

Developer Cost Optimization Checklist

30-Day Implementation Guide for Building Cost-Efficient Cloud Applications

Your practical roadmap to embedding cost optimization into your development workflow

Overview

This comprehensive checklist provides a structured 30-day approach to implementing cost optimization practices in your development workflow. Each week focuses on specific areas, building upon previous foundations to create a sustainable cost-aware development culture.

Expected Outcomes: - 20-40% reduction in development environment costs - Improved cost visibility and accountability - Automated cost optimization processes - Team-wide cost awareness and best practices

Week 1: Foundation Setup

Building the groundwork for cost-aware development

Day 1-2: Cost Monitoring Setup

☐ **Set up cost monitoring and alerting** - ☐ Configure cost alerts for your development subscription/account - ☐ Set budget thresholds: 80% warning, 100% critical - ☐ Create cost dashboard accessible to all team members - ☐ Document baseline costs for current development environment

Tools & Resources: - Azure: Cost Management + Billing alerts - AWS: Cost Explorer + Budgets - GCP: Cloud Billing + Budget alerts

Success Criteria: ✓ Alerts configured and tested ✓ Team has access to cost dashboard
✓ Baseline costs documented

Day 3-4: Resource Tagging Strategy

☐ **Implement consistent resource tagging** - [] Define tagging taxonomy (Environment, Owner, Project, CostCenter) - [] Create tagging policy document - [] Tag all existing development resources - [] Set up automated tagging for new resources

Required Tags:

```
Environment: Development/Staging/Production
Owner: team-email@company.com
Project: project-name
CostCenter: department-code
AutoShutdown: true/false
```

Success Criteria: ✓ All resources properly tagged ✓ Tagging automation in place ✓ Team trained on tagging standards

Day 5-7: Cost Estimation Process

☐ **Create cost estimation workflow** - [] Install and configure cost estimation tools (Infracost, Azure Pricing Calculator) - [] Create cost estimation template for new features - [] Integrate cost estimation into planning process - [] Train team on cost estimation tools

Cost Estimation Template: - Infrastructure components and sizing - Expected usage patterns - Data transfer requirements - Storage needs and access patterns - Estimated monthly cost range

Success Criteria: ✓ Cost estimation tools configured ✓ Template created and documented ✓ Team trained on estimation process



Week 2: Architecture Review

Optimizing your application architecture for cost efficiency

Day 8-10: Serverless Opportunities Audit

☐ **Identify serverless migration candidates** - [] Audit current application components - [] Identify event-driven workloads - [] Evaluate API endpoints for serverless conversion - [] Assess batch processing jobs for serverless

Evaluation Criteria: - Sporadic or event-driven usage patterns - Stateless processing requirements - Short execution times (< 15 minutes) - Variable load patterns

☐ **Plan serverless migrations** - [] Prioritize components by cost impact - [] Create migration timeline - [] Estimate cost savings potential - [] Document migration approach

Success Criteria: ✓ Serverless candidates identified ✓ Migration plan created ✓ Cost savings estimated

Day 11-12: Caching Strategy Implementation

☐ **Implement multi-level caching** - [] Audit current data access patterns - [] Identify frequently accessed data - [] Implement application-level caching - [] Configure CDN for static content - [] Set up database query caching

Caching Layers: 1. Browser caching (static assets) 2. CDN caching (global content) 3. Application caching (Redis/Memcached) 4. Database query caching

☐ **Measure caching effectiveness** - [] Monitor cache hit rates - [] Track performance improvements - [] Calculate cost savings from reduced compute

Success Criteria: ✓ Multi-level caching implemented ✓ Cache hit rates > 80% ✓ Measurable performance improvement

Day 13-14: Database Optimization

☐ **Optimize database performance and costs** - [] Audit database queries and performance - [] Implement proper indexing strategy - [] Optimize expensive queries - [] Review database sizing and scaling - [] Implement connection pooling

Database Optimization Checklist: - ☐ Identify slow queries (> 1 second) - ☐ Add missing indexes - ☐ Optimize JOIN operations - ☐ Implement query result caching - ☐ Right-size database instances

Success Criteria: ✓ Query performance improved by 50%+ ✓ Database costs optimized ✓ Proper indexing in place



Week 3: Automation Implementation

Automating cost optimization processes

Day 15-17: Environment Lifecycle Automation

☐ **Implement automated environment management** - ☐ Set up automated environment creation - ☐ Configure environment destruction policies - ☐ Implement branch-based environment lifecycle - ☐ Create environment scheduling (auto-shutdown)

Environment Lifecycle Rules: - Development environments: Auto-shutdown after 8 hours - Feature branch environments: Auto-destroy after 7 days of inactivity - Staging environments: Auto-shutdown outside business hours - Temporary environments: Auto-destroy after 24 hours

☐ **Configure auto-shutdown policies** - ☐ Implement VM auto-shutdown schedules - ☐ Configure database auto-pause for dev environments - ☐ Set up container auto-scaling to zero - ☐ Create weekend shutdown automation

Success Criteria: ✓ Environment lifecycle automation active ✓ Auto-shutdown policies implemented ✓ 40%+ reduction in off-hours costs

Day 18-19: Data Lifecycle Policies

☐ **Implement storage optimization** - ☐ Audit current storage usage and costs - ☐ Implement data lifecycle policies - ☐ Configure automated data archival - ☐ Set up automated cleanup of temporary data

Storage Lifecycle Policies: - Logs: Move to cool storage after 30 days, archive after 90 days - Backups: Move to archive storage after 30 days - Test data: Auto-delete after 7

days - Build artifacts: Retain for 30 days, then delete

☐ **Optimize storage tiers** - [] Move infrequently accessed data to cool storage - [] Implement intelligent tiering - [] Configure compression for archived data - [] Set up automated cleanup scripts

Success Criteria: ✓ Data lifecycle policies active ✓ Storage costs reduced by 30%+ ✓ Automated cleanup processes running

Day 20-21: Auto-Scaling Configuration

☐ **Configure intelligent auto-scaling** - [] Implement horizontal auto-scaling policies - [] Set appropriate scaling thresholds - [] Configure scale-down policies - [] Test scaling behavior under load

Auto-Scaling Best Practices: - Scale up quickly (2-3 minutes) - Scale down slowly (10-15 minutes) - Set minimum instances to 1 for dev environments - Use predictive scaling for known patterns

☐ **Optimize scaling policies** - [] Monitor scaling events and costs - [] Adjust thresholds based on actual usage - [] Implement scheduled scaling for predictable loads - [] Configure alerts for scaling anomalies

Success Criteria: ✓ Auto-scaling policies configured ✓ Scaling behavior optimized ✓ Cost-effective scaling thresholds set



Week 4: Monitoring & Optimization

Establishing ongoing cost optimization practices

Day 22-24: Cost Dashboard Creation

☐ **Build comprehensive cost dashboard** - [] Create team cost visibility dashboard - [] Implement cost trend analysis - [] Set up cost anomaly detection - [] Configure automated cost reports

Dashboard Components: - Daily/weekly/monthly cost trends - Cost breakdown by service/environment - Cost per feature/project - Budget vs. actual spending - Top cost

contributors - Optimization opportunities

☐ **Implement cost alerting** - [] Set up intelligent cost alerts - [] Configure anomaly detection - [] Create escalation procedures - [] Test alert responsiveness

Success Criteria: ✓ Comprehensive dashboard deployed ✓ Team has cost visibility ✓ Alerts configured and tested

Day 25-26: Team Cost Review Process

☐ **Establish regular cost review meetings** - [] Schedule monthly cost review meetings - [] Create cost review agenda template - [] Define cost optimization KPIs - [] Assign cost optimization responsibilities

Monthly Cost Review Agenda: 1. Cost trends and anomalies 2. Budget vs. actual analysis 3. Optimization opportunities identified 4. Action items from previous month 5. New optimization initiatives 6. Team cost awareness training

☐ **Create cost optimization playbook** - [] Document common optimization scenarios - [] Create troubleshooting guides - [] Establish escalation procedures - [] Define success metrics

Success Criteria: ✓ Regular cost review process established ✓ Team cost responsibilities defined ✓ Optimization playbook created

Day 27-30: Documentation & Knowledge Sharing

☐ **Document lessons learned** - [] Create cost optimization knowledge base - [] Document successful optimization strategies - [] Share lessons learned with broader organization - [] Create training materials for new team members

Knowledge Base Contents: - Cost optimization strategies that worked - Common pitfalls and how to avoid them - Tool configurations and best practices - Cost monitoring and alerting setup - Troubleshooting guides

☐ **Plan continuous improvement** - [] Identify areas for further optimization - [] Plan next month's optimization initiatives - [] Set up quarterly cost optimization reviews - [] Create feedback loop for ongoing improvement

Success Criteria: ✓ Knowledge base created and populated ✓ Lessons learned documented ✓ Continuous improvement plan established



Success Metrics & KPIs

Week 1 Targets

- ☐ 100% of resources properly tagged
- ☐ Cost monitoring and alerting active
- ☐ Baseline costs documented

Week 2 Targets

- ☐ 50% improvement in query performance
- ☐ Caching hit rates > 80%
- ☐ Serverless migration plan created

Week 3 Targets

- ☐ 40% reduction in off-hours costs
- ☐ 30% reduction in storage costs
- ☐ Auto-scaling policies active

Week 4 Targets

- ☐ Cost dashboard deployed and used
- ☐ Monthly cost review process established
- ☐ Team cost optimization knowledge documented

Overall 30-Day Targets

- ☐ 20-40% reduction in development environment costs
 - ☐ 100% cost visibility across all resources
 - ☐ Automated cost optimization processes in place
 - ☐ Team-wide cost awareness established
-

Tools & Resources

Cost Monitoring Tools

- **Azure:** Cost Management + Billing, Azure Advisor
- **AWS:** Cost Explorer, AWS Budgets, Trusted Advisor
- **GCP:** Cloud Billing, Recommender
- **Multi-Cloud:** CloudHealth, Cloudability

Infrastructure as Code

- **Terraform** with Infracost for cost estimation
- **ARM Templates** with Azure Cost Estimator
- **CloudFormation** with AWS Cost Calculator

Automation Tools

- **Azure:** Logic Apps, Azure Functions, Azure Automation
- **AWS:** Lambda, EventBridge, Systems Manager
- **GCP:** Cloud Functions, Cloud Scheduler, Cloud Workflows

Monitoring & Alerting

- **Application Performance:** Application Insights, CloudWatch, Stackdriver
- **Cost Alerting:** Native cloud provider tools + PagerDuty/Slack integration
- **Dashboard:** Grafana, Power BI, CloudWatch Dashboards

Next Steps

After completing this 30-day checklist:

1. **Quarterly Reviews:** Schedule quarterly cost optimization reviews

2. **Advanced Optimization:** Explore advanced techniques like spot instances, reserved capacity
 3. **Cross-Team Sharing:** Share learnings with other development teams
 4. **Tool Evaluation:** Evaluate additional cost optimization tools and services
 5. **Continuous Learning:** Stay updated on new cloud cost optimization features and best practices
-

Support & Resources

- **CloudCostChefs Community:** Join our community for ongoing support and best practice sharing
 - **Documentation:** Access detailed guides and tutorials at cloudcostchefs.com
 - **Tools:** Download additional tools and templates from our resource library
-

Remember: Cost optimization is a journey, not a destination. Small, consistent improvements compound into significant savings over time. Start with the basics and build upon your successes!