HTTP Prompt Documentation

Release 1.0.0

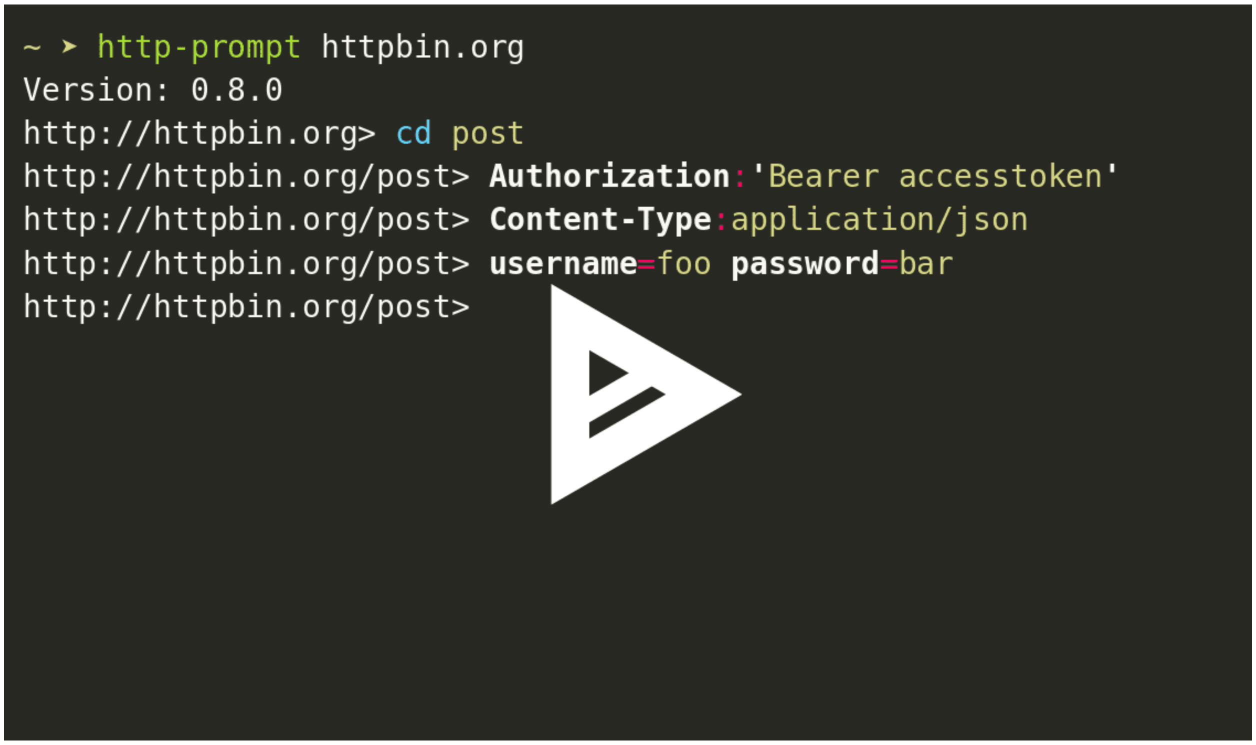
Chang-Hung Liang

Jan 15, 2019

Contents

HTTP Prompt is an interactive command-line HTTP client featuring autocomplete and syntax highlighting, built on HTTPie and prompt_toolkit.

See it in action:



```
~ ➤ http-prompt httpbin.org
Version: 0.8.0
http://httpbin.org> cd post
http://httpbin.org/post> Authorization:'Bearer accesstoken'
http://httpbin.org/post> Content-Type:application/json
http://httpbin.org/post> username=foo password=bar
http://httpbin.org/post>
```

CHAPTER 1

Contents

1.1 User Guide

1.1.1 Installation

Just install it like a regular Python package:

```
$ pip install http-prompt
```

You'll probably see some permission errors if you're trying to install it on the system-wide Python. It isn't recommended. But if that's what you want to do, you need to `sudo`:

```
$ sudo pip install http-prompt
```

Another alternative is to use `--user` option to install the package into your user directory:

```
$ pip install --user http-prompt
```

To upgrade HTTP Prompt, do:

```
$ pip install -U http-prompt
```

1.1.2 Quickstart

To start a session, you use the `http-prompt` executable:

```
# Start with the last session or http://localhost:8000
$ http-prompt

# Start with the given URL
$ http-prompt http://httpbin.org

```

(continues on next page)

(continued from previous page)

```
# Start with some initial options
$ http-prompt localhost:8000/api --auth user:pass username=somebody
```

Once you're in a session, you can use the following commands.

To change URL address, use `cd`:

```
# Relative URL path
> cd api/v1

# Absolute URL
> cd http://localhost/api
```

To add headers, querystring, or body parameters, use the syntax as in HTTPie. The following are all valid:

```
# Header
> Content-Type:application/json

# Querystring parameter
> page==2

# Body parameters
> username=foo
> full_name='foo bar'

# Body parameters in raw JSON (new in v0.9.0)
> number:=1234
> is_ok:=true
> names:=["foo","bar"]
> user:='{"username": "foo", "password": "bar"}'

# Write them in one line
> Content-Type:application/json page==2 username=foo
```

You can also add HTTPie options like this:

```
> --form --auth user:pass
> --verify=no

# HTTPie options and request parameters in one line
> --form --auth user:pass username=foo Content-Type:application/json
```

To preview how HTTP Prompt is going to call HTTPie, do:

```
> httpie post
http --auth user:pass --form POST http://localhost/api apikey==abc username=john
```

You can temporarily override the request parameters by supplying options and parameters in `httpie` command. The overrides won't affect the later requests.

```
# No parameters initially
> httpie
http http://localhost

# Override parameters temporarily
> httpie /api/something page==2 --json
http --json http://localhost/api/something page==2
```

(continues on next page)

(continued from previous page)

```
# Current state is not affected by the above overrides
> httpie
http http://localhost
```

Since v0.6.0, apart from `httpie` command, you can also use `env` to print the current session:

```
> env
--verify=no
cd http://localhost
page==10
limit==20
```

To actually send an HTTP request, enter one of the HTTP methods:

```
> get
> post
> put
> patch
> delete
> head
> options (new in v0.8.0)
```

The above HTTP methods also support temporary overriding:

```
# No parameters initially
> httpie
http http://localhost

# Send a request with some overrided parameters
> post /api/v1 --form name=jane

# Current state remains intact
> httpie
http http://localhost
```

To remove an existing header, a querystring parameter, a body parameter, or an HTTPie option:

```
# Remove a header
> rm -h Content-Type

# Remove a querystring parameter
> rm -q apikey

# Remove a body parameter
> rm -b username

# Remove an HTTPie option
> rm -o --auth
```

To reset the session, i.e., clear all parameters and options:

```
> rm *
```

To exit a session, simply enter:

```
> exit
```

1.1.3 Output Redirection

New in v0.6.0.

You can redirect the output of a command to a file by using the syntax:

```
# Write output to a file
> COMMAND > /path/to/file

# Append output to a file
> COMMAND >> /path/to/file
```

where `COMMAND` can be one of the following:

- `env`
- `httpie`
- HTTP actions: `get`, `post`, `put`, `patch`, `delete`, `head`, `options`

Saving and Loading Sessions

One of the use cases of output redirection is to save and load sessions, which is especially useful for team collaboration, where you want to share your sessions with your team members.

To save your current session, you redirect the output of `env` to a file:

```
> env > /path/to/file
```

To load a saved session, you can use `source` or `exec`. Their only difference is that `exec` wipes out the current session before loading. Usage:

```
# Update the current session
> source /path/to/file

# Wipe out the current session and load from a file
> exec /path/to/file
```

New in v0.11.0.

Load a saved session from the command line directly with the `--env` option. This allows you for example to define aliases and easily start HTTP Prompt with a full configuration already loaded for each of your projects.

```
# Define alias for project1
$ alias http_project1='http-prompt --env /path/to/project1/env/file'

# Launch HTTP Prompt for project1
$ http_project1
```

Any extra argument in the command line is still used and overwrites the value from the session file if already present

```
# Use saved session but overwrite the URL and add a parameter
$ http-prompt --env /path/to/file localhost:8080 page==2
```

Saving HTTP Responses

Printing HTTP responses to the console is good for small text responses. For larger text or binary data, you may want to save the response to a file. Usage:

```
# Save http://httpbin.org/image/png to a file
> cd http://httpbin.org/image/png
> get > pig.png

# Or use this one-liner
> get http://httpbin.org/image/png > pig.png
```

1.1.4 Pipeline

New in v0.7.0.

HTTP Prompt supports simplified pipeline syntax, where you can pipe the output to a shell command:

```
# Replace 'localhost' to '127.0.0.1'
> httpie POST http://localhost | sed 's/localhost/127.0.0.1/'
http http://127.0.0.1

# Only print the line that contains 'User-Agent' using grep
> get http://httpbin.org/get | grep 'User-Agent'
    "User-Agent": "HTTPie/0.9.6"
```

On macOS, you can even copy the result to the clipboard using `pbcopy`:

```
# Copy the HTTPie command to the clipboard (macOS only)
> httpie | pbcopy
```

Another cool trick is to use jq to parse JSON data:

```
> get http://httpbin.org/get | jq '.headers."User-Agent"'
"HTTPie/0.9.6"
```

Note: Syntax with multiple pipes is not supported currently.

1.1.5 Shell Substitution

New in v0.7.0.

Shell substitution happens when you put a shell command between two backticks like `` `...` ``. This syntax allows you compute a value from the shell environment and assign the value to a parameter:

```
# Set date to current time
> date==`date -u +"%Y-%m-%d %H:%M:%S"`
> httpie
http http://localhost:8000 'date==2016-10-08 09:45:00'

# Get password from a file. Suppose the file has a content of
# "secret_api_key".
> password==`cat ./apikey.txt`
> httpie
http http://localhost:8000 password==secret_api_key
```

1.1.6 Configuration

New in v0.4.0.

When launched for the first time, HTTP Prompt creates a user config file at `$XDG_CONFIG_HOME/http-prompt/config.py` (or `%LOCALAPPDATA%/http-prompt/config.py` on Windows). By default, it's `~/.config/http-prompt/config.py` (or `~/AppData/Local/http-prompt/config.py`).

`config.py` is a Python module with all the available options you can customize. Don't worry. You don't need to know Python to edit it. Just open it up with a text editor and follow the guidance inside.

1.1.7 Persistent Context

New in v0.4.0.

HTTP Prompt keeps a data structure called *context* to represent your current session. Every time you enter a command modifying your context, HTTP Prompt saves the context to your filesystem, enabling you to resume your previous session when you restart `http-prompt`.

The last saved context is located at `$XDG_DATA_HOME/http-prompt/context.hp` (or `%LOCALAPPDATA%/http-prompt/context.hp` on Windows). By default, it's `~/.local/share/http-prompt/context.hp` (or `~/AppData/Local/http-prompt/context.hp`).

As context data may contain sensitive data like API keys, you should keep the user data directory private. By default, HTTP Prompt sets the modes of `$XDG_DATA_HOME/http-prompt` to `rwx------` (i.e., `700`) so that the only person who can read it is the owner (you).

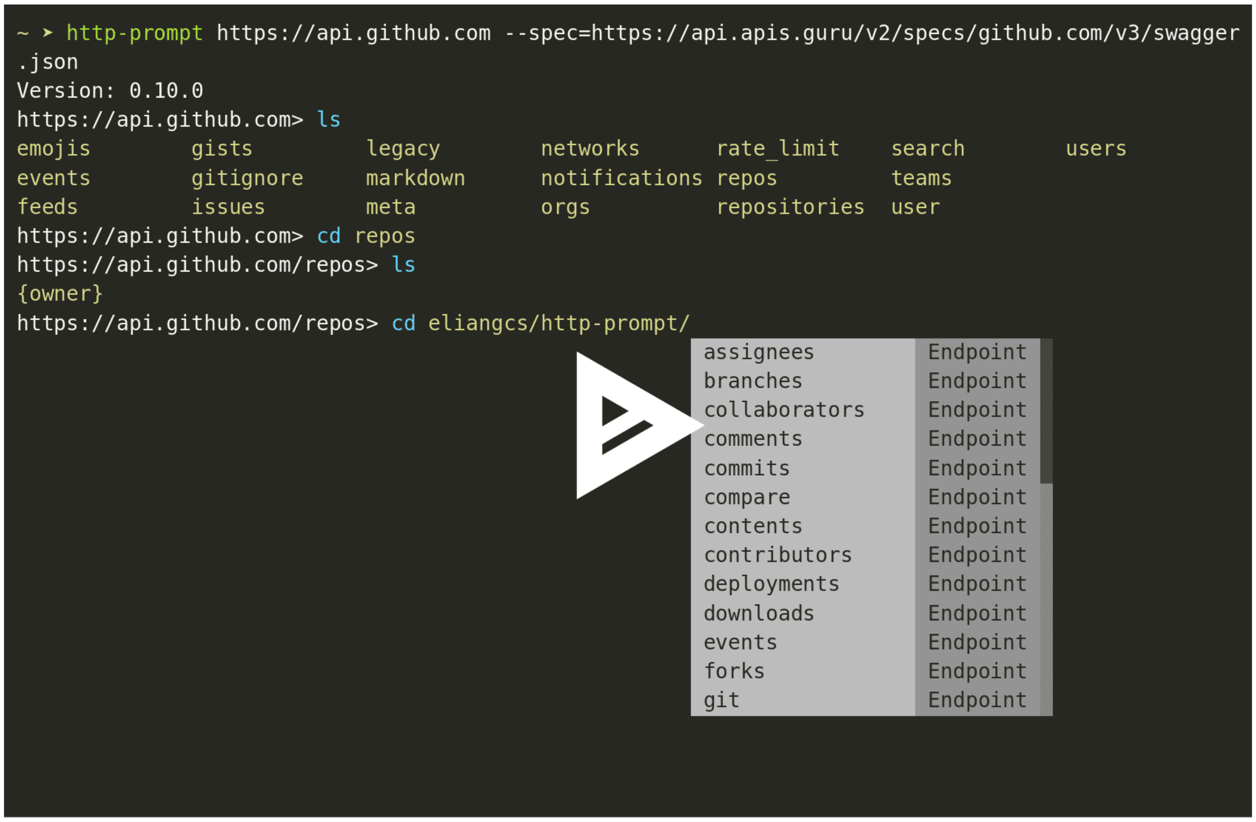
Note for users of older versions: Since 0.6.0, HTTP Prompt only stores the last context instead of grouping multiple contexts by hostnames and ports like it did previously. We changed the behavior because the feature can be simply replaced by `env`, `exec` and `source` commands. See the discussion in issue #70 for detail.

1.1.8 `ls`, `cd`, and OpenAPI/Swagger Specification

New in v0.10.0.

OpenAPI (formerly known as Swagger) is a specification that describes an HTTP/REST API. The `http-prompt` has a `--spec` option for you to provide an OpenAPI specification in JSON format. The specification enables HTTP Prompt to do some cool things like autocomplete API endpoint paths and parameters for you.

See it in action:



```
~ ➤ http-prompt https://api.github.com --spec=https://api.apis.guru/v2/specs/github.com/v3/swagger
.json
Version: 0.10.0
https://api.github.com> ls
emojis        gists         legacy        networks      rate_limit    search        users
events        gitignore     markdown      notifications repos         teams
feeds         issues        meta          orgs          repositories  user
https://api.github.com> cd repos
https://api.github.com/repos> ls
{owner}
https://api.github.com/repos> cd eliangcs/http-prompt/
                                                       assignees        Endpoint
                                                       branches         Endpoint
                                                       collaborators    Endpoint
                                                       comments         Endpoint
                                                       commits          Endpoint
                                                       compare          Endpoint
                                                       contents         Endpoint
                                                       contributors     Endpoint
                                                       deployments      Endpoint
                                                       downloads        Endpoint
                                                       events           Endpoint
                                                       forks            Endpoint
                                                       git              Endpoint
```

To use this feature, specify an OpenAPI/Swagger specification file with `--spec` command line option:

```
# Specify a spec on local filesystem
$ http-prompt http://localhost:8000 --spec /path/to/spec.json

# Specify a spec on the internet (https://apis.guru has lots of them)
$ http-prompt https://api.github.com --spec https://api.apis.guru/v2/specs/github.com/
↪v3/swagger.json
```

Then you can use `ls` and `cd` commands to navigate API endpoints with autocomplete!

1.2 Contributor Guide

This document is for developers who want to contribute code to this project. Any contributions are welcome and greatly appreciated!

This project follows the common conventions of a Python/GitHub project. So if you're already an experienced Python/GitHub user, it should be straightforward for you to set up your development environment and send patches. Generally, the steps include:

1. Fork and clone the repo
2. Create a virtualenv for this project
3. Install dependent packages with `pip install -e .`
4. Install test dependent packages with `pip install -r requirements-test.txt`
5. Make your changes to the code

6. Run tests with `pytest` and `tox`
7. Commit and push your changes
8. Send a pull request
9. Wait to be reviewed and get merged!

If you're not familiar with any of the above steps, read the following instructions.

1.2.1 Forking

Fork is like copying someone else's project to your account, so you can start your own independent development without interfering with the original one.

To fork HTTP Prompt, just click the **Fork** button on HTTP Prompt's GitHub project page. Then you clone your fork to your local computer:

```
$ cd ~/Projects
$ git clone git@github.com:{YOUR_USERNAME}/http-prompt.git
```

Read Forking Projects on GitHub to learn more.

1.2.2 Working with virtualenv

virtualenv is the de facto standard tool when developing a Python project. Instead of polluting your system-wide Python installation with different Python projects, virtualenv creates an isolated Python environment exclusively for a Python project.

There are several tools you can use for managing virtualenvs. In this guide, we'll show you how to use pyenv and pyenv-virtualenv, which is one of the most popular virtualenv management tools.

Make sure you have installed pyenv and pyenv-virtualenv first.

HTTP Prompt should work on Python 2.6, 2.7, 3.3 to 3.6. You can use any of these Python versions as your development environment, but using the latest version (3.6.x) is probably the best. You can install the latest Python with pyenv:

```
$ pyenv install 3.6.0
```

This will install Python 3.6.0 in `~/.pyenv/versions/3.6.0` directory. To create a virtualenv for HTTP Prompt, do:

```
$ pyenv virtualenv 3.6.0 http-prompt
```

The command means: create a virtualenv named "http-prompt" based on Python 3.6.0. The virtualenv can be found at `~/.pyenv/versions/3.6.0/envs/http-prompt`.

To activate the virtualenv, do:

```
$ pyenv activate http-prompt
```

This will switch your Python environment from the system-wide Python to the virtualenv's (named "http-prompt") Python.

To go back to the system-wide Python, you have to deactivate the virtualenv:

```
$ pyenv deactivate
```

Refer to pyenv and pyenv-virtualenv if anything else is unclear.

1.2.3 Installing Dependent Packages

The dependent packages should be installed on a virtualenv, so make sure you activate your virtualenv first. If not, do:

```
$ pyenv activate http-prompt
```

It is also recommended to use the latest version of pip. You can upgrade it with:

```
$ pip install -U pip
```

Install HTTP Prompt with its dependent packages:

```
$ cd ~/Projects/http-prompt
$ pip install -e .
```

`pip install -e .` means install the `http-prompt` package in editable mode (or developer mode). This allows you to edit code directly in `~/Projects/http-prompt` without reinstalling the package. Without the `-e` option, the package will be installed to Python's `site-packages` directory, which is not convenient for developing.

1.2.4 Installing Test Dependent Packages

Test requirements are placed in a separate file named `requirements-test.txt`. To install them, do:

```
$ cd ~/Projects/http-prompt
$ pip install -r requirements-test.txt
```

1.2.5 Making Your Changes

Code Style

Always lint your code with Flake8. You can set it up in your code editor or simply use `flake8` in the command line.

The Hitchhiker's Guide to Python provides the best Python coding practices. We recommend anyone who wants to write good Python code to read it.

Adding Features

Before you add a new feature, make sure you create an issue making a proposal first, because you don't want to waste your time on something that the community don't agree upon.

Python 2 and 3 Compatibility

HTTP Prompt is compatible with Python 2 and 3. Keep in mind that you're coding for Python 2 and 3 at the same time. You can use Tox (see below) to make sure the code is runnable on both Python 2 and 3.

Documentation

Documentation is written in Sphinx. To build documentation, you need to install Sphinx first:

```
$ pip install sphinx
```

To build and view documentation in HTML, do:

```
$ cd ~/Projects/http-prompt/docs
$ make html
$ open _build/html/index.html
```

1.2.6 Running Tests

Single Python Version

Make sure your virtualenv is activated. To run tests, do:

```
$ cd ~/Projects/http-prompt
$ pytest
```

`pytest` runs the tests with your virtualenv's Python version. This is good for fast testing. To test the code against multiple Python versions, you use Tox.

Multiple Python Versions

All the commands in this section should **NOT** be run in a virtualenv. Deactivate it first if you're in a virtualenv:

```
$ pyenv deactivate
```

Make sure you have installed all the Python versions we're targeting. If not, do:

```
$ pyenv install 2.6.9
$ pyenv install 2.7.12
$ pyenv install 3.3.6
$ pyenv install 3.4.5
$ pyenv install 3.5.2
$ pyenv install 3.6.0
$ pyenv install pypy-5.3.1
$ pyenv install pypy3-2.4.0
```

To use Tox with pyenv, you have to instruct pyenv to use multiple Python versions for the project:

```
$ cd ~/Projects/http-prompt
$ pyenv local 3.6.0 3.5.2 3.4.5 3.3.6 2.7.12 2.6.9 pypy-5.3.1 pypy3-2.4.0
```

This will generate a `.python-version` in the project directory:

```
$ cat ~/Projects/http-prompt/.python-version
3.6.0
3.5.2
3.4.5
3.3.6
2.7.12
```

(continues on next page)

(continued from previous page)

```
2.6.9
pypy-5.3.1
pypy3-2.4.0
```

This tells pyenv to choose a Python version based on the above order. In this case, 3.6.0 is the first choice, so any Python executables (such as `python` and `pip`) will be automatically mapped to the ones in `~/.pyenv/versions/3.6.0/bin`.

We want to run `tox` using on Python 3.6.0. Make sure you have installed Tox:

```
$ pip install tox
```

To run tests, execute `tox`:

```
$ cd ~/Projects/http-prompt
$ tox
```

Tox will install the test Python environments in the `.tox/` directory in the project directory, and run the test code against all the Python versions listed above.

1.2.7 Code Review

Once you made changes and all the tests pass, push your modified code to your GitHub account. Submit a pull request (PR) on GitHub for the maintainers to review. If the patch is good, The maintainers will merge it to the master branch and ship the new code in the next release. If the patch needs improvements, we'll give you feedback so you can modify accordingly and resubmit it to the PR.

CHAPTER 2

Roadmap

- Support for advanced HTTPie syntax, e.g, `field=@file.json`
- Support for cURL command and raw format preview
- Improve autocomplete
- Python syntax evaluation
- HTTP/2 support

CHAPTER 3

User Support

We'd love to hear more from our users! Please use the following channels for bug reports, feature requests, and questions:

- GitHub issues
- Gitter chat room

CHAPTER 4

Contributing

Are you a developer and interested in contributing to HTTP Prompt? See *Contributor Guide*.

CHAPTER 5

Thanks

- HTTPie: for designing such a user-friendly HTTP CLI
- prompt_toolkit: for simplifying the work of building an interactive CLI
- Parsimonious: for the PEG parser used by this project
- pgcli: for the inspiration of this project
- Contributors: for improving this project