

Final Project  
**Azure Batch**

Martin Bertrand

**Deep Azure**  
Dr. Zoran B. Djordjević

# The problem...

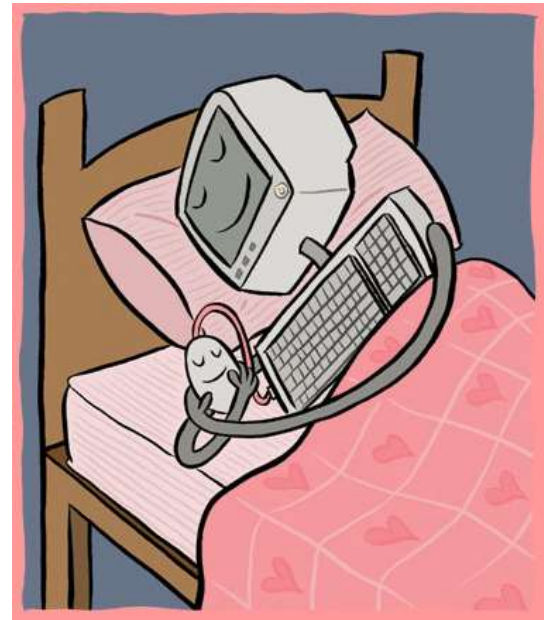
- You are receiving a steady flow of X12 EDI files that need to be transformed to XML format for further processing.
- Once a week a client sends you a LARGE number of X12 files (40 000), which overwhelms your X12 to XML conversion infrastructure.
- How would you solve such problem?

# The traditional solution

- Your infrastructure needs to be large enough to handle the highest workload possible.
  - Analysis and architecture;
  - Hardware purchase (RFP, PO, ...);
  - Computer room infrastructure (floor space, power, UPS, ...)
  - Network infrastructure;
  - Systems installation and configuration;
  - Software installation and configuration;
  - Support all this!

## The traditional solution - 2

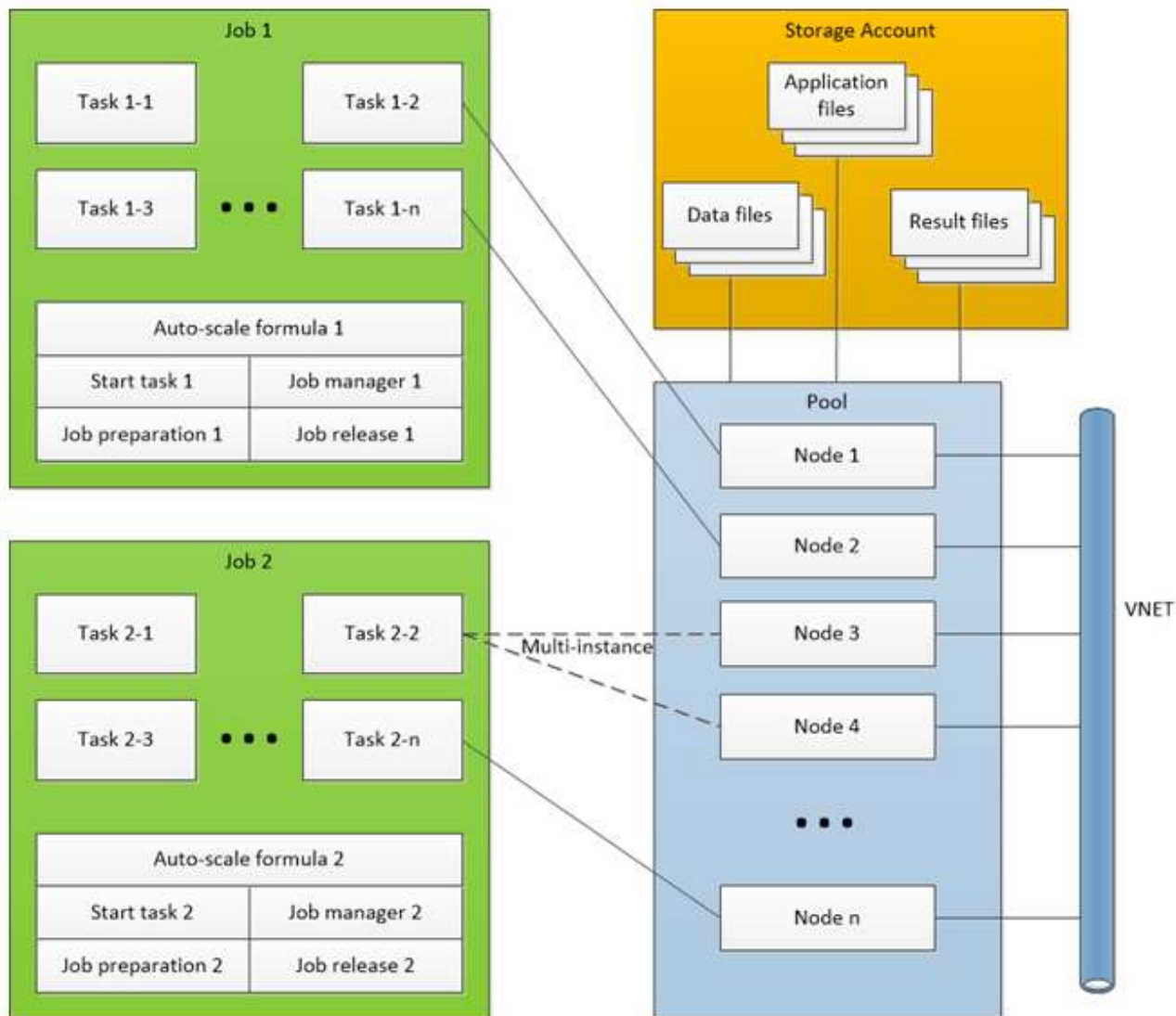
- Most of the time, that infrastructure would sit there doing nothing, since the large bursts of files are infrequent.
- This implies costs and waste of resources.



# The Azure Batch solution

- Azure Batch is a platform created to run parallel, high-performance computing.
- It also offers auto-scaling of compute resources, to meet varying demands.
- This reduces costs and resources waste.

# Azure Batch - Resources



# Azure Batch - Jobs

- Application are broken down in to jobs.
- Jobs are collections of tasks and define on which compute node pool(s) they will run.
- Jobs also define
  - Tasks priority and constraints.
  - The auto-scaling formula (based on the number of queue tasks, completion rate, time, resources, other metrics).

# Azure Batch - Tasks

- Tasks are units of computation.
- Each task can be executed on one or more compute nodes.
- They define:
  - What command to execute.
  - What files are required (application and data).
  - Environment variables.
  - Constraints.
  - Application packages or container images to use.
- Tasks can have dependencies between one another.
- The output of a task can be the input of another.



# Azure Batch – Compute nodes

- Compute nodes are virtual machines (Windows or Linux) or cloud service VMs (Windows only).
- Provide CPU, memory and disks resources.
- Are all identical within a unique pool. Create other pools if different nodes are required.
- Can be :
  - accessed like a regular VM (RDP or SSH).
  - based on standard or custom images.
  - dedicated (more expensive but never pre-empted) or low-priority (less expensive, uses surplus capacity).
  - added to the pool (scaling up), or removed from the pool (scaling down), depending on the auto-scaling formula, defined at the job level.
  - created for each job, and deleted as soon as it is complete, or be created ahead of time, thus reducing the start time, but increasing costs.

# Azure Batch - Files

- Types: application, data files (input) and result files (output).
- They are stored in a storage account blob.
- Application files are downloaded on compute nodes and executed.
- Data files are downloaded and processed by the application.
- Result files are uploaded back to the storage account.
- Files associated to a compute node are lost when the node is destroyed.

# Azure Batch - Applications

- They can be managed via packages.
- Can have many versions of an application used at the same time.
- Can be defined at the job, or task level.
- Job level: deployed to all nodes in the pool.
- Task level: only on nodes that are defined to run that particular package.

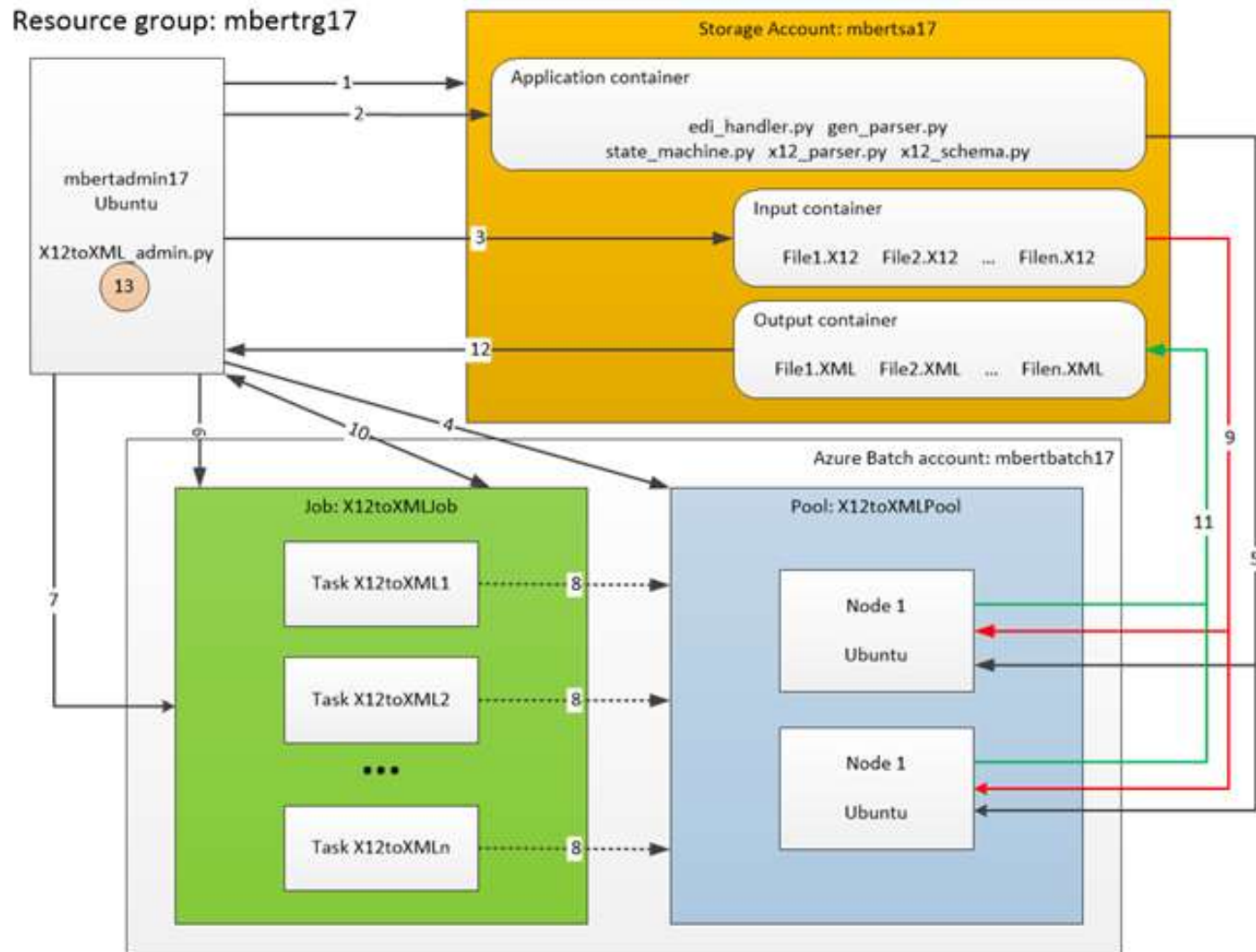
# Azure Batch - Network

- All nodes must be in the same region, in the same batch account, under the same subscription.
- Network Security Groups must allow communications between batch services and the nodes.

# Azure Batch - API

- Communicate with Azure Batch services.
- Create and manage nodes.
- Schedule jobs and tasks.
- Can be used via CLI, REST, .NET, Python, Node.js or Java.

# X12 to XML conversion - Infrastructure



## X12 to XML conversion – Infrastructure - 2

- An administration VM (Linux based) runs a Python script to administer Batch.
- This Python script uses the Azure Batch API to communicate with Batch and the storage account.
- The administration script creates required components and prepares files for processing using Batch.
- It also cleans up when the processing is done, to keep costs to a minimum.

# X12 to XML conversion - Executing

- A Linux bash script creates the required components on Azure, using Azure CLI commands.
- It creates:
  - The resource group.
  - A storage account.
  - A batch account.
  - The administration VM, with associated external IP and network components.
  - It configures the administration VM to run the conversion script.
- It uploads the application and data files to the administration VM.



# X12 to XML conversion – Executing - 2

- The Python administration script performs:
  - Create 3 containers in the storage account:
    - “application”: contains all the scripts that will perform the transformation.
    - “input”: will contain all the X12 format data files.
    - “output”: will contain the XML transformed files.
  - Upload the application and input files in the appropriate containers.
  - Create a pool that will contain compute nodes.
  - Create a job which will contain the tasks.
  - Create tasks inside the job. One task is created per input file. Each task uploads the converted file to the “output” storage container.
  - Waits for all tasks to complete.
  - Download the output files from the output container.
  - Delete the storage containers.
  - Delete the job.
  - Delete the pool.

# X12 to XML conversion – Portal

- While files are converted, it is possible to view the status of Batch resources via the Azure Portal. This view shows the Storage Account containers:

The screenshot shows the Azure Portal interface for the storage account 'mbertsa17'. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. Below these are settings for Containers, Access keys, Configuration, and Custom domain. The main content area shows the 'Containers' view with a search bar and a table of containers. The table has columns for NAME, LAST MODIFIED, PUBLIC ACCESS L..., and LEASE STATE. The containers listed are 'application', 'input', and 'output', all with a last modified time of 1/10/2018, 8:25:43 PM and a lease state of 'Available'.

mbertsa17 - Containers  
Storage account

Search (Ctrl+/)

+ Container Refresh

Storage account  
[mbertsa17](#)

Status  
Primary: Available

Location  
Canada East

Subscription ([change](#))  
[McKesson Deep Dive Training \(7\)](#)

Subscription ID  
6f5d1e5e-5295-4b19-9069-76aa53bdb9c

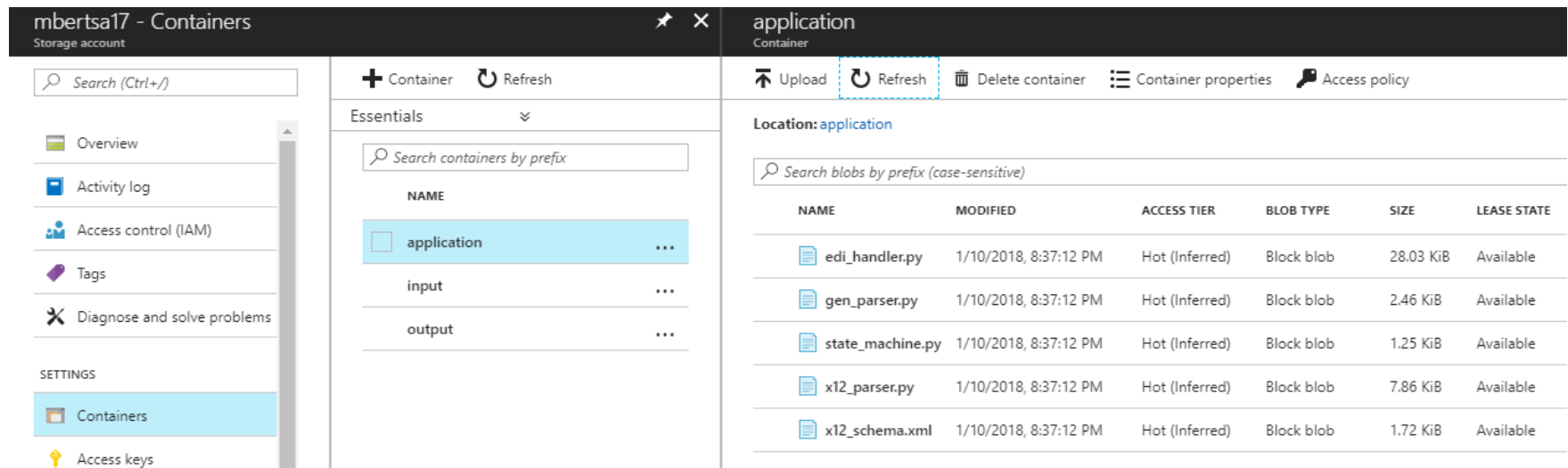
Blob service endpoint  
<https://mbertsa17.blob.core.windows.net/>

Search containers by prefix

NAME	LAST MODIFIED	PUBLIC ACCESS L...	LEASE STATE
application	1/10/2018, 8:25:43 PM	Private	Available
input	1/10/2018, 8:25:43 PM	Private	Available
output	1/10/2018, 8:25:43 PM	Private	Available

# X12 to XML conversion – Portal - 2

- The application container contains the scripts:



The screenshot displays the Azure Storage portal interface for a storage account named 'mberts17 - Containers'. The left sidebar shows navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and SETTINGS (Containers, Access keys). The main area is divided into two panes. The left pane shows the 'Essentials' section with a search bar and a list of containers: 'application', 'input', and 'output'. The 'application' container is selected. The right pane shows the 'application' container details, including a search bar and a table of blobs.

NAME	MODIFIED	ACCESS TIER	BLOB TYPE	SIZE	LEASE STATE
edi_handler.py	1/10/2018, 8:37:12 PM	Hot (Inferred)	Block blob	28.03 KiB	Available
gen_parser.py	1/10/2018, 8:37:12 PM	Hot (Inferred)	Block blob	2.46 KiB	Available
state_machine.py	1/10/2018, 8:37:12 PM	Hot (Inferred)	Block blob	1.25 KiB	Available
x12_parser.py	1/10/2018, 8:37:12 PM	Hot (Inferred)	Block blob	7.86 KiB	Available
x12_schema.xml	1/10/2018, 8:37:12 PM	Hot (Inferred)	Block blob	1.72 KiB	Available

# X12 to XML conversion – Portal - 3

- The input container contains the X12 files:

The screenshot displays the Azure Portal interface for a storage account named 'mberts17'. The left sidebar shows navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and a SETTINGS section with Containers, Access keys, Configuration, Custom domain, Encryption, and Azure CDN. The main area is divided into two panes. The left pane, titled 'input Container', shows a list of containers under 'Essentials': 'application', 'input' (selected), and 'output'. The right pane, titled 'input', shows a list of blobs under 'Location: input'. The blobs are listed in a table with columns: NAME, MODIFIED, ACCESS TIER, BLOB TYPE, SIZE, and LEASE STATE. All blobs are 'FileX.X12' files, modified on 1/10/2018 at 8:25:43 PM, with an 'Access tier' of 'Hot (Inferred)', 'Block blob' type, and 'Available' lease state.

NAME	MODIFIED	ACCESS TIER	BLOB TYPE	SIZE	LEASE STATE
File1.X12	1/10/2018, 8:25:43 PM	Hot (Inferred)	Block blob	514 B	Available
File2.X12	1/10/2018, 8:25:43 PM	Hot (Inferred)	Block blob	852 B	Available
File3.X12	1/10/2018, 8:25:43 PM	Hot (Inferred)	Block blob	672 B	Available
File4.X12	1/10/2018, 8:25:43 PM	Hot (Inferred)	Block blob	908 B	Available
File5.X12	1/10/2018, 8:25:43 PM	Hot (Inferred)	Block blob	514 B	Available
File6.X12	1/10/2018, 8:25:43 PM	Hot (Inferred)	Block blob	514 B	Available
File7.X12	1/10/2018, 8:25:43 PM	Hot (Inferred)	Block blob	514 B	Available
File8.X12	1/10/2018, 8:25:43 PM	Hot (Inferred)	Block blob	514 B	Available
File9.X12	1/10/2018, 8:25:43 PM	Hot (Inferred)	Block blob	514 B	Available

# X12 to XML conversion – Portal - 4

- A batch pool is created:

The screenshot shows the 'mbertbatch17 - Pools' management page in the Azure portal. On the left is a navigation pane with links to Properties, Quotas, Storage account, Keys, and Locks. The main area has a toolbar with 'Add', 'Columns', and 'Refresh' buttons. Below the toolbar is a dropdown menu set to 'All pools' and an 'Advanced query' section. A table lists the batch pools with columns for Pool ID, Dedicated Nodes, Low-Priority Nodes, Current Cores, VM Size, and Allocation State. One pool, 'X12toXMLPool', is shown with 0 dedicated nodes and 0 low-priority nodes, and its allocation state is 'Resizing'.

POOL ID	DEDICATED NODES	LOW-PRIORITY NODES	CURRENT CORES	VM SIZE	ALLOCATION STATE
X12toXMLPool	0 → 1	0	0	basic_a1	Resizing

# X12 to XML conversion – Portal - 5

- A job is created:

The screenshot shows the 'Jobs' page for a batch account named 'mberbatch17'. The left sidebar contains navigation links: Properties, Quotas, Storage account, Keys, and Locks. The main area has a toolbar with 'Add', 'Columns', and 'Refresh' buttons. Below the toolbar is a dropdown menu set to 'All jobs' and an 'Advanced query' section. A table displays the job details.

ID	STATE	POOL	CREATED
X12toXMLJob	Active	X12toXMLPool	Jan 10, 21:26:48

# X12 to XML conversion – Portal - 6

- Tasks are added to the job:

X12toXMLJob - Tasks

Search (Ctrl+/)

Overview

GENERAL

Properties

Environment settings

Metadata

✓ Tasks

Preparation tasks

Release tasks

SETTINGS

Priority

Constraints

Pool information

Auto complete settings

+ Add

Columns

Refresh

Task counts: Active: 9, Running: 0, Completed: 0, Succeeded: 0, Failed: 0

All tasks

Advanced query

Filter by task ID

TASK	STATE	CREATED	EXIT CODE
topNtask0	Active	Jan 12, 19:47:48	
topNtask1	Active	Jan 12, 19:47:48	
topNtask2	Active	Jan 12, 19:47:48	
topNtask3	Active	Jan 12, 19:47:48	
topNtask4	Active	Jan 12, 19:47:48	
topNtask5	Active	Jan 12, 19:47:48	
topNtask6	Active	Jan 12, 19:47:48	
topNtask7	Active	Jan 12, 19:47:48	
topNtask8	Active	Jan 12, 19:47:48	

Martin Bertrand

23

# X12 to XML conversion – Portal - 7

- As tasks complete the conversion, result files are uploaded to the Storage Account, in the “output” container:

The screenshot shows the Azure Storage Portal interface for a container named 'output'. The left sidebar shows a list of containers: 'application', 'input', and 'output' (which is selected and highlighted in blue). The main area displays the 'output' container with a search bar and a table of blobs. The table has columns for NAME, MODIFIED, ACCESS TIER, BLOB TYPE, SIZE, and LEASE STATE. There are 9 rows of data, each representing a file named 'File1\_1.xml' through 'File9\_1.xml'. All files are of type 'Block blob' and are 'Available'.

NAME	MODIFIED	ACCESS TIER	BLOB TYPE	SIZE	LEASE STATE
File1_1.xml	1/20/2018, 3:22:55 PM	Hot (Inferred)	Block blob	3.34 KiB	Available
File2_1.xml	1/20/2018, 3:23:01 PM	Hot (Inferred)	Block blob	6.02 KiB	Available
File3_1.xml	1/20/2018, 3:22:57 PM	Hot (Inferred)	Block blob	4.75 KiB	Available
File4_1.xml	1/20/2018, 3:22:47 PM	Hot (Inferred)	Block blob	6.28 KiB	Available
File5_1.xml	1/20/2018, 3:22:32 PM	Hot (Inferred)	Block blob	3.34 KiB	Available
File6_1.xml	1/20/2018, 3:22:51 PM	Hot (Inferred)	Block blob	3.34 KiB	Available
File7_1.xml	1/20/2018, 3:22:43 PM	Hot (Inferred)	Block blob	3.34 KiB	Available
File8_1.xml	1/20/2018, 3:22:39 PM	Hot (Inferred)	Block blob	3.34 KiB	Available
File9_1.xml	1/20/2018, 3:22:35 PM	Hot (Inferred)	Block blob	3.34 KiB	Available



# X12 to XML conversion – Results

- Result files are downloaded from the “output” container of the Storage Account by the administration VM.
- If all goes well, these will now be XML representations of X12 files.

```
<?xml version="1.0" ?>
<Interchange>
  <AuthorizationInformation id="" qualifier=""/>
  <SecurityInformation id="" qualifier=""/>
  <Sender id="SENDER" qualifier="ZZ"/>
  <Receiver id="RECEIVER" qualifier="ZZ"/>
  <DateTime date="041201" time="1200"/>
  <EdiControlInformation number="000000101" standards_id="U" version_number="00305"/>
  <AcknowledgementRequested id="1"/>
  <TestIndicator id="P"/>
  <FunctionalGroup>
    <FunctionalIdentifier code="PO" name="Purchase Order"/>
    <Sender id="SENDER"/>
    <Receiver id="RECEIVER"/>
    <DateTime date="041201" time="1200"/>
    <Control number="101"/>
    <EdiIndustryIdentifier code="X" id="003050"/>
    <TransactionSet>
      <Id code="850" name="Purchase Order"/>
      <ControlNumber value="000000101"/>
      <PoInfo>
        <Purpose code="22" name="Information Conv"/>
      </PoInfo>
    </TransactionSet>
  </FunctionalGroup>
</Interchange>
```

# Troubleshooting Batch

- The administration script only shows the tasks are completed, but does not show the exit code. While it is running, the status and exit codes can be seen in the console:

X12toXMLJob - Tasks

Search (Ctrl+J)

Overview

GENERAL

Properties

Environment settings

Metadata

✓ Tasks

Preparation tasks

Release tasks

SETTINGS

Priority

Constraints

Pool information

Auto complete settings

+ Add

Columns

Refresh

Task counts: Active: 7, Running: 0, Completed: 2, Succeeded: 0, Failed: 2

All tasks

▼

📄

🗑️

Advanced query ▼

Filter by task ID

TASK	STATE	CREATED	EXIT CODE
topNtask0	Completed	Jan 12, 20:36:19	1
topNtask1	Active	Jan 12, 20:36:19	
topNtask2	Completed	Jan 12, 20:36:19	1
topNtask3	Completed	Jan 12, 20:36:19	1
topNtask4	Active	Jan 12, 20:36:19	
topNtask5	Active	Jan 12, 20:36:19	
topNtask6	Active	Jan 12, 20:36:19	
topNtask7	Completed	Jan 12, 20:36:19	1
topNtask8	Active	Jan 12, 20:36:19	

Martin Bertrand

26

# Troubleshooting Batch - 2

- You can verify the properties of a task, even after it is completed.
- The most crucial property is the command line.

The screenshot shows a web-based interface for viewing task properties. The title bar reads 'topNtask0 - Properties'. On the left, there is a sidebar with a search bar labeled 'Search (Ctrl+)' and two menu items: 'Overview' and 'Properties' (which is highlighted). The main content area is titled 'General' and contains three properties: 'ID' with the value 'topNtask0', 'Display name' (empty), and 'Command line' with the value '/bin/bash -c 'set -e; set -o pipefail; python \$AZ\_BATCH\_NODE\_SHARED\_DIR/gen\_parser.py --filepath File4.X12 --stora...'. Each property value is displayed in a light gray box with a blue document icon to its right. A 'Refresh' button is located at the top of the main content area.

- Ensure the command line is correct and the number of arguments to the application is correct.

# Troubleshooting Batch - 3

- It is also possible to view certain files on the node that ran a task.

topNtask0 - Files on node

Columns Refresh

Filter by file name ...

FILE NAME	SIZE	CONTENT TYPE	LAST MODIFIED
stderr.txt	0 Bytes	text/plain	Jan 12, 19:47:58
wd/File4.X12	908 Bytes	application/octet-stream	Jan 12, 19:47:58
stdout.txt	74 Bytes	text/plain	Jan 12, 19:47:58

Overview

GENERAL


Properties

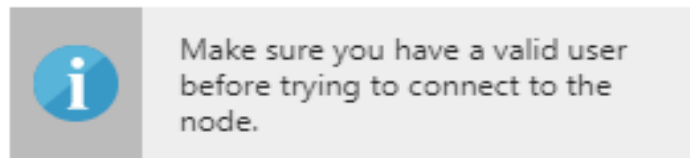
Environment settings


Application packages


- The most useful for debugging are “stderr.txt” and “stdout.txt”, which contain the output of the conversion script.


# Troubleshooting Batch - 4


- It is also possible to SSH connect directly to a node for further troubleshooting and tests.
- Select “Connect” in the sub-menu displayed when clicking on the  icon.
- You will have the option of adding a user.
- Azure will then display connect information.



Username  
 

IP  
 

Port  
 

SSH command line  
 

# YouTube URLs, GitHub URL, Last Page

- Two minute (short):
- 15 minutes (long):
- GitHub Repository with all artifacts: