Getting started with Auto-GPT

Learn More | GitHub repository

Getting started with Auto-GPT

In this guide, we will learn how to get started playing with Auto-GPT on your local computer or a server.

If you have any issues installing this locally or if you want to contribute to the project, please visit the official GitHub repository. You can open a new ticket if you encounter any installation or other problems.

Requirements

You can choose between these environments:

- VSCode + devcontainer: It has been configured in the. dev container folder and can be used directly
- Docker
- Python 3.10 or later (instructions: for Windows)

Optional

Memory backend

- Pinecone
- Milvus
- Redis
- Weaviate

If you want the AI to speak, you will need to get the ElevenLabs Key.

OpenAI API Keys Configuration

Obtain your OpenAI API key from: https://platform.openai.com/account/api-keys.

To use the OpenAI API key for Auto-GPT, you need to have billing set up (paid account).

You can set up a paid account at https://platform.openai.com/account/billing/overview.

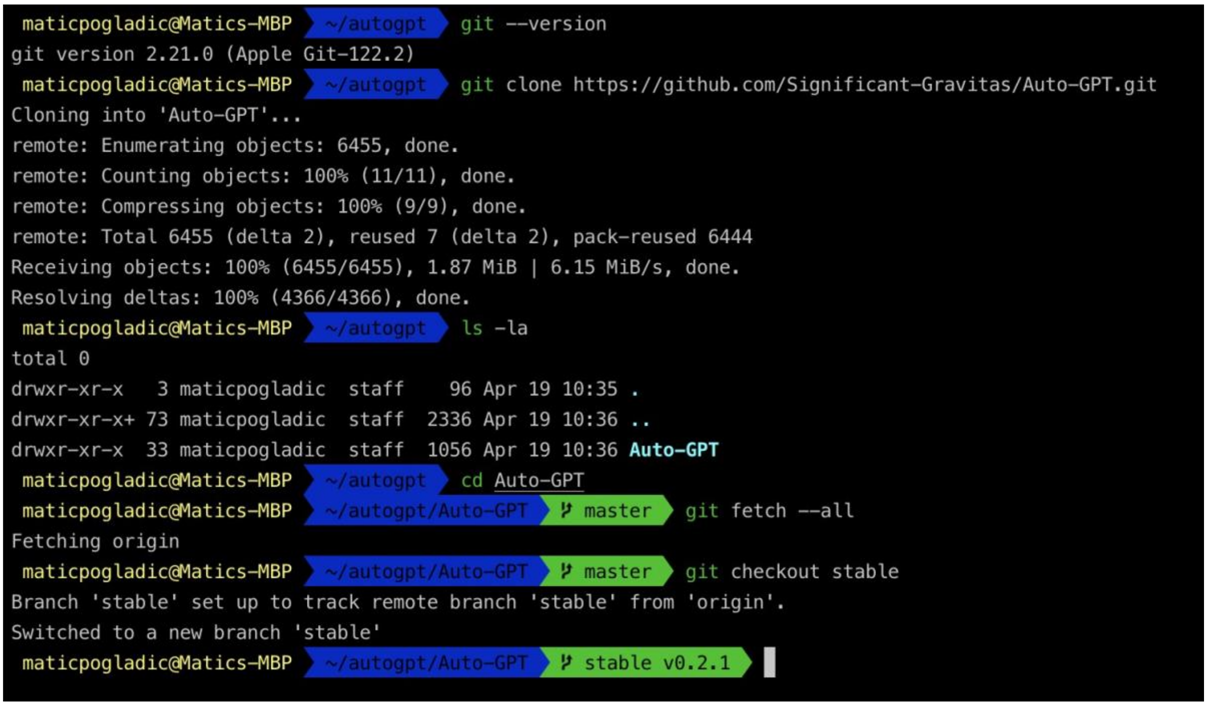
Installation

To install Auto-GPT, follow these steps:

1. Make sure you have all the requirements listed above, if not, install/get them

To execute the following commands, open a CMD, Bash, or Powershell window by navigating to a folder on your computer and typing `CMD` in the folder path at the top, then press enter.

2. Clone the repository: For this step, you need Git installed. Alternatively, you can download the latest stable release (`Source code (zip)`, bottom of the page).
   `git clone https://github.com/Significant-Gravitas/Auto-GPT.git`
3. Navigate to the directory where the repository was downloaded
   `cd Auto-GPT`
4. Make sure to use `stable` branch and not `master` because it might be broken often.



```
 maticpogladic@Matics-MBP  ~/autogpt  git --version
git version 2.21.0 (Apple Git-122.2)
 maticpogladic@Matics-MBP  ~/autogpt  git clone https://github.com/Significant-Gravitas/Auto-GPT.git
Cloning into 'Auto-GPT'...
remote: Enumerating objects: 6455, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 6455 (delta 2), reused 7 (delta 2), pack-reused 6444
Receiving objects: 100% (6455/6455), 1.87 MiB | 6.15 MiB/s, done.
Resolving deltas: 100% (4366/4366), done.
 maticpogladic@Matics-MBP  ~/autogpt  ls -la
total 0
drwxr-xr-x   3 maticpogladic  staff    96 Apr 19 10:35 .
drwxr-xr-x+ 73 maticpogladic  staff  2336 Apr 19 10:36 ..
drwxr-xr-x  33 maticpogladic  staff  1056 Apr 19 10:36 Auto-GPT
 maticpogladic@Matics-MBP  ~/autogpt  cd Auto-GPT
 maticpogladic@Matics-MBP  ~/autogpt/Auto-GPT  master  git fetch --all
Fetching origin
 maticpogladic@Matics-MBP  ~/autogpt/Auto-GPT  master  git checkout stable
Branch 'stable' set up to track remote branch 'stable' from 'origin'.
Switched to a new branch 'stable'
 maticpogladic@Matics-MBP  ~/autogpt/Auto-GPT  stable v0.2.1 
```

5. Install the required dependencies.

   `pip install -r requirements.txt`

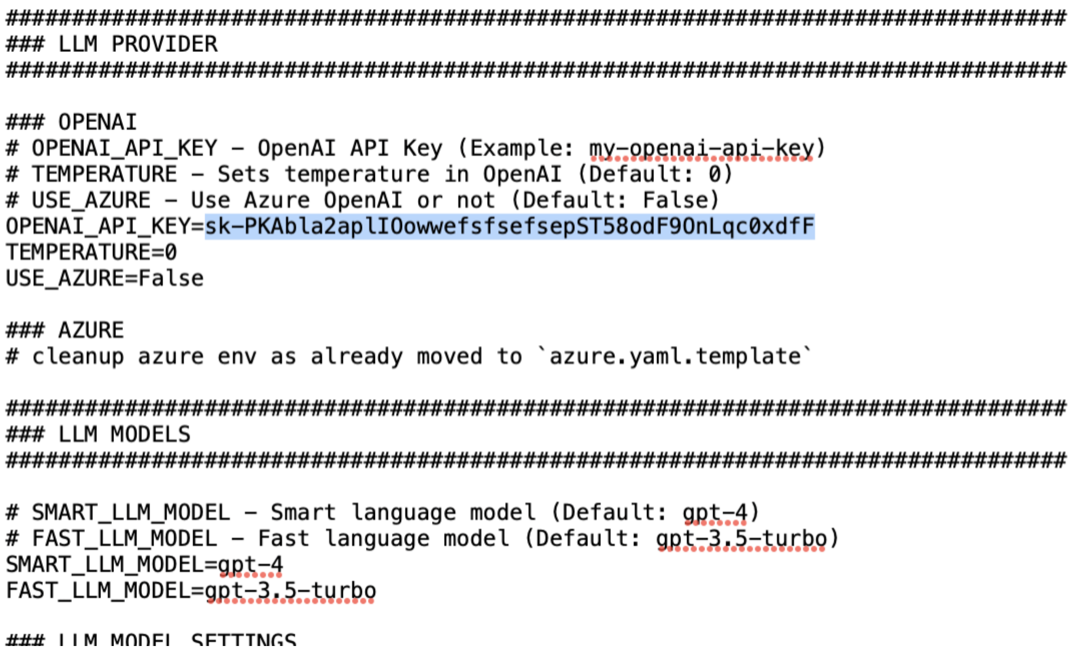
   or

   `pip3 install -r requirements.txt`

6. Configure Auto-GPT
   ○ Locate the file named `.env.template` in the main `/Auto-GPT` folder.

- Create a copy of this file, called `.env` by removing the `template` extension. The easiest way is to do this in a command prompt/terminal window `cp .env.template .env`.
- Open the `.env` file in a text editor (Files starting with a dot might be hidden by your Operating System).
- Find the line that says `OPENAI_API_KEY=`.
- After the `"="`, enter your unique OpenAI API Key (without any quotes or spaces).
- Enter any other API keys or Tokens for services you would like to utilize.
- Save and close the `.env` file.

Here is a part of the `.env` file:



```
################################################################################
### LLM PROVIDER
################################################################################

### OPENAI
# OPENAI_API_KEY - OpenAI API Key (Example: my-openai-api-key)
# TEMPERATURE - Sets temperature in OpenAI (Default: 0)
# USE_AZURE - Use Azure OpenAI or not (Default: False)
OPENAI_API_KEY=sk-PKAbla2aplIOowwefsfsefsepST58odF9OnLqc0xdfF
TEMPERATURE=0
USE_AZURE=False

### AZURE
# cleanup azure env as already moved to `azure.yaml.template`

################################################################################
### LLM MODELS
################################################################################

# SMART_LLM_MODEL - Smart language model (Default: gpt-4)
# FAST_LLM_MODEL - Fast language model (Default: gpt-3.5-turbo)
SMART_LLM_MODEL=gpt-4
FAST_LLM_MODEL=gpt-3.5-turbo

### LLM MODEL SETTINGS
```

7. You have properly configured the API Keys for your project. You can now update some other parts of the `.env` file:
    - Obtain your ElevenLabs API key from: https://elevenlabs.io. You can view your xi-api-key using the "Profile" tab on the website.
    - If you want to use GPT on an Azure instance, set `USE_AZURE` to `True` and then follow these steps:
        - Rename `azure.yaml.template` to `azure.yaml` and provide the relevant `azure_api_base`, `azure_api_version` and all the deployment IDs for the relevant models in the `azure_model_map` section:
            - `fast_llm_model_deployment_id` - your gpt-3.5-turbo or gpt-4 deployment ID
            - `smart_llm_model_deployment_id` - your gpt-4 deployment ID
            - `embedding_model_deployment_id` - your text-embedding-ada-002 v2 deployment ID

- Please specify all of these values as double-quoted strings:
  ```
  # Replace string in angled brackets (<>) to your own ID
  azure_model_map: fast_llm_model_deployment_id:
  "<my-fast-llm-deployment-id>"
  ```
- Details can be found here: https://pypi.org/project/openai/ in the `Microsoft Azure Endpoints` section and here: https://learn.microsoft.com/en-us/azure/cognitive-services/openai/tutorials/embeddings?tabs=command-line for the embedding model.

Usage

Run the Auto-GPT Python module in your terminal:

```
# On Linux of Mac:
./run.sh start
# On Windows:
.\run.bat
```

Running with --help after .\run.bat lists all the possible command line arguments you can pass.

After each action, choose from options to authorize command(s), exit the program, or provide feedback to the AI.

1. Authorize a single command, enter “y”.
2. Authorize a series of N continuous commands, enter “y -N”.
3. Exit the program, and enter “n”.

Logs

Activity and error logs are located in the “./output/logs” folder.

To print out debug logs:

```
python -m autogpt --debug
```

Docker

You can also build this into a docker image and run it:

```
docker build -t autogpt .
docker run -it --env-file=./.env -v
$PWD/auto_gpt_workspace:/app/auto_gpt_workspace autogpt
```

Or if you have `docker-compose`:

```
docker-compose run --build --rm auto-gpt
```

You can pass extra arguments, for instance, running with `--gpt3only` and `--continuous` mode:

```
docker run -it --env-file=./.env -v
$PWD/auto_gpt_workspace:/app/auto_gpt_workspace autogpt --gpt3only
--continuous

docker-compose run --build --rm auto-gpt --gpt3only --continuous
```

Command Line Arguments

Here are some common arguments you can use when running Auto-GPT:

Replace anything in angled brackets (<>) to a value you want to specify

- View all available command line arguments
  `python -m autogpt --help`
- Run Auto-GPT with a different AI Settings file
  `python -m autogpt --ai-settings <filename>`
- Specify a memory backend
  `python -m autogpt --use-memory  <memory-backend>`

NOTE: There are shorthands for some of these flags, for example `-m` for `--use-memory`. Use `python -m autogpt --help` for more information

Speech Mode

Use this to use TTS *(Text-to-Speech)* for Auto-GPT

```
python -m autogpt --speak
```

List of IDs with names from eleven labs, you can use the name or ID

- Rachel : 21m00Tcm4TlvDq8ikWAM

- Domi : AZnzlk1XvdvUeBnXmlld
- Bella : EXAVITQu4vr4xnSDxMaL
- Antoni : ErXwobaYiN019PkySvjV
- Elli : MF3mGyEYCl7XYWbV9V6O
- Josh : TxGEqnHWrfWFTfGW9XjX
- Arnold : VR6AewLTigWG4xSOukaG
- Adam : pNInz6obpgDQGcFmaJgB
- Sam : yoZ06aMxZJJ28mfd3POQ

Google API Keys Configuration

This section is optional, use the official Google API if you are having issues with error 429 when running a Google search. To use the `google_official_search` command, you need to set up your Google API keys in your environment variables.

1. Go to the Google Cloud Console.
2. If you don't already have an account, create one and log in.
3. Create a new project by clicking on the "Select a Project" dropdown at the top of the page and clicking "New Project". Give it a name and click "Create".
4. Go to the APIs & Services Dashboard and click "Enable APIs and Services". Search for "Custom Search API" and click on it, then click "Enable".
5. Go to the Credentials page and click "Create Credentials". Choose "API Key".
6. Copy the API key and set it as an environment variable named `GOOGLE_API_KEY` on your machine. See setting up environment variables below.
7. Enable the Custom Search API on your project. (Might need to wait a few minutes to propagate)
8. Go to the Custom Search Engine page and click "Add".
9. Set up your search engine by following the prompts. You can choose to search the entire web or specific sites.
10. Once you've created your search engine, click on "Control Panel" and then "Basics". Copy the "Search engine ID" and set it as an environment variable named `CUSTOM_SEARCH_ENGINE_ID` on your machine. See setting up environment variables below.

Remember that your free daily custom search quota allows only up to 100 searches. To increase this limit, you need to assign a billing account to the project to profit from up to 10K daily searches.

Setting up environment variables

For Windows Users:

```
setx GOOGLE_API_KEY "YOUR_GOOGLE_API_KEY"
setx CUSTOM_SEARCH_ENGINE_ID "YOUR_CUSTOM_SEARCH_ENGINE_ID"
```

For macOS and Linux users:

```
export GOOGLE_API_KEY="YOUR_GOOGLE_API_KEY"
export CUSTOM_SEARCH_ENGINE_ID="YOUR_CUSTOM_SEARCH_ENGINE_ID"
```

Setting Your Cache Type

By default, Auto-GPT is going to use LocalCache instead of Redis or Pinecone.

To switch to either, change the `MEMORY_BACKEND` env variable to the value that you want:

- `local` (default) uses a local JSON cache file
- `pinecone` uses the Pinecone.io account you configured in your ENV settings
- `redis` will use the Redis cache that you configured
- `milvus` will use the milvus cache that you configured
- `weaviate` will use the weaviate cache that you configured

Memory Backend Setup

Redis Setup

CAUTION
This is not intended to be publicly accessible and lacks security measures. Therefore, avoid exposing Redis to the internet without a password or at all

1. Install Docker (or Docker Desktop on Windows)
2. Launch Redis container
   ```
   docker run -d --name redis-stack-server -p 6379:6379
   redis/redis-stack-server:latest
   ```
   See https://hub.docker.com/r/redis/redis-stack-server for setting a password and additional configuration.

   Set the following settings in `.env`
   Replace PASSWORD in angled brackets (<>)
   ```
   MEMORY_BACKEND=redis
   REDIS_HOST=localhost
   REDIS_PORT=6379
   REDIS_PASSWORD=<PASSWORD>
   ```
3. You can optionally set `WIPE_REDIS_ON_START=False` to persist memory stored in Redis.

You can specify the memory index for redis using the following:

```
MEMORY_INDEX=<WHATEVER>
```

Pinecone API Key Setup

Pinecone enables the storage of vast amounts of vector-based memory, allowing for only relevant memories to be loaded for the agent at any given time.

1. Go to pinecone and make an account if you don't already have one.
2. Choose the `Starter` plan to avoid being charged.
3. Find your API key and region under the default project in the left sidebar.

In the `.env` file set:

- `PINECONE_API_KEY`
- `PINECONE_ENV` (example: *"us-east4-gcp"*)
- `MEMORY_BACKEND=pinecone`

Alternatively, you can set them from the command line (advanced):

For Windows Users:

```
setx PINECONE_API_KEY "<YOUR_PINECONE_API_KEY>"
setx PINECONE_ENV "<YOUR_PINECONE_REGION>" # e.g: "us-east4-gcp"
setx MEMORY_BACKEND "pinecone"
```

For macOS and Linux users:

```
export PINECONE_API_KEY="<YOUR_PINECONE_API_KEY>"
export PINECONE_ENV="<YOUR_PINECONE_REGION>" # e.g: "us-east4-gcp"
export MEMORY_BACKEND="pinecone"
```

Milvus Setup

Milvus is an open-source, highly scalable vector database to store huge amounts of vector-based memory and provides fast relevant search.

- setup Milvus database, keep your pymilvus version and Milvus version the same to avoid compatible issues.
  - setup by open source Install Milvus
  - or setup by Zilliz Cloud
- set `MILVUS_ADDR` in `.env` to your Milvus address `host:ip`.
- set `MEMORY_BACKEND` in `.env` to `milvus` to enable Milvus as backend.

Optional:

- set `MILVUS_COLLECTION` in `.env` to change Milvus collection name as you want, `autogpt` is the default name.

Weaviate Setup

Weaviate is an open-source vector database. It allows to store data objects and vector embeddings from ML-models and scales seamlessly to billion of data objects. An instance of Weaviate can be created locally (using Docker), on Kubernetes or using Weaviate Cloud Services. Although still experimental, Embedded Weaviate is supported which allows the Auto-GPT process itself to start a Weaviate instance. To enable it, set `USE_WEAVIATE_EMBEDDED` to `True` and make sure you `pip install "weaviate-client>=3.15.4"`.

Install the Weaviate client

Install the Weaviate client before usage.

```
$ pip install weaviate-client
```

Setting up environment variables

In your `.env` file set the following:

```
MEMORY_BACKEND=weaviate

WEAVIATE_HOST="127.0.0.1" # the IP or domain of the running Weaviate instance
WEAVIATE_PORT="8080"
WEAVIATE_PROTOCOL="http"
WEAVIATE_USERNAME="your username"
WEAVIATE_PASSWORD="your password"
WEAVIATE_API_KEY="your weaviate API key if you have one"
WEAVIATE_EMBEDDED_PATH="/home/me/.local/share/weaviate" # this is optional and
   indicates where the data should be persisted when running an embedded
   instance
USE_WEAVIATE_EMBEDDED=False # set to True to run Embedded Weaviate
MEMORY_INDEX="Autogpt" # name of the index to create for the application
```

View Memory Usage

View memory usage by using the `--debug` flag :)

Memory pre-seeding

Memory pre-seeding allows you to ingest files into memory and pre-seed it before running Auto-GPT.

```
# python data_ingestion.py -h
usage: data_ingestion.py [-h] (--file FILE | --dir DIR) [--init] [--overlap
   OVERLAP] [--max_length MAX_LENGTH]

Ingest a file or a directory with multiple files into memory. Make sure to set
   your .env before running this script.
```

```
options:
  -h, --help            show this help message and exit
  --file FILE           The file to ingest.
  --dir DIR             The directory containing the files to ingest.
  --init                Init the memory and wipe its content (default:
   False)
  --overlap OVERLAP     The overlap size between chunks when ingesting files
   (default: 200)
  --max_length MAX_LENGTH  The max_length of each chunk when ingesting files
   (default: 4000)

# python data_ingestion.py --dir DataFolder --init --overlap 100 --max_length
   2000
```

In the example above, the script initializes the memory, and ingests all files within the `Auto-Gpt/autogpt/auto_gpt_workspace/DataFolder` directory into memory with an overlap between chunks of 100 and a maximum length of each chunk of 2000.

Note that you can also use the `--file` argument to ingest a single file into memory and that data_ingestion.py will only ingest files within the `/auto_gpt_workspace` directory.

The DIR path is relative to the auto_gpt_workspace directory, so `python data_ingestion.py --dir . --init` will ingest everything in `auto_gpt_workspace` directory.

You can adjust the `max_length` and overlap parameters to fine-tune the way the documents are presented to the AI when it "recalls" that memory:

- Adjusting the overlap value allows the AI to access more contextual information from each chunk when recalling information, but will result in more chunks being created and therefore increase memory backend usage and OpenAI API requests.
- Reducing the `max_length` value will create more chunks, which can save prompt tokens by allowing for more message history in the context, but will also increase the number of chunks.
- Increasing the `max_length` value will provide the AI with more contextual information from each chunk, reducing the number of chunks created and saving on OpenAI API requests. However, this may also use more prompt tokens and decrease the overall context available to the AI.

Memory pre-seeding is a technique for improving AI accuracy by ingesting relevant data into its memory. Chunks of data are split and added to memory, allowing the AI to access them quickly and generate more accurate responses. It's useful for large datasets or when specific information needs to be accessed quickly. Examples include ingesting API or GitHub documentation before running Auto-GPT.

If you use Redis as your memory, make sure to run Auto-GPT with the `WIPE_REDIS_ON_START=False` in your `.env` file.

For another memory backend, we currently forcefully wipe the memory when starting Auto-GPT. To ingest data with that memory backend, you can call the `data_ingestion.py` script anytime during an Auto-GPT run.

Memories will be available to the AI immediately as they are ingested, even if ingested while Auto-GPT is running.

Continuous Mode

Run the AI without user authorization, 100% automated. Continuous mode is NOT recommended. It is potentially dangerous and may cause your AI to run forever or carry out actions you would not usually authorize. Use at your own risk.

1. Run the `autogpt` python module in your terminal:
   ```
   python -m autogpt --speak --continuous
   ```
2. To exit the program, press Ctrl + C

GPT3.5 ONLY Mode

If you don't have access to the GPT4 API, this mode will allow you to use Auto-GPT!

```
python -m autogpt --speak --gpt3only
```

It is recommended to use a virtual machine for tasks that require high-security measures to prevent any potential harm to the main computer's system and data.

Image Generation

By default, Auto-GPT uses DALL-e for image generation. To use Stable Diffusion, a Hugging Face API Token is required.

Once you have a token, set these variables in your `.env`:

```
IMAGE_PROVIDER=sd
HUGGINGFACE_API_TOKEN="YOUR_HUGGINGFACE_API_TOKEN"
```

Selenium

```
sudo Xvfb :10 -ac -screen 0 1024x768x24 & DISPLAY=:10 <YOUR_CLIENT>
```

Limitations

This experiment aims to showcase the potential of GPT-4 but comes with some limitations:

1. Not a polished application or product, just an experiment
2. May not perform well in complex, real-world business scenarios. In fact, if it actually does, please share your results!
3. Quite expensive to run, so set and monitor your API key limits with OpenAI!

Disclaimer

Disclaimer This project, Auto-GPT, is an experimental application and is provided "as-is" without any warranty, express or implied. By using this software, you agree to assume all risks associated with its use, including but not limited to data loss, system failure, or any other issues that may arise.

The developers and contributors of this project do not accept any responsibility or liability for any losses, damages, or other consequences that may occur as a result of using this software. You are solely responsible for any decisions and actions taken based on the information provided by Auto-GPT.

Please note that the use of the GPT-4 language model can be expensive due to its token usage. By utilizing this project, you acknowledge that you are responsible for monitoring and managing your own token usage and the associated costs. It is highly recommended to check your OpenAI API usage regularly and set up any necessary limits or alerts to prevent unexpected charges.

As an autonomous experiment, Auto-GPT may generate content or take actions that are not in line with real-world business practices or legal requirements. It is your responsibility to ensure that any actions or decisions made based on the output of this software comply with all applicable laws, regulations, and ethical standards. The developers and contributors of this project shall not be held responsible for any consequences arising from the use of this software.

By using Auto-GPT, you agree to indemnify, defend, and hold harmless the developers, contributors, and any affiliated parties from and against any and all claims, damages, losses, liabilities, costs, and expenses (including reasonable attorneys' fees) arising from your use of this software or your violation of these terms.