PingIdentity.PT-AM-CPE.v2026-11-13.q37

Exam Code:	PT-AM-CPE
Exam Name:	Certified Professional - PingAM Exam
Certification Provider:	Ping Identity
Free Question Number:	37
Version:	v2026-11-13
# of views:	106
# of Questions views:	370
https://www.exam-tests.com/PT-AM-CPE-exam/PingIdentity.PT-AM-CPE.v2026-11-13.q37.html	

NEW QUESTION: 1
In which OAuth2 grant would you find a user code?
A. Client credentials grant
B. Authorization code grant
C. Resource owner password credentials grant
D. Device flow
Answer: (SHOW ANSWER)
The Device Authorization Grant (commonly referred to as the Device Flow, RFC 8628) is a specialized OAuth 2.0 grant flow supported by PingAM 8.0.2. It is designed for internet-connected devices that either lack a browser or have limited input capabilities (e.g., Smart TVs, IoT devices, or CLI tools).
In this flow, the interaction is split between the "Device" and a "Secondary Device" (like a smartphone or laptop) that has a full browser. The User Code is a fundamental component of this process:
Device Request: The device requests a code from PingAM.
PingAM Response: AM returns a Device Code (for the device) and a User Code (a short, human-readable string like BCDF-GHJK).
User Action: The device displays the User Code and a verification URL to the user.
Authorization: The user navigates to the URL on their smartphone, logs into PingAM, and enters the User Code.
Token Issuance: Once the user authorizes the request, the device (which has been polling AM using the Device Code) receives the Access and Refresh tokens.
The User Code is unique to the Device Flow (Option D). It is not used in the Client Credentials Grant (which is machine-to-machine), the Authorization Code Grant (which uses a redirect-based code), or the Resource Owner Password Credentials Grant (which uses direct username/password submission). In PingAM 8.0.2, administrators can

configure the length, character set, and expiration time of these user codes within the OAuth2 Provider settings.

NEW QUESTION: 2
In PingAM, which OpenID Connect endpoint can be used to validate an unencrypted ID token?
A. /oauth2/idtokeninfo
B. /oauth2/userinfo
C. /oauth2/introspect
D. /oauth2/tokeninfo
Answer: A (LEAVE A REPLY)
While OpenID Connect (OIDC) is built on top of OAuth2, it introduces specific endpoints for handling ID Tokens (the identity layer). In PingAM 8.0.2, when a client receives an ID Token, it is recommended to validate it locally using the provider's public keys. However, PingAM also provides a convenience endpoint for validation.
According to the "OpenID Connect 1.0 Endpoints" documentation:
/oauth2/idtokeninfo (Option A): This is the dedicated endpoint designed to receive an ID Token as a parameter.8 It validates the token's signature, checks the expiration and audience, and returns the claims contained within the token in a JSON format. This is specifically used for unencrypted ID tokens.
/oauth2/userinfo (Option B): This endpoint returns claims about the authenticated user but requires a valid Access Token in the authorization header, not an ID Token.9
/oauth2/introspect (Option C): This is a standard OAuth2 endpoint (RFC 7662) used to check the metadata and "activeness" of Access Tokens or Refresh Tokens, not the internal identity claims of an OIDC ID Token.10
/oauth2/tokeninfo (Option D): This is a legacy/non-standard endpoint that was used in older versions for Access Token validation and is not the primary OIDC validation endpoint in version 8.0.2.11 Therefore, for the specific task of validating an ID Token and retrieving its claims, /oauth2/idtokeninfo is the correct and authoritative endpoint in the PingAM 8.0.2 OIDC implementation.

NEW QUESTION: 3
In a PingAM cluster, how is the debug level set?
A. On each server in the debug.properties file
B. On a per-instance basis in the admin console
C. On a per-site basis in the admin console
D. It is not recommended to change the level at all
Answer: B (LEAVE A REPLY)
Debugging a PingAM 8.0.2 environment is essential for troubleshooting issues that occur at the engine level. In a multi-server deployment (a cluster), different servers may be experiencing different local issues (e.g., filesystem permissions or local JVM constraints).

Therefore, debug settings are managed at the server-specific level rather than the global site level.
According to the "Debug Logging" and "Server Settings" documentation:
The debug level (e.g., error, warning, message, info) is configured on a per-instance basis. In the PingAM Administrative Console, an administrator navigates to Deployment > Servers > [Server Name] > Debugging. Here, they can set the "Debug Level" and "Debug Output" (file vs. console).
Setting the level per instance allows an administrator to increase verbosity on a single "problematic" node without flooding the logs and impacting the performance of the entire healthy cluster. While these settings eventually modify internal properties, the Admin Console is the primary and recommended interface for making these changes in version 8.0.2.
Why other options are incorrect:
Option A: While legacy versions of OpenAM used a local debug.properties file, modern PingAM stores these settings in the Configuration Store, though they are applied to specific server instances.
Option C: A "Site" is a logical grouping for load balancing. Setting a debug level on a site would force all servers in that site to change simultaneously, which is often undesirable for targeted troubleshooting.
Option D: Changing the debug level is a standard and recommended practice for troubleshooting, provided it is returned to a lower level (like error or warning) once the issue is resolved to save disk space and CPU.

NEW QUESTION: 4
What does the acronym SAML stand for?
A. Scoped Assertion Markup Language
B. Secure Assertion Markup Language
C. Security Assertion Markup Language
D. Strong Assertion Markup Language
Answer: C (LEAVE A REPLY)
As defined in the PingAM 8.0.2 documentation under "Introduction to SAML 2.0," the acronym SAML stands for Security Assertion Markup Language. It is an XML-based framework specifically designed for communicating user authentication, entitlement, and attribute information between distinct entities. In a typical federation scenario, these entities are the Identity Provider (IdP), which asserts the identity of the user, and the Service Provider (SP), which consumes the assertion to grant access to resources.
SAML is governed by OASIS and has become the industry standard for cross-domain Single Sign-On (SSO). The "Security" aspect of the name refers to the cryptographic methods used to ensure the integrity and confidentiality of the assertions. "Assertion" refers to the specific statements made by the IdP about a subject (usually a user). These assertions can include Authentication Statements (proving the user logged in), Attribute

Statements (providing data like email or group membership), and Authorization Decision Statements (indicating what the user is permitted to do). PingAM 8.0.2 fully supports the SAML 2.0 core specifications, protocols, bindings, and profiles. Understanding this fundamental terminology is essential for administrators configuring "Circle of Trust" (CoT) environments or importing metadata from external partners, as the XML namespaces and schema definitions consistently reference the "urn:oasis:names:tc:SAML:2.0" identifier.

NEW QUESTION: 5
After installing a PingAM instance with the configuration directory path set to /home/forgerock/am, where is the default directory that contains the debug log files?
A. /home/forgerock/am/logs
B. /home/forgerock/am/var/logs
C. /home/forgerock/am/var/debug
D. /home/forgerock/am/debug
Answer: D (LEAVE A REPLY)
When PingAM is installed, it creates a specific directory structure within its Configuration Directory (also known as the AM_HOME or .openamcfg pointer target). This structure is standardized across versions to ensure that administrators and automated scripts can locate critical files.
According to the PingAM 8.0.2 "File System Reference" and "Debug Logging" documentation:
The primary directory for engine-level troubleshooting files (debug logs) is named debug.17 This directory is located immediately within the root of the configuration directory.
Therefore, if the configuration path is explicitly set to /home/forgerock/am, the resulting path for debug files will be /home/forgerock/am/debug (Option D).
It is important to distinguish between Audit Logs and Debug Logs:
Audit Logs: (e.g., access.audit.json) are usually found in the .../openam/log or .../openam/logs directory (making Option A a common distractor).
Debug Logs: (e.g., amAuth, amSession, amCore) are strictly stored in the debug directory.
The var directory (Options B and C) is a convention used in some ForgeRock "ForgeOps" containerized deployments (like those in Kubernetes) to separate variable data from static config. However, in a standard standalone installation as described in the question, the direct .../debug path is the verified default behavior of the PingAM installation wizard and configurator tool.

NEW QUESTION: 6
Which statements are correct in relation to an OAuth2 token exchange impersonation pattern?
A) The client may want to act as the subject on another service.
B) The client is used by a subject to act on behalf of another subject.
C) The requested token exchange involves a subject token only.

D) The requested token exchange involves a subject and actor token.

A. A and C only

B. B and D only

C. B and C only

D. A and D only

Answer: D (LEAVE A REPLY)

In PingAM 8.0.2, the OAuth 2.0 Token Exchange (RFC 8693) supports two primary patterns: delegation and impersonation. Understanding the difference between these is critical for secure microservices architecture.

According to the "Demonstrate Impersonation" section of the PingAM documentation, impersonation is a pattern where a client (the "Actor") acts as another identity (the "Subject") in a way that the downstream resource server sees only the Subject's identity.

Statement A is correct: In an impersonation flow, the client (which has been authorized by the user or is a trusted service) requests a token where it effectively "becomes" the subject to interact with another service. The downstream service treats the request as if it were coming directly from the subject, often with the same set of permissions.

Statement D is correct: To perform a token exchange for impersonation, the client must provide specific parameters to the /oauth2/access_token endpoint. It provides the subject_token (representing the identity to be impersonated) and the actor_token (representing the identity of the client/service that is performing the impersonation). PingAM validates both tokens to ensure the "Actor" has the permission to impersonate the "Subject." Why other statements are incorrect: Statement B describes delegation (where an actor acts on behalf of a subject but maintains their own identity in the act claim). Statement C is incorrect because a token exchange inherently requires proving who the requester is (the actor) and whom they represent (the subject). Without both tokens, the AM server cannot verify the authorization relationship required for impersonation. Therefore, the combination of A and D accurately reflects the impersonation pattern in PingAM 8.0.2.

NEW QUESTION: 7

Which of the following multi-factor authentication protocols are supported by PingAM?

A) Open authentication

B) Security questions

C) Web authentication

D) Universal 2nd factor authentication

E) Push authentication

A. B, C, and D

B. A, B, and E

C. A, C, and E

D. A, B, and C

Answer: C (LEAVE A REPLY)

PingAM 8.0.2 provides a robust framework for Multi-Factor Authentication (MFA) centered around modern, secure protocols and the Intelligent Access (Authentication Trees) engine. When discussing supported "protocols" in the context of MFA in PingAM documentation, the focus is on standardized methods for secondary verification.
The primary supported MFA pillars in PingAM 8.0.2 are:
Open Authentication (OATH): AM supports the OATH standards, specifically TOTP (Time-based One-Time Password) and HOTP (HMAC-based One-Time Password). This is implemented through the "OATH" authentication nodes, allowing users to use apps like ForgeRock Authenticator, Google Authenticator, or YubiKeys in OATH mode.
Web Authentication (WebAuthn): This is the implementation of the FIDO2 standard. It allows for passwordless and secure second-factor authentication using biometrics (like TouchID/FaceID) or hardware security keys (like YubiKeys). It is the successor to older standards and is natively supported via WebAuthn nodes.
Push Authentication: This is a proprietary but highly secure protocol used specifically with the ForgeRock/Ping Authenticator app. It allows a "Push" notification to be sent to a registered mobile device, which the user then approves or denies.
Why others are excluded from the selection: While PingAM supports Security Questions (KBA) and Universal 2nd Factor (U2F), they are often categorized differently in the 8.0.2 documentation. Security Questions are considered a "User Self-Service" or "Legacy" validation method rather than a modern MFA protocol. U2F is technically superseded by and included within the WebAuthn framework in PingAM 8.0.2. Thus, the most accurate grouping of distinct, core MFA protocols supported in the current version is A, C, and E, making Option C the correct answer.

NEW QUESTION: 8
During the PingAM startup process, what is the location and name of the file that the PingAM bootstrap process uses to connect to the configuration Directory Services repository?
A. <user-home-dir>/.openam/config/boot.json
B. /path/to/tomcat/<tomcat-instance-dir>/webapps/<am-instance-dir>/boot.json
C. <user-home>/<am-instance-dir>/boot.json
D. <user-home-dir>/<am-instance-dir>/config/boot.json
Answer: C (LEAVE A REPLY)
In PingAM 8.0.2, especially when utilizing File-Based Configuration (FBC), the startup sequence relies on a "bootstrap" phase to locate the system's configuration. According to the "Installation Guide" and "Configuration Directory Structure," the primary file involved in this process is named boot.json.
The boot.json file contains the essential connection details required for the AM binaries to find and unlock the configuration store (usually PingDS). This includes the LDAP host, port, bind DN, and references to the secret stores needed to decrypt the configuration.

The location of this file is determined by the Configuration Directory path specified during the initial setup. By default, PingAM creates its configuration directory in the home directory of the user running the web container. The standard path structure is <user-home>/<am-instance-dir>/. Therefore, the boot.json file is located at the root of this instance directory: <user-home>/<am-instance-dir>/boot.json.
Options A and D are incorrect because they place the file inside a /config subdirectory; while AM has many config files in subdirectories, the boot.json sits at the root to be accessible as the first point of entry.
Option B is incorrect because it suggests the file is stored within the Tomcat webapps folder. PingAM specifically avoids storing configuration data within the web application binaries to ensure that configuration persists even if the .war file is deleted or redeployed.
Understanding the location of boot.json is vital for DevOps engineers who need to automate the deployment of PingAM using tools like Amster or when troubleshooting a "Failed to connect to the configuration store" error during server startup.

NEW QUESTION: 9
In a multi-server deployment, what is the impact of not ensuring stickiness in the load balancer configuration?
A. Session failover will not work
B. The browser will not be able to validate the user session with the correct PingAM server
C. The user will see more redirects in their browser
D. Performance may decrease as load on the system will be higher
Answer: D (LEAVE A REPLY)
In a high-availability PingAM 8.0.2 cluster, the Load Balancer (LB) is responsible for distributing traffic across multiple AM instances. Session Stickiness (also known as session affinity) ensures that all requests from a specific user session are routed to the same AM server that initially created the session.
According to the PingAM "Deployment Planning" and "Load Balancing" documentation, PingAM is designed to be "sticky-preferred" but not "sticky-required" if the Core Token Service (CTS) is used. If stickiness is not ensured:
Performance Impact: Every time a user request lands on a different AM server (Server B) than the one that holds the session in local memory (Server A), Server B must query the CTS (External Store) to retrieve the session details, deserialize the object, and reconstruct the session state. This cross-server look-up introduces significant latency and increases the load on the PingDS instances hosting the CTS.
CTS Load: Without stickiness, every single request becomes a "Global" session lookup. This drastically increases the I/O and CPU overhead on the back-end directory servers, potentially leading to performance degradation of the entire identity platform.
Why other options are incorrect:

Option A: Session failover requires the CTS, but stickiness actually minimizes the need for failover logic during normal operation. Failover still works without stickiness, it just becomes the "default" behavior for every request.
Option B: AM servers in a cluster share the same encryption keys and back-end stores. Any server can technically validate a session by looking it up in the CTS; the browser doesn't "know" which server is correct.
Option C: Redirects are handled at the application logic level. While some internal processing changes, it doesn't necessarily result in extra browser-level HTTP redirects.
Thus, the primary negative impact of lacking stickiness in a correctly configured cluster is a decrease in performance (Option D) due to the constant session synchronization overhead.

NEW QUESTION: 10
In the OAuth2 Device Flow, which of the following HTTP codes is returned if a user has not yet authorized a client device?
A. HTTP 403
B. HTTP 400
C. HTTP 401
D. HTTP 302
Answer: (SHOW ANSWER)
The Device Authorization Grant (Device Flow), defined in RFC 8628 and implemented in PingAM 8.0.2, involves a polling mechanism where the device repeatedly asks the token endpoint for an access token using the device_code it received earlier.1 According to the PingAM documentation on "Device Authorization Grant" and "OAuth 2.0 Endpoints," during the period when the user is still navigating to the verification URL and entering their user code, the device's polling requests to the /oauth2/access_token endpoint will not result in a successful token issuance. Instead, PingAM returns a 400 Bad Request status code.
It is important to look at the JSON response body accompanying the 400 error. The body contains an error field with the value authorization_pending. 2This specific error code tells the device that the authorization request is still valid and in progress, but the user has not yet completed their part. The device should continue to poll at the interval specified in the initial response.
Other error codes like 403 Forbidden (Option A) would typically indicate a permanent rejection or that the device is polling too frequently (slow_down). 401 Unauthorized (Option C) is generally reserved for invalid client credentials when the client is confidential. 302 Found (Option D) is a redirect, which is not used in the back-channel polling phase of the Device Flow. Therefore, while a 400 error usually suggests a client error, in the context of the Device Flow, it is the standard protocol-level response used to communicate that the token is not yet ready because the user hasn't finished authorizing.

NEW QUESTION: 11

Which authentication node can you use in PingAM to add a key:value property to the user's session after successful authentication?

A. The Get Session Data node

B. You have to use a webhook, not a node

C. The Provision Dynamic Account node

D. The Set Session Properties node

Answer: (SHOW ANSWER)

In PingAM 8.0.2 Intelligent Access, the Set Session Properties node is a specialized utility node designed to modify the session object once it is created.

According to the "Authentication Node Reference":

During an authentication journey, data is typically stored in the sharedState. However, sharedState is transient and is destroyed once the tree finishes. If an administrator wants to take a piece of information (e.g., a "Risk Score" calculated during the tree, or a "Branch ID" retrieved from a legacy system) and make it a permanent part of the user's session, they must use the Set Session Properties node.

Functionality: This node allows you to map a value from the sharedState or transientState to a session property name. After the tree reaches a Success node, these properties are persisted in the session (either in the CTS for server-side sessions or the JWT for client-side sessions).

Usage: Once set, these properties can be retrieved later for Response Attributes in policies, or by applications using the /json/sessions endpoint.

Option A (Get Session Data node) is used to retrieve existing properties from an active session, not set them. Option B is incorrect because while webhooks can trigger external logic, the native way to modify the session within a tree is a node. Option C (Provision Dynamic Account node) is for creating user entries in the Identity Store (LDAP), not for managing session-level properties. Therefore, Set Session Properties (Option D) is the correct technical tool for this requirement in version 8.0.2.

NEW QUESTION: 12

A customer wishes to customize the OpenID Connect (OIDC) id_token JSON Web Token (JWT) to include the subject's employee number. Which of the following scripts should be customized to meet this requirement?

A. OIDC parameters script

B. OIDC claims script

C. OIDC attributes script

D. OIDC JWT script

Answer: B (LEAVE A REPLY)

In PingAM 8.0.2, the OpenID Connect (OIDC) Claims Script is the specific extensibility point designed to govern how user information is mapped and transformed into claims within an OIDC ID token or the UserInfo response. While PingAM supports standard scopes like profile and email out of the box, specialized business requirements-such as

including an "employee number" which might be stored as employeenumber in an LDAP directory-require a custom transformation.
According to the "OIDC Claims Script" reference in the PingAM documentation:
The script acts as a bridge between the Identity Store (the source of truth) and the OIDC Provider (the issuer). When a client requests a token, PingAM executes this script, providing it with a claimObjects map and the userProfile. The developer can then write Groovy or JavaScript logic to retrieve the employeeNumber attribute from the user's profile and add it to the resulting claims set.
The script typically follows this logical flow:
Identify the requested claims from the OIDC scope.
Fetch the corresponding raw attributes from the Identity Store (e.g., PingDS or AD).
Format and name the claim as per the OIDC specification or the specific client requirement (e.g., mapping LDAP employeenumber to OIDC claim emp_id).
Return the claims to be signed and embedded into the JWT.
Why other options are incorrect: Options A, C, and D reference script types that do not exist under those specific names in the standard PingAM 8.0.2 scripting engine. While there are "Access Token Modification" scripts and "Client Registration" scripts, the OIDC Claims Script is the only one authorized and designed to manage the payload of the id_token.

NEW QUESTION: 13
Does the user who runs the PingAM process need to have a home directory?
A. Yes, because this is where PingAM stores a pointer to the configuration
B. Yes, because this is where PingAM stores some of the configuration
C. Yes, because otherwise the process cannot listen on a port below 1024
D. No, not at all
Answer: A (LEAVE A REPLY)
According to the PingAM 8.0.2 Installation Guide, the user account on the operating system that runs the web application container (such as Apache Tomcat) must have a home directory. This requirement is critical for the "Bootstrap" process of the application. When PingAM starts for the first time or after a restart, the binaries need to know where the configuration data resides. PingAM looks for a hidden directory in the user's home directory named .openamcfg (or a similar name based on the deployment path). Inside this directory, AM creates and reads a file that contains the absolute path to the actual configuration directory (e.g., /home/tomcat/openam). This file acts as the pointer or "bootstrap" record.
If the user running the process does not have a home directory, the AM application will fail to initialize because it cannot create this bootstrap pointer. This often results in a "Configuration failed" error or the application reverting to an "unconfigured" state upon every restart. While it is possible to override the location of the configuration directory using JVM system properties (like -Dcom.sun.identity.configuration.directory), the default

behavior and best practice documented for standard deployments assume the existence of a home directory for the service user. This ensures that configuration remains persistent and isolated from the web container's temporary application files. Option C is incorrect as port listening restrictions are handled by the OS kernel/root privileges, not the existence of a home directory.

NEW QUESTION: 14
Which organization sets, maintains, and governs the SAML2 standard?
A. WC3
B. IETF
C. OASIS
D. ISC2
Answer: (SHOW ANSWER)
PingAM 8.0.2 is strictly compliant with various identity standards to ensure interoperability between different vendors and platforms. The Security Assertion Markup Language (SAML) V2.0 is the cornerstone of modern XML-based federation.7 According to the PingAM "SAML 2.0 Introduction" and "Supported Standards" documentation, the SAML 2.0 standard is developed and maintained by OASIS (the Organization for the Advancement of Structured Information Standards).8 Specifically, the OASIS Security Services Technical Committee (SSTC) is responsible for the specifications that define the SAML core (assertions and protocols), bindings (how SAML messages are mapped onto transport protocols like HTTP), and profiles (how SAML is used to solve specific use cases like Web Browser SSO).
Knowing the governing body is important for administrators when reviewing the "Technical Metadata" and "Schema" sections of PingAM, as AM's implementation follows the OASIS SAML 2.0 standards for XML signing, encryption, and assertion structure. Other organizations listed, such as the IETF (Internet Engineering Task Force), govern protocols like OAuth2 and OpenID Connect, while the W3C (World Wide Web Consortium) handles general web standards like XML and WebAuthn. However, for SAML2, OASIS remains the authoritative governing body.

NEW QUESTION: 15
A user enters their credentials, but is faced with the error message "user requires profile to login". What is a possible cause of this message?
A. Policies have not been defined to allow a user to access their profile page
B. The realm has not been set to user profile ignore mode
C. The user has not filled in the required information in their profile
D. The user has not entered the correct credentials
Answer: B (LEAVE A REPLY)
This error message is directly related to the User Profile configuration within a specific realm in PingAM 8.0.2. In the "Core Authentication Attributes" of a realm, PingAM defines

how it should handle user identities after they have successfully provided valid credentials through an authentication tree or chain.
There are primarily four modes for the User Profile setting:
Required: This is often the default. It specifies that after a user successfully authenticates, PingAM must be able to locate a corresponding user entry in the configured Identity Store. If the user exists in the datastore, the session is created. If the user does not exist, authentication fails with the error message "user requires profile to login" (or a similar profile-related exception in the logs).
Ignored: In this mode, PingAM issues an SSO session token immediately upon successful credential validation, regardless of whether a user profile exists in the back-end repository. This is useful for temporary or guest access where no permanent record is needed.
Dynamic: AM attempts to find the user; if the user is not found, it automatically creates a new profile in the identity store.
Dynamic with User Alias: Similar to dynamic creation but supports aliasing.
If an administrator sees the "user requires profile to login" error, it confirms that the credentials themselves were technically correct (the user passed the authentication nodes), but the realm is currently in Required mode (it has not been set to Ignore or Dynamic) and no matching entry exists in the identity store. This frequently happens in migration scenarios or when using external identity providers (like Social IDPs) where the "Link" or "Provisioning" step has not been properly configured in the authentication journey. To resolve this, the administrator must either pre-provision the user, set the mode to Ignore, or implement a Create Object node within the authentication tree to handle dynamic provisioning.

NEW QUESTION: 16
In the default Cloud Developer Kit (CDK) deployment of the forgeops repository, which pods provide the user interface functionality?
A. admin-ui, end-user-ui, login-ui
B. amadmin-ui, idmadmin-ui, login-ui
C. am-ui, idm-ui, login-ui
D. am-ui, idm-ui, end-user-ui
Answer: A (LEAVE A REPLY)
The Cloud Developer Kit (CDK), part of the forgeops repository, represents the modern approach to deploying the Ping Identity Platform (including PingAM 8.0.2) in a containerized, Kubernetes-native environment. According to the PingAM deployment and ForgeOps documentation, the platform has transitioned from a monolithic architecture-where the user interface was embedded within the AM web application-to a decoupled, microservices-aligned architecture. In a standard CDK deployment, the user interface components are separated into their own distinct pods to allow for independent scaling, updates, and management.

The three specific pods that provide user interface functionality in a default CDK environment are:
admin-ui: This pod hosts the administrative console. It is the centralized interface that administrators use to configure realms, manage identity stores, define authentication trees, and oversee the general health of both PingAM and PingIDM. By separating the administrative UI from the core engine, the platform reduces the attack surface and allows for more granular resource allocation.
end-user-ui: This pod serves the self-service portal for end-users. It is responsible for providing the interface where users can manage their own profiles, update passwords, register Multi-Factor Authentication (MFA) devices, and manage their consent for OAuth2/UMA applications. This UI interacts with the back-end via REST APIs to ensure a seamless and responsive user experience.
login-ui: This is a specialized pod dedicated to the authentication journey. When a user interacts with an "Intelligent Access" tree, the login-ui pod renders the callbacks (such as username prompts, password fields, or MFA challenges). This pod ensures that the presentation layer of the authentication process is modernized and distinct from the heavy processing logic of the PingAM core.
Collectively, these three pods ensure that the "User Interface" layer of the deployment is modular. This architecture is a prerequisite for high-availability deployments and is the standard configuration verified in the ForgeOps documentation for version 8.0.2 deployments.

Valid PT-AM-CPE Dumps shared by BraindumpsPass.com for Helping Passing PT-AM-CPE Exam! BraindumpsPass.com now offer the **newest PT-AM-CPE exam dumps**, the BraindumpsPass.com PT-AM-CPE exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com PT-AM-CPE dumps with Test Engine here: https://www.braindumpspass.com/Ping-Identity/PT-AM-CPE-practice-exam-dumps.html (**102** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 17
Which type of logs are written by PingAM?
A. Debug logs and Java logs
B. Audit logs and Java logs
C. Debug logs and audit logs
D. Java logs, debug logs, and audit logs
Answer: (SHOW ANSWER)

According to the PingAM 8.0.2 "Maintenance and Troubleshooting" documentation, the system generates two primary, distinct categories of logs for monitoring and problem-solving: Audit Logs and Debug Logs.
Audit Logs: These are high-level logs intended for security auditing, compliance, and reporting. They record specific "business events" or "state changes" within the system. Examples include successful logins, failed authentication attempts, administrative configuration changes (logged in config.audit.json), and policy evaluation decisions (logged in access.audit.json). These logs are structured (often in JSON) to be easily consumed by SIEM (Security Information and Event Management) tools.
Debug Logs: These are low-level, highly verbose logs intended for developers and support engineers. They record the internal "thought process" of the PingAM engine. They track the execution of specific Java classes, the results of LDAP queries, and the movement of data between authentication nodes. These logs are stored in the /debug directory and can be adjusted to different levels of verbosity (Error, Warning, Message, Info).
While PingAM runs within a Java Virtual Machine (JVM), and you may see container logs (like catalina.out in Tomcat) or "Java logs" from the underlying web server, these are technically external to the PingAM application itself. The PingAM application's internal logging framework is strictly split between Audit (what happened at a functional level) and Debug (why it happened at a code level). Therefore, Option C is the most accurate technical description of the logs natively managed and written by the PingAM service.

NEW QUESTION: 18
Which of the following is considered a confidential OAuth2 client?
A. Desktop clients
B. JavaScript clients
C. Web browsers
D. Web applications
Answer: (SHOW ANSWER)
According to the PingAM 8.0.2 documentation on "OAuth 2.0 Client Authentication," clients are categorized into two types based on their ability to maintain the confidentiality of their credentials: Public and Confidential.
A Confidential Client is defined as an application that is capable of securely storing a client_secret or a private key.1 These are typically applications where the code and configuration are not exposed to the end user. Web Applications (Option D) are the classic example of confidential clients because they run on a secure back-end server.2 The server-side code can store and use a secret to authenticate with PingAM's token endpoint without the risk of the secret being leaked to the user-agent or a third party.
In contrast:
Web Browsers (Option C) and JavaScript clients (Option B) are considered Public Clients.3 Since the code runs within the user's browser, any secret embedded in the application would be visible to the user via "View Source" or developer tools.4 Desktop

clients (Option A) and native mobile apps are also categorized as public clients in the OAuth2 specification (RFC 6749) because they are distributed to end-user devices.5 Even if the secret is obfuscated, it can be extracted through reverse engineering or decompilation.
For confidential clients, PingAM 8.0.2 supports various authentication methods at the token endpoint, including client_secret_basic, client_secret_post, and more secure options like Mutual TLS (mTLS) or Private Key JWT. By correctly identifying a client as confidential, administrators can enforce these stronger authentication requirements, ensuring that the client is indeed the entity it claims to be before granting access or refresh tokens.

NEW QUESTION: 19
Which of the following components is used to return data to PingGateway or the agent to be included with the policy decision?
A. Subjects
B. Resources
C. Response attributes
D. Actions
Answer: C (LEAVE A REPLY)
When PingAM 8.0.2 evaluates an authorization policy, the primary output is a "Permit" or "Deny" decision. However, applications and Policy Enforcement Points (PEPs)-like PingGateway or a Web Agent-often require additional metadata about the user or the session to function correctly (e.g., the user's employee ID, department, or a specific preference).
According to the PingAM documentation on "Policies" and "Requesting Decisions":
The mechanism used to provide this extra information is Response Attributes. When defining a policy in the PingAM UI or via REST, an administrator can configure "Response Attributes" which map internal attributes (from the User Profile or the Session) to keys that are sent back in the policy decision payload.
How it works: If a policy is configured with a response attribute mapping uid to User-ID, when PingGateway asks "Can user X access resource Y?", PingAM responds with "Permit" AND a map containing User-ID: X.
Consumption: PingGateway or the Web Agent can then take these attributes and inject them into HTTP headers (e.g., X-User-ID) so the downstream application can consume them without having to query AM again.
Subjects (Option A), Resources (Option B), and Actions (Option D) are all input components used to define the scope of a policy; they are not used to return data to the enforcer. Only Response Attributes serve the purpose of enriching the decision response with additional context.

NEW QUESTION: 20

Which one of the default PingAM audit log file contains messages related to changes made to sessions by end users?

A. access.audit.json

B. config.audit.json

C. authentication.audit.json

D. activity.audit.json

Answer: (SHOW ANSWER)

In PingAM 8.0.2, the audit logging service is designed to provide a comprehensive record of events for security, compliance, and troubleshooting. The audit logs are categorized by the type of event they record. According to the "Audit Logging Reference," PingAM generates several default log files, typically in JSON format.

The access.audit.json file is the primary log for events related to the lifecycle of a session and access to resources. This includes:

Session Creation: When a user successfully authenticates and a new session is established.

Session Termination: When a user logs out or a session expires.

Session Updates: Any changes made to the session, such as a Session Upgrade or modification of session properties by the end user or an application.

Policy Evaluations: Records of when a user requests access to a protected resource and the resulting permit or deny decision.

By contrast, the config.audit.json (Option B) records administrative changes to the system configuration (e.g., modifying a realm or a node). The authentication.audit.json (Option C) focuses specifically on the steps within an authentication tree, such as which nodes were visited and whether they succeeded or failed. While session changes happen after or as a result of authentication, the resulting session management event is logged in the access audit. The activity.audit.json (Option D) is generally used for internal system tasks and background processes. Therefore, for monitoring end-user session modifications, the access.audit.json is the correct authoritative source defined in the PingAM 8 documentation.

NEW QUESTION: 21

What is the purpose of HTTP-only cookies?

A. Cookies can not be read by client-side scripts

B. Cookies can only be transmitted over HTTPS

C. Cookies can not be read by the server

D. Cookies can only be transmitted over HTTP

Answer: (SHOW ANSWER)

In the "Additional Cookie Security" section of the PingAM 8.0.2 documentation, HttpOnly is described as a critical security attribute for session cookies (like iPlanetDirectoryPro). Its primary purpose is to mitigate the risk of session hijacking via Cross-Site Scripting (XSS) attacks.

When a cookie is marked with the HttpOnly flag, the browser is instructed to restrict access to that cookie. Specifically, it prevents client-side scripts-such as those written in JavaScript-from accessing the cookie through the document.cookie API. If an attacker successfully injects a malicious script into a page, the script will be unable to "read" the session token, even though the cookie is still automatically sent by the browser with every valid HTTP request to the server.
Option B describes the Secure flag, which ensures cookies are only sent over encrypted (HTTPS) connections.
Option C is incorrect because the server must be able to read the cookie to validate the user's session.
Option D is a common misconception; the HttpOnly flag does not restrict the transport to "HTTP-only" (non-secure) protocols; rather, it restricts the access method within the browser environment.
By default, PingAM 8.0.2 enables the HttpOnly flag for all session cookies. This is considered a best practice in modern identity management because it ensures that even if a web application has a vulnerability that allows for script injection, the user's primary authentication token remains protected from being exfiltrated by the attacker's script.

NEW QUESTION: 22
What are the possible outcomes of the Push Result Verifier node?
A. Success, Failure, Waiting, Retry
B. Success, Failure, Expired, Retry
C. Success, Failure, Expired, Waiting
D. Success, Failure, Expired, Waiting, Retry
Answer: (SHOW ANSWER)
The Push Result Verifier node is a core component of the "MFA: Push Authentication" journey in PingAM 8.0.2. Its primary function is to check the status of a push notification that was previously dispatched to a user's mobile device (usually via the Push Sender node).22 According to the "Authentication Node Reference" for version 8.0.2, the node evaluates the state of the push request and yields exactly four distinct outcomes:
Success: This path is followed if the user has actively approved the push notification on their registered device using the ForgeRock/Ping Authenticator app.
Failure: This path is taken if the user explicitly denies or rejects the push notification on their device, indicating a potential unauthorized login attempt.
Expired: This outcome occurs if the notification reaches its "Message Timeout" limit (defined in the Push Sender node) without any response from the user.23 In standard trees, this path often loops back to allow the user to try a different MFA method or resend the push.
Waiting: This outcome is triggered if a response has not yet been received but the timeout has not yet been reached. This is used in conjunction with a Push Wait or Polling

mechanism to create a "check-and-loop" logic until a final result (Success, Failure, or Expired) is determined.
The Retry outcome (mentioned in other options) is notably absent from this specific node's metadata. While a "Retry" might be implemented in the overall tree logic (for example, by using a Retry Limit Decision node after an Expired outcome), the Push Result Verifier node itself only reports the state of the specific push transaction it is tracking. Understanding these four discrete states is vital for designing resilient authentication journeys that handle user delays or network issues gracefully.

NEW QUESTION: 23
Which of the following statements about the PingAM tree designer is not true?
A. The PingAM tree designer lets you terminate the tree with a success node, a failure node, or a node linking to another authentication tree
B. The PingAM tree designer lets you create complex authentication by linking nodes together, creating loops, and nesting nodes within a tree
C. The PingAM tree designer is able to display custom and Ping Identity Marketplace nodes to use together with shipped nodes
D. The PingAM tree designer lets you integrate inner trees in the authentication flow
Answer: B (LEAVE A REPLY)
The Tree Designer in PingAM 8.0.2 is a visual, drag-and-drop tool used to build sophisticated login journeys. While it is highly flexible, it follows specific structural rules to ensure the authentication engine can execute the logic predictably.
Analysis of the statements:
Statement A is true: Trees must terminate in an outcome. Success and Failure nodes are standard. Additionally, the Inner Tree Evaluator node allows one tree to hand off processing to another "child" tree.
Statement C is true: The designer is extensible. Administrators can develop their own Java or Scripted nodes, and the Ping Identity Marketplace provides a wide range of third-party nodes (e.g., for biometric providers or specialized risk engines) that appear in the designer palette once installed.
Statement D is true: "Inner trees" are a supported concept, allowing for modularity where common logic (like MFA) can be built once and called from multiple parent trees.
Statement B is the "not true" statement. While the designer allows for complex logic and loops (e.g., looping back to a username prompt if a password is wrong), it does not support nesting nodes within a tree. In PingAM architecture, nodes are atomic components placed on a flat canvas. You cannot "nest" a node inside another node's configuration in the visual designer. Complexity is achieved through the branching and linking of these atomic nodes. If logic needs to be "nested" or grouped, it is done by creating a separate tree and calling it as an Inner Tree. Understanding this structural limitation is key for architects designing modular authentication frameworks.

NEW QUESTION: 24
A non-authenticated user requests a resource protected by PingGateway or a Web Agent.
Put the following events of the authentication lifecycle in chronological order:
User answers the "questions asked" (callbacks) by PingAM.
User tries to access a resource protected by PingGateway or a Web Agent.
Session reaches a timeout value or user logs out.
PingGateway or the Web Agent validates the session.
User is redirected to the authentication user interface of PingAM.
User is redirected to the resource.
A. 2-1-4-3-5-6
B. 2-5-1-6-4-3
C. 2-5-1-6-3-4
D. 2-1-5-6-4-3
Answer: (SHOW ANSWER)
The authentication lifecycle in a Ping Identity environment follows a strict sequence to ensure that only authorized users can access protected resources. This process is governed by the interaction between a Policy Enforcement Point (PEP), such as a Web Agent or PingGateway, and the Policy Decision Point (PDP), which is PingAM.
Following the chronological flow according to the PingAM 8.0.2 "Introduction to Authentication" and "Web Agent User Guide":
Step 2: The process begins when an unauthenticated user attempts to access a protected URL.
Step 5: The Agent/PingGateway intercepts the request, detects the absence of a valid session cookie, and redirects the user to the PingAM login URL (the UI).
Step 1: The user interacts with the AM UI, providing the necessary credentials or answering the "callbacks" (username, password, MFA) defined in the authentication tree.
Step 6: Upon successful authentication, PingAM issues a session token and redirects the user back to the original resource they were trying to access.
Step 4: The Agent/PingGateway receives the request again, but this time it contains a session token. The agent then validates the session with PingAM to ensure it is still active and possesses the correct permissions.
Step 3: Finally, the lifecycle ends when the session expires due to inactivity (Idle Timeout), reaches its Max Session Time, or the user explicitly logs out.
Sequence 2-5-1-6-4-3 (Option B) accurately captures this "Round-Trip" nature of modern web authentication. Options A and D are incorrect because they place the callback interaction before the initial redirect or the resource access. Option C is incorrect because it suggests the session reaches a timeout before the agent has a chance to validate the session for the current request.

NEW QUESTION: 25

When making a request to the /oauth2/access_token endpoint using the JWT profile client authentication method, which parameter is used to provide the JWT value?

A. client_credentials

B. client_token_value

C. client_id

D. client_assertion

Answer: D (LEAVE A REPLY)

PingAM 8.0.2 supports advanced client authentication methods defined in the OpenID Connect and OAuth 2.0 specifications, including private_key_jwt and client_secret_jwt. These methods allow a client to authenticate without sending a static password/secret in the request. Instead, the client generates and signs a JSON Web Token (JWT).

According to the "OAuth 2.0 Client Authentication" and "JWT Profile for Client Authentication" (RFC 7523) documentation, when a client sends this JWT to the /oauth2/access_token endpoint, it must use the client_assertion parameter.

The request must also include the client_assertion_type parameter, which must be set to the constant value: urn:ietf:params:oauth:client-assertion-type:jwt-bearer.

Option A (client_credentials) is a grant type, not a parameter for providing a JWT.

Option B (client_token_value) is not a standard OAuth2 parameter name.

Option C (client_id) is often included in the request, but it is the identifier of the client, not the container for the cryptographic assertion itself.

When PingAM receives a request with a client_assertion, it extracts the JWT, verifies the signature using the client's public key (stored in the client's profile or retrieved via a JWKS URI), and validates the standard claims (iss, sub, aud, exp). This method is significantly more secure than simple secrets because it proves the client possesses the private key and limits the window for replay attacks through the token's expiration claim.

NEW QUESTION: 26

Which area of PingAM does affinity mode relate to?

A. Authentication

B. Load balancing

C. Self-service

D. Authorization

Answer: B (LEAVE A REPLY)

In PingAM 8.0.2, the term Affinity Mode (or session affinity) is strictly related to Load Balancing (Option B). It describes a configuration where a load balancer ensures that all requests belonging to a specific user session are consistently routed to the same PingAM server instance in a cluster.

According to the "Load Balancing" and "Deployment Planning" documentation:

Affinity is critical for performance in stateful deployments. While PingAM can operate in a "stateless" manner by retrieving sessions from the Core Token Service (CTS) on every

request, this creates unnecessary overhead. Affinity Mode allows the AM server to satisfy requests using its local "In-memory" session cache.
There are two primary levels of affinity discussed in PingAM documentation:
Client-to-AM Affinity: Usually handled by the load balancer using a cookie (like the AMLB cookie) to keep the user on the same AM node.
AM-to-DS Affinity: Used when AM connects to the CTS (PingDS). This ensures that an AM server always talks to the same directory server node to avoid "replication lag" where a session might be written to one DS node but not yet visible on another.
Without affinity, the system remains functional due to the CTS, but performance decreases as every request requires a cross-network database lookup. Therefore, affinity is a core concept of the Load Balancing and high-availability architecture.

NEW QUESTION: 27
Which OAuth2 flow is most appropriate to support the use case of a client application implemented in a browser using a scripted language such as JavaScript?
A. Authorization code grant flow with PKCE
B. Implicit grant flow
C. Resource owner grant flow
D. Client credentials grant flow
Answer: A (LEAVE A REPLY)
In PingAM 8.0.2, the recommended and most secure flow for "Public Clients"-such as Single Page Applications (SPAs) written in JavaScript-is the Authorization Code Grant Flow with PKCE (Proof Key for Code Exchange).
Historically, the Implicit Grant Flow (Option B) was used for browser-based apps because they could not securely store a client_secret. However, the Implicit flow is now considered legacy and insecure due to the risk of access token leakage in the browser history or via referrer headers. The Resource Owner Password Credentials Grant (Option C) is also discouraged as it requires the application to handle user credentials directly, violating the core principle of delegated authorization. Client Credentials (Option D) is reserved strictly for machine-to-machine communication where no user is involved.
The Authorization Code Grant with PKCE addresses the security limitations of public clients by replacing the static client_secret with a dynamically generated "code verifier" and "code challenge." The process works as follows:
Challenge Generation: The JavaScript app creates a cryptographically strong random string (Verifier) and transforms it (Challenge).
Authorization Request: The app sends the challenge to PingAM.21
Code Exchange: After user login, AM returns an authorization code. The app then sends the code and the original verifier to the token endpoint.
Verification: AM verifies that the verifier matches the initial challenge before issuing the Access Token.

This flow ensures that even if an attacker intercepts the authorization code, they cannot exchange it for a token without the original verifier, which never left the browser's execution context. PingAM 8.0.2 fully supports this flow and provides specific configuration options in the OAuth2 Provider settings to enforce PKCE for all public clients.

NEW QUESTION: 28
Sam wants to start a service provider-initiated single sign-on and redirect to their own application, myapp.com. Which of the following URLs is the correct one to perform this action?
A. http://sso.domain.com/openam/saml2/jsp/idpSSOInit.jsp&RelayState=http%3A%2F%2Fmyapp.com
B. http://sso.domain.com/openam/saml2/jsp/idpSSOInit.jsp&goto=http%3A%2F%2Fmyapp.com
C. http://sso.domain.com/openam/saml2/jsp/spSSOInit.jsp&goto=http%3A%2F%2Fmyapp.com
D. http://sso.domain.com/openam/saml2/jsp/spSSOInit.jsp&RelayState=http%3A%2F%2Fmyapp.com
Answer: (SHOW ANSWER)
In SAML 2.0 federation with PingAM 8.0.2, there are two ways to initiate SSO: IdP-Initiated (where the user starts at the Identity Provider) and SP-Initiated (where the user starts at the Service Provider).3 According to the "SAML 2.0 Guide" for PingAM:
SP-Initiated SSO: The correct JSP file for an SP-initiated flow is spSSOInit.jsp. 4This script is used by an SP (in this case, PingAM acting as an SP or a "Fedlet") to generate a SAML AuthnRequest and send it to the IdP.
Redirecting to the Application: In the SAML 2.0 standard, the mechanism used to preserve state (like the final destination URL) across the redirect-heavy SSO process is the RelayState parameter. When the IdP sends the SAML assertion back to the SP, it also returns the RelayState value. The SP then uses this value to redirect the user to the final application.
While PingAM uses the goto parameter for internal redirects (like standard web login), RelayState is the required parameter name for SAML-related JSPs to ensure interoperability with the SAML specification. Therefore, the correct URL is .../spSSOInit.jsp combined with the RelayState parameter (Option D). Using idpSSOInit.jsp (Options A and B) would trigger an IdP-initiated flow, which is not what the question describes. Option C is incorrect because it uses the non-SAML goto parameter in a SAML initialization context.

NEW QUESTION: 29
Which statement differentiates the ForgeOps Cloud Deployment Model (CDM) from the Cloud Developer Kit (CDK) deployment?
A. Deployment generates random secrets

B. Supports deployment with Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), or Azure Kubernetes Service (AKS) clusters
C. Provides replicated directory services
D. Fully integrated PingAM, PingIDM, and PingDS installations
Answer: (SHOW ANSWER)
In the Ping Identity ForgeOps methodology for version 8.0.2, there are two primary deployment patterns used in Kubernetes: the Cloud Developer Kit (CDK) and the Cloud Deployment Model (CDM).
CDK (Cloud Developer Kit): This is intended for development and demonstration purposes. It is a "minimized" version of the platform. Crucially, in the CDK, the PingDS (directory service) is typically deployed as a single instance. It lacks the redundancy and replication required for production, as the goal is to reduce resource consumption on a developer's machine or a small test cluster.
CDM (Cloud Deployment Model): This is the reference architecture for production-grade environments. The CDM is designed for high availability and scale. According to the "ForgeOps Documentation," the primary differentiator is that the CDM provides replicated directory services. In a CDM deployment, PingDS is deployed in a multi-instance, replicated state (using a Kubernetes StateFulSet) to ensure that if one DS pod fails, the session and configuration data remain available.
While both models support major cloud providers like GKE, EKS, and AKS (Option B), generate random secrets (Option A), and provide integrated AM/IDM/DS stacks (Option D), the presence of multi-node replication in the directory layer is the definitive technical boundary between the "Developer" kit and the "Production" model.

NEW QUESTION: 30
What is the purpose of the SAML2 account mapper on the service provider (SP) side?
A. Maps multiple identity provider assertions together
B. Maps multiple SP user accounts together
C. Maps local user attributes to remote users' attributes
D. Maps remote users to local user profiles
Answer: (SHOW ANSWER)
In a SAML 2.0 Federation flow, once the Service Provider (SP) receives and validates a SAML Assertion from an Identity Provider (IdP), it must determine which local user account the assertion corresponds to. This is the role of the SAML2 Account Mapper.
According to the PingAM 8.0.2 documentation on "Federate Identities" and the "SAML 2.0 Reference":
The SP-side account mapper (specifically the SPAccountMapper interface or its scripted equivalent) is responsible for mapping the remote user (identified in the SAML assertion) to a local user profile in the SP's identity store.
This mapping can be achieved in several ways:

Account Linking: Finding an existing link between the NameID in the assertion and a local DN.
Attribute Matching: Using an attribute from the assertion (like mail) to search the local directory for a matching user.
Auto-Federation: If configured, creating a link or a new profile automatically based on the incoming data.
If the account mapper cannot find a corresponding local profile, the SP cannot create a local session, and the SSO process will fail, typically with a "User not found" or "Local identity not found" error. Thus, the purpose is strictly the identification of the local subject based on the remote assertion (Option D). Options A and B are incorrect as they describe aggregation or account merging which are not the primary function of the SAML mapper. Option C describes "Attribute Mapping," which is a separate step (handled by the Attribute Mapper) that occurs after the identity has been successfully mapped.

NEW QUESTION: 31
Which of the following approaches can be used to configure a basic installation of PingAM?
A. Either the graphical user interface in a browser, or a command-line program
B. The graphical user interface in a browser
C. There is no basic configuration needed
D. A command-line program
Answer: (SHOW ANSWER)
According to the PingAM 8.0.2 Installation Guide, once the am.war file has been deployed into a web container (such as Apache Tomcat), the administrator must perform an initial configuration to set up the configuration store and the primary administrative user (amAdmin). PingAM provides two primary pathways for this "basic" configuration to accommodate different environment needs:
GUI-based Configuration (Interactive): By accessing the AM deployment URL (e.g., https://auth.example.com:8443/am) in a standard web browser, the administrator is presented with an interactive setup wizard. This wizard guides the user through the license agreement, setting the amAdmin password, and defining the connection details for the Configuration Store (typically PingDS). This is the preferred method for single-instance setups or initial trials.
Command-Line Configuration (Automated/Passive): For DevOps-centric deployments, headless environments, or automated scripts, PingAM provides the configurator.jar (often used for "Passive" installations). Additionally, for version 8 deployments, Amster is the primary command-line interface (CLI) tool. Amster allows administrators to import a full configuration state from JSON files, bypassing the GUI entirely. This is crucial for CI/CD pipelines and Kubernetes-based deployments (like the ForgeOps CDK/CDP).
The flexibility to use either the browser-based GUI or command-line tools ensures that PingAM can be deployed efficiently across diverse infrastructures, from traditional on-

premises servers to modern cloud-native orchestration platforms. Therefore, Option A is the correct answer as it recognizes both valid administrative interfaces for the initial setup.

Valid PT-AM-CPE Dumps shared by BraindumpsPass.com for Helping Passing PT-AM-CPE Exam! BraindumpsPass.com now offer the **newest PT-AM-CPE exam dumps**, the BraindumpsPass.com PT-AM-CPE exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com PT-AM-CPE dumps with Test Engine here: https://www.braindumpspass.com/Ping-Identity/PT-AM-CPE-practice-exam-dumps.html (**102** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 32
A user's SSOTokenID is stored in a cookie when they successfully log in. What is the name of the PingAM property used to configure this cookie?
A. com.iplanet.am.cookie.name
B. iPlanetDirectoryPro
C. comIplanetAmSessionCookieName
D. com.sun.identity.agents.config.cookie.name
Answer: (SHOW ANSWER)
In PingAM 8.0.2, once a user successfully completes an authentication journey, the server issues a session token. For browser-based clients, this token is typically delivered via an HTTP cookie. The server-side property that defines the name of this specific cookie is com.iplanet.am.cookie.name.
According to the "Global Properties" and "System Configuration" documentation, this property is found under the Session service settings. By default, its value is set to iPlanetDirectoryPro (Option B). While iPlanetDirectoryPro is the default value of the cookie name, the question asks for the name of the property used to configure it.
Administrators often modify this property for security reasons to hide the fact that they are using PingAM or to avoid conflicts with other legacy systems. It is important to note that if this property is changed in the AM server, the corresponding configuration in all Policy Agents (Web or Java Agents) must also be updated to look for the new cookie name, otherwise, the agents will not be able to find the user's session and will redirect them to login repeatedly. Option D refers to an agent-side configuration key, but the central "Source of Truth" for the session cookie name within the AM platform is the server-side property com.iplanet.am.cookie.name.

NEW QUESTION: 33
Examine the following JWT client assertion in JSON format. From the subset of claims listed below, which claim can be optional?

JSON
JSON
{
"iss": "myClient",
"sub": "myClient",
"aud": "https://am.example.com/login/oauth2/access_token",
"jti": "id012345",
"exp": 1633363568,
"iat": 1633356368
}
A. jti
B. aud
C. iss
D. sub
Answer: A (LEAVE A REPLY)
When an OAuth2 client uses Private Key JWT or Client Secret JWT for authentication at the PingAM 8.0.2 token endpoint, it must present a JWT (JSON Web Token) containing specific claims that identify and authorize the client. This is governed by the OIDC and OAuth2 JWT Profile specifications (RFC 7523).
According to the PingAM documentation on "OAuth 2.0 Client Authentication" and the "JWT Profile for Client Authentication":
iss (Issuer): Mandatory. This must be the client_id of the OAuth2 client.
sub (Subject): Mandatory. This must also be the client_id of the OAuth2 client (as the client is the subject of the authentication).
aud (Audience): Mandatory. This must be the URL of the PingAM OAuth2 service (the token endpoint) or the issuer URL.
exp (Expiration Time): Mandatory. This protects against the long-term use of intercepted assertions.
The jti (JWT ID) (Option A) provides a unique identifier for the token. In the context of standard JWT validation, jti is used to prevent replay attacks by ensuring that a specific token is only processed once. While highly recommended for security hardening, the PingAM 8.0.2 technical reference for OAuth2 client assertions marks jti as optional unless the server is explicitly configured to require it for replay detection. Without a jti, PingAM will still validate the iss, sub, aud, and exp claims to authenticate the client. Therefore, among the choices provided, jti is the claim that can be omitted without inherently violating the base OAuth2 JWT authentication request requirements.

NEW QUESTION: 34
Which authentication node checks and validates a recovery code used during a multi-factor authentication challenge sequence?
A. Recovery Code Comparator node

B. Recovery Code Collector Decision node
C. Recovery Code Verifier node
D. Recovery Code Display node
Answer: B (LEAVE A REPLY)

NEW QUESTION: 35
To ensure the user's full name is displayed on the consent screen for an OpenID Connect application, which string should be added into the Support Claims property on the OpenID Connect tab page of the OAuth2 Provider service in PingAM?
A. name|en|Full name
B. Full name|en|name
C. full_name|Full name
D. name|en|given_name+' '+family_name
Answer: (SHOW ANSWER)
When a client requests an OpenID Connect (OIDC) scope (like profile), PingAM 8.0.2 may present a Consent Screen to the user, asking permission to share specific claims. To make this screen user-friendly, PingAM allows administrators to map technical claim names to human-readable labels and specify localizations.
According to the PingAM documentation on "Supported Claims" in the OAuth2/OIDC Provider settings:
The format for the Supported Claims property entry is:
ClaimName|Locale|DisplayName
In this syntax:
ClaimName: The technical OIDC claim (e.g., name, email, given_name).
Locale: The ISO language code (e.g., en, fr).
DisplayName: The text that will actually appear on the UI (the "Full name" label).
Therefore, the string name|en|Full name (Option A) is the correct configuration.
Option B is incorrect because it reverses the technical name and the display name.
Option C is incorrect as it lacks the required locale component and uses full_name (which is not the standard OIDC claim name; the standard is name).
Option D attempts to perform a logic operation (+) within a configuration field where only static mapping strings are allowed. Claim composition (concatenating first and last names) is handled by the OIDC Claims Script, not by the Supported Claims UI property.

NEW QUESTION: 36
When removing a forgeops deployment created with the Cloud Developer Kit (CDK) with the following command:
$ /path/to/forgeops/bin/forgeops delete
What components are removed from the deployment?
A. The Ping Identity Platform (CDK artifacts, PVCs, and the Access Management and Identity Management configurations) and ingress controller pods

B. The Ping Identity Platform (CDK artifacts, PVCs, and the Access Management and Identity Management configurations), certificate manager, and secret agent pods
C. The Ping Identity Platform (CDK artifacts, PVCs, and the Access Management and Identity Management configurations), ingress controller, DS operator, certificate manager, and secret agent pods
D. The Ping Identity Platform (CDK artifacts, PVCs, and the Access Management and Identity Management configurations) pods
Answer: D (LEAVE A REPLY)
The forgeops command-line tool is used to manage the lifecycle of the Ping Identity Platform in Kubernetes environments.9 When using the Cloud Developer Kit (CDK) for version 8.0.2, the delete subcommand is used to tear down the environment.
According to the "ForgeOps CLI Reference" and "CDK Shutdown and Removal" documentation:
The forgeops delete command (without additional flags like --force or specific component names) is designed to remove the Ping Identity Platform pods. This includes the core applications like PingAM, PingIDM, and PingDS, as well as the specialized UI pods (login-ui, etc.). It also removes the standard CDK artifacts and configuration manifests associated with that specific namespace.
However, the command follows a "safe delete" philosophy regarding infrastructure-level components:
Ingress Controllers, Certificate Managers (cert-manager), and the DS Operator are considered part of the "Base" or "Infrastructure" layer. These are typically installed once per cluster or namespace and are shared across multiple deployments. The forgeops delete command does not remove these by default, as doing so could disrupt other services.
PVCs (Persistent Volume Claims) and Secrets are also preserved unless the --force or -f flag is explicitly added to the command.
Thus, the answer is D. The command focuses strictly on the platform pods and their immediate deployment artifacts. If a developer wishes to perform a "deep clean" that removes the ingress and operators, they would need to use more specific commands like forgeops delete base or kubectl commands.10 This distinction is vital for developers to avoid accidentally deleting shared cluster infrastructure when they only intended to restart the Ping platform.

NEW QUESTION: 37
For Proof of Possession OAuth2 tokens, in addition to the access token, what must be presented to the authorization server?
A. Nonce
B. Client JSON Web Key (JWK)
C. State
D. Client private certificate

Answer: D (LEAVE A REPLY)
Proof of Possession (PoP) tokens, specifically Certificate-Bound Access Tokens as defined in RFC 8705 and supported by PingAM 8.0.2, are designed to prevent token misuse by binding the access token to a specific client's cryptographic material.9 According to the PingAM documentation on "Certificate-Bound Proof-of-Possession," when an OAuth2 client requests a token, PingAM retrieves the client's public key (either from a provided certificate or a JWK) and embeds a thumbprint (the cnf claim) of that material into the issued token. When the client subsequently presents this token to the Resource Server (or the Authorization Server's introspection endpoint), it must also provide "Proof" that it possesses the private key corresponding to that thumbprint.
In the Mutual TLS (mTLS) approach, this proof is provided by the Client private certificate presented during the TLS handshake.10 The server verifies that the certificate used to establish the secure connection matches the one bound to the token. Without presenting the certificate (Option D), the token is considered "unbound" or invalid, even if the token itself is otherwise well-formed. This mechanism effectively "pins" the token to the client, ensuring that if the token is stolen, it cannot be used by any other entity that does not possess the matching private key. Nonce and State (Options A and C) are used during the initial authorization request for different security purposes (replay protection and CSRF), and while a JWK (Option B) can be used to define the public key, the actual presentation of proof during an mTLS transaction is the certificate.

Valid PT-AM-CPE Dumps shared by BraindumpsPass.com for Helping Passing PT-AM-CPE Exam! BraindumpsPass.com now offer the **newest PT-AM-CPE exam dumps**, the BraindumpsPass.com PT-AM-CPE exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com PT-AM-CPE dumps with Test Engine here: https://www.braindumpspass.com/Ping-Identity/PT-AM-CPE-practice-exam-dumps.html (**102** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)