

FACULDADE DE TECNOLOGIA SENAC GOIÁS
Gestão de Tecnologia da Informação



GERÊNCIA DE CONFIGURAÇÃO

Componentes: **Victor Hugo P. Costa,**

Goiânia,

2016

GERÊNCIA DE CONFIGURAÇÃO

Trabalho apresentado como requisito parcial para obtenção de aprovação na disciplina Auditoria e Qualidade de Software, no Curso de Gestão de Tecnologia da Informação, na Faculdade de Tecnologia Senac Goiás.

Professor Elias Ferreira.

Goiânia,

2016

SUMÁRIO

O que é a gerência de configuração?	4
O que são itens de configuração?	4
Resultados esperados de acordo com a MPS.BR	5
Processo de Gerência de Configuração	6
GitHub	8
Bibliográfica.....	17

Gerência de Configuração

O que é a gerência de configuração?

Os sistemas de software estão em constante evolução. A manutenção do software, isto é, modificações em artefatos existentes, chega a consumir 75% do custo total do seu ciclo de vida. Aproximadamente, 20% de todo o esforço de manutenção é usado para consertar erros de implementação e os outros 80% são utilizados na adaptação do software em função de modificações em requisitos funcionais, regras de negócios e na reengenharia da aplicação. A Gerência de Configuração de Software surgiu da necessidade de controlar estas modificações, por meio de métodos e ferramentas, com o intuito de maximizar a produtividade e minimizar os erros cometidos durante a evolução.

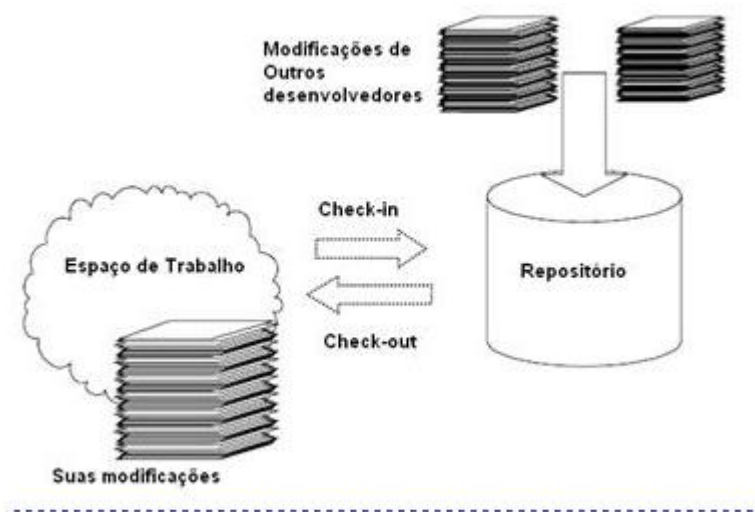
É uma disciplina que controla e notifica as inúmeras correções, extensões e adaptações aplicadas durante o ciclo de vida do software de forma a assegurar um processo de desenvolvimento e evolução sistemático e rastreável, sendo indispensável quando equipes manipulam, muitas vezes em conjunto, artefatos comuns.

Apesar de existir um forte apelo para o uso da Gerência de Configuração de Software durante a etapa de manutenção, a sua aplicação não se restringe somente a essa etapa do ciclo de vida do software. O uso dos sistemas de Gerência de Configuração é fundamental para prover controle sobre os artefatos produzidos e modificados por diferentes recursos desde o planejamento e levantamento de requisitos até a construção e entrega do produto. O motivo da sua importância está geralmente associado aos problemas identificados quando a Gerência de Configuração não é utilizada no desenvolvimento de software.

O que são itens de configuração?

A melhor forma de explicar é citando exemplo, então vamos lá.

O sistema de controle de versões (**Subversion**, por exemplo) permite que os artefatos sejam obtidos, por meio de uma operação conhecida como check-out, modificados dentro do espaço de trabalho do desenvolvedor e, depois, retornados ao repositório, por meio de uma operação conhecida como check-in, como exemplifica a Figura 1. O repositório é o local de armazenamento dos artefatos que estão sob controle da Gerência de Configuração de Software. **Estes artefatos recebem o nome de itens de configuração.** A cada operação de check-in realizada, a versão do item de configuração é incrementada de uma unidade. Quando o item é adicionado pela primeira vez no repositório, este item passa a ter a versão igual a 1. Para cada item de configuração armazenado, são anexadas informações como: datas da criação ou alteração, comentários e versões.



Neste cenário, não há perdas ou sobreposições porque políticas de trabalho foram estabelecidas, restringindo ou controlando as modificações no repositório. As ferramentas de controle de versões normalmente suportam a definição de diferentes políticas de trabalho. Dentre essas políticas, podemos citar a política pessimista, que enfatiza o uso de check-out reservado, fazendo bloqueio e inibindo o paralelismo do desenvolvimento sobre o mesmo artefato.

Resultados esperados de acordo com a MPS.BR

São 7 (sete) no total.

1. GCO 1 – Um Sistema de Gerência de Configuração é estabelecido e mantido.
2. GCO 2 – Os itens de configuração são identificados com base em critérios estabelecidos.
3. GCO 3 – Os itens de configuração sujeitos a um controle formal são colocados sob baselines.
4. GCO 4 – A situação dos itens de configuração e das baselines é registrada ao longo do tempo e disponibilizada.
5. GCO 5 – Modificações em itens de configuração são controladas.
6. GCO 6 – O armazenamento, o manuseio e a liberação de itens de configuração e baselines são controlados.
7. GCO 7 – Auditorias de configuração são realizadas objetivamente para assegurar que as baselines e os itens de configuração estejam íntegros, completos e consistentes.

Processo de Gerência de Configuração

Do ponto de vista gerencial, o processo de Gerência de Configuração de Software é dividido em cinco funções: **identificação da configuração, controle da configuração, acompanhamento da situação da configuração, auditoria da configuração e gerenciamento de entrega.**

A função de **identificação da configuração** tem por objetivo: (1) a seleção de quais artefatos serão itens de configuração; (2) a definição de uma nomenclatura, que possibilite a identificação inequívoca dos itens de configuração, baselines e releases; e (3) a descrição dos itens, tanto física quanto funcionalmente.

A seleção de itens de configuração é feita no início da fase de planejamento e leva em conta: (1) se o artefato é crítico para o projeto; (2) a dependência entre artefatos; (3) o impacto que uma modificação do item tem no produto; (4) se o artefato pode ser modificado por dois ou mais grupos; (5) se é frequentemente alterado devido a sua complexidade e (6) se é gerado manualmente, automaticamente ou ambos.

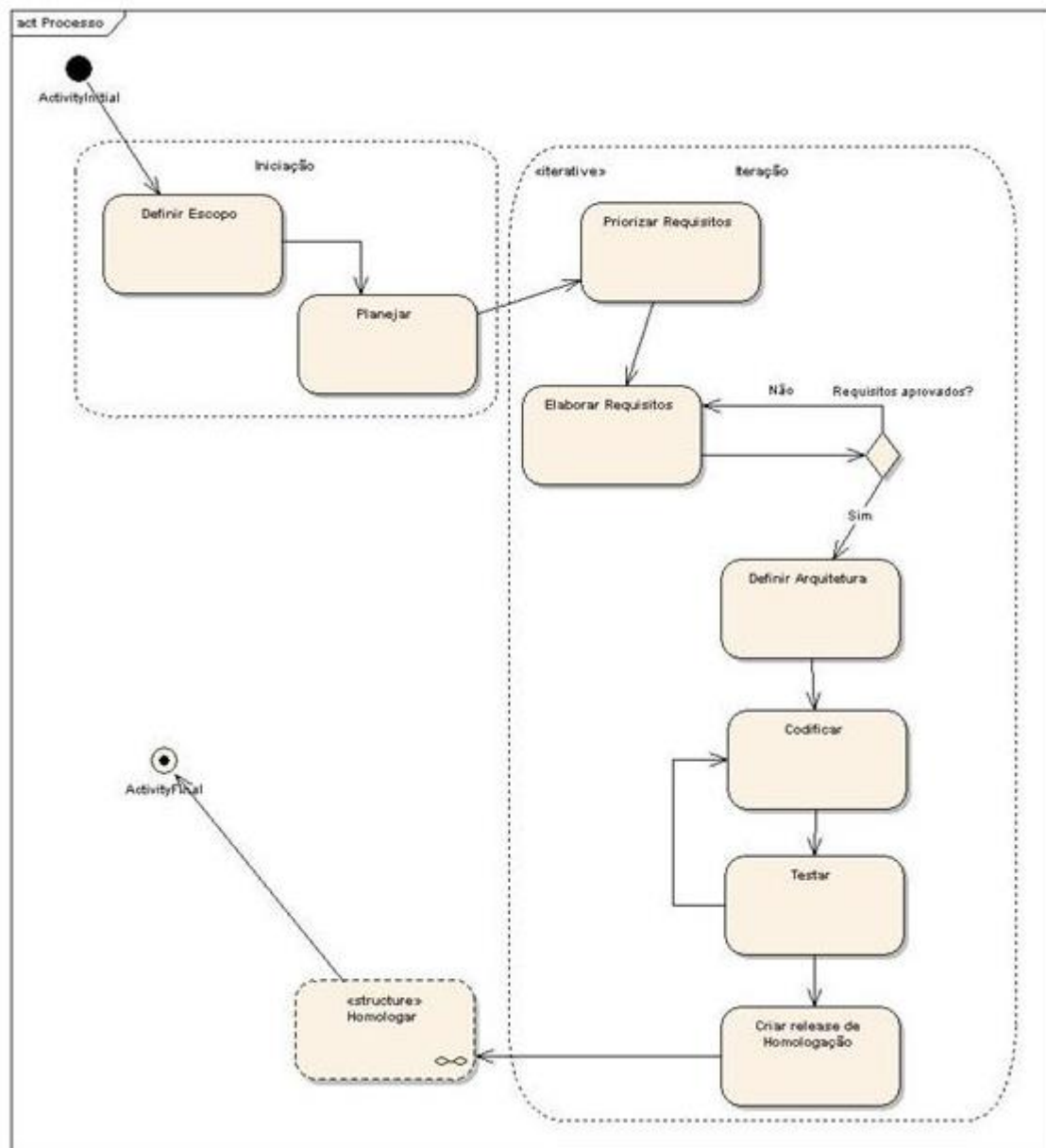
A função de **controle da configuração** é designada para controlar e acompanhar a evolução dos itens de configuração selecionados na função de identificação. Ferramentas como JIRA, Bugzilla, dentre outras, apoiam, em conjunto com as ferramentas de controle de versões, as atividades desta função.

A função de **acompanhamento da situação da configuração** visa: (1) armazenar as informações geradas pelas demais funções; e (2) permitir que essas informações possam ser acessadas em função de necessidades específicas, por exemplo, para a melhoria do processo, para a estimativa de custos futuros e para a geração de relatórios gerenciais. Estas informações podem ser obtidas, no decorrer do projeto, a partir dos sistemas de controle de versões e modificações.

A função de **auditoria da configuração** ocorre geralmente quando uma release deve ser criada. Suas atividades compreendem: (1) auditoria funcional, que abrange a revisão dos planos, dados, metodologia e resultados dos testes, assegurando que a release cumpre corretamente o que foi especificado; e (2) auditoria física, com o objetivo de certificar que a release é completa em relação ao que foi acertado em cláusulas contratuais. A auditoria pode ser feita com base nos relatórios obtidos na função anterior.

Já a função de **gerenciamento de liberação e entrega** descreve o processo formal de: (1) construção e liberação de uma release do produto; e (2) entrega, com informações de como implantar o software no ambiente final de execução. Ferramentas, como Ant, permitem que roteiros de construção sejam escritos e executados no apoio a esta fase.

Para exemplificar, vamos considerar um ciclo de vida iterativo e incremental que propõe inicialmente a identificação do escopo do projeto e a aprovação deste escopo pelo cliente. Posteriormente, nesta abordagem, o software é construído em ciclos sucessivos denominados iterações. A cada iteração, os requisitos são priorizados, detalhados, aprovados e o software são modelados, construído e testado, como ilustra a **Figura 2**. Ao final, um release do produto é entregue para homologação e aprovação do cliente.



Podemos citar como exemplos de ferramentas de mercado: CVS, Subversion, IBM Rational ClearCase e Microsoft Visual Source Safe.

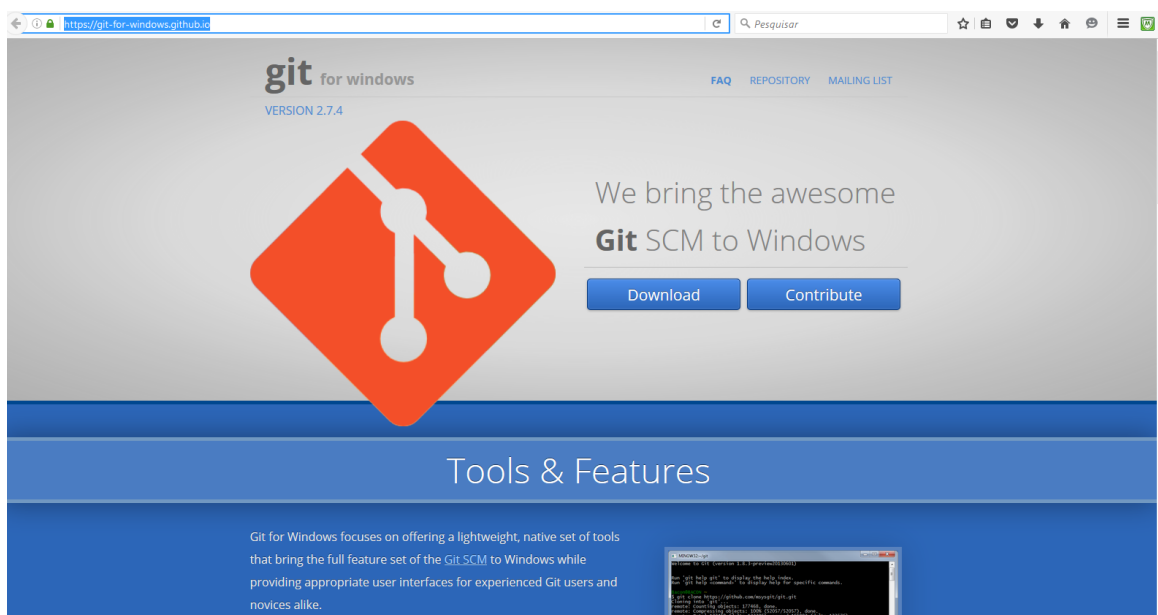
GitHub

GitHub é um Serviço de Web Hosting Compartilhado para projetos que usam o controle de versionamento Git. É escrito em Ruby on Rails pelos desenvolvedores da Logical Awesome (Chris Wanstrath, PJ Hyett e Tom Preston - Wernder). O GitHub possui planos comerciais e gratuitos para projetos de código aberto.

Este site possui funcionalidades de uma rede social como feeds, followers, wiki e um gráfico que mostra como os desenvolvedores trabalham as versões de seus repositórios.

Passos para utilizar.

Primeiramente, baixa-lo e instalar utilizando o default.



Abrir o GitBash e gerar a chave pública SSH apontando para a conta que você criará no GitHub.


```
MINGW64:/c/Users/vichugo/.ssh

vichugo@VICTOR MINGW64 ~
$ .ssh
bash: .ssh: command not found

vichugo@VICTOR MINGW64 ~
$ ~/.ssh
bash: /c/Users/vichugo/.ssh: Is a directory

vichugo@VICTOR MINGW64 ~
$ ssh-keygen -t rsa -C "victorhpc.ti@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/vichugo/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/vichugo/.ssh/id_rsa.
Your public key has been saved in /c/Users/vichugo/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:4GpJxAvPt38bYBfgEaeUDqrnYki9vCrEN9CDa1NcYt0 victorhpc.ti@gmail.com
The key's randomart image is:
+---[RSA 2048]-----+
|  .  +0 |
| +..E . |
| *..0* . |
| o.X +. + |
| o. + B + S |
| o*.. + + |
| =0+ * o |
| +..+ |
| oo... |
+---[SHA256]-----+

vichugo@VICTOR MINGW64 ~
$ AC

vichugo@VICTOR MINGW64 ~
```

```
MINGW64:/c/Users/vichugo/.ssh

Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/vichugo/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/vichugo/.ssh/id_rsa.
Your public key has been saved in /c/Users/vichugo/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:4GpJxAvPt38bYBfgEaeUDqrnYki9vCrEN9CDa1NcYt0 victorhpc.ti@gmail.com
The key's randomart image is:
+---[RSA 2048]-----+
|  .  +0 |
| +..E . |
| *..0* . |
| o.X +. + |
| o. + B + S |
| o*.. + + |
| =0+ * o |
| +..+ |
| oo... |
+---[SHA256]-----+

vichugo@VICTOR MINGW64 ~
$ AC

vichugo@VICTOR MINGW64 ~
$ AC

vichugo@VICTOR MINGW64 ~
$ cd ~/.ssh

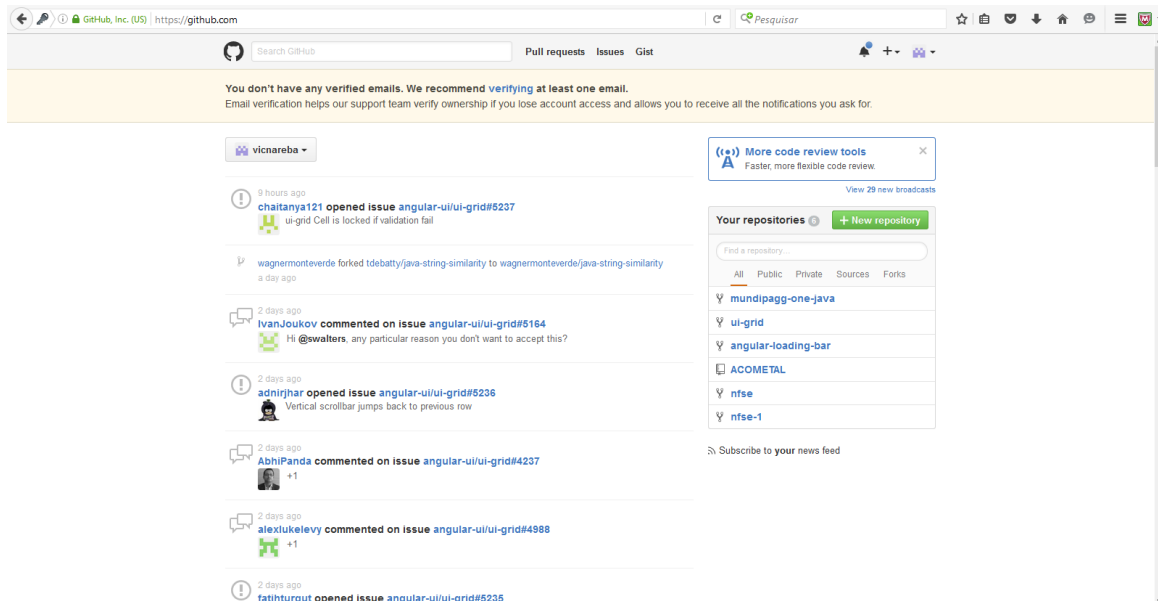
vichugo@VICTOR MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub

vichugo@VICTOR MINGW64 ~/.ssh
$ |
```

```
C:\Users\vichugo\ssh\id_rsa.pub - Notepad++
Arquivo  Editor  Localizar  Visualizar  Formatar  Linguagem  Configurações  Macro  Executar  Plugins  Janela  ?

1  ssh-pub AAAAB3NzaC1yc2EAAAADAQABAAQCAwEh/2YicRE1h8c3W7K+2DABY3DOH9cA4cW1TaK9nAcM4Fn1S14/w+1XKDgeft0t08q5bF27iedf6y3p2HFK21hRv57m19hKdEh9w1uWA9vW1d9dcUc5qUyqzVrDeXjecB3dKtQ1/IQ12PF4Ob8
2
```

Após feito isso, acessar a sua conta GitHub e ir em Settings/SSH Keys. Em seguida, criar um novo SSH Key e adicionar a Key gerada.



SSH keysNew SSH key

There are no SSH keys with access to your account.

Title

Key


```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDM84h/2YltzRE1h8hc3wVh+2UAYBjUOH9tA4rW1Taik9nACzW4PniS14
/w+ixXDgtfUeT08grSbFZ7ledfm6y3pqZHPk2ihHwS7mi9hkDEh9wluWA9wld9dcUc5gUyqzVrDeXjecB3dkVQl
/TQl2PF4ObPpj
/Q3BJA5d8eFsQkNG868int8X3Fv9F1c13JhH4taqehRPo2UlrH6mFpaboYOO+bAGneHgntiCTDU8Jhbc9wtmOboBG1MB
94RdTZ8W5mXW4gDLdSvOrgQqvFYQmtj31v6HEL1IVf63Z7qnNKw/XOEirOii0Elt04jm26Ac4mglDmsxR3OTzeIO/9rJ1
victorhpc.ti@gmail.com
```


Add SSH key

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

SSH keys
New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 <div> Desktop Home Fingerprint: 12:84:93:29:89:ff:69:ec:bd:d9:16:50:bc:f2:24:8e <div>SSH</div> <div>Added on 20 Mar 2016 — Never used</div> </div> <div>Delete</div>
--

 Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

Verificar a conexão com o GitHub.

```

MINGW64:/c/Users/vichugo/.ssh
+---[RSA 2048]-----+
  .  +0
  +. =. E .
  * . =0* .
  o.X +.+
  o.+ B + S
  o* = * +
  =0+ * o
  +..+
  |oo..
+---[SHA256]-----+

vichugo@VICTOR MINGW64 ~
$ AC

vichugo@VICTOR MINGW64 ~
$ AC

vichugo@VICTOR MINGW64 ~
$ cd ~/.ssh

vichugo@VICTOR MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub

vichugo@VICTOR MINGW64 ~/.ssh
$ ssh -T git@github.com
The authenticity of host 'github.com (192.30.252.128)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWG17E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.252.128' (RSA) to the list of known hosts.
Enter passphrase for key '/c/Users/vichugo/.ssh/id_rsa':
Hi vicnareba! You've successfully authenticated, but GitHub does not provide shell access.

vichugo@VICTOR MINGW64 ~/.ssh
$

```

Criar uma pasta que será o seu repositório. Em seguida, utilizar o comando git init para inicializar o repositório.

```
MINGW64:/c/Users/vichugo/.ssh/Trabalhos
$ AC
vichugo@VICTOR MINGW64 ~
$ AC
vichugo@VICTOR MINGW64 ~
$ cd ~/.ssh
vichugo@VICTOR MINGW64 ~/.ssh
$ ls
id_rsa id_rsa.pub
vichugo@VICTOR MINGW64 ~/.ssh
$ ssh -T git@github.com
The authenticity of host 'github.com (192.30.252.128)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.252.128' (RSA) to the list of known hosts.
Enter passphrase for key '/c/Users/vichugo/.ssh/id_rsa':
Hi vicnareba! You've successfully authenticated, but GitHub does not provide shell access.
vichugo@VICTOR MINGW64 ~/.ssh
$ mkdir Trabalhos
vichugo@VICTOR MINGW64 ~/.ssh
$ cd Trabalhos/
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos
$ ls
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos
$ git init
Initialized empty Git repository in C:/Users/vichugo/.ssh/Trabalhos/.git/
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$
```

Adicionar no repositório o que você deseja disponibilizar. Utilizar o comando git status para verificar o status do repositório, se à algo para commitar.

```
MINGW64:/c/Users/vichugo/.ssh/Trabalhos
vichugo@VICTOR MINGW64 ~/.ssh
$ mkdir Trabalhos
vichugo@VICTOR MINGW64 ~/.ssh
$ cd Trabalhos/
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos
$ ls
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos
$ git init
Initialized empty Git repository in C:/Users/vichugo/.ssh/Trabalhos/.git/
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    GerenciaDeConfiguracao.docx

nothing added to commit but untracked files present (use "git add" to track)
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ |
```

Feito isso, utilizar o comando git add . para adicionar o que será disponibilizado no commit.

```
MINGW64:/c/Users/vichugo/.ssh/Trabalhos
vichugo@VICTOR MINGW64 ~/.ssh
$ cd Trabalhos/

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos
$ ls

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos
$ git init
Initialized empty Git repository in C:/Users/vichugo/.ssh/Trabalhos/.git/

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        GerenciaDeConfiguracao.docx

nothing added to commit but untracked files present (use "git add" to track)

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git add .

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$
```

Utilizar em seguida o comando `git commit -m 'COMENTARIO'` para commitar o que deseja.

```
MINGW64:/c/Users/vichugo/.ssh/Trabalhos
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git add .

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   GerenciaDeConfiguracao.docx

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git commit -m 'initial commit. Trabalho Gerencia de Configuracao'
[master (root-commit) b47ba23] initial commit. Trabalho Gerencia de Configuracao
Committer: Victor Hugo de Paula Costa <Victor Hugo de Paula Costa>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 GerenciaDeConfiguracao.docx

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$
```

Para identificar o usuário do commit, usar os comandos `git config --global user.name 'usuário'` e `git config --global user.email 'e-mail do usuário'`.

```
MINGW64:/c/Users/vichugo/.ssh/Trabalhos

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   GerenciaDeConfiguracao.docx

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git commit -m 'initial commit. Trabalho Gerencia de Configuracao'
[master (root-commit) b47ba23] initial commit. Trabalho Gerencia de Configuracao
Committer: Victor Hugo de Paula Costa <Victor Hugo de Paula Costa>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 GerenciaDeConfiguracao.docx

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git config --global user.name 'vicnareba'

vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git config --global user.email 'victorhpc.ti@gmail.com'


vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$
```

Agora vamos criar um repositório no GitHub.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 vicnareba ▾

Repository name

/

Great repository names are short and memorable. Need inspiration? How about **probable-octo-fortnight**.

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

vicnareba / TrabalhosAuditoria

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or **HTTPS** SSH <https://github.com/vicnareba/TrabalhosAuditoria.git>

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# TrabalhosAuditoria" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/vicnareba/TrabalhosAuditoria.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/vicnareba/TrabalhosAuditoria.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

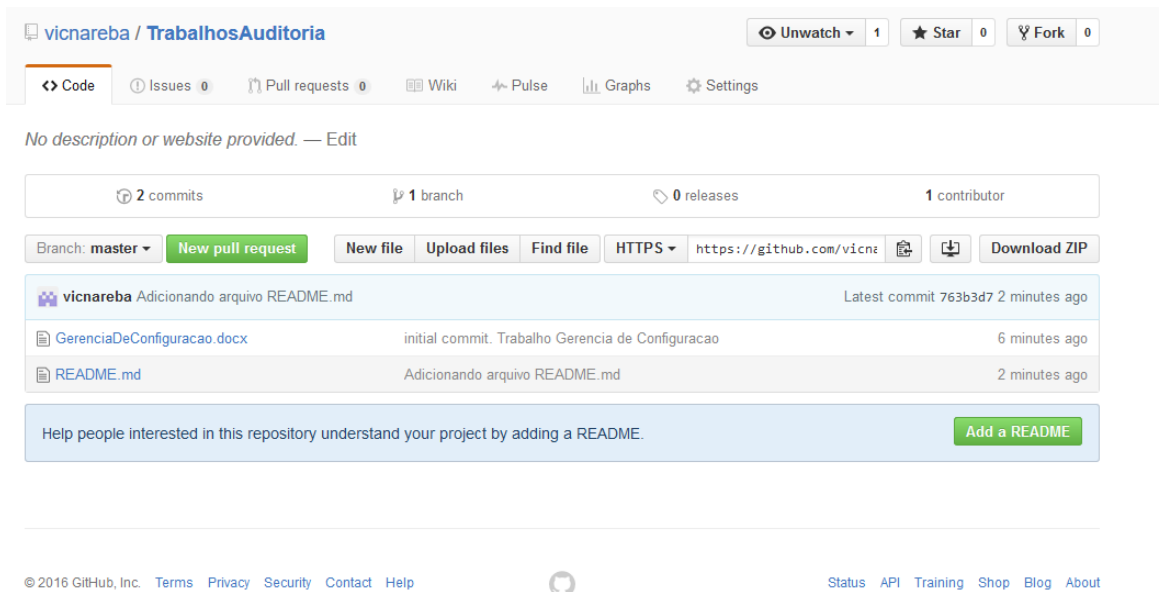
Feito isso, vamos sincronizar o repositório do GitHub com o seu repositório criado na máquina.

```
MINGW64:/c/Users/vichugo/.ssh/Trabalhos
$ touch README.md
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

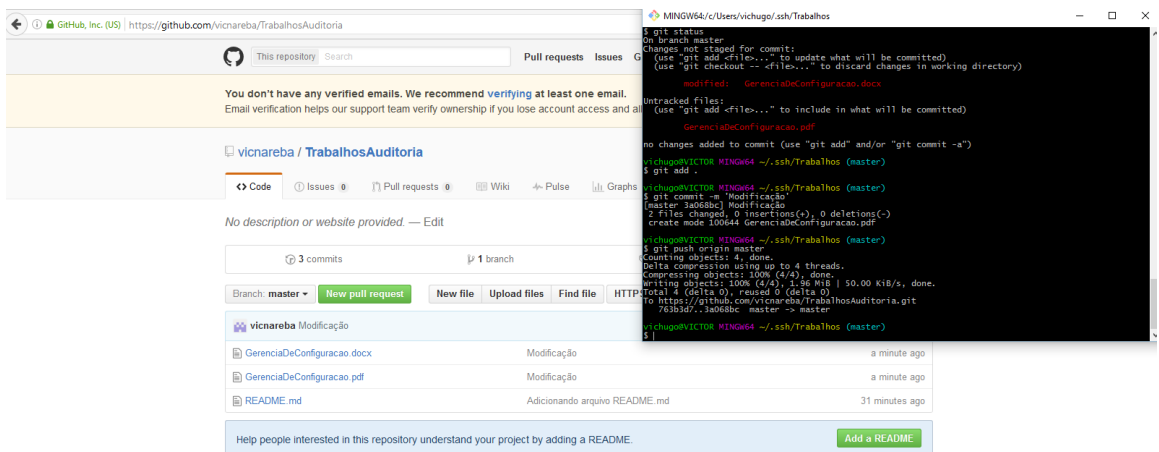
        README.md

nothing added to commit but untracked files present (use "git add" to track)
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git add .
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git commit -m 'Adicionando arquivo README.md'
[master 763b3d7] Adicionando arquivo README.md
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git remote add origin https://github.com/vicnareba/TrabalhosAuditoria.git
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ git push origin master
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 198.13 KiB | 0 bytes/s, done.
Total 6 (delta 1), reused 0 (delta 0)
To https://github.com/vicnareba/TrabalhosAuditoria.git
 * [new branch]      master -> master
vichugo@VICTOR MINGW64 ~/.ssh/Trabalhos (master)
$ |
```

E pronto, o mesmo estará tanto na sua maquina, quanto na nuvem.



Qualquer modificação do arquivo, o mesmo identificará uma modificação e exigirá um commit. Utilizar os comandos ensinados para commitar e sincronizar com o repositório na nuvem.



Bibliográfica

<http://www.devmedia.com.br/gerencia-de-configuracao-de-software/9145>

[https://www.youtube.com/watch?v= Ci1q53DgvY](https://www.youtube.com/watch?v=Ci1q53DgvY)

<https://git-for-windows.github.io/>

<http://pt.slideshare.net/vaniltonpinheiro/mps-br-garantia-da-qualidade>