# Gemini Chatbot Deployment – Step-by-Step Guide

## Project Goal

Deploy a Streamlit-based chatbot using the Google Gemini API, containerized via Docker, and hosted inside an Ubuntu VM (VirtualBox) on your own machine.

## Technology Stack

* Streamlit (Python frontend)
* Google Generative AI SDK (Gemini API)
* Docker (containerization)
* VirtualBox (on-premise VM environment)
* Ubuntu 24 (guest OS)

## Project Structure

chatbot\_project/  
├── app.py  
├── requirements.txt # freeze requirements  
├── Dockerfile  
└── gemini\_chatbot.tar (Generated after build)

## Step 1: Project Setup on Windows Host

### 1. app.py

import streamlit as st  
import google.generativeai as genai  
import os  
  
genai.configure(api\_key=os.getenv("GEMINI\_API\_KEY"))  
model = genai.GenerativeModel('gemini-pro') # choose your model  
  
st.set\_page\_config(page\_title="Gemini Chatbot", page\_icon="🤖")  
st.title("🤖 Gemini Chatbot")  
  
if "messages" not in st.session\_state:  
 st.session\_state.messages = []  
  
for msg in st.session\_state.messages:  
 with st.chat\_message(msg["role"]):  
 st.markdown(msg["content"])  
  
prompt = st.chat\_input("Ask me anything...")  
  
if prompt:  
 st.session\_state.messages.append({"role": "user", "content": prompt})  
 with st.chat\_message("user"):  
 st.markdown(prompt)  
  
 response = model.generate\_content(prompt)  
 reply = response.text  
  
 st.session\_state.messages.append({"role": "assistant", "content": reply})  
 with st.chat\_message("assistant"):  
 st.markdown(reply)

### 2. requirements.txt

streamlit  
google-generativeai

### 3. Dockerfile

FROM python:3.11  
  
WORKDIR /app  
  
COPY requirements.txt .  
RUN pip install --no-cache-dir -r requirements.txt  
  
COPY app.py .  
  
EXPOSE 8501  
  
CMD ["streamlit", "run", "app.py", "--server.port=8501", "--server.address=0.0.0.0"]

### 4. Local Build and Test

python -m venv .venv  
.venv\Scripts\activate  
pip install -r requirements.txt  
streamlit run app.py

Build Docker Image:

docker build -t gemini-chatbot .

Save Docker Image:

docker save gemini-chatbot > gemini\_chatbot.tar

## Step 2: Transfer Docker Image to Ubuntu VM

### Ubuntu VM Setup

* Use Bridged Adapter networking mode.
* Confirm VM IP:

ip addr

* Enable SSH:

sudo apt update  
sudo apt install openssh-server  
sudo systemctl start ssh  
sudo systemctl enable ssh

### Transfer Docker Image via SCP

scp gemini\_chatbot.tar admin@<VM\_IP>:/home/admin/

Example:

scp gemini\_chatbot.tar admin@192.168.249.198:/home/admin/

## Step 3: Load & Run Docker Image on Ubuntu VM

Inside Ubuntu VM:

Load Docker Image:

cd /home/admin/  
docker load < gemini\_chatbot.tar  
docker images

Run Container:

docker run -e GEMINI\_API\_KEY="YOUR\_ACTUAL\_API\_KEY" -p 8501:8501 gemini-chatbot

Access chatbot in VM browser:

http://localhost:8501

Or from LAN (if applicable):

http://<VM\_IP>:8501

Optional - Run in Background:

docker run -d -e GEMINI\_API\_KEY="YOUR\_ACTUAL\_API\_KEY" -p 8501:8501 gemini-chatbot  
docker ps

## Notes

* Ensure valid Gemini API key.
* Use firewall (ufw) for port control.
* Optional: Configure Docker Compose, logging, and auto-restart for production.

## Conclusion

Your Gemini Chatbot is now: - Dockerized - Portable via Docker image - Running inside your Ubuntu VM

Ready for production or further enhancements.