



Ask PanDA

Paul Nilsson

January 22, 2025

TIM @ Stony Brook



Introduction

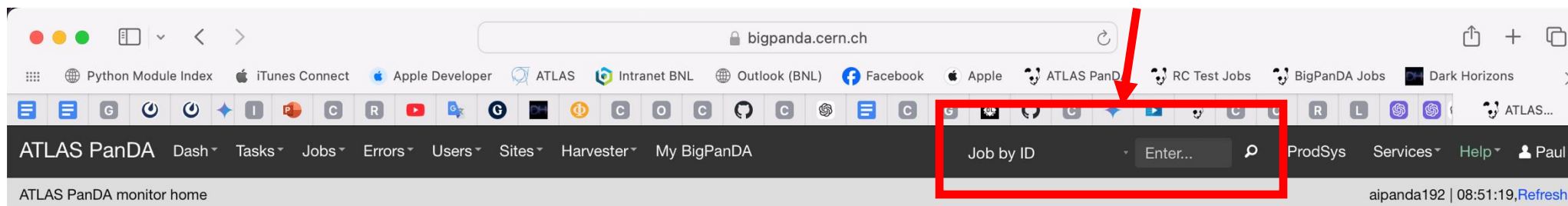
- PanDA (e.g.) is a complex beast
 - We all struggle with finding answers to questions that are outside of our own expertise
 - A significant amount of time is spent repeating answering questions that are often already answered in the documentation
- We also all know, that AI is now mature enough to help with these problems
- Enter “Ask PanDA”

Initial Ideas (1/4)

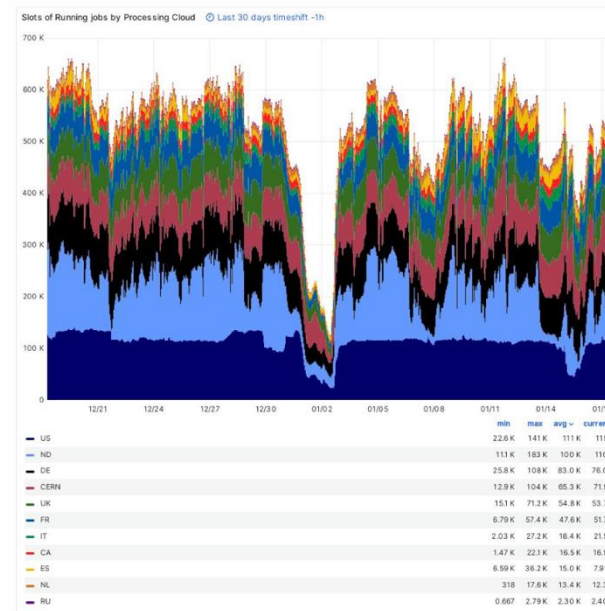
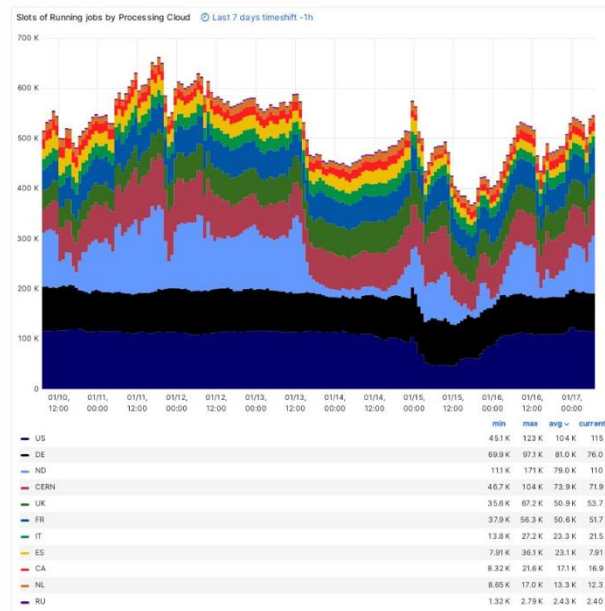
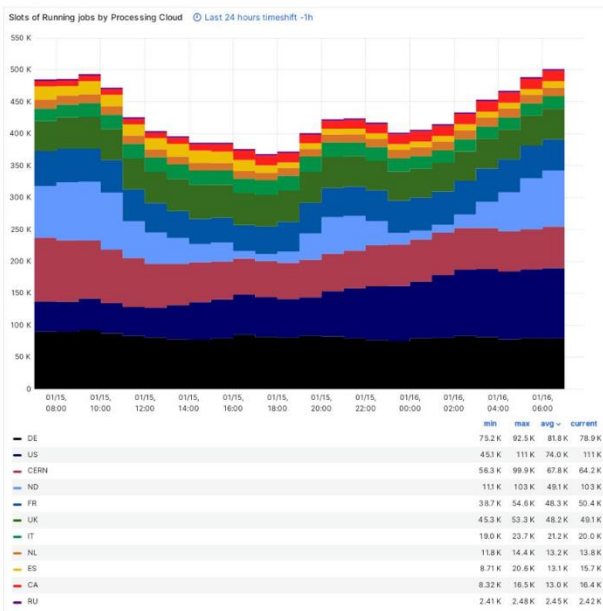
- Use AI to answer questions about the PanDA documentation
 - No AI is better than the information it is trained on
 - Trick is to use well written, relevant and up-to-date documentation along with a proper prompt template (only answer relevant questions)
- Provide access via standard entry place, i.e. the PanDA monitor
 - Reuse job id (etc) search field for free form prompt field and display answer in main monitor window area
 - Parse input to keep current functionality
 - If e.g. “Job by ID” is selected (default) and user enters “12345678”, it should display job page as before

Initial Ideas (2/4)

Move and update this area



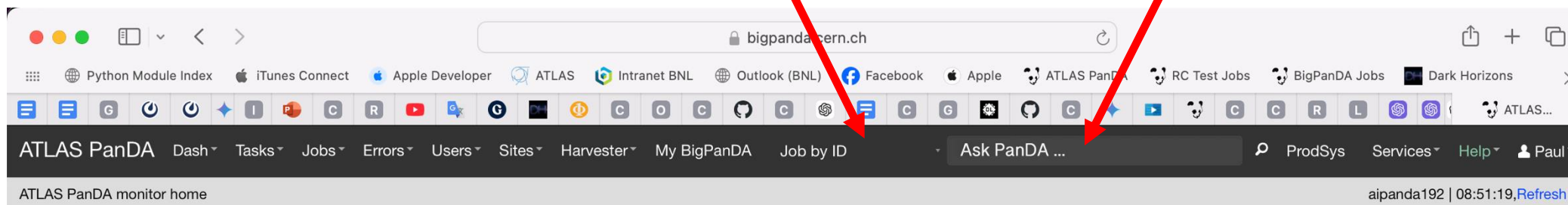
Global concurrent running job core counts, all sites, all job types, by cloud, last 1, 7, 30 days (new Monit)



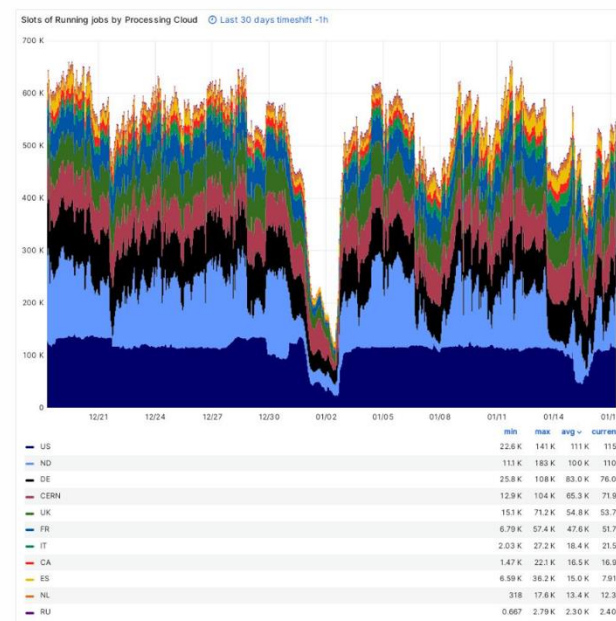
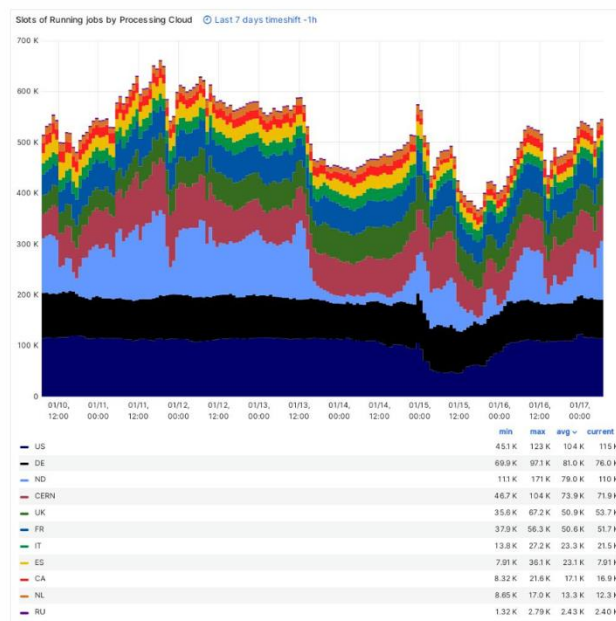
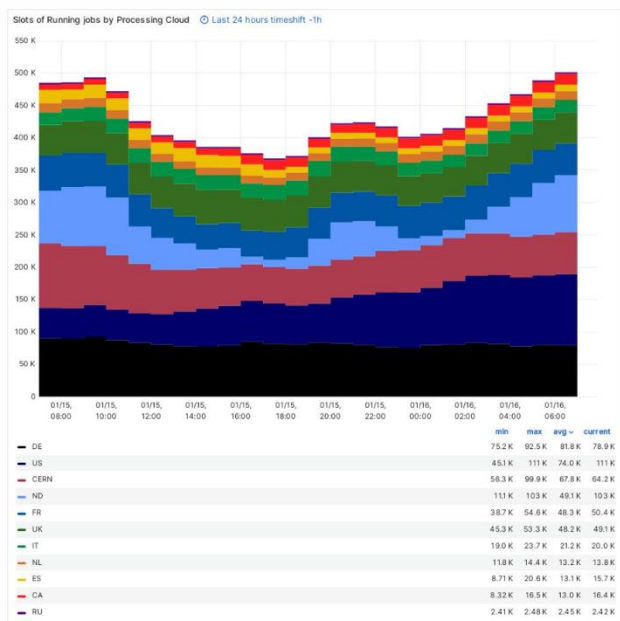
Initial Ideas (3/4)

Same drop list

Make this field as big as possible



Global concurrent running job core counts, all sites, all job types, by cloud, last 1, 7, 30 days (new Monit)



Initial Ideas (4/4)

- Alternative: “Ask PanDA” button is perhaps better
 - Complete dialogue will be in the main window
 - This also has the advantage that more of the input can be made visible on the screen (old search field is small)



Ask PanDA

Current Implementation

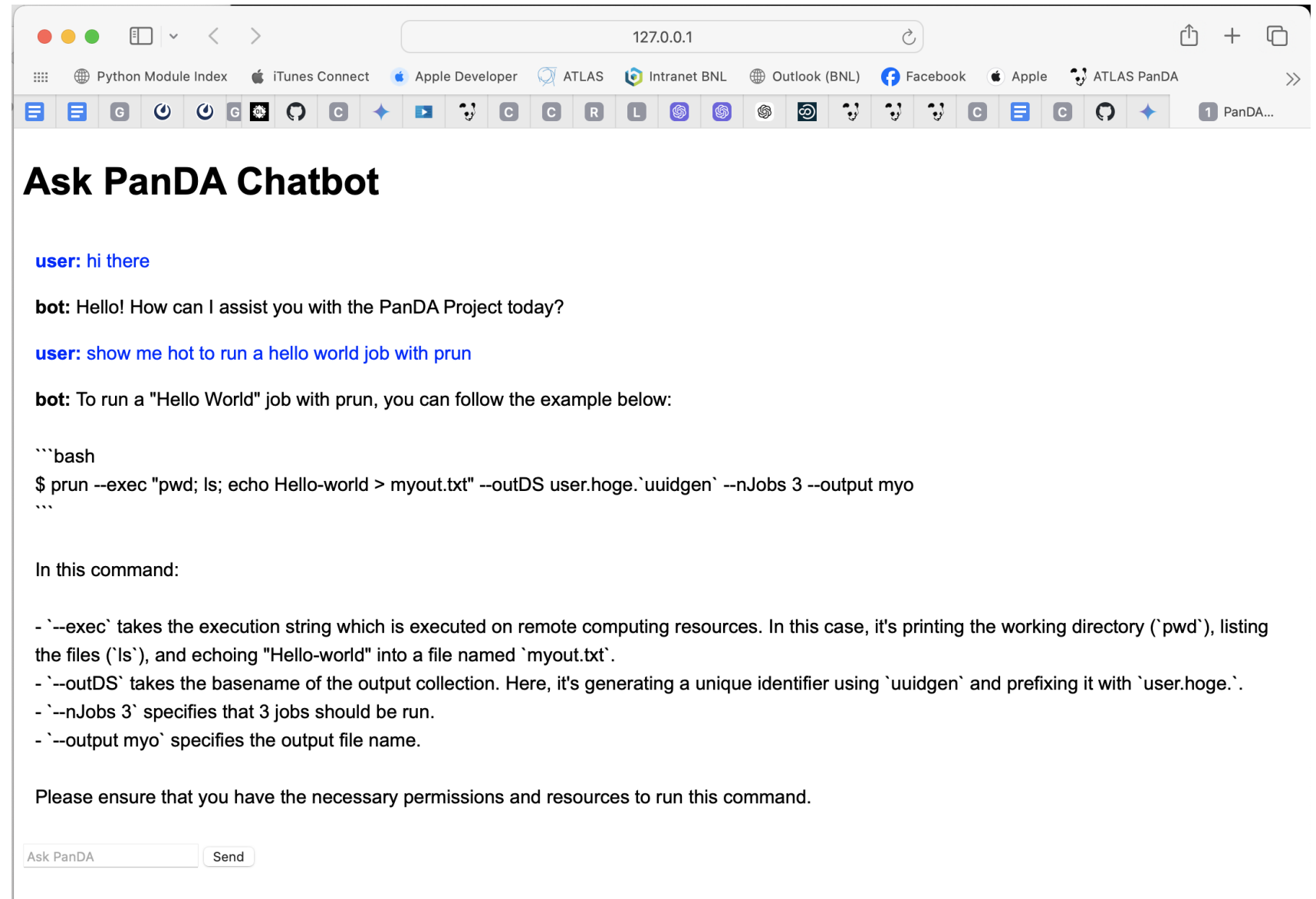
- PanDA documentation (readthedocs) was exported as pdf, then converted to plain text
- A python script using OpenAI LangChain modules
 - For testing purposes flask is currently used for creating simple web interface
- A proper prompt template is used

```
panda_expert_template = """
You are Dr PanDA, the expert who knows almost everything about the PanDA Project.
Your expertise is exclusively in providing information and advice about anything related to the PanDA Project.
This includes any general PanDA Project-related queries.
If the user asks about panda, it is understood that they mean PanDA and not the animal.
If a question is not about the PanDA project, respond with, "I can't assist you with that, sorry!", unless it is about CERN, BNL.
Context: {context}
Question: {question}
Answer:
"""
```

- Problem: Can only send chunks of 15kB to OpenAI
 - Script therefore divides the text in chunks and once it gets user input, it tries to find the most likely chunk (using OpenAI embedding model) before sending it off to the ChatGPT model
- Tried ChatGPT models
 - gpt-3.5 (fast but not so good answers, sometimes cut in the middle of a sentence – but is very cheap)
 - gpt-4 (much better, more expensive of course)

Example 1

- How to run a hello world job with prun
- It also remembers previous questions, i.e. one can continue the topic with follow up questions



Ask PanDA Chatbot

user: hi there

bot: Hello! How can I assist you with the PanDA Project today?

user: show me hot to run a hello world job with prun

bot: To run a "Hello World" job with prun, you can follow the example below:

```
```bash
$ prun --exec "pwd; ls; echo Hello-world > myout.txt" --outDS user.hoge.`uuidgen` --nJobs 3 --output myo
```
```

In this command:

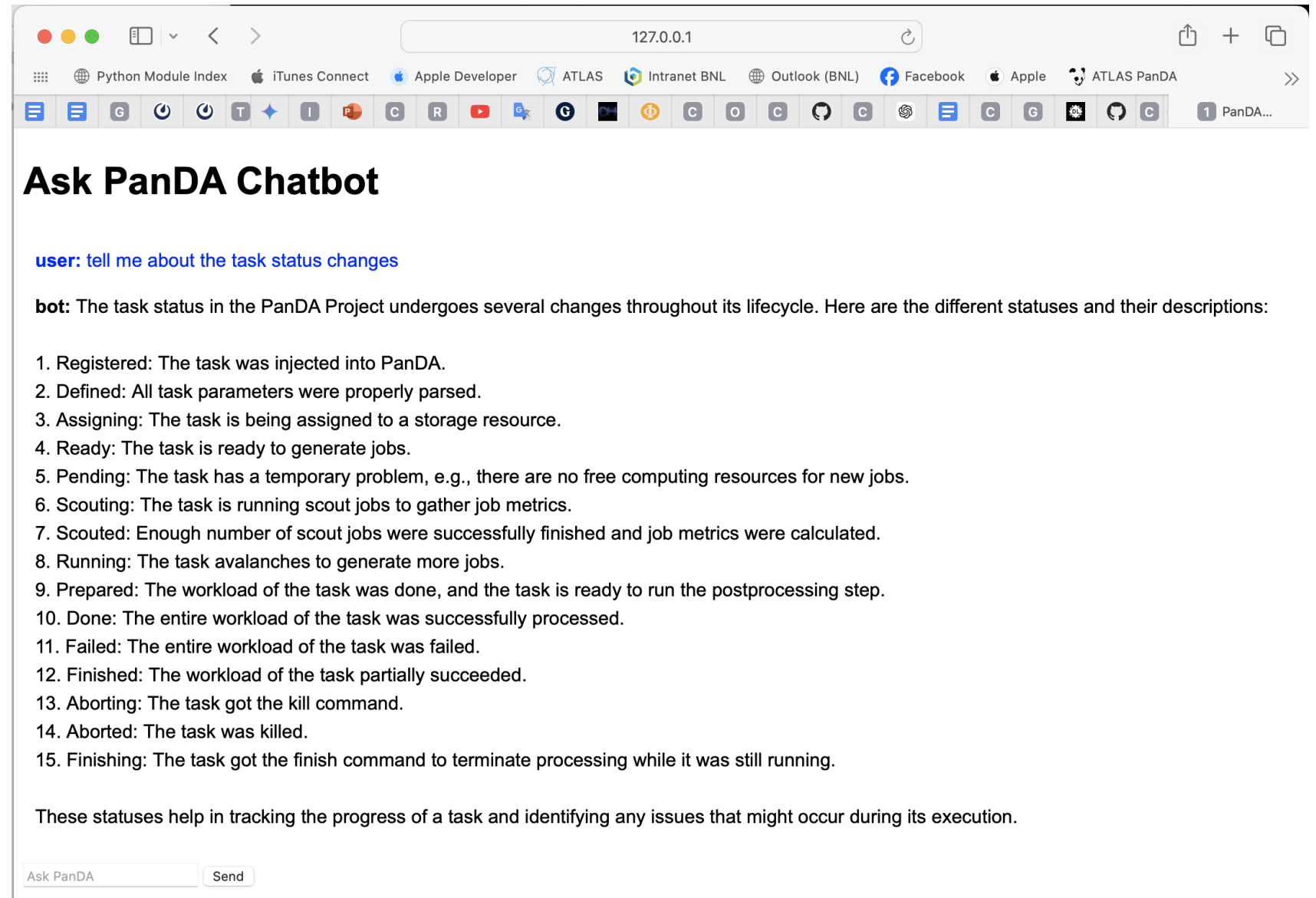
- `--exec`` takes the execution string which is executed on remote computing resources. In this case, it's printing the working directory (``pwd``), listing the files (``ls``), and echoing "Hello-world" into a file named ``myout.txt``.
- `--outDS`` takes the basename of the output collection. Here, it's generating a unique identifier using ``uuidgen`` and prefixing it with ``user.hoge.``.
- `--nJobs 3`` specifies that 3 jobs should be run.
- `--output myo`` specifies the output file name.

Please ensure that you have the necessary permissions and resources to run this command.

Ask PanDA

Example 2

- Task status changes are explained
- Formatting and list numbering done by ChatGPT



The screenshot shows a web browser window with the address bar displaying "127.0.0.1". The browser's address bar and tabs are visible at the top. The main content area displays a chatbot interface titled "Ask PanDA Chatbot".

user: tell me about the task status changes

bot: The task status in the PanDA Project undergoes several changes throughout its lifecycle. Here are the different statuses and their descriptions:

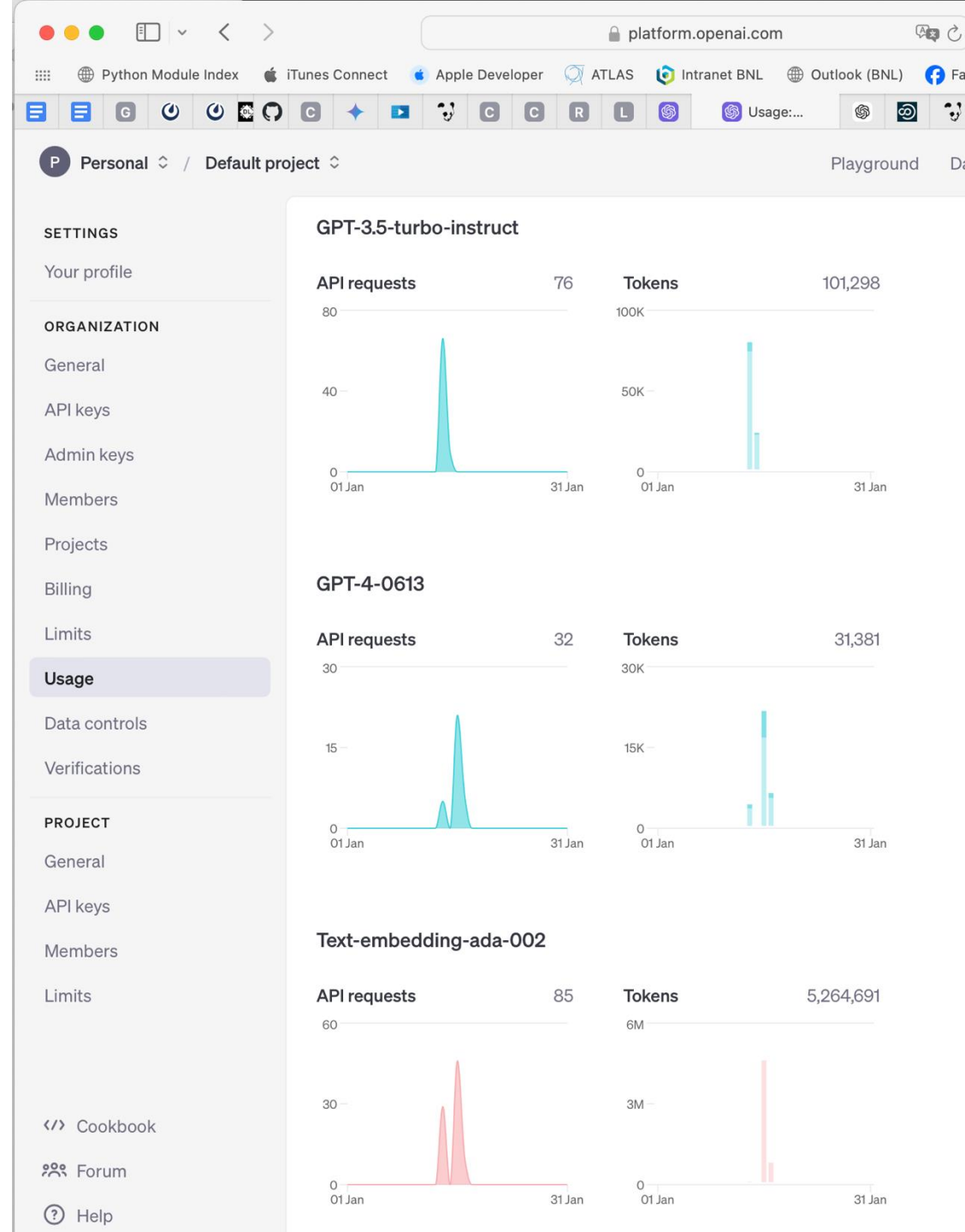
1. Registered: The task was injected into PanDA.
2. Defined: All task parameters were properly parsed.
3. Assigning: The task is being assigned to a storage resource.
4. Ready: The task is ready to generate jobs.
5. Pending: The task has a temporary problem, e.g., there are no free computing resources for new jobs.
6. Scouting: The task is running scout jobs to gather job metrics.
7. Scouted: Enough number of scout jobs were successfully finished and job metrics were calculated.
8. Running: The task avalanches to generate more jobs.
9. Prepared: The workload of the task was done, and the task is ready to run the postprocessing step.
10. Done: The entire workload of the task was successfully processed.
11. Failed: The entire workload of the task was failed.
12. Finished: The workload of the task partially succeeded.
13. Aborting: The task got the kill command.
14. Aborted: The task was killed.
15. Finishing: The task got the finish command to terminate processing while it was still running.

These statuses help in tracking the progress of a task and identifying any issues that might occur during its execution.

At the bottom of the chat interface, there is a text input field containing "Ask PanDA" and a "Send" button.

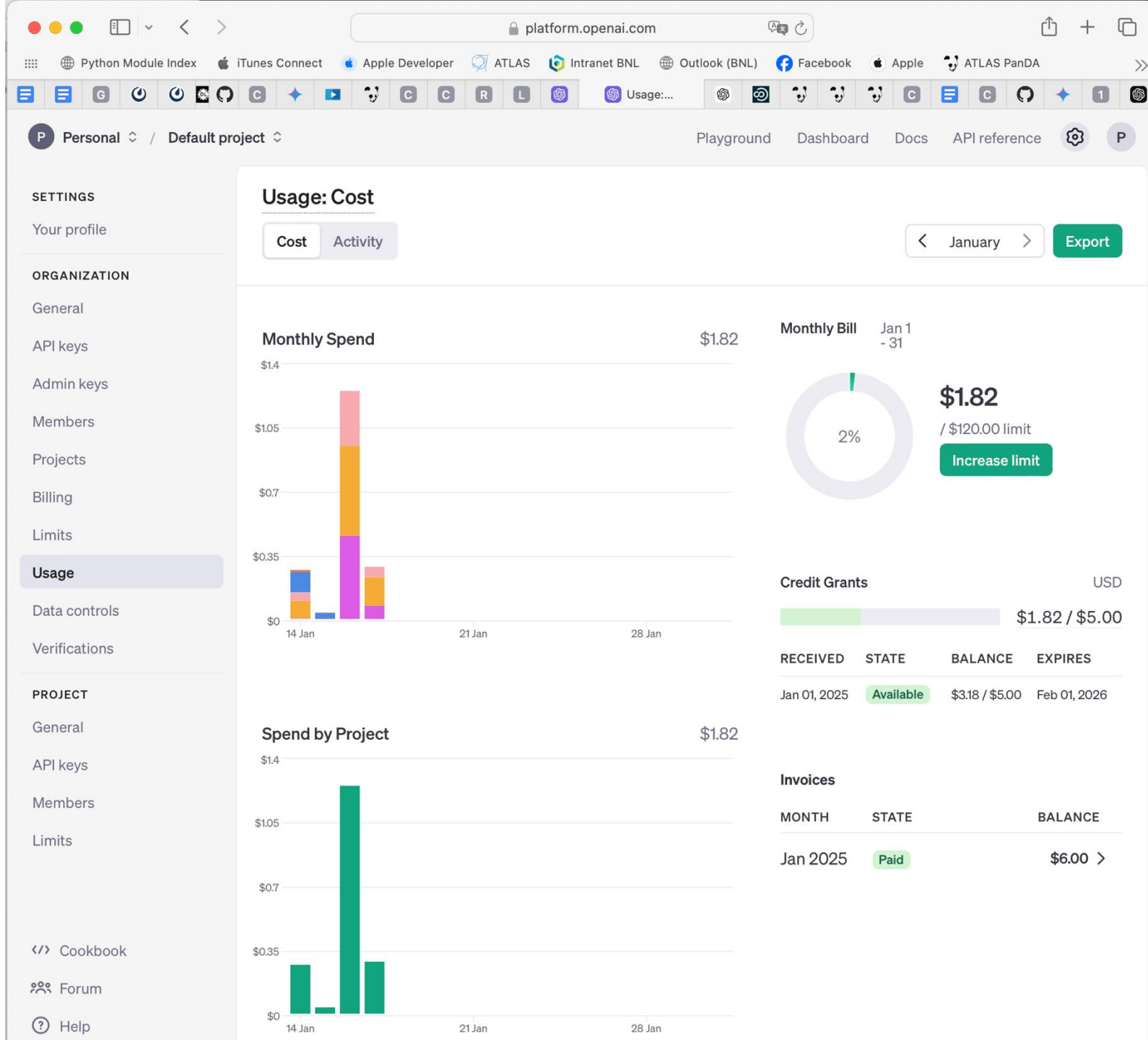
Usage statistics

- Models are priced individually
- 76 API requests with 101k tokens sent to gpt-3.5
 - User questions
- 32 API requests with 31k tokens sent to gpt-4
 - User questions
- 85 API requests with 5.2M tokens sent to embedding model
 - To find the most relevant chunk for a given question



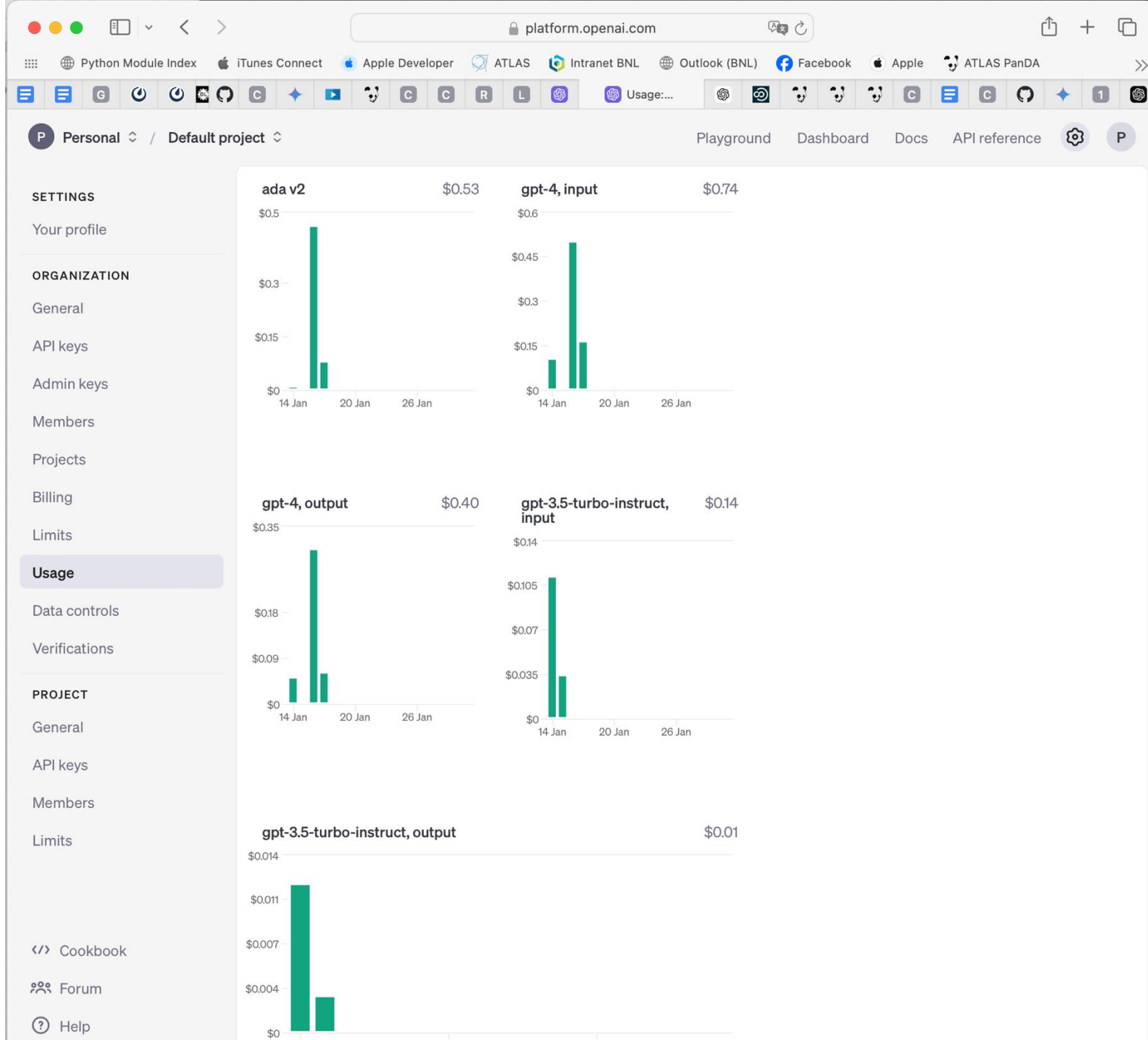
Usage Costs

- Total cost after around 100 questions: \$1.82
 - (But mostly using gpt-3.5)



Breakdown

- Cost for embedding model: \$0.53
- Cost for questions + responses: \$1.29
- gpt-3.5 model really cheap, but also not so good..



More advanced prompts

- Querying static documentation is “easy”
 - As long as documentation is written properly
 - Some quick testing revealed that answers to some questions could be improved, i.e. the documentation could be improved
- What about more advanced / dynamic questions?
 - User: Why did my job fail? [-> Why did job 12345678 fail?]
 - A proper prompt could be automatically generated that will get the job failure info via a URL, then the error info could be explained to some level
 - If the stdout/stderr is available, that could be downloaded and interpreted (extraction of relevant info)
 - Expert: how many pilots failed during the last hour at site XXX?
 - Generate corresponding URL etc

Outlook

- Improve training document where ever needed, collect feedback
 - Pilot documentation not part of PanDA readthedocs (only linked), so add that
- Implement handling of more advanced queries
- Demo available on my laptop
- Obviously not for free
 - Need to discuss funding at some point
- It might also be a good idea to try other technologies and methods