2. Environment setup

Create a new resource group

2.1 Create storage account - stmasterclass01

2.2 Create storage account - dlsmasterclass01

1. Advanced --> Enable "Hierarchical namespace"

2.3 Create Azure Synapse Workspace - asaworkspacemasterclass01

1. Create new file system --> "defaultfs"

2. Enable "Assign myself the Storage Blob Data Contributor role on the Data Lake Storage Gen2 account 'dlsmasterclass01'."

3. Security:

    - Admin username: asa.sql.admin
    - Password: ***1!s

4. Networking:

    - Disable "Allow connections from all IP addresses"

2.4 Create Key Vault - akv-masterclass-01

2.5 SQL pool - SQLPool01

2.6 Spark pool - SparkSmall01

- Small
- Max 4 nodes
- Check Storage Blob Data Contributor (workspace MSI and potential users on the storage account)

2.7 Azure Synapse Workspace – asaworkspacemasterclass01

- Configure firewall for Azure Services and Client IP
- Set “Owner” role on resource, if not already having it

****

3. Configuring environment

3.1 Azure Key Vault – akv-masterclass-01

- Allow Synapse Workspace to get secrets
- Create new secret with password for asa.sql.admin – SQL-ADMINUSER-ASA - ***1!s
- Create new secret for dlsmasterclass01 (Key1) – ADLS-masterclass01-Key1

3.2.1 Azure Data Lake - dlsmasterclass01

- Create containers for your data: raw, stage, dev and prod

3.2.2 Azure Synapse Workspace – asaworkspacemasterclass01

- Show where we are going to get our data from - https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- Create HTTP linked service for nyctlc yellow trip data – nyctlc_tripdata
    - Base url: https://s3.amazonaws.com/nyc-tlc/
    - Authentication: Anonymous
- Create AKV linked Service - akvmasterclass01
- Create data lake dataset – binary_dlsmasterclass01_raw_parameterized
    - Data lake → binary → container: raw
    - Parameters:
        - DirectoryPath → directory
        - FileName → file
- Create nyc_tlc dataset → binary_nyctlc_parameterized
    - HTTP → Binary
    - Parameters:
        - RelativeUrl → relative url
- Create pipeline – NYCTLC_to_ADLS
    - Create ForEach-activity – For each month – NYCTLC to ADLS
    - Parameters:
        - Year – string
            - Default value: 2020
        - ArrayMonth – array
            - Default value: ["01","02","03","04","05","06","07","08","09","10","11","12"]
    - Variables:
        - RelativeUrl
        - ADLSFileName
        - DirectoryPath
    - Settings:
        - Sequential: On
        - Items: parameter ArrayMonth

        - Activities
            - Set variable – “Set relative url”
                - Name: RelativeUrl
                - Value: @concat('trip+data/yellow_tripdata_', pipeline().parameters.Year, '-', item(), '.csv')
            - Set variable – “Set ADLS file name”
                - Name: ADLSFileName
                - Value: @concat('yellow_tripdata_', pipeline().parameters.Year, '-', item(), '.csv')
            - Set variable – “Set directory path”
                - Name: DirectoryPath
                - Value: @concat('nyctlc/tripdata/', pipeline().parameters.Year, '/', item())
            - Copy data – “NYCTLC_HTTP_to_ADLS”
                - Source: binary_nyctlc_ parameterized
                    - Dataset properties: RelativeUrl
                    - Request method: GET
                - Sink: binary_dlsmasterclass01_raw_parameterized
                    - Dataset properties: DirectoryPath and ADLSFileName
        - Debug – will most likely take 60+ minutes
- Create pipeline – NYCTLC_LOOKUP_to_ADLS
    - Copy Data activity – NYCTLC lookup data to ADLS
        - Source: binary_nyctlc_ parameterized
            - Dataset properties: RelativeUrl
        - Sink: binary_dlsmasterclass01_raw_parameterized
            - Dataset properties: DirectoryPath and FileName
        - Variables:
            - RelativeUrl: misc/taxi+_zone_lookup.csv
            - DirectoryPath: nyctlc/misc
            - FileName: taxi+_zone_lookup.csv
    - Debug

3.2.3 Azure Synapse Workspace – asaworkspacemasterclass01

- Create data lake dataset – csv_dlsmasterclass01_raw_parameterized
    - Data lake → binary → choose a file to obtain the schema and then keep just “raw”
    - First row as header
- Create schema nyctlc
- Create table in SQL Pool

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [nyctlc].[YellowTaxiTripRecords]
(
```

```
    [VendorID] char(1)  NULL, --
A code indicating the TPEP provider that provided the record. 1= Creative Mobi
le Technologies, LLC; 2= VeriFone Inc
    [Pickup_Datetime] [datetime2](0)  NULL, --
The date and time when the meter was engaged.
    [Dropoff_Datetime] [datetime2](0)  NULL, --
The date and time when the meter was disengaged.
    [PassengerCount] varchar(5) null, --
The number of passengers in the vehicle. This is a driver-entered value
    [TripDistance] DECIMAL(10,2)  NULL, --
The elapsed trip distance in miles reported by the taximeter.
    -- excluded [RatecodeID] char(1)  NULL, --
The final rate code in effect at the end of the trip. 1= Standard rate, 2=JFK,
 3=Newark, 4=Nassau or Westchester, 5=Negotiated fare, 6=Group ride
    -- excluded [store_and_fwd_flag], -
- This flag indicates whether the trip record was held in vehicle memory befor
e sending to the vendor, aka “store and forward,” because the vehicle did not
have a connection to the server. Y= store and forward trip, N= not a store and
 forward trip
    [PickupLocationID] varchar(5)  NULL, --
TLC Taxi Zone in which the taximeter was engaged
    [DropoffLocationID] varchar(5)  NULL, --
TLC Taxi Zone in which the taximeter was disengaged
    [PickupBorough] varchar(50) NULL, --borough, from lookup table
    [PickupZone] varchar(100) NULL, --zone, from lookup table
    [DropoffBorough] varchar(50) NULL, --borough, from lookup table
    [DropoffZone] varchar(100) NULL, --zone, from lookup table
    [PaymentType] varchar(50)  NULL, --
A numeric code signifying how the passenger paid for the trip. 1= Credit card,
 2= Cash, 3= No charge, 4= Dispute, 5= Unknown, 6= Voided trip
    [FareAmount] DECIMAL(10,2)  NULL, --The time-and-
distance fare calculated by the meter.
    [ExtraAmount] DECIMAL(10,2)  NULL, --
Miscellaneous extras and surcharges. Currently, this only includes the $0.50 a
nd $1 rush hour and overnight charges.
    [MtaTaxAmount] DECIMAL(10,2)  NULL, --
$0.50 MTA tax that is automatically triggered based on the metered rate in use
.
    [TipAmount] DECIMAL(10,2)  NULL, --
Tip amount – This field is automatically populated for credit card tips. Cash
tips are not included.
    [TollsAmount] DECIMAL(10,2)  NULL, --
Total amount of all tolls paid in trip.
    [ImprovementSurchargeAmount] DECIMAL(10,2)  NULL, --
$0.30 improvement surcharge assessed trips at the flag drop. The improvement s
urcharge began being levied in 2015.
    [TotalAmount] DECIMAL(10,2)  NULL --
The total amount charged to passengers. Does not include cash tips.
    -- excluded [congestion_surcharge] --no dictionary?
```

```
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
GO

***
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [nyctlc].[SummaryYellowTaxiTripRecords]
(
    [TripDate] [date]  NULL,
    [PickupBorough] [varchar](25)  NULL,
    [PickupZone] [varchar](50)  NULL,
    [DropoffBorough] [varchar](25)  NULL,
    [DropoffZone] [varchar](50)  NULL,
    [TotalTripCount] [int]  NULL,
    [TotalPassengerCount] [int]  NULL,
    [TotalDistanceTravelled] [decimal](18,2)  NULL,
    [TotalFareAmount] [decimal](18,2)  NULL,
    [TotalTipAmount] [decimal](18,2)  NULL,
    [TotalTripAmount] [decimal](18,2)  NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
```

- Create master key – SQLPool01
- Create dataset Azure Synapse Analytics – sqlpool01_nyctlc_summaryyellowtaxitriprecords
    - Parameters: DBName
    - Set table: nyctlc.SummaryYellowTaxiTripRecords
- Create dataset Azure Synapse Analytics – sqlpool01_nyctlc_yellowtaxitriprecords
    - Parameters: DBName
    - Set table: nyctlc.YellowTaxiTripRecords
- Create data flow - NYCTLC_ADLS_to_SQLPOOL01
    - Two source inputs: csv_dlsmasterclass_raw_parameterized
        - RawNyctlcTripdata
            - Wildcard paths: nyctlc/tripdata/2020/*/*.csv
            - Import projection
                - Use sample file

            - Set all data types to string
        - RawNyctlcLookupdata
            - Wildcard paths: nyctlc/misc/taxi+_zone_lookup.csv
            - Import projection
                - Use sample file
                - Set all data types to string
    - Select – SelectNycctlcTripdata
        - VendorID → VendorID
        - Tpep_pickup_datetime → Pickup_Datetime
        - tpep_dropoff_datetime→ Dropoff_Datetime
        - passenger_count → PassengerCount
        - trip_distance → TripDistance
        - PULocationID → PickupLocationID
        - DOLocationID→ DropoffLocationID
        - payment_type → PaymentType
        - fare_amount → FareAmount
        - extra → ExtraAmount
        - mta_tax → MtaTaxAmount
        - tip_amount→ TipAmount TollsAmount
        - tolls_amount → TollsAmount
        - improvement_surcharge → ImprovementSurchargeAmount
        - total_amount → TotalAmount
    - Derived column – TranslatingNumericColumns
        - PaymentType: case (PaymentType == '1', 'Credit card' , PaymentType == '2', 'Cash' , PaymentType == '3', 'No charge' , PaymentType == '4', 'Dispute' , PaymentType == '5', 'Unknown' , PaymentType == '6', 'Voided trip')
    - EnsuringDataConsistency
        - toDecimal(TripDistance)
        - toDecimal(FareAmount)
        - toDecimal(ExtraAmount)
        - toDecimal(MtaTaxAmount)
        - toDecimal(TipAmount)
        - toDecimal(TollsAmount)
        - toDecimal(ImprovementSurchargeAmount)
        - toDecimal(TotalAmount)
        - toTimestamp(Pickup_Datetime, 'yyyy-MM-dd HH:mm:ss')
        - toTimestamp(Dropoff_Datetime, 'yyyy-MM-dd HH:mm:ss')
    - Join – JoinPickupLocation
        - Left outer
        - PickupLocationID == LocationID
    - Select – SelectPickupInformation
        - Borough → PickupBorough
        - Zone → PickupZone
    - Join – JoinDropoffLocation
        - Left outer
        - DropoffLocationID == LocationID
    - Select – SelectDropoffInformation

        - Borough → DropoffBorough
        - Zone → DropoffZone
    - New branch
    - Sink – yellowtaxitriprecords
        - Dataset: sqlpool01_nyctlc_yellowtaxitriprecords
        - Check mapping
    - Select – SelectAggData
        - Pickup_Datetime → TripDate
        - PassengerCount
        - TripDistance
        - FareAmount
        - TipAmount
        - TotalAmount
        - PickupBorough
        - PickupZone
        - DropoffBorough
        - DropoffZone
    - Derived column – TransformAggData
        - toDate(TripDate)
        - toInteger(PassengerCount)
    - Aggregate – AggregateData
        - Group by
            - TripDate
            - PickupBorough
            - DropoffBorough
            - PickupZone
            - DropoffZone
        - Aggregates
            - TotalTripCount – count()
            - TotalPassengerCount – sum(PassengerCount)
            - TotalDistanceTravelled – sum(TripDistance)
            - TotalTipAmount – sum(TipAmount)
            - TotalFareAmount – sum(FareAmount)
            - TotalTripAmount – sum(TotalAmount)
    - Sink – summaryyellowtaxitriprecords
        - Dataset: sqlpool01_nyctlc_summaryyellowtaxitriprecords
        - Check mapping
- Create Synapse Pipeline – NYCTLC_ADLS_to_SQLPOOL
    - Create linked service for data lake – dlsmasterclass01
        - https://dlsmasterclass02.dfs.core.windows.net/
        - Use Key Vault Secret – ADLS-masterclass01-Key1
    - Activity Data flow
        - Settings:
            - Parameters: SQLPool01
            - Core count: 8 (+ 8 Driver cores)
            - Staging linked service: dlsmasterlass01
            - Staging storage folder: container → stage & directory → nyctlc

3.4 Setting up our BI solution

Open Power BI Desktop

Connect to data source asaworkspacemasterclass01.database.windows.net

Database: SQLPool01

Import (Direct Query will cause problems on date hierarchy)

Microsoft Account – credential

Lag rapport.

Lag roller. Vis f.eks DataAnalystManhattan → [PickupBorough] = "Manhattan"

3.5.1 Security – Synapse Analytics

Preliminaries:

Creating users to test different access

```
create user [CEO] without login;
create user [Manhattan] without login;
create user [Brooklyn] without login;
create user [Queens] without login;
```

We want to give CEO access to all data and the different analysts access to data from their boroughs.

Column-level security:

```
GRANT SELECT ON nyctlc.SummaryYellowTaxiTripRecords ([TripDate], [PickupBorough],
[DropoffBorough], [TotalDistanceTravelled], [TotalTripAmount]) TO Manhattan, Brooklyn,
Queens

EXECUTE AS USER = 'DataAnalystManhatten';
SELECT TOP 100 * FROM nyctlc.SummaryYellowTaxiTripRecords;

EXECUTE AS USER = 'DataAnalystManhatten';
SELECT TOP 100 [TripDate], [PickupBorough], [DropoffBorough],
[TotalDistanceTravelled], [TotalTripAmount] FROM nyctlc.SummaryYellowTaxiTripRecords;

GRANT SELECT ON nyctlc.SummaryYellowTaxiTripRecords TO CEO;

EXECUTE AS USER = 'CEO';
SELECT TOP 100 * FROM nyctlc.SummaryYellowTaxiTripRecords;
```

Row-level security:

```
-- Review any existing security predicates in the database
SELECT * FROM sys.security_predicates
```

```
CREATE SCHEMA Security

CREATE FUNCTION Security.fn_securitypredicate(@Analyst AS sysname)
    RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN SELECT 1 AS fn_securitypredicate_result
    WHERE @Analyst = USER_NAME() OR USER_NAME() = 'CEO'

CREATE SECURITY POLICY NYCTLCSummaryAnalystFilter
ADD FILTER PREDICATE Security.fn_securitypredicate(PickupBorough)
ON nyctlc.SummaryYellowTaxiTripRecords
WITH (STATE = ON);



EXECUTE AS USER = 'Brooklyn';
SELECT TOP 100 [TripDate], [PickupBorough], [DropoffBorough],
[TotalDistanceTravelled], [TotalTripAmount] FROM nyctlc.SummaryYellowTaxiTripRecords;



ALTER SECURITY POLICY NYCTLCSummaryAnalystFilter
WITH (STATE = OFF);

DROP SECURITY POLICY NYCTLCSummaryAnalystFilter;
DROP FUNCTION Security.fn_securitypredicate;
DROP SCHEMA Security;

DROP USER CEO;
DROP USER Manhattan;
DROP USER Brooklyn;
DROP USER Queens;
```

Dynamic data masking:

Demonstrate portal.

```
ALTER TABLE nyctlc.SummaryYellowTaxiTripRecords
ALTER COLUMN [TotalTipAmount] ADD MASKED WITH (FUNCTION = 'default()');

SELECT TOP 100 * FROM nyctlc.SummaryYellowTaxiTripRecords;

ALTER TABLE nyctlc.SummaryYellowTaxiTripRecords
ALTER COLUMN [TotalTipAmount] DROP MASKED;
```

3.6 Monitoring

```
CREATE WORKLOAD CLASSIFIER classifier_name
WITH
    (   WORKLOAD_GROUP = 'name'  --maps the request to a workload group
    ,   MEMBERNAME = 'security_account' --role or user. Determines weighting of
classifiers
[ [ , ] WLM_LABEL = 'label' ]  --label value that a request can be classified against
[ [ , ] WLM_CONTEXT = 'context' ]  --the session context value that a request can be
classified against
[ [ , ] START_TIME = 'HH:MM' ]  --start_time that a request can be classified against
[ [ , ] END_TIME = 'HH:MM' ]  --end_time that a request can be classified against
[ [ , ] IMPORTANCE = { LOW | BELOW_NORMAL | NORMAL | ABOVE_NORMAL | HIGH }]) --the
relative importance of a request. If not specified, the importance setting of the
workload group is used
```

```
[;]

CREATE WORKLOAD GROUP group_name
 WITH
 (   MIN_PERCENTAGE_RESOURCE = value --guaranteed minimum resource allocation
   , CAP_PERCENTAGE_RESOURCE = value --maximum resource utilization
   , REQUEST_MIN_RESOURCE_GRANT_PERCENT = value --minimum amount of resources
allocated per request
  [ [ , ] REQUEST_MAX_RESOURCE_GRANT_PERCENT = value ] --maximum amount of resources
allocated per request
  [ [ , ] IMPORTANCE = { LOW | BELOW_NORMAL | NORMAL | ABOVE_NORMAL | HIGH } ] --
default importance of a request for the workload group. A user can also set importance
at the classifier level, which can override the workload group importance setting.
  [ [ , ] QUERY_EXECUTION_TIMEOUT_SEC = value ] )
  [ ; ]
```

Example for query user:

```
CREATE WORKLOAD CLASSIFIER CEO
WITH   (
       WORKLOAD_GROUP = 'largerc' --could be user-defined workload group
       ,MEMBERNAME = 'david.aas.correia@inmeta.no'
       ,IMPORTANCE = High
       );
```

Example for load user:

```
-- Connect to master
CREATE LOGIN LoaderRC20 WITH PASSWORD = '...1!lrc20';

 -- Connect to the database
 CREATE USER LoaderRC20 FOR LOGIN LoaderRC20;
 GRANT CONTROL ON DATABASE::[SQLPool01] to LoaderRC20;
 EXEC sp_addrolemember 'staticrc20', 'LoaderRC20';
```

See DMVs and Azure portal

```
SELECT * FROM sys.dm_pdw_exec_sessions
```

See the *Monitor* tab for SQL requests, spark application, trigger runs, pipeline runs ++

3.7 Optimizing environment

```
SELECT name, is_auto_create_stats_on
FROM sys.databases

--nyctlc.YellowTaxiTripRecords
CREATE STATISTICS stats_Pickup_Datetime on nyctlc.YellowTaxiTripRecords
(Pickup_Datetime);
CREATE STATISTICS stats_Dropoff_Datetime on nyctlc.YellowTaxiTripRecords
(Dropoff_Datetime);
CREATE STATISTICS stats_PassengerCount on nyctlc.YellowTaxiTripRecords
(PassengerCount);
CREATE STATISTICS stats_TripDistance on nyctlc.YellowTaxiTripRecords (TripDistance);
CREATE STATISTICS stats_PickupBorough on nyctlc.YellowTaxiTripRecords (PickupBorough);
CREATE STATISTICS stats_PickupZone on nyctlc.YellowTaxiTripRecords (PickupZone);
CREATE STATISTICS stats_DropoffBorough on nyctlc.YellowTaxiTripRecords
(DropoffBorough);
```

```
CREATE STATISTICS stats_DropoffZone on nyctlc.YellowTaxiTripRecords (DropoffZone);
CREATE STATISTICS stats_PaymentType on nyctlc.YellowTaxiTripRecords (PaymentType);
CREATE STATISTICS stats_FareAmount on nyctlc.YellowTaxiTripRecords (FareAmount);
CREATE STATISTICS stats_ExtraAmount on nyctlc.YellowTaxiTripRecords (ExtraAmount);
CREATE STATISTICS stats_MtaTaxAmount on nyctlc.YellowTaxiTripRecords (MtaTaxAmount);
CREATE STATISTICS stats_TipAmount on nyctlc.YellowTaxiTripRecords (TipAmount);
CREATE STATISTICS stats_TollsAmount on nyctlc.YellowTaxiTripRecords (TollsAmount);
CREATE STATISTICS stats_ImprovementSurchargeAmount on nyctlc.YellowTaxiTripRecords
(ImprovementSurchargeAmount);
CREATE STATISTICS stats_TotalAmount on nyctlc.YellowTaxiTripRecords (TotalAmount);

--nyctlc.SummaryYellowTaxiTripRecords
CREATE STATISTICS stats_TripDate on nyctlc.SummaryYellowTaxiTripRecords (TripDate);
CREATE STATISTICS stats_PickupBorough on nyctlc.SummaryYellowTaxiTripRecords
(PickupBorough);
CREATE STATISTICS stats_PickupZone on nyctlc.SummaryYellowTaxiTripRecords
(PickupZone);
CREATE STATISTICS stats_DropoffBorough on nyctlc.SummaryYellowTaxiTripRecords
(DropoffBorough);
CREATE STATISTICS stats_DropoffZone on nyctlc.SummaryYellowTaxiTripRecords
(DropoffZone);
CREATE STATISTICS stats_TotalTripCount on nyctlc.SummaryYellowTaxiTripRecords
(TotalTripCount);
CREATE STATISTICS stats_TotalPassengerCount on nyctlc.SummaryYellowTaxiTripRecords
(TotalPassengerCount);
CREATE STATISTICS stats_TotalDistanceTravelled on nyctlc.SummaryYellowTaxiTripRecords
(TotalDistanceTravelled);
CREATE STATISTICS stats_TotalFareAmount on nyctlc.SummaryYellowTaxiTripRecords
(TotalFareAmount);
CREATE STATISTICS stats_TotalTipAmount on nyctlc.SummaryYellowTaxiTripRecords
(TotalTipAmount);
CREATE STATISTICS stats_TotalTripAmount on nyctlc.SummaryYellowTaxiTripRecords
(TotalTripAmount);
```

4.1 Create solution for IoT data

- Create Logic App – la-nysestream-masterclass
- Create integration account – integrationacc-masterclass
- Logic App → Workflow settings → Set integration account
- Recurrence
  - 1 second
- Data Operations → Compose → “Initalize Config Settings”
  - Inputs:

```
{
  "numberOfMessages": 10,
  "priceRangeMax": 100,
  "priceRangeMin": 70,
  "quantityRangeMax": 300,
  "quantityRangeMin": 100,
  "stockTickers": [
    "MSFT",
    "AMZN",
    "GOOGL",
    "FB",
    "TWTR",
    "CRAYON"
  ]
```

```
}
```

- Data Operations → Parse JSON → “Parse Config Setting”
  - Content: Outputs (Initalize Config Settings)
  - Schema:

```
{
    "properties": {
        "numberOfMessages": {
            "type": "integer"
        },
        "priceRangeMax": {
            "type": "integer"
        },
        "priceRangeMin": {
            "type": "integer"
        },
        "quantityRangeMax": {
            "type": "integer"
        },
        "quantityRangeMin": {
            "type": "integer"
        },
        "stockTickers": {
            "items": {
                "type": "string"
            },
            "type": "array"
        }
    },
    "type": "object"
}
```

- Initialize variable → “Initialize messageCount”
  - Name: messageCount
  - Type: Integer
  - Value: 1
- Until – “Until numberOfMessages is achieved”
  - messageCount = rand(1, body('Parse_Config_Settings')?['numberOfMessages'])
    Inline code → JavaScript (preview) → “Generate Stock Trade Message”

```
var configSettings = workflowContext.actions.Initialize_Config_Settings.outputs;

stockTickerIndex = Math.round(Math.random()*100) % configSettings.stockTickers.length;

var stockTrade = { StockTicker: configSettings.stockTickers[stockTickerIndex],
Quantity: Math.round(configSettings.quantityRangeMin +
(configSettings.quantityRangeMax - configSettings.quantityRangeMin)*Math.random()),
Price: configSettings.priceRangeMin + (configSettings.priceRangeMax -
configSettings.priceRangeMin)*Math.random(),
TradeTimestamp: new Date().toJSON()};

return stockTrade;
```

- Capture Stock Trade Message
  - Inputs: Result (Generate Stock Trade Message)
- Create Event Hubs
  - Namespace name: eh-masterclass-01
  - Pricing tier: Standard

        - Scale – Enable “AUTO-INFLATE”, max 3 throughput units
        - Create Event Hub → nysestocktrade
            - Capture ON
                - “Do not emit empty files when no events occur during the Capture time window”
                - Time-window: 1 minutes
                - Container: eh-masterclass
            - Create shared access policy – send – send-nysestocktrade
            - Create shared access policy – listen – listen-nysestocktrade
- Event Hub – Send Event – “Send event to NYSEStockTradeSimm Event Hub”
    - Create connection
        - Connection: nysestocktrade
        - Content: Output (Capture Stock Trade Message)
- Increment variable – “Increment messageCount”
    - Name: messageCount
    - Value: 1

```
CREATE SCHEMA nyse;

CREATE TABLE nyse.StockTradeCompanyReferenceData
(
	StockTicker varchar(10),
	CompanyName varchar(20)
);

INSERT INTO nyse.StockTradeCompanyReferenceData(StockTicker, CompanyName) VALUES
('MSFT', 'Microsoft');
INSERT INTO nyse.StockTradeCompanyReferenceData(StockTicker, CompanyName) VALUES
('FB', 'Facebook');
INSERT INTO nyse.StockTradeCompanyReferenceData(StockTicker, CompanyName) VALUES
('AMZN', 'Amazon');
INSERT INTO nyse.StockTradeCompanyReferenceData(StockTicker, CompanyName) VALUES
('GOOGL', 'Google');
INSERT INTO nyse.StockTradeCompanyReferenceData(StockTicker, CompanyName) VALUES
('TWTR', 'Twitter');
INSERT INTO nyse.StockTradeCompanyReferenceData(StockTicker, CompanyName) VALUES
('CRAYON', 'Crayon');
```

- Create stream input – Event Hub - NYSEStockTrades
- Create reference data input – SQL Database – NYSEStockCompanies
    - User name
    - Password
    - Storage account – stmasterclass01
        - Connection string
    - Refresh periodically: every day
    - Query

```
SELECT
    [StockTicker]
    ,[CompanyName]
FROM [nyse].[StockTradeCompanyReferenceData]
```

- Create Stream Analytics Job – sa-nysestocktrade

- Outputs
    - Power BI – StockTradeByCompany
        - Authentication mode: User Token
        - Group Workspace: My workspace
        - Dataset name: StreamStockTradeByCompany
        - Table name: StreamStockTradeByCompany
    - Power BI – StockTradeTotals
        - Authentication mode: User Token
        - Group Workspace: My workspace
        - Dataset name: StreamStockTradeTotals
        - Table name: StreamStockTradeTotals
- Start job
- Start logic app
- See activity in all applications
- Power BI
    - My Workspace
        - Create dashboard – NYSE Trade Activity
            - Add tile
                - StockTradeTotals
                    - Card
                    - Fields: TotalTradedAmount
                    - Value decimal places: 2
                    - Title: Total Traded Amount
                    - Subtitles: in the last 30 seconds
                - StockTradeTotals
                    - Card
                    - Fields: TotalTradeCount
                    - Value decimal places: 0
                    - Title: Total Trade Count
                    - Subtitles: in the last 30 seconds
                - StockTradeTotals
                    - Line chart
                    - Axis: WindowDateTime
                    - Values: TotalTradeCount
                    - Time window to display: 5 minutes
                    - Value decimal places: 0
                    - Title: Total Trade Count
                    - Subtitles: 5 min history window
                - StockTradeByCompany
                    - Clustered bar chart
                    - Axis: CompanyName
                    - Legend: CompanyName
                    - Values: TradedAmount
                    - Title: Traded Amount by Company
                    - Subtitles: in the last 30 seconds

4.2 ML in Synapse Spark

Share Jupyter notebook

Develop → Import

Run the notebook