**Azure Data & Machine Learning Integration Guide**

***End-to-End Setup for SQL, Compute, and ML Pipelines in Azure***

**1. Create an Azure SQL Database**

**Overview:**

Azure SQL Database provides a fully managed relational database with high availability and built-in security. It serves as the central storage for transformed or production-ready datasets.

**Steps:**

1. Sign into the [Azure Portal](https://portal.azure.com).
2. In the global search bar, type **“SQL databases”** and select it under Services.
3. Click **+ Create** to begin the setup.
4. Under the *Basics* tab:
   * Choose your **Subscription**.
   * Select or create a **Resource Group**.
   * Provide a **Database Name** (e.g., ProductionDB).
   * Create a new SQL **Server** with:
     + Globally unique server name
     + SQL Authentication: username and password
     + Location/Region close to your deployment needs
5. Under *Compute + Storage*:
   * Choose **Serverless** or **Basic** depending on usage volume.
   * For trial users, **Serverless (2 vCores, 32GB)** is recommended to stay within free limits.
6. Review the configuration and click **Create**.

**2. Configure Network Access to Azure SQL**

**Overview:**

Allow secure, remote access to your SQL database for development tools, pipelines, or direct Python integration.

**Steps:**

1. After deployment, navigate to the **SQL Server** resource (not the database).
2. Click **Networking** in the sidebar.
3. Under *Public network access*:
   * Set to **Selected networks**.
4. Click **+ Add client IP** to whitelist your device.
5. (Optional but recommended) Enable:
   * **Allow Azure services to connect**
6. Click **Save**.

**3. Define Tables Within Azure SQL**

**Overview:**

Create schema structures in your database to mirror staging, analytics, or logging needs.

**Steps:**

1. From the SQL Database overview, click **Query Editor (Preview)**.
2. Authenticate using your SQL credentials.
3. Manually define tables (e.g., staging\_data, event\_log) appropriate for your data pipeline.
4. Verify creation via SELECT statements or SSMS.

**4. Dual-Environment ETL Support (Local + Azure SQL)**

**Overview:**

Modern pipelines often ingest data both locally and into cloud environments. Azure SQL can serve as a mirror or primary destination.

**Best Practices:**

* Maintain both **local** and **cloud** database connections.
* Log inserts and row counts to a shared event\_log table.
* Avoid hardcoding credentials; use environment variables or secure key vaults.
* Parameterize target environment selection (e.g., via CLI flags or config files).

**5. Create an Azure Machine Learning Workspace**

**Overview:**

Azure Machine Learning provides a collaborative, cloud-based environment for training and deploying ML models at scale.

**Steps:**

1. In Azure Portal, search for **“Machine Learning”** and click **+ Create**.
2. Fill in:
   * **Workspace Name**
   * **Resource Group**
   * **Region** (match existing services if possible)
3. Accept default settings for Networking, Encryption, and Tags unless required otherwise.
4. Click **Review + Create**, then **Create**.

**6. Provision an Azure Compute Cluster**

**Overview:**

Compute clusters in Azure ML allow you to run notebooks, train models, and deploy jobs with scalable virtual machines.

**Steps:**

1. In **Azure ML Studio**, go to **Compute > Compute Clusters**.
2. Click **+ New** and configure:
   * **Name**: e.g., ml-dev-cpu
   * **VM Size**: Use low-cost options like Standard\_A1\_v2 or Standard\_DS1\_v2
   * **Min Nodes**: 0
   * **Max Nodes**: 1
   * **Idle Shutdown**: 15 minutes (to reduce cost)
3. Leave Security, Applications, and Tags at defaults unless needed.
4. Click **Create**.

**Notes:**

* Clusters auto-scale and auto-pause.
* You only pay when compute is active.
* Suitable for lightweight ML tasks like regression, classification, or preprocessing.

**7. Launch Azure ML Studio and JupyterLab**

**Overview:**

Azure ML Studio provides browser-based tools to develop and monitor ML experiments.

**Steps:**

1. From **Azure ML Studio**, go to **Compute > Compute Instances**.
2. Click on your instance and select **JupyterLab**.
3. Use this environment to:
   * Connect to Azure SQL
   * Register datasets
   * Build, train, and evaluate ML models
   * Track metrics using MLFlow or the built-in tracking dashboard

**8. Integrate Azure SQL Data into Azure ML**

**Overview:**

Bring structured data from Azure SQL directly into your ML workflows.

**Methods:**

* Use built-in connectors (e.g., azureml.dataprep) to read SQL tables.
* Register the resulting DataFrame as a **Dataset** in ML Studio.
* Use versioning, profiles, and labeling to track changes.

**9. Train and Monitor Machine Learning Models**

**Overview:**

Train scikit-learn, XGBoost, or PyTorch models using your compute cluster.

**Best Practices:**

* Store experiment metadata (e.g., accuracy, loss, hyperparameters).
* Use **experiment tracking** to compare model runs.
* Schedule automated runs for retraining or inference.

**10. Cost Optimization Tips**

To stay within free or trial limits:

* Use **Serverless SQL** or Basic SKU only.
* Choose **low-tier VM sizes** for compute.
* Set compute clusters to:
  + **Min nodes = 0**
  + **Idle shutdown = 15 minutes**
* Use **small, lightweight datasets** for experimentation.
* Monitor usage via **Azure Cost Management + Budgets**.

**Summary**

| **Component** | **Purpose** | **Azure Service** |
| --- | --- | --- |
| Data Storage | Store transformed/tabular data | Azure SQL Database |
| Compute for ML | Run training jobs, notebooks | Azure Compute Cluster |
| Pipeline Execution | Reproducible ML workflow orchestration | Azure ML Studio |
| Security | Access control and encryption | Network + SQL Firewall |
| Cost Monitoring | Budgeting and trial protection | Azure Cost Management |