Exam

350-901

Designing, Deploying and Managing Network Automation Systems

Free demo	80 sample questions
Full exam	549 questions with answers and explanations
Contents	Questions with answers and explanations

Exam domains

Domain	Exam weight	Questions
D1 Network Automation	30%	44
D2 Infrastructure as Code	30%	14
D3 Operations	20%	17
D4 AI in Automation	20%	3

This is a free demo from Dumpsgate: 80 of the exam's 549 questions. **Get the full 350-901 exam** with every question, answer and explanation.

Question 1

Single choice · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

You are designing a microservices architecture for a network automation platform that will be deployed across multiple data centers. The system must support independent scaling of different components, handle transient failures gracefully, and maintain low latency communication between services. One critical microservice retrieves configuration data from devices, another processes and transforms that data, and a third stores results in a database. During peak load testing, you observe that when the configuration retrieval service experiences a temporary outage, the entire platform becomes unresponsive.

Which architectural pattern change would best address this resilience issue while maintaining the design's modularity?

A. Implement an event-driven architecture with asynchronous message queues between services, allowing decoupling and graceful degradation

B. Convert all services to a monolithic application to reduce network calls and potential failures between components

C. Implement a synchronous service-oriented architecture with load balancing only at the front-end tier

D. Deploy all services in a single container to eliminate inter-service communication latency

Answer: A. Implement an event-driven architecture with asynchronous message queues between services, allowing decoupling and graceful degradation

Explanation

Option A addresses the failure mode by making communication between the retrieval, processing, and storage services asynchronous. If retrieval temporarily fails, the other services do not have to wait on a synchronous call or become unresponsive; messages can be buffered for processing when the retrieval service recovers. This preserves independently scalable, modular services and supports graceful degradation, though it does not by itself guarantee low latency in every case.

✓ **A** Implement an event-driven architecture with asynchronous message queues between services, allowing decoupling and graceful degradation
Asynchronous queues decouple the processing and storage services from the immediate availability of configuration retrieval. During a temporary outage, work can be buffered rather than causing dependent synchronous calls to block the platform, and services can continue operating independently where possible. This also retains the modularity needed for independent scaling.

✗ **B** Convert all services to a monolithic application to reduce network calls and potential failures between components
A monolith may reduce some inter-service network calls, but it abandons the required modular microservices design and independent scaling. It also does not specifically provide graceful handling of the retrieval service's temporary outage; a failure in that component could still affect the whole application.

✗ **C** Implement a synchronous service-oriented architecture with load balancing only at the front-end tier
Synchronous service calls can leave processing or storage requests waiting on the unavailable retrieval service, preserving the dependency that made the platform unresponsive. Front-end load balancing does not address failures or bottlenecks between the internal services.

Free demo · full exam: 549 questions

✗ **D** Deploy all services in a single container to eliminate inter-service communication latency
Putting the services in one container does not decouple them or provide a queue to absorb work during a retrieval outage. It also undermines independent deployment and scaling, and eliminating some communication latency does not solve the observed cascading unavailability.

Question 2

Single choice · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

A network environment requires automation for configuration backups and rapid user interface development to meet tight deadlines. The team wants a solution that uses a no-code approach to streamline operations and enables intuitive dashboard creation for quick adaptation. Which approach meets the requirements?

A. scheduled sync via batch processes
B. limited asset management features
C. visual dashboard assembly
D. form-based static input entry

Answer: C. visual dashboard assembly

Explanation

Visual dashboard assembly best fits the need for quick, intuitive interface development under tight deadlines. It enables the team to build and adapt dashboards visually rather than relying on static forms or batch processes. This approach addresses the interface requirement while supporting a no-code style of work.

✗ **A** scheduled sync via batch processes
Scheduled batch processes can automate recurring synchronization, but they do not provide an intuitive way to create or rapidly adapt a user interface. They therefore do not meet the dashboard-development requirement.

✗ **B** limited asset management features
Limited asset management features do not address the need for configuration-backup automation or rapid interface development. They are not a dashboard-creation approach.

✓ **C** visual dashboard assembly
Visual dashboard assembly lets the team create an interface intuitively and make changes quickly, matching the tight-deadline and rapid-adaptation requirements. It is the option most consistent with a no-code approach to streamlining operations.

✗ **D** form-based static input entry
Static forms collect inputs but do not provide the flexible dashboard-building capability requested. They are a poor fit when the interface must be adapted quickly.

Free demo · full exam: 549 questions

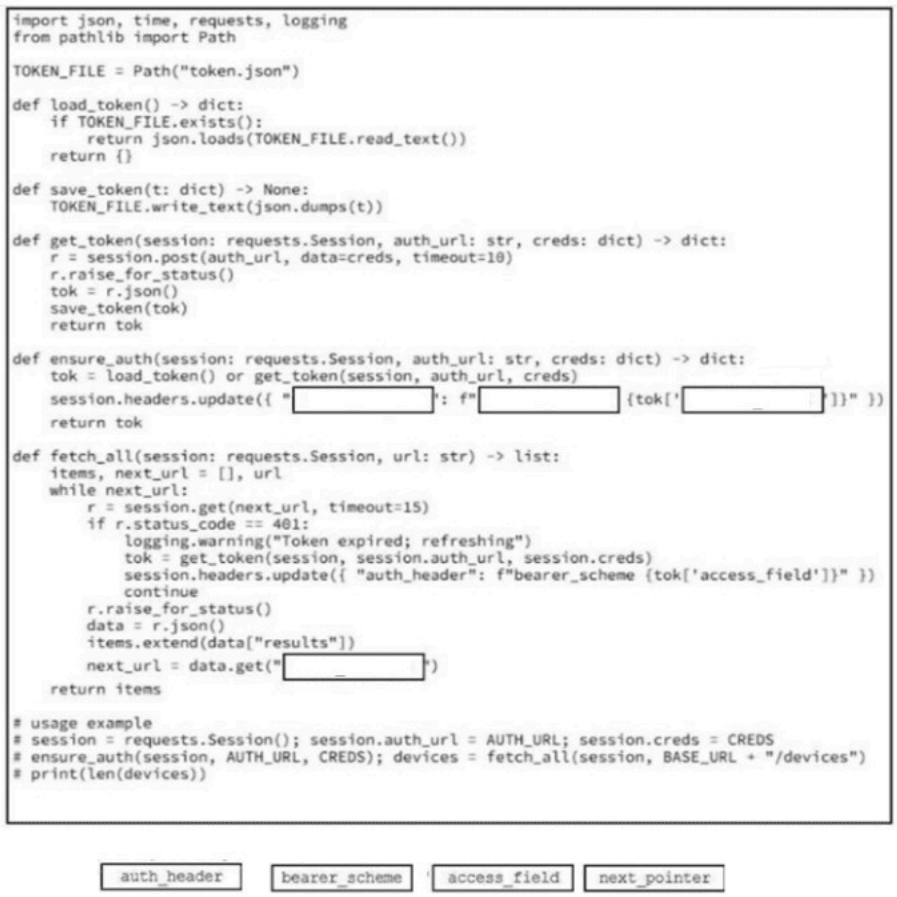
Question 3

Drag and drop · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

An engineer must build a Python client that pulls device inventory from a REST API using OAuth2 and paginates results in the response body. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the client. Not all options are used.

```
import json, time, requests, logging
from pathlib import Path

TOKEN_FILE = Path("token.json")

def load_token() -> dict:
    if TOKEN_FILE.exists():
        return json.loads(TOKEN_FILE.read_text())
    return {}

def save_token(t: dict) -> None:
    TOKEN_FILE.write_text(json.dumps(t))

def get_token(session: requests.Session, auth_url: str, creds: dict) -> dict:
    r = session.post(auth_url, data=creds, timeout=10)
    r.raise_for_status()
    tok = r.json()
    save_token(tok)
    return tok

def ensure_auth(session: requests.Session, auth_url: str, creds: dict) -> dict:
    tok = load_token() or get_token(session, auth_url, creds)
    session.headers.update({"[Blank 1]": f"[Blank 2] {tok['[Blank 3]']}"})
    return tok

def fetch_all(session: requests.Session, url: str) -> list:
    items, next_url = [], url
    while next_url:
        r = session.get(next_url, timeout=15)
        if r.status_code == 401:
            logging.warning("Token expired; refreshing")
            tok = get_token(session, session.auth_url, session.creds)
            session.headers.update({"auth_header": f"bearer_scheme {tok['access_field']}"})
            continue
        r.raise_for_status()
        data = r.json()
        items.extend(data["results"])
        next_url = data.get("[Blank 4]")
    return items

# usage example
# session = requests.Session(); session.auth_url = AUTH_URL; session.creds = CREDS
# ensure_auth(session, AUTH_URL, CREDS); devices = fetch_all(session, BASE_URL + "/devices"
# print(len(devices))
```


Free demo · full exam: 549 questions



```
import json, time, requests, logging
from pathlib import Path

TOKEN_FILE = Path("token.json")

def load_token() -> dict:
    if TOKEN_FILE.exists():
        return json.loads(TOKEN_FILE.read_text())
    return {}

def save_token(t: dict) -> None:
    TOKEN_FILE.write_text(json.dumps(t))

def get_token(session: requests.Session, auth_url: str, creds: dict) -> dict:
    r = session.post(auth_url, data=creds, timeout=10)
    r.raise_for_status()
    tok = r.json()
    save_token(tok)
    return tok

def ensure_auth(session: requests.Session, auth_url: str, creds: dict) -> dict:
    tok = load_token() or get_token(session, auth_url, creds)
    session.headers.update({ "[Blank 1]": f"[Blank 2] {tok['[Blank 3]']}" })
    return tok

def fetch_all(session: requests.Session, url: str) -> list:
    items, next_url = [], url
    while next_url:
        r = session.get(next_url, timeout=15)
        if r.status_code == 401:
            logging.warning("Token expired; refreshing")
            tok = get_token(session, session.auth_url, session.creds)
            session.headers.update({ "auth_header": f"bearer_scheme {tok['access_field']}" })
            continue
        r.raise_for_status()
        data = r.json()
        items.extend(data["results"])
        next_url = data.get("[Blank 4]")
    return items

# usage example
# session = requests.Session(); session.auth_url = AUTH_URL; session.creds = CREDS
# ensure_auth(session, AUTH_URL, CREDS); devices = fetch_all(session, BASE_URL + "/devices")
# print(len(devices))
```

auth_header | bearer_scheme | access_field | next_pointer

Items:

- A. auth_header
- B. bearer_scheme
- C. access_field
- D. next_pointer

Place them on:

1. Blank 1 (`session.headers.update({"[Blank 1]": ...})`)
2. Blank 2 (`f"[Blank 2] {tok[...]}`)
3. Blank 3 (`tok['[Blank 3]']`)
4. Blank 4 (`next_url = data.get("[Blank 4]")`)

Answer

- Blank 1 (`session.headers.update({"[Blank 1]": ...})`) → **A. auth_header**
- Blank 2 (`f"[Blank 2] {tok[...]}`) → **B. bearer_scheme**
- Blank 3 (`tok['[Blank 3]']`) → **C. access_field**
- Blank 4 (`next_url = data.get("[Blank 4]")`) → **D. next_pointer**

Explanation

The client needs to send the OAuth2 access token in its request headers and follow the pagination pointer returned in each response body. The header key is `auth_header`, and its value combines the

`bearer_scheme` with the token stored under `access_field`. The response's `next_pointer` value supplies the URL or cursor for the next page; the other drag items are all used in these four blanks.

✓ **Blank 1: auth_header** auth_header
This fills the header name passed to `session.headers.update`, so the client sends its authorization value with each request. The code also uses this same header key in the 401 refresh path.

✓ **Blank 2: bearer_scheme** bearer_scheme
This is the scheme placed before the token in the header value, producing the required scheme-and-token format. It is also used in the code's token-refresh branch.

https://developer.cisco.com/docs/ftd-api-reference/latest/authenticating-your-rest-api-client-using-oauth/

✓ **Blank 3: access_field** access_field
This selects the access-token value from the token response dictionary, which is needed to authenticate API requests. The refresh branch uses the same field to rebuild the header after a 401 response.

https://developer.cisco.com/docs/ftd-api-reference/latest/authenticating-your-rest-api-client-using-oauth/

✓ **Blank 4: next_pointer** next_pointer
The response body contains the pagination pointer under this key, so `data.get` can retrieve the next URL or cursor. Assigning it to `next_url` lets the loop continue until no next page is provided.

https://developer.cisco.com/docs/network-assurance-engine/query-and-response-structure/

Dumpsgate

Question 4

Drag and drop · Domain: D1 Network Automation › 1.1 Construct a network automation solution with Ansible to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

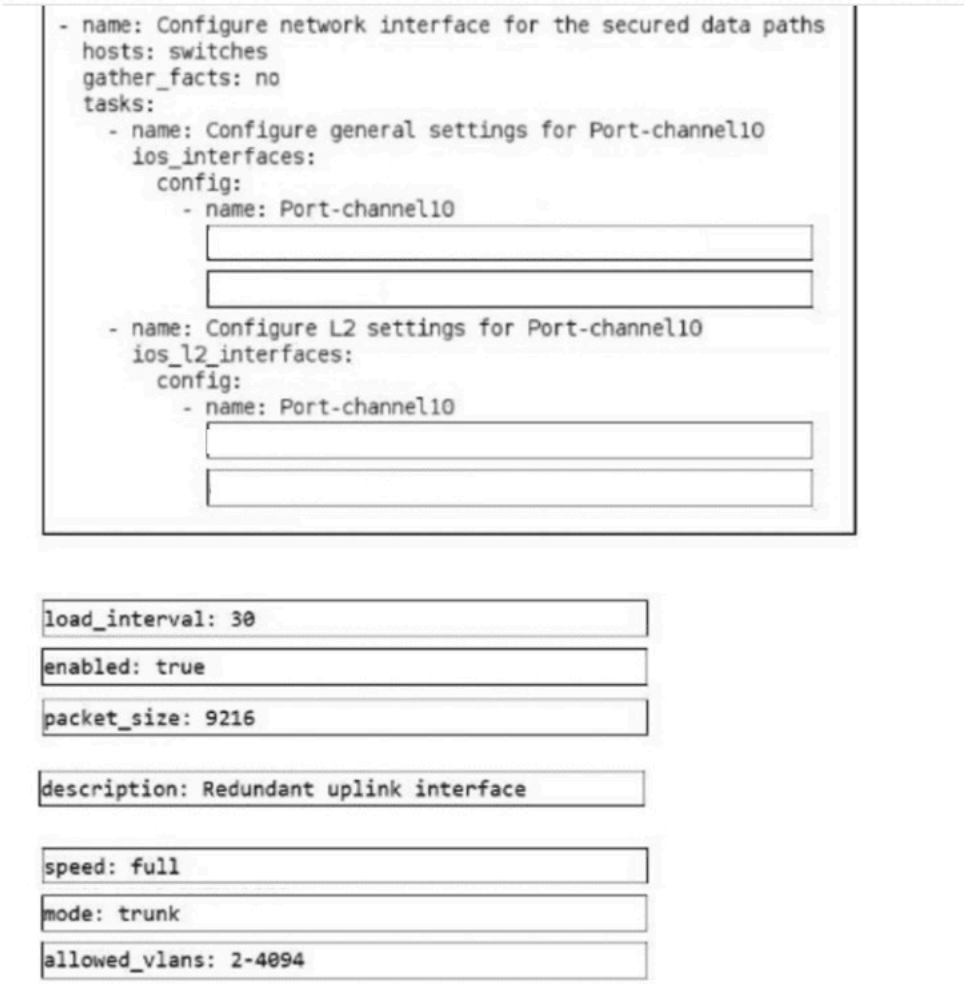
A network automation engineer is building an Ansible playbook to deploy interface configurations for high-availability uplinks on designated switches. The configuration must set all required parameters for Port-channel 10 without missing any steps. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Ansible playbook. Not all options are used.

```
- name: Configure network interface for the secured data paths
  hosts: switches
  gather_facts: no
  tasks:
    - name: Configure general settings for Port-channel10
      ios_interfaces:
        config:
          - name: Port-channel10
            [Blank 1]
            [Blank 2]
    - name: Configure L2 settings for Port-channel10
      ios_l2_interfaces:
        config:
          - name: Port-channel10
            [Blank 3]
            [Blank 4]
```

Drag items:

- load_interval: 30
- enabled: true
- packet_size: 9216
- description: Redundant uplink interface
- speed: full
- mode: trunk
- allowed_vlans: 2-4094

Free demo · full exam: 549 questions



Items:

- A. load_interval: 30
- B. enabled: true
- C. packet_size: 9216
- D. description: Redundant uplink interface
- E. speed: full
- F. mode: trunk
- G. allowed_vlans: 2-4094

Place them on:

1. Blank 1 (ios_interfaces config for Port-channel10)
2. Blank 2 (ios_interfaces config for Port-channel10)
3. Blank 3 (ios_l2_interfaces config for Port-channel10)
4. Blank 4 (ios_l2_interfaces config for Port-channel10)

Answer

- Blank 1 (ios_interfaces config for Port-channel10) → **D. description: Redundant uplink interface**
- Blank 2 (ios_interfaces config for Port-channel10) → **B. enabled: true**
- Blank 3 (ios_l2_interfaces config for Port-channel10) → **F. mode: trunk**
- Blank 4 (ios_l2_interfaces config for Port-channel10) → **G. allowed_vlans: 2-4094**

Free demo · full exam: 549 questions

Dumpsgate

Question 5

Drag and drop · Domain: D1 Network Automation › 1.3 Construct a network automation solution with RESTCONF (RFC 8040), given the YANG model, to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Refer to the exhibit.

```
module Cisco-IOS-XE-native {
  namespace "http://cisco.com/ns/yang/Cisco-IOS-XE-native";
  prefix "ios";

  import ietf-inet-types {
    prefix inet;
  }

  container router {
    container ospf {
      list process-id {
        key "id";
        leaf id {
          type uint32;
        }
        leaf router-id {
          type inet:ipv4-address;
        }
        list network {
          key "ip mask";
          leaf ip {
            type inet:ipv4-address;
          }
          leaf mask {
            type inet:ipv4-address;
          }
          leaf area {
            type uint32;
          }
        }
      }
    }
  }
}
```

A network engineer must use a RESTCONF-based Python script to deploy OSPF routing on a Cisco IOS XE device. The engineer must complete a Python script that sends a request to update the device configuration with OSPF process ID 1, router ID 1.1.1.1, and a network statement for 10.0.0.0/24 in area 0. The script leverages the Cisco IOS XE native YANG model, specifically the Cisco-IOS-XE-native native/router/ospf container, to structure the OSPF configuration payload. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Python script. Not all options are used.

```
import requests, json
from requests.auth import HTTPBasicAuth

device_ip, username, password = "192.0.2.1", "admin", "password"
url = f"https://{device_ip}/restconf/data/Cisco-IOS-XE-native:native/router/ospf"

headers = {"Content-Type": "application/yang-data+json", "Accept": "application/yang-data+j

[Blank 1]
```

```
    "Cisco-IOS-XE-native:ospf": {
        [Blank 2]
        "network": [{"ip": "10.0.0.0", "mask": "0.0.0.255",
        [Blank 3]
    }
}

try:
    r = requests.patch(url, auth=HTTPBasicAuth(username, password), headers=headers,
                       data=json.dumps(payload), verify=False)
    [Blank 4]
        print("OSPF configured successfully.")
    else:
        print(f"Failed to configure OSPF. Status code: {r.status_code} Response: {r.text}")
except Exception as e:
    print(f"Error: {e}")
```

Dumpsgate

```
module Cisco-IOS-XE-native {
  namespace "http://cisco.com/ns/yang/Cisco-IOS-XE-native";
  prefix "ios";

  import ietf-inet-types {
    prefix inet;
  }

  container router {
    container ospf {
      list process-id {
        key "id";
        leaf id {
          type uint32;
        }
        leaf router-id {
          type inet:ipv4-address;
        }
        list network {
          key "ip mask";
          leaf ip {
            type inet:ipv4-address;
          }
          leaf mask {
            type inet:ipv4-address;
          }
          leaf area {
            type uint32;
          }
        }
      }
    }
  }
}
```

```
import requests, json
from requests.auth import HTTPBasicAuth

device_ip, username, password = "192.0.2.1", "admin", "password"
url = f"https://{device_ip}/restconf/data/Cisco-IOS-XE-native:native/router/ospf"

headers = {"Content-Type": "application/yang-data+json", "Accept": "application/yang-data+json"}

    "Cisco-IOS-XE-native:ospf": {
        "network": [{"ip": "10.0.0.0", "mask": "0.0.0.255",
    }
}

try:
    r = requests.patch(url, auth=HTTPBasicAuth(username, password), headers=headers,
        data=json.dumps(payload), verify=False)

        print("OSPF configured successfully.")
    else:
        print(f"Failed to configure OSPF. Status code: {r.status_code} Response: {r.text}")
except Exception as e:
    print(f"Error: {e}")
```

```
if r.status_code in (200, 204):
payload = {
"area": 0}]}]
"process-id": [{"id": 1, "router-id": "1.1.1.1",
"ospf-process-id": [{"id": 1, "router-id": "1.1.1.1",
data = {
```

```
"area-id": 0}]}]
```

Items:

- A. if r.status_code in (200, 204):
- B. payload = {
- C. "area": 0}]}]
- D. "process-id": [{"id": 1, "router-id": "1.1.1.1",
- E. "ospf-process-id": [{"id": 1, "router-id": "1.1.1.1",
- F. data = {
- G. "area-id": 0}]}]

Place them on:

1. Blank 1 (before "Cisco-IOS-XE-native:ospf" in the payload)
2. Blank 2 (inside "Cisco-IOS-XE-native:ospf", before the network entry)
3. Blank 3 (after the network entry's "mask" value)
4. Blank 4 (after the requests.patch call)

Answer

- Blank 1 (before "Cisco-IOS-XE-native:ospf" in the payload) → **B. payload = {**
- Blank 2 (inside "Cisco-IOS-XE-native:ospf", before the network entry) → **D. "process-id": [{"id": 1, "router-id": "1.1.1.1",**
- Blank 3 (after the network entry's "mask" value) → **C. "area": 0}]}]**
- Blank 4 (after the requests.patch call) → **A. if r.status_code in (200, 204):**

Explanation

The payload must follow the YANG structure shown in the exhibit: the top-level `ospf` container contains a `process-id` list entry, and that entry contains the network list with its `area` value. The selected snippets provide process ID 1, router ID 1.1.1.1, and area 0 for the specified network. The final condition treats HTTP 200 or 204 as a successful PATCH response.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1718/b-1718-programmability-cg/m_1718_prog_restconf.html
https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✓ **Blank 1: payload = {** payload = {
This initializes the dictionary that the script serializes and sends as the RESTCONF request body. The payload then contains the `Cisco-IOS-XE-native:ospf` top-level key.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1718/b-1718-programmability-cg/m_1718_prog_restconf.html

✓ **Blank 2: "process-id": [{"id": 1, "router-id": "1.1.1.1",** "process-id": [{"id": 1, "router-id": "1.1.1.1",
The exhibit defines `process-id` as the list under `ospf`, with `id` as its key and `router-id` as a leaf. This entry therefore specifies process ID 1 and router ID 1.1.1.1 at the correct level of the payload.

https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✓ **Blank 3: "area": 0}]}]** "area": 0}]}]
The YANG model defines `area` as a leaf within each `network` list entry. This snippet sets the network's area to 0 and closes the network, process, and surrounding list structures.

https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✓ **Blank 4: if r.status_code in (200, 204):** if r.status_code in (200, 204):
This checks the PATCH response before printing the success message. The condition treats HTTP 200 and 204 responses as successful, matching the script's success and failure branches.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1718/b-1718-programmability-cg/m_1718_prog_restconf.html

✗ **"area": 0}]}]** "area": 0}]}]
This is a valid choice for Blank 3, but it is not the requested snippet for Blank 4, which must begin the response-status check. Used after the PATCH call, it would not provide the `if` condition needed by the following `else`.

✗ **"process-id": [{"id": 1, "router-id": "1.1.1.1",** "process-id": [{"id": 1, "router-id": "1.1.1.1",
This is the correct content for Blank 2, but not for Blank 4. After the PATCH call, the script needs a status-code condition rather than another configuration-payload field.

✗ **"ospf-process-id": [{"id": 1, "router-id": "1.1.1.1",** "ospf-process-id": [{"id": 1, "router-id": "1.1.1.1",
The exhibit names the list `process-id`, not `ospf-process-id`. Using this alternative would not match the YANG structure shown for the OSPF process list.

https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✗ **data = {** data = {
The request body is serialized from the variable `payload` in the PATCH call, so initializing `data` would leave `payload` undefined. The required dictionary assignment is `payload = {`.

✗ **"area-id": 0}]}]** "area-id": 0}]}]
The exhibit defines the network's area leaf as `area`, not `area-id`. This key would not match the shown YANG model for the network entry.

https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

Dumpsgate

Question 6

Drag and drop · Domain: D1 Network Automation › 1.1 Construct a network automation solution with Ansible to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

A network automation engineer is building an Ansible playbook to deploy interface configurations for the new VLAN segment on designated switches The configuration must set all required parameters for GigabitEthernet 1/0/2 without missing any steps Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Ansible playbook Not all options are used.

```
---
- name: Configure interface for new VLAN segment
  hosts: switches
  gather_facts: no
  tasks:
    - name: Set L1 attributes (description, enable, speed, duplex)
      ios_interfaces:
        config:
          - name: GigabitEthernet1/0/2
            [Blank 1]
            [Blank 2]
            speed: 1000
            duplex: full
      state: merged

    - name: Set L2 access mode and VLAN
      ios_l2_interfaces:
        config:
          - name: GigabitEthernet1/0/2
            [Blank 3]
            access:
              [Blank 4]
      state: merged
```

Drag items:

- vlan: 25
- trunk: 25
- description: Configured for new VLAN
- mode: switch
- enabled: true
- mode: access
- shutdown: no

Free demo · full exam: 549 questions



vlan: 25

trunk: 25

description: Configured for new VLAN

mode: switch

enabled: true

mode: access

shutdown: no

Items:

- A. vlan: 25
- B. trunk: 25
- C. description: Configured for new VLAN
- D. mode: switch
- E. enabled: true
- F. mode: access
- G. shutdown: no

Place them on:

1. Blank 1 (ios_interfaces config for GigabitEthernet1/0/2, before speed)
2. Blank 2 (ios_interfaces config for GigabitEthernet1/0/2, before speed)
3. Blank 3 (ios_l2_interfaces config for GigabitEthernet1/0/2, before access)
4. Blank 4 (access VLAN)

Free demo · full exam: 549 questions

Answer

- Blank 1 (ios_interfaces config for GigabitEthernet1/0/2, before speed) → **C. description: Configured for new VLAN**
- Blank 2 (ios_interfaces config for GigabitEthernet1/0/2, before speed) → **E. enabled: true**
- Blank 3 (ios_l2_interfaces config for GigabitEthernet1/0/2, before access) → **F. mode: access**
- Blank 4 (access VLAN) → **A. vlan: 25**

Explanation

The `ios_interfaces` task configures Layer 1 interface attributes, so it needs the description and enabled state alongside the specified speed and duplex. The `ios_l2_interfaces` task configures Layer 2 behavior: `mode: access` makes GigabitEthernet1/0/2 an access port, and `vlan: 25` assigns it to the new VLAN. The trunk option would conflict with the required access mode, while the remaining choices are not needed for the stated configuration.

https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/infra/interface-characteristics/interface-characteristics-configuration-guide.pdf
https://spq.xgslb-v3.cisco.com/c/en/us/td/docs/switches/lan/catalyst9200/software/release/26-x/command_reference/b_26x_9200_cr/layer_2_3_commands.pdf

✓ **Blank 1: description: Configured for new VLAN** description: Configured for new VLAN
This sets the requested interface description in the `ios_interfaces` configuration. It belongs with the Layer 1 interface attributes specified by the task.

https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/infra/interface-characteristics/interface-characteristics-configuration-guide.pdf

✓ **Blank 2: enabled: true** enabled: true
This ensures the interface is enabled, as required by the task's interface attributes. It is appropriate in the `ios_interfaces` configuration alongside the description, speed, and duplex.

https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/infra/interface-characteristics/interface-characteristics-configuration-guide.pdf

✓ **Blank 3: mode: access** mode: access
The port must be configured as an access port for the new VLAN segment. This setting belongs in the `ios_l2_interfaces` configuration before the access VLAN is specified.

https://spq.xgslb-v3.cisco.com/c/en/us/td/docs/switches/lan/catalyst9200/software/release/26-x/command_reference/b_26x_9200_cr/layer_2_3_commands.pdf

✓ **Blank 4: vlan: 25** vlan: 25
This assigns the access port to VLAN 25, the VLAN specified for the new segment. It is nested under `access` in the Layer 2 interface configuration.

https://spq.xgslb-v3.cisco.com/c/en/us/td/docs/switches/lan/catalyst9200/software/release/26-x/command_reference/b_26x_9200_cr/layer_2_3_commands.pdf

✗ **trunk: 25** trunk: 25
The scenario requires an access port, not a trunk port. Using a trunk VLAN setting would not meet the requested access-mode configuration.

https://spq.xgslb-v3.cisco.com/c/en/us/td/docs/switches/lan/catalyst9200/software/release/26-x/command_reference/b_26x_9200_cr/layer_2_3_commands.pdf

✗ **mode: switch** mode: switch
This is not the required Layer 2 setting for the port. The scenario specifically calls for access mode, which is expressed by `mode: access`.

✗ **shutdown: no** shutdown: no
This is not the correct option for the `ios_interfaces` schema shown in the playbook; the required enabled-state setting is `enabled: true`. The snippet also does not configure the port's access mode or VLAN.

Free demo · full exam: 549 questions

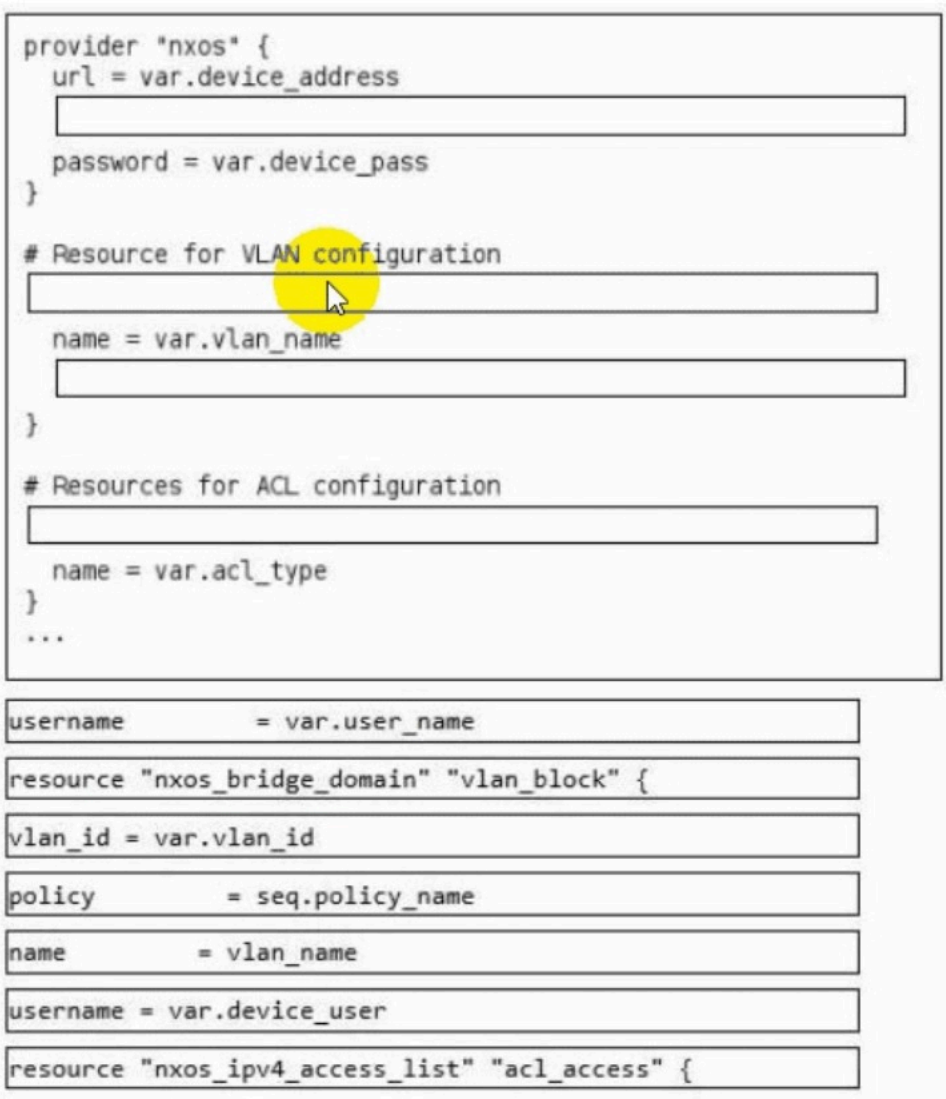
Question 7

Drag and drop · Domain: D1 Network Automation › 1.2 Construct a network automation solution with Terraform to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

A network automation engineer must provision infrastructure to support cloud edge system environments. The engineer must configure network devices to ensure secure and efficient data flow throughout the organization. They set up ACLs to enforce security policies and they implement logical segmentation Leveraging the Cisco IOS XE Software platform, the engineer uses Terraform to automate the deployment and management of these configurations Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Terraform configuration file Not all options are used

```
provider "nxos" {
  url = var.device_address
  [Blank 1]
  password = var.device_pass
}

# Resource for VLAN configuration
[Blank 2]
  name = var.vlan_name
  [Blank 3]
}

# Resources for ACL configuration
[Blank 4]
  name = var.acl_type
}
...
```


Free demo · full exam: 549 questions



Items:

- A. username = var.user_name
- B. resource "nxos_bridge_domain" "vlan_block" {
- C. vlan_id = var.vlan_id
- D. policy = seq.policy_name
- E. name = vlan_name
- F. username = var.device_user
- G. resource "nxos_ipv4_access_list" "acl_access" {

Place them on:

1. Blank 1 (provider "nxos"; after url = var.device_address and before password = var.device_pass)
2. Blank 2 (# Resource for VLAN configuration; resource declaration)
3. Blank 3 (VLAN resource; after name = var.vlan_name)
4. Blank 4 (# Resources for ACL configuration; resource declaration)

Answer

- Blank 1 (provider "nxos"; after url = var.device_address and before password = var.device_pass) → **F. username = var.device_user**
- Blank 2 (# Resource for VLAN configuration; resource declaration) → **B. resource "nxos_bridge_domain" "vlan_block" {**
- Blank 3 (VLAN resource; after name = var.vlan_name) → **C. vlan_id = var.vlan_id**
- Blank 4 (# Resources for ACL configuration; resource declaration) → **G. resource "nxos_ipv4_access_list" "acl_access" {**

Explanation

The NX-OS provider block needs the device username in addition to its address and password, so `username = var.device_user` fills Blank 1. The VLAN block is declared as an `nxos_bridge_domain` resource and includes both the VLAN name and `vlan_id`. The ACL block is declared as an `nxos_ipv4_access_list` resource, matching the requirement to configure an access list. These selections form the Terraform provider and resource structure shown in the question; the other snippets do not fit the relevant fields.

https://netascode.cisco.com/docs/data_models/nxos/security/ip_access_list
https://netascode.cisco.com/docs/start/nxos/first_steps/

✗ **Box 1: username = var.user_name** username = var.user_name
Blank 1 is the provider's username setting, but the supplied configuration uses `var.device_user` for that value. This snippet uses a different variable name and is not the matching choice for the shown provider block.

https://netascode.cisco.com/docs/start/nxos/first_steps/

✓ **Box 2: resource "nxos_bridge_domain" "vlan_block" {** resource "nxos_bridge_domain" "vlan_block" {
This declares the NX-OS bridge-domain resource used for the VLAN configuration block. Its resource type and local name match the VLAN-oriented section of the Terraform configuration.

https://netascode.cisco.com/docs/data_models/nxos/security/ip_access_list

✓ **Box 3: vlan_id = var.vlan_id** vlan_id = var.vlan_id
The VLAN resource needs the VLAN identifier as well as the `name` argument already shown. This snippet assigns that identifier from the `vlan_id` input variable.

https://netascode.cisco.com/docs/data_models/nxos/security/ip_access_list

✗ **Box 4: policy = seq.policy_name** policy = seq.policy_name
This is not the resource declaration needed for the ACL configuration block, and it does not match the ACL resource structure shown in the question. The correct choice for Blank 4 is the `nxos_ipv4_access_list` resource declaration.

https://netascode.cisco.com/docs/data_models/nxos/security/ip_access_list

✗ **Box 5: name = vlan_name** name = vlan_name
The VLAN block already has `name = var.vlan_name` in the displayed code. This snippet would duplicate that field and omits the required `vlan_id` assignment.

✓ **Box 6: username = var.device_user** username = var.device_user
The provider block requires a username setting alongside the device address and password. This snippet uses the device username variable shown by the intended configuration.

https://netascode.cisco.com/docs/start/nxos/first_steps/

✓ **Box 7: resource "nxos_ipv4_access_list" "acl_access" {** resource "nxos_ipv4_access_list" "acl_access" {
This declares an NX-OS IPv4 access-list resource, which fits the ACL configuration section and the scenario's requirement to enforce security policies with ACLs.

https://netascode.cisco.com/docs/data_models/nxos/security/ip_access_list

Free demo · full exam: 549 questions

Question 8

Single choice · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

A financial institution wants to implement automation and adhere to security and operational diversity requirements Its automation strategy emphasizes leveraging infrastructure as code implementation for seamless integration and using modular service components for enhanced flexibility Which approach meets the requirements?

A. statically defined command parser
B. global settings override per run
C. plug-in enabled orchestration layer
D. scheduled file transport scripts

Answer: C. plug-in enabled orchestration layer

Explanation

A plug-in-enabled orchestration layer best matches the requirement for modular service components: plug-ins can provide separable capabilities that the orchestration layer coordinates. This approach is more aligned with integrating automation components than a fixed command parser, a settings override, or standalone scheduled scripts. References [2] and [4] cover automation orchestration and service implementation, respectively.

https://developer.cisco.com/docs/automation-orchestration-intersight/
https://developer.cisco.com/docs/nso-guides-6.3/implementing-services/

✗ **A** statically defined command parser
A statically defined parser is a fixed interface for interpreting commands, not an orchestration approach built around modular service components. It does not directly address the requested flexibility or seamless integration of infrastructure-as-code automation.

✗ **B** global settings override per run
Overriding global settings for each run changes configuration, but it does not provide a modular orchestration layer. It therefore does not meet the stated emphasis on flexible service components and integrated automation.

✓ **C** plug-in enabled orchestration layer
An orchestration layer enabled by plug-ins can coordinate automation while allowing capabilities to be added as modular components. That makes it the best fit for the institution's requirements for flexibility and integration; orchestration and service implementation are covered in references [2] and [4].

https://developer.cisco.com/docs/automation-orchestration-intersight/
https://developer.cisco.com/docs/nso-guides-6.3/implementing-services/

✗ **D** scheduled file transport scripts
Scheduled scripts that transport files automate a narrow, time-based task, but do not constitute a flexible modular orchestration layer. They are not the approach described for integrating infrastructure-as-code implementation with modular services.

Free demo · full exam: 549 questions

Question 9

Drag and drop · Domain: D1 Network Automation › 1.4 Construct a network automation solution with Python to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

In a dynamic network environment, manually configuring devices is a significant bottleneck, which leads to inefficiencies and potential errors. To address the issue, a network engineer must develop a Python script to automate basic configuration tasks on Cisco IOS devices This initial script focuses on establishing connectivity and applying commands, and it uses print statements to provide immediate feedback on its execution Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Python script Not all options are used

```
from netmiko import ConnectHandler

def configure_cisco_device(device_params, config_commands):
    net_connect = None
    [Blank 1]
    net_connect = ConnectHandler(**device_params)
    [Blank 2]
    print("Sending configuration commands...")
    output = net_connect.send_config_set(config_commands)
    print("Configuration commands sent successfully.")
    print(f"Configuration output:\n{output}")
    [Blank 3]
    if net_connect:
        [Blank 4]
        net_connect.disconnect()

if __name__ == "__main__":
    device = {
        "device_type": "cisco_ios",
        "ip": "10.1.1.1",
        "username": "admin",
        "password": "cisco123"
    }
    config = [
        "interface GigabitEthernet0/1",
        "description Basic_Uplink",
        "no shutdown"
    ]
    print("\n--- Running Configuration Attempt ---")
    configure_cisco_device(device, config)
```


Free demo · full exam: 549 questions



```
from netmiko import ConnectHandler

def configure_cisco_device(device_params, config_commands):
    net_connect = None
    [                                                    ]
    net_connect = ConnectHandler(**device_params)
    [                                                    ]
    print("Sending configuration commands...")
    output = net_connect.send_config_set(config_commands)
    print("Configuration commands sent successfully.")
    print(f"Configuration output:\n{output}")
    [                                                    ]
    if net_connect:
        [                                                ]
        net_connect.disconnect()

if __name__ == "__main__":
    device = {
        "device_type": "cisco_ios",
        "ip": "10.1.1.1",
        "username": "admin",
        "password": "cisco123"
    }
    config = [
        "interface GigabitEthernet0/1",
        "description Basic_Uplink",
        "no shutdown"
    ]
    print("\n--- Running Configuration Attempt ---")
    configure_cisco_device(device, config)
```

```
print(f"Successfully connected to {device_params['ip']}")
print(f"Configuration complete on {device_params['ip']}.")
print(f"Configuration complete on {device['ip']}.")
print(f"Disconnecting from {device['ip']}")
print(f"Successfully connected to {device['ip']}")
print(f"Attempting to connect to device: {device_params['ip']}")
print(f"Disconnecting from {device_params['ip']}")
```

Items:

- A. print(f"Successfully connected to {device_params['ip']}")
- B. print(f"Configuration complete on {device_params['ip']}.")
- C. print(f"Configuration complete on {device['ip']}.")
- D. print(f"Disconnecting from {device['ip']}")
- E. print(f"Successfully connected to {device['ip']}")
- F. print(f"Attempting to connect to device: {device_params['ip']}")
- G. print(f"Disconnecting from {device_params['ip']}")

Place them on:

1. Blank 1 (after `net_connect = None`, before `net_connect = ConnectHandler(**device_params)`)
2. Blank 2 (after `net_connect = ConnectHandler(**device_params)`, before `print("Sending configuration commands...")`)
3. Blank 3 (after `print(f"Configuration output:\n{output}")`, before `if net_connect:`)
4. Blank 4 (inside `if net_connect:`, before `net_connect.disconnect()`)

Answer

- Blank 1 (after `net_connect = None`, before `net_connect = ConnectHandler(**device_params)`) → **F. print(f"Attempting to connect to device: {device_params['ip']}")**
- Blank 2 (after `net_connect = ConnectHandler(**device_params)`, before `print("Sending configuration commands...")`) → **A. print(f"Successfully connected to {device_params['ip']}")**
- Blank 3 (after `print(f"Configuration output:\n{output}")`, before `if net_connect:`) → **B. print(f"Configuration complete on {device_params['ip']}.")**
- Blank 4 (inside `if net_connect:`, before `net_connect.disconnect()`) → **G. print(f"Disconnecting from {device_params['ip']}")**

Explanation

The snippets should report each stage using the function's `device_params` argument, which contains the IP address available inside `configure_cisco_device`. Before connecting, the script prints an attempt message; after `ConnectHandler` succeeds, it reports the connection. It then reports configuration completion after sending and displaying the commands, and announces disconnection immediately before disconnecting. The options that refer to `device['ip']` use the variable from the separate main block rather than the function parameter.

✗ **Blank 1: print(f"Successfully connected to {device_params['ip']}")** print(f"Successfully connected to {device_params['ip']}")
This is a success message, so it belongs after `ConnectHandler` establishes the connection, not before the connection attempt. Blank 1 should announce that the script is attempting to connect.

✗ **Blank 2: print(f"Configuration complete on {device_params['ip']}.")** print(f"Configuration complete on {device_params['ip']}.")
This message reports completion of configuration, but Blank 2 occurs before the configuration commands are sent. It belongs after the output is displayed; Blank 2 should report a successful connection.

✗ **Blank 3: print(f"Configuration complete on {device['ip']}.")** print(f"Configuration complete on {device['ip']}.")
The configuration-complete message is appropriate for Blank 3, but `device` is defined in the main block, not as a parameter in `configure_cisco_device`. Use `device_params['ip']` so the message refers to the device available in the function.

✗ **Blank 4: print(f"Disconnecting from {device['ip']}")** print(f"Disconnecting from {device['ip']}")
A disconnection message belongs before `net_connect.disconnect()`, so its placement is appropriate. However, `device` is not the function's parameter; the snippet should use `device_params['ip']`.

✗ **Blank 2: print(f"Successfully connected to {device['ip']}")** print(f"Successfully connected to {device['ip']}")
Blank 2 is the right point for a connection-success message, but `device` is defined outside the function. Use the function parameter `device_params` to access the IP address here.

✓ **Blank 1: print(f"Attempting to connect to device: {device_params['ip']}")** print(f"Attempting to connect to device: {device_params['ip']}")
This message announces the connection attempt before `ConnectHandler(**device_params)` runs. It also uses `device_params`, which is available inside the function and contains the target IP.

✓ **Blank 4: print(f"Disconnecting from {device_params['ip']}")** print(f"Disconnecting from {device_params['ip']}")
This reports that the script is disconnecting immediately before `net_connect.disconnect()` is called. The function parameter provides the correct device IP for the message.

Free demo · full exam: 549 questions

Question 10

Drag and drop · Domain: D1 Network Automation › 1.4 Construct a network automation solution with Python to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

A network engineer must complete a Python script that can reliably connect to Cisco IOS devices and apply configurations programmatically to incorporate essential error handling. Drag and drop the code from the bottom onto the boxes in the code to construct the Python script. Not all options are used.

```
import logging
[Blank 1]
from netmiko.exceptions import NetmikoTimeoutException, NetmikoAuthenticationException, Net
logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
def configure_cisco_device(device_params, config_commands):
    [Blank 2]
    try:
        logging.info(f"Connecting {device_params['ip']}...")
        net_connect = ConnectHandler(**device_params)
        net_connect.send_config_set(config_commands)
        logging.info(f"Config successful on {device_params['ip']}.")
        return True
    except NetmikoTimeoutException:
        logging.error(f"Timeout {device_params['ip']}. [1, 9]")
    except NetmikoAuthenticationException:
        logging.error(f"Auth failed {device_params['ip']}. [1, 5]")
    except NetmikoException as e:
        logging.error(f"Netmiko error {device_params['ip']}: {e}. [2]")
    except Exception as e:
        logging.critical(f"Unexpected error {device_params['ip']}: {e}")
    finally:
        [Blank 3]
            net_connect.disconnect()
        logging.info(f"Disconnected {device_params['ip']}.")
    return False
if __name__ == "__main__":
    [Blank 4]
        "device_type": "cisco_ios",
        "ip": "10.1.1.1",
        "username": "admin",
        "password": "cisco123"
    }
    config = ["interface GigabitEthernet0/1", "description Shortest Uplink", "no shutdown"]
    configure_cisco_device(device, config)
```


Free demo · full exam: 549 questions



```
import logging
[ ]
from netmiko.exceptions import NetmikoTimeoutException, NetmikoAuthenticationException, NetmikoException
logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
def configure_cisco_device(device_params, config_commands):
    [ ]
    try:
        logging.info(f"Connecting {device_params['ip']}...")
        net_connect = ConnectHandler(**device_params)
        net_connect.send_config_set(config_commands)
        logging.info(f"Config successful on {device_params['ip']}.")
        return True
    except NetmikoTimeoutException:
        logging.error(f"Timeout {device_params['ip']}. [1, 9]")
    except NetmikoAuthenticationException:
        logging.error(f"Auth failed {device_params['ip']}. [1, 5]")
    except NetmikoException as e:
        logging.error(f"Netmiko error {device_params['ip']}: {e}. [2]")
    except Exception as e:
        logging.critical(f"Unexpected error {device_params['ip']}: {e}")
    finally:
        [ ]
            net_connect.disconnect()
            logging.info(f"Disconnected {device_params['ip']}.")
    return False
if __name__ == "__main__":
    [ ]
        "device_type": "cisco_ios",
        "ip": "10.1.1.1",
        "username": "admin",
        "password": "cisco123"
    }
    config = ["interface GigabitEthernet0/1", "description Shortest_Uplink", "no shutdown"]
    configure_cisco_device(device, config)
```

```
try net_connect:
from netmiko import Handler
net_connect = True
from netmiko import ConnectHandler
device = {
if net_connect:
net_connect = None
```

Items:

- A. try net_connect:
- B. from netmiko import Handler
- C. net_connect = True
- D. from netmiko import ConnectHandler
- E. device = {
- F. if net_connect:
- G. net_connect = None

Place them on:

1. Blank 1 (after `import logging`, before `from netmiko.exceptions import ...`)
2. Blank 2 (inside `configure_cisco_device`, immediately before `try:`)
3. Blank 3 (inside `finally:`, immediately before `net_connect.disconnect()`)
4. Blank 4 (inside `if __name__ == "__main__":`, before the device dictionary contents)

Answer

- Blank 1 (after `import logging`, before `from netmiko.exceptions import ...`) → **D. from netmiko import ConnectHandler**
- Blank 2 (inside `configure_cisco_device`, immediately before `try:`) → **G. net_connect = None**
- Blank 3 (inside `finally:`, immediately before `net_connect.disconnect()`) → **F. if net_connect:**
- Blank 4 (inside `if __name__ == "__main__":`, before the device dictionary contents) → **E. device = {**

Dumpsgate

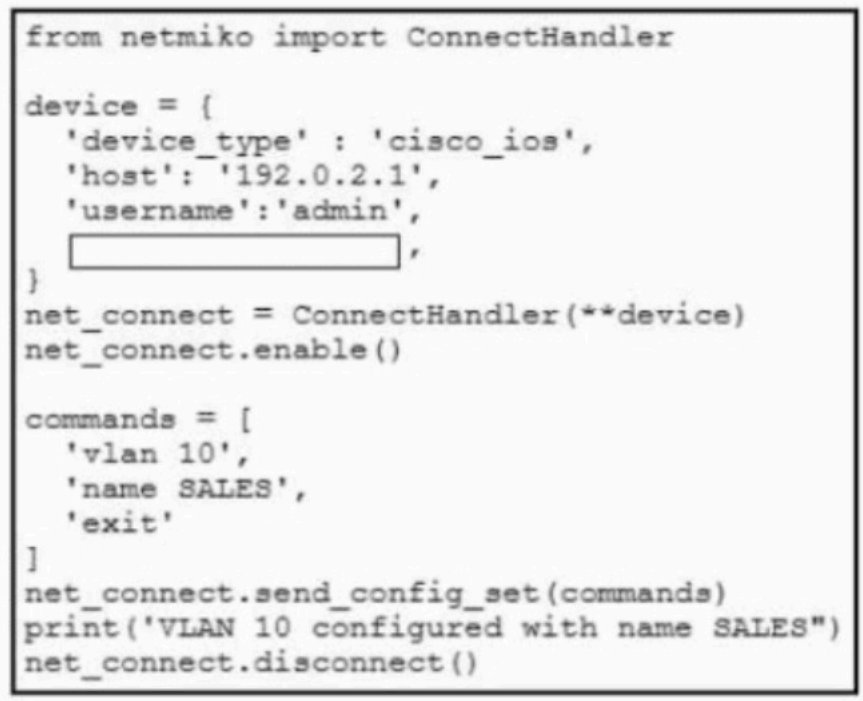
Question 11

Single choice · Domain: D1 Network Automation › 1.4 Construct a network automation solution with Python to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Refer to the exhibit.

```
from netmiko import ConnectHandler

device = {
    'device_type': 'cisco_ios',
    'host': '192.0.2.1',
    'username':'admin',
    [Blank 1],
}
net_connect = ConnectHandler(**device)
net_connect.enable()

commands = [
    'vlan 10',
    'name SALES',
    'exit'
]
net_connect.send_config_set(commands)
print("VLAN 10 configured with name SALES")
net_connect.disconnect()
```

Refer to the exhibit An engineer is tasked with creating a Python script to configure a VLAN on ios device Which line of code must be added to the box where the code is missing to complete the Python request?



A. 'enable' 7\P0Dgth5x2vw7fyTz5b2JiX2kuTys3ez6A8GVM'
B. 'password' 'AP0Dgth5x2vw7fyTz5b2JiX2kuTys3ez6A8GVM'
C. 'hash' 'AP0Dgth5x2vw7fyTz5b2JiX2kuTys3ez6A8GVM'
D. 'Authorization' 'AP0Dgth5x2vw7fyTz5b2JiX2kuTys3ez6A8GVM'

Answer: B. 'password' 'AP0Dgth5x2vw7fyTz5b2JiX2kuTys3ez6A8GVM'

Explanation

The missing dictionary entry must provide the device password used by Netmiko to authenticate to the Cisco IOS device. Option B supplies a `password` key with the credential, completing the connection details before the script enables privileged mode and sends the VLAN configuration commands. [1]

https://community.cisco.com/kxiwq67737/attachments/kxiwq67737/documents-general/69/1/Webcast_Python-mar11-2025.pdf

✗ **A** 'enable' 7\P0Dgth5x2vw7fyTz5b2JiX2kuTys3ez6A8GVM'
This is not the needed connection credential field in the device dictionary. The script already calls `net_connect.enable()` to enter privileged EXEC mode; the missing entry is the login password.

https://community.cisco.com/kxiwq67737/attachments/kxiwq67737/documents-general/69/1/Webcast_Python-mar11-2025.pdf

✓ **B** 'password' 'AP0Dgth5x2vw7fyTz5b2JiX2kuTys3ez6A8GVM'
The device dictionary needs a `password` entry so `ConnectHandler(**device)` can authenticate to the IOS device using the supplied username. This allows the script to establish a session before configuring VLAN 10.

https://community.cisco.com/kxiwq67737/attachments/kxiwq67737/documents-general/69/1/Webcast_Python-mar11-2025.pdf

✗ **C** 'hash' 'AP0Dgth5x2vw7fyTz5b2JiX2kuTys3ez6A8GVM'
A `hash` entry is not the password field expected for the Netmiko device connection shown. `ConnectHandler` needs the device's login password in the dictionary.

https://community.cisco.com/kxiwq67737/attachments/kxiwq67737/documents-general/69/1/Webcast_Python-mar11-2025.pdf

✗ **D** 'Authorization' 'AP0Dgth5x2vw7fyTz5b2JiX2kuTys3ez6A8GVM'
An `Authorization` entry is not the connection password parameter required by this Netmiko script. The dictionary should provide the credential under the `password` key.

https://community.cisco.com/kxiwq67737/attachments/kxiwq67737/documents-general/69/1/Webcast_Python-mar11-2025.pdf

Question 12

Single choice · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

A service provider must improve deployment velocity and maintain customization across customer environments. The requirements include GUI-based logic composition for network element interaction and scalable configuration modifications to ensure adaptability Which approach meets the requirements?

A. concurrent template application at scale
B. role-locked data views
C. serialized deployment trigger queues
D. isolated worker thread synchronization

Answer: A. concurrent template application at scale

Explanation

Concurrent application of configurable templates fits both requirements: templates provide reusable, customizable configuration for different customer environments, while applying them concurrently supports deployment at scale. This is the approach among the options that addresses configuration customization as well as deployment velocity.

https://nso-docs.cisco.com/guides/development/advanced-development/scaling-and-performance-optimization
https://developer.cisco.com/docs/nso-guides-5.7/templates/
https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/crosswork-network-controller/7-1/Articles/Solution-Workflow/custom-templates.pdf

✓ **A** concurrent template application at scale
Templates can capture reusable configuration that is adapted for different customer environments, supporting scalable configuration changes. Applying them concurrently helps improve deployment velocity across multiple targets; this aligns with the template and scaling guidance.

https://nso-docs.cisco.com/guides/development/advanced-development/scaling-and-performance-optimization
https://developer.cisco.com/docs/nso-guides-5.7/templates/
https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/crosswork-network-controller/7-1/Articles/Solution-Workflow/custom-templates.pdf

✗ **B** role-locked data views
Restricting data views by role concerns access and visibility, not composing network-element interaction logic or scaling configuration changes across customer environments.

✗ **C** serialized deployment trigger queues
A serialized queue processes deployment triggers in sequence, which does not directly provide customizable configuration logic and can limit deployment velocity compared with concurrent template application.

✗ **D** isolated worker thread synchronization
Worker-thread synchronization is a concurrency implementation concern, not a method for composing reusable, customer-specific network configuration. It does not by itself meet the stated customization requirement.

Question 13

Drag and drop · Domain: D1 Network Automation › 1.1 Construct a network automation solution with Ansible to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

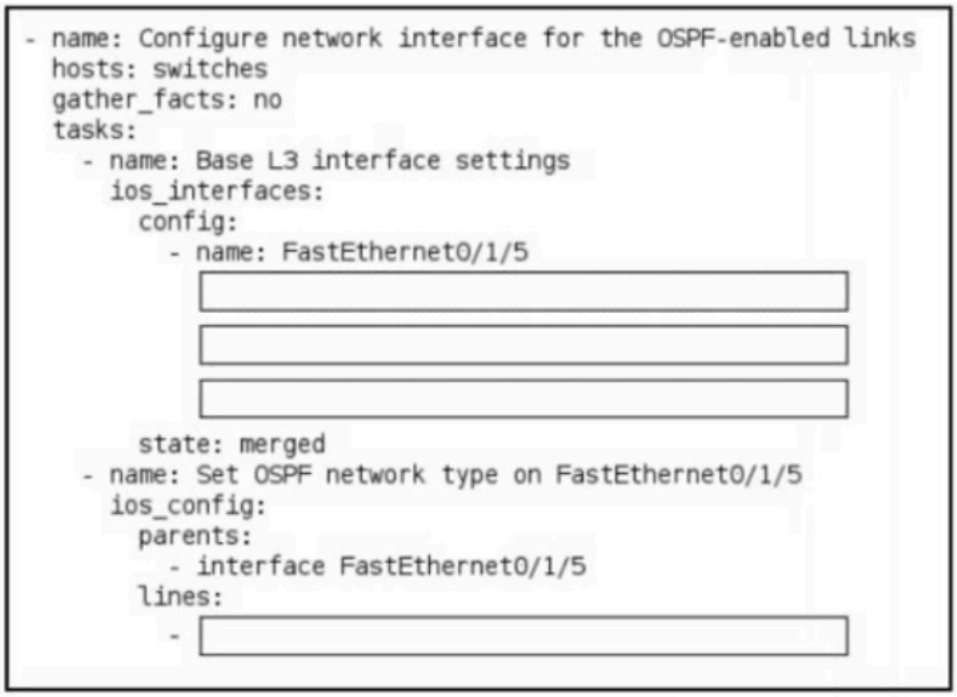
A network automation engineer is building an Ansible playbook to deploy interface configurations for OSPF-enabled links on designated switches. The configuration must set all required parameters for FastEthernet0/1/5 without missing any steps. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Ansible playbook. Not all options are used.

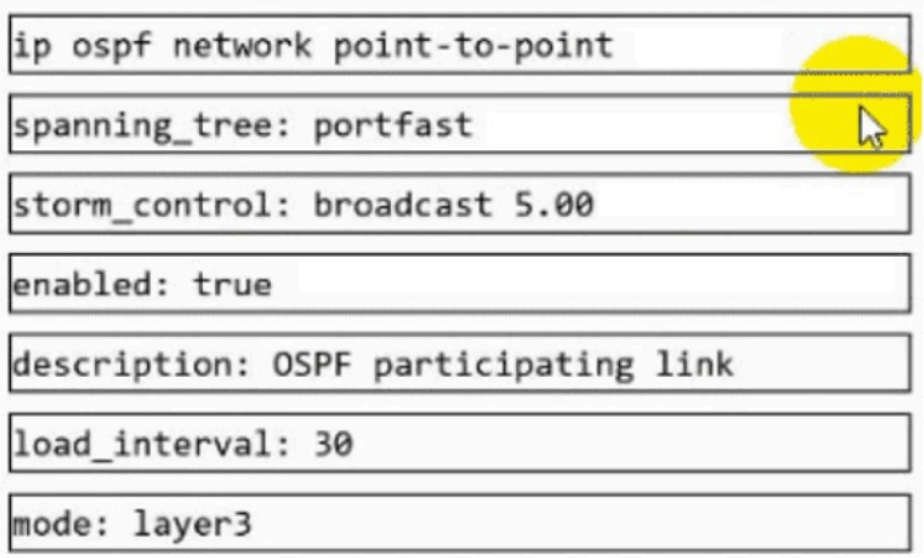
```
- name: Configure network interface for the OSPF-enabled links
  hosts: switches
  gather_facts: no
  tasks:
    - name: Base L3 interface settings
      ios_interfaces:
        config:
          - name: FastEthernet0/1/5
            [Blank 1]
            [Blank 2]
            [Blank 3]
        state: merged
    - name: Set OSPF network type on FastEthernet0/1/5
      ios_config:
        parents:
          - interface FastEthernet0/1/5
        lines:
          - [Blank 4]
```

Options:

- ip ospf network point-to-point
- spanning_tree: portfast
- storm_control: broadcast 5.00
- enabled: true
- description: OSPF participating link
- load_interval: 30
- mode: layer3

Free demo · full exam: 549 questions





Items:

- A. ip ospf network point-to-point
- B. spanning_tree: portfast
- C. storm_control: broadcast 5.00
- D. enabled: true
- E. description: OSPF participating link
- F. load_interval: 30
- G. mode: layer3

Place them on:

1. Blank 1 (under name: FastEthernet0/1/5 in ios_interfaces config)
2. Blank 2 (under name: FastEthernet0/1/5 in ios_interfaces config)
3. Blank 3 (under name: FastEthernet0/1/5 in ios_interfaces config)
4. Blank 4 (under lines for interface FastEthernet0/1/5 in ios_config)

Dumpsgate

Question 14

Single choice · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

Multiple business units are collaborating to automate device provisioning across distributed branches. The solution must accommodate different levels of expertise with template-driven configuration and orchestration features. The solution also must work with nonstandard devices to ensure business process consistency. Which approach meets the requirements?

A. syntax highlighting in terminal editors
B. extensible device connector support
C. isolated static inventory per user
D. command injection safeguards for scripts

Answer: B. extensible device connector support

Explanation

Extensible device connector support is the best fit because it lets the provisioning solution work with nonstandard devices as well as the devices supported out of the box. That broader device compatibility helps the business units apply consistent provisioning processes across distributed branches. Template-driven configuration and orchestration can then support users with different levels of expertise.

https://team-development.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/catalyst-center/2-3-7/user_guide/b_cisco_catalyst_center_user_guide_237/b_cisco_dna_center_ug_2_3_7_chapter_01000.html
https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/catalyst-center/3-3-x/user-guide/cisco-catalyst-center-user-guide-3-3-x/b_cisco_catalyst_center_ug_3_1_x_chapter_01000.html

✗ **A** syntax highlighting in terminal editors
Syntax highlighting can help people read or edit scripts, but it does not provide device connectivity or support provisioning nonstandard devices. It therefore does not meet the cross-device consistency requirement.

✓ **B** extensible device connector support
Extensible connectors allow the solution to interact with nonstandard devices, addressing the requirement to include them in automated provisioning. This compatibility helps the business units apply their provisioning processes consistently across different device types.

https://team-development.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/catalyst-center/2-3-7/user_guide/b_cisco_catalyst_center_user_guide_237/b_cisco_dna_center_ug_2_3_7_chapter_01000.html
https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/catalyst-center/3-3-x/user-guide/cisco-catalyst-center-user-guide-3-3-x/b_cisco_catalyst_center_ug_3_1_x_chapter_01000.html

✗ **C** isolated static inventory per user
Separate static inventories would make it harder for multiple business units to collaborate on provisioning across branches. They do not provide the required support for nonstandard devices or template-driven orchestration.

✗ **D** command injection safeguards for scripts
Safeguards help prevent unsafe script execution, but they do not add device compatibility or provide the template-driven configuration and orchestration described in the scenario.

Free demo · full exam: 549 questions

Question 15

Drag and drop · Domain: D1 Network Automation › 1.2 Construct a network automation solution with Terraform to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

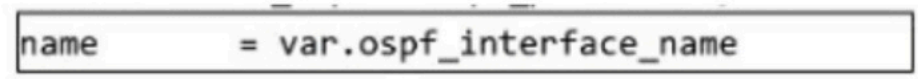
A network automation engineer must use Terraform to configure several network features for the corporate router, including essential interface details and an OSPF area as specified by the backbone segment. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Terraform configuration file. Not all options are used.

```
terraform {
  required_providers {
    [Blank 1]
    source  = "CiscoDevNet/iosxe"
    version = "~> 0.5"
  }
}

provider "iosxe" {
  [Blank 2]
  username = "your_username"
  password = "your_password"
}

# OSPF process definition
resource "iosxe_ospf" "ospf_process" {
  [Blank 3]
}

# Bind OSPF to an interface
resource "iosxe_interface_ospf" "ospf_on_interface" {
  [Blank 4]
  process_id = iosxe_ospf.ospf_process.process_id
  area       = var.ospf_area
}
```

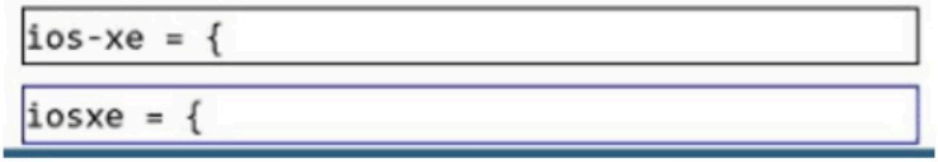

Free demo · full exam: 549 questions



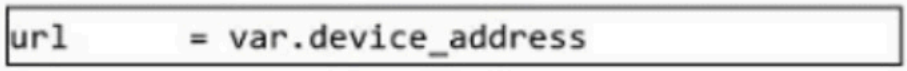
```
terraform {
  required_providers {
    [                                    ]
      source  = "CiscoDevNet/iosxe"
      version = "~> 0.5"
    }
  }
}

provider "iosxe" {
  [                                      ]
  username = "your_username"
  password = "your_password"
}

# OSPF process definition
resource "iosxe_ospf" "ospf_process" {
  [                                      ]
}

# Bind OSPF to an interface
resource "iosxe_interface_ospf" "ospf_on_interface" {
  [                                      ]
  process_id = iosxe_ospf.ospf_process.process_id
  area       = var.ospf_area
}
```



```
name        = var.ospf_interface_name
```



```
ios-xe = {
iosxe = {
```



```
url       = var.device_address
```



```
process_id = ospf_process_id
```


Free demo · full exam: 549 questions

Items:

- A. host = device_address
- B. process_id = var.ospf_process_id
- C. name = var.ospf_interface_name
- D. ios-xe = {
- E. iosxe = {
- F. url = var.device_address
- G. process_id = ospf_process_id

Place them on:

1. Blank 1 (inside required_providers)
2. Blank 2 (inside provider "iosxe")
3. Blank 3 (inside resource "iosxe_ospf" "ospf_process")
4. Blank 4 (inside resource "iosxe_interface_ospf" "ospf_on_interface")

Answer

- Blank 1 (inside required_providers) → **E. iosxe = {**
- Blank 2 (inside provider "iosxe") → **F. url = var.device_address**
- Blank 3 (inside resource "iosxe_ospf" "ospf_process") → **B. process_id = var.ospf_process_id**
- Blank 4 (inside resource "iosxe_interface_ospf" "ospf_on_interface") → **C. name = var.ospf_interface_name**

Explanation

The Terraform configuration needs the provider-local name `iosxe` under `required_providers`, and the provider block uses `url` to identify the device address. The OSPF process resource takes its process ID from `var.ospf_process_id`, while the interface OSPF resource needs the interface name supplied by `var.ospf_interface_name`. The remaining snippets either use the wrong field or do not match the required provider name or resource values.

https://developer.cisco.com/codeexchange/github/repo/muhammad-rafi/terraform-iosxe/
https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✗ **host = device_address** host = device_address
This does not provide the provider's device URL in the form used by the configuration. The provider block needs `url = var.device_address` to set the router address from the Terraform variable.

https://developer.cisco.com/codeexchange/github/repo/muhammad-rafi/terraform-iosxe/

✓ **process_id = var.ospf_process_id** process_id = var.ospf_process_id
This sets the OSPF process resource's ID from the configured Terraform variable. The interface resource then references that process resource's `process_id`.

https://developer.cisco.com/codeexchange/github/repo/muhammad-rafi/terraform-iosxe/
https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✓ **name = var.ospf_interface_name** name = var.ospf_interface_name
The OSPF interface resource must identify which interface to bind to the OSPF process. This assigns its name from the interface-name variable, while the resource's area is specified separately.

https://developer.cisco.com/codeexchange/github/repo/muhammad-rafi/terraform-iosxe/
https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✗ **ios-xe = {** ios-xe = {
The required provider entry must use the local name `iosxe`, matching the provider block and resource namespace. `ios-xe` is a different name and does not supply the expected provider entry.

https://developer.cisco.com/codeexchange/github/repo/muhammad-rafi/terraform-iosxe/

✓ **iosxe = {** iosxe = {

This declares the required provider using the local name `iosxe`, which matches the provider block and the `iosxe_ospf` and `iosxe_interface_ospf` resources.

https://developer.cisco.com/codeexchange/github/repo/muhammad-rafi/terraform-iosxe/

✓ **url = var.device_address** url = var.device_address
This configures the provider with the router address stored in `var.device_address`. It belongs inside the `provider "iosxe"` block.

https://developer.cisco.com/codeexchange/github/repo/muhammad-rafi/terraform-iosxe/

✗ **process_id = ospf_process_id** process_id = ospf_process_id
This does not use the configured Terraform variable `var.ospf_process_id` required for the OSPF process resource. It is also not the interface name needed in the interface OSPF resource.

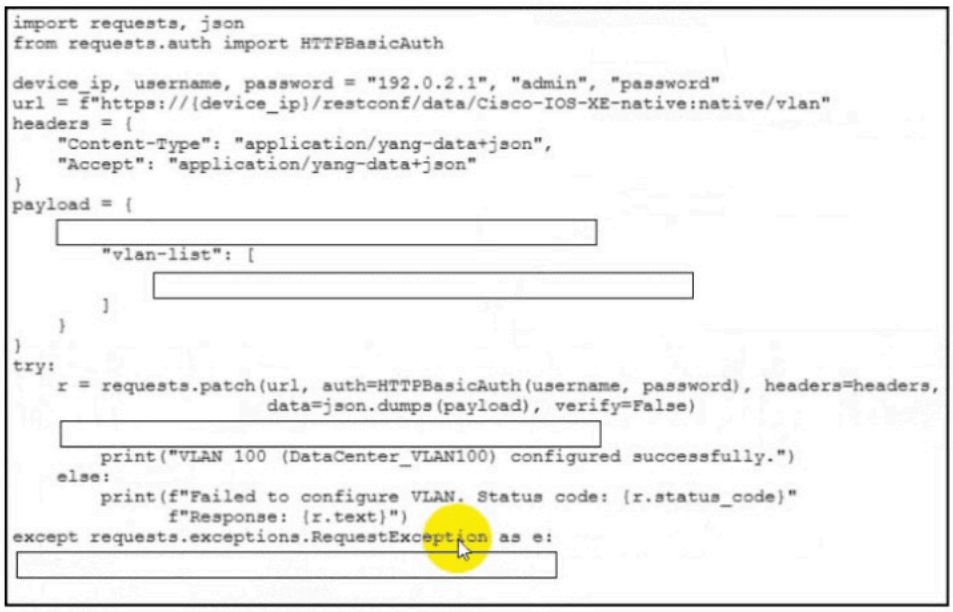
https://developer.cisco.com/codeexchange/github/repo/muhammad-rafi/terraform-iosxe/
https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

Question 16

Drag and drop · Domain: D1 Network Automation › 1.3 Construct a network automation solution with RESTCONF (RFC 8040), given the YANG model, to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Refer to the exhibit. A network engineer must use a RESTCONF-based Python script to roll out a new network service in a data center that requires VLAN provisioning on devices. The engineer must complete the Python script that updates the device configuration by sending a PATCH request with the appropriate VLAN details. The script leverages the Cisco IOS XE native YANG model, specifically the Cisco-IOS-XE-native:native/vlan container, to structure the VLAN configuration payload. The engineer must ensure that the script aligns with this YANG model to successfully provision VLAN 100 named DataCenter_VLAN100 on the target device Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Python script. Not all options are used

```
import requests, json
from requests.auth import HTTPBasicAuth

device_ip, username, password = "192.0.2.1", "admin", "password"
url = f"https://{device_ip}/restconf/data/Cisco-IOS-XE-native:native/vlan"
headers = {
    "Content-Type": "application/yang-data+json",
    "Accept": "application/yang-data+json"
}
payload = {
    [Blank 1]
        "vlan-list": [
            [Blank 2]
        ]
    }
}
try:
    r = requests.patch(url, auth=HTTPBasicAuth(username, password), headers=headers,
                       data=json.dumps(payload), verify=False)
    [Blank 3]
        print("VLAN 100 (DataCenter_VLAN100) configured successfully.")
    else:
        print(f"Failed to configure VLAN. Status code: {r.status_code}"
              f"Response: {r.text}")
except requests.exceptions.RequestException as e:
    [Blank 4]
```



```
import requests, json
from requests.auth import HTTPBasicAuth

device_ip, username, password = "192.0.2.1", "admin", "password"
url = f"https://{device_ip}/restconf/data/Cisco-IOS-XE-native:native/vlan"
headers = {
    "Content-Type": "application/yang-data+json",
    "Accept": "application/yang-data+json"
}
payload = {
    [                                                    ]
        "vlan-list": [
            [                                                    ]
        ]
    }
}
try:
    r = requests.patch(url, auth=HTTPBasicAuth(username, password), headers=headers,
                       data=json.dumps(payload), verify=False)
    [                                                    ]
        print("VLAN 100 (DataCenter_VLAN100) configured successfully.")
    else:
        print(f"Failed to configure VLAN. Status code: {r.status_code}"
              f"Response: {r.text}")
except requests.exceptions.RequestException as e:
[                                                    ]
```

```
if r.status_code in (401, 403):
print(f"Error communicating with device: {e}")
if r.status_code in (200, 204):
{"num": 100, "site": "DataCenter_VLAN100"}
"Cisco-IOS-XE-native:vlan": {
"Cisco-IOS-XE:vlan": {
{"id": 100, "name": "DataCenter_VLAN100"}
```

Items:

- A. if r.status_code in (401, 403):
- B. print(f"Error communicating with device: {e}")
- C. if r.status_code in (200, 204):
- D. {"num": 100, "site": "DataCenter_VLAN100"}
- E. "Cisco-IOS-XE-native:vlan": {
- F. "Cisco-IOS-XE:vlan": (
- G. {"id": 100, "name": "DataCenter_VLAN100"}

Place them on:

1. Blank 1 (inside `payload = {`)
2. Blank 2 (inside the `"vlan-list": [` array)
3. Blank 3 (after the PATCH request and before the success `print` statement)
4. Blank 4 (inside `except requests.exceptions.RequestException as e:`)

Answer

- Blank 1 (inside `payload = {`) → **E. "Cisco-IOS-XE-native:vlan": {**
- Blank 2 (inside the `"vlan-list": [` array) → **G. {"id": 100, "name": "DataCenter_VLAN100"}**
- Blank 3 (after the PATCH request and before `print("VLAN 100 (DataCenter_VLAN100) configured successfully.")`) → **C. if r.status_code in (200, 204):**
- Blank 4 (inside `except requests.exceptions.RequestException as e:`) → **B. print(f"Error communicating with device: {e}")**

Free demo · full exam: 549 questions

Dumpsgate

Question 17

Single choice · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

A development team is preparing to deploy a next-generation network architecture for an organization that wants to improve efficiency. The team is evaluating various solutions because zero-touch provisioning capability is required for the project, and the timeline demands rapid implementation. After the project needs are discussed, the organization decides to prioritize visual workflow design. Which technology meets the network automation needs of the deployment?

A. encrypted traffic analysis engine
B. drag-and-drop automation platform
C. high-performance virtual network appliance
D. centralized traffic monitoring dashboard

Answer: B. drag-and-drop automation platform

Explanation

The drag-and-drop automation platform best matches the organization's priority of visual workflow design while enabling network tasks to be automated for rapid deployment. Zero-touch provisioning is specifically an onboarding and provisioning workflow, which can be automated rather than configured manually device by device. References [3] and [4] support workflow-based network automation and automated ZTP provisioning.

https://www.cisco.com/c/en/us/products/collateral/networking/software/workflows/network-auto-cloud-workflows-faq.pdf
https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/crosswork-network-controller/7-2/Solution-Workflow-Guide/bk-crosswork-network-controller-7-2-solution-workflow-guide/m-automation-of-onboarding-and-provisioning-of-ios-xr-devices-using-ztp.html

✗ **A** encrypted traffic analysis engine
An encrypted traffic analysis engine is intended to analyze network traffic, not visually design deployment workflows or automate zero-touch provisioning. It therefore does not meet the stated automation and rapid-implementation requirements.

✓ **B** drag-and-drop automation platform
A drag-and-drop platform directly supports the organization's priority of visual workflow design. Its workflows can automate provisioning tasks, including ZTP-related onboarding, reducing manual setup and supporting rapid implementation.

https://www.cisco.com/c/en/us/products/collateral/networking/software/workflows/network-auto-cloud-workflows-faq.pdf
https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/crosswork-network-controller/7-2/Solution-Workflow-Guide/bk-crosswork-network-controller-7-2-solution-workflow-guide/m-automation-of-onboarding-and-provisioning-of-ios-xr-devices-using-ztp.html

✗ **C** high-performance virtual network appliance
A virtual network appliance provides network functionality, but its description does not indicate visual workflow design or automated zero-touch provisioning. High performance alone does not address the project's specific automation requirements.

✗ **D** centralized traffic monitoring dashboard
A centralized traffic monitoring dashboard is suited to observing network activity, not designing automation workflows or provisioning devices automatically. Monitoring does not satisfy the stated need for visual automation and rapid deployment.

Free demo · full exam: 549 questions

Question 18

Drag and drop · Domain: D1 Network Automation › 1.2 Construct a network automation solution with Terraform to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

A network automation engineer must use Terraform to configure several network features for the data center gateway, including essential interface details and an OSPF area as specified by the isolated subnet. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Terraform configuration file. Not all options are used.

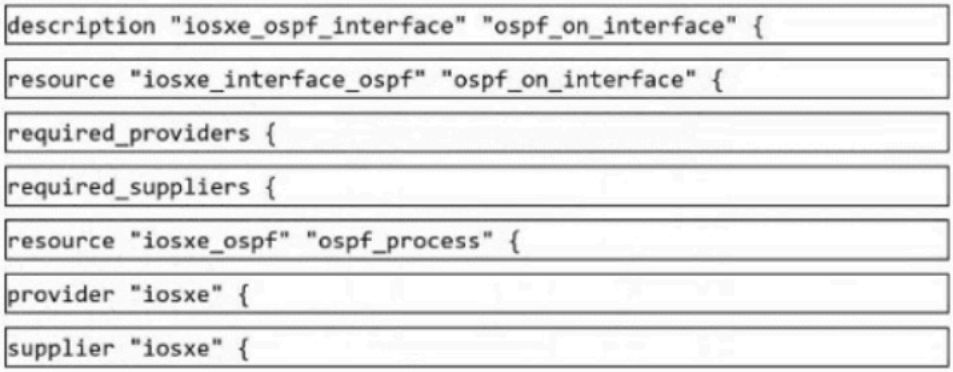
```
terraform {
  [Blank 1]
  iosxe = {
    source  = "CiscoDevNet/iosxe"
    version = "~> 0.5"
  }
}

[Blank 2]
  url      = var.device_address
  username = "your_username"
  password = "your_password"
}

# OSPF process definition
[Blank 3]
  process_id = var.ospf_process_id
}

# Bind OSPF to an interface
[Blank 4]
  name      = var.ospf_interface_name
  process_id = iosxe_ospf.ospf_process.process_id
  area      = var.ospf_area
}
```

Drag items:

- description "iosxe_ospf_interface" "ospf_on_interface" {
- resource "iosxe_interface_ospf" "ospf_on_interface" {
- required_providers {
- required_suppliers {
- resource "iosxe_ospf" "ospf_process" {
- provider "iosxe" {
- supplier "iosxe" {

Free demo · full exam: 549 questions





Items:

- A. description "iosxe_ospf_interface" "ospf_on_interface" {
- B. resource "iosxe_interface_ospf" "ospf_on_interface" {
- C. required_providers {
- D. required_suppliers {
- E. resource "iosxe_ospf" "ospf_process" {
- F. provider "iosxe" {
- G. supplier "iosxe" {

Place them on:

1. Blank 1 (inside the terraform block, before the iosxe provider configuration)
2. Blank 2 (before the url, username, and password settings)
3. Blank 3 (OSPF process definition, before process_id)
4. Blank 4 (bind OSPF to an interface, before name)

Answer

- Blank 1 (inside the terraform block, before the iosxe provider configuration) → **C. required_providers {**
- Blank 2 (before the url, username, and password settings) → **F. provider "iosxe" {**
- Blank 3 (OSPF process definition, before process_id) → **E. resource "iosxe_ospf" "ospf_process" {**
- Blank 4 (bind OSPF to an interface, before name) → **B. resource "iosxe_interface_ospf" "ospf_on_interface" {**

Explanation

The Terraform block needs `required_providers` to declare the IOS XE provider before its source and version configuration. The device connection settings belong inside a `provider "iosxe"` block, while the OSPF process and its interface binding are separate resources with the respective resource types shown. These selections construct the required configuration using the provided process ID, interface name, and OSPF area. [2]

https://netascode.cisco.com/docs/start/iosxe/introduction/

✓ **Box 1: required_providers {** required_providers {
This is the Terraform block used to declare required providers, matching the `iosxe` provider source and version configuration that follows. [2]

https://netascode.cisco.com/docs/start/iosxe/introduction/

✓ **Box 2: provider "iosxe" {** provider "iosxe" {
The URL, username, and password configure the IOS XE provider connection to the device, so they belong in this provider block. [2]

https://netascode.cisco.com/docs/start/iosxe/introduction/

✓ **Box 3: resource "iosxe_ospf" "ospf_process" {** resource "iosxe_ospf" "ospf_process" {
This declares the IOS XE OSPF process resource, which is where the supplied `process_id` is configured. [10]

https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✓ **Box 4: resource "iosxe_interface_ospf" "ospf_on_interface" {** resource "iosxe_interface_ospf" "ospf_on_interface" {
This resource binds OSPF to an interface; its body supplies the interface name, process ID reference, and OSPF area. [10]

https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✗ **required_suppliers {** required_suppliers {
This is not the Terraform block for declaring provider dependencies. The `iosxe` provider source and version shown in the configuration require `required_providers` instead.

✗ **description "iosxe_ospf_interface" "ospf_on_interface" {** description "iosxe_ospf_interface" "ospf_on_interface" {
A `description` block does not declare the interface OSPF resource needed to apply the interface name, process ID, and area. Use the `iosxe_interface_ospf` resource block.

✗ **supplier "iosxe" {** supplier "iosxe" {
This is not the Terraform provider configuration block that accepts the device URL and credentials. Those settings belong under `provider "iosxe"`.

✓ **resource "iosxe_ospf" "ospf_process" {** resource "iosxe_ospf" "ospf_process" {
The OSPF process definition needs a resource block, and this item names the IOS XE OSPF resource and its `ospf_process` instance.

https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

✓ **provider "iosxe" {** provider "iosxe" {
This block configures the IOS XE provider connection using the device address and login credentials

Free demo · full exam: 549 questions

specified in the body.

https://netascode.cisco.com/docs/start/iosxe/introduction/

✓ **resource "iosxe_interface_ospf" "ospf_on_interface" {** resource "iosxe_interface_ospf" "ospf_on_interface" {
This is the resource required to attach OSPF settings to the specified interface, including the process ID and area.

https://netascode.cisco.com/docs/data_models/iosxe/device/ospf/

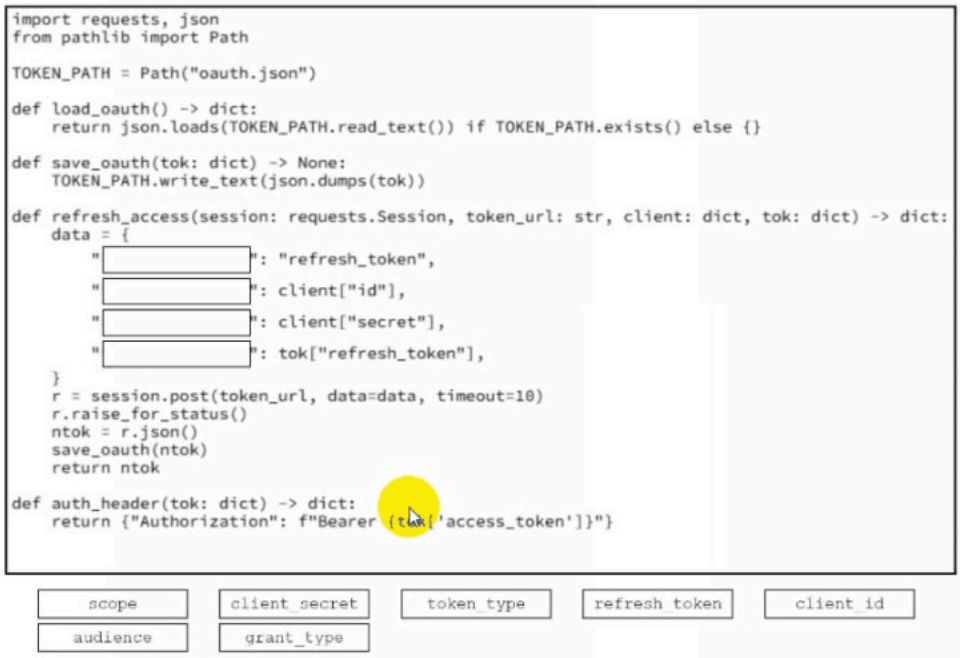
Question 19

Drag and drop · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

An engineer must implement an OAuth2 refresh flow against a controller that returns refresh and access tokens. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.

```
import requests, json
from pathlib import Path

TOKEN_PATH = Path("oauth.json")

def load_oauth() -> dict:
    return json.loads(TOKEN_PATH.read_text()) if TOKEN_PATH.exists() else {}

def save_oauth(tok: dict) -> None:
    TOKEN_PATH.write_text(json.dumps(tok))

def refresh_access(session: requests.Session, token_url: str, client: dict, tok: dict) -> d
    data = {
        "[Blank 1]": "refresh_token",
        "[Blank 2]": client["id"],
        "[Blank 3]": client["secret"],
        "[Blank 4]": tok["refresh_token"],
    }
    r = session.post(token_url, data=data, timeout=10)
    r.raise_for_status()
    ntok = r.json()
    save_oauth(ntok)
    return ntok

def auth_header(tok: dict) -> dict:
    return {"Authorization": f"Bearer {tok['access_token']}"}
```

Drag items: scope, client_secret, token_type, refresh_token, client_id, audience, grant_type.



Items:

- A. scope
- B. client_secret
- C. token_type
- D. refresh_token
- E. client_id
- F. audience
- G. grant_type

Place them on:

1. Blank 1 (data key for "refresh_token")
2. Blank 2 (data key for client["id"])
3. Blank 3 (data key for client["secret"])
4. Blank 4 (data key for tok["refresh_token"])

Answer

- Blank 1 (data key for "refresh_token") → **G. grant_type**
- Blank 2 (data key for client["id"]) → **E. client_id**
- Blank 3 (data key for client["secret"]) → **B. client_secret**
- Blank 4 (data key for tok["refresh_token"]) → **D. refresh_token**

Explanation

The refresh request must identify the OAuth grant as a refresh-token grant and provide the registered client credentials along with the saved refresh token. Therefore, the request-data keys are `grant_type`, `client_id`, `client_secret`, and `refresh_token`, respectively. The other available snippets are not needed to construct these four fields.

https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/
https://developer.cisco.com/docs/cisco-xdr/generate-access-and-refresh-tokens/

✗ **scope** scope
This is not the key for any of the four values shown in the request data. The code uses the refresh-token grant and supplies client credentials and a refresh token; no scope value is given.

✓ **client_secret** client_secret
The third data field contains `client["secret"]`, the client's secret credential. The appropriate request key for that value is `client_secret`.

https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/

✗ **token_type** token_type
`token_type` describes the type of token returned by the token endpoint, not a request parameter for the client secret or other values in this refresh request.

✓ **refresh_token** refresh_token
The fourth data field contains `tok["refresh_token"]`, the saved refresh token used to obtain a new access token. Its request key is `refresh_token`.

https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/
https://developer.cisco.com/docs/cisco-xdr/generate-access-and-refresh-tokens/

✓ **client_id** client_id
The second data field contains `client["id"]`, which is the client identifier. The corresponding request key is `client_id`.

https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/

✗ **audience** audience
An audience parameter is not one of the required fields represented by these blanks. The code instead needs the grant type, client credentials, and saved refresh token.

✓ **grant_type** grant_type
The first data field has the value `"refresh_token"`, which specifies the OAuth grant being used. The parameter name for that value is `grant_type`.

https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/
https://developer.cisco.com/docs/cisco-xdr/generate-access-and-refresh-tokens/

Dumpsgate

Question 20

Drag and drop · Domain: D1 Network Automation › 1.3 Construct a network automation solution with RESTCONF (RFC 8040), given the YANG model, to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Refer to the exhibit.

```
module Cisco-IOS-XE-native {
  +--rw native
     +--rw asset-labeling
        +--rw label        string
        +--rw description  string
}
```

Refer to the exhibit. An engineer must complete a RESTCONF-based Python script to configure device settings based on the provided YANG model. The YANG model defines the hierarchical structure for asset labeling within the Cisco IOS XE native configuration. This script will enable automated updates to the device configuration to ensure accurate asset labeling in the new network service rollout at the edge location. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Python script. Not all options are used.

```
import requests
import json
from requests.auth import HTTPBasicAuth

device_ip = "192.0.2.1"
username = "admin"
password = "password"

url = f"https://{device_ip}/restconf/data/Cisco-IOS-XE-native:native/asset-labeling"

[Blank 1]
    "Content-Type": "application/yang-data+json",
    "Accept": "application/yang-data+json"
}
payload = {
    [Blank 2]
        "label": "EdgeLocationAsset",
        [Blank 3]
    }
}
try:
    response = requests.put(
        url,
        auth=HTTPBasicAuth(username, password),
        headers=headers,
        data=json.dumps(payload),
        verify=False
    )
    [Blank 4]
        print("Asset labeling configuration updated successfully.")
    else:
        print(f"Failed to update asset labeling: {response.status_code} {response.text}")
except Exception as e:
    print(f"Error occurred: {e}")
```

```
module Cisco-IOS-XE-native {
  +--rw native
     +--rw asset-labeling
        +--rw label          string
        +--rw description    string
}
```

```
import requests
import json
from requests.auth import HTTPBasicAuth

device_ip = "192.0.2.1"
username = "admin"
password = "password"

url = f"https://{device_ip}/restconf/data/Cisco-IOS-XE-native:native/asset-labeling"

    "Content-Type": "application/yang-data+json",
    "Accept": "application/yang-data+json"
}
payload = {
        "label": "EdgeLocationAsset",
    }
}
try:
    response = requests.put(
        url,
        auth=HTTPBasicAuth(username, password),
        headers=headers,
        data=json.dumps(payload),
        verify=False
    )
        print("Asset labeling configuration updated successfully.")
    else:
        print(f"Failed to update asset labeling: {response.status_code} {response.text}")
except Exception as e:
    print(f"Error occurred: {e}")
```

```
if response.status_code in (200, 201, 204):
headers = {
"label": "Asset label for new edge location rollout"
"Cisco-IOS-XE:asset-labeling": {
"description": "Asset label for new edge location rollout"
"Cisco-IOS-XE-native:asset-labeling": {
head = {
```

Items:

- A. if response.status_code in (200, 201, 204):
- B. headers = {
- C. "label": "Asset label for new edge location rollout"
- D. "Cisco-IOS-XE:asset-labeling": {
- E. "description": "Asset label for new edge location rollout"
- F. "Cisco-IOS-XE-native:asset-labeling": {
- G. head = {

Place them on:

1. Blank 1 (before the Content-Type and Accept entries)
2. Blank 2 (inside payload, before the "label" entry)
3. Blank 3 (inside the asset-labeling object, after the "label" entry)
4. Blank 4 (after the requests.put call, before the success print)

Answer

- Blank 1 (before the Content-Type and Accept entries) → **B. headers = {**
- Blank 2 (inside payload, before the "label" entry) → **F. "Cisco-IOS-XE-native:asset-labeling": {**
- Blank 3 (inside the asset-labeling object, after the "label" entry) → **E. "description": "Asset label for new edge location rollout"**
- Blank 4 (after the requests.put call, before the success print) → **A. if response.status_code in (200, 201, 204):**

Explanation

The YANG model places the `label` and `description` leaves inside `native/asset-labeling`, so the payload needs the `Cisco-IOS-XE-native:asset-labeling` object containing both values. The first blank begins the HTTP headers dictionary, and the final blank checks for successful RESTCONF response status codes before printing the success message. These selections complete the script consistently with the supplied model and the shown PUT request.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/26x/26x-programmability-cg/restconf_protocol.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/restconf_prog_int.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b_179_programmability_cg/m_179_prog_restconf.html

✓ **Box 1: headers = {** headers = {
This opens the dictionary that contains the `Content-Type` and `Accept` entries already shown below the blank. The other header-related option, `head = {`, does not define the variable named `headers` used in the PUT request.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/26x/26x-programmability-cg/restconf_protocol.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/restconf_prog_int.html

✓ **Box 2: "Cisco-IOS-XE-native:asset-labeling": {** "Cisco-IOS-XE-native:asset-labeling": {
The payload must identify the asset-labeling container in the Cisco IOS XE native model, and this key opens the object containing its leaf values. The model and RESTCONF path specify the native asset-labeling hierarchy.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/26x/26x-programmability-cg/restconf_protocol.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/

Free demo · full exam: 549 questions

restconf_prog_int.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b_179_programmability_cg/m_179_prog_restconf.html

✓ **Box 3: "description": "Asset label for new edge location rollout"** "description": "Asset label for new edge location rollout"
The YANG model defines `description` as a string leaf alongside `label` under `asset-labeling`, so this value belongs inside that object after the label entry. The other option that uses the key `label` would duplicate the existing `label` field rather than fill the missing leaf.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/26x/26x-programmability-cg/restconf_protocol.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/restconf_prog_int.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b_179_programmability_cg/m_179_prog_restconf.html

✓ **Box 4: if response.status_code in (200, 201, 204):** if response.status_code in (200, 201, 204):
This tests for the successful response codes indicated by the script and makes the success print conditional on the PUT succeeding. The remaining options are payload or header fragments, not a status check that can precede the success message.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/26x/26x-programmability-cg/restconf_protocol.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/restconf_prog_int.html

✗ **"label": "Asset label for new edge location rollout"** "label": "Asset label for new edge location rollout"
This is a string-valued `label` entry, but the payload already provides the `label` key with the value `EdgeLocationAsset`. The missing payload entry is the separate `description` leaf defined by the YANG model.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/26x/26x-programmability-cg/restconf_protocol.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/restconf_prog_int.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b_179_programmability_cg/m_179_prog_restconf.html

✗ **"Cisco-IOS-XE:asset-labeling": {** "Cisco-IOS-XE:asset-labeling": {
This uses a different module prefix from the Cisco IOS XE native model and the RESTCONF path shown in the question. The payload key must be `Cisco-IOS-XE-native:asset-labeling` to identify the container in this model.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/26x/26x-programmability-cg/restconf_protocol.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/restconf_prog_int.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b_179_programmability_cg/m_179_prog_restconf.html

✗ **head = {** head = {
The request passes `headers=headers`, so defining `head` would leave the variable used by the request undefined. The blank must start the `headers` dictionary.

Question 21

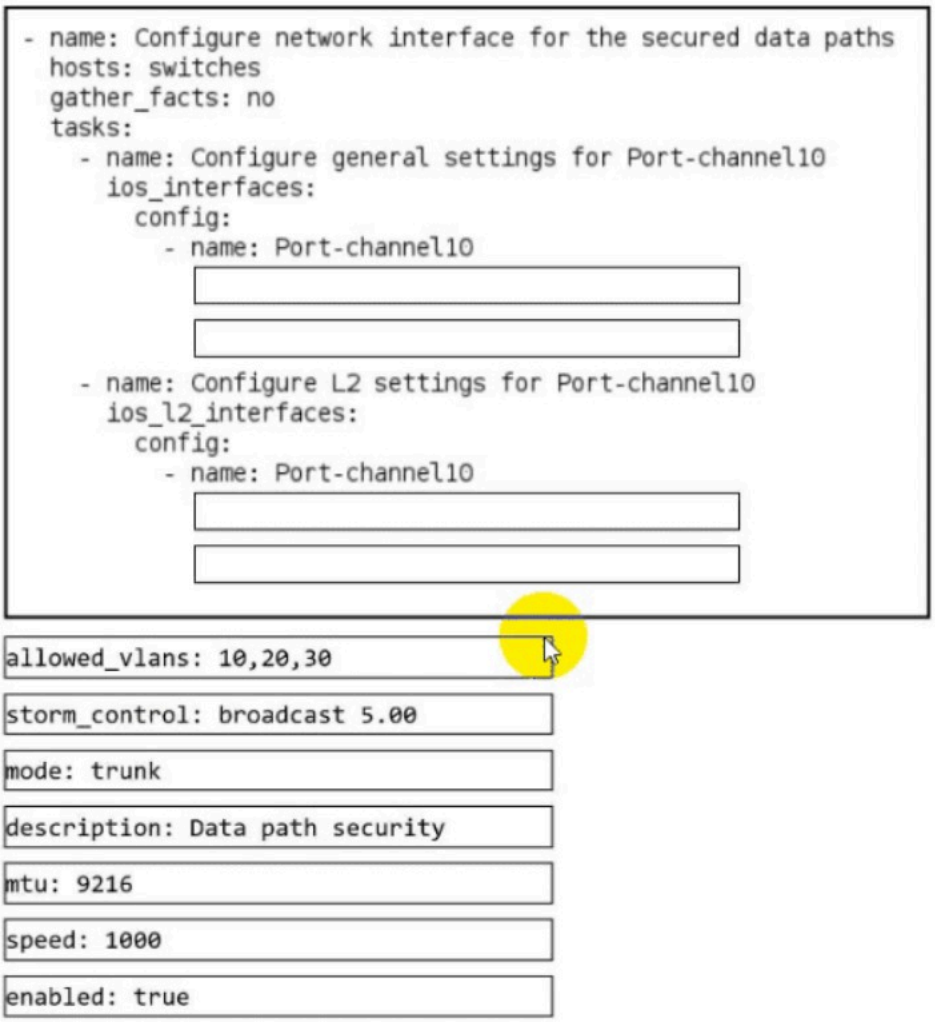
Drag and drop · Domain: D1 Network Automation › 1.1 Construct a network automation solution with Ansible to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

A network automation engineer is building an Ansible playbook to deploy interface configurations for secured data paths on designated switches. The configuration must set all required parameters for Port-channel 10 without missing any steps. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Ansible playbook. Not all options are used.

```
- name: Configure network interface for the secured data paths
  hosts: switches
  gather_facts: no
  tasks:
    - name: Configure general settings for Port-channel10
      ios_interfaces:
        config:
          - name: Port-channel10
            [Blank 1]
            [Blank 2]
    - name: Configure L2 settings for Port-channel10
      ios_l2_interfaces:
        config:
          - name: Port-channel10
            [Blank 3]
            [Blank 4]
```

Dumpsgate

Free demo · full exam: 549 questions



Items:

- A. allowed_vlans: 10,20,30
- B. storm_control: broadcast 5.00
- C. mode: trunk
- D. description: Data path security
- E. mtu: 9216
- F. speed: 1000
- G. enabled: true

Place them on:

1. Blank 1 (general settings for Port-channel10, under ios_interfaces)
2. Blank 2 (general settings for Port-channel10, under ios_interfaces)
3. Blank 3 (L2 settings for Port-channel10, under ios_l2_interfaces)
4. Blank 4 (L2 settings for Port-channel10, under ios_l2_interfaces)

Free demo · full exam: 549 questions

Answer

- Blank 1 (general settings for Port-channel10, under ios_interfaces) → **E. mtu: 9216**
- Blank 2 (general settings for Port-channel10, under ios_interfaces) → **G. enabled: true**
- Blank 3 (L2 settings for Port-channel10, under ios_l2_interfaces) → **C. mode: trunk**
- Blank 4 (L2 settings for Port-channel10, under ios_l2_interfaces) → **A. allowed_vlans: 10,20,30**

Explanation

The `ios_interfaces` task configures general interface properties, so `mtu: 9216` and `enabled: true` belong under Port-channel10 there. The `ios_l2_interfaces` task configures Layer 2 properties, so `mode: trunk` and `allowed_vlans: 10,20,30` complete the trunk configuration. The other snippets are not required for the stated configuration, so they are left unused.

https://www.cisco.com/en/US/docs/ios/interface/configuration/guide/ir_icfgenrh.html
https://developer.cisco.com/docs/modeling-labs/2-10/iosvl2/

✓ **<target>: allowed_vlans: 10,20,30** allowed_vlans: 10,20,30
This is an L2 interface setting, so it belongs under `ios_l2_interfaces` as Blank 4. It specifies which VLANs Port-channel10 allows on the trunk.

https://developer.cisco.com/docs/modeling-labs/2-10/iosvl2/

✗ **<target>: storm_control: broadcast 5.00** storm_control: broadcast 5.00
This snippet is not one of the required settings for the Port-channel10 configuration described in the question. The selected L2 settings specify trunk mode and the allowed VLANs instead.

✓ **<target>: mode: trunk** mode: trunk
The interface must carry the listed VLANs, and `mode: trunk` configures Port-channel10 as a trunk. It belongs under `ios_l2_interfaces` as Blank 3.

https://developer.cisco.com/docs/modeling-labs/2-10/iosvl2/

✗ **<target>: description: Data path security** description: Data path security
A description is not required to configure the specified MTU, enabled state, trunk mode, and allowed VLANs. It is therefore not used in the completed playbook for this question.

✓ **<target>: mtu: 9216** mtu: 9216
MTU is a general interface property, so it belongs under `ios_interfaces` as Blank 1. The snippet sets the requested interface MTU to 9216.

https://www.cisco.com/en/US/docs/ios/interface/configuration/guide/ir_icfgenrh.html

✗ **<target>: speed: 1000** speed: 1000
The scenario does not require setting Port-channel10's speed. The general settings selected for the task are the MTU and enabled state.

✓ **<target>: enabled: true** enabled: true
This is a general interface setting, so it belongs under `ios_interfaces` as Blank 2. Setting it to `true` enables Port-channel10.

https://www.cisco.com/en/US/docs/ios/interface/configuration/guide/ir_icfgenrh.html

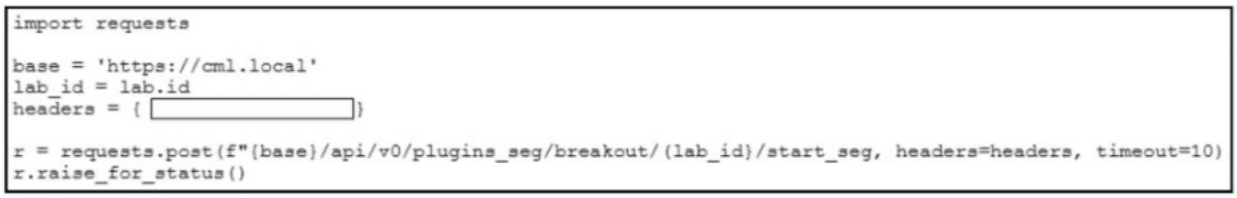
Question 22

Single choice · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

Refer to the exhibit.

```
import requests

base = 'https://cml.local'
lab_id = lab.id
headers = { [Blank 1] }

r = requests.post(f"{base}/api/v0/plugins_seg/breakout/{lab_id}/start_seg", headers=headers
r.raise_for_status()
```

Refer to the exhibit. An engineer is tasked with enabling the CML Breakout tool for a lab to provide TCP access to device consoles. The engineer must call the REST endpoint to start the breakout for the given lab ID using a session token. Which line of code must be added to the box where the code is missing to complete the Python request?



A. 'X-XSRF-TOKEN' = 'Y0W5R2ta8W5y2it6QW4RtaW1fMT9lzNCE'

B. 'password' = 'Y0W5R2ta8W5y2it6QW4RtaW1fMT9lzNCE'

C. 'headers' = 'Y0W5R2ta8W5y2it6QW4RtaW1fMT9lzNCE'

D. 'token_header' = 'Y0W5R2ta8W5y2it6QW4RtaW1fMT9lzNCE'

Answer: A. 'X-XSRF-TOKEN' = 'Y0W5R2ta8W5y2it6QW4RtaW1fMT9lzNCE'

Explanation

The request passes `headers` as a dictionary, so the missing entry must use the session token as the value for the `X-XSRF-TOKEN` header. This supplies the token in the form the CML Breakout REST endpoint expects, while allowing the POST request to start breakout for the specified lab ID. Therefore, option A is correct.

https://developer.cisco.com/docs/modeling-labs/2-10/installing-breakout-tool/

✓ **A** 'X-XSRF-TOKEN' = 'Y0W5R2ta8W5y2it6QW4RtaW1fMT9lzNCE'
The session token belongs in the `X-XSRF-TOKEN` HTTP header, which is what the request's `headers` dictionary is for. In Python dictionary syntax, the entry is written as `'X-XSRF-TOKEN': 'token-value'` (with a colon).

https://developer.cisco.com/docs/modeling-labs/2-10/installing-breakout-tool/

✗ **B** 'password' = 'Y0W5R2ta8W5y2it6QW4RtaW1fMT9lzNCE'
The scenario requires a session token, not a password. A `password` key would not provide the token in the expected request header for this endpoint.

https://developer.cisco.com/docs/modeling-labs/2-10/installing-breakout-tool/

✗ **C** 'headers' = 'Y0W5R2ta8W5y2it6QW4RtaW1fMT9lzNCE'
`headers` is already the name of the dictionary passed to `requests.post`; it is not the name of the token header. This entry would not identify the session token as the `X-XSRF-TOKEN` header value.

✗ **D** 'token_header' = 'Y0W5R2ta8W5y2it6QW4RtaW1fMT9lzNCE'
`token_header` is not the HTTP header name the request needs. The session token must be associated with the `X-XSRF-TOKEN` header instead.

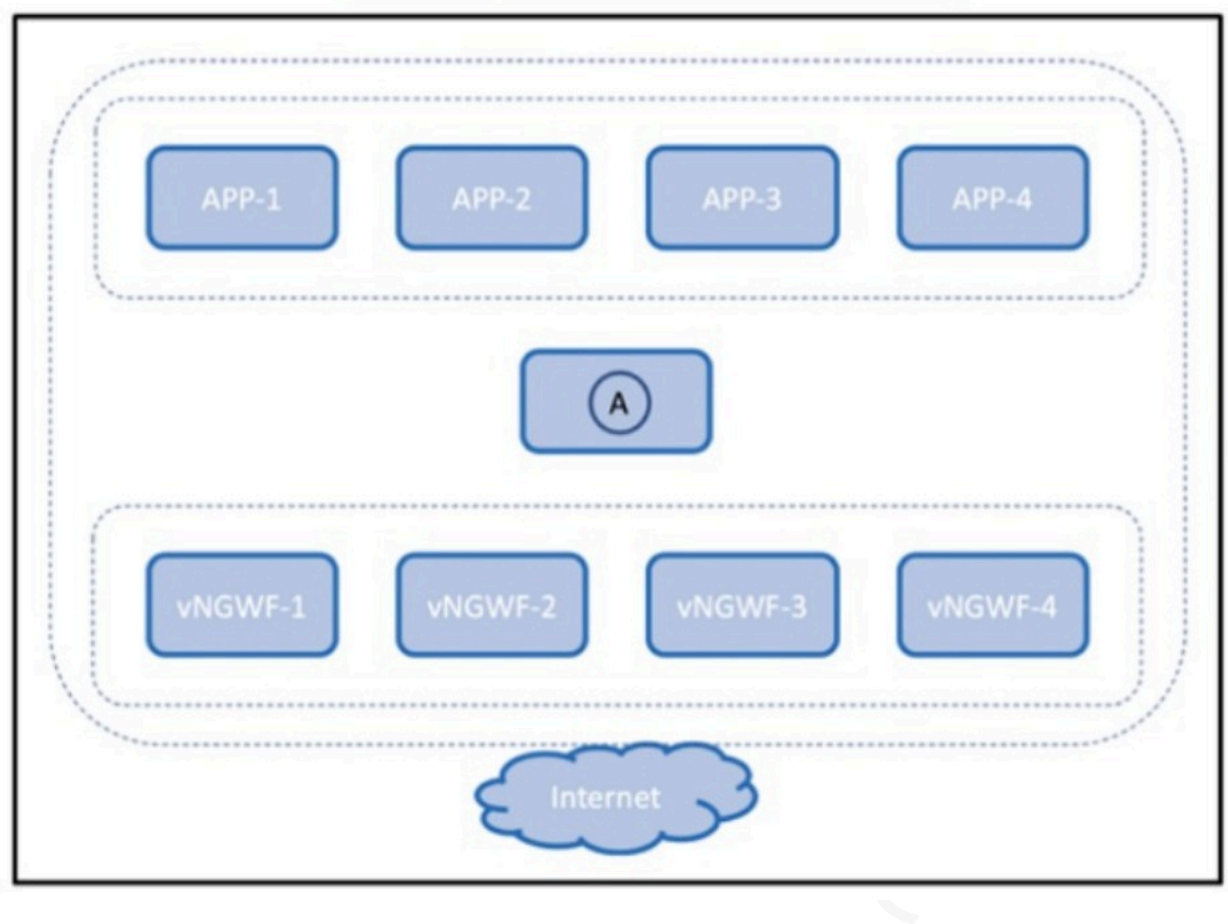
https://developer.cisco.com/docs/modeling-labs/2-10/installing-breakout-tool/

Dumpsgate

Question 23

Single choice · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

Refer to the exhibit.



The virtual next-generation firewalls running in the public cloud must be scaled out to meet the throughput requirements of the application layer. Which component is deployed to location A in the diagram?

A. frontend caching server
B. forward proxy
C. backend database
D. internal load balancer

Answer: D. internal load balancer

Explanation

The virtual next-generation firewalls sit between the Internet and the application layer, and the scenario requires scaling out those firewalls to satisfy application throughput. An internal load balancer at location

Free demo · full exam: 549 questions

A distributes application traffic across the firewall instances, enabling the scaled-out firewall tier to handle the required load. The other choices do not perform this traffic-distribution role.

https://www.cisco.com/c/en/us/td/docs/security/firepower/quick_start/consolidated_ftdv_qsg/threat-defense-virtual-77-qsg/m_threat-defense-virtual-auto-scale-solution-on-aws.html
https://meraki-go.xgslb-v3.cisco.com/c/en/us/td/docs/security/firepower/quick_start/consolidated_ftdv_qsg/threat-defense-virtual-73-qsg/m_threat-defense-virtual-auto-scale-solution-on-aws.html

✗ **A** frontend caching server
A frontend caching server stores or serves cached application content; it does not distribute traffic across the virtual firewall instances. That would not provide the scaling mechanism needed for the firewall tier.

✗ **B** forward proxy
A forward proxy mediates outbound client requests, rather than balancing incoming application traffic across the virtual next-generation firewalls. It therefore does not fit location A's role in this scale-out design.

✗ **C** backend database
A backend database stores application data and is not a traffic-distribution component for the firewall tier. Placing one at A would not address the stated firewall throughput requirement.

✓ **D** internal load balancer
The internal load balancer distributes application-layer traffic among the virtual next-generation firewall instances. This allows the firewall tier to scale out and meet the application's throughput requirements.

https://www.cisco.com/c/en/us/td/docs/security/firepower/quick_start/consolidated_ftdv_qsg/threat-defense-virtual-77-qsg/m_threat-defense-virtual-auto-scale-solution-on-aws.html
https://meraki-go.xgslb-v3.cisco.com/c/en/us/td/docs/security/firepower/quick_start/consolidated_ftdv_qsg/threat-defense-virtual-73-qsg/m_threat-defense-virtual-auto-scale-solution-on-aws.html

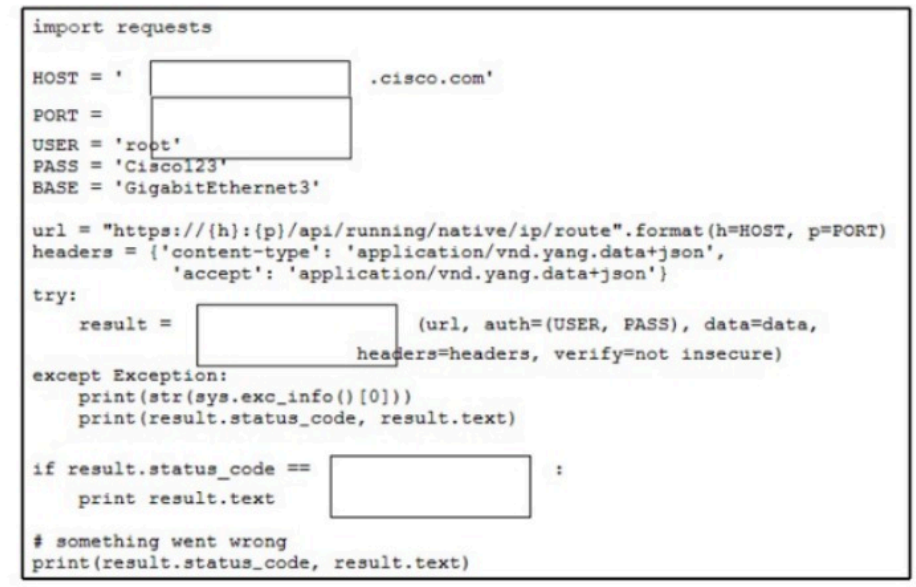
Dumpsgate

Question 24

Drag and drop · Domain: D1 Network Automation › 1.3 Construct a network automation solution with RESTCONF (RFC 8040), given the YANG model, to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Drag and drop the code from the bottom onto the box where the code is missing to configure a router that runs Cisco IOS XE by using RESTCONF. The API call is made to the management URL of the IOS XE device and on a registered port number. Errors that are caught during HTTP operation are registered. Not all options are used.

```
import requests

HOST = '[Blank 1].cisco.com'
PORT = [Blank 2]
USER = 'root'
PASS = 'Cisco123'
BASE = 'GigabitEthernet3'

url = "https://{h}:{p}/api/running/native/ip/route".format(h=HOST, p=PORT)
headers = {'content-type': 'application/vnd.yang.data+json',
           'accept': 'application/vnd.yang.data+json'}

try:
    result = [Blank 3] (url, auth=(USER, PASS), data=data,
                        headers=headers, verify=not insecure)
except Exception:
    print(str(sys.exc_info()[0]))
    print(result.status_code, result.text)

if result.status_code == [Blank 4] :
    print(result.text)
# something went wrong
print(result.status_code, result.text)
```



ios-xe-mgmt	401	requests.post
9443	201	requests.patch

Free demo · full exam: 549 questions

Answer

- Blank 1 (HOST = '[Blank 1].cisco.com') → **A. ios-xe-mgmt**
- Blank 2 (PORT = [Blank 2]) → **D. 9443**
- Blank 3 (result = [Blank 3] (url, auth=(USER, PASS), data=data, headers=headers, verify=not insecure)) → **C. requests.post**
- Blank 4 (if result.status_code == [Blank 4]) → **E. 201**

Explanation

Use `ios-xe-mgmt` as the host prefix and port `9443` to address the IOS XE management endpoint shown in the question. `requests.post` makes the RESTCONF request, and HTTP status `201` is the success code expected for this operation. The other choices do not match the specified endpoint or the successful POST response.

https://community.cisco.com/t5/devnet-sandbox/public-ios-xe-mgmt-cisco-com-api-not-responding/td-p/5273738
https://spg.xgslb-v3.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1718/b-1718-programmability-cg/restconf_protocol.html
https://community.cisco.com/t5/tools/what-is-restconf-http-port-on-xe-sandbox/m-p/4108337/highlight/true

✓ **Box 1: ios-xe-mgmt** ios-xe-mgmt
This is the host prefix used to form the IOS XE management hostname in the supplied code. It matches the management endpoint context in the question.

https://community.cisco.com/t5/devnet-sandbox/public-ios-xe-mgmt-cisco-com-api-not-responding/td-p/5273738

✓ **Box 2: 9443** 9443
The URL uses HTTPS and the question specifies the registered port for this management endpoint; `9443` is the matching port choice. The other numeric options are HTTP status codes, not the endpoint port.

https://community.cisco.com/t5/tools/what-is-restconf-http-port-on-xe-sandbox/m-p/4108337/highlight/true

✓ **Box 3: requests.post** requests.post
This calls the HTTP POST method for the RESTCONF URL, passing the authentication, request data, headers, and TLS verification setting shown. `requests.patch` would use a different HTTP method and is not the selected operation in the given code.

https://spg.xgslb-v3.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1718/b-1718-programmability-cg/restconf_protocol.html

✓ **Box 4: 201** 201
A `201` status indicates successful creation for the POST operation, so it is the success code tested before printing the response text. `401` indicates an unauthorized request, not success.

https://spg.xgslb-v3.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1718/b-1718-programmability-cg/restconf_protocol.html

✗ **Unused: 401** 401
`401` represents an unauthorized response, so it does not belong in the success-status check. It is not the registered port, host prefix, or HTTP method required by the blanks.

✗ **Unused: requests.patch** requests.patch
This is an HTTP PATCH call, whereas the correct code uses `requests.post` for the RESTCONF operation in this question. It therefore does not fill the request-method blank.

https://spg.xgslb-v3.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1718/b-1718-programmability-cg/restconf_protocol.html

✗ **Unused: 201** 201
This option is not unused: it is the correct value for Blank 4, where the code checks for a successful POST response. It should not be placed in any other blank.

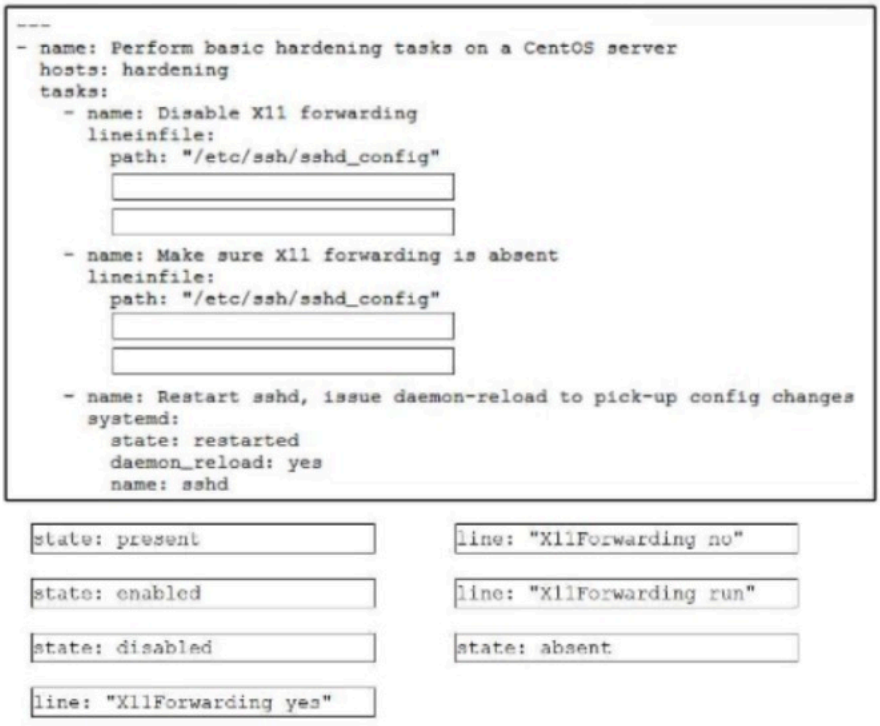
https://spg.xgslb-v3.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1718/b-1718-programmability-cg/restconf_protocol.html

Question 25

Drag and drop · Domain: D1 Network Automation › 1.1 Construct a network automation solution with Ansible to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Drag and drop the code snippets from the bottom onto the blanks in the code to disable X11Forwarding on the SSH service of a CentOS server. Not all options are used.

```
---
- name: Perform basic hardening tasks on a CentOS server
  hosts: hardening
  tasks:
    - name: Disable X11 forwarding
      lineinfile:
        path: "/etc/ssh/sshd_config"
        [Blank 1]
        [Blank 2]
    - name: Make sure X11 forwarding is absent
      lineinfile:
        path: "/etc/ssh/sshd_config"
        [Blank 3]
        [Blank 4]
    - name: Restart sshd, issue daemon-reload to pick-up config changes
      systemd:
        state: restarted
        daemon_reload: yes
        name: sshd
```



Free demo · full exam: 549 questions

Items:

- A. state: present
- B. state: enabled
- C. state: disabled
- D. line: "X11Forwarding yes"
- E. line: "X11Forwarding no"
- F. line: "X11Forwarding run"
- G. state: absent

Place them on:

1. Blank 1 (Disable X11 forwarding, after path: "/etc/ssh/sshd_config")
2. Blank 2 (Disable X11 forwarding, after Blank 1)
3. Blank 3 (Make sure X11 forwarding is absent, after path: "/etc/ssh/sshd_config")
4. Blank 4 (Make sure X11 forwarding is absent, after Blank 3)

Answer

- Blank 1 (Disable X11 forwarding, after path: "/etc/ssh/sshd_config") → **A. state: present**
- Blank 2 (Disable X11 forwarding, after Blank 1) → **E. line: "X11Forwarding no"**
- Blank 3 (Make sure X11 forwarding is absent, after path: "/etc/ssh/sshd_config") → **G. state: absent**
- Blank 4 (Make sure X11 forwarding is absent, after Blank 3) → **D. line: "X11Forwarding yes"**

Explanation

The first `lineinfile` task ensures the SSH configuration contains `X11Forwarding no`, which disables X11 forwarding. The second task removes any `X11Forwarding yes` entry so it cannot remain in the configuration and conflict with the intended setting. The play then restarts `sshd` to apply the configuration changes.

✓ **Box 1: state: present** state: present
This makes the first `lineinfile` task ensure the specified configuration line is present in `/etc/ssh/sshd_config`.

✓ **Box 2: line: "X11Forwarding no"** line: "X11Forwarding no"
This is the setting that disables X11 forwarding, and the first task ensures it is present in the SSH daemon configuration.

✓ **Box 3: state: absent** state: absent
This makes the second `lineinfile` task remove the specified line if it exists in the configuration file.

✓ **Box 4: line: "X11Forwarding yes"** line: "X11Forwarding yes"
Removing this line prevents an explicit enabled setting from remaining in the configuration, matching the task's requirement to ensure it is absent.

✗ **state: enabled** state: enabled
This is not the appropriate `lineinfile` state for ensuring a line is present or absent. The tasks require `present` and `absent` to manage the configuration lines.

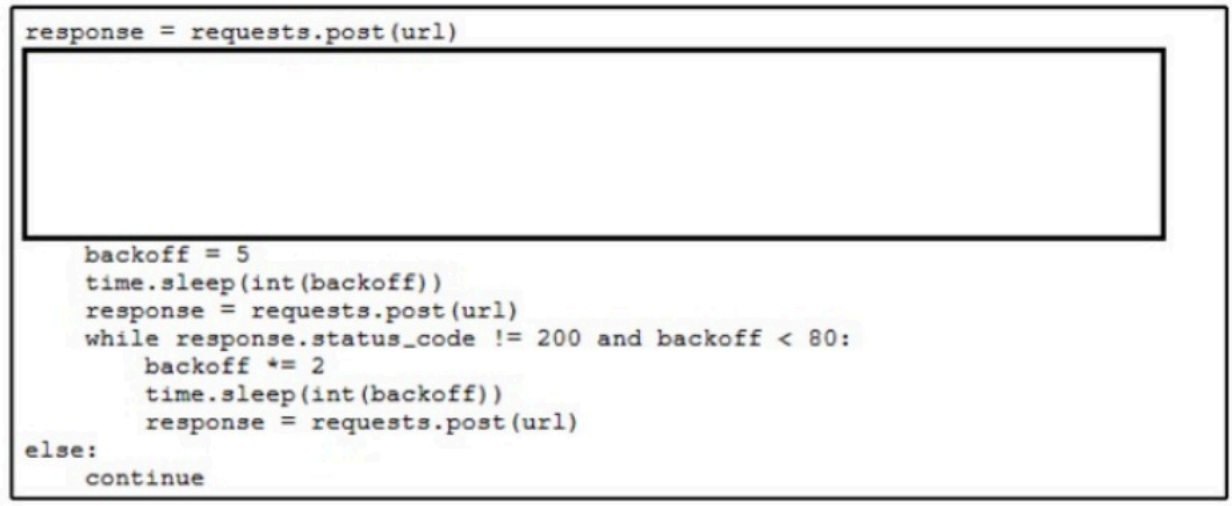
✗ **state: disabled** state: disabled
This does not set the desired SSH configuration line or remove the conflicting line. The `lineinfile` tasks use `present` and `absent` instead.

✗ **line: "X11Forwarding run"** line: "X11Forwarding run"
`run` is not the required SSH configuration value for disabling X11 forwarding. The intended setting is `X11Forwarding no`.

Question 26

Single choice · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

Refer to the exhibit.



```
response = requests.post(url)

    backoff = 5
    time.sleep(int(backoff))
    response = requests.post(url)
    while response.status_code != 200 and backoff < 80:
        backoff *= 2
        time.sleep(int(backoff))
        response = requests.post(url)
else:
    continue
```

An engineer needs to implement REST API error handling when a timeout or rate limit scenario is present.

Which code snippet must be placed into the blank in the code to complete the API request?

A.

```
if response.status_code == 429:
    wait = response.headers.get('Retry-After', 99)
    print(f'-> got {response.status_code} from {url}. retrying after {wait}s')
    time.sleep(int(wait))
    response = requests.post(url)
elif response.status_code == 408:
```

B.

```
if response.status_code == 401:
    wait = response.headers.get('Retry-After', 99)
    print(f'-> got {response.status_code} from {url}. retrying after {wait}s')
    time.sleep(int(wait))
    response = requests.post(url)
elif response.status_code == 408:
```

C.

Free demo · full exam: 549 questions

```
if response.status_code == 408:
    wait = response.headers.get('Retry-After', 99)
    print(f'-> got {response.status_code} from {url}. retrying after {wait}s')
    time.sleep(int(wait))
    response = requests.post(url)
elif response.status_code == 429:
```

D.

```
if response.status_code == 429:
    wait = response.headers.get('Retry-After', 99)
    print(f'-> got {response.status_code} from {url}. retrying after {wait}s')
    time.sleep(int(wait))
    response = requests.post(url)
elif response.status_code == 401:
```

Answer: A

Explanation

Option A checks for HTTP 429, the status used for rate limiting, and reads the Retry-After header before retrying. It also includes an HTTP 408 branch for a timeout, matching both error scenarios in the question. The later backoff loop can then continue retrying when the response is not successful.

https://developer.cisco.com/meraki/api-v1/rate-limit/
https://community.cisco.com/t5/network-platform-api/how-to-access-retry-after-header-in-api-response/m-p/5425571

✓ **A** Handle status 429 using Retry-After, then check for status 408.
HTTP 429 addresses the rate-limit case, and the code waits for the server-provided Retry-After value before retrying. The 408 branch addresses the timeout case, so this option covers both required scenarios.

https://developer.cisco.com/meraki/api-v1/rate-limit/
https://community.cisco.com/t5/network-platform-api/how-to-access-retry-after-header-in-api-response/m-p/5425571

✗ **B** Handle status 401 using Retry-After, then check for status 408.
This handles a timeout with status 408, but uses 401 (unauthorized) instead of 429 for rate limiting. It therefore does not correctly cover the rate-limit scenario.

✗ **C** Handle status 408 using Retry-After, then check for status 429.
It identifies both relevant status codes, but applies the Retry-After handling to 408 rather than 429. The rate-limit response is the one that should use the Retry-After value in this snippet.

https://developer.cisco.com/meraki/api-v1/rate-limit/
https://community.cisco.com/t5/network-platform-api/how-to-access-retry-after-header-in-api-response/m-p/5425571

✗ **D** Handle status 429 using Retry-After, then check for status 401.
This handles the rate-limit case, but checks for 401 rather than 408 in the other branch. It therefore misses the timeout scenario specified in the question.

https://developer.cisco.com/meraki/api-v1/rate-limit/

Question 27

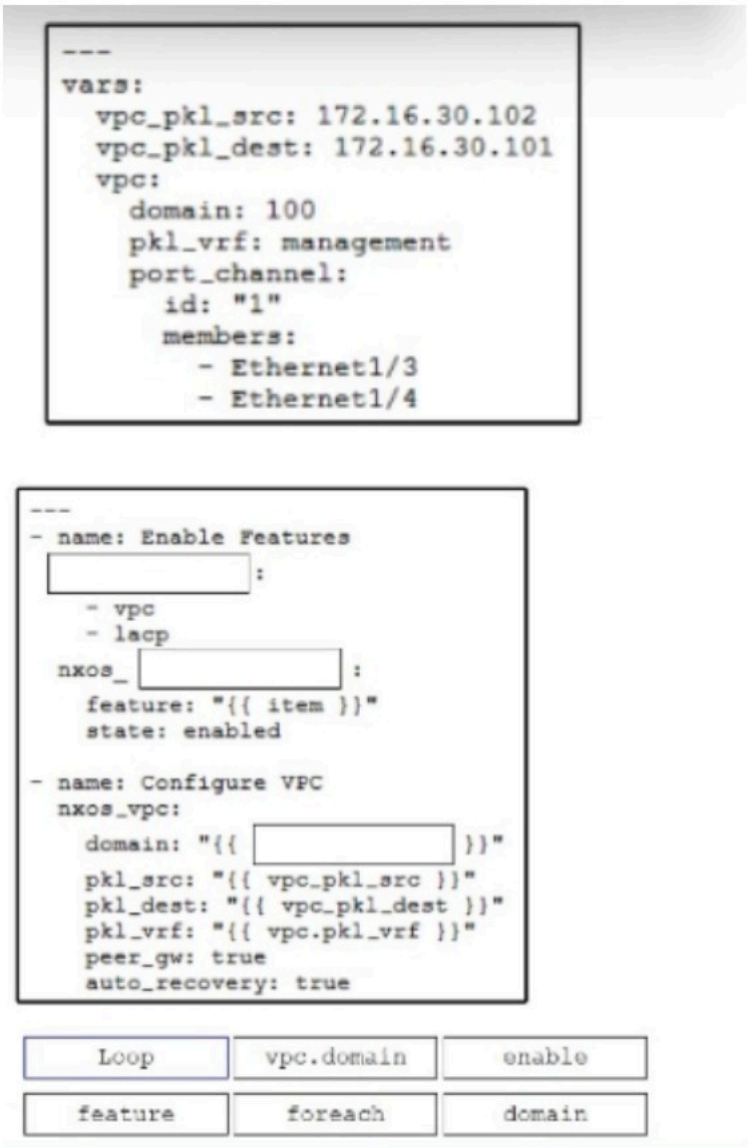
Drag and drop · Domain: D1 Network Automation › 1.1 Construct a network automation solution with Ansible to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Refer to the exhibit. Drag and drop the code snippets from the bottom onto the blanks in the code to complete the playbook for creation of an Ansible role that configures VPC on a Cisco Nexus switch. Not all options are used.

Exhibit:

```
---
vars:
  vpc_pkl_src: 172.16.30.102
  vpc_pkl_dest: 172.16.30.101
  vpc:
    domain: 100
    pkl_vrf: management
  port_channel:
    id: "1"
    members:
      - Ethernet1/3
      - Ethernet1/4
```

```
---
- name: Enable Features
  [Blank 1]:
    - vpc
    - lacp
  nxos_[Blank 2]:
    feature: "{{ item }}"
    state: enabled

- name: Configure VPC
  nxos_vpc:
    domain: "{{ [Blank 3] }}"
    pkl_src: "{{ vpc_pkl_src }}"
    pkl_dest: "{{ vpc_pkl_dest }}"
    pkl_vrf: "{{ vpc.pkl_vrf }}"
    peer_gw: true
    auto_recovery: true
```



Items:

- A. Loop
- B. vpc.domain
- C. enable
- D. feature
- E. foreach
- F. domain

Place them on:

1. Blank 1 (`[Blank 1]:` under `- name: Enable Features`)
2. Blank 2 (`nxos_[Blank 2]:`)
3. Blank 3 (`domain: "{{ [Blank 3] }}"`)

Answer

- Blank 1 (`[Blank 1]:` under `- name: Enable Features`) → **A. Loop**
- Blank 2 (`nxos_[Blank 2]:`) → **D. feature**
- Blank 3 (`domain: "{{ [Blank 3] }}"`) → **B. vpc.domain**

Free demo · full exam: 549 questions

Dumpsgate

Question 28

Single choice · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

How is an OAulh2 three-legged authentication flow initiated?

A. The user makes a roquost to the OAuth client
B. Exchange the key f<x an access token
C. Construct an API call to retrieve the credentials.
D. Get the authorization code.

Answer: A. The user makes a roquost to the OAuth client

Explanation

The three-legged OAuth2 flow begins when the user makes a request to the OAuth client for a protected resource or action. The client then starts the authorization process; obtaining an authorization code and exchanging it for an access token happen later in the flow. [3]

https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/

✓ **A** The user makes a request to the OAuth client
This is the initiating step: the user's request prompts the client to begin the authorization flow. The client

Free demo · full exam: 549 questions

can then direct the user through authorization before accessing the protected resource. [3]
https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/

✗ **B** Exchange the key for an access token
Exchanging a credential for an access token is a later token-acquisition step, not how the three-legged flow is initiated. The flow starts with the user's request to the client. [3]
https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/

✗ **C** Construct an API call to retrieve the credentials.
Constructing an API call to retrieve credentials is not the initiating action in the three-legged OAuth2 flow. The user first makes a request to the OAuth client, which starts authorization. [3]
https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/

✗ **D** Get the authorization code.
The authorization code is obtained after the authorization process has begun; it is not the first step. The flow is initiated by the user's request to the OAuth client. [3]
https://developer.cisco.com/docs/cisco-xdr/oauth2-authorization-code-grant-credentials-web-applications/

Question 29

Drag and drop · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

Refer to the exhibit.

```
{
  "Moid": "2f5gsegse4f057feer234",
  "CreateTime": "2021-02-08T14:30:00.826-07:00",
  "ModTime": "2021-02-08T14:30:00.826-07:00",
  "Tags": [
    {
      "Key": "DataCenter",
      "Value": "City-A"
    },
    {
      "Key": "Environment",
      "Value": "Testing"
    }
  ]
}
```

An engineer is attempting to retrieve information about free resources in the City-A data center. The engineer wants to know only about resources in the testing environment. Drag and drop the code snippets from the bottom onto the blanks in the API call to meet this requirement. Some options may be used more than once. Not all options are used.

```
GET /api/v1/compute/[Blank 1] ?$filter=[Blank 2] /
any(t: [Blank 3] eq 'DataCenter' and [Blank 4] eq 'City-A' )
and Tags/any(t: [Blank 5] eq 'Environment' and [Blank 6] eq 'Testing')
```



Items:

- A. Tags
- B. t/Key
- C. Key
- D. t/Value
- E. Tag
- F. RackUnits
- G. Value

Place them on:

1. Blank 1 (/api/v1/compute/[Blank 1])
2. Blank 2 ($filter=[Blank 2])
3. Blank 3 (any(t: [Blank 3] eq 'DataCenter')
4. Blank 4 (and [Blank 4] eq 'City-A')
5. Blank 5 (Tags/any(t: [Blank 5] eq 'Environment')
6. Blank 6 (and [Blank 6] eq 'Testing')

Answer

- Blank 1 (/api/v1/compute/[Blank 1]) → **F. RackUnits**
- Blank 2 ($filter=[Blank 2]) → **A. Tags**
- Blank 3 (any(t: [Blank 3] eq 'DataCenter') → **B. t/Key**
- Blank 4 (and [Blank 4] eq 'City-A') → **D. t/Value**
- Blank 5 (Tags/any(t: [Blank 5] eq 'Environment') → **B. t/Key**
- Blank 6 (and [Blank 6] eq 'Testing') → **D. t/Value**

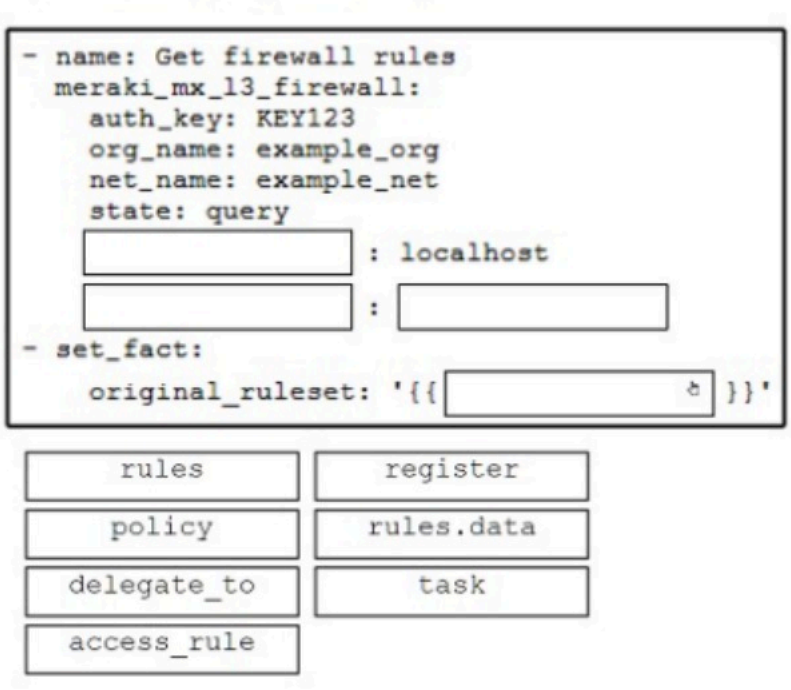
Dumpsgate

Question 30

Drag and drop · Domain: D1 Network Automation › 1.1 Construct a network automation solution with Ansible to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Drag and drop the code from the bottom onto the box where the code is missing in the Ansible script to get the existing firewall rules from Cisco Meraki and store the results to a new variable. Not all options are used.

```
- name: Get firewall rules
  meraki_mx_l3_firewall:
    auth_key: KEY123
    org_name: example_org
    net_name: example_net
    state: query
    [Blank 1]: localhost
    [Blank 2]: [Blank 3]
- set_fact:
    original_ruleset: '{{[Blank 4]}}'
```



Items:

- A. rules
- B. register
- C. policy
- D. rules.data
- E. delegate_to
- F. task
- G. access_rule

Place them on:

1. Blank 1 (before : localhost)
2. Blank 2 (before the value for the module result)
3. Blank 3 (the value assigned to register)
4. Blank 4 (value of original_ruleset)

Answer

- Blank 1 (before : localhost) → **E. delegate_to**
- Blank 2 (before the value for the module result) → **B. register**
- Blank 3 (the value assigned to register) → **A. rules**
- Blank 4 (value of original_ruleset) → **D. rules.data**

Explanation

`delegate_to: localhost` runs the Meraki module on the control host. `register: rules` captures the module result in a variable named `rules`, and `rules.data` selects the returned data for `original_ruleset`. The other options do not fill these YAML fields with the required meaning.

✓ **delegate_to** delegate_to
This fills Blank 1, producing `delegate_to: localhost`. It directs Ansible to run the Meraki module on localhost, as required by the script.

✓ **rules** rules
This fills Blank 3 as the variable name in `register: rules`. Ansible stores the module result under that name, allowing the next task to reference it.

✗ **policy** policy
`policy` is not the directive needed to capture the module result, nor is it the result variable or the requested result-data expression. It does not fit any blank in this script.

✓ **rules.data** rules.data
This fills Blank 4, making `original_ruleset: '{{rules.data}}'`. It accesses the `data` field from the registered `rules` result and assigns it to the new fact.

✓ **register** register
This fills Blank 2, introducing Ansible's `register` directive so the module output can be saved for use by the following task.

✗ **task** task
`task` is not the Ansible keyword for saving a module's output or selecting its returned data. It does not belong in any of the blanks shown.

✗ **access_rule** access_rule
`access_rule` is not the needed result-capture directive or the registered result expression. The script requires the module result to be registered as `rules` and then reads `rules.data`.

Question 31

Multiple choice · Domain: D1 Network Automation › 1.1 Construct a network automation solution with Ansible to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

An engineer needs to automate the configuration on a Cisco NX-OS switch. The solution must meet these requirements:

- Ansible is used
- The transport protocol is encrypted
- Support for Asynchronous requests.

Which two access mechanisms must be used? (Choose two.)

A. SNMP Version 2c
B. NETCONF
C. NX-API
D. HTTPS
E. JSON-RPC

Answer

- C. **NX-API**
- D. **HTTPS**

Explanation

Use NX-API with HTTPS: NX-API provides the switch's programmable interface and supports asynchronous requests, while HTTPS encrypts the transport. Ansible can use NX-API to automate Cisco NX-OS configuration, so these two choices meet the stated requirements together.

https://www.cisco.com/c/en/us/td/docs/dcn/nx-os/nexus3000/105x/programmability/cisco-nexus-3500-series-nx-os-programmability-guide_105x/m_n3500_nx-api_9x.html
https://www.cisco.com/c/en/us/td/docs/dcn/nx-os/nexus9000/106x/programmability/cisco-nexus-9000-series-nx-os-programmability-guide-106x/m-using-ansible-with-cisco-nx-os.html
https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/7-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_7x/NX_API.html

✗ **A** SNMP Version 2c
SNMPv2c does not encrypt its traffic, so it fails the encrypted-transport requirement. It is also not the NX-API mechanism used for the Ansible automation described here.

✗ **B** NETCONF
NETCONF is not the access mechanism selected for this scenario's NX-OS Ansible workflow. The required combination is NX-API for the programmable interface and HTTPS for encrypted transport.

https://www.cisco.com/c/en/us/td/docs/dcn/nx-os/nexus9000/106x/programmability/cisco-nexus-9000-series-nx-os-programmability-guide-106x/m-using-ansible-with-cisco-nx-os.html

✓ **C** NX-API
NX-API is the NX-OS programmable interface that Ansible can use to automate switch configuration. It supports asynchronous requests, satisfying that requirement.

https://www.cisco.com/c/en/us/td/docs/dcn/nx-os/nexus3000/105x/programmability/cisco-nexus-3500-series-nx-os-programmability-guide_105x/m_n3500_nx-api_9x.html
https://www.cisco.com/c/en/us/td/docs/dcn/nx-os/nexus9000/106x/programmability/cisco-nexus-9000-series-nx-os-programmability-guide-106x/m-using-ansible-with-cisco-nx-os.html
https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/7-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_7x/NX_API.html

Free demo · full exam: 549 questions

✓ **D** HTTPS
HTTPS provides encryption for communications to the NX-API interface, meeting the transport-security requirement. It is used together with NX-API rather than replacing the programmable interface.
https://www.cisco.com/c/en/us/td/docs/dcn/nx-os/nexus3000/105x/programmability/cisco-nexus-3500-series-nx-os-programmability-guide_105x/m_n3500_nx-api_9x.html
https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/7-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_7x/NX_API.html

✗ **E** JSON-RPC
JSON-RPC is not the access-mechanism-and-transport pair needed to meet the requirements. The specified combination is NX-API for access and HTTPS for encrypted transport.
https://www.cisco.com/c/en/us/td/docs/dcn/nx-os/nexus3000/105x/programmability/cisco-nexus-3500-series-nx-os-programmability-guide_105x/m_n3500_nx-api_9x.html
https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/7-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_7x/NX_API.html

Question 32

Single choice · Domain: D1 Network Automation › 1.3 Construct a network automation solution with RESTCONF (RFC 8040), given the YANG model, to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Refer to the exhibit.

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
  <patch-id>add-hostname-patch</patch-id>
  <edit>
    [Blank 1]
    <target>/hostname</target>
    <value>
      <hostname xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">testname
      </hostname>
    </value>
  </edit>
</yang-patch>
```

Refer to the exhibit. An engineer must add a new hostname configuration to a Cisco IOS device by using RESTCONF to make a YANG patch request. Which code snippet completes the configuration?

```
1  <yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
2      <patch-id>add-hostname-patch</patch-id>
3      <edit>
4          [          ]
5
6          <target>/hostname</target>
7          <value>
8              <hostname xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">testname
9  </hostname>
10         </value>
11     </edit>
12 </yang-patch>
```

A. ```xml <edit-id>edit1</edit-id> <operation>edit</operation> ```

```
<edit-id>edit1</edit-id>
<operation>edit</operation>
```

B. ```xml <edit-id>edit1</edit-id> <operation>new</operation> ```

```
<edit-id>edit1</edit-id>
<operation>new</operation>
```

C. ```xml <edit-id>edit1</edit-id> <operation>create</operation> ```

```
<edit-id>edit1</edit-id>
<operation>create</operation>
```

D. ```xml <create-id>create1</create-id> <operation>new</operation> ```

```
<create-id>create1</create-id>
<operation>new</operation>
```

Answer: C. ```xml <edit-id>edit1</edit-id> <operation>create</operation> ```

Explanation

A YANG Patch edit is identified by an <edit-id>, and its operation must be <create> when the request is adding a new hostname configuration. Option C supplies both required elements, allowing the patch to create the hostname at the specified target with the value shown. This matches the requirement to add a new configuration rather than modify an existing one.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1715/b_1715_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/174/b_174_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1714/b_1714_programmability_cg/m_1714_prog_restconf.html

✗ **A** <edit-id>edit1</edit-id><operation>edit</operation>
The <edit-id> element is appropriate, but <operation>edit</operation> does not specify creating a new configuration. The scenario explicitly requires adding a new hostname.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1715/b_1715_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/174/b_174_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1714/b_1714_programmability_cg/m_1714_prog_restconf.html

✗ **B** <edit-id>edit1</edit-id><operation>new</operation>

Free demo · full exam: 549 questions

The edit identifier is correctly structured, but “new” is not the YANG Patch operation needed here. The operation should be “create” to add the hostname configuration.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1715/b_1715_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/174/b_174_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1714/b_1714_programmability_cg/m_1714_prog_restconf.html

✓ **C** <edit-id>edit1</edit-id><operation>create</operation>
This provides the required edit identifier and specifies the create operation. Creating is appropriate because the engineer is adding a new hostname configuration at the target path.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1715/b_1715_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/174/b_174_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1714/b_1714_programmability_cg/m_1714_prog_restconf.html

✗ **D** <create-id>create1</create-id><operation>new</operation>
This uses <create-id> rather than the required <edit-id>, and “new” is not the operation value for adding a YANG Patch edit. Therefore it does not correctly complete the patch edit.

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1715/b_1715_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/174/b_174_programmability_cg.pdf
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1714/b_1714_programmability_cg/m_1714_prog_restconf.html

Question 33

Single choice · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

An engineer is designing an application to interact with Cisco Catalyst Center (formerly DNA Center). The application must use multiple components that serve unique functions that range from device data collection, analysis, and monitoring to a custom dashboard. It must meet these development requirements:

- Application components are easy to replace.
- Application components are easy to re-use.
- It can scale up at short notice.
- It allows work to be done on specific components independently.

Which design practice must the engineer apply?

A. object-oriented design
B. modular design
C. single-responsibility principle
D. software configuration management

Answer: B. modular design

Explanation
The application has distinct components for collection, analysis, monitoring, and dashboards, and the requirements call for those parts to be independently replaceable, reusable, scalable, and developed

separately. Modular design organizes software into components with defined responsibilities and interfaces, making those forms of independent change and scaling practical. Therefore, modular design is the appropriate practice.

✗ **A** object-oriented design
Object-oriented design structures software around objects and their data and behavior. It does not, by itself, ensure the application's larger components can be independently replaced, reused, scaled, and worked on as required.

✓ **B** modular design
Modular design separates an application into distinct components, so a component can be developed or replaced without requiring changes to the whole application. This separation also supports reuse and scaling particular components when needed.

✗ **C** single-responsibility principle
The single-responsibility principle advises that a unit have one reason to change, which can help keep individual components focused. However, the question asks for the broader design practice that enables components to be replaceable, reusable, independently developed, and scaled; that is modular design.

✗ **D** software configuration management
Software configuration management tracks and controls changes to software artifacts and versions. It does not define an application architecture that separates functional components for independent replacement, reuse, and scaling.

Dumpsgate

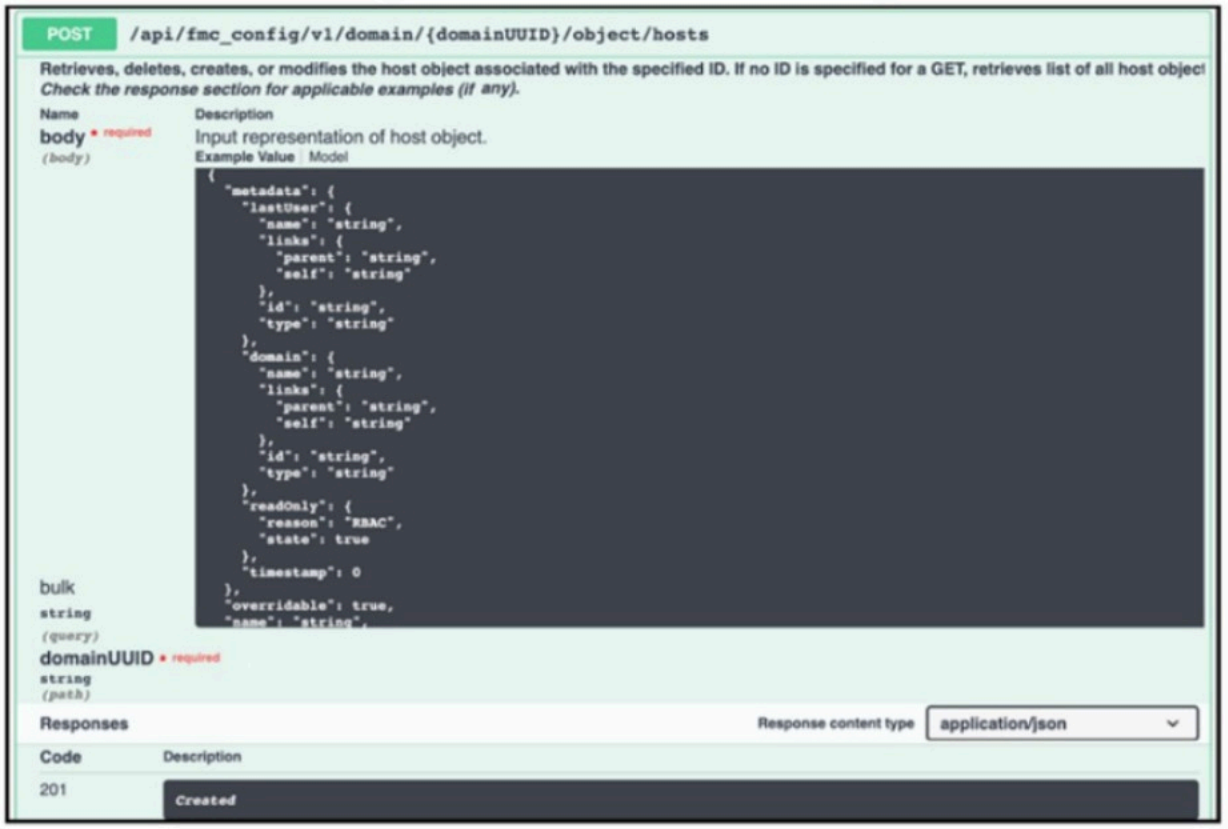
Question 34

Drag and drop · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

Refer to the exhibit.

The API documentation shows `POST /api/fmc_config/v1/domain/{domainUUID}/object/hosts`. It describes retrieving, deleting, creating, or modifying the host object associated with the specified ID. If no ID is specified for a GET, it retrieves a list of all host objects. The body is a required input representation of the host object. The response code shown is `201 Created`.

Free demo · full exam: 549 questions





Drag and drop the code from the bottom onto the box where the code is missing in the script to read from a .csv file and import the host entries. The script uses the Cisco Firepower API to create the appropriate entries. The script will be used as part of the security analysis dashboard application to visualize data about network security and host information. Not all options are used.

```
import requests, csv, json
```

```
from requests.auth import HTTPBasicAuth
address="fmcrestapisandbox.cisco.com"
username="networkuser"
password="cisco123"
api_uri = "/api/fmc_platform/v1/auth/generatetoken"
url = "https://" + address + api_uri
response = requests.request("POST", url, verify=False,
                            auth=HTTPBasicAuth(username, password))
accesstoken = response.headers["x-auth-access-token"]
refreshtoken = response.headers["x-auth-refresh-token"]
DOMAIN_UUID = response.headers["DOMAIN_UUID"]
csvFilePath = input("Please enter the CSV Filepath:")
host = []
csvReader = csv.DictReader(open(csvFilePath))
for rows in csvReader:
    if rows['type'] == "Host":
        host.append(rows)
host_payload = json.dumps(host)
host_api_uri = "/api/fmc_config/v1/domain/" + [Blank 1] +
               "/object/[Blank 2]"
host_url = "https://" + address + host_api_uri
headers = { 'Content-Type': 'application/json', 'x-auth-access-token': accesstoken }
response = requests.request("[Blank 3]", host_url, headers=headers,
                            data = host_payload, / verify = False)
if response.status_code == 201 or response.status_code == 202:
    print("Host Objects successfully pushed")
```

Options: `hosts`, `DOMAIN_UUID`, `PUT`, `POST`, `hosts?bulk=true`, `refreshtoken`

Items:

- A. hosts
- B. DOMAIN_UUID
- C. PUT
- D. POST
- E. hosts?bulk=true
- F. refreshtoken

Place them on:

1. Blank 1 (`/api/fmc_config/v1/domain/` + [Blank 1] +)
2. Blank 2 (`"/object/[Blank 2]"`)
3. Blank 3 (`requests.request("[Blank 3]", host_url, ...)`)

Answer

- Blank 1 (`/api/fmc_config/v1/domain/` + [Blank 1] +) → **B. DOMAIN_UUID**
- Blank 2 (`"/object/[Blank 2]"`) → **A. hosts**
- Blank 3 (`requests.request("[Blank 3]", host_url, ...)`) → **D. POST**

Explanation

The script constructs a host-object endpoint using the domain UUID returned by the token request, followed by the `/object/hosts` path shown in the exhibit. It must send the CSV-derived host payload with `POST` because the task is to create host entries, and the API documentation identifies `POST` as the creation operation with a `201 Created` response. Therefore, the correct values are `DOMAIN_UUID`, `hosts`, and `POST`.

https://www.cisco.com/c/en/us/support/docs/security/firepower-management-center/215972-push-objects-in-bulk-to-fmc-using-rest-a.pdf
https://www.cisco.com/c/en/us/td/docs/security/firepower/770/API/REST/secure_firewall_management_center_rest_api_quick_start_guide_770.pdf
https://www.cisco.com/c/en/us/td/docs/security/firepower/770/API/REST/secure_firewall_management_center_rest_api_quick_start_guide_770/Objects_In_The_REST_API.pdf

✓ **DOMAIN_UUID** DOMAIN_UUID
The script reads the domain UUID from the authentication response header and needs it in the path `/api/fmc_config/v1/domain/{domainUUID}/...`. This selects the domain in which to create the host objects.

https://www.cisco.com/c/en/us/td/docs/security/firepower/770/API/REST/secure_firewall_management_center_rest_api_quick_start_guide_770.pdf

✓ **hosts** hosts
The exhibit gives the host-object endpoint as `/object/hosts`, so `hosts` completes the object path. The `hosts?bulk=true` option is not the path shown in the exhibit for this request.

https://www.cisco.com/c/en/us/td/docs/security/firepower/770/API/REST/secure_firewall_management_center_rest_api_quick_start_guide_770.pdf
https://www.cisco.com/c/en/us/td/docs/security/firepower/770/API/REST/secure_firewall_management_center_rest_api_quick_start_guide_770/Objects_In_The_REST_API.pdf

✗ **PUT** PUT
The script is importing host entries by creating them, not modifying an existing object. The exhibit identifies `POST` as the creation operation, while `PUT` is not the method required here.

https://www.cisco.com/c/en/us/support/docs/security/firepower-management-center/215972-push-objects-in-bulk-to-fmc-using-rest-a.pdf
https://www.cisco.com/c/en/us/td/docs/security/firepower/770/API/REST/secure_firewall_management_center_rest_api_quick_start_guide_770.pdf

✓ **POST** POST
The requirement is to create host objects from the CSV payload, and the documented endpoint uses `POST` for creation. The exhibit's `201 Created` response is consistent with this method.

https://www.cisco.com/c/en/us/support/docs/security/firepower-management-center/215972-push-objects-in-bulk-to-fmc-using-rest-a.pdf
https://www.cisco.com/c/en/us/td/docs/security/firepower/770/API/REST/secure_firewall_management_center_rest_api_quick_start_guide_770.pdf

✗ **hosts?bulk=true** hosts?bulk=true
This option adds a bulk query parameter that is not part of the `/object/hosts` endpoint shown in the exhibit. The supplied answer uses the documented path without that parameter.

https://www.cisco.com/c/en/us/support/docs/security/firepower-management-center/215972-push-objects-in-bulk-to-fmc-using-rest-a.pdf
https://www.cisco.com/c/en/us/td/docs/security/firepower/770/API/REST/secure_firewall_management_center_rest_api_quick_start_guide_770.pdf

✗ **refreshtoken** refreshtoken
The endpoint path requires the domain UUID, not an authentication refresh token. The script already uses the access token in the request header for authorization.

https://www.cisco.com/c/en/us/td/docs/security/firepower/770/API/REST/secure_firewall_management_center_rest_api_quick_start_guide_770.pdf

Question 35

Multiple choice · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

A team is developing a cloud-native application. The project has these architecture requirements:

- Leverage the use of containers for effective usage of resources.
- Scale up the application automatically when more resources are needed.
- Ensure that the application is developed as a stateless application.

Which two guidelines must be met? (Choose two.)

A. Rely on the web server to help provide sticky sessions for web transactions.
B. Use Python class definition to leverage object-oriented techniques for development.
C. Ensure that all data that should persist is stored in a data store such as a database.
D. Use memory or disk space for temporary data that should not persist between nodes.
E. Develop the application using Python and virtual environments.

Answer

- C. **Ensure that all data that should persist is stored in a data store such as a database.**
- D. **Use memory or disk space for temporary data that should not persist between nodes.**

Explanation

A stateless application does not depend on a particular instance to retain data needed by later requests. Store data that must survive in an external data store, and use instance memory or disk only for temporary data that can be discarded when an instance is replaced or requests move to another node. These practices support automatic scaling and container-based deployment.

✗ **A** Rely on the web server to help provide sticky sessions for web transactions.
Sticky sessions direct a user's requests back to the same instance, making the application depend on that instance's state. That conflicts with stateless design and can complicate automatic scaling across nodes.

✗ **B** Use Python class definition to leverage object-oriented techniques for development.
Using classes is a programming-style choice, not a guideline for statelessness, container resource use, or automatic scaling. It does not ensure that application state is handled correctly across instances.

✓ **C** Ensure that all data that should persist is stored in a data store such as a database.
Persistent data belongs in a shared data store rather than in an individual container or application instance. This lets requests be handled by different instances as the application scales.

✓ **D** Use memory or disk space for temporary data that should not persist between nodes.
Instance memory or disk can hold temporary data when that data does not need to survive instance replacement or be available on another node. Keeping required persistent state out of local resources supports stateless operation and flexible scaling.

✗ **E** Develop the application using Python and virtual environments.
Python and virtual environments are implementation and dependency-management choices. They do not by themselves make the application stateless or enable it to scale automatically.

Free demo · full exam: 549 questions

Question 36

Multiple choice · Domain: D1 Network Automation › 1.4 Construct a network automation solution with Python to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

Refer to the exhibit.

```
from paramiko import SSHClient
from os import environ

host = ["N3172-TOR-01.widgets.com", "N3172-TOR-02.widgets.com", "N9336C-LEAF-01.widgets.com",
        "N31108-BORDER-LEAF-01.widgets.com"]
backup_server = "central-server-01.widget.com"


class ConnectionManager:

    def nc(u, p):
        client = SSHClient()
        return client.connect(host, username=u, password=p)

    def nc(key):
        client = SSHClient()
        return client.connect(host, pkey=key)

if __name__ == "__main__":
       cm = ConnectionManager()
       for i in host:
            try:
                if i.index("TOR") != -1:
                    conn = cm.nc(environ["PRIVATE_KEY"])
                else:
                    conn = cm.nc(environ["USER"], environ["PASSWD"])
                    conn.exec_command(f"copy running-config scp://{backup_server}/backups/{i}")
                except Exception as e:
                    print(f"The host {i} failed to backup properly. ({str(e)})")
                else:
                    conn.close()
```

A developer must review an intern's code for a script they wrote to automate backups to the storage server.

The script must connect to the network device and copy the running-config to the server. When considering maintainability, which two changes must be made to the code? (Choose two.)

A. Rename the class to 'ArchiveManager'.

B. The code is incorrect because the class does not have an__init__() method.

C. The command sent to the network device is incorrect.

D. Refactor the code placing the 'for' loop steps inside a single nc method.

E. The intern must use IP addresses because DNS is unreliable.

Answer

- C. **The command sent to the network device is incorrect.**
- D. **Refactor the code placing the 'for' loop steps inside a single nc method.**

Explanation

The script needs to be refactored so the connection and backup steps are handled consistently in one `nc`

method, rather than using two methods with the same name and separate logic in the loop. It also needs a corrected command/approach for copying the running configuration: as written, the code does not correctly issue a backup command through the returned SSH connection. Renaming the class, adding an initializer, or replacing DNS names with IP addresses does not address these problems.

✗ **A** Rename the class to 'ArchiveManager'.
The class name does not cause the backup failure or make the code harder to maintain in the way shown. Renaming it would not correct the connection handling or the backup command.

✗ **B** The code is incorrect because the class does not have an __init__() method.
A Python class does not need an `__init__()` method when it has no instance state that must be initialized. The shown code can instantiate the class without one, so this is not a required fix for the backup behavior.

✓ **C** The command sent to the network device is incorrect.
The code attempts to run a `copy running-config scp://...` command through `exec_command`, but its SSH setup does not provide a valid connected command session: Paramiko's `connect()` method does not return such a session object. The backup command and the way it is sent must be corrected for the script to copy the device configuration to the server.

✓ **D** Refactor the code placing the 'for' loop steps inside a single nc method.
The class defines `nc` twice, so the second definition replaces the first, and the loop separately chooses between authentication paths and performs backup work. Consolidating the connection and per-host backup steps into one method avoids duplicated and conflicting logic, making the code easier to maintain.

✗ **E** The intern must use IP addresses because DNS is unreliable.
Nothing in the scenario indicates that DNS resolution is failing, and the hostnames shown are not inherently unsuitable for this script. Replacing them with IP addresses would not fix the duplicated method or the command/connection problems.

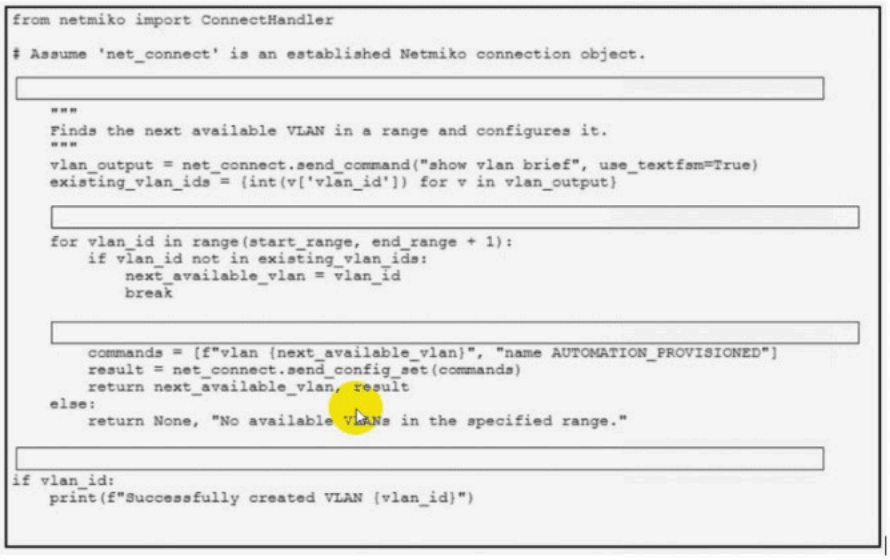
Dumpsgate

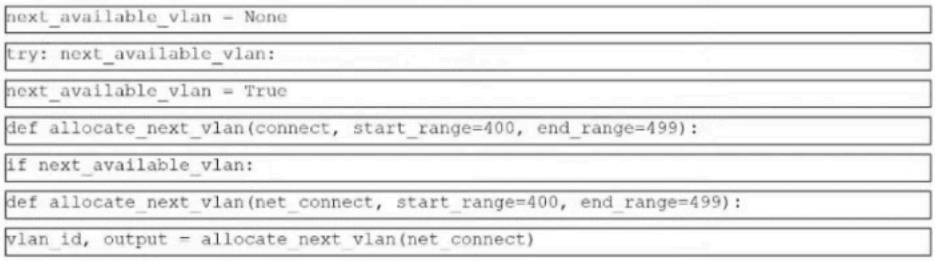
Question 37

Drag and drop · Domain: D1 Network Automation › 1.4 Construct a network automation solution with Python to manage configurations such as VLANs, OSPF, asset management, interface settings, and ACLs

To accelerate service delivery, the provisioning of new environments is being automated. A key requirement is the dynamic creation of a new VLAN. A script must intelligently find the next available VLAN ID within the 400-499 range and immediately configure the VLAN with a standard name. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the Python script. Not all options are used.

```
from netmiko import ConnectHandler

# Assume 'net_connect' is an established Netmiko connection object.
[Blank 1]
    """
    Finds the next available VLAN in a range and configures it.
    """
    vlan_output = net_connect.send_command("show vlan brief", use_textfsm=True)
    existing_vlan_ids = {int(v['vlan_id']) for v in vlan_output}

[Blank 2]
    for vlan_id in range(start_range, end_range + 1):
        if vlan_id not in existing_vlan_ids:
            next_available_vlan = vlan_id
            break

[Blank 3]
        commands = [f"vlan {next_available_vlan}", "name AUTOMATION_PROVISIONED"]
        result = net_connect.send_config_set(commands)
        return next_available_vlan, result
    else:
        return None, "No available VLANs in the specified range."

[Blank 4]
if vlan_id:
    print(f"Successfully created VLAN {vlan_id}")
```





Items:

- A. next_available_vlan = None
- B. try: next_available_vlan:
- C. next_available_vlan = True
- D. def allocate_next_vlan(net_connect, start_range=400, end_range=499):
- E. if next_available_vlan:
- F. def allocate_next_vlan(net_connect, start_range=400, end_range=499):
- G. vlan_id, output = allocate_next_vlan(net_connect)

Place them on:

1. Blank 1 (function definition before the docstring)
2. Blank 2 (initialization before the for loop)
3. Blank 3 (conditional before commands = [...])
4. Blank 4 (function call before if vlan_id:)

Answer

- Blank 1 (function definition before the docstring) → **D. def allocate_next_vlan(net_connect, start_range=400, end_range=499):**
- Blank 2 (initialization before the for loop) → **A. next_available_vlan = None**
- Blank 3 (conditional before commands = [...]) → **E. if next_available_vlan:**
- Blank 4 (function call before if vlan_id:) → **G. vlan_id, output = allocate_next_vlan(net_connect)**

Dumpsgate

Question 38

Single choice · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

An engineer creates an application that manages the parsing and analysis of data. The data is continuously generated in bursts and is disrupting the performance of the application by overloading the provided resources. The engineer wants to implement rate limiting on the receiving endpoint to ensure that a minimum data sample from every timeframe is parsed and that no more than a specific number of requests are processed each time. Which rate limiting algorithm must the engineer use?

A. leaky bucket
B. sliding window
C. fixed window
D. event queue

Free demo · full exam: 549 questions

Answer: A. leaky bucket

Explanation

The leaky bucket algorithm is the intended choice because it smooths bursty incoming traffic by regulating how requests are released for processing at a controlled rate. This helps prevent the receiving endpoint from being overloaded while limiting how much work it processes over time. The listed references do not clearly establish this application-level rate-limiting behavior, so no citations are included.

✓ **A** leaky bucket
The leaky bucket regulates bursty arrivals by allowing requests to be processed at a controlled rate, smoothing peaks that would otherwise overwhelm the application. This matches the requirement to limit requests reaching the receiving endpoint.

✗ **B** sliding window
A sliding window tracks requests across a moving time interval to enforce a limit. It is not the intended algorithm here for smoothing bursty input into a controlled processing rate.

✗ **C** fixed window
A fixed window counts requests within discrete time blocks, which can permit a burst near the boundary between two windows. It does not provide the intended smoothing of bursts described in the scenario.

✗ **D** event queue
An event queue can buffer work, but it is not the rate-limiting algorithm requested. By itself, it does not ensure that requests are processed at a controlled rate.

Dumpsgate

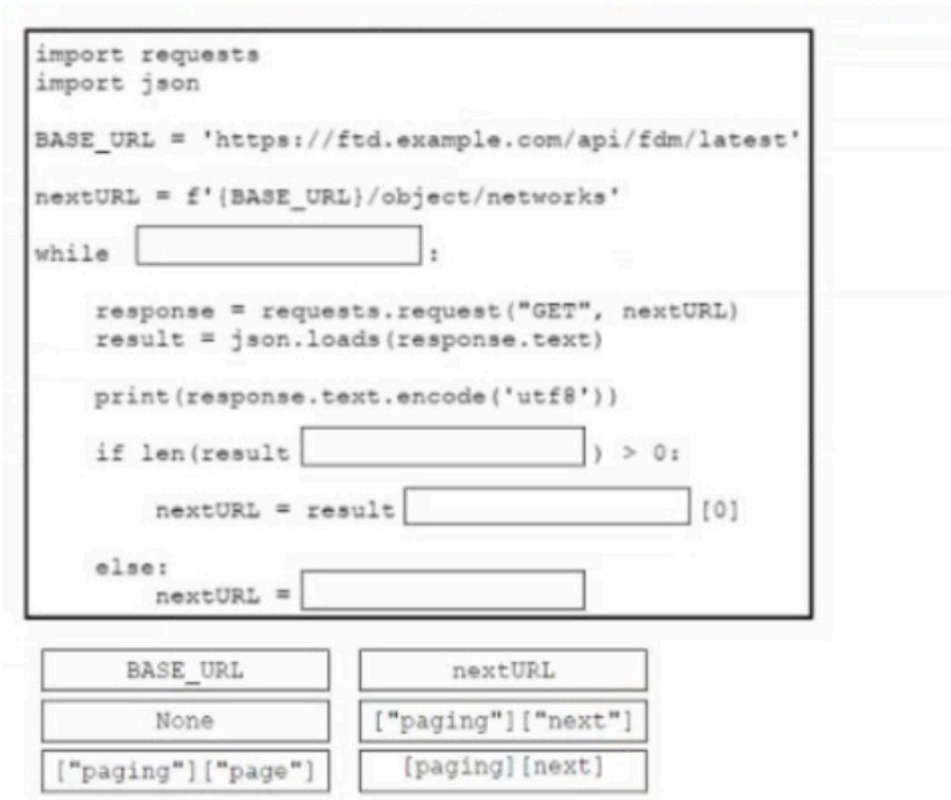
Question 39

Drag and drop · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

Drag and drop the code from the bottom onto the box where the code is missing to list objects that use the paginated Cisco FDM API. Some options may be used more than once. Not all options are used.

```
import requests
import json

BASE_URL = 'https://ftd.example.com/api/fdm/latest'

nextURL = f'{BASE_URL}/object/networks'

while [Blank 1]:
    response = requests.request("GET", nextURL)
    result = json.loads(response.text)

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

    if len(result [Blank 2]) > 0:
        nextURL = result [Blank 3][0]
    else:
        nextURL = [Blank 4]
```

Options: BASE_URL; nextURL; None; ["paging"]["next"]; ["paging"]["page"]; [paging][next]



Free demo · full exam: 549 questions

Items:

- A. BASE_URL
- B. nextURL
- C. None
- D. ["paging"]["next"]
- E. ["paging"]["page"]
- F. [paging][next]

Place them on:

1. Blank 1 (after `while`)
2. Blank 2 (in `if len(result [Blank 2]) > 0`)
3. Blank 3 (in `nextURL = result [Blank 3][0]`)
4. Blank 4 (in `else: nextURL = [Blank 4]`)

Answer

- Blank 1 (after `while`) → **B. nextURL**
- Blank 2 (in `if len(result [Blank 2]) > 0`) → **D. ["paging"]["next"]**
- Blank 3 (in `nextURL = result [Blank 3][0]`) → **D. ["paging"]["next"]**
- Blank 4 (in `else: nextURL = [Blank 4]`) → **C. None**

Explanation

The loop should continue while `nextURL` contains a URL to request. The response's `paging.next` field supplies the next-page link; checking its length determines whether another page is available, and its first entry becomes the next URL. When that list is empty, setting `nextURL` to `None` ends the loop. [2]

https://developer.cisco.com/docs/ftd-ansible/latest/paging/

✗ **BASE_URL** BASE_URL
This is the API base address, not the condition that indicates whether another request should be made. It is also not the pagination field containing the next-page link.

✓ **nextURL** nextURL
Using `nextURL` as the `while` condition continues the loop while a URL is available. Assigning `None` when there is no next page makes the condition false and stops pagination.

https://developer.cisco.com/docs/ftd-ansible/latest/paging/

✓ **None** None
When the response has no next-page link, assigning `None` to `nextURL` makes the loop condition false. This is the termination value for the pagination loop.

https://developer.cisco.com/docs/ftd-ansible/latest/paging/

✓ **["paging"]["next"]** ["paging"]["next"]
This accesses the response's `paging.next` value, which contains the next-page links. It is used to check whether any links remain and to select the first link for the next request.

https://developer.cisco.com/docs/ftd-ansible/latest/paging/

✗ **["paging"]["page"]** ["paging"]["page"]
The required field is `paging.next`, which provides the next-page link. `paging.page` is not the field used by this code to find that link.

https://developer.cisco.com/docs/ftd-ansible/latest/paging/

✗ **[paging][next]** [paging][next]
This is not valid Python dictionary-key syntax because `paging` and `next` are unquoted names. The response fields must be accessed with quoted string keys, as in `result["paging"]["next"]`.

Free demo · full exam: 549 questions

Question 40

Drag and drop · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

List Rooms

List rooms.

The title of the room for 1:1 rooms will be the display name of the other person.

By default, lists rooms to which the authenticated user belongs.

Long result sets will be split into pages.

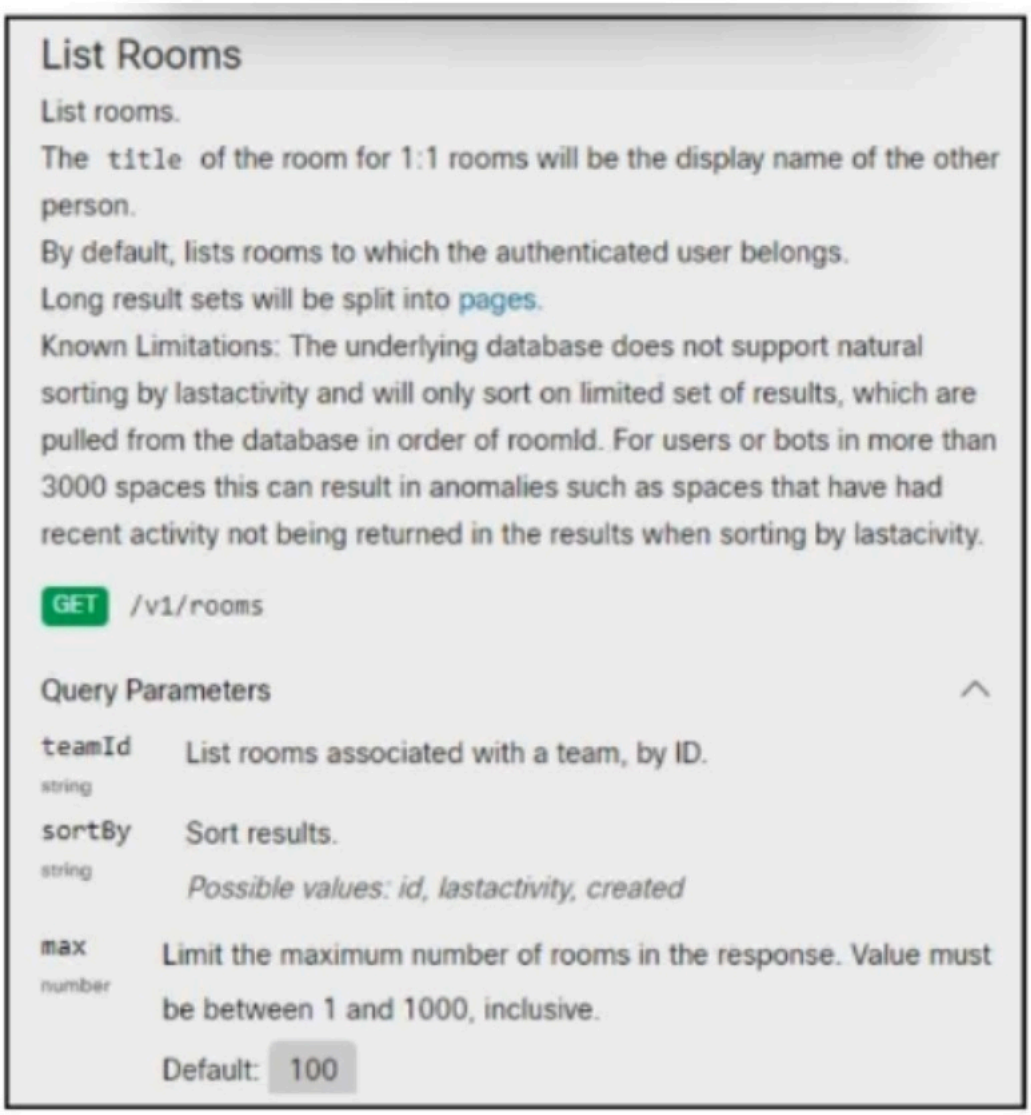
Known Limitations: The underlying database does not support natural sorting by lastactivity and will only sort on limited set of results, which are pulled from the database in order of roomId. For users or bots in more than 3000 spaces this can result in anomalies such as spaces that have had recent activity not being returned in the results when sorting by lastactivity.

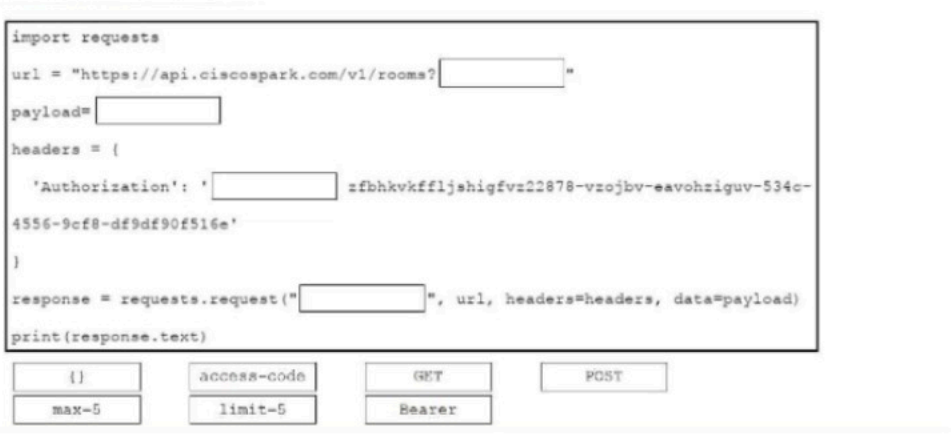
GET /v1/rooms

Query Parameters

- teamId (string): List rooms associated with a team, by ID.
- sortBy (string): Sort results. Possible values: id, lastactivity, created
- max (number): Limit the maximum number of rooms in the response. Value must be between 1 and 1000, inclusive. Default: 100

Dumpsgate





Refer to the exhibit. Drag and drop the code snippets from the bottom onto the blanks in the Python script to list all rooms and use pagination to restrict the number of results to five by using the Cisco Webex API. Not all options are used.

```
import requests

url = "https://api.ciscospark.com/v1/rooms?[Blank 1]"
```


Free demo · full exam: 549 questions

```
payload=[Blank 2]
headers = {
    'Authorization': '[Blank 3] fbhkvkffljshigfv=22878-vzojbv-eavohgigu-534c-4556-9cf8-df9d
}
response = requests.request("[Blank 4]", url, headers=headers, data=payload)
print(response.text)
```

Items:

- A. {}
- B. access-code
- C. GET
- D. POST
- E. max=5
- F. limit=5
- G. Bearer

Place them on:

1. Blank 1 (url query string after /v1/rooms?)
2. Blank 2 (payload=...)
3. Blank 3 (Authorization header value before the token)
4. Blank 4 (requests.request method)

Answer

- Blank 1 (url query string after /v1/rooms?) → **E. max=5**
- Blank 2 (payload=...) → **A. {}**
- Blank 3 (Authorization header value before the token) → **G. Bearer**
- Blank 4 (requests.request method) → **C. GET**

Explanation

The endpoint is GET /v1/rooms, so the request method is GET and no request body is needed, represented by an empty dictionary. The exhibit specifies the `max` query parameter to set the response limit, and `max=5` restricts the results to five rooms. The authorization token must be prefixed with `Bearer`.

✓ **Blank 1 (url query string after /v1/rooms?) — max=5** max=5
The exhibit identifies `max` as the parameter that limits the maximum number of rooms returned. Setting it to 5 meets the requested limit.

✓ **Blank 2 (payload=...) — {}** {}
Listing rooms is a GET request and does not require a request body. An empty dictionary supplies no payload.

✓ **Blank 3 (Authorization header value before the token) — Bearer** Bearer
The authorization header uses the `Bearer` scheme before the access token. This supplies the required prefix for the token shown in the script.

✓ **Blank 4 (requests.request method) — GET** GET
The listed rooms operation in the exhibit is `GET /v1/rooms`. Therefore, the request method must be GET rather than POST.

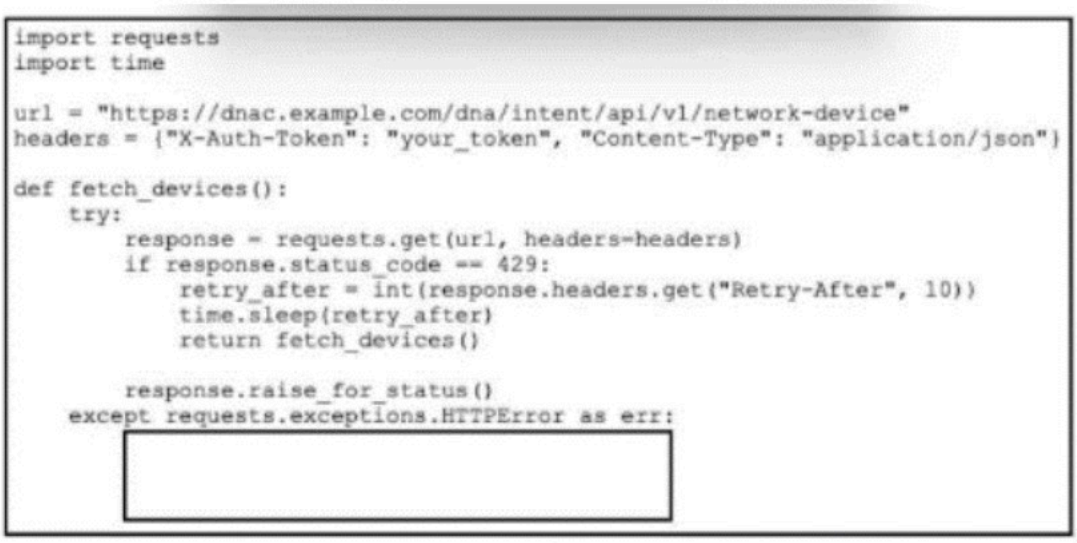
Question 41

Single choice · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

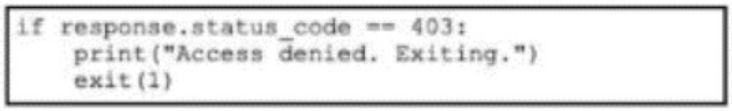
Refer to the exhibit.

```
import requests
import time

url = "https://dnac.example.com/dna/intent/api/v1/network-device"
headers = {"X-Auth-Token": "your_token", "Content-Type": "application/json"}

def fetch_devices():
    try:
        response = requests.get(url, headers=headers)
        if response.status_code == 429:
            retry_after = int(response.headers.get("Retry-After", 10))
            time.sleep(retry_after)
            return fetch_devices()

        response.raise_for_status()
    except requests.exceptions.HTTPError as err:
        [Blank 1]
```

Refer to the exhibit. An engineer must improve error handling for a script calling /dna/intent/api/v1/network-device. It should exit on 403 and retry after a delay on 429. Which code snippet must be placed on the box in the code to complete the script?



A. if response.status_code == 403: print("Access denied. Exiting.") exit(1)



Free demo · full exam: 549 questions

B. if response.status_code == 403: time.sleep(60) return fetch_devices()

```
if response.status_code == 403:
    time.sleep(60)
    return fetch_devices()
```

C. if response.status_code == 403: print("Permission denied.") return None

```
if response.status_code == 403:
    print("Permission denied.")
    return None
```

D. if response.status_code == 403: print("Rate limited. Waiting default time.") return fetch_devices()

```
if response.status_code == 403:
    print("Rate limited. Waiting default time.")
    return fetch_devices()
```

Answer: A. if response.status_code == 403: print("Access denied. Exiting.") exit(1)

Explanation

The code already handles HTTP 429 by waiting for the Retry-After value (or a default) and retrying. The missing handler should therefore treat HTTP 403 as a permission denial that terminates the script; option A prints an access-denied message and exits with a nonzero status. The other options either retry a forbidden request or fail to exit as required.

✓ **A** if response.status_code == 403: print("Access denied. Exiting.") exit(1)
This detects the 403 response, reports that access was denied, and exits with a nonzero status. That satisfies the requirement to exit on 403 while leaving the existing 429 retry logic intact.

✗ **B** if response.status_code == 403: time.sleep(60) return fetch_devices()
This waits and retries after a 403 instead of exiting. A 403 indicates access is forbidden, whereas the script's existing retry behavior is specifically for 429 responses.

✗ **C** if response.status_code == 403: print("Permission denied.") return None
This prints an appropriate message but returns from the function rather than exiting the script. It therefore does not meet the explicit requirement to exit on 403.

✗ **D** if response.status_code == 403: print("Rate limited. Waiting default time.") return fetch_devices()
This incorrectly treats 403 as rate limiting and retries the request. It neither exits on 403 nor matches the existing 429-specific retry handling.

Question 42

Single choice · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

A developer is designing a modem, distributed microservice enterprise application. The application will be integrating with other systems and focus on a large deployment, so control of API calls is necessary. What is the best practice to reduce application response latency and protect the application from excessive use?

A. Implement rate limiting on the client side.
B. Implement rate limiting on the client and server sides.
C. Do not enforce any rate Imiting
D. Implement rate limiting on the server side.

Answer: B. Implement rate limiting on the client and server sides.

Explanation

Implementing rate limiting on both the client and server best meets the need to control API calls in a large, integrated deployment. Client-side limits can reduce unnecessary requests before they are sent, helping avoid added latency; server-side limits provide enforcement when clients send too many requests. This combination helps protect the application from excessive use.

✗ **A** Implement rate limiting on the client side.
Client-side limits can reduce unnecessary calls from a well-behaved client, but they cannot reliably protect the application on their own. A client can bypass or misconfigure its limit, so server-side enforcement is also needed.

✓ **B** Implement rate limiting on the client and server sides.
Client-side limits can curb unnecessary requests before they reach the application, while server-side limits enforce request thresholds at the point of service. Using both helps control load and protect the application from excessive use in a large deployment.

✗ **C** Do not enforce any rate Imiting
Without rate limiting, clients can make excessive API calls, increasing load and potentially worsening response latency. This does not satisfy the requirement to control API usage and protect the application.

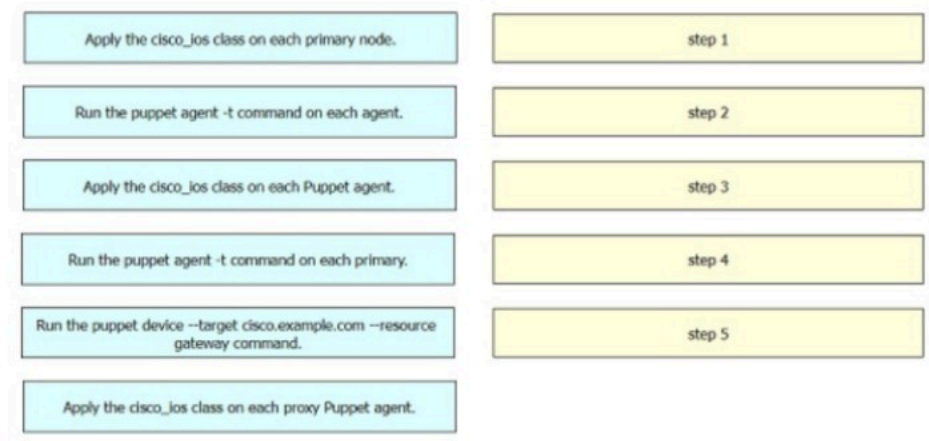
✗ **D** Implement rate limiting on the server side.
Server-side limits are important for enforcing usage thresholds, but this option omits client-side limiting. The intended best practice is to limit calls on both sides, so clients can avoid unnecessary requests while the server remains protected.

Free demo · full exam: 549 questions

Question 43

Drag and drop · Domain: D1 Network Automation › 1.5 Select the network automation approach to achieve technical and business requirements considering options such as infrastructure as code framework, low code/no code, and custom applications

Drag and drop the steps from the left into the order on the right to create the workflow to retrieve the gateway information from a set of Cisco IOS devices. Not all options are used.



Items:

- A. Apply the cisco_ios class on each primary node.
- B. Run the puppet agent -t command on each agent.
- C. Apply the cisco_ios class on each Puppet agent.
- D. Run the puppet agent -t command on each primary.
- E. Run the puppet device --target cisco.example.com --resource gateway command.
- F. Apply the cisco_ios class on each proxy Puppet agent.

Place them on:

1. Step 1
2. Step 2
3. Step 3
4. Step 4
5. Step 5

Answer

- Step 1 → **A. Apply the cisco_ios class on each primary node.**
- Step 2 → **F. Apply the cisco_ios class on each proxy Puppet agent.**
- Step 3 → **D. Run the puppet agent -t command on each primary.**
- Step 4 → **B. Run the puppet agent -t command on each agent.**
- Step 5 → **E. Run the puppet device --target cisco.example.com --resource gateway command.**

Explanation

The workflow first configures the Cisco IOS class on the primary nodes and the proxy Puppet agents that will manage the IOS devices. Run the Puppet agent on the primary nodes and then on each agent to apply the configurations. Finally, run the puppet device command for the Cisco device and gateway resource to retrieve its gateway information.

✓ **Box 1: Apply the cisco_ios class on each primary node.** Apply the cisco_ios class on each primary node.
This is Step 1: configure the class on the primary nodes before running the Puppet agent commands. It prepares the primary-side configuration for the workflow.

✓ **Box 2: Run the puppet agent -t command on each agent.** Run the puppet agent -t command on each agent.
This is Step 4, after the primary-side configuration has been applied. Running the command on each agent applies the configuration needed before the device command is run.

✗ **Box 3: Apply the cisco_ios class on each Puppet agent.** Apply the cisco_ios class on each Puppet agent.
This is not the class-placement step specified in the workflow. The class must be applied to each proxy Puppet agent, rather than to each Puppet agent generally.

✓ **Box 4: Run the puppet agent -t command on each primary.** Run the puppet agent -t command on each primary.
This is Step 3, after applying the class on the primary nodes and proxy Puppet agents. Running the command on each primary applies its configuration before proceeding to the agents.

✓ **Box 5: Run the puppet device --target cisco.example.com --resource gateway command.** Run the puppet device --target cisco.example.com --resource gateway command.
This is Step 5, the final step. It targets the Cisco device and specifies the gateway resource whose information the workflow must retrieve.

✓ **Box 6: Apply the cisco_ios class on each proxy Puppet agent.** Apply the cisco_ios class on each proxy Puppet agent.
This is Step 2, after applying the class on the primary nodes. The proxy Puppet agents need the class before the agent runs and the final device command is issued.

Dumpsgate

Question 44

Single choice · Domain: D1 Network Automation › 1.6 Construct a network automation solution that consumes REST APIs including extended API attributes (such as pagination, complex authentication workflows, and rate limiting), error handling, and persistent authentication

Refer to the exhibit.

```
import requests
import time

token = 'XXXXX'
headers = {
    "Authorization": f"Bearer {token}"
}
url = "https://webexapis.com/v1/rooms"

def try_request(url,headers):
    request = requests.request('GET', url = url,headers = headers)
    if request.status_code == 200:
        successful_response = request.json()
        return successful_response
    elif request.status_code in [Blank 1]:
        time.sleep(60)
        try_request(url,headers)
    else:
        raise Exception('Failed Request')
```

Refer to the exhibit. An engineer creates a function that gathers information from the Cisco Webex RESTful API. To enable increased durability, the engineer needs to implement a retry mechanism that pauses the script and retries the request if specific response codes are received. Which code snippet must be placed in the blank in the code to meet this requirement?



A. 400,403,409,504

Free demo · full exam: 549 questions

400,403,409,504

B. 400,404,502,503

400,404,502,503

C. 403,404,409,429

403,404,409,429

D. 429,502,503,504

429,502,503,504

Answer: D. 429,502,503,504

Explanation

The function should pause and retry for response codes that indicate rate limiting or a transient server-side failure. Option D lists 429 (Too Many Requests) and 502, 503, and 504 (gateway/server errors), which are appropriate conditions for a delayed retry. The other choices include client or request errors that generally will not be fixed simply by waiting.

✗ **A** 400, 403, 409, 504
This list includes 400 (a bad request), 403 (forbidden), and 409 (a conflict), which typically require changing the request or permissions rather than retrying after a delay. It includes only one transient gateway failure, 504, and omits rate limiting and other common transient server responses.

✗ **B** 400, 404, 502, 503
Although 502 and 503 can indicate transient server-side failures, 400 and 404 generally indicate a malformed request or a resource that cannot be found. Waiting 60 seconds will not normally correct those client-side conditions, and this list omits 429 and 504.

✗ **C** 403, 404, 409, 429
The list includes 429, which signals rate limiting and can justify waiting before retrying. However, 403, 404, and 409 generally require addressing authorization, resource, or request conflicts, and the list leaves out transient gateway/server failures.

✓ **D** 429, 502, 503, 504
These codes represent rate limiting or transient gateway/server failures, for which pausing and retrying can be appropriate. The list therefore matches the function's intended retry behavior without treating ordinary client/request errors as retryable.

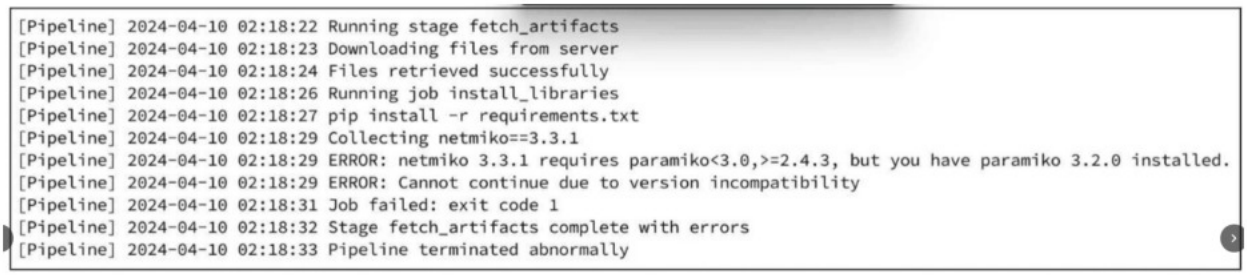
Question 45

Single choice · Domain: D2 Infrastructure as Code › 2.2 Diagnose a GitLab CE CI/CD pipeline failure such as missing dependency, incompatible versions of components, and failed tests

Refer to the exhibit.

```
[Pipeline] 2024-04-10 02:18:22 Running stage fetch_artifacts
[Pipeline] 2024-04-10 02:18:23 Downloading files from server
[Pipeline] 2024-04-10 02:18:24 Files retrieved successfully
[Pipeline] 2024-04-10 02:18:26 Running job install_libraries
[Pipeline] 2024-04-10 02:18:27 pip install -r requirements.txt
[Pipeline] 2024-04-10 02:18:29 Collecting netmiko==3.3.1
[Pipeline] 2024-04-10 02:18:29 ERROR: netmiko 3.3.1 requires paramiko<3.0,>=2.4.3, but you
[Pipeline] 2024-04-10 02:18:29 ERROR: Cannot continue due to version incompatibility
[Pipeline] 2024-04-10 02:18:31 Job failed: exit code 1
[Pipeline] 2024-04-10 02:18:32 Stage fetch_artifacts complete with errors
[Pipeline] 2024-04-10 02:18:33 Pipeline terminated abnormally
```

Refer to the exhibit The network infrastructure team designed a Gitlab pipeline that initializes standard base templates before initiating session configuration tasks for integration tasks Recent project updates included downstream requirement adjustments and significant upgrades to core frameworks and external dependencies. Despite these changes builds execute successfully until the install libraries step fails with a critical error. Previous pipeline runs completed with no errors. The job runners and artifact storage were validated as healthy after the failure event What is the cause of the issue?



A. An incorrect environment variable was injected at runtime
B. The registry URL override was missing for dependency pull
C. The deployment log in the artifact handler was truncated
D. An incompatible package was referenced during dependency resolution

Answer: D. An incompatible package was referenced during dependency resolution

Explanation

The pipeline fails during dependency installation because the resolved versions are incompatible: netmiko 3.3.1 requires paramiko below 3.0, while paramiko 3.2.0 is installed. This directly matches the error in the exhibit and indicates an incompatible package combination in dependency resolution, rather than a runner, artifact-storage, or logging problem.

✗ **A** An incorrect environment variable was injected at runtime
The log identifies a specific version conflict between netmiko and paramiko, not an environment-variable issue. No incorrect variable is reported in the exhibit.

✗ **B** The registry URL override was missing for dependency pull
The packages were retrieved far enough for pip to report their versions and compatibility requirements. The failure shown is a dependency version conflict, not a registry URL or package-download failure.

✗ **C** The deployment log in the artifact handler was truncated
The exhibit includes the decisive error message explaining why installation stopped. It reports a package compatibility failure, not missing or truncated deployment logs.

✓ **D** An incompatible package was referenced during dependency resolution
The log states that netmiko 3.3.1 requires paramiko below version 3.0, but paramiko 3.2.0 is installed. That incompatibility causes pip to stop and the job to fail.

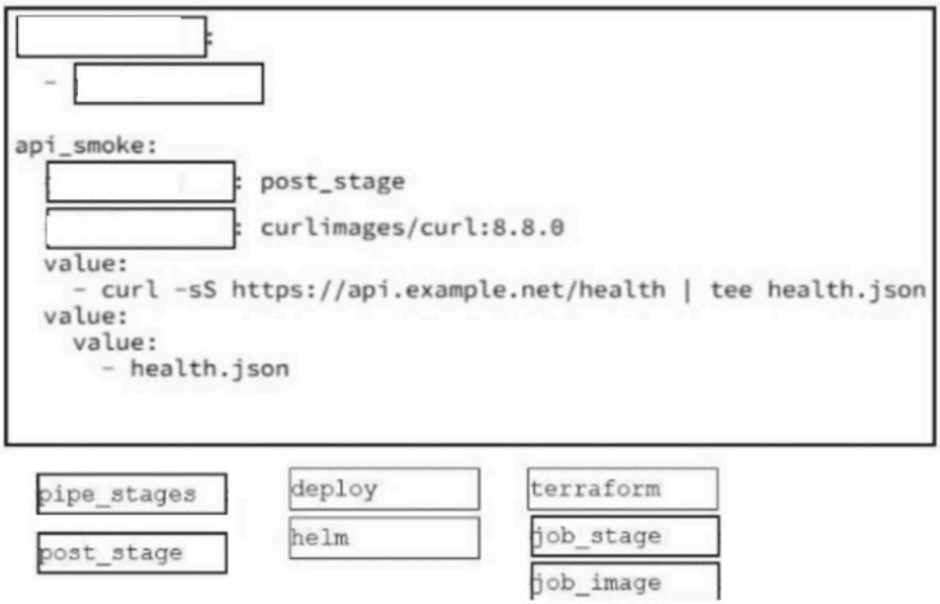
Dumpsgate

Question 46

Drag and drop · Domain: D2 Infrastructure as Code › 2.3.d post-validation

An engineer must add a post-validation stage that performs an HTTP smoke test of the freshly deployed API. The engineer must declare a post-validation stage, run curl against a health endpoint, and publish the response body as an artifact. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.

```
[Blank 1]:
  - [Blank 2]
api_smoke:
  [Blank 3]: post_stage
  [Blank 4]: curlimages/curl:8.8.0
  value:
    - curl -sS https://api.example.net/health | tee health.json
  value:
    value:
      - health.json
```



Items:

- A. pipe_stages
- B. post_stage
- C. deploy
- D. helm
- E. terraform
- F. job_stage
- G. job_image

Place them on:

1. Blank 1 (top-level key before the colon)
2. Blank 2 (list item under Blank 1)
3. Blank 3 (key before `: post_stage` in `api_smoke`)
4. Blank 4 (key before `: curlimages/curl:8.8.0` in `api_smoke`)

Answer

- Blank 1 (top-level key before the colon) → **A. pipe_stages**
- Blank 2 (list item under Blank 1) → **B. post_stage**
- Blank 3 (key before `: post_stage` in `api_smoke`) → **F. job_stage**
- Blank 4 (key before `: curlimages/curl:8.8.0` in `api_smoke`) → **G. job_image**

Explanation

`pipe_stages` declares the pipeline's stage list, and `post_stage` adds the requested post-validation stage. Within `api_smoke`, `job_stage` identifies the job's stage and `job_image` specifies the curl container image. The curl command performs the HTTP health check and tees its response body to `health.json`, which the artifact configuration publishes.

✓ **pipe_stages** pipe_stages
This is the top-level key for declaring pipeline stages. It lets the configuration include the post-validation stage in the pipeline.

✓ **post_stage** post_stage
This is the stage name required for the post-validation job. Listing it under `pipe_stages` makes it available for `api_smoke` to use.

✗ **deploy** deploy
The requested stage is a post-validation stage, not a deployment stage. The freshly deployed API is the target of the smoke test, so `deploy` does not fit this job's role.

✗ **helm** helm
`helm` is not the required stage-list key, stage name, or job field for this configuration. The specified job uses the curl image and command to test the API.

✗ **terraform** terraform
`terraform` does not declare the pipeline stages or identify the job's stage or container image. It is not needed for the HTTP smoke test or artifact publication shown.

✓ **job_stage** job_stage
This field assigns `api_smoke` to the `post_stage` stage. That places the health check after deployment, as required.

✓ **job_image** job_image
This field selects `curlimages/curl:8.8.0` as the job's container image. That image provides curl for running the health-endpoint request.

Dumpsgate

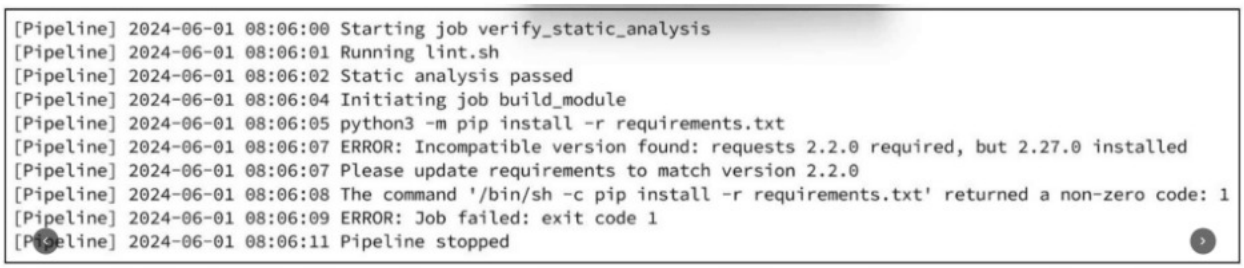
Question 47

Single choice · Domain: D2 Infrastructure as Code › 2.2 Diagnose a GitLab CE CI/CD pipeline failure such as missing dependency, incompatible versions of components, and failed tests

Refer to the exhibit.

```
[Pipeline] 2024-06-01 08:06:00 Starting job verify_static_analysis
[Pipeline] 2024-06-01 08:06:01 Running lint.sh
[Pipeline] 2024-06-01 08:06:02 Static analysis passed
[Pipeline] 2024-06-01 08:06:04 Initiating job build_module
[Pipeline] 2024-06-01 08:06:05 python3 -m pip install -r requirements.txt
[Pipeline] 2024-06-01 08:06:07 ERROR: Incompatible version found: requests 2.2.0 required,
[Pipeline] 2024-06-01 08:06:07 Please update requirements to match version 2.2.0
[Pipeline] 2024-06-01 08:06:08 The command '/bin/sh -c pip install -r requirements.txt' ret
[Pipeline] 2024-06-01 08:06:09 ERROR: Job failed: exit code 1
[Pipeline] 2024-06-01 08:06:11 Pipeline stopped
```

Refer to the exhibit. The deployment pipeline executes baseline input verification and then applies advanced module logic scans to validate automation changes throughout the codebase The code and related configuration files were committed after a recent feature implementation The process includes automated environment configuration and package installation before each pipeline run After the most recent changeset. the pipeline fails and halts prior to the modulo build completion Logistics such as runner availability and accessible artifacts have been independently confirmed What caused the pipeline problem?



A. A version mismatch is detected in the dependency constraint
B. Multiple branch triggers occur in the pipeline settings.
C. Container user privileges are insufficient for the build.
D. The scheduled job window overlaps with code deployment

Answer: A. A version mismatch is detected in the dependency constraint

Explanation

The pipeline passes static analysis, then fails during `pip install -r requirements.txt` in the build job. The log explicitly reports that `requests` version 2.2.0 is required but version 2.27.0 is installed, so the dependency version constraint is incompatible and causes the install command to exit with an error before the build completes.

✓ **A** A version mismatch is detected in the dependency constraint
The log identifies a `requests` version mismatch: the requirements specify 2.2.0 while 2.27.0 is installed. The package-install command then returns a non-zero exit code, stopping the pipeline before the module build completes.

✗ **B** Multiple branch triggers occur in the pipeline settings.
The failure occurs during package installation, and the log gives a specific dependency-version error. It contains no indication that branch triggers started duplicate or conflicting pipeline runs.

✗ **C** Container user privileges are insufficient for the build.
The install command fails because the installed `requests` version does not match the required version, not because of a permissions error. The log reports no access-denied or privilege-related failure.

✗ **D** The scheduled job window overlaps with code deployment
The pipeline proceeds through static analysis and reaches the build job, where package installation fails on the version mismatch. Nothing in the log suggests a scheduling conflict or overlap with deployment.

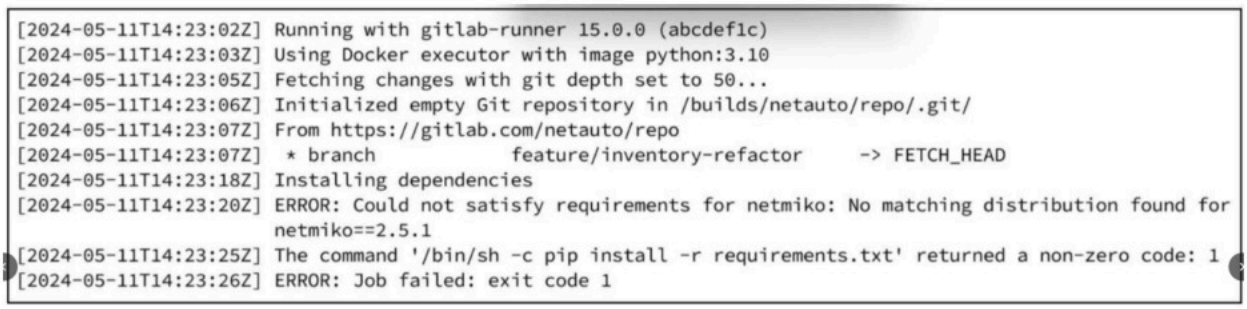
Question 48

Single choice · Domain: D2 Infrastructure as Code › 2.2 Diagnose a GitLab CE CI/CD pipeline failure such as missing dependency, incompatible versions of components, and failed tests

Refer to the exhibit.

```
[2024-05-11T14:23:02Z] Running with gitlab-runner 15.0.0 (abcdef1c)
[2024-05-11T14:23:03Z] Using Docker executor with image python:3.10
[2024-05-11T14:23:05Z] Fetching changes with git depth set to 50...
[2024-05-11T14:23:06Z] Initialized empty Git repository in /builds/netauto/repo/.git/
[2024-05-11T14:23:07Z] From https://gitlab.com/netauto/repo
[2024-05-11T14:23:07Z] * branch                  feature/inventory-refactor -> FETCH_HEAD
[2024-05-11T14:23:18Z] Installing dependencies
[2024-05-11T14:23:20Z] ERROR: Could not satisfy requirements for netmiko: No matching distr
[2024-05-11T14:23:25Z] The command '/bin/sh -c pip install -r requirements.txt' returned a
[2024-05-11T14:23:26Z] ERROR: Job failed: exit code 1
```

Refer to the exhibit A build pipeline for a network automation project using feature/inventory-refactor repeatedly failed during dependency installation The logs indicate that the job fails specifically when processing component netmiko The development team updated their requirements without additional testing Code updates were pushed this morning and the previous pipeline succeeded on a different runner The error occurs regardless of variable substitutions in environment files What is the root cause for the failed dependency installation?



A. The version does not exist in the repository index for the required package
B. An incorrect interpreter version is specified for the virtual environment path.
C. The package mirror address is unreachable during network resolution
D. The file configuration contains a syntax mismatch that affects the package install.

Answer: A. The version does not exist in the repository index for the required package

Explanation

The log identifies the exact requirement pip cannot install: `netmiko==2.5.1`, and reports “No matching distribution found.” This indicates that the pinned version is not available from the package index being used, consistent with the team changing requirements without testing. The successful run on another runner and the failure being independent of environment-variable substitutions do not point to the other proposed causes.

✓ **A** The version does not exist in the repository index for the required package
The failure names the exact pinned requirement, `netmiko==2.5.1`, and says no matching distribution was found. This supports that the requested version is unavailable from the package index used by the job.

✗ **B** An incorrect interpreter version is specified for the virtual environment path.
The exhibit shows the job using a `python:3.10` Docker image, but it does not show a virtual-environment path or an interpreter-path error. The reported failure is specifically that pip cannot find the requested Netmiko version.

✗ **C** The package mirror address is unreachable during network resolution
The log does not report a network-resolution or connection error; it reports that no matching distribution was found for `netmiko==2.5.1`. The package index or mirror may have been contacted, but the evidence points to the requested version being unavailable rather than unreachable.

✗ **D** The file configuration contains a syntax mismatch that affects the package install.
A syntax mismatch would typically produce a requirements-file parsing or configuration error. Here, pip identifies a specific package and pinned version but cannot find a matching distribution, so the shown failure is not a syntax error.

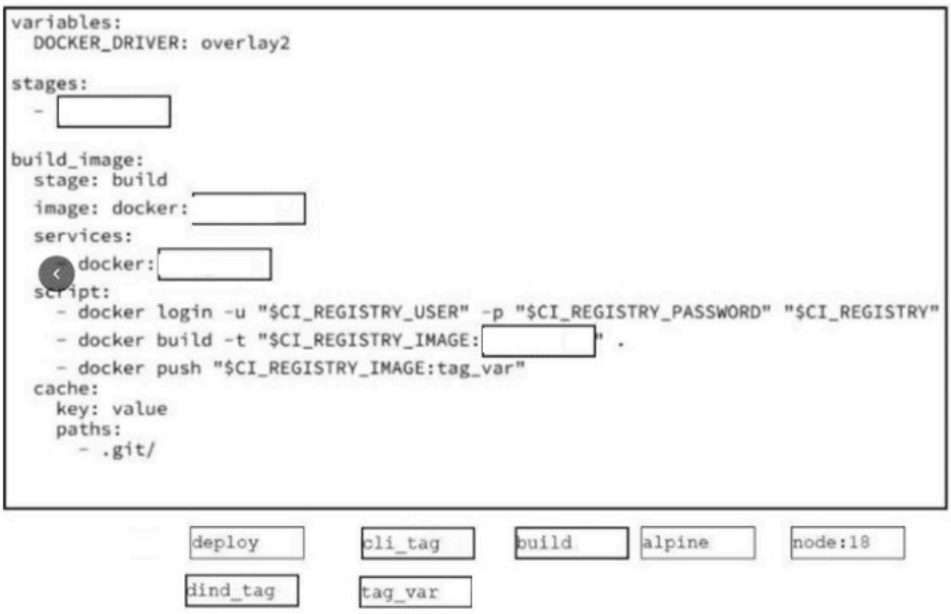
Dumpsgate

Question 49

Drag and drop · Domain: D2 Infrastructure as Code › 2.3.a build

An engineer must define a GitLab CE pipeline that builds a container image for a network automation tool. The engineer must use Docker-in-Docker service, set the correct stage and script to build and push with a variable registry name, and cache the build context. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.

```
variables:
  DOCKER_DRIVER: overlay2

stages:
  - [Blank 1]

build_image:
  stage: build
  image: docker:[Blank 2]
  services:
    - docker:[Blank 3]
  script:
    - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" "$CI_REGISTRY"
    - docker build -t "$CI_REGISTRY_IMAGE:[Blank 4]" .
    - docker push "$CI_REGISTRY_IMAGE:tag_var"
  cache:
    key: value
    paths:
      - .git/
```



Free demo · full exam: 549 questions

Items:

- A. deploy
- B. cli_tag
- C. build
- D. alpine
- E. node:18
- F. dind_tag
- G. tag_var

Place them on:

1. Blank 1 (the item under stages)
2. Blank 2 (image: docker:[Blank 2])
3. Blank 3 (services: docker:[Blank 3])
4. Blank 4 (docker build -t "$CI_REGISTRY_IMAGE:[Blank 4]" .)

Answer

- Blank 1 (the item under stages) → **C. build**
- Blank 2 (image: docker:[Blank 2]) → **B. cli_tag**
- Blank 3 (services: docker:[Blank 3]) → **F. dind_tag**
- Blank 4 (docker build -t "$CI_REGISTRY_IMAGE:[Blank 4]" .) → **G. tag_var**

Explanation

Use `build` as the pipeline stage because the job builds the container image. The Docker CLI image uses the `cli_tag` suffix, while the Docker-in-Docker service uses `dind_tag`, matching their distinct roles. For the build command, `tag_var` supplies the image tag so the build and the existing push command refer to the same registry image tag.

✗ **Box 1: deploy** deploy
The stage is for building the container image, not deploying it. The provided `build` choice matches the job's purpose.

✓ **Box 2: cli_tag** cli_tag
This completes the Docker CLI image reference as `docker:cli_tag`. The job needs a Docker image to run its Docker commands.

✗ **Box 3: build** build
This would make the service reference `docker:build`, rather than selecting the Docker-in-Docker service. The `dind_tag` choice is intended for that service.

✗ **Box 4: alpine** alpine
This would create the image reference `docker:alpine` for the job's Docker CLI image, not the specified CLI-tagged image. The correct suffix for this box is `cli_tag`.

✗ **Box 5: node:18** node:18
This is not a Docker image tag suffix suited to the `docker:` reference shown in the job. The pipeline needs the Docker CLI image, represented by `cli_tag`.

✓ **Box 6: dind_tag** dind_tag
This completes the service reference as `docker:dind_tag`, selecting the Docker-in-Docker service required by the scenario. It belongs in the `services` entry, not the job image.

✓ **Box 7: tag_var** tag_var
This makes the build command tag the image as `$CI_REGISTRY_IMAGE:tag_var`. That tag matches the one in the existing `docker push` command, so the built image is the one pushed.

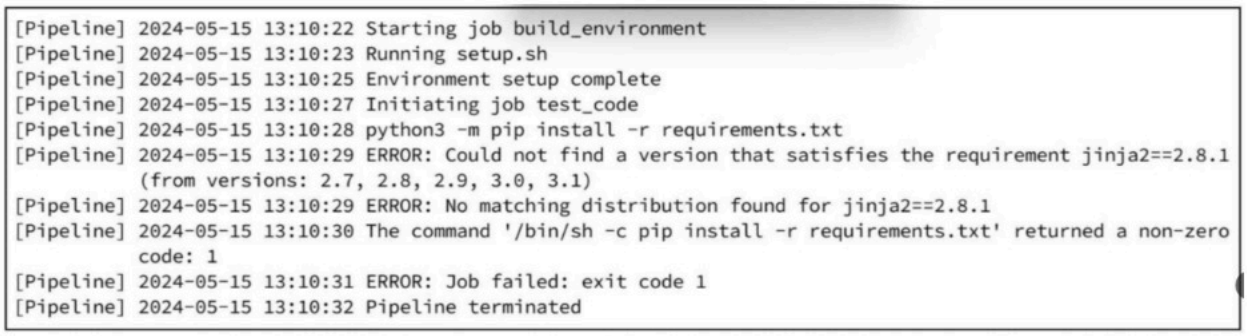
Question 50

Single choice · Domain: D2 Infrastructure as Code › 2.2 Diagnose a GitLab CE CI/CD pipeline failure such as missing dependency, incompatible versions of components, and failed tests

Refer to the exhibit.

```
[Pipeline] 2024-05-15 13:10:22 Starting job build_environment
[Pipeline] 2024-05-15 13:10:23 Running setup.sh
[Pipeline] 2024-05-15 13:10:25 Environment setup complete
[Pipeline] 2024-05-15 13:10:27 Initiating job test_code
[Pipeline] 2024-05-15 13:10:28 python3 -m pip install -r requirements.txt
[Pipeline] 2024-05-15 13:10:29 ERROR: Could not find a version that satisfies the requireme
    (from versions: 2.7, 2.8, 2.9, 3.0, 3.1)
[Pipeline] 2024-05-15 13:10:29 ERROR: No matching distribution found for jinja2==2.8.1
[Pipeline] 2024-05-15 13:10:30 The command '/bin/sh -c pip install -r requirements.txt' ret
    code: 1
[Pipeline] 2024-05-15 13:10:31 ERROR: Job failed: exit code 1
[Pipeline] 2024-05-15 13:10:32 Pipeline terminated
```

Refer to the exhibit The pipeline is configured to deploy automated firewall rules and performs additional validation using static compliance checks before the deployment is complete Recently the automation scripts were updated and committed to the shared repository A new dependency was introduced into the requirements and was pushed to the remote branch along with other unrelated changes After the latest pipeline was run an error was produced and the test phase does not proceed to subsequent stages The team verified network connectivity and confirmed that the container images are up to date Which issue caused the pipeline failure?



A. requirement version unavailable in package repository
B. disabling automatic renewal of secure communications
C. establishing unencrypted communication endpoints
D. path invalid in pipeline variable syntax

Answer: A. requirement version unavailable in package repository

Explanation

The pipeline fails when pip tries to install `jinja2==2.8.1` from `requirements.txt`. The error explicitly lists the available versions and does not include 2.8.1, so the pinned requirement cannot be satisfied and the

install command exits with code 1. This prevents the test job from proceeding, regardless of network connectivity or whether the container image is up to date.

✓ **A** requirement version unavailable in package repository
The log says no version satisfies `jinja2==2.8.1` and lists available versions as 2.7, 2.8, 2.9, 3.0, and 3.1. Because the exact pinned version is unavailable, pip fails and the pipeline stops.

✗ **B** disabling automatic renewal of secure communications
The failure occurs during installation of a Python package, and the error identifies an unavailable Jinja2 version. Nothing in the log indicates a secure-communications renewal problem.

✗ **C** establishing unencrypted communication endpoints
The pipeline error is a package-version resolution failure, not a communication endpoint or encryption issue. The scenario also says network connectivity was verified.

✗ **D** path invalid in pipeline variable syntax
The pipeline successfully runs the install command and pip reads the requirements file far enough to report the specific unavailable Jinja2 version. The log does not indicate a malformed path or variable syntax.

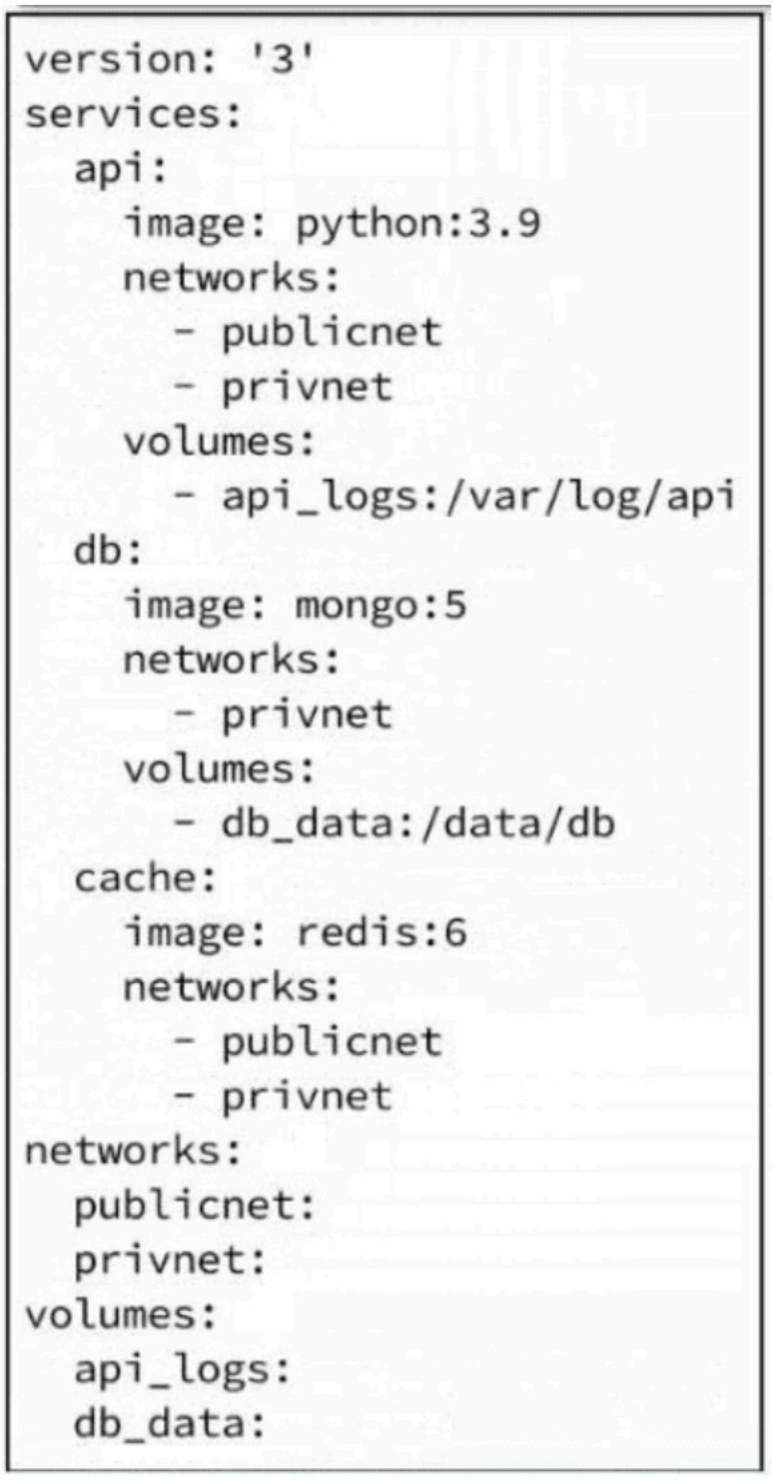
Question 51

Single choice · Domain: D2 Infrastructure as Code › 2.5 Interpret a Docker Compose file including services, networks, volumes, and links

Refer to the exhibit.

```
version: '3'
services:
  api:
    image: python:3.9
    networks:
      - publicnet
      - privnet
    volumes:
      - api_logs:/var/log/api
  db:
    image: mongo:5
    networks:
      - privnet
    volumes:
      - db_data:/data/db
  cache:
    image: redis:6
    networks:
      - publicnet
      - privnet
networks:
  publicnet:
  privnet:
volumes:
  api_logs:
  db_data:
```

Refer to the exhibit The deployment team prepared the docker config considering service uptime security, and access from other services They want to ensure that the configuration allows integration with an external network Based on the Docker Compose configuration what should the team expect regarding the network setup?



```
version: '3'
services:
  api:
    image: python:3.9
    networks:
      - publicnet
      - privnet
    volumes:
      - api_logs:/var/log/api
  db:
    image: mongo:5
    networks:
      - privnet
    volumes:
      - db_data:/data/db
  cache:
    image: redis:6
    networks:
      - publicnet
      - privnet
networks:
  publicnet:
  privnet:
volumes:
  api_logs:
  db_data:
```

A. cache uses db_data volume
B. API is configured with two networks simultaneously
C. publicnet links cache and db together.
D. DB is configured with two networks simultaneously

Answer: B. API is configured with two networks simultaneously

Question 52

Single choice · Domain: D2 Infrastructure as Code › 2.1.a Merge a branch including squash and conflict resolution

A project team must maintain code consistency across several feature developments using forked development for collaborative workflows. Recently, multiple contributors finished their work and pushed to remote repositories. Three-way reconciliation is the most efficient strategy to ensure a single commit history during integration and keep accurate conflict detection for critical files. What must be used to meet the requirements?

A. conflict-aware merge with squash
B. shallow clone adjustment
C. hard reset after pull
D. recursive submodule update

Answer: A. conflict-aware merge with squash

Explanation

A conflict-aware squash merge combines the contributors' changes into a single commit on the target branch while using merge logic to identify conflicts that need resolution. This fits the requirement for one streamlined commit history without giving up conflict detection for critical files. The other options do not perform this integration strategy.

✓ **A** conflict-aware merge with squash
Squashing combines the merged changes into one commit, keeping the target branch's history concise. Conflict-aware merge behavior can still detect conflicting edits so contributors can resolve them during integration.

✗ **B** shallow clone adjustment

Free demo · full exam: 549 questions

A shallow clone limits how much repository history is fetched; adjusting it does not integrate contributors' changes or create a single commit for them. It does not meet the conflict-detection requirement by itself.

✗ **C** hard reset after pull
A hard reset moves the working tree and index to a selected commit, potentially discarding local changes. It is not a collaborative merge strategy and does not reconcile contributors' work with conflict detection.

✗ **D** recursive submodule update
Updating submodules synchronizes nested repository contents to recorded revisions. It does not merge feature contributions into one commit history or provide the required reconciliation of their changes.

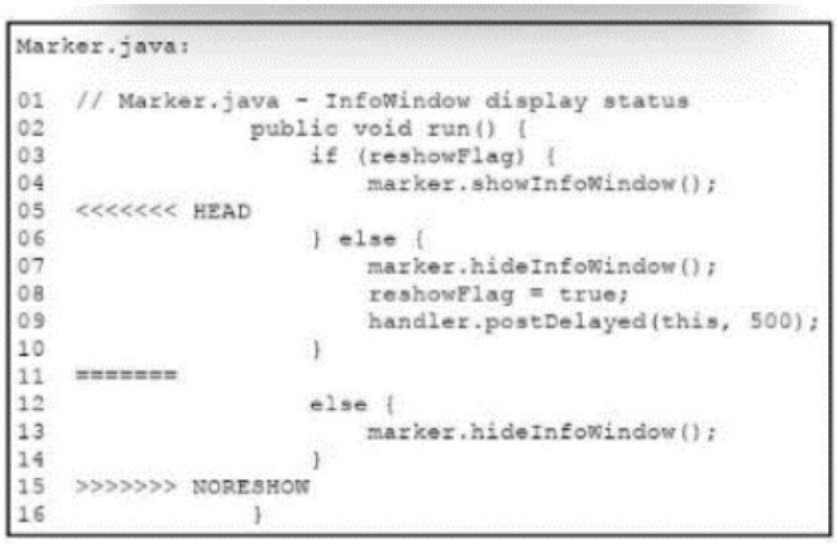
Question 53

Single choice · Domain: D2 Infrastructure as Code › 2.1.a Merge a branch including squash and conflict resolution

Refer to the exhibit.

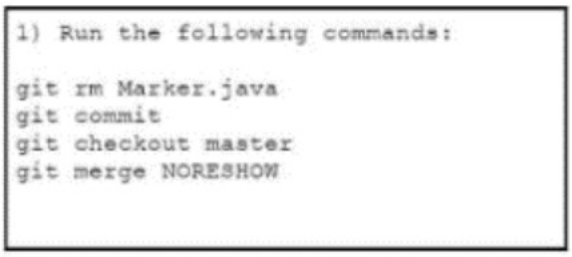
```
Marker.java:
01  // Marker.java - InfoWindow display status
02      public void run() {
03          if (reshowFlag) {
04              marker.showInfoWindow();
05  <<<<<<< HEAD
06          } else {
07              marker.hideInfoWindow();
08              reshowFlag = true;
09              handler.postDelayed(this, 500);
10          }
11  =======
12          else {
13              marker.hideInfoWindow();
14          }
15  >>>>>>> NORESHOW
16      }
```

Refer to the exhibit. Which set of actions resolves the conflict in the MarkerJava file to accept the incoming changes?

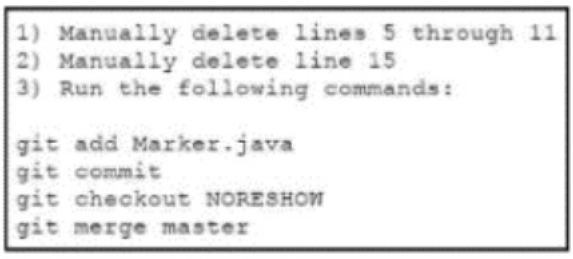


Free demo · full exam: 549 questions

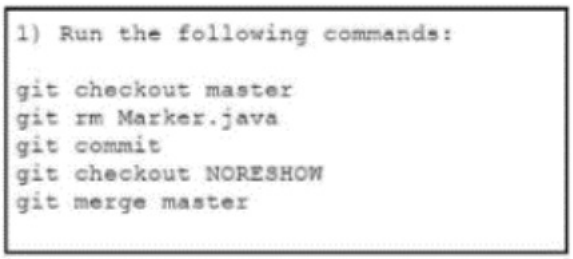
A. 1) Run the following commands: git rm Marker.java git commit git checkout master git merge NORESHOW



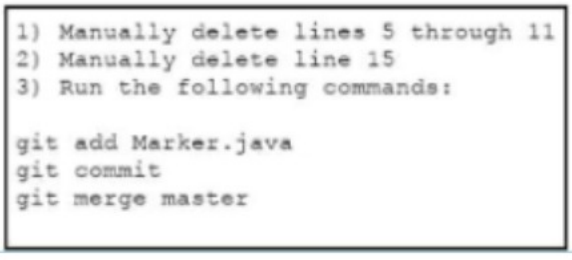
B. 1) Manually delete lines 5 through 11 2) Manually delete line 15 3) Run the following commands: git add Marker.java git commit git checkout NORESHOW git merge master



C. 1) Run the following commands: git checkout master git rm Marker.java git commit git checkout NORESHOW git merge master



D. 1) Manually delete lines 5 through 11 2) Manually delete line 15 3) Run the following commands: git add Marker.java git commit git merge master



Dumpsgate

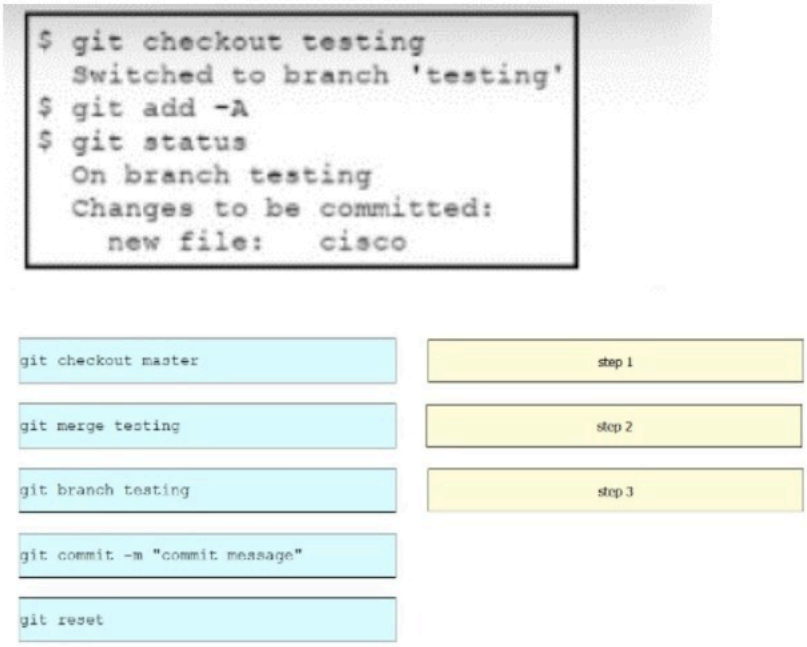
Question 54

Drag and drop · Domain: D2 Infrastructure as Code › 2.1.a Merge a branch including squash and conflict resolution

Refer to the exhibit.

```
$ git checkout testing
  Switched to branch 'testing'
$ git add -A
$ git status
  On branch testing
  Changes to be committed:
    new file:   cisco
```

Refer to the exhibit. A new file named Cisco is added to a branch named testing, but it has not been committed. Drag and drop the Git commands from the left into the order on the right to merge the testing branch, with the new Cisco file, into the master branch. Not all options are used.



Items:

- A. git checkout master
- B. git merge testing
- C. git branch testing
- D. git commit -m "commit message"
- E. git reset

Place them on:

1. Step 1
2. Step 2
3. Step 3

Answer

- Step 1 → **D. git commit -m "commit message"**
- Step 2 → **A. git checkout master**
- Step 3 → **B. git merge testing**

Explanation

The new file is staged on the testing branch but is not yet committed, so commit it before switching branches. Then check out master and merge testing to bring the branch's committed changes, including the Cisco file, into master. Reference [1] supports the Git workflow involved in committing and merging changes.

https://netascode.cisco.com/docs/labs/sdwan/reference/understanding-git/

✓ **git checkout master** git checkout master
Use this after committing the staged file to switch to the branch that must receive the changes. The merge should be run while master is checked out.

https://netascode.cisco.com/docs/labs/sdwan/reference/understanding-git/

✓ **git merge testing** git merge testing
Run this on master to merge the committed changes from testing into it. This brings the new Cisco file into master along with the other changes on testing.

https://netascode.cisco.com/docs/labs/sdwan/reference/understanding-git/

✗ **git branch testing** git branch testing
The exhibit shows that the testing branch already exists and is currently checked out. Creating it again is unnecessary and does not commit the staged file or merge it into master.

✓ **git commit -m "commit message"** git commit -m "commit message"
The Cisco file is staged but has not been committed, so commit it while still on testing before attempting the merge. A merge brings committed branch history across; it does not substitute for committing this staged file first.

https://netascode.cisco.com/docs/labs/sdwan/reference/understanding-git/

✗ **git reset** git reset
Reset is not the required step: the staged Cisco file needs to be committed so it can be included in the merge. Using reset could unstage the change rather than create the commit needed for this workflow.

Question 55

Drag and drop · Domain: D2 Infrastructure as Code › 2.7 Construct a YAML or JSON representation of a network configuration given a YANG-based data model

Refer to the exhibit.

```
module ex-device-if-multiaddr {
  namespace "urn:ex:dev:if-multiaddr";
  prefix exm;

  container interfaces {
    list interface {
      key "name";
      leaf name { type string; }
      container ipv4 {
        list address {
          key "ip";
          leaf ip { type string; }
          leaf prefix-length { type uint8; }
        }
      }
    }
  }
}
```

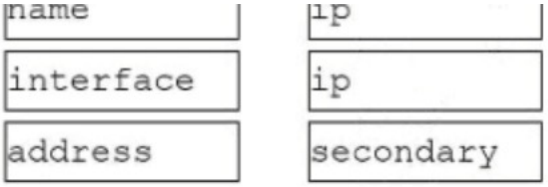
Dumpsgate

```
module ex-device-if-multiaddr {
  namespace "urn:ex:dev:if-multiaddr";
  prefix exm;

  container interfaces {
    list interface {
      key "name";
      leaf name { type string; }
      container ipv4 {
        list address {
          key "ip";
          leaf ip { type string; }
          leaf prefix-length { type uint8; }
        }
      }
    }
  }
}
```

```
interfaces:
  [          ]:
    - [          ]: "GigabitEthernet0/1"
      ipv4:
        [          ]:
          - [          ]: "198.51.100.1"
            prefix-length: 30
          - [          ]: "198.51.100.5"
            prefix-length: 30
```

primary mask

Free demo · full exam: 549 questions



Two IPv4 addresses must be set on interface Gi0/1. The model shows an ipv4 container with an address list keyed by ip with prefix-length leaf. The engineer must produce valid YAML under the correct containers.

Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.

```
interfaces:
  [Blank 1]:
    - [Blank 2]: "GigabitEthernet0/1"
      ipv4:
        [Blank 3]:
          - [Blank 4]: "198.51.100.1"
            prefix-length: 30
          - [Blank 5]: "198.51.100.5"
            prefix-length: 30
```

Code snippets: `primary`, `mask`, `name`, `ip`, `interface`, `ip`, `address`, `secondary`.

Items:

- A. primary
- B. mask
- C. name
- D. ip
- E. interface
- F. ip
- G. address
- H. secondary

Place them on:

1. Blank 1 (`interfaces:`)
2. Blank 2 (list item under `interfaces:`)
3. Blank 3 (`ipv4:` container)
4. Blank 4 (first address list item)
5. Blank 5 (second address list item)

Answer

- Blank 1 (`interfaces:`) → **E. interface**
- Blank 2 (list item under `interfaces:`) → **C. name**
- Blank 3 (`ipv4:` container) → **G. address**
- Blank 4 (first address list item) → **D. ip**
- Blank 5 (second address list item) → **D. ip**

Explanation

The YANG model defines `interface` as the list under `interfaces`, keyed by `name`, so the first two blanks must be `interface` and `name`. Inside each interface, the `ipv4` container holds an `address` list, whose entries are keyed by `ip`; therefore `address` fills Blank 3 and `ip` fills both address-entry blanks. The `primary`, `mask`, and `secondary` snippets do not match any node in the model and are unused.

✓ **interface** interface
This names the list nested under `interfaces` in the YANG model. The YAML list entry for the interface belongs under this key.

✓ **name** name
The `interface` list is keyed by `name`, so this is the key for the list item containing `GigabitEthernet0/1`.

✓ **ip** ip
The `address` list is keyed by `ip`, so this snippet supplies the key for one address entry. Both occurrences of the `ip` snippet are needed, one for each address.

✓ **ip** ip
This is the key leaf for an entry in the `address` list. The model uses `ip` for both address entries, so this second occurrence fills the other address blank.

✗ **primary** primary
The YANG model has no `primary` node in the interface or IPv4 address structure. Using it would make the YAML key inconsistent with the modeled data.

✗ **mask** mask
The modeled address entry contains an `ip` leaf and a `prefix-length` leaf, not a `mask` leaf. The prefix length is already provided in the YAML.

✓ **address** address
The model defines an `address` list inside the `ipv4` container. This key introduces the list that holds the two address entries.

✗ **secondary** secondary
There is no `secondary` node in the displayed YANG model. The second address is another entry in the same `address` list, not a separately named secondary field.

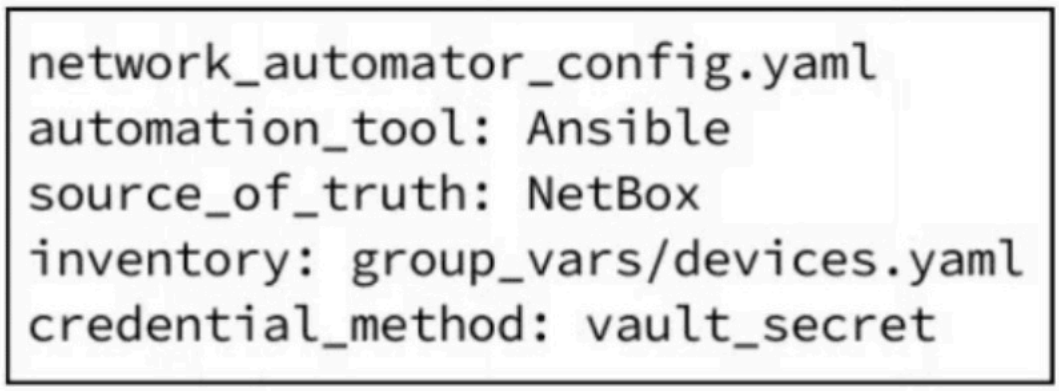
Dumpsgate

Question 56

Single choice · Domain: D2 Infrastructure as Code › 2.6 Integrate source of truth into a network automation solution

Refer to the exhibit.

```
network_automator_config.yaml
automation_tool: Ansible
source_of_truth: NetBox
inventory: group_vars/devices.yaml
credential_method: vault_secret
```



A network automation solution is being constructed to integrate inventory data using a workflow orchestrator for orchestration and a REST API service as the source of the data. The infrastructure team organized variable files for device management. When building the integration workflow, which action must be taken to ensure that the orchestration layer can reliably access the inventory data from the designated system?

A. Update a template filter.
B. Configure an HTTP connection.
C. Use a configuration lookup.
D. Declare a static variable set.

Answer: B. Configure an HTTP connection.

Explanation

The designated inventory source is NetBox, described in the scenario as a REST API service. To let the workflow orchestrator access data from that service, configure an HTTP connection for the API. The Ansible inventory file and vault setting in the exhibit do not replace the connection needed to reach the external source.

✗ **A** Update a template filter.
A template filter transforms or formats values that are already available to a workflow. It does not establish access to the NetBox REST API, which is the requirement in this scenario.

✓ **B** Configure an HTTP connection.
NetBox is identified as the source of inventory data and is accessed through a REST API. Configuring an HTTP connection gives the orchestration workflow a way to communicate with that API and retrieve the

inventory data.

✗ **C** Use a configuration lookup.
A configuration lookup is not the action that provides connectivity to the designated REST API source. The workflow needs an HTTP connection to reach NetBox and request its inventory data.

✗ **D** Declare a static variable set.
A static variable set would provide fixed values rather than reliably retrieving inventory from the designated REST API. The scenario requires access to NetBox as the source of truth, so an HTTP connection is needed.

Dumpsgate

Question 57

Drag and drop · Domain: D2 Infrastructure as Code › 2.7 Construct a YAML or JSON representation of a network configuration given a YANG-based data model

Refer to the exhibit. The L3VPN service must include two customer sites. The model shows sites list keyed by site-id under each l3vpn record. The engineer must construct YAML with both sites under the proper list. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used. Options may be used more than once.

YANG model:

```
module ex-svc-l3vpn-sites {
  namespace "urn:ex:svc:l3vpn:sites";
  prefix svs;

  container services {
    list l3vpn {
      key "name";
      leaf name { type string; }
      list sites {
        key "site-id";
        leaf site-id { type string; }
        leaf ce-interface { type string; }
        leaf pe { type string; }
        container ipv4 {
          leaf address { type string; }
          leaf mask-length { type uint8; }
        }
      }
    }
  }
}
```

Complete the YAML:

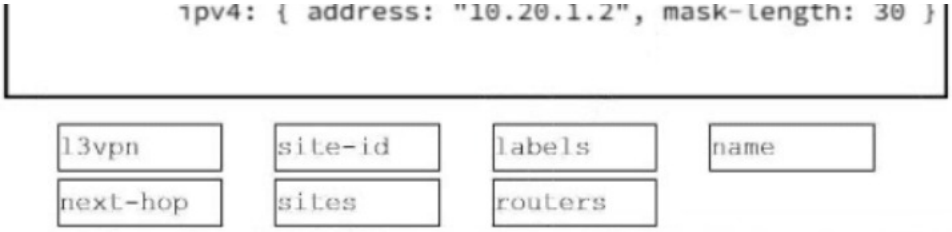
```
services:
  [Blank 1]:
    - [Blank 2]: "vpn-b"
      customer: "Globex"
      [Blank 3]:
        - [Blank 4]: "nyc1"
          ce-interface: "GigabitEthernet0/0"
          pe: "pe-nyc-1"
          ipv4: { address: "10.20.0.2", mask-length: 30 }
        - [Blank 5]: "dal1"
          ce-interface: "GigabitEthernet0/1"
          pe: "pe-dal-1"
          ipv4: { address: "10.20.1.2", mask-length: 30 }
```

Available code snippets: l3vpn, site-id, labels, name, next-hop, sites, routers.

Free demo · full exam: 549 questions

```
module ex-svc-l3vpn-sites {
  namespace "urn:ex:svc:l3vpn:sites";
  prefix svs;

  container services {
    list l3vpn {
      key "name";
      leaf name { type string; }
      list sites {
        key "site-id";
        leaf site-id { type string; }
        leaf ce-interface { type string; }
        leaf pe { type string; }
        container ipv4 {
          leaf address { type string; }
          leaf mask-length { type uint8; }
        }
      }
    }
  }
}
```

```
services:
  [redacted]:
    - [redacted]: "vpn-b"
      customer: "Globex"
      [redacted]:
        - [redacted]: "nyc1"
          ce-interface: "GigabitEthernet0/0"
          pe: "pe-nyc-1"
          ipv4: { address: "10.20.0.2", mask-length: 30 }
        - [redacted]: "dal1"
          ce-interface: "GigabitEthernet0/1"
          pe: "pe-dal-1"
```



Items:

- A. l3vpn
- B. site-id
- C. labels
- D. name
- E. next-hop
- F. sites
- G. routers

Place them on:

1. Blank 1 (services: [Blank 1]:)
2. Blank 2 (list entry key before "vpn-b")
3. Blank 3 (list containing the customer sites)
4. Blank 4 (first site entry key before "nyc1")
5. Blank 5 (second site entry key before "dal1")

Answer

- Blank 1 (services: [Blank 1]:) → **A. l3vpn**
- Blank 2 (list entry key before "vpn-b") → **D. name**
- Blank 3 (list containing the customer sites) → **F. sites**
- Blank 4 (first site entry key before "nyc1") → **B. site-id**
- Blank 5 (second site entry key before "dal1") → **B. site-id**

Explanation

The YANG model places the L3VPN records in the `l3vpn` list under `services`, with each record keyed by `name`. Each record contains a `sites` list keyed by `site-id`, so the YAML uses `sites` for the customer-site collection and `site-id` for both site entries. This puts both required sites under the `vpn-b` L3VPN record.

https://developer.cisco.com/docs/nso/guides/the-yang-data-modeling-language/

✓ **Blank 1: l3vpn** l3vpn
The model defines `l3vpn` as the list directly inside the `services` container. Therefore, the YAML list under `services` must be named `l3vpn`.

https://developer.cisco.com/docs/nso/guides/the-yang-data-modeling-language/

✓ **Blank 2: name** name
The `l3vpn` list is keyed by `name` in the model. The value `vpn-b` is the name of this L3VPN list entry, so the entry uses `name` as its key field.

https://developer.cisco.com/docs/nso/guides/the-yang-data-modeling-language/

✓ **Blank 3: sites** sites
Each `l3vpn` record contains a list named `sites`. This is the correct list for placing both customer-site entries beneath the `vpn-b` record.

https://developer.cisco.com/docs/nso/guides/the-yang-data-modeling-language/

Free demo · full exam: 549 questions

✓ **Blank 4: site-id** site-id
The `sites` list is keyed by `site-id`, so the first site's identifier `nyc1` must be assigned to `site-id`.
https://developer.cisco.com/docs/nso/guides/the-yang-data-modeling-language/

✓ **Blank 5: site-id** site-id
The second entry is also a member of the `sites` list, which uses `site-id` as its key. Thus `dal1` must also appear under `site-id`.
https://developer.cisco.com/docs/nso/guides/the-yang-data-modeling-language/

✗ **Option: labels** labels
The shown model defines no `labels` node under `services`, `l3vpn`, or `sites`. It cannot name either the L3VPN list or the customer-sites list in this YAML.

✗ **Option: next-hop** next-hop
The shown model defines no `next-hop` node, and this option does not match the list names or key leaves required to represent the two sites.

✗ **Option: routers** routers
The model defines the relevant nested list as `sites`, not `routers`. Using `routers` would not place the entries under the modeled customer-site list.

Question 58

Single choice · Domain: D2 Infrastructure as Code › 2.1.c git reset

While maintaining a continuous delivery pipeline, engineers detect errors just before a new deployment The staging area must be updated to discard certain modifications, but all previous changes must remain intact and unaltered What successfully rolls back those specific changes?

A. fast-forward merge strategy
B. soft reset application
C. commit history rewind
D. hard reset command

Answer: B. soft reset application

Explanation

A soft reset moves the current branch tip back while leaving the index and working-tree contents unchanged. This preserves the existing changes rather than erasing or rewriting them, making it the intended choice when rolling back specific changes while keeping prior work intact. [4]
https://netascode.cisco.com/docs/labs/sdwan/reference/understanding-git/

✗ **A** fast-forward merge strategy
A fast-forward merge advances a branch pointer to include another branch's commits; it does not roll back selected changes or preserve the current state by resetting the branch tip.

✓ **B** soft reset application
A soft reset moves the branch tip to an earlier commit without changing the index or working tree, so existing modifications remain available and unaltered. This is the choice that avoids discarding prior work while backing out the targeted change. [4]
https://netascode.cisco.com/docs/labs/sdwan/reference/understanding-git/

✗ **C** commit history rewind

Rewinding commit history changes the branch to an earlier point and can remove later commits from the branch's visible history. The scenario emphasizes preserving previous changes rather than rewriting history in that way.

✗ **D** hard reset command
A hard reset changes the branch tip and resets the index and working tree to match it, potentially discarding modifications. That conflicts with the requirement to keep previous changes intact.

https://netascode.cisco.com/docs/labs/sdwan/reference/understanding-git/

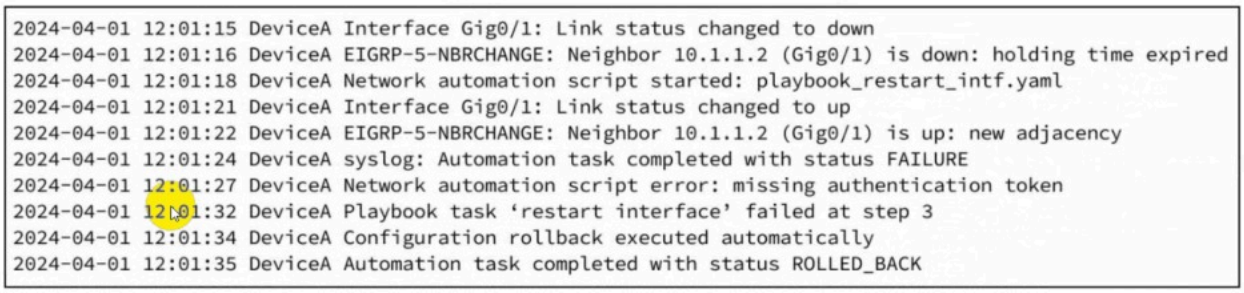
Question 59

Single choice · Domain: D3 Operations › 3.3 Diagnose problems with network automation given logs and output related to an event

Refer to the exhibit.

```
2024-04-01 12:01:15 DeviceA Interface Gig0/1: Link status changed to down
2024-04-01 12:01:16 DeviceA EIGRP-5-NBRCHANGE: Neighbor 10.1.1.2 (Gig0/1) is down: holding
2024-04-01 12:01:18 DeviceA Network automation script started: playbook_restart_intf.yaml
2024-04-01 12:01:21 DeviceA Interface Gig0/1: Link status changed to up
2024-04-01 12:01:22 DeviceA EIGRP-5-NBRCHANGE: Neighbor 10.1.1.2 (Gig0/1) is up: new adjace
2024-04-01 12:01:24 DeviceA syslog: Automation task completed with status FAILURE
2024-04-01 12:01:27 DeviceA Network automation script error: missing authentication token
2024-04-01 12:01:32 DeviceA Playbook task 'restart interface' failed at step 3
2024-04-01 12:01:34 DeviceA Configuration rollback executed automatically
2024-04-01 12:01:35 DeviceA Automation task completed with status ROLLED_BACK
```

Refer to the exhibit During a scheduled network automation task for switch provisioning, an event involving interface reset was executed in the device as planned The automation platform initiated the process at the expected time and the log indicates that a series of actions were attempted Shortly after the event, the system identified an error and triggered an automatic configuration rollback. The network administrator observed that the automation job status reported as FAILURE and network connectivity was impacted during the operation What is the primary cause of the issue?



A. unreachable API endpoint
B. connection timeout threshold
C. misconfiguration of the authentication mechanism
D. syntax issue in verification process

Answer: C. misconfiguration of the authentication mechanism

Question 60

Single choice · Domain: D3 Operations › 3.2 Implement a logging strategy for a network automation solution targeting destinations such as syslog or webhooks

A network management stack is being refactored for enhanced visibility and automation To achieve effective reporting for microservice health monitoring, system log outputs must be directed to a RESTful webhook receiver Operational requirements specify timely and reliable log notification without modifying core network devices Which logging approach meets the requirements?

A. SMB share mount for file-based log access
B. polling mechanism using SNMP table queries
C. email-based notification script for periodic alerts
D. push-based notification handler for operational telemetry

Answer: D. push-based notification handler for operational telemetry

Explanation

A push-based operational telemetry handler best matches the need to send timely notifications to a RESTful webhook receiver. Unlike periodic polling or file- and email-based approaches, telemetry can stream updates to a consumer as they become available, supporting responsive microservice health monitoring. References 1 and 2 describe network telemetry and its streaming model.

https://www.cisco.com/c/en/us/support/docs/switches/nexus-9000-series-switches/226393-configure-and-verify-telemetry-on-nexus.html
https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/prog/mdt/model-driven-telemetry.pdf

Free demo · full exam: 549 questions

✗ **A** SMB share mount for file-based log access
An SMB share makes log files available for a receiver to access, but it does not push timely notifications to a RESTful webhook. It would require a separate process to check or retrieve files.

✗ **B** polling mechanism using SNMP table queries
SNMP polling retrieves values when the monitoring system queries the device, so updates can be delayed until the next poll. This does not meet the requirement for timely push notifications to the webhook; telemetry is an alternative to polling-based monitoring. [10]

https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-xr-software/216947-snmp-migration-to-telemetry-on-ios-xr.html

✗ **C** email-based notification script for periodic alerts
Periodic email alerts are not direct webhook notifications and may report changes only after the script's next scheduled run. That makes them a poorer fit for timely, automated microservice health reporting.

✓ **D** push-based notification handler for operational telemetry
A push-based telemetry handler sends updates to a consumer without waiting for repeated polling requests, which fits the requirement for timely notifications. It is the best match among the options for delivering operational data to the webhook receiver. [1][2]

https://www.cisco.com/c/en/us/support/docs/switches/nexus-9000-series-switches/226393-configure-and-verify-telemetry-on-nexus.html
https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/prog/mdt/model-driven-telemetry.pdf

Dumpsgate

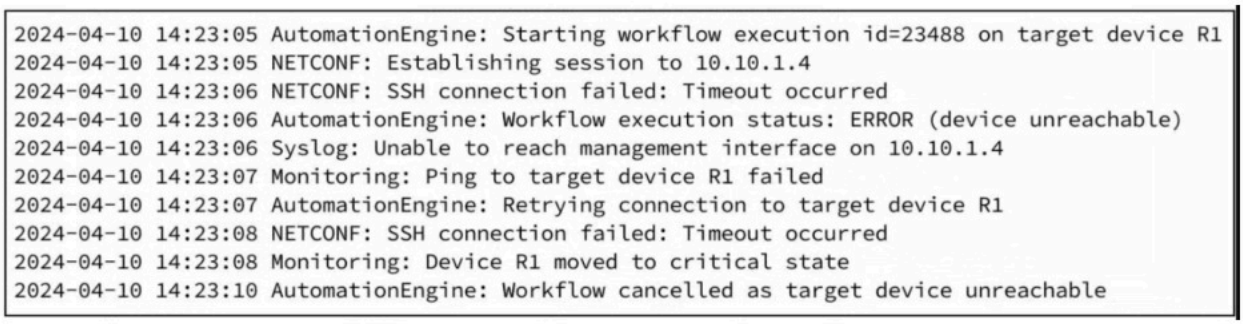
Question 61

Single choice · Domain: D3 Operations › 3.3 Diagnose problems with network automation given logs and output related to an event

Refer to the exhibit.

2024-04-10 14:23:05 AutomationEngine: Starting workflow execution id=23488 on target device R1
2024-04-10 14:23:05 NETCONF: Establishing session to 10.10.1.4
2024-04-10 14:23:06 NETCONF: SSH connection failed: Timeout occurred
2024-04-10 14:23:06 AutomationEngine: Workflow execution status: ERROR (device unreachable)
2024-04-10 14:23:06 Syslog: Unable to reach management interface on 10.10.1.4
2024-04-10 14:23:07 Monitoring: Ping to target device R1 failed
2024-04-10 14:23:07 AutomationEngine: Retrying connection to target device R1
2024-04-10 14:23:08 NETCONF: SSH connection failed: Timeout occurred
2024-04-10 14:23:08 Monitoring: Device R1 moved to critical state
2024-04-10 14:23:10 AutomationEngine: Workflow cancelled as target device unreachable

Refer to the exhibit. A network automation job was scheduled to run during the standard maintenance window on device R1. The job was configured to use connection method NETCONF, and execution was triggered from the orchestrator server. During the maintenance window, the system logs indicate that the workflow could not reach its intended target. Further investigation into monitoring alerts reveals persistent connection failures for the duration of the job window. Operations reported that other devices were reachable from the server. The orchestration logs do not show issues with job scheduling or agent workflows. What is the root network automation issue?



A. Hostname mismatch is detected on controller
B. The management firewall is missing an access rule.
C. Incorrect credentials prevent session establishment
D. The NTP server is unreachable from the device.

Answer: B. The management firewall is missing an access rule.

Explanation

The logs show timeouts while the orchestrator attempts to establish a NETCONF-over-SSH session, followed by failed pings and inability to reach R1's management interface. Because other devices remain reachable and orchestration scheduling is functioning, the failure is specific to network access to R1; a missing management-firewall access rule is the best fit. NETCONF uses SSH for its transport, so blocking

that management path can prevent the session from being established. [1]

https://www.cisco.com/c/en/us/td/docs/routers/ios-xe/network-services/network-services/m_netconf-sshv2.html

✗ **A** Hostname mismatch is detected on controller
The connection attempt is shown going to the device's IP address, 10.10.1.4, and fails with a timeout. The exhibit does not report a hostname or identity mismatch; it reports that R1's management interface is unreachable.

✓ **B** The management firewall is missing an access rule.
A firewall rule blocking management traffic can cause NETCONF-over-SSH attempts to time out and prevent the orchestrator from reaching R1's management interface. Since other devices are reachable and the orchestration workflow itself is functioning, a network access restriction specific to R1 is the best explanation. [1]

https://www.cisco.com/c/en/us/td/docs/routers/ios-xe/network-services/network-services/m_netconf-sshv2.html

✗ **C** Incorrect credentials prevent session establishment
Incorrect credentials would generally cause an authentication failure after the SSH connection reaches the device, not a connection timeout accompanied by failed pings and an unreachable management interface. The log points to a reachability problem rather than rejected credentials.

https://www.cisco.com/c/en/us/td/docs/routers/ios-xe/network-services/network-services/m_netconf-sshv2.html

✗ **D** The NTP server is unreachable from the device.
An NTP reachability issue does not explain why the orchestrator cannot ping R1 or establish a NETCONF-over-SSH connection to its management interface. The exhibit contains no indication that time synchronization is involved.

https://www.cisco.com/c/en/us/support/docs/ip/network-time-protocol-ntp/116161-trouble-ntp-00.html

Dumpsgate

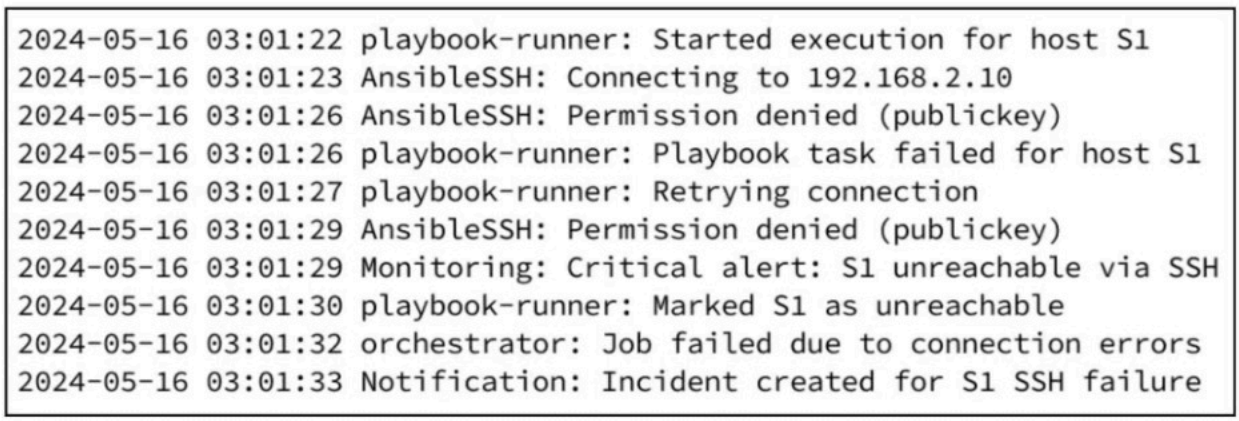
Question 62

Single choice · Domain: D3 Operations › 3.3 Diagnose problems with network automation given logs and output related to an event

Refer to the exhibit.

```
2024-05-16 03:01:22 playbook-runner: Started execution for host S1
2024-05-16 03:01:23 AnsibleSSH: Connecting to 192.168.2.10
2024-05-16 03:01:26 AnsibleSSH: Permission denied (publickey)
2024-05-16 03:01:26 playbook-runner: Playbook task failed for host S1
2024-05-16 03:01:27 playbook-runner: Retrying connection
2024-05-16 03:01:29 AnsibleSSH: Permission denied (publickey)
2024-05-16 03:01:29 Monitoring: Critical alert: S1 unreachable via SSH
2024-05-16 03:01:30 playbook-runner: Marked S1 as unreachable
2024-05-16 03:01:32 orchestrator: Job failed due to connection errors
2024-05-16 03:01:33 Notification: Incident created for S1 SSH failure
```

Refer to the exhibit. The automation workflow was set to deploy configuration to target node S1 using method Ansible SSH. The orchestrator attempted to execute the job, but the targeted system did not allow access. Multiple retries were initiated but were unsuccessful. The monitoring solution issued critical alerts throughout the operation window for persistent connection failures. The job logs reported successive permission denials. Other systems handled by the orchestrator during the same time performed as expected. What is the underlying automation error?



A. NTP is not synchronized with the server.
B. The public key is missing on the target.
C. OSPF is not enabled on the router.
D. The loopback interface is disabled on the device

Answer: B. The public key is missing on the target.

Explanation

The log explicitly reports “Permission denied (publickey)” for SSH connections to S1, including after a retry. This indicates public-key authentication is failing for the target, consistent with the public key being missing or not installed on S1. Since other systems worked and the failure is specifically an SSH authentication denial, the issue is not a general orchestration or routing problem.

https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-shell-ssh/4145-ssh.html
https://www.cisco.com/c/en/us/td/docs/routers/ios-xe/security-vpn/security-vpn/m_sec-usr-ssh-config-auth-xe.html

✗ **A** NTP is not synchronized with the server.
The failure occurs during SSH public-key authentication, as shown by the explicit "Permission denied (publickey)" messages. A time synchronization issue is not indicated by the logs and would not explain this authentication denial.

✓ **B** The public key is missing on the target.
The target repeatedly rejects the SSH connection with "Permission denied (publickey)," which points to a public-key authentication problem on S1. The public key needed to authenticate the automation connection is missing or otherwise unavailable to the target.

https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-shell-ssh/4145-ssh.html
https://www.cisco.com/c/en/us/td/docs/routers/ios-xe/security-vpn/security-vpn/m_sec-usr-ssh-config-auth-xe.html

✗ **C** OSPF is not enabled on the router.
The log shows that the orchestrator reached an SSH authentication step and was denied by public-key authentication. OSPF configuration is unrelated to this specific permission-denied error.

https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst9500/software/release/26-x/configuration_guide/rtng/b_26x_rtng_9500_cg/configuring_ospf.pdf

✗ **D** The loopback interface is disabled on the device
The connection attempt to S1 at 192.168.2.10 reaches an SSH authentication failure, rather than reporting an inability to reach the device due to an interface state. The evidence points to key-based authentication, not a disabled loopback interface.

Question 63

Single choice · Domain: D3 Operations › 3.4 Implement change validation for a network automation solution using pyATS CLI tools

A network automation engineer is working on a project that involves precheck validation. The application must perform change validation using pyATS quick run as part of the deployment pipeline. The process is intended to detect configuration drift and ensure consistency before making changes. The organization must minimize risks and maintain compliance by verifying the network state after modifications. Which implementation step helps to meet the requirements?

A. structured verification using automated comparison
B. inventory reporting via API endpoint integration
C. dynamic routing adjustments using custom scripts
D. log file monitoring with indirect event triggers

Answer: A. structured verification using automated comparison

Explanation

A pyATS quick run can collect and validate network state, and structured automated comparisons can reveal drift between the expected and actual state. Using those comparisons in the deployment pipeline supports pre-change validation and post-change verification, helping maintain consistency and reduce deployment risk. [1]

https://developer.cisco.com/codeexchange/github/repo/sttrayno/pyATS-Lab-Guide

✓ **A** structured verification using automated comparison

Automated comparison checks structured network state against the expected state, making configuration drift visible. Running this validation before and after modifications helps confirm consistency as part of the deployment process.

https://developer.cisco.com/codeexchange/github/repo/sttrayno/pyATS-Lab-Guide

✗ **B** inventory reporting via API endpoint integration
An inventory API can report what devices or resources exist, but reporting alone does not compare configuration state or verify that changes produced the expected result. It therefore does not directly meet the drift-validation requirement.

✗ **C** dynamic routing adjustments using custom scripts
This describes making routing changes rather than validating the network before and after deployment. It could introduce additional changes without providing the requested consistency check.

✗ **D** log file monitoring with indirect event triggers
Monitoring logs for events is indirect and does not establish whether the current network configuration matches the intended state. It is not a substitute for structured comparison-based validation.

Question 64

Single choice · Domain: D3 Operations › 3.6 Implement secure coding practices into a network automation solution to meet input validation, authentication, and secret management requirements

A network engineering project involves automating device onboarding, and the code must use a dedicated input schema validator for increased reliability. The engineers must ensure that input is strictly checked and that malicious file paths cannot be injected by external sources. Security reviewers requested that validation controls be implemented early in the automation logic. Which approach integrates robust input assessment into this workflow?

A. initial execution of compliant pattern checks
B. passing of values through generic converters
C. transformation of data into nullable lists
D. usage of unchecked input data

Answer: A. initial execution of compliant pattern checks

Explanation

The correct approach is to validate inputs immediately with compliant pattern checks, before those values influence device-onboarding operations. This catches values that do not match the expected input format and helps prevent externally supplied malicious paths from reaching file-handling logic. It aligns with the requirement to apply a dedicated, strict input-validation step early in the workflow.

https://blogs.cisco.com/cisco-devnet/pythonvulnerabilities01
https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-ftdfmc-dir-trav-wERqjhWq?vs_f=Cisco+Security+Advisory%26vs_cat%3DSecurity+Intelligence%26vs_type%3DRSS%26vs_p%3DCisco+Secure+Firewall+M
https://developer.cisco.com/codeexchange/github/repo/waitai/cwm-json-schema-validator/

✓ **A** initial execution of compliant pattern checks
Applying pattern checks at the start of the automation logic lets the workflow reject inputs that fail the expected format before processing them. This is the option that best addresses strict validation and reduces the opportunity for malicious file paths to be used.

https://blogs.cisco.com/cisco-devnet/pythonvulnerabilities01

https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-ftdfmc-dir-trav-wERqjhWq?vs_f=Cisco+Security+Advisory%26vs_cat%3DSecurity+Intelligence%26vs_type%3DRSS%26vs_p%3DCisco+Secure+Firewall+M
https://developer.cisco.com/codeexchange/github/repo/waitai/cwm-json-schema-validator/

✗ **B** passing of values through generic converters
Generic converters may change a value's representation, but that does not establish that it conforms to the required input schema or exclude malicious path content. The scenario calls for explicit validation before further processing.

https://blogs.cisco.com/cisco-devnet/pythonvulnerabilities01
https://developer.cisco.com/codeexchange/github/repo/waitai/cwm-json-schema-validator/

✗ **C** transformation of data into nullable lists
Changing data into nullable lists is a data-structure transformation, not a strict assessment of whether inputs meet an allowed schema or pattern. It therefore does not address the stated path-injection concern.

✗ **D** usage of unchecked input data
Using unchecked external input directly contradicts the request for strict validation early in the workflow. It can allow malicious path values to reach later processing.

https://blogs.cisco.com/cisco-devnet/pythonvulnerabilities01
https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-ftdfmc-dir-trav-wERqjhWq?vs_f=Cisco+Security+Advisory%26vs_cat%3DSecurity+Intelligence%26vs_type%3DRSS%26vs_p%3DCisco+Secure+Firewall+M

Question 65

Single choice · Domain: D3 Operations › 3.4 Implement change validation for a network automation solution using pyATS CLI tools

deployment project targets infrastructure that uses multivendor platforms. The team incorporated change validation into their pipeline and applied pyATS test discovery as a critical part of their process.

II modifications are subject to checks for accuracy, and unauthorized deviations must be detected automatically. Reducing human oversight while maintaining traceable records is mandated. Which task integration meets the requirements?

A. automated scenario mapping prior to staging
B. access list propagation via exported templates
C. traffic shaping policies from route reflectors
D. time zone assignment using individual timers

Answer: A. automated scenario mapping prior to staging

Explanation

Automated scenario mapping before staging is the best fit because it lets the pipeline identify and validate relevant test scenarios before changes are deployed. Combined with pyATS test discovery, this supports repeatable checks across the target infrastructure and helps detect deviations without relying on manual review. Running the checks in the pipeline also provides a traceable record of validation results. [1]

https://developer.cisco.com/docs/pyats/26-5/

✓ **A** automated scenario mapping prior to staging

Mapping scenarios before staging gives the pipeline a basis for automatically checking proposed changes for accuracy before they proceed. This fits the use of pyATS test discovery and the requirement to detect unauthorized deviations while reducing manual oversight and retaining validation results. [1]

https://developer.cisco.com/docs/pyats/26-5/

✗ **B** access list propagation via exported templates
Propagating access lists with templates concerns distributing configuration, not discovering and running change-validation scenarios. By itself, it does not meet the stated need to automatically detect deviations and record validation.

✗ **C** traffic shaping policies from route reflectors
This concerns traffic-policy configuration and does not describe a pipeline task for validating proposed modifications. It does not address automated accuracy checks or traceable validation records.

✗ **D** time zone assignment using individual timers
Assigning time zones with individual timers is unrelated to automated scenario discovery or change validation. It also does not address detection of unauthorized changes across the multivendor infrastructure.

Question 66

Single choice · Domain: D3 Operations › 3.5 Describe the process to obtain and deploy CA-signed TLS certificates

What is the task for generating a certificate signing request in the process to obtain and deploy CA-signed TLS certificates?

A. establishing unencrypted communication endpoints
B. preparing a file with encoded identification data and a public key
C. disabling automatic renewal of secure communications
D. adding signature chains during encryption processes

Answer: B. preparing a file with encoded identification data and a public key

Explanation

A certificate signing request (CSR) is prepared for submission to a certificate authority when obtaining a CA-signed TLS certificate. It contains encoded identification information and a public key, allowing the CA to issue a certificate associated with that identity and key. Therefore, the task is preparing that file, not configuring unencrypted endpoints or changing renewal or encryption behavior.

https://ci.manage.security.cisco.com/content/docs/t-adding-an-identity-certificate-object-for-certificate-signing-request--csr-_2.html
https://meeting-infohub.cisco.com/faq/content/8/251/en/what-information-is-required-to-create-a-certificate-signing-request-file-csr.html
https://www.cisco.com/c/en/us/support/docs/smb/routers/cisco-rv-series-small-business-routers/Certificate_Import_Export_Generate_CSR_on_the_RV160_and_RV260_Series_Router.html

✗ **A** establishing unencrypted communication endpoints
A CSR is part of requesting a certificate; it does not establish unencrypted endpoints. The certificate is used to support secure communications after it is issued and deployed.

https://ci.manage.security.cisco.com/content/docs/t-adding-an-identity-certificate-object-for-certificate-signing-request--csr-_2.html
https://www.cisco.com/c/en/us/support/docs/smb/routers/cisco-rv-series-small-business-routers/Certificate_Import_Export_Generate_CSR_on_the_RV160_and_RV260_Series_Router.html

✓ **B** preparing a file with encoded identification data and a public key
A CSR is a file containing the requester's identifying information and public key, which is sent to a certificate authority as part of obtaining a signed certificate. This is the task required in the certificate-request process.

https://ci.manage.security.cisco.com/content/docs/t-adding-an-identity-certificate-object-for-certificate-signing-request--csr-_2.html
https://meeting-infohub.cisco.com/faq/content/8/251/en/what-information-is-required-to-create-a-certificate-signing-request-file-csr.html
https://www.cisco.com/c/en/us/support/docs/smb/routers/cisco-rv-series-small-business-routers/Certificate_Import_Export_Generate_CSR_on_the_RV160_and_RV260_Series_Router.html

✗ **C** disabling automatic renewal of secure communications
Generating a CSR does not disable certificate renewal or configure automatic renewal. It prepares the certificate request for the CA.

https://ci.manage.security.cisco.com/content/docs/t-adding-an-identity-certificate-object-for-certificate-signing-request--csr-_2.html
https://www.cisco.com/c/en/us/support/docs/smb/routers/cisco-rv-series-small-business-routers/Certificate_Import_Export_Generate_CSR_on_the_RV160_and_RV260_Series_Router.html

✗ **D** adding signature chains during encryption processes
A CSR is created to request a certificate; it does not add signature chains during encryption. Certificate chains concern certificates used to establish trust, rather than the task of generating the request file.

https://ci.manage.security.cisco.com/content/docs/t-adding-an-identity-certificate-object-for-certificate-signing-request--csr-_2.html
https://www.cisco.com/c/en/us/support/docs/smb/routers/cisco-rv-series-small-business-routers/Certificate_Import_Export_Generate_CSR_on_the_RV160_and_RV260_Series_Router.html

Dumpsgate

Question 67

Single choice · Domain: D3 Operations › 3.3 Diagnose problems with network automation given logs and output related to an event

Refer to the exhibit.

```
2024-06-03 08:22:11 orchestrator: Starting deployment to PE1 using NETCONF
2024-06-03 08:22:12 session_manager: Opening NETCONF session to 10.33.1.31
2024-06-03 08:22:12 session_manager: NETCONF session established
2024-06-03 08:22:15 NETCONF: Sending validate RPC
2024-06-03 08:22:16 NETCONF: Unable to validate configuration - schema not found
2024-06-03 08:22:16 NETCONF: Closing session to 10.33.1.31
2024-06-03 08:22:17 orchestrator: NETCONF job failed on PE1
2024-06-03 08:22:18 notification: Validation method failed on target device
2024-06-03 08:22:18 monitoring: PE1 entered warning state (config validation failure)
2024-06-03 08:22:19 orchestrator: Job marked as failed
```

Refer to the exhibit. The configuration automation process for system PE1 was triggered through interface method NETCONF during a routine rollout window. According to orchestrator and session logs, the session successfully initiated and authenticated, but the deployment halted before actual configuration updates.

Further logs indicate failures related to validation processes, and monitoring displays an incomplete state for the action. No errors occurred in device reachability, and other automations to similar devices completed without issue. Administrative teams confirmed that no access restrictions or connectivity problems occurred at the time. Which network automation failure occurred?


```
2024-06-03 08:22:11 orchestrator: Starting deployment to PE1 using NETCONF
2024-06-03 08:22:12 session_manager: Opening NETCONF session to 10.33.1.31
2024-06-03 08:22:12 session_manager: NETCONF session established
2024-06-03 08:22:15 NETCONF: Sending validate RPC
2024-06-03 08:22:16 NETCONF: Unable to validate configuration - schema not found
2024-06-03 08:22:16 NETCONF: Closing session to 10.33.1.31
2024-06-03 08:22:17 orchestrator: NETCONF job failed on PE1
2024-06-03 08:22:18 notification: Validation method failed on target device
2024-06-03 08:22:18 monitoring: PE1 entered warning state (config validation failure)
2024-06-03 08:22:19 orchestrator: Job marked as failed
```


A. The static route is missing in the table.
B. The DNS resolver is unreachable.
C. The device data model is not installed.
D. The logging facility is misconfigured.

Answer: C. The device data model is not installed.

Explanation

The NETCONF session was established, so the failure was not caused by device reachability or access. The validate RPC then failed because the device reported that the schema was not found, indicating that the required device data model was unavailable or not installed. This stopped the deployment before configuration updates could be applied.

Free demo · full exam: 549 questions

https://www.cisco.com/c/en/us/support/docs/storage-networking/management/200933-YANG-NETCONF-Configuration-Validation.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/167/b_167_programmability_cg/configuring_yang_datamodel.pdf

✗ **A** The static route is missing in the table.
The logs show that a NETCONF session was established and validation failed because the schema was not found. They do not report a routing or reachability failure, and the scenario explicitly rules out connectivity problems.

✗ **B** The DNS resolver is unreachable.
The session successfully connected to the device at 10.33.1.31, and the failure occurred later during the validation RPC. Nothing in the logs identifies DNS resolution as a problem.

✓ **C** The device data model is not installed.
The NETCONF validate RPC returned “schema not found,” which indicates that the schema or device data model needed to validate the configuration is unavailable. That explains why the process stopped before applying configuration changes.

https://www.cisco.com/c/en/us/support/docs/storage-networking/management/200933-YANG-NETCONF-Configuration-Validation.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/167/b_167_programmability_cg/configuring_yang_datamodel.pdf

✗ **D** The logging facility is misconfigured.
The logs provide a clear validation error from NETCONF rather than evidence of a logging-system failure. A logging problem would not explain why the device reported that the schema was not found.

Question 68

Single choice · Domain: D3 Operations › 3.6 Implement secure coding practices into a network automation solution to meet input validation, authentication, and secret management requirements

Automation developers are integrating secure practices into a code repository and must use encrypted file storage for effective credential protection. Private configuration secrets must not be exposed in any build artifacts or logs. The automation must allow controlled access and deny unauthorized retrievals. Which solution enforces data confidentiality throughout the automation system?

A. generation of descriptive error outputs
B. embedding identifiers in test suites
C. creation of secret data mount points
D. exposure of sensitive details during process dump

Answer: C. creation of secret data mount points

Explanation

Creating secret data mount points lets automation consume private configuration secrets through a controlled mechanism rather than embedding them in build artifacts or printing them in logs. This supports confidentiality and helps restrict retrieval to authorized workloads. The listed guidance on managing secrets for Kubernetes workloads is relevant to this approach.

https://developer.cisco.com/articles/secrets-management-for-your-kubernetes-workloads/

✗ **A** generation of descriptive error outputs

Descriptive errors can help diagnose failures, but they do not provide encrypted storage or control access to credentials. If they include secret values, they could expose them in logs.

✗ **B** embedding identifiers in test suites
Identifiers in tests do not encrypt or securely store configuration secrets. This does not prevent secrets from appearing in artifacts or being retrieved by unauthorized users.

✓ **C** creation of secret data mount points
Secret data mount points provide a way for automation to access sensitive configuration without embedding it in build artifacts or exposing it in ordinary output. Access to the mounted secret can be controlled for authorized workloads.

https://developer.cisco.com/articles/secrets-management-for-your-kubernetes-workloads/

✗ **D** exposure of sensitive details during process dump
Exposing sensitive details in a process dump directly violates the requirement to keep secrets out of logs and other outputs. It increases the chance of unauthorized retrieval rather than protecting confidentiality.

Dumpsgate

Question 69

Single choice · Domain: D3 Operations › 3.3 Diagnose problems with network automation given logs and output related to an event

Refer to the exhibit.

```
2024-05-11 09:22:01 Controller-Automation: Startup initiated for wireless provisioning
2024-05-11 09:22:03 Controller-Automation: Importing playbook site_add_AP.yaml
2024-05-11 09:22:05 Controller-Device: SSH session opened
2024-05-11 09:22:05 Controller-Device: Sending configuration commands
2024-05-11 09:22:07 Controller-Device: Error: command execution unsuccessful
2024-05-11 09:22:08 Controller-Device: Error: privilege level too low
2024-05-11 09:22:11 Controller-Automation: Task execution halted
2024-05-11 09:22:14 Controller-Automation: Recording failure result for ticket #456345
2024-05-11 09:22:15 Controller-Automation: Notification sent to admin group
2024-05-11 09:22:16 Controller: Begin rollback of applied changes
```

Refer to the exhibit. After a network automation scheduled event for access control application, a subsequent port configuration was implemented as part of the workflow. System logs show a series of sequential tasks, with the automation process halting unexpectedly. The notification system recorded a failure and initiated a rollback procedure to discard any applied changes. Network stability after the event was affected, which required further review. What is the primary cause of the issue?


```
2024-05-11 09:22:01 Controller-Automation: Startup initiated for wireless provisioning
2024-05-11 09:22:03 Controller-Automation: Importing playbook site_add_AP.yaml
2024-05-11 09:22:05 Controller-Device: SSH session opened
2024-05-11 09:22:05 Controller-Device: Sending configuration commands
2024-05-11 09:22:07 Controller-Device: Error: command execution unsuccessful
2024-05-11 09:22:08 Controller-Device: Error: privilege level too low
2024-05-11 09:22:11 Controller-Automation: Task execution halted
2024-05-11 09:22:14 Controller-Automation: Recording failure result for ticket #456345
2024-05-11 09:22:15 Controller-Automation: Notification sent to admin group
2024-05-11 09:22:16 Controller: Begin rollback of applied changes
```


A. missing VRF instance
B. issue with command execution privilege
C. exceeded hardware constraint
D. BGP session reset

Answer: B. issue with command execution privilege

Explanation

The logs identify insufficient command privileges as the direct cause: configuration commands were sent, execution failed, and the device reported “privilege level too low.” The automation then halted, recorded a failure, notified administrators, and began rollback. This supports an execution-privilege issue rather than a VRF, hardware, or BGP problem.

https://www.cisco.com/c/en/us/td/docs/ios/fundamentals/command/reference/cf_book/cf_d1.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/fundamentals/command/cf_command_ref.pdf
https://www.cisco.com/c/en/us/support/docs/security-vpn/remote-authentication-dial-user-service-radius/13860-

PRIV.html
https://www.cisco.com/c/en/us/td/docs/ios/sec_user_services/configuration/guide/convert/aaa/sec_cfg_authorizatn.html
https://www.cisco.com/c/en/us/td/docs/routers/secure-routers/sec-8100-series/software-config-guide/cisco-8100-series-secure-routers-software-configuration-guide/sec_router_access_pwd_privilege.html

✗ **A** missing VRF instance
A missing VRF would relate to routing or interface context, but the log explicitly reports that command execution failed because the privilege level was too low. There is no indication of a VRF lookup or configuration error.

https://www.cisco.com/c/en/us/td/docs/routers/ios-xe/ip-routing/b-ip-routing/m_mp-multi-vrf-vrf-lite.html

✓ **B** issue with command execution privilege
The device reports “privilege level too low” immediately after the configuration commands fail. That error explains why the automation halted and subsequently recorded a failure and began rollback.

https://www.cisco.com/c/en/us/td/docs/ios/fundamentals/command/reference/cf_book/cf_d1.html
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/fundamentals/command/cf_command_ref.pdf
https://www.cisco.com/c/en/us/support/docs/security-vpn/remote-authentication-dial-user-service-radius/13860-PRIV.html
https://www.cisco.com/c/en/us/td/docs/ios/sec_user_services/configuration/guide/convert/aaa/sec_cfg_authorizatn.html
https://www.cisco.com/c/en/us/td/docs/routers/secure-routers/sec-8100-series/software-config-guide/cisco-8100-series-secure-routers-software-configuration-guide/sec_router_access_pwd_privilege.html

✗ **C** exceeded hardware constraint
The logs do not report resource exhaustion or a hardware limit. They identify insufficient command privilege as the reason the configuration commands could not be executed.

✗ **D** BGP session reset
No BGP session or reset is mentioned in the logs. The recorded failure occurs while executing configuration commands and is attributed to a privilege level that is too low.

Question 70

Single choice · Domain: D3 Operations › 3.5 Describe the process to obtain and deploy CA-signed TLS certificates

What is the task for synchronizing network device clocks in the sequence for deploying CA-signed TLS certificates?

A. modifying routing table configurations
B. compressing voice traffic for efficiency
C. aligning time data for accurate validation
D. disabling audit trails for system logs

Answer: C. aligning time data for accurate validation

Explanation

The sequence requires synchronizing network device clocks so certificate validity dates can be checked against accurate time. If a device clock is significantly incorrect, a CA-signed TLS certificate may appear not yet valid or expired even when it is otherwise valid. Therefore, aligning time data is the relevant task. [4]
https://www.cisco.com/c/en/us/support/docs/security-vpn/public-key-infrastructure-pki/220422-configure-ca-signed-

Free demo · full exam: 549 questions

certificates-with-io.pdf

✗ **A** modifying routing table configurations
Routing tables determine how network traffic is forwarded; changing them does not synchronize device clocks. It does not address the accurate time needed to validate a TLS certificate.

✗ **B** compressing voice traffic for efficiency
Voice-traffic compression is unrelated to clock synchronization or certificate validity checks. It would not ensure that devices can accurately validate CA-signed TLS certificates.

✓ **C** aligning time data for accurate validation
Synchronizing clocks gives devices accurate time for checking certificate validity periods. This is why clock alignment belongs in the deployment sequence for CA-signed TLS certificates. [4]

https://www.cisco.com/c/en/us/support/docs/security-vpn/public-key-infrastructure-pki/220422-configure-ca-signed-certificates-with-io.pdf

✗ **D** disabling audit trails for system logs
Disabling audit trails changes logging behavior, not device time. It neither synchronizes clocks nor helps validate a TLS certificate.

Question 71

Single choice · Domain: D3 Operations › 3.6 Implement secure coding practices into a network automation solution to meet input validation, authentication, and secret management requirements

A company uses an API to share sensitive customer data with third-party partners. To ensure the security of this data, how should end-to-end encryption be applied?

A. Mask or obfuscate sensitive data, such as passwords and API keys, to protect it from unauthorized access.

B. Impose API rate limits to prevent abuse and protect against denial-of-service attacks.

C. Encrypt API data from the client to the server and back, which ensures confidentiality at all stages of transmission.

D. Employ robust authentication to verify the identity of users and grants access based on predefined roles and permissions.

Answer: C. Encrypt API data from the client to the server and back, which ensures confidentiality at all stages of transmission.

Explanation

The requirement is to keep sensitive customer data confidential while it is exchanged through the API. Encrypting the data between the client and server, and on the return path, protects it during transmission; masking, rate limiting, and authentication address different security concerns rather than providing end-to-end encryption. [3][7][11]

https://outshift.cisco.com/blog/api-security
https://blogs.cisco.com/cisco-devnet/securing-apis-from-left-to-right-and-everywhere-in-between
https://radkit.cisco.com/docs/security/security_e2ee.html

✗ **A** Mask or obfuscate sensitive data, such as passwords and API keys, to protect it from unauthorized access.
Masking or obfuscation can hide sensitive values in displayed or logged data, but it does not encrypt the customer data as it travels between the client and API server. It therefore does not meet the stated

Free demo · full exam: 549 questions

requirement for end-to-end encryption.

https://outshift.cisco.com/blog/api-security
https://blogs.cisco.com/cisco-devnet/securing-apis-from-left-to-right-and-everywhere-in-between

✗ **B** Impose API rate limits to prevent abuse and protect against denial-of-service attacks.
Rate limits can restrict excessive requests and help mitigate abuse or denial-of-service attacks. They do not encrypt the data exchanged with third-party partners or protect its confidentiality in transit.

https://outshift.cisco.com/blog/api-security
https://blogs.cisco.com/cisco-devnet/securing-apis-from-left-to-right-and-everywhere-in-between

✓ **C** Encrypt API data from the client to the server and back, which ensures confidentiality at all stages of transmission.
Encrypting data in both directions protects the customer information while it is transmitted between the client and the API server. This directly addresses the requirement to preserve confidentiality during API exchanges.

https://outshift.cisco.com/blog/api-security
https://blogs.cisco.com/cisco-devnet/securing-apis-from-left-to-right-and-everywhere-in-between
https://radkit.cisco.com/docs/security/security_e2ee.html

✗ **D** Employ robust authentication to verify the identity of users and grants access based on predefined roles and permissions.
Authentication and role-based authorization help ensure that only permitted identities can access API resources. They do not encrypt the data exchanged with partners, so they cannot substitute for end-to-end encryption.

https://developer.cisco.com/docs/crosswork/network-controller/rbacsecuredapi/
https://developer.cisco.com/docs/security-cloud-control/authentication/
https://outshift.cisco.com/blog/api-security

Question 72

Multiple choice · Domain: D3 Operations › 3.6 Implement secure coding practices into a network automation solution to meet input validation, authentication, and secret management requirements

A developer is asked to implement an infrastructure pipeline within the organization. To implement the infrastructure pipeline, the developer must identify the methods that are required for storing and working with credentials, such as API key, usernames, and passwords. Which two methods must be used? (Choose two.)

A. Inject the secrets into the pipeline at runtime.
B. Store the secrets in a text file on your local machine.
C. Implement a secret store, such as Vault.
D. Store the secrets in a vars file, and encrypt and store in a version control system.
E. Use SHA-1 hashing and store in a version control system.

Answer

- A. **Inject the secrets into the pipeline at runtime.**
- C. **Implement a secret store, such as Vault.**

Explanation

The pipeline must handle credentials without embedding them in source files or exposing them in

version control. Injecting secrets only when the pipeline runs limits their exposure, while a dedicated secret store such as Vault provides a managed place to store and retrieve them. Together, these methods address both secret storage and use in the pipeline.

https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.html
https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.pdf

✓ **A** Inject the secrets into the pipeline at runtime.
Runtime injection lets the pipeline use credentials without hardcoding them into pipeline definitions or other files. This is one of the required methods for working with secrets in the infrastructure pipeline.

https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.html
https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.pdf

✗ **B** Store the secrets in a text file on your local machine.
A local text file is not a managed, secure way to store credentials for an infrastructure pipeline. It also does not provide a reliable mechanism for the pipeline to retrieve secrets safely.

https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.html
https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.pdf

✓ **C** Implement a secret store, such as Vault.
A secret store provides a centralized mechanism for storing and retrieving sensitive values such as API keys and passwords. Vault is an example of such a store and is one of the required methods.

https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.html
https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.pdf

✗ **D** Store the secrets in a vars file, and encrypt and store in a version control system.
Even if the vars file is encrypted, this approach places the credential material in version control and relies on separate key management and decryption practices. The required methods are runtime injection and a dedicated secret store, rather than storing the secrets in a repository.

https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.html
https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.pdf

✗ **E** Use SHA-1 hashing and store in a version control system.
Hashing is not a way to securely store credentials that the pipeline needs to use, because the original values generally cannot be recovered from a hash. SHA-1 is also not an appropriate substitute for a secret store or runtime secret injection.

https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.html
https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/technology-perspectives/automate-network-infrastructure-code-wp.pdf

Free demo · full exam: 549 questions

Question 73

Single choice · Domain: D3 Operations › 3.6 Implement secure coding practices into a network automation solution to meet input validation, authentication, and secret management requirements

A network automation team is developing an orchestration solution and must integrate a centralized secrets vault within the codebase. The engineers must prevent unauthorized access to the automated device configurations. The implementation must ensure that confidential connection strings are securely handled throughout the automation workflow. Which approach allows secure management and integration of sensitive data into network automation?

A. execution of plain text logs
B. use of managed identity authentication
C. inclusion of static code values
D. use of local variable caching

Answer: B. use of managed identity authentication

Explanation

Managed identity authentication lets the automation workflow authenticate to a secrets vault without embedding credentials or connection strings in the code. The vault can then control access to confidential values used by the workflow, helping prevent unauthorized access to device configurations. This directly meets the requirement to integrate centralized secrets management securely. [2, 5]

https://spg.xgslb-v3.cisco.com/c/en/us/td/docs/security/cdo/multicloud-defense/user-guide/cisco-multicloud-defense-userguide/g_certificate-and-keys-tech-notes/c_azure-key-vault.html
https://staging.manage.security.cisco.com/content/docs/multicloud/c_optional-user-assigned-managed-identity-for-key-vault-and-blob-storage-access.html

✗ **A** execution of plain text logs
Plain text logs can expose connection strings or other confidential values if they are recorded during automation. They do not provide secure authentication to or management of a centralized secrets vault.

✓ **B** use of managed identity authentication
A managed identity allows the automation workload to authenticate to a secrets vault without storing credentials in the codebase. The workflow can retrieve protected values under the vault's access controls, supporting secure handling of connection strings. [2, 5]

https://spg.xgslb-v3.cisco.com/c/en/us/td/docs/security/cdo/multicloud-defense/user-guide/cisco-multicloud-defense-userguide/g_certificate-and-keys-tech-notes/c_azure-key-vault.html
https://staging.manage.security.cisco.com/content/docs/multicloud/c_optional-user-assigned-managed-identity-for-key-vault-and-blob-storage-access.html

✗ **C** inclusion of static code values
Hard-coding connection strings or secrets makes them accessible to anyone who can inspect the code and complicates rotation. It contradicts the requirement to integrate a centralized vault and protect confidential values.

✗ **D** use of local variable caching
Caching sensitive values locally can leave them exposed beyond the point of use and does not itself provide secure access control or centralized secrets management. It is not a substitute for authenticating to the vault with a managed identity.

Free demo · full exam: 549 questions

Question 74

Single choice · Domain: D3 Operations › 3.4 Implement change validation for a network automation solution using pyATS CLI tools

The operations team is assigned a mission-critical environment that features high-availability architectures.

The workflow requires change validation steps using gyATS golden snapshot to ensure that all systems remain stable after upgrades. Procedures are established to detect discrepancies quickly and validate deployment outcomes systematically. Operational efficiency is a top concern, with minimal allowance for manual processes. Which step in the workflow addresses the requirements?

A. command scheduling with system crons
B. template upload through file synchronization services
C. baseline storage prior to changes
D. log extraction from external monitoring nodes

Answer: C. baseline storage prior to changes

Explanation

Storing a baseline snapshot before making changes gives the team a reference state to compare against after an upgrade. That comparison supports systematic validation of deployment outcomes and helps reveal discrepancies quickly, which is especially important in a mission-critical, high-availability environment. Establishing this baseline also addresses the workflow requirement without relying on additional manual checking steps. [1]

https://www.cisco.com/c/en/us/td/docs/routers/sdwan/configuration/government/sdwan-for-gov/m-snapshots.pdf

✗ **A** command scheduling with system crons
Scheduling commands can automate when tasks run, but it does not itself provide the golden snapshot baseline needed to validate system state after an upgrade. It therefore does not directly meet the stated requirement to compare deployment outcomes against a known-good state.

✗ **B** template upload through file synchronization services
Uploading templates can distribute configuration files, but it does not establish a pre-change snapshot for post-upgrade validation. The workflow specifically needs a reference state against which changes can be checked.

✓ **C** baseline storage prior to changes
Saving a baseline before changes preserves the known-good state for comparison after an upgrade. This lets the team identify discrepancies and validate whether deployment outcomes remain stable, aligning with the golden snapshot workflow. [1]

https://www.cisco.com/c/en/us/td/docs/routers/sdwan/configuration/government/sdwan-for-gov/m-snapshots.pdf

✗ **D** log extraction from external monitoring nodes
External logs may provide diagnostic evidence, but extracting them is not the step that creates a golden snapshot baseline for before-and-after validation. It does not by itself provide the reference state needed to check whether systems changed unexpectedly.

Question 75

Single choice · Domain: D3 Operations › 3.4 Implement change validation for a network automation solution using pyATS CLI tools

During operational readiness checks, the management team requires device onboarding to be implemented for smooth workflow progression. Automated change validation with pyATS information gatherer has been introduced to help maintain accurate infrastructure records. Ongoing change detection and streamlined troubleshooting are vital to keeping environments reliable. All checks must be performed as part of the regular pipeline. How can the process be implemented practically?

A. resource allocation through distributed clusters
B. connection status collection during initial assessment
C. section labeling via custom HTML scripts
D. parameter retrieval with RESTful endpoints

Answer: B. connection status collection during initial assessment

Explanation

Connection-status collection during initial assessment gives the pipeline a practical onboarding check: it determines whether devices can be reached before automated validation proceeds. This supports reliable information gathering and helps surface connectivity issues that would otherwise interfere with change detection or troubleshooting. pyATS is used for network-device testing and automation, making this kind of device-level check consistent with the scenario. [1][2]

https://developer.cisco.com/docs/pyats/open-source-documentation/
https://developer.cisco.com/pyats/

✗ **A** resource allocation through distributed clusters
Allocating resources across clusters is not a device-onboarding check and does not establish whether network devices are reachable. It therefore does not directly support the pipeline's initial information gathering or change validation.

✓ **B** connection status collection during initial assessment
Collecting connection status during the initial assessment verifies which devices the pipeline can reach before it gathers information or performs automated checks. This makes onboarding practical and helps identify connectivity problems that could undermine ongoing validation and troubleshooting. [1][2]

https://developer.cisco.com/docs/pyats/open-source-documentation/
https://developer.cisco.com/pyats/

✗ **C** section labeling via custom HTML scripts
Custom HTML section labels affect presentation, not device onboarding or operational checks. They do not provide the device information or reachability status needed for automated validation.

✗ **D** parameter retrieval with RESTful endpoints
Retrieving parameters through REST endpoints may be useful in some integrations, but the scenario asks for a practical onboarding check within a pyATS-based validation pipeline. The listed choice that directly checks device readiness is collecting connection status during the initial assessment.

Free demo · full exam: 549 questions

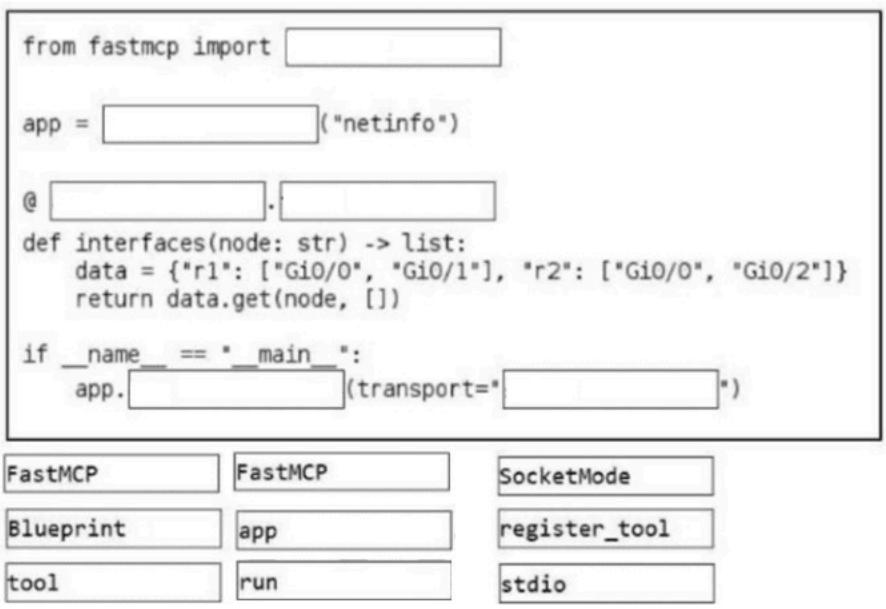
Question 76

Drag and drop · Domain: D4 AI in Automation › 4.3 Construct an MCP server to provide network information to an AI-agent using Python FastMCP

An engineer must expose an interface summary tool callable through FastMCP

Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used. Options may be used more than once.

```
from fastmcp import [Blank 1]

app = [Blank 2]("netinfo")

@[Blank 3].[Blank 4]
def interfaces(node: str) -> list:
    data = {"r1": ["Gi0/0", "Gi0/1"], "r2": ["Gi0/0", "Gi0/2"]}
    return data.get(node, [])

if __name__ == "__main__":
    app.[Blank 5](transport="[Blank 6]")
```



Items:

- A. FastMCP
- B. FastMCP
- C. SocketMode
- D. Blueprint
- E. app
- F. register_tool
- G. tool
- H. run
- I. stdio

Place them on:

1. Blank 1 (from fastmcp import [Blank 1])
2. Blank 2 (app = [Blank 2]("netinfo"))
3. Blank 3 (@[Blank 3].[Blank 4])
4. Blank 4 (@[Blank 3].[Blank 4])
5. Blank 5 (app.[Blank 5](transport="[Blank 6]"))
6. Blank 6 (app.run(transport="[Blank 6]"))

Answer

- Blank 1 (from fastmcp import [Blank 1]) → **A. FastMCP**
- Blank 2 (app = [Blank 2]("netinfo")) → **A. FastMCP**
- Blank 3 (@[Blank 3].[Blank 4]) → **E. app**
- Blank 4 (@[Blank 3].[Blank 4]) → **G. tool**
- Blank 5 (app.[Blank 5](transport="[Blank 6]")) → **H. run**
- Blank 6 (app.run(transport="[Blank 6]")) → **I. stdio**

Explanation

Importing and instantiating `FastMCP` creates the named `netinfo` server. The `@app.tool` decorator exposes `interfaces` as a callable tool, and `app.run(transport="stdio")` starts the server using standard input/output transport. This fills each blank with the supplied snippets needed for the requested FastMCP tool.

✓ **FastMCP (first option)** FastMCP
This is the class imported from `fastmcp` and used to create the server. It belongs in Blank 1.

✓ **FastMCP (second option)** FastMCP
Calling `FastMCP("netinfo")` instantiates the server assigned to `app`. This belongs in Blank 2.

✗ **SocketMode** SocketMode
The code needs the string value `stdio` for its transport argument, not a `SocketMode` identifier. This option does not fill either transport blank as specified.

✗ **Blueprint** Blueprint
The decorator target is the FastMCP server object `app`, not a `Blueprint`. This option does not provide the required tool registration decorator.

✓ **app** app
The server instance is named `app`, so `@app.tool` applies its tool decorator to the `interfaces` function. This belongs in Blank 3.

✗ **register_tool** register_tool
The required decorator in the supplied code is `@app.tool`; the given answer does not use `register_tool`. It therefore does not fill Blank 4 in this artifact.

Free demo · full exam: 549 questions

✓ **tool** tool
The FastMCP server's `tool` decorator registers `interfaces` as a callable tool. It completes `@app.tool` in Blank 4.

✓ **run** run
The server instance is started by calling its `run` method. It belongs in Blank 5 to form `app.run(...)`.

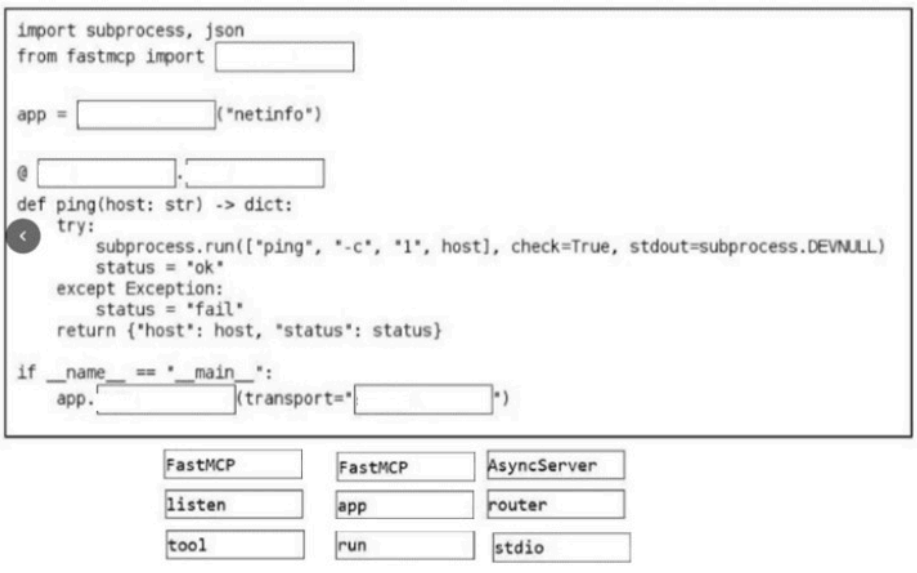
✓ **stdio** stdio
This is the transport value used to run the server over standard input and output. It belongs as the string value in Blank 6.

Question 77

Drag and drop · Domain: D4 AI in Automation › 4.3 Construct an MCP server to provide network information to an AI-agent using Python FastMCP

An engineer must add a reachability probe tool to a FastMCP server. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used. Options may be used more than once.

```
import subprocess, json
from fastmcp import [Blank 1]

app = [Blank 2]("netinfo")

@ [Blank 3].[Blank 4]
def ping(host: str) -> dict:
    try:
        subprocess.run(["ping", "-c", "1", host], check=True, stdout=subprocess.DEVNULL)
        status = "ok"
    except Exception:
        status = "fail"
    return {"host": host, "status": status}

if __name__ == "__main__":
    app.[Blank 5](transport="[Blank 6]")
```



Answer

- Blank 1 (from fastmcp import [Blank 1]) → **A. FastMCP**
- Blank 2 (app = [Blank 2]("netinfo")) → **A. FastMCP**
- Blank 3 (@ [Blank 3].[Blank 4]) → **E. app**
- Blank 4 (@ [Blank 3].[Blank 4]) → **G. tool**
- Blank 5 (app.[Blank 5](transport="[Blank 6]")) → **H. run**
- Blank 6 (app.[Blank 5](transport="[Blank 6]")) → **I. stdio**

Explanation

Importing `FastMCP` lets the code create the named server with `FastMCP("netinfo")`. The `@app.tool` decorator registers `ping` as a tool, and `app.run(transport="stdio")` starts the server using the requested transport. The remaining snippets do not fit these positions in this artifact.

✓ **Box 1: FastMCP (first occurrence)** FastMCP
This fills the import so the code can use `FastMCP` to construct the server. It is one of the two required FastMCP snippets.

✓ **Box 2: FastMCP (second occurrence)** FastMCP
This fills the constructor call `FastMCP("netinfo")`, creating the server instance assigned to `app`.

✗ **Unused option: AsyncServer** AsyncServer
The artifact uses the `FastMCP` class to create its server. `AsyncServer` is not the class shown in the required constructor call.

✗ **Unused option: listen** listen
The server is started with `app.run(...)` in the provided code structure. `listen` does not complete the required method call here.

✓ **Box 3: app** app
This forms the decorator prefix `@app.tool`, using the server instance created earlier to register the function.

✗ **Unused option: router** router
The code defines and uses the server instance as `app`; no router object is created or used in this decorator.

✓ **Box 4: tool** tool
With `app`, this forms `@app.tool`, registering `ping` as a tool that the FastMCP server can expose.

✓ **Box 5: run** run
This completes `app.run(...)`, which starts the server after the function has been defined.

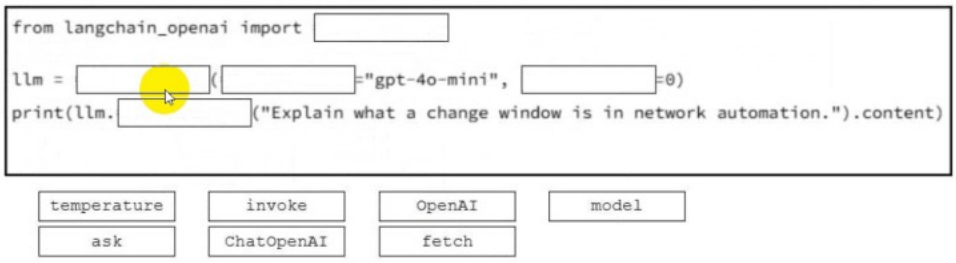
✓ **Box 6: stdio** stdio
This sets the transport argument to `stdio`, producing `app.run(transport="stdio")` as required by the completed code.

Question 78

Drag and drop · Domain: D4 AI in Automation › 4.4 Construct a conversational agent that leverages LLMs for network automation

An engineer must prototype a conversational agent using OpenAI via LangChain. The engineer must create the client for a specified model and invoke it with a short instruction. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used. Options may be used more than once.

```
from langchain_openai import [Blank 1]
llm = [Blank 2]([Blank 3]="gpt-4o-mini", [Blank 4]=0)
print(llm.[Blank 5]("Explain what a change window is in network automation.").content)
```



Items:

- A. temperature
- B. invoke
- C. OpenAI
- D. model
- E. ask
- F. ChatOpenAI
- G. fetch

Place them on:

1. Blank 1 (from langchain_openai import [Blank 1])
2. Blank 2 (llm = [Blank 2]([Blank 3]="gpt-4o-mini", [Blank 4]=0))
3. Blank 3 (llm = ChatOpenAI([Blank 3]="gpt-4o-mini", [Blank 4]=0))
4. Blank 4 (llm = ChatOpenAI(model="gpt-4o-mini", [Blank 4]=0))
5. Blank 5 (print(llm.[Blank 5]("Explain what a change window is in network automation.").content))

Answer

- Blank 1 (from langchain_openai import [Blank 1]) → **F. ChatOpenAI**
- Blank 2 (llm = [Blank 2]([Blank 3]="gpt-4o-mini", [Blank 4]=0)) → **F. ChatOpenAI**
- Blank 3 (llm = ChatOpenAI([Blank 3]="gpt-4o-mini", [Blank 4]=0)) → **D. model**
- Blank 4 (llm = ChatOpenAI(model="gpt-4o-mini", [Blank 4]=0)) → **A. temperature**
- Blank 5 (print(llm.[Blank 5]("Explain what a change window is in network automation.").content)) → **B. invoke**

Explanation

`ChatOpenAI` is the LangChain class used to create a chat model client, so it belongs both in the import and as the constructor. The `model` argument selects `gpt-4o-mini`, and `temperature=0` sets the requested generation setting. Calling `invoke` with the instruction sends it to the model; `.content` reads the response text.

✓ **ChatOpenAI** ChatOpenAI
Use `ChatOpenAI` in the import and to construct the client, as shown by the completed code. It provides the chat-model interface needed to configure the specified model and invoke it.

✓ **temperature** temperature
`temperature` is the constructor argument for the value `0` in the client configuration. It completes the intended `ChatOpenAI(model="gpt-4o-mini", temperature=0)` call.

✓ **invoke** invoke
`invoke` sends the short instruction to the chat model and returns its response. The code then accesses the response's `.content` property for printing.

✗ **OpenAI** OpenAI
The required client in this artifact is `ChatOpenAI`, not `OpenAI`. Using `OpenAI` would not match the chat-model constructor used with this `invoke` call.

✓ **model** model
`model` is the constructor argument that specifies which model the client should use. Here it assigns the required model name, `gpt-4o-mini`.

✗ **ask** ask
`ask` is not the method used by this LangChain chat-model artifact to submit the instruction. The required method is `invoke`, which returns a response whose content can be printed.

✗ **fetch** fetch
`fetch` does not serve as the model invocation method in this code. The instruction must be submitted with `invoke` instead.

Question 79

Single choice

A Cisco Catalyst 9000 switch has guest shell enabled. Which Linux command installs a third-party application?

A. yum Install
B. apt-get install
C. zypper install
D. dnf install

Answer: A. yum Install

Explanation

The Catalyst 9000 Guest Shell uses the yum package manager to install third-party applications. Therefore, option A is the command that matches the switch's Guest Shell environment; the other package-manager commands are not the one specified for this scenario. [2]

https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/prog/b-programmability-cg/m_prog_guestshell.pdf

✓ **A** yum Install

Free demo · full exam: 549 questions

Guest Shell on the Catalyst 9000 uses yum to install packages, so this is the correct choice. The command is conventionally written `yum install` with a lowercase subcommand. [2]

https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/prog/b-programmability-cg/m_prog_questshell.pdf

✗ **B** apt-get install
The Catalyst 9000 Guest Shell uses yum, not apt-get, for installing third-party applications. [2]

https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/prog/b-programmability-cg/m_prog_questshell.pdf

✗ **C** zypper install
The command expected for the Catalyst 9000 Guest Shell is yum install, not zypper install. [2]

https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/prog/b-programmability-cg/m_prog_questshell.pdf

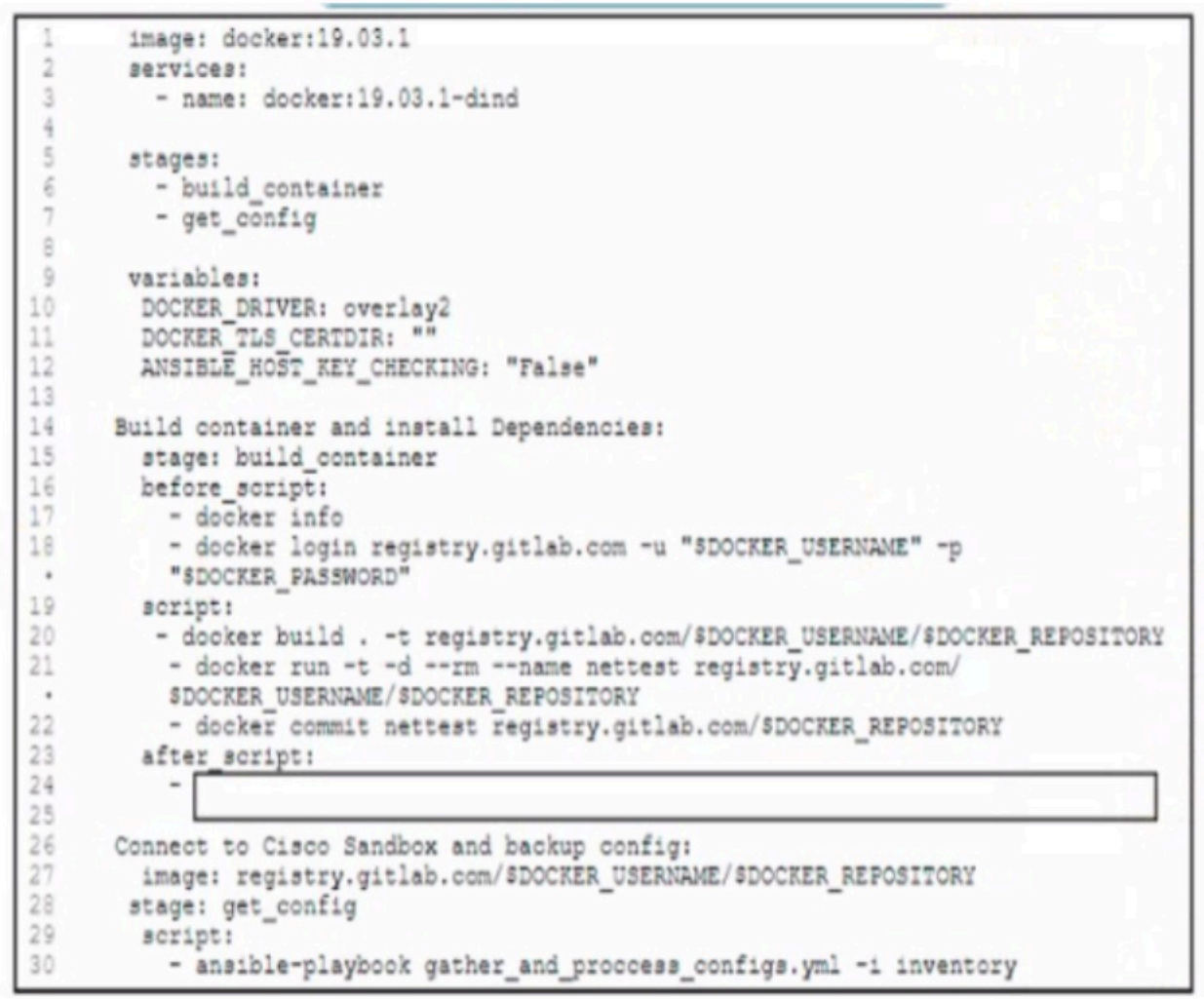
✗ **D** dnf install
The documented package installation command for this Guest Shell scenario is yum install, not dnf install. [2]

https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/prog/b-programmability-cg/m_prog_questshell.pdf

Question 80

Single choice

Refer to the exhibit.



```
image: docker:19.03.1
services:
  - name: docker:19.03.1-dind

stages:
  - build_container
  - get_config

variables:
 DOCKER_DRIVER: overlay2
 DOCKER_TLS_CERTDIR: ""
 ANSIBLE_HOST_KEY_CHECKING: "False"

Build container and install Dependencies:
  stage: build_container
  before_script:
    - docker info
    - docker login registry.gitlab.com -u "$DOCKER_USERNAME" -p
    "$DOCKER_PASSWORD"
  script:
   - docker build . -t registry.gitlab.com/$DOCKER_USERNAME/$DOCKER_REPOSITORY
    - docker run -t -d --rm --name nettest registry.gitlab.com/
    $DOCKER_USERNAME/$DOCKER_REPOSITORY
    - docker commit nettest registry.gitlab.com/$DOCKER_REPOSITORY
  after_script:
    -

Connect to Cisco Sandbox and backup config:
  image: registry.gitlab.com/$DOCKER_USERNAME/$DOCKER_REPOSITORY
 stage: get_config
  script:
    - ansible-playbook gather_and_proccess_configs.yml -i inventory
```

A.

```
docker assign nettest
registry.gitlab.com/DOCKER_USERNAME/$DOCKER_REPOSITORY
```

B.

```
docker info registry.gitlab.com/$DOCKER_REPOSITORY
```

C.

```
docker logout registry.gitlab.com
```

D.

```
docker push registry.gitlab.com/
$DOCKER_USERNAME/$DOCKER_REPOSITORY
```

Answer: D

Explanation

The build job builds and tags the image, starts it, and commits the resulting container as `nettest` under `registry.gitlab.com/$DOCKER_USERNAME/$DOCKER_REPOSITORY`. The next job uses that registry image, so the `after_script` must push the committed image to the GitLab registry. Option D performs that push; the other commands do not publish the image for the next job to retrieve.

✗ **A** `docker assign nettest registry.gitlab.com/$DOCKER_USERNAME/$DOCKER_REPOSITORY`
`docker assign` is not the command needed to publish an image to the registry. The next job needs the image pushed to the registry, which this command does not do.

✗ **B** `docker info registry.gitlab.com/$DOCKER_REPOSITORY`
`docker info` displays information about the Docker environment; it does not upload the committed image to GitLab. Consequently, the later job would not receive the image from this step.

✗ **C** `docker logout registry.gitlab.com`
Logging out removes the registry authentication rather than publishing the image. The build job has already logged in, and it needs to retain that access long enough to push the image.

✓ **D** `docker push registry.gitlab.com/$DOCKER_USERNAME/$DOCKER_REPOSITORY`
The build job has committed the running container as `nettest` using the registry repository name. Pushing that repository uploads the image to GitLab, where the `get_config` job can use the image specified in its `image` field.

Free demo · full exam: 549 questions

Dumpsgate

Free demo · full exam: 549 questions