PassQueen
a.k.a dumpsqueen.com

DevNet Associate (DEVASC)

Cisco 200-901

Version Demo

Total Demo Questions: 76

Total Premium Questions: 765

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Software Development and Design	154
Topic 2, Understanding and Using APIs	268
Topic 3, Cisco Platforms and Application Development	88
Topic 4, Application Deployment and Security	92
Topic 5, Infrastructure and Automation	86
Topic 6, Network Fundamentals	77
Total	765

QUESTION NO: 1

FILL BLANK

Fill in the blanks to complete the statement.

Given a username of “devnet” and a password of “cisco123”, applications must create a base64 encoding of the string “_____” when sending HTTP requests to an API that uses _____ authentication.

A. See explanation below.

B. devnet:cisco123 and Basic

ANSWER: B

Explanation:

The correct completion is `devnet:cisco123 and Basic`. HTTP Basic authentication defines the credential value as a username and password concatenated with a single colon, in the precise form `username:password`. Therefore, using the supplied credentials produces `devnet:cisco123` as the input string. The client Base64-encodes the bytes of that full string and sends the result in the HTTP `Authorization` request header with the authentication scheme identifier, such as `Authorization: Basic ZGV2bmV0OmNpc2NvMTIz`. The Base64 value is an encoding for HTTP header transport, not encryption or a password hash. Consequently, APIs using this mechanism should be accessed through HTTPS so that the credentials are protected while in transit. RFC 7617 specifies that the Basic scheme obtains the user-pass value by concatenating the user ID, a colon character, and the password before Base64 encoding it. See RFC 7617: The Basic HTTP Authentication Scheme and MDN Web Docs: HTTP authentication.

QUESTION NO: 2

```
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: A

Explanation:

`enable_function(Loopback1, true, 'iana-if-type:softwareLoopback')` is the command that supplies the required values to enable the specified loopback interface in the expected order: the interface name, the enabled state, and the interface type. The interface name identifies the particular configured interface instance, while the Boolean value `true` represents its desired administrative enabled state. The `iana-if-type:softwareLoopback` identity identifies the interface as a software loopback under the IANA interface-type definitions used by the standard interface YANG model. In RESTCONF automation, these inputs are used to construct a request targeting the appropriate interface resource and JSON content containing the interface identity and its `enabled` setting. The standard `ietf-interfaces` model defines `enabled` as the configurable administrative-status control for an interface, and identifies the interface list using its name. Cisco RESTCONF implementations use this model-driven approach when standard IETF interface resources are addressed. See RFC 8343, A YANG Data Model for Interface Management and RFC 8040, RESTCONF Protocol.

QUESTION NO: 3

Which configuration management tool has an agentless capability?

A. Chef

B. Puppet

C. Ansible

D. CFEngine

ANSWER: C

Explanation:

Ansible is the configuration management tool with an agentless capability. Its control node connects to managed Linux and Unix hosts primarily over SSH and executes automation tasks remotely; a persistent Ansible agent or daemon is not required on each managed host. For Windows targets, Ansible commonly uses WinRM or PowerShell remoting. This architecture lets teams automate configuration, application deployment, and orchestration while avoiding endpoint agent installation, upgrades, and lifecycle management. Ansible modules are transferred and run as needed for a task, and the control node receives the execution result afterward. Cisco DevNet candidates should associate Ansible's agentless model with its use of standard remote-management protocols and inventory-driven automation. The model is particularly useful when rapid onboarding, minimal host-side software, and centralized automation control are desired. Ansible's official documentation describes how it manages hosts through connections such as SSH and notes that managed nodes do not need Ansible installed. Red Hat also identifies Ansible Automation Platform as agentless automation, emphasizing that it can work without deploying agents across managed systems. See the Ansible Getting Started documentation and Red Hat's What is Ansible? overview.

QUESTION NO: 4

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:

interface management is represented by the YANG module. YANG is a data-modeling language used to define a device's manageable configuration and operational-state data, along with the operations that can act on that data. An interface-focused module models network interfaces as structured data, commonly with a container or list representing interfaces and child nodes that identify each interface and expose its configurable and observed properties. Typical modeled values include an interface name or type, administrative state, operational state, addressing, traffic counters, and link-related attributes.

This schema lets automation clients interact with interfaces in a vendor-neutral, hierarchical format rather than relying on manually parsed command-line output. A client can use NETCONF or RESTCONF to retrieve interface state, configure supported interface properties, and validate data according to the constraints defined in the model. The standardized `ietf-interfaces` model is a representative example: it defines interface configuration and state concepts and is intended for use by management protocols that operate on YANG data. Cisco platforms similarly expose interface data through relevant native and standards-based YANG models. See RFC 7950, The YANG 1.1 Data Modeling Language and RFC 8343, A YANG Data Model for Interface Management.

QUESTION NO: 5

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 address space is specified by RFC 1918 for use within private internets. “They are not globally unique” is correct because organizations may independently use the same addresses from the private ranges—10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16—without obtaining those addresses from a global allocation authority. These addresses are intended for internal addressing and are not globally routable on the public Internet.

“They can be reused in different private networks without conflict” is also correct. For example, separate companies can both use 10.0.0.0/24 internally because their networks are distinct routing domains. Reuse does not create an addressing conflict unless the networks must be interconnected with overlapping address space. In that circumstance, network address translation, renumbering, or another overlap-management design may be necessary. Private-address reuse conserves scarce public IPv4 addresses, while NAT commonly allows privately addressed hosts to communicate externally through one or more public addresses. RFC 1918 defines the private-use ranges and states that they are not to be routed on the public Internet. See RFC 1918: Address Allocation for Private Internets and Cisco’s NAT Overview.

QUESTION NO: 6

Exhibit.

```
api_endpoint = f"https://veryusefulapi/v2/authenticate"

headers = {
    "Authorization": "Basic YWRtaW46c3VwZXJzZWNyZXQ="
    "Accept": "application/json"
}

req = requests.get(api_endpoint, headers=headers)
```

Refer to the exhibit. A developer is part of a team that is working on an open-source project in which source code is hosted in a public GitHub repository. While the application was built, security concerns were addressed by encrypting the credentials on the server. After a few months, the developer realized that a hacker managed to gain access to the account. The exhibit contains part of the source code for the login process. Why was the attacker able to access the developer's account?

A. The encoded credentials were available in the source code.

B. The application was not encrypting the communication with the server.

C. The credentials were encrypted in the source code.

D. An SSL certificate was used instead of the TLS protocol to authenticate.

ANSWER: A

Explanation:

The encoded credentials were available in the source code. Because the repository is public, anyone can inspect its files, retrieve the embedded credential value, and decode it if it was merely encoded rather than protected by a cryptographic secret-management mechanism. Encoding formats, such as Base64, transform data for transport or representation but do not provide confidentiality: the original value can be recovered without a secret key. Consequently, placing an encoded username, password, API key, or token in public source code exposes a reusable authentication secret to every repository viewer, including attackers and automated secret-scanning tools.

Secure development practice is to keep credentials out of source control entirely. Applications should receive secrets at runtime from protected environment variables, a CI/CD secret store, or a dedicated secrets-management service. Once a credential has been committed to a public repository, it should be treated as compromised and rotated, because deleting it from a later commit does not remove it from repository history or from copies that may already exist. GitHub documents how exposed tokens and other credentials can be detected through secret scanning, while OWASP identifies hard-coded passwords as a security vulnerability. See GitHub Docs: About secret scanning and OWASP: Use of Hard-coded Password.

QUESTION NO: 7

A distributed application was developed using the Cisco intersight SDK. While testing the interaction between the application and the intersight API. the requests that were sent to ..server could not execute because the application was blocked by the firewall. Which URL and port must be provided to the firewall administrator to allow the traffic?

A. svc.intersight.com port 443

B. intersight.com port 443

C. intersight.com port 80

D. svc.intersight.com port 80

ANSWER: B

Explanation:

intersight.com port 443 is correct because Cisco Intersight's public REST API is accessed through the Intersight API base endpoint at `https://intersight.com/api/v1`. Applications built with an Intersight SDK authenticate and submit API requests to that HTTPS endpoint, so an egress firewall policy must permit the application host to establish TCP connections to the `intersight.com` FQDN on port 443. Port 443 is required for HTTPS, which provides the TLS-protected transport used for API authentication, request submission, and response handling. The firewall administrator should implement an outbound rule that supports DNS resolution for the FQDN and permits the corresponding HTTPS sessions. If the organization uses a forward proxy or TLS inspection, its configuration must also allow the application to complete a valid TLS connection to the Intersight API endpoint. Cisco's API documentation identifies the Intersight REST API path under the `intersight.com` host, and Cisco's SDK documentation uses that host when configuring API clients. See the Cisco Intersight API overview and Cisco DevNet Intersight documentation.

QUESTION NO: 8

A network engineer makes several API calls lo Cisco Prime to retrieve a list of all devices. Each time a response is received, only a subset of the devices is returned. The engineer :ices that HTTP code 429

is returned instead of 200 for some API calls. Why did the response exclude some of the devices?

A. The API applied an offset that was indicated In the request.

B. The API failed to identify how many items to retrieve.

C. The API timed out the request.

D. The API rate limited the request

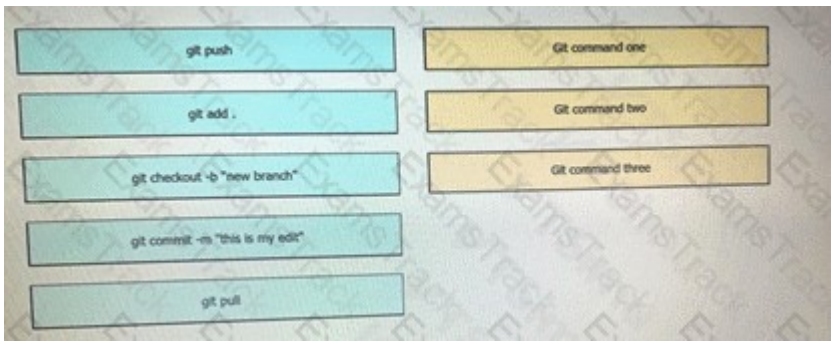
ANSWER: D

Explanation:

The API rate limited the request is correct. HTTP status code 429 means *Too Many Requests*, indicating that the API server has received requests from the client more frequently than its configured allowance permits. Rate limiting is commonly used by API platforms to preserve service availability, prevent excessive load, and ensure that a single client does not consume disproportionate server resources. In this situation, requests that receive a 429 response were not successfully processed as normal device-list retrieval requests. If the client continues through its sequence without appropriately retrying those rejected calls, the collected results can contain only the devices returned by successful requests, making the overall device inventory appear incomplete. A client should detect 429 responses, reduce its request frequency, and retry the affected request after the server-specified delay when a `Retry-After` header is present. Using exponential backoff with retry logic is also a standard resilient-client practice for temporary throttling conditions. The HTTP definition of status 429 explicitly identifies it as a rate-control response and permits servers to include guidance about when the client may retry. See RFC 6585, Section 4 and MDN Web Docs: 429 Too Many Requests.

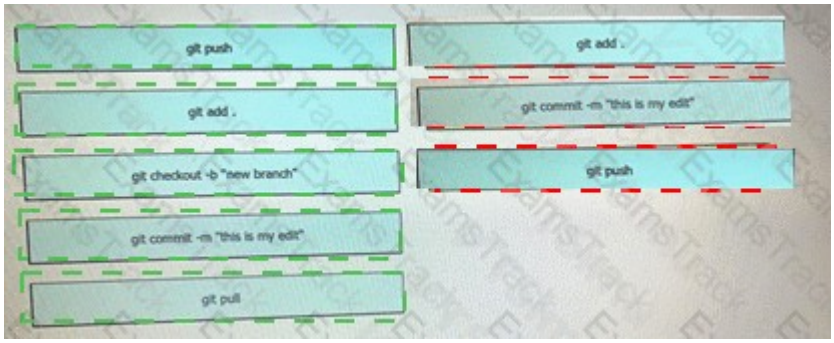
QUESTION NO: 9 - (DRAG DROP)

Drag and Drop the GIT commands from the left onto the right that add modified local files to a remote repository. Not all options are used



ANSWER:

Answer:



Answer:

Explanation:

git add .git commit -m “this is my edit”git push

Explanation:

The correct workflow is `git add .`, followed by `git commit -m "this is my edit"`, followed by `git push`. Git uses a deliberate sequence when local file changes must become part of a remote repository. First, `git add .` places changes in the working directory into Git’s staging area. The period tells Git to consider changes under the current directory, which is useful when the intended update includes all modified local files in that project location. Staging establishes the exact content intended for the next snapshot.

Next, `git commit -m "this is my edit"` records the staged content as a new commit in the local repository. A commit is Git’s local, durable project-history snapshot, and the message documents the purpose of that snapshot. Until this command is run, the staged edits are not yet represented by a new local commit that can be shared as repository history.

Finally, `git push` transfers the local commit to its configured remote and updates the corresponding remote branch. This is the operation that publishes the completed local work for other collaborators or remote services to access. GitHub's documentation describes this same flow: stage changes, commit the staged snapshot, and push commits to the remote repository. See GitHub's Hello World Git workflow and the official git push documentation. Therefore, the shown placement correctly stages the modified files, saves them as a local commit, and publishes that commit remotely.

QUESTION NO: 10

Which Python function is used to parse a string that contains JSON data into a Python dictionary?

A. json.dumps()

B. json.to_json()

C. json.parse()

D. json.loads()

ANSWER: D

Explanation:

`json.loads()` is used to deserialize JSON content held in a Python string. The name reflects its purpose: `loads` means "load string." After importing Python's standard-library `json` module, an application can call `json.loads(json_string)` to convert JSON text into the equivalent native Python value. When the input JSON is an object, represented by curly braces containing name/value pairs, the returned value is a `dict`. For example, `json.loads('{"device":"router","ports":8}')` produces a dictionary from which values can be retrieved with normal dictionary syntax, such as `data["device"]`. This is especially common in DevNet workflows, where an API response body containing JSON must be processed programmatically. Python's JSON documentation defines `loads()` as deserializing a `str`, `bytes`, or `bytearray` containing a JSON document. The documentation also notes that the resulting Python object depends on the JSON top-level value; specifically, a JSON object maps to a Python dictionary. See the official Python json module documentation and the json.loads() API reference.

QUESTION NO: 11

Refer to the exhibit.

```
{
    "ietf-interfaces:interfaces": {
        "interface": [
            {
                "name": "GigabitEthernet1",
                "description": "Network interface",
                "type": "iana-if-type:ethernetCsmacd",
                "enabled": true,
                "ietf-ip:ipv4": {
                    "address": [
                        {
                            "ip": "10.10.20.48",
                            "netmask": "255.255.255.0"
                        }
                    ]
                }
            },
            {
                "name": "GigabitEthernet2",
                "description": "Network Interface",
                "type": "iana-if-type:ethernetCsmacd",
                "enabled": false,
                "ietf-ip:ipv4": {}
            },
            {
                "name": "GigabitEthernet3",
                "description": "Network Interface",
                "type": "iana-if-type:ethernetCsmacd",
                "enabled": false,
                "ietf-ip:ipv4": {}
            }
        ]
    }
}
```

What are two results of running the RESTCONF query? (Choose two.)

A. Interfaces are created as containers.

B. The module letf-interfaces:interfaces is created.

C. GigabitEthemet2 becomes shut down.

D. GigabitEthernetl becomes shut down.

E. Descriptions for two interfaces are applied.

ANSWER: C E

Explanation:

The RESTCONF request applies the configuration represented by the supplied YANG-modeled JSON payload to the targeted interface data resource. In the payload, the configuration for GigabitEthernet2 includes an `enabled` value of `false`. Within the standard interface YANG model, this leaf controls the intended administrative state; setting it to false administratively disables the interface, which is equivalent to placing it in the shutdown state. The request also supplies description values for two interface list entries. Because descriptions are configurable interface attributes, those values are written as part of the configuration update and become the descriptions associated with the respective interfaces. RESTCONF operates on data resources defined by YANG, and a request that creates or replaces configuration updates the addressed configuration datastore according to the request method and resource path. Cisco IOS XE RESTCONF uses YANG models to expose and configure these resources. See Cisco's IOS XE RESTCONF documentation and the IETF Interface Management YANG Model, which defines the `enabled` and `description` interface configuration leaves.

QUESTION NO: 12

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:

"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." is correct because version control systems such as Git let teams create branches that preserve an independent line of development. A developer can implement a feature, fix a defect, or test an experimental change without immediately affecting the shared codebase. Once the work is ready and has been reviewed, merging integrates the selected changes into the target branch. This supports controlled releases and parallel development while retaining a complete history of the integration work.

"It allows multiple engineers to work against the same code and configuration files and manage differences and conflicts." is also correct because a version control system records revisions, identifies overlapping edits, and provides workflows for reconciling conflicts. Team members can collaborate on application source code, API definitions, device configurations, and infrastructure-as-code while retaining authorship and change history. This traceability enables teams to inspect prior versions, compare revisions, and safely coordinate work across a shared repository. Git documentation describes branching as a lightweight mechanism for divergent work and explains how distributed repositories enable collaboration and history tracking. See Git: About Version Control and GitHub Docs: About Git.

QUESTION NO: 13

What is a capability of the AXL API?

A. It signs a user in to a phone that is configured for extension mobility.

B. It pulls logs for the Cisco Tomcat service.

C. It authenticates users who exist in Cisco Unified Communications Manager.

D. It provides support for HTTP and HTTPS communications.

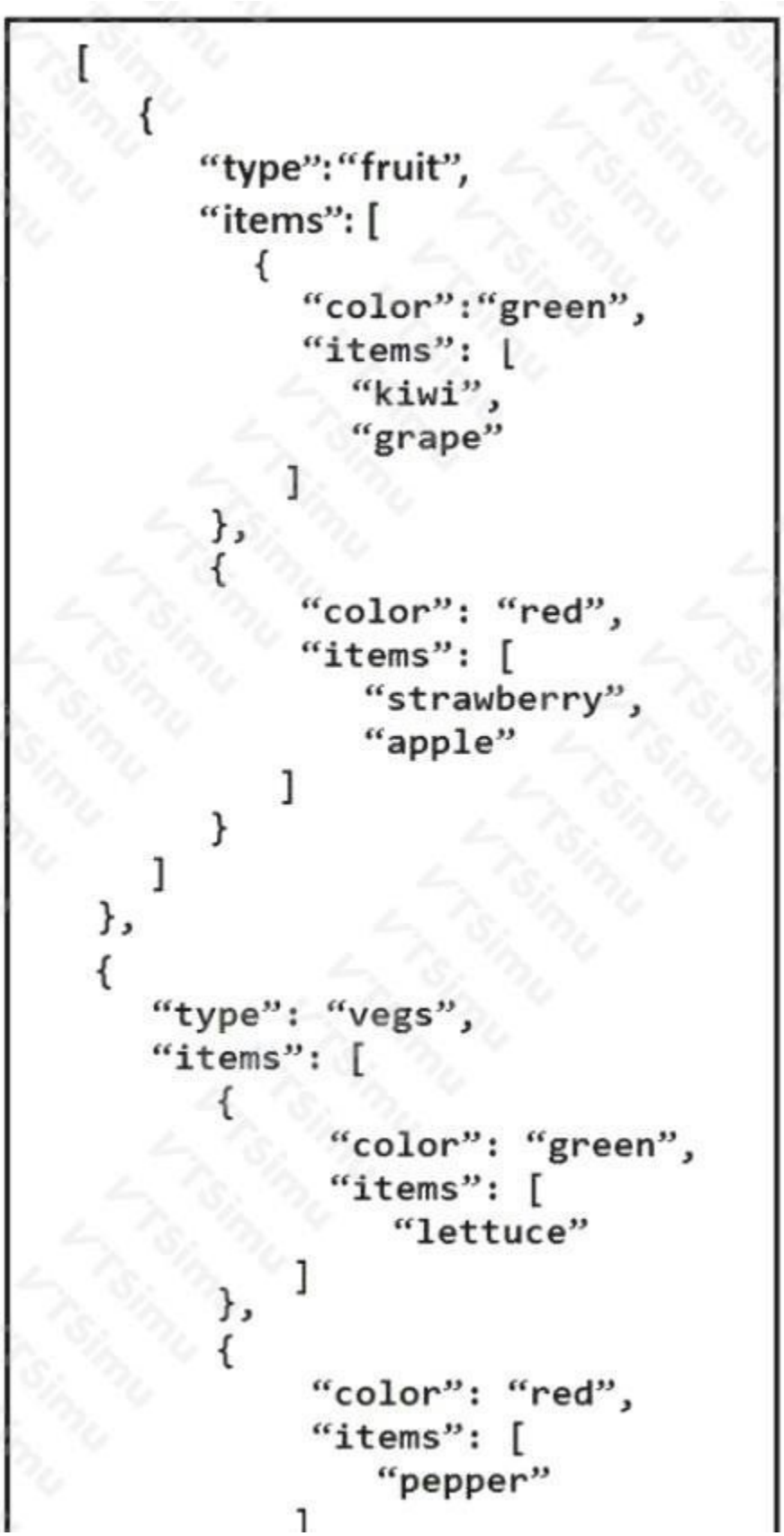
ANSWER: D

Explanation:

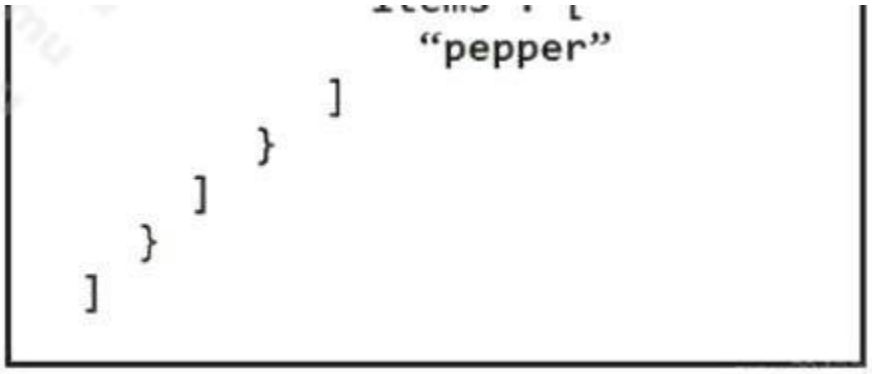
It provides support for HTTP and HTTPS communications. The Administrative XML Layer (AXL) API is Cisco Unified Communications Manager's SOAP-based administrative interface. Client applications invoke AXL operations by sending SOAP requests to the CUCM AXL web service endpoint, so its communication model is based on web protocols. Cisco documents the AXL service URL in the form `https://<CUCM-server>:8443/axl/`, with SOAP requests transported over HTTPS. This allows provisioning and administration applications to communicate with CUCM programmatically while using TLS to protect credentials and configuration data in transit. AXL supports administrative operations such as retrieving, adding, updating, and deleting CUCM configuration objects, but the key capability described here is its use of HTTP-based web-service communication, typically secured with HTTPS. CUCM administrators must enable the AXL Web Service and use an account with the appropriate AXL access permissions before an application can submit requests. Cisco's AXL documentation describes the API as a SOAP interface and provides the service endpoint and client integration requirements. See the Cisco DevNet AXL API documentation and Cisco's CUCM Developer Guide.

QUESTION NO: 14

Refer to the exhibit.



```
[
  {
    "type":"fruit",
    "items": [
      {
        "color":"green",
        "items": [
          "kiwi",
          "grape"
        ]
      },
      {
        "color": "red",
        "items": [
          "strawberry",
          "apple"
        ]
      }
    ]
  },
  {
    "type": "vegs",
    "items": [
      {
        "color": "green",
        "items": [
          "lettuce"
        ]
      },
      {
        "color": "red",
        "items": [
          "pepper"
        ]
```



A REST API returns this JSON output for a GET HTTP request, which has been assigned to a variable called "vegetables". Using Python, which output is the result of this command?

print(filter(lambda 1: 1['type'] == 'fruit', vegetables) [0]['items'][0]['items'][0])

A. {'color': 'green', 'items': ['kiwi', 'grape']}

B. ['kiwi', 'grape']

C. lettuce

D. kiwi

ANSWER: D

Explanation:

`kiwi` is the intended output because the expression traverses the decoded JSON structure from the outer list to a nested string value. The filter condition retains the object whose `type` value is `'fruit'`. Selecting the first matching object accesses that fruit object's `items` list. Its first element is the green-fruit dictionary, which contains an `items` list holding the fruit names. Selecting element zero from that final list returns `kiwi`, and `print()` outputs that string. JSON arrays are represented as Python lists and JSON objects as Python dictionaries after decoding, so chained list indexes and dictionary keys are the appropriate way to reach a deeply nested value. The command's formatting in the prompt reflects OCR substitutions: the lambda parameter is intended to be a valid identifier and the quoted dictionary keys are intended to use normal Python quote characters. In modern Python 3, `filter()` returns an iterator; equivalent indexable code would use `list(filter(...))[0]`. The requested nested value remains `kiwi`. See the Python documentation for filter() and for Python lists and dictionaries.

QUESTION NO: 15

Which two statements about JSON and XML are true? (Choose two.)

A. The syntax of JSON contains tags, elements, and attributes.

B. XML objects are collections of key-value pairs.

C. JSON objects are collections of key-value pairs.

D. JSON arrays are an unordered set of key-value pairs.

E. The syntax of XML contains tags, elements, and attributes.

ANSWER: C E

Explanation:

"JSON objects are collections of key-value pairs." is correct because the JSON standard defines an object as an unordered collection of zero or more name/value pairs. A name is a string, and its associated value may be a string, number, Boolean,

null, array, or another object. This representation makes JSON especially common for API request and response bodies, where named fields describe structured resources.

“The syntax of XML contains tags, elements, and attributes.” is also correct. XML represents structured information as elements identified by markup tags, including start-tags and end-tags (or an empty-element tag). Elements may carry attributes in their start-tags, allowing additional properties or metadata to be associated with the element. Nested XML elements create a hierarchical document structure that applications can parse and validate.

These two data formats are important in Cisco DevNet workflows because network APIs and automation tools frequently exchange structured payloads in JSON or XML. Understanding whether a payload is expressed through JSON name/value pairs or XML elements and attributes helps developers correctly construct, parse, and validate API data. The normative JSON definition is available in RFC 8259, and XML syntax is specified by the W3C XML Recommendation.

QUESTION NO: 16

What are two key capabilities of Cisco Finesse? (Choose two.)

A. Finesse includes an RPC API that enables the development of custom gadgets.

B. Agents access Finesse from a browser without needing to install or configure anything on the client machine.

C. Finesse automatically collects telemetry data

D. An OpenDNS utility is preconfigured and ready to use on Finesse.

E. Gadget containers provide a seamless experience in a single user interface.

ANSWER: B E

Explanation:

Cisco Finesse is a browser-based contact-center desktop designed for agents and supervisors. Its thin-client model lets agents sign in and use the desktop through a supported web browser, without installing a separate Finesse application on each client computer. This architecture centralizes application delivery and maintenance on the Finesse server, helping organizations simplify desktop deployment while providing agents with call-control and contact-center workflow functions in a familiar web interface.

Finesse also uses a gadget-based user-interface framework. Gadget containers allow multiple gadgets to be displayed within one integrated desktop, creating a seamless experience rather than requiring agents to move among separate applications. Organizations can use the standard gadgets and add custom gadgets to surface relevant customer information, workflow tools, or integrated business applications in the same interface. This supports a consolidated agent workflow and is a principal extensibility capability of Finesse.

Cisco documents Finesse as a web-based desktop and describes its gadget framework and development model in the Cisco Finesse Developer Guide. Additional platform documentation is available through Cisco’s Finesse installation and configuration guides.

QUESTION NO: 17

Which two practices are appropriate for unit tests? (Choose two.)

A. Test one small unit of behavior.

B. Require access to a production service.

C. Use mocks or stubs for external dependencies.

D. Execute only after deployment to production.

E. Validate all components as one end-to-end workflow.

ANSWER: A C

Explanation:

Test one small unit of behavior is appropriate because a unit test should focus on a limited, clearly defined behavior of the code under test. Small tests make failures easier to diagnose and allow developers to identify the specific behavior that needs correction.

Use mocks or stubs for external dependencies is also appropriate when a unit interacts with systems such as databases, remote APIs, or filesystems. Replacing those dependencies with controlled test doubles keeps the test isolated, fast, and repeatable. This avoids failures caused by network availability, external service state, or timing. See the Python unittest.mock documentation and the Practical Test Pyramid.

QUESTION NO: 18

Refer to the exhibit.

Which command needs to be placed on the box where the code is missing to output the value of page_jd in the Python 3.7 script?

A)

B)

C)

D)

A. Option A

B. Option B

C. Option C

D. Option D

E. print(page_jd)

ANSWER: E

Explanation:

`print(page_jd)` is the Python 3.7 statement that outputs the current value held by the variable named `page_jd`. Python's built-in `print()` function evaluates the expression supplied as its argument, converts the resulting object to a displayable text representation, and writes that representation to standard output. Therefore, placing `page_jd` inside `print()` directly displays the variable's content when execution reaches that line. This works whether the variable contains a string, number, list, dictionary, or another Python object; Python uses the object's string representation for output. The parentheses are required because, in Python 3, `print` is a function rather than the statement form used by older Python versions. This is appropriate for simple script output and troubleshooting because it lets the developer inspect the value assigned earlier in the program without changing that value. Cisco DevNet candidates should recognize this as standard Python syntax for displaying a variable's contents, independent of how the value was obtained or processed. See the official Python documentation for print() and the Python tutorial's examples of interactive output.

QUESTION NO: 19

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: A C

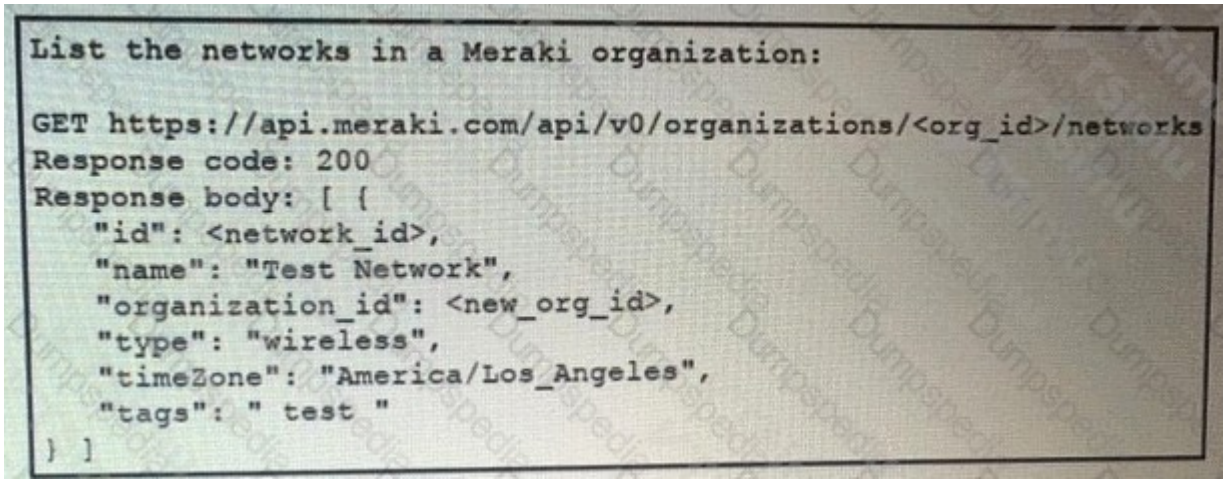
Explanation:

“easier to design, deploy, and manage APIs” is a benefit because a formal data model gives API designers and consumers a shared, machine-readable contract. In Cisco model-driven interfaces, YANG defines configuration and operational data structures, data types, constraints, and relationships. Tooling can use that schema to generate documentation, client code, validations, and consistent API behaviors. This reduces the effort involved in designing an interface and makes deployed automation easier to maintain as platforms and software versions evolve.

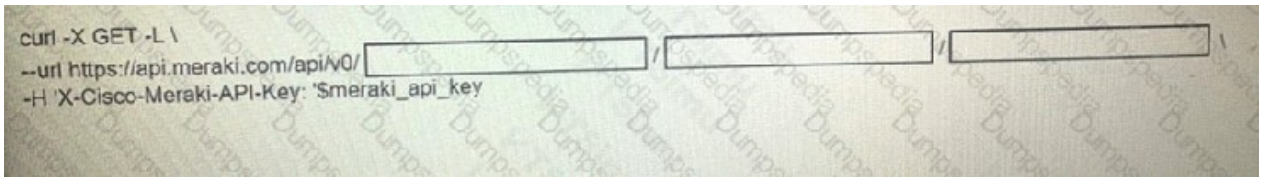
“models decoupled from transport, protocol, and encoding” is also fundamental to model-driven programmability. The data model represents the network data independently from the mechanism used to access or serialize it. A YANG model can therefore be exposed through management protocols such as NETCONF or RESTCONF and encoded in formats such as XML or JSON without redefining the underlying data hierarchy and semantics. This separation promotes reusable automation, consistent tooling, and interoperability across management interfaces. Cisco describes model-driven programmability as using YANG models with protocol and encoding choices separated from the modeled data, while the YANG specification defines the language used to model configuration and state data. See RFC 7950: The YANG 1.1 Data Modeling Language and Cisco Model-Driven Programmability documentation.

QUESTION NO: 20 - (SIMULATION)

Refer to the exhibit.



Fill in the blanks to complete the cURL command to the list of networks in the Meraki organization with an id of 384279060



ANSWER: See the explanation for the answer

Explanation:

Fill the three URL blanks, in order, with:

The completed command is:

This follows the endpoint format shown in the exhibit: `GET /api/v0/organizations/{org_id}/networks`.

QUESTION NO: 21

Which network constraint causes the performance of the application to decrease as the nu m ber of users accessing the application increases?

A. latency

B. loss

C. bandwidth

D. jitter

ANSWER: C

Explanation:

Bandwidth is correct because it represents the capacity of a network path to carry data over a given period, commonly measured in bits per second. As more users access an application, the aggregate amount of traffic generated by requests, responses, downloads, API calls, and other application activity rises. Once that offered traffic approaches or exceeds the available capacity of a link, packets must be queued and transmitted more slowly. This increases application response time and reduces the effective throughput available to each user. Under sustained congestion, queues can fill, which can also lead to dropped packets and retransmissions; these effects further reduce perceived application performance. Therefore, a growth-related slowdown that occurs as the number of concurrent users increases is characteristically a bandwidth-capacity constraint. Application developers and network engineers should account for expected concurrent demand, payload sizes, and traffic patterns when designing and testing services, then provision or scale network capacity accordingly. Cisco describes bandwidth as the amount of data that can be transmitted over a connection in a specified time, while throughput is the actual achieved transfer rate. See Cisco's bandwidth overview and its network bandwidth guidance.

QUESTION NO: 22

Which two statements describe the purpose of a continuous integration process? (Choose two.)

A. It integrates changes into a shared repository frequently.

B. It runs automated builds and tests for integrated changes.

C. It requires every successful build to be deployed to production.

D. It eliminates the need for version control.

E. It prevents developers from working on branches.

ANSWER: A B

Explanation:

Continuous integration requires developers to integrate changes into a shared codebase frequently. Each integration can trigger an automated build and test workflow, providing prompt feedback when a change introduces a defect, dependency issue, or integration conflict. Running automated builds and tests on each change helps identify issues before they reach later delivery stages.

Continuous integration does not require automatic production deployment. Continuous delivery and continuous deployment address the later stages of making validated software available for release or deploying it automatically. The Martin Fowler continuous integration article describes maintaining a shared repository and verifying changes through automated builds and tests.

QUESTION NO: 23

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:

An application that accepts webhook notifications is accurately characterized as a *receiver* and a *listener*. A webhook is an event-driven HTTP callback: when an event occurs in the source service, that service initiates an HTTP request, typically a POST request containing an event payload, to a URL exposed by the destination application. The destination therefore receives the notification and listens on its registered HTTP endpoint for these incoming event requests. In practice, the receiver/listener verifies the request as appropriate, parses the payload, and starts the relevant processing, such as updating application state or invoking a workflow. This push-based interaction lets the destination react when an event occurs rather than repeatedly polling the source service for changes. Cisco DevNet material uses webhook concepts in the broader context of event notifications and REST-based integrations, where an application provides a callback endpoint to accept notifications. GitHub's webhook documentation likewise describes payload delivery to a configured payload URL, which is the endpoint implemented by the receiving/listening application. See GitHub Docs: About webhooks and Cisco DevNet Learning.

QUESTION NO: 24

Which type of HTTP method is used by the Meraki nad webex teams APIs to send webhook notifications?

A. HTTP GET

B. HTTP PUT

C. HTTP HEAD

D. HTTP POST

ANSWER: D

Explanation:

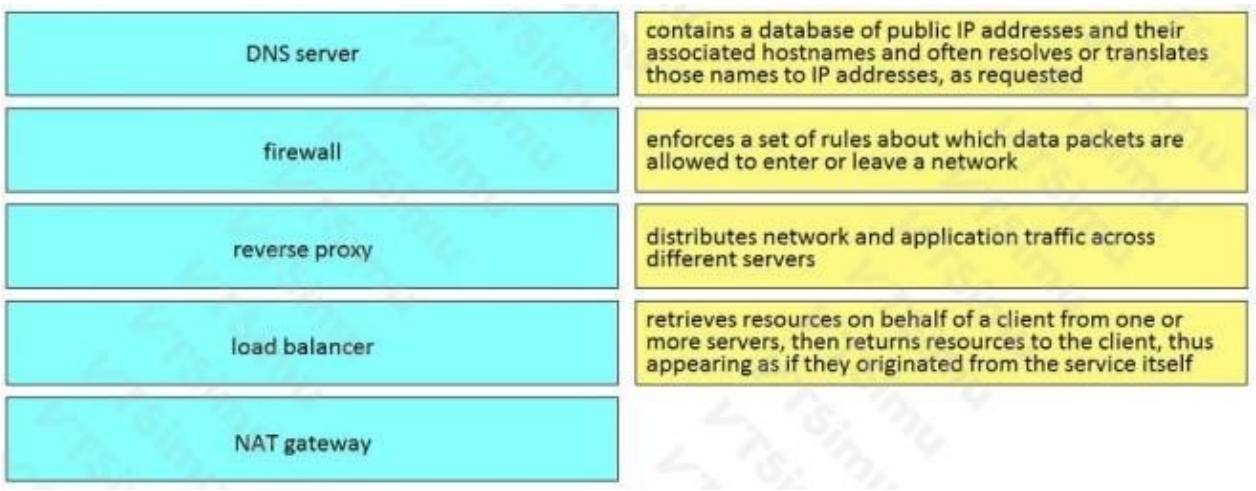
HTTP POST is used to deliver webhook notifications from both Cisco Meraki and Webex. A webhook is an event-driven HTTP callback: after a subscriber configures a destination URL, the service sends an HTTP request to that URL whenever the relevant event occurs. The notification payload, normally formatted as JSON, is carried in the body of the POST request so that the receiving application can parse the event details and take appropriate action. Meraki webhook alert delivery uses HTTP POST requests to the configured receiver URL. Likewise, Webex webhook notifications are delivered as HTTP POST requests; the initial payload identifies the event and resource, and an integration can then retrieve full resource details through the Webex API when needed. Therefore, an endpoint intended to receive notifications from either platform must expose a route that accepts POST requests and processes the incoming request body. Cisco's Meraki documentation describes webhook receiver requirements and payload delivery, while the Webex developer documentation specifies POST delivery for webhook events. See the Cisco Meraki Webhooks documentation and the Webex Webhooks guide.

QUESTION NO: 25 - (DRAG DROP)

DRAG DROP

Drag and drop the network component names from the left onto the correct descriptions on the right. Not all options are used.

Select and Place:



ANSWER:

Explanation:

The completed mapping is correct because each selected component corresponds to its standard role in a networked application architecture. A **DNS server** maintains or accesses records that associate names with IP addresses, enabling a client to request a hostname and receive the appropriate IP address for the destination. This name-resolution process lets users and applications connect by stable, human-readable service names rather than needing to know numeric addresses directly. Cisco describes DNS as the service used to translate hostnames into IP addresses in its DNS security overview.

A **firewall** is the traffic-policy enforcement boundary. It applies configured security rules to determine whether traffic is permitted to enter or leave a protected network or workload. This function is fundamental to controlling access between trusted and untrusted zones and is accurately represented by the packet-rule description. A **load balancer** receives traffic intended for an application and distributes that traffic across multiple available servers. This improves capacity, resiliency, and service continuity by preventing a single server from handling every request. Cisco outlines this traffic-distribution role in its load balancing overview.

A **reverse proxy** accepts requests on behalf of one or more backend services, obtains the requested resource from the appropriate origin server, and returns the response to the client. The client interacts with the reverse proxy endpoint rather than directly with the backend server, which is precisely the behavior described. The unused NAT gateway is appropriate because none of the displayed descriptions concern translating private addresses for outbound or inbound connectivity.

QUESTION NO: 26

What is a benefit of test-driven development?

A. early customer involvement

B. increased code quality

C. faster releases that have minimal features

D. strict adherence to product requirements

ANSWER: B

Explanation:

Increased code quality is a central benefit of test-driven development (TDD). TDD uses a short, repeated cycle: first write an automated test that describes the expected behavior, then implement only enough production code for the test to pass, and finally refactor the code while continuously rerunning the tests. Defining behavior before implementation encourages developers to think through interfaces, inputs, outputs, and edge cases early. It also promotes code that is modular and easier to test, because the design must support automated verification from the outset.

The resulting test suite provides rapid feedback during development and a regression safety net as the application evolves. When a change unintentionally alters existing behavior, an automated test can expose the problem promptly, allowing it to be corrected before release. Regular refactoring with passing tests further helps keep implementation understandable and maintainable. These practices support reliability, maintainability, and defect prevention, which are important dimensions of code quality. Cisco DevNet candidates should recognize TDD as an automated-testing practice that improves confidence in code changes throughout the software lifecycle. See Martin Fowler's overview of Test-Driven Development and the Agile Alliance TDD glossary.

QUESTION NO: 27

A new application is being developed that must be hardware independent. The application includes an administrative component which is accessed using a Windows desktop GUI. Installation and management of the environment must be fully automated. Which application deployment type meets the requirements?

A. bare metal

B. virtual Python environment

C. container

D. virtual machine

ANSWER: C

Explanation:

container is the best deployment type because a container packages an application with its required runtime, libraries, and dependencies into a consistent, portable image. This separates the application from specific physical hardware and helps it run consistently wherever a compatible container runtime is available. The Windows desktop GUI requirement does not require the entire application to be installed directly on each administrator's desktop: the GUI can act as a Windows client that connects to containerized application services through an API or web-based interface.

Containers also suit fully automated installation and operations. A container image can be built reproducibly from a Dockerfile as part of a continuous integration pipeline, stored in an image registry, and deployed automatically. Orchestration platforms such as Kubernetes support declarative management of the desired application state, including automated rollout, restart after failures, scaling, and updates. This makes container-based deployment a strong fit for an environment intended to be provisioned and managed through automation rather than manual host-by-host installation.

Docker describes containers as isolated processes containing everything needed to run an application in its container overview. Kubernetes documents declarative workload management and automated deployment capabilities in its Deployment documentation.

QUESTION NO: 28

Where is an IP packet routed if the packet does not match any routes in the routing table?

A. firewall

B. load balancer

C. central switch

D. default gateway

ANSWER: D

Explanation:

The correct answer is **default gateway**. When a router receives an IP packet, it compares the destination address to entries in its routing table and selects the most-specific matching route. If no specific route matches, the router uses its default route, also called the gateway of last resort. This route is commonly represented by `0.0.0.0/0` in IPv4 or `::/0` in IPv6 and identifies the next-hop router or outgoing interface to which otherwise unmatched traffic should be forwarded. In practical network designs, that next-hop device is commonly described as the default gateway because it provides the path out of the local routing domain toward other networks. The default route must be configured or learned by the router; if it is present, it provides a catch-all match for destinations that lack a more-specific route. Cisco describes a default route as the route used when no other route in the routing table matches the destination address. See Cisco's Understanding and Configuring Default Routes and the Cisco IOS XE route concepts documentation.

QUESTION NO: 29

What is a capability of the AXL API?

A. It adds a user to a collaboration space to share information and files.

B. It executes SQL commands in Cisco Unified Communications Manager.

C. It allows a meeting to be created with users that do not belong to same organization.

D. It collects information about system, cluster, and database settings.

ANSWER: B

Explanation:

It executes SQL commands in Cisco Unified Communications Manager. AXL, or Administrative XML, is the SOAP-based administrative API for Cisco Unified Communications Manager. In addition to providing structured operations for provisioning and managing CUCM configuration objects, AXL exposes the `executeSQLQuery` and `executeSQLUpdate` operations. These operations enable an authorized API client to submit supported SQL queries and updates to CUCM's Informix database. This capability is useful when an application needs to retrieve configuration or inventory data that is not conveniently available through a specific AXL object operation, such as information associated with devices, directory numbers, users, or other call-control configuration records. Cisco documents these SQL methods directly in the AXL API schema and developer documentation, including their required authentication, permissions, and operational constraints. Because SQL execution is an explicit AXL web-service operation, executing SQL commands against CUCM through the supported AXL interface is a valid AXL capability. For Cisco's AXL API documentation, see Cisco AXL API Documentation and the Cisco AXL Developer Guide.

QUESTION NO: 30

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 built on a cultural and operational shift in which people responsible for building software and people responsible for operating it collaborate throughout the delivery lifecycle. Therefore, *organizing cross-functional teams over organizational silos* is foundational: shared ownership, direct communication, and rapid feedback reduce handoffs and align work with reliable delivery of customer value. DevOps is not merely a collection of tools; its culture depends on cooperation among development, operations, quality, security, and other stakeholders.

Automating over documenting is also a foundational DevOps principle because automation makes recurring delivery and operational work repeatable, testable, consistent, and fast. Automated builds, tests, deployments, monitoring responses, and infrastructure provisioning enable continuous integration and delivery while reducing avoidable manual variation. This wording does not mean documentation has no value; instead, DevOps favors executable, version-controlled automation for processes that must be performed reliably and frequently. Culture and automation are central pillars of the widely used CALMS model for DevOps, and both support faster feedback and dependable releases. Cisco describes DevOps as bringing development and operations together to improve software delivery, while Microsoft similarly emphasizes collaboration and automation. See Cisco DevOps solutions and Microsoft Learn: What is DevOps?.

QUESTION NO: 31

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

A. MAC address

B. metrics

C. administrative distance

D. IP address

E. subnet mask

ANSWER: B C

Explanation:

metrics and **administrative distance** are the key route-selection considerations when a Cisco device has competing routes to a destination. Administrative distance expresses the trustworthiness of the source that supplied a route. For example, a route learned through one routing source can be preferred over a route to the same prefix learned through another source when its administrative distance is lower. Cisco devices use this value to decide which route source is more credible before comparing route costs from different sources.

A metric is the numerical path cost calculated according to the rules of the routing protocol that installed the route. Its inputs vary by protocol: hop count is used by RIP, while other protocols can incorporate characteristics such as bandwidth and delay. When routes to the same destination originate from the same routing protocol, the route with the preferred metric is selected according to that protocol's calculation. Therefore, administrative distance resolves preference between route

sources, and metrics resolve path preference among routes supplied by the same source. Cisco documents these concepts in its Administrative Distance documentation and IP Routing Route Concepts.

QUESTION NO: 32

Which network device plane is responsible for handling QoS?

A. management plane

B. data plane

C. configuration plane

D. control plane

ANSWER: B

Explanation:

The **data plane** is responsible for handling QoS because QoS policy must be applied to traffic as packets enter, traverse, and leave a network device. This is the packet-forwarding path, where the device performs operations such as traffic classification, marking, policing, shaping, queueing, scheduling, and congestion avoidance. These actions determine the treatment each packet receives, including whether it is prioritized, delayed, remarked, limited, or dropped during congestion.

QoS must operate at forwarding speed, often in dedicated hardware or ASIC-based forwarding pipelines on Cisco platforms. For example, an interface service policy can classify packets into traffic classes and apply bandwidth, priority, policing, or queue-management behavior while those packets are forwarded. The data plane therefore enforces the configured QoS policy on individual packets and queues. Cisco describes QoS as a collection of mechanisms for managing network traffic and controlling how packets are treated, including classification, marking, policing, shaping, and congestion management. See the Cisco QoS Tutorial and the Cisco IOS XE QoS Concepts Overview.

QUESTION NO: 33 - (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:

The correct command is `git push`. After the engineer changes the Python script locally, the updated work must be committed and sent to the GitLab repository. Running `git push` publishes the local commit to the remote branch on GitLab. That remote push is an event GitLab CI/CD can detect, and it creates a pipeline for the commit when the project includes a valid `.gitlab-ci.yml` configuration file.

The pipeline executes on GitLab infrastructure through configured runners rather than on the engineer's laptop. Its jobs follow the stages and rules defined in the CI/CD configuration, such as running unit tests, style checks, security scans, build steps, or deployment validation. This gives the engineer a repeatable automated validation process tied to the exact revision that was uploaded to the shared repository. The pipeline results, job logs, and test status are then available in the GitLab project interface and can be used to confirm whether the change is ready to merge.

In a normal Git workflow, committing records the change in the local repository, while pushing transfers that commit to GitLab. The push therefore provides the server-side repository event needed to start the configured automated workflow. GitLab documents that pipelines are commonly created for pushes, and its CI/CD pipeline documentation explains how jobs are defined and executed from the project configuration. See GitLab CI/CD pipelines and GitLab pipeline settings and pipeline creation.

QUESTION NO: 34

A 401 HTTP response code is returned when calling a REST API. What is the error state identified by this response code?

A. The server cannot process the request as it has detected an issue in the request syntax or body.

B. The server accepted the request but the client is not authorized for this content.

C. The request has not been accepted because it requires authentication.

D. The server cannot find the requested resource because the path specified is incorrect.

ANSWER: C

Explanation:

The request has not been accepted because it requires authentication. HTTP status 401 means that the target resource requires valid authentication credentials and the request either did not include them or included credentials that could not be accepted. In an API workflow, the client should authenticate using the mechanism required by the service, such as an access token, API credential, or another HTTP authentication scheme, then send the request again with the appropriate authentication information. A 401 response commonly includes a `WWW-Authenticate` response header describing the authentication scheme that the server expects. This status therefore identifies an authentication requirement or authentication failure at the time the request is made. The HTTP Semantics specification defines 401 as indicating that the request has not been applied because it lacks valid authentication credentials for the target resource. Cisco DevNet developers should use this response as a signal to inspect token validity, expiration, formatting, and the required authentication header for the API endpoint. See the RFC 9110 definition of 401 Unauthorized and the Cisco DevNet REST API authentication learning material.

QUESTION NO: 35

A new application is being developed that requires the ability to be copied and moved from one location to another. The existing infrastructure is already heavily utilized, so the new a p plication must have a low resource footprint. The application includes a small PostgreSQL d a tabase component. Which application deployment type meets the requirements?

A. Virtual machine

B. Python virtual environment

C. Container

D. Bare metal

ANSWER: C

Explanation:

Container is the appropriate deployment type because containers package an application together with its required libraries, dependencies, and runtime configuration into a portable image. That image can be copied, versioned, and deployed consistently across a developer workstation, test environment, or production host. This portability directly supports the requirement to move the application between locations without rebuilding the environment each time.

Containers also share the host operating system kernel rather than running a complete guest operating system for every application instance. As a result, they generally consume substantially less CPU, memory, and storage than full virtual machines, which is especially valuable when the existing infrastructure is already heavily utilized. The application and its small PostgreSQL component can be deployed as separate coordinated containers, allowing each component to be updated, scaled, and configured independently while remaining lightweight. Persistent storage can be attached to the PostgreSQL container so that database data remains available when the container itself is recreated or moved.

Docker describes containers as isolated processes that include everything needed to run an application and can run consistently in different environments. PostgreSQL also provides official container images for this deployment model. See Docker: What is a container? and Docker Hub: PostgreSQL Official Image.

QUESTION NO: 36

Which IP service is used to monitor the performance of network devices?

A. SNMP

B. DHCP

C. DNS

D. NTP

ANSWER: A

Explanation:

SNMP is the IP service used to monitor the performance and operational health of network devices. It defines a manager-and-agent model: monitoring software acts as an SNMP manager, while routers, switches, firewalls, and other managed devices run SNMP agents that expose operational data. The manager can poll an agent for values held in Management Information Bases (MIBs), including interface traffic counters, packet errors, device uptime, CPU load, memory utilization, and hardware or environmental status. Monitoring systems use these values to establish baselines, graph trends, identify capacity issues, and alert operators when performance crosses defined thresholds. SNMP also supports unsolicited notifications, called traps and informs, allowing a device to report significant events such as an interface state change without waiting for a poll. Cisco platforms support SNMP for network-management integration, and Cisco recommends SNMPv3 where possible because it provides authentication and privacy features for management traffic. The SNMP architecture and its standardized monitoring role are defined by the IETF SNMP Management Framework. See Cisco's SNMP documentation and RFC 3411: An Architecture for Describing SNMP Management Frameworks.

QUESTION NO: 37

What are two use cause where webhooks are effective? (Choose two.)

A. Close a session with a web server after a specific amount of time.

B. Filter out information from a response to an API call

C. Change the response format or content type of an API call.

D. Inform a previously defined chat channel after a deployment completes.

E. Send an email to a customer of an online store after payment is complete.

ANSWER: D E

Explanation:

Webhooks are effective for event-driven integrations in which one application must notify another application promptly after a defined event occurs. A webhook producer sends an outbound HTTP request to a preconfigured endpoint when the event happens, allowing the receiving service to start its workflow without continuously polling for status changes. Informing a previously defined chat channel after a deployment completes is a common delivery-pipeline use case. A CI/CD platform can emit a deployment event, and the webhook receiver can publish an immediate success or failure notification to the appropriate collaboration channel. This gives engineering teams timely operational visibility and supports automated incident or release workflows.

Sending an email to a customer of an online store after payment is complete is likewise an event-based workflow. When a payment platform confirms the transaction, it can deliver a webhook event containing the payment status and relevant transaction identifiers. The store's backend can validate the request, record the order state, and initiate its confirmation-email process. This asynchronous notification model is especially useful because payment completion may occur after the original browser interaction and must be handled reliably by backend systems. GitHub describes webhooks as event-triggered HTTP notifications in its webhook documentation. Stripe similarly documents payment-event delivery and secure handling in its webhook guide.

QUESTION NO: 38

```
api_endpoint = f"https://veryusefulapi/v2/authenticate"

headers = {
    "Authorization": "Basic YWRtaW46c3VwZXJzZWNyZXQ="
    "Accept": "application/json"
}

req = requests.get(api_endpoint, headers=headers)
```

Refer to the exhibit. A developer is part of a team that is working on an open-source project in which source code is hosted in a public GitHub repository. While the application was built, s e curity concerns were addressed by encrypting the credentials on the server. After a few months, the developer realized that a hacker managed to gain access to the account. The e x hibit contains part of the source code for the login process. Why was the attacker able to a c cess the developer ' s account?

A. The encoded credentials were available in the source code.

B. The application was not encrypting the communication with the server.

C. The credentials were encrypted in the source code.

D. An SSL certificate was used instead of the TLS protocol to authenticate.

ANSWER: A

Explanation:

The encoded credentials were available in the source code. In a public GitHub repository, any person who can view the repository can retrieve hardcoded credential values and inspect the application logic that uses them. Encoding a value with Base64 changes its representation but does not provide cryptographic protection: the original username and password can be recovered trivially with standard tools. Therefore, placing Base64-encoded credentials directly in publicly accessible source code effectively exposes the secret and enables an attacker to authenticate as the developer or application account. Credentials must be treated as secrets throughout the development lifecycle, including when code is open source. A safer design removes secrets from the repository and injects them at runtime through protected environment variables, a secrets-management service, or a platform-managed identity mechanism. Secret values should also be rotated after exposure because repository history, forks, clones, build logs, and cached artifacts may retain the disclosed value even after the visible source is changed. OWASP identifies hardcoded credentials as a security weakness and recommends externalized secret management: OWASP Broken Authentication. GitHub also provides guidance for removing and rotating exposed repository secrets: GitHub Docs: Remediating a leaked secret.

QUESTION NO: 39

What are two advantages of YANG-based approaches for infrastructure automation? (Choose two.)

A. multi-platform vendor abstraction

B. compiles to executables that run on network devices

C. designed to reflect networking concepts

D. directly maps to JavaScript

E. command line driven interface

ANSWER: A C

Explanation:

YANG-based automation provides **multi-platform vendor abstraction** because YANG defines structured data models independently of a device's command-line syntax. Automation tools can use standardized models, such as IETF YANG modules, across platforms that support the same model. Vendors can also provide native models with a consistent schema for their operating systems. This lets developers work with predictable configuration and operational-state data through protocols such as NETCONF and RESTCONF, reducing dependence on device-specific command templates and text parsing. RFC 7950 defines YANG as a language for modeling configuration data, state data, remote procedure calls, and notifications; these machine-readable models can be used by management applications to interact programmatically with network devices.

YANG is also **designed to reflect networking concepts**. Its hierarchical tree structure naturally models familiar network constructs, including interfaces, routing protocols, access-control lists, and services. Model definitions can include data types, mandatory values, ranges, patterns, relationships, reusable groupings, and validation constraints. As a result, an automation client can construct and validate configuration data according to the network feature's intended structure before sending it to a device. This improves consistency and supports safer, more reliable infrastructure automation. See RFC 7950: The YANG 1.1 Data Modeling Language and Cisco IOS XE Model-Driven Programmability.

QUESTION NO: 40

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: B D

Explanation:

UCS Director API interactions can be XML- or JSON-encoded and require an API key in the HTTP header for authentication. UCS Director exposes REST APIs that support both XML and JSON representations, allowing a client to select the representation appropriate to its integration. Its API authentication model uses an API access key generated for an authenticated user; clients present that key in the HTTP request header when calling the API. This supports automation without requiring an interactive UI session for every request.

UCS Manager API interactions are XML-encoded and require a cookie in the method for authentication. The UCS Manager native management API is XML-based: API requests contain XML method elements, and the XML API login operation returns a session cookie. Subsequent XML API method calls include that cookie value so UCS Manager can associate the request with the authenticated session. This cookie-based session mechanism is fundamental to UCS Manager XML API clients and applies to programmatic operations such as querying managed objects and making configuration changes.

See the Cisco UCS Manager XML API Programmer's Guide and the Cisco UCS Director API User Guide.

QUESTION NO: 41

Which tool is used to block all traffic to the domain by using a single API call?

A. Cisco ISE

B. Cisco Firepower

C. Cisco AMP

D. Cisco Umbrella

ANSWER: D

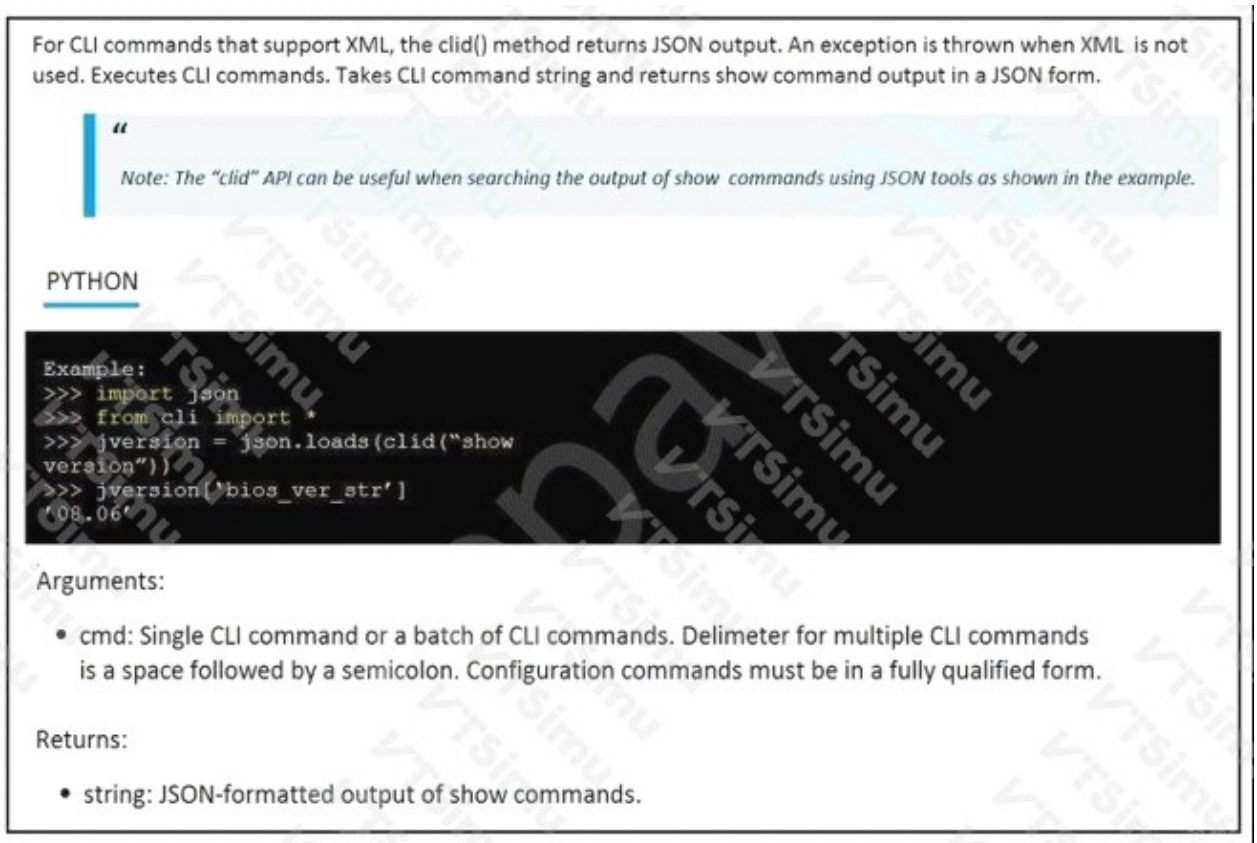
Explanation:

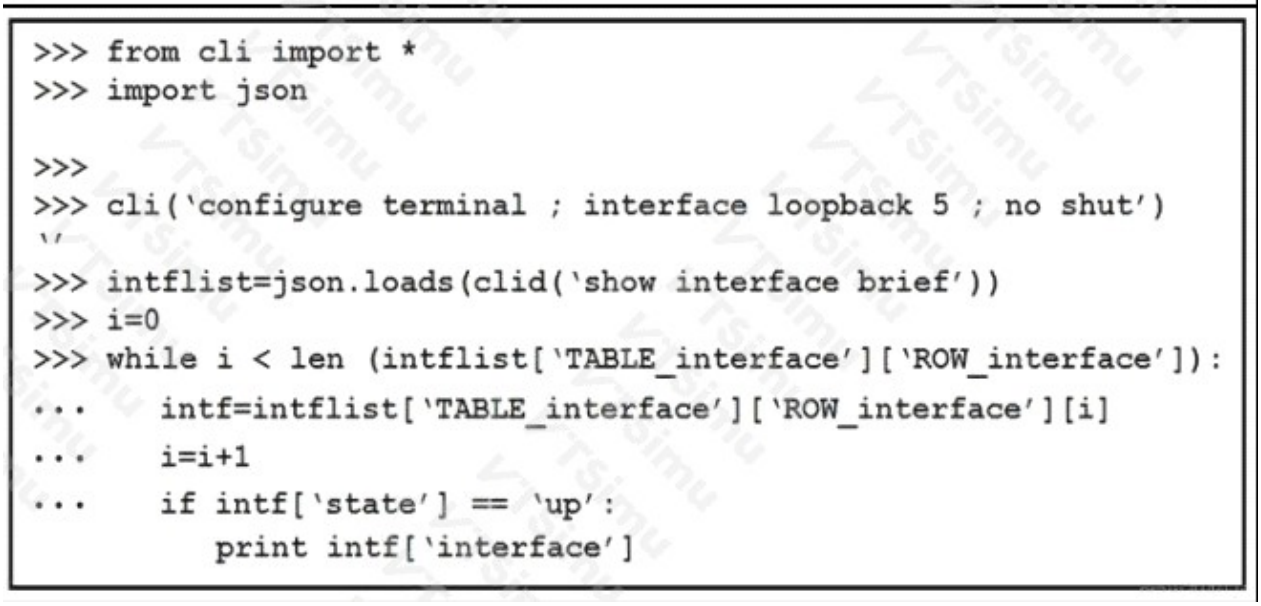
Cisco Umbrella is the appropriate tool because it provides cloud-delivered DNS-layer security and APIs that support automated enforcement against malicious or unwanted internet destinations. An automation workflow can submit a domain to an Umbrella destination list through the Umbrella APIs, then associate that list with a DNS policy. Once the policy is applied, DNS requests for that domain are blocked for identities, networks, or roaming endpoints covered by the policy. Because clients cannot resolve the prohibited domain through Umbrella's protected DNS service, this prevents access before a connection to the destination is established.

This is especially useful for incident-response automation. For example, after a threat-intelligence system identifies a phishing or command-and-control domain, a script or SOAR platform can programmatically add it to a block list without manually updating individual endpoint configurations. Umbrella centrally distributes and enforces the resulting DNS policy for protected users and networks, including roaming users using the Umbrella roaming client. Cisco documents destination-list APIs and policy configuration as part of its Umbrella management APIs. See the Cisco Cloud Security API documentation and Cisco's destination list documentation.

QUESTION NO: 42

Refer to the exhibits.





The Python interpreter and the Cisco Python SDK are available by default in the Cisco NX-OS Software. The SDK documentation shows how the clid() API can be used when working with JSON and XML.

What are two effects of running the script? (Choose two.)

A. configure interface loopback 5

B. show details for the TABLE interface

C. issue shutdown on interface loopback 5

D. show only the interfaces in the up status

E. show only the interfaces in admin shut status

ANSWER: A D

Explanation:

The script configures Loopback 5 because it sends the NX-OS configuration commands needed to enter the loopback interface context. Commands supplied through the NX-OS Python SDK can be passed to the CLI execution API, enabling Python code to apply the same configuration that an administrator would enter interactively. Consequently, the interface configuration context for Loopback 5 is reached when the script runs.

The script also shows only the interfaces in the up status. The `clid()` API obtains structured CLI output, and the JSON/XML hierarchy contains interface rows beneath `TABLE_interface`. The script iterates through those interface records and evaluates the relevant operational-state field. It prints or otherwise retains records whose state is `up`, so the resulting displayed interface set is filtered rather than being the complete unfiltered command output. This combination of structured command retrieval and Python-based filtering is a standard NX-OS programmability workflow. Cisco documents the Python CLI APIs and their use for executing CLI commands in the Cisco NX-OS Python API documentation. Additional NX-OS programmability resources are available through Cisco DevNet NX-OS.

QUESTION NO: 43

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. Bearer-Token: dXNlcm5hbWU6cGFzc3dvcmQ=

B. Content-Type: application/xml

C. Authentication: Bearer dXNlcm5hbWU6cGFzc3dvcmQ=

D. Accept: application/xml

E. Authorization: Bearer dXNlcm5hbWU6cGFzc3dvcmQ=

ANSWER: D E

Explanation:

A `401 Unauthorized` response means that the request has not supplied valid credentials for the protected API resource. `Authorization: Bearer dXNlcm5hbWU6cGFzc3dvcmQ=` provides credentials in the standard HTTP Authorization header using the Bearer authentication scheme. Bearer-token usage is defined by RFC 6750: the access token is carried in the `Authorization` request header as `Bearer <token>`. The API can then validate the token and authorize access to the inventory endpoint.

`Accept: application/xml` expresses the representation that the client wants the server to return. The Accept header is HTTP content negotiation: it identifies media types that the client can process, and `application/xml` requests an XML response. This is particularly relevant for a retrieval request because the desired response format is XML. Together, these headers communicate the required bearer credentials and the requested XML representation, allowing an authenticated inventory request to be processed with an XML response when that media type is supported by the API. See RFC 6750: OAuth 2.0 Bearer Token Usage and the MDN Accept header reference.

QUESTION NO: 44

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

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

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

C. It provides wiki collaboration software for documentation.

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

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

ANSWER: D E

Explanation:

Version control software supports comparisons between revisions of source code files by recording changes as a project evolves. Developers can inspect the differences between revisions, identify when a change was introduced, review its context, and restore an earlier version when necessary. This revision history makes collaborative development more reliable and gives teams an auditable record of source changes. Git documentation describes how commits record project history and how comparison tools such as `git diff` show changes between commits, branches, or files.

Version control also allows new team members to access the current code and history. A developer can clone a repository to obtain a working copy of the current project and its recorded commit history, allowing that person to understand previous implementation decisions and begin contributing using the same shared source of truth as the rest of the team. This improves onboarding, collaboration, and continuity when team membership changes. See the Pro Git overview of Git and the Git documentation for git diff.

QUESTION NO: 45

Several teams at a company are developing a new CRM solution to track customer interactions with a goal of improving customer satisfaction and driving higher revenue. The proposed solution contains these components:

MySQL database that stores data about customers HTML5 and JavaScript UI that runs on Apache REST API written in Python

What are two advantages of applying the MVC design pattern to the development of the solution? (Choose two.)

A. to enable multiple views of the same data to be presented to different groups of users

B. to provide separation between the view and the model by ensuring that all logic is separated out into the controller

C. to ensure data consistency, which requires that changes to the view are also made to the model

D. to ensure that only one instance of the data model can be created

E. to provide only a single view of the data to ensure consistency

ANSWER: A B

Explanation:

MVC divides an application into model, view, and controller responsibilities. In this CRM, the model represents customer and interaction data plus the associated application behavior, while the HTML5/JavaScript interface is a view that renders that information for users. MVC permits multiple views to use the same underlying model, allowing sales representatives, support personnel, and managers to receive interfaces or dashboards tailored to their needs without duplicating the customer data or core application behavior. This is particularly valuable when the organization later adds another consumer, such as a mobile interface or reporting dashboard.

MVC also establishes a separation between the view and model. The controller manages user input and application flow, coordinating requests from a view with operations on the model. That separation reduces direct dependencies between presentation code and data/application code, so teams responsible for the Python REST API and database can evolve their work with less impact on the team building the browser UI. It also makes the components easier to test and maintain independently. MDN describes MVC as separating an application's input, processing, and output concerns, while Oracle's Java EE tutorial explains the controller's role in handling requests and coordinating views and models. See MDN Web Docs: MVC and Oracle Java EE Tutorial: MVC Architecture.

QUESTION NO: 46

Which two statements describe YANG data models? (Choose two.)

A. They define configuration and state data.

B. They replace all network transport protocols.

C. They can define RPC operations and notifications.

D. They require CLI commands as their only encoding.

E. They are used only for IPv6 routing tables.

ANSWER: A C

Explanation:

They define configuration and state data because YANG is a data modeling language used to describe the hierarchy, names, types, constraints, and relationships of managed data. A YANG module can identify which data nodes are configuration data and which report operational state.

They can define RPC operations and notifications because YANG supports constructs for remote procedure calls and event notifications in addition to data-tree definitions. NETCONF and RESTCONF implementations use YANG models to provide structured interfaces for configuration, state retrieval, operations, and event handling. See RFC 7950: The YANG 1.1 Data Modeling Language.

QUESTION NO: 47

What is the outcome of executing this command?

git clone ssh://john@exmaple.com/path/to/my-project_git

A. Creates a local copy of a repository called â€œmy projectâ€□

B. Creates a copy of a branch called â€œmy projectâ€□

C. Initiates a new Git repository called â€œmy projectâ€□

D. Creates a new branch called â€œmy projectâ€□

E. Creates a local clone of the remote repository in a directory called “my-project_git”

ANSWER: E

Explanation:

The command `git clone ssh://john@exmaple.com/path/to/my-project_git` attempts to create a local clone of the remote Git repository identified by the SSH URL. Git uses SSH to connect to the host as user `john`, then downloads the repository’s history, references, and working files into a newly created local directory. When no destination directory is supplied, Git derives that directory name from the final component of the repository path. Here, the final component is `my-project_git`, so that is the local directory name Git will use. Git removes a trailing `.git` suffix when present, but `_git` is not that suffix and remains part of the name. The operation will succeed only if the hostname is reachable and the specified SSH account has access to that repository; otherwise Git reports a connection, authentication, or repository-access error. The hostname is spelled `exmaple.com` in the command, so it must be intentional and resolvable for the clone to complete. See the Git clone documentation and Git’s SSH protocol documentation.

QUESTION NO: 48

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

A.

B.

C.

D.

E.

ANSWER: A B

Explanation:

The NETCONF operations that cover RESTCONF GET read behavior are `<get>` and `<get-config>`. RESTCONF uses HTTP GET to retrieve a representation of a YANG-modeled resource. Depending on the target resource and query parameters, that representation can include configuration data, operational state data, or both. NETCONF provides these read capabilities through its RPC operations.

`<get>` retrieves both configuration and state data from the device’s running datastore, making it the appropriate NETCONF operation when a RESTCONF GET request needs operational information or a combined data view. `<get-config>` retrieves configuration data from a specified configuration datastore, such as `running`, `candidate`, or `startup`. It therefore corresponds to RESTCONF GET requests focused on configuration data. Both operations also support subtree filtering, allowing a client to request only the relevant YANG-modeled data rather than an entire datastore.

This relationship reflects the difference in protocol style: RESTCONF exposes data through HTTP methods and resource URIs, while NETCONF exposes comparable datastore functions through XML-encoded RPC operations. See the NETCONF operation definitions in RFC 6241, section 7 and RESTCONF data retrieval semantics in RFC 8040, section 4.3.

QUESTION NO: 49

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 E

Explanation:

A routing table provides the Layer 3 forwarding information a router or multilayer switch needs to deliver packets toward destination IP networks. It lists routes to a particular destination as IP prefixes and associates the selected route with forwarding information, such as a next-hop address, exit interface, or both. When traffic arrives, the device performs a longest-prefix-match lookup against these destination prefixes to identify the most specific available route and forward the packet accordingly.

A routing table also contains static and dynamic entries. Administrators configure static routes directly, while dynamic routing protocols install routes learned from neighboring devices. Cisco IOS identifies route sources with route codes, including connected routes, static routes, and routes learned through protocols such as OSPF, EIGRP, RIP, and BGP. If several candidate routes exist for the same destination, the device uses route-selection attributes such as administrative distance and metrics to choose the route installed for forwarding. Thus, both manually defined and protocol-learned reachability information can coexist in the table and support packet-forwarding decisions. Cisco describes route-table fields and route codes in its Understanding and Interpreting the Routing Table documentation; its Cisco IOS IP Routing documentation further describes route selection and forwarding behavior.

QUESTION NO: 50

Which two descriptions can be given to an application that is interacting with a webhook? (Choose two.)

A. Processor

B. Codec

C. Listener

D. Receiver

E. Transaction monitor

ANSWER: C D

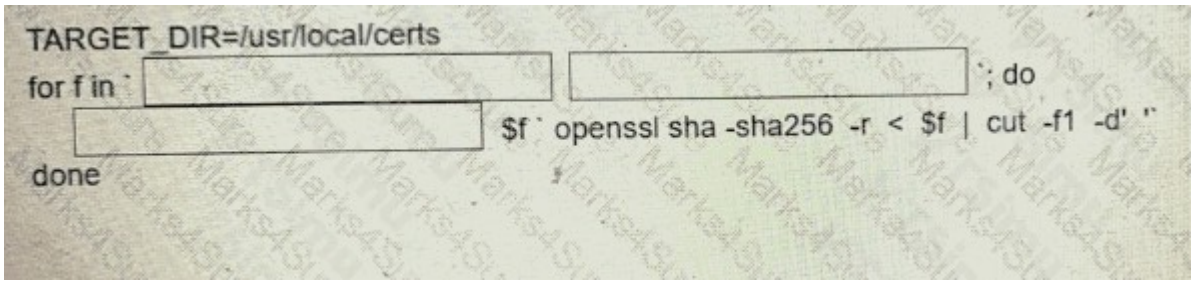
Explanation:

A webhook is an event-driven HTTP callback: when an event occurs, the service that produces the event sends an HTTP request, commonly a POST request, to a preconfigured URL. The application exposed at that destination URL is appropriately called a **Listener** because it waits for and accepts incoming webhook notifications. It is also appropriately called a **Receiver** because it receives the request body, headers, and event data sent by the webhook provider.

These names describe complementary aspects of the same webhook-consuming application. The listener role emphasizes that the application runs an endpoint that is available to accept asynchronous inbound requests. The receiver role emphasizes that the endpoint obtains and processes the event notification payload. In practical implementations, this application validates the request as appropriate, acknowledges successful delivery with a suitable HTTP response, and then processes the received event data. Cisco Webex documentation describes webhooks as a mechanism for receiving real-time notifications at a specified target URL, while Cisco Meraki documentation similarly describes HTTP POST delivery to a receiver URL. See Cisco Webex Webhooks guide and Cisco Meraki Webhooks documentation.

QUESTION NO: 51 - (SIMULATION)

Fill in the blanks to complete the Bash script in which each file in a directory is renamed to Its SHA256 hash?



ANSWER: See the explanation for the answer

Explanation:

Fill the blanks with:

First blank: `$TARGET_DIR/*`
Second blank: `mv`

The completed script is:

```
TARGET_DIR=/usr/local/certs
for f in $TARGET_DIR/*; do
  mv $f `openssl sha -sha256 -r < $f | cut -f1 -d' '`
done
```

`$TARGET_DIR/*` iterates through every file in the certificates directory. For each file, `openssl sha -sha256 -r` produces the SHA-256 value, `cut` selects the hash field, and `mv` renames the file using that hash.

QUESTION NO: 52

Which two commands download and execute an Apache web server container in Docker with a port binding 8080 in the container to 80 on the host? (Choose two.)

A. docker pull apache

B. docker run -p 8080:80 httpd

C. docker run -p 80:8080 httpd

D. docker pull httpd

E. docker pull https

ANSWER: B D

Explanation:

`docker pull httpd` retrieves the official Apache HTTP Server image, whose Docker Hub repository name is `httpd`, from the configured registry and stores it in the local image cache. This explicitly performs the download portion of the task. Docker documents `docker pull` as the command that downloads an image or repository from a registry.

`docker run -p 8080:80 httpd` creates and starts a container from that Apache HTTP Server image. Docker port publishing uses the format `-p HOST_PORT:CONTAINER_PORT`; consequently, this command publishes port 8080 on the Docker host to port 80 in the container, which is the HTTP port exposed by the standard `httpd` image. After the container starts, requests sent to `http://<host>:8080` are forwarded to Apache in the container. Docker can also pull an image automatically during `docker run` when it is not already local, but using the explicit pull command together with the run command clearly separates downloading the image from starting the published service. See the Docker references for docker image pull and docker container run and port publishing.

QUESTION NO: 53

Which two functions are commonly performed by a Cisco IOS XE RESTCONF server? (Choose two.)

A. Retrieve YANG-modeled operational data.

B. Modify YANG-modeled configuration data.

C. Forward packets between IP networks.

D. Resolve hostnames into IP addresses.

E. Provide Layer 2 MAC address learning.

ANSWER: A B

Explanation:

Retrieve YANG-modeled operational data is a RESTCONF function. A client can use HTTP GET requests to obtain representations of resources defined by supported YANG models, including operational state such as interface status and counters.

Modify YANG-modeled configuration data is also a RESTCONF function. A client can use methods such as PATCH, PUT, POST, and DELETE, according to the resource and intended operation, to create, change, or remove configuration through the RESTCONF API. RESTCONF provides an HTTP-based interface to datastore resources defined by YANG. See RFC 8040: RESTCONF Protocol and Cisco IOS XE RESTCONF documentation.

QUESTION NO: 54

Which Cisco IOS XE command enables the NETCONF server for model-driven configuration management?

A. `netconf-yang`

B. `restconf`

C. `ip http secure-server`

D. `yang-server enable`

ANSWER: A

Explanation:

The `netconf-yang` global configuration command enables the NETCONF server on Cisco IOS XE. After NETCONF is enabled and the required SSH and user authentication configuration is in place, a NETCONF client can establish a secure management session and use YANG-modeled operations such as `<get-config>` and `<edit-config>`.

The `restconf` command enables the RESTCONF API instead of NETCONF. Commands such as `ip http secure-server` enable HTTPS services, but they do not by themselves enable the IOS XE NETCONF server. Cisco documents IOS XE model-driven interfaces in the Cisco IOS XE NETCONF documentation.

QUESTION NO: 55

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

A. Cisco Intersight API interactions are JSON-encoded and require an API key in the HTTP header for authentication.

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

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

D. Cisco Intersight uses XML to encoded API interactions and requires an API key pair for authentication.

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

ANSWER: A C

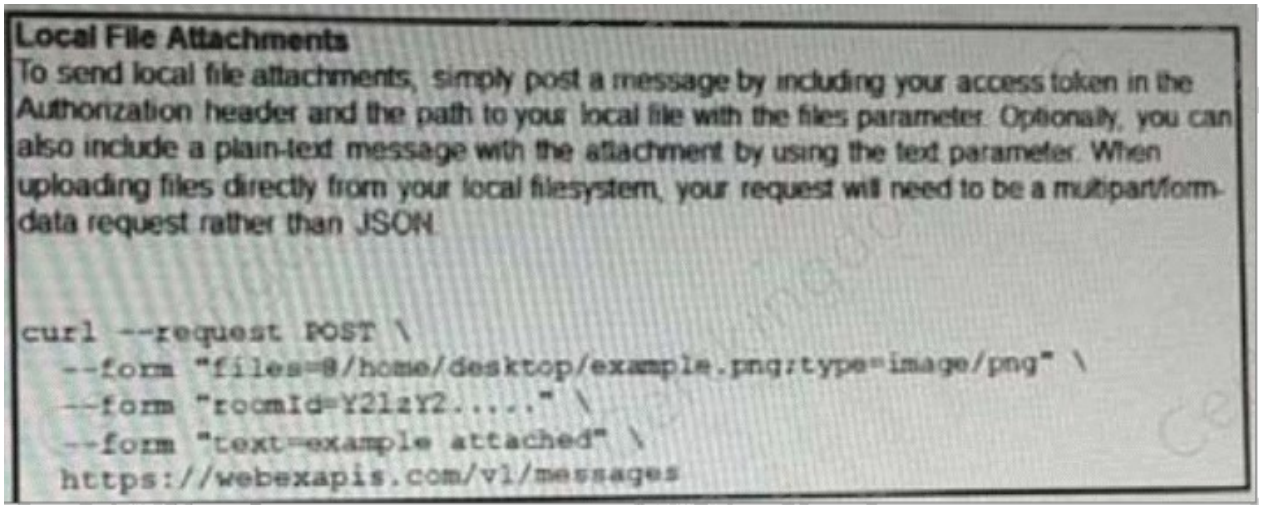
Explanation:

Cisco Intersight API interactions are JSON-encoded and use API-key-based authentication. An Intersight API key consists of a key identifier and a private key used to sign requests. The resulting authentication information, including the key identifier, signature, and signed headers, is supplied in HTTP headers. This signed-request approach permits secure, programmatic access to Intersight resources without using an interactive username-and-password login session. Cisco documents the required HTTP headers and the JSON request and response representations in its Intersight API overview.

UCS Manager API interactions are XML-encoded and use a session cookie for authenticated operations. A client first sends an XML login request to the UCS Manager XML API. The login response provides a cookie value, which the client includes as the cookie attribute in later XML API method requests. The cookie identifies the authenticated session and enables the client to perform management actions until the session expires or is logged out. This is the standard session model described in the Cisco UCS Manager XML API Programmer Guide. These two mechanisms reflect the distinct API designs: signed HTTP requests for Intersight and XML session-based requests for UCS Manager.

QUESTION NO: 56

Refer to the exhibit.



A developer needs to upload a local file by using the REST API. The developer gathers information according to the documentation and sends the request by using the cURL command in the exhibit but gets an error code. Which action should be followed to get valid response?

A. change content type as JSON

B. add the authorization header with the access token

C. add a username-password combination to request command

D. change request method as GET

ANSWER: B

Explanation:

add the authorization header with the access token is the required action because a file-upload endpoint is typically protected and requires the caller to present valid credentials on the API request itself. The cURL multipart upload syntax sends the local file as form data, but it does not establish the identity or permissions of the client. Supplying the access token in the request's `Authorization` header allows the API gateway or application to validate the token, identify the caller, and authorize access to the upload resource.

For bearer-token authentication, the header is normally formatted as `Authorization: Bearer <access-token>`. For example, a multipart cURL upload can include both `-H "Authorization: Bearer <access-token>"` and the appropriate `-F` argument for the local file. This preserves the required multipart request body while adding the security context required for the server to return a successful response. OAuth 2.0 bearer-token usage specifies that clients send access tokens in the HTTP Authorization request header: RFC 6750. Cisco's API guidance likewise documents obtaining and using access tokens for authenticated API calls: Cisco DevNet OAuth 2.0 overview.

QUESTION NO: 57

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:

server successfully processed request; no content returned is correct because HTTP status code 204 is defined as *No Content*. It indicates that the server has successfully fulfilled the request, but there is no message content to send in the response body. A 204 response is therefore a successful response in the 2xx class, while explicitly communicating that the client does not need to receive, parse, or render a representation of a resource.

This status is commonly appropriate for successful API operations such as deleting a resource, updating a resource when no updated representation needs to be returned, or acknowledging an action whose result requires no additional client-side display. HTTP semantics specify that a 204 response is terminated by the end of the header section and cannot contain content. The response may still include useful header fields, such as metadata or cache-related headers, even though it has no body. In REST-oriented API design, using 204 can avoid transferring redundant data when the client only needs confirmation that the requested operation succeeded. See RFC 9110, Section 15.3.5: 204 No Content and MDN Web Docs: 204 No Content.

QUESTION NO: 58

Which platform is run directly using a hypervisor?

A. bare metal systems

B. containers

C. virtual machines

D. applications

ANSWER: C

Explanation:

virtual machines is correct because a hypervisor is the virtualization layer that creates, runs, and manages virtual machines. It abstracts the underlying physical hardware—such as processor, memory, storage, and network interfaces—and presents virtualized versions of those resources to each guest operating system. Each virtual machine can therefore run its own operating system and software workload while remaining logically isolated from other virtual machines hosted on the same physical server. In a type-one deployment, the hypervisor runs directly on server hardware; in a type-two deployment, it runs on a host operating system. In both models, the virtual machine is the guest platform executed and controlled by the hypervisor. This model enables server consolidation, efficient hardware utilization, workload isolation, and flexible provisioning of environments for development, testing, and production. Cisco describes virtualization as using a hypervisor to create and run virtual machines on a physical host, while VMware explains that the hypervisor allocates physical resources to the VMs it manages. See Cisco: What Is Virtualization? and VMware: What Is a Hypervisor?.

QUESTION NO: 59

What are two benefits of using a Docker volume for persistent application data? (Choose two.)

A. Data is included automatically in every image build.

B. Data persists after a container is removed.

C. Data can be shared by multiple containers.

D. Volumes eliminate the need for container images.

E. Volumes require the application to use HTTP.

ANSWER: B C

Explanation:

Data persists after a container is removed because a Docker volume has a lifecycle independent of the container that mounts it. Removing and recreating an application container does not remove the volume unless the volume is explicitly removed. This makes volumes appropriate for data such as databases, application uploads, and logs that must survive container replacement.

Data can be shared by multiple containers when those containers are configured to mount the same volume. This can support architectures in which one container writes generated content and another container serves or processes that content. Access mode and application-level coordination must still be designed appropriately. Docker documents volumes as persistent storage managed outside a container's writable layer. See Docker volume documentation.

QUESTION NO: 60

What are two benefit of managing network configuration via APIs? (Choose two.)

A. configuration on devices becomes less complex

B. more security due to locking out manual device configuration

C. increased scalability and consistency of network changes

D. eliminates the need of legacy management protocols like SNMP

E. reduction in network changes performed manually

ANSWER: C E

Explanation:

Managing network configuration through APIs enables software to interact directly with network platforms using defined, repeatable interfaces. **Increased scalability and consistency of network changes** is a key benefit because the same programmatic workflow, template, or source-of-truth data can be applied across many devices. This reduces configuration drift and allows an organization to make changes at a scale that would be difficult to achieve through individual device sessions. APIs also support validation and integration with automation tools and CI/CD-style workflows, helping teams apply intended configuration state consistently.

Reduction in network changes performed manually is also a direct benefit. Rather than an engineer repeatedly entering equivalent commands device by device, an application or automation workflow can submit configuration changes through the API. This improves operational efficiency and makes changes more repeatable, while allowing engineers to focus on design, review, testing, and exception handling. Cisco describes APIs as a means for applications to programmatically interact with network devices and controllers, supporting automation and orchestration. Cisco's DevNet learning materials discuss using APIs for network programmability at Cisco DevNet Learning, and Cisco provides API documentation and automation resources at Cisco DevNet Documentation.

QUESTION NO: 61

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 E

Explanation:

Model-driven programmability provides **model-driven APIs for abstraction and simplification** because a data model defines a consistent schema for configuration and operational data. Rather than requiring an automation client to handle device-specific CLI syntax and unstructured output, the client interacts with well-defined data nodes, types, constraints, and relationships. This creates a predictable interface that is easier for software to consume and validate at scale.

It also provides **models decoupled from transport, protocol, and encoding**. A YANG model defines the data independently of the mechanism used to exchange that data. Consequently, the same modeled resources can be accessed through interfaces such as NETCONF or RESTCONF and represented using encodings appropriate to the protocol, including XML or JSON. This separation lets the model remain the stable description of the device data while transports and encodings can vary according to operational requirements. YANG's purpose as a language for modeling configuration and state data is specified in RFC 7950. Cisco also documents how RESTCONF exposes modeled data on IOS XE in its IOS XE RESTCONF documentation.

QUESTION NO: 62

Which two characteristics distinguish a message queue from a synchronous API request? (Choose two.)

A. It decouples message producers from consumers.

B. It can buffer work until a consumer processes it.

C. It requires the producer to wait for a completed result.

D. It uses only HTTP GET requests.

E. It eliminates the need for message validation.

ANSWER: A B

Explanation:

A message queue decouples a producer from a consumer. The producer places a message on the queue without requiring the consumer to process the message during the same interaction. A consumer can retrieve and process the message later, which allows components to operate independently and at different rates.

A queue can also buffer messages when consumers are temporarily busy or unavailable. This helps absorb bursts of work and supports resilient asynchronous processing. In contrast, a synchronous API client normally waits for the server to process the request and return a response before continuing dependent work. The RabbitMQ work queue tutorial describes queues as a mechanism for sending messages from producers to consumers for later processing.

QUESTION NO: 63

Which type of threat occurs when an attacker can send hostile data to an interpreter within an application?

A. sensitive data exposure

B. broken authentication

C. cross-site scripting

D. injection

ANSWER: D

Explanation:

Injection is the threat that occurs when an application sends untrusted, attacker-controlled data to an interpreter and the interpreter treats that data as executable commands, query syntax, or other instructions rather than solely as data. Interpreters commonly involved include SQL database engines, operating-system command shells, LDAP query processors, and expression-language parsers. For example, an application that concatenates user input directly into a SQL statement can allow an attacker to alter the statement's logic, potentially reading, changing, or deleting data beyond what the application intended to permit. The defining feature is the unsafe crossing of a trust boundary: hostile input reaches an interpreter without a mechanism that reliably separates program instructions from supplied values. Effective prevention uses parameterized queries or prepared statements for database access, safe APIs, allow-list validation where appropriate, and least-privilege permissions for accounts used by the application. These controls reduce the chance that supplied input can change the meaning or execution path of an interpreted command. OWASP identifies injection as a major application-security risk and provides detailed practices for preventing it in the OWASP Injection overview and the OWASP Injection Prevention Cheat Sheet.

QUESTION NO: 64

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 characterized by writing an automated test before implementing the production code that fulfills the specified behavior. The developer begins with a small test for a required outcome, runs it to establish that the behavior is not yet implemented, writes only enough code to make the test pass, and then improves the internal design while retaining passing tests. This short, repeated red-green-refactor cycle makes the tests an active guide for implementation rather than a final verification activity.

It also enables code refactoring because the accumulating automated test suite acts as a safety net. After behavior is covered by tests, developers can reorganize, simplify, or otherwise improve the implementation and run the suite immediately to verify that externally observable behavior remains intact. Refactoring is therefore an integral part of the TDD cycle, helping maintain clean design as functionality is added incrementally. Cisco DevNet candidates should recognize that TDD is centered on developer-written automated tests and continuous feedback throughout implementation. See Martin Fowler's overview of Test-Driven Development and the Agile Alliance TDD glossary.

QUESTION NO: 65

```
$ find /home/user/backup -mtime +30 -type f -print
$ find /home/user/backup -mtime +30 -type f -delete
```

Refer to the exhibit. An engineer is managing the network of an enterprise. The network is using a distributed deployment model. The enterprise uses database to store logs. The current policy requires logs to be stored if changes are made to the infrastructure in any of the devices on the data centers. Which workflow is being automated by the Bash script?

A. returning backup files that are older than 30 days

B. deleting backup files that are older than 30 days

C. configuring the directory to delete files automatically

D. automating deletion of all backup files every 30 days

ANSWER: B

Explanation:

The workflow being automated is **deleting backup files that are older than 30 days**. The script uses the `find` utility to search the `/home/user/backup` directory for regular files whose modification time meets the expression `-mtime +30`. In GNU find, a value of `+30` matches files modified more than 30 24-hour periods ago. One command prints the matching file paths, which can provide visibility into the files selected for cleanup. The command using `-delete` then removes the matching regular files. This is a common retention-cleanup workflow: backups are retained for a defined period and files older than that retention threshold are purged. The script performs deletion when it runs; a separate scheduler, such as cron, would be required to run it at a particular recurring interval. The GNU find manual documents both age-based file selection with `-mtime` and deletion with the `-delete` action. See the GNU Findutils manual and the find(1) manual page.

QUESTION NO: 66

Which two encoding formats do YANG interfaces support?

A. JSON

B. XML

C. XHTML

D. Plain text

E. BER

ANSWER: A B

Explanation:

JSON and **XML** are the two standard data encodings used by YANG-based management interfaces. YANG defines the hierarchical schema for configuration and operational state data, while management protocols serialize instances of that schema for transport. NETCONF uses XML as its required encoding: protocol operations, replies, notifications, and YANG-modeled data are represented in XML. RESTCONF is designed to expose YANG data resources through HTTP and supports both XML and JSON representations, with content negotiation used to select the media type for requests and responses. JSON support follows the defined YANG-to-JSON mapping, which preserves YANG data-model structure, namespaces, types, lists, and leaf values in a standardized interoperable format. These formats allow applications to work with the same YANG model through commonly deployed programmatic APIs, selecting XML where NETCONF-oriented tooling is used or JSON where HTTP/REST-oriented clients are preferred. Cisco IOS XE model-driven programmability similarly uses YANG models with NETCONF and RESTCONF interfaces, making XML and JSON the relevant payload encodings. See the NETCONF Protocol specification and the RESTCONF Protocol specification.

QUESTION NO: 67

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:

The command `devnet > output.txt` saves the standard output produced by `devnet` to a file named `output.txt`. In Bash, the `>` operator performs output redirection: rather than displaying the command’s standard output in the terminal, the shell sends that stream to the specified file. Bash creates `output.txt` if it does not already exist. If the file already exists, this form of redirection replaces its current contents with the new command output. This behavior is useful when command results need to be retained for inspection, supplied to another process later, or incorporated into automation workflows. The redirection is handled by the shell before the command runs, so it works with ordinary commands, scripts, and executable programs that write to standard output. Standard error is a separate stream and is not included by this command; the question specifically asks to save the command output, for which standard-output redirection is the appropriate syntax. For detailed Bash redirection semantics, see the GNU Bash Reference Manual: Redirections and the bash(1) manual page.

QUESTION NO: 68

Which two statements about JSON and XML are true? (Choose two.)

A. JSON objects are collection of key value pair.

B. The syntax of JSON contains tags, elements, and attributes.

C. JSON arrays are an unordered set of key value pairs.

D. The syntax of XML contains tags, elements, and attributes.

E. XML objects are collections of key-value pairs.

ANSWER: A D

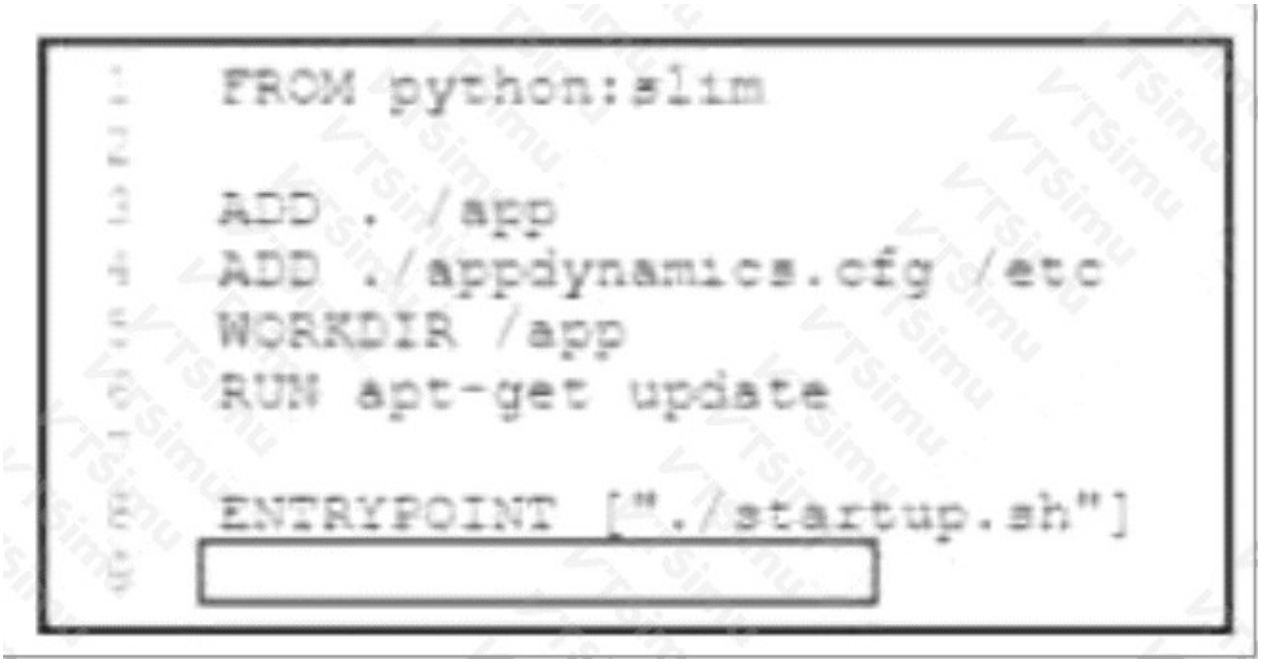
Explanation:

“JSON objects are collection of key value pair.” is correct in substance: a JSON object is a collection of name/value members enclosed in braces. Each member name is a string, and its value can be a string, number, Boolean, null, array, or another object. This compact object representation makes JSON especially common in REST API requests and responses, where clients and services exchange structured data. The formal JSON specification defines an object as an unordered collection of zero or more name/value pairs; see RFC 8259.

“The syntax of XML contains tags, elements, and attributes.” is also correct. XML represents structured information using markup. Elements are delimited by start and end tags (or expressed as empty-element tags), can be nested to form a document hierarchy, and can carry attributes that provide additional information associated with an element. For example, an element can identify an interface while attributes specify properties of that interface. This tag-based syntax is fundamental to XML document structure and allows XML data to be self-describing. The World Wide Web Consortium’s XML recommendation specifies the grammar for elements, tags, and attributes; see Extensible Markup Language (XML) 1.0.

QUESTION NO: 69

Refer to the exhibit.



A developer needs to create a Docker image that listens on port 5000. Which code snippet must be placed onto the blank in the code?

A. PORT 5000

B. LISTEN 5000

C. EXPOSE 5000

D. OPEN 5000

ANSWER: C

Explanation:

`EXPOSE 5000` is the Dockerfile instruction used to declare that the application in the image is intended to accept network connections on TCP port 5000. The instruction adds port metadata to the built image, making the expected service port clear to users of the image and to container tooling. For example, an application such as a web API can run in the container and bind to port 5000, while the Dockerfile communicates that network expectation with `EXPOSE 5000`.

Exposing a port in the Dockerfile does not by itself make the service reachable from outside the host. A user running a container can publish the declared container port by using a command such as `docker run -p 5000:5000 image-`

`name`, which maps host port 5000 to container port 5000. Alternatively, `docker run -P` can publish exposed ports to dynamically selected host ports. Therefore, placing `EXPOSE 5000` in the blank correctly documents the image's listening port and supports standard Docker port-publishing workflows. See the Dockerfile EXPOSE reference and Docker's guide to publishing and exposing ports.

QUESTION NO: 70

A development team is creating an application used for contactless payments. The application must: Be web-based

Capture and process the credit card information for a purchase.

Which security action must the web application use to gather and process the private customer data?

A. Enable RATs to monitor the web application remotely.

B. Disable botnets to eliminate risks.

C. Disable TLS to increase the connection speed.

D. Enable the encryption of network traffic.

ANSWER: D

Explanation:

Enable the encryption of network traffic. A web-based application that captures and processes payment-card information must protect cardholder data as it travels between the customer's browser or device and the application's web endpoint. In practice, this means enforcing HTTPS with properly configured TLS for every page, API call, redirect, and third-party interaction that can carry payment or session data. TLS provides confidentiality, so intercepted traffic does not disclose card details; integrity, so an attacker cannot silently alter transmitted data; and server authentication, so customers can verify they are communicating with the intended payment site. These protections are essential when users connect through untrusted networks, such as public Wi-Fi, and are a baseline expectation for payment applications. PCI DSS requires strong cryptography and security protocols to safeguard cardholder data during transmission over open, public networks. Secure implementation also includes using current TLS versions, trusted and valid certificates, secure cookie settings, and redirecting HTTP requests to HTTPS. While an application handling payment data needs additional controls, including secure coding, access control, and minimizing stored cardholder data, encrypting network traffic is the required foundational action for safely collecting that information through a web application. See the PCI Security Standards Council PCI DSS overview and the OWASP Transport Layer Security Cheat Sheet.

QUESTION NO: 71

An engineer must configure Cisco Nexus devices and wants to automate this workflow. The engineer will use an Ansible playbook to configure devices through Cisco NX REST API. Before the code is run, which resource must be used to verify that the REST API requests work as expected?

A. Cisco Open NX-OS

B. Cisco NX-OS SDK

C. Cisco Learning Labs

D. Cisco Code Exchange for Nexus

E. NX-API REST API browser (Swagger-like documentation on the Nexus switch)

ANSWER: E

Explanation:

NX-API REST API browser (Swagger-like documentation on the Nexus switch) is the correct resource because it provides an interactive, device-hosted interface for discovering and validating NX-API REST operations before those calls are incorporated into an Ansible playbook. The browser exposes the REST endpoints supported by the switch, including the HTTP methods, URI paths, request parameters, expected JSON payloads, and response structures. An engineer can authenticate to the target Nexus device and submit a test request directly, confirming that the request syntax and payload are accepted by the actual NX-OS release and device configuration in use. This is especially useful for confirming the data model and exact distinguished names or managed-object paths required for configuration requests. Validating requests first helps ensure that Ansible tasks send known-good API calls and makes troubleshooting automation substantially easier. Cisco documents access to the NX-API REST API browser through the switch's NX-API service and describes it as an interface for browsing and trying REST methods. See the Cisco DevNet NX-API REST documentation and the Cisco Nexus 9000 NX-OS Programmability Guide.

QUESTION NO: 72

What is a characteristic of the Cisco Finesse platform?

A. Applications allow services to be invoked on a network triggered event.

B. The platform provides a ready-to-go platform for HD video and audio conferencing.

C. Applications are added to the platform from the desktop remotely.

D. The platform includes an enterprise-class IM engine.

ANSWER: A

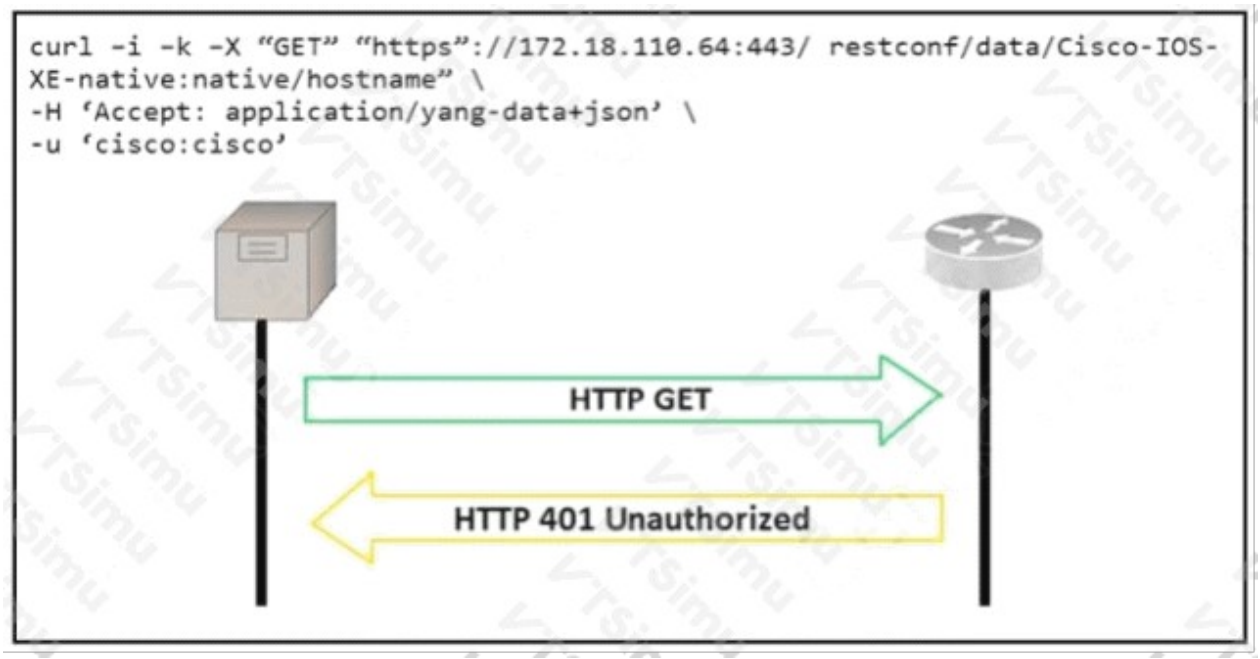
Explanation:

Applications allow services to be invoked on a network triggered event. Cisco Finesse is an extensible, web-based contact-center agent and supervisor desktop platform for Cisco contact-center deployments. Its design enables custom web applications, commonly called gadgets, to be integrated into the desktop experience. These applications can use Finesse APIs and event notifications to respond dynamically when relevant contact-center activity occurs, such as a change in an agent's state, the creation or update of a call dialog, or other workflow-related events.

This event-driven model lets developers connect business systems and services to the agent workflow at the appropriate moment. For example, a gadget can react to call-related context by retrieving customer information from a CRM system, presenting a screen pop, or initiating an approved business process. Finesse therefore provides a framework for event-aware applications rather than simply a fixed desktop interface. Cisco's developer documentation describes the JavaScript APIs, REST APIs, and event mechanisms that support gadget development and real-time desktop integrations. See the Cisco Finesse Developer Documentation and Cisco's Cisco Finesse product information for details on its extensible contact-center desktop architecture.

QUESTION NO: 73

Refer to the exhibit.



An administrator attempts to perform a GET operation by using the Cisco IOS XE RESTCONF API to return the hostname of a device. The sequence diagram in the exhibit illustrates the HTTP messages observed. Which change to the API request resolves the issue?

A. Remove the -H 'Accept: application/yang-data+json' header.

B. Replace -u cisco:cisco parameter with -u '~cisco:cisco'.

C. Change the request method from -X '~GET' to -X '~POST'.

D. Add the -H '~Content-Type: application/yang-data+json' header.

ANSWER: D

Explanation:

Add the -H '~Content-Type: application/yang-data+json' header. Cisco IOS XE RESTCONF exposes configuration and operational resources through YANG models, and YANG JSON is identified with the `application/yang-data+json` media type. Supplying this header makes the requested RESTCONF/YANG JSON representation explicit to the device and allows the request to be processed in the expected YANG data context. In the illustrated exchange, the device is rejecting the request because the required media-type information is not fully provided. Once the YANG JSON content type is included, the RESTCONF service can handle the request and return the modeled hostname resource in JSON form. RESTCONF defines the YANG data JSON media type and its use for RESTCONF data resources in RFC 8040. Cisco documents IOS XE RESTCONF support, including requests against YANG-modeled native IOS XE data, in its IOS XE Programmability Configuration Guide. Using the standard YANG JSON media type also keeps the request consistent with RESTCONF tooling and IOS XE API examples.

QUESTION NO: 74

Which two HTTP methods are idempotent when used according to their HTTP semantics? (Choose two.)

A. POST

B. PUT

C. PATCH

D. DELETE

E. CONNECT

ANSWER: B D

Explanation:

PUT is idempotent because repeating the same request has the same intended effect as sending it once. For example, sending an identical PUT request to replace a resource representation multiple times leaves the resource in the same intended state after each request.

DELETE is also idempotent. A first successful DELETE can remove the target resource, and subsequent equivalent DELETE requests do not cause additional deletion effects. The later responses can differ, such as returning a not-found response after the resource has been removed, but the intended server state remains the same. HTTP defines PUT and DELETE as idempotent request methods. See RFC 9110, Idempotent Methods.

QUESTION NO: 75

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

A. simplifies network automation

B. allows for multiple views of the same model

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

D. reduces need for error handling

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

ANSWER: B E

Explanation:

The Model-View-Controller pattern separates an application into distinct concerns: the model manages application data and business rules, the view presents information to users, and the controller coordinates user input and application behavior. Consequently, it **allows for multiple views of the same model**. Because presentation is separated from the data and business logic, the same model can support different interfaces, such as an HTML page, a mobile-oriented display, or an API-oriented representation, without duplicating the underlying domain logic.

MVC also **separates responsibilities of the code, which makes future modifications easier**. This organization establishes clearer boundaries between user-interface work, request/input handling, and business/data behavior. Teams can update a view while minimizing changes to model code, or revise business rules while preserving the presentation layer. The resulting separation of concerns improves maintainability, testability, and the ability to evolve an application over time. Cisco DevNet training discusses applying software design patterns and modular application design, while the architectural definition and responsibilities of MVC are described by the IBM documentation on MVC. See Cisco DevNet Associate learning track and IBM MVC pattern overview.

QUESTION NO: 76

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

A. 400

B. 200

C. 500

D. 300

E. 100

ANSWER: A C

Explanation:

The **400** and **500** HTTP status-code series are the two error classes. HTTP status codes are categorized by their first digit, which communicates the general result of a request. The 400 series represents client errors: the server has received the request but cannot process it as submitted. This class includes situations such as invalid request syntax, failed authentication or authorization, and requests for resources that cannot be found. In API development, a 400-series response indicates that the API consumer should inspect and correct the request, such as its URL, headers, credentials, parameters, or body.

The 500 series represents server errors. These responses indicate that the server encountered a problem while attempting to fulfill an apparently valid request. Common server-side conditions include an application exception, an unavailable dependency, a gateway failure, or a configuration problem. For an API developer or operator, identifying a 500-series response helps direct investigation toward the service implementation, hosting environment, logs, and downstream systems. HTTP semantics formally defines 4xx as client error responses and 5xx as server error responses. See RFC 9110, client error responses and RFC 9110, server error responses.