Databricks Workspace Issues with Mitigations

Databricks Workspace Issues Comparison Table with Mitigations

#	Workspace Issue	Description	Where It Typically Arises	Workspace Areas Most Affected	Mitigation Strategies
1	Permission Misconfiguration	Users have incorrect access levels, leading to security risks or blocked collaboration.	User management, ACL configuration	Clusters, Notebooks, Repos	Define RBAC roles carefully; use Unity Catalog and workspace access controls to enforce policies.
2	Cluster Sprawl	Too many clusters running without governance, driving up unnecessary costs.	Resource provisioning	Compute resources	Use cluster policies, auto-termination settings, and chargeback reports to control usage.
3	Library Dependency Conflicts	Inconsistent package versions break notebooks and jobs.	Job execution, interactive notebooks	Jobs, Clusters	Pin library versions; leverage cluster init scripts or container services for consistent environments.
4	Inefficient Job Scheduling	Jobs overlap or over-consume resources due to poor scheduling practices.	Workflow orchestration	Jobs, Pipelines	Use Databricks Workflows with dependencies and alerts; stagger schedules to balance workloads.
5	Secret Management Risks	Hard-coded secrets in notebooks compromise security.	Notebook development	Repos, Jobs	Store credentials securely in Databricks Secrets and reference them via environment variables.
6	Ineffective Resource Tagging	Lack of tagging prevents clear cost attribution across teams and projects.	Workspace setup	Clusters, Jobs, Workflows	Apply consistent tagging standards for clusters and jobs to track usage by department or project.
7	Data Access	Inconsistent table	Table and	Unity Catalog,	Use Unity Catalog for

	Inconsistencies	permissions cause confusion and data errors.	catalog access	External Tables	centralized permissions; audit privileges regularly.
8	Notebook Version Drift	Multiple versions of the same notebook create confusion over the source of truth.	Collaborative development	Notebooks, Repos	Use Git integration and enforce version control workflows with pull requests.
9	Inefficient Autoscaling Settings	Clusters over-scale or under-scale, affecting performance and cost.	Cluster configuration	Compute resources	Tune min/max workers; monitor utilization; apply cluster policies to guide configuration.
10	Lack of Monitoring and Alerting	Failures and performance issues go unnoticed due to missing observability.	Production workloads	Jobs, Clusters, Pipelines	Enable job and cluster alerts; integrate with monitoring tools (e.g., Datadog, Azure Monitor).

Quick Reference

- **Clusters** = Compute resources used for notebooks and jobs.
- **Jobs** = Scheduled or triggered workloads.
- **Unity Catalog** = Centralized governance for data access.
- **Repos** = Source control-integrated development.
- **Pipelines** = Orchestrated workflows.

Example Mitigation Code Snippets

Assign RBAC Permissions (Unity Catalog):

sql

CopyEdit

```
GRANT SELECT ON CATALOG main TO `finance_group`
```

Reference Secrets in Notebooks:

python

CopyEdit

```
spark.conf.set("fs.azure.account.key", dbutils.secrets.get(scope="storage-secrets", key="account-key"))
```

Define Cluster Policy:

json

CopyEdit

```
{
  "spark_conf.spark.databricks.cluster.profile": {
    "type": "fixed",
    "value": "serverless"
  },
  "autoscale.min_workers": {
    "type": "range",
    "minValue": 1,
    "maxValue": 2
  }
}
```

Enable Job Alerting:

- In the Jobs UI, configure **“Alerts”** to notify on failures or SLA breaches.