



HTCondor Python Bindings

Todd Tannenbaum
European HTCondor Workshop 2019 Ispra Italy

HTCondor Clients in 2013

Command Line Clients

Fully Featured!

Requires fork/exec and process handling

Outputs in multiple formats

Something
Missing
In
The
Middle

SOAP Clients

Features! (Some)

Language agnostic (everyone hates XML equally?)

Caveats with respect to scalability, security.

Command Line Clients
Fully Featured!
Requires fork/exec and process handling
Outputs in multiple formats



SOAP Clients
Language (everyone has?)
Can't with respect to scalability, security.

Command Line Clients

Fully Featured!

Requires fork/exec and process handling

Outputs in multiple formats



SOAP C

RESTful Clients

Features! (Some)

Language agnostic (everyone feels better w/ JSON?)

Caveats with respect to scalability, security.

Why Python?

Design Philosophy

ClassAds: Everything based on ClassAds; make these the “core” of the bindings.

Pythonic: Semantics and APIs should feel natural to a python programmer.

- Use iterators, exceptions, guards.
- ClassAds behave as much like a dict as reasonable.

Backward compatible: APIs are here to stay for as long as possible.

- When we absolutely must, use standard python `DeprecationWarning` techniques.
- Yes, this means that we keep even design warts for far longer than we'd like!

Native code: Call same HTCondor library code as CLI; identical in performance.

Complete: If you can do it with the command line tools, you should be able to do it with python.

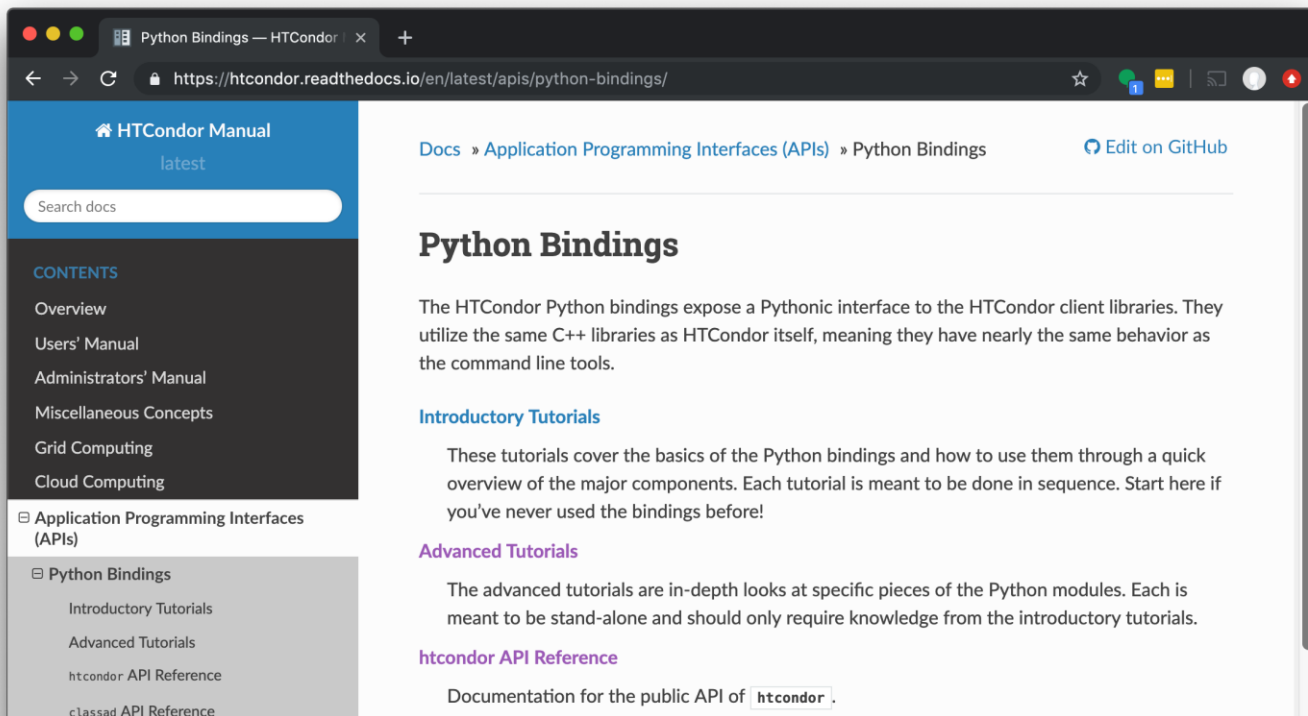
Pythonic!

Since *pythonic* is in our design philosophy, the education tools should use the tools favored by the python community:

- Sphinx-based documentation. Hosted on ReadTheDocs; looks / feels / smells like python documentation.
 - *Also now used by the HTCondor Manual!*
- Jupyter-based tutorials. Use Binder.org service to spawn a Docker container with a private HTCondor instance (or use Docker locally). Interact via your browser.

Documentation in the HTCondor Manual

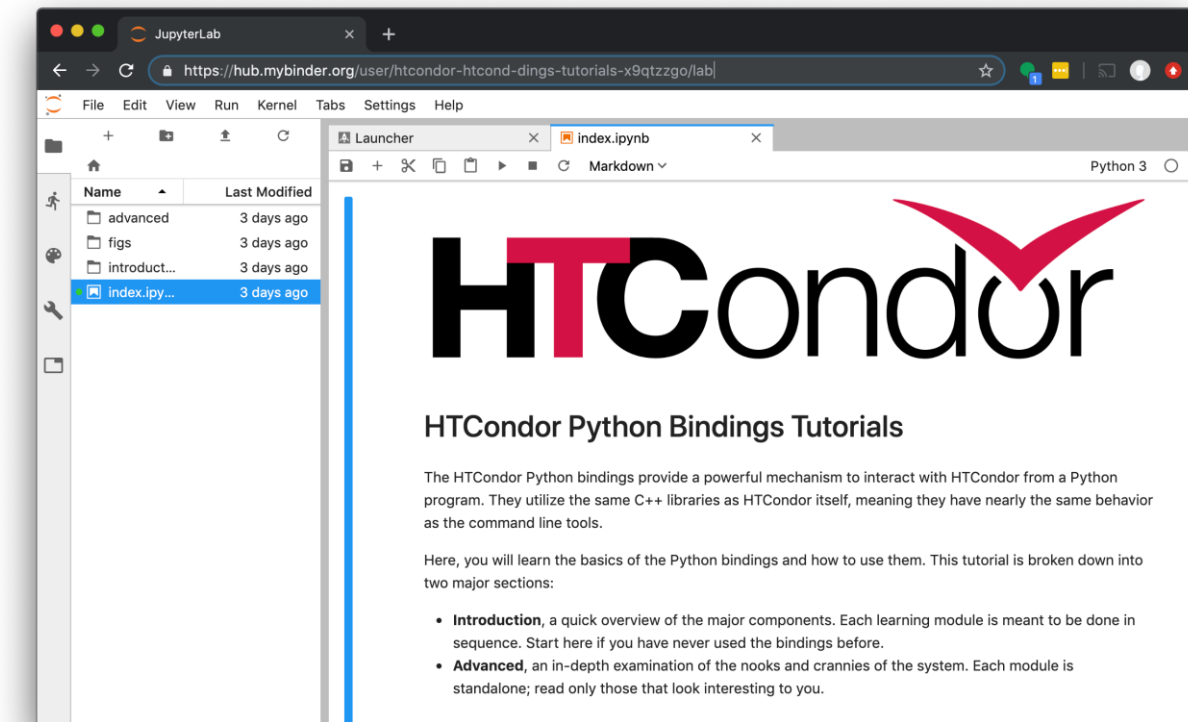
<https://htcondor.readthedocs.io/en/latest/apis/python-bindings>



The screenshot shows a web browser displaying the HTCondor Manual page for Python Bindings. The browser's address bar shows the URL <https://htcondor.readthedocs.io/en/latest/apis/python-bindings/>. The page has a blue header with the HTCondor Manual logo and the word "latest". Below the header is a search bar labeled "Search docs". On the left side, there is a dark sidebar with a "CONTENTS" section. The "CONTENTS" section lists various manual sections, with "Application Programming Interfaces (APIs)" and "Python Bindings" expanded. Under "Python Bindings", there are links for "Introductory Tutorials", "Advanced Tutorials", "htcondor API Reference", and "classad API Reference". The main content area on the right has a breadcrumb trail: "Docs » Application Programming Interfaces (APIs) » Python Bindings" and a link to "Edit on GitHub". The main heading is "Python Bindings". The text below the heading states: "The HTCondor Python bindings expose a Pythonic interface to the HTCondor client libraries. They utilize the same C++ libraries as HTCondor itself, meaning they have nearly the same behavior as the command line tools." Below this text are three sections: "Introductory Tutorials" (with a paragraph about covering basics), "Advanced Tutorials" (with a paragraph about in-depth looks), and "htcondor API Reference" (with a paragraph about documentation for the public API of `htcondor`).

Live Jupyter-based tutorials

<https://github.com/htcondor/htcondor-python-bindings-tutorials>



The screenshot shows a JupyterLab interface in a web browser. The browser address bar displays the URL: <https://hub.mybinder.org/user/htcondor-htcond-dings-tutorials-x9qtzgo/lab>. The JupyterLab interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help), a file browser on the left, and a main content area. The file browser shows a list of files and folders:

Name	Last Modified
advanced	3 days ago
figs	3 days ago
introduc...	3 days ago
index.ipy...	3 days ago

The main content area displays the HTCondor logo and the title "HTCondor Python Bindings Tutorials". Below the title, there is a paragraph of text and a bulleted list of sections:

The HTCondor Python bindings provide a powerful mechanism to interact with HTCondor from a Python program. They utilize the same C++ libraries as HTCondor itself, meaning they have nearly the same behavior as the command line tools.

Here, you will learn the basics of the Python bindings and how to use them. This tutorial is broken down into two major sections:

- **Introduction**, a quick overview of the major components. Each learning module is meant to be done in sequence. Start here if you have never used the bindings before.
- **Advanced**, an in-depth examination of the nooks and crannies of the system. Each module is standalone; read only those that look interesting to you.

You can help!

The contents of the tutorials and documentation are kept on GitHub:

- <https://github.com/htcondor/htcondor-python-bindings-tutorials>
- Note the new location for 2019! JupyterLab & Binder integration recently overhauled by Josh Karpel.

Find a bug? Spot some missing content?

- Simply send a pull request; Travis-CI will test and update the static content once merged.

Installing Python Bindings

- › On Linux
 - Included in RPM and DEB packages for the system python (usually some Python 2 version)
 - Python 3 bindings also included in packaging for Debian, Ubuntu, Centos7, RHEL7 as of v8.8.4.
 - Also available in PyPI:
 - Just do `pip install htcondor`
 - For a specific version `pip install htcondor==8.8.4`
 - Working on adding into Anaconda
- › On Windows
 - Windows MSI installs bindings for both Python 2.7 and Python 3.6
- › On Mac
 - Nothing (yet)

Key Concept #1: ClassAds

The *lingua franca* of HTCCondor



Simple Example

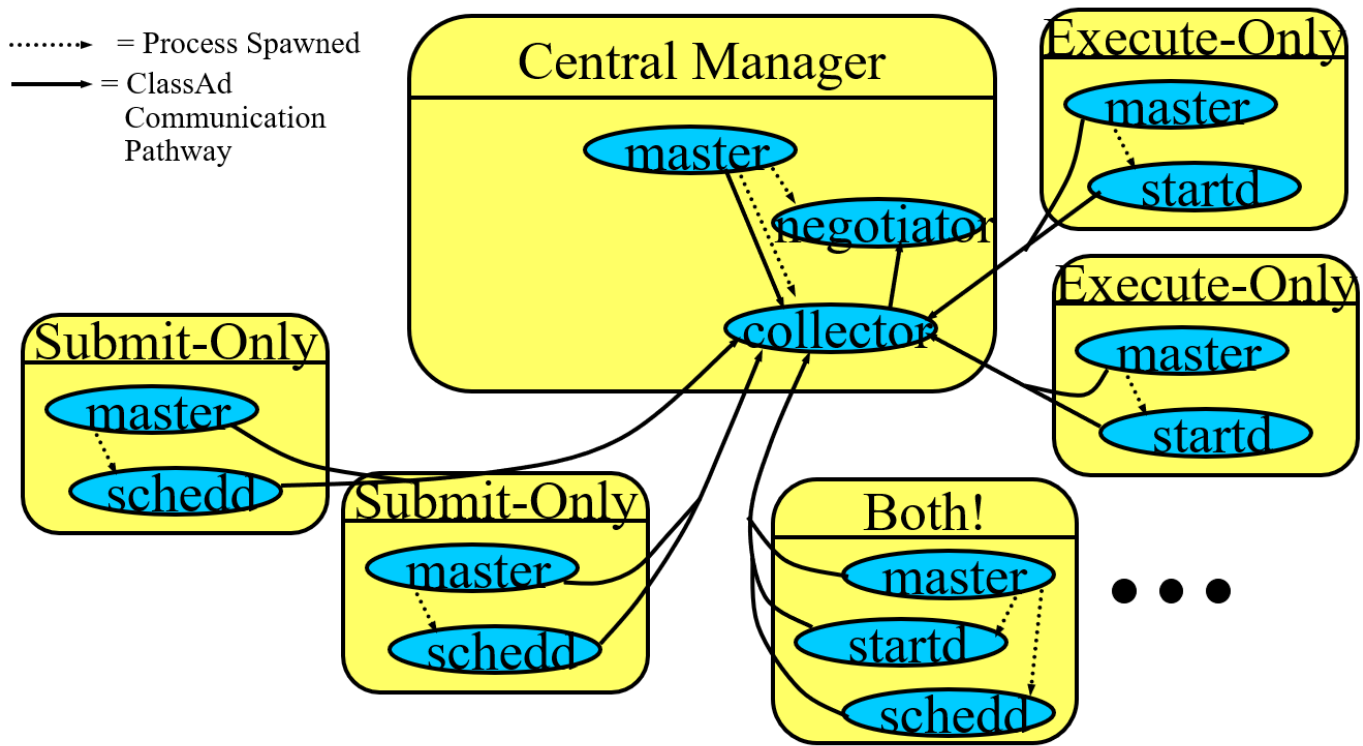
Job Ad

```
Type = "Job"
Requirements =
  HasMatlabLicense
    == True &&
  Memory >= 1024
Rank = kflops + 1000000
  * Memory
Cmd= "/bin/sleep"
Args = "3600"
Owner = "gthain"
NumJobStarts = 8
KindOfJob = "simulation"
Department = "Math"
```

Machine Ad

```
Type = "Machine"
Cpus = 40
Memory = 2048
Requirements =
  (Owner == "gthain") ||
  (KindOfJob ==
    "simulation")
Rank = Department == "Math"
HasMatlabLicense = true
MaxTries = 4
kflops = 41403
```

Concept #2: HTCondor Daemons!



Two key Concepts, two modules

> `import classad`

- Provides two classes: ClassAd and ExprTree
- ClassAds are the *lingua franca* of HTCondor
- TL;DR : The ClassAd Python class behaves a lot like a Python dictionary (key : value)

> `import htcondor`

- Provides a class for each interesting daemon type (Schedd,Collector,Negotiator,Startd), and methods to perform operations against them
- Provides a Submit class for `condor_submit` and `condor_submit_dag` type functionality
- Some misc other stuff 😊

Classes for interesting Daemons

› import htcondor

- htcondor.Collector()

- query, directQuery, locate, advertise : condor_status, advertise

- htcondor.Schedd()

- query, xquery, history : look at jobs
- act, edit : remove, hold, change jobs
- transaction, spool, retrieve : submit jobs
- submit, submitMany : obsolete submit interface!

Classes for daemons, cont.

- htcondor.Startd()
 - drainJobs, cancelDrainJobs
- htcondor.Negotiator()
 - setPriority, setFactor, resetUsage, ...
 - getPriorities, getResourceUsage
 - (In v8.8+ you can query Accounting ads from the Collector)

Submit Class

› htcondor.Submit()

- queue : submit 1 or more jobs
- queue_with_itemdata : submit 1 job for each item
- from_dag : submit a DAGMan workflow file

Helpful HOWTO overview on submitting jobs from Python:

http://htcondor.org/python_notebook_examples/HOWTO_Submit_bag_of_jobs.html

HTCondor configuration methods

- › `htcondor.version()`
 - get the HTCondor version
- › `htcondor.param['knob']`
 - get the expanded value of the config knob
- › `htcondor.reload_config()`
 - reread the HTCondor config files
- › `htcondor.RemoteParam(daemonAd)`
 - query the configuration of a daemon

Log Readers

- htcondor.JobEventLog
 - Iterate a job event log as a stream of JobEvent(s)

Now, let's try it out!

So - What's Missing?

› condor_chirp ?

- See <https://github.com/htcondor/htchirp>
- Available via `pip install htchirp`

› A way to construct a DAG in Python?

- We are experimenting with this... feedback welcome!
- Available via `pip install htcondor-dags`
- Examples at
 - <https://github.com/htcondor/htcondor-dags/tree/master/examples>

Thank you!

Questions?