TESTES DE SOFTWARE E GERÊNCIA DE CONFIGURAÇÃO

Maristela Regina Weinfurter Teixeira

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Ferramentas de gerenciamento de configuração de software

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar a função das ferramentas de gerenciamento de configuração de software.
- Listar as principais ferramentas de gerenciamento de configuração de software.
- Utilizar ferramentas de gerenciamento de configuração de software.

Introdução

Nas últimas décadas, com as inovações tecnológicas e os novos comportamentos sociais, as equipes de desenvolvimento de software têm se tornado multidisciplinares e independentes geograficamente. Nesse cenário, as ferramentas de gerenciamento de configuração de software (GCSs) se destacam, pois automatizam o processo de desenvolvimento de softwares. As GCSs estabelecem e mantêm a integridade e a rastreabilidade de todo o processo de mudanças, contribuindo no monitoramento da qualidade e do ciclo de vida do software.

Neste capítulo, você vai verificar como o uso adequado de ferramentas de GCS agiliza, automatiza e garante a qualidade em todo o processo de desenvolvimento de softwares.

Função

Você já deve ter ouvido falar sobre ferramentas de configuração de software. Elas aparecem muitas vezes em contextos profissionais, mas é comum que não haja muita referência ao modo como se integram e se organizam conceitual-

mente. De acordo com o Swebok elaborado pela IEEE Computer Society, a GCS é bastante alinhada à garantia da qualidade de software (GQS), produzindo conhecimento e informações importantes para todo o gerenciamento do ciclo de vida do software, bem como para a aplicação de melhorias continuadas (BOURQUE; FAIRLEY, 2014). Atingir as metas da GQS exige a manipulação de um volume considerável de dados, que podem ser suportados por meio do uso de ferramentas de GCS, capazes de tornar a tarefa mais ágil.

Saiba mais

O Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) é uma organização profissional sem fins lucrativos que promove o conhecimento nas áreas de engenharia elétrica, eletrônica e computação. Ele foi fundado nos Estados Unidos.

Por sua vez, o Swebok (Guide to the Software Engineering Body of Knowledge) é um documento criado pelo IEEE que serve como referência para a comunidade de engenharia de software (BOURQUE; FAIRLEY, 2014).

O Swebok define a divisão dos tópicos da GCS conforme a Figura 1. As atividades são organizadas em: gestão dos processos de GCS, identificação de configuração de software (CS), controle de CS, medições e correções de CS, auditoria de CS, gestão de controle de versões e distribuição e a própria gestão das ferramentas de GCS (BOURQUE; FAIRLEY, 2014).

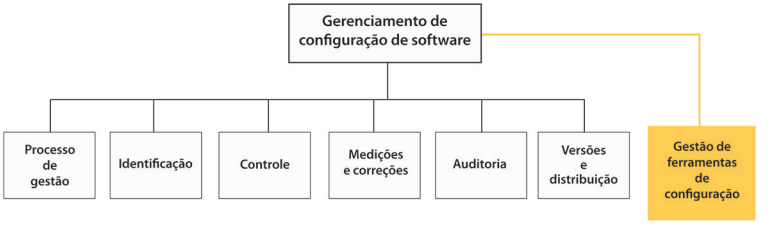

Figura 1. Organização das áreas de GCS.

Fonte: Adaptada de Bourque e Fairley (2014, documento *on-line*).

À medida que a complexidade do desenvolvimento de softwares aumenta, faz-se cada vez mais necessária a implementação tanto de processos de GCS quanto de ferramentas que automatizem os processos. Com essa implementação, cada alteração no software fica vinculada a registros dentro das ferramentas de GCS, garantindo a rastreabilidade de todas as mudanças e maior qualidade nos serviços e produtos entregues (PRESSMAN; MAXIM, 2016). Assim, a GCS é importante para a garantia da qualidade de software, a sua verificação e a sua validação, bem como para as revisões e auditorias.

O uso de ferramentas de GCS não é recente. Essas ferramentas são utilizadas para controle de revisão há muito tempo no ambiente UNIX, com funções mais limitadas e simplificadas (LEON, 2015). Seu foco baseava-se na gestão dos códigos, mas, com o passar do tempo, as ferramentas se tornaram mais completas, gerenciando várias funcionalidades do processo de desenvolvimento de softwares.

O aumento na busca por ferramentas de GCS pode ser atribuído a alguns fatores. Por exemplo: redução do tempo de desenvolvimento do software, melhoria na produtividade, agilidade no rastreamento de problemas, agilidade na liberação de novos softwares, redução de erros e maior integração das informações para auditorias (LEON, 2015).

Todas as características e funcionalidades existentes e a serem implementadas em ferramentas de GCS serão inúteis se as equipes de desenvolvimento não garantirem a atualização e o uso correto das ferramentas. A ideia de que o desenvolvimento de softwares pode ser feito por um único programador ficou no passado. E todas as vezes em que duas ou mais pessoas estão envolvidas num mesmo trabalho, um fator crucial para o bom desempenho das atividades é a comunicação interpessoal, que é gerenciada pela maioria das ferramentas de GCS.

Como você sabe, processos repetitivos que envolvem documentação, modificações, rastreamento, revisões e correções de problemas consomem muito tempo dos desenvolvedores. É por isso que, se bem utilizada, a ferramenta de GCS agiliza os trabalhos e garante a produtividade (LEON, 2015).

Automatização

As ferramentas que automatizam os processos de GCS se baseiam na organização proposta na Figura 1. Ou seja, cada uma das atividades é composta por um conjunto de tarefas que definem e manipulam os itens de configuração

(ICs). Os ICs podem ser documentos de quaisquer tipos, casos de testes, códigos de programas, componentes, bibliotecas, *frameworks*, modelos de bancos de dados, entre tantos outros artefatos próprios ao desenvolvimento de softwares (PRESSMAN; MAXIM, 2016). A seguir, você vai conhecer mais sobre a estruturação da GCS (BOURQUE; FAIRLEY, 2014).

- **Gestão de processos de GCS:** descreve padrões e procedimentos usados na definição da configuração do software (como o que será gerenciado), identifica os atores responsáveis por cada atividade, idealiza o cronograma, define a estrutura de repositório, estabelece as políticas de trabalho, determina as linhas de base e a sua estrutura e determina quais ferramentas são importantes para o auxílio de toda a gestão. É responsável pela organização do contexto, pela definição de regras e guias para todo o processo de GCS, pelo planejamento e pela supervisão da GCS.
- **Identificação de GCS:** controla e administra os itens de configuração de software de acordo com a especificação a seguir.
  - Identificação de cada item de configuração.
  - Relacionamento entre os itens.
  - Versões de software.
  - Linha de base e aquisição de artefatos de configuração.
- **Controle de CS:** realiza a gestão durante o ciclo de vida, determinando quais mudanças serão realizadas. Autoriza e aprova as alterações, apoiando a implementação das mudanças.
- **Medições de correções de CS:** controlam a correção de defeitos e erros durante o processo de desenvolvimento do software.
- **Auditoria de CS:** avalia a conformidade dos produtos de software, bem como os processos de regulamentação deles.
- **Gestão de versões e distribuição:** é a gestão do controle de versões de todos os artefatos, em especial do software desenvolvido, garantindo que os artefatos sejam liberados e documentados com confiabilidade. Inclui também o empacotamento das versões para instalação, bem como seus testes.
- **Gestão de ferramentas de configuração:** é responsável pelas definições e pela administração do uso das ferramentas mais adequadas para apoio a todo o processo de gestão de configuração.

Fique atento

Há três situações diferentes em relação à adoção de ferramentas de GCS:

- aquisição de ferramenta proprietária;
- adoção de ferramenta de código aberto (*open source*);
- implementação conforme os requisitos particulares do projeto.

Há certa variedade de ferramentas de GCS tanto proprietárias quanto de código aberto. Um dos aspectos importantes da análise e da definição de ferramentas é perceber o que é mais adequado a cada projeto. Muitas empresas optam por desenvolver uma metodologia de configuração por conta da complexidade e do alto custo do processo de GCS. Outras resolvem comprar ferramentas de empresas, muitas vezes consolidadas e que geram certa segurança no momento de treinamento e suporte. Finalmente, há a opção pelas ferramentas de código aberto, que são baixadas gratuitamente e que podem contar com suporte e treinamento ou não.

Antes de adotar ferramentas de GCS, é importante compreender todos os processos e atividades que devem ser automatizados no projeto de desenvolvimento de softwares. Não é indicado adotar uma alternativa simplesmente porque o mercado a utiliza.

Ferramentas, repositório e equipe de trabalho

Além de todos os requisitos necessários para a adoção de ferramentas de configuração, as próprias ferramentas devem estar sob a gestão de configuração — ou seja, é necessário atentar a ambientes de desenvolvimento integrado (*integrated development environment* — IDEs), compiladores, navegadores, pacotes, bibliotecas e outros ICs. Deve-se propor o congelamento de versões no repositório de configuração para que se estabeleça um conjunto de itens que estejam em pleno funcionamento (PRESSMAN; MAXIM, 2016).

Assim, os repositórios de configuração de software devem também contemplar o gerenciamento das ferramentas de desenvolvimento e os testes em suas versões compatíveis. Isso é importante para o bom andamento do ciclo de desenvolvimento do software e do processo de configuração. Uma alternativa interessante para que haja cobertura de tais dependências é a comunicação eletrônica padronizada, por meio de e-mails, wikis, gerenciadores de conteúdos

e mídias sociais internas (LEON, 2015). O repositório de GCS centraliza a integração das ferramentas, dos artefatos e de toda a comunicação, gerando um centro de fluxo do processo e das informações em formato uniforme e padronizado.

Fique atento

De nada adiantam esforço, padronização, documentação e comunicação se a equipe não estiver alinhada às atividades de configuração para garantir que cada alteração seja devidamente adicionada no repositório de CS.

Ferramentas, cenários, planejamento e funcionalidades

Um processo de configuração pode ser burocrático e exigir muito trabalho em paralelo ao próprio projeto de desenvolvimento de softwares. É por isso que as empresas planejam a configuração antes da escolha das ferramentas. Há ferramentas para todos os tamanhos de negócios e necessidades. Em alguns casos, devido à complexidade da ferramenta e de seu destaque no mercado de configuração de software, há treinamentos e certificações dos produtos.

A seguir, você vai ver um panorama sobre questões mais práticas. Primeiramente, define-se um cenário por meio de uma simples pergunta: quando se deve de fato implementar um sistema de gerenciamento de configuração? Não existe uma resposta única e exata, porém um bom momento é quando se agregam várias pessoas à equipe, que por vezes trabalham dispersas geograficamente. Assim, várias questões surgem, envolvendo autoria, políticas, orçamentos, cronogramas, a própria coordenação do time e dos processos e, finalmente, os custos envolvidos diante das constantes mudanças (LEON, 2015).

No roteiro de questionamentos importantes para o planejamento, podem constar as perguntas listadas a seguir (BOURQUE; FAIRLEY, 2014).

1. Organização: quais são as motivações para a aquisição das ferramentas?
2. Ferramentas: serão adquiridas ou desenvolvidas?
3. Ambiente: quais regras organizacionais e técnicas estarão no contexto do uso das ferramentas?
4. Legado: como os projetos futuros usarão ou não as ferramentas?

5. Financeiro: quem pagará pela aquisição, pela manutenção, pelo treinamento e pela customização das ferramentas?
6. Escopo: como as ferramentas serão implantadas em toda a organização e em cada projeto específico?
7. Responsabilidades: quem será responsável pela introdução de novas ferramentas?
8. Futuro: quais são os planos para as ferramentas no futuro?
9. Mudanças: quão adaptáveis são as ferramentas?
10. Divisões e fusões: as ferramentas são compatíveis com os planejamentos de divisões organizacionais e as estratégias de fusão?
11. Integração: já há alguma ferramenta em uso? As ferramentas escolhidas, bem como as existentes, possuem mecanismos de integração?
12. Migração: é possível fazer a migração dos repositórios existentes para outras ferramentas enquanto ocorre a manutenção completa de todo o histórico de registros?

Um detalhe não menos importante é o tamanho atual da equipe e as prospecções futuras de crescimento. Ao serem respondidas algumas questões, é indicado o início do planejamento, mesmo que minimalista, seguindo um padrão público ou uma metodologia própria de gestão de mudanças. Em linhas gerais, definem-se as pessoas envolvidas e as suas devidas responsabilidades. Também é importante estabelecer uma linha de base com especificações de tempo, fixação de políticas de controle de versões, definição das ferramentas básicas e do que será atualizado no repositório de configuração (LEON, 2015).

Um passo importante para complementar a base do planejamento é a formulação de um comparativo de ferramentas prontas, propondo um plano de aderência aos processos de GCS que serão implantados. Nesse caso, o Quadro 1 serve de modelo, considerando ferramentas proprietárias e de código aberto. Esse quadro traz algumas das funcionalidades mais importantes da automação do GCS das ferramentas elencadas.

O primeiro ponto a ser considerado é a questão de código aberto ou proprietário. Isso faz diferença principalmente para pequenas empresas, que possuem um orçamento enxuto. Empresas de médio a grande porte já podem considerar o pagamento de ferramentas, dando preferência à adoção de uma opção mais completa, por exemplo, o que pode simplificar o processo de treinamento e implantação.

Solucionado esse ponto, a segunda consideração é relativa ao alcance da ferramenta no mercado. Por exemplo, grande parcela de desenvolvedores possui experiência com a ferramenta GIT, utilizando-a geralmente em associação

à plataforma GitHub. Quanto mais usuários proficientes numa determinada ferramenta existem, mais agilidade há na equipe de trabalho.

O Quadro 1, a seguir, não contém todas as ferramentas existentes, porém considera as mais utilizadas e que estão em destaque no mercado.

Quadro 1. Comparativo de ferramentas de configuração de software

Ferramenta	Funcionalidades							
Nome	Processos	Identificação	Controle	Medições	Auditoria	Versões	Distribuição	O/P
Ansible Configuration Tool (ANSIBLE, 2019)	√	√	√				√	P
CFEngine Configuration Tool (CFEngine, 2019)	√	√	√					P/O
CHEF Configuration Tool (CHEF, 2019)	√	√	√	√	√	√	√	P
ConfigHub (2018)	√	√	√	√	√	√	√	O
GIT (2019c)						√		O
IBM Rational ClearCase (IBM, 2019)	√	√	√	√	√	√	√	P
JIRA (ATLASSIAN, 2019)	√	√	√	√				P
JUJU Configuration Tool (CANONICAL LTD., 2018)	√	√	√	√	√	√	√	O
Octopus Deploy (2019)	√	√	√	√	√		√	P
Puppet Labs (PUPPET, 2019)	√	√	√					O
RUDDER (2019)	√	√	√	√	√	√	√	O
SALTSTACK Configuration Tool (SALTSTACK, 2018)	√	√	√	√	√	√	√	P
SubVersion (2018)						√		O
TeamCity Configuration Tool (KHALUSOVA, 2015)	√	√	√	√	√		√	O

Fique atento

Para conhecer melhor as ferramentas apresentadas no Quadro 1, você pode conferir as páginas de cada uma delas na internet. Nelas, você vai encontrar documentação e tutoriais de instalação e uso.

Estudo de caso

A seguir, você vai conhecer melhor três ferramentas de gerenciamento de configuração: Git/GitHub, Puppet e TeamCity. O Git/GitHub é responsável pelo controle de versões de códigos. Já o Puppet ocupa-se do gerenciamento da infraestrutura. Finalmente, o TeamCity automatiza o processo de empacotamento e distribuição do software.

Os exemplos a seguir consideram um cenário hipotético de uma pequena empresa. A ideia é que você perceba como a implantação e a integração das ferramentas podem ser planejadas. Durante a apresentação do estudo de caso, você vai conhecer algumas funcionalidades das três ferramentas.

Git e GitHub: controle de versões e trabalho colaborativo

O GitHub é uma plataforma que simplifica o controle de versões de código e facilita a colaboração entre equipes de desenvolvimento de software. Ela é associada à ferramenta Git e ajuda na construção de um repositório centralizado, no qual todos os desenvolvedores podem trabalhar simultaneamente baixando, editando e gerenciando códigos de programas. O GitHub documenta quando e o que cada pessoa atualizou no mesmo código, simultaneamente ou não, ajudando no rastreamento das alterações por pessoas e por data (GIT, 2019a, 2019b, 2019c).

Seja para trabalhar com um grupo de desenvolvedores ou para criar um repositório com seu portfólio profissional, o GitHub é uma ferramenta interessante. Ele permite que seus repositórios estejam abertos à colaboração geral ou fechados para que somente você e seus colegas de projeto os enxerguem.

Fazendo uma busca rápida em páginas de empresas de recursos humanos na internet, você vai ver que várias empresas de tecnologia solicitam que o profissional tenha o seu próprio repositório, compondo um portfólio de trabalhos feitos. Assim, o GitHub acaba se tornando uma grande rede social

de profissionais ligados ao desenvolvimento de software e que prezam pela colaboração.

Você pode criar o seu perfil e os seus repositórios nessa ferramenta a qualquer momento, o que pode alavancar a sua carreira profissional. Além disso, com custo zero, pequenas empresas de desenvolvimento de software podem iniciar a automatização de processos de controle de versões e implantar os primeiros passos do gerenciamento de configuração de software.

Exemplo

A empresa Algoritmos Integrados Associados iniciou suas atividades há menos de um ano e já conta com uma pequena equipe multidisciplinar de 10 profissionais. Como a estrutura é pequena e inicial, a organização pretende trilhar caminhos mais automatizados no que diz respeito à qualidade e à gerência de configuração de software.

Depois de algumas reuniões para decisões sobre vários assuntos, mas principalmente focando na certeza de que precisam utilizar ferramentas de código aberto, os membros da equipe esperam desenhar uma trajetória de sucesso nos processos e aplicações. Para isso, eles optaram pela ferramenta de controle de versões Git, acompanhada da plataforma GitHub. Tal decisão se deu pelo fato de que esse processo de controle de versões é reconhecidamente o mais difícil de se gerir quando a equipe e a complexidade dos requisitos aumentam. Havia outras ferramentas disponíveis, como a SubVersion, porém, após a análise de uma tabela que comparava todas as ferramentas de controle de versões, a empresa optou pelo Git. A oferta de mão de obra que já tem experiência e utiliza a ferramenta para a colaboração em códigos foi um fator decisivo para que a Algoritmos Integrados Associados finalizasse a sua decisão.

A equipe definiu e estabeleceu os seus repositórios no GitHub. Antes de tudo, todos participaram de estudos e treinamentos, tentando aumentar a produtividade com o uso adequado das ferramentas. Toda documentação, os tutoriais e as demais informações técnicas foram extraídas das próprias páginas do Git e do GitHub. A ferramenta Git conta com um livro eletrônico considerando os primeiros passos e demais funcionalidades (GIT, 2019a, 2019b, 2019c). Também há muitas plataformas de ensino a distância oferecendo cursos básicos gratuitamente e com os custos mais variados. Não foi difícil optar pelos melhores cursos, uma vez que a Algoritmos Integrados possui um forte foco em estudos colaborativos entre os profissionais da equipe. Como você sabe, é imprescindível para empresas de desenvolvimento de softwares o estudo contínuo de ferramentas, tecnologias e metodologias de trabalho.

Como tarefa inicial, após os treinamentos, todos os profissionais criaram os seus perfis e se propuseram a colaborar com códigos abertos e pequenos tutoriais para fomentar a área de desenvolvimento de software. Os repositórios proprietários foram definidos como privados para garantir o sigilo dos trabalhos da empresa. Além de tudo, os membros da equipe se propuseram a conectar as suas IDEs padrões (Visual Studio Code e Sublime Text) à ferramenta Git.

Puppet: arquitetura e integração

O Puppet é uma ferramenta de código aberto direcionada à administração e à configuração de servidores, bem como à infraestrutura da tecnologia da informação (TI). Sua estrutura básica é composta por recursos e seus relacionamentos; estes são comparados com o estado atual do sistema e fazem alterações automaticamente, em conformidade com um catálogo preestabelecido. Essa ferramenta simula mudanças de configuração, faz o rastreamento do histórico do sistema em todo o seu ciclo de vida e identifica se um código ainda produz o mesmo estado do software.

Sua arquitetura é representada por três camadas (PUPPET, 2019):

- linguagem de configuração;
- camada de transação;
- camada de abstração de recursos.

A camada de linguagem de configuração é uma interface com linguagem natural que o ser humano pode compreender, e não uma linguagem de programação ou de modelo de dados. Ela foi elaborada para facilitar a interação.

Já a camada de transação é considerada o motor da ferramenta. Ela é responsável por interpretar e compilar a configuração, enviá-la para o agente, aplicá-la ao nó e relatar os resultados para o servidor. Tudo isso é feito por meio de um grafo que contém a lista de todos os recursos e de todos os seus relacionamentos. O grafo permite que a decisão seja ordenada para a aplicação da configuração com base nos relacionamentos configurados pelo administrador da ferramenta (PUPPET, 2019).

Finalmente, a camada de abstração de recursos simplifica o trabalho do administrador do sistema, que muitas vezes precisa lidar com diversos sistemas operacionais. Lembre-se de que cada sistema operacional gerencia arquivos, pacotes, processos e serviços em execução, programas, contas de usuários, grupos, entre outros elementos. A ferramenta Puppet compreende todos esses elementos como recursos.

Exemplo

Alguns meses se passaram na Algoritmos Integrados Associados e o controle de versões começou a ter bons resultados. A equipe aumentava e sentia-se a necessidade de contratar um profissional para DevOps. Novamente, depois de reuniões, foi contratado um profissional com experiência na ferramenta Puppet. Sua função era trabalhar em tudo o que se referisse à infraestrutura de TI da Algoritmos Integrados.

Um dos critérios utilizados para a implantação do Puppet foi a sua integração com ferramentas de controle de versões, entre elas o Git. A equipe elaborou então um esquema básico (Figura 2) de como a integração se daria e do que seria prioritário no gerenciamento pelo novo integrante da equipe.

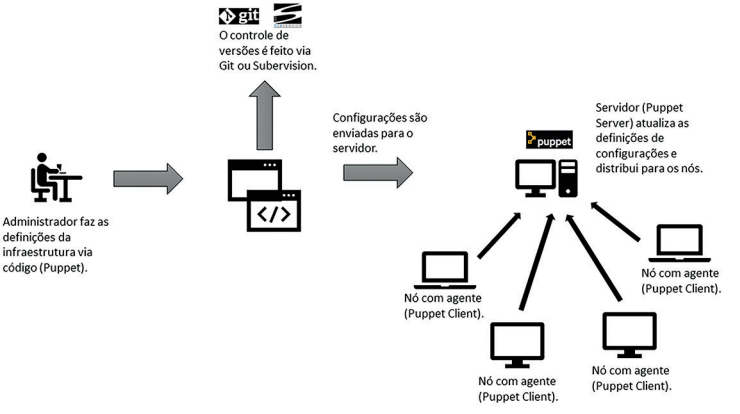

Figura 2. Funcionalidades básicas do Puppet integrado a um gerenciador de versões Git ou SubVersion.

Fonte: Adaptada de Puppet (2019, documento *on-line*).

O Puppet é geralmente usado como cliente/servidor, de acordo com o ciclo de operação a seguir.

- Cada cliente, também conhecido como nó, tem um aplicativo (agente) instalado e em execução, que se conecta com o servidor periodicamente. O tempo é configurado, mas o padrão é que a conexão ocorra de 30 em 30 minutos.

- Uma vez que atinja o tempo de sincronização, o nó envia a sua configuração atual compilada para o servidor.
- Cada configuração compilada é conhecida como "chamada de catálogo".
- O resultado no ato da sincronização é relatado ao servidor, demonstrando se há ou não divergências entre ele e o cliente.
- A configuração do código do Puppet é armazenada num sistema de controle de versão como Git ou SubVersion.

Um aspecto que a empresa considerou ao optar pelo Puppet foi a vasta literatura, os tutoriais e principalmente o fato de saber que havia uma comunidade brasileira (Puppet–BR) engajada em disseminar conteúdos sobre a ferramenta.

TeamCity: distribuição e integração

A ferramenta TeamCity tem por finalidade gerenciar a distribuição da construção de testes e a integração contínua. Ela possui compatibilidade com várias tecnologias, como Java, .NET, Docker, SubVersion, Git, JIRA, Octopus, entre outras. Também possui plug-ins de fácil configuração para serviços de nuvem, como Aws, Azure, entre outros.

Diferentemente do Git/GitHub e do Puppet, essa ferramenta possui dois tipos de licenças de uso: profissional (gratuita) e *enterprise* (paga/proprietária). Empresas pequenas podem iniciar a utilização pela versão profissional, uma vez que esta dá direito a 10 configurações de *build*. A persistência dos dados é feita por meio de um gerenciador de banco de dados MySQL, PostgreSQL, Oracle ou MS SQL. A TeamCity possui ainda uma wiki com muitos detalhes sobre todas as possibilidades de integração com outras ferramentas, seja por meio de APIs, XML ou JSON (KHALUSOVA, 2015).

Um conceito muito importante para a ferramenta é o *build*, uma vez que o seu foco principal é justamente o gerenciamento das entregas das aplicações. Ao empacotar o software para distribuição por meio do TeamCity, você pode acionar o gerenciamento de disparo de comunicação para os integrantes da equipe por meio de e-mail, Slack, Telegram e outros plug-ins.

À primeira vista, utilizar uma ferramenta de distribuição parece gerar um trabalho duplicado, porque o processo manual de implantação de software parece ser bem mais simples. No entanto, conforme a equipe de desenvolvimento torna-se maior, o TeamCity auxilia na distribuição de versões corretas e garante a rastreabilidade por meio de relatórios e métricas. A partir disso, é possível identificar problemas em testes e no próprio empacotamento do software para distribuição.

Exemplo

Depois do sucesso alcançado com a implantação do Puppet e do Git como ferramentas de GCS na Algoritmos Integrados Associados, novamente, após exaustivas reuniões e definições, a equipe optou por implantar uma ferramenta para cuidar do empacotamento e da distribuição de seus aplicativos.

A TeamCity foi a ferramenta escolhida depois de uma seleção minuciosa. O número de clientes aumentava e, consequentemente, cresciam os trabalhos de distribuição das aplicações, de acordo com todos os requisitos de entrega necessários. Embora a TeamCity se torne paga depois de determinado número de projetos, ainda sim a escolha foi vantajosa de acordo com a lista de requisitos.

Logo, a equipe identificou que seria necessário elaborar um plano de boas práticas para o uso correto da ferramenta. Então, criou um pequeno manual de boas práticas, com os itens listados a seguir.

- Após a instalação do TeamCity, o administrador de configurações de software criará um repositório que conterá um projeto.
- O projeto poderá ser acessado e configurado num repositório manual ou em outro repositório, como no GitHub.
- Todo projeto receberá um Project ID preenchido automaticamente. Ele será um identificador para as configurações da própria ferramenta e gerará uma URL apontando para tal Project ID.
- Uma implantação futura será o Container Docker, pois ambos trabalham muito bem em conjunto.
- Sempre é necessária a realização de backups e a configuração de notificações de *builds* para a geração de relatórios de controle.
- Outra parametrização importante na ferramenta é a implementação de medidas para a resolução de falhas nos *builds*.
- A integração com um sistema de controle de versão gera maior confiabilidade nos artefatos gerados em cada *build*; nesse caso, o Git.
- Para maiores informações sobre conceitos, instalação e uso do TeamCity, siga corretamente os manuais e tutoriais disponibilizados pelo desenvolvedor.

Mais agilidade nos processos de controle de versões, infraestrutura e distribuição garantiram resultados importantes à empresa, como maior produtividade dos desenvolvedores e maior confiabilidade na liberação de versões e aplicativos para seus clientes. Agora, a Algoritmos Integrados almeja alçar novos voos. Para isso, está estudando outros recursos, bibliotecas e *frameworks* que possam colaborar na automatização de suas ferramentas de GCS.

Saiba mais

O Docker é uma plataforma (de código aberto) criada pelo Google para facilitar o processo de virtualização. Ele facilita a criação e a administração de ambientes para a disponibilização mais rápida de programas para os clientes. O Docker cria, testa e implementa aplicações num ambiente chamado *container*.

O desenvolvedor empacota o seu aplicativo de forma padronizada e o disponibiliza para execução. Empacotar prevê colocar no *container* os códigos, as bibliotecas, algum *runtime* (se houver) e outras ferramentas necessárias para a execução do software. Saiba mais no link a seguir.

https://qrgo.page.link/ihbS4

Como você viu, o mundo do gerenciamento de configuração de software é extremamente amplo. Assim, é importante que as equipes de desenvolvimento de software busquem periodicamente apoio e informações sobre novas plataformas, *frameworks*, bibliotecas e tudo mais que automatize o processo de configuração.

Referências

ANSIBLE. [*Site*]. 2019. Disponível em: https://www.ansible.com/. Acesso em: 26 jun. 2019.

ATLASSIAN. *Jira software.* c2019. Disponível em: https://www.atlassian.com/software/jira/. Acesso em: 27 jun. 2019.

BOURQUE, P.; FAIRLEY, R. E. D. (ed.). *Swebok v3.0:* guide to the software engineering body of knowledge. Piscataway: IEEE, 2014. Disponível em: https://ieeecs-media.computer.org/media/education/swebok/swebok-v3.pdf. Acesso em: 26 jun. 2019.

CANONICAL LTD. *Juju.* c2018. Disponível em: https://jujucharms.com/. Acesso em: 27 jun. 2019.

CFENGINE. [*Site*]. c2019. Disponível em: https://cfengine.com/. Acesso em: 26 jun. 2019.

CHEF. [*Site*]. [2019]. Disponível em: https://www.chef.io/. Acesso em: 26 jun. 2019.

CONFIGHUB. [*Site*]. c2018. Disponível em: https://www.confighub.com/. Acesso em: 26 jun. 2019.

GIT. [*Site*]. [2019c]. Disponível em: https://git-scm.com/. Acesso em: 26 jun. 2019.

GIT. *Primeiros passos:* sobre controle de versão. [2019a]. Disponível em: https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o. Acesso em: 26 jun. 2019.

GITHUB. [*Site*] c2019b. Disponível em: https://github.com/puppet-br/apostila-puppet/releases. Acesso em: 26 jun. 2019.

IBM. [*Site*]. [2019]. Disponível em: https://www.ibm.com/us-en/marketplace/rational-clearcase. Acesso em: 26 jun. 2019.

KHALUSOVA, M. *Getting started with TeamCity.* 2015. Disponível em: https://confluence.jetbrains.com/display/TCD9/Getting+Started+with+TeamCity. Acesso em: 26 jun. 2019.

LEON, A. *Software configuration management handbook*. 3rd ed. Norwood: Artech House, 2015.

OCTOPUS DEPLOY. [*Site*]. c2019. Disponível em: https://octopus.com. Acesso em: 27 jun. 2019.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software:* uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

PUPPET. [*Site*]. [2019]. Disponível em: http://puppet-br.org/. Acesso em: 26 jun. 2019.

RUDDER. [*Site*]. 2019. Disponível em: http://www.rudder-project.org/. Acesso em: 27 jun. 2019.

SALTSTACK. [*Site*]. c2018. Disponível em: https://www.saltstack.com/. Acesso em: 27 jun. 2019.

SUBVERSION. *Apache™ Subversion®*. c2018. Disponível em: https://subversion.apache.org/. Acesso em: 27 jun. 2019.

Leituras recomendadas

FERNANDES, J. M. *Metodologia para a implantação de gerência de configuração de software em empresas de médio porte.* 2011. Dissertação (Mestrado em Computação Aplicada) – Centro de Ciências e Tecnologia, Universidade Estadual do Ceará, Rio de Janeiro, 2011.

MUNDO DOCKER. [*Site*]. c2019. Disponível em: https://www.mundodocker.com.br/tag/docker-brasil/. Acesso em: 27 jun. 2019.

SOMMERVILLE, I. *Engenharia de software*. 9. ed. São Paulo: Pearson, 2011.

THE ITIL Open Guide. [2019]. Disponível em: https://www.itlibrary.org/. Acesso em: 27 jun. 2019.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS