Based on the requirements of your project and the resources provided, let's outline a complete solution using Microsoft Azure and other tools to build a data pipeline with the chosen data source, which in this case can be either the Wikipedia recent changes stream or the New York Taxi trip records. Here's how you can structure your data pipeline:

### Data Pipeline Overview

1. \*\*Data Ingestion\*\*: Fetch data from the Wikipedia recent changes API or the NYC Taxi trip records through Azure Event Hubs.

2. \*\*Data Storage\*\*: Store the ingested data into Azure Data Lake for raw storage and Azure SQL Database for structured storage and easy querying.

3. \*\*Data Processing\*\*: Use Azure Databricks for processing and transforming the data.

4. \*\*Data Visualization\*\*: Connect the processed data to Tableau for visualization.

5. \*\*Monitoring\*\*: Implement Azure Monitor and Azure Dashboards to monitor the pipeline's performance and health.

### Step-by-Step Implementation

#### 1. Data Ingestion

- \*\*Wikipedia Recent Changes Stream\*\*:

- Use Python scripts to subscribe to the event stream using SSE (Server-Sent Events) and ingest data in real-time.

- Push the data into Azure Event Hubs, which serves as the entry point for real-time data ingestion.

- \*\*NYC Taxi Data\*\*:

- Automate data fetches for the monthly updated Parquet files using Azure Functions.

- Upload the data to Azure Event Hubs.

#### 2. Data Storage

- Use Azure Data Lake to store the raw data in its native format (JSON for Wikipedia, Parquet for taxi data).

- Utilize Azure SQL Database to store processed and structured data, which provides robust querying capabilities and integration with Azure services.

#### 3. Data Processing

- Deploy Azure Databricks to read the raw data from Azure Data Lake.

- Process and transform the data using PySpark within Databricks notebooks. This might involve cleaning data, aggregating statistics, and preparing the data for analysis and reporting.

- Save the transformed data back to Azure SQL Database for querying and analysis.

#### 4. Data Visualization

- Connect Tableau to Azure SQL Database using the built-in connector.

- Create interactive dashboards in Tableau to visualize key metrics such as edit types and user activity for Wikipedia, or trip duration, fare amounts, and pickup-dropoff heatmaps for taxi data.

#### 5. Monitoring and Dashboarding

- Set up Azure Monitor to track the performance and health of the data pipeline components like Event Hubs, Azure SQL Database, and Databricks.

- Use Azure Dashboards to create a visual interface displaying key performance indicators and logs from Azure Monitor.

### Code and Configuration

- \*\*Python scripts\*\* for API data fetching and initial data processing.

- \*\*Azure ARM templates or Terraform scripts\*\* for infrastructure setup.

- \*\*Databricks notebooks\*\* (PySpark) for detailed data processing.

- \*\*Tableau workbook files\*\* for visualization.

### Challenges and Solutions

- \*\*Data Volume and Velocity\*\*: Managing high throughput and data volume especially with the Wikipedia stream. This can be addressed by scaling up Azure Event Hubs and ensuring Databricks clusters are adequately sized.

- \*\*Data Quality\*\*: The taxi data may contain anomalies or incomplete records. Implement data quality checks within Databricks.

- \*\*Real-Time Processing\*\*: For Wikipedia data, ensure the processing latency is minimal to allow near real-time analysis.

This structured approach should provide a comprehensive pathway from data ingestion to visualization, incorporating robust Azure services and external tools like Tableau for a complete data pipeline solution.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**With limits’’’**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

With your current Azure student account limits in mind, we can optimize the use of Azure services to fit within these constraints while still building an effective data pipeline. Here’s a refined approach:

### Adjusted Data Pipeline Architecture:

1. \*\*Data Ingestion\*\*:

- For the \*\*Wikipedia Recent Changes Stream\*\*, set up a small-scale Azure Event Hub to handle the data stream. This will involve configuring the service to handle a feasible volume of data under the free tier limits.

- For the \*\*NYC Taxi Data\*\*, considering the volume of the files, use Azure Blob Storage to store the monthly Parquet files. Use an Azure Function to automate the downloading and pushing of these files into Azure Blob Storage.

2. \*\*Data Storage\*\*:

- Utilize \*\*Azure Blob Storage\*\* for raw data storage, which provides 5 GB of LRS Hot Block Blob storage under the free tier. This is adequate for initial stages or smaller datasets.

- For structured data storage that enables querying, use the \*\*Azure SQL Database\*\*. You have up to 32 GB per month for backups and stored data, which should be sufficient for structured data after processing.

3. \*\*Data Processing\*\*:

- Deploy \*\*Azure Databricks\*\* using the free DBU credits available under your academic account, if any. If not, processing can be done on a small-scale cluster or alternatively, for very light processing tasks, consider using Azure Functions to run Python scripts.

4. \*\*Data Visualization\*\*:

- Connect \*\*Tableau\*\* to Azure SQL Database. Given your account limitations, ensure that you aggregate and filter data during processing to avoid storing unnecessary volumes into your SQL database.

5. \*\*Monitoring and Dashboarding\*\*:

- Set up \*\*Azure Monitor\*\* to keep track of operations and usage to stay within the free tier limits. Use Azure Dashboards to create a visual monitoring tool for the health and metrics of your services.

### Considerations and Optimizations:

- \*\*Event Hub Usage\*\*: Configure the throughput units and partitions in Azure Event Hubs to manage data ingestion without exceeding free tier limitations.

- \*\*Data Processing\*\*: Depending on your Azure Databricks access in the student account, consider lighter processing tasks or pre-aggregate data during ingestion to reduce load.

- \*\*SQL Database Management\*\*: Given the data storage limits, regularly archive or clean up old data in Azure SQL Database to prevent exceeding storage limits.

- \*\*Cost Monitoring\*\*: Regularly check Azure's cost management tools to ensure you are not exceeding free service limits. Be ready to adjust configurations or deactivate services temporarily if needed.

This setup should allow you to run a basic version of your data pipeline within the constraints of a typical Azure student account. Keep in mind that as your requirements grow, you might need to either optimize further or consider funding options to increase your service limits.