Architecture Decision Record (ADR) Template:

```markdown
# ADR-001: Time Series Database Selection

## Status
Accepted

## Context
We need to store and query 500M metric data points per day with 90-day retention. Current metrics include:
- API request rates and latencies
- Error rates and status codes
- Resource utilization (CPU, memory, network)
- Business metrics (transactions, revenue)

Requirements:
- Write throughput: 5,000+ writes/second
- Query performance: <3 seconds for dashboard queries
- Retention: 90 days minimum
- Cost: <$5,000/month for infrastructure

## Options Considered

### Option 1: Prometheus
**Pros:**
- Native Kubernetes integration
- Pull-based model (service discovery)
- Powerful query language (PromQL)
- Wide ecosystem support
- Open source (no licensing cost)

**Cons:**
- Limited long-term storage (15-30 days)
- No built-in high availability
```

- Resource intensive for very large scale

Cost: ~$2,000/month (infrastructure only)

Option 2: InfluxDB
Pros:
- Excellent write performance
- Built-in retention policies
- Clustering for high availability
- Good for time-series workloads

Cons:
- Enterprise features require license
- Less Kubernetes-native
- Smaller ecosystem vs. Prometheus

Cost: ~$4,000/month (including license)

Option 3: TimescaleDB
Pros:
- PostgreSQL compatibility (SQL)
- Unlimited retention
- Compression capabilities
- Familiar tooling

Cons:
- More complex queries for time-series
- Less purpose-built for metrics
- Requires more optimization

Cost: ~$3,000/month

Decision
Selected: Prometheus + Thanos

We chose Prometheus with Thanos for long-term storage because:
1. Best integration with our Kubernetes infrastructure
2. Team already familiar with PromQL
3. Thanos solves long-term storage limitation
4. Cost-effective for our scale
5. Strong community and ecosystem

Consequences

Positive:
- Seamless Kubernetes service discovery
- No application code changes needed
- Powerful query capabilities
- Cost within budget

Negative:
- Need to operate Thanos for long-term storage
- Pull model requires network configuration
- Learning curve for Thanos architecture

Mitigation:
- Implement Thanos with sidecar pattern
- Document Thanos operation procedures
- Provide team training on Thanos architecture

Implementation Plan
1. Deploy Prometheus in each cluster (Week 1)
2. Configure service monitors for APIs (Week 2)
3. Deploy Thanos components (Week 3)
4. Migrate historical data (Week 4)
5. Validate and document (Week 5)

Review Date

90 days post-implementation

References
- [Prometheus Documentation](https://prometheus.io/docs/)
- [Thanos Architecture](https://thanos.io/tip/thanos/design.md/)
- [Cost Analysis Spreadsheet](link)