Search for articles...

Best Practices for API Key Safety

Updated over 4 months ago

1. Always use a unique API key for each team member on your account.

An API key is a unique code that identifies your requests to the API. Your API key is intended to be used by you. The sharing of API keys is against the Terms of Use.

As you begin experimenting, you may want to expand API access to your team. OpenAI does not support the sharing of API keys. Please invite new members to your account from the Members page and they will quickly receive their own unique key upon sign-in. You can assign permissions to individual API keys as well.

2. Never deploy your key in client-side environments like browsers or mobile apps.

Exposing your OpenAI API key in client-side environments like browsers or mobile apps allows malicious users to take that key and make requests on your behalf – which may lead to unexpected charges or compromise of certain account data. Requests should *always* be routed through your own backend server where you can keep your API key secure.

3. Never commit your key to your repository

Committing an API key to source code is a common vector for credential compromise. For those with public repositories, this is a common way that you can unknowingly share your key with the internet. Private repositories are more secure, but a data breach can also result in your keys being leaked. For these reasons we strongly recommend the use of the environment variables as a proactive key safety measure.

4. Use Environment Variables in place of your API key

An environment variable is a variable that is set on your operating system, rather than within your application. It consists of a name and value.We recommend that you set the name of the variable to OPENAI_API_KEY. By keeping this variable name consistent across your team, you can commit and share your code without the risk of exposing your API key.

Windows Set-up

Option 1: Set your 'OPENAI_API_KEY' Environment Variable via the cmd prompt

Run the following in the cmd prompt, replacing <yourkey> with your API key:

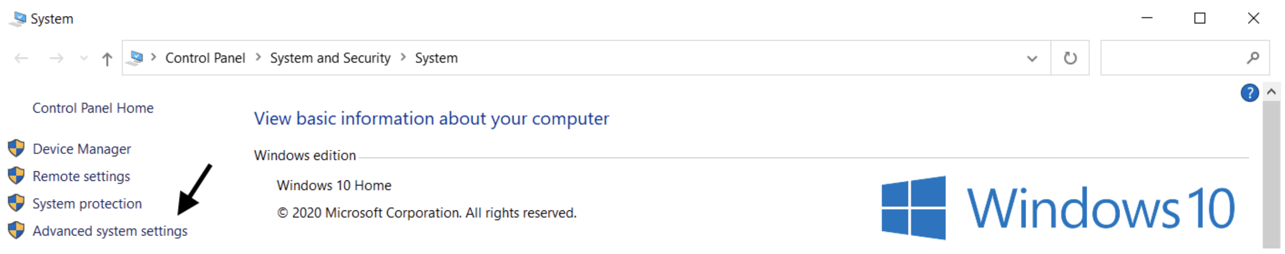
```
setx OPENAI_API_KEY "<yourkey>"
```

This will apply to future cmd prompt window, so you will need to open a new one to use that variable with curl. You can validate that this variable has been set by opening a new cmd prompt window and typing in

```
echo %OPENAI_API_KEY%
```

Option 2: Set your 'OPENAI_API_KEY' Environment Variable through the Control Panel

1. Open **System** properties and select **Advanced system settings**



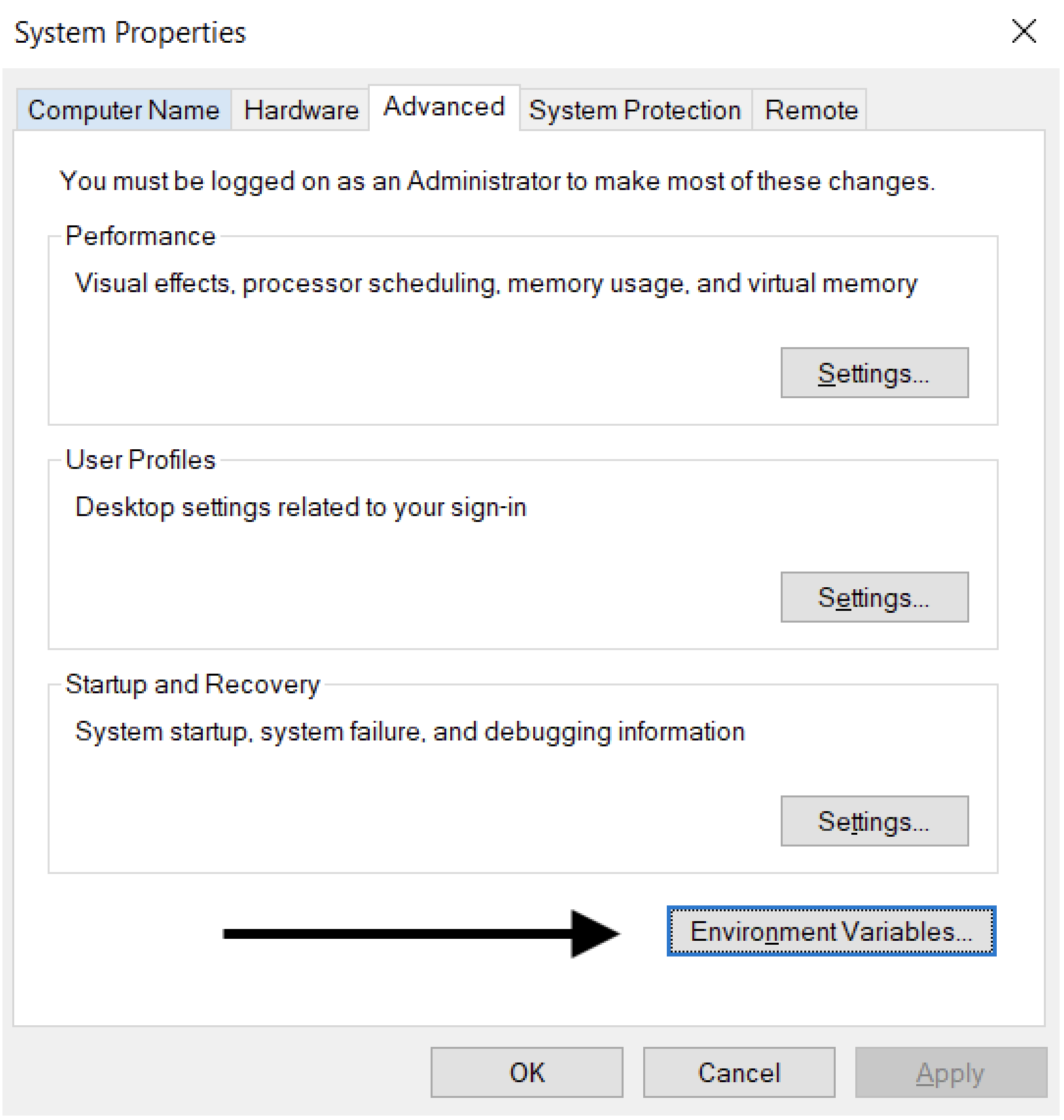
View basic information about your computer

Windows edition

Windows 10 Home

© 2020 Microsoft Corporation. All rights reserved.

2. Select **Environment Variables...**


System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK | Cancel | Apply

3. Select **New...** from the User variables section(top). Add your name/key value pair, replacing <yourkey> with your API key.

```
Variable name: OPENAI_API_KEY
Variable value: <yourkey>
```

Linux / MacOS Set-up

Option 1: Set your 'OPENAI_API_KEY' Environment Variable using zsh

1. Run the following command in your terminal, replacing yourkey with your API key.

```
echo "export OPENAI_API_KEY='yourkey'" >> ~/.zshrc
```

2. Update the shell with the new variable:

```
source ~/.zshrc
```

3. Confirm that you have set your environment variable using the following command.

```
echo $OPENAI_API_KEY
```

The value of your API key will be the resulting output.

Option 2: Set your ‘OPENAI_API_KEY’ Environment Variable using bash
Follow the directions in Option 1, replacing **.zshrc** with **.bash_profile.**

You’re all set! You can now reference the key in curl or load it in your Python:

```
import os
import openai

openai.api_key = os.environ["OPENAI_API_KEY"]
```

5. Use a Key Management Service

There are a variety of products available for safely managing secret API keys. These tools allow you to control access to your keys and improve your overall data security. In the event of a data breach to your application, your key(s) would not be compromised, as they would be encrypted and managed in a completely separate location.

For teams deploying their applications into production, we recommend you consider one of these services.

6. Monitor your account usage and rotate your keys when needed

A compromised API key allows a person to gain access to your account quota, without your consent. This can result in data loss, unexpected charges, a depletion of your monthly quota, and interruption in your API access.

Your teams’ Usage can be tracked via the Usage page. If you ever have concerns about misuse there are a few actions you can take to protect your account:

- Review your usage to see if it aligns with your team’s work. For users belonging to multiple organizations (ex. corporate and personal), make sure the user has enabled tracking and set their default organization for usage and tracking.
- If you believe your key has been leaked, rotate your key immediately from the API Keys page. For customers with applications in production, you will need to update

your key values accordingly.
- Contact us through help.openai.com for further investigating.

Related Articles

- Best practices for prompt engineering with the OpenAI API ›
- How can I add my API client to the Community Libraries page? ›
- How can I keep my OpenAI accounts secure? ›
- Understanding your API Usage ›
- Phone number requirement for new API keys ›

Did this answer your question?

😞 😐 😃