Cheat Sheet for comprehensive Cisco DevNet Certifications - DevNet Associate

1. Introduction to Cisco DevNet

- **DevNet Overview**

- Cisco's platform for developers to build, test, and deploy network automation solutions.
- Focuses on APIs, automation, and software development practices.

- **Key Components**

  - **DevNet Sandbox**: Free, always-on labs for hands-on experience.

  - **DevNet Learning Labs**: Guided tutorials and exercises.

  - **DevNet Code Exchange**: Open-source code samples and projects.

---

2. Networking Fundamentals

- **Network Topologies**

  - **LAN**: Local Area Network

  - **WAN**: Wide Area Network

  - **VLAN**: Virtual LAN

  - **VPN**: Virtual Private Network

- **Network Devices**

  - **Router**: Connects different networks.

  - **Switch**: Connects devices within a network.

  - **Firewall**: Protects networks from unauthorized access.

  - **Access Point**: Provides wireless connectivity.

- **Protocols**

  - **TCP/IP**: Transmission Control Protocol/Internet Protocol

  - **HTTP/HTTPS**: Hypertext Transfer Protocol (Secure)

- **DNS**: Domain Name System
- **DHCP**: Dynamic Host Configuration Protocol

---

3. Python Programming for Network Automation

- **Basic Syntax**
  - **Variables**: `x = 10`
  - **Data Types**: `int`, `float`, `str`, `list`, `dict`
  - **Control Structures**: `if`, `for`, `while`
  - **Functions**: `def my_function():`
- **Libraries**
  - **Requests**: HTTP library for API interactions.

```
import requests
response = requests.get('https://api.example.com')
```

  - **Paramiko**: SSH library for remote device management.

```
import paramiko
ssh = paramiko.SSHClient()
ssh.connect('hostname', username='user', password='pass')
```

  - **Netmiko**: Simplifies SSH management of network devices.

```
from netmiko import ConnectHandler
device = {
    'device_type': 'cisco_ios',
    'host': 'hostname',
    'username': 'user',
    'password': 'pass'
}
connection = ConnectHandler(**device)
```

- **Examples**
  - **Ping a Device**:

```
    import os
    os.system("ping -c 4 192.168.1.1")
```

- **Get Interface Status**:

```
    output = connection.send_command('show ip interface brief')
    print(output)
```

---

4. RESTful APIs and HTTP Basics*

- **HTTP Methods**
  - **GET**: Retrieve data.
  - **POST**: Submit data.
  - **PUT**: Update data.
  - **DELETE**: Remove data.
- **HTTP Status Codes**
  - **200 OK**: Successful request.
  - **400 Bad Request**: Client error.
  - **404 Not Found**: Resource not found.
  - **500 Internal Server Error**: Server error.
- **API Authentication**
  - **Basic Auth**: `Authorization: Basic <base64encoded_username:password>`
  - **Token Auth**: `Authorization: Bearer <token>`
  - **OAuth2**: `Authorization: Bearer <access_token>`
- **Examples**
  - **GET Request**:

```
    response = requests.get('https://api.example.com/data',
auth=('user', 'pass'))
    print(response.json())
```

- **POST Request**:

```
data = {'key': 'value'}
response = requests.post('https://api.example.com/data', json=data)
print(response.status_code)
```

---

5. Cisco DNA Center and Cisco Application Policy Infrastructure Controller (APIC)

- **Cisco DNA Center**

  - **Features**: Network automation, orchestration, and management.

  - **API Endpoints**: `/dna/intent/api/v1/`

  - **Examples**:

```
url = 'https://dnac.example.com/dna/intent/api/v1/network-device'
headers = {'X-Auth-Token': 'your_token'}
response = requests.get(url, headers=headers, verify=False)
```

- **Cisco APIC**

  - **Features**: Centralized policy management for ACI (Application Centric Infrastructure).

  - **API Endpoints**: `/api/node/class/`

  - **Examples**:

```
url = 'https://apic.example.com/api/node/class/fvTenant.json'
headers = {'Cookie': 'APIC-cookie=your_cookie'}
response = requests.get(url, headers=headers, verify=False)
```

---

6. Cisco IOS XE and NX-OS Programmability

- **Cisco IOS XE**

  - **NETCONF**: Network Configuration Protocol.

```
    from ncclient import manager
    m = manager.connect(host='hostname', port=830, username='user',
password='pass', hostkey_verify=False)
```

- **RESTCONF**: RESTful API for network device configuration.

```
    url = 'https://hostname/restconf/data/ietf-interfaces:interfaces'
    headers = {'Authorization': 'Basic
base64encoded_username:password'}
    response = requests.get(url, headers=headers, verify=False)
```

- **Cisco NX-OS**

- **NX-API**: JSON-RPC API for NX-OS devices.

```
    url = 'https://hostname/ins'
    headers = {'Content-Type': 'application/json-rpc'}
    payload = {
        "jsonrpc": "2.0",
        "method": "cli",
        "params": {
            "cmd": "show version",
            "version": 1.2
        },
        "id": 1
    }
    response = requests.post(url, headers=headers, json=payload,
verify=False)
```

---

7. Containerization and Orchestration

- **Docker**

- **Commands**:

  - **Run a Container**: `docker run -it ubuntu:latest /bin/bash`

  - **List Containers**: `docker ps`

  - **Build an Image**: `docker build -t myimage:1.0 .`

- **Kubernetes**

- **Commands**:
  - **Deploy a Pod**: `kubectl run mypod --image=nginx`
  - **List Pods**: `kubectl get pods`
  - **Expose a Service**: `kubectl expose pod mypod --port=80 --type=LoadBalancer`
- **Examples**
  - **Dockerfile**:

```
FROM ubuntu:latest
RUN apt-get update && apt-get install -y nginx
CMD ["nginx", "-g", "daemon off;"]
```

  - **Kubernetes YAML**:

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mycontainer
    image: nginx
```

---

8. Version Control with Git

- **Basic Commands**
  - **Initialize a Repo**: `git init`
  - **Clone a Repo**: `git clone https://github.com/user/repo.git`
  - **Add Files**: `git add .`
  - **Commit Changes**: `git commit -m "Commit message"`
  - **Push Changes**: `git push origin main`
- **Branching and Merging**
  - **Create a Branch**: `git branch new-feature`

- **Switch to Branch**: `git checkout new-feature`
- **Merge Branches**: `git merge new-feature`

- **Examples**
- **Git Workflow**:

```
git init
git add .
git commit -m "Initial commit"
git branch new-feature
git checkout new-feature
git add .
git commit -m "Added new feature"
git checkout main
git merge new-feature
git push origin main
```

---

9. Continuous Integration/Continuous Deployment (CI/CD)

- **CI/CD Tools**
- **Jenkins**: Open-source automation server.
- **GitLab CI**: Integrated CI/CD pipeline.
- **GitHub Actions**: CI/CD directly from GitHub.

- **Pipeline Stages**
- **Build**: Compile code.
- **Test**: Run unit and integration tests.
- **Deploy**: Push to production.

- **Examples**
- **Jenkinsfile**:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
```

```
                sh 'make build'
            }
        }
        stage('Test') {
            steps {
                sh 'make test'
            }
        }
        stage('Deploy') {
            steps {
                sh 'make deploy'
            }
        }
    }
}
```

---

10. Security Best Practices

- **Authentication**
  - **Multi-Factor Authentication (MFA)**: Additional layer of security.
  - **API Tokens**: Use tokens instead of passwords.
- **Authorization**
  - **Role-Based Access Control (RBAC)**: Assign roles to users.
  - **Least Privilege**: Grant minimal access necessary.
- **Encryption**
  - **HTTPS**: Use SSL/TLS for secure communication.
  - **SSH**: Secure shell for remote access.
- **Examples**
  - **Secure API Call**:

```
url = 'https://api.example.com/data'
headers = {'Authorization': 'Bearer your_token'}
response = requests.get(url, headers=headers, verify=True)
```

---

11. Troubleshooting and Debugging

- **Common Issues**
  - **API Errors**: Check status codes and response bodies.
  - **Network Connectivity**: Use `ping` and `traceroute`.
  - **Python Errors**: Use `try-except` blocks.
- **Debugging Tools**
  - **Postman**: Test APIs.
  - **Wireshark**: Network packet analyzer.
  - **Python Debugger**: `import pdb; pdb.set_trace()`
- **Examples**
  - **Error Handling**:

```
try:
    response = requests.get('https://api.example.com/data')
    response.raise_for_status()
except requests.exceptions.HTTPError as err:
    print(f"HTTP error occurred: {err}")
```

---

12. Resources and Further Learning

- **Official Documentation**

- [Cisco DevNet Documentation](https://developer.cisco.com/docs/)
- [Python Documentation](https://docs.python.org/3/)

- **Community and Forums**

- [Cisco DevNet Community](https://developer.cisco.com/community/)
- [Stack Overflow](https://stackoverflow.com/)

- **Books**

- "Network Programmability and Automation" by Jason Edelman, Scott S. Lowe, and Matt Oswalt.
- "Python for Network Engineers" by Mircea Ulinic.

This cheat sheet provides a comprehensive overview of the essential topics covered in the Cisco DevNet Associate certification. Use it as a quick reference guide while preparing for the exam or during your daily work in network automation.

By Ahmed Baheeg Khorshid

ver 1.0