Microsoft

Microsoft Game Stack Live

Microsoft

Azure Data Services: Make Better Business Decisions with Gaming Analytics and Data Insights in the Cloud

JaeHoon Ahn

What we can use in Azure Data Services

Operational database

Azure SQL Database
Database as a Service
DBaaS for game service with
high performance and availability;
scalable up to 100TB

Azure Cosmos DB
NoSQL Database
Support high availability and
SLA for performance: <10ms latency

Azure databases for MySQL, PostgreSQL, and MariaDB
Open Source Database
Enterprise-ready, fully managed
database as a service

Azure Data Services

Data warehouses

Azure Synapse Analytics
Query data over PB
Cloud DW supported in
40 Azure regions

Data lakes and analytics

Azure Data Lake Storage Gen2
Data lake for gaming los
Storing various data to analyze
Provide 99.9% SLA

Azure Data Explorer
Log data exploration
Support fast and efficient
exploration of gaming log

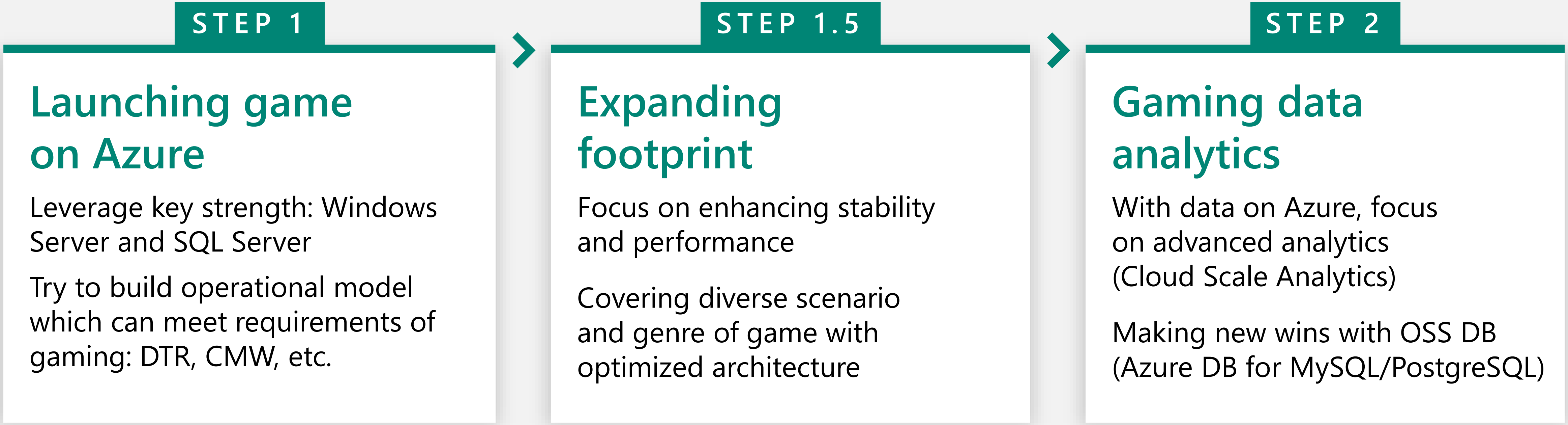
Azure Databricks
Advanced analytics
Provide analytic experience
on Apache Spark with better
performance and usability

Reason over any data, anywhere

Flexibility of choice

Security and performance

How we could be successful in gaming



STEP 1

Launching game on Azure

Leverage key strength: Windows Server and SQL Server

Try to build operational model which can meet requirements of gaming: DTR, CMW, etc.

STEP 1.5

Expanding footprint

Focus on enhancing stability and performance

Covering diverse scenario and genre of game with optimized architecture

STEP 2

Gaming data analytics

With data on Azure, focus on advanced analytics (Cloud Scale Analytics)

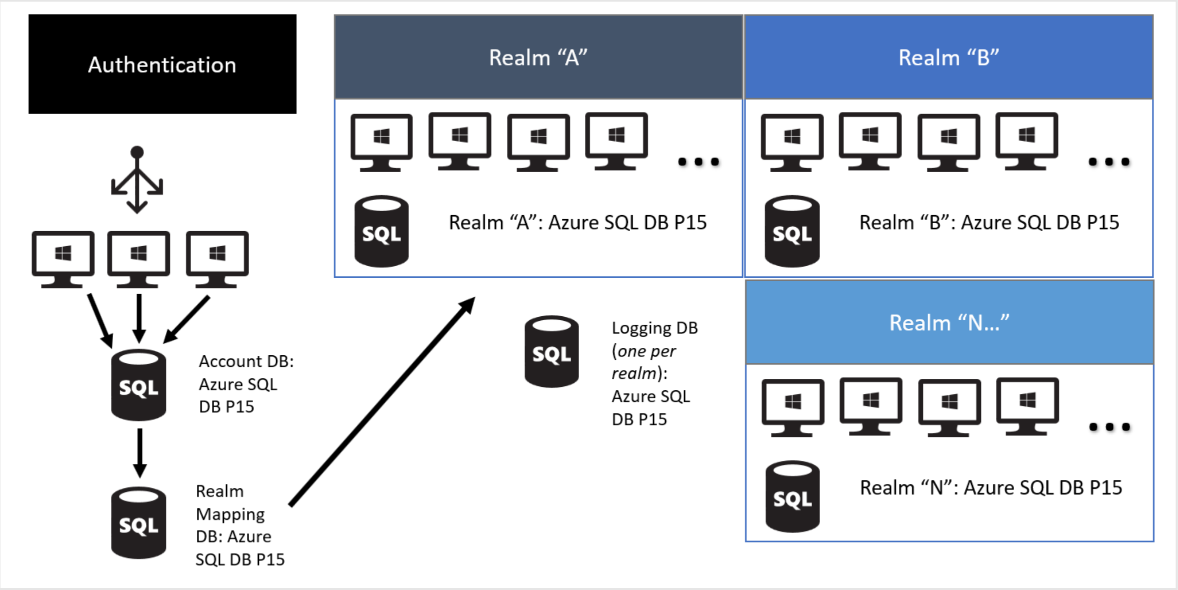
Making new wins with OSS DB (Azure DB for MySQL/PostgreSQL)

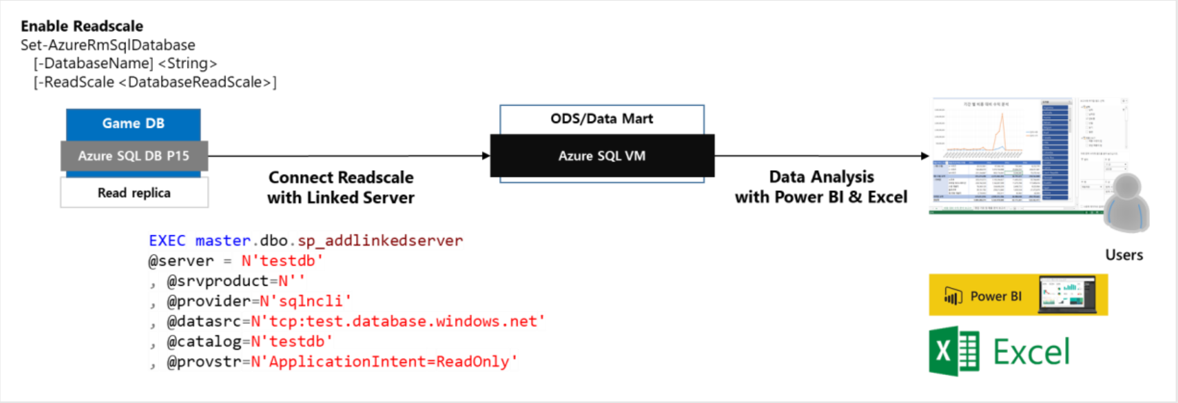
Step 1
Launching game on Azure

Implementing massive scale data tier in Azure

Reference architecture for data tier

- Store player-realm mapping and shard database
- Up to 500,000 queries per second at peak time generating 1GB/second of transaction log during that time
- Use in-memory technology for better performance
  - Use In-memory table with native compiled stored procedure to avoid wait and latch
  - Use TVP with In-memory option instead of temp table for better performance
  - For archiving data, use in-memory column store to provide data compression (> 10x) and high performance for aggregation queries
- Readscale for load balancing read-only workload
  - Read and aggregate data without load in primary game database


Authentication
Account DB: Azure SQL DB P15
Realm Mapping DB: Azure SQL DB P15
Realm "A"
Realm "A": Azure SQL DB P15
Realm "B"
Realm "B": Azure SQL DB P15
Logging DB (*one per realm*): Azure SQL DB P15
Realm "N…"
Realm "N": Azure SQL DB P15
SQL
…



Enable Readscale

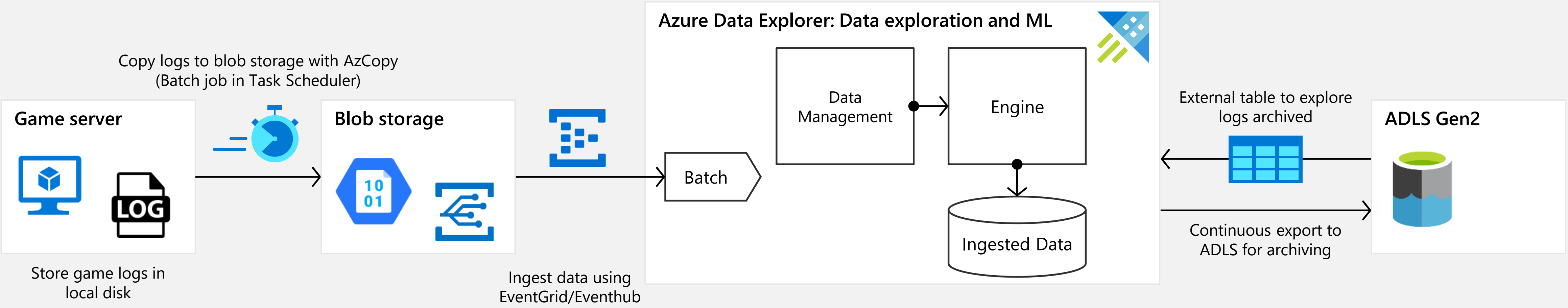
```
Set-AzureRmSqlDatabase
  [-DatabaseName] <String>
  [-ReadScale <DatabaseReadScale>]
```

```
EXEC master.dbo.sp_addlinkedserver
@server = N'testdb'
, @srvproduct=N''
, @provider=N'sqlncli'
, @datasrc=N'tcp:test.database.windows.net'
, @catalog=N'testdb'
, @provstr=N'ApplicationIntent=ReadOnly'
```

More efficient way to store gaming logs

Storing unstructured/semi-structured gaming logs

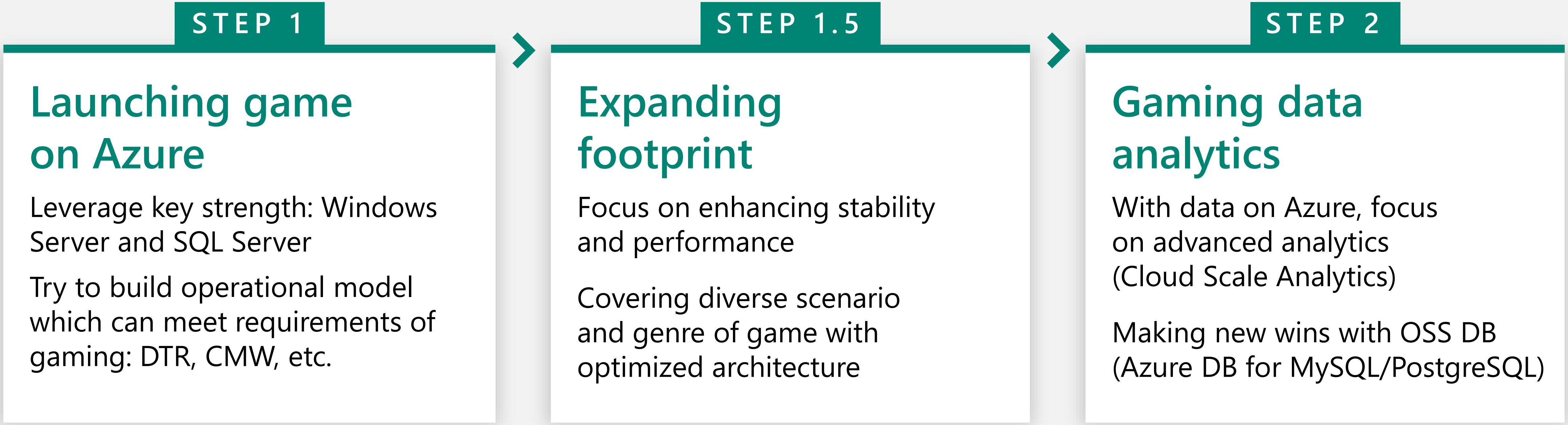
- Game server store logs with JSON for flexibility (adding more items if needed)
- To store and handling JSON with more ease, Use Azure Data Explorer as log database
- Use "update policy" feature to broadcast logs to multiple tables in ingestion process
- Use function to parse JSON with "Extend" to make structed view
- Continuously export logs to Azure Data Lake Storage Gen2 for archiving purpose and use external table to explore


Copy logs to blob storage with AzCopy
(Batch job in Task Scheduler)
Game server
LOG
Store game logs in local disk
Blob storage
10
01
Ingest data using EventGrid/Eventhub
Azure Data Explorer: Data exploration and ML
Batch
Data Management
Engine
Ingested Data
External table to explore logs archived
Continuous export to ADLS for archiving
ADLS Gen2

Demo

Ingesting data and exploration

How we could be successful in gaming



STEP 1

Launching game on Azure

Leverage key strength: Windows Server and SQL Server

Try to build operational model which can meet requirements of gaming: DTR, CMW, etc.

STEP 1.5

Expanding footprint

Focus on enhancing stability and performance

Covering diverse scenario and genre of game with optimized architecture

STEP 2

Gaming data analytics

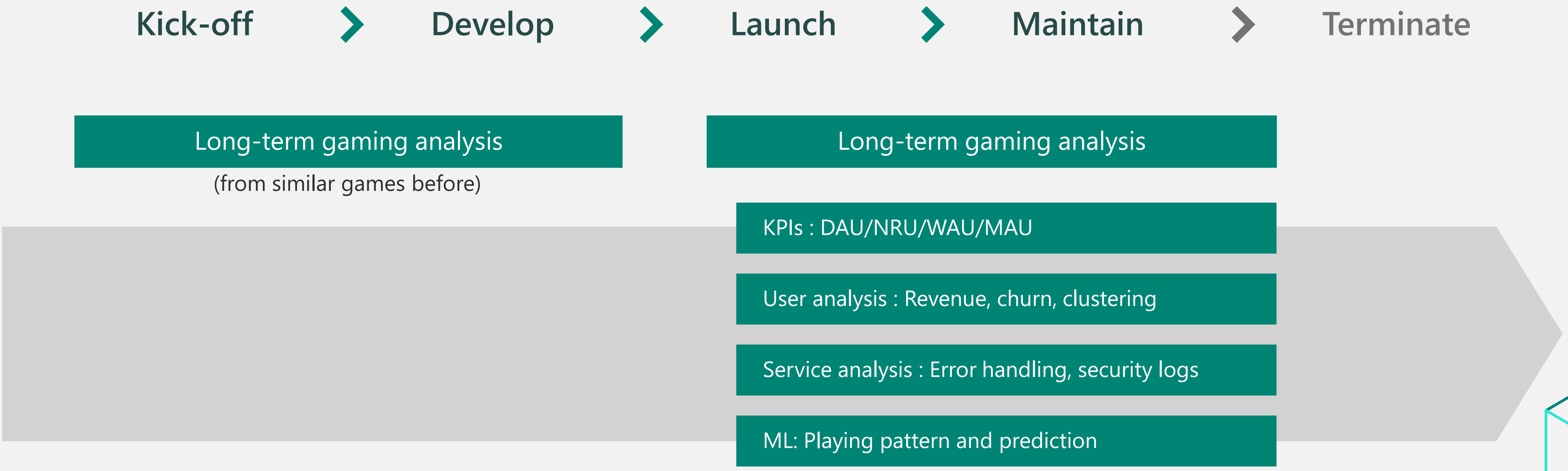
With data on Azure, focus on advanced analytics (Cloud Scale Analytics)

Making new wins with OSS DB (Azure DB for MySQL/PostgreSQL)

Step 2
Gaming data analytics

Gaming data analytics

Gaming life cycle



Kick-off › Develop › Launch › Maintain › Terminate

Long-term gaming analysis

(from similar games before)

Long-term gaming analysis

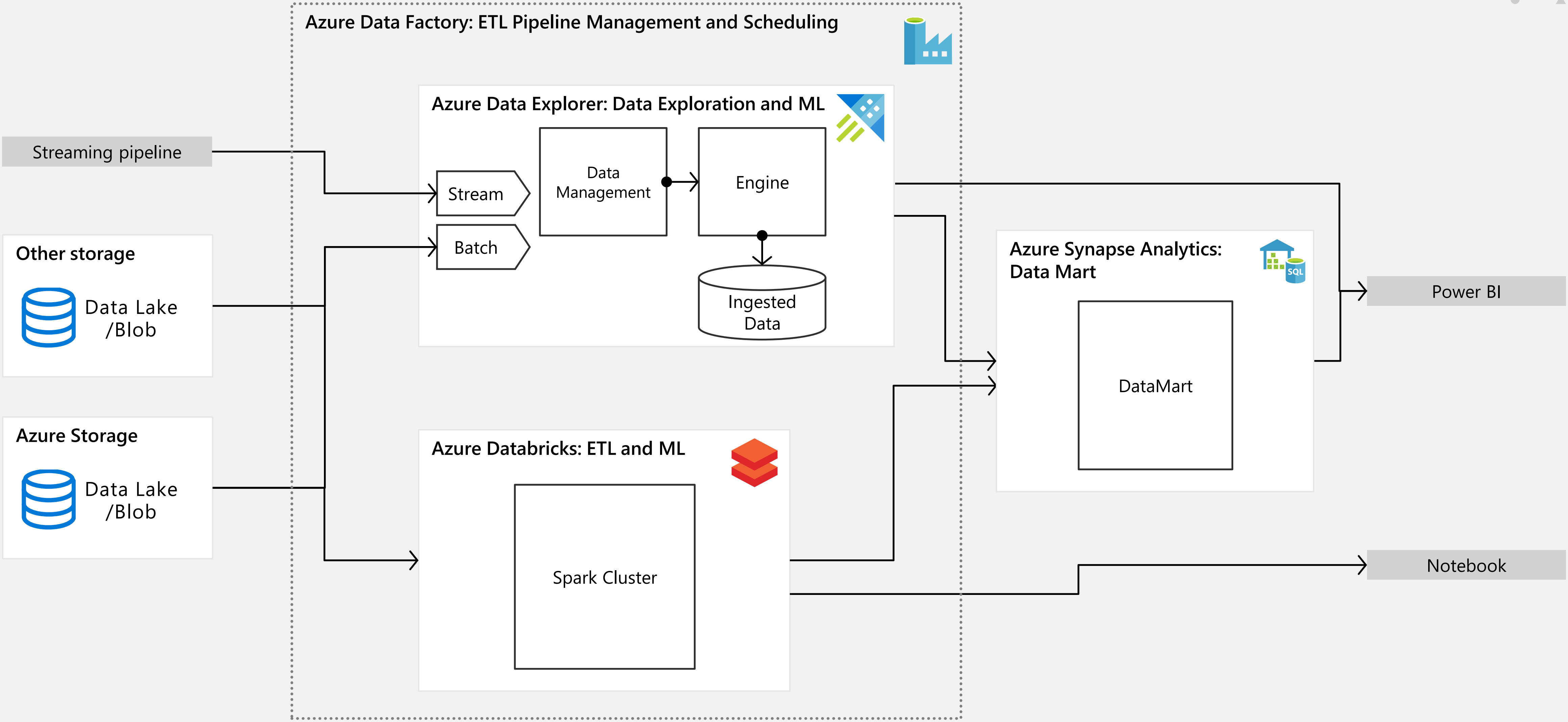
KPIs : DAU/NRU/WAU/MAU

User analysis : Revenue, churn, clustering

Service analysis : Error handling, security logs

ML: Playing pattern and prediction

Building standard architecture


Azure Data Factory: ETL Pipeline Management and Scheduling
Azure Data Explorer: Data Exploration and ML
Streaming pipeline
Stream
Data Management
Engine
Batch
Other storage
Data Lake /Blob
Ingested Data
Azure Synapse Analytics: Data Mart
Power BI
DataMart
Azure Storage
Data Lake /Blob
Azure Databricks: ETL and ML
Spark Cluster
Notebook

CASE STUDY #1

Storing and exploring security logs

KEY BENEFIT

Store vast security logs with cost effective way; Explorer logs with query sample with easy

Background

Launched mobile/PC (online)/console on Azure with several regions

Need to detect detail route and cause of any security issue (SIEM—Security Information and Event Management)

Challenge

ELK is too expensive to expand cluster and hold more data

Azure Sentinel does not meet request due to alerting interval

Solutions

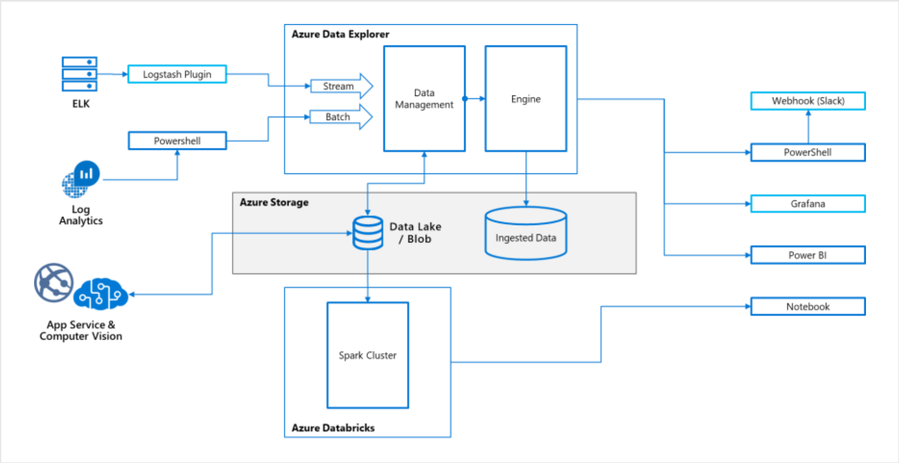
Use ELK for short-term real time data analytics (<1month)

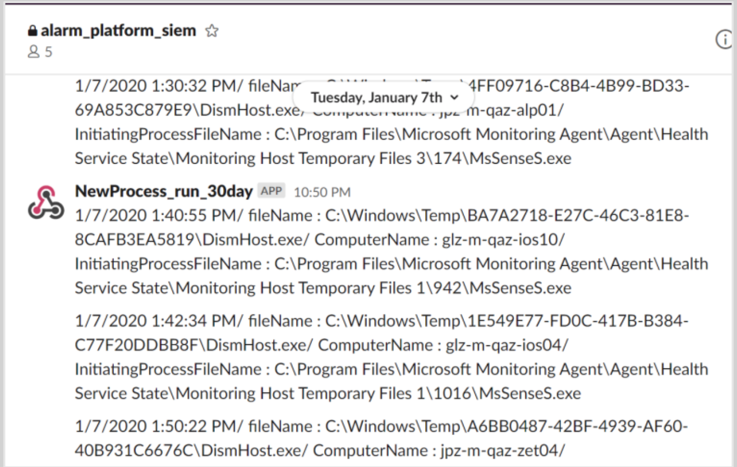
Use Azure Data Explorer for long-term data store, exploration and analysis

Refer hunting queries of Azure sentinel for faster adoption

CASE STUDY #1

Storing and exploring security logs





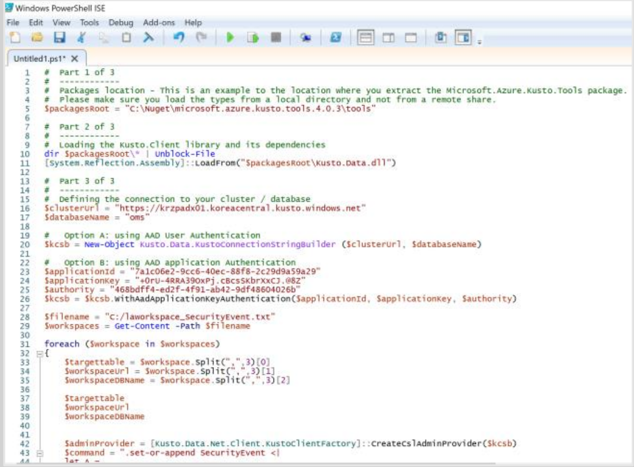
1/7/2020 1:30:32 PM/ fileNam 4FF09716-C8B4-4B99-BD33-69A853C879E9\DismHost.exe/ m-qaz-alp01/ InitiatingProcessFileName : C:\Program Files\Microsoft Monitoring Agent\Agent\Health Service State\Monitoring Host Temporary Files 3\174\MsSenseS.exe

NewProcess_run_30day APP 10:50 PM

1/7/2020 1:40:55 PM/ fileName : C:\Windows\Temp\BA7A2718-E27C-46C3-81E8-8CAFB3EA5819\DismHost.exe/ ComputerName : glz-m-qaz-ios10/ InitiatingProcessFileName : C:\Program Files\Microsoft Monitoring Agent\Agent\Health Service State\Monitoring Host Temporary Files 1\942\MsSenseS.exe

1/7/2020 1:42:34 PM/ fileName : C:\Windows\Temp\1E549E77-FD0C-417B-B384-C77F20DDBB8F\DismHost.exe/ ComputerName : glz-m-qaz-ios04/ InitiatingProcessFileName : C:\Program Files\Microsoft Monitoring Agent\Agent\Health Service State\Monitoring Host Temporary Files 1\1016\MsSenseS.exe

1/7/2020 1:50:22 PM/ fileName : C:\Windows\Temp\A6BB0487-42BF-4939-AF60-40B931C6676C\DismHost.exe/ ComputerName : jpz-m-qaz-zet04/



```
#  Part 1 of 3
#  ------------
#  Packages location - This is an example to the location where you extract the Microsoft.Azure.Kusto.Tools package.
#  Please make sure you load the types from a local directory and not from a remote share.
$packagesRoot = "C:\Nuget\microsoft.azure.kusto.tools.4.0.3\tools"

#  Part 2 of 3
#  ------------
#  Loading the Kusto.Client library and its dependencies
dir $packagesRoot\* | Unblock-File
[System.Reflection.Assembly]::LoadFrom("$packagesRoot\Kusto.Data.dll")

#  Part 3 of 3
#  ------------
#  Defining the connection to your cluster / database
$clusterUrl = "https://krzpadx01.koreacentral.kusto.windows.net"
$databaseName = "oms"

#   Option A: using AAD User Authentication
$kcsb = New-Object Kusto.Data.KustoConnectionStringBuilder ($clusterUrl, $databaseName)

#   Option B: using AAD application Authentication
$applicationId = "7a1c06e2-9cc6-40ec-88f8-2c29d9a59a29"
$applicationKey = "+0rU-4RRA39OxPj.cBcsSkbrXxCJ.@8Z"
$authority = "468bdff4-ed2f-4f91-ab42-9df48604026b"
$kcsb = $kcsb.WithAadApplicationKeyAuthentication($applicationId, $applicationKey, $authority)

$filename = "C:/laworkspace_SecurityEvent.txt"
$workspaces = Get-Content -Path $filename

foreach ($workspace in $workspaces)
{
    $targettable = $workspace.Split(",",3)[0]
    $workspaceUrl = $workspace.Split(",",3)[1]
    $workspaceDBName = $workspace.Split(",",3)[2]

    $targettable
    $workspaceUrl
    $workspaceDBName


    $adminProvider = [Kusto.Data.Net.Client.KustoClientFactory]::CreateCslAdminProvider($kcsb)
    $command = ".set-or-append SecurityEvent <|
```

Key products

Azure Blob Storage

Azure Data Explorer

Azure Log Analytics

Requirement

Ingest data from multiple sources for long-term analytics: ELK, Azure Log Analytics

Real-time data ingestion without latency

Send alert message to Slack when action is needed

Solution

Ingest log on ELK to ADX with Logstash Plugin

Ingest data on Log Analytics with Powershell Script and proxy connection

Ad-hoc data exploration on ADX

Send alert message to Slack with Webhook

Next step

Ingest log on ELK to ingest game service log

Adopt ADX built-in dashboard

Research for visualization method

CASE STUDY #2

Minimizing time needed to find issue in game service

KEY BENEFIT

Minimize time to find issue in debugging with fast game service exploration

Background

After launch, to enhance quality of game, need to analyze game services logs and battle logs

Challenge

Used table storage at first, but impossible to see data due to slow performance

As scheme of logs cannot be fixed, needs solution that can store and handle JSON data efficiently

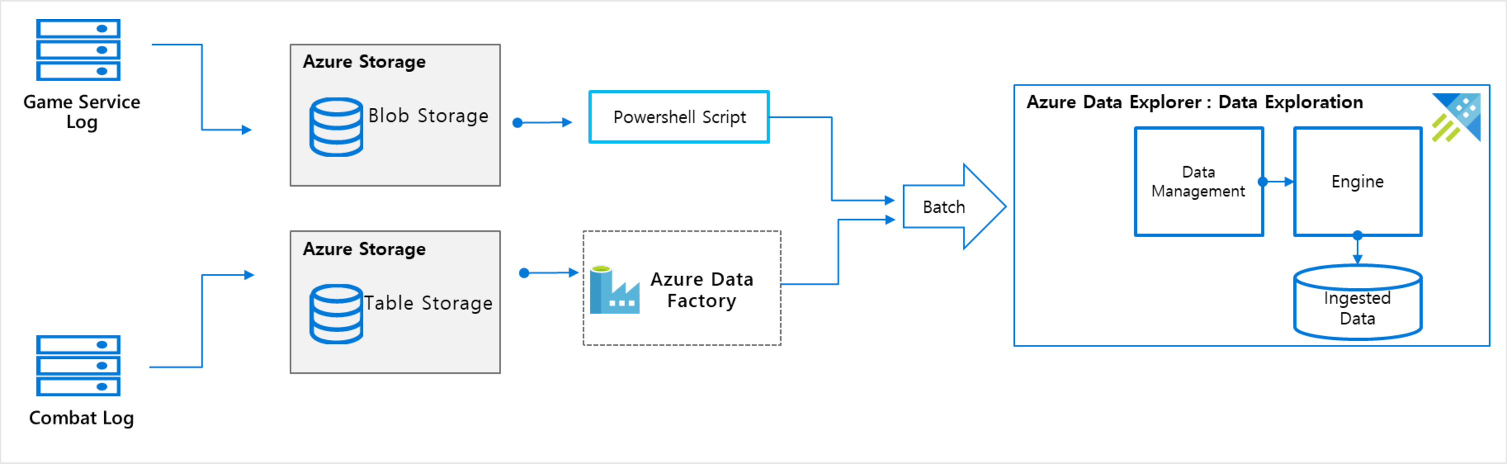
Solutions

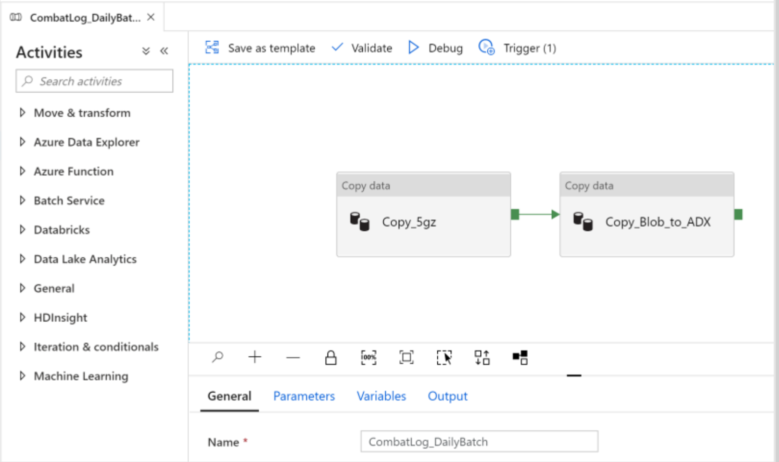
Replace table storage to Azure Data Explorer to store game services log and battle logs

Use Azure Data Factory for ETL pipeline

CASE STUDY #2

Minimizing time needed to find issue in game service





Name * CombatLog_DailyBatch

Key products

Azure Blob Storage

Azure Data Explorer

Azure Data Factory

Azure Function

Requirement

Ingest game service log and combat log

Ad-hoc query to json type data

Daily/Hourly batch ingestion job

Solution

Send log from game server to Azure Blob Storage with Azure Function

Ingest game service log from Azure Blob Storage to Azure Data Explorer with Powershell Script

Ingest combat log from Azure Table Storage to Azure Data Explorer with Data Factory

Ad-hoc query in Azure Data Explorer

Next step

Expand coverage to all game log

CASE STUDY #3

Building end-to-end game service platform

KEY BENEFIT

Build end-to-end game services platform using Azure; Implementing ML-based abuser detection

Background

Building common service platform to launch several games planned

Challenge

As size of log data is very big in general, needs solution to store, explorer and analysis logs minimizing cost

To store several types of logs with different scheme, needs solution that can store and handle JSON data efficiently

Solutions

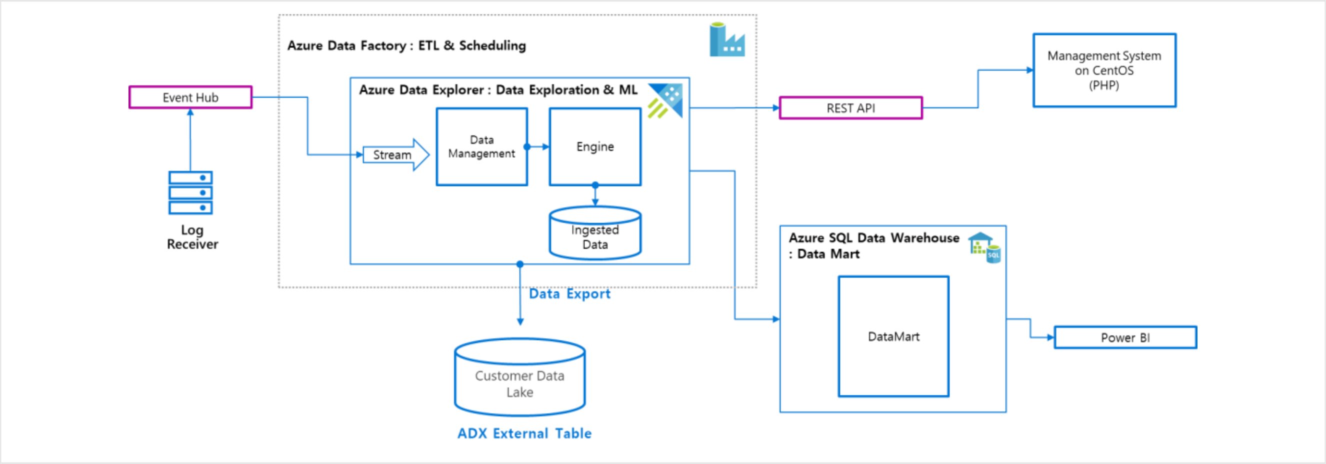
Use Azure Data Explorer to store logs and visualize data with dashboard

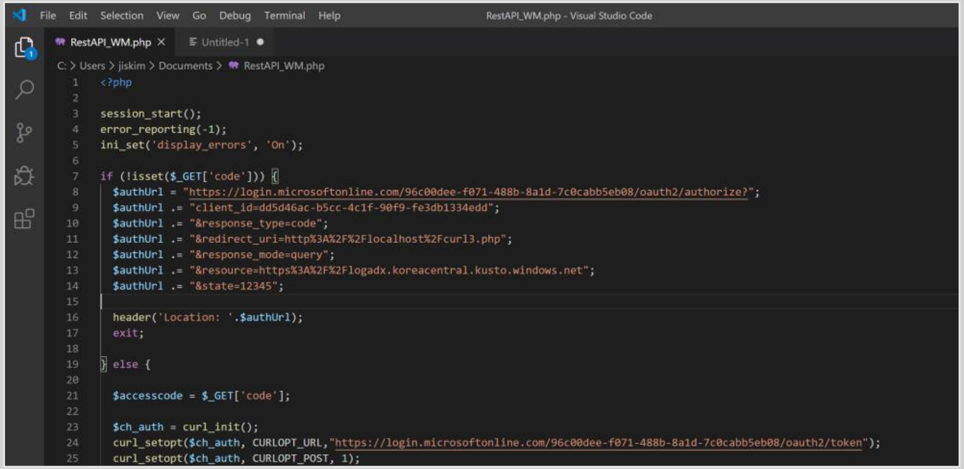
Use Power BI for KPI reports

Export data continuously to Azure Blob Storage with cool tier to store archive data with less cost

CASE STUDY #3

Building end-to-end game service platform


Azure Data Factory : ETL & Scheduling
Event Hub
Log Receiver
Azure Data Explorer : Data Exploration & ML
Stream
Data Management
Engine
Ingested Data
Data Export
Customer Data Lake
ADX External Table
REST API
Management System on CentOS (PHP)
Azure SQL Data Warehouse : Data Mart
DataMart
Power BI


File
Edit
Selection
View
Go
Debug
Terminal
Help
RestAPI_WM.php - Visual Studio Code
RestAPI_WM.php
Untitled-1
C: > Users > jiskim > Documents > RestAPI_WM.php

```php
<?php

session_start();
error_reporting(-1);
ini_set('display_errors', 'On');

if (!isset($_GET['code'])) {
  $authUrl = "https://login.microsoftonline.com/96c00dee-f071-488b-8a1d-7c0cabb5eb08/oauth2/authorize?";
  $authUrl .= "client_id=dd5d46ac-b5cc-4c1f-90f9-fe3db1334edd";
  $authUrl .= "&response_type=code";
  $authUrl .= "&redirect_uri=http%3A%2F%2Flocalhost%2Fcurl3.php";
  $authUrl .= "&response_mode=query";
  $authUrl .= "&resource=https%3A%2F%2Flogadx.koreacentral.kusto.windows.net";
  $authUrl .= "&state=12345";

  header('Location: '.$authUrl);
  exit;

} else {

  $accesscode = $_GET['code'];

  $ch_auth = curl_init();
  curl_setopt($ch_auth, CURLOPT_URL,"https://login.microsoftonline.com/96c00dee-f071-488b-8a1d-7c0cabb5eb08/oauth2/token");
  curl_setopt($ch_auth, CURLOPT_POST, 1);
```

Key products

Azure Blob Storage

Azure Data Explorer

Azure Data Factory

Azure SQL VM (Power BI Report Server)

Requirement

Integrate data analytics platform with management system on CentOS (PHP)

Ad-hoc data analytics environment with json type data

Archive old data to customer data lake (Blob Storage)

Run daily batch ETL job and put summarized data to data mart

Connect data mart with Power BI

Solution

Connect Azure Data Explorer to CentOS with Rest API

Ad-hoc query environment on Azure Data Explorer

Run batch process with Azure Data Factory

Next step

Advanced Analytics pilot for abuse detection

CASE STUDY #3

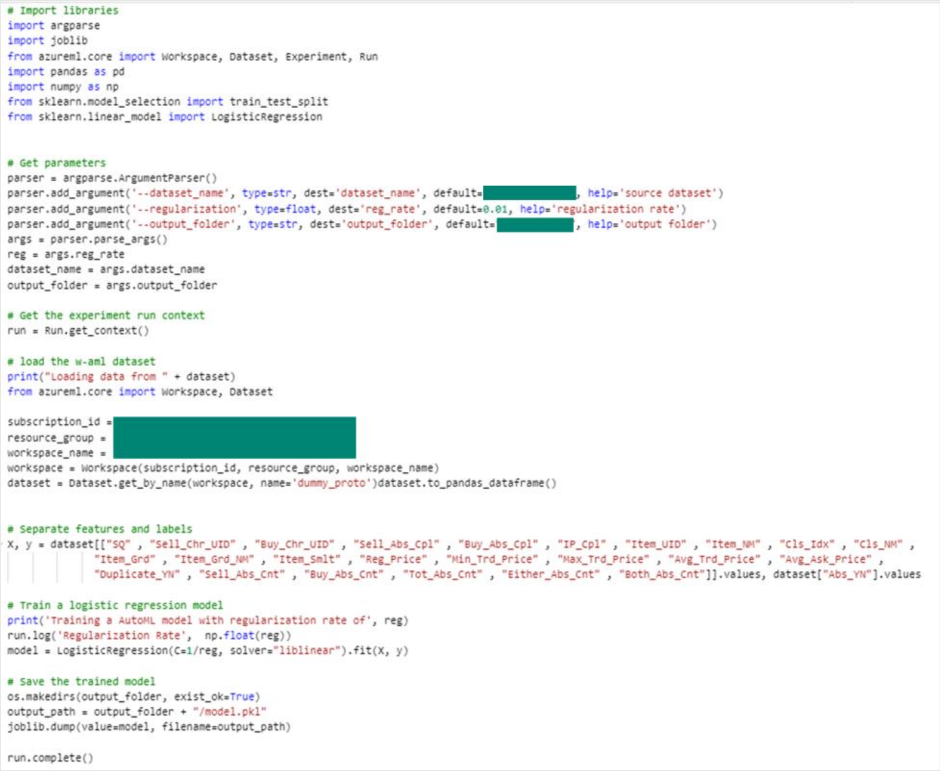
Building end-to-end game service platform

Advanced analytics and AI with key scenarios

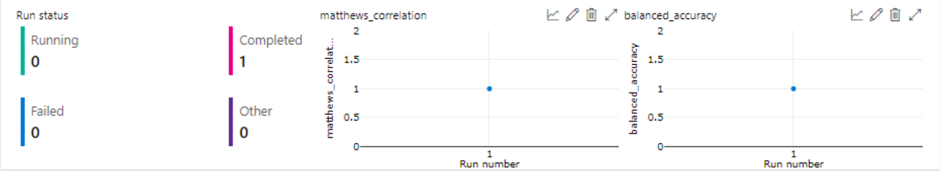
- Abuser detection: Built machine learning model to identify abuser with game marketplace logs
- Bot for community website: Automated reply for FAQ from gaming users

Solutions

- Use Azure ML to build abuser detection model
- Integrated model with LiveOps for automated abuser handling
- MLOps with Azure DevOps to re-train and deploy model
- Build Q&A Maker knowledge base with FAQ from gaming users
- Build Bot and integrated it to community website to provide automated first reply to users



```
# Import libraries
import argparse
import joblib
from azureml.core import Workspace, Dataset, Experiment, Run
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression


# Get parameters
parser = argparse.ArgumentParser()
parser.add_argument('--dataset_name', type=str, dest='dataset_name', default=[redacted], help='source dataset')
parser.add_argument('--regularization', type=float, dest='reg_rate', default=0.01, help='regularization rate')
parser.add_argument('--output_folder', type=str, dest='output_folder', default=[redacted], help='output folder')
args = parser.parse_args()
reg = args.reg_rate
dataset_name = args.dataset_name
output_folder = args.output_folder

# Get the experiment run context
run = Run.get_context()

# load the w-aml dataset
print("Loading data from " + dataset)
from azureml.core import Workspace, Dataset

subscription_id =[redacted]
resource_group =
workspace_name =
workspace = Workspace(subscription_id, resource_group, workspace_name)
dataset = Dataset.get_by_name(workspace, name='dummy_proto')dataset.to_pandas_dataframe()


# Separate features and labels
X, y = dataset[["SQ" , "Sell_Chr_UID" , "Buy_Chr_UID" , "Sell_Abs_Cpl" , "Buy_Abs_Cpl" , "IP_Cpl" , "Item_UID" , "Item_NM" , "Cls_Idx" , "Cls_NM" ,
            "Item_Grd" , "Item_Grd_NM" , "Item_Smlt" , "Reg_Price" , "Min_Trd_Price" , "Max_Trd_Price" , "Avg_Trd_Price" , "Avg_Ask_Price" ,
            "Duplicate_YN" , "Sell_Abs_Cnt" , "Buy_Abs_Cnt" , "Tot_Abs_Cnt" , "Either_Abs_Cnt" , "Both_Abs_Cnt"]].values, dataset["Abs_YN"].values

# Train a logistic regression model
print('Training a AutoML model with regularization rate of', reg)
run.log('Regularization Rate',  np.float(reg))
model = LogisticRegression(C=1/reg, solver="liblinear").fit(X, y)

# Save the trained model
os.makedirs(output_folder, exist_ok=True)
output_path = output_folder + "/model.pkl"
joblib.dump(value=model, filename=output_path)

run.complete()
```



Run status

Running 0

Completed 1

Failed 0

Other 0

CASE STUDY #4

Move from on-premises to Azure

KEY BENEFIT

Moving whole analytical platform to cloud with ease; Implementing more enhanced user analytics

Challenge

Seamless migration from On-premises Hadoop with minimal effort

Improve analytic system to expand scenario in advanced analytics

Solutions

At first, beginning of migration of Hadoop, adopted HDInsight for Hadoop/Spark migration, Azure SQL Data Warehouse as Data Mart and Data Warehouse

As 2nd step, moved Azure Databricks to get more flexible, cost efficient Spark environment and Azure Data Explorer for real-time data analytics/exploration

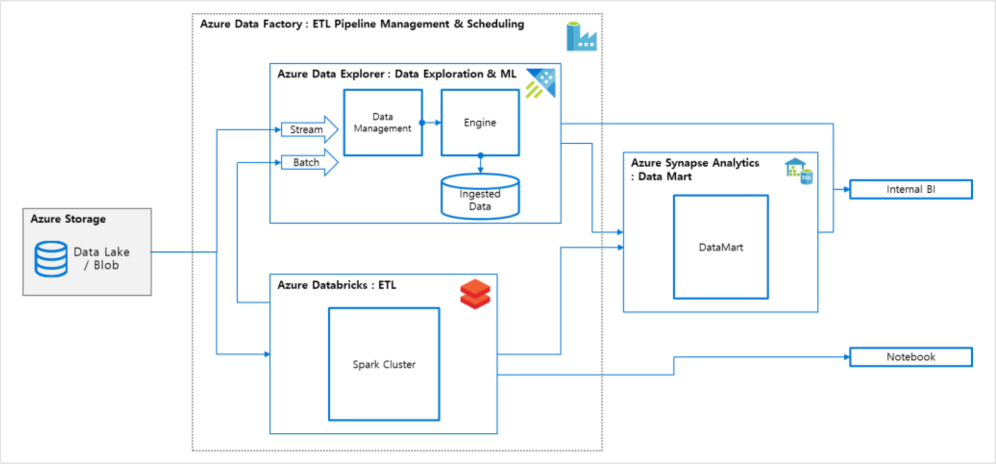
Next step

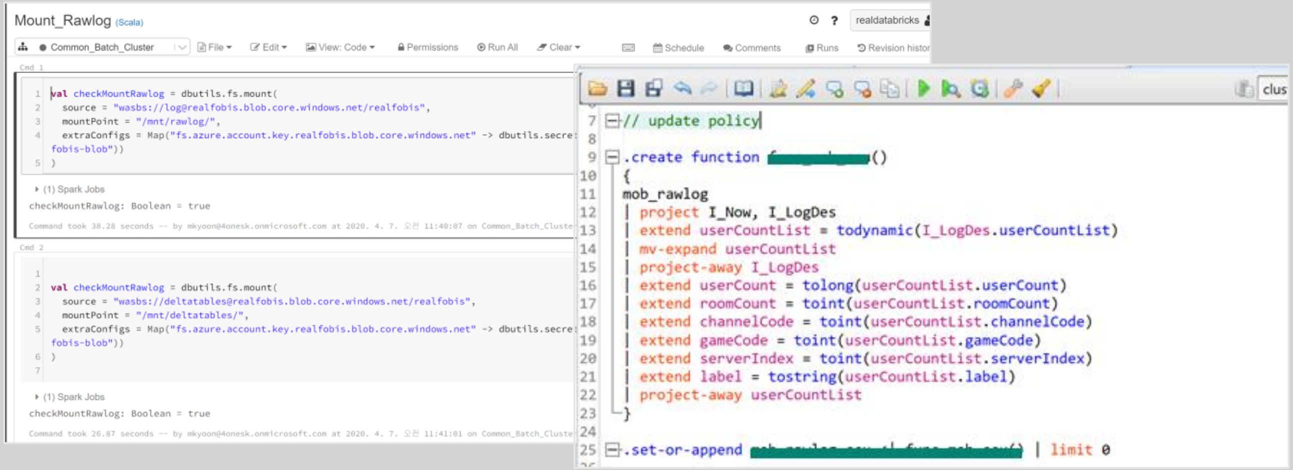
Advanced Analytics pilot for user segmentation

As future roadmap, introduce fully-integrated analytics system with Synapse analytics

CASE STUDY #4

Move from on-premises to Azure





Cmd 1

```
val checkMountRawlog = dbutils.fs.mount(
  source = "wasbs://log@realfobis.blob.core.windows.net/realfobis",
  mountPoint = "/mnt/rawlog/",
  extraConfigs = Map("fs.azure.account.key.realfobis.blob.core.windows.net" -> dbutils.secre
fobis-blob"))
)
```

(1) Spark Jobs

```
checkMountRawlog: Boolean = true
Command took 38.28 seconds -- by mkyoon@4onesk.onmicrosoft.com at 2020. 4. 7. 오전 11:40:07 on Common_Batch_Cluste
```

Cmd 2

```

val checkMountRawlog = dbutils.fs.mount(
  source = "wasbs://deltatables@realfobis.blob.core.windows.net/realfobis",
  mountPoint = "/mnt/deltatables/",
  extraConfigs = Map("fs.azure.account.key.realfobis.blob.core.windows.net" -> dbutils.secre
fobis-blob"))
)

```

(1) Spark Jobs

```
checkMountRawlog: Boolean = true
Command took 26.87 seconds -- by mkyoon@4onesk.onmicrosoft.com at 2020. 4. 7. 오전 11:41:01 on Common_Batch_Cluste
```

```
// update policy

.create function [redacted]()
{
mob_rawlog
| project I_Now, I_LogDes
| extend userCountList = todynamic(I_LogDes.userCountList)
| mv-expand userCountList
| project-away I_LogDes
| extend userCount = tolong(userCountList.userCount)
| extend roomCount = toint(userCountList.roomCount)
| extend channelCode = toint(userCountList.channelCode)
| extend gameCode = toint(userCountList.gameCode)
| extend serverIndex = toint(userCountList.serverIndex)
| extend label = tostring(userCountList.label)
| project-away userCountList
}

.set-or-append [redacted] | limit 0
```

Key products

Azure Blob Storage

Azure Data Explorer

Azure Data Factory

Azure Synapse Analytics (Azure SQL DW)

Azure Databricks

Requirement

Adopt high-performance gaming data analytics platform

Ad-hoc query for json type data

Advanced analytics including ML for next step of data analytics scenario

Cost effective spark engine that can perform batch ETL workload

Solution

ETL and batch ingestion to Azure Data Explorer with Azure Databricks (reduce cost with auto termination of Azure Databricks)

Stream ingestion to Azure Data Explorer with Event hub

Next step

Advanced Analytics pilot for user segmentation

As future roadmap, introduce fully-integrated analytics system with Synapse analytics

CASE STUDY #4

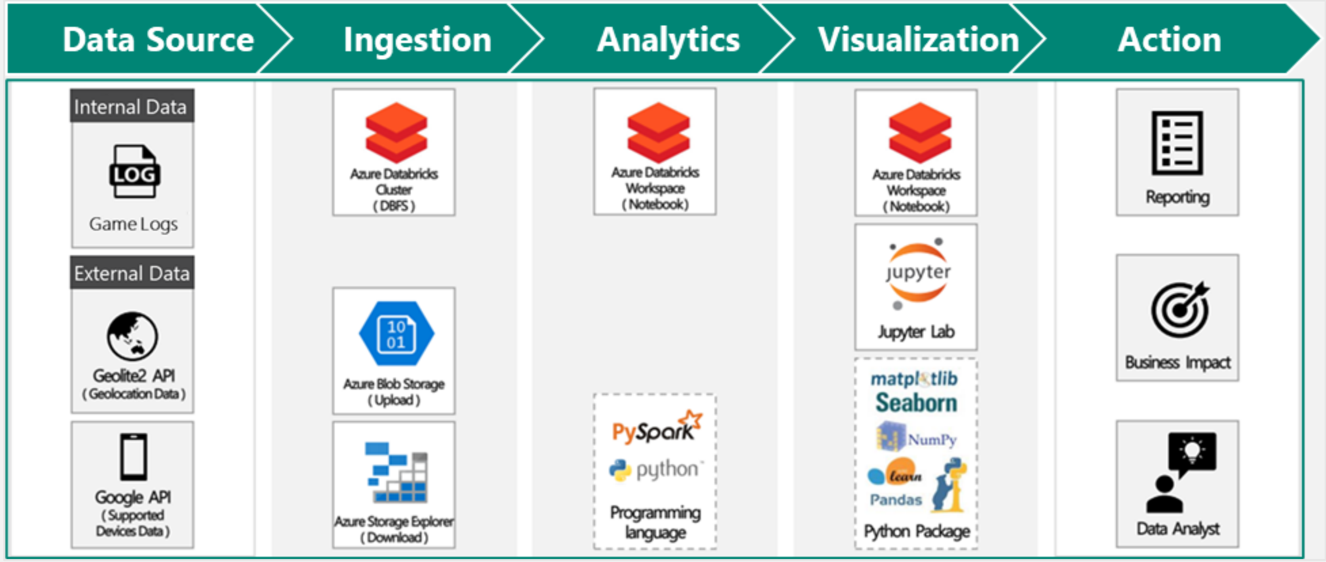
Move from on-premises to Azure

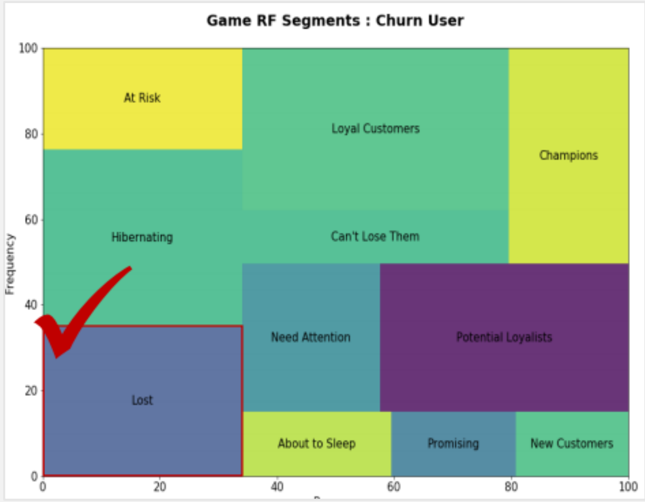
Advanced analytics and AI with key scenarios

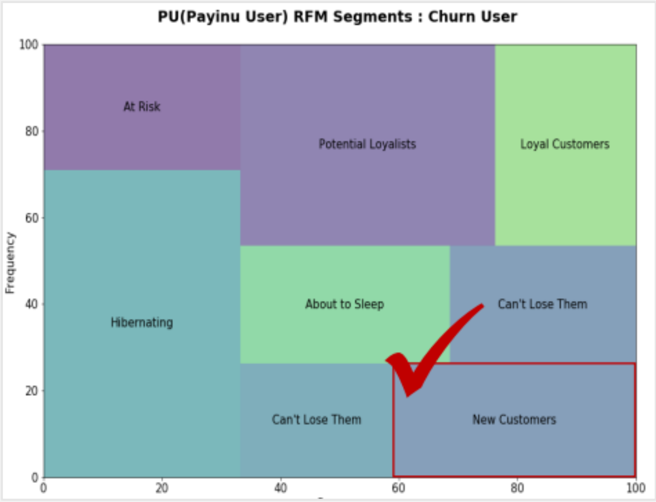
- User segmentation and marketing analysis: Build machine learning model for user segmentation

Solutions

- Use Azure Databricks to build machine learning model for user segmentation
- Define segmentation criteria: In-game factor (item purchase, time etc.)/others (region, age, etc.)
- Propose proper action for each segment




Game RF Segments : Churn User


PU(Payinu User) RFM Segments : Churn User

Microsoft