



DP-750 Domain 1a: Select and Configure Compute

DP-750 study guide, 'Select and configure compute in a workspace'; Azure Databricks compute docs, as of 2026 | Compute types, access modes, sizing, autoscaling, termination, pools, Photon, runtimes, libraries, permissions, policies, cost

DP-750 Domain 1 Compute Photon Cluster policies

Choose the compute type

Compute types side by side

	Recommended			
	Serverless (notebooks, jobs, pipelines)	Job compute (classic)	All-purpose (classic)	SQL warehouse
Starts in	seconds	3-6 min (less from a pool)	3-6 min	seconds (serverless) to minutes (pro, classic)
VMs in	Databricks serverless plane	your subscription (managed RG or VNet injection)	your subscription	serverless plane or your subscription
You configure	almost nothing (environment, budget policy)	node type, workers, runtime, Spark conf	everything + auto termination	size, min/max clusters, auto stop
Photon	always on	option	option	always on
Best for	default for new jobs and pipelines	scheduled jobs needing custom VMs, init scripts, network	interactive development, shared exploration	SQL, dashboards, BI tools, materialized views
Bill	DBUs only	DBUs (job rate) + VMs	DBUs (higher rate) + VMs	DBUs (+ VMs if not serverless)

Access mode (API value)	Who	Unity Catalog	Use it when
Standard, formerly Shared (USER_ISOLATION)	many users, isolated	yes	shared team cluster for SQL and Python; row filters and column masks enforced; no RDD APIs, limited Scala UDFs, no GPU ML runtime, GPUs, RDD or JVM libraries, R; jobs that run as a service principal
Dedicated, formerly Single user (SINGLE_USER)	one user or one group	yes	never for governed data: no Unity Catalog, no user isolation
No isolation shared (legacy)	anyone	no	

EXAM TIP

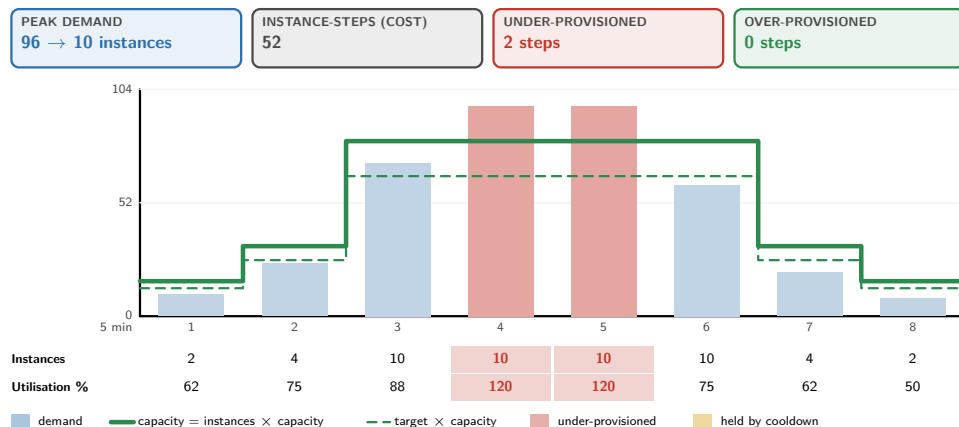
'Shared compute' in the study guide = Standard access mode. A scheduled nightly job → job compute or serverless jobs, never an all-purpose cluster (higher DBU rate, idle time). 'Fastest start, no infrastructure to manage' → serverless. 'BI tool or dashboard runs SQL' → SQL warehouse. 'Needs GPU or ML libraries' → Dedicated access mode + ML runtime.

Configure performance and features

Cluster settings that the exam names (Clusters API / UI)

Flag	Arg	Description
SIZE		
node_type_id / driver_node_type_id	VM	worker and driver Azure VM sizes: memory-optimised (E) for joins and caching, compute-optimised (F), storage-optimised (L) for disk cache, GPU (NC) e.g. Standard_E8ds_v5
num_workers	N	fixed worker count; 0 + single-node profile = driver only e.g. 4
autoscale.min_workers / max_workers	N / M	optimised autoscaling between the bounds from pending tasks e.g. 2 / 10
instance_pool_id	ID	draw VMs from a warm pool: faster start, no DBU while idle
COST		
autotermination_minutes	MIN	stop an idle all-purpose cluster; 0 disables e.g. 30 note: UI default 120 min
azure_attributes.availability	MODE	SPOT_WITH_FALLBACK_AZURE puts workers on spot VMs and falls back to on-demand; the driver stays on-demand e.g. SPOT_WITH_FALLBACK_AZURE
FEATURES		
spark_version	DBR	Databricks Runtime; choose an LTS, an ML variant for ML libraries e.g. an LTS such as 15.4.x-scala2.12
runtime_engine	PHOTON STANDARD	enable Photon on classic compute e.g. PHOTON
data_security_mode	MODE	access mode: USER_ISOLATION (Standard) or SINGLE_USER (Dedicated) + single_user_name
spark_conf / spark_env_vars	K=V	Spark settings and environment variables; secrets as {{secrets/scope/key}} e.g. spark.sql.shuffle.partitions=auto
GOVERNANCE		
policy_id / custom_tags	ID / K=V	apply a cluster policy; tags flow to Azure cost reports e.g. team=data-eng

Autoscaling a nightly job: tasks waiting vs workers (8 cores each)



Wanted instances = $\lceil \text{demand} \div (\text{capacity per instance} \times \text{target}) \rceil = \lceil \text{demand} \div (8 \times 0.8) \rceil$, kept between min 2 and max 10;
 utilisation = $\text{demand} \div (\text{instances} \times \text{capacity})$. Scale-out happens at once, scale-in waits 1 step after the last change.

Demand = concurrent Spark tasks; capacity = cores per worker. Max 10 caps the peak (under-provisioned steps), min 2 keeps a floor; Databricks scales down only workers that are idle

Photon helps	Photon does not help
SQL and DataFrame scans, joins, aggregations on Delta/Parquet	Python or Scala UDFs, RDD code, pandas on the driver
MERGE, UPDATE, DELETE on Delta, writes with many columns	short queries dominated by start-up time
wide tables, large shuffles, SQL warehouses (always on)	costs a higher DBU rate on classic: measure before/after

Runtime choice

Pin an **LTS** Databricks Runtime for production (three years of support) and test upgrades on a copy of the job; choose **ML** runtimes for MLflow, TensorFlow, PyTorch and GPU drivers. Serverless has no version to pick: it follows Databricks' release train with an environment version for Python dependencies.

Libraries, permissions and policies

Install where	Scope	How	Watch out
Compute-scoped library	every notebook and job on the cluster	Libraries tab or API: PyPI, Maven, CRAN, wheel/JAR from a Unity Catalog volume or workspace file	restart may be needed to uninstall; on Standard access mode JARs must be on the allowlist
Notebook-scoped	this notebook session only	%pip install pkg==1.2.3 at the top, then dbutils.library.restartPython()	does not survive a detach; pin versions
Serverless environment	a serverless notebook or job task	Environment side panel or environments in the job: dependencies list	no init scripts, no JARs on serverless notebooks
Job task dependency	one task of a job	libraries in the task (wheel from a volume)	build the wheel in CI, store it versioned
Init script	cluster start	script in a volume or workspace file	DBFS-hosted init scripts are deprecated; prefer volumes

cluster-policy.json: limit what data engineers can create

```
{
  "spark_version": {
    "type": "unlimited",
    "defaultValue": "auto:latest-lts",
  },
  "node_type_id": {
    "type": "allowlist",
    "values": ["Standard_E8ds_v5", "Standard_E16ds_v5"],
  },
  "autoscale.max_workers": {
    "type": "range",
    "maxValue": 10,
  },
  "autotermination_minutes": {
    "type": "range",
    "minValue": 10,
    "maxValue": 60,
    "defaultValue": 30,
  },
  "data_security_mode": {
    "type": "fixed",
    "value": "USER_ISOLATION",
  },
  "custom_tags.cost_center": {
    "type": "fixed",
    "value": "DE-42"
  }
}
```

Permission on compute Can do

CAN ATTACH TO	attach notebooks, run code, see the Spark UI and logs
CAN RESTART	+ start, restart and terminate
CAN MANAGE	+ edit configuration, libraries and permissions, delete
CAN USE (on a policy)	create compute with that policy; without the unrestricted cluster creation entitlement users can only create policy-based compute

Databricks CLI	Does
databricks clusters create --json @cluster.json	create a cluster from a Clusters API body
databricks clusters list	IDs, names and states (RUNNING, TERMINATED)
databricks clusters start <cluster-id>	start a terminated cluster (same ID and config)
databricks clusters delete <cluster-id>	terminate (config kept 30 days unless pinned)
databricks libraries install --json @libs.json	install compute-scoped libraries
databricks cluster-policies list	policies you can use
databricks permissions update clusters <id> --json @acl.json	grant CAN ATTACH TO / CAN RESTART / CAN MANAGE

③ Cost of compute

Worked example (illustrative prices, not a quote)

PROBLEM A nightly job runs 2 hours on 1 driver + 4 workers, Standard_E8ds_v5. Compare job compute with an all-purpose cluster, 30 runs a month.

GIVEN

- D = 2 DBU per VM-hour for this VM size (illustrative)
- Pj = 0.30 USD per DBU on job compute, Pa = 0.55 USD per DBU all-purpose (illustrative)
- V = 0.60 USD per VM-hour (Azure VM price, illustrative)
- N = 5 VMs, H = 2 hours per run, R = 30 runs per month

1 Cost per hour on job compute

$$N \times (D \times P_j + V) = 5 \times (2 \times 0.30 + 0.60) = 6.00 \text{ USD/h}$$

Why: the bill is DBUs plus VMs on classic compute

2 Monthly, job compute

$$6.00 \times H \times R = 6.00 \times 2 \times 30 = 360 \text{ USD}$$

Why: the cluster exists only during the run

3 Cost per hour on all-purpose

$$5 \times (2 \times 0.55 + 0.60) = 8.50 \text{ USD/h}$$

Why: higher DBU rate for interactive compute

4 Monthly, all-purpose

$$8.50 \times 2 \times 30 = 510 \text{ USD, + idle time until auto termination}$$

Why: 30 idle minutes per night add $8.50 \times 0.5 \times 30 = 127.50 \text{ USD}$

ANSWER Job compute $\approx$ 360 USD vs $\approx$ 510-640 USD on all-purpose: schedule production on job compute or serverless.

CHECK ✓ Read real rates on the Azure Databricks pricing page per region and tier; query `system.billing.usage` for actual DBUs.

VM part of a dev cluster used 40 % of the month (illustrative discounts)

WORKLOAD
40 % of 730 h = 292 h / month

ON-DEMAND
1.20 USD / h → 350.40 USD / month

CHEAPEST
Spot workers: 155.73 USD / month

Option	Terms	Monthly	Over 12 mo	Saving	Break-even
Pay as you go	pay per hour	350.40 USD	4,204.80 USD	0 %	--
Azure reserved VM, 1 year	—40 %, commit 100 % of hours, 12 mo	525.60 USD	6,307.20 USD	-50 %	60 %
Spot workers	—60 %, 10 % redone	155.73 USD	1,868.80 USD	+56 %	--

Monthly = $\text{rate} \times (1 - \text{discount}) \times \text{committed hours} + \text{rate} \times \text{hours above the commitment} + \text{upfront} \div \text{term}$; pay as you go = $\text{rate} \times (1 - \text{discount}) \times \text{used hours} \div (1 - \text{interruption})$. Break-even utilisation = $(1 - \text{discount}) \times \text{commitment} + \text{upfront} \div (\text{term} \times \text{rate} \times \text{hours})$: above it the commitment beats on-demand. Illustrative prices: check the provider's current price list.

A reservation pays every hour: it only wins for clusters (or pools) busy most of the month. Spot suits fault-tolerant workers; the driver stays on-demand. Databricks also sells DBU pre-purchase plans (DBCUs) on the DBU part

Compute cost and safety baseline

✓ baseline / mandatory □ recommended – verify in your context

✓ Production jobs on job compute or serverless, never on all-purpose clusters

- ✓ Auto termination 15-60 min on every all-purpose cluster enforce it with a cluster policy range
- ✓ Cluster policies + tags (team, cost centre) for every user-created cluster
- ✓ Standard or Dedicated access mode only (Unity Catalog)
- **SHOULD** Pools for jobs that start many short clusters
- **COULD** Spot with fallback on workers of retry-safe batch jobs
- **SHOULD** Serverless budget policies to tag serverless usage

🚩 Traps and recap

MYTH

✗ Autoscaling makes every job cheaper

FACT

✓ It helps variable load; steady jobs run as well on a fixed size
Scale-up takes minutes on classic compute; streaming needs pipelines' enhanced autoscaling.

✗ Photon speeds up any code

✓ Only SQL and DataFrame operators run in Photon
Python UDFs and RDD code fall back to the JVM engine while you pay the Photon DBU rate.

✗ A terminated cluster keeps billing DBUs

✓ Terminated compute costs nothing
Idle pool VMs and the storage of the managed resource group do cost Azure money.

✗ Any cluster can read Unity Catalog tables

✓ Only Standard or Dedicated access modes (and serverless, SQL warehouses)
No isolation shared clusters see only the legacy Hive metastore.

Q1 Nightly ETL, custom VNet routes and an init script needed: which compute?

A Classic job compute (serverless allows no init scripts or custom VM settings).

Q2 Ten analysts share one cluster with row filters enforced: access mode?

A Standard (formerly Shared, USER_ISOLATION).

Q3 Job needs GPUs and PyTorch: runtime and access mode?

A ML runtime (GPU) on Dedicated (single user) access mode.

Q4 Cut cluster start time for many short jobs on classic compute?

A Instance pools with idle instances (or move the jobs to serverless).

Q5 Which permission lets a user start but not edit a cluster?

A CAN RESTART.

Q6 How do you force auto termination $\leq$ 60 minutes for everyone?

A A cluster policy with a range on autotermination_minutes, and remove unrestricted cluster creation.

Q7 Install a pinned package for one notebook only?

A `%pip install pkg==x.y.z` (notebook-scoped), then restart Python.

Q8 Where should wheels and JARs live for compute libraries?

A In a Unity Catalog volume (or workspace files), not in DBFS root.

Q9 Which compute serves Power BI or a dashboard over SQL?

A A SQL warehouse (serverless preferred), Photon always on.

Serverless first

seconds to start, no VM tuning

Jobs on job compute

cheaper DBU rate, no idle

Standard vs Dedicated

shared isolated vs one user/group

Policies and tags

limit sizes, force termination

Photon for SQL/DataFrames

not for UDFs

Libraries from volumes

pin versions, notebook-scoped with
`%pip`