**HACKATHON - Tinkerers Lab**

**Packaged Application Delivery Pipeline Suite**

A robust CI/CD pipeline solution for any third-party platform to streamline the end-to-end software delivery lifecycle.

Current Walmart CICD pipelines primarily support the standard application development and deployment journey via kitt, looper pro, infinity, WCNP.  There are a lot of packaged application based solutions utilized within Walmart like SAP, Workday, Success Factors, Service Now, etc. that either do not have any defined CICD pipelines or are unable to follow Walmart standards and processes. The vision is to build a pipeline solution/starter kit that can be easily plugged with 3rd party packaged applications and enable a standard application delivery experience. Developers should be able to interact with the pipleine via Concord, Wibey or Copilot.

Flow:

GIT hub (CPI) (dev) -> webhook -> Concord -> PR to test branch -> once merged

GIT hub (IIB) -> webhook -> concord -> looper pro -> Artifactory -> Concord -> deployment

Concord: It’s an orchestration application, to decide which application should call first and what is the next this all defined here only

Looper pro: this is useful for doing build for any code, similar to concord but used mostly for build purpose

Artifactory: this is the place where build files are stored like bar or jar or any files. Even in Walmart the decencies used in coding are stored here based on version

https://mvn.ci.artifacts.prod.walmart.com/artifactory/armada-mvn

Concord.yml – informs concord what to do

Looper.yml – informs looper what to do

Kitt.yml - ?

Overall approach is that,

Whenever code gets committed to git by a developer the webhook in place gets triggered and call concord job and concord.yml file configuration gets executed and validates whether the code needs a build or not. (The build is that for iib, code gets developed in local and then the build is done to generate .bar file and deployed into other environment to make it work. But for CPI the code developed in dev env itself and it doesn’t require build mechanism, only deployment is enough.) If build is required, it calls looper pro for iib scenario as looper pro is useful for doing build and the generated bar file pushed to artifactory then next step in concord to proceed with deployment process. In case, the build is not necessary then from concord it again calls the further steps to deploy the code to another env

Kalpit actual code

https://gecgithub01.walmart.com/intech/cpi-cicd/blob/Hackathon\_CICD/concord.yml

Kalpit modified details for hackathon

Main process - <https://gecgithub01.walmart.com/intech/cicd-test>

Main Concord Job and CPI Deployment and AI Code Review

<https://gecgithub01.walmart.com/intech/cpi-cicd>

CPI Code

<https://gecgithub01.walmart.com/intech/CICDDemoPackage->[A\_CICD\_Demo\_IFlow/tree/dev](https://gecgithub01.walmart.com/intech/CICDDemoPackage-A_CICD_Demo_IFlow/tree/dev)​

​

IIB Code Repo - <https://gecgithub01.walmart.com/intech/looper_generic>​

​

​

IIB Looper Build - <https://gecgithub01.walmart.com/intech/concord_test/tree/looper-iib>

Concord - <https://concord.prod.walmart.com/#/org/MMI/project/cpi-cicd/process>

Artifactory - <https://generic.ci.artifacts.walmart.com/ui/native/temporary-storage-generic/esb-routing-rules/>

Arun details all tested:

looper pro Anisha tested:

<https://dx.walmart.com/nextgenci/v2/detail/68f39c7b-9635-439c-8b7d-ba4b25062c39>

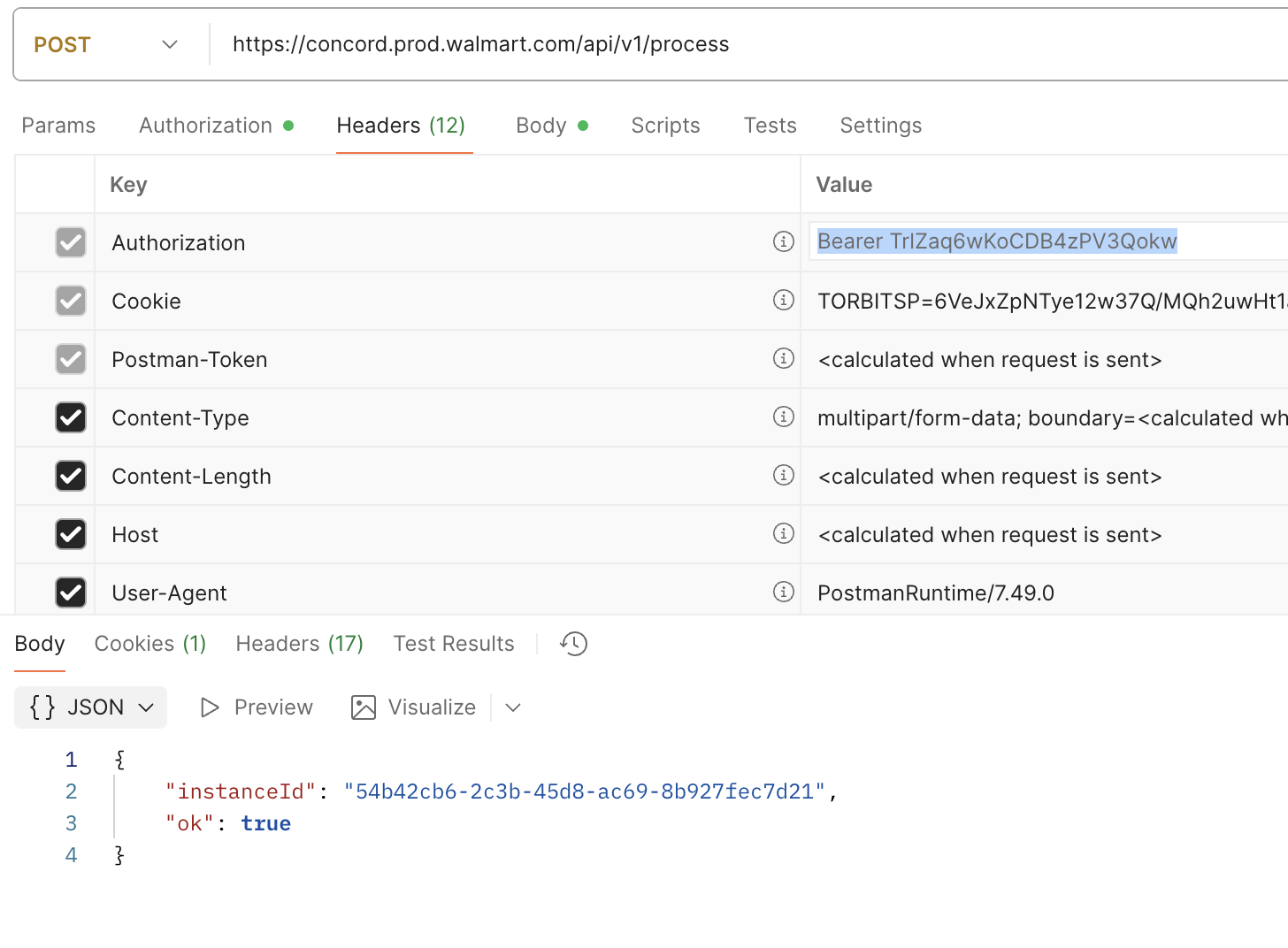
this the git where I test how the concord and looper can get trigger

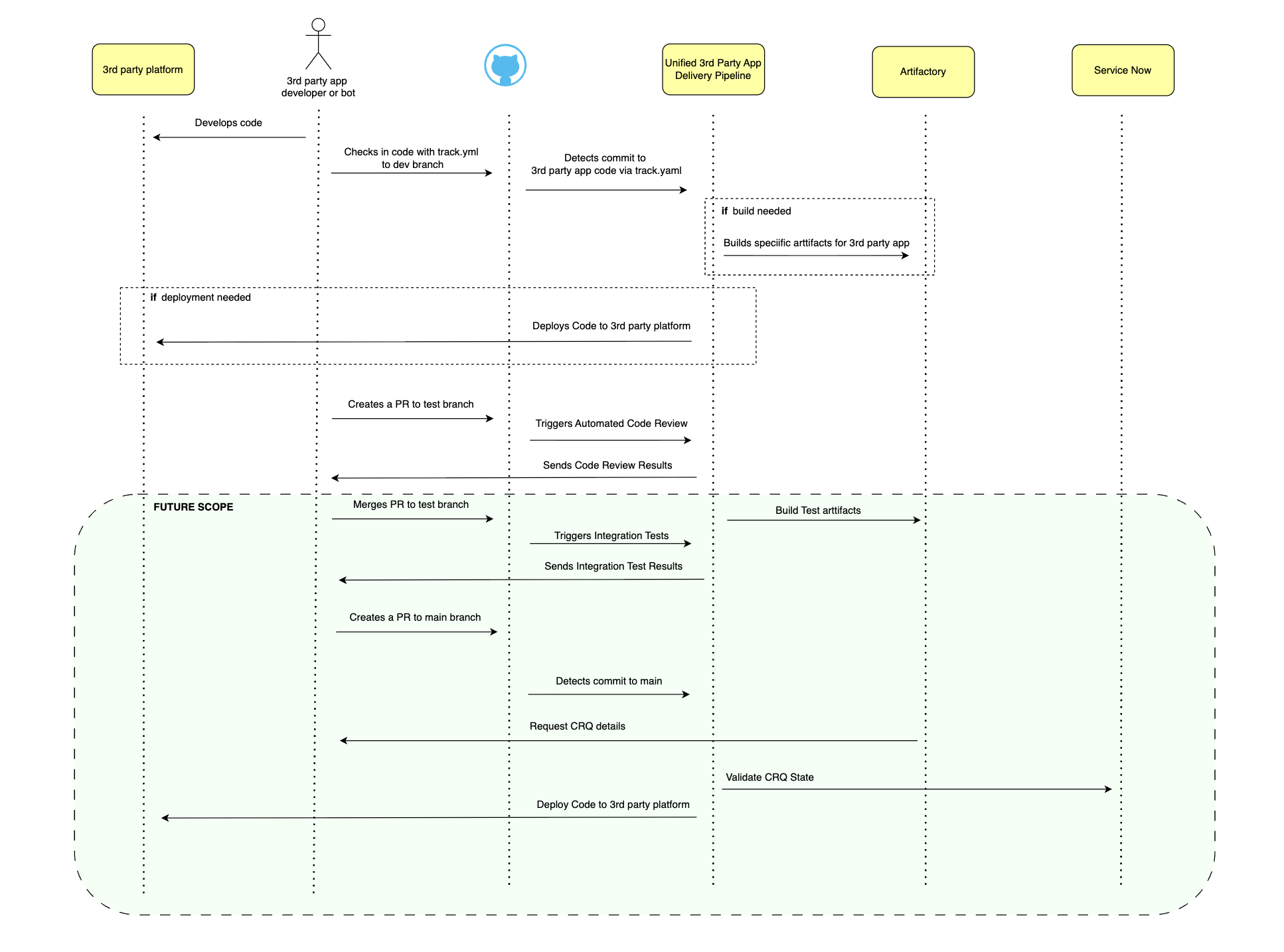
<https://gecgithub01.walmart.com/intech/concord_test>

<https://concord.prod.walmart.com/#/org/GBS-Integration-Technologies/project/call-looper/process>

<https://dx.walmart.com/nextgenci/v2/detail/1c9bda37-531e-4bc7-917a-9df8de48f507>

the way to trigger concord job





### [cpi\_cicd\_api.py](https://gecgithub01.walmart.com/intech/cpi-cicd/blob/main/cpi_cicd_api.py)

this has full code similar to github file but with AI based code review as extras

### [cpi-tech-doc-generator.ipynb](https://gecgithub01.walmart.com/intech/cpi-cicd/blob/main/cpi-tech-doc-generator.ipynb)

has AI based doc generator

### [cpi\_github\_api.py](https://gecgithub01.walmart.com/intech/cpi-cicd/blob/main/cpi_github_api.py)

similar to 1st one but no AI and it works with diff artifactory

1. Overview

This script is a FastAPI-based automation API for deploying SAP CPI (Cloud Platform Integration) IFlow artifacts to GitHub. It supports:

Downloading IFlows from SAP CPI using OData APIs.

Creating/managing GitHub repositories and branches for different environments (dev, test, prod).

Automated pull request creation and webhook-driven deployments.

Proxy support for network environments.

RESTful endpoints for all major operations.

2. Configuration and Models

CPI\_TENANTS: Maps environment names (dev, test, prod) to their SAP CPI tenant URLs.

CONNECTION\_CONFIG: Sets timeouts, retry logic, SSL verification, and chunk size for HTTP requests.

GLOBAL\_PROXY\_CONFIG: Proxy settings for all CPI API calls.

ProxyConfig: Class to encapsulate proxy settings and provide them to requests.

Pydantic Models: Define the structure and validation for API requests and responses (e.g., GitUploadRequest, GitUploadResponse, Error

3. CPIClient Class

Handles all interactions with SAP CPI OData APIs:

Authentication: Uses Basic Auth (S-User and password).

Session Management: Sets up retry logic and proxy support.

CSRF Token Handling: Fetches and caches CSRF tokens for POST/PUT/DELETE requests.

IFlow Operations:

get\_iflow\_details: Fetches IFlow metadata.

download\_iflow\_artifact: Downloads the IFlow as a ZIP file.

upload\_iflow\_artifact / update\_iflow\_artifact: Uploads or updates IFlow artifacts in CPI.

deploy\_iflow: Deploys an IFlow to runtime.

get\_deployment\_status: Checks deployment status.

package\_exists / create\_package\_if\_not\_exists: Ensures the integration package exists.

upload\_or\_update\_iflow\_artifact: Combines upload/update logic.

test\_connection / validate\_api\_endpoints: For connectivity and compliance checks.

4. GitHubClient Class

Handles all interactions with GitHub:

Repository Management: Creates or fetches repositories.

Branch Management: Checks for branch existence, creates branches, ensures correct branch structure.

Pull Requests: Creates pull requests for code review and promotion.

Webhooks: Sets up webhooks for automated deployments.

Git Operations: Clones repositories, checks out branches, commits, and pushes changes.

Download/Extract: Downloads repository contents as ZIP and extracts them.

5. Utility Functions

extract\_zip\_file: Extracts ZIP files to a directory.

create\_metadata\_file: Writes deployment metadata to a file.

validate\_environment: Checks if the environment is supported.

cleanup\_temp\_directory: Deletes temporary directories.

6. FastAPI Endpoints

/gitUpload:

Main endpoint. Accepts a request to deploy an IFlow from CPI to GitHub. Steps:

Validates environment.

Downloads IFlow from CPI.

Creates/gets GitHub repo and branch.

Extracts artifact, creates metadata, commits, and pushes.

Creates a pull request.

Sets up webhooks for test/main branches.

/health:

Health check endpoint.

/environments:

Lists supported environments and their URLs.

/setup-webhooks:

Manually sets up webhooks for a repository.

/validate-cpi-connection:

Validates CPI credentials and connectivity.

/validate-api-compliance:

Checks if CPI OData API endpoints are compliant.

/iflows/{environment}:

Lists available IFlows in a CPI environment.

/test-upload-update:

Test endpoint to upload/update an IFlow from GitHub to CPI and deploy it.

/webhook/github:

GitHub webhook endpoint. On push to test/main branches, triggers deployment to the corresponding CPI environment.

/webhook/status:

Returns webhook configuration and status.

7. Webhook and Automation Logic

Webhook Signature Verification:

(Commented out for testing) Ensures that incoming webhook requests are from GitHub.

Branch-to-Environment Mapping:

main branch → prod environment

test branch → test environment

dev branch → dev environment

Webhook Flow:

On push to test/main, clones the repo, zips the IFlow, uploads/updates in CPI, and deploys it.

8. Server Startup

Main Block:

Sets up webhook secret and base URL from environment or generates them.

Prints startup info and runs the FastAPI app with Uvicorn.

9. Security and Best Practices

Credentials and secrets should be stored in environment variables.

Webhook signature verification should be enabled in production.

SSL verification is enabled by default.

All inputs are validated using Pydantic models.

======

CPIClient class:

Summary Table

Method/Property Purpose

\_\_init\_\_ Initialize client, session, auth, proxy, CSRF token

\_create\_auth\_header Build HTTP Basic Auth header

\_create\_session\_with\_retry Setup session with retry and proxy

\_get\_csrf\_token / \_refresh\_csrf\_token Manage CSRF tokens for write operations

\_make\_request Centralized HTTP request logic with error handling

get\_iflow\_details Fetch IFlow metadata

download\_iflow\_artifact Download IFlow ZIP

upload\_iflow\_artifact Upload new IFlow

update\_iflow\_artifact Update existing IFlow

upload\_or\_update\_iflow\_artifact Upload or update IFlow

deploy\_iflow Deploy IFlow to runtime

get\_deployment\_status Get deployment status

package\_exists Check if package exists

create\_package\_if\_not\_exists Create package if missing

iflow\_exists\_in\_designtime Check if IFlow exists in design-time

test\_connection Test CPI connectivity

validate\_api\_endpoints Validate OData API endpoints

========

📋 Complete List of API Endpoints for Postman Testing

GET Endpoints (5):

GET /health - Health check endpoint

GET /proxy-config-info - Get proxy configuration information

GET /environments - Get supported CPI environments

GET /iflows/{environment} - List available IFlows in a CPI environment

GET /webhook/status - Check webhook configuration and status

POST Endpoints (6):

POST /gitUpload - Main deployment endpoint (CPI IFlow to GitHub)

POST /setup-webhooks - Manually setup webhooks for repositories

POST /validate-cpi-connection - Validate CPI credentials and connection

POST /validate-api-compliance - Validate CPI OData API compliance

POST /test-upload-update - Test upload/update IFlow from GitHub to CPI

POST /webhook/github - GitHub webhook endpoint for automatic deployment

========

Simple Test Endpoints (No authentication needed):

GET http://localhost:8000/health

GET http://localhost:8000/environments

GET http://localhost:8000/proxy-config-info

GET http://localhost:8000/webhook/status

Complex Endpoints (Require request body):

POST http://localhost:8000/gitUpload

POST http://localhost:8000/validate-cpi-connection

POST http://localhost:8000/setup-webhooks

graph TD

A[POST /gitUpload] --> B[Validate Environment]

B --> C[Get CPI Tenant URL]

C --> D[Create Temp Directory]

D --> E[Initialize CPI Client with Proxy]

E --> F[Get IFlow Details from CPI]

F --> G[Download IFlow Artifact ZIP]

G --> H[Initialize GitHub Client]

H --> I[Create/Get GitHub Repository]

I --> J[Clone Repository & Setup Branch]

J --> K[Extract ZIP to Repository]

K --> L[Create Metadata File]

L --> M{Environment = dev?}

M -->|Yes| N[Create Empty kitt.yml]

M -->|No| O[Skip kitt.yml]

N --> O

O --> P[Commit & Push Changes]

P --> Q{Environment = dev?}

Q -->|Yes| R[Create PR to test branch]

Q -->|No| S[Create PR to main branch]

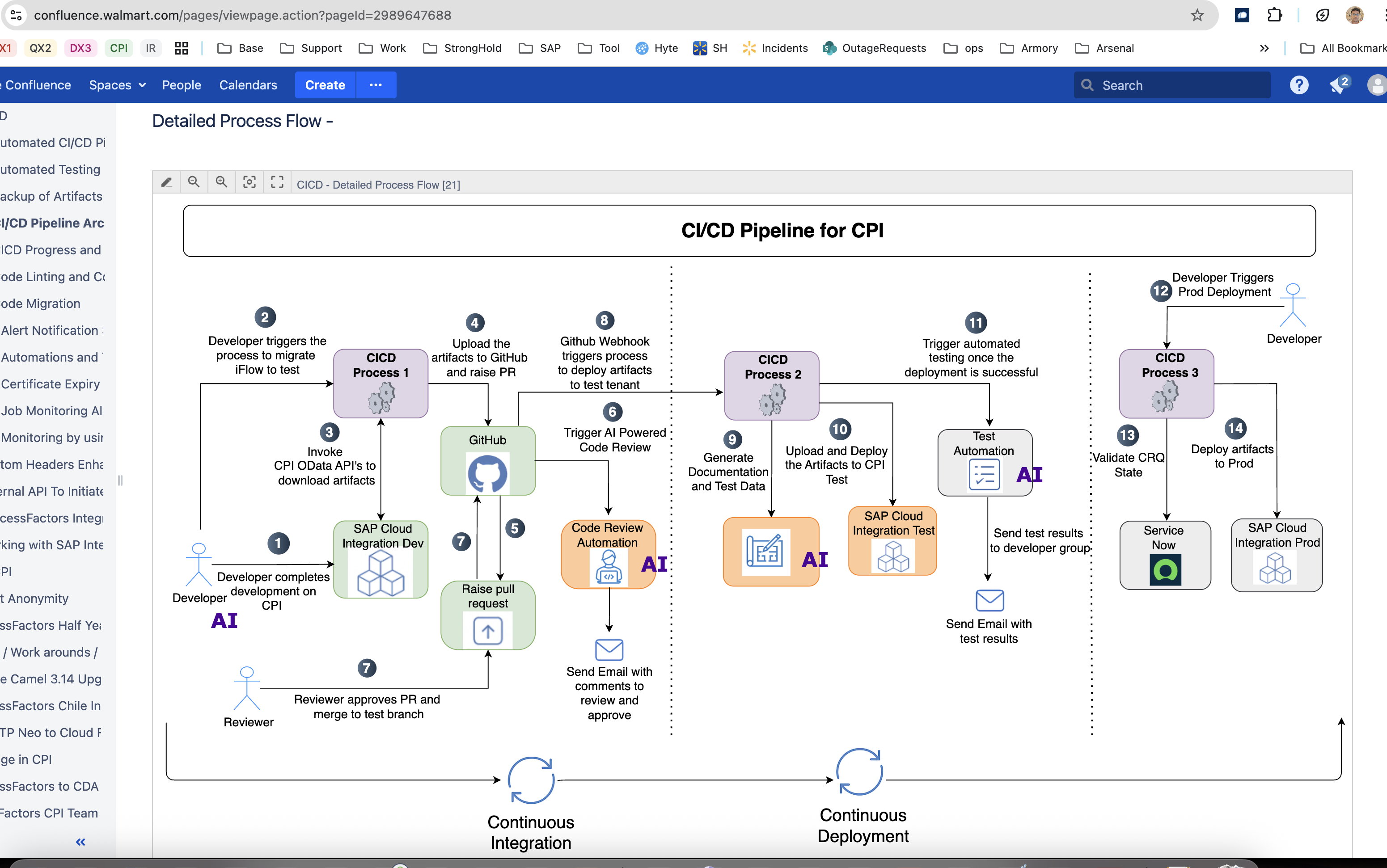
R --> T[Setup Webhooks for test/main]

S --> U[Return Success Response]

T --> U

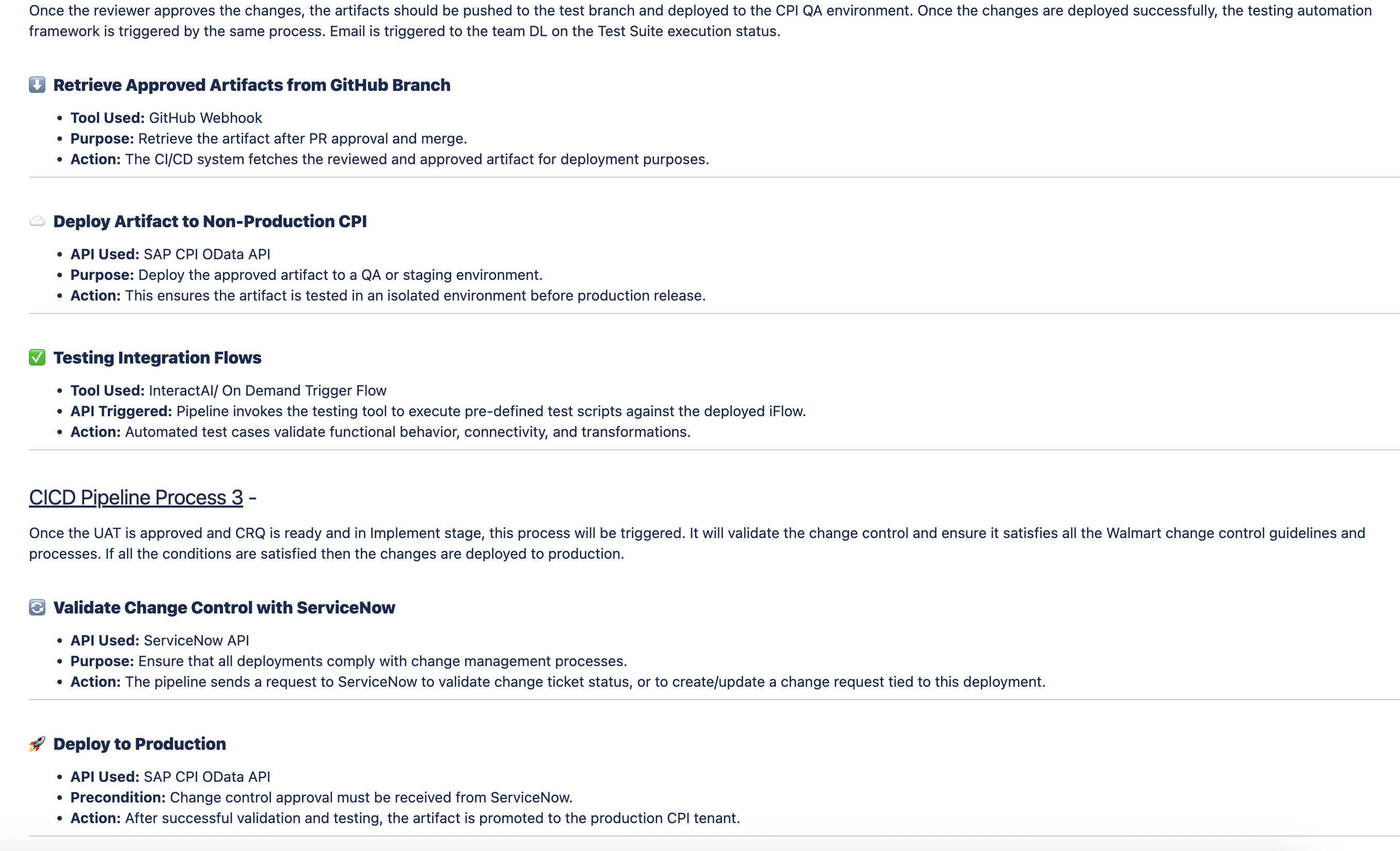
A diagram of a software application

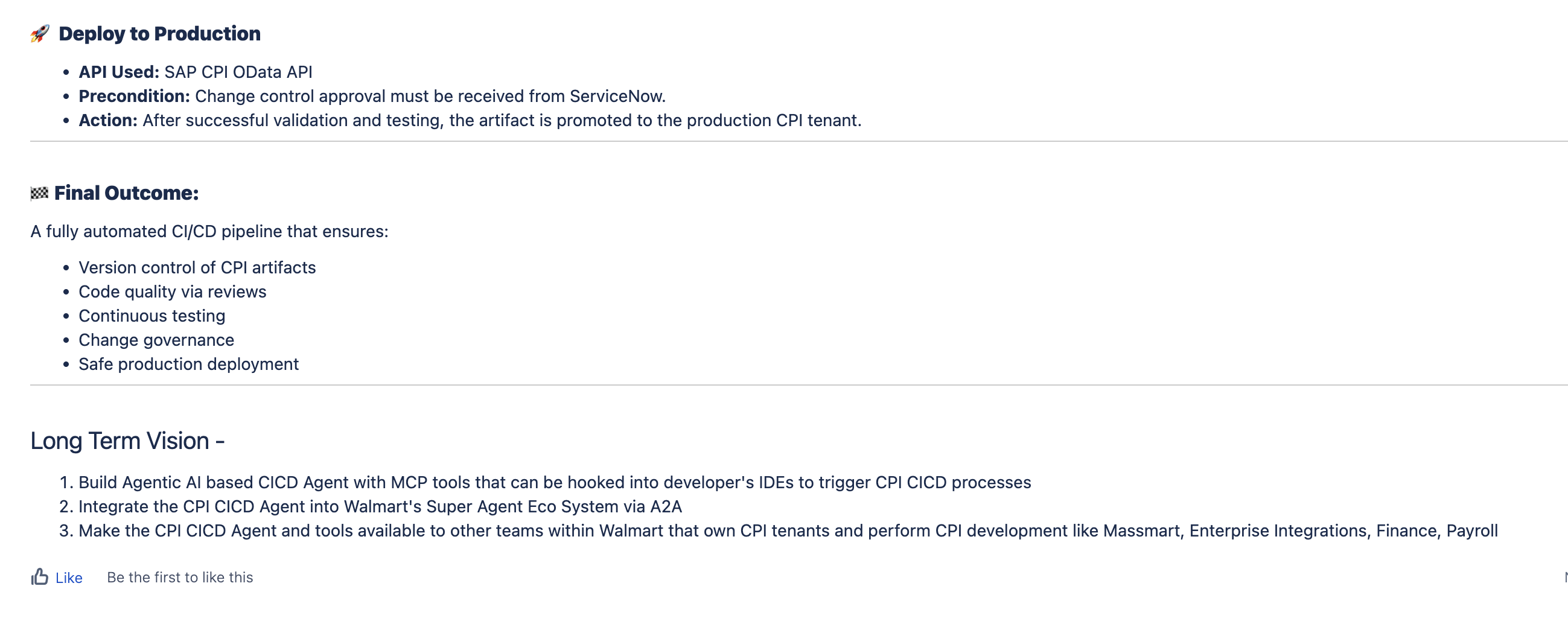
AI-generated content may be incorrect.



A screenshot of a computer

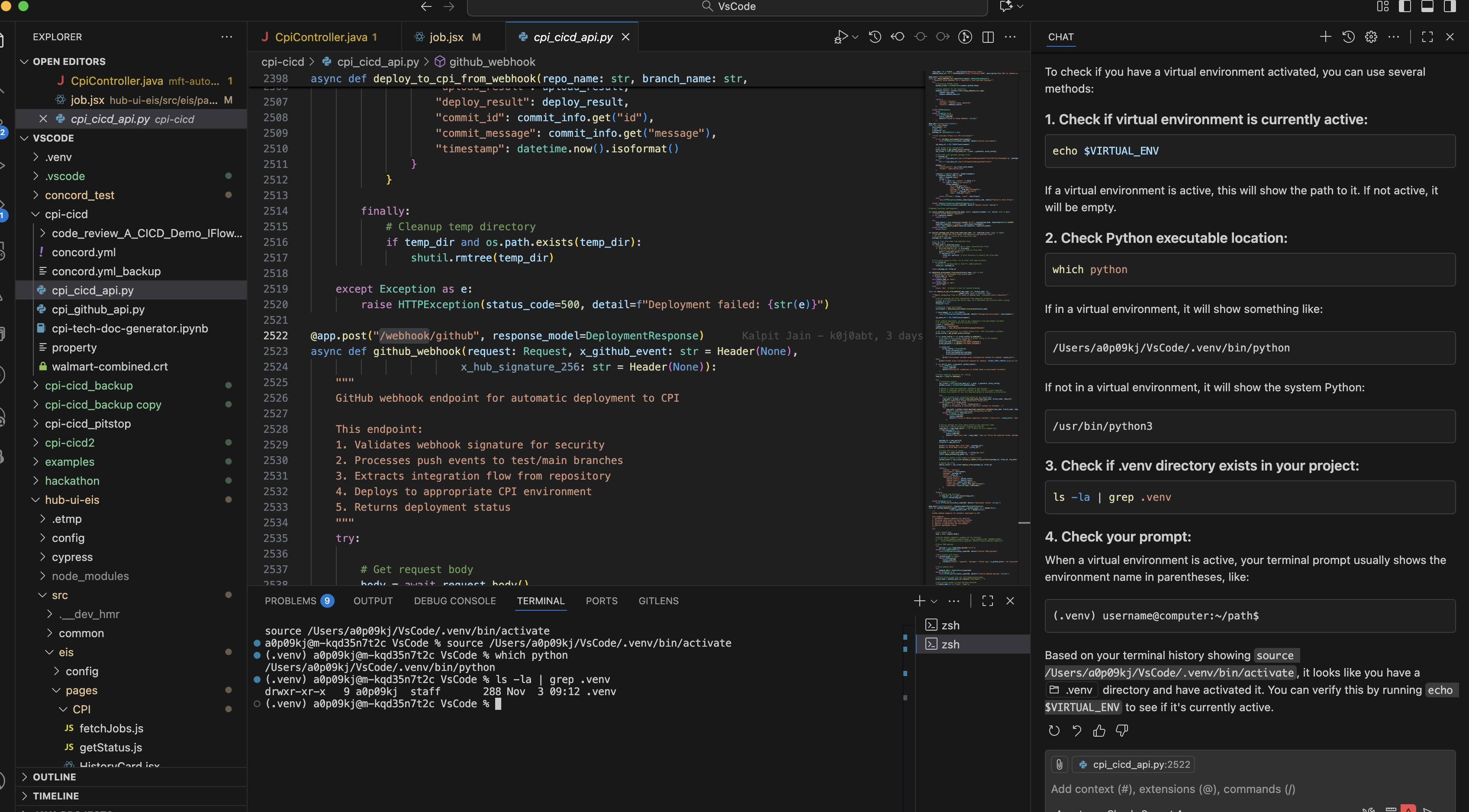
AI-generated content may be incorrect.





A screenshot of a computer program

AI-generated content may be incorrect.



To run the code we need to install many packages but some package installed and still showing it was not. When I check I understood they installed in virtual mode so I switched to .venv this way the code gets executed

cd /Users/a0p09kj/VsCode/cpi-cicd && /Users/a0p09kj/VsCode/.venv/bin/python cpi\_cicd\_api.py

pkill -f "uvicorn.\*cpi\_cicd\_api"

To start the .venv then

source /Users/a0p09kj/VsCode/.venv/bin/activate

to stop type deactivate

to execute the code

python3 -m uvicorn cpi\_github\_api:app –reload

or

python3 -m cpi\_cicd\_api:app 🡪 kalpit executed in his system

Git PAT is required in some place

Git settings -> developer options -> classic token -> enable repo and user

Using concord encrypt it which is valid within that project only – this can be useful if concord is calling looper pro

to trigger git\_.py file then another payload format is required

