Azure OpenAI Batch API Accelerator

DJ Dean – Principal Cloud Solution Architect

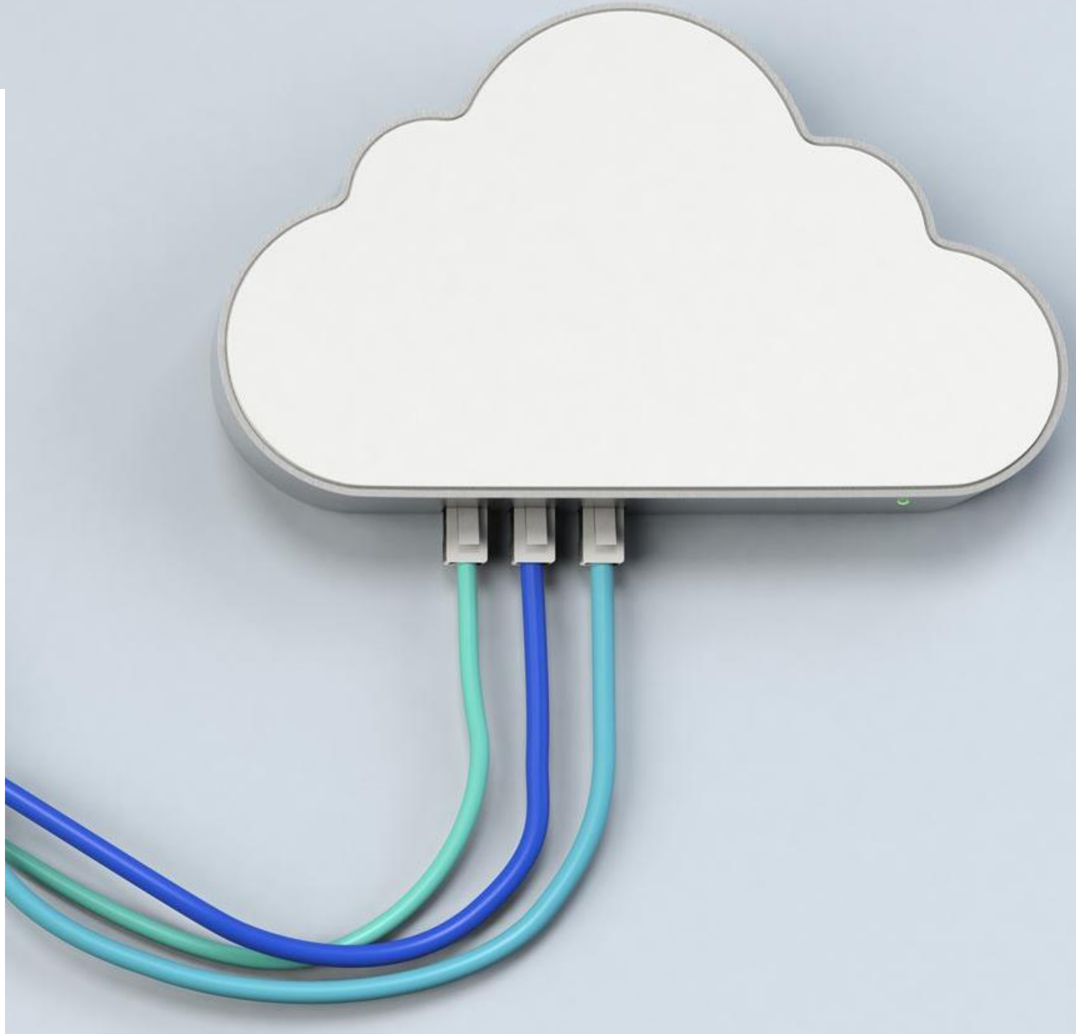
Amit Mukherjee – Prin. Technical Specialist – GenAI

October 2nd, 2024

Microsoft

Why We Need the AOAI Batch API

- The Batch API significantly reduces costs for non-time sensitive workloads.
- Implementing multi-threaded/multi-process functionality can be difficult and time-consuming.
- The Batch API provides significant performance benefits when processing large quantities of files.



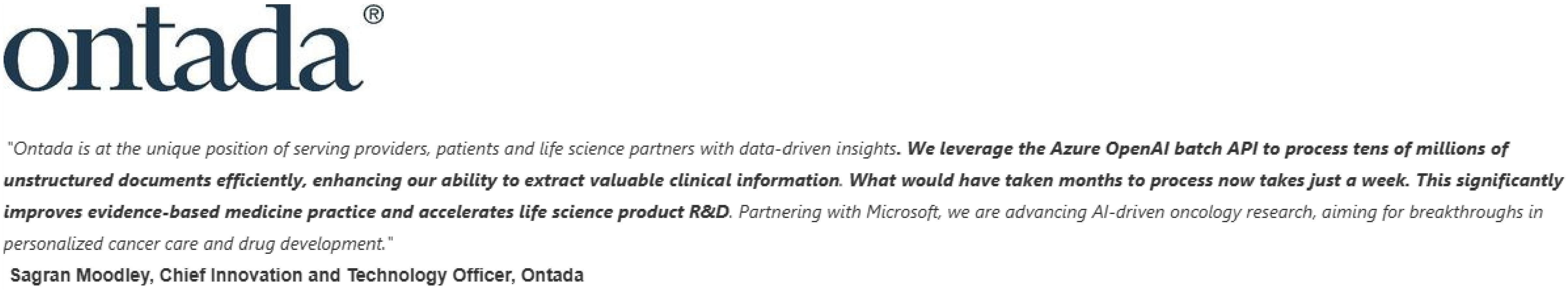
Customer Problem statement

- How to process tens of thousands of files to extract information
- How do track errors at the time of processing
- How to automate entire process
- How to make cost effective

AOAI Bath Accelerator

- Automated Batch Job Submission and Creation
- Multi-threaded Async Processing to Reduce Overall Processing Time
- Automated Error Tracking
- Multi-directory Hierarchy Support
- Configurable Micro-batch Support
- Automated Post-job Cleanup

Voice of Customer

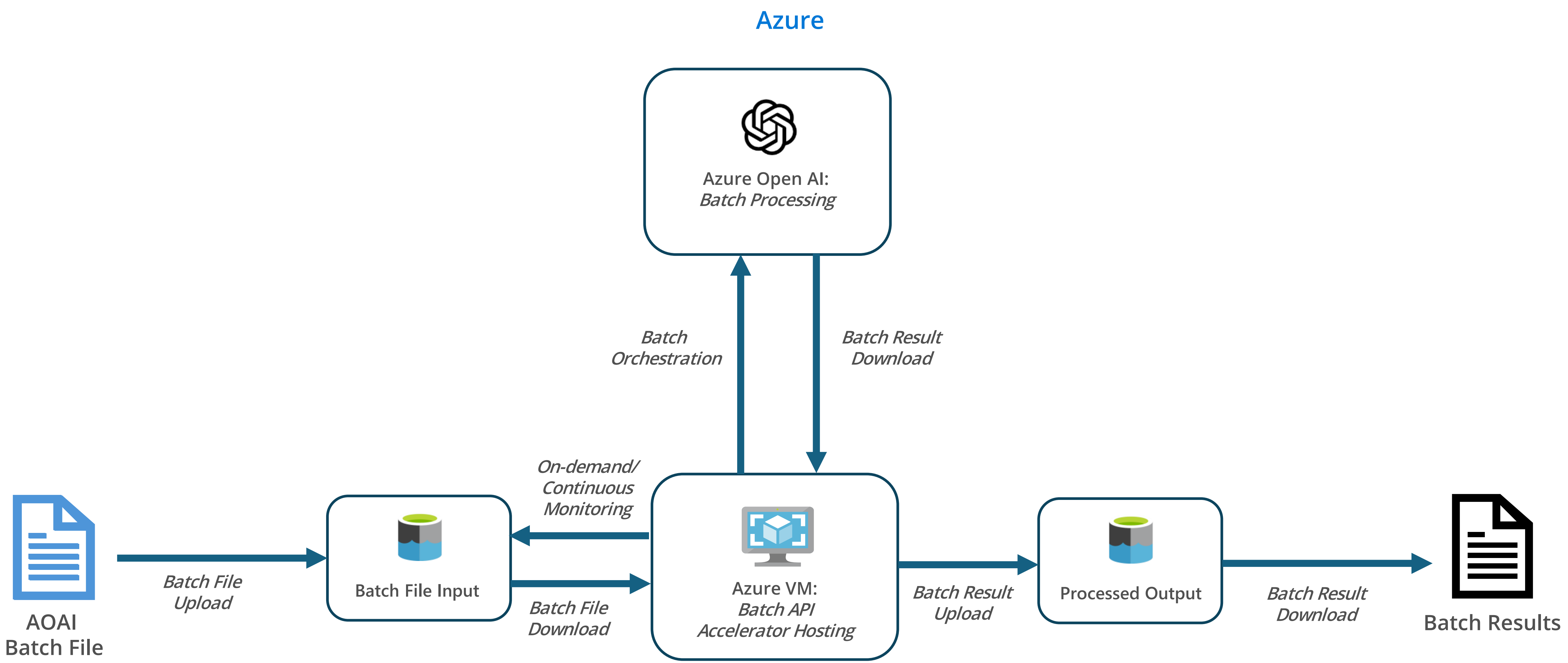


ontada®

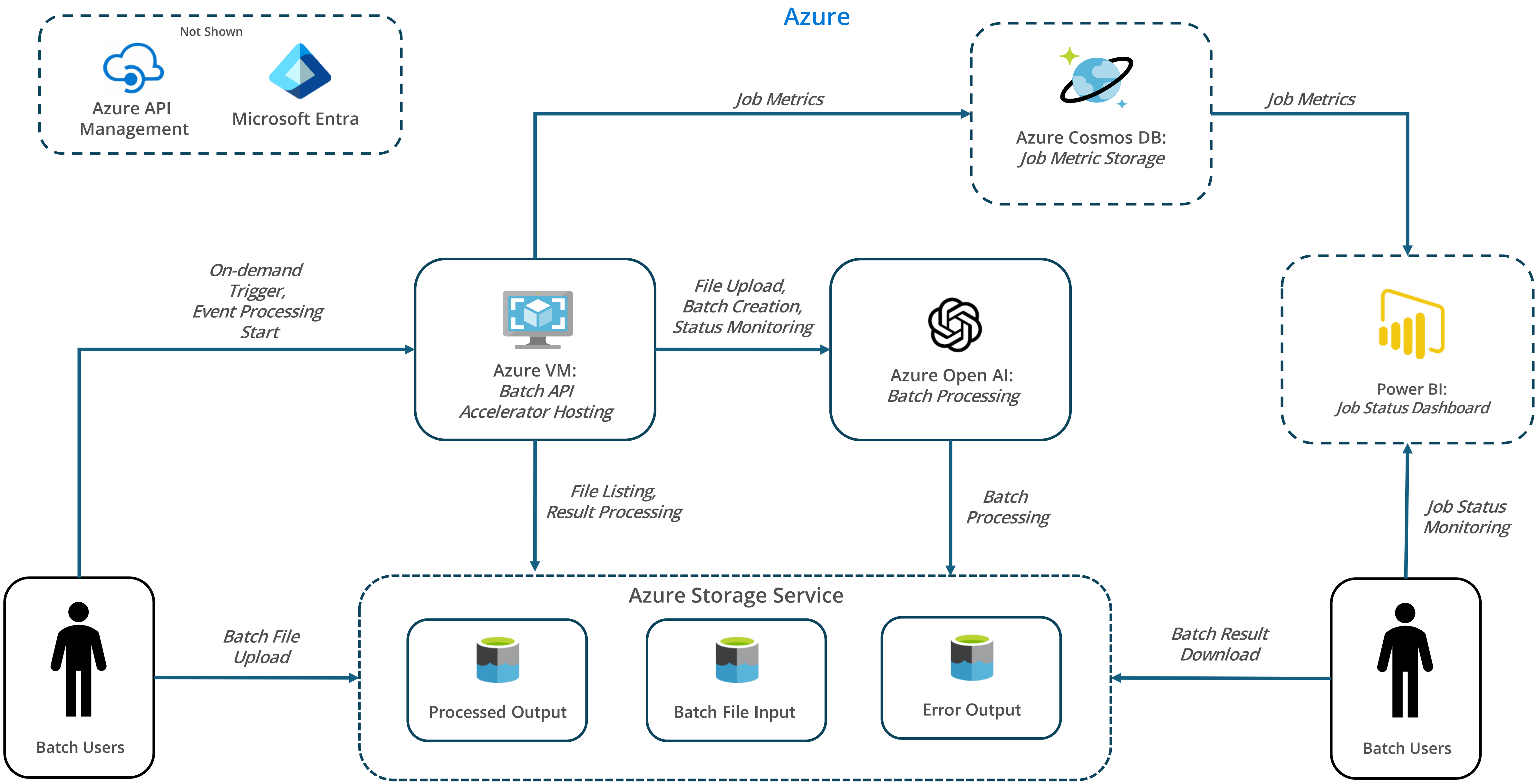
"*Ontada is at the unique position of serving providers, patients and life science partners with data-driven insights*. ***We leverage the Azure OpenAI batch API to process tens of millions of unstructured documents efficiently, enhancing our ability to extract valuable clinical information***. ***What would have taken months to process now takes just a week. This significantly improves evidence-based medicine practice and accelerates life science product R&D***. *Partnering with Microsoft, we are advancing AI-driven oncology research, aiming for breakthroughs in personalized cancer care and drug development.*"

Sagran Moodley, Chief Innovation and Technology Officer, Ontada

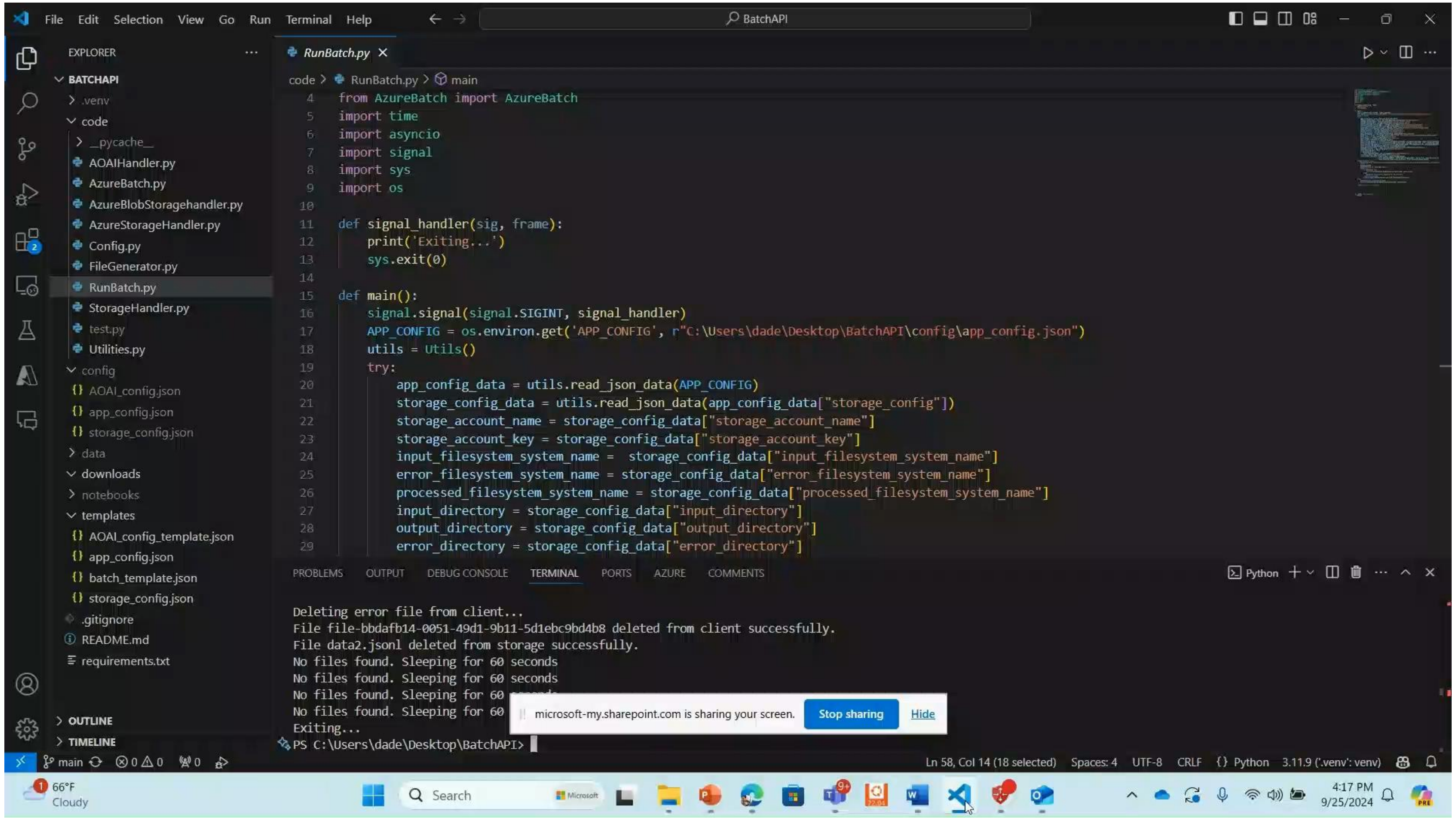
AOAI Batch Accelerator Overview



High-level Architecture (Detailed) | Data Flow and Consumption



Demo



```python
from AzureBatch import AzureBatch
import time
import asyncio
import signal
import sys
import os

def signal_handler(sig, frame):
    print('Exiting...')
    sys.exit(0)

def main():
    signal.signal(signal.SIGINT, signal_handler)
    APP_CONFIG = os.environ.get('APP_CONFIG', r"C:\Users\dade\Desktop\BatchAPI\config\app_config.json")
    utils = Utils()
    try:
        app_config_data = utils.read_json_data(APP_CONFIG)
        storage_config_data = utils.read_json_data(app_config_data["storage_config"])
        storage_account_name = storage_config_data["storage_account_name"]
        storage_account_key = storage_config_data["storage_account_key"]
        input_filesystem_system_name =  storage_config_data["input_filesystem_system_name"]
        error_filesystem_system_name = storage_config_data["error_filesystem_system_name"]
        processed_filesystem_system_name = storage_config_data["processed_filesystem_system_name"]
        input_directory = storage_config_data["input_directory"]
        output_directory = storage_config_data["output_directory"]
        error_directory = storage_config_data["error_directory"]
```

```
Deleting error file from client...
File file-bbdafb14-0051-49d1-9b11-5d1ebc9bd4b8 deleted from client successfully.
File data2.jsonl deleted from storage successfully.
No files found. Sleeping for 60 seconds
No files found. Sleeping for 60 seconds
No files found. Sleeping for 60
No files found. Sleeping for 60
Exiting...
PS C:\Users\dade\Desktop\BatchAPI>
```

microsoft-my.sharepoint.com is sharing your screen. Stop sharing Hide