HTCondor Python Bindings Tutorial

Brian Bockelman
HTCondor Week 2019
HTCondor Clients in 2012

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

- **Sphinx**-based documentation. Hosted on ReadTheDocs; looks / feels / smells like python documentation.

  - *Hey, this is what HTCondor just adopted!*

- **Jupyter**-based tutorials. Login with a university credential; spawns a Docker container with a private HTCondor instance. Interact via your browser.
Python Bindings

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.

Introductory Tutorials

These tutorials cover the basics of the Python bindings and how to use them through a quick overview of the major components. Each tutorial is meant to be done in sequence. Start here if you've never used the bindings before!

Advanced Tutorials

The advanced tutorials are in-depth looks at specific pieces of the Python modules. Each is meant to be stand-alone and should only require knowledge from the introductory tutorials.

htcondor API Reference

Documentation for the public API of `htcondor`.
Jupyter-based Tutorials

HTCondor Python Bindings Tutorials

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.
Interacting With Daemons

In this module, we'll look at how the HTCondor Python bindings can be used to interact with running daemons.

Let's start by importing the correct modules:

```python
import htcondor
```

Configuration

The HTCondor configuration is exposed to Python in two ways:

- The local process's configuration is available in the module-level `param` object.
- A remote daemon's configuration may be queried using a `RemoteParam`

The `param` object emulates a Python dictionary:

```python
print(htcondor.param['SCHEDD_LOG'])  # Prints the schedd's current log file.
print(htcondor.param.get('TOOL_LOG'))  # Print None as TOOL_LOG isn't set by default.
print(htcondor.param.setdefault('TOOL_LOG', '/tmp/log'))  # Sets TOOL_LOG to /tmp/log.
print(htcondor.param['TOOL_LOG'])  # Prints /tmp/log, as set above.
```
Terminal View
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.
Let’s Proceed!