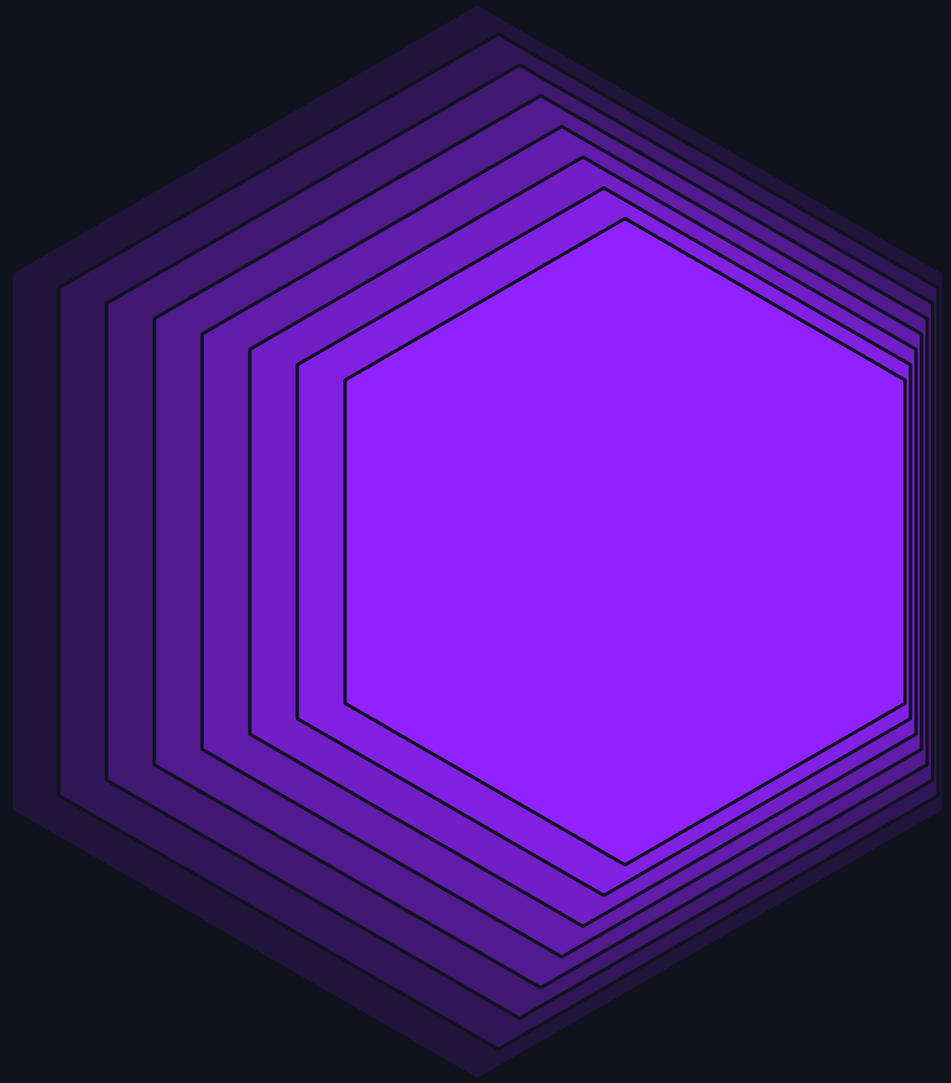
DATA+AI SUMMIT BY databricks

Learn how to manage cost for data and AI workloads with Databricks on AWS



Venkat Viswanathan (AWS) and Jeroen Meulemans (Databricks)

June 12th 2024

Agenda

Customer challenges

Principles of cost optimization

Best practices

Adopt AWS cloud financial management (CFM)

Customer challenges

Global usage tracking

Track and interpret usage across regions, workspaces and accounts

Take corrective action through identifying anti patterns and outliers

Measuring value

Track return on investments - ensuring that projects deliver more value than they spend to run

Objectively justify spend for early stage developments and deployments

Governance

- CoEs want to maximize their efforts and articulate the value they bring to teams
- Establish monitoring and enforcing behaviors across diverse requirements

Principles of cost optimization

1. Choose optimal resources.
2. Dynamically allocate resources.
3. Monitor and control cost.

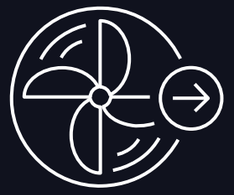
Simplify your right sizing journey with AWS Compute Optimizer

1 Make the right choice

Applies insights from millions of workloads to make recommendations

Saves time comparing and selecting optimal resources for your workload

2 Ensure you are **consistently** making the right choice



Continuously scan your resource usage and match your workload to optimal resources

Example: Amazon EC2 instances

M5.2xlarge
vCPU: 8
RAM: 32 GiB
Instance storage: EBS only
Network: Up to 10 Gbps
Estimated monthly cost: $280.32

General Purpose VM

Databricks workload

~ 40% CPU during the day

~ 10% CPU during the night

~ 30% RAM usage

< 1 Mbps network usage more than 99 percent of the time

< 2 IOPS more than 99 percent of the time

Option 1

M5.xlarge
vCPU: 4
RAM: 16 GiB
Instance storage: EBS only
Network: Up to 10 Gbps
Estimated monthly cost: $140.16
Savings: 50.0 percent
Risk: Low

General Purpose VM

Option 2

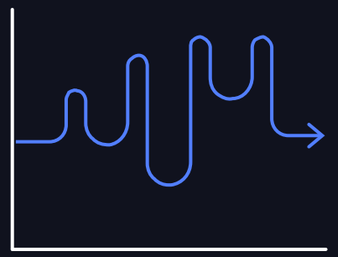
R5.large
vCPU: 2
RAM: 16 GiB
Instance storage: EBS only
Network: Up to 10 Gbps
Estimated monthly cost: $91.98 **Savings: 67.2 percent**
Risk: Medium

Memory Optimized VM

Optimize compute costs with multiple purchase options

On-demand

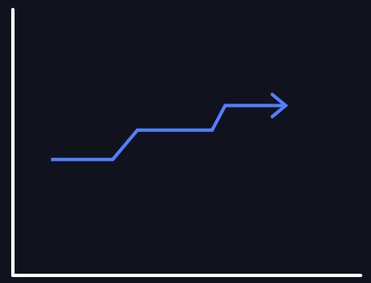
Pay-for-what you use with no long-term commitments



Stateful spiky workloads

AWS Savings Plans

Up to 72 percent savings for one- or three-year hourly usage commitments



Committed and steady-state usage

Spot

Spare capacity at up to 90 percent off on-demand prices

Fault-tolerant, flexible, stateless workloads

The best practice is to combine all three purchase options

Types of Savings Plans

Compute Savings Plans

Offers the **greatest flexibility**

discounts of **up to 66 percent**

Automatically applied to any usage across:

- Region
- Instance family
- Instance sizes
- Tenancy
- Operating system
- Compute service Options

Amazon EC2 Instance Savings Plans

Provides the **deepest savings**

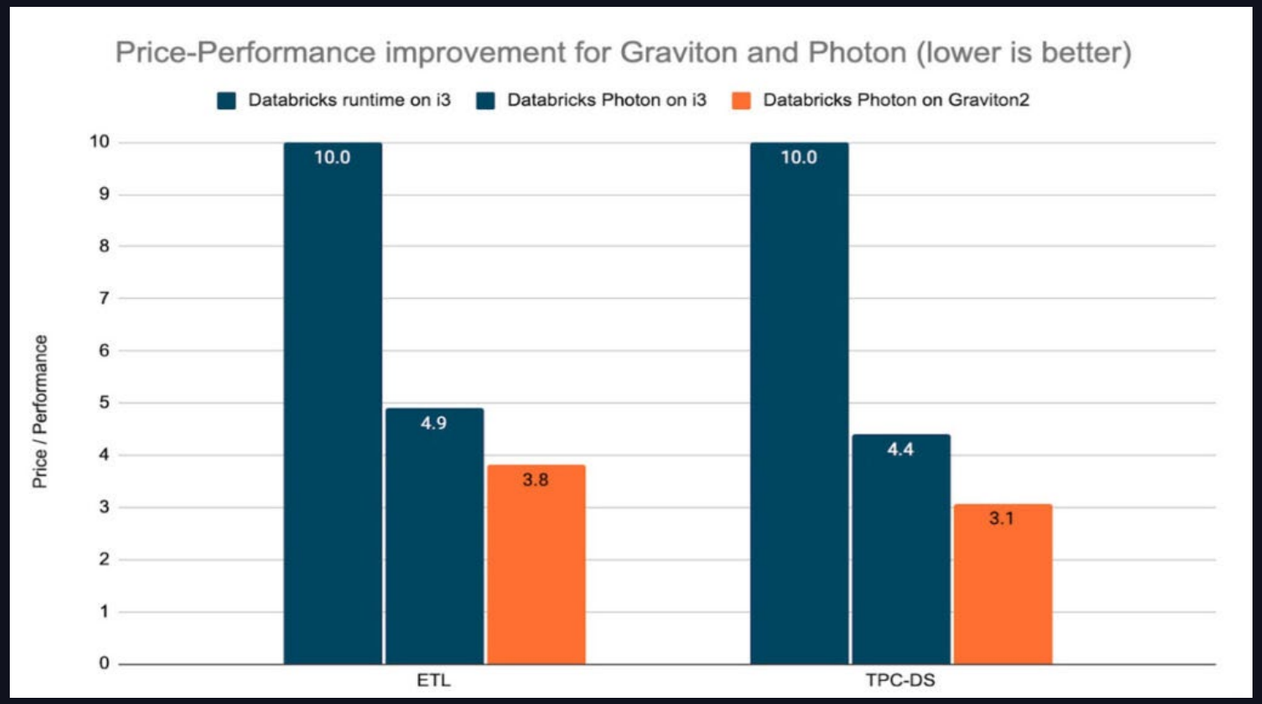
discounts of **up to 72 percent**

Automatically applied to selected Amazon EC2 instances and regions across:

- Instance sizes
- Tenancy
- Operating system

Leverage custom built AWS Graviton3

M7g and R7g AWS Graviton3 instances are now supported on Databricks



Graviton2 - 3.2x better price-performance for the TPC-DS workload

Graviton2 - 2.6x better price-performance for the ETL workload

Graviton3 - Designed to deliver up to 25% better performance than Graviton2

Graviton4 - Up to 40% faster for databases, 30% faster for web applications, and 45% faster for large Java applications than the Graviton3. It is in AWS Preview*

Announcing Databricks support for AWS Graviton2 with up to 3x better price-performance

Graviton3 delivers up to 25% better performance than Graviton2

Join AWS Preview - Graviton4 performance.

Adopt Delta format

Reduced storage costs
Faster data transfers
Improved query performance

Open format based on Parquet
Columnar data format
Mutable

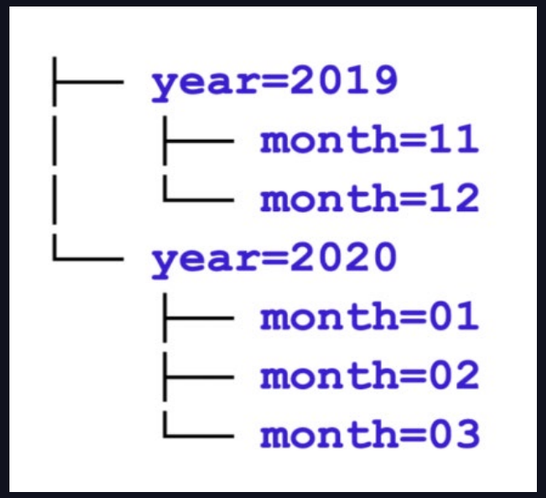
Dataset	Size on Amazon S3	Query Run Time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet Format	130 GB	6.78 seconds	2.51 GB	$0.01
Savings	87% less when using Parquet	34x faster	99% less data scanned	99.7% savings

May 2024 post

Reduce IO through data skipping

Hive-style partitioning is a hierarchical partitioning.

```
--
CREATE TABLE delta_table (name
STRING, date DATE, amount INT)
USING delta
PARTITIONED BY (date);
```



Z-ordering co-locates related information

```
--
OPTIMIZE events
WHERE date >= current_timestamp()
- INTERVAL 1 day;
ZORDER BY (eventType);
```

- By adding data column for Z-ordering, query time went down from 120 minutes to 20 minutes, divided by 6!
- In combination with Ingestion Time Clustering, a large online retailer saw SELECT queries with performance gains of 19x on average.

Liquid clustering replaces table partitioning.

```
--
ALTER TABLE <table_name>
CLUSTER BY (<clustering_columns>)
```

Key benefits:
Automatically clusters data based on query patterns and data distribution.
Improves write time by up to 7x compared to Hive-style partitioning and Z-order.
Improves query performance by up to 12x.

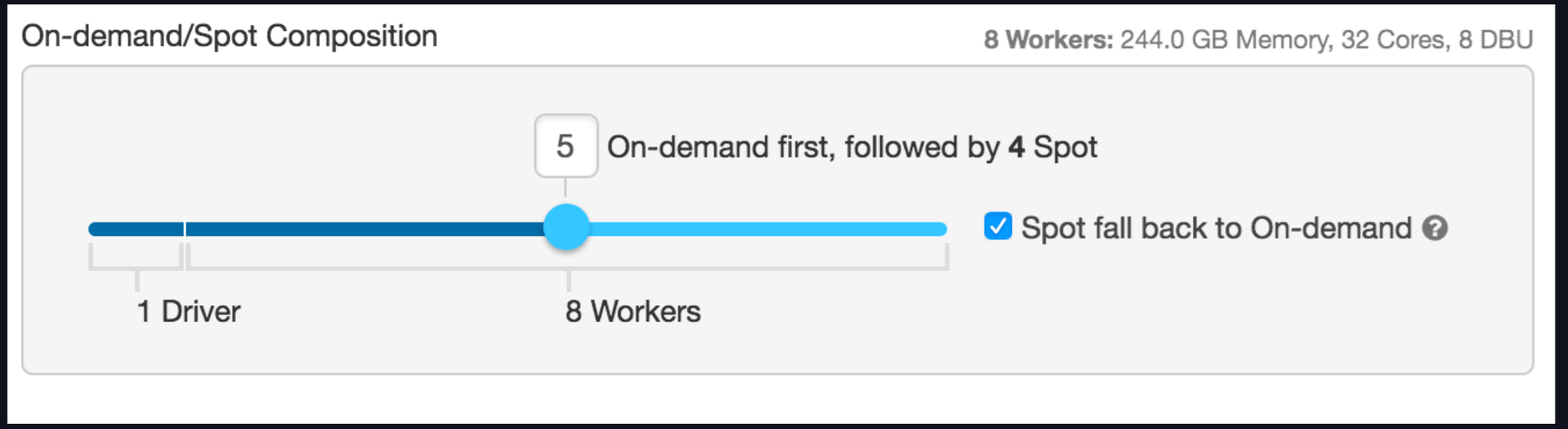
Balance between on-demand and spot

Use a mix in your cluster and keep driver on-demand.

Is there an SLA where on-demand must be there?

Can you trade off with spot instances for long running jobs?

Anti pattern = 100% spot



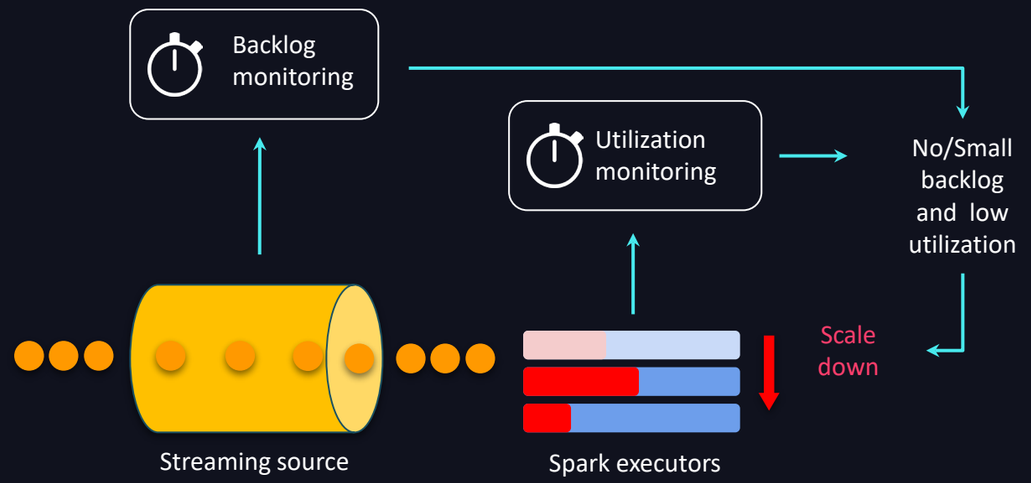
Dynamically Allocate Resources

Enhanced Autoscaling with Delta Live Tables

Save infrastructure costs while maintaining end-to-end latency SLAs for streaming workloads

Problem

Optimize infrastructure spend when making scaling decisions for streaming workloads



- Built to handle streaming workloads which are spiky and unpredictable
- Shuts down nodes when utilization is low while guaranteeing task execution
- Only scales up to needed number of nodes

Experiment and Evaluate Serverless through TCO model

Dependent on understanding your usage (i.e., query serving)

Key cost savings are idle time removal, performance optimization, infrastructure management

Photon enabled by default

AWS TCO		unit
Job length	8.0	hours
Instance type	SPOT	On Demand, RI or SPOT
Classic idle time	15.0%	% of classic instance time not spent querying
Serverless performance gains	0.0%	(Serverless - Classic query time)/(Classic query time)(%)
Serverless efficiency	85.0%	Net serverless/classic runtime
T-shirt Size	Large	DB SQL tshirt sizes
Region	US East (N.VA)	Select from US, EU, AP
Databricks Discount	0.0%	applied to Databricks spend
AWS On Demand Discount	0.0%	applied to AWS spend

Mosaic AI Foundation Model Serving

Cost of your Generative AI usecase depends on

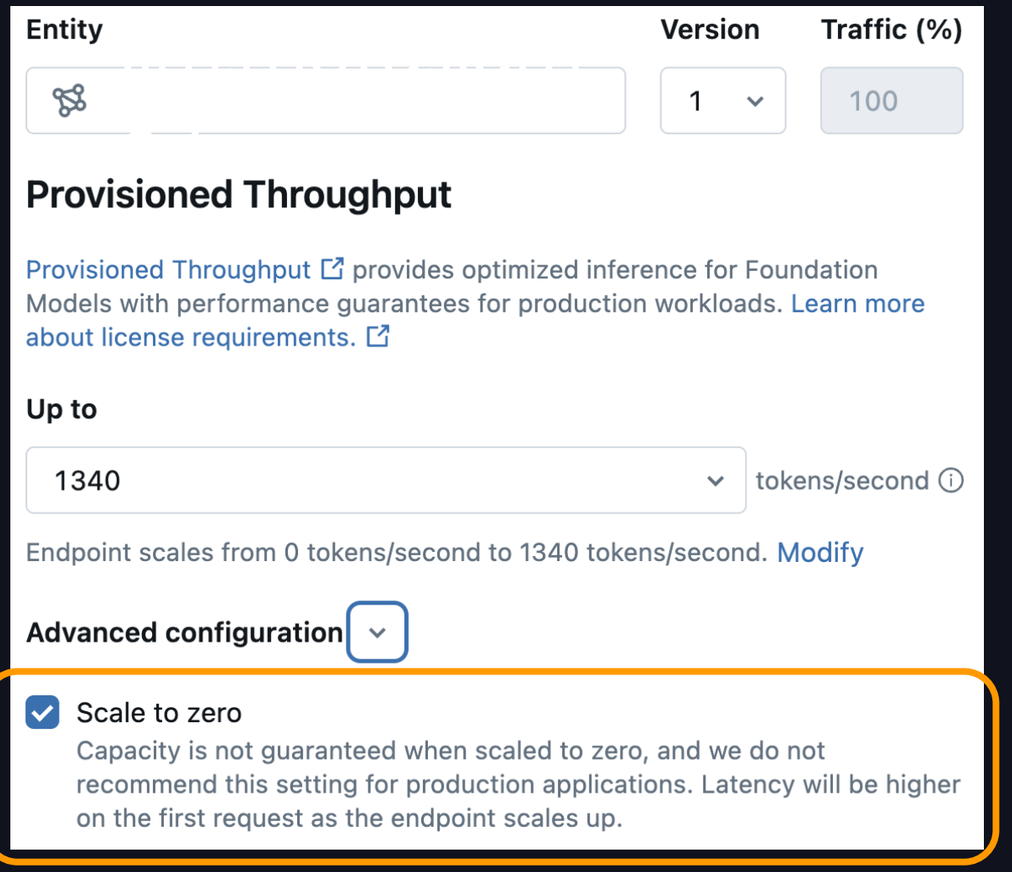
- Number of users
- Tokens per query
- Number of queries
- Performance (MoE vs. dense)

Both pay-per-token and provisioned throughput

Provisioned throughput for PROD

Scale to zero available

No upfront reservation required

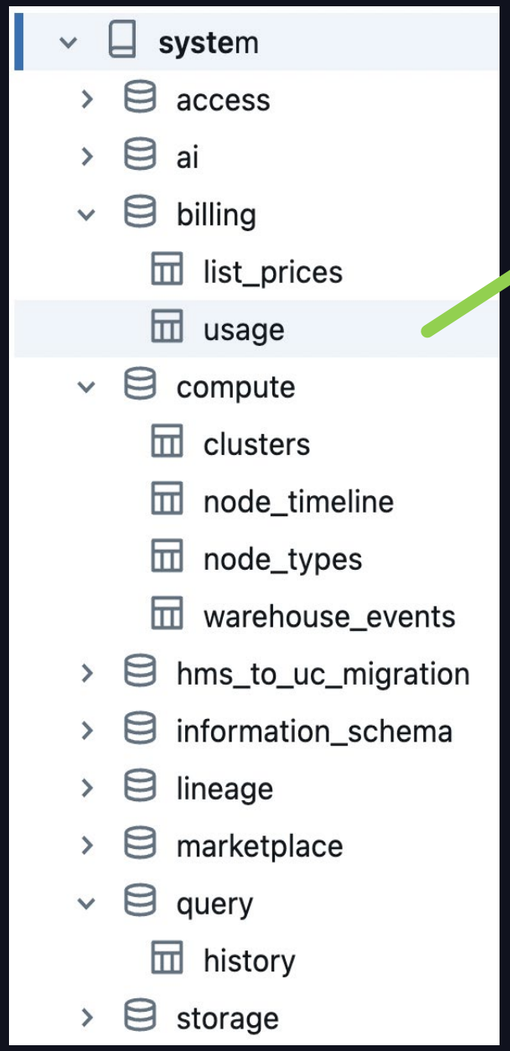


Monitor and Control Cost - SYSTEM TABLES for observability

They can be used for granular cost reporting, forecasting, and controlling your cost.

- Out of the box Lakeview usage dashboards
- Build custom usage reports
- Join AWS cost reporting data with SYSTEM TABLES

Analyze and optimize DBU consumption through SYSTEM TABLES

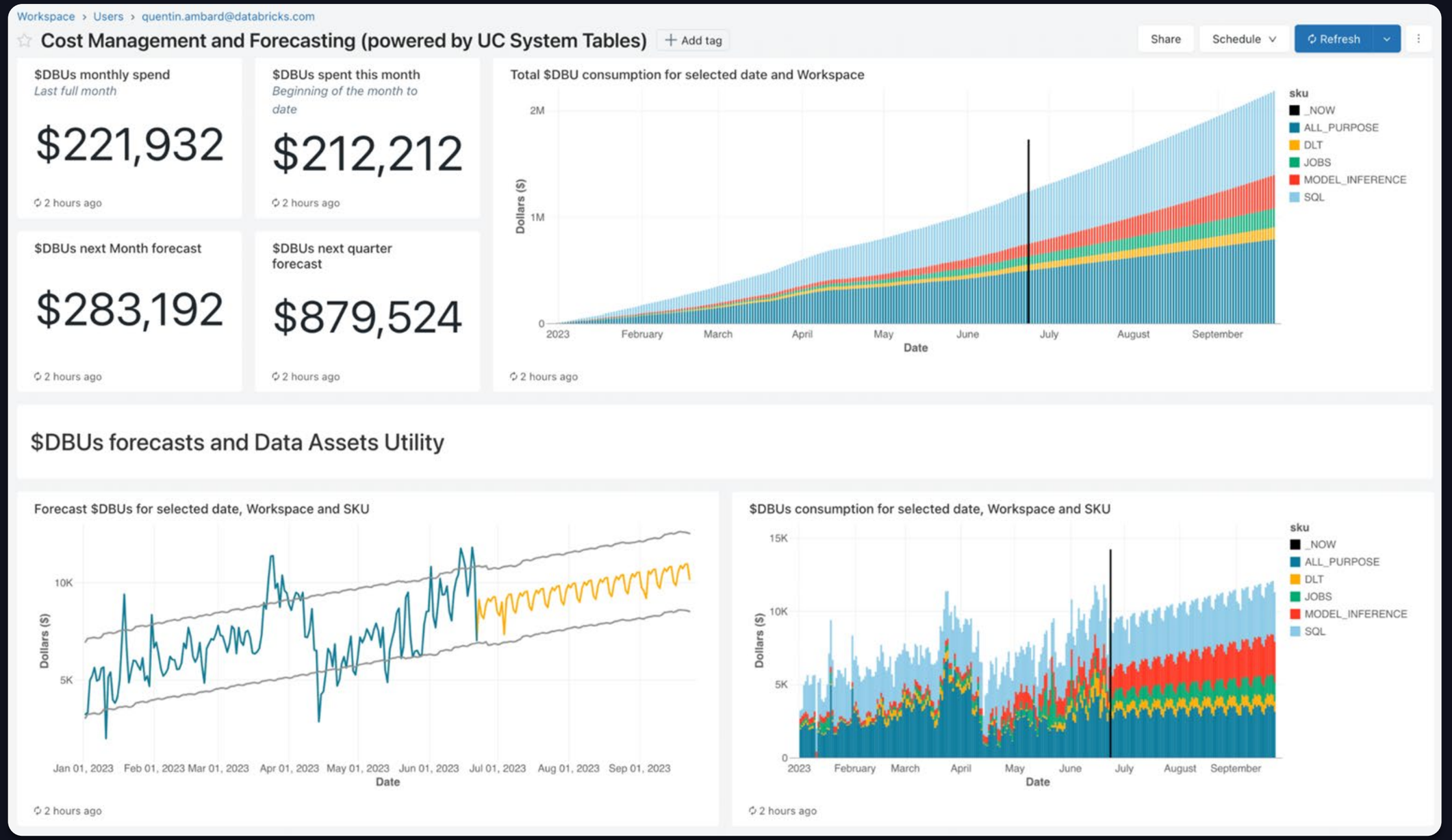


Billing logs materialize as system tables as they are processed through the billing pipeline.

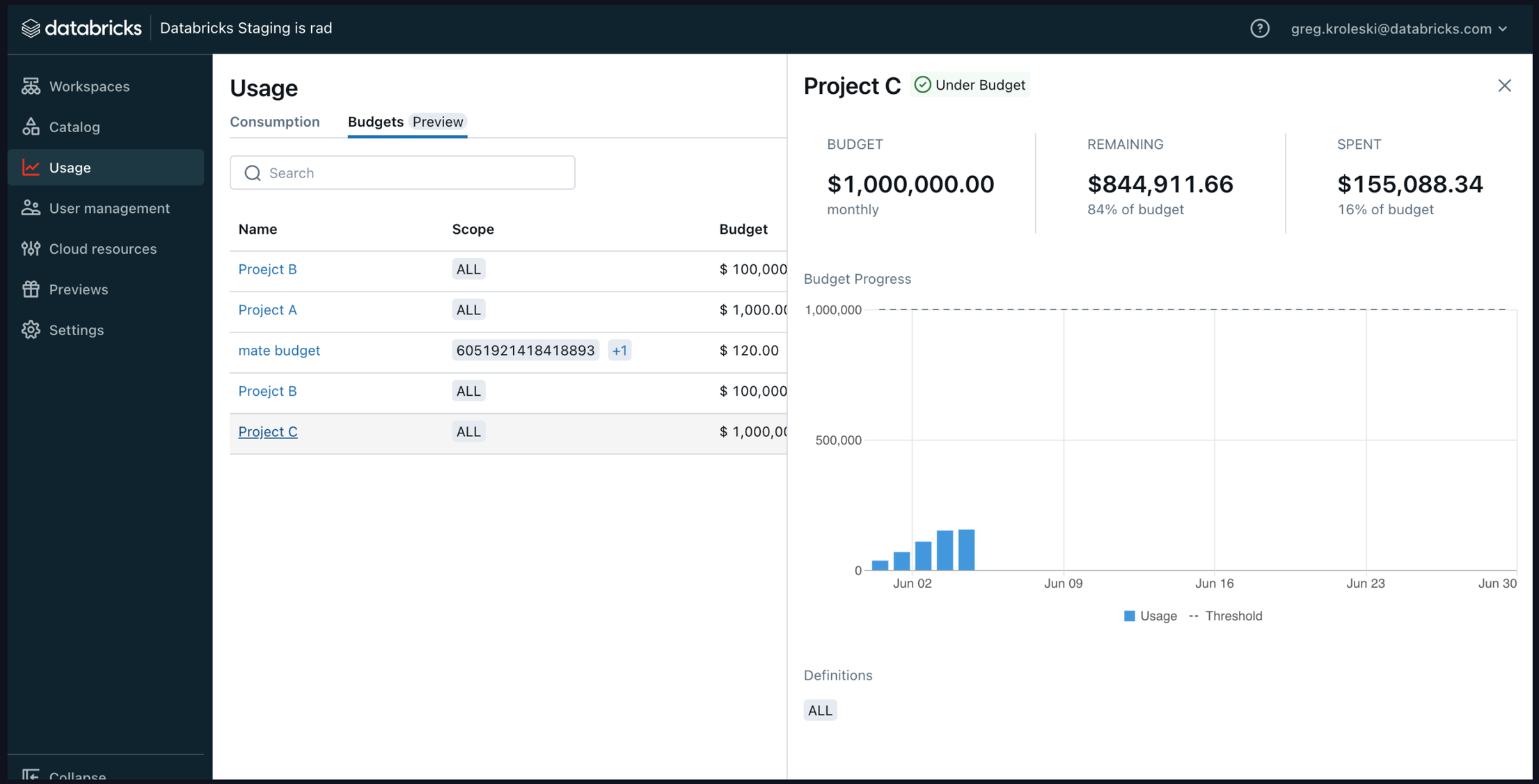
Column	Type
account_id	string
workspace_id	string
usage_record_id	string
sku_name	string
cloud	string
usage_start_time	timestamp
usage_end_time	timestamp
usage_date	date
custom_tags	map
usage_unit	string
usage_quantity	decimal
system_metadata	struct

Data available with one-hour p99 latency and 365-days retention

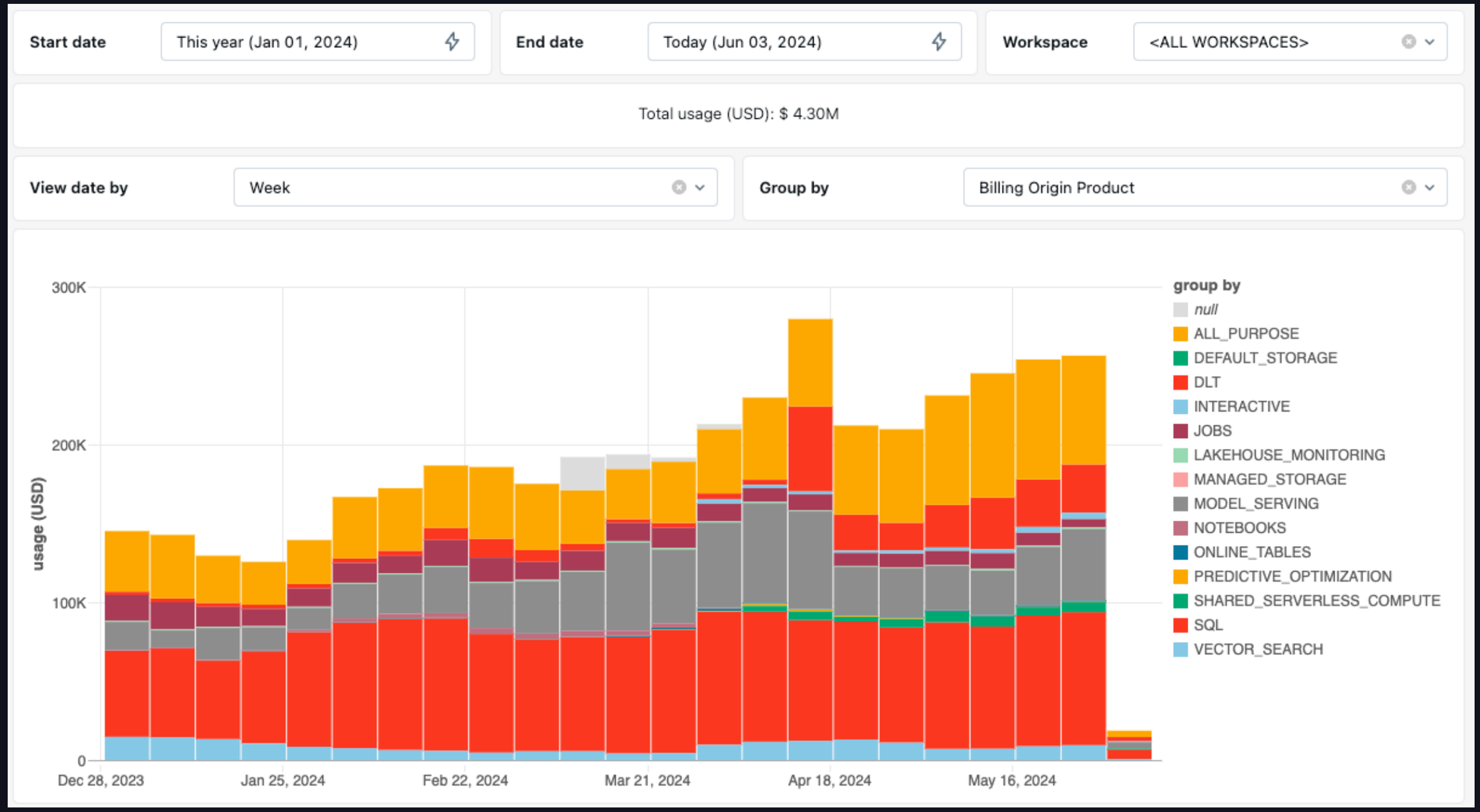
Share cost reports regularly



New capability - Budget Alerts in PuPr (only on AWS)



Lakeview Cost and Usage Dashboard (Public Preview)



Adopt AWS cloud financial management (CFM)

See

Measurement and accountability

Organize and report

Near real-time visibility to make informed cost optimization decisions

Save

Cost optimization

Optimize costs

Take control of cost and continuously optimize spend

Plan

Planning and forecasting

Improved planning

Set expectations around cloud costs for your projects and applications

Run

Cloud financial operations

Billing with built-in controls

Automatic, policy-based account creation, management, and billing access at scale

21

Thank you!

Venkatavaradhan Viswanathan,
AWS, WW Databricks Architect
visvenky@amazon.com