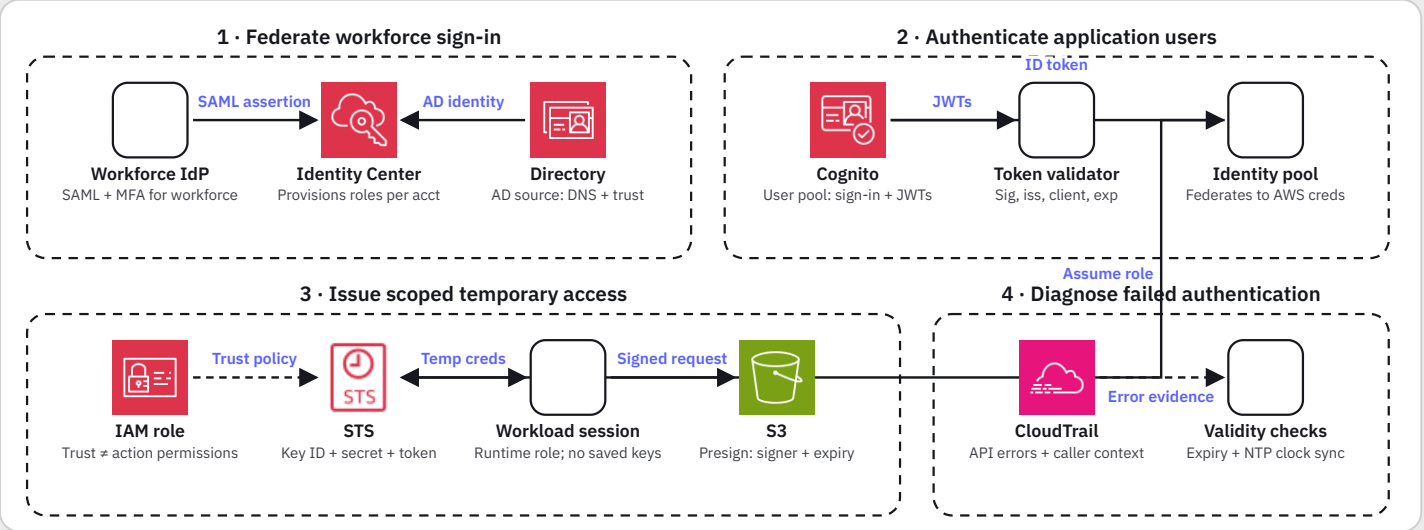
DOMAIN 4 · IDENTITY AND ACCESS MANAGEMENT · 20% OF THE EXAM · TASK STATEMENT 4.1

Design, implement, and troubleshoot authentication strategies

Federate, validate, issue, troubleshoot



1 Federate workforce sign-in

Choose **Identity Center** for workforce access across AWS accounts. Permission sets become account roles. External federation uses SAML; MFA is enforced at the external IdP. AD integration requires working DNS and connectivity.

2 Authenticate application users

User pools authenticate app users and issue JWTs; identity pools exchange trusted tokens for AWS credentials. Verify signature, issuer, aud/client_id, token_use, and expiry; decoding a JWT is not validation.

3 Issue scoped temporary access

Choose **runtime roles** over stored keys; SDK role providers refresh credentials, manually supplied STS credentials do not. Trust permits assumption; policies limit actions. Presigned URLs cannot outlive signing credentials.

4 Diagnose failed authentication

Use **CloudTrail error context** to identify the caller and failing API. Expired credentials require renewal; clock skew requires time synchronization. Application JWT validation failures also require application-side logs.

Exam traps

- **User pools ≠ identity pools**: user pools issue JWTs for app sign-in; identity pools obtain AWS credentials. A JWT alone cannot sign an AWS API request.
- **Presigned URLs inherit limits**: a URL cannot outlive its signing credentials; it grants only the signed operation, within the signer's permissions. It is not an independent IAM grant.
- **AccessDenied is authorization**: check role policies and permission-set provisioning. Invalid or expired tokens and clock skew instead point to authentication or request-validation failures.

Get practice exams and lessons for AWS Certified Security - Specialty

AWS Certified Security - Specialty Cheat Sheet

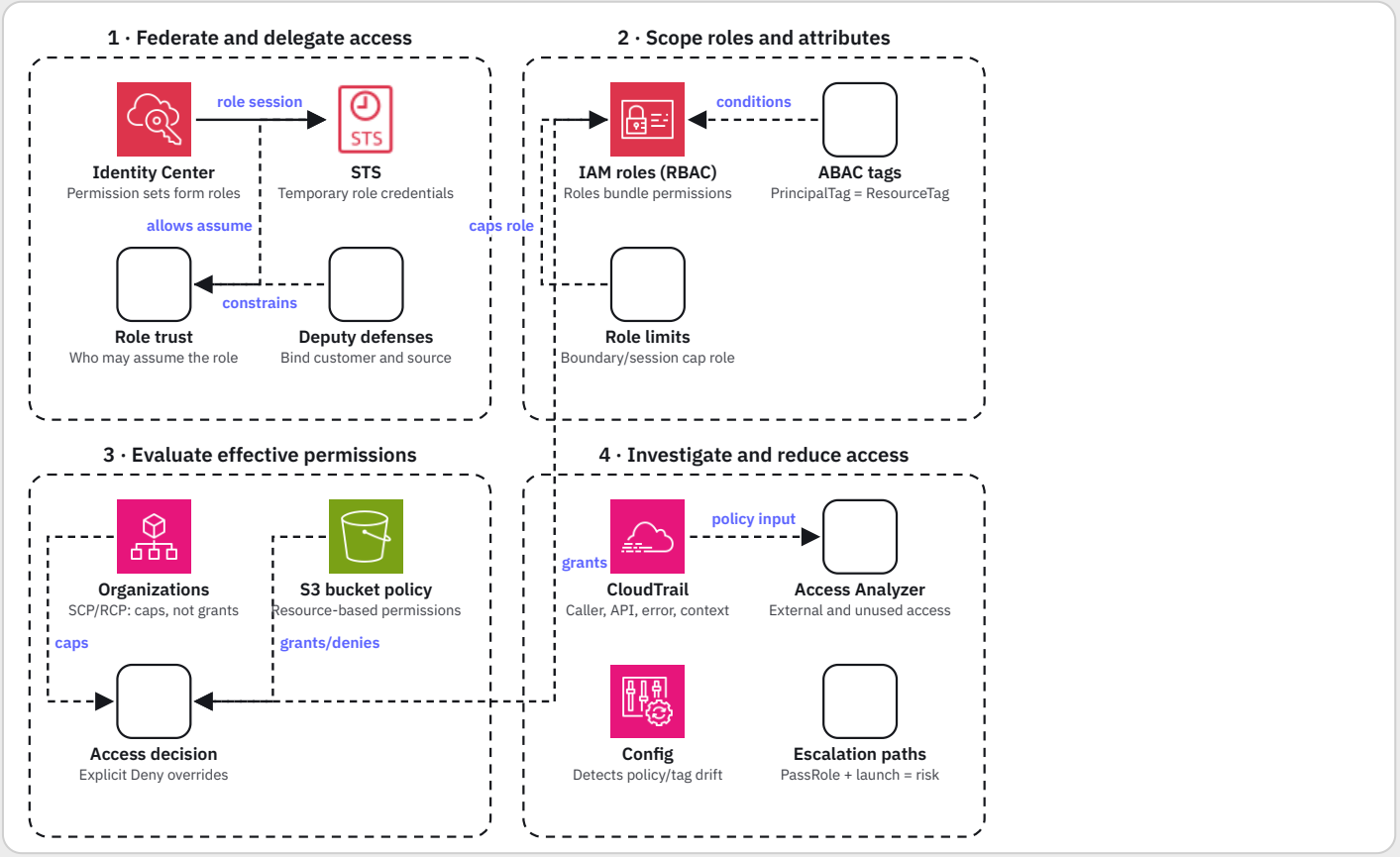
SCS-C03 · Domain 4 of 6

BlowStack

DOMAIN 4 · IDENTITY AND ACCESS MANAGEMENT · TASK STATEMENT 4.2

Design, implement, and troubleshoot authorization strategies

Delegate, scope, evaluate, investigate



1 Federate and delegate access

Cross-account AssumeRole needs caller Allow and target trust. Confused deputy defenses: sts:ExternalId for third parties; aws:SourceArn/SourceAccount for AWS services. IAM Roles Anywhere supports X.509 workloads.

2 Scope roles and attributes

Choose **ABAC** for changing teams/resources: match trusted principal or session tags to resource tags; restrict tag writes. RBAC assigns scoped roles. IAM paths organize names, not authorization boundaries.

3 Evaluate effective permissions

Cross-account resource access needs caller identity and resource-policy authorization. SCPs/RCPs, boundaries and session policies limit applicable grants; they never grant access. No applicable Allow means implicit deny.

4 Investigate and reduce access

AccessDenied: inspect CloudTrail context; test policies with IAM Policy Simulator and check its coverage limits. Access Analyzer identifies exposure and unused access; findings and Config evaluations do not themselves repair access.

Exam traps

- **Resource-policy exceptions** matter: same-account grants directly to a user or role-session ARN can bypass implicit boundary denies; session grants also bypass implicit session-policy denies.
- **ABAC is not safe with editable tags**: unrestricted tagging or sts:TagSession can enable escalation. Govern attribute sources, allowed tag keys/values and transitive session tags.
- **iam:PassRole is not AssumeRole**. Restrict role ARNs and iam:PassedToService; combining PassRole with service creation can expose the passed role's privileges.

Get practice exams and lessons for AWS Certified Security - Specialty

DOMAIN 4 · IDENTITY AND ACCESS MANAGEMENT · WHAT TO REMEMBER

Concepts and services of Identity and Access Management

Concepts to know

Effective Permissions

Authorization combines policy grants, applicable guardrails, and request context; **an applicable explicit deny wins**. Adding an allow cannot overcome a deny, and boundaries and organizational policies do not themselves grant access.

Cross-Account Access

Cross-account role assumption requires **caller permission plus role trust**; the role's permissions govern subsequent access. A third-party deputy uses an external ID condition to prevent another customer from misusing that trust.

Identity Federation

Federation trusts an external identity provider instead of duplicating identities. **Match the identity population**: workforce access uses centralized account assignments, customer applications use user authentication, and workloads use roles or token federation.

Core services



AWS Identity and Access Management (IAM)

Use **trust policies for who can assume a role** and permissions policies for allowed actions. Boundaries cap identity permissions without granting access; tag conditions support scalable ABAC.



AWS Security Token Service (AWS STS)

Use **temporary role credentials** instead of stored access keys. Choose AssumeRole for trusted AWS principals, AssumeRoleWithSAML for SAML, and AssumeRoleWithWebIdentity for OIDC tokens.



AWS IAM Identity Center (AWS Single Sign-On)

Use **centralized workforce access** across AWS accounts and applications. Permission sets provision account roles; troubleshoot assignments and provisioning when expected access is missing.

Get practice exams and lessons for AWS Certified Security - Specialty