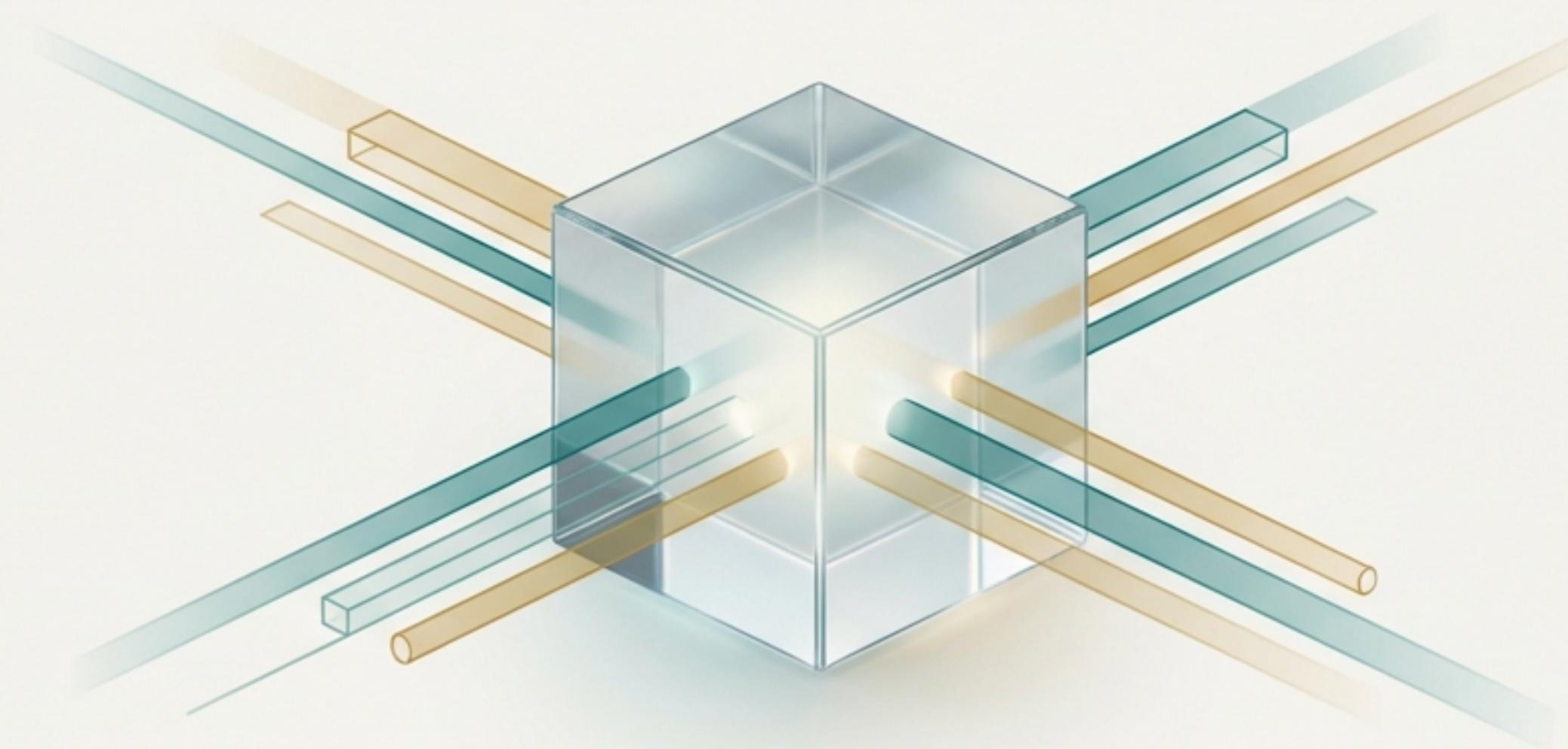


Dominando Multitenancy com SAP CAP: A Jornada SaaS Completa

Um guia prático, do conceito à nuvem, para construir e escalar suas aplicações multitenant.



Baseado no pacote `@sap/cds-mtxs`, esta apresentação guiará você através da arquitetura, implementação, teste, implantação e modernização de aplicações multitenant.

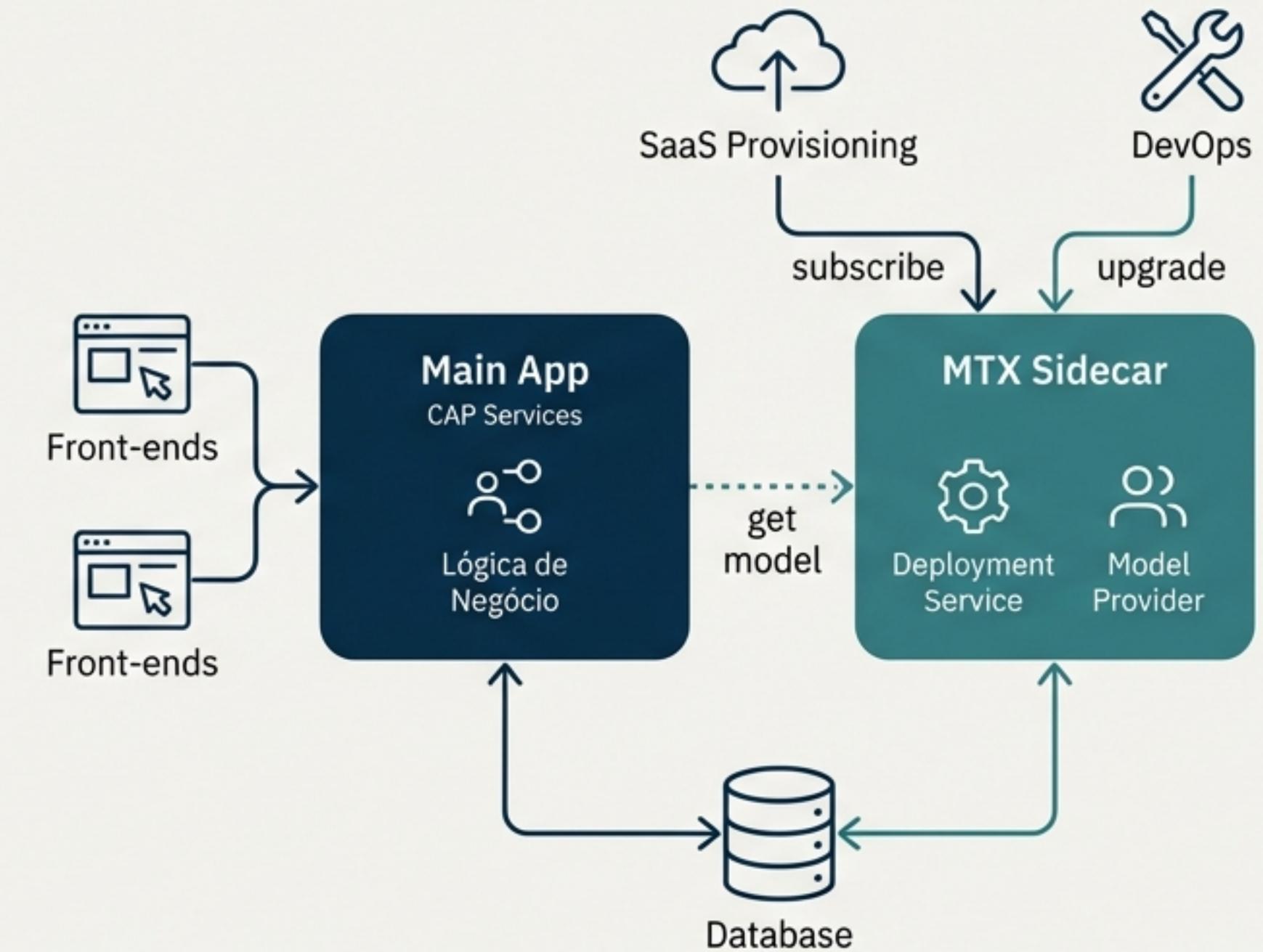
A Arquitetura Central: Multitenancy e o Padrão Sidecar

O que é Multitenancy?

A capacidade de servir múltiplos clientes (tenants) através de uma única instância da aplicação, garantindo o isolamento estrito dos dados de cada tenant. Em vez de criar recursos no deploy, eles são provisionados dinamicamente na subscrição de um novo tenant.

O Padrão Sidecar

Para escalar de forma independente e isolar tarefas de provisionamento intensivas (como `subscribe` e `upgrade`), usamos um microserviço “Sidecar”. Ele lida com a gestão dos tenants enquanto a aplicação principal foca na lógica de negócio. Esta abordagem é obrigatória para projetos Java e recomendada para Node.js em produção.



Passo 1: Habilitando a Multitenancy em seu Projeto

Pré-requisitos

Garanta que você tem a última versão do `@sap/cds-dk`.

```
npm update -g @sap/cds-dk
```

Iniciando a Aplicação

Use o `cds init` para criar um projeto de exemplo, como o 'bookshop'.

```
cds init bookshop --add sample\  
cd bookshop
```

Habilitação com um Comando

O CLI do CAP simplifica a configuração inicial.

```
cds add multitenancy
```

Habilitação com um Comando

O CLI do CAP simplifica a configuração inicial.

```
cds add multitenancy
```

O Que Acontece nos Bastidores?

Um snippet do `package.json` destacando as seções adicionadas. Principalmente, a adição da dependência `@sap/cds-mtxs` e a configuração `multitenancy: true` dentro de `cds.requires`.

Instale as novas dependências:

```
npm i
```

```
{  
  "dependencies": {  
    ...  
    "@sap/cds-mtxs": "^..."  
  },  
  "cds": {  
    "requires": {  
      "db": { ... },  
      "multitenancy": true  
    }  
  }  
}
```

Passo 2: Test-Drive Local em 4 Etapas

Simule operações SaaS (startup, subscrição, upgrade) localmente usando múltiplos terminais para cada processo.

1.



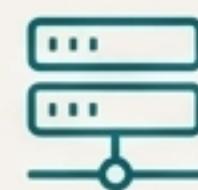
Inicie o MTX Sidecar

Responsável pela gestão dos tenants.

```
cds watch mtx/sidecar
```

Observação: O sidecar geralmente inicia na porta 4005.

2.



Inicie o App Server

Sua aplicação principal, agora ciente do sidecar.

```
cds watch --with-mtx
```

Observação: O servidor da aplicação inicia na porta 4004.

3.



Subscreva os Tenants

Crie dois tenants isolados, 't1' e 't2'.

```
cds subscribe t1 --to http://localhost:4005  
cds subscribe t2 --to http://localhost:4005
```

Observação: O comando 'cds subscribe' interage com a API do Sidecar.

4.



Faça o Upgrade de um Tenant

Aplique mudanças de modelo/dados a um tenant específico. Exemplo: Adicione um novo livro ao arquivo 'db/data/sap.capire.bookshop-Books.csv'.

```
cds upgrade t1 --at http://localhost:4005
```

Observação: O novo livro aparecerá para o tenant 't1', mas não para o 't2', demonstrando o isolamento.

Verificando o Isolamento de Dados dos Tenants

Após subscrever `t1` e `t2` e modificar um dado apenas para `t1` (por exemplo, alterando o título de um livro), podemos verificar que a mudança não afeta `t2`.

Login como `alice` (Tenant t1)

Acesse <http://localhost:4004/#Books-manage>.

Manage Books	
Title	Author
<input checked="" type="checkbox"/> Wuthering Heights (only in t1)	Emily Brontë
<input type="checkbox"/> Jane Eyre	Charlotte Brontë
<input type="checkbox"/> Wuthering Heights (only in t1)	Emily Brontë
<input type="checkbox"/> Jane Eyre	Charlotte Brontë
<input type="checkbox"/> Wuthering Heights (only in t1)	Emily Brontë

Login como `erin` (Tenant t2)

Use uma janela de navegador anônima/privada para uma nova sessão.

Manage Books	
Title	Author
<input checked="" type="checkbox"/> Wuthering Heights	Emily Brontë
<input type="checkbox"/> Jane Eyre	Charlotte Brontë
<input type="checkbox"/> Wuthering Heights (only in t1)	Emily Brontë
<input type="checkbox"/> Jane Eyre	Charlotte Brontë
<input type="checkbox"/> Wuthering Heights (only in t1)	Charlotte Brontë

O CAP, através do @sap/cds-mtxs, gerencia automaticamente conexões de banco de dados distintas (ou contêineres HDI) para cada tenant, garantindo que os dados de um cliente nunca sejam visíveis para outro.

Passo 3: Preparando a Aplicação para a Nuvem

Configurando serviços essenciais e compilando para produção.

Adicionar Dependências de Nuvem

SAP HANA Cloud & XSUAA

Para persistência de dados e autenticação/autorização.

```
cds add hana,xsuaa
```

App Router & Portal (Opcional)

Para servir UIs e atuar como ponto de entrada único.

```
cds add portal
```

Deployment Descriptor (MTA)

Para definir a estrutura de implantação.

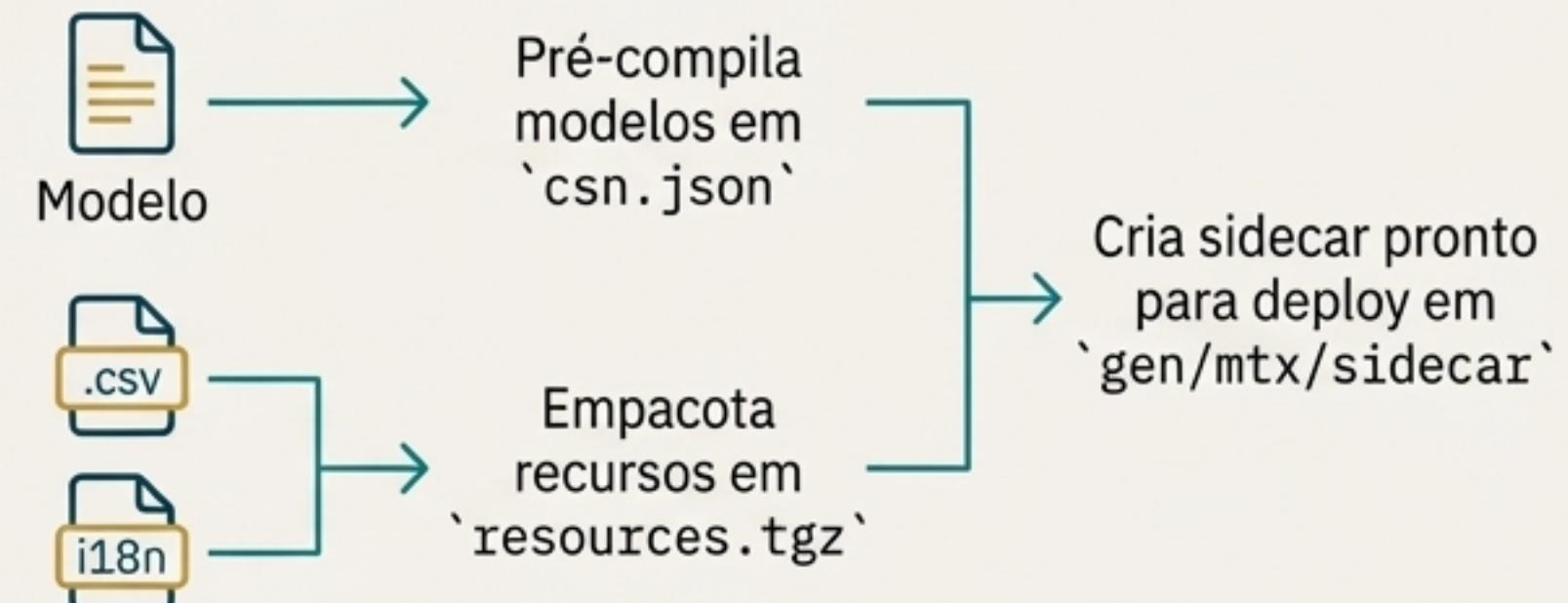
```
cds add mta
```

Compilar para Produção

O comando `cds build` prepara os artefatos para implantação, incluindo a compilação do sidecar.

```
cds build --production
```

O que o `build` faz para o Sidecar?



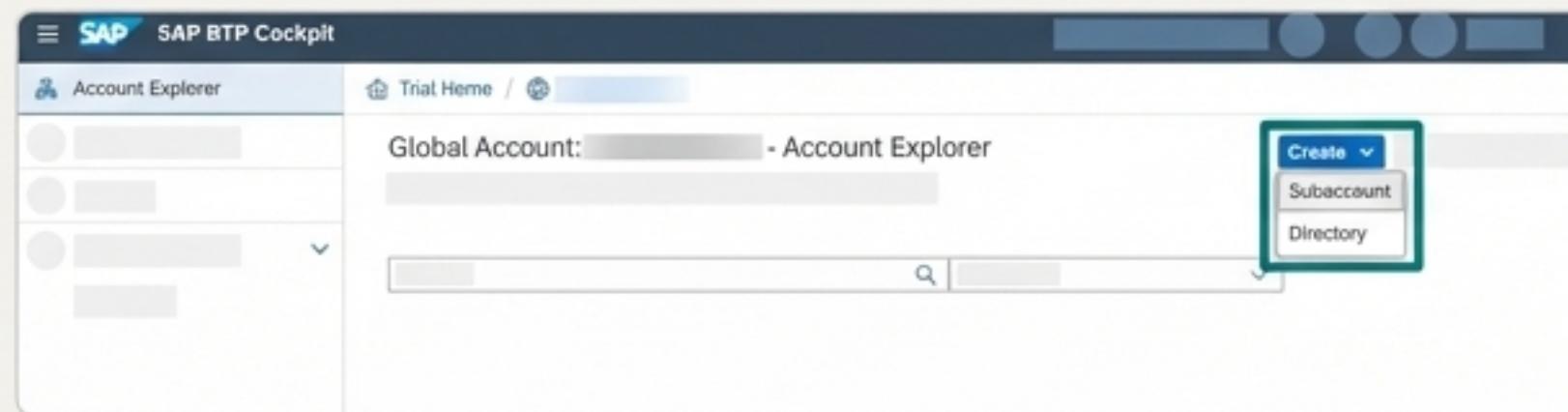
Passo 4: Implantando e Subscrevendo no SAP BTP

Um guia visual passo a passo, organizado de forma clara e sequencial, usando texto e as imagens de referência.

Etapa 1: Implantação

Com a configuração do MTA pronta, o deploy é um único comando.

`cds up`



Etapa 2: Criar uma Subconta para o Tenant

No BTP Cockpit, crie uma nova subconta que atuará como o cliente (tenant) da sua aplicação SaaS.

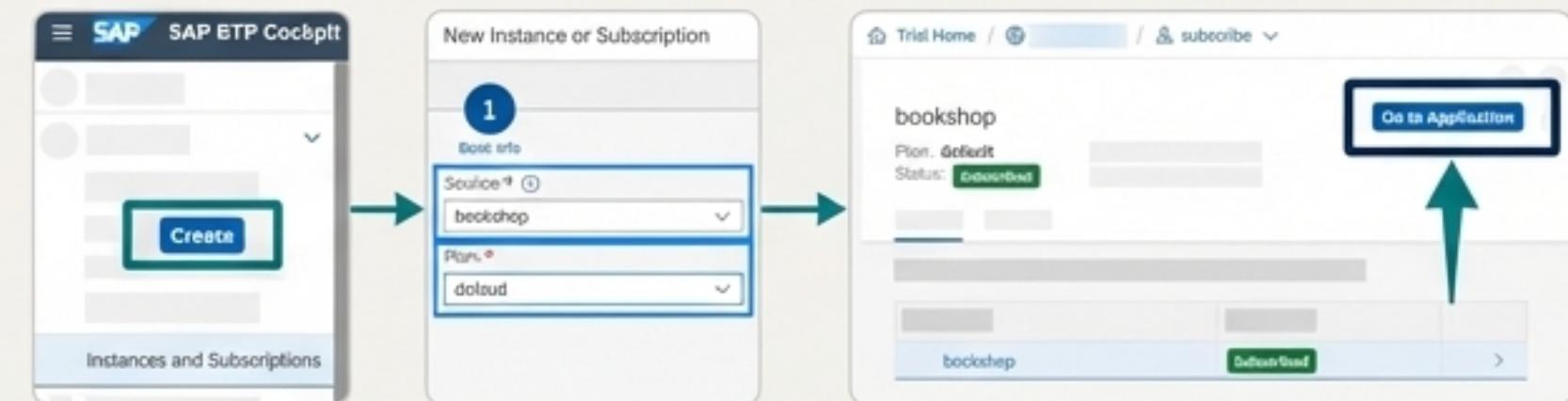
Dica: A subconta deve estar na mesma região da subconta provedora.

Etapa 3: Subscrever à Aplicação

Na subconta do cliente, navegue para 'Instances and Subscriptions' e clique em 'Create'.

Selecione sua aplicação (ex: 'bookshop') e o plano 'default'.

A aplicação aparecerá como 'Subscribed'.



Etapa 4: Acessar a Aplicação

Clique em 'Go to Application' para acessar a URL específica do tenant.

(Uma etapa de mapeamento de rota pode ser necessária, conforme detalhado no próximo slide).

Gerenciando a Aplicação SaaS na Nuvem

IBM Plex Sans com o enouorre o cornercar os subdomínios passu a zllicação ds bancos de dados dos tenants existentes euperticamente a aplicação.

Mapeamento de Rota do Tenant

Problema: Erro 404 Not Found

Após a subscrição, a URL do tenant pode resultar em um erro "404 Not Found" porque a rota específica do subdomínio ainda não foi mapeada para a aplicação.

Solução: Comando `cf map-route`

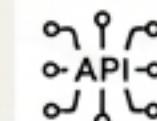
Use o comando `cf map-route` para vincular o subdomínio do tenant à sua aplicação implantada.

```
cf map-route bookshop cfapps.us10.hana.ondemand.com --  
hostname <subscriberSubdomain>-bookshop-<org>-<space>
```

Atualização do Esquema de Banco de Dados

Quando você implanta uma nova versão da sua aplicação com mudanças no modelo de dados, os bancos de dados dos tenants existentes precisam ser atualizados.

Métodos de Atualização



Invoque o endpoint de `upgrade` programaticamente.



Execute `cds-mtx upgrade <tenant|*>` via `cf run-task`.



Automatize a atualização durante o deploy, adicionando um hook do tipo `task` no seu `mta.yaml`.

```
hooks:  
- name: upgrade-all  
  type: task  
  phases: [deploy.application.before-start]  
parameters:  
  command: cds-mtx upgrade '*'
```

Além do Básico: Integrando Serviços BTP (SaaS Dependencies)

Aplicações SaaS frequentemente precisam interagir com outros serviços (Audit Log, Event Mesh, Destination Service) de forma tenant-aware. O `@sap/cds-mtxs` facilita essa integração.



Como Funciona

O CAP gerencia a criação de dependências de serviço por tenant durante o processo de subscrição.

Configuração (Node.js)

A ativação é feita de forma declarativa no `package.json` do seu sidecar.

mtx/sidecar/package.json

```
"cds": {  
  "requires": {  
    "audit-log": true,  
    "connectivity": true,  
    "destinations": true,  
    "html5-repo": true,  
    "portal": true  
  }  
}
```

Flexibilidade

Para serviços não suportados nativamente, você pode definir dependências personalizadas através de uma entrada `subscriptionDependency` ou implementando um handler de `dependencies` para total controle.

Customizando o Comportamento com Handlers de Eventos

Os serviços MTX são serviços CAP padrão. Isso significa que você pode registrar handlers de eventos `before` e `after` para customizar ou estender seu comportamento.

Onde Implementar

Adicione sua lógica em um arquivo `server.js` dentro do projeto do sidecar (`mtx/sidecar`).

Eventos Chave do Ciclo de Vida

 `subscribe`: Acionado quando um novo tenant se inscreve.

 `upgrade`: Acionado ao atualizar um tenant para uma nova versão.

 `deploy`: Acionado após a criação/atualização dos recursos do banco de dados.

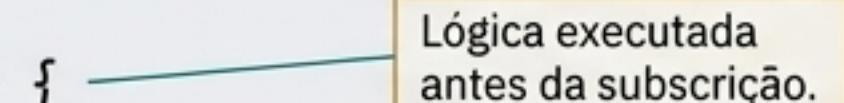
 `unsubscribe`: Acionado quando um tenant cancela a subscrição.

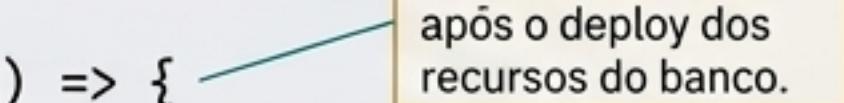
`mtx/sidecar/server.js`

```
const cds = require('@sap/cds')
cds.on('served', () => {
  const { 'cds.xt.DeploymentService': ds } = cds.services

  ds.before('subscribe', async (req) => {
    // Lógica customizada ANTES da criação do contêiner HDI
    const { tenant } = req.data
    console.log(`Subscribing tenant: ${tenant}`)
  })

  ds.after('deploy', async (result, req) => {
    // Lógica customizada APÓS o deploy no banco de dados
    const { tenant } = req.data
    // Ex: Chamar uma API externa para provisionar outros recursos
  })
})
```

 Lógica executada antes da subscrição.

 Lógica executada após o deploy dos recursos do banco.

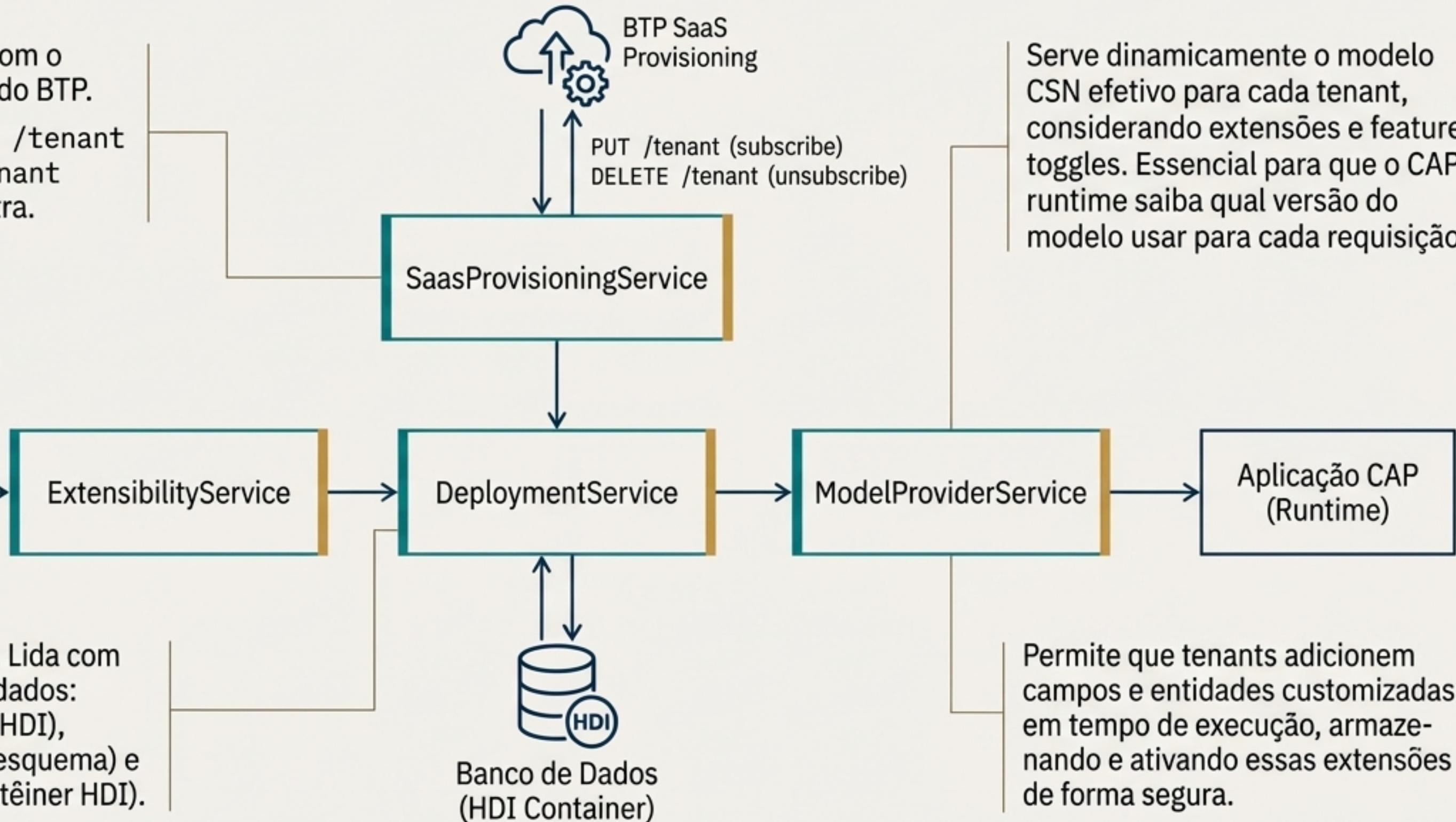
O Ecossistema de Serviços do @sap/cds-mtxs

A fachada que se integra com o SaaS Provisioning Service do BTP. Recebe os eventos de PUT /tenant (subscribe) e DELETE /tenant (unsubscribe) e os orquestra.



Enviar extensões

O "motor" da multitenancy. Lida com as operações de banco de dados: subscribe (cria contêiner HDI), upgrade (aplica deltas de esquema) e unsubscribe (remove contêiner HDI).



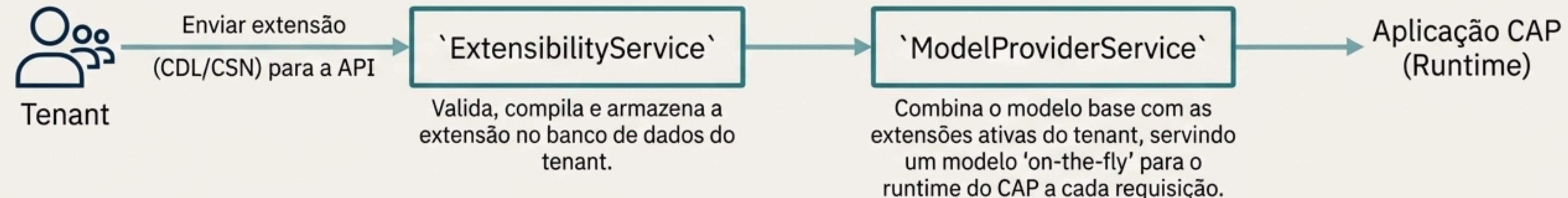
Serve dinamicamente o modelo CSN efetivo para cada tenant, considerando extensões e feature toggles. Essencial para que o CAP runtime saiba qual versão do modelo usar para cada requisição.

Permite que tenants adicionem campos e entidades customizadas em tempo de execução, armazenando e ativando essas extensões de forma segura.

Potencializando o SaaS: Extensibilidade por Tenant

Permita que seus clientes estendam o modelo de dados da sua aplicação (adicionando novos campos, por exemplo) sem alterar a base de código principal. As extensões são específicas para cada tenant.

Como Funciona



Como Funciona

Controlando as Extensões ('extension-allowlist')

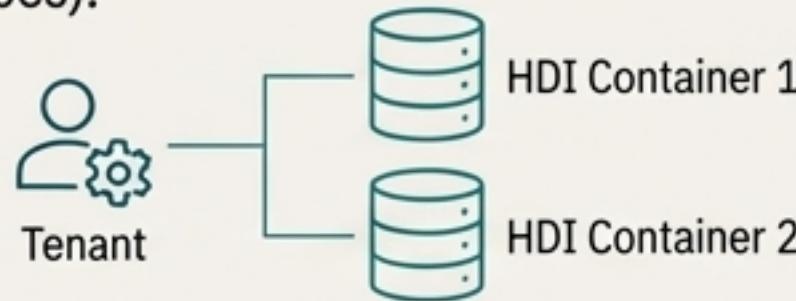
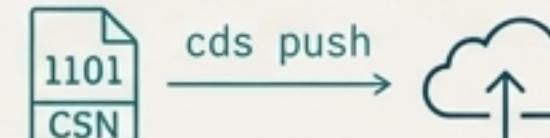
Para segurança e estabilidade, você define regras sobre o que pode ser estendido.

```
"cds.xt.ExtensibilityService": {  
    "extension-allowlist": [  
        { // Permite até 2 novos campos em entidades do namespace my.bookshop  
            "for": ["my.bookshop"],  
            "kind": "entity",  
            "new-fields": 2,  
            "fields": ["description"] // Apenas o campo 'description' pode ser estendido  
        },  
        { // Permite até 2 novas entidades no serviço CatalogService  
            "for": ["CatalogService"],  
            "new-entities": 2  
        }  
    ]  
}
```

Modernizando sua Aplicação: Migrando do `@sap/cds-mtx`

O pacote **@sap/cds-mtx** (antigo MTX) foi substituído pelo **@sap/cds-mtxs**, que oferece uma arquitetura mais simples e eficiente.

Principais Diferenças Funcionais

Antigo (`@sap/cds-mtx`)	Novo (`@sap/cds-mtxs`)
Persistência Simplificada Usava dois contêineres HDI por tenant (um para dados, um para metadados/extensões). 	Usa um único contêiner por tenant. Metadados e extensões são armazenados junto com os dados da aplicação. Um tenant especial t0 é usado para dados de runtime (ex: logs de jobs). 
Gerenciamento de Extensões Armazenava o código-fonte (CDL) das extensões e as compilava no servidor. 	Armazena apenas o CSN compilado. O build da extensão é feito localmente antes do deploy, usando cds push. 
Segurança Roles como ExtendCDS e ExtendCDSdelete.	Uma única role consolidada: cds.ExtensionDeveloper.

O Roteiro da Migração em 3 Passos

1 Passo 1: Adaptar a Configuração do Projeto

- Remover o pacote antigo: `npm remove @sap/cds-mtx`
- Adicionar o novo pacote: `npm add @sap/cds-mtxs`
- Atualizar `package.json` / `.cdsrc.json`: Adicionar `"multitenancy": true` (e `"extensibility": true`, se aplicável) na seção `cds.requires`.
- Atualizar `xs-security.json`: Substituir as roles antigas por `cds.ExtensionDeveloper`.

2 Passo 2: Executar o Script de Migração

O `@sap/cds-mtxs` inclui um script para migrar os dados existentes (metadados de subscrição e extensões) para o novo formato de persistência.

```
cds migrate "*" --profile hybrid,production --resolve-bindings
```

Use a opção `--dry` para um teste sem alterações no banco. O script pode ser automatizado como uma task do Cloud Foundry no `mta.yaml`.

3 Passo 3: Verificar e Gerenciar Extensões Migradas

Verifique a tabela `CDS_XT_EXTENSIONS` no contêiner HDI do tenant para confirmar que as extensões foram migradas com a tag `migrated`.

CDS_XT_EXTENSIONS			
	ID	Tag	CDN
1	e0f1967bc-cf08-4l27-84d5-aAAdeLaceoba	tag1	[{"requires":[], "definitions":[]}, {"extension": "sap."}]
2	a275ecb3-e575-4888-8080-0256455138f5	tag2	[{"requires":[], "definitions":[]}, {"extension": "sap."}]

Use o script com a opção `-d <diretório>` para salvar o código-fonte das extensões antigas localmente. É crucial versionar esses projetos em um repositório, pois o novo MTX não armazena mais o código-fonte.

Sua Jornada SaaS com CAP: Simples, Robusta e Extensível



Pilar 1: Desenvolvimento Acelerado

Habilite a multitenancy com um comando e use o test-drive local para um ciclo de desenvolvimento rápido e eficiente.

Pilar 2: Arquitetura Robusta

O padrão sidecar garante escalabilidade e isolamento, enquanto a integração nativa com serviços BTP constrói soluções completas.

Pilar 3: Flexibilidade e Controle

Use handlers para customizar o ciclo de vida e a `extension-allowlist` para oferecer extensibilidade de forma segura e controlada.

Pilar 4: Modernização Clara

Uma via de migração definida e ferramental de suporte garantem que sua aplicação evolua com a plataforma.

O SAP Cloud Application Programming Model, com o pacote '@sap/cds-mtxs', oferece uma base completa e integrada para construir a próxima geração de aplicações SaaS na BTP.