complete your programming course about resources, doubts and more!

Cisco

(350-901)

Developing Applications using Cisco Core Platforms and APIs
(DEVCOR)

Total: **383 Questions**
Link:

Question: 1

A developer has created an application based on customer requirements. The customer needs to run the application with the minimum downtime.
Which design approach regarding high-availability applications, Recovery Time Objective, and Recovery Point Objective must be taken?

A.Active/passive results in lower RTO and RPO. For RPO, data synchronization between the two data centers must be timely to allow seamless request flow.

B.Active/passive results in lower RTO and RPO. For RPO, data synchronization between the two data centers does not need to be timely to allow seamless request flow.

C.Active/active results in lower RTO and RPO. For RPO, data synchronization between the two data centers does not need to be timely to allow seamless request flow.

D.Active/active results in lower RTO and RPO. For RPO, data synchronization between the two data centers must be timely to allow seamless request flow.

Answer: D

Explanation:

The correct answer is D, which states: "Active/active results in lower RTO and RPO. For RPO, data synchronization between the two data centers must be timely to allow seamless request flow."

Here's why:

Active/Active vs. Active/Passive: Active/active deployments distribute application traffic across multiple active instances simultaneously. This contrasts with active/passive setups where a standby instance remains idle until the primary fails. Active/active provides inherent redundancy and faster failover, directly contributing to lower RTO (Recovery Time Objective). RTO is the maximum acceptable downtime after a failure. With active/active, a failing instance has minimal impact because other instances are immediately available to take the load, leading to near-instant recovery.

Recovery Point Objective (RPO): RPO defines the maximum acceptable data loss in the event of a failure. To ensure minimal data loss with active/active deployment, data synchronization is crucial. Timely data synchronization between all instances within the active/active configuration ensures that if one instance fails, the other instances hold near up-to-date data, thus lowering the RPO. Without timely synchronization, failing over to another instance would result in potential data loss since data might not be identical across all nodes.

Therefore, for seamless request flow and minimal data loss, timely data synchronization between data centers is essential.

Option A is incorrect because while timely synchronization is required, active/passive doesn't yield the lowest RTO and RPO values that active/active does. Options B and C are incorrect due to a combination of reasons. Active/passive doesn't yield the lowest RTO or RPO. Additionally, timely data synchronization is imperative for achieving the lowest RPO, regardless of the active/active or active/passive configuration in an HA setup.

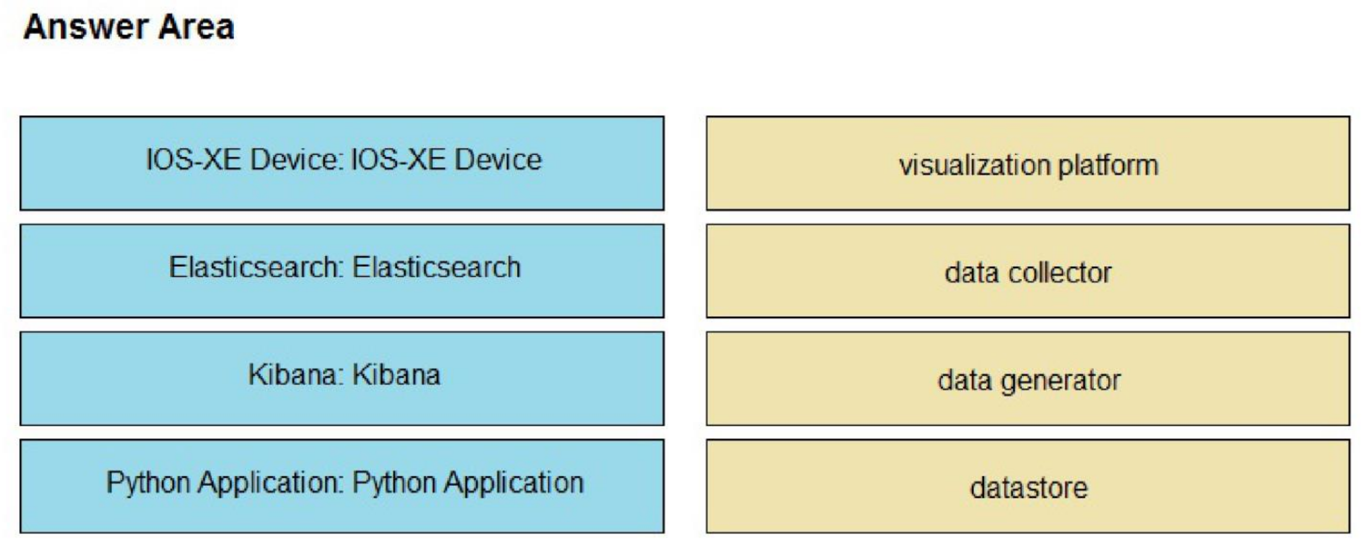
In summary, for applications requiring minimal downtime, an active/active configuration with timely data synchronization delivers the lowest RTO and RPO, meeting the customer's requirement for high availability.

Authoritative Links:

Microsoft Azure - High Availability:https://learn.microsoft.com/en-us/azure/architecture/framework/resiliency/design-patterns#active-active (Focus on the active-active section)
AWS - High Availability (HA) Architecture:https://aws.amazon.com/architecture/well-architected/reliability/
Cloudflare - RTO and RPO: What's the Difference?: https://www.cloudflare.com/learning/cloud/what-is-rto-rpo/

Question: 2

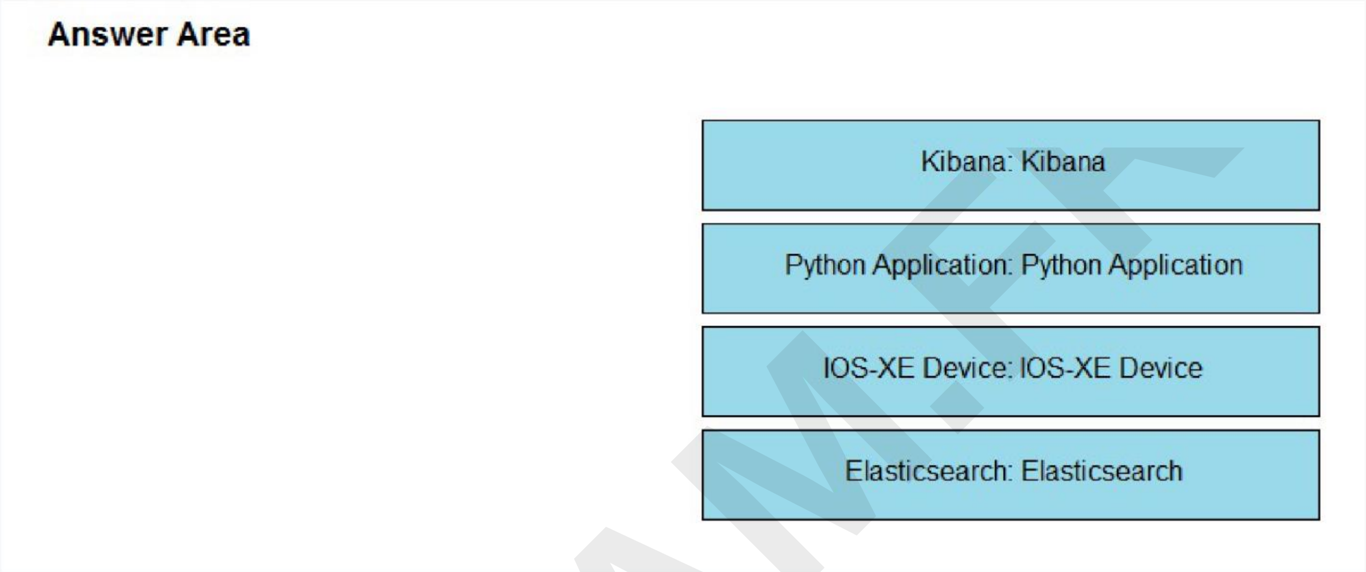
DRAG DROP -
An application is being built to collect and display telemetry streaming data. Drag and drop the elements of this stack from the left onto the correct element functions on the right.
Select and Place:



Answer Area

IOS-XE Device: IOS-XE Device	visualization platform
Elasticsearch: Elasticsearch	data collector
Kibana: Kibana	data generator
Python Application: Python Application	datastore

Answer:



Answer Area

Kibana: Kibana

Python Application: Python Application

IOS-XE Device: IOS-XE Device

Elasticsearch: Elasticsearch

Explanation:

Kibana: Kibana.

Python Application: Python Application.

IOS-XE Device: IOS-XE Device.

ElasticSearch:ElasticSearch.

Question: 3

A cloud native project is being worked on in which all source code and dependencies are written in Python, Ruby, and/or JavaScnpt. A change in code triggers a notification to the CI/CD tool to run the CI/CD pipeline.
Which step should be omitted from the pipeline?

A.Deploy the code to one or more environments, such as staging and/or production.

B.Build one of more containers that package up code and all its dependencies.

C.Compile code.

D.Run automated tests to validate the code.

Answer: C

Explanation:

The correct answer is C. Compile code. This is because Python, Ruby, and JavaScript are interpreted languages, not compiled languages. Compiled languages, like Java or C++, require an explicit compilation step to translate human-readable code into machine code before execution. Interpreted languages, on the other hand, are directly executed by an interpreter, which reads and executes the code line by line. Therefore, a CI/CD pipeline for a project using only these languages does not need a compilation stage. Instead, the source code is directly packaged, often into containers alongside the interpreter and necessary dependencies, and then deployed. This speeds up the pipeline process by removing a step that's irrelevant to the project's architecture. The automated tests are still necessary to ensure quality, and deployment is the culmination of the CI/CD cycle.

Further Research:

Interpreted vs. Compiled Languages:https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/
CI/CD Pipelines:https://www.redhat.com/en/topics/devops/what-is-ci-cd
Containerization:https://www.docker.com/resources/what-container/

Question: 4

Which two statements are considered best practices according to the 12-factor app methodology for application design? (Choose two.)

A.Application code writes its event stream to stdout.

B.Application log streams are archived in multiple replicated databases.

C.Application log streams are sent to log indexing and analysis systems.

D.Application code writes its event stream to specific log files.

E.Log files are aggregated into a single file on individual nodes.

Answer: AC

Explanation:

The correct answers are A and C. The 12-factor app methodology emphasizes stateless processes and treat logs as event streams. Option A aligns with this by stating that application code writes its event stream to stdout (standard output). This enables applications to be easily monitored and have their logs captured by container orchestration platforms like Kubernetes or cloud logging services, as these platforms generally gather output from stdout. This approach decouples logging from the application itself and allows for flexibility in how logs are handled.

Option C, stating that application log streams are sent to log indexing and analysis systems, is also a best practice. The 12-factor methodology views logs as event streams that should be routed to external systems, rather than managed directly by the application or stored in local files. By using indexing and analysis systems, developers can gain valuable insights into application behavior, performance, and identify potential problems. Log aggregation and analysis are important for monitoring and troubleshooting distributed applications.

Option B, D, and E are not consistent with the 12-factor principles. B proposes archiving logs in databases, which goes against the transient and event-stream view. D suggests logging to specific files which can be problematic with container environments and can be harder to gather and analyze. E suggesting log aggregation on individual nodes also goes against the central collection and analysis approach.

Authoritative links:

12-Factor App Methodology:https://12factor.net/ - This website provides a detailed explanation of each of the twelve factors for building scalable and maintainable applications.

Cloud Native Computing Foundation (CNCF):https://www.cncf.io/ - The CNCF provides a vendor-neutral home for many cloud native projects, including tools for logging and monitoring.

Question: 5

An organization manages a large cloud-deployed application that employs a microservices architecture. No notable issues occur with downtime because the services of this application are redundantly deployed over three or more data center regions. However, several times a week reports are received about application slowness. The container orchestration logs show faults in a variety of containers that cause them to fail and then spin up brand new.
Which action must be taken to improve the resiliency design of the application while maintaining current scale?

A.Update the base image of the containers.

B.Test the execution of the application with another cloud services platform.

C.Increase the number of containers running per service.

D.Add consistent try/catch(exception) clauses to the code.

Answer: D

Explanation:

The correct answer is **D. Add consistent try/catch(exception) clauses to the code.** Here's why:

The problem isn't about redundancy or scaling, as the application already has multiple deployments across regions and container restarts are happening as expected. The core issue is that container failures are causing a temporary slowdown as new containers are brought online. This suggests that the application code is not handling exceptions gracefully. When an unexpected condition arises (like a temporary network hiccup, a missing dependency, or an invalid data input), the microservices are crashing instead of recovering, triggering the container restarts.

Adding try/catch blocks around potentially problematic code segments allows the application to intercept errors, handle them gracefully, and prevent the microservice from crashing. Instead of abruptly failing, the service can log the error, retry the operation, or return an appropriate error response. This significantly improves application resilience by minimizing failures and preventing the cascading impact of uncontrolled exceptions. Options A, B, and C address different aspects but do not address the root cause of error handling. Option A deals with potential image issues but does not prevent runtime exceptions. Option B is testing, which would be an additional solution but would not address runtime errors. Option C addresses scaling and also does not directly prevent errors. Therefore, the most effective immediate action to reduce container restarts and improve responsiveness is to focus on robust exception handling within the application code.

Here are some authoritative resources to further explore this topic:

Microsoft Documentation on Exception Handling:https://learn.microsoft.com/en-us/dotnet/standard/exceptions/ This article covers exception handling in .NET, but the concepts are broadly applicable to any programming language.

Google Cloud Architecture Center on Reliability:https://cloud.google.com/architecture/reliability Explores general principles and patterns for building reliable cloud applications, including error handling.

AWS Well-Architected Framework on Reliability Pillar:https://aws.amazon.com/architecture/well-architected/reliability/ Provides guidance on designing reliable systems on AWS, including strategies for handling failures.

Kubernetes Best Practices on Resource Management:
https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ Although this is about managing resources in containers, understanding this will allow a better understanding of issues that cause restarts and how exception handling can address these issues.

In summary, well-implemented exception handling is a key practice for building resilient microservices, helping to minimize downtime and improve overall application stability and performance.

Question: 6

How should a web application be designed to work on a platform where up to 1000 requests per second can be served?

A.Use algorithms like random early detection to deny excessive requests.

B.Set a per-user limit (for example, 5 requests/minute/user) and deny the requests from the users who have reached the limit.

C.Only 1000 user connections are allowed; further connections are denied so that all connected users can be served.

D.All requests are saved and processed one by one so that all users can be served eventually.

Answer: B

Explanation:

The correct answer is **B. Set a per-user limit (for example, 5 requests/minute/user) and deny the requests from the users who have reached the limit.** This approach, known as rate limiting, is a fundamental technique for managing web application traffic and ensuring fair resource allocation.

Option A, using algorithms like Random Early Detection (RED), is primarily used for network congestion control, not for application-level request management. RED focuses on dropping packets at the network layer to prevent network overload; it's not directly applicable to managing user requests within an application. Option C, limiting to 1000 concurrent connections, could lead to unfair service as new users might be entirely blocked even if existing connections are idle. Option D, queuing all requests, can result in significant latency if there are more than 1000 requests in a second, potentially leading to a backlog and poor user experience.

Rate limiting, option B, provides a more granular control. By setting a per-user limit, the system prevents any single user from overwhelming the application with requests, ensuring that others receive their fair share of resources. This is a common strategy to mitigate denial-of-service (DoS) attacks or accidental traffic spikes caused by a single user. It allows for continuous service for most users while preventing resource exhaustion. Key benefits include fairness, improved response times, and enhanced security.

For further research, see these authoritative resources:

Cloudflare's article on Rate Limiting:https://www.cloudflare.com/learning/ddos/rate-limiting/
AWS documentation on API Gateway Throttling:
https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html **Google Cloud documentation on Rate Limiting:**https://cloud.google.com/api-gateway/docs/rate-limiting

Question: 7

An organization manages a large cloud-deployed application that employs a microservices architecture across multiple data centers. Reports have been received about application slowness. The container orchestration logs show that faults have been raised in a variety of containers that caused them to fail and then spin up brand new instances.
Which two actions can improve the design of the application to identify the faults? (Choose two.)

A.Automatically pull out the container that fails the most over a time period.

B.Implement a tagging methodology that follows the application execution from service to service.

C.Add logging on exception and provide immediate notification.

D.Do a write to the datastore every time there is an application failure.

E.Implement an SNMP logging system with alerts in case a network link is slow.

Answer: BC

Explanation:

Okay, let's break down why options B and C are the correct choices for identifying faults in the described microservices application, and why the others are less effective.

Why B is correct: Implementing a tagging methodology (also known as tracing) is crucial for understanding the flow of a request through a complex microservices architecture. When a request enters the system, it's assigned a unique identifier (tag). This tag is then propagated as the request moves from one service to another. This allows you to reconstruct the entire path of the request, pinpointing exactly which service(s) are experiencing issues (latency, errors, etc.). Without tracing, identifying bottlenecks across multiple services becomes exceptionally difficult, making fault diagnosis a guessing game. It helps connect the dots when problems occur.
https://opentelemetry.io/docs/concepts/tracing/

Why C is correct: Adding logging on exceptions and providing immediate notifications is a cornerstone of effective monitoring and error handling in cloud environments. When an exception occurs within a service, capturing detailed information about the error (type, stack trace, timestamp, etc.) is invaluable for debugging and root cause analysis. Furthermore, sending immediate notifications (alerts) to operations teams enables them to react swiftly, minimizing the impact of errors. This pro-active approach enables quick problem resolution. https://cloud.google.com/logging/docs/

Why A is incorrect: Automatically removing a container based on failure frequency is a reactive, not a diagnostic, measure. While it may temporarily alleviate an issue, it doesn't identify the underlying cause of the failures. In fact, a container failing repeatedly might signal a deeper configuration or code problem that needs investigation, not just a band-aid solution. It does not contribute towards fixing the root cause.

Why D is incorrect: Writing to a datastore on every application failure creates unnecessary overhead and can potentially exacerbate performance issues. Moreover, this approach does not offer specific insight into where the fault lies within the service flow, unlike tracing. Storing error logs to datastore is a better fit.

Why E is incorrect: SNMP is a protocol mainly used for network device monitoring. While network slowness could indirectly affect the application, focusing on service-level logging and tracing is more relevant for diagnosing application-specific faults. Slow networks may not necessarily be the primary reason for application slowness. It is an infra issue, not an application issue.

In summary, options B and C are critical tools for gaining visibility into a distributed microservices architecture and enabling proactive fault identification. Tracing gives you the big picture, while detailed logging with notifications facilitates granular understanding and quick responses to exceptions.

Question: 8

Which two situations are flagged by software tools designed for dependency checking in continuous integration environments, such as OWASP? (Choose two.)

A.publicly disclosed vulnerabilities related to the included dependencies
B.mismatches in coding styles and conventions in the included dependencies
C.incompatible licenses in the included dependencies
D.test case failures introduced by bugs in the included dependencies
E.buffer overflows to occur as the result of a combination of the included dependencies

Answer: AE

Explanation:

Here's a breakdown of why options A and E are correct, and why the others aren't, in the context of dependency checking tools within CI/CD:

Dependency checking tools in continuous integration (CI) environments, like those guided by OWASP guidelines, primarily focus on security and legal compliance regarding the third-party libraries your application uses. Option A is correct because a crucial function of these tools is to identify publicly known vulnerabilities in your dependencies. These vulnerabilities, often listed in databases like the National Vulnerability Database (NVD), represent potential security risks that attackers could exploit. Therefore, flagging these vulnerabilities is essential for proactive security management. Similarly, option E is correct. Dependency interactions can sometimes introduce unintended behavior, such as buffer overflows, by using components with different assumptions or handling data inconsistently. Detection of these interactions are important in the CI environment.

Option B, mismatches in coding styles, isn't typically the concern of dependency checking tools. They focus more on functionality and security than stylistic adherence; this is usually addressed by code linters and style checkers. Option C, while important, is not directly detected by typical dependency checking tools; license compliance tools or processes handle this as a separate compliance check. Option D, test case failures, are typically identified through unit tests and other testing methods, not dependency checks, which focus on known issues in used code rather than the results of a tests.

Therefore, the core task of dependency checkers lies in identifying security vulnerabilities (A) and potential runtime errors like overflows (E) introduced by combined dependencies, thereby strengthening the security posture of the application.

Authoritative Links for further research:

OWASP Dependency-Check:https://owasp.org/www-project-dependency-check/
National Vulnerability Database (NVD):https://nvd.nist.gov/
Snyk (Commercial Dependency Scanner):https://snyk.io/ (Example of a tool following OWASP principles).
CISA - Known Exploited Vulnerabilities Catalog:https://www.cisa.gov/known-exploited-vulnerabilities-catalog

Question: 9

A network operations team is using the cloud to automate some of their managed customer and branch locations. They require that all of their tooling be ephemeral by design and that the entire automation environment can be recreated without manual commands. Automation code and configuration state will be stored in git for change control and versioning. The engineering high-level plan is to use VMs in a cloud-provider environment, then configure open source tooling onto these VMs to poll, test, and configure the remote devices, as well as deploy the tooling itself.

Which configuration management and/or automation tooling is needed for this solution?

A.Ansible
B.Ansible and Terraform
C.NSO
D.Terraform
E.Ansible and NSO

Answer: B

Explanation:

The correct answer is **B. Ansible and Terraform**. Here's why:

The scenario calls for ephemeral infrastructure that can be recreated programmatically, along with configuration management for the tools deployed on those VMs. Terraform excels at infrastructure as code (IaC). It can provision and manage the virtual machines in the cloud environment, ensuring that the entire automation environment, from VMs to network configurations, is defined in code. This addresses the requirement for ephemeral design and fully automated recreation.

Ansible, on the other hand, is a powerful configuration management and automation tool. After Terraform provisions the VMs, Ansible can be used to configure the operating systems, install the necessary open-source tools (like monitoring or testing applications), and manage their configurations. It ensures that the required tools are correctly set up and consistently configured across all VMs.

Both tools complement each other in this solution. Terraform manages the lifecycle of the infrastructure (VMs), while Ansible manages the software and configurations on those VMs. This separation of concerns promotes a more manageable and scalable environment. Using just Ansible wouldn't address the ephemeral infrastructure requirement, and using just Terraform would leave the configuration of tools on the VMs unmanaged. NSO (Network Services Orchestrator) is not appropriate here as its primary function is network service orchestration and automation on Cisco devices.

Therefore, combining Terraform for infrastructure provisioning and Ansible for configuration management provides a complete solution. They both use declarative configurations that make automation repeatable and version-controlled through Git.

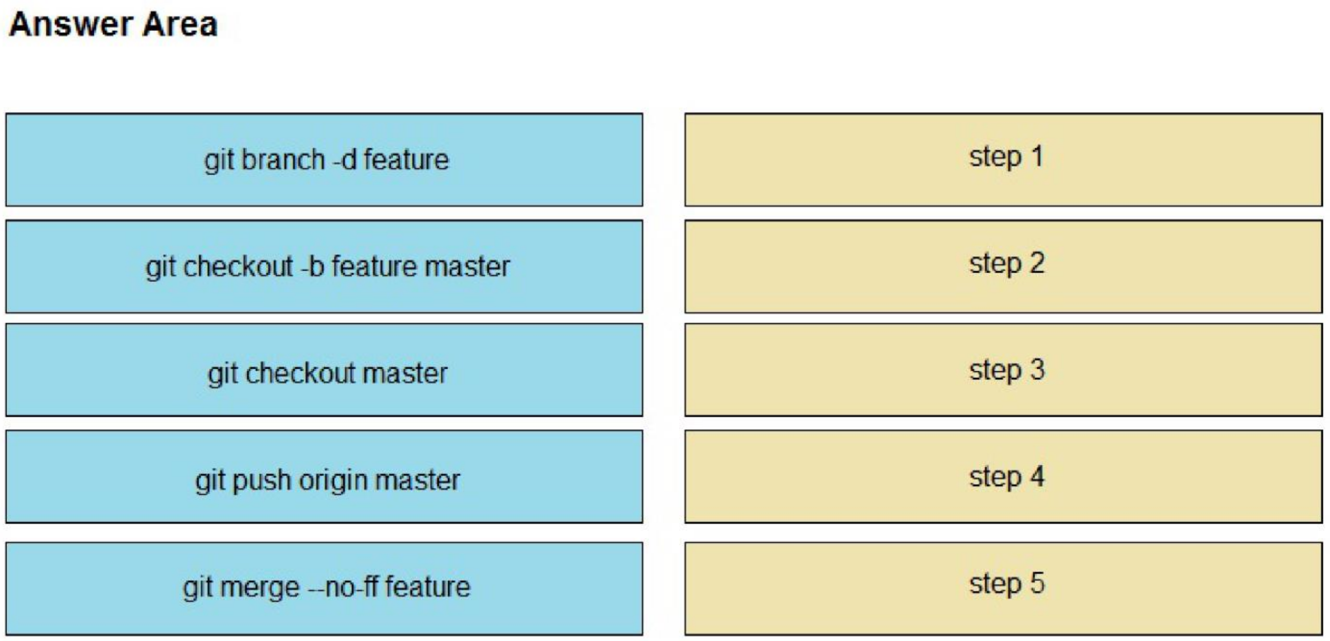
Authoritative Links:

Terraform:https://www.terraform.io/ - Official website for Terraform documentation and tutorials.
Ansible:https://www.ansible.com/ - Official website for Ansible documentation, use cases, and tutorials.

This combined approach allows the network operations team to achieve their goal of a fully automated, ephemeral, and reproducible automation environment.

Question: 10

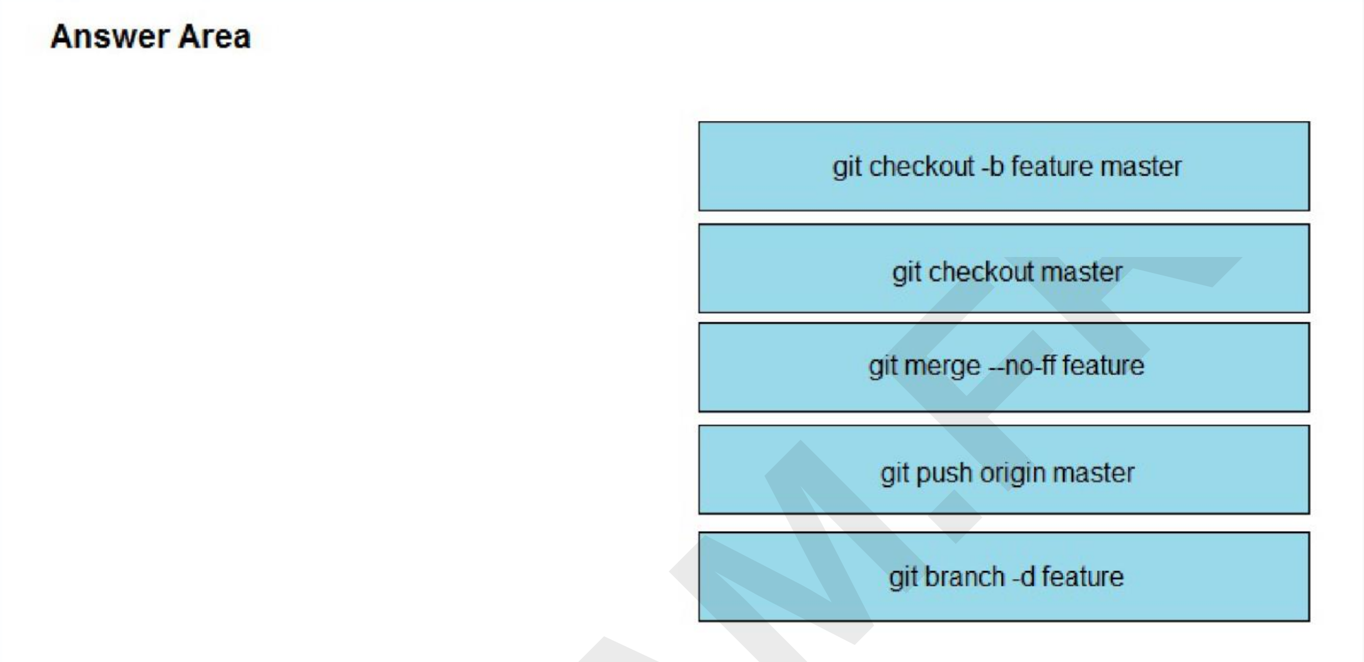
DRAG DROP -
Drag and drop the git commands from the left into the correct order on the right to create a feature branch from the master and then incorporate that feature branch into the master.
Select and Place:



Answer Area

git branch -d feature	step 1
git checkout -b feature master	step 2
git checkout master	step 3
git push origin master	step 4
git merge --no-ff feature	step 5

Answer:



Answer Area

git checkout -b feature master

git checkout master

git merge --no-ff feature

git push origin master

git branch -d feature

Explanation:

Step 1 . git checkout -b feature master (Create new branch called feature and switch) Step 2 .

git checkout master (switch back to Master branch)

Step 3 . git merge –no-ff feature (Merge changes from feature to master branch) Step 4 .git

push origin master (Push changes to remote repo)

Step 5 . git branch -d feature (Delete the branch called feature)

Question: 11

An application is hosted on Google Kubernetes Engine. A new JavaScript module is created to work with the existing application. Which task is mandatory to make the code ready to deploy?

A.Create a Dockerfile for the code base.
B.Rewrite the code in Python.
C.Build a wrapper for the code to "containerize" it.
D.Rebase the code from the upstream git repo.

Answer: A

Explanation:

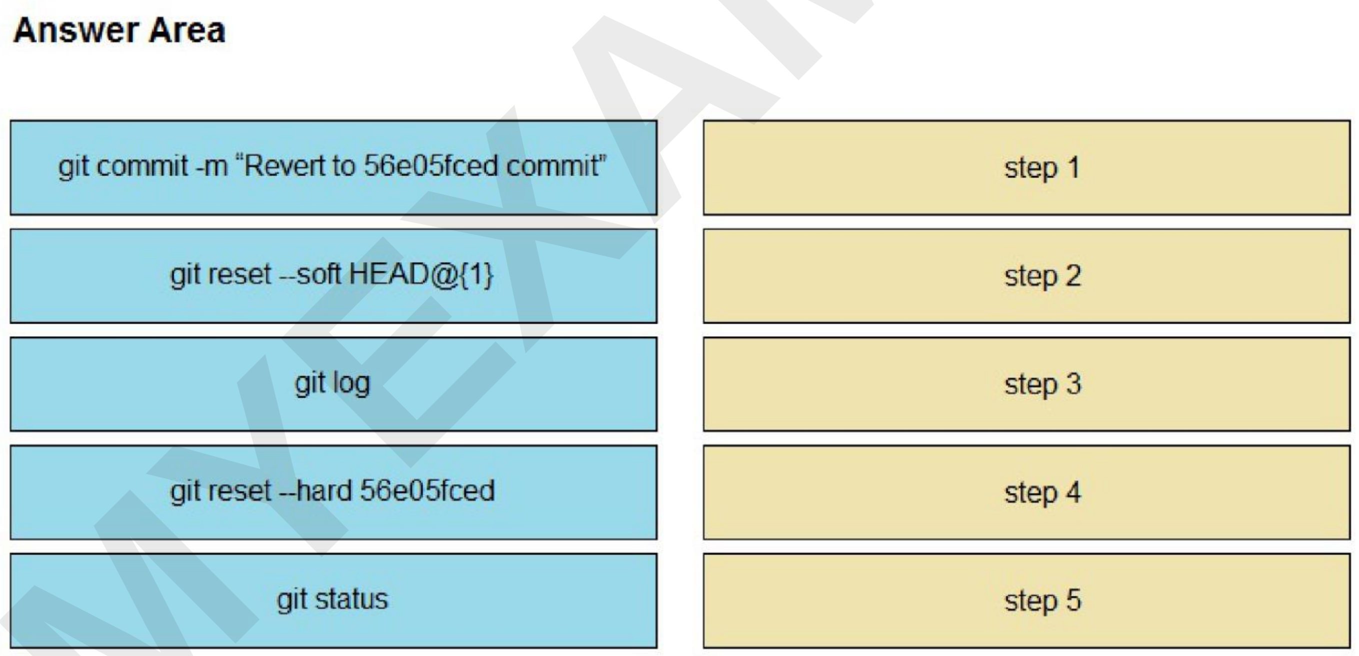
The correct answer is A, creating a Dockerfile for the code base. This is mandatory because Kubernetes, including Google Kubernetes Engine (GKE), primarily manages and deploys applications packaged as containers. A Dockerfile is a text document containing instructions to build a Docker image, which is a lightweight, stand-alone executable package that includes everything needed to run the application. Options B, C, and D are not mandatory. Rewriting the code in Python (B) is unnecessary if the JavaScript module functions correctly. While you could create a "wrapper" (C), this step is effectively accomplished by Dockerizing the application. Rebasing the code from an upstream Git repo (D) is unrelated to preparing the application for deployment within a containerized environment. A Docker image ensures portability and consistency by encapsulating the code, its dependencies, and runtime environment. Kubernetes then uses these images to deploy and scale applications across clusters. Without containerization, Kubernetes cannot directly manage and run the JavaScript module as it lacks the necessary isolation and packaging to work within the container orchestration system. A Dockerfile is crucial for automating the image creation process, guaranteeing repeatable builds and streamlined deployments on GKE. Therefore, containerization via a Dockerfile is the fundamental step before deploying applications on Kubernetes and GKE.

Further Research:

Docker Documentation:https://docs.docker.com/
Kubernetes Documentation:https://kubernetes.io/docs/
Google Kubernetes Engine (GKE) Documentation:https://cloud.google.com/kubernetes-engine/docs

Question: 12

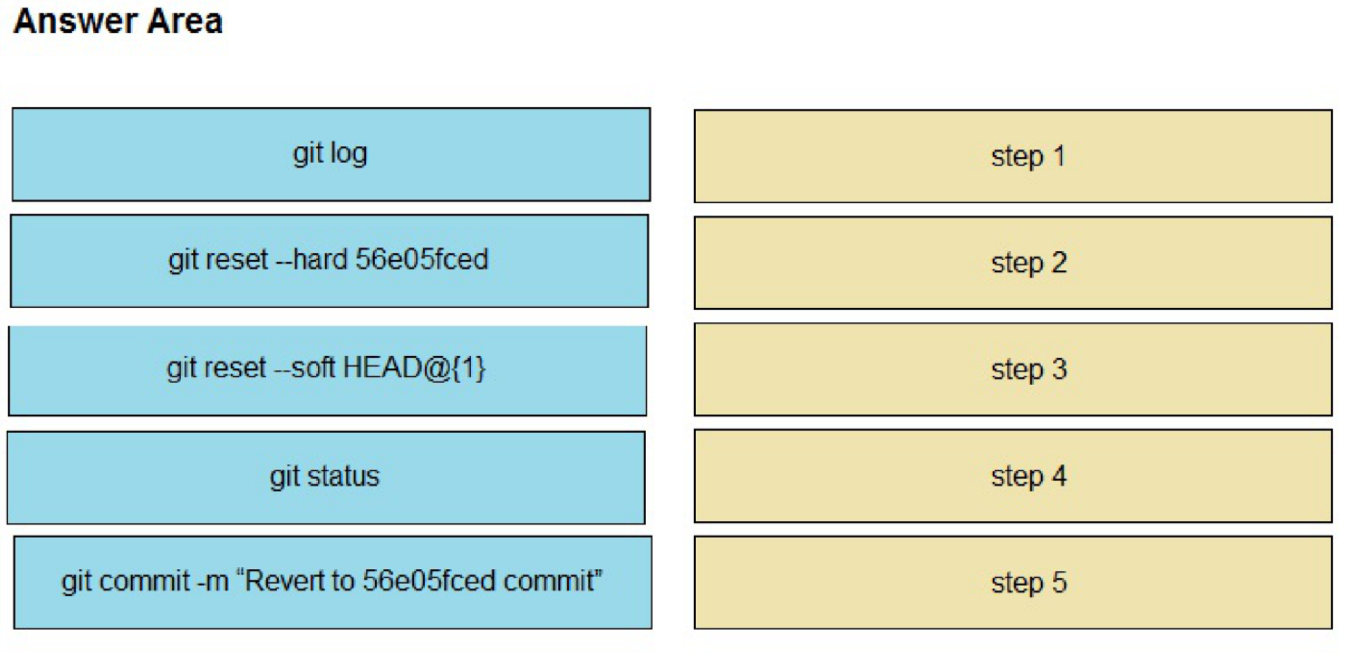
DRAG DROP -
Drag and drop the steps from the left into the correct sequence on the right to describe how to use Git to maintain the current HEAD and revert back to a previous commit, while undoing all intermediate commits.
Select and Place:



Answer Area

git commit -m “Revert to 56e05fced commit”	step 1
git reset --soft HEAD@{1}	step 2
git log	step 3
git reset --hard 56e05fced	step 4
git status	step 5

Answer:



Answer Area

git log	step 1
git reset --hard 56e05fced	step 2
git reset --soft HEAD@{1}	step 3
git status	step 4
git commit -m “Revert to 56e05fced commit”	step 5

Explanation:

Step1. git log

Step2. git reset --hard 56e05fced

Step3. git reset --soft HEAD@ 1

Step4. git status

Step5. git commit -m "Revert to 56e05fced commit"

Question: 13

Which database type should be used with highly structured data and provides support for ACID transactions?

A.time series
B.document
C.graph
D.relational

Answer: D

Explanation:

The correct answer is **D. Relational**. Relational databases are specifically designed to handle highly structured data. This structure is achieved through the use of tables with predefined schemas, consisting of rows and columns, where each column represents a specific attribute and each row a record. A key strength of relational databases is their support for ACID transactions. ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties ensure data validity and reliability, especially in environments with concurrent operations. Atomicity guarantees that a transaction is treated as a single unit of work (all changes succeed or none do). Consistency ensures that transactions maintain data integrity, moving the database from one valid state to another. Isolation keeps transactions separate and avoids interference from concurrent operations. Durability ensures that committed changes remain persistent even in case of system failures.

Other database types have different strengths. Time series databases are optimized for time-stamped data, document databases handle semi-structured data in JSON or XML, and graph databases are suitable for interconnected data relationships. These aren't designed for the same level of structure and transactional guarantees as a relational database. Therefore, for structured data requiring ACID transactional support, relational databases are the optimal choice. For deeper understanding, you can review resources from Oracle (https://docs.oracle.com/cd/E11882_01/server.112/e10729/transact.htm), a leading relational database provider, or from Microsoft on SQL Server (https://learn.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-ver16).

Question: 14

Where should distributed load balancing occur in a horizontally scalable architecture?

A.firewall-side/policy load balancing
B.network-side/central load balancing
C.service-side/remote load balancing
D.client-side/local load balancing

Answer: D

Explanation:

The correct answer is **D. client-side/local load balancing.** This approach distributes load balancing logic directly to the client application. In a horizontally scalable architecture, distributing requests from the client rather than relying on a central load balancer is crucial for maximizing flexibility, reducing latency, and avoiding single points of failure. Client-side load balancing allows the client to have awareness of available service instances. It can then choose the optimal instance based on factors like latency or load. With this approach, clients typically maintain a list of available servers, and employ algorithms (like round-robin or weighted random) to distribute requests. This eliminates the overhead and potential bottleneck of a central load balancer, particularly as the system scales out. Central load balancers can become single points of contention. Local load balancing enhances resilience because if a service instance goes down, the client can redirect requests to another healthy instance, without needing to notify and rely on a central intermediary. It is important to remember this distributes complexity from a single point to each client, which might involve added considerations in client implementation. This method is also suitable for microservices-based systems where multiple instances of services are deployed.For further research, consider exploring these resources:Load Balancing MethodsClient-Side Load BalancingHorizontal Scaling

Question: 15

Which two statements about a stateless application are true? (Choose two.)

A.Different requests can be processed by different servers.
B.Requests are based only on information relayed with each request.
C.Information about earlier requests must be kept and must be accessible.
D.The same server must be used to process all requests that are linked to the same state.
E.No state information can be shared across servers.

Answer: AB

Explanation:

The correct options are A and B. A stateless application, by definition, does not maintain any session data or information about prior interactions. Option A is correct because each request is self-contained and can be handled by any available server instance. Load balancing is simplified, as requests can be routed to different servers without concern for session affinity. Option B is also correct; the server only requires information provided within the current request itself to process it. The request carries all necessary data. There is no need for the server to refer to past interactions or maintained state. This self-contained nature allows for easy scaling and resilience. Option C is incorrect because stateless applications explicitly avoid keeping
information about prior requests. Option D is incorrect because stateful applications are the ones requiring the same server for requests related to the same state. Option E is not entirely true, state information is simply not maintained at the application layer; other services like databases or caching systems can manage the state.

Further research on stateless applications can be found here:

AWS:https://aws.amazon.com/what-is/stateless-application/
Microsoft Azure:https://learn.microsoft.com/en-us/azure/architecture/best-practices/stateless-applications
Google Cloud:https://cloud.google.com/architecture/best-practices-for-building-stateless-applications

Question: 16

Which statement about microservices architecture is true?

A.Applications are written in a single unit.

B.It is a complex application composed of multiple independent parts.

C.It is often a challenge to scale individual parts.

D.A single faulty service can bring the whole application down.

Answer: B

Explanation:

The correct answer is B, "It is a complex application composed of multiple independent parts." Microservices architecture fundamentally breaks down a monolithic application into a suite of small, independent services.
Each service focuses on a specific business capability and can be developed, deployed, and scaled independently. This approach promotes modularity and allows for greater flexibility and agility in development. Option A is incorrect because microservices explicitly avoid a single monolithic unit. Option C is incorrect, as a key benefit of microservices is the ability to scale individual components as needed. Option D is also incorrect, as a well-designed microservices architecture aims for fault isolation, meaning that failure in one service should not necessarily bring down the entire application. Robustness is achieved through techniques like circuit breakers and graceful degradation. The independent nature of microservices allows for teams to work on different aspects of the application simultaneously, enhancing parallel development efforts. Furthermore, this architecture enables the utilization of different technologies for different services based on the most suitable tool for the specific job. This offers a significant advantage in adapting to changing technology landscapes.

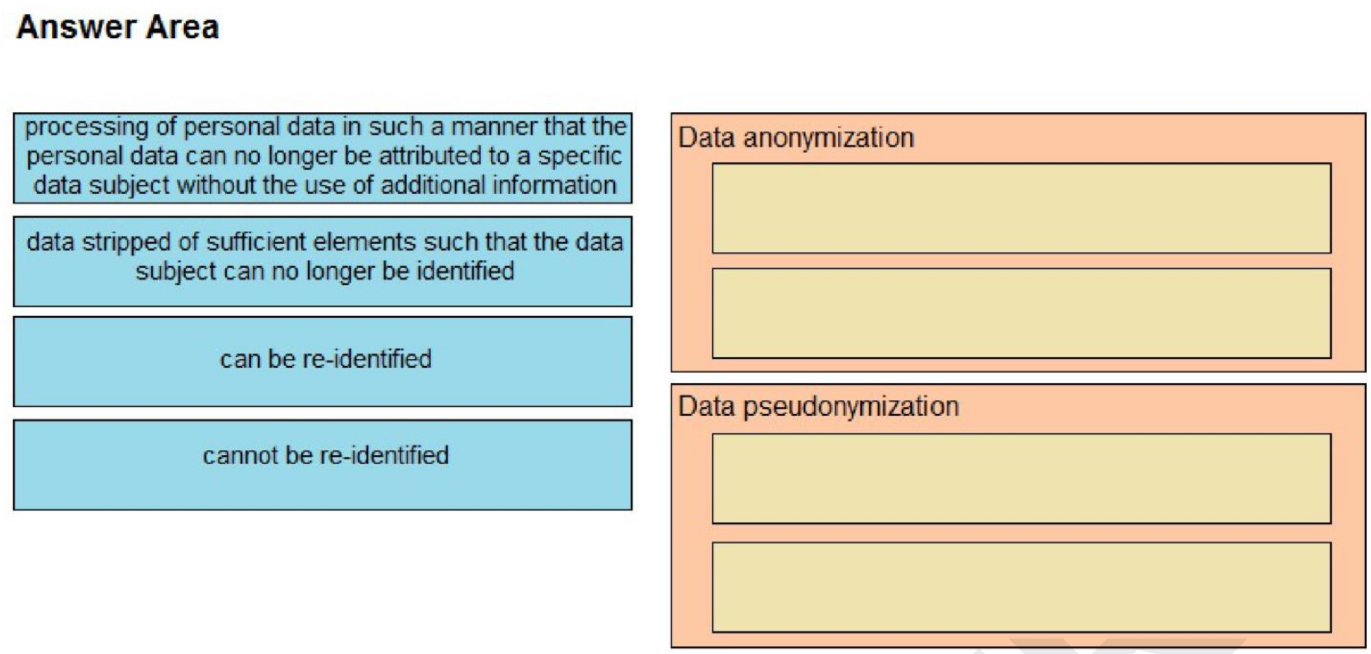
Further research can be conducted on the following resources:

Microservices.io: This site provides comprehensive information on microservices architecture, including patterns, principles, and benefits: https://microservices.io/
Martin Fowler's Blog: Martin Fowler, a pioneer in software architecture, has written extensively on microservices: https://martinfowler.com/microservices/
AWS Microservices: Amazon Web Services provides resources on using microservices architecture within their cloud environment: https://aws.amazon.com/microservices/

Microsoft Azure Microservices: Microsoft's documentation for microservices on Azure provides another view of its implementation in their cloud: https://azure.microsoft.com/en-us/solutions/microservices/These links offer detailed explanations and various perspectives on this crucial architectural paradigm.

Question: 17

DRAG DROP -
Drag and drop the characteristics from the left onto the correct data processing techniques on the right, in the context of GDPR.
Select and Place:



Answer Area

processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information

data stripped of sufficient elements such that the data subject can no longer be identified

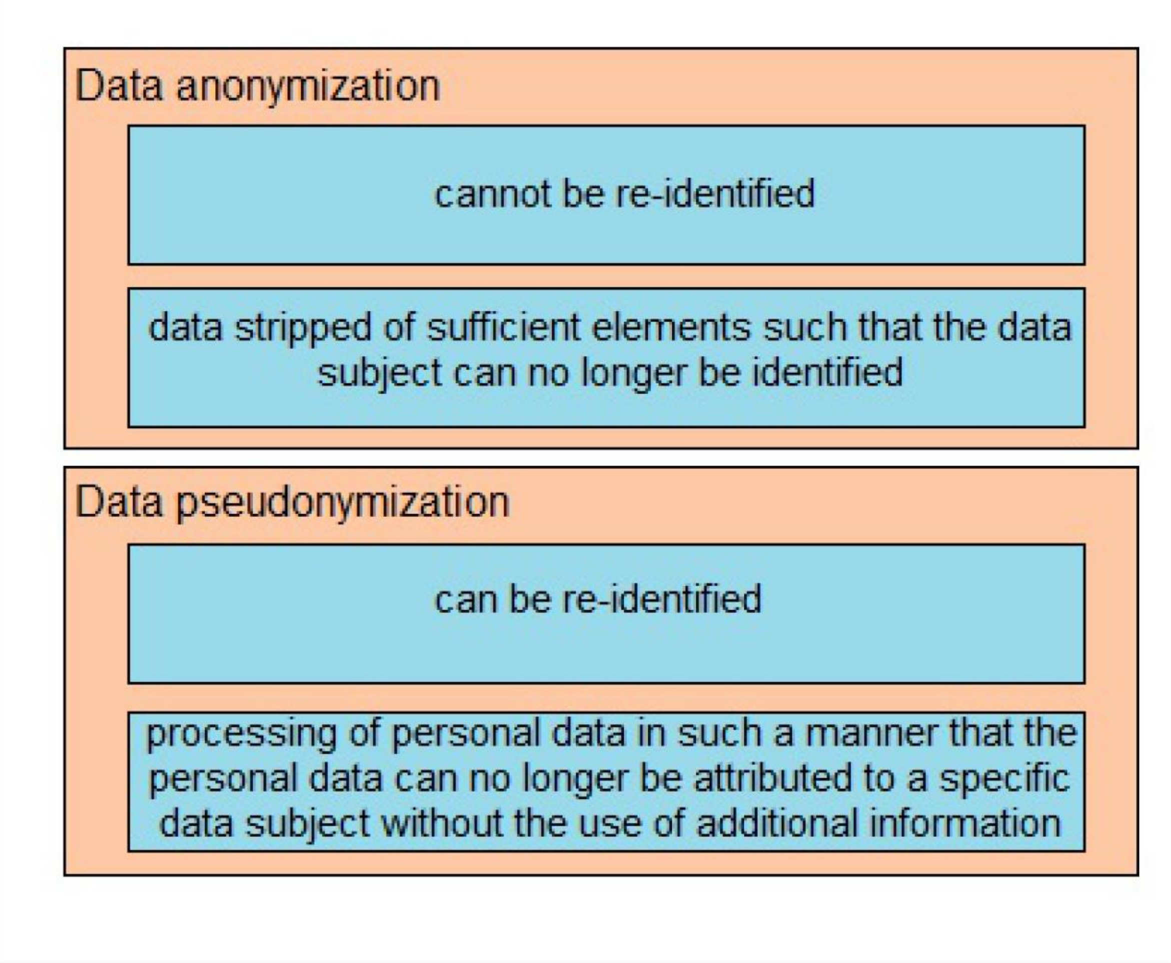
can be re-identified

cannot be re-identified

Data anonymization

Data pseudonymization

Answer:



Data anonymization

cannot be re-identified

data stripped of sufficient elements such that the data subject can no longer be identified

Data pseudonymization

can be re-identified

processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information

Explanation:

Data anonymization:- cannot be re-identified.

data stripped of sufficient element Such that the data Subject Can no longer be identified.

Data pseudonymization:- can be re-identified-
processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information.

Question: 18 Which two data encoding techniques are supported by gRPC? (Choose two.)

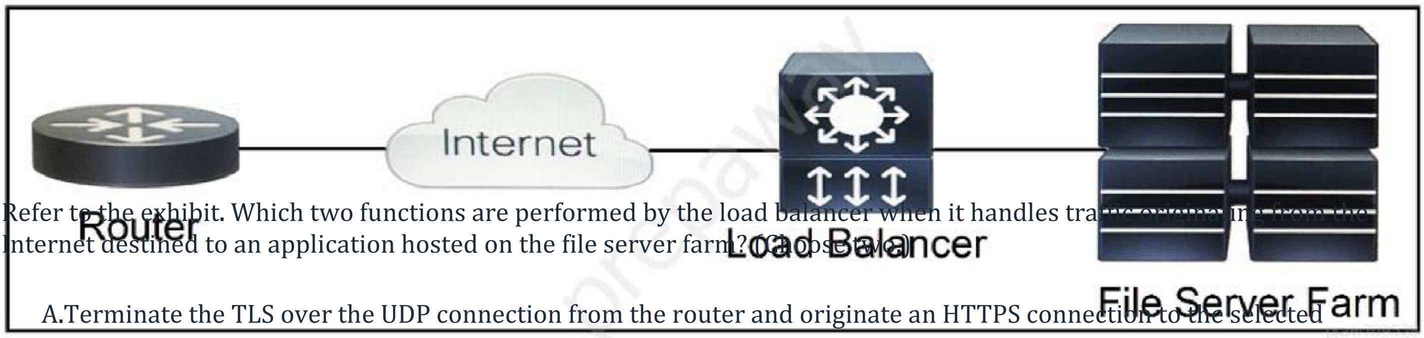
A.XML
B.JSON
C.ASCII
D.ProtoBuf
E.YAML

Answer: BD

Explanation:

The correct answer is B (JSON) and D (ProtoBuf). gRPC, a high-performance, open-source universal RPC framework, primarily uses Protocol Buffers (ProtoBuf) for data serialization by default. ProtoBuf provides a compact, efficient, and language-neutral mechanism to define data structures and serialize data for transmission over the network, thus minimizing bandwidth consumption and improving performance. While ProtoBuf is the primary and highly recommended encoding for gRPC, JSON is also supported as an alternative data encoding format. This flexibility allows gRPC to integrate with systems that already utilize JSON or where human-readability is a priority. JSON's text-based nature makes it suitable for debugging and quick integrations, though it generally results in larger message sizes compared to ProtoBuf. XML, ASCII, and YAML are not officially supported encoding formats within the gRPC framework. gRPC focuses on binary and highly performant encodings, making options like ProtoBuf a priority. While one could theoretically implement custom serialization, it goes beyond the supported framework and isn't the typical usage. ProtoBuf benefits gRPC by providing stronger typing and compatibility, enhancing developer productivity and ensuring reliable data exchange between services. For further research, refer to the official gRPC documentation https://grpc.io/docs/what-is-grpc/core-concepts/ and specific details on supported encodings in the gRPC specification. Information on protocol buffers can be found at https://developers.google.com/protocol-buffers.

Question: 19



Refer to the exhibit. Which two functions are performed by the load balancer when it handles traffic originating from the Internet destined to an application hosted on the file server farm? (Choose two.)

A.Terminate the TLS over the UDP connection from the router and originate an HTTPS connection to the selected server.

B.Terminate the TLS over the UDP connection from the router and originate an HTTP connection to the selected server.

C.Terminate the TLS over the TCP connection from the router and originate an HTTP connection to the selected server.

D.Terminate the TLS over the TCP connection from the router and originate an HTTPS connection to the selected server.

E.Terminate the TLS over the SCTP connection from the router and originate an HTTPS connection to the selected server.

Answer: CD

Explanation:

C.Terminate the TLS over the TCP connection from the router and originate an HTTP connection to the selected server.

D.Terminate the TLS over the TCP connection from the router and originate an HTTPS connection to the selected server.

Question: 20

Which transport protocol is used by gNMI?

A.HTTP/2
B.HTTP 1.1
C.SSH
D.MQTT

Answer: A

Explanation:

The correct answer is **A. HTTP/2.** gNMI (gRPC Network Management Interface) leverages gRPC as its underlying framework. gRPC, in turn, commonly uses HTTP/2 as its transport protocol. This choice is fundamental because HTTP/2 offers significant performance benefits over HTTP 1.1, such as multiplexing, header compression, and server push, all of which are crucial for high-throughput, low-latency communication needed in network management scenarios. Multiplexing allows multiple requests and responses to share a single TCP connection, reducing the overhead of establishing new connections for each interaction. Header compression minimizes the size of transmitted headers, further optimizing bandwidth usage. Server push enables the server to proactively send data that it anticipates the client will need, reducing latency. These advantages make HTTP/2 the ideal transport for gNMI, which frequently handles large volumes of telemetry data and configuration updates. While other protocols like SSH and MQTT have their specific uses, they are not part of the gNMI standard. SSH is primarily for secure shell access, and MQTT is often used in IoT scenarios with constrained devices and unreliable networks. In contrast, gNMI is intended for robust, high-performance network management within a data center or service provider environment where HTTP/2's characteristics are a better fit.

Authoritative Links for Further Research:

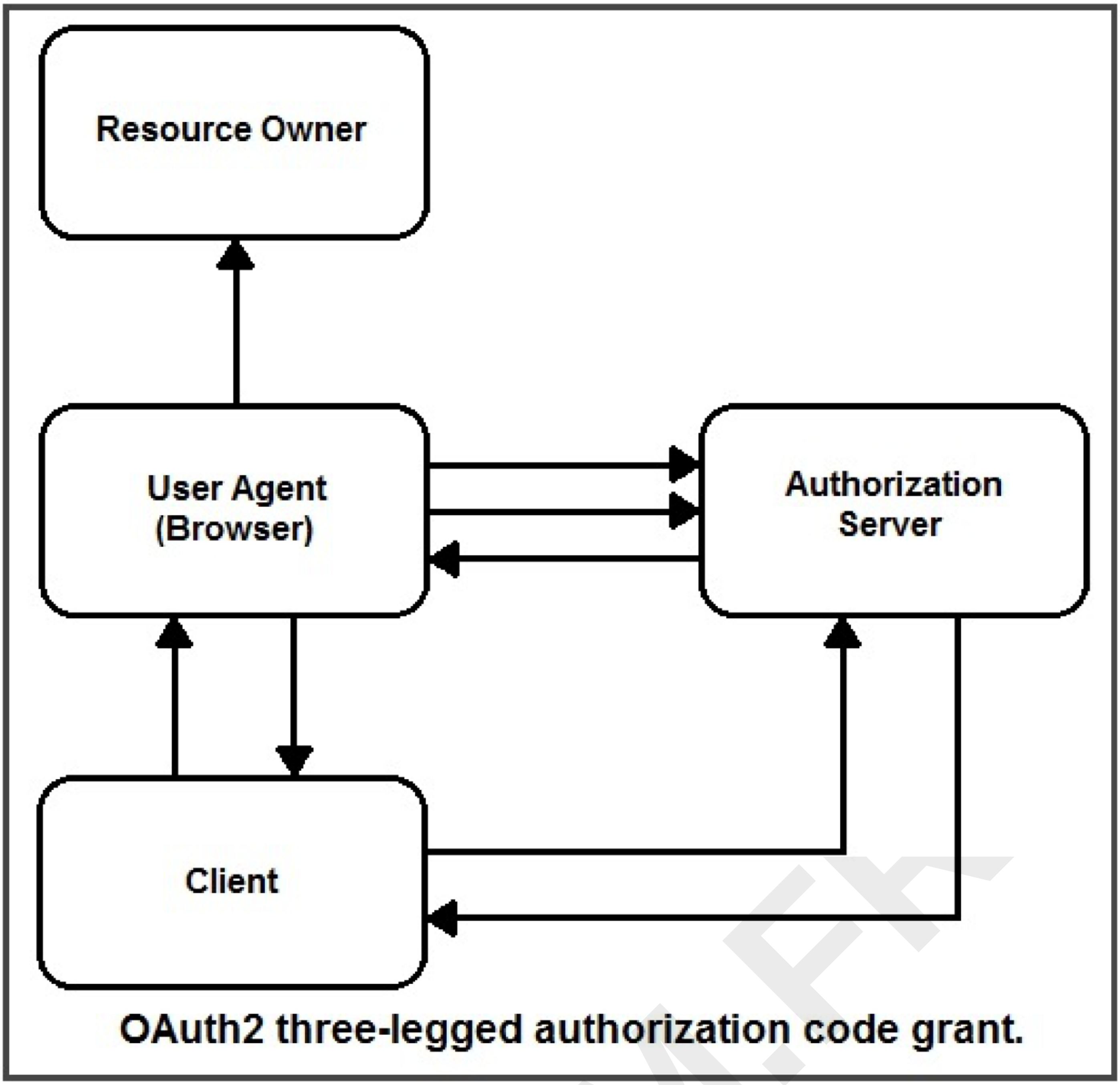
gRPC Documentation:https://grpc.io/docs/ **-** Provides comprehensive information about gRPC, including its usage of HTTP/2.

HTTP/2 RFC:https://datatracker.ietf.org/doc/html/rfc7540 **-** The official specification for HTTP/2. **gNMI Specification:**https://github.com/openconfig/gnmi **-** The official repository for the gNMI specification, outlining its reliance on gRPC.

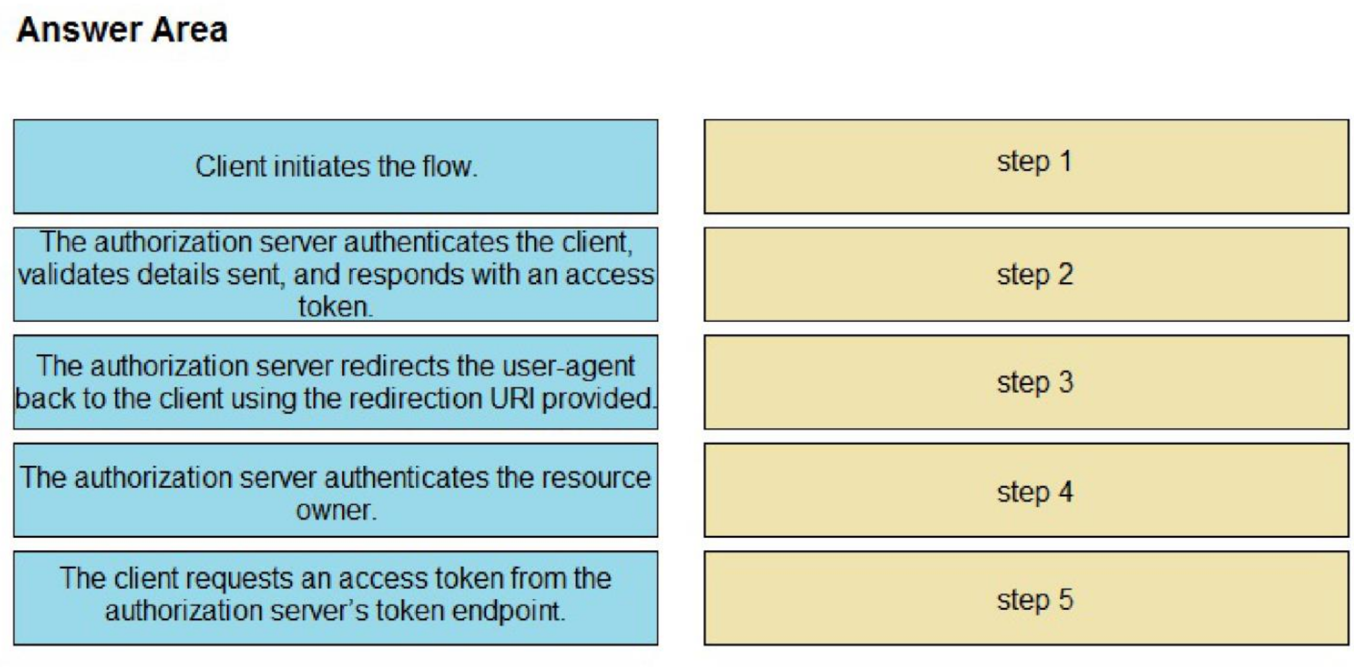
Cisco DevNet:https://developer.cisco.com/ **-** Explore Cisco's developer resources to find practical use cases and documentation related to gNMI and its implementation on Cisco platforms.

Question: 21

DRAG DROP -



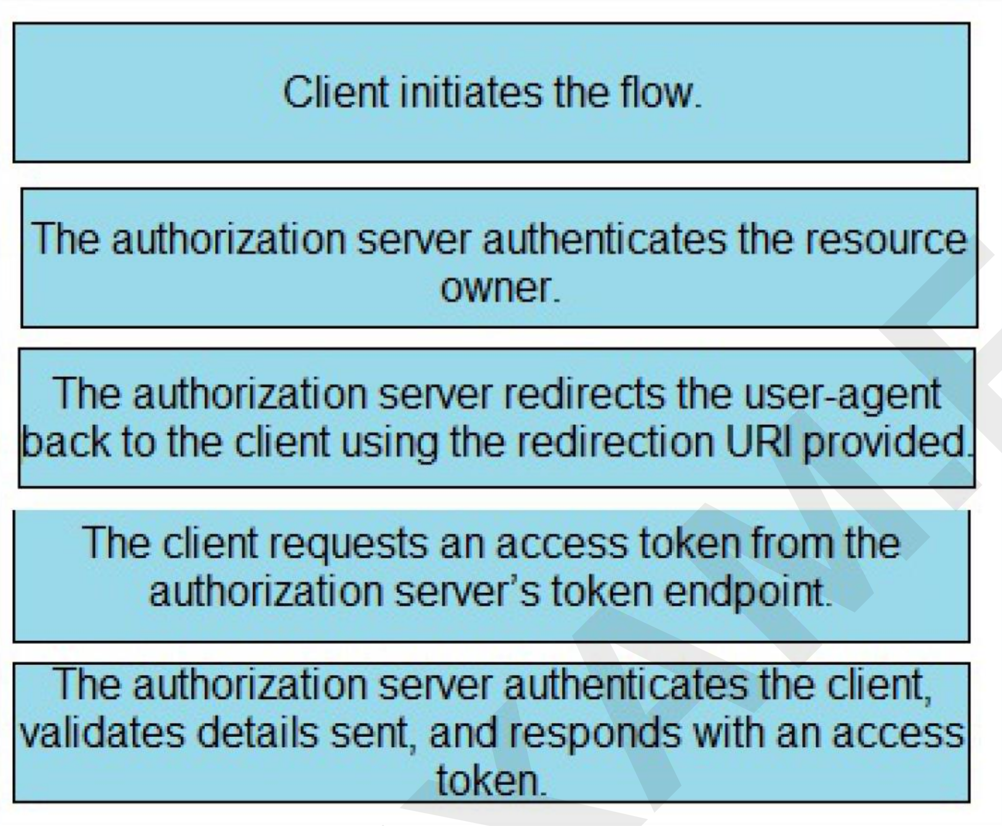
Refer to the exhibit. Drag and drop the steps from the left into the correct order of operation on the right for a successful OAuth2 three-legged authorization code grant flow.
Select and Place:



Answer Area

Client initiates the flow.	step 1
The authorization server authenticates the client, validates details sent, and responds with an access token.	step 2
The authorization server redirects the user-agent back to the client using the redirection URI provided.	step 3
The authorization server authenticates the resource owner.	step 4
The client requests an access token from the authorization server's token endpoint.	step 5

Answer:



Client initiates the flow.

The authorization server authenticates the resource owner.

The authorization server redirects the user-agent back to the client using the redirection URI provided.

The client requests an access token from the authorization server's token endpoint.

The authorization server authenticates the client, validates details sent, and responds with an access token.

Explanation:

1. Client initiates the flow.

2. The authorization server authenticates the resource owner.

3. The authorization server redirects the user-agent back to the client using the redirection URI provided. 4. The client requests an access token from the authorization server's token endpoint.

5. The authorization server authenticates the client, validates details send, and responds with an access token.

Question: 22

A developer plans to create a new bugfix branch to fix a bug that was found on the release branch. Which command completes the task?

A.git checkout -t RELEASE BUGFIX
B.git checkout -b RELEASE BUGFIX
C.git checkout -t BUGFIX RELEASE
D.git checkout -b BUGFIX RELEASE

Answer: D

Explanation:

The correct command is git checkout -b BUGFIX RELEASE. Here's why:

The goal is to create a new branch named BUGFIX starting from the RELEASE branch. The git checkout command is used for switching between branches and creating new ones. The -b option specifically instructs Git to create a new branch. The syntax for creating a new branch and switching to it is git checkout -b <new_branch_name> <starting_point_branch>. Therefore, git checkout -b BUGFIX RELEASE correctly achieves the desired outcome, creating a branch named BUGFIX based on the current state of the RELEASE branch and then switching to it.

Option A, git checkout -t RELEASE BUGFIX, attempts to track a remote branch with a local name, which is not the desired behavior here. Option B, git checkout -b RELEASE BUGFIX, reverses the desired action; it will create a branch RELEASE starting from BUGFIX. Option C, git checkout -t BUGFIX RELEASE, also attempts to track a remote branch, which is incorrect in this scenario.

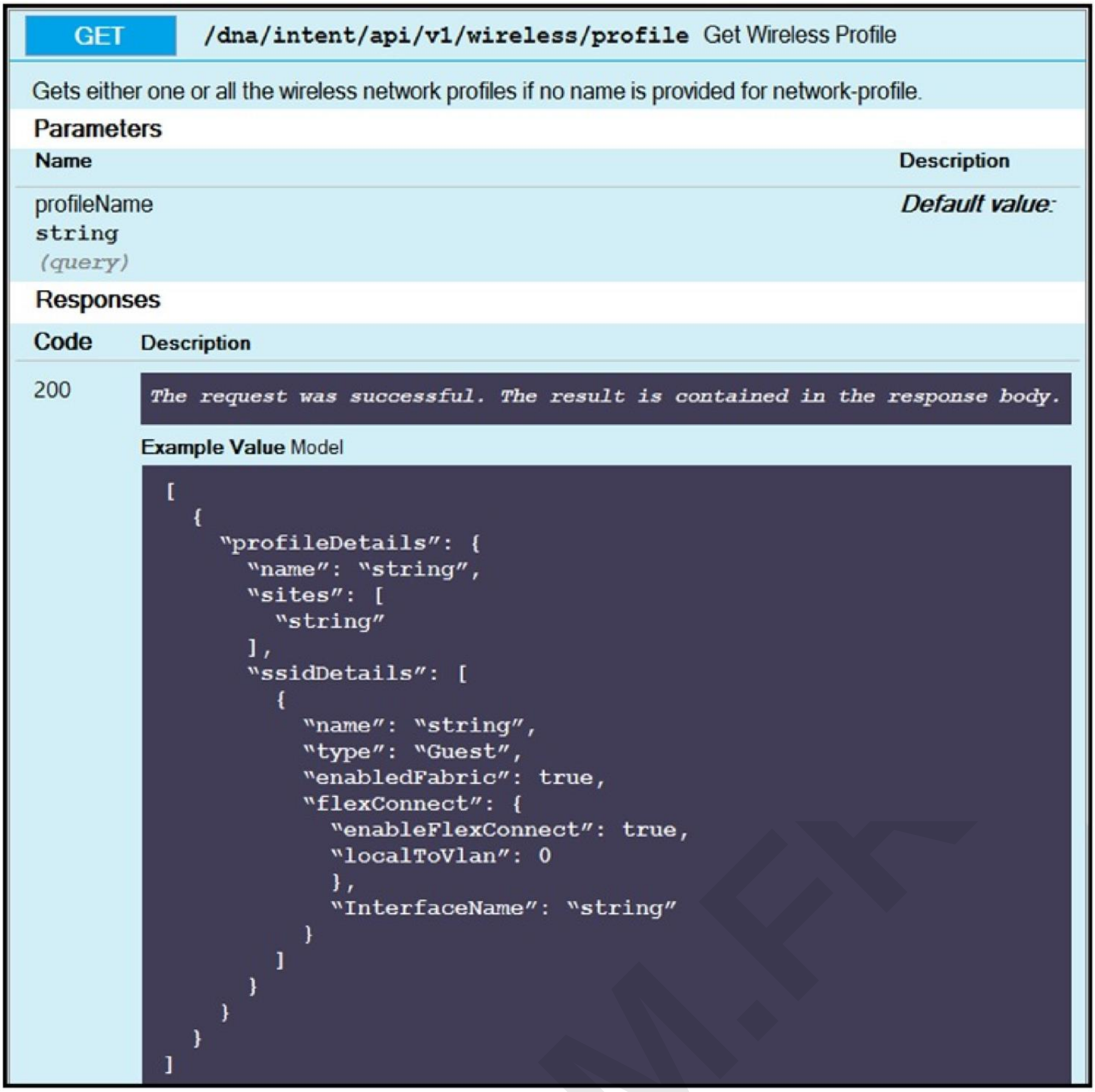
The process of branching is fundamental to version control, enabling developers to work on features, bug fixes, or experiments in isolation before merging them into the main codebase. This facilitates parallel development, reduces conflicts, and provides a structured way to manage changes. Branching is a core component of Git's workflow and is widely used in software development teams.

Here are some helpful resources for further understanding of Git and branching:

1. **Official Git Documentation on Branching:**https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
2. **Atlassian Git Branching Tutorial:**https://www.atlassian.com/git/tutorials/using-branches 3. **GitHub Guides on Git Branching:**https://guides.github.com/introduction/flow/

Question: 23

DRAG DROP -



GET **/dna/intent/api/v1/wireless/profile** Get Wireless Profile

Gets either one or all the wireless network profiles if no name is provided for network-profile.

Parameters

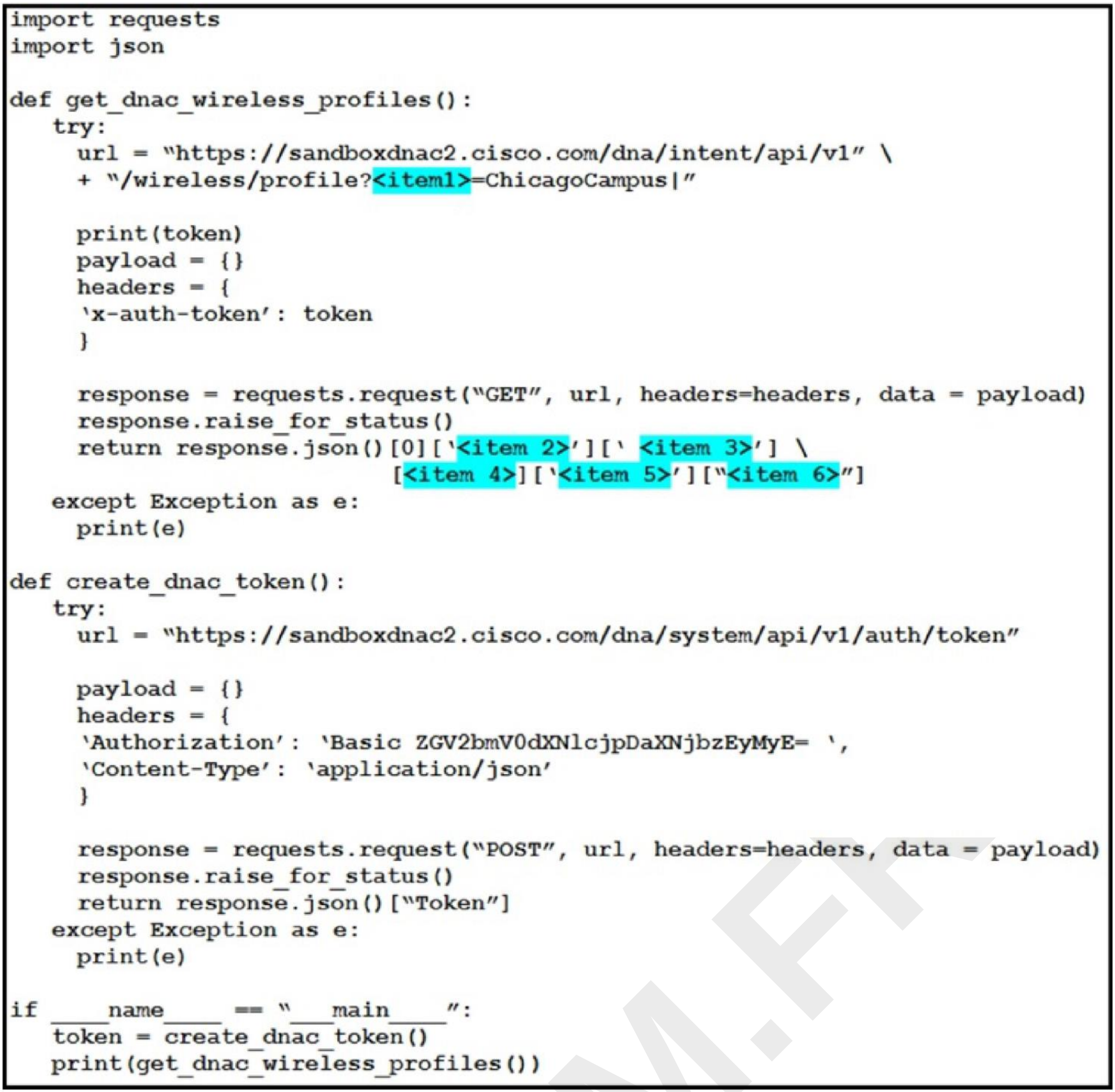
Name	Description
profileName `string` *(query)*	***Default value:***

Responses

Code	Description
200	*The request was successful. The result is contained in the response body.*

Example Value Model

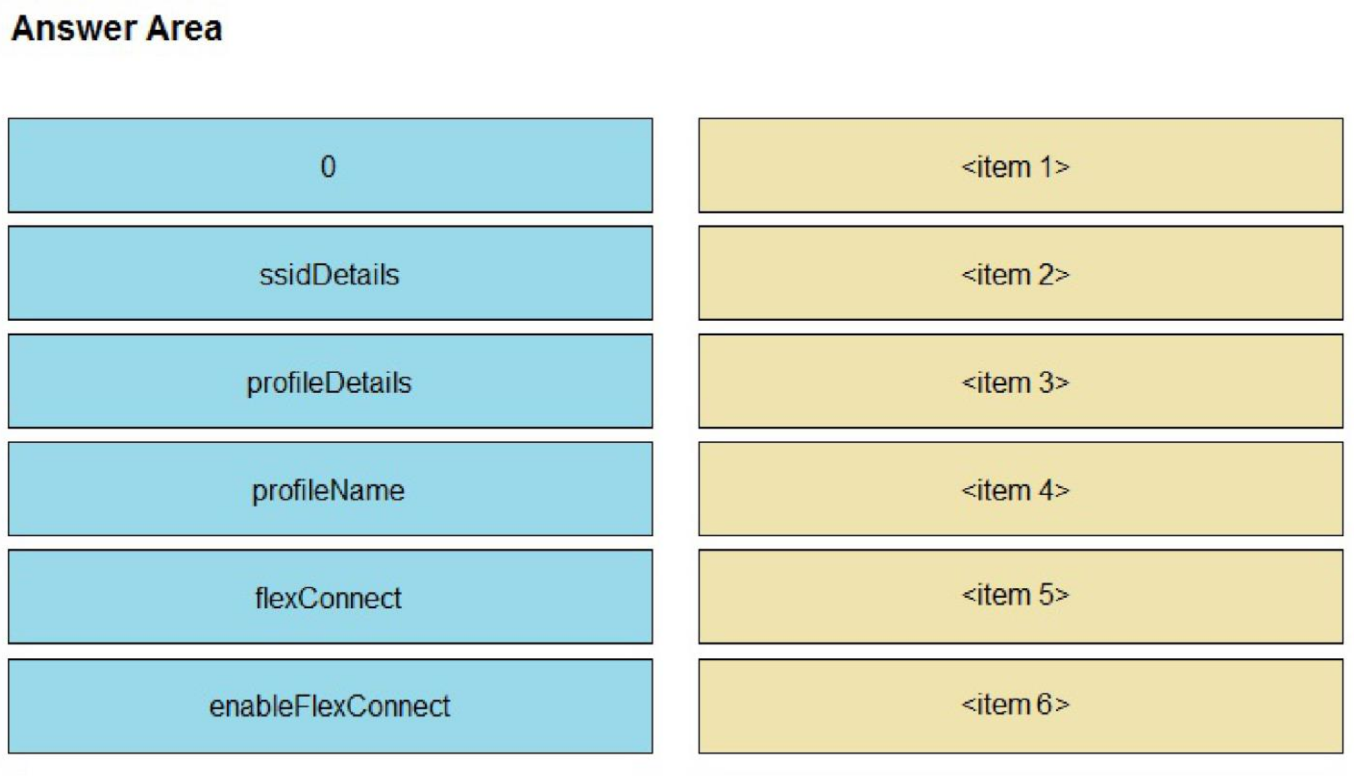
```
[
  {
    “profileDetails”: {
      “name”: “string”,
      “sites”: [
        “string”
      ],
      “ssidDetails”: [
        {
          “name”: “string”,
          “type”: “Guest”,
          “enabledFabric”: true,
          “flexConnect”: {
            “enableFlexConnect”: true,
            “localToVlan”: 0
          },
          “InterfaceName”: “string”
        }
      ]
    }
  }
]
```



```
import requests
import json

def get_dnac_wireless_profiles():
   try:
     url = “https://sandboxdnac2.cisco.com/dna/intent/api/v1” \
     + “/wireless/profile?<item1>=ChicagoCampus|”

     print(token)
     payload = {}
     headers = {
     ‘x-auth-token’: token
     }

     response = requests.request(“GET”, url, headers=headers, data = payload)
     response.raise_for_status()
     return response.json()[0][‘<item 2>’][‘ <item 3>’] \
                             [<item 4>][‘<item 5>’][“<item 6>”]
   except Exception as e:
     print(e)

def create_dnac_token():
   try:
     url = “https://sandboxdnac2.cisco.com/dna/system/api/v1/auth/token”

     payload = {}
     headers = {
     ‘Authorization’: ‘Basic ZGV2bmV0dXNlcjpDaXNjbzEyMyE= ’,
     ‘Content-Type’: ‘application/json’
     }

     response = requests.request(“POST”, url, headers=headers, data = payload)
     response.raise_for_status()
     return response.json()[“Token”]
   except Exception as e:
     print(e)

if ____name____ == “___main____”:
   token = create_dnac_token()
   print(get_dnac_wireless_profiles())
```

Refer to the exhibit. The Python script is supposed to make an API call to Cisco DNA Center querying a wireless profile for the "ChicagoCampus" and then parsing out its enable FlexConnect value. Drag and drop the parts of the Python code from the left onto the item numbers on the right that match the missing sections in the exhibit.

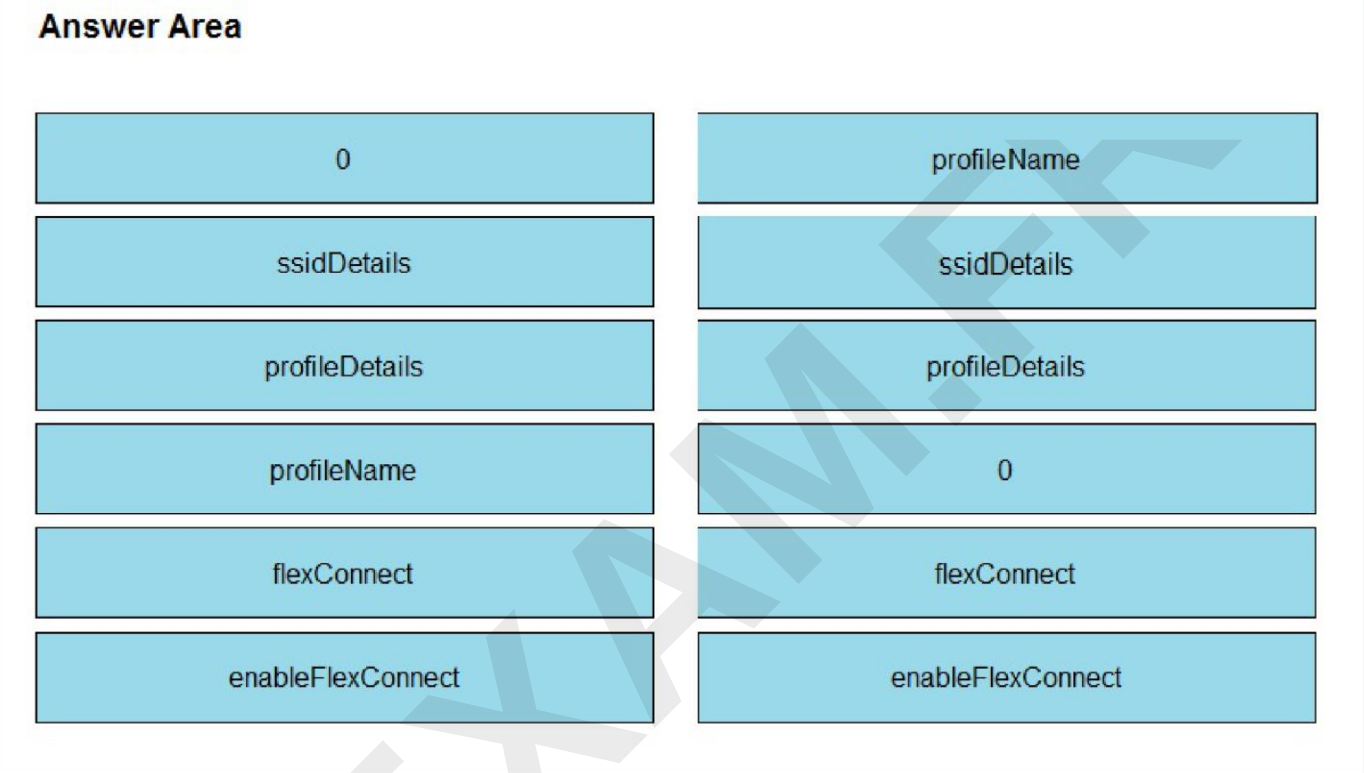
Select and Place:



Answer Area

0	<item 1>
ssidDetails	<item 2>
profileDetails	<item 3>
profileName	<item 4>
flexConnect	<item 5>
enableFlexConnect	<item 6>

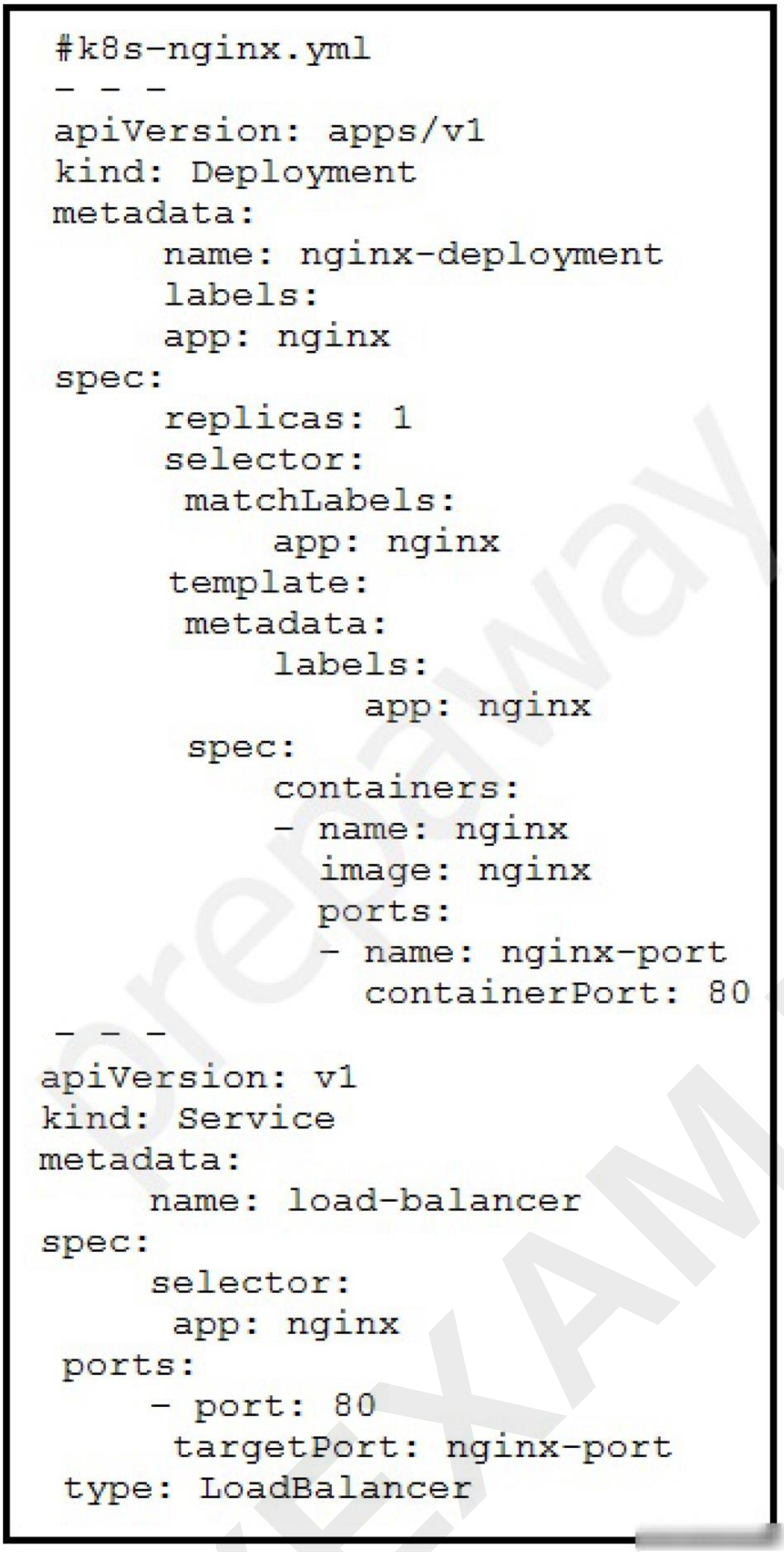
Answer:



Answer Area

0	profileName
ssidDetails	ssidDetails
profileDetails	profileDetails
profileName	0
flexConnect	flexConnect
enableFlexConnect	enableFlexConnect

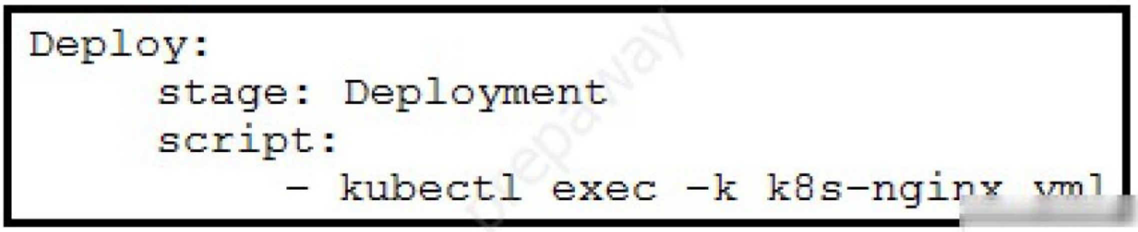
Question: 24



```
#k8s-nginx.yml
- - -
apiVersion: apps/v1
kind: Deployment
metadata:
     name: nginx-deployment
     labels:
     app: nginx
spec:
     replicas: 1
     selector:
      matchLabels:
          app: nginx
      template:
      metadata:
          labels:
              app: nginx
      spec:
          containers:
          - name: nginx
            image: nginx
            ports:
            - name: nginx-port
              containerPort: 80
- - -
apiVersion: v1
kind: Service
metadata:
     name: load-balancer
spec:
     selector:
      app: nginx
 ports:
     - port: 80
      targetPort: nginx-port
 type: LoadBalancer
```

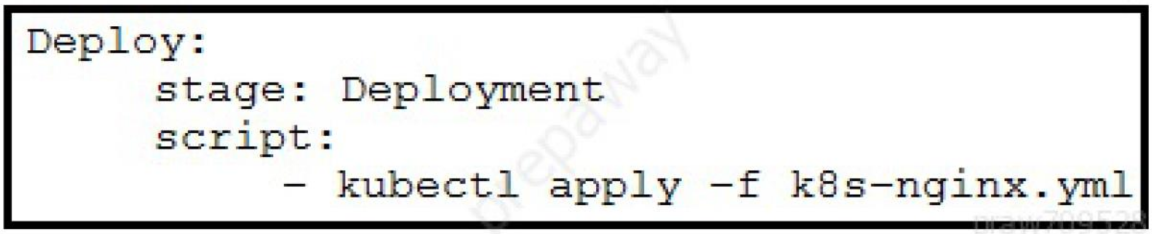
Refer to the exhibit. The presented application consists of a Nginx container and a load balancer service. Which GitLab CI/CD configuration implements the Kubernetes deployment?

A.



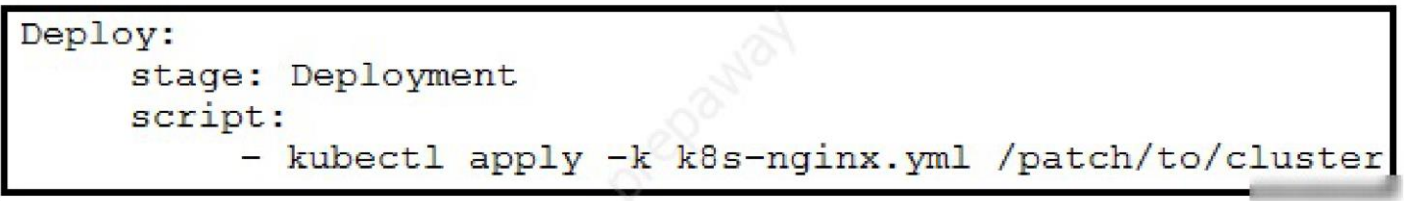
```
Deploy:
     stage: Deployment
     script:
          - kubectl exec -k k8s-nginx.yml
```

B.



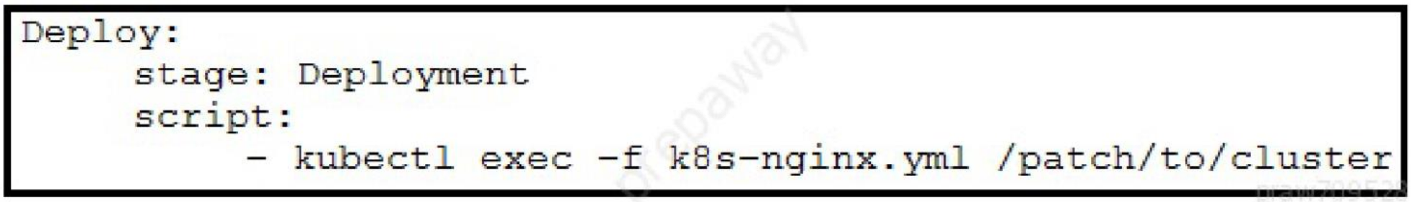
```
Deploy:
     stage: Deployment
     script:
          - kubectl apply -f k8s-nginx.yml
```

C.



```
Deploy:
     stage: Deployment
     script:
          - kubectl apply -k k8s-nginx.yml /patch/to/cluster
```

D.



```
Deploy:
     stage: Deployment
     script:
          - kubectl exec -f k8s-nginx.yml /patch/to/cluster
```

Answer: B

Explanation:

Reference:

https://kubernetes.io/docs/reference/kubectl/kubectl/

Question: 25

An application has these characteristics:
* provide one service or function
* distributed database
* API gateway
* central repository for code
* configuration database
* uses session management
Which two design approaches contribute to the scalability of the application? (Choose two.)

A.session management in a stateless architecture

B.modular design iteration

C.distributed computing with tightly coupled components

D.built to scale based on a star topology

E.planned before the first device is deployed

Answer: AB

Explanation:

Here's a justification for why options A and B are the correct choices for enhancing application scalability:

A. Session management in a stateless architecture: Statelessness is a cornerstone of scalable systems. When session information isn't tied to a specific server, requests can be routed to any available instance. This eliminates bottlenecks caused by single servers managing all session data and allows for horizontal scaling (adding more server instances). Session management, typically implemented through techniques like tokens or distributed caches, then manages the user session independent of the server, enabling seamless scaling. This is unlike stateful architectures where the server maintains the session information, hindering scalability by making load balancing difficult.

B. Modular design iteration: Modular design, where an application is broken into independent, smaller components or modules, promotes scalability because individual modules can be scaled independently. This allows for scaling only the necessary parts of the application when demand changes, rather than scaling the entire application. This approach also promotes easier development, debugging, and code reuse. It promotes iterative development where new features can be added and existing modules can be modified without affecting other areas.

Why the other options are incorrect:

C. Distributed computing with tightly coupled components: Tightly coupled components hinder scalability. If one component fails or needs to scale, it often affects other components, creating a single point of failure and limiting independent scaling. Distributed systems need components to be loosely coupled to improve scalability.

D. Built to scale based on a star topology: Star topologies can create central bottlenecks as all nodes depend on the central hub. While a star topology is relevant to network architecture, it doesn't generally apply to the application architecture as related to scalability.

E. Planned before the first device is deployed: While planning is vital for successful software development, it's not a scalability approach itself. Planning is a necessary prerequisite, not the feature that allows a system to scale.

Authoritative Links:

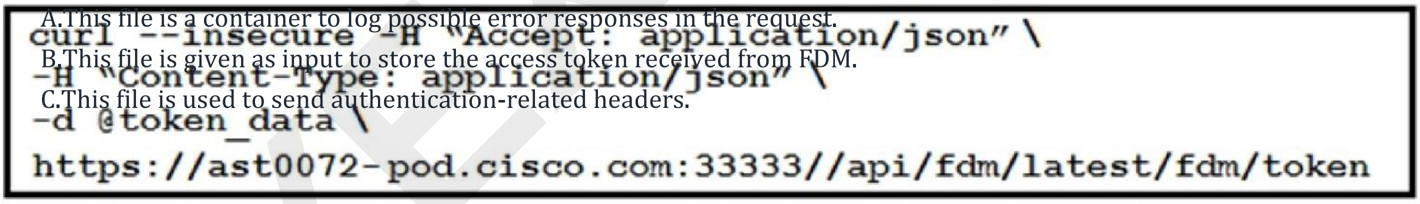
Microservices:https://martinfowler.com/articles/microservices.html - Provides an in-depth look at modular design and its scalability benefits.

Stateless vs Stateful:https://www.nginx.com/blog/stateful-vs-stateless-application-architectures/ -Explains the differences between stateful and stateless systems, focusing on scalability.

Horizontal Scaling:https://aws.amazon.com/what-is/horizontal-scaling/ - Describes horizontal scaling and why it is a key to scalability.

Question: 26

Refer to the exhibit. The cURL POST request creates an OAuth access token for authentication with FDM API requests. What is the purpose of the file `@token_data` that cURL is handling?



```
curl --insecure -H “Accept: application/json” \
-H “Content-Type: application/json” \
-d @token_data \
https://ast0072-pod.cisco.com:33333//api/fdm/latest/fdm/token
```

A.This file is a container to log possible error responses in the request.

B.This file is given as input to store the access token received from FDM.

C.This file is used to send authentication-related headers.

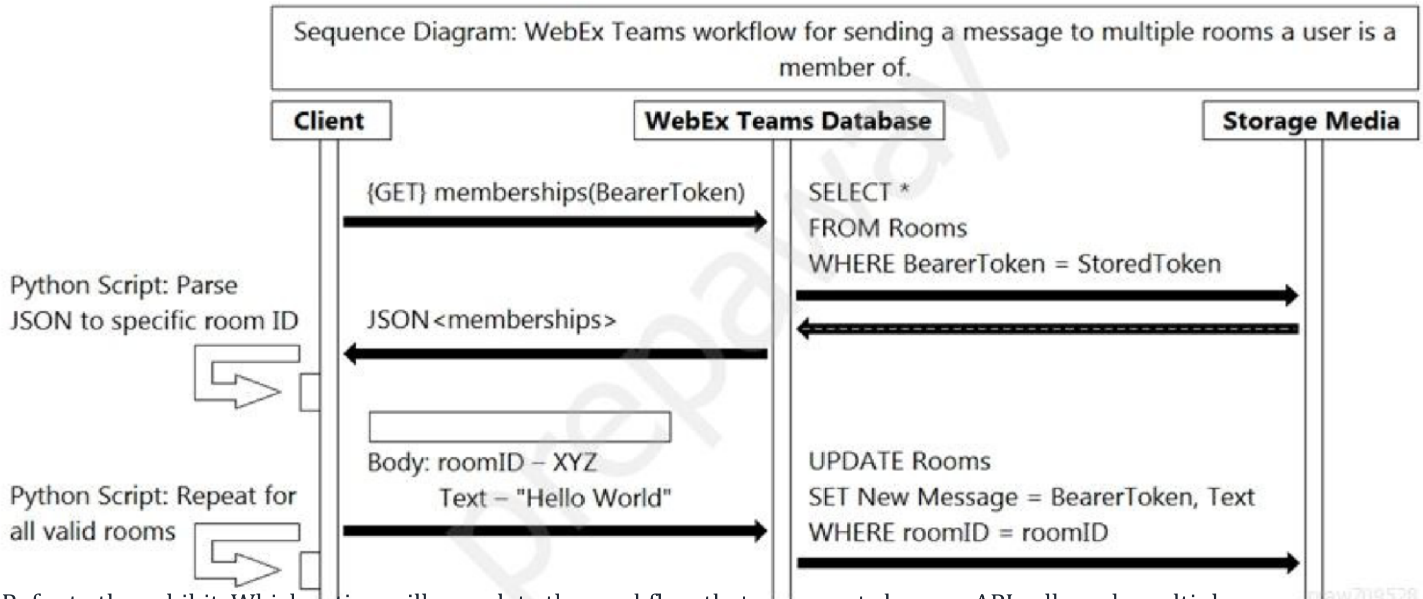
D.This file contains raw data that is needed for token authentication.

Answer: D

Explanation:

This file contains raw data that is needed for token authentication.

Question: 27


Sequence Diagram: WebEx Teams workflow for sending a message to multiple rooms a user is a member of.
Client
WebEx Teams Database
Storage Media
{GET} memberships(BearerToken)
SELECT *
FROM Rooms
WHERE BearerToken = StoredToken
Python Script: Parse
JSON to specific room ID
JSON<memberships>
Body: roomID – XYZ
Text – "Hello World"
UPDATE Rooms
SET New Message = BearerToken, Text
WHERE roomID = roomID
Python Script: Repeat for
all valid rooms

Refer to the exhibit. Which action will complete the workflow that represents how an API call sends multiple messages?

A. PUT messages(roomID)

B. PUT messages(BearerToken)

C. POST messages(roomID)

D. POST messages(BearerToken)

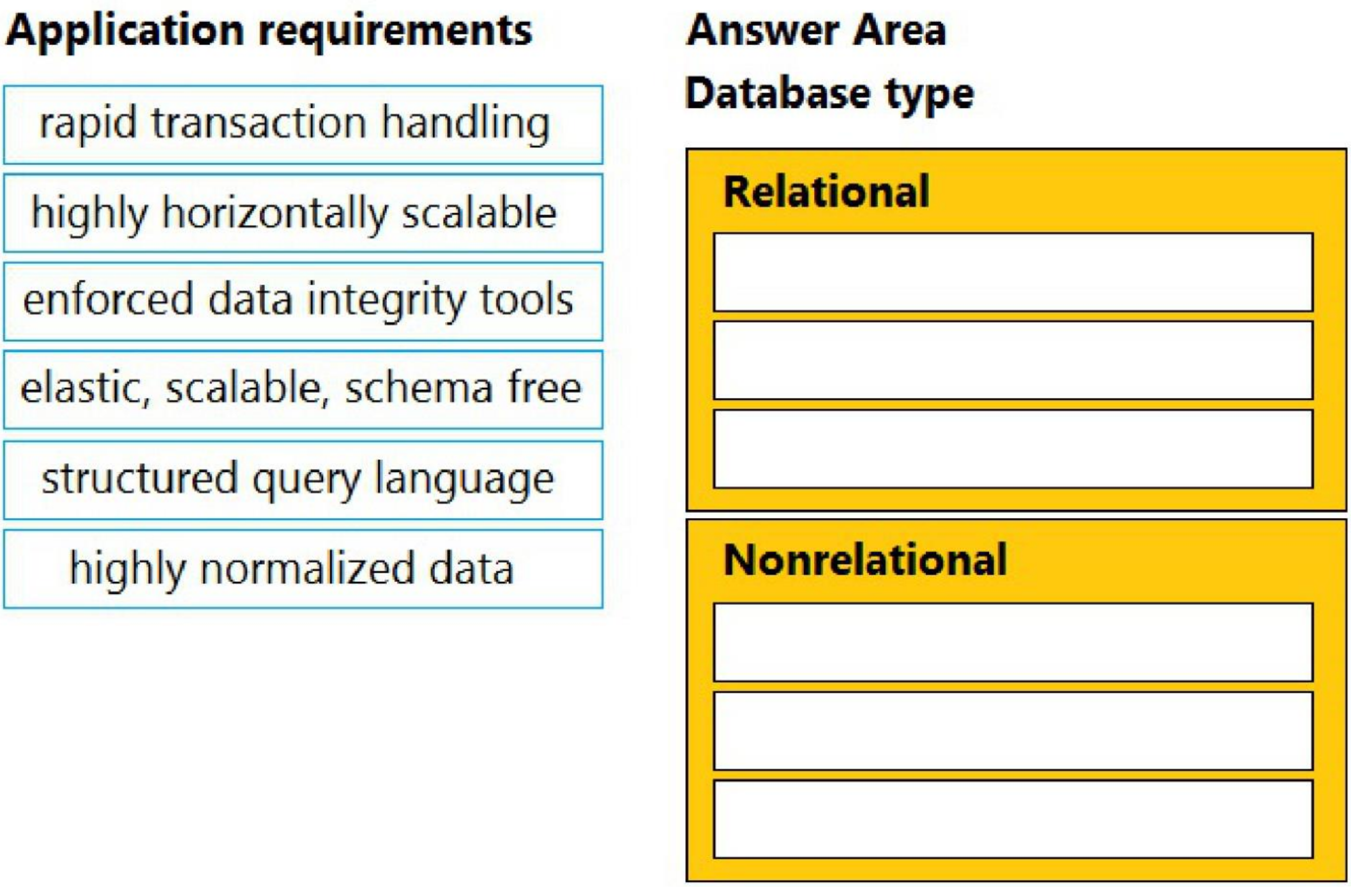
Answer: D

Explanation:

POST messages(BearerToken).

Question: 28

DRAG DROP -
Drag and drop the application requirement on the left onto the database type that should be selected for the requirement on the right.
Select and Place:



Application requirements

- rapid transaction handling
- highly horizontally scalable
- enforced data integrity tools
- elastic, scalable, schema free
- structured query language
- highly normalized data

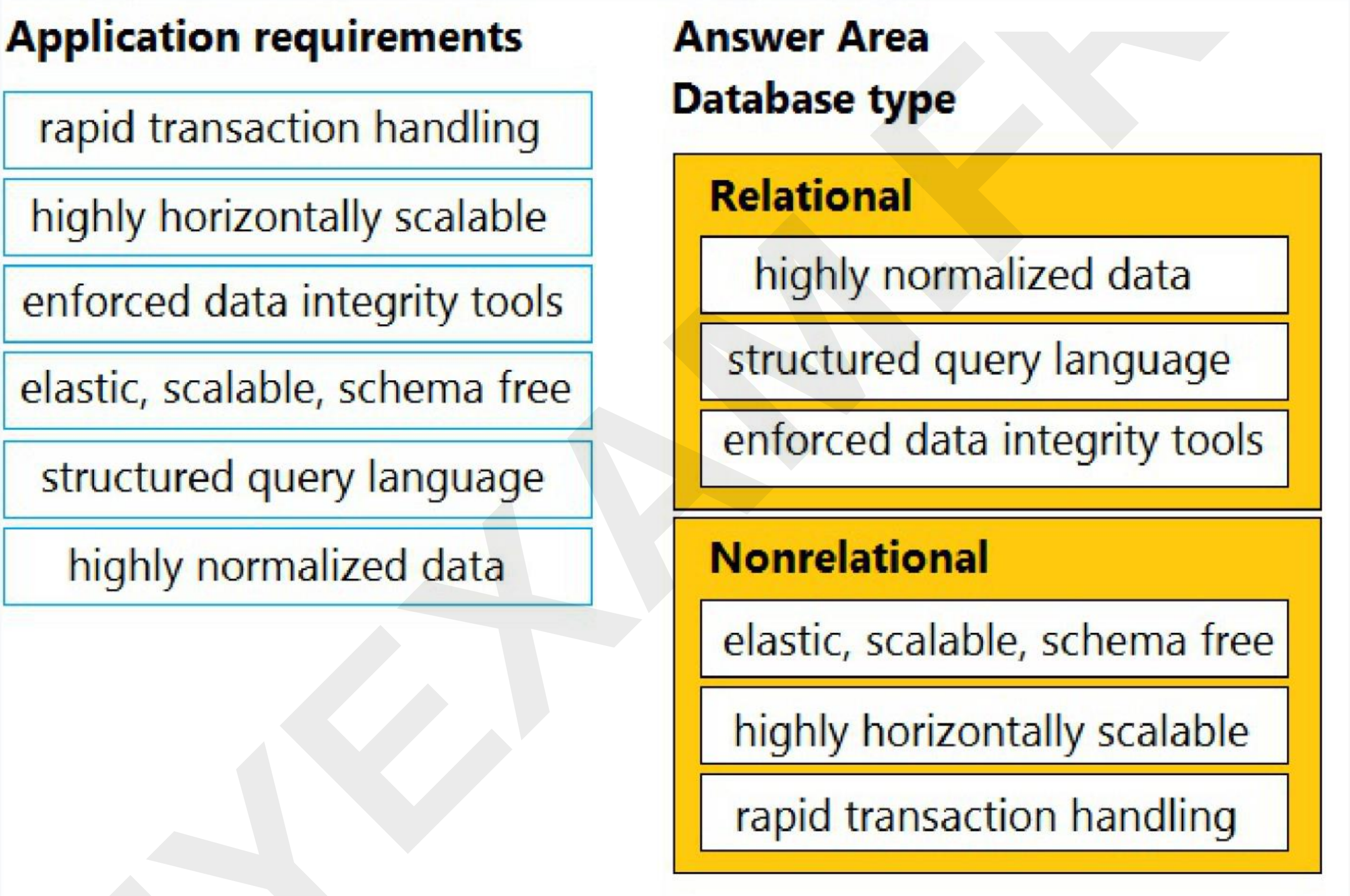
Answer Area

Database type

Relational

Nonrelational

Answer:



Application requirements

- rapid transaction handling
- highly horizontally scalable
- enforced data integrity tools
- elastic, scalable, schema free
- structured query language
- highly normalized data

Answer Area

Database type

Relational

- highly normalized data
- structured query language
- enforced data integrity tools

Nonrelational

- elastic, scalable, schema free
- highly horizontally scalable
- rapid transaction handling

Explanation:

Relational:

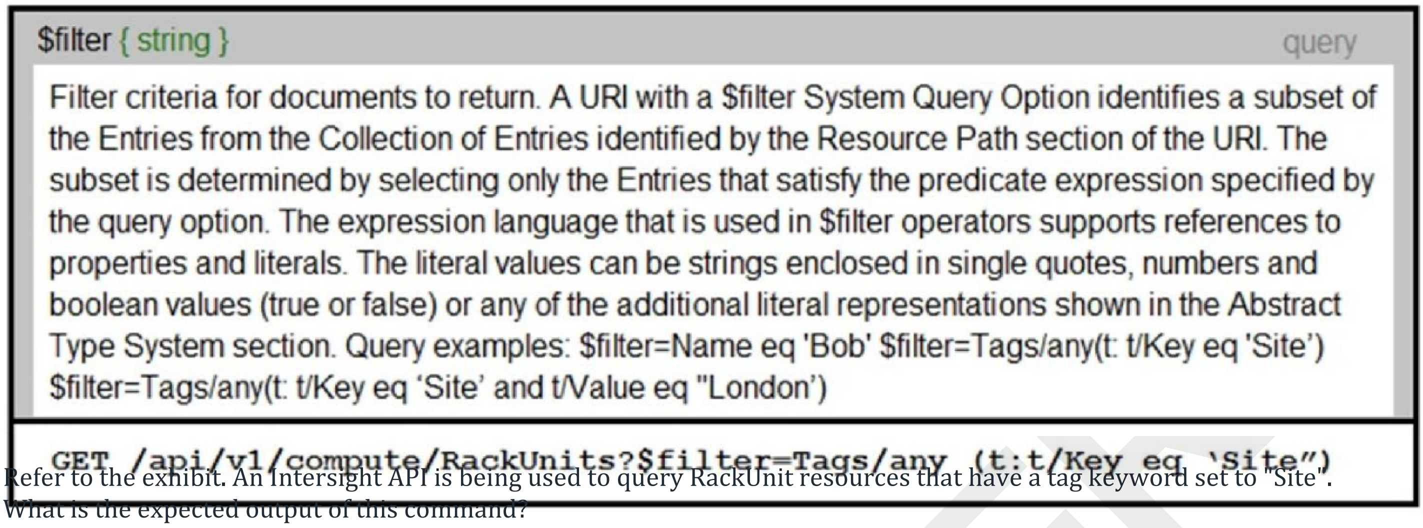
- Highly normalized data.

- Structured query language.
- Enforced data integrity tools.

Nonrelational:
- elastic, Scalable, Schema free.

- highly horizontally Scalable.
- rapid transaction handling.

Question: 29



$filter { string } query

Filter criteria for documents to return. A URI with a $filter System Query Option identifies a subset of the Entries from the Collection of Entries identified by the Resource Path section of the URI. The subset is determined by selecting only the Entries that satisfy the predicate expression specified by the query option. The expression language that is used in $filter operators supports references to properties and literals. The literal values can be strings enclosed in single quotes, numbers and boolean values (true or false) or any of the additional literal representations shown in the Abstract Type System section. Query examples: $filter=Name eq 'Bob' $filter=Tags/any(t: t/Key eq 'Site') $filter=Tags/any(t: t/Key eq ‘Site’ and t/Value eq "London')

```
GET /api/v1/compute/RackUnits?$filter=Tags/any (t:t/Key eq ‘Site”)
```

Refer to the exhibit. An Intersight API is being used to query RackUnit resources that have a tag keyword set to "Site". What is the expected output of this command?

A.list of all resources that have a tag with the keyword "Site" B.error message because the Value field was not specified
C.error message because the tag filter should be lowercase D.list of all sites that contain RackUnit tagged compute resources

Answer: A

Explanation:

list of all resources that have a tag with the keyword "Site".

Question: 30 What is the function of dependency management?

A.separating code into modules that execute independently
B.utilizing a single programming language/framework for each code project
C.automating the identification and resolution of code dependencies D.managing and enforcing unique software version names or numbers

Answer: C

Explanation:

Dependency management, in the context of software development, focuses on handling the external libraries, packages, or other modules that a project relies upon. Option C, "automating the identification and resolution of code dependencies," accurately describes this core function. Software projects rarely exist in isolation; they often need to leverage functionality provided by external resources. Dependency management tools track these dependencies, ensuring that the correct versions are available and compatible with the project.

These tools also automatically download, install, and update the required dependencies, simplifying the development process. This automation prevents manual, error-prone dependency handling, reducing the risk of conflicts and build failures.

Options A, B, and D, while related to software development, do not define the primary role of dependency management. Option A, "separating code into modules that execute independently," describes modular design or microservices, not dependency management. Option B, "utilizing a single programming language/framework for each code project," is a guideline for development practices but is not directly related to dependency management. Option D, "managing and enforcing unique software version names or numbers," is related to version control, which is a separate concept although intertwined with dependency management. Dependency management is vital for maintainability, scalability, and reproducibility of software projects, especially in complex environments and agile development processes. The goal is to ensure that the right resources are available at the correct versions without developers having to manually handle each dependency.

Authoritative Links:

Wikipedia on Dependency Management:https://en.wikipedia.org/wiki/Dependency_management
Atlassian's Explanation on Dependency Management:https://www.atlassian.com/continuous-delivery/principles/dependency-management
Cloud Native Computing Foundation on Dependency Management:
https://landscape.cncf.io/category=dependency-management

Question: 31

Which two types of organization are subject to GDPR? (Choose two.)

A.only organizations that operate outside the EU

B.any organization that offers goods or services to customers in the EU

C.only organizations that have offices in countries that are part of the EU

D.any organization that operates within the EU

E.only organizations that physically reside in the EU

Answer: BD

Explanation:

The General Data Protection Regulation (GDPR) applies to organizations based on their processing of personal data, not solely on their physical location. Specifically, GDPR impacts two key types of organizations. Option B is correct because any organization offering goods or services to individuals within the European Union (EU) is subject to GDPR, regardless of where the organization is based. This "targeting" of the EU market triggers GDPR compliance requirements. Option D is also correct because any organization that operates within the EU, including those that process personal data of EU citizens, also falls under GDPR's jurisdiction. This means if an organization has an established physical presence or conducts its core business activities in the EU, it is bound by GDPR. Therefore, options A, C, and E are incorrect as they place geographical limitations or ownership factors as the sole determinant of GDPR compliance, while GDPR focuses on where the personal data is handled and who it concerns. The fundamental aspect of GDPR is the protection of personal data of EU citizens, regardless of where the processing occurs. This territorial scope ensures that EU citizens' data rights are respected across global interactions.

Authoritative Links for Further Research:

Official GDPR Text:https://eur-lex.europa.eu/eli/reg/2016/679/oj
Information Commissioner's Office (ICO) - UK:https://ico.org.uk/for-organisations/guide-to-the-general-data-protection-regulation-gdpr/
European Data Protection Board (EDPB):https://edpb.europa.eu/our-work-tools/general-guidance/gdpr-territorial-scope_en

Question: 32

What is a benefit of continuous testing?

A.increases the number of bugs found in production
B.enables parallel testing
C.removes the requirement for test environments
D.decreases the frequency of code check-ins

Answer: B

Explanation:

The correct answer is **B. enables parallel testing.** Continuous testing is a core practice within DevOps and continuous integration/continuous delivery (CI/CD) pipelines. Its primary goal is to provide rapid feedback on code changes by automating tests at every stage of the development lifecycle. Parallel testing, a direct benefit of continuous testing, allows multiple tests to run concurrently, drastically reducing the overall testing time. This accelerated feedback loop is essential for maintaining agility and speed in modern application development. By running tests in parallel, teams can quickly identify and fix issues, preventing them from escalating into more significant problems later in the process. This is in stark contrast to traditional approaches where testing is often a serial bottleneck, slowing down releases. Options A, C, and D are incorrect. Continuous testing aims to decrease, not increase, bugs found in production. It still requires test environments, and it encourages, not discourages, frequent code check-ins as part of the CI/CD workflow. In essence, parallel testing is a cornerstone of continuous testing's efficiency, allowing for faster, more reliable, and higher-quality software delivery.

Authoritative Links:

Continuous Testing:https://www.guru99.com/continuous-testing.html
Parallel Testing:https://testsigma.com/blog/parallel-testing/
DevOps Principles:https://aws.amazon.com/devops/what-is-devops/

Question: 33

What is a characteristic of a monolithic architecture?

A.It is an application with multiple independent parts.
B.New capabilities are deployed by restarting a component of the application.
C.A service failure can bring down the whole application.

D.The components are platform-agnostic.

Answer: C

Explanation:

The correct answer is **C. A service failure can bring down the whole application.** This is a defining characteristic of a monolithic architecture. Monolithic applications are structured as a single, indivisible unit. All functionalities and components are tightly coupled and deployed together. Because of this tight coupling, if one part of the application fails, it can cascade and cause the entire application to fail.

Options A, B, and D describe aspects opposite of a monolithic architecture. Option A refers to a distributed or microservices architecture where components are independent. Option B describes component-level restarts, which is not often part of a monolithic setup; entire application redeployments are the norm after code changes. Option D, platform-agnostic components, is also not generally a characteristic of monolithic apps as they are typically built for a specific environment. The interdependence of components in a monolithic architecture contributes to difficulties in scaling, agility, and fault isolation, often leading to a single point of failure.

In contrast, modern architectural styles like microservices strive to be loosely coupled, allowing for more resilient applications. With microservices, if one service goes down, it shouldn't bring the entire application offline, as other parts can still function independently. This is a key difference when comparing monolithic vs microservices. A single deployment unit makes scaling complex, as you need to scale the entire application, even if only a small part is under heavy load. These drawbacks have led to the move toward distributed systems and microservices.

For further research, you can explore the following resources:

Microsoft Azure Documentation on Microservices Architecture:https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices
Amazon Web Services Documentation on Monolithic vs Microservices:
https://aws.amazon.com/compare/the-difference-between-monolithic-and-microservices/
Cloudflare Blog on Monoliths:https://www.cloudflare.com/learning/serverless/glossary/monolithic-architecture/

Question: 34

What are two principles according to the build, release, run principle of the twelve-factor app methodology? (Choose two.)

A.Code changes are able to be made at runtime.

B.Separation between the build, release, and run phases.

C.Releases should have a unique identifier.

D.Existing releases are able to be mutated after creation.

E.Release stage is responsible for compilation of assets and binaries.

Answer: BC

Explanation:

The twelve-factor app methodology, designed for building scalable and maintainable software-as-a-service applications, emphasizes a clear separation of the build, release, and run stages. This principle ensures that each stage has a specific responsibility and is treated as a distinct entity. Option B, stating "Separation between the build, release, and run phases," directly reflects this core tenet. The build stage transforms the code repository into an executable bundle, the release stage combines the build output with configuration to create a specific release, and the run stage executes this release in the target environment. Option C, "Releases should have a unique identifier," is also correct because it ensures traceability and reproducibility. Each release represents a specific snapshot of the application with a distinct set of configuration and built code, allowing for easy rollback and auditability. Unique identifiers make each release immutable, making debugging and version management straightforward.

Options A, D and E are incorrect. Option A, which suggests “Code changes are able to be made at runtime”contradicts the core philosophy of immutable releases and well-defined pipelines. Option D, "Existing releases are able to be mutated after creation," violates the principle of immutability, making debugging very difficult. Finally, Option E, “Release stage is responsible for compilation of assets and binaries”, is incorrect since the compilation of assets and binaries happens in the build stage, not the release stage. The release phase’s main purpose is to tie the built code to the specific environment configurations.

For further understanding, refer to resources such as:

The official 12-factor app documentation: https://12factor.net/
Relevant cloud computing resources on DevOps and application deployment methodologies.

Question: 35

What is a well-defined concept for GDPR compliance?

A.Data controllers must confirm to data subjects as to whether, where, and why personal data is being processed.

B.Personal data that was collected before the compliance standards were set do not need to be protected.

C.Compliance standards apply to organizations that have a physical presence in Europe.

D.Records that are relevant to an existing contract agreement can be retained as long as the contract is in effect.

Answer: A

Explanation:

Justification for Answer A:

Answer A, "Data controllers must confirm to data subjects as to whether, where, and why personal data is being processed," is the most accurate reflection of a core principle of the General Data Protection Regulation (GDPR). GDPR fundamentally emphasizes transparency and control for individuals (data subjects) over their personal data. This principle is enshrined in articles 12-14 of the regulation which detail the information that data controllers must provide. These articles require controllers to inform subjects about the nature of data being processed, the purpose, the legal basis for processing, and retention periods. This proactive disclosure ensures individuals are aware of how their data is being used and can exercise their rights.

The other options are inaccurate:

Option B is incorrect because GDPR applies to personal data regardless of when it was collected. There is no grandfather clause exempting data collected before the regulation's effective date.

Option C is a misleading oversimplification. GDPR applies to any organization processing the personal data of individuals within the EU, regardless of the organization's physical location. This includes entities outside of Europe offering goods or services to EU citizens. This is the principle of extraterritoriality.

Option D is also incorrect because GDPR stipulates that data should only be kept as long as necessary for the purpose for which it was collected. While data might be needed for contract fulfillment, there are retention limitations that go beyond the mere validity of the contract and should be deleted according to set periods.

Authoritative Links for further research:

Official GDPR Text:https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679 **Information Commissioner's Office (UK) - Guide to GDPR:**https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/
European Data Protection Board (EDPB):https://edpb.europa.eu/

In summary, GDPR's focus on transparency and individual control makes Option A the correct answer, emphasizing the importance of informing data subjects about data processing activities.

Question: 36

Given an application that implements a basic search function as well as a video upload function, which two load-balancing approaches optimize the application's user experience? (Choose two.)

A.Video upload requests should be routed to the endpoint using an intermediate hop.

B.Search requests should be routed to the endpoint with lowest round-trip latency.

C.Video upload requests should be routed to the endpoint with lowest round-trip latency.

D.Video upload requests should be routed to the endpoint with highest data throughput.

E.Search requests should be routed to the endpoint with highest data throughput.

Answer: BD

Explanation:

Here's a detailed justification for why options B and D are the optimal load balancing approaches, and why A, C, and E are not ideal, given the scenario of a search and video upload application:

Why B is correct: Routing search requests to the endpoint with the lowest round-trip latency directly enhances user experience. Search operations are typically latency-sensitive; users expect near-instantaneous results. Minimizing latency means faster response times, leading to a more responsive and satisfying search experience. This aligns with the principle of optimizing for interactive tasks. Sources such as cloud load balancing documentation from AWS https://aws.amazon.com/elasticloadbalancing/ or Azure https://azure.microsoft.com/en-us/products/load-balancer emphasize routing based on latency for latency-critical applications.

Why D is correct: Video uploads are data-intensive, requiring high bandwidth to complete quickly. Routing upload requests to endpoints with the highest data throughput ensures that uploads can complete rapidly, minimizing wait times. This improves user experience, as users can avoid extended periods of waiting for their uploads to finish. Throughput optimization is a crucial aspect of network and application performance, especially for operations with large data volume. Resources on network performance optimization like Google Cloud's guide to optimizing network performance https://cloud.google.com/network-connectivity/docs/network-performance demonstrate the importance of matching tasks to throughput capabilities.

Why A is incorrect: Introducing an intermediate hop for video uploads adds complexity and latency, worsening user experience. Additional hops introduce potential points of failure and delays, which are undesirable for large file uploads. It's often best to directly route uploads to the appropriate endpoint to minimize the path.

Why C is incorrect: While minimizing latency is generally good, it's not the primary focus for video uploads. Uploads are more dependent on bandwidth and throughput; latency optimization alone may not be effective, especially if the endpoint is bandwidth-constrained. Optimizing for throughput is more critical.

Why E is incorrect: Routing search requests to the highest throughput endpoint is less optimal than routing them based on latency. Search performance is dictated primarily by how quickly responses can be received, not necessarily by the total throughput that an endpoint is capable of handling. User perception for search is more tied to latency than the potential throughput of the backend.

Question: 37

A developer is working on a bug fix. The existing branch named 'bugfix05328' needs to be merged with the current working primary branch named
'prim404880077'. All changes must be integrated into a single commit instead of preserving them as individual commits.
Which git command must be used?

A.git checkout " "squash bugfix05328
B.git merge " "squash bugfix05328
C.git rebase " "merge bugfix05328
D.git push " "rebase bugfix05328

Answer: B

Explanation:

The correct Git command is git merge --squash bugfix05328. Here's why:

The scenario requires merging the changes from the bugfix05328 branch into the prim404880077 branch, but instead of creating a merge commit that preserves the history of the bugfix05328 branch as separate commits, it should integrate the changes as a single commit. This is accomplished using the --squash option with the git merge command.

git merge combines changes from one branch into another. By adding the --squash flag, the changes from the specified branch (in this case, bugfix05328) are incorporated into the working directory of the current branch (prim404880077), but they are not committed yet. This means that the user has to stage and commit these changes manually, which creates the desired single commit. The merge will not create a merge commit, and the history will be linear. git checkout is not a merge command. git rebase could accomplish this, but it is more complex for this task, and not the best practice to rebase shared branch. git push is a command to push to remote and does not merge.

Therefore, git merge --squash bugfix05328 is the optimal and recommended method for this scenario.

Authoritative Links:

git merge:https://git-scm.com/docs/git-merge
git merge --squash:https://git-scm.com/docs/git-merge#Documentation/git-merge.txt---squash
git rebase:https://git-scm.com/docs/git-rebase

Question: 38

[1]
needs to be removed so that the commit is performed and branches merged. Which git command must be used to unstage the file?
[1
]
[1
]
[1
]
[1
]

Answer: C

Explanation:

Correction in question:

A developer is working in a branch to develop a new feature named 'newfeat404880077'. A file named 'devcoreg13642911.jpg' has accidentally been staged. This needs to be removed so that the commit is performed and branches merged. Which git command must be used to unstage the file?

A. git delete HEAD devcoreg13642911.jpg

B. git remove HEAD devcoreg13642911.jpg

C. git reset HEAD devcoreg13642911.jpg

D. git revert HEAD devcoreg13642911.jpg

Correct answer C:git reset HEAD devcoreg13642911.jpg.

Question: 39

A developer has completed the implementation of a REST API, but when it is executed, it returns a 401 error message. What must be done on the API to resolve the issue?

A.Access permission to the resource must be granted, before the request.
B.Configure new valid credentials.
C.The requested API endpoint does not exist, and the request URL must be changed.
D.Additional permission must be granted before the request can submitted.

Answer: B

Explanation:

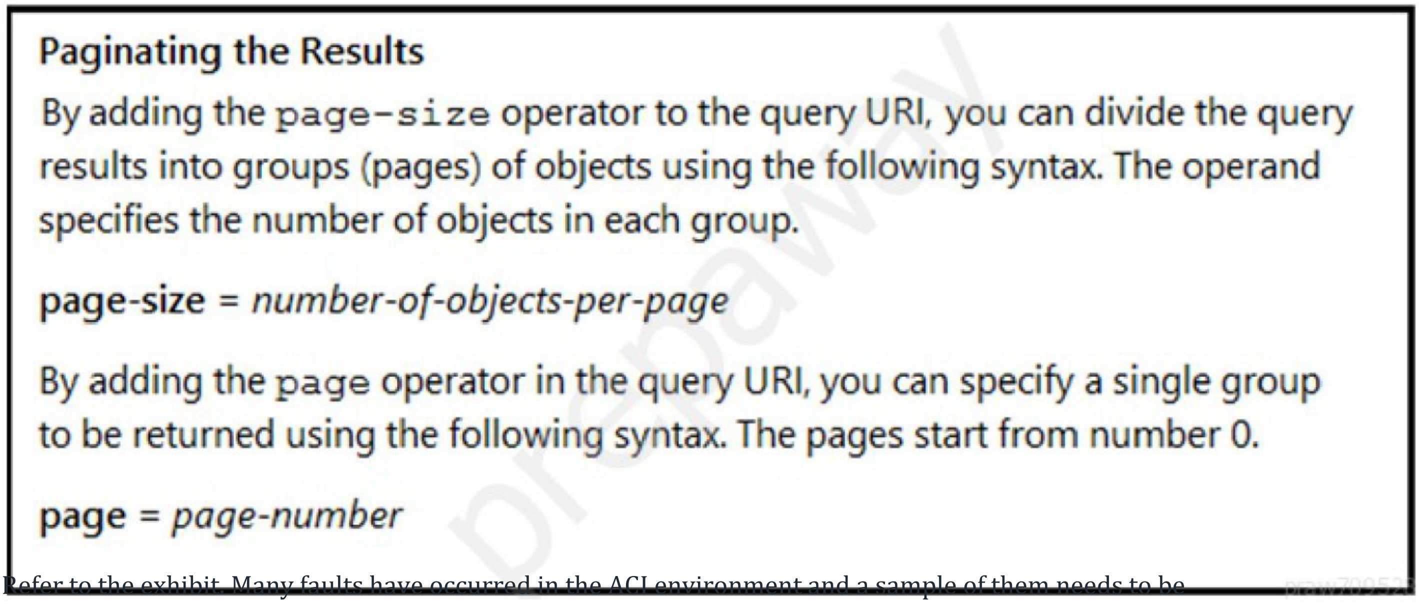
The correct answer is **B. Configure new valid credentials.** A 401 Unauthorized error in the context of a REST API unequivocally indicates an authentication failure. This means the client's provided credentials, or lack thereof, were deemed invalid or insufficient by the server to authorize access to the requested resource. Options A and D, while touching on permissions, don't address the core issue of authentication, which comes before authorization. Option C, regarding endpoint existence, would more likely result in a 404 Not Found error, not a 401. The error explicitly states "Unauthorized," pointing directly to a credential issue, meaning that the server was able to locate the endpoint, but has refused access due to missing or incorrect credentials, such as an invalid token, username/password combination, or API key. Therefore, the immediate action required is to verify and, if necessary, obtain and configure valid credentials, such as a valid API key, bearer token, or user authentication information. This will allow the server to authenticate the request and grant access, if authorization policies also allow it. It doesn't mean permissions are not relevant, but without successful authentication, authorization will never happen. To elaborate, REST API authentication commonly uses mechanisms like API keys, OAuth 2.0, Basic Authentication, or other token-based methods. Ensuring the client is providing valid and correct credentials for the authentication method the API employs is crucial for eliminating 401 errors.

Authoritative Links for further research:

Mozilla Developer Network (MDN) - HTTP Status Codes:https://developer.mozilla.org/en-US/docs/Web/HTTP/Status (Specifically, check the 401 Unauthorized entry)

OAuth 2.0 Documentation:https://oauth.net/2/ (For understanding OAuth 2.0 as a common authentication method)
REST API Security Best Practices: You can find many articles on this topic, but searching for terms like "REST API Authentication best practices" will yield useful results. For instance https://owasp.org/www-project-api-security/ has a good section on authentication.

Question: 40



Paginating the Results

By adding the `page-size` operator to the query URI, you can divide the query results into groups (pages) of objects using the following syntax. The operand specifies the number of objects in each group.

page-size = *number-of-objects-per-page*

By adding the `page` operator in the query URI, you can specify a single group to be returned using the following syntax. The pages start from number 0.

page = *page-number*

Refer to the exhibit. Many faults have occurred in the ACI environment and a sample of them needs to be examined.
Which API call retrieves faults 31 through 45?

A.GET https://apic-ip-address/api/class/faultInfo.json\?order-by=faultInst.severity|desc&page=1&page-size=15

B.GET https://apic-ip-address/api/class/faultInfo.json\?order-by=faultInst.severity|desc&page=2&page-size=15

C.GET https://apic-ip-address/api/class/faultInfo.json\?order-by=faultInst.severity|desc&page-size=30

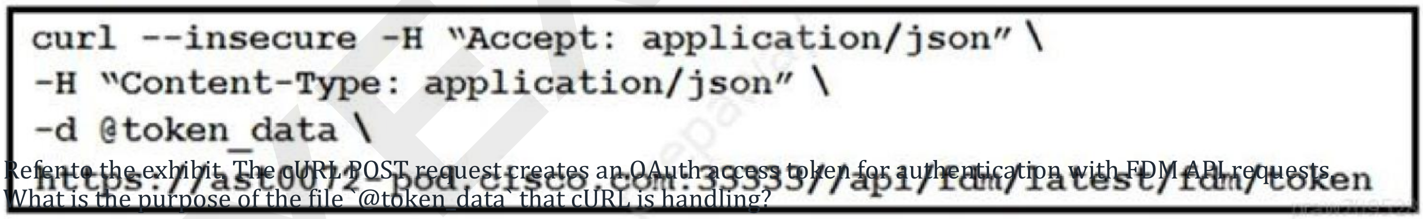
D.GET https://apic-ip-address/api/class/faultInfo.json\?order-by=faultInst.severity|desc&page=2&page-size=30

Answer: B

Explanation:

GET https://apic-ip-address/api/class/faultInfo.json\?order-by=faultInst.severity|desc&page=2&page-size=15.

Question: 41



```
curl --insecure -H "Accept: application/json" \
-H "Content-Type: application/json" \
-d @token_data \
https://ast0072-pod.cisco.com:33333/api/fdm/latest/fdm/token
```

Refer to the exhibit. The cURL POST request creates an OAuth access token for authentication with FDM API requests. What is the purpose of the file `@token_data` that cURL is handling?

A.This file is a container to log possible error responses in the request.

B.This file is given as input to store the access token received from FDM.

C.This file is used to send authentication-related headers.

D.This file contains raw data that is needed for token authentication.

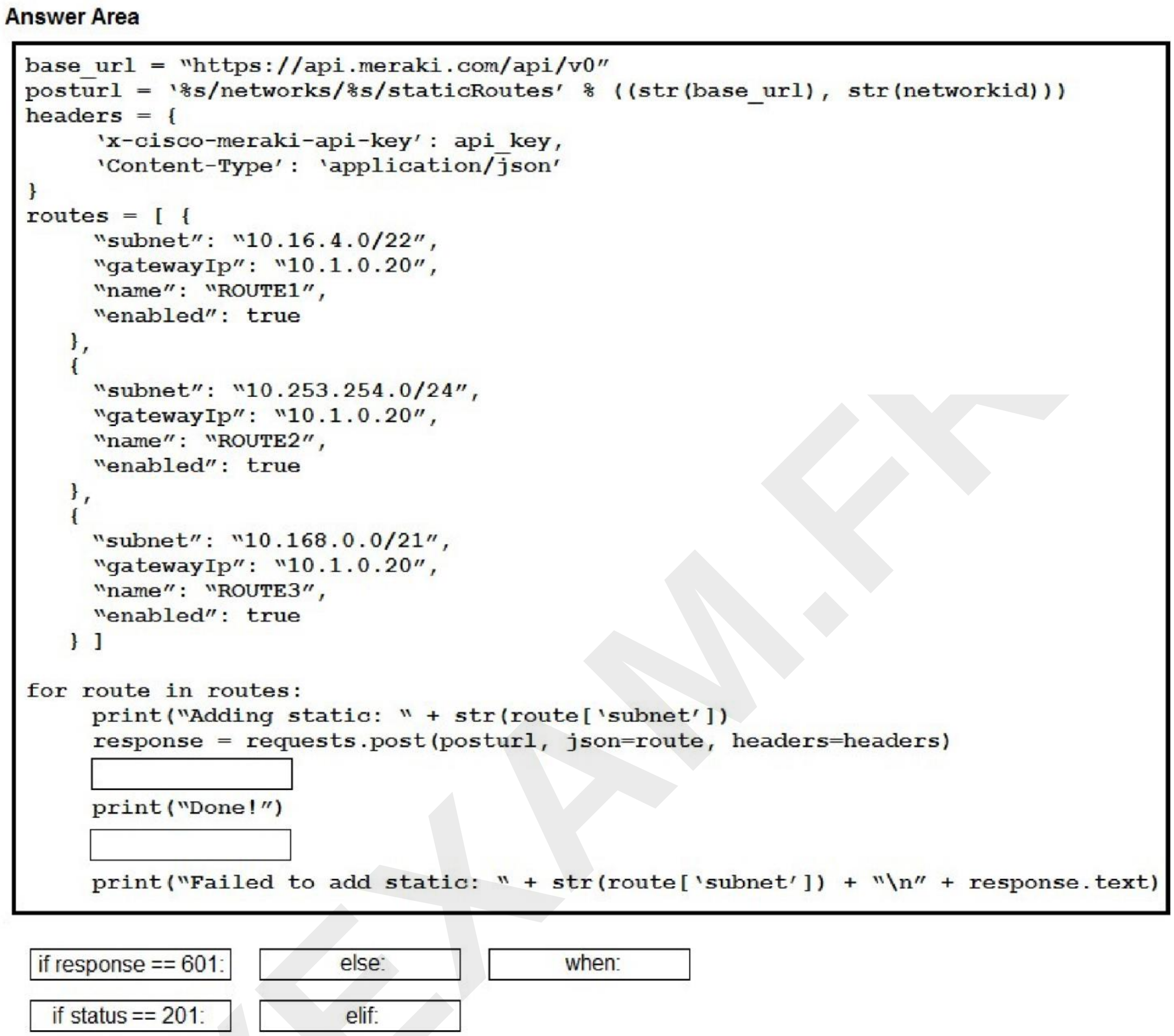
Answer: D

Explanation:

This file contains raw data that is needed for token authentication.

Question: 42

DRAG DROP -
Drag and drop the expressions from below onto the code to implement error handling. Not all options are used.
Select and Place:



Answer Area

```
base_url = “https://api.meraki.com/api/v0”
posturl = ‘%s/networks/%s/staticRoutes’ % ((str(base_url), str(networkid))
headers = {
      ‘x-cisco-meraki-api-key’: api_key,
      ‘Content-Type’: ‘application/json’
}
routes = [ {
      “subnet”: “10.16.4.0/22”,
      “gatewayIp”: “10.1.0.20”,
      “name”: “ROUTE1”,
      “enabled”: true
    },
    {
      “subnet”: “10.253.254.0/24”,
      “gatewayIp”: “10.1.0.20”,
      “name”: “ROUTE2”,
      “enabled”: true
    },
    {
      “subnet”: “10.168.0.0/21”,
      “gatewayIp”: “10.1.0.20”,
      “name”: “ROUTE3”,
      “enabled”: true
    } ]

for route in routes:
      print(“Adding static: ” + str(route[‘subnet’])
      response = requests.post(posturl, json=route, headers=headers)
      □
      print(“Done!”)
      □
      print(“Failed to add static: ” + str(route[‘subnet’]) + “\n” + response.text)
```

if response == 601:

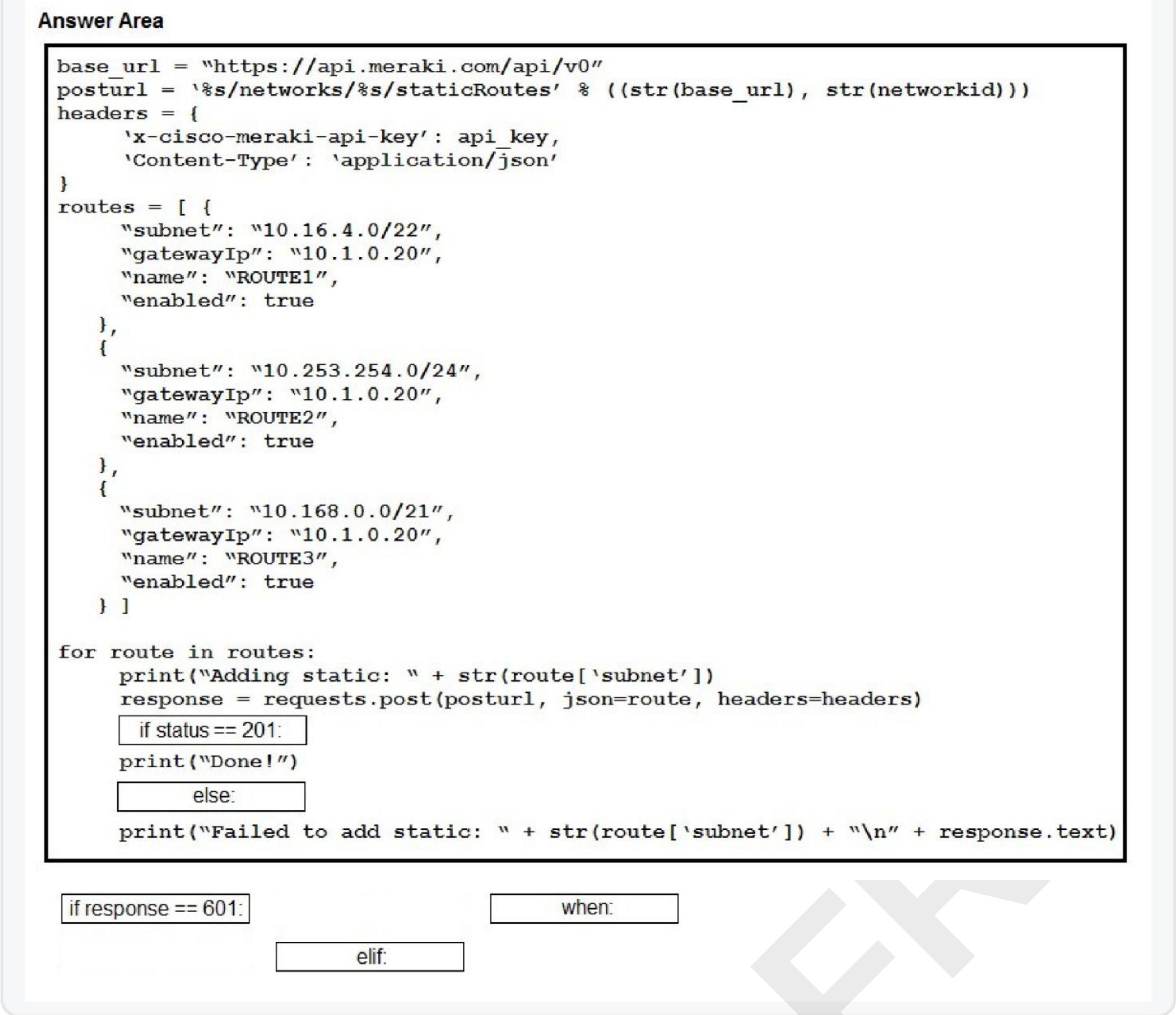
else:

when:

if status == 201:

elif:

Answer:



Answer Area

```
base_url = “https://api.meraki.com/api/v0”
posturl = ‘%s/networks/%s/staticRoutes’ % ((str(base_url), str(networkid))
headers = {
      ‘x-cisco-meraki-api-key’: api_key,
      ‘Content-Type’: ‘application/json’
}
routes = [ {
      “subnet”: “10.16.4.0/22”,
      “gatewayIp”: “10.1.0.20”,
      “name”: “ROUTE1”,
      “enabled”: true
    },
    {
      “subnet”: “10.253.254.0/24”,
      “gatewayIp”: “10.1.0.20”,
      “name”: “ROUTE2”,
      “enabled”: true
    },
    {
      “subnet”: “10.168.0.0/21”,
      “gatewayIp”: “10.1.0.20”,
      “name”: “ROUTE3”,
      “enabled”: true
    } ]

for route in routes:
      print(“Adding static: ” + str(route[‘subnet’])
      response = requests.post(posturl, json=route, headers=headers)
      if status == 201:
      print(“Done!”)
      else:
      print(“Failed to add static: ” + str(route[‘subnet’]) + “\n” + response.text)
```

Question: 43

User report that they can no longer able to process transactions with the online ordering application, and the logging dashboard is displaying these messages.

Fri Jan 10 19:37:31.123 EST 2020 [FRONTEND] INFO: Incoming request to add item to cart from user 45834534858 Fri Jan 10 19:37:31.247 EST 2020 [BACKEND] INFO: Attempting to add item to cart

Fri Jan 10 19:37:31.250 EST 2020 [BACKEND] ERROR: Failed to add item: MYSQLDB ERROR: Connection refused What is causing the problem seen in these log messages?

A.The database server container has crashed.

B.The backend process is overwhelmed with too many transactions.

C.The backend is not authorized to commit to the database.

D.The user is not authorized to add the item to their cart.

Answer: C

Explanation:

The correct answer is **C. The backend is not authorized to commit to the database.**

Here's the justification:

The log messages clearly indicate a failure in communication between the backend application and the

MySQL database. The specific error, "MYSQLDB ERROR: Connection refused," points towards a network-level issue or an authorization problem rather than a crashed container or overwhelming transactions. While a crashed container could lead to connection issues, the log doesn't explicitly mention container related errors or shutdown messages. Furthermore, “Connection Refused” generally indicates that either no service is listening on the specified port or that a firewall is blocking the traffic. In this case, the backend application attempts to connect to the database (as shown in the "[BACKEND] INFO: Attempting to add item to cart" log) but is denied. This indicates the service is online, so a more likely cause is that the application isn't authorized or allowed to connect by the database. This could result from several scenarios, including:

Database User Permissions: The backend application might be using database credentials that lack the necessary privileges to connect. Databases typically employ user accounts with specific authorization to access and modify data.

Network ACLs/Firewall Rules: There might be network-level access control lists (ACLs) or firewall rules that restrict the backend application from reaching the database server.

Incorrect Database Configuration: The backend application could be configured with incorrect connection parameters like IP address, port, or username. However the log indicates the connection was initially successful, indicating the configuration is likely correct and the permissions have been revoked or not yet created.

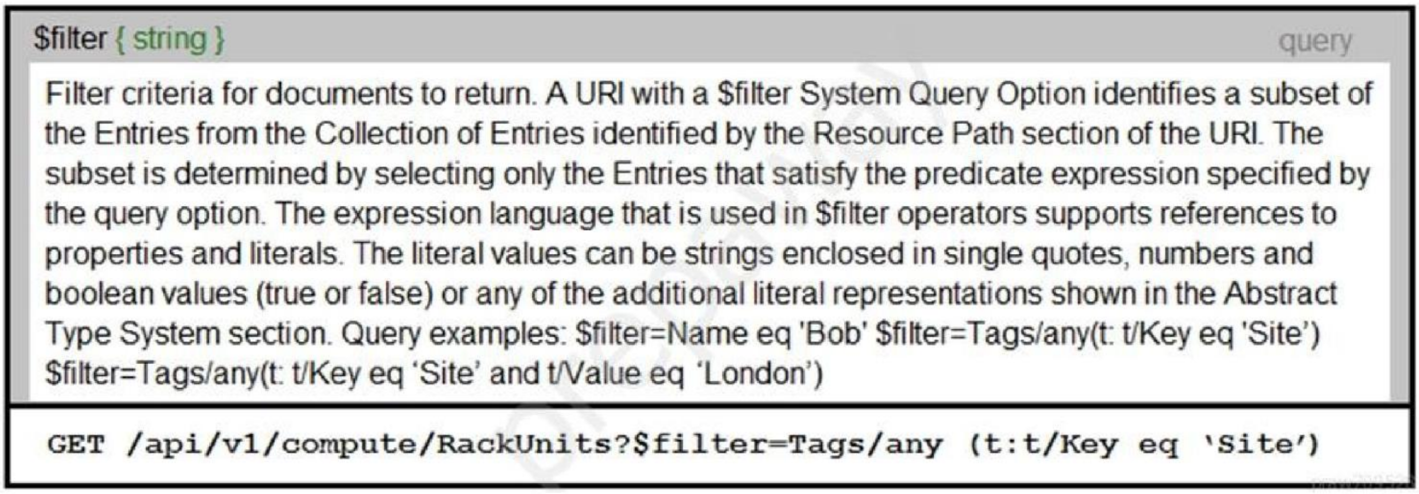
Option A, "The database server container has crashed," is not the most likely reason because the error specifically says "Connection refused" rather than, for example, "database server unavailable". Connection Refused indicates the service is running but is refusing connections from the backend. Option B, "The backend process is overwhelmed with too many transactions," would likely manifest as timeouts or slowdowns, not a "connection refused" error. Option D, "The user is not authorized to add the item to their cart," would typically be reflected in an error message from the frontend or backend related to user roles/permissions and not a database connection refusal.

Therefore, the "MYSQLDB ERROR: Connection refused" indicates a more fundamental authorization or network issue between the backend and the database, pointing towards the backend lacking authorization.

Authoritative Links for Further Research:

1. **MySQL Connection Errors:**https://dev.mysql.com/doc/refman/8.0/en/connection-errors.html (This official MySQL documentation provides detailed information about various connection errors, including "Connection refused".)
2. **Database Security Best Practices:**https://owasp.org/www-project-top-ten/ (OWASP (Open Web Application Security Project) provides guidelines on secure database configurations and access control.)
3. **Network Security Fundamentals:** Explore resources on network ACLs and firewall rules to understand how these can affect connectivity between services. (e.g., search for "firewall rules" or "network access control lists").

Question: 44



$filter { string } query

Filter criteria for documents to return. A URI with a $filter System Query Option identifies a subset of the Entries from the Collection of Entries identified by the Resource Path section of the URI. The subset is determined by selecting only the Entries that satisfy the predicate expression specified by the query option. The expression language that is used in $filter operators supports references to properties and literals. The literal values can be strings enclosed in single quotes, numbers and boolean values (true or false) or any of the additional literal representations shown in the Abstract Type System section. Query examples: $filter=Name eq 'Bob' $filter=Tags/any(t: t/Key eq 'Site') $filter=Tags/any(t: t/Key eq ‘Site’ and t/Value eq ‘London’)

```
GET /api/v1/compute/RackUnits?$filter=Tags/any (t:t/Key eq ‘Site’)
```

Refer to the exhibit. An Intersight API is being used to query RackUnit resources that have a tag keyword set to `Site`.
What is the expected output of this command?

A.list of all resources that have a tag with the keyword Site

B.error message because the Value field was not specified

C.error message because the tag filter should be lowercase

D.list of all sites that contain RackUnit tagged compute resources

Answer: A

Explanation:

list of all resources that have a tag with the keyword Site.

Question: 45

A user is receiving a 429 Too Many Requests error.
Which scheme is the server employing that causes this error?

A.rate limiting

B.time outs

C.caching

D.redirection

Answer: A

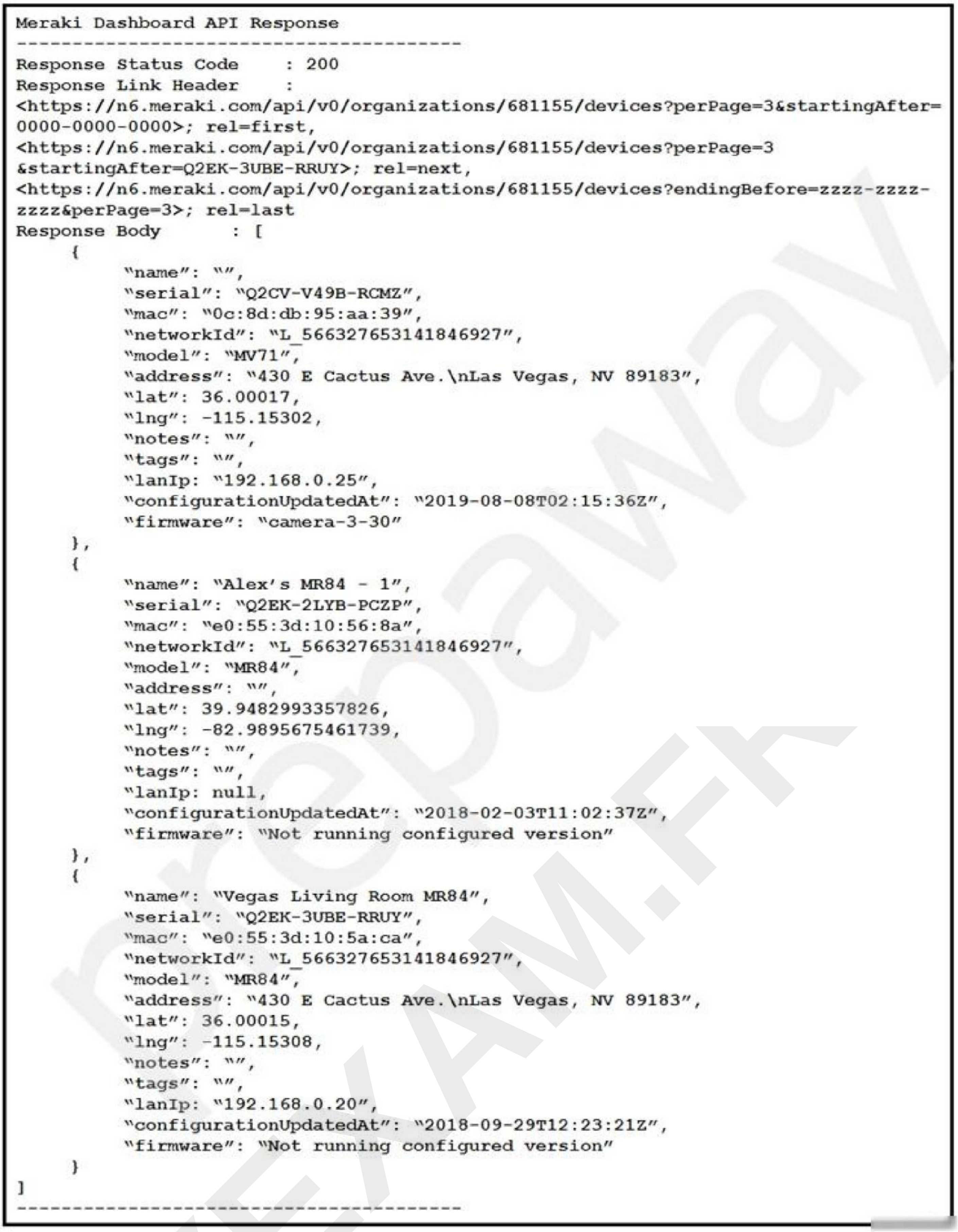
Explanation:

The correct answer is **A. rate limiting**. A 429 Too Many Requests error is a specific HTTP status code indicating that a client has sent too many requests in a given timeframe. This error directly stems from a server implementing a rate-limiting mechanism. Rate limiting is a common technique used to control the volume of requests a server processes from a single client or source, preventing abuse, protecting resources, and ensuring fair access to services. When a client exceeds the defined rate limit, the server responds with a 429 error, signaling that the client needs to reduce their request frequency. Options B, C, and D are not direct causes of a 429 error. Timeouts (B) usually result in a 504 Gateway Timeout or similar errors, signifying that a request took too long to complete. Caching (C), although related to performance and response times, does not intrinsically cause a 429 error; rather, effective caching can often reduce the need for frequent requests.

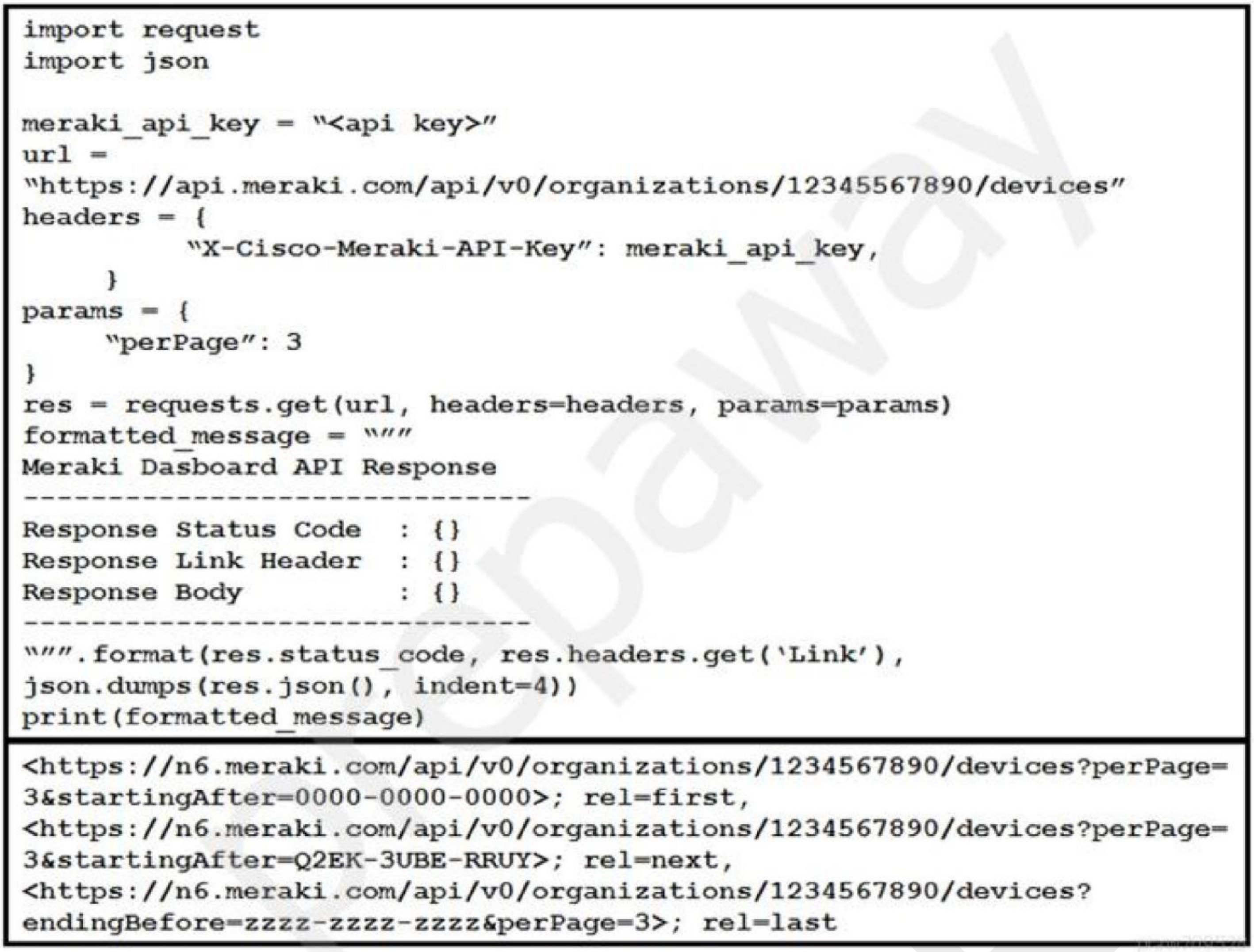
Redirection (D) involves a different set of HTTP status codes (3xx) to guide the client to a new location. Rate limiting is a core concept in API design and cloud-based services. For further information, you can explore resources like the following:

1. **MDN Web Docs on HTTP Status Codes**: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/429 This document provides in-depth details on the 429 Too Many Requests status code and its implications.
2. **API Rate Limiting Best Practices**: Search for "API rate limiting best practices" on any major search engine (Google, Bing, DuckDuckGo) to find articles and tutorials explaining its purpose and implementation.
3. **Cloud provider documentation**: Most major cloud providers like AWS, Azure, and Google Cloud have specific documentation on how they implement rate limiting within their services. Search their specific documentation to see examples of rate limiting in action.

Question: 46



```
Meraki Dashboard API Response
----------------------------------------
Response Status Code     : 200
Response Link Header     :
<https://n6.meraki.com/api/v0/organizations/681155/devices?perPage=3&startingAfter=
0000-0000-0000>; rel=first,
<https://n6.meraki.com/api/v0/organizations/681155/devices?perPage=3
&startingAfter=Q2EK-3UBE-RRUY>; rel=next,
<https://n6.meraki.com/api/v0/organizations/681155/devices?endingBefore=zzzz-zzzz-
zzzz&perPage=3>; rel=last
Response Body       : [
     {
          “name”: “”,
          “serial”: “Q2CV-V49B-RCMZ”,
          “mac”: “0c:8d:db:95:aa:39”,
          “networkId”: “L_566327653141846927”,
          “model”: “MV71”,
          “address”: “430 E Cactus Ave.\nLas Vegas, NV 89183”,
          “lat”: 36.00017,
          “lng”: -115.15302,
          “notes”: “”,
          “tags”: “”,
          “lanIp: “192.168.0.25”,
          “configurationUpdatedAt”: “2019-08-08T02:15:36Z”,
          “firmware”: “camera-3-30”
     },
     {
          “name”: “Alex’s MR84 - 1”,
          “serial”: “Q2EK-2LYB-PCZP”,
          “mac”: “e0:55:3d:10:56:8a”,
          “networkId”: “L_566327653141846927”,
          “model”: “MR84”,
          “address”: “”,
          “lat”: 39.9482993357826,
          “lng”: -82.9895675461739,
          “notes”: “”,
          “tags”: “”,
          “lanIp: null,
          “configurationUpdatedAt”: “2018-02-03T11:02:37Z”,
          “firmware”: “Not running configured version”
     },
     {
          “name”: “Vegas Living Room MR84”,
          “serial”: “Q2EK-3UBE-RRUY”,
          “mac”: “e0:55:3d:10:5a:ca”,
          “networkId”: “L_566327653141846927”,
          “model”: “MR84”,
          “address”: “430 E Cactus Ave.\nLas Vegas, NV 89183”,
          “lat”: 36.00015,
          “lng”: -115.15308,
          “notes”: “”,
          “tags”: “”,
          “lanIp: “192.168.0.20”,
          “configurationUpdatedAt”: “2018-09-29T12:23:21Z”,
          “firmware”: “Not running configured version”
     }
]
----------------------------------------
```



```
import request
import json

meraki_api_key = "<api key>"
url =
"https://api.meraki.com/api/v0/organizations/12345567890/devices"
headers = {
        "X-Cisco-Meraki-API-Key": meraki_api_key,
    }
params = {
    "perPage": 3
}
res = requests.get(url, headers=headers, params=params)
formatted_message = """
Meraki Dasboard API Response
------------------------------
Response Status Code   : {}
Response Link Header   : {}
Response Body          : {}
------------------------------
""".format(res.status_code, res.headers.get('Link'),
json.dumps(res.json(), indent=4))
print(formatted_message)
```

```
<https://n6.meraki.com/api/v0/organizations/1234567890/devices?perPage=
3&startingAfter=0000-0000-0000>; rel=first,
<https://n6.meraki.com/api/v0/organizations/1234567890/devices?perPage=
3&startingAfter=Q2EK-3UBE-RRUY>; rel=next,
<https://n6.meraki.com/api/v0/organizations/1234567890/devices?
endingBefore=zzzz-zzzz-zzzz&perPage=3>; rel=last
```

Refer to the exhibit. Which line of code must be added to this code snippet to allow an application to pull the next set of paginated items?

A.requests.get(url, links=['next']['url'])
B.requests.get(url, headers=links['next']['url'])
C.requests.get(res.links['next']['url'], headers=headers)
D.requests.get(res.headers.get('Link')['next']['url'], headers=headers)

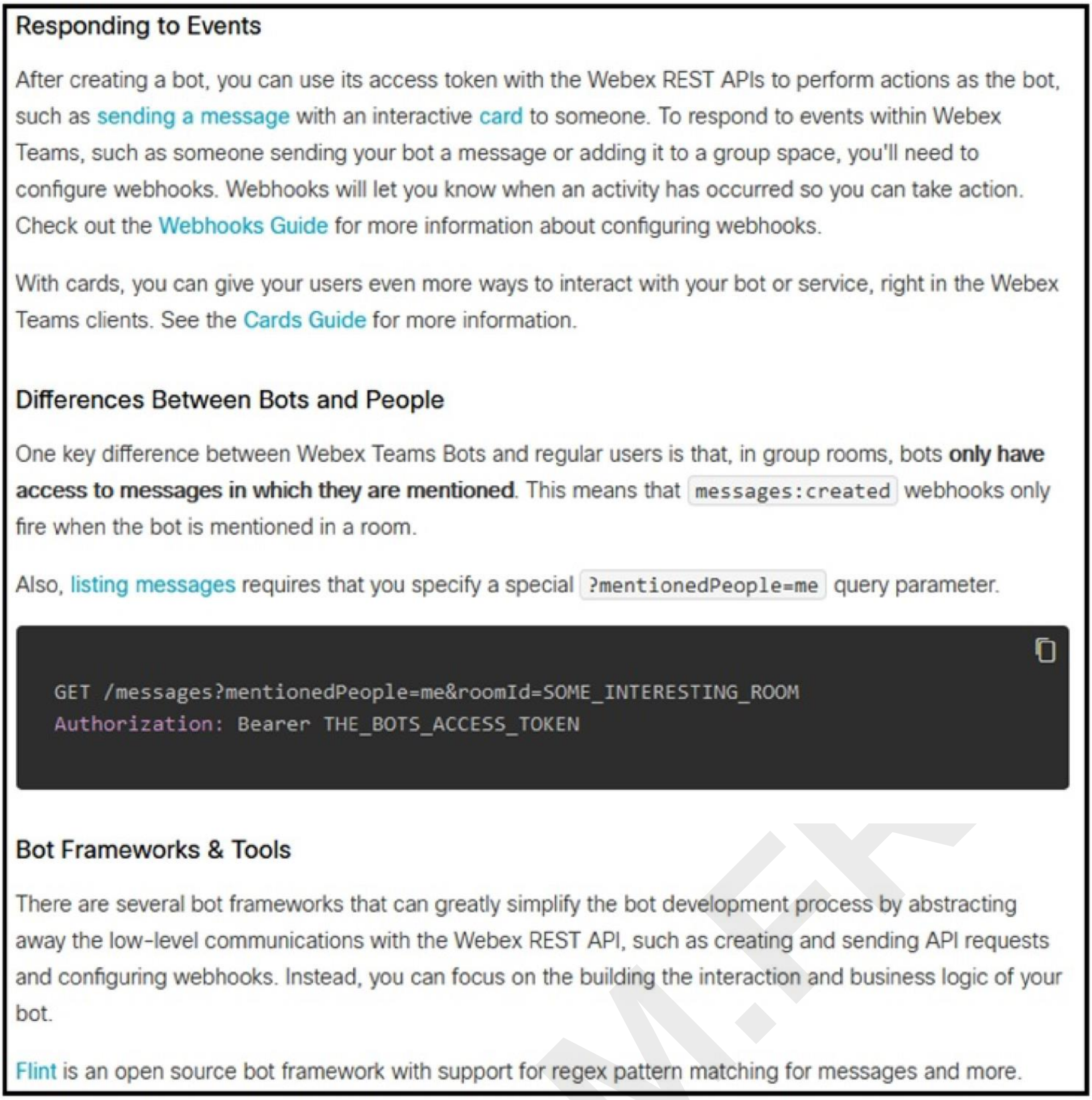
Answer: C

Explanation:

Reference:

https://docs.python-requests.org/en/master/user/advanced/#link-headers

Question: 47



Responding to Events

After creating a bot, you can use its access token with the Webex REST APIs to perform actions as the bot, such as sending a message with an interactive card to someone. To respond to events within Webex Teams, such as someone sending your bot a message or adding it to a group space, you'll need to configure webhooks. Webhooks will let you know when an activity has occurred so you can take action. Check out the Webhooks Guide for more information about configuring webhooks.

With cards, you can give your users even more ways to interact with your bot or service, right in the Webex Teams clients. See the Cards Guide for more information.

Differences Between Bots and People

One key difference between Webex Teams Bots and regular users is that, in group rooms, bots **only have access to messages in which they are mentioned**. This means that `messages:created` webhooks only fire when the bot is mentioned in a room.

Also, listing messages requires that you specify a special `?mentionedPeople=me` query parameter.

```
GET /messages?mentionedPeople=me&roomId=SOME_INTERESTING_ROOM
Authorization: Bearer THE_BOTS_ACCESS_TOKEN
```

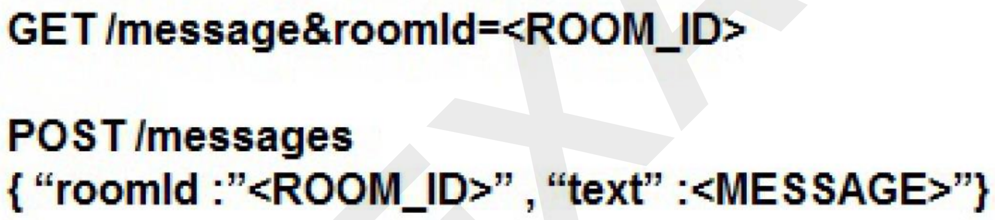
Bot Frameworks & Tools

There are several bot frameworks that can greatly simplify the bot development process by abstracting away the low-level communications with the Webex REST API, such as creating and sending API requests and configuring webhooks. Instead, you can focus on the building the interaction and business logic of your bot.

Flint is an open source bot framework with support for regex pattern matching for messages and more.

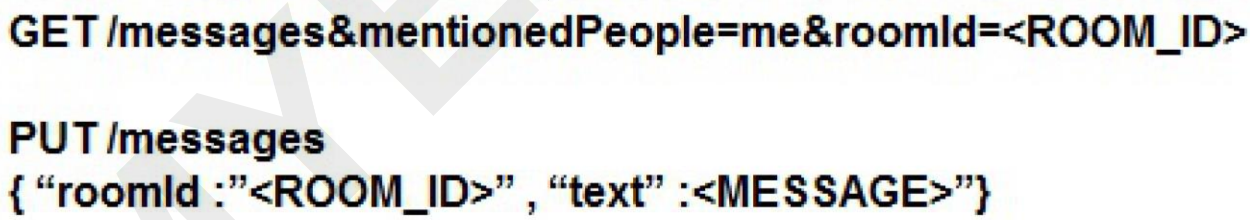
Refer to the exhibit. Which set of API requests must be executed by a Webex Teams bot after receiving a webhook callback to process messages in a room and reply with a new message back to the same room?

A.



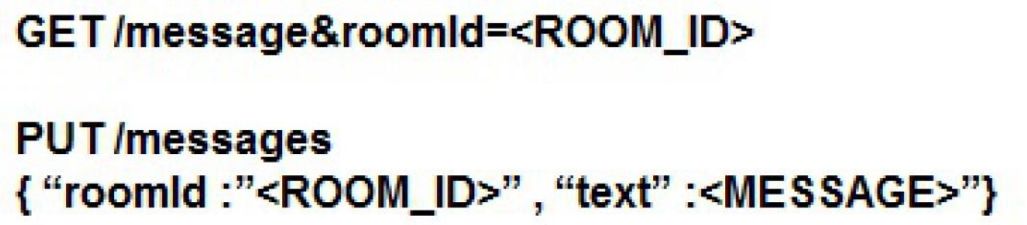
```
GET /message&roomId=<ROOM_ID>

POST /messages
{ "roomId :"<ROOM_ID>" , "text" :<MESSAGE>"}
```

B.



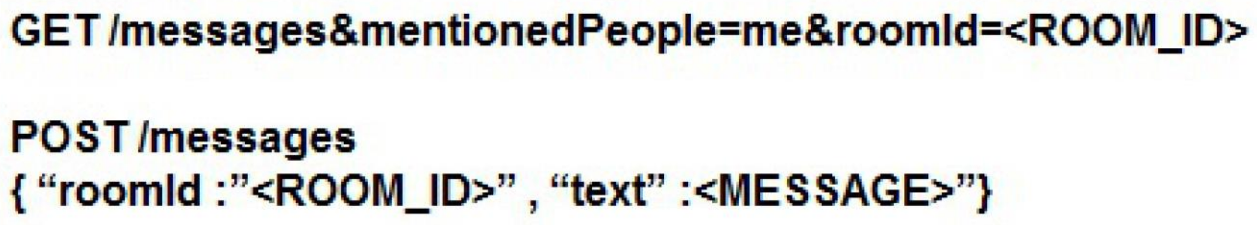
```
GET /messages&mentionedPeople=me&roomId=<ROOM_ID>

PUT /messages
{ "roomId :"<ROOM_ID>" , "text" :<MESSAGE>"}
```

C.



```
GET /message&roomId=<ROOM_ID>

PUT /messages
{ "roomId :"<ROOM_ID>" , "text" :<MESSAGE>"}
```

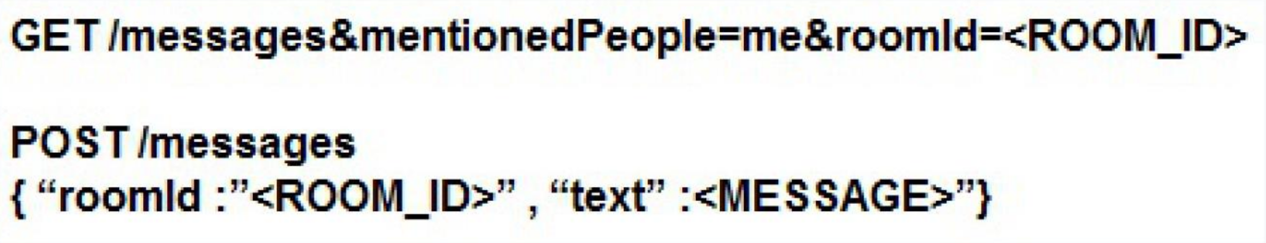
D.



```
GET /messages&mentionedPeople=me&roomId=<ROOM_ID>

POST /messages
{ "roomId :"<ROOM_ID>" , "text" :<MESSAGE>"}
```

Answer: D

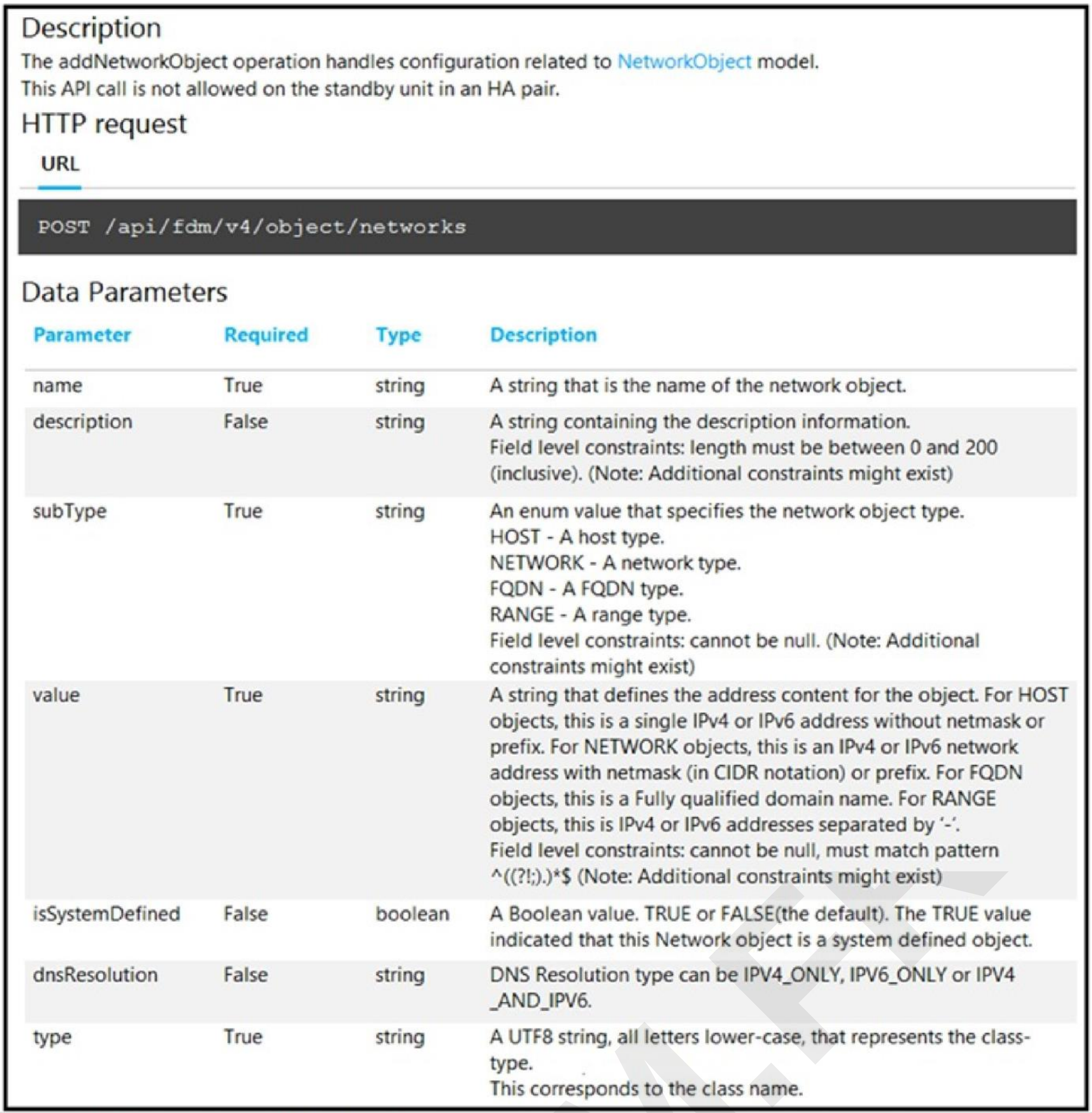
Explanation:



```
GET /messages&mentionedPeople=me&roomId=<ROOM_ID>

POST /messages
{ "roomId :"<ROOM_ID>" , "text" :<MESSAGE>"}
```

Question: 49

DRAG DROP -



Description

The addNetworkObject operation handles configuration related to NetworkObject model.
This API call is not allowed on the standby unit in an HA pair.

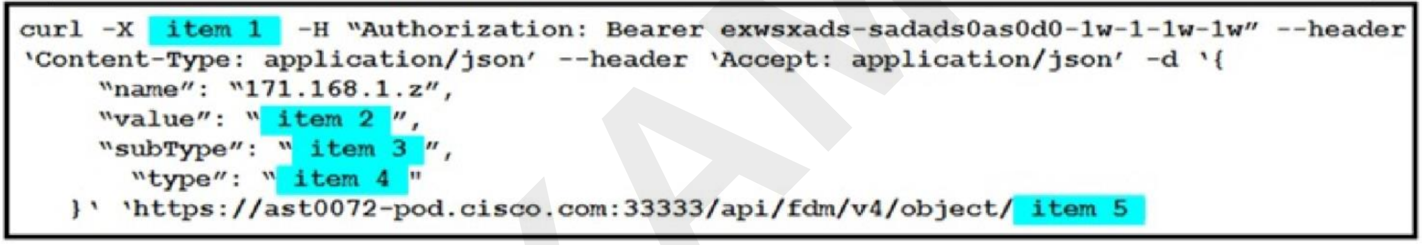
HTTP request

URL

```
POST /api/fdm/v4/object/networks
```

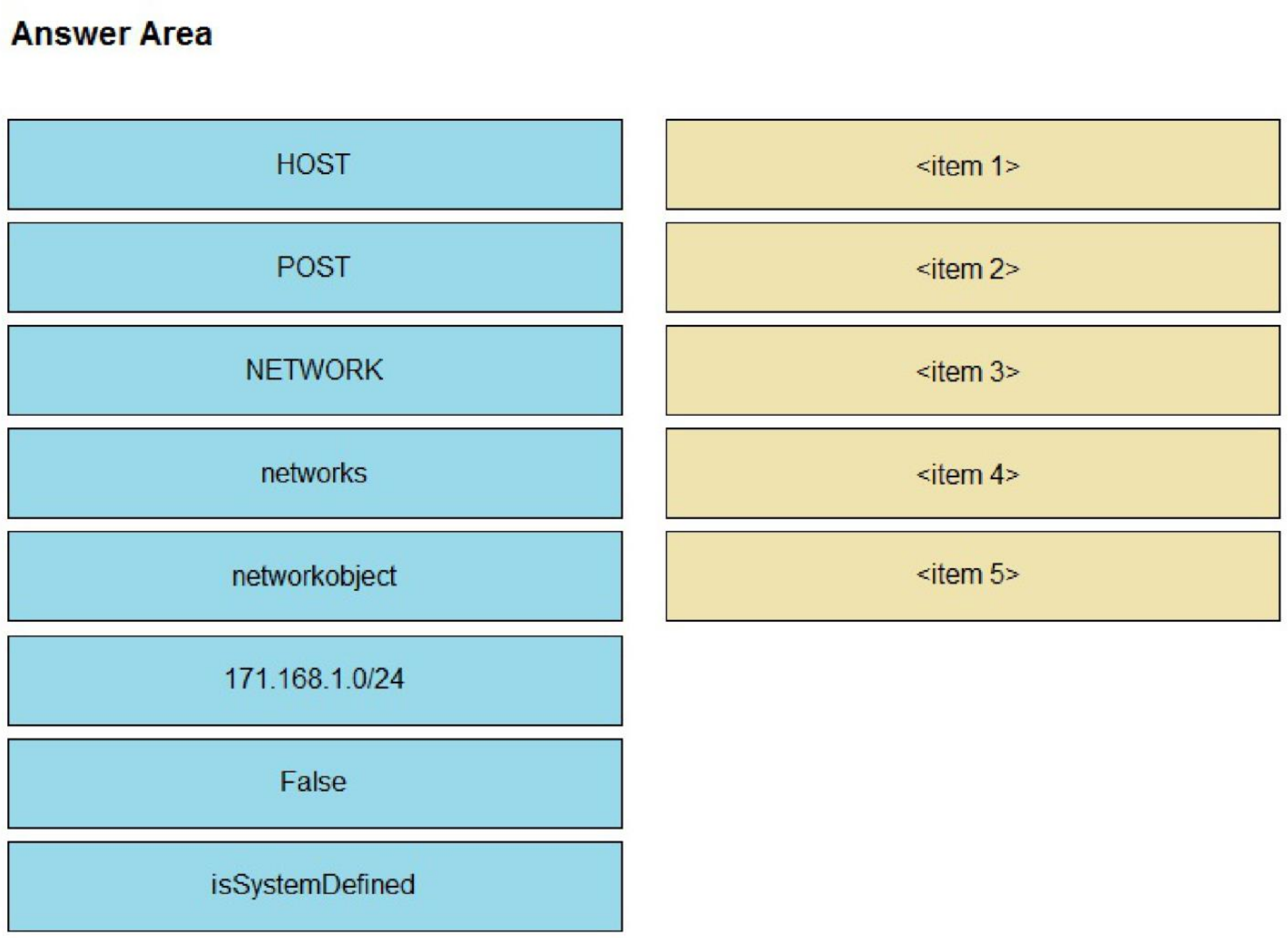
Data Parameters

Parameter	Required	Type	Description
name	True	string	A string that is the name of the network object.
description	False	string	A string containing the description information. Field level constraints: length must be between 0 and 200 (inclusive). (Note: Additional constraints might exist)
subType	True	string	An enum value that specifies the network object type. HOST - A host type. NETWORK - A network type. FQDN - A FQDN type. RANGE - A range type. Field level constraints: cannot be null. (Note: Additional constraints might exist)
value	True	string	A string that defines the address content for the object. For HOST objects, this is a single IPv4 or IPv6 address without netmask or prefix. For NETWORK objects, this is an IPv4 or IPv6 network address with netmask (in CIDR notation) or prefix. For FQDN objects, this is a Fully qualified domain name. For RANGE objects, this is IPv4 or IPv6 addresses separated by '-'. Field level constraints: cannot be null, must match pattern ^((?!;).)*$ (Note: Additional constraints might exist)
isSystemDefined	False	boolean	A Boolean value. TRUE or FALSE(the default). The TRUE value indicated that this Network object is a system defined object.
dnsResolution	False	string	DNS Resolution type can be IPV4_ONLY, IPV6_ONLY or IPV4 _AND_IPV6.
type	True	string	A UTF8 string, all letters lower-case, that represents the class-type. This corresponds to the class name.



```
curl -X  item 1  -H "Authorization: Bearer exwsxads-sadads0as0d0-1w-1-1w-1w" --header
'Content-Type: application/json' --header 'Accept: application/json' -d '{
     "name": "171.168.1.z",
     "value": " item 2 ",
     "subType": " item 3 ",
       "type": " item 4 "
   }' 'https://ast0072-pod.cisco.com:33333/api/fdm/v4/object/ item 5
```

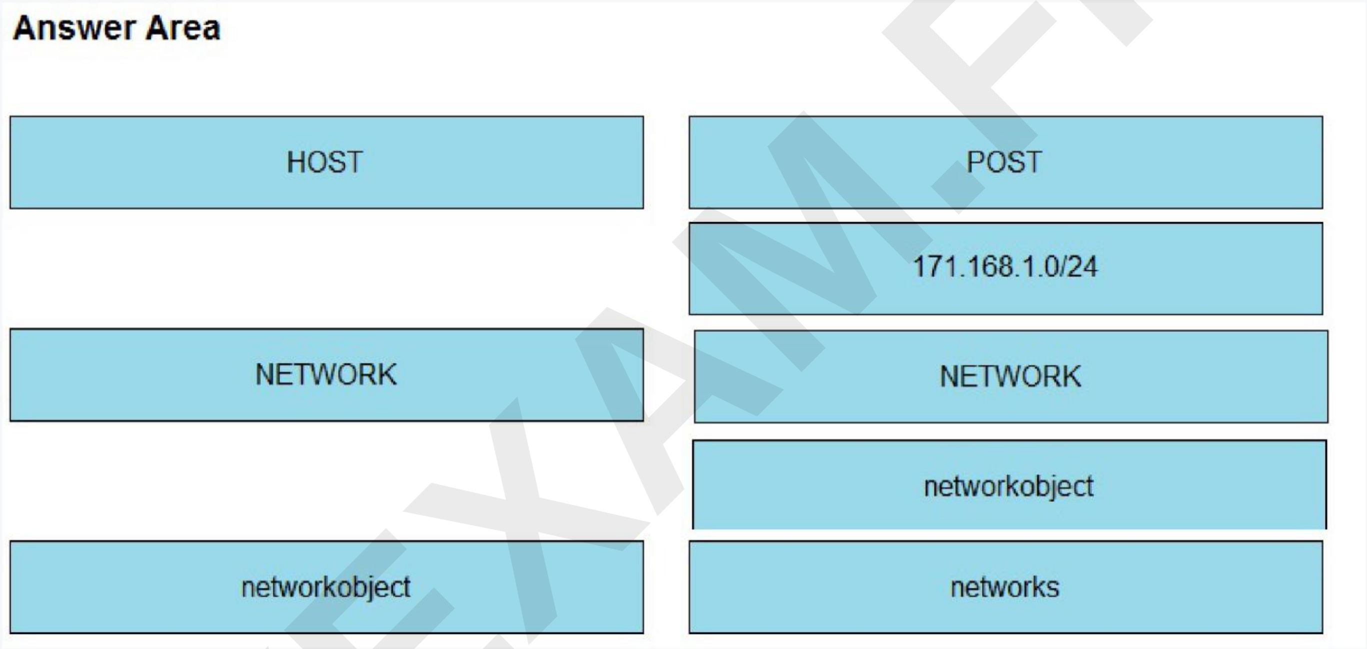
Refer to the exhibit. Drag and drop the code snippets from the left onto the item numbers on the right that match the missing sections in the cURL exhibit to complete the cURL request to FirePower Device Manager API to create objects. Not all code snippets are used.
Select and Place:



Answer Area

HOST	<item 1>
POST	<item 2>
NETWORK	<item 3>
networks	<item 4>
networkobject	<item 5>
171.168.1.0/24	
False	
isSystemDefined	

Answer:



Answer Area

HOST	POST
	171.168.1.0/24
NETWORK	NETWORK
	networkobject
networkobject	networks

Explanation:

1. POST

2. 171.168.1.0/24

3. NETWORK

4. networkobject

5. networks

Question: 50

Which RFC5988 (Web Linking) relation type is used in the Link header to control pagination in APIs?

A.rel=index
B.rel=page
C.rel=next
D.rel=section

Answer: C

Explanation:

The correct answer is **C. rel=next.** RFC5988, commonly known as Web Linking, defines how to establish relationships between resources using the HTTP Link header. Pagination, a crucial aspect of API design for handling large datasets, benefits significantly from this mechanism. The rel=next link relation type signals the presence of a subsequent resource in a sequence, typically used to navigate through paginated results. For example, a GET request might return a JSON object containing data and also include a Link header like Link: <https://api.example.com/items?page=2>; rel="next". This indicates that the next set of results can be obtained from the URL specified. The rel=index (option A) typically signifies a link to the index or starting point of a collection. The rel=page (option B) is not a standard link relation type defined in RFC5988 or commonly used for pagination. Lastly, rel=section (option D) typically relates to parts of a document or information, not specifically to navigation between paginated datasets. The widespread usage of rel=next for API pagination makes it the standard and preferred way to inform clients about how to retrieve the subsequent page of results within a larger collection.

Authoritative Link:

RFC5988: https://datatracker.ietf.org/doc/html/rfc5988

Question: 51

A client is written that uses a REST API to interact with a server. Using HTTPS as the transport, an HTTP request is sent and received an HTTP response. The response contains the HTTP response status code: 503 Service Unavailable.
Which action is the appropriate response?

A.Add an Authorization header that supplies appropriate credentials and sends the updated request.
B.Resend the request using HTTP as the transport instead of HTTPS.
C.Add an Accept header that indicates the content types that the client understands and send the updated request.
D.Look for a Retry-After header in the response and resend the request after the amount of time indicated.

Answer: D

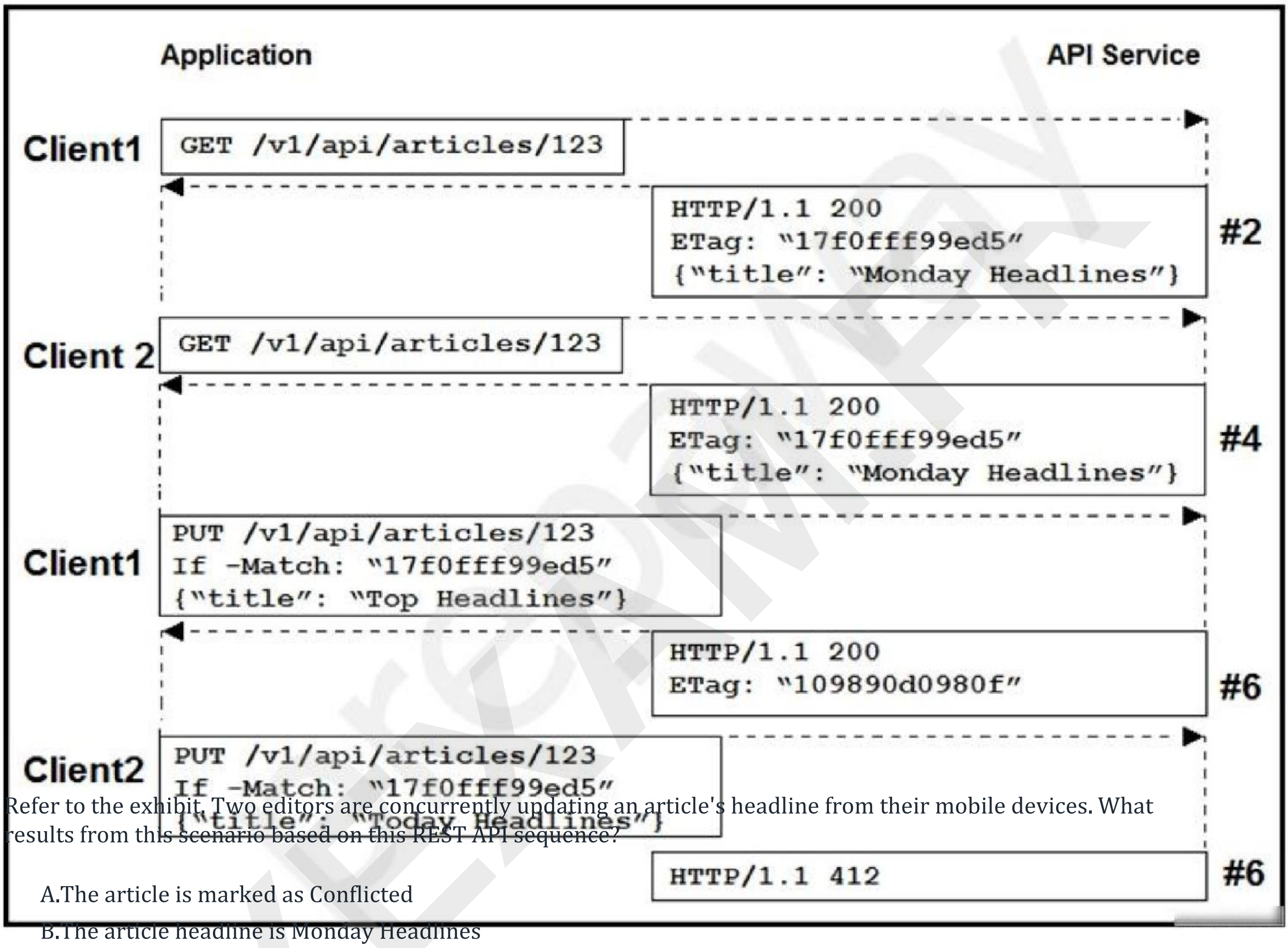
Explanation:

The correct action for a 503 Service Unavailable response is **D. Look for a Retry-After header in the response and resend the request after the amount of time indicated.** A 503 status code signifies that the server is temporarily unable to handle the request due to overload, maintenance, or another transient issue. It's a server-side problem, not necessarily related to the client's credentials, transport protocol, or content negotiation. Adding an Authorization header (A) or changing the protocol to HTTP (B) won't solve the server's unavailability. Including an Accept header (C) is relevant for content negotiation, but not the fundamental issue of a server being down. The presence of a 'Retry-After' header, which is optional in a 503 response, specifically instructs the client when to try again, avoiding further strain on the server. This demonstrates good API design principles promoting resilience. The client should respect the 'Retry-After' header, if present, to avoid exacerbating the server's temporary unavailability. Retrying prematurely may further overload the system. This behavior exemplifies a common practice in handling transient failures in distributed systems, enhancing robustness and user experience. Therefore, checking for and honoring a 'Retry-After' header is the most appropriate and efficient action for a client encountering a 503 error.

Authoritative Links:

MDN Web Docs on HTTP Status Codes:https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/503 **RFC 7231, Section 6.6.4 on 503 Service Unavailable:**https://datatracker.ietf.org/doc/html/rfc7231#section-6.6.4
Wikipedia Entry on HTTP 503:https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#5xx_server_errors

Question: 52



```
GET /v1/api/articles/123

HTTP/1.1 200
ETag: "17f0fff99ed5"
{"title": "Monday Headlines"}

GET /v1/api/articles/123

HTTP/1.1 200
ETag: "17f0fff99ed5"
{"title": "Monday Headlines"}

PUT /v1/api/articles/123
If -Match: "17f0fff99ed5"
{"title": "Top Headlines"}

HTTP/1.1 200
ETag: "109890d0980f"

PUT /v1/api/articles/123
If -Match: "17f0fff99ed5"
{"title": "Today Headlines"}

HTTP/1.1 412
```

Refer to the exhibit. Two editors are concurrently updating an article's headline from their mobile devices. What results from this scenario based on this REST API sequence?

A.The article is marked as Conflicted

B.The article headline is Monday Headlines

C.The article headline is Today Headlines

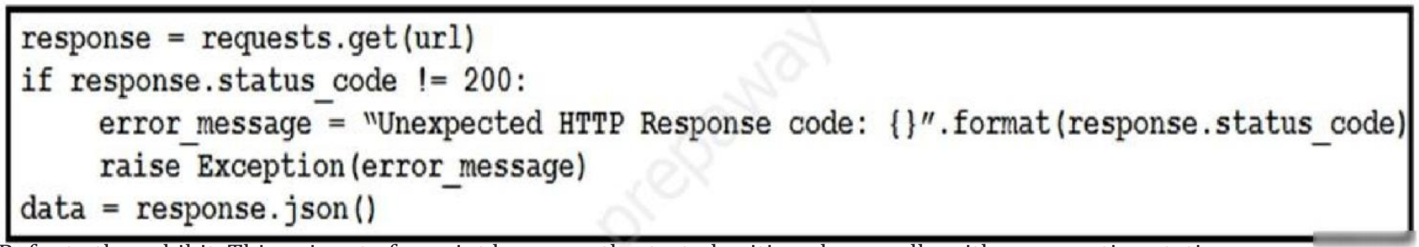
D.The article headline is Top Headlines

Answer: D

Explanation:

The article headline is Top Headlines.

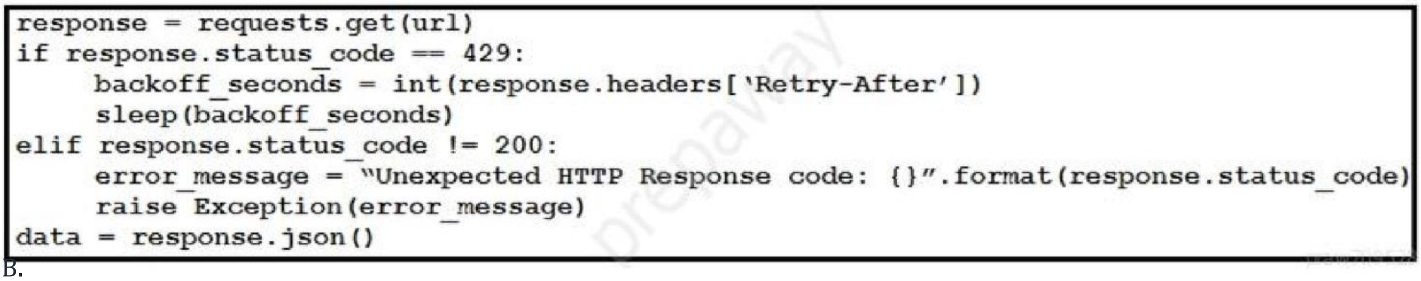
Question: 53



```
response = requests.get(url)
if response.status_code != 200:
     error_message = "Unexpected HTTP Response code: {}".format(response.status_code)
     raise Exception(error_message)
data = response.json()
```

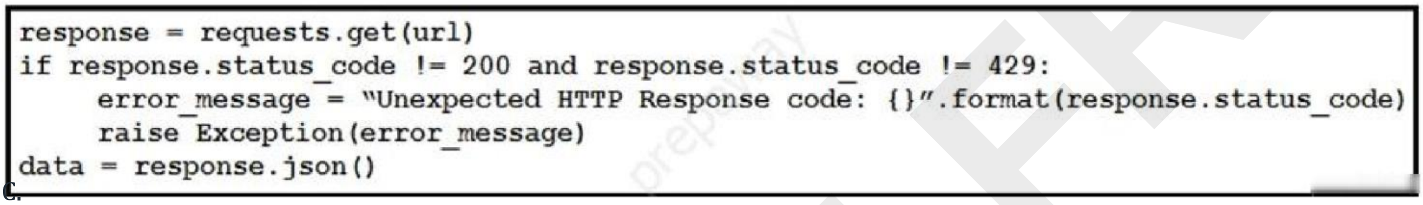
Refer to the exhibit. This snippet of a script has recently started exiting abnormally with an exception stating `Unexpected HTTP Response code: 429`.
Which solution handles rate limiting by the remote API?

A.



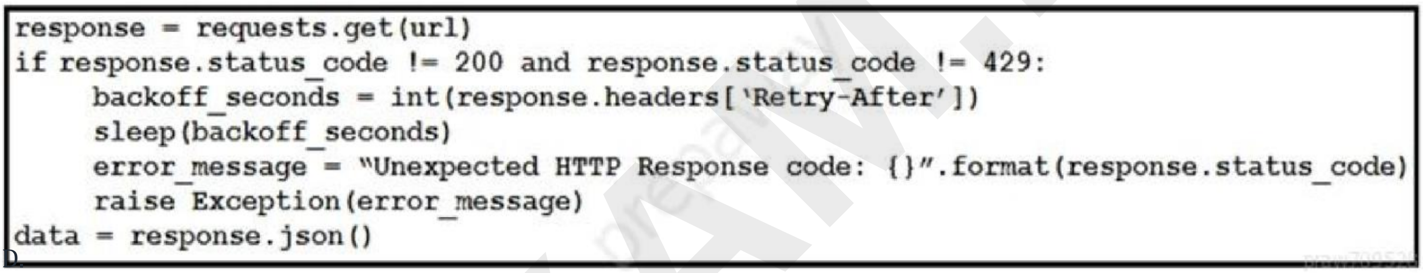
```
response = requests.get(url)
if response.status_code == 429:
     backoff_seconds = int(response.headers['Retry-After'])
     sleep(backoff_seconds)
elif response.status_code != 200:
     error_message = "Unexpected HTTP Response code: {}".format(response.status_code)
     raise Exception(error_message)
data = response.json()
```

B.



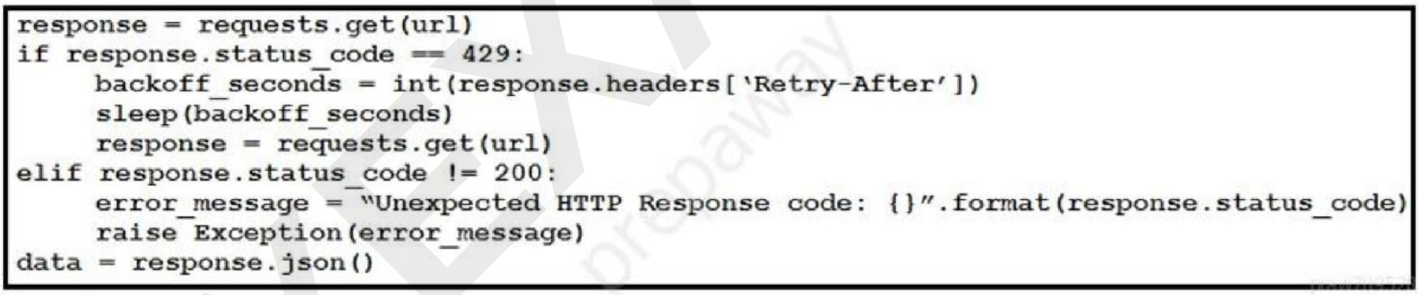
```
response = requests.get(url)
if response.status_code != 200 and response.status_code != 429:
     error_message = "Unexpected HTTP Response code: {}".format(response.status_code)
     raise Exception(error_message)
data = response.json()
```

C.



```
response = requests.get(url)
if response.status_code != 200 and response.status_code != 429:
     backoff_seconds = int(response.headers['Retry-After'])
     sleep(backoff_seconds)
     error_message = "Unexpected HTTP Response code: {}".format(response.status_code)
     raise Exception(error_message)
data = response.json()
```

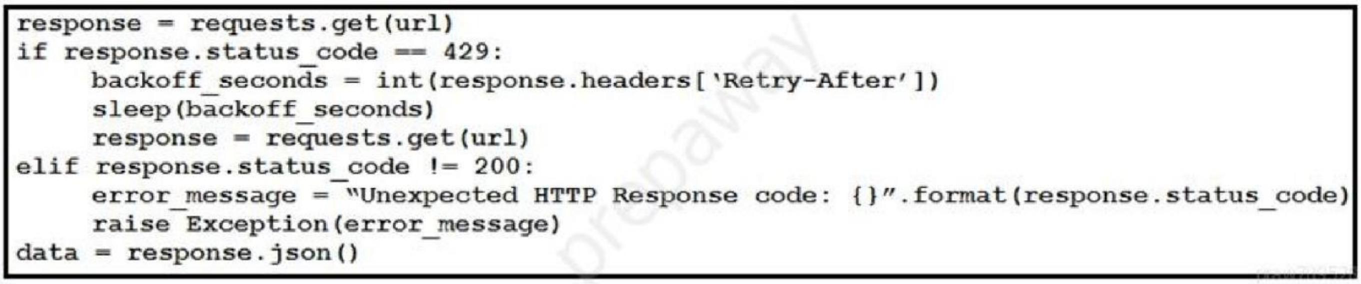
D.



```
response = requests.get(url)
if response.status_code == 429:
     backoff_seconds = int(response.headers['Retry-After'])
     sleep(backoff_seconds)
     response = requests.get(url)
elif response.status_code != 200:
     error_message = "Unexpected HTTP Response code: {}".format(response.status_code)
     raise Exception(error_message)
data = response.json()
```

Answer: D

Explanation:



```
response = requests.get(url)
if response.status_code == 429:
     backoff_seconds = int(response.headers['Retry-After'])
     sleep(backoff_seconds)
     response = requests.get(url)
elif response.status_code != 200:
     error_message = "Unexpected HTTP Response code: {}".format(response.status_code)
     raise Exception(error_message)
data = response.json()
```

Question: 54

An application uses OAuth to get access to several API resources on behalf of an end user.
What are two valid parameters to send to the authorization server as part of the first step of an authorization code grant flow? (Choose two.)

A.URI to which the authorization server will send the user-agent back when access is granted or denied

B.list of the API resources that the application is requesting to access

C.secret that was generated by the authorization server when the application registered as an OAuth integration

D.list of scopes that correspond to the API resources to which the application is requesting to access

E.name of the application under which the application registered as an OAuth integration

Answer: AD

Explanation:

Okay, let's break down why options A and D are the correct parameters in the first step of an OAuth 2.0 authorization code grant flow, and why the others are incorrect.

The authorization code grant flow is a standard OAuth mechanism where an application requests authorization from a user to access their resources. The first step involves redirecting the user's browser to the authorization server. This request includes specific parameters, two of which are critical:

A. URI to which the authorization server will send the user-agent back when access is granted or denied (redirect_uri): This is the callback URL where the authorization server will redirect the user's browser after the user has authenticated and potentially authorized the application. This is vital to return the user and the authorization code back to the application. Without this, the process breaks down. This URI must be pre-registered with the authorization server when the application is initially setup.

D. list of scopes that correspond to the API resources to which the application is requesting to access (scope): The scope parameter defines the permissions the application is requesting from the user. Instead of blindly requesting full access, an application uses specific scope values to ask for access to only the resources it needs (principle of least privilege). For example, a calendar app might request a "calendar.read" scope, and a contact app might request "contacts.read" and "contacts.write".

Here's why the other options are incorrect:

B. list of the API resources that the application is requesting to access: While the idea of identifying resources is related, OAuth uses scopes to control access to specific actions or groups of resources, rather than directly listing the resources themselves.

C. secret that was generated by the authorization server when the application registered as an OAuth integration (client_secret): The client secret is used later in the authorization code flow during the exchange of the authorization code for an access token. It is used for authentication of the application with the authorization server, not as part of the initial redirection.

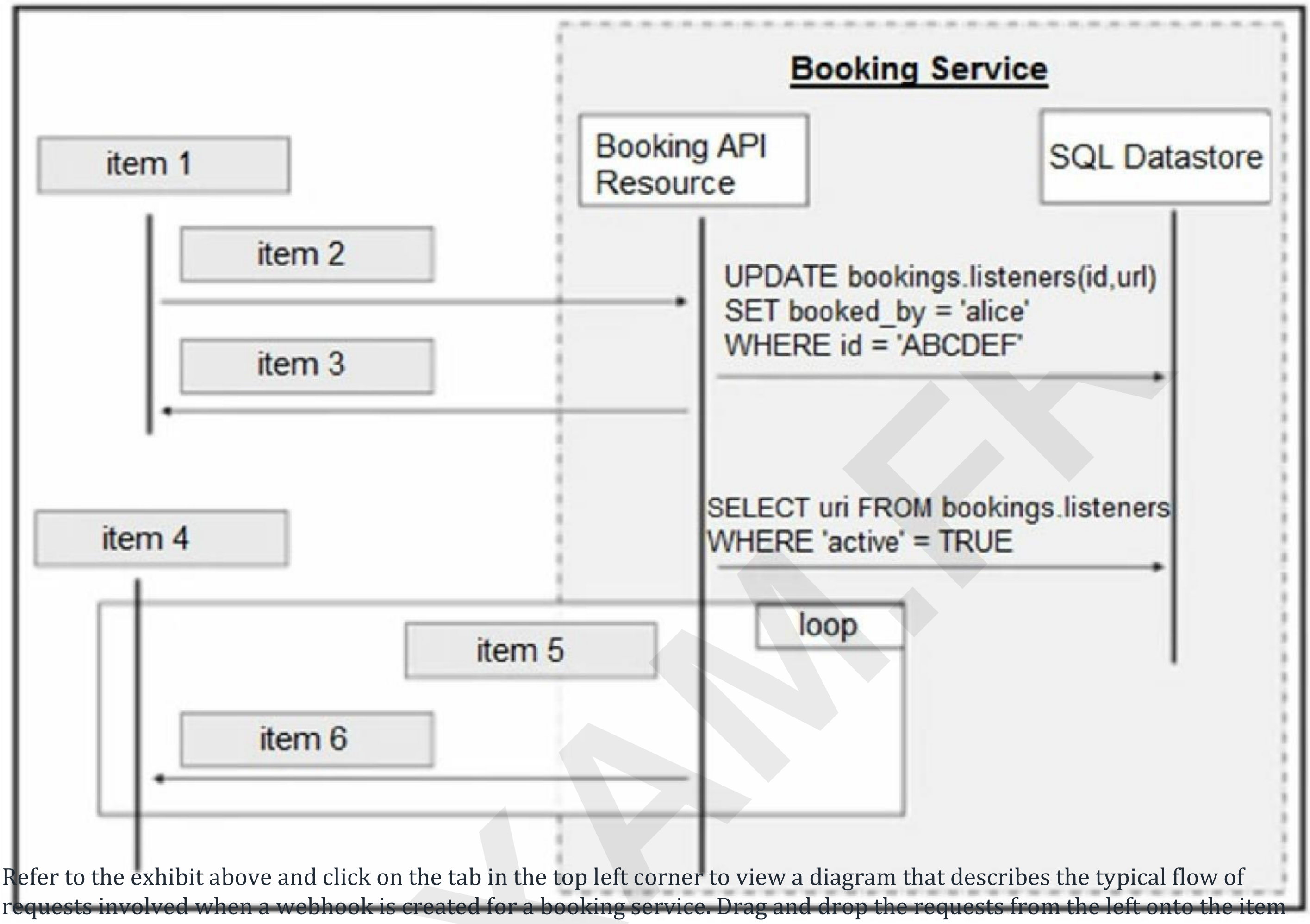
In summary, for the initial authorization request in the OAuth 2.0 code grant flow, the redirect_uri and scope parameters are essential in directing the user to the appropriate authorization server and setting proper authorization boundaries respectively.

Authoritative Links for Further Research:

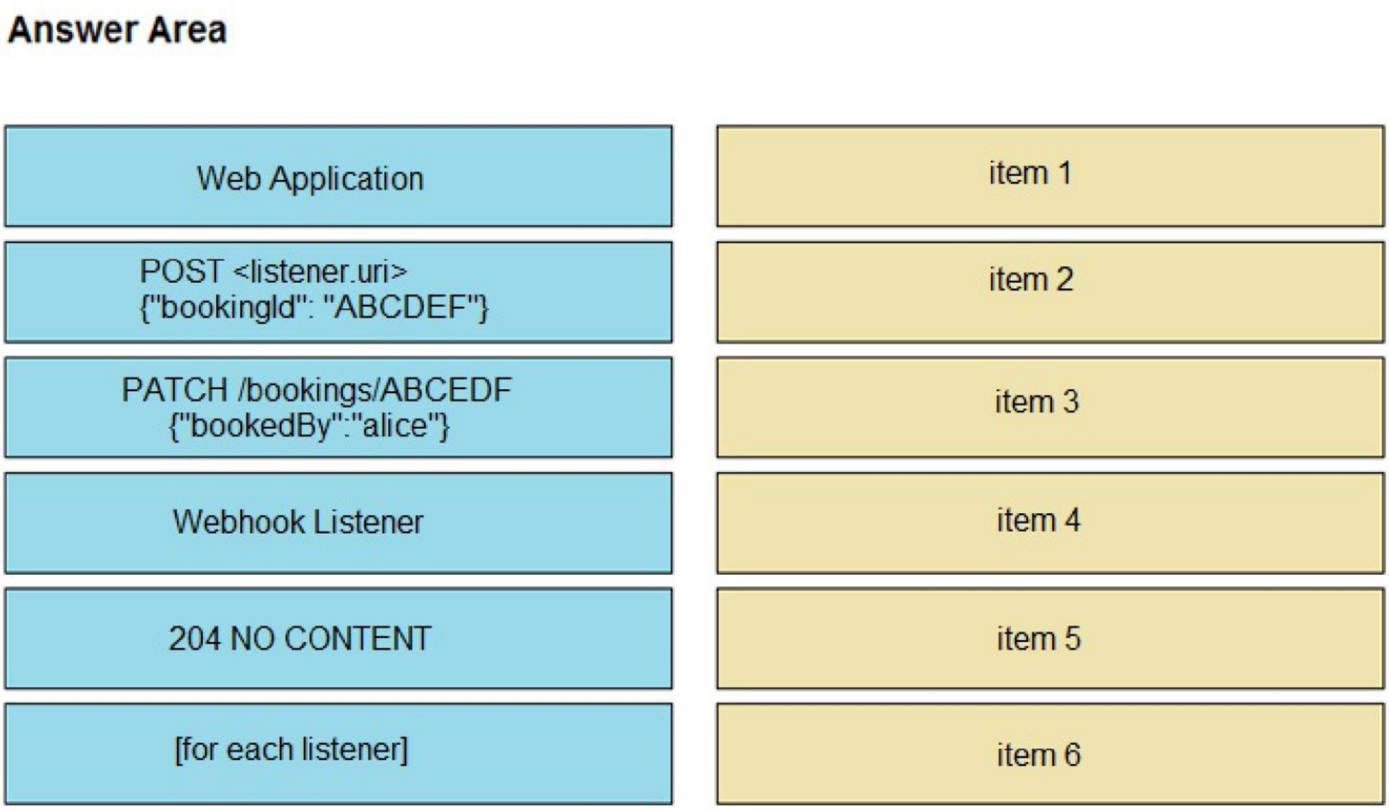
OAuth 2.0 RFC 6749: This is the core specification document for OAuth 2.0.
https://datatracker.ietf.org/doc/html/rfc6749
DigitalOcean - An Introduction to OAuth 2: A practical guide that provides a good overall explanation of OAuth 2.0. https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2
Auth0 - OAuth 2.0 Authorization Code Flow: Explains this specific flow in more detail.
https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow

Question: 55

DRAG DROP -



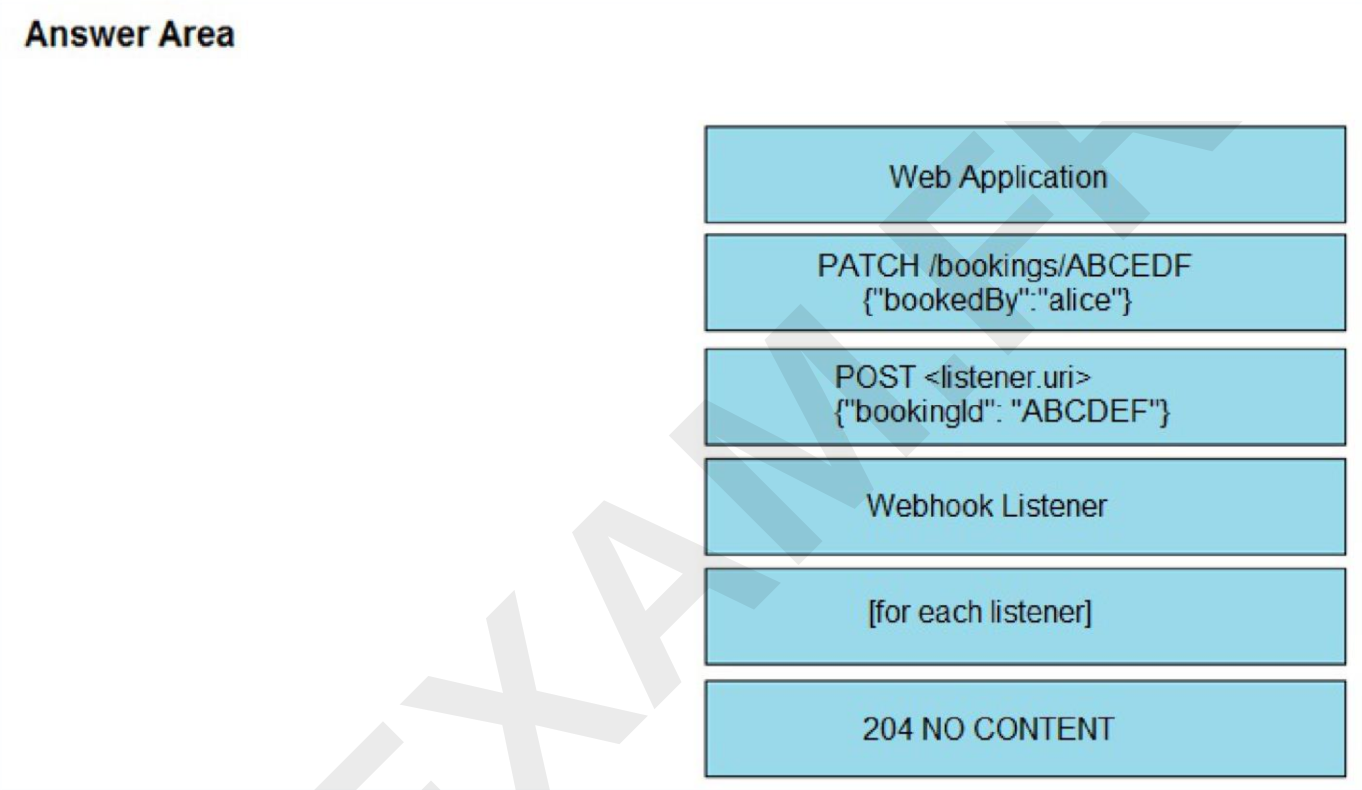
Refer to the exhibit above and click on the tab in the top left corner to view a diagram that describes the typical flow of requests involved when a webhook is created for a booking service. Drag and drop the requests from the left onto the item numbers on the right that match the missing sections in the sequence diagram to design the complete flow of requests involved as a booking is updated from a web application.
Select and Place:



Answer Area

Web Application	item 1
POST <listener.uri> {"bookingId": "ABCDEF"}	item 2
PATCH /bookings/ABCEDF {"bookedBy":"alice"}	item 3
Webhook Listener	item 4
204 NO CONTENT	item 5
[for each listener]	item 6

Answer:



Answer Area

Web Application

PATCH /bookings/ABCEDF
{"bookedBy":"alice"}

POST <listener.uri>
{"bookingId": "ABCDEF"}

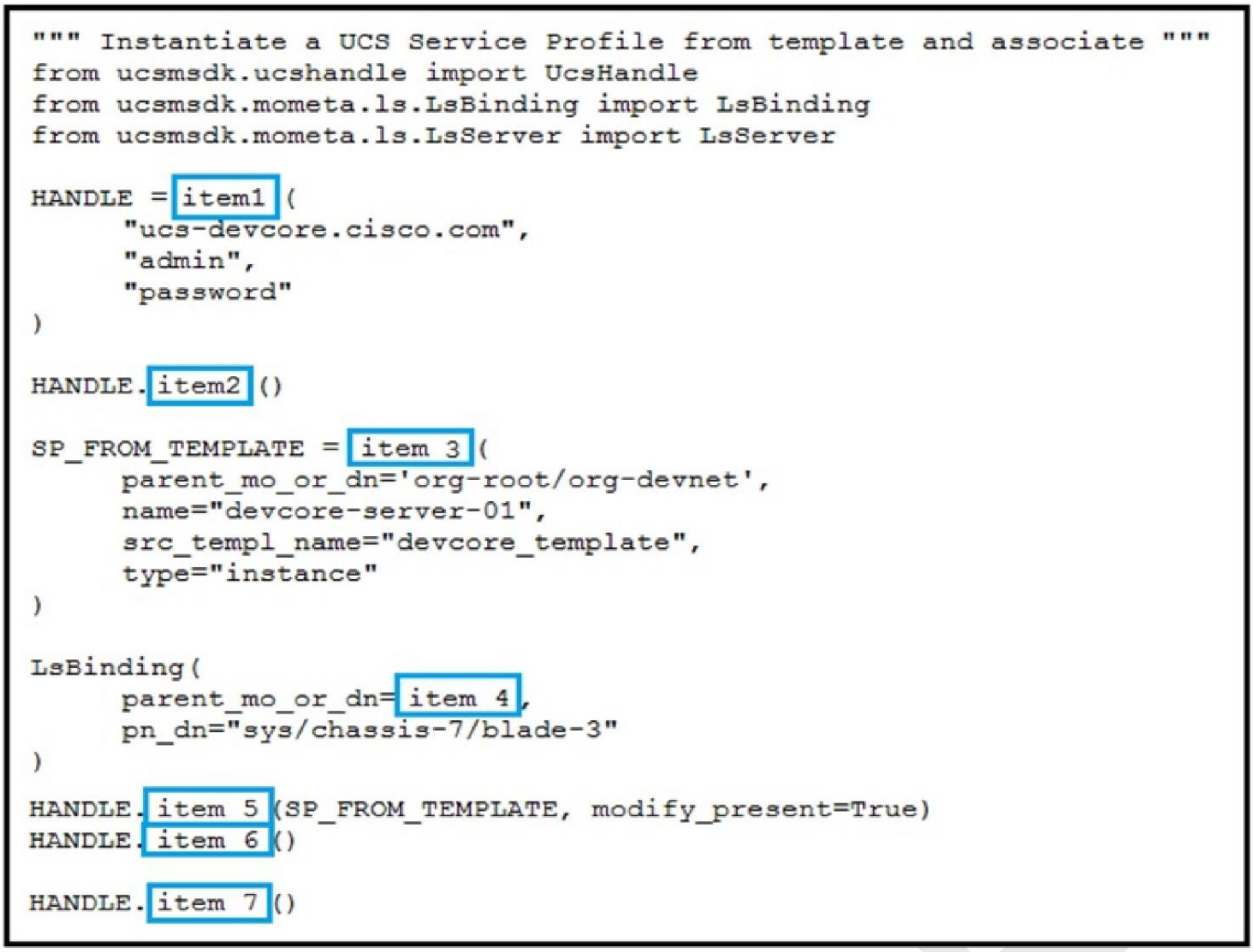
Webhook Listener

[for each listener]

204 NO CONTENT

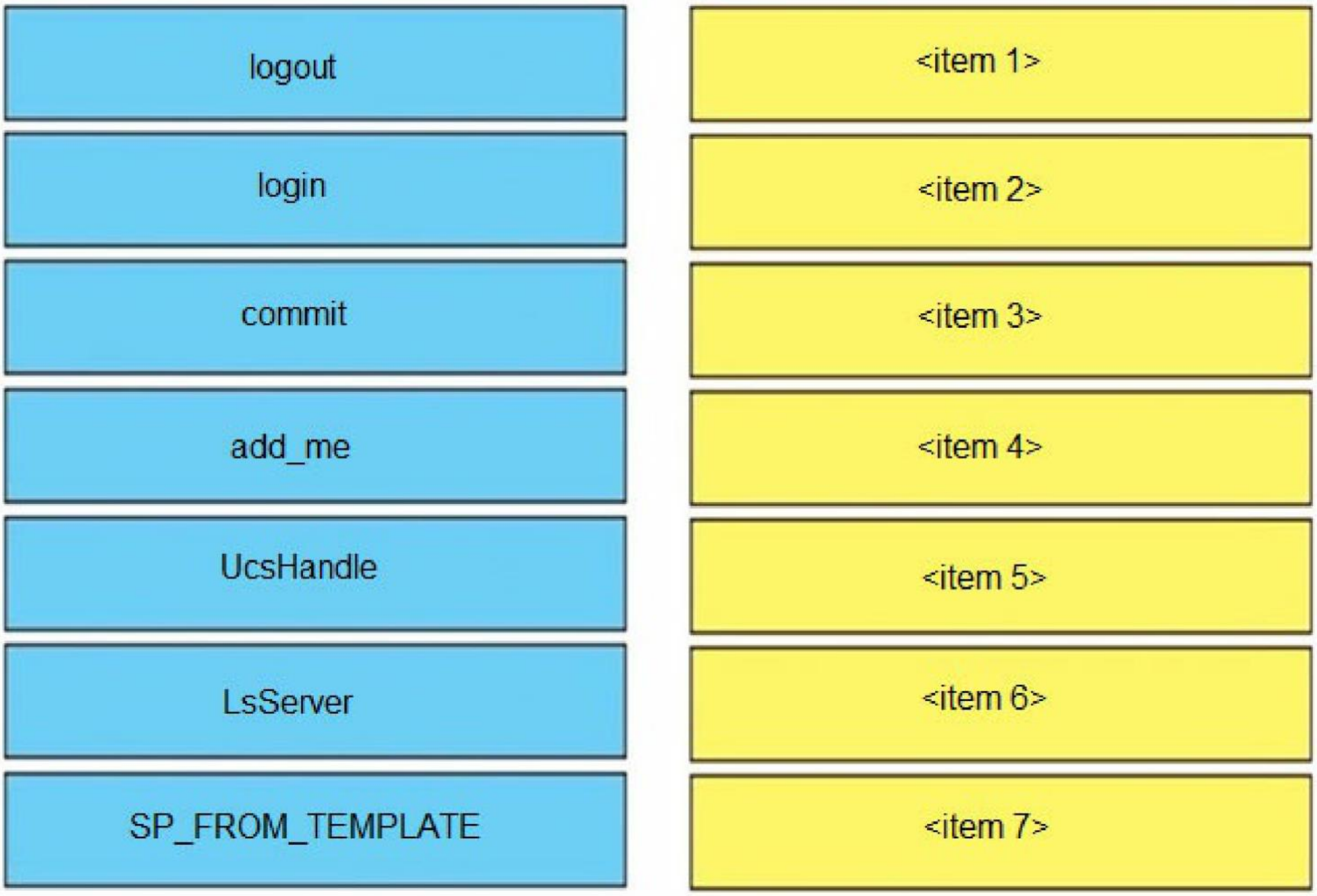
Question: 56

DRAG DROP -



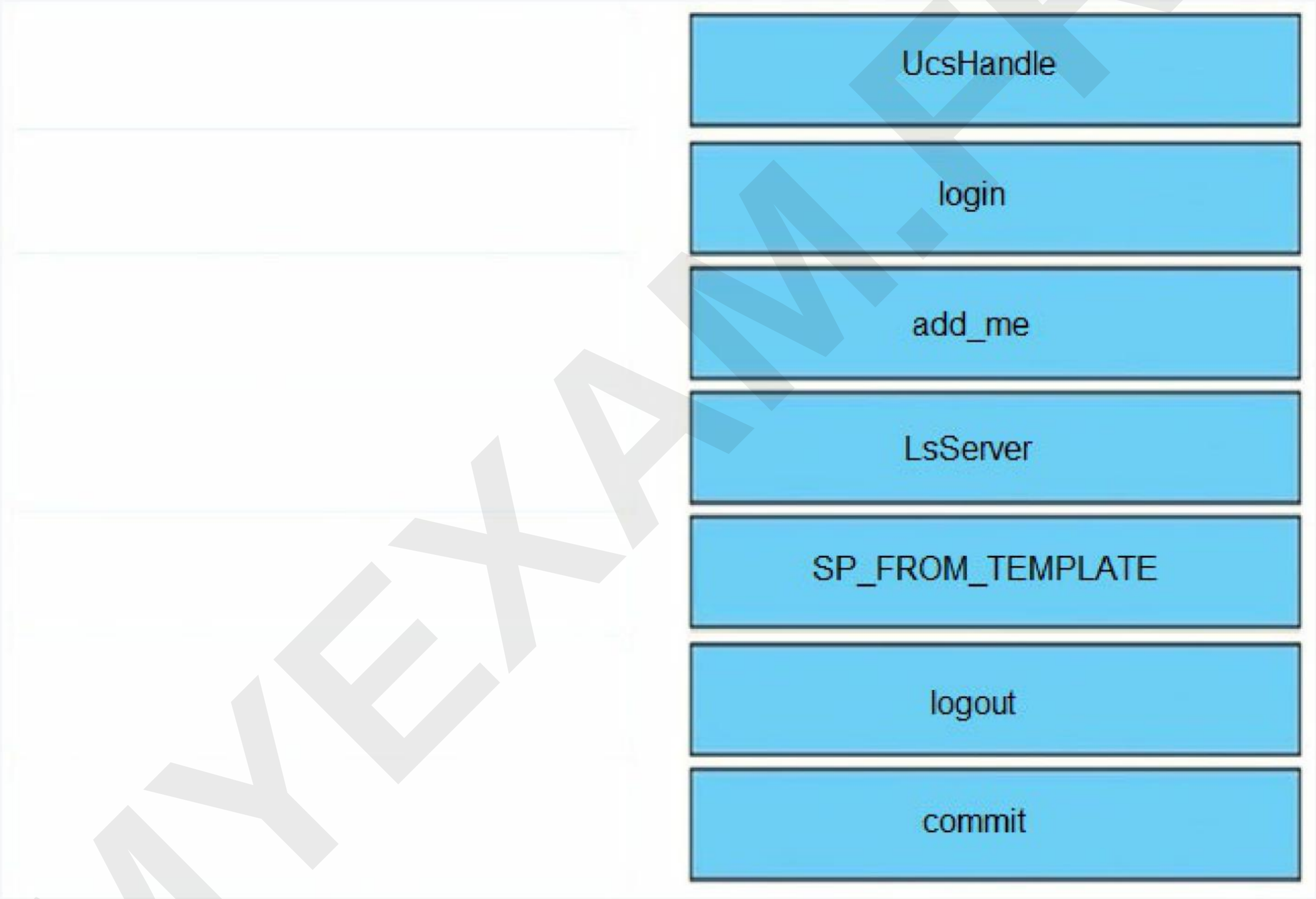
```
""" Instantiate a UCS Service Profile from template and associate """
from ucsmsdk.ucshandle import UcsHandle
from ucsmsdk.mometa.ls.LsBinding import LsBinding
from ucsmsdk.mometa.ls.LsServer import LsServer

HANDLE = item1 (
      "ucs-devcore.cisco.com",
      "admin",
      "password"
)

HANDLE. item2 ()

SP_FROM_TEMPLATE = item 3 (
      parent_mo_or_dn='org-root/org-devnet',
      name="devcore-server-01",
      src_templ_name="devcore_template",
      type="instance"
)

LsBinding(
      parent_mo_or_dn= item 4 ,
      pn_dn="sys/chassis-7/blade-3"
)

HANDLE. item 5 (SP_FROM_TEMPLATE, modify_present=True)
HANDLE. item 6 ()

HANDLE. item 7 ()
```

Refer to the exhibit above and click on the resource tabs in the top left corner to view resources to help with this question. Python code that uses the UCS Python
SDK is instantiating a service profile named `devcore-server-01` from service profile template `devcore_template`, then associating the service profile instance to blade 3 in chassis 7. Drag and drop the code snippets from the left onto the item numbers on the right that match the missing sections in the Python exhibit.
Select and Place:



logout	<item 1>
login	<item 2>
commit	<item 3>
add_me	<item 4>
UcsHandle	<item 5>
LsServer	<item 6>
SP_FROM_TEMPLATE	<item 7>

Answer:



UcsHandle

login

add_me

LsServer

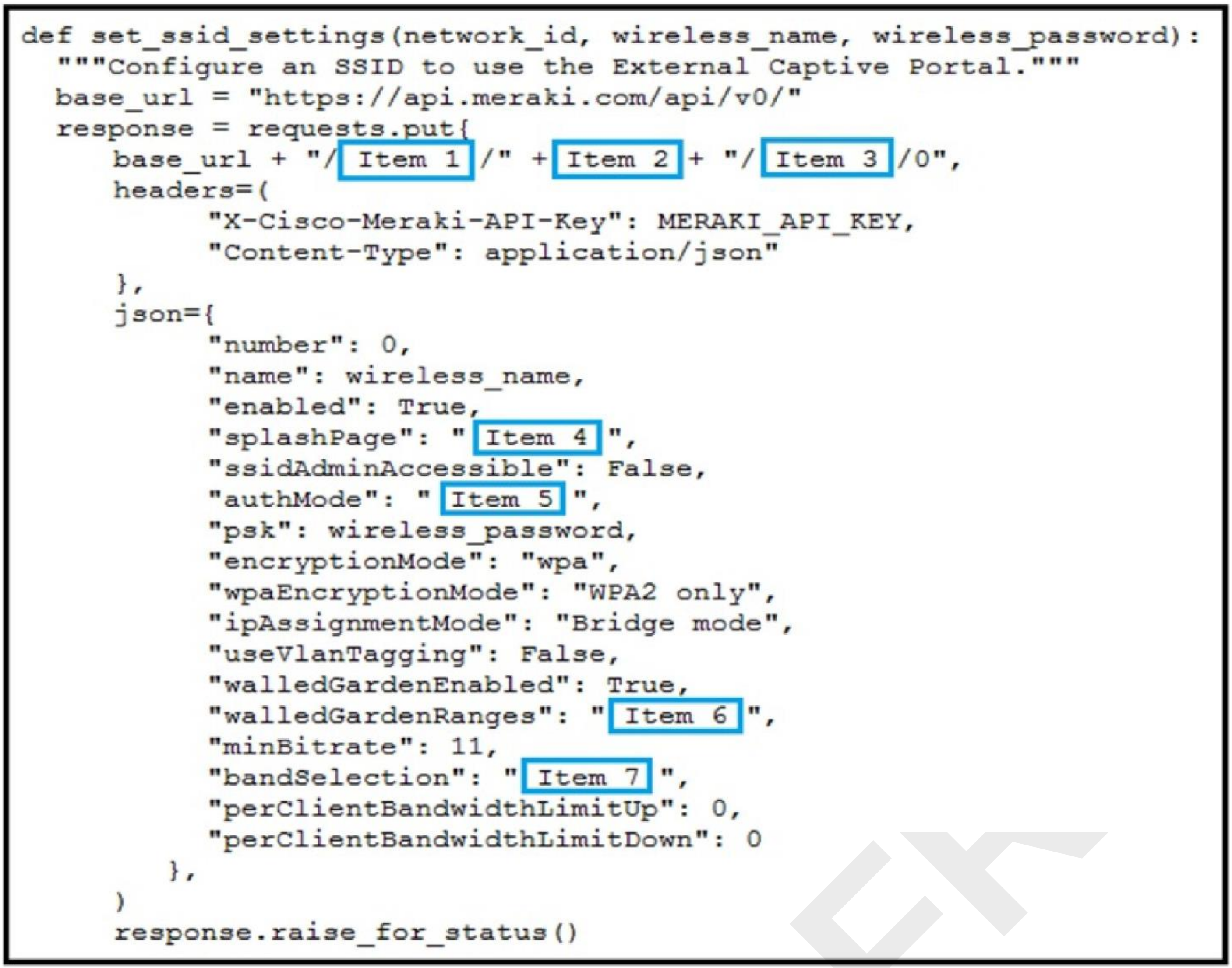
SP_FROM_TEMPLATE

logout

commit

Question: 57

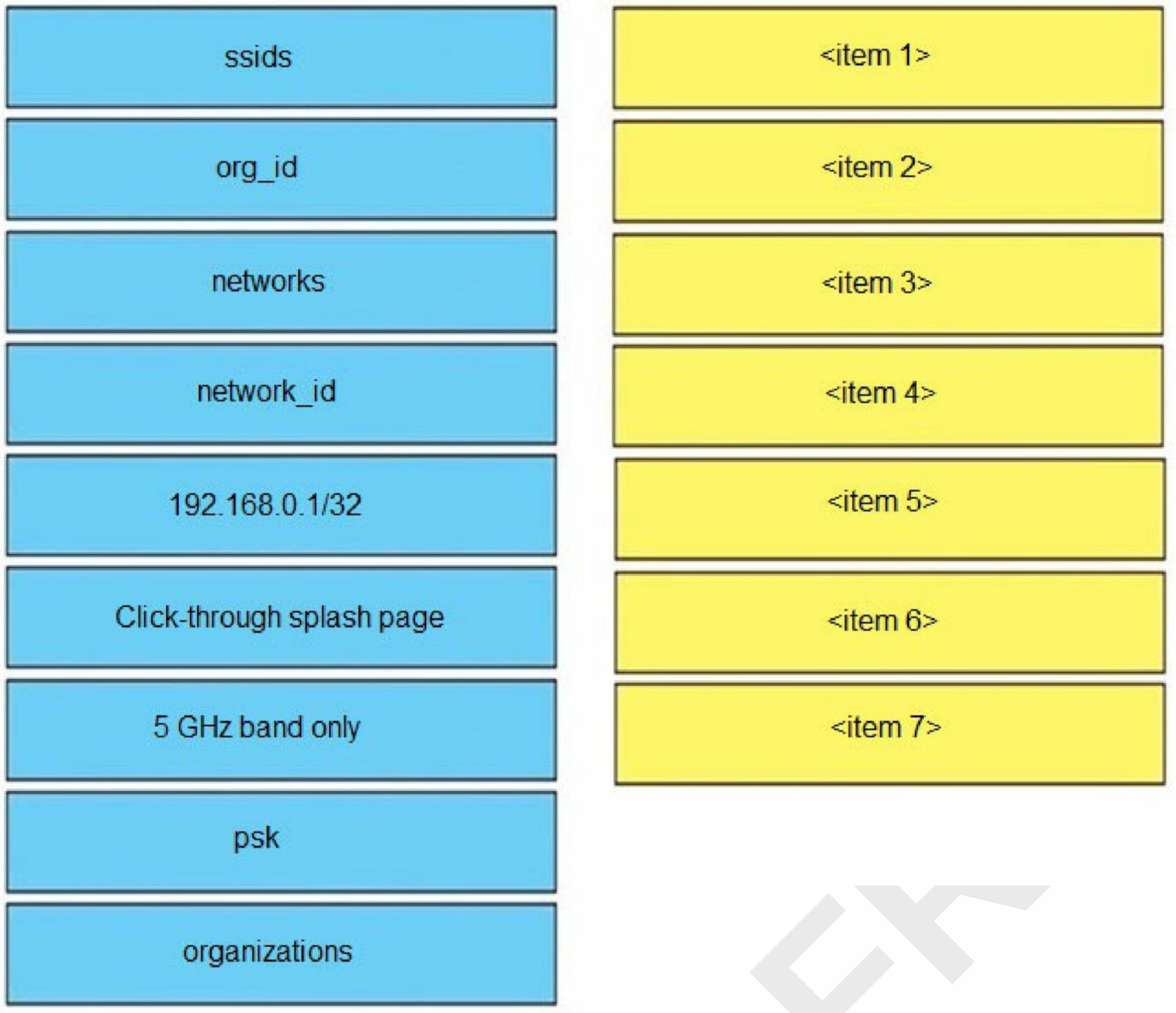
DRAG DROP -



```
def set_ssid_settings(network_id, wireless_name, wireless_password):
  """Configure an SSID to use the External Captive Portal."""
  base_url = "https://api.meraki.com/api/v0/"
  response = requests.put{
      base_url + "/ Item 1 /" + Item 2 + "/ Item 3 /0",
      headers=(
            "X-Cisco-Meraki-API-Key": MERAKI_API_KEY,
            "Content-Type": application/json"
      },
      json={
            "number": 0,
            "name": wireless_name,
            "enabled": True,
            "splashPage": " Item 4 ",
            "ssidAdminAccessible": False,
            "authMode": " Item 5 ",
            "psk": wireless_password,
            "encryptionMode": "wpa",
            "wpaEncryptionMode": "WPA2 only",
            "ipAssignmentMode": "Bridge mode",
            "useVlanTagging": False,
            "walledGardenEnabled": True,
            "walledGardenRanges": " Item 6 ",
            "minBitrate": 11,
            "bandSelection": " Item 7 ",
            "perClientBandwidthLimitUp": 0,
            "perClientBandwidthLimitDown": 0
         },
      )
      response.raise_for_status()
```

Refer to the exhibit above and click on the Meraki Resources tab in the top left corner to view Meraki documentation to help with this question. Drag and drop the parts of the Python code from the left onto the item numbers on the right that match the missing sections in the exhibit to enable an SSID. Not all code parts are used.

Select and Place:



ssids

org_id

networks

network_id

192.168.0.1/32

Click-through splash page

5 GHz band only

psk

organizations

<item 1>

<item 2>

<item 3>

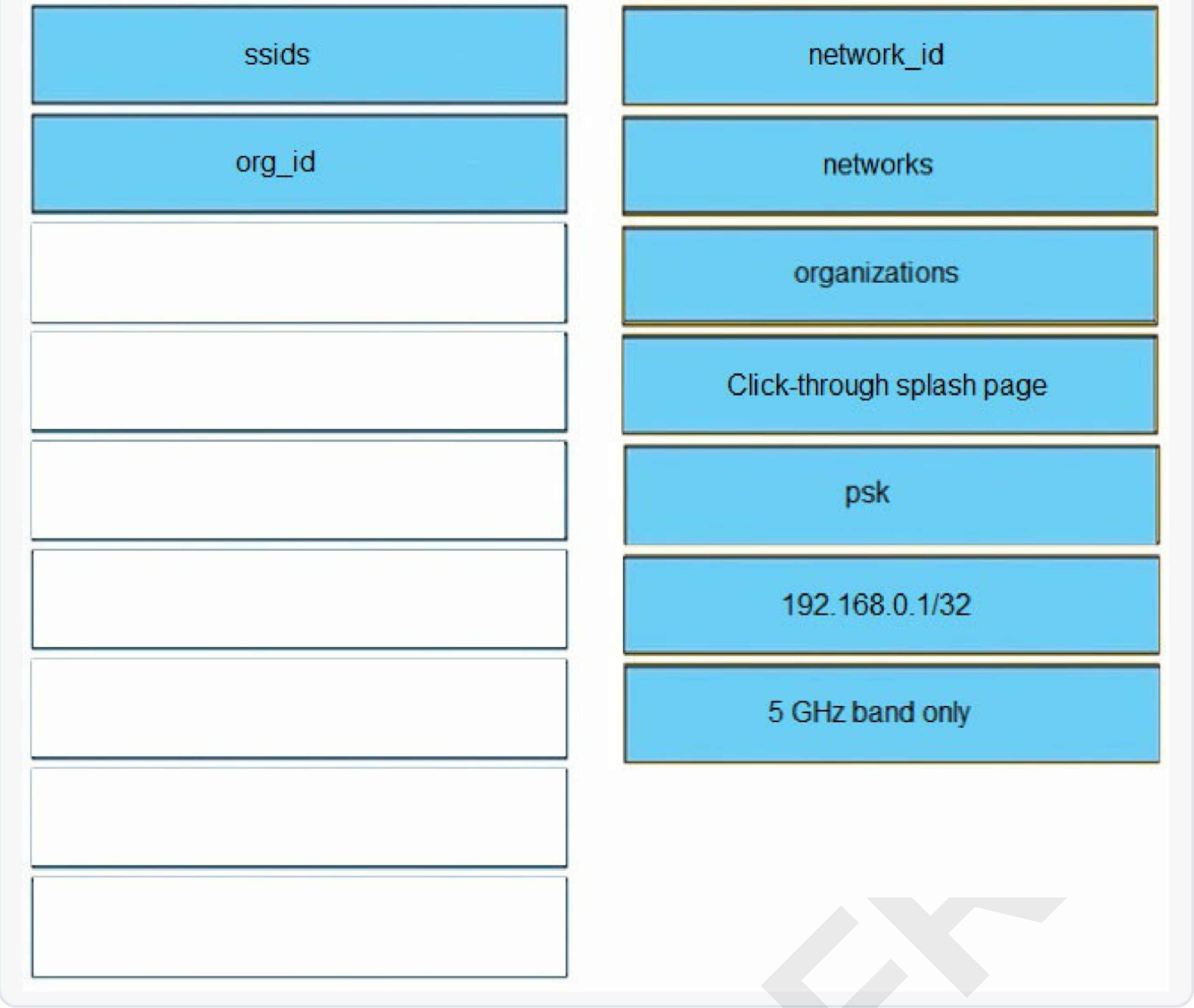
<item 4>

<item 5>

<item 6>

<item 7>

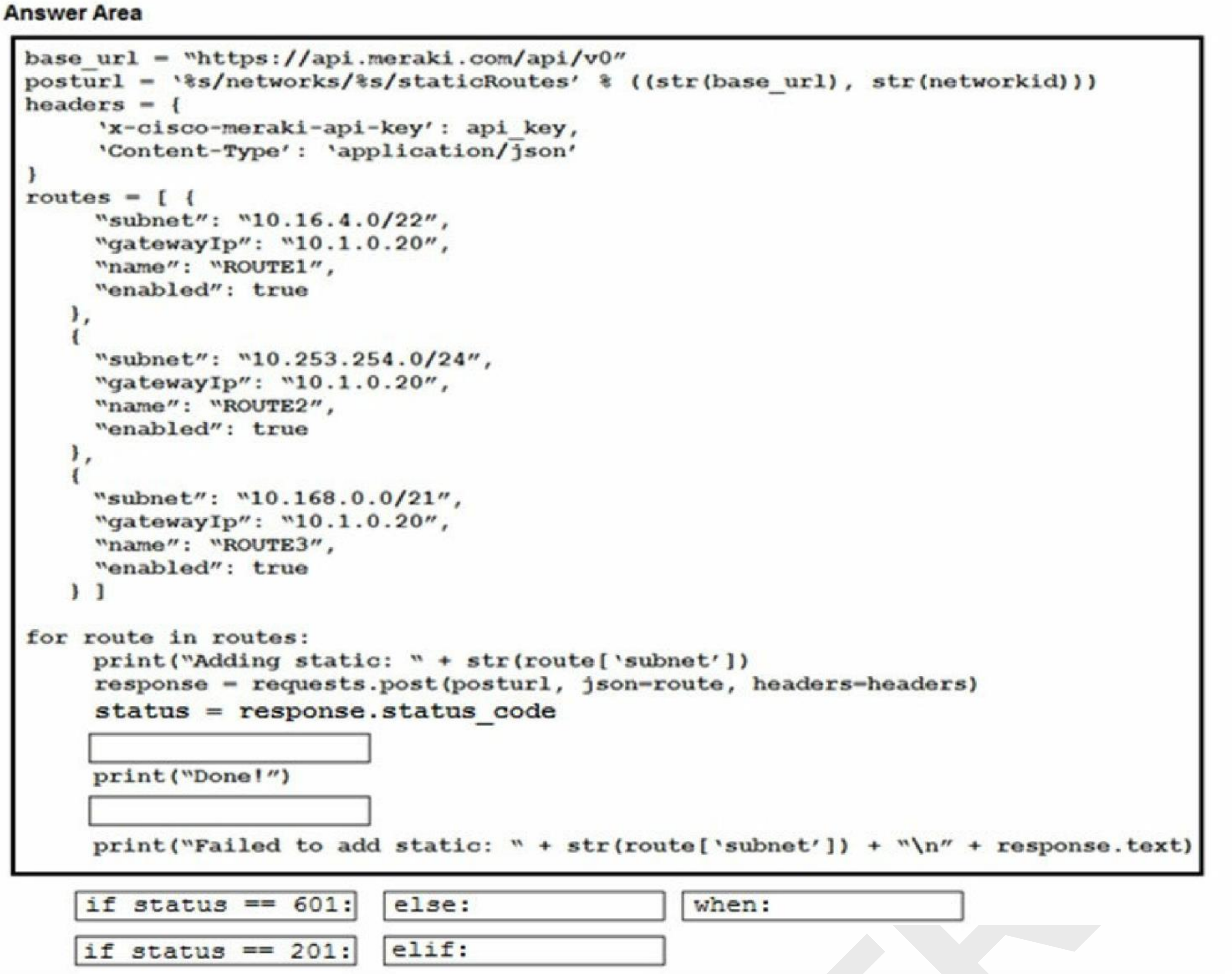
Answer:



Question: 58

DRAG DROP -
Drag and drop the expressions from below onto the code to implement error handling. Not all options are used.
Select and Place:

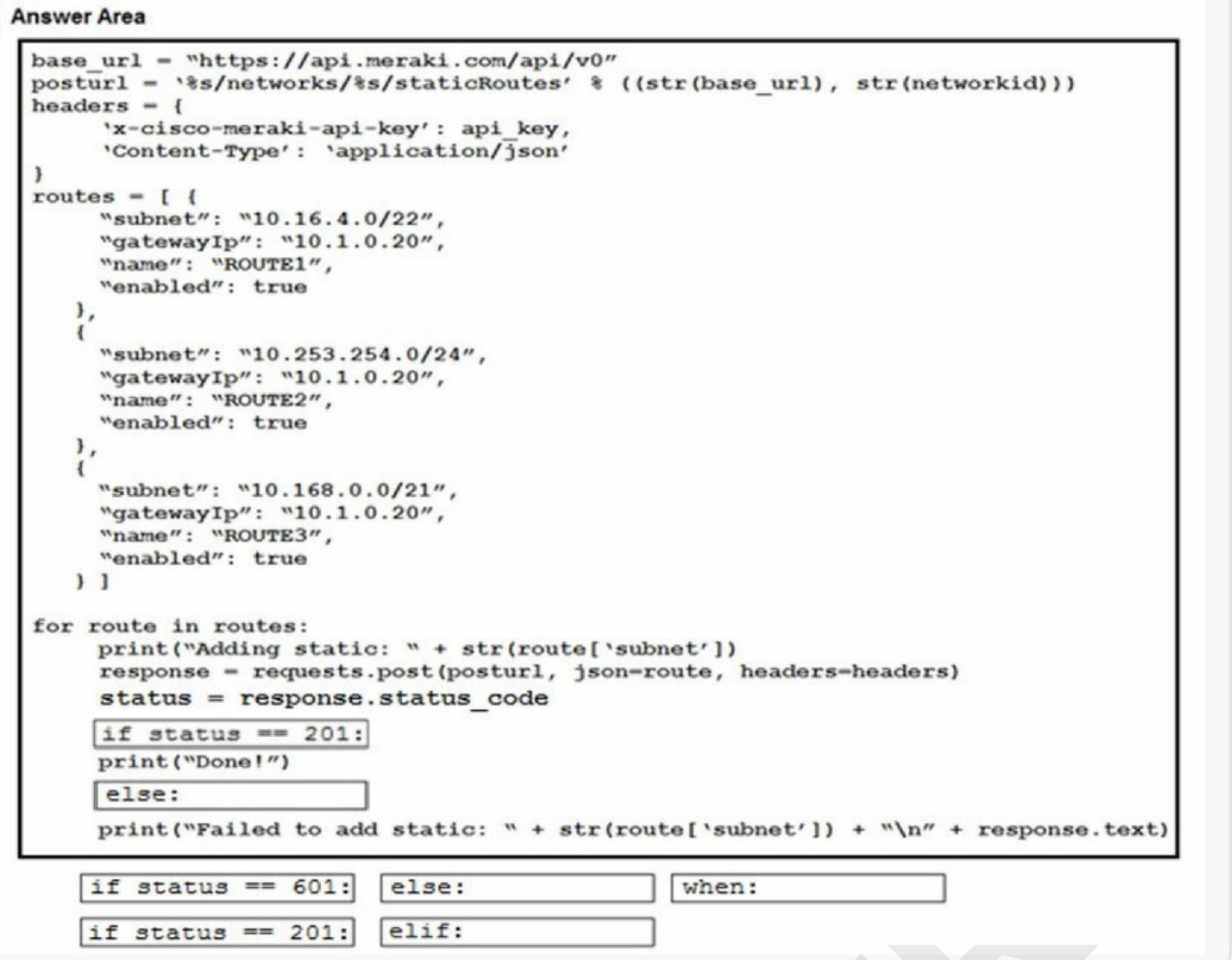
Answer Area



```
base_url = “https://api.meraki.com/api/v0”
posturl = ‘%s/networks/%s/staticRoutes’ % ((str(base_url), str(networkid))
headers = {
      ‘x-cisco-meraki-api-key’: api_key,
      ‘Content-Type’: ‘application/json’
}
routes = [ {
      “subnet”: “10.16.4.0/22”,
      “gatewayIp”: “10.1.0.20”,
      “name”: “ROUTE1”,
      “enabled”: true
   },
   {
      “subnet”: “10.253.254.0/24”,
      “gatewayIp”: “10.1.0.20”,
      “name”: “ROUTE2”,
      “enabled”: true
   },
   {
      “subnet”: “10.168.0.0/21”,
      “gatewayIp”: “10.1.0.20”,
      “name”: “ROUTE3”,
      “enabled”: true
   } ]

for route in routes:
      print(“Adding static: ” + str(route[‘subnet’])
      response = requests.post(posturl, json=route, headers=headers)
      status = response.status_code

      print(“Done!”)

      print(“Failed to add static: ” + str(route[‘subnet’]) + “\n” + response.text)
```

```
if status == 601:    else:    when:
if status == 201:    elif:
```

Answer:

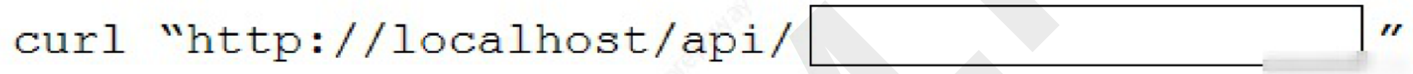


Answer Area

```
base_url = "https://api.meraki.com/api/v0"
posturl = '%s/networks/%s/staticRoutes' % ((str(base_url), str(networkid)))
headers = {
     'x-cisco-meraki-api-key': api_key,
     'Content-Type': 'application/json'
}
routes = [ {
     "subnet": "10.16.4.0/22",
     "gatewayIp": "10.1.0.20",
     "name": "ROUTE1",
     "enabled": true
   },
   {
     "subnet": "10.253.254.0/24",
     "gatewayIp": "10.1.0.20",
     "name": "ROUTE2",
     "enabled": true
   },
   {
     "subnet": "10.168.0.0/21",
     "gatewayIp": "10.1.0.20",
     "name": "ROUTE3",
     "enabled": true
   } ]

for route in routes:
     print("Adding static: " + str(route['subnet'])
     response = requests.post(posturl, json=route, headers=headers)
     status = response.status_code
     if status == 201:
     print("Done!")
     else:
     print("Failed to add static: " + str(route['subnet']) + "\n" + response.text)
```

```
if status == 601:    else:    when:
if status == 201:    elif:
```

Question: 59



```
curl "http://localhost/api/                "
```

Refer to the exhibit. An application's REST API GET call is tested to the inventory database.
Which missing code must be included to limit the number of values that are returned from the query to 20?

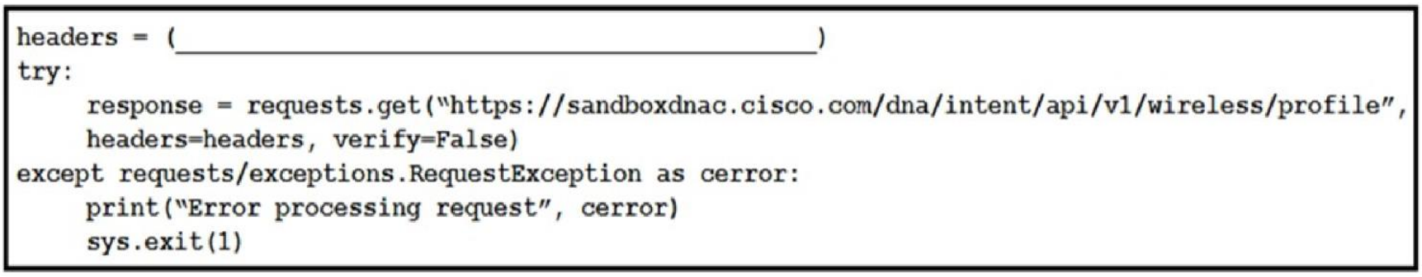
A.?inventory=20
B.inventory?limit=20
C.limit=?20
D.inventory=limit?20

Answer: B

Explanation:

The correct answer is Binventory?limit=20path = inventoryquery = ? offset or limit = limitequal = =query number = 20.

Question: 60



```
headers = (_____________________________________________)
try:
     response = requests.get("https://sandboxdnac.cisco.com/dna/intent/api/v1/wireless/profile",
     headers=headers, verify=False)
except requests/exceptions.RequestException as cerror:
     print("Error processing request", cerror)
     sys.exit(1)
```

Refer to the exhibit. Which code snippet is required in the headers to successfully authorize wireless information from Cisco DNA Center?

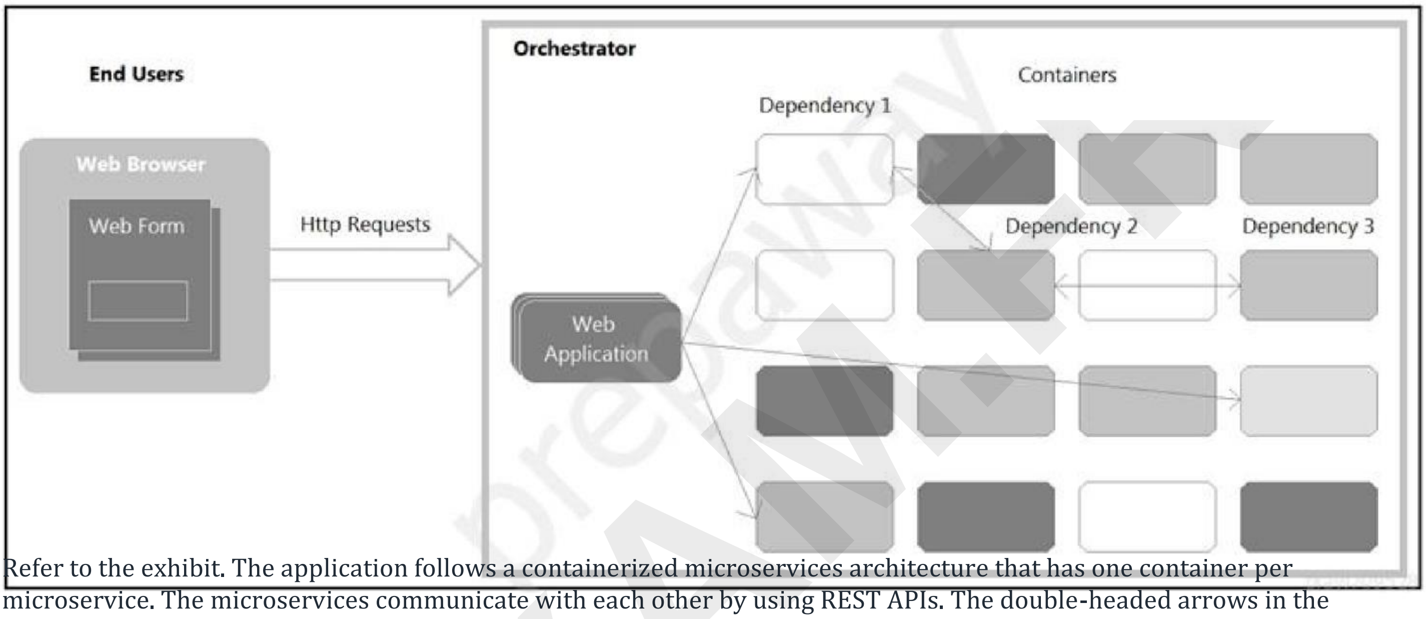
A.headers = "˜X-auth-token':'fa8426a0-8eaf-4d22-8e13-7c1b16a9370c'
B.headers = "˜Authorization':'Basic YWRtaW46R3JhcGV2aW5IMQ=='
C.headers = "˜Authorization':'Bearer ASDNFALKJER23412RKDALSNKF"
D.headers = "˜Content-type':'application/json

Answer: A

Explanation:

headers = "˜X-auth-token':'fa8426a0-8eaf-4d22-8e13-7c1b16a9370c' .

Question: 61



Refer to the exhibit. The application follows a containerized microservices architecture that has one container per microservice. The microservices communicate with each other by using REST APIs. The double-headed arrows in the diagram display chains of synchronous HTTP calls needed for a single user request.
Which action ensures the resilience of the application in the scope of a single user request?

A.Implement retries with exponential backoff during HTTP API calls.
B.Set up multiple instances of each microservice in active/active mode by using the Orchestrator.
C.Redesign the application to be separated into these three layers: Presentation, API, and Data. D.Create two virtual machines that each host an instance of the application and set up a cluster.

Answer: A

Explanation:

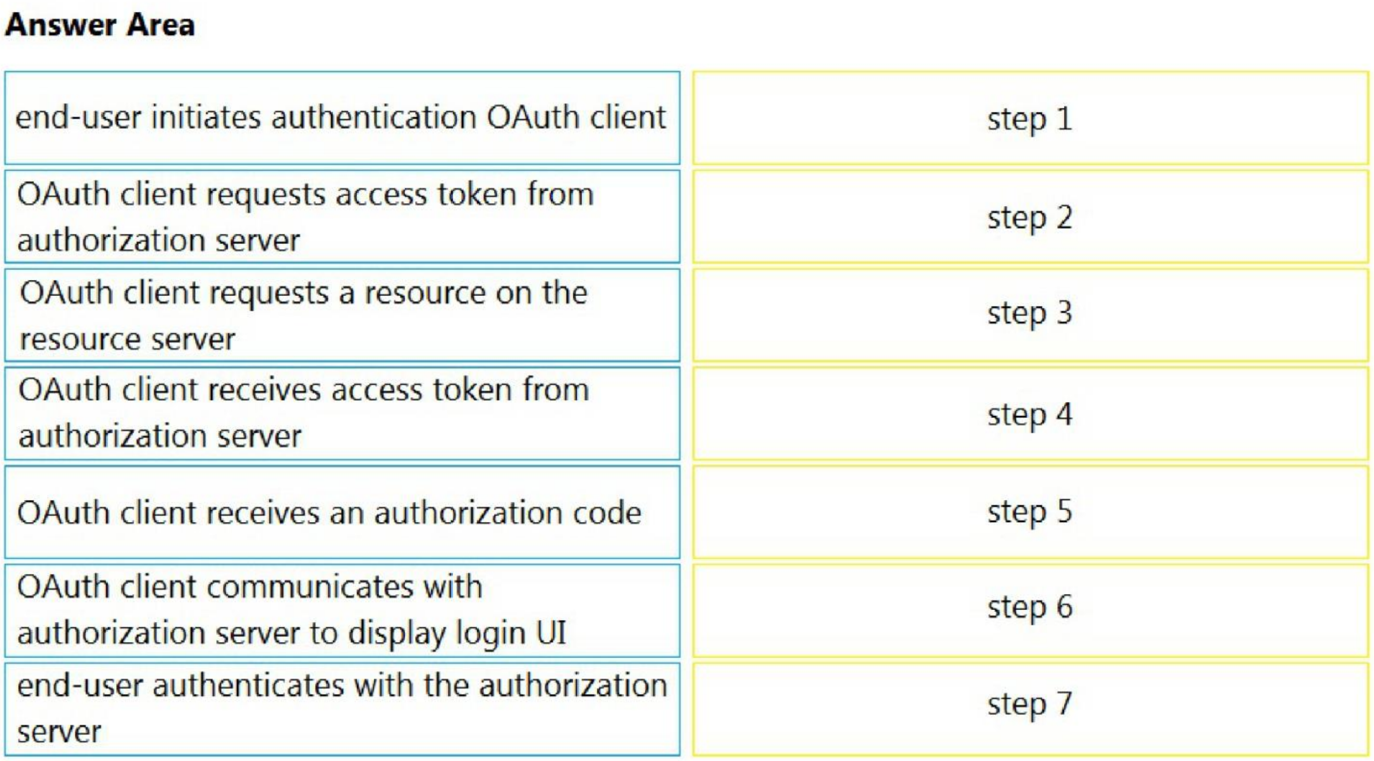
Implement retries with exponential backoff during HTTP API calls.

Question: 62

DRAG DROP -

Drag and drop the steps on the left into the order on the right for an end-user to access an OAuth2 protected resource using the 'Authorization Code Grant' flow.

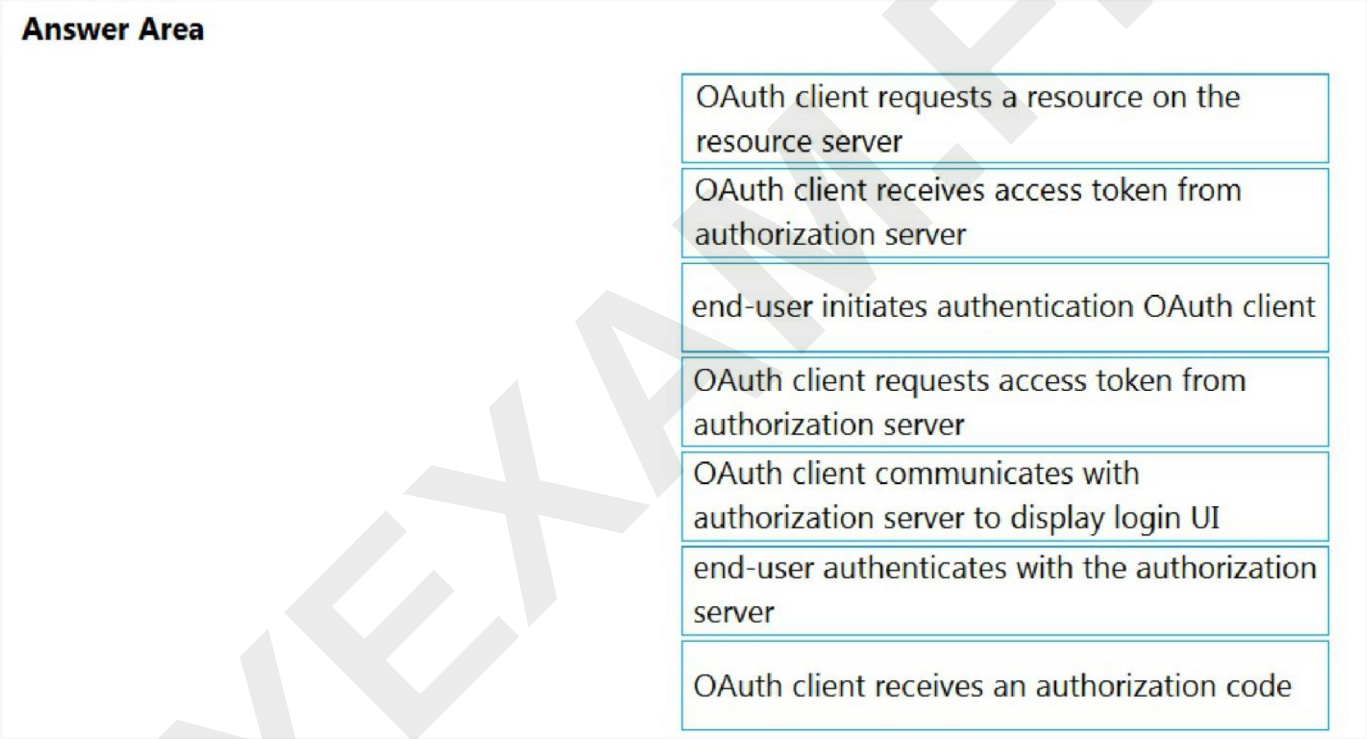
Select and Place:



Answer Area

end-user initiates authentication OAuth client	step 1
OAuth client requests access token from authorization server	step 2
OAuth client requests a resource on the resource server	step 3
OAuth client receives access token from authorization server	step 4
OAuth client receives an authorization code	step 5
OAuth client communicates with authorization server to display login UI	step 6
end-user authenticates with the authorization server	step 7

Answer:



Answer Area

OAuth client requests a resource on the resource server

OAuth client receives access token from authorization server

end-user initiates authentication OAuth client

OAuth client requests access token from authorization server

OAuth client communicates with authorization server to display login UI

end-user authenticates with the authorization server

OAuth client receives an authorization code

Question: 63

In the three-legged OAuth2 authorization workflow, which entity grants access to a protected resource?

A.resource owner

B.client

C.resource server

D.authorization server

Answer: A

Explanation:

The correct answer is A, the resource owner. In the three-legged OAuth2 authorization flow, the resource owner is the entity that ultimately grants permission for a client application to access their protected resources. This process revolves around delegation of authorization rather than direct sharing of credentials. The resource owner, typically a user, interacts with the authorization server to authenticate and then consents to grant specific access scopes to the client application. This consent is formalized through the issuance of an authorization grant, often a temporary code or token. The authorization server then provides the client with access tokens, but these tokens are solely the result of the resource owner's explicit permission. The resource server, which hosts the protected resources, trusts the authorization server's tokens, ensuring only clients with valid tokens granted by the resource owner can access those resources. The client application acts upon the tokens obtained to access the resources. Essentially, the resource owner maintains control and dictates which clients can access which resources, making them the grantor of access.

Here are some authoritative resources for further understanding of OAuth2:

1. **RFC 6749 - The OAuth 2.0 Authorization Framework:**https://datatracker.ietf.org/doc/html/rfc6749 (This is the foundational RFC document defining OAuth 2.0)
2. **OAuth 2.0: The Big Picture:**https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2 (A good explanation for beginners)
3. **Okta Developer Blog - What is OAuth 2.0?:**https://developer.okta.com/blog/2017/06/21/what-is-oauth-2-0 (An explanation from a developer's perspective)

Question: 64

What are two steps in the OAuth2 protocol flow? (Choose two.)

A.The user is authenticated by the authorization server and granted an access token.

B.The user's original credentials are validated by the resource server and authorization is granted.

C.The user indirectly requests authorization through the authorization server.

D.The user requests an access token by authentication and authorization grant presentation.

E.The user requests the protected resource from the resource server using the original credentials.

Answer: CE

Explanation:

Okay, let's break down why options C and E are correct steps in the OAuth 2.0 protocol flow, and why the others are not.

OAuth 2.0 is a standard authorization protocol that allows a user to grant third-party applications limited access to their resources without exposing their credentials. It involves multiple parties and a specific flow.

Option C accurately describes a key stage: "The user indirectly requests authorization through the authorization server." In OAuth 2.0, the client application (the app requesting access) doesn't directly ask the user for their credentials. Instead, it redirects the user to the authorization server (e.g., Google, Facebook), which handles the authentication process on behalf of the app. This indirect approach is central to OAuth 2.0's security model, protecting user credentials from the third-party app.

Option E, "The user requests the protected resource from the resource server using the original credentials," is incorrect. The user never uses their original credentials to access the resource directly after the initial authentication. Instead, once the authorization flow is complete, the client application uses an access token obtained earlier.

Option A, "The user is authenticated by the authorization server and granted an access token," is part of the process but isn't a 'step' in the initial flow. The server does authenticate the user, and it ultimately grants an access token. However, this happens after the user has been redirected and given consent. Option A lacks the redirection.

Option B, "The user's original credentials are validated by the resource server and authorization is granted," is incorrect. Resource servers do not validate the user's credentials. They only validate access tokens. This is a critical aspect of the OAuth architecture as it separates the responsibilities of authentication (authorization server) and resource access (resource server).

Option D, "The user requests an access token by authentication and authorization grant presentation," is an inaccurate summation. The client application requests the access token after user authorization, presenting the authorization grant received from the authorization server, not the user.

Therefore, the core steps of the initial OAuth 2.0 interaction, are C, the client redirects the user to the authorization server, and E, after the access token is acquired, the client requests the protected resource from the resource server using the token. Options A, B, and D inaccurately describe the flow or misplace the actor in the steps.

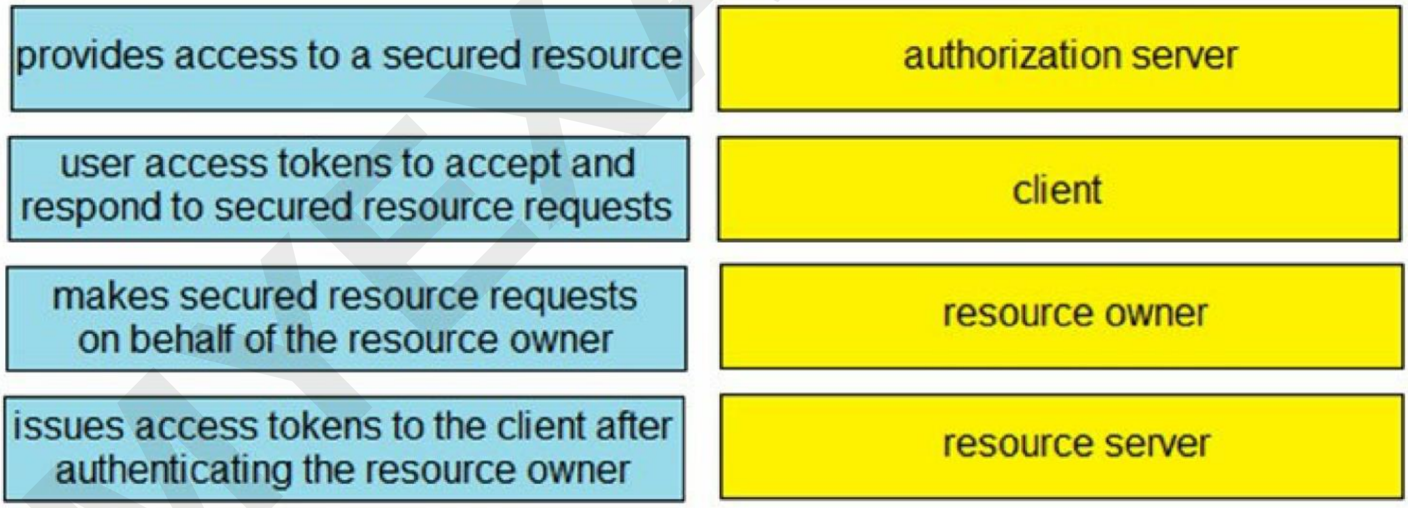
For further information, you can review the official OAuth 2.0 specification and documentation:

RFC 6749 - The OAuth 2.0 Authorization Framework:https://datatracker.ietf.org/doc/html/rfc6749
OAuth 2.0 Overview:https://oauth.net/2/

These resources will give you a deep understanding of the OAuth 2.0 protocol and its various flows and components.

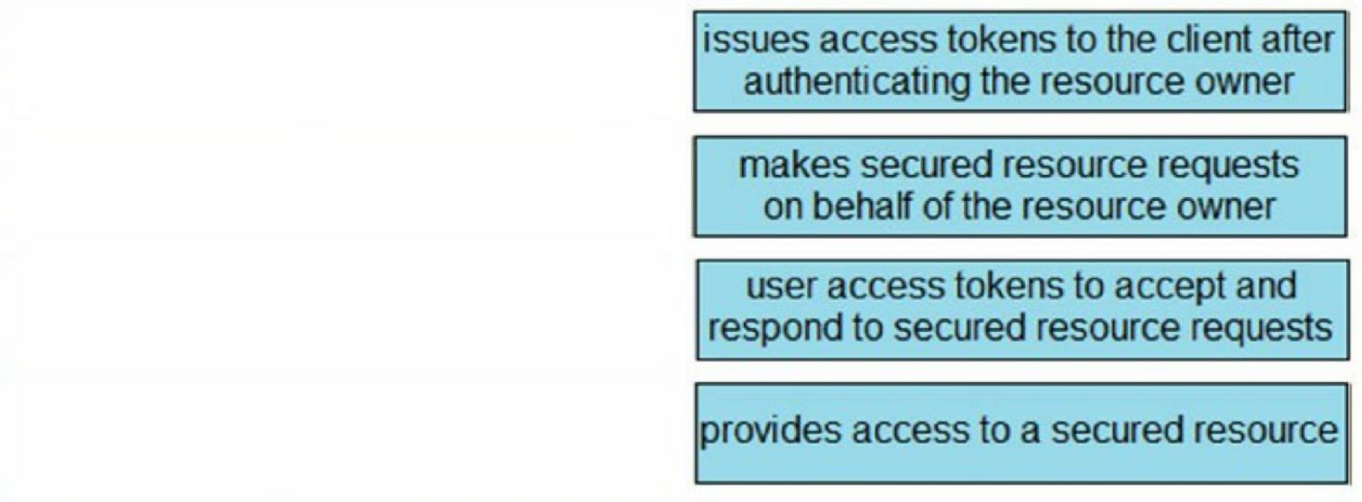
Question: 65

DRAG DROP -
Drag and drop the descriptions from the left onto the related OAuth-defined roles on the right.
Select and Place:



provides access to a secured resource	authorization server
user access tokens to accept and respond to secured resource requests	client
makes secured resource requests on behalf of the resource owner	resource owner
issues access tokens to the client after authenticating the resource owner	resource server

Answer:

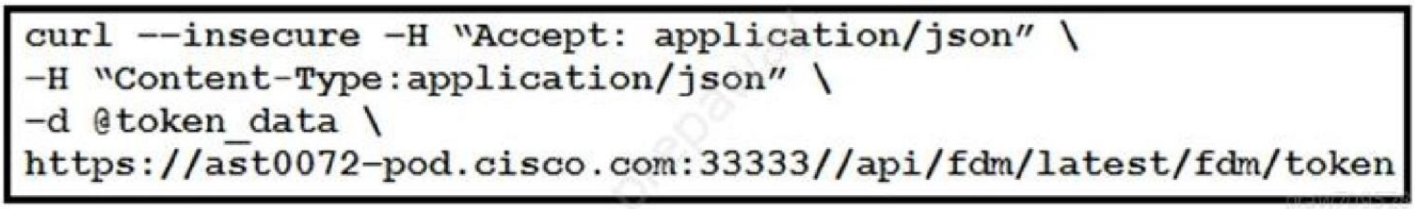

issues access tokens to the client after authenticating the resource owner

makes secured resource requests on behalf of the resource owner

user access tokens to accept and respond to secured resource requests

provides access to a secured resource

Question: 66



```
curl --insecure -H "Accept: application/json" \
-H "Content-Type:application/json" \
-d @token_data \
https://ast0072-pod.cisco.com:33333//api/fdm/latest/fdm/token
```

Refer to the exhibit. This cURL POST request creates an OAuth access token for authentication with FDM API requests. What is the purpose of the file `@token_data` that cURL is handling?

A.This file is given as input to store the access token received from FDM. B.This file is used to send authentication-related headers.

C.This file contains raw data that is needed for token authentication. D.This file is a container to log possible error responses in the request.

Answer: C

Explanation:

This file contains raw data that is needed for token authentication.

Question: 67

What is the result of a successful OAuth2 authorization grant flow?

A.The third-party service is provided with a token that allows actions to be performed.

B.The user has the application rights that correspond to the user's role within the application's database. C.The application is provided with a token that allows actions on services on the user's behalf.

D.The user has administrative rights to the application's backend services.

Answer: C

Explanation:

The correct answer is **C. The application is provided with a token that allows actions on services on the user's behalf.**

OAuth2 is an authorization protocol designed to grant limited access to resources without sharing user credentials directly. The successful authorization grant flow, a core mechanism of OAuth2, concludes with the client application receiving an access token. This token is not directly linked to the user's role within the application's database (as suggested in B). Instead, it empowers the application to interact with protected resources on the user's behalf, such as accessing APIs. Option A is incorrect because the access token is provided to the requesting application, not the service being accessed. It is a client application that needs the token to gain access and perform actions on the user's behalf, and the service that is providing access to resources does not get an access token. Option D is not relevant to the OAuth2 flow because the protocol does not involve granting any administrative privileges to the application. The access token embodies the permissions granted by the user during the authorization process, and it typically has a limited lifespan, enhancing security. The user doesn't gain any rights within the application using OAuth2, rather the application receives the necessary permission to act on behalf of the user. This token enables API calls without exposing user credentials directly, supporting a secure and delegated authorization model.

Here are some authoritative links for further reading:

RFC 6749 - The OAuth 2.0 Authorization Framework:https://datatracker.ietf.org/doc/html/rfc6749 (This is the official specification)
DigitalOcean Tutorial on OAuth 2.0:https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2 (A good explanation of the concepts)
Okta's explanation on OAuth 2.0:https://www.okta.com/identity-101/oauth-2-0/ (Provides a clear and business-focused explanation)

Question: 68

In the three-legged OAuth2 process, after the authorization server presents a form to the resource owner to grant access, what is the next step?

A.The resource owner authenticates and optionally authorizes with the authorization server.

B.The user who owns the resource initiates a request to the OAuth client.

C.If the resource owner allows access, the authorization server sends the OAuth client a redirection.

D.A form to allow or restrict access is submitted by the owner of the resource.

Answer: D

Explanation:

The correct answer is **D. A form to allow or restrict access is submitted by the owner of the resource.** Here's why:

The three-legged OAuth 2.0 flow involves a series of interactions between the client application, the resource owner (user), and the authorization server. After the authorization server presents the consent form to the resource owner, this form explicitly seeks the resource owner's permission. The next logical step is the resource owner actively interacting with that form.

Option A is incorrect as the resource owner has already authenticated before being presented with the authorization form. Option B is premature; the user action here is related to providing consent, not the initial request to the client. Option C is the subsequent step, happening after the resource owner interacts with the form.

Therefore, after the form is presented, the resource owner reviews the requested permissions. They then explicitly submit the form, choosing to grant or deny the client application access to the protected resource. This action is fundamental to the OAuth 2.0 principle of user-controlled access. Upon submission, the server proceeds with generating the necessary tokens based on the granted access, which would be followed by the redirection in C, but not before D.

The choice is consistent with OAuth 2.0's design focusing on delegating authorization, where resource owners are in control of what data they share. This step is vital for security and privacy. The resource owner is not automatically granting access; there is explicit consent required, enforced via this form submission.

For further research, refer to:

OAuth 2.0 Specification:https://oauth.net/2/ - The official OAuth 2.0 specification provides a detailed understanding of the protocol.

RFC 6749 (OAuth 2.0):https://datatracker.ietf.org/doc/html/rfc6749 - The formal document outlining OAuth 2.0.

DigitalOcean's Understanding OAuth 2.0:https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2 - Provides a comprehensive guide with illustrative examples.

Question: 69

Which OAuth mechanism enables clients to continue to have an active access token without further interaction from the user?

A.refresh grant
B.JWT
C.password grant
D.preshared key

Answer: A

Explanation:

The correct answer is **A. refresh grant**.

OAuth 2.0 provides several grant types for obtaining access tokens. Among them, the refresh grant is specifically designed to address the scenario where a client needs to maintain access without repeatedly involving the user. When an application initially obtains an access token, it often receives a refresh token alongside it. Access tokens have a short lifespan for security purposes. When the access token expires, the client can use the refresh token to request a new access token without requiring the user to re-authenticate.

This eliminates the need for the user to interactively log in or provide credentials each time access expires.

The process avoids interruption and ensures seamless application functionality. The refresh token acts as a long-lived credential solely used for obtaining new access tokens, which allows the refresh token to be stored securely while also allowing the application to have continued access. JWT (JSON Web Token) is a token format and not an OAuth grant type. Password grants are generally discouraged due to security risks. Pre-shared keys are not a mechanism used in OAuth authentication. Therefore, the refresh grant is the only mechanism among the options that allows for continuous access token renewal without user interaction. This is crucial for maintaining application sessions and providing a smooth user experience.

Authoritative Links:

RFC 6749 - The OAuth 2.0 Authorization Framework:https://datatracker.ietf.org/doc/html/rfc6749#section-6 (Section 6 details the different OAuth grant types, including the refresh token grant).

Auth0: Refresh Tokens:https://auth0.com/docs/tokens/concepts/refresh-tokens (Explains refresh token concepts and usage)

Okta: OAuth 2.0 and OpenID Connect Overview:https://developer.okta.com/docs/concepts/auth-overview/
(Provides a broad overview of OAuth 2.0, including the different grant types).

Question: 70

A developer is building an application to access a website. When running the application, an HTTP 403 error code has been received.
How should the application be modified to handle this code?

A.Create a loop on cancel the operation and run a new one after the code is received.

B.Use exponential backoff when retrying distributed services and other remote endpoints.

C.Build a try/except around the urlopen to find errors occurring in the request.

D.Redirect the request to an internal web server and make a new request from the internal resource.

Answer: C

Explanation:

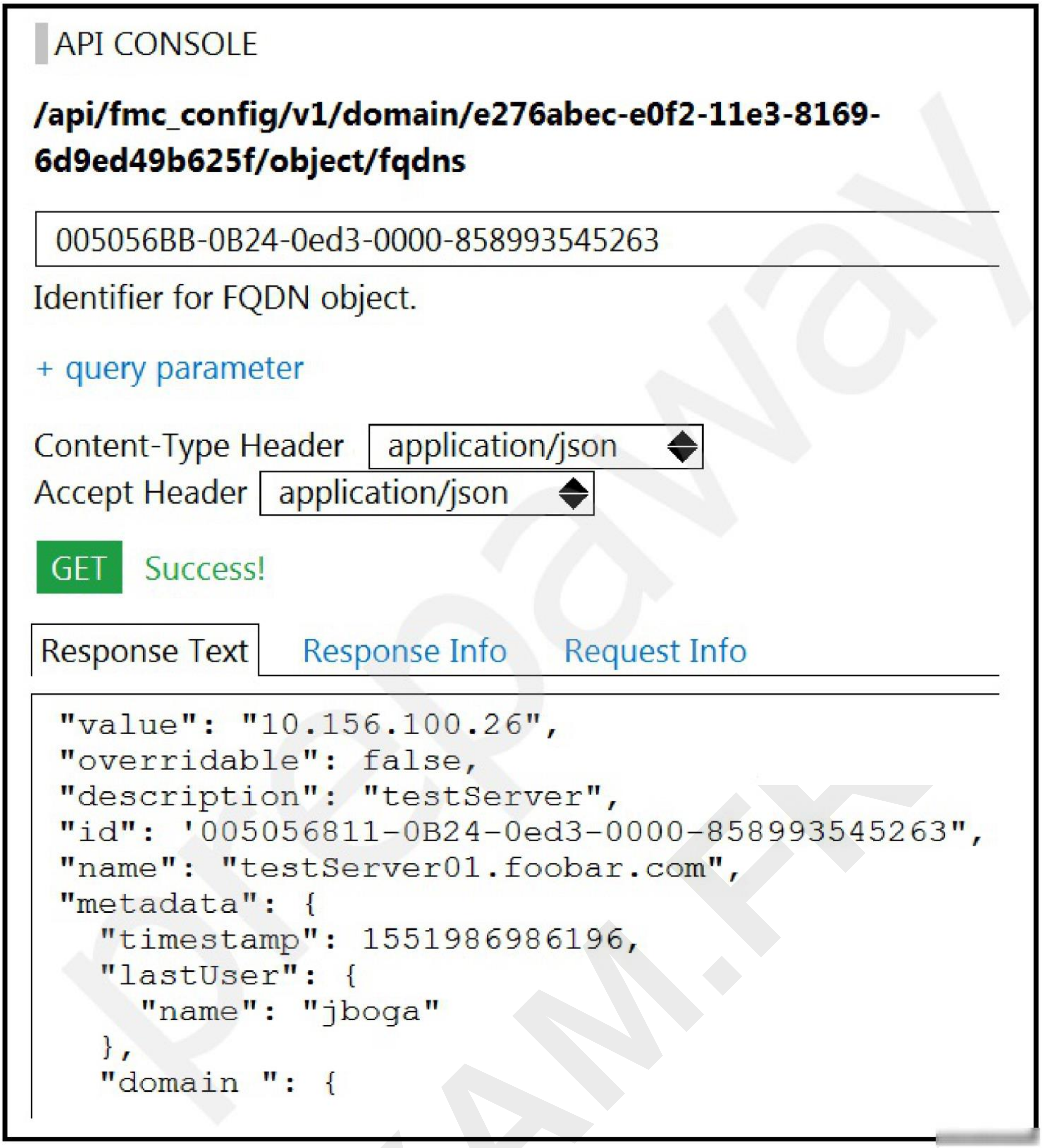
The correct answer is **C. Build a try/except around the urlopen to find errors occurring in the request.**

An HTTP 403 error indicates "Forbidden," meaning the server understands the request but refuses to authorize it. This typically occurs when the client lacks the necessary permissions or authentication credentials to access the requested resource. Option C addresses this directly by using a try/except block, a fundamental error-handling mechanism in programming. When the urlopen function (or equivalent for making HTTP requests) encounters a 403 error, it raises an exception. The except block catches this exception, allowing the application to gracefully handle the error without crashing. This approach provides a structured way to identify and respond to the specific error, which could involve logging the issue, displaying an informative message to the user, or attempting alternative access methods. Options A, B, and D are not appropriate for handling this specific error. Creating a loop (A) might result in an infinite loop if the issue is not resolved. Using exponential backoff (B) is better suited for transient errors like temporary server unavailability or rate limiting, not a permissions error. Redirecting to an internal server (D) is overly complex and doesn't address the core permission problem. Therefore, C is the most appropriate solution as it provides a targeted and standard approach to handling the 403 error. This solution adheres to best practices in software development, where proper error handling prevents application failures and improves robustness.

Authoritative Links:

Python try/except:https://docs.python.org/3/tutorial/errors.html
HTTP 403 Forbidden Status:https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/403 **HTTP Error Handling:**https://restfulapi.net/http-error-codes/

Question: 71



API CONSOLE

/api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/fqdns

005056BB-0B24-0ed3-0000-858993545263

Identifier for FQDN object.

Content-Type Header: application/json

Accept Header: application/json

```
"value": "10.156.100.26",
"overridable": false,
"description": "testServer",
"id": '005056811-0B24-0ed3-0000-858993545263",
"name": "testServer01.foobar.com",
"metadata": {
  "timestamp": 1551986986196,
  "lastUser": {
    "name": "jboga"
  },
  "domain ": {
```

Refer to the exhibit. Which API call does an engineer use to delete the FQDN object?

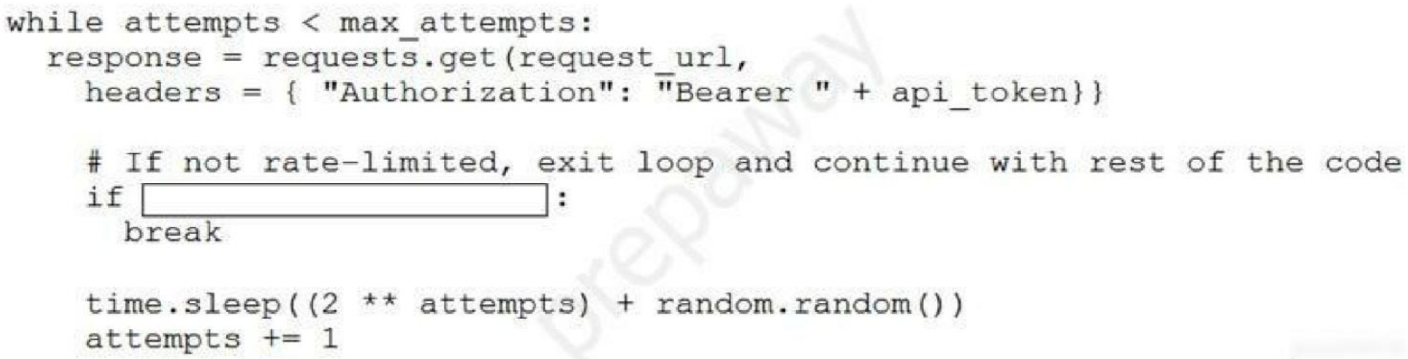
A.DELETE /api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f
B.DELETE /api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/fqdns/005056BB-0B24-0ed3-0000-858993545263
C.DELETE /api/fmc_config/v1/domain?id=e276abec-e0f2-11e3-8169-6d9ed49b625f
D.DELETE /api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/fqdns?id=005056BB-0B24-0ed3-0000-858993545263

Answer: B

Explanation:

DELETE /api/fmc_config/v1/domain/e276abec-e0f2-11e3-8169-6d9ed49b625f/object/fqdns/005056BB-0B24-0ed3-0000-858993545263.

Question: 72



```
while attempts < max_attempts:
  response = requests.get(request_url,
    headers = { "Authorization": "Bearer " + api_token}}

    # If not rate-limited, exit loop and continue with rest of the code
    if                     :
      break

    time.sleep((2 ** attempts) + random.random())
    attempts += 1
```

Refer to the exhibit. Which code snippet completes this code to handle API rate-limit?

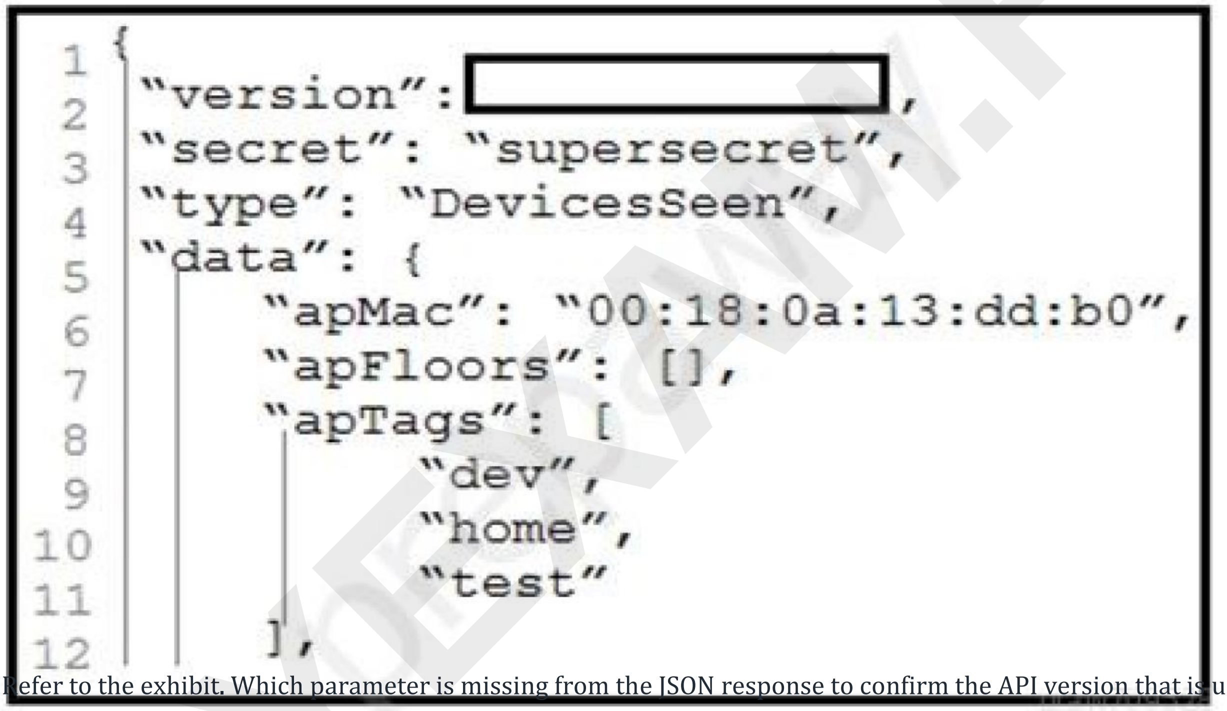
A.response.status_code == 429
B.response.status_code != 408
C.response.status != 408
D.response.status_code != 429

Answer: D

Explanation:

Correct answer is D:response.status_code != 429.

Question: 73



```
{
 "version":                     ,
 "secret": "supersecret",
 "type": "DevicesSeen",
 "data": {
     "apMac": "00:18:0a:13:dd:b0",
     "apFloors": [],
     "apTags": [
         "dev",
         "home",
         "test"
     ],
```

Refer to the exhibit. Which parameter is missing from the JSON response to confirm the API version that is used?

A.version 4
B.v 10
C.2

D.version 2

Answer: C

Explanation:

It is C since there is no quotation mark in the field, so the field only can be filled using a number which is 2. The other answers are string type.

Question: 74

Which two methods are API security best practices? (Choose two.)

A.Use tokens after the identity of a client has been established.
B.Use the same operating system throughout the infrastructure.
C.Use encryption and signatures to secure data.
D.Use basic auth credentials over all internal API interactions.
E.Use cloud hosting services to manage security configuration.

Answer: AC

Explanation:

Here's the justification for why options A and C are the correct API security best practices:

A. Use tokens after the identity of a client has been established: This practice aligns with modern authentication and authorization workflows. After a user or application is authenticated (usually by providing valid credentials), a token (like a JWT - JSON Web Token) is issued. This token acts as a temporary key, allowing the client to access protected API endpoints without repeatedly providing credentials. This approach enhances security by limiting the exposure of sensitive credentials and enabling fine-grained access control based on token claims. This aligns with the principle of least privilege and improves overall API security posture. * Link 1: JWT Explanation * Link 2: OAuth 2.0

C. Use encryption and signatures to secure data: Encrypting data in transit (using protocols like TLS/HTTPS) and at rest (using encryption algorithms) is critical to protect sensitive information from eavesdropping and unauthorized access. Digital signatures, on the other hand, are used to ensure the integrity of data and verify the sender's identity. These measures prevent man-in-the-middle attacks, data tampering, and impersonation. Implementing encryption and signatures is a fundamental requirement for building secure APIs. This practice also enables secure communication between services and clients, ensuring data confidentiality and integrity.
* Link 3: TLS/HTTPS Explained * Link 4: Digital Signatures Explained

Options B, D, and E are incorrect:

B. Use the same operating system throughout the infrastructure: While consistency can simplify management, it's not a primary security concern. Having diverse operating systems can actually offer better resilience in case of exploits targeting specific OS vulnerabilities.

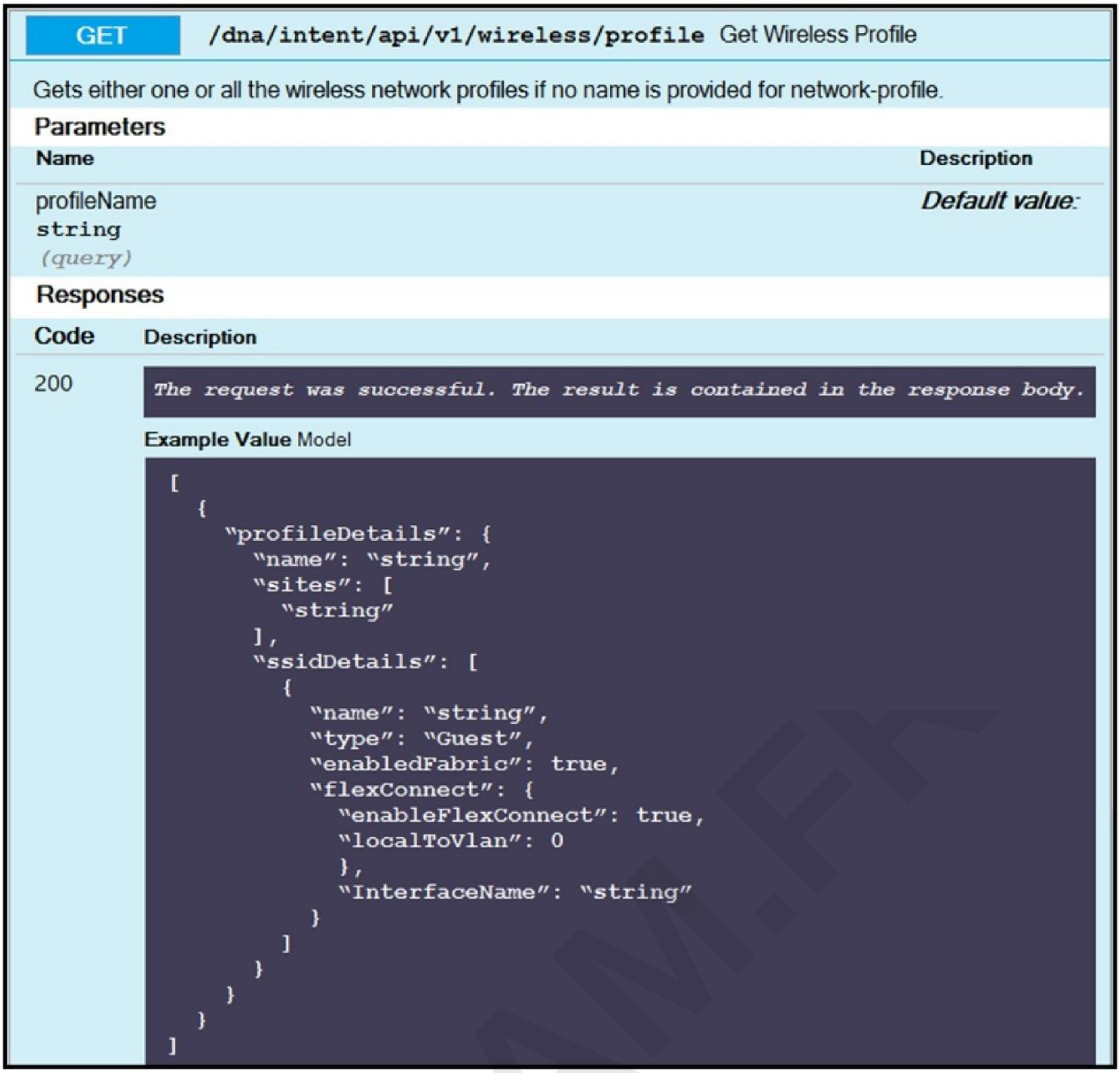
D. Use basic auth credentials over all internal API interactions: Basic auth sends credentials in plaintext (base64 encoded) making it insecure, especially in internal API interactions. It should be avoided and replaced with token-based authentication.

E. Use cloud hosting services to manage security configuration: While cloud providers offer security services, relying solely on them without proper configuration and understanding could lead to vulnerabilities.

Security is a shared responsibility model requiring active participation from both the cloud provider and the user/application developer.

Question: 75

DRAG DROP -



GET **/dna/intent/api/v1/wireless/profile** Get Wireless Profile

Gets either one or all the wireless network profiles if no name is provided for network-profile.

Parameters

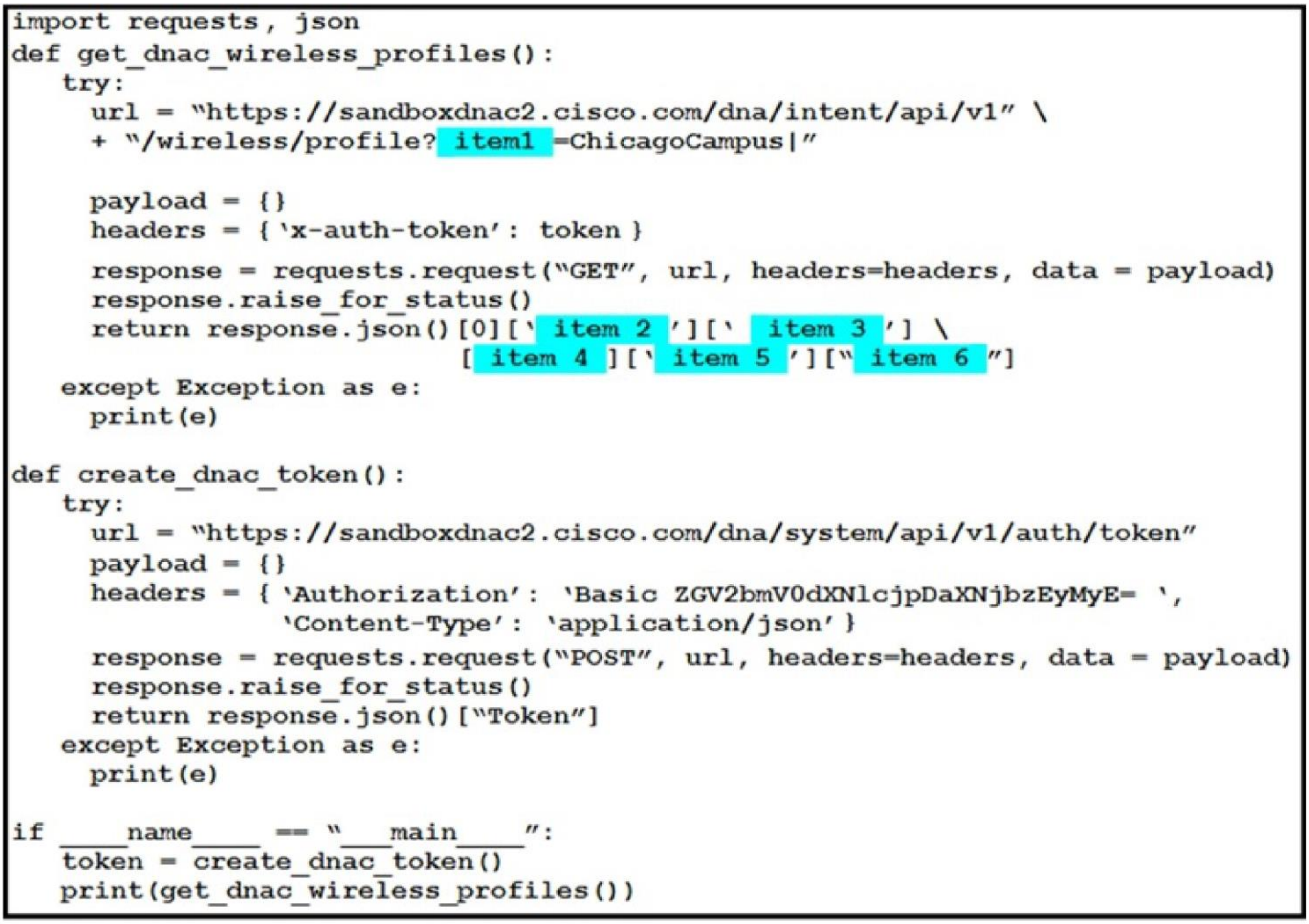
Name	Description
profileName `string` *(query)*	***Default value:***

Responses

Code	Description
200	*The request was successful. The result is contained in the response body.*

Example Value Model

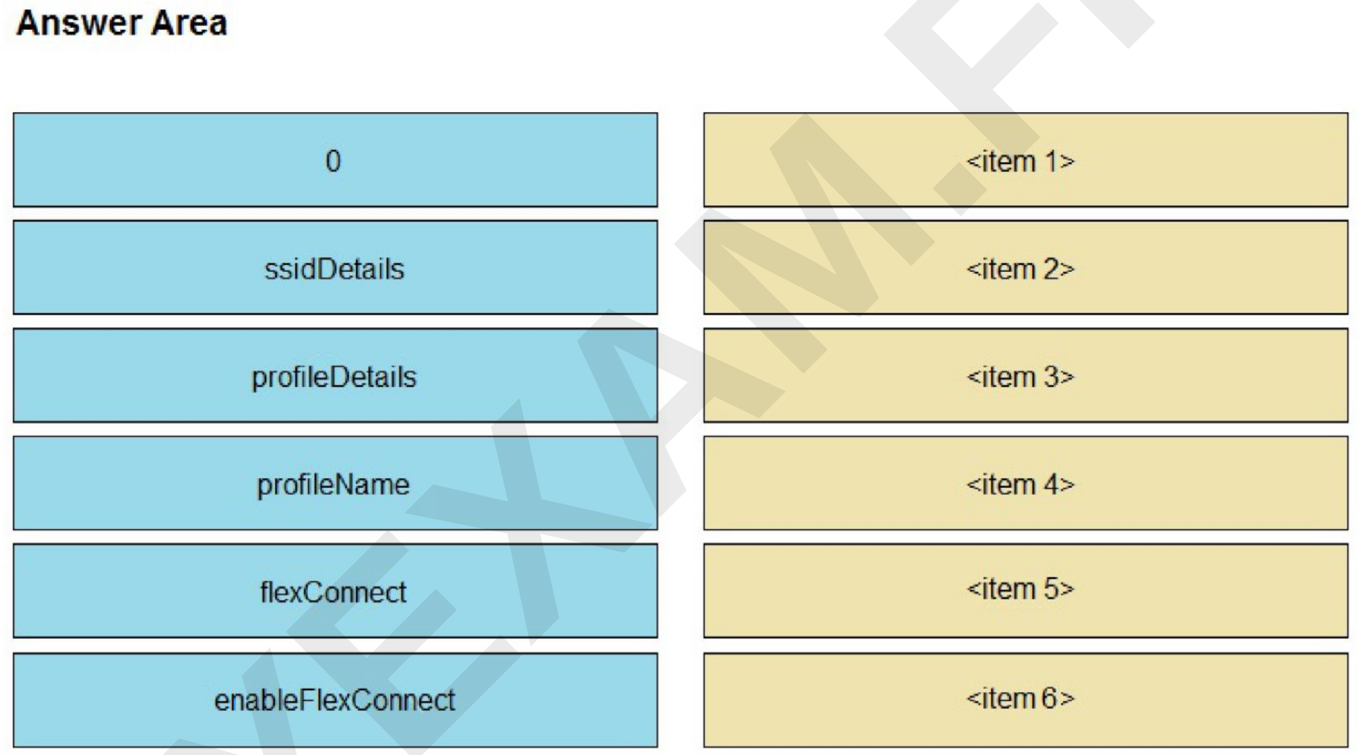
```
[
  {
    “profileDetails”: {
      “name”: “string”,
      “sites”: [
        “string”
      ],
      “ssidDetails”: [
        {
          “name”: “string”,
          “type”: “Guest”,
          “enabledFabric”: true,
          “flexConnect”: {
            “enableFlexConnect”: true,
            “localToVlan”: 0
          },
          “InterfaceName”: “string”
        }
      ]
    }
  }
}
]
```



```
import requests, json
def get_dnac_wireless_profiles():
    try:
      url = “https://sandboxdnac2.cisco.com/dna/intent/api/v1” \
      + “/wireless/profile? item1 =ChicagoCampus|”

      payload = {}
      headers = {‘x-auth-token’: token }
      response = requests.request(“GET”, url, headers=headers, data = payload)
      response.raise_for_status()
      return response.json()[0][‘ item 2 ’][‘ item 3 ’] \
                              [ item 4 ][‘ item 5 ’][“ item 6 ”]
    except Exception as e:
      print(e)

def create_dnac_token():
    try:
      url = “https://sandboxdnac2.cisco.com/dna/system/api/v1/auth/token”
      payload = {}
      headers = { ‘Authorization’: ‘Basic ZGV2bmV0dXNlcjpDaXNjbzEyMyE= ’,
                  ‘Content-Type’: ‘application/json’ }
      response = requests.request(“POST”, url, headers=headers, data = payload)
      response.raise_for_status()
      return response.json()[“Token”]
    except Exception as e:
      print(e)

if ____name____ == “___main____”:
    token = create_dnac_token()
    print(get_dnac_wireless_profiles())
```

Refer to the exhibit. The Python script is supposed to make an API call to Cisco DNA Center querying a wireless profile for the `ChicagoCampus` and then parsing out its enable FlexConnect value. Drag and drop the parts of the Python code from the left onto the item numbers on the right that match the missing sections in the exhibit.

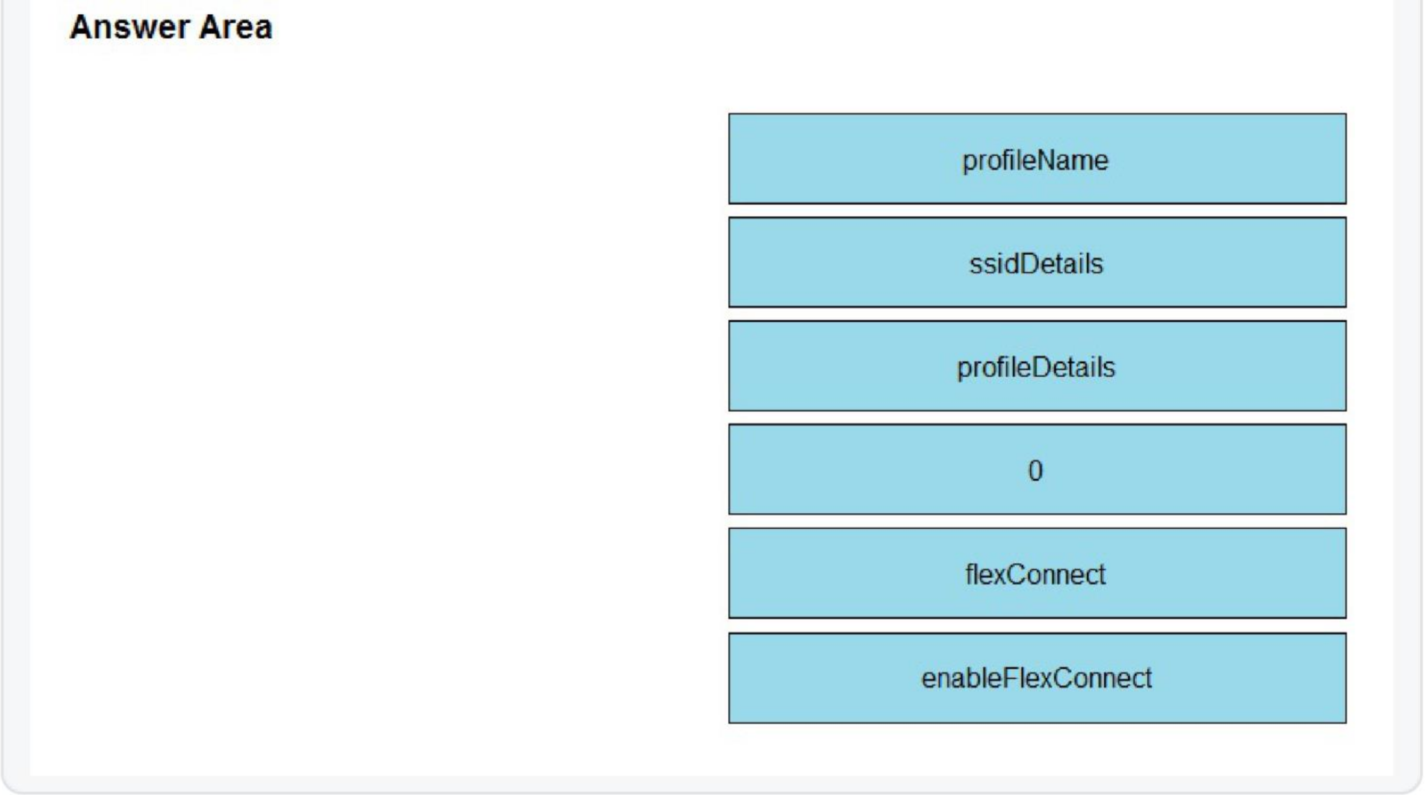
Select and Place:



Answer Area

0	<item 1>
ssidDetails	<item 2>
profileDetails	<item 3>
profileName	<item 4>
flexConnect	<item 5>
enableFlexConnect	<item 6>

Answer:



Answer Area

profileName

ssidDetails

profileDetails

0

flexConnect

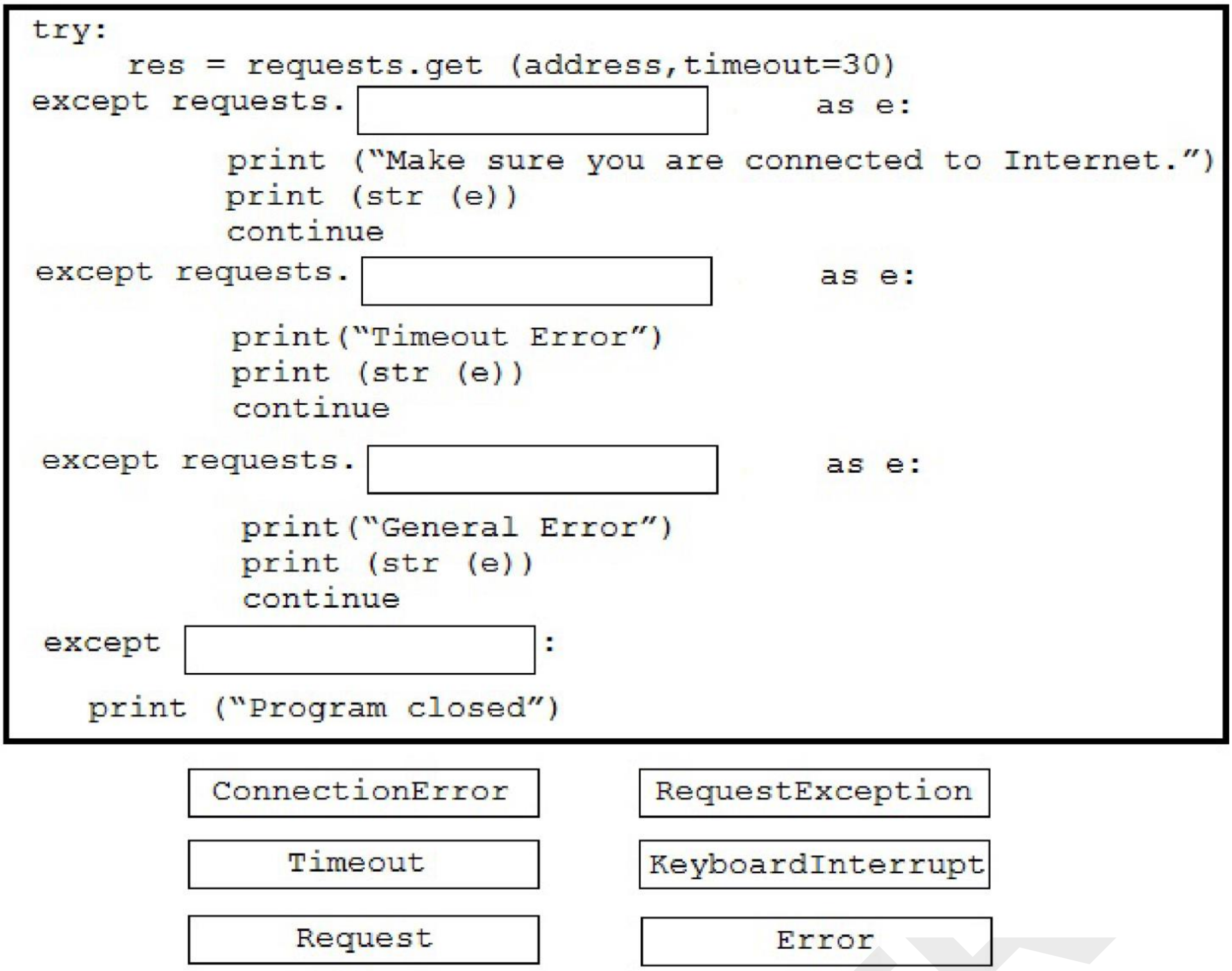
enableFlexConnect

Question: 76

DRAG DROP -

A developer is creating a Python script to catch errors using REST API calls and to aid in debugging. Drag and drop the code from the bottom onto the box where the code is missing to implement control flow for REST API errors. Not all options are used.

Select and Place:



```
try:
     res = requests.get (address,timeout=30)
except requests.                        as e:

          print ("Make sure you are connected to Internet.")
          print (str (e))
          continue
except requests.                        as e:

          print("Timeout Error")
          print (str (e))
          continue
except requests.                        as e:

          print("General Error")
          print (str (e))
          continue
except                        :
   print ("Program closed")
```

ConnectionError

RequestException

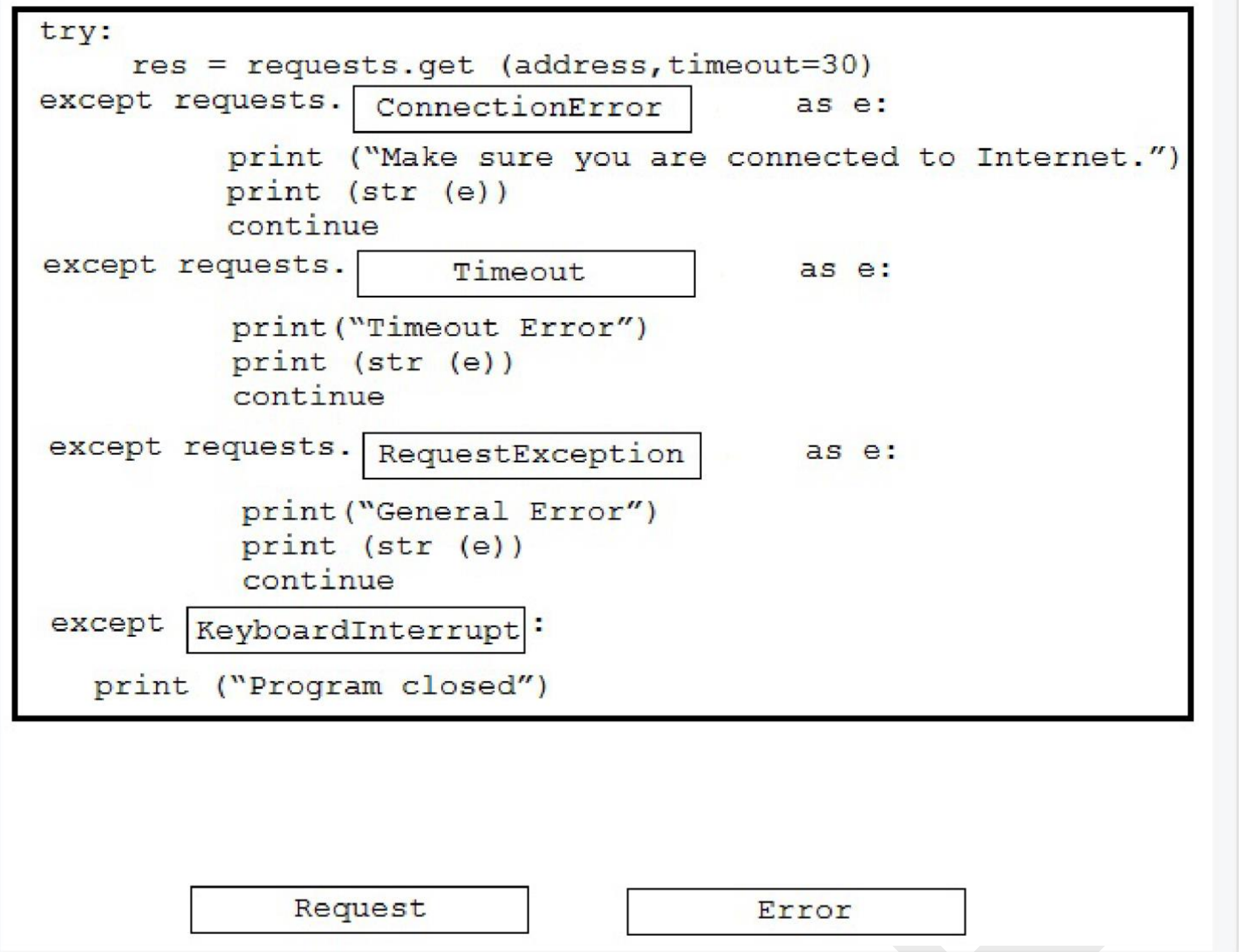
Timeout

KeyboardInterrupt

Request

Error

Answer:



```
try:
    res = requests.get (address,timeout=30)
except requests. ConnectionError      as e:
        print ("Make sure you are connected to Internet.")
        print (str (e))
        continue
except requests.   Timeout           as e:
        print("Timeout Error")
        print (str (e))
        continue
except requests. RequestException     as e:
        print("General Error")
        print (str (e))
        continue
except KeyboardInterrupt:
   print ("Program closed")
```

Question: 77

Which two encryption principles should be applied to secure APIs? (Choose two.)

A.Use temporary files as part of the encryption and decryption process.
B.Use encrypted connections to protect data in transit.
C.Reuse source code that contain existing UUIDs.
D.Embed keys in code to simplify the decryption process.
E.Transmit authorization information by using digitally signed payloads.

Answer: BE

Explanation:

The correct answer is B and E. Securing APIs requires a multi-faceted approach, and encryption is paramount. Option B, "Use encrypted connections to protect data in transit," is crucial because it leverages protocols like TLS/SSL to establish a secure, encrypted channel for data transmission between the API client and server. This prevents eavesdropping and man-in-the-middle attacks where sensitive information could be intercepted during transfer. Cloud environments often rely heavily on these protocols for securing API communications.

Option E, "Transmit authorization information by using digitally signed payloads," ensures data integrity and authenticity of the authorization claims. Digital signatures utilize cryptography to confirm the sender's identity and verify that the payload hasn't been tampered with. This is essential for authorization processes as it prevents unauthorized access by rogue parties. Methods like JWTs (JSON Web Tokens) often employ digital signatures to securely convey authorization details.

Option A, "Use temporary files as part of the encryption and decryption process," introduces unnecessary complexity and potential security risks, as these temporary files could become targets. Option C, "Reuse source code that contain existing UUIDs," has no direct bearing on encryption and may introduce security vulnerabilities if the UUIDs are predictable. Option D, "Embed keys in code to simplify the decryption process," is a severe security flaw; embedding keys makes them readily accessible to anyone who can access the codebase and can lead to a major breach if the code is compromised.

For more information on securing APIs, consider exploring the following resources:

OWASP (Open Web Application Security Project):https://owasp.org/www-project-top-ten/ - OWASP provides guidance on API security best practices.

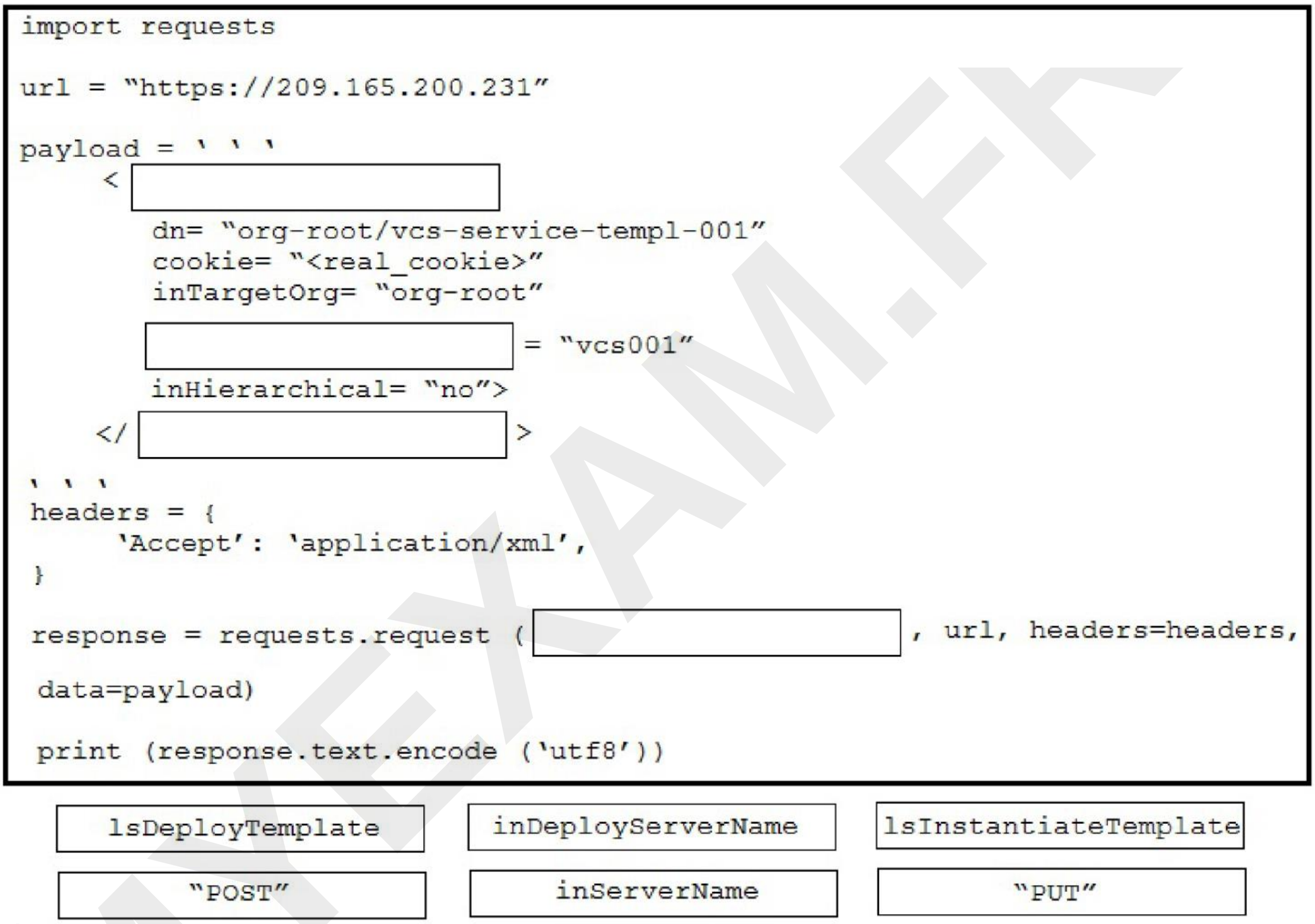
NIST (National Institute of Standards and Technology):https://www.nist.gov/cyberframework - Provides cybersecurity frameworks that address data protection.

Cloud Security Alliance:https://cloudsecurityalliance.org/ - Offers resources for cloud security best practices.

IETF (Internet Engineering Task Force):https://www.ietf.org/ - Specifies the technical standards for internet protocols like TLS/SSL.

Question: 78

DRAG DROP -
Drag and drop the code from the bottom onto the box where the code is missing to provision a new Cisco Unified Computing System server by using the UCS
XML API. Options may be used more than once. Not all options are used.
Select and Place:



```
import requests

url = “https://209.165.200.231”

payload = ' ' '
      < [                    ]
        dn= “org-root/vcs-service-templ-001”
        cookie= “<real_cookie>”
        inTargetOrg= “org-root”
        [                    ] = “vcs001”
        inHierarchical= “no”>
     </ [                    ] >
' ' '
headers = {
      'Accept': 'application/xml',
}

response = requests.request ( [                    ] , url, headers=headers,

data=payload)

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

lsDeployTemplate

inDeployServerName

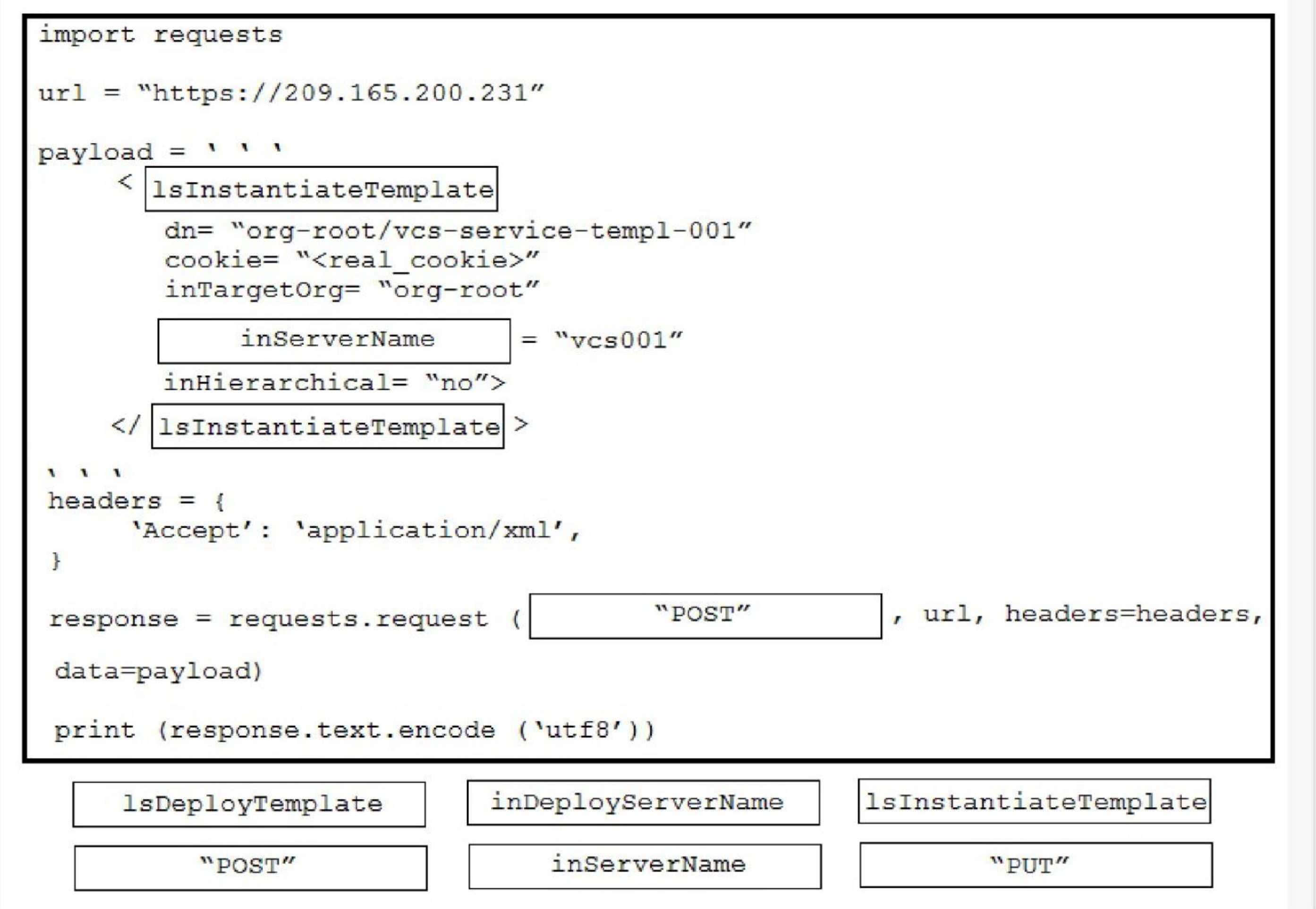
lsInstantiateTemplate

“POST”

inServerName

“PUT”

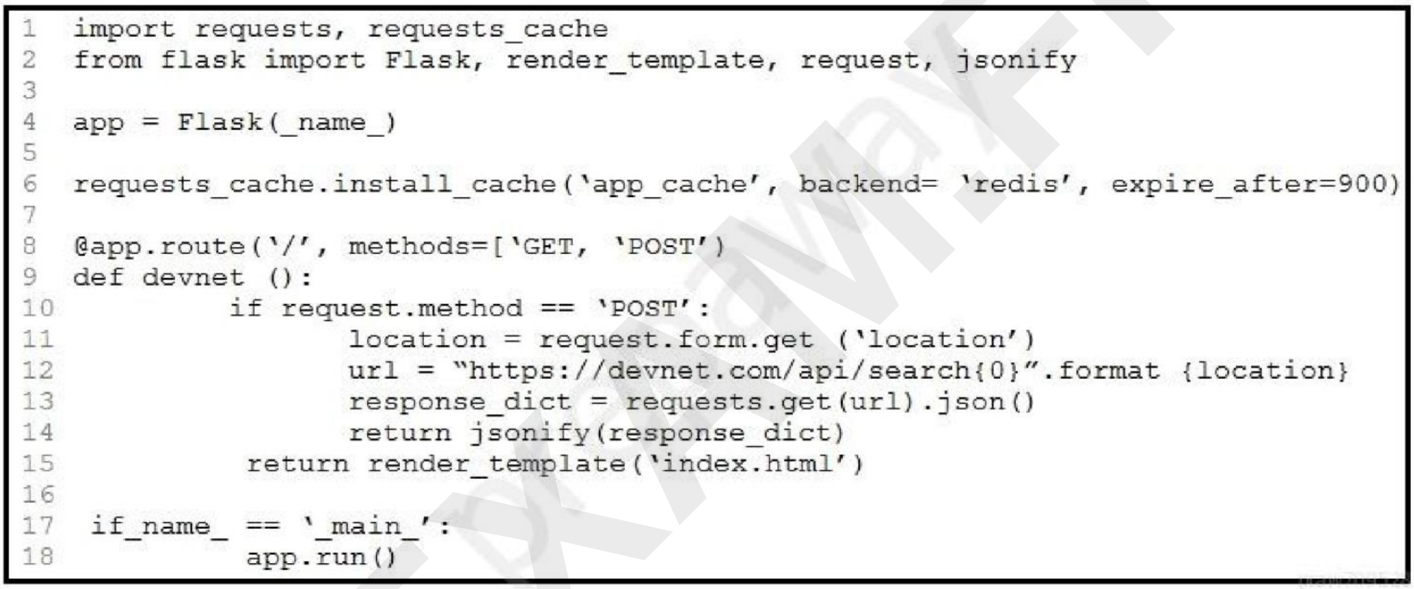
Answer:



```
import requests

url = “https://209.165.200.231”

payload = ‘ ‘ ‘
     < lsInstantiateTemplate
       dn= “org-root/vcs-service-templ-001”
       cookie= “<real_cookie>”
       inTargetOrg= “org-root”
       inServerName = “vcs001”
       inHierarchical= “no”>
     </ lsInstantiateTemplate >
‘ ‘ ‘
headers = {
     ‘Accept’: ‘application/xml’,
}

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

print (response.text.encode (‘utf8’))
```

lsDeployTemplate	inDeployServerName	lsInstantiateTemplate
“POST”	inServerName	“PUT”

Question: 79



```
import requests, requests_cache
from flask import Flask, render_template, request, jsonify

app = Flask(_name_)

requests_cache.install_cache(‘app_cache’, backend= ‘redis’, expire_after=900)

@app.route(‘/’, methods=[‘GET, ‘POST’)
def devnet ():
        if request.method == ‘POST’:
              location = request.form.get (‘location’)
              url = “https://devnet.com/api/search{0}”.format {location}
              response_dict = requests.get(url).json()
              return jsonify(response_dict)
         return render_template(‘index.html’)

 if_name_ == ‘_main_’:
         app.run()
```

Refer to the exhibit. An application has been developed to serve the users in an enterprise. After HTTP cache controls are implemented in the application, users report that they receive stale data when they refresh the page. Without removing HTTP cache controls, which change ensures that the users get current data when refreshing the page?

A.Add an if-None-Match header that has a value of an Entity Tag.

B.Reduce the expire_after value to 60.

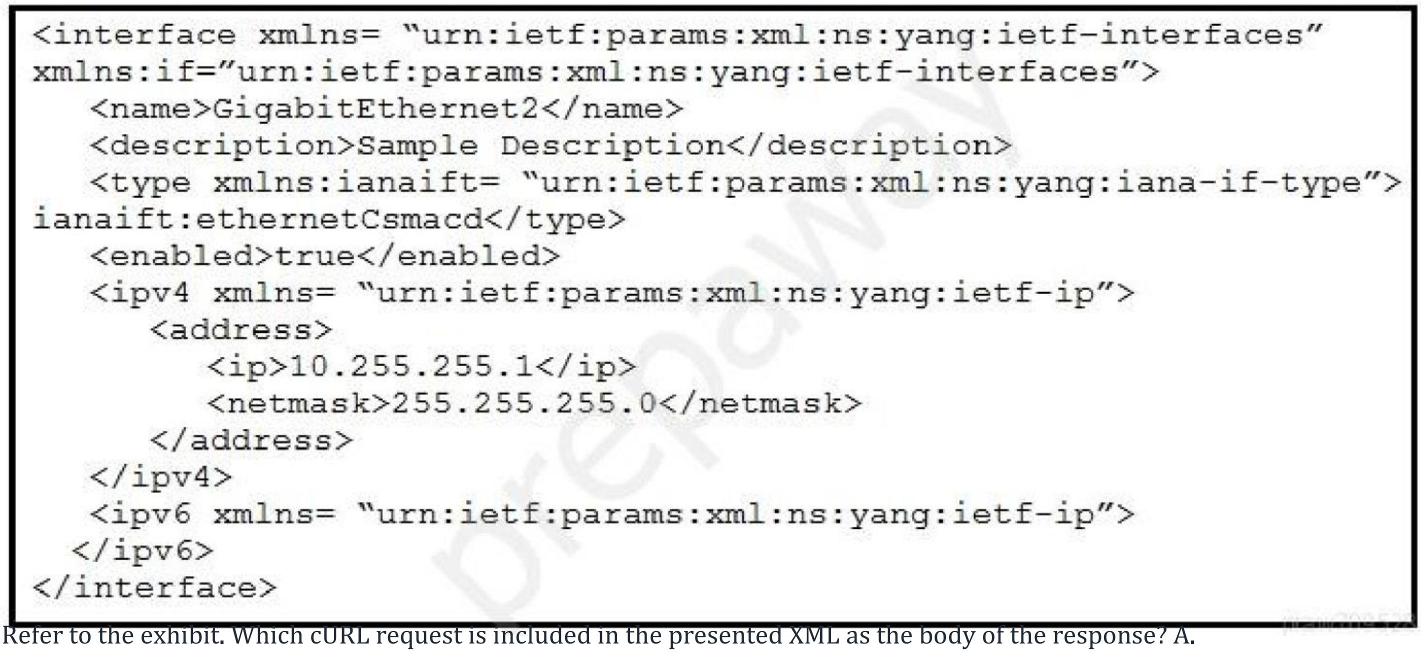
C.Add an Expires header that has a value of 0.

D.Add a Cache-Control header that has a value of no-cache, no-store, must-revalidate.

Answer: D
Explanation:

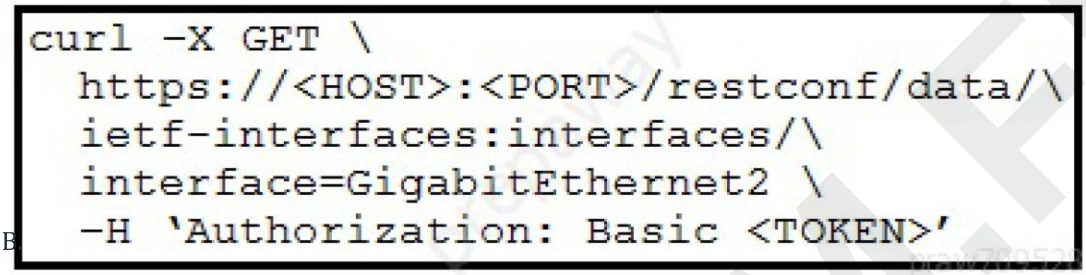
Add a Cache-Control header that has a value of no-cache, no-store, must-revalidate.

Question: 80



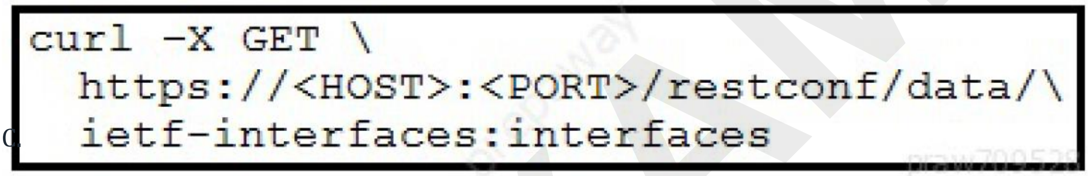
```
<interface xmlns= “urn:ietf:params:xml:ns:yang:ietf-interfaces”
xmlns:if=”urn:ietf:params:xml:ns:yang:ietf-interfaces”>
   <name>GigabitEthernet2</name>
   <description>Sample Description</description>
   <type xmlns:ianaift= “urn:ietf:params:xml:ns:yang:iana-if-type”>
ianaift:ethernetCsmacd</type>
   <enabled>true</enabled>
   <ipv4 xmlns= “urn:ietf:params:xml:ns:yang:ietf-ip”>
      <address>
         <ip>10.255.255.1</ip>
         <netmask>255.255.255.0</netmask>
      </address>
   </ipv4>
   <ipv6 xmlns= “urn:ietf:params:xml:ns:yang:ietf-ip”>
  </ipv6>
</interface>
```

Refer to the exhibit. Which cURL request is included in the presented XML as the body of the response? A.



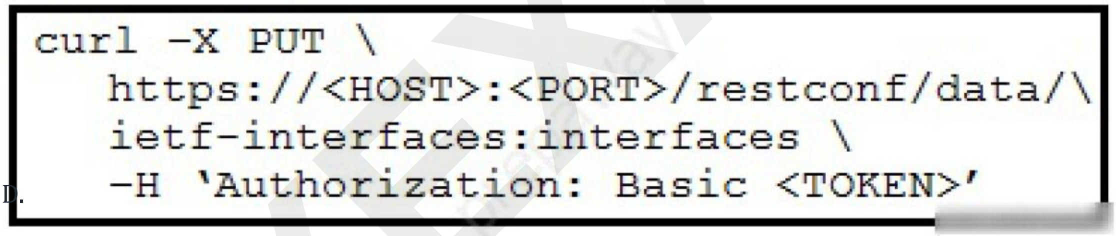
```
curl -X GET \
  https://<HOST>:<PORT>/restconf/data/\
  ietf-interfaces:interfaces/\
  interface=GigabitEthernet2 \
  -H 'Authorization: Basic <TOKEN>'
```

B.



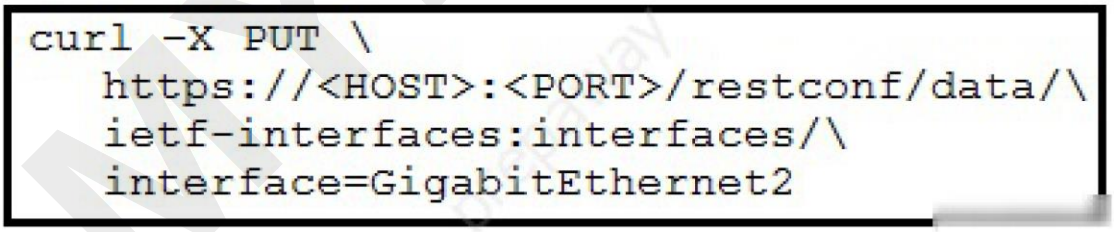
```
curl -X GET \
  https://<HOST>:<PORT>/restconf/data/\
  ietf-interfaces:interfaces
```

C.



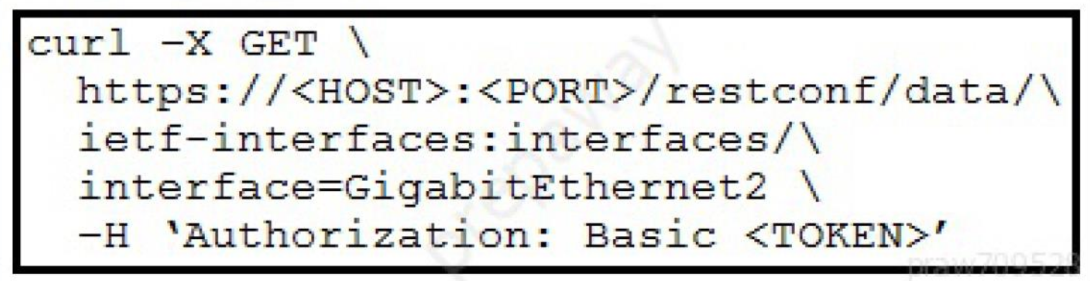
```
curl -X PUT \
   https://<HOST>:<PORT>/restconf/data/\
   ietf-interfaces:interfaces \
   -H 'Authorization: Basic <TOKEN>'
```

D.



```
curl -X PUT \
   https://<HOST>:<PORT>/restconf/data/\
   ietf-interfaces:interfaces/\
   interface=GigabitEthernet2
```

Answer: A
Explanation:



```
curl -X GET \
  https://<HOST>:<PORT>/restconf/data/\
  ietf-interfaces:interfaces/\
  interface=GigabitEthernet2 \
  -H ‘Authorization: Basic <TOKEN>’
```