

# Oracle Developer Cloud Serviceを使用した Oracle FormsでのDevOps

アプリケーション・デプロイメントの自動化

Oracleホワイト・ペーパー | 2018年11月

## 免責事項

下記事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。マテリアルやコード、機能の提供をコミットメント（確約）するものではなく、購買を決定する際の判断材料になさらないで下さい。オラクルの製品に関して記載されている機能の開発、リリース、および時期については、弊社の裁量により決定されます。

## 目次

免責事項.....	1
はじめに.....	1
概要.....	2
Oracle Developer Cloud Service .....	3
Forms Application Deployment Services .....	4
セットアップ.....	5
環境およびセットアップ .....	5
Git および DCS の初期セットアップ .....	6
新しい Forms DCS プロジェクトの作成 .....	8
初回使用前の新しいプロジェクトのセットアップ.....	10
ローカルの Git リポジトリと DCS プロジェクトの Git リポジトリの接続.....	12
Forms アプリケーションと Git および DCS との統合 .....	13
新しリポジトリへの Forms アプリケーションの挿入 .....	13
Developer Cloud Service の使用開始.....	15
ジョブの手動実行.....	15
スケジュールに基づいたジョブの実行 .....	16
パイプラインの作成 .....	16
パイプラインを実行する時刻のスケジュール設定 .....	18
デフォルト・パラメータ値の設定 .....	20
コード変更が検出された場合のジョブの実行.....	21
参考 .....	23
結論 .....	24

## はじめに

Oracle Forms は 1980 年代にまで遡るアプリケーション開発テクノロジーです。それ以来、キャラクタ・モードのアプリケーションを作成するためのツールから、Web アーキテクチャにデプロイするアプリケーションを作成するためのツールへと進化してきました。時間の経過とともに、このテクノロジーを使用して開発されるアプリケーションのサイズと複雑さが大幅に増えています。今日、これらの複雑なエンタープライズ・アプリケーションが世界中のサーバーや数十万ものエンドユーザーにデプロイされています。このような複雑かつ重要なアプリケーションでは、アプリケーションの安定性とセキュリティを確保してアプリケーションを最新の状態に保つために、メンテナンスを簡単、迅速、低リスクで行えることが非常に重要になっています。この領域で、最新の DevOps が役立ちます。

アプリケーション開発がより"俊敏な"開発およびデプロイメント・モデルへと積極的に移行している傾向の中で、この方法で Forms アプリケーションを管理するためには最新のツールが必要です。Developer Cloud Service はこのようなソリューションを提供します。

このホワイト・ペーパーは、Oracle Developer Cloud Service および Oracle Cloud Infrastructure がバージョン管理の簡素化とアプリケーション・デプロイメントの自動化にいかに関与するかを説明することを目的としています。このような自動化は、Forms アプリケーションのメンテナンスに伴う作業やリスクを軽減するのに役立ちます。

簡単に言うと、このホワイト・ペーパーは最新の DevOps 環境における Oracle Forms の入門書です。

## 概要

技術的な運用では、手順を人間が実行するたびに、エラーや障害が生じるリスクが増します。手順や標準の誤解や誤字をはじめとする数多くの要因のために、多くの場合、望ましくない結果が生じています。このような問題は Oracle Forms に限ったものではなく、すべてのテクノロジーに存在します。プロセスを自動化すると、エラーや障害のリスクが大幅に軽減されます。これは、自動化されたプロセス内に一連の固定の命令と標準が含まれているためです。予測どおりの結果が得られる場合が増え、効率性も向上します。

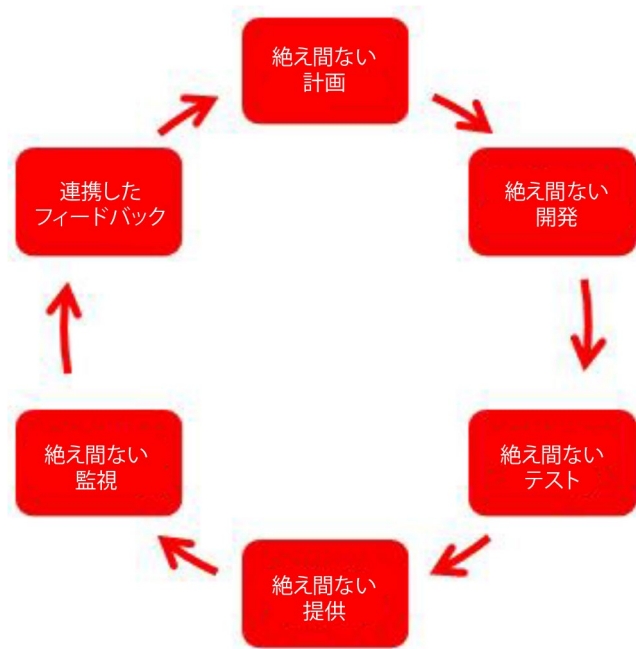
"DevOps"の概念は、変更の展開を迅速化する取組みとして絶え間ない開発と絶え間ない提供におもに焦点を置いています。DevOps は本質的には前述のことを達成します。アプリケーションの変更を自動的に処理、テスト、提供できる場合は、アプリケーションの更新をさらに頻繁に提供できるだけでなく、リスクを低く抑えながら提供できます。十分に確立された DevOps 計画には予測可能なリリース・サイクルも含まれるため、問題が生じた場合でも、自動化された次のリリース・サイクルに修正プログラムを含めることができます。言うまでもなく、リリース・サイクル外での更新も常に可能です。

この自動化されたアプリケーション更新/メンテナンス方式に Oracle Forms を取り入れるには、Developer Cloud Service (DCS) および Forms Application Deployment Services (FADS) の使用が必要です。DCS および FADS の概要については、以降のセクションで説明します。詳細については、このホワイト・ペーパーの最後に紹介しているドキュメント一覧を参照してください。

## Oracle Developer Cloud Service

Oracle Developer Cloud Service は、最新のさまざまな DevOps 機能を 1 つのインタフェースで提供する Oracle Cloud サービスです。

DevOps では、最初にアプリケーションを計画し、最後にサービス全体にアプリケーションを展開して、このサイクルを再度開始します。DCS はこのプロセスのフェーズを 1 つのインタフェースにするため、プロセスの合理化と効率化が促進されます。また、自動化された DevOps サイクルを作成するために、これらの複数のプロセスをまとめて関連付ける作業も簡単に行えます。



DCS が提供する機能には、以下のものが含まれます。

- プロジェクト構成およびユーザー管理
- Git および Maven を使用したソース管理およびアーティファクト・リポジトリ
- タスク、バグ、拡張機能を追跡するための問題追跡システム
- 開発スプリントの計画、追跡、およびレポート
- コード・レビューのプラットフォーム
- 複数のビルド・フレームワークをサポートするビルド自動化
- 絶え間ない統合およびデプロイメント
- Wiki によるドキュメント・コラボレーション
- チーム・アクティビティ・ストリーム

すべての機能があらゆるユースケースに役立つとは限りませんが、役立つ場合は、大幅な手動作業や個別のツールなしでは以前は不可能であった機能を利用できます。

DCSの詳細については、OracleのWebサイト ([https://cloud.oracle.com/ja\\_JP/developer-service](https://cloud.oracle.com/ja_JP/developer-service)) を参照してください。

## Forms Application Deployment Services

バージョン 12.2.1.3 で導入された Forms Application Deployment Services を使用すると、管理者または開発者は簡単な手順をいくつか実行するだけで、Forms アプリケーションをパッケージ化して、アプリケーションのアーカイブ・コピーをデプロイ、構成、および格納できます。FADS Web インタフェースを使用して、デプロイメント・プロセスのステータスの監視、デプロイメントの更新の簡単な提供、不要になったアプリケーションの削除などの操作を実行できます。

The screenshot displays the Oracle Forms Application Deployment Services (FADS) web interface. The top navigation bar includes the Oracle logo and the text "Forms Application Deployment Services", along with "Dashboard" and "Archives" buttons. The main content area is divided into three sections:

- Applications:** A table listing applications with their status. The "sales" application is "In Progress", "finance" is "Deployed", and "hr" is "Failed".
- Deploy:** A section for configuring deployment, including fields for "WLS Servers", "Username", "Password", and "Database". It also has an "Archive" field with a "Browse..." button and "No file selected." text. "Deploy" and "Reset" buttons are present.
- Contents:** A summary of application components with counts: 6 FMBs, 1 MMBs, 0 OLBs, 0 PLLs, 0 User Exits, 1 SQL Files, 0 Client Files, 0 Server JARs, and 0 Misc. Files. Below this is the text "Last Modified 17 Sep 2018".

コマンドライン・インタフェースも使用でき、Web インタフェースを使用できない Forms ビルド統合や自動デプロイメント・ジョブのスクリプト化を行う場合に役立ちます。

DCS で FADS コマンドライン・インタフェース (FADS-CLI) を使用して、パッケージ化およびデプロイメント機能を自動化できます。Forms コンパイラ/ジェネレータを使用して、自動ビルド・テストを実行できます。これらの両方の Forms ユーティリティは、カスタム・プロジェクトで使用する VM テンプレートの作成時に使用する、特別に設計されたソフトウェア・パッケージ (DCS 組込み) で提供されています。

---

Developer Cloud Service からのデプロイが必要な場合、完全な Forms 12.2.1.3 インストールが適切に構成、パッチ適用、実行されており、HTTP/S リクエストを使用して Developer Cloud Service からアクセスできる必要があります。製品のインストール・ガイド、システム要件ガイド、およびリリース・ノートを十分に確認し、インストールの構成とパッチ適用が適切に行われていることを確認してください。オンプレミスのインストールまたは Oracle Cloud Infrastructure (別名 Compute) 内のインストールを Forms 12.2.1.3 環境として使用できます。

---

FADS の詳細については、Forms のデプロイメント・ガイド (<https://docs.oracle.com/middleware/12213/formsandreports/deploy-forms>) を参照してください。

## セットアップ

DCS の使用手順については、[DCS](#) のドキュメントを参照してください。このドキュメントには、サード・パーティ・バージョン制御テクノロジーである [Git](#) への参照が含まれています。[Git](#) の使用方法については、[Git](#) のドキュメントを参照してください。

---

DCS はクラウド・サービスであるため、更新がよく行われます。そのため、このホワイト・ペーパーで示しているスクリーンショットおよび手順が、タスクを実行する際に必要となるスクリーンショットおよび手順と異なる場合があります。このホワイト・ペーパーは、DCS で Forms を使用するための総合ガイドとしてのみ使用し、使用に関する最新情報については、最新の DCS ドキュメントを参照してください。

---

## 環境およびセットアップ

DevOps が成功している要因として、アプリケーションの変更を追跡できることが挙げられます。これを行うには、アプリケーションのソース・ファイルをバージョン管理システムに含める必要があります。Git はこのために使用します。Git は開発者の環境で使用されるバージョン管理ユーティリティであり、ローカル・システムで行われたアプリケーションの変更が DCS 内に保存されているアプリケーションと確実に同期されるようにします。

このホワイト・ペーパーでは、Git のセットアップ方法や使用方法については詳しく説明しませんが、Forms で Git を使用する場合、および Forms で Git を使用する状況で DCS を使用する場合に関連する基本手順について説明します。

Git の詳細、使用方法、およびダウンロード方法については、以下のページを参照してください。

» <https://git-scm.com/>

» <https://git-scm.com/book>

ほとんどの Unix/Linux マシンでは、Git は事前にインストールされています。事前にインストールされていない場合は、インストールする必要があります。Microsoft 環境では、Git for Windows をダウンロードしてインストールします。ダウンロード場所については、次の Git のサイトを参照してください。

» <https://git-scm.com/downloads>

## Git および DCS の初期セットアップ

以下に、Microsoft Windows で Git 環境をセットアップするための基本的な手順をいくつか示します。これらの手順は他のプラットフォームの場合と同様ですが、ディレクトリへの参照と参照の修飾パスが異なります。以下の手順では、Git をすでにダウンロードしてインストールしていることを前提としています。

**手順 1:** 開発者のマシンで、Git シェルを開いて、ssh-agent が実行中であることを確認します。次のコマンドを実行します。

```
eval $(ssh-agent -s)
```

次のレスポンスが返されます。

```
Agent pid <SOME NUMBER>
```

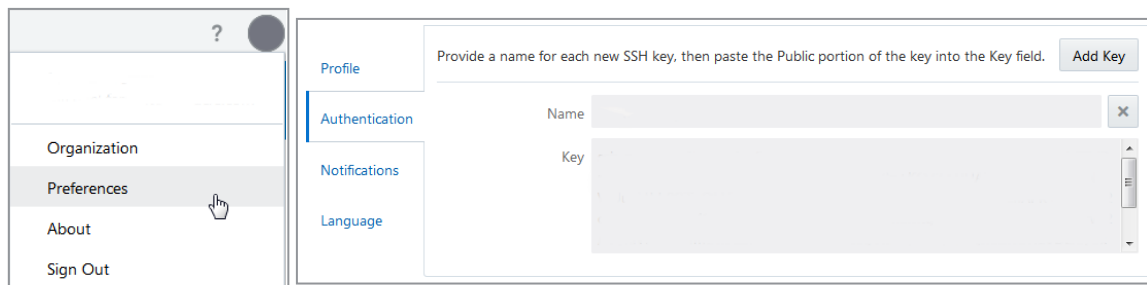
**手順 2:** Git SSH 鍵ペアを生成します。通常、この手順は、初期インストールの後で、または鍵タイプかユーザー名の変更後に 1 回のみ実行する必要があります。新しい鍵ファイルがユーザーのホーム・ディレクトリのサブディレクトリ `.ssh` に保存されます。次のコマンドを実行します。

```
ssh-keygen -t rsa -b 2048 -C "<ADD YOUR EMAIL ADDRESS HERE>"
```

**手順 3:** 新しく作成した秘密鍵を Git 構成に追加します。次のコマンドを実行します（Microsoft Windows 環境を前提としています）。

```
ssh-add /c/Users/<USER NAME>/.ssh/id_rsa
```

**手順 4:** DCS 環境にアクセスし、新しく作成した公開鍵のテキスト（`id_rsa.pub` など）をユーザーの DCS プロファイルに追加します。このテキストは、クラウド環境にローカルの Git がアクセスするために使用されます。DCS で、「User Preferences」にアクセスし、「Authentication」タブを選択して「Add Key」をクリックします。この公開鍵に適切な名前を指定し、公開鍵ファイルの内容を Key フィールドに貼り付けます。「Create」をクリックしてこの追加をコミットします。

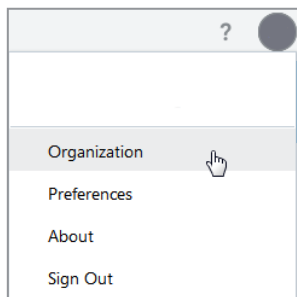


---

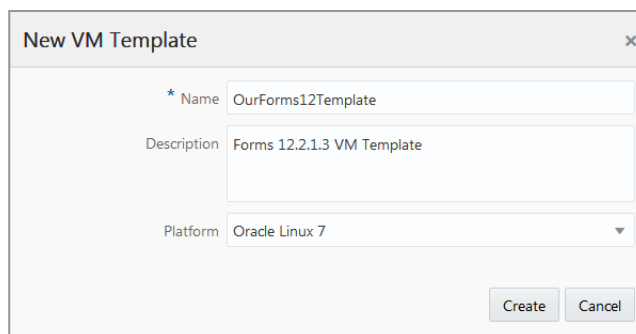
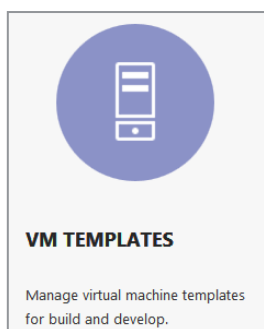
手順 5~8 には、DCS 組織管理者の権限が必要です。

---

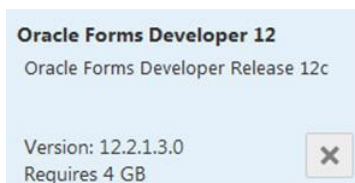
**手順 5 :** DCS 画面の右上にあるバブルをクリックして、Organization 構成画面にアクセスします。



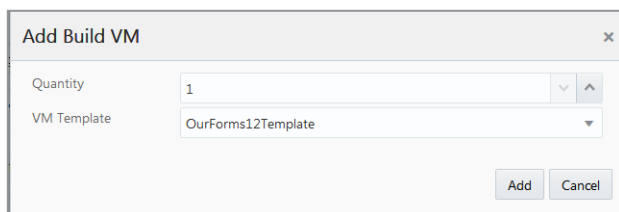
**手順 6 :** 新しい VM テンプレートを作成します。この VM テンプレートには、Forms モジュールの生成、Forms xml ファイルのバイナリへの変換、DCS からターゲット・サーバーへのアプリケーションのデプロイなどのタスクを実行するために必要となる Forms ソフトウェアが含まれます。適切な名前と説明を入力し、テスト・ビルドを実行する適切なプラットフォームを選択して、「Create」をクリックします。Forms アプリケーション生成プロセスは基本的にアプリケーションの PL/SQL の検証であり、すべてのプラットフォームで同じであるため、どのプラットフォームを選択するかは重要ではありませんが、Oracle Forms での使用が認定されているプラットフォームである必要があります。



**手順 7 :** このテンプレートの「Configure Software」ボタンをクリックします。リストから「Oracle Forms Developer 12」を選択して、「Done」をクリックします。



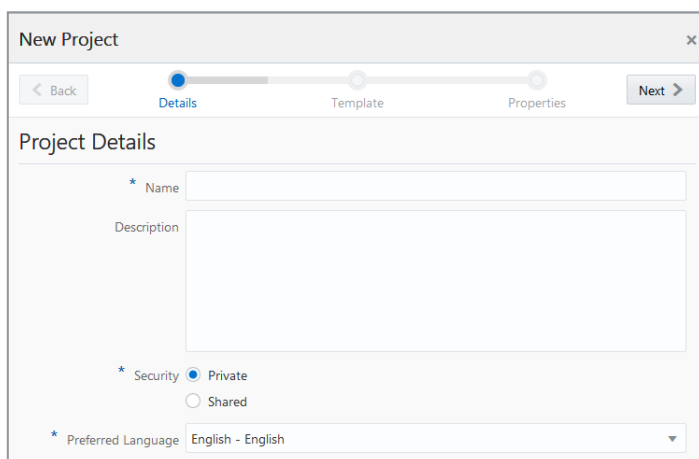
**手順 8:** 「*Organization*」 ブレッドクラムをクリックして *Organization* 画面に戻ります。「*Virtual Machine*」をクリックして *Virtual Machine* 画面にアクセスします。*Virtual Machine* 画面で、「*New VM*」 ボタンをクリックして新しい仮想マシンを作成します。ドロップダウン・リストで、前の手順の *VM* テンプレートを選択して、「*Add*」をクリックします。



The screenshot shows a dialog box titled "Add Build VM" with a close button (X) in the top right corner. It contains two input fields: "Quantity" with a value of "1" and a spinner control, and "VM Template" with a dropdown menu showing "OurForms12Template". At the bottom right, there are two buttons: "Add" and "Cancel".

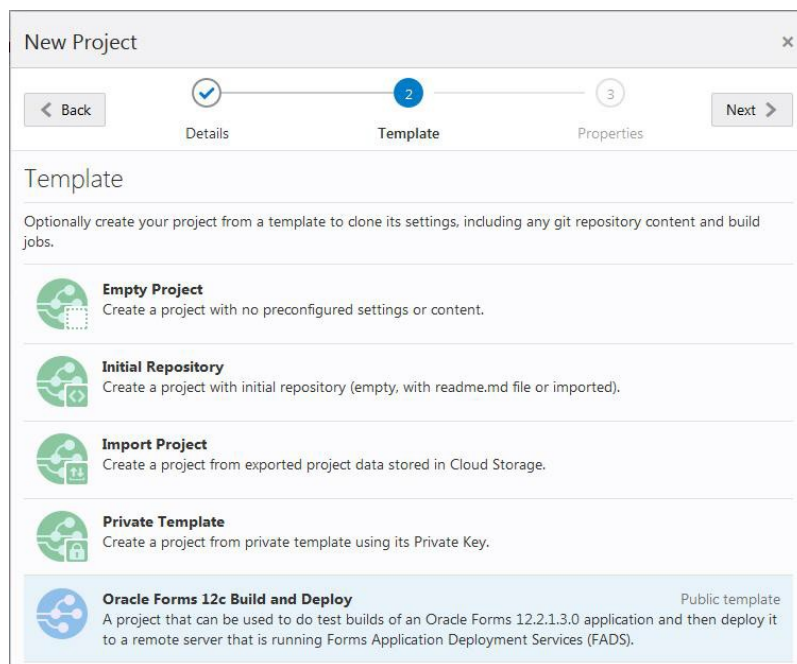
## 新しいForms DCSプロジェクトの作成

**手順 1:** *DCS* で、「*New Project*」 ボタンをクリックし、*New Project* ウィザードを表示します。適切な名前と説明を入力して、「*Next*」をクリックします。

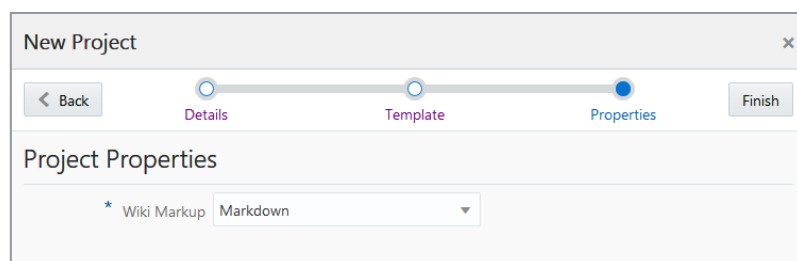


The screenshot shows a wizard titled "New Project" with a close button (X) in the top right corner. At the top, there is a progress bar with four steps: "Back", "Details", "Template", and "Properties". The "Details" step is currently active. Below the progress bar, there are four input fields: "Name" (required, marked with an asterisk), "Description", "Security" (with radio buttons for "Private" and "Shared", where "Private" is selected), and "Preferred Language" (required, marked with an asterisk, with a dropdown menu showing "English - English"). At the top right, there is a "Next" button with a right-pointing arrow.

**手順 2 :** Template 画面で、「Oracle Forms 12c Build and Deploy」テンプレートを選択します。「Next」をクリックします。（一部の地域では Forms テンプレート・プロジェクトが利用できない場合があります）



**手順 3 :** Wiki Markup で「Markdown」を選択し、「Finish」をクリックしてプロジェクトを作成します。



新しいプロジェクトの作成にしばらく時間がかかります。プロジェクトが作成されたら、プロジェクトを使用する前に、作成後の構成手順をいくつか実行する必要があります。

## 初回使用前の新しいプロジェクトのセットアップ

**手順 1:** プロジェクト・ナビゲータ・サイド・パネルが表示されない場合（左側）、左上の Developer Cloud Service 名の横にあるハンバーガー・ボタンをクリックします。



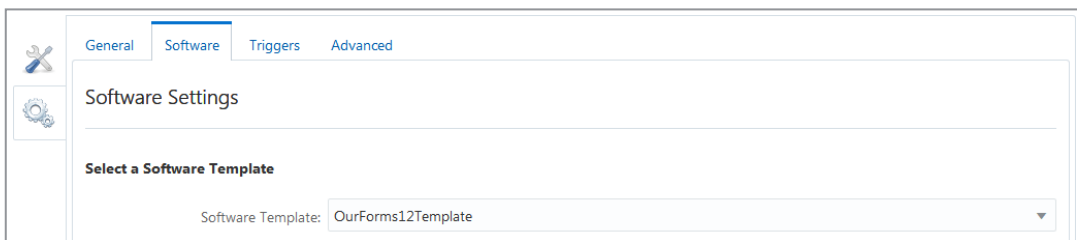
**手順 2:** 「Build」をクリックして、このプロジェクト・テンプレートで提供されているビルド・ジョブにアクセスします。



**手順 3:** Compile\_and\_Package ジョブの「Configure」ボタンをクリックします。

Status	Weather	Job	Last Success	Last Unsuccess	Duration	Actions
		Compile_and_Package	N/A	N/A	< 1 Sec	
		Deploy_FAR_file	N/A	N/A	< 1 Sec	

**手順 4:** プロジェクトの初回入力時に、Software Configuration タブが自動的に表示される場合があります。表示されない場合は、左側の垂直タブ・バーにある「Settings」歯車アイコン/タブをクリックします。「Software」タブを選択します。「Software Template」ドロップダウン・フィールドで、初期セットアップ手順で作成した VM テンプレートを選択します。



**手順 5:** 左側の垂直タブ・バーにあるツール・アイコン/タブをクリックします。「Source Control」タブを選択します。「Add Source Control」をクリックして「Git」を選択します。Repository ドロップダウンが表示されます。ドロップダウンで、使用するプロジェクト名が含まれており末尾が.gitのエントリを選択します。「Save」をクリックします。



**手順 6:** プロジェクト・ナビゲータで、「Build」を再度クリックしてジョブ・リストに戻ります。Deploy\_FAR\_File ジョブで、「Configure」ボタンをクリックします。Deploy\_FAR\_File ジョブに対して、上記の手順 4 を繰り返します。

Status	Weather	Job	Last Success	Last Unsuccess	Duration	Actions
		Compile_and_Package	N/A	N/A	< 1 Sec	
		Deploy_FAR_file	N/A	N/A	< 1 Sec	 <b>Configure</b>

これらのジョブをカスタマイズすることを予定している場合は、これらのジョブをコピーして、これらのコピーを以降の手順およびカスタマイズに使用することを推奨します。ジョブをコピーするには、変更を加える前に、「New Job」ボタンをクリックし、「Copy existing job」オプションを使用して2つのジョブをそれぞれコピーします。

既存のジョブのコピーとして新しいジョブを作成した場合、コピーには元のジョブへの参照が含まれるため、Build Environment 設定を確認して適切に更新することが重要です。そのため、たとえば、「Compile\_and\_Package」からコピーして新しいジョブを作成し、新しいジョブに「My Compile and Package」という名前を指定する場合、「From job」の値を「Compile\_and\_Package」から新しいジョブ「My Compile and Package」に変更することが重要です。このようにしないと、アーティファクトが新しいジョブからではなく元のジョブから取得されます。

Jobs Overview > My Compile and Package > **Configure**

Job Configuration

Source Control   
 Build Parameters   
 **Build Environment**   
 Builders   
 Post Build

Configure Build Environment

**Copy Artifacts**

From job: My Compile and Package

Which build:

- Compile\_and\_Package
- Deploy\_FAR\_file
- My Compile and Package**

## ローカルのGitリポジトリとDCSプロジェクトのGitリポジトリの接続

この時点で初期セットアップは完了しており、DCS プロジェクトが作成されています。残りの手順では、ローカルの Git リポジトリを作成して DCS プロジェクトのリポジトリと接続します。

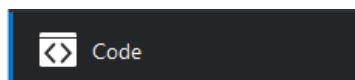
これらの手順では、既存のローカル Git リポジトリがまだ存在しないことを前提としています。既存のローカル・ディレクトリや既存のリポジトリを使用することもできますが、これらの手順ではその構成に対応していません。既存のローカル・リポジトリを使用する方法については、Git と DCS のドキュメントを参照してください。

**手順 1：**開発者のマシンで、アプリケーションのプロジェクト最上位ディレクトリを挿入する、必要な最上位ディレクトリを作成します。アプリケーション・モジュールは、後の手順で作成されるサブディレクトリに保存されます。たとえば、次のようになります：C:\Git\dcs\sales

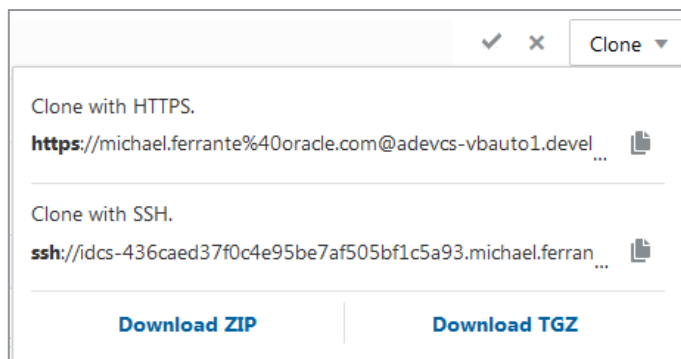
**手順 2：**Git シェルを開いて、ディレクトリ (cd) を上記の手順 1 で作成したディレクトリに変更します。

```
MINGW64 ~
$ cd /c/Git/dcs/sales
```

**手順 3：**Git ウィンドウを開いたままにし、DCS ブラウザ・ウィンドウに一時的に戻ります。プロジェクト・ナビゲータで、「Code」をクリックします。



**手順 4：**右側で、「Clone」ドロップダウンをクリックします。HTTPS または SSH を選択して、使用する URL の右側にある「Copy to clipboard」アイコンをクリックします。このプロジェクト用の DCS 内の Git の場所がコピーされます。この URL は次の手順で必要になります。ユーザーのマシンとインターネット間にプロキシ・サーバーを使用する場合、SSH ではなく HTTPS オプションを使用することを推奨します。



**手順 5 :** Git シェルに戻ります。次のコマンドを実行します。

```
git clone <PASTE THE CLIPBOARD CONTENTS HERE>
```

上記のコマンドが正常に実行されると、プロジェクト名を使用してサブディレクトリが作成され、この新しいサブディレクトリにプロジェクトの Code README.md ファイルがコピーされます。プロジェクト名を使用したこのサブディレクトリはローカル・リポジトリとして参照され、このプロジェクトのアプリケーション・ファイルが保存される場所としても参照されます。このローカル・リポジトリと DCS リポジトリが相互に関連付けられます。

```
MINGW64 /c/Git/dcs/sales/oracle-forms-12c-build-and-deploy (master)
$ ls
README.md
```

## FormsアプリケーションとGitおよびDCSとの統合

上記の手順でローカル Git リポジトリを構成し、新しい DCS プロジェクトを作成して、ローカル Git をリモート (DCS) Git に関連付けました。以降の手順を使用して、ファイルが変更されたらローカル・リポジトリとリモート・リポジトリを更新します。これらの手順では、リポジトリでファイルを追加/更新する方法の概要を示しています。変更のロールバック、変更の削除、コードの分岐については示していません。これらの高度なコマンドの使用方法については、Git のドキュメントを参照してください。


### 新しリポジトリへの Forms アプリケーションの挿入

Git および最終的に DCS を使用して Forms アプリケーション・ファイルを管理するためには、前述の手順で説明したリポジトリ・ディレクトリにアプリケーションに関連するすべてのファイルを作成またはコピーする必要があります。

**手順 1 :** 既存のすべてのアプリケーション・ファイルをこのプロジェクトのリポジトリ・ディレクトリにコピーします。

**手順 2 :** ローカル・リポジトリ・インデックスを更新するために、Git シェルで次のコマンドを実行します。これらのコマンドを実行する前に、リポジトリ・ディレクトリを開いている必要があります。このコマンドでは、変更を保留にするようにのみ Git に通知します。変更はまだコミットされていません。この例では、ワイルドカード (\*) を使用していますが、代わりに個々のファイル名を使用することもできます。他のファイルに対する追加の変更が見込まれる場合は、具体的なファイル名を使用すると便利です。

```
git add *
```



**手順 3：** 次のコマンドを実行して、上記の手順で更新をフラグ付けした変更をコミットします。このコマンドにコメントを含めることを強く推奨します。コメントを含めるには、以下に示すように `-m` スイッチを使用します。

```
git commit -m "変更の説明をここに入力"
```

**手順 4：** 上記の手順で、ローカル・リポジトリを更新しました。これらの変更を DCS と同期させるために、次のコマンドを実行します。以下の `"master"` は、コードのマスター・ブランチへの参照です。ブランチの使用を実装する場合、マスターへのこの参照が、変更を受け取る現在のブランチを代わりに参照します。

```
git push origin master
```

このリポジトリ内のファイルを更新/追加するたびに、上記の手順 2~4 を実行します。ファイルを適切に削除する方法については、Git のドキュメントを参照してください。見込まれているすべての変更が完了するまで、Git コマンドの実行を遅らせることを推奨します。

## Developer Cloud Serviceの使用開始

開発者のマシンを構成し、DCS に Forms アプリケーション用の新しいプロジェクトを作成し、2 つを相互に関連付けたので、DCS を機能させます。提供されているサンプル・プロジェクトは、ジョブを手動で実行するように設定されています。ただし、スケジュールに基づいてまたはコード変更の検出に基づいて、ジョブを自動的に実行させることも可能です。


### ジョブの手動実行

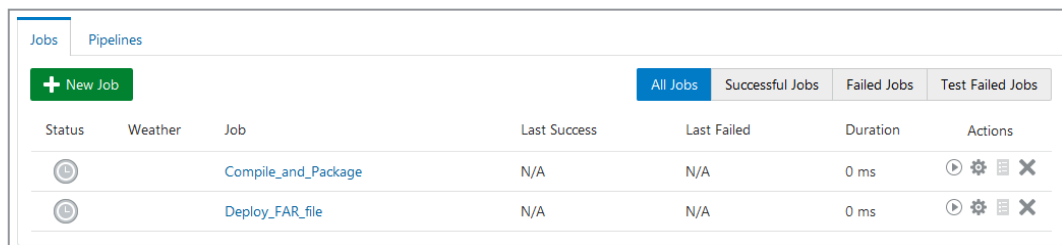
ビルド・ジョブの手動実行は一目瞭然であるため、ここでは詳しい説明は省いています。提供されているサンプル・ジョブを実行するために必要となるジョブのパラメータおよび値の詳細については、プロジェクトの Wiki の Readme を参照してください。また、Wiki の Readme に記載されているこのサンプル・プロジェクトの制限についても、確認して把握するようにしてください。











このサンプル・プロジェクトには 2 つのビルド・ジョブがあります。2 つ目のジョブ (“Deploy\_FAR\_File”) は 1 つ目のジョブ (“Compile\_and\_Package”) に依存します。つまり、2 つ目のジョブを実行する前に 1 つ目のジョブを実行する必要があります。1 つ目のジョブを実行せずに 2 つ目のジョブを実行すると、“Compile\_and\_Package”の実行が最後に成功した以前の状態でアプリケーションが再デプロイされます。1 つ目のジョブを実行していない場合や、実行が成功しなかった場合は、2 つ目のジョブは失敗します。

**手順 1:** プロジェクト・ナビゲータで「Build」をクリックして、Build ページを開きます。ナビゲータが表示されない場合は、ページ左上の Oracle ロゴの横にあるハンバーガー・アイコンをクリックします。



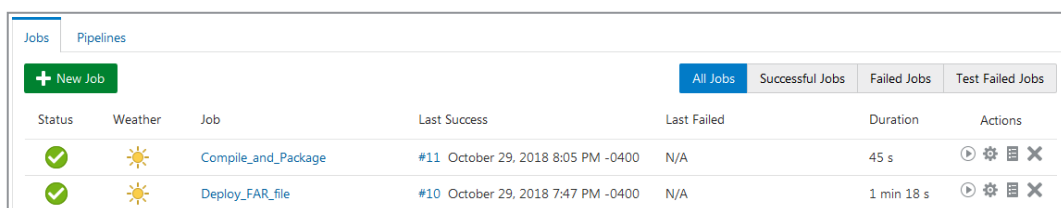
**手順 2:** “Compile\_and\_Package”の右側にある「Run」ボタン  をクリックします。パラメータ・フォームが表示されます。説明に従って空白箇所に入力し、パラメータ・フォームの下部にある「Build Now」ボタンをクリックします。前述したように、パラメータおよび有効な値の詳細については、Wiki の Readme を参照してください。



Jobs		Pipelines				
+ New Job		All Jobs	Successful Jobs	Failed Jobs	Test Failed Jobs	
Status	Weather	Job	Last Success	Last Failed	Duration	Actions
		Compile_and_Package	N/A	N/A	0 ms	   
		Deploy_FAR_file	N/A	N/A	0 ms	   

**手順 3:** “Compile\_and\_Package”の実行が成功していることを前提に、“Deploy\_FAR\_file”を実行できます。“Deploy\_FAR\_file”を実行するには、“Compile\_and\_Package”に使用したのと同じプロセスに従いますが、“Deploy\_FAR\_file”の「Run」ボタンを使用します。

両方のジョブの実行が成功した場合、緑のバブルの中央に白いチェックマークが付き（以下の図を参照）、ジョブの表に成功のステータスが反映されます。表の Weather 列には、以前に実行されて完了した、成功したジョブの平均が表示されます。各列の詳細については、DCS のドキュメントを参照してください。



Status	Weather	Job	Last Success	Last Failed	Duration	Actions
✓	☀️	Compile_and_Package	#11 October 29, 2018 8:05 PM -0400	N/A	45 s	🔍 ⚙️ 🗑️
✓	☀️	Deploy_FAR_file	#10 October 29, 2018 7:47 PM -0400	N/A	1 min 18 s	🔍 ⚙️ 🗑️

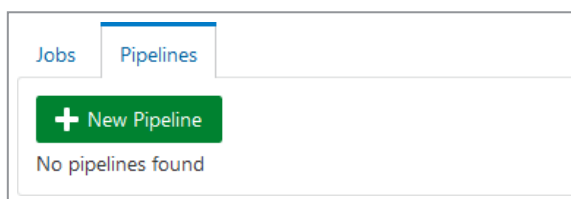
## スケジュールに基づいたジョブの実行

スケジュールに基づいて、または変更が検出された場合に、ジョブを自動的に実行するように構成できます。また、パイプラインを使用してジョブを相互に連鎖させることもできます。このセクションでは、パイプラインを作成して、1つ目のジョブが成功した場合に、1つ目のジョブの後に2つ目のジョブを実行するように両方のジョブをスケジュールする方法について説明します。パイプラインが望ましくない場合は、パイプラインの作成をスキップし、各ジョブを個別に処理することもできます。ただし、その場合は、デプロイメント・プロセスに関連する直接的なコードがないため、コード変更に基づいて“Deploy\_FAR\_file”ジョブをトリガーすることはできません。したがって、パイプラインが望ましくない場合、このジョブには、手動の実行か、スケジュールされた時刻のみを使用できます。

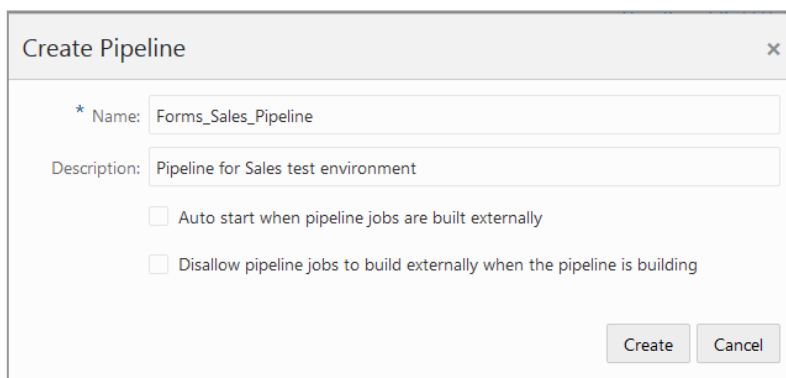
## パイプラインの作成

2つのジョブを相互に関連付けて1つの効率的な自動プロセスを作成するためには、まず、パイプラインを作成する必要があります。

**手順 1:** Build ページで、「Pipelines」タブをクリックします。「New Pipeline」をクリックします。



**手順 2:** パイプライン用の適切な名前と説明を入力します。ジョブを手動で開始する場合にパイプラインを実行しないようにするには、「Auto start when pipeline jobs are built externally」をオフにします。終了したら、「Create」をクリックします。



Create Pipeline

\* Name: Forms\_Sales\_Pipeline

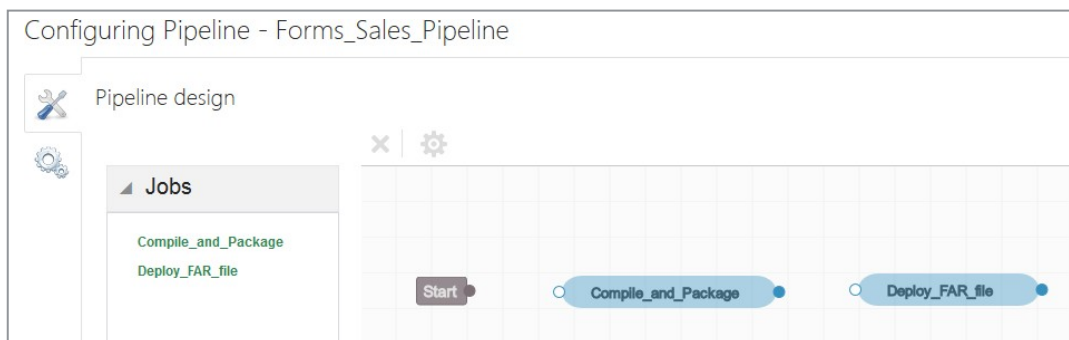
Description: Pipeline for Sales test environment

Auto start when pipeline jobs are built externally

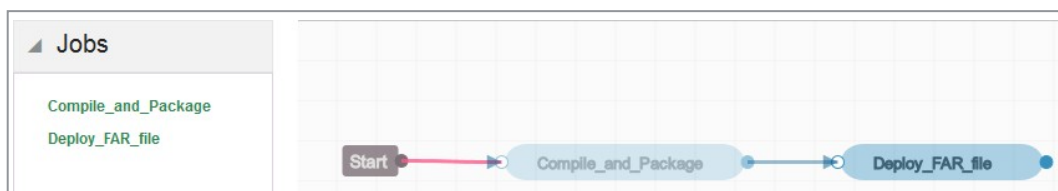
Disallow pipeline jobs to build externally when the pipeline is building

Create Cancel

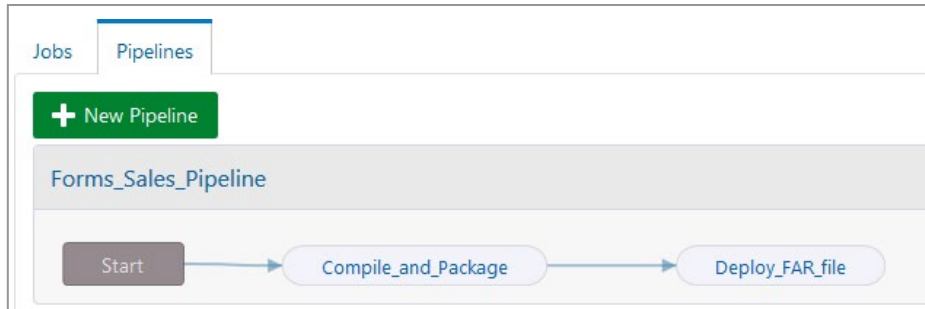
**手順 3:** マウスを使用して、Jobs ペインの「Compile\_and\_Package」ジョブを Start の横にある描画ボックスにドラッグします。その横に「Deploy\_FAR\_file」をドラッグします。



**手順 4:** マウスを使用して"Start"ボックスから接続ドット/ハンドルをドラッグし、各ポイントを次のポイントに接続します。これは、プロセスのフローを表します。デフォルトでは、各ジョブの実行は成功することを前提としています。“失敗”ケースを処理するジョブを追加することもできます。接続先のジョブに応じて、必要な場合は、コネクタ矢印をダブルクリックすると、結果の状態を“成功”から“失敗”または“テスト失敗”に変更できます。このサンプル・プロジェクトでは、失敗ケースを処理するジョブは指定されていません。終了したら、「Save」をクリックします。





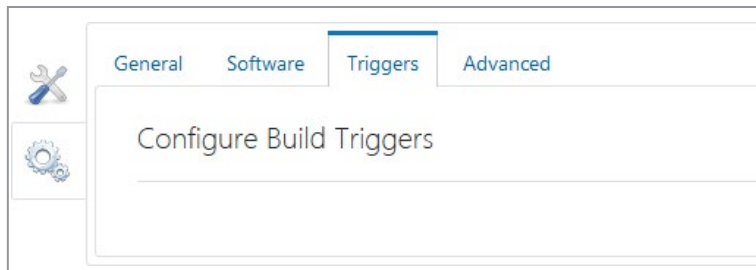
**手順 5 :** Pipelines 表に新しいパイプラインが表示されます。「Jobs」タブをクリックします。



#### パイプラインを実行する時刻のスケジュール設定

2 つの異なるタイプのスケジュールを設定できます。変更があったかどうかに関係なく実行される定期的なスケジュールと、その時間間隔内に変更があったかどうかを確認し、その時点で変更が検出された場合のみ実行されるスケジュールです。1 つ目のジョブが成功した場合は、パイプラインによって 2 つ目のジョブ (“Deploy\_FAR\_file”) が自動的に開始されるため、このスケジュールは “Compile\_and\_Package”ジョブにのみ作成する必要があります。

**手順 1 :** Jobs 表で、“Compile\_and\_Package”ジョブの構成歯車  を ク リ ッ ク し ま す 。 Configuration タブが表示されたら、垂直タブバーの歯車アイコン  を ク リ ッ ク し ま す 。 「Triggers」タブをクリックします。



**手順 2:** 「Triggers」タブの右側で、「Add Trigger」ドロップダウンをクリックして展開します。「Periodic Trigger」または「SCM Polling Trigger」を選択します。「Periodic Trigger」を選択すると、コード変更が行われたかどうかに関係なく、選択したスケジュールまたは時間間隔に基づいてジョブが実行されます。「SCM Polling Trigger」を選択すると、構成したスケジュールまたは時間間隔に基づいて変更がチェックされますが、変更が検出された場合にのみジョブが実行されます。どちらを選択するかに関係なく、表示されるスケジュール画面は同じです。



この例では、トリガーは 30 分ごとに実行されます。希望の時刻をスケジュールしたら、「Save」をクリックして変更をコミットします。

Schedules are set in UTC. To display the schedule in a different timezone, use 'View schedule' below.

Expert mode  0/30 \* \* \* \* #Every 30 minutes

Minute <input checked="" type="checkbox"/>	<input type="text" value="30"/>	<input type="button" value="Toggle Recurrence"/>
Hour <input type="checkbox"/>	<input type="text"/>	<input type="button" value="Toggle Recurrence"/>
Day of the Month <input type="checkbox"/>	<input type="text"/>	<input type="button" value="Toggle Recurrence"/>
Month <input type="checkbox"/>	<input type="text"/>	<input type="button" value="Toggle Recurrence"/>
Day of the Week <input type="checkbox"/>	<input type="text"/>	<input type="button" value="Toggle Recurrence"/>

Comment

(UTC-05:00) New York - Eastern Time (ET) ▼

## デフォルト・パラメータ値の設定

提供されているサンプル・ジョブのパイプラインをユーザー操作なしに実行するためには、すべてのジョブ・パラメータにデフォルト値を設定する必要があります。複数の固定選択（ドロップダウン）があるパラメータでは、リスト内の最初の値がデフォルト値として使用されます。提供されている2つのジョブごとに、以下の手順を実行します。

**手順 1：**ジョブの構成画面を開いて、"*Build Parameters*"タブに移動します。Default Value に、ページ上の各パラメータのデフォルト値を入力します。下方向にスクロールし、すべてのパラメータに対して入力します。終了したら、「Save」をクリックします。

The screenshot shows the 'Configure Build Parameters' interface with the following configurations:

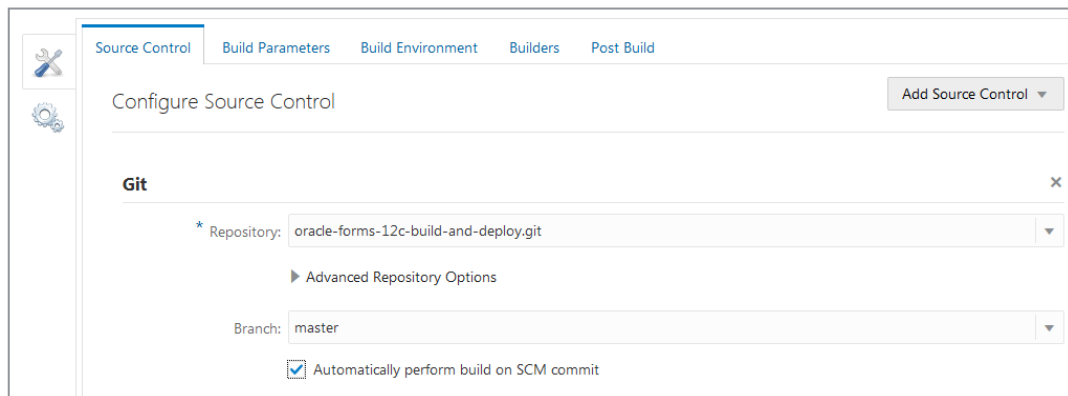
- Choice Parameter:**
  - Name: FILES\_TO\_BUILD
  - Choices: all, rebuild\_changed\_files, rebuild\_failed\_files, rebuild\_since\_success
  - Description: Determines which files should be built: all: Rebuild all files
- String Parameter:**
  - Name: APPLICATION\_NAME
  - Default Value: Sales
  - Description: If the build is successful and this parameter is provided, it will be used as the name of the .FAR file that will be generated for deployment. Leave this blank if no .FAR file is desired.
- String Parameter:**
  - Name: APPLICATION\_VERSION
  - Default Value: 1.0
  - Description: If a .FAR file is generated, this will be the application version number used.

## コード変更が検出された場合のジョブの実行

コード変更が検出された場合にジョブまたはパイプラインを実行するのは、通常はテスト環境の場合が望ましいですが、必要なすべての場合に可能です。コード変更が検出された場合にジョブを実行するには、以下の手順を実行します。

前述の手順を使用してスケジュールを作成した場合は、そのことを以下の手順を実行する際に念頭に置いてください。スケジュールされたトリガーと変更検出時の両方について、ジョブを実行したくない場合があります。両方について実行したくない場合は、以前に作成したスケジュールを削除します。

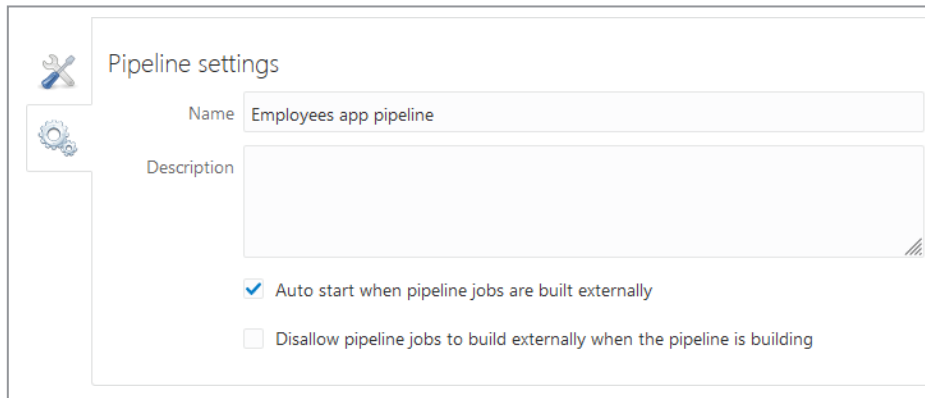
**手順 1：**ジョブの構成画面を開いて、"Source Control"タブに移動します。「*Automatically perform build on SCM commit*」をオンにします。「Save」をクリックして変更をコミットします。この手順は"Compile\_and\_Package"ジョブに対してのみ必要です。



The screenshot shows the 'Configure Source Control' dialog box with the following details:

- Tab: Source Control
- Buttons: Build Parameters, Build Environment, Builders, Post Build
- Section: Git (with a close button 'x')
- Repository: oracle-forms-12c-build-and-deploy.git
- Advanced Repository Options (expanded)
- Branch: master
- Checkbox:  Automatically perform build on SCM commit

**手順 2:** メインの Build ページに戻り、完全な"Jobs"表を表示します。「Pipelines」タブをクリックして、「Settings」サイドバー・タブをクリックします。「Auto start when pipeline jobs are built externally」をオンにします。

The image shows a screenshot of the 'Pipeline settings' interface. On the left, there is a sidebar with a gear icon. The main area is titled 'Pipeline settings' and contains a 'Name' field with the value 'Employees app pipeline' and a 'Description' field which is empty. Below these fields are two checkboxes: the first is checked and labeled 'Auto start when pipeline jobs are built externally', and the second is unchecked and labeled 'Disallow pipeline jobs to build externally when the pipeline is building'.

DCS コード・リポジトリに対してコード更新が行われると、パイプラインが自動的に実行されるようになります。コード変更が可能なのは、DCS で直接変更が行われた場合、またはローカル Git リポジトリから同期された場合です。

Forms テンプレート・プロジェクトに提供されているサンプル・ジョブを使用する方法については、プロジェクトの Wiki、特に含まれている Readme を参照してください。

## 参考

<https://git-scm.com>

<https://docs.oracle.com/en/cloud/paas/developer-cloud>

<https://docs.oracle.com/middleware/12213/formsandreports/deploy-forms/oracle-forms-application-deployment-services-fads.htm>

<https://docs.oracle.com/middleware/12213/formsandreports/releasenotes-fnr/known-issues-and-workarounds.htm#FRREL-GUID-EC385A18-F70F-4699-8BCE-E648544F3726>

## 結論

最新の DevOps を使用すると、アプリケーションの更新をさらに簡単かつ迅速に提供できます。Oracle Developer Cloud Service を Oracle Forms と併用すると、レガシー・アプリケーションで最新の DevOps を使用できます。計画を慎重に検討するとともに強力な DevOps ツールを使用すると、アプリケーションの更新、テスト、デプロイを簡単かつこれまで以上に迅速に行えます。定期的なバグ修正や、組織のユーザー・コミュニティのニーズに基づいた最新機能の提供により、アプリケーションを最新の状態に保つと、ユーザーの作業効率が最大限に高まるとともに、正確なデータの記録が可能になります。




**Oracle Corporation, World Headquarters**

500 Oracle Parkway  
Redwood Shores, CA 94065, USA

海外からのお問い合わせ窓口

電話：+1.650.506.7000  
ファクシミリ：+1.650.506.7200

CONNECT WITH US

 [blogs.oracle.com/oracle](https://blogs.oracle.com/oracle)

 [facebook.com/oracle](https://facebook.com/oracle)

 [twitter.com/oracle](https://twitter.com/oracle)

 [oracle.com](https://oracle.com)

## Integrated Cloud Applications & Platform Services

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. 本文書は情報提供のみを目的として提供されており、ここに記載される内容は予告なく変更されることがあります。本文書は、その内容に誤りがないことを保証するものではなく、また、口頭による明示的保証や法律による黙示的保証を含め、商品性ないし特定目的適合性に関する黙示的保証および条件などのいかなる保証および条件も提供するものではありません。オラクルは本文書に関するいかなる法的責任も明確に否認し、本文書によって直接的または間接的に確立される契約義務はないものとします。本文書はオラクルの書面による許可を前もって得ることなく、いかなる目的のためにも、電子または印刷を含むいかなる形式や手段によっても再作成または送信することはできません。

Oracle および Java は Oracle およびその子会社、関連会社の登録商標です。その他の名称はそれぞれの会社の商標です。Intel および Intel Xeon は Intel Corporation の商標または登録商標です。すべての SPARC 商標はライセンスに基づいて使用される SPARC International, Inc. の商標または登録商標です。AMD、Opteron、AMD ロゴおよび AMD Opteron ロゴは、Advanced Micro Devices の商標または登録商標です。UNIX は、The Open Group の登録商標です。1118

Oracle Developer Cloud Service による Oracle Forms とライフサイクル管理

2018 年 11 月

著者：Michael Ferrante 共著者：Phil Kuhn