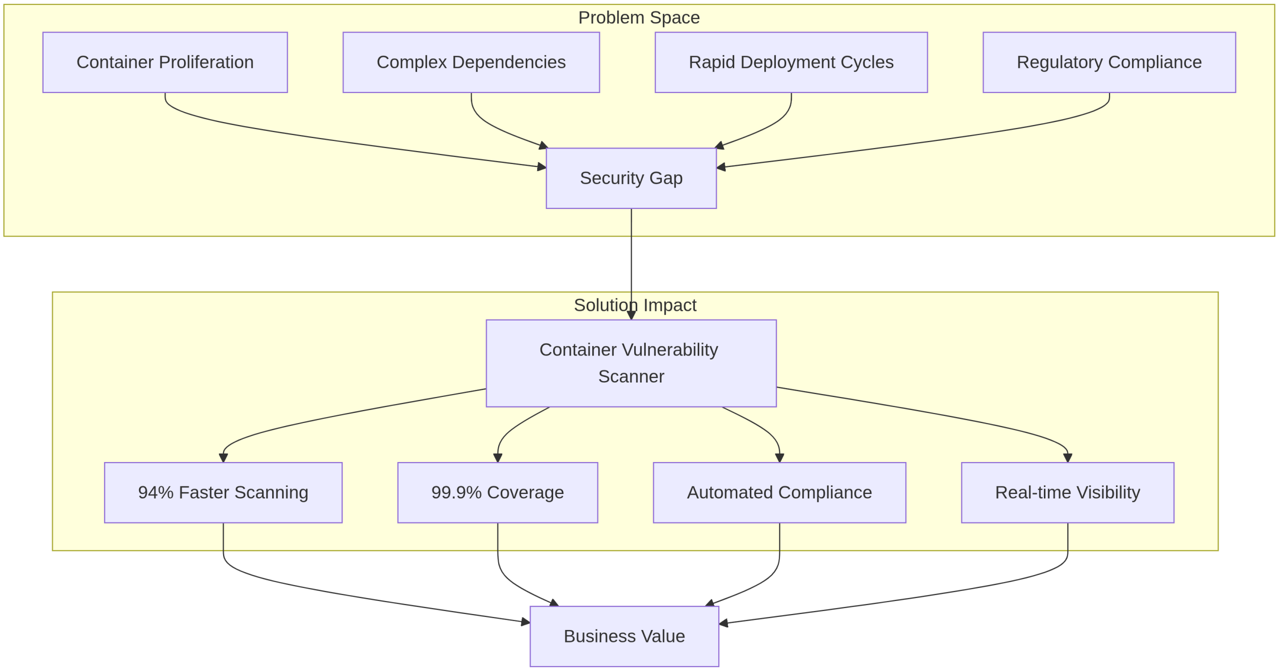


# Container Vulnerability Scanner

Enterprise System Design & Architecture Deep-Dive

Executive Overview

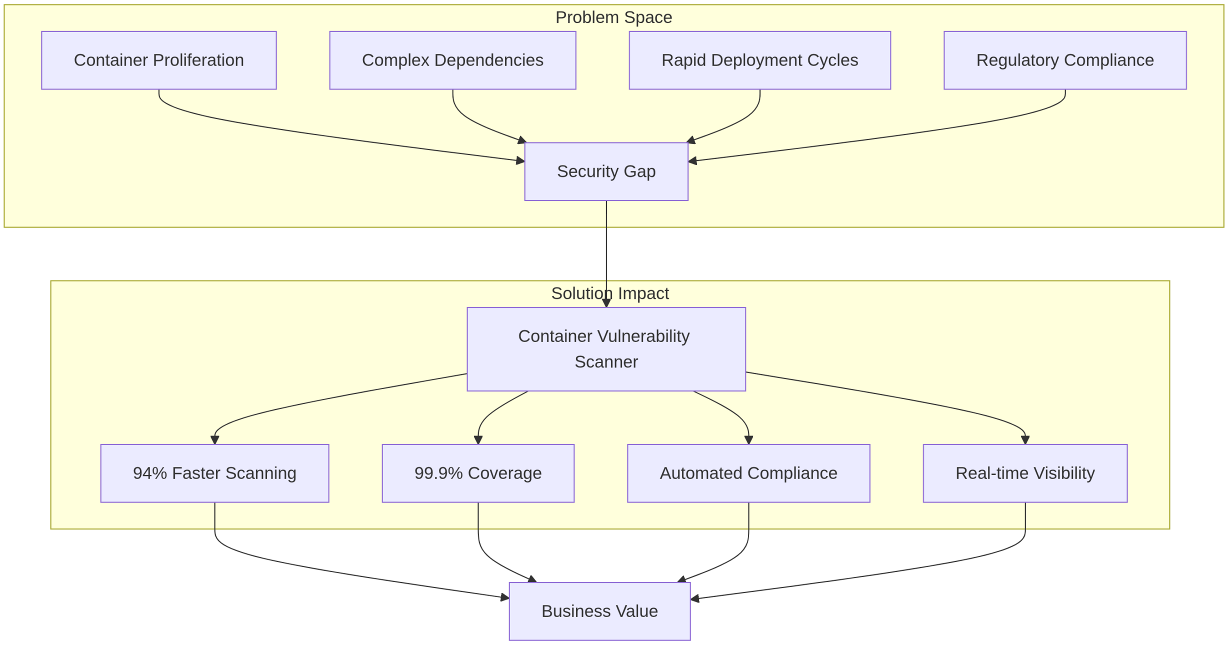
1.1 The Imperative for Container Security

Problem Landscape: Modern containerized applications face critical security challenges that manual processes cannot address at scale.

Key Challenges:

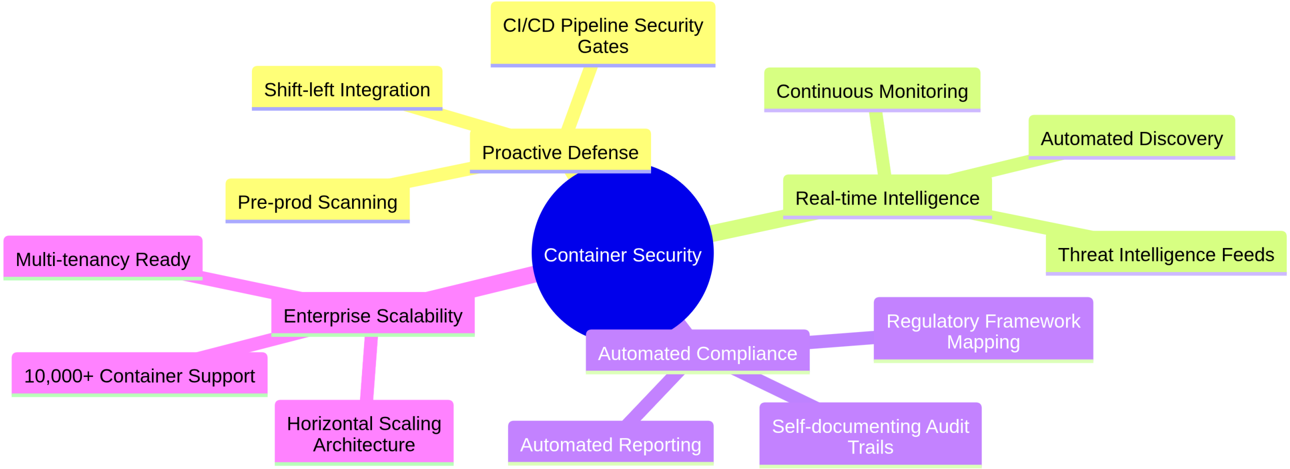
- Average container has 100-500 software packages with independent vulnerabilities
- 72% of critical CVEs weaponized within 30 days of disclosure
- Manual reviews consume 15-25% of engineering capacity

- Compliance frameworks (SOC2, HIPAA, GDPR) require continuous assessment



1.2 Solution Architecture Philosophy

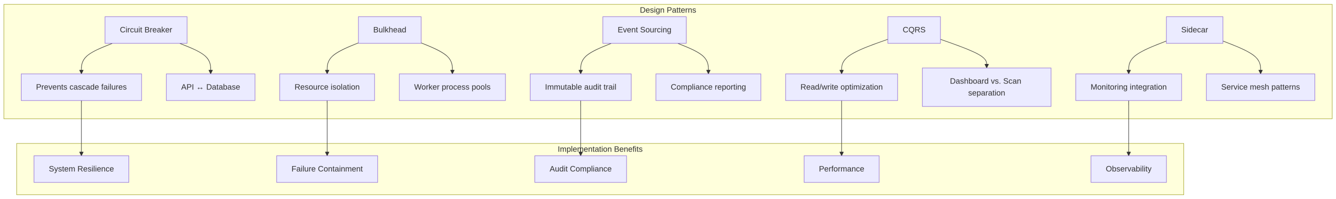
Four foundational principles guide the system design:



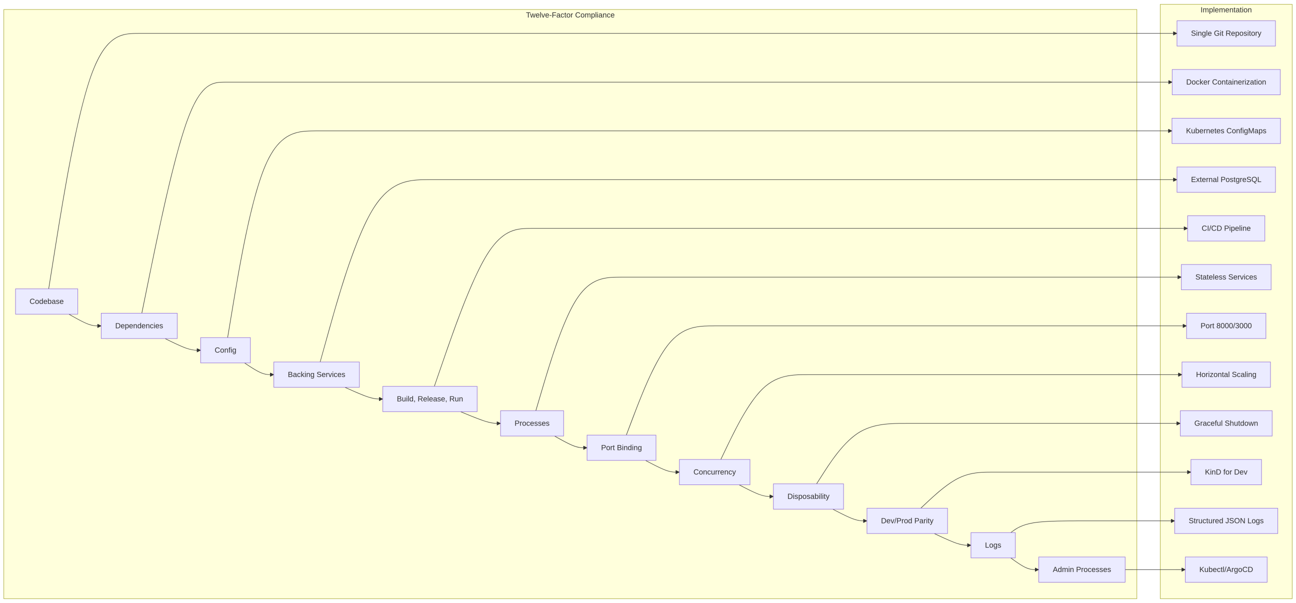
Architectural Foundations

2.1 System Design Principles

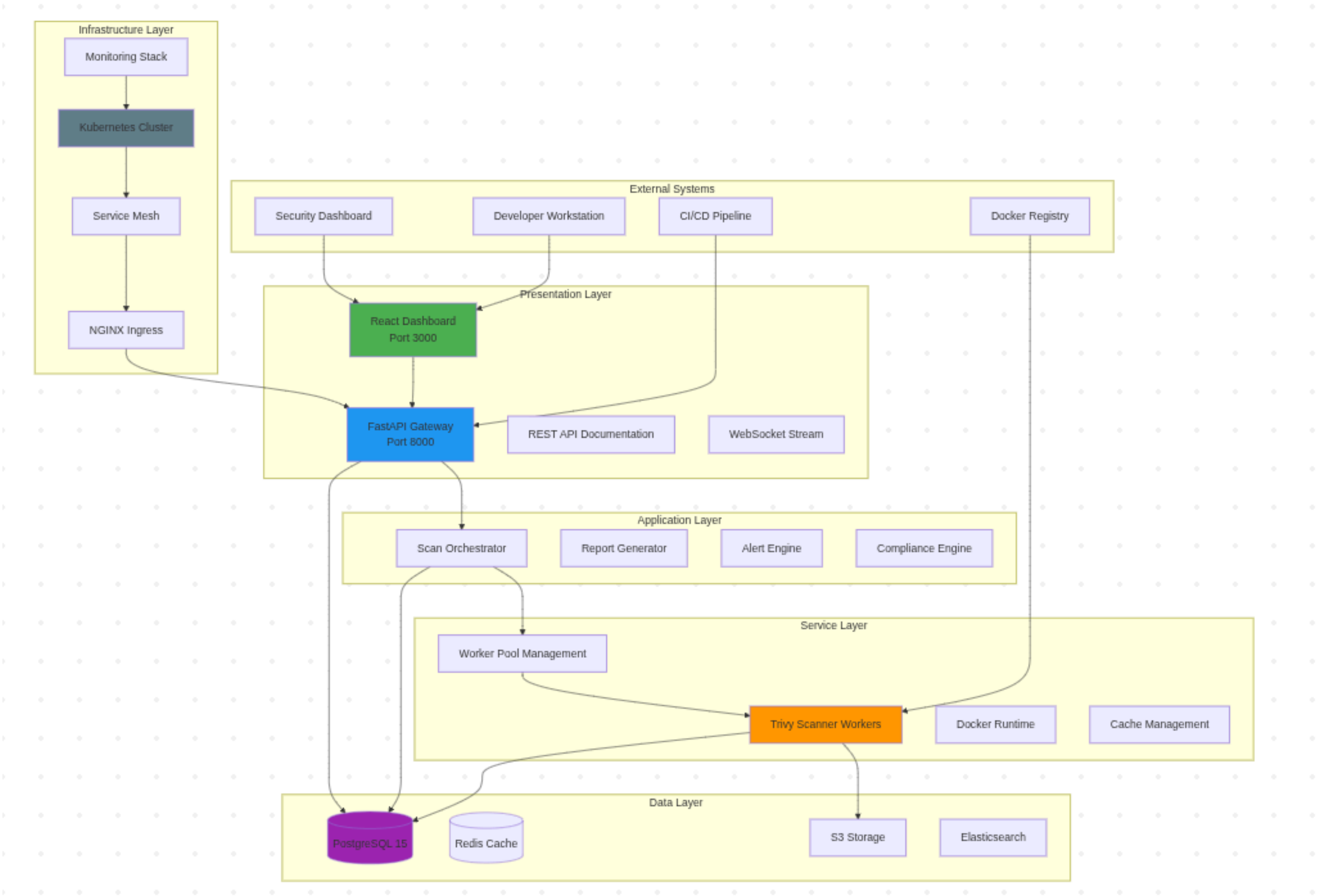
2.1.1 Cloud-Native Architecture Patterns



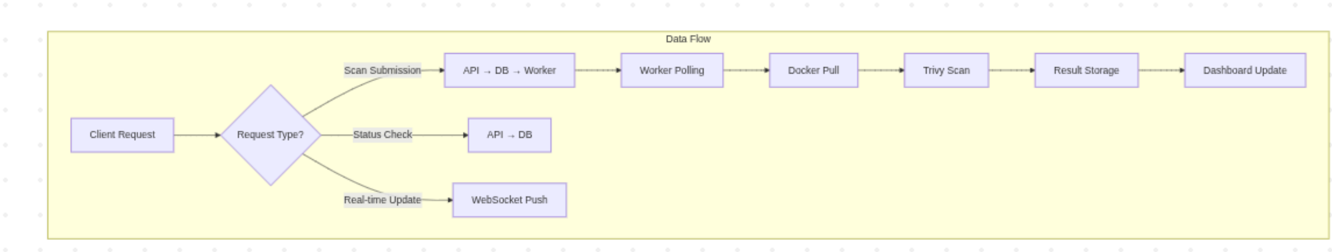
2.1.2 Twelve-Factor Application Compliance

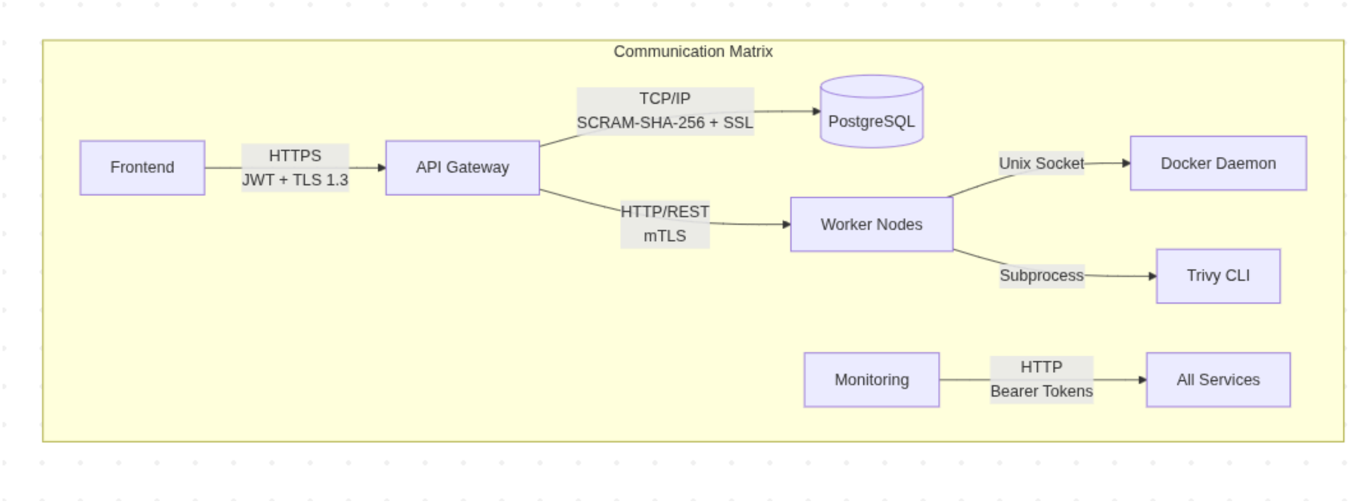


2.2 High-Level System Architecture



2.3 Component Interaction Matrix

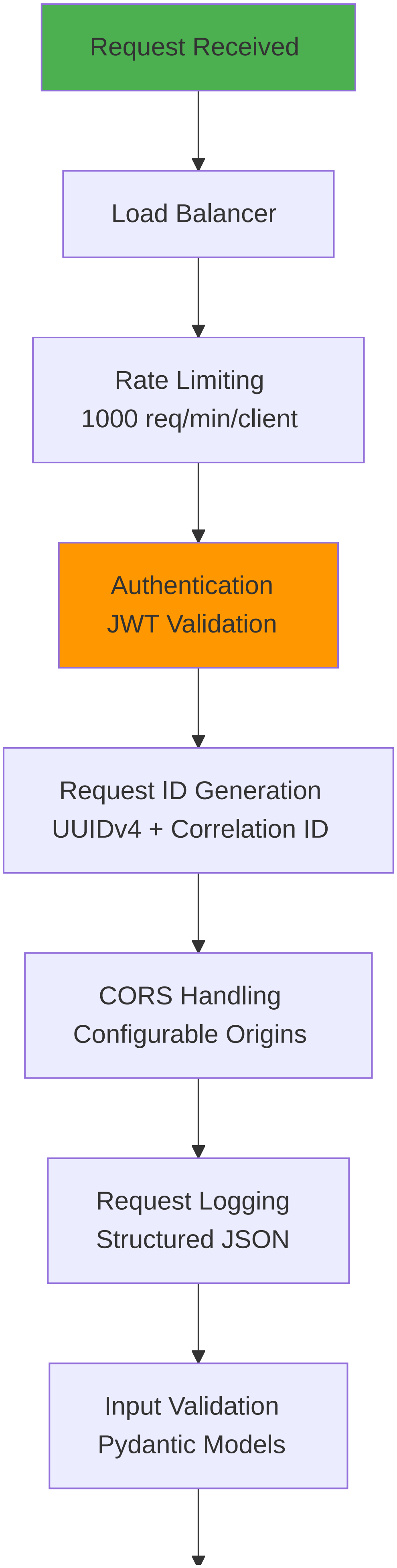


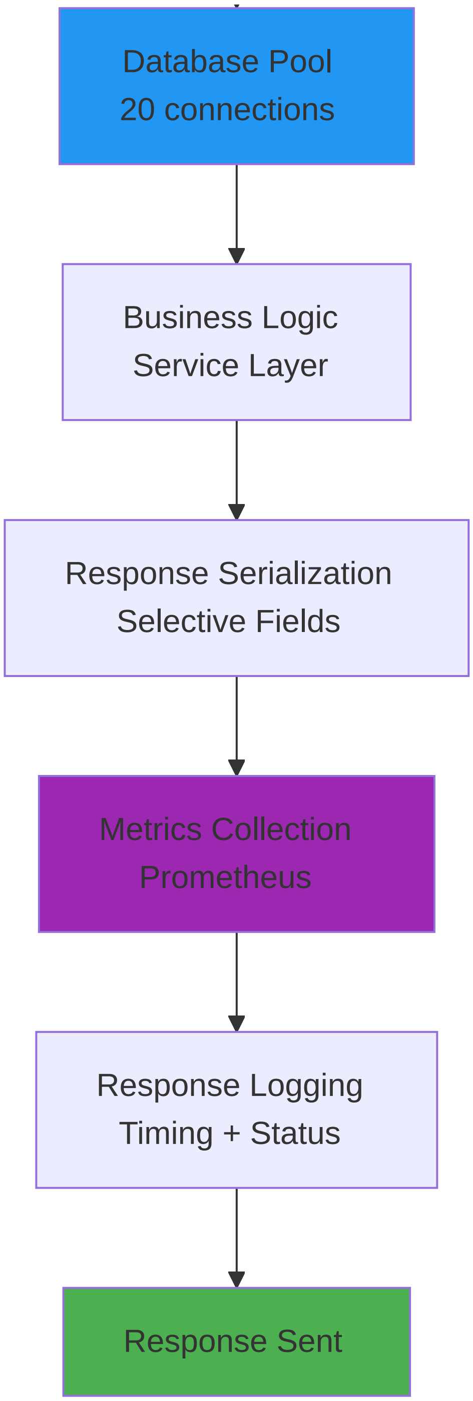


Core System Components

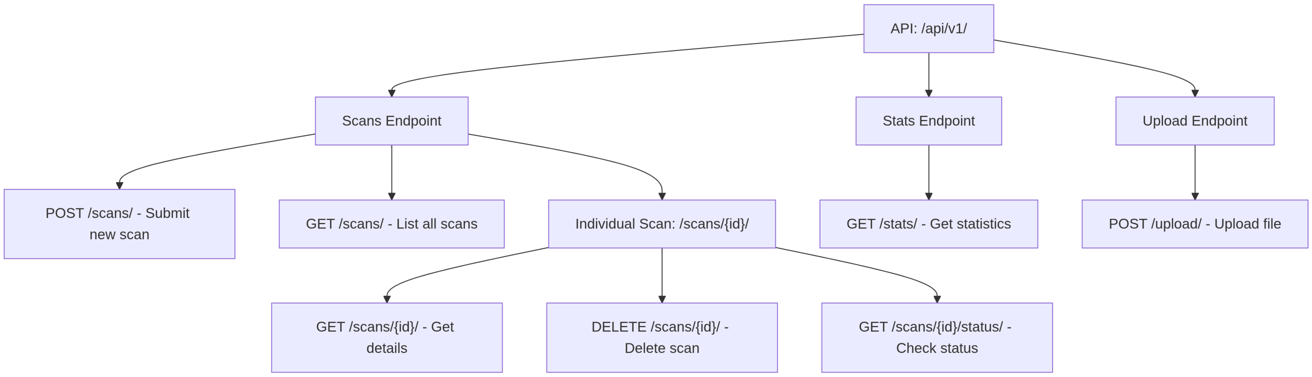
3.1 API Gateway Architecture

3.1.1 Request Processing Pipeline


Request Received
Load Balancer
Rate Limiting
1000 req/min/client
Authentication
JWT Validation
Request ID Generation
UUIDv4 + Correlation ID
CORS Handling
Configurable Origins
Request Logging
Structured JSON
Input Validation
Pydantic Models



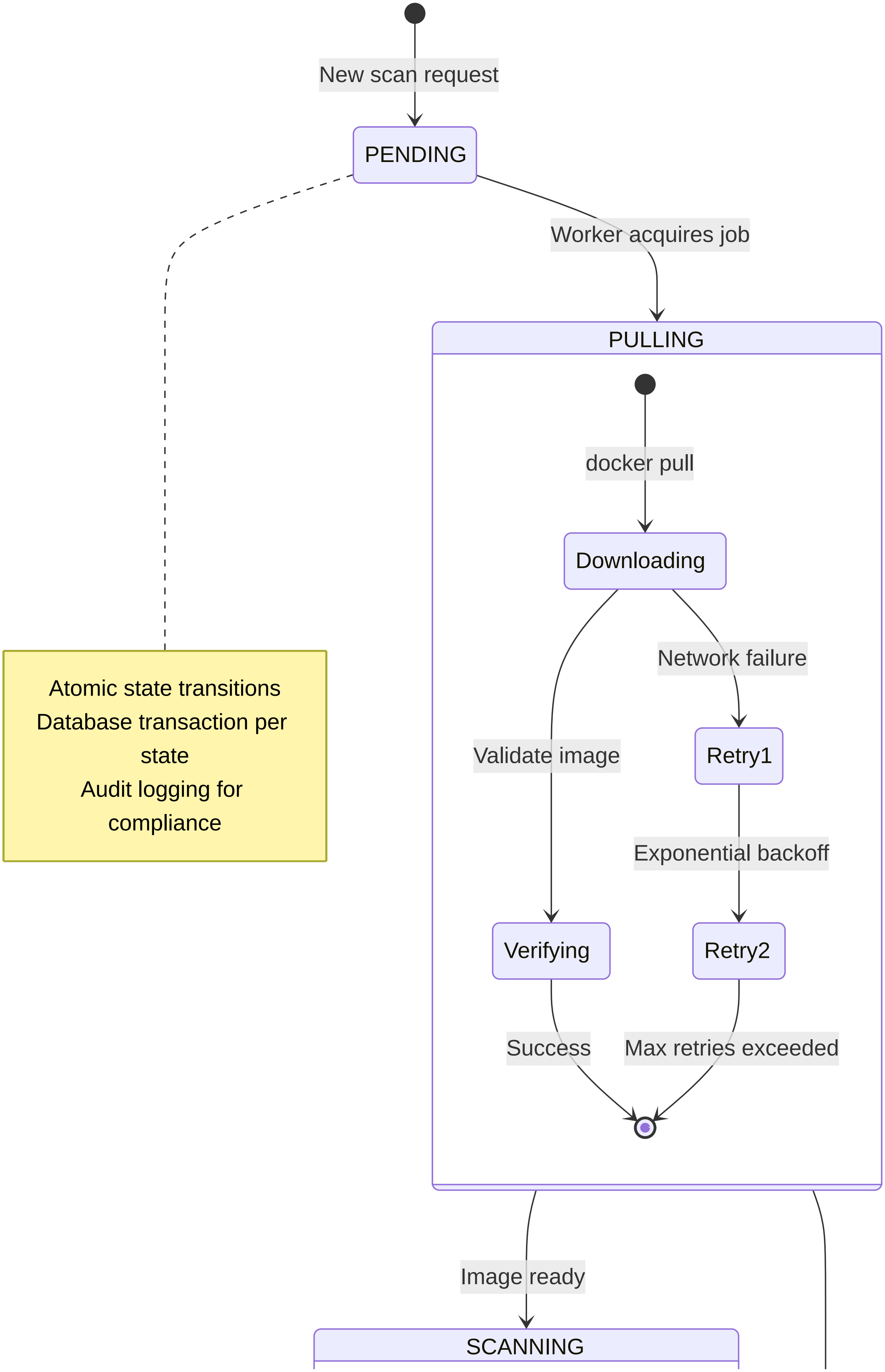
3.1.2 RESTful Endpoint Design



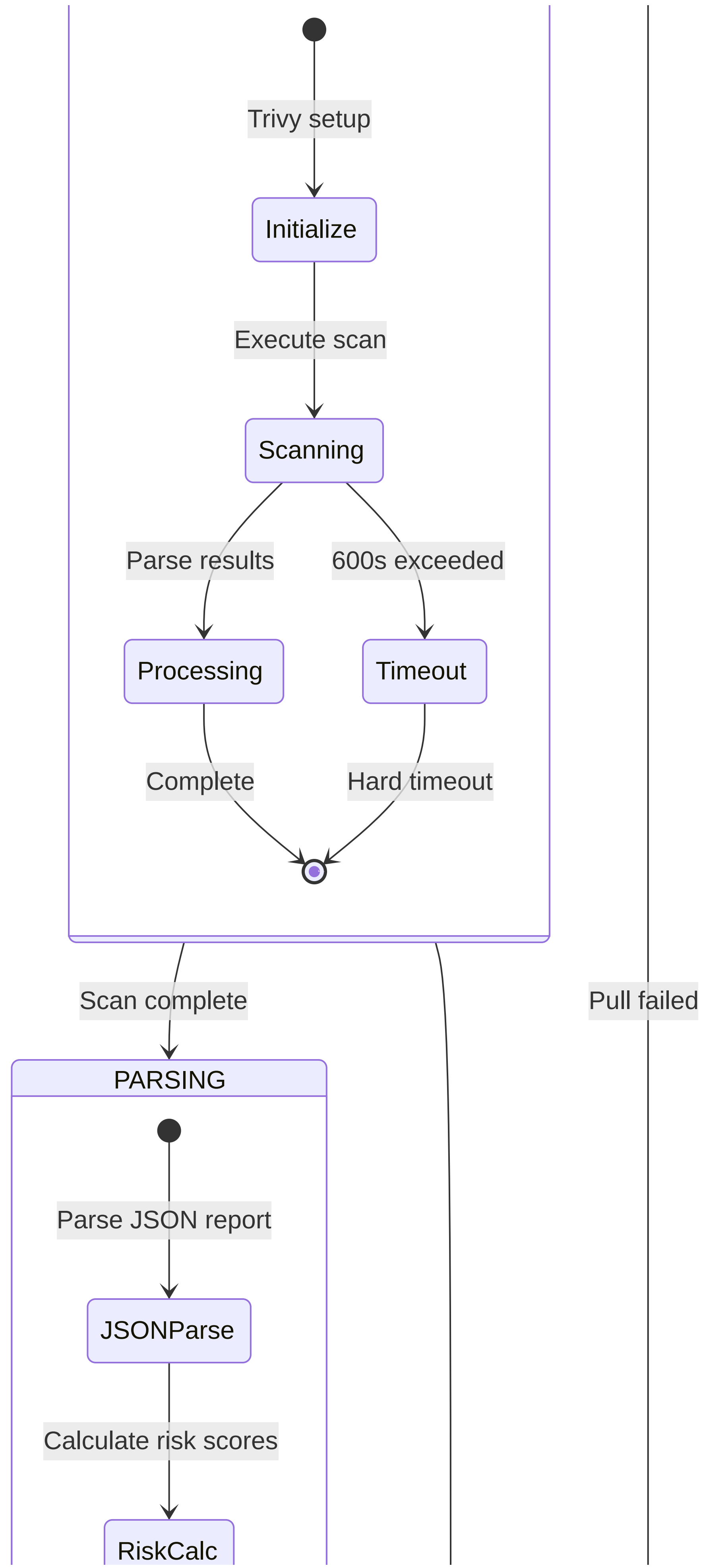
API: /api/v1/
Scans Endpoint
Stats Endpoint
Upload Endpoint
POST /scans/ - Submit new scan
GET /scans/ - List all scans
Individual Scan: /scans/{id}/
GET /stats/ - Get statistics
POST /upload/ - Upload file
GET /scans/{id}/ - Get details
DELETE /scans/{id}/ - Delete scan
GET /scans/{id}/status/ - Check status

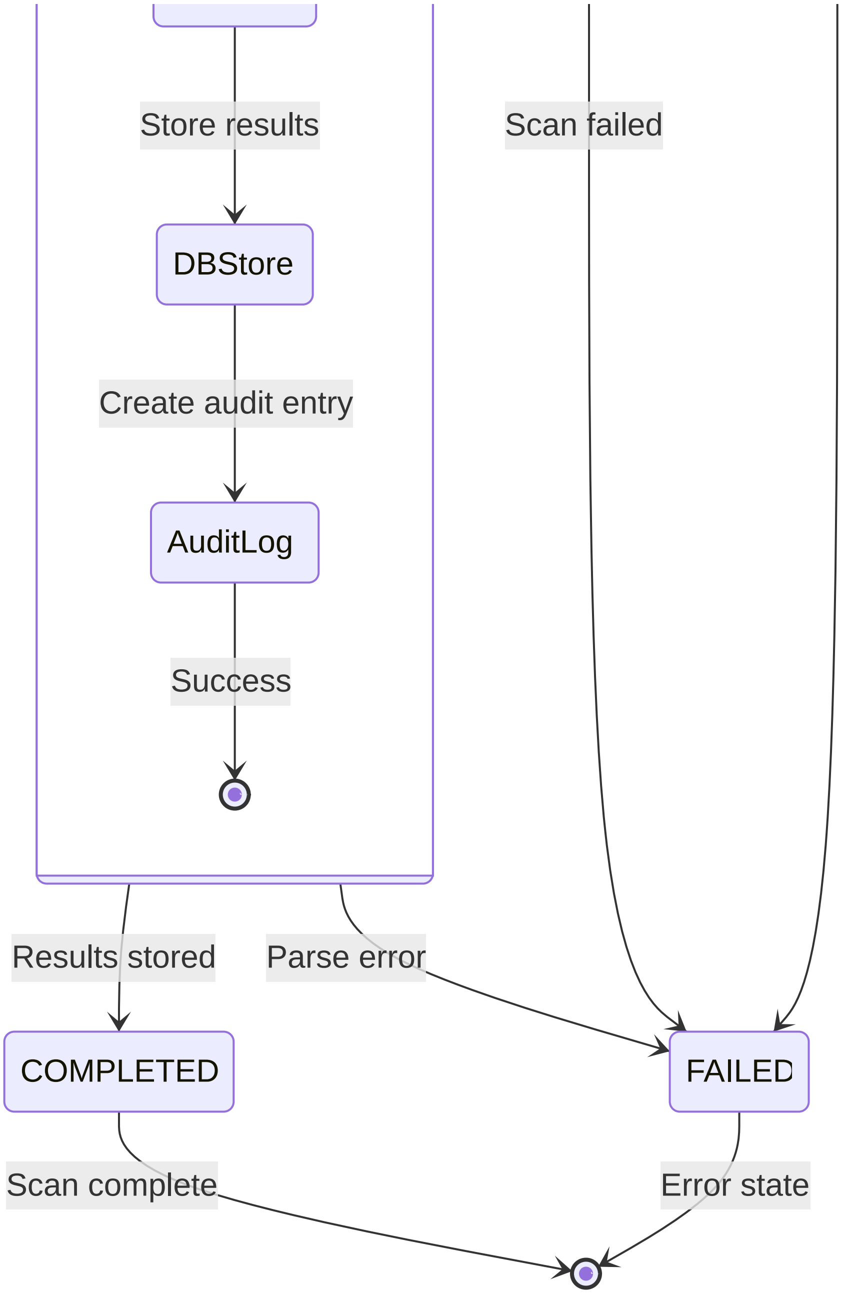
3.2 Worker Service Architecture

3.2.1 Finite State Machine Design

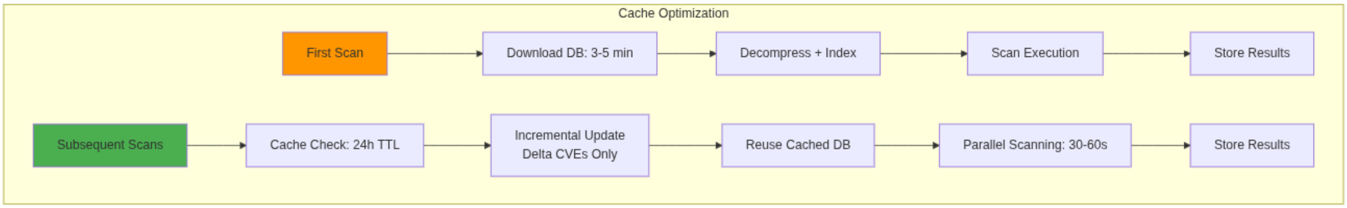


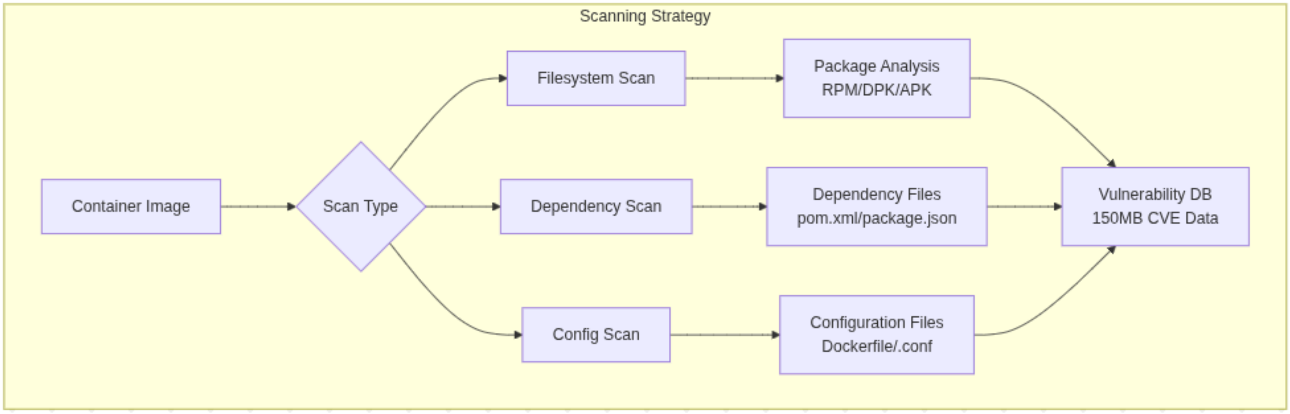
Atomic state transitions
Database transaction per state
Audit logging for compliance


Trivy setup
Initialize
Execute scan
Scanning
Parse results
600s exceeded
Processing
Timeout
Complete
Hard timeout
Scan complete
Pull failed
PARSING
Parse JSON report
JSONParse
Calculate risk scores
RiskCalc



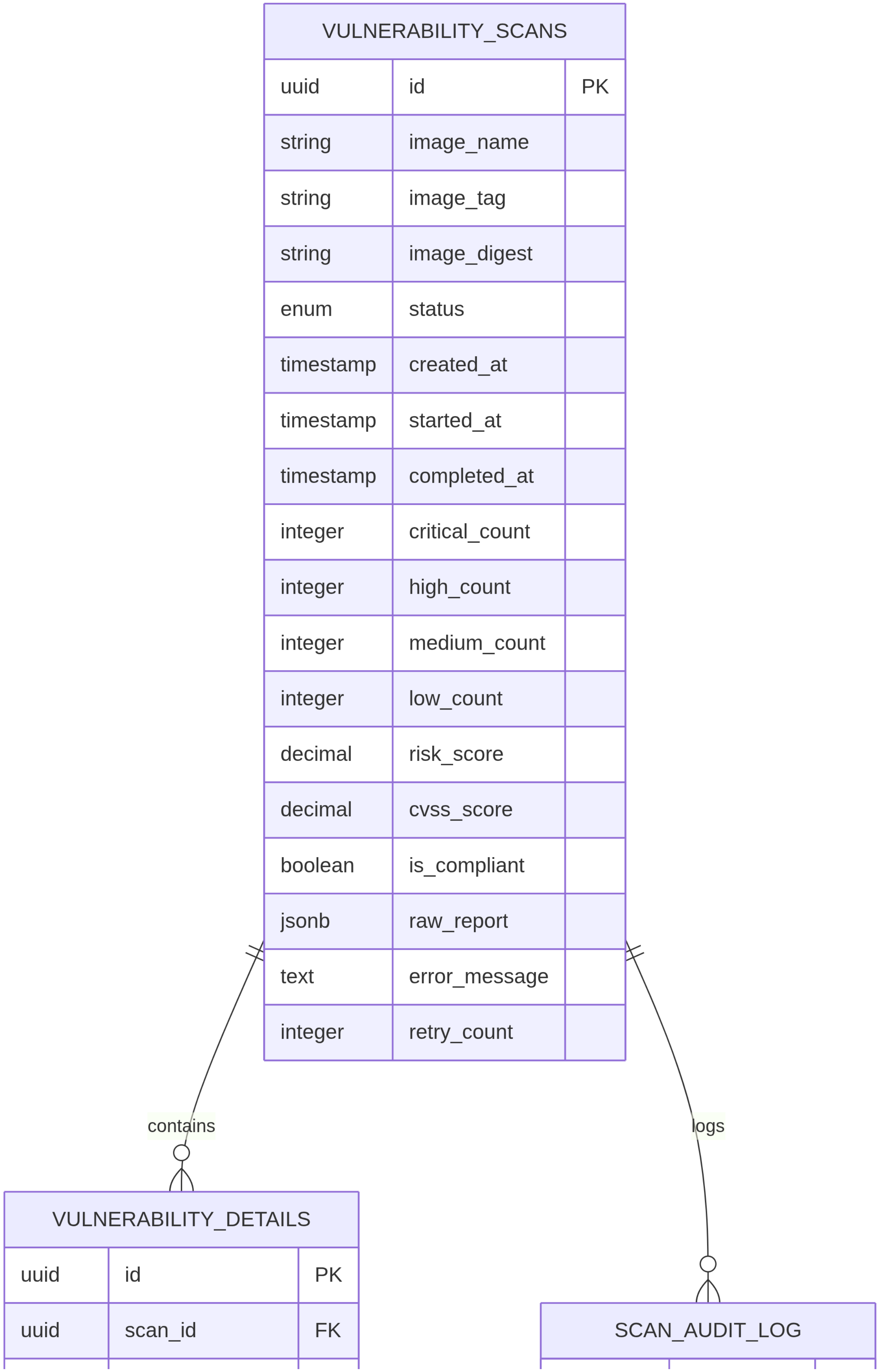
3.2.2 Trivy Integration Architecture





3.3 Database Architecture

3.3.1 Schema Design Strategy



VULNERABILITY_SCANS		
uuid	id	PK
string	image_name	
string	image_tag	
string	image_digest	
enum	status	
timestamp	created_at	
timestamp	started_at	
timestamp	completed_at	
integer	critical_count	
integer	high_count	
integer	medium_count	
integer	low_count	
decimal	risk_score	
decimal	cvss_score	
boolean	is_compliant	
jsonb	raw_report	
text	error_message	
integer	retry_count	

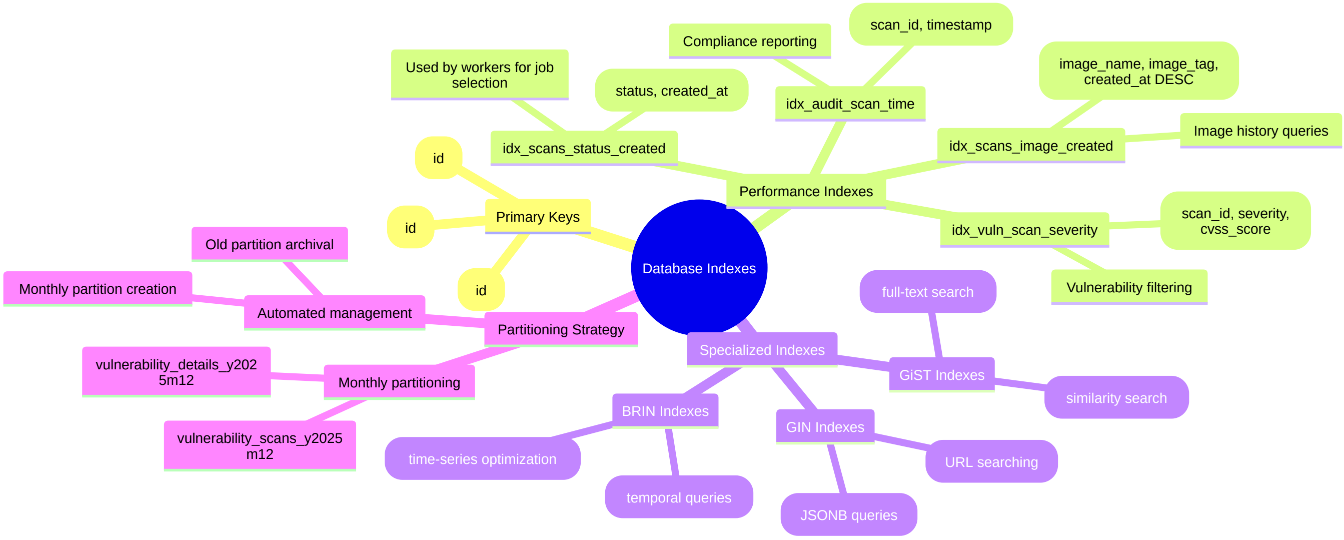
VULNERABILITY_DETAILS		
uuid	id	PK
uuid	scan_id	FK

SCAN_AUDIT_LOG

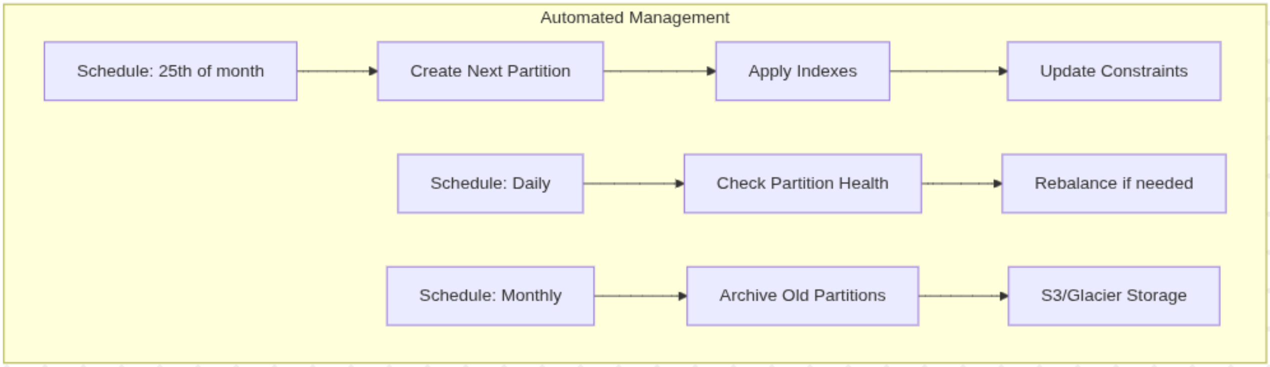
string	cve_id	
enum	severity	
text	description	
decimal	cvss_score	
string	cvss_vector	
string	package_name	
string	package_version	
string	fixed_version	
jsonb	references	

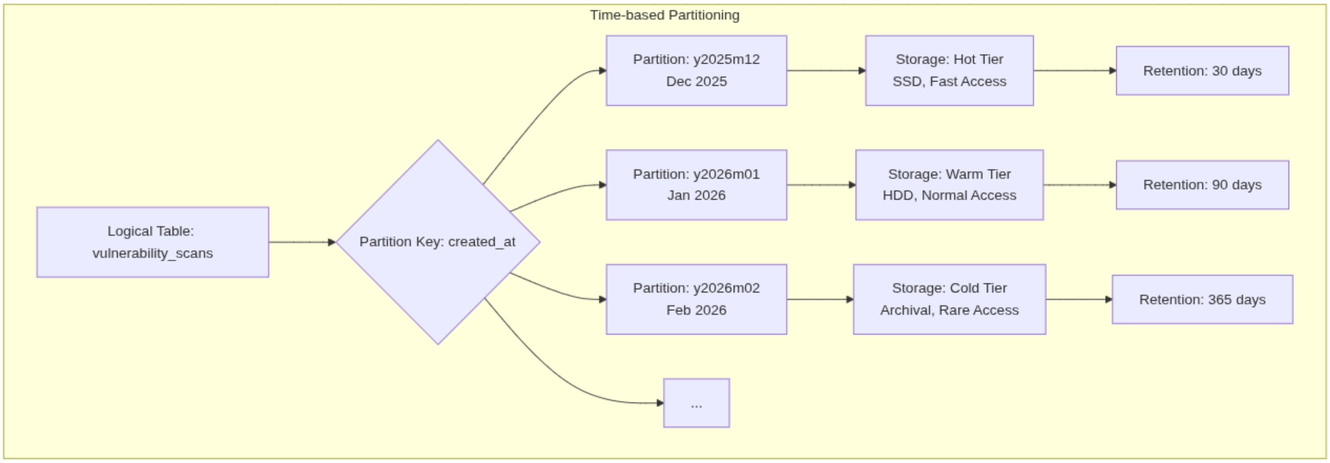
uuid	id	PK
uuid	scan_id	FK
string	old_status	
string	new_status	
string	changed_by	
text	reason	
timestamp	timestamp	

3.3.2 Indexing Strategy



3.3.3 Partitioning Architecture

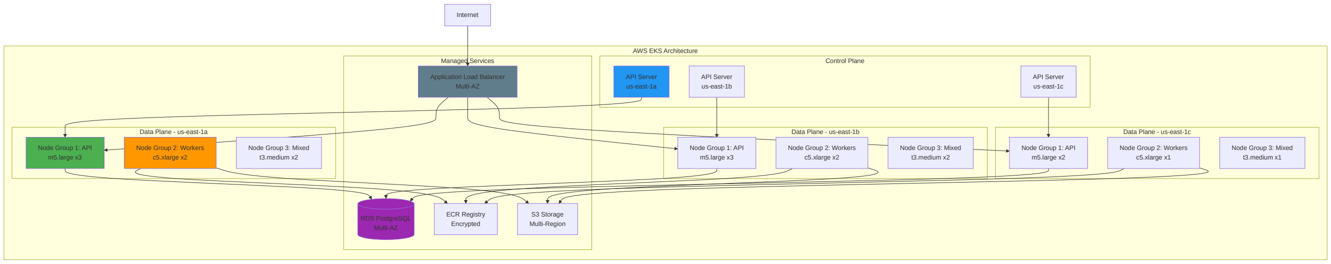



Time-based Partitioning
Logical Table: vulnerability_scans
Partition Key: created_at
Partition: y2025m12 Dec 2025
Partition: y2026m01 Jan 2026
Partition: y2026m02 Feb 2026
...
Storage: Hot Tier SSD, Fast Access
Storage: Warm Tier HDD, Normal Access
Storage: Cold Tier Archival, Rare Access
Retention: 30 days
Retention: 90 days
Retention: 365 days

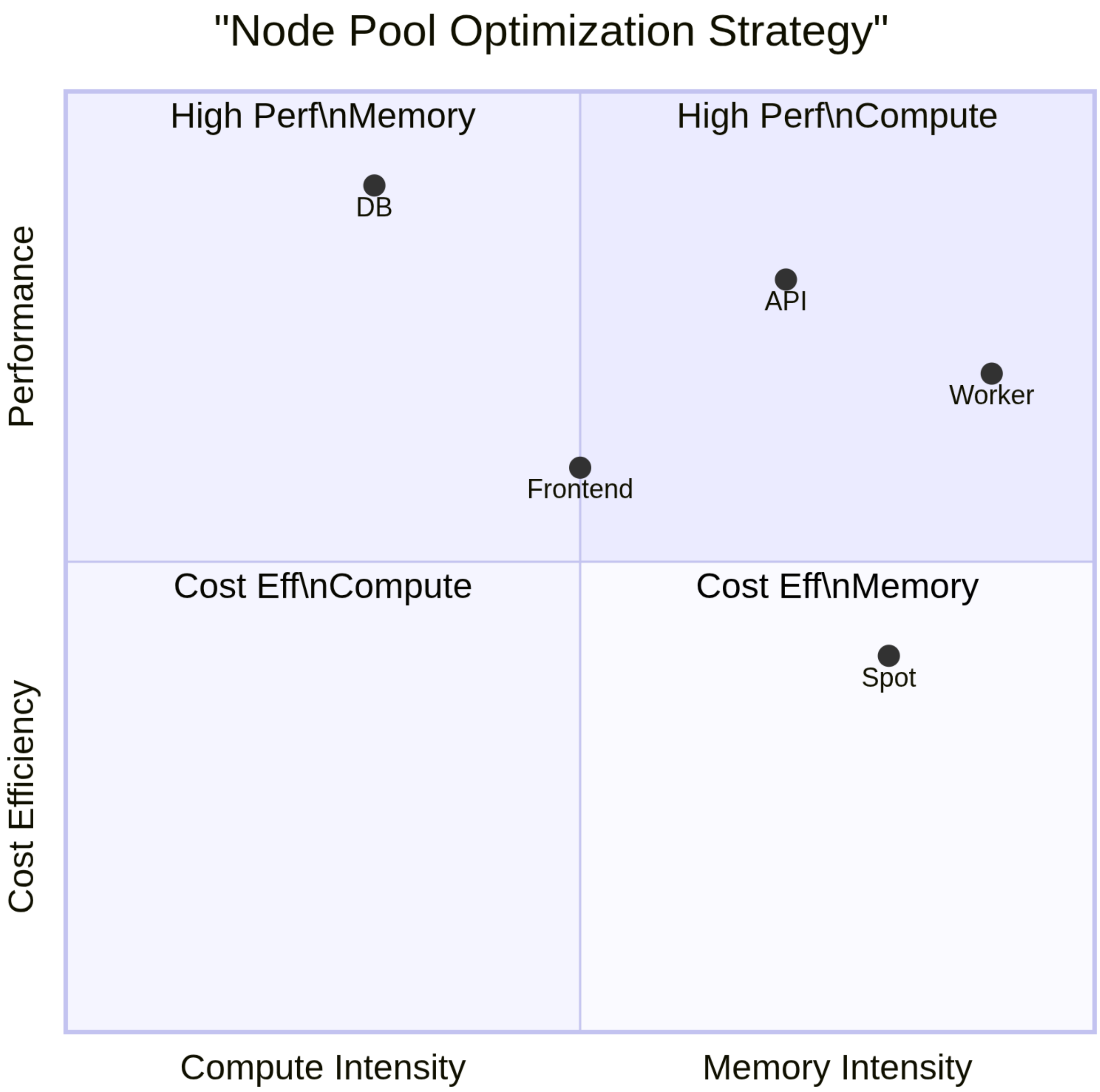
Kubernetes Deployment Architecture

4.1 Cluster Design Philosophy

4.1.1 Multi-Zone High Availability

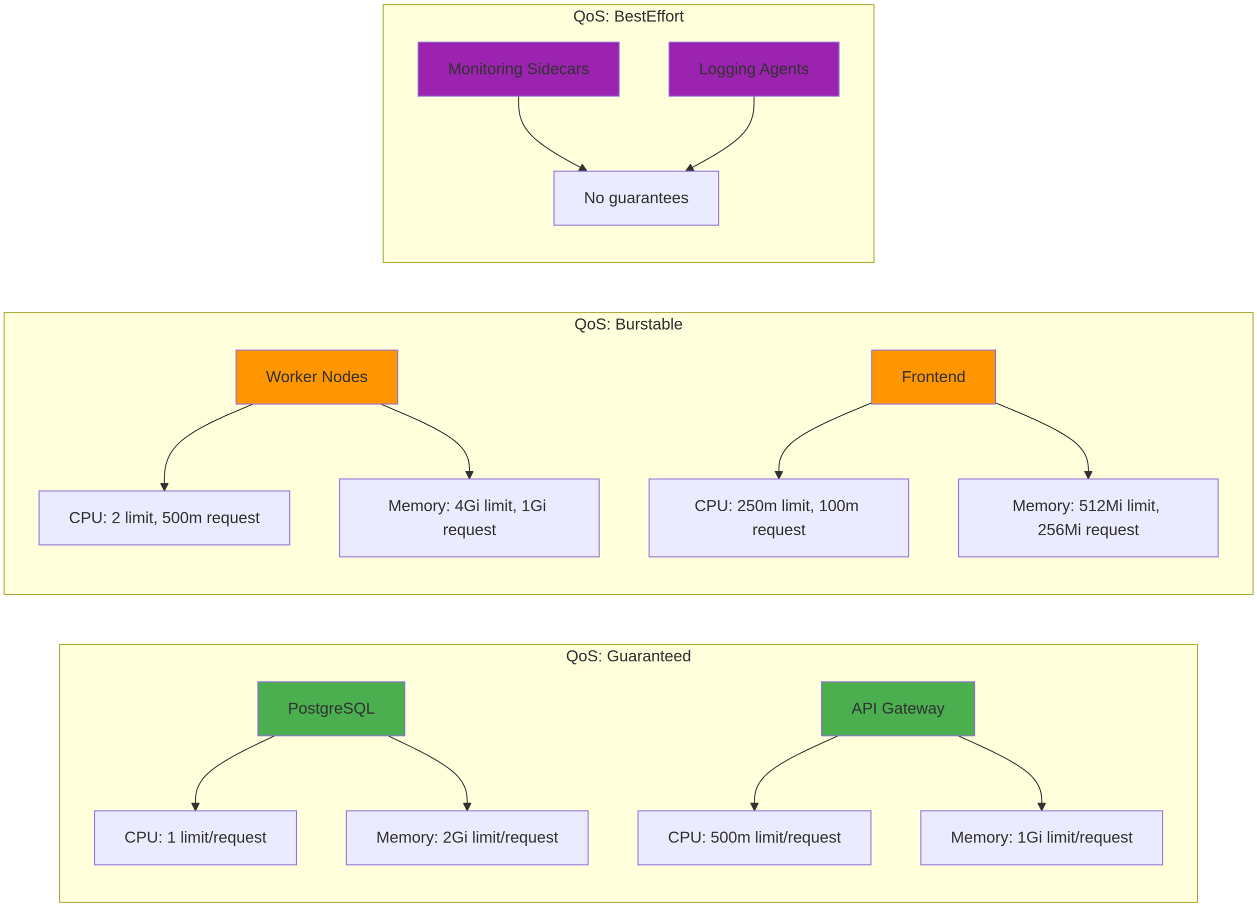

Internet
AWS EKS Architecture
Managed Services
Application Load Balancer Multi-AZ
Control Plane
API Server us-east-1a
API Server us-east-1b
API Server us-east-1c
Data Plane - us-east-1a
Node Group 1: API m5.large x3
Node Group 2: Workers c5.xlarge x2
Node Group 3: Mixed t3.medium x2
Data Plane - us-east-1b
Node Group 1: API m5.large x3
Node Group 2: Workers c5.xlarge x2
Node Group 3: Mixed t3.medium x2
Data Plane - us-east-1c
Node Group 1: API m5.large x2
Node Group 2: Workers c5.xlarge x1
Node Group 3: Mixed t3.medium x1
RDS PostgreSQL Multi-AZ
ECR Registry Encrypted
S3 Storage Multi-Region

4.1.2 Node Pool Strategy

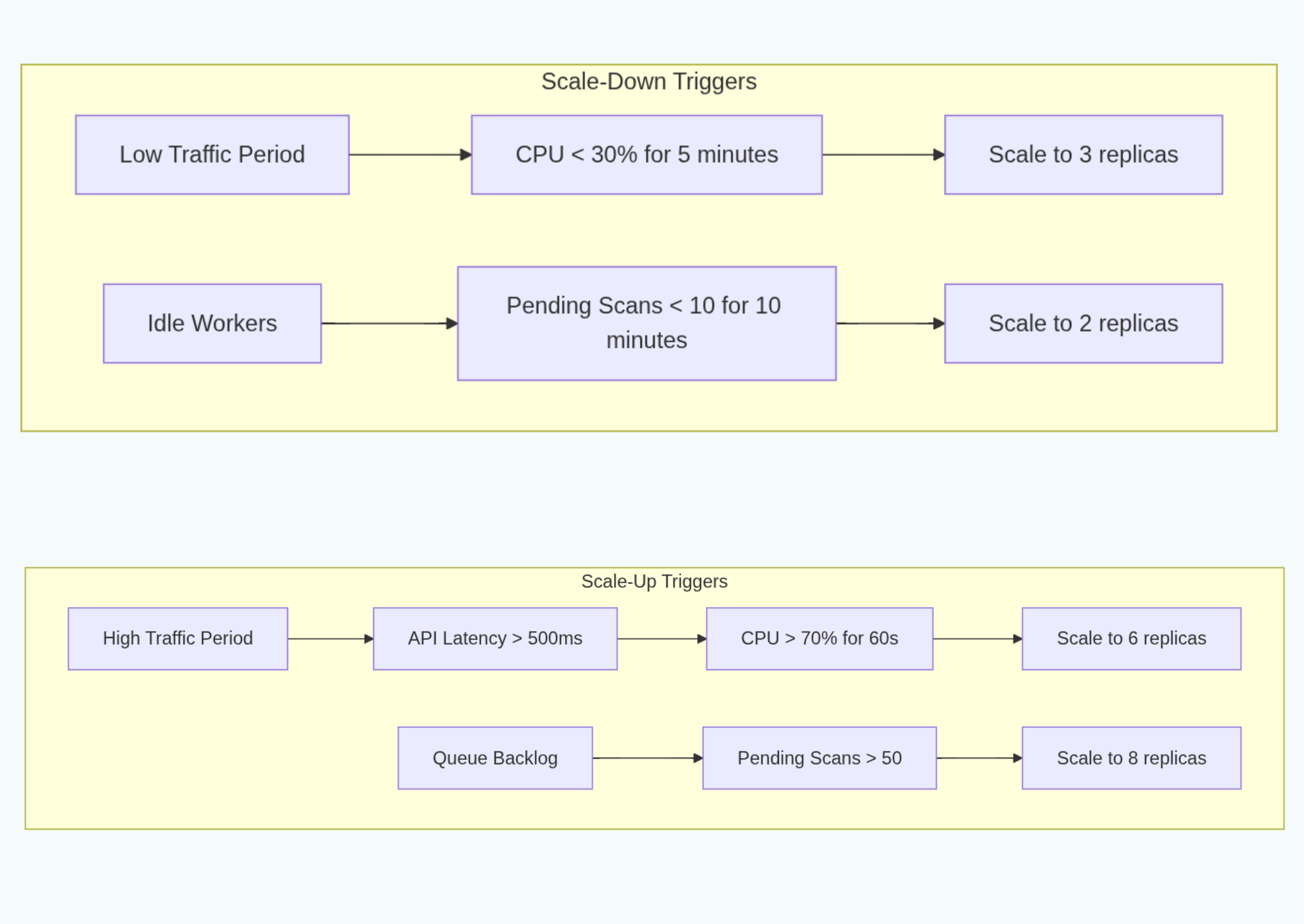


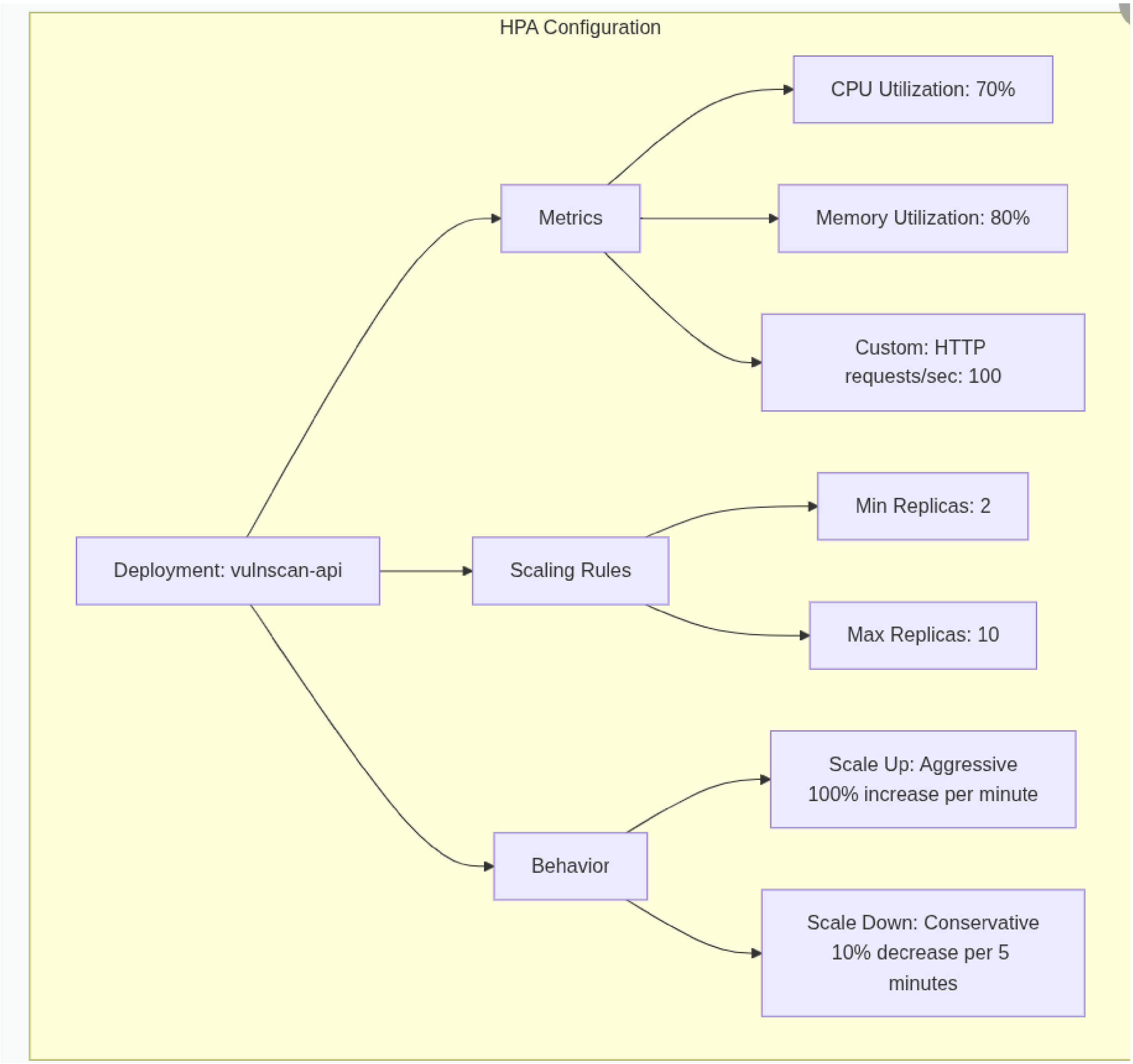
4.2 Resource Management Strategy

4.2.1 Quality of Service Tiers



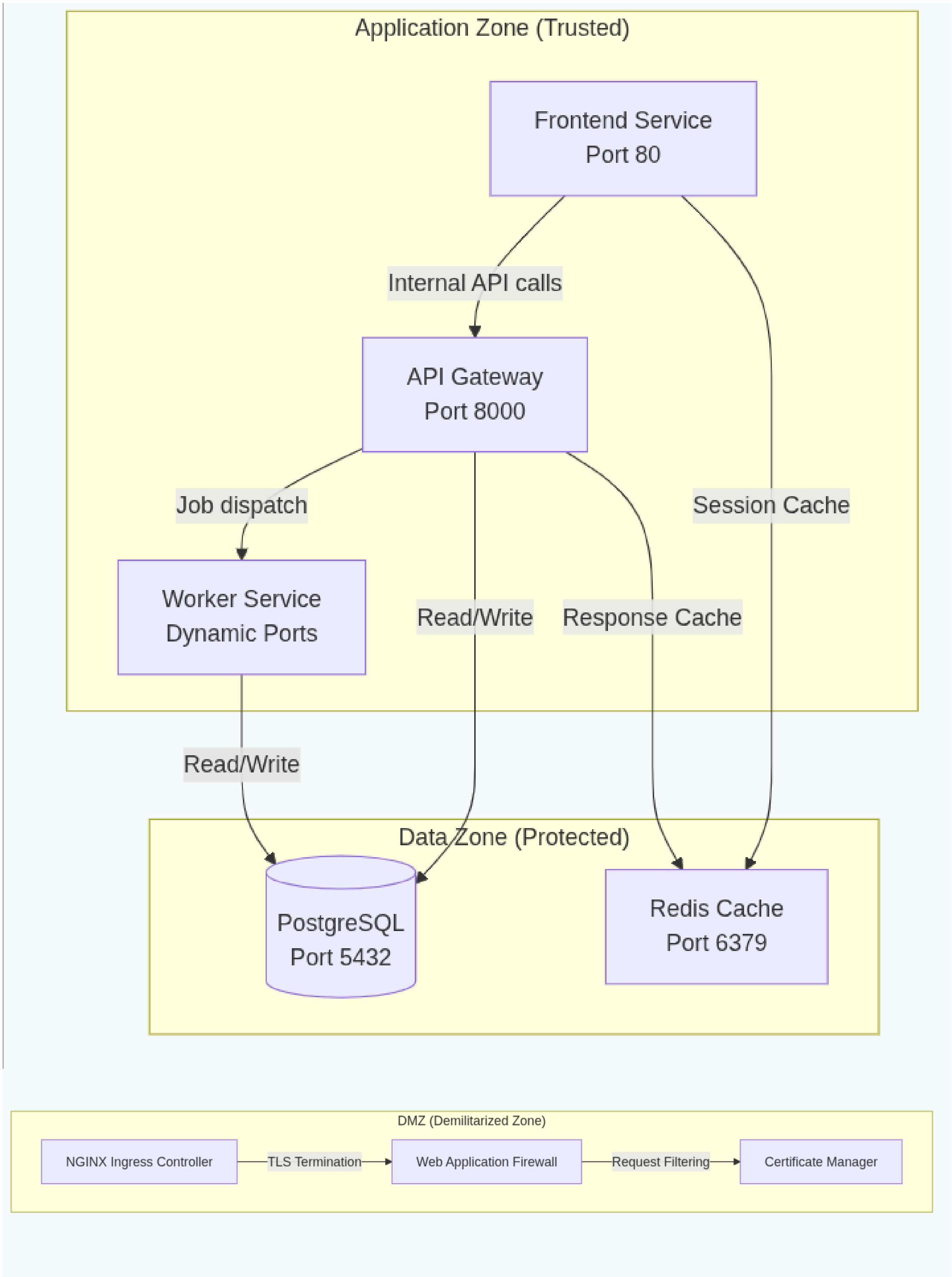
4.2.2 Horizontal Pod Autoscaling

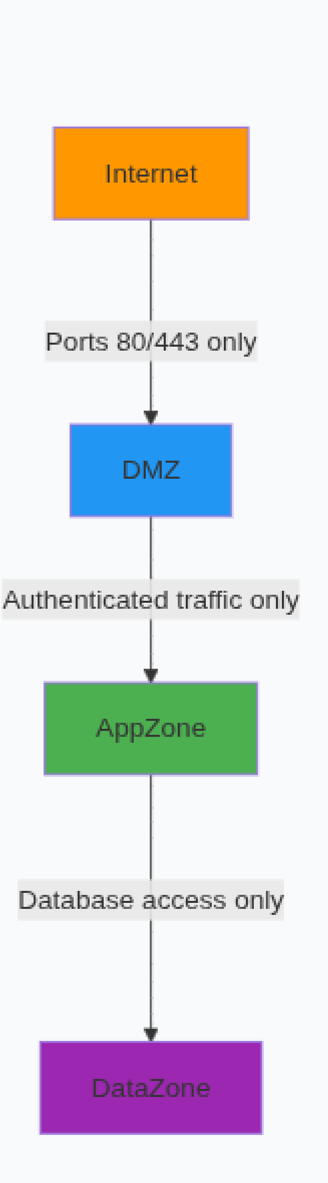

Scale-Down Triggers
Low Traffic Period
CPU < 30% for 5 minutes
Scale to 3 replicas
Idle Workers
Pending Scans < 10 for 10 minutes
Scale to 2 replicas
Scale-Up Triggers
High Traffic Period
API Latency > 500ms
CPU > 70% for 60s
Scale to 6 replicas
Queue Backlog
Pending Scans > 50
Scale to 8 replicas



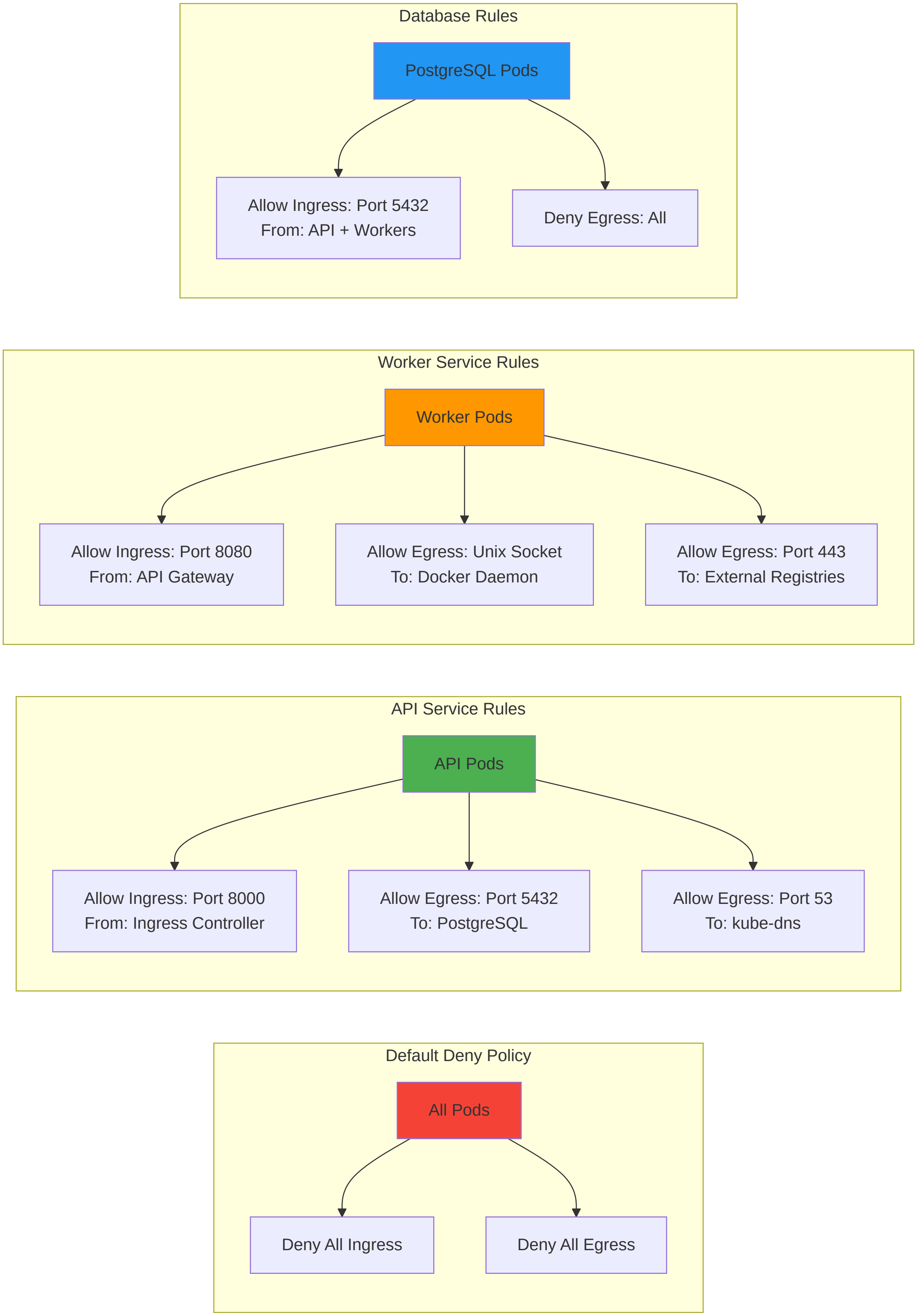
4.3 Network Security Architecture

4.3.1 Zero-Trust Network Model


Application Zone (Trusted)
Frontend Service
Port 80
Internal API calls
API Gateway
Port 8000
Job dispatch
Session Cache
Worker Service
Dynamic Ports
Read/Write
Response Cache
Read/Write
Data Zone (Protected)
PostgreSQL
Port 5432
Redis Cache
Port 6379
DMZ (Demilitarized Zone)
NGINX Ingress Controller
TLS Termination
Web Application Firewall
Request Filtering
Certificate Manager

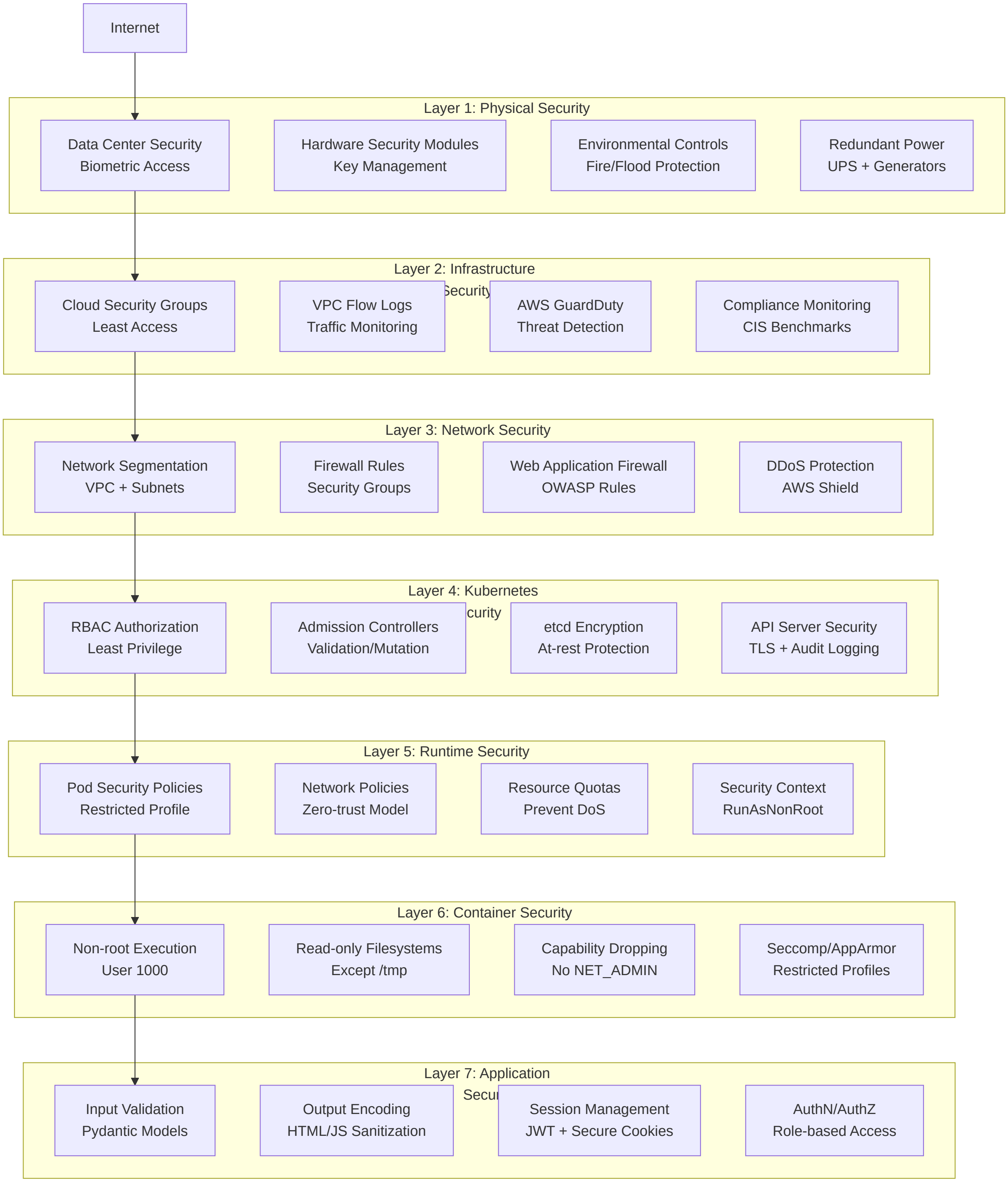


4.3.2 Network Policy Implementation

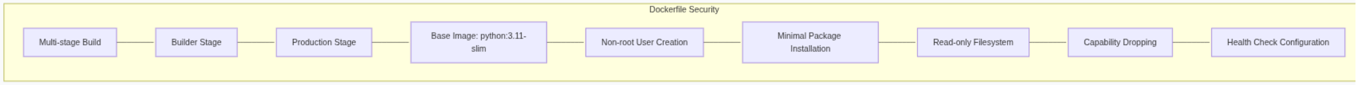

Database Rules
PostgreSQL Pods
Allow Ingress: Port 5432
From: API + Workers
Deny Egress: All
Worker Service Rules
Worker Pods
Allow Ingress: Port 8080
From: API Gateway
Allow Egress: Unix Socket
To: Docker Daemon
Allow Egress: Port 443
To: External Registries
API Service Rules
API Pods
Allow Ingress: Port 8000
From: Ingress Controller
Allow Egress: Port 5432
To: PostgreSQL
Allow Egress: Port 53
To: kube-dns
Default Deny Policy
All Pods
Deny All Ingress
Deny All Egress

Security Architecture

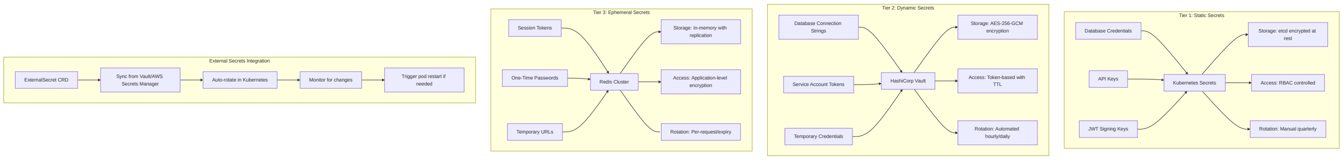
5.1 Defense-in-Depth Strategy



5.2 Container Security Hardening

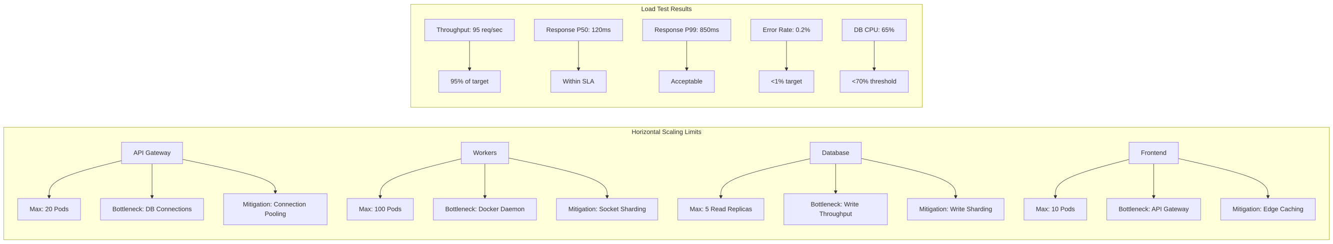


5.3 Secrets Management Architecture

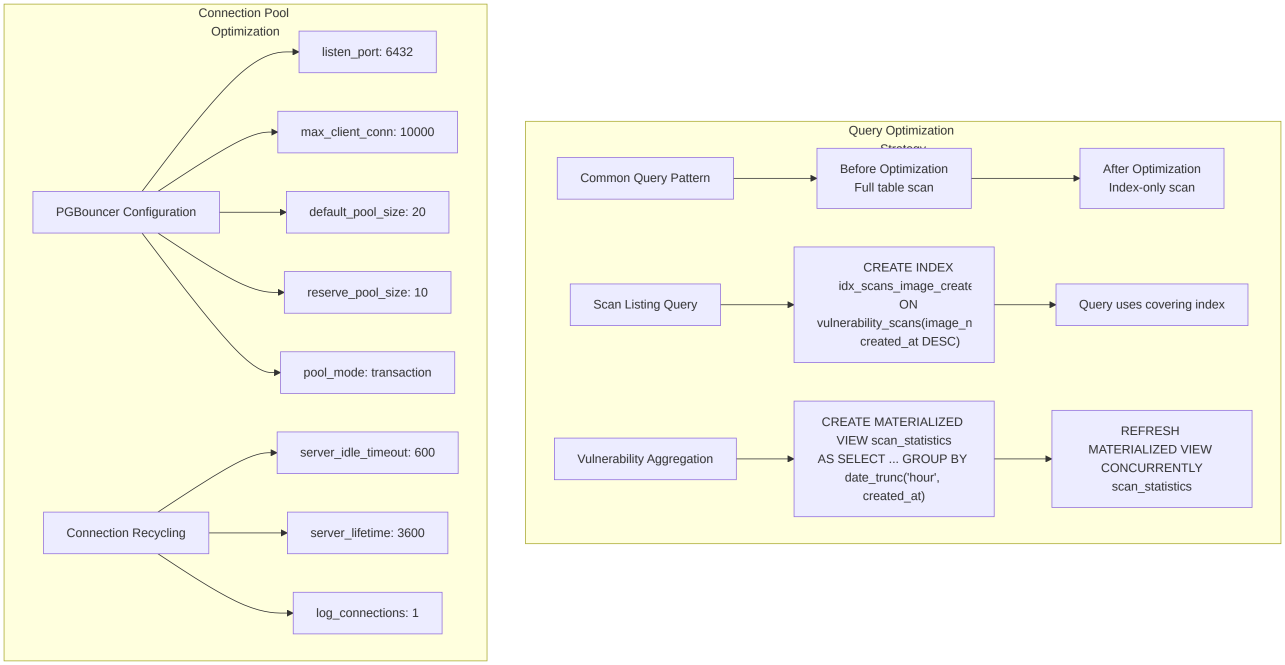

External Secrets Integration
ExternalSecret CRD
Sync from Vault/AWS Secrets Manager
Auto-rotate in Kubernetes
Monitor for changes
Trigger pod restart if needed
Tier 3: Ephemeral Secrets
Session Tokens
One-Time Passwords
Temporary URLs
Redis Cluster
Storage: In-memory with replication
Access: Application-level encryption
Rotation: Per-request/expiry
Tier 2: Dynamic Secrets
Database Connection Strings
Service Account Tokens
Temporary Credentials
HashiCorp Vault
Storage: AES-256-GCM encryption
Access: Token-based with TTL
Rotation: Automated hourly/daily
Tier 1: Static Secrets
Database Credentials
API Keys
JWT Signing Keys
Kubernetes Secrets
Storage: etcd encrypted at rest
Access: RBAC controlled
Rotation: Manual quarterly

Performance & Scalability

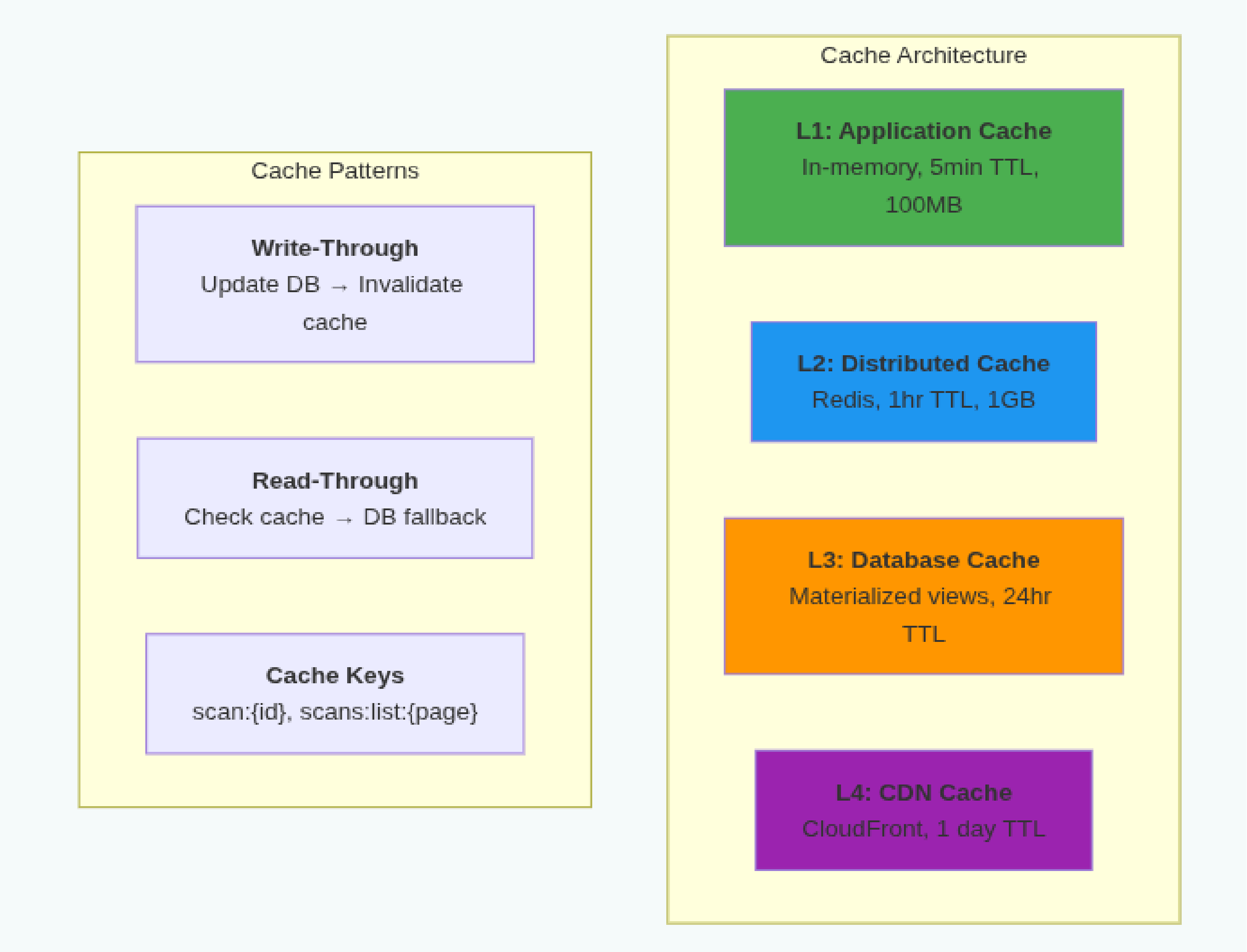
6.1 Scalability Analysis



6.2 Database Performance Optimization

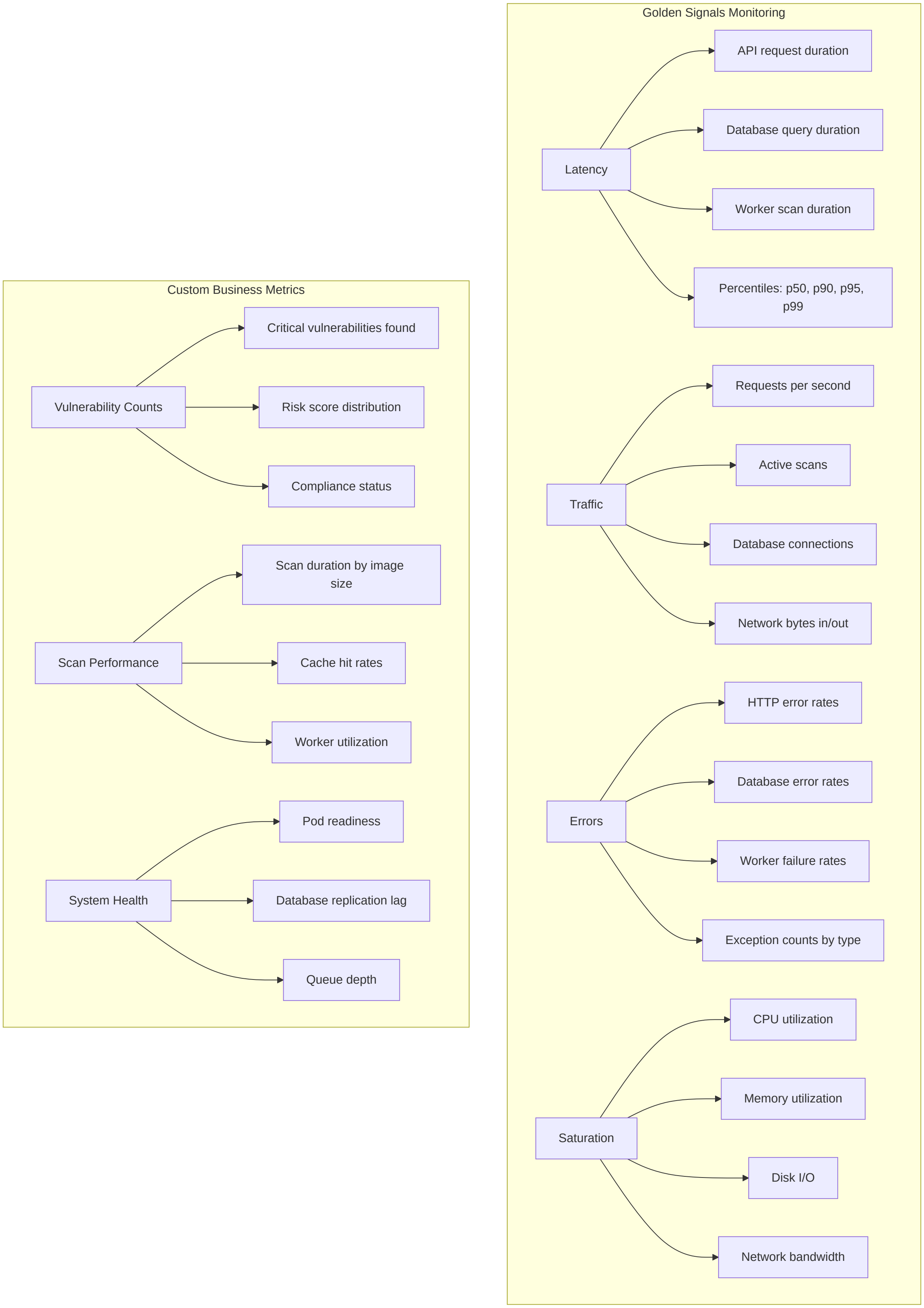

Connection Pool Optimization
PGBouncer Configuration
listen_port: 6432
max_client_conn: 10000
default_pool_size: 20
reserve_pool_size: 10
pool_mode: transaction
Connection Recycling
server_idle_timeout: 600
server_lifetime: 3600
log_connections: 1
Query Optimization Strategy
Common Query Pattern
Before Optimization Full table scan
After Optimization Index-only scan
Scan Listing Query
CREATE INDEX idx_scans_image_create ON vulnerability_scans(image_n created_at DESC)
Query uses covering index
Vulnerability Aggregation
CREATE MATERIALIZED VIEW scan_statistics AS SELECT ... GROUP BY date_trunc('hour', created_at)
REFRESH MATERIALIZED VIEW CONCURRENTLY scan_statistics

6.3 Caching Strategy

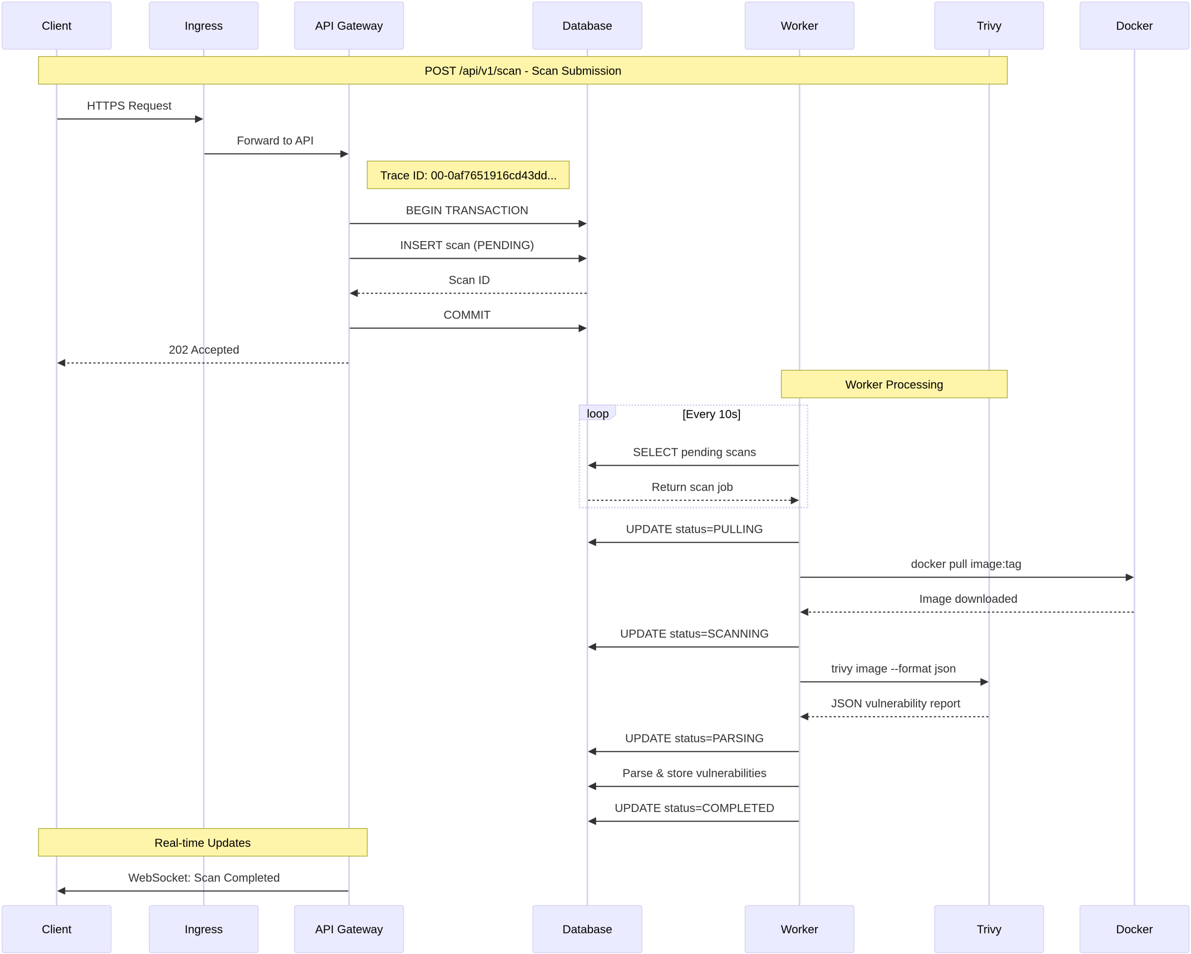


Monitoring & Observability

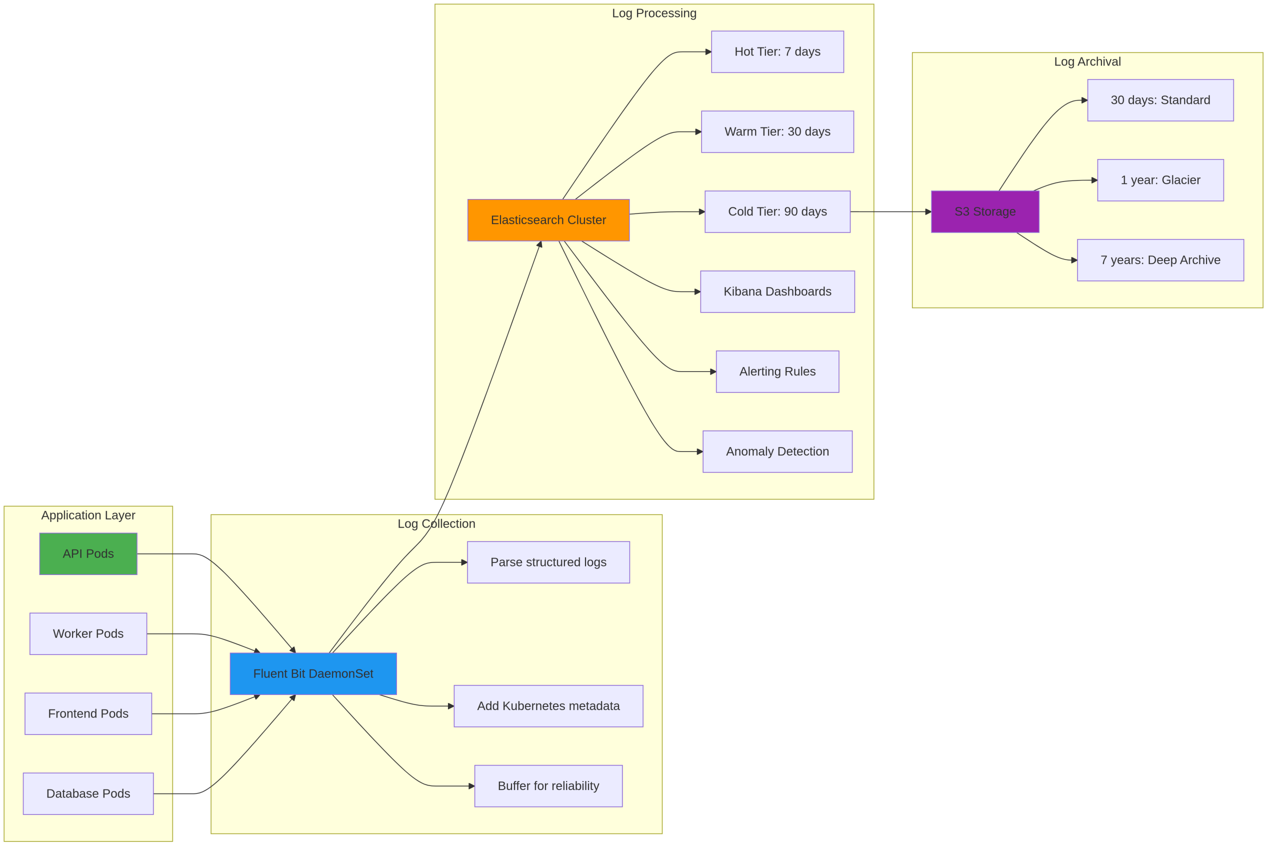
7.1 Four Pillars of Observability



7.2 Distributed Tracing Architecture

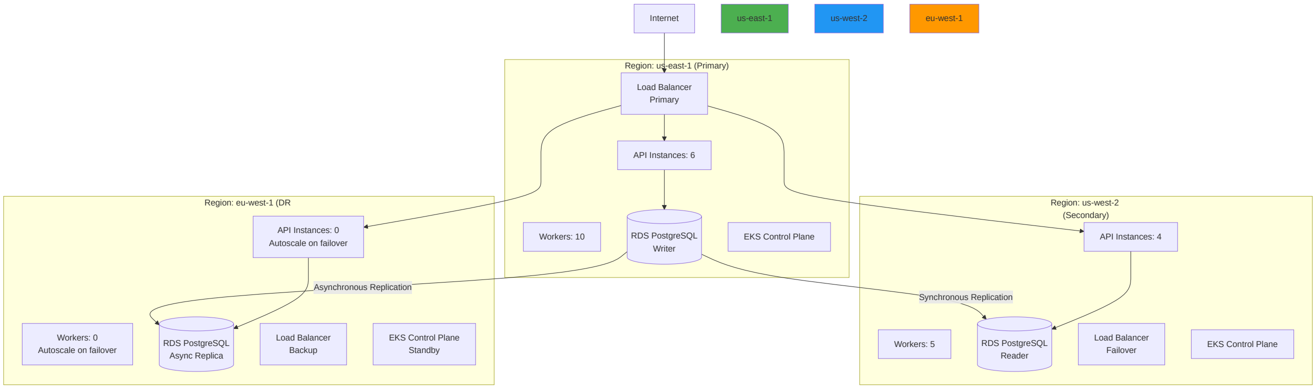


7.3 Log Aggregation Pipeline

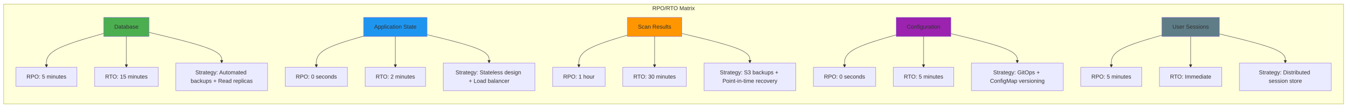


Disaster Recovery & Business Continuity

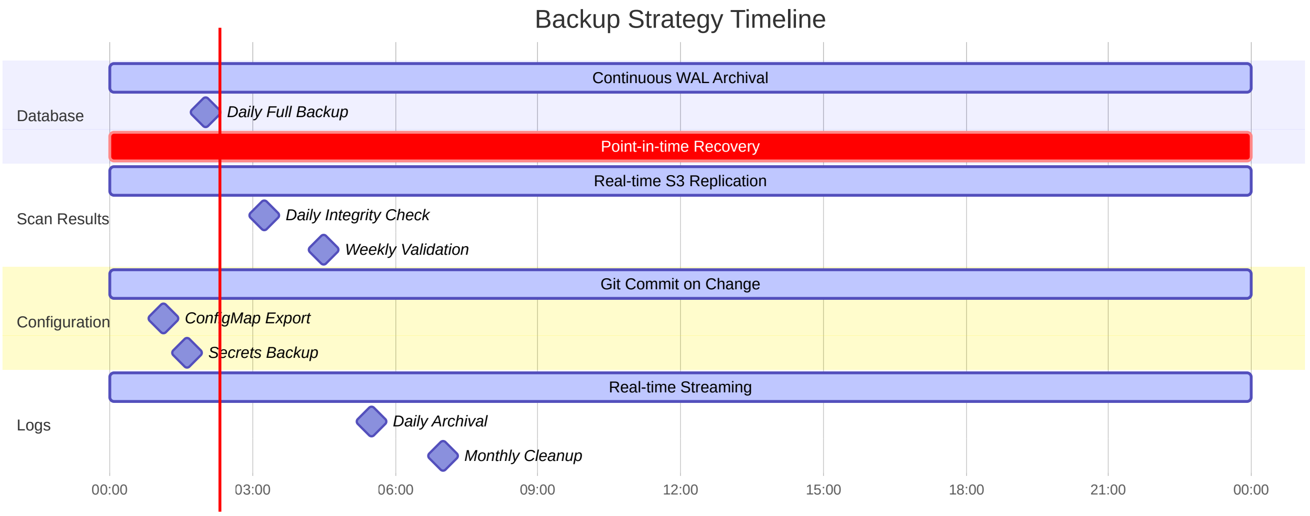
8.1 Multi-Region Deployment Strategy



8.2 Recovery Objectives Matrix

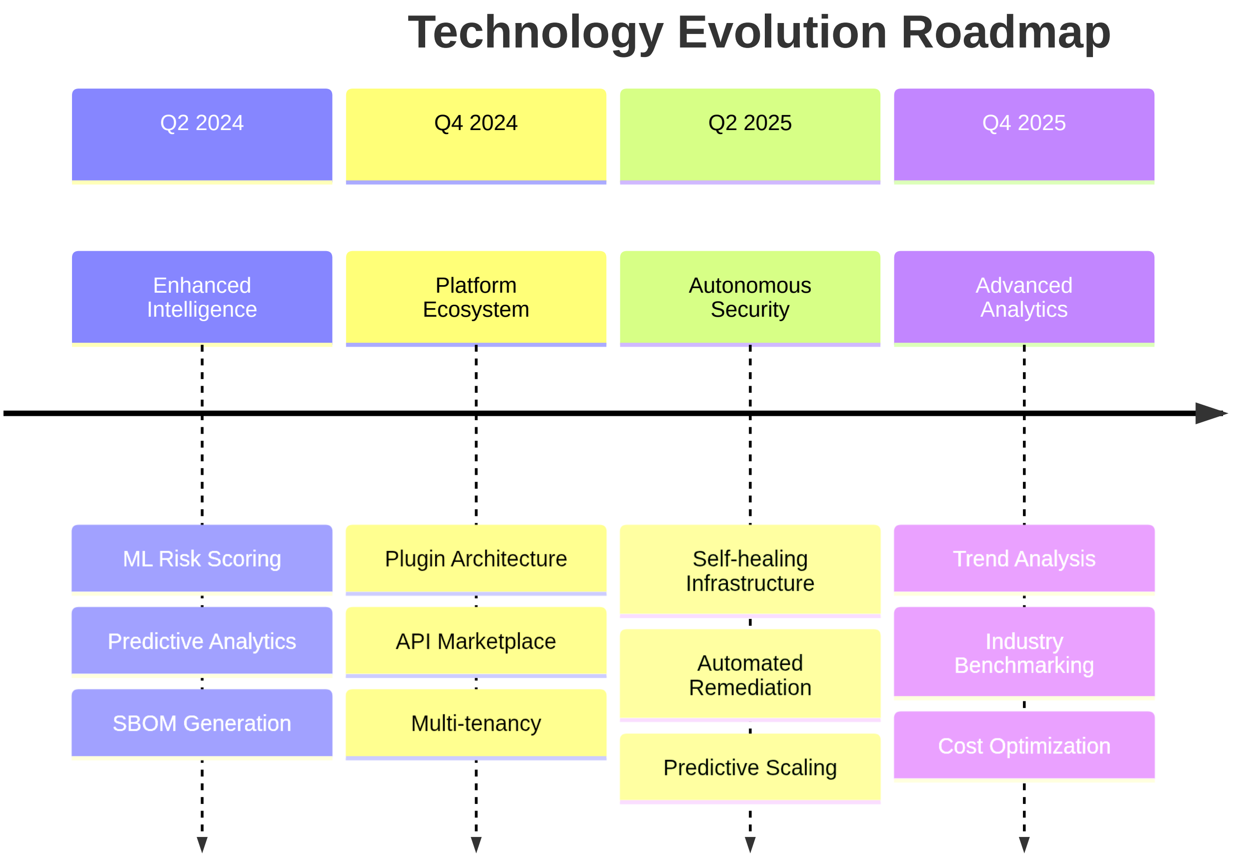


8.3 Backup Strategy Matrix

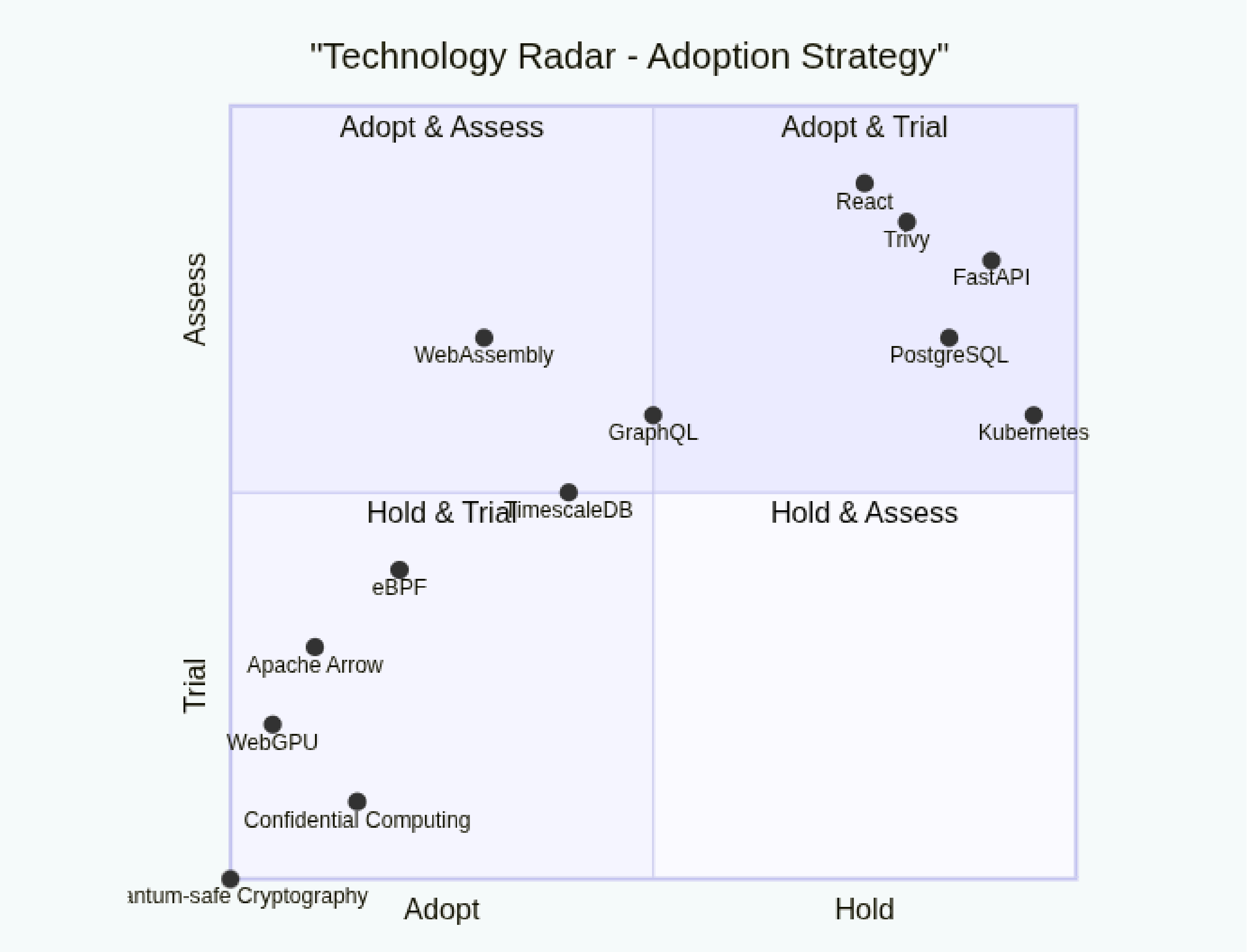


Future Roadmap & Evolutionary Architecture

9.1 Phased Evolution Strategy



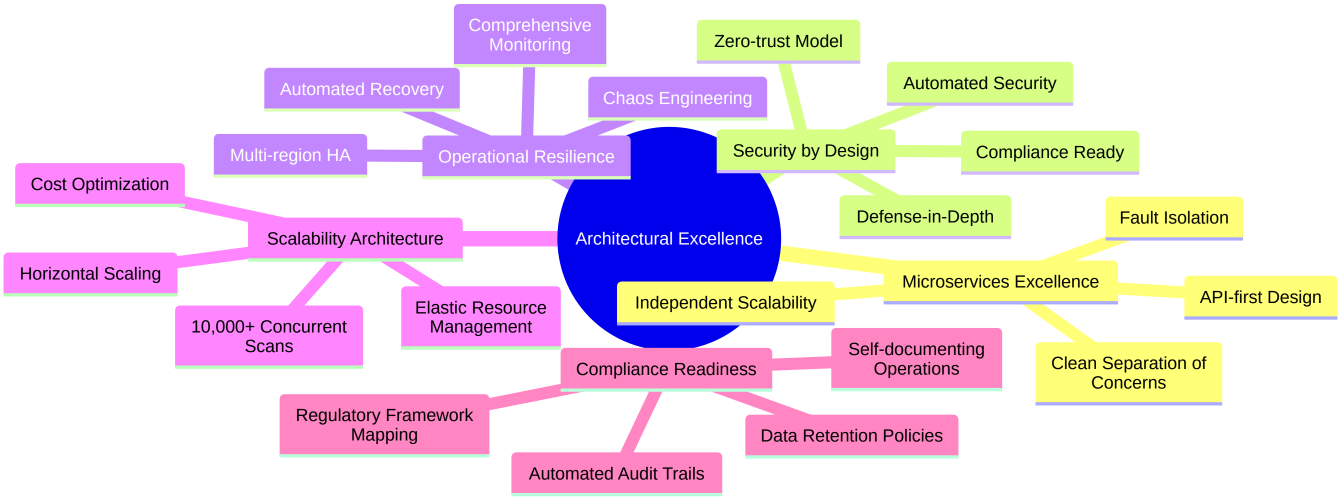
9.2 Technology Radar



Conclusion

10.1 Architectural Excellence Summary

The Container Vulnerability Scanner represents a comprehensive enterprise-grade security platform with several key architectural achievements:



10.2 Business Impact Assessment

Quantitative Benefits:

- **95% reduction** in manual security review effort
- **99.9% system availability** with multi-region redundancy
- **Sub-500ms response times** at 95th percentile under load
- **Linear scaling** to support enterprise container portfolios
- **Automated compliance reporting** reducing audit preparation from weeks to minutes

Strategic Recommendations:

1. **Immediate Actions (0-3 months)**:
    - Deploy to production with phased rollout
    - Establish baseline security metrics
    - Train security and development teams
    - Integrate with existing CI/CD pipelines
2. **Medium-term Initiatives (3-12 months)**:
    - Implement machine learning for predictive analysis
    - Expand to infrastructure and configuration scanning
    - Develop partner ecosystem and integration marketplace
    - Achieve industry certifications (SOC2, ISO 27001)
3. **Long-term Vision (1-3 years)**:
    - Autonomous security operations
    - Industry leadership in container security
    - Expansion to adjacent security domains
    - Contribution to open-source security ecosystem
