**Deployment Documentation**

**Infrastructure Requirements**

This document outlines the deployment architecture and specifications for a 3 VM setup designed to support MongoDB, Chroma, and a staging environment. Each VM is configured to meet the performance, and reliability needs of the respective service it hosts.

1. VM1 MongoDB Server

* Purpose: Hosts the MongoDB Database to store and manage application data.
* Operating system: Ubuntu 20.04.6 LTS
* CPU: 4 vCPUs
* Memory: 16GB RAM
* Storage: 256GB

1. VM2 ChromaDB Server

* Purpose: Hosts ChromaDB for vector storage and retrieval, supporting high-performance data retrieval operations.
* Operating system: Ubuntu 20.04.6 LTS
* CPU: 4 vCPUs
* Memory: 16GB RAM
* Storage: 256GB

1. VM3 Staging Environment

* Purpose: Hosts the staging environment for testing and validation of application changes before production deployment.
* Operating system: Ubuntu 20.04.6 LTS
* CPU: 8 vCPUs
* Memory: 32GB RAM
* Storage: 256GB

**Networking for 3 VMs**

1. Virtual Network (VNet):

* All VMs must be hosted within the same Azure Virtual Network (VNet). This configuration ensures that they can communicate with each other efficiently and securely.
* The VNet should be configured with a custom address space, ensuring that IP addresses do not overlap with other networks.

1. Subnets:

* The VMs should be placed in the same subnet within the VNet. This allows for low-latency communication and simplifies the network configuration since no additional routing or network security group (NSG) rules are needed for internal communication between the VMs.
* Each VM should be assigned a private IP address within this subnet.

1. Network Security Groups (NSG):

* An NSG must be associated with the subnet or individual VMs to control inbound and outbound traffic.

**MongoDB Installation on Ubuntu VM**

1. Prerequisites –

* Ubuntu VM: Ensure that an Ubuntu VM is set up and accessible.
* Root or Sudo Access: Installation requires root or sudo privileges.
* Firewall Configuration: Ensure that necessary ports are open, specifically port 27017 for MongoDB.

1. Update the System –

* sudo apt-get update
* sudo apt-get upgrade -y

1. Install **nano** Text Editor –

* sudo apt-get install -y nano

1. Import the public key used by the package management system –

* sudo apt-get install gnupg curl
* curl -fsSL https://www.mongodb.org/static/pgp/server-7.0.asc | \ sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg \ --dearmor

1. Create a list file for MongoDB –

* echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-7.0.gpg ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/7.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-7.0.list

1. Reload local package database –

* sudo apt-get update

1. Install the MongoDB packages –

* sudo apt-get install -y mongodb-org

1. Start and Enable MongoDB Service –

* sudo systemctl start mongod

If receiving an error similar to the following when starting mongod:

* Failed to start mongod.service: Unit mongod.service not found.

Run the following command first:

* sudo systemctl daemon-reload

Then run the start command above again.

1. Verify that MongoDB has started successfully –

* sudo systemctl status mongod
* sudo systemctl enable mongod

1. Verify MongoDB Installation –

* sudo systemctl status mongod

1. Configure MongoDB –

* Bind IP Address –

1. To configure MongoDB to accept connections from specific IP addresses, edit the MongoDB configuration file.

* sudo nano /etc/mongod.conf

In the configuration file, locate the bindIp line and modify it as needed:

* bindIp: <staging\_vm\_private\_ip>

Save the file. And to ensure that the changes take effect.

* sudo systemctl daemon-reload
* sudo systemctl restart mongod

**Deployment Documentation for ChromaDB Docker Container on Ubuntu VM**

1. Prerequisites –

* Ubuntu VM: Ensure that an Ubuntu VM is set up and accessible.
* Root or Sudo Access: Installation requires root or sudo privileges.
* Firewall Configuration: Ensure that necessary ports are open, specifically port 8000 for ChromaDB.

1. Update the System –

* sudo apt-get update
* sudo apt-get upgrade -y

1. Install Docker –

* sudo apt install docker.io
* sudo snap install docker

1. Verify the Docker Installation –

* sudo systemctl status docker
* sudo docker –version

Ensure Docker is running, and installation was successful.

1. Loading the ChromaDB Docker Image –

* Load the ChromaDB Docker Image from the .tar file. (Transferred file)

1. sudo docker load -I /path/to/chromadb\_image.tar

Replace /path/to/chromadb\_image.tar with the actual path to the image file. Verify the image by listing Docker Images –

1. sudo docker images
2. Running the Docker container –

* sudo docker run --network host --name chromadb\_container -d <image\_name>

1. <image\_name> should be replaced with the actual name of the ChromaDB image.
2. The --network host option allows the container to share the host’s network.
3. The -d flag runs the container in detached mode.
4. Ensuring the Container Runs Continuously –

* To ensure that the ChromaDB container runs continuously, configure it to start automatically on boot:

1. Enable Docker service to start at boot:

* sudo systemctl enable docker

1. Restart the container automatically on failure or reboot by setting the restart policy:

* sudo docker update --restart unless-stopeed chromadb\_container

This command configures the container to restart automatically unless it is explicitly stopped.

1. Verification –

* List all running containers
* sudo docker ps
* Check the container’s restart policy
* sudo docker inspect -f “{{ .HostConfig.RestartPolicy.Name }}” chromadb\_container

This should return unless-stopped, confirming that the container is configured to restart as required.

1. Container Management –

* Stopping the container
* sudo docker stop chromadb\_container
* Starting the container
* sudo docker start chromadb\_container
* Removing the container
* sudo docker rm chromadb\_container
* Removing the image
* sudo docker rmi <image\_name>

1. Monitoring and Logging –

* View container logs
* sudo docker logs chromadb\_container
* Monitor the container’s status
* sudo docker ps -a

**Deployment Documentation for the Application**

1. Prerequisites –

* Ubuntu VM: Ensure that an Ubuntu VM is set up and accessible.
* Root or Sudo Access: Installation requires root or sudo privileges.
* Firewall Configuration: Ensure that necessary ports are open. (Example – 80,443, or custom port)

1. Update the System –

* sudo apt-get update
* sudo apt-get upgrade -y

1. Install Essential Tools –

* sudo apt install -y build-essential curl git nano

1. Install Python3.11 and setting up venv –

* Add the Deadsnakes PPA
* sudo add-apt-repository ppa:deadsnakes/ppa
* sudo apt update
* Install python3.11
* sudo apt install python3.11 python3.11-venc python3.11-dev -y

1. Install Node.js and npm –

* curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh | bash
* nvm install 20
* node -v # should print `v20.17.0`
* npm -v # should print `10.8.2`

1. Install nginx –

* sudo apt install nginx

1. Install LibreOffice

* sudo apt install libreoffice

1. Install Poppler and tesseractOCR –

* sudo apt-get install poppler-utils
* sudo apt install tesseract-ocr
* sudo apt install libtesseract-dev

1. Clone the Repository –

* Navigate to Desired Directory
* git clone <git\_repo\_link>

1. Navigate to Project Directory –
2. Express Application –
3. Navigate to express folder
4. Install project Dependencies

* npm install

1. Create .env file

* sudo nano .env
* Create necessary variables
* MONGO\_API\_KEY=mongodb://<piravte\_ip\_vm\_with\_mongo>:27017/GatesVentures\_Scientia

1. Setting up PM2

* sudo npm install -g pm2
* pm -v # verify installation

1. Configure ecosystem.config.js
2. Start the Application –

* pm2 start ecosystem.config.js
* pm2 status # verify the application is running

1. Generate Startup Script –

* pm2 startup

This command will output a command tailored to the system. Copy and execute the command provided by the above command.

1. Save the Process List

* pm2 save

1. Managing the application

* List All PM2 Process
* pm2 list
* View Detailed Status of all Processes
* pm2 status
* View Logs for a Specific Application
* pm2 logs <app\_name\_in\_ecosystem.config.js>
* Restart an application
* pm2 restart <app\_name\_in\_ecosystem.config.js>
* Stop an application
* pm2 stop <app\_name\_in\_ecosystem.config.js>
* Delete an application from PM2
* pm2 delete <app\_name\_in\_ecosystem.config.js>
* Reload all applications
* pm2 reload all

1. FastAPI Application –
2. Navigate to fast directory
3. Create a venv

* python3.11 -m venv /path/to/venv/<name\_of\_fast\_app\_venv>

Replace /path/to/env with actual path where to create venv.

Replace < name\_of\_fast\_app\_venv > with desired name environment.

1. Activate venv

* source /path/to/venv/< name\_of\_fast\_app\_venv >/bin/activate
* python3.11 -m pip install -U pip

1. Install FastAPI and Dependencies

* pip install -r requirements.txt

1. Set Up Environment Variables by creating .env file

* sudo nano .env
* Create necessary variables
* SECURITY\_HEADER=testing123
* MONGODB\_COLLECTION=GatesVentures\_Scientia
* MONGO\_API\_KEY=mongodb<private\_ip\_vm\_with\_mongo>:27017/
* AZURE\_OPENAI\_CHAT\_DEPLOYMENT\_NAME\_GPT\_35=gpt-35-turbo-16k
* AZURE\_OPENAI\_CHAT\_DEPLOYMENT\_NAME\_GPT\_4O=gpt-4o
* AZURE\_OPENAI\_CHAT\_DEPLOYMENT\_NAME\_EMBEDDING=text-embedding-3-large
* AZURE\_OPENAI\_API\_KEY=029e338b04494297ab81c72496f68939
* AZURE\_OPENAI\_ENDPOINT=https://scientia-docx-staging-azureopenai.openai.azure.com/
* AZURE\_OPENAI\_API\_VERSION=2024-02-01
* CHROMADB\_HOST=<private\_ip\_vm\_with\_chromaDB>

1. Edit the chromaDB package

* Edit init file of ChromaDB
* sudo nano /path/to/venv/< name\_of\_fast\_app\_venv >/lib/python3.11/site-packages/chromadb/\_\_init\_\_.py
* Paste the below line of code at top and save the changes.

\_\_import\_\_('pysqlite3')

import sys

sys.modules['sqlite3'] = sys.modules.pop('pysqlite3')

1. Create a service file for the fast Application

* sudo nano /etc/systemd/system/<name\_of\_fast\_app\_service>.service
* Paste the below configurations

[Unit] Description=Gunicorn instance serve Fast app After=network.target [Service] User=<name\_of\_user> # example - mayank.sharma9@evalueserve.com Group=www-data WorkingDirectory=/path/to/working/directory # example-/home/mayank.sharma9/GV/fast Environment="PATH=/path/to/fast\_app\_venv /bin" # example -/home/mayank.sharma9/fast\_application/bin ExecStart=/path/to/fast\_app\_venv /bin/gunicorn main:app --workers 4 --worker-class uvicorn.workers.UvicornWorker --preload --bind 0.0.0.0:6677 # example - /home/mayank.sharma9/fast\_application

Restart=on-failure [Install] WantedBy=multi-user.target

1. Reload system manager configuration and start the FastAPI service

* sudo systemctl daemon-reload
* sudo systemctl start <name\_of\_fast\_app\_service>.service
* sudo systemctl enable <name\_of\_fast\_app\_service>.service

1. Ingestion pipeline
2. Navigate to ingestion directory
3. Create a venv

* python3.11 -m venv /path/to/venv/<name\_of\_ingestion\_app\_venv>

Replace /path/to/env with actual path where to create venv.

Replace < name\_of\_ingestion\_app\_venv > with desired name environment.

1. Activate venv

* source /path/to/venv/< name\_of\_ingestion\_app\_venv >/bin/activate
* python3.11 -m pip install -U pip

1. Install Dependencies

* pip install -r requirements.txt

1. Set Up Environment Variables by creating .env file

* sudo nano .env
* Create necessary variables
* CLIENT\_ID=fd864c82-a498-4db7-97b5-5e0ce24ec700
* CLIENT\_SECRET=BBn8Q~FZW47Cf3ax9bKI5dfIYQSRAuvy.lBTMcug
* SHAREPOINT\_SITE=https://gatesventures.sharepoint.com/sites/scientia
* SHAREPOINT\_SITE\_NAM=scientia
* TENANT\_ID=e38fdd56-dd64-4d67-b048-e7c0ae321d11
* USER\_PERMISSION\_LIST=167bbde3-1341-4d29-9447-0996b92c26ef
* DELIVERABLES\_LIST=a76c34cd-0a87-4947-9881-54a32eb64b4e
* DRIVE\_ID=b!XLuFWblTu06sb5qUAEHt9zdujDxJ7RRCuVCHQZTlYonNNGynhwpHSZiBVKMutktO
* AZURE\_OPENAI\_API\_KEY=029e338b04494297ab81c72496f68939
* AZURE\_OPENAI\_ENDPOINT=https://scientia-docx-staging-azureopenai.openai.azure.com/
* AZURE\_OPENAI\_API\_VERSION=2024-02-01
* AZURE\_OPENAI\_CHAT\_DEPLOYMENT\_NAME=gpt-4o
* AZURE\_OPENAI\_EMBEDDINGS\_MODEL=text-embedding-3-large
* MONGO\_API\_KEY=mongodb<private\_ip\_vm\_with\_mongo>:27017/
* MONGODB\_COLLECTION=GatesVentures\_Scientia
* CHROMADB\_HOST=<private\_vm\_ip\_with\_chromaDB>

1. Edit the chromaDB package

* Edit init file of ChromaDB
* sudo nano /path/to/venv/< name\_of\_ ingestion \_app\_venv >/lib/python3.11/site-packages/chromadb/\_\_init\_\_.py
* Paste the below line of code at top and save the changes.

\_\_import\_\_('pysqlite3')

import sys

sys.modules['sqlite3'] = sys.modules.pop('pysqlite3')

1. Setup cron job

* Edit the crontab
* crontab -e
* Add the cron job
* 0 0 \* \* \* /bin/bash -c “source /path/to/venv/< name\_of\_ ingestion \_app\_venv >/bin/activate && python3.11 /path/to/ingestion/directory/sharepoint\_file\_acquisition.py

* Save and exit

The script will run at midnight every day.

1. Table Detection API
2. Navigate to api directory
3. Create a venv

* python3.11 -m venv /path/to/venv/<name\_of\_api\_app\_venv>

Replace /path/to/env with actual path where to create venv.

Replace < name\_of\_api\_app\_venv > with desired name environment.

1. Activate venv

* source /path/to/venv/< name\_of\_api\_app\_venv >/bin/activate
* python3.11 -m pip install -U pip

1. Install Dependencies

* pip install -r requirements.txt

1. Create a service file for the fast Application

* sudo nano /etc/systemd/system/<name\_of\_api\_app\_service>.service
* Paste the below configurations

[Unit] Description=Gunicorn instance serve Table Detection API After=network.target [Service] User=<name\_of\_user> # example - mayank.sharma9@evalueserve.com Group=www-data WorkingDirectory=/path/to/working/directory # example-/home/mayank.sharma9/GV/api Environment="PATH=/path/to/api\_app\_venv /bin" # example -/home/mayank.sharma9/api\_application/bin ExecStart=/path/to/api\_app\_venv /bin/gunicorn image2table:app --workers 2 --worker-class uvicorn.workers.UvicornWorker --bind 0.0.0.0:8183 # example - /home/mayank.sharma9/api\_application

Restart=on-failure [Install] WantedBy=multi-user.target

1. Reload system manager configuration and start the Table API service

* sudo system daemon-reload
* sudo systemctl start <name\_of\_api\_app\_service>.service
* sudo systemctl enable <name\_of\_api\_app\_service>.service

1. Setting up nginx

* Create a new configuration file for nginx
* sudo nano /etc/nginx/sites-available/GatesVentures
* Add the following configuration

server {

listen 80;

server\_name scientia-docx-stg.gates.ventures; #replace with server name

return 301 https://$host$request\_uri;

}

server {

listen 443 ssl;

server\_name scientia-docx-stg.gates.ventures; #replace with server name

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_certificate /path/to/fast/directory /certificates/certificate.pem;

ssl\_certificate\_key /path/to/fast/directory /certificates/private\_key.pem;

location /express{

proxy\_pass https://localhost:8080;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

location /fast{

proxy\_pass http://127.0.0.1:6677;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_buffering off;

proxy\_cache off;

proxy\_request\_buffering off;

}

}

* Enable configuration by creating a symbolic link to sites-enabled
* Sudo ln -s /etc/nginx/sites-available/GatesVentures /etc/nginx/sites-enabled/
* Test the configuration for syntax errors
* sudo nginx -t
* Reload nginx to apply the changes
* sudo systemctl reload nginx