

Oracle PaaSで LINE botを開発してみよう

日本オラクル株式会社
クラウドプラットフォームソリューション統括
アライアンス事業統括
クラウドアライアンス推進本部

橋野 薫



アジェンダ

- 1 → はじめに
- 2 → サンプル・アプリケーションの概要と構成を理解しよう
- 3 → サンプル・アプリケーションの技術要素を理解しよう
- 4 → サンプル・アプリケーションを使ってみよう

アジェンダ

- 1  はじめに
- 2  サンプル・アプリケーションの概要と構成を理解しよう
- 3  サンプル・アプリケーションの技術要素を理解しよう
- 4  サンプル・アプリケーションを使ってみよう

セミナーの趣旨

- Oracle DBアプリのLINE bot化の方法を学んでいただく
- サンプルを持ち帰って、実際に試していただく
- お客様に提案／導入していただく

アプリ開発者：

サンプル・アプリを参考 or カスタマイズすることで、お客様のLINE Botアプリ開発に役立てていただく

非アプリ開発者：

LINE Botアプリ仕組みを理解いただき、サンプルアプリのデプロイ方法をもとに実際に試していただく

「Bot」とは

Robotの短縮形

自動化されたS/W

「チャットボット」

メッセージング・サービスの特徴

わかりやすい

- ・シンプルな画面と操作性
- ・基本的にはテキスト・ベース

どこからでも

- ・スマホで使える
- ・どこにいてもアクセスできる

フレンドリー

- ・メールに比べて会話的
- ・絵文字なども使える

チャットボット・アプリケーションのタイプ

入力された文章を
人工知能で判断し、
適切な返答を行う

どこからでも使える
シンプルなアプリ

Oracleの場合

- SaaS + chat機能
 - 例) Service Cloud + chatbot
- Oracle Mobile Cloud Enterprise
(Oracle Intelligent Bots)

→ コールセンターを
中心としたニーズ

企業向け

- 在庫確認
- 新製品情報
- 営業別売り上げ
状況、達成度
- 総議ステータス
確認
- etc...

コンシューマ/
不特定多数ユーザー向け

- シンプルな検索処理
- 株価／為替などの変動データの問合せ
 - 申込み状況確認や配達状況確認

シンプルな登録処理

- 場所や部屋の予約
- 習い事やセミナーなどの申し込み
- デリバリーサービスの注文

チャットボット・アプリケーションのタイプ

入力された文章を
人工知能で判断し、
適切な返答を行う

Oracleの場合

- SaaS + chat機能
 - 例) Service Cloud + chatbot
- Oracle Mobile Cloud Enterprise
(Oracle Intelligent Bots)

→ コールセンターを
中心としたニーズ

どこからでも使える
シンプルなアプリ

本日ご紹介する
サンプル・
アプリケーション
はこちらの領域

企業向け

- 在庫確認
- 新製品情報
- 営業別売り上げ
状況、達成度
- 総議ステータス
確認
- etc...

コンシューマ/
不特定多数ユーザー向け

- シンプルな検索処理
- 株価／為替などの変動データの問合せ
 - 申込み状況確認や配達状況確認

シンプルな登録処理

- 場所や部屋の予約
- 習い事やセミナーなどの申し込み
- デリバリーサービスの注文

日本企業のチャットボット例

- **ヤマト運輸**

- 配達状況確認
- 受取日時変更
- 配送予定、不在時配達通知
- 送り状発行



- **JR東日本**

- 電車の運行情報
- 駅コインロッカー空き状況



日本企業のチャットボット例

- ・リクルートジョブズ(パン田一郎)
— アルバイト情報配信

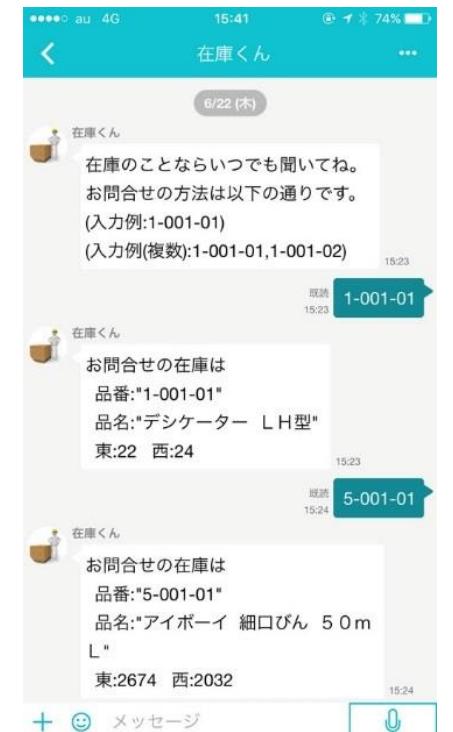
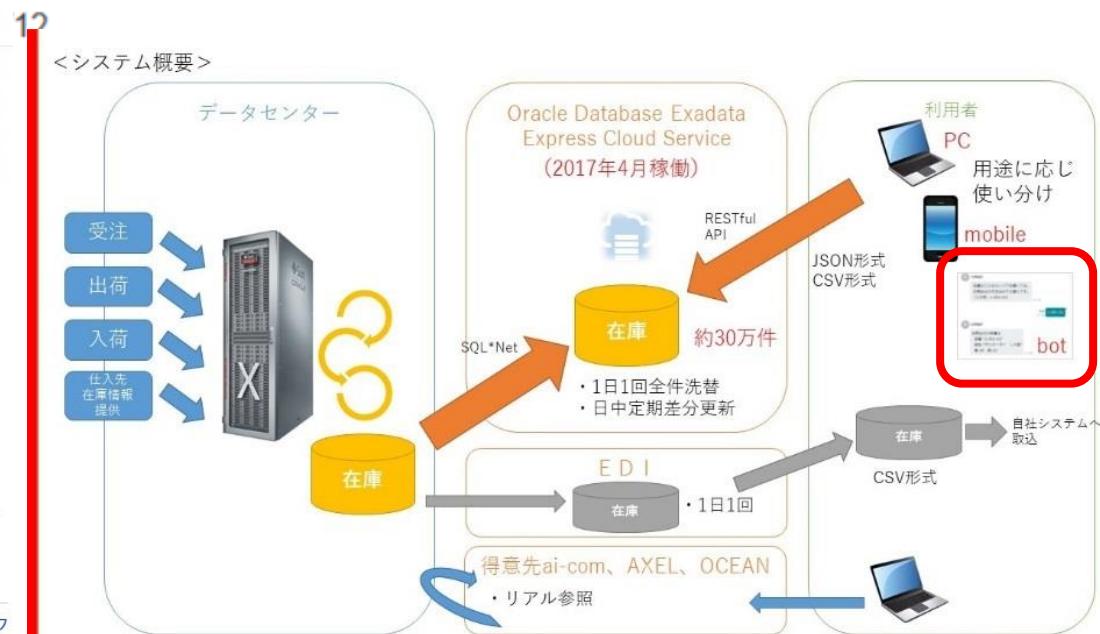


- ・アスクル(「マナミさん」)
— LOHACOのFAQ



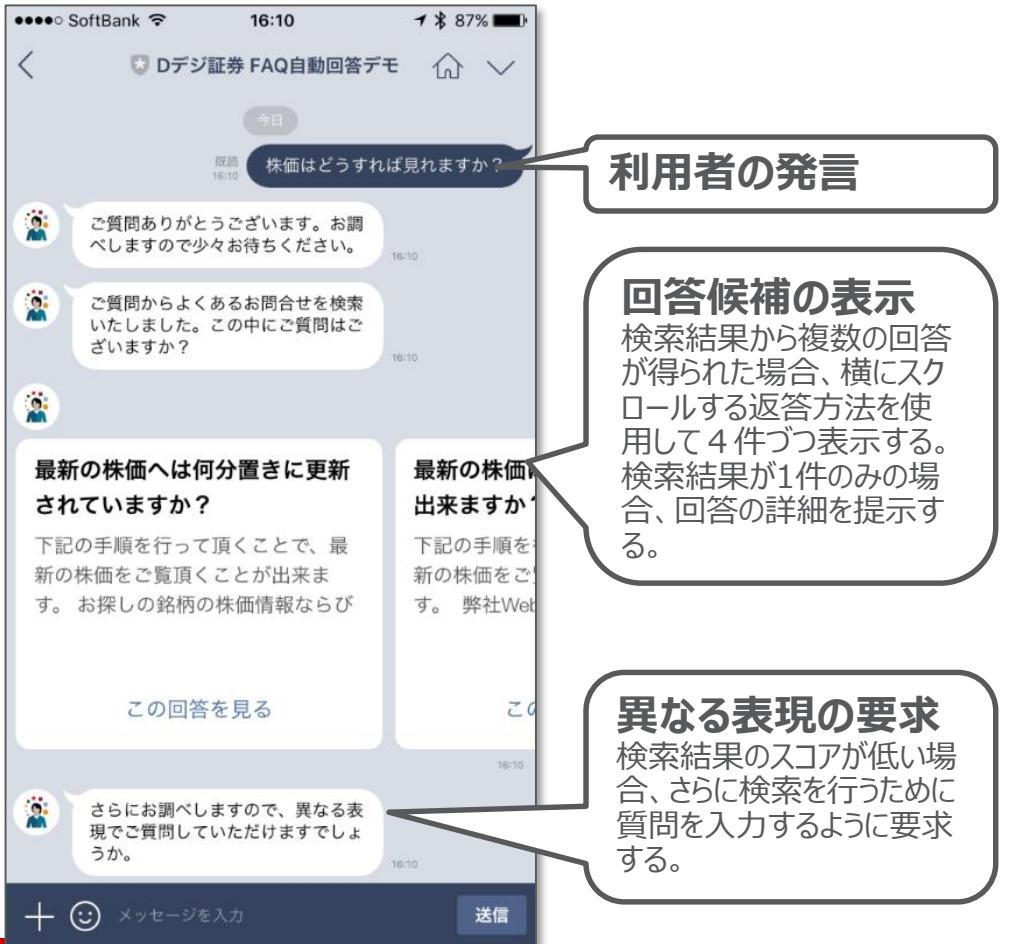
アズワン、オラクルのデータベース・クラウドを導入し、4,300社の販売店へリアルタイムに在庫データを公開

オラクルのデータベース最新版「Oracle Database 12c Release 2」をクラウドで提供する「Oracle Database Exadata Express Cloud Service」を導入し、拡大する販売店ネットワークとの連携を容易にし、事業成長とコスト削減に貢献

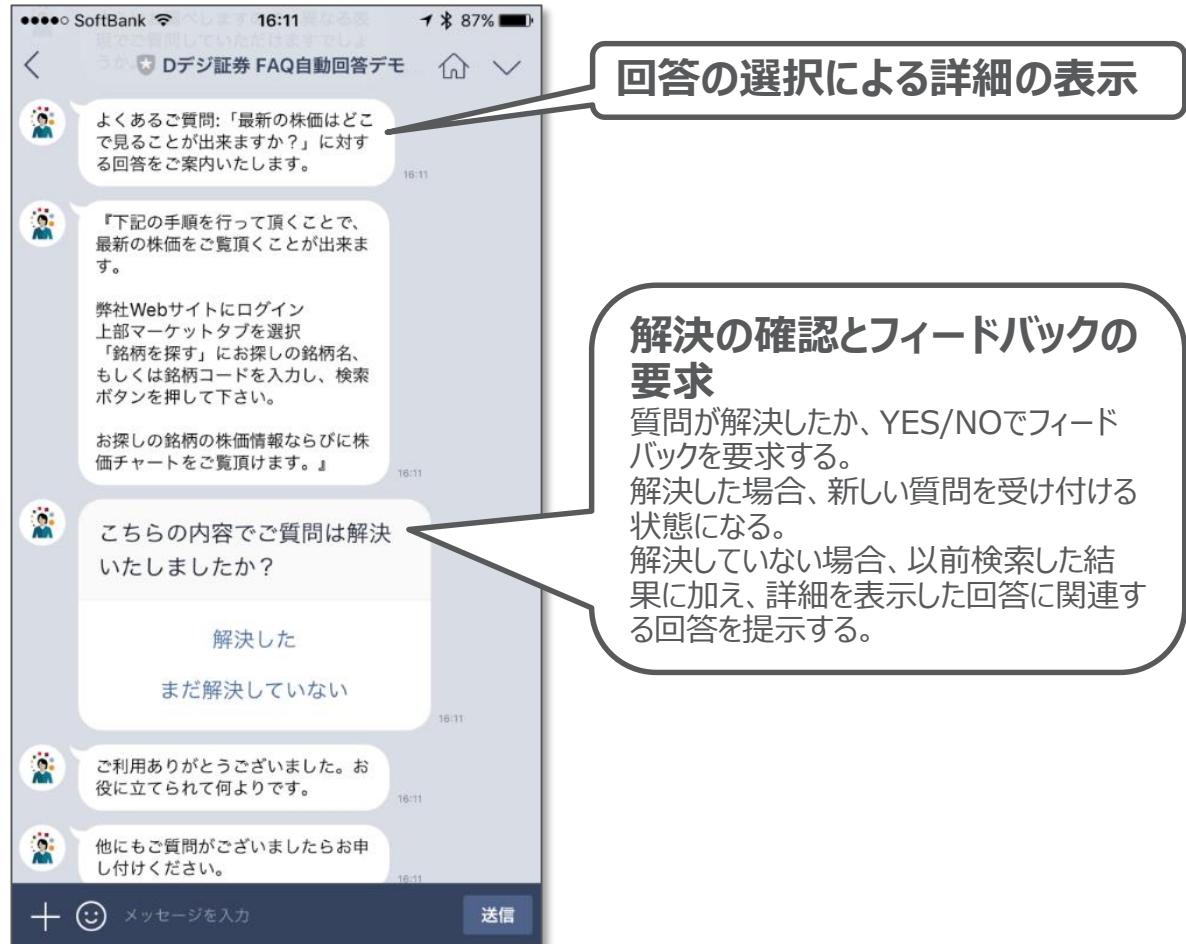


(参考) Oracle SaaS - LINEカスタマーコネクト連携

■質問の受け付けと回答候補の提示



■回答詳細の提示



アジェンダ

- 1 ➔ はじめに
- 2 ➔ サンプル・アプリケーションの概要と構成を理解しよう
- 3 ➔ サンプル・アプリケーションの技術要素を理解しよう
- 4 ➔ サンプル・アプリケーションを使ってみよう

2 サンプル・アプリケーションの概要と構成を理解しよう

- 2-1 → サンプルアプリケーションのフロー
- 2-2 → アプリケーション構成
- 2-3 → Database Cloud Service概要
- 2-4 → Application Container Cloud概要

2 サンプル・アプリケーションの概要と構成を理解しよう

2-1

サンプルアプリケーションのフロー

2-2

アプリケーション構成

2-3

Database Cloud Service概要

2-4

Application Container Cloud概要

サンプルデータ

- delivery_status表
 - order_id : 伝票番号
 - Location : 場所(事業所や倉庫)
 - Status : 配送状況
 - complete_flg : 配達完了フラグ
 - delivery_dt : 配達指定日
 - delivery_tm : 配達指定時間帯
- 伝票番号「10000」～「10031」までのサンプル・データ
 - デモなどをしやすくするため、10000, 10005, 10010のように5の倍数の値は配達未完了(それ以外にもあり)
 - 配達未完了のデータは「配送日指定」処理ができる

メニュー・ボタン



「メニュー」「はじめる」「スタート」などと入力

配送状況確認

DBのデータを検索して表示するサンプル・プログラム



①「伝表番号を入力してください」

②伝票番号を入力

⑤配達状況を送信
(該当伝票がない場合は
「該当伝票はありません」)

Bot

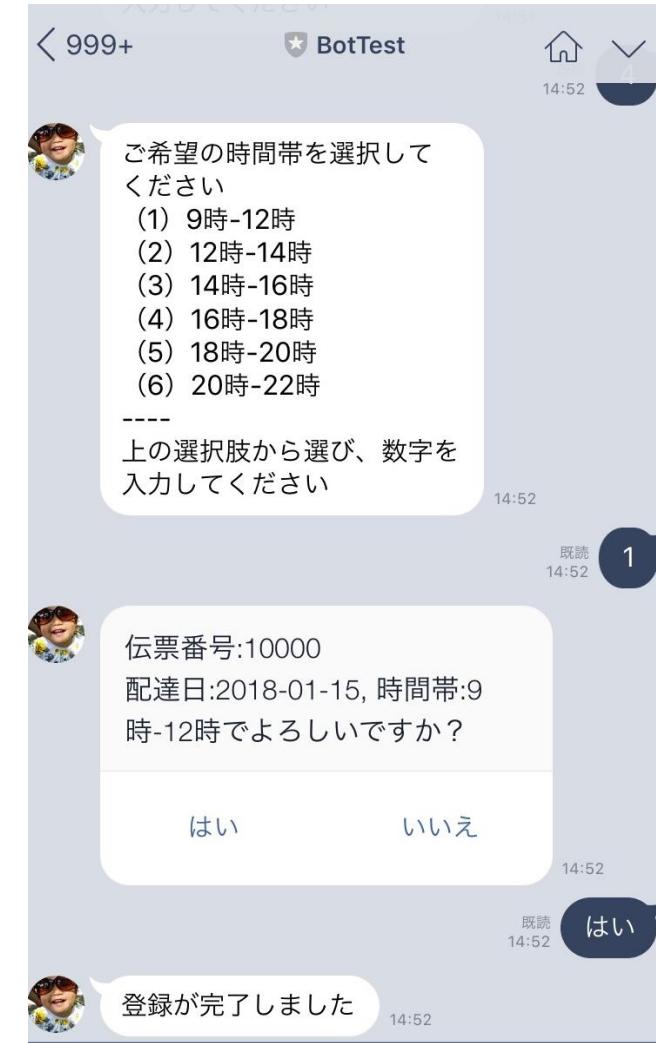
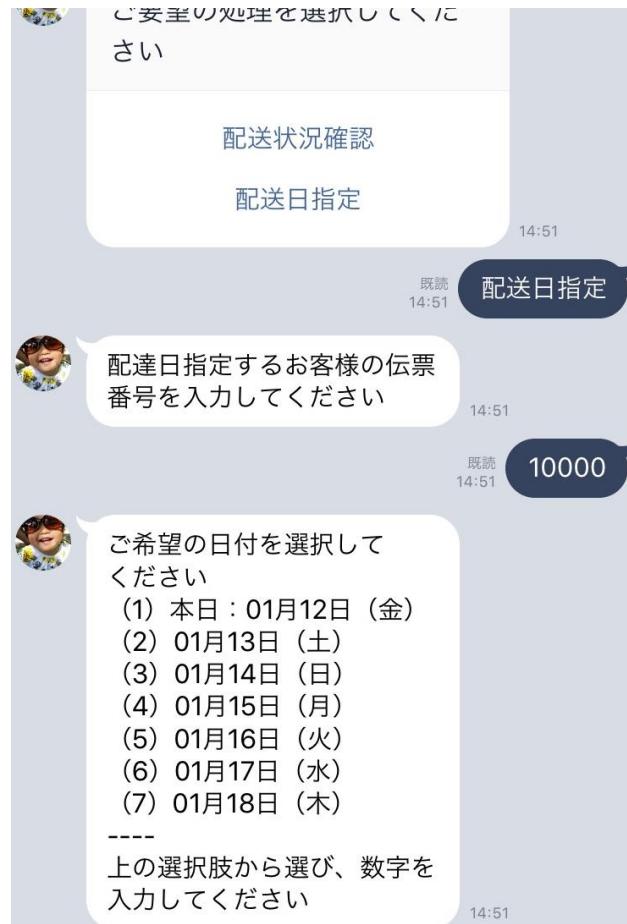
③伝票番号をキーに検索
④検索結果を送信



Database Cloud Service

配送日指定

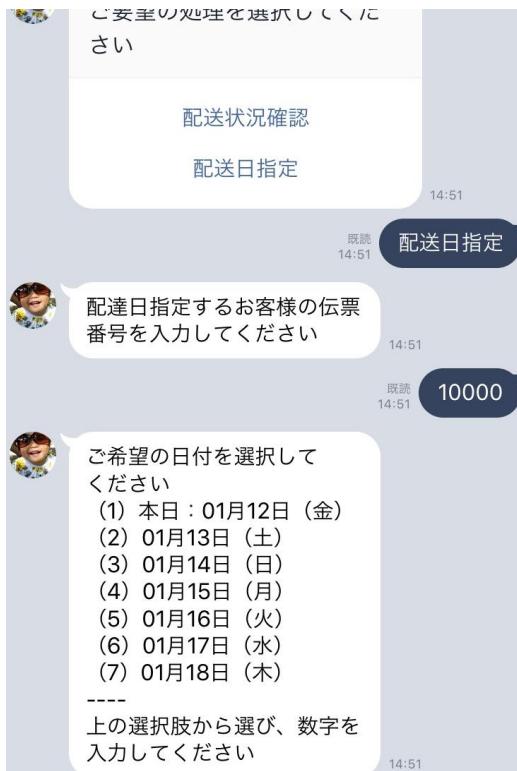
DBのデータを更新するサンプル・プログラム



配送日指定

ユーザーの複数の入力情報をもとにDBを更新するサンプル・プログラム

LINE画面



- ①「伝表番号を入力してください」
- ②伝表番号を送信
- ④日付指定メッセージを送信
- ⑤配送日を選択
- ⑥時間帯選択メニューを送信
- ⑦時間帯を選択
- ⑧登録内容確認ボタンを送信
- ⑨登録確認OK
- ⑪「変更を登録しました」

Bot

③伝票番号をもとに配送済みか
どうか等をチェック

⑩DBを更新



Database Cloud Service

2 サンプル・アプリケーションの概要と構成を理解しよう

2-1

サンプルアプリケーションのフロー

2-2

アプリケーション構成

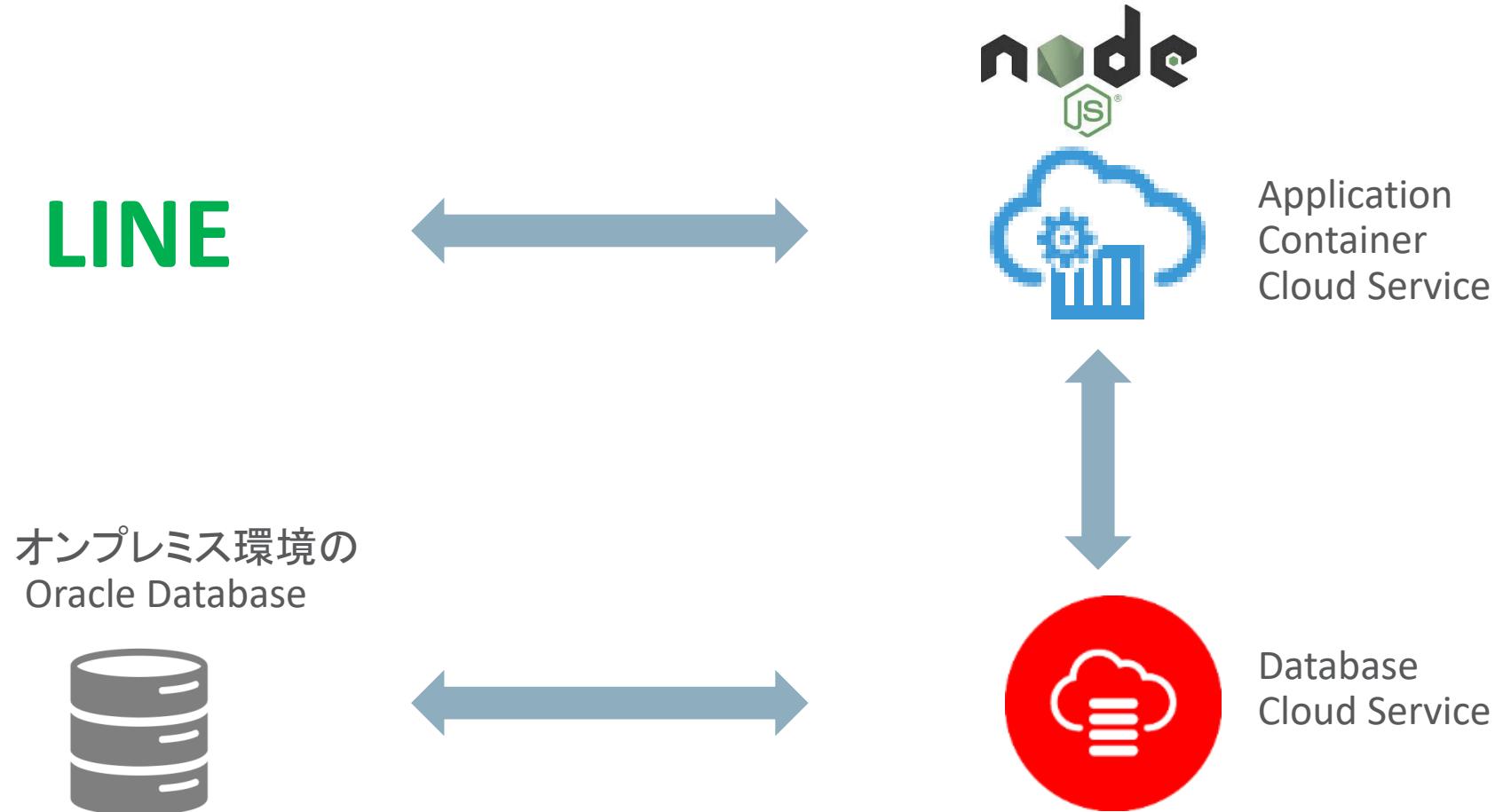
2-3

Database Cloud Service概要

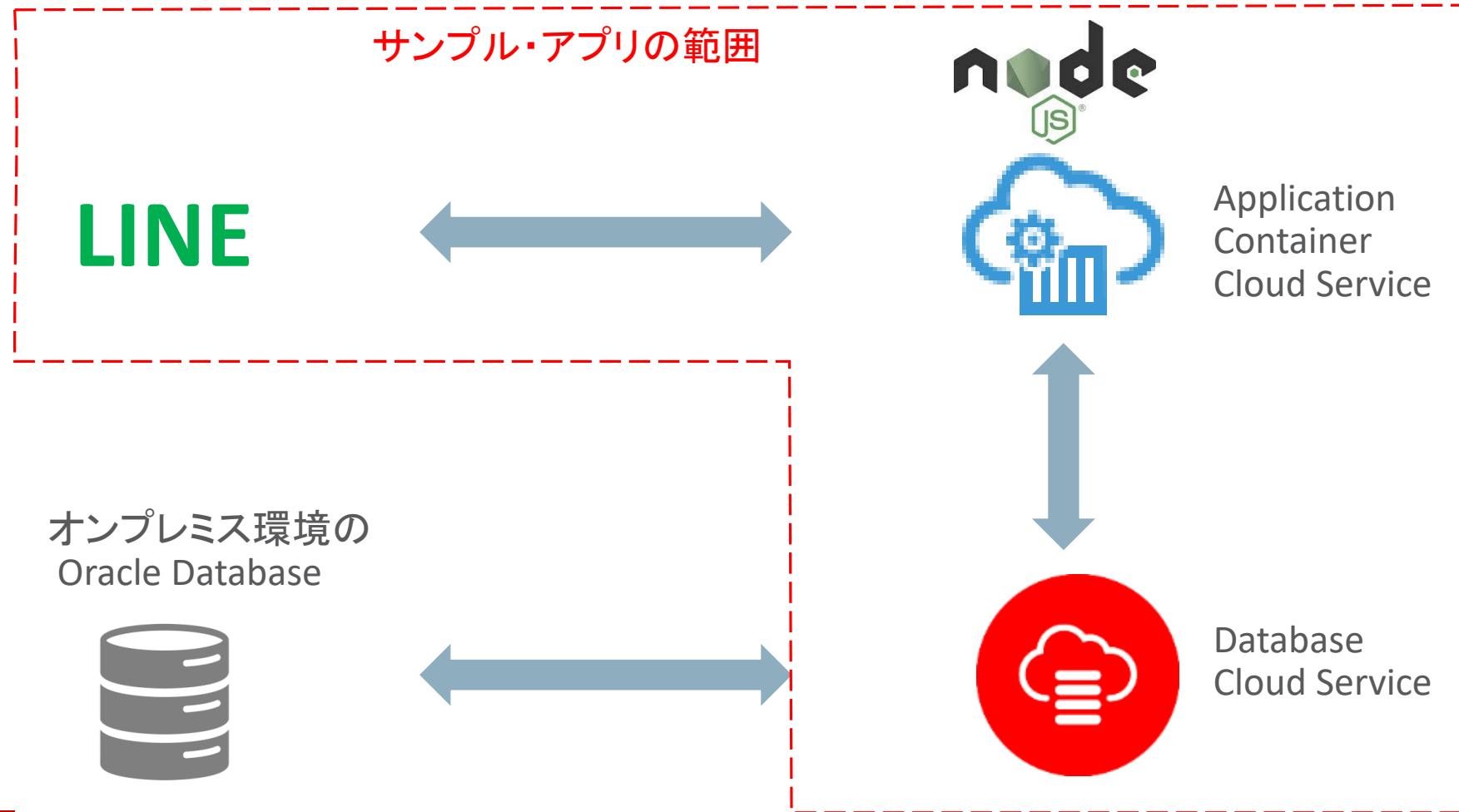
2-4

Application Container Cloud概要

アーキテクチャ

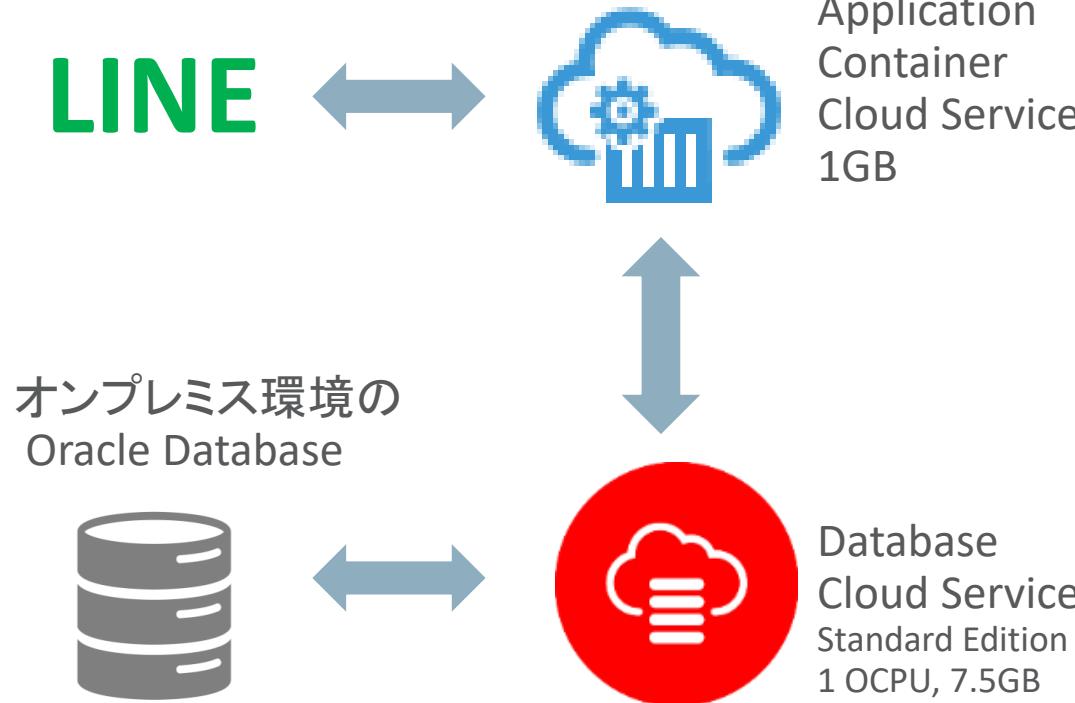


アーキテクチャ



最小構成

Universal Credits



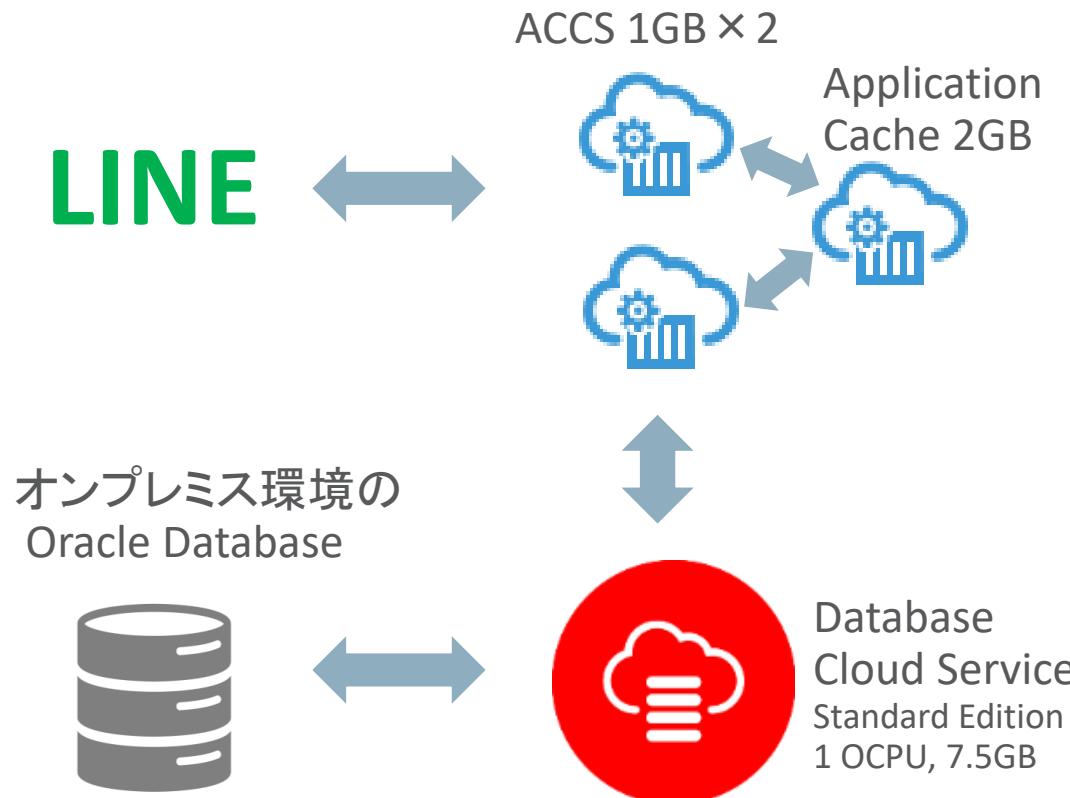
Universal Creditsの場合の最小価格構成

サービス名	単位	金額
Application Container Cloud Service	1 GB	4354.56円/月 (6.048円/時間)
Oracle Database Cloud Service Standard Edition General Purpose	1 OCPU	34836.48円/月 (48.384円/時間)
Compute Classic – Block Storage	150 GB	765円/月
計		約 40,000円

(*) DBバックアップ、アプリログ用 Storage Cloud分は除く

スケール構成の最小構成

Universal Credits



Universal Creditsの場合の最小価格構成

サービス名	単位	金額
Application Container Cloud Service	4 GB	17418.24/月
Oracle Database Cloud Service Standard Edition General Purpose	1 OCPU	34836.48円/月 (48.384円/時間)
Compute Classic – Block Storage	150 GB	765円/月
	計	約 53,000円

(*) DBバックアップ、アプリログ用 Storage Cloud分は除く

(参考)Non-Metered / Metered課金の最小構成価格

Non-Metered

サービス名	単位	金額
Application Container Cloud Service	1 GB (Min 10GB~)	54,000円
Oracle Database Cloud Service Standard Edition General Purpose	1 OCPU	36,000円
Oracle Cloud Infrastructure Block Storage Classic	1 TB	6,000円
Oracle Cloud Infrastructure - Object Storage Classic - Non-metered	1TB	3,600円
		9万9600円



DBバックアップ分は除く。
アプリ・ログ用Object Storage Classicは含む。

スケール構成の最小構成でも金額は同じ

Metered

サービス名	単位	金額
Application Container Cloud Service	1 GB	7,200円
Oracle Database Cloud Service Standard Edition General Purpose	1 OCPU	72,000円
Oracle Cloud Infrastructure Block Storage Classic	100 GB	600円
		約79,800円

(*) DBバックアップ、アプリログ用 Storage Cloud分は除く

2 サンプル・アプリケーションの概要と構成を理解しよう

2-1 サンプルアプリケーションのフロー

2-2 アプリケーション構成

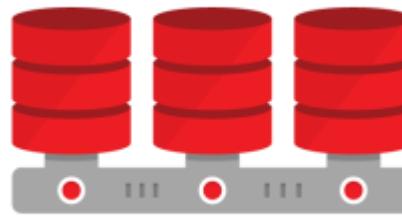
2-3 Database Cloud Service概要

2-4 Application Container Cloud概要

Oracle Database Cloud Service

あらゆるサービスレベルに対応するラインナップ

Exadata Express



1 PDB

~1000 GB

1-4 CPU

シンプルにデータ格納
DB運用不要

Database



VM

1 DB

~11 TB

1~16 CPU

VM環境

Bare Metal

1 Machine

~9 TB

2~36 CPU

専有ベアメタル
サーバー環境

Exadata



1 Exadata

~342 TB

16~336 CPU

Oracle Database
が超高速基盤で稼働

Oracle Database Cloud Service (DBCS): エディション

Standard Edition

- ・ 完全なデータベース・インスタンス
- ・ 表領域暗号化

Enterprise Edition

Adds...

- ・ 全てのEE 標準機能
 - Data Guard
 - Hybrid Columnar Compression(HCC)
 - パラレル処理
 - etc

Management Packs
(Data Masking and Subsetting Pack, Diagnostics and Tuning Packs)



Real Application Testing



Oracle Database Cloud では、全てのエディションで**表領域暗号化**機能を提供

High Performance

Adds...



Multitenant



Partitioning

Advanced Compression



Advanced Security,
Label Security, Database
Vault



Advanced Analytics,
Spatial and Graph, OLAP



Management Packs
(Database Lifecycle
Management Pack, Cloud
Management Pack for
Oracle Database)

**主要なデータベース・オプション機能
が利用可能**

Extreme Performance

Adds...



Real Application Clusters



DB In-Memory

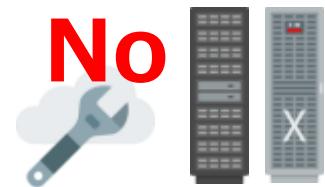


Active Data Guard

**全てのデータベース・オプション機能
が利用可能**

潜在的な管理コストを含めたTCO削減

IaaSはここだけ

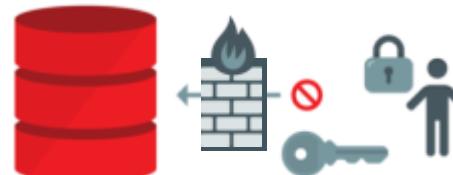


インフラ管理は
不要

さらにPaaSで実現



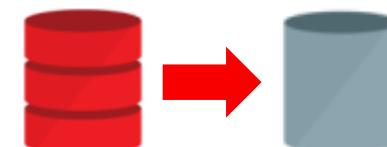
数クリックで
プロビジョニング



データ/通信
デフォルト暗号化



ワンクリックで
パッチ適用



容易なテスト・開発
スナップショット
クローニング



容易な可用性対策
Backup/HA/DR



DB Monitor
クラウド専用
モニタリングツール

2 サンプル・アプリケーションの概要と構成を理解しよう

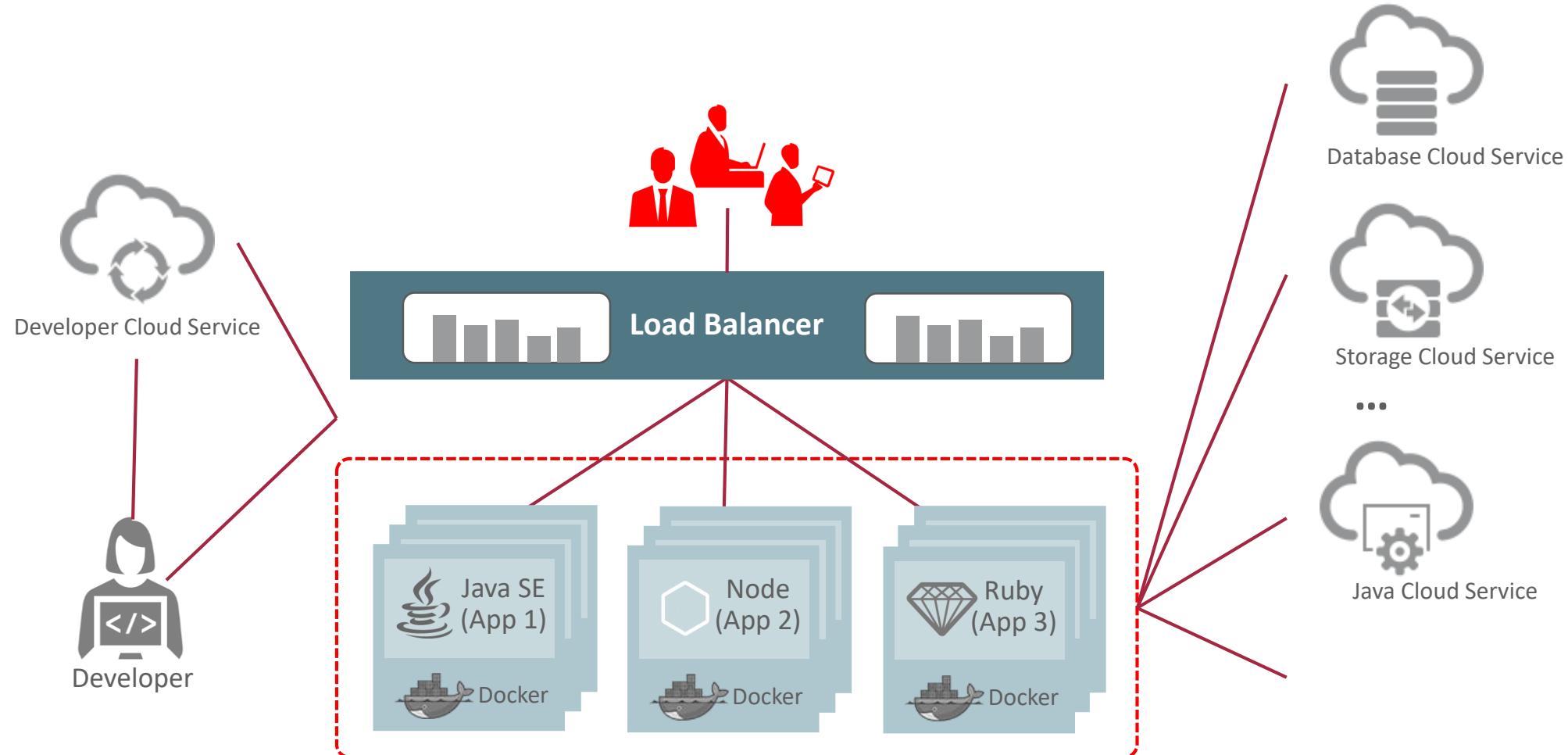
2-1 サンプルアプリケーションのフロー

2-2 アプリケーション構成

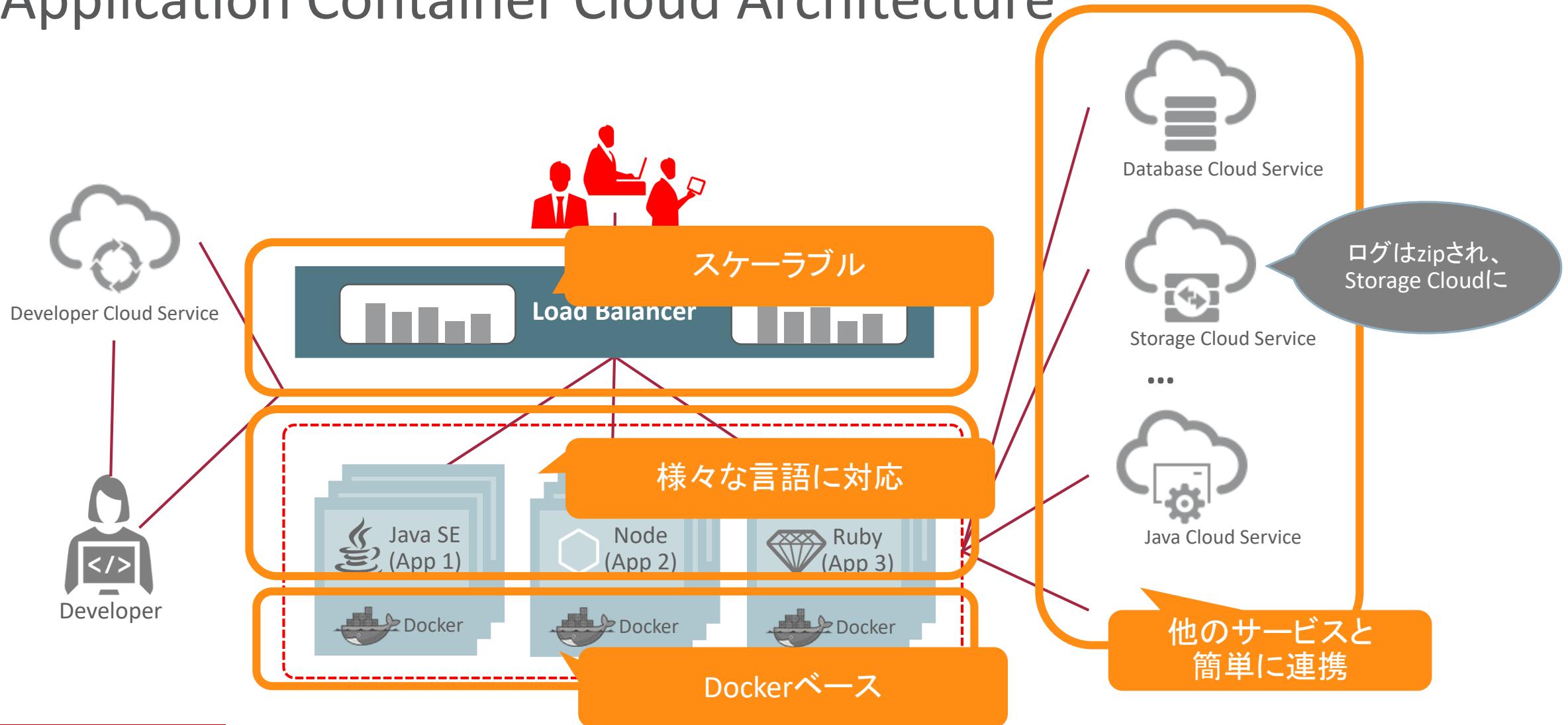
2-3 Database Cloud Service概要

2-4 Application Container Cloud概要

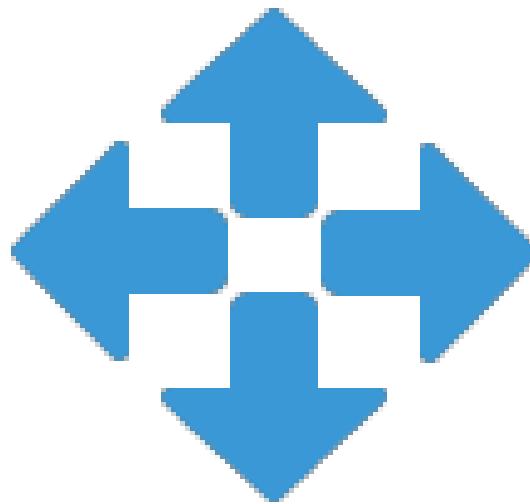
Application Container Cloud (ACCS)



Application Container Cloud Architecture

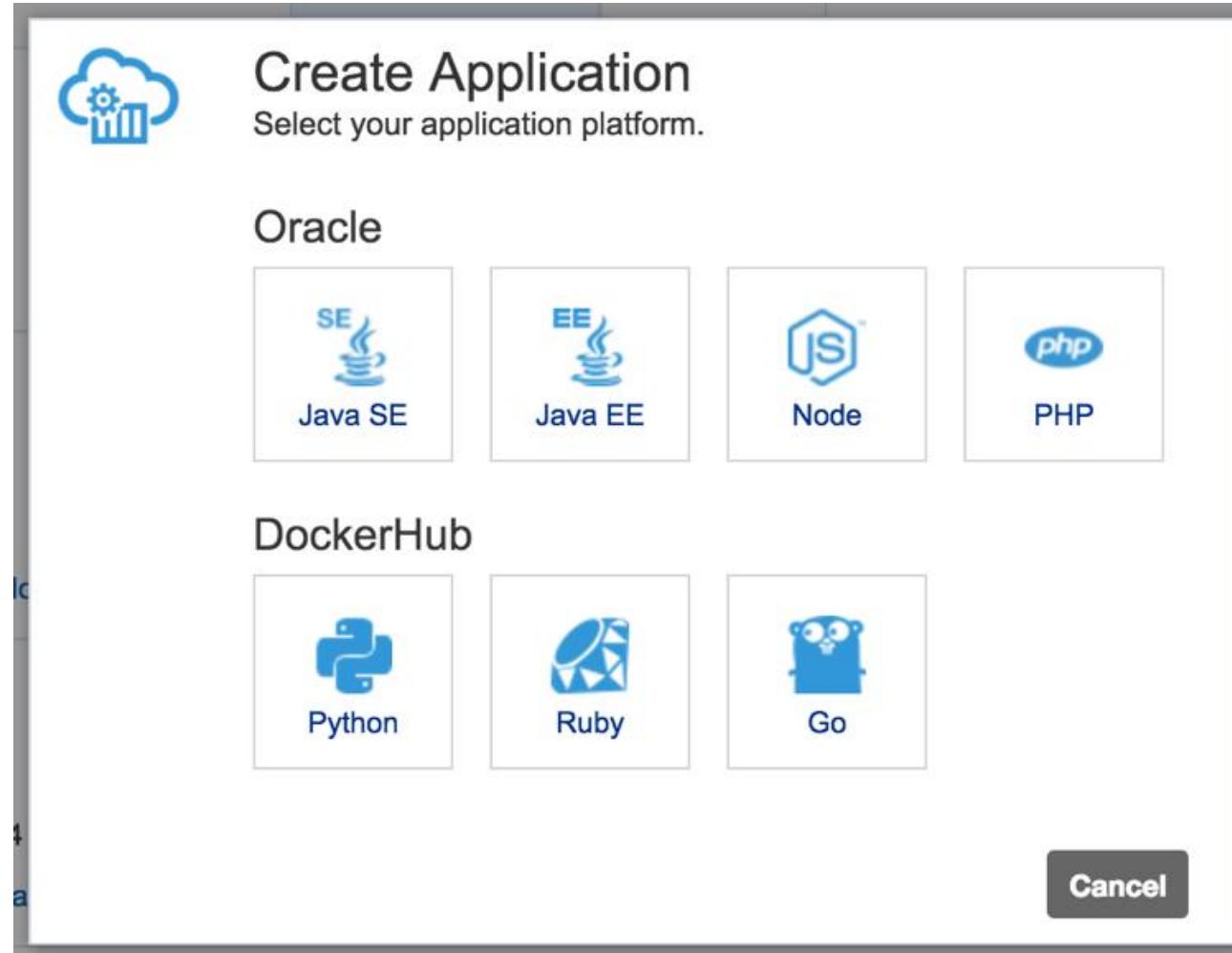


スケーラブル

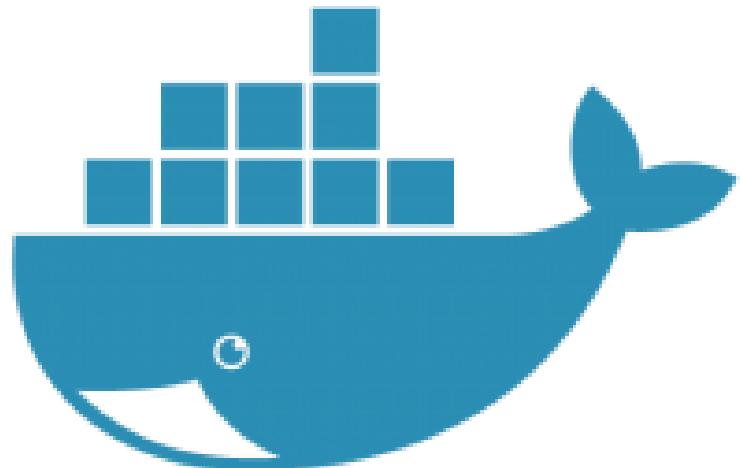


- サービスコンソールやREST APIでスケール変更
 - スケールアウト
 - インスタンス数の増減
 - スケールアップ
 - インスタンス当たりのメモリサイズ増減
- ロードバランサ
 - 完全自动化：ユーザーによる管理不要
 - スケールアウト/イン時に自動認識

様々な言語に対応



Dockerベース



- アプリケーションはDockerコンテナ内のOracle Linuxで動く
 - しかし、OSを意識する必要はない
- ステートレスなアプリケーション向き
 - ステートフルにする場合
 - DBCS/MySQL CSやStorage Classicなどを利用
 - Oracle CoherenceをベースにしたApplication Cacheを提供

他のOracle Cloud Serviceと簡単に連携

REST



Non-REST



- 外部とのREST API経由でアクセス可能なサービスは特に設定せずにアクセス可能
- 非REST経由のサービスは「サービス・バインド」の設定で通信可能
 - 例
 - Java Cloud Service
 - Database Cloud Service

...

アジェンダ

- 1 → はじめに
- 2 → サンプル・アプリケーションの概要と構成を理解しよう
- 3 → サンプル・アプリケーションの技術要素を理解しよう
- 4 → サンプル・アプリケーションを使ってみよう

3 サンプル・アプリケーションの技術要素を理解しよう

3-1

Node.jsでのREST API実装

3-2

LINE Messaging APIとLINE Bot SDK

3-3

ステートの保持、およびApplication Cacheの利用

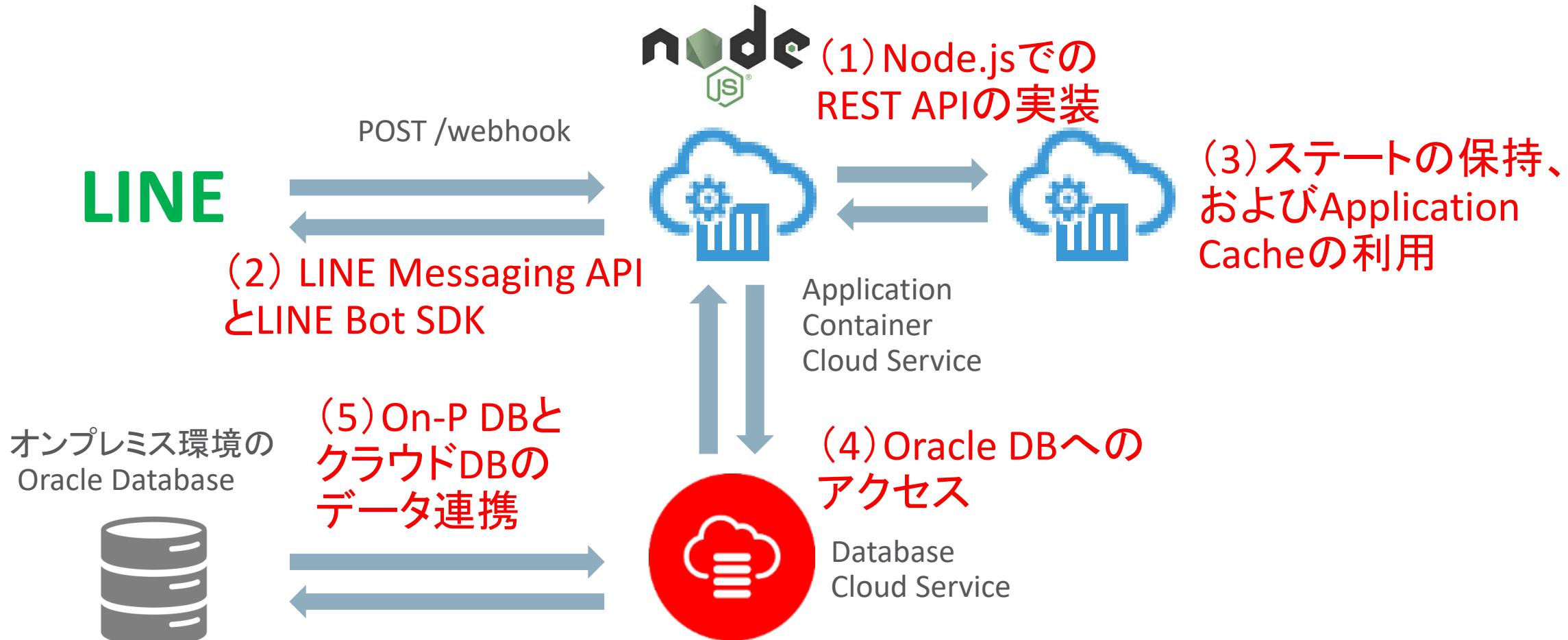
3-4

Node.jsを使ったOracle DBへのアクセス

3-5

On-Premiseとクラウド間のデータ連携

『サンプル・アプリケーションの技術要素を理解しよう』



3 サンプル・アプリケーションの技術要素を理解しよう

3-1

Node.jsでのREST API実装

3-2

LINE Messaging APIとLINE Bot SDK

3-3

ステートの保持、およびApplication Cacheの利用

3-4

Node.jsを使ったOracle DBへのアクセス

3-5

On-Premiseとクラウド間のデータ連携

3 サンプル・アプリケーションの技術要素を理解しよう』

3-1

Node.jsでのREST API実装

3-1-1

Node.js

3-1-2

REST

3-1-3

Restify

Node.js

サーバーサイドのJavaScript実行プラットフォーム

- 特徴

- イベント・ドリブン
- ノンブロッキングI/O(I/Oの処理結果を待たず処理を進める)
 - リアルタイムなアプリケーション向き
 - 多数のアクセスがあるWeb上での処理などに使える
- プロセスはシングルスレッドで、メモリ消費が少ない

Node.jsの使い方

- npm (Node Packaged Modules)
 - Node.jsのライブラリやパッケージを管理するパッケージ・マネージャを実行
 - 利用するコマンド
 - npm init
 - 最初に実行し、プログラムの初期環境設定を作る
 - package.jsonという設定ファイルが作成される
 - npm install <パッケージ名> --save
 - プログラムで使用するパッケージをダウンロードし、package.jsonにパッケージ名とバージョンを追加する
- node
 - Node.jsプログラムを実行
 - 例: node server.js
-

Node.jsのファイル構造

 node_modules	2018/01/10 15:32	ファイル フォルダー
 oracle.js	2017/12/22 14:49	JavaScript ファイル
 server.js	2017/12/19 15:30	JavaScript ファイル
 package.json	2017/12/19 13:09	JSON ファイル

「npm install」でパッケージをインストールすると
Node_modulesディレクトリの下にインストールされる

「npm install」でパッケージをインストールすると
インストールしたパッケージがpackage.jsonに
記録される

「node server.js」で実行

server.js

```
const db = require('./oracle.js');
```

....

oracle.js

```
const oracle = require('oracledb');
```

package.jsonに記録された
node_module下のディレクトリから
oracledbパッケージを読み込む

※日本語の文字を含めている場合、
JSファイルはUTF-8で保存すること

Application Container Cloudへのデプロイ

ZIP化

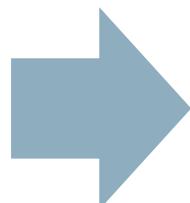
+

 node_modules	2018/01/10 15:32	ファイル フォルダー
 oracle.js	2017/12/22 14:49	JavaScript ファイル
 server.js	2017/12/19 15:30	JavaScript ファイル
 package.json	2017/12/19 13:09	JSON ファイル

- Application Container Cloud固有の設定ファイル
- manifest.json (**必須**)
 - deployment.json (オプション)

Node.js内のプログラム実行の順序

1. 「イベントループ」を起動
2. 読み込んだプログラムを実行
3. イベントキューにイベントが登録
4. 保留されたコールバックのうち、実行可能なものの(I/Oが終わったもの)を選び実行
5. 2.を繰り返し、保留されたコールバックがなくなるとプロセスを終了



- 変数に値をセットしたつもりでも入っていなかったりする
- Returnを使うと値が空のまま返されてしまったりする

非同期処理

コールバック関数

```
const postCallback = function(req, res, next) {  
  (コールバック関数の処理内容)  
}  
  
server.post('/webhook', postCallback);
```

server.postが実行されるとき、
コールバック関数として postCallback が
呼び出される

様々なコールバックの表記方法

関数の引数の中に直接関数を記述

```
server.post('/webhook', function(req, res, next) {  
  (コールバック関数の処理内容)  
});
```

アロー関数と呼ばれる表記方法

```
server.post('/webhook', ((req, res, next) => {  
  (コールバック関数の処理内容)  
});
```

非同期処理

「コールバック地獄」

コールバック関数内から
呼び出した非同期関数で
別のコールバック関数を呼び、
その中に指定された
非同期関数からさらに
コールバック関数が呼び出され...

```
1stAsyncFunction(A, B, function(a, b) {  
    2ndAsyncFunction(e, d, function(E, D, F) {  
        3rdAsyncFunction(i, j, fuction(I, J) {  
            4thAsyncFunction(m, n, fuction(M, N, O) {  
                .....  
            });  
        });  
    });  
});
```

非同期処理

プロミス

- ・「コールバック地獄」による複雑さを回避する仕組み
- ・非同期処理の成功時(resolve)、失敗時(reject)の処理を明示的に書くことができる

```
const funcA = (valA) => {
  return new Promise((resolve, reject) => {
    非同期関数(() => {
      resolve(valA);
    });
  })
}

const funcB = (valB) => {
  return new Promise((resolve, reject) => {
    非同期関数(() => {
      resolve(valB);
    });
  })
}

Promise.resolve()
  .then(() => {
    return funcA(xxx);
  })
  .then((val)=> {
    return funcB(val);
  })
  .then((val) => {
    console.log(val);
  })
  .catch((error) => {
    console.log(error);
  });
}
```

ただ、
この書き方にも
抵抗がある....

非同期処理

async / await

- プロミス固有の then などの表記ではなく、より一般的なプログラム記法に近づける
- asyncを指定した関数内では、awaitをつければ、プロミスが完了してから次の処理に遷移できる
- Node.jsでは7.4以降で利用可能

```
const funcA = (valA) => {
  return new Promise((resolve, reject) => {
    非同期関数(() => {
      resolve(valA);
    });
  }
}

const funcB = (valB) => {
  return new Promise((resolve, reject) => {
    非同期関数(() => {
      resolve(valB);
    });
  }
}

const aAsyncFunc = async (xxx)=> {
  try {
    const valA = await funcA(xxx);
    const valB = await funcB(valA);
    console.log(valB);
  } catch (err) {
    console.log(error);
  }
}

aAsycFunc(xxx);
```

かなりスッキリ

3 サンプル・アプリケーションの技術要素を理解しよう

3-1

Node.jsでのREST API実装

3-1-1

Node.js

3-1-2

REST

3-1-3

Restify

REST(Representational State Transfer)とは

- HTTPをベースにしたステートレスなやりとり
- URIで表現
- HTTPメソッド(GET、POST、PUT、DELETE)を利用

} 疎結合なので
クラウドで利用される



RESTで使われるURIの例

- `http://api.example.com/v1/objects`
 - objects全てを指す
- `http://api.example.com/v1/objects/123`
 - objectsの中のID:123を指す
- `http://api.example.com/v1/objects/123/items`
 - objectsの中のID:123についてitems全てを指す
- `http://api.example.com/v1/objects/123/items/45`
 - objectsの中のID:123についてitemsのID:45を指す

HTTPのメソッド

- HTTPではリクエスト行に {method} {パス名} {HTTP/バージョン} を指定
 - GETメソッド
 - サーバに対してページの取得を要求
 - PUTメソッド
 - URIが指示するリソースを直接作成することをサーバーに要求
 - IDが指定されたURIに対して、更新をする場合など
 - (一般のHTTPではファイルをサーバにアップロードする際に用いられる)
 - POSTメソッド
 - URIを「親」とみなし、その子供のリソースをサーバーが生成することを要求
 - 新規にリソースを追加する場合など
 - (一般のHTTPでは入力フォームで入力した値をサーバに送信する際に用いられる)
 - DELETEメソッド
 - 指定したリソースを削除することをサーバに要求

HTTPのメソッドの例

- `http://api.example.com/objects` に対して GET
 - object情報全体を取得
- `http://api.example.com/v1/objects /123` に対して GET
 - IDが123のobjectを取り出す
- `http://api.example.com/objects` に対して POST
 - object情報を追加など(一意とは限らない)
- `http://api.example.com/v1/objects/123` に対して PUT
 - IDが123のobjectを更新する
- `http://api.example.com/v1/objects /123` に対して DELETE
 - IDが123のobjectを削除する

3 サンプル・アプリケーションの技術要素を理解しよう

3-1

Node.jsでのREST API実装

3-1-1

Node.js

3-1-2

REST

3-1-3

Restify

Restify

- Node.jsでは、アプリ開発を効率的に行うために、様々なフレームワークが提供
- 本サンプル・プログラムではREST API開発向けのRestifyというフレームワークを使用
- Restify Documentation (<http://restify.com/docs/home/>)

The screenshot shows the Restify documentation website. At the top, there is a dark header bar with the Restify logo and a 'Docs' button. Below the header is a navigation sidebar containing links to 'Getting Started', 'API', 'Server API', 'Request API', 'Response API', 'Plugins API', and 'Guides'. The main content area has a large title 'Quick Start' and a sub-section titled 'Setting up a server is quick'. It includes a code snippet demonstrating how to set up a basic server using the Restify framework.

```
var restify = require('restify');

function respond(req, res) {
    res.send('hello ' + req.params.name);
    next();
}

var server = restify.createServer();
server.get('/hello/:name', respond);
server.head('/hello/:name', respond);

server.listen(3000, function() {
    console.log('server listening at 3000');
});
```

Restifyの基本的な流れ

- フレームワークの呼び出し

```
const restify = require('restify');
```

- Restify Serverの作成

```
const server = restify.createServer();
```

- 設定

- sever.pre()もしくはserver.use()で各処理に共通する設定を行う
- sever.pre()
 - ルーター(後述)に処理が振られる前にインタラプト
 - リクエストやヘッダの変更などを行う
- sever.use()
 - 各ルートに共通の処理を指定
 - bodyParser()やqueryParser()など、パーサー指定に応じて、どういったデータが渡されるかが決まる

Restifyの基本的な流れ

- ルート

- GET, POST, PUT, DELETEといったリクエストに対し、実行する処理をマッピング

- GET ⇒ server.get()
- POST ⇒ server.post()
- PUT ⇒ server.put()
- DELETE ⇒ server.del()

- リスニング

```
server.listen(process.env.PORT || 3000, function() {  
    console.log("Node is running...");  
});
```

```
function send(req, res, next) {  
    res.send('hello ' + req.params.name);  
    return next();  
}  
  
server.post('/hello', function create(req, res, next) {  
    res.send(201, Math.random().toString(36).substr(3, 8));  
    return next();  
});  
server.put('/hello', send);  
server.get('/hello/:name', send);  
server.head('/hello/:name', send);  
server.del('/hello/:name', function rm(req, res, next) {  
    res.send(204);  
    return next();  
});
```

3 サンプル・アプリケーションの技術要素を理解しよう

3-1

Node.jsでのREST API実装

3-2

LINE Messaging APIとLINE Bot SDK

3-3

ステートの保持、およびApplication Cacheの利用

3-4

Node.jsを使ったOracle DBへのアクセス

3-5

On-Premiseとクラウド間のデータ連携

3 サンプル・アプリケーションの技術要素を理解しよう

3-2

LINE Messaging APIとLINE Bot SDK

3-2-1

LINE Messaging API

3-2-2

LINE Bot SDK

LINE Messaging API

- ボットアプリとLINEプラットフォーム間でデータをやりとりできる
 - <https://developers.line.me/ja/docs/messaging-api/overview/>
- 処理の流れ
 1. ユーザーがボットにメッセージを送る
 2. webhookがトリガーされる
 3. LINEプラットフォームからアプリのwebhook URLにリクエスト送信
 4. ボットアプリからLINEプラットフォームに、ユーザーへの応答リクエストが送信

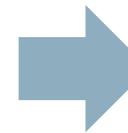
LINEのMessaging API

<https://developers.line.me/ja/index.html> にアクセス

The screenshot shows the main page of the LINE Developers website. On the left is a vertical navigation menu with links like Home, Services, LINE Login, Messaging API, etc. The main area features a large green background with the text "GROW YOUR BUSINESS". At the bottom, there are two prominent green buttons: "LINEログインをはじめる" and "Messaging API(ボット)をはじめる". The second button is highlighted with a red rectangular box. Below each button is a link to a "Start Guide": "LINEログインスタートガイド" and "Messaging API スタートガイド".

本資料では無償の範囲で利用可能な機能のみを利用

月額プラン	メッセージ/月	ホームへの投稿/月	ログインユーザー数	追加機能
フリー ¥0/月	1000通 (1通=1吹き出し) / 月	4	20	
ベーシック ¥5,400/ 月	無制限 (5000人まで※)	無制限	100	<ul style="list-style-type: none">・リッチメッセージ*・リッチメニュー*・音声メッセージ*・動画メッセージ*
プロ ¥21,600/ 月	無制限 (100000人まで※)	無制限	100	<ul style="list-style-type: none">・リッチメッセージ*・リッチメニュー*・リッチビデオメッセージ*・音声メッセージ*・動画メッセージ*・ターゲット指定メッセージ・統計情報(年齢、性別、地域)
プロ(API) ¥32,400/ 月	無制限 (100000人まで※)	無制限	100	<ul style="list-style-type: none">・リッチメッセージ*・リッチメニュー*・リッチビデオメッセージ*・音声メッセージ*・動画メッセージ*・ターゲット指定メッセージ・統計情報(年齢、性別、地域)・Messaging API (Push API含む)



ReplyMessageのみを利用

webhook

アプリの更新情報・イベントを他のアプリにリアルタイム通知

LINEの場合、ユーザーがメッセージを送信すると、Webhook URLとして指定したURLにPOSTリクエストする

受け側のアプリでは、POSTリクエストを受け取ったときに行う処理を記述することになる



Webhookでリクエストを受け取ってから、 リプライするまでの処理の流れ

①Webhook URLに対して
LINEからJSON形式のリクエストが
ポストされる

②署名を検証し、
LINEプラットフォームから送信され
たことを確認

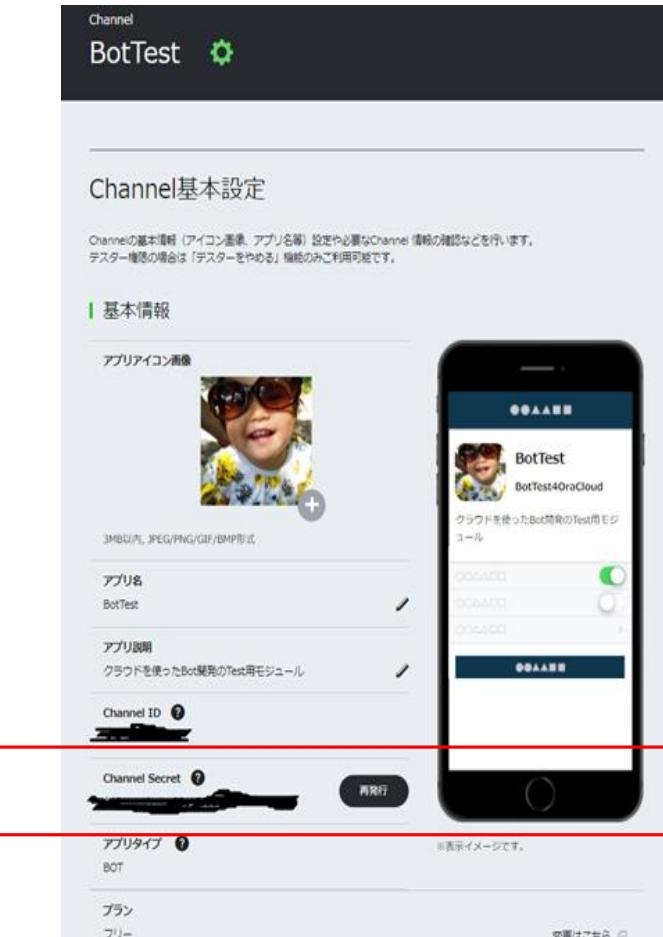
③ステータス 200を返す

④なんらかのメッセージをリプライ

```
{  
  "events": [  
    {  
      "replyToken": "nHuyWiB7yP5Zw52FIkcQobQuGDXCTA",  
      "type": "message",  
      "timestamp": 1462629479859,  
      "source": {  
        "type": "user",  
        "userId": "U4af4980629..."  
      },  
      "message": {  
        "id": "325708",  
        "type": "text",  
        "text": "Hello, world"  
      }  
    }  
  ]  
}
```

署名の検証

チャネル・シークレット、アクセストークン



This screenshot shows the 'Message Send/Receive Settings' page. It includes fields for 'Access Token' (highlighted by a red box) and 'Channel Secret'. Below these are sections for 'Webhook URL' and 'Bot Group Token'. A red box also highlights the 'Channel Secret' field on this page.

チャネル・シークレット:
署名の検証で必要になる秘密鍵

アクセス・トークン:
Authorizationで利用される値

※サンプル・アプリでは、
line.config.json内で
アクセス・トークンと
チャネル・シークレットを
設定します

replyToken

ユーザーからのメッセージのリプライで必要になる値

ユーザーからのリクエスト

```
{  
  "events": [  
    {  
      "replyToken": "nHuyWiB7yP5Zw52FIkcQobQuGDXCTA",  
      "type": "message",  
      "timestamp": 1462629479859,  
      "source": {  
        "type": "user",  
        "userId": "U4af4980629..."  
      },  
      "message": {  
        "id": "325708",  
        "type": "text",  
        "text": "Hello, world"  
      }  
    }  
  ]  
}
```



ユーザーからの送信メッセージ
(リクエスト)ごとに、異なる replyToken が
セットされる

リプライの際にこのreplyTokenを指定

ReplyMessageでは、1リクエストに対し
1回だけリプライ可能
※何らかの理由でリプライに失敗すると、
それ以上リプライできないことに注意

ReplyMessage

1回のリプライには最大5件の複数メッセージを含めることが可能

Navitimeの例



ユーザーが到着駅を入れると、
それに対する1回のリプライで
スタンプと2メッセージの
計3メッセージを返信している

送信するメッセージのタイプ

- ・テキスト
- ・スタンプ
- ・画像
- ・動画
- ・音声
- ・位置情報
- ・イメージマップ

有償プランのみ

サンプル・プログラムで
利用してい
る機能

- ・テンプレート
 - ボタン
 - ・ボタンは4つまで
 - 確認
 - カルーセル
 - 画像カルーセル

PC版のLINEクライアント
では表示できない



テンプレート・メッセージに対するアクション

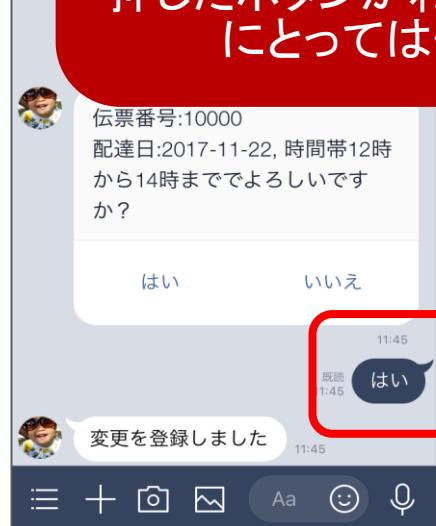
- ・ボタンや画像をタップしたときに実行するアクションの4つのタイプ
 - メッセージアクション (**本サンプル・アプリでメッセージ・アクションを利用する**)
 - ポストバックアクション
 - ・パラメータを指定して送信できる
 - ・textを指定する場合、ポスト・バックアクションとテキストメッセージの2メッセージが送信される
 - ボタンだけを押すと、誤操作したかどうかがわかりにくいので、テキストメッセージを送ることが可能
 - URIアクション
 - ・特定のURIにリダイレクト
 - 日時選択アクション



(参考)メッセージ・アクションとポストバック・アクション

メッセージ表示の有無

メッセージ・アクション、もしくはポストバック・アクションで textを指定した場合、選択したボタンに割り当てられた文字列が表示される。押したボタンがわかるので、利用者にとっては使いやすい。



ポストバック・アクション

Textを指定しない場合

予約しますか？

はい	いいえ
----	-----

ポストバックが送信される

Webhook

Textを指定した場合、送信が2回行われる

予約しますか？

はい	いいえ
----	-----

ポストバックが送信される

textが送信される

Webhook

3 サンプル・アプリケーションの技術要素を理解しよう

3-2

LINE Messaging APIとLINE Bot SDK

3-2-1

LINE Messaging API

3-2-2

LINE Bot SDK

SDKの種類

- <https://developers.line.me/ja/docs/messaging-api/line-bot-sdk/>
- さまざまな言語に対応
 - Java
 - PHP
 - Go
 - Perl
 - Ruby
 - Python
 - **Node.js** サンプル・アプリではこれを利用

SDKの機能

- Webhook
 - 署名のチェック
 - Webhookのイベント・オブジェクト処理
 - イベント・オブジェクトのパース(メッセージの内容を読み取る)
 - Client
 - メッセージ送信の簡易化(**本サンプル・アプリでは ReplyMessageのみ使用**)
 - リッチ・メニュー
 - ユーザー・プロファイル
 - グループ
 - ルーム
- 
- LINE Social API

Webhookでリクエストを受け取ってから、 リプライするまでの処理の流れ

サンプル・アプリケーションでの実装

①Webhook URLに対して
LINEからJSON形式のリクエストが
ポストされる

②署名を検証し、
LINEプラットフォームから送信され
たことを確認

③ステータス 200を返す

④なんらかのメッセージをリプライ

server.js

```
const middleware = require('@line/bot-sdk').middleware;
const restify = require('restify');
const server = restify.createServer();
```

(略)

```
server.post( '/webhook' , //①
  middleware(LINE_CONFIG), //②
  ((req, res, next)=> {
    res.send(200); //③
    req.body.events.map((event) => {
      control.doAction(event); //④はこの中
    });
    next();
  })
);
```

LINE_CONFIGにアクセ
ストokenとチャネル・シー
クレットが設定

3 サンプル・アプリケーションの技術要素を理解しよう

3-1

Node.jsでのREST API実装

3-2

LINE Messaging APIとLINE Bot SDK

3-3

ステートの保持、およびApplication Cacheの利用

3-4

Node.jsを使ったOracle DBへのアクセス

3-5

On-Premiseとクラウド間のデータ連携

3 サンプル・アプリケーションの技術要素を理解しよう

3-3

ステートの保持、およびApplication Cacheの利用

3-3-1

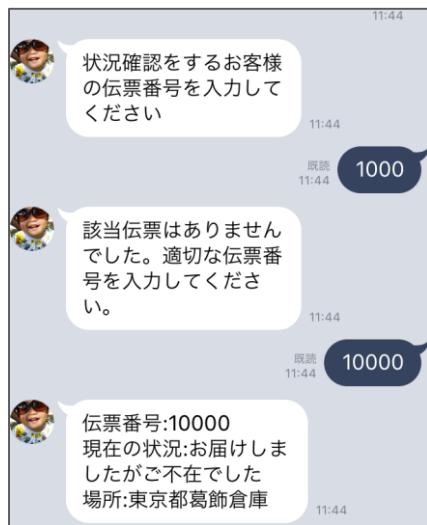
ステートの保存方法

3-3-2

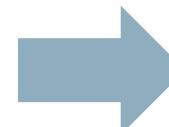
Application Cache

ステートフル／ステートレス

- ・単純なやりとり



伝票番号



Bot

伝票番号の
ステータス

- ・文脈のあるやりとり



Bot



ステートを保持しなくても処理可能

ステートの保持が必要

ステートの保存方法

- Node.jsのメモリキャッシュに格納
 - [memory-cache](#)
 - 高速だがシングル・ノードのみ
- クラスタ化する場合、別の場所(データストア)に格納
 - NoSQL DBやKVSを使用する
 - RDBMSを使用する

データストア層の利用用途

- ・業務データ等の格納とアクセスの提供
- ・アプリ管理用メタデータの保存
 - アプリケーションのコンテキスト
 - ・セッション情報、ステート情報など
- ・アプリケーション・ログの保存
- ・他アプリケーションとのデータ連携の仲介

Oracle PaaSで提供可能なデータストア

- RDBMS

Oracle Database



- Database Cloud Service
※東日本DCで提供
- Exadata Express Cloud Service

MySQL



- MySQL Cloud Service

- No SQL

Apache Cassandra



- Oracle Data Hub Cloud Service

- In-Memory KVS



Application Cache (Application Container Cloud)

※東日本DCで提供

Oracle Coherenceのテクノロジ
を内部的に利用

3 サンプル・アプリケーションの技術要素を理解しよう

3-3

ステートの保持、およびApplication Cacheの利用

3-3-1

ステートの保存方法

3-3-2

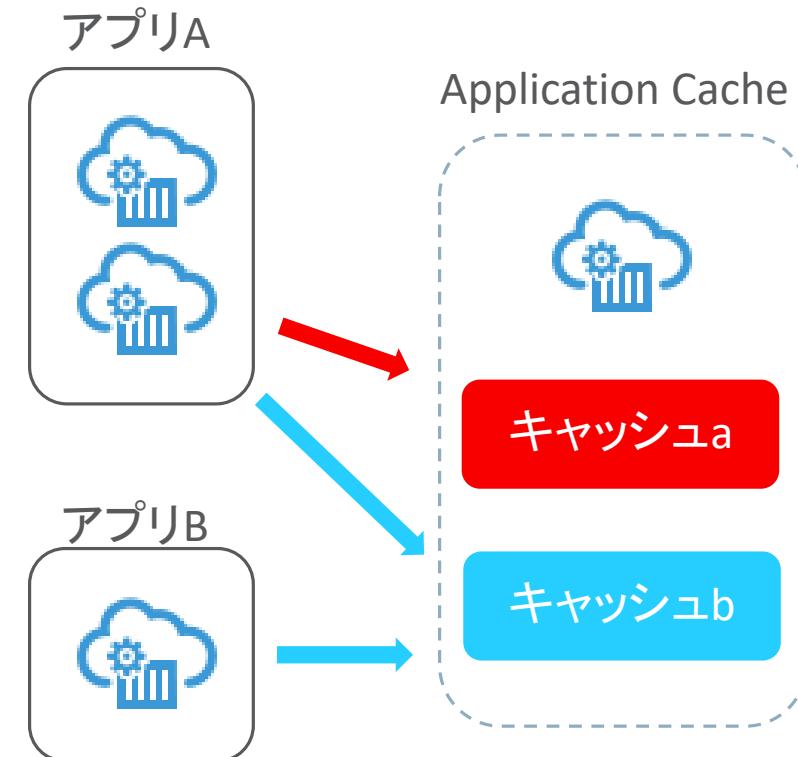
Application Cache

Application Cache

- Application Container Cloudの1コンポーネント
 - KVS型のキャッシュ・サービス
 - Oracle Coherenceベース
 - アクセスは Application Container Cloud経由
- Application Cacheサービス内に複数のキャッシュを使用できる
- クラスタ化可能
 - データは冗長化されて自動配置
 - 障害が発生すると、データの自動再配置
- メモリサイズは、1 GB利用する場合、全体で 2GB必要

ACCSアプリとApplication Cacheの関係

- 1アプリケーションは1キャッシュサービスにだけバインドできる
- 1キャッシュ・サービスに複数のキャッシュを含められる
- 複数アプリが1つのキャッシュサービスや、その中の同一のキャッシュを利用することができる



1 Application Cacheサービスを複数ノード構成すると、データが冗長化されながら自動分散配置される

accs-cache-handler

- Node.jsでApplication Cacheにアクセスするためのパッケージ
 - ただし Oracleが提供しているわけではない
- accs-cache-handlerのメリット
 - 利用方法が簡単
 - Application Cacheの有無にかかわらず利用が可能
 - Application Cacheと BINDされた環境 ⇒ Application Cacheを利用
 - Application Cacheと BINDされない環境 ⇒ ハッシュマップを利用
 - プログラムを変更せずに、どちらの環境でも利用可能
- 参考記事「Oracle ACCS の Application Cacheのアクセスに、NPMパッケージ accs-cache-handlerを使ってみた」
 - <https://qiita.com/kaohas/items/a2a0a1f0880109aa8377>

accs-cache-handlerの例

```
var Cache = require('accs-cache-handler');

var cacheName = "BrandNewAppCache";
var objCache = new Cache(cacheName);

//Store an object
objCache.put("CachingKey", "CachingValue", function(err) {
  if(err){
    console.log(err);
  }
  //Retrieve the object
  objCache.get("CachingKey", function(err, result){
    if(err){
      console.log(err);
    }
    console.log(result); // "CachingValue"
    //Delete the object
    objCache.delete("CachingKey", function(err){
      if(err){
        console.log(err);
      }
    });
  });
});
```

<https://www.npmjs.com/package/accs-cache-handler>

← キャッシュ名を設定

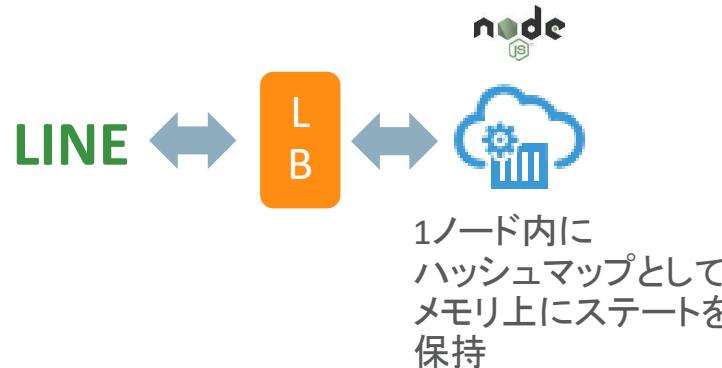
← putでデータを保存 or 更新

← getでデータを取得

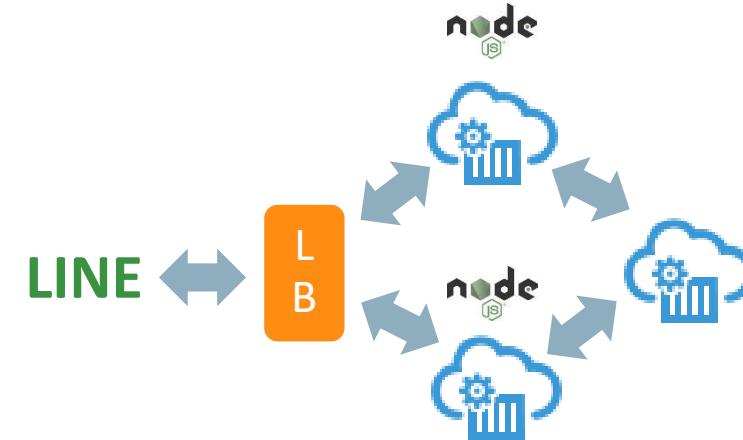
← deleteでデータを削除

サンプル・アプリの Application Container Cloud構成

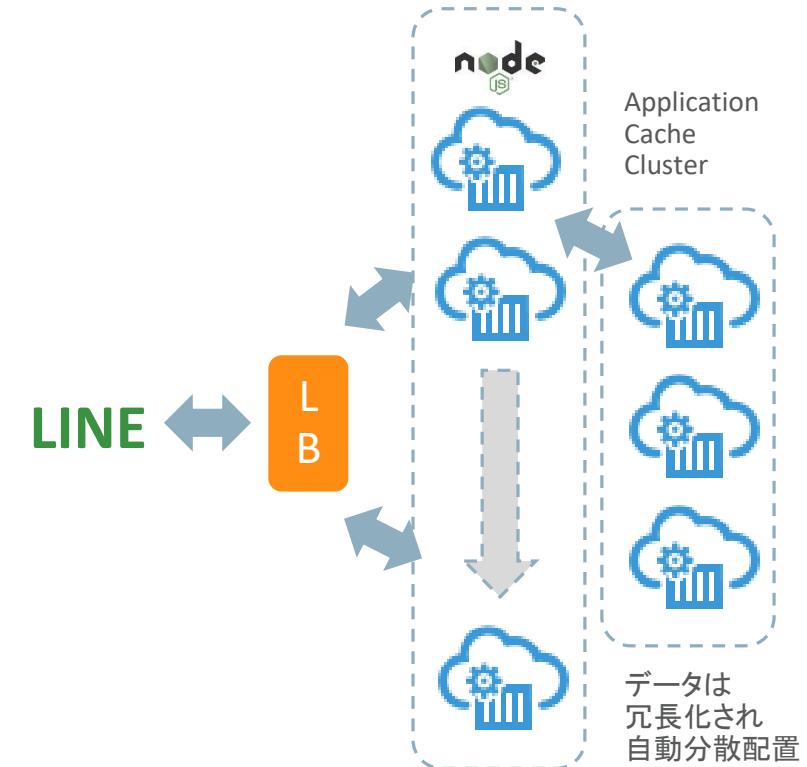
- ・シングル・ノード



- ・複数ノード(minimum)



- ・スケール構成



accs-cache-handlerを使うと、これらの構成でアプリの書き換えは不要

3 サンプル・アプリケーションの技術要素を理解しよう

3-1

Node.jsでのREST API実装

3-2

LINE Messaging APIとLINE Bot SDK

3-3

ステートの保持、およびApplication Cacheの利用

3-4

Node.jsを使ったOracle DBへのアクセス

3-5

On-Premiseとクラウド間のデータ連携

node-oracledb(oracledb) パッケージ

- Node.js で Oracle Database にアクセスするプログラムを作成するにあたり、本資料では node-oracledb を使用
- (参考)
 - [HTTPS://WWW.NPMJS.COM/PACKAGE/ORACLEDDB](https://www.npmjs.com/package/oracledb)
 - [HTTPS://GITHUB.COM/ORACLE/NODE-ORACLEDDB/BLOB/MASTER/DOC/API.MD](https://github.com/oracle/node-oracledb/blob/master/doc/api.md)
- node-oracledb は オラクルが 提供
 - oracledb パッケージは node-oracledb によって 提供される パッケージ
 - Node.js プログラム内では、 var oracle = require('oracledb') と 指定して呼びだす

oracledbパッケージでのSQL実行

- DBへのアクセスはパターン化可能
- プログラムの流れ
 1. Connection PoolingからConnectionを取得
 2. SQLをExecute
 - バインド変数に値を指定
 - SELECTの場合はresultを取得
 3. (必要に応じて commit)
 4. Connectionをクローズ

oracledbパッケージでのSQL実行

oracle.js内の使用例

```
// コネクションをリリース
const doRelease = (conn) => {
  conn.release((err) => {
    if (err) {
      logger.warn(`doRelease: ${err}`);
    }
  });
}
```

※オレンジ色の文字が
Oracledbの提供する関数

```
// SQL実行用共通ファンクション
exports.executeSQL = async (sqltext, bind, option) => {
  let conn = null;
  try {
    // コネクションをプールから取得
    conn = await oracledb.getConnection(connectionProperties);
    // SQLを実行
    const result = (option == null) ? await conn.execute(sqltext, bind) :
      await conn.execute(sqltext, bind, option);
    // SELECT以外はcommit
    if(sqltext.trim().toUpperCase().indexOf("SELECT") != 0)
      await conn.commit();
    doRelease(conn); // コネクションをリリース
    return result;
  } catch(err) {
    logger.error(`executeSQL: ${err.message}`);
    if(conn != null)
      doRelease(conn);
    throw err;
  }
}
```

ACCS上のoracledbパッケージの実行

- ACCSの設定でDBCSをサービス・バインド
 - プログラム内ではユーザー名、パスワード、接続文字列などは変数化しておく
 - 例
 - process.env.DBaaS_USER_NAME
 - process.env.DBaaS_USER_PASSWORD
 - process.env.DBaaS_DEFAULT_CONNECT_DESCRIPTOR
 - バインド時にこれらの変数に値がセットされる
- npm install oracledb を実行しない
 - ACCS上にはnode-oracledbが組み込まれている
 - アップロードするZIP内に oracledbが含まれていると、アプリが起動しないことも

DBアクセス用ユーザー/パスワードの環境変数

Application Container Cloudでは、DBCSのサービス・バインド時に
ユーザー名やパスワードが環境変数としてセットされる。
環境変数への参照は「process.env.環境変数」で行う。

サンプル・アプリケーションのoracle.jsの一部

```
let connectionProperties = {
  user: process.env.DBAAS_USER_NAME
  password: process.env.DBAAS_USER_PASSWORD
  connectString: process.env.DBAAS_DEFAULT_CONNECT_DESCRIPTOR
  stmtCacheSize: process.env.DBAAS_STATEMENT_CACHE_SIZE
  poolMin: 1,
  poolMax: 10
};
```

```
"c##bot",
"c##bot",
"",
10,
```

環境変数が設定されていない
場合はこちらが参照される

DBアクセス用ユーザー/パスワードの環境変数

- Database Cloud Service
 - DBAAS_USER_NAME
 - DBAAS_USER_PASSWORD
- Exadata Express Service
 - EECS_USER
 - EECS_PASSWORD
- MySQL Cloud Service
 - MYSQLCS_USER_NAME
 - MYSQLCS_USER_PASSWORD
- Data Hub Cloud Service
 - DHCS_USER_NAME
 - DHCS_USER_PASSWORD

3 サンプル・アプリケーションの技術要素を理解しよう

3-1

Node.jsでのREST API実装

3-2

LINE Messaging APIとLINE Bot SDK

3-3

ステートの保持、およびApplication Cacheの利用

3-4

Node.jsを使った Oracle DBへのアクセス

3-5

On-Premiseとクラウド間のデータ連携

データ連携方式の検討

- データ連携のタイミング
 - リアルタイム
 - バッチ連携
- アクセス方法
 - Oracle DB固有
 - 汎用的(Oracle DB以外にも適用可能)

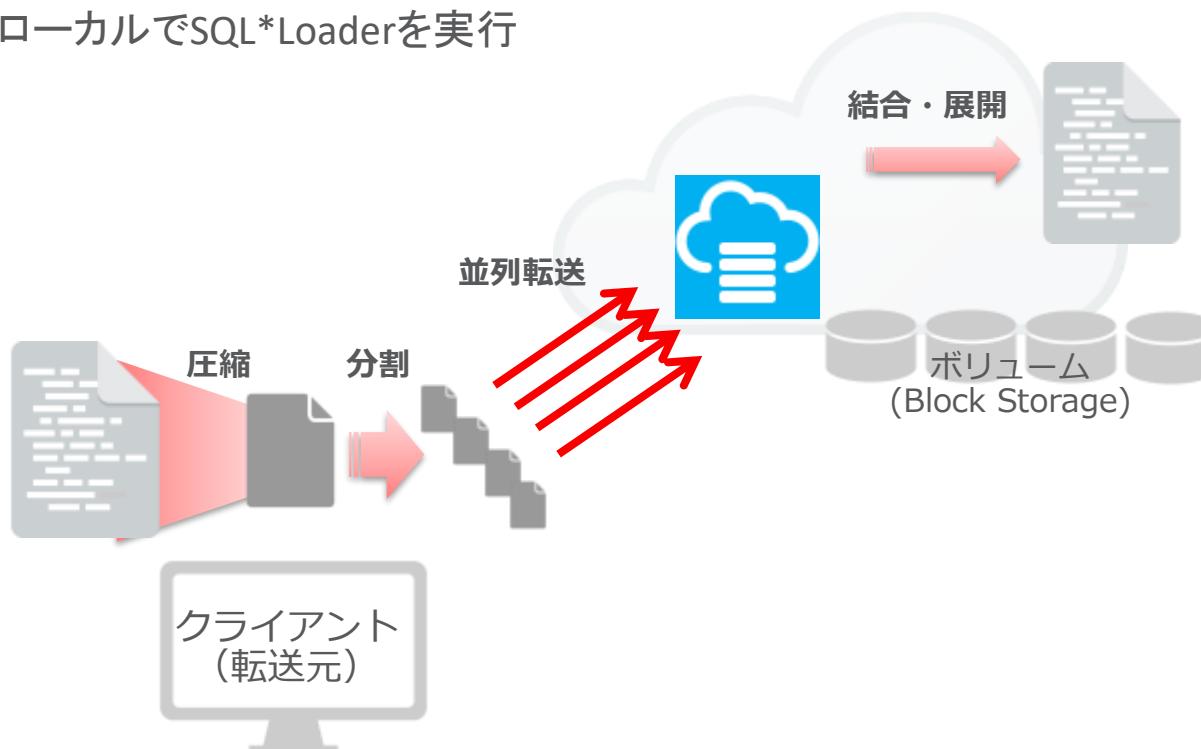
データ連携方式の検討

方式	リアルタイム/バッチ	備考
製品固有(Oracle DB固有)	Database Link	リアルタイム
	レプリケーション	リアルタイム DB 12.2では未サポート
	ファイル転送 + SQL*Loader	バッチ DBCSはPaaSだがOSにログイン可能
汎用的	DBアクセス用REST APIにHTTPSでアクセ ス	リアルタイム/バッチ Oracle DBの場合、 ・ORDSを利用する ・APサーバーと組み合わせてREST APIを実装する
	GoldenGate Cloud Service	リアルタイム

Database Cloud Serviceインスタンスへのファイル転送

- DBCSは PaaSだが、OSへのログインが可能
- OS機能を利用したデータ転送／ロードが可能

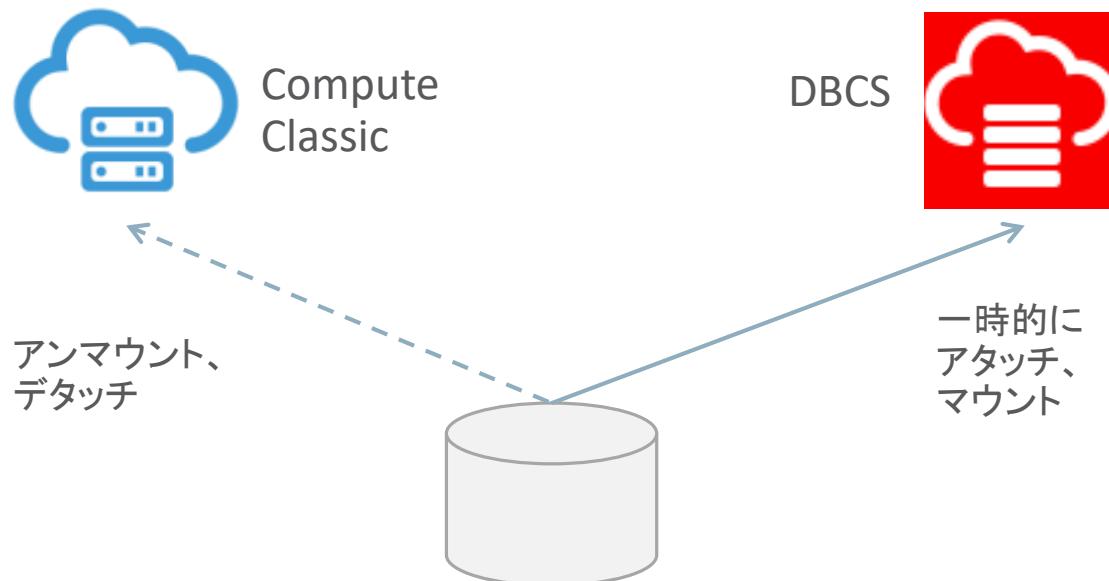
SCPやSFTP(SSHアクセス可能なネットワーク設定に)
ローカルでSQL*Loaderを実行



- 考慮点1 : ボリュームの空き領域
 - 転送先インスタンスに空き容量が必要
 - 必要であれば、一時的にIaaSのインスタンスを立ち上げて、そこをステージング領域として利用
- 考慮点2 : 転送の高速化の検討
 - 圧縮 → 転送 → 展開
 - 分割 → 転送 → 結合
 - ツールやプロトコルの工夫

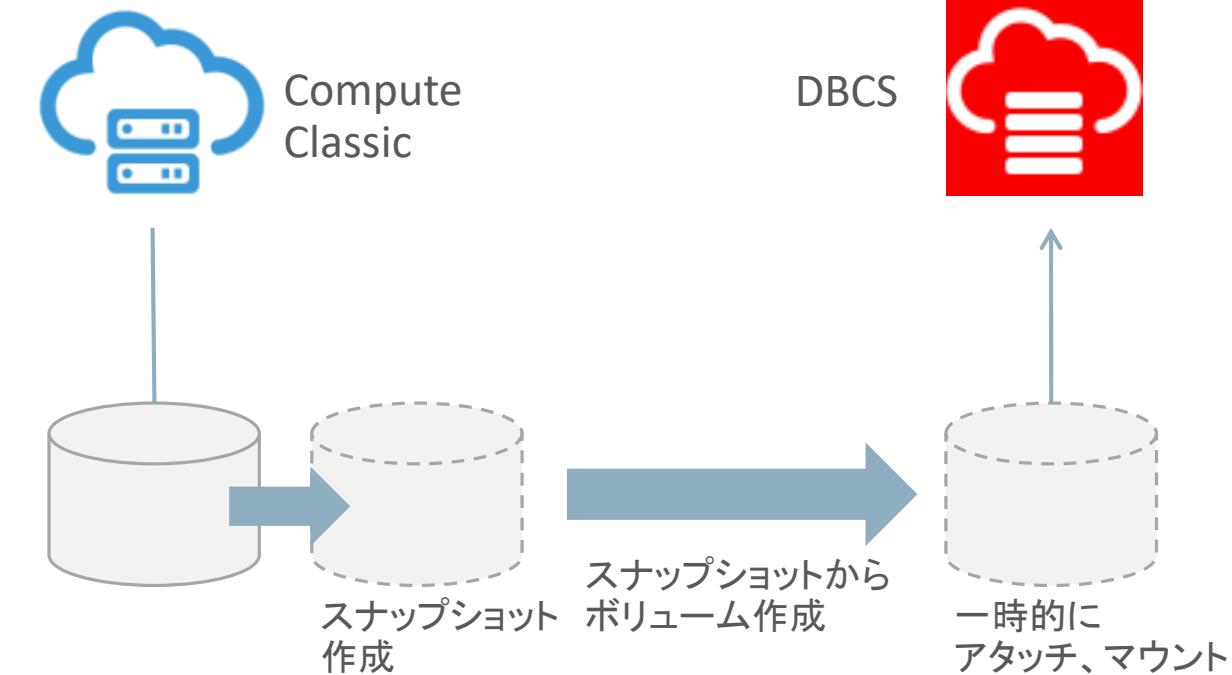
(参考)ステージング用IaaSインスタンスからのデータ移動

ストレージのアタッチの切り替え
による他インスタンスへのデータ共有



※オーケストレーション内でアタッチを指定せず、
REST APIやCLIでアタッチ／デタッチすること

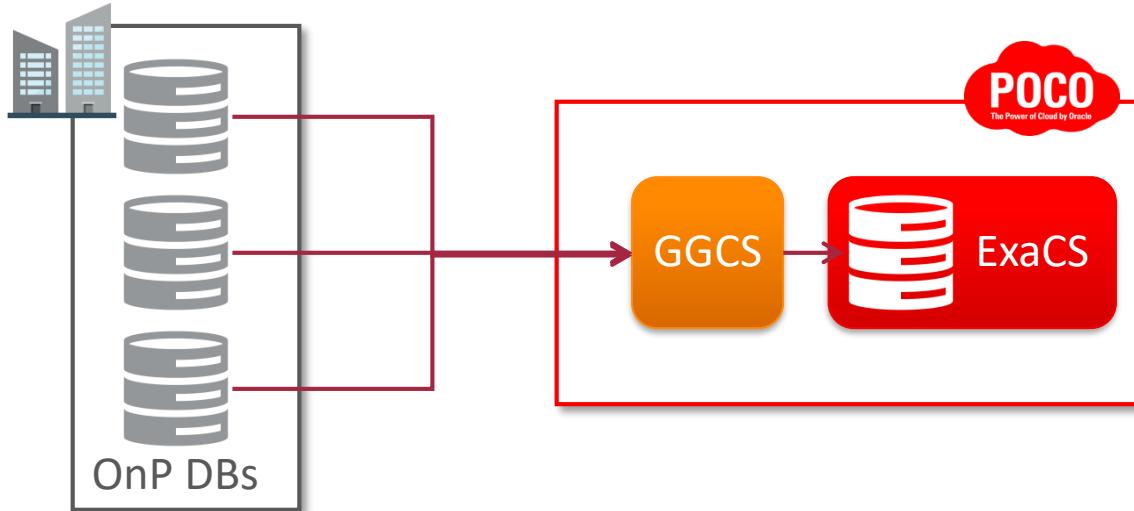
スナップショットを使った
他インスタンスへのデータ共有



- 同一サイト内ならコロケート・スナップショットで可能
- サイトをまたがる場合はリモート・スナップショットを利用

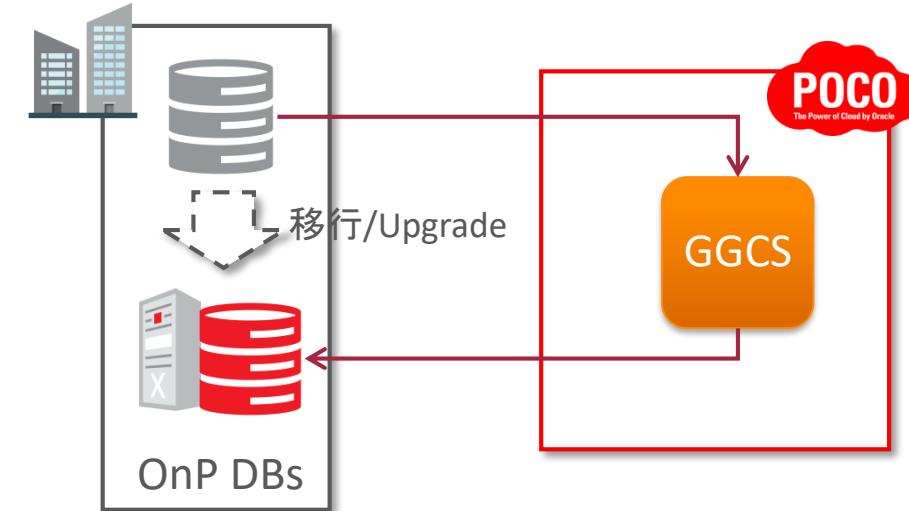
GoldenGate Cloud Service

既存DBのデータをクラウドで有効活用



- 既存システムはそのままに、データだけExaCSとリアルタイム連携
- クラウド上でのリアルタイムデータ活用が可能
- OnP側へのOGGインストールは不要
(latencyによってOGGの導入を要検討)

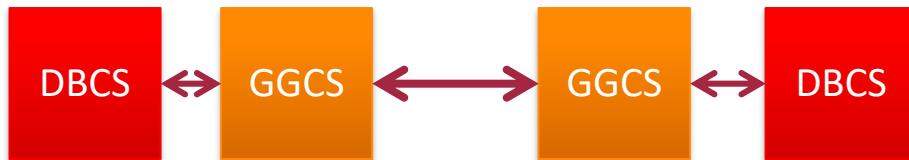
既存DBの移行/UpgradeにGGCSを活用



- クラウド上のGGCSを経由して新旧DBの連携を実施することで、OnP側のDB移行を実現。
- サービス停止時間極小化に加え、ソフトウェアのインストールも不要
- ※システム規模によって、OnPのOGGを用いる必要がある場合もあるので注意

GoldenGate Cloud Service : Connection(接続)について

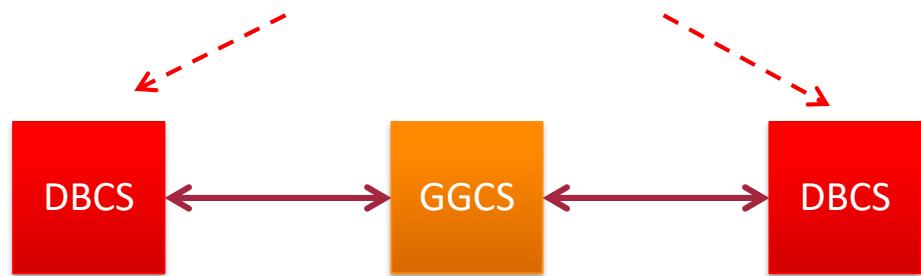
今まで



- DBCSに対して1:1でGGCSが必要
(特にConnectionという考え方無し)

これから

接続対象1つにつき 1Connection と理解してください。
DBユーザー や Session 数ではありません。



- GGCS1つで複数接続が可能
- GGCSの1OCPUあたり、1 Connection 接続可能
- GGCSは4OCPU 単位で追加可能

アジェンダ

- 1 ➔ はじめに
- 2 ➔ サンプル・アプリケーションの概要と構成を理解しよう
- 3 ➔ サンプル・アプリケーションの技術要素を理解しよう
- 4 ➔ サンプル・アプリケーションを使ってみよう

4 サンプル・アプリケーションを使ってみよう

- 4-1 → サンプル・アプリケーションのダウンロード
- 4-2 → サンプル・アプリケーションのプログラム構造
- 4-3 → サンプル・アプリケーションのローカル環境での実行
- 4-4 → サンプル・アプリケーションのクラウドへのデプロイ

4 サンプル・アプリケーションを使ってみよう

4-1

サンプル・アプリケーションのダウンロード

4-2

サンプル・アプリケーションのプログラム構造

4-3

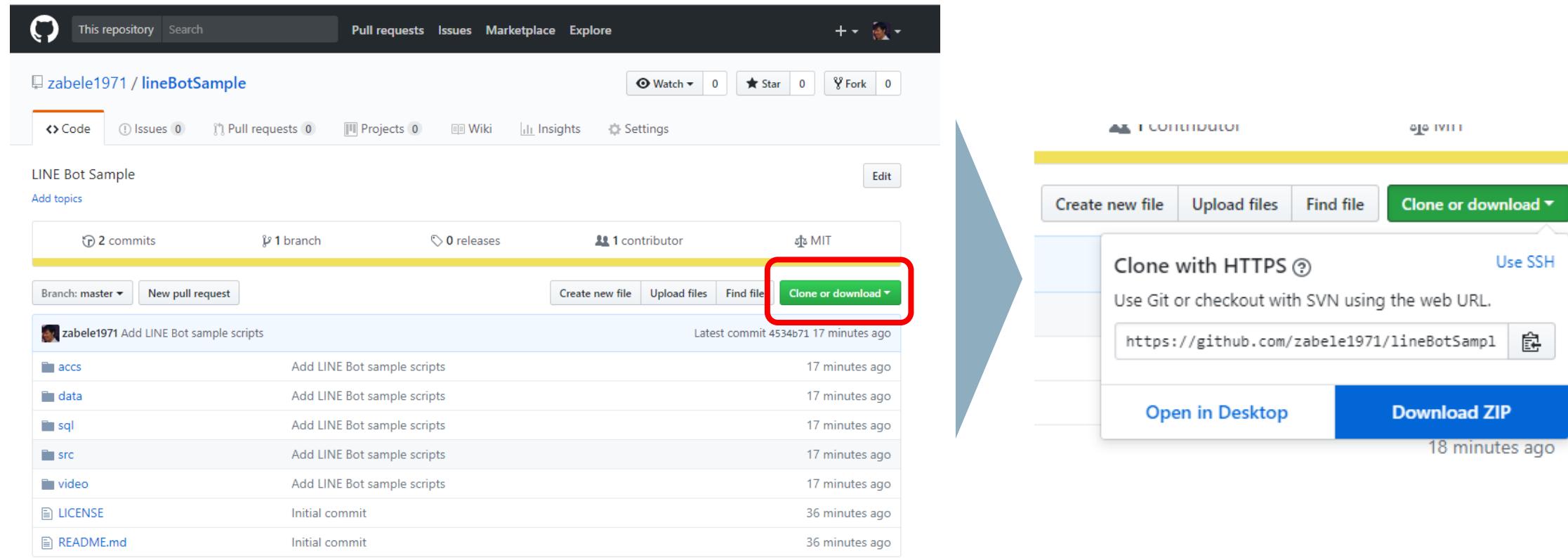
サンプル・アプリケーションのローカル環境での実行

4-4

サンプル・アプリケーションのクラウドへのデプロイ

GitHubからダウンロード

- <https://github.com/zabele1971/lineBotSample>



ディレクトリ構造



zabele1971 Add LINE Bot sample scripts



Application Container Cloudアップロード用ファイル



DBロード用サンプル・データ



DBユーザー作成／表作成SQL



Node.jsサンプル・プログラム



アプリの動画



LICENSE



README.md

/accs/

Application Container Cloudアップロード用ファイル

Branch: master ▾ lineBotSample / accs /

zabele1971 :q

..

botSample.zip :q Add LINE Bot sample scripts

line.config.json Add LINE Bot sample scripts

manifest.json Add LINE Bot sample scripts

readme.txt ^-d LINE Bot sample scripts

readme.txt

Oracle Application Container Cloudに対して
botSample.zipをアップロードする前に
以下を行ってください

(1) line.config.jsonに適切な値をセットする
(2) line.config.jsonをbotSample.zipに含める

botSample.zip
Application Container Cloudにアップロード

これには line.config.jsonが含まれていない。
自分の LINE Messaging APIのチャネル設定をして、
botSample.zipに含めてからアップロード

Manifest.jsonは、zipアップロード時に一緒に指定する

/data/ DBにロードするサンプル・データ

Branch: master ▾

[lineBotSample / data /](#)

[Create new file](#)

[Upload files](#)

[Find file](#)

[History](#)



[zabele1971](#) Add LINE Bot sample scripts

Latest commit 4534b71 2 days ago

..

[delivery_status.csv](#)

Add LINE Bot sample scripts

2 days ago

/sql/

Oracle DBにBot用ユーザー作成とテーブル作成するスクリプト

Branch: master ▾ [lineBotSample](#) / [sql](#) /

Create new file Upload files Find file History

 zabele1971 Add LINE Bot sample scripts Latest commit 4534b71 2 days ago

..

[create_table.sql](#)

[create_user.sql](#)

Add LINE Bot sample scripts

Oracle DBに Botで使用するテーブルを作成するスクリプト

Oracle DBに PDBの共通ユーザーとして c##botを作成するスクリプト

/src/

サンプル・アプリのソースコード (説明は後述)

Branch: master ▾ lineBotSample / src / Create new file Upload files Find file History

zabele1971 :q Latest commit 0a4e18c 21 hours ago

..

context.js	Add LINE Bot sample scripts	2 days ago
control.js	:q	21 hours ago
line.config.json	Add LINE Bot sample scripts	2 days ago
line.js	Add LINE Bot sample scripts	2 days ago
line_message.js	Add LINE Bot sample scripts	2 days ago
log4js.config.json	Add LINE Bot sample scripts	2 days ago
oracle.js	Add LINE Bot sample scripts	2 days ago
question.js	Add LINE Bot sample scripts	2 days ago
readme.txt	Add LINE Bot sample scripts	2 days ago
server.js	Add LINE Bot sample scripts	2 days ago

readme.txt

(1)以下はプラットフォーム共通で実行します。

```
npm init
npm install --save restify
npm install --save restify-errors
npm install --save log4js
npm install --save jsonfile
npm install --save @line/bot-sdk
```

/video/

Branch: master ▾ [lineBotSample](#) / video /

Create new file Upload files Find file History

 zabele1971 Add LINE Bot sample scripts	Latest commit 4534b71 2 days ago	
..		
 botSample.mp4	Add LINE Bot sample scripts	2 days ago

LINE上での動きを記録した動画。
デモなどにご利用ください。

4 サンプル・アプリケーションを使ってみよう

4-1

サンプル・アプリケーションのダウンロード

4-2

サンプル・アプリケーションのプログラム構造

4-3

サンプル・アプリケーションのローカル環境での実行

4-4

サンプル・アプリケーションのクラウドへのデプロイ

サンプル・アプリのソース(/src/)

- server.js
 - manifest.json内に「node server.js」と指定
 - /webhookに対するPOSTリクエストを処理
- control.js
 - アプリの全体のフロー制御フレームワーク
- context.js
 - ユーザーごとのステート情報を保持
 - Application Cacheへのアクセス
- question.js
 - ユーザーに入力を促す質問やメッセージの順番、
 - ユーザーによる入力に対する処理
- line.js
 - ユーザーの入力に対するBotからLINEへのメッセージ返信
- line_message.js
 - LINEのメッセージ形式の生成
- oracle.js
 - Oracle DBへのアクセス

赤: 実装内容に応じてカスタマイズ必要
オレンジ: 場合によってはカスタマイズ必要

- log4js.config.json
 - Log4js(ログ出力用パッケージ)の設定
- line.config.json
 - チャネル・シークレットやアクセストークンを指定

各ファイル共通(ログ出力)

```
const log4js = require('log4js');
log4js.configure('log4js.config.json');
const logger = log4js.getLogger("ソースコード名");
```

```
logger.error(エラーメッセージ);
logger.trace(トレース用メッセージ);
等
```

Application Container Cloudでは、コンソール出力が自動的にログとして保存されるので、出力設定は console にしている (log4js.config.json)

server.js

LINEから/webhookへのPOSTリクエストを受け取り、control.jsを呼び出す

REST API(Restifyパッケージ関連)

- let restify = require('restify'); // Restifyパッケージ呼び出し
- let errs = require('restify-errors'); // Restifyのエラーディファインメント呼び出し
- let server = restify.createServer(); // Restifyサーバーの生成
- server.post('/webhook',); // webhookへのPOSTに対する処理。
// control.jsを呼び出す。
- server.on('restifyError',); // エラーに対する処理
- server.listen(process.env.PORT,) // サーバーを起動してリスニング

control.js

アプリケーションのフロー制御

```
doAction: // server.jsから呼び出される
  → getContext(event)          // LINEからのイベントメッセージのユーザーIDをキーに
                                // ステート(Context)を取得
  → interpretUserAnswer();     // Contextとユーザーの入力値をもとに新ステータスを判断
      → validateSelectedNumber() // 数値の場合に、適切な数値かをチェック
  → Question.get()            // 新ステータスをもとに、該当するQuestionを取得
  → createReplyMessage();     // QuestionとContextをもとに、ユーザーに返信するメッセージを作成
  → line.replyMessageToLine(); // 作成したメッセージを送信
  → postAction(ctx);          // 事後処理
```

control.jsは、あまりカスタマイズをしなくとも動くように設計

context.js

アプリがステートフルに動作するためのコンテキスト情報の定義と保持

Contextクラスのデータ構造

- userid // LINEのユーザID。ユーザーIDをキーにして Contextを識別
- msgType // LINEのメッセージタイプ:
- status // ステータス。ステータス自体は question.jsで定義されている
- question // Questionオブジェクト
- data // アプリ固有情報
 - orderid // 注文番号
 - date // 再配達日
 - time // 再配達時間帯

コンテキスト情報のライフサイクル

- record() // Contextを保存
- remove(userid) // useridをキーに、保存された Contextを削除
- get(userid) // useridをキーに、保存された Contextを取得。該当なしの場合、生成

question.js

ステータスごとの処理内容や、ユーザーに対するメッセージなどを定義

questionArray

- id: // ステータスがID
- text: // 表示文字列
- Type: // LINE表示用タイプ buttons, text, confirm, sticker
- class: // 該当するQuestionクラス名(もしくはその子クラス名)
- next: // 次に呼び出すステータス
 - success: // 処理成功時に呼び出すステータス
 - failure: // 処理失敗時に呼び出すステータス
- previous: // やり直しで戻るときの戻り先ステータス
- answerCandidates: // 選択肢
 - text: // 選択肢の表示文字列
 - next: // buttonsやconfirmの場合は、選択したボタンに対する次のステータス
 - success: // 処理成功時に呼び出すステータス
 - failure: // 処理失敗時に呼び出すステータス



Questionクラス

- setNewStatusToContext
// ユーザー入力値をもとに、カレントステータスを設定する
// ために、control.jsのinterpretUserAnswerから呼び出される
- setQuestionMessage()
// LINE上でユーザーに返信するためのメッセージの作成

※Questionの派生クラスで、この二つの処理をカスタマイズする

ステータスをもとに Questionオブジェクトの生成

- Get() // control.jsの doAction内から呼び出される

line_message.js

Questionオブジェクトのtypeをもとに、LINEメッセージを作成

```
exports.getMessageObject = function(question) {  
  
  switch(question.type) {  
    case "text":  
      return text(question);  
    case "buttons":  
      return buttons(question);  
    case "confirm":  
      return confirm(question);  
    case "sticker":  
      return sticker(question);  
  }  
}
```

- text()
// Questionオブジェクトをもとにtextメッセージを生成
// 選択肢がある場合、それを文字列で表示
- buttons()
// Questionオブジェクトをもとにbuttonsメッセージを生成
// 選択肢がある場合、それをボタンにする
- confirm()
// Yes/Noボタンメッセージを生成
- sticker()
// 絵文字メッセージを生成
// サンプルアプリではENDステータスが sticker

line.js

LINEプラットフォームにメッセージを送信

```
let LINE_CONFIG = jsonfile.readFileSync('line.config.json');
  // Line.config.jsonを読み込み、アクセストークンとチャネル・シークレットをセット
```

```
replyMessageToLine(replyToken, message)
  // LINEに返信
```

```
handleLineError(err)
  // LINEによるエラーのログ出力
```

oracle.js

Oracle DBの表に対するSQLアクセス

- executeSQL(sqltext, bind, option) // 汎用的なSQL実行用プログラム
→doRelease() // コネクションをプールに戻す

4 サンプル・アプリケーションを使ってみよう

- 4-1 → サンプル・アプリケーションのダウンロード
- 4-2 → サンプル・アプリケーションのプログラム構造
- 4-3 → サンプル・アプリケーションのローカル環境での実行
- 4-4 → サンプル・アプリケーションのクラウドへのデプロイ

4 サンプル・アプリケーションを使ってみよう

4-3

サンプル・アプリケーションのローカル環境での実行

4-3-1

LINEのMessaging APIを始める

4-3-2

ローカルのOracle Databaseにスキーマを作成する

4-3-3

Node.js環境を作成する

4-3-4

LINEとローカルPC間をトンネリングする

4-3-5

サンプル・アプリケーションを実行する

4 サンプル・アプリケーションを使ってみよう

4-3

サンプル・アプリケーションのローカル環境での実行

4-3-1

LINEのMessaging APIを始める

4-3-2

ローカルのOracle Databaseにスキーマを作成する

4-3-3

Node.js環境を作成する

4-3-4

LINEとローカルPC間をトンネリングする

4-3-5

サンプル・アプリケーションを実行する

LINEのMessaging APIを始める

<https://developers.line.me/ja/index.html> にアクセス

The screenshot shows the main page of the LINE Developers website. The top navigation bar includes links for Services, Documentation, News, FAQ, Blog, and Login. On the left, a sidebar menu lists Home, Services, LINE Login, Messaging API, LINE Partners, Documentation, News, FAQ, and two buttons for starting with LINE Login and Messaging API. The central area features a large green gradient background with the text "GROW YOUR BUSINESS". Below this, there are two prominent green buttons: "LINEログイン をはじめる" and "Messaging API (ボット) をはじめる". The second button is highlighted with a red rectangular box. At the bottom, there are links for "LINEログインスタートガイド" and "Messaging API スタートガイド".

LINEのMessaging APIを始める



すでにLINE登録している
ユーザーのアドレスとパ
スワードでログイン

LINEのMessaging APIを始める

The screenshot shows the LINE Developers Console login page. At the top, there is a dark header bar with the "LINE developers" logo on the left and navigation links like "サービス", "ドキュメント", "ニュース", "FAQ", "ブログ", and "ログアウト" on the right. Below the header, a large banner displays the message "LINE Developers コンソールへようこそ！" and "開発者情報 登録完了". In the center, there is a circular profile picture of a person named "Zabele". Below the profile picture, the name "Zabele" is displayed. A message states "開発者情報が登録されました。LINE Developersコンソールでは、LINEが提供するサービスを利用するためのchannelの作成から管理まで行うことができます。" At the bottom, there is a green button labeled "はじめる".

LINEのMessaging APIを始める プロバイダを作成する

The screenshot shows the LINE Developers Console interface. On the left, there's a sidebar with navigation links: ホーム, zab_dev, Zabele, 新規プロバイダー作成 (highlighted in green), and プロバイダー一覧. The main area has a dark header with the text "LINE Developers コンソールへようこそ！". Below it, there's a message: "あなたの開発技術とLINE Platformを利用して人と人をつなぐアプリを開発するためには、まずはプロバイダーを作成しましょう！" followed by "プロバイダーとはサービス提供者（企業・個人）です。". A large green button labeled "プロバイダーを作成する" is centered. A blue arrow points from this button to a modal window titled "新規プロバイダー作成". This modal has a sub-header "プロバイダー名を入力してください" and a note "プロバイダーとはサービス提供者（企業・個人）の名前です。". It contains a text input field with the placeholder "BotTest4OraCloud" and a character limit note "最大100文字". At the bottom of the modal is a green "確認" button. To the right of the modal, there's a small image of a smartphone displaying a messaging interface with the text "BotTest4OraCloud".

LINEのMessaging APIを始める プロバイダを作成する

The screenshot shows the LINE Developers portal interface. On the left, there is a sidebar with the following items:

- ホーム
- zab_dev
-  Zabele
- [新規プロバイダー作成](#)
- プロバイダー一覧
- BotTest4OraCloud (0)

The main content area displays the following information:

プロバイダー
BotTest4OraCloud 

新規プロバイダーBotTest4OraCloudが作成されました。 

 LINEログイン  Messaging API

まだchannelがありません。

A red box highlights the "Messaging API" button.

LINEのMessaging APIを始める

新規チャネルを作成する

The screenshot shows the LINE developers portal interface. The top navigation bar includes links for サービス, ドキュメント, ニュース, FAQ, ブログ, and ログアウト. On the left, a sidebar shows a user profile (Zabele) and navigation links for ホーム, zab_dev, and 新規プロバイダー作成 (which is highlighted in green). The main content area is titled "新規channel作成" and displays a three-step process: "プロバイダー選択" (Step 1), "Channel情報入力" (Step 2, currently active), and "確認" (Step 3). Step 2 is titled "STEP2 Messaging APIの情報を入力してください". It features a placeholder image for an application icon labeled "アプリアイコン画像" and a preview of a mobile phone screen showing a bot named "OraAPEXTest" from "BotTest4OraCloud".



STEP2 Messaging APIの情報を入力してください

アプリアイコン画像



3MB以内、JPEG/PNG/GIF/BMP形式

アプリ名

BotTest

最大20文字

アプリ説明

クラウドを使ったBot開発のTest用モジュール

最大500文字

プラン

Developer Trial

MessagingAPIを利用したBotを試すプランです。友だちとメッセージの送受信を行うことができます。
※追加可能友だち数は50人に制限されています。また、Developer Trialからプランの切り替えやプレミアムIDの購入はできません。

フリー

MessagingAPIを利用したBotを開発するプランです。友だちの人数に制限はありませんが、Push messagesを利用してBotから友だちにメッセージを送信することはできません。

※サービス拡張に向けプラン変更が可能です。



※表示イメージです。



サンプル・アプリは
フリーの範囲で動作

Channel

BotTest

Channel基本設定

Channelの基本情報（アイコン画像、アプリ名等）設定や必要なChannel情報の確認などを行います。
テスター権限の場合は「テスターをやめる」機能のみご利用可能です。

基本情報

アプリアイコン画像



3MB以内、JPEG/PNG/GIF/BMP形式

アプリ名

BotTest

アプリ説明

クラウドを使ったBot開発のTest用モジュール

Channel ID

[REDACTED]

再発行

Channel Secret

[REDACTED]



※表示イメージです。



チャネル・シークレットとアクセストークンはアプリを実行するときに使います

アクセストークンを発行する

LINEのMessaging APIを始める

その他のLINE関連の設定

The screenshot shows the 'Other LINE-related settings' section of the LINE@ Manager. It includes:

- Webhook送信**: Set to '利用しない' (Not in use). A red callout bubble points to this setting with the text: 'Webhook関連は後で設定する' (Webhook-related settings will be configured later).
- Webhook URL**: Set to '-' (SSL only support). A red callout bubble points to this setting with the text: '自動応答メッセージは「利用しない」にしたほうがよい(ただし、アプリによる)' (Automatic response message is better set to 'Not in use' (unless specified by the app)).
- Botのグループトーク参加**: Set to '利用しない'.
- LINE@機能の利用**: Includes:
 - 自動応答メッセージ**: Set to '利用しない'.
 - 友だち追加時あいさつ**: Set to '利用する' (Use). An orange callout bubble points to this setting with the text: '友だち追加時あいさつは利用する場合はメッセージを指定しておくこと' (If you use friend addition greetings, specify the message in advance).
- LINEアプリへのQRコード**: Shows a QR code for app verification.

4 サンプル・アプリケーションを使ってみよう

4-3

サンプル・アプリケーションのローカル環境での実行

4-3-1

LINEのMessaging APIを始める

4-3-2

ローカルのOracle Databaseにスキーマを作成する

4-3-3

Node.js環境を作成する

4-3-4

LINEとローカルPC間をトンネリングする

4-3-5

サンプル・アプリケーションを実行する

ローカルのOracle Databaseにスキーマを作成する

- 以下から Oracle Databaseをダウンロード
 - <http://www.oracle.com/technetwork/jp/database/enterprise-edition/downloads/index.html>
- 本サンプル・アプリケーションは Oracle Database Standard Editionの範囲で利用可能です

ローカルのOracle Databaseにスキーマを作成する

SQL Developerのダウンロード

- 以下のURLから SQL Developerをダウンロードします
- <http://www.oracle.com/technetwork/jp/developer-tools/sql-developer/downloads/index.html>
- 2018年1月15日時点のバージョンは17.4.0.355.2349ですが、本資料はバージョン17.2.0.188で作成しています

ローカルのOracle Databaseにスキーマを作成する

SQL Developerの起動



Sqldeveloper.exeを起動
(Windowsの場合)

ローカルのOracle Databaseにスキーマを作成する

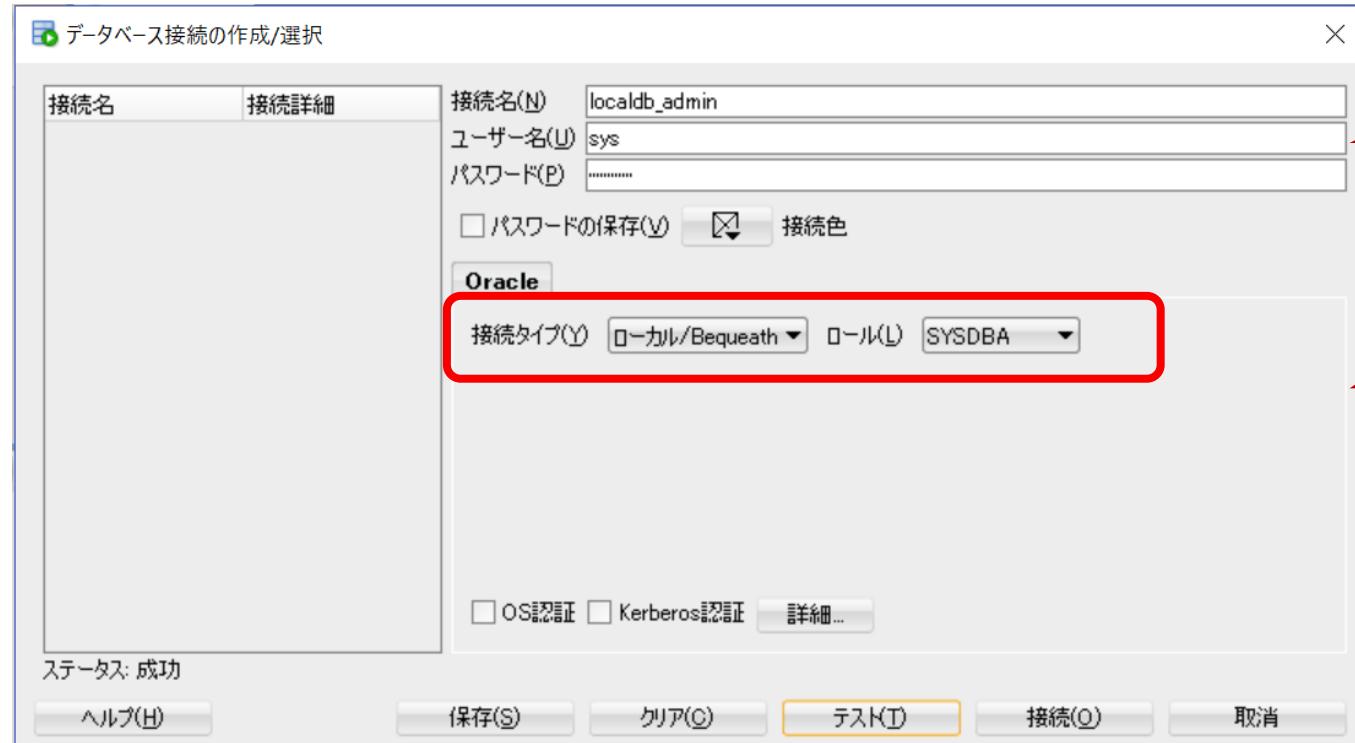
SQL Developerの起動



ローカルのOracle Databaseにスキーマを作成する接続の作成



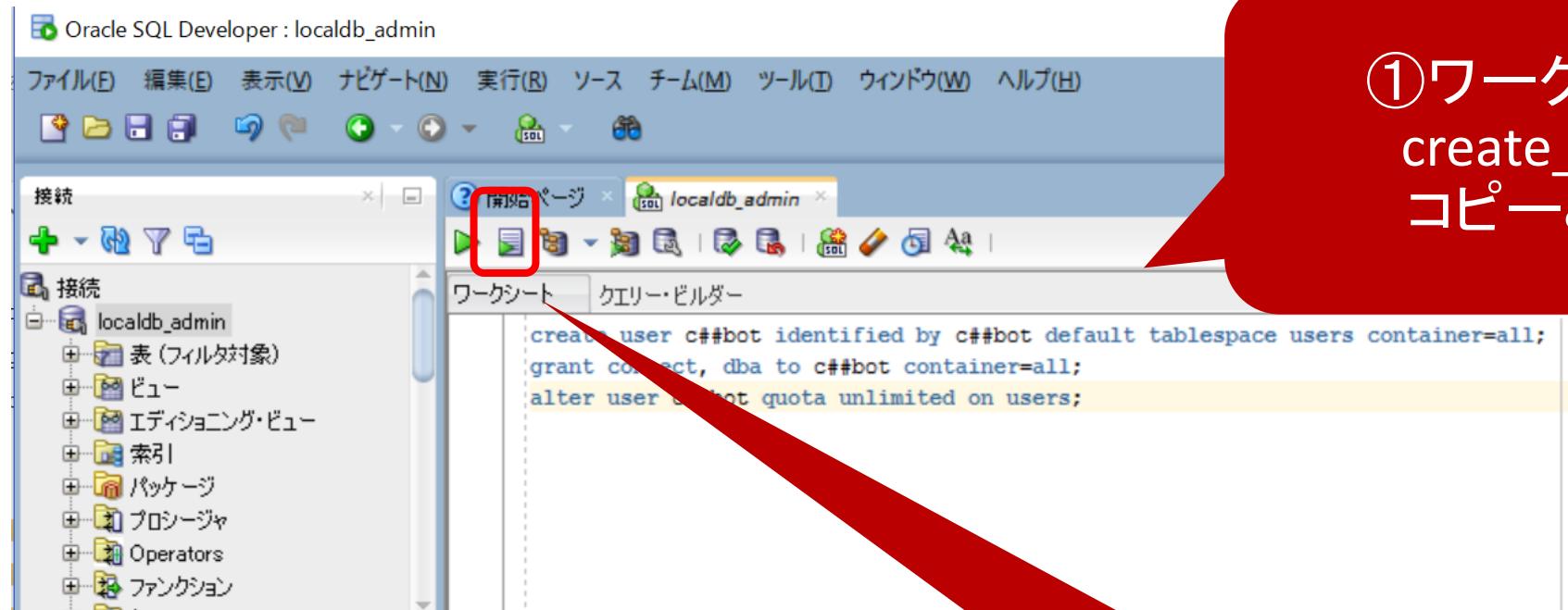
ローカルのOracle Databaseにスキーマを作成する SQL Developerから管理者でアクセス



SYSユーザーのパスワードを
指定

ローカルのDBの場合、
Bequeath接続。
ロールはSYSDBAにする。

ローカルのOracle Databaseにスキーマを作成する



①ワークシート上に
create_user.sqlを
コピー&ペースト

②「スクリプトの実行」
を押す

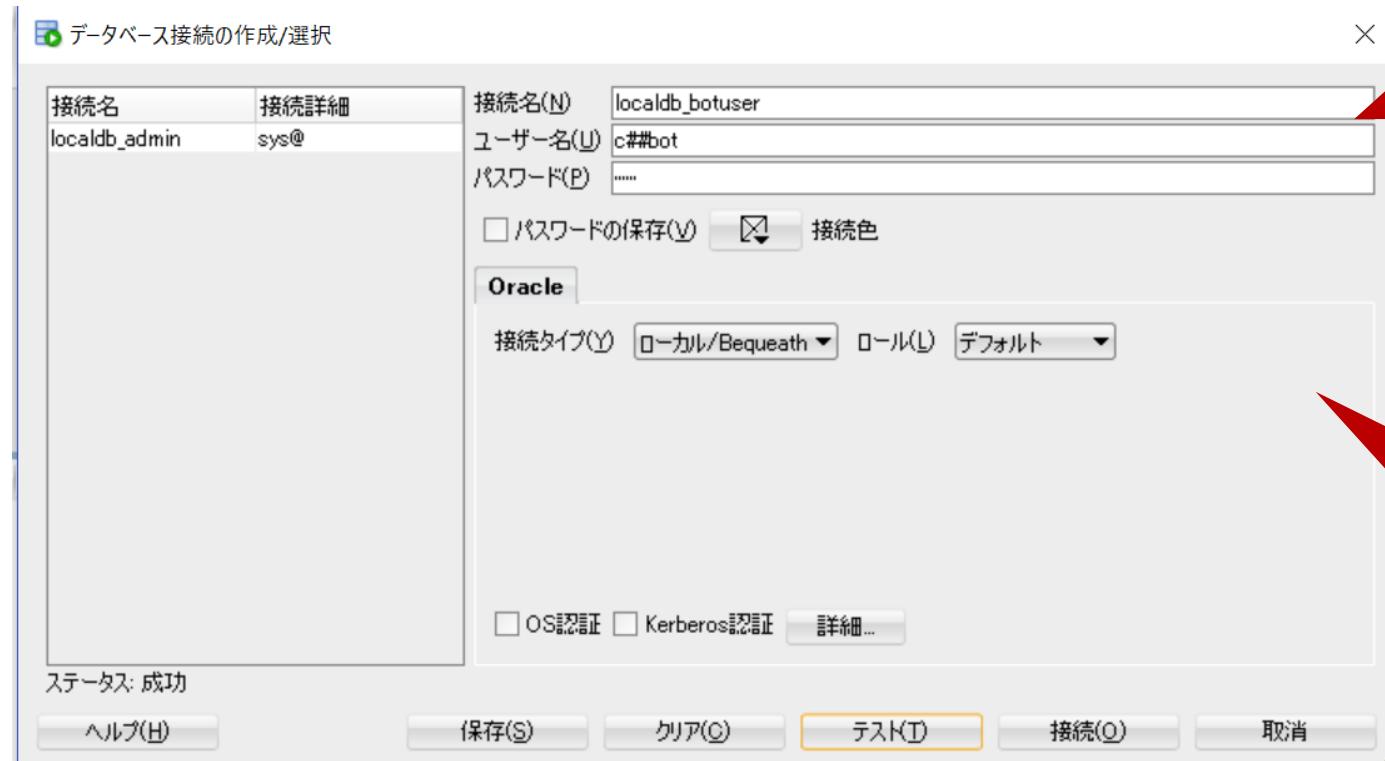
ローカルのOracle Databaseにスキーマを作成する

The screenshot shows the Oracle SQL Developer interface. On the left, the '接続' (Connection) sidebar has '接続の作成(A)...' selected. In the center, a worksheet contains the following SQL script:

```
create user c##bot identified by c##bot default tablespace users
grant connect, dba to c##bot container=all;
alter user c##bot quota unlimited on users;
```

This script creates a user 'c##bot' with password 'c##bot', sets the default tablespace to 'users', grants DBA privileges, and sets an unlimited quota on the 'users' tablespace. A dashed box highlights this code. On the right, the 'スクリプトの出力' (Script Output) window shows the message: 'User C##BOTは作成されました。' (User C##BOT was created).

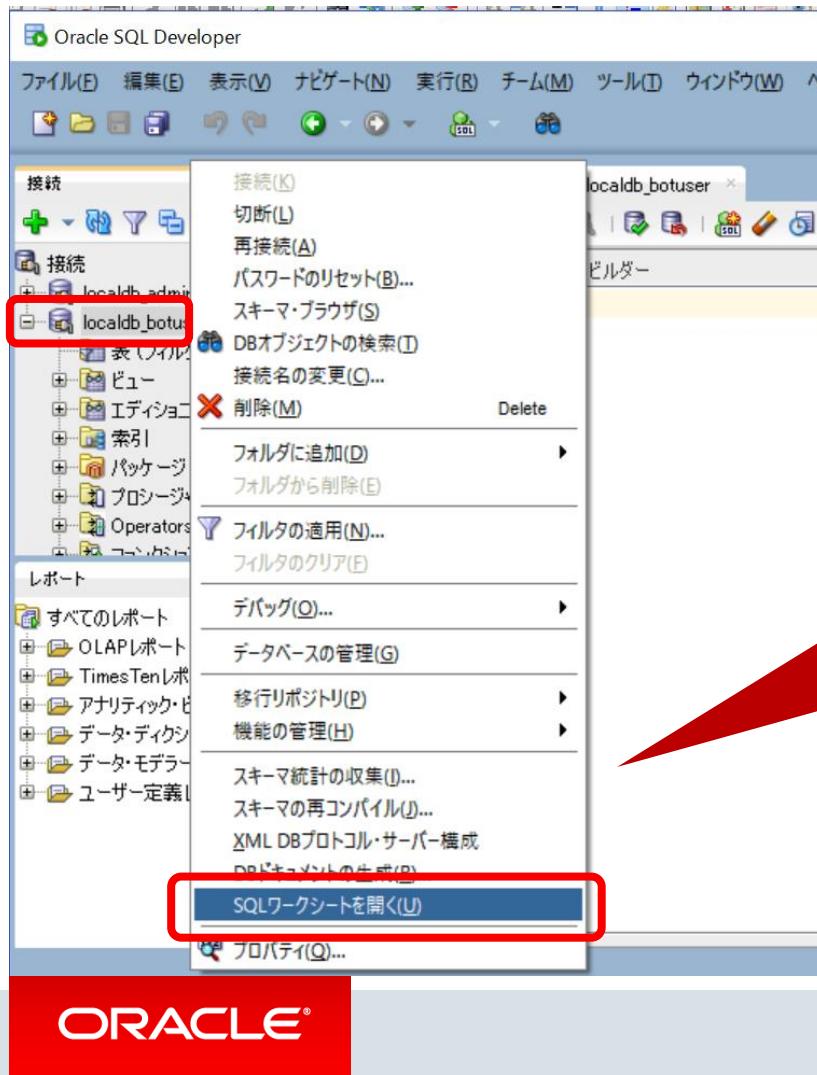
ローカルのOracle Databaseにスキーマを作成する SQL DeveloperからBot用ユーザーでアクセス



作成した c##bot ユーザー / パスワードを指定

ローカルなので
Bequeath 接続

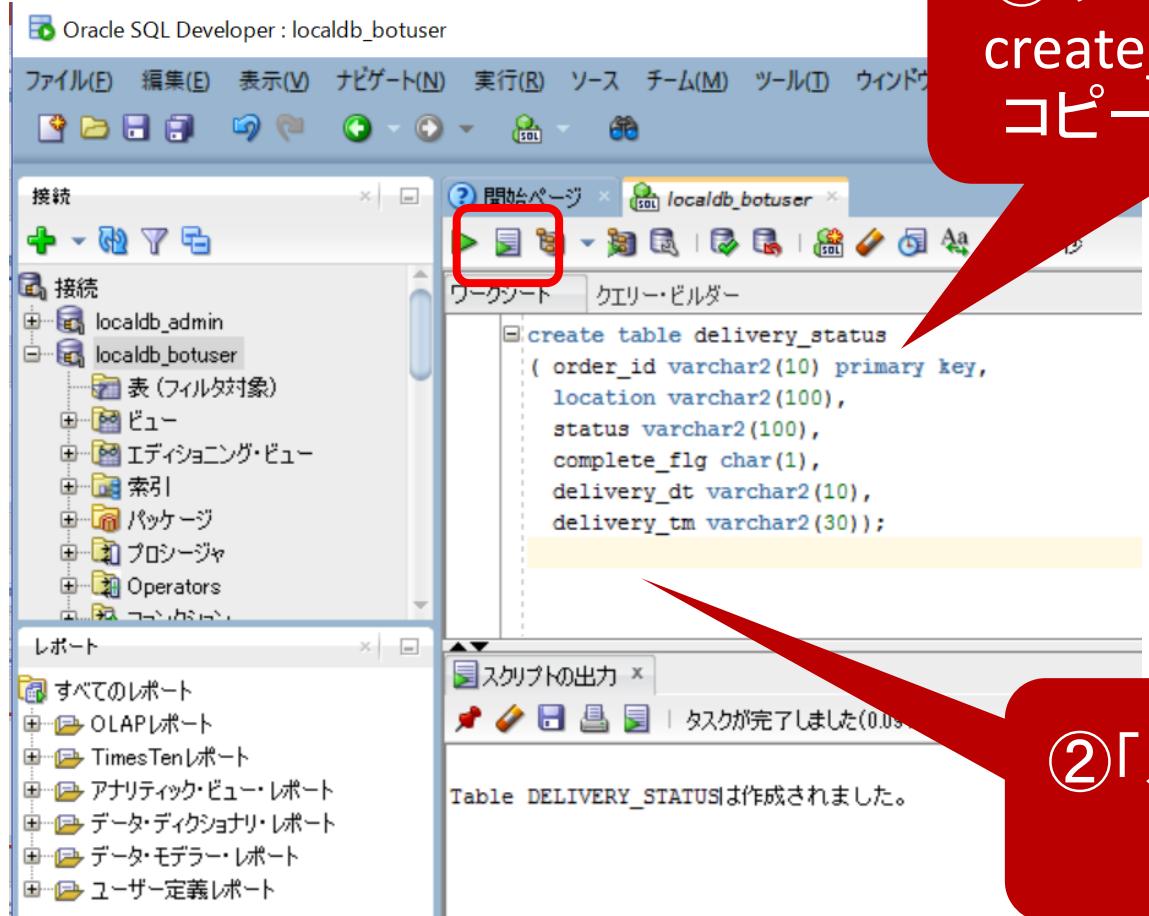
ローカルのOracle Databaseにスキーマを作成する



作成したlocaldb_botuser接続を右クリック。

リストが表示されるので
「SQLワークシートを開く」を選択。

ローカルのOracle Databaseにスキーマを作成する

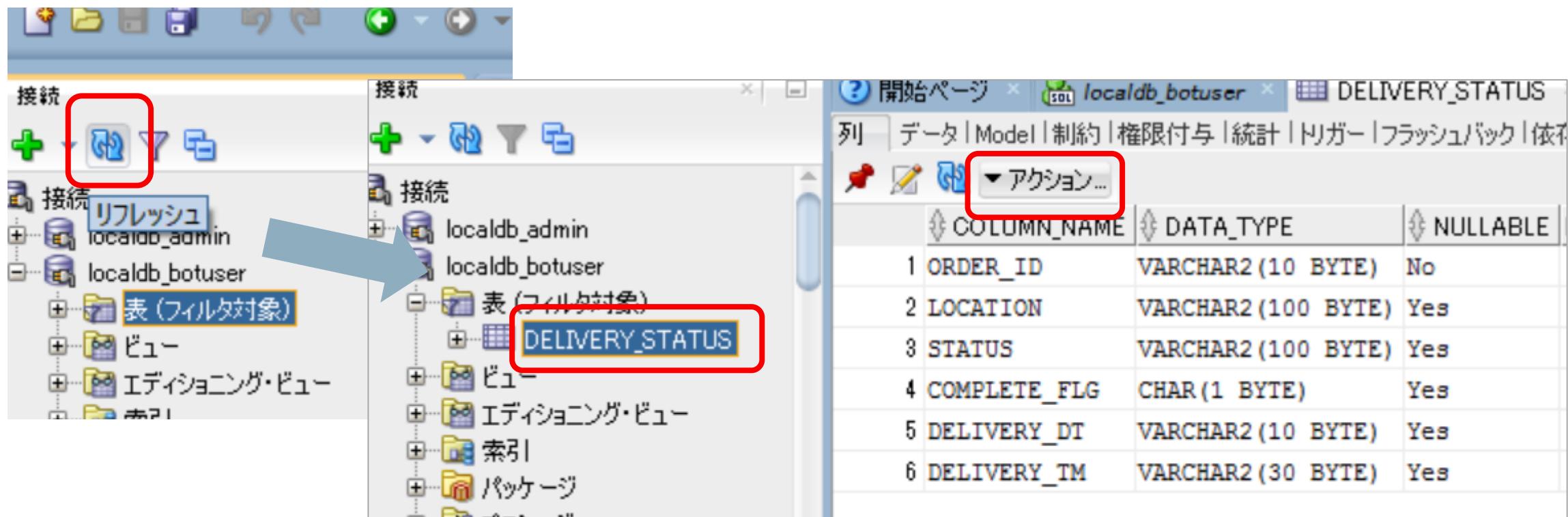


①ワークシートに
create_table.sqlを
コピー&ペースト

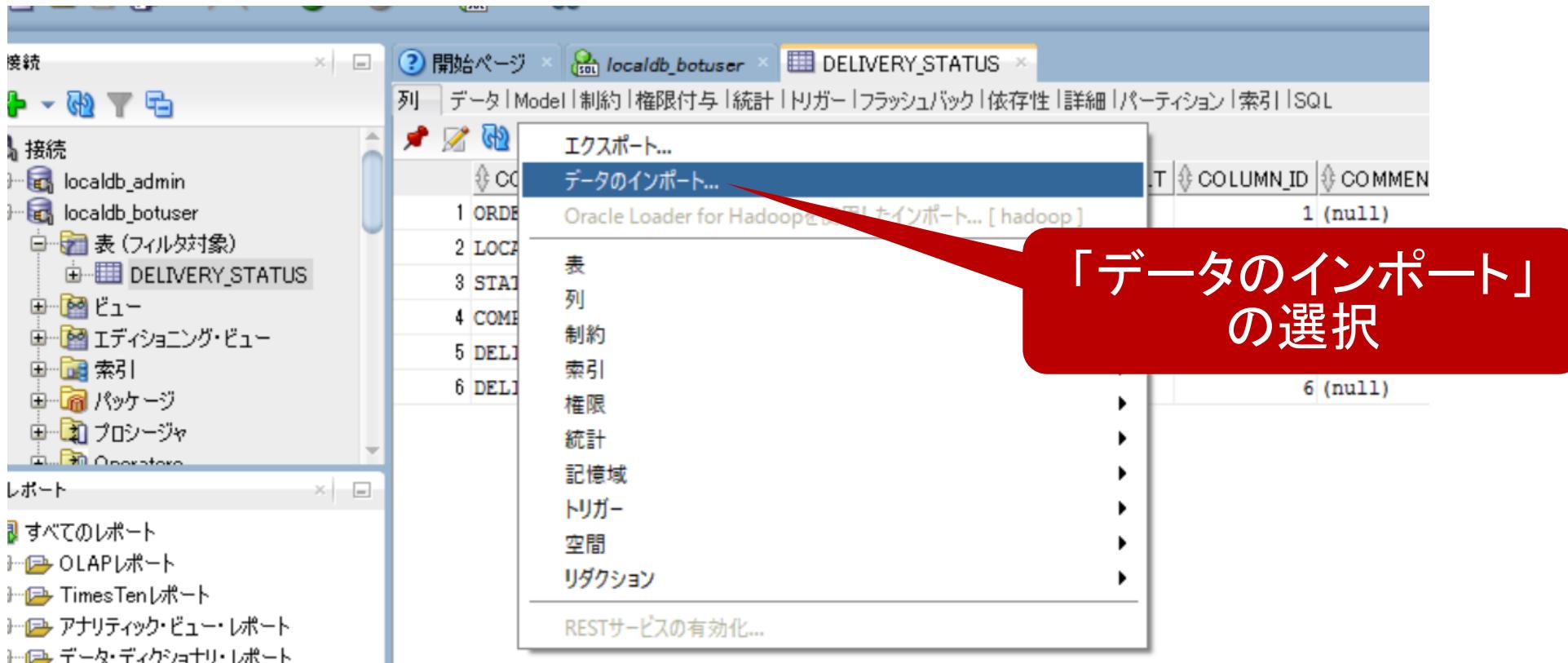
```
create table delivery_status
( order_id varchar2(10) primary key,
  location varchar2(100),
  status varchar2(100),
  complete_flg char(1),
  delivery_dt varchar2(10),
  delivery_tm varchar2(30));
```

②「スクリプトの実行」
を押す

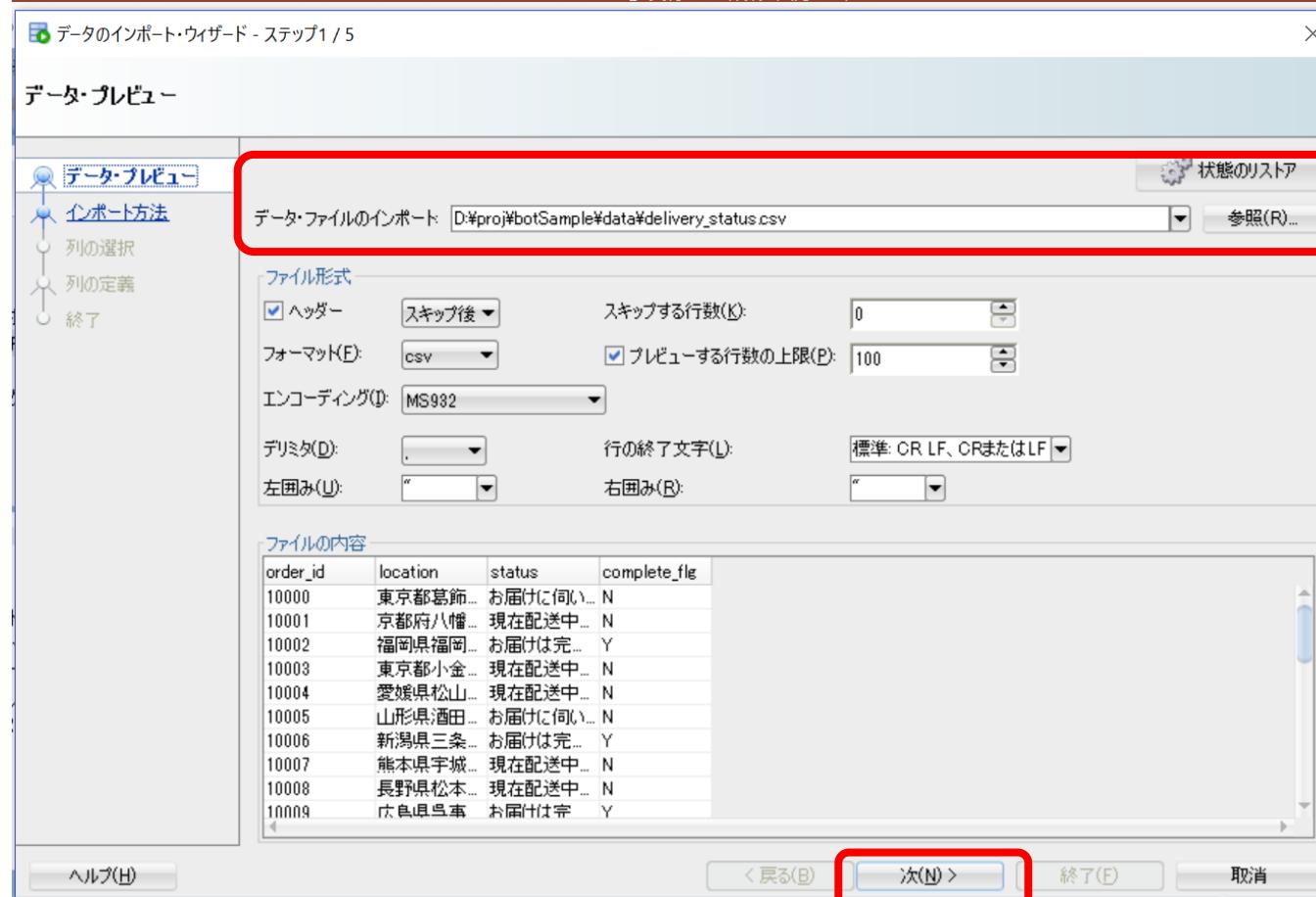
ローカルのOracle Databaseにスキーマを作成する SQL Developerからのサンプルデータの登録



ローカルのOracle Databaseにスキーマを作成する SQL Developerからのサンプルデータの登録

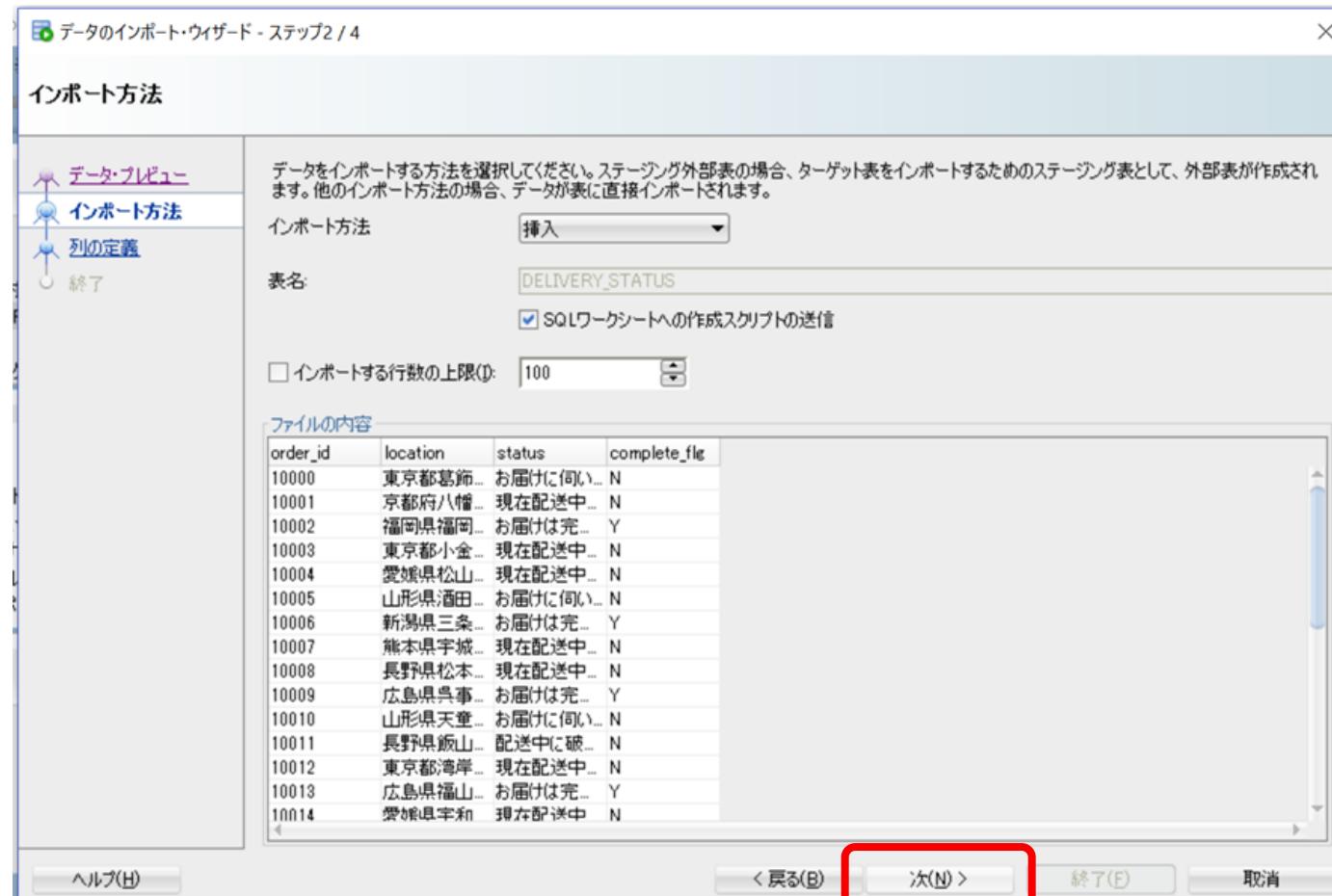


ローカルのOracle Databaseにスキーマを作成する SQL Developerからのサンプルデータの登録

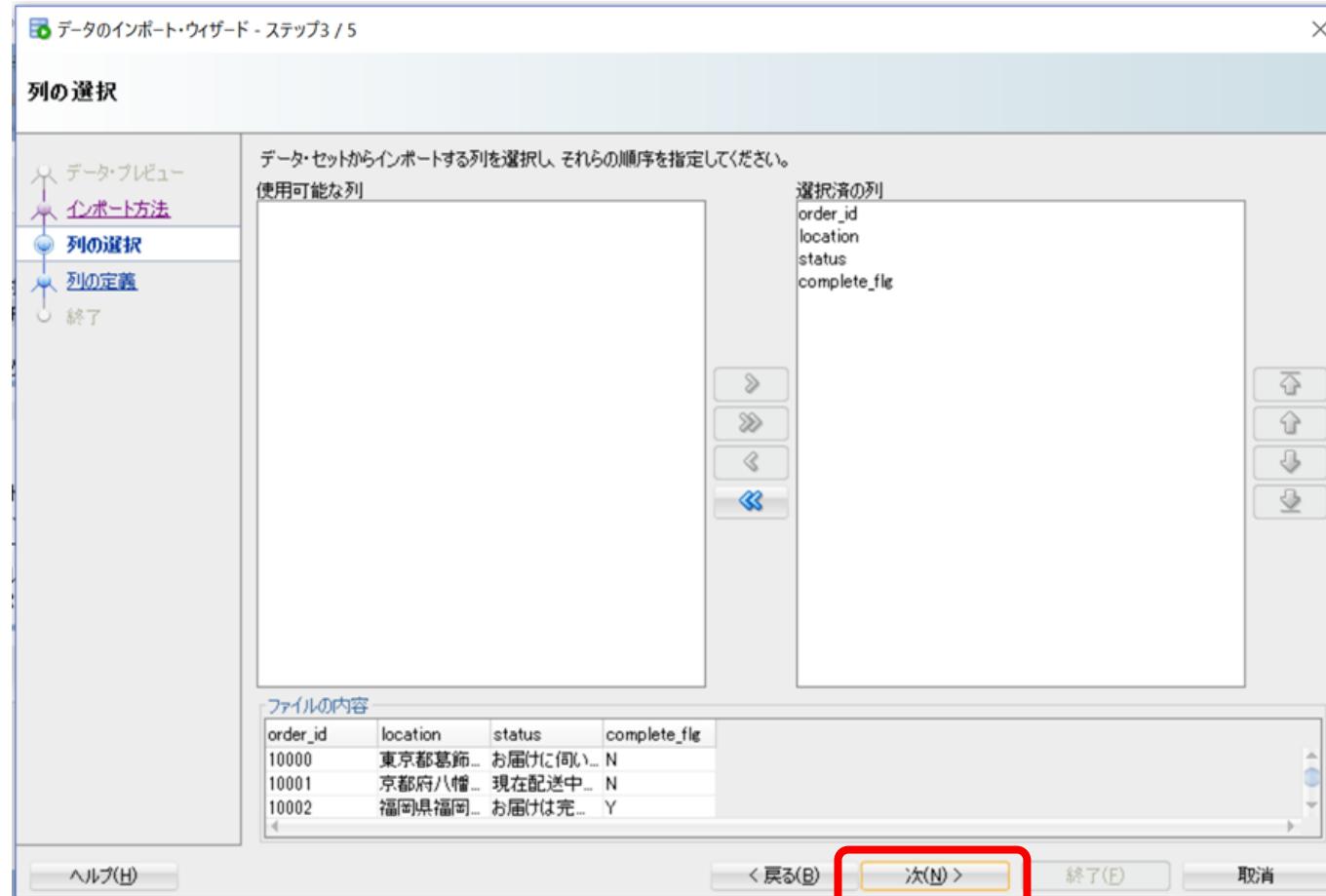


delivery_status.csvを設定

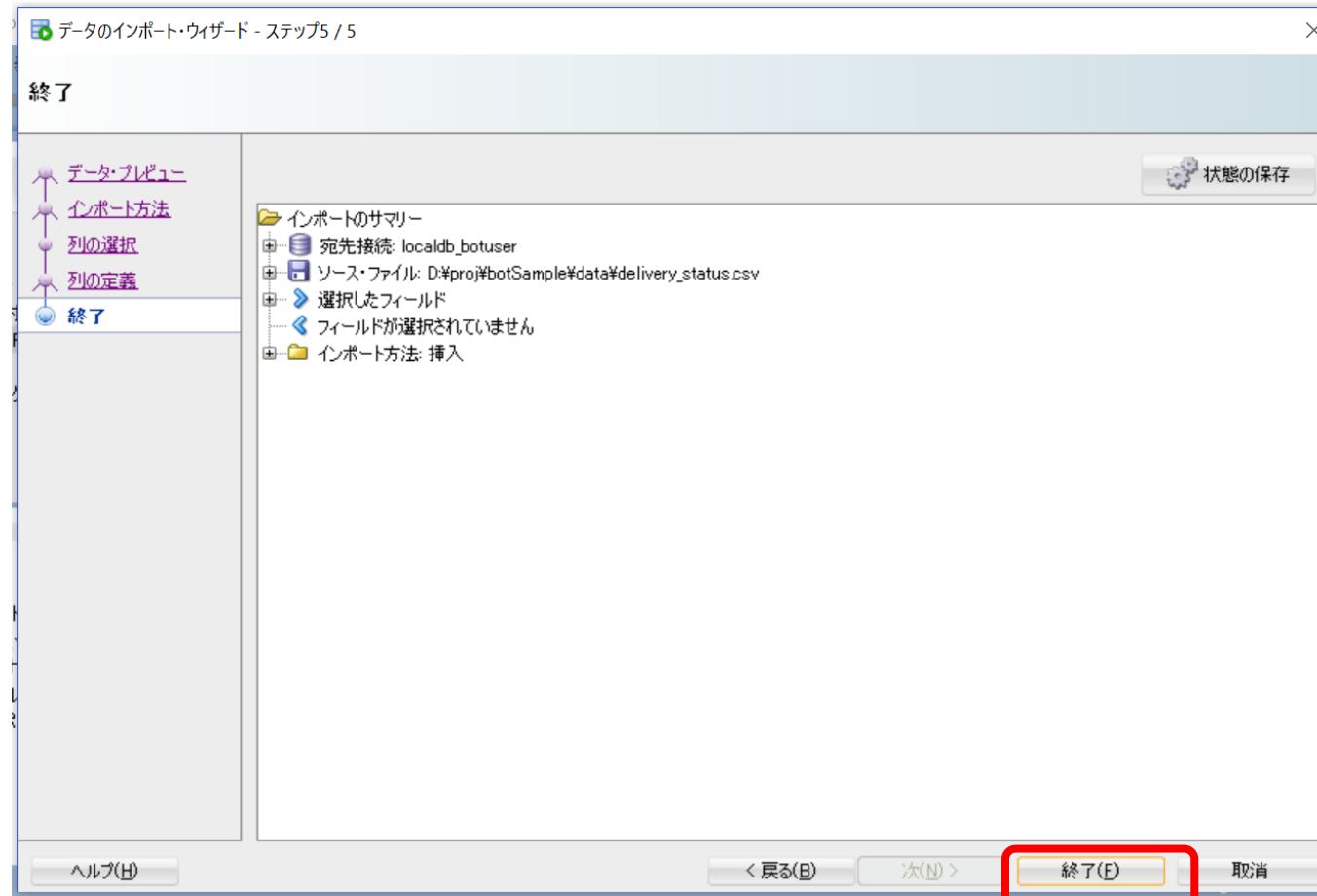
ローカルのOracle Databaseにスキーマを作成する SQL Developerからのサンプルデータの登録



ローカルのOracle Databaseにスキーマを作成する SQL Developerからのサンプルデータの登録

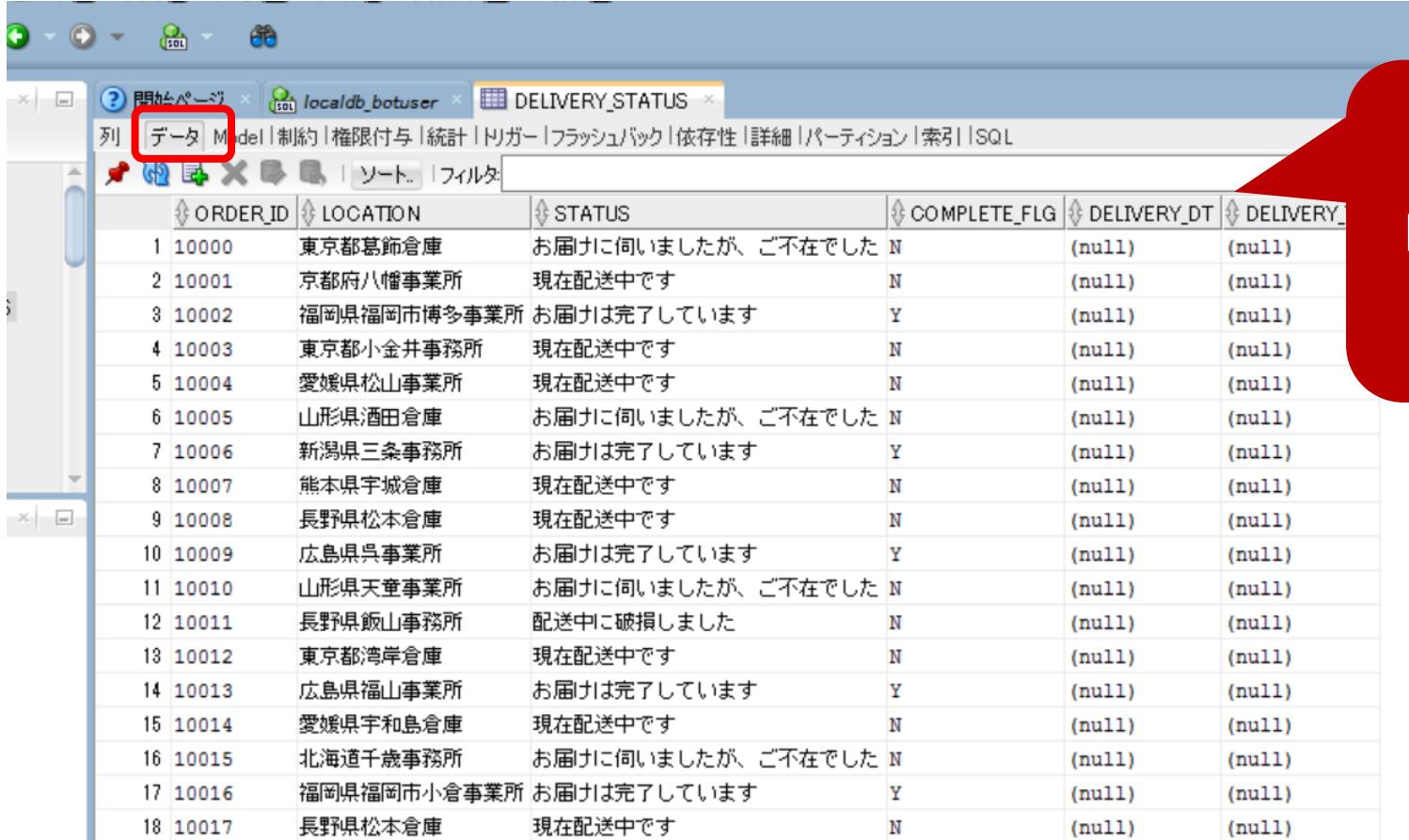


ローカルのOracle Databaseにスキーマを作成する SQL Developerからのサンプルデータの登録



口一カルのOracle Databaseにスキーマを作成する

SQL Developerからのサンプルデータの登録



The screenshot shows the SQL Developer interface with the 'DELIVERY_STATUS' table selected. The 'Data' tab is highlighted with a red box. The table contains 18 rows of sample data. The columns are ORDER_ID, LOCATION, STATUS, COMPLETE_FLG, DELIVERY_DT, and DELIVERY_BY.

ORDER_ID	LOCATION	STATUS	COMPLETE_FLG	DELIVERY_DT	DELIVERY_BY
1 10000	東京都葛飾倉庫	お届けに伺いましたが、ご不在でした	N	(null)	(null)
2 10001	京都府八幡事業所	現在配送中です	N	(null)	(null)
3 10002	福岡県福岡市博多事業所	お届けは完了しています	Y	(null)	(null)
4 10003	東京都小金井事務所	現在配送中です	N	(null)	(null)
5 10004	愛媛県松山事業所	現在配送中です	N	(null)	(null)
6 10005	山形県酒田倉庫	お届けに伺いましたが、ご不在でした	N	(null)	(null)
7 10006	新潟県三条事務所	お届けは完了しています	Y	(null)	(null)
8 10007	熊本県宇城倉庫	現在配送中です	N	(null)	(null)
9 10008	長野県松本倉庫	現在配送中です	N	(null)	(null)
10 10009	広島県呉事業所	お届けは完了しています	Y	(null)	(null)
11 10010	山形県天童事業所	お届けに伺いましたが、ご不在でした	N	(null)	(null)
12 10011	長野県飯山事務所	配送中に破損しました	N	(null)	(null)
13 10012	東京都湾岸倉庫	現在配送中です	N	(null)	(null)
14 10013	広島県福山事業所	お届けは完了しています	Y	(null)	(null)
15 10014	愛媛県宇和島倉庫	現在配送中です	N	(null)	(null)
16 10015	北海道千歳事務所	お届けに伺いましたが、ご不在でした	N	(null)	(null)
17 10016	福岡県福岡市小倉事業所	お届けは完了しています	Y	(null)	(null)
18 10017	長野県松本倉庫	現在配送中です	N	(null)	(null)

「データ」を選択すると
ロードされたデータが表
示されることを確認

4 サンプル・アプリケーションを使ってみよう

4-3

サンプル・アプリケーションのローカル環境での実行

4-3-1

LINEのMessaging APIを始める

4-3-2

ローカルのOracle Databaseにスキーマを作成する

4-3-3

Node.js環境を作成する

4-3-4

LINEとローカルPC間をトンネリングする

4-3-5

サンプル・アプリケーションを実行する

Node.js環境を作成する

Node.jsをインストール



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Spectre and Meltdown in the context of Node.js.

Download for Windows (x64)

8.9.4 LTS

Recommended For Most Users

9.4.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#) [Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the LTS schedule.

Sign up for [Node.js Everywhere](#), the official Node.js Weekly Newsletter.

Node.jsをインストール

本資料では 8.xを使用

LINUX FOUNDATION COLLABORATIVE PROJECTS

[Report Node.js issue](#) | [Report website issue](#) | [Get Help](#)

© 2018 Node.js Foundation. All Rights Reserved. Portions of this site originally © 2018 Joyent.

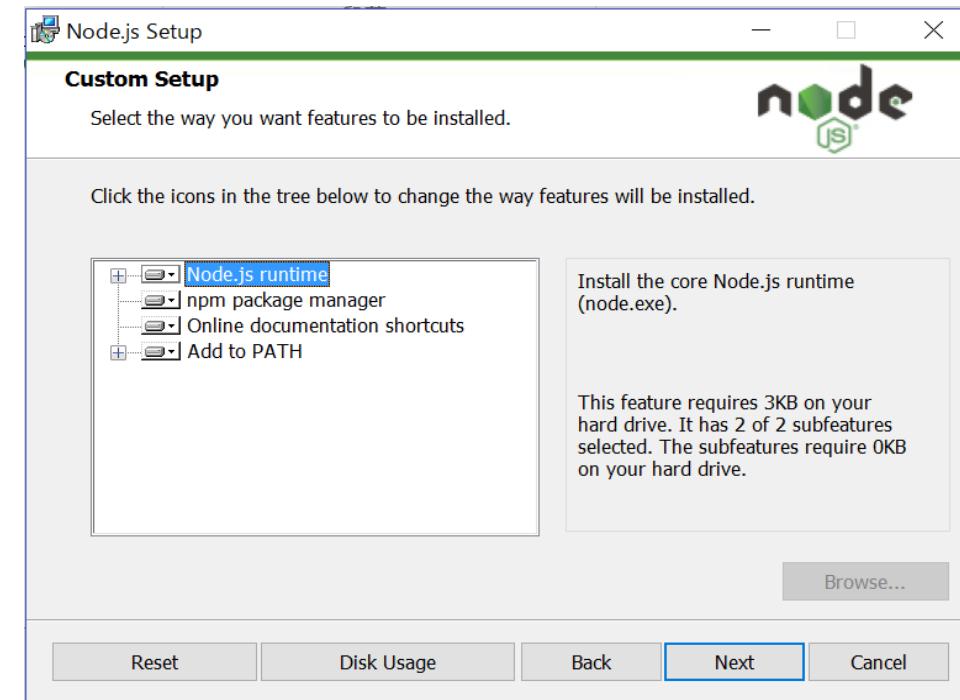
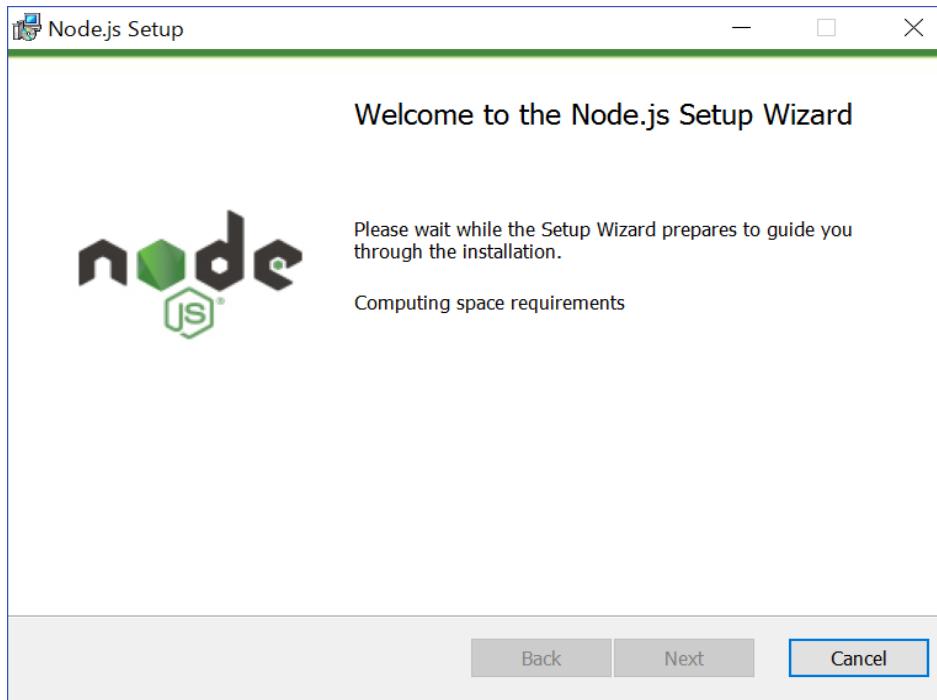
Node.js is a trademark of Joyent, Inc. and is used with its permission. Please review the Trademark Guidelines of the Node.js Foundation.

ORACLE®

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Node.js環境を作成する

Node.jsをインストール



Node.js実行環境と、パッケージ・マネージャであるnpmがインストールされる

Node.js環境を作成する

srcフォルダにあるソース・ファイルに対して Node.js環境を設定する

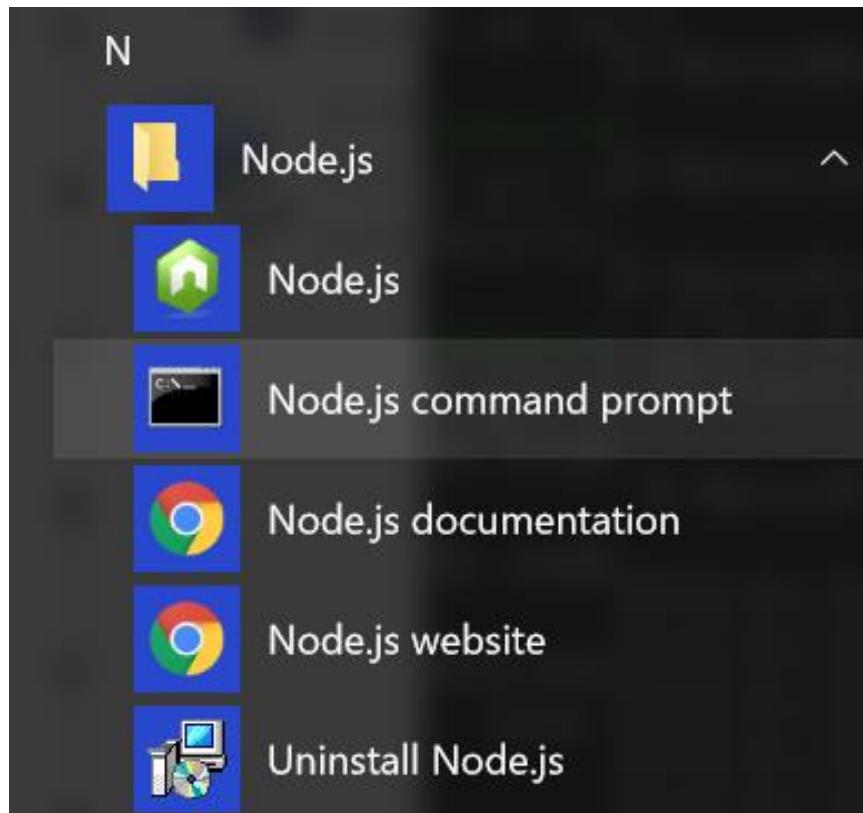
名前

-  context.js
-  control.js
-  line.js
-  line_message.js
-  oracle.js
-  question.js
-  server.js
-  line.config.json
-  log4js.config.json
-  readme.txt

line.config.jsonにアクセストークとチャネルシークレットを指定する

```
{  
  "channelAccessToken": "ここに割り当てられたアクセストークンを入れる",  
  "channelSecret": "ここに割り当てられたチャネル・シークレットを入れる"  
}
```

(Windowsの場合) Node.js command promptを実行する



Node.js環境を作成する

npm initを実行してプロジェクトの初期設定を行う

```
D:\botSample> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

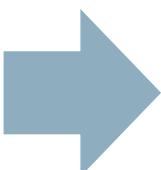
Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (botSample) botsample
version: (1.0.0)
description:
entry point: (context.js) server.js
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\botSample\package.json:

{
  "name": "botsample",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo ¥"Error: no test specified¥" && exit 1",
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC"
}

Is this ok? (yes)
```

Entry pointは
server.jsにする



package.jsonファイルが生成される

Node.js環境を作成する

npmパッケージのインストール(1) `npm install --save パッケージ名` を実行する

```
npm install --save restify
npm install --save restify-errors
npm install --save log4js
npm install --save jsonfile
npm install --save @line/bot-sdk
npm install --save accs-cache-handler
```

※ご利用のネットワーク環境のProxy設定などによっては、
そのままでは実行でない場合があります。

Node.js環境を作成する

npmパッケージのインストール(2) oracledbパッケージのインストール

以下はWindowsの場合

詳細は『Windows環境でのNode.jsのoracledbのインストール』を参照

<https://qiita.com/kaohas/items/e4fe9e741d62e3169224>

(1) PowerShellの管理モードでwindows-build-toolパッケージをインストール

```
npm install --global --production windows-build-tool
```

(2) Oracle Instant Clientをインストール

(3) Oracle Instant Clientのパスを設定

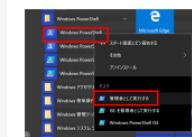
(例)

```
set PATH= D:\instantclient_12_2;%PATH%
set OCI_LIB_DIR=D:\instantclient_12_2\ sdk\lib\msvc
set OCI_INC_DIR=D:\instantclient_12_2\ sdk\include
```

(4) Oracledbパッケージをインストール

```
npm install --save oracledb
```

The screenshot shows a GitHub post with the following content:

- oracledbパッケージとは**
node-oracledbはオラクルが提供している、Node.jsでOracle DBにアクセスするためのnpmパッケージです。Windows上でこれをインストールするのに、若干の手間が必要となるので、メモ的に書いてみました。
- インストールの前提**
oracledbパッケージをインストールには以下が必要になります。
 1. Python 2.7
 2. C Compiler with support for C++ 11 (Xcode, gcc, Visual Studio or similar)
 3. Oracle 11.2, 12.1 or 12.2クライアント・ライブラリWindows環境ではwindows-build-toolsと呼ばれるNode.jsパッケージをインストールすると、Pythonも一緒にセットアップされるので、とても便利です。
- windows-build-toolsのインストール**
windows-build-toolsをインストールすると、WindowsにPythonとVisual Studio環境が組み込まれます。インストールするには、管理者モードで実行したPowerShell上で行う必要があります。

```
PowerShellから以下を実行します。  
npm install --global --production windows-build-tools
```
- Oracle Instant Clientのインストール**

Node.js環境を作成する oracledbへのアクセスのカスタマイズ oracle.jsの一部

```
let connectionProperties = {  
    user: process.env.DBAAS_USER_NAME ||  
    password: process.env.DBAAS_USER_PASSWORD ||  
    connectString: process.env.DBAAS_DEFAULT_CONNECT_DESCRIPTOR ||  
    stmtCacheSize: process.env.DBAAS_STATEMENT_CACHE_SIZE ||  
    poolMin: 1,  
    poolMax: 10  
};
```

Application Container
CloudとDBCSをバインドした
場合は、変数化された値
が参照される

```
"c##bot",  
"c##bot",  
"  
",  
10,
```

(OSに環境変数を設定してい
ない場合)
ローカルDBのアクセスには
こちらが参照される

ユーザー名、パスワードなど、
ローカルDBの設定に応じて、
カスタマイズすること

Node.js環境を作成する

サンプル・ソースコードを実行

node server.js を実行

```
D:\$proj\$botSample>node server.js
Internal Caching URL is not set. Falling back on using a local hashmap instead.
[2018-01-16T16:04:41.938] [INFO] server.js - Node is running...
```

※シンタックス・エラーがある場合は、起動できずにここでエラー表示される

4 サンプル・アプリケーションを使ってみよう

4-3

サンプル・アプリケーションのローカル環境での実行

4-3-1

LINEのMessaging APIを始める

4-3-2

ローカルのOracle Databaseにスキーマを作成する

4-3-3

Node.js環境を作成する

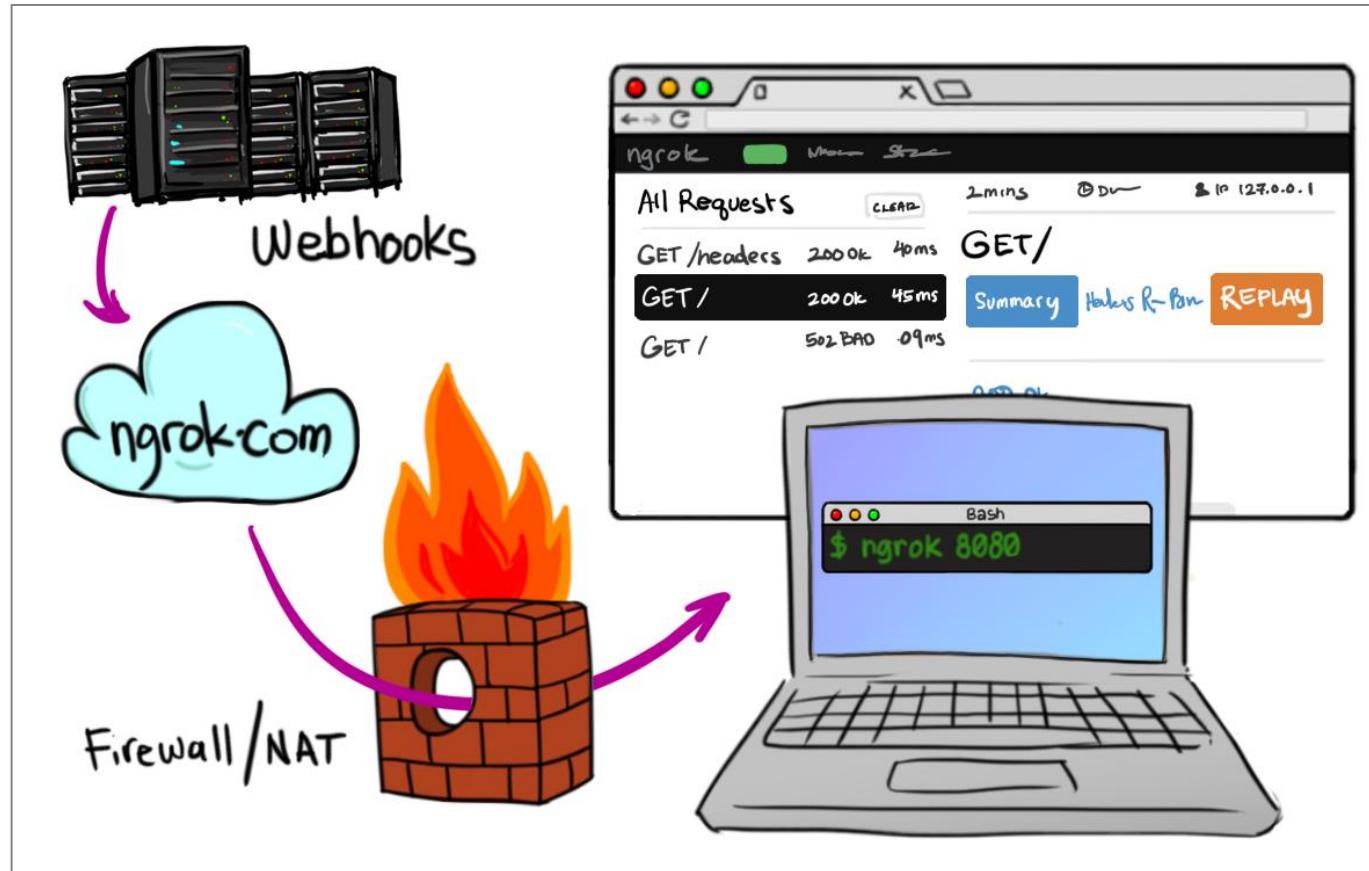
4-3-4

LINEとローカルPC間をトンネリングする

4-3-5

サンプル・アプリケーションを実行する

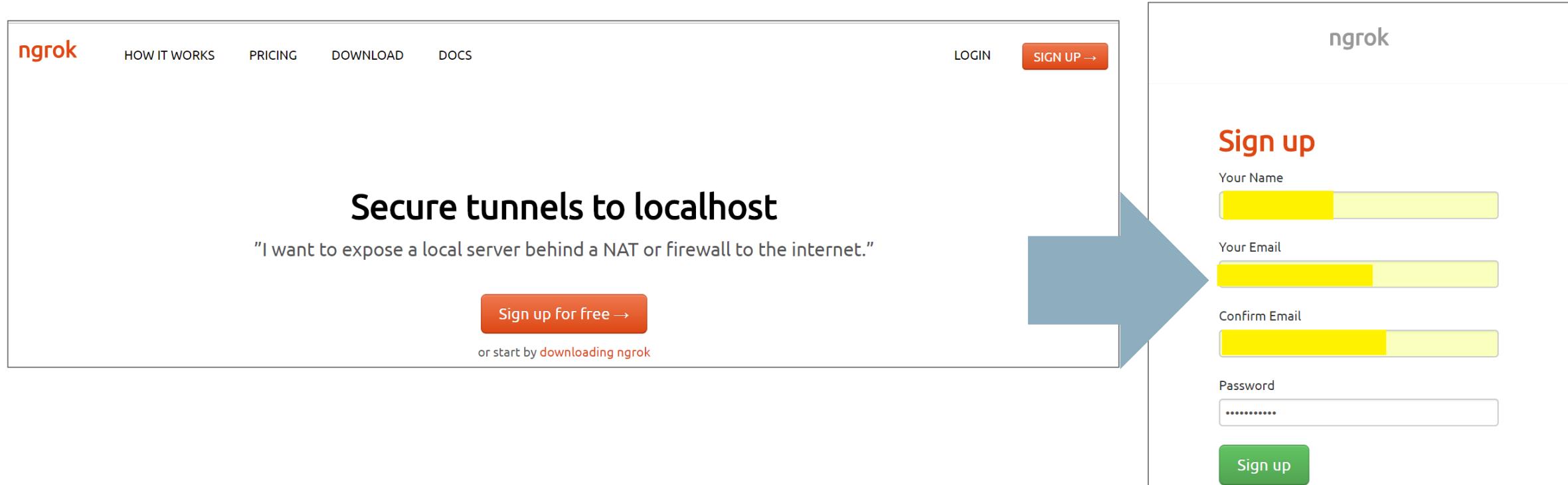
LINEとローカルPC間を接続する ngrokを利用したトンネリング



ngrok社サイト(<https://ngrok.com/>)より引用

LINEとローカルPC間を接続する ngrokのアカウント登録とダウンロード

- <https://ngrok.com/>にアクセスし、アカウント登録



LINEとローカルPC間を接続する ngrokのアカウント登録とダウンロード

The diagram illustrates the process of connecting LINE to a local PC using ngrok. It shows two main sections: the left section is the official ngrok documentation page, and the right section is the download page. A large blue arrow points from the documentation to the download page. Red arrows point from specific sections of the documentation to corresponding features on the download page.

Download & install ngrok

You can [download ngrok here](#). Follow the instructions to make sure it's installed correctly.

Connect your account

Running this command will add your account's authtoken to your `ngrok.yml` file. This will give you more features and all open tunnels will be listed here in the dashboard.

```
./ngrok authtoken 3fNEg[REDACTED]iKhDH
```

Start your first tunnel

Once you have your local server running, you can open a tunnel to the port where your application is running. Not working? Read the [documentation](#) to set up other tunneling options.

```
./ngrok http 80
```

The ngrok inspector

Visit <http://localhost:4040> anytime ngrok is running to see a live feed of requests to your tunnel. You can inspect the headers and responses for each request, or replace them with your own values.

Reserve a domain

Upgrade your account if you want to get the most out of ngrok. Once you upgrade, you can [reserve a ngrok subdomain](#) or [setup a custom CNAME](#) for each of your projects.

Explore the documentation

✉ As

Download ngrok

ngrok is easy to install. Download a single binary with *zero run-time dependencies* for any major platform.

[Download for Windows](#) [OS X](#) [Linux](#)

[Mac OS X \(32-bit\)](#) [Windows \(32-bit\)](#) [Linux \(ARM\)](#) [Linux \(32-bit\)](#) [FreeBSD \(64-Bit\)](#) [FreeBSD \(32-bit\)](#)

ユーザーによって
一意の値になっている

ローカルPCで実行する
モジュールをダウンロード

LINEとローカルPC間を接続する

ngrokモジュールを実行し、ローカルPCとnogrok間をトンネリング

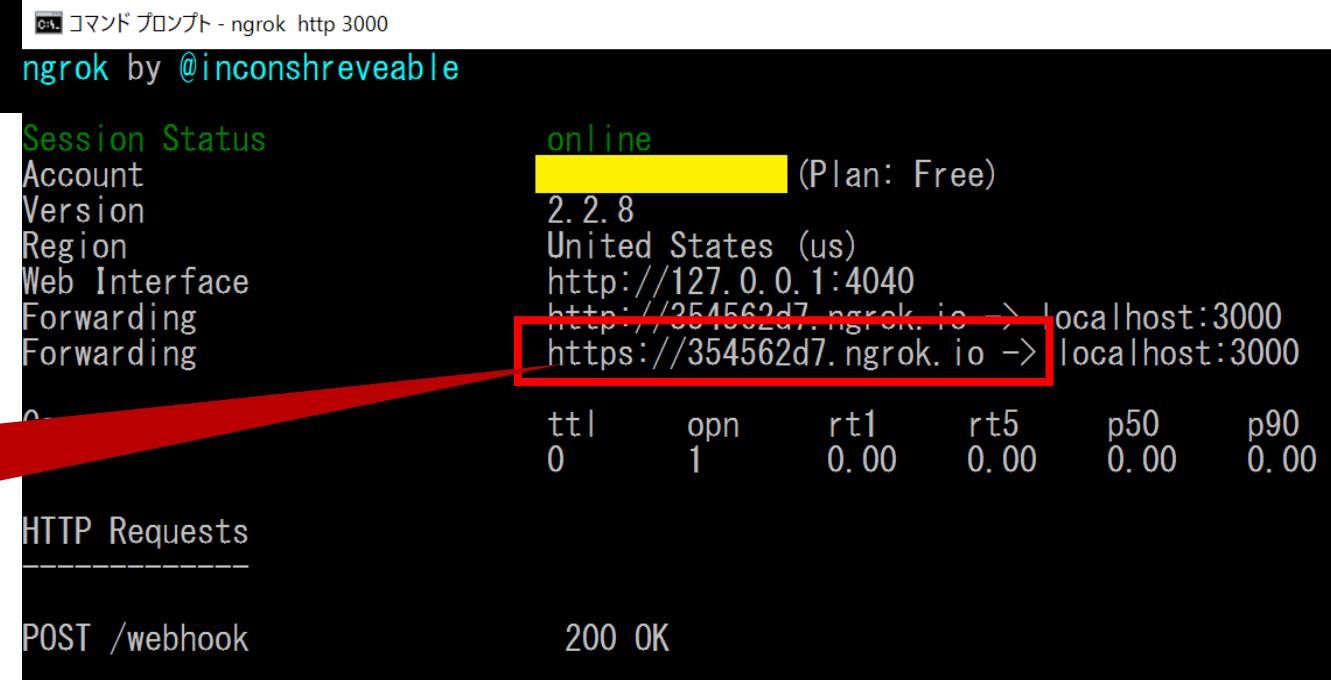
ngrok authtoken XXXX を実行(1回のみでよい)、ngrok http 3000を実行

```
D:\$proj\$botSample>ngrok authtoken 3fNE[REDACTED]t jKhdH  
Authtoken saved to configuration file: C:\[REDACTED]\NO/.ngrok2/ngrok.yml
```

```
D:\$proj\$botSample>ngrok http 3000
```

※サンプルアプリは、ローカルでは
PORT 3000でリスニングするので、
3000を指定

ngrokを実行する毎に割り当てら
れる URLが変わるので注意



LINEとローカルPC間を接続する

NogrokにトンネリングされたURLをLINEのWebhook URLとして設定する



4 サンプル・アプリケーションを使ってみよう

4-3

サンプル・アプリケーションのローカル環境での実行

4-3-1

LINEのMessaging APIを始める

4-3-2

ローカルのOracle Databaseにスキーマを作成する

4-3-3

Node.js環境を作成する

4-3-4

LINEとローカルPC間をトンネリングする

4-3-5

サンプル・アプリケーションを実行する

サンプルアプリを実行する

Webhook URL ※SSLのみ対応 ?
https://80e91e29.ngrok.io/webhook

Botのグループトーク参加 ?
利用しない

LINE@機能の利用

メッセージ本文はLINE@Managerの設定画面にて設定することができます。

自動応答メッセージ ?
利用しない

友だち追加時あいさつ ?
利用する

設定はこちら

QRコード

QRコードを LINEクライアントから読み込むと、アプリが利用可能に



node server.jsを実行中のコンソールには、このようにメッセージが表示される

```
[2] [TRACE] question.js - question.get: status: RESCHEDULE_END, "text": "14時でよつといで9か? ", type: "comment", "msgType": "text", "ct": "success", "id": "10000", "date": "2018-01-22", "time": "12時14分57秒", "previous": null, "next": null, "replyMsg": "", "data": {"order": 1, "id": "10000", "text": "1000", "status": "RESCHEDULE_END"}, "question": {"status": "RESCHEDULE_END", "text": "14時でよつといで9か? ", "type": "text", "next": null}, "previous": null, "answerCandidates": null}], "replyMsg": [{"type": "text", "text": "14時でよつといで9か? ", "status": "RESCHEDULE_END", "id": "10000", "date": "2018-01-22", "time": "12時14分57秒", "previous": null, "next": null, "data": {"order": 1, "id": "10000", "text": "1000", "status": "RESCHEDULE_END"}, "question": {"status": "RESCHEDULE_END", "text": "14時でよつといで9か? ", "type": "text", "next": null}, "previous": null, "answerCandidates": null}], "failure": "RESCHEDULE_01"}]
```

4 サンプル・アプリケーションを使ってみよう

- 4-1  サンプル・アプリケーションのダウンロード
- 4-2  サンプル・アプリケーションのプログラム構造
- 4-3  サンプル・アプリケーションのローカル環境での実行
- 4-4  サンプル・アプリケーションのクラウドへのデプロイ

4 サンプル・アプリケーションを使ってみよう

4-4

サンプル・アプリケーションのクラウドへのデプロイ

4-4-1

Oracle Database Cloudのインスタンスを作成する

4-4-2

Oracle Database Cloudへの接続設定を行う

4-4-3

Oracle Database Cloudにスキーマを作成する

4-4-4

Application Cacheを作成する

4-4-5

Application Container Cloudにアプリをデプロイする

4-4-6

Application Container Cloud と Database Cloudのバインド

4-4-7

LINEへのURLの反映

4 サンプル・アプリケーションを使ってみよう

4-4

サンプル・アプリケーションのクラウドへのデプロイ

4-4-1

Oracle Database Cloudのインスタンスを作成する

4-4-2

Oracle Database Cloudへの接続設定を行う

4-4-3

Oracle Database Cloudにスキーマを作成する

4-4-4

Application Cacheを作成する

4-4-5

Application Container Cloudにアプリをデプロイする

4-4-6

Application Container Cloud と Database Cloudのバインド

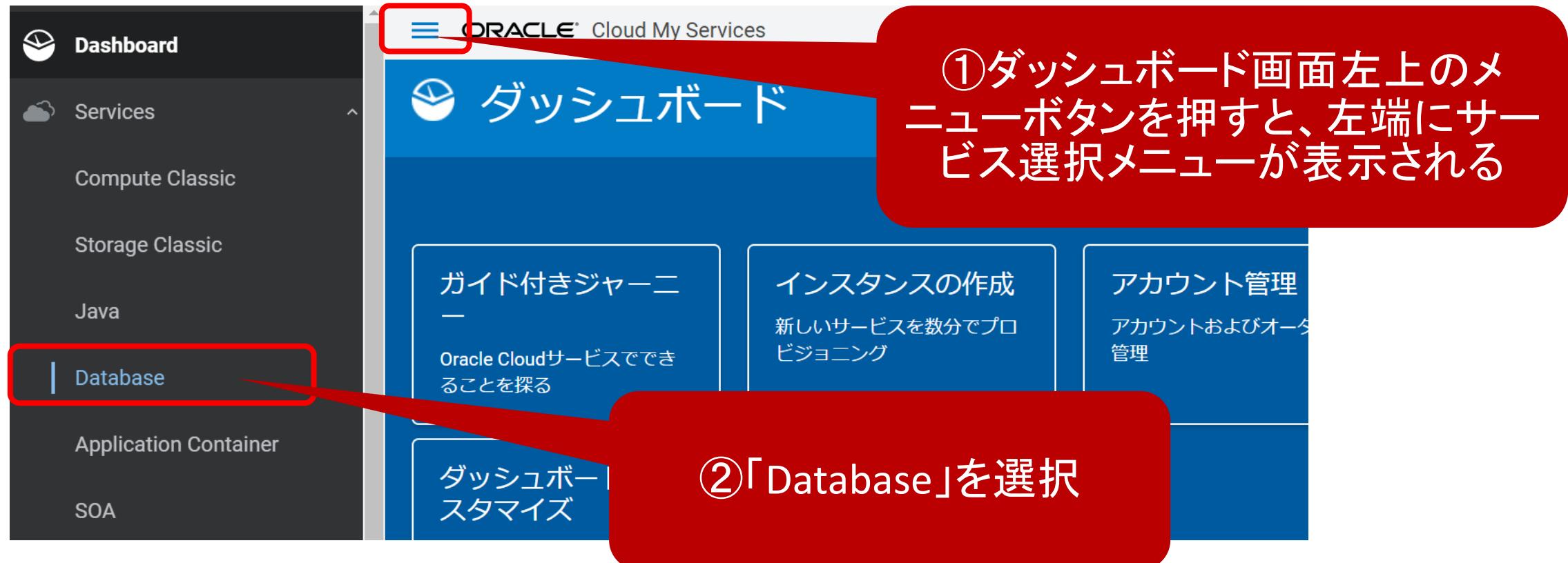
4-4-7

LINEへのURLの反映

Oracle Database Cloudのインスタンスを作成する

- Oracle Database Cloud Service Standard Editionの環境を作成
 - シェイプ: OC3 (OCPU 1, 7.5 GB Memory) でよい
 - Oracle Cloudへの SSH接続 (Port 22) が必要となる
 - Port 22アクセスが可能かどうかはネットワーク環境に依存
- 作成方法は以下を参照
 - <https://www.slideshare.net/oracle4engineer/oracle-database-cloud>

Oracle Database Cloudのインスタンスを作成する ダッシュボード画面から Database Cloud画面へのアクセス



Oracle Database Cloudのインスタンスを作成する

The screenshot shows the Oracle Database Cloud Service dashboard. At the top, there's a summary section with the following data:

サービス	OCPUs	メモリー	記憶域	パブリックIP
11	15	112.5 GB	2,719 GB	12

Below this is a search bar labeled "サービス名の全部または一部を入力します" and a blue button labeled "サービスの作成". A red callout box with the text "「サービスの作成」ボタンを押す" points to this button.

The main area displays a list of existing services, each with a thumbnail, service name, creation date, OCPUs, memory, and storage details. The first service in the list has its thumbnail and name redacted with yellow boxes.

サービス	作成日	OCPUs	メモリー	ストレージ
[Redacted]	2017/11/14 4時31分39秒 UTC	1	7.5 GB	150 GB
[Redacted]	2017/10/25 4時48分20秒 UTC	1	7.5 GB	150 GB

Oracle Database Cloudのインスタンスを作成する

Oracle Database Cloud Service
Create Service

取消 Service 詳細 確認 次 >

Service
Provide basic service instance information.

* サービス名: BotDB	* サブスクリプション・タイプ: Oracle Database Cloud Service
説明:	* 請求頻度: 時間
Notification Email: kaoru.hashino@oracle.com	* ソフトウェア・リリース: Oracle Database 12cリリース2
	* ソフトウェア・エディション: Standard Edition
	* Database Type: Single Instance

作成するDBの内容を指定。

今回は12cR2のStandard Editionを選択。

Oracle Database Cloudのインスタンスを作成する

The screenshot shows the Oracle Database Cloud instance creation process. It consists of three main panels:

- Service Step:** Shows basic service details like name and type.
- Detailed Step (highlighted):** Shows detailed configuration options:
 - サービス構成:** Includes fields for Database Name (SID: ORCL), PDB Name (PDB1), Management Password, and Confirmation. These fields are highlighted with a red box.
 - バックアップおよびリカバリ構成:** Includes a note about backup storage and a dropdown menu.
 - VMアクセスの公開鍵の入力:** A modal dialog for specifying an SSH public key, with the 'File Selection' tab selected. A blue arrow points from the 'SSH Public Key' field in the main panel to this dialog.
- 確認 Step:** Shows a summary of the configuration.

Red callout boxes provide additional context:

- A red callout points to the '次 >' (Next) button in the top right corner of the Detailed step, with the text: "データベース名、PDB名、管理パスワード等を設定" (Set database name, PDB name, management password, etc.).
- A red callout points to the 'SSH Public Key' input field in the 'VMアクセスの公開鍵の入力' dialog, with the text: "クラウドにSSHアクセスするためのSSH公開鍵を指定。" (Specify the SSH public key for SSH access to the cloud).
- A red callout points to the bottom left of the Detailed step panel, with the text: "検証用途の場合、最小の構成でかまいません" (In case of validation purposes, it's fine to use the minimum configuration).

Oracle Database Cloudのインスタンスを作成する

その他の設定

サービス構成

* データベース名(SID) ORCL

* PDB名 PDB1

* 管理パスワード
* パスワードの確認

* 使用可能なデータベース記憶域(GB) 40

合計データ・ファイル記憶域(GB) 105

* コンピュート・シェイプ OC3 - 1.0 OCPU, 7.5 GB RAM

* SSH公開鍵 labkey.pub

Use High Performance Storage

Advanced Settings

* Listener Port 1521

* タイムゾーン (UTC) Coordinated Universal Time

* 文字セット AL32UTF8 - Unicode汎用文字セ

* 各国語文字セット AL16UTF16 - Unicode UTF-16

Oracle GoldenGateの有効化

バックアップおよびリカバリ構成

* バックアップの保存先 なし

Initialize Data From Backup

* 既存のバックアップからのインスタンスの作成 No

検証用途のため、バックアップ等は指定していません

Oracle Database Cloudのインスタンスを作成する

Oracle Database Cloud Service
Create Service

前 取消 Service 詳細 確認 作成 >

確認
レスポンスを確認して、このOracle Database Cloud Serviceインスタンスを作成してください。

Service		Database Configuration	
サービス名:	BotDB	データベース名(SID):	ORCL
説明:		PDB名:	PDB1
Bring Your Own License:	いいえ	使用可能なデータベース記憶域:	40
サービス・レベル:	Oracle Database Cloud Service	合計データ・ファイル記憶域:	105
請求頻度:	時間	Listener Port:	1521
ソフトウェア・リリース:	Oracle Database 12cリリース2	タイムゾーン:	(UTC) Coordinated Univers...
ソフトウェア・エディション:	Standard Edition	文字セット:	AL32UTF8 - Unicode汎用文字セット...
コンピュート・シェイプ:	OC3 - 1.0 OCPU, 7.5 GB RAM	各国語文字セット:	AL16UTF16 - Unicode UTF-1...
キー:	labkey.pub	GoldenGateを組み込む:	いいえ
Use High Performance Storage:	いいえ		

Standby Database Configuration

Backup and Recovery Configuration

Oracle Database Cloudのインスタンスを作成する

The screenshot shows the Oracle Cloud My Services dashboard. At the top, there's a navigation bar with icons for Dashboard, User, and Notifications. Below it, a sidebar has a cloud icon and the text "Oracle Database Cloud Service". The main content area has tabs for "サービス" (selected), "アクティビティ", and "SSHアクセス". A timestamp at the top right indicates "2017/11/22 3時26分11秒 UTC現在". Below this, a summary table provides resource details:

サマリー	12	16	120 GB	2,719 GB	12	-
サービス	サービス	OCPUs	メモリー	記憶域	パブリックIP	

Below the summary, there's a "サービス" section with a search bar and a "サービスの作成" button. A message indicates that a database instance creation request has been accepted. The detailed view for the "BotDB" instance shows its status as "進行中", version "12.2.0.1", edition "Standard Edition", and configuration details like OCPUs, memory, and storage.

4 サンプル・アプリケーションを使ってみよう

4-4

サンプル・アプリケーションのクラウドへのデプロイ

4-4-1

Oracle Database Cloudのインスタンスを作成する

4-4-2

Oracle Database Cloudへの接続設定を行う

4-4-3

Oracle Database Cloudにスキーマを作成する

4-4-4

Application Cacheを作成する

4-4-5

Application Container Cloudにアプリをデプロイする

4-4-6

Application Container Cloud と Database Cloudのバインド

4-4-7

LINEへのURLの反映

Oracle Database Cloudのインスタンスへの接続設定を行う 接続文字列の確認

The screenshot shows the Oracle Database Cloud Service dashboard. At the top, there's a navigation bar with a cloud icon, the text "Oracle Database Cloud Service", and tabs for "サービス", "アクティビティ", "SSHアクセス", and "ようこそ". The date "2017/11/22 4時01分45秒 UTC現在" is also displayed. Below the summary, there are resource details: 12 services, 16 OCPU, 120 GB memory, 2,883 GB storage, and 13 public IPs. The "サービス" section contains a search bar and a "サービスの作成" button. A red box highlights the "BotDB" service entry, which includes a cloud icon, version "12.2.0.1", edition "Standard Edition", and creation date "2017/11/22 3時26分10秒 UTC". A red callout bubble points to the "BotDB" entry with the text "作成したサービス名をクリックする".

作成したサービス名を
クリックする

Oracle Database Cloudのインスタンスへの接続設定を行う 接続文字列の確認

The screenshot shows the Oracle Database Cloud Service Overview page. On the left, there's a sidebar with '概要' (Summary) showing 1ノード and '管理' (Management) showing 0 patches and 0 snapshots. The main area has a 'Service Overview' section with resource details: 1ノード (OCPUs), 7.5 GB メモリー (Memory), and 164 GB 記憶域 (Storage). Below this, the 'ステータス' (Status) is '実行中' (Running), and the '接続文字列' (Connection String) is displayed as 'BotDB:1521/PDB1.jptest01.oraclecloud.internal'. Further down, under 'Resources', the 'インスタンス' (Instance) is listed as 'BotDB', with 'パブリックIP' (Public IP) '146.56.1.237' and 'SID' (System Identifier) 'ORCL'. A red box highlights the connection string and another red box highlights the instance details.

接続文字列をメモしておく
(このあと SQL Developer で接続する際に利用)

この例での
「BotDB:1521/PDB1.jptest01.oracle
cloud.internal」のうち、/以下の
「PDB1.jptest01.oraclecloud.intern
al」がサービス名

IPアドレス、SIDをメモしておく

Oracle Database Cloudのインスタンスへの接続設定を行う Oracle Net接続(Port 1521)のための設定



Oracle Database Cloudのインスタンスへの接続設定を行う Oracle Net接続(Port 1521)のための設定



Oracle Database Cloudのインスタンスへの接続設定を行う

Oracle Net接続(Port 1521)のための設定

①作成したサービスから、
Ora_p2_dblistenerを探す

BotDB/db_1/ora_p2_dbconsole		BotDB/db_1/ora_dbconsole	public-internet	BotDB/db_1/ora_db
BotDB/db_1/ora_p2_dbexpress	無効	BotDB/db_1/ora_dbexpress	public-internet	BotDB/db_1/ora_db
BotDB/db_1/ora_p2_dblistener	有効	BotDB/db_1/ora_dblistener	public-internet	BotDB/db_1/ora_db
BotDB/db_1/ora_p2_http	無効	BotDB/db_1/ora_http	public-internet	BotDB/db_1/ora_db
BotDB/db_1/ora_p2_httpadmin	無効	BotDB/db_1/ora_httpadmin	public-internet	BotDB/db_1/ora_db
BotDB/db_1/ora_p2_httpsl	無効	BotDB/db_1/ora_httpsl	public-internet	BotDB/db_1/ora_db
BotDB/db_1/ora_p2_ssh	有効	ssh	public-internet	
BotDB/db_1/ora_trusted_hosts_dblistener	有効	BotDB/db_1/ora_dblistener	BotDB/db_1/ora_trusted_hosts_db	
BotDB/db_1/sys_infra2db_ssh	有効	ssh	paas-infra	

②右端のメニューボタンを押す

更新
削除

③「更新」を選択

Oracle Database Cloudのインスタンスへの接続設定を行う Oracle Net接続(Port 1521)のための設定



ステータスを「有効」に設定する

これで作成した Oracle Database Cloud Serviceは、
PORT 1521でアクセスできるようになります。

※ユーザー側もPORT 1521でのアクセスを許可する
設定になっている必要があります

4 サンプル・アプリケーションを使ってみよう

4-4

サンプル・アプリケーションのクラウドへのデプロイ

4-4-1

Oracle Database Cloudのインスタンスを作成する

4-4-2

Oracle Database Cloudへの接続設定を行う

4-4-3

Oracle Database Cloudにスキーマを作成する

4-4-4

Application Cacheを作成する

4-4-5

Application Container Cloudにアプリをデプロイする

4-4-6

Application Container Cloud と Database Cloudのバインド

4-4-7

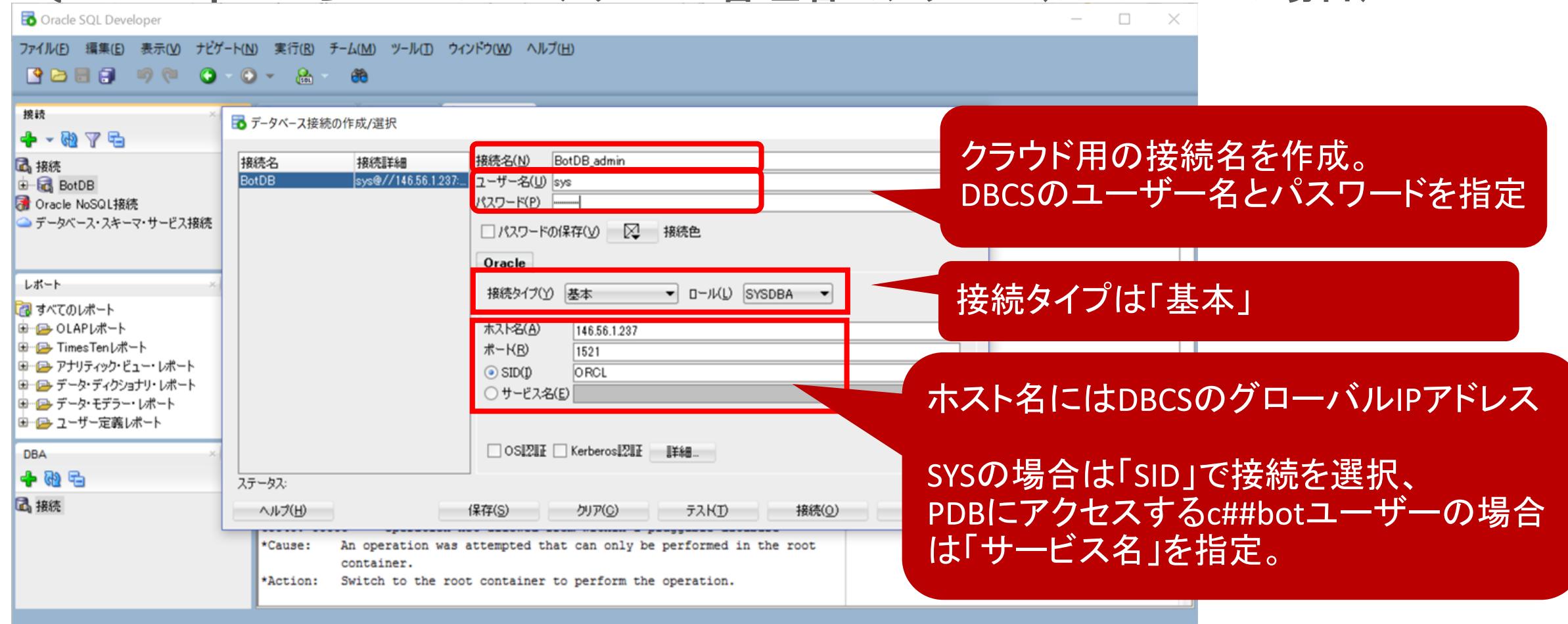
LINEへのURLの反映

Oracle Database Cloudにスキーマを作成する

- ローカルのDBに対して行った方法と基本的に同じ
- 差分として、クラウド上のDBCSにアクセスする方法を説明します
 - PORT 1521で接続する方法
 - PORT 22(SSH)で接続する方法

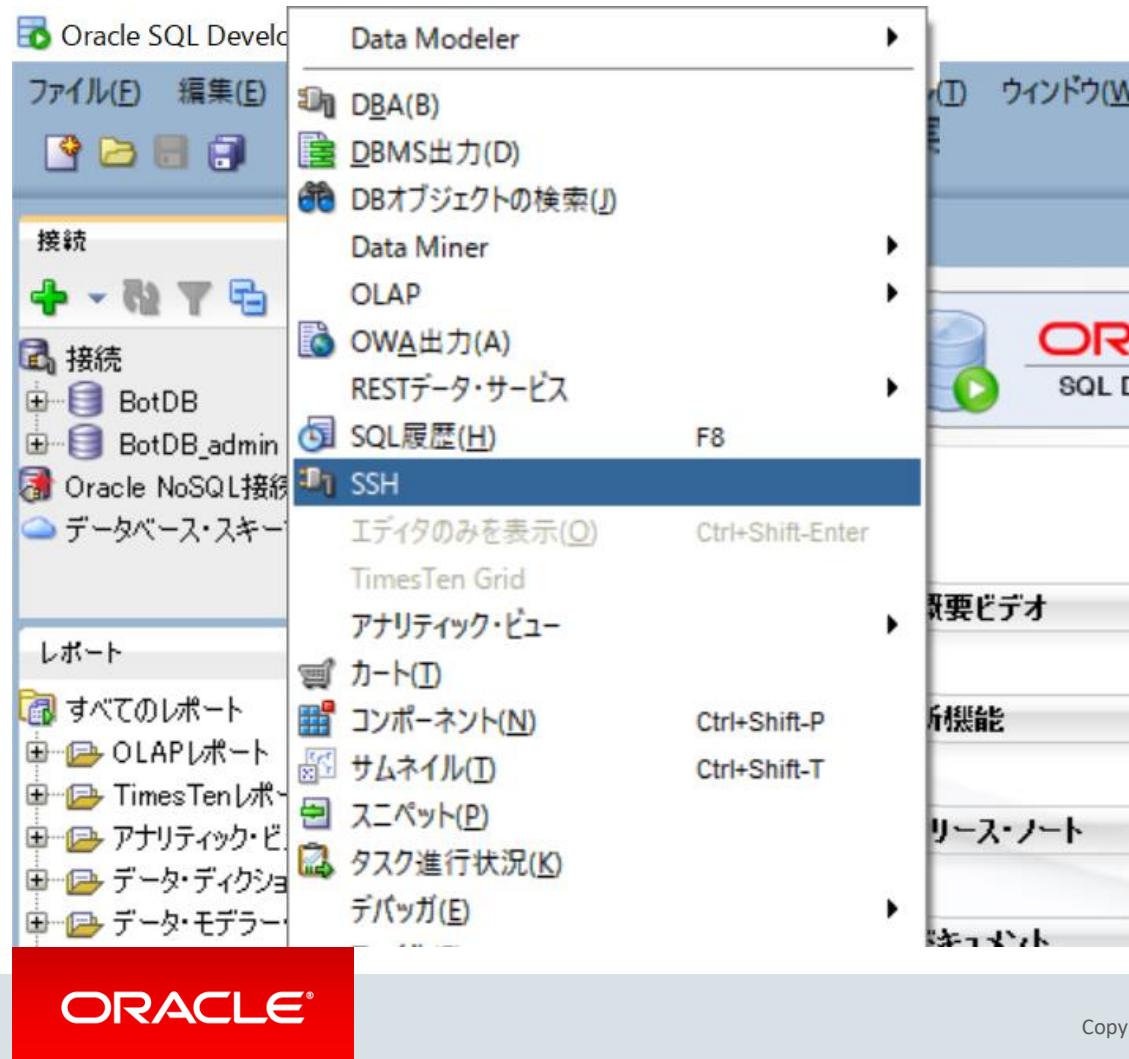
Oracle Database Cloudにスキーマを作成する

SQL DeveloperからDBCSのコンテナDBに管理者でアクセス(PORT1521の場合)



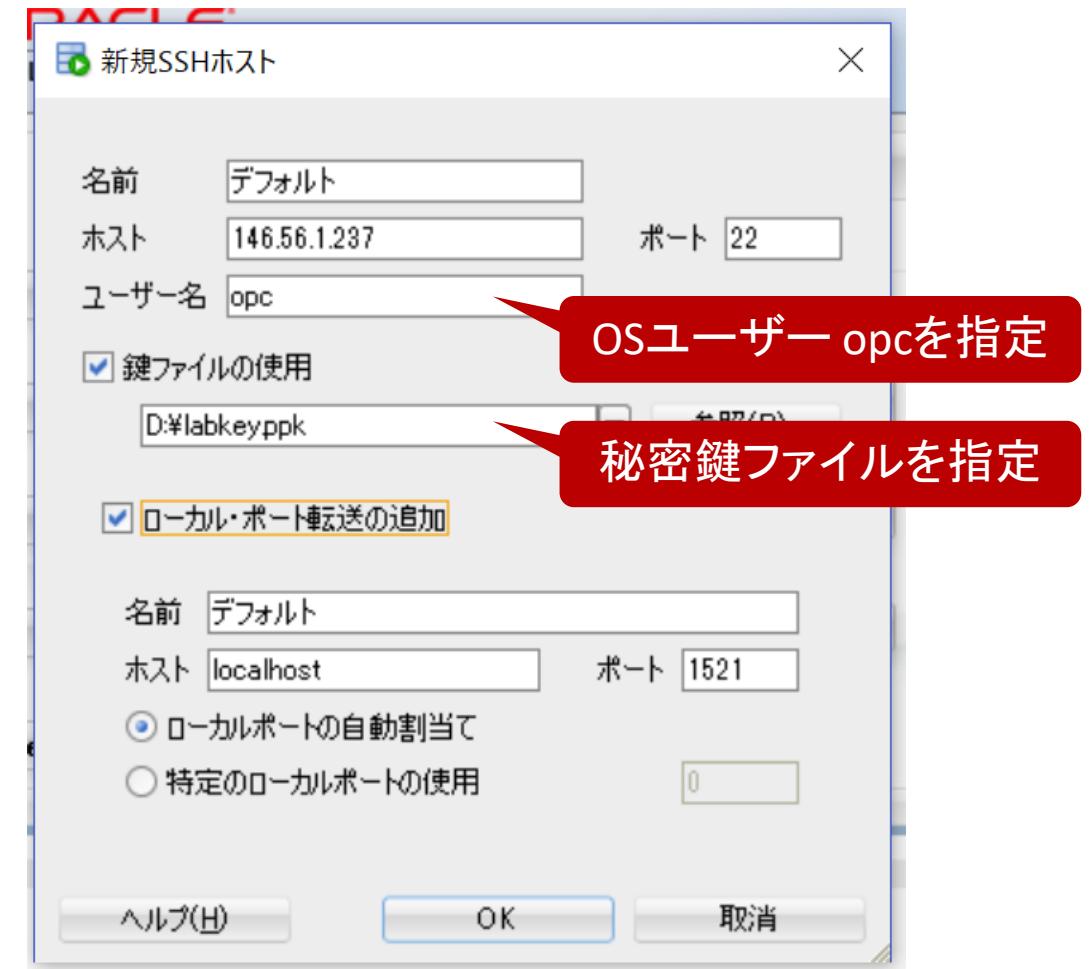
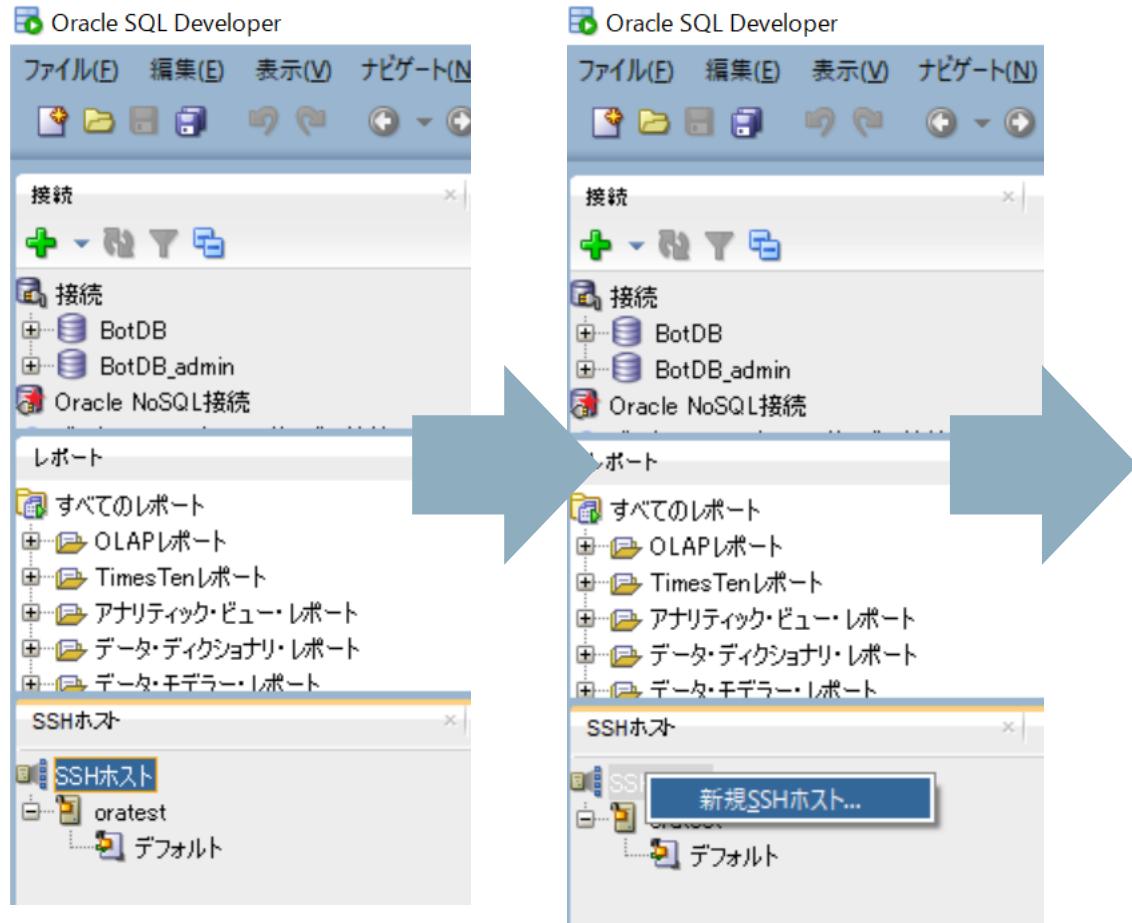
Oracle Database Cloudにスキーマを作成する

SQL DeveloperからDBCSのコンテナDBに管理者でアクセス(PORT 22の場合)



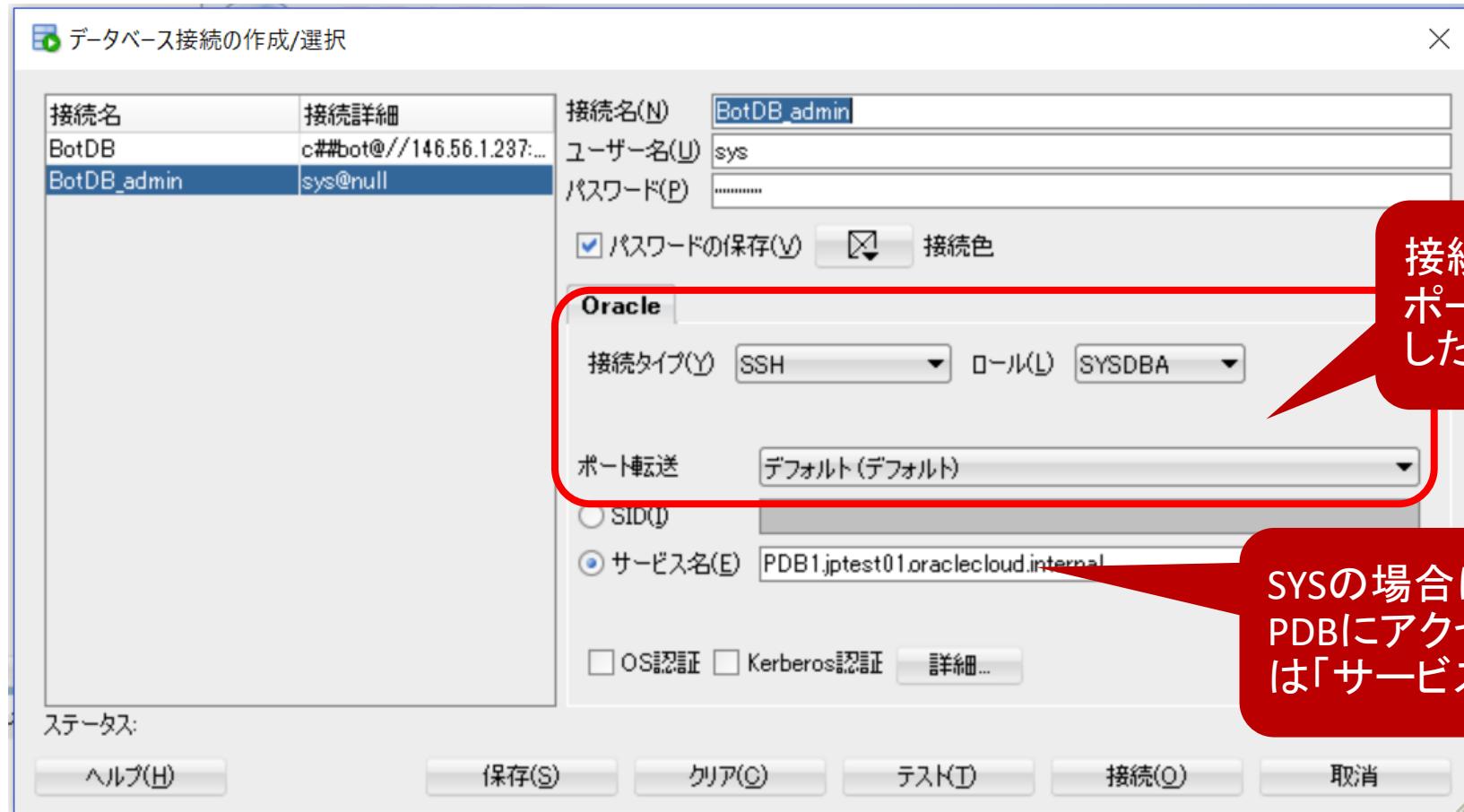
Oracle Database Cloudにスキーマを作成する

SQL DeveloperからDBCSのコンテナDBに管理者でアクセス(PORT 22の場合)



Oracle Database Cloudにスキーマを作成する

SQL DeveloperからDBCSのコンテナDBに管理者でアクセス(PORT 22の場合)



4 サンプル・アプリケーションを使ってみよう

4-4

サンプル・アプリケーションのクラウドへのデプロイ

4-4-1

Oracle Database Cloudのインスタンスを作成する

4-4-2

Oracle Database Cloudへの接続設定を行う

4-4-3

Oracle Database Cloudにスキーマを作成する

4-4-4

Application Cacheを作成する

4-4-5

Application Container Cloudにアプリをデプロイする

4-4-6

Application Container Cloud と Database Cloudのバインド

4-4-7

LINEへのURLの反映

Application Cacheを作成する

ダッシュボードから Application Container画面に移動する

The screenshot shows the Oracle Cloud My Services dashboard. On the left, there's a sidebar with various service categories: Dashboard, Services, Compute Classic, Storage Classic, Java, Database, Application Container (which is highlighted with a red box), SOA, and Monitoring. At the top center, there's a navigation bar with the Oracle logo and the text "ORACLE® Cloud My Services". Below the navigation bar, the main area is titled "ダッシュボード" (Dashboard) and contains three cards: "ガイド付きジャーニー" (Guided Journey), "インスタンスの作成" (Instance Creation), and "アカウント" (Account). A red callout bubble points to the "Application Container" menu item in the sidebar with the text "②「Application Container」を選択" (Select "Application Container"). Another red callout bubble points to the top-left menu icon (three horizontal lines) with the text "①ダッシュボード画面左上のメニュー ボタンを押すと、左端にサービス選択 メニューが表示される" (When you press the menu button at the top left of the dashboard, a service selection menu will be displayed on the left side).

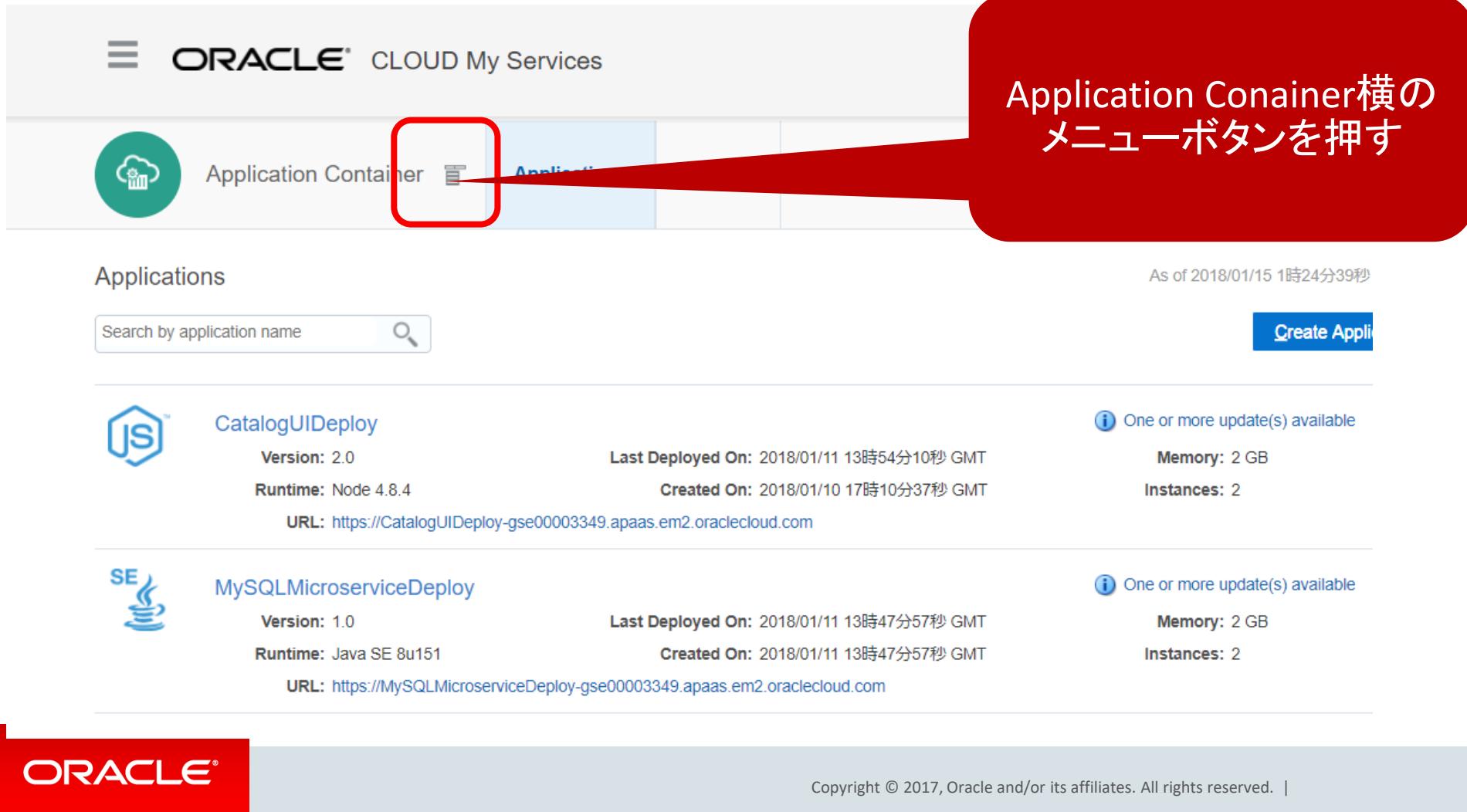
①ダッシュボード画面左上のメニュー ボタンを押すと、左端にサービス選択 メニューが表示される

②「Application Container」を選択

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. | 198

Application Cacheを作成する

Application Container画面からApplication Cache画面に移動



Application Container 横の
メニュー ボタンを押す

☰ ORACLE® CLOUD My Services

Application Container

Applications

As of 2018/01/15 1時24分39秒

Search by application name 🔍

Create Appli

CatalogUIDeploy

Version: 2.0 Last Deployed On: 2018/01/11 13時54分10秒 GMT
Runtime: Node 4.8.4 Created On: 2018/01/10 17時10分37秒 GMT
URL: <https://CatalogUIDeploy-gse00003349.apaas.em2.oraclecloud.com>

MySQLMicroserviceDeploy

Version: 1.0 Last Deployed On: 2018/01/11 13時47分57秒 GMT
Runtime: Java SE 8u151 Created On: 2018/01/11 13時47分57秒 GMT
URL: <https://MySQLMicroserviceDeploy-gse00003349.apaas.em2.oraclecloud.com>

ORACLE®

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. | 199

Application Cacheを作成する

Application Container画面からApplication Cache画面に移動

The screenshot shows the Oracle Cloud My Services dashboard. On the left, there is a sidebar with various service categories: Analytics Cloud, Big Data Cloud, Container, Event Hub, Event Hub - Dedicated, GoldenGate, Internet of Things Cloud - Enterprise, MySQL, SOA, Application Container, Applications, Application Cache, and Cloud Stack. The 'Application Container' section is currently selected, indicated by a red box around its name. On the right, there is a main content area for the 'Application Container'. It displays deployment details: Last Deployed On: 2018/01/11 13時54分10秒 GMT and Created On: 2018/01/10 17時10分37秒 GMT. Below this, there is a link to the application's URL: oy-gse00003349.apaas.em2.oraclecloud.com. Further down, another deployment entry is shown: Last Deployed On: 2018/01/11 13時47分57秒 GMT and Created On: 2018/01/11 13時47分57秒 GMT, with the URL: serviceDeploy-gse00003349.apaas.em2.oraclecloud.com. The top navigation bar includes 'Dashboard', 'Users', 'Notifications', and user information 'cloud.admin'. A 'Create Application' button is also visible.

Application Cacheを
選択する

Application Cacheを作成する

Application Cache画面

The screenshot shows the Oracle Cloud My Services interface. At the top, there's a navigation bar with the Oracle logo and 'CLOUD My Services'. Below it, there are three tabs: 'Application Container' (with a green icon), 'Application Caches' (which is highlighted with a red box and has a red arrow pointing to a callout), and 'Activity'. The main content area has a blue background with a white cloud icon containing a gear and a server icon. It features a large 'Welcome to Application Cache!' heading and a section titled 'GO FROM ZERO TO DEPLOYED APPLICATION CACHE TODAY'. Below that, it says 'Let's get started. We'll show you how to create a new cache, deploy a sample application, and start working with the cache. Let's go!'. At the bottom of this section are three yellow buttons: 'Watch Video', 'Follow Tutorial', and 'Go to Console'. A yellow banner at the very bottom says 'See what's new and noteworthy in this release'. A red callout bubble points from the 'Application Caches' tab towards the text 'Application Cache画面になる。「Application Cache」タブを選択。'.

Application Cache画面になる。「Application Cache」タブを選択。

Application Cacheを作成する

Application Cacheインスタンスの作成

The screenshot shows the Oracle Cloud My Services interface. At the top, there's a navigation bar with the Oracle logo and 'CLOUD My Services'. Below it is a menu bar with 'Application Container' (selected), 'Application Caches' (highlighted in blue), and 'Activity'. On the left, there's a green circular icon with a grid pattern. In the center, there's a message: 'You don't have any services. After meeting the [prerequisites](#), use this button to create a service.' Below this, there's a section titled 'Need help creating the service?' with two options: '- Watch a video' and '- Step through a tutorial'. A large red speech bubble on the right contains the text '「インスタンスの作成」ボタンを押す。' (Press the 'Create instance' button). A red rectangular box highlights the 'Create instance' button, which is located at the bottom right of the main content area. A grey curved arrow points from the text in the red bubble towards the highlighted button.

「インスタンスの作成」
ボタンを押す。

インスタンスの作成

インスタンス

You don't have any services. After meeting the [prerequisites](#), use this button to create a service.

Need help creating the service?

- Watch a video
- Step through a tutorial

▶ インスタンス作成および削除履歴

Application Cacheを作成する

Application Cacheインスタンスの作成

Application Cache
Create Service

取消

Service 確認

次 >

Service
Provide basic service instance information.

Details

* Service Name: botCache

Description:

Notification Email: kaoru.hashino@oracle.com

* Region: AP005_Z11

Metering Frequency: HOURLY

Cache Configuration

Deployment Type: Basic

* Cache Capacity [GB]: 1

Total Memory Allocated [GB]: 2.0

サービス名、リージョン、デプロイメント・タイプ、キャッシュサイズを選択

Application Cacheを作成する

Application Cacheインスタンスの作成

Application Cache Create Service

前 取消

Service 確認

作成 >

Confirmation
Confirm your responses and create this Application Cache instance.

Service		Cache Configuration	
Service Name:	botCache	Deployment Type:	Basic
Notification Email:	kaoru.hashino@oracle.com	Cache Capacity [GB]:	1
Service Level:	Service with tooling support	Total Memory Allocated [GB]:	2.0
Software Release:	Update to 1.7.0.0.1710231629		
Software Edition:	Grid Edition		
Metering Frequency:	Hourly		
Region:	AP005_Z11		

Application Cacheを作成する

Application Cacheインスタンスの作成

2018/01/15 1時37分06秒 UTC現在 

インスタンス

インスタンス名別検索 

インスタンスの作成

 Oracle Database Cloud Serviceインスタンス作成リクエストが{0}でした



botCache

Status: Creating service ...
バージョン: 1.7.0.0.1710231629

送信日: 2018/01/15 1時37分06秒 UTC

メモリー: 2 GB



▶ インスタンス作成および削除履歴

4 サンプル・アプリケーションを使ってみよう

4-4

サンプル・アプリケーションのクラウドへのデプロイ

4-4-1

Oracle Database Cloudのインスタンスを作成する

4-4-2

Oracle Database Cloudへの接続設定を行う

4-4-3

Oracle Database Cloudにスキーマを作成する

4-4-4

Application Cacheを作成する

4-4-5

Application Container Cloudにアプリをデプロイする

4-4-6

Application Container Cloud と Database Cloudのバインド

4-4-7

LINEへのURLの反映

アップロードするzipファイルを用意

アップロードするファイルに応じて作業

1. accsフォルダにあるbotSample.zipファイルをアップロードする場合
 - 同じフォルダにあるline.config.jsonファイルを編集し ZIPファイルに追加
 - 利用するLINEチャネルのアクセストークン、チャネル・シークレットの値を設定
2. ローカル環境で試したsrcフォルダのファイルを自分でZIPする場合
 - oracledbパッケージを削除してからZIPすること(OS依存のため)
 - コマンド
 - npm uninstall oracledb --save
 - accsフォルダには oracledbパッケージは含まれていない

Application Container Cloudにアプリをデプロイする ダッシュボードから Application Container画面に移動する

The screenshot shows the Oracle Cloud My Services dashboard. On the left, there's a sidebar with various service categories: Dashboard, Services, Compute Classic, Storage Classic, Java, Database, Application Container (which is highlighted with a red box), SOA, and Monitoring. At the top center, there's a navigation bar with the Oracle logo and the text "ORACLE® Cloud My Services". Below the navigation bar, the main area is titled "ダッシュボード" (Dashboard) and contains three cards: "ガイド付きジャーニー" (Guided Journey), "インスタンスの作成" (Instance Creation), and "アカウント" (Account). A red box highlights the "Application Container" menu item in the sidebar. Another red box highlights the three-line menu icon at the top left of the dashboard area. Red arrows point from the numbered callouts to their respective targets.

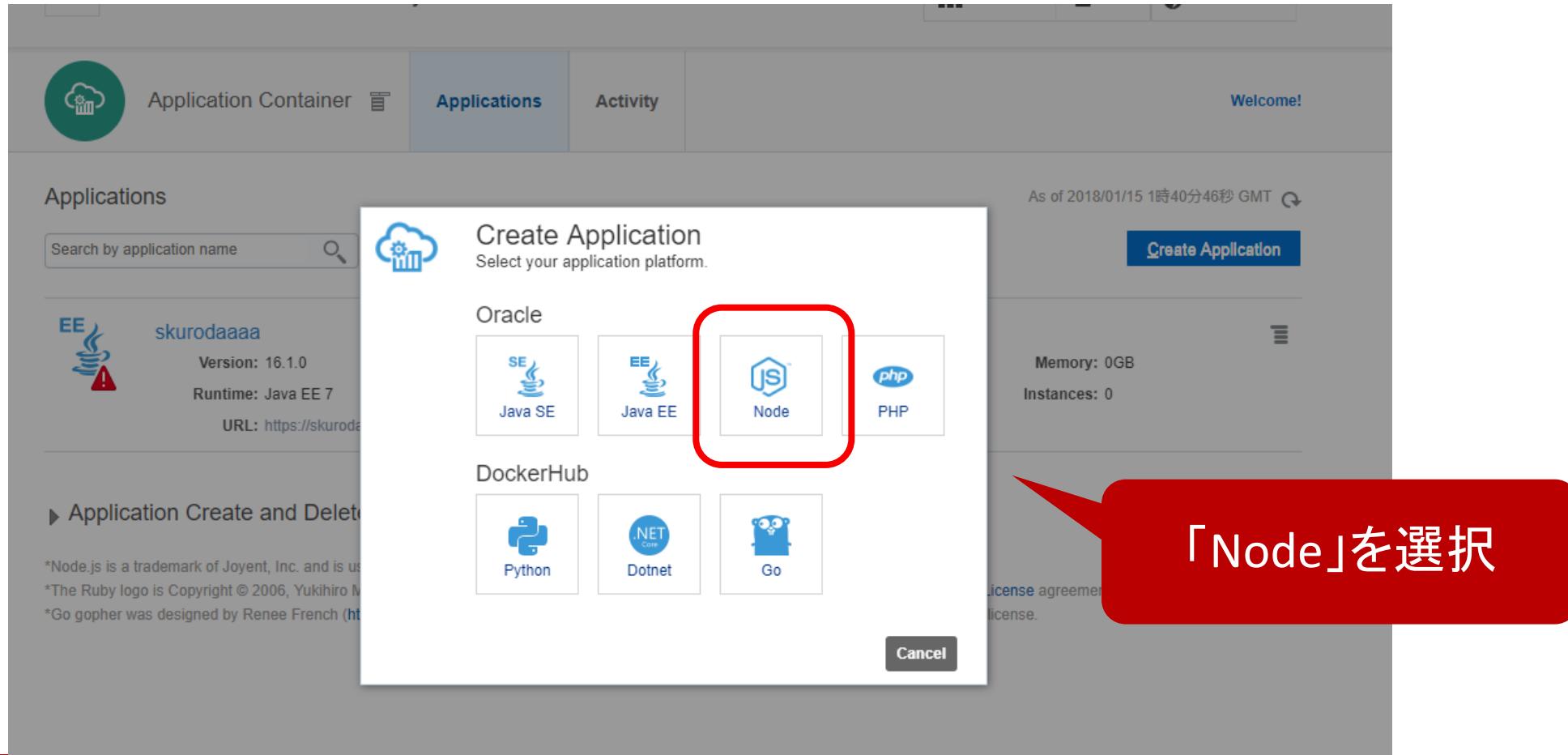
- ① ダッシュボード画面左上のメニュー
ボタンを押すと、左端にサービス選択
メニューが表示される
- ② 「Application Container」を選択

Application Container Cloudにアプリをデプロイする

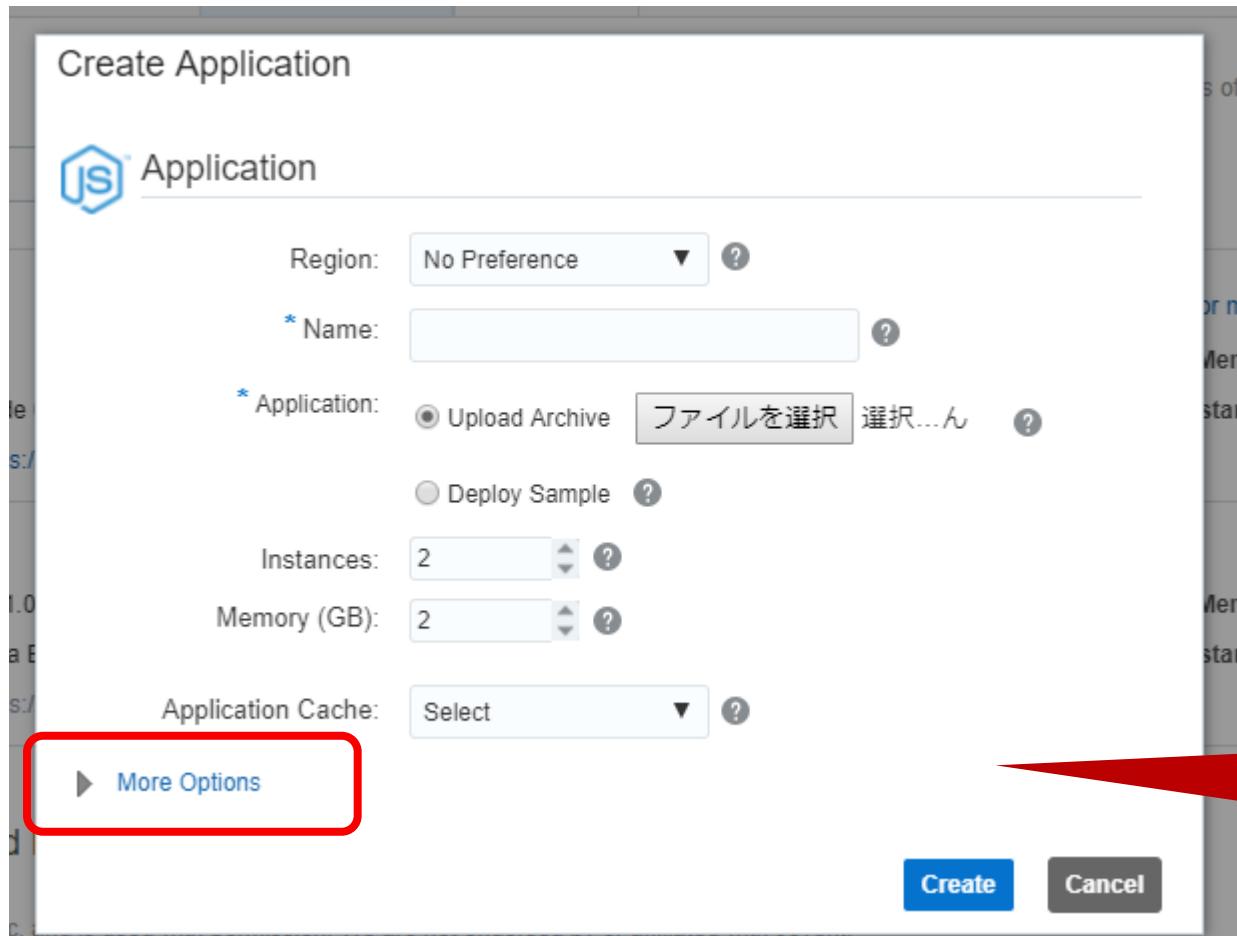
The screenshot shows the Oracle Cloud My Services Application Container dashboard. At the top, there's a navigation bar with 'Dashboard', 'Users', and 'Notifications'. Below it, a main menu includes 'Application Container' (with a cloud icon), 'Applications' (which is selected and highlighted in blue), 'Activity', and 'Welcome!'. A search bar for 'Search by application name' is also present. On the right side, there's a large 'Create Application' button, which is highlighted with a red rectangular box. In the center, there's a summary card for an application named 'skurodaaaa-jptest01.apaas.ap5.oraclecloud.com'. The card displays the following details: Version: 16.1.0, Runtime: Java EE 7, URL: <https://skurodaaaa-jptest01.apaas.ap5.oraclecloud.com>, Last Deployed On: 2017/12/27 7時58分09秒 GMT, Created On: 2017/12/27 7時58分09秒 GMT, Memory: 0GB, and Instances: 0. At the bottom left, there's a link to 'Application Create and Delete History'. A large red callout bubble on the right side points to the 'Create Application' button with the text '「Create Application」を押す'.

「Create Application」を押す

Application Container Cloudにアプリをデプロイする



Application Container Cloudにアプリをデプロイする



More Optionsをクリック

Application Container Cloudにアプリをデプロイする

The screenshot shows the 'Create Application' screen in the Oracle Application Container Cloud interface. The application name is set to 'botSample'. The deployment method is chosen as 'Upload Archive' and the archive file is 'botS...zip'. The application has 1 instance and 1 GB of memory. The application cache is set to 'botCache'. In the 'More Options' section, the manifest file is 'manifest.json', deployment configuration is 'Deployment Configuration', notification email is 'kaoru.hashino@oracle.com', and the node version is set to 8. A note field is present at the bottom.

botSample.zipを指定

Application Cacheを指定
※Application Cache未作成の場合は表示されない

Manifest.jsonを指定

Node Versionは 8

Application Container Cloudにアプリをデプロイする

The screenshot shows the Oracle Application Container Cloud interface. At the top, there's a navigation bar with a green circular icon containing a cloud and trash can, followed by the text "Application Container" and a refresh icon. The "Applications" tab is selected, and the "Activity" tab is also visible. On the right, a "Welcome!" message is displayed. Below the navigation, the word "Applications" is centered, along with a timestamp "As of 2018/01/21 3時06分41秒 GMT" and a refresh icon. A search bar with the placeholder "Search by application name" and a magnifying glass icon is on the left. On the right, a blue button labeled "Create Application" is visible. The main content area displays a single application entry for "botSample". It includes a blue hexagonal icon with "JS" inside, the application name "botSample" in blue, and deployment details: Version 1.0, Last Deployed On: 2018/01/21 3時06分02秒 GMT, Runtime: Node 8.1.4, Created On: 2018/01/21 3時06分02秒 GMT, Memory: 1 GB, Instances: 1, and a URL link: <https://botSample-jptest01.apaas.ap5.oraclecloud.com>. To the right of the application details is a three-line menu icon.

Application Container

Applications

Activity

Welcome!

Applications

As of 2018/01/21 3時06分41秒 GMT

Create Application

botSample

Version: 1.0

Last Deployed On: 2018/01/21 3時06分02秒 GMT

Runtime: Node 8.1.4

Created On: 2018/01/21 3時06分02秒 GMT

Memory: 1 GB

Instances: 1

URL: <https://botSample-jptest01.apaas.ap5.oraclecloud.com>

ORACLE®

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. | 213

4 サンプル・アプリケーションを使ってみよう

4-4

サンプル・アプリケーションのクラウドへのデプロイ

4-4-1

Oracle Database Cloudのインスタンスを作成する

4-4-2

Oracle Database Cloudへの接続設定を行う

4-4-3

Oracle Database Cloudにスキーマを作成する

4-4-4

Application Cacheを作成する

4-4-5

Application Container Cloudにアプリをデプロイする

4-4-6

Application Container Cloud と Database Cloudのバインド

4-4-7

LINEへのURLの反映

Application Container Cloud と Database Cloud のバインド

The screenshot shows the Oracle Cloud Application Container interface. At the top, there's a navigation bar with icons for Application Container, Applications (which is selected), Activity, and Welcome!. Below this is a search bar labeled "Search by application name" and a "Create Application" button. The main area is titled "Applications" and shows a list of applications. One application, "botSample", is highlighted with a red box and a red arrow pointing to it from a red button at the bottom. The "botSample" entry includes details: Version 1.0, Runtime Node 6.11.1, URL <https://botSample-jptest.ap5.oraclecloud.com>, Last Deployed On: 2018/01/15 1時44分54秒 GMT, Created On: 2018/01/15 1時44分54秒 GMT, and a note indicating "One or more update(s) available". The red button at the bottom contains the Japanese text "作成したアプリをクリック" (Click the created app).

作成したアプリをクリック

Application Container Cloud と Database Cloud のバインド

The screenshot shows the Oracle Application Container Cloud (ACM) interface. On the left, a sidebar lists sections: Overview (1 Instances), Deployments (1 Service Bindings, 0 User-defined Variables), Resources, Administration (0 Updates Available, 2018/01/21 3時06分02秒 GMT, 0 Logs). The main area is the 'Summary' card, which displays 1 Instance, 1 Memory (GB), and an Average Memory Usage of 12.8%. Below the card, deployment details are shown: Current Version: 1.0, Runtime: Node 8.1.4 (v8.1.4), Entitlement: 575053509, Security IP List: /Compute-jptest01/kaoru.hashino@or..., Last Deployed On: 2018/01/21 3時06分02秒 GMT, Type: web, Region: AP005_Z11, Application Cache: botCache. A red arrow points from the text '「Deployments」の選択' to the 'Deployments' section in the sidebar.

「Deployments」の選択

Application Container Cloud と Database Cloud のバインド Deployments 画面

The screenshot shows the Oracle Application Container Cloud Deployment page for the 'botSample' application. The left sidebar includes sections for Overview, Deployments (selected), Topology, Service Bindings, Environment Variables, Administration, and Logs. The main content area displays the Deployed Application details (Current Version: 1.0, Last Deployed On: 2018/01/21 3時06分02秒 GMT, Source: ORACLE, Archive Size: 8.14 MB) and Runtime details (Launch Command: node server.js, Node Version: 8). A red callout bubble points to the 'Add' button in the Service Bindings section, which lists one entry: Application Cache named 'botCache'. The bottom section shows Environment Variables with a note about predefined variables like APP_HOME.

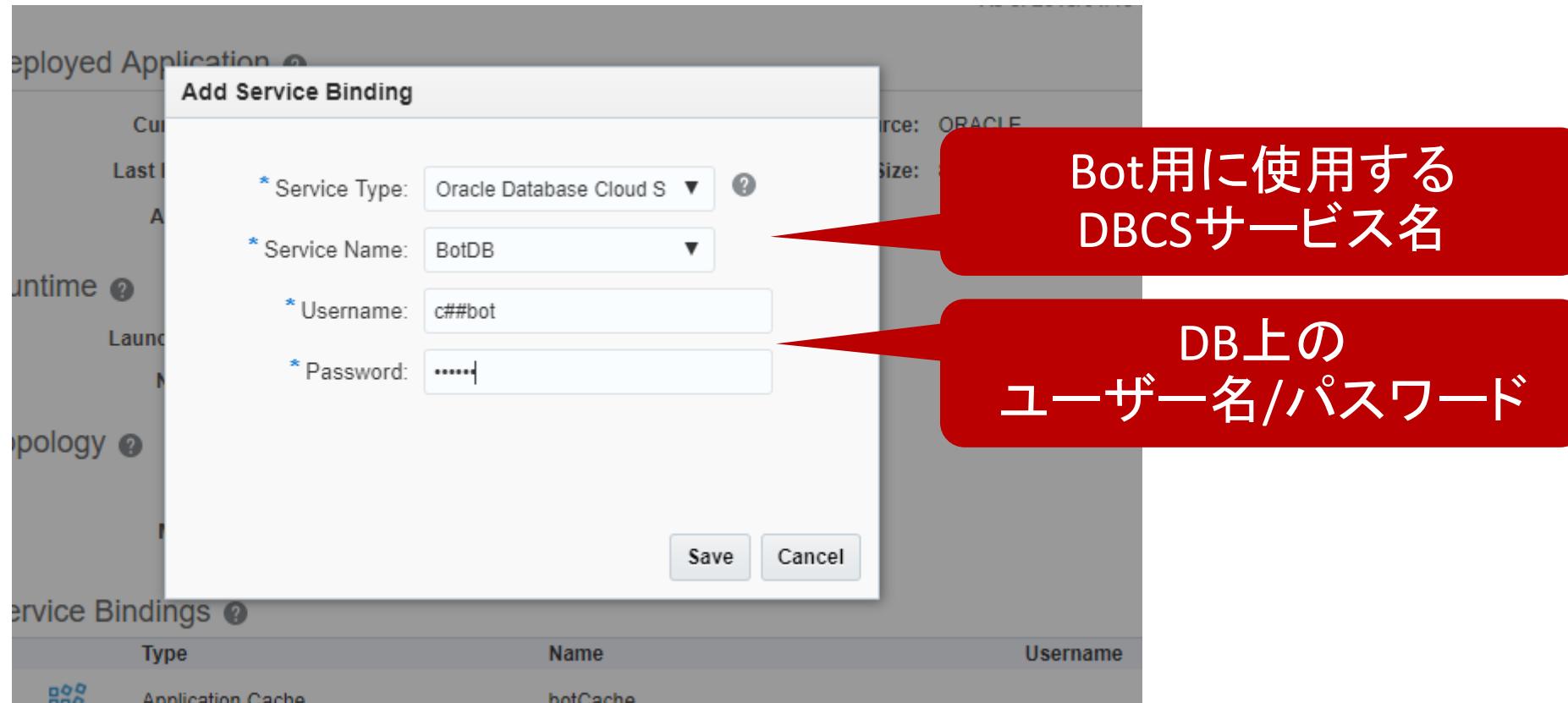
Service Bindings の
「Add」ボタンを押す

Type	Name	Username	Actions
Application Cache	botCache		

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. | 217

Application Container Cloud と Database Cloudのバインド

Add Service Binding



Application Container Cloud と Database Cloudのバインド変更の適用

The screenshot shows the Oracle Application Container Cloud interface for an application named 'botSample'. The 'Deployments' section is highlighted with a blue border. Within this section, the 'Service Bindings' tab is also highlighted with a blue border. Other tabs visible include 'Overview', 'Runtime', 'Topology', and 'Administration'. A message at the top states: 'Changes Not Saved. Apply Edits to save changes. Your application will restart.' with 'Apply Edits' and 'Cancel' buttons. The 'Service Bindings' table lists two entries:

Type	Name	Username	Actions
Application Cache	botCache		
Oracle Database Cloud Service	BotDB	c##bot	

「Apply Edits」で
変更の適用

Application Container Cloud と Database Cloudのバインド バインド後の確認

Service Bindings

Type	Name	Username	Actions
Application Cache	botCache		
Oracle Database Cloud Service	BotDB	kaoru.hashino@oracle.com	

Environment Variables

Your application can access these environment variables at runtime. You cannot modify predefined variables such as APP_HOME. To change an environment variable for another cloud service, edit the service binding. You can add or update custom environment variables.

Name	Value	Actions
(x) APP_HOME		
(x) PORT		
CACHING_USER_NAME		
CACHING_USER_PASSWORD	*****	
CACHING_INTERNAL_CACHE_URL	botCache-acc	

ページ 1 /3 (1-5/12個のアイテム) | | K < 1 2 3 > K

Application CacheとDatabase Cloud Serviceがバインドされた

Custom Environment Variables

You can add or update custom environment variables such as APP_HOME. To change an environment variable for another cloud service, edit the service binding. You can add or update custom environment variables.

Name	Value	Actions
DBAAS_DEFAULT_CONNECT_DESCRIPTOR	146.56.1.237:1521/PDB1.jptest01.oraclecloud.internl	
DBAAS_USER_NAME	c##bot	
DBAAS_USER_PASSWORD	*****	
DBAAS_LISTENER_HOST_NAME		
DBAAS_LISTENER_PORT	1521	

ページ 2 /3 (6-10/12個のアイテム) | K < 1 2 3 > K

バインドされた
Application Cacheの設定

バインドされたDatabase Cloud Serviceの設定

4 サンプル・アプリケーションを使ってみよう

4-4

サンプル・アプリケーションのクラウドへのデプロイ

4-4-1

Oracle Database Cloudのインスタンスを作成する

4-4-2

Oracle Database Cloudへの接続設定を行う

4-4-3

Oracle Database Cloudにスキーマを作成する

4-4-4

Application Cacheを作成する

4-4-5

Application Container Cloudにアプリをデプロイする

4-4-6

Application Container Cloud と Database Cloudのバインド

4-4-7

LINEへのURLの反映

LINEへの追加設定

メッセージ送受信設定

アクセストークン
(ロングターム) [?](#)
osnUf53H6FVKjSyJAeaDOL
eHZA0eXlabhJ4rz33aBD2Rz

再発行

Webhook送信 [?](#)

利用する

Webhook URL *SSLのみ対応 [?](#)

https://botSample-jptest01.apaas.ap5.oraclecloud.com/webhook

更新

キャンセル

Botのグループトーク参加 [?](#)

利用しない

LINE@機能の利用

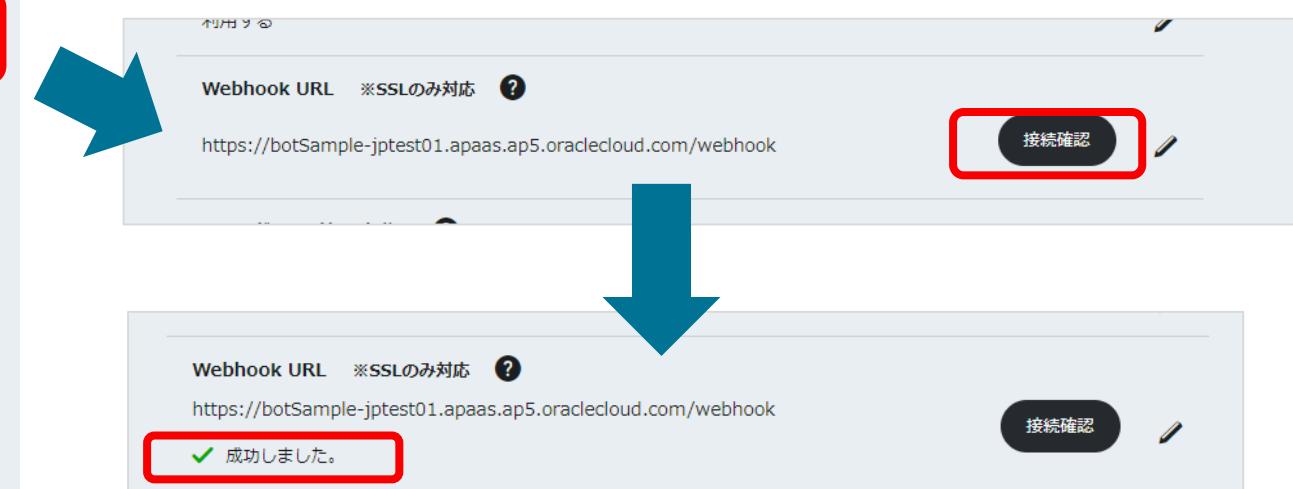
メッセージ本文はLINE@Managerの設定画面にて設定することができます。

自動応答メッセージ [?](#)

利用しない

友だち追加時あいさつ [?](#)

Application Container
Cloud上の
Webhook URLを指定し、
接続確認を行う



(注意) LINEのセキュリティ管理設定をしないこと

The screenshot shows the LINE Platform's security management interface. On the left, there's a sidebar with user profiles (Zabele, a child with sunglasses), a BotTest entry, and links for Channel基本設定 and セキュリティ管理. The main area is titled "セキュリティ管理" and contains instructions about managing communication servers. It features a table with a single row: "IP アドレス" with a question mark icon, a "+" button, and a "追加" (Add) button. A message at the bottom states "まだ登録されているIPアドレスがありません。". An orange callout bubble on the right contains the text "ここはIPアドレスを直接指定するようになっている".

セキュリティ管理

LINE Platformとの通信可能なサーバーを管理します。通信可能なサーバーを制限したい場合（初期設定では無制限）ご指定のサーバーのIPアドレスを追加してください。

IP アドレス ?

+ 追加

まだ登録されているIPアドレスがありません。

ここはIPアドレスを直接指定するようになっている

Application Container Cloudは動的グローバルIPアドレス
セキュリティ管理にIPアドレスを指定すると、
再起動で違うIPアドレスが割り当てられるために通信できなくなる

Integrated Cloud Applications & Platform Services

ORACLE®