ESTUDOS SOBRE DEVOPS

Sprint 1:

- Aprendendo os principais comandos do git e a utilizar o GitLab: É um controlador de versão de código fonte que permite rastrear e armazenar todas as mudanças em um código fonte, sem precisar sobrescrever as linhas de código.
- Quando se cria um repositório existe somente um único Branch (ramo) com o nome master, que representa o principal ramo. Após a criação do Branch master, deve-se criar um outro ramo com o nome de “develop”, esse ramo é o que vai receber todas as modificações ou evoluções para o desenvolvimento das funcionalidades. Logo após a criação do ramo “develop”, cria-se o ramo “staging”, que é responsável por realizar testes de software, garantindo confiabilidade para o sistema.
- **Conceitos básicos**:

  **Staging área**: Local onde é armazenado todas as alterações em que serão adicionadas no próximo commit.

  **Working directory:** Local no qual todos arquivos estão sendo trabalhados.

  **Commit**: Local em que são aplicadas as alterações no código.

  **Branch**: É uma ramificação do código que aponta para um commit.

  **Head**: É um ponteiro que aponta para algum commit.

  **Merge**: é uma funcionalidade que permite combinar alterações de um branch (ramificação) em outro, geralmente de um branch de desenvolvimento para o branch principal (como main ou master).

  Além disso, um arquivo no git pode estar em 4 estados diferentes:
  **Untracked:** Arquivos que não estavam no último commit;
  **unmodified:** Arquivos não modificados desde o último commit;
  **modified:** Arquivos modificados desde o último commit; e
  **staged** : Arquivos preparados para comitar.

- **Criando um repositório**: Para criar um repositório utiliza o comando “**git init**”, inicialmente vazio, e após isso, cria-se um arquivo README.md somente com textos, no intuito de todos acompanharem com clareza o que está acontecendo com os códigos e etc. Depois, vamos adicionar este arquivo à staging area para que faça parte do próximo commit. Fazemos isso com o comando “**git add**”.

  O próximo passo agora é realizar os commits, para que os arquivos na staging área passem a ser versionados. Para isso, utiliza-se o comando “**git commit**” e a partir desse momento o arquivo já pode ser versionado. Para uma melhor compreensão pode também usar o comando “**git log**” para ver o histórico de commits realizados.

Para evitar versionar aqrquivos binários usa o comando **“.gitignore**” onde vamos colocar todos os arquivos que queremos ignorar ao realizar um commit. O comando **echo** é utilizado apenas para adicionar a nova linha “main” ao arquivo “**.gitignore”**.

Mas e se nos arrependermos de algum commit e desejarmos desfazê-lo? Bom, para isso temos dois comandos, “**git reset”** e “**git revert”**.
No comando git reset existem três opções, soft, mixed e hard. A opção soft, move o HEAD para o commit indicado, mas mantém o staging e o working directory inalterados. A opção mixed, move o HEAD para o commit indicado, altera o staging e mantém o working directory. A opção hard faz com que o HEAD aponte para algum commit anterior, mas também altera a staging area e o working directory para o estado do commit indicado, ou seja, todas as alterações realizadas após o commit ao qual retornamos serão perdidas. Isso não é recomendável se as alterações já tiverem sido enviadas para o repositório remoto. Nesse caso devemos utilizar o git revert.

- **Uso de um repositório remoto**:
  Para baixar um repositório em nossa máquina, utiliza o comando “**git clone**”, e para subir as alterações para o repositório remoto no Gitlab, fazemos isso com o comando “**git push -u origin master”.** Para criar os ramos develop e staging utiliza o comando git **checkout -b develop master**, que cria o ramo develop a partir do ramo master. Para o staging usa o comando **git checkout -b staging develop**. Após criarmos os ramos, utilizamos os comandos **git push -u origin develop** e **git push -u origin staging** para subir as alterações no repositório remoto.

- **Cultura organizacional do TerraLAB**: Metodologias ágeis é a principal influência para o desenvolvimento de software dentro do TerraLAB, além do uso seguindo um padrão específico do Gitlab. São fornecidas métricas pelo Gitlab, uma delas está contida no campo *Value Stream Analytics*, que focam em medidas da “velocidade” de desenvolvimento de projetos. Essas métricas são muito úteis para identificar gargalos no processo de desenvolvimento, possibilitando à gerência revelar e triar causas raízes de lentidão no ciclo de desenvolvimento de softwares.

  **Issue**: essa métrica reporta o tempo gasto para criar determinada *issue* e tomar a decisão de “resolvê-la”, seja atribuindo uma *label* a ela ou adicionando-a a uma *milestone.*

  **Plan:** mede o tempo entre a ação tomada no estágio anterior e enviar o primeiro commit para a branch. O primeiro commit para a branch é o gatilho que diferencia a métrica Plan da Code, e ao menos um dos commits na branch precisa conter o número da issue relacionada

  **Code:** mede o tempo entre enviar o primeiro *commit* (estágio anterior) e criar uma *merge request* (MR) relacionada a esse *commit*.

**Test:** registra o tempo para executar todo o *pipeline* desse projeto. Está relacionado ao tempo que o CI/CD leve para rodar todos os *commits* enviados naquela MR definida no estágio anterior.

**Review**: retorna o tempo gasto para revisar a MR, desde o momento que ela é criada até o momento onde é concluída.

**Staging**: registra o tempo entre realizar o merging até o deploy para produção de uma MR. Para que isso ocorra, é necessário que exista um ambiente de produção

**Total:** essa métrica registra o tempo total do processo. Ou seja, desde a criação de uma issue até o deploy do código no ambiente de produção.

Qualquer coisa que não siga o fluxo do GitLab flow não será rastreado e o dashboard do Value Stream Analytics não apresentará nenhum dado para que os MRS não fecham uma issue, não apresentara dados para issues não rotuladas ou issues as quais não foram atribuídas a uma milestone e por fim não apresentara dados para o staging.

- **Resolvendo issues Boards no GitLAB**:

  Resolver as *issues* no GitLab é crucial para o desenvolvimento eficiente e organizado de projetos. As *issues* no GitLab são usadas para acompanhar tarefas, bugs, melhorias e outros aspectos do processo de desenvolvimento. Resolver essas *issues* de forma eficaz oferece uma série de benefícios, tanto para a equipe quanto para o projeto como um todo. Para isso, são divididos detalhes em cada um dos boards, entre eles:

  **Open**: É onde encontra-se as atividades abertas que ainda não estão no backlog ou não vão.

  **Backlog:** Lugar onde ficam todas as funcionalidades do backlog do produto.

  **Sprint Backlog:** Lugar em que ficam as *issues* que serão trabalhadas na sprint atual ou, no máximo, na próxima.

  **Doing**: São as issues que dos devs estão trabalhando

  **Waiting Acceptance**: São *issues* que foram implementadas pelo desenvolvedor e estão passando por testes feitos pelo time de QA.

  ***Done*****:** São *issues* que passaram pelos testes (previamente validados pelo *Product Owner*) e cuja implementação é considerada concluída.

  **Automating Tests e Deploy:** São boards especiais referentes as partes de CI – Continuous Integration e CD – Continuous Deployment do projeto. A parte CI é onde o projeto vai passar por testes automatizados e ao passarem nos testes, esse projeto é direcionado para o board “deploy” onde ocorre o CD.

  **Waiting Validation (externally):** É validado ou não finalmente por usuários reais do produto, caso seja validado a issue é movida para o próximo board.

**Done:** *Issues* que passaram por todo processo de produção e foram validadas pelos clientes dos produtos e já podem ser implantadas no ambiente de produção dos clientes do produto.

Nesse sentido, O Product Owner é o principal responsável por gerenciar o *Issue Board*. É ele quem define as “histórias de usuário” e é quem valida os “cenários de teste” especificados pelo *Tester. Ele é também* o principal responsável por coordenar as as atribuições de issues aos membros da equipe de um projeto, em consonância com seus perfis profissionais, ocupação e habilidades.