**Monitoring & Troubleshooting Data Services in Azure**

**Lab 1: Logs, Metrics, and Alerts in Azure Monitor**

**Objective**

Learn how to collect, analyze, and configure alerts using Azure Monitor.

**Prerequisites**

* Azure subscription
* At least one deployed service (ADF or Databricks)
* Access to **Azure Monitor** and **Log Analytics**

**Steps**

1. **Navigate to Azure Monitor**
   * In Azure Portal, search for **Azure Monitor**.
   * Overview shows **Metrics, Logs, Alerts, Workbooks**.
2. **Enable Diagnostic Settings**
   * Select a resource (ADF or Databricks).
   * Go to **Diagnostic settings → Add diagnostic setting**.
   * Send data to **Log Analytics workspace**.
3. **View Metrics**
   * From Azure Monitor → Metrics.
   * Choose a resource, e.g., ADF pipeline.
   * Metric examples: Pipeline Runs, Succeeded, Failed.
   * Use chart to visualize failure trends.
4. **Create an Alert Rule**
   * In Azure Monitor → Alerts → New alert rule.
   * Scope: Select your resource.
   * Condition: Metric signal (e.g., Pipeline Failures > 5).
   * Action: Create an **Action Group** (email, Teams, SMS).
   * Review and Create.

**Result:** You can now view logs, analyze metrics, and receive alerts for failures.

**Lab 2: Setting up Application Insights for Data Services**

**Objective**

Enable and configure Application Insights to monitor performance and usage of a data service.

**Prerequisites**

* Application Insights resource created in Azure
* Data service like ADF pipeline or Databricks cluster

**Steps**

1. **Create Application Insights Resource**
   * Azure Portal → **Create a resource → Application Insights**.
   * Choose Subscription, Resource Group, Name, and Region.
   * Select **Workspace-based mode**.
2. **Link Application Insights**
   * For **ADF**: In Data Factory → Diagnostic settings → Add Application Insights.
   * For **Databricks**: Enable logging to Application Insights via cluster configuration.
3. **Collect Telemetry**
   * Enable automatic collection of:
     + Requests
     + Dependencies
     + Exceptions
     + Performance counters
4. **Analyze Data**
   * Navigate to Application Insights → **Performance blade**.
   * Check request success rate, failures, and latency.
   * Use **Application Map** to visualize dependencies.
5. **Custom Events (Optional)**
   * Use SDK (opencensus, applicationinsights for Python) to log custom events.

**Result:** Application Insights now collects telemetry and gives you performance visibility.

**Lab 3: Diagnosing Performance Issues in ADF and Databricks**

**Objective**

Learn how to troubleshoot slow or failed pipelines in **ADF** and **Databricks**.

**Steps**

**Part A: Azure Data Factory (ADF)**

1. Navigate to **ADF Studio → Monitor → Pipeline runs**.
2. Filter by **Failed pipelines**.
3. Click on a pipeline → review **Activity run details**.
   * Duration
   * Error messages
   * Input/output datasets
4. Check linked service configurations (e.g., wrong connection string).
5. Enable **Diagnostic logs** → Send to Log Analytics.
6. In **Log Analytics**, run query:
7. ADFPipelineRun
8. | where Status == "Failed"
9. | summarize count() by PipelineName

**Part B: Azure Databricks**

1. Navigate to **Databricks Workspace → Jobs → View Job Runs**.
2. For a failed run, open **Run details**.
3. Review:
   * **DAG (Execution graph)** for bottlenecks.
   * **Driver/Executor logs**.
   * Cluster utilization (CPU, Memory).
4. Use **Ganglia/Spark UI** to analyze stages.
   * Check for skewed partitions or excessive shuffle.
5. Export diagnostic logs to Azure Monitor for further analysis.

**Result:** You can now diagnose pipeline failures in ADF and performance issues in Databricks.

**Lab 4: Creating a Troubleshooting Playbook for Production Issues**

**Objective**

Document a **standard playbook** for common monitoring and troubleshooting scenarios.

**Steps**

1. **Identify Common Issues**
   * Pipeline failures in ADF.
   * Spark job crashes in Databricks.
   * Slow performance due to cluster scaling.
2. **Define Troubleshooting Steps**  
   Example: **ADF Pipeline Failure**
   * Step 1: Check Alerts in Azure Monitor.
   * Step 2: Review Pipeline run logs in ADF.
   * Step 3: Query Log Analytics for error trends.
   * Step 4: Validate data source/linked services.
   * Step 5: Retry or re-run pipeline.
   * Step 6: Escalate to DevOps/Data Engineering team if unresolved.
3. **Document Escalation Paths**
   * Tier 1: On-call support → Check alerts and logs.
   * Tier 2: Data Engineer → Debug pipeline/job code.
   * Tier 3: Platform/Cloud Engineer → Check cluster/network.
4. **Store Playbook**
   * Use **Azure DevOps Wiki, SharePoint, or Confluence**.
   * Make it accessible to the on-call team.
5. **Automate Common Fixes**
   * Example: Auto-restart a failed Databricks cluster via **Logic App** triggered by an alert.

**Result:** You have a structured troubleshooting playbook to handle recurring issues effectively.