APRESENTAÇÃO
DE APOIO

DEVOPS BÁSICO

Marco Aurélio Souza Mangan – Aula 03

PUCRS online uol edtech

Professores

FABRÍCIO VERONEZ
Professor Convidado

Fundador da Formação KubeDev, é um grande apaixonado por tecnologias. Com mais de 13 anos de experiência no mercado de tecnologia, atuando como desenvolvedor e arquiteto em projetos de pequeno e grande porte, atualmente dedica-se a transformar a carreira de profissionais de TI, compartilhando conhecimento sobre o universo de containers e a cultura DevOps. Em 2018, passou a compartilhar a sua experiência através da criação de conteúdos. Foi assim que surgiu o seu site veronez.dev e, em seguida, o seu canal no YouTube.

MARCO AURÉLIO SOUZA MANGAN
Professor PUCRS

Possui graduação em Ciências da Computação pelo Instituto de Informática/UFRGS, mestrado em Ciências da Computação pelo Instituto de Informática/UFRGS e doutorado em Engenharia de Sistemas e Computação pelo PESC/UFRJ. Atualmente, é professor da Faculdade de Informática/PUCRS e da Faculdade Senacrs Porto Alegre. Tem experiência na área de Ciência da Computação, com ênfase em linguagens de programação, atuando principalmente nos seguintes temas: engenharia de software, CSCW, ambientes de desenvolvimento de software, reutilização de software e mecanismos de cooperação.

Ementa da disciplina

Introdução aos fundamentos de gerência de configuração. Estudo sobre Integração contínua (CI). Utilização de contêineres, ferramentas e ambientes direcionados ao desenvolvimento de software como Git, GitHub, Maven, Gradle, Npm, Yarn, GitHub Actions, Jenkins, Travis e Docker.

Ementa

- Introdução aos fundamentos de gerência de configuração. Estudo sobre Integração Contínua (CI). Utilização de contêineres, ferramentas e ambientes direcionados ao desenvolvimento de software como Git, GitHub, Maven, Gradle, Npm, Yarn, GitHub Actions, Jenkins, Travis e Docker

Roteiro

- Revisar conceitos e motivação para DevOps
- Apresentar as principais categorias de ferramentas e práticas relacionadas.
- Apresentar um exemplo de repositório estável que demonstra algumas das práticas de DevOps que permitem o sucesso da Integração Contínua.

Um exemplo mais extenso

- A aplicação PetClinic serve como demonstração das capacidades de um framework de desenvolvimento de software.
- Inicialmente, escrita e adaptada para plataforma Java.
- Existem adaptações para as mais diversas arquiteturas e frameworks, servindo de divulgação de boas práticas.

Repositórios da PetClinic

- Petclinic J2EE

https://github.com/Jakarta-EE-Petclinic/javaee7-petclinic

- Petclinic REST

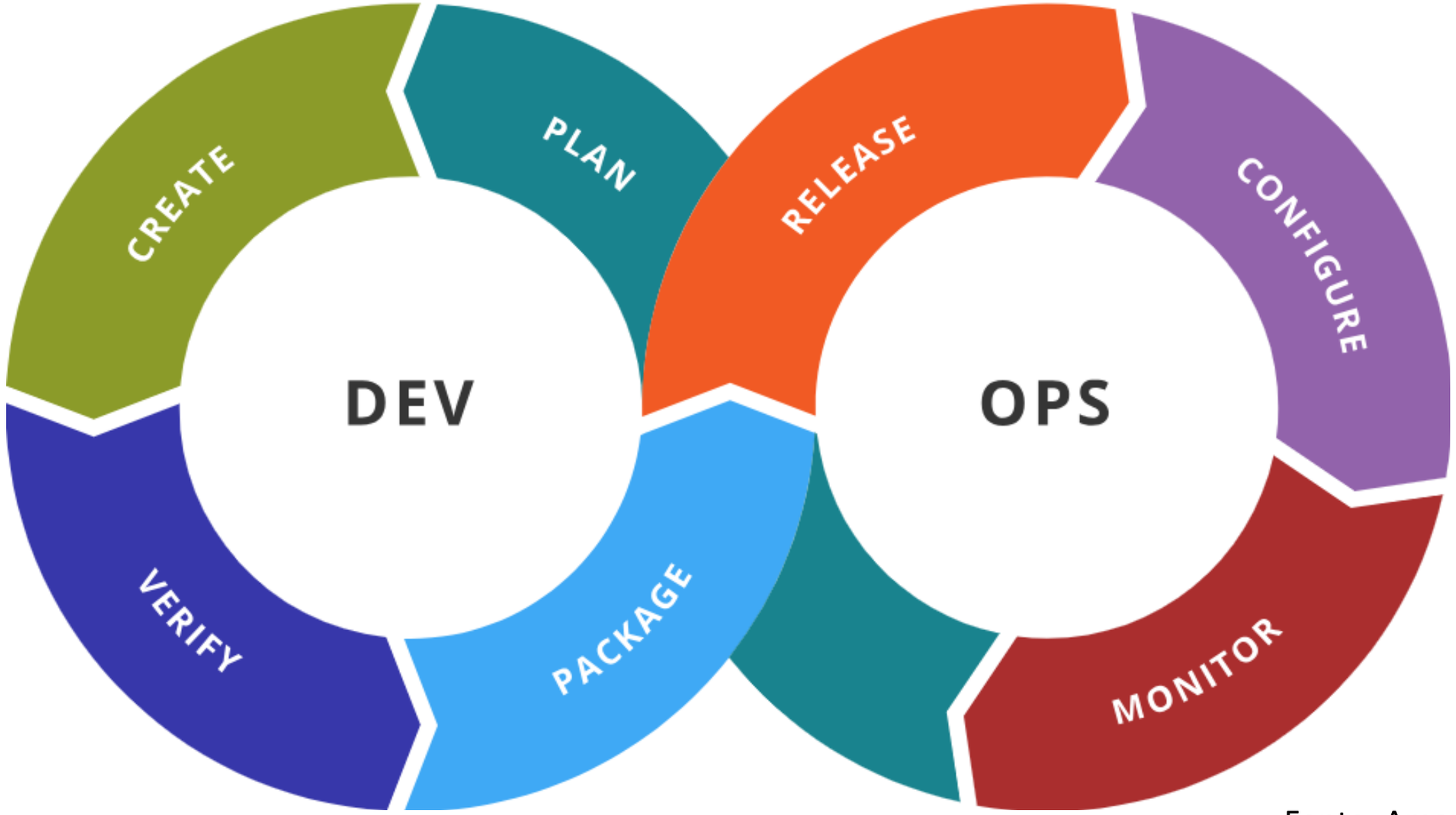
https://github.com/spring-petclinic/spring-petclinic-rest

- Petclinic Angular

https://github.com/spring-petclinic/spring-petclinic-angular

DevOps

- Definir, controlar e aprimorar os processos do ciclo de vida do software.
- Construir, testar, empacotar e implantar software e sistemas de forma segura e confiável.
- Ampliar a comunicação e integração de desenvolvedores e demais envolvidos.



Fonte: Amazon.com

Preocupações

- Construir (*to build*): O processo de gerar um sistema executável e testável a partir de código-fonte.
- Empacotar (*to package*): combinar componentes relacionados em um único item implantável.
- Implantar (*to deploy*): iniciar a etapa do ciclo de vida do software em que a operação acontece.

Uma semana antigamente (antes de DevOps)

- Segunda-feira: equipe de desenvolvimento termina um módulo e solicita à equipe de teste que verifique-o.
- Terça-feira: A equipe de teste percebe o pedido e descobre que o módulo é novo, não há instruções de implantação. Teste solicita instruções.
- Quarta-feira: equipe de desenvolvimento estava em um evento.
- Quinta-feira: equipe de desenvolvimento envia instruções de implantação..

Parte do livro: The DevOps Adoption Playbook

Simplificando ao extremo...

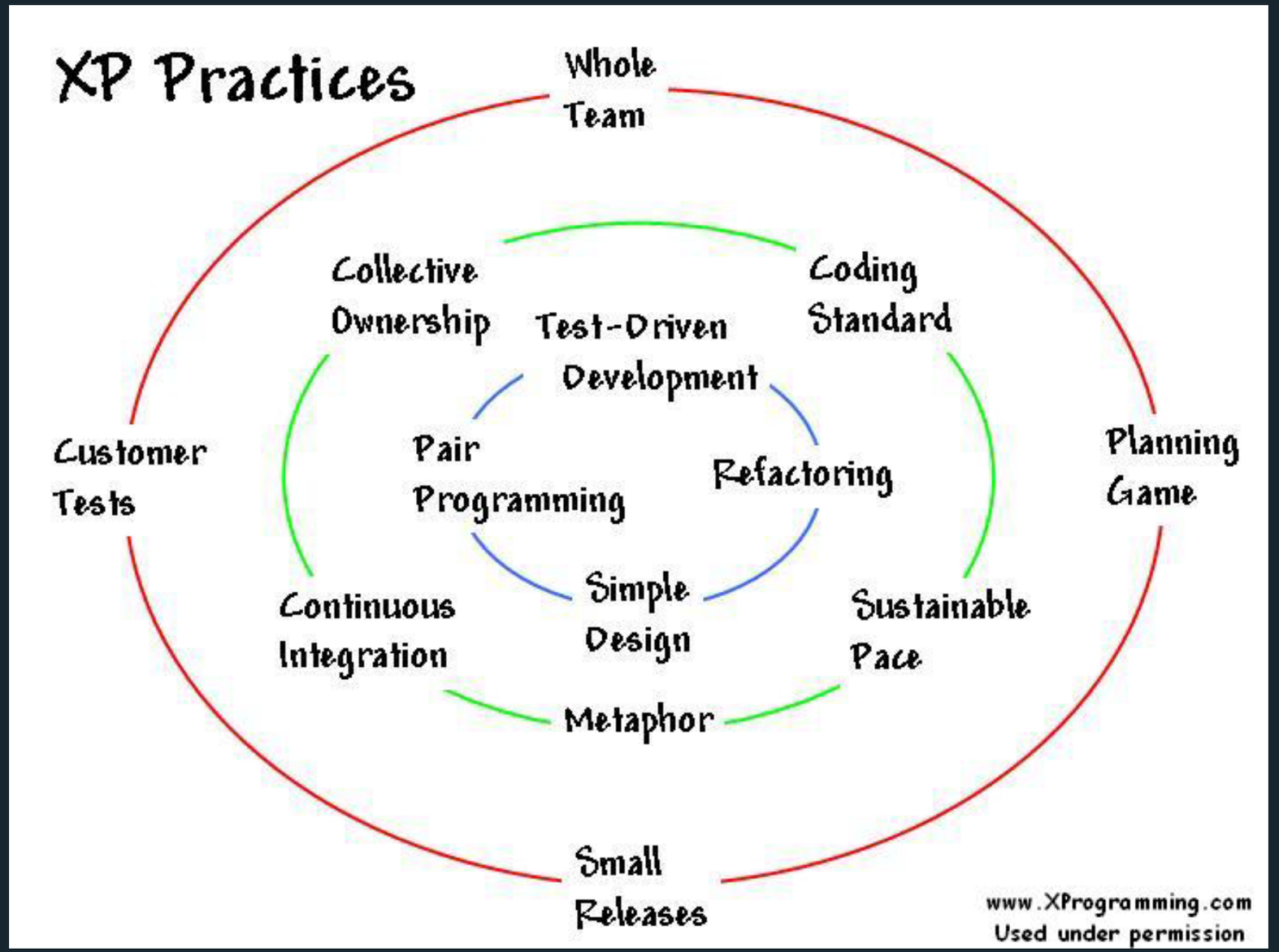
- *Desenvolvedores (Dev) se preocupam com velocidade, em realizar mudanças.*
- *Operações (Ops) se preocupam com estabilidade e segurança.*

Flickr/Yahoo

- ALLSPAW e HAMMOND (2009) 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr. In: O'Reilly Velocity.
- Antecessores:
  - Plan-Do-Check-Act
  - Lean
  - Toyota

Mudanças com DevOps

- *Left-shift ou shift-left*: priorizar participação dos envolvidos ao aplicar atividades de garantia de qualidade, segurança, privacidade, desempenho, verificação e validação **mais cedo** no ciclo de vida.
- A **automação** produz registro de auditoria ao longo das etapas do ciclo de vida, o que inclui validação e validação do software.
- Mudanças culturais: implantações pequenas, simples e automatizadas que cabem no dia-a-dia da equipe.
- Objetivos diversos: velocidade, inovação, escala...


XP Practices
Whole Team
Collective Ownership
Coding Standard
Test-Driven Development
Pair Programming
Refactoring
Customer Tests
Planning Game
Continuous Integration
Simple Design
Sustainable Pace
Metaphor
Small Releases
www.XProgramming.com
Used under permission

Competência técnicas

A. *Utilizar ferramentas de gerência de configuração.*
B. *Administrar sistemas operacionais.*
C. *Adotar integração contínua.*
D. Implantar registros de auditoria (*logs*).
E. Aprimorar o monitoramento de execução.
F. Facilitar a configuração de senhas, credenciais e conexões.
G. Automatizar tarefas: construção, teste, revisão, instalação.
H. [Aprimorar a segurança da aplicação. DevSecOps]

Automação de tarefas

- *A automação reduz o tempo de executar a atividade e reduz a chance de erros.*
- *A automação pressupõe que o conhecimento necessário nem sempre está disponível durante a realização da tarefa.*
- O roteiro da automação está sujeito à revisão, melhorias e pode ser transferido.

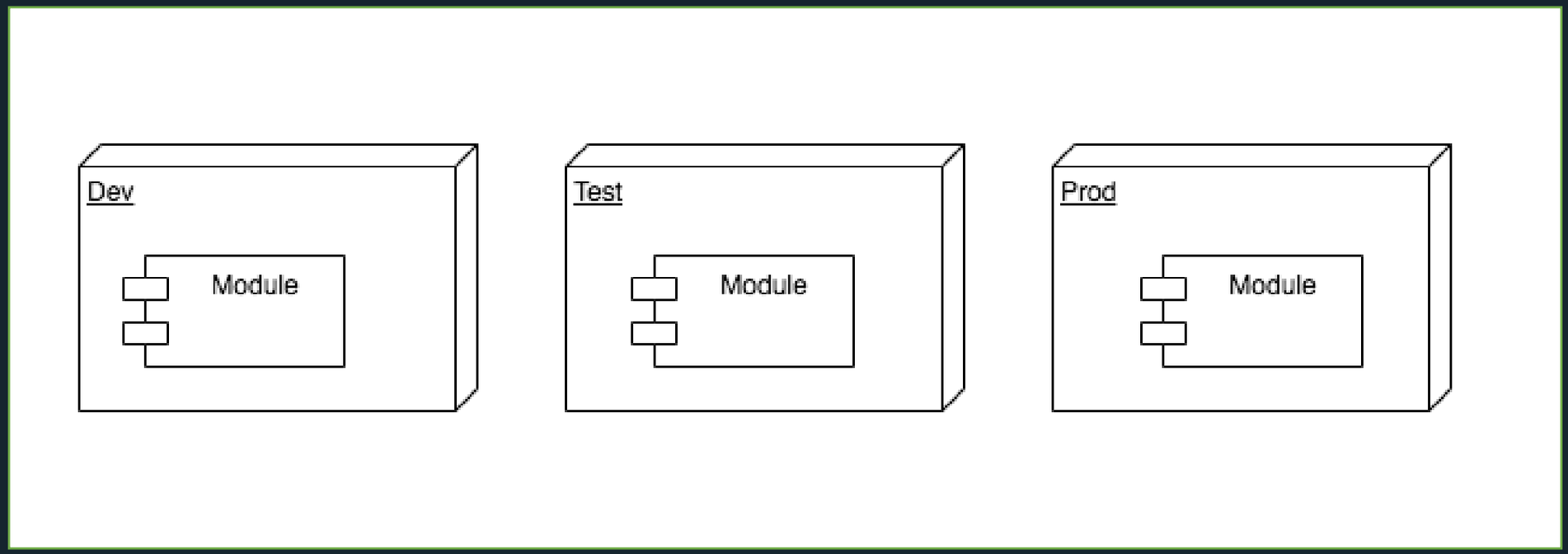
Resumo

- DevOps é um método ou conjunto de práticas.
- As competências técnicas básicas dependem do uso de ferramentas.
- As ferramentas evoluem e existem muitas alternativas.
- Procure perceber quais as possibilidades das ferramentas do restante da apresentação.
- Priorize e adapte essas possibilidades aos seus projetos e localize nas suas ferramentas como realizar essas possibilidades.

Gerência de configuração

- Gerência de versões e mudanças
  - Git, GitHub
- Gerência de dependências internas e automação de construção
  - Maven, Gradle, Npm, Yarn
- Gerência de pacotes
  - apt

Três ambientes



Gerência de versões e mudanças

- Registra quem realizou mudanças no repositório.
- Permite recuperar configurações específicas, selecionando mudanças.

- Git, GitHub

Dependências e automação de construção

- Reduz a diferença de configuração de dependências.
- Gerência de dependências internas e automação de construção

- Maven, Gradle, Npm, Yarn

Gerência de pacotes

- Instala programas executáveis.
- Parte da tarefa de administrar um computador.

- apt, homebrew, Windows Package Manager

Guias

- Configuração e mudanças:
  - https://git-scm.com
  - https://learngitbranching.js.org
- Dependências e construção:
  - https://mvnrepository.com/repos/central
  - https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html
- Pacotes:
  - https://help.ubuntu.com/community/AptGet/Howto

Estudo de caso: SDKMAN

- Instalador de ferramentas de desenvolvimento.

- Realiza o levantamento da configuração de software e hardware.
- Filtra ferramentas compatíveis a partir de índice geral.
- Transfere, instala e configura o JDK.

- Exemplo: equipamento com mais de cinco anos, versão adequada não está disponível, somente versão mais recente.

- Guia:

https://sdkman.io

Estudo de caso: Spring Boot

- Guias:

https://spring.io/quickstart

https://spring.io/guides/gs/spring-boot/

Resumo

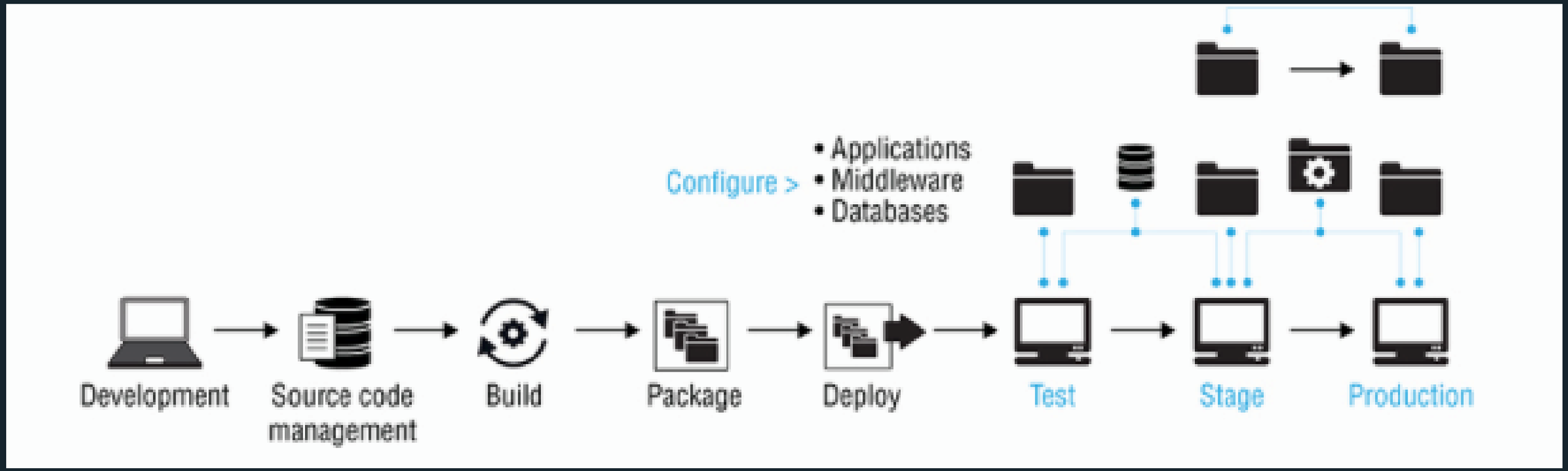
- A gerência de configuração inclui o esforço de gerência de versões.
- O repositório não armazena somente código-fonte: requisitos, ordens de serviço, testes, automações estão visíveis e versionados.
- O repositório também oferece a oportunidade de comunicação e planejamento.

Integração contínua

- Processo recorrente de tentar combinar os diversos elementos da aplicação.
- Primeiras tarefas são mover aplicação para fora do computador do desenvolvedor e combinar todas as contribuições.

- GitHub Actions, Jenkins, Travis

O pipeline CI/CD



Kent Beck sobre 10 Práticas em IC

1. Mantenha um repositório unificado.
2. Automatize a construção.
3. Automatize o teste da construção.
4. Incentive todos a contribuir no produto, todos os dias.
5. Cada contribuição executa testes de integração.
6. Mantenha o build rápido.
7. Teste em um ambiente igual ao de produção.
8. Facilite a todos a obter um executável.
9. Verifique que todos possam saber o que está acontecendo.
10. Por fim, automatize a implantação.

Entrega ou Implantação?

- A Entrega Contínua tem por objetivo verificar que o sistema está pronto para implantação.
- A Implantação Contínua tem o objetivo de colocar o sistema em produção.

Guias

- GitHub Actions

  https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions

  https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration

  https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-java-with-maven

Estudo de caso: Sonar

- Avaliação estática de um conjunto de regras de verificação.
- Automatiza a revisão de código que seria realizada por um programador experiente.

- Guia:

https://sonarcloud.io

Resumo

- A integração contínua é uma prática que depende da adoção de práticas relacionadas, como a propriedade coletiva e a automação de testes.
- A integração contínua sinaliza a possibilidade de combinar as contribuições dos diversos desenvolvedores em uma única entrega.
- O resultado da integração pode ser implantado manualmente ou automaticamente.

Contêineres

- Isolamento de serviços
- Empacotamento da aplicação

- Docker (ex.: para Node.js)
- Servidores de aplicação (ex.: para Java)

O que é um contêiner (a)?

- Um processo do sistema operacional que executa restrito a um conjunto de recursos.
- Isolamento provido em parte por recursos do sistema operacional.

- Ex.: docker run

O que é um contêiner (b)?

- Um ambiente gerenciado que organiza serviços e dependências.
- A aplicação solicita recursos ao ambiente gerenciado.

- Ex.: Java Enterprise Edition, Google Android

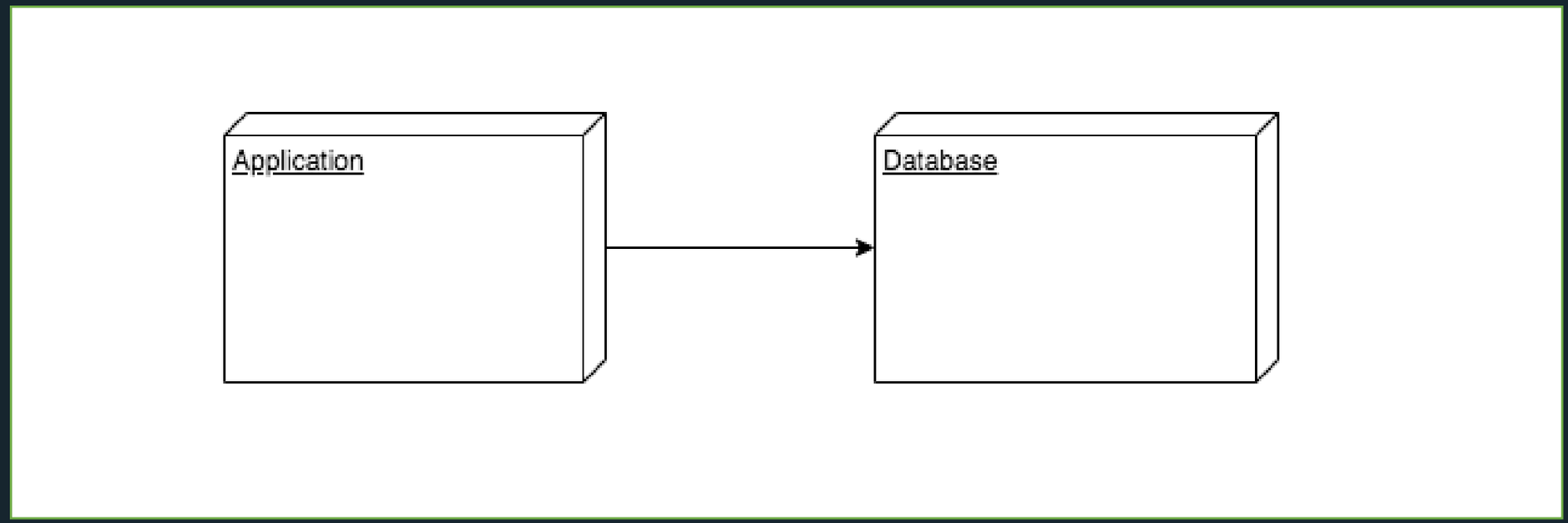
Isolamento de serviços

- Uma aplicação necessita de diversos serviços.
- A instalação e configuração de um serviço em um mesmo computador pode causar interferência em outros serviços instalados.
- Um contêiner pode ser utilizado para isolar serviços.
- Neste caso, um contêiner para cada serviço.
  - Exemplo: a aplicação necessita de um serviço de banco de dados.
  - A imagem contém o serviço que a aplicação necessita.
- Ex.: docker composer

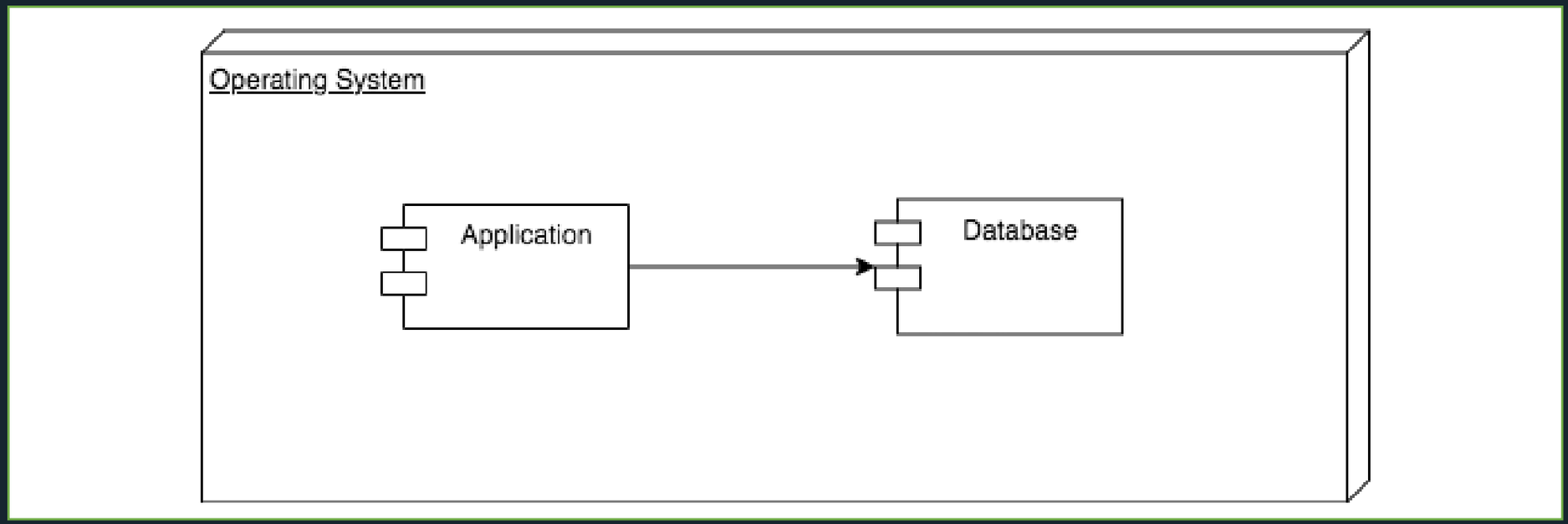
Empacotamento da aplicação

- Implantar a aplicação em novo ambiente.
- Gerenciar a instalação de dependências da aplicação.
- A aplicação permanece em um contêiner.
- Caso existam serviços, podem ser isolados em seus próprios contêineres.
- Exemplo, a imagem da aplicação necessita de um serviço de banco de dados que foi isolado em outro contêiner.

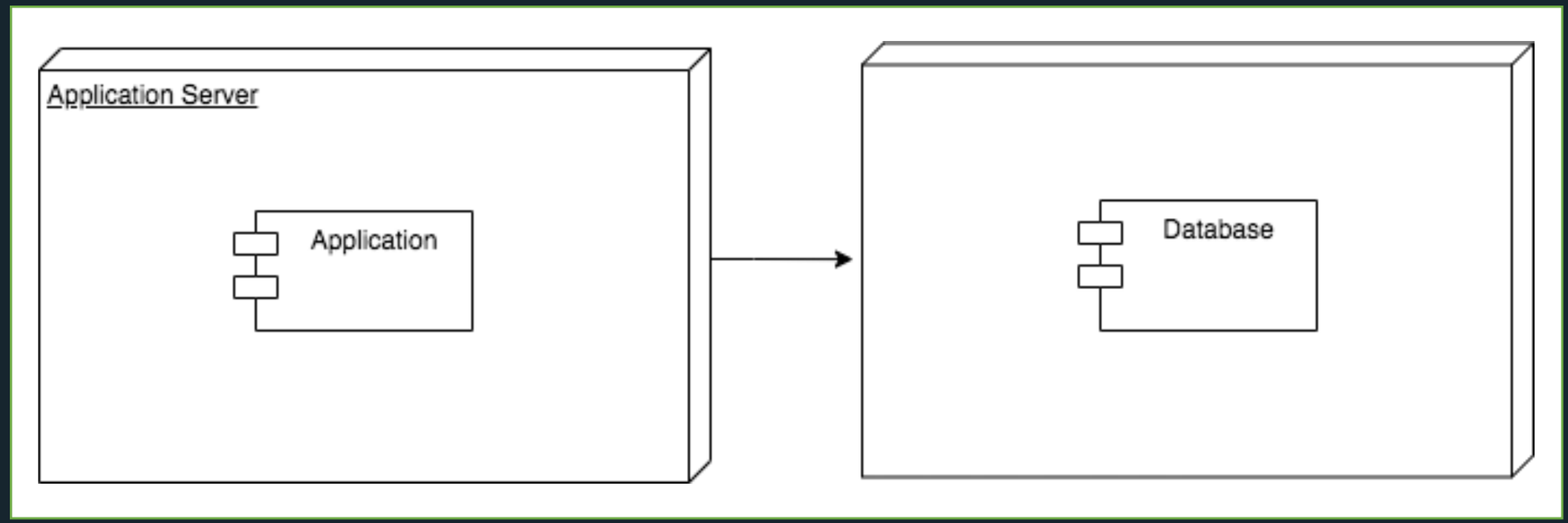
Aplicação e serviço (Infraestrutura)



Aplicação e serviço (IaaC)



Aplicação e serviço (*Application server*)



Guias

- Docker Orientation and Setup

https://docs.docker.com/get-started/

- Play with Docker

https://www.docker.com/101-tutorial

Estudo de caso: Spring

- Guias:

https://spring.io/guides/gs/spring-boot-docker/

Estudo de caso: Google Jib

- Plugin que permite construir uma imagem Docker sem a necessidade de instalação local.
- Incentiva a adoção de contêiner e da nuvem.

- Guia:

https://cloud.google.com/blog/products/application-development/introducing-jib-build-java-docker-images-better

Resumo

- O contêiner auxilia na redução de variabilidade entre ambientes de desenvolvimento, teste e produção.

PUCRS online uol edtech