

Package: trama.github (via r-universe)

October 22, 2026

Title GitHub Connection for trama

Version 0.2.2

Description Connects trama to GitHub without git or a terminal:
account guide, login by the OAuth device flow, reuse of
existing credentials, token kept in the system keyring, and an
internal API to create repositories and make atomic commits and
tags through the Git data API. It is a trama plugin, not a
block collection: it has no nodes. When installed, the trama
editor shows a GitHub menu.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1)

Imports trama (>= 0.5.26), gitcreds, http2 (>= 1.0.0), jsonlite,
keyring, later, openssl, shiny, stats, utils

Suggests testthat (>= 3.0.0), webfakes, withr

Config/testthat/edition 3

Config/trama/plugin true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake git make libsecret-1-dev libuv1-dev libssl-dev zlib1g-dev

Repository <https://uaipetro.r-universe.dev>

Date/Publication 2026-10-22 09:49:44 UTC

RemoteUrl <https://github.com/uaipetro/trama>

RemoteRef HEAD

RemoteSha b1ae926d83b864b7eee3a29b5753b2ad39cae62a

RemoteSubdir plugins/trama.github

Contents

tr_github_ambiente_tags	2
tr_github_api_url	3
tr_github_chaveiro	3
tr_github_client_id	4
tr_github_commit	4
tr_github_criar_repo	5
tr_github_device_iniciar	6
tr_github_execucoes	7
tr_github_guia_conta	8
tr_github_pages	8
tr_github_release	9
tr_github_restaurar	9
tr_github_salvar_token	10
tr_github_salvar_versao	10
tr_github_tag	11
tr_github_tags	11
tr_github_token	12
tr_github_usuario	12
tr_github_versionar_arquivos	13
tr_github_versionar_config	13
tr_github_versionar_criar	14
tr_github_versoes	14
trama_plugin	15
Index	16

tr_github_ambiente_tags	
<i>Deixa uma tag publicar no ambiente do Pages</i>	

Description

O ambiente github-pages, que o GitHub cria ao ligar o Pages, só aceita publicação da branch padrão. A publicação do trama sai de um workflow disparado por tag (vX.Y.Z): sem uma regra para as tags, o deploy é recusado. A regra é acrescentada, e regra igual já existente não é erro.

Usage

```
tr_github_ambiente_tags(  
  repo,  
  ambiente = "github-pages",  
  padrao = "v*",  
  token = NULL  
)
```

Arguments

repo	Repositório como "dono/nome".
ambiente	Nome do ambiente.
padrao	Padrão de nome das tags que podem publicar.
token	Token do GitHub; o padrão é o que <code>tr_github_token()</code> achar.

Value

Invisível, TRUE.

tr_github_api_url	<i>Endereços do GitHub</i>
-------------------	----------------------------

Description

`tr_github_api_url()` é a API REST (<https://api.github.com>) e `tr_github_url()` é o site (<https://github.com>), onde moram as páginas do device flow e do cadastro. Os dois mudam por opção (`trama.github.api_url`, `trama.github.url`) ou variável de ambiente (`TRAMA_GITHUB_API_URL`, `TRAMA_GITHUB_URL`): é assim que os testes apontam para um GitHub falso.

Usage

```
tr_github_api_url()
```

```
tr_github_url()
```

Value

Uma URL sem barra no fim.

tr_github_chaveiro	<i>Onde o token do trama fica guardado</i>
--------------------	--

Description

Onde o token do trama fica guardado

Usage

```
tr_github_chaveiro()
```

Value

Lista com backend (nome do backend do keyring) e persistente (FALSE quando o token só vale enquanto o R estiver aberto).

tr_github_client_id	<i>Client ID do OAuth App usado no login</i>
---------------------	--

Description

Ordem: `getOption("trama.github.client_id")`, variável de ambiente `TRAMA_GITHUB_CLIENT_ID` e, por fim, o Client ID do app "trama" fixado no pacote. Só identifica o app; não é segredo.

Usage

```
tr_github_client_id(obrigatorio = TRUE)
```

Arguments

`obrigatorio` Se TRUE, erro claro quando não há nenhum configurado.

Value

O Client ID, ou "" quando não há e `obrigatorio = FALSE`.

tr_github_commit	<i>Commit atômico pela API de dados do Git</i>
------------------	--

Description

Grava arquivos num repositório sem git instalado: cria os blobs, monta a árvore sobre a do último commit, cria o commit e só então move a branch. Se a rede cair no meio, sobram objetos soltos que o GitHub descarta, e a branch continua onde estava: repetir o envio é seguro.

Usage

```
tr_github_commit(  
  repo,  
  mensagem,  
  arquivos = character(),  
  conteudos = list(),  
  apagar = character(),  
  branch = NULL,  
  manter_outros = TRUE,  
  pular_iguais = FALSE,  
  token = NULL  
)
```

Arguments

repo	Repositório como "dono/nome".
mensagem	Mensagem do commit.
arquivos	Arquivos locais: vetor nomeado caminho no repositório = caminho no disco.
conteudos	Conteúdo em memória: lista nomeada caminho no repositório = texto ou raw.
apagar	Caminhos a remover do repositório.
branch	Branch; NULL usa a padrão do repositório.
manter_outros	TRUE (padrão) mantém os arquivos que já estão no repositório; FALSE faz a árvore ter exatamente os arquivos enviados.
pular_iguais	Não reenviar arquivo que o repositório já tem igual (compara o sha de blob do git, calculado aqui). Custa uma leitura da árvore e poupa o envio de tudo o que não mudou.
token	Token do GitHub; o padrão é o que <code>tr_github_token()</code> achar.

Value

Lista com sha, url e mudou (FALSE quando a árvore ficou igual e nenhum commit foi criado).

tr_github_criar_repo	<i>Repositórios</i>
----------------------	---------------------

Description

`tr_github_criar_repo()` cria um repositório já com um primeiro commit (`auto_init`): a API de dados do Git recusa gravar num repositório vazio. `tr_github_repo()` lê um repositório e devolve NULL se ele não existe.

Usage

```
tr_github_criar_repo(  
  nome,  
  privado = TRUE,  
  descricao = NULL,  
  dono = NULL,  
  token = NULL  
)  
  
tr_github_repo(repo, token = NULL)
```

Arguments

nome	Nome do repositório (sem o dono).
privado	Repositório privado (o padrão) ou público.
descricao	Descrição curta, opcional.
dono	Organização onde criar; NULL cria na conta de quem entrou.
token	Token do GitHub; o padrão é o que <code>tr_github_token()</code> achar.
repo	Repositório como "dono/nome".

Value

Lista com repo ("dono/nome"), url, privado e branch (a branch padrão).

tr_github_device_iniciar

Login pelo device flow do GitHub

Description

O mesmo caminho do `gh auth login`: o trama pede um código ao GitHub, a pessoa abre o link no navegador, digita o código e autoriza. Não precisa de client secret nem de servidor de retorno.

Usage

```
tr_github_device_iniciar(client_id = tr_github_client_id(), escopos = .ESCOPOS)
```

```
tr_github_device_conferir(pedido)
```

```
tr_github_entrar(abrir = interactive(), salvar = TRUE)
```

Arguments

client_id	Client ID do OAuth App (ver <code>tr_github_client_id()</code>).
escopos	Escopos pedidos.
pedido	O que <code>tr_github_device_iniciar()</code> devolveu.
abrir	Abrir o link no navegador.
salvar	Guardar o token no chaveiro.

Details

`tr_github_device_iniciar()` pede o código; `tr_github_device_conferir()` pergunta uma vez se a pessoa já autorizou (o editor chama em intervalos sem travar a tela); `tr_github_entrar()` faz tudo no console e guarda o token no chaveiro.

Value

tr_github_device_iniciar(): lista com código (o que a pessoa digita), link, device_code, intervalo (segundos entre conferências) e expira_em.

tr_github_device_conferir(): lista com estado ("pendente", "ok", "expirado" ou "negado"), intervalo e, quando "ok", token.

tr_github_entrar(): o usuário conectado (ver [tr_github_usuario\(\)](#)), invisível.

tr_github_execucoes	<i>Execuções do GitHub Actions</i>
---------------------	------------------------------------

Description

tr_github_execucoes() lista as execuções de um commit (o da tag); tr_github_falha() diz em que job e etapa uma execução falhou.

Usage

```
tr_github_execucoes(repo, sha = NULL, token = NULL)
```

```
tr_github_falha(repo, id, token = NULL)
```

Arguments

repo	Repositório como "dono/nome".
sha	Commit (head_sha) cujas execuções se quer; NULL traz as mais recentes do repositório, a última primeiro.
token	Token do GitHub; o padrão é o que tr_github_token() achar.
id	Id da execução.

Value

tr_github_execucoes(): lista de execuções, cada uma com id, nome, estado (queued, in_progress, completed...), conclusao (success, failure... ou NA enquanto roda) e url. tr_github_falha(): lista com job, etapa e url (do log), ou NULL.

tr_github_guia_conta	<i>Guia de criação de conta no GitHub</i>
----------------------	---

Description

O GitHub exige uma pessoa no cadastro, então o trama não cria a conta: abre a página certa e explica cada passo. O editor mostra os mesmos passos no menu GitHub.

Usage

```
tr_github_guia_conta(abrir = FALSE)
```

Arguments

abrir	Abrir a página de cadastro no navegador.
-------	--

Value

Lista de passos, cada um com título, texto e, quando há, link e rotulo (texto do link). Invisível quando imprime no console.

tr_github_pages	<i>GitHub Pages em modo Actions</i>
-----------------	-------------------------------------

Description

tr_github_pages() lê a configuração (NULL se o Pages está desligado). tr_github_ativar_pages() liga em modo Actions (o site sai do workflow, não de uma branch) ou, já ligado, troca o modo para Actions. Repositório privado só tem Pages em plano pago: o GitHub recusa, e o erro diz isso.

Usage

```
tr_github_pages(repo, token = NULL)
```

```
tr_github_ativar_pages(repo, token = NULL)
```

Arguments

repo	Repositório como "dono/nome".
token	Token do GitHub; o padrão é o que tr_github_token() achar.

Value

Lista com url (endereço do site), modo ("workflow" ou "legacy") e estado.

tr_github_release	<i>Release de uma tag</i>
-------------------	---------------------------

Description

Release de uma tag

Usage

```
tr_github_release(repo, tag, token = NULL)
```

Arguments

repo	Repositório como "dono/nome".
tag	Nome da tag ("v1.0.0").
token	Token do GitHub; o padrão é o que <code>tr_github_token()</code> achar.

Value

NULL se a release ainda não existe; senão lista com tag, url e arquivos (nome -> URL de download).

tr_github_restaurar	<i>Restaurar uma versão</i>
---------------------	-----------------------------

Description

Traz os arquivos do projeto de volta a como estavam numa versão salva. Tudo é baixado antes de mexer em qualquer arquivo: se a rede cair, nada muda. Antes de escrever, o estado atual de cada arquivo que vai ser trocado ou removido é copiado para `.trama/restauracao/<data-hora>/.trama/`. Arquivos que não vão em versão (`.trama/`, dados não marcados) não são tocados. No `trama.json`, as chaves `versionar` e `publicacoes` ficam como estão agora: a ligação com o GitHub não volta no tempo.

Usage

```
tr_github_restaurar(root = ".", sha, token = NULL)
```

Arguments

root	Pasta do projeto.
sha	Commit da versão (de <code>tr_github_versoes()</code>).
token	Token do GitHub; o padrão é o que <code>tr_github_token()</code> achar.

Value

Lista com escritos, removidos (caminhos) e copia (pasta com o estado anterior, ou NA se nada foi trocado).

tr_github_salvar_token

Guarda e apaga o token do trama no chaveiro

Description

tr_github_salvar_token() é o que o login faz no fim; serve também para colar um token criado à mão. tr_github_sair() apaga só o que o trama guardou: GITHUB_PAT, gh e gitcreds são de outras ferramentas e continuam valendo.

Usage

```
tr_github_salvar_token(token)
```

```
tr_github_sair()
```

Arguments

token	Token do GitHub.
-------	------------------

tr_github_salvar_versao

Salvar versão

Description

Grava no repositório privado do projeto um commit com exatamente os arquivos que vão numa versão (ver [tr_github_versionar_arquivos\(\)](#)). Arquivo apagado do projeto, ou dado desmarcado, some da versão nova e continua nas anteriores. Só o que mudou é enviado.

Usage

```
tr_github_salvar_versao(root = ".", mensagem, token = NULL)
```

Arguments

root	Pasta do projeto.
mensagem	Uma frase sobre o que mudou.
token	Token do GitHub; o padrão é o que tr_github_token() achar.

Value

Lista com sha, url, mudou (FALSE se nada mudou desde a última versão), arquivos e bytes.

tr_github_tag	<i>Cria uma tag</i>
---------------	---------------------

Description

Tag já existente é recusada (erro de classe tr_github_erro_tag_existe): versão publicada nunca muda.

Usage

```
tr_github_tag(repo, tag, sha = NULL, mensagem = NULL, token = NULL)
```

Arguments

repo	Repositório como "dono/nome".
tag	Nome da tag, por exemplo "v1.0.0".
sha	Commit marcado; NULL usa o último da branch padrão.
mensagem	Com mensagem, cria tag anotada; sem, tag leve.
token	Token do GitHub; o padrão é o que tr_github_token() achar.

Value

Lista com tag, sha (do commit) e url.

tr_github_tags	<i>Tags de um repositório</i>
----------------	-------------------------------

Description

Tags de um repositório

Usage

```
tr_github_tags(repo, token = NULL)
```

Arguments

repo	Repositório como "dono/nome".
token	Token do GitHub; o padrão é o que tr_github_token() achar.

Value

Vetor de nomes de tag (até 100, as mais recentes primeiro, na ordem da API).

tr_github_token	<i>Credencial do GitHub</i>
-----------------	-----------------------------

Description

tr_github_token() procura, nesta ordem:

Usage

```
tr_github_token()
```

Details

1. o token que o próprio trama guardou no chaveiro do sistema, no login;
2. as variáveis de ambiente GITHUB_PAT e GITHUB_TOKEN;
3. o gh (GitHub CLI), por gh auth token;
4. o gerenciador de credenciais do git, pelo gitcreds (o mesmo que o usethis usa).

O token nunca é gravado no fluxo, no projeto ou no pacote publicado: só no chaveiro (keyring, serviço trama-github).

Value

O token (texto) com o atributo fonte ("chaveiro", "GITHUB_PAT", "GITHUB_TOKEN", "gh" ou "gitcreds"), ou NULL se não houver.

tr_github_usuario	<i>Usuário conectado</i>
-------------------	--------------------------

Description

Usuário conectado

Usage

```
tr_github_usuario(token = NULL)
```

Arguments

token	Token do GitHub; o padrão é o que tr_github_token() achar.
-------	--

Value

Lista com login, nome, avatar, perfil (URL) e plano.

`tr_github_versionar_arquivos`*Arquivos do projeto e o que vai numa versão*

Description

Classifica os arquivos do projeto. Vão sempre (tipo = "projeto"): trama.json, fluxos, código (R, Rmd, qmd, py, sql), metadados (md, cff, yml, toml...) e as imagens das notas. São dados (tipo = "dados") todos os outros: ficam de fora até serem marcados. Não aparecem, e nunca vão: .trama/ (store, execuções, token do editor), .git/, arquivos de sessão do R, .Renviron, .env e chaves.

Usage

```
tr_github_versionar_arquivos(root = ".", dados = NULL)
```

Arguments

root	Pasta do projeto.
dados	Arquivos de dados marcados; o padrão é o que está salvo no trama.json.

Value

Data frame com caminho, tamanho (bytes), tipo, incluído, bloqueado (acima de 100 MB: o GitHub recusa) e aviso (texto ou NA).

`tr_github_versionar_config`*Configuração do versionamento do projeto*

Description

`tr_github_versionar_config()` lê a chave versionar do trama.json (NULL se o projeto nunca foi versionado). `tr_github_versionar_marcar()` marca ou desmarca arquivos de dados para entrar nas próximas versões.

Usage

```
tr_github_versionar_config(root = ".")
```

```
tr_github_versionar_marcar(root = ".", caminhos, incluir = TRUE)
```

Arguments

root	Pasta do projeto.
caminhos	Caminhos relativos à pasta do projeto.
incluir	TRUE marca, FALSE desmarca.

Value

A configuração (lista), invisível em `tr_github_versionar_marcar()`.

<code>tr_github_versionar_criar</code>	<i>Repositório privado do projeto</i>
--	---------------------------------------

Description

Cria o repositório privado onde ficam as versões do projeto e o registra no `trama.json`. É outro repositório, nunca o de uma publicação: o rascunho não vaza junto com o que se publica.

Usage

```
tr_github_versionar_criar(root = ".", nome = NULL, token = NULL)

tr_github_versionar_nome(root = ".")
```

Arguments

<code>root</code>	Pasta do projeto.
<code>nome</code>	Nome do repositório; o padrão vem do nome da pasta.
<code>token</code>	Token do GitHub; o padrão é o que <code>tr_github_token()</code> achar.

Value

A configuração gravada.
`tr_github_versionar_nome()`: nome sugerido (pasta em minúsculas, sem acento, com hífen).

<code>tr_github_versoes</code>	<i>Versões salvas</i>
--------------------------------	-----------------------

Description

Versões salvas

Usage

```
tr_github_versoes(root = ".", n = 30, token = NULL)
```

Arguments

<code>root</code>	Pasta do projeto.
<code>n</code>	Quantas, das mais recentes.
<code>token</code>	Token do GitHub; o padrão é o que <code>tr_github_token()</code> achar.

Value

Data frame com sha, mensagem (primeira linha), data (POSIXct, UTC), autor e url, a mais recente primeiro.

trama_plugin

Menu GitHub no editor

Description

Contrato de plugin do trama (ver `trama::tr_plugins()`): o editor carrega `inst/trama/github.js` e liga o server abaixo a cada sessão. O canal é próprio: o JS manda `input$trama_github ({acao})` e o R responde com a mensagem `trama_github ({tipo, ...})`.

Usage

```
trama_plugin()
```

Value

Lista com js, css e server.

Index

tr_github_ambiente_tags, [2](#)
tr_github_api_url, [3](#)
tr_github_ativar_pages
 (tr_github_pages), [8](#)
tr_github_chaveiro, [3](#)
tr_github_client_id, [4](#)
tr_github_client_id(), [6](#)
tr_github_commit, [4](#)
tr_github_criar_repo, [5](#)
tr_github_device_conferir
 (tr_github_device_iniciar), [6](#)
tr_github_device_iniciar, [6](#)
tr_github_entrar
 (tr_github_device_iniciar), [6](#)
tr_github_execucoes, [7](#)
tr_github_falha (tr_github_execucoes), [7](#)
tr_github_guia_conta, [8](#)
tr_github_pages, [8](#)
tr_github_release, [9](#)
tr_github_repo (tr_github_criar_repo), [5](#)
tr_github_restaurar, [9](#)
tr_github_sair
 (tr_github_salvar_token), [10](#)
tr_github_salvar_token, [10](#)
tr_github_salvar_versao, [10](#)
tr_github_tag, [11](#)
tr_github_tags, [11](#)
tr_github_token, [12](#)
tr_github_token(), [3](#), [5–12](#), [14](#)
tr_github_url (tr_github_api_url), [3](#)
tr_github_usuario, [12](#)
tr_github_usuario(), [7](#)
tr_github_versionar_arquivos, [13](#)
tr_github_versionar_arquivos(), [10](#)
tr_github_versionar_config, [13](#)
tr_github_versionar_criar, [14](#)
tr_github_versionar_marcar
 (tr_github_versionar_config),
 [13](#)
tr_github_versionar_nome
 (tr_github_versionar_criar), [14](#)
tr_github_versoes, [14](#)
tr_github_versoes(), [9](#)
trama_plugin, [15](#)