Docker + Git Bash Notes (Step-by-Step with Q&A)
Starter Guide: Running Docker from Git Bash

Step 1: Install Docker Desktop

Action: Download and install Docker Desktop for Windows.

Purpose: Docker Desktop is the engine that runs containers. Git Bash is just the shell you'll use to talk to Docker.

Step 2: Open Git Bash

Action: Launch Git Bash from the Start menu.

Purpose: Git Bash gives you a Linux-like terminal on Windows, closer to industry environments.

Step 3: Check Docker

Action: Run:

bash

```
docker version
```

Purpose: Confirms Docker is installed and accessible from Git Bash.

Step 4: Run a Test Container

Action: Run:

bash

```
docker run hello-world
```

Purpose: Downloads a test image and runs it. If you see "Hello from Docker!", setup is working.

Step 5: Fix Path Issues

Action: Use POSIX-style paths when mounting volumes:

bash

```
docker run --rm -v /c/Users/Sourav:/data busybox ls /data
```

Purpose: Git Bash uses /c/... instead of C:\.... This ensures Docker can access your files.

Step 6: Handle Interactive Containers

Action: Use winpty for interactive shells:

bash

```
winpty docker run -it ubuntu bash
```

Purpose: Fixes Git Bash's terminal quirks so you can type inside containers.

Step 7: Practice with a Simple Project

Action: Create a folder, add a file, mount it into a container:

bash

```
mkdir -p /c/Users/Sourav/docker-testcd /c/Users/Sourav/docker-testecho "Hello Sourav" > hello.txtdocker run --rm -v "$PWD:/app" busybox cat /app/hello.txt
```

Purpose: Shows how containers can read files from your host.

Step 8: Build Your First Image

Action: Write a Dockerfile in VS Code:

dockerfile

```
FROM alpine:3.20WORKDIR /appCOPY hello.txt .CMD ["cat", "hello.txt"]
```

Build and run:

bash

```
docker build -t sourav-test .docker run --rm sourav-test
```

Purpose: Teaches you how to package code into an image — the foundation of industry deployments.

?Questions & Answers
Q1: Do I need to start from Step 5 if Docker is already working?

Answer: No. Since docker version shows client/server info and you've run a container successfully, Steps 1–4 are complete. You can jump straight to Step 5 onward (mounts, interactive sessions, building images).

Q2: Why is winpty required?
Answer:

Git Bash doesn't provide a proper interactive terminal (TTY).

When you run interactive containers (docker run -it ubuntu bash), input/output may break.

winpty acts as a bridge, fixing TTY handling so you can type inside containers normally.

Needed only for interactive sessions (like bash, mysql, psql), not for simple commands.

Q3: What does docker run -it ubuntu bash mean?
Answer:

docker run → Start a new container.

-it → Interactive + TTY (lets you type inside).

ubuntu → Use the Ubuntu image.

bash → Run the Bash shell inside the container. Result: You get dropped into an Ubuntu shell inside the container, where you can run Linux commands.

Q4: Why did you use echo to create a Dockerfile? I usually use VS Code.
Answer:

The echo method was just a quick demo to create a file from Git Bash.

In real projects, engineers use VS Code to write and store Dockerfiles alongside their code.

You can open your project folder in VS Code (code .), create a Dockerfile, and edit it normally.

Then build/run from Git Bash — exactly like industry workflows.

Q5: Why use docker run --rm?
Answer:

By default, stopped containers remain on your system.

--rm tells Docker to auto-remove the container after it exits, preventing clutter.

Useful for testing and one-off commands.

Not used for long-running apps where you want logs or to restart containers later.

Q6: What does docker run --rm -v /c/Users/Sourav:/data busybox ls /data mean?
Answer:

docker run → Start a container.

--rm → Remove it after exit.

-v /c/Users/Sourav:/data → Mount your Windows folder into the container at /data.

busybox → Use the BusyBox image (tiny Linux with basic commands).

ls /data → Run ls inside the container to list files in /data. Result: You see the contents of your Windows folder inside the container. Purpose: Demonstrates how containers access host files — critical in industry for configs, logs, and data.