Azure generated batch file - patterns

Duplicated from: Azure Cloud connection to external node -> MQFTE: - Application Catalog - Confluence (kroger.com)

Option 1 (Desired state ; not ideal for MVP) - Creating an AZ storage account , setting up new pipelines and new function apps to pick up the file and call a REST API to perform an AZ copy of the file(s) down to the on-prem server location and using TWS scheduler to push it out to MQFTE (Azure Cloud connection to external node -> MQFTE: - Application Catalog - Confluence (kroger.com))

a. Pros:
   i. This solution would replicate what POSH team had implemented and would be consider a reusable pattern
b. Cons:
   i. This will require some heavy engagement with Dev/Sec Ops team and will certainly add risk to our timeline.
   ii. This creates additional resources in Azure that we don't currently have setup in any of our other capabilities – new and require some discovery for devSec ops to understand how to implement this new solution even though POSH already has it completed.
c. SPM updates = med/ heavy
d. Will be consider short-term and throw away solution until POSH replaces POS-HOST with MQFTE process

POSH reference material to assist in replicating this solution in Option 1:

1. https://github.com/krogertechnology/posh-azure-functionapps
2. https://github.com/krogertechnology/posh-restapi
3. https://portal.azure.com/#@kproductivity.onmicrosoft.com/resource/subscriptions/7955bb86-14ca-4ca9-a6b9-978136873edd/resourceGroups/rg-posh-deva52-eastus2/overview
4. https://github.com/krogertechnology/posh-azure-terraform
5. https://confluence.kroger.com/confluence/display/POSHOST/POSH+System+Home
6. https://confluence.kroger.com/confluence/pages/viewpage.action?pageId=130522877

Option 2 (possibly, but dependency is on ETL team) - Create an ETL process to pick up the file from a storage account and transfer it over to an on-prem server location using AZ copy

a. Pros:
   i. This solution would replicate what the Data Estate team implemented and remove having to deal with creating new functions, pipeline and REST API.
   ii. Less involvement with Dev/Sec Ops team

a. Cons:
   i. Leverages ETL process, not preferred
   ii. Need to create a new ETL, which is something it sounds like we're going to get push-back from the team that handles this.

NOTE I have something setup with Gus Stropolis next Monday to explore our opportunity to move forward with this option and get this prioritize for SPA so we can make it for MVP

   i. Need to create nw firewall and security to access the AZ script to copy the file back into on-prem. This might delay the timeline as well
   ii. SPM updates = med/heavy
   iii. Will be consider short-term and throw away solution until POSH replaces POS-HOST with MQFTE process

ETL reference material to assist in Option 2:

azcopy_wrapper - Systems Integration Solutions - Confluence (kroger.com)

AZ copy script on unix server.docx - *Per Gus Stropolus this is the AZ copy script that would need to be installed on the unix server to perform the copy function from Azure. Uses Azure CLI scripts to perform the copy and other various commands: upload, download, list, delete*

Option 3 (seems more suitable, less dependency on other teams) – Similar to Option 1, but shifting the REST API into our own Azure Edge Node and eliminating the need to use function or create new pipelines to support the functions per division

a. Pros:
   i. This solution will somewhat still replicate the POSH architecture to a smaller scale, but quicker to implement since we don't need to create new Azure functions and pipeline
   ii. Quicker to implement. Per Kiran – this might be more of a 3-4 weeks effort vs. more than 1 month effort in option 1 and 2 since we will own the API development work in SpringBoot
   iii. Aligns with modern tech by utilizing REST API + additional Cloud base tools
b. Cons:
   i. Still requires engagement with Dev/Sec Ops

ii. Still need to create a deployment process to deploy the API jar file into the Edge Node – need some additional discovery discussions with Suvarna (POSH SME) next week to help aid in this topic.
iii. SPM updates = med
iv. Will be consider short-term and throw away solution until POSH replaces POS-HOST with MQFTE process

Option 4 (not applicable/ no accessData Lakes available for SPA team to use) - Standing up a shared server on -prem and using Data Lakes to create a process and storage blob to pull the data/files to/from on-prem and Azure cloud

a. Pros - have not vetted out due to time constraints to look into this option
b. Cons- have not vetted out due to time constraints to look into this option

Option 5 (NEW*** - created on 8/25 - guidance from Samir and Manoj) - Use existing B2B service to call into on-prem landing zone.   Additional details to perform setup can be found on this page here :**

Option 5 - Usage of B2B service on-prem and create a file creation process in kubernetes cluster on prem

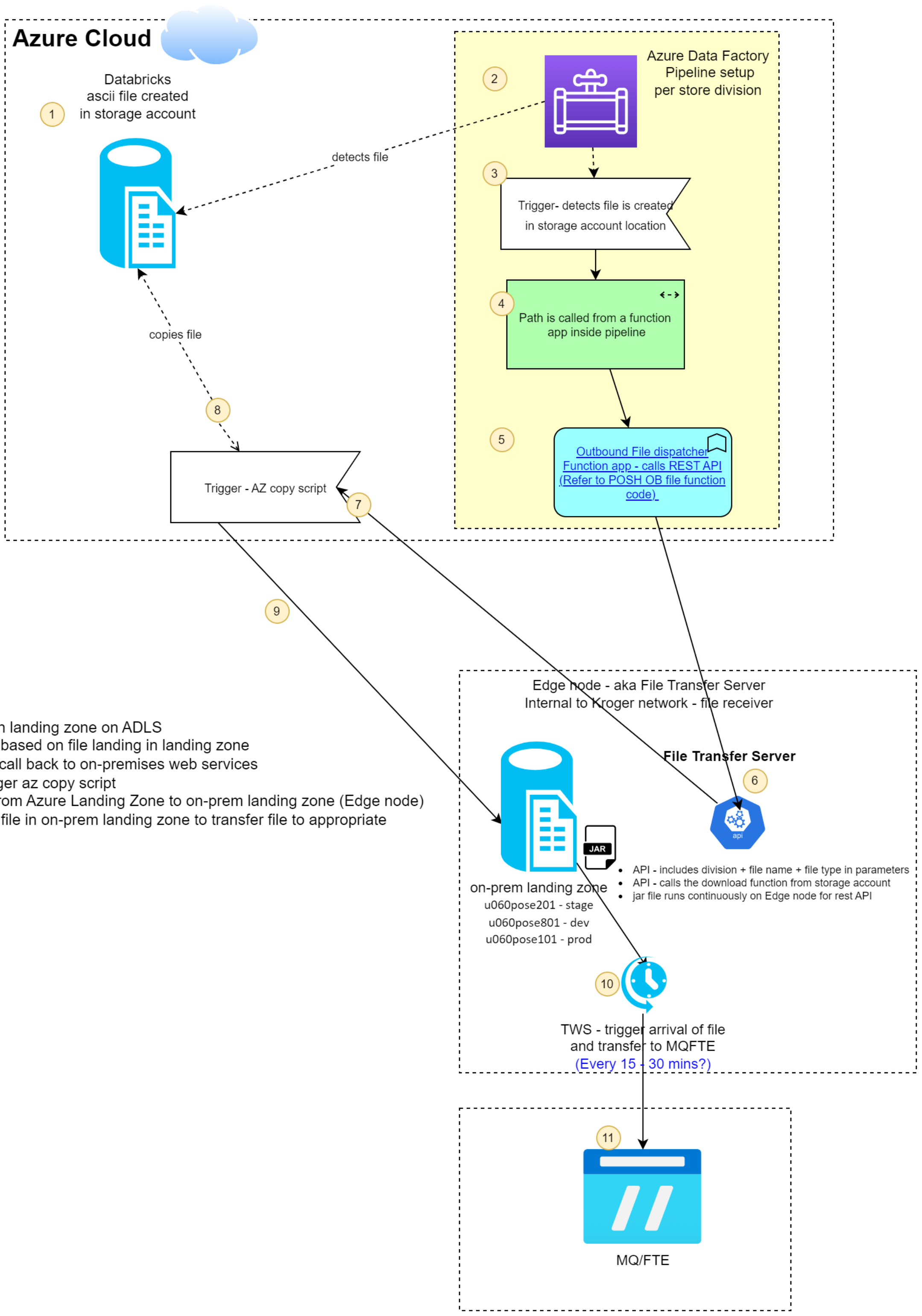
1. Pros:
    a. least amount of setup required since B2B service is already made available
    b. allows the process to create the ASCII file to be handle within on-prem vs in the Azure process
    c. allows the setup of the file creation to be created in an on-prem kubernetes cluster that is already setup out there for enterprise usage
2. Cons
    a. Still require some effort of dev-ops support
    b. Need to update SPM to get server access to B2B service - low risk
    c. Will be consider short-term and throw away solution until POSH replaces POS-HOST with MQFTE process

Other options explored, but not considered to be the best approach for MVP (Sept/Oct):

1. ***Usage of a direct ODBC connection between on-prem and Azure DB - this was consider a read-back mode and is not a pattern we want to implement***
2. ***Usage of event bus to pubish to DESP (pass through) and push down to MQFTE - this option required more dependency on numerous teams and adding more hand-offs into the process***

Option 1 - replicate POSH architecture:

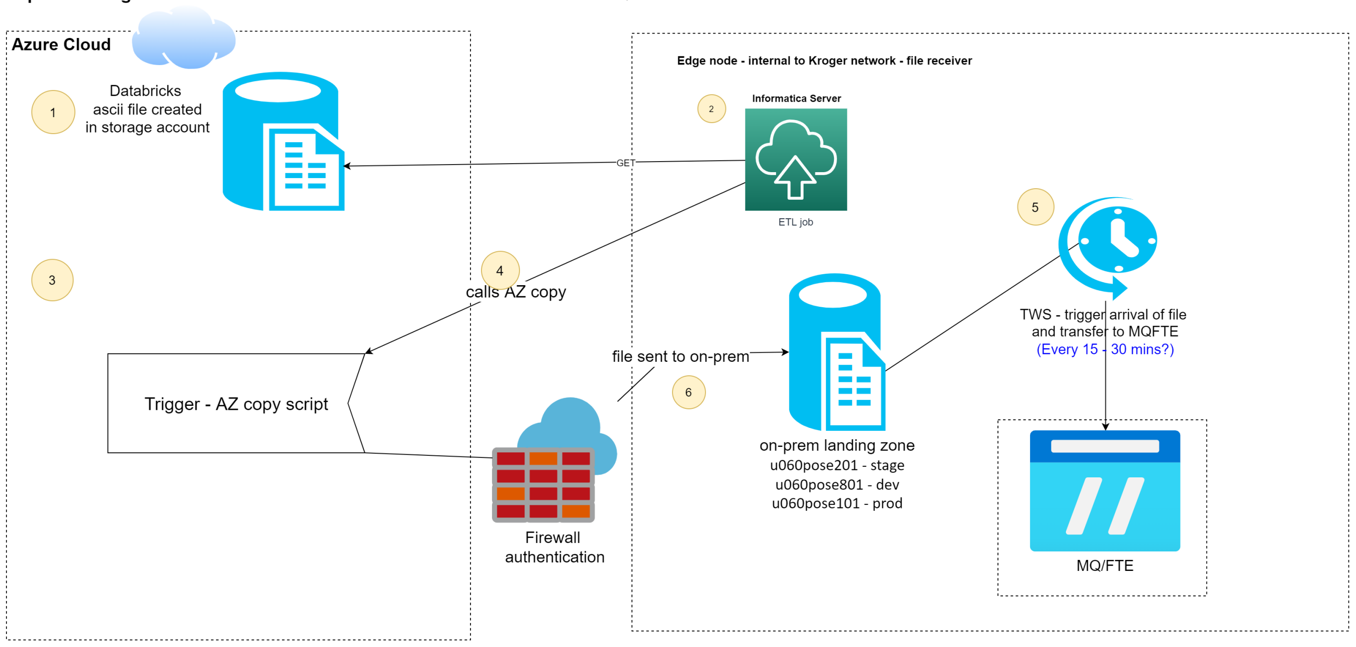
Option 1 - Tag order - Azure cloud transfer API call to External Node / MQFTE


Azure Cloud
1
Databricks
ascii file created
in storage account
2
Azure Data Factory
Pipeline setup
per store division
detects file
3
Trigger- detects file is created
in storage account location
4
Path is called from a function
app inside pipeline
copies file
8
5
Outbound File dispatcher
Function app - calls REST API
(Refer to POSH OB file function
code)
Trigger - AZ copy script
7
9
Edge node - aka File Transfer Server
Internal to Kroger network - file receiver
File Transfer Server
6
api
JAR
API - includes division + file name + file type in parameters
API - calls the download function from storage account
jar file runs continuously on Edge node for rest API
on-prem landing zone
u060pose201 - stage
u060pose801 - dev
u060pose101 - prod
10
TWS - trigger arrival of file
and transfer to MQFTE
(Every 15 - 30 mins?)
11
MQ/FTE

Step-by-step Flow:
1. Extract process creates file in landing zone on ADLS
2. ADF triggers an Az Function based on file landing in landing zone
3. Azure function makes a rest call back to on-premises web services
4. on-prem web servce will trigger az copy script
5. AZ copy script will copy file from Azure Landing Zone to on-prem landing zone (Edge node)
6. TWS will trigger on arrival of file in on-prem landing zone to transfer file to appropriate MQFTE agent

Option 2 - replicate Data Estate Team by using an modified ETL process that can copy data from cloud into on-prem (Hadoop_ETL_EDW)

Option 2 - Tag order - Azure cloud transfer API call to External Node / MQFTE

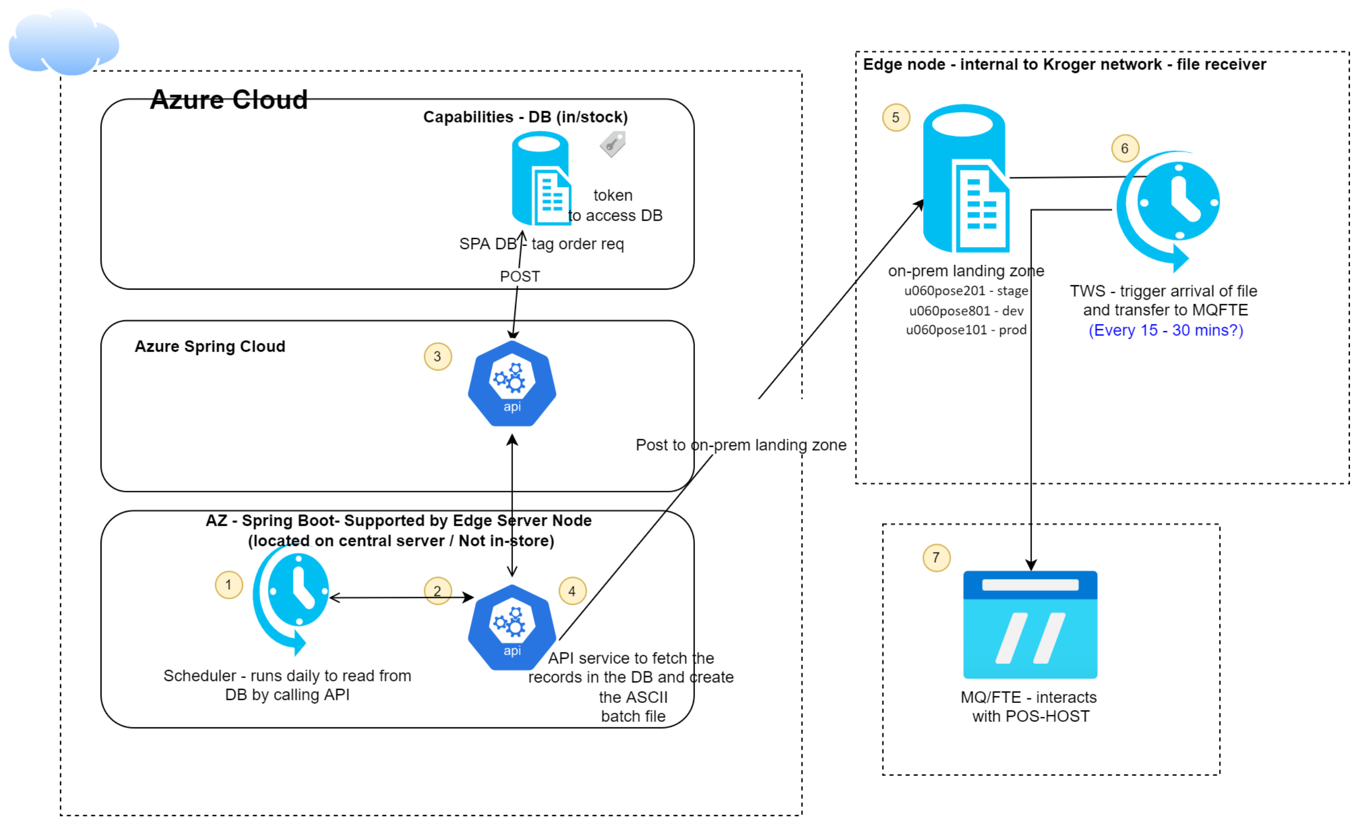


Step-by-step Flow:
1. ETL job runs to look at storage account to determine if any files are present
2. If files are present then call the AZ copy to copy it down to on-prem landing zone
3. TWS will trigger on arrival of file in on-prem landing zone to transfer file to appropriate MQFTE agent

Option 3 - Similar to Option1, but moving the API into Edge Node and build under Spring Boot. Additionally, removing functions/pipeline creations to simplified the approach vs. doing the exact pattern as POSH.

1. SPA to call into the DB with a request via API
2. A response will then get generated to create the file to push out to on-prem
3. Need to determine authentication between Azure Node and On-Prem
   a. Refer to Step 4 - request for token using a post action (example below shared by Rohith Katakam)
      ```
      curl --location --request POST 'https://uat.instock.kroger.biz/spsauth/v2/access/token' \

      --header 'accept: application/json' \

      --header 'X-Correlation-Id: 1234' \

      --header 'Content-Type: application/json' \

      --header 'X-Token-Type: api' \

      --data-raw '{

          "grant_type": "client_credentials"

      }'
      ```
   b. Scope added to authenticate on-prem app
   c. Add o-auth into audience to provision edit access into on-prem application - verify with Manoj Suman how this is setup

Option 3 - Tag order - Azure cloud transfer API call to External Node / MQFTE using Azure SPA - Spring boot API

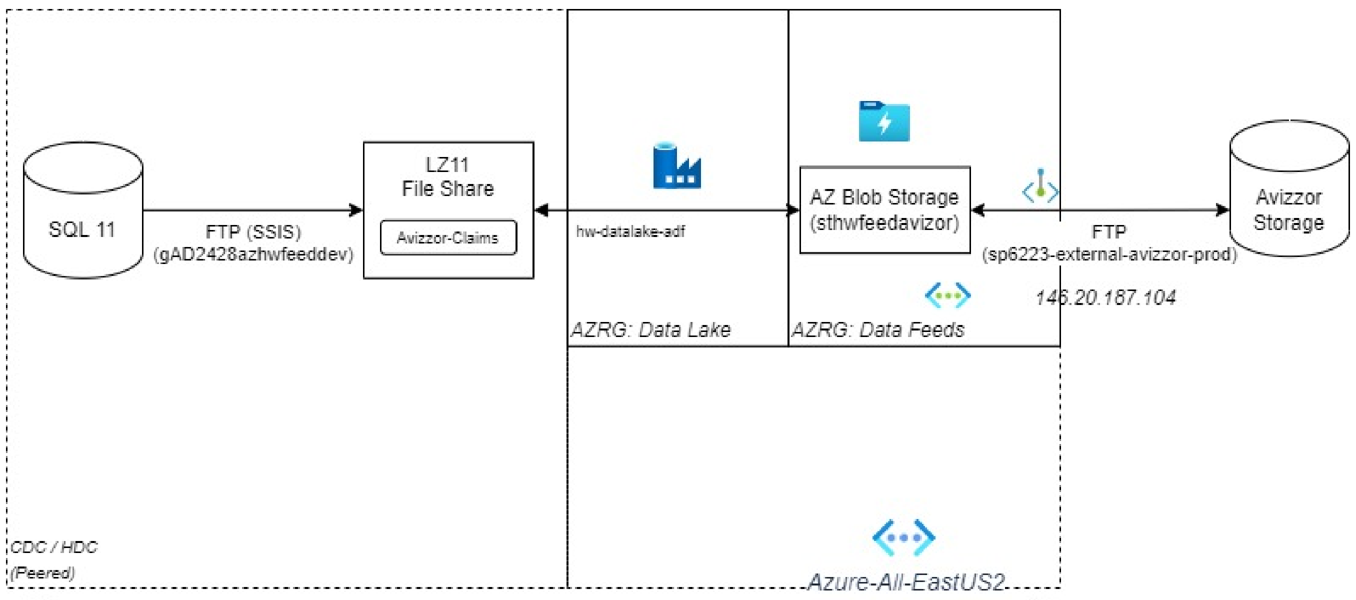

Azure Cloud
Capabilities - DB (in/stock)
token
to access DB
SPA DB - tag order req
POST
Azure Spring Cloud
3
api
Post to on-prem landing zone
AZ - Spring Boot- Supported by Edge Server Node
(located on central server / Not in-store)
1
2
4
api
Scheduler - runs daily to read from DB by calling API
API service to fetch the records in the DB and create the ASCII batch file
Edge node - internal to Kroger network - file receiver
5
6
on-prem landing zone
u060pose201 - stage
u060pose801 - dev
u060pose101 - prod
TWS - trigger arrival of file and transfer to MQFTE
(Every 15 - 30 mins?)
7
MQ/FTE - interacts with POS-HOST

Step-by-step Flow:
1. Records are created in the SPA DB
2. Cron Job - scheduler to read from DB via API
3. API calls out into the DB to retrieve the records to support tag order requests.
API and jar file will be built on the Azure Spa Edge Node.
4. API will create an ascii Batch file, which is then pushed over to the on-prem landing zone
5. API (Edge server) will request for a token to pass to header
6. Token will establish the authentication between Edge node and landing zone
7. File in landing zone will trigger an agent that is pulled into the POS-HOST MQFTE

Access Requirement
1. Create APT to resource group establish to access Edge Server Node
2. Update SPM to include the Azure Spring Cloud

Option 4 - Using Azure File Share to perform copy to on-prem. This solution is still in trila under the Health & Wellness team (Joseph Rudnicki)

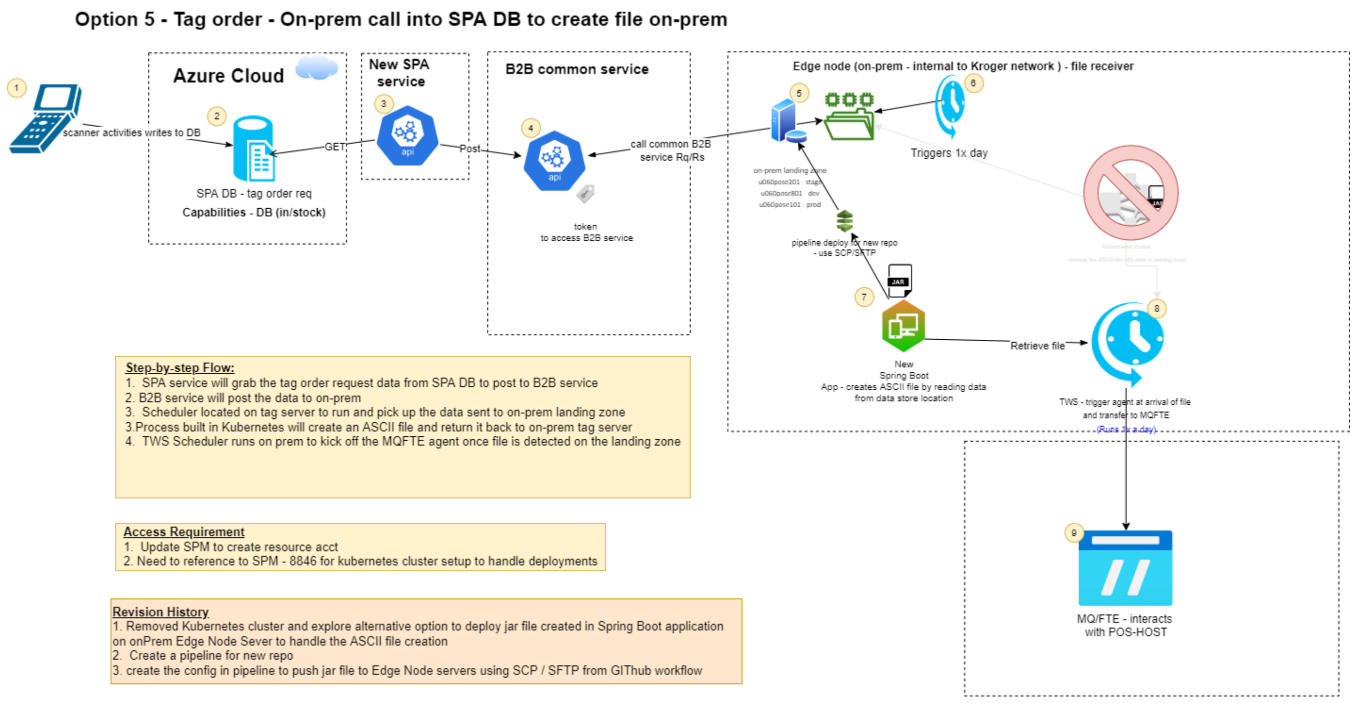
1. Uses Az Data Lake to run a copy activity to copy data into File Share server on-prem
2. ADF needs a **self-hosted** integration runtime (IR) to connect any copy activities between cloud stores & the on-prem network.
   - This IR needs to be hosted on a VM or server that allows reach-back between Azure and KTD networks (i.e. network peered).
3. Azure Data Factory needs to have a Windows Service Account created, so it can be joined with the on-prem resources with Active Directory.
   - By default the ADF Managed Identity only exists in Azure AD – Kroger has a hybrid AD / AAD setup, so you need to create an identity in AD to run any on-prem jobs.



Process Overview:

- BI ETL (SSIS) PGP encrtypts CPR + Claims data from BI SQL 11 Server
- BI ETL FTP drops encrypted files to LZ11 (5:00am)
- Data Lake ADF Copies .pgp intpo Data Feeds Storage (sthwfeedavizzor) (6:00 am)
- Avizzor (B2B) copies .pgp files via Service endpoint / Service Principal (7:00 am)
    - drops receipt / recon files to Data Feeds Azure storage (7:00 am)
- Data Lake ADF Copies reconcilliation files to LZ11 (8:00 am)
- BI ETL picks up recon files from LZ11 (9:00 am)

<u>Option 5 - Using B2B service from on-prem to Azure</u>


Option 5 - Tag order - On-prem call into SPA DB to create file on-prem
1
scanner activities writes to DB
Azure Cloud
2
SPA DB - tag order req
Capabilities - DB (in/stock)
GET
New SPA service
3
api
Post
B2B common service
4
api
token
to access B2B service
call common B2B service Rq/Rs
Edge node (on-prem - internal to Kroger network) - file receiver
5
on-prem landing zone
u060posc201 - stage
u060posc801 - dev
u060posc101 - prod
6
Triggers 1x day
pipeline deploy for new repo - use SCP/SFTP
JAR
7
New
Spring Boot
App - creates ASCII file by reading data from data store location
Retrieve file
8
TWS - trigger agent at arrival of file and transfer to MQFTE
(Runs 1x a day)
9
MQ/FTE - interacts with POS-HOST

<u>Step-by-step Flow:</u>
1. SPA service will grab the tag order request data from SPA DB to post to B2B service
2. B2B service will post the data to on-prem
3. Scheduler located on tag server to run and pick up the data sent to on-prem landing zone
3.Process built in Kubernetes will create an ASCII file and return it back to on-prem tag server
4. TWS Scheduler runs on prem to kick off the MQFTE agent once file is detected on the landing zone

<u>Access Requirement</u>
1. Update SPM to create resource acct
2. Need to reference to SPM - 8846 for kubernetes cluster setup to handle deployments

<u>Revision History</u>
1. Removed Kubernetes cluster and explore alternative option to deploy jar file created in Spring Boot application on onPrem Edge Node Sever to handle the ASCII file creation
2. Create a pipeline for new repo
3. create the config in pipeline to push jar file to Edge Node servers using SCP / SFTP from GIThub workflow