How to set up and configure APIs

This guide records every step of **Phase 1—API setup and configuration—** including how we obtain and store credentials and set environment variables so anyone can reproduce or review the work.

---

[1] Slack:

URL: https://api.slack.com/

Steps:

1. Create a Slack app via the apps page
2. Choose “From Scratch” option
3. Enter the App name and choose the workspace
4. Navigate to **“Socket Mode”** on the left panel and enable it. It will ask for **app-level token** for the name choose any and for the scope add the following [app_configurations:write] a window with the token will pop up save the token as we will need it in the future.
5. Navigate to **“Socket Mode”** on the left panel under the **“Features”** section. Navigate to **Scopes** section then **Bot Token Scopes** and add the following scopes [app_mentions:read, assistant:write, channels:history, channels:read, chat:write, chat:write.customize, files:read, files:write, groups:history, groups:read, im:history, im:read, mpim:read, users.profile:read, users:read, users:read.email, im:write, mpim:write]
6. Navigate to “**OAuth & Permissions**” on the top of the page then “**OAuth Tokens**” and click on “**Install [App Name]**”. copy and save the **Bot User OAuth Token** as we will need it in the future.
7. Navigate to **“Event Subscriptions”** on the left panel under the **“Features”** section and switch the button form off to on. Choose “**Subscribe to bot events**” and add the following bot

user events [app_mention, message.channles, message.im] and click save changes.

8. Navigate to **“App Home”** on the left panel under the **“Features”** and scroll down to the end of the page and check the following box **”Allow users to send Slash commands and messages from the chat tab”**
9. Navigate to **“Slash Commands”** on the left panel under the **“Features”** and **Create a new command (deep-search)** it’s important to use the name **deep-search**
10. Open Slack workspace and create a new channel
11. In the prompt message section write **“/add”** and choose **add apps to this channel**. Choose your app.
12. Save the channel name as we will need in the further steps.

Tokens and needed info:

1. app-level token
2. Bot User OAuth Token
3. Channel name

Limitation:

Link in docs**:** https://api.slack.com/apis/rate-limits

1. only 10 concurrent WebSocket connections per app
2. **Soc**ket-Mode apps can’t be listed in the public Slack Marketplace
3. Web API calls still count against the normal rate limits, and new non-Marketplace apps are throttled to just 1 conversations.history request per minute (limit ≤ 15)

Will limitations affect the performance:

As the current level of operation, no it will be good to go, the problem will occur if we decide to make massive horizontal scaling.

[2] Groq:

URL: https://console.groq.com/keys

Steps:

1. Create an account and log in
2. Create an API key by clicking in Create API Key
3. Enter the name of your choice [e.g. Praxilab-groq-API-key]
4. Save the API key as it will be needed for further steps

Tokens and needed info:

1. Groq API key

Limitation:

Link in docs: https://console.groq.com/docs/rate-limits

How Groq meters usage:

- **RPM / RPD:** Requests per minute & per day
- **TPM / TPD:** Tokens per minute & per day

RPM / RPD / TPM / TPD change from model to another (use the lookup table in the docs for more detailed info)

Another limitation is the context windows. It changes from model to model to another as well. It is the fixed max measured in tokens on the combined length of all input sequences (system, user, tool messages, etc.) plus the tokens the model is allowed to generate in its response.

[2] Gemini:

URL: https://aistudio.google.com/

Steps:

1. Create an account and log in
2. Create an API key by clicking in + Create API Key
3. Enter the name of your choice [e.g. Praxilab-gemini-API-key]
4. Save the API key as it will be needed for further steps

Tokens and needed info:

1. Gemini API key

Limitation:

Use the following URL to know more about the Rate limits
https://ai.google.dev/gemini-api/docs/rate-limits