Cloud Cost Optimization Playbook

Step-by-Step Strategies to Reduce AWS, Azure, and GCP Costs by 40%

Introduction

Cloud computing offers incredible flexibility and scalability, but without proper management, costs can spiral out of control. Studies show that companies waste 30-40% of their cloud spend on unused resources, over-provisioned instances, and poor architecture choices.

This playbook provides actionable strategies to optimize cloud costs across AWS, Azure, and Google Cloud Platform (GCP), with specific tactics that can reduce your cloud bill by 30-50% within 90 days.

Table of Contents

Section 1: Understanding Your Cloud Bill

The Four Pillars of Cloud Costs

1. Compute (40-50% of typical bill)
- Virtual machines / EC2 instances / Compute Engine
- Containers (ECS, AKS, GKE)
- Serverless functions (Lambda, Azure Functions, Cloud Functions)

2. Storage (20-30%)
- Object storage (S3, Blob Storage, Cloud Storage)
- Block storage (EBS, Managed Disks, Persistent Disks)
- Databases (RDS, CosmosDB, Cloud SQL)

3. Networking (10-20%)
- Data transfer between regions
- Load balancers
- VPNs and Direct Connect

4. Other Services (10-20%)
- Managed services
- Monitoring and logging
- Support plans

First Step: Tag Everything

Before optimizing, you need visibility. Implement tagging strategy:

```
Environment: production / staging / development
Team: engineering / marketing / data
Project: project-name
Cost-Center: department-code
Owner: email@company.com
```

Why it matters: You can't optimize what you can't measure. Tags enable cost allocation and accountability.

Section 2: Quick Wins (0-30 Days)

These strategies require minimal effort and can yield 15-30% savings immediately.

Win #1: Delete Unused Resources

What to look for:
- ✓ Stopped but not terminated EC2/VM instances (still incur storage costs)
- ✓ Unattached EBS volumes / Azure disks (orphaned storage)
- ✓ Old snapshots (especially automated backups)
- ✓ Unused Elastic IPs / Public IPs
- ✓ Idle load balancers
- ✓ Development/test resources left running

How to find them:

AWS:

```
# List stopped EC2 instances
aws ec2 describe-instances --filters "Name=instance-state-name,Values=stopped" --
query "Reservations[*].Instances[*].[InstanceId,Tags[?Key=='Name'].Value|[0]]" --out-
put table

# Find unattached EBS volumes
aws ec2 describe-volumes --filters "Name=status,Values=available" --query "Volumes[*].
[VolumeId,Size,CreateTime]" --output table
```

Azure:

```
# Find stopped VMs
Get-AzVM -Status | Where-Object {$_.PowerState -eq "VM deallocated"}

# Find unattached disks
Get-AzDisk | Where-Object {$_.ManagedBy -eq $null}
```

GCP:

```
# List stopped instances
gcloud compute instances list --filter="status:TERMINATED"

# Find unattached disks
gcloud compute disks list --filter="users:*" --format="table(name,zone,sizeGb,type)"
```

Expected savings: 10-20% of total bill

Win #2: Right-Size Over-Provisioned Instances

Most organizations over-provision compute resources “to be safe.” Reality: 60% of instances use <50% of allocated CPU/memory.

Process:
1. Monitor CPU and memory utilization for 7-14 days
2. Identify instances with <40% average utilization
3. Downsize to smaller instance types
4. Test and monitor

Tools to use:
- AWS: CloudWatch + Cost Explorer Rightsizing Recommendations
- Azure: Azure Advisor
- GCP: Active Assist Recommender

Example:
- Current: AWS m5.2xlarge (8 vCPU, 32GB RAM) = $0.384/hour
- Actual usage: 2-3 vCPU, 8GB RAM
- Right-sized: m5.large (2 vCPU, 8GB RAM) = $0.096/hour
- **Savings: 75% = $2,073/month per instance**

Expected savings: 20-40% on compute costs

Win #3: Implement Auto-Shutdown for Dev/Test

Development and test environments don’t need to run 24/7.

Strategy:
- Shut down automatically at 7pm on weekdays
- Shut down all weekend
- Result: Run 45 hours/week instead of 168 hours/week
- **Savings: 73% on dev/test resources**

Implementation:

AWS Lambda Auto-Shutdown:

```python
import boto3

def lambda_handler(event, context):
    ec2 = boto3.client('ec2')

    # Find instances tagged Environment=development
    instances = ec2.describe_instances(
        Filters=[{'Name': 'tag:Environment', 'Values': ['development']}]
    )

    instance_ids = []
    for reservation in instances['Reservations']:
        for instance in reservation['Instances']:
            instance_ids.append(instance['InstanceId'])

    if instance_ids:
        ec2.stop_instances(InstanceIds=instance_ids)
        return f"Stopped {len(instance_ids)} instances"
```

Schedule with CloudWatch Events:
- Stop: `0 19 * * MON-FRI` (7pm weekdays)
- Start: `0 8 * * MON-FRI` (8am weekdays)

Azure Automation:
Use Azure Automation Runbooks or set schedules in Azure Portal.

GCP:
Use Cloud Scheduler + Cloud Functions with similar logic.

Expected savings: 50-75% on dev/test environments

Win #4: Delete Old Snapshots and Backups

Snapshots accumulate quickly with automated backups, but old ones are rarely needed.

Retention policy recommendation:
- Daily backups: Keep 7 days
- Weekly backups: Keep 4 weeks
- Monthly backups: Keep 12 months
- Delete everything older

AWS Snapshot Cleanup:

```
# Find snapshots older than 90 days
aws ec2 describe-snapshots --owner-ids self --query "Snapshots[?StartTime<='$(date -d
'90 days ago' --iso-8601)'].[SnapshotId,StartTime,Description]" --output table
```

Automate with AWS Data Lifecycle Manager or Lambda function.

Expected savings: 5-15% of storage costs

Win #5: Reserve Capacity for Predictable Workloads

If you have steady-state production workloads, reserved instances offer 40-70% discounts.

Commitment levels:

Commitment	AWS EC2 Discount	Azure VM Discount	GCP CUD Discount
1-year, partial up-front	20-40%	24-40%	25-37%
1-year, all upfront	30-50%	N/A	N/A
3-year, partial up-front	40-60%	38-62%	40-52%
3-year, all upfront	50-70%	N/A	N/A

How to buy:
1. Analyze 3-6 months of usage
2. Identify baseline (minimum consistent usage)
3. Reserve only the baseline (not peaks)
4. Start with 1-year commitments

Important: Only reserve production resources that run 24/7. Don’t reserve dev/test or variable workloads.

Expected savings: 40-70% on reserved resources

Section 3: Medium-Term Optimizations (30-90 Days)

Optimization #1: Move to Spot/Preemptible Instances

What they are: Unused cloud capacity sold at 60-90% discount. Can be interrupted with short notice.

Best for:
- Batch processing
- Data analysis
- CI/CD build servers
- Machine learning training
- Rendering farms
- Development/test environments

Not for:
- Production web servers
- Databases
- Real-time applications

Implementation strategy:
1. Identify interruptible workloads
2. Make applications fault-tolerant (checkpointing)

3. Use spot fleets / preemptible VM groups
4. Mix with on-demand for critical components

AWS Spot Instances:
- 70-90% cheaper than on-demand
- 2-minute termination warning

Azure Spot VMs:
- Up to 90% discount
- Eviction notice provided

GCP Preemptible VMs:
- Up to 80% discount
- Max 24-hour runtime

Expected savings: 60-90% on applicable workloads

Optimization #2: Optimize Storage Classes

Not all data needs premium performance. Use tiered storage:

AWS S3 Storage Classes:
- **S3 Standard:** $0.023/GB - Frequent access
- **S3 Standard-IA:** $0.0125/GB - Infrequent (monthly) access
- **S3 Glacier:** $0.004/GB - Archive (rarely accessed)
- **S3 Glacier Deep Archive:** $0.00099/GB - Long-term archive

Azure Blob Storage Tiers:
- **Hot:** $0.0184/GB - Frequent access
- **Cool:** $0.01/GB - Infrequent access
- **Archive:** $0.002/GB - Rare access

GCP Cloud Storage Classes:
- **Standard:** $0.020/GB - Frequent
- **Nearline:** $0.010/GB - Monthly access
- **Coldline:** $0.004/GB - Quarterly access
- **Archive:** $0.0012/GB - Annual access

Strategy:
1. Audit storage by access patterns
2. Move old data to cheaper tiers
3. Use lifecycle policies for automatic tiering

AWS S3 Lifecycle Policy Example:

```
{
  "Rules": [
    {
      "Id": "Move to IA after 30 days",
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        },
        {
          "Days": 90,
          "StorageClass": "GLACIER"
        }
      ]
    }
  ]
}
```

Expected savings: 50-90% on archived/infrequently accessed data

---

Optimization #3: Reduce Data Transfer Costs

Data transfer (egress) is expensive. Strategies to reduce:

1. Use Content Delivery Networks (CDNs)
- CloudFront (AWS), Azure CDN, Cloud CDN (GCP)
- Cache content closer to users
- Reduce origin data transfer by 70-90%

2. Keep data in same region
- Cross-region transfer: $0.02/GB (AWS)
- Same-region transfer: Free (usually)
- Design architecture to minimize cross-region traffic

3. Compress data
- Enable gzip/brotli compression
- Reduce transfer size by 60-80%

4. Use private connectivity
- AWS Direct Connect, Azure ExpressRoute, GCP Interconnect
- Cheaper than internet transfer for high volume

Expected savings: 30-70% on data transfer costs

Optimization #4: Database Optimization

Databases are often the most expensive component.

Strategies:

1. Right-size database instances
- Monitor CPU, memory, IOPS utilization

- Downsize over-provisioned databases
- Consider burstable instances for dev/test

2. Use read replicas intelligently
- Offload read traffic from primary
- Use smaller instance types for replicas
- Delete unused replicas

3. Reduce storage
- Delete old data
- Archive to cheaper storage (S3, Blob, GCS)
- Use compression

4. Optimize backup retention
- Keep only necessary backups
- Use cross-region backups sparingly

5. Consider serverless options
- Aurora Serverless (AWS)
- Azure SQL Database Serverless
- Cloud SQL (GCP) with auto-scaling

Example:
- Current: RDS db.m5.2xlarge (8 vCPU, 32GB) = $1,642/month
- Optimized: Aurora Serverless (auto-scales) = avg $400-600/month
- **Savings: 60-70%**

Expected savings: 30-50% on database costs

Optimization #5: Implement Cost Allocation and Chargebacks

Why it matters: Teams overspend when they don’t see the bill.

Implementation:
1. **Tag all resources** (Team, Project, Environment)
2. **Create cost allocation reports** by team/project
3. **Share monthly cost reports** with each team
4. **Implement chargebacks** - bill departments for their usage
5. **Set budgets and alerts** per team

Result: When teams see their costs, they optimize. Typical reduction: 20-30% within 3 months.

Tools:
- AWS: Cost Allocation Tags + Cost Explorer
- Azure: Cost Management + Tags
- GCP: Labels + Billing Reports

Section 4: Long-Term Strategy (90+ Days)

Strategy #1: Architecture Optimization

Move to serverless where applicable:

Traditional Architecture:
- ALB + EC2 Auto Scaling Group
- Minimum 2 instances running 24/7
- Cost: $150-300/month baseline

Serverless Alternative:
- API Gateway + Lambda Functions
- Pay only for actual requests
- Cost: $10-50/month for similar traffic

Best candidates for serverless:
- APIs with variable traffic
- Event-driven processing
- Scheduled tasks (cron jobs)
- Image/video processing
- Data pipelines

Expected savings: 60-90% for low-to-medium traffic applications

---

Strategy #2: Use Multi-Cloud Strategically

Cost arbitrage opportunities:

Service	Cheapest Provider	Savings vs. Most Expensive
Object Storage	GCP ($0.020/GB)	15-25% vs AWS
VM Compute	Often Azure	10-20% depending on type
Serverless Functions	GCP (free tier)	Varies significantly
Managed Kubernetes	GCP (free control plane)	$70-140/month vs AWS

Caution: Multi-cloud adds complexity. Only pursue if savings justify operational overhead.

Strategy #3: Negotiate Enterprise Agreements

Once you hit $100K+/year in cloud spend, you have negotiation leverage.

What to negotiate:
- Volume discounts (10-30% additional off)
- Committed use credits (prepay for discount)
- Free support plans

- Free training/certifications
- Waived fees (e.g., data transfer)

How to negotiate:
- Compare quotes from AWS, Azure, GCP
- Work with cloud resellers (can offer better terms)
- Time negotiations for end of quarter/year
- Commit to growth targets for better discounts

Expected savings: 10-30% additional on top of other optimizations

---

Section 5: Platform-Specific Tips

AWS-Specific Optimizations

1. Use AWS Savings Plans
- More flexible than Reserved Instances
- Applies across services (EC2, Fargate, Lambda)
- 1 or 3-year commitment
- Discount: 30-60%

2. Leverage AWS Free Tier
- Always-free services (DynamoDB, Lambda limits)
- CloudFront 1TB free data transfer out/month
- CloudWatch 10 custom metrics and dashboards

3. Use S3 Intelligent-Tiering
- Automatically moves objects between access tiers
- No retrieval fees (unlike Glacier)
- Monitors access patterns

4. Enable Cost Anomaly Detection
- AI-powered alerts for unusual spending
- Catch misconfigurations early
- Free service

5. Use AWS Compute Optimizer
- ML-powered rightsizing recommendations
- Free service

Azure-Specific Optimizations

1. Use Azure Hybrid Benefit
- Bring existing Windows Server licenses
- Up to 85% savings on VMs
- Also applies to SQL Server

2. Leverage Azure Dev/Test Pricing
- Reduced rates for non-production

- Requires Visual Studio subscription
- 40-60% off standard rates

3. Use Azure Reservations
- Similar to AWS Reserved Instances
- Exchange and refund flexibility
- Can be shared across subscriptions

4. Enable Azure Advisor
- Free recommendations
- Rightsizing, unused resources, best practices

5. Use Azure Cost Management
- Set budgets and alerts
- Cost analysis by resource group
- Free for Azure customers

GCP-Specific Optimizations

1. Use Sustained Use Discounts
- Automatic discounts for consistent usage
- No commitment needed
- Up to 30% off

2. Leverage Free Tier
- Always-free products (Cloud Functions, Cloud Run limits)
- GKE free cluster management
- Monthly credits for VMs

3. Use Committed Use Discounts (CUDs)
- 1 or 3-year commitments
- 57% discount for 3-year
- Flexible across instance types

4. Active Assist Recommender
- AI-powered optimization suggestions
- Idle resources, rightsizing, committed use
- Free service

5. BigQuery Flat-Rate Pricing
- If running heavy analytics
- Switch from on-demand to flat-rate
- Can save 50%+ for high-volume users

Section 6: Cost Optimization Tools

Cloud-Native Tools (Free)

AWS:
- Cost Explorer - Visualize and analyze costs

- Trusted Advisor - Best practice recommendations
- Compute Optimizer - ML-powered rightsizing
- Cost Anomaly Detection - Alert on unusual spending

Azure:
- Cost Management + Billing - Comprehensive cost analysis
- Azure Advisor - Optimization recommendations
- Azure Monitor - Resource utilization tracking

GCP:
- Cloud Billing Reports - Cost visibility
- Recommender - Optimization suggestions
- Cloud Monitoring - Performance and usage metrics

Third-Party Tools (Paid, but worth it at scale)

Multi-Cloud Management:
- **CloudHealth (VMware)** - Enterprise-grade, all clouds
- **Cloudability (Apptio)** - Cost optimization platform
- **Spot.io** - Automated spot instance management
- **Cast.ai** - Kubernetes cost optimization

Specialized Tools:
- **Kubecost** - Kubernetes cost allocation
- **Harness** - Cloud cost management + FinOps
- **CloudForecast** - AWS cost reduction automation
- **ProsperOps** - Autonomous AWS discount management

When to invest in tools:
- $50K+/month cloud spend
- Multiple teams/projects
- Limited dedicated FinOps resources

Section 7: Governance & Ongoing Management

Establish FinOps Practice

FinOps = Financial Operations for Cloud

Key roles:
- **FinOps Lead:** Drives cost optimization culture
- **Engineering:** Implements optimizations
- **Finance:** Budgeting and forecasting
- **Leadership:** Sets cost efficiency goals

Monthly FinOps Meeting Agenda:
1. Review previous month’s spend vs. budget
2. Discuss cost anomalies
3. Review optimization recommendations
4. Assign action items
5. Set targets for next month

Implement Cost Controls

1. Budget alerts
- Set alerts at 50%, 80%, 100% of budget
- Alert specific teams when their budgets hit thresholds

2. Spending limits (where supported)
- Hard caps on non-production accounts
- Prevent runaway costs

3. Approval workflows
- Require approval for expensive instance types
- Review before launching multi-region resources

4. Automated enforcement
- Auto-shutdown policies
- Prevent deployment without proper tags
- Block expensive services in dev/test

Measure Success

Key Metrics to Track:

1. **Total Cloud Spend** - Month-over-month trend
2. **Cost per Customer** - Unit economics
3. **Cost per Transaction** - Efficiency metric
4. **Waste Percentage** - Unused/idle resources
5. **Optimization Rate** - % of recommendations implemented

Goal: Reduce cloud spend by 30-50% in first year, then maintain 10-15% YoY optimization as you scale.

90-Day Action Plan

Days 1-30: Quick Wins

- [ ] Implement tagging strategy
- [ ] Delete unused resources (snapshots, volumes, IPs)
- [ ] Set up auto-shutdown for dev/test
- [ ] Purchase reserved instances for baseline workloads
- [ ] Enable cost anomaly detection
- **Target: 15-25% savings**

Days 31-60: Medium Optimizations

- [ ] Right-size over-provisioned instances
- [ ] Implement storage tiering
- [ ] Optimize databases
- [ ] Set up cost allocation reports
- [ ] Reduce data transfer with CDN

- **Target: Additional 10-20% savings**

Days 61-90: Strategic Changes

- [ ] Migrate workloads to spot instances
- [ ] Implement serverless where applicable
- [ ] Establish FinOps practice
- [ ] Set up governance and policies
- [ ] Plan architecture improvements
- **Target: Additional 5-15% savings**

Total 90-Day Target: 30-50% cost reduction

Real-World Case Studies

Case Study 1: SaaS Startup

Before: $25K/month AWS spend
Actions:
- Right-sized RDS databases (40% reduction)
- Moved dev/test to spot instances (70% reduction on those resources)
- Implemented S3 lifecycle policies (50% storage cost reduction)
- Reserved production EC2 instances (60% discount)

After: $11K/month
Savings: 56% ($168K/year)

Case Study 2: E-commerce Company

Before: $180K/month multi-cloud spend
Actions:
- Deleted 2TB of old snapshots and logs
- Moved static assets to CloudFront CDN (reduced origin transfer by 85%)
- Implemented auto-scaling for variable traffic (reduced over-provisioning)
- Negotiated Enterprise Agreement with AWS

After: $98K/month
Savings: 46% ($984K/year)

Case Study 3: Enterprise SaaS

Before: $450K/month AWS spend
Actions:
- Established FinOps team with dedicated lead
- Implemented cost allocation and chargebacks to product teams
- Optimized Kubernetes cluster with Kubecost (35% reduction)
- Migrated batch workloads to Lambda (90% cost reduction on those workloads)
- Purchased 3-year Savings Plans for baseline usage

After: $245K/month
Savings: 46% ($2.46M/year)

Common Mistakes to Avoid

❌ **Over-optimizing variable workloads**
- Don't reserve capacity for unpredictable usage
- Use on-demand or spot for variable loads

❌ **Ignoring hidden costs**
- Data transfer is often overlooked
- Snapshots and backups accumulate silently

❌ **Optimizing without monitoring**
- Always monitor for 7-14 days before changes
- Validate performance after optimization

❌ **Cutting too aggressively**
- Maintain performance and reliability
- Some "waste" is acceptable for resilience

❌ **Not involving engineering early**
- Optimizations require technical implementation
- Engineering buy-in is critical

Conclusion

Cloud cost optimization is not a one-time project—it's an ongoing practice. By implementing the strategies in this playbook, you can:

✓ Reduce cloud costs by 30-50% in 90 days
✓ Establish sustainable FinOps practices
✓ Align cloud spend with business value
✓ Scale efficiently as you grow

Remember: The goal isn't zero cloud spend—it's optimal spend. Invest in what drives business value, eliminate what doesn't.

Next Steps

1. **Audit your current cloud spend** using native tools
2. **Implement quick wins** from Section 2 this week
3. **Schedule monthly cost reviews** with your team
4. **Track progress** against the 90-day action plan
5. **Celebrate wins** and share savings across the organization

About Cancel Costs

Cancel Costs specializes in cloud cost optimization and FinOps implementation. Our team has helped companies save millions in cloud infrastructure costs while improving performance and reliability.

Need help optimizing your cloud costs?
Book a free assessment: https://cancelcosts.com/contact
