Gebo.ai configuration manual (1.0.0.0-rc1/SNAPSHOT)

Enterprise open source retrieve augmented generation, chatbot and agents platform.

Sommario

Gebo.ai “monolithic version” on premise installation structure

This document lists both already developed and under development integrations.

The “monolithic version” is called so contraposed to the “Microservices version” that will be available soon, even if suboptimal “Monolithic” software in this case is easier to install and already tested for installations with up to 10000 documents indexed and retrieved in the chat context.

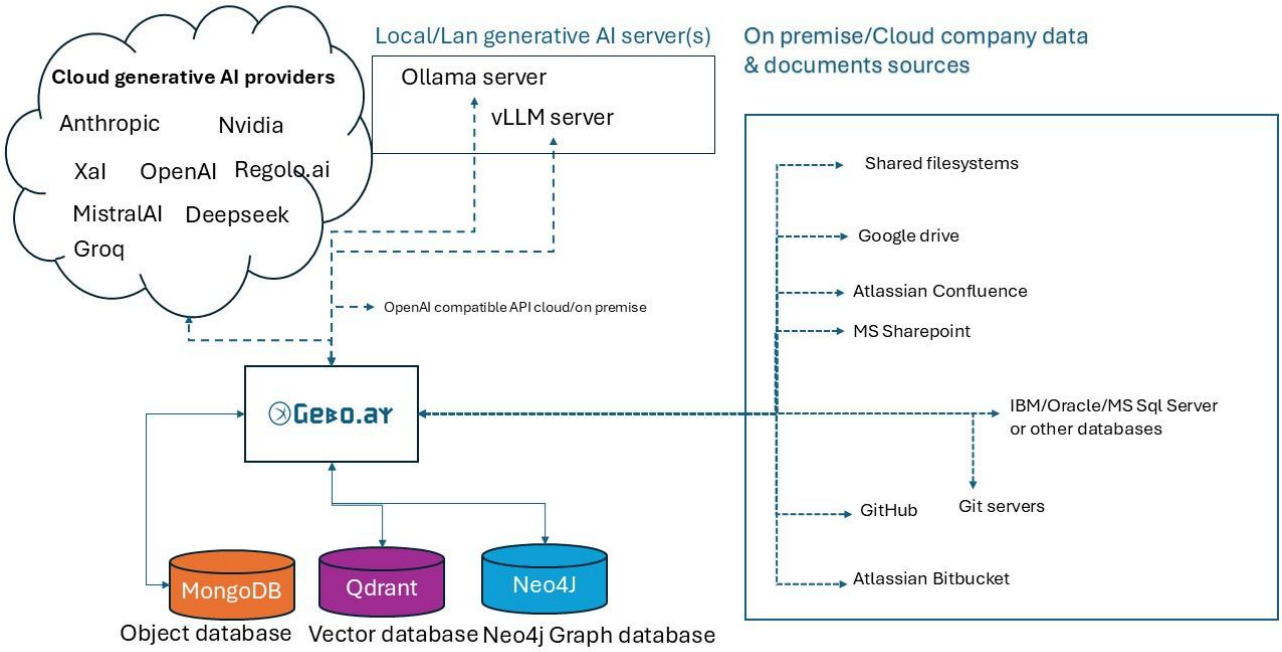


Figure 1Gebo.ai on premise installation components and 3rd party integration schema

Esyinstall docker installation

Image name: geboai/easyinstall.gebo.ai

Public docker repository: https://hub.docker.com/r/geboai/easyinstall.gebo.ai

This is the “go to” installation for initial tests, will not enable graphrag feature but you can use all the retrieve augmented generation features, erogating chatbots with a fully featured multilanguage/multiuser application from a single docker image.

You can simply launch:

docker run -p 12999:12999 –name local-gebo-ai geboai/easyinstall.gebo.ai:latest

After few seconds you can connect to: http://<your server>:12999/ ad start configuring your software.

With this image you can scale to a full onpremise installation by extracting all its components backups (gebo.ai software, mongodb backup, qdrant backup) and restoring them in your custom installation.

Important prerequisites: **if you plan to use the easyinstall installation on a linux host operating system, ensure to run a "UTF-8" locale like en_US.UTF-8 localization or it_IT.UTF-8 (for Italian) this is mandatorly if you plan to mount windows filesystem shares to be accessible to Gebo.ai.**

Docker compose installation

Image name: geboai/gebo.ai

Public docker repository: https://hub.docker.com/r/geboai/gebo.ai

Important prerequisites: **if you plan to use the docker-compose installation on a linux host operating system, ensure to run a "UTF-8" locale like en_US.UTF-8 localization or it_IT.UTF-8 (for Italian) this is mandatory if you plan to mount windows filesystem shares to be accessible to Gebo.ai.**

On ubuntu server for example run:

#sudo apt-get install language-pack-en

#sudo locale-gen en_US.UTF-8

#update-locale LANG=en_US.UTF-8

The provided docker compose **docker-compose.yml** configuration files works without any required modification.

It setups both MongDB, Qdrant, Gebo.ai setting up their connectivity and exporting the port 12999 where the gebo.ai application is bound.

The docker-compose configuration let the 3 applications run as a service, so if in your system dockerd or the containerized subsystem is running as a service with automatic startup at system boot, you are fully ready on erogating this service on your local network.

It is very productive to create an "up and running" quite performant installation but contains security settings like mongo password and qdrant api key that we suggest to change for production use (go to the security considerations section to check these topics).

1st Starting up the application

The fastest way to obtain an up & running environment is copying the provided docker-compose.yml in a folder and run:

user@localhost#docker-compose up -d

2nd point your browser to the <server address>:12999 port

This will start the application setup in your browser, the absolute mandatory step here is to register an administrator user with a password.

All others configuration steps now are possible with a system that will be already having a restrict access to an administrator user that will be able to configure new users and all the other technical details of the system.

Preparation from scratch of an Ubuntu server 24.04 system to run the docker-compose distribution of Gebo.ai

If you require to prepare a system from scratch to use Gebo.ai follow the instructions:

- Install Ubuntu server 24.04 LTS
- Adjust the character encoding of your system
    - Install language pack with:"**sudo apt-get install language-pack-en**" (or other language pack of your local language or company filesystems locales)
    - Run "**locale-gen en_US.UTF-8**"
    - Run "**sudo update-locale LANG=en_US.UTF-8**"
- Install and configure docker
    - Run "**sudo apt-get install docker.io**"
    - Run "**sudo apt-get install docker-compose**"
    - Run "**sudo service docker start**"
- Create gebo.ai user to run docker-compose system:
    - check docker group gid in /etc/group (110)
    - create gebo.ai user with group docker and bourn shell as a shell, home directory /home/gebo.ai with: "**sudo useradd gebo.ai -g 110 -m -s /bin/bash**"
    - create a gebo.ai use password with: "**sudo passwd gebo.ai**"
- Launching gebo.ai infrastructure
    - Download docker-compose.yml from https://gebo.ai/ in /home/gebo.ai
    - Switch user to gebo.ai "**sudo su – gebo.ai**"
    - Launching the infrastructure: "**sudo docker-compose up -d**"

Mounting windows shared filesystems (SMB CIFS) to be visible to your gebo.ai linux server:

If you plan to host gebo.ai in a linux server or in a linux server using docker-compose (see preceiding chapters), please ensurie that your linux service ha an extensive support for eventually present UTF-8 characters encoding in file names (view preceiding chapters).

To successfully let your windows shared filesystem being visible to your gebo.ai server let's mount them with the following commands:

#mount //<server name>/<share path> /mnt/windows-share -t cifs -o username=<username>,password=<password>,uid=<uid nr>,gid=<gid nr>,iocharset=utf8

Or

#mount //<server name>/<share path> /mnt/windows-share -t cifs -o username=<username>,password=<password>,uid=<uid nr>,gid=<gid nr>,iocharset=utf8,codepage=cp850

You can statically provide the reference to your company's windows shares to your /etc/fstab configuration file and configure those mounted volumes to be visible to your gebo.ai system.

Installing Gebo.ai from a preconfigured appliance

You can download a preconfigured virtual machine with all required software already installed

Download the appliance from: https://gebo.ai/downloads/

On the appliance accounts are predetermined, please change them to harden your installation.

Accounts informations are in accounts.txt file contained in downloadable appliances archives.

Installing Gebo.ai on premise or in your private/public cloud

This is the "go to" installation for companies who want to take the most out of being fully configurable with custom security settings.

Check on: https://gebo.ai/downloads/ latest Gebo.ai installation image for your system.

GitHub releases are on: https://github.com/geboai/Gebo.ai/releases/

Installation steps on your on custom installation:

Prerequisites: if you use a linux operating system ensure to install full UTF-8 support for your filesystems, otherwise accessing company shared Unicode filesystems will be nearly impossible.

1st install mongodb community edition https://www.mongodb.com/try/download/community

2nd install QDRANT https://qdrant.tech/documentation/guides/installation/#installation-options

3rd install NEO4J Graph database, is optional, you can disable it, but it enables Graphrag features with unprecedented documents retrieval performance. It is experimental in the version 1.0.0.0-rc1 but will be improved on following versions.

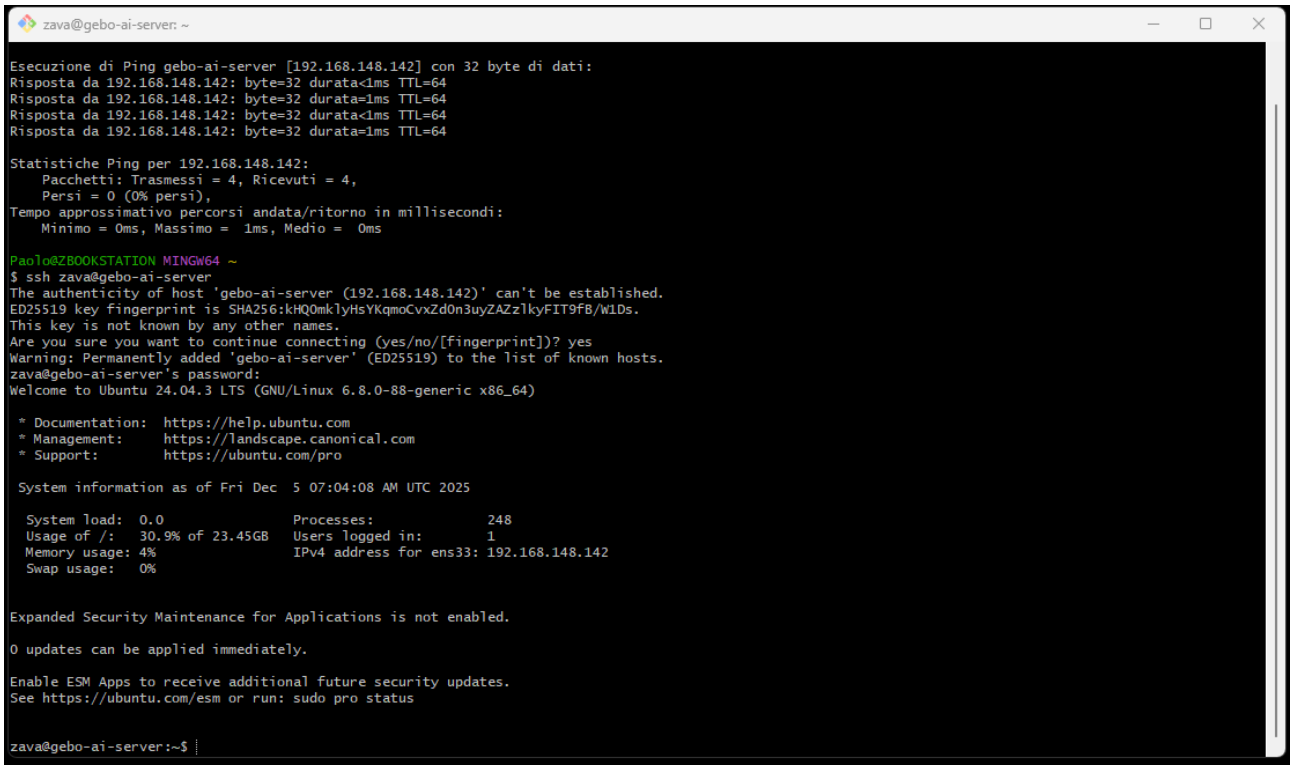
4rd install Gebo.ai from a binary distribution (or directly the spring boot bootable jar).

After software installation step

Go with your browser to: http:<your server address>:12999/ create your administrator account and start using/administering the software for all the company.

Linux Ubuntu and “Debian style” installation:

We’ll use ubuntu server 24.04.3, after the operating system installation, log in your server:



Prerequisites:

Adjust the character encoding of your system

- Install language pack with:”**sudo apt-get install language-pack-en**” (or other language pack of your local language or company filesystems locales)
- Run “**locale-gen en_US.UTF-8**”
- Run “**sudo update-locale LANG=en_US.UTF-8**”

Install a java jdk version 21 or upper:

Run:

sudo apt install java-common

Install java jdk 21 (for qdrant or to run gebo.ai directly from spring boot jar)

sudo apt install openjdk-21-jre-headless

To check your installation run:

#**java -version**

The correct answer is similar to

openjdk version "21.0.9" 2025-10-21

Run commands (responding Y as yes to interactive questions):

sudo apt-get install language-pack-en

#**sudo locale-gen en_US.UTF-8**

sudo update-locale LANG=en_US.UTF-8

This will enable general UTF-8 support, select your custom localization here will improve the adaptation to your Unicode company systems.

Install mongodb community edition

Go to: https://www.mongodb.com/try/download/community

Select your operating system.

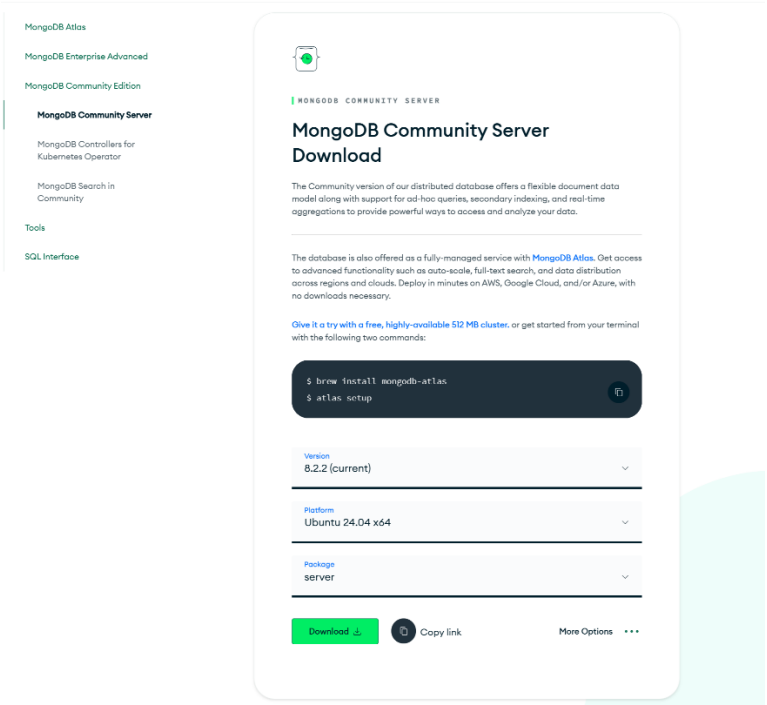


Figure 2 mongodb community download user interface

You can download directly in your server by running:

#**wget** **https://repo.mongodb.org/apt/ubuntu/dists/noble/mongodb-org/8.2/multiverse/binary-amd64/mongodb-org-server_8.2.2_amd64.deb**

After the download install mongodb running:

sudo apt install ./mongodb-org-server_8.2.2_amd64.deb

Let the service start with:

service mongod start

Now install utils and mongo administrative tools:

sudo apt-get update

sudo apt-get install -y gnupg curl

Install the mongo repository key and repository references

#**curl -fsSL https://www.mongodb.org/static/pgp/server-8.0.asc **

| sudo gpg -o /usr/share/keyrings/mongodb-server-8.0.gpg –dearmor

#**echo "deb [arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-8.0.gpg] https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 multiverse" **

| sudo tee /etc/apt/sources.list.d/mongodb-org-8.0.list

#**sudo apt-get update**

#sudo apt install mongodb-mongosh

Now create a mongo general administrator with its username and password

Access the mongosh shell:

#**mongosh "mongodb://localhost:27017/admin"**

Now on the mongosh shell:

Superuser creation (restricts the full amministrative features access)

> **use admin**

admin> **db.createUser({**

user: "mongoadmin",

pwd: "<put here your password>",

roles: [

{ role: "root", db: "admin" }

]

})

Create a restricted user capable of accessing/creating databases

admin> **db.createUser({**

user: "gebo-ai",

pwd: ""<put here your password>",

roles: [

{ role: "readWriteAnyDatabase", db: "admin" },

{ role: "dbAdminAnyDatabase", db: "admin" },

{ role: "userAdminAnyDatabase", db: "admin" }

]

})

Quit mongosh

admin>**quit**

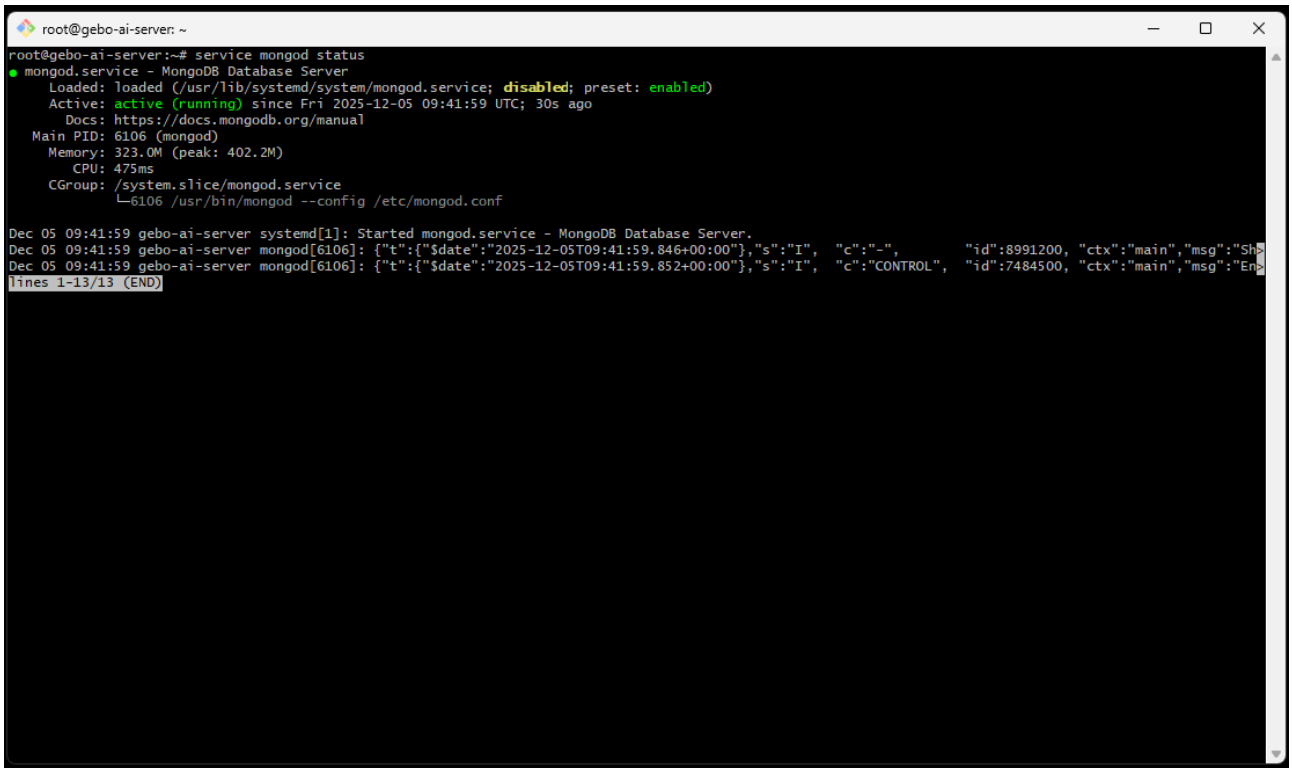
Startup the mongod service

#**service mongod start**

Check the mongod service with:

#service mongod status

The correct status is this:



Install QDRANT vector database

Get latest release from this base url: https://github.com/qdrant/qdrant/releases

Download binaries directly from **GitHub**, run:

#**wget https://github.com/qdrant/qdrant/releases/download/v1.16.2/qdrant_1.16.2-1_amd64.deb**

Install with:

#**sudo apt install ./qdrant_1.16.2-1_amd64.deb**

If this release does not install a proper systemd compliant service, create a qdrant user

#**sudo useradd --system --no-create-home --shell /usr/sbin/nologin qdrant**

Check that standard folders exists

#**sudo mkdir -p /var/lib/qdrant/storage **

**/var/lib/qdrant/snapshots **

**/var/lib/qdrant/static **

/etc/qdrant

Change ownership of all folders to qdrant user

#**sudo chown -R qdrant:qdrant /var/lib/qdrant /etc/qdrant**

Check if standard qdrant configuration file is in place, provide a configuration file with a proper api key to protect your accesses:

sudo nano /etc/qdrant/config.yaml

Provide the following entries to the .yml file (pay attention to yml formatting standard)

```
log_level: INFO
storage:
  storage_path: "/var/lib/qdrant/storage"
  snapshot_path: "/var/lib/qdrant/snapshots"
server:
  http_port: 6333
  grpc_port: 6334
# basic API key protection
service:
   api_key: "<enter here a uuid api key>"
```

now we create the systemd service:

sudo nano /etc/systemd/system/qdrant.service

Copy this code inside

```
[Unit]
Description=Qdrant Vector Database
After=network.target

[Service]
User=qdrant
Group=qdrant
WorkingDirectory=/var/lib/qdrant
ExecStart=/usr/bin/qdrant --config-path /etc/qdrant/config.yaml
Restart=on-failure
# Allow Qdrant to keep many files open (good for bigger workloads)
LimitNOFILE=10000

[Install]
WantedBy=multi-user.target
```

Save the code and enable the qdrant service running:

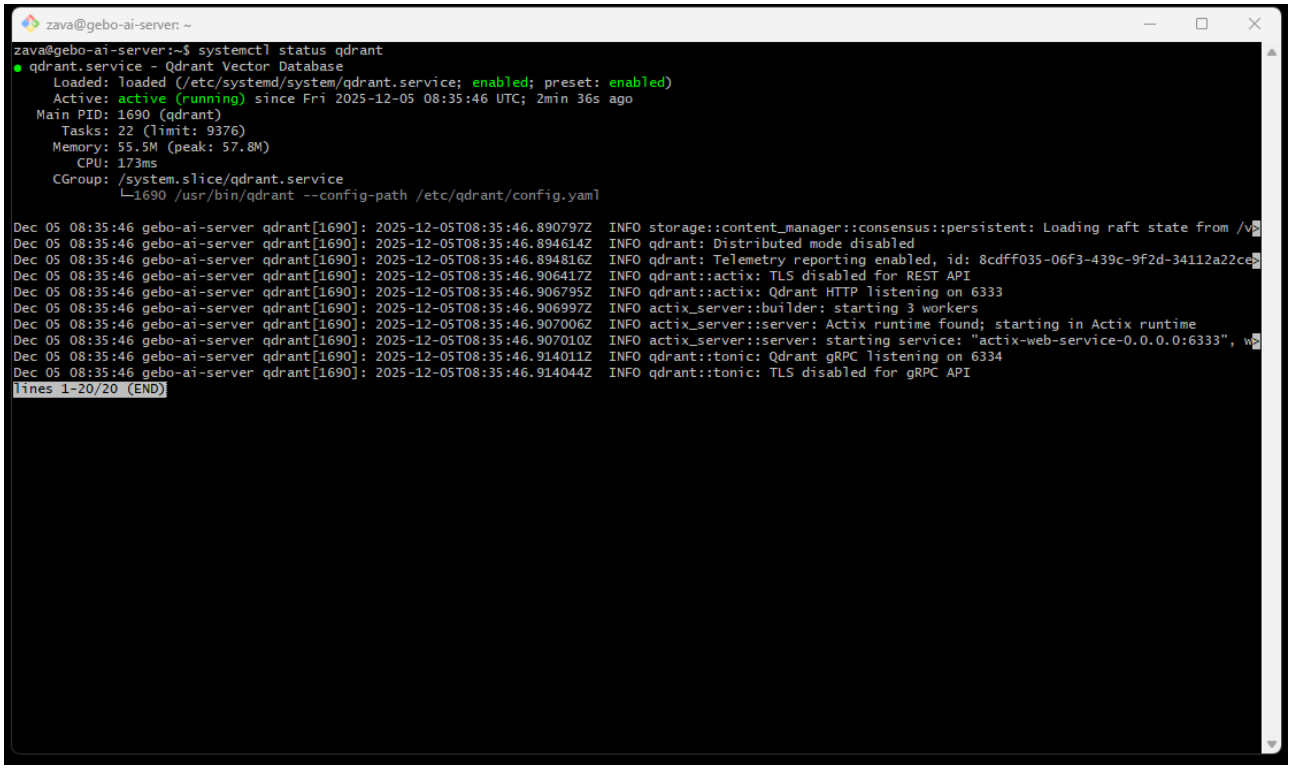
#**sudo systemctl daemon-reload**

#**sudo systemctl enable --now qdrant**

Check the service status:

#systemctl status qdrant

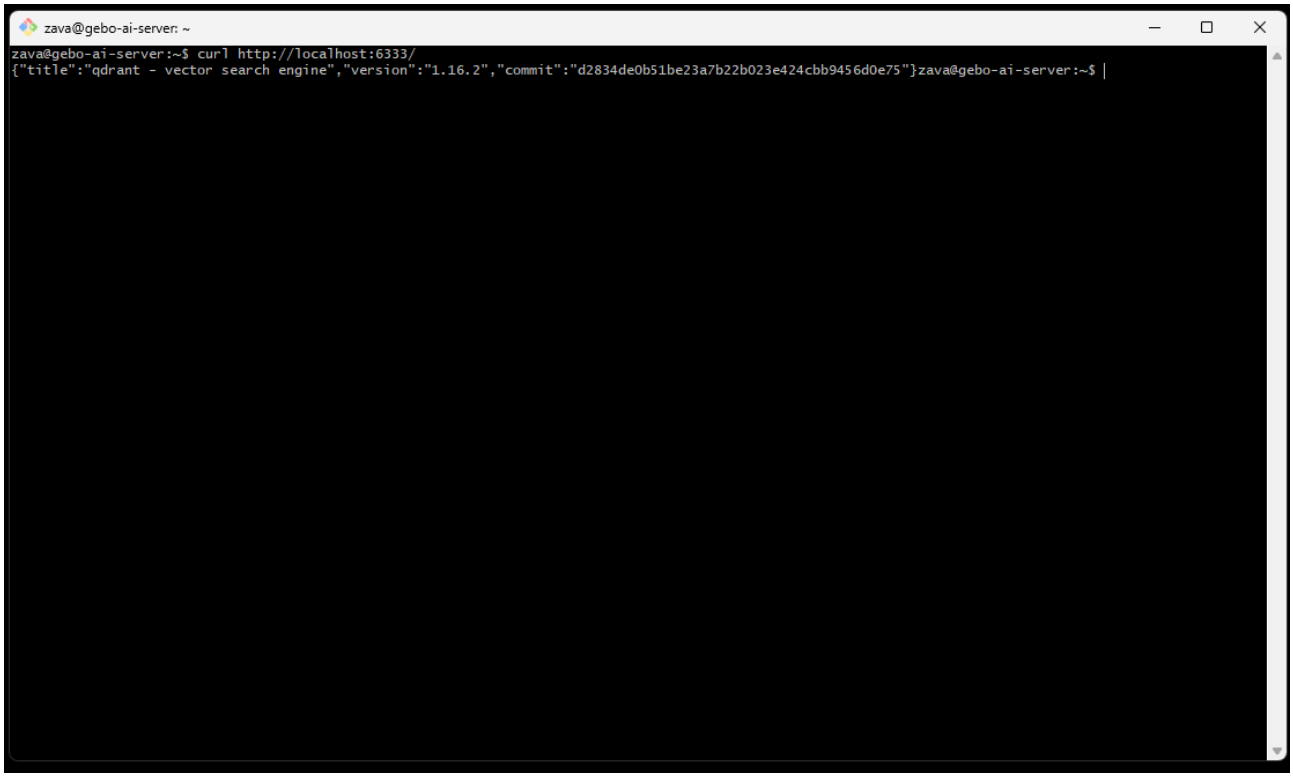
This is a correct status screen (if you don’t see this try restarting the qdrant service):



Check local connectivity:

#**curl** http://localhost:6333/

A correct response looks like this:



Refer to: https://qdrant.tech/documentation/guides/installation/

Install the Neo4J graph database (community)

References: https://neo4j.com/docs/operations-manual/current/installation/linux/debian/

Ubuntu/Debian references: https://debian.neo4j.com/

Run commands:

#**wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key add -**

echo 'deb https://debian.neo4j.com stable 5' | sudo tee /etc/apt/sources.list.d/neo4j.list

#**sudo apt-get update**

Now install neo4j:

#**sudo apt-get install neo4j**

Enable the neo4j service:

#sudo systemctl enable neo4j

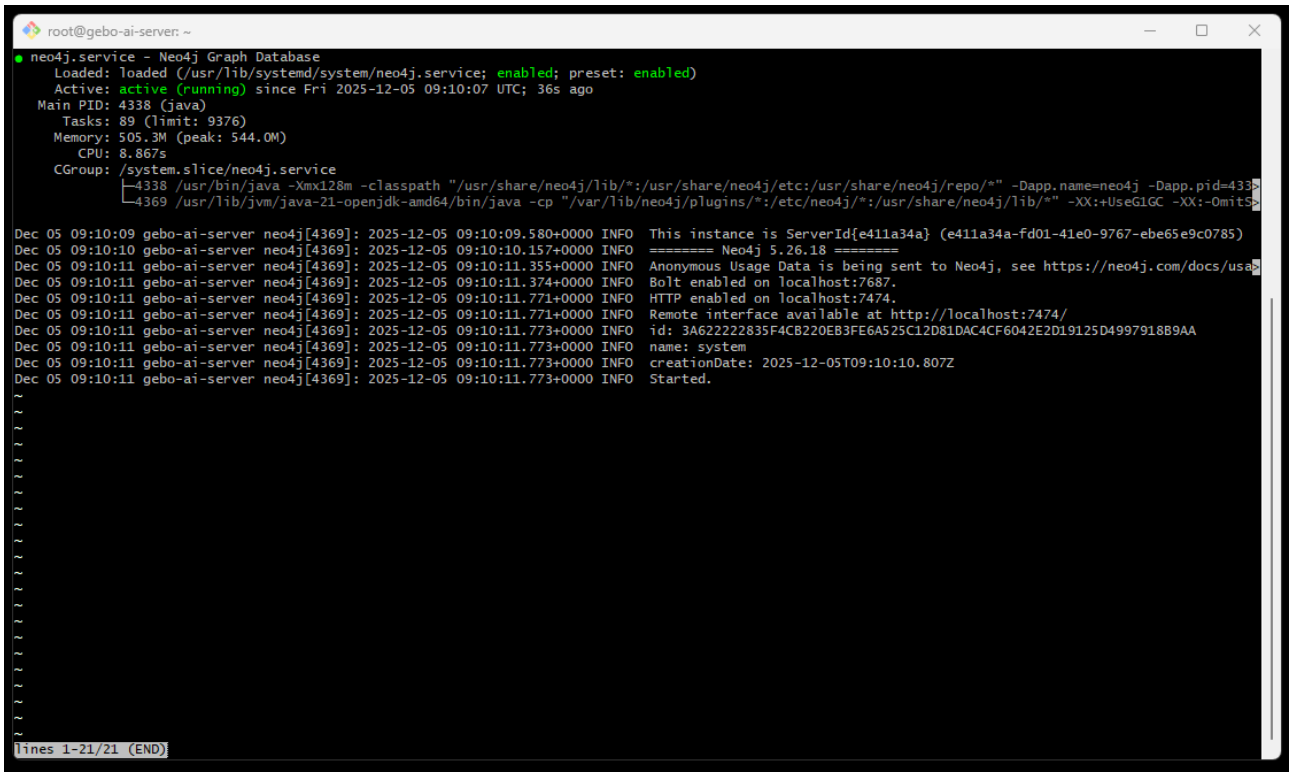
Start the service:

#service neo4j start

Check the status with:

#service neo4j status

A correct answer is like this:



Now neo4j is set with user: neo4j and password: neo4j, change the password with:

#**cypher-shell -u neo4j -p neo4j**

The software will ask you a new password.

If the system does not ask you the password let's run :

Neo4j@neo4j> **ALTER CURRENT USER SET PASSWORD FROM 'neo4j' TO '<neo4j password>;**

Gebo.ai software installation

Download actual gebo-ai version from the **https://gebo.ai/downloads/** or from https://github.com/geboai/Gebo.ai/releases/ web page (select .deb format).

Run:

#**sudo apt install ./gebo-ai_1.0.0.0-rc1_amd64.deb**

Now edit the configuration file for the gebo.ai application reporting there all your connectivity

Informations for mongo, qdrant and neo4j:

#**sudo nano /etc/gebo-ai/application.properties**

Now edit the file**:**

```
logging.level.root=INFO
logging.level.org.springframework=INFO
logging.level.org.springframework.web=INFO
logging.level.org.springframework.core=INFO
logging.level.org.springframework.core.codec=INFO

server.port=12999
server.compression.enabled=true
server.compression.mime-
types=text/html,text/xml,text/plain,text/css,text/javascript,application/javascript,appli
cation/json
server.compression.min-response-size=1024
server.http2.enabled=true
server.servlet.contextPath=/
spring.servlet.multipart.enabled=true
spring.servlet.multipart.maxFileSize=100MB
spring.servlet.multipart.maxRequestSize=100MB

#Neo4J connectivity configuration
ai.gebo.neo4j.enabled=true
spring.neo4j.uri=bolt://localhost:7687
spring.neo4j.authentication.username=neo4j
spring.neo4j.authentication.password=<write here your neo4j password>

#Eventual custom token secret for unique JWT in installation
#ai.gebo.security.auth.tokenSecret=04ca023b3...
#ai.gebo.security.auth.tokenExpirationMsec=120000
```

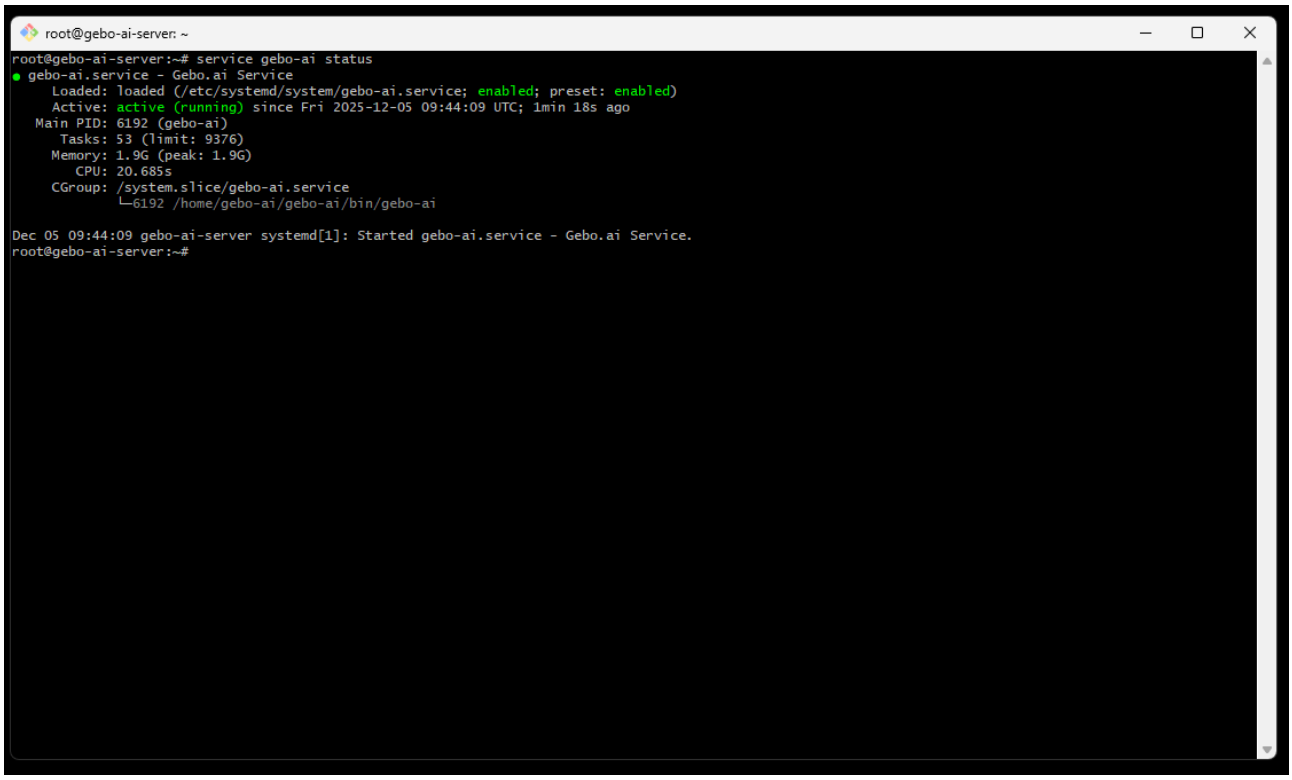
```
#ai.gebo.security.cors.allowedOrigins=http://localhost:12999,http://localhost:4200

#Qdrant connectivity configuration
ai.gebo.vectorstore.use=QDRANT
ai.gebo.vectorstore.qdrant.host=127.0.0.1
ai.gebo.vectorstore.qdrant.port=6334
ai.gebo.vectorstore.qdrant.tls=false
ai.gebo.vectorstore.qdrant.apiKey=<write here your qdrant api key>
#Work directory can be configured using UI
ai.gebo.config.setupConfiguresWorkdir=false

#MongoDB connectivity data
ai.gebo.mongodb.enabled=true
ai.gebo.mongodb.databaseName=gebo-ai
ai.gebo.mongodb.connectionString=mongodb://gebo-ai:<wite the mongo gebo-ai
password >@localhost:27017/gebo-ai?authSource=admin

#Allow user connecting configuring wich share(s) to configure from UI
ai.gebo.filesystem.allowFilesystemSharesUI=true
ai.gebo.filesystem.shares=
```

Startup the gebo-ai service:

#**sudo service gebo-ai start**

Check the gebo-ai service with:

#sudo service gebo-ai status



Enabling the firewall to let you reach the gebo.ai web port:

#**sudo ufw allow 12999/tcp**

Ensure your server time is ok.

For security reasons, some clients checked the server has correct date settings, be sure to run the system with correct date/time and timezone.

Ensure all services are booted at system startup

#**sudo systemctl enable qdrant**

#**sudo systemctl enable mongod**

#**sudo systemctl enable neo4j**

#**sudo systemctl enable gebo-ai**

Gebo.ai on premise linux standard paths and infos:

The software runs as user **gebo-ai** user.

Main configuration file → **/etc/gebo-ai/application.properties**

Its home directory is **/home/gebo-ai** with software startup scripts and binaries.

Logs are placed in **/var/log/gebo-ai.**

The main log is:

/var/log/gebo-ai/ai.gebo.monolithic.app.log, periodically compressed and archived by the system.

Java garbage collection log → **/var/log/gebo-ai/gc.log***

Redirected startup console logs: standard output → **/var/log/gebo-ai/service.log** standard error → **/var/log/gebo-ai/service.err**

Data paths (to be backupped properly)

Backup the software home directory→ **/home/gebo-ai**

Backup the software data directory→ **/var/gebo-ai/data**

After software installation step

Go with your browser to: http:<your server address>:12999/ create your administrator account and start using/administering the software for all the company.

Installing Gebo.ai on a windows server

Download actual gebo-ai version from the **https://gebo.ai/downloads/** or from https://github.com/geboai/Gebo.ai/releases/ web page (select .msi format).

Hybrid docker compose/windows installation

To install an hybrid solution having mongodb/qdrant/neo4j installed by docker composer and the gebo.ai application running on a windows installation.

Use the following docker-compose.yml to launch the 3 server applications:

```
version: '3.1'
services:
 # ======================
 # MongoDB
 # ======================
 # Bound only on loopback interface for security reasons
 mongo:
  image: mongo
  restart: always
  ports:
   - "127.0.0.1:27017:27017"
  environment:
```

```
    MONGO_INITDB_ROOT_USERNAME: <provide mongo user name>
    MONGO_INITDB_ROOT_PASSWORD: <provide mongo password>
  command: ["mongod", "--wiredTigerCacheSizeGB=1"]


# ======================
# Qdrant
# ======================
# Bound only on loopback interface for security reasons
qdrant:
  image: qdrant/qdrant
  restart: always
  ports:
    - "127.0.0.1:6333:6333"
    - "127.0.0.1:6334:6334"
  environment:
    QDRANT__SERVICE__API_KEY: <provide qdrant api key>
    QDRANT__STORAGE__OPTIMIZE_INDEXING: "true"
    QDRANT__STORAGE__RAM_DISK_SIZE: "256MB"
    QDRANT__STORAGE__MAX_SEGMENT_SIZE: "1000000"

neo4j:
  image: neo4j:5
  restart: always
  ports:
    - "127.0.0.1:7474:7474"
    - "127.0.0.1:7687:7687"
  environment:
    NEO4J_AUTH: "neo4j/neo4jmaster"
    NEO4J_server_memory_heap_initial__size: 1G
    NEO4J_server_memory_heap_max__size: 2G
    NEO4J_server_memory_pagecache_size: 512M
    NEO4JLABS_PLUGINS: '["apoc"]'
    NEO4J_apoc_export_file_enabled: "true"
    NEO4J_apoc_import_file_enabled: "true"
    NEO4J_apoc_import_file_use__neo4j__config: "true"
```

Launch the docker-compose in the folder you've created the docker-compose.yml file running:

#**docker-compose up -d**

After these services are up and running you install Gebo.ai using the gebo-ai msi installer,

Go to the c:\Program Files\GeboAI\app\instance\config, copy the example-application.properties to application.properties (in the same folder), edit the file to provide connectivity information(s) for all the 3 different server applications (mongo, qdrant, neo4j) and restart the gebo-ai service from the windows services application.

After software installation step

Go with your browser to: http:<your server address>:12999/ create your administrator account and start using/administering the software for all the company.

Installing MongoDB/Qdrant/Neo4J/Gebo.ai on the windows operating system.

This is actually the hardest installation to do, unfortunately most of the latest AI oriented software stack is not thought to run on windows but mostly on linux infrastructure or in docker infrastructure.

1st install actual version of MongoDB community edition → https://www.mongodb.com/try/download/community (select community version and windows operating system). Create a mongo login profile in mongosh for the gebo.ai application (see how in the linux ubuntu section).

2nd install QDRANT Vector database → https://github.com/qdrant/qdrant/releases/ (select the pc-windows-msvc*.zip version) . Create a qdrant api-key for the gebo.ai application. You can set a qdrant configuration coherent with the one in the linux ubuntu section. Unfortunately there are no QDRANT distributions running as a service, so it will run as a console application.

3rd install Neo4J graph database → https://neo4j.com/deployment-center/?ref=subscription#community (select the community edition), install the application and create/test an account protected by password in the neo4j system.

After these services are up and running you install Gebo.ai using the gebo-ai msi installer,

Go to the c:\Program Files\GeboAI\app\instance\config, copy the example-application.properties to application.properties (in the same folder), edit the file to provide connectivity information(s) for all the 3 different server applications (mongo, qdrant, neo4j) and restart the gebo-ai service from the windows services application.

After software installation step

Go with your browser to: http:<your server address>:12999/ create your administrator account and start using/administering the software for all the company.

“Monolithic version” base installation components & integration with 3rd party systems.

The gebo.ai monolithic version software is an application that runs on linux/unix/windows and on containerized systems using java virtual machines 17 or higher (preferable the 21 or 22 versions) runtimes.

Minimum software stack for the installation

- Jvm 21 or 22 (already shipped in bundle with the geboai/gebo.ai docker container).
- A reachable **MongoDB** installation (also a common edition is OK)
- A reachable Vector database, preferably **QDRANT** that the more tested one.
- A reachable **Neo4J** graph database (optional but suggested) for Graphrag and Knowledge extraction

MongoDB is the only part of the infrastructure that must be configured dealing directly with the software configuration files, almost all the other integrations or required infrastructure parts can be configured from the administrative UI.

Connectivity with cloud generative AI providers or local generative AI server

The software can use the following cloud or on-premise generative AI providers/servers, you require at least to configure 2 default models for the software:

- a “chat large language model” (like GPT4o/O1/Claude 3.X/Llama 3.X or others)→ this is the chatbot that will perform generative AI session elaboration and output.
- an “embedding model” like OpenAi text-embedding-3-large or NVIDIA/NV-Embed-v2 or others. This model is responsible for transforming document contents on their semantic numerical vector representation to support data retrieval.

Supported providers (and some supported models examples)

From networking/technical point of view those of the following cloud providers chosen must be configured to have full accessibility from the server where Gebo.ai is installed on. In other words if you chose to use OpenAI or Xai or other suppliers, at least https networking with those domains have to be permitted on the company firewalls.

All the cloud or on premise generative AI providers or servers can be directly configured using the software administrative UI, inserting directly in the user interface api key or other company subscription credentials.

The credentials management uses cryptographic algorithms to let those access informations being securely stored and unreadable even if the database is hacked (go to the security considerations sections).

- OpenAi, chat models (suggested, best multilanguage)

- Antrophic, chat models: Claude
- MistralAI, chat models: Mistral Large/Codestral/Mistral mini embedding models: Mistral Embed (English only).
- Xai, chat models: Grok, embedding models: embedding-beta
- Groq provider, chat models: Meta Llama 3.1 3.2 3.3 8b/70b
- Regolo.ai provider (Italian player developing interesting offers)
- Nvidia Nim/Cloud platform (mostly for models trials purposes), chat models: https://build.nvidia.com/explore/reasoning embedding models: https://build.nvidia.com/explore/retrieval
- Supported soon: Google AI infrastructures.
- Other infrastructures compatible with OpenAI API can be tested with the generical OpenAI API connection profile.

Supported local/lan generative AI server

- Ollama: mostly suggested for models trials for its easy setup, does not scale well to an enterprise use.
- vLLM: the "go to" solution for on premise artificial intelligence, scales to multiple devices/servers taking the most from actual Nvidia/Amd hardware.
- All other on premise applications providing full OpenAI compatible REST api.

Connectivity with company used on premise or cloud systems

1st connectivity with various "content systems" or "file sharing systems"

The software has an administrative area where users with administrative rights can configure integrations with all the Cloud or "On premise" content management systems or control version systems or files/documents sharing systems.

2nd composing knowledge bases

Administrative users will composes huge knowledge bases adding documents chosen from all the available CMS/shared filesystems/versioning systems to be available via retrieve augmented generation while Gebo.ai users interact with chat bots.

The server where Gebo.ai is installed must have access to all of those network resources.

All the content management services/version management services/sharing filesystem services/database services providers or servers can be directly configured using the software administrative UI, inserting directly in the user interface alla required credentials.

The credentials management uses cryptographic algorithms to let those access informations being securely stored and unreadable even if the database is hacked (go to the security considerations sections).

Gebo.ai detailed configuration

The application(s) distributed in the docker environment with an application.yml configuration that already works in the context where the provided docker-compose.yml is used.

The internal path on the container is /opt/gebo.ai/config/application.yml, if necessary it can be extracted from the standard container running:

docker cp <container name>:/opt/gebo.ai/config/application.yml ./application.yml

or it can be edited directly inside the container.

For on premise “non docker” installations there is a shipped application.properties file under /etc/gebo-ai/ in linux or in c:\Program Files/GeboAI/app/instance/config in windows operating system.

All the following chapters mention configuration keys and values that already exist in this file or that can be added to change the software behavior.

Be careful when handling the yml file format to use a software with proper syntax management like visual code with the yml format plugin. This format requires proper formatting to be accepted by the application.

Use an application.properties format if not familiar with the yaml format, you can simply use this translator: https://mageddo.com/tools/yaml-converter to convert the application.yml to an application.properties file and use this in place of the first once removed.

GEBO_HOME environment variable or system property

When the application is started it requires a GEBO_HOME environment variable (already set in the docker container to /opt/gebo.ai/home or by installers in “on premise” installations) .

This variable must point to an existing folder that is usable to store minimal application internal configurations. It is convenient to backup this folder periodically.

To ensure coherency on persistence using the docker image we suggest to mount this folder to an existing folder on the host system and backup it periodically.

GEBO_WORK_DIRECTORY environment variable or system property or interactive admin UI setting.

The application requires the setting of a “work directory” used to store its working files, It is convenient to backup this folder periodically.

The work directory can be handle in 2 different way:

- Work directory interactively set at gebo.ai interactive setup by the administrative user using the administrator user interface, behavior configured when the configuration key:

ai.gebo.config.setupConfiguresWorkdir is set to true. In this state the work directory path is configured in the MongoDB system.

- Work directory configured via configuration file, behavior set with: ***ai.gebo.config.setupConfiguresWorkdir*** is set to false with **GEBO_WORK_DIRECTORY** environment or system property pointing to the chosen work directory (that is required to be an existing and empty folder).

In the containerized docker distribution the system work with this 2nd type of configuration with work directory being set to /opt/gebo.ai/work folder.

To ensure coherency on persistence using the docker image we suggest to mount this folder to an existing folder on the host system and backup it periodically.

MongoDB connectivity settings

This base configuration is mandatory and required for the application to start, the configuration is in the following application.yml configuration keys (or if you choose to use application.properties format simply change ":" with "=" in the following keys configuration)

ai.gebo.mongodb.enabled: true

ai.gebo.mongodb.databaseName: gebo-ai

ai.gebo.mongodb.connectionString: mongodb://<user>:<pwd>@localhost:27017/gebo-ai?authSource=admin

The connection string grammar reference is standard for MongoDB configurations and is documented here: https://www.mongodb.com/docs/manual/reference/connection-string/

If you are using the docker-compose launch system the provided docker-compose.yml file sets up MongoDB already coherent with these settings shipped with the provided configuration.

MongoDB is used as NOSQL database for configurations, application data and much more, so it must be considered part of the "persistent storage class" of the application, managed taking into account security and under backup management.

QDRANT or other vector database configuration

This base configuration can be provided from the application.yml configuration file (or if you choose to use application.properties format simply change ":" with "=" in the following keys configuration), however if is not provided a clean new installation started up without it and let the administrator user configure it interactively in the administration UI "setup" section.

Configuration keys:

ai.gebo.vectorstore.use: QDRANT

ai.gebo.vectorstore.qdrant.host: 127.0.0.1

ai.gebo.vectorstore.qdrant.port: 6334

ai.gebo.vectorstore.qdrant.tls: false

ai.gebo.vectorstore.qdrant.apiKey: <api key value>

The vector database is a mandatory component of the system when we have configured at least one large language model with “embedding” capabilities, Gebo.ai create separate vector databases connection for each “embedding model” and configured Knowledge base, so once configured and your company started to use the retrieve augmented generation system it has to be considered as a part of the “persistent storage class” in the application stack keeping it secure and with managed backup.

Neo4J database configuration

The graph database configuration can be provided from the application.yml configuration file (or if you choose to use application.properties format simply change “:” with “=” in the following keys configuration).

Configuration keys:

ai.gebo.neo4j.enabled: <true to enable, false to disable>

spring.neo4j.uri: bolt://localhost:7687

spring.neo4j.authentication.username: <neo4j user name>

spring.neo4j.authentication.password: secret

Company shared filesystem

Company shared filesystems paths, due to their delicate handling, must first being configured to be reachable from the Gebo.ai software.

For the docker compose or docker easyinstall image:

If you are using the provided docker-compose.yml file to install/launch the system you need to mount your filesystems under the **/opt/gebo.ai/shares** on the linux docker installation using the standard documer (or integrating it in docker-compose.yml) “volume mounting” commands.