DUMPSBOSS.

DevNet Associate (DEVASC)

Cisco 200-901

Version Demo

Total Demo Questions: 52

Total Premium Questions: 524

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Software Development and Design	97
Topic 2, Understanding and Using APIs	168
Topic 3, Cisco Platforms and Development	48
Topic 4, Application Deployment and Security	66
Topic 5, Infrastructure and Automation	74
Topic 6, Network Fundamentals	71
Total	524

QUESTION NO: 1

Refer to the exhibit.

```
module ex-ethernet {
     namespace "http://example.com/ethernet";
     prefix "eth";
     import ietf-interfaces {
       prefix if;
     }
     augment "/if:interfaces/if:interface" {
       when "if:type = 'ethernetCsmacd'";
       container ethernet {
         must "../if:location" {
           description
             "An ethernet interface must specify the physical location of the ethernet hardware.";
         }
         choice transmission-params {
           case auto {
             leaf auto-negotiate {
               type empty;
             }
           }
           case manual {
             leaf duplex {
               type enumeration {
                 enum "half";
                 enum "full";
               }
             }
             leaf speed {
               type enumeration {
                 enum "10Mb";
                 enum "100Mb";
                 enum "1Gb";
                 enum "10Gb";
               }
             }
           }
         } // other ethernet specific params...
       }
     }
```

What is represented in this YANG module?

A. interface management

B. BGP

C. OpenFlow

D. topology

ANSWER: A

Explanation:

The YANG module shown represents interface management/operational interface data. In Cisco IOS XE, interface-related YANG models (such as Cisco-IOS-XE-interfaces-oper.yang) define the schema used by NETCONF/RESTCONF to retrieve and manipulate interface state and configuration in a structured, programmatic way. These models typically include containers and lists for interfaces and their attributes (for example, interface names, administrative/operational status, counters, and other interface-specific operational fields). This aligns with the purpose of YANG as a data modeling language for network configuration and state data, where “interfaces” are a core managed object across devices. In practice, an application would use the paths defined by this module to query interface operational data (like up/down state and statistics) or to correlate interface identifiers with other telemetry and automation workflows. Cisco publishes these IOS XE native YANG modules to describe device capabilities and expose them consistently via model-driven programmability interfaces.

References: CiscoDevNet example: Cisco-IOS-XE-interfaces-oper.yang, RFC 7950: The YANG 1.1 Data Modeling Language

QUESTION NO: 2

Exhibit.

```
HTTPS/1.1 201 Created
 Date: Tue, 08 Nov 2020 09:37:27 GMT
 Server: example-server
 Location: https://example.com/restconf/data\
       example-jukebox/library/dogbreed=Australian%20Cattle%20Dog
 Last Modified: Tue 08 Nov 2020 09:37:27 GMT
 ETag: "s1878124t4c"
```

Refer to the exhibit. Which RESTCONF request results in this response?

A)

```
GET /restconf/data/example-jukebox:jukebox/library HTTP/1.1
     Host: example.com
     Content-Type: application/yang-data-json

     {
      "Example-jukebox:dogbreed" : [
        {
          "name" : "Australian Cattle Dog
        }
      ]
   }
```

B)

```
CONFIG /restconf/data/example-jukebox:jukebox/library HTTP/1.1
     Host: example.com
     Content-Type: application/yang-data-json

     {
      "Example-jukebox:dogbreed" : [
        {
          "name" : "Australian Cattle Dog
        }
      ]
   }
```

C)

```
PUT /restconf/data/example-jukebox:jukebox/library HTTP/1.1
     Host: example.com
     Content-Type: application/yang-data-json

     {
      "Example-jukebox:dogbreed" : [
        {
          "name" : "Australian Cattle Dog
        }
      ]
   }
```

D)

```
POST /restconf/data/example-jukebox:jukebox/library HTTP/1.1
    Host: example.com
    Content-Type: application/yang-data+json

    {
      "Example:dogbreed" : [
        {
          "name" : "Australian Cattle Dog
        }
      ]
    }
```

A. Option A

B. Option B

C. Option C

D. Option D

ANSWER: B

Explanation:

The response shown in the exhibit indicates that a new RESTCONF resource was successfully created, which is typically represented by HTTP status code 201 (Created) along with a server-assigned resource location. In RESTCONF (per RFC 8040), creating a new list entry is most commonly done with an HTTP POST to the parent list resource, allowing the server to create the new entry (and often to determine the final resource URI). While PUT can create a resource, it is generally used when the client is targeting a specific, fully-known resource URI (including the key) and is replacing/creating that exact resource. The request that results in a “created” response for adding a new list item under a collection is therefore the POST-style request shown among the choices. This aligns with RESTCONF’s use of standard HTTP semantics for datastore resources and is consistent with Cisco RESTCONF examples for creating new configuration objects under a container/list.

References: RFC 8040 - RESTCONF Protocol, Cisco IOS XE RESTCONF Documentation

QUESTION NO: 3

Which HTTP code group is issued when a request is received successfully, understood, and processed?

A. 2xx

B. 3xx

C. 4xx

D. 5xx

ANSWER: A

Explanation:

The correct HTTP status code group for a request that is received successfully, understood, and processed is the 2xx class. In HTTP semantics, status codes are grouped by their first digit to indicate the broad outcome of the request. The 2xx group specifically represents “Successful” responses, meaning the server accepted the request, understood it, and completed the processing. Common examples include 200 (OK) for a successful retrieval, 201 (Created) when a new resource is created (often after POST), and 204 (No Content) when the request succeeds but there is no response body to return. This grouping is foundational for API consumers and automation because it allows clients to quickly interpret outcomes and implement logic such as retry behavior, error handling, or follow-up actions based on the response class. Cisco DevNet exam objectives commonly expect you to recognize these HTTP code families and their meanings when working with REST APIs.

References: MDN Web Docs: HTTP response status codes, RFC 9110: HTTP Semantics (Status Codes)

QUESTION NO: 4

Refer to the exhibit.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:Netconf:base:1.0">
  <get-conflg>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns=" urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name/>
        </interface>
      </interfaces>
    </filter>
  </get-config>
</rpc>
```

A network engineer uses model-driven programmability to monitor and perform changes on the network. The network engineer decides to use the NETCONF RPC message to complete one of their tasks. What is accomplished by sending the RPC message?

A. The running-config of the device is returned.

B. The name of each interface is reset to a default name.

C. All the YANG capabilities supported by the device are returned.

D. A list of interface names is returned.

ANSWER: A

Explanation:

Sending the NETCONF RPC message accomplishes retrieval of configuration data from the device. In NETCONF, operations are carried inside an element, and the operation is specifically used to read configuration datastores such as . When the RPC targets the running datastore, the device returns the current running configuration (or the relevant subset of it) in an . If the RPC includes a subtree filter (for example, filtering down to interface elements like interface names), the server still returns configuration data from the running configuration, but limited to what matches the filter. This is a core NETCONF use case: programmatically querying the device's intended configuration state in a structured, model-driven way rather than scraping CLI output. The key point is that is a read-only retrieval operation against a configuration datastore, not a change operation, and the result is the running configuration content constrained by any provided filter.

References: RFC 6241 - NETCONF Protocol, NETCONF <get-config> operation

QUESTION NO: 5

Which two statements are true about Cisco UCS Manager, Cisco UCS Director, or Cisco Intersight APIs? (Choose two.)

A. UCS Manager uses JSON to encode API interactions and utilizes Base64-encoded credentials in the HTTP header for authentication.

B. UCS Director API interactions can be XML- or JSON-encoded and require an API key in the HTTP header for authentication.

C. Cisco Intersight uses XML to encode API interactions and requires an API key pair for authentication.

D. UCS Manager API interactions are XML-encoded and require a cookie in the method for authentication.

E. Cisco Intersight API interactions can be encoded in XML or JSON and require an API key in the HTTP header for authentication.

ANSWER: C D

Explanation:

Cisco UCS Manager's native API is the XML API. Clients authenticate by first performing a login operation that returns an authentication cookie (often called a session cookie) which is then presented on subsequent API calls, so the statement describing XML-encoded interactions and cookie-based authentication aligns with how UCS Manager works.

Cisco Intersight's REST API is JSON-based and uses an API key pair model for authentication: you create an API key (key ID) and sign requests using the associated secret (HMAC signature) that is sent via HTTP headers. This matches the statement that Intersight requires an API key pair for authentication (even though Intersight does not use XML for payload encoding). In practice, Intersight requests are signed and include headers such as a key identifier and signature-related metadata.

References: Cisco UCS Manager XML API Programmer's Guide, Cisco Intersight API Docs – Security

QUESTION NO: 6

A developer is trying to retrieve data over a REST API. The API server responds with an HTTP client error response code. After investigating the response the developer realizes the realizes the response has a Retry-After header. What is the root cause of this error?

A. The REST service is unreachable at the time of the REST request

B. Too many requests were sent to the REST service in a given amount of time.

C. An appliance limited the rate of requests to the transport layer.

D. An appliance limited the rate of requests to the application layer

ANSWER: B

Explanation:

The presence of a `Retry-After` header alongside an HTTP client error response strongly indicates the server is rate limiting the client due to excessive request volume. In REST APIs, this is most commonly expressed as `429 Too Many Requests`, which is a client error (4xx) returned when the client has exceeded a rate limit policy enforced by the API. The `Retry-After` header is the server's way of telling the client when it is acceptable to try again (either after a number of seconds or at a specific HTTP-date), enabling well-behaved backoff and retry logic. This pattern is a standard mechanism for protecting API availability and fairness across consumers, and it is widely used by API gateways and services to prevent abuse and to manage load. In practice, the correct remediation is to reduce request frequency, implement exponential backoff, honor `Retry-After`, and/or request higher rate limits if appropriate. References: MDN: Retry-After header, MDN: 429 Too Many Requests.

QUESTION NO: 7

Which two HTTP code series relate to errors? (Choose two.

A. 400

B. 200

C. 500

D. 300

E. 100

ANSWER: A C

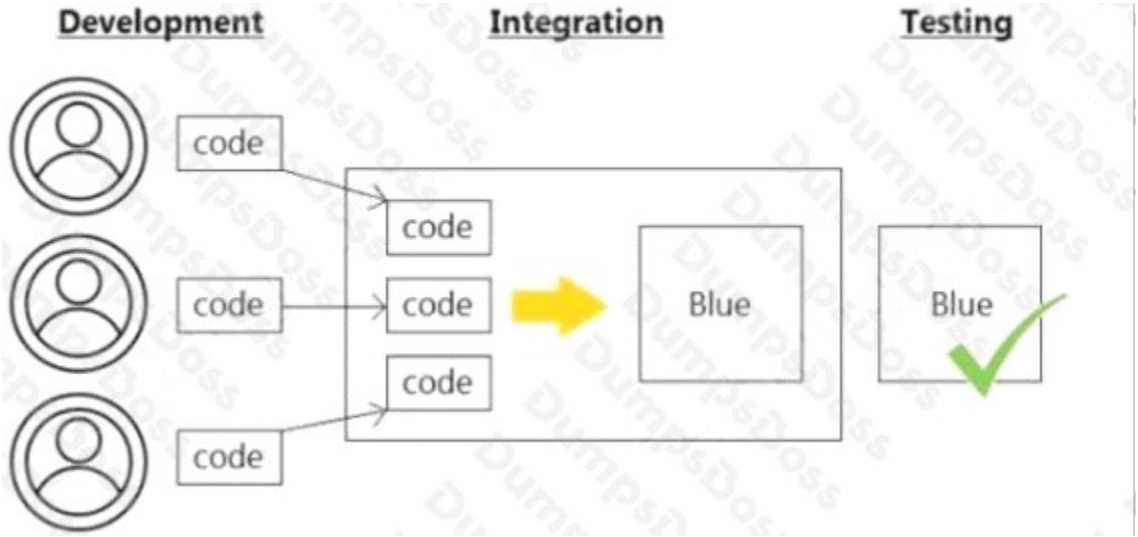
Explanation:

HTTP status codes are grouped into classes by their first digit, and the classes that represent errors are the client-error and server-error ranges. The **400** series indicates a client error, meaning the request is invalid in some way (for example, malformed syntax, missing authentication, or requesting a resource that does not exist). These responses tell the client that the problem is with the request and that changing the request is typically required to succeed. The **500** series indicates a server error, meaning the server received what appears to be a valid request but failed to fulfill it due to an issue on the server side (for example, an unhandled exception, misconfiguration, or upstream dependency failure). In API development and troubleshooting (including DevNet exam topics), recognizing these two series is essential for quickly determining whether to focus debugging on the client/request construction or on server-side logic and infrastructure. This classification is defined in the HTTP semantics specification and is consistently reflected in modern HTTP references.

References: RFC 9110 (HTTP Semantics), Status Codes, MDN Web Docs: HTTP response status codes

QUESTION NO: 8

Refer to the exhibit.



Which infrastructure automation method is shown?

A. Waterfall

B. CI/CD pipeline

C. Agile

D. Lean

ANSWER: B

Explanation:

The infrastructure automation method shown is a CI/CD pipeline. A CI/CD pipeline is an automated workflow that takes code changes from a source repository through repeatable stages such as build, test, and deploy. In the DevNet/DEVASC context, this is a core automation approach because it replaces manual, error-prone release steps with consistent automation, enabling frequent, reliable delivery of changes to infrastructure and applications. Continuous Integration focuses

on automatically building and validating each change (often via unit/integration tests) as it is merged, while Continuous Delivery/Deployment extends that automation to packaging and releasing changes to staging or production environments. This pipeline-based approach is commonly represented visually as sequential stages with automated gates, which matches typical “pipeline” exhibits used in Cisco exam materials. Cisco emphasizes CI/CD as a practical method to operationalize Infrastructure as Code and improve speed and quality of deployments across network and cloud environments.

References: Cisco DEVASC (200-901) exam topics, Red Hat: What is CI/CD?

QUESTION NO: 9

What are two benefits of model-driven programmability?

A. model-driven APIs for abstraction and simplification

B. single choice of transport, protocol, and encoding

C. model-based, structured, and human friendly

D. easier to design, deploy, and manage APIs

E. models decoupled from transport, protocol, and encoding

ANSWER: A C E

Explanation:

Model-driven programmability is built around using a formal data model (commonly YANG) as the source of truth for how configuration and operational state are represented. One key benefit is “model-driven APIs for abstraction and simplification” because the model provides a consistent, structured schema that tools and controllers can rely on, reducing ambiguity and enabling automation to validate data types, constraints, and relationships. Another major benefit is “models decoupled from transport, protocol, and encoding” because the same underlying model can be exposed via different protocols and encodings (for example, NETCONF with XML or RESTCONF with JSON) without redefining the data structure each time. This separation improves portability, interoperability, and long-term maintainability as transports evolve while the model remains stable. Together, these benefits make network automation more predictable and scalable by standardizing how devices describe and accept data, and by allowing clients to choose the most appropriate protocol/encoding while still interacting with the same modeled resources. For more details, see Cisco’s NETCONF/RESTCONF and YANG overviews: Cisco IOS XE RESTCONF and RFC 7950 (YANG 1.1).

QUESTION NO: 10

What are two benefits of model-driven programmability? (Choose two.)

A. easier to design, deploy, and manage APIs

B. single choice of transport, protocol, and encoding

C. models decoupled from transport, protocol, and encoding

D. model-based, structured, and human friendly

E. model-driven APIs for abstraction and simplification

ANSWER: C E

Explanation:

Model-driven programmability is built around a formal data model (commonly YANG) that defines the structure, constraints, and semantics of configuration and operational state. A key benefit is that the same model can be used across different management interfaces, which is why “models decoupled from transport, protocol, and encoding” is correct: the model is

independent of whether you access it via NETCONF, RESTCONF, gNMI, etc., and independent of encodings like XML or JSON. This separation enables consistent automation and tooling regardless of the underlying transport.

Another major benefit is that the model provides a higher-level, consistent abstraction of device capabilities, making automation simpler and less error-prone. That's captured by "model-driven APIs for abstraction and simplification

", because consumers can rely on a predictable schema, validation rules, and well-defined paths/objects rather than screen-scraping CLI output or dealing with vendor-unique, unstructured payloads. In practice, this improves interoperability, supports programmatic validation, and enables better lifecycle management of APIs and automation workflows.

References: https://datatracker.ietf.org/doc/html/rfc7950 and https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1612/b_1612_programmability_cg/model_driven_programmability.html

QUESTION NO: 11

In test-driven development, what are two of the green bar patterns? (Choose two.)

A. another test

B. break

C. triangulate

D. starter test

E. fake it

ANSWER: C E

Explanation:

In test-driven development (TDD), "green bar patterns" are techniques used during the "make it pass" step—after writing a failing test—to get the tests to pass with the simplest possible change, before later refactoring. Two commonly cited green bar patterns are "fake it" and "triangulate." "fake it" means you implement the smallest, often hard-coded behavior that satisfies the current test (for example, returning a constant) to quickly reach a passing state; you then generalize the implementation as additional tests are added. "triangulate" is the technique of adding a second (or third) test case to force the code to become more general, preventing an overfitted implementation that only satisfies one example. Together, these patterns help keep changes small, feedback fast, and design emerging from tests rather than being overdesigned up front. These terms are widely discussed in TDD literature and summaries of TDD patterns, including the classic "Test Driven Development: By Example" approach and related pattern catalogs.

References: https://martinfowler.com/bliki/TestDrivenDevelopment.html, https://en.wikipedia.org/wiki/Test-driven_development

QUESTION NO: 12

Exhibit.

```
{
    "Cisco-IOS-XE-native:GigabitEthernet": {
        "name": "2",
        "vrf": {
            "forwarding": "MANAGEMENT"
        },
        "ip": {
            "address": {
                "primary": {
                    "address": "192.168.1.10",
                    "mask": "255.255.0.0"
                }
            }
        },
        "mop": {
            "enabled": true
        },
        "Cisco-IOS-XE-ethernet:negotiation": {
            "auto": true
        }
    }
}
```

Refer to the exhibit. A network engineer makes a RESTCONF API call to Cisco IOS XE to retrieve information. Which HTTP method and URL does the engineer use to make the ..call?

A)

```
GET https://devnetlab.local/restconf/api/running/native/GigabitEthernet
```

B)

```
GET https://devnetlab.local/restconf/api/running/native/interface/GigabitEthernet
```

C)

```
GET https://devnetlab.local/restconf/api/running/native/interface/GigabitEthernet/2/
```

D)

```
GET https://devnetlab.local/restconf/api/running/ios/native/interface/GigabitEthernet/1/
```

A. Option A

B. Option B

C. Option C

D. Option D

ANSWER: C

Explanation:

To retrieve operational or configuration data from a RESTCONF-enabled Cisco IOS XE device, the client uses the HTTP GET method. RESTCONF maps HTTP methods to datastore operations, and GET is specifically used for reading a resource (for example, an interface subtree) without modifying it. The URL must point to the RESTCONF resource path that represents the YANG data model node being queried. On IOS XE, RESTCONF requests are made under the RESTCONF root (commonly `/restconf/data` for NMDA "data" resources, or `/restconf/api` on some IOS XE/DevNet lab variants), followed by the module/container hierarchy such as `native/interface/GigabitEthernet` and then the specific list key (the interface name/number) to retrieve a single interface instance. Therefore, the correct call is the one that uses GET and targets the GigabitEthernet interface resource in the running configuration via the RESTCONF URL structure shown in the exhibit. This aligns with Cisco's RESTCONF usage where GET reads YANG-modeled resources and the path encodes the container/list keys for the desired interface.

References: RFC 8040 - RESTCONF Protocol, Cisco IOS XE RESTCONF (DevNet)

QUESTION NO: 13

What are two considerations when selecting the "best route" for a network device to reach its destination? (Choose two.)

A. MAC address

B. IP address

C. metrics

D. administrative distance

E. subnet mask

ANSWER: C D

Explanation:

When a router selects the "best route" to a destination, it primarily evaluates information in its routing table using a predictable decision process. One key consideration is administrative distance, which is the trustworthiness ranking of the route source (for example, a route learned from a dynamic routing protocol versus a static route). If multiple routes to the same destination prefix exist from different sources, the router prefers the one with the lowest administrative distance.

Another key consideration is metrics. When there are multiple candidate routes to the same destination learned from the same routing source, the router uses that protocol's metric (such as cost, bandwidth-based values, or hop count depending on the protocol) to choose the most preferred path. In practice, this means administrative distance helps pick the best route between different routing sources, and metrics help pick the best route within the same source. This route selection behavior is foundational to how Cisco routers build and maintain the routing table and forward packets efficiently.

References: Cisco - Administrative Distance, Cisco - OSPF Metric/Cost Overview

QUESTION NO: 14

Which mechanism is used to consume a RESTful API design when large amounts of data are returned?

A. data sets

B. scrolling

C. pagination

D. blobs

ANSWER: C

Explanation:

Pagination is the standard mechanism used in RESTful API designs to handle large result sets by splitting responses into smaller, manageable “pages” of data. Instead of returning an entire collection in a single response (which can be slow, memory-intensive, and prone to timeouts), the API returns a subset of items along with metadata or links that allow the client to request the next (or previous) page. Common patterns include query parameters such as `page` and `limit`/`per_page`, or cursor-based pagination using an opaque token like `cursor` or `nextToken` for more stable traversal when the underlying dataset changes. This approach improves performance, reduces bandwidth usage, and provides a better consumer experience by enabling incremental loading and predictable response sizes. Many API specifications and best-practice guides also recommend including navigation links (for example, in response headers or a `links` object) to support discoverability and consistent client behavior. For practical guidance on designing and consuming paginated REST APIs, see Nordic APIs: API Pagination and the OpenAPI Specification links documentation.

QUESTION NO: 15

Refer to the exhibit.

```
---
- hosts: switch2960cx
  gather_facts: no

  tasks:
    - ios_l2_interface:
        name: GigabitEthernet0/1
        state: unconfigured

    - ios_l2_interface:
        name: GigabitEthernet0/1
        mode: trunk
        native_vlan: 1
        trunk_allowed_vlans: 6-8
        state: present

    - ios_vlan:
        vlan_id: 6
        name: guest-vlan
        interfaces:
          - GigabitEthernet0/2
          - GigabitEthernet0/3

    - ios_vlan:
        vlan_id: 7
        name: corporate-vlan
        interfaces:
          - GigabitEthernet0/4

    - ios_vlan:
        vlan_id: 8
```

```
- ios_vlan:
    vlan_id: 8
    name: iot-vlan
    interfaces:
      - GigabitEthernet0/5
```

Which two statements describe the configuration of the switch after the Ansible script is run? (Choose two.)

A. Traffic from ports 0/2 to 0/5 is able to flow on port 0/1 due to the trunk.

B. GigabitEthernet0/1 is left unconfigured.

C. GigabitEthernet0/2 and GigabitEthernet0/3 are access ports for VLAN 6.

D. Traffic is able to flow between ports 0/2 to 0/5 due to the trunk on port 0/1.

E. Traffic on ports 0/2 and 0/3 is connected to port 0/6.

ANSWER: A C

Explanation:

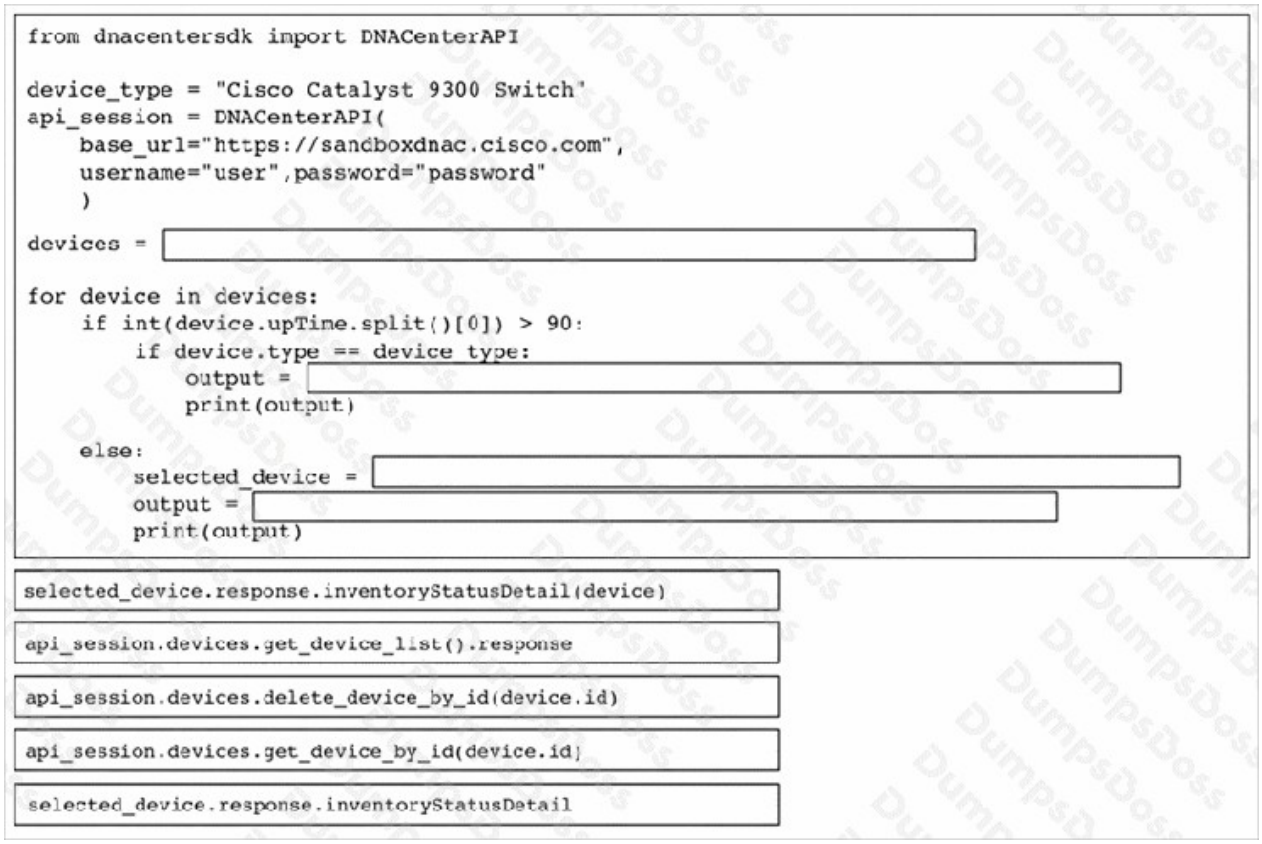
The resulting switch configuration reflects a common Ansible-driven Layer 2 setup where specific interfaces are placed into an access VLAN and an uplink is configured as an 802.1Q trunk. When GigabitEthernet0/2 and GigabitEthernet0/3 are configured as access ports in VLAN 6, any end hosts connected to those ports are placed into the same Layer 2 broadcast domain, meaning they can communicate at Layer 2 (subject to spanning-tree and security features) without requiring routing. In addition, when GigabitEthernet0/1 is configured as a trunk, it can carry VLAN 6 (and potentially other VLANs, depending on the allowed VLAN list) toward another switch or upstream device. That trunk capability is what enables traffic from the access ports (including ports 0/2 through 0/5 if they are also assigned to VLAN 6 by the playbook) to traverse the uplink on port 0/1 while preserving VLAN tagging. This is the standard behavior of Cisco IOS switching: access ports send/receive untagged frames mapped to a single VLAN, while trunk ports carry tagged frames for one or more VLANs. For details, see Cisco's VLAN/trunking configuration guidance and Ansible's network automation approach: Cisco VLAN/trunking overview and Ansible network automation for Cisco IOS.

QUESTION NO: 16 - (DRAG DROP)

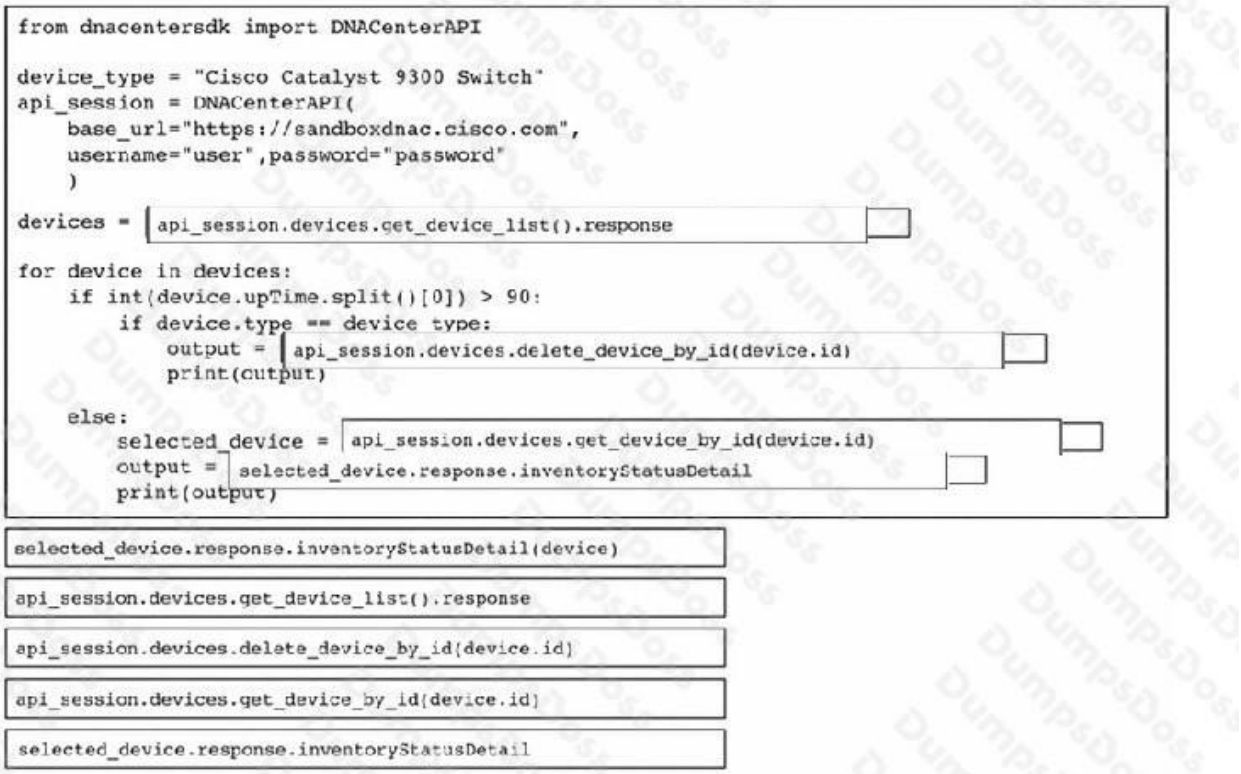
Refer to the exhibit.

A Python script must delete all Cisco Catayst 9300 Series switches that have an uptime that is greater than 90 days The script must also query for the status of all the other devices Drag and drop the code

from the bottom onto the bottom box the code is missing to complete the script Not at options are used



ANSWER:



Explanation:

The script is using the Cisco DNA Center Python SDK, so the missing lines must call the SDK's `devices` methods exactly as they're defined. First, the code needs a list of devices to iterate over, which comes from `api_session.devices.get_device_list()`. In this SDK, the returned object exposes the actual payload under `.response`, so the devices list is populated with `api_session.devices.get_device_list().response`.

Next, the requirement says to delete only Cisco Catalyst 9300 Series switches whose uptime is greater than 90 days. The loop already checks uptime and then checks `device.type` against the `device_type` string. Once both conditions are met, the correct action is to delete that device by its identifier. The Devices API method for that is `delete_device_by_id(id)`, and the device object provides the identifier as `device.id`. The return value can be stored in `output` and printed, matching the existing `print(output)` line.

For all other devices (the `else` branch), the script must query status. The SDK provides `get_device_by_id(id)` to retrieve a specific device record, so `selected_device` should be set to `api_session.devices.get_device_by_id(device.id)`. The exhibit also references "Device status: inventoryStatusDetail", which is exposed in the returned object's `.response` as `inventoryStatusDetail`. That means the status output line should read `selected_device.response.inventoryStatusDetail` (a field/property), not a function call. This satisfies the requirement to query and print the status for devices that are not being deleted.

References: Cisco DNA Center Platform API documentation and the SDK usage patterns in the dnacentersdk project: dnacentersdk on GitHub.

QUESTION NO: 17

```
def enable_function(if_name, if_status, if_type):
    headers = {'Accept': 'application/yang-data+json',
               'Content-Type': 'application/yang-data+json'}
    payload = {
              "ietf-interfaces:interface": {
                  "name": if_name,
                  "enabled": if_status,
                  "type": if_type,
                  }
              }
    base_url = 'https://192.168.1.1:8443'
    restconf_url = '/restconf/data/ietf-interfaces:interfaces/interface'

    res = requests.put(f'{base_url}{restconf}={if name}',
                               headers=headers, json=payload,
                                  auth=('cisco', 'secret'), verify=False)
```

Refer to the exhibit. A network engineer wants to automate the port enable/disable process on specific Cisco switches. The engineer creates a script to send a request through RESTCONF and uses ietf as the YANG model and JSON as payload. Which command enables an interface named Loopback1?

A. enable_function(Loopback1, true, 'iana-if-type:softwareLoopback')

B. enable_function('iana-if-type:softwareLoopback', Loopback1, true,)

C. def enable_function('iana-if-type:softwareLoopback', Loopback1, false,)

D. def enable_function(Loopback1, true, 'iana-if-type:softwareLoopback')

ANSWER: D

Explanation:

To enable an interface via RESTCONF using the IETF interfaces YANG model, the payload must set the interface's `enabled` leaf to `true` for the specific interface instance identified by its `name` and (commonly in examples) its `type`. In the IETF model (`ietf-interfaces`), a loopback interface is represented with the IANA interface type value `iana-if-type:softwareLoopback`, and the administrative state is controlled by `enabled`. Therefore, the correct command is the one that clearly targets the interface named `Loopback1` and sets the enable flag to `true` while using the appropriate interface type for a loopback. This aligns with how RESTCONF clients construct JSON bodies for `/restconf/data/ietf-interfaces:interfaces/interface=Loopback1` updates, where `"enabled": true` is the key element to administratively bring the interface up. See the IETF interfaces model definition and RESTCONF usage patterns in the RFCs: RFC 8343 (A YANG Data Model for Interface Management) and RFC 8040 (RESTCONF Protocol).

QUESTION NO: 18

Which application should be used to externally access all applications deployed in the same host machine with a single public IP address and port, when each application listens on different local ports?

A. reverse proxy

B. load balancer

C. DNS

D. firewall

ANSWER: A

Explanation:

The correct choice is

reverse proxy

because a reverse proxy is specifically designed to sit in front of one or more backend applications and present a single “public” endpoint (one IP and typically one port like 80/443) to external clients. It then forwards each incoming request to the appropriate internal application based on Layer 7 attributes such as the HTTP Host header (virtual hosts), URL path (for example, /app1 vs /app2), or other request properties. This is exactly the pattern used when multiple services run on the same machine but listen on different local ports (for example, 127.0.0.1:3000, :4000, :5000) and you want them all reachable via a single public address without exposing multiple public ports. In practice, common reverse proxies include NGINX, HAProxy (in HTTP mode), and Apache httpd, and they also often provide TLS termination, centralized authentication, and request/response controls. This aligns with standard reverse-proxy behavior described in vendor documentation such as NGINX’s reverse proxy guide and HAProxy’s HTTP reverse-proxy capabilities.

References: NGINX Reverse Proxy, HAProxy Reverse Proxy

QUESTION NO: 19

Which two items are Cisco DevNet resources? (Choose two.)

A. TAC Support

B. Software Research

C. API Documentation

D. Bitbucket

E. Sandbox

ANSWER: C E

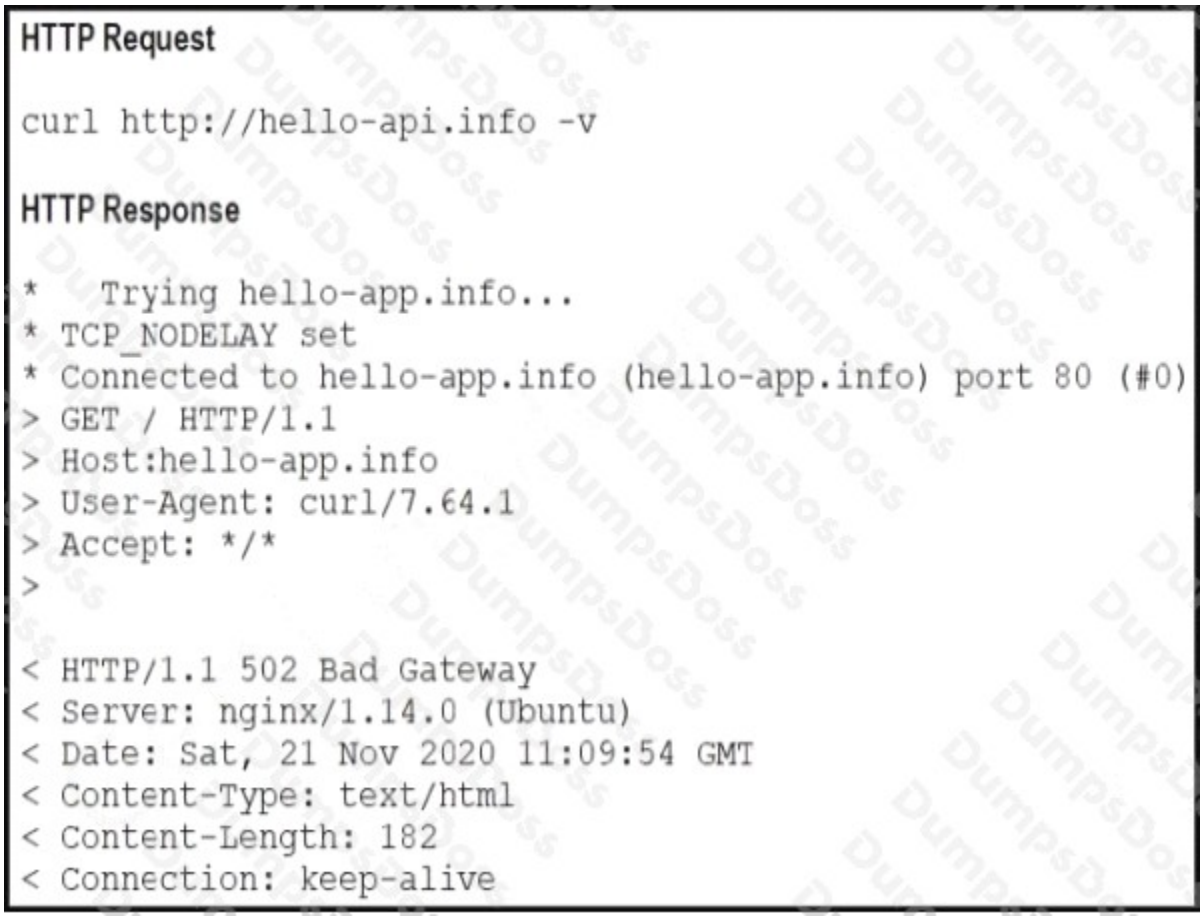
Explanation:

Cisco DevNet is Cisco’s official developer program and portal, and two of its core resources are API documentation and hands-on sandbox environments. API documentation is a primary DevNet offering because Cisco publishes developer guides, reference docs, and “try it” resources for Cisco platforms (for example, networking, security, collaboration, and cloud) so developers can understand endpoints, authentication methods, data models, and example requests/responses. DevNet also provides Sandboxes, which are reservable or always-on lab environments that let you experiment with real Cisco software and hardware (or simulations) without needing to build your own lab. These sandboxes commonly include preconfigured devices, credentials, and connectivity details so you can test automation scripts, REST APIs, and workflows in a safe environment. Both resources are explicitly featured and maintained on the DevNet portal and are central to how Cisco supports learning and application development around its products. See the DevNet home page and the DevNet Sandboxes landing page for official descriptions and access points.

References: https://developer.cisco.com/, https://developer.cisco.com/site/sandbox/

QUESTION NO: 20

Refer to the exhibit.



A developer cannot reach the web application behind an NGINX load balancer. The developer sends

a request to an application FQDN with cURL but gets an HTTP 502 response. Which action solves the problem?

A. Fix errors in the server configuration, which is behind the load balancer.

B. Bring up the load balancer to the active state.

C. Fix errors in the cURL request sent by the client.

D. Change the default gateway on the load balancer to an active one.

ANSWER: A

Explanation:

An HTTP 502 Bad Gateway from NGINX means NGINX (acting as a reverse proxy/load balancer) successfully received the client request, but it could not get a valid response from the configured upstream (the application server pool). In practice, this is most commonly caused by an upstream being unreachable (service down, wrong IP/port, firewall), refusing connections, timing out, or returning an invalid/empty response due to application/server-side misconfiguration. Because the error is generated at the proxy boundary, the corrective action is to fix the upstream server side (including its listener/port, health, and any configuration issues) so that NGINX can establish a successful connection and receive a proper HTTP response. This aligns with NGINX's documented meaning of 502 in reverse-proxy scenarios: the gateway/proxy received an invalid response from the upstream server, so remediation focuses on upstream availability/configuration and proxy-to-upstream connectivity. See NGINX's error code guidance and reverse proxy troubleshooting references for typical 502 causes and fixes: https://nginx.org/en/docs/http/ngx_http_proxy_module.html and https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/.

QUESTION NO: 21

What are the two principles of an infrastructure as code environment? (Choose two)

A. Complete complex systems must be able to be built from reusable infrastructure definitions.

B. Environments must be provisioned consistently using the same inputs.

C. Redeployments cause varying environment definitions.

D. Service overlap is encouraged to cater for unique environment needs.

E. Components are coupled, and definitions must be deployed for the environment to function.

ANSWER: A B

Explanation:

Infrastructure as Code (IaC) is centered on defining and managing infrastructure through code so that environments can be created reliably and repeatedly. A key principle is that complete complex systems must be able to be built from reusable infrastructure definitions. In practice, this means you design infrastructure modules (for example, network, compute, and security building blocks) that can be composed and reused across projects and stages (dev/test/prod) rather than rebuilt manually each time. Reusability improves speed, reduces human error, and encourages standardization.

Another core principle is that environments must be provisioned consistently using the same inputs. IaC aims for deterministic outcomes: given the same code and parameters, you should get the same infrastructure state every time. This consistency is what enables repeatable deployments, predictable troubleshooting, and safer automation in CI/CD pipelines. It also supports practices like immutable infrastructure and drift reduction because the desired state is expressed in code and applied uniformly.

These principles align with widely accepted IaC guidance from major tooling and cloud providers, emphasizing repeatability, consistency, and modular reuse as foundational behaviors of IaC-driven operations.

References: https://developer.hashicorp.com/terraform/intro/core-workflow, https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

QUESTION NO: 22

What are two functions of a routing table on a network device? (Choose two.)

A. It lists entries more than two hops away.

B. It lists the routes to a particular destination.

C. It lists the routes to drop traffic.

D. It lists hosts that are one hop away.

E. It lists the static and dynamic entries.

ANSWER: B C E

Explanation:

A routing table is the device's forwarding "map" that tells it how to reach destination networks. One core function is that it lists the routes to a particular destination, typically as destination prefixes (networks) along with the best next hop and/or outgoing interface to use. When a packet arrives, the device consults this table (via longest-prefix match) to decide where to send the packet next.

Another key function is that it lists the static and dynamic entries. Static routes are manually configured by an administrator, while dynamic routes are learned and maintained through routing protocols (for example, OSPF, EIGRP, or BGP). Both types can coexist in the routing table, and the device uses administrative distance and metrics to select the best route when multiple candidates exist.

These behaviors are fundamental to how Cisco routers and multilayer switches perform Layer 3 forwarding and are reflected in Cisco IOS route displays (for example, codes for connected, static, and protocol-learned routes). References: Cisco - Understanding and Interpreting the Routing Table, Cisco - OSPF and routing table concepts.

QUESTION NO: 23

Which Cisco product provides malware and content filtering through DNS?

A. Cisco ASA Firepower module

B. Cisco AMP

C. Cisco ISE

D. Cisco Umbrella

ANSWER: D

Explanation:

Cisco Umbrella provides malware protection and content filtering at the DNS layer by acting as a cloud-delivered secure DNS service. When a user or device attempts to resolve a domain name, Umbrella evaluates the DNS request against Cisco threat intelligence and policy controls. If the destination is known (or suspected) to be malicious—such as phishing, malware distribution, or command-and-control infrastructure—Umbrella can block the resolution so the connection never starts. This “DNS-layer security” approach is effective because it works early in the connection process and can protect users both on and off the corporate network without requiring full traffic inspection. In addition to security enforcement, Umbrella supports category-based web/content filtering (acceptable use controls) using DNS and related enforcement methods, allowing organizations to block access to non-business or inappropriate site categories according to policy. These capabilities align directly with the requirement for malware and content filtering through DNS.

References: Cisco Umbrella, Cisco Umbrella User Guide - About Umbrella

QUESTION NO: 24

Which two query types does a client make to a DNS server? (Choose two.)

A. Inverse

B. PTR

C. AAAA

D. ACK

E. DISCOVER

ANSWER: B C

Explanation:

DNS clients query DNS servers for specific resource record (RR) types in order to resolve names to addresses or addresses back to names. The query type

AAAA

is used when a client needs an IPv6 address for a hostname; the server answers with an IPv6 address mapping if one exists. The query type

PTR

is used for reverse lookups, where the client starts with an IP address and asks for the corresponding domain name (typically under the in-addr.arpa or ip6.arpa reverse-mapping domains). These two are common, standard DNS RR query types that clients routinely send as part of normal name resolution workflows, including dual-stack environments where a client may try AAAA first (or in parallel) and fall back to A records if needed, and troubleshooting or logging workflows that perform reverse DNS via PTR. This behavior is defined by DNS standards for RR types and reverse mapping. References: RFC 1035 (Domain Names - Implementation and Specification), RFC 3596 (DNS Extensions to Support IPv6).

QUESTION NO: 25

When using the Bash shell, how is the output of the devnet command saved to a file named “output.txt”?

A. devnet > output.txt

B. devnet | output.txt

C. devnet < output.txt

D. devnet & output.txt

ANSWER: A

Explanation:

In Bash, saving a command’s output to a file is done with output redirection using the `>` operator. Running `devnet > output.txt` redirects the command’s standard output stream (stdout) into the file named `output.txt`. If the file does not exist, Bash creates it; if it already exists, Bash overwrites it. This is the canonical way to capture the textual output of a command for later review, scripting, or sharing. If you wanted to append instead of overwrite, you would use `>>`, and if you also needed to capture error output (stderr), you could use `devnet > output.txt 2>&1` or the shorthand `&> output.txt` in Bash. The key concept tested here is that `>` is the standard output redirection operator in POSIX-like shells, including Bash, and it directly answers how to save the output of a command into a named file.

References: GNU Bash Manual — Redirections, bash(1) man page

QUESTION NO: 26

Refer to the exhibit.

```
import requests

url = "https://api.meraki.com/api/v0/organizations/{{organizationId}}/insight/
monitoredMediaServers"

payload = {
   "name": "Sample VoIP Provider",
   "address": "123.123.123.1"
}
headers = {
  'Accept': '*/*',
  'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.text.encode('utf8'))
```

Which workflow does the script automate?

A. retrieves a media server that is being monitored

B. updates a media server that is being monitored

C. adds a media server to be monitored

D. deletes a media server that is being monitored

ANSWER: C

Explanation:

The workflow being automated is adding a media server to be monitored. In REST APIs, creating a new resource is typically done with an HTTP POST to a collection endpoint (for example, posting to a “mediaServers” or similar resource path). A script that builds a JSON payload containing identifying attributes like a server *name* and *address* (IP/FQDN) and then sends that payload in a POST request is performing a create operation: it is registering a new media server object with the monitoring system so it can begin tracking it. This aligns with common API semantics where GET retrieves existing resources, PUT/PATCH updates an existing resource, and DELETE removes one, while POST is used to add a new item. Cisco Meraki’s API follows these standard REST patterns, using POST operations to create/configure new objects under a given network/organization scope. You can confirm the general REST method behavior and Meraki API conventions in the Meraki developer documentation and HTTP method references.

References: Cisco Meraki Dashboard API v1, MDN Web Docs: HTTP POST

QUESTION NO: 27

Which two statements describe the advantages of using a version control system? (Choose two.)

A. It allows for branching and merging so that different tasks are worked on in isolation before they are merged into a feature or master branch.

B. It provides tooling to automate application builds and infrastructure provisioning.

C. It allows multiple engineers to work against the same code and configuration files and manage differences and conflicts.

D. It provides a system to track User Stories and allocate to backlogs.

E. It allows developers to write effective unit tests.

ANSWER: A C

Explanation:

A version control system’s core advantages are enabling safe parallel development and maintaining a controlled history of changes. Branching and merging let teams isolate work (features, bug fixes, experiments) without destabilizing the main line, then integrate changes in a structured way once they’re reviewed and ready. This workflow supports common practices like feature branches, release branches, and hotfix branches, and it makes it easier to manage multiple versions of a codebase over time.

Version control also enables collaboration by allowing multiple engineers to work on the same code and configuration files while tracking exactly what changed, when, and by whom. When changes overlap, the system helps detect conflicts and provides mechanisms to resolve them, which is essential for team-based development and infrastructure-as-code. These capabilities are foundational to modern DevOps workflows and are explicitly described in Git-based version control documentation and Cisco DevNet guidance around source control and collaboration.

References: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control, https://docs.github.com/en/get-started/using-git/about-git

QUESTION NO: 28

What are two advantages of version control software? (Choose two.)

A. It supports tracking and comparison of changes in binary format files.

B. It allows new team members to access the current code and history.

C. It supports comparisons between revisions of source code files.

D. It provides wiki collaboration software for documentation.

E. It allows old versions of packaged applications to be hosted on the Internet.

ANSWER: B C

Explanation:

Version control software is designed to help teams collaborate on code safely and efficiently by maintaining a complete history of changes. One key advantage is that it allows new team members to access the current code and history, which makes onboarding much faster: they can clone or check out the repository, review commit history, and understand how and why the code evolved over time. Another core advantage is that it supports comparisons between revisions of source code files. This “diff” capability lets developers see exactly what changed between two versions, which is essential for code reviews, debugging regressions, and auditing changes over time.

These capabilities are fundamental to modern distributed version control systems like Git, where commits form an immutable history and tools like diff/log/blame enable traceability and collaboration workflows (feature branches, pull requests, and reviews). While some platforms bundle extra features, the primary advantages tested at the associate level are history/traceability and the ability to compare revisions of text-based source code. References: Git SCM Book: About Version Control, GitHub Docs: About Git.

QUESTION NO: 29

What are two properties of private IP addresses? (Choose two.)

A. They can be used to access the Internet directly.

B. They are more secure than public IP addresses.

C. They are not globally unique.

D. They can be repeated within the same local network.

E. They are controlled globally by an IP address registry.

F. They can be reused in different private networks without conflict.

ANSWER: C F

Explanation:

Private IPv4 addresses are defined by RFC 1918 and are intended for use inside private networks (such as enterprise LANs, home networks, and lab environments). A key property is that *they are not globally unique*: the same private address ranges can be used by many different organizations at the same time without coordination, because these addresses are not meant to be routed on the public Internet. This reuse is exactly what enables large-scale internal addressing without consuming public IPv4 space.

Another core property is that private addressing is designed to be *reusable across different local networks*. In practice, that means the same private IP (for example, 10.0.0.1) can exist in multiple separate private networks without conflict, as long as those networks are not directly merged without NAT, renumbering, or other overlap-mitigation techniques. This reuse characteristic is central to how private addressing works and is why NAT is commonly used to translate private addresses to public addresses for Internet access.

References: RFC 1918 - Address Allocation for Private Internets, Cisco NAT Overview

QUESTION NO: 30

Which two NETCONF operations cover the RESTCONF GET operation? (Choose two.)

A. Option A

B. Option B

C. Option C

D. Option D

E. Option E

ANSWER: A B

Explanation:

RESTCONF GET is a read-only retrieval of data from a YANG-modeled datastore. In NETCONF, the equivalent “read” capabilities are provided by the `<get>` and `<get-config>` operations. The `<get>` operation retrieves both configuration and operational (state) data, which aligns with RESTCONF GET requests to operational/state resources as well as combined views when supported. The `<get-config>` operation retrieves configuration data from a specific datastore (such as running or candidate), matching RESTCONF GET requests that target configuration resources. Together, these two NETCONF operations cover the same read semantics that RESTCONF exposes via HTTP GET, differing mainly in transport and encoding (NETCONF RPC over SSH vs. RESTCONF over HTTP with XML/JSON). This mapping is a common way to relate RESTCONF’s uniform interface methods to NETCONF’s RPC-based operations for YANG datastores.

References: RFC 6241 (NETCONF Protocol), RFC 8040 (RESTCONF Protocol)

QUESTION NO: 31

What is the meaning of the HTTP status code 204?

A. request completed; new resource created

B. server successfully processed request; no content returned

C. standard response for successful requests

D. invalid query parameters

ANSWER: B

Explanation:

The HTTP status code 204 means “No Content”: the server successfully processed the request and is indicating success, but it is intentionally not returning a message body in the response. This is commonly used for operations where the client doesn’t need to change its current view or render new data—such as a successful DELETE, a PUT that doesn’t need to return a representation, or an AJAX call where the client already has everything it needs. A key behavioral detail is that a 204 response must not include a response body; the semantics are that there is nothing to display, so the client should not expect content to parse. This makes it useful for lightweight acknowledgements of success while avoiding unnecessary payload transfer. In REST-style APIs, 204 is often chosen when the action succeeded but there is no resource representation to return (or returning one would be redundant). For the DevNet Associate exam, remember it as “success with no response content.”

References: MDN Web Docs: 204 No Content, RFC 9110 (HTTP Semantics)

QUESTION NO: 32

Which network constraint causes the performance of the application to decrease as the number of users accessing the application increases?

A. latency

B. loss

C. bandwidth

D. jitter

ANSWER: C

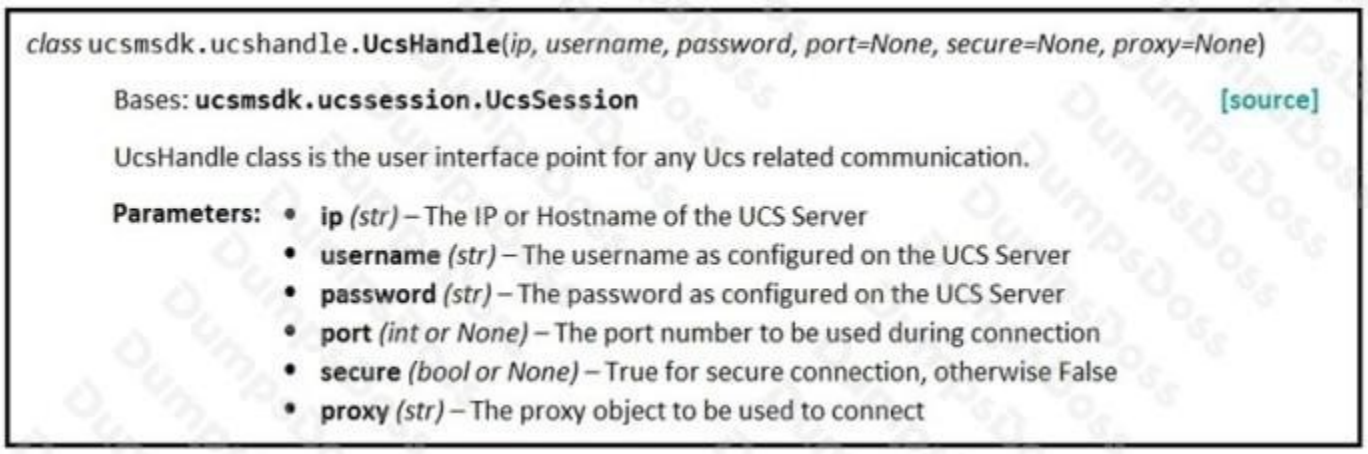
Explanation:

bandwidth is the network constraint that most directly causes application performance to degrade as the number of concurrent users increases. Bandwidth is the maximum amount of data that can be transmitted over a network link in a given time. As more users access an application, aggregate traffic demand rises (more requests, responses, media streams, API calls, etc.). Once total demand approaches or exceeds available link capacity, the network becomes congested and queues build up in interfaces and buffers. This congestion increases response times and can trigger retransmissions and timeouts, which users experience as a “slow” application. In other words, scaling user count increases offered load; when offered load outgrows capacity, throughput per user drops and latency often increases as a secondary effect. This is why capacity planning and bandwidth provisioning (or using QoS, caching, CDNs, and horizontal scaling) are key to maintaining performance under higher concurrency.

References: Cisco QoS: Congestion and Queuing Overview, Bandwidth (computing)

QUESTION NO: 33

Refer to the exhibit.


class ucsmsdk.ucshandle.**UcsHandle**(*ip, username, password, port=None, secure=None, proxy=None*)

Bases: **ucsmsdk.ucssession.UcsSession** [source]

UcsHandle class is the user interface point for any Ucs related communication.

Parameters:
- **ip** *(str)* – The IP or Hostname of the UCS Server
- **username** *(str)* – The username as configured on the UCS Server
- **password** *(str)* – The password as configured on the UCS Server
- **port** *(int or None)* – The port number to be used during connection
- **secure** *(bool or None)* – True for secure connection, otherwise False
- **proxy** *(str)* – The proxy object to be used to connect

Given the API documentation for the UCS SDK Python class, UcsHandle, which code snippet creates a handle instance?

A.
```
#!/usr/bin/env python3
from ucsmsdk.ucshandle import UcsHandle
handle = UcsHandle("10.1.2.254", "admin", "password",
    port="443", secure=True)
```

B.
```
#!/usr/bin/env python3
from ucsmsdk.ucshandle import UcsHandle
handle = UcsHandle(ip="10.1.2.254", username="admin",
    password="password", port=443, secure=True)
```

C.
```
#!/usr/bin/env python3
from ucsmsdk.ucshandle import UcsHandle
handle = UcsHandle("10.1.2.254", "admin", "password",
    port="443", secure=1)
```

D.
```
#!/usr/bin/env python3
import ucsmsdk.ucshandle.UcsHandle
handle = UcsHandle("10.1.2.254", "admin", "password",
    port="443", secure=1)
```

A. Option A

B. Option B

C. Option C

D. Option D

ANSWER: A

Explanation:

Creating a handle instance in the Cisco UCS Python SDK is done by instantiating the `UcsHandle` class (from `ucsmsdk.ucshandle`) with the UCS Manager address (IP or hostname) and user credentials. In other words, you create the object first (the "handle"), and then typically call `login()` on that handle to establish the session. This aligns with the SDK's programming model where the handle represents the authenticated session context used for subsequent managed-object operations (queries, commits, configuration changes). The correct snippet is therefore the one that shows constructing `UcsHandle(...)` to obtain a handle object (for example, `handle = UcsHandle(ip, username, password)`), which is the required first step before any UCSM API calls can be executed through that session.

References: Cisco UCS Python SDK (ucsmsdk) GitHub, ucsmsdk documentation.

QUESTION NO: 34

Which two protocols are associated with the control plane on a network device? (Choose two.)

A. BGP

B. UDP

C. SNMP

D. FTP

E. OSPF

ANSWER: A E

Explanation:

The control plane is the part of a network device responsible for learning network topology, exchanging reachability information, and building the routing/forwarding decisions that the data plane will then use to move packets. Routing protocols are classic control-plane protocols because they form adjacencies/peerings, exchange routing updates, and compute best paths.

BGP is a control-plane protocol used to exchange routing information between autonomous systems (and also within an AS in some designs). It establishes peerings, advertises prefixes, and applies policy to select best paths—functions that directly create the routing table and influence forwarding decisions. OSPF is also a control-plane protocol; it forms neighbor relationships, floods link-state information, runs SPF to compute shortest paths, and installs routes into the routing table. These behaviors are fundamental control-plane operations on routers and multilayer switches.

References: Cisco BGP Technical Overview, Cisco OSPF Overview

QUESTION NO: 35

A file in a local Git repository has been updated and issued the git add . command. The git diff command has been run to compare the changes to the previous commit, but nothing shows. Which action identifies the problem?

A. Run the git add . command again in the correct subdirectory to ensure changes added to the staging area.

B. Run the git commit command before the differences are compared to receive the end state of the code.

C. Run the git status command to see the differences between current and previous code review stages.

D. Run the git diff --staged command to compare the code added to the staging area.

ANSWER: D

Explanation:

Run the git diff --staged command to compare the code added to the staging area.

After you run `git add .`, your modifications are no longer only in the working directory; they are also recorded in the index (the staging area). By default, `git diff` shows the differences between your working directory and the index (what is staged). If the changes have already been staged and your working directory matches the staged content, `git diff` will appear to show “nothing,” even though you do have changes relative to the last commit. To identify what’s happening, you need to compare the staging area against the last commit (HEAD). That is exactly what `git diff --staged` (synonym: `git diff --cached`) does: it displays the patch that is staged and will be included in the next commit. This directly reveals that the changes are present but staged, which explains why the default `git diff` output is empty.

References: https://git-scm.com/docs/git-diff, https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

QUESTION NO: 36

Into which type of Python data structure should common data formats be parsed?

A. sequence

B. set

C. dictionary

D. list

ANSWER: C

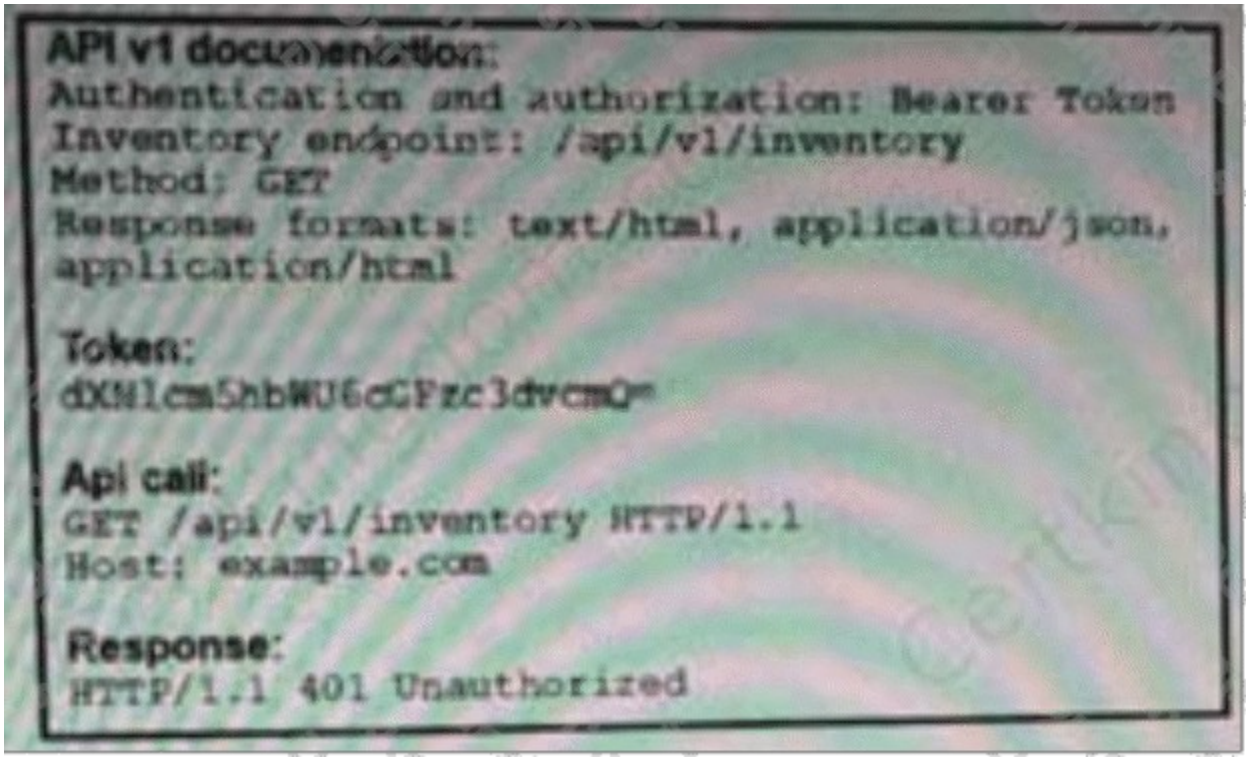
Explanation:

Common structured data formats used in APIs and automation—especially JSON and YAML—are most naturally parsed into a Python dictionary because these formats frequently represent objects/mappings as key-value pairs. In Python, a dictionary (dict) is the native mapping type designed for associating keys to values, which aligns directly with JSON objects and YAML mappings. When you parse JSON with `json.loads()`, JSON objects become Python dicts (and JSON arrays become Python lists), enabling straightforward access like `data["key"]` and easy iteration over fields. Similarly, YAML parsers (for example, PyYAML's `safe_load`) load YAML mappings into dicts. This makes dictionaries the standard target structure for working with API responses, configuration files, and payloads in DevNet-style workflows, because you can reliably navigate nested structures using keys and then serialize back to JSON/YAML when needed.

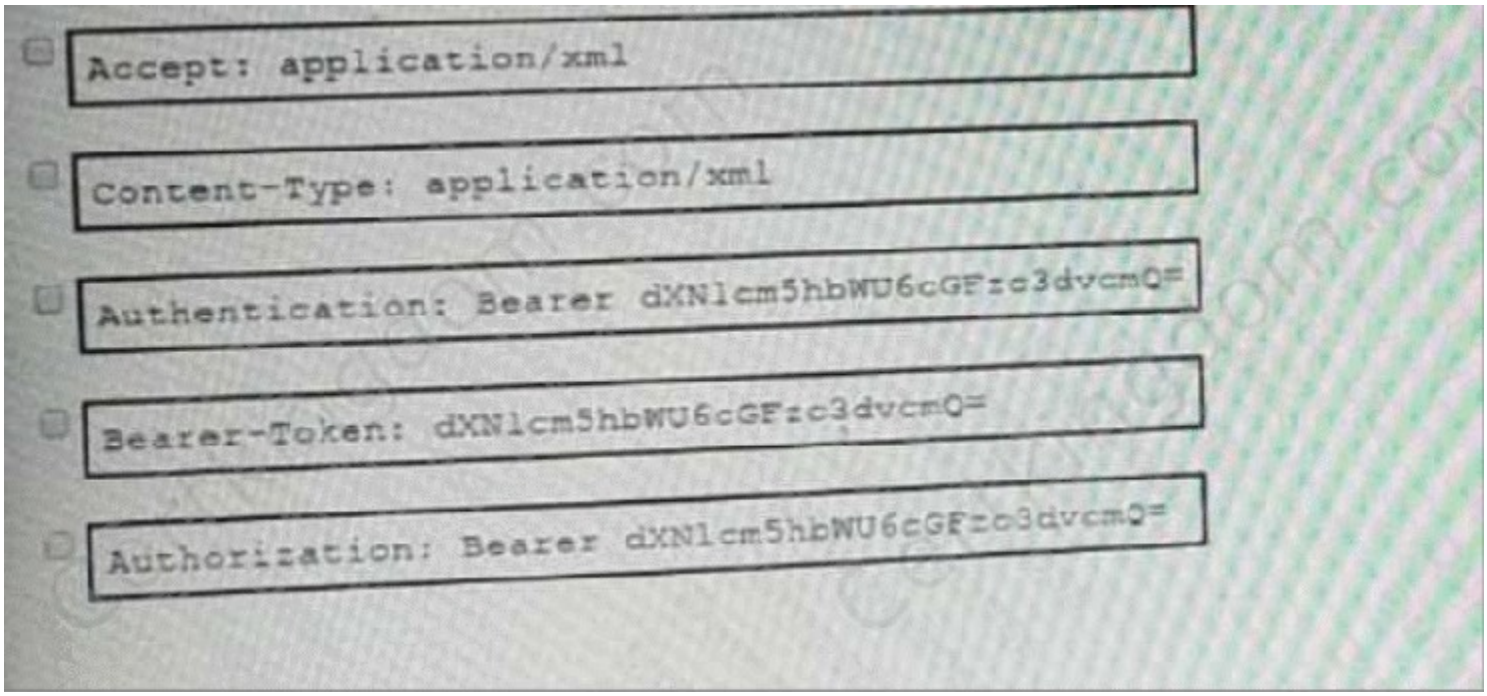
References: https://docs.python.org/3/library/json.html, https://pyyaml.org/wiki/PyYAMLDocumentation

QUESTION NO: 37

Refer to the exhibit.



An API call is constructed to retrieve the inventory in XML format by using the API. The response to the call is 401 Unauthorized. Which two headers must be added to the API call? (Choose two.)



A. Option A

B. Option B

C. Option C

D. Option D

E. Option E

ANSWER: C D

Explanation:

A 401 Unauthorized response from a REST API indicates the server did not accept the request because the client did not present valid authentication credentials. For token-based authentication, the standard way to send the access token is via the HTTP Authorization header using the Bearer scheme (for example, `Authorization: Bearer <access_token>`). Without that header, many Cisco APIs (and REST APIs in general) will reject the request before even evaluating the resource path or response format negotiation.

Because the goal is to retrieve the inventory in XML format, the client should also include an `Accept` header indicating it can accept an XML representation (for example, `Accept: application/xml`). This is the standard HTTP content negotiation mechanism used by REST APIs to determine the response media type. Together, providing the Bearer token for authentication and the Accept header for XML response formatting aligns with common Cisco API patterns and HTTP best practices.

References: MDN Web Docs: Authorization header, MDN Web Docs: Accept header

QUESTION NO: 38

What is a benefit of version control?

A. application of code directly to hardware

B. tracking development changes

C. reuse of code-on-code patches

D. compatibility with back-end systems

ANSWER: B

Explanation:

tracking development changes is a core benefit of version control systems such as Git. A VCS records a project's history over time by storing snapshots (commits) of files along with metadata like author, timestamp, and a message describing the intent of the change. This creates an auditable timeline that helps individuals and teams understand what changed, when it changed, and how the code evolved. Practically, this enables safer collaboration because developers can review differences between versions, identify when a bug was introduced, and roll back to a known-good state if a change causes problems. It also supports workflows like feature branching and code review because changes can be isolated, compared, and merged while preserving the full history. In DevNet-style automation and software development, this traceability is essential for repeatability, troubleshooting, and maintaining quality across iterative releases. Git's own documentation emphasizes that commits and history are fundamental to how Git tracks changes and supports collaboration over time.

References: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control, https://git-scm.com/docs/git-log

QUESTION NO: 39

What is an example of a network interface hardware address?

A. workstation name

B. IP address

C. domain name

D. MAC address

ANSWER: D

Explanation:

A MAC address is an example of a network interface hardware address because it is the link-layer identifier assigned to a network interface controller (NIC) and used for local network communication on technologies like Ethernet and Wi-Fi. In IEEE 802 networks, the MAC address is typically a 48-bit value (often shown as six hexadecimal octets such as 00:1A:2B:3C:4D:5E) that uniquely identifies the interface on the local broadcast domain. Unlike an IP address, which is a logical Layer 3 address that can change based on network configuration (DHCP/static addressing, subnet moves, etc.), the MAC address is intended to identify the physical interface at Layer 2 and is what switches use to learn and forward frames within a LAN. This is why tasks like ARP (IPv4) and Neighbor Discovery (IPv6) ultimately map IP addresses to a MAC address for delivery on the local segment.

References: Cisco: Ethernet and MAC addressing concepts, MAC address overview

QUESTION NO: 40

Which two use cases are supported by Meraki APIs? (Choose two.)

A. Build location-aware apps from Wi-Fi and LoRaWAN devices.

B. Build a custom Captive Portal for Mobile Apps.

C. Configure network devices via the Dashboard API.

D. Deploy applications onto the devices.

E. Retrieve live streams from a Meraki Camera.

ANSWER: A B C

Explanation:

Meraki provides multiple APIs that enable automation and integration use cases across the Meraki platform. A core supported use case is to configure and manage Meraki organizations, networks, and devices programmatically using the Dashboard API. This includes tasks like creating networks, claiming devices, configuring SSIDs, applying firewall rules, and monitoring health/status at scale—exactly what "Configure network devices via the Dashboard API." describes.

Another supported use case is building a custom captive portal experience by integrating with Meraki's captive portal capabilities (often implemented via external splash pages and authentication workflows). This aligns with "Build a custom Captive Portal for Mobile Apps." because developers can control user onboarding/authentication flows and integrate with external identity systems while still leveraging Meraki wireless access control features.

These are well-documented, common Meraki API-driven workflows used for network automation and user access experiences. For details, see the Meraki Dashboard API documentation and the Meraki developer resources around wireless splash/captive portal integrations: https://developer.cisco.com/meraki/api-v1/ and https://documentation.meraki.com/General_Administration/Other_Topics/Cisco_Meraki_Dashboard_API.

QUESTION NO: 41

Which two details are captured from the subnet mask? (Choose two.)

A. portion of an IP address that refers to the subnet

B. default gateway of the host

C. unique number ID assigned to the host

D. part of an IP address that refers to the host

E. network connection of a host or interface

ANSWER: A D

Explanation:

A subnet mask (or prefix length) is used together with an IPv4 address to determine which bits represent the network/subnet portion and which bits represent the host portion. In other words, the mask is a bitwise selector: the bits set to 1 identify the subnet (network) part of the address, and the bits set to 0 identify the host part. From this, devices can determine whether a destination IP is local (same subnet) or remote (different subnet), and they can derive key subnet characteristics such as the number of host addresses available and the subnet's address range. Therefore, the details captured from the subnet mask are the portion of an IP address that refers to the subnet and the part of an IP address that refers to the host. These are foundational concepts in IPv4 addressing and subnetting and are consistently described in Cisco networking materials and standard IP addressing references.

References: Cisco - IP Addressing and Subnetting for New Users, Subnetwork (subnet mask overview)

QUESTION NO: 42

A developer creates a script that configured multiple Cisco IOS XE devices in a corporate infrastructure. The internal test environment is unavailable, and no maintenance window is available to test on a low-priority production environment. Which resource is used to test the code before it is applied to the production environment?

A. Code Exchange

B. Cisco DevNet Learning Labs

C. Cisco DevNet Sandbox

D. Cisco Support

ANSWER: C

Explanation:

The correct resource to test automation code and configuration changes safely before touching production is the Cisco DevNet Sandbox. DevNet Sandbox provides reservable, cloud-hosted lab environments that include real Cisco IOS XE devices (or realistic virtual equivalents) and supporting infrastructure, allowing you to validate connectivity, authentication, API calls, and configuration workflows end-to-end. This is exactly what you need when you cannot access an internal test environment and cannot risk running changes in production without a maintenance window. By using a sandbox, you can iterate on the script, confirm idempotency and error handling, and verify the resulting device state in an isolated environment that mirrors real operational conditions. Cisco offers multiple sandbox types (always-on and reservable) so you can choose an environment that matches your use case and test schedule. This aligns with DevNet best practices for pre-production validation of network automation and infrastructure-as-code workflows.

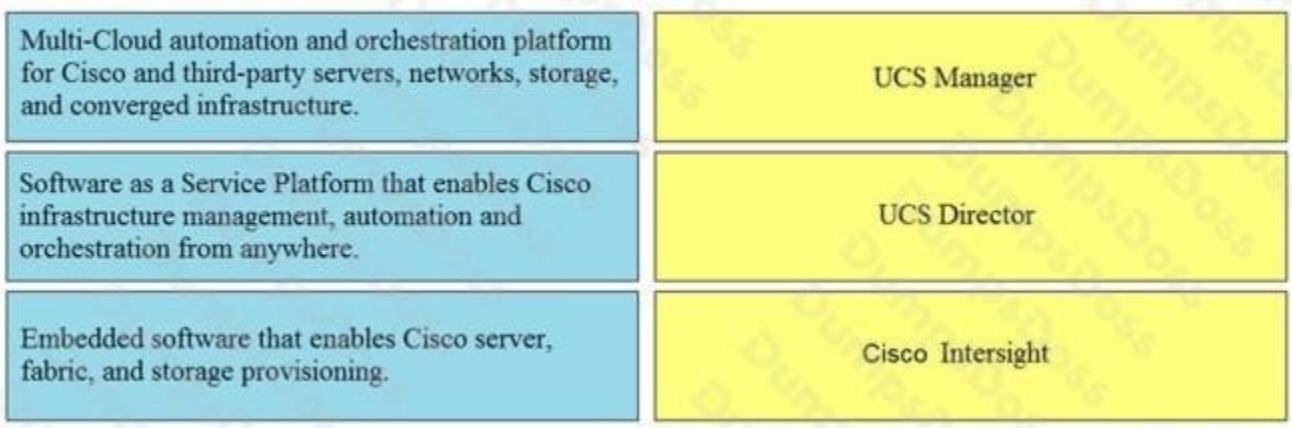
References: Cisco DevNet Sandbox, DevNet Sandbox Documentation

QUESTION NO: 43 - (DRAG DROP)

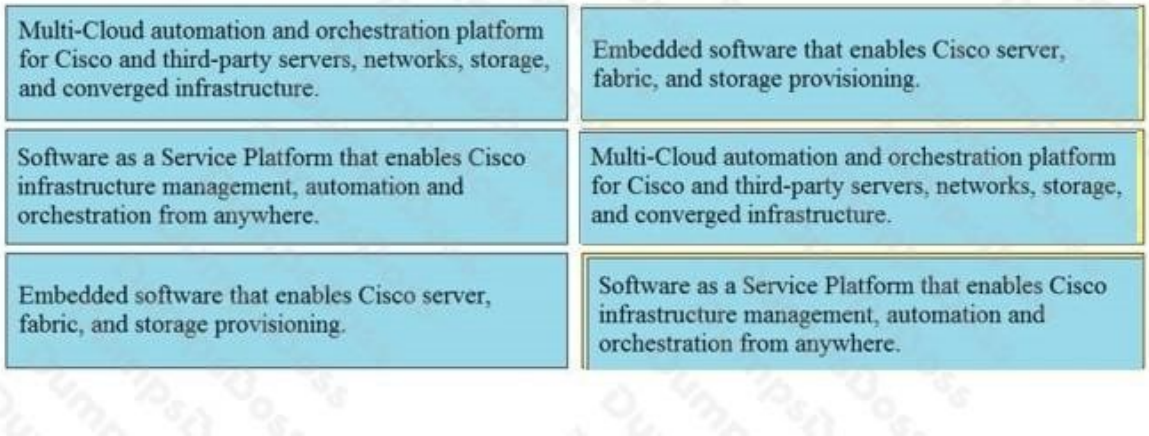
DRAG DROP

Drag and drop the capability on the left onto the Cisco compute management platform that supports the capability on the right.

Select and Place:



ANSWER:



Explanation:

This match is based on what each Cisco compute management product is designed to do. Cisco UCS Manager is the embedded, on-premises management component that lives with the UCS domain and provides the core “UCS” functions such as defining service profiles and provisioning/configuring server identity, fabric connectivity, and related storage access.

That's why the capability describing "embedded software that enables Cisco server, fabric, and storage provisioning" aligns with UCS Manager.

Cisco UCS Director is positioned as an automation and orchestration platform that can coordinate workflows across compute, network, and storage, and it can extend beyond UCS into third-party infrastructure. That broader, cross-domain orchestration focus is why the "multi-cloud automation and orchestration platform for Cisco and third-party servers, networks, storage, and converged infrastructure" capability maps to UCS Director.

Cisco Intersight is Cisco's cloud-delivered (SaaS) operations and management platform for UCS and other Cisco infrastructure, providing centralized management, automation, and orchestration accessible from anywhere with an internet connection. That directly matches the capability describing a "Software as a Service Platform that enables Cisco infrastructure management, automation and orchestration from anywhere."

References: Cisco UCS Manager, Cisco UCS Director, Cisco Intersight.

QUESTION NO: 44

In which two ways is an application characterized when interacting with a webhook? (Choose two.)

A. codec

B. receiver

C. transaction monitor

D. processor

E. listener

ANSWER: B E

Explanation:

When working with webhooks, the application on the "consuming" side is commonly described as a *listener* or a *receiver* because it exposes an HTTP endpoint and waits for inbound HTTP requests that are triggered by events in another system. In an event-driven pattern, the producing system sends an HTTP POST (or similar) to the webhook URL whenever a subscribed event occurs, and the receiving application handles that request immediately—often validating the request, parsing the payload, and then taking follow-on actions (such as updating a database, triggering a workflow, or calling another API). This is why "listener" and "receiver" are the best characterizations: they emphasize that the application is passively awaiting event notifications rather than polling for changes. This aligns with the general webhook model used across many platforms, where the sender pushes event data to the receiver's callback URL in near real time. For additional background on webhooks and event-driven integrations, see https://developer.github.com/webhooks/ and https://stripe.com/docs/webhooks.

QUESTION NO: 45

What are two characteristics of Bare Metal environments that are related to application deployment? (Choose two.)

A. specifically designed for container-based workloads

B. suitable for legacy applications that do not support virtualization

C. provides workloads with access to hardware features

D. not compatible with other cloud services such as PaaS or SaaS offerings

E. provides the hypervisor to host virtual servers

ANSWER: B C

Explanation:

Bare metal environments provide direct access to the underlying physical server without an abstraction layer like a hypervisor managed by the provider. This makes them a strong fit when applications need to use specific CPU instructions, specialized NIC capabilities, GPU/FPGA acceleration, or other hardware-dependent features; in other words, the deployment can take advantage of “access to hardware features” with minimal overhead and maximum performance consistency.

Bare metal is also commonly used for workloads that cannot be easily virtualized or are not supported in virtualized environments. Many legacy applications (or appliances with strict licensing, timing, or kernel/driver dependencies) may require installation directly on physical hardware, making “suitable for legacy applications that do not support virtualization” a key deployment-related characteristic.

In cloud contexts, bare metal can still be integrated with higher-level services, but the defining deployment traits are the lack of provider-managed virtualization and the ability to run software directly on the server OS with full hardware control. This aligns with common cloud and infrastructure guidance on when to choose bare metal for performance and hardware dependency reasons.

References: Cisco: What is bare metal?, AWS: What is bare metal?

QUESTION NO: 46

What are two advantages of the Model-View-Controller software design pattern? (Choose two.)

A. allows for multiple views of the same model

B. separates responsibilities of the code, which makes future modifications easier

C. simplifies network automation

D. makes code easier to deploy using CI/CD pipelines

E. reduces the need for error handling

ANSWER: A B

Explanation:

The Model-View-Controller (MVC) pattern is designed to improve application structure by splitting an app into three cooperating components: the model (data and business rules), the view (presentation), and the controller (input/flow). A key advantage is that it allows for multiple views of the same model, meaning the same underlying data and logic can be rendered in different formats (for example, HTML, JSON, or a mobile UI) without duplicating the core business logic. This supports reuse and consistent behavior across different user interfaces.

Another major advantage is that it separates responsibilities of the code, which makes future modifications easier. Because presentation concerns are isolated from business logic and request-handling flow, teams can change UI layouts, update validation/business rules, or adjust routing/controller behavior with less risk of unintended side effects in other layers. This separation also tends to improve testability (unit testing models/controllers independently of views) and maintainability over time.

References: MDN Web Docs - MVC, Microsoft Learn - ASP.NET Core MVC overview

QUESTION NO: 47

Which two concepts describe test-driven development? (Choose two.)

A. User acceptance testers develop the test requirements.

B. It enables code refactoring.

C. Tests are created when code is ready for release.

D. Implementation is driven by incremental testing of release candidates.

E. Write a test before writing code.

ANSWER: B E

Explanation:

Test-driven development is a practice where you first define the expected behavior in an automated test, then write the minimum code needed to make that test pass, and finally improve the design while keeping the tests green. Because tests are written up front and run frequently, the tests effectively guide (or “drive”) the implementation work in small, incremental steps. This is why “Write a test before writing code.

” is a core concept of TDD: the failing test comes first, then code is added to satisfy it.

TDD also strongly supports “It enables code refactoring.” Once you have a reliable suite of automated tests, you can safely restructure and clean up code (refactor) with confidence that you haven’t broken existing behavior, since the tests provide rapid feedback. This tight loop of test → code → refactor is central to TDD and is widely documented as a primary benefit of the approach.

References: Martin Fowler – TestDrivenDevelopment, Agile Alliance – Test Driven Development (TDD)

QUESTION NO: 48

How are load balancers used in modern application deployment?

A. Allow traffic to continue as new compute units are brought up and old compute units are taken down.

B. Allow http and https traffic to continue as old compute units are discontinued before new units are brought up.

C. Turn off traffic and take down compute units, then update and bring the compute units back up.

D. Bring up new compute units, test the compute units, and switch the traffic from old units to new units.

ANSWER: A

Explanation:

Allow traffic to continue as new compute units are brought up and old compute units are taken down is the best description of how load balancers are used in modern application deployments. A load balancer sits in front of a pool of application instances and distributes incoming requests across healthy backends. As you scale out (adding instances) or scale in (removing instances), the load balancer updates its backend membership and health checks so users keep reaching available instances without needing to know which specific server is serving them. This pattern is fundamental to elastic scaling, rolling updates, and high availability: new instances can be registered and start receiving traffic once healthy, while old instances can be drained (stop receiving new connections) and then terminated, minimizing disruption. In cloud-native environments (including Kubernetes Services/Ingress and cloud LBs), this enables continuous delivery practices where capacity and versions change dynamically while the service endpoint remains stable. Cisco’s DevNet curriculum emphasizes these deployment concepts as part of designing resilient applications and infrastructure that can change underneath without interrupting consumers.

References: Kubernetes Service concepts, AWS Application Load Balancer introduction

QUESTION NO: 49

Which Cisco DevNet resource allows access to products in a development lab to explore, learn, and build applications that use Cisco APIs?

A. DevNet Code Exchange

B. DevNet Sandbox

C. DevNet Communities

D. DevNet Automation Exchange

ANSWER: B

Explanation:

DevNet Sandbox is the Cisco DevNet resource specifically designed to provide on-demand (and reservable) access to real Cisco products and prebuilt lab environments so developers can explore features, run API calls, and test integrations without needing to own the hardware or build the infrastructure themselves. These sandbox environments commonly include always-on labs and time-based reservations, along with credentials and connectivity details, enabling you to learn and build against Cisco APIs in a realistic setting. This directly matches the idea of “access to products in a development lab” for exploration and application development. DevNet Sandbox is widely used for practicing with platforms like Cisco DNA Center, Meraki, Webex, IOS XE, and others, making it a core DevNet learning and experimentation resource for API-driven development and automation workflows. For official details, see the DevNet Sandbox landing page and overview documentation on Cisco DevNet.

References: https://developer.cisco.com/site/sandbox/, https://developer.cisco.com/docs/sandbox/

QUESTION NO: 50 - (HOTSPOT)

An engineer clones a repository that contains a Python script from GitLab to a laptop. After modifying the code, the engineer wants to validate the changes. Which command starts '¦ CI/CD pipeline and runs the automated tests?

ANSWER:

Explanation:

To start a GitLab CI/CD pipeline, you need to create an event on the remote GitLab repository that GitLab can react to. After you clone a repo and modify code on your laptop, those changes exist only in your local working directory until you record them in Git history and then send them back to the remote server. The command that actually publishes your new commit(s) to the GitLab remote is `git push`. A push updates the branch on the GitLab server, and that server-side update is what triggers the CI/CD pipeline defined for the project (typically via a `.gitlab-ci.yml` file). Once triggered, the pipeline runs the configured stages, which commonly include automated unit tests, linting, and other validation steps.

In practical workflow terms, you usually run `git commit` first to create a local commit, and then `git push` to upload that commit to GitLab. The key point for CI/CD is that GitLab runners execute jobs based on repository events occurring on GitLab itself, not based on local-only actions. That's why pushing your changes is the action that starts the pipeline and runs the automated tests.

References: GitLab CI/CD pipelines, GitLab pipeline triggers (push events)

QUESTION NO: 51

Which two elements are foundational principles of DevOps? (Choose two.)

A. organizing cross-functional teams over organizational silos

B. designing applications as microservices

C. encouraging containers for the deployment of applications

D. automating over documenting

E. optimizing the cost of infrastructures

ANSWER: A D

Explanation:

DevOps is fundamentally about improving the flow of work from development to operations by removing friction between groups and enabling fast, reliable delivery. A core principle is organizing cross-functional teams over organizational silos, because DevOps emphasizes shared ownership and collaboration across development, operations, security, and other stakeholders. This reduces handoff delays, improves feedback loops, and aligns everyone around delivering customer value.

Another foundational principle is automating over documenting. DevOps practices prioritize automation of build, test, deployment, and infrastructure changes (often expressed as code) so that releases are repeatable, consistent, and less error-prone. Documentation still matters, but DevOps leans on automation to make processes executable and verifiable, which supports continuous integration and continuous delivery.

These principles are widely reflected in DevOps guidance such as the CALMS model (Culture and Automation as key pillars) and in industry definitions that stress collaboration and automation to accelerate delivery while maintaining reliability. For additional background, see AWS: What is DevOps? and Microsoft Learn: What is DevOps?.

QUESTION NO: 52

A team of developers is responsible for a network orchestration application in the company. The responsibilities also include:

developing and improving the application in a continuous manner deployment of the application and management of CI/CD frameworks

monitoring the usage and problems and managing the performance improvements Which principle best describes this DevOps practice?

A. responsible for IT operations

B. automation of processes

C. end-to-end responsibility

D. quality assurance checks

ANSWER: C

Explanation:

The DevOps principle described is end-to-end responsibility: the same team owns the application across its full lifecycle, not just writing code. In practice, that means the developers who build and continuously improve the network orchestration app also take responsibility for how it is delivered (deployment and CI/CD pipeline ownership), how it behaves in production (monitoring usage, detecting problems), and how it evolves operationally (performance tuning and reliability improvements). This “you build it, you run it” model reduces handoff friction between development and operations, shortens feedback loops, and encourages designing for operability from the start (instrumentation, logging, safe deployments, rollback strategies). It also aligns incentives: the team that introduces changes is accountable for their impact in production, which typically improves quality and stability while enabling faster iteration. This is a core DevOps cultural practice that complements automation and CI/CD tooling, but is fundamentally about ownership and accountability across development, delivery, and operations.

References: AWS: What is DevOps?, Google Cloud: DevOps culture