ArchGuard
AI-assisted · Expert-informed

AWS WELL-ARCHITECTED REVIEW

Architecture Review Report

PROJECT
Synthr API — Production

REVIEW DATE	ENVIRONMENT	INPUT
2026-04-05	Production	Terraform + brief

MEDIUM RISK OVERALL

62	62	89	74
SECURITY	RELIABILITY	COST	OPERATIONS

11 findings total: 5 HIGH 4 MEDIUM 2 LOW

Prepared by: ArchGuard (AI-assisted, expert-informed)
Scope: Terraform code analysis only — no AWS credentials, no state file, no infrastructure access

1. Project Overview

Project	Synthr API
Environment	Production
Input received	Terraform (main.tf, variables.tf, outputs.tf) + architecture brief
Review date	2026-04-05
Prepared by	ArchGuard (AI-assisted, expert-informed)

Key assumptions

· The workload actively serves B2B customers in production — all findings are time-sensitive.

· ECS Fargate task scheduling managed outside Terraform — no scheduler resource visible.

· No CI/CD pipeline provided — deployment determinism findings assume no external pinning.

· No VPC endpoints defined — Lambda routes AWS service traffic via the public internet gateway.

---

2. Executive Summary

Synthr API is a document-processing B2B SaaS that ingests customer files via API Gateway, processes them asynchronously through an SQS-backed Lambda pipeline, and stores state in RDS Postgres. The overall risk posture is **Medium** across Security, Reliability, and Operations — driven by three production-grade gaps that require immediate attention.

Finding	Pillar	Impact
RDS directly reachable from internet on port 5432	Security	Full database exposure to any attacker
Lambda role grants wildcard access to all S3, SQS, DynamoDB	Security	Full account blast radius on any Lambda compromise
SQS visibility timeout (30s) < Lambda timeout (300s)	Reliability	Systematic duplicate processing — direct cause of cost creep

The only notable positive: the processed-output S3 bucket is correctly protected with a full public access block.

---

3. Scorecard

Pillar	Score	Risk Level
Security	**62 / 100**	**Medium**
Reliability	**62 / 100**	**Medium**
Cost	**89 / 100**	**Low**
Operational Excellence	**74 / 100**	**Medium**

---

4. Top Priority Findings

These are the issues most likely to cause a material incident if unaddressed.

HIGH · Security · Confidence: High

RDS instance directly accessible from the public internet

Evidence

```
aws_security_group.rds: cidr_blocks = ["0.0.0.0/0"] on port 5432.
aws_db_instance.main: publicly_accessible = true.
aws_db_subnet_group.main references only aws_subnet.public_a (map_public_ip_on_launch = true).
```

Why it matters

Any attacker who discovers the RDS endpoint — exposed in outputs.tf and injected into every Lambda environment — can attempt direct authentication against the production Postgres database. The only protection is the database password, stored as a plain Lambda environment variable. A B2B platform undergoing Series A due diligence with a publicly reachable production database is a material finding in any security review.

Recommendation

1. Set publicly_accessible = false on aws_db_instance.main.
2. Replace cidr_blocks = ["0.0.0.0/0"] with source_security_group_id = aws_security_group.lambda.id.
3. Add private subnets and move RDS there.
4. Add a NAT Gateway so Lambda retains outbound access.

HIGH · Security · Confidence: High

Lambda execution role grants wildcard access to S3, SQS, and DynamoDB

Evidence

```
aws_iam_role_policy.lambda_policy: "s3:*" on "*", "sqs:*" on "*", "dynamodb:*" on "*".
Both api_handler and worker share this single role.
```

Why it matters

If either Lambda function is compromised, an attacker inherits full control of every S3 bucket, SQS queue, and DynamoDB table in the AWS account. With s3:* on *, they can exfiltrate all customer uploads. This is the blast-radius problem most commonly flagged in investor security reviews of early-stage SaaS.

Recommendation

1. Create two separate IAM roles: synthr-api-handler and synthr-doc-worker.
2. api-handler: s3:PutObject on uploads/*, sqs:SendMessage on jobs queue, dynamodb scoped actions.
3. doc-worker: s3:GetObject on uploads, s3:PutObject on processed, sqs scoped actions.
4. Remove all wildcard Resource values.

HIGH · Reliability · Confidence: High

RDS single-AZ with automated backups disabled

Evidence

```
aws_db_instance.main: multi_az not set (defaults false). backup_retention_period not set (defaults 0).
skip_final_snapshot = true. Subnet group references only one AZ — multi-AZ topologically impossible.
```

Why it matters

The architecture brief describes a 20-minute outage caused by an RDS issue. With multi_az = false, that outage was a host-level failure requiring a new instance — the exact failure mode multi-AZ eliminates with automatic failover in 60–120 seconds. With backup_retention_period = 0, if the instance is corrupted or deleted, all customer job records are permanently gone.

Recommendation

1. Set multi_az = true.
2. Set backup_retention_period = 7.
3. Set storage_encrypted = true with kms_key_id.
4. Add aws_subnet.private_b in eu-west-1b — required for multi-AZ.
5. Remove skip_final_snapshot = true.

HIGH · Reliability · Confidence: High

SQS visibility timeout (30s) is shorter than the Lambda worker timeout (300s)

Evidence

```
aws_sqs_queue.jobs: visibility_timeout_seconds not set (defaults 30s).
aws_lambda_function.worker: timeout = 300 seconds. batch_size = 10.
```

Why it matters

Every document taking more than 30 seconds to process is processed at least twice. For a B2B extraction pipeline, this produces duplicate records in RDS, duplicate webhook deliveries, and Lambda + S3 + RDS costs that are a multiple of the true workload volume. This self-amplifying duplication loop is the most likely explanation for the cost creep the team has observed.

Recommendation

1. Set visibility_timeout_seconds = 360 (worker timeout + 20% buffer).
2. Set bisect_batch_on_function_error = true.
3. Create jobs_dlq and attach via redrive_policy with maxReceiveCount = 3.

5. Detailed Findings

5.1 Security

MEDIUM · Security · Confidence: High

Database password stored as plaintext Lambda environment variable

Evidence

```
api_handler: DB_PASSWORD = var.db_password, API_KEY_SECRET = var.api_key_secret in environment.variables.
worker: DB_PASSWORD = var.db_password. ECS task definition passes DB_PASSWORD as plain environment entry.
```

Why it matters

Lambda environment variables are visible in plaintext to any IAM principal with lambda:GetFunction. ECS task definition environment variables are stored in plaintext across all revision history. Any developer with read access can retrieve the production database password without touching the database.

Recommendation

1. Create aws_secretsmanager_secret resources for DB credentials and API key.
2. Replace environment.variables entries with runtime secretsmanager:GetSecretValue calls.
3. Rotate DB_PASSWORD and API_KEY_SECRET after migration.

5.2 Reliability

MEDIUM · Reliability · Confidence: High

No dead-letter queue on the jobs SQS queue

Evidence

```
aws_sqs_queue.jobs: no redrive_policy block. aws_lambda_event_source_mapping.worker_sqs: no destination_config.
A message causing repeated Lambda failures retries until the 4-day SQS retention period expires.
```

Why it matters

A single malformed document causes that message to consume worker capacity continuously for up to 4 days, blocking legitimate jobs. With no DLQ there is no mechanism to inspect, quarantine, or replay failed messages. A customer job silently disappears with no recovery path.

Recommendation

1. Create aws_sqs_queue.jobs_dlq with message_retention_seconds = 1209600.
2. Add redrive_policy with maxReceiveCount = 3.
3. Add CloudWatch alarm on jobs_dlq ApproximateNumberOfMessagesNotVisible.

5.3 Cost Optimization

MEDIUM · Cost · Confidence: High

DynamoDB sessions table has no TTL — grows unboundedly

Evidence

```
aws_dynamodb_table.sessions: billing_mode = "PAY_PER_REQUEST" with no ttl block.
Session records accumulate with no deletion mechanism.
```

Why it matters

DynamoDB PAY_PER_REQUEST charges per GB stored. The table's storage footprint and cost grow monotonically with no ceiling. This is consistent with the unexplained bill creep the team reported. At scale, reads against unpruned records also consume more read request units.

Recommendation

1. Add ttl block: attribute_name = "expires_at", enabled = true.
2. Set expires_at in application code on write.
3. Add point_in_time_recovery { enabled = true }.

LOW · Cost · Confidence: Low

Lambda worker memory allocation is unvalidated

Evidence

```
aws_lambda_function.worker: memory_size = 1024 MB, timeout = 300 seconds. No profiling data available.
```

Why it matters

Lambda charges per GB-second. At 1024 MB and worst-case 300s, each execution costs ~300 GB-seconds. A 25% memory reduction yields proportional cost savings if the workload supports it.

Recommendation

1. Run AWS Lambda Power Tuning on synthr-doc-worker with representative production payloads.
2. Set memory_size to the optimal cost-duration tradeoff value.

5.4 Operational Excellence

HIGH · Operations · Confidence: High

No CloudWatch alarms exist for any service in the stack

Evidence

```
No aws_cloudwatch_metric_alarm resources in main.tf.
No alarms for Lambda errors/throttles, SQS queue depth/age, RDS CPU/storage/connections, or ECS failures.
```

Why it matters

The team described a 20-minute RDS outage. With no alarms, that outage was discovered through user impact — not automated detection. Every failure mode in this stack is currently invisible until a customer reports a problem. Mean time to detect is unbounded.

Recommendation

1. Lambda error alarms: Errors > 5 in 5 minutes on both functions.
2. SQS alarm: ApproximateAgeOfOldestMessage > 300s on jobs queue.
3. RDS alarms: CPUUtilization > 80%, FreeStorageSpace < 10 GB, DatabaseConnections > 80.
4. Create aws_sns_topic and route all alarms to on-call email or PagerDuty.

MEDIUM · Operations · Confidence: High

No access logging on the API Gateway stage

Evidence

```
aws_apigatewayv2_stage.default: no access_log_settings block. No CloudWatch log group for API Gateway.
```

Why it matters

Without access logs there is no record of which clients called which endpoints, HTTP status codes, or latency per route. Removes ability to investigate customer complaints, detect abuse, or correlate API traffic with downstream failures.

Recommendation

1. Create CloudWatch log group "/aws/apigateway/synthr-api" with retention 30 days.
2. Add access_log_settings to aws_apigatewayv2_stage.default.
3. Set detailed_metrics_enabled = true on default_route_settings.

LOW · Operations · Confidence: High

ECS container image uses the mutable :latest tag

Evidence

```
Container definition: image = "...synthr-reporter:latest". The :latest tag is mutable — any ECR push replaces
the image on next task execution without a task definition update.
```

Why it matters

An untested image pushed to ECR runs on the next nightly report without any deployment gate. A broken image causes silent nightly job failures with no pinned version to roll back to.

Recommendation

1. Replace :latest with an immutable digest reference: @sha256:.
2. Update the digest in CI/CD after a validated image build.
3. Enable ECR image tag immutability on the synthr-reporter repository.

---

6. Assumptions & Missing Information

ECS task scheduling not in Terraform

No EventBridge Scheduler or CloudWatch event rule present. Findings about the ECS task are limited to the task definition.

No WAF configuration visible

No aws_wafv2_web_acl present. The API Gateway is assessed as unprotected.

VPC endpoint coverage unknown

No aws_vpc_endpoint resources defined. Lambda routes S3, SQS, DynamoDB traffic via the public internet gateway.

Terraform state backend not visible

If state is stored in an unencrypted S3 bucket, db_password and api_key_secret are stored in plaintext state.

Application-layer auth not assessed

No Lambda authorizer visible on API Gateway routes. Auth assumed to be in Lambda function code, which was not provided.

---

7. Risk Distribution

Severity	Count
HIGH	5
MEDIUM	4
LOW	2
Total	**11**

---

8. Recommended Next Steps

1. Fix RDS open to the internet

Set publicly_accessible = false and replace the 0.0.0.0/0 security group rule with a Lambda-sourced rule. Highest security impact, reversible in minutes.

2. Fix SQS visibility timeout

Set visibility_timeout_seconds = 360 on aws_sqs_queue.jobs. One-line change that stops duplicate processing immediately — the most direct lever for the cost creep observed.

3. Enable RDS multi-AZ and backups

Set multi_az = true and backup_retention_period = 7. Directly addresses the past 20-minute outage.

4. Restrict Lambda IAM permissions

Split the shared execution role into two scoped roles. The security change most likely to be flagged in investor due diligence — requires only IAM policy updates.

5. Add a baseline CloudWatch alarm set

Without alarms none of the above fixes can be operationally validated. Add Lambda error rate, SQS age of oldest message, and RDS free storage alarms before any changes go to production.

---

9. Disclaimer

This report is a first-pass architecture review based on the Terraform code and architecture brief provided. It does not constitute a full security audit, compliance certification, or guarantee of correctness. Coverage is limited to what is observable in the provided inputs.

ArchGuard analyses Terraform code only. No AWS credentials required. No state file access. No infrastructure changes — ever. AI generates the analysis; you decide what to act on.