Architecture Decision Record: BigCorp Integration

ADR-001 Status: Approved Date: 2024-03-15

Context

BigCorp integration requires processing large volumes of data (500k records per import) with strict security and performance requirements. We need to design a scalable, secure, and maintainable architecture.

Decision

We will implement an event-driven architecture using AWS services with the following components:

API Layer

- API Gateway for request handling
- Custom authorizer for HMAC validation
- Rate limiting via Redis cluster

Processing Layer

- SQS for job queue management
- Lambda functions for async processing
- DynamoDB for job status tracking

Real-time Updates

- WebSocket API for progress updates
- Redis pub/sub for worker communication

Security Layer

- HMAC authentication
- AES-256 encryption at rest
- Customer-specific encryption keys

Alternatives Considered

Synchronous Processing

❌ Rejected due to: - Timeout limitations - Resource inefficiency - Poor user experience for large imports

Self-hosted Queue

❌ Rejected due to: - Operational overhead - Scaling complexity - Reliability concerns

Direct Database Import

❌ Rejected due to: - Lack of validation - Security concerns - No progress tracking

Consequences

Positive

- Scalable to handle varying load
- Built-in fault tolerance
- Real-time progress updates
- Clear separation of concerns
- Easy to monitor and debug

Negative

- More complex infrastructure
- Multiple components to maintain
- Higher AWS costs
- Need for careful error handling

Implementation Details

Request Flow

```
Client -> API Gateway
       -> HMAC Validation
       -> Rate Limit Check
       -> Create Job (DynamoDB)
       -> Queue Message (SQS)
       -> Return Job ID
```

Processing Flow

```
SQS -> Lambda Worker
    -> Process Batch
    -> Update Progress (DynamoDB)
    -> Notify WebSocket
    -> Handle Errors
```

Monitoring Points

1. API Gateway metrics
2. Lambda execution stats
3. Queue depth
4. Processing speed
5. Error rates
6. WebSocket connections

Security Considerations

Authentication

- HMAC signatures required
- API keys with expiration
- IP whitelisting option

Encryption

- TLS 1.3 in transit
- AES-256 at rest
- Key rotation policy

Audit

- CloudWatch logs
- CloudTrail for API calls
- Custom audit events

Performance Targets

Throughput

- 1000 req/min API rate
- 5 concurrent imports
- 100k records/10min

Latency

- API response: < 100ms
- WebSocket updates: < 1s
- Job status: < 500ms

Cost Considerations

- Lambda execution time
- API Gateway requests
- WebSocket connections
- DynamoDB throughput
- SQS message volume

Future Considerations

1. Multi-region deployment
2. Customer-specific queues
3. Enhanced monitoring
4. Automated scaling
5. Backup strategy

Team Impact

- New AWS services to learn
- WebSocket implementation
- Security best practices
- Performance optimization

Dependencies

1. AWS Account
2. Redis Enterprise
3. Security review
4. Performance testing

Success Metrics

1. Processing speed
2. Error rates
3. Customer satisfaction
4. System stability
5. Operational overhead

Rollout Plan

1. Infrastructure setup
2. Security review
3. Load testing
4. Gradual customer onboarding