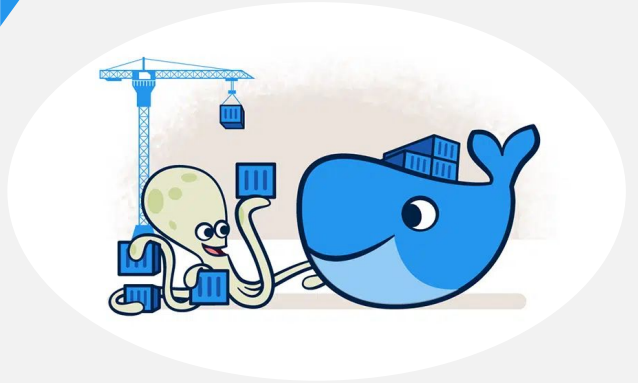


Boost Your Developer Productivity with Docker AI

Docker DevTools Day

March 16, 2024



Speaker

- Developer Advocate at Docker
- Author @ <https://collabnix.com>
- Former Docker Captain
- Docker Community Leader, Bangalore
- Distinguished ARM Ambassador
- Worked @ Dell EMC, VMware, Redis & CGI

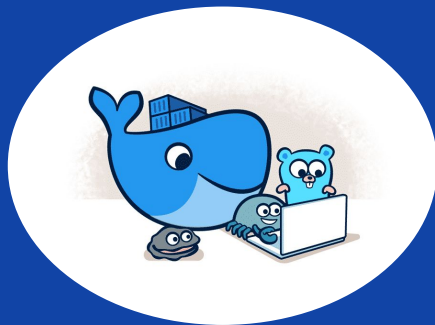


Agenda

1. Inner-loop Development Workflow
2. Image Deep-Dive and Best Practices
3. Introduction to Docker AI
4. What problems does Docker AI solve?
5. Getting Started

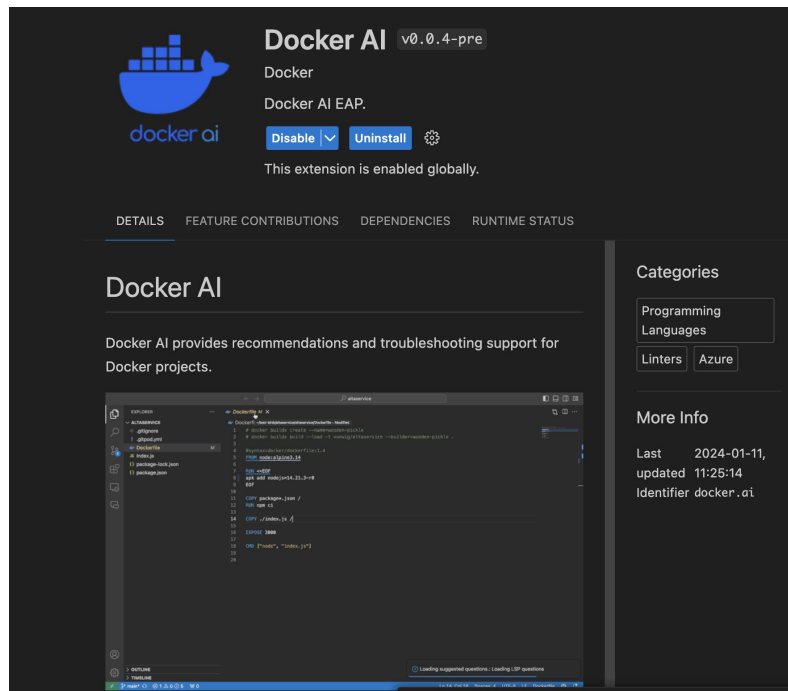


Introduction to Docker AI




What is Docker AI?

- An Extension for Visual Studio Code
- Provides runnable terminals inside of notebooks
- Integrates AI recommendations to assist with debugging and improving your Docker projects
- Provides recommendations and troubleshooting support for Docker projects
- Docker AI offers **context-specific guidance**: It doesn't just suggest generic best practices. Instead, it analyzes the specific context of your Docker workflow and provides targeted recommendations based on community knowledge and AI insights.



What is Docker AI?




Docker AI

v0.0.4-pre

Docker

Docker AI EAP.

Disable Uninstall 


This extension is enabled globally.

DETAILSFEATURESDependencies

Runtime Status
Activation Events
Commands
Notebook Rendering
Notebooks
Programming Language
Settings

Notebooks

| ID | Name |
|-----------------|-----------------|
| docker-notebook | Docker Notebook |

 Loading suggested questions.: Loading LSP questions

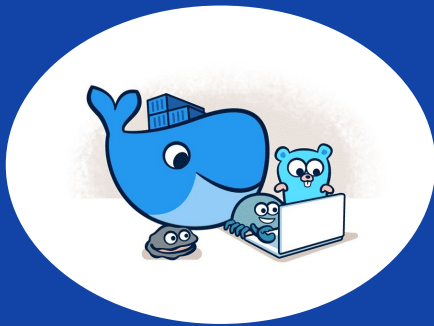


Please note that Docker AI is currently in the early access phase, during which we welcome your active participation.





Getting Started



Getting Started

VSCode Extension Setup & Installation Installation Instructions

Step 1: Download Extension File

Download the latest .vsix file [here](#).

Step 2: Enable Extension in VSCode

Option 1: Via the command line

Must have VSCode code command in PATH

```
code --install-extension <PATH_TO_DOWNLOADED_.VSIX> --force
```

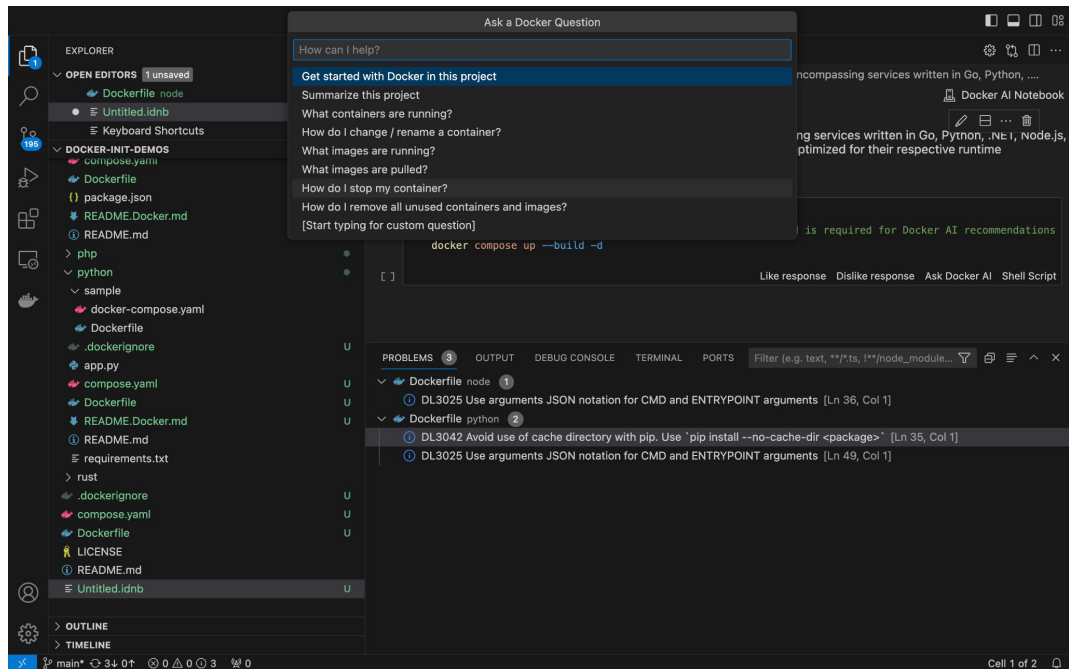
If you don't have the `code` command, you can [enable it in VSCode](#) or skip to Option 2 below.

Option 2: Via the command palette in VSCode

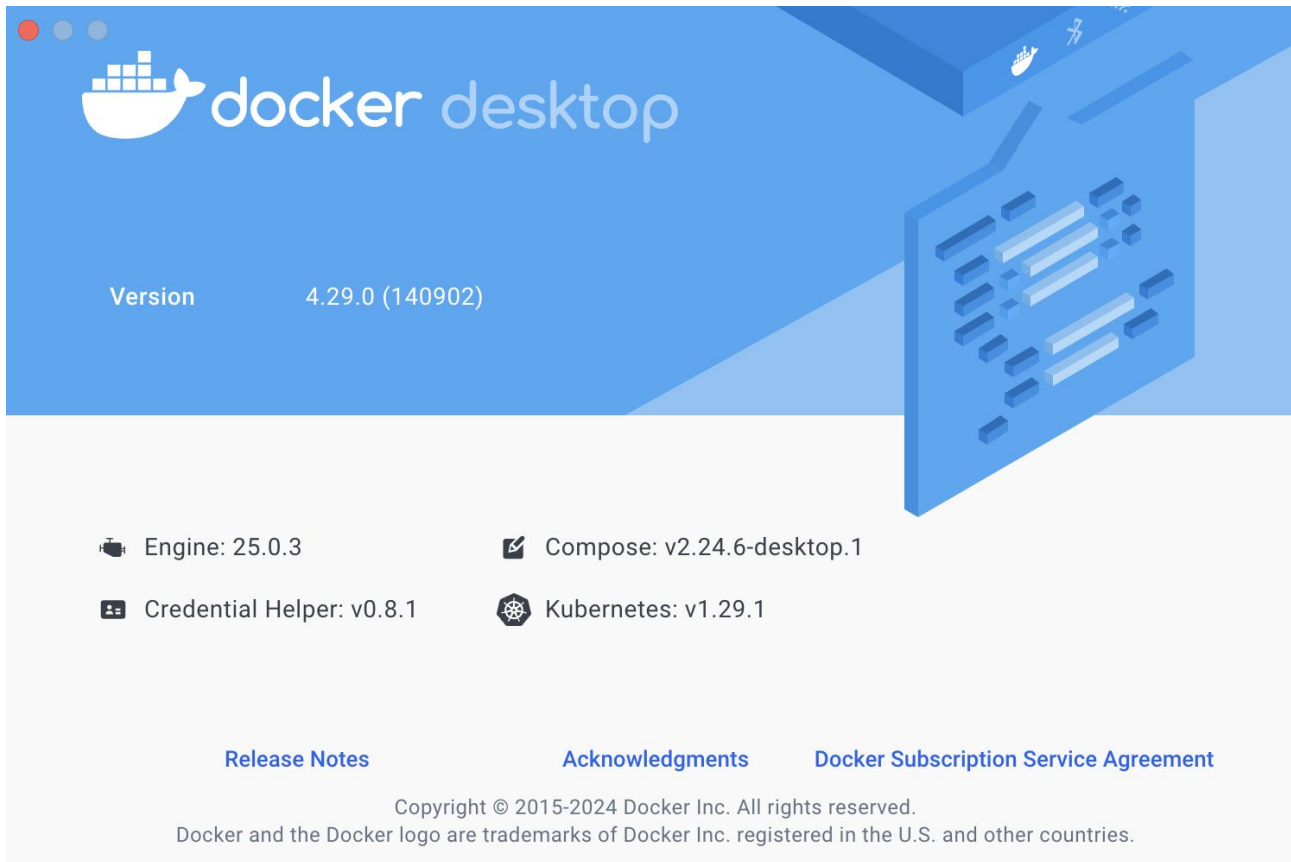
1. Type > into top search bar
 - a. Shortcut: Press **⇧⌘P** (OSX) or **CTRL⇧P** (Win/Linux)
2. Use command **Extensions: Install from VSIX...**
3. Select .VSIX file downloaded in Step 1
4. Hit the "Reload Now" button in the notification at the bottom right corner

Step 3: Open the Command Palette and select Ask Docker AI for a list of questions to get started

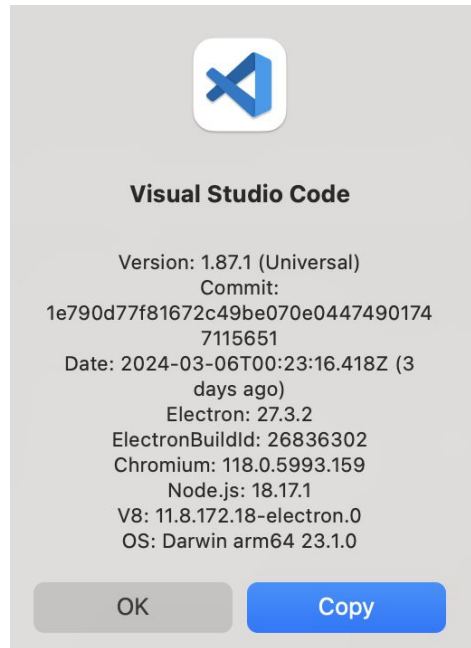
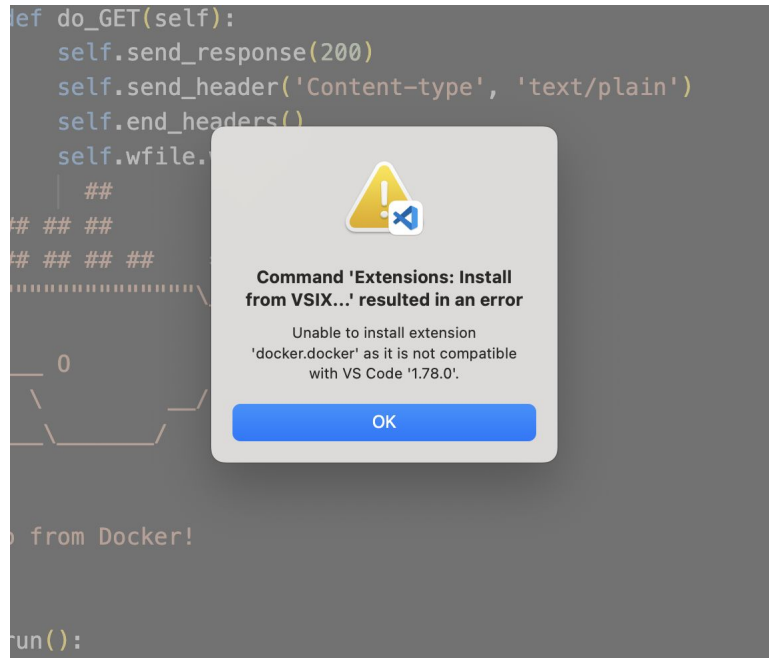
Step 4: Please install and have Docker Desktop running



Prereq: Install Docker Desktop

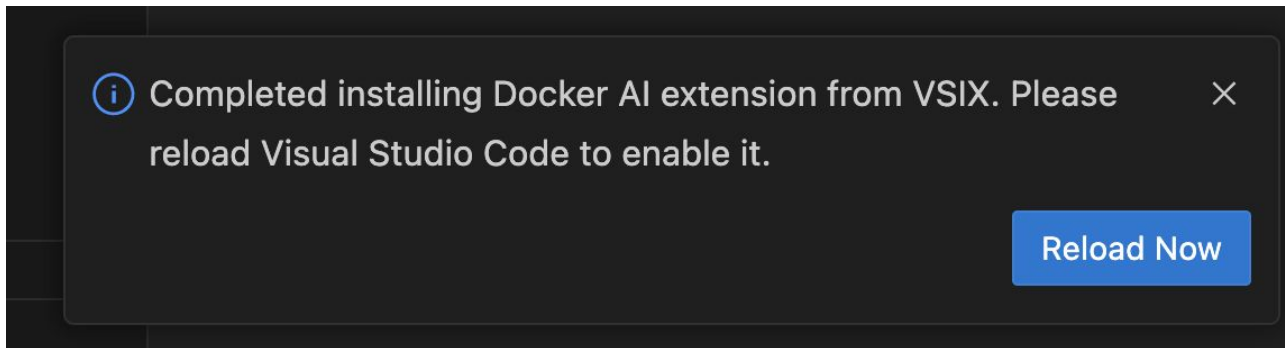


1. Install the latest version of Visual Studio Code



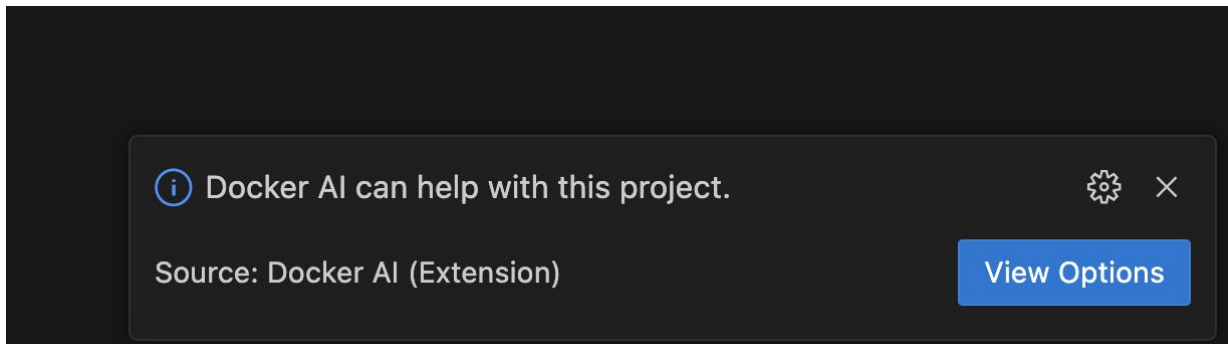
2. Download the Pre-Release Extension File

<https://github.com/docker/docker-ai/releases>

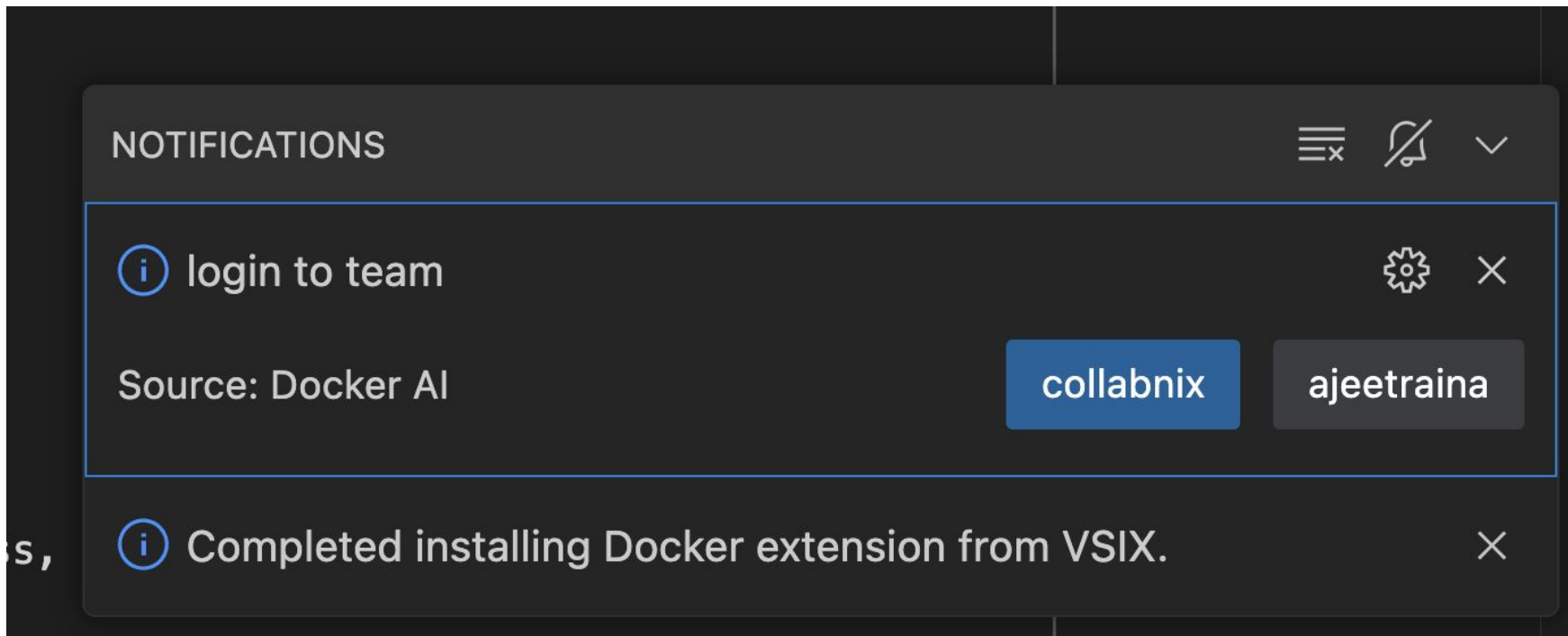


3. Enable Extension in VS Code

<https://github.com/docker/docker-ai/releases>



4. Login to Docker Hub/Org



4. Start Asking Docker AI

Ask a Docker Question

How can I help?

Get started with Docker in this project

Summarize this project

What containers are running?

How do I change / rename a container?

What images are running?

What images are pulled?

How do I stop my container?

How do I remove all unused containers and images?

[Start typing for custom question]



4. Clone the repository

<https://github.com/dockersamples/docker-init-demos>



5. Ask: Summarise this project

Untitled.idnb U ●

Untitled.idnb > M+This project represents a Python-based application containerized with Docker, utilizing multi-stage bu...

+ Code

+ Markdown

Run All

Clear All Outputs

...

Docker AI Notebook

Add Markdown Cell

+ Code

+ Markdown

...

This project represents a Python-based application containerized with Docker, utilizing multi-stage builds and non-root users for security as well as Docker Compose for orchestrating the container. The tech stack also involves optimizing Docker builds through cache mounts and adheres to Docker's best practices for managing dependencies and running services.

```
# If you want to containerize your application
# --build forces a fresh build; -d runs in detached mode and is required for
Docker AI recommendations
docker compose up --build -d
```

[]

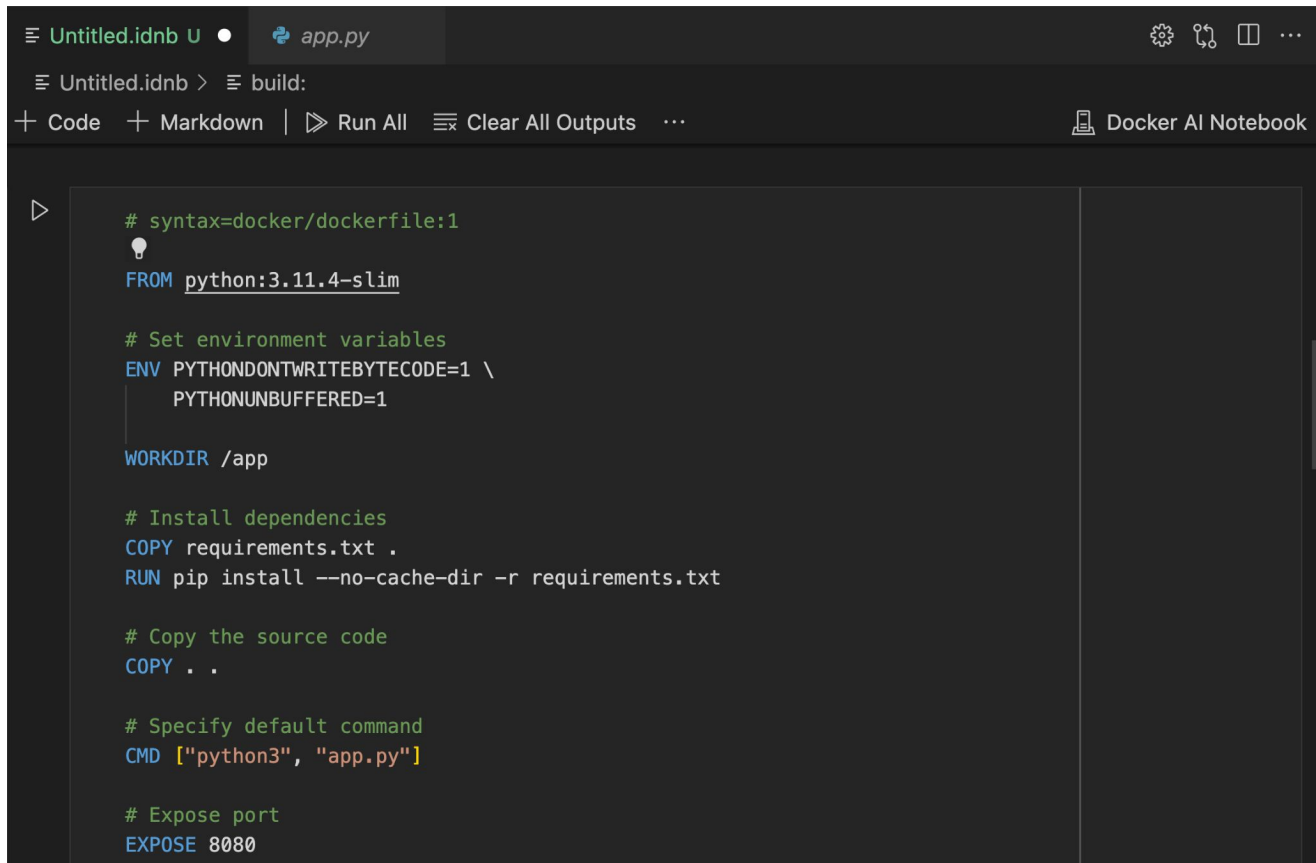
Like response

Dislike response

Ask Docker AI

Shell Script

Ask: Can you write a new Dockerfile?



The screenshot shows a web-based code editor for Dockerfiles. The interface includes a top bar with tabs for 'Untitled.idnb U' and 'app.py', and a right sidebar with icons for settings, search, and a 'Docker AI Notebook' button. The main editor area contains a Dockerfile with the following content:

```
# syntax=docker/dockerfile:1
FROM python:3.11.4-slim

# Set environment variables
ENV PYTHONDONTWRITEBYTECODE=1 \
    PYTHONUNBUFFERED=1

WORKDIR /app

# Install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy the source code
COPY . .

# Specify default command
CMD ["python3", "app.py"]

# Expose port
EXPOSE 8080
```



Docker AI creates and run 2 containers

Containers [Give feedback](#)

Container CPU usage ⓘ

0.00% / 800% (8 CPUs available)

Container memory usage ⓘ

0B / 7.48GB

[Show charts](#)



Only show running containers



Name ↑

Image

Status

CPU (%)

Port(s)

Last

Actions



[inspiring_dirac](#)

a4734294bf6b

[vonwig/lsp](#) Running

0%

6 m



[xenodochial_wescoff](#)

882e925a6632

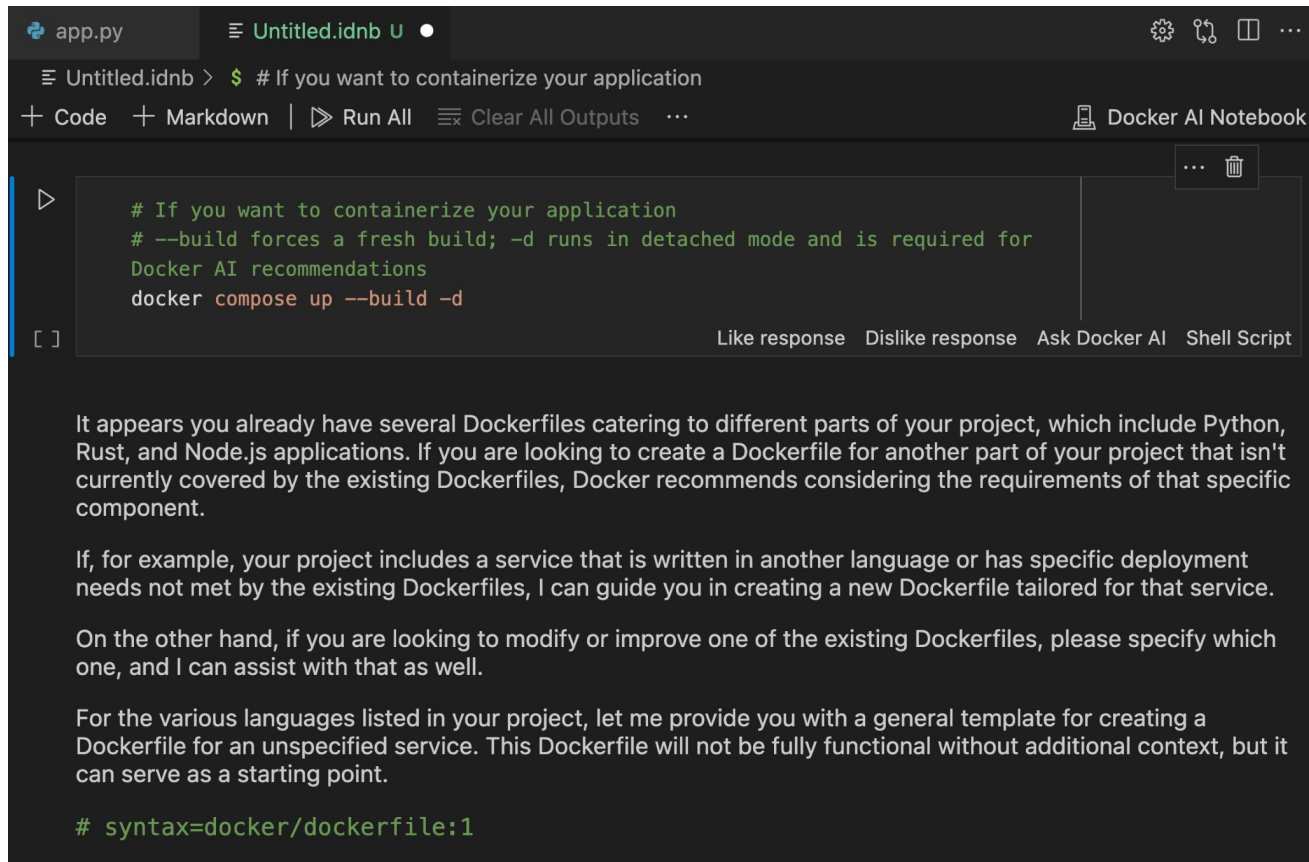
[vonwig/lab](#) Running

0%

6 m



5. Que: Can you write a Dockerfile?



The screenshot shows the Docker AI Notebook interface. At the top, there's a tab labeled 'app.py' and another labeled 'Untitled.idnb'. Below the tabs, there's a prompt bar with the text: 'Untitled.idnb > \$ # If you want to containerize your application'. To the right of the prompt bar, there are icons for settings, a refresh button, a window icon, and a menu icon. Below the prompt bar, there's a toolbar with buttons for '+ Code', '+ Markdown', 'Run All', 'Clear All Outputs', and a menu icon. To the right of the toolbar, there's a button labeled 'Docker AI Notebook'. The main area of the interface is a code editor with a dark background. It contains the following text:

```
# If you want to containerize your application
# --build forces a fresh build; -d runs in detached mode and is required for
Docker AI recommendations
docker compose up --build -d
```

 To the right of the code editor, there's a button with three dots and a trash icon. Below the code editor, there's a row of buttons: 'Like response', 'Dislike response', 'Ask Docker AI', and 'Shell Script'. Below this row, there's a text block that reads:

It appears you already have several Dockerfiles catering to different parts of your project, which include Python, Rust, and Node.js applications. If you are looking to create a Dockerfile for another part of your project that isn't currently covered by the existing Dockerfiles, Docker recommends considering the requirements of that specific component.

If, for example, your project includes a service that is written in another language or has specific deployment needs not met by the existing Dockerfiles, I can guide you in creating a new Dockerfile tailored for that service.

On the other hand, if you are looking to modify or improve one of the existing Dockerfiles, please specify which one, and I can assist with that as well.

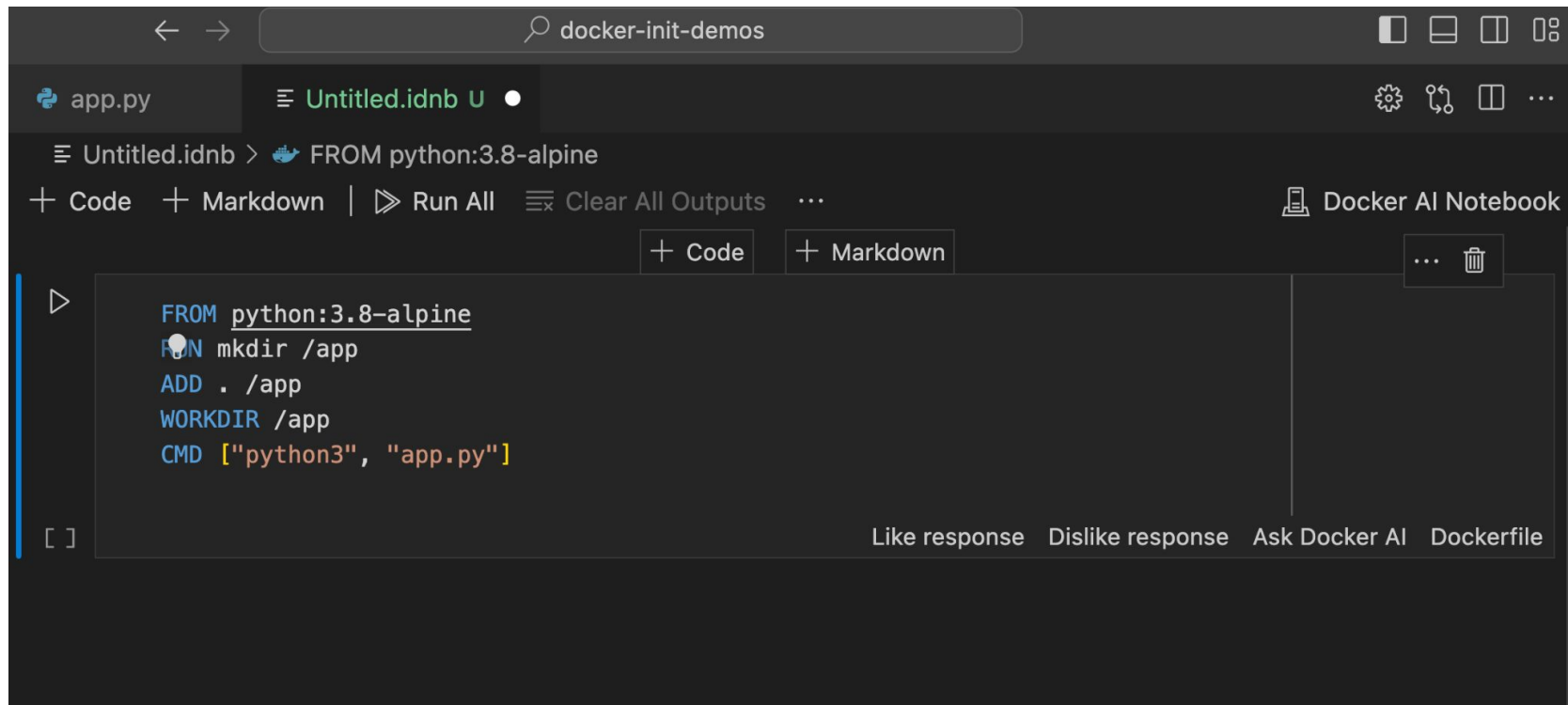
For the various languages listed in your project, let me provide you with a general template for creating a Dockerfile for an unspecified service. This Dockerfile will not be fully functional without additional context, but it can serve as a starting point.

 At the bottom of the interface, there's a text block that reads:

```
# syntax=docker/dockerfile:1
```



6. Ask: Can you write a Dockerfile?



The screenshot shows the Docker AI Notebook interface. At the top, there's a search bar with "docker-init-demos" and a sidebar with "app.py" and "Untitled.idnb". The main area displays a Dockerfile snippet:

```
FROM python:3.8-alpine
RUN mkdir /app
ADD . /app
WORKDIR /app
CMD ["python3", "app.py"]
```

Below the code, there are buttons for "Like response", "Dislike response", "Ask Docker AI", and "Dockerfile".



7. Ask: Can you write a Docker Compose file?

```
services:
  app:
    build: .
    ports:
      - "8080:8080"
    command: python3 app.py
```

Like response

Dislike response

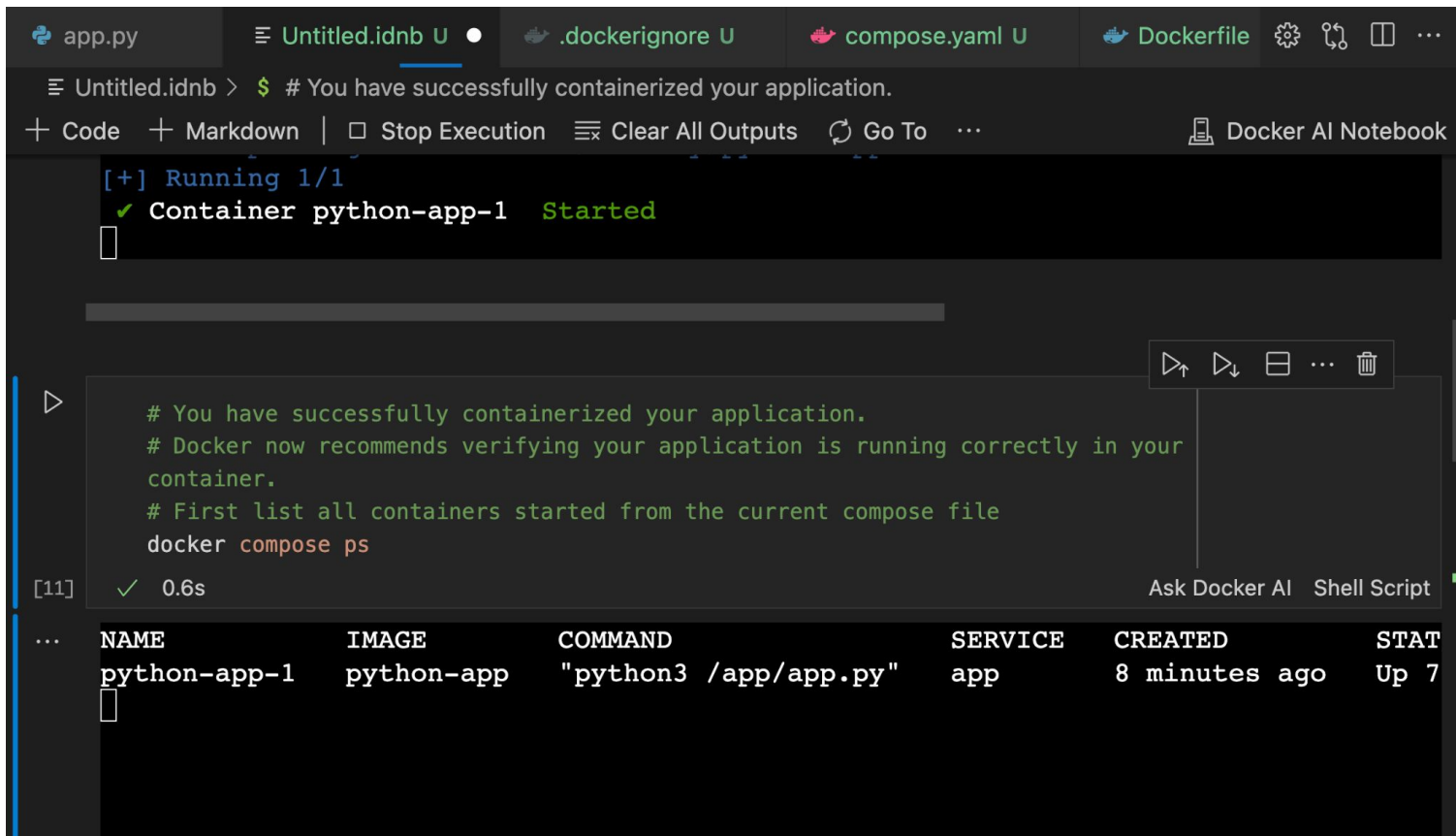
Ask Docker AI



docker-compose



8. Ask: What images are running?



The screenshot displays the Docker AI Notebook interface. At the top, there are tabs for 'app.py', 'Untitled.idnb U', '.dockerignore U', 'compose.yaml U', and 'Dockerfile'. Below the tabs, a message states: 'You have successfully containerized your application.' The interface includes a toolbar with options like 'Code', 'Markdown', 'Stop Execution', 'Clear All Outputs', and 'Go To'. A section titled 'Running 1/1' shows a green checkmark and the text 'Container python-app-1 Started'. Below this, a code editor shows the command 'docker compose ps' being executed. The output of the command is displayed in a table format.

```
[+] Running 1/1
✓ Container python-app-1 Started

# You have successfully containerized your application.
# Docker now recommends verifying your application is running correctly in your
container.
# First list all containers started from the current compose file
docker compose ps
```

[11] ✓ 0.6s

| NAME | IMAGE | COMMAND | SERVICE | CREATED | STAT |
|--------------|------------|-----------------------|---------|---------------|------|
| python-app-1 | python-app | "python3 /app/app.py" | app | 8 minutes ago | Up 7 |



10. Ask: How do I stop the container?

```
# Because you are in a compose project, use compose commands to manage your containers
docker compose down
```

[12] ↺ 9.8s

Like response Dislike response Ask Docker AI Shell Scrip

```
[+] Running 0/1
  ... Container python-app-1 Stopping
```



11. Ask: What images are pulled?

▶

```
# Docker AI detected 5 images in your environment
# Manually check all your images for more information
docker images
```

[13]

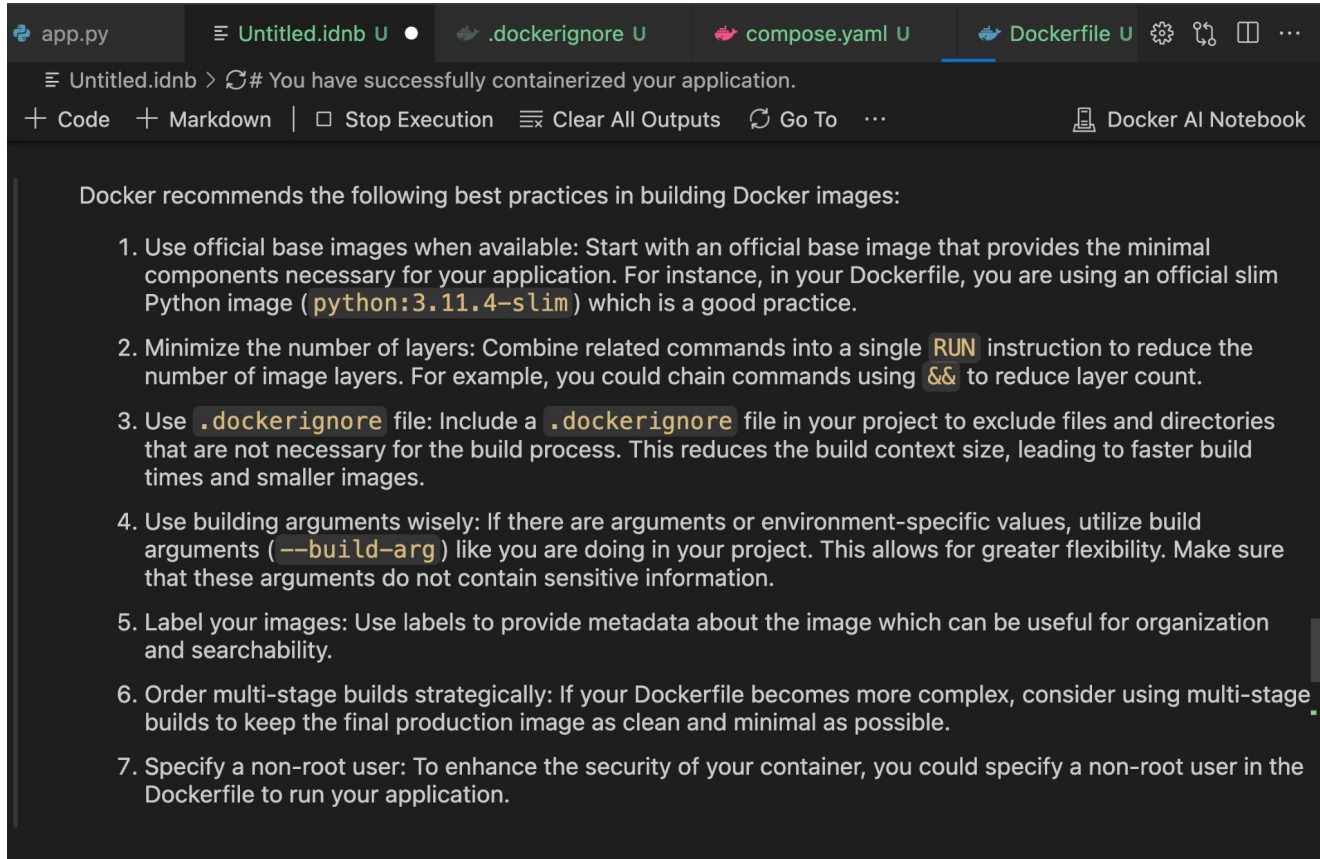
✓ 0.5s

Ask Docker AI Shell Script

| ... | REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|-----|--------------------------|--------|--------------|----------------|-------|
| | python-app | latest | ef59b3bd1f19 | 22 minutes ago | 265MB |
| | docker/labs-assistant-ml | latest | d4709f4b7bf5 | 4 weeks ago | 682MB |
| | docker/lsp | latest | 247c241aa68b | 6 weeks ago | 277MB |
| | vonwig/lsp | latest | 4151d0f1ec26 | 2 months ago | 276MB |
| | vonwig/labs-assistant-ml | latest | f406187b7b87 | 2 months ago | 354MB |



12. Ask: What are the best practices to build a Docker Image?



The screenshot shows a web-based interface for a Docker AI Notebook. At the top, there's a navigation bar with tabs for 'app.py', 'Untitled.idnb', '.dockerignore', 'compose.yaml', and 'Dockerfile'. Below the tabs, a status bar indicates 'Untitled.idnb > # You have successfully containerized your application.' and a toolbar with options like '+ Code', '+ Markdown', 'Stop Execution', 'Clear All Outputs', 'Go To', and a 'Docker AI Notebook' button. The main content area displays a list of seven best practices for building Docker images, each numbered and preceded by a small icon.

Docker recommends the following best practices in building Docker images:

1. Use official base images when available: Start with an official base image that provides the minimal components necessary for your application. For instance, in your Dockerfile, you are using an official slim Python image (`python:3.11.4-slim`) which is a good practice.
2. Minimize the number of layers: Combine related commands into a single `RUN` instruction to reduce the number of image layers. For example, you could chain commands using `&&` to reduce layer count.
3. Use `.dockerignore` file: Include a `.dockerignore` file in your project to exclude files and directories that are not necessary for the build process. This reduces the build context size, leading to faster build times and smaller images.
4. Use building arguments wisely: If there are arguments or environment-specific values, utilize build arguments (`--build-arg`) like you are doing in your project. This allows for greater flexibility. Make sure that these arguments do not contain sensitive information.
5. Label your images: Use labels to provide metadata about the image which can be useful for organization and searchability.
6. Order multi-stage builds strategically: If your Dockerfile becomes more complex, consider using multi-stage builds to keep the final production image as clean and minimal as possible.
7. Specify a non-root user: To enhance the security of your container, you could specify a non-root user in the Dockerfile to run your application.

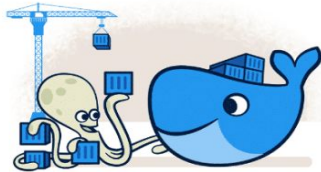


Community-driven Projects



List of major projects contributed by the community contributors

<https://dockerlabs.collabnix.com>



Docker Labs

With 4,500 GitHub stars, Dockerlabs is an initiative within the Collabnix community that focuses specifically on Docker-related learning, training, and hands-on labs.

<https://kubetools.collabnix.com>



KubeTools

With over 1,100 GitHub stars, Kubetools is a curated list of over 350+ Kubernetes tools and utilities targeted at DevOps Engineers

<https://kubelabs.collabnix.com>



KubeLabs

Kubelabs just crossed 2.2k GitHub stars is an initiative within the Collabnix community that focuses on Kubernetes-related learning, training, and hands-on labs.

Community-driven projects



<https://dockerextensions.collabnix.com>

<https://github.com/collabnix/docker-scout-community>





Thank You

