レガシー刷新とデータ活用
~OSSで始めるCOBOL基幹のモダン化~

2025年10月17日（金）

OSSコンソーシアム
オープンCOBOLソリューション部会
井坂　徳恭

自己紹介

井坂　徳恭
いさか　　のりゆき

#nor

- OSSコンソーシアム オープンCOBOLソリューション部会
  - opensource COBOL等のコミッターを担当
  - https://github.com/opensourcecobol

- 業務経験
  - COBOLを始め、様々な言語、システムのマイグレーションを担当
  - 社内製品およびOSSの開発、管理

アジェンダ

1. 基幹システムとCOBOLの課題
2. データ活用の展望
3. opensouce COBOLの紹介
4. まとめ

レガシー移行とデータ活用

1. 基幹システムとCOBOL課題

COBOL について考える

- COBOL 言語の近年の話題
  - 基本情報処理技術者試験から COBOL の出題が終了
  - COBOL は古い、新しいことできない･･･
  - COBOL のシステムは複雑、やりたくない･･･
  - COBOL 技術者の引退、技術者不足
  - 空前の脱 COBOL ブーム（Javaリライト）
  - 今も COBOL 資産は世界に 8000 億行ある
    （Micro Focus 社 2022年2月調査）

黙々と処理をこなして社会基盤を支える巨人COBOL
その姿はギリシャ神話で天空を背負うアトラス
COBOLが手を離せば天空が落ちてきて社会が大混乱する

問題の本質は、COBOL 言語ではなく「IT資産の健全性」の問題

IT資産の健全性について

- 本来 COBOL は保守性や変更容易性が高い言語
  - 国際規格で標準化、2023年1月に新規格公開（ISO/IEC 1989:2023）
  - 英語を用いた分かり易い構文、事務処理計算の業務を抽象的に記述
  - 構造化プログラミング、コード再利用の仕組み

NEW

- 手入れ不足で不健全になり、維持・運用にコストと人手がかかっている
  - **長年の増改築**（コピペ量産、念のため残した処理、その場しのぎのバイパス処理）
  - **標準でない記述**（ベンダー独自機能、特定機器向け制御コード）
  - **仕様書**や**テストケースの消失** (元々は綺麗に整っていたはず)
  - COBOL 技術者不足よりも深刻な**仕様理解者不足** (ベテラン引退、引継ぎ不足)

マイグレーションは IT 資産の健全性の回復チャンス

COBOL ユーザーが抱える課題

脱レガシーの課題

- メインフレーム機能の代替方式の検討
- 高額な移行費用と、残された制限時間
- 移行に伴うリスク

レガシー・マイグレーション

COBOL資産を活用

COBOL の今後

- COBOL は安定稼働もメーカーがサポート終了
- Java 移行するにしても期間やコストが合わない
- スキルチェンジへの不安

二刀流移行

（COBOL リホストと Java リライト）

OSSなCOBOL基盤

DX への対応

- 柔軟性や拡張性が低く変化に迅速に対応できない
- 基幹系データを活用したい
- 攻めの ITと連携したい

クラウドネイティブ

COBOL システムの DX を加速
OSSとクラウドの親和性

エンジニアの課題

- ベテランエンジニア引退
- COBOL 人材の不足
- 仕様書が消失、分かるエンジニアも減ってブラックボックス化が進んでいる

AIソリューションの活用

AI が COBOL 仕様書作成 や
COBOL システムの質疑応答 に対応

レガシー移行とデータ活用

2. データ活用の展望

COBOL ユーザーが抱える課題

データ活用に向けて

脱レガシーの課題

- メインフレーム機能の代替方式の検討
- 高額な移行費用と、残された制限時間
- 移行に伴うリスク

レガシー・マイグレーション

COBOL資産を活用

COBOL の今後

- COBOL は安定稼働もメーカーがサポート終了
- Java 移行するにしても期間やコストが合わない
- スキルチェンジへの不安

二刀流移行

（COBOL リホストと Java リライト）

OSSなCOBOL基盤

DX への対応

- 柔軟性や拡張性が低く変化に迅速に対応できない
- 基幹系データを活用したい
- 攻めの ITと連携したい

クラウドネイティブ

COBOL システムの DX を加速
OSSとクラウドの親和性

エンジニアの課題

- ベテランエンジニア引退
- COBOL 人材の不足
- 仕様書が消失、分かるエンジニアも減ってブラックボックス化が進んでいる

AIソリューションの活用

AI が COBOL 仕様書作成 や
COBOL システムの質疑応答 に対応

DX への対応

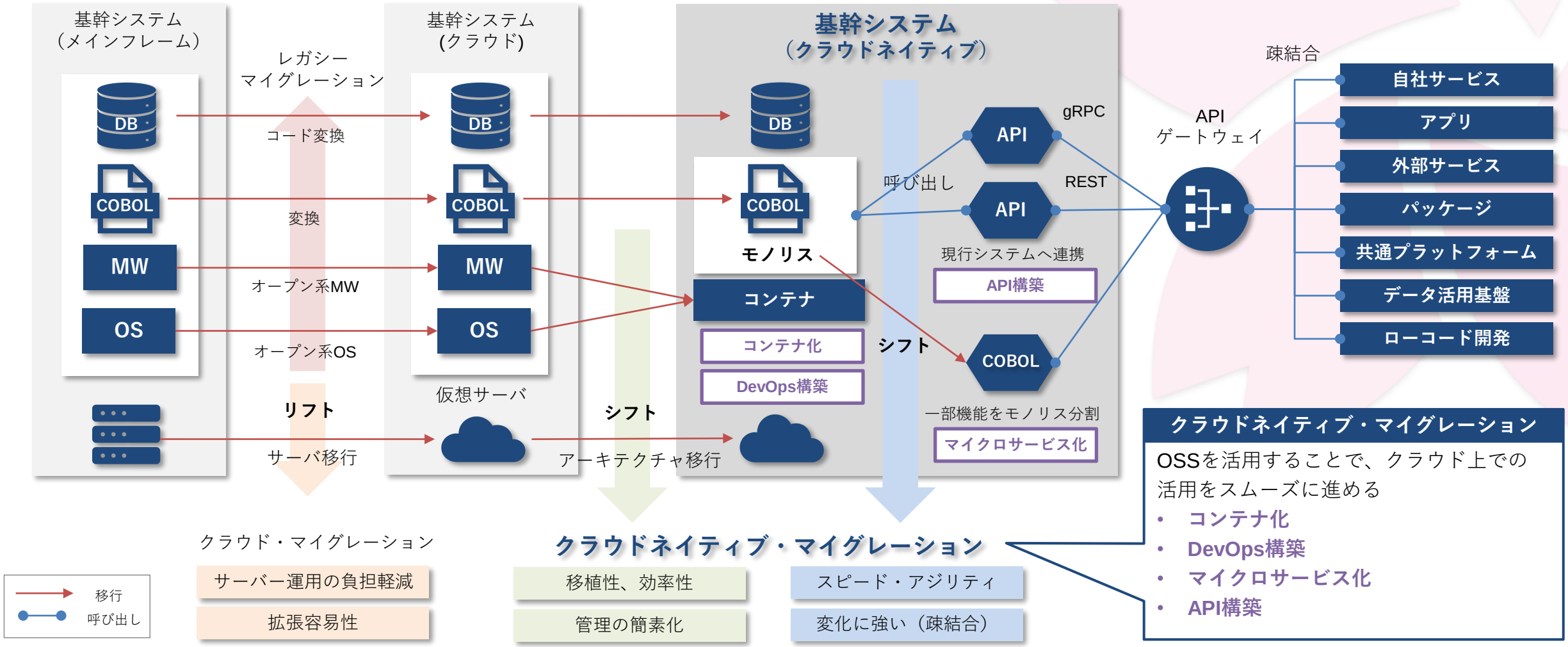
- 柔軟性や拡張性が低く変化に迅速に対応できない
- 基幹系データを活用したい
- 攻めの IT と連携したい

基幹システム内のデータ活用が必要

- ファイルデータではデータ分析が難しい
- 電文など特殊な形式で連携が困難

デジタル変革に求められるさまざまなピースと
COBOL が上手くつながっていない

DXとデータ連携



クラウドネイティブ・マイグレーション

OSSを活用することで、クラウド上での活用をスムーズに進める

- コンテナ化
- DevOps構築
- マイクロサービス化
- API構築

レガシー移行とデータ活用

3. opensource COBOLの紹介

COBOL ユーザーが抱える課題

OSSなCOBOL

脱レガシーの課題

- メインフレーム機能の代替方式の検討
- 高額な移行費用と、残された制限時間
- 移行に伴うリスク

レガシー・マイグレーション

COBOL資産を活用

COBOL の今後

- COBOL は安定稼働もメーカーがサポート終了
- Java 移行するにしても期間やコストが合わない
- スキルチェンジへの不安

二刀流移行
（COBOL リホストと Java リライト）

OSSなCOBOL基盤

DX への対応

- 柔軟性や拡張性が低く変化に迅速に対応できない
- 基幹系データを活用したい
- 攻めの ITと連携したい

クラウドネイティブ

COBOL システムの DX を加速
OSSとクラウドの親和性

エンジニアの課題

- ベテランエンジニア引退
- COBOL 人材の不足
- 仕様書が消失、分かるエンジニアも減ってブラックボックス化が進んでいる

AIソリューションの活用

AI が COBOL 仕様書作成 や
COBOL システムの質疑応答 に対応

opensource COBOLシリーズ

当部会では opensource COBOLシリーズの5つのプロダクトの開発とサポートを提供します

opensource COBOL 1.5.2J
・COBOLコンパイラ（C版）
・OpenCOBOLに日本語拡張や国産ホスト互換を強化
・国内で多くの導入実績あり

opensource COBOL 4J 1.0.22
・COBOLコンパイラ（Java版）
・opensource COBOL 1.5.2JからFork
・Javaトランスレート版、効率的なJava移行
・ほぼ毎月バージョンアップ中

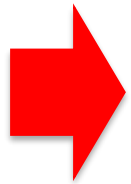
Open COBOL ESQL 1.3
・埋め込みSQLプリコンパイラ（C版）
・PostgreSQLに対応（libpq）
・商用COBOLも利用可能、海外で多数の導入実績

Open COBOL ESQL 4J 1.0.3
・埋め込みSQLプリコンパイラ（Java版）
・PostgreSQLに対応（jdbc）
・Javaの実行ライブラリを提供

GnuCOBOL 3.2 osscons patch 2
・COBOLコンパイラ（C版）
・最新バージョンに対する日本向け拡張のパッチ（forkではない）
・Pull Requestで本家へマージしつつ、最新版に追随したパッチを提供

開発実績

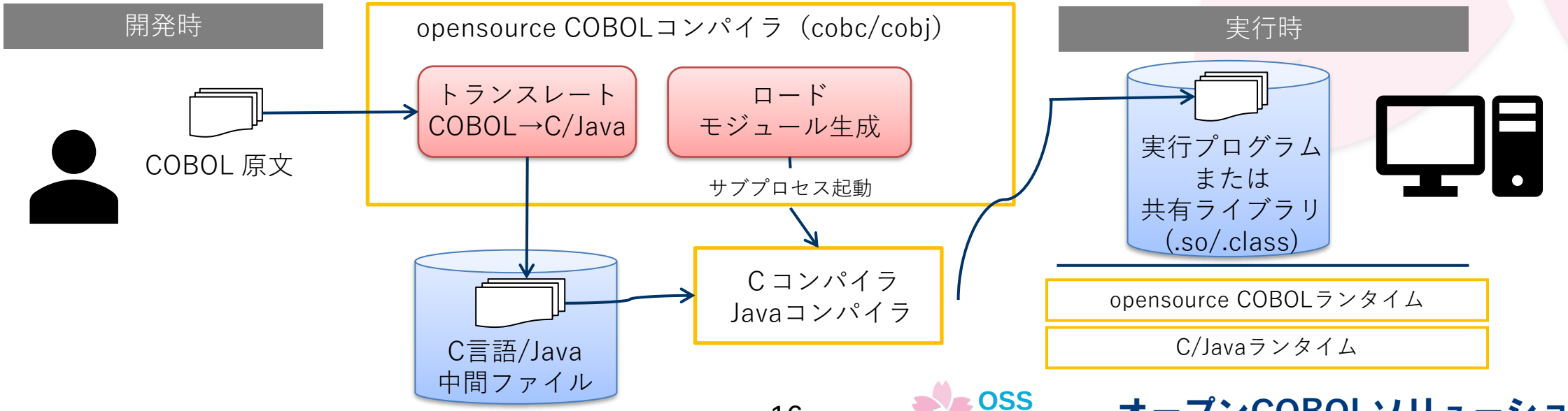
- opensource COBOLへの日本向け拡張
  - 日本語処理のサポート(SJIS、UTF-8対応)
  - ユーザファイルハンドラのサポート
  - Windows cl コンパイラのサポート
  - SCREEN SECTIONを拡張（罫線、日本語表示、等）
  - ホストや他のオープン系COBOLの構文、比較処理、関数、緩い仕様をサポート
- データベース連携ツール（Open COBOL ESQL）の開発
- opensource COBOL 4Jの開発
  - 可読性のある中間（Java）ソース
  - COBOL API生成機能
- GnuCOBOL向けの拡張パッチの作成 NEW



日本のビジネス適用に
必要な機能を拡張

コンパイルと実行（１）

- COBOLをC言語/Javaにトランスレート
- 『gcc(linux)』『cl(windows)』『java』といった一般のコンパイラで生成
- それぞれの言語で作られたモジュールとの連携も容易

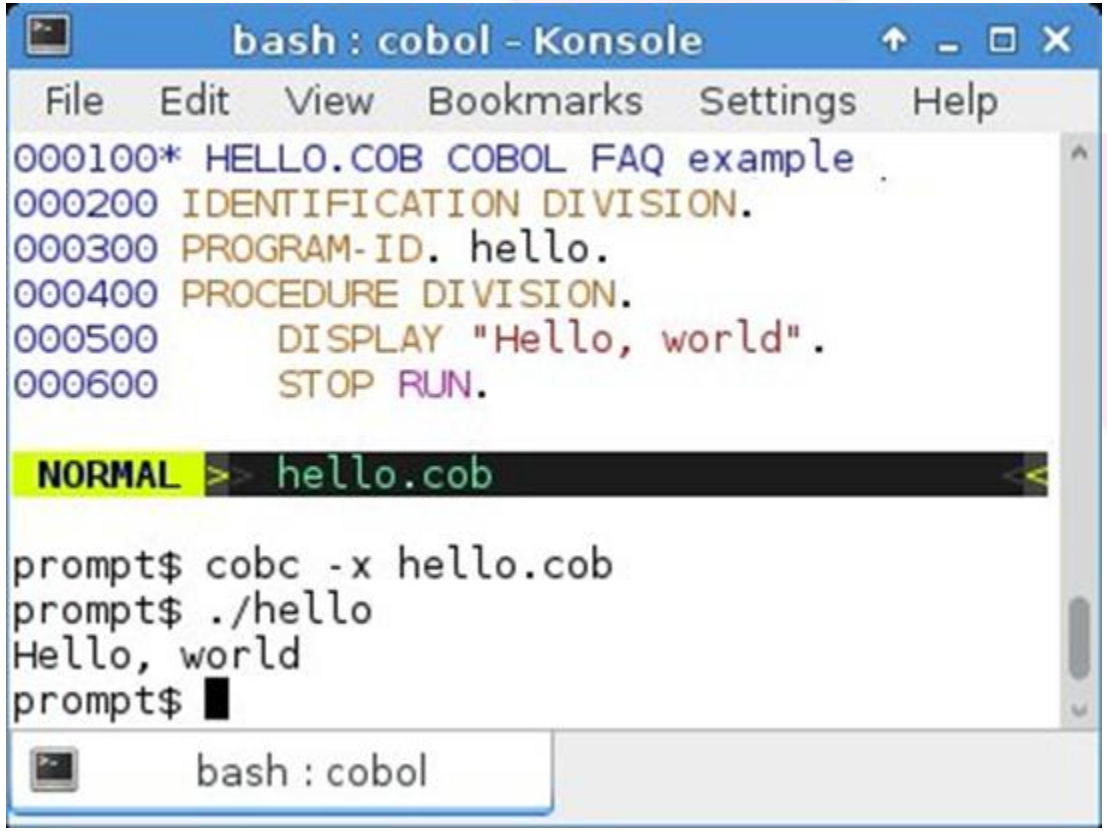


コンパイルと実行（2）

- COBOL言語のロジックをそのまま実行可能

```
$ cobc -x hello.cob
$ ./hello
Hello, world

$ cobc hello.cob
$ cobcrun hello
Hello, world
```

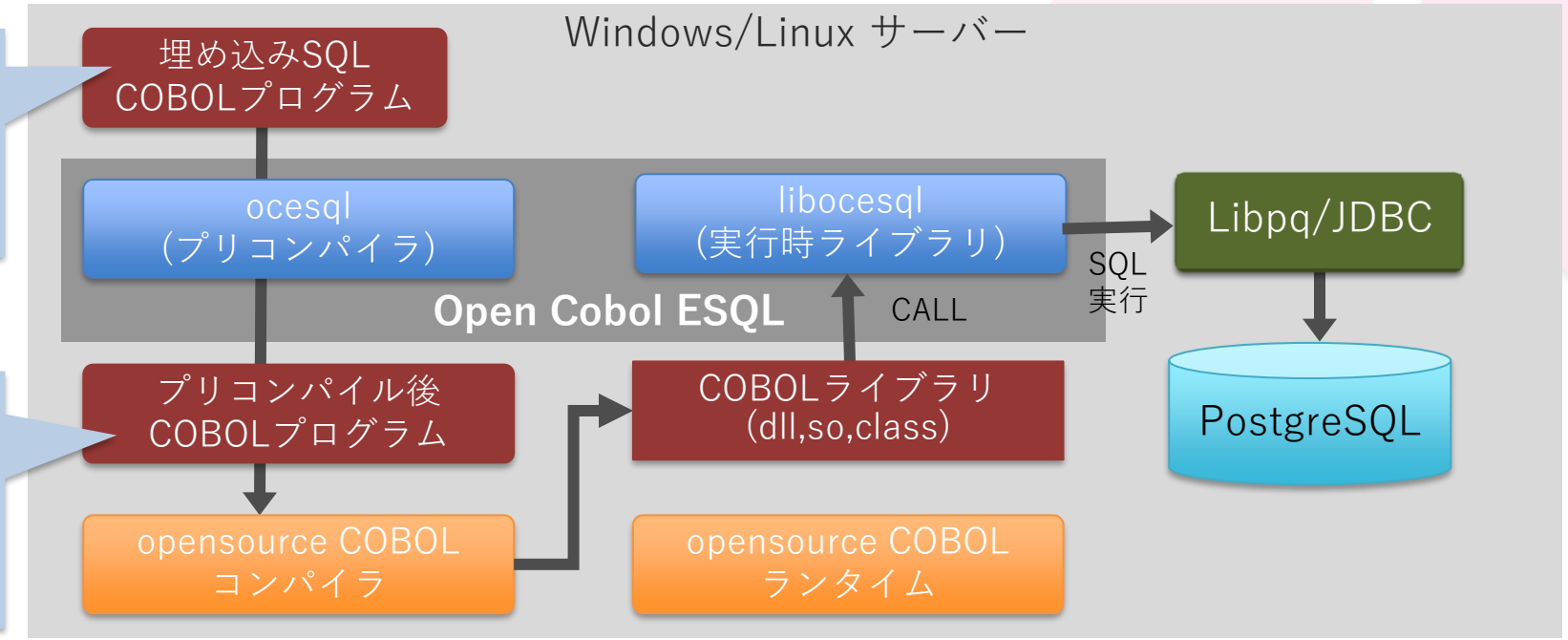


Open COBOL ESQL（4J）

- COBOLからPostgreSQLにアクセスするツール
  - プリコンパイラ：EXEC SQLをCALLに変換する
  - 実行時ライブラリ：COBOL用APIを提供
  - 地味に世界中で使われている

```
EXEC SQL
    FETCH C1 INTO
      :EMP-NO,
      :EMP-NAME,
      :EMP-SALARY
END-EXEC.
```

```
CALL "OCESQLStartSQL".
CALL "OCESQLSetResultParams"
USING
    BY VALUE 3
    BY VALUE 4
    BY VALUE 0
    BY REFERENCE EMP-NO.
```



OSS Consortium オープンCOBOLソリューション部会

COBOL API の公開

cobj-api コマンドが COBOL の LINKAGE から REST API を自動生成

▶ サブルーチン “LOANSUB”を CALL

LOANCOBOL.cbl (呼ぶ側)

```
MAIN-100.
    CALL "LOANSUB" USING WK-LOAN,
                         WK-INTEREST-RATE,
                         WK-PERIODS,
                         WK-PAYMENT,
                         WK-DATETIME.
MAIN-200.
    DISPLAY "返済額(月) = " NO ADVANCING.
    MOVE WK-PAYMENT TO DSP-PAYMENT.
    DISPLAY DSP-PAYMENT.
```

CALL

GOBACK

LOANSUB.cbl (LINKAGEあり)

```
*********************************
 LINKAGE                SECTION.
*********************************
 01  LK-LOAN             PIC 9(09).       *>借入金額
 01  LK-INTEREST-RATE    PIC 9(02)V9(03). *>年利 (%)
 01  LK-PERIODS          PIC 999.         *>ローン期間 (月)
 01  LK-PAYMENT          PIC 9(09).       *>返済額 (月)
 01  LK-DATETIME.                         *>現在日時
   03  LK-DATE           PIC 9(08).
   03  LK-TIME           PIC 9(08).
*************************************************************
```

Java生成

cobj

cobj-api

@RestController
@RequestMapping
自動生成

LOANSUB.java

```
public class LOANSUB implements CobolRunnable {
  private boolean initialized = false;
  private CobolModule module;
  private int entry;
  private CobolRunnable cob_unifunc;

  @Override
  public int run(CobolDataStorage... argStorages) {
    return LOANSUB_(entry:0, argStorages);
  }
```

業務ロジックを
実行する

LOANSUBController.java

```
@RestController
@RequestMapping("/LOANSUB")
public class LOANSUBController {

    @GetMapping
    public LOANSUBRecord LOANSUBController(
        @RequestParam(value = "LK_LOAN", defaultValue = "0") int LK_LOAN,
        @RequestParam(value = "LK_INTEREST_RATE", defaultValue = "0") double LK_INTEREST_RATE,
        @RequestParam(value = "LK_PERIODS", defaultValue = "0") int LK_PERIODS,
        @RequestParam(value = "LK_PAYMENT", defaultValue = "0") int LK_PAYMENT,
        @RequestParam(value = "LK_DATETIME", defaultValue = "") String LK_DATETIME
    ) {
        return LOANSUBExecute(LK_LOAN, LK_INTEREST_RATE, LK_PERIODS, LK_PAYMENT, LK_DATETIME);
    }

    @PostMapping
    public LOANSUBRecord LOANSUBControllerExecute(@RequestBody LOANSUBRecord LOANSUBRecord) {
```

COBOL REST API
Cloud Run Service

▶ REST API “LOANSUB” をリクエスト

API Clients

WEB

モダンアプリ

https://demo-cobol-api-999999999999.us-central1.run.app/**LOANSUB**?**LK_LOAN**=30000000&**LK_INTEREST_RATE**=1.5&**LK_PERIODS**=360

HTTPリクエスト(GET) ※POSTも可

HTTPレスポンス(JSON)

{"statuscode":200,"LK_LOAN":30000000,"LK_INTEREST_RATE":1.5,"LK_PERIODS":360,"**LK_PAYMENT**":**103536**,"**LK_DATETIME**":"**2024111700144843**"}

COBOLの選択

- opensource COBOL 4J（二刀流）
  - 基幹系の開発言語として利用されるJavaとの連携
  - 生成されるJava中間コードの活用

- opensource COBOL / GnuCOBOL（COBOL活用）
  - COBOL機能の再現性・処理速度
  - Gnuを含む広い開発コミュニティ

3. まとめ

まとめ

- 基幹システムのデータをスムーズに活用していきたい
  - マイグレーション、DXをして活用をしていくべき
- OSSなCOBOLも活用されている
  - 現代の主要技術と親和性が高い
- COBOLは悪くない

本資料に記載の会社名、製品名は、それぞれの会社の商号、商標もしくは登録商標です。