Package ‘multideploy’

August 27, 2025

Title Deploy File Changes Across Multiple 'GitHub' Repositories

Version 0.1.0

Description Deploy file changes across multiple 'GitHub' repositories using the 'GitHub' 'Web API' <https://docs.github.com/en/rest>. Allows synchronizing common files, Continuous Integration ('CI') workflows, or configurations across many repositories with a single command.

License AGPL (>= 3)

URL https://r-pkg.thecoatlessprofessor.com/multideploy/, https://github.com/coatless-rpkg/multideploy

BugReports https://github.com/coatless-rpkg/multideploy/issues

Imports gh (>= 1.3.0), base64enc, cli

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author James Joseph Balamuta [aut, cre] (ORCID: <https://orcid.org/0000-0003-2826-8458>)

Maintainer James Joseph Balamuta <james.balamuta@gmail.com>

Repository CRAN

Date/Publication 2025-08-27 17:10:12 UTC

Contents

`file_content` *Retrieve the content of a file from a GitHub repository*

Description

This function fetches a file from a GitHub repository and returns its content and SHA. If the file cannot be retrieved, it returns NULL and optionally displays a warning message.

Usage

```
file_content(repo, path, ref = NULL)
```

Arguments

`repo`	Character string specifying the full name of the repository (format: "owner/repo")
`path`	Character string specifying the path to the file within the repository
`ref`	Character string specifying the branch name, tag, or commit SHA. Default is NULL (uses default branch).

Value

When successful, returns a `list` with two elements:

content Character string containing the decoded file content

sha Character string with the file's blob SHA for use in update operations

When the file cannot be retrieved (e.g., does not exist or no access), returns `NULL`.

Examples

```
# Get content from default branch
file_info <- file_content("username/repository", "path/to/file.R")
if (!is.null(file_info)) {
  # Access the content and SHA
  content <- file_info$content
  sha <- file_info$sha
}

# Get content from specific branch
file_info <- file_content("username/repository", "path/to/file.R", ref = "develop")
```

```
# Suppress warnings
file_info <- file_content("username/repository", "path/to/file.R")
```

file_deploy — *Deploy a file to multiple GitHub repositories*

Description

This function deploys a local file to multiple GitHub repositories. It can create new files or update existing ones, and provides detailed status reporting for each operation.

Usage

```
file_deploy(
  source_file,
  target_path,
  repos,
  commit_message = NULL,
  branch = NULL,
  create_if_missing = TRUE,
  dry_run = FALSE,
  quiet = FALSE
)
```

Arguments

`source_file`	Character string specifying the local file path to deploy
`target_path`	Character string specifying the path in the repositories where the file should be placed
`repos`	Data frame of repositories as returned by `repos()` function, with at least a `full_name` column
`commit_message`	Character string with the commit message. Default automatically generates a message.
`branch`	Character string specifying the branch name. Default is NULL (uses default branch).
`create_if_missing`	Logical indicating whether to create the file if it doesn't exist. Default is TRUE.
`dry_run`	Logical indicating whether to only simulate the changes without making actual commits. Default is FALSE.
`quiet`	Logical; if TRUE, suppresses progress and status messages. Default is FALSE.

Value

Returns a `data.frame` with class `"file_deploy_result"` containing the following columns:

repository Character, the full repository name (owner/repo)

status Character, indicating the operation result with one of these values: "created", "updated", "unchanged", "skipped", "error", "would_create", "would_update"

message Character, a description of the action taken or error encountered

See Also

`print.file_deploy_result()` for a formatted summary of deployment results.

Examples

```
# Get list of repositories
repositories <- repos("my-organization")

# Deploy a workflow file to all repositories
results <- file_deploy(
  source_file = "local/path/to/workflow.yml",
  target_path = ".github/workflows/ci.yml",
  repos = repositories
)

# Filter to see only successfully updated repositories
updated <- results[results$status == "updated", ]

# Check for any errors
errors <- results[results$status == "error", ]
```

`file_mapping` *Create a file mapping for multi-repository deployment*

Description

This function builds a mapping between local files and their target paths in repositories, supporting both individual file mapping and bulk directory processing.

Usage

```
file_mapping(
  ...,
  dir = NULL,
  pattern = NULL,
  target_prefix = "",
  preserve_structure = FALSE,
  quiet = FALSE
)
```

Arguments

`...`	Named arguments where names are local file paths and values are repository paths
`dir`	Character string specifying a directory to search for files. Default is NULL.
`pattern`	Character string with a regular expression pattern to match files in dir. Default is NULL.
`target_prefix`	Character string to prefix to all target paths. Default is "".
`preserve_structure`	Logical indicating whether to preserve directory structure in target. Default is FALSE.
`quiet`	Logical; if TRUE, suppresses information messages. Default is FALSE.

Details

The `dir` argument requires a valid directory path currently on the local filesystem. This directory is scanned for files matching the `pattern` regular expression, and each file is mapped to a target path in repositories. If the directory is not found, an error is thrown.

Value

Returns an object of class `"file_mapping"` (which is just a marked up `"list"`) containing:

- A named list where each entry maps a local file path (name) to a target repository path (value)
- Each key is the full path to a local file
- Each value is the corresponding path where the file should be placed in repositories

See Also

`print.file_mapping()` to display the mapping in a formatted way.

Examples

```
# Map individual files with explicit source-to-target paths
mapping <- file_mapping(
  "local/path/ci.yml" = ".github/workflows/ci.yml",
  "local/path/lint.R" = ".lintr"
)

# Automatically map all R files from a directory to backup/R2/
workflow_mapping <- file_mapping(
  dir = system.file(package = "multideploy"),
  pattern = "\\.R$",
  target_prefix = "backup/R2/"
)

# Preserve directory structure when mapping files
template_mapping <- file_mapping(
  dir = system.file(package = "multideploy"),
  preserve_structure = TRUE
)

# Combine explicit mappings with directory-based mappings
combined_mapping <- file_mapping(
  "specific/file.R" = "R/functions.R",
  dir = system.file(package = "multideploy"),
  target_prefix = ".github/"
)
```

`file_update` *Create or update a file in a GitHub repository*

Description

This function creates a new file or updates an existing file in a GitHub repository. For updating existing files, the SHA of the current file must be provided.

Usage

```
file_update(
  repo,
  path,
  content,
  message,
  branch = NULL,
  sha = NULL,
  quiet = FALSE
)
```

Arguments

`repo`	Character string specifying the full name of the repository (format: "owner/repo")
`path`	Character string specifying the path to the file within the repository
`content`	Character string with the new content of the file
`message`	Character string with the commit message
`branch`	Character string specifying the branch name. Default is NULL (uses default branch).
`sha`	Character string with the blob SHA of the file being replaced. Required for updating existing files; omit for creating new files. Default is NULL.
`quiet`	Logical; if TRUE, suppresses progress and status messages. Default is FALSE.

Value

When successful, returns a `list` containing the GitHub API response with details about the commit, including:

content Information about the updated file

commit Details about the created commit

When the operation fails (e.g., permission issues, invalid SHA), returns `NULL`.

Examples

```
# Create a new file
result <- file_update(
  repo = "username/repository",
  path = "path/to/new_file.R",
  content = "# New R script\n\nprint('Hello world')",
  message = "Add new script file"
)
# Check if operation was successful
if (!is.null(result)) {
  # Access commit information
  commit_sha <- result$commit$sha
}

# Update an existing file (requires SHA)
file_info <- file_content("username/repository", "path/to/existing_file.R")
if (!is.null(file_info)) {
  result <- file_update(
    repo = "username/repository",
    path = "path/to/existing_file.R",
    content = "# Updated content\n\nprint('Hello updated world')",
    message = "Update file content",
    sha = file_info$sha
  )
}
```

`orgs` *List organizations for the authenticated user*

Description

This function retrieves all organizations associated with the currently authenticated GitHub user, with options to control pagination.

Usage

```
orgs(per_page = 100, max_pages = 5, quiet = FALSE)
```

Arguments

`per_page`	Number of organizations to return per page. Default is 100.
`max_pages`	Maximum number of pages to retrieve. Default is 5.
`quiet`	Logical; if TRUE, suppresses progress and status messages. Default is FALSE.

Value

Returns a `data.frame` of organizations with the following columns:

login Character, the organization's username/login name

url Character, the API URL for the organization

The `data.frame` is ordered as returned by the GitHub API (typically alphabetically).

Examples

```
# Get all organizations for the authenticated user
my_orgs <- orgs()

# Retrieve silently without progress messages
my_orgs <- orgs(quiet = TRUE)

# Extract just the organization names
org_names <- orgs()$login
```

`print.file_deploy_result` *Print method for* `"file_deploy_result"` *objects*

Description

This method provides a formatted summary of file deployment results, showing counts by status and details for any errors encountered.

Usage

```
## S3 method for class 'file_deploy_result'
print(x, ...)
```

Arguments

`x`	An object of class `"file_deploy_result"` as returned by `file_deploy()`
`...`	Additional arguments passed to other print methods (not used)

Value

Invisibly returns the original input data frame unchanged.

Displays a formatted summary of deployment results to the console.

Examples

```
# Get list of repositories
repositories <- repos("my-organization")

# Deploy files
results <- file_deploy("local/file.R", "remote/file.R", repositories)

# Explicitly print the summary
print(results)
```

`print.file_mapping` *Print method for file_mapping objects*

Description

This method provides a formatted display of file mappings, showing the relationship between local files and their target repository paths with visual indicators for file existence.

Usage

```
## S3 method for class 'file_mapping'
print(x, max_files = 20, ...)
```

Arguments

`x`	An object of class "`file_mapping`" as returned by `file_mapping()`
`max_files`	Maximum number of files to display. Default is 20.
`...`	Additional arguments passed to other print methods (not used)

Value

Invisibly returns the original `file_mapping` object unchanged, allowing for chained operations.

Displays a formatted representation of the mapping to the console, including:

- Total count of mapped files
- Visual indicators showing which local files exist (checkmark) or are missing (x)
- Source-to-target mapping for each file (limited by `max_files`)

Examples

```
# Create and display a mapping
mapping <- file_mapping(
  "R/functions.R" = "R/utils.R",
  dir = system.file(package = "multideploy")
)
# The mapping is automatically printed when not assigned

# Control how many files are displayed
mapping <- file_mapping(dir = system.file(package = "multideploy"))
print(mapping, max_files = 5)  # Show only first 5 mappings
```

print.pr_create_result   *Print method for pr_create_result objects*

Description

This method provides a formatted summary of pull request creation results, showing counts by status and details for created PRs and any errors encountered.

Usage

```
## S3 method for class 'pr_create_result'
print(x, ...)
```

Arguments

x	An object of class "pr_create_result" as returned by `pr_create()`
...	Additional arguments passed to other print methods (not used)

Value

Invisibly returns the original input data frame (x) unchanged, allowing for chained operations. The function's primary purpose is displaying a formatted summary to the console, including:

- Counts of PRs by status (created, would_create, skipped, error)
- List of successfully created PRs with clickable URLs
- Details about any errors encountered during the process

Examples

```
# Create PRs
results <- pr_create(
  repos = repos("my-organization"),
  branch_name = "feature-branch",
  title = "Update configuration",
  body = "Standardize configuration across repos",
  file_mapping = file_mapping("config.yml" = ".github/config.yml")
)

print(results)  # Explicitly print the summary
```

---

pr_create *Create a pull request for changes in multiple repositories*

Description

This function creates pull requests across multiple GitHub repositories, applying the same set of file changes to each repository. It can create new branches as needed, add or update files, and then open pull requests.

Usage

```
pr_create(
  repos,
  branch_name,
  base_branch = NULL,
  title,
  body,
  create_branch = TRUE,
  file_mapping,
  dry_run = FALSE,
  quiet = FALSE
)
```

Arguments

`repos`	Data frame of repositories as returned by `repos()`, with at least columns for `full_name` and `default_branch`
`branch_name`	Character string with the name of the branch to create for the changes
`base_branch`	Character string with the name of the base branch. Default is NULL (uses default branch).
`title`	Character string with the PR title
`body`	Character string with the PR description
`create_branch`	Logical indicating whether to create the branch if it doesn't exist. Default is TRUE.
`file_mapping`	List mapping local file paths to repository paths, as created by `file_mapping()`
`dry_run`	Logical indicating whether to only simulate the changes. Default is FALSE.
`quiet`	Logical; if TRUE, suppresses progress and status messages. Default is FALSE.

Value

Returns a `data.frame` with class `"pr_create_result"` containing the following columns:

repository Character, the full repository name (owner/repo)

pr_url Character, the URL of the created pull request, or NA if no PR was created

status Character, indicating the operation result: "created", "would_create", "skipped", or "error"

message Character, a description of the action taken or error encountered

See Also

`print.pr_create_result()` to display the results in a formatted way.

Examples

```
# Get repositories and create file mapping
repositories <- repos("my-organization")
mapping <- file_mapping(
  "local/path/file1.R" = ".github/workflows/ci.yml",
  "local/path/file2.R" = "R/utils.R"
)

# Create pull requests in all repositories
results <- pr_create(
  repos = repositories,
  branch_name = "feature-branch",
  title = "Update CI workflow",
  body = "Standardizing CI workflow across repositories",
  file_mapping = mapping
)

# Simulate without making actual changes
dry_run_results <- pr_create(
  repos = repositories,
  branch_name = "feature-branch",
  title = "Update documentation",
  body = "Updating documentation with new examples",
  file_mapping = mapping,
  dry_run = TRUE
)

# Only create PRs in repositories where the branch already exists
existing_branch_results <- pr_create(
  repos = repositories,
  branch_name = "existing-branch",
  title = "Fix existing branch",
  body = "Apply fixes to existing branch",
  file_mapping = mapping,
  create_branch = FALSE
)
```

repos — *List repositories for a user or organization*

Description

This function fetches repository information from GitHub for a specified user or organization, with options to filter and limit the results.

Usage

```
repos(
  owner,
  type = "owner",
  per_page = 100,
  max_pages = 10,
  filter_regex = NULL,
  quiet = FALSE
)
```

Arguments

`owner`	Character string specifying the GitHub username or organization name
`type`	Character string specifying the type of repositories to list: "all", "owner", "public", "private", or "member". Default is "owner".
`per_page`	Number of repositories to return per page. Default is 100.
`max_pages`	Maximum number of pages to retrieve. Default is 10.
`filter_regex`	Optional regular expression to filter repositories by name
`quiet`	Logical; if TRUE, suppresses progress and status messages. Default is FALSE.

Value

Returns a `data.frame` of repositories with the following columns:

name Character, repository name without owner prefix

full_name Character, complete repository identifier (owner/repo)

default_branch Character, the name of the default branch (e.g., "main" or "master")

private Logical, TRUE if repository is private, FALSE if public

Examples

```
# Get all repositories owned by a user
user_repos <- repos("username")

# Get only public repositories for an organization
org_public_repos <- repos("orgname", type = "public")
```

```
# Filter repositories by name pattern
api_repos <- repos("orgname", filter_regex = "^api-")

# Limit the number of fetched repositories
limited_repos <- repos("large-org", per_page = 50, max_pages = 2)
```

Index