Part 1: **Databricks Serverless Costs**

Serverless Cost Heuristic:
Workload Cost ≈ DBU Price * DBU Emission Rate + Other - Discounts

(1) Serverless Cost Estimation Strategies

Most serverless prices depend on the geography and billing plan so determine your cloud vendor, workspace region, and pricing tier first:

- The workspace **region** is displayed on the account page in the *Workspaces* sidebar or may be shown in the workspace UI within the top right drop-down menu
- The **pricing/platform tier** is displayed on the account page in the *Workspaces* or *Settings* sidebar

Next, the Databricks products in use should be determined as different compute types may have different costs. The compute **service** should be apparent from the workspace UI or compute configuration pages, the first two columns of the DBU price table in section (2) provide pointers to and typical use cases for various serverless services. Multiple products may be billed via the same SKU which means that they share the same list price, the table below has several merged cost cells.

The individual cost factors in the heuristic can now be addressed:
Section (2) explains the **DBU price** factor in more detail, section (3) the DBU **Emission Rate**
Other refers to costs that may be incurred via secondary services:

- Data transfer and networking cost may be charged, for example when data needs to move across cloud regions
- The use of some Databricks Managed Services may activate DBU multipliers that increase the emission rate for associated services

Discounts are temporary or customer specific price reductions like promotions for new products

The last two serverless cost factors are partially reflected in system tables: Their usage quantities subsume DBU multipliers and the *pricing.promotional.default* field of the `system.billing.list_prices` table captures promotional prices. Its *pricing.effective_list.default* value indicates whether a promotional or list price was used for cost calculations. This page and the official docs contain more information about system tables.

(2) Serverless DBU Prices

The **system table** `system.billing.list_prices` is a Type 2 Slowly Changing Dimension table, it contains a complete DBU price history. Its *sku_name* column values need to be matched to relevant Databricks service properties. **Serverless SKU**s consist of up to three segments: A tier prefix like *PREMIUM* is followed by a product infix like *SERVERLESS_SQL_COMPUTE*. If the price is localised, a region suffix like *US_EAST_N_VIRGINIA* (= AWS region code *us-east-1*) completes the SKU name: *PREMIUM_SERVERLESS_SQL_COMPUTE_US_EAST_N_VIRGINIA*

The following Databricks SQL query collects the region suffixes by filtering the SKU names for a widely available service, Serverless Job Compute:

```
SELECT DISTINCT sku_name FROM system.billing.list_prices
WHERE sku_name LIKE SOME('%JOBS_SERVERLESS_COMPUTE%', '%AUTOMATED_SERVERLESS_COMPUTE%')
ORDER BY sku_name
```

Once the Databricks tier and region suffix are determined, all associated serverless SKUs and prices can be retrieved with a query like the following:

```
-- Sample query for a premium AWS Databricks workspace in us-east-1
DECLARE VARIABLE region STRING DEFAULT 'US_EAST_N_VIRGINIA'; -- Adapt
DECLARE VARIABLE tier STRING DEFAULT 'PREMIUM'; -- or 'ENTERPRISE'
DECLARE VARIABLE sku_pattern STRING DEFAULT tier || "%SERVERLESS%" || region;

SELECT sku_name, pricing.default AS DBU_list_price, pricing.effective_list.default
  AS DBU_effective_price, currency_code, price_start_time
FROM system.billing.list_prices lp_1
WHERE price_start_time = (
  SELECT MAX(price_start_time) FROM system.billing.list_prices lp_2
  WHERE lp_1.sku_name = lp_2.sku_name AND lp_2.sku_name LIKE sku_pattern
)
```

Alternative sources for DBU costs are the official Pricing Calculator and **documentation**. The following table provides a summary, links to the service-specific pricing pages are included in its *Compute Type* column. As most serverless DBU costs are regionalized, the cost cells below only mention sample list prices for workspaces in "US East"

		Cloud Vendor:	AWS		Azure	Google Cloud	
Compute Type	Typical Use Cases	Pricing Tier:	Premium	Enterprise	Premium	Premium	Enterprise
Jobs Clusters Declarative Pipelines Lakeflow Connect	Data Engineering: Procedural Approach Declarative Approach Data Ingestion	DBU Cost (regionalized):	US East (N Virginia): $0.35	US East (N Virginia): $0.45	US East: $0.45	US (Virginia): $0.35	US (Virginia): $0.45
All-Purpose Clusters Databricks Apps	Interactive Code, Data Science, ML Native Data Applications		US East (N Virginia): $0.75	US East (N Virginia): $0.95	US East: $0.95	US (Virginia): $0.75	US (Virginia): $0.95
SQL Warehouses	SQL Analytics, BI Apps, Native Dashboards	DBU Cost (regionalized):	US East (N Virginia) / US East / US (Virginia): $0.7				
Lakebase	Transactional Database	Regionalized Cost:	US East (N Virginia): $0.092 (Capacity Unit H)	US East (N Virginia): $0.111 (Capacity Unit H)	US East: $0.52 (per DBU)	--	
		DSU Cost (regionalized):	US East (N Virginia): $0.023		US East: $0.026	--	
Services for AI / Machine Learning		Various services with regionalized and specialized prices					

(3) DBU Emission Rates

Estimating the DBU consumption for a particular serverless workload may not be easy. The official docs mention emission rates for some services but they may not always be constant: Whereas a Databricks App consumes a fixed amount of DBUs for each hour it is active, a serverless warehouse or compute cluster may scale up or down which leads to higher/lower DBU consumption. Furthermore, their underlying instance types may not be visible to the user or change.

=> It is easier to derive the total DBU usage and cost for specific serverless workloads after their completion by querying various system tables: The `system.billing.usage` **fact table** contains usage data including SKU consumption and metadata for various workspace objects. This allows users to join its records with other system tables like `system.billing.list_prices` and determine the total DBU consumption and cost of specific activities.

The following sample query aggregates the total DBUs and costs for specific **job runs** with tasks executed on serverless infrastructure during the last two months:

```
WITH job_serverless_consumption AS (
  SELECT usage_metadata.job_id, usage_metadata.job_run_id, usage_quantity, usage.usage_unit,
    (usage_quantity * pricing.effective_list.default) AS cost, currency_code AS currency, usage.sku_name,
    usage_start_time, usage_end_time, concat(usage.account_id, '#', workspace_id) AS workspace
  FROM system.billing.usage usage LEFT JOIN system.billing.list_prices prices
    ON usage.sku_name = prices.sku_name AND price_start_time <= usage_start_time AND (price_end_time IS NULL
    OR price_end_time >= usage_start_time)
  WHERE usage_date >= CURRENT_DATE() - INTERVAL 2 MONTHS AND product_features.is_serverless = true
    AND usage_metadata.job_id IS NOT NULL AND usage_metadata.job_run_id IS NOT NULL
)

SELECT job_id, job_run_id, SUM(usage_quantity) AS total_usage, usage_unit, SUM(cost) AS total_cost,
  currency, sku_name, MIN(usage_start_time) AS usage_start, MAX(usage_end_time) AS usage_end, workspace
FROM job_serverless_consumption
GROUP BY ALL ORDER BY total_cost DESC
```

Not all service consumption can be analysed to such a fine-grained level with the `usage` table alone: The following query aggregates costs per serverless **DLT pipeline** and **SQL warehouse**, its result subsumes multiple runs or queries from different users:

```
WITH dlt_warehouse_consumption AS (
  SELECT coalesce(usage_metadata.dlt_pipeline_id, usage_metadata.warehouse_id) AS id, billing_origin_product
    AS product, usage_quantity, usage.usage_unit, (usage_quantity * pricing.effective_list.default) AS cost,
    currency_code AS currency, usage.sku_name, concat(usage.account_id, '#', workspace_id) AS workspace
  FROM system.billing.usage usage LEFT JOIN system.billing.list_prices prices
  ON usage.sku_name = prices.sku_name AND price_start_time <= usage_start_time
    AND (price_end_time IS NULL or price_end_time >= usage_start_time)
  WHERE usage_date >= CURRENT_DATE() - INTERVAL 2 MONTHS AND product_features.is_serverless = true AND
    (usage_metadata.dlt_pipeline_id IS NOT NULL OR usage_metadata.warehouse_id IS NOT NULL)
)

SELECT id, SUM(usage_quantity) AS usage, usage_unit, SUM(cost) AS total_cost, currency, sku_name,
  product, workspace
FROM dlt_warehouse_consumption GROUP BY ALL
```

To derive more detailed insights like query time rankings for specific warehouses, joins to other tables that contain more granular metadata (like `system.query.history`) are required. This page contains more system table queries for various use cases.

made by / bigdatadude