

# Curso Azure DevOps LinkedIn

## Sumário

1. *Da Ideia à Entrega com Azure DevOps*
  2. *Gerenciamento de Código*
  3. *Como Compilar seu Código*
  4. *Como Gerenciar seus Testes*
  5. *Azure DevOps Extensões e Integrações*
- 

## 1. Da Ideia à Entrega com Azure DevOps

Para executar, rastrear e acompanhar os trabalhos devemos usar Work Item. Pois ele é utilizado no Azure DevOps para rastrear diferentes tipos de trabalhos.

Existem vários tipos de Work Item que podem ser rastreados, como:

- Tarefas
- Épicos
- Features
- Histórias de Usuários

Onde podemos aplicar em nossos backlogs ou aplicar em sprints posteriormente, com objetivo de trabalhar com gerenciamento de tarefas do meu time de forma mais assertiva.

---

### O QUE SÃO WORK ITENS E SUA HIERARQUIA

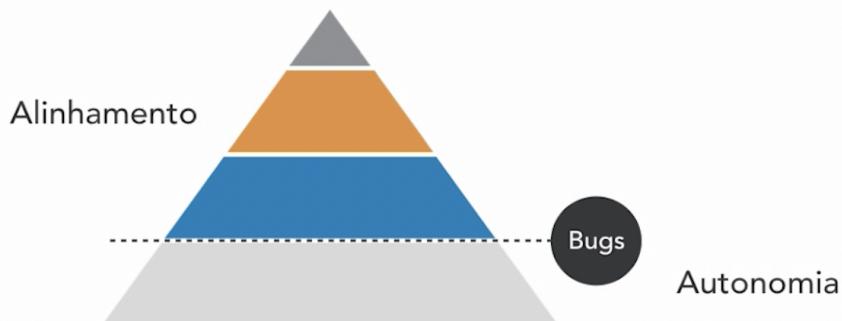
Os *Épicos* são o nível mais alto dos tipos de work itens e podem abranger vários tipos de liberações de entrega.

Abaixo temos as *Features*, que são utilizadas para determinar objetivos para um único sprint.

As *Histórias* dividem os recursos em peças de trabalho independente.

As *Tarefas* fornecem informações adicionais para histórias e são usadas para o planejamento da capacidade.

Os erros são tratados como Histórias ou Tarefas do usuário e representam problemas específicos no produto.



Épico = Sistema

Feature = Partes Específicas do Sistema

Histórias = Peças específicas de trabalho relacionada a essa feature que vamos trabalhar

Tarefas = Onde o time de desenvolvimento trabalhará

---

## BOARDS, BACKLOGS E SPRINTS

Desafios aos planos dos projetos:

- Surpresas
- Mudança de contexto
- Objetivos não compartilhados
- Silos de Conhecimento

### 🔍 Backlog

É a lista priorizada de recursos que contém descrições breves da funcionalidade desejada do produto.

### ↔️ Sprint

É o período de tempo durante o qual um trabalho específico deve ser concluído. Onde trabalhamos os itens priorizados do backlog.

No Azure DevOps, isso é chamado de Iteração.

### Primeira Sprint

Imagine que em uma primeira Sprint, nós separamos os itens do backlog e assim determinamos tudo que acreditamos que o time tem capacidade de entregar nesta sprint, especificamente.

Imagine que a lista abaixo é nosso backlog:

- Realidade** -----
- Doc: Update Create/Delete Team Project (Buck Hodges feedback)
  - Virtual Application Hosts: HostSyncService ("Host Fault-in") changes
  - Virtual Application Hosts: Host Routing reaction
  - ▶ ■ Virtual Application Hosts: Core Services Reaction / TFS Reaction
    - [SPIKE] Determine appropriate thresholds for users for public SUs
    - Push QDS data into Kusto.
    - ResourceUtilization UI notifications
    - Prep for collection the domain rollout to MSENG and the public scale units
    - Rename project named 'DefaultCollection'
    - Remove profile-only identities from the SPS configuration database
    - Make project creation automatically queue the project deletion job on failure
  - DataImport: Running custom hooks at the right time during import
    - Add L0 and L2 test to cover the ServiceHostHistoryJobExtension and prc\_QueryService...
    - Clean up artifacts after a data import fails or completes
    - Set up Test MDS Account and Kusto Cluster for Resource Utilization Development
    - Data Import: Generate the import package
    - ResourceUtilization Email notifications

A linha de Realidade divide onde realmente estamos entregando nossos itens. Portanto nesse caso, não consegui entregar todos os itens na primeira sprint, o que pode acontecer.

Já em uma segunda sprint, podemos chegar no cenário abaixo:

- Doc: Update Create/Delete Team Project (Buck Hodges feedback)
- Virtual Application Hosts: HostSyncService ("Host Fault-in") changes
- Virtual Application Hosts: Host Routing reaction
- ■ Virtual Application Hosts: Core Services Reaction / TFS Reaction
- [SPIKE] Determine appropriate thresholds for users for public SUs
- Push QDS data into Kusto.
- ResourceUtilization UI notifications
- Prep for collection the domain rollout to MSENG and the public scale units
- Rename project named 'DefaultCollection'
- Remove profile-only identities from the SPS configuration database
- Make project creation automatically queue the project deletion job on failure
- DataImport: Running custom hooks at the right time during import
- Add L0 and L2 test to cover the ServiceHostHistoryJobExtension and prc\_QueryService...
- Clean up artifacts after a data import fails or completes
- Set up Test MDS Account and Kusto Cluster for Resource Utilization Development
- Data Import: Generate the import package
- ResourceUtilization Email notifications

## Realidade

Com o backlog completo ao final da sprint.

### Cadênciā

É o ritmo ou timebox específico que os sprints seguem.

Para não acontecer como no caso acima, de acabar a sprint e ficar itens sem fazer no backlog, devemos sempre medir a cadênciā da nossa equipe (normalmente utilizando a média de tarefas entregues das duas ultimas sprints), e selecionar a quantidade de tarefas mais adequada.

O melhor é puxar menos tarefas e caso sobre tempo, puxar mais tarefas do backlog durante o sprint, do que acabar o sprint com tarefas por fazer, pois isso pode acabar frustando a equipe.

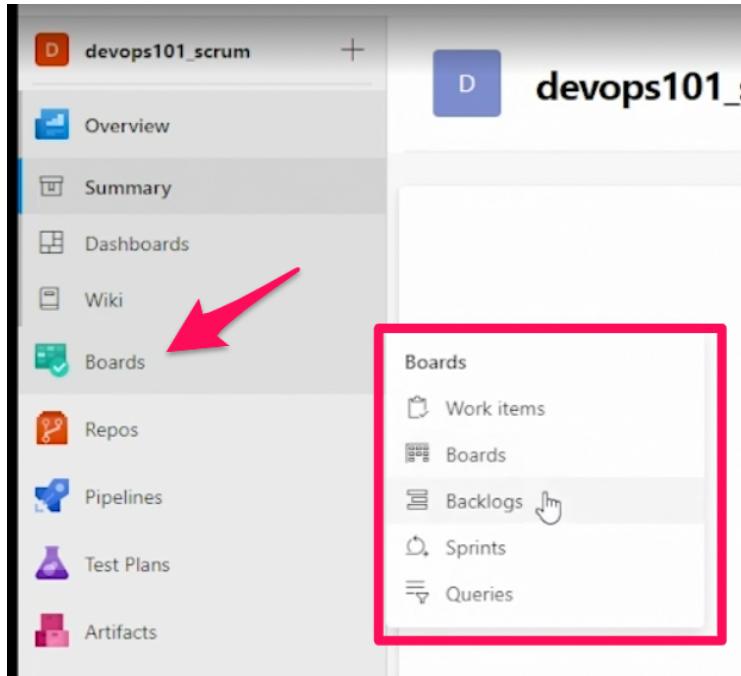
---

## TEAM PROJECTS, TEMPLATES E METODOLOGIAS ÁGEIS

Para utilizar um projeto com scrum, devemos ao criar o projeto, ir nas configurações avançadas da criação e ao invés de 'Basic' (como vem por padrão), devemos trocar para opção 'Scrum'.

Após realizar a criação do projeto, é necessário realizar algumas configurações para podermos trabalhar gerenciando backlogs, sprints, etc.

Para gerenciar Backlogs e Sprints vamos utilizar o 'Boards'

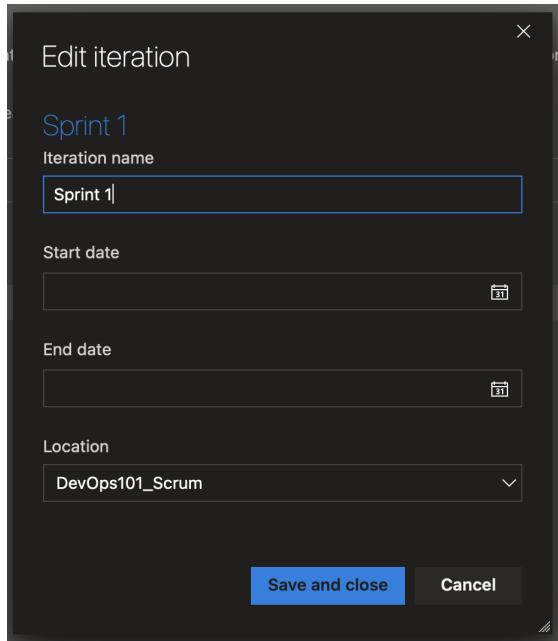


Para realizar a configuração, vamos em 'Project Settings' > 'Boards > Project Configuration'

Nesta tela vemos as sprints já criadas (como padrão) e podemos criar datas de início e fim para cada sprint.

The screenshot shows the 'Project Settings' page for the 'DevOps101\_Scrum' project. The left sidebar has sections like General, Boards (selected), Project configuration (highlighted in blue), Team configuration, GitHub connections, Pipelines, Agent pools, Parallel jobs, and Settings. The main area is titled 'Boards' and says 'This project is currently using the Scrum process. To customize your work item types, go to the process customization page.' It has tabs for Iterations and Areas. Below that, it says 'Create and manage the iterations for this project. These iterations will be used by teams for iteration planning (sprint planning).'. It has a 'Iterations' table with columns: New, New child, Iterations, Start Date, and End Date. The table shows iterations for 'DevOps101\_Scrum': Sprint 1, Sprint 2, Sprint 3, Sprint 4, Sprint 5, and Sprint 6. Red arrows point to the 'Start Date' and 'End Date' headers, and to the 'Sprint 3' row.

Para adicionar data, basta clicar no botão que aparece na coluna de 'Start Date', que aparece quando você passa o mouse sobre ele, que abrirá o modal para colocar os dados.



Feito isso, teremos uma data de início e fim para essa sprint.

New	New child	+	-
Iterations	Start Date	End Date	
▼ DevOps101_Scrum			
Sprint 1	... 14/03/2022	18/03/2022	
Sprint 2			
Sprint 3			
Sprint 4			

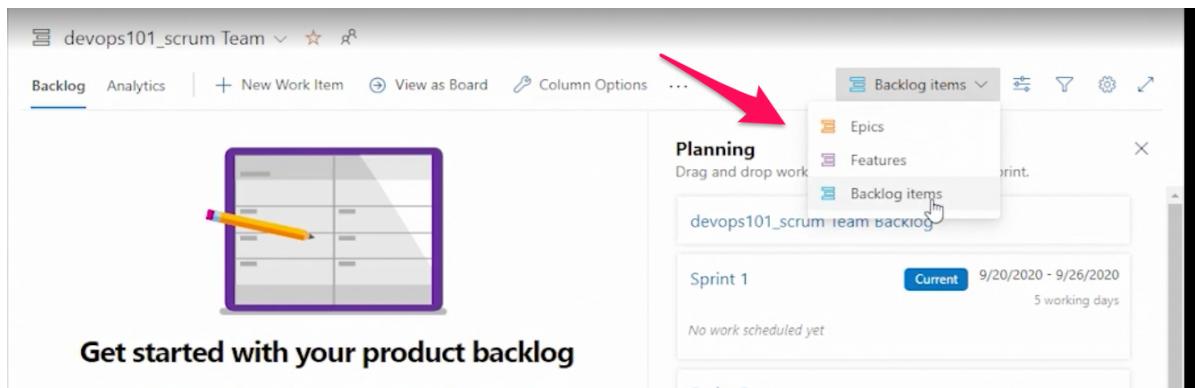
Agora podemos voltar em 'Boards', para criarmos nossas work items em nossas sprints.

## PLANEJAR SEU BACKLOG

Para gerenciar seu backlogs e sprints vamos acessar nosso projeto e clicar em 'Boards'.

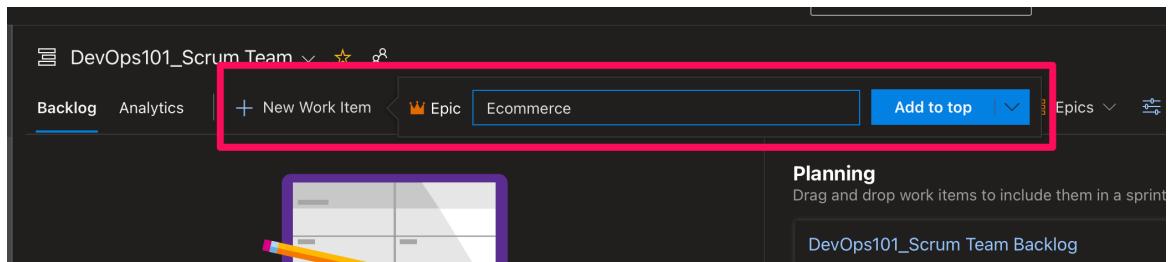
Para trabalhar com os work itens, vamos acessar dentro do boards o menu 'Backlogs', nele é onde vamos trabalhar e gerenciar todo nosso backlog. Podemos criar novas sprints também, do lado direito da tela, no menu '**Planning**'. Podemos acessar qual hierarquia queremos acessar (um ponto importante é que o épico não aparece na hierarquia, pois não vem marcado como padrão, e para

mostrá-lo devemos acessar o settings - engrenagem - e marca-lo para que ele apareça na seleção de hierarquia).



Selecionando no menu mostrado acima qual hierarquia deseja trabalhar, vamos agora criar as Work Item desejada.

Para isso vamos utilizar o botão "+ New Work Item", onde o sistema abrirá a caixa de texto para digitarmos o nome do épico, feature ou backlog e a opção de onde queremos inserí-lo (caso a lista já esteja grande).



Criamos o Épico como Ecommerce como nosso sistema principal.

Agora precisamos criar nossas Features. E para isso vamos clicar no "+" ao lado esquerdo do Épico.

The screenshot shows the Azure DevOps Backlog screen for the 'DevOps101\_Scrum Team'. At the top, there are navigation links for 'Backlog' (which is underlined), 'Analytics', 'New Work Item', 'View as Board', 'Column Options', and three dots. Below the header, there's a table with columns for 'Order', 'Work Item Type', 'Title', and 'State'. A single row is visible, labeled '1 Epic Ecommerce State: New'. A red arrow points to the '+' icon on the left side of the table.

O sistema vai abrir para preenchermos a tela de preenchimento de Features.  
No caso vamos cadastrar o Checkout do sistema.

The screenshot shows the 'Create Work Item' dialog for a 'Feature' type. The title bar says 'FEATURE 5' and '5 Checkout'. The main area has sections for 'Description' (with placeholder 'Click to add Description'), 'Acceptance Criteria' (with placeholder 'Click to add Acceptance Criteria'), and 'Discussion' (with placeholder 'Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.'). On the right, there are tabs for 'Status' (Start Date, Target Date), 'Deployment' (with a note about tracking releases), 'Details' (Priority: 2, Effort, Business Value, Time Criticality, Value area: Business), 'Development' (with a note about linking to Azure Repos), and 'Related Work' (Add link, Parent: '4 Ecommerce').

Cadastrada a feature, precisamos criar agora o backlog. Para isso vamos clicar no "+" ao lado esquerdo da Feature.

The screenshot shows the Microsoft Azure DevOps interface for the 'DevOps101\_Scrum Team'. The 'Backlog' tab is selected. At the top, there are navigation links for 'Backlog' (highlighted in blue), 'Analytics', 'New Work Item' (with a plus sign icon), 'View as Board', 'Column Options', and three dots for more options. Below the header is a table with columns: 'Order', 'Work Item Type', 'Title', 'State', and 'Reason'. One row is visible: '1', 'Epic', 'Ecommerce', '● New', and '● New'. Below this table is a large dark area where new items can be added. In the top right, there's a 'Planning' section with a 'Drag and drop' placeholder and a 'DevOps101' card. On the far right, there's a 'Sprint 1' section with a 'No work scheduled' message.

Nesse ponto podemos escolher criar:

- Product Backlog Item
- Bug

Vamos escolher Product Backlog Item (PBI).

O sistema abrirá a tela de cadastro de PBI.

The screenshot shows the 'New Product Backlog Item' creation form. At the top, it says 'NEW PRODUCT BACKLOG ITEM \*' and has a title field containing 'Implementação de nova versão x da página de checkout|'. Below the title are fields for 'Unassigned' (User), '0 comments', and 'Add tag'. On the right, there are buttons for 'Save & Close' and other options. The main form is divided into sections: 'Description' (with a note to 'Click to add Description'), 'Details' (Priority: 2, Effort, Business Value), 'Deployment' (with a note about tracking releases), 'Development' (with a note to 'Add link'), and 'Related Work' (with a note to 'Add link', Parent: '5 Checkout', and a status update). The 'Description' section is highlighted with a red arrow.

Porém para que o time de desenvolvimento consiga trabalhar com esse backlog, é preciso que sejam criadas as tarefas (tasks).

Para criar as tasks, vamos clicar no "+" ao lado esquerdo do PBI.

The screenshot shows the Azure DevOps Backlog board for the 'DevOps101\_Scrum Team'. A red arrow points to the '+ New Work Item' button at the top left of the backlog grid. The backlog table has columns for Order, Work Item Type, Title, State, and more. One row is selected, showing an Epic titled 'Ecommerce' and a Feature titled 'Checkout'. The backlog is part of a larger 'Planning' board view.

O sistema abrirá a tela de cadastro de cadastro de New Task.

The screenshot shows the 'New Task' creation dialog. A red arrow points to the title field where 'Testes' is typed. Another red arrow points to the 'Remaining Work' field, which is set to 8 hours. The dialog includes sections for Description, Details, Deployment, Development, and Related Work.

Nele eu posso cadastrar a task, o esforço para a atividade (em horas).

The screenshot shows the Azure DevOps Backlog board again. A red arrow points to the backlog table, specifically to a 'Task' row under the 'Product Back...' epic. This task is titled 'Testes' and is listed as 'To Do'. Another red arrow points to a second 'Task' row below it, also titled 'Testes' and listed as 'To Do'. The backlog table includes columns for Order, Work Item Type, Title, State, and more.

Pronto, agora que já cadastrei as tasks, temos o backlog pronto para realizar a Planning.

## SPRINTS EFICAZES

Para o planejamento da sprint (planning), vamos utilizar a mesma tela de Backlog. Para incluir uma task ou PBI em uma sprint, basta segurar com o clique do mouse e arrastá-la para a sprint desejada. Feito isso, a task ou PBI estará na sprint que colocou.

No menu 'Sprints', podemos ver o quadro da nossa sprint, podemos definir os membros da nossa equipe e também o capacity dela.

Caso um membro não trabalhe alguns dias/horas, podemos cadastrar esse tempo, pois assim ele não é considerado e não corremos o risco de contar com um colaborador que não estará na sprint por "x" motivo ou tempo.

Podemos também definir qual atividade aquele membro faz na sprint e qual capacidade (em horas) ele tem para atuar por dia.

Em Taskboard temos a visão de Kanban da nossa sprint, onde podemos arrastar os cards.

---

## MÉTRICAS E QUERY

Os desafios aos planos do projeto quando se fala em DevOps, o principal deles é o:

- Continuos Deployment

Essa é uma das principais práticas que se deve trabalhar quando falamos da cultura de DevOps.

Realizar entregas contínuas do seu software de features, de um ambiente específico de forma automáticos e validações ao longo desse ciclo. Porém para atingirmos esse nível de maturidade precisamos entender como está o ciclo do nosso projeto, e o planejamento de um projeto é o ponto de partida quando falamos de trabalhar com ciclo de vida de um software.

Para entendermos o nível de maturidade da nossa equipe, podemos trabalhar com algumas métricas dentro do Azure DevOps.

Para isso podemos utilizar as Query.

- Queries são um conjunto de cláusulas que ajudam a encontrar itens de trabalho que um usuário pode revisar, triar, atualizar ou listar em um relatório.
-

## BOAS PRÁTICAS COM AGILIDADE

Devemos sempre trabalhar com o framework junto com o DevOps, para melhorar a efetividade da ferramenta.

## 2. Gerenciamento de Código

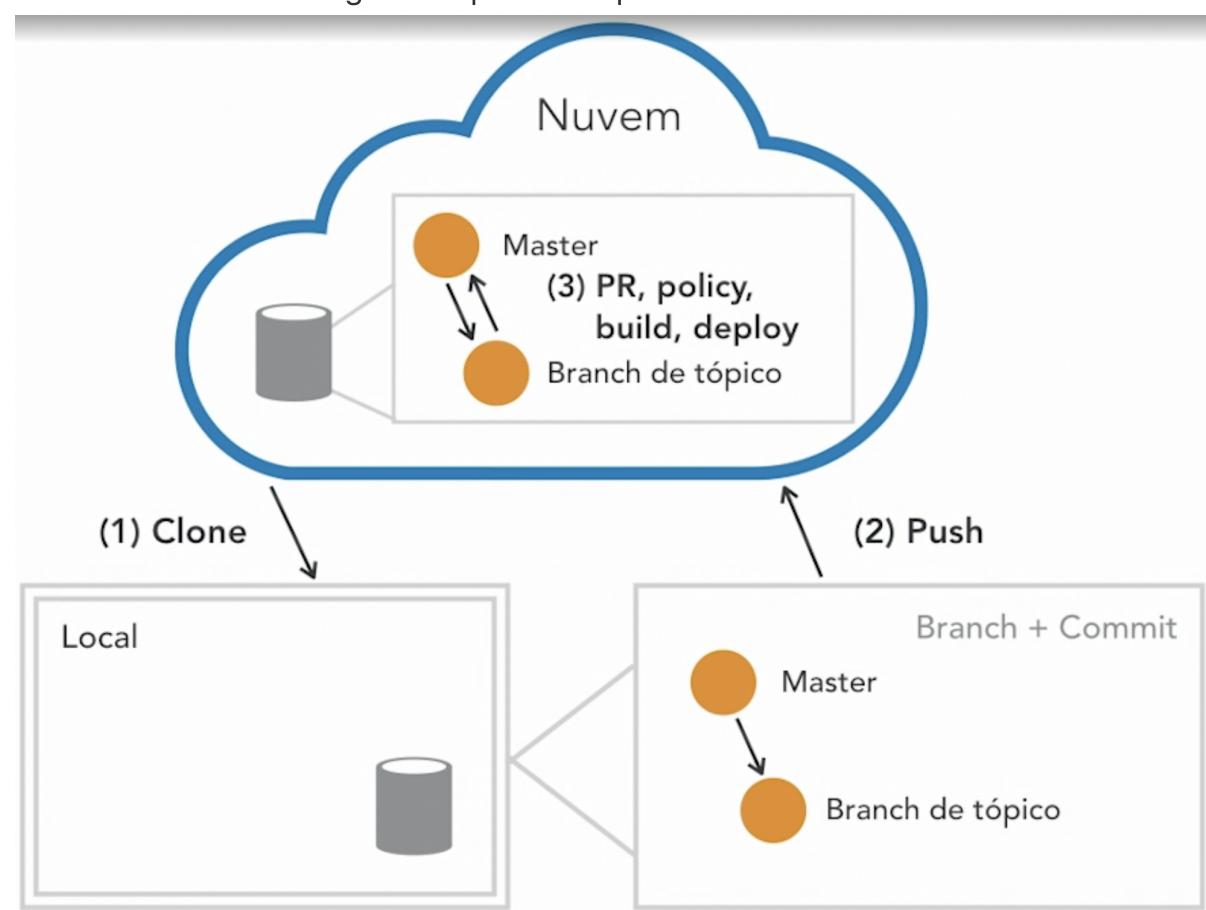
### APRENDER GERENCIAMENTO DE CÓDIGO

No gerenciamento de código, podemos gerenciar milhares de arquivos, assim como várias extensões desses arquivos.

E com isso, realizar uma integração e colaboração entre times de desenvolvimento.

Para isso, utilizamos o Git.

Podemos criar um código ou copiar um repositório externo.



### COMO VERSIONAR UM CÓDIGO NO AZURE REPOS

## **Repositório Git**

Um repositório, é um diretório ou espaço de armazenamento onde vivem seus projetos.

Podemos ter um ou mais repositórios Git.

## **Commit**

Comando para salvar o status das alterações feitas no repositório em um ponto no tempo.

Sendo assim, a cada nova alteração que fazemos, realizamos um commit.

## **Pull Request**

Processo que pode ser usado para revisar o código em uma ramificação de recurso, antes de mesclá-lo em outra ramificação (master).

Após realizar o commit, podemos fazer o pull request. Como boa prática. Pois assim podemos fazer alguns testes específicos e a revisão do nosso código.

---

## **IMPORTE UM REPOSITÓRIO DE OUTRO GIT**

Para utilizar o versionamento de código, devemos utilizar o menu do sistema chamado 'Repos'.

Nesse menu teremos nosso primeiro repositório, porém vazio.

Aqui podemos ver os repositórios já existentes, criar um novo repositório ou importar.

---

## **CLONE SEU REPOSITÓRIO DE CÓDIGO**

Para usar o código localmente, precisamos acessar no azure > 'Repos'.

Vamos clonar através do Visual Studio Code.

Clicamos no botão "Clone", no canto superior direito da tela.

The screenshot shows a dark-themed Azure DevOps repository page for a project named 'DevOps101\_Scrum'. On the left, there's a sidebar with project navigation and a file tree. The main area shows a list of files with their last change dates and commit history. At the top right, there are two buttons: 'Set up build' and 'Clone'. A red arrow points to the 'Clone' button. Below the file list, there's a section titled 'DevOps 101' with language information ('Language: ASP.NET Core') and a course identifier ('AZ-400 - Dia 2').

Name ↑	Last change	Commits
.github	3 de jan.	bca7e73b Add or update the Azur...
WebApplication	24 de ago. de 2021	bfd52d18 Update Index.cshtml J...
.gitignore	19 de jan. de 2019	8676ac14 Initial commit Jaquelin...
azure-pipelines.yml	1 de out. de 2021	d73de3f5 Update azure-pipeline...
README.md	23 de nov. de 2021	c5c72db5 Update README.md J...
SECURITY.md	14 de ago. de 2020	e1851c2b Create SECURITY.md J...
WebApplication.sln	28 de jul. de 2020	7a9d7225 change jaquecr

## APlique seu primeiro commit local e remotamente

Primeiramente precisamos criar uma Branch/Feature.

Ela pode ser criada tanto no Visual Studio Code ou no Azure DevOps.

Após realizar a criação da branch, precisamos alterar nosso repositório apontando para a branch criada, realizar a alteração desejada e fazer o commit (Push e Pull).

Pronto, se olhar para o site do Azure DevOps a nova branch com a alteração realizada, já deve constar lá.

## CONCEITOS DE PULL REQUESTS

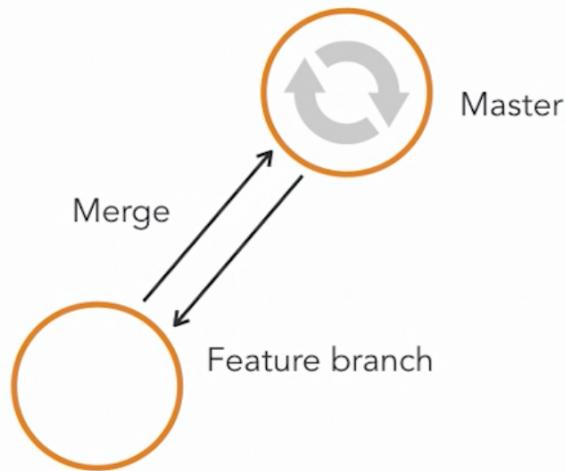
O Pull é uma maneira de aplicarmos políticas e evitarmos que um código incorreto siga para uma branch específica, seja ela master, develop, etc.

Com ele, podemos aplicar as políticas:

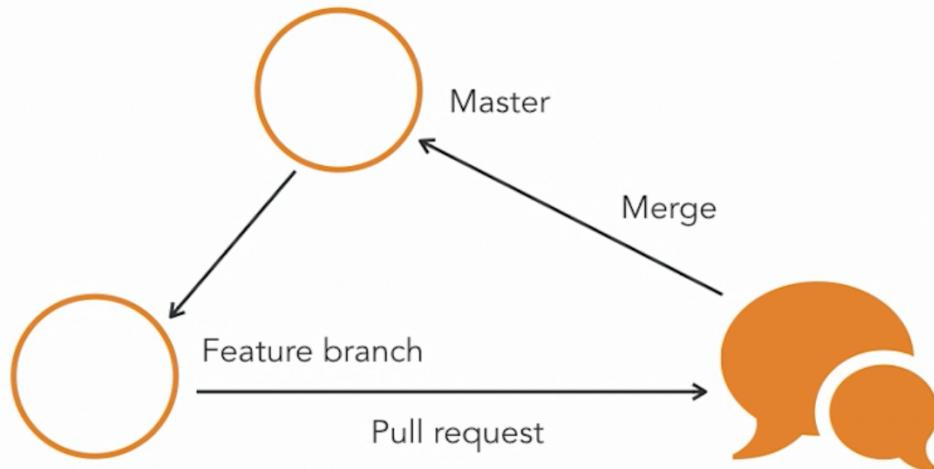
- Incluir um build para compilar e testar esse código antes que ele passe para uma branch;
- Política de Work item;

- Revisor de código.

O mais comum é quando criamos uma branch, desenvolvemos e depois voltamos isso para a master fazendo o merge. Como mostra a imagem abaixo.



Com o Pull request, também será criada a branch, porém antes de realizar o merge, o código passará pela validação de política do Pull e só depois de aprovado que o merge é realizado na master. Como a imagem abaixo.



## PULL REQUEST E AZURE REPOS

Para aplicar o primeiro Pull, precisamos entender a alteração que você realizou.

No site do Azure DevOps, temos um menu na seção 'Repos' chamado 'Pull requests'.

A screenshot of the Azure DevOps interface, specifically the 'Pull requests' section for the 'DevOps101\_Scrum' repository. The left sidebar shows navigation options like Overview, Boards, Repos, and Pull requests, with a red arrow pointing to the 'Pull requests' link. The main content area displays a message: 'No pull requests match the given criteria'. Below this message is a brief description of what pull requests are used for and a 'New pull request' button.

Nesse caso, não temos nenhum Pull request em aberto.

Aqui é onde criamos as regras de Pull, para uma branch Master (por exemplo).

**\*\*Lembrando que branch master NUNCA deve receber código diretamente, sem uma validação\*\***

Clicando em "New Pull request" o sistema abre a tela para você criar o Pull request puxando a ultima alteração realizada, inclusive com a descrição que foi colocada no commit.

Após preencher as informações (aprovador, work item, tags), tem opção de ver os arquivos que foram alterados e quais são os commits que estão nesse Pull request.

Após salvar, o Pull é criado.

Ele já valida se existe conflitos também.

The screenshot shows a pull request interface in a DevOps tool. The title of the pull request is "inclusão de versão". The status bar at the top right shows "Active" and "Gustavo Ragghi feature into master". On the left sidebar, "Pull requests" is selected. The main area has tabs for "Overview", "Files", "Updates", and "Commits", with "Overview" selected. Below the tabs, there's a message "No merge conflicts Last checked 2m ago". The "Description" field contains the text "inclusão de versão". A "Show everything (1)" button is visible. In the "Reviewers" section, it says "Required" and "No required reviewers". In the "Optional" section, there's a single entry for "Gustavo Ragghi" with the note "No review yet". The "Tags" section shows "No tags". The "Work items" section shows "No work items". At the bottom, there's a comment input field with "Add a comment..." placeholder text. A history entry at the bottom says "Gustavo Ragghi created the pull request 2m ago".

Podemos também adicionar comentários, para tratar algum ponto que não está ok.

Antes de completar, precisamos aprová-lo.

## BOAS PRÁTICAS PARA ESTRATÉGIAS DE BRANCHES E PULL REQUESTS

- Criar *feature branches* para trabalhar em novas funcionalidades;
- Use sintaxe comum para nomes de *branches* para identificar o *owner* e o escopo;
- Use nomes descritivos para commits, víncule work item;
- Alavanque a utilização de *pull request* para controle de qualidade e dependências;
- Delete *branches* depois do merge.

## 3. Como Compilar seu Código