1. System Design Overview

Our cloud cost optimization system leverages Atlan's existing architecture while introducing specialized components to monitor, analyze, and optimize cloud costs in real-time. The solution addresses critical challenges including delayed cost detection, source identification, and streamlined optimization.

Key Components

Data Collection Layer

- **Cloud Provider APIs**: Direct integration with AWS, Azure, and GCP cost management APIs to extract billing data at hourly intervals.
- **VictoriaMetrics**: Time-series database that stores all cost metrics with minimal storage footprint and high query performance.
- **Fluent Bit**: Collects and processes resource logs to identify usage patterns contributing to costs.

Analysis & Visualization Layer

- **PostgreSQL**: Stores resource metadata including tags, team allocations, and historical cost patterns.
- **Grafana**: Provides interactive dashboards for cost visualization across services, teams, and projects.

Automation Layer

- **Apache Kafka**: Event streaming platform that triggers optimization workflows when cost anomalies are detected.
- **Argo Workflows**: Orchestrates automated optimization tasks such as resource rightsizing and cleanup.
- **Admission Controller**: Enforces cost-saving policies for new workloads and resources.

Security Layer

- **HashiCorp Vault**: Securely manages cloud provider credentials with strict access controls.

Alerting Layer

- **Alertmanager**: Processes cost threshold violations and anomalies.
- **Zenduty**: Manages incident response for critical cost issues.

2. Design Decisions & Rationale

Why VictoriaMetrics?

VictoriaMetrics was selected over Prometheus for several key reasons:

- **Scalability**: Capable of ingesting millions of metrics per second
- **Long-term storage**: More efficient for storing historical cost data (up to years)

- **Cost-effectiveness**: Requires significantly less storage than alternatives
- **Compatibility**: Native support for PromQL while offering better performance

Why Kafka for Event Processing?

- **Decoupling**: Separates anomaly detection from remediation actions
- **Reliability**: Ensures no optimization opportunities are missed, even during system maintenance
- **Scalability**: Can handle spikes in cost events during cloud provider price changes

Why Argo Workflows?

- **Kubernetes-native**: Integrates seamlessly with Atlan's existing infrastructure
- **Auditable**: Every optimization action is tracked and can be rolled back if necessary
- **Templatable**: Common optimization patterns can be standardized and reused

3. Proof of Solution Effectiveness

Metrics & KPIs

The solution's effectiveness can be measured through:

1. **Detection Latency**: Reduction from days to hours or minutes
    - Before: Cost spikes detected after monthly billing cycle
    - After: Anomalies flagged within 1 hour of occurrence
2. **Cost Attribution Accuracy**: Improvement in identifying cost sources
    - Before: Attribution limited to service-level (e.g., "S3 costs increased")
    - After: Granular attribution (e.g., "Team A's project X storage increase of 30%")
3. **Optimization Time**: Reduction in time to implement cost-saving measures
    - Before: Manual process taking days to weeks
    - After: Automated remediation within hours for common issues
4. **Return on Investment**: Projected savings vs. implementation cost
    - Implementation cost: Approximately 80 engineering hours
    - Projected annual savings: 15-25% of cloud spend

Implementation Impact

Based on industry benchmarks and internal testing:

- Potential to reduce "zombie" infrastructure by 30%
- Rightsizing opportunities typically yield 20-40% savings
- Enhanced visibility leads to 15% better resource planning

4. Known Gaps & Limitations

1. **Provider-Specific Limitations**
    - Cloud provider APIs have varying delays in cost data availability
    - GCP billing data typically has a 6-hour lag compared to AWS/Azure
2. **Tagging Dependencies**
    - System effectiveness depends on consistent resource tagging
    - Historical resources may lack proper tags, limiting initial analysis
3. **Optimization Constraints**
    - Some optimizations require maintenance windows
    - Mission-critical systems may have limited optimization opportunities
4. **Edge Cases**
    - Reserved instance/commitment recommendations require human review
    - Multi-cloud cost optimization is more complex than single-cloud

5. Implementation Roadmap

Phase 1: Foundation (Weeks 1-2)

- Set up VictoriaMetrics and initial data collection
- Implement basic Grafana dashboards
- Configure cloud provider API integrations

Phase 2: Analysis & Alerting (Weeks 3-4)

- Develop comprehensive tagging strategy
- Implement alerting thresholds
- Create team-specific dashboards

Phase 3: Automation (Weeks 5-6)

- Set up Kafka streams for anomaly events
- Develop initial Argo Workflows for common optimizations
- Implement feedback loops for optimization verification

Phase 4: Refinement (Weeks 7-8)

- Fine-tune alerting thresholds
- Add machine learning for cost prediction
- Document best practices for teams