Module 11 CCNA -Automation and Programmability

**1.Explain How Automation Impacts**

**Network Management**

Automation significantly impacts network management by transforming how networks are monitored, configured, maintained, and secured. Here's a breakdown of its effects:

1. Increased Efficiency and Speed

* Automated Configuration: Automation tools can configure network devices in seconds, replacing manual processes that may take hours or days.
* RapidProvisioning: New devices or services can be deployed quickly, improving time-to-service.

2. Reduced Human Error

* Manual configurations are prone to mistakes, which can lead to outages or vulnerabilities.
* Automation ensures consistent implementation of policies and settings, reducing misconfigurations.

3. Improved Scalability

* Automation allows network teams to manage large-scale and complex networks with minimal manual intervention.
* Scaling up (or down) infrastructure becomes easier as repetitive tasks are handled by scripts or platforms.

4. Enhanced Monitoring and Troubleshooting

* Automated tools continuously monitor network health, detect anomalies, and alert teams in real time.
* Root cause analysis and remediation can be automated, reducing downtime and improving reliability.

5. Policy Enforcement and Compliance

* Network policies (e.g., security, QoS) can be automatically enforced across devices and locations.
* Automation helps maintain compliance with industry regulations by ensuring consistent application of rules.

6. Cost Reduction

* Reduces the need for large manual labor forces to manage the network.
* Minimizes costly outages and improves overall operational efficiency.

7. Support for Dynamic Environments

* With the rise of cloud and software**-**definednetworking **(**SDN**)**, networks are more dynamic.
* Automation adapts network configurations and routing to changing demands in real-time.

8. Integration with AI and Machine Learning

* AI-driven automation can predict network failures, optimize traffic flows, and suggest improvements.
* Self-healing networks become possible, where the system detects and corrects issues autonomously.

**2. Compare Traditional network with Controller based networking**

|  |  |  |
| --- | --- | --- |
| Aspect | Traditional Network | Controller-Based Network (SDN) |
|  |  |  |
| Architecture | Decentralized | Centralized |
| Configuration | Manual | Automated |
| Scalability | Limited | Highly scalable |
| Policy Enforcement | Manual | Centralized and consistent |
| Agility | Low | High |
| Security | Local and isolated | Centralized and holistic |
| Use Cases | Simple networks | Complex, dynamic, virtualized networks |

**3.** **Explain Virtualization**

Virtualization is the process of creating a virtual **(**ratherthanphysical**)** version of something—such as an operating system, server, storage device, or network resource.

It allows multiple simulated environments or virtualmachines **(**VMs**)** to run on a singlephysicalsystem, making better use of hardware resources.

**Types of Virtualization**

1. Server Virtualization
   * Divides a single physical server into multiple virtual servers.
   * Each VM runs its own operating system and applications.
   * Example**:** Using VMware or Hyper-V to host multiple VMs on one server.
2. Desktop Virtualization
   * Hosts desktop environments on a central server.
   * Users access their desktops remotely.
   * Example**:** Virtual Desktop Infrastructure (VDI).
3. Storage Virtualization
   * Combines multiple physical storage devices into a single virtual storage pool.
   * Easier to manage and allocate space dynamically.
4. Network Virtualization
   * Creates a virtual version of network resources like routers, switches, and firewalls.
   * Allows software-defined networking (SDN).
5. Application Virtualization
   * Runs applications in isolated environments.
   * Apps don’t need to be installed directly on the user’s device.
   * Example**:** Citrix or Microsoft App-V.

**4. Describe Characteristics of REST-based API**

REST (Representational State Transfer) is an architectural style used for designing networkedapplications, particularly APIs that communicate over HTTP. A RESTful API adheres to certain principles and characteristics:

1. Statelessness

* Eachrequest from a client to the server must contain all the information needed to understand and process it.
* The server does not store any client context between requests.
* This simplifies scalability and reliability.

2. Client-Server Architecture

* REST separates the client (frontend) from the server (backend).
* Clients and servers can evolve independently as long as the interface between them doesn’t change.

3. Use of HTTP Methods

RESTful APIs use standard HTTPmethods to perform operations:

|  |  |
| --- | --- |
| **HTTP Method** | **Operation** |
| GET | Retrieve data |
| POST | Create new resource |
| PUT | Update an existing resource |
| DELETE | Remove a resource |
| PATCH | Partially update a resource |

4. Resource-Based

* REST treats everything as a resource, identified by a URI **(**UniformResourceIdentifier**)**.
* Example:
  + GET /users → retrieves all users
  + GET /users/5 → retrieves user with ID 5

5. Uniform Interface

A consistent and standardized way of interacting with the API, including:

* Resource identification via URIs
* Standard methods (GET, POST, etc.)
* Self-descriptive messages
* Hypermedia links (HATEOAS – optional in some REST APIs)

6. Representation of Resources

* Resources can be represented in multiple formats such as JSON, XML, or HTML.
* JSON is the most commonly used format in modern RESTful APIs.

**7.** Cache ability

* Responses from the server can be explicitlymarkedascacheable or non-cacheable.
* Improves performance by reducing unnecessary requests to the server.

8. Layered System

* The architecture can be composed of multiplelayers, such as:
  + Load balancers
  + Authentication servers
  + Caching layers
* The client doesn’t need to know if it's communicating with the actual server or a proxy/intermediate

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**5.Explain methods of Automation**

1. Scripting

Writing custom scripts using programming languages like Python**,** Bash**,** orPowerShell to automate tasks.

* **Use Cases:**
* Automating backups
* Monitoring log files
* Configuring network devices
* **Pros:**
* Flexible and customizable
* Easy to start with
* **Cons**:
* Can become complex to manage at scale
* Requires programming knowledge

2. Configuration Management Tools

Use of specialized tools to automate configuration and management of infrastructure.

* Common Tools:

1. Ansible
2. Puppet
3. Chef
4. SaltStack

* UseCases**:**
* Deploying and maintaining server configurations
* Ensuring consistent network device setups
* Infrastructure as Code (IaC)
* Pros**:**
* Scalable and repeatable
* Supports version control
* Cons**:**
* Learning curve for tools and syntax
* Requires initial setup

3. Orchestration Platforms

Coordinate and automate complex workflows across multiple systems and tools.

* Common Platforms:
* Kubernetes (for container orchestration)
* Apache Airflow
* AWS Step Functions
* UseCases**:**
* Automating deployment pipelines
* Managing container lifecycles
* Multi-step process automation
* **Pros:**
* Manages dependencies and sequencing
* Ideal for large-scale, distributed systems
* **Cons:**
* More complex to configure
* May require infrastructure changes

4. GUI-Based Automation Tools

Graphical tools or dashboards that allow automation through point**-**and**-**clickinterfaces.

**Common Examples:**

* Cisco DNA Centre
* Microsoft System Centre
* SolarWinds Network Automation Manager

**Use Cases:**

* Network monitoring and automated alerts
* Scheduling maintenance tasks
* Automating device discovery and updates

**Pros:**

* User-friendly
* Requires minimal coding knowledge

**Cons:**

* Less flexibility compared to scripting
* May be limited to specific vendors

5. API-Based Automation

Using ApplicationProgrammingInterfaces **(**APIs**)** to programmatically interact with systems and devices.

**Use Cases:**

* Automating cloud infrastructure (e.g., AWS, Azure)
* Integrating different systems (monitoring, security, etc.)
* Real-time device configuration

**Pros:**

* Highly scalable and efficient
* Enables integration across diverse platforms

**Cons:**

* Requires understanding of API endpoints and authentication
* Potential security concerns if not managed properly

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**6.Explain SDN**

Software-Defined Networking (SDN**)** is an approach to networking that separates the controlplane (decision-making) from the dataplane (traffic forwarding). This allows network administrators to manage and configure networks centrally and programmatically rather than manually configuring individual devices.

 **SDN Controller** *(Brain of the network)*

* Centralized software that manages and makes decisions about traffic flow.
* Examples: OpenDaylight, ONOS, Cisco APIC.

 **Southbound APIs**

* Connect the controller to network devices (e.g., switches and routers).
* Most common: OpenFlow.

 **Northbound APIs**

* Allow applications and services to communicate with the SDN controller.
* Enable features like automation, analytics, and orchestration.

 **Networking Devices**

* Simple forwarding devices (data plane) that follow the controller’s instructions.

**7.Explain DNA Centre**

CiscoDNACentre (Digital Network Architecture Centre) is a centralized network management and automation platform developed by Cisco. It is designed to help organizations design, provision, monitor, and manage their network infrastructure more efficiently using automation, analytics, and AI/ML capabilities.

 **Design**

* Create network design templates and define network hierarchy (sites, buildings, floors).
* Define IP address pools, device roles, and wireless SSIDs.

 **Policy**

* Apply intent-based policies to users, devices, or applications.
* Enforce access control using Cisco ISE integration.

 **Provision**

* Automatically discover, configure, and deploy devices to the network.
* Supports zero-touch provisioning **(**ZTP).

 **Assurance**

* Provides real-time visibility, telemetry, and health scores for users, devices, and apps.
* Offers AI-driven troubleshooting and root cause analysis.

 **Platform**

* Offers open APIs to integrate with third-party applications and services.
* Enables developers to build custom apps using the Cisco DNA Centre SDK

**8.Explain SD-Access and SD-WAN**

**SD-Access**

SD**-**Access and SD**-**WAN are two key technologies under Cisco’s Software-Defined Networking (SDN) umbrella. Both aim to simplify and secure network management, but they focus on different environments and use cases.

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Fabric-based Networking** | Uses a virtualized network "fabric" for simplified design and management. |
| **Identity-based Access** | Users and devices are authenticated and authorized based on identity, not IP address. |
| **End-to-End Segmentation** | Enables macro and micro**-**segmentation for security and traffic isolation. |
| **Automation via Cisco DNA Centre** | Centralized configuration, provisioning, and assurance. |
| **Policy Consistency** | Policies follow users and devices as they move across the network. |

**SD-WAN:**

SD-WAN is a software-defined approach to managing and optimizing wide area networks (WANs) that connect remote branches, data centres, and cloud resources over multiple types of links (e.g., MPLS, broadband, LTE).

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Transport Independence** | Supports any type of WAN connection (MPLS, Internet, LTE). |
| **Application-Aware Routing** | Routes traffic based on app type, performance, and policies. |
| **Centralized Management** | Configured and managed via a central SD-WAN controller |
| **Security Integration** | Built-in firewall, IPS/IDS, and encryption. |
| **Cloud Optimization** | Direct access to SaaS/cloud apps (e.g., Office 365, AWS) without backhauling. |