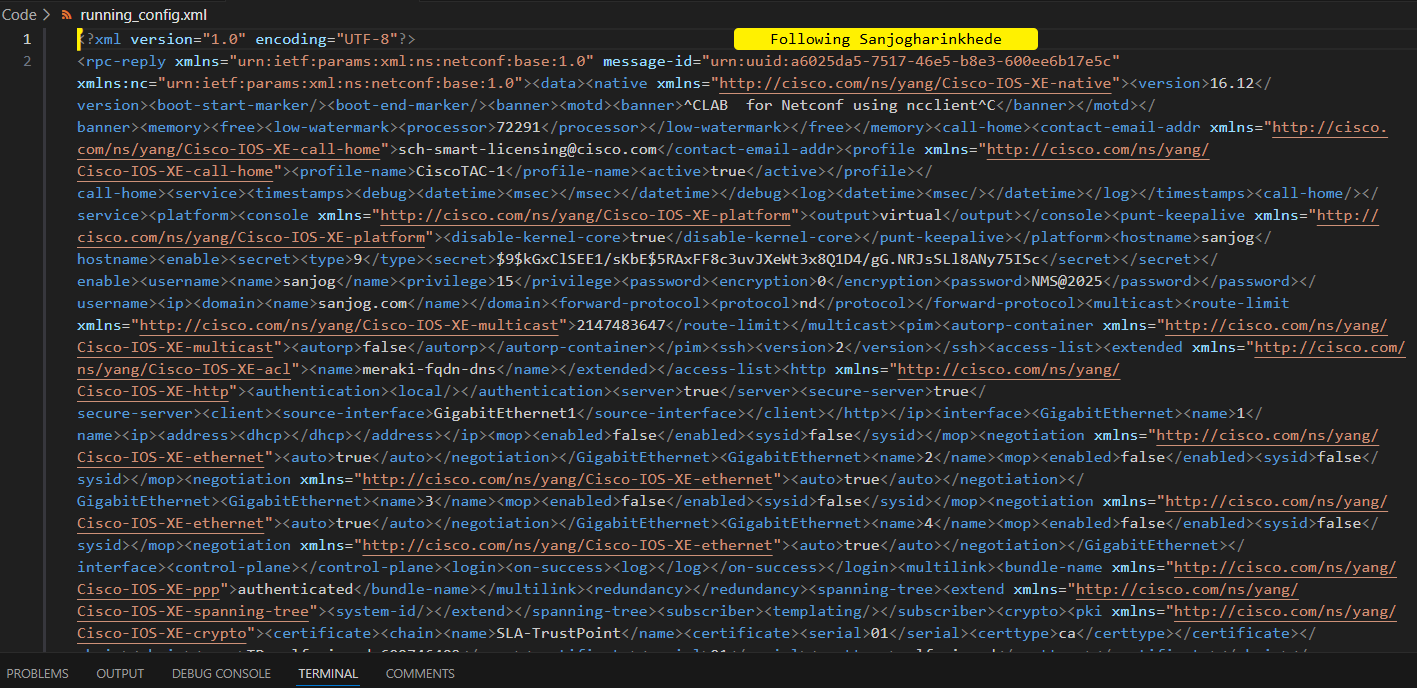


# Part 4: Use ncclient to Retrieve Running Configuration (get-config)

* + Python function for get\_config

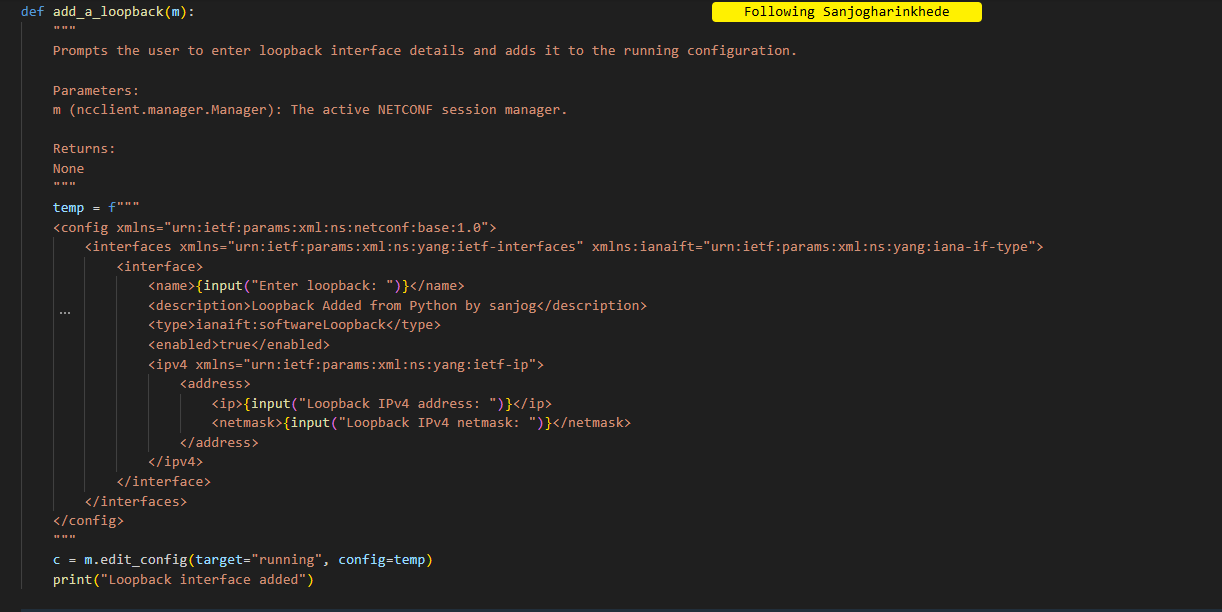


* + Generated xml file for get config

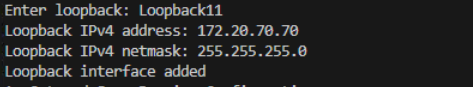


# Part 5: Use ncclient to Edit Configuration (edit-config)

* + Python script for edit config
  + It creates a loopback address with user input for name ip and subnet mask



* + User response on excuting the add\_a\_loopback(m) function:



* + Verify in Gns3

