Automation Accounts: Distributed Job Scheduler

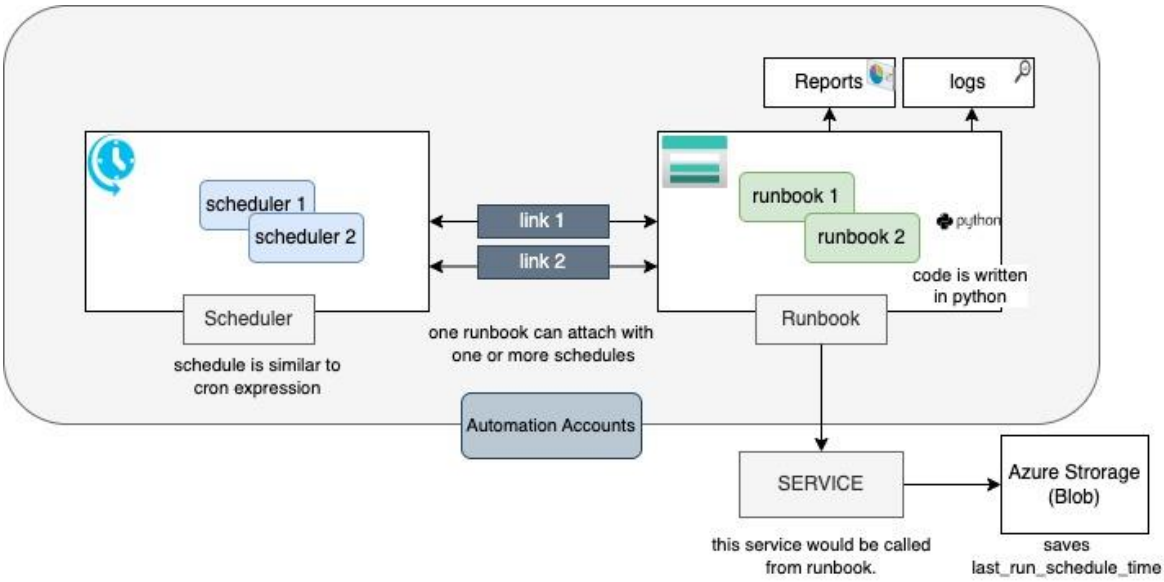
Problem Statement

Every business undergoes a major transformation wherein the integration of different systems like CRM, ERP, and internal & external channels happen. This is being done to achieve better traceability, transparency, & real-time insights. In order to maintain the eventual consistency of data across components, we needed a more reliable and stable configurable job scheduler considering 20+ runbooks (or distinct Jobs) running at different configurable frequencies.

Solution/Architecture

Having looked at different solutions, we released Azure offering - Automation Accounts a fit to requirements. This came as a suggestion from teammate (Naman Shukla). We evaluated and in minimum time constraints, we shipped to production.

Currently, we have 20+ runbooks (or distinct jobs) running each at different frequencies depending on business requirements, and a total of 100+ jobs run in a day with zero failure rates.



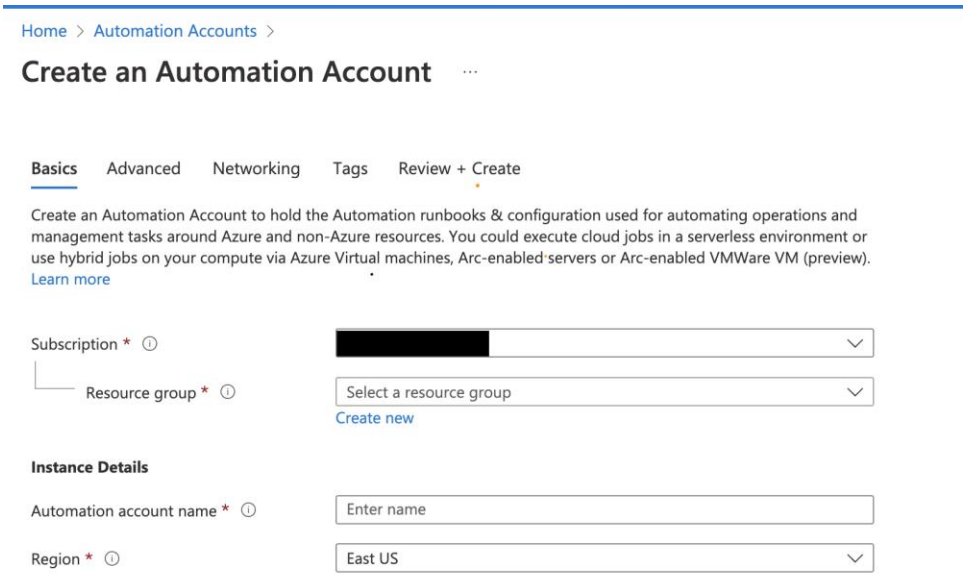
Above is the architecture of how we integrated Azure Automation Accounts for our Job Scheduler. It has three major components -

a. Scheduler
    i. This is a place where schedules are written e.g., hourly, monthly or specific time.
    ii. This will run your job according to given schedules.
b. Runbook
    i. This is the main code block where the actual job is registered.
    ii. We have written python script to call relevant service when runbook is invoked.
c. Azure Storage
    i. It stores the last run scheduled time.
    ii. Also, it helps re-run of those jobs that based on last processed data using timestamp fetched from Azure blob storage (i).

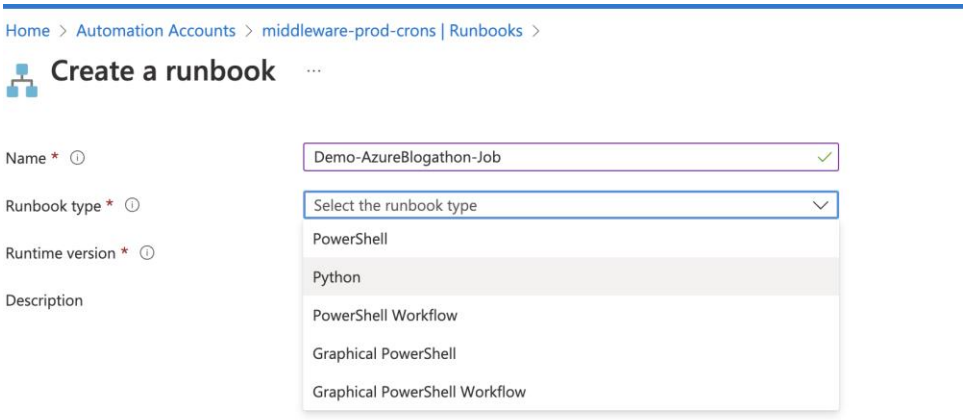
Technical Details and Implementation of solution

Listed down the implementation of Azure Automation Accounts.

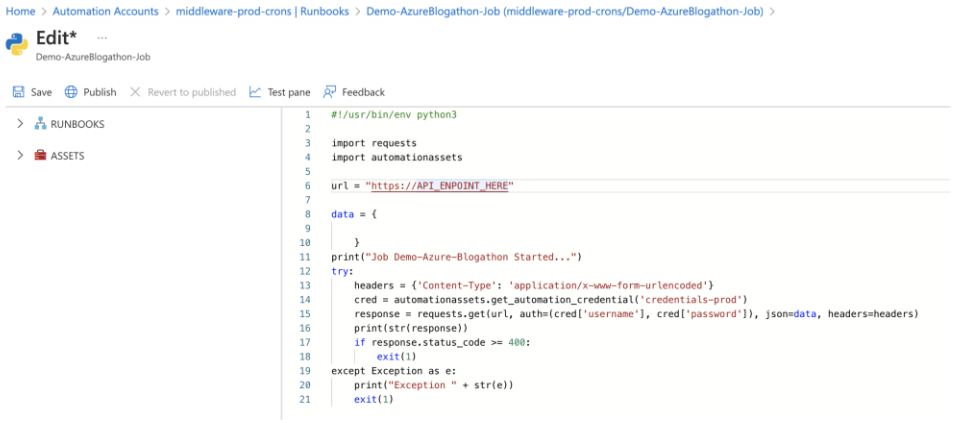
Step-1: Create an Automation Account under your subscription.



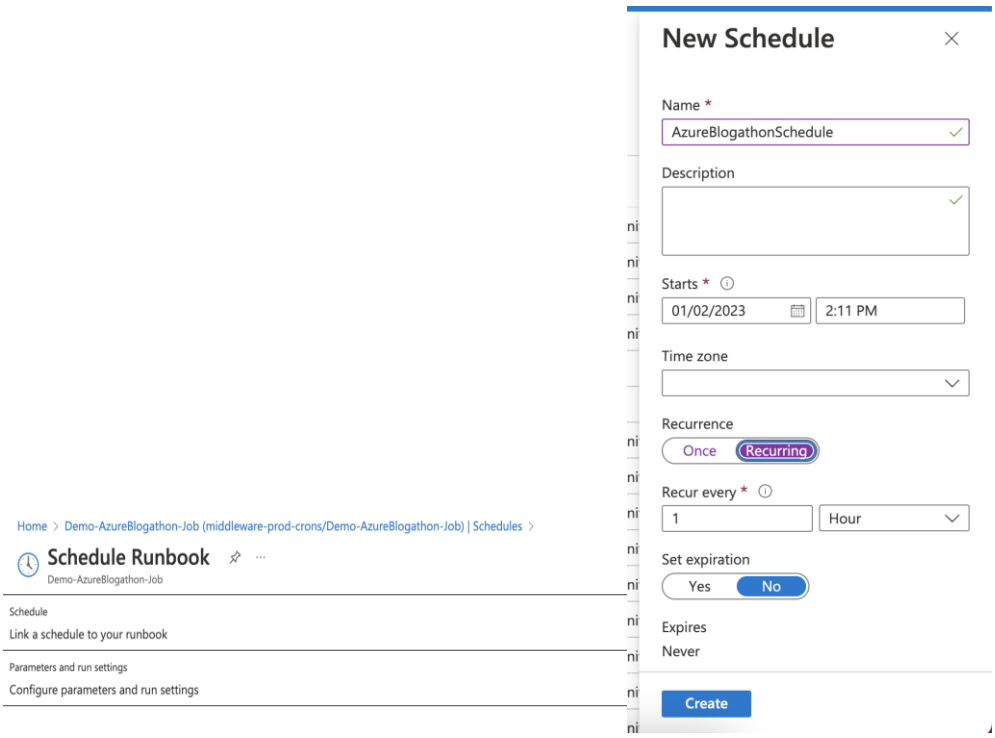
Step-2: Create a Runbook e.g., Demo-AzureBlogathon-Job being created. Remember to select the type. We have chosen Python and chosen the latest version of it.



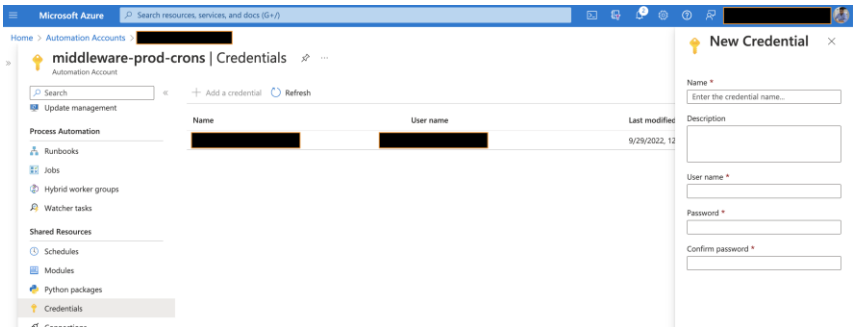
Step-3: Edit your python script as given below. Here we are calling our internal service. First, we fetch the credentials and pass them as headers, and then call our required endpoint. How to get and set up the credentials is given in Step-5.



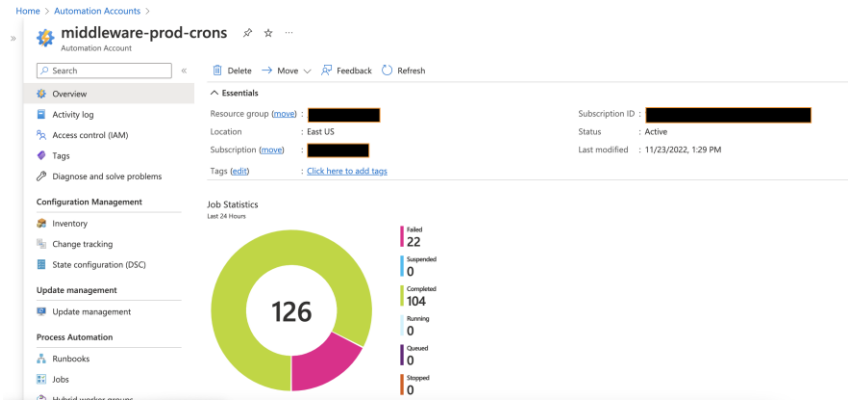
Step-4: Now, create a schedule. This means at what specific time period or frequency we want to run our runbook. First, link a schedule to your runbook and then add your Cron as per the requirement.



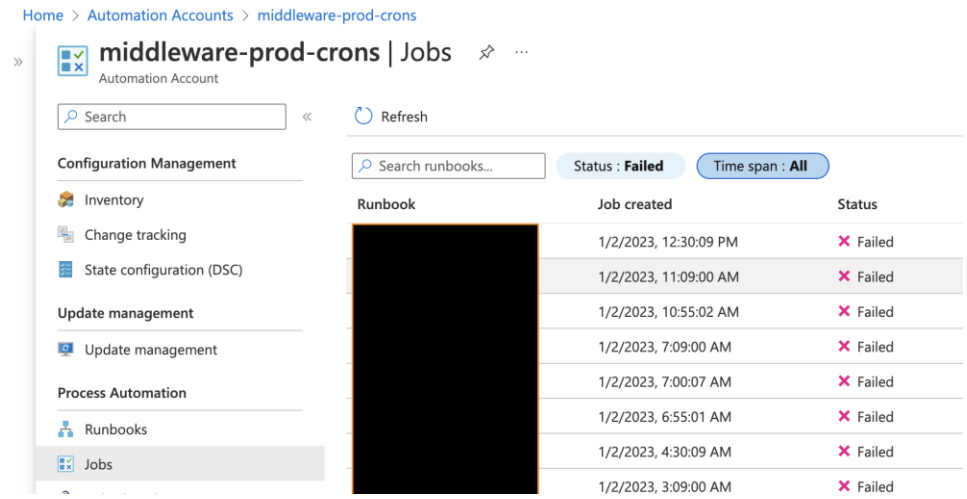
Step-5: Go to the home page of your Automation Account and select credentials in the left panel. Here, you can add your credentials if you have any.



Step-6: Again, go to the overview page of Automation Accounts. Here you can see a holistic report of your Jobs (failed, success, queued etc.)



Step-7: Suppose if any of your Job is running as expected. Go to Process Automation tab and select Jobs. You will see the list of all the Jobs along with status. In order to debug, you can further select particular Job and go to the details page where you can see the exceptions if any.



Challenges

Challenges lie with the predefined coding script in the runbook, if you are not comfortable with python, or PowerShell then you need to learn some basics. Although for this type of Job Schedule, one shouldn't require high skills. Azure already provides a simple interface to work with which is self-explanatory.

It doesn't provide you with a detailed stack trace hence you need to be careful while editing any of your runbooks. The good part is that it maintains the snapshot version.

Business Benefit

There were couple of business advantages we achieved using this approach which are as follows-

1. Faster to develop.
2. Ease of debuggability.
3. Handy reports.
4. Reliable tool.

Anything that reduces developers' time is a huge thumbs-up for any business as it provides businesses with scale at any time. For our business, eventual consistency is necessary because we have 10+ components where data flows to and from. For this, we have chosen the Azure offering.