**Trading Alert System Architecture**

A screenshot of a computer

AI-generated content may be incorrect.

1. TETCO scrapper function is a timer triggered function that scrapes the TETCO website for critical and planned outage notices. This function can be configured to set time to run.
   1. Once the notices are scraped the notice text is stripped off any html elements and only the text is retrieved.
2. TETCO, Send Alert and Notice Processor functions read their configuration settings from an instance of Azure App Config (except for when configuring event hub connection string, event hub name and consumer group name when used as an input parameter for Event Hub triggered function)
3. Azure App Config can be configured with key vault so that it can read secrets directly from the key vault. This is not implemented in this solution but is a best practice. E.g.: When reading a connection string the key can reside in azure app config and azure app config can be configured to read the value from key vault
4. Once any of the scrapper reads the notice text it stores the reference to notice in Azure table storage so that next time it runs it will parse everything but will send new notices to the next function for processing. This is not implemented.
5. Azure function send the notice text with html stripped off to an event hub for further processing.
6. Notice Processor is an event hub triggered function which receives and processes the notice events received from scrapper function.
7. Notice Processor send the notice text to OpenAI and instructs it to identify a trading signal, if it is a trading signal, OpenAI is instructed to create a html formatted email for the signal.
8. Email formatted as html is posted to an event hub for further processing.
9. Send Alert is another event hub triggered function which processes the email html from the event hub.
10. HTML text from the event is sent as an email to destination email addresses.

Comments:

1. While designing this architecture more emphasis was laid on using as many as Azure PaaS components for its implementation
2. Azure PaaS components offer scalability along with facility of many built-in security features like private end point, using Microsoft Entra ID to authenticate to name a few
3. Azure functions can be run on consumption plan which is highly scalable and offers pre-warmed instances to eliminate cold start issues
4. Azure event hub is a high-performance data streaming service in cloud which can ingest million of events per second and is relatively cheap even if it is used to ingest few hundred events with very low latency. Event hubs work in a pub-sub model having multiple subscribers and publishers and has very powerful SDK’s available in almost all major coding languages. Azure Service Bus is another choice and it offers ordered delivery of message if that’s the business need
5. Azure app config provides a one place for all the configs of the applications deployed in Azure there by eliminating maintaining in app config files or using the environment variables. Additionally, app config can make a PaaS-to-PaaS connection within cloud to key vault to retrieve secrets
6. OpenAi which powers ChatGpt is used in this implementation which accepts the prompts and returns the results. Azure Open AI resource which offers use of multiple LLM models including the latest available 4.1 model. Given these benefits and the availability of SDK’s for C# OpenAI was chosen to integrate with this solution.