



Learn Network Automation with Github Copilot

Gowtham Tamilselvan

Praveen Poojary

BRKOPS-1726

CISCO *Live!*



About Us

Praveen Poojary

Principal Architect

13 Years in Cisco

#3xCCIE

#CCDE



Gowtham Tamilselvan

Software Architect

13 Years in Cisco

Automation Engineer

Distinguished Speaker



Agenda

- AI Primer
- Introduction to Github Copilot
- Network Automation Use-Cases
- Conclusion

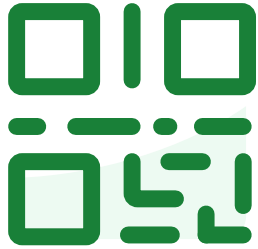


Visual Studio Code



slido

Please download and install the
Slido app on all computers you
use



Join at slido.com
#20251726

① Start presenting to display the joining instructions on this slide.



How do you rate your proficiency level with using AI?

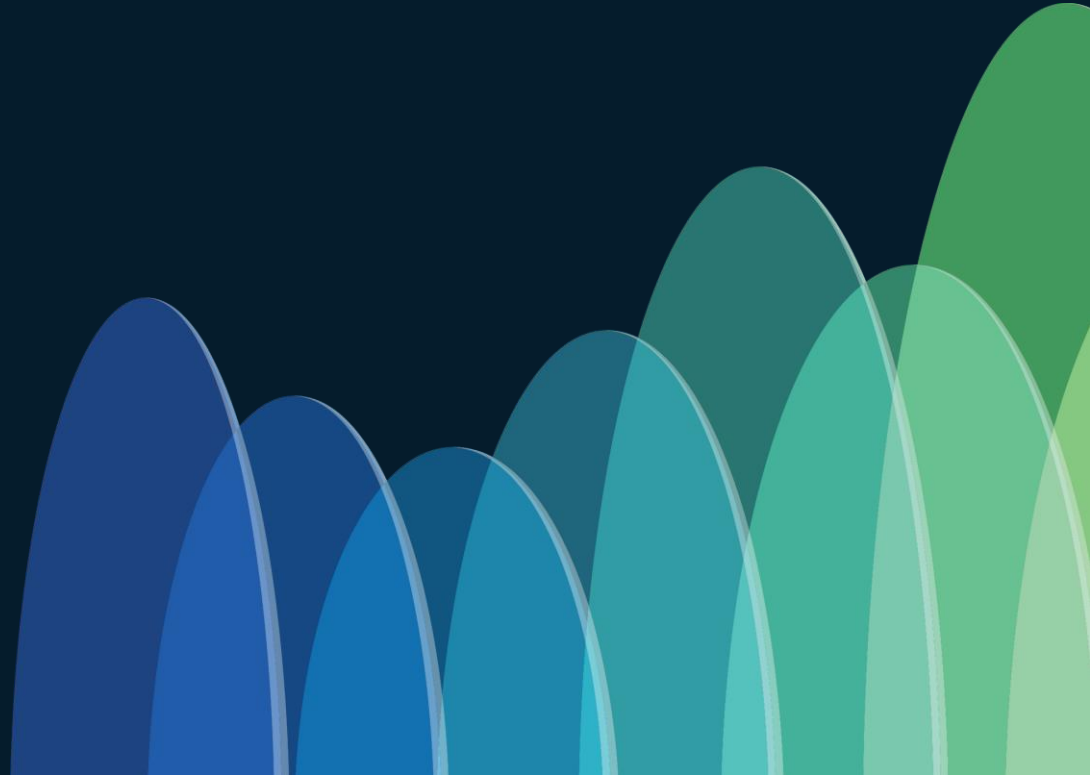
① Start presenting to display the poll results on this slide.



Which LLM Models are you using today?

① Start presenting to display the poll results on this slide.

AI Primer



Defining Artificial Intelligence

Artificial Intelligence

Simulates human intelligence to solve complex tasks

Machine Learning

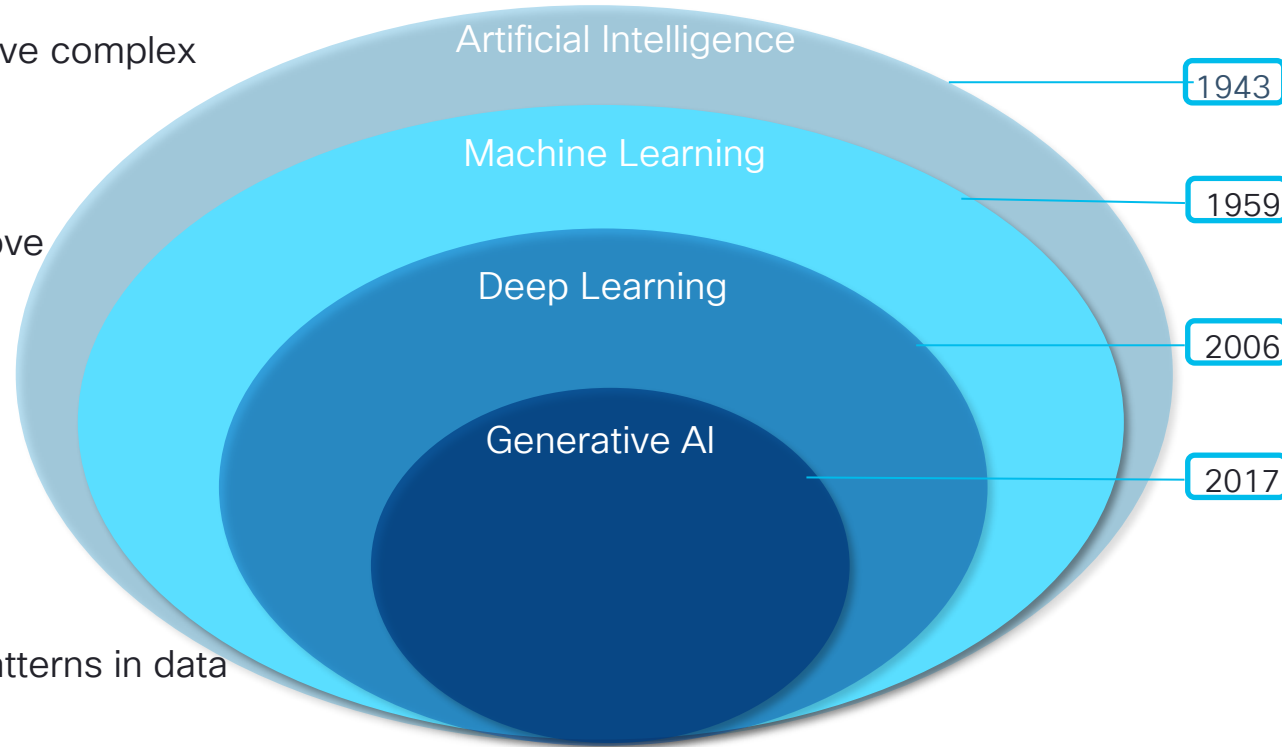
Enables systems to learn and improve from data

Deep Learning

Processes unstructured data using neural networks

Generative AI

A branch of deep learning
Creates new content by learning patterns in data



What is GPT?

Generative

AI models designed to create new content, such as text, images, or videos.

Pre-Trained

Models trained on vast datasets before application-specific fine-tuning.

Transformer

A neural network architecture focusing on contextual relationships in data (introduced by Google, 2017)

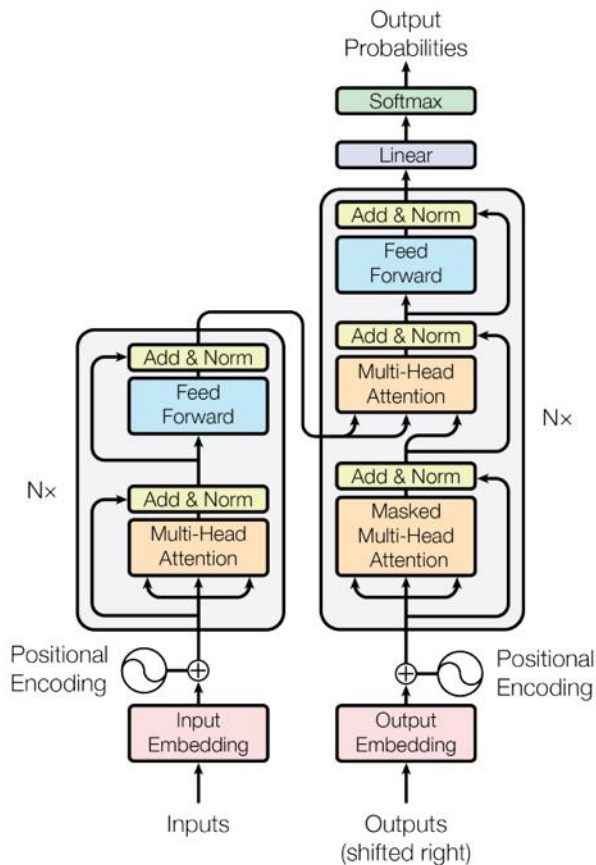


Figure 1: The Transformer - model architecture.

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* †
illia.polosukhin@gmail.com

[arXiv:1706.03762](https://arxiv.org/abs/1706.03762) [cs.CL]

What makes GenAI different?

Deterministic...

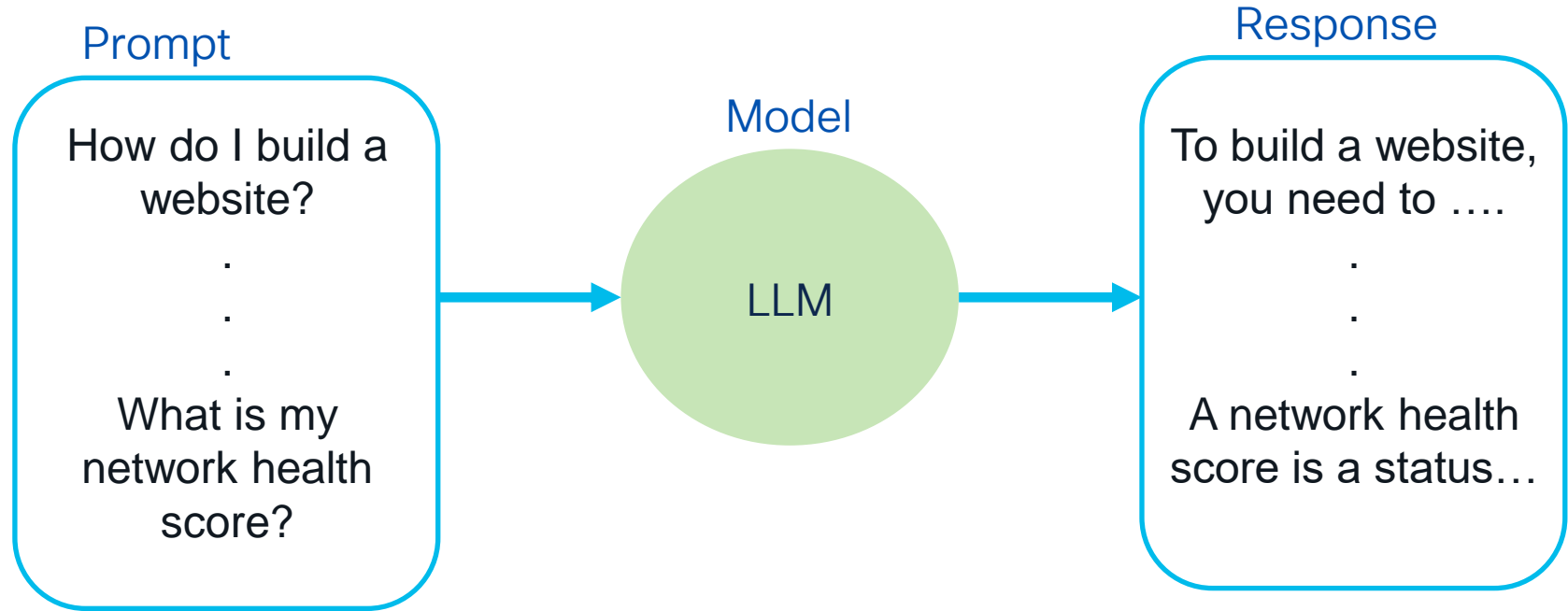
Vs

Probabilistic...

Input A equals output B

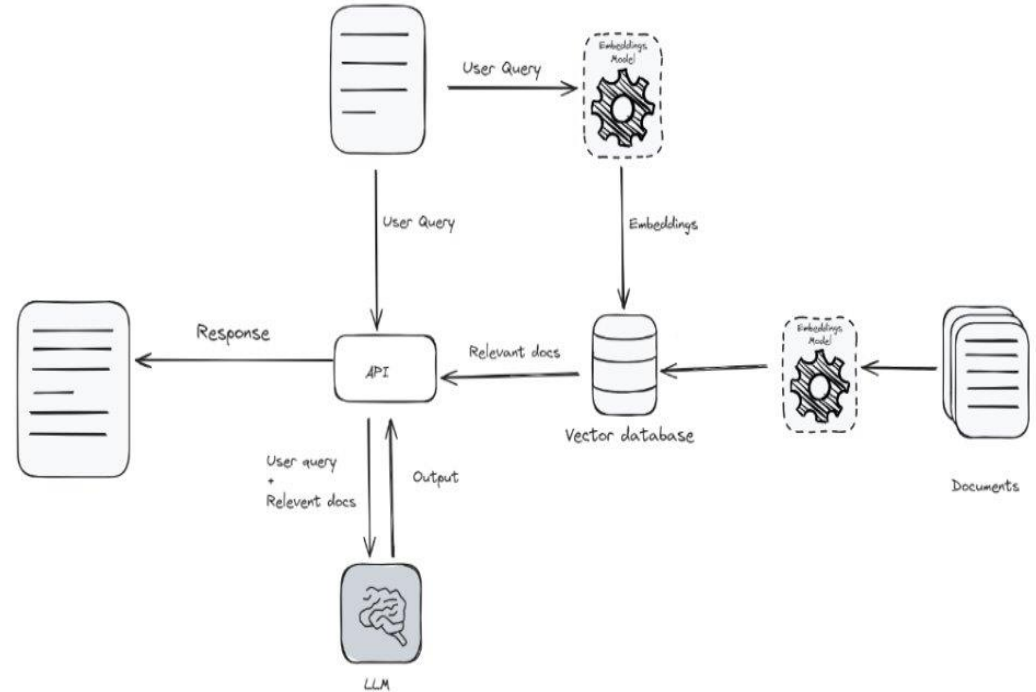
Input A could equal B, C, D, etc

GenAI Prompt and Response



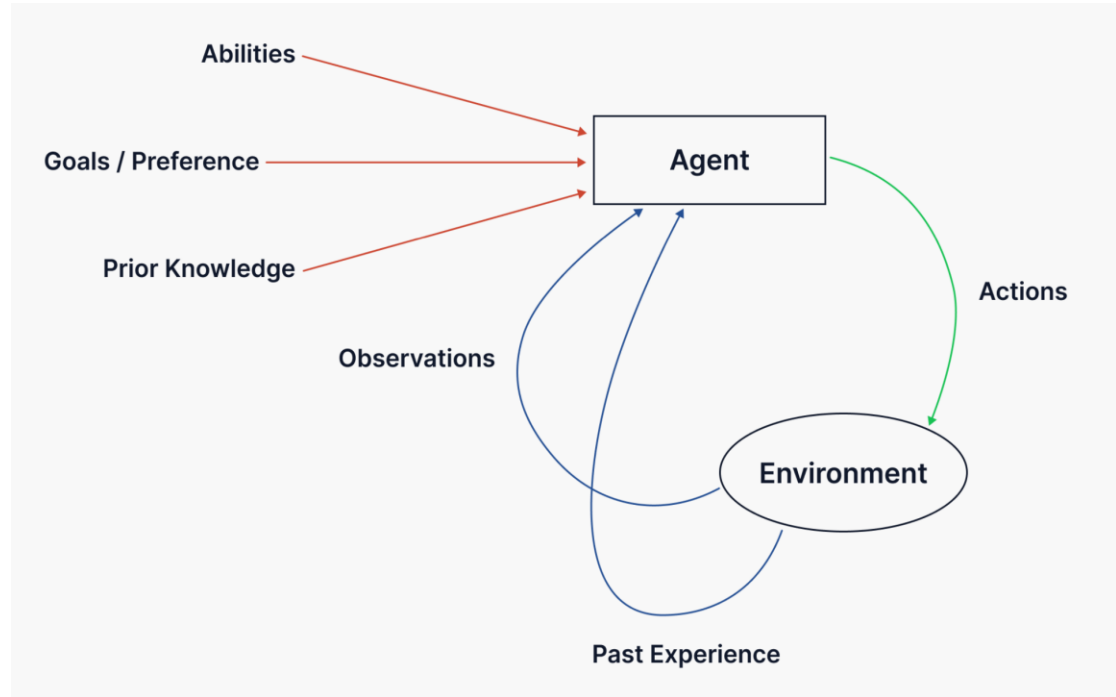
What is Retrieval Augmented Generation (RAG)

- The process of optimizing the output of a large language model, so it **references an authoritative knowledge base** outside of its training data sources before generating a response.

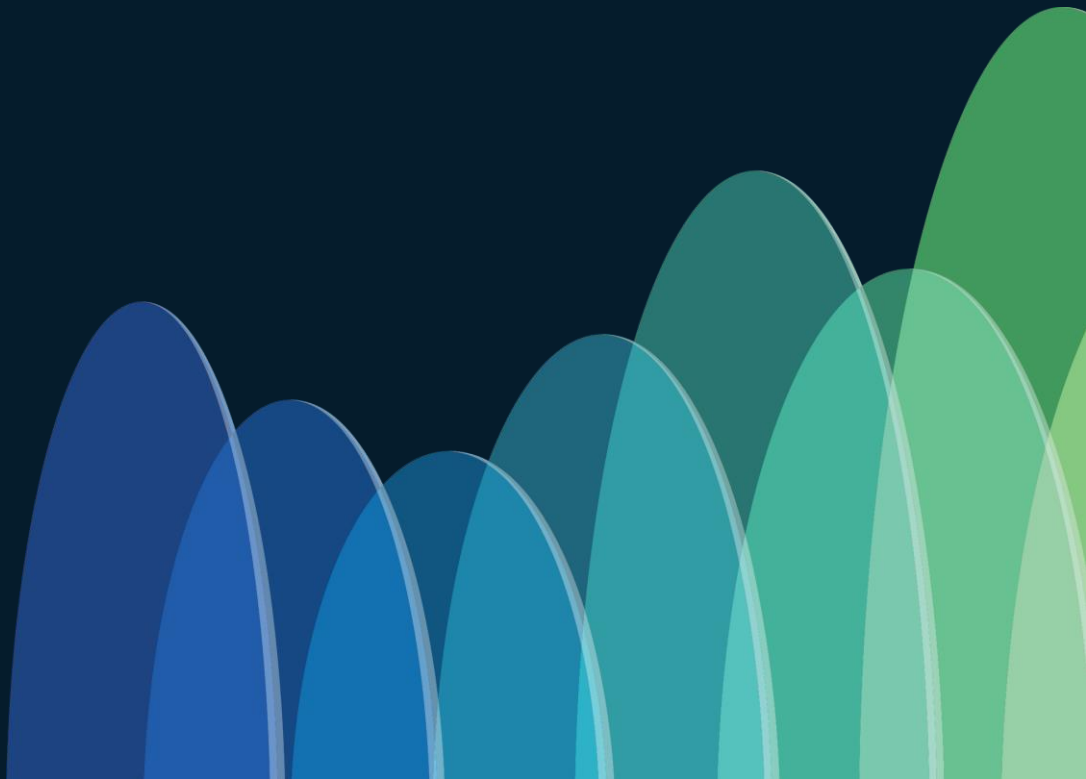


What is an AI Agent?

A software program that uses **artificial intelligence (AI)** to perform tasks, answer questions, and solve problems.



Introduction to GitHub Copilot





Rate your experience level using Github Copilot?

① Start presenting to display the poll results on this slide.

GitHub Copilot

- Developed by GitHub and OpenAI in 2021
- Supports Python, JavaScript, TypeScript, Ruby, Go, C# and C++
- Developers are using it to write code 55% faster

The screenshot displays the GitHub Copilot website with a dark blue background. At the top, the text "The AI editor for everyone" is prominently displayed. Below this, there are two buttons: "Get started for free" and "See plans & pricing". A smaller text line says "Already have Visual Studio Code? [Open now](#)". On the right side, there is a 3D rendering of the GitHub Copilot character, a blue robot head with large eyes. The central part of the image shows a code editor window with a file named "parse_expenses.py". The code is in Python and includes comments and a function definition. To the right of the code editor, there is a chat interface with a user named "adelevance" asking "Can you fix this for me?" and a response from "GitHub Copilot" stating "Sure, I can help with that. The issue seems to be with the order of the values in the tuple and the delimiter used in the split method. The". At the bottom, a section titled "GitHub Copilot is available on your favorite platforms:" features a row of icons for various development environments: GitHub, Visual Studio, Visual Studio Code, JetBrains IDEs (PyCharm, IntelliJ, WebStorm), and others.

The AI editor for everyone

Get started for free See plans & pricing

Already have Visual Studio Code? [Open now](#)

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the
5     list of triples (date, value, currency).
6     Ignore lines starting with #.
7     Parse the date using datetime.
8     Example expenses_string:
9     2023-01-02 -34.01 USD
10    2023-01-03 2.59 DKK
11    2023-01-03 -2.72 EUR
12    """
```

adelevance

Can you fix this for me?

parse_expenses.py 1-30

GitHub Copilot

> Used 1 reference

Sure, I can help with that. The issue seems to be with the order of the values in the tuple and the delimiter used in the split method. The

GitHub Copilot is available on your favorite platforms:

GitHub

Why Choose Copilot Over ChatGPT for Coding?

GitHub Copilot	ChatGPT
Purpose-Built for Developers	General Purpose
More Accurate Code	Less Accurate Code
Context-Aware	Limited Code Context
Seamless IDE Integration	No IDE Integration
Developer focused Interface	Familiar/Friendly Interface

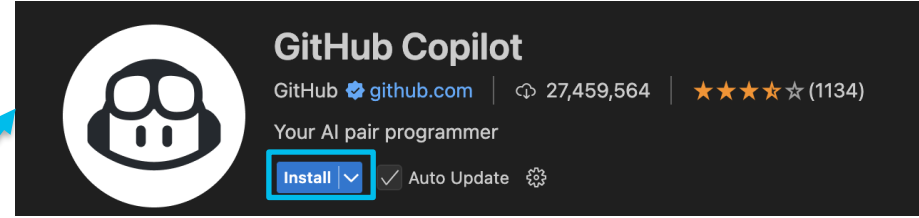
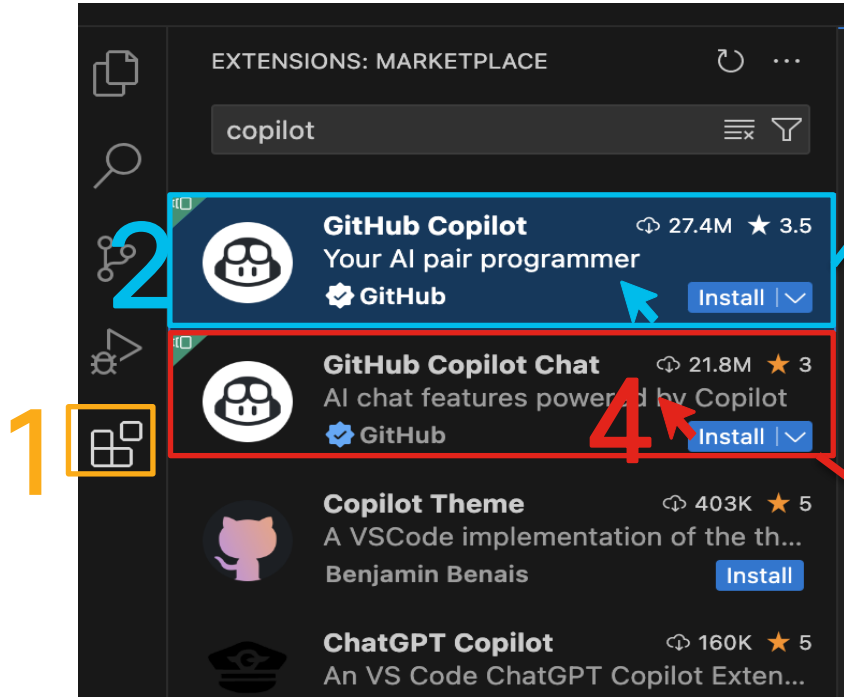
Prerequisites To Using GitHub Copilot

- Active GitHub account
- Subscription to GitHub Copilot
- Supported IDE
- Internet connection

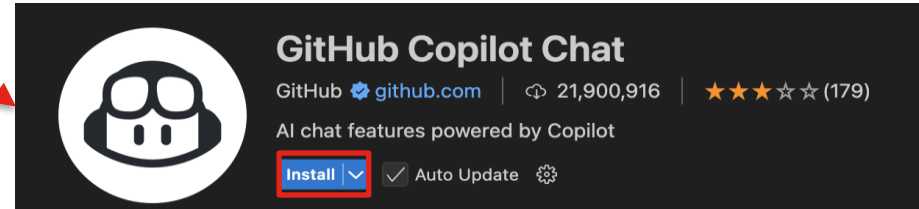
The screenshot shows the GitHub Copilot pricing page with four subscription tiers: Free, Pro, Business, and Enterprise. Each tier includes a description, price, and a 'Get started' button. The Free tier is marked as 'New' and includes a link to 'Open now' for existing VS Code users. The Pro tier is marked as 'Most popular' and includes a link to 'Learn more' for verified students and teachers. The Business and Enterprise tiers include 'Contact sales' buttons.

Tier	Price	Frequency	Key Features / Notes
Free	\$0 USD	-	For developers looking to get started with GitHub Copilot. Includes up to 2,000 completions and 50 chat requests per month.
Pro	\$10 USD	per month. First 30 days free.	For developers who want unlimited access to GitHub Copilot. Free for verified students, teachers, and maintainers of popular open source projects.
Business	\$19 USD	per user / month	For teams ready to accelerate their workflows with GitHub Copilot.
Enterprise	\$39 USD	per user / month	For organizations seeking a fully customized GitHub Copilot experience.

Install GitHub Copilot with Chat in VS Code



3 Click Install button for Copilot
(shows Uninstall when already installed)



5 Click Install button for Copilot Chat
(shows Uninstall when already installed)

VS Code – Copilot View

The image shows the Visual Studio Code (VS Code) interface with several key features highlighted by red boxes and labels:

- CHAT SIDEBAR:** Located on the left, it features the Copilot logo, the text "Ask Copilot", and instructions: "Copilot is powered by AI, so mistakes are possible. Review output carefully before use." Below this, it lists shortcuts: "U or type # to attach context", "@ to chat with extensions", and "Type / to use commands". At the bottom, there is an "Ask Copilot" input field with a dropdown menu showing "test.py Current file" and icons for attachments and voice input.
- INLINE CHAT:** Located in the center editor area, it shows a snippet of code in `test.py` with a comment: "Press [U] to ask GitHub Copilot to do something. Start typing to dismiss."
- PROJECT FILES:** Located on the right, it shows the "EXPLORER" sidebar with a tree view of the project structure, including files like `.github`, `container`, `uc1`, `uc2`, `uc3`, `uc4`, `.gitignore`, `changelog.md`, `LICENSE`, `notes.md`, and `README.md`.
- TERMINAL:** Located at the bottom, it shows a terminal window with the prompt `(brkops-venv) → BRKOPS-1726 git:(main) *`.

Additional interface elements visible include the top status bar showing "BRKOPS-1726", the bottom status bar showing "8 LF () Python 3.9.6 64-bit", and a "Copilot Status" indicator in the bottom right corner.

Copilot
Status



Github Copilot Prompt Engineering

Three “S” Principle

1. SIMPLE
2. SPECIFIC
3. SHORT

Github Copilot Prompt Engineering

Three “S” Principle

1. SIMPLE

- a) Breakdown the code to smaller steps and prompt Copilot one step at a time
- b) Copilot performs better with smaller steps, reduces Hallucinations
- c) Can help to use the word “SIMPLE” in your prompt

Github Copilot Prompt Engineering

Three “S” Principle

2. SPECIFIC

- a) Provide Specific Context to Copilot by leveraging in-build Agents
- b) Use @workspace agent to have Copilot review all of the project files when answering questions

Github Copilot Prompt Engineering

Three “S” Principle

3. SHORT

1. You don't need proper grammar, spelling, or complete sentences when Prompting Copilot
2. Keep Prompt short and to the point for better results

Demo 1 – Create a testbed File

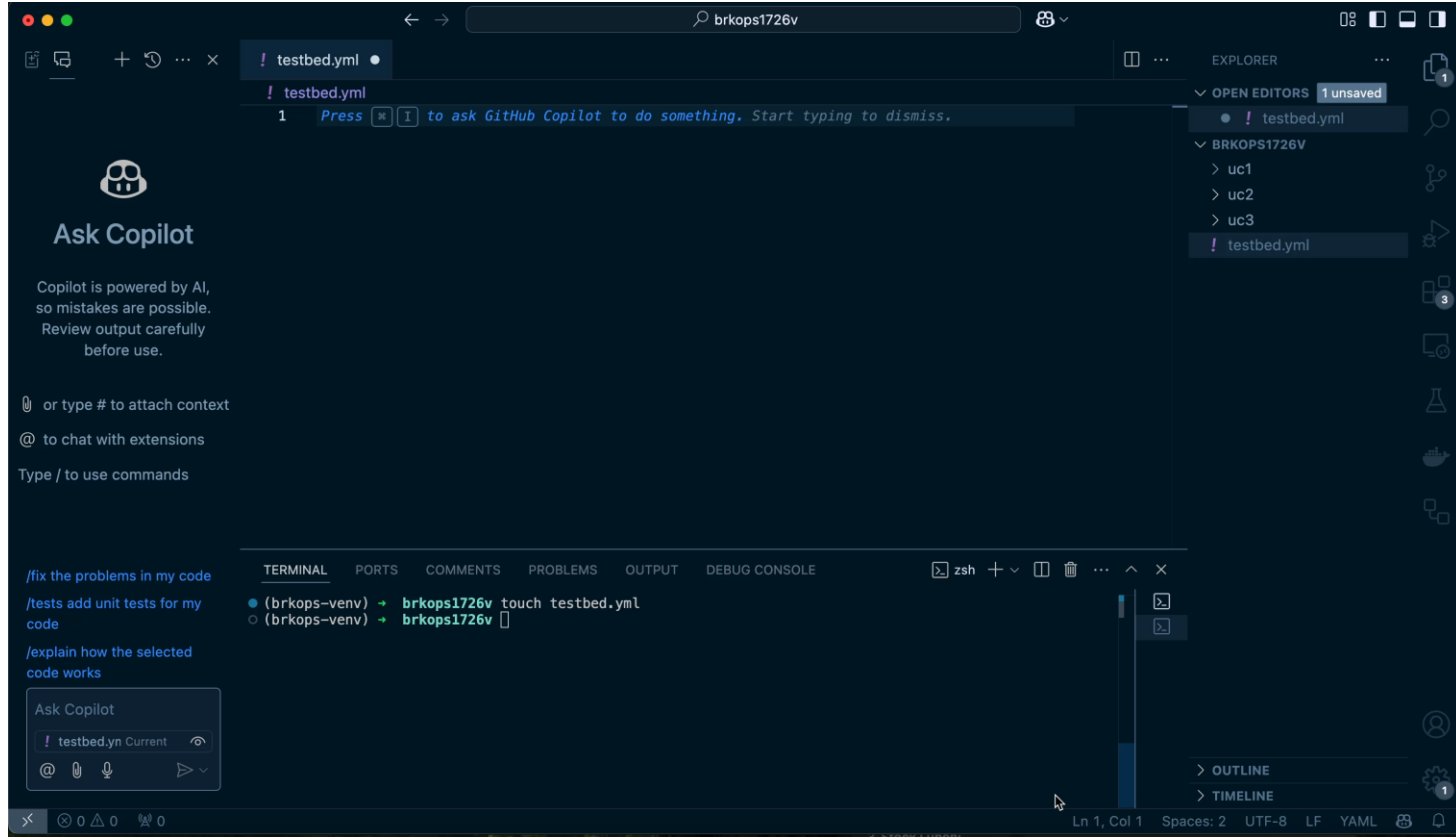
Single Prompt:

```
create a testbed yaml file with two Cisco devices
```

```
1) device name CPS-FE1; type switch; os iosxe;  
credentials: username admin, password Cisco;  
connections set to CLI: ip 10.1.1.5, protocol ssh;
```

```
2) device name dnac; type controller; os linux;  
credentials: username admin, password Cisco;  
connections set to rest: ip 10.1.1.1, protocol  
https;
```

Demo 1 – Create a testbed.yml File



Demo-1

Network Automation Use-Cases

Network Automation with Copilot and GenAI

Explore use-cases with Github Copilot and GenAI

- UC1: Network Device Data Extraction
- UC2: Network Device Workflow Creation
- UC3: Gen AI Agent based Automation

UC1: Network Device Data Extraction

UC1: Network Device Data Extraction

Objective: Connect to Cisco Switch and extract a list of “show commands” using Github Copilot

Functional Steps:

1. Utilize testbed file from DEMO-1 with device connectivity details
2. Create “show_cmds.py” python file
3. Use python pyats libraries to connect to device via SSH
4. Retrieve a list of show commands [*show version, show ip int bri, show vlan*] from the device
5. Format the output and save it to a file

UC1: Network Device Data Extraction

MULTIPE PROMPTS:

Import genie testbed, logging, and datetime libraries, then load the testbed.yml file

Connect to device "FE1" and set prompt_recovery=true

Run following commands and parse the output: "show version", "show ip interface brief", "show vlan"
Then save that output to file with current date

Disconnect from device

Simplify and improve overall code debuggability

Demo 2 – Network Device Data Extraction

The screenshot shows a Visual Studio Code editor window with the following components:

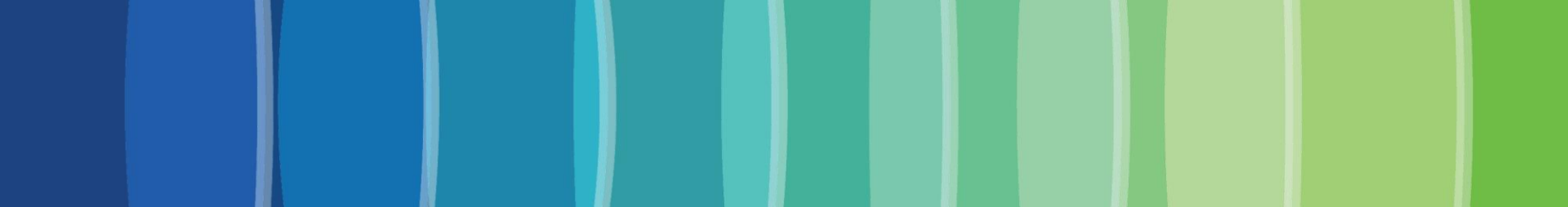
- Editor Area:** Displays a file named `sh_cmds.py` with a single line of text: `1 Press [Ctrl][I] to ask GitHub Copilot to do something. Start typing to dismiss.`
- Left Panel:** Contains the "Ask Copilot" chat window. It includes the Copilot logo, a warning that "Copilot is powered by AI, so mistakes are possible. Review output carefully before use.", and instructions on how to use the chat (e.g., "or type # to attach context", "@ to chat with extensions", "Type / to use commands"). Below these instructions is a text input field with the placeholder "Ask Copilot" and a list of suggested prompts: `/fix the problems in my code`, `/tests add unit tests for my code`, and `/explain how the selected code works`. A small dropdown menu shows the current file `sh_cmds.py` is attached to the chat.
- Right Panel:** Contains the "EXPLORER" view showing the project structure. It lists the files `prompt.md` and `sh_cmds.py` under the `uc1` folder, and `testbed.yml` under the `BRKOPS1726V` folder.
- Terminal:** At the bottom, it shows a terminal window with the prompt `(brkops-venv) → uc1`.

Demo-2

CISCO *Live!*

UC1: Network Device Data Extraction Recap

- Learned how to breakdown a project into smaller steps to better utilize Github Copilot capabilities
- Importance of Prompting with the Three “S” principle
- Show-cased a simple example of Network automation use-case using Copilot



Okay that's great, but what-if I didn't have
to **trigger** my script **manually** every time!

UC2: Network Device Workflow Creation

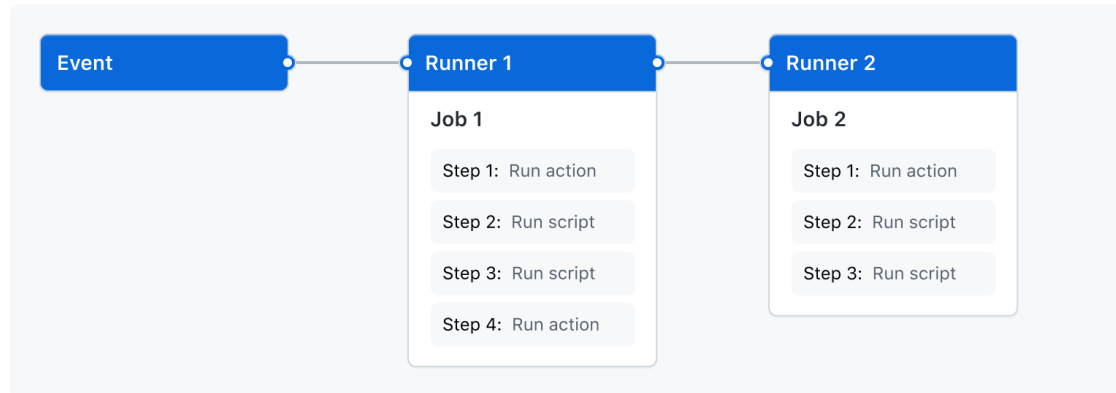
Github Actions – Workflows

Workflows are configurable automated process that will run one or more jobs.

Workflows will run when triggered by an **Event** in your repository, or they can be triggered manually, or at a defined schedule.

Workflows YAML files are defined in the `.github/workflows` directory.

A **Runner** is a server that runs your workflows when they're triggered.



Github Actions – Runner Setup

The screenshot shows the GitHub Actions 'Runners' configuration page. The left sidebar contains navigation links: General, Access, Collaborators, Code and automation (with sub-links for Branches, Tags, Rules, Actions, and Webhooks), and Runners (which is currently selected). The main content area is titled 'Runners' and includes a 'New self-hosted runner' button. Below this is a table listing the configured runners.

Runners	Status
GT-MAC1 self-hosted macOS X64 gtmac	● Idle ...
WIN-Q73ABTFIH81 self-hosted X64 Windows win	● Idle ...

Github Actions – Runner Setup

The screenshot shows the GitHub Actions 'Runners' configuration page. The left sidebar contains navigation links: Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings (highlighted). Under 'Settings', there are sections for General, Access, Collaborators, Code and automation (with sub-links for Branches, Tags, Rules, and Actions), Runners (highlighted), Webhooks, Codespaces, Pages, Security (with sub-links for Code security, Deploy keys, and Secrets and variables), and Integrations. The main content area is titled 'Runners / Add new self-hosted runner · gtilm1se1/BRKOPS-1726'. It includes a paragraph explaining the requirements for adding a self-hosted runner, a 'Runner image' section with three buttons for macOS, Linux, and Windows (Linux is selected), an 'Architecture' section with a dropdown menu set to 'x64', and a 'Download' section containing a terminal script for installing the runner on Linux.

Runners / Add new self-hosted runner · gtilm1se1/BRKOPS-1726

Adding a self-hosted runner requires that you download, configure, and execute the GitHub Actions Runner. By downloading and configuring the GitHub Actions Runner, you agree to the [GitHub Terms of Service](#) or [GitHub Corporate Terms of Service](#), as applicable.

Runner image

☒ macOS ☐ Linux ☐ Windows

Architecture

x64

Download

```
# Create a folder
$ mkdir actions-runner && cd actions-runner
# Download the latest runner package
$ curl -o actions-runner-osx-x64-2.321.0.tar.gz -L https://github.com/actions/runner/releases/download/v2.321.0/actions-runner-osx-x64-2.321.0.tar.gz
# Optional: Validate the hash
$ echo "b2c91416b3e4d579ae69fc2c381fc50dbda13f1b3fcc283187e2c75d1b173072  actions-runner-osx-x64-2.321.0.tar.gz" | shasum -a 256 -c
# Extract the installer
$ tar xzf ./actions-runner-osx-x64-2.321.0.tar.gz
```

UC2: Network Device Workflow Creation

Objective: Create a workflow to automate a method of procedure (MOP) for device provisioning

Workflow automation steps:

1. Connect to Cisco switch
2. Collect pre-check commands
3. Configure new Vlan
4. Collect post-check commands
5. Ensure configuration was successful

UC2: Network Device Workflow Creation

Functional Steps:

1. Utilize “[testbed.yml](#)” file from DEMO-1 with device connectivity details
2. Utilize “[show_cmds.py](#)” file from DEMO-2 to collect pre and post check commands
3. Create “[create_vlan.py](#)” file to provision new VLAN 1111
4. Create “[diff_compare.sh](#)” file to confirm VLAN creation
5. Create a “[github-ci.yml](#)” workflow file with all steps in sequence to execute as single flow

UC2: Network Device Workflow Creation

Prompts for “`create_vlan.py`”:

Import `genie` testbed, `logging`, and `datetime` libraries, then load the `testbed.yml` file

Define function `configure_vlan` which will create and configure a `vlan_id` and `vlan_name`

Connect to device “FE1” and set `prompt_recovery=true`

Configure VLAN 1111 with name test

Disconnect from the device

Improve overall code debuggability

UC2: Network Device Workflow Creation

Prompts for “diff_compare.sh”:

create a shell script that diff compares two files that start with the name command outputs. And remove unnecessary lines in diff-output

```
files=(command_outputs*)
if [ ${#files[@]} -eq 2 ]; then
    diff -u "${files[0]}" "${files[1]}" | grep -E "^\\+|^-"
else
    echo "There are not exactly two files matching the pattern."
fi
```

UC2: Network Device Workflow Creation

- [Github-ci.yml](#) – Pipeline File ->
- Located - .github/workflows/ dir
- Triggers on git push/pull request
- Contains 6 Jobs:
 - Build, Pre-Check, Create-Vlan,
 - Post-Check, Diff-Compare, Cleanup
- Executes the individual scripts in sequence

```
1  name: BRKOPS Pipeline
2  on:
3    push:
4      branches: [ main ]
5    pull_request:
6      branches: [ main ]
7  jobs:
8    Build:
9      runs-on:
10       - macos
11      steps:
12       - uses: actions/checkout@v2
13       - name: Set up Python 3.12
14         uses: actions/setup-python@v3
15         with:
16           python-version: "3.12"
17       - name: Install dependencies
18         run: |
19           python3 -m pip install --upgrade pip
20           pip3 install flake8 pytest pyats genie
21           if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
22    Pre-check:
23      needs: Build
24      runs-on:
25       - macos
26      steps:
27       - name: Run Pre-Checks
28         run: |
29           pwd
30           ls -ltr
31           cd uc2/demo/
32           python sh_cmds.py
33           ls -ltr
34
```

UC2: Network Device Workflow Creation

The screenshot displays the GitHub Actions interface for a workflow named 'BRKOPS Pipeline'. The workflow run is titled 'UC2 updated ci file #12' and is marked as successful with a green checkmark. The interface includes a navigation bar with tabs for Code, Issues, Pull requests, Actions (selected), Projects, Security, Insights, and Settings. On the left, a sidebar shows the 'Summary' tab selected, with a list of jobs: Build, Pre-check, Create-Vlan, Post-check, Diff-Compare, and Cleanup, all marked as successful. The main content area shows the workflow run details, including the trigger 'gtamilise1 pushed' and the status 'Success'. Below this, the 'github-ci.yml' file content is displayed, showing a sequence of steps: Build (15s), Pre-check (1m 45s), Create-Vlan (1m 13s), Post-check (1m 27s), Diff-Compare (3s), and Cleanup (3s). The interface also includes a 'Re-run all jobs' button and a 'Run details' section with links for Usage and Workflow file.

← BRKOPS Pipeline

✓ UC2 updated ci file #12

Re-run all jobs ...

Summary

Jobs

- ✓ Build
- ✓ Pre-check
- ✓ Create-Vlan
- ✓ Post-check
- ✓ Diff-Compare
- ✓ Cleanup

Run details

- Usage
- Workflow file

Triggered via push 42 minutes ago

Status: Success

Total duration: 5m 24s

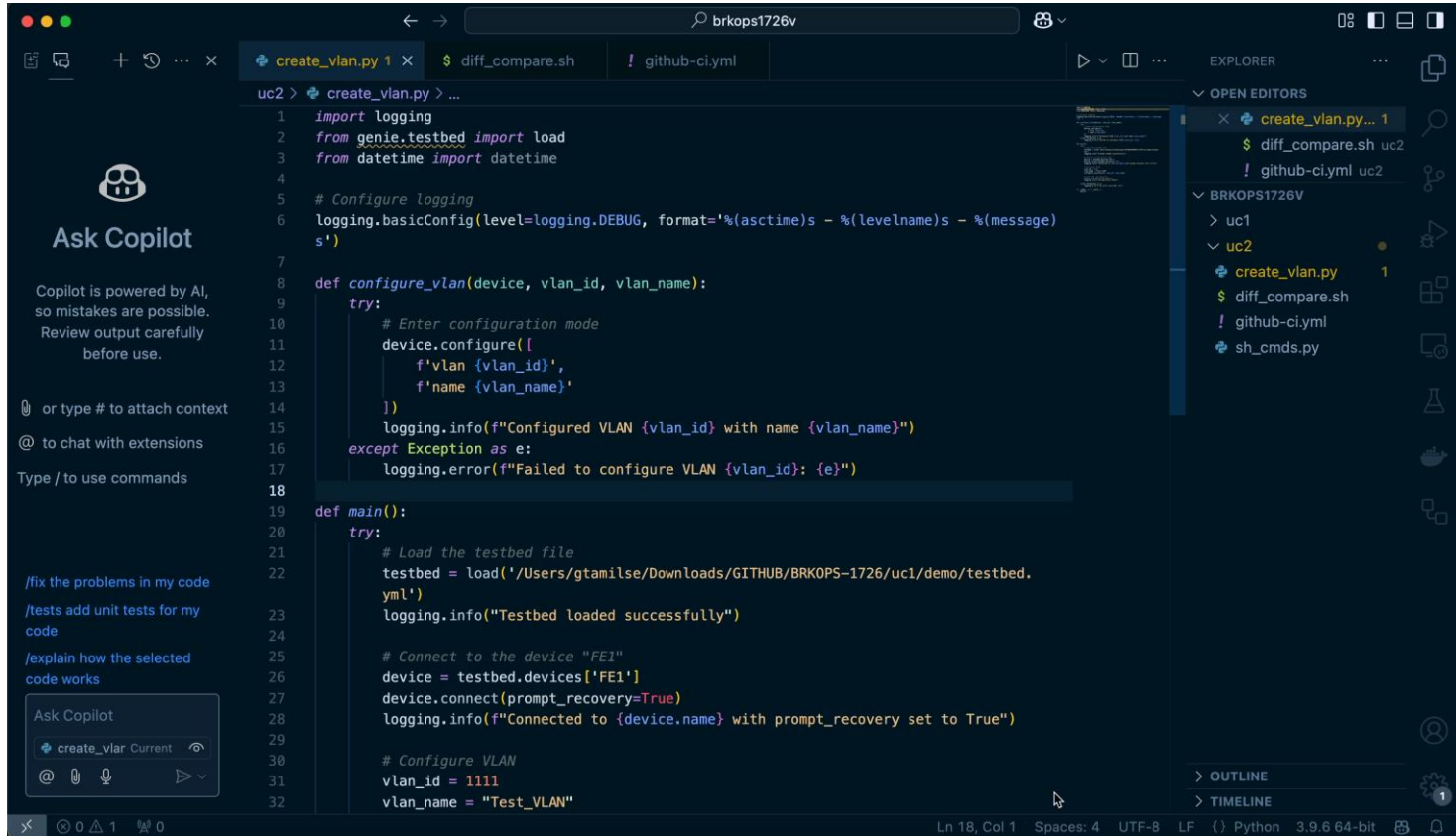
Artifacts: -

github-ci.yml

on: push

Build (15s) → Pre-check (1m 45s) → Create-Vlan (1m 13s) → Post-check (1m 27s) → Diff-Compare (3s) → Cleanup (3s)

Demo 3 – Network Device Workflow Creation




```
uc2 > create_vlan.py > ...
1 import logging
2 from genie.testbed import load
3 from datetime import datetime
4
5 # Configure logging
6 logging.basicConfig(level=logging.DEBUG, format='%asctime)s - %(levelname)s - %(message)s')
7
8 def configure_vlan(device, vlan_id, vlan_name):
9     try:
10         # Enter configuration mode
11         device.configure([
12             f'vlan {vlan_id}',
13             f'name {vlan_name}'
14         ])
15         logging.info(f"Configured VLAN {vlan_id} with name {vlan_name}")
16     except Exception as e:
17         logging.error(f"Failed to configure VLAN {vlan_id}: {e}")
18
19 def main():
20     try:
21         # Load the testbed file
22         testbed = load('/Users/gtamilse/Downloads/GITHUB/BRKOPS-1726/uc1/demo/testbed.
23             yml')
24         logging.info("Testbed loaded successfully")
25
26         # Connect to the device "FE1"
27         device = testbed.devices['FE1']
28         device.connect(prompt_recovery=True)
29         logging.info(f"Connected to {device.name} with prompt_recovery set to True")
30
31         # Configure VLAN
32         vlan_id = 1111
33         vlan_name = "Test_VLAN"
```

Demo-3

UC2: Network Device Workflow Creation Recap

- Learned about Github Actions feature to create workflows that can trigger automatically
- Utilized Github Copilot to create sequential scripts that achieve specific tasks
- Show-cased a simple example of Network Method of Procedure (MOP) automation



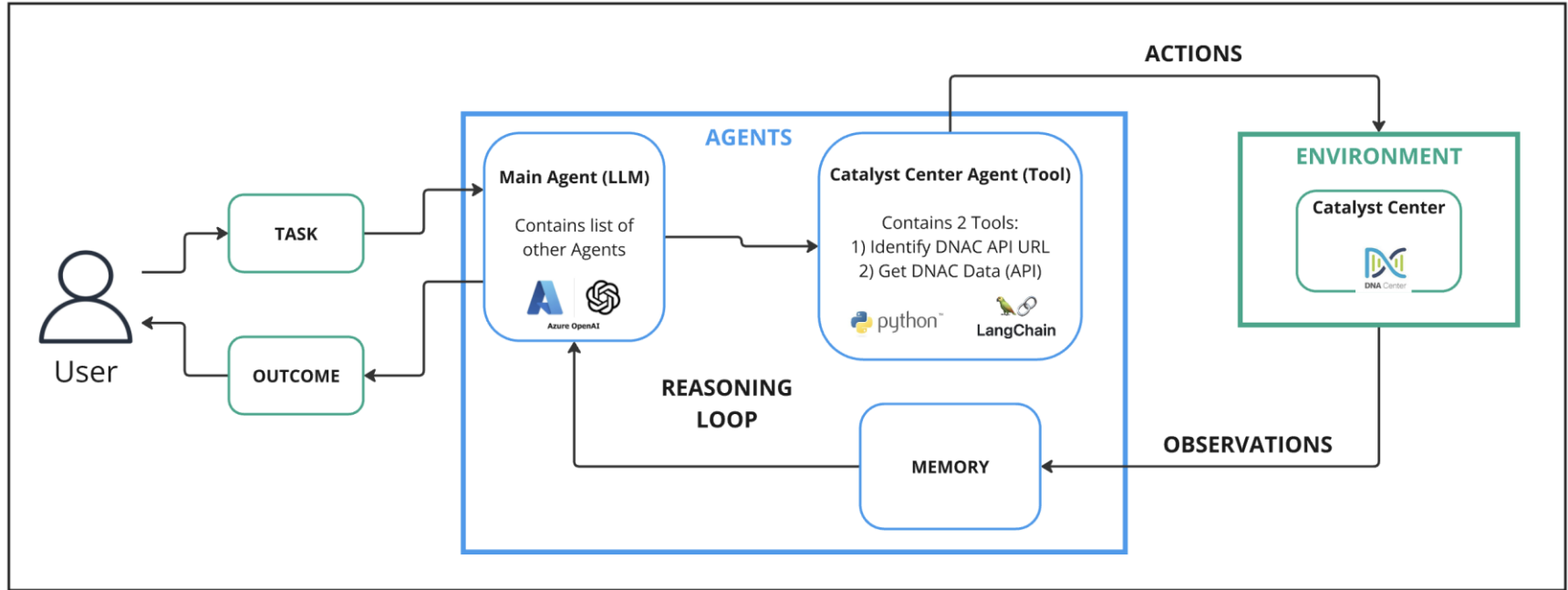
Hey that's cool, but what about automation
using **Controllers** and can **GenAI help**?

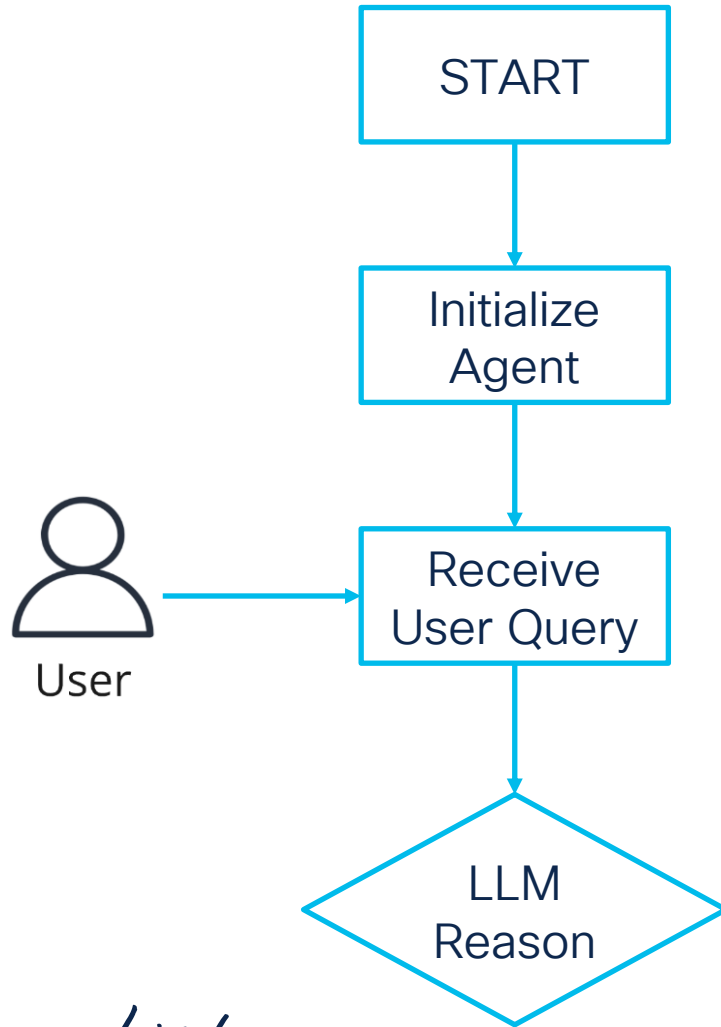
UC3: Gen AI Agent based Automation

GEN AI – ReAct Framework

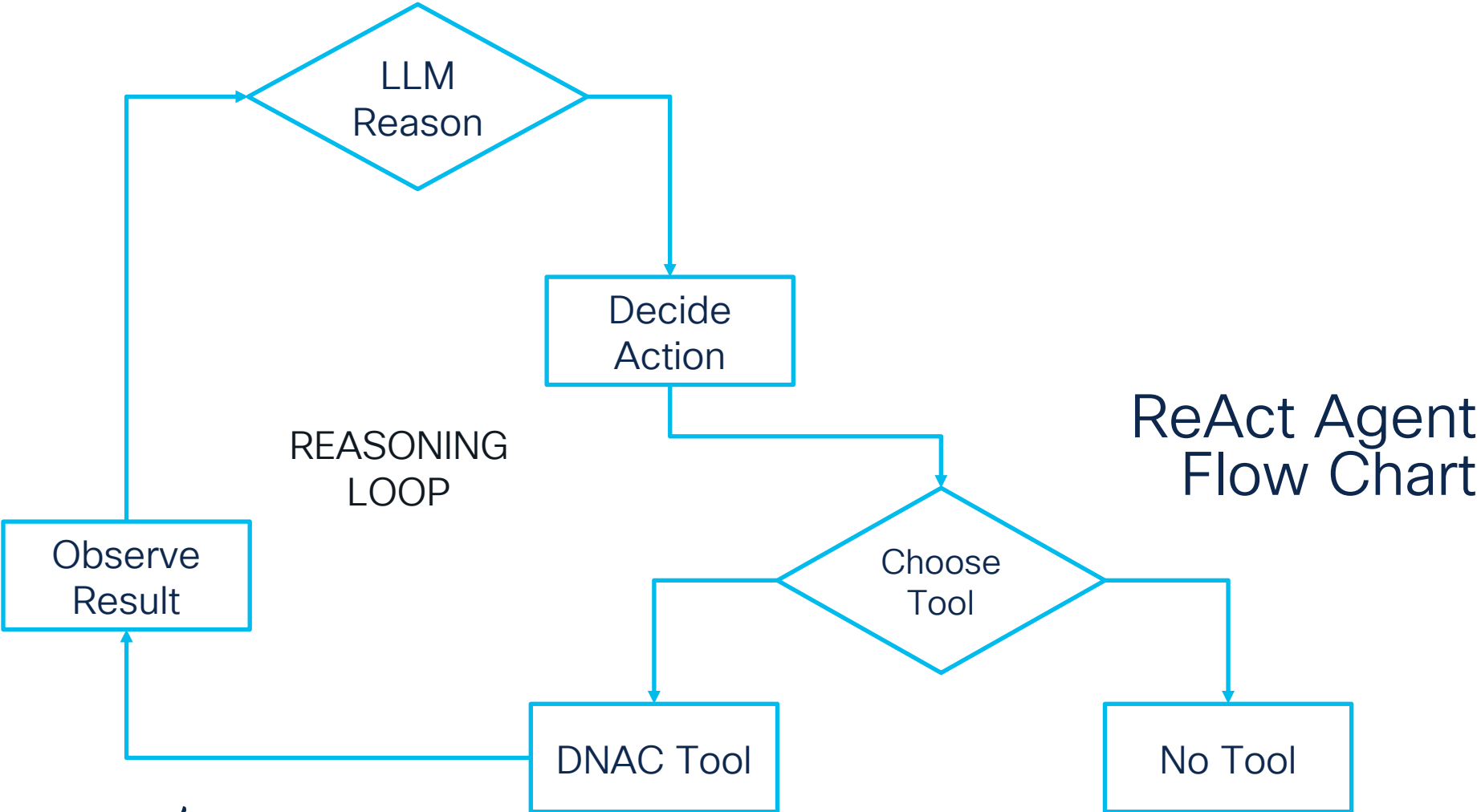
- *Reasoning and Acting (ReACT):*
 - ReAct agents utilize **LLMs** as centralized components (brain) that concurrently reason about the environment and determine appropriate actions.
 - The agent processes **generate plans (Thought), execute Actions, and Observe the response** in an iterative cycle.
 - ReAct agents can integrate with **external Tools** (scripts/code) and **APIs**, selecting and employing them based on the current context and objectives.

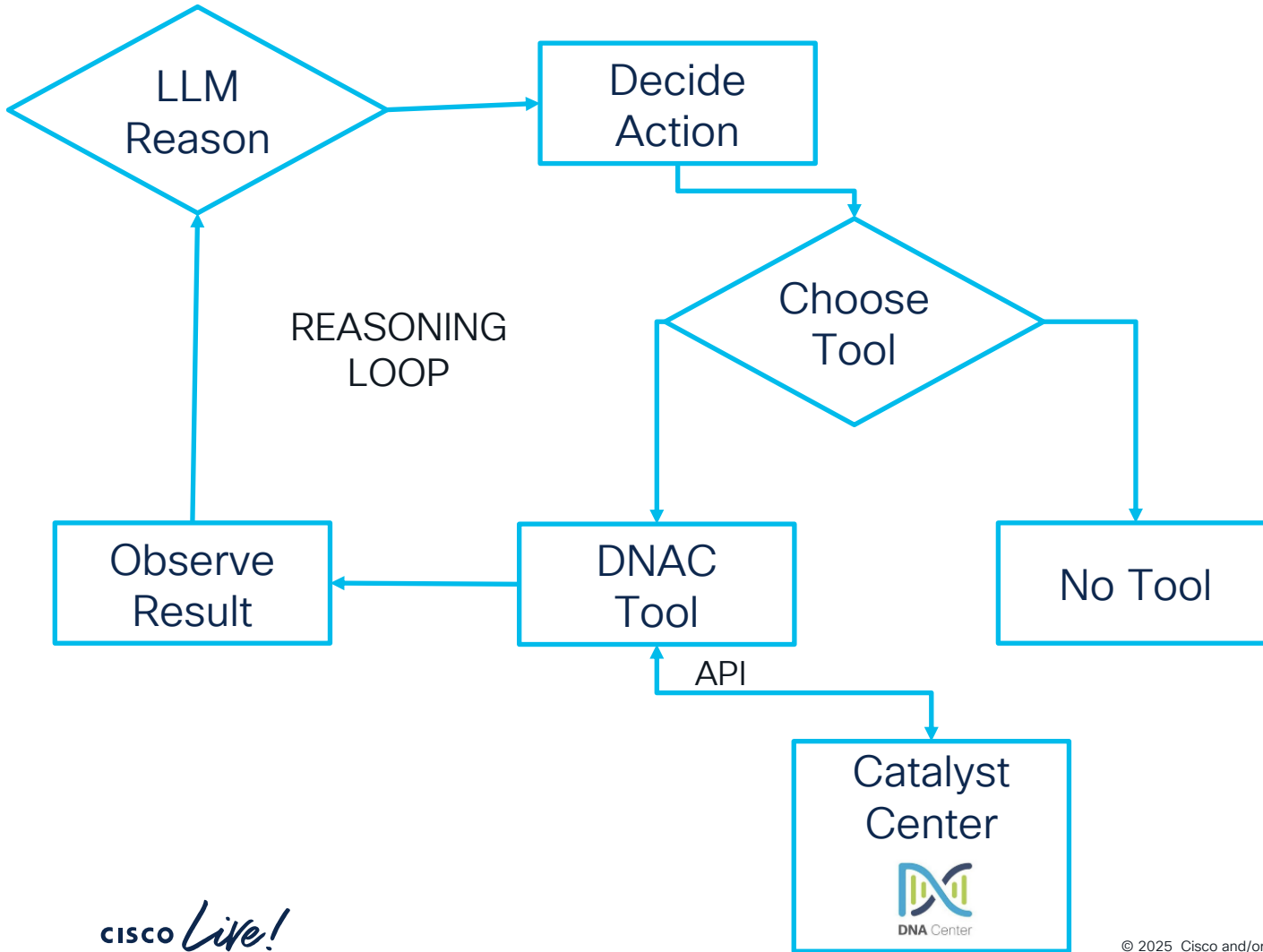
ReAct Agent Architecture



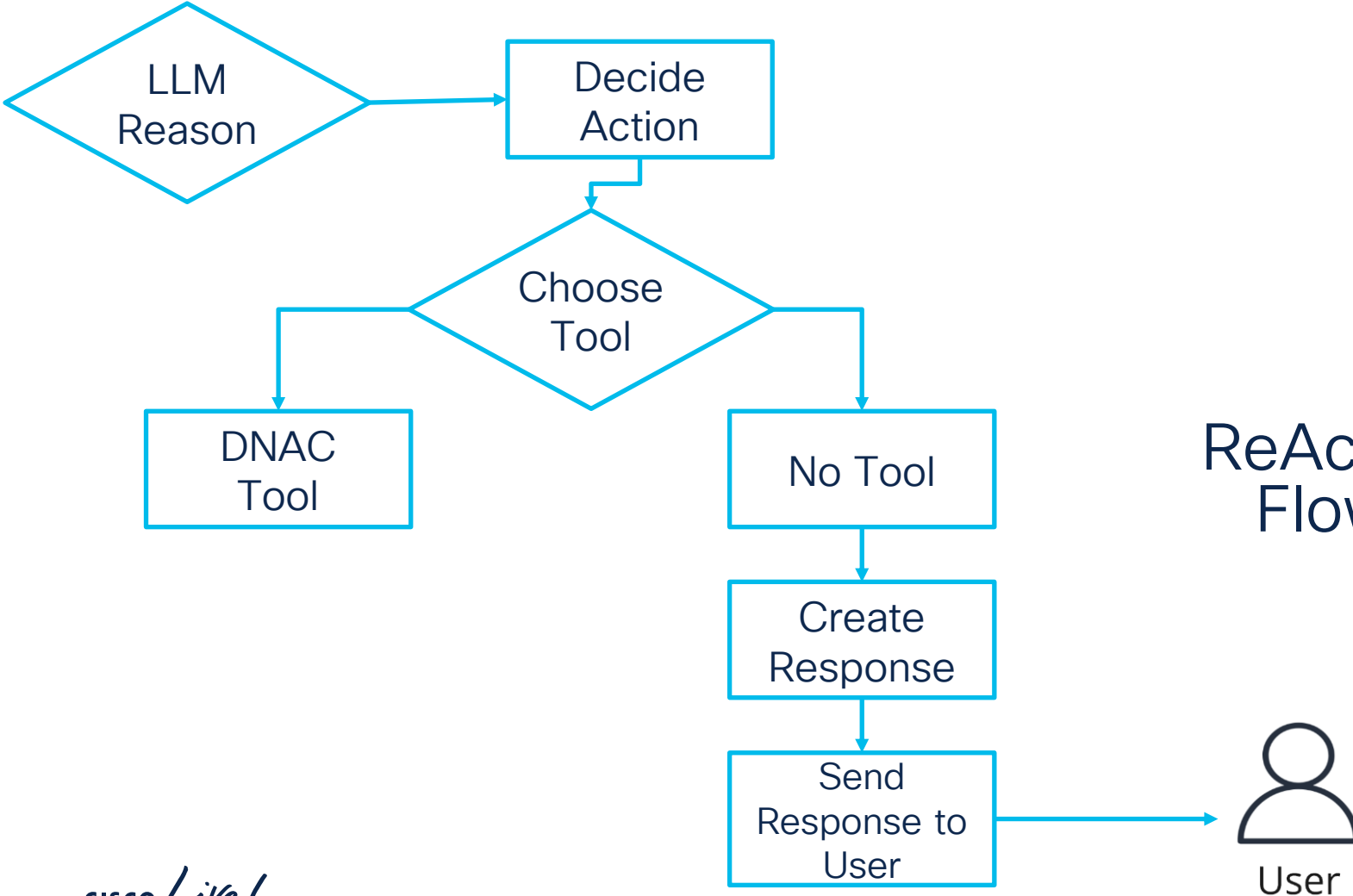


ReAct Agent Flow Chart





ReAct Agent Flow Chart



ReAct Agent Crafting a Prompt Template

ReAct prompt typically includes four components:

1. Current user query
2. Any previous reasoning steps and observations (memory)
3. Available tools
4. Output format instructions

ReAct Agent Prompt Template

```
template = ""
```

Agent is a network assistant with the capability to manage data from Cisco Catalyst Center controllers using API Requests.

Network Instructions:

Assistant is designed to retrieve information from the Cisco Catalyst Center controller using provided tools. You MUST use these tools for checking available data and fetching that data.

Assistant has access to a list of API URLs and their associated Names provided in a 'dnac_urls.json' file. You can use the 'Name' field to find the appropriate API URL to use.

ReAct Agent Prompt Template cont'd

****Important Guidelines:****

1. ****If you are certain of the API URL or the Name of the data you want, use the `'get_dnac_data_tool'` to fetch data.****
2. ****If you are unsure of the API URL or Name, or if there is ambiguity, use the `'check_supported_url_tool'` to verify the URL or Name or get a list of available ones.****
3. ****If the `'check_supported_url_tool'` finds a valid URL or Name, then use the appropriate tool to perform the action.****
4. ****Do NOT use any unsupported URLs or Names.****

ReAct Agent Prompt Template cont'd

To use a tool, follow this format:

Thought: Do I need to use a tool? Yes

Action: the action to take, should be one of [{tool_names}]

Action Input: the input to the action

Observation: the result of the action

If the first tool provides a valid URL or Name, you MUST immediately run the correct tool for the operation (fetch, create, update, or delete) without waiting for another input.

"""

UC3: Gen AI Agent based Automation

Objective: Utilize OpenAI ChatGPT and Copilot to create a ReAct Agent that performs “Get” requests to Catalyst Center via API

5 Components:

1. **Env File** – which contains the DNAC Server IP, Username, and Password
2. **Main Agent** –the ReAct Agent that utilizes LLM (OpenAI GPT-4o-mini)
3. **DNAC Agent** – which contains python code to check and make API calls to DNAC; as known as the “tool agent”
4. **DNAC APIs List** – contains a list of Catalyst Center APIs in JSON format for DNAC agent tool to use
5. **Browser based Chat interface** – handles user queries and displays the response as well as past conversation history

UC3: Gen AI Agent based Automation

Functional Steps:

1. Create an “dnac_urls.json” file that contains list of Catalyst Center APIs and their names.
2. Create the “dnac_agent.py” file which contains 2 Tools (python code):
 - a) Check the dnac_urls.json file to find the closest matching API call to user query
 - b) Executes the api call against Catalyst Center
3. Create the “main_agent.py” that initializes the LLM Agent (OpenAI) and connects it to the dnac tools. Also adds frontend chat interface.
4. Initiate user request in chat window and observe ReAct Agent retrieve results from Catalyst Center.

Cisco DNA Center APIs

Cisco DNA Center

Design

Policy

Provision

Assurance

Workflows

Tools

Platform

Activities

Reports

System

Explore

Overview

Manage

Developer Toolkit

Runtime Dashboard

<<

Cisco DNA Center

Platform / Developer Toolkit

APIs Integration Flows Event Notifications

Search

Authentication

Cisco DNA Center System

Disaster Recovery

Health and Performance

Licenses

Platform

User and Roles

Connectivity

Fabric Wireless

SDA

Wireless

Ecosystem Integrations

ITSM

Event Management

Integrations

Search API

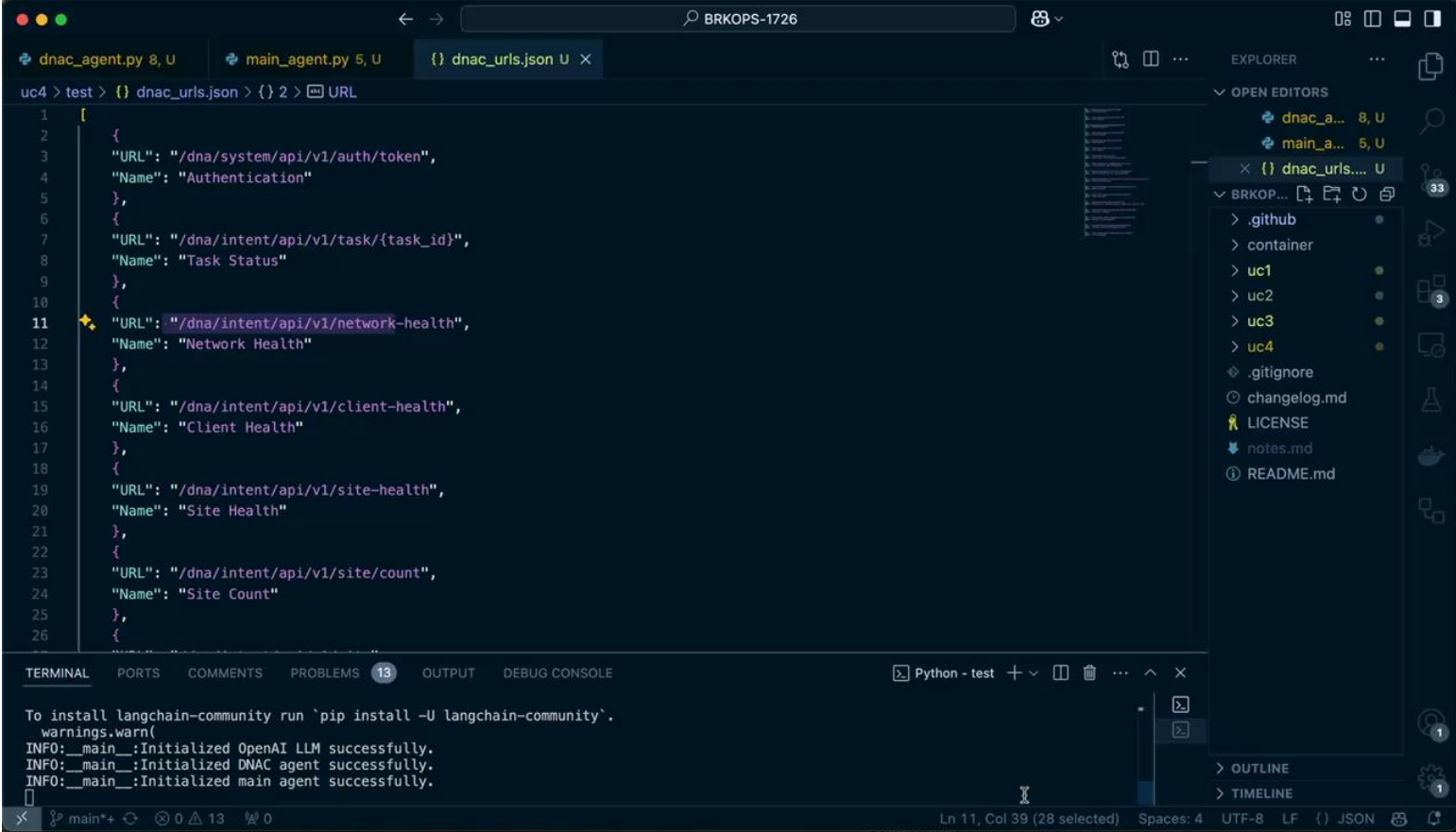
Authentication

Authentication APIs provide an authorized token for accessing any REST API.
***Prerequisite*:** Add the request header 'x-auth-token' with the generated authorized token to get a successful API response.

Method	Name	Description	URL	Actions
POST	importCertificate	This method is used to upload a certificate	/certificate	...
POST	importCertificateP12	This method is used to upload a PKCS#12 file	/certificate-p12	...
POST	Authentication API	API to obtain an access token, which remains valid for 1 hour. The token obtained using this API is required to be set as value to the X-Auth-Token HTTP...	/auth/token	...

Cisco DNA Center System

Demo 4 – Gen AI Agent based Automation



```
uc4 > test > {} dnac_urls.json > {} 2 > URL
1
2 {
3   "URL": "/dna/system/api/v1/auth/token",
4   "Name": "Authentication"
5 },
6 {
7   "URL": "/dna/intent/api/v1/task/{task_id}",
8   "Name": "Task Status"
9 },
10 {
11   "URL": "/dna/intent/api/v1/network-health",
12   "Name": "Network Health"
13 },
14 {
15   "URL": "/dna/intent/api/v1/client-health",
16   "Name": "Client Health"
17 },
18 {
19   "URL": "/dna/intent/api/v1/site-health",
20   "Name": "Site Health"
21 },
22 {
23   "URL": "/dna/intent/api/v1/site/count",
24   "Name": "Site Count"
25 },
26 {
27
```

TERMINAL

Python - test

```
To install langchain-community run 'pip install -U langchain-community'.
warnings.warn(
INFO:_main_:Initialized OpenAI LLM successfully.
INFO:_main_:Initialized DNAC agent successfully.
INFO:_main_:Initialized main agent successfully.
```

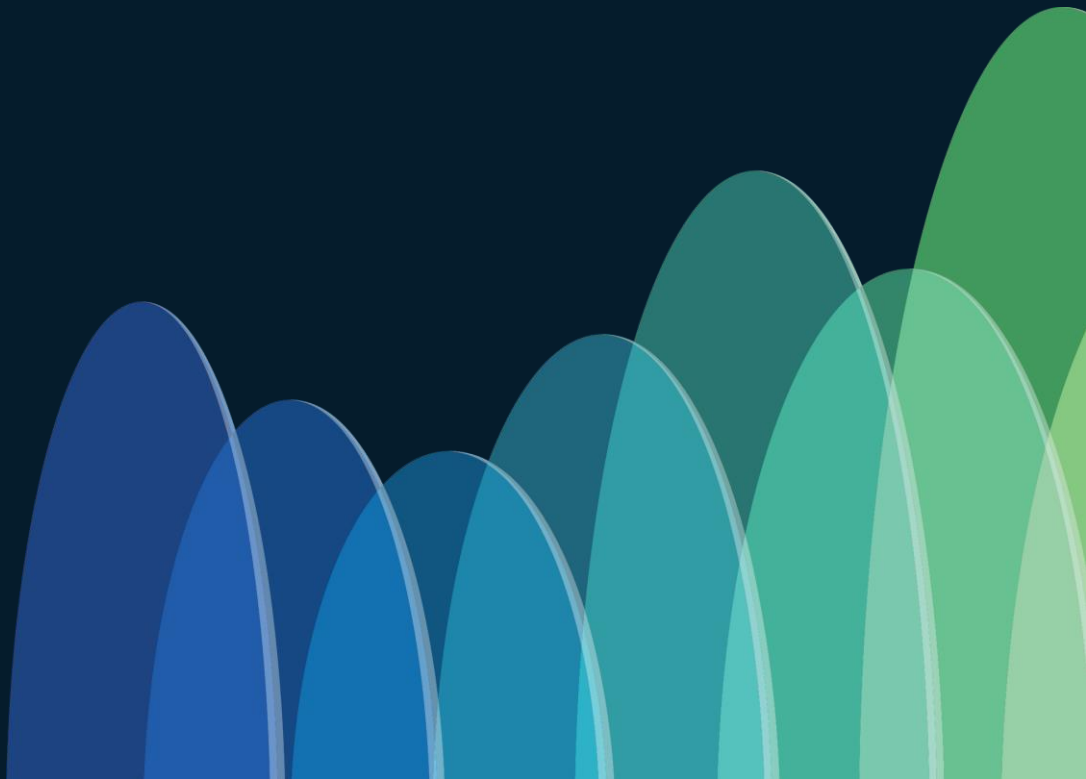
Demo-4

cisco *Live!*

UC3: Gen AI Agent Based Automation Recap

- Learned about ReAct Framework and how to integrate LLM models into automation projects
- Utilize the Thought-Action-Observation loop via Prompt Engineering to influence LLM behavior
- Show-cased how agent based architecture allows LLMs to utilize Tools to act on behalf of the user

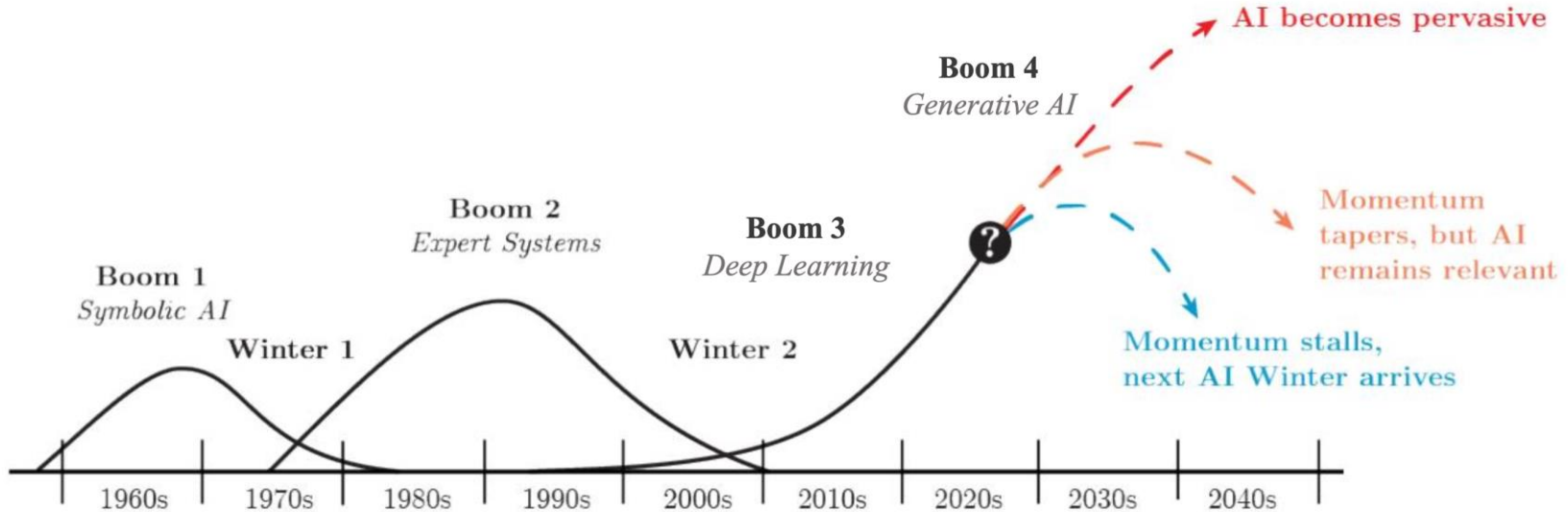
Conclusion



Github Copilot Best Practice Recommendations

1. You are the main architect and Copilot is your assistant.
2. Outline the project pseudo-code before starting and break down the steps.
3. Keep prompts simple, specific, and short.
4. Use Inline Chat to build code in small steps.
5. Use available Copilot features (/functions) for quick support.

Where do we go with AI?





On a scale of 1 to 5, how
excited are you to go try
Copilot and AI Automation
now?

① Start presenting to display the poll results on this slide.

Webex App

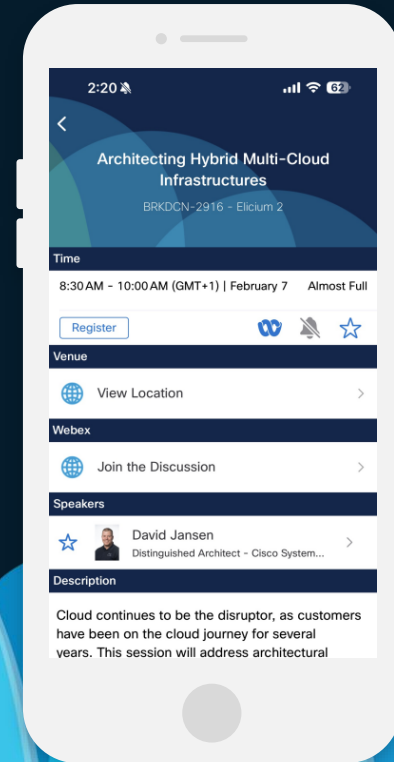
Questions?

Use the Webex app to chat with the speaker after the session

How

- 1 Find this session in the Cisco Events mobile app
- 2 Click “Join the Discussion”
- 3 Install the Webex app or go directly to the Webex space
- 4 Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until February 28, 2025.



Fill Out Your Session Surveys



Participants who fill out a minimum of 4 session surveys and the overall event survey will get a unique Cisco Live t-shirt.

(from 11:30 on Thursday, while supplies last)



All surveys can be taken in the Cisco Events mobile app or by logging in to the Session Catalog and clicking the 'Participant Dashboard'



Content Catalog

Continue your education



- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at ciscolive.com/on-demand. Sessions from this event will be available from March 3.

Contact us at: [BRKOPS-1726 Webex Space](#)



Thank you

CISCO *Live!*



CISCO *Live!*

GO BEYOND

The background of the slide features a series of overlapping, teardrop-shaped elements in various shades of blue, ranging from light sky blue to deep navy blue. These shapes are arranged in a way that creates a sense of depth and movement, resembling a stylized mountain range or a series of waves. The overall aesthetic is clean and modern.