Postman

- Style Guide
- Running Tests in ADO Release Pipeline
- Environment Variables

Style Guide

Workspace Organization

As a general rule of thumb, we have a workspace in postman for every git repository.

Organizing workspaces in this way...

- keeps the number of collections and environments in check, and thus easier for us to find what we are looking for
- makes it much easier to quickly understand which environments are compatible with a given collection

Environment Variables

Camel Casing

We use ***camel casing*** for naming environment variables. For example, `baseUrl` rather than `BaseUrl`, or `base-url`.

Name Standardization

Some environment variables will likely be used across several different workspaces and collections. For this reason, we want to have consistent names for certain variables. For example, for the domain of a given API we could name a variable several different things that are all great names like `domain`, `origin`, or `baseUrl`, but having a standardized name we use across all collections will help eliminate the need to figure out what the name is used in a particular collection or environment.

Running Tests in ADO Release Pipeline

> Don't put credentials or keys for any Revver user in postman environment variables that have access to real customer data, or any other sensitive data in case these API keys were ever compromised.

You can run a Postman collection with a specific environment in an ADO release pipeline stage. The following steps can be taken to set this up.

1. Click on the ***...*** (three dots) icon next to the collection you want to run in the pipeline, which is outlined in **BLUE** in image 1.
2. Select **Run Collection** option in the dropdown that appears, which is outlined in **YELLOW** in Image 1.
3. In the right panel, in the ***Choose how to run your collection*** section, select **Automate runs via CLI,** which is outlined in **RED** in Image 1.
4. In the bottom of the right panel, click the ***Configure command*** link, which is outlined in **GREEN** in Image 1.
5. You will be taken to a new page.
6. In the ***Choos the collections you want to run*** section, make sure the correct ***Collection*** and ***Environment*** are selected. This is outlined in **ORANGE** in image 2.
    1. You can select multiple collection / environment pairs if you wish
7. In ***CI/CD configuration*** section, make sure ***Azure Pipelines*** and ***Windows*** is selected. This is outlined in **PURPLE** *in* image 2.
8. You will need to generate a new ***API Key*** if you don't already have one setup. You can do this by clicking the ***Generate API Key*** button, outlined in **PINK** in image 2.
9. In the azure pipeline stage that you wish to run the Postman tests in you will need to **Powershell** tasks. The first task will install the Postman CLI on the build agent, and the second task will authenticate with your API key and run the collection. On each task select the ***Inline*** type, and copy the scripts (highlighted in **BROWN** in image 2) into the ***script*** section of the task.
    1. In the second script you will need to replace ***$(POSTMAN_API_KEY)*** with your actual postman API key.

Image 1

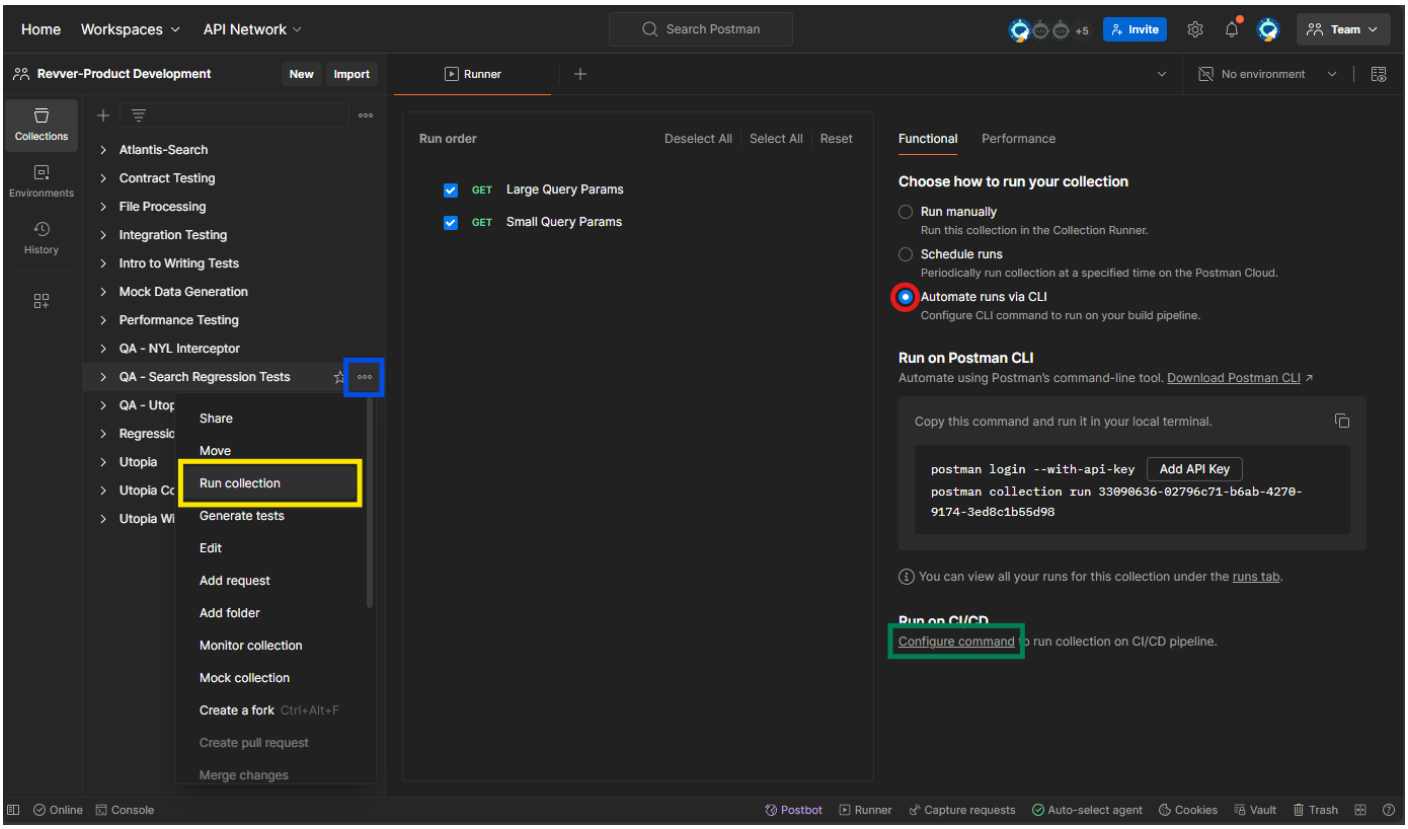

Home
Workspaces
API Network
Search Postman
Invite
Team
Revver-Product Development
New
Import
Runner
No environment
Collections
Environments
History
Atlantis-Search
Contract Testing
File Processing
Integration Testing
Intro to Writing Tests
Mock Data Generation
Performance Testing
QA - NYL Interceptor
QA - Search Regression Tests
Share
Move
Run collection
Generate tests
Edit
Add request
Add folder
Monitor collection
Mock collection
Create a fork Ctrl+Alt+F
Create pull request
Merge changes
Run order
Deselect All
Select All
Reset
GET Large Query Params
GET Small Query Params
Functional
Performance
Choose how to run your collection
Run manually
Run this collection in the Collection Runner.
Schedule runs
Periodically run collection at a specified time on the Postman Cloud.
Automate runs via CLI
Configure CLI command to run on your build pipeline.
Run on Postman CLI
Automate using Postman's command-line tool. Download Postman CLI
Copy this command and run it in your local terminal.
postman login --with-api-key
Add API Key
postman collection run 33090636-02796c71-b6ab-4270-9174-3ed8c1b55d98
You can view all your runs for this collection under the runs tab.
Configure command
Online
Console
Postbot
Runner
Capture requests
Auto-select agent
Cookies
Vault
Trash

Image 2

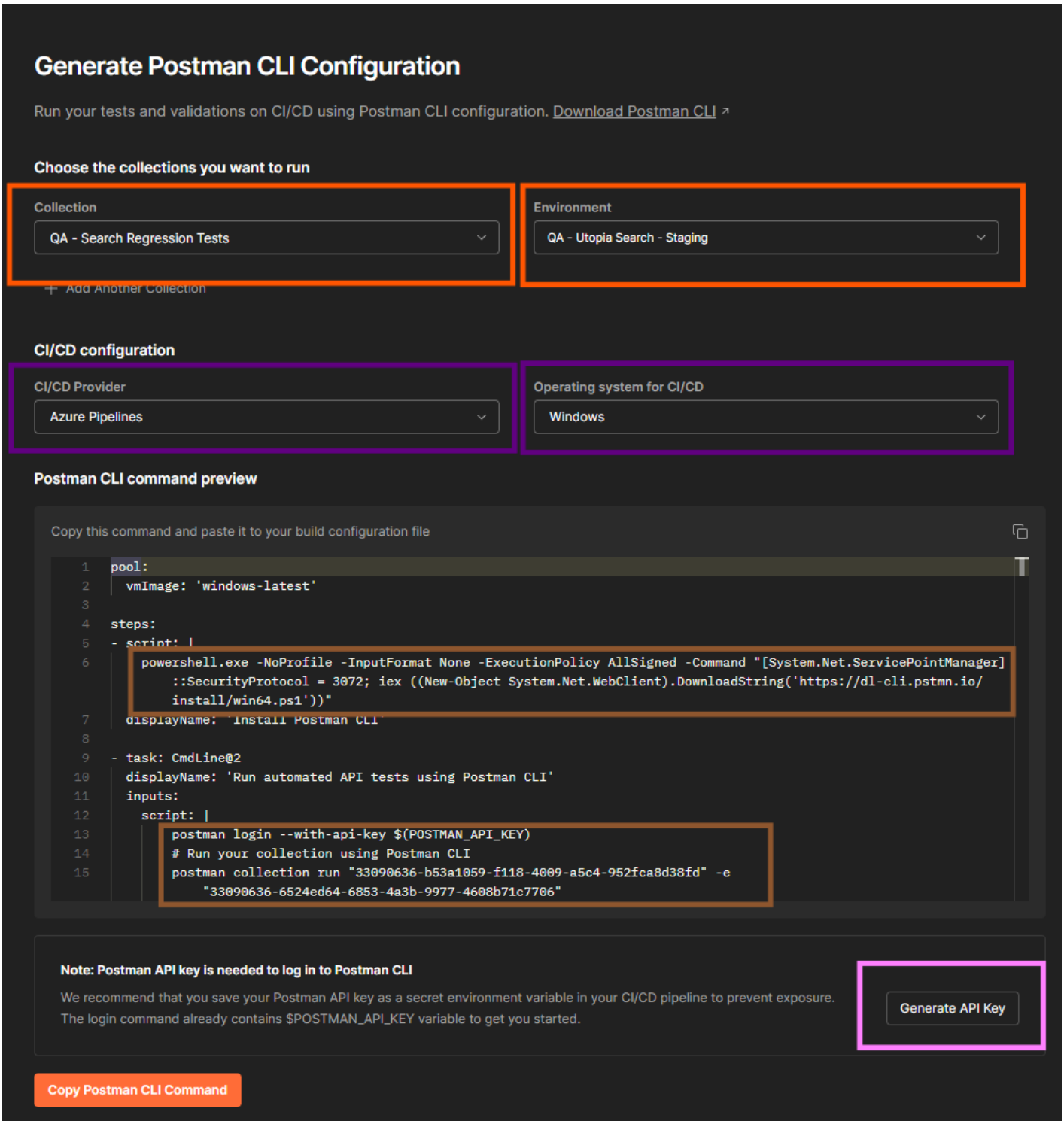
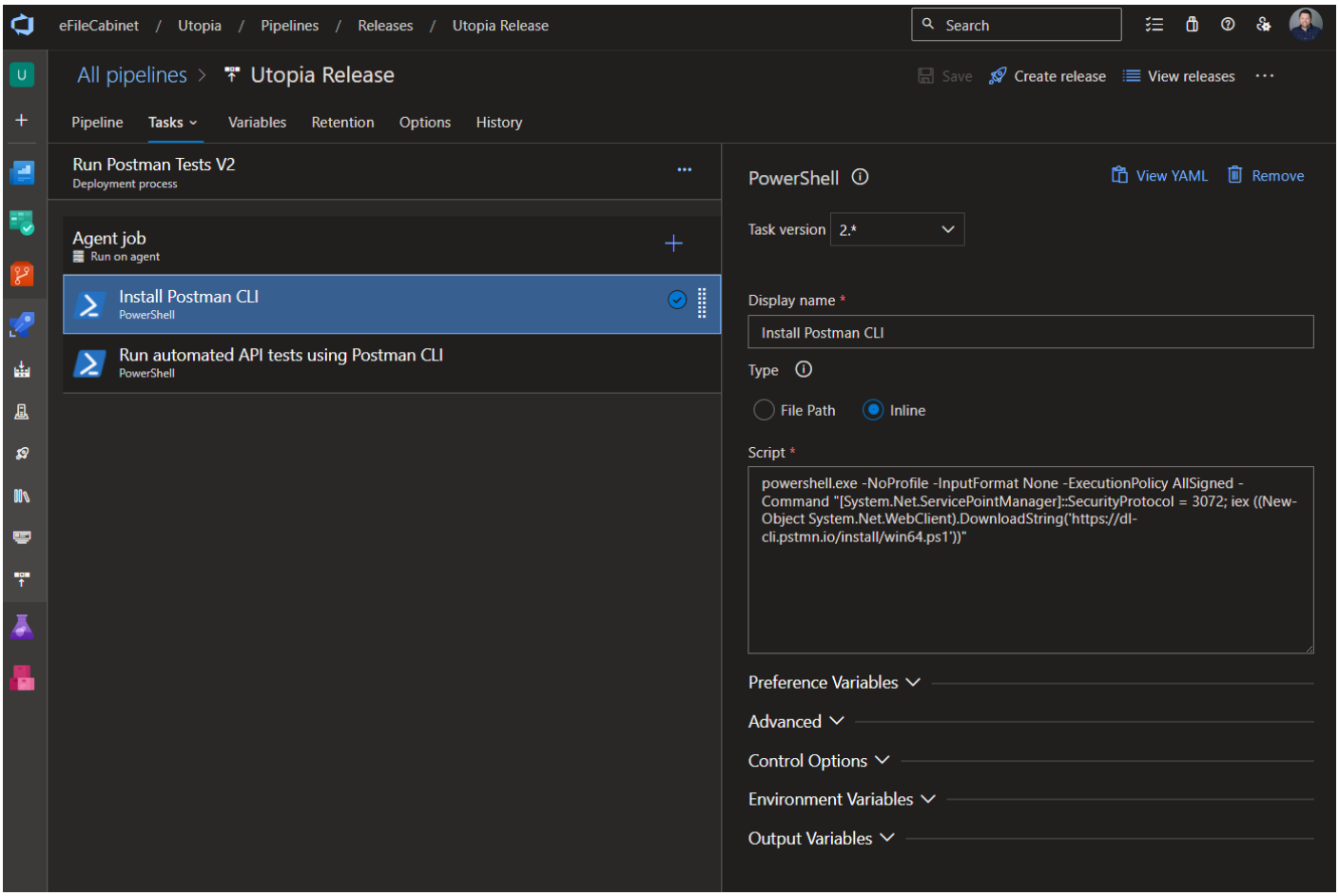

Generate Postman CLI Configuration
Run your tests and validations on CI/CD using Postman CLI configuration. Download Postman CLI
Choose the collections you want to run
Collection
QA - Search Regression Tests
Environment
QA - Utopia Search - Staging
Add Another Collection
CI/CD configuration
CI/CD Provider
Azure Pipelines
Operating system for CI/CD
Windows
Postman CLI command preview
Copy this command and paste it to your build configuration file
pool:
vmImage: 'windows-latest'
steps:
- script: |
powershell.exe -NoProfile -InputFormat None -ExecutionPolicy AllSigned -Command "[System.Net.ServicePointManager]
::SecurityProtocol = 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://dl-cli.pstmn.io/
install/win64.ps1'))"
displayName: 'Install Postman CLI'
- task: CmdLine@2
displayName: 'Run automated API tests using Postman CLI'
inputs:
script: |
postman login --with-api-key $(POSTMAN_API_KEY)
Run your collection using Postman CLI
postman collection run "33090636-b53a1059-f118-4009-a5c4-952fca8d38fd" -e
"33090636-6524ed64-6853-4a3b-9977-4608b71c7706"
Note: Postman API key is needed to log in to Postman CLI
We recommend that you save your Postman API key as a secret environment variable in your CI/CD pipeline to prevent exposure.
The login command already contains $POSTMAN_API_KEY variable to get you started.
Generate API Key
Copy Postman CLI Command

Image 3


eFileCabinet / Utopia / Pipelines / Releases / Utopia Release
Search
All pipelines > Utopia Release
Save
Create release
View releases
Pipeline
Tasks
Variables
Retention
Options
History
Run Postman Tests V2
Deployment process
Agent job
Run on agent
Install Postman CLI
PowerShell
Run automated API tests using Postman CLI
PowerShell
PowerShell
View YAML
Remove
Task version
2.*
Display name *
Install Postman CLI
Type
File Path
Inline
Script *
powershell.exe -NoProfile -InputFormat None -ExecutionPolicy AllSigned -Command "[System.Net.ServicePointManager]::SecurityProtocol = 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://dl-cli.pstmn.io/install/win64.ps1'))"
Preference Variables
Advanced
Control Options
Environment Variables
Output Variables

Environment Variables

after initial review the only consistent variables across the existing environments seem to be:

- baseUrl
- username
- password
- token
- appKey

depending on how auth happens it might be smart to also have as variables as well so that you can handle auth:

- clientId
- clientSecret
- tokenUrl