Decentralized RAG: Combining Retrieval Augmented Generation with Blockchains

Reed White (12367576)

# System Manual

This system is available in two flavors: one that stores all corpus text fully in the blockchain, and one that only stores hashes on-chain with all text stored off-chain. The setup and administration for each are simple, but the off-chain version requires extra setup. Details for each are outlined below.

Please note that this document will not delve into setting up a blockchain. An existing blockchain is required for this system to function. It is intended to be used with internal, private blockchains. It was tested using a local Hardhat environment.

## Node Installation

Node is required to run the provided Corpus Administrator Frontend (and Hardhat, if desired). Node can be installed in the /src/ directory on a Linux server with:

curl -fsSL https://fnm.vercel.app/install | bash

source ~/.bashrc

fnm use --install-if-missing 22

More details and options (for installation into other architectures) can be found here: <https://nodejs.org/en/download/package-manager>.

## Hardhat Installation

For development purposes, Hardhat can be installed to simulate a private blockchain. Hardat can be installed in /src/hardhat directory with:

npm install hardhat

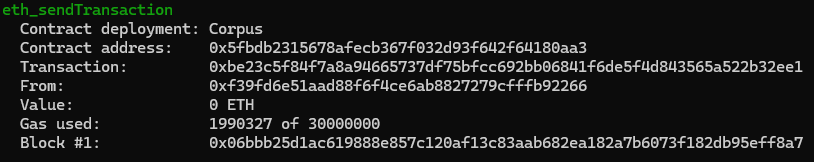
npm install '@nomicfoundation/hardhat-toolbox'

npx hardhat node

More details can be found in: <https://hardhat.org/hardhat-runner/docs/getting-started>.

## On-Chain Corpus

Deploying the on-chain version of this system is relatively straightforward. The following steps are required:

1. Deploy the Corpus smart contract in DocumentStore.sol contract located in /src/hardhat/contracts to the blockchain.
   1. This was done locally with the following command in the /src/hardhat directory:  
        
      npx hardhat run scripts/deploy.js --network localhost
   2. Take note of the deployed smart contract address. This can be retrieved from the block in the target blockchain or from Hardhat, if deployed locally:  
      FIGURE I. On-Chain Hardhat Corpus Contract Address
2. Install the Corpus Administrator Frontend
   1. This requires a web server that is ONLY available to internal, administrative users.
   2. This frontend does not have any authentication to access, but files can only be added by users that have the MetaMask browser extension (installation is outlined in the User Manual) and a valid account for the blockchain where the Corpus contract was deployed.
   3. This requires Node.js with Express, which can be installed in the src/frontend directory with:  
        
      npm install express
   4. This will expose port 8080of the web server, so any firewalls must allow traffic.
   5. Additional networking devices are required to direct traffic to port 8080 on the web server.
   6. The following line in corpus\_interact.js must be updated with the correct smart contract address, retrieved in the above steps:  
        
      Line 179: this.contractAddress = '<smart\_contract\_address>';
   7. Apart from the above, no installations are required and this can be run on a server in the /src/frontend directory with:  
        
      node corpus\_server.js
3. Install the RAG Chatbot
   1. This requires a web server that is available to the desired user base.
   2. It does not have any authentication. Authentication can be added using [Streamlit Authentication](https://blog.streamlit.io/streamlit-authenticator-part-1-adding-an-authentication-component-to-your-app/).
   3. While this server doesn’t necessarily need a GPU, it will likely benefit performance if a GPU is installed and enabled.
   4. This will expose port 8501 of the web server, so any firewalls must allow traffic.
   5. Additional networking devices are required to direct traffic to port 8501 on the web server.
   6. This requires Python3
   7. This requires pip with the following installed into the active environment:  
        
      streamlit

chromadb

ragatouille

ollama

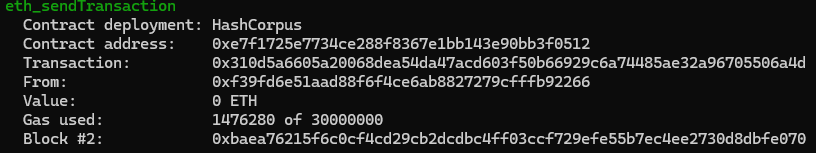
web3

pandas

* 1. These can be installed in the /src/rag directory with:  
       
     python -m ensurepip --upgrade  
     pip install -r ./requirements.txt
  2. Apart from the above, no installations are required and this can be run on a server in the /src/rag directory with:  
       
     python -m streamlit run rag\_blockchain\_streamlit.py

## Off-Chain Corpus

Deploying the off-chain version of this system is also relatively straightforward, but as the corpus is stored off-chain additional steps must be taken for storage. The following steps are required:

1. Deploy the HashCorpus smart contract in DocumentHashStore.sol contract located in /src/hardhat/contracts to the blockchain.
   1. This was done locally with the following command in the /src/hardhat directory:  
        
      npx hardhat run scripts/hash\_deploy.js --network localhost
   2. Take note of the deployed smart contract address. This can be retrieved from the block in the target blockchain or from Hardhat, if deployed locally:  
        
      FIGURE II. Off-Chain Hardhat HashCorpus Contract Address
2. Install the Corpus Administrator Frontend
   1. This requires a web server that is ONLY available to internal, administrative users.
   2. This frontend does not have any authentication to access, but files can only be added by users that have the MetaMask browser extension (installation is outlined in the User Manual) and a valid account for the blockchain where the Corpus contract was deployed.
   3. This requires Node.js with Express and Multer, which can be installed in the src/hash\_frontend directory with:  
        
      npm install express  
      npm install multer
   4. This will expose port 8080 of the web server, so any firewalls must allow traffic.
   5. Additional networking devices are required to direct traffic to port 8080 on the web server.
   6. The following line in corpus\_interact.js must be updated with the correct smart contract address, retrieved in the above steps:  
        
      Line 173: this.contractAddress = '<smart\_contract\_address>';
   7. The Corpus Administrator Frontend will write inputted documents to local storage by default. This is not best practice and so should be updated to write/read from shared storage, which the server will need access to. This should be updated in /src/hash\_frontend/corpus\_hash\_server.js at:
      1. Line 7, change const storage\_directory = '/upload/corpus/' to: change const storage\_directory = ‘<path\_to\_shared\_storage>'
   8. Apart from the above, no installations are required and this can be run on a server in the /src/hash\_frontend directory with:  
        
      node corpus\_hash\_server.js
3. Install the RAG Chatbot
   1. This requires a web server that is available to the desired user base.
   2. It does not have any authentication. Authentication can be added using [Streamlit Authentication](https://blog.streamlit.io/streamlit-authenticator-part-1-adding-an-authentication-component-to-your-app/).
   3. While this server doesn’t necessarily need a GPU, it will likely benefit performance if a GPU is installed and enabled.
   4. This will expose port 8501 of the web server, so any firewalls must allow traffic.
   5. Additional networking devices are required to direct traffic to port 8501 on the web server
   6. This requires Python3
   7. This requires pip with the following installed into the active environment:  
        
      streamlit

chromadb

ragatouille

ollama

web3

pandas

langchain

* 1. These can be installed in the /src/hash\_rag directory with:  
       
     python -m ensurepip --upgrade  
     pip install -r ./requirements.txt
  2. By default, the RAG will attempt to read from the Corpus Administrator Frontend located at <http://localhost:8080/download/>. Assuming the Frontend is not located on the same machine (per best practice, it should not be), then the following line in /src/hash\_rag/rag\_blochain\_hash\_stream.py should be updated:
     1. Line 199, change url = '<http://localhost:8080/download/>' to: url = 'http://<frontend\_url>:8080/download/'
        1. Note that if available, HTTPS should be used.
  3. However, it would be preferred if the RAG instead read the file directly from some shared storage. To enable this functionality, do the following:
     1. Line 183, change shared\_storage = False to: shared\_storage = True
     2. Line 184, change shared\_directory = 'None' to: shared\_directory = '<path\_to\_shared\_storage>'
  4. Apart from the above, no installations are required and this can be run on a server in the /src/hash\_rag directory with:  
       
     python -m streamlit run rag\_blockchain\_hash\_streamlit.py