AWS CDKとCodePipelineを利用したビルド/デプロイ

1. はじめに

- AWS CDK/CodePipeline+ECSを組み合わせることで、共通の構成管理ツール/コードベースでデプロイを制御できるようになる。

- AWS CDKでAWSアカウントにECSやCloudFrontなどのインフラをデプロイし、CodePipelineをつかってECSにコンテナを展開・稼動させることができるようになる。

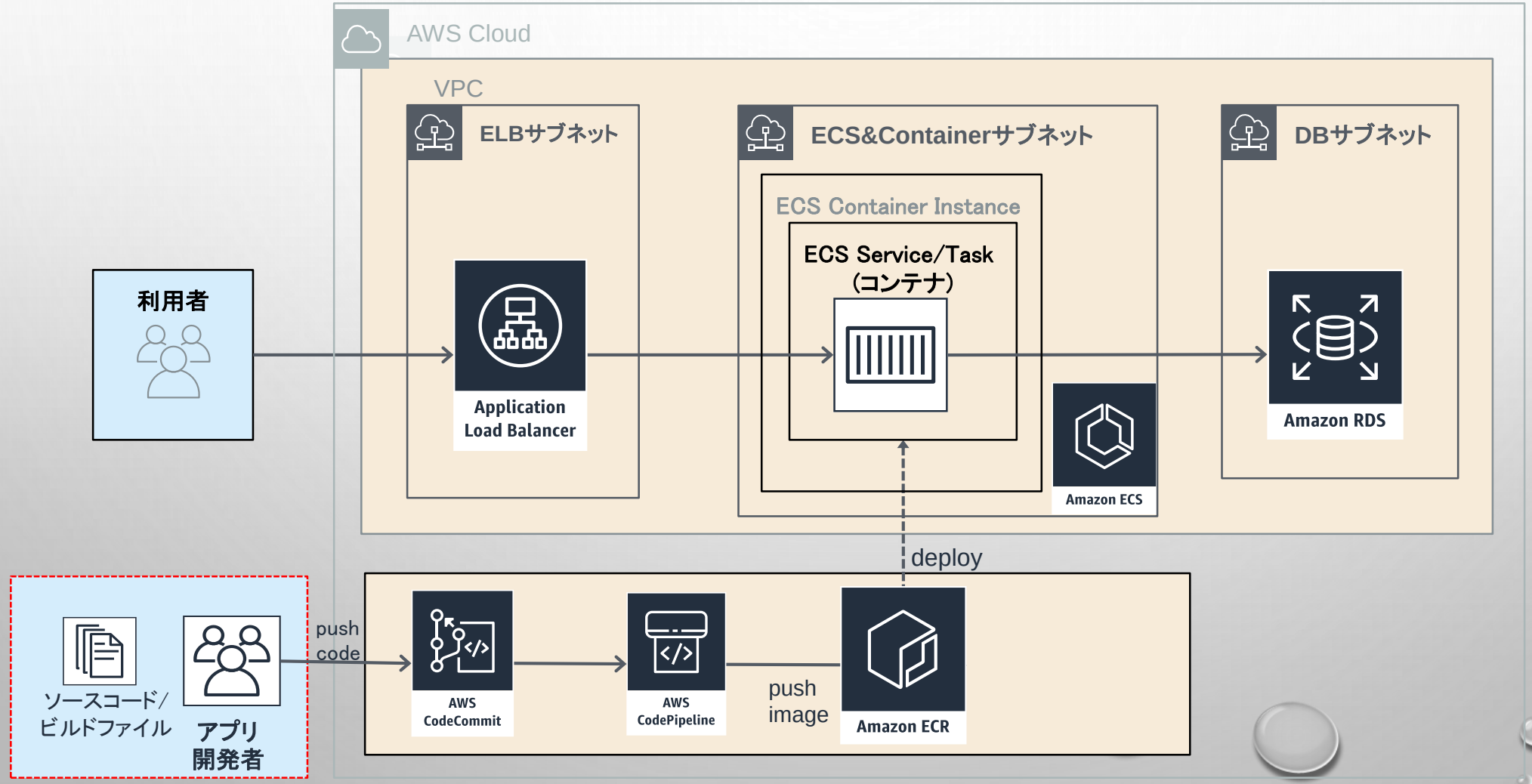
- AWS CDKではCloudFormationに対応したAWSリソースであればデプロイが可能のため、
  複数ツールを組み合わせることなく(または最小組み合わせで)、AWSのインフラ構成が可能になる。
  - ✓ AWSのインフラ構成管理は今までは様々ツールを組み合わせて実現することが多かった。
  - ✓ CloudFormation(YAML)の場合は展開するリソースが多いと記述が煩雑になりやすい。

2.今回の説明で中心となる構成要素

No	分類	AWSサービス/ツール	内容
1	インフラ構成管理	・CDK	・各種言語で記述したCDKコードをコンパイルしCloudFromationテンプレートに変換する。 ・デプロイ時に既存のCloudFormationスタックの差分を比較し、変化点のみ更新が可能。
2	ソース管理	・CodeCommit	・プライベートなGitリポジトリ。 ・CodePipelineと組み合わせることで、リポジトリへのプッシュでパイプラインを実行可能。
3	パイプライン (ビルド/テスト/デプロイ)	・CodeBuild ・CodePipeline	・ビルド/テスト/デプロイまでの一連の流れをパイプラインとして定義し自動で実行
4	コンテナ	・ECS ・ECR	・アプリケーションを稼働させるためのコンテナと、プライベートなDockerレジストリ。 ・パイプラインと組み合わせるとビルド後にECRにイメージをプッシュして 新ビルドイメージでECSのコンテナを実行できる ・コンテナを採用することで、EC2などサーバを用意する場合はOS設定やアプリのデプロイをパイプラインの処理で実現できる。

3.想定する構成とCDK適用箇所

- アプリケーションを実行するコンテナと依存するAWSリソース群
- アプリケーションのビルド/デプロイの実行に必要なリポジトリやパイプライン



4. まとめ

- まとめ
  - ✓ 既存ツールでは未対応だったAWSリソースの構成管理について、CDKを利用することで一括して対応できる。
  - ✓ CDKによりコードとCloudFormationスタックの両方の視点からデプロイされるリソースをチェックできる
  - ✓ アプリケーションのソース管理からビルド・テスト、デプロイまでの一連の流れをAWSのサービス・ツールで実現できる。

- 注意したい点
  - ✓ CDKはプレビュー段階で事例が多くないため、CDKの記述作法は手探りで進めている段階。
  - ✓ 少しずつ適用範囲を広げ、採用できる場所をふやしていく方がよい。
  - ✓ インフラ・アプリ構成の組み合わせによっては、CDKで一括構成管理できない場合がある。
  - ✓ CDKは命令的にAWSリソースのデプロイを定義していくので記述方法によってはわかりづらくなる。