Certified Tester
Quality in DevOps (CT-QDO)
Syllabus

V1.0

International Software Testing Qualifications Board

BSTQB®
CT-QDO 1.0

Direitos Autorais

Aviso de direitos autorais © International Software Testing Qualifications Board (doravante denominado ISTQB®). ISTQB® é uma marca registrada do International Software Testing Qualifications Board.

Direitos autorais © 2026, aos autores Szilárd Széll (product owner), Kari Kakkonen, Rik Marselis, Katja Obring e Yaron Tsubery, e aos autores das provas de exemplo Tal Pe'er, Daniel Poľan, Tomas Tumasonis e Aneta Derková.

Todos os direitos reservados. Os autores, por meio deste, transferem os direitos autorais para o ISTQB®. Os autores (como atuais detentores dos direitos autorais) e o ISTQB® (como futuro detentor dos direitos autorais) concordaram com as seguintes condições de uso:

Trechos deste documento, para uso não comercial, podem ser copiados desde que a fonte seja citada. Qualquer Provedor de Treinamento Credenciado poderá utilizar este syllabus como base para um curso de treinamento, desde que os autores e o ISTQB® sejam citados como fonte e detentores dos direitos autorais do syllabus e desde que qualquer propaganda desse curso de treinamento só mencione o syllabus após o recebimento do credenciamento oficial dos materiais de treinamento por parte de um Conselho Membro reconhecido pelo ISTQB®.

Qualquer pessoa física ou grupo de pessoas pode utilizar este syllabus como base para artigos e livros, desde que os autores e o ISTQB® sejam citados como fonte e detentores dos direitos autorais do syllabus.

Qualquer outro uso deste syllabus é proibido sem a prévia obtenção da aprovação por escrito do ISTQB®.

Qualquer Conselho Membro reconhecido pelo ISTQB® pode traduzir este syllabus, desde que reproduza o Aviso de Direitos Autorais acima mencionado na versão traduzida do syllabus.

Histórico de revisões

Versão	Data	Observações
v1.0	17/04/2026	Lançamento geral

Histórico de revisões do BSTQB

Data	Seção	Observações
22/07/2026	1.1.2	Correção do artigo do termo ‘muro da Copnfusão’

© International Software Testing Qualifications Board

Índice

© International Software Testing Qualifications Board

© International Software Testing Qualifications Board

© International Software Testing Qualifications Board

Agradecimentos

Este documento foi formalmente divulgado pela Assembleia Geral do ISTQB® em <data

Foi elaborado por uma equipe do International Software Testing Qualifications Board: Vipul Kocher (Presidente do Grupo de Trabalho de Tecnologias e Abordagens Especializadas), Tamás Horváth (Vice-Presidente do Grupo de Trabalho de Tecnologias e Abordagens Especializadas), Szilárd Széll (product owner), Kari Kakkonen, Rik Marselis, Katja Obring, Tal Pe'er, Daniel Poľan, Yaron Tsubery, Tomas Tumasonis, Aneta Derková

As seguintes pessoas, em ordem alfabética, são os autores deste syllabus: Kari Kakkonen, Rik Marselis, Katja Obring, Szilárd Széll, Yaron Tsubery.

As seguintes pessoas, em ordem alfabética, foram colaboradores importantes deste syllabus: Mantas Aniulis, Aneta Derkova, Matthias Hamburg, Tamas Horvath, Vipul Kocher, Gary Mogyorodi, Tal Pe'er, Daniel Poľan, Patrick Quilter, Jean-Francois Riverin, Tomas Tumasonis e Galit Zucker.

A equipe agradece às seguintes organizações por contribuírem para o conteúdo deste syllabus: ASTQB, AT*SQA, DevOps United.

A equipe agradece a Gary Mogyorodi pela revisão técnica, bem como à equipe de revisão e aos Conselhos de Membros por suas sugestões e contribuições.

As seguintes pessoas participaram da revisão, dos comentários e da votação deste syllabus: Mariam Abdellatif, May Abu-Sbeit, Gergely Agnecz, Amer Alatrash, Laura Albert, Chris van Bael, Jürgen Beniermann, Earl Burba, Tamás Csákó, Wim Decoutere, Walter Eder, Raymond Gillespie, Ole Christian Hansen, Josephine Irungu, Mattijs Kemmink, John Kurowski, Jędrzej Kwapiński, Anders Larsen, Roberta Lippelli, Sebastian Malyska, Vincenzo Marrazzo, Gary Mogyorodi, Frank Neirynck, Ingvar Nordström, Giorgio Pisani, Andrew Pollner, Nishan Portoyan, Meile Posthuma, Patrick Quilter, Miroslav Renda, Piet de Roo, Adam Roman, Jan Sabak, Adam Scierski, Márton Siska, Péter Sótér, Benjamin Timmermans, Giancarlo Tomasig, Stephanie Ulrich, Linda Vreeswijk, Dominik Weber, Geoffrey Wemans, Claude Zhang

Aliança especial

O TMMi® e o ISTQB® firmaram uma aliança para promover ainda mais, em conjunto, a profissão de Testador de Software. Durante a elaboração deste documento, o comitê técnico do TMMi® e o grupo de trabalho do ISTQB® responsável pelo syllabus do Certificado de Testador de Qualidade em DevOps trabalharam em conjunto. O TMMi® é um modelo de melhoria de testes. Ele oferece uma abordagem de melhoria predefinida com prioridades baseadas na estrutura do modelo TMMi®. No documento "TMMi® no mundo do DevOps" (E. v. Veenendaal, 2025), o foco está em fornecer uma interpretação das metas de melhoria do TMMi® para aqueles que trabalham no contexto do DevOps.

Ele não fornece uma descrição detalhada das boas práticas de engenharia de DevOps. O syllabus do curso "Certified Tester Quality in DevOps" do ISTQB® é um documento baseado em conteúdo. Seu objetivo é fornecer uma descrição detalhada das boas práticas de engenharia em DevOps, por exemplo, atividades de teste. Os dois documentos são, portanto, altamente complementares. O TMMi® identifica quais processos precisam de melhoria, enquanto o syllabus do curso "Certified Tester Quality in DevOps" do ISTQB® descreve as melhores práticas de engenharia para implementação.

O BSTQB® agradece aos voluntários de seu grupo de trabalho de traduções (WG Traduções) pelo empenho e esforço na tradução deste material: Gabriel Almeida Barrio Nuevo, George Fialkovitz, Irene Nagase, Osmar Higashi, Paula de Oliveira, Stênio Viveiros, Thiago Cesar Andrade.

O BSTQB® também agradece aos ISTQB® Accredited Training Providers no Brasil que foram convidados a contribuir na revisão da tradução. No momento deste trabalho os seguintes provedores estavam

© International Software Testing Qualifications Board

credenciados: Conecteseaqui (conecteseaqui.com.br), Exseed (www.exseed.com.br), Iterasys (www.iterasys.com.br).

O BSTQB® também agradece às empresas brasileiras credenciadas no ISTQB® Partner Program no Brasil que foram convidados a contribuir na revisão da tradução. No momento deste trabalho as seguintes empresas estavam credenciadas: Keeggo (www.keeggo.com), Deltapoint (www.deltapoint.com.br), TIS (tis.ao), Venturus (www.venturus.org.br).

© International Software Testing Qualifications Board

0 Introdução

0.1 Objetivo deste syllabus

Este syllabus constitui a base para o syllabus da Qualificação Internacional em Testes de Software para o Testador Certificado em Qualidade em DevOps (CT-QDO). O ISTQB® disponibiliza este syllabus da seguinte forma:

1. Aos conselhos membros, para tradução para seus idiomas locais e para credenciamento de prestadores de treinamento. Os conselhos membros podem adaptar o syllabus às suas necessidades específicas de idioma e modificar as referências para adequá-lo às suas publicações locais.
2. Aos órgãos de certificação, para elaborar questões de exame em seu idioma local, adaptadas aos objetivos de aprendizagem deste syllabus.
3. Aos provedores de treinamento, para produzir material didático e determinar métodos de ensino adequados.
4. Aos candidatos à certificação, para se prepararem para o exame de certificação (seja como parte de um curso de treinamento ou de forma independente).
5. À comunidade internacional de engenharia de software e sistemas: promover a profissão de testes de software e sistemas e servir de base para livros e artigos.

0.2 O Testador certificado em Quality in DevOps

A certificação ISTQB® Certified Tester Quality in DevOps (CT-QDO) apoia as pessoas envolvidas no desenvolvimento de software baseado em DevOps na aplicação de atividades adequadas de qualidade e testes para atingir o nível de qualidade adequado para suas soluções.

A certificação CT-QDO foi concebida para profissionais que atuam nas áreas de qualidade e testes em um ambiente DevOps. A ideia é proporcionar aos especialistas em DevOps a oportunidade de adquirir uma certificação internacionalmente reconhecida na área e estabelecer as vantagens específicas dos especialistas certificados em qualidade de DevOps por meio de competências específicas em qualidade e metodologia de testes em DevOps.

O DevOps tornou-se a forma predominante de desenvolvimento de software moderno em muitos domínios. Ele amplia os conceitos do desenvolvimento ágil de software — autonomia, entregas frequentes e foco no feedback precoce do cliente — para alcançar a qualidade adequada do software. A garantia da qualidade (QA) e os testes têm grande destaque na cultura DevOps. Em um ambiente de DevOps, a engenharia de qualidade (QE), incluindo QA e testes, ocorre durante todas as fases do ciclo de vida de desenvolvimento de software (SDLC). O DevOps abrange conceitos como entrega contínua (CD) e lançamento sob demanda, que os testadores de software devem compreender para desenvolver práticas de teste eficazes e de qualidade.

0.3 Trajetória profissional para testadores

O esquema ISTQB® apoia profissionais de testes em todas as etapas de suas carreiras, oferecendo conhecimento tanto em amplitude quanto em profundidade.

© International Software Testing Qualifications Board

A certificação Certified Tester Quality in DevOps baseia-se na qualificação de um Certified Tester Foundation Level do ISTQB® (ISTQB® CTFL, v4.0). Os syllabi relacionados ao ISTQB® Agile e os syllabi relacionados ao ISTQB® Test Automation são úteis para a compreensão deste syllabus. Enquanto o syllabus do ISTQB® Certified Tester Foundation Level fornece o conhecimento e as competências básicas em testes de software, os syllabi relacionados ao ISTQB® Agile ampliam o syllabus do Certified Tester Foundation Level e explicam como os testes são realizados em uma equipe ágil, enquanto os syllabi relacionados ao ISTQB® Test Automation explicam como a automação de testes é construída e utilizada para os testes.

Como este syllabus se concentra na cultura DevOps, ele amplia o portfólio de produtos do ISTQB® na direção do DevOps, à medida que o desenvolvimento de software segue nessa direção, trazendo benefícios como tempo de lançamento no mercado mais rápido, clientes mais satisfeitos e melhor qualidade.

Pessoas que obtiverem a certificação ISTQB® CT-QDO também podem se interessar por outras certificações de Especialista do ISTQB® , especialmente a ISTQB® Certified Tester Foundation Level Agile Tester (ISTQB® CT-AT, v1.1), a ISTQB® Certified Tester Agile Technical Tester (ISTQB® CT-ATT, v1.1) e a ISTQB® Certified Tester Agile Test Leadership at Scale (ISTQB® CT-ATLaS, v2.0), bem como o syllabus da ISTQB® Certified Tester Test Automation Strategy (ISTQB® CT-TAS, v1.0) e, ainda, o ISTQB® Certified Tester Test Automation Engineer (ISTQB® CTAL-TAE, v2.0).

0.4 Resultados de Negócios

Esta seção lista os resultados de negócio esperados de um candidato que tenha obtido a certificação CT-QDO.

Uma pessoa certificada como CT-QDO pode:

Codificar	Descrição
QDO-BO1	Explicar como a garantia da qualidade é apoiada pelos conceitos de DevOps e como contribui para eles
QDO-BO2	Compreender os conceitos de implementação do DevOps em uma organização
QDO-BO3	Contribuir para todas as etapas do fluxo de valor a fim de auxiliar na engenharia da qualidade
QDO-BO4	Contribuir para a implementação do ciclo de DevOps
QDO-BO5	Descrever como a automação apoia a garantia da qualidade no DevOps
QDO-BO6	Implementar a automação de testes em equipes de DevOps
QDO-BO7	Implementar testes manuais em equipes de DevOps
QDO-BO8	Selecionar ferramentas que ofereçam suporte ao DevOps dentro da organização
QDO-BO9	Compreender tecnologias e práticas para garantir a qualidade em DevOps

© International Software Testing Qualifications Board

0.5 Objetivos de aprendizagem, objetivos práticos e nível cognitivo de conhecimento

Os objetivos de aprendizagem e os objetivos práticos apoiam os resultados de negócios e são utilizados para elaborar os exames de Certificação de Testador de Qualidade em DevOps (CT-QDO).

Em geral, todo o conteúdo deste syllabus está sujeito a avaliação, exceto a Introdução, os Objetivos Práticos e os Anexos. As questões do exame avaliarão o conhecimento de palavras-chave no nível K2 (veja abaixo) ou dos objetivos de aprendizagem no respectivo nível de conhecimento. Os objetivos de aprendizagem específicos e seus níveis de conhecimento são apresentados no início de cada capítulo e classificados da seguinte forma:

- K1: Recordar
- K2: Compreender
- K3: Aplicar

Mais detalhes e exemplos de objetivos de aprendizagem são fornecidos na *Seção 7*.

Para todos os termos listados como palavras-chave logo abaixo dos títulos dos capítulos, o nome e a definição corretos do Glossário do ISTQB® (ISTQB, 2025) devem ser compreendidos (K2), mesmo que não sejam explicitamente mencionados no objetivo de aprendizagem.

Os Objetivos Práticos específicos são apresentados no início de cada capítulo. O nível de um OP é classificado da seguinte forma:

- H0: Pode incluir uma demonstração ao vivo de um exercício ou um vídeo gravado. Como isso não é realizado pelo aluno, não se trata estritamente de um exercício.
- H1: Exercício guiado. Os participantes seguem uma sequência de etapas realizadas pelo instrutor.
- H2: Exercício com dicas. O aluno recebe um exercício com dicas para resolvê-lo dentro do tempo estipulado.
- H3: Exercícios não guiados e sem dicas.

0.6 O Exame de Certificação de Testador de Qualidade em DevOps

O exame de certificação CT-QDO será baseado neste syllabus. As respostas às questões do exame podem exigir o uso de material baseado em mais de uma seção deste syllabus. Todas as seções do syllabus são objeto de avaliação, exceto a Introdução, os Objetivos Práticos e os Anexos. Normas e livros estão incluídos como referências, mas seu conteúdo não é objeto de avaliação, além do que está resumido no próprio syllabus a partir dessas normas e livros.

Consulte o documento “Exam Structures and Rules” (ISTQB® Exam Structures and Rules, 2025. Version 1.2) para obter mais detalhes.

Os critérios de entrada para realizar o exame ISTQB® Certified Tester Quality in DevOps são que os candidatos tenham

- Uma certificação ISTQB® Certified Testador Nível Básico
- Ter interesse em testes de software e DevOps

Os candidatos também são fortemente incentivados a:

© International Software Testing Qualifications Board

- Ter pelo menos uma experiência mínima em desenvolvimento de software ou em testes de software, como seis meses de experiência como testador de sistemas ou de teste de aceite de usuário, ou como desenvolvedor de software.
- Fazer um curso com credenciamento de acordo com os padrões do ISTQB® (por um dos conselhos membros reconhecidos pelo ISTQB).

0.7 Credenciamento

Um Conselho Membro do ISTQB® pode credenciar provedores de treinamento e proprietários de materiais didáticos cujo conteúdo siga este syllabus. Os provedores de treinamento devem obter as diretrizes de credenciamento junto ao Conselho Membro ou ao órgão responsável pelo credenciamento. Um curso credenciado é reconhecido como estando em conformidade com este syllabus e está autorizado a incluir um exame do ISTQB® como parte do curso.

As diretrizes de credenciamento para este syllabus seguem as Diretrizes Gerais de Credenciamento (ISTQB® Generic Accreditation Guidelines, 2024. Version 2.4) publicadas pelo Processes Management and Compliance Working Group do ISTQB®

0.8 Tratamento de Normas

Organizações internacionais de padronização, como o IEEE e a ISO, publicaram normas relacionadas a características de qualidade e testes de software. Essas normas são citadas neste syllabus. O objetivo dessas referências é fornecer um framework (como nas referências à ISO 25010 relativas às características de qualidade) ou servir como fonte de informações adicionais, caso o leitor deseje.

Observe que os syllabi do ISTQB® utilizam os documentos normativos como referência. Os documentos normativos não fazem parte do conteúdo da prova. Consulte o Capítulo 10 para obter mais informações sobre normas.

0.9 Nível de Detalhamento

O nível de detalhamento deste syllabus permite a realização de cursos e exames consistentemente padronizados internacionalmente. Para atingir esse objetivo, o syllabus consiste em:

- Objetivos instrucionais gerais que descrevem a intenção do syllabus do curso “Certified Tester Quality in DevOps”
- Uma lista de termos que os alunos devem ser capazes de recordar
- Objetivos de aprendizagem para cada área de conhecimento, descrevendo os resultados cognitivos a serem alcançados
- Uma descrição dos conceitos-chave, incluindo referências a fontes como literatura ou normas reconhecidas

O syllabus não é uma descrição completa da área de conhecimento de testes de software; ele reflete o nível de detalhamento a ser abordado nos cursos de treinamento do CT-QDO.

O syllabus utiliza a terminologia (o nome e o significado) dos termos empregados em QA e testes, de acordo com o Glossário do ISTQB®.

© International Software Testing Qualifications Board

0.10 Como este syllabus está organizado

Existem quatro capítulos com conteúdo passível de avaliação. O título de nível superior de cada capítulo especifica a duração do capítulo; a duração não é fornecida abaixo do nível do capítulo. Para cursos de treinamento com credenciamento, o syllabus exige um mínimo de 20,2 horas de instrução distribuídas por pelo menos três dias. O tempo mínimo de treinamento distribuído pelos quatro capítulos é o seguinte:

Capítulo 1: 140 minutos, Fundamentos de DevOps

- Explicar como a garantia da qualidade é apoiada pelos conceitos de DevOps e contribui para eles
- Compreender os conceitos de implementação de DevOps em uma organização

Capítulo 2: 360 minutos, garantia da qualidade e testes em DevOps

- Contribuir para todas as etapas do fluxo de valor para auxiliar na engenharia de qualidade da qualidade
- Contribuir para a implementação do ciclo de DevOps

Capítulo 3: 510 minutos, Tarefas automatizadas e manuais no DevOps

- Descrever como a automação apoia a garantia da qualidade no DevOps
- Implementar a automação de testes em equipes de DevOps
- Implementar testes manuais em equipes de DevOps

Capítulo 4: 200 minutos, Ferramentas e práticas no DevOps

- Selecionar ferramentas que dão suporte ao DevOps dentro da organização
- Compreender tecnologias e práticas para apoiar a qualidade no DevOps

© International Software Testing Qualifications Board

1 Fundamentos do DevOps - 140 minutos

Palavras-chave

garantia da qualidade, testes

Palavras-chave específicas de qualidade em DevOps

CALMS, porcentagem de falhas de mudança, lead-time da mudança, equipe multifuncional de DevOps, frequência de implantação, DevOps, tempo de recuperação de implantação que falhou, ciclo de feedback, fluxo, acordo de nível de serviço, indicador de nível de serviço, objetivo de nível de serviço, engenharia de confiabilidade do site, fluxo de valor, muro da confusão

Objetivos de aprendizagem do Capítulo 1:

1.1 Explicar como a garantia da qualidade é apoiada pelos conceitos de DevOps e como contribui para eles

QDO-1.1.1 (K1) Recordar os conceitos-chave do DevOps

QDO-1.1.2 (K1) Recordar os conceitos de “muro da confusão” em DevOps

QDO-1.1.3 (K2) Explicar as métricas DORA de performance na entrega e performance operacional

QDO-1.1.4 (K2) Diferenciar os elementos do CALMS em termos de garantia da qualidade

QDO-1.1.5 (K2) Explicar como a garantia da qualidade apoia as três vertentes do DevOps

QDO-1.1.6 (K2) Explicar os benefícios, riscos e armadilhas do DevOps

1.2 Compreender os conceitos de implementação do DevOps em uma organização

QDO-1.2.1 (K2) Distinguir as funções da equipe de DevOps

QDO-1.2.2 (K2) Explicar o conceito de engenharia de confiabilidade do site (SRE) relacionado ao DevOps

QDO-1.2.3 (K2) Distinguir os diferentes padrões e antipadrões da equipe de DevOps

Objetivos práticos:

1.2 Implementação de DevOps em uma organização

QDO-1.2 (H0) Resumir como os princípios de DevOps se manifestam em um estudo de caso

© International Software Testing Qualifications Board

1.1 Garantia da Qualidade (QA) no DevOps

DevOps é a combinação de filosofias culturais, práticas e ferramentas que aumentam a capacidade de uma organização de entregar aplicativos e serviços em alta velocidade, evoluindo e aprimorando produtos em um ritmo mais rápido do que as organizações que utilizam modelos de desenvolvimento sequenciais e processos de gerenciamento de infraestrutura (Amazon:2024).

O DevOps ganhou destaque pela primeira vez em 2008, quando Patrick Debois (Debois:2011) convidou pessoas com ideias semelhantes para uma conferência a fim de discutir como a entrega de software e infraestrutura poderia utilizar o desenvolvimento ágil de software. O termo DevOps surgiu após a Conferência Velocity de 2009 ter introduzido conceitos de entrega rápida, frequente e bem-sucedida (Debois:2011).

Este capítulo discute os conceitos de DevOps e como a engenharia de qualidade (QE), incluindo garantia da qualidade (QA) e testes, se relaciona com eles. QA refere-se a todas as atividades focadas em proporcionar confiança de que os requisitos de qualidade serão atendidos, ou seja, ao longo de todo o ciclo de vida de desenvolvimento de software (SDLC). QE refere-se a todas as atividades relacionadas à qualidade que contribuem para a criação de software que atenda às necessidades do cliente. O DevOps vê o desenvolvimento de software como um fluxo de valor que orienta o software desde o conceito até o lançamento no mercado (ISTQB CT-ATLAS). Este capítulo também explica as equipes de DevOps, ou seja, qualquer equipe que aplique práticas de DevOps.

1.1.1 Conceitos-chave do DevOps

DevOps significa “Desenvolvimento e Operações”. Refere-se a pessoas que trabalham juntas para construir, entregar e executar software. No âmbito do DevOps, o processo de desenvolvimento de software deve, além da escrita de código, considerar também outros aspectos do desenvolvimento, como análise, integração contínua (CI), testes funcionais, testes não funcionais — especialmente de segurança —, implantação, entrega, lançamento, gerenciamento de infraestrutura, manutenção e monitoramento da produção. A medição do sucesso do desenvolvimento de software deve ser feita com base nos resultados da entrega e na qualidade do software utilizado.

DevOps é uma cultura que promove a colaboração, a comunicação e a melhoria contínua, em vez de ser um framework, um padrão, uma função, uma equipe separada, uma ferramenta ou uma meta (DASA, 2024). O DevOps defende fortemente a automação e o monitoramento em todas as fases da entrega de software. Isso geralmente se dá na forma de pipelines de integração contínua/entrega contínua (CI/CD). O DevOps promove autonomia, liderança, colaboração e inovação, ao mesmo tempo em que impulsiona o sucesso dos negócios por meio de ciclos de entrega mais curtos, maior frequência de implantação e lançamentos confiáveis e de alta qualidade que atendem às necessidades dos clientes.

As práticas de DevOps baseiam-se nos seguintes princípios (DASA, 2019):

1. Ação centrada no cliente: Ter ciclos de feedback curtos com clientes reais.
2. Criar com o fim em mente: Focar explicitamente na construção de produtos funcionais entregues aos clientes.
3. Responsabilidade de ponta a ponta: Criar equipes de DevOps que projetem, construam, testem e executem software.
4. Equipes autônomas multifuncionais: Crie equipes de DevOps com uma variedade de habilidades, incluindo testes, para que sejam autônomas.
5. Melhoria contínua: Adote uma cultura de aprimoramento diário do produto, do processo e das habilidades das pessoas para se adaptar com base no feedback da experimentação.

© International Software Testing Qualifications Board

6. Automatize tudo o que puder: busque um pipeline de CI/CD automatizado que entregue com ciclos curtos.

O DevOps é frequentemente representado como um ciclo infinito, com etapas manuais e automatizadas. O ciclo típico de DevOps inclui várias etapas que podem ser agrupadas como descoberta contínua, CI, CD, implantação contínua e lançamento sob demanda. Ele é descrito em Seção 2.2, e existem múltiplas variações do ciclo de DevOps.

1.1.2 Quebrando o Muro da confusão

O muro da confusão é uma metáfora para a separação entre as equipes de desenvolvimento e operações em modelos de desenvolvimento sequenciais. Tradicionalmente, o QA e os testes ficam do lado da equipe de desenvolvimento da barreira.

Enquanto a equipe de desenvolvimento busca alterar o software continuamente para lançar novos recursos, a equipe de operações prefere estabilidade, minimizando assim o número de alterações na produção. Isso resulta em silos de objetivos conflitantes, falta de comunicação, processos manuais e ambientes inconsistentes. Em resumo, existem diferentes mentalidades (ou seja, motivação e objetivos), processos e ferramentas. As diferenças são mais acentuadas quanto mais isoladas as equipes estão. No DevOps, o muro da confusão é derrubada pela integração das equipes para que trabalhem juntas, melhorando a comunicação e aumentando a colaboração. É necessária liderança para melhorar ou eliminar processos, atividades e práticas ineficientes (por exemplo, evitar uma cultura de culpa ao enfatizar a responsabilidade compartilhada) (DASA, 2024).

O controle de qualidade e os testes devem ser realizados ao longo de todo o processo, tanto no Dev quanto no Ops, tornando os testes um fator-chave para derrubar a barreira.

A adoção do DevOps pode derrubar as barreiras entre várias equipes isoladas (por exemplo, segurança, análise de negócios ou usabilidade).

1.1.3 Métricas de Performance da DORA

O programa DevOps Research & Assessment (DORA) publica o relatório anual State of DevOps. A pesquisa (Dora, 2024) apresenta quatro métricas de performance de entrega que oferecem uma maneira eficaz de realizar a medição do processo de entrega de software.

Duas métricas estão relacionadas à produtividade (ou seja, a velocidade de realização de alterações):

- Lead time das mudanças: o tempo decorrido entre o commit do código e a implantação bem-sucedida em produção. Reflete a eficiência do processo de entrega e tende a ser curto para equipes de alto desempenho.
- Frequência de implantação: a frequência com que as alterações de software são implantadas na produção. Ela reflete o quão ágil e responsivo é o processo de entrega e tende a ser alta para equipes de alta performance.

Duas métricas estão relacionadas à estabilidade (ou seja, a qualidade das alterações entregues e a capacidade da equipe de DevOps de corrigir falhas):

- Porcentagem de falhas na mudança: a porcentagem de implantações que causam falhas em produção, exigindo hotfixes ou rollbacks. Ela mostra o grau de confiabilidade do processo de entrega e tende a ser baixa em equipes de alta performance.
- Tempo de recuperação da implantação que falhou: O tempo necessário para se recuperar de uma implantação que falhou. Mostra o quão resiliente e ágil a organização é e tende a ser curto para equipes de alta performance.

© International Software Testing Qualifications Board

Equipes de DevOps de alto desempenho tendem a ter bons resultados nas quatro métricas. Para isso, elas precisam de boas práticas de QE, incluindo automação de testes, automação de CI/CD, colaboração, abordagens de testar primeiro, uma abordagem de equipe completa (ISTQB® CTFL), estratégias de implantação bem selecionadas e monitoramento contínuo. Quanto mais rápido o fluxo de valor, melhor.

O DORA adicionou a confiabilidade como um indicador de performance operacional. De acordo com o DORA, a confiabilidade consiste em métricas ou características de qualidade, como disponibilidade, latência, eficiência de performance e escalabilidade. As equipes de DevOps alcançam resultados de performance de entrega melhores (por exemplo, menos esgotamento da equipe) ao se concentrarem na confiabilidade (Dora, 2022). As quatro métricas de performance de entrega estão relacionadas ao processo, enquanto a confiabilidade está relacionada ao software em uso.

QA e testes contribuem diretamente para uma melhor confiabilidade, especialmente por meio de abordagens de testar primeiro, testes não funcionais, engenharia de caos (ver Seção 4.2.6) e busca pela qualidade (ver Seção 3.3.4).

1.1.4 Elementos do CALMS no contexto do QA

CALMS significa cultura, automação, lean, medição e compartilhamento (Kim, Humble, et al., 2016). É um framework que avalia a capacidade da organização de adotar processos DevOps e medir o sucesso durante uma transformação DevOps.

A cultura reflete que a mera mudança de processos e tecnologia não é suficiente para promover uma mudança duradoura. Uma cultura colaborativa e voltada para a resolução de problemas é essencial. A cultura DevOps é uma extensão do desenvolvimento ágil de software, onde equipes ágeis têm autonomia para entregar valor aos clientes enquanto buscam a colaboração necessária com outras equipes. Essa autonomia inclui testes e uma abordagem de equipe completa em relação à responsabilidade pela qualidade, de modo que os testes não ficam fora da equipe DevOps (Crispin et al., 2008).

Automação refere-se a automatizar o máximo possível no pipeline de CI/CD. Isso inclui compilação, implantação, provisionamento, automação de processos, análise estática e automação de testes, ou seja, “tudo como código”. A automação nos pipelines de CI/CD permite que a equipe de DevOps implemente pequenas alterações continuamente, mantendo a boa qualidade e acelerando o ciclo de feedback.

Lean (Humble, Molesky, et al., 2020) significa otimizar processos e eliminar desperdícios (por exemplo, etapas desnecessárias no processo) para que o processo seja o mais eficiente possível. A melhoria contínua faz parte do lean e se aplica também às atividades de teste, que não devem se tornar gargalos para o fluxo de trabalho. Portanto, repense quando e como os testes são realizados no processo.

A medição fornece dados para decisões sobre como o software é usado e como o SDLC funciona. A medição envolve o uso de métricas acionáveis sobre qualidade e eficiência para melhorar os produtos e aprimorar continuamente o SDLC.

O compartilhamento diz respeito à cultura aberta de discutir desafios e sucessos do desenvolvimento de software nas equipes de DevOps. Trata-se também de compartilhar boas práticas e ideias para o aprendizado e a melhoria contínua. A QA e os testes desempenham um papel vital na geração de informações a serem compartilhadas (por exemplo, feedback de testes automatizados relacionados à funcionalidade e à testabilidade).

Cada elemento do CALMS reforça o papel do QA e dos testes em garantir que a qualidade seja incorporada em todas as fases do SDLC, permitindo a entrega mais rápida de software de alta qualidade e, ao mesmo tempo, reduzindo riscos.

Existem outras versões do CALMS (por exemplo, o CALMR no SAFe (Scaled Agile Framework, 2023a), onde R significa recuperação).

© International Software Testing Qualifications Board

1.1.5 Os Três Caminhos do DevOps

Para entender como o QA se encaixa nas práticas de desenvolvimento de software, é essencial examinar os princípios do DevOps. O DevOps representa um movimento cultural e técnico que visa unificar desenvolvimento e operações, mas seus princípios centrais vão muito além desses dois domínios. Esses princípios, conhecidos como os três caminhos do DevOps, fornecem um framework que ajuda as organizações a transformar seu processo de entrega de software, mantendo padrões de alta qualidade. Compreender esses princípios explica como as práticas de QA evoluem e se integram no contexto do DevOps.

Em “The Phoenix Project” (Kim, Behr, et al., 2018), os três caminhos do DevOps são as seguintes:

1. O primeiro caminho: fluxo
   Este princípio se concentra no fluxo contínuo do trabalho, do desenvolvimento às operações, garantindo que o trabalho avance de forma rápida e eficiente ao longo do processo. O fluxo enfatiza a importância de minimizar os momentos de transferência, reduzir os tamanhos dos lotes e eliminar gargalos para criar um processo simplificado e previsível.
2. O segundo caminho: ciclo de feedback
   Isso enfatiza a criação de ciclos de feedback rápidos e contínuos. Isso permite a detecção e resolução rápidas de problemas, evitando que eles se agravem. Também incentiva uma cultura em que todos possam aprender com os erros e melhorar continuamente o produto e o processo de entrega.
3. O terceiro caminho: aprendizagem e experimentação contínuas
   Este princípio destaca a importância de fomentar uma cultura de melhoria contínua e experimentação. Ele incentiva a inovação, o aceite calculado de riscos e o aprendizado tanto com os sucessos quanto com as falhas. Este princípio sustenta a ideia de que o aprendizado contínuo é essencial para se adaptar às mudanças e melhorar o sistema como um todo.

O QA é parte integrante dos três caminhos do DevOps:

Fluxo:
O QA permite uma progressão suave do código por meio de abordagens de testar primeiro.

Feedback:
O QA cria ciclos rápidos de feedback por meio da automação de testes contínua, ambientes de pré-produção padronizados e monitoramento para coletar dados de uso no mundo real para melhorias.

Aprendizado contínuo:
O QA contribui para a inovação por meio de testes direcionados de novos recursos, eventos de qualidade organizados, como a “caça da qualidade”, e testes de regressão abrangentes.

1.1.6 Benefícios, riscos e armadilhas do DevOps

O DevOps traz oportunidades, mas as organizações devem estar atentas aos riscos e armadilhas ao implementar o DevOps.

Benefícios

O DevOps pode trazer benefícios significativos para as organizações (Dora, 2024). Entre eles estão:

- Capacitar as equipes de DevOps para avaliar a qualidade em todas as fases de desenvolvimento, implantação e produção
- Capturar feedback de forma precoce e frequente para melhorar tanto o software quanto o SDLC

© International Software Testing Qualifications Board

- Entrega dentro do prazo com custos mais baixos devido a software de maior qualidade e processos mais eficientes de eficiência
- Responder mais rapidamente às necessidades em constante mudança
- Recuperar-se de falhas na produção mais rapidamente
- Melhorar o tempo de lançamento no mercado por meio de processos e comunicação com eficiência e efetividade
- Agilizar o dimensionamento do software com o aumento do número de usuários graças à infraestrutura como código (IaC) (ver Seção 4.2.3)
- Melhorar a colaboração nas equipes de DevOps ao longo do SDLC
- Reduzir o estresse dos funcionários, permitindo lançamentos mais frequentes de pequenos incrementos de funcionalidades
- Reduzir problemas com fornecedores e terceiros ao aumentar a colaboração e a transparência em todas as atividades da equipe de DevOps
- Melhorar o retorno sobre o investimento na automação de testes (Humble, Farley, 2010)

Riscos e armadilhas

Embora os benefícios do DevOps sejam bastante óbvios, muitas coisas podem dar errado. As equipes de DevOps devem antecipar esses riscos para que eles não se tornem armadilhas que façam a transformação do DevOps falhar. Riscos e armadilhas típicos incluem:

- Falta de apoio da gerência
- Remover ou substituir a área de operações em vez de transformá-la
- Substituir o desenvolvimento ágil de software em vez de ampliá-lo
- Acreditar que existe apenas uma maneira de fazer DevOps
- Usar DevOps apenas como um cargo
- Criar uma equipe de DevOps, mas ignorar a necessidade de mudança em outras equipes e na organização
- Reduzir o número de pessoas aumentando a automação em vez de tornar as pessoas mais eficientes
- Introduzir apenas ferramentas de pipeline de CI/CD e esquecer a mudança cultural do DevOps
- Implementar um pipeline de CI/CD com pouca ou nenhuma automação de testes
- Usar apenas testes automatizados, quando testes manuais seriam benéficos

1.2 Implementando DevOps em uma organização

O DevOps pode ser implementado de várias maneiras em uma organização. Independentemente de como o DevOps é implementado, há sempre demanda por QE, incluindo testes. Os membros da equipe precisam compreender esses conceitos para trabalhar com eles de forma eficaz.

Esta seção discute as funções típicas em equipes de DevOps, incluindo variações da função de testador, Engenharia de Confiabilidade do Site (SRE) em DevOps e topologias, padrões e antipadrões de equipe que

© International Software Testing Qualifications Board

afetam o sucesso do DevOps. A experiência dos membros da equipe relacionada ao ambiente de desenvolvimento também é considerada. Isso é conhecido como experiência do desenvolvedor (DevEx).

(H0) Resumir como os princípios do DevOps são visíveis em um estudo de caso

Para este objetivo prático, o exercício de nível H0 auxilia os alunos na preparação para os objetivos práticos subsequentes, fornecendo o contexto de um estudo de caso. A organização imaginária descrita no estudo de caso deve ser considerada apenas como uma organização DevOps parcial, de modo que os exercícios baseados nos objetivos práticos subsequentes possam implementar melhorias nas formas de trabalho dentro dessa organização.

1.2.1 Funções em uma equipe DevOps

Uma função é um conjunto de responsabilidades e tarefas, e não necessariamente um cargo. As funções centrais em uma equipe DevOps multifuncional são analista de negócios, desenvolvedor, testador e engenheiro de operações, frequentemente apoiados por um product owner e um scrum master. Todos os membros da equipe, em conjunto, são responsáveis pela QE.

Os membros de uma equipe DevOps multifuncional podem selecionar qualquer tarefa que esteja alinhada com seus conhecimentos e habilidades. Isso significa que uma pessoa pode assumir várias funções ao longo do tempo e pode até mesmo desempenhar múltiplas funções em paralelo. Os membros da equipe podem colaborar em pares em uma tarefa. Essa abordagem otimiza o fluxo de trabalho na equipe DevOps.

Uma equipe multifuncional de DevOps possui todo o conhecimento e as habilidades necessárias para realizar qualquer tarefa comum à equipe de DevOps. Nenhuma parte do conhecimento e das habilidades deve ficar restrita a apenas um membro da equipe. Dessa forma, uma equipe multifuncional de DevOps pode continuar funcionando mesmo quando um dos membros estiver temporariamente indisponível.

No entanto, algumas tarefas especializadas podem estar além das habilidades disponíveis nas equipes de DevOps (por exemplo, a configuração e a manutenção da plataforma). Por esse motivo, as organizações podem optar por ter equipes de plataforma específicas que forneçam ferramentas de autoatendimento e infraestrutura confiável que abstraiam tarefas operacionais complexas, permitindo uma entrega de software mais rápida e consistente pelas equipes de DevOps. Dessa forma, a equipe de plataforma reduz a carga operacional sobre as equipes de DevOps e possibilita implantações mais rápidas, seguras e com alta confiabilidade.

A qualidade dos produtos de trabalho da equipe de DevOps é mantida por meio da colaboração, do aprendizado contínuo e da melhoria. Qualquer produto de trabalho é tratado por pelo menos dois membros da equipe. O processo de revisão garante isso (por exemplo, com pull requests (PRs)) (ver Seção 2.1.4). Se o produto de trabalho não for aceito, o revisor fornece feedback para que o autor possa melhorá-lo, por conta própria ou em colaboração com outros membros da equipe. Qualquer membro da equipe pode ser um autor, um colaborador ou um revisor de produtos de trabalho.

Uma equipe DevOps multifuncional, composta por um grupo diversificado de pessoas qualificadas e motivadas que colaboram e assumem responsabilidades em conjunto, é uma maneira efetiva de criar e manter software. Essa é a base da cultura DevOps.

1.2.2 Engenharia de Confiabilidade do Site (SRE)

SRE é um conjunto de práticas que aplica aspectos da engenharia de software às operações e à infraestrutura de TI. O SRE tem origem na organização Google (Beyer et al., 2016). O SRE concentra-se em garantir que um sistema cumpra continuamente os requisitos e necessidades funcionais e não funcionais. Seu principal objetivo é manter a confiabilidade do serviço, equilibrando a velocidade de lançamento com a estabilidade, geralmente por meio de automação, monitoramento e planejamento

© International Software Testing Qualifications Board

proativo de capacidade. Ao criar ciclos de feedback automatizados e usar o feedback para fazer melhorias, o SRE apoia a equipe para garantir que os sistemas permaneçam dentro dos níveis de serviço definidos, minimizando interrupções e otimizando a eficiência, preenchendo a lacuna entre desenvolvimento e operações em ambientes de produção.

É comum que um cliente e um fornecedor definam os níveis de serviço exigidos para garantir a performance do sistema de software. Um nível de serviço descreve um aspecto específico de um sistema de software que é relevante para as partes interessadas. As partes envolvidas formalizam seus acordos em acordos de nível de serviço (SLA). Os SLAs são compromissos vinculativos que ajudam a priorizar os esforços de desenvolvimento e operações e dão suporte na concepção de casos de teste relevantes, garantindo que o serviço esteja consistentemente alinhado com as metas de negócios e as necessidades dos usuários.

Os objetivos de nível de serviço (SLOs) e os indicadores de nível de serviço (SLIs) são conceitos importantes em SRE. Eles são necessários para cumprir esses SLAs. Os SLOs descrevem o que um sistema de software deve cumprir, e os SLIs são usados para realizar a medição de se os SLOs foram atingidos. Um SLO inclui o valor-alvo ou intervalo de valores para um nível de serviço.

Um SLI é uma medição quantitativa bem definida dos aspectos do nível de serviço que são fornecidos. Um ou mais SLIs são usados para determinar se os SLOs foram cumpridos, garantindo que o serviço seja prestado no nível acordado. Os SLIs são medidos durante as fases de desenvolvimento por meio de testes. Durante o processo de operações, os SLIs são medidos por meio de monitoramento (por exemplo, com observabilidade e telemetria). A execução automatizada de testes de regressão é importante para a medição dos SLIs em vários estágios do pipeline de CI/CD.

A criação e implementação de sistemas e serviços em SRE seguem três fases distintas:

- Planejamento e projeto da infraestrutura necessária: isso abrange características de qualidade, como confiabilidade, escalabilidade e segurança.
- Implementação e implantação da infraestrutura: isso inclui configurar a infraestrutura, implantar o sistema ou serviço e garantir que o sistema funcione de acordo com o nível de serviço acordado.
- Operação, manutenção e otimização contínuas de um sistema ou serviço para mantê-lo com alta confiabilidade e eficiência ao longo do tempo.

1.2.3 Padrões e antipadrões de equipes DevOps

Para atingir o objetivo principal de entregar valor ao cliente, diferentes organizações podem adotar diferentes estruturas de equipes DevOps.

A estrutura da equipe da organização depende de:

- Da complexidade do software, pois maior complexidade favorece a formação de silos
- Liderança técnica e sua capacidade de criar uma meta comum para as equipes de Dev e Ops
- A disposição para mudar para um modelo DevOps no departamento de operações de TI
- As habilidades necessárias para liderar a transformação para o modelo DevOps

Uma integração adequada do DevOps nas topologias de equipe (Matthew Skelton, 2023) facilitará a colaboração e a entrega de valor, enquanto antipadrões podem prejudicar esses objetivos ao destacar problemas organizacionais.

Topologias de equipes DevOps

A seguir, discutimos três modelos. Eles são independentes e não estão listados por ordem de preferência.

© International Software Testing Qualifications Board

- Colaboração entre Dev e Ops:
  Há uma cooperação harmoniosa entre as equipes de Dev e Ops. Ambas as equipes aprendem o básico sobre como a outra equipe funciona. Isso requer um objetivo compartilhado claramente definido e uma mudança cultural organizacional significativa.
- Responsabilidades de operações totalmente compartilhadas: O pessoal de operações está totalmente integrado às equipes de Dev. Organizações como Facebook e Netflix popularizaram esse modelo no início da década de 2010. Nesse modelo, todos estão focados no mesmo objetivo. As equipes que desenvolvem um recurso também são responsáveis por manter suas operações.
- Equipe SRE:
  Organizações como o Google definem a colaboração entre as equipes de Dev e Ops como a equipe de SRE que colabora com as equipes de Dev. Para uma explicação detalhada sobre as equipes de SRE, consulte Seção 1.2.2. As equipes de Dev precisam fornecer evidências de testes de que seus produtos atendem aos padrões estabelecidos pela equipe de SRE, sem se envolverem diretamente nas questões operacionais. Esse padrão pode facilmente se transformar em antipadrões e silos.

Antipadrões da equipe DevOps

- Silos de Dev e Ops:
  Essa estrutura de equipe tem uma mentalidade de “muralha”. “Feito” é tipicamente definido como recurso completo e não lançado em produção.
- Dev não precisa de Ops:
  Novas equipes de desenvolvimento em ambientes que utilizam serviços em nuvem podem pensar que não precisam da equipe de operações.
- DevOps como uma equipe de ferramentas:
  Uma equipe separada de DevOps implementa as ferramentas necessárias para o DevOps. Isso pode beneficiar as ferramentas do departamento de Desenvolvimento, mas fica aquém do impacto que verdadeiras equipes de DevOps poderiam ter devido à falta de integração nas práticas de trabalho diárias das equipes de desenvolvimento, onde muitos benefícios das formas de trabalho do DevOps são encontrados.

© International Software Testing Qualifications Board

2 Garantia da Qualidade (QA) e Teste em DevOps - 360 minutos

Palavras-chave

Testes A/B, desenvolvimento orientado por teste de aceite, desenvolvimento ágil de software, testes automatizados, desenvolvimento orientado pelo comportamento, lançamento canário, integração contínua, testes contínuos, testes holísticos, engenharia de qualidade, política de qualidade, modelo de desenvolvimento sequencial, análise estática, testes estáticos, política de testes

Palavras-chave específicas de qualidade em DevOps

implantação blue-green, pipeline de CI/CD, entrega contínua, implantação contínua, descoberta contínua, aprendizagem e experimentação contínuas, monitoramento contínuo, lançamento discreto, pull request, lançamento sob demanda, fluxo de valor

Objetivos de aprendizagem do Capítulo 2:

2.1 Contribuir para todas as etapas do fluxo de valor para auxiliar na engenharia de qualidade da qualidade

QDO-2.1.1 (K3) Implementar atividades de garantia da qualidade em um contexto de DevOps

QDO-2.1.2 (K2) Comparar os objetivos do teste que apoiam o DevOps com modelos de desenvolvimento sequencial e desenvolvimento ágil de software

QDO-2.1.3 (K2) Explicar os testes contínuos no contexto do DevOps

QDO-2.1.4 (K2) Explicar como as solicitações pull apoiam a qualidade da aplicação

2.2 Contribuir para a implementação do ciclo DevOps

QDO-2.2.1 (K2) Explicar a garantia da qualidade e os testes na descoberta contínua e suas práticas de teste

QDO-2.2.2 (K2) Explicar a garantia da qualidade e os testes na integração contínua e suas práticas

QDO-2.2.3 (K2) Explicar a garantia da qualidade e os testes na entrega contínua e suas práticas

QDO-2.2.4 (K2) Explicar a garantia da qualidade e os testes na implantação contínua e suas práticas

QDO-2.2.5 (K2) Explicar a garantia da qualidade e os testes no lançamento sob demanda e suas práticas

QDO-2.2.6 (K3) Aplicar conhecimentos de garantia da qualidade e testes para implementar um pipeline de CI/CD

Objetivos práticos:

2.1 Contribuição para o fluxo de valor com o objetivo de auxiliar na engenharia da qualidade (QE)

QDO-2.1 (H2) Contribuir para o fluxo de valor a fim de auxiliar na engenharia da qualidade

2.2 Implementação do ciclo de DevOps

QDO-2.2 (H2) Contribuir com a implementação do DevOps Loop

© International Software Testing Qualifications Board

2.1 Contribuição para o fluxo de valor para auxiliar na engenharia de qualidade (QE)

QA e testes são fundamentais no contexto DevOps, ao longo de todo o fluxo de valor. Um fluxo de valor é uma série de processos, atividades e fluxos de trabalho interconectados que entregam valor aos clientes por meio de um produto, serviço ou experiência. Embora o QA e os testes sejam realizados de maneira diferente no desenvolvimento ágil de software e nos modelos de desenvolvimento sequencial, ambos têm um grande impacto na qualidade do software. De uma perspectiva geral, alcançar a qualidade adequada envolve mais do que apenas QA e testes. A QE abrange o QA e os testes; trata-se de todas as atividades que contribuem para a criação do software de que o cliente necessita.

Esta seção aborda as atividades e práticas no contexto do DevOps, especificamente, com um exercício prático. Os objetivos da QE no DevOps são comparados ao desenvolvimento ágil de software e aos modelos de desenvolvimento sequencial. A seção final descreve a importância de um processo de pull request (PR).

(H2) Contribuir para o fluxo de valor para auxiliar na engenharia de qualidade

Para este objetivo prático, o exercício de nível H2 auxilia os alunos a aprender como criar uma visão geral das atividades de QE específicas do DevOps para um caso específico. Isso inclui quais entregáveis de QA e testes devem ser definidos, com o objetivo de alcançar a qualidade adequada para os três aspectos da qualidade: produtos, processos e pessoas.

2.1.1 QA e Teste no Contexto do DevOps

Os profissionais de DevOps se esforçam para incorporar a qualidade desde o início e implementar uma abordagem de “acertar na primeira vez”. Isso promoverá uma maneira eficiente de trabalhar ao longo do SDLC e implementará efetivamente as necessidades dos stakeholders. Como tal, o DevOps é um aprofundamento da mentalidade Ágil. Para alcançar esse aprofundamento, qualquer equipe de DevOps ou grupo de equipes de DevOps colaboradoras precisa implementar atividades relacionadas à obtenção de qualidade integrada e ao fornecimento de informações sobre o nível de qualidade alcançado. Uma política de qualidade e uma política de teste documentadas darão suporte a isso. As listas a seguir mostram um agrupamento desses princípios, atividades e práticas para os três aspectos da qualidade: produtos, processos e pessoas (Marselis et al., 2020):

Qualidade dos Produtos

A qualidade do produto criada pela(s) equipe(s) de DevOps depende, por exemplo, dos seguintes princípios:

- Alcançar valor de negócio é o principal objetivo do desenvolvimento
- Implementar uma arquitetura que possibilite a produtividade da equipe de DevOps
- Implementar corretamente as características de qualidade funcionais e não funcionais do software (Kim, Humble, et al., 2016) (Systems and software engineering 25010)
- Prestar atenção às atividades de engenharia de qualidade que precisam ser aplicadas durante todas as atividades do ciclo de vida do DevOps, o que inclui atividades de qualidade preventivas, como escrita colaborativa de histórias do usuário e programação em pares, e atividades de qualidade detectivas, como testes estáticos e testes dinâmicos.

© International Software Testing Qualifications Board

Qualidade dos processos

Os processos que visam construir a qualidade adequada consistem, por exemplo, nas seguintes atividades:

- Fornecer monitoramento, controle, relatórios e alertas totalmente integrados relacionados aos indicadores de medição de valor de negócio e nível de qualidade
- Aplicar uma abordagem baseada no risco para estimar os esforços e programar as tarefas e atividades posteriores
- Estabelecer a infraestrutura e as ferramentas necessárias para as atividades de QE integradas a um pipeline de CI/CD
- Implementar práticas de melhoria contínua

Qualidade das Pessoas

As pessoas envolvidas precisam aplicar, como parte da mentalidade Ágil, por exemplo, as seguintes práticas para entregar os produtos de forma adequada:

- Desenvolvimento orientado por hipóteses: (Acreditamos que <UMA determinada ação resultará em <UM objetivo pretendido. Teremos confiança para prosseguir quando <UMA medição exceder uma determinada meta) (Kim, Humble, et al., 2016)
- Um ambiente de trabalho colaborativo
- Foco na melhoria contínua

Para implementar esses princípios, atividades e práticas, uma organização precisa estabelecer diretrizes gerais para todas as equipes de DevOps e stakeholders envolvidos. Os principais produtos de trabalho são a política de qualidade e a política de teste, que podem ser combinadas em um único documento, com uma abordagem de testar primeiro para qualidade integrada e uma mentalidade de “automatizar tudo o que for possível”. A política de qualidade e a política de teste descrevem as funções e responsabilidades das equipes multifuncionais, descrevem a organização de várias equipes e delineiam sua implementação. A implementação das políticas pode basear-se em uma análise de risco de produto, na pirâmide de teste e/ou nos quadrantes de teste Ágeis (ISTQB® CTFL) e resultar em uma estratégia de QE (Marselis et al., 2020). A implementação requer a colaboração dos membros da equipe de DevOps e entre várias equipes de DevOps.

2.1.2 Compare os objetivos do teste em diferentes modelos de SDLC

Os três caminhos do DevOps (ou seja, fluxo, feedback e aprendizagem contínua) (Kim, Humble, et al., 2016) exigem uma abordagem totalmente integrada para QA e testes. A qualidade deve ser incorporada desde o início, e os testes fornecem informações sobre se o nível de qualidade adequado é entregue no momento certo.

Em modelos de desenvolvimento sequenciais, os testes são realizados em níveis de teste separados e conduzidos por equipes de teste especializadas.

No desenvolvimento ágil de software, a equipe ágil realiza testes para confirmar que uma história do usuário está concluída e que os critérios de aceite foram atendidos. Adiar as tarefas de teste para o final de uma iteração pode levar a ineficiências e riscos de qualidade.

O DevOps amplia a mentalidade ágil; o princípio do fluxo estabelece que cada produto de trabalho deve ser concluído e entregue de forma independente, e os testes são parte integrante do desenvolvimento.

Os objetivos dos testes são semelhantes nas três abordagens de teste mencionadas acima, na medida em que fornecem informações sobre a qualidade para gerar confiança, enquanto os objetivos dos testes no

© International Software Testing Qualifications Board

desenvolvimento ágil de software e no DevOps também incluem a necessidade de fornecer feedback antecipado e contínuo.

No DevOps, a execução automatizada de testes é essencial em um pipeline de CI/CD. A automação de testes é essencial para alcançar o fluxo. Em contraste com os modelos de desenvolvimento sequencial, nos quais as equipes podem ter sucesso sem qualquer execução automatizada de testes, e com o desenvolvimento ágil de software, onde os testes automatizados costumam se concentrar em testes de regressão, no DevOps, a automação de testes é aplicada a todos os níveis e tipos de testes. No entanto, mesmo no DevOps, usando pipelines automatizados de CI/CD, ainda haverá alguma necessidade de testes manuais, especialmente testes exploratórios. Os testes manuais são necessários porque alguns aspectos da qualidade são difíceis de avaliar de forma automatizada (por exemplo, usabilidade). E, por fim, os testes manuais atendem às partes interessadas que desejam ver os resultados dos testes por si mesmas antes de se convencerem de que o nível de qualidade do software está correto.

2.1.3 Testes Contínuos

Os testes contínuos integram atividades de teste ao longo de todo o ciclo de vida de desenvolvimento de software (SDLC), de modo que os testes ocorram desde o conceito inicial até a implantação em produção e além, em vez de em uma fase separada.

Princípios fundamentais dos testes contínuos

- Teste cedo e com frequência
  Os testes contínuos enfatizam o início das atividades de teste o mais cedo possível e sua manutenção consistente ao longo do SDLC. Ao verificar e validar o software em todas as fases, as equipes de DevOps avaliam e relatam o nível de qualidade antecipadamente, reduzindo significativamente o custo e o impacto dos defeitos.
- Colaboração multifuncional
  Nos testes contínuos, a qualidade é responsabilidade de todos. A abordagem requer a participação ativa de todos os membros da equipe de DevOps, não apenas dos testadores dedicados. Essa abordagem colaborativa garante perspectivas diversas e avaliações de qualidade abrangentes. Isso é semelhante à “abordagem de equipe completa” (ISTQB® CT-TAS).
- Automação e integração de ferramentas
  Para possibilitar testes contínuos de eficiência, as equipes de DevOps dependem fortemente da automação. Testes automatizados no pipeline de CI/CD fornecem feedback rápido e garantem qualidade consistente. Ferramentas para vários tipos de teste são utilizadas para alcançar uma cobertura abrangente.
- Monitoramento de produção e observabilidade
  Os testes contínuos vão além dos ambientes de pré-produção, estendendo-se à produção. Os testes contínuos enfatizam a importância do monitoramento e da observabilidade em ambientes de produção. As equipes de DevOps utilizam ferramentas especializadas para acompanhar a performance do software em produção, detectar anomalias e coletar dados sobre o comportamento do usuário em tempo real, a fim de estender a fase de testes à produção. Consulte a seção (ver Seção 4.2.7) para obter mais informações sobre monitoramento e telemetria.
- Melhoria adaptativa
  Os testes contínuos promovem o aprimoramento constante das estratégias de teste. As equipes de DevOps coletam continuamente dados e feedback de várias atividades de teste, usando esses insights para adaptar suas estratégias de teste e aprimorar suas abordagens de teste ao longo do tempo.

© International Software Testing Qualifications Board

Exemplo de framework

Um framework que exemplifica os testes contínuos é o teste holístico desenvolvido por Janet Gregory e Lisa Crispin (Janet Gregory, 2023).

Esse framework ajuda as equipes de DevOps a estruturar e aplicar testes contínuos em cenários reais de desenvolvimento, simplificando a adoção e a adaptação às suas necessidades.

2.1.4 Pull Requests

Um PR, às vezes chamado de solicitação de merge (combinação), é um acordo que exige que o autor do código solicite a sua revisão por um ou mais membros da equipe antes que o ele possa ser inserido no ramo principal. Em um PR, um ramo é uma linha de desenvolvimento separada em um sistema de controle de versão (VCS), onde as alterações ocorrem antes de serem integradas à base de código principal.

O conceito de PRs evoluiu a partir de práticas em VCSs. Antes de o armazenamento em nuvem se tornar o padrão de fato para VCSs, os desenvolvedores escreviam novos códigos em suas máquinas locais, e outros desenvolvedores realizavam “pull” para ajudar a determinar sua qualidade.

O feedback deles é incorporado ao produto e contribui para uma qualidade superior de várias maneiras.

- Os PRs apoiam a qualidade no DevOps ao incentivar o desenvolvimento colaborativo e o compartilhamento de conhecimento entre os membros da equipe, levando a melhores práticas de codificação e maior conscientização sobre possíveis problemas.
- Os PRs podem servir como um dos critérios de qualidade para iniciar uma rodada de teste automatizada nos pipelines de CI/CD, conforme discutido na seção 2.2.
- Os PRs ajudam a garantir o cumprimento de padrões de codificação, a adesão a guias de estilo e as melhores práticas por meio de revisões manuais de código apoiadas por análise estática realizada no pipeline de CI/CD.
- Os PRs melhoram a rastreabilidade ao fornecer um histórico claro das alterações, discussões e decisões tomadas sobre recursos ou correções específicas, promovendo a contabilização.
- Ao incentivar pequenas alterações de código gerenciáveis em vez de atualizações grandes e complexas, os PRs limitam a quantidade de alterações e os riscos associados.
- Atuando como um filtro de qualidade, os PRs garantem que apenas as alterações aprovadas cheguem ao ramo principal.
- Os PRs promovem a responsabilização ao fazer com que membros específicos da equipe de DevOps realizem revisões e aprovem as alterações, apoiando o compartilhamento intencional de conhecimento, incluindo revisões por todas as equipes afetadas pelas alterações no código.

2.2 Implementação do Ciclo DevOps

Um fluxo de valor é uma série de atividades que vão desde a captura da ideia até a entrega de valor aos clientes. O desenvolvimento, a entrega e as operações de software não são conjuntos discretos de atividades que ocorrem isoladamente; em vez disso, elas ocorrem em estreita proximidade, especialmente com o desenvolvimento ágil de software e as práticas de trabalho DevOps (DASA, 2019).

Um fluxo de valor é ilustrado como um ciclo infinito, o chamado ciclo DevOps. Embora existam diferentes ilustrações do ciclo DevOps com diferentes sequências de atividades, o grupo principal de atividades é o mesmo. Esta seção discute como QE, QA e testes são considerados nos grupos de atividades do fluxo de

© International Software Testing Qualifications Board

valor, especificamente na descoberta contínua, integração contínua (CI), entrega contínua (CD) e implantação contínua, lançamento sob demanda e monitoramento contínuo.

O QE, incluindo o QA e os testes, ocorre em todas as atividades do ciclo de DevOps, contribuindo para a qualidade do software. Essas atividades podem ocorrer rapidamente ou lentamente, dependendo do número de fatores conhecidos ou desconhecidos, e, às vezes, uma equipe de DevOps precisará fazer uma pausa e voltar a uma ou duas atividades anteriores para repeti-las antes de seguir em frente novamente (por exemplo, para entender se um recurso faz sentido) (Janet Gregory, 2023).

Embora possa haver etapas manuais no fluxo de valor, buscar a automação é comum no DevOps.

O pipeline de CI/CD refere-se à automação relacionada ao código, incluindo relatórios sobre o resultado das atividades. No entanto, ele não inclui todas as atividades do fluxo de valor. Outras ferramentas podem dar suporte a essas atividades, como ferramentas de gerenciamento de tarefas, nas quais a automação de tarefas é possível. Ferramentas de gerenciamento de tarefas e ferramentas de CI/CD costumam estar intimamente ligadas.

As seções 2.2.1-2.2.5 descrevem as atividades do ciclo DevOps. A seção 2.2.6 descreve as etapas para implementar incrementalmente pipelines de CI/CD na organização, abrangendo essas atividades.

(H2) Contribuir para a implementação do ciclo DevOps

Para este objetivo prático, o exercício de nível H2 auxilia os alunos a aprender como projetar e implementar um pipeline de CI/CD, quais etapas e quais recursos de ferramentas devem ser incluídos para representar um ciclo de DevOps.

2.2.1 QA e testes na descoberta contínua

A descoberta contínua é uma prática de descobrir e avaliar frequentemente resultados, necessidades de clientes e stakeholders, ideias inovadoras, tecnologias modernas e outras oportunidades para decidir qual valor desenvolver a seguir. As equipes de DevOps se concentram na criação de recursos que geram valor por meio do envolvimento contínuo com os stakeholders, da aplicação da centralidade no cliente e do design thinking, e da descoberta de suas necessidades (Torres, 2024), (Scaled Agile Framework, 2023b), (Scaled Agile Framework, 2023c).

Normalmente, três atividades ocorrem na descoberta contínua: descobrir, planejar e compreender (Janet Gregory, 2023).

- Descobrir consiste em adicionar novos recursos e alterações ao software para que ele ajude os stakeholders a resolver seus problemas e atender às suas necessidades. Os stakeholders validam essas necessidades e ideias antes de definir os recursos. Isso pode ser alcançado, por exemplo, por meio de entrevistas com os stakeholders ou pelo desenvolvimento de protótipos rápidos de experiência do usuário (UX). Os product owners, apoiados pela equipe de DevOps, identificam os recursos que permitirão à equipe criar, melhorar e entregar as soluções para os stakeholders.
- O Planejamento determina as características de qualidade e identifica os riscos importantes para o recurso, ao mesmo tempo em que detalha os recursos e os divide em histórias do usuário menores. Como o DevOps usa uma abordagem iterativa/incremental, os itens do backlog são identificados e detalhados de forma incremental, com base em ciclos de aprendizagem curtos, em vez de definir tudo antecipadamente.
- A compreensão consiste em obter uma visão compartilhada dos detalhes das histórias do usuário dentro da equipe de DevOps ou entre várias equipes envolvidas.
  Os membros da equipe DevOps aplicam uma abordagem de testar primeiro, utilizando o desenvolvimento orientado por teste de aceite (ATDD) ou o desenvolvimento orientado pelo comportamento (BDD), o que inclui descrever a especificação por exemplo (Adzic, 2011). Esses

© International Software Testing Qualifications Board

cenários de exemplo ajudam os desenvolvedores a compreender e testar a solução. Essa abordagem de testar primeiro é uma atividade de teste colaborativa que apoia a qualidade integrada. A compreensão do item do backlog pelos membros da equipe de DevOps é abordada pelo cumprimento da definição de pronto (DoR) (por exemplo, definir e compreender os critérios de aceite e estimar os itens do backlog) (Alliance, 2026). Uma definição de feito (DoD) orienta a identificação de todos os produtos de trabalho a serem entregues para qualquer item do backlog (Alliance, 2026).

Uma prática recomendada de QA é ter listas de verificação de DoR e DoD em vigor, definidas, acordadas e seguidas pelos membros da equipe de DevOps.

2.2.2 QA e Teste em CI

Para cada alteração incorporada, a CI exige a compilação de todo o software e a execução de um conjunto abrangente de testes automatizados, tanto estáticos quanto dinâmicos, contra ele (Beck, 2000). Violações do estilo de codificação, não atingir metas de cobertura ou ciclos de teste lentos podem resultar em uma compilação com falhas. Se a compilação ou os testes falharem, os membros da equipe interrompem tudo o que estão fazendo e corrigem a compilação imediatamente. Essa abordagem de “parar e corrigir” é um aspecto da gestão da qualidade no lean, conforme discutido em (Humble, Molesky, et al., 2020). Essa correção pode consistir em reverter para a versão anterior e funcional do software.

O objetivo da CI é manter o software em um estado funcional o tempo todo, de modo que esteja potencialmente pronto para lançamento. “Potencialmente pronto para lançamento” é uma afirmação sobre a qualidade do software e não sobre o valor ou a comercialização do software (Larman et al., 2016).

As equipes de DevOps precisam adotar as seguintes práticas para implementar a CI, conforme descrito em (Humble, Farley, 2010):

- Incluir tudo no gerenciamento de configuração (por exemplo, scripts de teste automatizados, scripts de implantação automatizados, dados de teste e configurações de ambiente).
- Os scripts de automação para as etapas de compilação ou de teste no pipeline de CI/CD devem facilitar a investigação em caso de falhas na compilação.
- As equipes de DevOps fazem check-in de pequenas alterações incrementais e aguardam que os testes passem ou seguem uma abordagem de “parar e corrigir” caso algum teste falhe.

Conforme descrito em (Humble, Farley, 2010), o pipeline de CI geralmente consiste em duas partes principais:

- As ações na fase de commit testam se o software funciona no nível técnico. Isso aciona compilações, execução automatizada de testes de baixo nível e análise estática do código. A revisão de código por outros membros da equipe de DevOps geralmente ocorre somente após as ações na fase de commit passarem.
- As ações na fase de aceite automatizada testam se o software funciona, está em conformidade com a especificação e atende às necessidades do usuário. A fase de aceite pode envolver quaisquer tipos ou níveis de teste necessários para a aceite do software. As atividades de teste podem ser executadas em paralelo nesta fase.

Quando várias equipes de DevOps contribuem para o software, geralmente cada equipe tem sua própria parte de CI específica da equipe no pipeline de CI/CD, que está conectada a uma parte de CD comum, no nível organizacional, do pipeline de CI/CD (ver Seção 2.2.3 e Seção 2.2.4).

© International Software Testing Qualifications Board

2.2.3 QA e Teste na CD

A entrega contínua (CD) garante que o software esteja sempre em um estado pronto para implantação, inclusive para ambientes de produção. A decisão de se o software será realmente implantado em produção cabe a uma pessoa, com base nos resultados dos testes e nas avaliações de prontidão (em comparação com a implantação contínua, em que a implantação ocorre automaticamente se todos os testes forem aprovados, sem qualquer intervenção humana, conforme descrito em Seção 2.2.4).

Após as etapas de CI (ver Seção 2.2.2), o software é implantado em ambiente(s) de teste semelhante(s) ao de produção para testes adicionais. Esses testes podem utilizar automação de testes, técnicas de teste baseadas na experiência ou ambas. É comum realizar *smoke tests* após a implantação em um ambiente e antes da execução de quaisquer testes planejados, a fim de verificar se a implantação foi bem-sucedida. Os testes podem ser executados em vários ambientes em paralelo para fornecer feedback rápido, caso os recursos necessários estejam disponíveis.

A qualidade é alcançada por meio de testes e das seguintes práticas de QE:

- Automação da implantação usando ferramentas de automação para garantir consistência, conformidade e fornecer uma trilha de auditoria (ver Seção 4.1)
- Gerenciamento padronizado do ambiente de teste e do gerenciamento de dados de teste (ver Seção 3.1.6)
- IaC para gerenciar a configuração da infraestrutura usando código para garantir consistência e repetibilidade (ver Seção 4.2.3)

A estratégia de lançamento adotada pela equipe de DevOps (ver Seção 4.2.2) fornece diretrizes sobre a frequência e a forma de implantação em produção (ver Seção 2.2.4) e como lançar para os usuários (ver Seção 2.2.5).

2.2.4 QA e testes na implantação contínua

A implantação contínua baseia-se no pipeline de CI/CD e nas práticas de CI e CD. A implantação contínua difere da entrega contínua (CD) porque implanta automaticamente em produção todas as alterações que passaram nos testes automatizados. Não há envolvimento humano nas decisões de implantação. Essa abordagem garante que o software esteja sempre em um estado implantável e que a versão mais recente seja implantada em produção.

A implantação em produção não significa necessariamente que o software seja lançado para os usuários. Consulte Seção 2.2.5 para obter detalhes. No entanto, o procedimento de implantação repetido com frequência e de alta confiabilidade reduz o risco de que a implantação falhe durante o lançamento do software.

A confiança em sempre implantar na produção baseia-se nos seguintes fatores:

- Ter práticas de QE em vigor para incorporar qualidade por meio da descoberta contínua, CI e CD
- Ter um processo de implantação automatizado e controlado, sem envolvimento humano
- Confiar na suíte de testes automatizados no pipeline de CI/CD para atingir níveis de qualidade acordados e aceitáveis
- Aplicar estratégias de implantação e lançamento para controlar como implantar a aplicação e fornecer acesso ao usuário (por exemplo, lançamento discreto) (ver Seção 4.2.2)

© International Software Testing Qualifications Board

- Ter a capacidade de responder rapidamente às consequências indesejadas da implantação (por exemplo, reverter para um estado previamente definido do sistema ou avançar com a implantação de uma versão mais recente do sistema para corrigir defeitos)

Os testes ocorrem em produção antes do lançamento do software para os usuários, a fim de reduzir ainda mais o risco e gerar feedback sobre a nova funcionalidade ou outras características de qualidade (ver Seção 3.3.3).

- Com um lançamento oculto, o novo recurso é executado no ambiente de produção junto com as funcionalidades existentes sem ser exposto aos usuários, permitindo que as equipes de DevOps monitorem a performance e o comportamento, testem e observem o sistema em condições reais.
- Na implantação blue-green, dois ambientes idênticos, “blue” (atual em produção) e “green” (nova versão), são usados para minimizar o tempo de inatividade e o risco durante a implantação. Os testes ocorrem na nova versão (green) enquanto a versão antiga (blue) permanece em produção. Depois que todos os testes passaram, o ambiente green entra em produção, e o ambiente blue fica pronto para a próxima mudança e seus testes, de modo que eles trocam de papéis. Dois ambientes permitem um rápido rollback caso surjam problemas após o lançamento de uma nova versão.
- Um lançamento canário primeiro expõe um pequeno subconjunto de usuários a uma alteração de software para validar a qualidade e, em seguida, implementa-a gradualmente para toda a base de usuários, caso não existam defeitos críticos. O primeiro subconjunto de usuários pode ser uma porcentagem aleatória da base total de usuários, um grupo pré-selecionado de usuários conhecidos (ou seja, testadores-beta), uma área geográfica ou qualquer outro agrupamento.
- O teste A/B é uma abordagem de teste para comparar duas ou mais variações de um recurso, interface ou funcionalidade, apresentando cada variante a diferentes segmentos de usuários e analisando métricas de performance para determinar qual versão alcança o resultado desejado de forma mais eficaz, mantendo essa variante em produção e removendo as outras.

2.2.5 QA e testes no lançamento sob demanda

O lançamento sob demanda é uma prática de DevOps para lançar novos recursos imediatamente ou de forma incremental com base nas necessidades da empresa, dos clientes e de quaisquer outros stakeholders, separando a implantação do lançamento. Quando um novo recurso é implantado na produção, ele permanece oculto dos usuários até que o acesso seja fornecido de forma controlada usando práticas e ferramentas de gerenciamento de recursos (ver Seção 4.2.2 e Seção 4.2.4). Essa abordagem permite que a empresa lance recursos para os usuários no momento mais oportuno do mercado (por exemplo, em uma determinada data antes da temporada de festas). Ela promove a agilidade de negócios, entregando e mantendo o maior valor para os usuários quando eles precisam e pelo tempo que precisarem.

A separação de elementos da arquitetura para que possam ser lançados separadamente reduz o risco de interrupções no serviço durante a implantação e o lançamento. Essa técnica identifica blocos de construção específicos da arquitetura, cada um dos quais pode ser lançado de forma independente (por exemplo, bancos de dados de back-end de código aberto seguem ciclos de lançamento principais, enquanto microsserviços de front-end são atualizados várias vezes por iteração). Os diferentes blocos de construção podem ter estratégias e frequências de lançamento diferentes.

O lançamento pode envolver atividades não relacionadas a software (por exemplo, atualização de manuais do usuário, materiais de treinamento, notas de lançamento, procedimentos operacionais, contratos legais ou atividades de marketing e vendas).

Após o lançamento do software, as operações se concentram em atividades críticas que preservam a estabilidade e a segurança do software (ver Seção 1.2.2). Práticas de monitoramento contínuo são utilizadas para observar o software em seu ambiente, para fornecer informações sobre se a qualidade do

© International Software Testing Qualifications Board

software atende aos objetivos acordados e para poder tomar medidas corretivas após ser alertado sobre a degradação da qualidade (ver Seção 4.2.7). As atividades de gerenciamento de serviços de TI (ITSM) também apoiam a qualidade (por exemplo, estabelecimento de uma central de atendimento para fornecer suporte a clientes e usuários).

A medição da observabilidade mede o comportamento do sistema em relação aos seus requisitos funcionais e não funcionais (ver Seção 4.2.7). Além disso, são coletadas informações de outros canais (por exemplo, feedback de clientes recebido pelo service desk ou pelas redes sociais). Os dados coletados ajudam a compreender como a mudança lançada proporciona benefícios de negócio (por exemplo, aumento no número de assinaturas ou melhoria nas vendas).

A organização pode usar métricas para aprender e identificar o próximo valor potencial a ser entregue. Isso pode ser um aprimoramento do recurso mais recente, uma melhoria não funcional ou o início de um novo recurso. Além disso, a organização pode aplicar o aprendizado para melhorar os processos, as práticas, as pessoas e a arquitetura do produto.

2.2.6 Implementação do Pipeline de CI/CD

A implementação de um pipeline de CI/CD deve seguir uma abordagem incremental, com as seguintes etapas, conforme definido por (Humble, Farley, 2010).

1. Modele o fluxo de valor desde o commit até o lançamento e crie um esboço inicial

A técnica de mapeamento do fluxo de valor (VSM) mapeia a parte do fluxo de valor desde o check-in do código até o lançamento (ver (ISTQB® CT-ATLaS) para obter detalhes sobre VSM). Em seguida, as etapas mapeadas são implementadas sem conteúdo detalhado no pipeline de CI/CD. Quando os critérios de qualidade predefinidos são atendidos, isso aciona a próxima etapa do processo mapeado, e a saída de uma etapa se torna uma entrada para outras (por exemplo, artefatos de compilação e bancos de dados de teste).

O código do pipeline de CI/CD geralmente é armazenado no mesmo repositório que o código-fonte do aplicativo. No entanto, em projetos mais complexos, os fluxos de trabalho podem ser reutilizados entre repositórios.

Um esqueleto funcional mostra o fluxo e integra as etapas antes de adicionar detalhes.

2. Automatizar o processo de compilação e implantação

O conjunto de ferramentas de CI deve acionar uma compilação a cada check-in. O processo de compilação toma o código-fonte como entrada e produz programas como saídas. Após a compilação, a saída é automaticamente implantada em um ambiente de teste e os testes são executados.

O provisionamento de um ambiente e dos dados de teste deve ser totalmente automatizado e reproduzível. As tecnologias de IaC, virtualização e conteinerização dão suporte a esta etapa (ver Seção 4.2.3 e Seção 4.2.9, respectivamente). As funções de operações geralmente estão envolvidas no desenvolvimento dessa automação (ver Seção 1.2.1 e Seção 1.2.2). Os ambientes de teste são implantados da mesma forma que os ambientes de produção.

3. Automatizar a etapa de commit

A próxima etapa é implementar uma fase de commit completa (ver Seção 2.2.2), executando análises estáticas e testes de unidade, e incluindo uma seleção de testes de componentes, testes de integração e testes de sistema em cada commit. Nesta etapa, a varredura de vulnerabilidades e a mapeamento de dependências implementam práticas de segurança. É comum implementar um controle de qualidade para verificar se a compilação precisa ser interrompida caso os limites acordados não sejam atingidos (por

© International Software Testing Qualifications Board

exemplo, cobertura de código insuficiente ou tempo de execução de teste muito longo). Quando a etapa de commit se torna muito longa (por exemplo, mais de cinco minutos), faz sentido dividi-la em tarefas paralelas.

4. Automatizar a etapa de aceite

A implementação da etapa de aceite (ver Seção 2.2.2) deve começar com a automação de alguns casos de teste de todos os diferentes tipos de teste no pipeline de CI/CD e o crescimento incremental das suítes de teste, em vez de tentar automatizar a maioria dos casos de teste de um tipo de teste antes de passar para o próximo.

Os testes são executados sequencialmente ou em paralelo, cada um em seu próprio ambiente, para otimizar o feedback. No entanto, leve em consideração a disponibilidade de recursos, o custo e as metas de sustentabilidade.

5. Automatize o lançamento

Automatizar esse processo requer compreender a forma como a decisão de lançamento é tomada durante o processo de lançamento (ver Seção 4.2.2) e exige a coleta de informações para tomar essa decisão, seja ela automatizada ou manual. Ferramentas de gerenciamento de recursos podem auxiliar os tomadores de decisão a gerenciar suas decisões (por exemplo, atualizando features toggles no código) (ver Seção 4.2.4). A automação incremental também se aplica a essas atividades de lançamento.

O processo implementado deve seguir a estratégia de implantação e lançamento acordada, oferecer suporte a rollbacks rápidos e fornecer uma trilha de auditoria de todas as atividades para fins de conformidade.

6. Evolução do pipeline de CI/CD

Implemente o pipeline de CI/CD de forma incremental. À medida que o projeto se torna mais complexo, o fluxo de valor evoluirá. Avalie a eficiência do pipeline de CI/CD usando métricas acordadas (por exemplo, tempo de execução de qualquer atividade específica). Evolua e aprimore continuamente o pipeline de CI/CD de acordo com os princípios lean e o pensamento sistêmico (por exemplo, dividindo execuções de testes longas em tarefas paralelas ou removendo tarefas que não agregam valor). Consulte (ISTQB® CT-ATLaS) e (ISTQB® CTAL-TM) para práticas de melhoria contínua.

© International Software Testing Qualifications Board

3 Tarefas automatizadas e manuais em DevOps - 510 minutos

Palavras-chave

Teste de API, teste automatizado, teste de contrato, teste colaborativo, teste exploratório, busca de qualidade, relatório de qualidade, relatórios de qualidade, teste de regressão, automação de testes, dados de teste, gerenciamento de dados de teste, resultado do teste, rastreabilidade

Palavras-chave específicas de qualidade em DevOps

artefato de compilação, pipeline de CI/CD, tempo de recuperação de implantação que falhou, informações de identificação pessoal, relatório de qualidade, fonte única de verdade, binário de software, análise estatística

Objetivos de aprendizagem no Capítulo 3:

3.1 Descrever como a automação apoia a garantia da qualidade em DevOps

QDO-3.1.1 (K2) Explicar como uma fonte única de verdade para o testware apoia a garantia da qualidade

QDO-3.1.2 (K2) Explicar como a automação apoia a rastreabilidade entre a base de teste e o testware

QDO-3.1.3 (K3) Implementar práticas uniformes de relatórios de qualidade no pipeline de CI/CD com informações tanto de testes automatizados quanto de testes manuais

QDO-3.1.4 (K2) Explicar os benefícios da automação do gerenciamento de dados de teste

QDO-3.1.5 (K2) Explique os benefícios da análise estatística dos resultados dos testes ao longo de longos períodos

QDO-3.1.6 (K2) Explicar os benefícios do gerenciamento padronizado, controlado e automatizado do ambiente de teste

3.2 Implementar testes automatizados em equipes de DevOps

QDO-3.2.1 (K2) Explicar como os testes de regressão se integram às diversas atividades do pipeline de CI/CD

QDO-3.2.2 (K3) Preparar testes de API

QDO-3.2.3 (K2) Comparar o grau de automação de testes em DevOps com o desenvolvimento ágil de software e modelos de desenvolvimento sequencial

3.3 Implementar testes manuais em equipes de DevOps

QDO-3.3.1 (K2) Dar exemplos de testes manuais para uma organização de DevOps

QDO-3.3.2 (K2) Explicar como os testes exploratórios podem funcionar com um pipeline de CI/CD

QDO-3.3.3 (K2) Explicar como o teste de multidão pode apoiar a cultura de feedback

QDO-3.3.4 (K3) Aplicar eventos de busca pela qualidade para apoiar a cultura de aprendizagem

Objetivos práticos:

3.1 O apoio da automação à garantia de qualidade (QA) no DevOps

QDO-3.1 (H2) Descrever como a automação contribui para a garantia de qualidade no DevOps

3.2 Testes automatizados em equipes de DevOps

© International Software Testing Qualifications Board

QDO-3.2 (H2) Automatizar de testes em equipes de DevOps

3.3 Testes manuais em equipes de DevOps

QDO-3.3 (H2) Implementar testes manuais em equipes de DevOps

© International Software Testing Qualifications Board

3.1 O apoio da automação à garantia da qualidade (QA) em DevOps

A automação desempenha um papel crucial no QA em DevOps, melhorando a consistência, a rastreabilidade e a eficiência ao longo do ciclo de vida do desenvolvimento de software (SDLC). A implementação de uma fonte única de verdade (SSOT) ajuda a centralizar e padronizar as informações, garantindo que as equipes trabalhem com dados precisos e atualizados, o que aprimora a colaboração e reduz erros. A rastreabilidade automatizada entre requisitos, testware e implantações garante a contabilização em cada alteração, melhorando o gerenciamento de defeitos e a conformidade. Os relatórios de qualidade integrados ao pipeline de CI/CD fornecem insights em tempo real por meio de painéis de controle, combinando resultados de testes automatizados e manuais para análises abrangentes e alertas. Isso proporciona um feedback rápido e confiável sobre a qualidade para as equipes e as partes interessadas. A automação no gerenciamento de dados de teste, na análise estatística e no provisionamento do ambiente de teste acelera os ciclos de teste, ajuda a garantir a segurança dos dados, otimiza o uso de recursos e aprimora a tomada de decisões, impulsionando melhorias contínuas de qualidade no DevOps.

(H2) Descreva como a automação apoia a garantia da qualidade em DevOps

Para este objetivo prático, o exercício de nível H2 auxilia os alunos a aprender como implementar a automação para apoiar a garantia de qualidade no DevOps. Este exercício permite que os alunos apliquem conceitos de automação a um contexto organizacional conhecido e reflitam sobre como a automação apoia a garantia de qualidade em todo o fluxo de valor do DevOps.

3.1.1 Fonte Única de Verdade (SSOT) para testware

À medida que as organizações de DevOps crescem, gerenciar múltiplas fontes de informação torna-se cada vez mais desafiador. Informações fragmentadas, inconsistentes e duplicadas entre equipes e ferramentas de DevOps levam a erros, colaboração deficiente e dificuldade em manter a transparência e a rastreabilidade entre itens de configuração. Ao aplicar uma arquitetura e práticas de fonte única de verdade (SSOT) em nível organizacional, as equipes de DevOps podem centralizar e padronizar as informações, garantindo que um tipo de informação seja armazenado em apenas um local de armazenamento. Isso garante que todas as equipes tenham acesso a dados precisos, consistentes e atualizados, o que ajuda a evitar erros causados pela confiança em informações incorretas.

Armazene produtos de trabalho de desenvolvimento e teste em sistemas SSOT (ver Seção 4.1.1) da seguinte forma:

- Os sistemas de gerenciamento de informações fornecem gerenciamento do ciclo de vida da informação ao armazenar itens do backlog (por exemplo, requisitos, relatórios de defeitos e tarefas) e documentação de projetos e produtos
- Os sistemas de controle de versão (VCSs) oferecem suporte ao gerenciamento controlado de mudanças, armazenando código-fonte de software, scripts de compilação e implantação, configurações de infraestrutura como código (IaC), definições de pipeline de CI/CD, casos de teste, código de automação de testes e dados de teste (por exemplo, dados em formato de texto)
- Os sistemas de gerenciamento de artefatos oferecem suporte à reusabilidade de artefatos de compilação ao armazenar dependências, binários de software e dados de teste (por exemplo, dados binários)
- Os sistemas de observabilidade fornecem rastreabilidade e gerenciamento de acesso ao armazenar rastreamentos, logs e métricas

Em geral, os sistemas SSOT oferecem às equipes de DevOps os seguintes benefícios:

© International Software Testing Qualifications Board

- Centralização: atua como o único local onde dados críticos são armazenados, atualizados e recuperados
- Consistência: garante que todas as equipes trabalhem com as mesmas informações atualizadas, evitando discrepâncias
- Automação: facilita fluxos de trabalho automatizados por meio da integração com ferramentas e sistemas em toda a cadeia de ferramentas de DevOps
- Colaboração: promove o alinhamento entre equipes, proporcionando uma compreensão comum das informações
- Auditabilidade: fornece um histórico claro das alterações, o que ajuda a rastrear e garantir a conformidade

3.1.2 Rastreabilidade entre a Base de Teste e o Testware

É importante que as equipes de DevOps registrem a relação entre produtos de trabalho relacionados, como requisitos, histórias do usuário, código e testware. A rastreabilidade garante que todos os produtos de trabalho do SDLC estejam conectados, permitindo que as equipes verifiquem se cada requisito foi implementado, testado, entregue, implantado e lançado. A rastreabilidade bidirecional também apoia a manutenção (ISTQB® CTFL). Por exemplo, quando uma alteração é feita em um requisito, é fácil descobrir onde implementar tal alteração. Além disso, a causa-raiz de um defeito de código pode ser rapidamente encontrada em uma história do usuário ou requisito. A política de teste da organização descreve a necessidade de rastreabilidade (Marselis et al., 2020).

Ferramentas podem registrar e monitorar com eficiência a rastreabilidade de todos os produtos de trabalho relacionados. Ferramentas de automação são fundamentais para fornecer atualizações dinâmicas às matrizes de rastreabilidade e reduzir os riscos associados ao rastreamento manual. Ferramentas de gerenciamento de configuração e/ou ferramentas de gerenciamento de tarefas, capazes de registrar várias relações entre produtos de trabalho, fornecem a automação. Essas ferramentas fornecem relatórios e visões gerais dos produtos de trabalho disponíveis na organização e detalham seu uso.

A rastreabilidade promove a colaboração entre as funções de desenvolvimento, testes e operações dentro da equipe de DevOps, tornando as relações entre os produtos de trabalho transparentes e auditáveis. Essa transparência melhora o rastreamento e a resolução de defeitos e apoia a conformidade com normas e regulamentações. É um facilitador essencial da descoberta contínua, integração contínua, entrega contínua, implantação contínua, testes contínuos e monitoramento contínuo, e contribui para manter a qualidade e a agilidade no DevOps.

3.1.3 Relatórios de qualidade para um pipeline de CI/CD

A qualquer momento, é importante compreender o nível de qualidade do software ao longo do SDLC. Isso fornece feedback à equipe de DevOps sobre a qualidade do código e à gestão de produtos sobre a prontidão para lançamento. A integração de relatórios de qualidade no pipeline de CI/CD fornece o status mais recente do desenvolvimento. Cada etapa do pipeline de CI/CD gera informações para relatórios de qualidade. Alguns dados de qualidade são gerados automaticamente pelo pipeline de CI/CD (por exemplo, sucesso da compilação, status de aprovação/reprovação do teste), enquanto outros dados provêm de testes manuais (por exemplo, teste de aceite de usuário, teste exploratório ou cobertura em uma ferramenta de gerenciamento de teste). Essas informações criam uma visão abrangente da qualidade do software.

Os relatórios de qualidade devem ocorrer de forma consistente em cada ciclo do pipeline de CI/CD e, portanto, devem ser automatizados. Os resultados dos testes manuais devem ser incluídos nos relatórios de qualidade, normalmente por meio da integração com uma ferramenta de gerenciamento de teste. As

© International Software Testing Qualifications Board

informações são coletadas em métricas e combinadas em um relatório de qualidade. O relatório de qualidade pode ser entregue por diversos meios (por exemplo, por meio de um painel de controle em uma página wiki, em uma ferramenta compartilhada, como um e-mail ou relatório baseado em um modelo, ou como uma combinação desses) (ISTQB® CTFL).

A geração automatizada de relatórios de qualidade requer a integração de várias ferramentas. Entre elas estão as funcionalidades de registro de logs, monitoramento e geração de relatórios de diversas ferramentas de pipeline de CI/CD (ver Seção 4.1.2). As ferramentas de gerenciamento de tarefas e de testes registram os resultados dos testes manuais (por exemplo, o nível estimado de qualidade dos recursos de software ou os resultados de aprovação/reprovação dos testes). As ferramentas de monitoramento de produção fornecem telemetria (ver Seção 4.2.7). Os geradores de relatórios devem se integrar a todas essas ferramentas para criar relatórios fáceis de entender para as diversas partes interessadas. Crie uma única fonte de verdade combinando as informações de métricas de várias fontes em um único local com acessibilidade (ver Seção 3.1.1).

O processo de implementação de relatórios de qualidade tem várias etapas:

1. Determine os indicadores-chave de desempenho (KPIs) e as métricas relacionadas a serem monitoradas, tais como qualidade do código, cobertura, performance, estabilidade da compilação, taxa de sucesso da compilação, frequência de implantação, tempo de recuperação de implantações que falharam e vulnerabilidades de segurança.
2. Colete dados usando as métricas escolhidas.
3. Automatize a coleta de dados de vários estágios do pipeline de CI/CD.
4. Limpe e normalize os dados para obter maior consistência com ferramentas de preparação de dados (ver Seção 3.1.4).
5. Armazenar os dados coletados em um banco de dados centralizado, caso as ferramentas não ofereçam essa funcionalidade.
6. Visualize as métricas usando painéis de controle, como gráficos, tabelas e diagramas que exibam as principais métricas, incluindo seções diferentes para cada etapa do pipeline de CI/CD.
7. Habilite o monitoramento contínuo no painel de controle para atualizações em tempo real.
8. Garanta a acessibilidade do painel de controle para as partes interessadas relevantes, como desenvolvedores, testadores, product owners, scrum masters e gerentes.

Um painel de controle de relatórios de qualidade pode aprimorar um pipeline de CI/CD. Os benefícios incluem:

- Acompanhamento contínuo das principais métricas para resposta e resolução rápidas
- Análise das tendências de performance do pipeline de CI/CD para identificar gargalos e otimizar a eficiência do pipeline
- Revisão proativa do status de conformidade de segurança para garantir que a base de código atenda aos padrões de segurança
- Promover uma cultura de transparência e colaboração por meio de painéis de controle compartilhados com os stakeholders, alinhando todos quanto ao status de qualidade e ao progresso do projeto
- Apoiar retrospectivas com insights baseados em dados para identificar áreas de melhoria

© International Software Testing Qualifications Board

3.1.4 Automação do gerenciamento de dados de teste

O gerenciamento manual de dados de teste pode reduzir o fluxo e retardar os ciclos de feedback (ver Seção 1.1.5). As equipes de DevOps utilizam a automação do gerenciamento de dados de teste para criar, atualizar e gerenciar dados de teste dentro do pipeline de CI/CD. O uso de dados de teste padronizados ajuda a acelerar os ciclos de teste, melhorar a cobertura, reduzir erros, diminuir a inconsistência dos resultados de teste, melhorar a segurança dos dados e cumprir as regulamentações de privacidade.

A automação eficaz do gerenciamento de dados de teste requer colaboração entre todas as funções da equipe de DevOps para identificar em quais dados se concentrar e quais práticas utilizar.

Os principais aspectos do gerenciamento de dados de teste incluem:

- Projeto de dados: analisar e definir os dados de teste necessários para um determinado objetivo do teste (por exemplo, tipos de dados, tamanhos e complexidade).
- Geração de dados: criar ou atualizar dados de teste para permitir uma cobertura abrangente e uma variedade de entradas (por exemplo, dados válidos e inválidos, casos extremos e valores limite).
- Validação de dados: automatizar a validação para determinar a precisão dos dados e a conformidade com normas e regulamentações.
- Organização de dados: Organize e armazene os dados de teste de forma estruturada em um VCS e garanta a sua disponibilidade sob demanda dentro do pipeline de CI/CD ou para testes manuais.
- Anonimização: Mascare ou anonimize os dados de produção antes de usá-los como dados de teste para proteger a privacidade e a confidencialidade.
- Atualização e limpeza de dados: atualize ou redefina automaticamente os dados de teste como parte da automação de testes.

As equipes de DevOps aplicam as seguintes práticas na automação do gerenciamento de dados de teste:

- Criação de dados de testes sintéticos: Regras de formatação de dados, análises da estrutura dos dados de produção ou modelos GenAI pré-treinados podem gerar dados artificiais para corresponder às condições de teste, sem usar dados reais de produção.
- Clonagem de dados de teste: crie dados de teste válidos e em grande volume para testes de performance e de carga clonando dados existentes.
- Subdivisão de dados de teste: Selecione um subconjunto menor e representativo dos dados de produção para testes, mantendo as referências entre os itens de dados, a fim de reduzir as necessidades de armazenamento e acelerar o provisionamento de dados de teste.
- Mascaramento de dados de teste: Mascarar ou modificar informações de identificação pessoal (PII) dos dados de produção para tornar os dados anônimos e proteger a identidade de um indivíduo (por exemplo, nome, endereço, e-mail, número de telefone, endereço IP e dados de localização).

A equipe de DevOps deve compreender os riscos da automação do gerenciamento de dados de teste e gerenciá-los usando processos relevantes e ferramentas bem selecionadas e configuradas. Esses riscos incluem:

- Cobertura incompleta: os dados gerados podem não refletir a complexidade do mundo real.
- Problemas de utilização de recursos: gerar, armazenar e distribuir grandes quantidades de dados de teste pode consumir recursos significativos.
- Violação de regras de segurança e conformidade: a exposição de informações confidenciais por mascaramento inadequado de dados pode resultar em multas.

© International Software Testing Qualifications Board

3.1.5 Análise estatística dos resultados dos testes

A execução de teste produz vários tipos de resultados. O uso de curto prazo dos resultados individuais dos testes é informar os stakeholders sobre a qualidade e os riscos relacionados, para que possam estabelecer sua confiança no valor de negócio do software. O uso de longo prazo dos resultados de teste é obter uma visão mais ampla da qualidade do software, dos processos aplicados e das pessoas envolvidas. Em ambos os casos, ao aplicar a análise estatística, a equipe de DevOps pode identificar tendências e padrões relacionados à qualidade do software e aos riscos residuais, e usar essas informações para apoiar a tomada de decisões. Para que as partes interessadas estabeleçam confiança, elas precisam de métricas como cobertura de requisitos, cobertura de código, densidade do defeito e tempo médio entre falhas.

A equipe de DevOps pode usar ferramentas como análises preditivas baseadas em IA para prever resultados futuros que o software produzirá. As análises preditivas apoiam a melhoria tanto do processo quanto das pessoas envolvidas quando a situação futura prevista não atende à situação desejada. A análise preditiva pode resultar na necessidade de melhoria do processo com base na efetividade do teste medido ou pode desencadear a alocação de mais recursos ou de recursos diferentes.

O teste A/B utiliza análise estatística para comparar múltiplas variantes de software apresentadas a grupos de usuários distintos, a fim de determinar qual variante é a mais valiosa. Outro caso de uso da análise estatística é a avaliação dos resultados dos testes não funcionais, como em testes de performance baseados em perfis de carga (Marselis et al., 2020).

A análise estatística é valiosa para avaliar os resultados de uma ou várias rodadas de teste (por exemplo, para identificar tendências na qualidade do software e determinar se a qualidade aumenta após cada iteração). A análise estatística pode ser utilizada para a melhoria do processo de teste. Isso ajuda a detectar testes inconsistentes e a analisar palavras-chave de testes automatizados com tempos de execução prolongados, o que contribui para a otimização do tempo de execução dos testes e dos custos associados. Um painel de controle pode exibir os resultados da análise estatística por meio de ferramentas.

As ferramentas para análise estatística devem ser integradas ao pipeline de CI/CD.

3.1.6 Gerenciamento padronizado, controlado e automatizado do ambiente de teste

O gerenciamento padronizado, controlado e automatizado do ambiente de teste é um pilar do desenvolvimento de software. Ao padronizar os ambientes de teste, as organizações eliminam a variabilidade causada por configurações inconsistentes, garantindo que os testes produzam resultados confiáveis e reproduzíveis. Essa consistência reduz o tempo gasto na correção de defeitos decorrentes de diferenças ambientais, permitindo que as equipes de DevOps se concentrem nos defeitos reais no código, em vez de discrepâncias de configuração.

A automação agiliza o provisionamento e o gerenciamento de ambientes de teste, substituindo processos manuais demorados por configurações rápidas e repetíveis. Isso permite ciclos de teste mais rápidos, nos quais os ambientes podem ser criados e desativados conforme necessário. Isso pode incluir o gerenciamento de dados de teste (ver Seção 3.1.4 e Seção 4.2.9). As equipes de DevOps podem realizar testes e iteragir em seu código com mais frequência, acelerando a entrega de software por meio do pipeline de CI/CD e aumentando a produtividade geral.

Ambientes controlados permitem a eficiência de recursos, garantindo que os ambientes tenham o tamanho adequado e sejam adaptados às necessidades de testes específicos. Isso reduz o risco de provisionamento excessivo, em que recursos são alocados desnecessariamente, bem como de provisionamento insuficiente, o que pode causar gargalos de desempenho. O provisionamento de ambientes sob demanda reduz os custos operacionais e utiliza recursos apenas conforme necessário.

O uso de IaC para criar ambientes replicáveis (ver Seção 4.2.3) elimina ambiguidades sobre configurações, melhorando a colaboração entre testadores, desenvolvedores e operações. O uso de IaC apoia o objetivo

© International Software Testing Qualifications Board

de remover o muro da confusão. Esses ambientes servem então como modelos que podem ser facilmente replicados na nuvem, de modo que podem ser dimensionados com facilidade, acomodando demandas crescentes de testes ou equipes maiores.

3.2 Teste automatizado em equipes DevOps

Os testes de regressão são cruciais no DevOps para manter a qualidade em casos de implantações frequentes. Os testes de regressão envolvem a automação de testes para cobrir recursos novos e existentes, equilibrando otimização e cobertura com base nos níveis de risco e nos níveis de qualidade almejados.

O Teste de API verifica as interfaces do sistema quanto à adequação funcional, confiabilidade, eficiência de performance, segurança e outras características de qualidade. Ele oferece feedback rápido e se integra bem aos pipelines de CI/CD.

Em modelos de desenvolvimento sequenciais, a automação de testes ocorre após o desenvolvimento, enquanto no desenvolvimento ágil de software ela ocorre ao longo de todo o SDLC. O DevOps depende fortemente da automação de testes em todas as fases do SDLC, criando ciclos de feedback rápidos e atendendo a várias necessidades de teste com práticas como a conteinerização.

(H2) Automação de testes em equipes DevOps

Para este objetivo prático, o exercício de nível H2 auxilia os alunos a aprender como definir o escopo e o papel da automação de testes no contexto do DevOps. O foco não está nas ferramentas ou na implementação, mas nas decisões estratégicas sobre:

- O que deve ser automatizado
- Onde a automação é essencial, útil ou opcional
- O que não deve ser automatizado e por que

3.2.1 Teste de regressão no pipeline de CI/CD

O objetivo do pipeline de CI/CD é entregar software em produção com boa qualidade. Os testes de regressão determinam se as alterações no código não afetam negativamente a funcionalidade existente da aplicação. Trata-se de uma rede de segurança crítica devido às frequentes alterações e implantações de código. Na prática, toda a automação de testes no pipeline de CI/CD é usada inicialmente para cobrir novas funcionalidades e, posteriormente, pode se tornar um teste de regressão.

Os testes de regressão geralmente utilizam múltiplos ambientes de teste do mesmo tipo e os executam em paralelo para reduzir o tempo de execução dos testes. Cada ambiente de teste suporta cada tipo de teste para que haja um número adequado de recursos e integrações com outros ambientes disponíveis (ver Seção 3.1.6).

Teoricamente, todos os testes podem ser executados a cada execução do pipeline de CI/CD. No entanto, mesmo que isso fosse rápido o suficiente, seria prejudicial para a viabilidade e sustentabilidade do negócio, pois consome recursos desnecessários (ou seja, tempo, custo e energia). Na prática, a equipe de DevOps pode criar várias suítes de testes de regressão com diferentes níveis de cobertura e executá-las para diferentes finalidades, com diferentes frequências ou com base nas alterações no sistema em teste. Alguns testes de regressão podem ser executados antes da incorporação ao código-fonte, outros após a incorporação e outros ainda durante a madrugada. A equipe de DevOps geralmente seleciona os testes de regressão com base no risco (por exemplo, ela pode fazer com que o pipeline de CI/CD execute testes de componentes com base em mapas de calor, mostrando as áreas alteradas do código no commit mais

© International Software Testing Qualifications Board

recente). Tags e filtros permitem a escolha dinâmica dos testes de regressão. Outro exemplo é a execução de testes de regressão para recursos existentes que são afetados por uma alteração.

O pipeline de CI/CD deve executar uma suíte completa de testes de regressão antes do lançamento do software. A equipe de DevOps deve equilibrar a otimização do número de testes de regressão e a necessidade de boa cobertura antes de um lançamento. Dependendo do nível de qualidade almejado, os testes de regressão podem ser executados apenas ocasionalmente (por exemplo, para um relatório de teste de segurança). Os testes de regressão devem ser o mais automatizados possível e fazer parte do projeto do pipeline de CI/CD (ver Seção 2.2.6).

3.2.2 Pontos-chave do Teste de API

Os testes de API concentram-se nas interfaces entre sistemas e na forma como estes comunicam. Envolvem a verificação da adequação funcional, confiabilidade, eficiência de performance, segurança e outras características de qualidade da API. As APIs são a espinha dorsal das aplicações modernas, e testá-las o mais cedo possível evita defeitos dispendiosos.

O Teste de API é utilizado para determinar se a troca de dados entre sistemas é consistente, correta, oportuna e segura.

Aspectos-chave do teste de API

- Teste funcional: determina se a API se comporta conforme o esperado para todas as entradas e produz resultados corretos.
- Teste de confiabilidade: determina se a API funciona conforme o esperado em diferentes cenários.
- Teste de performance: medem os tempos de resposta, a taxa de transferência e a escalabilidade.
- Teste de segurança: validam os mecanismos de autenticação, autorização e proteção de dados.

Existem muitas ferramentas de Teste de API disponíveis, e a maioria das linguagens de programação oferece bibliotecas que podem ser utilizadas.

Os benefícios do Teste de API incluem:

- Feedback mais rápido em comparação com o teste de interface gráfica
- Fácil integração em pipelines de CI/CD
- Teste no início do SDLC (ou seja, shift left)

Mas também há desafios:

- Garantir cobertura completa para APIs complexas
- Manter os testes atualizados com contratos de API em evolução
- Gerenciamento de dependências de APIs ou serviços externos

Teste de contrato

O teste de contrato é um tipo de teste de integração que testa cada aplicativo isoladamente para determinar se as mensagens que cada um envia ou recebe estão em conformidade com um entendimento comum documentado em um “contrato” (Fellows, 2022).

O teste de contrato valida o acordo (ou seja, o contrato) entre um provedor e seus consumidores. O provedor é acessado por meio de uma API e, em vez de testar a implementação ou a lógica de negócios por trás da API, o teste de contrato verifica se a API segue uma estrutura, um formato e um conjunto de

© International Software Testing Qualifications Board

expectativas definidos para cargas de solicitação e resposta, cabeçalhos, códigos de status e tipos de dados.

O teste de contrato é importante em arquiteturas de microsserviços, onde muitos serviços desenvolvidos de forma independente devem se integrar de forma confiável.

O teste de contrato apoia o ciclo de feedback rápido no DevOps, fornecendo verificações rápidas e isoladas para determinar se as alterações não afetarão negativamente os sistemas subsequentes.

Implementação de testes de contrato

1. Defina o contrato: comece especificando claramente a estrutura e o comportamento esperados da API, incluindo formatos de solicitação, campos de resposta, códigos de status e tipos de dados.
2. Escreva testes de contrato: com base no contrato definido, os consumidores especificam como pretendem interagir com a API. Essas especificações orientam os casos de teste que validam a implementação do provedor. Tanto o consumidor quanto o provedor usam o mesmo contrato para verificar se a implementação está correta do lado do provedor e se o consumidor está chamando a API corretamente.
3. Execute no pipeline de CI/CD: integre os testes de contrato ao pipeline de CI/CD para fornecer feedback antecipado e detectar alterações que causam quebras de compatibilidade antes da implantação.
4. Controle de versão e gerenciamento de contratos: Em organizações com várias equipes ou serviços, os contratos devem ser versionados e armazenados no controle de versão (ver Seção 3.1.1). Isso permite que as alterações sejam rastreadas, submetidas a revisão e coordenadas entre os serviços.

Considerações e trade-offs com testes de contrato

- Os testes de contrato não cobrem todos os aspectos funcionais do Teste de API e complementam outros tipos de teste.
- Alterações no contrato exigem coordenação cuidadosa entre diferentes equipes.
- A equipe de desenvolvimento precisa investir em ferramentas e definir padrões compartilhados.

3.2.3 Compare a automação de testes em diferentes modelos de SDLC

A automação de testes se aplica a qualquer SDLC, embora seu momento de aplicação e ênfase variem (ISTQB® CTAL-TAE).

Os modelos de desenvolvimento sequencial introduzem a automação de testes além dos testes de unidade na fase de testes após a conclusão do desenvolvimento e oferecem um ciclo de feedback limitado entre as fases de desenvolvimento e de testes. O desenvolvimento ágil de software integra a automação de testes em todo o SDLC. No DevOps, a automação de testes torna-se um facilitador central de CI/CD, com um escopo mais amplo que abrange vários níveis e tipos de testes. Ela envolve uma forte dependência de IaC (ver Seção 4.2.3) para o provisionamento de ambientes de teste e frequentemente inclui testes em produção por meio de práticas como implantação blue-green e lançamentos canários (ver Seção 2.2.4).

Em modelos de desenvolvimento sequenciais, a automação de testes além dos testes de unidade geralmente se limita às fases finais do desenvolvimento. Por meio da abordagem incremental dentro das iterações no desenvolvimento ágil de software, há um foco maior em testes de unidade, testes de componentes e testes de integração de componentes. No DevOps, a dependência da automação de testes é fundamental em todas as etapas do pipeline de CI/CD. Isso cria ciclos de feedback rápidos e automatizados por meio de testes pré-merge, testes em produção e monitoramento.

© International Software Testing Qualifications Board

A automação de testes traz seus desafios únicos, dependendo do SDLC. O principal desafio da automação de testes no DevOps é, muitas vezes, gerenciar a complexidade de diferentes requisitos para os níveis de teste separados (por exemplo, componentes de sistema totalmente integrados em testes de ponta-a-ponta versus serviços simulados para testes de componentes). A conteinerização ajuda a enfrentar esses desafios, fornecendo ambientes isolados e reproduzíveis (ver Seção 4.2.9). Considerações culturais e de processo influenciam o uso da automação de testes em diferentes SDLCs. Modelos de desenvolvimento sequenciais veem os testes como uma fase distinta. O desenvolvimento ágil de software promove uma mentalidade colaborativa entre desenvolvedores, testadores e product owners no que diz respeito aos testes. O DevOps dilui ainda mais as fronteiras tradicionais de funções ao envolver testadores em revisões de código, monitoramento e configurações de observabilidade, e ao envolver desenvolvedores em atividades de teste e monitoramento.

3.3 Testes manuais em equipes DevOps

Apesar da forte ênfase na automação no DevOps, os testes manuais continuam essenciais para coletar informações críticas que os testes automatizados não podem fornecer. Os testes manuais permitem que as partes interessadas avaliem diretamente a usabilidade, ganhem confiança na qualidade do software e avaliem aspectos difíceis de automatizar, como a experiência do usuário (UX). Os testes manuais desempenham um papel crucial nos pipelines de CI/CD e em ambientes de produção por meio de testes A/B e ativação/desativação de recursos (ver Seção 4.2.4).

As principais abordagens de testes manuais em DevOps incluem testes exploratórios, testes de multidão e busca de qualidade, cada uma oferecendo benefícios únicos para avaliar a qualidade do software de forma dinâmica. Essas abordagens de teste complementam os testes automatizados, fornecendo insights diversos e garantindo que o software atenda tanto às expectativas técnicas quanto às dos usuários.

(H2) Implementar testes manuais em equipes de DevOps

Para este objetivo prático, o exercício de nível H2 auxilia os alunos a aprender como identificar, implementar e analisar abordagens de testes manuais em DevOps, por exemplo, com testes exploratórios, testes colaborativos ou caça da qualidade, para complementar os testes automatizados.

3.3.1 Exemplos de testes manuais

Um dos princípios do DevOps é automatizar o máximo possível. Isso leva à crença de que não há espaço para testes manuais no pipeline definitivo de CI/CD. No entanto, testar consiste em coletar informações sobre a qualidade do objeto de teste, para que os stakeholders possam estabelecer sua confiança na solução e, portanto, esses stakeholders precisam de vários tipos de informações (Marselis et al., 2020).

Os testes automatizados no pipeline de CI/CD podem fornecer muitas informações sobre o nível de qualidade do objeto de teste (ver Seção 2.2.4 e Seção 2.2.5). No entanto, os testes manuais continuam sendo uma atividade essencial por várias razões. Uma delas é que os testes manuais fornecem outros tipos de informações que os testes automatizados não oferecem (por exemplo, determinar a usabilidade do software é difícil de automatizar, mas fácil de realizar com testes manuais). Em segundo lugar, algumas partes interessadas ganham maior confiança no nível de qualidade do software quando o veem com seus próprios olhos. Em terceiro lugar, os testes manuais são flexíveis, utilizam o julgamento humano e podem ser mais eficientes do que os testes automatizados, dependendo da situação específica.

Por essas razões, os testes manuais são necessários, tanto durante as atividades do pipeline de CI/CD quanto, possivelmente, durante a operação em produção (por exemplo, em testes A/B ou quando o software foi implantado no ambiente de produção, mas ainda não foi lançado para os usuários, por meio da aplicação de feature toggles) (ver Seção 4.2.4). O teste manual inicial consiste em testes estáticos de requisitos e/ou histórias do usuário (por exemplo, em refinamentos usando sessões dos 4 amigos — com

© International Software Testing Qualifications Board

representação de análise de negócios, desenvolvimento, testes e operações) e revisões de código e produtos de trabalho relacionados como parte de pull requests (PRs) (ver Seção 2.1.4). Outras abordagens relevantes para testes manuais em DevOps são testes exploratórios (ver Seção 3.3.2), testes de multidão (ver Seção 3.3.3) e caça da qualidade (ver Seção 3.3.4). Essas abordagens de teste fornecem testes dinâmicos antecipados que complementam os testes automatizados no pipeline de CI/CD.

3.3.2 Teste exploratório no pipeline de CI/CD

Alguns pensam que o teste exploratório significa apenas testar sem casos de teste ou procedimentos de teste pré-documentados, mas há mais do que isso. Explorar envolve investigação rigorosa e analítica (Hendrickson, 2013). A exploração é a contrapartida da automação. Enquanto o feedback automatizado vem de ferramentas, o feedback da exploração vem das pessoas (Clokie, 2017).

Nos testes exploratórios, os testes são simultaneamente projetados, executados e avaliados enquanto o testador aprende sobre o objeto de teste. Consulte (ISTQB® CTFL) para a descrição dos testes exploratórios. Os testes exploratórios podem ser realizados como testes dinâmicos por um representante do usuário e um membro da equipe para investigar o comportamento em situações específicas e ganhar confiança. Embora os testes exploratórios sejam executados manualmente, eles podem ser acionados a partir do pipeline de CI/CD. Ferramentas podem ser usadas para apoiar a criação de casos de teste durante a exploração (não antes), para apoiar a execução de testes, para registrar resultados de testes e para armazenar certos tipos de evidências (por exemplo, capturas de tela). Os resultados dos testes exploratórios devem ser registrados no pipeline de CI/CD para serem incluídos nos critérios de qualidade e nos relatórios de teste, juntamente com os resultados dos testes automatizados.

Os termos de referência dos testes exploratórios podem ser definidos em uma fase inicial (por exemplo, durante o refinamento da história do usuário), para se ter um backlog de itens de teste como parte do backlog da equipe de DevOps (ver Seção 2.2.1). Os termos de referência dos testes também podem ser criados quando surge uma determinada necessidade de testes manuais (por exemplo, com base em resultados inesperados de testes automatizados no pipeline de CI/CD).

Os testes exploratórios podem ser executados em vários ambientes de teste, variando de um ambiente de teste temporário específico criado a partir do pipeline de CI/CD até o ambiente de preparação ou no ambiente de produção (por exemplo, para alguns fins específicos controlados por feature toggles) (ver Seção 2.2.5).

O IaC pode ser usado para explorar o impacto de várias versões de infraestrutura no software. Essa abordagem fornece evidências documentadas de compatibilidade entre plataformas, permitindo que as equipes identifiquem e rastreiem defeitos específicos da infraestrutura até suas causas-raiz antes da implantação.

3.3.3 Teste de Multidão

O teste de multidão refere-se ao uso de um grupo virtual de pessoas que geralmente são externas à organização e podem estar geograficamente distribuídas, que representam usuários reais, usam dispositivos reais e executam testes individualmente de forma simultânea. O teste de multidão é usado, por exemplo, no beta teste após um lançamento canário, no teste A/B, na coleta de feedback pós-lançamento e no teste de dispositivos no mundo real. O teste de multidão geralmente aplica uma abordagem de teste semelhante ao teste exploratório, em que os membros do grupo recebem uma carta de teste para orientar seus testes sem preparar casos de teste detalhados antecipadamente.

A chave para o sucesso do teste de multidão é projetar e executar muitos testes, reunindo rapidamente informações abrangentes sobre a qualidade do software.

© International Software Testing Qualifications Board

O foco do teste de multidão está, em primeiro lugar, na amplitude, para abranger muitas personas diferentes ou combinações de software e hardware (como no beta teste de jogos ou aplicativos móveis).

Uma das principais vantagens do teste de multidão é que muitas situações e configurações diferentes podem ser testadas rapidamente. Organizações especializadas gerenciam a seleção dos testadores para teste de multidão, distribuem as cartas de teste e coletam os resultados dos testes.

Uma desvantagem ao usar testadores fora da equipe de DevOps é que a preparação exige mais esforço da equipe de DevOps (especialmente quando os testes precisam ser realizados em uma determinada ordem), e a preparação pode precisar ser mais aprofundada (por exemplo, definindo personas específicas) em comparação com os membros da equipe de DevOps realizando os testes eles mesmos.

A elaboração de relatórios sobre os resultados dos testes de multidão pode ser um desafio, por exemplo, devido à grande quantidade de dados coletados em um curto espaço de tempo. As equipes de DevOps devem realizar a revisão dos relatórios de defeitos para evitar feedback injustificado. Os testadores de teste de multidão precisam de boas instruções sobre quais informações coletar e como registrar seus resultados de teste. Algumas organizações que organizam teste de multidão oferecem suporte ativo para fornecer essas instruções e realizar essas tarefas de relatório.

Os riscos e desafios a serem considerados ao aplicar o teste de multidão são:

- Confidencialidade: quanto maior for a multidão e quanto mais ela estiver fora da esfera de influência da equipe de DevOps, mais difícil se torna gerenciar a confidencialidade.
- Conhecimento: nem todo aplicativo é adequado para testes de multidão do ponto de vista do conhecimento. Muitos aplicativos usados dentro de uma organização exigem conhecimento específico de seus produtos e processos para executar o software.
- Motivação: O método de recompensas pode influenciar a efetividade (por exemplo, testadores que são pagos “por bug” frequentemente procuram defeitos fáceis de descobrir, em vez de procurar os mais críticos).

3.3.4 Eventos de Caça da Qualidade

O software só pode ser entregue com a qualidade certa no momento certo se houver uma visão clara dos requisitos de qualidade necessários para obter o valor de negócio almejado. As pessoas envolvidas precisam ter uma visão clara dos níveis de qualidade atuais. Testes estáticos e dinâmicos, juntos, determinam o nível de qualidade atual. A imagem da qualidade também depende da impressão de que os stakeholders têm a partir de sua experiência pessoal. Para apoiar a construção dessa experiência pessoal, uma equipe de DevOps pode organizar a caça da qualidade, uma atividade colaborativa com uma abordagem de gamificação (observação: isso é diferente da caça a bugs) (Ruigrok et al., 2025).

A “caça da qualidade” é um evento em que equipes de caça da qualidade (por exemplo, com metade do tamanho de uma equipe ágil) competem para apresentar a avaliação mais valiosa do nível de qualidade do objeto de teste em um tempo limitado. O principal objetivo da caça da qualidade é obter uma visão compartilhada do nível de qualidade do objeto de teste. Ao ter várias equipes de caça da qualidade competindo para obter a melhor visão do nível de qualidade atual, essas equipes são desafiadas a criar maneiras novas e inovadoras de avaliar a qualidade e a fornecer uma boa percepção sobre um ou mais aspectos do nível de qualidade. Durante a caça da qualidade, as descobertas são registradas de forma semelhante às informações da reunião de avaliação nos testes exploratórios (ver (ISTQB® CTFL)). Os defeitos encontrados durante a caça da qualidade são tratados de acordo com o processo de gerenciamento de defeitos da organização. As informações de várias equipes geralmente se complementam e ampliam as informações obtidas com testes e outras atividades de engenharia da qualidade (QE).

© International Software Testing Qualifications Board

Os stakeholders se beneficiam da caça da qualidade porque, em pouco tempo, muitas ideias e novas perspectivas para aspectos relevantes de qualidade são investigados, de modo que obtêm uma visão geral ampla e variada da qualidade de seu sistema.

A caça da qualidade visa obter as informações mais valiosas sobre o nível de qualidade do software. O elemento lúdico consiste em uma recompensa (geralmente pequena) para a equipe de caça da qualidade que apresentar o relatório mais informativo e/ou perspectivas mais específicas sobre o nível de qualidade. A caça da qualidade é uma atividade diferente e mais ampla do que a caça a bugs (introduzida em (ISTQB® CT-MAT)). Trata-se de uma abordagem de teste gamificada que motiva a descoberta de defeitos, o que pode levar os testadores a tentar encontrar defeitos irrelevantes sem obter uma visão do nível de qualidade e do nível de risco.

A caça da qualidade é uma atividade de testes manuais integrada à etapa do pipeline de CI/CD para testar o processo de negócios como um todo.

A interação entre as diferentes equipes de caça da qualidade ao discutir seus resultados contribui para a cultura de colaboração e aprendizagem no DevOps.

© International Software Testing Qualifications Board

4 Ferramentas e práticas em DevOps - 200 minutos

Palavras-chave

injeção de falha, tolerância a falha, hotfix, teste de performance, teste de segurança, testes de unidade

Palavras-chave específicas de qualidade no DevOps

artefato de compilação, ramificação, estratégia de ramificação, pipeline de CI/CD, engenharia de caos, conteinerização, mapeamento de dependências, ativação/desativação de recursos, infraestrutura como código, fusão, observabilidade, gerenciamento de lançamento, estratégia de lançamento, lista de materiais de software, gerenciamento de código-fonte, controle de versão, virtualização

Objetivos de aprendizagem no Capítulo 4:

4.1 Selecionar ferramentas que dão suporte ao DevOps dentro da organização

QDO-4.1.1 (K1) Recordar os recursos das ferramentas em DevOps

QDO-4.1.2 (K2) Explicar as ferramentas que dão suporte à garantia da qualidade e aos testes em DevOps

4.2 Compreender tecnologias e práticas para apoiar a qualidade no DevOps

QDO-4.2.1 (K1) Recordar as atividades não relacionadas a teste no pipeline de CI/CD

QDO-4.2.2 (K2) Explicar as diferentes estratégias-chave de lançamento no DevOps

QDO-4.2.3 (K2) Resumir os conceitos de infraestrutura como código

QDO-4.2.4 (K1) Recordar os feature toggles como uma forma de separar a implantação do lançamento

QDO-4.2.5 (K1) Recordar estratégias de ramificação

QDO-4.2.6 (K1) Recordar a engenharia do caos

QDO-4.2.7 (K1) Recordar telemetria e observabilidade

QDO-4.2.8 (K1) Recordar a lista de materiais de software para apoiar o gerenciamento de configuração

QDO-4.2.9 (K1) Recordar a conteinerização

Objetivos práticos:

4.1 Ferramentas que apoiam o DevOps na organização

QDO-4.1 (H2) Ferramentas e práticas em DevOps

4.2.3 Infraestrutura como código (IaC)

QDO-4.2.3 (H0) Resuma os conceitos de Infraestrutura como Código (IaC)

© International Software Testing Qualifications Board

4.1 Ferramentas que dão suporte ao DevOps dentro da organização

As ferramentas de DevOps possibilitam recursos como descoberta contínua, integração contínua (CI), testes contínuos, entrega contínua (CD), implantação contínua e monitoramento contínuo, automatizando fluxos de trabalho, aprimorando a colaboração e melhorando a confiabilidade da entrega de software. Essas ferramentas geralmente se enquadram em categorias como controle de versão, CI/CD, gerenciamento de configuração, gerenciamento, monitoramento e registro, testes, segurança e conformidade, colaboração e planejamento. Essas ferramentas otimizam atividades essenciais. Um conjunto de ferramentas DevOps bem integrado aumenta a velocidade, a eficiência e a confiabilidade, ajudando a organização a atingir suas metas de negócios ao oferecer suporte a lançamentos de software de alta qualidade, rápidos e seguros. Uma fonte abrangente e holística de informações sobre ferramentas DevOps é a tabela periódica de ferramentas DevSecOps (digital.ai, 2023).

(H2) Ferramentas e práticas em DevOps

Para este objetivo prático, o exercício de nível H2 auxilia os alunos na escolha de diferentes ferramentas para o pipeline de CI/CD, incluindo tanto ferramentas de automação de testes quanto outras ferramentas de automação.

4.1.1 Recursos das ferramentas

Esta seção explora como as ferramentas para QE, incluindo QA e testes, são integradas aos grupos de atividades do fluxo de valor, especificamente na descoberta contínua, integração contínua (CI), entrega contínua (CD), implantação contínua e lançamento sob demanda (Seção 2.2). Abaixo está um resumo das principais categorias, suas descrições e os recursos relacionados dessas ferramentas.

1. Gerenciamento de informações: Gerencia o controle de versão, a colaboração e o rastreamento de alterações em documentos e itens do backlog, garantindo uma única fonte de verdade (SSOT) (por exemplo, documentação de projetos e produtos, requisitos, relatórios de defeitos e tarefas)
   Recursos: controle de versão, rastreabilidade, transparência e controle de alterações
2. Gerenciamento de código-fonte: Gerencia o controle de versão, a colaboração e o rastreamento de alterações em bases de código, garantindo uma SSOT
   Recursos: controle de versão, ramificação, fusão e resolução de conflitos
3. Integração contínua: automatiza a integração de alterações de código em repositórios compartilhados de software em execução, promovendo a detecção precoce de defeitos e ciclos contínuos de feedback
   Recursos: servidores de CI, pipelines de compilação automatizados e testes de integração
4. Implantação contínua: automatiza o lançamento de código validado em qualquer ambiente, reduzindo erros manuais e garantindo implantações consistentes
   Recursos: pipelines de implantação, automação de implantação e mecanismos de roll forward e rollback
5. Gerenciamento de ambiente e infraestrutura: automatiza o provisionamento e a configuração de ambientes consistentes em desenvolvimento, testes e produção
   Recursos: Infraestrutura como código (IaC), provisionamento de ambiente e gerenciamento de configuração
6. Gerenciamento de artefatos de compilação: facilita o armazenamento, a recuperação e a distribuição eficientes de artefatos de compilação, dependências e pacotes, garantindo consistência e repetibilidade entre ambientes
   Recursos: armazenamento de artefatos de compilação, gerenciamento de dependências,

© International Software Testing Qualifications Board

mapeamento de dependências, mapeamento de vulnerabilidades e distribuição segura de segurança

7. Qualidade e segurança do código: Melhora a qualidade e a resiliência do software ao detectar defeitos e vulnerabilidades antecipadamente, garantindo a adesão a práticas seguras de codificação ao longo do desenvolvimento
   Recursos: análise estática, testes de segurança de aplicativos estáticos e dinâmicos (SAST/DAST), análise de composição e dependências de software (SCA), validação de segurança da cadeia de suprimentos (ISTQB® CT-SEC), testes de unidade e avaliação de cobertura de código
8. Integração e automação de testes: valida as interações entre os componentes do sistema e a funcionalidade geral, ao mesmo tempo em que automatiza os testes para maior eficiência
   Recursos: testes de contrato, testes de API, testes de integração de componentes, testes de interface do usuário e frameworks de automação de testes reutilizáveis
9. Teste de performance e teste de segurança: avalia a escalabilidade, a tolerância a falha e a segurança da aplicação, por meio de simulações de condições de carga e mapeamento de vulnerabilidades (ISTQB® CT-PT) e (ISTQB® CT-SEC)
   Recursos: testes de performance, testes de carga, testes de estresse, testes de pico, testes de volume, teste de resistência, mapeamento de vulnerabilidades e testes de penetração
10. Telemetria (monitoramento e observabilidade): fornece insights em tempo real sobre a integridade da aplicação e da infraestrutura, ajudando a identificar defeitos de forma proativa
    Recursos: monitoramento de performance, agregação de logs, análise de logs, rastreamento distribuído e alertas
11. Engenharia de caos e injeção de falhas: simula falhas para testar a resiliência e a tolerância a falha do sistema, identificando fraquezas sob estresse ou em quaisquer condições inesperadas
    Recursos: injeção de falhas, testes de resiliência e validação de tolerância a falha
12. Dados de teste: Gere, gerencie e garanta a segurança dos dados de teste para apoiar processos de teste confiáveis e em conformidade
    Recursos: geração de dados de testes sintéticos, anonimização, mascaramento, subconjuntos e clonagem de dados de testes semelhantes aos de produção, provisionamento sob demanda de dados de testes e conformidade com regulamentações de privacidade (por exemplo, Regulamento Geral sobre a Proteção de Dados (RGPD))
13. Gerenciamento de teste: Organize e controle o processo de teste gerenciando casos de teste, planos de teste, resultados do teste e rastreabilidade
    Recursos: criação de casos de teste e controle de versão, criação e agendamento de planos de teste, rastreabilidade entre requisitos, casos de teste e defeitos, relatórios e painéis em tempo real, e integração com pipelines de CI/CD e sistemas de rastreamento de defeitos (ver (ISTQB® CTFL))
14. Colaboração e comunicação: Aumenta a produtividade e o alinhamento da equipe por meio de ferramentas simplificadas de comunicação e gerenciamento de projetos
    Recursos: mensagens em tempo real, gerenciamento de projetos, compartilhamento de documentos e controle de versões
15. Qualidade e confiabilidade das operações: oferece suporte a ambientes de produção, garantindo a confiabilidade operacional por meio de backup, restauração e gerenciamento de incidentes (Axelos, 2019) Recursos: backup de dados, recuperação, gerenciamento de incidentes e mapeamento de vulnerabilidades

Essas categorias formam um framework organizado dentro do pipeline de CI/CD, com cada ferramenta servindo a um propósito específico para criar um ambiente de desenvolvimento e operações contínuo, com alta confiabilidade e eficiência. Alinhadas com o fluxo de trabalho de DevOps, cada categoria se integra

© International Software Testing Qualifications Board

perfeitamente ao ciclo de DevOps, promovendo ciclos de feedback para melhoria contínua, desde o desenvolvimento até a implantação e a produção.

4.1.2 Ferramentas de suporte à garantia da qualidade (QA) e testes

Em um ambiente DevOps, as ferramentas que dão suporte à QA e aos testes são fundamentais para entregar software confiável, de alto desempenho e seguro em um ritmo acelerado. Essas ferramentas se integram perfeitamente ao pipeline de CI/CD, incorporando qualidade em todas as fases do SDLC (ver Seção 2.2). Automatizar e otimizar várias atividades de teste permite que as equipes de DevOps detectem defeitos antecipadamente, mantenham a qualidade do código e forneçam informações sobre a qualidade do software. As ferramentas podem ser agrupadas por função.

Durante as fases de codificação e integração, as ferramentas se concentram em garantir o controle e a integridade da base de código e de seus componentes (ver Seção 4.2.8).

- Ferramentas de análise estática e de qualidade de código avaliam o código-fonte sem execução, ajudando a aplicar padrões de codificação, detectar possíveis defeitos e identificar indícios de código duvidoso ou vulnerabilidades de segurança.
- As ferramentas de testes de unidade verificam unidades individuais de código, fornecendo feedback rápido aos desenvolvedores.
- As ferramentas de mapeamento de dependências analisam bibliotecas de terceiros em busca de riscos de segurança, pacotes desatualizados ou problemas de licença, ajudando a manter uma base de código segura e em conformidade.
- Ferramentas de testes de integração e de API verificam se os componentes ou serviços conectados funcionam juntos corretamente e atendem às expectativas de interface.

À medida que o trabalho no software se aproxima da implantação, ferramentas projetadas para testes funcionais e de interface do usuário, testes de performance, testes de segurança e gerenciamento de dados de teste passam a ter prioridade.

- Ferramentas de testes funcionais e de interface do usuário simulam o comportamento do usuário para validar se o software atende aos requisitos funcionais da perspectiva do usuário final.
- Ferramentas de teste de performance avaliam o comportamento de um sistema sob diferentes condições de carga, identificam gargalos de performance e garantem a escalabilidade.
- Ferramentas de teste de segurança detectam proativamente vulnerabilidades, como defeitos de injeção ou configurações incorretas, reduzindo a exposição a ameaças.
- Ferramentas de gerenciamento de dados de teste geram e gerenciam dados de teste realistas, anônimos ou sintéticos, garantindo que as condições de teste reflitam as condições do mundo real sem violar a privacidade dos dados.

Frameworks e ferramentas que oferecem suporte à automação de testes, gerenciamento de teste e provisionamento de ambiente desempenham um papel vital na viabilização de testes eficazes. (ver (ISTQB® CTFL) seção 6.1 Suporte de Ferramentas para Testes)

- Ferramentas de provisionamento de ambiente automatizam a criação de ambientes de teste que espelham as configurações de produção, apoiando a execução confiável e consistente dos testes.
- As ferramentas de gerenciamento de configuração garantem a consistência nas configurações do ambiente e nas configurações das aplicações em ambientes de desenvolvimento, teste e produção. (ver (ISTQB® CTFL), seção 5.4. Gerenciamento de Configuração) Essa consistência reduz erros e aumenta a precisão dos resultados dos testes, apoiando processos de teste confiáveis e repetíveis.

© International Software Testing Qualifications Board

Juntas, essas ferramentas apoiam as equipes de DevOps na obtenção e manutenção da qualidade exigida.

4.2 Tecnologias e práticas para apoiar a qualidade em DevOps

As tecnologias e práticas de DevOps promovem a qualidade por meio da integração de atividades não relacionadas a testes, estratégias eficazes de ramificação, automação de infraestrutura, gerenciamento de lançamentos e ativação/desativação de recursos. Atividades não relacionadas a testes, como compilação, empacotamento e implantação, automatizam etapas-chave no pipeline de CI/CD, enquanto estratégias de lançamento, como lançamentos canários e implantações blue-green, controlam o processo de implementação. O IaC permite a consistência do ambiente, a ativação/desativação de recursos separa o lançamento da implantação, e o gerenciamento de lançamentos e as estratégias de ramificação otimizam a colaboração. Essas tecnologias e práticas criam um ambiente coeso que garante consistência, confiabilidade, segurança e velocidade na entrega de software.

4.2.1 Atividades não relacionadas a testes dentro de um pipeline de CI/CD

Um pipeline de CI/CD automatiza vários processos no SDLC. Dividir esses processos em várias etapas sequenciais garante uma entrega suave e eficiente. Cada etapa possui tarefas específicas envolvendo atividades não relacionadas a testes, essenciais para o sucesso do pipeline de CI/CD.

Aqui está uma lista dessas atividades não relacionadas a testes: (ver Seção 2.1.4 e Seção 3.1.1)

1. Gerenciamento de código-fonte: supervisionar commits de código e controle de versão dentro de um repositório compartilhado para manter a colaboração e a organização.
2. Ramificação e merge: Realização de tarefas como ramificação, combinação e criação de pull requests (PRs) para organizar e realizar a revisão com eficiência das alterações no código.
3. Gerenciamento de artefatos de compilação e dependências: (ver Seção 4.2.8) Resolução de dependências de software, compilação de código e geração de artefatos de compilação versionados. Esses artefatos de compilação são armazenados para etapas posteriores no pipeline de CI/CD.
4. Resolução de conflitos e padrões de codificação: (ver Seção 3.1.6 e Seção 4.2.3) Resolver conflitos de código, provisionar ambientes de integração e executar análises estáticas para aplicar padrões de codificação e garantir a qualidade do código.
5. Configuração do pipeline de CI/CD: Configurar pipelines de CI/CD e automatizar o provisionamento de ambientes isolados e consistentes, o que geralmente é feito por meio de ferramentas de conteinerização. Além disso, gerenciar configurações específicas do ambiente e ativação/desativação de recursos por meio de argumentos de linha de comando ou variáveis de ambiente, para oferecer flexibilidade e escalabilidade (ver Seção 4.2.4).
6. Gerenciamento de lançamentos: (ver Seção 2.2.5) Coordenar e aprovar lançamentos, gerenciar estratégias de rollback (ver Seção 4.2.2) e marcar ou rotular versões. Isso inclui implementar estratégias de implantação, como implantação blue-green ou lançamentos canários.
7. Monitoramento e feedback: uso de ferramentas para monitorar a integridade do sistema e estabelecer ciclos de feedback para melhoria contínua. Isso envolve configurar sistemas de log e telemetria, analisar métricas para avaliar a performance e as tendências da aplicação e configurar alertas para eventos críticos. (ver Seção 4.2.7) Essas atividades fornecem insights críticos sobre a integridade e a estabilidade da aplicação após a implantação em produção.

© International Software Testing Qualifications Board

4.2.2 Principais estratégias de lançamento

As estratégias de lançamento do DevOps definem a entrega controlada, eficiente e confiável de atualizações de software e novos recursos aos usuários. Essas estratégias de lançamento se concentram nos seguintes pontos:

- minimizar riscos ao reduzir o tempo de inatividade, permitir a exposição gradual às mudanças e detectar incidentes de forma proativa (Axelos, 2019)
- otimizar os processos de implantação, aumentando a velocidade de lançamento, eliminando tarefas repetitivas e reduzindo o tempo entre o desenvolvimento e a produção
- proporcionar valor contínuo ao cliente, garantindo uma experiência do usuário ininterrupta e permitindo a rápida resolução de incidentes pelas equipes operacionais (Axelos, 2019)

Compreender essas estratégias de lançamento pode ajudar as organizações a selecionar a estratégia mais adequada com base em suas necessidades específicas e/ou operacionais.

O DevOps possui várias estratégias-chave de lançamento, cada uma com seu próprio objetivo e caso de uso.
Essas estratégias incluem:

1. Lançamentos canários (ver Seção 2.2.4)
2. Implantações blue-green (ver Seção 2.2.4)
3. Lançamentos contínuos: substituição gradual de instâncias do sistema por versões atualizadas, garantindo a disponibilidade contínua do serviço. Comumente usada em sistemas de grande escala, essa abordagem mitiga os riscos de interrupções generalizadas.
4. Lançamento em fases: Implantação de atualizações de forma incremental por região, grupo de usuários ou outros critérios de segmentação. Isso permite o monitoramento e a resolução de problemas em fases.
5. Entrega progressiva: Combinação de elementos de lançamentos canários e implementações em fases, lançando atualizações de forma incremental para usuários ou regiões específicos, a fim de obter validação precisa e coleta de feedback.
6. Implantação de hotfix: um lançamento urgente para resolver defeitos críticos em produção, priorizando a velocidade em detrimento de testes exaustivos.
7. Lançamento oculto (ver Seção 2.2.4): Implantação de recursos na produção, mas marcando-os como inacessíveis aos usuários. Isso permite que as equipes testem e monitorem a performance em condições reais sem afetar a experiência do usuário.

As principais estratégias de lançamento variam de acordo com as necessidades e o contexto específicos de cada organização, já que existem muitas estratégias de lançamento. No entanto, as estratégias mencionadas acima representam as opções mais essenciais alinhadas às práticas modernas de DevOps. Além disso, as organizações frequentemente combinam várias estratégias de lançamento para otimizar suas operações diárias (por exemplo, lançar um hotfix usando uma implementação em fases).

4.2.3 Infraestrutura como código (IaC)

A IaC gerencia e provisiona a infraestrutura por meio de arquivos de definição legíveis por máquina, tratando a infraestrutura como software. Ela utiliza modelos descritivos e código com controle de versão para garantir consistência e repetibilidade. O QA na IaC envolve testes estáticos automatizados, revisões por pares e validação de configuração para determinar o grau de confiabilidade, segurança e conformidade antes da implantação.

© International Software Testing Qualifications Board

Com a IaC, as equipes editam o código-fonte da IaC em vez de fazer alterações diretamente no ambiente de destino. Essa prática integra-se perfeitamente aos fluxos de trabalho de DevOps, permitindo o provisionamento e o gerenciamento automatizados, garantindo consistência, escalabilidade e eficiência. A IaC dá suporte ao CD ao possibilitar ambientes de infraestrutura reproduzíveis e previsíveis (ver Seção 3.1.6).

O IaC se baseia em vários conceitos centrais para oferecer seus benefícios de forma eficaz. Primeiro, o IaC usa abordagens declarativas e imperativas para definir a infraestrutura. A abordagem declarativa se concentra em especificar o estado desejado da infraestrutura (por exemplo, garantir que três servidores estejam sempre em execução), enquanto a abordagem imperativa fornece instruções detalhadas sobre como alcançar esse estado (por exemplo, instalar software e definir configurações).

Outro conceito crítico é a idempotência (Kundu, 2019), que garante que os scripts de IaC produzam os mesmos resultados mesmo quando aplicados várias vezes, evitando desvios de configuração (Moustakis, 2024). A IaC integra-se perfeitamente com sistemas de gerenciamento de configuração, permitindo que as equipes rastreiem alterações, colaborem de forma eficiente e revertam para configurações anteriores, se necessário (ver Seção 3.1.1 e Seção 4.1.1). A IaC elimina a necessidade de tarefas manuais repetitivas por meio da automação, garantindo consistência e reduzindo erros humanos. Ela oferece suporte à paridade de ambiente, mantendo configurações idênticas nos ambientes de desenvolvimento, teste e produção. Isso reduz o número de situações do tipo “funciona na minha máquina” e permite transições mais suaves entre os níveis de teste.

(H0) Resumir os conceitos de Infraestrutura como Código (IaC)

Para este objetivo prático, o exercício de nível H0 auxilia os alunos na compreensão do conceito de IaC.

4.2.4 Feature Toggles

O uso de feature toggles pode ajudar as equipes de DevOps a modificar o comportamento do sistema para entregar novas funcionalidades aos usuários de forma rápida, mas segura, sem alterar o código (Fowler, 2017). Ele oferece uma abordagem estruturada para gerenciar a implementação de funcionalidades, permitindo lançamentos incrementais, mitigação de risco e experimentação controlada. Em um ambiente de produção, os feature toggles definem quais usuários ou ambientes podem acessar um recurso específico. Por exemplo, um recurso pode ser desativado para a maioria dos usuários enquanto é ativado para testadores, garantindo uma validação controlada antes de uma implantação mais ampla. O uso de feature toggles apoia a separação entre implantação e lançamento.

Os feature toggles são categorizados de acordo com o uso pretendido:

- Chaves de lançamento: ativam novos recursos de acordo com as metas de lançamento sob demanda (por exemplo, um lançamento canário); consulte Seção 2.2.4
- Chaves operacionais: permitem o controle em tempo real do comportamento do sistema
- Chaves de experimentação: oferecem suporte a testes A/B, expondo recursos distintos a segmentos de usuários
- Chaves de permissão: controlam o acesso a recursos para usuários ou funções específicas

Os feature toggles introduzem complexidade. Gerenciar feature toggles requer acompanhamento e organização sistemáticos para evitar o acúmulo excessivo de feature toggles ao longo do tempo. Como os feature toggles adicionam lógica condicional à base de código, é essencial monitorar, documentar e desativar feature toggles obsoletos para manter a clareza do sistema. As equipes de DevOps devem testar as implementações de feature toggles com as configurações ativadas e desativadas e testar recursos interdependentes para reduzir riscos.

© International Software Testing Qualifications Board

4.2.5 Estratégias de ramificação

A ramificação permite o desenvolvimento paralelo dentro e entre equipes de DevOps, proporcionando múltiplos fluxos de trabalho simultaneamente, minimizando conflitos e defeitos de integração. A integração dessas alterações de código de um ramo para outro é chamada de merge. Um conflito de merge ocorre quando ambos os ramos modificam a mesma seção do código. As chances de conflitos de merge dependem de múltiplos fatores (por exemplo, o tamanho das alterações a serem integradas ou o número de desenvolvedores que introduzem alterações). Resolver esses conflitos pode exigir muito tempo e esforço.

Embora os VCSs ofereçam suporte à ramificação, merge e detecção de conflitos, as equipes de DevOps devem implementar estratégias de ramificação adequadas para reduzir o risco de conflitos, garantindo ao mesmo tempo a colaboração e a qualidade.

Como todo o código e o testware que contribuem para o desenvolvimento de software estão em um VCS (ver Seção 3.1.1), as estratégias de ramificação também se aplicam a ele e devem ser compreendidas por todos os membros da equipe de DevOps.

Entre as múltiplas estratégias de ramificação definidas, as equipes de DevOps geralmente aplicam as duas a seguir para alcançar a capacidade de CD:

- Desenvolvimento baseado no trunk: Após testes bem-sucedidos durante a fase de commit, todas as alterações são enviadas diretamente para o ramo principal (ou seja, anteriormente chamado de trunk ou master). Essa abordagem requer uma equipe madura e um alto nível de automação de testes dentro do pipeline de CI/CD.
- Ramos de recurso: são criados para alterações específicas e excluídos após um merge. Um PR é usado para incluir alterações ao ramo principal ou para obter feedback de outros membros da equipe de DevOps durante o desenvolvimento (ver Seção 2.1.4). Esses ramos de recurso devem ter uma vida útil curta, no máximo alguns dias, com o objetivo de reduzi-la para poucas horas.

4.2.6 Engenharia do Caos

A engenharia do caos é uma disciplina que envolve a realização de experimentos em um sistema para construir confiança em sua capacidade de resistir a condições turbulentas em produção (Ranganathan, 2023). Essa prática é crucial para determinar a resiliência e a confiabilidade de sistemas de software, injetando falhas intencionalmente e observando como o sistema responde.

A engenharia do caos realiza simulações de falhas em vários níveis, variando de instâncias de computação individuais (por exemplo, máquinas virtuais ou contêineres executando aplicativos) a regiões geográficas inteiras de infraestrutura em nuvem, onde vários data centers operam em conjunto (Netflix, 2025). Essas experiências ajudam as equipes a compreender como os serviços se comportam durante interrupções e a determinar a continuidade do serviço. Os insights obtidos a partir dessas experiências levam a melhorias no projeto do sistema, nas práticas operacionais e nas estratégias de resposta a incidentes.

A engenharia do caos enfatiza a importância de compreender o comportamento do sistema sob estresse. As equipes de DevOps podem descobrir defeitos ocultos que podem não ser aparentes durante a operação normal ou em um ambiente de teste. Essa disciplina proativa identifica possíveis fraquezas antes que elas afetem os usuários, aprimorando a qualidade do software.

A engenharia do caos é viabilizada por um pipeline de CI/CD altamente eficaz, rápido e com alta confiabilidade, que permite a implantação de alterações em produção e rollbacks sem complicações, minimizando o impacto sobre os usuários. Os provedores de nuvem pública geralmente oferecem ferramentas de injeção de falha para apoiar a engenharia do caos de maneira controlada.

© International Software Testing Qualifications Board

4.2.7 Telemetria e Observabilidade

Quando o software é lançado no ambiente de produção, o monitoramento contínuo coleta informações sobre a qualidade do software e, quando a qualidade não está evoluindo conforme o esperado, ações de controle são iniciadas. O monitoramento contínuo observa o software em seu ambiente, incluindo outros softwares, hardware e as pessoas envolvidas. O monitoramento contínuo em muitas organizações começa no ambiente de produção, mas pode ser relevante para qualquer outro ambiente, especialmente ambientes de teste. A telemetria (coleta de dados) e a observabilidade (análise de dados) possibilitam o monitoramento contínuo (que verifica em relação aos KPIs).

A observabilidade utiliza registros, rastreamento e métricas para fornecer informações e criar uma visão coerente do estado do software. Essas informações são utilizadas para fins de manutenção reativa e proativa, auditoria, controlabilidade e depuração (Marselis et al., 2020). As informações são coletadas automaticamente por meio da telemetria. A telemetria coleta medições de características de qualidade funcionais e não funcionais, que atendem a diferentes propósitos e objetivos.

Exemplos de telemetria são:

- A telemetria de produção realiza medições sobre como o software é executado, especialmente quando se utiliza infraestrutura em nuvem (por exemplo, para medir o uso de memória, a carga da CPU, a latência e a contagem de solicitações).
- A telemetria do usuário realiza a medição da interação do usuário, o que fornece informações essenciais sobre o sucesso do software (por exemplo, rastreamento de cliques para uso de recursos, rastreamento da taxa de preenchimento de formulários, análise do fluxo de navegação e monitoramento da duração da sessão).
- A telemetria de endpoint e outras abordagens de monitoramento de segurança permitem a detecção de ameaças, a análise forense e a geração de relatórios de conformidade.

O monitoramento (ou seja, o uso de telemetria e observabilidade) pode aplicar inteligência artificial (IA), como machine learning, aos dados de log, para compreender melhor e mais facilmente o comportamento do software e detectar quaisquer anomalias.

Observe que a telemetria da equipe pode ser usada para realizar a medição da maturidade da equipe de DevOps (ver Seção 1.1.3 para métricas DORA) e sugerir como a equipe de DevOps pode melhorar (ver o final da Seção 2.2.5 para obter mais informações sobre o aprendizado da equipe).

4.2.8 Lista de Materiais de Software (SBOM)

Ao desenvolver software, as equipes de DevOps frequentemente utilizam componentes de terceiros. Isso inclui frameworks, middleware e bibliotecas que fornecem funcionalidades específicas, além de código de outras equipes de DevOps. A reutilização acelera o desenvolvimento e promove a consistência no desenvolvimento de software dentro das organizações. A reusabilidade (como parte da manutenibilidade) (ver (Systems and software engineering 25010)) é uma característica de qualidade importante e um sinal de componentes construídos adequadamente. Ao reutilizar componentes, é essencial para o gerenciamento do SDLC, a manutenção futura e a transparência ter informações sobre cada componente, incluindo os componentes que a própria equipe de DevOps cria, e as relações entre os componentes.

Os seguintes requisitos são necessários:

- Identificação do componente (por exemplo, nome ou número)
- A versão atual do componente
- Autor e/ou proprietário do componente

© International Software Testing Qualifications Board

- A funcionalidade do componente
- Consistência do componente com outros componentes
- Dependências do componente em relação a outros componentes
- Conflitos da versão atual com versões anteriores
- Hash(es) criptográfico(s) para autenticar o(s) componente(s)
- Condições de licenciamento do componente

Essas informações são registradas em um SBOM (Marselis et al., 2020). As equipes de DevOps devem manter um SBOM para seu software usando ferramentas integradas ao pipeline de CI/CD. Para garantir que o SBOM esteja atualizado, ele deve fazer parte da definição de conclusão da equipe de DevOps. Além da manutenibilidade, um SBOM auxilia no gerenciamento de vulnerabilidades, que faz parte da segurança, ao associar vulnerabilidades conhecidas aos componentes utilizados. Sempre que uma vulnerabilidade for detectada, o pipeline de CI/CD será interrompido, para evitar a implantação de software inseguro. Depois disso, a equipe de DevOps pode tomar as medidas necessárias para resolver tais vulnerabilidades.

4.2.9 Containerização

A conteinerização é uma forma leve de virtualização que empacota um aplicativo e suas dependências em uma única unidade executável chamada contêiner.

Os contêineres permitem que a equipe de DevOps construa ambientes consistentes para cada nível de teste e elimine as armadilhas dos testes locais, como no caso de “funciona na minha máquina” (ver Seção 3.1.6). Outro benefício é que permite a execução de vários testes em paralelo sem interferência entre si e mantendo os dados de teste limpos (ver Seção 3.1.4). Um contêiner pode ser implantado sob demanda e removido assim que o ciclo de teste for concluída. Isso ajuda a provisionar ambientes quando eles estão sendo usados ativamente e a fornecer rapidamente ambientes adicionais quando a demanda aumenta. Como os contêineres são definidos e armazenados como código em uma base de código compartilhada, quaisquer alterações de ambiente feitas centralmente podem ser aplicadas automaticamente a ambientes futuros com a mesma configuração.

A containerização traz alguns desafios conhecidos. Como cada contêiner é uma representação limpa do ambiente, lidar com testes de aplicativos com estado pode ser difícil e requer etapas extras de configuração. A segurança pode ser uma preocupação devido ao uso de componentes de terceiros e requer um gerenciamento diligente de dependências (ver Seção 4.2.8).

© International Software Testing Qualifications Board

5 Qualidade em termos específicos de DevOps

Nome do termo	Definição
artefato de compilação	Um produto de trabalho gerado pelo processo de compilação e que pode ser usado para implantação.
implantação blue-green	Uma estratégia de implantação na qual dois ambientes de produção idênticos, um ativo e outro inativo, são mantidos para realizar um teste com uma nova versão de software no ambiente inativo antes de redirecionar todo o tráfego do ambiente ativo.
ramificação	O processo de trabalhar em alterações de software de forma independente, por meio da duplicação do código-fonte sob controle de versão em um ramo relacionado à alteração e da fusão posterior com outro ramo.
estratégia de ramificação	Um conjunto de regras para gerenciar ramificações em sistemas de controle de versão.
CALMS	A sigla que significa cultura, automação, lean, medição e compartilhamento (*culture, automation, lean, measurement e sharing*), usada para estruturar as práticas de DevOps.
percentual de falhas nas alterações	O número de alterações implantadas que resultaram em falhas, dividido pelo número total de alterações implantadas, expresso em porcentagem.
tempo de lead time	O tempo decorrido desde que um commit é feito até que ele chegue à produção.
engenharia do caos	A prática de injetar falhas aleatoriamente para coletar informações sobre a resiliência do sistema.
pipeline de CI/CD	A série automatizada de etapas para entregar uma nova versão de software.
code smell	Uma característica do código-fonte que indica um possível problema de design ou de manutenibilidade, sem necessariamente impedir o funcionamento do código
containerização	Uma forma leve de virtualização que empacota um aplicativo e suas dependências em uma única unidade executável chamada contêiner.
entrega contínua (CD)	Um procedimento automatizado de desenvolvimento de software no qual as alterações no código são automaticamente compiladas, testadas e podem ser implantadas em produção.

© International Software Testing Qualifications Board

Nome do termo	Definição
implantação contínua	Um procedimento automatizado de lançamento de software em que as alterações de código que passaram em todos os testes são automaticamente implantadas em produção sem intervenção manual.
descoberta contínua	Um processo contínuo de pesquisa com usuários e feedback dos stakeholders para tomar decisões informadas, melhorar o produto e garantir a entrega de valor real aos usuários.
Nome do termo	Definição
aprendizado e experimentação contínuos	Um princípio do DevOps que promove uma cultura de alta confiança e melhoria contínua, incentivando as equipes a experimentar, assumir riscos calculados e aprender com os sucessos e as falhas.
monitoramento contínuo	Um processo automatizado de coleta de indicadores de performance para compreender o comportamento do usuário e do sistema em tempo real.
equipe multifuncional de DevOps	Um grupo de pessoas com conjuntos de conhecimentos, habilidades e capacidades diferentes, mas que se complementam, trabalhando em conjunto em direção a um objetivo comum para criar e operar um sistema.
lançamento oculto	Uma estratégia de implantação na qual um novo recurso ou alteração de software é implantado sem ser exposto aos usuários, permitindo que as equipes testem e monitorem a performance em produção.
mapeamento de dependências	O processo de identificar e analisar dependências de software para reduzir riscos introduzidos por componentes externos.
frequência de implantação	A frequência de lançamentos de software em produção.
DevOps	Uma mentalidade, cultura e conjunto de práticas técnicas que apoiam a integração, a automação e a colaboração necessárias para desenvolver e operar um sistema de forma eficaz.
tempo de recuperação após falha na implantação	O tempo necessário para restaurar a operabilidade normal após uma implantação que falhou.
Feature Toggles	Um mecanismo para modificar o comportamento do sistema sem alterar o código.
ciclo de feedback	A troca contínua de informações para melhorar produtos, serviços e processos.
fluxo	O movimento suave, linear e rápido dos produtos de trabalho de uma etapa para outra em um fluxo de valor relevante.
idempotência	A propriedade de uma operação que produz o mesmo resultado, seja executada uma ou várias vezes.

© International Software Testing Qualifications Board

Nome do termo	Definição
infraestrutura como código (IaC - infrastructure as code)	A prática de gerenciar e provisionar infraestrutura de TI usando arquivos de configuração legíveis por máquina e aplicando práticas de desenvolvimento de software a ela.
merge	A prática de integrar alterações de código de uma ramificação para outra.
observabilidade	A capacidade de realizar a medição do estado interno de um sistema por meio da análise de sua saída.
superprovisionamento	A prática de alocar mais recursos de computação do que o necessário a um componente ou sistema para melhorar a performance ou a confiabilidade.
informações de identificação pessoal (PII - personally identifiable information)	Qualquer informação relacionada a um indivíduo específico que possa ser usada para revelar a identidade desse indivíduo
pull request (PR)	Um mecanismo que permite notificar os membros da equipe sobre alterações feitas em uma base de código, propondo que essas alterações sejam submetidas a revisão, discutidas e integradas.
gestão de lançamentos	Um processo de planejamento e programação de lançamentos, incluindo testes e implantação de software.
lançamento sob demanda	Uma prática de lançar novos recursos com base nas necessidades dos stakeholders, separando a implantação do lançamento.
estratégia de lançamento	Uma estratégia para determinar como implantar o software em produção, como lançá-lo para os usuários e com que frequência.
acordo de nível de serviço (SLA - service level agreement)	Um acordo com um cliente de que uma meta de nível de serviço será cumprida durante um determinado período.
indicador de nível de serviço (SLI - service level indicator)	Uma medição quantificável da confiabilidade do serviço
objetivo de nível de serviço (SLO - service level objective)	Uma meta de confiabilidade para um indicador de nível de serviço
fonte única de verdade (SSOT - single source of truth)	Uma prática de normalização de dados que utiliza uma única fonte para um elemento de dados específico.
engenharia de confiabilidade do site (SRE - site reliability engineering)	Uma abordagem que aplica práticas e princípios de desenvolvimento de software à infraestrutura e às operações, equilibrando a confiabilidade do serviço com o ritmo de desenvolvimento de novos recursos por meio de ciclos de feedback automatizados.

© International Software Testing Qualifications Board

Nome do termo	Definição
lista de materiais de software (SBOM - software bill of materials)	Um registro legível por máquina que detalha os componentes de software e suas relações na cadeia de suprimentos, garantindo transparência, auditabilidade e rastreabilidade.
binário de software	O arquivo resultante da compilação do código-fonte escrito em uma linguagem compilada.
gestão de código-fonte (SCM - source code management)	A prática de rastrear, gerenciar e armazenar alterações no software, permitindo o controle de versões, o desenvolvimento paralelo e a resolução de conflitos entre colaboradores.
análise estatística	Técnica para coletar, analisar, interpretar, apresentar e extrair conclusões a partir de dados.
telemetria	Coleta, agregação e análise automatizadas de métricas de aplicativos e infraestrutura em produção para monitorar a integridade do sistema e auxiliar na identificação de defeitos.
subprovisionamento	A prática de alocar menos recursos de computação do que o necessário a um componente ou sistema, levando à degradação da performance e a possíveis falhas do sistema.
fluxo de valor	A sequência completa de atividades requisitadas para entregar um produto ou serviço a um cliente, incluindo o fluxo de informações e materiais.
controle de versão	Um processo que registra as alterações nos itens de configuração ao longo do tempo.
sistema de controle de versão (VCS - version control system)	Uma ferramenta de software que automatiza o controle de versão.
virtualização	Técnica que permite a prestação de serviços virtuais que são implantados, acessados e gerenciados remotamente.
barreira de confusão	A falta de comunicação entre as equipes de desenvolvimento e operações dentro de uma organização.

© International Software Testing Qualifications Board

6 Marcas registradas

Liste as marcas registradas mencionadas neste syllabus em ordem alfabética.

DASA® é uma marca registrada da DevOps Agile Skills Association.
DORA® é uma marca registrada da DevOps Research and Assessment.
ISTQB® é uma marca registrada do International Software Testing Qualifications Board.
TMMi® é uma marca registrada da TMMi Foundation.

© International Software Testing Qualifications Board

7 Objetivos de Aprendizagem/Nível Cognitivo de Conhecimento

Os objetivos de aprendizagem específicos aplicáveis a este syllabus são apresentados no início de cada capítulo. Cada tópico do syllabus será analisado de acordo com o respectivo objetivo de aprendizagem.

Os objetivos de aprendizagem começam com um verbo de ação correspondente ao seu nível cognitivo de conhecimento, conforme listado abaixo.

Nível 1: Lembrar (K1)

O candidato irá lembrar, reconhecer e recordar um termo ou conceito.

Verbos de ação: Recordar, reconhecer.

Exemplos
Recordar os conceitos da pirâmide de teste.
Reconhecer os objetivos típicos dos testes.

Nível 2: Compreender (K2)

O candidato é capaz de selecionar as razões ou explicações para afirmações relacionadas ao tema e pode resumir, comparar, classificar e dar exemplos para o conceito de teste.

Verbos de ação: Classificar, comparar, diferenciar, distinguir, explicar, dar exemplos, interpretar, resumir

Exemplos	**Notas**
Classifique as ferramentas de teste de acordo com sua finalidade e as atividades de teste que elas suportam.	
Comparar os diferentes níveis de teste.	Pode ser usado para procurar semelhanças, diferenças ou ambos.
Diferenciar testes de depuração.	Procure diferenças entre conceitos.
Distinguir entre riscos de projeto e riscos de produto.	Permite que dois (ou mais) conceitos sejam classificados separadamente.
Explique o impacto do contexto no processo de teste.	
Dê exemplos de por que o teste é necessário.	
Inferir a causa-raiz dos defeitos a partir de um determinado perfil de falhas.	
Resuma as atividades do processo de revisão do produto de trabalho.	

© International Software Testing Qualifications Board

Nível 3: Aplicar (K3)

O candidato é capaz de executar um procedimento quando confrontado com uma tarefa familiar ou selecionar o procedimento correto e aplicá-lo a um determinado contexto.

Verbos de ação: Aplicar, implementar, preparar, usar

Exemplos	Notas
Aplicar a análise de valor limite para derivar casos de teste a partir de requisitos dados.	Deve consultar um procedimento/técnica/processo etc.
Implementar métodos de coleta de métricas para atender aos requisitos técnicos e de gestão.	
Prepare testes de instalabilidade para aplicativos móveis.	
Use a rastreabilidade para monitorar o progresso do teste quanto à integridade e consistência com os objetivos do teste, a estratégia de teste e o plano de teste.	Pode ser usado em um LO que deseja que o candidato seja capaz de usar uma técnica ou procedimento. Semelhante a “aplicar”.

Os níveis cognitivos dos objetivos de aprendizagem baseiam-se em (Anderson et al., 2001).

© International Software Testing Qualifications Board

8 Apêndice B – Matriz de rastreabilidade dos Resultados de Negócios com Objetivos de Aprendizagem

Esta seção lista a rastreabilidade entre os Resultados de Negócios e os Objetivos de Aprendizagem de Qualidade em DevOps.

Resultados de Negócios: Qualidade em DevOps		QDO-BO1	QDO-BO2	QDO-BO3	QDO-BO4	QDO-BO5	QDO-BO6	QDO-BO7	QDO-BO8	QDO-BO9
QDO-BO1	Explique como a garantia da qualidade é apoiada pelos conceitos de DevOps e como contribui para eles	6								
QDO-BO2	Compreender os conceitos de implementação do DevOps em uma organização		3							
QDO-BO3	Contribuir para todas as etapas do fluxo de valor para auxiliar na engenharia da qualidade			4						
QDO-BO4	Contribuir para a implementação do ciclo DevOps				6					
QDO-BO5	Descrever como a automação apoia a garantia da qualidade no DevOps					6				
QDO-BO6	Implementar a automação de testes em equipes de DevOps						3			
QDO-BO7	Implementação de testes manuais em equipes de DevOps							4		
QDO-BO8	Selecionar ferramentas que ofereçam suporte ao DevOps dentro da organização								2	
QDO-BO9	Compreender as tecnologias e práticas para apoiar a qualidade no DevOps									9

© International Software Testing Qualifications Board

Resultados de negócio: Quality in DevOps		QDO-BO1	QDO-BO2	QDO-BO3	QDO-BO4	QDO-BO5	QDO-BO6	QDO-BO7	QDO-BO8	QDO-BO9
Número do LO	Objetivo de Aprendizagem (Nível K)									
1	Fundamentos do DevOps – 140 minutos									
1.1	Explicar como a garantia da qualidade é apoiada pelos conceitos de DevOps e como contribui para eles									
QDO-1.1.1	Recordar os conceitos-chave do DevOps (K1)	X								
QDO-1.1.2	Recordar os conceitos de muro da confusão no DevOps (K1)	X								
QDO-1.1.3	Explicar as métricas DORA de performance de entrega e performance operacional (K2)	X								
QDO-1.1.4	Diferenciar os elementos do CALMS em termos de garantia da qualidade (K2)	X								
QDO-1.1.5	Explicar como a garantia da qualidade apoia as três vertentes do DevOps (K2)	X								
QDO-1.1.6	Explique os benefícios, riscos e armadilhas do DevOps (K2)	X								
1.2	Compreender os conceitos da implementação do DevOps em uma organização									
QDO-1.2.1	Distinguir as funções da equipe de DevOps (K2)		X							
QDO-1.2.2	Explicar o conceito de engenharia de confiabilidade do site (SRE) relacionado ao DevOps (K2)		X							
QDO-1.2.3	Distinguir os diferentes padrões e antipadrões de equipes de DevOps (K2)		X							
2	Garantia da Qualidade (QA) e Teste em DevOps — 360 minutos									
2.1	Contribuir para todas as etapas do fluxo de valor para auxiliar na engenharia de qualidade da qualidade									
QDO-2.1.1	Implementar atividades de garantia da qualidade no contexto do DevOps (K3)			X						

© International Software Testing Qualifications Board

Resultados de negócio: Quality in DevOps		QDO-BO1	QDO-BO2	QDO-BO3	QDO-BO4	QDO-BO5	QDO-BO6	QDO-BO7	QDO-BO8	QDO-BO9
QDO-2.1.2	Comparar os objetivos do teste que apoiam o DevOps com modelos de desenvolvimento sequenciais e o desenvolvimento ágil de software (K2)			X						
QDO-2.1.3	Explique o que são os testes contínuos no contexto de DevOps (K2)			X						
QDO-2.1.4	Explicar como as solicitações de pull contribuem para a qualidade (K2)			X						
2.2	Contribuir para a implementação do ciclo de DevOps									
QDO-2.2.1	Explicar a garantia da qualidade e os testes na descoberta contínua e suas práticas (K2)				X					
QDO-2.2.2	Explicar a garantia da qualidade e os testes na integração contínua e suas práticas (K2)				X					
QDO-2.2.3	Explicar a garantia da qualidade e os testes na entrega contínua e suas práticas (K2)				X					
QDO-2.2.4	Explicar a garantia da qualidade e os testes na implantação contínua e suas práticas (K2)				X					
QDO-2.2.5	Explicar a garantia da qualidade e os testes no lançamento sob demanda e suas práticas (K2)				X					
QDO-2.2.6	Aplicar conhecimentos de garantia da qualidade e testes para implementar um pipeline de CI/CD (K3)				X					
3	Tarefas automatizadas e manuais em DevOps — 510 minutos									
3.1	Descrever como a automação apoia a garantia da qualidade em DevOps									
QDO-3.1.1	Explicar como uma fonte única de verdade para o testware apoia a garantia da qualidade (K2)					X				
QDO-3.1.2	Explique como a automação apoia a rastreabilidade entre a base de teste e o testware (K2)					X				
QDO-3.1.3	Implementar práticas uniformes de relatórios de qualidade no pipeline de CI/CD com informações provenientes tanto de testes automatizados quanto de testes manuais (K3)					X				

© International Software Testing Qualifications Board

Resultados de negócio: Quality in DevOps		QDO-BO1	QDO-BO2	QDO-BO3	QDO-BO4	QDO-BO5	QDO-BO6	QDO-BO7	QDO-BO8	QDO-BO9
QDO-3.1.4	Explicar os benefícios da automação do gerenciamento de dados de teste (K2)					X				
QDO-3.1.5	Explicar os benefícios da análise estatística dos resultados dos testes ao longo de longos períodos (K2)					X				
QDO-3.1.6	Explicar os benefícios do gerenciamento padronizado, controlado e automatizado do ambiente de teste (K2)					X				
3.2	Implementação de testes automatizados em equipes de DevOps									
QDO-3.2.1	Explicar como os testes de regressão se integram às diversas atividades do pipeline de CI/CD (K2)						X			
QDO-3.2.2	Preparar testes de API (K3)						X			
QDO-3.2.3	Comparar o grau de automação de testes no DevOps com o desenvolvimento ágil de software e os modelos de desenvolvimento sequencial (K2)						X			
3.3	Implementar testes manuais em equipes de DevOps									
QDO-3.3.1	Apresente exemplos de testes manuais para uma organização de DevOps (K2)							X		
QDO-3.3.2	Explicar como os testes exploratórios podem funcionar com um pipeline de CI/CD (K2)							X		
QDO-3.3.3	Explique como o teste colaborativo pode apoiar a cultura de feedback (K2)							X		
QDO-3.3.4	Aplicar eventos de busca pela qualidade para apoiar a cultura de aprendizagem (K3)							X		
4	Ferramentas e práticas em DevOps — 200 minutos									
4.1	Selecionar ferramentas que apoiem o DevOps dentro da organização									
QDO-4.1.1	Recordar os recursos das ferramentas em DevOps (K1)								X	
QDO-4.1.2	Explicar as ferramentas que dão suporte à garantia da qualidade e aos testes no DevOps (K2)								X	

© International Software Testing Qualifications Board

Resultados de negócio: Quality in DevOps		QDO-BO1	QDO-BO2	QDO-BO3	QDO-BO4	QDO-BO5	QDO-BO6	QDO-BO7	QDO-BO8	QDO-BO9
4.2	Compreender as tecnologias e práticas que apoiam a qualidade no DevOps									
QDO-4.2.1	Relatar as atividades não relacionadas a testes no pipeline de CI/CD (K1)									X
QDO-4.2.2	Explicar as diferentes estratégias-chave de lançamento no DevOps (K2)									X
QDO-4.2.3	Resumir os conceitos de infraestrutura como código (K2)									X
QDO-4.2.4	Recordar os “feature toggles” como uma forma de separar a implantação do lançamento (K1)									X
QDO-4.2.5	Revisão das estratégias de ramificação (K1)									X
QDO-4.2.6	Recordar a engenharia do caos (K1)									X
QDO-4.2.7	Recorde telemetria e observabilidade (K1)									X
QDO-4.2.8	Recuperar a lista de materiais de software para apoiar o gerenciamento de configuração (K1)									X
QDO-4.2.9	Recuperar a conteinerização (K1)									X

© International Software Testing Qualifications Board

9 Apêndice C – Notas de lançamento

Este syllabus é a primeira versão do syllabus do Certificado de Testador de Qualidade em DevOps (CT-QDO) do International Software Testing Qualification Board.

10 Referências

10.1 Normas

Systems and software engineering (25010): Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model, 2023. 2023-01. Standard. International Organization for Standardization.

Documentos ISTQB®

ISTQB® CT-AT: Agile Tester Syllabus, 2014. Version 1.1.

ISTQB® CT-ATLaS: Agile Test Leadership at Scale Syllabus, 2023. Version 2.0.

ISTQB® CT-ATT: Agile Technical Tester Syllabus, 2019. Version 1.1.

ISTQB® CT-MAT: Mobile Application Testing Syllabus, 2019. Version 2019.

ISTQB® CT-PT: Performance Testing Syllabus, 2018. Version 2018.

ISTQB® CT-SEC: Security Testing Syllabus, 2016. Version 1.0.

ISTQB® CT-TAS: Test Automation Strategy Syllabus, 2024. Version 1.0.

ISTQB® CTAL-TAE: Advanced Level Test Automation Engineer Syllabus, 2024. Version 2.0.

ISTQB® CTAL-TM: Advanced Level Test Management Syllabus, 2024. Version 3.0.

ISTQB® CTFL: Foundation Level Syllabus, 2023. Version 4.0.

ISTQB® Exam Structures and Rules, 2025. Version 1.2.

ISTQB® Generic Accreditation Guidelines, 2024. Version 2.4.

10.2 Livros

ADZIC, Gojko, 2011. Specification by Example: How Successful Teams Deliver the Right Software. Shelter Island, NY: Manning Publications. isbn 9781935182553.

ANDERSON, L.W.; KRATHWOHL, D.R., 2001. A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. Longman. isbn 9780801319037.

AXELOS, 2019. ITIL Foundation: ITIL 4 Edition. London, UK: TSO (The Stationery Office). ISBN: 978 0113316076.

BECK, Kent, 2000. Extreme Programming Explained: Embrace Change. Addison-Wesley.

BEYER, Betsy; JONES, Chris; PETOFF, Jennifer; MURPHY, Niall Richard, 2016. Site Reliability Engineering: How Google Runs Production Systems. 1st. O'Reilly Media, Inc. isbn 149192912X.

CLOKIE, K., 2017. A Practical Guide to Testing in DevOps. Leanpub. Available also from: https://leanpub.com/testingindevops.

CRISPIN, L.; GREGORY, J., 2008. Agile Testing: A Practical Guide for Testers and Agile Teams. Pearson Education. Addison-Wesley Signature Series (Cohn). isbn 9780321616937. Available also from: https://books.google.hu/books?id=68_IhPvoKS8C.

HENDRICKSON, E., 2013. Explore It!: Reduce Risk and Increase Confidence with Exploratory Testing. Pragmatic Bookshelf. The Pragmatic Bookshelf. isbn 9781937785024. Available also from: https://books.google.nl/books?id=jxqpMQEACAAJ.

HUMBLE, J.; FARLEY, D., 2010. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Pearson Education. Addison-Wesley Signature Series (Fowler). isbn 9780321670229. Available also from: https://books.google.hu/books?id=6ADDuzere-YC.

© International Software Testing Qualifications Board

HUMBLE, J.; MOLESKY, J.; O'REILLY, B., 2020. Lean Enterprise. O'Reilly Media. isbn 9781492092223. Available also from: https://books.google.hu/books?id=BmbyDwAAQBAJ.

KIM, G.; BEHR, K.; SPAFFORD, G., 2018. The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win. IT Revolution Press. isbn 9781942788300. Available also from: https://books.google.hu/books?id=H6x-DwAAQBAJ.

KIM, G.; HUMBLE, J.; DEBOIS, P.; WILLIS, J., 2016. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press. ITpro collection. isbn 9781942788072. Available also from: https://books.google.hu/books?id=ui8hDgAAQBAJ.

LARMAN, Craig; VODDE, Bas, 2016. Large-scale scrum: More with LeSS. Addison-Wesley Professional.

MARSELIS, R.; GEURTS, D.; RUIGROK, W.; VEENENDAAL, B. van, 2020. Quality for DevOps teams. Sogeti Nederland B.V. isbn 9789075414905. Available also from: https://books.google.hu/books?id=ICHXDwAAQBAJ.

RUIGROK, W.; EGELMEERS, J.; MARSELIS, R., 2025. Amplified Quality Engineering. Sogeti Nederland B.V. isbn 9789075414929.

10.3 Artigos

DEBOIS, Patrick, 2011. Devops: A software revolution in the making. Cutter IT Journal. Vol. 24, no. 8, pp. 3–5.

10.4 Páginas da Web

ALLIANCE, Agile, 2026. Agile Glossary and Terminology [online]. Agile Alliance, 2026-01-31 [visited on 2026-01-31]. Available from: https://agilealliance.org/agile101/agile-glossary/.

AMAZON, 2024. What is DevOps? [online]. Amazon, 2024-10-15 [visited on 2024-10-15]. Available from: https://aws.amazon.com/devops/what-is-devops/.

DASA, 2019. Embracing digital disruption by adopting DevOps practices [online]. 2019-01-01. [visited on 2024-07-09]. Available from: https://www.dasa.org/wp-content/uploads/DASA-White-Paper-1/Embracing-Digital-Disruption-1.pdf.

DASA, 2024. Challenges in Breaking The Wall of Confusion: DASA DevOps Fundamentals, Knowledge Bytes [online]. DASA, 2024-07-05 [visited on 2024-08-09]. Available from: https://www.dasa.org/blog/challenges-in-breaking-the-wall-of-confusion/.

DIGITAL.AI, 2023. DevSecOps Tools Periodic Table [online]. [visited on 2024-11-13]. Available from: https://digital.ai/learn/devsecops-periodic-table/.

DORA, 2022. 2022 Accelerate State of DevOps Report [online]. DORA, 2022-12-01 [visited on 2024-08-12]. Available from: https://dora.dev/research/2022/dora-report/2022-dora-accelerate-state-of-devops/report.pdf.

DORA, 2024. DORA's software delivery metrics: the four keys [online]. DORA, 2024-05-30 [visited on 2024-08-12]. Available from: https://dora.dev/guides/dora-metrics-four-keys/.

FELLOWS, Matt, 2022. Pact- Getting started [online]. 2022-08-30. [visited on 2025-06-24]. Available from: https://docs.pact.io/.

FOWLER, Martin, 2017. Feature Toggles [online]. 2017-10-09. [visited on 2017-10-09]. Available from: https://martinfowler.com/articles/feature-toggles.html.

ISTQB, 2025. Glossary [online]. ISTQB, 2025-05-15 [visited on 2025-05-15]. Available from: https://glossary.istqb.org/.

JANET GREGORY, Lisa Crispin, 2023. Holistic Testing [online]. Gregory/Crispin [visited on 2024-09-05]. Available from: https://agiletestingfellow.com/.

© International Software Testing Qualifications Board

KUNDU, Sourav, 2019. Idempotency in Infrastructure as Code [online]. [visited on 2025-01-24]. Available from: https://skundunotes.com/2019/04/19/idempotency-in-infrastructure-as-code/.

MATTHEW SKELTON, Manuel Pais, 2023. What Team Structure is Right for DevOps to Flourish? [online]. Skelton/Pais, 2023-03-21 [visited on 2024-09-03]. Available from: https://web.devopstopologies.com/.

MOUSTAKIS, Ioannis, 2024. Idempotency in Infrastructure as Code [online]. 2024-10-31. [visited on 2025 01-26]. Available from: https://spacelift.io/blog/what-is-configuration-drift.

NETFLIX, 2025. Chaos Engineering [online]. netflixtechblog, 2025-01-05 [visited on 2025-01-05]. Available from: https://netflixtechblog.com/tagged/chaos-engineering.

RANGANATHAN, Rahul, 2023. Building Resilient Systems with Chaos Engineering [online]. Medium, 2023-07-27 [visited on 2025-01-05]. Available from: https://medium.com/google-cloud/building-resilientsystems-with-chaos-engineering-91f5b6adc972.

Scaled Agile Framework: CALMR, 2023a [online]. © Scaled Agile, Inc., 2023-03-14 [visited on 2024-08 12]. Available from: https://scaledagileframework.com/calmr/.

Scaled Agile Framework: Customer Centricity, 2023b [online]. © Scaled Agile, Inc., 2023-03-14 [visited on 2024-10-31]. Available from: https://scaledagileframework.com/customer-centricity/.

Scaled Agile Framework: Design Thinking, 2023c [online]. © Scaled Agile, Inc., 2023-03-14 [visited on 2024-10-31]. Available from: https://scaledagileframework.com/design-thinking/.

TORRES, Teresa, 2024. Blog: Continuous Discovery Framework [online]. https://userpilot.com/, 2024-09-16 [visited on 2024-10-31]. Available from: https://userpilot.com/blog/continuous-discovery-frameworkteresa-torres/#What-is-continuous-discovery?.

VEENENDAAL, Erik van, 2025. TMMi® in the DevOps world [online]. [visited on 2025]. Available from: https://www.tmmi.org/download/tmmi-in-devops/.

As referências anteriores remetem a informações disponíveis na Internet e em outros locais. Embora essas referências tenham sido verificadas no momento da publicação deste syllabus, o ISTQB® não se responsabiliza caso a disponibilidade das referências tenha diminuído.

© International Software Testing Qualifications Board

11 Leituras complementares

ARIOLA, W.; DUNLOP, C., 2014. *Testes contínuos*. CreateSpace Independent Publishing Platform. ISBN 9781494859756. Disponível também com a possibilidade de compra: https://books.google.hu/books?id=MAtcngEACAAJ.

BELMONT, J.M., 2018. *Hands-On Integração Contínua e Entrega Contínua: Construa e release software de qualidade em escala com Jenkins, Travis CI, e CircleCI*. Packt Publishing. ISBN 9781789133073. Disponível também em: https://books.google.hu/books?id=hh9sDwAAQBAJ.

BLANK-EDELMAN, D.N., 2018. *Seeking SRE: Conversas sobre a operação de sistemas de produção em escala*. O'Reilly Media. ISBN 9781491978818. Disponibilidade também em: https://books.google.hu/books?id= tmhqDwAAQBAJ.

BRYKCZYNSKI, Bill, 1992. Uma pesquisa sobre listas de verificação para inspeção de software. *ACM SIGSOFT Software Engineering Notes*. Vol. 24, n.º 1, pp. 82–89.

DUNLOP, C.; PLATZ, W., 2019. *Testes Contínuos Corporativos: Transformando os Testes para Agile e DevOps*. Publicação independente. ISBN 9781699022948. Disponibilidade também em: https://books.google.hu/books?id=ms1ZywEACAAJ.

DUVALL, P.M.; MATYAS, S.; GLOVER, A., 2007. *Integração contínua: melhorando a qualidade do software e reduzindo riscos*. Pearson Education. Série Addison-Wesley Signature. ISBN 9780321630148. Disponível também em: https://books.google.hu/books?id=PV9qfEdv9L0C.

FORSGREN, N.; HUMBLE, J.; KIM, G., 2018. *Accelerate: A ciência do software enxuto e do DevOps: construindo e escalando organizações de tecnologia de alta performance*. IT Revolution Press. ISBN 9781942788355. Disponível também na disponibilidade: https://books.google.hu/books?id=Kax-DwAAQBAJ.

GREGORY, J.; CRISPIN, L., 2019. *Teste Ágeis Resumidos: Uma Breve Introdução*. Leanpub. ISBN 9781999220518. Disponibilidade também em: https://books.google.hu/books?id=yP-yzwEACAAJ.

KIM, G., 2019. *O Projeto Unicórnio: Um romance sobre desenvolvedores, disrupção digital e prosperidade na era dos dados*. IT Revolution Press. ISBN 9781942788775. Disponibilidade também em: https://books.google.hu/books?id=kNSSDwAAQBAJ.

KINSBRUNER, E., 2018. *Testes Contínuos para Profissionais de DevOps: Um Guia Prático de Especialistas do Setor*. CreateSpace Independent Publishing Platform. ISBN 9781727132175. Disponível também em: https://books.google.fi/books?id=_I_ruwEACAAJ.

MURPHY, N.R.; BEYER, B.; JONES, C.; PETOFF, J., 2016. *Engenharia de Confiabilidade de Sistemas: Como o Google opera sistemas de produção*. O'Reilly Media. ISBN 9781491951170. Disponível também em: https://books.google.hu/books?id=tYrPCwAAQBAJ.

NOVOTNÝ, Vít, 2017. Usando Markdown em documentos TEX. *TUGboat*. Vol. 38, n.º 2, pp. 214–217.

ROSSEL, S., 2017. *Integração contínua, Entrega contínua e Implantação contínua: Lançamentos de software confiáveis e mais rápidos com a automação de compilações, testes e implantação*. Packt Publishing. ISBN 9781787284180. Disponível também em: https://books.google.hu/books?id=1xhKDwAAQBAJ.

SCHEAFFER, J.; RAVICHANDRAN, A.; MARTINS, A., 2018. *The Kitty Hawk Venture: Um romance sobre testes contínuos em DevOps para apoiar a entrega contínua e o sucesso empresarial*. Apress. ISBN 9781484236611. Disponível também em: https://books.google.hu/books?id=asRlDwAAQBAJ.

VADAPALLI, S., 2018. *DevOps: Entrega contínua, integração e implantação com DevOps: Mergulhe nas principais estratégias de DevOps*. Packt Publishing. ISBN 9781789131253. Disponível também em: https:

//books.google.hu/books?id=N5RRDwAAQBAJ.

VEENENDAAL, Erik P.W.M. van; WELLS, B., 2012. *Integração do Modelo de Maturidade de Testes (TMMi): Diretrizes para a Melhoria de Processo de Teste*. Tutein Nolthenius. ISBN 9789490986100. Disponível também em: https :

© International Software Testing Qualifications Board

//books.google.fi/books?id=6y8QnQEACAAJ.

ZHAN, C.; ZHAN, Z., 2021. *Testes Contínuos Práticos: Tornando o Agile/DevOps uma Realidade*. CreateSpace Independent Publishing Platform. ISBN 9781507742112. Disponível também em: https://books.google.hu/books?id=6QG1zgEACAAJ.

© International Software Testing Qualifications Board