# Package 'future.batchtools'

**Version** 0.12.2

**Depends** R (>= 3.2.0), parallelly, future (>= 1.49.0)

**Imports** batchtools (>= 0.9.16), utils

**Suggests** globals, future.apply, listenv, markdown, R.rsp

**VignetteBuilder** R.rsp

**Title** A Future API for Parallel and Distributed Processing using
'batchtools'

**Description** Implementation of the Future API <doi:10.32614/RJ-2021-
048> on top of the 'batchtools' package.
This allows you to process futures, as defined by the 'future' package,
in parallel out of the box, not only on your local machine or ad-hoc
cluster of machines, but also via high-performance compute ('HPC') job
schedulers such as 'LSF', 'OpenLava', 'Slurm', 'SGE', and 'TORQUE' / 'PBS',
e.g. 'y <- future.apply::future_lapply(files, FUN = process)'.

**License** LGPL (>= 2.1)

**LazyLoad** TRUE

**URL** https://future.batchtools.futureverse.org,

https://github.com/futureverse/future.batchtools

**BugReports** https://github.com/futureverse/future.batchtools/issues

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Henrik Bengtsson [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-7579-5165>)

**Maintainer** Henrik Bengtsson <henrikb@braju.com>

**Repository** CRAN

**Date/Publication** 2025-06-06 02:50:02 UTC

# Contents

**Index** **[10](#)**

---

batchtools_custom         *Batchtools futures for custom batchtools configuration*

---

## Description

Batchtools futures for custom batchtools configuration

## Usage

```
batchtools_custom(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  resources = list(),
  workers = NULL,
  conf.file = findConfFile(),
  cluster.functions = NULL,
  registry = list(),
  ...
)
```

## Arguments

| | |
|---|---|
| expr | The R expression to be evaluated |
| envir | The environment in which global environment should be located. |
| substitute | Controls whether expr should be substitute():d or not. |
| globals | (optional) a logical, a character vector, a named list, or a [Globals] object. If TRUE, globals are identified by code inspection based on expr and tweak searching from environment envir. If FALSE, no globals are used. If a character vector, then globals are identified by lookup based their names globals searching from environment envir. If a named list or a Globals object, the globals are used as is. |
| label | (optional) Label of the future (where applicable, becomes the job name for most job schedulers). |

| | |
|---|---|
| resources | (optional) A named list passed to the **batchtools** template (available as variable resources).  See Section 'Resources' in `batchtools::submitJobs()` more details. |
| workers | (optional) The maximum number of workers the batchtools backend may use at any time.  Interactive and "local" backends can only process one future at the time (workers = 1L), whereas HPC backends, where futures are resolved via separate jobs on a scheduler, can have multiple workers. In the latter, the default is workers = NULL, which will resolve to getOption("future.batchtools.workers"). If neither are specified, then the default is 100. |
| conf.file | (character) A batchtools configuration file as for instance returned by `batchtools::findConfFile()`. |
| cluster.functions | |
| | A ClusterFunctions object. |
| registry | (optional) A named list of settings to control the setup of the batchtools registry. |
| ... | Additional arguments passed to `BatchtoolsFuture()`. |

## Value

An object of class BatchtoolsFuture.

## Examples

```
options(error = function(...) {
  print(traceback())
})

cf <- batchtools::makeClusterFunctionsInteractive(external = TRUE)
print(cf)
str(cf)
plan(batchtools_custom, cluster.functions = cf)
print(plan())
print(nbrOfWorkers())

## Create explicit future
f <- future({
  cat("PID:", Sys.getpid(), "\n")
  42L
})
print(f)
v <- value(f)
print(v)

options(error = NULL)


## Create explicit future
f <- future({
  cat("PID:", Sys.getpid(), "\n")
  42L
})
print(f)
```

```
v <- value(f)
print(v)



## Create explicit future
f <- future({
  cat("PID:", Sys.getpid(), "\n")
  42L
})
v <- value(f)
print(v)
```

---

batchtools_local            *batchtools local and interactive futures*

---

### Description

A batchtools local future is an synchronous uniprocess future that will be evaluated in a background
R session. A batchtools interactive future is an synchronous uniprocess future that will be evaluated
in the current R session (and variables will be assigned to the calling environment rather than to a
local one). Both types of futures will block until the futures are resolved.

### Usage

```
batchtools_local(..., envir = parent.frame())
```

### Arguments

| | |
|---|---|
| envir | The environment in which global environment should be located. |
| ... | Additional arguments passed to `BatchtoolsUniprocessFuture()`. |

### Details

batchtools local futures rely on the batchtools backend set up by `batchtools::makeClusterFunctionsInteractive(external`
`= TRUE)` and batchtools interactive futures on the one set up by `batchtools::makeClusterFunctionsInteractive()`.
These are supported by all operating systems.

An alternative to batchtools local futures is to use cluster futures of the **future** package with a single
local background session, i.e. `plan(cluster, workers = "localhost")`.

An alternative to batchtools interactive futures is to use `plan(sequential, split = TRUE)` futures
of the **future** package.

### Value

An object of class `BatchtoolsUniprocessFuture`.

## Examples

```
## Use local batchtools futures
plan(batchtools_local)

## A global variable
a <- 1

## Create explicit future
f <- future({
  b <- 3
  c <- 2
  a * b * c
})
v <- value(f)
print(v)


## Create implicit future
v %<-% {
  b <- 3
  c <- 2
  a * b * c
}
print(v)
```

---

batchtools_template     *Batchtools futures for LSF, OpenLava, SGE, Slurm, TORQUE etc.*

---

## Description

Batchtools futures for LSF, OpenLava, SGE, Slurm, TORQUE etc. are asynchronous multiprocess futures that will be evaluated on a compute cluster via a job scheduler.

## Usage

```
batchtools_lsf(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
  registry = list(),
  ...
)
```

```
batchtools_openlava(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
  registry = list(),
  ...
)

batchtools_sge(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
  registry = list(),
  ...
)

batchtools_slurm(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
  registry = list(),
  ...
)

batchtools_torque(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
```

```
    registry = list(),
    ...
)
```

## Arguments

| | |
|---|---|
| expr | The R expression to be evaluated |
| envir | The environment in which global environment should be located. |
| substitute | Controls whether expr should be substitute():d or not. |
| globals | (optional) a logical, a character vector, a named list, or a [Globals](#) object. If TRUE, globals are identified by code inspection based on expr and tweak searching from environment envir. If FALSE, no globals are used. If a character vector, then globals are identified by lookup based their names globals searching from environment envir. If a named list or a Globals object, the globals are used as is. |
| label | (optional) Label of the future (where applicable, becomes the job name for most job schedulers). |
| template | (optional) A batchtools template file or a template string (in **brew** format). If not specified, it is left to the **batchtools** package to locate such file using its search rules. |
| resources | (optional) A named list passed to the **batchtools** template (available as variable resources). See Section 'Resources' in [batchtools::submitJobs()](#) more details. |
| workers | (optional) The maximum number of workers the batchtools backend may use at any time. Interactive and "local" backends can only process one future at the time (workers = 1L), whereas HPC backends, where futures are resolved via separate jobs on a scheduler, can have multiple workers. In the latter, the default is workers = NULL, which will resolve to getOption(["future.batchtools.workers"](#)). If neither are specified, then the default is 100. |
| registry | (optional) A named list of settings to control the setup of the batchtools registry. |
| ... | Additional arguments passed to [BatchtoolsFuture()](#). |

## Details

These type of batchtools futures rely on batchtools backends set up using the following **batchtools** functions:

- [batchtools::makeClusterFunctionsLSF()](#) for [Load Sharing Facility (LSF)](#)
- [batchtools::makeClusterFunctionsOpenLava()](#) for [OpenLava](#)
- [batchtools::makeClusterFunctionsSGE()](#) for [Sun/Oracle Grid Engine (SGE)](#)
- [batchtools::makeClusterFunctionsSlurm()](#) for [Slurm](#)
- [batchtools::makeClusterFunctionsTORQUE()](#) for [TORQUE](#) / PBS

## Value

An object of class BatchtoolsFuture.

---

future.batchtools             *future.batchtools: A Future for batchtools*

---

### Description

The **future.batchtools** package implements the Future API on top of **batchtools** such that futures can be resolved on for instance high-performance compute (HPC) clusters via job schedulers. The Future API is defined by the **future** package.

### Details

To use batchtools futures, load **future.batchtools**, and select the type of future you wish to use via `future::plan()`.

### Author(s)

**Maintainer**: Henrik Bengtsson <henrikb@braju.com> (ORCID) [copyright holder]

### See Also

Useful links:

- <https://future.batchtools.futureverse.org>
- <https://github.com/futureverse/future.batchtools>
- Report bugs at <https://github.com/futureverse/future.batchtools/issues>

### Examples

```
library(future.batchtools)

## Use local batchtools futures
plan(batchtools_local)

## A global variable
a <- 1

v %<-% {
  b <- 3
  c <- 2
  a * b * c
}

print(v)

plan(batchtools_local)
demo("mandelbrot", package = "future", ask = FALSE)
```

---

future.batchtools.options

*Options used for batchtools futures*

---

### Description

Below are the R options and environment variables that are used by the **future.batchtools** package. See [future::future.options](future::future.options) for additional ones that apply to futures in general.

*WARNING: Note that the names and the default values of these options may change in future versions of the package. Please use with care until further notice.*

### Settings for batchtools futures

'`future.batchtools.workers`': (a positive numeric or +Inf) The default number of workers available on HPC schedulers with job queues. (Default: 100)

'`future.batchtools.output`': (logical) If TRUE, **batchtools** will produce extra output. If FALSE, such output will be disabled by setting **batchtools** options '`batchtools.verbose`' and '`batchtools.progress`' to FALSE. (Default: getOption("future.debug", FALSE))

'`future.batchtools.expiration.tail`': (a positive numeric) When a **batchtools** job expires, the last few lines will be relayed by batchtools futures to help troubleshooting. This option controls how many lines are displayed. (Default: 48L)

'`future.cache.path`': (character string) An absolute or relative path specifying the root folder in which **batchtools** registry folders are stored. This folder needs to be accessible from all hosts ("workers"). Specifically, it must *not* be a folder that is only local to the machine such as file.path(tempdir(), ".future" if an job scheduler on a HPC environment is used. (Default: .future in the current working directory)

'`future.delete`': (logical) Controls whether or not the future's **batchtools** registry folder is deleted after the future result has been collected. If TRUE, it is always deleted. If FALSE, it is never deleted. If not set or NULL, the it is deleted, unless running in non-interactive mode and the batchtools job failed or expired, which helps to troubleshoot when running in batch mode. (Default: NULL (not set))

### Environment variables that set R options

All of the above R '`future.batchtools.*`' options can be set by corresponding environment variable R_FUTURE_BATCHTOOLS_* *when the* **future.batchtools** *package is loaded*. This means that those environment variables must be set before the **future.batchtools** package is loaded in order to have an effect. For example, if R_FUTURE_BATCHTOOLS_WORKERS="200" is set, then option '`future.batchtools.workers`' is set to 200 (numeric).

### Examples

```
# Set an R option:
options(future.cache.path = "/cluster-wide/folder/.future")
```

# Index