Assumptions

Frequency: export once a month = rough estimate of frequency, not a hard requirement -> will allow more often

Data Store: Amazon Aurora Postgres (also works with Aurora MySQL and RDS Postgres, sorry RDS MySQL)

DLQs for SQS queues, EventBridge Pipes and async invoked lambdas are implicitly hidden for readability

Results via Email: depending on the size of the file it might not be ideal to send the results directly via email, but rather a download link, so I included that but one could easily send the file directly

API: I chose an API Gateway HTTP Api & Lambda Function to handle HTTP requests but this can obviously be replaced by other methods (e.g. ALB/NLB + ECS/EKS/EC2, CF + FURL, etc)

Firewall: a WAF should be added to whatever HTTP handler is used

Email: I chose SES to send emails, like many email providers it uses events/webhooks for delivery status updates. Polling could also be used

Other Options

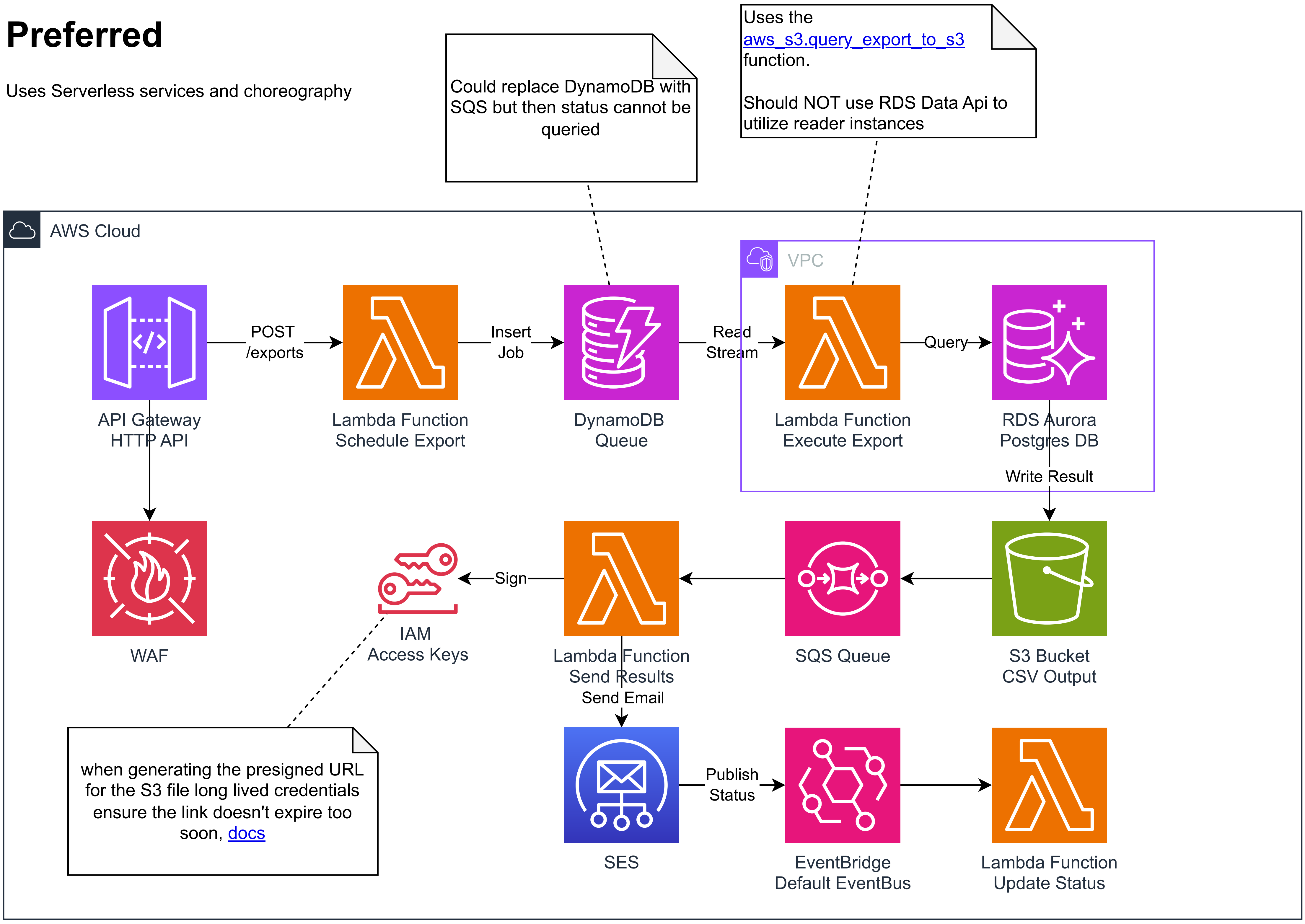
Some options not included:

- RDS Activity streams to Kinesis as a Queue/Outbox

- Athena to query data stores and produce exports

Preferred

Uses Serverless services and choreography



Could replace DynamoDB with SQS but then status cannot be queried

Uses the aws_s3.query_export_to_s3 function.

Should NOT use RDS Data Api to utilize reader instances

when generating the presigned URL for the S3 file long lived credentials ensure the link doesn't expire too soon, docs

👍 Pro

- Scale to zero for infrequently used feature

- cheap per execution

- scalable to many concurrent requests

- batching possible

- job status available via API

- deduplication possible

👎 Con

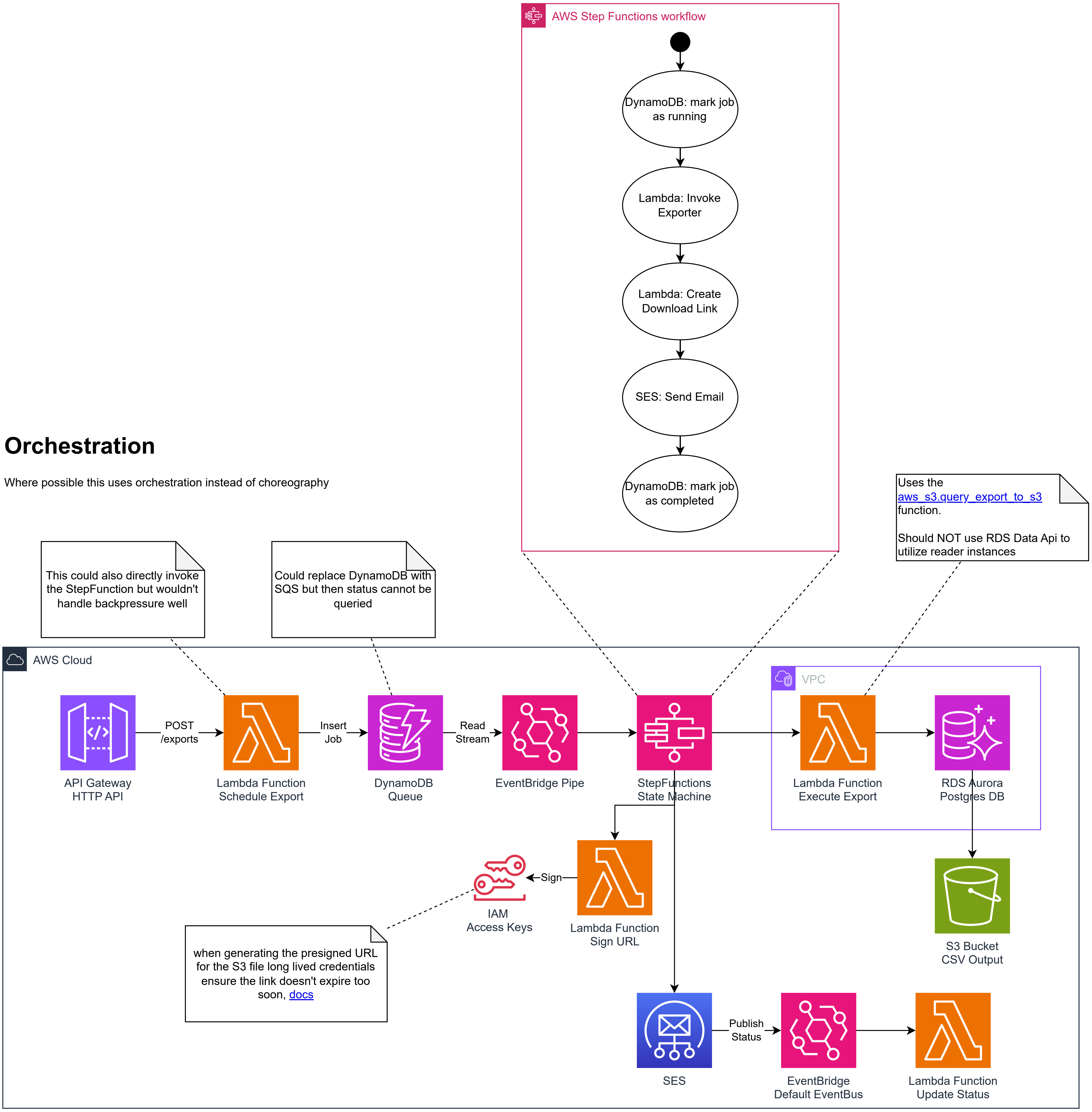
- more services than other solutions

- long queries might exceed lambda timeout. Although >15min queries are questionable anyway

- large exports (>6GB) might result in multiple files, seems unlikely

Orchestration

Where possible this uses orchestration instead of choreography


AWS Step Functions workflow
DynamoDB: mark job as running
Lambda: Invoke Exporter
Lambda: Create Download Link
SES: Send Email
DynamoDB: mark job as completed
AWS Cloud
API Gateway HTTP API
POST /exports
Lambda Function Schedule Export
Insert Job
DynamoDB Queue
Read Stream
EventBridge Pipe
StepFunctions State Machine
VPC
Lambda Function Execute Export
RDS Aurora Postgres DB
S3 Bucket CSV Output
IAM Access Keys
Sign
Lambda Function Sign URL
SES
Publish Status
EventBridge Default EventBus
Lambda Function Update Status

This could also directly invoke the StepFunction but wouldn't handle backpressure well

Could replace DynamoDB with SQS but then status cannot be queried

Uses the aws_s3.query_export_to_s3 function.

Should NOT use RDS Data Api to utilize reader instances

when generating the presigned URL for the S3 file long lived credentials ensure the link doesn't expire too soon, docs

👍 Pro

- Scale to zero for infrequently used feature

- cheap per execution

- scalable to many concurrent requests

- good visibility in workflow

- job status available via API

- can handle multiple files from export

- deduplication possilble

👎 Con

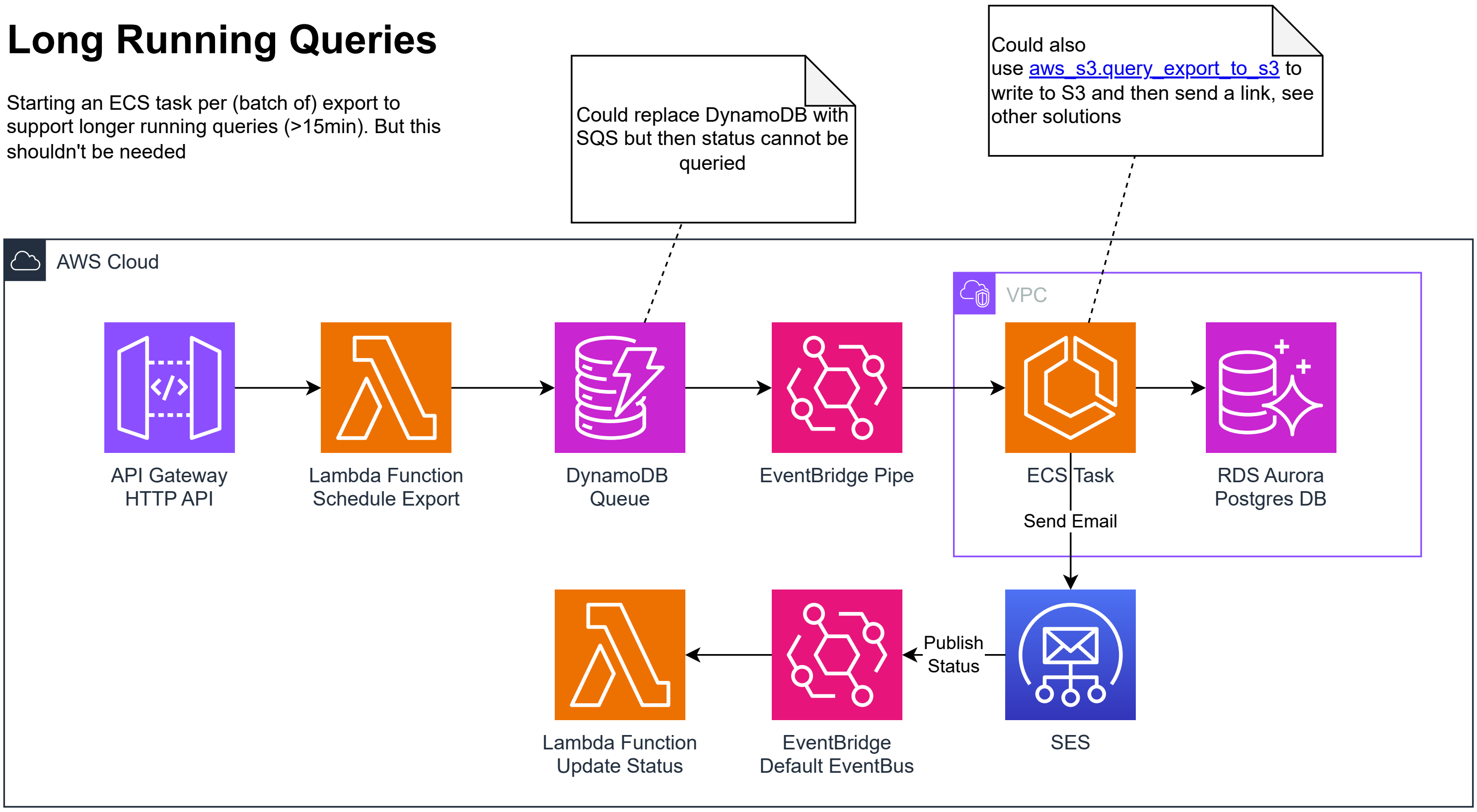
- more services than other solutions

- long queries might exceed lambda timeout. Although >15min queries are questionable anyway

- due to SES status notifications, uses mix of orchestration & choreography

Long Running Queries

Starting an ECS task per (batch of) export to support longer running queries (>15min). But this shouldn't be needed


Could replace DynamoDB with SQS but then status cannot be queried
Could also use aws_s3.query_export_to_s3 to write to S3 and then send a link, see other solutions
AWS Cloud
VPC
API Gateway HTTP API
Lambda Function Schedule Export
DynamoDB Queue
EventBridge Pipe
ECS Task
RDS Aurora Postgres DB
Send Email
SES
Publish Status
EventBridge Default EventBus
Lambda Function Update Status

👍 Pro

- Scale to zero for infrequently used feature

- can run very long queries

- fewer services used

- when replacing EventBridge Pipes with a Lambda Function one can batch exports

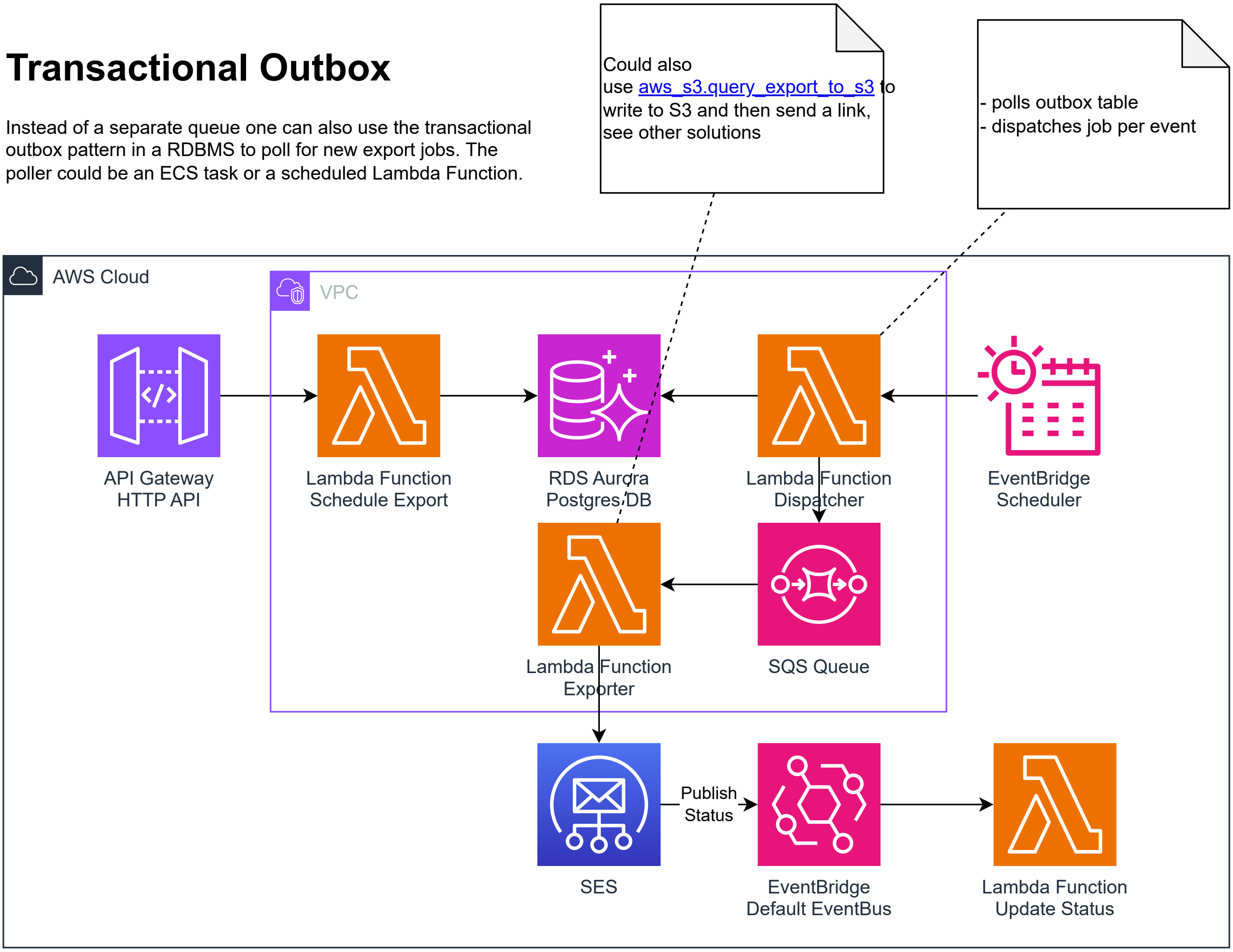
👎 Con

- less resilient export jobs that do query & send

- more expensive per execution

Transactional Outbox

Instead of a separate queue one can also use the transactional outbox pattern in a RDBMS to poll for new export jobs. The poller could be an ECS task or a scheduled Lambda Function.


Could also use aws_s3.query_export_to_s3 to write to S3 and then send a link, see other solutions
- polls outbox table
- dispatches job per event
AWS Cloud
VPC
API Gateway HTTP API
Lambda Function Schedule Export
RDS Aurora Postgres DB
Lambda Function Dispatcher
EventBridge Scheduler
Lambda Function Exporter
SQS Queue
SES
Publish Status
EventBridge Default EventBus
Lambda Function Update Status

👍 Pro

- Scale to zero for infrequently used feature

- cheap per execution

- scalable to many concurrent requests

- batching possible

- deduplication possible

- job status available via API

👎 Con

- many useless polling executions

- manual implementation of job timeouts etc in DB

- long queries might exceed lambda timeout. Although >15min queries are questionable anyway