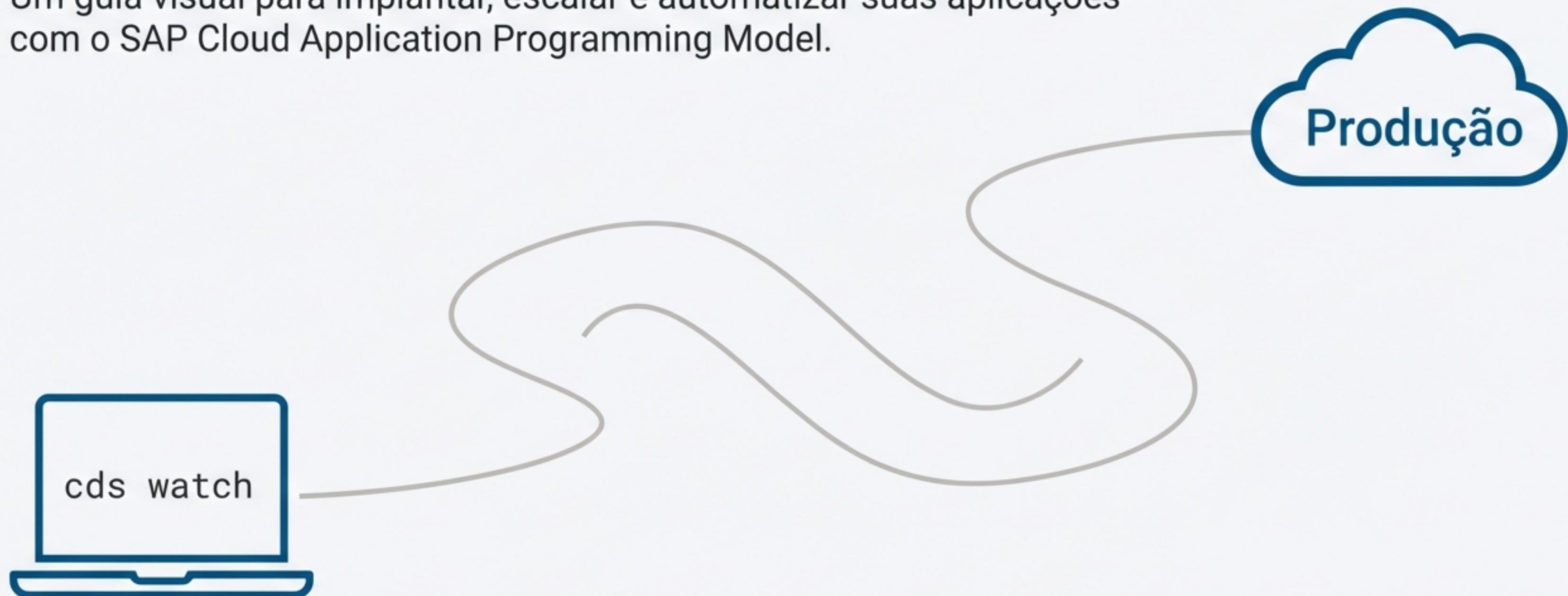


Do `cds watch` à Produção: A Jornada de Deploy de Aplicações CAP

Um guia visual para implantar, escalar e automatizar suas aplicações com o SAP Cloud Application Programming Model.



O Caminho para a Produção

Implantar uma aplicação CAP é uma jornada com etapas bem definidas. Começamos com a decisão fundamental do ambiente e avançamos progressivamente em direção a uma solução totalmente automatizada e resiliente. Este guia irá percorrer cada etapa.



Etapa 1: Escolha do Ambiente

Onde sua aplicação vai operar.

Etapa 2: Arquitetura Escalável

Estruturando para o crescimento.

Etapa 3: Automação do Fluxo

Implementando CI/CD.

Etapa 4: Domínio do Build

Personalizando os artefatos de deploy.

Etapa 5: Prontidão Operacional

Garantindo a resiliência.

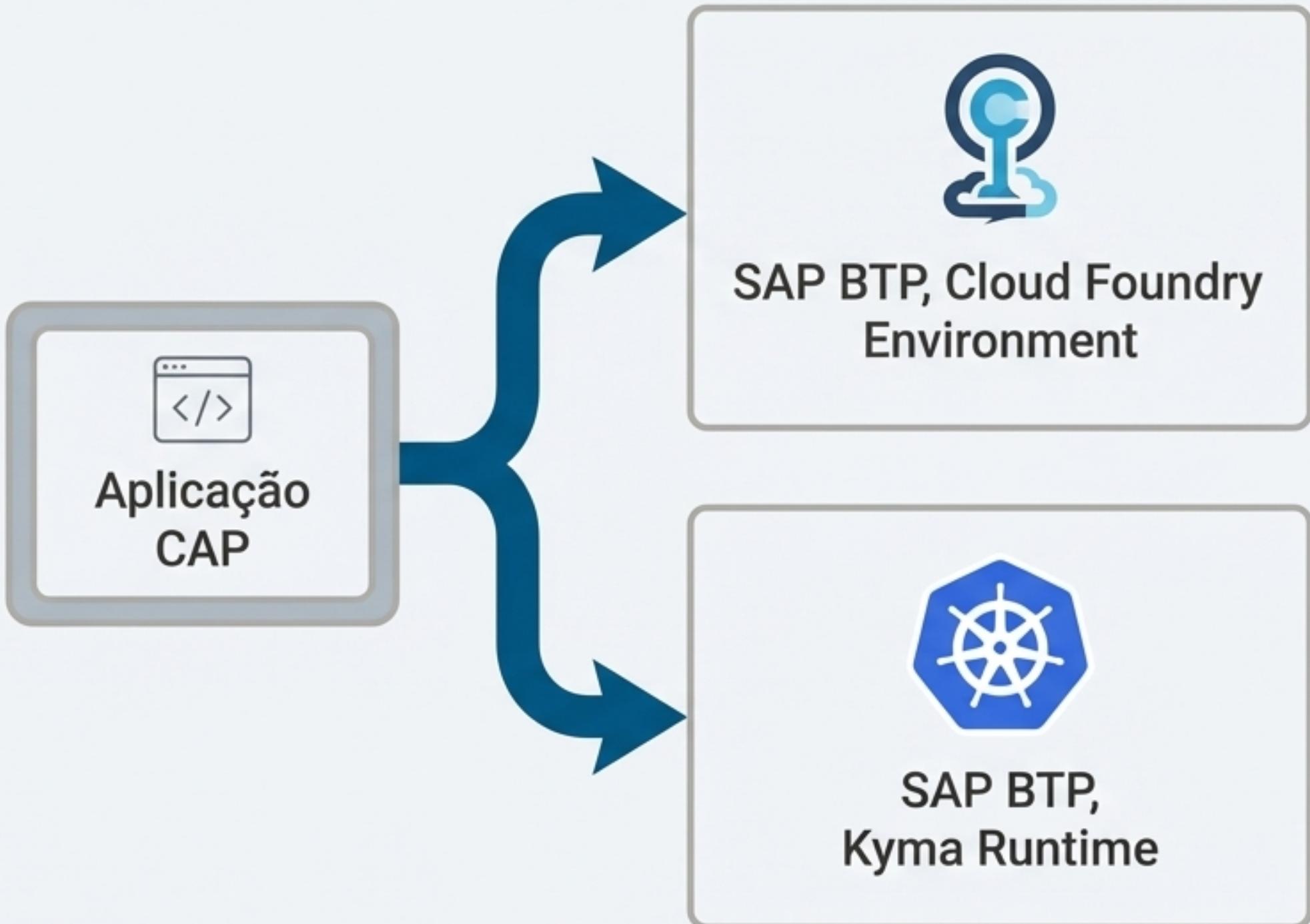
Etapa 1: Escolha do Ambiente de Execução

Onde sua Aplicação CAP Irá Residir?

A primeira decisão estratégica na sua jornada para a produção é selecionar o ambiente de execução no SAP BTP. A

escolha entre Cloud Foundry e Kyma impactará a forma como sua aplicação é empacotada, implantada e gerenciada.

A decisão depende dos seus requisitos de arquitetura, habilidades da equipe e necessidades de escalabilidade.



Cloud Foundry vs. Kyma: Uma Comparação Estratégica



Cloud Foundry

**Paradigma:**

Platform-as-a-Service (PaaS)

**Abstração:**

Foco na aplicação (cf push). A plataforma gerencia servidores, roteamento e runtimes.

**Unidade de Deploy:**

Arquivo MTA (mta.yaml), que descreve módulos de aplicação e dependências de serviço.

**Ideal para:**

Aplicações que se beneficiam de uma experiência de implantação simplificada e gerenciada, com integração profunda aos serviços do BTP.

**Comando chave:**

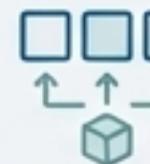
cds up



Kyma

**Paradigma:**

Kubernetes / Containers

**Abstração:**

Controle granular sobre o ambiente. Você gerencia imagens de container e recursos do Kubernetes.

**Unidade de Deploy:**

Imagens de container (Docker) e configurações de recursos do Kubernetes (Helm Charts).

**Ideal para:**

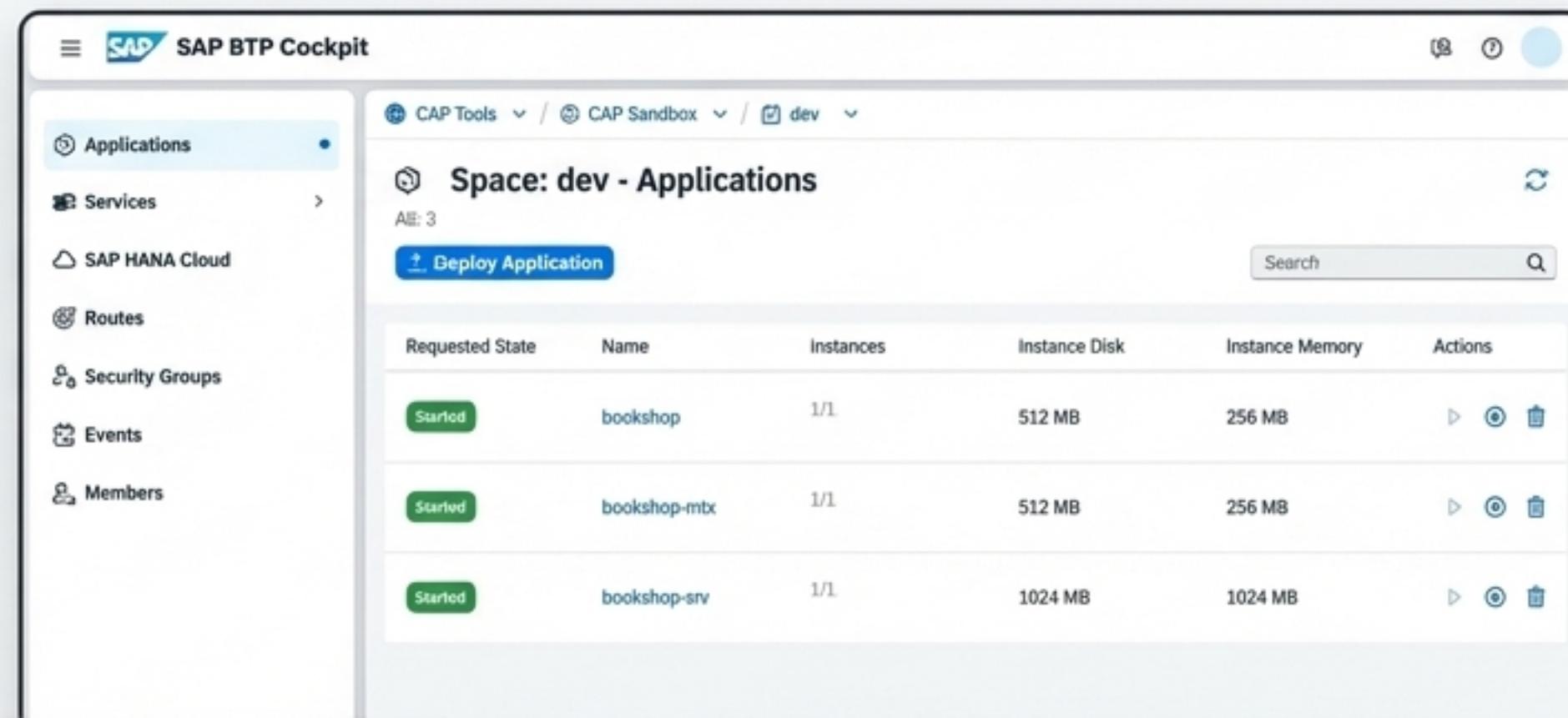
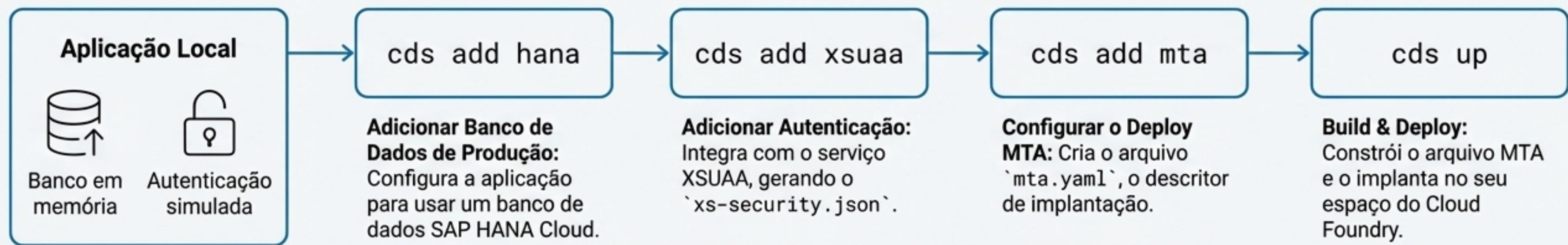
Aplicações nativas da nuvem, arquiteturas de microsserviços e cenários que exigem extensibilidade e padrões abertos do Kubernetes.

**Comando chave:**

cds up -2 k8s

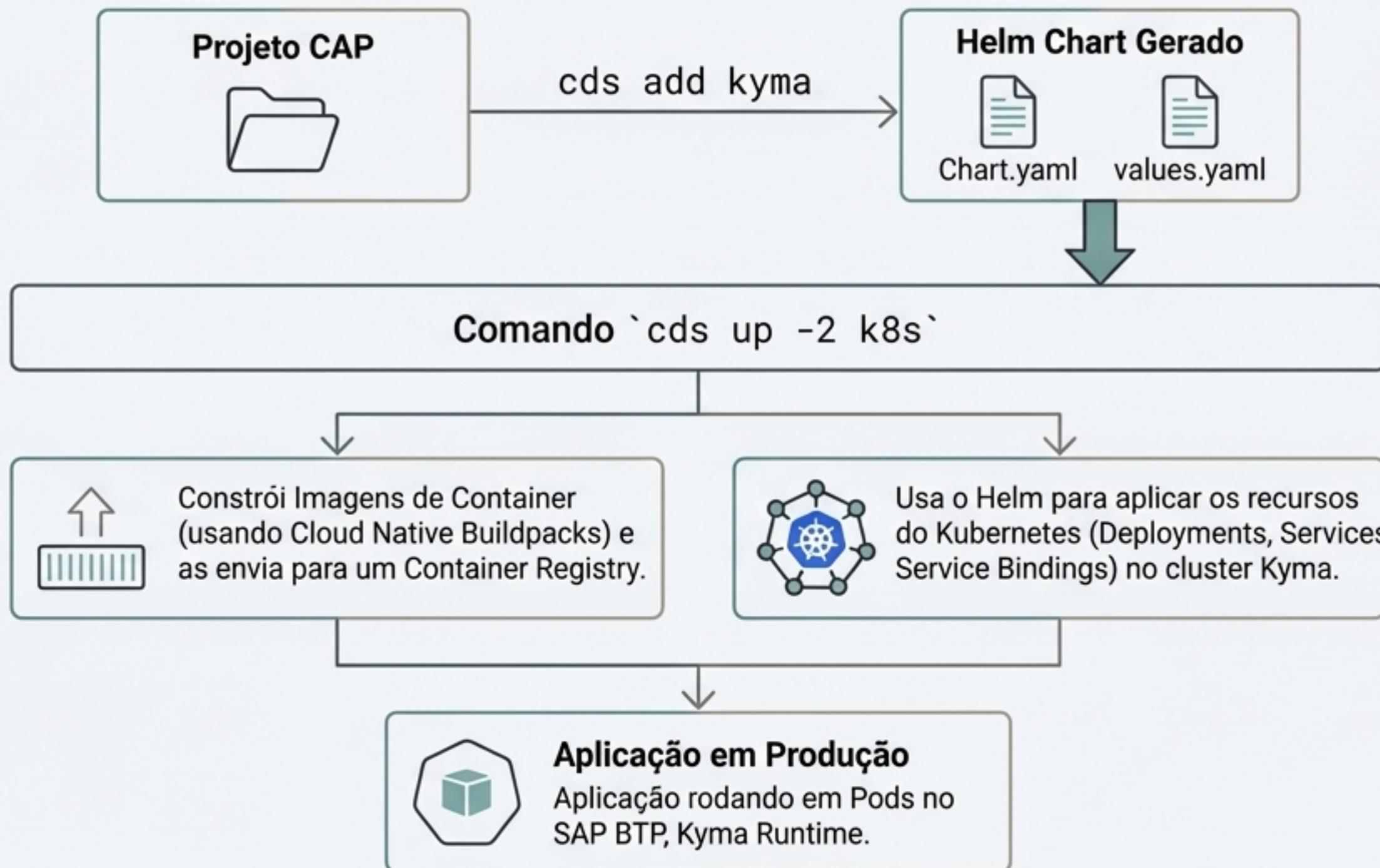
O Caminho do Cloud Foundry: Preparando para Produção

Para implantar no Cloud Foundry, preparamos a aplicação para serviços de nível de produção. O CAP automatiza a maior parte dessa configuração através de comandos cds add.



O Caminho do Kyma: Empacotando para Kubernetes

No Kyma, o deploy se baseia em padrões nativos do Kubernetes. Sua aplicação é empacotada como uma imagem de container, e sua implantação é definida por recursos do Kubernetes, gerenciados por um Helm Chart fornecido pelo CAP.



Deep Dive: Configurando Deployments Kyma com `values.yaml`

O poder da implantação do CAP no Kyma reside no Helm Chart gerado. O arquivo `chart/values.yaml` é o seu painel de controle central para personalizar todos os aspectos da implantação, desde recursos de container até a vinculação com serviços do BTP.

Configurações Globais

```
# Domínio do cluster e segredo do registro
domain: <cluster-domain>
imagePullSecret:
  name: <docker-secret>
```

Configurações da Aplicação (`srv`)

```
srv:
  image:
    registry: <registry-server>
  # Recursos do Kubernetes (CPU, Memória)
  resources: {}
  # Variáveis de ambiente adicionais
  env:
    MY_ENV_VAR: 1
  # Configuração de Health Probes
  health:
    liveness:
      path: /health
```

Vinculação de Serviços (`bindings`)

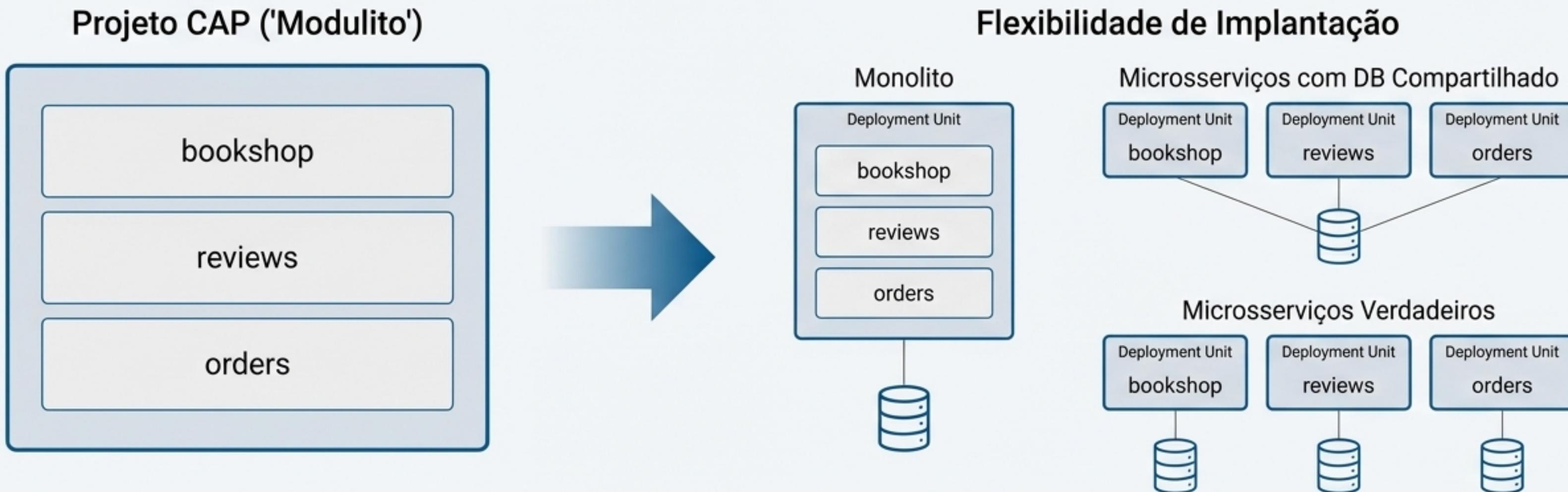
```
# Vinculação a serviços do BTP
bindings:
  db:
    serviceInstanceId: hana-instance
  # Definição de instâncias de serviço
  hana:
    serviceOfferingName: hana
    servicePlanName: hdi-shared
```

Modifique o `values.yaml` para adaptar o deployment às suas necessidades, sem precisar escrever manifestos Kubernetes do zero.

Etapa 2: Arquitetura para Escala

A Flexibilidade dos "Late-Cut Microservices"

Em vez de uma escolha binária entre monolito e **microsserviços**, o CAP promove uma abordagem flexível. Você pode desenvolver sua aplicação como um "modulito" coeso e, mais tarde, decidir como implantá-lo – tudo através de mudanças de configuração, sem alterar a lógica de negócios.



"Separe os domínios em módulos semanticamente, e decida sobre a topologia de implantação como um passo independente."

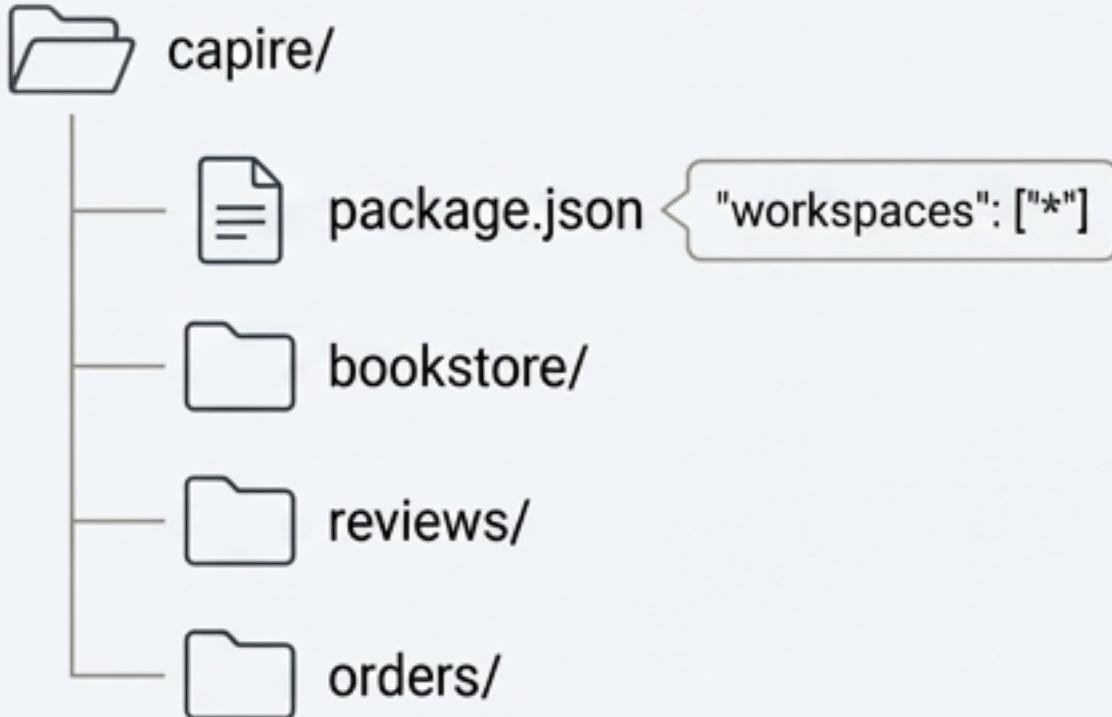
Microsserviços na Prática: Monorepo e Banco de Dados Compartilhado

Para gerenciar múltiplos serviços, usamos uma estrutura de monorepo com NPM Workspaces e Git Submodules. Para a persistência de dados, uma estratégia comum é começar com um único banco de dados compartilhado, implantado como um projeto dedicado.

Estrutura do Monorepo

Um único repositório (`capire`) contendo múltiplos projetos de microsserviços (`bookstore`, `reviews`, `orders`) como subdiretórios.

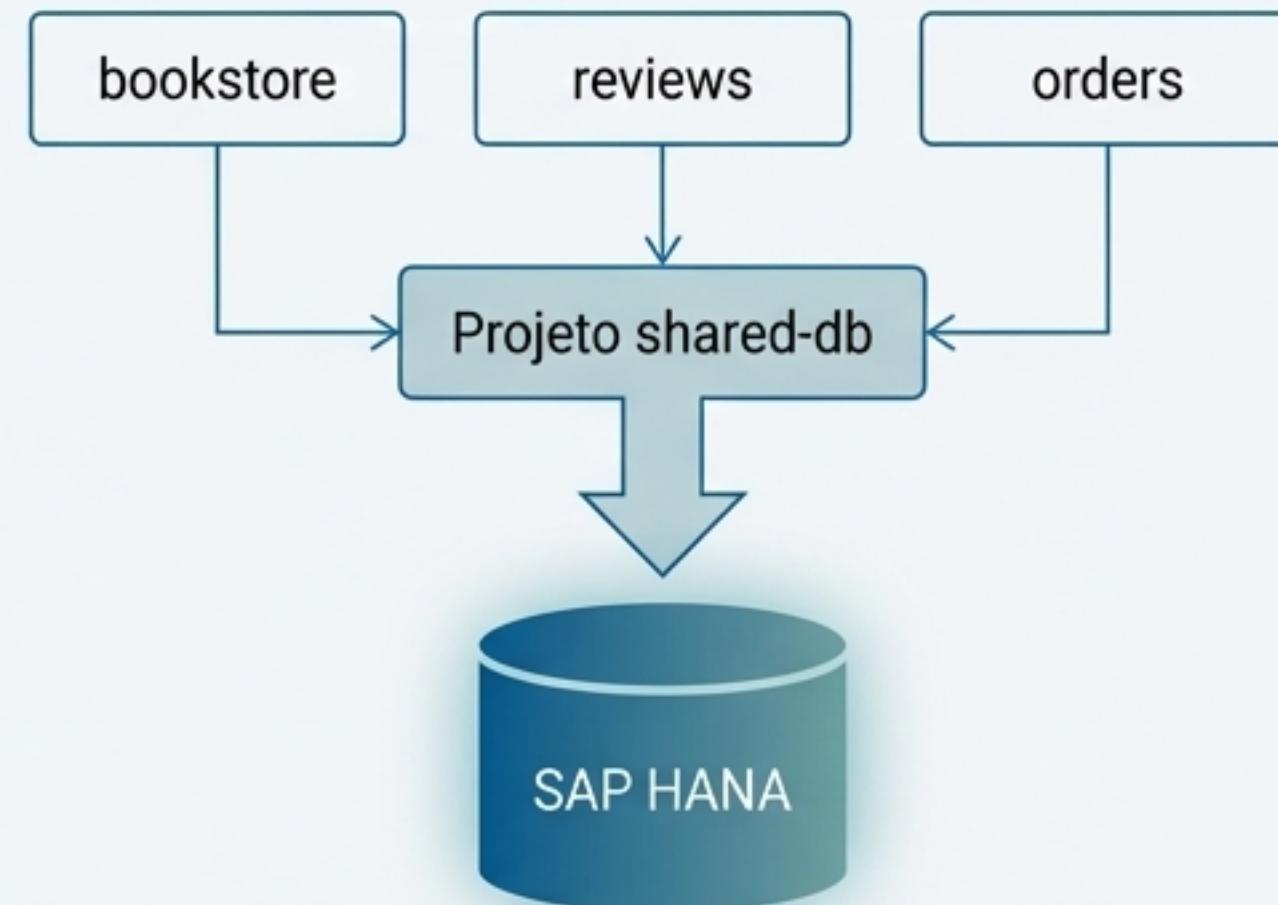
NPM Workspaces gerenciam as dependências locais entre os projetos de forma eficiente.



Padrão de Banco de Dados Compartilhado

Um projeto CAP separado (`shared-db`) é criado com o único propósito de agregar todos os modelos de dados dos outros microsserviços.

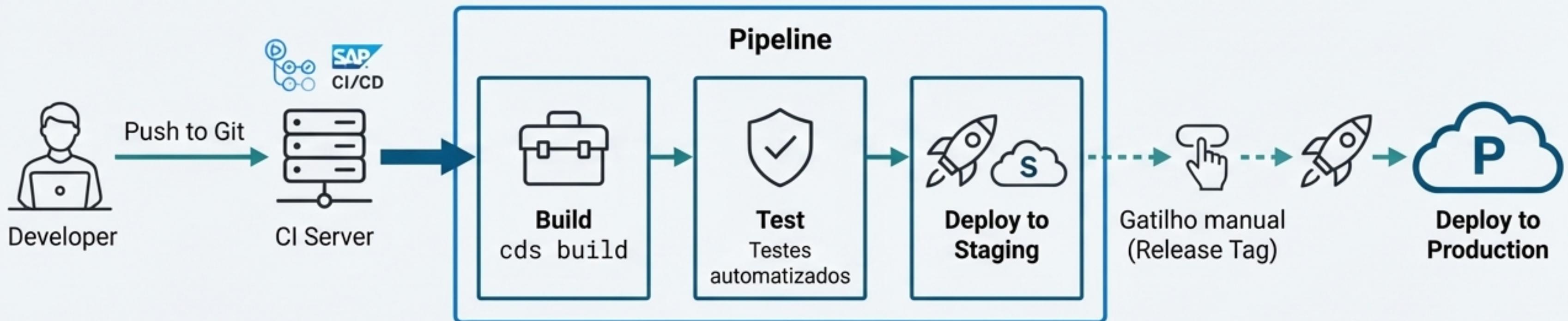
O arquivo `shared-db/db/schema.cds` usa diretivas `using from` para importar os modelos de `@capire/bookstore`, `@capire/reviews`, etc.



Etapa 3: Automação do Fluxo

De Implantações Manuais a Pipelines de CI/CD

Uma vez que sua aplicação esteja configurada para a nuvem, o próximo passo é automatizar o processo de build, teste e deploy. Pipelines de Continuous Integration e Continuous Deployment (CI/CD) garantem entregas rápidas, consistentes e confiáveis.



Opções de Ferramentas: GitHub Actions, SAP Continuous Integration and Delivery, SAP Piper.

CI/CD em Ação com GitHub Actions

O CAP oferece uma integração pronta para uso com o GitHub Actions. Com um único comando, você pode adicionar workflows pré-configurados ao seu projeto para implantar automaticamente em ambientes de Staging e Produção.

Seção 1: Passo 1: Adicionar Workflows

Use o comando `cds add github-actions` (ou `cds add gha`) no seu projeto. Isso cria os arquivos de workflow necessários.

Seção 2: Passo 2: Configurar Segredos e Variáveis

No seu repositório GitHub (Settings → Secrets and variables → Actions), configure as credenciais para o ambiente de destino.



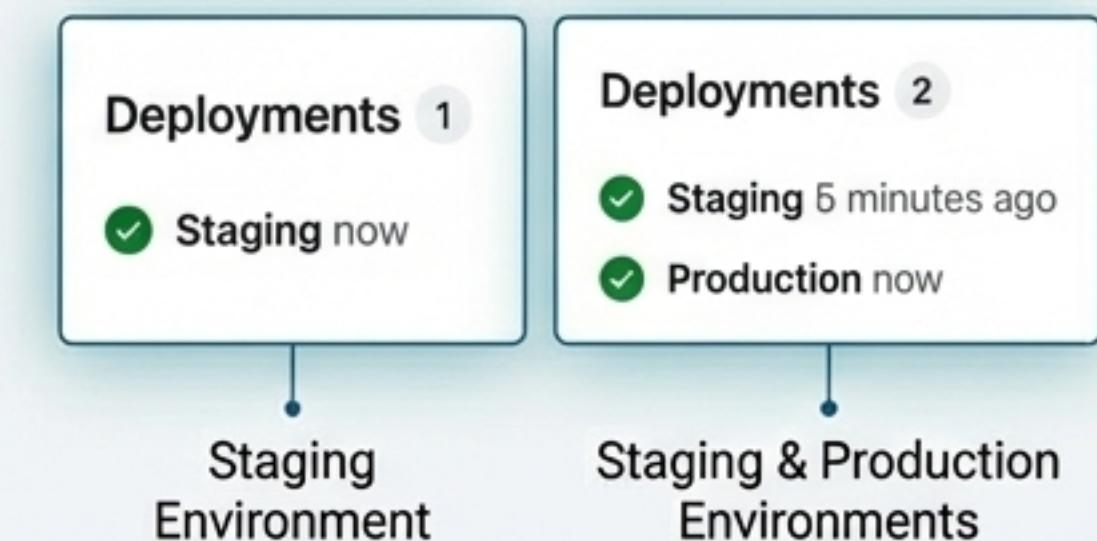
Tabela de Variáveis

Ambiente	Tipo	Nome	Exemplo
Cloud Foundry	Variable	CF_API	https://api.cf.example.com
Cloud Foundry	Variable	CF_ORG	my-org
Cloud Foundry	Variable	CF_SPACE	my-space
Cloud Foundry	Secret	CF_PASSWORD	*****
Kyma	Secret	KUBE_CONFIG	(Conteúdo do kubeconfig em Base64)



Resultado Final:

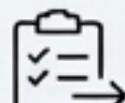
- Merges na branch `main` implantam em **Staging**.
- A criação de uma nova 'Release' no GitHub implanta em **Production**.



Etapa 4: Dominando o Processo de Build **Controle Fino com `cds build`**

O comando `cds build` é o motor que transforma seu código-fonte nos artefatos prontos para deploy. Embora os padrões sejam suficientes para muitos casos, o CAP permite que você personalize extensivamente esse processo para atender a requisitos específicos.

O que `cds build` faz?

-  **Compila Modelos CDS:** Gera artefatos como EDMX, CSN e esquemas de banco de dados (HDI).
-  **Prepara Pastas de Aplicação:** Copia os arquivos relevantes para uma pasta de destino (gen/ por padrão), criando um pacote autossuficiente.
-  **Executa Tarefas de Build (Build Tasks):** Orquestra diferentes tipos de builds (para hana, nodejs, etc.) com base na configuração do seu projeto.



Personalizando o Build com Tarefas (Tasks)

Para controlar o processo de build, você pode definir "build tasks" na seção `cds.build` do seu `package.json`. Cada tarefa especifica o tipo de build (`for`), a pasta de origem (`src`) e opções detalhadas (`options`).

```
// in package.json
"cds": {
  "build": {
    "target": "gen", // Pasta de destino global
    "tasks": [
      {
        // Tarefa para gerar artefatos SAP HANA
        "for": "hana",
        "src": "db",
        "options": {
          // Inclui modelos de 'db' e 'srv' no build do banco
          "model": ["db", "srv"]
        }
      },
      {
        // Tarefa para preparar a aplicação Node.js
        "for": "nodejs",
        "src": "srv",
        "options": {
          // Inclui modelos de 'db' e 'srv' para o serviço
          "model": ["db", "srv"]
        }
      }
    ],
    // Adicione tarefas personalizadas ou de plugins aqui
  }
}
```

Você pode usar esta estrutura como um modelo. O CAP registra as tarefas que executa no log, facilitando a cópia e a adaptação para suas necessidades.

Etapa 5: Prontidão Operacional

Sua Aplicação está Saudável? Health Checks

Uma aplicação em produção precisa comunicar seu estado de saúde para a plataforma de orquestração. O CAP fornece endpoints de health check prontos para uso, permitindo que o Cloud Foundry ou Kubernetes gerenciem suas instâncias de forma eficaz.

✗ Liveness Probe (“Você está vivo?”)

Propósito	Verifica se a aplicação ainda está funcionando.
Endpoint (Node.js)	/health
Endpoint (Java)	/actuator/health/liveness
Resultado de Falha	A plataforma reinicia o container/instância da aplicação, assumindo que ele travou.



✓ Readiness Probe (“Você está pronto?”)

Propósito	Verifica se a aplicação está pronta para aceitar novo tráfego. Útil durante a inicialização ou quando está sobrecarregada.
Endpoint (Node.js)	/health
Endpoint (Java)	/actuator/health/readiness
Resultado de Falha	A plataforma remove temporariamente a instância do平衡ador de carga até que ela se torne 'ready' novamente.

A configuração desses probes é adicionada automaticamente aos Helm Charts (`cds add kyma`) e aos descriptores de implantação quando suportado.

A Jornada Completa: Da Ideia à Operação em Escala

Percorremos o caminho completo, transformando uma simples aplicação aplicação local em uma solução de produção robusta, escalável, automatizada e monitorada.

O SAP Cloud Application Programming Model fornece as ferramentas e os padrões para guiá-lo em cada etapa desta jornada.

