Contratos versionados

Planeje a convivência entre versões

Pré-requisitos

Conheça eventos, campos obrigatórios e a relação entre produtor e consumidor. O exercício usa Python 3 e fastavro para codificar e ler um evento de confirmação. O princípio também se aplica a contratos HTTP, mas as regras concretas dependem do formato e dos clientes.

O que um contrato promete

O contrato inclui estrutura e significado. Em ReservaConfirmada, reservaId identifica a reserva confirmada. Se o mesmo campo passar a identificar o pagamento, o formato pode continuar válido e a integração ainda assim quebrar. Documente tipos, significado, valores aceitos, unidade e condições em que o evento é emitido.

Dê nome à direção da compatibilidade [1]

No exercício, um leitor novo deve conseguir ler eventos produzidos pelo escritor antigo, inclusive os já armazenados. Um leitor antigo também deve conseguir ler os eventos novos durante a migração. Teste cada direção separadamente. A ordem de implantação depende de quais direções você consegue preservar.

Evolua o esquema conforme o formato

No Avro, a resolução usa o esquema do escritor e o do leitor. Campos do escritor sem correspondente no leitor são ignorados. Se o leitor espera um campo ausente no escritor, ele usa o valor padrão indicado no próprio esquema do leitor ou gera erro se o padrão estiver ausente. Essas regras valem para a resolução Avro deste exemplo.

Planeje uma mudança incompatível

Se precisar retirar ou redefinir campos usados por consumidores, introduza um contrato separado e planeje a transição. Inventarie consumidores e eventos históricos, defina como observar adesão e mantenha a leitura antiga até que os dependentes migrem ou que o período acordado termine.

Contratos versionados

Exercício resolvido • Acrescente a origem do evento

Enunciado e dados de partida

A v1 registra somente reservald. A v2 acrescenta origem. Eventos históricos não têm origem. Defina ‘desconhecida’ como padrão de leitura e teste um evento antigo R45 e um novo R46 com origem portal.

1. Salve contratos.py

```
from io import BytesIO
from fastavro import schemaless_reader, schemaless_writer

def esquema(*extras):
    return {
        "type": "record",
        "name": "ReservaConfirmada",
        "fields": [{"name": "reservaId", "type": "string"}, *extras],
    }

v1 = esquema()
v2 = esquema({"name": "origem", "type": "string", "default": "desconhecida"})

def ler(escritor, leitor, evento):
    fluxo = BytesIO()
    schemaless_writer(fluxo, escritor, evento)
    fluxo.seek(0)
    return schemaless_reader(fluxo, escritor, leitor)

antigo = {"reservaId": "R45"}
novo = {"reservaId": "R46", "origem": "portal"}
assert ler(v1, v2, antigo) == {**antigo, "origem": "desconhecida"}
assert ler(v2, v1, novo) == {"reservaId": "R46"}
print("Leituras nas duas direcoes conferidas")
```

Entenda a leitura

BytesIO armazena o conteúdo binário em memória. Cada leitura recebe os dois esquemas explicitamente. Em um fluxo real, o consumidor precisa identificar o esquema que escreveu a mensagem, por exemplo por um identificador associado ao envelope.

Contratos versionados

Confirme as duas direções e a evolução

2. Prepare o ambiente e execute

```
python -m venv .venv
.venv/Scripts/python.exe -m pip install fastavro==1.12.1
.venv/Scripts/python.exe contratos.py
```

Leia o resultado

Execute os comandos no PowerShell, na pasta do arquivo. A saída indica que as duas leituras passaram. R45 recebe origem desconhecida no leitor v2. R46 mantém reservald e o leitor v1 ignora origem. O valor padrão de leitura não afirma que o evento antigo veio do portal.

Caso de erro • Campo novo sem valor padrão

Retire default apenas da definição de v2 e execute de novo. A leitura da v1 pelo leitor v2 falha porque o evento antigo não tem origem. O caminho inverso ainda pode passar. Para comprová-lo isoladamente, rode somente a segunda asserção. Restaure o padrão antes de repetir o teste completo.

Critérios para conferir

Guarde exemplos de dados anteriores e compare os resultados nas duas direções. Mesmo após a aprovação estrutural, confira se os valores novos podem ser aceitos pelo fluxo de negócio. Em produção também verifique a associação ao esquema correto e o comportamento dos consumidores implantados.

Referências primárias

[1] Apache Avro 1.12.0 • Resolução de esquemas

Contratos versionados

Verifique compatibilidade com consumidores reais

Identifique o contrato que o cliente usa

A compatibilidade envolve tipos, campos obrigatórios e significado. Um campo novo costuma ser aceito por clientes que toleram propriedades extras. Alterar o tipo ou significado de um campo existente pode quebrar até clientes que continuem recebendo 200.

Exercício adicional

A API retorna horario como texto. A equipe quer substituí-lo por um objeto com início e fim. Como migrar?

Resolução

Mantenha horario na versão atual e lance a representação nova em uma versão distinta. Teste as duas com consumidores representativos e acompanhe a migração antes de retirar a anterior.

Ao assistir ao vídeo

Observe como o exemplo mantém rotas de mais de uma versão durante a migração.

Vídeo complementar

Implementing Modern API Versioning in .NET (em inglês)