Rolling HTCondor Upgrades Without Rolling Over

Upgrading your HTCondor pool without users noticing

By Tim Theisen

HTC 25

CHTC HTCondor Software Suite PATh

How to Upgrade a Large Pool?

- Let OS package installer upgrade all nodes at once
  - Chaos ensues, your users will notice
- Drain the nodes, upgrade during downtime
  - Unhappy users during downtime
- Upgrade HTCondor while maintaining good throughput
  - Monthly/weekly upgrades at CHTC

HTCondor Pool

- Distributed System
  - Recovers from network hiccups
  - Since 23.10.2 can tolerate collector IP address changing
  - Machines can come and go
- Version compatibility
  - HTCondor version will operate with adjacent major versions
    - 23.0.1 will operate with 24.8.1 (may not get all features)
    - Version skew is not a concern

Tale of Two Pools

- CHTC Pool
  - Runs HTCondor feature (24.x) release candidates
  - CM runs the feature (24.x) release candidate
  - One feature (24.x) Test AP – Used for early testing before rolling out to rest of pool
  - A few test EPs representing types (EL8, EL9, GPU or not, LVM or not, X86_64, ARM64)
  - One LTS AP - runs nightly OSG VM Universe tests
  - A few LTS EPs representing most types (23.0, 23.x, 24.0, EL7, EL8, EL9)
- BaTLab Pool (HTCSS Build and Test Pool, All test nodes)
  - CM runs the LTS (23.0) version
  - AP runs the feature (24.x) version
  - One LTS x86_64 EP (in both pools)
  - Two feature x86_64 EPs (in both pools)
  - One feature ARM64 EP (in both pools)
  - A few Windows and Mac EPs

CHTC Pool

Master Versions
2 23.0.25 2025-05-02 BuildID: 806486 PackageID: 23.0.25-0.806486 RC $
31 23.10.25 2025-05-07 BuildID: 807389 PackageID: 23.10.25-0.807389 GitSHA: c34da554 RC $
5 24.0.8 2025-05-03 BuildID: 806929 PackageID: 24.0.8-0.806929 GitSHA: 74a4b0a1 RC $
265 24.8.0 2025-05-02 BuildID: 806621 PackageID: 24.8.0-0.806621 GitSHA: 75eeded5 RC $
Collector Versions
2 24.8.0 2025-05-02 BuildID: 806621 PackageID: 24.8.0-0.806621 GitSHA: 75eeded5 RC $
Negotiator Versions
3 24.8.0 2025-05-02 BuildID: 806621 PackageID: 24.8.0-0.806621 GitSHA: 75eeded5 RC $
Schedd Versions
1 23.0.25 2025-05-02 BuildID: 806486 PackageID: 23.0.25-0.806486 RC $
2 23.10.25 2025-05-07 BuildID: 807389 PackageID: 23.10.25-0.807389 GitSHA: c34da554 RC $
1 24.0.8 2025-05-03 BuildID: 806929 PackageID: 24.0.8-0.806929 GitSHA: 74a4b0a1 RC $
8 24.8.0 2025-05-02 BuildID: 806621 PackageID: 24.8.0-0.806621 GitSHA: 75eeded5 RC $
Startd Versions
251 24.8.0
28 23.10.25
4 24.0.8
1 23.0.25

CHTC HTCondor Software Suite PATh

BaTLab Pool

Collector Versions
    1 23.0.25 2025-05-02 BuildID: 806486 PackageID: 23.0.25-0.806486 RC $
Negotiator Versions
    1 23.0.25 2025-05-02 BuildID: 806486 PackageID: 23.0.25-0.806486 RC $
Schedd Versions
    1 23.10.23 2025-04-16 BuildID: 802668 PackageID: 23.10.23-0.802668 GitSHA: 10005ba4 RC $
    1 24.7.0 2025-03-31 BuildID: 797043 PackageID: 24.7.0-0.797043 GitSHA: 90ce8521 RC $
Startd Versions
    1 9.3.0 Sep 28 2021 BuildID: 557631 RC $ WINDOWS1000 X86_64
    1 10.0.3 2023-Apr-06 BuildID: 638290 $ macOS13 X86_64
    1 10.0.7 2023-Jul-25 BuildID: 664317 $ macOS13 X86_64
    1 24.0.8 2025-05-03 BuildID: 806929 PackageID: 24.0.8-0.806929 GitSHA: 74a4b0a1 RC $ CentOS9 X86_64
    1 24.4.0 2025-01-14 BuildID: 780828 GitSHA: 0b44f753 RC $ WINDOWS1000 X86_64
    1 24.8.0 2025-05-02 BuildID: 806621 PackageID: 24.8.0-0.806621 GitSHA: 75eeded5 RC $ CentOS9 aarch64
    2 24.8.0 2025-05-02 BuildID: 806621 PackageID: 24.8.0-0.806621 GitSHA: 75eeded5 RC $ CentOS9 X86_64
Startd Platforms
    1 CentOS9 aarch64
    3 CentOS9 X86_64
    2 macOS13 X86_64
    2 WINDOWS1000 X86_64

CHTC HTCondor Software Suite PATh

Open Science Pool (OSPool)

Collector Versions
27 24.8.0 2025-05-02 PackageID: 24.8.0-0.806621 RC $
Negotiator Versions
2 24.8.0 2025-05-02 PackageID: 24.8.0-0.806621 RC $
Schedd Versions
1 9.0.20 Nov 15 2023 PackageID: 9.0.20-1 $
1 9.12.0 2022-10-05 PackageID: 9.12.0-1 $
1 10.0.2 2023-03-02 PackageID: 10.0.2-1 $
3 23.0.14 2024-08-07 PackageID: 23.0.14-1 $
1 23.0.21 2025-02-27 PackageID: 23.0.21-1 $
2 23.0.22 2025-03-05 PackageID: 23.0.22-1 $
2 23.0.24 2025-04-21 PackageID: 23.0.24-1 $
2 23.10.25 2025-05-07 PackageID: 23.10.25-0.807389 RC $
2 24.0.7 2025-04-22 PackageID: 24.0.7-1 $
10 24.8.0 2025-05-02 PackageID: 24.8.0-0.806621 RC $

Startd Versions
6624 24.8.1
952 24.7.3
676 23.10.24
203 24.8.0
169 23.10.25
Startd Platforms
6935 CentOS9 X86_64
771 CentOS8 X86_64
396 DEBIAN12 X86_64
347 UBUNTU22 X86_64
110 CentOS7 X86_64
35 AlmaLinux9 X86_64
1 Rocky9 aarch64
1 AlmaLinux9 aarch64

CHTC HTCondor Software Suite PATh

My Typical Upgrade Process

- Day 1
  - Update all LTS nodes and test nodes
  - Run test jobs; the BaTLab Pool and OSG VMU Access Point are all test nodes
- Day 2
  - Update CM, most APs, kick off upgrade of 25% of Execution Points
- Day 3
  - Update very busy APs, kick off upgrade of 25% of Execution Points
- Day 4 and 5
  - Kick off upgrade of 25% of Execution Points

Upgrade/Downgrade Mechanics

- For yum, we have Puppet rules to treat HTCondor RPMs as a unit to downgrade (yum/dnf is not able to downgrade dependent packages)
  - `condor python3-condor condor-vm-gahp condor-debug-info`
- Explicit version needed in Puppet:

```
htcondor:
  version: 24.8.0-0.806621

yum:
  repos_d:
    htcondor:
      ensure: present
      repos:
        htcondor_el9_v24x_alpha:
          includepkgs:
            '*-24.8.0-0.806621.*': true
        htcondor_el9_v24x_alpha_debug:
          includepkgs:
            '*-24.8.0-0.806621.*': true
```

HTCondor Versions by Node/Type

```
tim@tim-laptop:htcondor_upgrade_wave$ ls

ap2001.yaml              lts_startd.yaml  test_schedd.yaml
ap2002.yaml              mem2001.yaml     test_startd.yaml
automated_schedd.yaml    mem2002.yaml     wave_1.yaml
cm.yaml                  mem2003.yaml     wave_2.yaml
inf-2682.yaml            mem2004.yaml     wave_3.yaml
lts_cm.yaml              mem4000.yaml     wave_4.yaml
lts_schedd.yaml          ospool_ap.yaml
```

Central Manager Upgrade

- Collector
  - In-memory database of Pool. Nodes report in periodically.
    - `UPDATE_INTERVAL` – default 5 minutes until fully populated
- Negotiator
  - State saved in accountant log file (for fair share)
- Restart mode: `GRACEFUL`
- Allow APs to operate without Central Manager for a longer time
  - `CLAIM_WORKLIFE = 12 * 3600` (12 hours - default 20 minutes)

Access Point Upgrade

- Schedd
  - Saved state in the job queue log
- Restart mode: `FAST` (New in 24.2.1, if older then `condor_restart -fast`)
  - credmons and credd always exit quickly
  - A busy AP may take considerable time with `GRACEFUL` restart to invalidate ads in the collector (they will be re-established on restart anyway)
  - `MASTER_NEW_BINARY_RESTART = FAST`
  - `MASTER_NEW_BINARY_DELAY = 5` (seconds - default 2 minutes)
- Upgrade initiated by Puppet on even hours

Claims and Leases

- The AP claims a slot on an EP and reuses that claim for the lifetime of the claim
  - `CLAIM_WORKLIFE = 12 * 3600` (12 hrs - default 20 minutes)
- The Job leases the claim for 40 minutes, it renews the lease every 5 minutes. The schedd has up to 40 minutes to reconnect to a running job.

Execution Point Upgrades

- Regular EPs restart mode: `GRACEFUL` with long timeout
  - `SHUTDOWN_GRACEFUL_TIMEOUT = 18 * 3600` (18 hours - default 30 minutes)
  - Upgrade initiated by Puppet every 6 hours (midnight, 6:00, noon, 18:00)
- High Memory EPs: Manual Restart
  - `MASTER_NEW_BINARY_RESTART = NEVER`
  - `condor_drain -cancel mem2001`
  - `condor_restart -drain -reason Upgrade`

Do users not notice?

- We essentially have a rolling drain over the week
  - The first 20 minutes after restart, Regular EPs will only accept jobs that request at least 4 cores or half the disk on the EP
  - The first 20 minutes after restart, High Memory EPs will only accept jobs that request 100 Gbyte or more of memory
  - Users with these hard to match jobs get preferential treatment during an upgrade (may have to reset expectations)
  - Frequent upgrades have masked a broken defrag configuration

Summary of Configuration Macros

```
# Extend claim worklife to allow APs to carry on without Central Manager
CLAIM_WORKLIFE = 12 * 3600

# Restart HTCondor daemons much quicker on upgrade
MASTER_NEW_BINARY_DELAY = 5

# Restart HTCondor daemons quickly on Access Points
MASTER_NEW_BINARY_RESTART = FAST

# Allow time for jobs to complete before restarting on Execution Points
SHUTDOWN_GRACEFUL_TIMEOUT = 18 * 3600

# Don't automatically restart HTCondor daemons on nodes to be drained
MASTER_NEW_BINARY_RESTART = NEVER
```

DevOps

- This is great improvement over tossing the release candidates over to the infrastructure team
  - They have to fit it into their priorities – We get this updates out quicker
  - They don’t know what changed – We know where to look for possible issues
  - In the past, they have worked around issues and forgotten to give us feedback – We discover issues sooner
- We verify that every node in the pool gets an HTCondor upgrade
  - Fix stuck Puppet runs
  - Fix simple yum/dnf cache problems
  - Fix hung Ceph mounts
  - Anything else (i.e. Puppet configuration errors) gets reported to infrastructure to solve

Repositories
(for 23.0, 23.x, 24.0, 24.x)

- **Snapshot**: Latest build that passed unit tests → OSG development
  - Tested daily with OSG VMU tests (integration tests with HTCondor-CE)
  - Build number increments
- **Alpha**: Release Candidate in Testing on CHTC
  - Current and Previous version release candidates
  - Build number increments
- **Beta**: Release Candidate in Testing on OSPool → OSG testing
  - Current and Previous version release candidates
  - Patch number increments
- **Release**: General Availability → OSG release
  - All versions (trimmed back to latest version when out of support)

pool-versions script

```
#!/bin/sh
echo "Master Versions"
condor_status -master -af CondorVersion | sed -e 's/.*CondorVersion: //' | sort -V | uniq -c | sort -k2,2V -k3,3 -k1,1nr
echo "Collector Versions"
condor_status -collector -af CondorVersion | sed -e 's/.*CondorVersion: //' | sort -V | uniq -c | sort -k2,2V -k3,3 -k1,1nr
echo "Negotiator Versions"
condor_status -negotiator -af CondorVersion | sed -e 's/.*CondorVersion: //' | sort -V | uniq -c | sort -k2,2V -k3,3 -k1,1nr
echo "Schedd Versions"
condor_status -schedd -af CondorVersion | sed -e 's/.*CondorVersion: //' | sort -V | uniq -c | sort -k2,2V -k3,3 -k1,1nr
# Use “-slot -compact” instead of “-startd” for collectors older than 23.9.6
echo "Startd Versions"
condor_status -startd -af CondorVersion | sed -e 's/.*CondorVersion: //' | sed -e 's/ .*//g' | sort -V | uniq -c | sort -k1,1nr
# condor_status -slot -compact -af Machine CondorVersion | sort -u | sed -e 's/.*CondorVersion: //' | sed -e 's/ .*//g' | sort -V | uniq -c | sort -k1,1nr
echo "Startd Platforms"
condor_status -startd -af OpSysAndVer Arch | sed -e 's/"//g' | sort | uniq -c | sort -nr
# condor_status -slot -compact -af Machine OpSysAndVer Arch | sort -u | sed -e 's/[^ ]* //' | sed -e 's/"//g' | sort | uniq -c | sort -nr
echo "Startd Versions and Platforms"
condor_status -startd -af CondorVersion OpSysAndVer Arch | sed -e 's/.*CondorVersion: //' | sed -e 's/"//g' | sort -V | uniq -c | sort -k2,2V -k3,3 -k1,1nr
# condor_status -slot -compact -af Machine CondorVersion OpSysAndVer Arch | sort -u | sed -e 's/.*CondorVersion: //' | sed -e 's/"//g' | sort -V | uniq -c | sort -k2,2V -k3,3 -k1,1nr
```

Acknowledgment

This work is supported by NSF under Cooperative Agreement OAC-2030508 as part of the PATh Project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

CHTC HTCondor Software Suite PATh