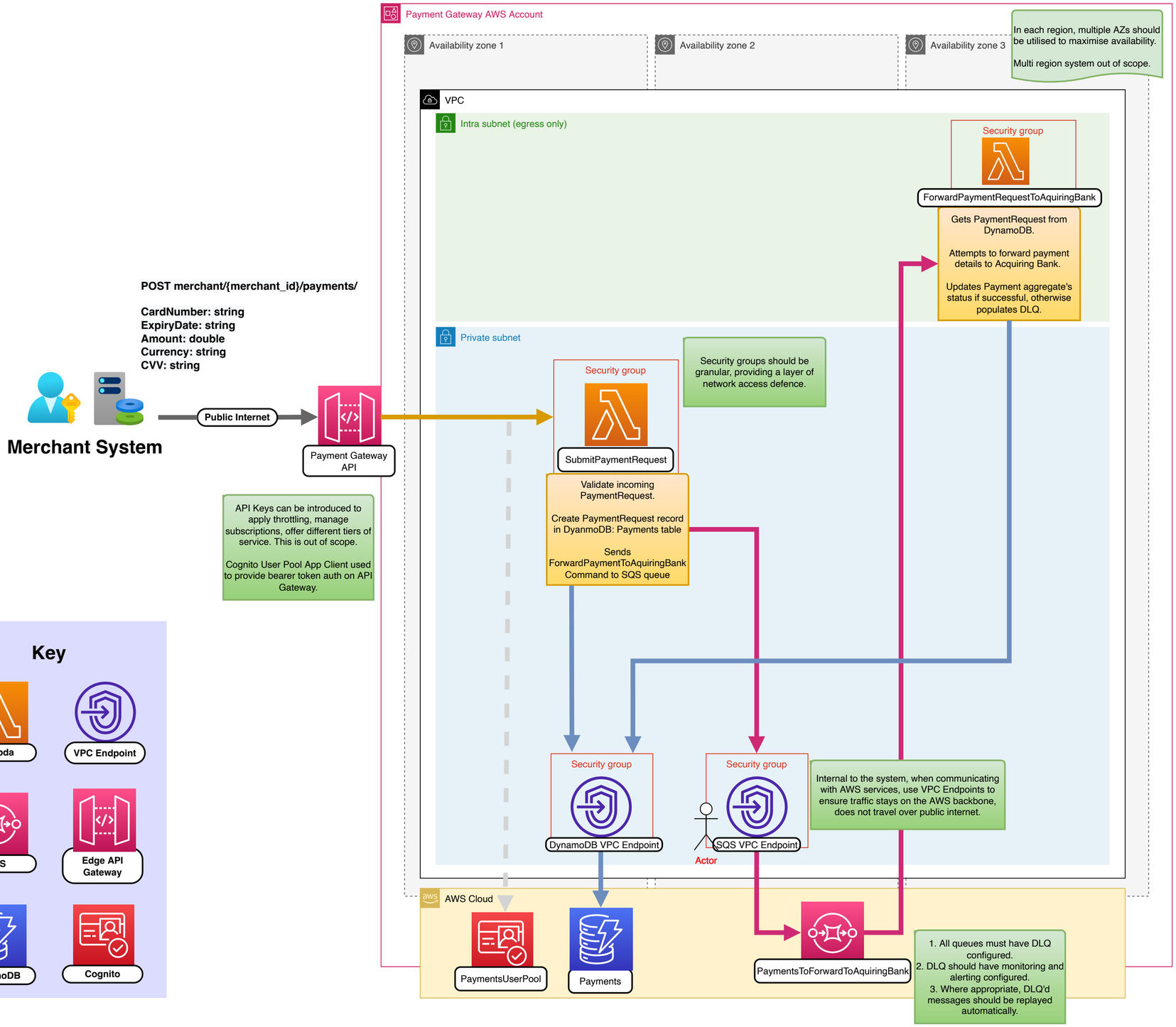
Merchant: Make Payment Request



Reasoning

Why Favour Managed Services / Serverless?

Security, speed of delivery, cost to maintain, can support quite high load.

Once scaling capacity of managed services has been reached and pushed, bottlenecks can be identified and system can be redesigned as required.

Why SQS?

Availability: Enables us to accept payment requests from merchants when the acquiring bank's APIs are unavailable. Failed requests can be replayed from the DLQ.

If the acquiring bank have down time we can pause processing messages from the queue automatically or via configuration.

Why Lambda?

Can rapidly scale out to support very high workloads.

If greater performance is required, we can migrate to other lower level services.

Why DynamoDB?

Can rapidly scale out to support very high throughput workloads.

Redundancy built in to the product.

Items will be accessed by primary key (Payment ID) and we do not require complex query capabilities.

Why Cognito User Pool + App Client?

Low maintenance, complete offering that supports most common authN / authZ requirements out-of-the-box (e.g. token invalidation).

Simple integration with AWS services, API Gateway in this case.

Simple to extend with lambda plug ins if required.

Why Edge API Gateway?

Low latency, backed by CloudFront.

REST API exposed over HTTPS with bearer authentication is an industry standard. This makes it simple to onboard clients to the system.

Supports API Keys, blue/green deployments, WAF, cache configuration, and more.

Integrates with many AWS services.

Alternatives: public facing ELB(s), shared SQS queues, shared SNS topics, Lambda HTTP endpoints