NYCPS TMS: Prescriptive Cloud Cost Management & FinOps Strategy

I. Introduction: The Financial Imperative in the Cloud

This document mandates the comprehensive, hyper-detailed Cloud Cost Management and Financial Operations (FinOps) strategy for the NYCPS Transportation Management System (TMS) project. Operating a large-scale, data-intensive system on AWS GovCloud requires a disciplined, proactive approach to managing expenditures to ensure fiscal responsibility, maximize the value derived from cloud investments, maintain budget predictability, and meet the stringent accountability standards expected in a public sector project.

This strategy integrates financial governance, cost visibility, accurate forecasting, continuous optimization, and transparent reporting into the daily operations and

development lifecycle. It explicitly expands upon the overall project Financial Management Plan, providing granular detail on *how* cloud-specific costs will be meticulously tracked, analyzed, controlled, and optimized.

Core Objective: To instill a culture of cost awareness and accountability across all teams interacting with cloud resources, enabling data-driven decisions that balance performance, security, compliance, and cost-effectiveness throughout the TMS lifecycle.

Core FinOps Principles (Mandatory Adherence):

- **Visibility is Foundational:** We cannot manage what we cannot see. Implement comprehensive cost visibility and allocation mechanisms.**
- **Accountability Drives Action:** Teams responsible for building and running services are accountable for their cloud consumption and empowered to optimize it.**
- ****Collaboration is Key:** FinOps requires collaboration between Finance, Engineering (Dev, Ops, SRE), Security, and Project Management/Leadership.**
- ****Decisions are Data-Driven:** Use accurate cost and usage data to inform architectural choices, optimization efforts, and budget forecasts.**
- ****Cloud Native Optimization:** Leverage cloud pricing models (RIs, SPs), elasticity, managed services, and specific AWS cost-saving features proactively.**
- ****Proactive Governance:** Establish automated guardrails and review processes to prevent cost**

anomalies and enforce policies.

- **Continuous Optimization:** Cost management is an ongoing process, not a one-time task. Regularly review and refine resource usage and purchasing strategies.

II. FinOps Governance & Roles

Effective FinOps requires dedicated focus and clearly defined responsibilities.

Implementation How-To:

1. Establish a **Cloud FinOps Lead/Analyst** role (can be a dedicated individual or a primary function within the SRE/Ops or PM team, requiring ~0.5-1 FTE focus).
2. Define specific FinOps responsibilities for existing roles:
    - **Cloud FinOps Lead/Analyst:** Owns the FinOps strategy execution, performs deep cost analysis (CUR/Cost Explorer), identifies/validates optimization

opportunities, prepares recommendations for RIs/SPs/right-sizing, develops/maintains cost dashboards/reports, configures AWS Budgets, promotes cost awareness, liaises with Finance/PMs.

- **NYCPS PM / Finance Liaison:** **Integrates cloud cost forecasts/actuals into overall project budget, approves cost-related CRs within delegated authority, presents financial status to Steering Committee, ensures compliance with NYCPS financial policies.**
- **Cloud Architect / Tech Leads:** **Incorporate cost considerations into architectural designs, evaluate cost-effectiveness of different AWS service options, review cost impact of proposed technical changes.**
- **SRE/Ops Team:** **Implement right-sizing changes based on**

analysis, implement non-prod shutdown automation, monitor resource utilization, respond to cost anomaly alerts.

- **Development Teams:** Develop cost-efficient code (e.g., optimizing Lambda memory/duration, efficient database queries), implement tagging correctly on resources defined in IaC, participate in right-sizing reviews for their services.
- **Steering Committee / Project Sponsor:** Approves overall cloud budget, approves significant RI/SP commitments or major cost optimization initiatives with trade-offs.

3. Establish a **Monthly FinOps Review Meeting:** Attended by FinOps Lead, PMs, Finance Liaison, SRE/Ops Lead, key Tech Leads. Agenda: Review previous month's spend vs budget/forecast, analyze major cost drivers/variances, review optimization progress/opportunities, plan upcoming cost impacts.

4. **Integrate FinOps updates as a standing item in MBR/Steering Committee meetings.**

Responsibility: Project Leadership (Establishing Role), FinOps Lead (Executing Strategy), All Leads/Teams (Participating).

III. Cost Visibility & Granular Allocation Strategy

Achieving granular visibility into cloud spending is the prerequisite for effective management and accountability.

A. Mandatory Tagging Strategy & Enforcement

Implementation How-To:

1. **Define a **Mandatory Tagging Policy** documented in Confluence, requiring specific tags on *all* taggable AWS resources provisioned via Terraform.**

2. **Mandatory Tags:**
    - `environment`: (e.g., `dev`, `qa`, `staging`, `perf`, `prod`, `mgmt`)
    - `project`: `nycps-tms` (Consistent identifier)
    - `service-component`: (Specific microservice or logical component name, e.g., `gps-ingestion`, `routing-engine`, `parent-api`, `data-lake-raw`)
    - `owner-team`: (Team responsible, e.g., `apollo-dev`, `platform-sre`, `data-eng`)
    - `cost-center`: (Link to NYCPS financial cost center, provided by Finance Liaison - essential for potential showback/chargeback)
    - *(Optional but Recommended)* `feature-id`: (Link to Jira Epic/Feature for new initiatives)
    - *(Optional)* `created-by`: (`terraform` or user/pipeline identifier)

3. **Implementation via Terraform:**
    - Configure default tags in the AWS provider block within each environment's `main.tf` (e.g., setting `environment`, `project`).
    - Mandate passing `service-component`, `owner-team`, `cost-center` variables into reusable modules and applying them to all resources created by the module.
    - Conduct code reviews on Terraform MRs specifically checking for correct tag application.
4. **Enforcement & Governance:**
    - Implement **AWS Config** custom rules (deployed via Terraform) to detect resources missing mandatory tags or having non-compliant tag values.
    - Configure Config rules to trigger non-compliance notifications (SNS -> Slack/Email) or potentially auto-remediation actions (e.g., stopping

untagged non-prod instances after a grace period - use with caution).

- **Periodically review untagged resources report in AWS Tag Editor / Cost Explorer.**
- **Establish a process (via Jira/Service Request) for requesting and approving new tag values (e.g., new cost centers, new service components).**

5. ****Activate Tags for Cost Allocation:** In the AWS Billing console, explicitly activate the user-defined tags (`environment`, `service-component`, `owner-team`, `cost-center`) as **Cost Allocation Tags**. This makes them available for filtering and grouping in Cost Explorer and CUR. Allow ~24 hours for activation.**

Responsibility: Cloud Architect/FinOps Lead (Defining Policy), DevOps Team (Implementing in IaC/Config Rules), Developers (Applying in module usage), Security Team (Auditing Config Rules).

Consistent and accurate tagging is absolutely mandatory for cost allocation, optimization analysis, and demonstrating fiscal responsibility.

B. Cost & Usage Report (CUR) Configuration & Analysis

Implementation How-To:

1. **Configure CUR in the AWS Billing console (ideally from the AWS Organizations management account).**
2. ****Settings:****
    - **Include Resource IDs.**
    - **Include Split Cost Allocation Data (for shared resources like data transfer, support).**
    - **Data granularity: **Hourly**.**
    - **Format: **Parquet** (for efficient querying with Athena).**
    - **Compression: GZIP or ZSTD.**
    - **Delivery: Target a dedicated, secure S3 bucket (in the logging/audit account preferred) with appropriate lifecycle policies (e.g., retain CUR files for 1-3 years).**

- **Integration: Enable integration with **Amazon Athena** and **Amazon QuickSight**.**

3. **Athena Setup:** Use AWS Glue Crawler to automatically catalog the CUR data structure in the S3 bucket. Create standard Athena views simplifying queries (e.g., flattening tags, joining with pricing data).

4. **Analysis:** Use Athena SQL queries (run manually, via scheduled Lambdas, or QuickSight) to perform granular cost analysis based on tags, resource IDs, usage types, time periods. Identify cost anomalies, track specific resource spend, allocate shared costs.

Tools: AWS Billing Console (CUR Setup), S3, AWS Glue Crawler, Amazon Athena, SQL.

Responsibility: FinOps Lead/Analyst, Cloud Ops (S3/Glue Setup).

C. AWS Cost Explorer Utilization

Implementation How-To:

1. **Utilize Cost Explorer regularly (daily/weekly checks by FinOps Lead, monthly reviews by PMs/Leads)**

for:

- ****Visualization:** Trend analysis (daily, monthly costs), cost breakdown by service, linked account, region, instance type, and activated **Cost Allocation Tags**.**
- ****Filtering & Grouping:** Filter by specific tags (`environment`, `service-component`, `cost-center`), services, accounts, time ranges to isolate costs.**
- ****Forecasting:** Use built-in forecasting capabilities (based on historical usage) as input for manual project financial forecasting.**
- ****Rightsizing Recommendations:** Regularly review EC2 instance rightsizing recommendations based on performance metrics (requires CloudWatch agent data). Evaluate recommendations carefully before acting.**

- **RI/SP Recommendations:** Review Reserved Instance and Savings Plans purchase recommendations based on historical usage patterns. Use as input for RI/SP analysis.

2. Save common views/reports within Cost Explorer for quick access.
3. Provide read-only Cost Explorer access (via IAM policies) to relevant PMs/Leads to foster cost awareness.

Tools: AWS Cost Explorer.

Responsibility: FinOps Lead/Analyst (Primary User), PMs/Leads (Reviewers).

D. Showback / Chargeback Model (Optional - Requires NYCPS Finance Alignment)

Implementation How-To (If Implemented):

1. Requires mature and consistently enforced tagging, especially the `cost-center` tag mapped accurately to NYCPS financial structures.

2. **Develop automated reports (using Athena queries on CUR data, visualized in QuickSight or exported) that group actual cloud spend by the `cost-center` tag.**

3. ****Showback:** Distribute these reports regularly (e.g., monthly) to the owners of each cost center to provide visibility into the cloud resources consumed by their department/function related to TMS.**

4. ****Chargeback:** If formal internal billing is required by NYCPS finance, use the generated showback reports as the basis for internal cost allocation according to established NYCPS financial procedures. This requires close collaboration with the NYCPS Finance Liaison.**

5. **Clearly define how shared infrastructure costs (e.g., network backbone, shared services, security tools) are allocated across cost centers (e.g., based on usage proportion, fixed percentage).**

Responsibility: FinOps Lead/Analyst (Reporting), NYCPS Finance Liaison (Policy/Process), Steering Committee (Approval of model).

> ***Implementing chargeback requires strong executive sponsorship and alignment with central NYCPS financial processes. Showback is often a valuable first step.***

V. Cloud Budgeting & Forecasting Integration

Cloud cost planning must be integrated with the overall project financial management.

Implementation How-To:

1. **Baseline Budgeting:** During initial project budgeting (see Financial Plan), estimate cloud costs per phase based on planned architecture, expected usage, data volumes, and AWS GovCloud pricing. Include separate lines for Prod vs. Non-Prod environments and major services (Compute, Storage, DB, Data Transfer). Incorporate tagging strategy into budget lines.
2. **AWS Budgets Configuration (Mandatory):**
    - Create AWS Budgets reflecting the baseline figures (monthly/quarterly).
    - Set multiple **alert thresholds** based on *both actual and forecasted* spend (e.g., 50%, 80%, 100%, 120% of budget).

- **Configure alerts to notify relevant stakeholders via **SNS** (-> Email/Slack/PagerDuty). At minimum, notify FinOps Lead, PMs, SRE/Ops Lead. Consider alerting specific Tech Leads for budgets tied to their services/tags.**
- **Use Cost Allocation Tags to create budgets for specific environments (Non-Prod!), services, or cost centers.**

> ***AWS Budgets provide crucial automated early warning for potential overruns.***

3. **Monthly Forecasting Input:** The Cloud Ops/FinOps Lead provides detailed cloud cost forecasts as input to the overall project EAC calculation (see Financial Plan). This forecast is based on:
    - **Current run rate (from Cost Explorer).**
    - **Cost Explorer's built-in forecast.**
    - **Adjustments for planned infrastructure changes (new service**

deployments, scaling events, decommissioning - derived from project plan/IaC changes).

- Anticipated impact of upcoming RI/SP purchases or optimization initiatives.
- Seasonality or known usage peaks (e.g., start of school year).

4. **Variance Analysis Integration:** Cloud cost variances identified via AWS Budgets or monthly reconciliation *must* feed into the overall project Variance Analysis process (Financial Plan, Section VI). Root cause analysis should pinpoint specific services, tags, or events driving the variance.

Responsibility: FinOps Lead/Analyst (AWS Budgets Config, Forecasting Input, Variance Analysis), PMs (Integrating into Overall Budget/Forecast), Tech Leads (Input on future usage).

VI. Continuous Cost Optimization Lifecycle

Optimization is an ongoing, iterative process involving multiple techniques applied systematically.

A. Right-Sizing Compute & Database Resources

Process (Mandatory Monthly/Quarterly Review Cadence):

1. **Identify Candidates:** Use **AWS Cost Explorer Rightsizing Recommendations** (based on historical CloudWatch metrics) and **AWS Compute Optimizer** recommendations as primary inputs. Supplement with detailed CloudWatch metric analysis (CPU/Memory Utilization - Avg vs. Max) for key EC2, RDS, ECS/Fargate tasks, Lambda functions over a significant period (e.g., 14-30 days). Pay attention to resources with consistently low average utilization but occasional peaks.
2. **Analyze & Validate:** SRE/Ops/Dev teams review recommendations. **Do not blindly apply.** Consider workload characteristics (spiky vs. steady), memory vs. CPU constraints, burst requirements, and potential impact of downsizing

on peak performance or SLOs. Use APM data (if available) for deeper application-level insights.

3. **Select Instance Family/Size:** Choose appropriate instance families (consider Graviton/ARM instances for better price/performance where compatible) and sizes based on analysis. For containers/Lambda, adjust CPU/Memory reservations/limits.

4. **Implement via IaC:** Modify instance types, task definitions, or Lambda configurations in Terraform code.

5. **Test:** Deploy changes first to non-production environments (QA/Staging). Conduct targeted performance tests and monitor application health after the change to validate no negative impact.

6. **Deploy to Production:** Roll out validated changes to production during a planned maintenance window using CI/CD pipeline (with necessary approvals).

7. **Monitor Post-Change:** Closely monitor performance and cost metrics after the change to confirm expected savings and ensure no performance degradation.

Responsibility: SRE/Ops Team (Analysis/Implementation), FinOps Lead (Tracking Savings), Dev Teams (Application Impact Assessment), QA (Testing).

B. Utilizing Purchase Models (Reserved Instances - RIs / Savings Plans - SPs)

Process (Mandatory Quarterly Analysis Cadence):

1. **Analyze Usage:** Use **AWS Cost Explorer RI/SP Recommendations** and detailed CUR analysis to identify stable, long-term usage patterns for EC2, Fargate, Lambda, RDS, Redshift, OpenSearch, SageMaker that could benefit from commitment discounts.
2. **Evaluate Recommendations:** Analyze Cost Explorer's recommended purchases (service, instance family, region, term - 1yr vs 3yr, payment option - All/Partial/No Upfront). Consider flexibility needs (e.g., Convertible RIs or Compute Savings Plans offer more flexibility than Standard RIs). Model potential savings vs. commitment lock-in risk. Aim for a target coverage percentage (e.g., 70-90%) of stable baseline usage.

3. **Develop Purchase Proposal:** FinOps Lead/Analyst prepares a detailed proposal outlining recommended RI/SP purchases, projected savings, commitment term, upfront cost (if any), and associated risks.

4. **Obtain Approval:** Present proposal to project leadership and **NYCPS Finance/Steering Committee** for formal approval, especially for significant upfront payments or long-term commitments.

5. **Execute Purchase:** Once approved, execute the purchase via AWS Billing Console or API.

6. **Track Utilization & Coverage:** Continuously monitor RI/SP utilization and coverage rates using Cost Explorer RI/SP reports. Identify underutilized reservations and explore modifications/exchanges where possible (especially for Convertible RIs). Set alerts for expiring commitments to plan renewals.

Responsibility: FinOps Lead/Analyst (Analysis/Proposal), Project Leadership/Finance/Steering Committee (Approval), Cloud Ops (Execution/Tracking).

> ***RI/SP purchases involve financial commitments and require formal financial approval beyond the project team.***

C. Storage Optimization (Tiering, Cleanup)

Process (Mandatory Quarterly Review Cadence + Automation):

1. **S3 Tiering:** Analyze object access patterns using **S3 Storage Lens** and S3 Analytics Storage Class Analysis. Implement/tune **S3 Intelligent-Tiering** configurations for buckets with unpredictable access patterns. Implement/verify **S3 Lifecycle Policies** (via Terraform) to automatically transition data to lower-cost tiers (Standard-IA -> Glacier Instant Retrieval -> Glacier Flexible Retrieval -> Glacier Deep Archive) based on age and access frequency, aligning with the 7-year retention requirement.
2. **EBS Volume Optimization:** Review EBS volume types (gp2 vs gp3 - gp3 often cheaper and more performant), resize underutilized volumes (identified via CloudWatch metrics), and delete unattached volumes or old snapshots. Use **AWS Backup lifecycle policies** or custom scripts to automatically delete EBS snapshots older than the required retention period.
3. **Database Storage:** Monitor RDS/DynamoDB storage growth. Archive historical data from active

databases to S3 (as per Data Archival plan) to manage primary storage costs and performance.

4. **Log Retention:** Ensure CloudWatch Log Groups and other log storage mechanisms have appropriate retention periods configured (via Terraform) to avoid indefinite storage costs, balancing operational needs with compliance requirements.

Responsibility: SRE/Ops Team, DevOps Team (IaC), Data Engineers (DB Archiving), FinOps Lead (Analysis).

> ***S3 Intelligent-Tiering and Lifecycle Policies are key automation tools for storage cost optimization.***

D. Scheduling Non-Production Resources (Automation Mandatory)

Process (Implement Once, Monitor Regularly):

1. **Identify Targets:** Tag all non-production resources (EC2, RDS, potentially Fargate services, SageMaker Notebooks) eligible for shutdown during non-business hours (nights, weekends). Use a specific tag like `schedule: office-hours`.

2. **Implement Automation:** Deploy and configure the **AWS Instance Scheduler** solution OR develop custom Lambda functions triggered by **EventBridge Scheduler** rules.

3. **Configuration:** Define schedules (e.g., Stop at 8 PM EST Mon-Fri, Start at 7 AM EST Mon-Fri). Configure the automation to target resources based on the `schedule` tag. Ensure RDS instances are stopped correctly (which automatically creates a final snapshot).

4. **Exclusions:** Provide a mechanism (e.g., another tag `schedule-exempt: true`) to exclude specific non-prod resources needed for overnight batch jobs or testing.

5. **Monitoring:** Monitor the execution of the scheduler jobs/Lambdas. Set alerts if shutdown/startup fails. Periodically review if resources are being correctly stopped/started.

Responsibility: DevOps/Cloud Ops Team (Implementation/Monitoring).

Ensure schedules accommodate globally distributed team working hours if applicable. Incorrectly stopped resources can disrupt development/testing.

E. Architectural & Development Optimization

Process (Ongoing Consideration):

1. **Design Phase:** During architecture/design reviews (TRB), explicitly consider the cost implications of different AWS service choices (e.g., Lambda vs. Fargate cost model, Serverless vs. Provisioned databases, data transfer patterns).
2. **Development Phase:** Encourage developers to write efficient code (optimize algorithms, minimize unnecessary processing/API calls). Choose appropriate Lambda memory sizes (use AWS Lambda Power Tuning tool). Optimize database queries.
3. **Technology Selection:** Favor cost-effective compute options like **AWS Graviton (ARM)** instances/Lambda where performance and compatibility allow. Evaluate serverless options (Lambda, Fargate Serverless, Aurora Serverless, DynamoDB On-Demand) where workloads are intermittent or unpredictable.

Responsibility: Cloud Architect, Tech Leads, Developers.

VIII. FinOps Reporting & Communication

Transparently communicating cost information and optimization progress is vital for accountability and continuous improvement.

Implementation How-To:

1. **FinOps Dashboards (QuickSight/CloudWatch):** Create role-specific dashboards:
    - **Executive Dashboard:** High-level spend vs. budget, forecast trends, key optimization savings, RI/SP coverage.
    - **PM/Lead Dashboard:** Spend breakdown by environment/service/component (using tags), variance analysis details, budget alerts status, optimization tracker.
    - **Dev Team Dashboard:** Costs associated with their specific services/components (via tags), non-prod environment costs,

potential optimization recommendations relevant to them.

2. **Monthly FinOps Report:** FinOps Lead prepares detailed report summarizing spend, forecast, variance analysis, optimization activities performed, savings realized, upcoming cost drivers, and recommendations. Input for MBR. Stored in Confluence.

3. **Integration into Project Reporting:** Include key FinOps metrics and status updates in Weekly Status Reports and Monthly Business Reviews (as defined in Comms Plan).

4. **Alert Notifications:** Ensure AWS Budget alerts are routed via SNS to designated Slack channels/email lists for immediate awareness among PMs, FinOps, and Ops leads.

5. **Cost Awareness Culture:** Regularly share cost information, optimization successes, and best practices during team meetings or internal tech talks to foster cost consciousness across all engineering roles.

Responsibility: FinOps Lead/Analyst (Dashboards/Reports), PMs (Integrating into project reporting), Comms Lead (Broader communication).

IX. Conclusion: Embedding Fiscal Responsibility in Cloud Operations

This Cloud Cost Management and FinOps Strategy provides the essential framework for ensuring fiscal discipline and maximizing the value of the significant AWS GovCloud investment required for the NYCPS TMS project. By establishing clear governance, mandating granular visibility through rigorous tagging and CUR analysis, implementing proactive budgeting and forecasting integrated with AWS tools, executing a continuous optimization lifecycle across multiple dimensions (right-sizing, purchase models, storage, scheduling), and fostering a culture of cost accountability through transparent reporting and communication, we embed financial responsibility directly into our engineering and operational practices.

Strict adherence to this prescriptive plan, particularly the emphasis on automation (tagging enforcement, non-prod scheduling, budget alerts, data collection) and data-driven decision-making, is critical for maintaining budget control, justifying expenditures, achieving cost efficiencies, and ensuring the long-term financial sustainability of this vital system for NYCPS.