

## Tutorial GitLab

(baseado em <https://youtu.be/Jt4Z1vwtXT0>)



# Análise e Projeto de Sistemas

## Parte 1 - Introdução



# Introdução

- O que é o GitLab?
- Como criar um conta no GitLab?
- Como criar um projeto no GitLab?



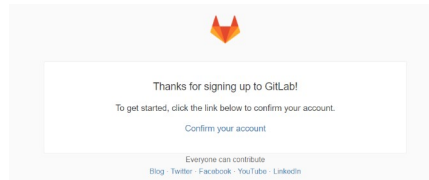
# Introdução

- O que é o GitLab?
- Git: é um sistema de controle de versão que checa, localmente, mudanças no seu projeto e/ou pasta. Além disso, efetua as operações push e pull sobre as mudanças no projeto em repositórios remotos, como o GitHub, BitBucket e o GitLab.
- GitLab: assim como o GitHub e o BitBucket, é um serviço que tem como objetivo armazenar o seu projeto em um repositório remoto, além de oferecer funcionalidades para o controle do ciclo de vida do projeto.
  - Gerenciamento;
  - Compartilhamento;
  - Páginas Wiki
  - Controle de Bugs
  - Integração Contínua e Entrega Contínua (CI & CD, Continuous Integration e Continuous Delivery,)



# Introdução

- Como criar um conta no GitLab
  - Vá para [www.gitlab.com](https://www.gitlab.com)



## GitLab.com

GitLab.com offers free unlimited (private) repositories and unlimited collaborators.

- [Explore projects on GitLab.com](#) (no login needed)
- [More information about GitLab.com](#)
- [GitLab Community Forum](#)
- [GitLab Homepage](#)

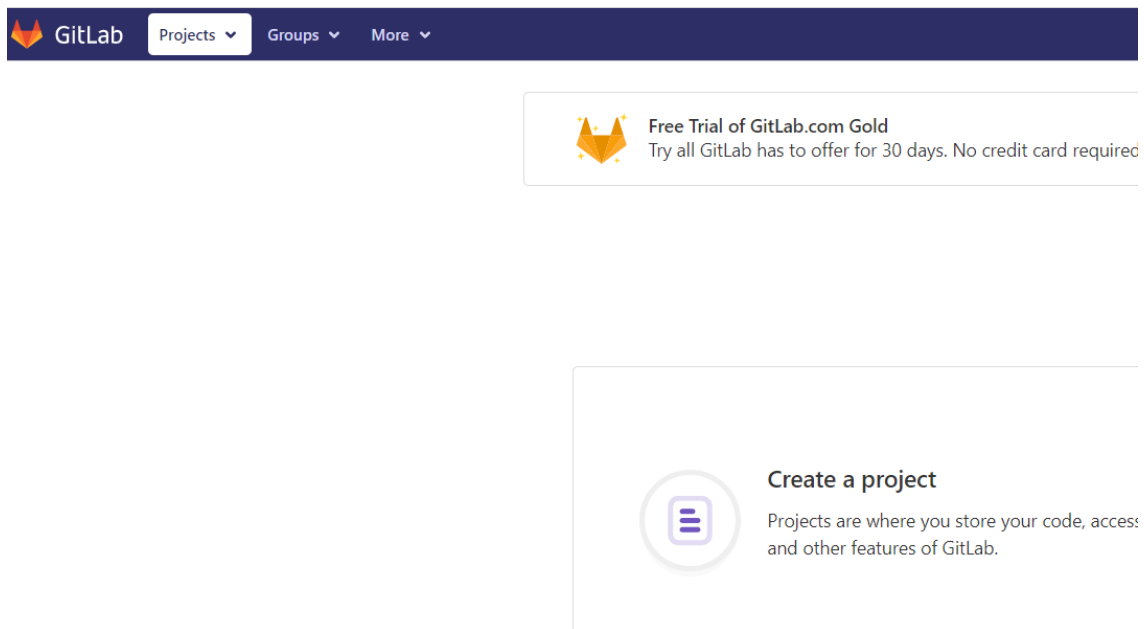
By signing up for and by signing in to this service you accept our:

- [Privacy policy](#)
- [GitLab.com Terms](#).

A screenshot of the GitLab registration form. It includes input fields for "First name" and "Last name", a "Username" field, an "Email" field, and a "Password" field with a note "Minimum length is 8 characters." Below these is a checkbox for "Não sou um robô" (I am not a robot) with a hCAPTCHA logo and links for "Privacidade" and "Termos". A green "Register" button is at the bottom. Below the button, it says "By clicking Register, I agree that I have read and accepted the GitLab Terms of Use and Privacy Policy" with a link to "or" for another option. At the bottom, under "Create an account using:", there are buttons for "Google", "GitHub", "Twitter", "Bitbucket", and "Salesforce".A screenshot of the GitLab welcome page for a user named "Jefferson". It says "Welcome to GitLab Jefferson!". Below, it asks to personalize the experience by selecting a "Role" (a dropdown menu is shown with "Software Developer" selected) and answering "Who will be using GitLab?" (radio buttons for "My company or team" and "Just me", with "Just me" selected). There is also an optional checkbox for "Email updates (optional)". A green "Get started!" button is at the bottom.

# Introdução

- Como criar um projeto no GitLab



# Introdução

- Projeto em branco (Blank Project)



## Create blank project

Create a blank project to house your files, plan your work, and collaborate on code, among other things.

New project › Create blank project

### Project name

Meu Primeiro Projeto

### Project URL

https://gitlab.com/jeffersoncarvalho/

### Project slug

meu-primeiro-projeto

Want to house several dependent projects under the same namespace? [Create a group](#).

### Project description (optional)

Apenas o meu primeiro projeto, nada demais.

### Visibility Level

☒ Private

Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

☐ Public

The project can be accessed without any authentication.

☐ Initialize repository with a README

Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

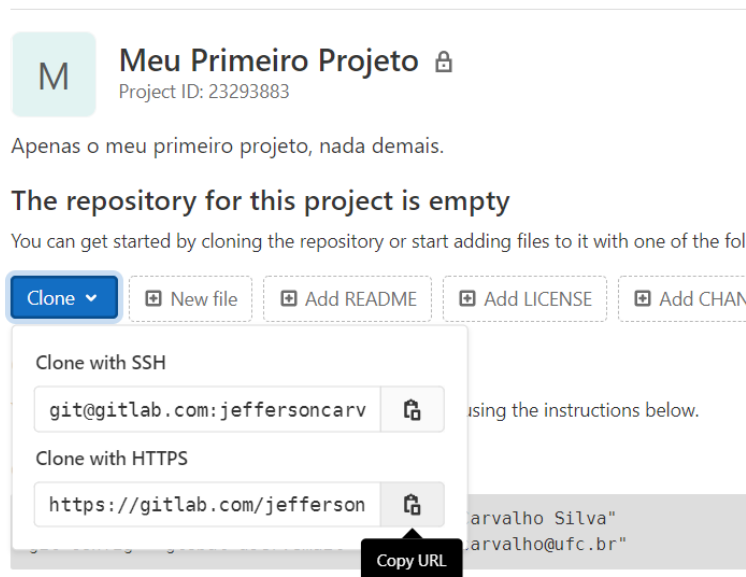
Create project

Cancel



# Introdução

- Copiar o link para clone/push/pull via HTTPS





# Análise e Projeto de Sistemas

## Parte 2 - Comandos Git



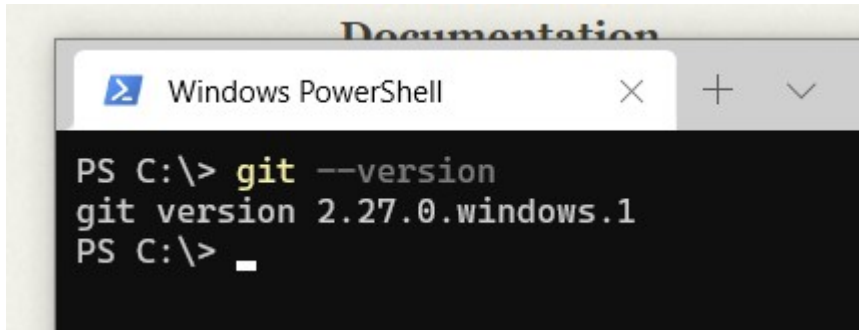
# Comandos Git

- Baixar e instalar o Git.
- Adicionar um projeto/pasta no Git.
- Commit e Push no GitLab.

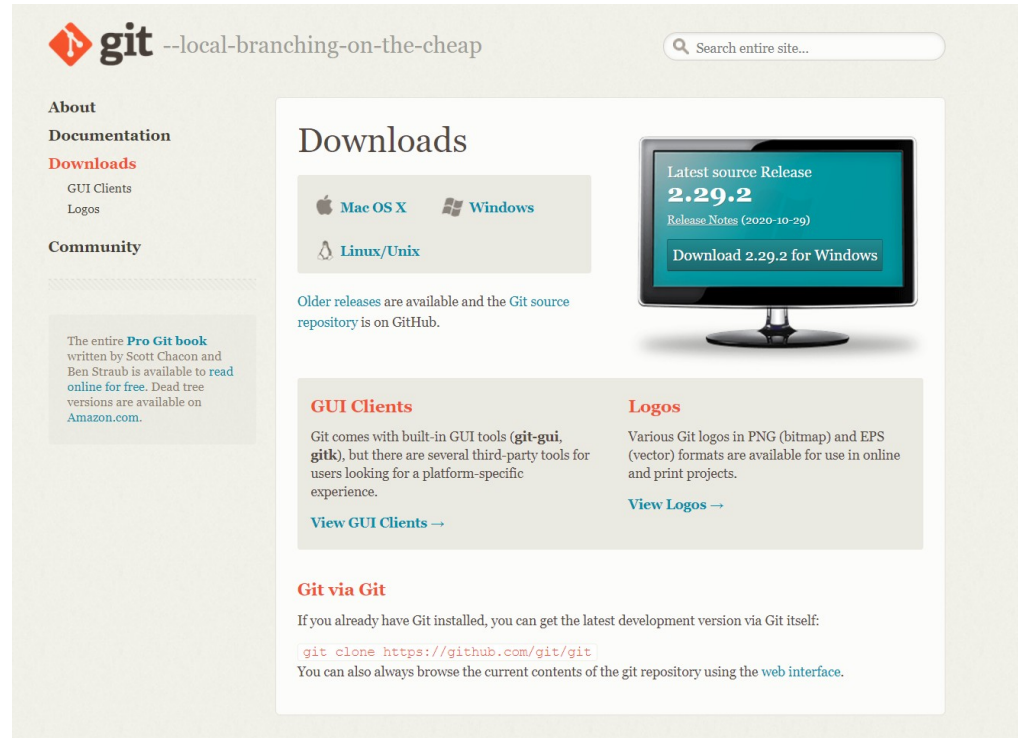


# Comandos Git

- Baixar e instalar o Git
  - <https://git-scm.com/downloads>
  - Instale (next→next...)



```
PS C:\> git --version
git version 2.27.0.windows.1
PS C:\> _
```



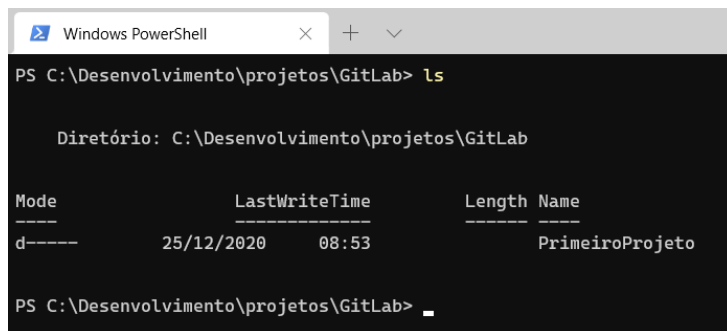
# Comandos Git

- Executando os comandos Git
  - `git config --global user.name <nome do usuário>`
  - `git config --global user.name`
  - `git config --global user.email <e-mail do usuário>`
  - `git config --global user.email`
  - `git config --global --list`



# Comandos Git

- Crie um pasta que será o seu projeto (use as funcionalidades do seu sistema operacional)
- Vá para o terminal e CD até o local da pasta do projeto



```
Windows PowerShell
PS C:\Desenvolvimento\projetos\GitLab> ls

Diretório: C:\Desenvolvimento\projetos\GitLab

Mode                LastWriteTime         Length Name
----                -
d-----         25/12/2020    08:53      PrimeiroProjeto

PS C:\Desenvolvimento\projetos\GitLab>
```

- Entre na pasta do projeto
  - cd PrimeiroProjeto



# Comandos Git

- Dentro da pasta do projeto:

- git init

```
PS C:\Desenvolvimento\projetos\GitLab> cd .\PrimeiroProjeto\  
PS C:\Desenvolvimento\projetos\GitLab\PrimeiroProjeto> git init  
Initialized empty Git repository in C:/Desenvolvimento/projetos/GitLab/PrimeiroProjeto/.git/
```

- Foi criada uma pasta oculta chamada “.git” (ver “mostrar itens ocultos”)

- git status

```
PS C:\Desenvolvimento\projetos\GitLab\PrimeiroProjeto> git status  
On branch master  
  
No commits yet  
  
nothing to commit (create/copy files and use "git add" to track)  
PS C:\Desenvolvimento\projetos\GitLab\PrimeiroProjeto> _
```

- Crie um arquivo (README.txt)
  - Execute “git status” novamente



# Comandos Git

- O próximo passo é adicionar todos os arquivos não rastreados no repositório LOCAL
  - `git add <nome do arquivo>`  
Ou, caso queria adicionar múltiplos arquivos:
    - `git add .`
- **Execute novamente:**
  - `git status`
- **Validar as alterações LOCALMENTE (famoso commit)**
  - `git commit -m “meu primeiro commit”`



# Comandos Git

- Refletir as alterações locais no repositório REMOTO
  - `git push -u "https://gitlab.com/jeffersoncarvalho/meu-primeiro-projeto.git" master`



- Veja no site do GitLab, se o arquivo foi carregado





# Análise e Projeto de Sistemas

## Parte 3 – Fork no Projeto



# Fork no Projeto

- **O que é um Fork?**
  - Podemos dizer que um “fork” é uma cópia de um projeto.
  - O “fork” de um projeto permite fazer mudanças sem afetar o projeto principal.
- **Passos:**
  - Faça o login no GitLab e vá até o seu projeto;
  - Clique no botão “Fork”



# Fork no Projeto

- Caso apareça a mensagem:
  - “No available groups to fork the project.”
  - Clique em “Groups→ New Group”

Grupo de APS 2020 2

 Group 'Grupo de APS 2020 2' was successfully created.

G

**Grupo de APS 2020 2** 

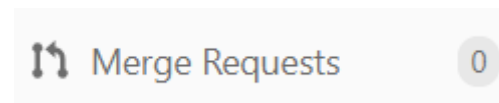
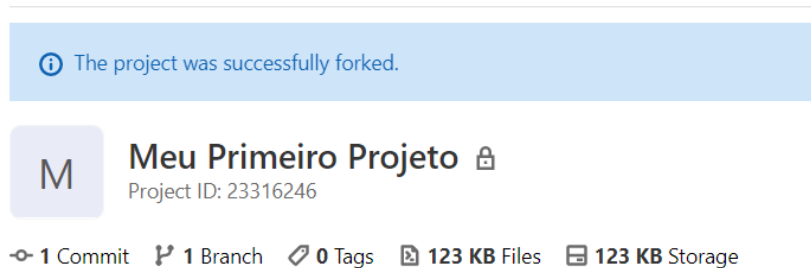
Group ID: 10491718

Subgroups and projects   Shared projects   Archived projects



# Fork no Projeto

- Volte para o seu projeto:
  - Clique novamente em “Fork”;
  - Selecione o grupo que você acabou de criar;



# Análise e Projeto de Sistemas

## Parte 4 – Chave SSH



# Chave SSH

- O que é o SSH
  - SSH vem de “Secured Shell”.
  - Uma chave SSH serve para identificar o seu computador no servidor sem precisar que seja feito o login/senha toda vez que uma operação for efetuada.
  - Na prática, uma chave SSH é um arquivo de texto, com um código que identifica a sua máquina.
  - Esse arquivo deve ser gerado por um programa.



# Chave SSH

- Gerando a chave
  - No Linux e Mac:
    - Basta executar o comando `ssh-keygen` no terminal.
  - No Windows:
    - Use o mesmo comando, `ssh-keygen.exe`, no aplicativo `git-bash` (instalado junto com o Git).



# Chave SSH

- Usando a chave
  - Faça o login no GitLab;
  - User → Settings → SSH keys;
  - Abra o arquivo .pub gerado na sua máquina;
  - Copie e cole o conteúdo em →

## Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id\_ed25519.pub' or '~/.ssh/id\_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Do not paste your private SSH key, as that can compromise your identity.

Typically starts with "ssh-ed25519 ..." or "ssh-rsa ..."





Parte 5 – Issues e Milestones  
([https://www.tutorialspoint.com/gitlab/  
gitlab\\_milestones.htm](https://www.tutorialspoint.com/gitlab/gitlab_milestones.htm))



# Issues e Milestones

- Uma “Issue” pode-se tratar de “questões” relacionadas ao desenvolvimento de um projeto. Elas podem tratar desde evoluções simples no código ou até mesmo problemas sérios que devem ser resolvidos imediatamente.
- Bug vs Issue: um bug comunica que um potencial problema existe no código que o time está trabalhando (ex.: um defeito no código que não deveria existir). Uma issue é um evento ou uma causa que atrasa ou bloqueia o processo de desenvolvimento (ex.: uma biblioteca de terceiros não será lançada a tempo).
- No GitLab, esses conceitos podem se confundir mas normalmente as Issues são usadas para reportar problemas em geral.



# Issues e Milestones

- Para criar uma Issue:
  - Issues → New Issue

New Issue

---

Title

Add [description templates](#) to help your contributors communicate effectively!

Type

Description

Write

Preview

**B** *I* ” ” <> @ ☰ ☷ ☰ ☷ ☰ ☷ ↵

Write a comment or drag your files here...

Markdown and quick actions are supported [Attach a file](#)

☐ This issue is confidential and should only be visible to team members with at least Reporter access.

---

Assignee  [Assign to me](#) Due date

Milestone

Labels

# Issues e Milestones

- Milestones são marcos no projeto. Geralmente podem ser relacionados ao lançamento de pequenas releases. Um conjunto de tarefas, ao finalizar, identificam um Milestone.
- No GitLab, Milestones são usados para organizar Issues e juntar requisições em um determinado grupo que podem ser resolvidas em uma data pré-estabelecida.



# Issues e Milestones

- Criando Milestones:
  - Issues → Milestones → New Milestone

New Milestone

Title	<input type="text" value="Interface Gráfica V0"/>	Start Date	<input type="text" value="2020-12-30"/>	<a href="#">Clear start date</a>
Description	<div><div>Write Preview</div><div><b>B</b> <i>I</i> “ ” &lt; &gt; @    </div><div>Desenvolver a primeira versão "oca" da interface gráfica.</div><div>Markdown is supported <a href="#">Attach a file</a></div></div>	Due Date	<input type="text" value="2021-01-31"/>	<a href="#">Clear due date</a>

Create milestone

Cancel



# Issues e Milestones

- Crie Issues para o Milestone:
  - Issues → New Issue

Open **Milestone** Dec 30, 2020–Jan 31, 2021 Edit Close milestone Delete

---

## Interface Gráfica V0

Desenvolver a primeira versão "oca" da interface gráfica.

📘 Assign some issues to this milestone.

**Issues** 0 Merge Requests 0 Participants 0 Labels 0

Unstarted Issues (open and unassigned) 0

Ongoing Issues (open and assigned) 0

Completed Issues (closed) 0



# Issues e Milestones

Open 2 Closed 1 All 3



Recent searches ▾

Milestone = "%Interface Gráfica V0" ✕

Created date ▾ ⌵

Conexão com o Banco de Dados



#4 · opened 5 days ago by Jefferson Carvalho Silva ⌚ Interface Gráfica V0

  0

updated 5 days ago

Interface do ADM

#3 · opened 5 days ago by Jefferson Carvalho Silva ⌚ Interface Gráfica V0

  0

updated 5 days ago

[Email a new issue to this project](#)

Open 1 Closed 0 All 1

Filter by milestone name

Due soon ▾

New milestone

Interface Gráfica V0

Dec 30, 2020–Jan 31, 2021

Jefferson Carvalho Silva / Meu Primeiro Projeto

3 Issues · 0 Merge Requests

33% complete

Close Milestone



# Análise e Projeto de Sistemas

## Parte 6 – Instalando e Criando Runners





# Instalando e Criando Runners

- **O que é o GitLab Runner?**
  - É um serviço do GitLab que fazer parte do CI & CD. O seu objetivo é executar tarefas e retornar os resultados para o GitLab. Para usar o GitLab Runner você deve:
    - Instalar o GitLab Runner;
    - Registrar o GitLab Runner;
      - Processo de ligar o runner com a instância do GitLab



# Instalando e Criando Runners

- Instalando...
  - <https://docs.gitlab.com/runner/install/>
  - Crie uma pasta. Ex.: `c:\GitLab-Runner`
  - Faça o download da versão adequada
  - Renomeie o arquivo para **gitlab-runner.exe**
  - Execute a linha de comando como administrador:
    - **`.\gitlab-runner.exe install`**

```
c:\GitLab-Runner>gitlab-runner.exe install
Runtime platform                                arch=386 os=windows pid=6396 revision=943fc252 version=13.7.0

c:\GitLab-Runner>gitlab-runner --version
Version:      13.7.0
Git revision: 943fc252
Git branch:   13-7-stable
GO version:   go1.13.8
Built:        2020-12-21T13:47:18+0000
OS/Arch:      windows/386

c:\GitLab-Runner>
```

# Instalando e Criando Runners

- Registrando...

- <https://docs.gitlab.com/runner/register/index.htm>
- Execute o comando **.\gitlab-runner.exe register**
- Copie a URL padrão e pressione enter
- Entre com o **token** do SEU projeto:
  - Settings → CI / CD → Runners (Expand)
  - Copie e cole o token no terminal
  - Entre com uma descrição
  - Entre com as tags:
    - ssh, ci
  - Escolha a opção do executar:
    - shell

```
Runtime platform arch=386 os=windows pi
Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com/
Enter the registration token:
62Dqvz5TZzx58Msc5bQ
Enter a description for the runner:
[jefferson-yoga]: my-runner-test
Enter tags for the runner (comma-separated):
ssh, ci
Registering runner... succeeded runner=62Dqvz5T
Enter an executor: virtualbox, custom, docker, docker-windows, docker-ssh,
-ssh+machine, kubernetes:
shell
Runner registered successfully. Feel free to start it, but if it's running
loaded!
```

## Specific Runners

### Set up a specific Runner automatically



You can easily install a Runner on a Kubernetes cluster.

[Learn more about Kubernetes](#)

1. Click the button below to begin the install process by navigating to the Kubernetes page
2. Select an existing Kubernetes cluster or create a new one
3. From the Kubernetes cluster details view, install Runner from the applications list

Install Runner on Kubernetes

### Set up a specific Runner manually

1. [Install GitLab Runner](#)
2. Specify the following URL during the Runner setup:  
<https://gitlab.com/> 
3. Use the following registration token during setup:  
62Dqvz5TZzx58Msc5bQ 

Reset runners registration token

4. Start the Runner!

# Instalando e Criando Runners

- Iniciando...
  - `\gitlab-runner.exe start`
  - Veja no projeto se seu Runner iniciou:
  - Agora você tem um Runner capaz de executar jobs CI/CD

Runners activated for this project

● DjFoQSb2...  

Pause

Remove Runner

my-runner-test

#3803749

ci ssh



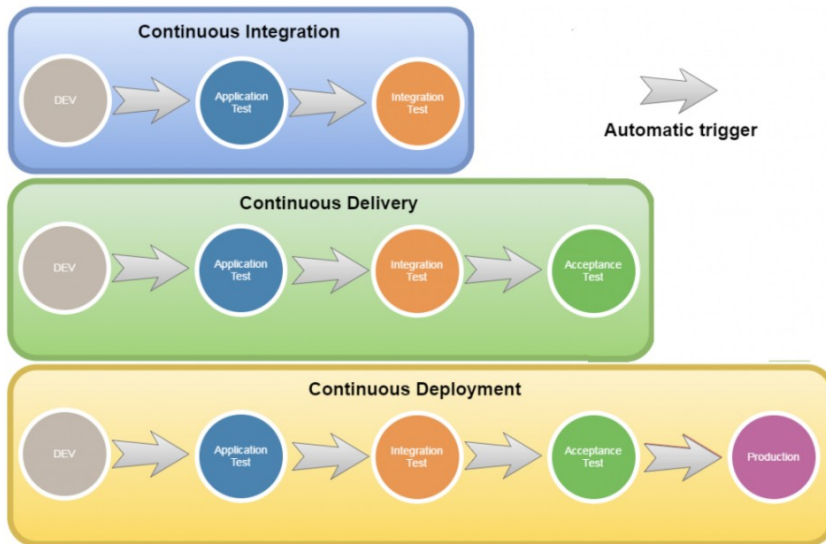
# Análise e Projeto de Sistemas

## Parte 7 – CI & CD



# CI & CD

- O que é CI & CD?
  - Em inglês: Continuous Integration & Delivery (Continuous Deployment)



Continuous integration vs. continuous delivery vs. continuous deployment

Source: [StackOverflow](#)

# CI & CD

- **Continuous Integration (Integração Contínua)**
  - Compreende os seguintes eventos ligados em uma cadeia:
    - Desenvolvimento (DEV);
    - Testes de Aplicação;
    - Teste de Integração.
  - Quando estes passos são disparados de forma automática, nós dizemos que temos um sistema de integração contínua.



- **Continuous Delivery (Entrega Contínua)**
  - Compreende os seguintes eventos ligados em uma cadeia:
    - Desenvolvimento (DEV);
    - Testes de Aplicação;
    - Testes de Integração;
    - Testes de Aceitação.
  - Praticamente os mesmo passos da CI, só que agora temos os Testes de Aceitação. Dizemos que a “entrega é contínua” pois após o último passo, o nosso produto está “pronto” para a entrega.





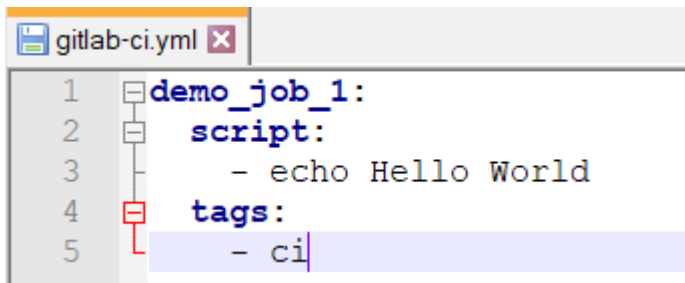
- **Continuous Deployment (Implantação Contínua)**
  - Compreende os seguintes eventos ligados em uma cadeia:
    - Desenvolvimento (DEV);
    - Testes de Aplicação;
    - Testes de Integração;
    - Testes de Aceitação;
    - Produção.
  - Praticamente os mesmo passos da CD, só que agora temos a Produção. Na Implantação Contínua, o sistema está pronto para entrar na fase de produção, ou seja, ser usado por clientes reais.



# CI & CD

- CI no GitLab

- É um serviço open-source implementado desde a versão 8.0 do GitLab que tem como objetivo automatizar as etapas da Integração Contínua. Para ter acesso e controle a esse serviço, você deve ser o Project Owner (Manager).
- Para iniciar a Integração Contínua, você deve:
  - 1 - Criar o arquivo **.gitlab-ci.yml** na raiz do seu projeto.



```
1 demo_job_1:
2   script:
3     - echo Hello World
4   tags:
5     - ci
```



# CI & CD

- **CI no GitLab**

- Continuando...

- 2 - Fazer o commit & push do arquivo no nosso repositório.

- 1) git add .

- 2) git commit -m "add gitlab-ci.yml"

- 3) git push -u "<https://gitlab.com/jeffersoncarvalho/meu-primeiro-projeto.git>" master



# CI & CD

- CI no GitLab

- Continuando...

- 3 – Criar o Runner do seu projeto.

- 1) Seu Projeto → Settings → CI / CD

- 2) Clique em “Runners” e depois “Expand”

## Runners

Collapse

Runners are processes that pick up and execute jobs for GitLab. Here you can register and see your Runners for this project. [More information](#)

You can set up as many Runners as you need to run your jobs.  
Runners can be placed on separate users, servers, and even on your local machine.

Each Runner can be in one of the following states:

- **active** - Runner is active and can process any new jobs
- **paused** - Runner is paused and will not receive any new jobs



# CI & CD

- CI no GitLab

- Continuando...

- 3 – Criar o Runner do seu projeto

3) Veja se o Runner criado em na Seção anterior está ativo:

Runners activated for this project

DjFoQsb2...

🔒

✎

Pause

Remove Runner

my-runner-test #3803749

ci

ssh

**Atenção! Essas tags devem ser as mesmas do arquivo yml!**

# CI & CD





- CI no GitLab

- Continuando...

- 4 – Faça qualquer modificação no seu projeto:

- **commit** → **push**

- Em CI/CD → Jobs

All 2 Pending 0 Running 1 Finished 1							CI lint
Status	Job	Pipeline	Stage	Name	Timing	Coverage	
	#950544004 ✓ master → ebc88b77 	#238838733 by 	test	demo_job_1	@ 00:00:10		

```
$ echo Hello World
Hello
World
Cleaning up file based variables
Job succeeded
```