**SpiceDB + Podman + VS Code 完整操作手冊**

**🎯 目標**

在 VS Code 中使用 Podman 開發 SpiceDB 應用程式

**📋 前置需求**

* ✅ 已安裝 Podman
* ✅ 已安裝 VS Code
* ✅ 確認 Podman 可正常運行：podman --version

**🚀 步驟 1：啟動 SpiceDB 服務**

# 啟動 SpiceDB 容器

podman run -d --name spicedb \

-p 50051:50051 \

-p 8080:8080 \

quay.io/authzed/spicedb serve \

--grpc-preshared-key "your-secret-key" \

--datastore-engine memory

# 檢查容器狀態

podman ps

# 查看日誌（如果有問題）

podman logs spicedb

# 停止服務（需要時）

podman stop spicedb

# 刪除容器（需要時）

podman rm spicedb

**重要說明：**

* 50051：gRPC API 端口
* 8080：HTTP API 端口
* memory：資料存在記憶體，重啟會遺失
* your-secret-key：連接密鑰，程式中要用相同的

**🛠️ 步驟 2：VS Code 環境設定**

**2.1 安裝必要擴展**

在 VS Code 中安裝：

* **Dev Containers** (Microsoft)

**2.2 設定 VS Code 使用 Podman**

按 Ctrl+Shift+P → 搜尋 "settings" → 選 "Preferences: Open Settings (JSON)"

加入：

{

"dev.containers.dockerPath": "podman"

}

**📁 步驟 3：創建專案**

**3.1 創建專案資料夾**

mkdir spicedb-demo

cd spicedb-demo

code .

**3.2 創建 .devcontainer/devcontainer.json**

**注意：需要手動創建，擴展不會自動產生**

{

"name": "SpiceDB Python 開發環境",

"image": "python:3.11",

"customizations": {

"vscode": {

"extensions": [

"ms-python.python",

"ms-python.pylint"

]

}

},

"postCreateCommand": "pip install authzed grpcio",

"forwardPorts": [50051, 8080],

"remoteUser": "root"

}

**3.3 創建 requirements.txt**

authzed==0.8.0

grpcio==1.54.0

**💻 步驟 4：進入容器開發環境**

1. 按 Ctrl+Shift+P
2. 搜尋 "reopen"
3. 選擇 "Dev Containers: Reopen in Container"
4. 等待容器建立完成

**VS Code 會自動：**

* 下載 Python 容器
* 安裝 Python 擴展
* 執行 pip install authzed grpcio

**🐍 步驟 5：SpiceDB 測試程式**

創建 spicedb\_test.py：

"""

SpiceDB Python 測試程式

"""

from authzed.api.v1 import Client

from authzed.api.v1.schema\_pb2 import WriteSchemaRequest

from authzed.api.v1.permission\_service\_pb2 import (

WriteRelationshipsRequest,

CheckPermissionRequest,

LookupResourcesRequest

)

from authzed.api.v1.core\_pb2 import RelationshipUpdate, Relationship, ObjectReference

import asyncio

class SpiceDBDemo:

def \_\_init\_\_(self):

# 連接 SpiceDB（密鑰要與 podman run 時一致）

self.client = Client("localhost:50051", "your-secret-key")

async def setup\_schema(self):

"""設定權限模式"""

print("📝 設定權限模式...")

schema = """

definition user {}

definition document {

relation owner: user

relation viewer: user

permission view = viewer + owner

permission edit = owner

}

definition folder {

relation owner: user

relation viewer: user

relation document: document

permission view = viewer + owner

permission manage = owner

}

"""

try:

await self.client.WriteSchema(WriteSchemaRequest(schema=schema))

print("✅ 權限模式設定成功！")

except Exception as e:

print(f"❌ 設定模式失敗: {e}")

async def add\_relationships(self):

"""添加權限關係"""

print("🔗 添加權限關係...")

relationships = [

# Alice 擁有 document1

Relationship(

resource=ObjectReference(object\_type="document", object\_id="doc1"),

relation="owner",

subject=ObjectReference(object\_type="user", object\_id="alice")

),

# Bob 可以查看 document1

Relationship(

resource=ObjectReference(object\_type="document", object\_id="doc1"),

relation="viewer",

subject=ObjectReference(object\_type="user", object\_id="bob")

),

# Charlie 擁有 document2

Relationship(

resource=ObjectReference(object\_type="document", object\_id="doc2"),

relation="owner",

subject=ObjectReference(object\_type="user", object\_id="charlie")

),

]

updates = [RelationshipUpdate(

operation=RelationshipUpdate.Operation.OPERATION\_CREATE,

relationship=rel

) for rel in relationships]

try:

await self.client.WriteRelationships(WriteRelationshipsRequest(updates=updates))

print("✅ 權限關係添加成功！")

except Exception as e:

print(f"❌ 添加關係失敗: {e}")

async def check\_permissions(self):

"""檢查權限"""

print("🔍 檢查權限...")

test\_cases = [

("alice", "doc1", "view", "Alice 能否查看 doc1？"),

("alice", "doc1", "edit", "Alice 能否編輯 doc1？"),

("bob", "doc1", "view", "Bob 能否查看 doc1？"),

("bob", "doc1", "edit", "Bob 能否編輯 doc1？"),

("charlie", "doc1", "view", "Charlie 能否查看 doc1？"),

("charlie", "doc2", "edit", "Charlie 能否編輯 doc2？"),

]

for user, resource, permission, description in test\_cases:

try:

response = await self.client.CheckPermission(CheckPermissionRequest(

resource=ObjectReference(object\_type="document", object\_id=resource),

permission=permission,

subject=ObjectReference(object\_type="user", object\_id=user)

))

result = "✅ 允許" if response.permissionship == 1 else "❌ 拒絕"

print(f" {description} {result}")

except Exception as e:

print(f" {description} ⚠️ 錯誤: {e}")

async def lookup\_resources(self):

"""查找用戶可以訪問的資源"""

print("📋 查找 Alice 可以查看的文件...")

try:

resources = []

async for response in self.client.LookupResources(LookupResourcesRequest(

resource\_object\_type="document",

permission="view",

subject=ObjectReference(object\_type="user", object\_id="alice")

)):

resources.append(response.resource\_object\_id)

print(f" Alice 可以查看的文件: {resources}")

except Exception as e:

print(f" 查找失敗: {e}")

async def main():

"""主函數"""

print("🚀 SpiceDB Python 測試開始")

print("=" \* 50)

demo = SpiceDBDemo()

try:

await demo.setup\_schema()

print()

await demo.add\_relationships()

print()

await demo.check\_permissions()

print()

await demo.lookup\_resources()

print()

print("=" \* 50)

print("🎉 測試完成！")

except Exception as e:

print(f"💥 程式執行失敗: {e}")

print("請確認 SpiceDB 服務是否正在運行")

if \_\_name\_\_ == "\_\_main\_\_":

# 測試連接

print("🔌 測試 SpiceDB 連接...")

try:

client = Client("localhost:50051", "your-secret-key")

print("✅ 連接成功！")

print()

# 執行主程式

asyncio.run(main())

except Exception as e:

print(f"❌ 連接失敗: {e}")

print("請先啟動 SpiceDB 服務")

**🏃 步驟 6：執行測試**

在 VS Code 容器終端中：

# 方法 1：命令列執行

python spicedb\_test.py

# 方法 2：VS Code 中按 F5

# 或右鍵選擇 "Run Python File in Terminal"

**預期輸出：**

🔌 測試 SpiceDB 連接...

✅ 連接成功！

🚀 SpiceDB Python 測試開始

==================================================

📝 設定權限模式...

✅ 權限模式設定成功！

🔗 添加權限關係...

✅ 權限關係添加成功！

🔍 檢查權限...

Alice 能否查看 doc1？ ✅ 允許

Alice 能否編輯 doc1？ ✅ 允許

Bob 能否查看 doc1？ ✅ 允許

Bob 能否編輯 doc1？ ❌ 拒絕

Charlie 能否查看 doc1？ ❌ 拒絕

Charlie 能否編輯 doc2？ ✅ 允許

📋 查找 Alice 可以查看的文件...

Alice 可以查看的文件: ['doc1']

==================================================

🎉 測試完成！

**🔧 常見問題解決**

**Q1: 連接失敗 "connection refused"**

# 檢查 SpiceDB 是否運行

podman ps

# 如果沒有運行，重新啟動

podman run -d --name spicedb -p 50051:50051 quay.io/authzed/spicedb serve --grpc-preshared-key "your-secret-key" --datastore-engine memory

**Q2: VS Code 找不到 Podman**

# 確認 Podman 在 PATH 中

which podman

podman --version

# 在 VS Code 設定中確認

"dev.containers.dockerPath": "podman"

**Q3: Python 套件安裝失敗**

# 在容器中手動安裝

pip install --upgrade pip

pip install authzed grpcio

**Q4: 權限模式設定失敗**

* 檢查 SpiceDB 服務是否正在運行
* 確認密鑰是否正確
* 查看 SpiceDB 日誌：podman logs spicedb

**Q5: 容器無法啟動**

# 重建容器

podman stop spicedb

podman rm spicedb

# 重新執行 podman run 命令

**📝 重要概念說明**

**SpiceDB 的 datastore 儲存：**

* **權限模式 (Schema)**：定義資源類型和權限規則
* **關係數據 (Relationships)**：具體的權限關係（誰對什麼有什麼權限）
* **版本控制信息**：每次變更的版本號
* **緩存數據**：查詢結果緩存

**記憶體模式 vs 持久化：**

* **Memory 模式**：重啟後資料消失，適合測試
* **PostgreSQL 模式**：資料永久保存，適合生產環境

**基本權限概念：**

* **Subject**：權限主體（使用者）
* **Resource**：資源（文件、資料夾等）
* **Relation**：關係（owner、viewer）
* **Permission**：權限（view、edit）

**🔄 快速重啟流程**

如果需要重新開始：

# 1. 停止並刪除 SpiceDB 容器

podman stop spicedb

podman rm spicedb

# 2. 重新啟動 SpiceDB

podman run -d --name spicedb -p 50051:50051 quay.io/authzed/spicedb serve --grpc-preshared-key "your-secret-key" --datastore-engine memory

# 3. 在 VS Code 中重新執行 Python 程式

python spicedb\_test.py

**🎯 下一步開發建議**

1. **學習 SpiceDB Schema 語法**：定義更複雜的權限模型
2. **整合到實際應用**：將權限檢查加入到 Web API 中
3. **使用持久化存儲**：改用 PostgreSQL 等資料庫
4. **性能優化**：使用快取和批次操作
5. **安全性**：使用更安全的認證方式

**📞 參考資源**

* SpiceDB 官方文檔：https://docs.authzed.com/
* Podman 官方文檔：https://podman.io/
* VS Code Dev Containers：https://code.visualstudio.com/docs/devcontainers/containers

**🎉 完成！現在你可以在公司離線環境中按照這份手冊進行開發了！**