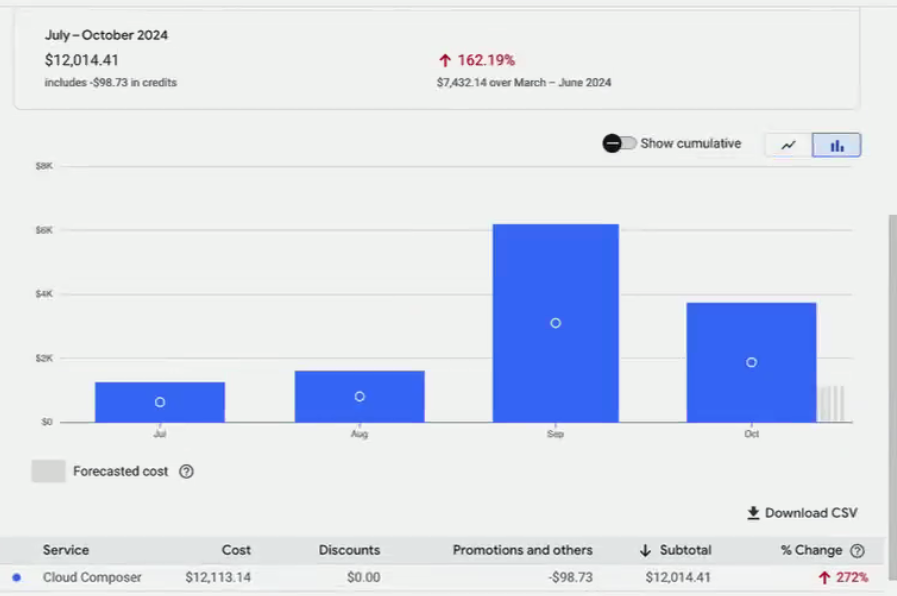
### **GCP Cost Optimization Report**

#### 

#### **Overview**

This report covers cost optimization strategies for **Google Cloud Composer2**, **Compute Engine**, **Storage Buckets**, and **Cloud Logging** resources. It highlights ongoing efforts to reduce costs and streamline cloud operations by focusing on resource provisioning, object deletion, and log filtering.

### **1. Composer2 Cost Optimization**



#### 

#### **Current Status:**

* **Load Testing**: Ongoing load testing for Composer2 in the UAT environment started in mid-September and is expected to continue until the end of October. This involves running 150 to 500 sessions daily.
* **Production Plans**: The same load testing patterns are expected to continue when Composer2 moves to the production environment after October.

#### **Key Findings:**

* **Resource Consumption**: The resource provisioning currently used in Composer2 UAT is likely over-provisioned and may not be fully optimized for the production environment.
* **Machine Configuration**: Optimization opportunities exist at the machine configuration level, which will directly affect the efficiency and cost when moving Composer2 to production.

#### **Recommendations:**

* **Collaborate for Optimization**: Schedule a meeting with Liang to ensure Composer2 is provisioned with the right machine configuration based on actual workload requirements. This will minimize over-provisioning and ensure cost-efficiency.

What we can recommend to the team:

1. **Choose Smaller Machine Types**: Select machine types that are appropriate for your workload. Composer2 allows you to specify the number of vCPUs and memory for the Airflow workers, schedulers, and web server.
2. **Autoscaling Workers**: Enable autoscaling of Airflow workers to automatically adjust the number of workers based on the actual workload, which prevents over-provisioning of resources.
3. **Control Scheduler Resources**: Ensure that the Airflow scheduler is right-sized and not overprovisioned. Adjust its CPU and memory based on your DAG processing needs.
4. **Optimize DAG Scheduling**: Tune the scheduling of DAGs to avoid unnecessary execution of tasks. DAGs that are not time-sensitive can be scheduled less frequently.
5. **Use External Triggers**: Instead of scheduling DAGs at fixed intervals, consider using external triggers or event-based scheduling to execute tasks only when necessary.
6. **Task Parallelization**: Improve parallelism and task execution efficiency within DAGs to reduce the overall runtime, which can reduce resource consumption.
7. **Use Lower-tier Cloud SQL Instances**: The Cloud SQL instance associated with Composer can be downsized to smaller machine types if your workload doesn’t require high performance.
8. **Limit Retention of Logs and Metrics**: Retaining large volumes of logs or metrics can increase storage and Cloud SQL costs. Regularly clean up old logs and tune Airflow's log\_retention and metrics\_retention settings.
9. **Minimize Data Transfers**: Composer environments that interact with other Google Cloud services or external APIs can incur networking costs, especially for cross-region transfers. Wherever possible, keep all your services within the same region to reduce egress costs.
10. **Scale Efficiently**: As Composer2 transitions to production, review the expected load and adjust the scale of resources accordingly. Avoid reserving resources for peak loads unless absolutely necessary to further reduce costs.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

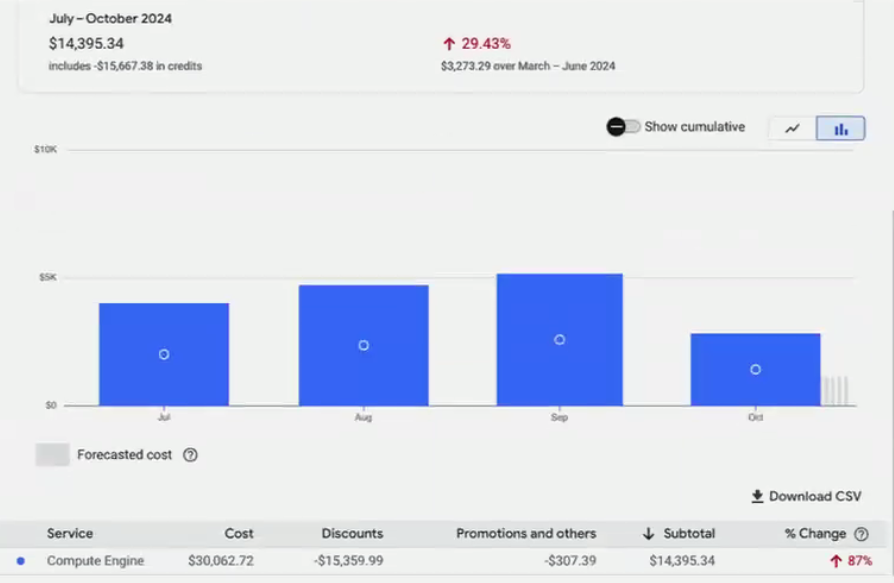
### 

### 

### 

### 

### **2. Compute Engine Cost Optimization**



#### **Current Status:**

* **Idle VMs**: Several VMs have been identified as idle and no longer in use, representing unnecessary cost.
* **Underutilized VMs**: Some VMs are not handling high-load tasks and are running on over-provisioned configurations, increasing costs without justified usage.
* **GKE Cluster Nodes**: A majority of the identified VMs are GKE cluster nodes that require further investigation to understand their configuration and usage patterns.

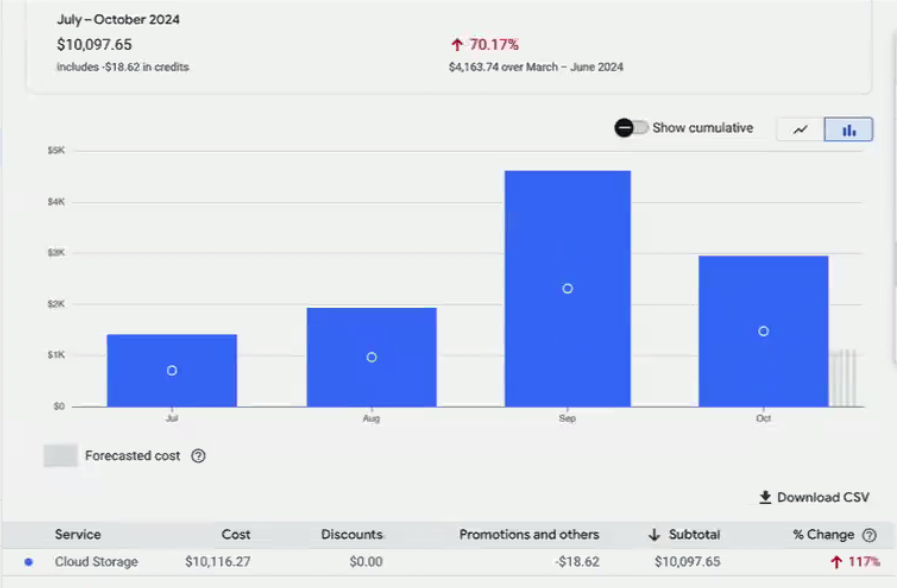
#### **Key Findings:**

* **Idle Resources**: Unused VMs contribute to avoidable costs and should be decommissioned.
* **Over-Provisioned Resources**: Over-provisioning on low-load VMs creates inefficiency in the cloud spend.

#### **Recommendations:**

* **Remove Idle VMs**: Decommission the idle VMs to stop incurring costs for unused resources.
* **Downgrade Underutilized VMs**: For VMs handling low-load tasks, downgrade their configurations to match the current workload. This will reduce the overall compute costs.
* **GKE Cluster Review**: Investigate the configuration of GKE cluster nodes further. This will ensure the clusters are using appropriate VM types and sizes based on workload requirements, preventing overspending on compute resources.

### **3. Storage Bucket Cost Optimization**



#### **Current Status:**

* **Object Deletion**: There are 4 buckets in the UAT environment, and a Python script has been created to automate the object deletion process for cost savings.

#### **Python Script Details:**

1. **Keyword Detection**: The script identifies specific folders in GCP buckets based on provided keywords.
2. **Date Filtering**: It filters folders older than 7 days.
3. **Deletion Process**: Deletes the filtered folders.
4. **Logging**: Logs the deleted objects, including their creation time.

#### **Progress:**

* **Completed**: Object deletion process for 1 bucket.
* **Ongoing**: Object deletion for 2 more buckets, currently running on the GCPOPS server.

#### **Recommendations:**

* **Continue Automation**: Complete the deletion process for the remaining buckets to achieve full cost savings.
* **Review Bucket Retention Policies**: Ensure appropriate lifecycle policies are in place to automatically delete or archive old objects, reducing manual intervention in the future.

### **4. Cloud Logger Optimization**

#### **Current Status:**

* **High Volume of Error Logs**: Over 8 million error logs are being generated daily in the production environment, significantly impacting costs and operational efficiency.

#### **Action Plan:**

* **Log Filtering**: Implement filters to separate useful logs from unimportant ones to reduce the volume of logs retained.
* **Identifying the Root Cause**: We have Identified, OpenCV-python is the package which is responsible for the majority of unnecessary error logs.
* **Changing logging level**: Now, we have started changing the logging level configuration for OpenCV package which will directly impact the error log generation in the cloud logger.
* **Migration to Datadog**: Identify essential logs that are valuable for monitoring and performance analysis, and migrate them to Datadog for dashboard integration.
* **Cost Control**: Reducing the number of stored and processed logs will help lower Cloud Logging expenses and improve log management efficiency.

#### **Recommendations:**

* **Analyze Error Logs**: Investigate the root causes of the high log volume and address any recurring issues generating unnecessary logs.
* **Set Up Log-Based Alerts**: Configure alerts for critical errors only, avoiding storage and alerting for less important logs.
* **Integrate with Datadog**: Ensure that only critical logs are migrated to Datadog to monitor application performance and incidents in a cost-effective manner.

### **Next Steps:**

1. **Composer2**:
   * Work with Liang to refine machine configuration for Composer2’s production environment.
   * Perform a cost analysis of different machine types and configurations to find an optimal setup for production.
2. **Compute Engine**:
   * Initiate the decommissioning process for idle VMs.
   * Evaluate the workload of underutilized VMs and adjust their configurations.
   * Perform a deeper analysis of the GKE cluster nodes to identify optimization opportunities.
3. **Storage Buckets**:
   * Complete the object deletion process for all UAT buckets and implement lifecycle policies for long-term management.
4. **Cloud Logger**:
   * Implement log filtering, analyze root causes of excessive error logs, and migrate useful logs to Datadog.

#### **Conclusion:**

By focusing on Composer2, Compute Engine, Storage Buckets, and Cloud Logger optimizations, the overall GCP spend can be significantly reduced. Implementing log filtering and migration strategies will further streamline costs while maintaining operational efficiency across the cloud environment.