

MBA  
USP  
ESALQ

# DevOps

Rogério Rodrigues

\*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados é do professor.

**Proibida a reprodução**, total ou parcial, sem autorização. Lei nº 9610/98



# Introdução ao DevOps

# O que é DevOps?

"O DevOps é a união de pessoas, processos e produtos que tem como objetivo gerar valor continuamente para os usuários finais."

— Donovan Brown,  
[O que é DevOps?](#)





# História do DevOps

- Origens e evolução
- A convergência de desenvolvimento e operações
- A influência do Agile

# A Cultura DevOps

- Mentalidade de colaboração
- Quebrando silos
- Cultura de aprendizado e experimentação
- Adaptação de processos





# Princípios Chave do DevOps

Culture – People > Process > tools

Automation – Infrastructure as Code

Lean – Focus on value and customer

Measurement – Measure everything

Sharing – Collaboration/Feedback



# Benefícios do DevOps

- Entregas mais rápidas
- Soluções mais confiáveis
- Maior colaboração entre equipes
- Melhor satisfação do cliente
- Velocidade em se adaptar ao mercado



# Boas Práticas populares no DevOps

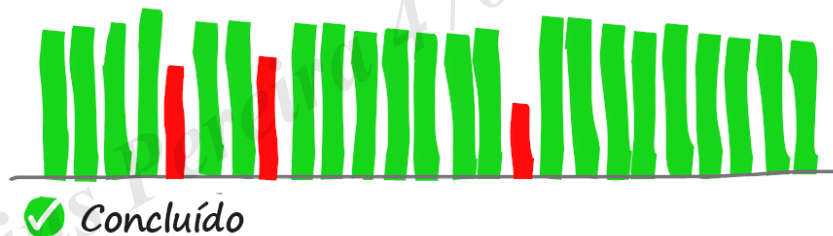
- Gestão Ágil
- Estratégias de Branch
- Estratégias de Release
- Contínuos Integration
- Continuos Deployment
- Contínuos Delivery
- Monitoring
- Feedback
- IaC
- SecOps

# Explorar o percurso do DevOps

COMPILAÇÃO BEM-SUCEDIDA

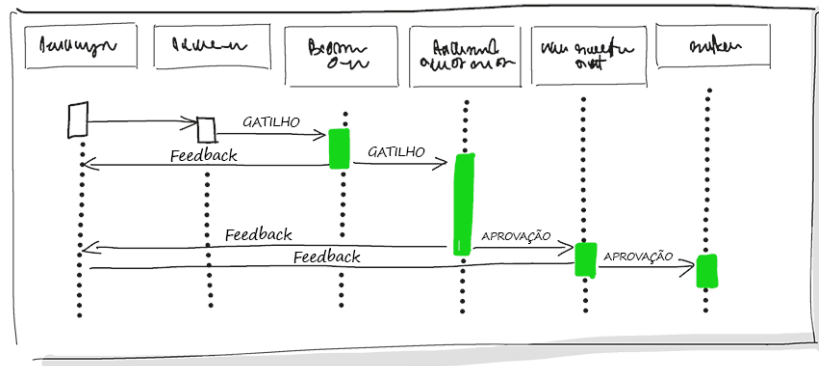
## Integração contínua

A Integração Contínua impulsiona a mesclagem contínua e o teste de código, o que leva à descoberta precoce de defeitos.



## Entrega contínua

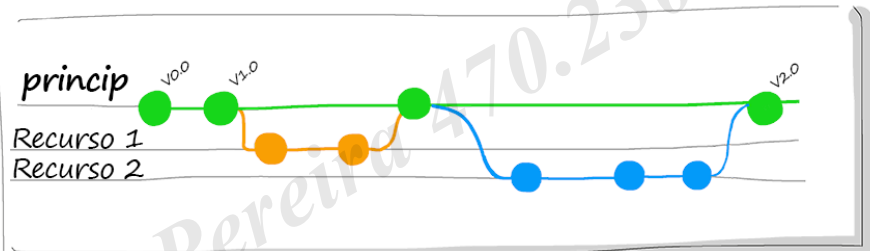
A entrega contínua de soluções de software para ambientes de produção e teste ajuda as organizações a corrigir rapidamente bugs e responder a requisitos de negócios em constante mudança.



# Explorar o percurso do DevOps

## Controle de versão

O controle de versão, geralmente com um repositório baseado em Git, permite que as equipes localizadas em qualquer lugar do mundo se comuniquem efetivamente durante as atividades de desenvolvimento diário.

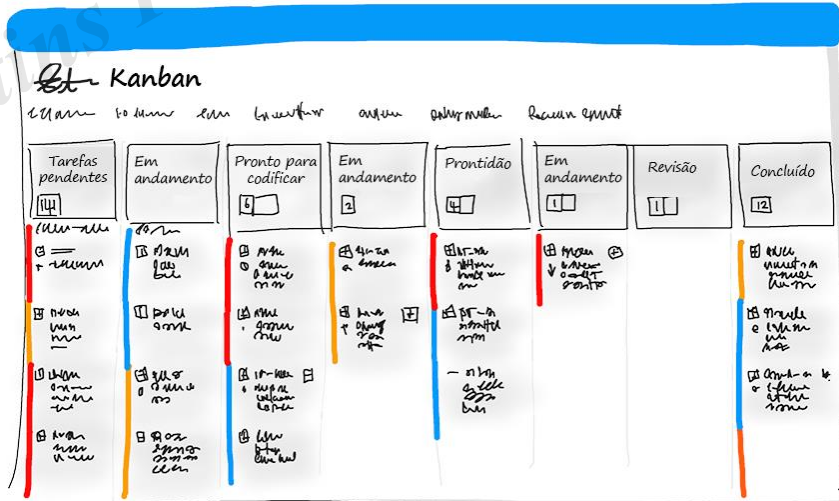


## Agile/lean

Planeje e isole o trabalho em sprints.

Gerencie a capacidade da equipe e ajude as equipes a se adaptarem rapidamente às necessidades de negócios em constante mudança.

Uma Definição de Concluído do DevOps é um software de trabalho que coleta telemetria em relação às metas de negócios pretendidas.





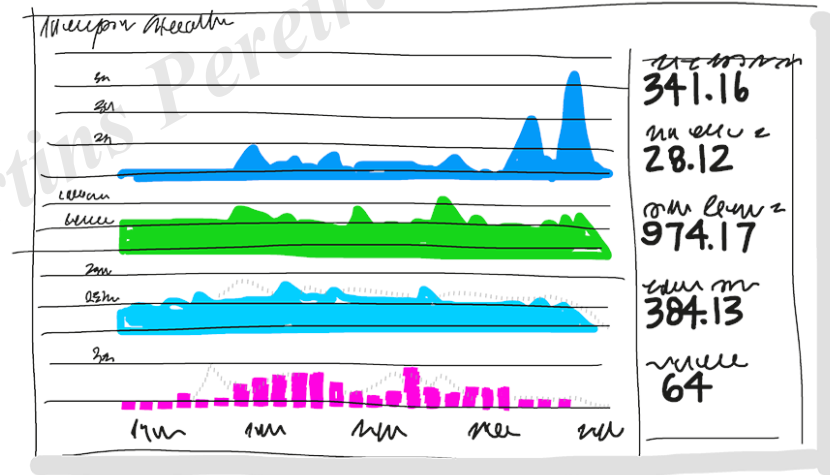
# Explorar o percurso do DevOps

## Monitoramento e registro em log

Monitoramento e registro em log de aplicativos em execução.

## Nuvem

- Nuvens públicas e híbridas facilitaram o impossível.



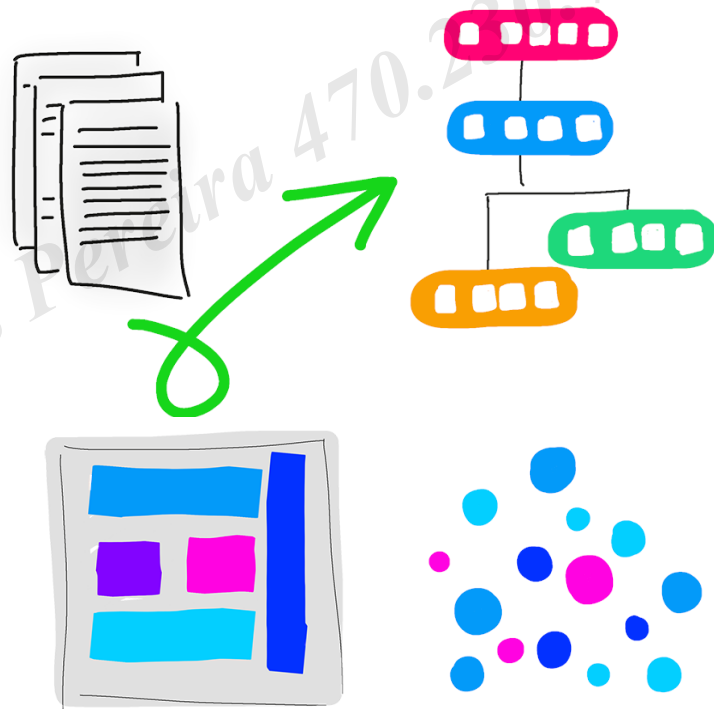
# Explorar o percurso do DevOps

## IaC (infraestrutura como código)

Permite a automação e a validação da criação e da desinstalação de ambientes para fornecer plataformas de hospedagem de aplicativos seguras e estáveis.

## Microserviços

Isole casos de uso empresarial em pequenos serviços reutilizáveis que se comunicam por meio de contratos de interface.



MONOLÍTICO/EM CAMADAS

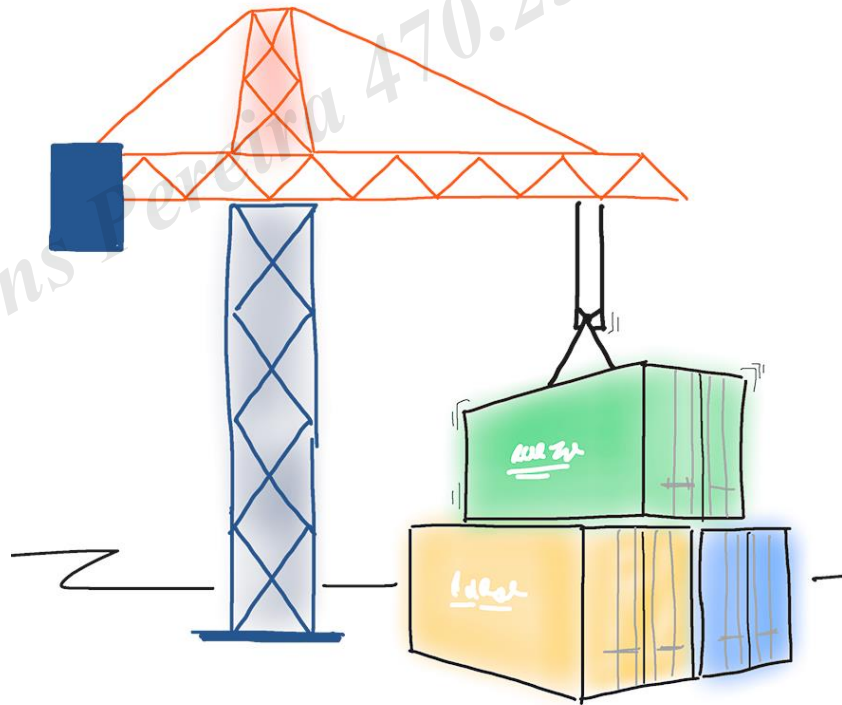
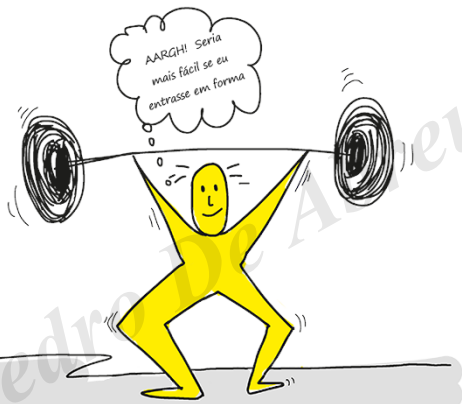
MICROSSERVIÇOS

# Explorar o percurso do DevOps

## Contêineres

Contêineres são a próxima evolução na virtualização.

**Inicialmente, o DevOps pode prejudicar**





# Ferramentas populares no DevOps

- Azure DevOps
- Github
- Gitlab
- CircleCI e CD
- Jenkins
- Cheff
- Puppet
- Octopus
- Etc...

# Explorar práticas de desenvolvimento Agile

1

## Abordagem em cascata:

- Define, analisa, constrói, testa e entrega
- Requisitos difíceis de definir com precisão, que podem mudar ao longo do tempo, inclusive durante o desenvolvimento
- Requer solicitações de alteração e custo adicional após a entrega

2

## Abordagem Agile:

- Destaca constantemente o planejamento adaptável e a entrega antecipada com melhoria contínua.
- Os métodos de desenvolvimento usam como base lançamentos e iterações
- No final de cada iteração, deve haver um código em funcionamento testado
- Foco em resultados de curto prazo

# Explorar os princípios do desenvolvimento Agile

- 1 Atender ao cliente por meio da entrega antecipada e contínua de um software valioso
- 2 Aceitar requisitos em mudança
- 3 Fornecer software funcional com frequência
- 4 Trabalhar em conjunto durante todo o projeto
- 5 Criar projetos com auxílio de indivíduos motivados
- 6 Usar conversas presenciais
- 7 Avaliar o progresso por meio do software de trabalho
- 8 Os processos do Agile promovem o desenvolvimento sustentável
- 9 Atenção contínua à excelência técnica e ao bom design
- 10 Simplicidade – A arte de maximizar a quantidade de trabalho não realizado
- 11 Usar equipes auto-organizáveis
- 12 Refletir sobre como aprimorar a eficácia

Fonte: <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>



# Identificar equipes de transformação

1

**Existem vários desafios ao criar equipes:**

- Disponibilidade da equipe
- Interrupção dos procedimentos e processos atuais

2

**Para superar os desafios, crie uma equipe que seja:**

- Focada na transformação
- Bem respeitada em suas áreas
- Interna e externa ao negócio



**Um projeto de transformação pode entrar em conflito com as necessidades contínuas dos negócios**



# Como começar a implementar a cultura DevOps?

- Dê um foco nas pessoas
- Entenda o negócio
- Converse muito
- Não seja arrogante ou qualquer outra expressão popular (-;
- Alinhe os processos
- Teste as soluções

# Explorar metas compartilhadas e definir linhas do tempo

**1** Os projetos devem ter um conjunto claramente definido de resultados mensuráveis, como:

- Reduzir o tempo gasto na correção de bugs em 60%
- Reduzir o tempo gasto com trabalho não planejado em 70%
- Reduzir a necessidade de horas extras da equipe para, no máximo, 10% do tempo de trabalho total
- Remover todos os patches diretos de sistemas de produção



Um dos objetivos principais do DevOps é fornecer um valor maior para o cliente, portanto, os resultados devem ter foco no valor para o cliente.



# Explorar metas compartilhadas e definir linhas do tempo

- 1 Metas mensuráveis devem ter prazos desafiadores, mas possíveis
- 2 As linhas do tempo devem ser uma série constante de metas de curto prazo, cada uma clara e mensurável
- 3 Linhas do tempo mais curtas têm vantagens:
  - Facilidade para alterar planos ou prioridades quando necessário
  - Redução do atraso entre a realização do trabalho e o recebimento de comentários
  - Facilidade para manter o suporte organizacional quando os resultados positivos são claros



# Desafios na Implementação

- Mudança de mentalidade
- Necessidade de treinamento
- Resistência à mudança

# Mitos sobre DevOps

- DevOps é Automação de Pipelines
- DevOps é resolver problema de Deploy
- DevOps é troubleshoot em produção
- DevOps é transportar toda a infraestrutura como código, vulgo IAC
- DevOps é contratar uma ferramenta top do mercado

# Escolhendo o projeto adequado

# Explorar projetos Greenfield e Brownfield

1

Projetos de software Greenfield são desenvolvidos em um ambiente totalmente novo.

---

2

Os projetos de software Brownfield são desenvolvidos na presença imediata de aplicativos/sistemas de software já existentes.



# Decidir quando usar projetos Greenfield e Brownfield

1

## Projetos Greenfield:

- Parece ser um ponto de partida mais fácil
- Uma tela em branco oferece a oportunidade de implementar tudo como você deseja.

2

## Projetos Brownfield:

- Vem com a bagagem de bases de código e equipes já existentes e, muitas vezes, uma grande quantidade de dívida técnica
- O gasto de tempo com a manutenção de aplicativos Brownfield já existentes limita a capacidade de trabalhar em novos códigos



Há um equívoco comum de que o DevOps é melhor para projetos Greenfield do que projetos Brownfield. Esse não é o caso.

# Decidir quando usar sistemas de registro em relação a sistemas de envolvimento

1

## Sistemas de registro:

- Destaque na precisão e na segurança
- Fornece a verdade sobre os elementos de dados
- Historicamente evoluem de forma lenta e cuidadosa

2

## Sistemas de participação:

- São mais exploratórios
- Usam a experimentação para resolver novos problemas
- São modificados regularmente
- Priorizam alterações rápidas em relação a garantir que as alterações estejam corretas



Ambos os tipos de sistemas são importantes

# Identificar grupos para minimizar a resistência inicial

1

## Diferentes tipos de membros da equipe:

- Canários testam voluntariamente características avançadas
- Os usuários pioneiros avaliam voluntariamente as versões prévias
- Usuários consomem os produtos depois dos canários e dos usuários pioneiros

2

## Metas de aprimoramentos ideais:

- Podem ser usadas para obter ganhos antecipados
- São pequenas o suficiente para ser alcançadas em um prazo razoável
- Trazem benefícios significativos o suficiente para serem evidentes para a organização

# Identificar métricas de projeto e KPIs (indicadores chave de desempenho)

**1** **Resultados mais rápidos** – Frequência, velocidade e tamanho da implantação, além do prazo de entrega.

**2** **Eficiência** – Proporção entre servidor e administrador, proporção de funcionários e clientes, uso e desempenho de aplicativos.

**3** **Qualidade e segurança** – Taxas de falha de implantação, taxas de falha de aplicativos, tempo médio de recuperação, taxas de relatórios de bugs, taxas de aprovação em testes, taxa de escape de defeitos, disponibilidade, cumprimento de SLA (Contrato de Nível de Serviço) e tempo médio de detecção.

**4** **Cultura** – Moral dos funcionários e taxas de retenção.



As metas devem ser específicas, mensuráveis e com prazo determinado

# Estruturas de equipe

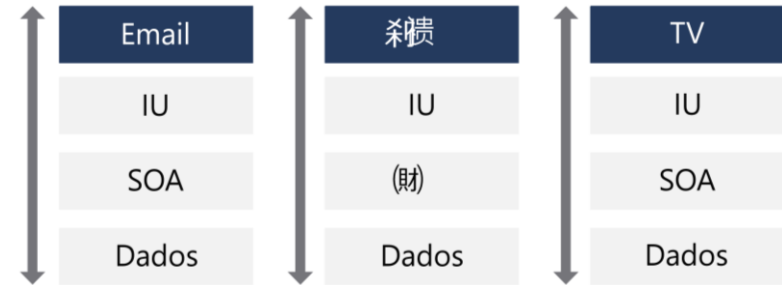


# Definir a estrutura da organização para práticas Agile

As estruturas horizontais dividem as equipes de acordo com a arquitetura do software.



Equipes verticais abrangem a arquitetura e são alinhadas aos resultados do produto. É possível dimensioná-las adicionando equipes.



Ficou comprovado que as equipes verticais fornecem resultados melhores em projetos do Agile

# Explorar os membros ideais da equipe de DevOps

- 1
  - Acreditam que é necessário fazer mudanças e demonstraram a capacidade de inovar
  - São respeitados e têm um amplo conhecimento da organização e de suas operações
  - O ideal é que eles já acreditem que as práticas de DevOps são necessárias

---

- 2

Muitas equipes contratam instrutores ou mentores externos para Agile

---

- 3

Instrutores do Agile têm habilidades de ensino e orientação

---

- 4

Os instrutores do Agile devem ser orientadores e consultores

---

- 5

Alguns instrutores são especialistas técnicos

---

- 6

Alguns instrutores focam em processos do Agile



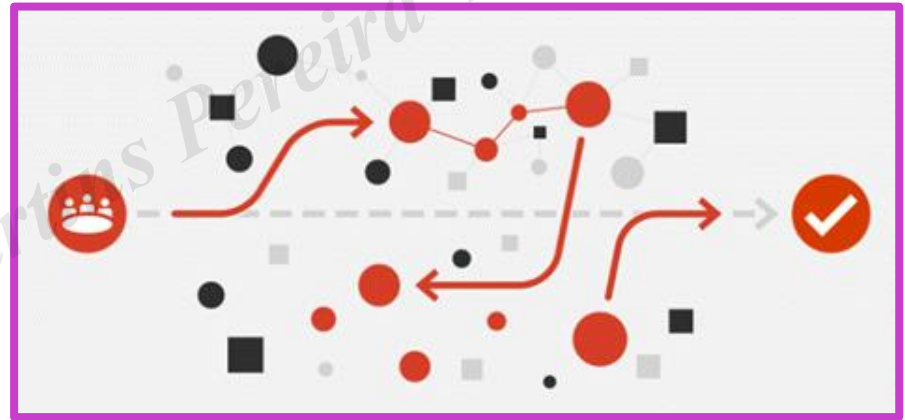
A expectativa é que os membros da equipe aprendam uns com os outros à medida que trabalham e adquirem habilidades

# Habilitar a colaboração na equipe e entre equipes

**Mudanças culturais** – Espaços de trabalho mais abertos, etiqueta de reunião, terceirização e comunicação aprimorada;

**Equipes multifuncionais** – Colaboração com outras pessoas, diversidade de opiniões, recompensas pelo comportamento coletivo;

**Ferramentas de colaboração** – Slack, Teams, Asana, Glip, JIRA.



# Selecionar ferramentas e processos para práticas Agile

Muitas vezes, as ferramentas podem melhorar os resultados alcançados;

Ferramentas físicas, como quadros brancos, catões de índice e notas adesivas;

Ferramentas de gerenciamento de projetos, como quadros Kanban para planejamento, monitoramento e visualização;

Ferramentas de gravação de tela para registrar bugs, criar instruções passo a passo e demonstrações.



# Escolhendo ferramentas de DevOps



# O que é o Azure DevOps?

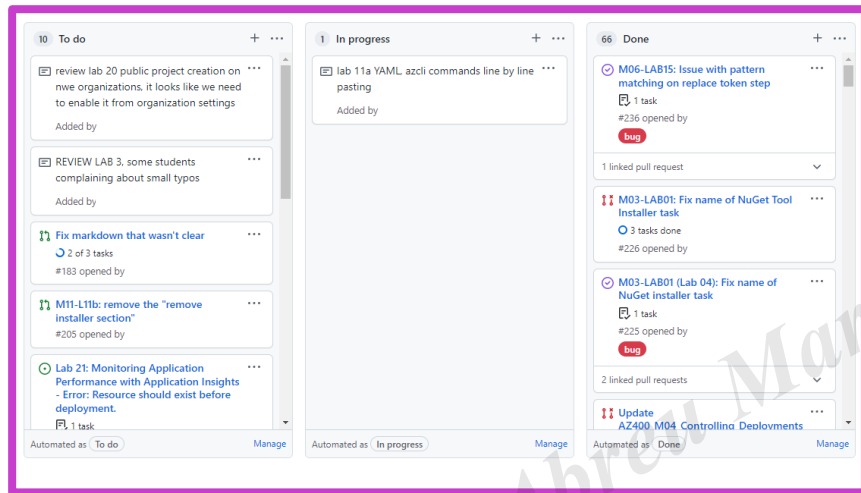
- 1 **Azure Boards:** planejamento do Agile, acompanhamento de itens de trabalho, visualização e ferramenta de relatório;
- 2 **Azure Pipelines:** uma plataforma de CI/CD de linguagem, plataforma e nuvem agnóstica com suporte para contêineres ou Kubernetes;
- 3 **Azure Repos:** oferece repositórios git privados hospedados em nuvem;
- 4 **Azure Artifacts:** oferece gerenciamento de pacotes integrado com suporte para os feeds de pacote Maven, npm, Python e NuGet de fontes públicas ou privadas;
- 5 **Azure Test Plans:** oferece uma solução integrada de testes planejados e exploratórios.

# O que é o GitHub?

- 1 **Codespaces:** fornece ambientes de desenvolvimento colaborativo hospedados em nuvem;
- 2 **Repos:** fornece repositórios git locais e hospedados em nuvem para projetos públicos e privados;
- 3 **Ações:** cria fluxos de trabalho de automação com variáveis de ambiente e scripts personalizados;
- 4 **Pacotes:** facilita a integração com vários pacotes já existentes e repositórios de código aberto;
- 5 **Segurança:** analisa as vulnerabilidades de código e identificação no início do ciclo de desenvolvimento.

# Planejamento ágil com projetos do GitHub e o Azure Boards

# Introdução aos painéis de projetos e projetos do GitHub



The screenshot shows the GitHub Projects board with a list of tasks. The table has three columns: Title, Assignees, and Status. The tasks are listed in a table format.

|   | Title                                      | Assignees | Status           |
|---|--|-----------|------------------|
| 1 | Add Travis CI migration table              | octocat   | Ready for review |
| 2 | Using a package without installing package | monalisa  | Ready            |
| 3 | Add functionality to hide images           | octocat   | In Progress      |
| 4 | Update contributing guide                  | octocat   | Ready for review |
| 5 | Fix markdown tables in translated content  | monalisa  | Ready            |
| 6 | add copy button to examples                | octocat   | In Progress      |
| 7 | Validate Java Gradle wrapper               |           | Todo             |
| 8 | GPC creation on linux                      |           | Todo             |
| 9 | Add domains for self-hosted runners        |           | Todo             |

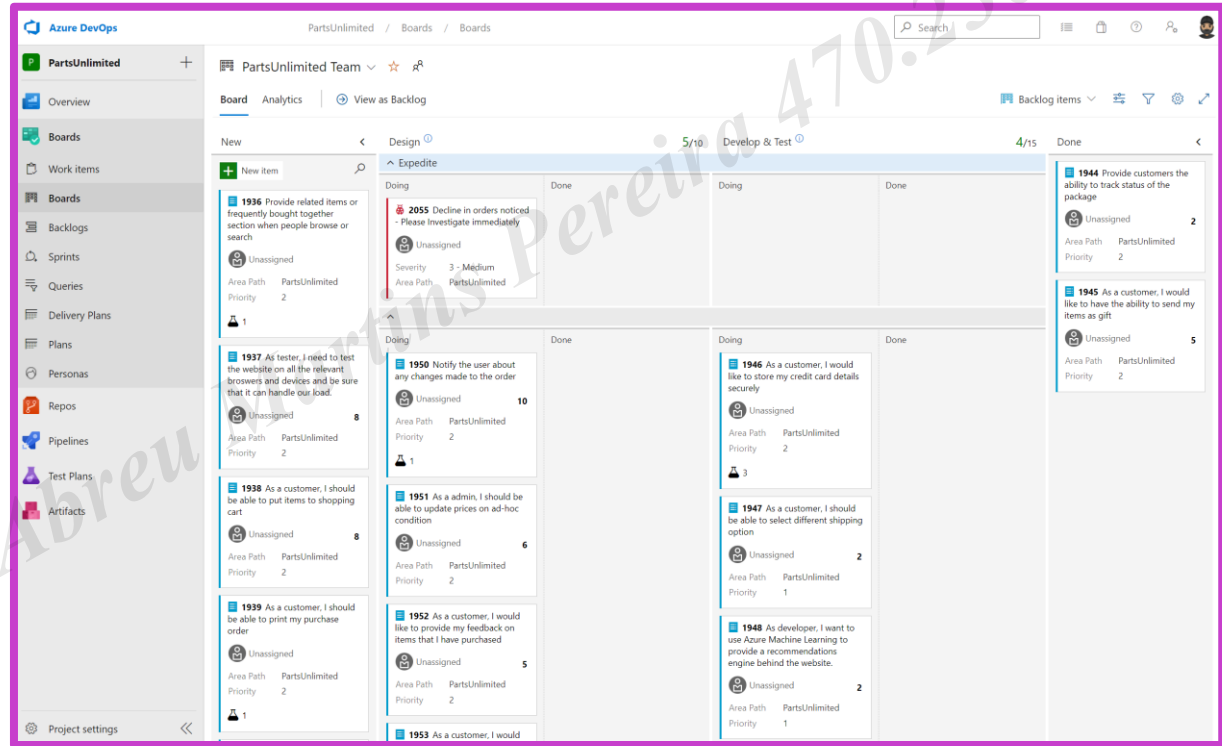
Os quadros de projeto do GitHub Projects são compostos por problemas, solicitações de pull e anotações categorizadas como cartões que você pode arrastar e soltar nas colunas escolhidas.

Os **GitHub Projects** são uma versão nova, personalizável e flexível das ferramentas de projetos para planejar e acompanhar o trabalho no GitHub.

# Introdução ao Azure Boards

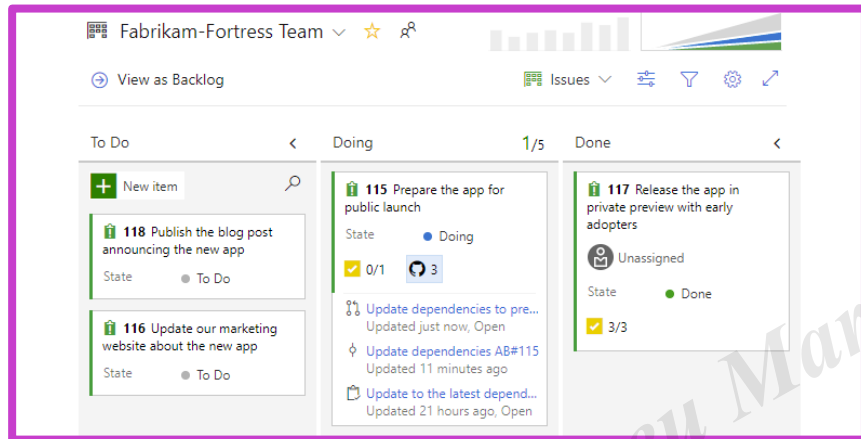
Processos do Agile, Scrum e Kanban por padrão.

- Acompanhe o trabalho, os problemas e os defeitos de códigos associados ao projeto.

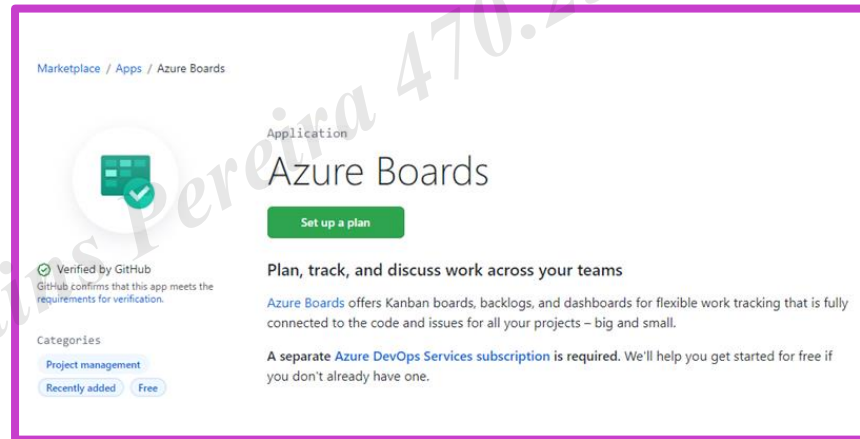




# Vincular GitHub a Azure Boards



O Aplicativo Azure Boards vincula itens de trabalho a commits, solicitações de pull e problemas.



Use Azure Boards para planejar e acompanhar seu trabalho e o GitHub como controle do código-fonte para desenvolvimento de software.

Autentique-se no GitHub usando um nome de usuário/senha ou um PAT (token de acesso pessoal).

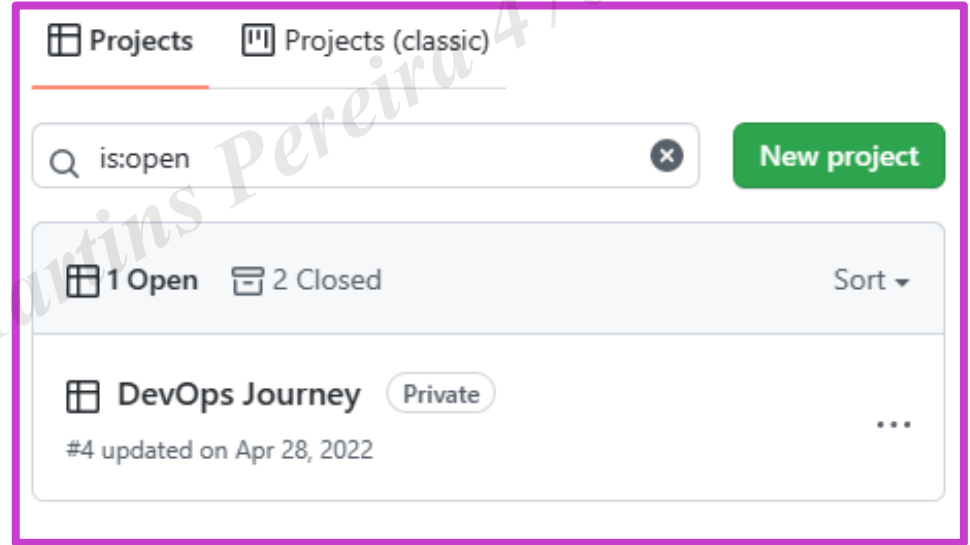
# Configurar projetos do GitHub

Você pode usar um modelo de quadro de projeto para criar um quadro de projeto com automação já configurada.

Você também pode copiar um quadro de projeto para reutilizar as personalizações dele em projetos semelhantes.

Você pode vincular até 25 repositórios a um de sua organização ou pertencente ao usuário.

Depois de criar seu quadro de projeto, você pode adicionar problemas, solicitações de pull e observações a ele.



# Gerenciar o trabalho com os quadros de projeto do GitHub

- Controle entregas de projetos
- Defina para qualquer período
- Inclua interrupções
- Datas de liberação
- Iterações para planejar os próximos trabalhos

The screenshot displays the GitHub Project Board interface. At the top, it shows '3 Active' and '4 Completed' items. A '+ Add iteration' button is in the top right. The main list contains four items:

| Status                       | Iteration   | Duration | Start Date | End Date | Action |
|------------------------------|-------------|----------|------------|----------|--------|
| Current                      | Iteration 5 | 2 weeks  | Feb 23     | Mar 08   | 🗑️     |
| <a href="#">Insert break</a> |             |          |            |          |        |
| Planned                      | Iteration 6 | 2 weeks  | Mar 09     | Mar 22   | 🗑️     |
| Break                        | 1 week      |          | Mar 23     | Mar 29   | 🗑️     |
| Planned                      | Iteration 7 | 2 weeks  | Mar 30     | Apr 12   | 🗑️     |

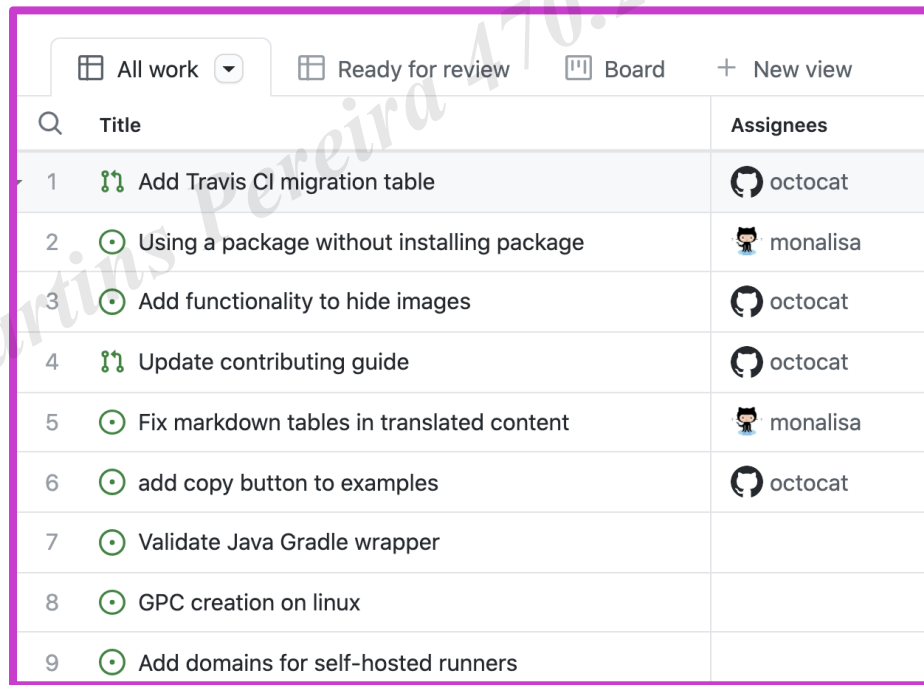
At the bottom, there are 'Save changes' and 'Reset' buttons.

# Personalizar exibições de projetos
















Organize as informações alterando o layout, agrupando, classificando e filtrando seu trabalho.

## Use paleta de comandos:

- Troca de layout: tabela.
- Mostrar: marcos.
- Classificar por: Destinatários, asc.
- Agrupar por: Status.
- Campo coluna por: Status.
- Filtrar por Status.
- Excluir exibição.



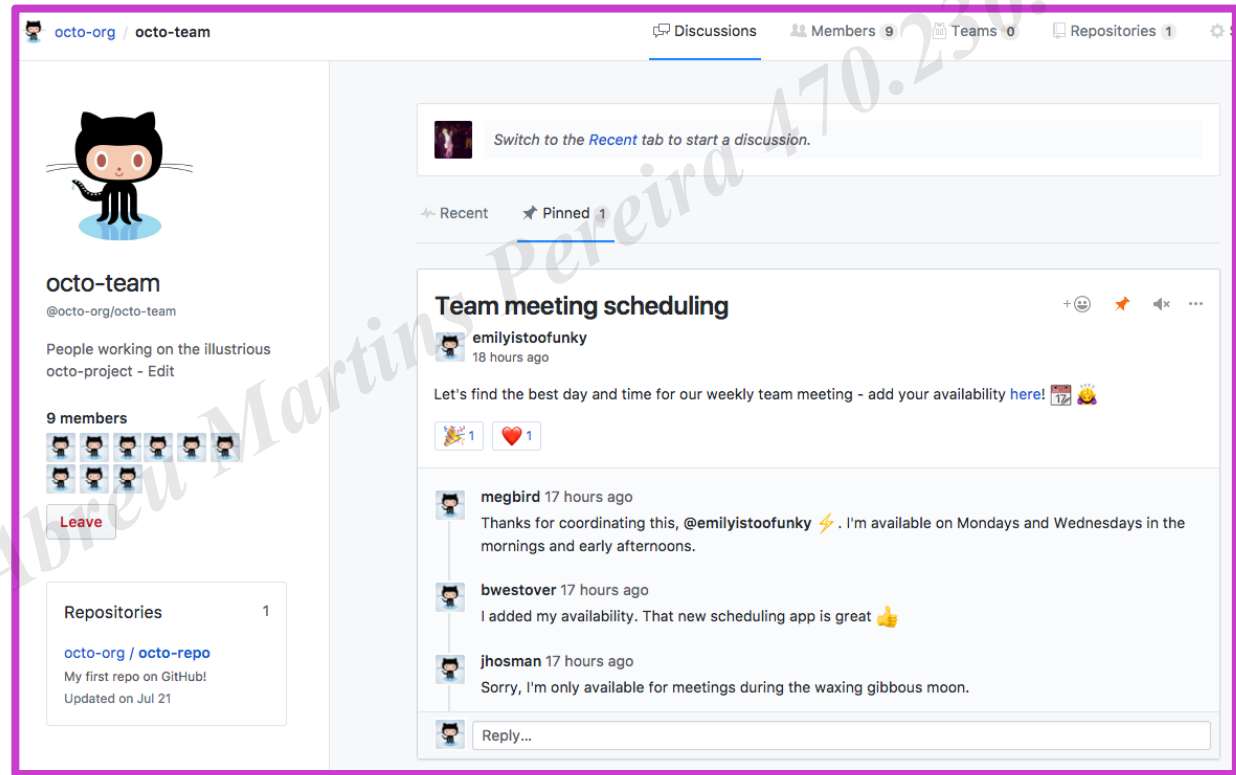
The screenshot shows a project management interface with a table of tasks. The table has two columns: 'Title' and 'Assignees'. The tasks are numbered 1 through 9. The assignees are either 'octocat' or 'monalisa'. The interface includes a search bar and a 'New view' button.

|   | Title   | Assignees  |
|---|---|--|
| 1 |  Add Travis CI migration table              |  octocat  |
| 2 |  Using a package without installing package |  monalisa |
| 3 |  Add functionality to hide images           |  octocat  |
| 4 |  Update contributing guide                  |  octocat  |
| 5 |  Fix markdown tables in translated content  |  monalisa |
| 6 |  add copy button to examples                |  octocat  |
| 7 |  Validate Java Gradle wrapper               |  |
| 8 |  GPC creation on linux                      |  |
| 9 |  Add domains for self-hosted runners        |  |

# Colaborar usando discussões em equipe

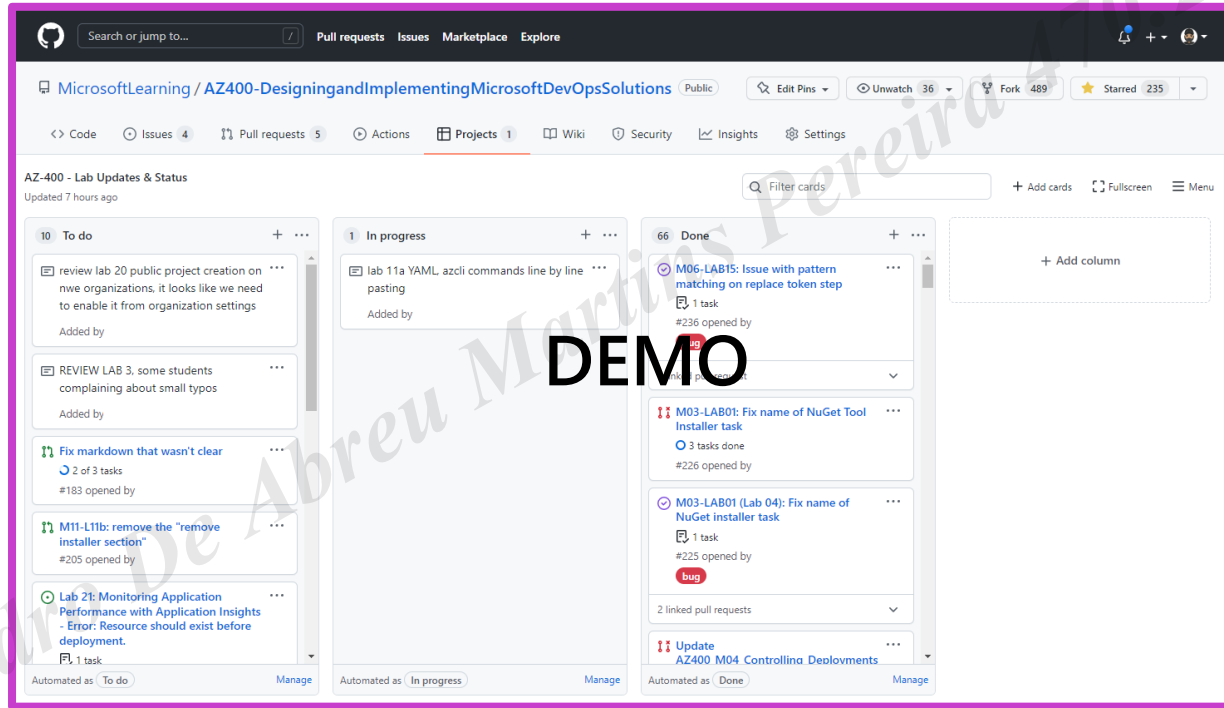
As discussões do GitHub podem ajudar a fazer sua equipe planejar em conjunto, atualizar umas às outras ou falar sobre qualquer tópico.

Conversas que abrangem projetos ou repositórios (problemas, solicitações de pull etc.).





# Demonstração: quadros de projeto do GitHub e projetos



# Introdução ao controle do código-fonte

# Explorar práticas fundamentais de DevOps

Práticas fundamentais e as fases da evolução do DevOps: fases 0 a 2

|         | Definição de práticas e práticas associadas   | Práticas que colaboram para o sucesso   |
|---------|---|---|
| Fase 0  | <ul style="list-style-type: none"><li>• O monitoramento e o alerta são configuráveis pela equipe que opera o serviço</li><li>• Padrões de implantação para criação de aplicativos ou serviços são reutilizados</li><li>• Padrões de teste para criação de aplicativos ou serviços são reutilizados</li><li>• Equipes colaboram com aprimoramentos para as ferramentas fornecidas por outras equipes</li><li>• Configurações são gerenciadas por uma ferramenta de gerenciamento de configuração</li></ul> |   |
| Fase 1  | <ul style="list-style-type: none"><li>• Equipes de desenvolvimento de aplicativo usam o <b>controle de versão</b></li><li>• Equipes fazem a implantação em um conjunto padrão de sistemas operacionais</li></ul>  | <ul style="list-style-type: none"><li>• Criação com base em um conjunto padrão de tecnologias</li><li>• Colocar as configurações de aplicativo em <b>controle de versão</b></li><li>• Teste das alterações da infraestrutura antes de serem implantadas em produção</li><li>• <b>Código-fonte</b> está disponível para outras equipes</li></ul> |
| Etapa 2 | <ul style="list-style-type: none"><li>• Criação com base em um conjunto padrão de tecnologias</li><li>• Equipes implantam em um único sistema operacional padrão</li></ul>  | <ul style="list-style-type: none"><li>• Padrões de implantação para criação de aplicativos e serviços são reutilizados</li><li>• Rearquitetura de aplicativos com base nas necessidades de negócios</li><li>• Colocação das configurações do sistema em <b>controle de versão</b></li></ul>   |

\* As práticas que definem cada fase são destacadas com a fonte em negrito

# Explorar práticas fundamentais de DevOps (continuação)

Práticas fundamentais e as fases da evolução do DevOps: fases três a cinco

|         | Definição de práticas e práticas associadas  | Práticas que colaboram para o sucesso  |
|---------|--|--|
| Etapa 3 | <ul style="list-style-type: none"><li>• Indivíduos podem trabalhar sem aprovação manual de fora da equipe</li><li>• Padrões de implantação para criação de aplicativos e serviços são reutilizados</li><li>• Alterações na infraestrutura são testadas antes de serem implantadas em produção</li></ul>                        | <ul style="list-style-type: none"><li>• Indivíduos podem fazer alterações sem tempos de espera significativos</li><li>• Alterações em serviços podem ser feitas durante o horário comercial</li><li>• Análises pós-incidente e os resultados são compartilhados</li><li>• Equipes criam com base em um conjunto padrão de tecnologias</li><li>• Equipes usam a integração contínua</li><li>• As equipes de infraestrutura usam o <b>controle de versão</b></li></ul> |
| Etapa 4 | <ul style="list-style-type: none"><li>• Configurações do sistema são automatizadas</li><li>• Provisionamento é automatizado</li><li>• As configurações de aplicativo estão em <b>controle de versão</b></li><li>• As equipes de infraestrutura usam o <b>controle de versão</b></li></ul>                                      | <ul style="list-style-type: none"><li>• Configurações de política de segurança são automatizadas</li><li>• Recursos disponibilizados por meio do autoatendimento</li></ul>   |
| Fase 5  | <ul style="list-style-type: none"><li>• Respostas a incidentes são automatizadas</li><li>• Recursos disponibilizados por meio do autoatendimento</li><li>• Rearquitetura de aplicativos com base nas necessidades de negócios</li><li>• Equipes de segurança estão envolvidas no design e implantação de tecnologias</li></ul> | <ul style="list-style-type: none"><li>• Configurações de política de segurança são automatizadas</li><li>• Desenvolvedores de aplicativos implantam ambientes de teste por conta própria</li><li>• Métricas de sucesso para os projetos ficam visíveis</li><li>• Provisionamento é automatizado</li></ul>  |

\* As práticas que definem cada fase são destacadas com a fonte em negrito

# O que é controle do código-fonte?

1

Um sistema de controle do código-fonte permite que os desenvolvedores colaborem no código e acompanhem as alterações. Use o controle de versão para salvar seu trabalho e coordenar as alterações de código em sua equipe.

2

O sistema de controle de versão salva um instantâneo de seus arquivos (histórico) para que você possa revisar e até mesmo reverter para qualquer versão do código com facilidade.

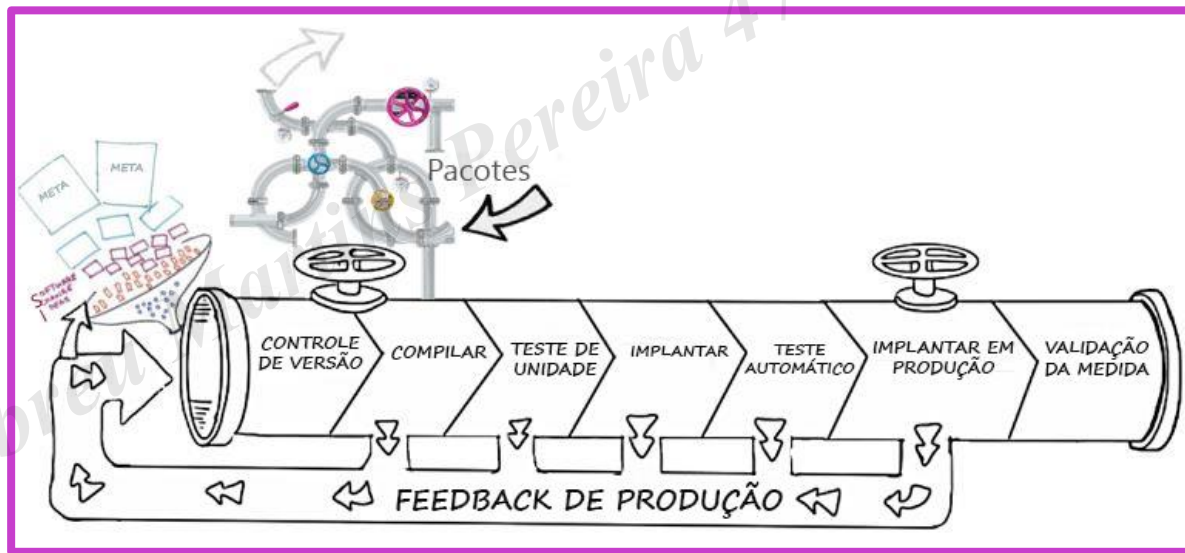
3

O controle do código-fonte protege o código-fonte não só contra catástrofes, mas também contra uma degradação casual provocada por erros humanos e consequências não intencionais.



# Explorar os benefícios do controle do código-fonte

- Criar fluxos de trabalho
- Trabalhar com versões
- Colaboração
- Mantém o histórico de alterações
- Automatizar tarefas



# Explorar melhores práticas para o controle do código-fonte

- 1 Faça alterações pequenas;
- 2 Não envie arquivos pessoais;
- 3 Atualize com frequência e logo antes do envio para evitar conflitos de mesclagem;
- 4 Verifique a alteração do código antes de enviá-la para um repositório. Garanta que a compilação e os testes sejam aprovados;
- 5 Preste muita atenção nas mensagens commit, pois elas informarão por que uma alteração foi feita;
- 6 Vincular alterações feitas no código a itens de trabalho;
- 7 Não importe seu histórico ou suas preferências, seja um bom parceiro de equipe e siga as convenções e os fluxos de trabalho acordados.

# Sistemas de controle do código-fonte

# Entender o controle do código-fonte centralizado



Centralizado

## Pontos fortes

- Escalonamento facilitado para bases de código muito grandes
- Controle de permissões granular
- Permite o monitoramento do uso
- Permitir o bloqueio de arquivo exclusivo

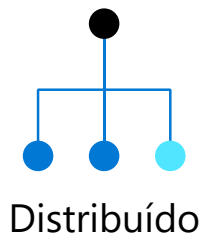
## Melhor usado para

- Bases de código grandes integradas
- Auditoria e controle de acesso até o nível de arquivo
- Tipos de arquivo difíceis de mesclar

Há uma única cópia central do seu projeto, e os programadores confirmam as alterações na cópia central.

Os sistemas de controle do código-fonte centralizados comuns são TFVC, CVS, Subversion (ou SVN) e Perforce.

# Entender o controle do código-fonte distribuído



## Pontos fortes

- Suporte entre plataformas
- Um modelo de revisão de código compatível com código aberto por meio de solicitações de pull
- Suporte offline completo
- Histórico portátil
- Uma base de usuários motivada e em crescimento

## Melhor usado para

- Tamanho menor (em bytes) e bases de código modulares
- Evolução por meio de código aberto
- Equipes altamente distribuídas
- Equipes trabalhando em multiplataformas
- Bases de código Greenfield

Cada desenvolvedor clona uma cópia de um repositório e tem o histórico completo do projeto.

Os sistemas comuns de controle de código-fonte distribuídos são Mercurial, Git e Bazaar.



# Explorar o Git e o Controle de Versão do Team Foundation

## GIT:

Sistema de controle de código-fonte distribuído. Cada desenvolvedor tem uma cópia do repositório do código-fonte em seu sistema de desenvolvimento.

## TFVC:

Sistema centralizado de controle de código-fonte.

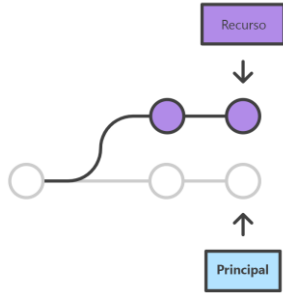
Os membros da equipe tem apenas uma versão de cada arquivo em seus computadores de desenvolvimento.

No modelo **workspaces do servidor**, antes de fazer alterações, os membros da equipe fazem check-out publicamente dos arquivos.

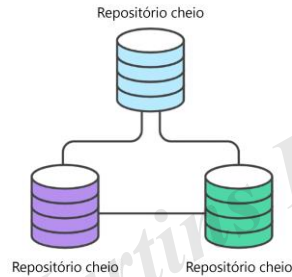
No modelo **workspaces locais**, cada membro da equipe usa uma cópia da versão mais recente da base de código para trabalhar off-line conforme necessário.

# Examinar e escolher o Git

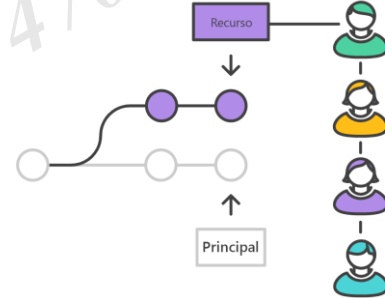
## Branches de recursos



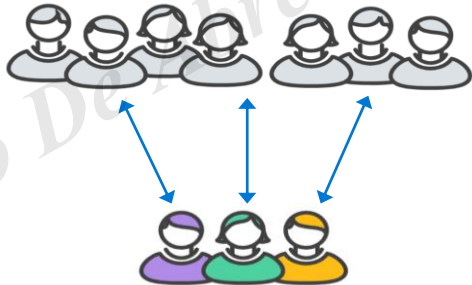
## Desenvolvimento distribuído



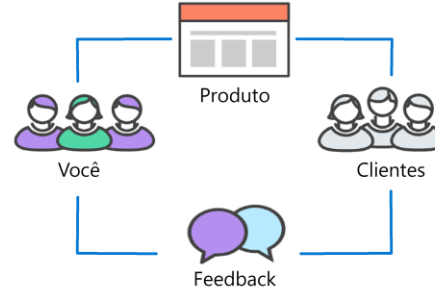
## Solicitações pull



## Comunidade



## Ciclo de lançamento



# Entender as objeções ao uso do Git

- 1 **Substituindo o histórico** – Se usado incorretamente, pode levar a conflitos;
- 2 **Arquivos grandes** – O Git funciona melhor com repositórios pequenos e que não contenham arquivos grandes (ou binários). Considere usar suporte do Git LFS;
- 3 **Curva de aprendizado** – Algum treinamento e instrução serão necessários. Os desenvolvedores já existentes talvez precisem ser retreinados;
- 4 **Discussão** – Quais objeções você tem?

**DEMO**

# Azure Repos e GitHub



# Introdução ao Azure Repos

## 1 Conjunto de ferramentas de controle de versão para gerenciar seu código:

- Totalmente integrado com outros recursos do Azure DevOps
- Pode se conectar a partir de ambientes de desenvolvimento comuns
- Gerenciamento baseado em políticas para proteger branches
- Oferece dois estilos de controle de versão:
  - **Git:** controle de versão distribuído
  - **TFVC (controle de versão do Team Foundation):** controle de versão centralizado

# Introdução ao GitHub

## 1 Maior comunidade de código aberto

---

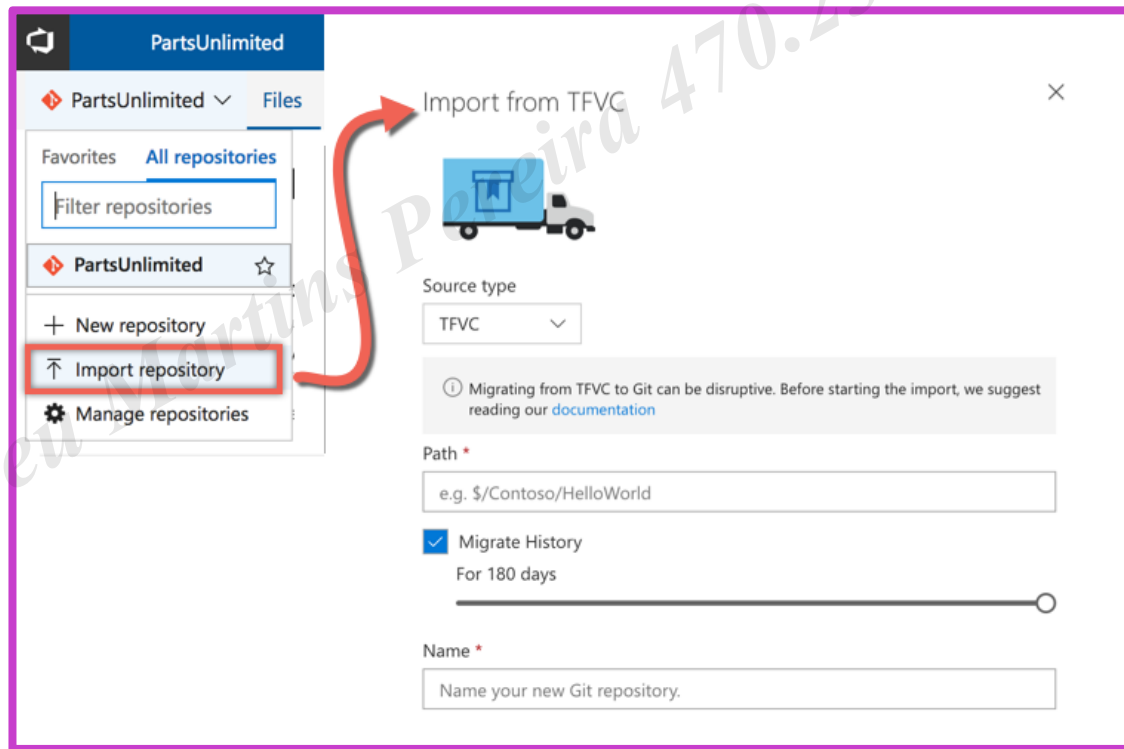
## 2 Recursos do GitHub:

- Automatize do código para a nuvem
- Proteção do software em conjunto
- Análise de código perfeita
- Código e documentação em um só lugar
- Coordenada
- Gerenciar equipes

# Migrar do TFVC para o Git

Importação de ramificação única

- Sincronização completa (git-tfs)



# Migrar do TFVC para o Git (continuação)

1

## Migração da dica:

- Somente a versão mais recente do código
  - O histórico permanece no servidor antigo
- 

2

## Migração com histórico:

Tenta imitar o histórico no Git

---

3

## Recomendado migrar a dica porque:

- O histórico é armazenado de modo diferente: – O TFVC armazena conjuntos de alterações, o Git armazena instantâneos do repositório
- As branches são armazenadas de modo diferente – Pastas de branches TFVC, GIT branches para todo o repositório

# Desenvolver online com GitHub Codespaces

Ambiente de desenvolvimento com base em nuvem hospedado pelo GitHub

1

Evita problemas com hardwares/softwarees antigos

---

2

Altamente portátil

---

3

Proteção contra proliferação de propriedade intelectual

4

Com base no Visual Studio Code

---

5

Trabalhar usando PCs, tablets e Chromebooks

---

6

Conectar-se ao Codespaces no Visual Studio Code

# Referências de livros

- The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win
- Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations
- DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations
- Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation
- Lean Software Development: An Agile Toolkit



# Referências de sites e recursos online

- <https://www.devopsinstitute.com/blog/>
- <https://git-scm.com/>
- <https://learn.microsoft.com/en-us/azure/devops/?view=azure-devops>
- <https://docs.github.com/pt>

# OBRIGADO!

[linkedin.com/in/rogeriorodrigues](https://www.linkedin.com/in/rogeriorodrigues)