**Databricks Operational Excellence Issues with Mitigations**

**Databricks Operational Excellence Issues Comparison Table with Mitigations**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Operational Issue | Description | Where It Typically Arises | Operational Areas Most Affected | Mitigation Strategies |
| 1 | Lack of Cost Visibility | Teams have no clear picture of compute, storage, or egress spending, leading to budget overruns. | Multi-team environments | Clusters, Warehouses, Storage | Enable cost dashboards; tag resources; set budget alerts in the cloud provider. |
| 2 | Inconsistent Environment Configurations | Environments drift across workspaces, causing unpredictable behavior and debugging headaches. | Workspace provisioning, project onboarding | Clusters, Jobs, Notebooks | Use Terraform or Databricks CLI to automate consistent configurations. |
| 3 | Ineffective Job Monitoring | Failed or delayed jobs go unnoticed, impacting downstream processes and SLAs. | Scheduled workflows, production pipelines | Jobs, Pipelines | Configure alerts on failures; monitor job run histories; use Service Principals for ownership clarity. |
| 4 | Unclear Ownership | It’s not clear who owns which jobs, tables, or clusters, making accountability difficult. | Shared environments, cross-team development | Jobs, Tables, Clusters | Assign owners explicitly via tags or metadata; document ownership in a central catalog. |
| 5 | Lack of Automation | Manual processes increase the risk of errors and slow down development and operations. | CI/CD pipelines, cluster provisioning | Jobs, Clusters, Unity Catalog | Automate with Databricks Repos, Git integration, and deployment pipelines. |
| 6 | Siloed Observability | Monitoring data is spread across logs, cloud tools, and Databricks, reducing visibility. | Production troubleshooting | Jobs, Clusters, SQL Warehouses | Centralize metrics and logs; integrate with cloud-native monitoring and alerting platforms. |
| 7 | Inefficient Resource Utilization | Clusters and warehouses are oversized or idle, driving unnecessary costs. | Ad hoc analytics, scheduled workloads | Compute Resources | Set auto-termination; review cluster sizing; monitor utilization metrics to optimize configurations. |
| 8 | Inconsistent Data Quality Checks | Data pipelines lack validation, letting bad data propagate downstream unnoticed. | Ingestion workflows, ETL pipelines | Bronze, Silver, and Gold Tables | Add validation steps in pipelines; implement data quality dashboards and alerts. |
| 9 | No Disaster Recovery Plan | There is no clear strategy to recover from critical failures or data loss. | Production workloads, critical data assets | Clusters, Jobs, Storage | Define backup policies; test recovery procedures; document disaster recovery playbooks. |
| 10 | Lack of Documentation | Key operational processes and configurations are undocumented, creating institutional knowledge gaps. | Onboarding, troubleshooting, scaling operations | All Platform Areas | Maintain centralized wikis and runbooks; require documentation updates as part of change processes. |

**Quick Reference**

* **Terraform:** Infrastructure as code for provisioning consistent environments.
* **Repos:** Git-backed code and configuration management.
* **Service Principals:** Non-human identities for jobs and automation.
* **Monitoring:** Logs, metrics, and alerts for operational health.
* **Data Quality:** Validations and checks to prevent bad data from spreading.

**Example Mitigation Actions and Configurations**

**Enable Cost Visibility:**

* Use cloud tags:

json

CopyEdit

"tags": {

"Environment": "Production",

"Project": "DataPlatform"

}

* Set budgets and alerts in Azure or AWS Cost Management.

**Automate Cluster Provisioning:**

hcl

CopyEdit

resource "databricks\_cluster" "job\_cluster" {

cluster\_name = "prod-job-cluster"

spark\_version = "13.3.x-scala2.12"

node\_type\_id = "Standard\_DS3\_v2"

autotermination\_minutes = 20

autoscale {

min\_workers = 2

max\_workers = 8

}

}

**Configure Job Alerts:**

* In the UI: *Jobs > Edit > Notifications*
* Send email or webhook on failure or SLA breach.

**Tag Ownership:**

sql

CopyEdit

ALTER TABLE main.sales.transactions SET TAGS ('owner' = 'finance\_team');

**Enable Auto-Termination:**

* Cluster settings:

json

CopyEdit

"autotermination\_minutes": 30

**Validate Data Quality in Pipelines:**

python

CopyEdit

assert df.filter("amount < 0").count() == 0, "Negative amounts detected!"

**Centralize Monitoring:**

* Export logs to cloud storage or SIEM.
* Use Databricks REST API to collect metrics:

bash

CopyEdit

databricks jobs list-runs --active-only