Creating an Azure Workbook with individual tabs for monitoring Azure services like Databricks, Storage Accounts, Data Factory, and Event Hubs involves using **Azure Monitor Workbooks**. These Workbooks provide a flexible platform to visualize data from Azure Monitor logs, metrics, and other sources.

Here’s how you can achieve this step-by-step:

### **Step 1: Define Workbook Tabs for Each Service**

Each tab will:

1. Query relevant Azure Monitor logs or metrics for the respective service.
2. Display the data using visualizations (charts, grids, etc.).

### **Step 2: Example Code for Workbook Creation**

Azure Workbooks can be exported as JSON templates. Below is an example JSON template that you can modify to include tabs for each service:

#### **JSON Template for Azure Workbook**

json

Copy code

{  
 "$schema": "<https://github.com/Microsoft/Application-Insights-Workbooks/blob/master/schema/settings.json>",  
 "version": "Notebook/1.0",  
 "items": [  
 {  
 "type": "textblock",  
 "content": "## Monitoring Overview for Azure Services",  
 "style": "title"  
 },  
 {  
 "type": "tab",  
 "tabs": [  
 {  
 "name": "Databricks",  
 "items": [  
 {  
 "type": "markdown",  
 "content": "### Databricks Logs and Metrics"  
 },  
 {  
 "type": "query",  
 "query": {  
 "version": 2,  
 "query": "AzureDiagnostics | where ResourceType == 'DATBRICKS\_WORKSPACE' | summarize Count = count() by LogLevel",  
 "subscriptions": ["<Your-Subscription-ID>"]  
 },  
 "chartSettings": {  
 "chart": "piechart",  
 "title": "Databricks Log Levels"  
 }  
 }  
 ]  
 },  
 {  
 "name": "Storage Account",  
 "items": [  
 {  
 "type": "markdown",  
 "content": "### Storage Account Metrics"  
 },  
 {  
 "type": "query",  
 "query": {  
 "version": 2,  
 "query": "AzureMetrics | where ResourceType == 'STORAGE\_ACCOUNT' and MetricName == 'Transactions' | summarize TotalTransactions = sum(Total) by bin(TimeGenerated, 1h)",  
 "subscriptions": ["<Your-Subscription-ID>"]  
 },  
 "chartSettings": {  
 "chart": "linechart",  
 "title": "Storage Account Transactions Over Time"  
 }  
 }  
 ]  
 },  
 {  
 "name": "Data Factory",  
 "items": [  
 {  
 "type": "markdown",  
 "content": "### Data Factory Pipeline Runs"  
 },  
 {  
 "type": "query",  
 "query": {  
 "version": 2,  
 "query": "AzureDiagnostics | where ResourceType == 'DATAFACTORY' | summarize Runs = count() by PipelineName, bin(TimeGenerated, 1h)",  
 "subscriptions": ["<Your-Subscription-ID>"]  
 },  
 "chartSettings": {  
 "chart": "barchart",  
 "title": "Pipeline Runs per Hour"  
 }  
 }  
 ]  
 },  
 {  
 "name": "Event Hub",  
 "items": [  
 {  
 "type": "markdown",  
 "content": "### Event Hub Throughput"  
 },  
 {  
 "type": "query",  
 "query": {  
 "version": 2,  
 "query": "AzureMetrics | where ResourceType == 'EVENTHUB' and MetricName == 'IncomingMessages' | summarize TotalMessages = sum(Total) by bin(TimeGenerated, 1h)",  
 "subscriptions": ["<Your-Subscription-ID>"]  
 },  
 "chartSettings": {  
 "chart": "areachart",  
 "title": "Event Hub Incoming Messages Over Time"  
 }  
 }  
 ]  
 }  
 ]  
 }  
 ]  
}

### **Step 3: Deploy the Workbook**

1. Save the JSON template as a file (e.g., AzureWorkbook.json).
2. Use the Azure Portal to create or import the workbook:
   1. Navigate to **Azure Monitor** > **Workbooks**.
   2. Click on **New** > **Import** and upload the JSON file.

### **Explanation of the JSON Structure**

* **Tabs (tab)**: Separate sections for each Azure service (e.g., Databricks, Storage Account).
* **Queries (query)**: Azure Monitor queries for logs or metrics. Replace ResourceType and MetricName with those matching your services.
* **Visualizations (chartSettings)**: Defines how the data will be displayed (e.g., line chart, bar chart, pie chart).
* **Markdown (markdown)**: Adds explanatory text for context.

### **Customizing for Your Needs**

1. Replace <Your-Subscription-ID> with your Azure Subscription ID.
2. Modify queries to reflect your specific use cases, such as:
   1. Databricks cluster utilization.
   2. Data Factory activity runs.
   3. Storage account read/write operations.
   4. Event Hub message processing metrics.

Would you like further guidance on creating or customizing the workbook?

4o

##### **You said:**

yes

###### **ChatGPT said:**

ChatGPT

Great! Let's go step-by-step to customize and optimize your Azure Workbook for logging and monitoring Databricks, Storage Accounts, Data Factory, and Event Hub.

### **Step 1: Identify Data Sources**

For each service, determine the type of logs or metrics you want to monitor. Here's a quick guide:

* **Databricks**:
  + Workspace logs from **AzureDiagnostics**.
  + Metrics for job runtimes, cluster utilization, and errors.
* **Storage Account**:
  + Metrics like **Transactions**, **Egress**, and **Ingress** from **AzureMetrics**.
* **Data Factory**:
  + Activity run logs and pipeline execution details from **AzureDiagnostics**.
* **Event Hub**:
  + Metrics such as **IncomingMessages** and **ThrottledRequests** from **AzureMetrics**.

### **Step 2: Define Custom Queries**

Here are sample KQL queries for each service that you can plug into your workbook:

#### **Databricks: Cluster Usage and Errors**

kql

Copy code

AzureDiagnostics  
| where ResourceType == "DATBRICKS\_WORKSPACE"  
| summarize Count = count() by LogLevel, bin(TimeGenerated, 1h)  
| order by LogLevel desc

#### **Storage Account: Transactions Over Time**

kql

Copy code

AzureMetrics  
| where ResourceType == "STORAGE\_ACCOUNT" and MetricName == "Transactions"  
| summarize TotalTransactions = sum(Total) by bin(TimeGenerated, 1h)

#### **Data Factory: Pipeline Success and Failures**

kql

Copy code

AzureDiagnostics  
| where ResourceType == "DATAFACTORY"  
| summarize Success = countif(ActivityStatus == "Succeeded"), Failures = countif(ActivityStatus == "Failed") by PipelineName, bin(TimeGenerated, 1h)

#### **Event Hub: Incoming Messages**

kql

Copy code

AzureMetrics  
| where ResourceType == "EVENTHUB" and MetricName == "IncomingMessages"  
| summarize TotalMessages = sum(Total) by bin(TimeGenerated, 1h)

### **Step 3: Customize Workbook JSON**

Modify the JSON template with:

1. **Query Tweaks**: Replace queries with those relevant to your services.
2. **Chart Settings**: Choose visualizations (e.g., pie charts, line charts).
3. **Filters**: Add dropdown filters (e.g., by service name, region, time range).

#### **Example Filtered Tab for Storage Account**

json

Copy code

{  
 "name": "Storage Account",  
 "items": [  
 {  
 "type": "markdown",  
 "content": "### Storage Account Monitoring"  
 },  
 {  
 "type": "parameters",  
 "parameters": [  
 {  
 "type": "Dropdown",  
 "name": "StorageAccountName",  
 "values": [  
 { "value": "storage1", "label": "Storage Account 1" },  
 { "value": "storage2", "label": "Storage Account 2" }  
 ],  
 "defaultValue": "storage1"  
 }  
 ]  
 },  
 {  
 "type": "query",  
 "query": {  
 "version": 2,  
 "query": "AzureMetrics | where ResourceType == 'STORAGE\_ACCOUNT' and Resource == {StorageAccountName} and MetricName == 'Transactions' | summarize TotalTransactions = sum(Total) by bin(TimeGenerated, 1h)",  
 "subscriptions": ["<Your-Subscription-ID>"]  
 },  
 "chartSettings": {  
 "chart": "linechart",  
 "title": "Storage Account Transactions"  
 }  
 }  
 ]  
}

### **Step 4: Add Interactive Features**

* **Time Range Picker**: Add a global time picker for consistency.
* **Dynamic Filters**: Allow users to select specific services, regions, or instances dynamically.

#### **Global Time Picker Example**

json

Copy code

{  
 "type": "timePicker",  
 "name": "TimeRange",  
 "value": {  
 "durationMs": 86400000, // Default to last 24 hours  
 "mode": "relative"  
 }  
}

### **Step 5: Deploy and Test the Workbook**

1. **Import Workbook**:
   1. Navigate to **Azure Monitor** > **Workbooks**.
   2. Click **New** > **Import** and upload your JSON template.
2. **Test Queries and Visualizations**:
   1. Check each tab for proper data retrieval and visualization.
   2. Adjust queries or filters as needed.
3. **Save and Share**:
   1. Save the workbook and share it with team members.
   2. Assign appropriate roles (e.g., Reader or Contributor).

### **Optional Enhancements**

1. **Integrate Alerts**:
   1. Set up Azure Monitor alerts to trigger notifications when thresholds (e.g., error counts) are breached.
2. **Embed in Dashboards**:
   1. Add the workbook or specific visualizations to Azure Dashboards for broader visibility.
3. **Role-Based Access**:
   1. Use Azure RBAC to control access to sensitive logs or metrics.

Would you like to focus on customizing specific tabs, or do you need help with additional Azure Monitor features?

4o

##### **You said:**

yes

###### **ChatGPT said:**

ChatGPT

Let's focus on specific aspects to refine your Azure Workbook for monitoring Databricks, Storage Accounts, Data Factory, and Event Hub. Let me know which of the following you’d like to focus on:

1. **Customizing Specific Tabs**:
   1. Enhance queries for better insights (e.g., show Databricks cluster runtimes, Event Hub throughput trends).
   2. Add interactivity (filters by resource name, region, or other properties).
   3. Optimize visualizations (line charts, bar charts, tables).
2. **Integrating Alerts**:
   1. Set up notifications for anomalies (e.g., high error rates, throughput limits).
3. **Embedding Workbooks**:
   1. Add the workbook or specific tabs into Azure Dashboards.
4. **Securing Workbook Access**:
   1. Use role-based access control (RBAC) to limit visibility.
5. **Advanced Workbook Features**:
   1. Add custom parameters, combine multiple datasets, or use advanced calculations.

Please share your preference, and I’ll provide tailored guidance!

Let's proceed with creating a **comprehensive Azure Workbook** for logging and monitoring Databricks, Storage Accounts, Data Factory, and Event Hub, ensuring full customization and interactivity. Here's an end-to-end approach:

### **Step 1: Define Tabs and Data Sources**

Each service will have its tab:

1. **Databricks**: Monitor cluster runtimes, job errors, and utilization.
2. **Storage Account**: Track read/write transactions and throughput.
3. **Data Factory**: Pipeline runs, success/failure rates, and execution times.
4. **Event Hub**: Incoming/processed messages, throttling, and latency.

### **Step 2: JSON Template for the Workbook**

Here is the full JSON template that includes all tabs, dynamic filters, and interactive visualizations:

json

Copy code

{  
 "$schema": "<https://github.com/Microsoft/Application-Insights-Workbooks/blob/master/schema/settings.json>",  
 "version": "Notebook/1.0",  
 "items": [  
 {  
 "type": "textblock",  
 "content": "## Azure Services Monitoring Dashboard",  
 "style": "title"  
 },  
 {  
 "type": "timePicker",  
 "name": "TimeRange",  
 "value": {  
 "durationMs": 86400000,  
 "mode": "relative"  
 }  
 },  
 {  
 "type": "tab",  
 "tabs": [  
 {  
 "name": "Databricks",  
 "items": [  
 {  
 "type": "markdown",  
 "content": "### Databricks Monitoring"  
 },  
 {  
 "type": "query",  
 "query": {  
 "version": 2,  
 "query": "AzureDiagnostics | where ResourceType == 'DATBRICKS\_WORKSPACE' | summarize ErrorCount = countif(Level == 'Error'), WarningCount = countif(Level == 'Warning') by bin(TimeGenerated, 1h)",  
 "subscriptions": ["<Your-Subscription-ID>"]  
 },  
 "chartSettings": {  
 "chart": "barchart",  
 "title": "Databricks Error and Warning Counts"  
 }  
 }  
 ]  
 },  
 {  
 "name": "Storage Account",  
 "items": [  
 {  
 "type": "markdown",  
 "content": "### Storage Account Metrics"  
 },  
 {  
 "type": "query",  
 "query": {  
 "version": 2,  
 "query": "AzureMetrics | where ResourceType == 'STORAGE\_ACCOUNT' and MetricName == 'Transactions' | summarize TotalTransactions = sum(Total) by bin(TimeGenerated, 1h)",  
 "subscriptions": ["<Your-Subscription-ID>"]  
 },  
 "chartSettings": {  
 "chart": "linechart",  
 "title": "Storage Transactions Over Time"  
 }  
 }  
 ]  
 },  
 {  
 "name": "Data Factory",  
 "items": [  
 {  
 "type": "markdown",  
 "content": "### Data Factory Monitoring"  
 },  
 {  
 "type": "query",  
 "query": {  
 "version": 2,  
 "query": "AzureDiagnostics | where ResourceType == 'DATAFACTORY' | summarize Success = countif(ActivityStatus == 'Succeeded'), Failures = countif(ActivityStatus == 'Failed') by PipelineName, bin(TimeGenerated, 1h)",  
 "subscriptions": ["<Your-Subscription-ID>"]  
 },  
 "chartSettings": {  
 "chart": "stackedbarchart",  
 "title": "Pipeline Success vs Failures"  
 }  
 }  
 ]  
 },  
 {  
 "name": "Event Hub",  
 "items": [  
 {  
 "type": "markdown",  
 "content": "### Event Hub Monitoring"  
 },  
 {  
 "type": "query",  
 "query": {  
 "version": 2,  
 "query": "AzureMetrics | where ResourceType == 'EVENTHUB' and MetricName == 'IncomingMessages' | summarize TotalMessages = sum(Total) by bin(TimeGenerated, 1h)",  
 "subscriptions": ["<Your-Subscription-ID>"]  
 },  
 "chartSettings": {  
 "chart": "areachart",  
 "title": "Event Hub Incoming Messages Over Time"  
 }  
 }  
 ]  
 }  
 ]  
 }  
 ]  
}

### **Step 3: Deploy Workbook in Azure Portal**

1. **Import Workbook**:
   1. Navigate to **Azure Monitor > Workbooks**.
   2. Click **New > Import** and upload the JSON file.
2. **Validate Queries**:
   1. Ensure data for each tab loads correctly.
   2. Adjust resource types or metrics as needed.
3. **Save and Share**:
   1. Save the workbook.
   2. Assign roles (Reader, Contributor) for access control.

### **Step 4: Enhance Features**

* **Add Filters**: Enable users to select resources dynamically.
* **Alerts**: Configure alert rules in Azure Monitor for anomalies (e.g., high error counts).
* **Embed in Dashboards**: Use Azure Dashboards to centralize visibility.

Would you like assistance with:

1. Testing the queries in Log Analytics?
2. Adding more interactive features (like dropdown filters)?
3. Configuring alerts or embedding into dashboards?

Azure Event Hub supports two protocols for message ingestion: **AMQP (Advanced Message Queuing Protocol)** and **Kafka** (via the Kafka API). The choice between these depends on the use case, existing infrastructure, and desired functionality.

### **AMQP**

1. **Overview**:
   1. AMQP is a standardized, open protocol for message-oriented middleware.
   2. It is natively supported by Azure Event Hubs, providing robust, reliable, and interoperable messaging.
2. **Key Features**:
   1. **Efficient Protocol**: Lightweight and optimized for high throughput and low latency.
   2. **Native Integration**: Fully compatible with Azure Event Hub’s native features, such as partitions, checkpoints, and consumer groups.
   3. **Flexible Client SDKs**: Supports the Azure SDKs for integration.
3. **When to Use**:
   1. **You are building new applications in Azure**: AMQP is the default protocol for Event Hubs, so it’s easier to configure and manage.
   2. **You need advanced Event Hub features**: Such as auto-incrementing offsets, checkpointing, or deep integration with other Azure services (Stream Analytics, Functions, etc.).
   3. **Protocol Independence**: Works across systems where AMQP clients are available.
4. **Example**:
   1. A web application sending telemetry data to Event Hubs using the Azure SDK with AMQP protocol to leverage Azure-native functionality and integrate with Stream Analytics.

### **Kafka**

1. **Overview**:
   1. Azure Event Hubs supports **Kafka APIs**, allowing existing Kafka-based applications to use Event Hubs as a backend without modification.
   2. Offers a Kafka endpoint, but Event Hubs itself is not a Kafka broker.
2. **Key Features**:
   1. **Seamless Integration**: Existing Kafka producers and consumers can interact with Event Hubs.
   2. **Compatibility**: Supports Kafka APIs (e.g., Produce and Consume) and integrates with tools like Kafka Connect and Kafka Streams.
   3. **Scalability**: Leverages Event Hubs’ scalability and partitioning.
3. **When to Use**:
   1. **Existing Kafka Applications**: If you already use Kafka, migrating to Event Hubs is straightforward while retaining your existing tools.
   2. **Hybrid Environments**: Integrate Kafka-based workloads with Azure services without managing a separate Kafka cluster.
   3. **Cost Optimization**: Offload Kafka infrastructure to Azure while maintaining compatibility with Kafka APIs.
4. **Example**:
   1. A company with an existing on-premises Kafka setup uses Kafka Connect to stream logs to Azure Event Hubs for analytics in Azure Synapse.

### **Comparison: AMQP vs. Kafka**

|  |  |  |
| --- | --- | --- |
| **Feature** | **AMQP** | **Kafka API** |
| **Native to Event Hub** | Yes | No (emulates Kafka interface) |
| **Best for** | Azure-native workloads | Migrating or extending Kafka apps |
| **Tooling** | Azure SDKs, Event Hub features | Kafka client libraries, Kafka tools |
| **Advanced Features** | Full Event Hubs features supported | Limited by Kafka API compatibility |
| **Complexity** | Simpler for Azure-centric apps | Easier for existing Kafka setups |

### **Which One is Better?**

* **Choose AMQP**:
  + You are starting fresh in Azure and want to leverage full Event Hubs capabilities.
  + You are integrating with Azure services like Stream Analytics or Functions.
* **Choose Kafka**:
  + You have existing Kafka applications or clients and want to migrate without significant changes.
  + You want to extend on-premises Kafka workloads to Azure without managing additional infrastructure.

Both protocols are highly efficient, but the decision should be based on your existing ecosystem and long-term strategy.