1. Fazer o download do git e instalar no computador
2. Criar uma pasta onde irá ficar o repositório local
3. Abrir o git bash ir até a pasta local e configurar o computador
   1. **Git config** global user.name “Paulo Takeyama”
   2. **Git config** global user.email “[p.tkeym@gmail.com](mailto:p.tkeym@gmail.com)”
4. Configurar a pasta criada para ser o repositório local do projeto
   1. **Git init**
5. **Git status** mostra o estado atual dos arquivos. Os arquivos em vermelho mostram que as modificações não estão monitoradas pelo git.
6. Ao usar o **git add <nome arq>** o status muda para verde, estando pronto para o commit. Para entender, o git contêm 3 áreas: 1ª é a de trabalho (working), onde fazemos as modificações, a 2ª ao colocar o comando git add, indexa no índice (index) e a 3ª é quando comita o repositório (head).
7. O comando **git commit –m “mensagem”** marca os arquivos como um ponto de controle, identificando o commit com uma id única. Seria a baseline.
8. Para saber o que foi alterado no arquivo temos os comandos
   1. **Git diff <nome arq>** mostra as modificações mas funciona somente antes do git add.
   2. **Git blame<nome arq>** mostra com mais detalhes as modificações linha por linha, quem, quando e o que foi modificado. Funciona mesmo depois do commit.
9. Para ver todos os commits que está no repositório usa o comando **git log**. A opção –p mostra a diferença de cada commit, --stat mostra quantas modificações foram feitas em cada commit e a opção –pretty=oneline mostra em uma linha o id e a mensagem do commit.
10. Para criar marcações mais intuitiva e fácil de trabalhar, pode criar tag em alguns commits. Exemplo: podemos marcar quando o site entra no ar como v1.0 com o comando **git tag v1.0** e a cada modificação significativa podemos marcar v1.1, v2.0, etc. se colocarmos o comando git tag, o git listará todas as tags existentes.
11. Github é um site onde podemos criar repositórios e colocar os projetos para compartilhar com outros colaboradores. Github é uma rede social para desenvolvedores com uma comunidade com mais de 14 milhões de pessoas desenvolvendo, descobrindo, usando e colaborando com mais de 35 milhões de projetos. Tem a conta pessoal que pode ser gratuita, mas os seus arquivos serão compartilhados com a comunidade, ou pagar 7 dólares por mês para ter repositórios privados. A conta business é para empresas e tem vários preços dependendo de quantos repositórios privados querem ter. Muitos profissionais usam o github no currículo, pois assim o empregador pode verificar projetos que o candidato participou.
12. Tem que criar a conta no git não obrigatoriamente para ser o repositório do projeto. Pode-se usar o Dropbox por exemplo como uma alternativa ao github, porem o github tem ferramentas importantes para controle de versão que um disco virtual normalmente não tem. Depois de configurar o git para poder mandar através de linha de comando o repositório local para o remoto, deve-se criar o repositório remoto e copiar a url que fica localizado na caixa de texto ao lado. Vou para o git bash e coloco o comando: **git remote <um alias> <url>**, pressiona a tecla enter e coloca login e senha do github. Está pronto para enviar o repositório local para o remoto. Para atualizar ou colocar o projeto no github digite o comando: **git push <alias> master**. O master é o branch padrão que é criado quando configuramos a pasta e quando criamos o repositório remoto.
13. Agora que o repositório está configurado e disponível, outros colaboradores podem contribuir com o projeto. É claro que os colaboradores devem ter o git instalado em suas máquinas também. O primeiro passo é clonar o repositório para seus computadores com o comando: **git clone <url>**.
14. Toda vez que fomos trabalhar com arquivos do projeto, devemos primeiro atualizar o nosso repositório, pois se alguém fez alguma mudança, teremos a versão mais atualizada e não dará conflito quando dermos o push para o repositório remoto. **Git pull <alias> master**.
15. Branch é um dos pontos fortes do git. Um exemplo de aplicação do branch é quando temos um site funcionando, para não mexermos na estrutura principal do projeto, criamos uma ramificação: **git branch <nome do branch>**. Assim teremos uma ramificação com cópia dos commits de um determinado ponto do branch master. Nesse branch podemos criar atualizações crítica que podem tirar o site do ar como teste até corrigir todos os erros. Após corrigir os erros podemos juntá-los com o comando **git merge teste** (comando deve ser dado a partir do branch master).

**Fim**