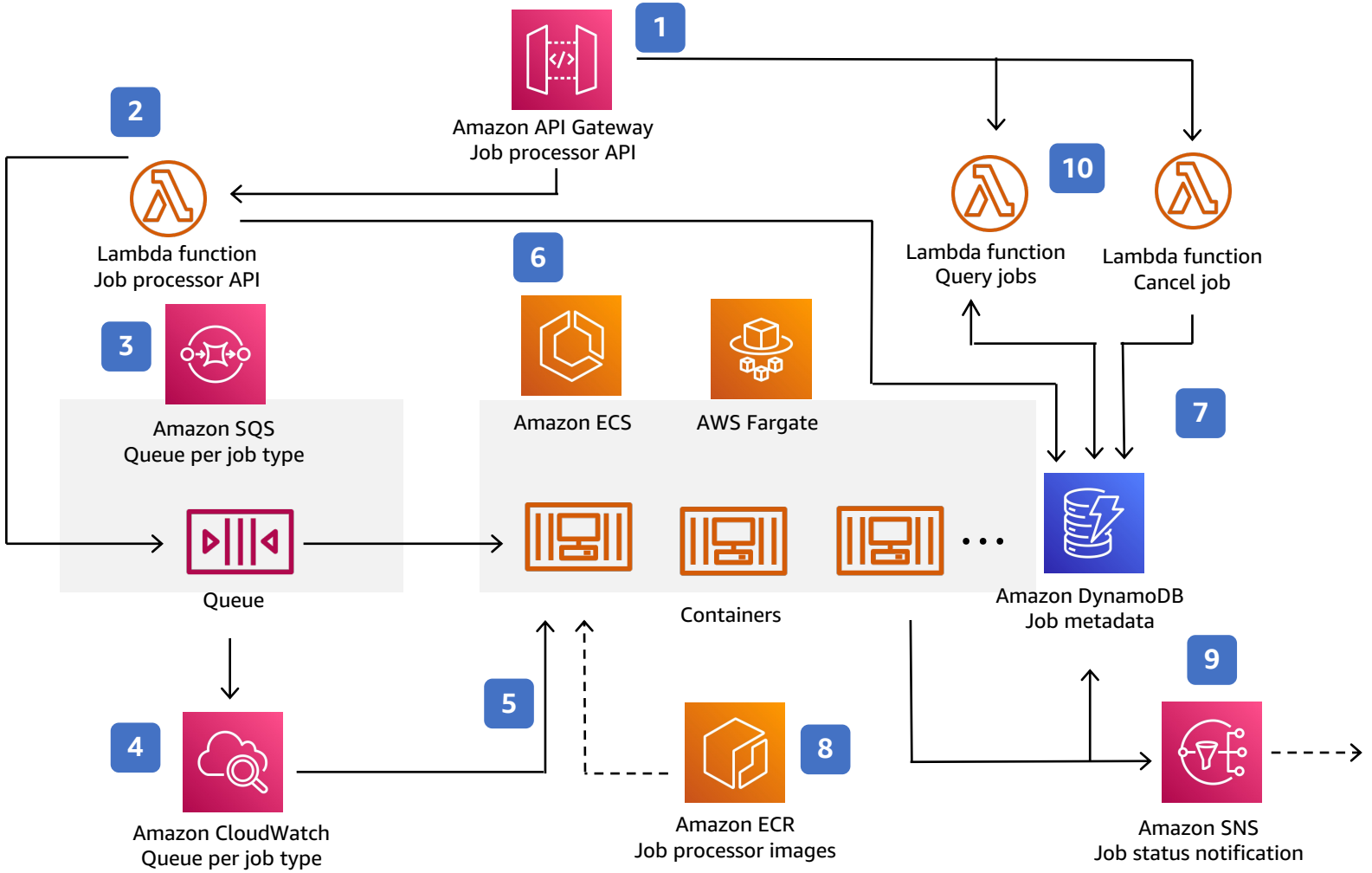
Autoscaling Asynchronous Job Queues

Using Amazon SQS, AWS Fargate, Amazon DynamoDB, and AWS Lambda

Perform background processing jobs such as document generation, extract, transform, and load (ETL) tasks, inference, and more with AWS Fargate



0 Each job type requires an **Amazon Elastic Container Services (Amazon ECS)** instance (desiredCount = 0), **Amazon Simple Queue Service (Amazon SQS)** queue, and associated **Amazon CloudWatch a**larms. See step 4.

1 **Amazon API Gateway** provides an interface to add, query and cancel jobs.

2 The “add job” **Lambda** function creates the job state item in **DynamoDB**, and enqueues the job ID on SQS.

3 The **SQS** job queue contains messages with the IDs of the jobs to be processed.

4 **Amazon CloudWatch** monitors the SQS queue depth. Alarms are triggered when the depth exceeds a configurable value, and when the queue is empty.

5 **Amazon ECS** responds to the CloudWatch alarms, changing the desired count of ECS and scaling the number of parallel jobs.

6 **AWS Fargate** runs job processing containers from **Amazon Elastic Container Registry (Amazon ECR)**. The containers poll **SQS** for job IDs, read the job item from **DynamoDB**, and perform processing, updating the job status in **DynamoDB** and publishing to an **Amazon Simple Notification Service (Amazon SNS)** topic on a status change.

7 A **DynamoDB** table stores the state, parameters and results of the jobs, by job ID.

8 **Amazon ECR** stores the job processing container images. An **SNS** topic publishes job items on state change notifications to subscribers.

9 An **SNS** topic publishes job items on state change notifications to subscribers.

10 Two “control" **Lambda** functions allow jobs to be queried and canceled from **API Gateway** by reading or updating the job item in **DynamoDB**.

AWS Reference Architecture