Package: submitr (via r-universe)

May 19, 2026

Title Scaffold and Submit Computational Jobs to HTC Schedulers

Version 0.1.0

Description Provides scaffolding tools to help researchers prepare and submit computational jobs to high-throughput computing (HTC) schedulers. Generates the files required to run containerized R analyses on 'HTCondor', including submit files and executable scripts, and wraps the system commands needed to stage files, submit jobs, monitor status, and retrieve results from a CHTC submit node. Provides 'htc_config()' for managing connection details and SSH connection reuse guidance. Works naturally alongside 'containr' for container image management and 'toolero' for dataset splitting and project scaffolding.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

SystemRequirements SSH client, HTCondor (https://htcondor.org)

Suggests containr, knitr, rmarkdown, spelling, testthat (>= 3.0.0), toolero, withr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Imports cli, glue, readr, yaml

VignetteBuilder knitr

URL https://erwinlares.github.io/submitr/

BugReports https://github.com/erwinlares/submitr/issues

NeedsCompilation no

Author Erwin Lares [aut, cre] (ORCID: <https://orcid.org/0000-0002-3284-828X>)

Maintainer Erwin Lares <erwin.lares@wisc.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-05-19 08:42:35 UTC

RemoteUrl https://github.com/cran/submitr

RemoteRef HEAD

RemoteSha e0c1c00ae6280128c764de2307daf3118168faf9

Contents

`htc_config` *Configure a connection to an HTC submit server*

Description

`htc_config()` creates or reads an `htc.cfg` file that stores the connection details needed by `htc_stage()`, `htc_submit()`, `htc_status()`, and `htc_fetch_results()`. On first use it prompts interactively for your username and server address, writes `htc.cfg` to `path`, and adds it to `.gitignore`. Subsequent calls read the existing file.

Usage

```
htc_config(username = NULL, server = NULL, path = ".", overwrite = FALSE)
```

Arguments

`username`	A character string. Your HTC username (NetID), e.g. `"erwin.lares"`. If `NULL` and no `htc.cfg` exists, the function prompts interactively.
`server`	A character string. The HTC submit server hostname. Defaults to `"ap2002.chtc.wisc.edu"`. If `NULL` and no `htc.cfg` exists, the function prompts interactively.
`path`	A character string. Directory where `htc.cfg` will be read from or written to. Defaults to `"."` (current working directory).
`overwrite`	Logical. If `TRUE`, recreates `htc.cfg` even if one already exists. Defaults to `FALSE`.

Value

A named list with elements `username` and `server`, returned invisibly.

SSH connection reuse

Each call to `htc_stage()`, `htc_submit()`, `htc_status()`, or `htc_fetch_results()` opens a new SSH connection to the submit server, which triggers a Duo MFA prompt each time. You can avoid this by configuring SSH connection reuse (ControlMaster) in your `~/.ssh/config` file. Add the following block:

```
Host *.chtc.wisc.edu
  ControlMaster auto
  ControlPersist 2h
  ControlPath ~/.ssh/connections/%r@%h:%p
```

Then create the connections directory:

```
mkdir -p ~/.ssh/connections
```

After this, only the first connection in a two-hour window will require Duo authentication. Full documentation: https://chtc.cs.wisc.edu/uw-research-computing/configure-ssh

Security

`htc.cfg` contains your username and server address. Neither is sensitive on its own, but `htc_config()` adds `htc.cfg` to `.gitignore` on creation to avoid accidentally committing institutional account details to a public repository.

Examples

```
# Preview what htc_config() would return without writing any files
cfg <- list(username = "netid", server = "ap2002.chtc.wisc.edu")
str(cfg)


## Not run:
# Interactive first-time setup
cfg <- htc_config()

# Non-interactive setup (for scripts)
cfg <- htc_config(
  username = "erwin.lares",
  server   = "ap2002.chtc.wisc.edu"
)

# Force recreation of htc.cfg
cfg <- htc_config(overwrite = TRUE)

# Use in other functions
htc_upload(files = c("job.sub", "job.sh"), config = cfg)

## End(Not run)
```

htc_download *Download files from an HTC submit node*

Description

`htc_download()` copies one or more files from a directory on an HTC submit node to a local directory via `scp`. It is the final step in the job submission workflow – called after `htc_status()` confirms all jobs have completed.

Usage

```
htc_download(
  files,
  remote_path = "~/",
  local_path = ".",
  config = NULL,
  dry_run = FALSE,
  verbose = FALSE
)
```

Arguments

`files`	A character vector. One or more filenames or glob patterns to download from `remote_path` on the submit node. Examples: `"results.tar.gz"`, `c("job.log", "job.err")`, `"*.tar.gz"`. Required.
`remote_path`	A character string. The directory on the submit node where the files are located. Defaults to `"~/"`. Should match the `remote_path` used in `htc_upload()` and `htc_submit()`.
`local_path`	A character string. The local directory where downloaded files will be saved. Defaults to `"."` (current working directory).
`config`	A named list as returned by `htc_config()`. Must contain `username` and `server`. If `NULL`, the function errors with instructions to call `htc_config()` first.
`dry_run`	Logical. If `TRUE`, prints the `scp` command that would be executed without running it. Defaults to `FALSE`.
`verbose`	Logical. If `TRUE`, prints progress messages. Defaults to `FALSE`.

Details

Glob patterns such as `"*.tar.gz"` are supported and are evaluated on the remote server, not locally, so they match files that exist on the submit node regardless of what is present on your local machine.

Value

Called for its side effects. Returns `invisible(NULL)`.

Workflow

`htc_download()` is the final system-facing step in the submitr workflow. Call it after `htc_status()` confirms all jobs have completed.

```
cfg <- htc_config()

htc_status(cluster_id = 6302877, config = cfg, watch = TRUE)

# Download all result tarballs
htc_download(
  files      = "*.tar.gz",
  config     = cfg,
  local_path = "results/"
)
```

Glob patterns

Glob patterns are passed to the remote shell for evaluation so they match files on the submit node, not on your local machine. The pattern is single-quoted in the `scp` command to prevent local shell expansion.

Common patterns:

- `"*.tar.gz"` – all result tarballs
- `"*.log"` – all log files
- `"*.out"` – all output files
- `"*.err"` – all error files

SSH connection reuse

Each call to `htc_download()` opens a new SSH connection. If you have not configured ControlMaster in your `~/.ssh/config`, this will trigger a Duo MFA prompt. Run `htc_config()` for setup guidance.

Examples

```
# Preview the scp command without connecting to CHTC
cfg <- list(username = "netid", server = "ap2002.chtc.wisc.edu")
htc_download(files = "*.tar.gz", config = cfg, dry_run = TRUE)


## Not run:
# All remaining examples require a live CHTC connection
cfg <- htc_config()

# Download a single file
htc_download(files = "r <- esults.tar.gz", config = cfg)

# Download multiple specific files
htc_download(
```

```
  files  = c("job.log", "job.err", "results.tar.gz"),
  config = cfg
)

# Download all result tarballs using a glob pattern
htc_download(
  files      = "*.tar.gz",
  config     = cfg,
  local_path = "results/"
)

# Download all log files from a specific remote directory
htc_download(
  files       = "*.log",
  remote_path = "~/projects/penguins/",
  local_path  = "logs/",
  config      = cfg
)

## End(Not run)
```

`htc_gen_executable` *Generate an HTCondor executable shell script for an R job*

Description

`htc_gen_executable()` writes a ready-to-use bash script (`.sh`) that HTCondor runs inside the container for each job. The script creates a results folder, runs the R script via `Rscript`, and compresses the results into a tarball for transfer back to the submit node.

Usage

```
htc_gen_executable(
  output_file = "job.sh",
  r_script = NULL,
  results_folder = "results-folder",
  mode = "single",
  set_executable = TRUE,
  verbose = FALSE,
  comments = FALSE,
  output = "."
)
```

Arguments

`output_file`	A character string. Name of the shell script to write. Must end in `".sh"`. Defaults to `"job.sh"`.

r_script	A character string. Name of the R script that HTCondor will run inside the container, e.g. "analysis.R". Must be supplied explicitly – there is no default. If you used toolero::create_qmd(use_purl = TRUE), the script is the .R file produced by purl.R after rendering.
results_folder	A character string. Name of the folder created inside the container to hold job outputs before compression. Defaults to "results-folder".
mode	A character string. Execution mode. "single" (the default) runs the R script without a positional argument, producing a single fixed-name results tarball. "multiple" passes the subset filename as a positional argument via ${1}, producing a per-job tarball named ${1}-results.tar.gz. Must match the mode used in htc_gen_submit().
set_executable	Logical. If TRUE, sets executable permissions on the generated script via Sys.chmod() so the file is ready to copy to the CHTC submit node without any additional steps. Defaults to TRUE. Set to FALSE if you prefer to manage permissions manually, in which case you must run chmod +x on the script before submitting your job.
verbose	Logical. If TRUE, prints progress messages as each section of the script is written. Defaults to FALSE.
comments	Logical. If TRUE, annotates each section with an explanatory comment describing what the line does. Useful for researchers learning the HTCondor executable script conventions. Defaults to FALSE.
output	A character string. Directory where the shell script will be written. Defaults to "." (current working directory).

Value

Called for its side effects. Writes a bash script to file.path(output, output_file) and sets executable permissions when set_executable = TRUE. Returns invisible(NULL).

Relationship to htc_gen_submit()

The executable script generated by htc_gen_executable() is the file referenced by the executable argument in htc_gen_submit(). The two functions should always use the same mode. In "multiple" mode, HTCondor passes each subset filename to the script as ${1}, which is forwarded to the R script as a positional argument. The R script must be written to accept this argument – the recommended approach is toolero::detect_execution_context():

```
context <- toolero::detect_execution_context()

input_file <- switch(context,
  interactive = "data/penguins.csv",
  quarto      = params$input_file,
  rscript     = commandArgs(trailingOnly = TRUE)[1]
)
```

Examples

```
# Single-job executable script
htc_gen_executable(r_script = "analysis.R", output = tempdir())

# Multiple-job executable script
htc_gen_executable(
  r_script = "analysis.R",
  mode     = "multiple",
  output   = tempdir()
)

# Custom R script name with annotations
htc_gen_executable(
  output_file = "run.sh",
  r_script    = "run-analysis.R",
  comments    = TRUE,
  verbose     = TRUE,
  output      = tempdir()
)
```

`htc_gen_submit` *Generate an HTCondor submit file for a containerized R job*

Description

`htc_gen_submit()` writes a ready-to-use HTCondor submit file (`.sub`) for running a containerized R job on an HTC cluster such as CHTC. It supports both single-job and multiple-job submission modes.

Usage

```
htc_gen_submit(
  output_file = "job.sub",
  container_image = NULL,
  executable = NULL,
  input_files = NULL,
  output_files = NULL,
  mode = "single",
  queue = 1L,
  queue_from = NULL,
  resources = "small",
  custom_resources = NULL,
  gpu = FALSE,
  gpu_options = NULL,
  verbose = FALSE,
  comments = FALSE,
  output = "."
)
```

Arguments

output_file	A character string. Name of the submit file to write. Must end in ".sub". Defaults to "job.sub".
container_image	A character string. The container image to use, including the registry prefix, e.g. "docker://registry.doit.wisc.edu/netid/myimage". Defaults to NULL, which writes a placeholder comment in the submit file.
executable	A character string. The shell script that HTCondor will run inside the container, e.g. "analysis.sh". Defaults to NULL, which writes a placeholder comment in the submit file.
input_files	A character vector. Files to transfer to the job's working directory before execution, e.g. c("analysis.R", "data.csv"). In "multiple" mode, the per-job subset file is added automatically from the manifest; use this argument for files shared across all jobs (e.g. the analysis script). Defaults to NULL.
output_files	A character vector. Files to transfer back from the job's working directory after execution. In "multiple" mode, this defaults to "$(file)-results.tar.gz" if not supplied. Defaults to NULL.
mode	A character string. Submission mode. "single" (the default) submits one job. "multiple" submits one job per row in the manifest supplied to queue_from, passing each subset file as a positional argument to the executable via arguments = $(file).
queue	A positive integer. Number of identical jobs to submit. Only used when mode = "single". Defaults to 1.
queue_from	A character string. Path to the manifest file produced by toolero::write_by_group(manifest = TRUE). Required when mode = "multiple". The file_path column is extracted and written alongside the submit file as subdatasets.csv, which HTCondor reads to generate one job per subset file.
resources	A character string. Compute resource preset. One of "small", "medium", "large", or "custom" (requires custom_resources). Default preset values reflect CHTC recommendations and are loaded from inst/extdata/htc-resources.yaml. A local htc-resources.yaml in the working directory takes precedence over the package default, allowing per-project customization. Defaults to "small".
custom_resources	A named list. Required when resources = "custom". Must contain cpus (integer), memory (character, e.g. "8GB"), and disk (character, e.g. "4GB"). Ignored when resources is not "custom".
gpu	Logical. If TRUE, adds GPU resource requests to the submit file. Defaults to FALSE.
gpu_options	A named list or NULL. Fine-grained GPU options applied when gpu = TRUE. Supported keys: request_gpus (integer, default 1), want_gpu_lab (logical, default TRUE), min_capability (numeric, e.g. 8.0 for A100; NULL to omit), min_memory_mb (integer in MB, e.g. 40000; NULL to omit). When gpu = TRUE and gpu_options = NULL, CHTC defaults are used.
verbose	Logical. If TRUE, prints progress messages as each section of the submit file is written. Defaults to FALSE.

comments	Logical. If TRUE, annotates each section with an explanatory comment describing what the section does and how to use it. Defaults to FALSE.
output	A character string. Directory where the submit file (and, in "multiple" mode, subdatasets.csv) will be written. Defaults to "." (current working directory).

Value

Called for its side effects. Writes an HTCondor submit file to file.path(output, output_file). In "multiple" mode also writes subdatasets.csv to output. Returns invisible(NULL).

Multiple-job mode and positional arguments

When mode = "multiple", HTCondor passes each subset filename to the executable as a positional argument via arguments = $(file). Your R script must be written to accept and use this argument. The recommended approach is to use toolero::detect_execution_context() in your analysis script, which resolves the input file path correctly across interactive, Quarto, and Rscript execution contexts:

```
context <- toolero::detect_execution_context()

input_file <- switch(context,
  interactive = "data/penguins.csv",
  quarto      = params$input_file,
  rscript     = commandArgs(trailingOnly = TRUE)[1]
)

data <- readr::read_csv(input_file)
```

The typical workflow is:

1. Write and develop your analysis in analysis.qmd using toolero::detect_execution_context() for data loading.
2. Split your dataset with toolero::write_by_group(manifest = TRUE) to produce subset CSV files and a manifest.csv.
3. Strip analysis.qmd to analysis.R with knitr::purl().
4. Call htc_gen_submit(mode = "multiple", queue_from = "manifest.csv") to produce the submit file and subdatasets.csv.
5. Copy analysis.R, the subset data files, analysis.sub, analysis.sh, and subdatasets.csv to CHTC and submit.

Resource presets

Resource presets are loaded at runtime from inst/extdata/htc-resources.yaml. To customize presets for a specific project, copy that file to your project directory as htc-resources.yaml and edit the values. htc_gen_submit() checks for a local htc-resources.yaml in the working directory first, falling back to the package default if none is found.

Examples

```
# Single-job submit file with default resource preset
htc_gen_submit(output = tempdir())

# Single-job submit file with medium resources and file transfer
htc_gen_submit(
  output_file     = "analysis.sub",
  container_image = "docker://registry.doit.wisc.edu/netid/myimage",
  executable      = "analysis.sh",
  input_files     = "analysis.R",
  output_files    = "results.tar.gz",
  resources       = "medium",
  output          = tempdir()
)

# Annotated submit file useful for learning HTCondor syntax
htc_gen_submit(
  output_file = "annotated.sub",
  comments    = TRUE,
  verbose     = TRUE,
  output      = tempdir()
)

# Custom resource request
htc_gen_submit(
  resources        = "custom",
  custom_resources = list(cpus = 2, memory = "8GB", disk = "4GB"),
  output           = tempdir()
)

## Not run:
# Multiple-job submit file driven by a write_by_group() manifest
htc_gen_submit(
  output_file     = "analysis.sub",
  container_image = "docker://registry.doit.wisc.edu/netid/myimage",
  executable      = "analysis.sh",
  input_files     = "analysis.R",
  mode            = "multiple",
  queue_from      = "data/manifest.csv",
  resources       = "medium",
  output          = "."
)

## End(Not run)
```

`htc_status` *Check the status of submitted HTCondor jobs*

Description

`htc_status()` connects to an HTC submit node via SSH and runs `condor_q` to report the status of jobs in the queue. By default it shows all of your jobs. Optionally filter by cluster ID to monitor a specific submission.

Usage

```
htc_status(
  cluster_id = NULL,
  config = NULL,
  watch = FALSE,
  interval = 60L,
  dry_run = FALSE,
  verbose = FALSE
)
```

Arguments

`cluster_id`	An integer or character string. The cluster ID returned by `htc_submit()`, e.g. `6302860`. If `NULL` (the default), shows all of your jobs currently in the queue. Required when `watch = TRUE`.
`config`	A named list as returned by `htc_config()`. Must contain `username` and `server`. If `NULL`, the function errors with instructions to call `htc_config()` first.
`watch`	Logical. If `TRUE`, polls the queue repeatedly at `interval` seconds until all jobs in `cluster_id` have completed. Requires `cluster_id` to be supplied. Defaults to `FALSE`.
`interval`	A positive integer. Number of seconds to wait between polls when `watch = TRUE`. Defaults to `60`.
`dry_run`	Logical. If `TRUE`, prints the SSH command that would be executed without running it. Defaults to `FALSE`.
`verbose`	Logical. If `TRUE`, prints progress messages. Defaults to `FALSE`.

Details

When `watch = TRUE`, `htc_status()` polls the queue repeatedly at a fixed interval until all jobs in the cluster have completed, printing a timestamped snapshot after each poll.

Value

Called for its side effects. Prints the `condor_q` output to the console. Returns the most recent output invisibly as a character vector.

Job status codes

HTCondor reports each job's status with a single letter:

Code	Meaning

I	Idle – waiting for a matching execute node
R	Running – currently executing
H	Held – paused, usually due to an error
C	Completed – finished successfully
X	Removed – cancelled
S	Suspended

Jobs disappear from `condor_q` once they complete and their output has been transferred back to the submit node. Use `htc_download()` to retrieve completed job output.

Workflow

```
cfg <- htc_config()

# One-shot status check
htc_status(config = cfg)

# Monitor a specific cluster until completion
htc_status(cluster_id = 6302860, config = cfg, watch = TRUE)
```

SSH connection reuse

Each poll in watch mode opens a new SSH connection. Configuring ControlMaster in your `~/.ssh/config` (see `htc_config()`) is strongly recommended when using `watch = TRUE` to avoid repeated Duo MFA prompts.

Examples

```
# Preview the SSH command without connecting to CHTC
cfg <- list(username = "netid", server = "ap2002.chtc.wisc.edu")
htc_status(config = cfg, dry_run = TRUE)

# Preview with a specific cluster ID
htc_status(cluster_id = 6302860, config = cfg, dry_run = TRUE)


## Not run:
# All remaining examples require a live CHTC connection
cfg <- htc_config()

# Check all your jobs
htc_status(config = cfg)

# Check a specific cluster
htc_status(cluster_id = 6302860, config = cfg)

# Watch a cluster until all jobs complete (polls every 60 seconds)
htc_status(cluster_id = 6302860, config = cfg, watch = TRUE)

# Watch with a shorter polling interval
```

```
htc_status(cluster_id = 6302860, config = cfg, watch = TRUE, interval = 30)

## End(Not run)
```

`htc_submit` *Submit an HTCondor job from a remote submit node*

Description

`htc_submit()` connects to an HTC submit node via SSH and runs `condor_submit` on a submit file that has already been uploaded with `htc_upload()`. It changes into the remote directory before submitting so that relative paths in the submit file resolve correctly.

Usage

```
htc_submit(
  submit_file = "job.sub",
  remote_path = "~/",
  config = NULL,
  dry_run = FALSE,
  verbose = FALSE
)
```

Arguments

`submit_file`	A character string. Name of the submit file on the remote node, e.g. `"job.sub"`. Must end in `".sub"`. Defaults to `"job.sub"`.
`remote_path`	A character string. The directory on the submit node where the submit file was uploaded. Defaults to `"~/"`. Must match the `remote_path` used in the preceding call to `htc_upload()`.
`config`	A named list as returned by `htc_config()`. Must contain `username` and `server`. If `NULL`, the function errors with instructions to call `htc_config()` first.
`dry_run`	Logical. If `TRUE`, prints the SSH command that would be executed without running it. Useful for verifying the command before submitting. Defaults to `FALSE`.
`verbose`	Logical. If `TRUE`, prints progress messages and the `condor_submit` output. Defaults to `FALSE`.

Value

The cluster ID assigned by HTCondor as a character string, returned invisibly. Pass it directly to `htc_status()` to monitor job progress. Returns `invisible(NULL)` if the cluster ID cannot be parsed from the `condor_submit` output.

Workflow

`htc_submit()` is the second system-facing step in the submitr workflow. Call it after uploading your files with `htc_upload()`. The returned cluster ID can be passed directly to `htc_status()`.

```
cfg <- htc_config()

htc_upload(
  files  = c("job.sub", "job.sh", "analysis.R"),
  config = cfg
)

cluster_id <- htc_submit(submit_file = "job.sub", config = cfg)
htc_status(cluster_id = cluster_id, config = cfg, watch = TRUE)
```

Why `remote_path` must match `htc_upload()`

`htc_submit()` runs `cd remote_path && condor_submit submit_file` on the submit node. HTCondor resolves all paths in the submit file relative to the directory where `condor_submit` is called. If `remote_path` does not match the directory where files were uploaded, HTCondor will not find the executable, input files, or output destinations and the job will fail.

SSH connection reuse

Each call to `htc_submit()` opens a new SSH connection. If you have not configured ControlMaster in your `~/.ssh/config`, this will trigger a Duo MFA prompt. Run `htc_config()` for setup guidance.

Examples

```
# Preview the SSH command without connecting to CHTC
cfg <- list(username = "netid", server = "ap2002.chtc.wisc.edu")
htc_submit(submit_file = "job.sub", config = cfg, dry_run = TRUE)


## Not run:
# All remaining examples require a live CHTC connection
cfg <- htc_config()

# Submit using default remote path
htc_submit(submit_file = "job.sub", config = cfg)

# Submit from a specific remote directory
htc_submit(
  submit_file = "analysis.sub",
  remote_path = "~/projects/penguins/",
  config      = cfg
)

# Submit with verbose output to see condor_submit response
htc_submit(
```

```
  submit_file = "job.sub",
  config      = cfg,
  verbose     = TRUE
)

## End(Not run)
```

`htc_upload` *Upload files to an HTC submit node*

Description

`htc_upload()` copies one or more local files or directories to a directory on an HTC submit node via `scp`. It is the first step in the job submission workflow – files must be present on the submit node before `htc_submit()` can run `condor_submit`.

Usage

```
htc_upload(
  files,
  remote_path = "~/",
  config = NULL,
  dry_run = FALSE,
  verbose = FALSE
)
```

Arguments

`files`	A character vector. One or more local file paths or directory paths to copy to the submit node. A single file, a vector of files, and a directory path are all accepted. Directories are copied recursively.
`remote_path`	A character string. The destination directory on the submit node. Defaults to `"~/"` (the user's home directory). This should match the path used in the subsequent call to `htc_submit()`.
`config`	A named list as returned by `htc_config()`. Must contain `username` and `server`. If `NULL`, the function errors with instructions to call `htc_config()` first.
`dry_run`	Logical. If `TRUE`, prints the `scp` command that would be executed without running it. Useful for verifying the command before transferring files. Defaults to `FALSE`.
`verbose`	Logical. If `TRUE`, prints progress messages. Defaults to `FALSE`.

Value

Called for its side effects. Returns `invisible(NULL)`.

Workflow

`htc_upload()` is the first system-facing step in the submitr workflow. Call it after generating your submit file and executable script with `htc_gen_submit()` and `htc_gen_executable()`, and before calling `htc_submit()`.

The typical sequence is:

```
cfg <- htc_config()

htc_upload(
  files  = c("job.sub", "job.sh", "analysis.R", "data.csv"),
  config = cfg
)

htc_submit(submit_file = "job.sub", config = cfg)
```

SSH connection reuse

Each call to `htc_upload()` opens a new SSH connection. If you have not configured ControlMaster in your `~/.ssh/config`, this will trigger a Duo MFA prompt. Run `htc_config()` for setup guidance.

Examples

```
# Preview the scp command without connecting to CHTC
cfg <- list(username = "netid", server = "ap2002.chtc.wisc.edu")
tmp <- tempfile(fileext = ".sub")
writeLines("queue 1", tmp)
htc_upload(files = tmp, config = cfg, dry_run = TRUE)


## Not run:
# All remaining examples require a live CHTC connection
cfg <- htc_config()

# Upload a single file
htc_upload(files = "job.sub", config = cfg)

# Upload multiple files
htc_upload(
  files  = c("job.sub", "job.sh", "analysis.R"),
  config = cfg
)

# Upload a directory
htc_upload(files = "jobs/", config = cfg)

# Upload to a specific remote directory
htc_upload(
  files       = c("job.sub", "job.sh"),
  remote_path = "~/projects/penguins/",
```

```
  config      = cfg
)

## End(Not run)
```

Index