

# Git + RStudio + GitHub via SSH

Como configurar commits e controle de versão a partir do RStudio conectado a um servidor Linux, integrado diretamente a uma conta no GitHub.

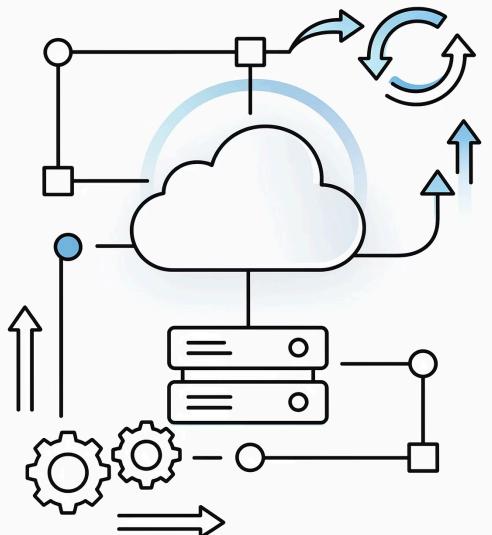
TUTORIAL

CONTROLE DE VERSÃO

Eduardo Ogasawara

[eduardo.ogasawara@cefet-rj.br](mailto:eduardo.ogasawara@cefet-rj.br)

<https://eic.cefet-rj.br/~eogasawara>



# Por que Versionar no Servidor?

## Controle de Código

Versione e acompanhe todo o código rodando remotamente no servidor Linux.

## Backup Seguro

Mantenha backup de todas as operações realizadas no ambiente remoto.

## Colaboração

Trabalhe em equipe de forma eficiente, com histórico completo de alterações.

# Passo 1: Configurar a Chave SSH no RStudio

The image consists of three side-by-side screenshots from the RStudio interface:

- Acessar Git**: Shows the "General" section of the "Options" dialog. The "Enable version control interface for RStudio projects" checkbox is checked. Under "Git executable", the path "/usr/bin/git" is listed with a "Browse..." button. Under "SVN executable", it says "(Not Found)" with a "Browse..." button. Under "SSH RSA key", it says "(None)" with a "Create RSA Key..." button. A link "Using Version Control with RStudio" is also visible.
- Criar chave RSA**: Shows the "Create RSA Key" dialog. It asks for the location where the key will be created: "/home/[REDACTED]/.ssh/id\_rsa". It has two input fields for "Passphrase (optional)" and "Confirm". At the bottom are "Create" and "Cancel" buttons.
- Reiniciar sessão**: Shows the "Create RSA Key" dialog again, but this time it's displaying the generated key pair. It shows the public key fingerprint: "SHA256:CfEEsoH1+zTlqvHZB5KdGFIhW8K9FMwkdmLStCCIjQ [REDACTED]" and the key's randomart image.

No RStudio, acesse as **configurações globais** e vá até a seção **Git**. O Git deve estar instalado como pré-requisito. Crie uma chave RSA — opcionalmente com passphrase para maior segurança. O sistema gera um arquivo privado e um arquivo público. Após aplicar, reinicie a sessão para confirmar a configuração.

## Passo 2: Copiar a Chave Pública no Servidor

No terminal do servidor, acesse a pasta `~/ssh` e localize o arquivo de chave pública gerado pelo RStudio.

```
:[~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQCoumGPcmQ90e4kXouF6alw
V04V+1lkhgX8okWfnzOyFDYQpf6wgBGNBcAHRxPd6FdJ20bJr47a4V8wPrqQ
bhd0Q30tAQszSwpf5QBHnyiAlPrdDq+6akRnW76PJ7uVxkpTh6LXwrrzYTE
ajV+wvx6vxd0ec+qEEbBXMEzM1gEBhZygw7gAyJC12exh75QcH8afwANRBD8
34f3iDvB1YmY0ecnjnbJHem4iTKhwMCPT3AaRsw36Ggc7W0uqLEqXRWZLk4d
Ssj7K036olsphyu1Bx0d5aeV0SiTvvi5GUFnixe6aDSw9PGsQ8/9I517rpFh
4+R/aXyd47vv9SXE9xEmG8mRDn0ryH1AL+7h0vqBs2nYVLKeQm7rX3Jsc9I3
zBY/fM1qY3ubZWVFxRx2hsX9t2BXk8Vh16W/SjaF5sjYXEgC8dcGFQHtCr9m
2PxyN21n9D7HfNNqwK4HEXH2ZT/W3VFLFD53THyCkwS7mlFMspJj/mhGamQ4
ltj/Cct21eM=
```

```
cat ~/.ssh/id_rsa.pub
```

- Copie todo o conteúdo exibido — da primeira à última linha. Esse é o texto que será registrado no GitHub.

# Passo 3: Registrar a Chave SSH no GitHub

Com a chave pública copiada, acesse sua conta no GitHub e registre-a nas configurações de acesso SSH.



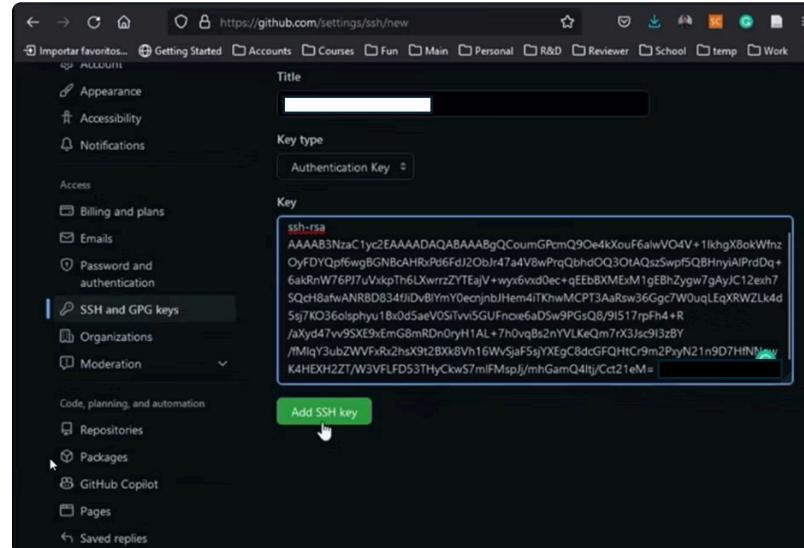
## Acessar SSH Keys

Vá em Settings → SSH and GPG Keys e clique em New SSH Key.

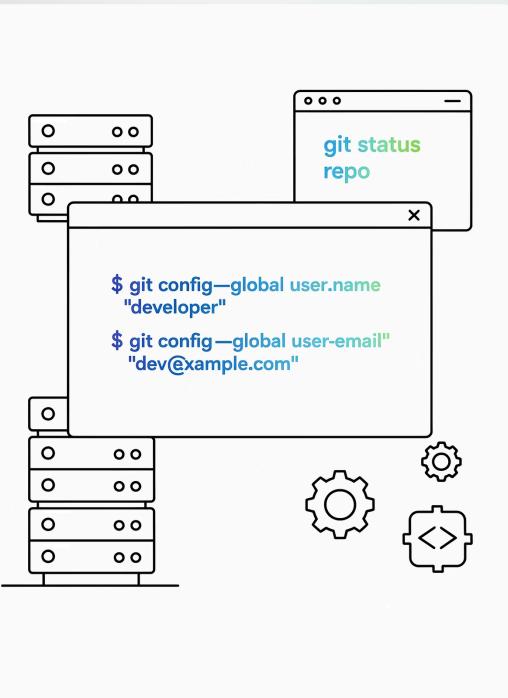


## Colar e Salvar

Dê um nome identificador, cole o conteúdo da chave pública e clique em Add SSH Key.



□ Atenção: o RStudio pode ter usado um e-mail local ao gerar a chave. Certifique-se de associar o e-mail correto da conta GitHub.



## Passo 4: Configurar Usuário Git Global

No terminal do servidor, navegue até a pasta raiz e execute os comandos de configuração global do Git para associar sua identidade aos commits:

```
git config --global user.email "seu@email.github.com"  
git config --global user.name "<usuario>"
```

Esses dados serão usados em todos os commits realizados a partir desse servidor, identificando corretamente o autor das alterações no histórico do GitHub.

# Passo 5: Clonar o Repositório via SSH

No RStudio, crie um **Novo Projeto** usando controle de versão com Git. Ao informar a URL do repositório, use o formato SSH — diferente do HTTPS:

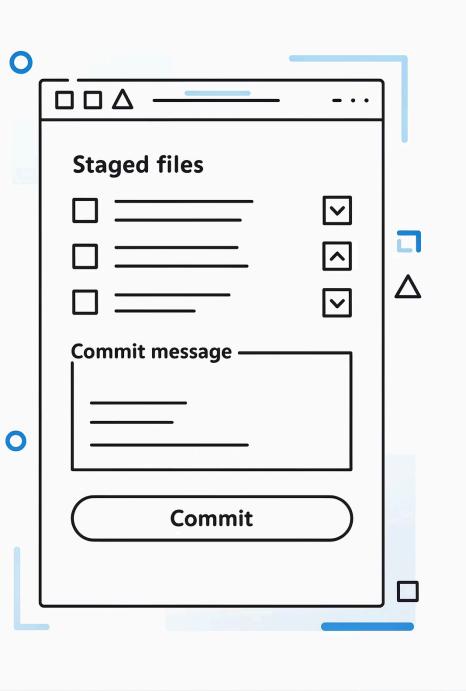
```
git@github.com:usuario/teste.git
```

## Formato SSH

Substitua `https://github.com/` por `git@github.com:` e adicione `.git` ao final do endereço do repositório.

## Escolha o Diretório

Selecione a pasta de destino no servidor onde o projeto será clonado. O padrão (default) funciona bem para a primeira configuração.



# Passo 6: Primeiro Commit e Push

01

## Criar arquivo de teste

Crie `HelloWorld.r` com `print("Hello World")`, execute e confirme o funcionamento.

02

## Marcar arquivos (Stage)

Na aba Git do RStudio, marque o arquivo `HelloWorld.r` e o arquivo de projeto para staging.

03

## Commit e Push

Clique em **Commit**, escreva a mensagem "*meu primeiro commit*", confirme e clique em **Push** para enviar ao GitHub.

# Fluxo Contínuo de Trabalho



**Abrir Projeto**  
File → Open Project → pasta do repositório clonado.

**Editar Código**  
Faça alterações nos arquivos R normalmente no RStudio.

**Push / Pull**  
Envie alterações ao GitHub ou traga atualizações com Pull.

**Commit**  
O Git detecta mudanças; escreva a mensagem e confirme o commit.

Uma vez configurado, o projeto fica pronto para operações contínuas de versionamento. O GitHub registra cada alteração com histórico completo, visível a todos os colaboradores.

# Exemplos Práticos: Clonagem e Commit

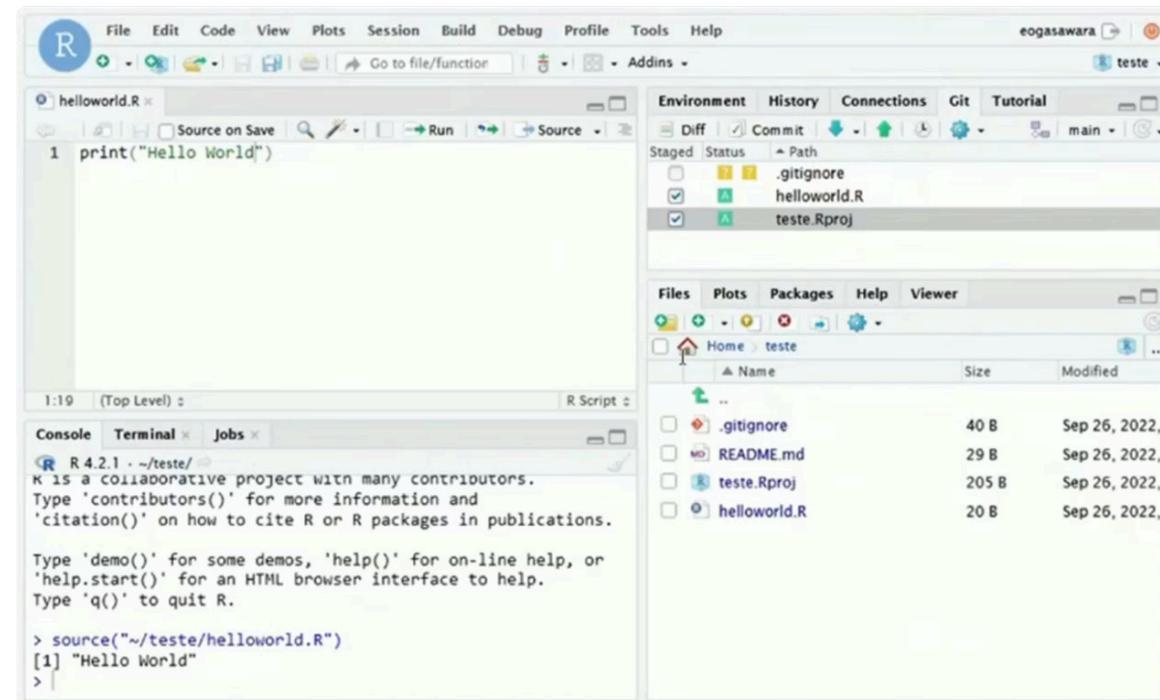
A seguir, veja na prática como clonar um repositório de teste via SSH no RStudio e como realizar um commit após editar um arquivo.

## Clonagem do Projeto via SSH



No RStudio, crie um novo projeto com controle de versão Git, informe a URL SSH do repositório e escolha o diretório de destino no servidor.

## Realizando um Commit



Na aba Git do RStudio, marque os arquivos alterados para staging, escreva a mensagem do commit e clique em Push para enviar ao GitHub.



# Resultado: Projeto Versionado no GitHub

## Histórico Completo

Cada commit registra quem alterou, o quê e quando — rastreabilidade total do código.

## Trabalho em Equipe

Colaboradores podem clonar, contribuir e sincronizar alterações com Pull e Push.

## Sem Perda de Código

Backup automático no GitHub evita perdas por falhas no servidor remoto.