AWS Certified Developer - Associate Cheat Sheet

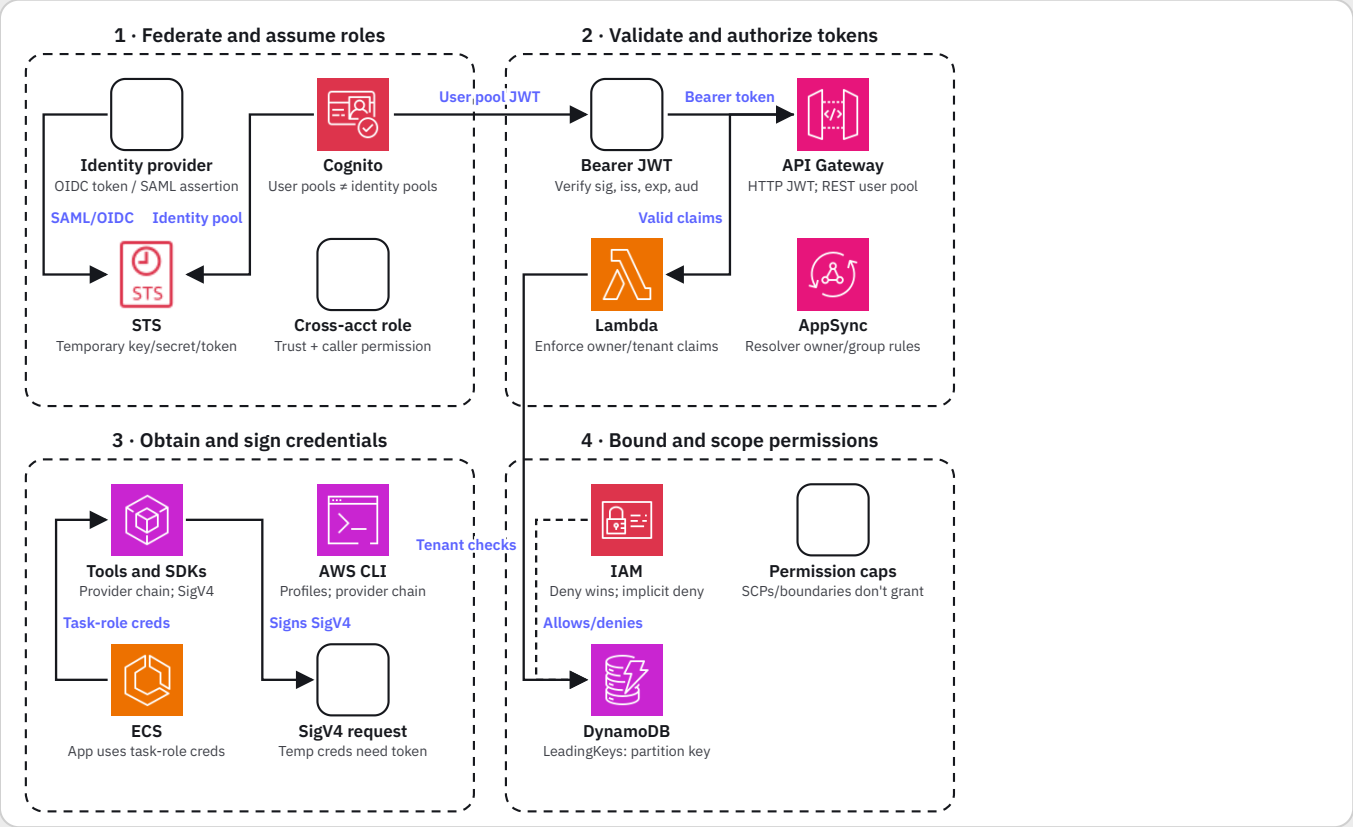
DVA-C02 · Domain 2 of 4

BlowStack

DOMAIN 2 · SECURITY · 26% OF THE EXAM · TASK STATEMENT 2.1

Implement authentication and/or authorization for applications and AWS services

Federate, validate, sign, authorize



1 Federate and assume roles

Use **user pools** for sign-in/JWTs; identity pools obtain AWS credentials. Direct federation uses AssumeRoleWithSAML or AssumeRoleWithWebIdentity. Cross-account AssumeRole needs caller permission and target-role trust.

2 Validate and authorize tokens

Use **access tokens + scopes** for API permissions; ID tokens describe sign-in. HTTP APIs support JWT authorizers; REST APIs support Cognito user-pool or Lambda authorizers. Custom authorization requires rules or code.

3 Obtain and sign credentials

Use **workload roles**, not stored keys. SDKs automatically obtain and refresh runtime role credentials and sign requests. Provider precedence can select unintended keys. ECS task roles serve app code; execution roles serve the agent.

4 Bound and scope permissions

Use **least privilege**: scope actions, resources and conditions. RBAC maps roles; ABAC matches trusted tags. Boundaries/SCPs cap grants. Lambda/AppSync enforce owner/tenant rules; DynamoDB LeadingKeys can restrict partition keys.

Exam traps

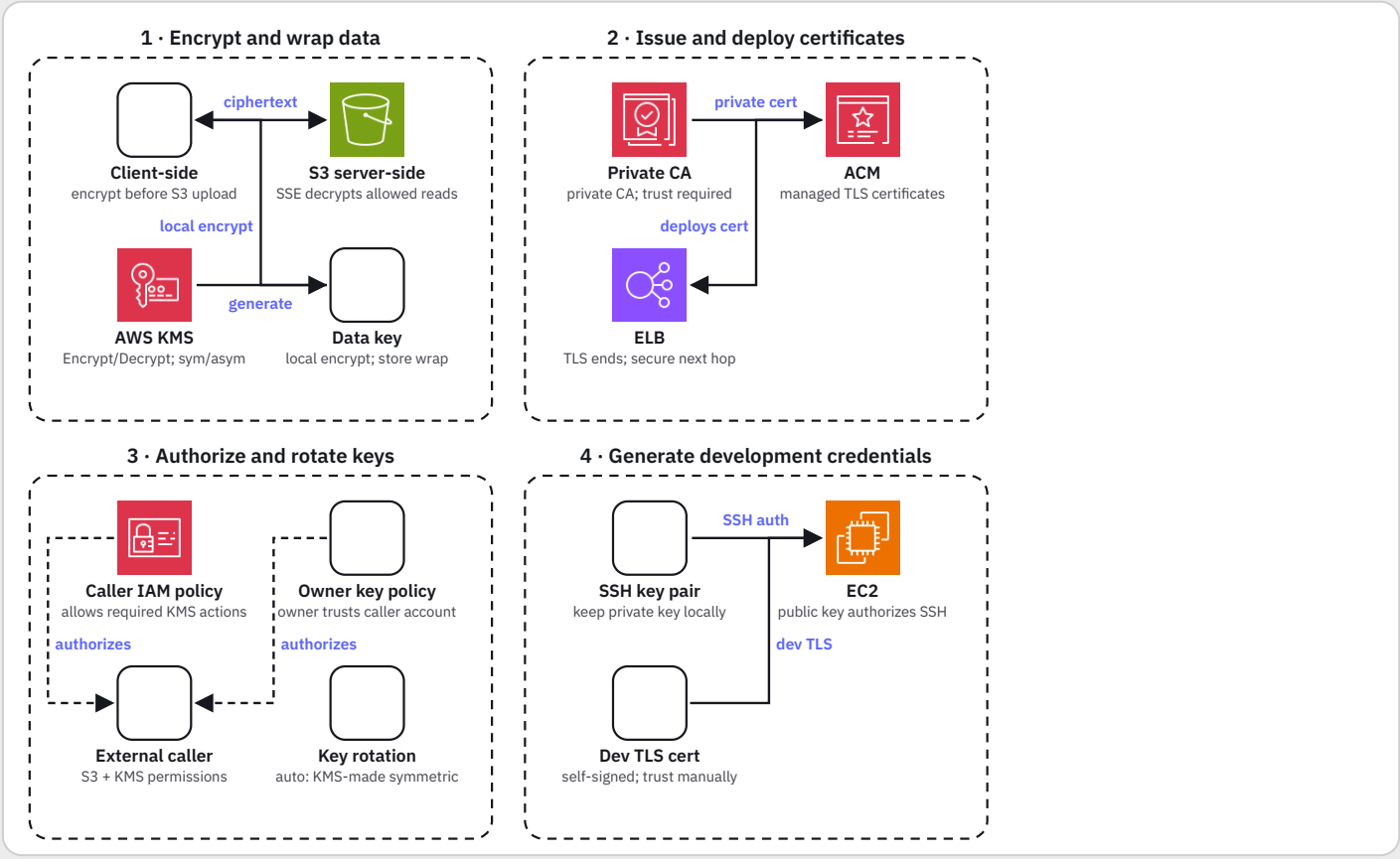
- **Decoding is not validation**: verify signature, issuer, expiry, intended audience/client and token type. Bearer tokens require TLS; possession enables use. Authentication alone does not grant access.
- **Confused deputy**: third-party role assumption uses a customer-specific ExternalId trust condition. AWS service delegation uses supported aws:SourceArn/aws:SourceAccount conditions.
- **mTLS is not authorization**: client certificates authenticate peers, not allowed operations. Microservices still need scoped workload roles or validated bearer tokens plus permission checks.

Get practice exams and lessons for AWS Certified Developer - Associate

DOMAIN 2 · SECURITY · TASK STATEMENT 2.2

Implement encryption by using AWS services

Encrypt, certify, authorize, rotate



1 Encrypt and wrap data

Use **envelope encryption** for bulk data: GenerateDataKey returns plaintext + wrapped key. Encrypt locally, erase plaintext, save wrapped key. SSE-KMS uses KMS; client-side encryption happens before upload.

2 Issue and deploy certificates

ACM renews eligible managed certificates; imported certificates need manual renewal. Private CA supplies private PKI, not public trust. Clients validate hostname and trust chain; backend TLS is configured separately.

3 Authorize and rotate keys

Cross-account use requires a customer managed KMS key, owner key policy AND caller IAM allow; S3 also needs object access. Automatic rotation applies to symmetric KMS-generated keys, not asymmetric keys.

4 Generate development credentials

SSH keys authenticate EC2 access; they are not TLS certificates. Generate locally or via EC2; retain the private key. Self-signed certificates suit development but require explicit client trust.

Exam traps

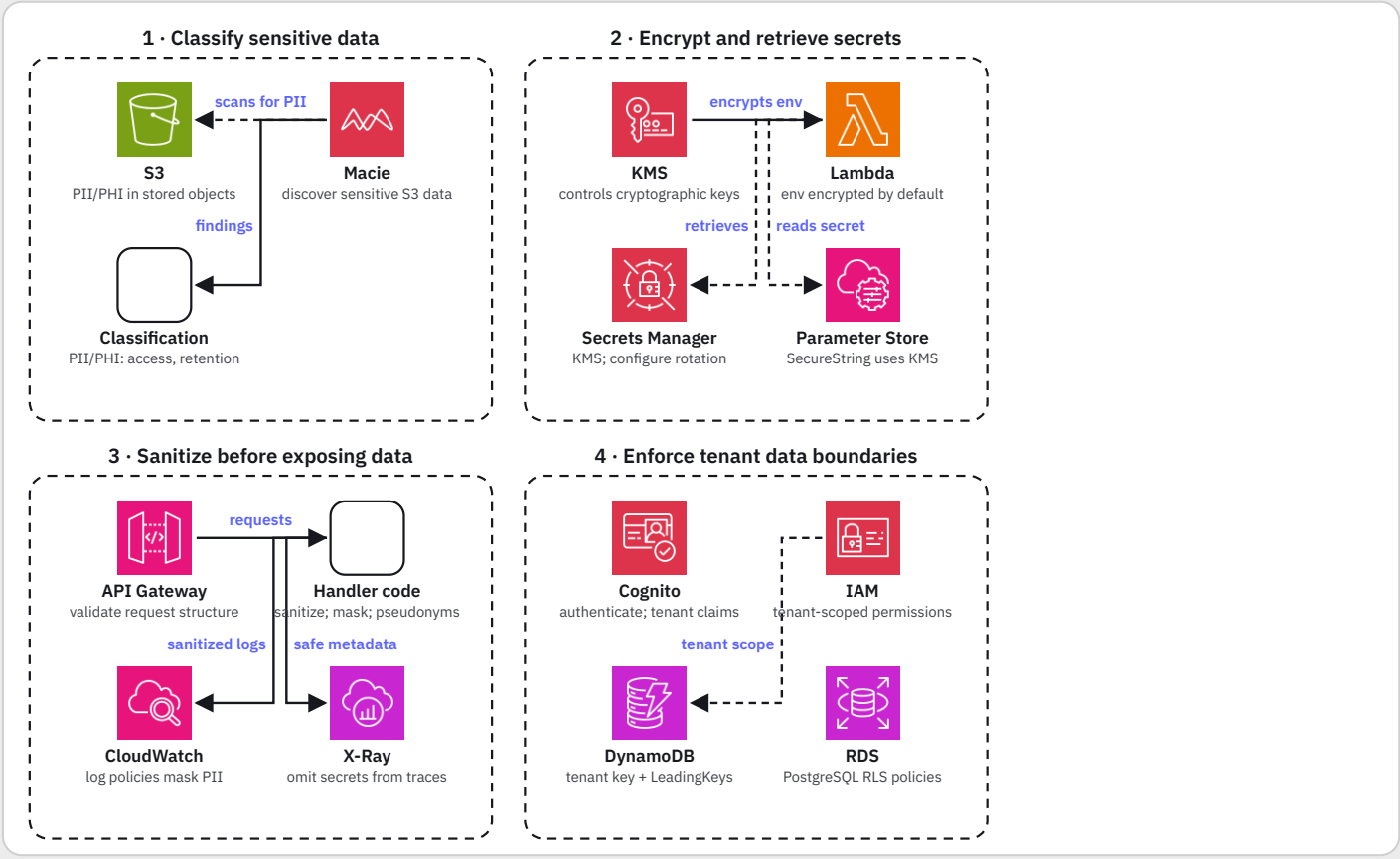
- **SSE is not TLS**: S3 encryption at rest does not protect network traffic or replace authorization. HTTPS protects transit; ELB-to-target encryption requires a separate HTTPS/TLS connection.
- Disabling **automatic rotation** stops new rotations, not decryption. Old key material remains; existing data is not re-encrypted. Asymmetric keys require manual replacement.
- **Symmetric KMS Encrypt** accepts at most 4 KB; use envelope encryption for bulk data. Asymmetric encryption uses the public key to encrypt and private key to decrypt; KMS retains its private key.

Get practice exams and lessons for AWS Certified Developer - Associate

DOMAIN 2 · SECURITY · TASK STATEMENT 2.3

Manage sensitive data in application code

Classify, encrypt, sanitize, isolate



1 Classify sensitive data

Use **classification** to set access, retention, and protection requirements. Macie discovers sensitive data in S3; discovery does not sanitize objects or establish regulatory compliance.

2 Encrypt and retrieve secrets

Lambda encrypts environment variables at rest automatically; client-side helpers require runtime decryption. Choose Secrets Manager for **automatic rotation**; Parameter Store SecureString has no built-in secret rotation.

3 Sanitize before exposing data

Minimize collection and sanitize in code before logging or tracing. Mask sensitive outputs and restrict CloudWatch unmask permission. API Gateway request validation is not application-level input sanitization.

4 Enforce tenant data boundaries

Use a **trusted tenant ID**, not a caller-supplied filter. IAM dynamodb:LeadingKeys can restrict partition-key access; PostgreSQL RLS enforces row access using trusted session identity. Cognito login alone is not isolation.

Exam traps

- **Encrypted environment variables** do not rotate credentials or hide plaintext from authorized code. Client-side encrypted values require explicit runtime decryption and kms:Decrypt permission.
- **Pseudonymization is not anonymization**: tokenized identifiers retain a reidentification path. Masking hides selected values; anonymization must also address linkage and indirect identifiers.
- **Partitioning is not authorization**: a tenant key or SQL WHERE filter alone is not an isolation boundary. Enforce tenant permissions; RDS IAM authentication does not provide row-level security.

Get practice exams and lessons for AWS Certified Developer - Associate

DOMAIN 2 · SECURITY · WHAT TO REMEMBER

Concepts and services of Security

Concepts to know

Effective Permissions

Effective access combines applicable identity policies, resource policies, and permission limits. **Explicit deny overrides allow**; permissions boundaries and service control policies limit permissions but do not grant them.

Temporary Credentials

Temporary credentials contain an access key ID, secret access key, and session token. **Use roles instead of embedded access keys** for workloads and federation; SDK credential providers obtain and refresh supported role credentials.

Envelope Encryption

Envelope encryption encrypts data locally with a data key and protects that key with a KMS key. **Choose GenerateDataKey for local encryption**; retain the encrypted data key with the ciphertext and discard the plaintext key after use.

Core services



AWS Identity and Access Management (IAM)

IAM controls AWS resource access. **Use a role for workload permissions**; its trust policy identifies who can assume it, while its permissions policies define allowed actions and resources.



AWS Key Management Service (AWS KMS)

KMS controls encryption keys and cryptographic operations. **Cross-account use requires both sides**: the key policy must permit the external account, and its IAM policy must authorize the caller.



Amazon Cognito

Cognito user pools authenticate application users and issue JWTs. **Choose identity pools for temporary AWS credentials**; choose user pools for application sign-in and token-based API access.

Get practice exams and lessons for AWS Certified Developer - Associate