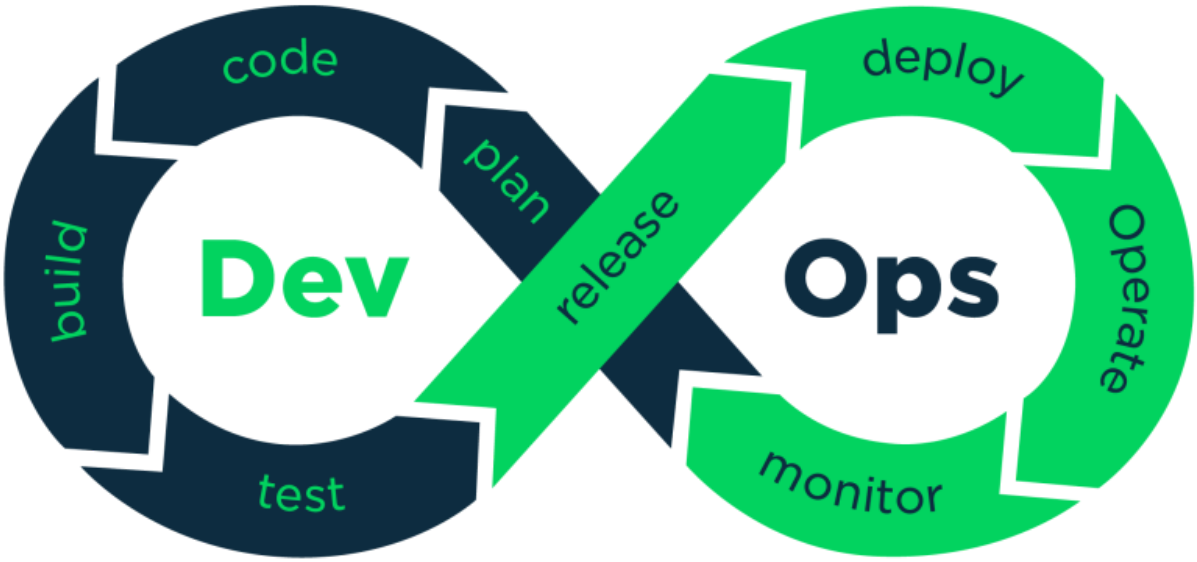
Estudos DevOps

CI/CD Pipelines



CI/CD significa Integração Contínua e Entrega/Implantação Contínua, e refere-se a um conjunto de práticas e ferramentas que ajudam a automatizar o processo de construção, teste e implantação de

aplicativos de software. Um pipeline é uma série de estágios, cada um com seu próprio conjunto de tarefas, que um aplicativo deve passar antes de ser lançado para produção.

Um pipeline típico para CI/CD inclui as seguintes etapas:

1. Mudanças de código: os desenvolvedores fazem alterações na base de código e as enviam para um sistema de controle de versão como o Git.
2. Construção: o código é compilado e quaisquer dependências são baixadas e configuradas.
3. Teste: o aplicativo é testado quanto a erros e bugs usando ferramentas de teste automatizadas.
4. Implantação: o aplicativo é implantado em um ambiente de preparo onde pode ser testado ainda mais.
5. Lançamento: se o aplicativo passar em todos os testes, ele é lançado para produção.

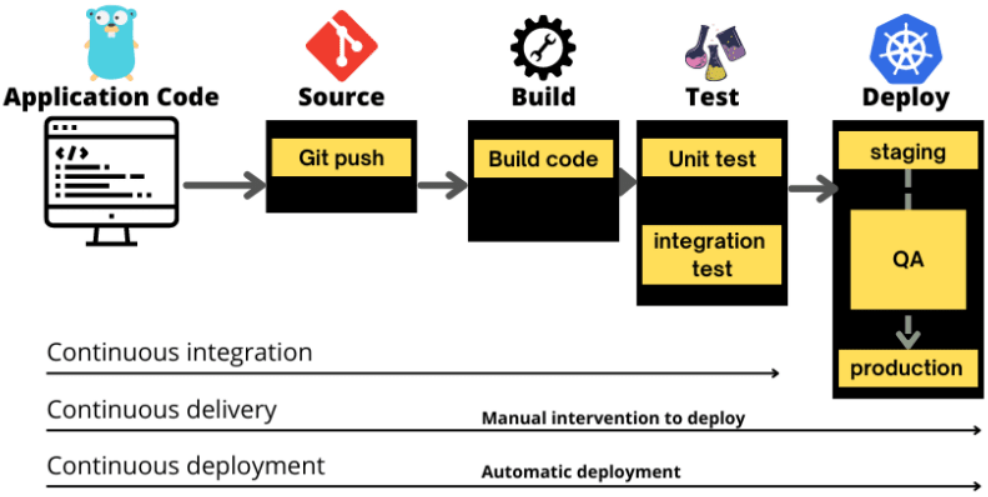
Integração contínua

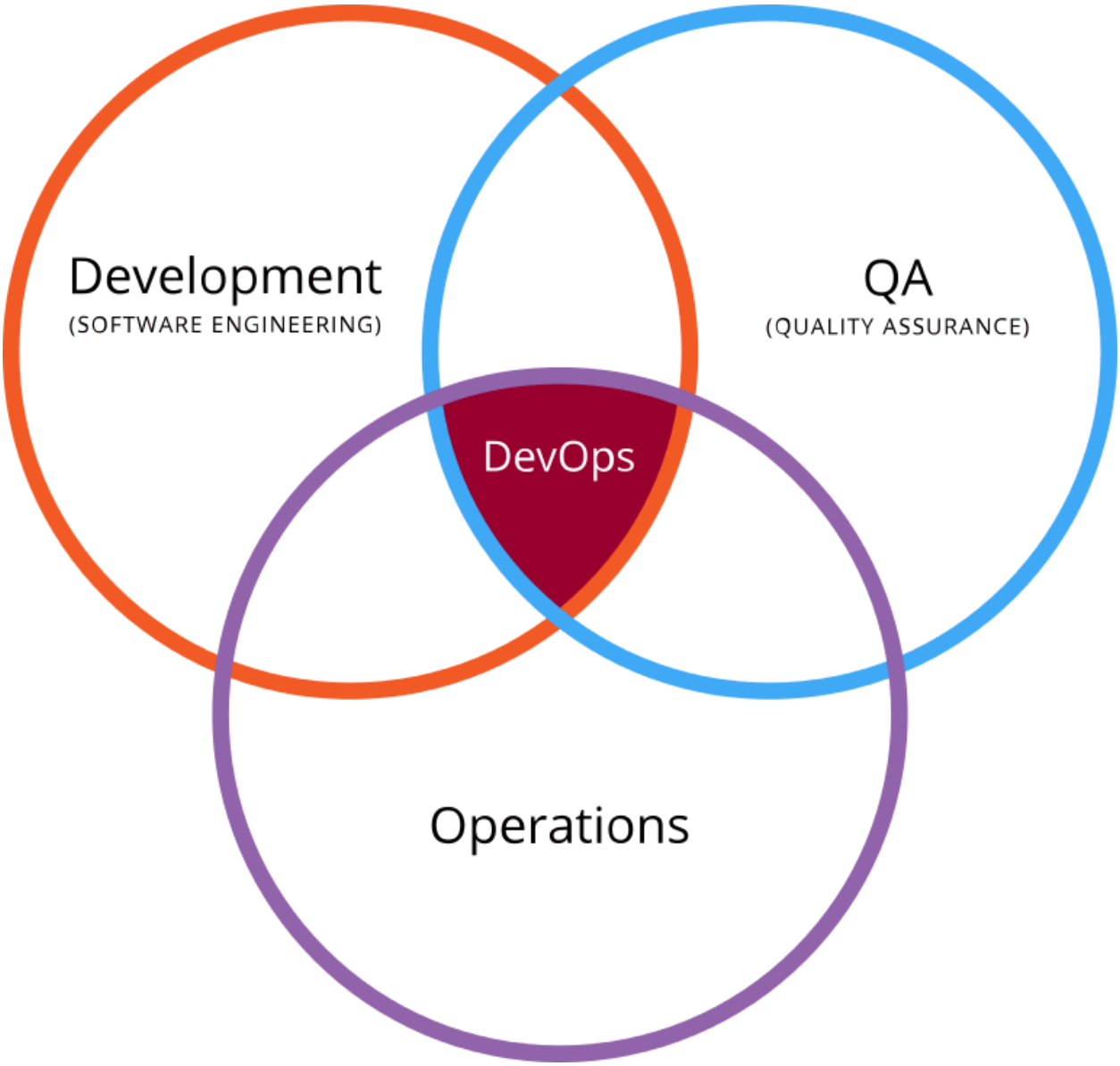
A Integração Contínua significa que os desenvolvedores integram suas mudanças na base de código principal com frequência, geralmente várias vezes ao dia. Isso garante que conflitos de código sejam identificados e resolvidos cedo, reduzindo o risco de bugs e erros no código.

A Entrega/Implantação Contínua significa que as mudanças são automaticamente implantadas na produção assim que passarem em todos os testes. A Implantação Contínua é uma abordagem mais automatizada, enquanto a Entrega Contínua requer que um humano aprove o lançamento para produção.

Em geral, um pipeline de CI/CD ajuda desenvolvedores e organizações a lançar software mais rapidamente, com menos bugs e com maior confiança.

A imagem abaixo ilustra bem o processo:


Application Code
Source
Build
Test
Deploy
Git push
Build code
Unit test
integration test
staging
QA
production
Continuous integration
Continuous delivery
Manual intervention to deploy
Continuous deployment
Automatic deployment



Projeto Esteira, Exercício T-Systems:

https://sbs.t-systems.com.br/gitlab/mercedes/collab/planos-de-desenvolvimento/desenvolvimento-de-software/plano-de-formacao-inicial-backend/-/blob/stable/semanas/6.md

Exercício

Um usuário nos informou que necessita realizar um **CRUD** de caminhões. Após levantamento, temos a seguinte especificação técnica:

- Precisaremos criar uma aplicação Web em **C#** do tipo **ASP.NET Core Web App (Model-View-Controller)** na versão **.NET 6.0 (Long-term support)** e sem autenticação;

A entidade **Caminhão** terá os seguintes atributos:
- Identificação: número inteiro;
- Nome: string;
- Descrição: string;
- Data de Criação: Data e hora;

- A tela inicial da aplicação deverá listar todos os caminhões cadastrados;
- Os campos para serem apresentados na listagem de caminhões são:
  - Nome
  - Descrição
  - Data de Criação
- Para cada caminhão da lista deve haver 3 botões:
  - *Editar* -> Deve direcionar o usuário para uma tela para editar os campos do caminhão selecionado;
  - *Detalhes* -> Deve direcionar o usuário para uma tela para uma tela para apresentar os detalhes do caminhão selecionado;
  - *Excluir* -> Deve direcionar o usuário para uma tela para uma tela com confirmação para que o usuário possa definir se realmente quer deletar o caminhão selecionado;
- Ainda na tela inicial deve haver um botão para cadastrar um novo caminhão;
- No menu principal da aplicação é necessário um item chamado Caminhões, onde direciona o usuário para a tela de listagem;

> Observação: Os dados dos caminhões precisam ser salvos em um banco de dados. Use um banco local ou em memória com o Entity Framework.

Vamos lá!

Para criar a aplicação ASP.NET Core Web App, siga os seguintes passos:

1. Abra o Visual Studio e selecione "Create a new project".
2. Selecione "ASP.NET Core Web App (Model-View-Controller)" e clique em "Next".
3. Dê um nome ao projeto e selecione a pasta onde deseja salvá-lo. Clique em "Create".
4. Selecione a versão .NET 6.0 e marque a opção "Enable long-term support (LTS)".
5. Selecione a opção "No Authentication" e clique em "Create".

Após criar o projeto, crie a entidade Caminhão e seu contexto no banco de dados usando o Entity Framework Core. Para isso, crie uma nova classe Caminhao com os atributos solicitados:

```
public class Caminhao
{
    public int Id { get; set; }
    public string Nome { get; set; }
    public string Descricao { get; set; }
    public DateTime DataCriacao { get; set; }
}
```

Em seguida, vamos criar um contexto para o banco de dados que herda de DbContext e definir a propriedade DbSet<Caminhao>:

```
public class CaminhaoContext : DbContext
{
    public CaminhaoContext(DbContextOptions<CaminhaoContext> options) : base(options) { }

    public DbSet<Caminhao> Caminhoes { get; set; }
}
```

Para modificar a tabela gerada pela scaffolding na view Index.cshtml, siga os seguintes passos:

1. Abra o arquivo Index.cshtml localizado em "Views/Caminhoes".
2. Remova as colunas que não são necessárias (por exemplo, "Id" e "Timestamp").
3. Adicione as colunas "Nome", "Descrição" e "Data de Criação" na tabela, utilizando as propriedades correspondentes da classe Caminhao.
4. Adicione os botões de "Editar", "Detalhes" e "Excluir" em cada linha da tabela, utilizando os links gerados pelo scaffolding.

```
<table class="table">
    <thead>
        <tr>
            <th>Nome</th>
            <th>Descrição</th>
            <th>Data de Criação</th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>@Html.DisplayFor(modelItem => item.Nome)</td>
                <td>@Html.DisplayFor(modelItem => item.Descricao)</td>
                <td>@Html.DisplayFor(modelItem => item.DataCriacao)</td>
                <td>
                    <div class="btn-group" role="group">
                        <a class="btn btn-primary" asp-action="Edit" asp-route-id="@item.Id">Editar</a>
                        <a class="btn btn-info" asp-action="Details" asp-route-id="@item.Id">Detalhes</a>
                        <a class="btn btn-danger" asp-action="Delete" asp-route-id="@item.Id">Excluir</a>
                    </div>
                </td>
            </tr>
        }
    </tbody>
</table>
```