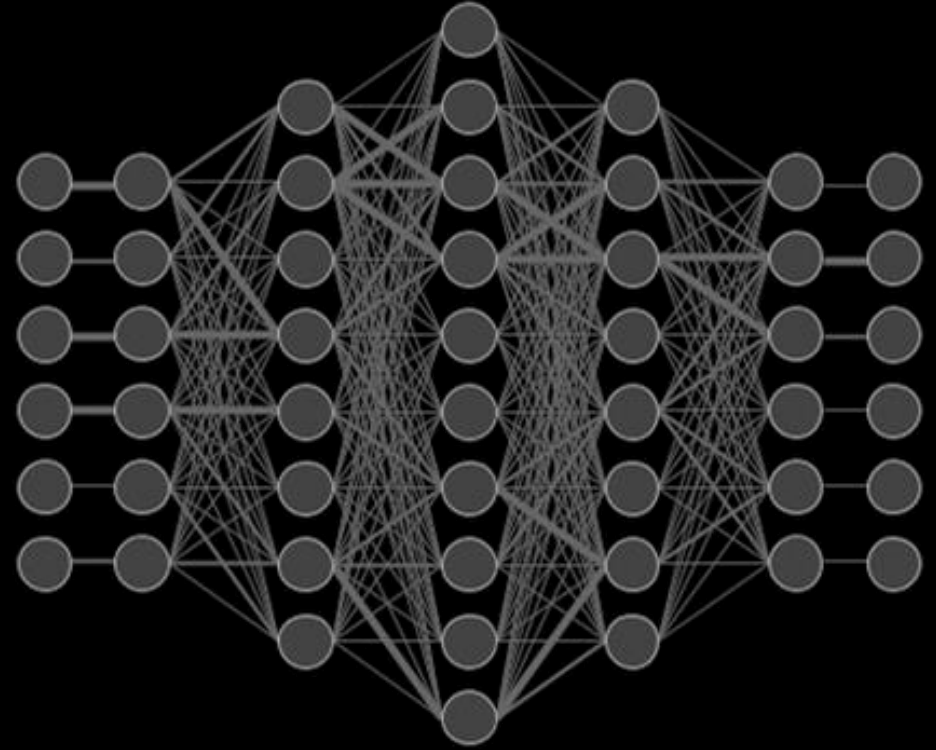


Microsoft와 함께하는
Azure OpenAI PromptThon



Microsoft 기술전략매니저 김현수

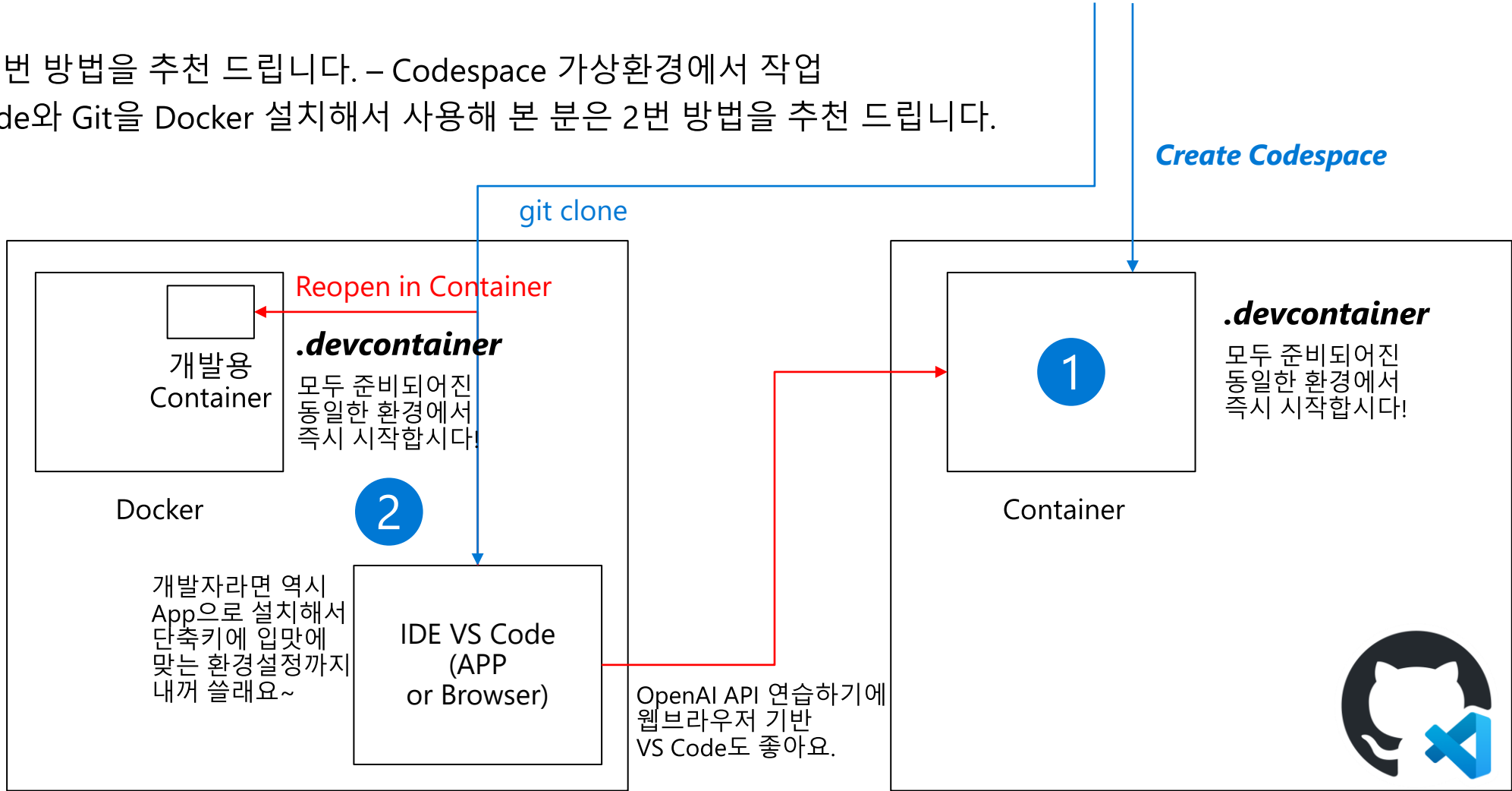
Agenda

1. 코드 기반 실습 소개

실습 환경 선택

<https://github.com/HyounsooKim/azure-openai-samples-kr/>

보통 1번 방법을 추천 드립니다. – Codespace 가상환경에서 작업
VS Code와 Git을 Docker 설치해서 사용해 본 분은 2번 방법을 추천 드립니다.



내 PC에서

OpenAI URL, Key, Model 정보만 입력하면
즉시 사용 가능한 환경

Codespace

“ GitHub - Codespace 등장 배경

「새로 조인한 개발자 환경 구축」

「내 Labtop에서는 동작하는데, 왜 다른 labtop에서는 동작하지 않지?」

「16코어의 PC가 필요한데...」

「컴퓨터에 무엇이 설치되어 있는지 전혀 모르겠습니다」



신속한 클라우드 개발 환경

Team 및 Enterprise Cloud

The image shows a screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left displays the project structure, including folders like .github, .vscode, build, extensions, node_modules, out, remote, resources, scripts, src, test, and files like gulpfile.js, cgmanifest.json, package.json, product.json, tslint.json, and README.md. The main editor area shows the contents of 'gulpfile.js', which includes a license header, 'use strict', and several Gulp tasks for cleaning and compiling client-side code. The terminal window at the bottom shows the output of these tasks, including timestamps and durations for various operations like 'clean-client', 'watch-client', and 'monaco.d.ts generation'.

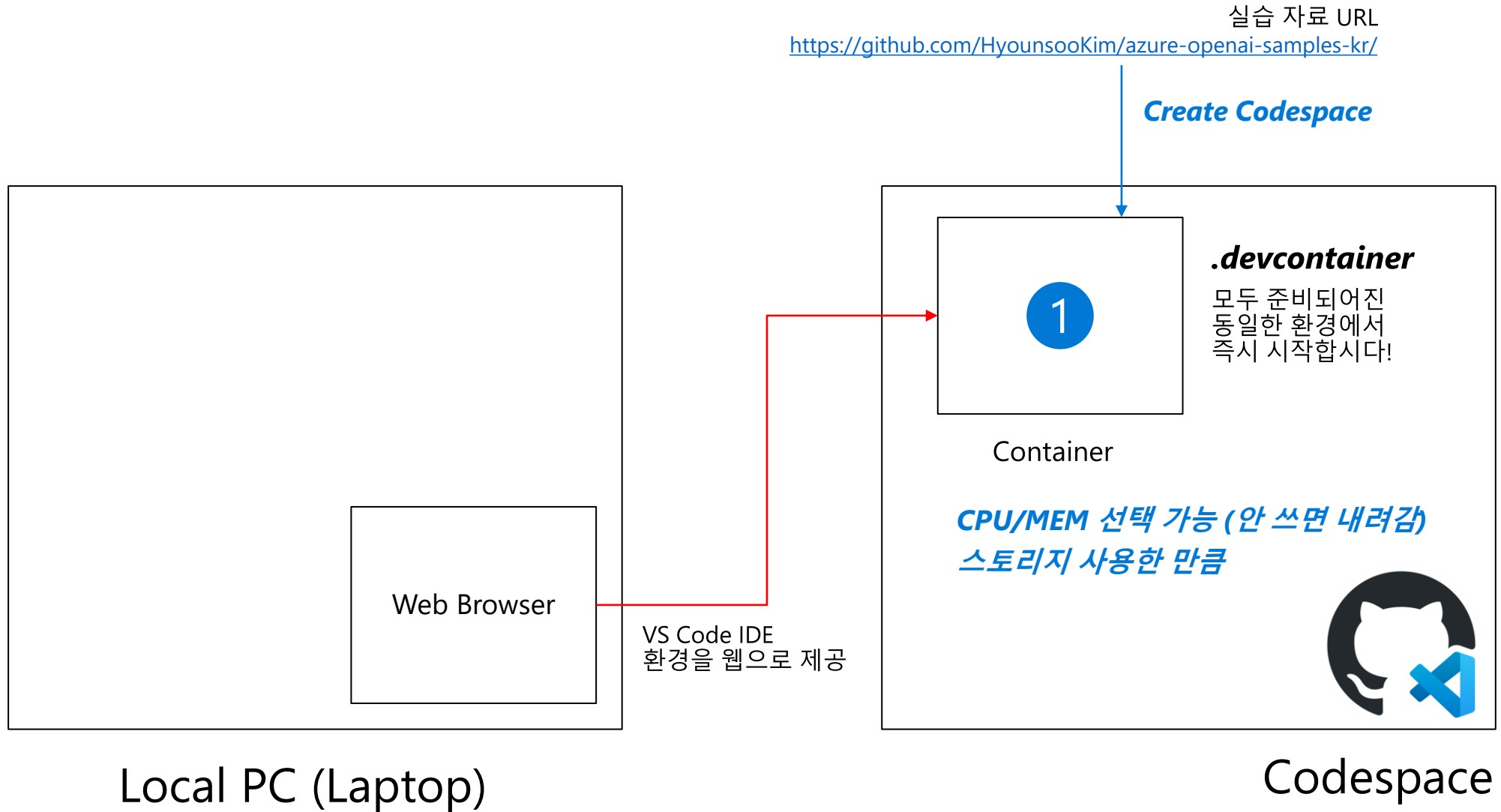
```
1  /*-----*/
2  * Copyright (c) Microsoft Corporation. All rights reserved.
3  * Licensed under the MIT License. See License.txt in the project root for license information.
4  *-----*/
5
6  'use strict';
7
8  // Increase max listeners for event emitters
9  require('events').EventEmitter.defaultMaxListeners = 100;
10
11 const gulp = require('gulp');
12 const util = require('./build/lib/util');
13 const path = require('path');
14 const compilation = require('./build/lib/compilation');
15
16 // Fast compile for development time
17 gulp.task('clean-client', util.rimraf('out'));
18 gulp.task('compile-client', ['clean-client'], compilation.compileTask('out', false));
19 gulp.task('watch-client', ['clean-client'], compilation.watchTask('out', false));
20
21 // Full compile, including nls and inline sources in sourcemaps, for build
22 gulp.task('clean-client-build', util.rimraf('out-build'));
23 gulp.task('compile-client-build', ['clean-client-build'], compilation.compileTask('out-build', true));
24 gulp.task('watch-client-build', ['clean-client-build'], compilation.watchTask('out-build', true));
25
```

Terminal Output:

```
[09:43:36] Starting 'watch-extension:vscode-api-tests'...
[09:43:36] Finished 'clean-extension:typescript-language-features' after 248 ms
[09:43:36] Starting 'watch-extension:typescript-language-features'...
[09:43:36] Finished 'clean-extension:php-language-features' after 384 ms
[09:43:36] Starting 'watch-extension:php-language-features'...
[09:43:40] Finished 'clean-extension:html-language-features-server' after 4.66 s
[09:43:40] Starting 'watch-extension:html-language-features-server'...
[09:43:43] Finished 'clean-client' after 7.33 s
[09:43:43] Starting 'watch-client'...
[09:44:50] [monaco.d.ts] Starting monaco.d.ts generation
[09:44:56] [monaco.d.ts] Finished monaco.d.ts generation
[09:44:56] Finished compilation with 557 errors after 80542 ms
```



Codespace 환경



GitHub Codespaces

- 코드 작성, 애플리케이션 구축, 테스트 실행, 배포, 디버깅과 같은 일련의 모든 작업을 브라우저에 실행
- 웹 기반 Visual Studio Code 또는 선호하는 IDE 사용
- 확장, 설정, 스크립트 및 기능으로 사용자 정의
- Docker 및 Docker Compose를 사용하여 추가로 사용자 정의
- 몇 분 안에 준비된 환경을 통해 새로운 개발자를 빠르게 온보딩
- 조직 내 개발자 환경 표준화
- 데모의 일부로 개발자 환경 구성 배포

추천 방법 1: CodeSpace

<https://github.com/HyounsooKim/azure-openai-samples-kr>

The screenshot shows a GitHub repository page for 'azure-openai-samples-kr'. The repository is public and forked from 'Azure/azure-openai-samples'. The current branch is 'main', which is 2 commits ahead of the upstream 'main' branch. The repository contains several folders and files, including '.devcontainer', 'archive/Labs/Clustering-and-Embedd...', 'documentation/lease_extraction', 'fundamentals', 'quick_start', 'solution_accelerators', 'use_cases', and '.env.sample'. A dropdown menu is open over the 'Code' button, showing options for 'Local' and 'Codespaces'. The 'Codespaces' section indicates that there are no codespaces currently checked out for this repository and provides a prominent green button labeled 'Create codespace on main' which is highlighted with a red rectangle. Below this button is a link to 'Learn more about codespaces...'. At the bottom of the dropdown, it states 'Codespace usage for this repository is paid for by HyounsooKim'.

azure-openai-samples-kr Public
forked from Azure/azure-openai-samples

main 1 branch 0 tags

Go to file Add file <> Code

This branch is 2 commits ahead of Azure:main.

HyounsooKim 기본 환경 설정 및 한글화

- .devcontainer 기본 환경 설정 및 한글화
- archive/Labs/Clustering-and-Embedd... moved from azure-openai-wor
- documentation/lease_extraction moved from azure-openai-wor
- fundamentals [tbarari] Added code for a new
- quick_start Add files via upload
- solution_accelerators Part 2-Video of OpenAI Demo
- use_cases Merge pull request Azure#28 f

.env.sample fix: add missing .environment variables 2 weeks ago

Local Codespaces

Codespaces + ...
Your workspaces in the cloud

No codespaces
You don't have any codespaces with this repository checked out

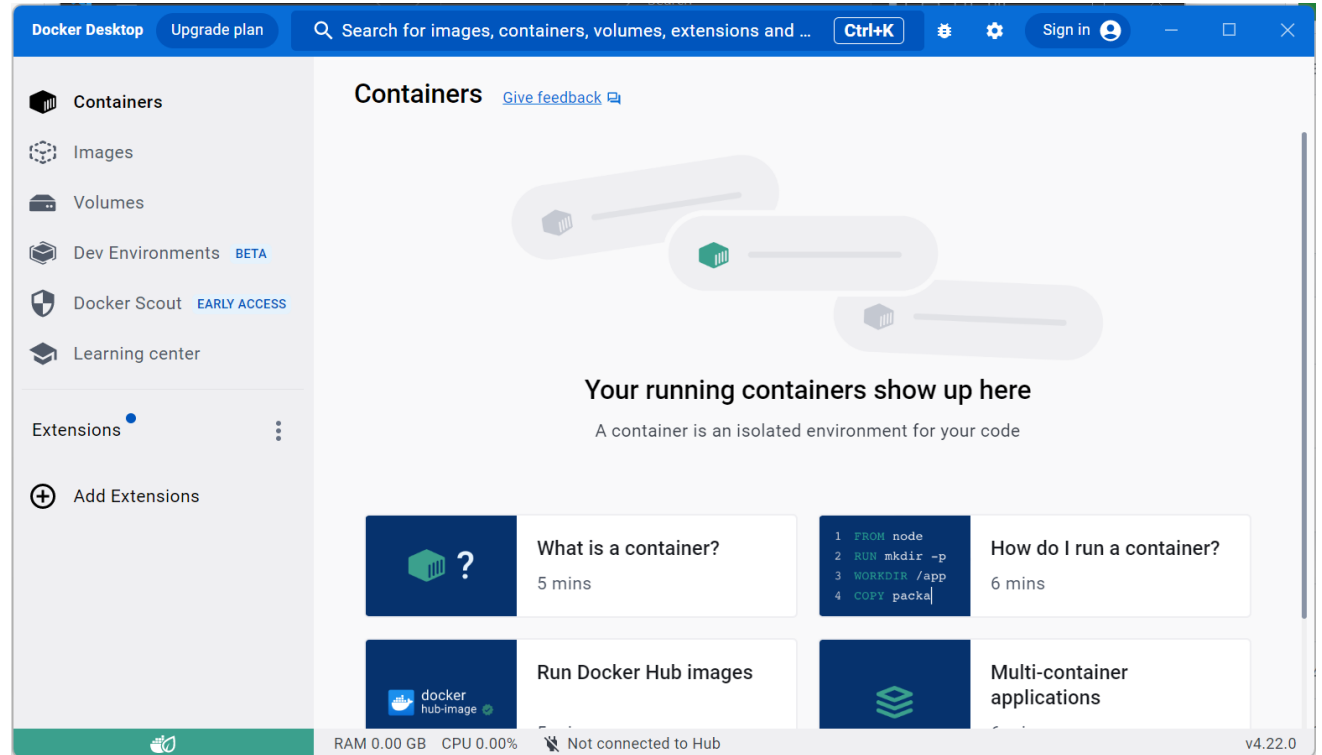
Create codespace on main

[Learn more about codespaces...](#)

Codespace usage for this repository is paid for by HyounsooKim

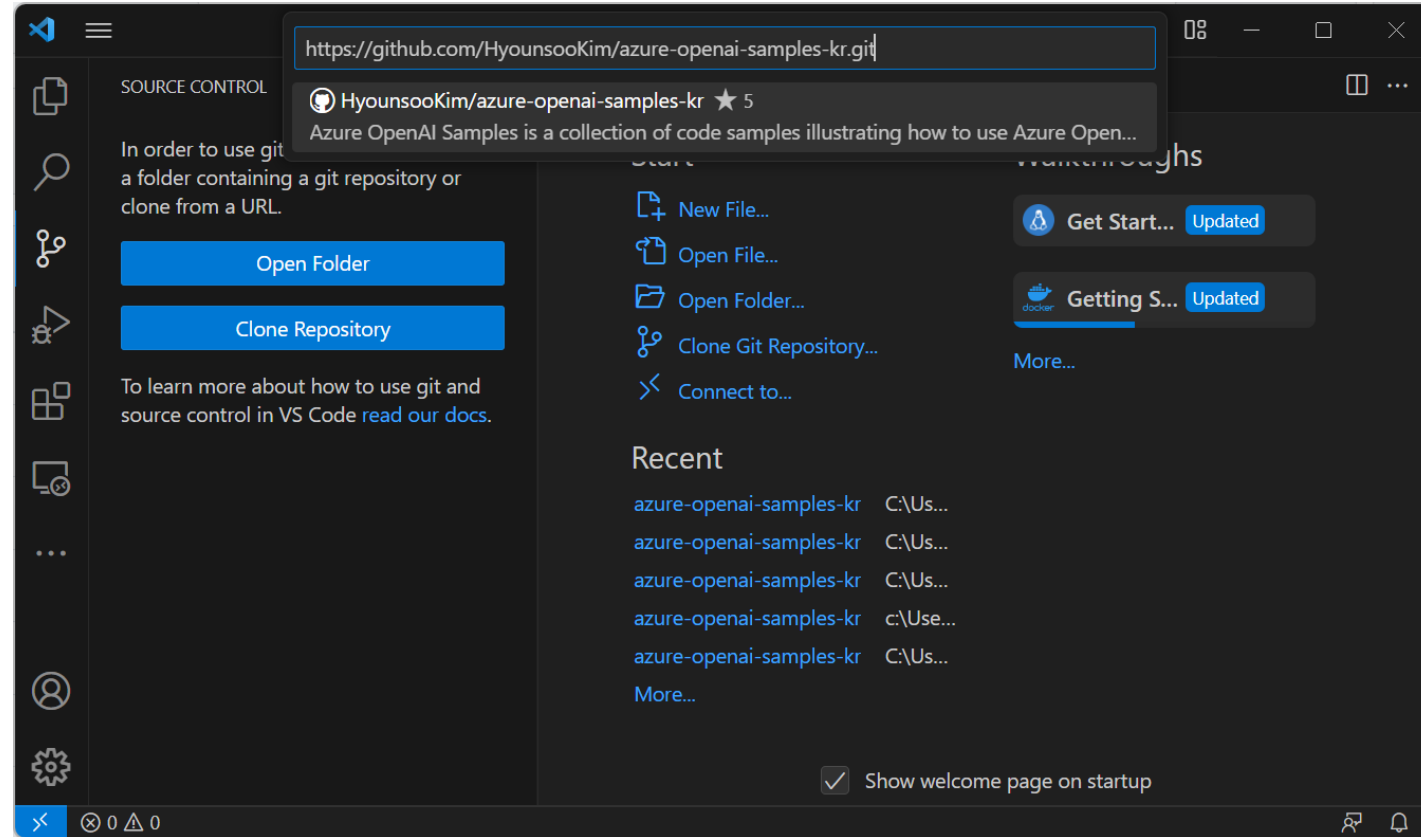
추천 방법 2: Local VS Code

1. **Docker 설치**
2. [Visual Studio Code](#) 설치
3. [원격 - 컨테이너 확장 설치](#)
4. Repo clone
5. Visual Studio Code에서 리포지토리 열기
6. 창의 우측 하단 모서리에 있는 "Reopen in Container" 버튼 클릭
7. 컨테이너가 빌드되고 시작될 때까지 기다립니다.



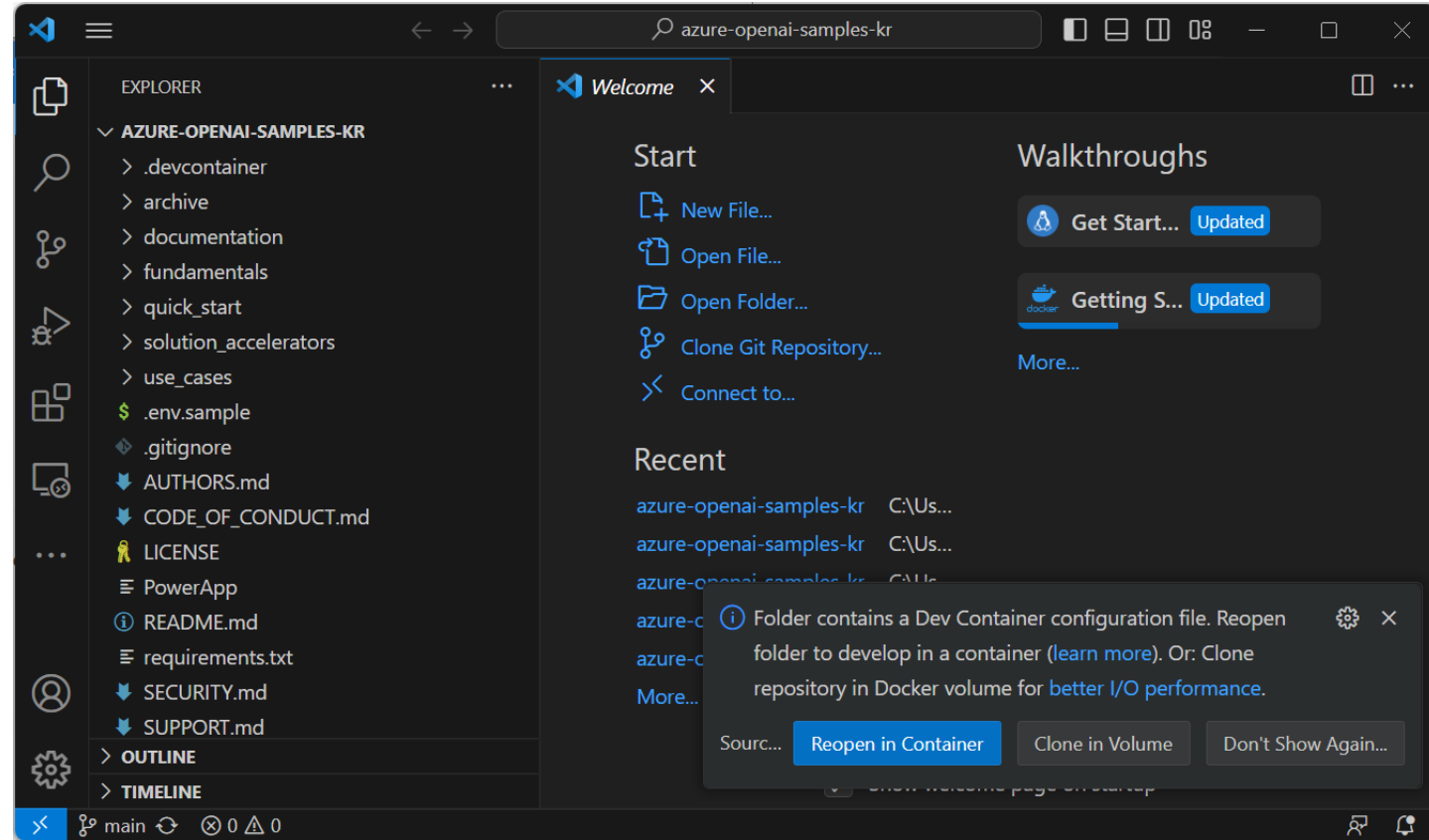
추천 방법 2: Local VS Code

1. [Docker](#) 설치
2. **Visual Studio Code** 설치
3. **원격 - 컨테이너 확장 설치**
4. **Repo clone**
5. **Visual Studio Code에서 리포지토리 열기**
6. 창의 우측 하단 모서리에 있는
"Reopen in Container" 버튼 클릭
7. 컨테이너가 빌드되고 시작될 때까지 기다립니다.



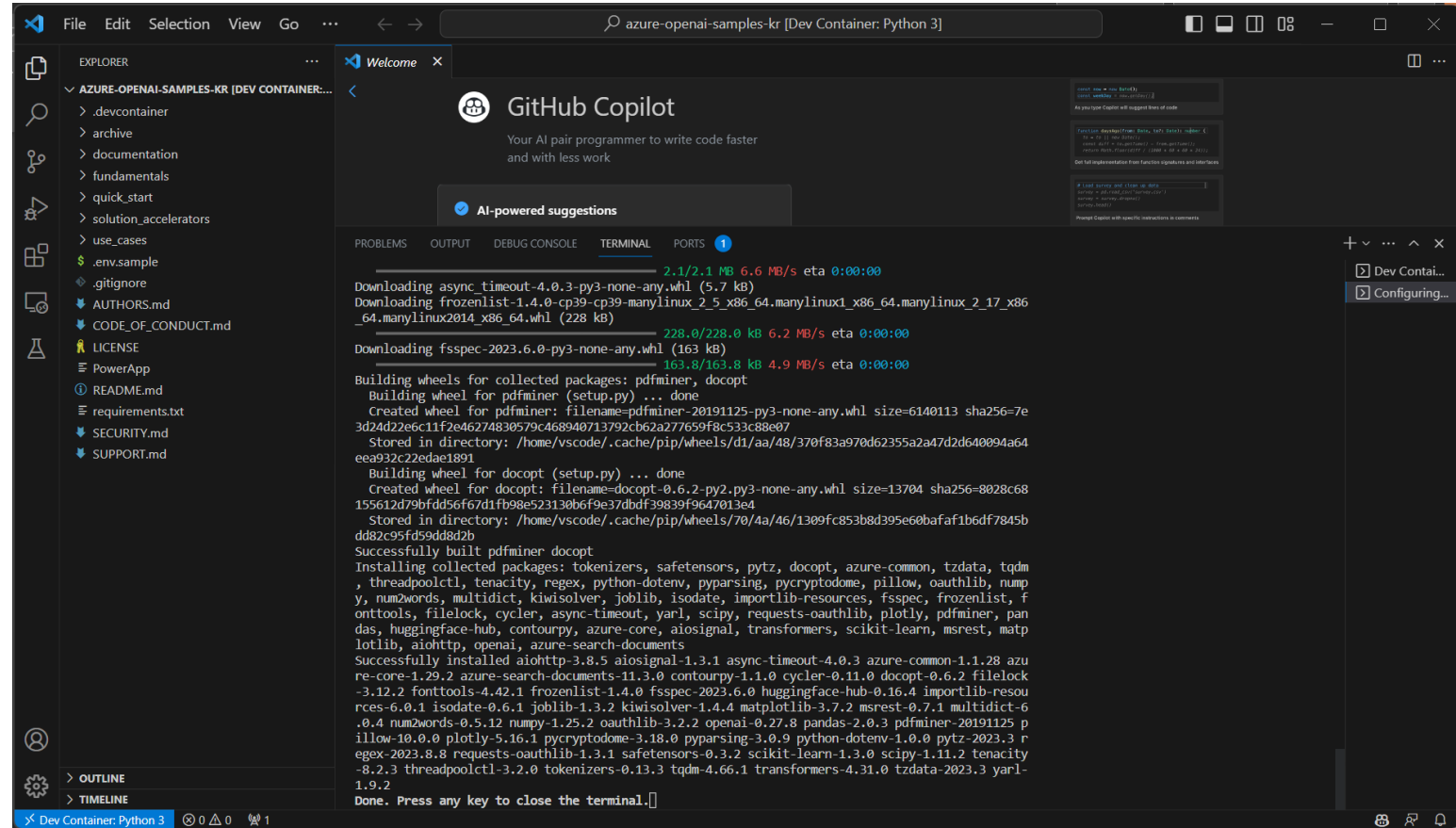
추천 방법 2: Local VS Code

1. [Docker](#) 설치
2. [Visual Studio Code](#) 설치
3. [원격 - 컨테이너 확장 설치](#)
4. Repo clone
5. Visual Studio Code에서 리포지토리 열기
6. **창의 우측 하단 모서리에 있는 "Reopen in Container" 버튼 클릭**
7. 컨테이너가 빌드되고 시작될 때까지 기다립니다.

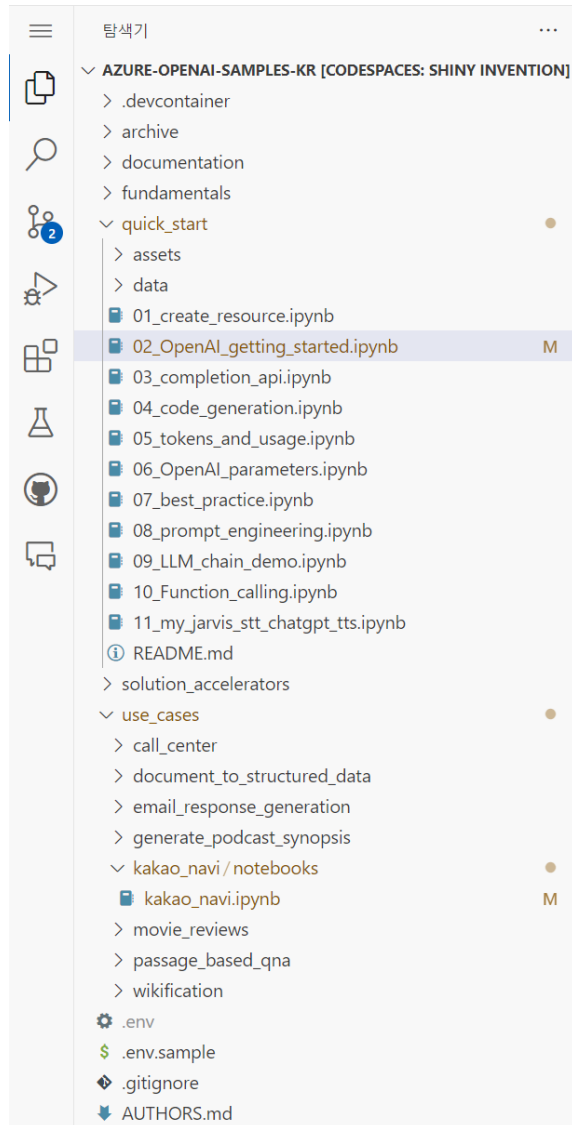


추천 방법 2: Local VS Code

1. [Docker](#) 설치
2. [Visual Studio Code](#) 설치
3. [원격 - 컨테이너 확장 설치](#)
4. Repo clone
5. Visual Studio Code에서 리포지토리 열기
6. 창의 우측 하단 모서리에 있는 "Reopen in Container" 버튼 클릭
7. **컨테이너가 빌드되고 시작될 때까지 기다립니다.**



실습 문서 소개



Self-study

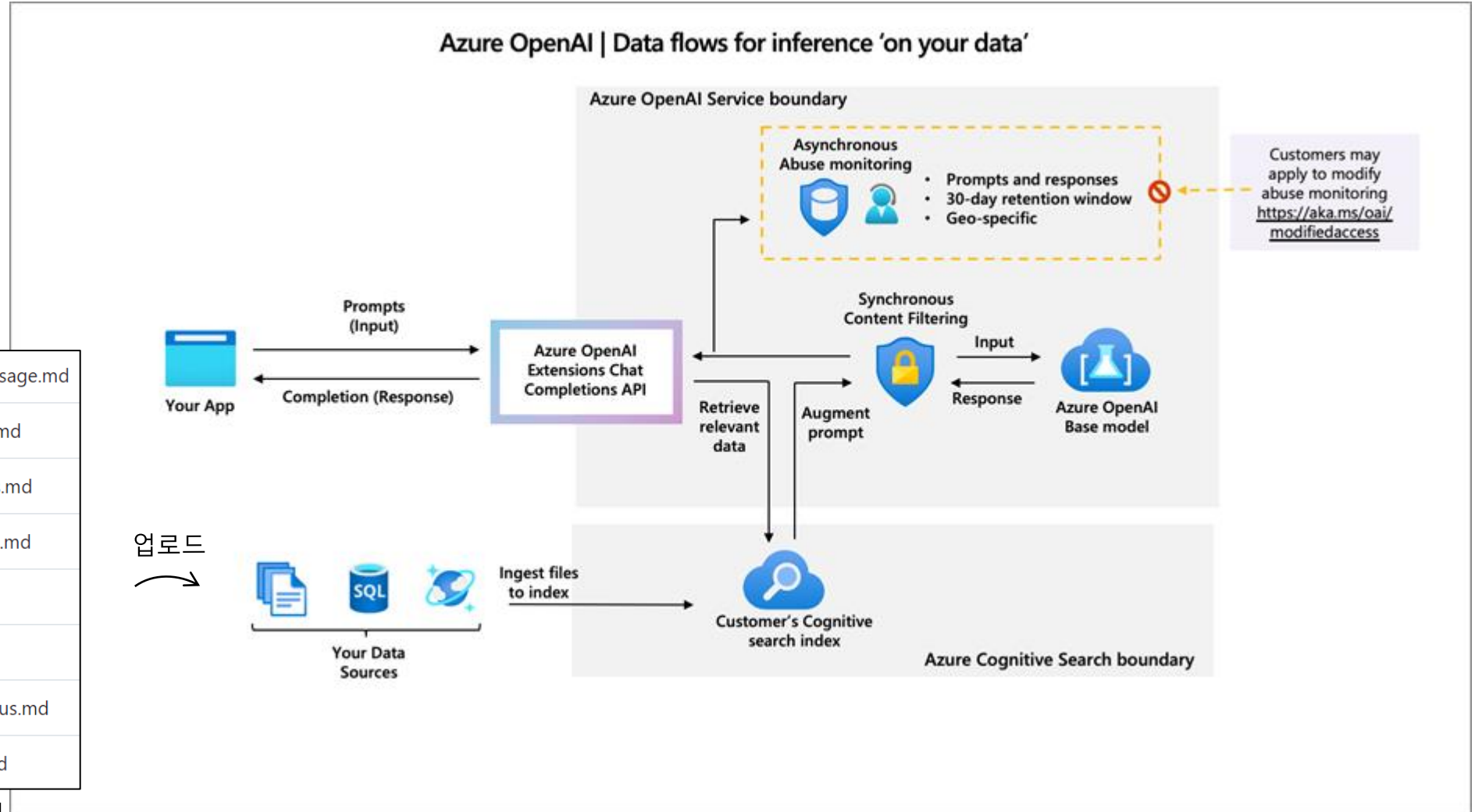
1. git clone 할 대상을 선정 (Codespace or Local)
2. quick_start/README.md 부터 시작하면서 환경 설정
3. 01_create_resource.ipynb로 시작해서 리소스 생성(있으면 생략)
4. .env 파일에 리소스 정보 등록(Open API URL, api-key, model 입력)
5. quick_start 02 ~ 08까지 실습을 진행
(오전 또는 첫째 날 진행한 Prompt Engineering 설명 복습 및 심화 과정)
6. 09는 Cognitive Search를 이용한 RAG이기에 생략 가능
또는 Azure OpenAI on your data 기능을 활용하여 테스트 합니다.
7. Fundamental 또는 Use_cases 를 진행합니다. (8월 이후)
- Function Calling 기능을 활용한 카카오맵 API 연동

나의 데이터를 쉽게 통합하여 조회하고 요약

프롬프트 엔지니어링을 위한 영어로 된 문서

- 📄 prompts-advanced-usage.md
- 📄 prompts-adversarial.md
- 📄 prompts-applications.md
- 📄 prompts-basic-usage.md
- 📄 prompts-chatgpt.md
- 📄 prompts-intro.md
- 📄 prompts-miscellaneous.md
- 📄 prompts-reliability.md

가능한 파일 형식
(txt, md, html, docx, pptx, PDF)



quick_start 구성

```

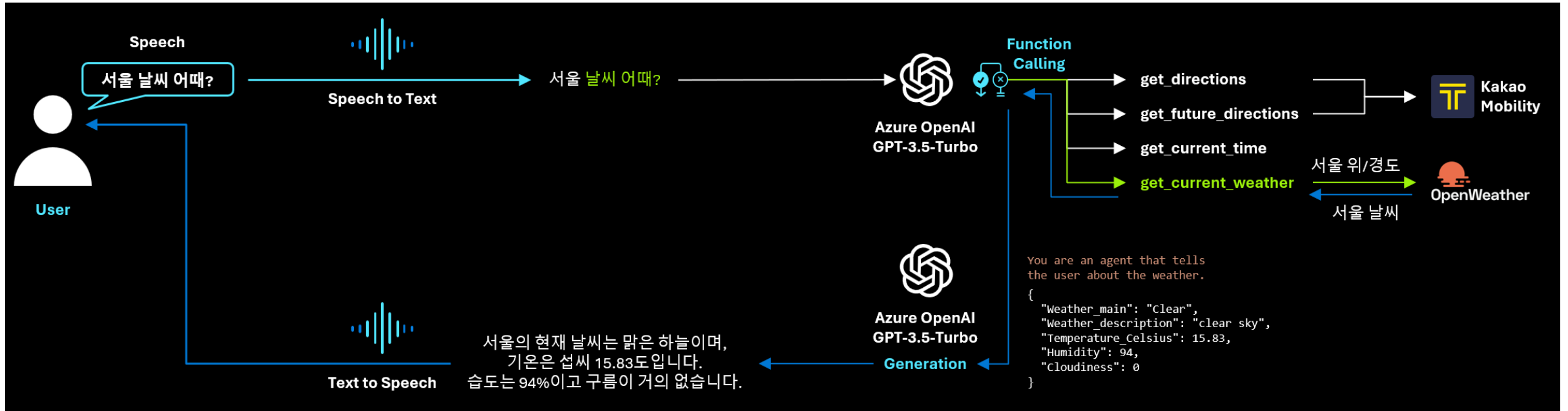
  ▾ quick_start
    > assets
    > data
    📄 01_create_resource.ipynb
    📄 02_OpenAI_getting_started.ipynb
    📄 03_completion_api.ipynb
    📄 04_code_generation.ipynb
    📄 05_tokens_and_usage.ipynb
    📄 06_OpenAI_parameters.ipynb
    📄 07_best_practice.ipynb
    📄 08_prompt_engineering.ipynb
    📄 09_LLM_chain_demo.ipynb
    📄 10_Function_calling.ipynb
    📄 11_my_jarvis_stt_chatgpt_tts.ipynb
    ⓘ README.md
    > solution_accelerators
  ▾ use_cases
    > call_center
    > document_to_structured_data
    > email_response_generation
    > generate_podcast_synopsis
  ▾ kakao_navi / notebooks
    📄 kakao_navi.ipynb

```

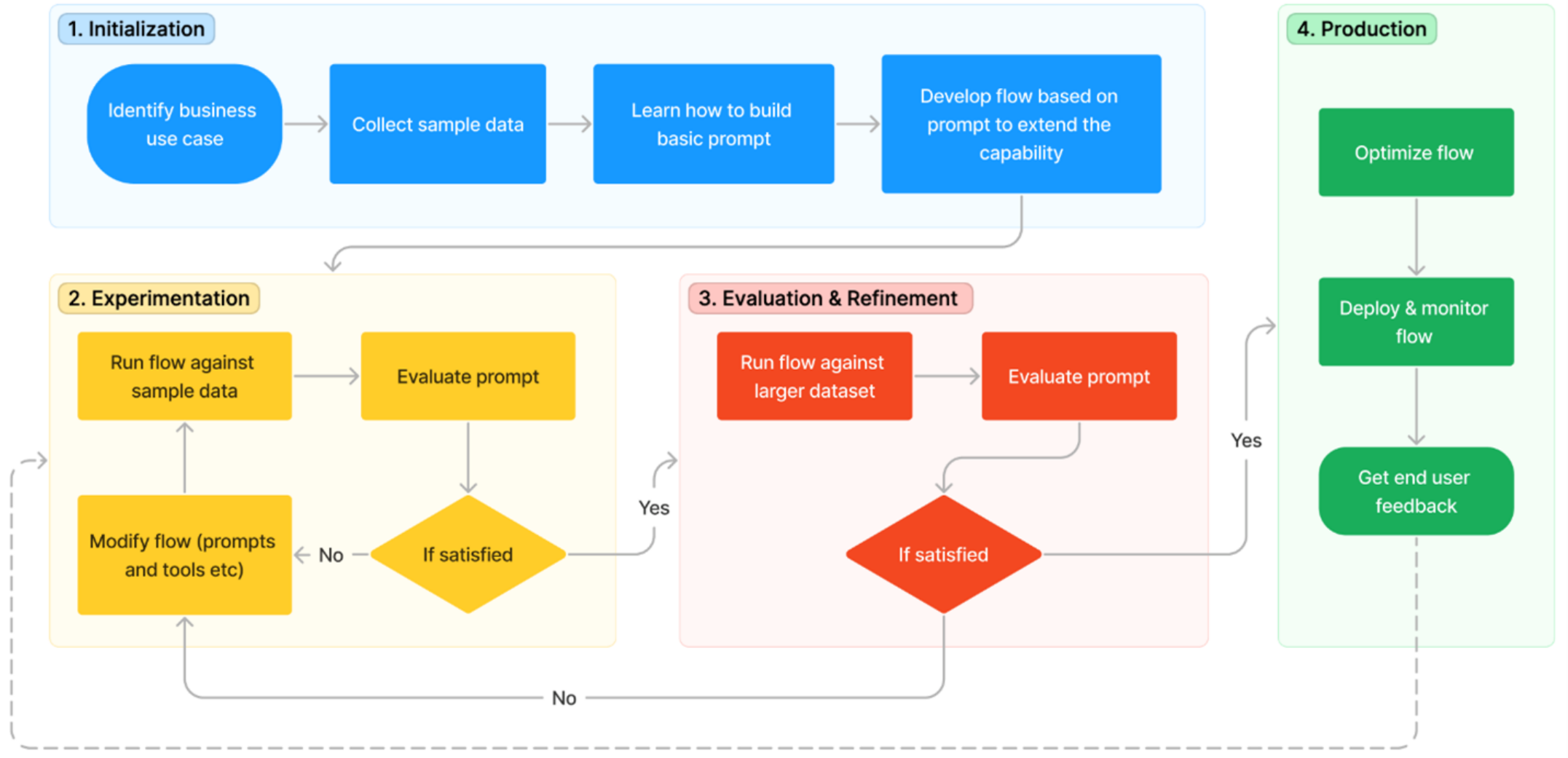
순서 소개

01. CLI를 이용한 리소스 생성
02. 기본 기능들 활용 방법
03. ChatCompletion API 활용 방법
04. SQL 코드 생성 및 코드 해석
05. API 사용에 따른 토큰 사용량 확인
06. 파라미터 사용 방법
07. 프롬프트 엔지니어링 모범 사례
08. 프롬프트 엔지니어링 예시
09. LLM 체이닝을 통한 RAG 구성
10. Function Calling에 대한 이해
11. (Local) 인공지능 비서 서비스 만들기

인공지능 비서 서비스 만들기



LLM 기반 애플리케이션 수명 주기



LLMOps - Azure Machine Learning Prompt Flow!

Microsoft Azure Machine Learning Studio

Microsoft > pf-metang > Flows > Bing Grounded QA

Bing Grounded QA Runtime: Select runtime View run history View endpoint Deploy Clone Save Bulk test Run

Search more tools + LLM + Prompt + Python

offset: int 0

process_search_result

Name	Type	Value
search_result	object	`\${search_on_bing.output}`

```
1 from promptflow import tool
2
3
4 @tool
5 def process_search_result(search_result):
6
7     def format(doc: dict):
8         return f"Content: {doc['Content']}\nSource: {doc['Source']}"
```

augmented_qna

Connection: Connection

deployment_name: text-davinci-001 max_tokens: 256 temperature: 0.5 stop:

Advanced

Inputs

Flow Diagram: inputs → extract_query_from_quest... → search_on_bing → process_search_result → augmented_qna → outputs



Thank you