**Companion Development: Installation & Deployment Guide**

# Table of Contents

[**Requirements** 3](#_Toc160133230)

[**VM details** 4](#_Toc160133231)

[**Environment Setup** 4](#_Toc160133232)

[Port Access 4](#_Toc160133233)

[SSH configuration for Git in VM 4](#_Toc160133234)

[Installing Apache server 5](#_Toc160133235)

[SSL configuration for the Web App 5](#_Toc160133236)

[Install/Upgrade Python 5](#_Toc160133237)

[Install/Upgrade NodeJs 5](#_Toc160133238)

[Install MongoDB 6](#_Toc160133239)

[Create Python Virtual environment 6](#_Toc160133240)

[Install/configure Milvus DB with Docker 6](#_Toc160133241)

[Creating screens in the VM 7](#_Toc160133242)

[Code deployment process 7](#_Toc160133243)

# **Requirements**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Current Architecture** | | **Approved AWS Architecture** | |
| **Components** | **Tech Stack** | **Specification** | **Tech Stack** | **Specification** |
| Hardware | Azure VM RAM | 32GB | AWS EC2 - RAM | 32 GB |
| Azure VM Storage | 128 GB | AWS EC2 - Storage | 128 GB |
| Software | NodeJS | 18.17.1 | NodeJS | 18.17.1 |
| Django | 4.2.7 | Django | 4.2.7 |
| Python | 3.10.13 | Python | 3.10.13 |
| Mongo DB metadata | Self managed | Mongo DB metadata | Self managed |
| Vector Database | Milvus (install in docker) | Vector Database | Milvus (install in docker) |
| Environment | Admin access to the VM |  | Admin access to the AWS |  |
| Access to Azure OpenAI API from VM | Azure Open AI | Access to OpenAI API from VM | OpenAI key from Amgen Account |
| Azure Linux VM | ubuntu 20.04 | Linux VM | ubuntu 20.04 |
| Apache HTTP server |  | Apache HTTP server |  |
| Allow ports for UI and Backend | ports - 443, 80, 8000 & 8001 | Allow ports for UI and Backend | ports - 443, 80, 8000 & 8001 |
| Networking | SSL certificate | 3 certificates (.key,.crt,.pfx) | SSL certificate | 3 certificates (.key ,.pfx or .pem,.crt) |
| DNS | prod - https://companion.horizontherapeutics.com/ | DNS | Dev - https://companion\_dev.amgen.com/ Stage - https://companion\_staging.amgen.com/ Prod - https://companion.amgen.com |
| Internet Access | Must have | Internet Access | Must have |
| Dev tools | Okta account with credentials | Out of scope | Okta account with credentials | Out of scope |
| Azure blob storage |  | S3 Bucket storage |  |
| GitHub HZN |  | GitHub AMGEN |  |
| GenAI | Azure-OpenAI Chat Model | GPT 3.5 Turbo 16K - 0613 | OpenAI Chat Model | GPT 3.5 Turbo 16K - 0125 |
| PDF Parser | Adobe PDF Service API | PDF Parser | AWS Textract |
| Azure-OpenAI Embedding Model | text-embedding-ada-002 | OpenAI Embedding Model | text-embedding-ada-002  text-embedding-ada-003 |

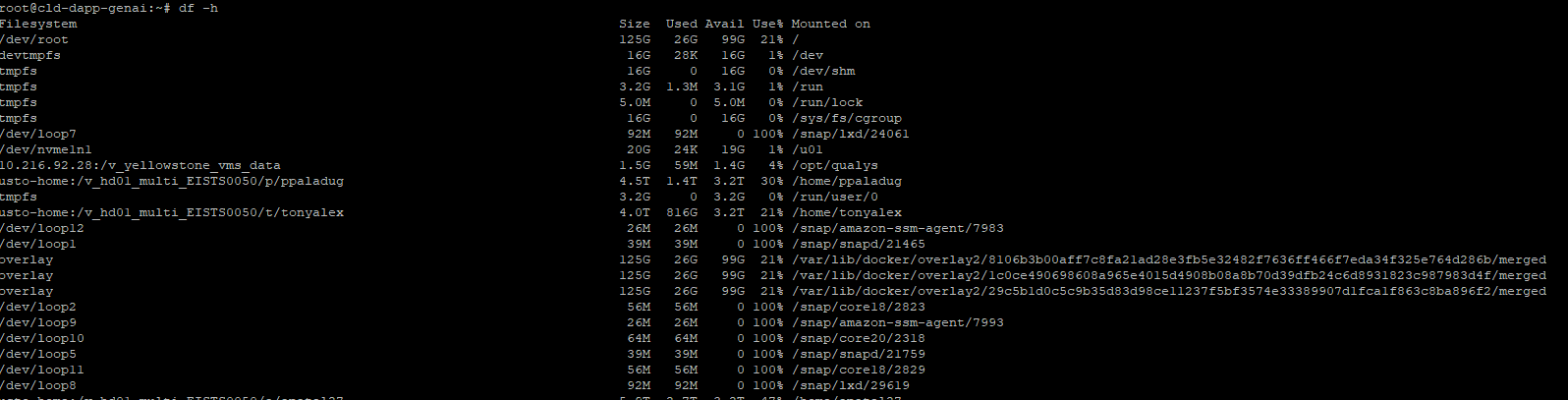
# **VM details**

Configuration details

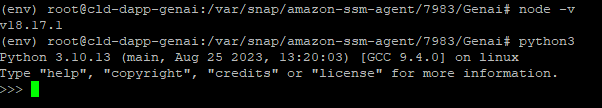
1. Memory - 32GB
2. RAM – 32GB
3. IP - 10.23.3.143

Steps to connect

1. Connect using putty –



1. Current Node version - 18.17.1
2. Current Python - 3.10.13



# **Environment Setup**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Packages** | **Commands to install** | **ETA (in days)** | **Comment** |
| 1 | Check port access | 1. ss –tlun | grep 443 2. ss –tlun | grep 8000 3. ss –tlun | grep 8001 | - | Command should list out the ports if access if enabled. |
| 2 | Install/Upgrade Python | 1. sudo su 2. apt update 3. apt upgrade 4. apt install software-properties-common 5. add-apt-repository ppa:deadsnakes/ppa 6. apt update 7. apt install python3.10 8. python3.10 --version 9. python3.10 10. exit() 11. apt-get install python3.10-venv | 1 | Python version – 3.10 |
| 3 | Install/Upgrade NodeJs | 1. sudo su 2. apt install nodejs 3. curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash 4. source ~/.bashrc 5. nvm install 18.17.1 6. nvm use 18.17.1 7. nvm alias default 18.17.1 8. node -v | 1 | Node version – 18.17.1 |
| 4 | Install MongoDB | 1. sudo apt update 2. wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add - 3. sudo nano /etc/apt/sources.list.d/mongodb-org-5.0.list 4. deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0 multiverse 5. sudo apt update 6. sudo apt install mongodb-org 7. sudo systemctl start mongod 8. sudo systemctl enable mongod | 1 | - |
| 5 | MongoDB – DB/Collection creation | 1. mongo 2. use genai 3. db.createCollection('app\_feedback') 4. db.createCollection('chat\_messages') 5. db.createCollection('chat\_session') 6. db.createCollection(' faq\_details') 7. db.createCollection(' upload\_meta\_data') 8. db.createCollection('user\_details') |  |  |
| 6 | Create Python Virtual environment | 1. python3.10 –m venv env 2. source env/bin/activate 3. pip install –r requirements.txt 4. Run server - python manage.py runserver 0.0.0.0:8000 | - |  |
| 7 | Install/configure Milvus DB with Docker | 1. Remove conflicting packages: *for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done* 2. apt-get autoclean 3. apt-get autoremove 4. apt-get update 5. apt-get update 6. apt-get install ca-certificates curl 7. install -m 0755 -d /etc/apt/keyrings 8. curl –fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc 9. chmod a+r /etc/apt/keyrings/docker.asc 10. apt-get update 11. Install Docker and Docker Compose: *$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin* ( Ref: [Install Docker](https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository)) 12. Check Docker version - docker –v 13. Check Docker compose version - docker compose version 14. Verify that the Docker Engine installation is successful by running the hello-world image –     1. docker run hello-world |  | Docker version 25.0.4 |
| 8 | Installing Milvus Standalone using Docker Compose | 1. Create a milvus\_docker folder in base directory - mkdir milvus\_docker 2. Go to this folder - cd milvus\_docker 3. Download YAML file – wget https://github.com/milvus-io/milvus/releases/download/v2.3.10/milvus-standalone-docker-compose.yml -O docker-compose.yml 4. Start Milvus - sudo docker compose up -d 5. Check status - sudo docker compose ps |  | Docker Compose 2.24.7  Milvus version 3.5 |
| 9 | Creating KnowledgeBase & Vectorization | 1. Migrate KnowledgeBase PDFs & adobe extracted chunks from dev s3 to prod s3   Folder names-   * 1. Tepezza\_KB\_Adobe\_Extracted\_Chunks   2. Tepezza\_KnowledgeBase\_Documents  1. cd Genai/genai\_env/ 2. source env/bin/activate 3. cd .. 4. cd companion-ai/Backend/gen\_ai/ai\_core/CreateKnowledgeBase 5. python milvus\_standalone.py 6. python check\_collection\_info.py |  |  |
| 9 | SSH configuration for Git in VM | 1. ssh-keygen -t ed25519 2. copy the code from the given file ~/.ssh/id\_ed25519.pub 3. Goto GitHub settings    1. In the "Access" section of the sidebar, click **SSH and GPG keys**    2. Click **New SSH key** or **Add SSH key**.    3. In the "Title" field, add a descriptive label for the new key. For example, if you're using a personal laptop, you might call this key "Personal laptop".    4. In the "Key" field, paste your public key.    5. Click **Add SSH key**. 4. Clone repo through SSH link 5. Ignoring file permission in git - git config --global core.fileMode false | 1 |  |
| 10 | SSL configuration | 1. Get .crt and .key files and create new ACM. 2. Under load balancer, add new listener for HTTPS with SSL certificate under ACM dropdown | - |  |
| 11 | Creating screens in the VM | 1. apt-get install screen 2. Create screen: screen –S screen\_name 3. List screens: screen –ls 4. Attaching screen: screen -x screen\_name 5. Detaching screen: Ctrl + a + d 6. Deleting screen: screen -X -S session\_id quit |  |  |
| 12 | Code deployment process | 1. Pull the code from Main branch using git command – git pull origin main 2. Deploying UI manually –    1. Goto - /GenAI\_Tool/Frontend/genai\_tool/    2. Run – npm install    3. npm run build 3. Deploying API manually    1. Activate screen – screen -x backend    2. Activate python virtual environment - source env/bin/activate    3. Run – pip install -r requirement.txt inside /GenAI\_Tool/Backend/gen\_ai    4. Run the server – python /manage.py runserver 0.0.0.0:8000 |  |  |

## Port Access

Check for the port access in VM. We are using port 443(https) for frontend and port 8000 for running backend server.

1. ss –tlun | grep 8001
2. ss –tlun | grep 8000

## SSH configuration for Git in VM

1. ssh-keygen -t ed25519
2. copy the code from the given file ~/.ssh/id\_ed25519.pub
3. Goto GitHub settings
   1. In the "Access" section of the sidebar, click **SSH and GPG keys**
   2. Click **New SSH key** or **Add SSH key**.
   3. In the "Title" field, add a descriptive label for the new key. For example, if you're using a personal laptop, you might call this key "Personal laptop".
   4. In the "Key" field, paste your public key.
   5. Click **Add SSH key**.
4. Clone repo through SSH link
5. Ignoring file permission in git - git config --global core.fileMode false

## Installing Apache server – **Not required for current architecture**

1. sudo apt update
2. sudo apt install apache2
3. sudo systemctl status apache2
4. Commands to start/stop/restart/status of Apache2 Service:  
   • To enable apache2 Service:  sudo systemctl enable apache2   
   • To Start Apache2 Service:  sudo systemctl start apache2   
   • To Restart Apache2 Service:  sudo systemctl restart apache2   
   • To Stop Apache2 Service:   sudo systemctl stop apache2   
   • To check status/errors of Apache2 Service:  sudo systemctl status apache2  
   • Install rewrite Module : sudo a2enmod rewrite  
   • Install Proxy Module:  sudo a2enmod proxy   
   •  Install proxy http: sudo a2enmod proxy\_http   
   • Install SSL Module: sudo a2enmod ssl

SSL configuration for the Web App - **Not required for current architecture**

1. Collect the SSL certificates and place it under a directory in VM
2. Convert the .pfx file into key file

openssl pkcs12 -in [yourfilename.pfx] -nocerts -out [keyfilename-encrypted.key]

1. Configure conf file with ssl certificate files - /etc/apache2/apache2.conf and code location for UI and API

## Install/Upgrade Python

1. sudo su
2. apt update
3. apt upgrade
4. apt install software-properties-common
5. add-apt-repository ppa:deadsnakes/ppa
6. apt update
7. apt install python3.10
8. python3.10 --version
9. python3.10
10. exit()
11. apt-get install python3.10-venv

## Install/Upgrade NodeJs

1. sudo su
2. apt install nodejs
3. curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
4. source ~/.bashrc
5. nvm install 18.17.1
6. nvm use 18.17.1
7. nvm alias default 18.17.1
8. node -v

## Install MongoDB

1. sudo apt update
2. wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add -
3. sudo nano /etc/apt/sources.list.d/mongodb-org-5.0.list
4. deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0 multiverse
5. sudo apt update
6. sudo apt install mongodb-org
7. sudo systemctl start mongod
8. sudo systemctl enable mongod

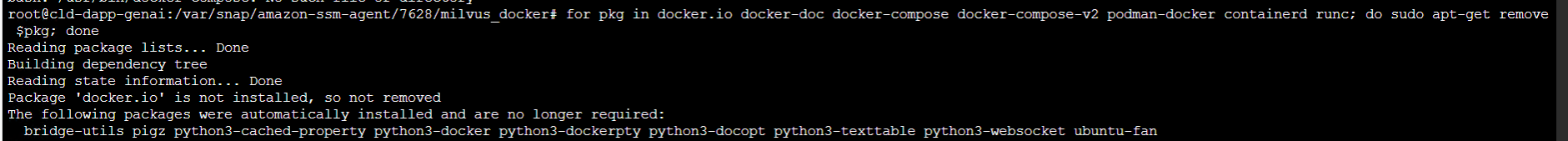
## Create Python Virtual environment

1. python3.10 –m venv env
2. source env/bin/activate
3. pip install –r requirements.txt
4. Run server - python manage.py runserver 0.0.0.0:8000

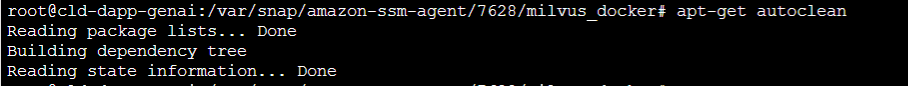
## Install/configure Milvus DB with Docker

Please follow the steps below to install the [**latest versions of Docker and Docker Compose**](https://docs.docker.com/engine/install/ubuntu/) on your Ubuntu machine:

1. Remove conflicting packages: *$ for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done*



1. $ apt-get autoclean



1. $ apt-get autoremove

A computer screen with white text

Description automatically generated

1. $ apt-get update

A computer screen with white and blue text

Description automatically generated

1. $ sudo apt-get install ca-certificates curl

A black screen with white text

Description automatically generated

1. $ sudo install -m 0755 -d /etc/apt/keyrings
2. $ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
3. $ sudo chmod a+r /etc/apt/keyrings/docker.asc
4. $ sudo apt-get update

A computer screen with white text

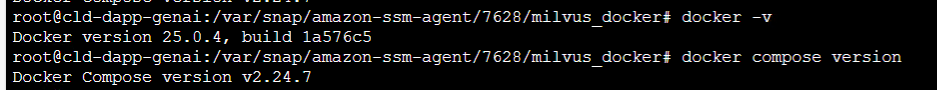
Description automatically generated

1. Install Docker and Docker Compose: *$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin* ( Ref: [Install Docker](https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository))

A computer screen with a black background

Description automatically generated

1. Check Docker & docker compose version –
   1. $ docker -v
   2. $ docker compose version



1. Verify that the Docker Engine installation is successful by running the hello-world image –
   1. $ sudo docker run hello-world

A screenshot of a computer program

Description automatically generated

**Other commands if this doesn’t work -**

1. Install Docker Compose plugin: *$ sudo apt-get install docker-compose-plugin* (Ref: [Update Package List](https://docs.docker.com/compose/install/linux/#install-using-the-repository))

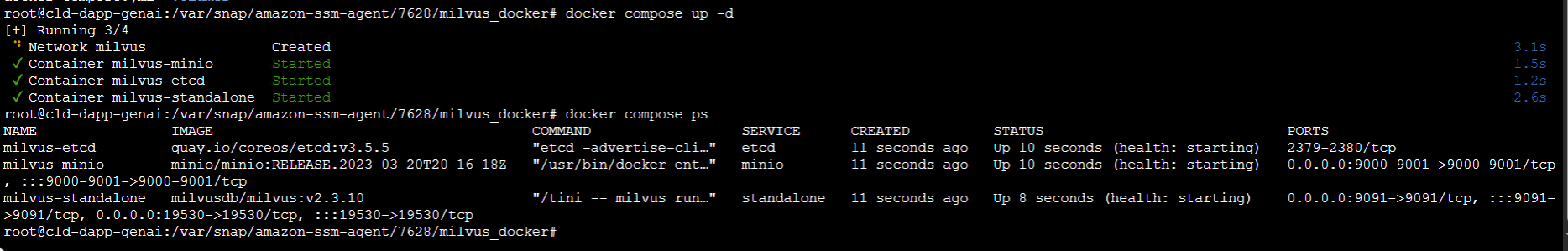
**Installing Milvus Standalone using Docker Compose:**

To install Milvus Standalone using Docker Compose on an Ubuntu machine, please follow the exact command provided in the link in a new directory ([Milvus Standalone](https://milvus.io/docs/install_standalone-docker-compose.md))

1. Create a milvus\_docker folder in base directory -> $ mkdir milvus\_docker
2. Go to this folder -> $ cd milvus\_docker
3. Download YAML file –

$ wget https://github.com/milvus-io/milvus/releases/download/v2.3.10/milvus-standalone-docker-compose.yml -O docker-compose.yml

1. Start Milvus - $ sudo docker compose up -d
2. Check status - $ sudo docker compose ps



----------------------------------------------------------------------------------------------------------------------------------------------------------------Helping commands to Troubleshoot

1. To check status – sudo docker compose ps
2. sudo docker compose ps -a
3. To start container by id - sudo docker start container\_id
4. To start & exit all containers – sudo docker compose down
5. sudo docker compose up
6. To check memory – free -h / df -h
7. if you get label error on docker compose up then do below – If the labels do match, you can try removing and recreating the network using the commands
   1. docker network rm milvus
   2. docker network create milvus

## Creating KnowledgeBase & Vectorization -

1. Migrate KnowledgeBase PDFs & adobe extracted chunks from dev s3 to prod s3

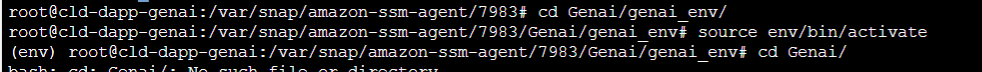
Folder names-

* 1. Tepezza\_KB\_Adobe\_Extracted\_Chunks
  2. Tepezza\_KnowledgeBase\_Documents

A black screen with white dots

Description automatically generated

1. Activate virtual env –
   1. $ cd Genai/genai\_env/
   2. $ source env/bin/activate



1. Run Python script to vectorize knowledge base-
   1. $ cd ..
   2. $ cd companion-ai/Backend/gen\_ai/ai\_core/CreateKnowledgeBase
   3. $ python milvus\_standalone.py

A black screen with white text

Description automatically generated

1. To check collections present in milvus –
   1. $ python check\_collection\_info.py

A black screen with white text

Description automatically generated

1. Deactivate env –
   1. $ deactivate

## Creating screens in the VM

1. apt-get install screen
2. Create screen: screen –S screen\_name
3. List screens: screen –ls
4. Attaching screen: screen -x screen\_name
5. Detaching screen: Ctrl + a + d
6. Deleting screen: screen -X -S session\_id quit

## Code deployment process

1. Pull the code from Main branch using git command – git pull origin main
2. Deploying UI manually –
   1. Goto - /GenAI\_Tool/Frontend/genai\_tool/
   2. Run – npm install --force
   3. npm run build
3. Deploying API manually
   1. Activate screen – screen -x backend
   2. Activate python virtual environment - source env/bin/activate
   3. Run – pip install -r requirement.txt inside /GenAI\_Tool/Backend/gen\_ai
   4. Run the server – python /manage.py runserver 0.0.0.0:8000

**Amazon load balancer and SSL configuration**

1. Goto ACM and create new with SSL certificate files (.crt and .key).
2. Update Listener under Load balancer to HTTPS
   1. Goto “Edit listener”
   2. Change Protocol to HTTPS
   3. Under Secure listener settings, select Default SSL/TLS server certificate as “from ACM” and select the SSL certificate which is already created and save the changes.
3. Create 2 Rules under the load balancer for Frontend and Backend
   1. Create new **Target Group** for Backend with backend port: 8000
   2. Create new **Target Group** for Frontend with backend port: 8001
   3. Under health check, add “404” code along with 200 in **Success code**.
   4. Add new rule under listener
      1. Goto Add rule button
      2. Click next -> Add condition -> Choose condition (Path) -> and put path - (/api/v1/\*)
      3. Click on Confirm
4. Run the application in those ports in EC2 for UI and BE.