Université IBM i

19 et 20 novembre 2024

IBM Innovation Studio Paris

uii2024
#ibmi
#uii2024

S29 – MAPEPIRE : le nouveau client pour se connecter à l'IBM i

20 novembre 10:15 - 11:15

Gautier Dumas
CFD-Innovation
gdumas@cfd-innovation.fr

IBM

common FRANCE

C4D

〈INNOVATION〉

- Conseil
- Formation
- Développement

ASSISTANCE

- Web et Open Source
  - php
  - git
  - open source
  - node.js
- Communication
  - OpenSSL – Cryptography and SSL/TLS Toolkit
  - SOAP / REST – Web Services
  - 
- Valorisation des données
  - SYD
  - SQL
  - IBM DB2

Université IBM i

19 et 20 novembre 2024

IBM i
continuous innovation
continuous integration

IBM

Mapepire

Qu'est-ce que c'est ?

Pourquoi ?

Comment ?

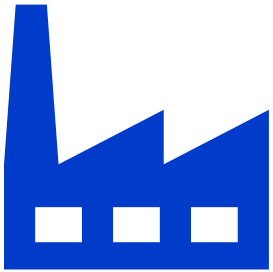
Qu'est-ce que Mapepire ? [MAH-pup-ee]

- Une couche d'accès base de données,
  - avec une approche de **simplicité** et de **performances**
  - **Cloud-Compatible**
    - Environnements à la volée ; scalable ; peu de prérequis ; multi-environnements

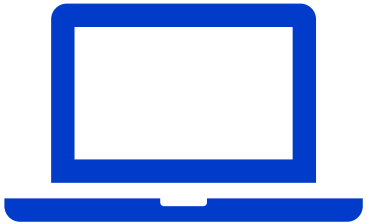
- S'appuyant sur une communication Web Socket sécurisée
- Permettant de simplifier l'intégration de DB2 for i dans le développement d'applications modernes utilisant
  - Node.js
  - Python
  - Java
  - PHP
  - .NET Core
  - …

- https://mapepire-ibmi.github.io/

Actuellement en Technology Preview



Déjà utilisé en production !

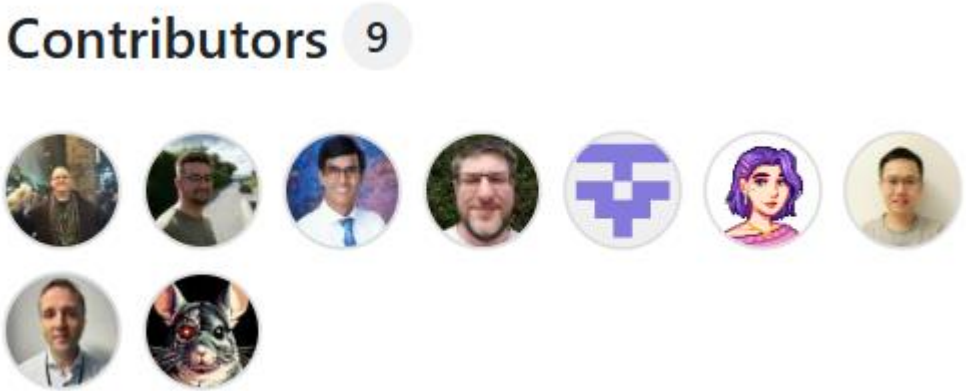


Technology Preview

=> les fonctionnalités sont complètes et prêtes à l'emploi ; des évolutions fréquentes et des améliorations en cours suite aux premiers retours et avant un déploiement à plus grande échelle

Les contributeurs actuels

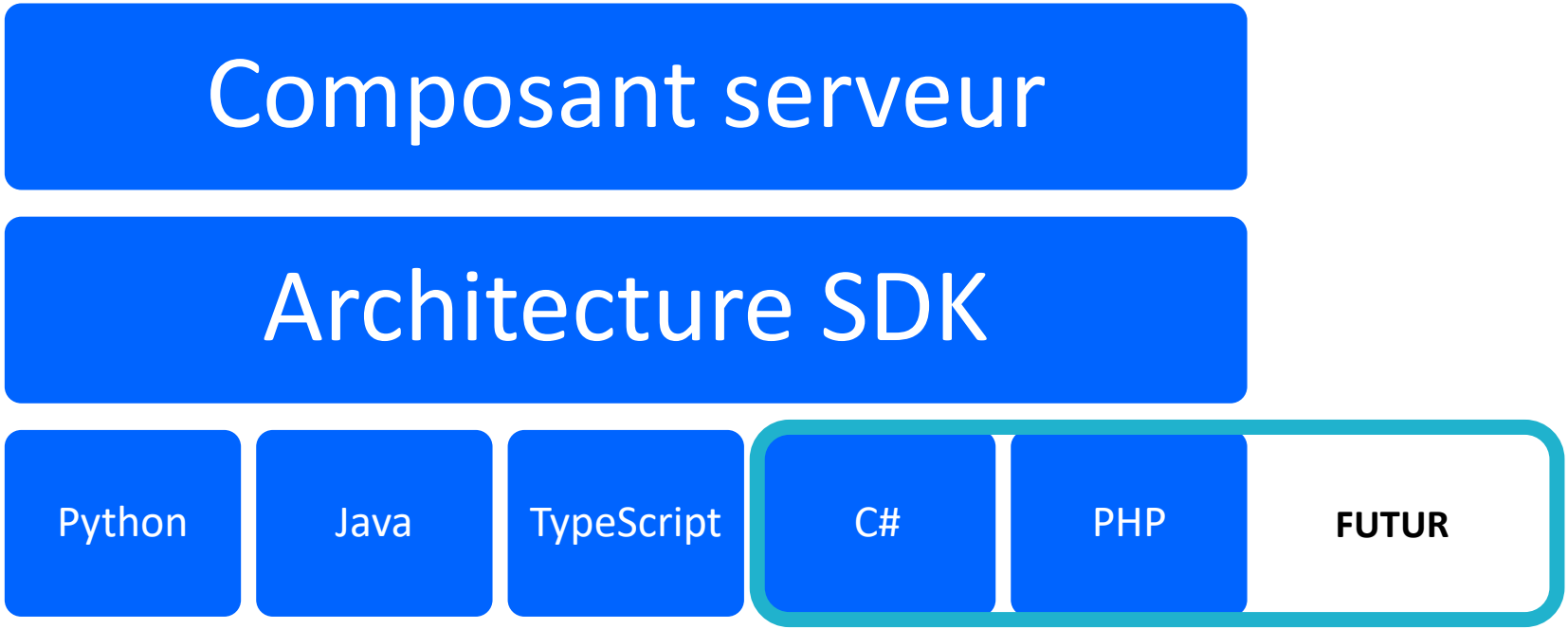
- Jesse Gorzinski (ThePrez)
- Liam Barry Alan (worksofliam)
- Sanjula Ganepola (SanjulaGanepola)
- Et plus..



À l'origine de Mapepire

Janvier 2020	Février 2022	Mars 2022	Juillet 2023	Aout 2024
• Intégration du support basique DB2 dans **VSCode "Code for IBM i"**	• Début des travaux sur le composant serveur pour alimenter des fonctionnalités DB2 dans VSCode	• Première release de l'extension VSCode **Db2 for i**	• Sortie de la v0.3.0 de l'extension VSCode Db2 for i (première version exploitant le composant serveur)	• Naissance de Mapepire

Composants Mapepire



… Mais qu'est-ce que c'est ?

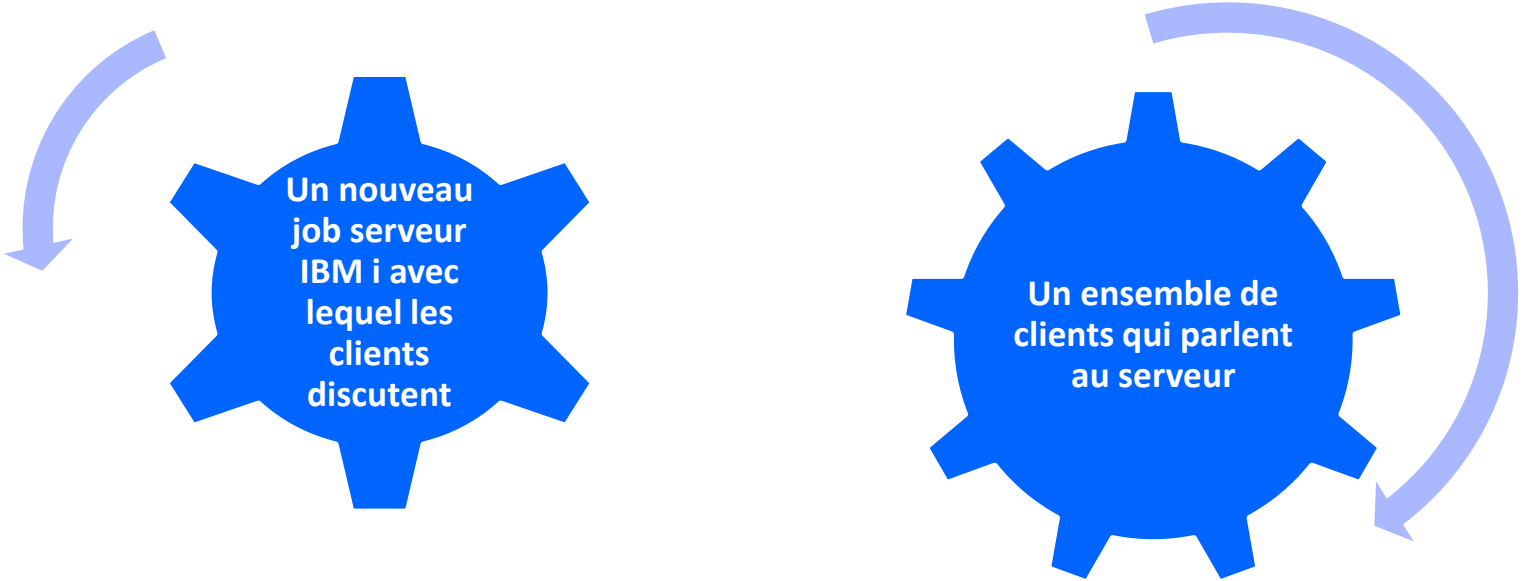
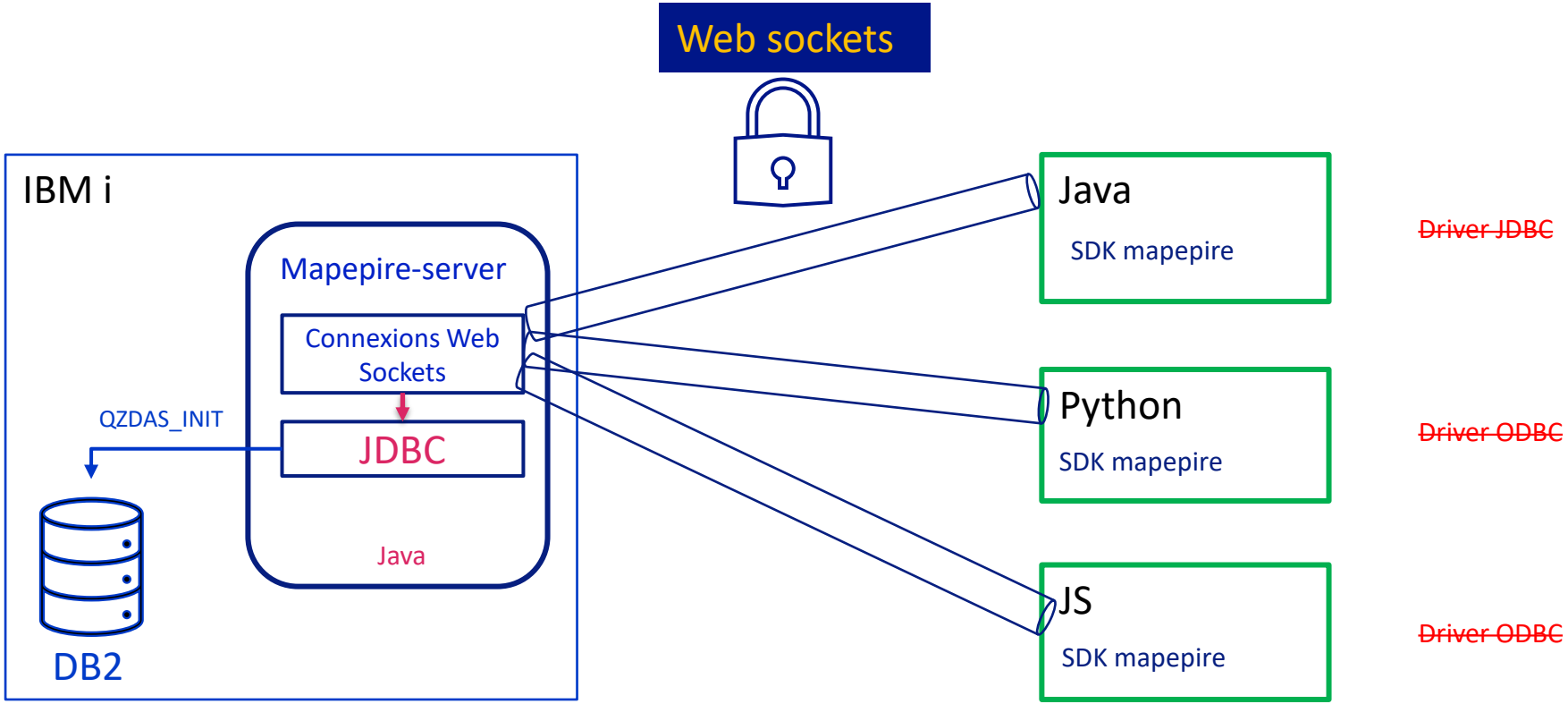


Schéma Mapepire



IBM i
continuous innovation
continuous integration

Mapepire

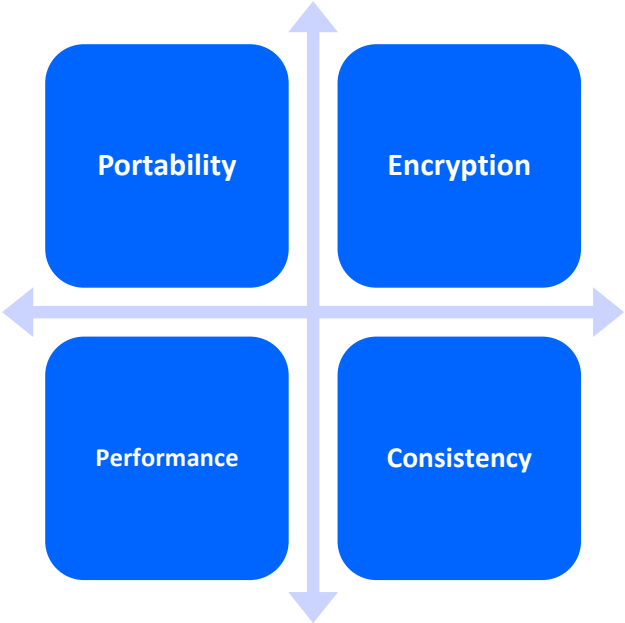
Qu'est-ce que c'est ?

Pourquoi ?

Comment ?

Principes fondamentaux

- Un SDK (*Software Development Kit*) cohérent dans tous les langages
- Un minimum de dépendances
- Aucune dépendance native (par exemple : driver) nécessaire sur le client
- La communication avec le serveur se fait via un seul port
- Les données sont toujours cryptées



Mapepire vs JDBC and ODBC

	JDBC	ODBC	Mapepire
Needs only a single port			✅
Data is always encrypted			✅
Manageable via system exit points	✅	✅	✅
Enhanced CCSID support	✅		✅
Runs in WatsonX.ai Jupyter notebooks			✅
Runs in lightweight containers (for instance Alpine Linux)	✅		✅
Directly supports multiple client languages			✅

Le plus gros avantage de Mapepire … La portabilité !!

	JDBC	ODBC	Mapepire
Runs in WatsonX.ai Jupyter notebooks	❌	❌	✓
Runs in Rocket AI Hub programmer portal	❌	❌	✓
Runs in Rocket Cognitive Environment	✓*	❌	✓
Runs in Alpine Linux containers	✓	❌	✓
Runs in Raspberry Pi	✓	❌	✓
Runs in Arduino	❌	❌	✓

Un seul port ? La belle affaire !

- De la connexion distante à la première opération sur la base de données

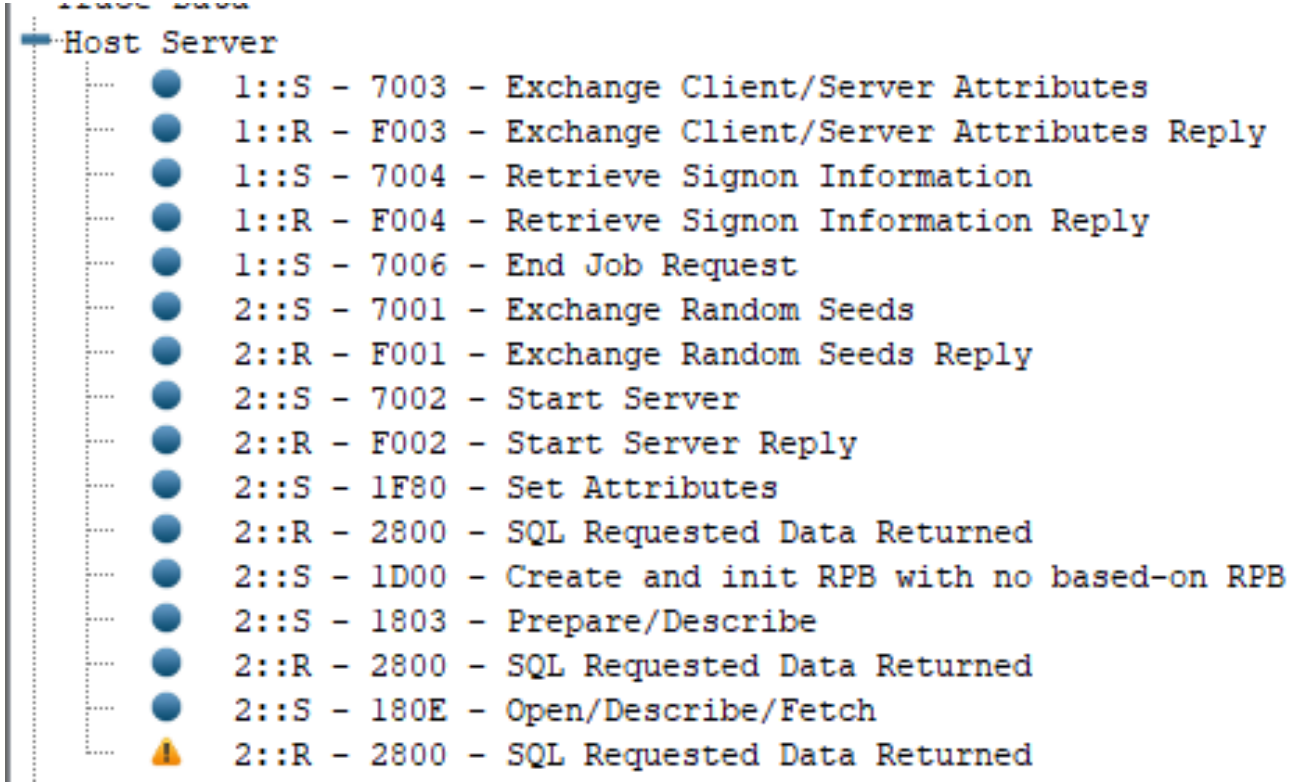
- JDBC/ODBC

- Mapepire

Exemple de programme JDBC

```
try (AS400 hi = new AS400("myhostname", "uid".toCharArray(), "password".toCharArray())) {
    AS400JDBCDataSource ds = new AS400JDBCDataSource(hi);
    Connection conn = ds.getConnection();
    Statement s = conn.createStatement();
    s.executeQuery("select * from QIWS.QCUSTCDT");
    ResultSet rs = s.getResultSet();
    while(rs.next()) {
        System.out.println(rs.getString(1));
    }
    System.out.println("done");
}
```

11 Distinct TCP Flows



Pourquoi ?

- Plus simple et plus léger que les solutions existantes
  - Pas de prérequis sur le système hôte
  - Un seul port pour une communication
- Performant
  - car circuit court : pas de driver intermédiaire côté client
  - utilisant les web sockets pour la communication
- Sécurisé par design
  - Une chose de moins à penser

➔ Une solution plus Cloud-Friendly que les autres solutions natives

IBM i
continuous innovation
continuous integration

Mapepire

Qu'est-ce que c'est ?

Pourquoi ?

Comment ?

Université IBM i

19 et 20 novembre 2024

IBM i
continuous innovation
continuous integration

IBM

Mapepire

côté serveur

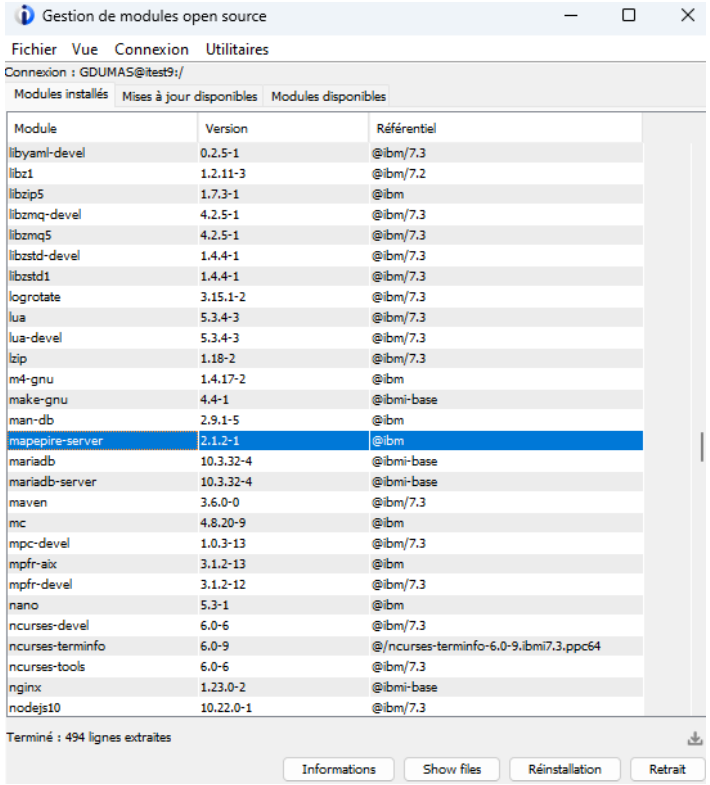
Installation Mapepire Server

- La méthode recommandée est d'utiliser **yum**
  - En ligne de commande :
    yum install mapepire-server
  - Ou par le gestionnaire de modules Open Source d'ACS

- On peut aussi le télécharger et l'installer manuellement

- Infos sur l'installation :

https://mapepire-ibmi.github.io/guides/sysadmin/



Lancement du serveur Mapepire – Service-commander

- Pour lancer le serveur **mapepire**
  - Utilisation de **service-commander** recommandée
  - Installation avec : yum install service-commander

- https://github.com/ThePrez/ServiceCommander-IBMi

- Lancer le service mapepire à l'aide de
  - sc start mapepire

```
-bash-5.2$ sc start mapepire
Performing operation 'START' on service 'mapepire'
Service 'Mapepire Server' successfully started
```

Voir présentation **S05**
Orchestration des travaux IBM i avec service-commander
Université IBM i 2023

Configurations mapepire-server

- Port par défaut : **8076**
  - Recommandation : ne pas le changer
  - Si besoin, manipulation de la variable d'environnement PORT ou en changeant la configuration du service de service-commander : scedit mapepire puis directive check_alive
- TLS
  - Option 1 : Utiliser Let's Encrypt
    - Si détection d'utilisation sur la LPAR (**/etc/letsencrypt/live/<hostname>**), utilisation automatique
  - Option 2 : Utiliser son propre certificat
    - Avec le magasin de certificat suivant **/QOpenSys/etc/mapepire/cert/server.jks** à configurer
  - Option 3 : Utiliser un certificat auto-signé
    - Si les options 1 et 2 non détectées, Mapepire génère et utilise son propre certificat auto-signé

Comment encrypter les données avec _DBC

1. Se connecter dans DCM
2. Créer un magasin d'autorité de certification locale (CA)
3. Créer un certificat local CA
4. Enregistrer la valeur auto-générée du label CA
5. Créer le magasin de certificats *SYSTEM (si besoin)
6. Créer un nouveau certificat serveur
7. Signer le certificat serveur avec la local CA.
8. Assigner le certificat serveur à une application (host server application)
9. Redémarrer le Host Server

10. Côté client, télécharger l'autorité de certification dans le magasin de confiance (ou configurer d'ignorer TLS => pas la bonne pratique)

Comment encrypter les données avec Mapepire

Étape 1 — Utiliser Mapepire

Étape 2 — Aller se reposer !

Qu'apporte TLS ?

- TLS permet :
  - De crypter les informations qui vont transiter sur le réseau
  - D'authentifier le serveur sur lequel on se connecte
    - En vérifiant la validité du certificat serveur
    - En vérifiant que le serveur est bien celui qu'il prétend être
      - En checkant que le certificat serveur est signé par une autorité de confiance du client
      - En checkant que le hostname correspond à celui du certificat

Configuration par défaut du service mapepire

- sc info mapepire

```
-------------------------------------------------------------------
mapepire (Mapepire Server)


Defined in: /QOpenSys/pkgs/lib/mapepire/mapepire.yaml

Working Directory: .

Startup Command: ../../bin/mapepire
Startup Wait Time (s): 60

Shutdown Wait Time (s): 45

Check-alive conditions: JOBNAME:MAPEPIRE, PORT:8076
Batch Mode: <submitted to batch>
   Batch Job Name: MAPEPIRE
   SBMJOB options: JOBQ(QUSRNOMAX)

Inherits environment variables?: true
Custom environment variables:
   PATH=/QOpenSys/pkgs/bin:/QOpenSys/usr/bin:/usr/ccs/bin:/QOpenSys/usr/bin/X11:/usr/sbin:.:/usr/bin
```

Considérations sur la version java

- Utilisation de la version 8 par défaut (plus petit dénominateur commun)
  - En cours de discussion pour passage au java 11 https://github.com/Mapepire-IBMi/mapepire-server/pull/80

- Possible d'avoir des problèmes du type SSL handshake failure
  - Erreur obtenue côté client lors d'une tentative de connexion avec Python ou Node dans leurs versions les plus récentes

```
RuntimeError: An error occurred while connecting to the server: [SSL: SSLV3_ALERT_HANDSHAKE_FAILURE] sslv3 alert handshake failure (_ssl.c:1000)
```

```
Error: 0C470000:error:0A000410:SSL routines:ssl3_read_bytes:sslv3 alert handshake failure:c:\ws\deps\openssl\openssl\ssl\record\rec_layer_s3.c:1586:SSL alert number 40
. Please ensure the Mapepire server is being run with Java version >= 11.
```

Considérations sur la version java

- cat /QOpenSys/pkgs/bin/mapepire

```
#!/QOpenSys/usr/bin/sh
exec /QOpenSys/QIBM/ProdData/JavaVM/jdk80/64bit/bin/java -Xsoftmx1G -Xmx16g -Xms256M -jar $(/QOpenSys/usr/bin/dirname $0)/../lib/mapepire/mapepire-server.jar
```

Java 8 par défaut

```
#!/QOpenSys/usr/bin/sh
exec /QOpenSys/pkgs/lib/jvm/openjdk-11/bin/java -Xsoftmx1G -Xmx16g -Xms256M -jar $(/QOpenSys/usr/bin/dirname $0)/../lib/mapepire/mapepire-server.jar
```

Modification du fichier pour utiliser Java 11 OSS
Installation java 11 OSS : yum install openjdk-11-ea

En résumé – côté serveur

- Installation du package rpm mapepire-server
- Lancement du serveur à l'aide de service-commander

- Et…

- … C'est tout !

- Le serveur est prêt à recevoir des demandes de connexions sécurisées

Université IBM i

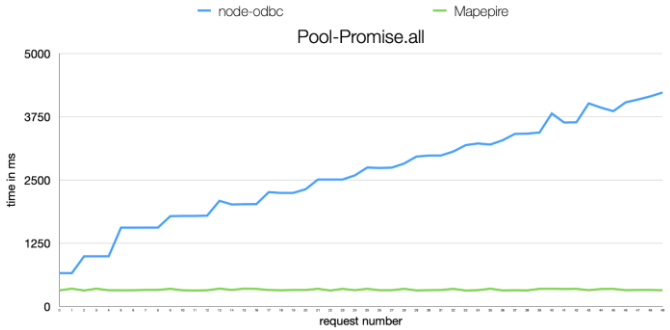
19 et 20 novembre 2024

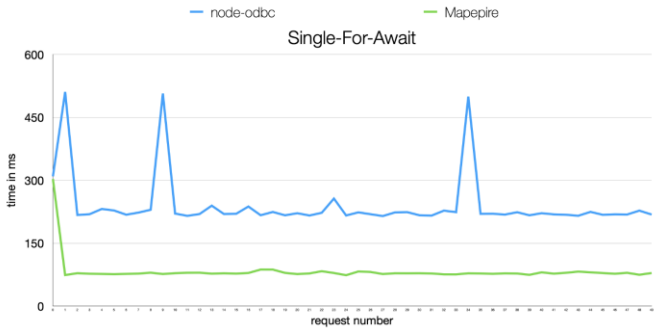
Mapepire

Performances, contrôles et optimisations

Quelques comparaisons de performances

- Benchmark de performances entre mapepire et odbc (avec node) réalisé en août 2024
- Les tests sont disponibles pour pouvoir les rejouer dans vos environnements
  - https://github.com/worksofliam/node-db-perf
- Il fait ressortir que :
  - mapepire gagne en termes de mutualisation et de performance avec un seul travail (une requête à la fois)
  - ODBC fonctionne bien mais on observe un phénomène de mise en file





- Contexte et résultats complets du test : https://github.com/worksofliam/blog/issues/69

Le backend Mapepire utilise JDBC

- Mapepire est une interface devant JTOpen et JDBC
- Mapepire utilise les travaux QZDASOINIT
  - Ou son équivalent sécurisé QZDASSINIT (S = "secure")

- Toutes les considérations concernant les travaux ODBC/JDBC s'appliquent toujours :
  - Sécurité
  - Scalabilité

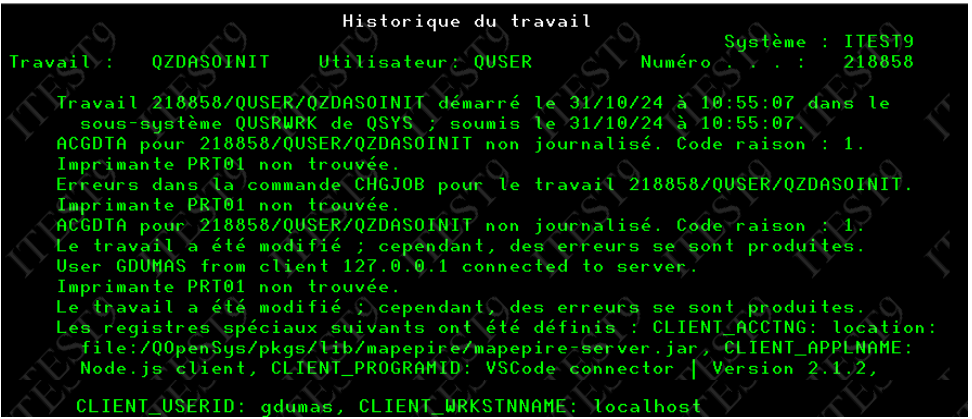
Sécurité JDBC

- Mapepire fonctionne "localement" sur l'IBM i
  - Ce qui aide à contrôler la prolifération des connexions ODBC/JDBC
  - Centralisation des connexions
- La sécurité sur les objets continue de s'appliquer
- Les points d'exit ODBC/JDBC continue de fonctionner pour contrôler le trafic et les accès

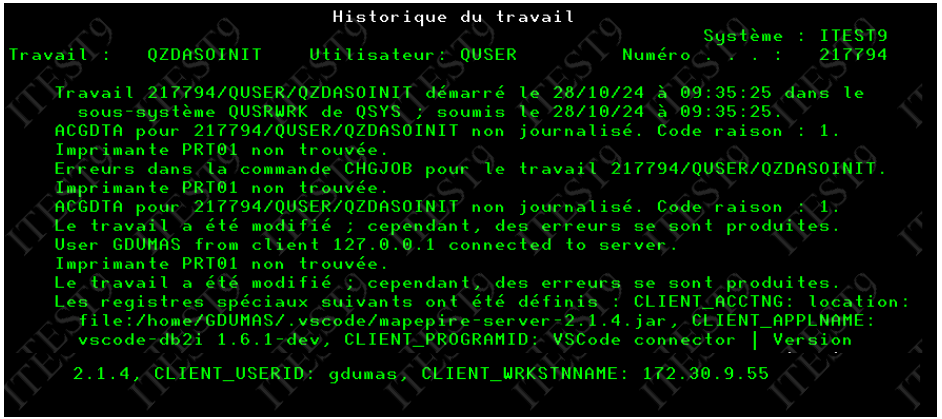
Sécurité JDBC – points d'exit

- A aujourd'hui, les informations disponibles dépendent de la configuration du client

Client mapepire js



Client mapepire VSCode Python Jupiter



- A noter que des évolutions sont en cours notamment pour faire remonter de façon systématiques l'IP du client qui se connecte dans les registres spéciaux
  - https://github.com/Mapepire-IBMi/mapepire-server/issues/83

Comment gérer la charge JDBC ?

- Par défaut, tous les travaux QZDASOINIT/ QZDASSINIT tournent dans QUSRWRK

- Questions:
  - Comment contrôler les requêtes provenant d'outils de requêtage ?
  - Comment savoir quelle application consomme des ressources ?
  - Comment gérer de manière efficace les travaux JDBC ?
  - Comment prioriser les ressources pour les besoins critiques ?

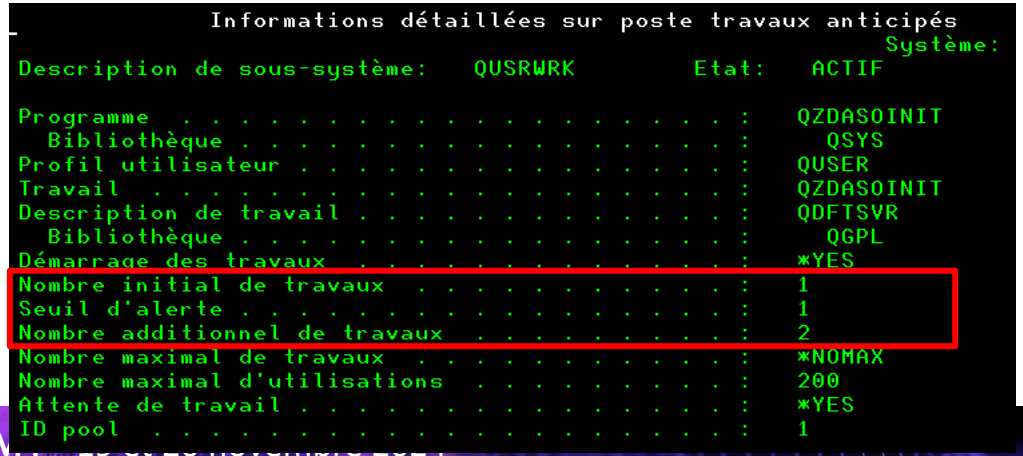
Séparer les travaux par application, utilisateur…

- Configuration des travaux QZDAS_INIT pour tourner dans des sous-systèmes séparés, basés sur vos critères
  - JDBCSHOP, JDBCADHOC, JDBCNODE …
- De cette façon, les mesures de performances peuvent être agrégées par sous-systèmes et nous pouvons configurer la mémoire par sous-systèmes
- Il est aussi possible de limiter les ressources consommées par ces travaux avec ces techniques (Query Supervisor, Workload capping, Pool …)
  - https://www.ibm.com/support/pages/setting-limitations-resources-used-qzdasoinit-prestart-jobs
- Bénéfice supplémentaire : les travaux JDBC de différentes applications n'interagissent pas. Plus facile à identifier / déboguer.

https://www.seidengroup.com/2022/05/04/simplify-with-subsystems/

Configuration des travaux à démarrage anticipé JDBC

- Configurer le bon nombre de travaux à démarrage anticipé
  - Les travaux à démarrage anticipés ODBC/JDBC sont les travaux QZDASO(S)INIT dans le sbs QUSRWRK
- Vérifier votre configuration :
  - DSPSBSD SBSD(qusrwrk)
  - Option 10, Postes travaux à démarrage anticipé
  - Option 5 devant QZDASOINIT



Configuration par défaut JDBC

- Valeurs par défaut (pas forcément optimisé)
  - inljobs= 1, threshold = 1, adljobs = 2
  - A changer en fonction du besoin

```
CHGPJE SBSD(QSYS/QUSRWRK) PGM(QSYS/QZDASOINIT)
       STRJOBS(*YES) INLJOBS(xx) THRESHOLD(xx)
       ADLJOBS(xx)
```

  - Comment déterminer les meilleures valeurs ?
    - DSPACTPJ (diapo suivante)

```
                Informations détaillées sur poste travaux anticipés
                                                                Système:
Description de sous-système:   QUSRWRK          Etat:   ACTIF

Programme . . . . . :   QZDASOINIT
  Bibliothèque . . . . . :     QSYS
Profil utilisateur . . . . . :   QUSER
Travail . . . . . :   QZDASOINIT
Description de travail . . . . . :   QDFTSVR
  Bibliothèque . . . . . :     QGPL
Démarrage des travaux . . . . . :   *YES
Nombre initial de travaux . . . . . :   1
Seuil d'alerte . . . . . :   1
Nombre additionnel de travaux . . . . . :   2
Nombre maximal de travaux . . . . . :   *NOMAX
Nombre maximal d'utilisations . . . . . :   200
Attente de travail . . . . . :   *YES
ID pool . . . . . :   1
```

Combien de jobs sont nécessaires ?

- DSPACTPJ SBS(QUSRWRK) PGM(QZDASOINIT)
- https://www.ibm.com/docs/en/i/7.4?topic=jobs-tuning-prestart-job-entries

```
Prestart jobs:
   Current number . . . . . :   29
   Average number . . . . . :   21.9
   Peak number  . . . . . :   49

 Prestart jobs in use:
   Current number . . . . . :   27
   Average number . . . . . :   18.4
   Peak number  . . . . . :   46
```

```
Program start requests:
   Current number waiting . . . . . :   0
   Average number waiting . . . . . :   .0
   Peak number waiting  . . . . . :   6
   Average wait time  . . . . . :   00:00:00.0
   Number accepted  . . . . . :   100299
   Number rejected  . . . . . :   0
```

IBM i
continuous innovation
continuous integration

Mapepire

côté client

Cohérence du SDK pour tous les langages

- Guidé par une architecture de référence unifiée

- Expériences similaires
  - Noms des classes
  - Nom des méthodes
  - Type Throwable (pour les exceptions)
  - Les paramètres
  - Les options de configuration

Les différents clients

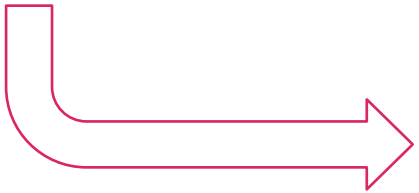
- Les clients disponibles :
  - Python
    - https://github.com/Mapepire-IBMi/mapepire-python
  - Node.js
    - https://github.com/Mapepire-IBMi/mapepire-js
  - Java
    - https://github.com/Mapepire-IBMi/mapepire-java
- Les clients à venir :
  - PHP
    - https://github.com/Mapepire-IBMi/mapepire-php (Work in progress)
  - JDBC
    - https://github.com/Mapepire-IBMi/mapepire-jdbc (Work in progress)
    - Connection connection = DriverManager.getConnection("**jdbc:mapepire://ossbuild.rzke.de:8076**;USER=myuser;PASSWORD=mypassword;naming=system;errors=full");
  - C# / .NET Code
  - Go

Client JavaScript

- Installation du client avec npm

npm install @ibm/mapepire-js

- Copie de .env.sample en .env
- Remplissage du .env avec les informations du serveur



.env

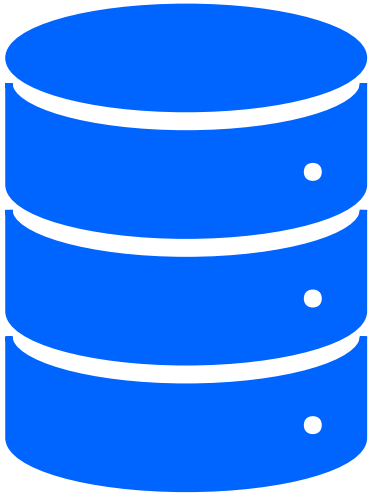
```
# defaults to localhost
VITE_SERVER=mymachine.somelab.com

# defaults to 8076
VITE_PORT=8076

VITE_DB_USER=jonsmith
VITE_DB_PASS=letmein
```

Les premières fonctionnalités à disposition

- Les fonctionnalités à disposition
  - Connexion à la base de données
  - Exécuter des requêtes SQL
  - Exécuter des requêtes SQL par lots
  - Exécuter des commandes CL
  - Obtenir un travail sur le serveur
  - Obtenir la version Mapepire
  - Check le serveur (isAlive)
  - Définir la configuration
  - Obtenir des traces de connexions
  - Fermer une connexion



Connexion base de données

- Méthode connect de la classe SQLJob

```
// Connect to the database
const creds = ENV_CREDS ;
const job = new mapepire.SQLJob();
await job.connect(creds);
```

Exécution d'une requête

- Méthode query de la classe SQLJob pour construire la requête
- Méthode execute pour exécuter la requête

```
await job.connect(creds);
const query = await job.query<any>("select * from sample.department");
const res = await query.execute();
await query.close()
await job.close();
```

Exécuter une requête préparée

- Méthode query de la classe SQLJob pour construire la requête
  - Les paramètres notés avec des ?
  - Deuxième paramètre de query avec un tableau de valeurs
- Méthode execute pour exécuter la requête

```
const job = new SQLJob();
await job.connect(creds);
const query = await job.query<any>(
  "SELECT * FROM SAMPLE.SYSCOLUMNS WHERE COLUMN_NAME = ?",
  {
    parameters: ["Value"],
  }
);
const res = await query.execute();
await query.close();
await job.close();
```

Exécuter une requête SQL par lot

- Méthode query de la classe SQLJob pour construire la requête
  - Les paramètres notés avec des ?
  - Deuxième paramètre de query avec un tableau de tableaux de valeurs

    => Pour envoyer un lot
- Méthode execute pour exécuter la requête

```
let query = job.query("INSERT INTO SAMPLE.EMPLOYEES values (?, ?)", {
  parameters: [
    ["SANJULA", "416 345 0879"],
    ["TONGKUN", "647 345 0879"],
    ["KATHERINE", "905 345 1879"],
    ["IRFAN", "647 345 0879"]
  ],
});
let res = await query.execute();
```

Exécuter une requête avec un pool

- Initialiser un objet Pool
- Appeler la méthode execute de l'objet Pool pour utiliser le premier job disponible pour exécuter la requête
- Appeler la méthode end de l'objet Pool pour le détruire et nettoyer les ressources

```
let pool = new Pool({ creds, maxSize: 5, startingSize: 5 });
await pool.init();
// Initiate a bunch of jobs
const executedPromises = [
  pool.execute("select * FROM SAMPLE.SYSCOLUMNS"),
  pool.execute("select * FROM SAMPLE.SYSCOLUMNS"),
  pool.execute("select * FROM SAMPLE.SYSCOLUMNS"),
];
const res = await Promise.all(executedPromises);
await pool.end();
```

Les exemples

- Référentiel d'exemples qui utilisent le client SDK Mapepire :
  - https://github.com/Mapepire-IBMi/samples
    - Java
    - Python ➔ Objet de la démonstration
    - Typescript

Exemple complet en js

- npm i @ibm/mapepire-js
- npm i dotenv
- Création du .env à la racine avec les infos de connexion
- Création de index.js
- Test avec node index

```
000010 - CHRISTINE - HAAS
000020 - MICHAEL - THOMPSON
000030 - SALLY - KWAN
000050 - JOHN - GEYER
000060 - IRVING - STERN
000070 - EVA - PULASKI
000090 - EILEEN - HENDERSON
000100 - THEODORE - SPENSER
000110 - VINCENZO - LUCCHESSI
000120 - SEAN - O'CONNELL
000130 - DELORES - QUINTANA
000140 - HEATHER - NICHOLLS
000150 - BRUCE - ADAMSON
000160 - ELIZABETH - PIANKA
000170 - MASATOSHI - YOSHIMURA
000180 - MARILYN - SCOUTTEN
```

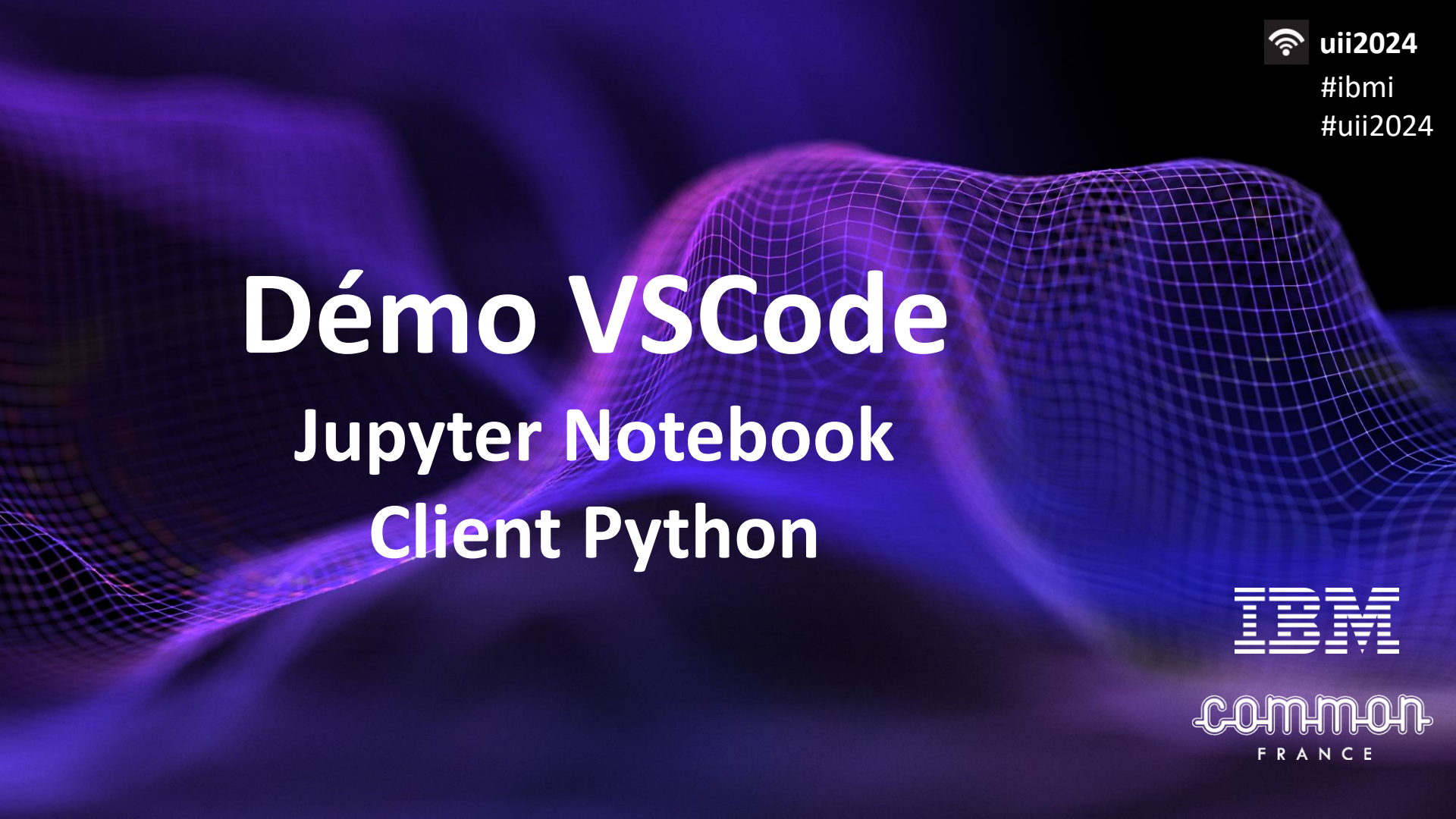
```
000190 - JAMES - WALKER
000200 - DAVID - BROWN
000210 - WILLIAM - JONES
000220 - JENNIFER - LUTZ
000230 - JAMES - JEFFERSON
000240 - SALVATORE - MARINO
000250 - DANIEL - SMITH
000260 - SYBIL - JOHNSON
000270 - MARIA - PEREZ
000280 - ETHEL - SCHNEIDER
000290 - JOHN - PARKER
000300 - PHILIP - SMITH
000310 - MAUDE - SETRIGHT
000320 - RAMLAL - MEHTA
000330 - WING - LEE
000340 - JASON - GOUNOT
```

```
require('dotenv').config();
const mapepire = require('@ibm/mapepire-js');
const creds = {
  host: process.env.VITE_SERVER,
  port: process.env.VITE_PORT,
  user: process.env.VITE_DB_USER,
  password: process.env.VITE_DB_PASS,
  ignoreUnauthorized: true
};
async function execRequest() {
    const job = new mapepire.SQLJob();
    await job.connect(creds);

    const query = job.query(`select EMPNO, FIRSTNME, LASTNAME from SAMPLE.EMPLOYEE`);
    const result = await query.execute();
    result.data.forEach(row => console.log(`${row.EMPNO} - ${row.FIRSTNME} -
${row.LASTNAME}`));

    await job.close();
  }
  execRequest();
```

Index.js


uii2024
#ibmi
#uii2024
Démo VSCode
Jupyter Notebook
Client Python
IBM
common
FRANCE

IBM i
continuous innovation
continuous integration

Mapepire

La suite ?

Prochaines évolutions

- Conformité aux normes
  - Python's PEP 249
  - Java JDBC client

- Technology Preview 2 ?
  - Compression des données
  - Amélioration de la gestion TLS (lien avec DCM)
  - Gestion des logs
  - Amélioration de la maintenance

En conclusion

- **Simplicité**
  - Côté serveur : package à installer et à démarrer
  - Côté client : natif dans les différents langages
- **Portabilité**
  - Tous les hardwares
  - Tous les langages
- **Rapidité**
  - Du fait de sa simplicité
- **Sécurité**
  - Toujours encrypté
  - Points d'exit pour contrôler le trafic


MERCI