Databricks Workflow Automation & Job Scheduling

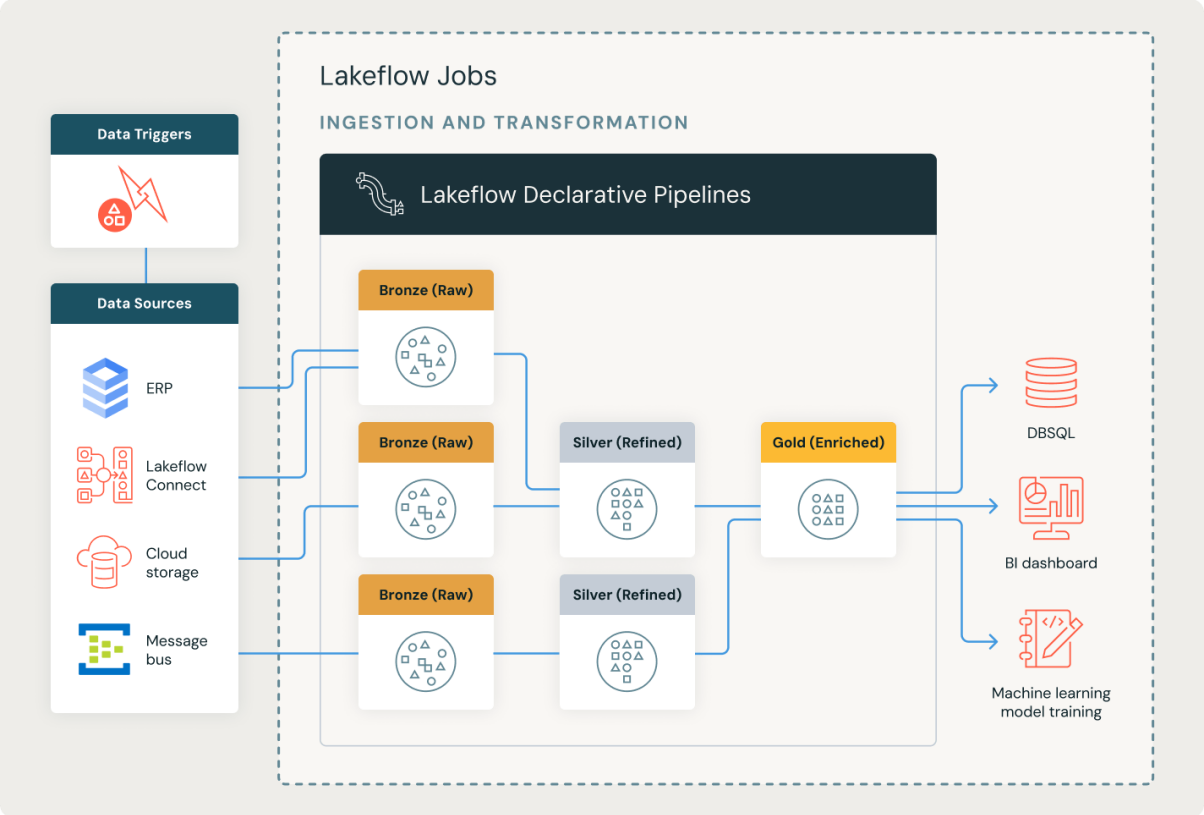
1. Job Scheduling Fundamentals

Core Concepts

- **Jobs**: Reusable execution units (notebooks, JARs, Python scripts)
- **Schedules**: Time-based triggers using cron syntax
- **Job Clusters**: Ephemeral compute resources (created/destroyed per job)
- **Instance Pools**: Pre-warmed VM pools for faster cluster starts

Key Benefits

✓ **Reliability**: Automatic retries and alerting
✓ **Cost Efficiency**: Cluster auto-termination after jobs
✓ **Scalability**: Parallel job execution
✓ **Visibility**: Centralized monitoring



2. Creating Jobs in Databricks

Method 1: UI Workflow

1. Navigate to **Workflows → Jobs → Create Job**
2. Configure:

markdown

```
- Name: "daily-data-pipeline"
- Task Type: Notebook/JAR/Spark Submit
- Cluster: New job cluster or existing
- Schedule: Cron expression (e.g., "0 0 2 * * ?" for daily 2AM)
```

Method 2: Notebook Code

```
# Run notebook as job
result = dbutils.notebook.run(
  "/Production/ETL",
  timeout_seconds=3600,
  arguments={"date": "2023-08-01", "mode": "full"}
)
```

Method 3: Jobs API

```
import requests

job_config = {
  "name": "monthly-report",
  "tasks": [{
    "task_key": "generate-report",
    "notebook_task": {
      "notebook_path": "/Reports/Monthly"
    },
    "new_cluster": {
      "num_workers": 4,
      "node_type_id": "Standard_DS3_v2"
    }
  }]
}

response = requests.post(
  "https://<workspace-url>/api/2.0/jobs/create",
  headers={"Authorization": "Bearer <token>"},
  json=job_config
)
```

3. Advanced Scheduling

Cron Syntax Guide

Example	Description
0 0 * * * ?	Hourly at :00
0 0 12 * * ?	Daily at noon UTC
0 0 9 ? * MON-FRI	Weekdays at 9AM
0 0/15 * * * ?	Every 15 minutes

Multi-Task Dependencies

```
{
  "tasks": [
    {
      "task_key": "ingest",
      "notebook_task": {...}
    },
    {
      "task_key": "transform",
      "depends_on": [{"task_key": "ingest"}],
      "notebook_task": {...}
    }
  ]
}
```

4. Alert Configuration

Email Notifications

```
{
  "email_notifications": {
    "on_start": ["team@company.com"],
    "on_success": ["analytics@company.com"],
    "on_failure": ["alerts@company.com", "admin@company.com"]
  }
}
```

Webhook Integration

```
# In notebook code:
import requests

def send_slack_alert(message):
    webhook = "https://hooks.slack.com/services/..."
    payload = {"text": f"Job Alert: {message}"}
    requests.post(webhook, json=payload)

# On failure:
try:
    main_process()
except Exception as e:
    send_slack_alert(f"Failed: {str(e)}")
    raise
```

5. Job Clusters & Pools

Cluster Pools Setup

```
# Create pool via API
pool_config = {
  "instance_pool_name": "etl-pool",
  "min_idle_instances": 2,
  "node_type_id": "Standard_DS3_v2",
  "idle_instance_autotermination_minutes": 30
}

requests.post(
  "https://<workspace-url>/api/2.0/instance-pools/create",
  headers={"Authorization": "Bearer <token>"},
  json=pool_config
)
```

Attaching Jobs to Pools

```
{
  "new_cluster": {
    "instance_pool_id": "pool-123456",
    "spark_version": "12.2.x-scala2.12"
  }
}
```

📌 **Benefits:**

- 50-75% faster cluster starts
- Cost savings through instance reuse
- Consistent worker configurations

6. Performance Optimization

Cluster Config Tips

```
{
  "new_cluster": {
    "spark_conf": {
      "spark.sql.shuffle.partitions": "200",
      "spark.executor.memory": "8g"
    },
    "aws_attributes": {
      "availability": "SPOT_WITH_FALLBACK"  # 60-80% cost savings
    },
    "autoscale": {
      "min_workers": 2,
      "max_workers": 10
    }
  }
}
```

Library Management

```
{
  "libraries": [
    {"pypi": {"package": "pandas==1.5.0"}},
    {"jar": "dbfs:/libs/custom.jar"}
  ]
}
```

7. Monitoring & Troubleshooting

Key Metrics to Track

- **Job Duration**: vs historical benchmarks
- **Cluster Utilization**: CPU/memory during runs
- **Failure Rates**: By job/task
- **Cost**: DBU consumption per job

Accessing Logs

```
# List recent runs
display(spark.sql("SELECT * FROM system.operational_data.job_runs"))

# Get specific run output
dbutils.fs.head(f"dbfs:/databricks/jobs/{job_id}/{run_id}/driver_logs/log4j-
active.log")
```

8. CI/CD Integration

Azure DevOps Example

```
- task: DatabricksDeployNotebooks@0
  inputs:
    notebooksFolderPath: 'notebooks'
    workspaceUrl: '$(DATABRICKS_URL)'

- task: DatabricksJobCreate@0
  inputs:
    jobDefinition: 'deploy/jobs/prod_etl.json'
```

Terraform Management

```
resource "databricks_job" "nightly" {
  name = "Nightly Processing"

  notebook_task {
    notebook_path = "/Production/ETL"
  }

  schedule {
    quartz_cron_expression = "0 0 22 * * ?"
  }
}
```

9. Security Best Practices

Secret Management

```
# Store credentials
databricks secrets put-scope production
databricks secrets put production db_password --string-value "..."

# Reference in jobs
{
  "spark_conf": {
    "db.password": "{{secrets/production/db_password}}"
  }
}
```

Access Controls

- **Job-specific permissions**
- **Cluster policies** to restrict configurations
- **Table ACLs** for data governance

10. Real-World Example

Production ETL Job

```
{
  "name": "prod_customer_etl",
  "tasks": [
    {
      "task_key": "ingest",
      "notebook_task": {
        "notebook_path": "/ETL/01_Customer_Ingest",
        "base_parameters": {
          "date": "{{yesterday}}"
        }
      },
      "new_cluster": {
        "instance_pool_id": "pool-123",
        "spark_version": "12.2.x-scala2.12",
        "spark_conf": {
          "spark.sql.shuffle.partitions": "200"
        }
      }
    },
    {
      "task_key": "transform",
      "depends_on": [{"task_key": "ingest"}],
      "notebook_task": {
        "notebook_path": "/ETL/02_Customer_Transform"
      }
    }
```

```
  ],
  "schedule": {
    "quartz_cron_expression": "0 0 3 * * ?",  # Daily 3AM
    "timezone_id": "America/New_York"
  },
  "email_notifications": {
    "on_failure": ["data-eng-alerts@company.com"]
  }
}
```

11. Troubleshooting Guide

Issue	Solution
Job not starting	Check schedule timezone, workspace permissions
Cluster fails to launch	Verify instance quotas, IAM roles
Library conflicts	Use isolated clusters, virtualenv
Timeout errors	Increase timeout, optimize notebook code

12. Key CLI Commands

```
# List jobs
databricks jobs list

# Trigger run
databricks jobs run-now --job-id 123

# Get run output
databricks jobs get-run --run-id 456
```

13. Learning Resources

- Databricks Jobs Documentation
- Jobs API Reference
- Instance Pools Guide