Domain 1 – Development with AWS Services

Task Statement 1 – Develop code for application hosted on AWS					
		Confidence			
		None	Low	Med	High
Knowledge of …	Architectural patterns (for example, event-driven, microservices, monolithic, choreography, orchestration, fanout)				
	Idempotency				
	Differences between stateful and stateless concepts				
	Differences between tightly couples and loosely coupled comments				
	Fault-tolerant design patterns (for example, retires with exponential backoff and jitter, dead-letter queues)				
	Differences between synchronous and asynchronous patterns				
Skills in …	Creating fault-tolerant and resilient applications in a programming language (for example, Java, C#, Python, JavaScript, Typescript, Go)				
	Creating, extending and maintaining APIs (for example, response/request transformations, enforcing validation rules, overriding status codes)				
	Writing and running unit tests in development environments (for example, using AWS Serverless Application Model [AWS SAM])				
	Writing code to use messaging services				
	Writing code that interacts with AWS services by using APOs and AWS SDKs				
	Handling data streaming by using AWS services				

Task Statement 2 – Developer Code for AWS Lambda					
		Confidence			
		None	Low	Med	High
Knowledge of …	Event source mapping				
	Stateless applications				
	Unit testing				
	Event-driven architecture				
	Scalability				
	The access of private resources in VPCs from Lambda code				

Skills in ...	Configuring Lambda functions by defining environment variables and parameters (for example, memory, concurrency, timeout, runtime, handler, layers, extensions, triggers, destinations)				
	Handling the event lifecycle and errors by using code (for example, Lambda Destinations, dead-letter queues)				
	Writing and running test code by using AWS services and tools				
	Integration Lambda functions with AWS services				
	Tuning Lambda functions for optimal performance				

Task Statement 3 – Use data stores in application development					
		Confidence			
		None	Low	Med	High
Knowledge of ...	Relational and non-relational databases				
	Create, read, update and delete (CRUD) operations				
	High-cardinality partition keys for balanced partition access				
	Cloud storage options (for example, file, object, databases)				
	Database consistency model (for example, strongly consistent, eventually consistent)				
	Differences between query and scan operations				
	Amazon DynamoDB keys and indexing				
	Caching strategies (for example, write-through, read-through, lazy loading, TTL)				
	Amazon S3 tiers and lifecycle management				
	Differences between ephemeral and persistent data storage patterns				
Skills in ...	Serialising and deserialising data to provide persistence to a data store				
	Using, managing and maintaining data stores				
	Managing data lifecycles				
	Using data caching services				

Domain 2 – Security

Task Statement 1 – Implement authentication and/or authorisation for applications and AWS Services					
		Confidence			
		None	Low	Med	High
Knowledge of …	Identity federation (for example, Security Assertion Markup Language [SAML], OpenID Connect [OIDC], Amazon Cognito)				
	Bearer tokens (for example, JSON Web Token [JWT], OAuth, AWS Security Token Service [AWS STS])				
	The comparison of user pools and identity pools in Amazon Cognito				
	Resource-based policies, service policies, and principal policies				
	Role-based access control (RBAC)				
	Application authorisation that uses ACLs				
	The principle of least privilege				
	Difference between AWS managed polices and customer-managed policies				
	Identity and access management				
Skills in …	Using an identity provider to implement federated access (for example, Amazon Cognito, AWS Identity and Access Management [IAM])				
	Securing applications by using bearer tokens				
	Configuring programmatic access to AWS				
	Making authenticated calls to AWS services				
	Assuming an IAM role				
	Defining permissions for principals				

Task Statement 2 – Implement encryption by using AWS services					
		Confidence			
		None	Low	Med	High
Knowledge of …	Encryption at rest and in transit				
	Certificate management (for example, AWS Private Certificate Authority)				
	Key protection (for example, key rotation)				
	Differences between client-side encryption and server-side encryption				
	Differences between AWS managed and customer managed AWS Key Management Service (AWS KMS) keys				
Skills in …	Using encryption keys to encrypt or decrypt data				

	Generating certificates and SSH keys for development purposes				
	Using encryption across account boundaries				
	Enabling and disabling key rotation				

Task Statement 3 – Manage sensitive data in application code					
		Confidence			
		None	Low	Med	High
Knowledge of …	Data classification (for example, personally identifiable information [PII], protected health information [PHI])				
	Environment variables				
	Secrets management (for example, AWS Secrets Manager, AWS Systems Manager Parameter Store)				
	Secure credential handling				
Skills in …	Encrypting environment variables that contain sensitive data				
	Using secret management services to secure sensitive data				
	Sanitizing sensitive data				

Domain 3 – Deployment

Task Statement 1 – Prepare application artifacts to be deployed to AWS					
		Confidence			
		None	Low	Med	High
Knowledge of …	Ways to access application configuration data (for example, AWS AppConfig, Secrets Manager, Parameter Store)				
	Lambda deployment packaging, layers, and configuration options				
	Git-based version control tools (for example, Git, AWS, CodeCommit)				
	Container images				
Skills in …	Managing the dependencies of the code module (for example, environment variables, configuration files, container images) within the package				
	Organising files and a directory structure for application deployment				
	Using code repositories in deployment environments				
	Applying application requirements for resources (for example, memory, cores)				

Task Statement 2 – Test applications in development environments					
		Confidence			
		None	Low	Med	High
Knowledge of …	Features in AWS services that perform application deployment				
	Integration testing that uses mock endpoints				
	Lambda versions and aliases				
Skills in …	Testing deployed code by using AWS services and tools				
	Performing mock integration for APIs and resolving integration dependencies				
	Testing applications by using development endpoints (for example, configuring stages in Amazon API Gateway)				
	Deploying application stack updates to existing environments (for example, deploying an AWS SAM template to a different staging environment)				

Task Statement 3 – Automate deployment testing					
		Confidence			
		None	Low	Med	High
Knowledge of …	API Gateway stages				

	Branches and actions in the continuous integration and continuous delivery (CI/CD) workflow				
	Automated software testing (for example, unit testing, mock testing)				
Skills in …	Creating application test events (for example, JSON payloads for testing Lambda, API Gateway, AWS SAM resources)				
	Deploying API resources to various environments				
	Creating application environments that use approved versions for integration testing (for example, lambda aliases, container image tags, AWS Amplify branches, AWS Copilot environments)				
	Implementing and deploying infrastructure as code (IaC) templates (for example, AWS SAM templates, AWS CloudFormation templates)				
	Managing environments in individual AWS services (for example, differentiating between development, test and production in API Gateway)				

Task Statement 4 – Deploy code by using AWS CI/CD services					
		Confidence			
		None	Low	Med	High
Knowledge of …	Git-based version control tools (for example, Git, AWS CodeCommit)				
	Manual and automated approvals in AWS CodePipeline				
	Access application configurations from AWS AppConfig and Secrets Manager				
	CI/CD workflows that use AWS services				
	Application deployment that uses AWS services and tools (for example, CloudFormation, AWS Cloud Development Kit [AWS CDK], AWS SAM, AWS CodeArtifact, AWS Copilot, Amplify, Lambda)				
	Lambda deployment packaging options				
	API Gateway stages and custom domains				
	Deployment strategies (for example, canary, blue/green, rolling)				
Skills in …	Updating existing IaC templates (for example, AWS SAM templates, CloudFormation templates)				
	Managing application environments by using AWS services				

	Deploying an application version by using deployment strategies				
	Committing code to a repository to invoke build, test, and deployment actions				
	Using orchestrated workflows to deploy code to different environments				
	Performing application rollbacks by using existing deployment strategies				
	Using labels and branches for version and release management				
	Using existing runtime configurations to create dynamic deployments (for example, using staging variables from API Gateway in Lambda functions)				

Domain 4 – Troubleshooting and Optimisation

Task Statement 1 – Assist in a root cause analysis					
		Confidence			
		None	Low	Med	High
Knowledge of …	Logging and monitoring systems				
	Languages for log queries (for example, Amazon Cloudwatch Logs Insights)				
	Data visualisations				
	Code analysis tools				
	Common HTTP error codes				
	Common exceptions generated by SDKs				
	Service maps in AWS X-Ray				
Skills of …	Debugging code to identify defects				
	Interpreting application metrics, logs and traces				
	Querying logs to find relevant data				
	Implementing custom metrics (for example, Cloudwatch embedded metric format [EMF])				
	Reviewing application health by using dashboards and insights				
	Troubleshooting deployment failures by using service output logs				

Task Statement 2 – Instrument code for observability					
		Confidence			
		None	Low	Med	High
Knowledge of …	Distributed tracing				
	Differences between logging, monitoring, and observability				
	Structured logging				
	Application metrics (for example, custom, embedded, built-in)				
Skills in …	Implementing an effective logging strategy to record application behaviour and state				
	Implementing code that emits custom metrics				
	Adding annotations for tracing services				
	Implementing notification alerts for specific actions (for example, notifications about quota limits or deployment completions)				
	Implementing tracing by using AWS services and tools				

Task Statement 3 – Optimise applications by using AWS services and features					
		Confidence			
		None	Low	Med	High

Knowledge of ...	Caching				
	Concurrency				
	Messaging services (for example, Amazon Simple Queue Service [Amazon SQS], Amazon Simple Notification Service [Amazon SNS])				
Skills in ...	Profiling application performance				
	Determining minimum memory and computer power for an application				
	Using subscription filter policies to optimise messaging				
	Caching context based on request headers				