Platform and Network Design: Self-Contained Enterprise Architecture

Technical Architecture for Cloud-Independent Enterprise Deployments

---

Executive Summary

This whitepaper details a complete, self-contained platform architecture that provides enterprise-grade capabilities without dependency on external cloud providers. The design centers on Kubernetes RKE2 orchestration with integrated data storage, security, and compute isolation to deliver a "batteries included" solution for organizations requiring infrastructure independence.

Key Components:

- **RKE2 Kubernetes**: Hardened, enterprise-ready Kubernetes distribution
- **Multi-Database Support**: PostgreSQL and MongoDB with automatic clustering
- **Object Storage**: MinIO S3-compatible storage with distributed architecture
- **Identity Management**: Authentik OIDC provider with RBAC integration
- **Compute Isolation**: Kata containers with QEMU-based VM isolation
- **Network Architecture**: Three-tier redundancy with automatic failover

Deployment Options:

- **Self-Hosted**: Complete on-premises deployment for maximum control
- **Managed Service**: Third-party hosted with enterprise SLA guarantees
- **Hybrid**: Mix of on-premises and managed components

Architecture Overview

Design Principles

Cloud Independence

The platform eliminates dependency on external cloud providers through:

- Complete self-contained infrastructure stack
- No external API dependencies for core functionality
- Data sovereignty with on-premises or private cloud deployment
- Vendor-neutral technology choices using open standards

Enterprise Readiness

Built for production enterprise environments:

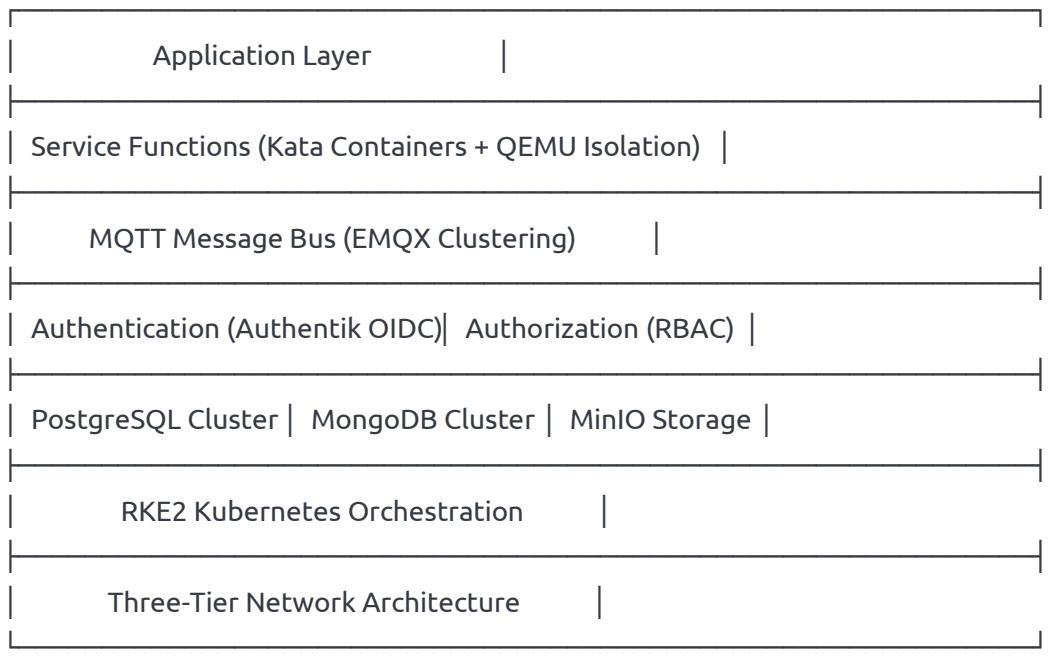
- High availability with no single points of failure
- Automatic scaling and load balancing
- Enterprise-grade security and compliance
- Comprehensive monitoring and observability

Operational Simplicity

Designed for easy deployment and management:

- Single-command installation and upgrades
- Automated configuration and service discovery
- Self-healing infrastructure components
- Unified monitoring and alerting

Core Architecture Stack



Kubernetes Foundation: RKE2

RKE2 Selection Rationale

Enterprise Security Focus:

- FIPS 140-2 compliance for government and financial sectors
- CIS Kubernetes Benchmark compliance out-of-the-box
- SELinux integration for enhanced security policies
- Rootless container runtime options

Hardened by Design:

- Minimal attack surface with reduced component count
- Secure defaults for all configuration options
- Regular security updates and vulnerability patching
- Air-gapped deployment capabilities

Operational Reliability:

- Self-healing cluster components
- Automatic certificate rotation
- Built-in backup and restore capabilities
- Multi-master high availability

Cluster Architecture

Control Plane Configuration

```yaml
# RKE2 Control Plane Setup
cluster-config:
 masters: 3 # Odd number for etcd quorum
 workers: 6 # Minimum for workload distribution

control-plane-nodes:
 - master-1: 10.0.1.10
 - master-2: 10.0.1.11
 - master-3: 10.0.1.12

worker-nodes:
 - worker-1: 10.0.2.10
 - worker-2: 10.0.2.11
 - worker-3: 10.0.2.12
 - worker-4: 10.0.2.13
 - worker-5: 10.0.2.14
 - worker-6: 10.0.2.15
```

Node Specifications

Control Plane Nodes:

- CPU: 8 cores minimum (16 cores recommended)
- Memory: 16GB minimum (32GB recommended)
- Storage: 100GB SSD for etcd
- Network: 10Gbps for inter-node communication

Worker Nodes:

- CPU: 16 cores minimum (32 cores recommended)
- Memory: 64GB minimum (128GB recommended)
- Storage: 500GB NVMe for container runtime
- Network: 10Gbps for service communication

High Availability Configuration

etcd Clustering:

- 3-node etcd cluster for metadata storage
- Automatic leader election and failover
- Regular backups to external storage
- Encryption at rest with configurable keys

Load Balancing:

- HAProxy for API server load balancing
- Keepalived for virtual IP management
- Health checks and automatic failover
- SSL termination and certificate management

Data Storage Architecture

PostgreSQL Clustering

Patroni-Based High Availability

```yaml
postgresql-cluster:
  primary: postgres-primary
  replicas: 2

cluster-config:
  synchronous_mode: true
  maximum_lag_on_failover: 1048576 # 1MB
  use_pg_rewind: true

patroni-settings:
  ttl: 30
  loop_wait: 10
  retry_timeout: 30
  maximum_lag_on_syncnode: -1
```

Features:

- Automatic failover with Patroni consensus
- Streaming replication with synchronous mode
- Point-in-time recovery capabilities
- Connection pooling with PgBouncer integration

Performance Configuration

Memory Tuning:

- shared_buffers: 25% of system RAM
- effective_cache_size: 75% of system RAM
- work_mem: Calculated based on connection count
- maintenance_work_mem: 1-2GB for maintenance operations

Storage Optimization:

- NVMe SSDs for primary storage
- WAL files on separate storage for performance
- Automated archiving to MinIO object storage
- Compression and encryption for archived data

MongoDB Replica Set

Replica Set Architecture

```yaml
mongodb-replicaset:
  primary: mongo-primary
  secondaries: 2
  arbiter: mongo-arbiter  # Optional for tie-breaking

replication-config:
  writeConcern: "majority"
  readConcern: "majority"
  readPreference: "secondaryPreferred"

sharding:
  enabled: true
  config-servers: 3
  shard-count: 2 # Start with 2, scale as needed
```

High Availability Features:

- Automatic primary election
- Replica set health monitoring
- Oplog-based replication
- Sharding for horizontal scaling

Performance Tuning

Memory Management:

- WiredTiger cache size: 50% of system RAM
- Index optimization for common query patterns
- Compression for collections and indexes
- TTL indexes for automatic data expiration

Storage Configuration:

- Dedicated storage for data files
- Separate journal storage for performance
- Automated backup to MinIO storage
- Point-in-time recovery capabilities

MinIO Object Storage

Distributed Architecture

```yaml
minio-cluster:
  nodes: 4
  drives-per-node: 4

erasure-coding:
  data-drives: 2
  parity-drives: 2

configuration:
  storage-class: "REDUCED_REDUNDANCY"
  versioning: enabled
  encryption: enabled
```

S3 Compatibility:

- Full S3 API compatibility for existing applications
- Multi-part upload support for large files
- Bucket policies and access controls
- Cross-region replication capabilities

Storage Features

Data Protection:

- Erasure coding for fault tolerance
- Bitrot protection with checksums
- Automatic healing for corrupted data
- Geographic distribution support

Performance Optimization:

- Parallel upload/download operations
- Intelligent tiering for cost optimization
- Caching layer for frequently accessed data
- Direct-attached storage for maximum performance

Security and Identity Management

Authentik OIDC Provider

Identity Provider Architecture

```yaml
authentik-config:
 database: postgresql
 cache: redis

oidc-providers:
 - name: "platform-sso"
   client-id: "platform-client"
   scopes: ["openid", "profile", "email", "groups"]

flows:
 - authentication: "default-authentication-flow"
 - authorization: "default-authorization-flow"
 - invalidation: "default-invalidation-flow"
```

Authentication Features:

- Multi-factor authentication (TOTP, WebAuthn, SMS)
- Social login integration (Google, Microsoft, LDAP)
- Custom authentication flows and policies
- Session management and timeout controls

User Management

Directory Integration:

- LDAP/Active Directory synchronization
- User provisioning and deprovisioning
- Group membership management
- Custom attribute mapping

Self-Service Capabilities:

- Password reset and account recovery
- Profile management and updates
- Application access requests
- Audit trail for user actions

Role-Based Access Control (RBAC)

Permission Model

```yaml
rbac-model:
  subjects: [users, groups, service-accounts]
  resources: [services, databases, storage, functions]
  actions: [create, read, update, delete, execute]

default-roles:
  - platform-admin: ["*:*:*"]
  - developer: ["services:*:*", "databases:read:*"]
  - operator: ["services:read:*", "databases:read:*"]
  - viewer: ["*:read:*"]
```

Authorization Features:

- Fine-grained permissions at resource level
- Inherited permissions through group membership
- Conditional access based on context
- API-level authorization enforcement

Integration Points

Platform Integration:

- MQTT broker authorization
- Database-level access controls
- Function execution permissions
- Storage bucket policies

Audit and Compliance:

- Complete access audit trail
- Permission change tracking
- Failed access attempt logging
- Compliance reporting capabilities

Compute Isolation: Kata Containers

VM-Based Container Isolation

Kata Container Architecture

```yaml
kata-runtime:
  hypervisor: qemu
  kernel: kata-kernel
  image: kata-image

security-features:
  - vm-isolation: true
  - secure-boot: enabled
  - memory-protection: enabled
  - network-isolation: enforced
```

Security Benefits:

- Hardware-level isolation between containers
- Separate kernel per container workload
- Memory protection and isolation
- Network namespace isolation

Performance Characteristics

Resource Overhead:

- Memory overhead: ~30MB per container
- CPU overhead: <5% for typical workloads
- Storage overhead: ~20MB per container image
- Network latency: <1ms additional latency

Scaling Capabilities:

- Support for 1000+ containers per node
- Fast container startup (sub-second)
- Resource allocation and limits
- Dynamic resource adjustment

QEMU Hypervisor Integration

Hypervisor Configuration

```yaml
qemu-config:
 machine-type: "q35"
 cpu-model: "host"
 memory-backend: "file"

security-options:
 - enable-kvm: true
 - disable-modern: false
 - disable-legacy: true
 - sandbox: "on"
```

Hardware Acceleration:

- KVM acceleration for near-native performance
- VT-x/AMD-V support for hardware virtualization
- IOMMU support for device passthrough
- SR-IOV for network performance

Resource Management

CPU Allocation:

- CPU pinning for performance-critical workloads
- NUMA awareness for memory locality
- CPU hotplug support for dynamic scaling
- Performance monitoring and profiling

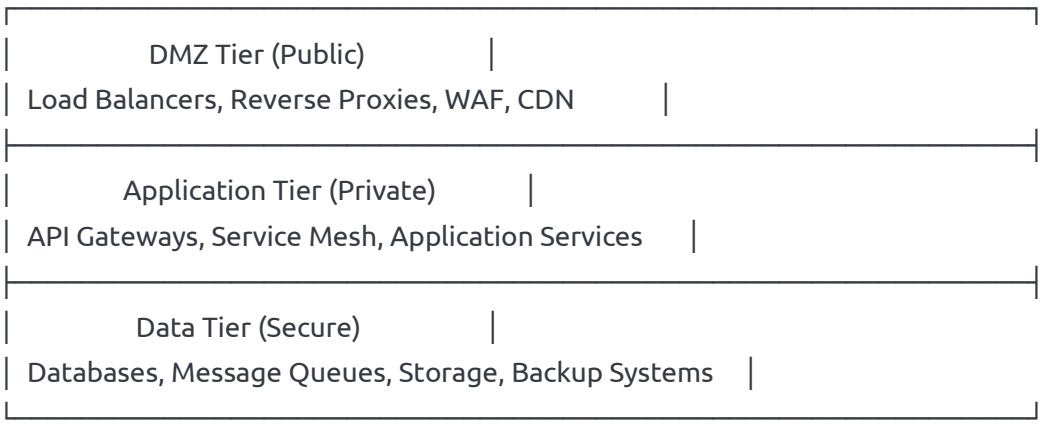
Memory Management:

- Memory ballooning for dynamic allocation
- Huge pages support for performance
- Memory overcommit with monitoring
- NUMA memory allocation policies

Network Architecture

Three-Tier Network Design

Network Topology



Network Segmentation

DMZ Tier (10.0.1.0/24):

- External load balancers and reverse proxies
- Web application firewall (WAF)
- SSL termination and certificate management
- DDoS protection and rate limiting

Application Tier (10.0.2.0/24):

- Kubernetes worker nodes
- Service mesh (Istio/Linkerd)
- Application services and APIs
- Internal load balancing

Data Tier (10.0.3.0/24):

- Database clusters (PostgreSQL, MongoDB)
- Message broker clusters (EMQX)
- Object storage (MinIO)
- Backup and archival systems

Redundancy and Failover

Multi-Site Architecture

```yaml
site-configuration:
  primary-site: "datacenter-1"
  secondary-sites: ["datacenter-2", "datacenter-3"]

replication:
  database: "asynchronous"
  storage: "synchronous"
  configuration: "real-time"

failover:
  automatic: true
  rto: "5 minutes"   # Recovery Time Objective
  rpo: "1 minute"    # Recovery Point Objective
```

Geographic Distribution:

- Active-active configuration across sites
- Automatic failover with health checks
- Data replication with conflict resolution
- Network path optimization

Load Balancing Strategy

Layer 4 Load Balancing:

- HAProxy for TCP/UDP load balancing
- Health checks and automatic failover
- Session persistence and sticky sessions
- SSL passthrough for encrypted traffic

Layer 7 Load Balancing:

- Application-aware routing decisions
- Content-based routing and caching
- API gateway integration
- Request/response transformation

Deployment Models

Self-Hosted Deployment

On-Premises Installation

```bash
# Single-command platform deployment
./platform-installer \
  --cluster-config cluster.yaml \
  --storage-config storage.yaml \
  --network-config network.yaml \
  --security-config security.yaml
```

Installation Process:

1. Hardware validation and preparation
2. Network configuration and validation
3. RKE2 cluster bootstrapping
4. Core service deployment
5. Platform configuration and testing

Resource Requirements:

- Minimum: 9 nodes (3 control plane, 6 workers)
- Recommended: 15 nodes (3 control plane, 12 workers)
- Storage: 10TB minimum (50TB recommended)
- Network: 10Gbps inter-node connectivity

Maintenance and Updates

Automated Operations:

- Rolling updates with zero downtime
- Automatic backup before updates
- Health checks and rollback capabilities
- Configuration drift detection and correction

Manual Operations:

- Hardware maintenance procedures
- Disaster recovery testing
- Performance tuning and optimization
- Security hardening and compliance

Managed Service Deployment

Third-Party Hosting

```yaml
managed-service:
  provider: "certified-partner"
  sla: "99.9% uptime"
  support: "24x7x365"

service-levels:
  - basic: "shared infrastructure"
  - professional: "dedicated nodes"
  - enterprise: "dedicated cluster"
```

Service Features:

- Fully managed infrastructure
- Automatic updates and patching
- 24/7 monitoring and support
- Disaster recovery and backup

SLA Guarantees:

- 99.9% uptime guarantee
- 4-hour response time for critical issues
- Data backup and recovery services
- Security monitoring and incident response

Hybrid Deployment

Mixed Architecture:

- Critical data on-premises
- Compute resources in managed service
- Development environments in cloud
- Disaster recovery across locations

Benefits:

- Risk distribution across environments
- Cost optimization through resource mixing
- Compliance with data sovereignty requirements
- Flexibility for different use cases

Monitoring and Observability

Platform Monitoring

Metrics Collection

```yaml
monitoring-stack:
  metrics: prometheus
  visualization: grafana
  alerting: alertmanager
  tracing: jaeger
  logging: loki

collection-intervals:
  infrastructure: "30s"
  applications: "15s"
  databases: "60s"
```

Infrastructure Metrics:

- CPU, memory, disk, and network utilization
- Kubernetes cluster health and resource usage
- Container and pod lifecycle metrics
- Storage performance and capacity

Application Metrics:

- Request rates, latency, and error rates
- Service dependency mapping
- Business metric tracking
- Custom application metrics

Alerting Strategy

Alert Hierarchy:

- Critical: Immediate action required (page on-call)
- Warning: Attention needed within hours
- Info: Awareness for trending and planning

Alert Channels:

- PagerDuty for critical infrastructure alerts
- Slack for team notifications
- Email for informational alerts
- SMS for out-of-hours critical alerts

Application Performance Monitoring

Distributed Tracing

```yaml
tracing-config:
  sampler: "probabilistic"
  sampling-rate: 0.1 # 10% of requests

collectors:
  - jaeger-collector
  - tempo-collector

exporters:
  - jaeger
  - prometheus
  - elasticsearch
```

Trace Collection:

- Automatic instrumentation for supported languages
- Custom span creation for business logic
- Correlation across service boundaries
- Performance bottleneck identification

Analysis Capabilities:

- Request flow visualization
- Latency analysis and optimization
- Error tracking and debugging
- Dependency mapping and impact analysis

Log Aggregation

Centralized Logging:

- Structured logging with JSON format
- Log correlation with trace IDs
- Real-time log streaming and alerts
- Long-term storage and archival

Log Sources:

- Application logs from containers
- Infrastructure logs from nodes
- Audit logs from security systems
- Performance logs from databases

Backup and Disaster Recovery

Data Protection Strategy

Automated Backup Systems

```yaml
backup-configuration:
 postgresql:
  method: "pg_basebackup + WAL archiving"
  frequency: "continuous WAL + daily base backup"
  retention: "30 days point-in-time recovery"

 mongodb:
  method: "mongodump + oplog tailing"
  frequency: "hourly incremental + daily full"
  retention: "30 days"

 minio:
  method: "bucket replication + versioning"
  frequency: "real-time replication"
  retention: "90 days with lifecycle policies"

 kubernetes:
  method: "velero cluster backup"
  frequency: "daily cluster state backup"
  retention: "14 days"
```

Backup Features:

- Automated scheduling and execution
- Incremental backups for efficiency
- Encryption at rest and in transit
- Cross-site replication for disaster recovery

Recovery Procedures

Recovery Time Objectives (RTO):

- Database recovery: 15 minutes
- Application recovery: 5 minutes
- Full cluster recovery: 2 hours
- Cross-site failover: 30 minutes

Recovery Point Objectives (RPO):

- Critical data: 1 minute
- Application data: 5 minutes
- Configuration data: 15 minutes
- Log data: 30 minutes

Disaster Recovery Architecture

Multi-Site Replication

```yaml
disaster-recovery:
  primary-site: "site-a"
  dr-sites: ["site-b", "site-c"]

replication-topology:
  databases: "master-slave with promotion"
  storage: "cross-site synchronous replication"
  configuration: "git-based configuration as code"

failover-automation:
  health-checks: "multi-layer health monitoring"
  decision-engine: "consensus-based failover"
  rollback: "automatic rollback on failure"
```

Failover Scenarios:

- Planned maintenance with zero downtime
- Site-level disaster with automatic failover
- Network partition with split-brain protection
- Cascading failure with graceful degradation

Business Continuity

Communication Plans:

- Automated stakeholder notifications
- Status page updates for users
- Internal team coordination procedures
- Customer communication templates

Recovery Validation:

- Regular disaster recovery testing
- Automated recovery verification
- Performance validation post-recovery
- Data integrity verification

Security Architecture

Network Security

Firewall Configuration

```yaml
firewall-rules:
  dmz-tier:
    inbound: ["80/tcp", "443/tcp", "22/tcp (admin only)"]
    outbound: ["application-tier:8080"]

  application-tier:
    inbound: ["dmz-tier:8080", "application-tier:*"]
    outbound: ["data-tier:5432", "data-tier:27017"]

  data-tier:
    inbound: ["application-tier:5432", "application-tier:27017"]
    outbound: ["backup-storage:443"]
```

Security Controls:

- Zero-trust network architecture
- Micro-segmentation between tiers
- Intrusion detection and prevention
- DDoS protection and rate limiting

Certificate Management

PKI Infrastructure:

- Internal certificate authority (CA)
- Automated certificate lifecycle management
- Regular rotation and renewal
- Certificate transparency logging

SSL/TLS Configuration:

- TLS 1.3 minimum for all connections
- Perfect forward secrecy (PFS)
- HSTS headers for web traffic
- Certificate pinning for critical services

Data Encryption

Encryption at Rest

```yaml
encryption-config:
 databases:
  postgresql: "LUKS + TDE"
  mongodb: "WiredTiger encryption"

 storage:
  minio: "AES-256-GCM"
  kubernetes-secrets: "envelope encryption"

 backups:
  method: "GPG encryption"
  key-rotation: "quarterly"
```

Key Management:

- Hardware Security Module (HSM) integration
- Key rotation policies and procedures
- Secure key distribution and storage
- Audit trail for key operations

Encryption in Transit

Network Encryption:

- mTLS for all service-to-service communication
- VPN tunnels for cross-site replication
- Encrypted backup transmission
- Secure remote access protocols

Performance and Scaling

Horizontal Scaling

Auto-Scaling Configuration

```yaml
scaling-policies:
 worker-nodes:
  min-nodes: 6
  max-nodes: 50
  scale-up-threshold: "CPU > 70% for 5 minutes"
  scale-down-threshold: "CPU < 30% for 10 minutes"

 databases:
  postgresql: "read replicas based on connection count"
  mongodb: "sharding based on collection size"

 storage:
  minio: "add nodes when storage > 80% capacity"
```

Scaling Triggers:

- Resource utilization thresholds
- Queue depth and processing time
- Response time degradation
- Business metric thresholds

Performance Optimization

Database Tuning:

- Connection pooling and optimization
- Query optimization and indexing
- Caching strategies and implementation
- Partitioning for large datasets

Network Optimization:

- CDN integration for static content
- Compression and optimization
- Connection pooling and reuse
- Load balancing algorithms

Resource Management

Capacity Planning

```yaml
capacity-planning:
 compute:
  target-utilization: "70%"
  growth-buffer: "30%"
  planning-horizon: "12 months"

 storage:
  growth-rate: "20% annually"
  retention-policies: "tiered storage"
  backup-overhead: "2x primary storage"

 network:
  bandwidth-utilization: "60%"
  latency-targets: "<10ms inter-tier"
  redundancy: "N+1 for critical paths"
```

Resource Allocation:

- Quality of Service (QoS) policies
- Resource quotas and limits
- Priority-based scheduling
- Resource reservation for critical workloads

Cost Analysis

Total Cost of Ownership

Infrastructure Costs (Annual)

```yaml
cost-breakdown:
  hardware:
    servers: "$200K - $500K"
    storage: "$100K - $300K"
    networking: "$50K - $150K"

  software:
    platform-license: "$100K - $300K"
    monitoring-tools: "$20K - $50K"
    security-tools: "$30K - $80K"

  operations:
    personnel: "$300K - $600K"
    maintenance: "$50K - $100K"
    utilities: "$40K - $80K"
```

5-Year TCO Projection:

- Initial investment: $500K - $1.2M
- Annual operational costs: $540K - $1.16M
- **Total 5-year cost: $3.2M - $7M**

Cost Comparison

vs. Public Cloud (AWS/Azure/GCP):

- 40-60% cost savings for steady-state workloads
- No data egress charges
- Predictable pricing model
- No vendor lock-in costs

vs. Traditional Data Center:

- 30-50% operational efficiency improvement
- Reduced management overhead
- Faster deployment and scaling
- Modern tooling and automation

Return on Investment

Cost Savings Sources

- **Infrastructure Consolidation**: Single platform vs. multiple systems
- **Operational Efficiency**: Automated management and scaling
- **Developer Productivity**: Simplified development and deployment
- **Reduced Vendor Costs**: Open-source foundation with enterprise features

Business Value Creation

- **Faster Time-to-Market**: Rapid application development and deployment
- **Improved Reliability**: Enterprise-grade availability and performance
- **Enhanced Security**: Built-in security controls and compliance
- **Data Sovereignty**: Complete control over data location and access

---

Implementation Roadmap

Phase 1: Foundation (Months 1-3)

Infrastructure Setup:

- Hardware procurement and data center preparation
- Network design and implementation
- RKE2 cluster deployment and validation
- Core platform services installation

Deliverables:

- Functional Kubernetes cluster
- Basic monitoring and alerting
- Network security implementation
- Initial platform deployment

Phase 2: Platform Services (Months 3-6)

Service Deployment:

- Database clusters (PostgreSQL, MongoDB)
- Object storage (MinIO) deployment
- Identity management (Authentik) setup
- Message broker (EMQX) clustering

Deliverables:

- Production-ready data services
- Complete authentication system
- Backup and disaster recovery
- Performance monitoring

Phase 3: Production Readiness (Months 6-9)

Hardening and Optimization:

- Security audit and hardening
- Performance tuning and optimization
- Disaster recovery testing
- Documentation and training

Deliverables:

- Production security certification
- Optimized performance baselines
- Validated disaster recovery procedures
- Operational runbooks and training

Phase 4: Advanced Features (Months 9-12)

Enhanced Capabilities:

- Multi-site deployment
- Advanced monitoring and analytics
- Automation and self-healing
- Integration with existing systems

Deliverables:

- Geographic redundancy
- Advanced operational capabilities
- Complete automation framework
- Integration with enterprise systems

Risk Assessment and Mitigation

Technical Risks

Infrastructure Complexity

Risk: Complex distributed systems may be difficult to troubleshoot **Mitigation**:

- Comprehensive monitoring and alerting
- Automated diagnostics and self-healing
- Expert support and training programs
- Detailed documentation and runbooks

Performance at Scale

Risk: Unproven performance characteristics at enterprise scale **Mitigation**:

- Extensive load testing and benchmarking
- Reference architectures for large deployments
- Performance monitoring and optimization tools
- Gradual scaling with validation

Security Vulnerabilities

Risk: New platform may have unknown security issues **Mitigation**:

- Regular security audits and penetration testing
- Automated vulnerability scanning
- Security-focused development practices
- Incident response procedures

Operational Risks

Skills Gap

Risk: Team may lack expertise in new technologies **Mitigation**:

- Comprehensive training programs
- Expert consulting and support
- Knowledge transfer and documentation
- Gradual migration with parallel systems

Vendor Dependency

Risk: Dependence on platform vendor for critical functions **Mitigation**:

- Open-source foundation with vendor-neutral components
- Source code access and escrow agreements
- Multi-vendor support options
- Migration and exit strategies

Conclusion

The platform and network design provides a complete, self-contained enterprise architecture that eliminates cloud provider dependencies while delivering enterprise-grade capabilities. The design balances operational simplicity with technical sophistication, providing organizations with the infrastructure foundation needed for modern application development.

Key Benefits:

- **Infrastructure Independence**: Complete freedom from cloud vendor lock-in
- **Enterprise Readiness**: Production-grade reliability, security, and scalability
- **Operational Simplicity**: Automated deployment, management, and scaling
- **Cost Effectiveness**: Significant savings compared to traditional and cloud approaches
- **Future-Proof Architecture**: Modern, scalable foundation for long-term growth

Strategic Advantages:

- Data sovereignty and regulatory compliance
- Predictable costs and performance
- Customizable to specific organizational needs
- Professional support and service options
- Clear migration path from existing systems

The architecture represents a modern approach to enterprise infrastructure that combines the best aspects of cloud-native design with the control and independence of on-premises deployment.