

AWS Builders Online Series

T5 - 1

AWS で始める スタートアップ最初の技術選定

野口 真吾

アマゾン ウェブ サービス ジャパン合同会社
スタートアップ事業本部 ソリューションアーキテクト



自己紹介

野口 真吾 (のぐち しんご)

Startup Solutions Architect

好きなAWSサービス

AWS Lambda, AWS Step Functions

好きなこと

たこ焼きを焼く



アジェンダ

- ・ スタートアップにおける最初の技術選定
- ・ クラウドに最適化された設計とは
- ・ プロダクト開発に専念できるAWSの使い方



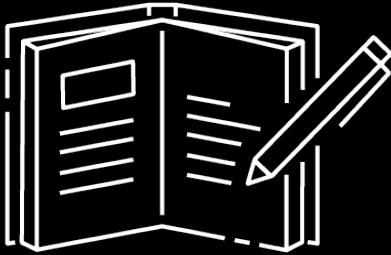
スタートアップにおける 最初の技術選定



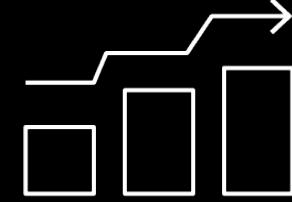
スタートアップにおける技術選定の課題



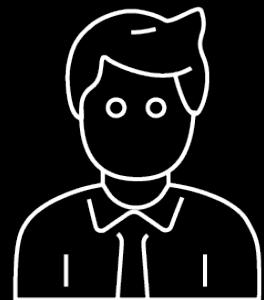
コストを抑えたい



短い期間で学習したい



スケーラビリティは確保したい



インフラ専任担当者
は不在なことが多い



セキュリティを出来るだけ
意識したくない

スタートアップにおける技術選定の課題



コストを抑えたい



短い期間で学習したい



スケーラビリティは確保したい

ビジネスの成長にフォーカスする



インフラ専任担当者は不在



セキュリティを出来るだけ
意識したくない

技術選定において気を付ける点

- Undifferentiated heavy lifting
(差別化に繋がらない重労働) の排除
- 目の前の判断がTwo-way Door
(後からの変更が難しくないもの) かどうか見極める

技術選定において気を付ける点

- Undifferentiated heavy lifting
(差別化に繋がらない重労働) の排除
- 目の前の判断がTwo-way Door
(後からの変更が難しくないもの) かどうか見極める

Undifferentiated heavy lifting

アイデアの実現の過程には
「**差別化に繋がらない重労働**」
が多く存在し、それらを減らすことができれば成功確率を上げることが出来る（意訳）

- Web 2.0 Summit での Tim O'Reilly と Jeff Bezos との対談



<https://www.flickr.com/photos/farber/292880154>



"Cloud computing is transforming IT and the industry. We are entering a new era," says Dr Werner Vogels, chief technology officer of Amazon.com. "There is no industry vertical not making use of Amazon Web Services (AWS) today," said Vogels, in his keynote at the first AWS

Amazon CTO の Werner Vogels も
メディア取材の際に
「Stop spending money on
“**undifferentiated heavy lifting**”」
とコメント

Undifferentiated Heavy Lifting の例

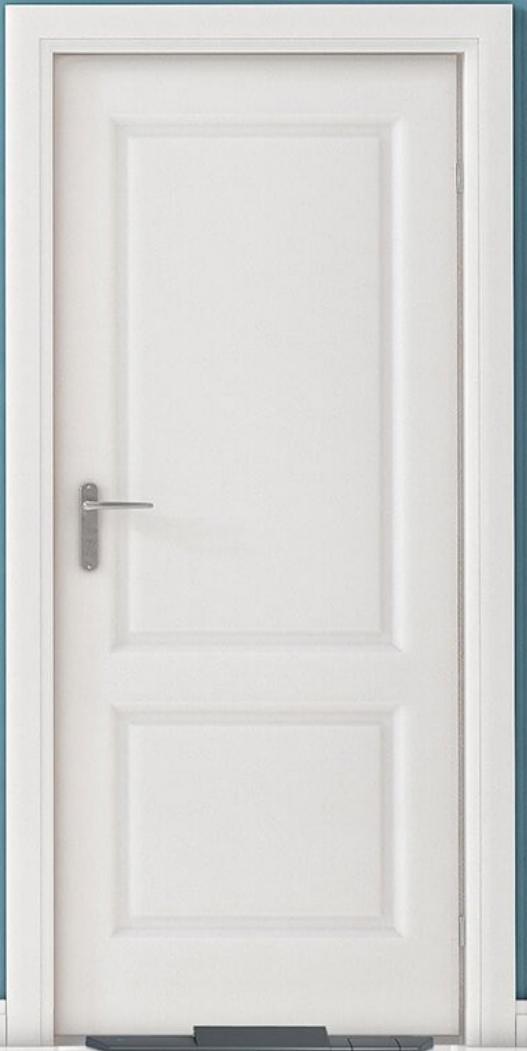
(差別化に繋がらない重労働)

- データベースの運用、バックアップ、レプリケーション設定
- 認証基盤の保守、管理、運用
- リアルタイム通信を行うサーバーの保守、管理、運用
- セキュリティパッチの管理、適用
- etc...



技術選定において気を付ける点

- Undifferentiated heavy lifting
(差別化に繋がらない重労働) の排除
- 目の前の判断がTwo-way Door
(後からの変更が難しくないもの) かどうか見極める



Is it a **one-way** or
a two-way door?

技術選定や設計における One-way Door となること

(後から変更が難しい)

- 特に全体に大きな影響を与えるもの、成長後に変更するコストが大きいものに特に気をつける。
 - データの置き場、将来的な活用方法
 - データベース設計
 - クラウドプラットフォーム
 - etc...

目の前の判断が Two-way Door なものではないか考える

(後から容易に変更できる)

- 逆に、「**それ試しにやってから後で変えることもできるよね**」という要素 (=Two-way Door) では時間を使いすぎずに意思決定する
 - 「試しにやる」ハードルを下げる
- 目の前の決断が One-way door なのか Two-way door なのか見極める
 - 後から変更が可能なことに時間をかけすぎていないか？
- 少しの工夫で Two-way door な状況に出来ないか？



技術選定において気を付ける点

- Undifferentiated heavy lifting
(差別化に繋がらない重労働) の排除
- 目の前の判断がTwo-way Door
(後からの変更が難しくないもの) かどうか見極める

クラウドに最適化された 設計とは



Design for Failure

障害が発生する前提で
アプリケーションを設計する

“Everything fails all the time”

Dr. Werner Vogels
Chief Technology Officer of Amazon.com



スタートアップが最初に目指す Design for Failure

- 想定以上のアクセスが来てもダウンしない
- 問題のあるインフラを置き換えてくれる
- データを誤って消してしまっても復旧できる
- $(+\alpha)$ アプリケーションがエラーになった際のリトライ処理やキューイング



Design for Failure な設計になつていないとどうなるか？

(障害が発生する前提)

- 例1: 大事な時(e.g. VC ヘデモを見せる時等) にダウンしていることによる機会損失
- 例2: メンバーも少ないため、その都度発生する障害対応だけで1日が過ぎてしまう



シード期において、Design for Failure を実現するために

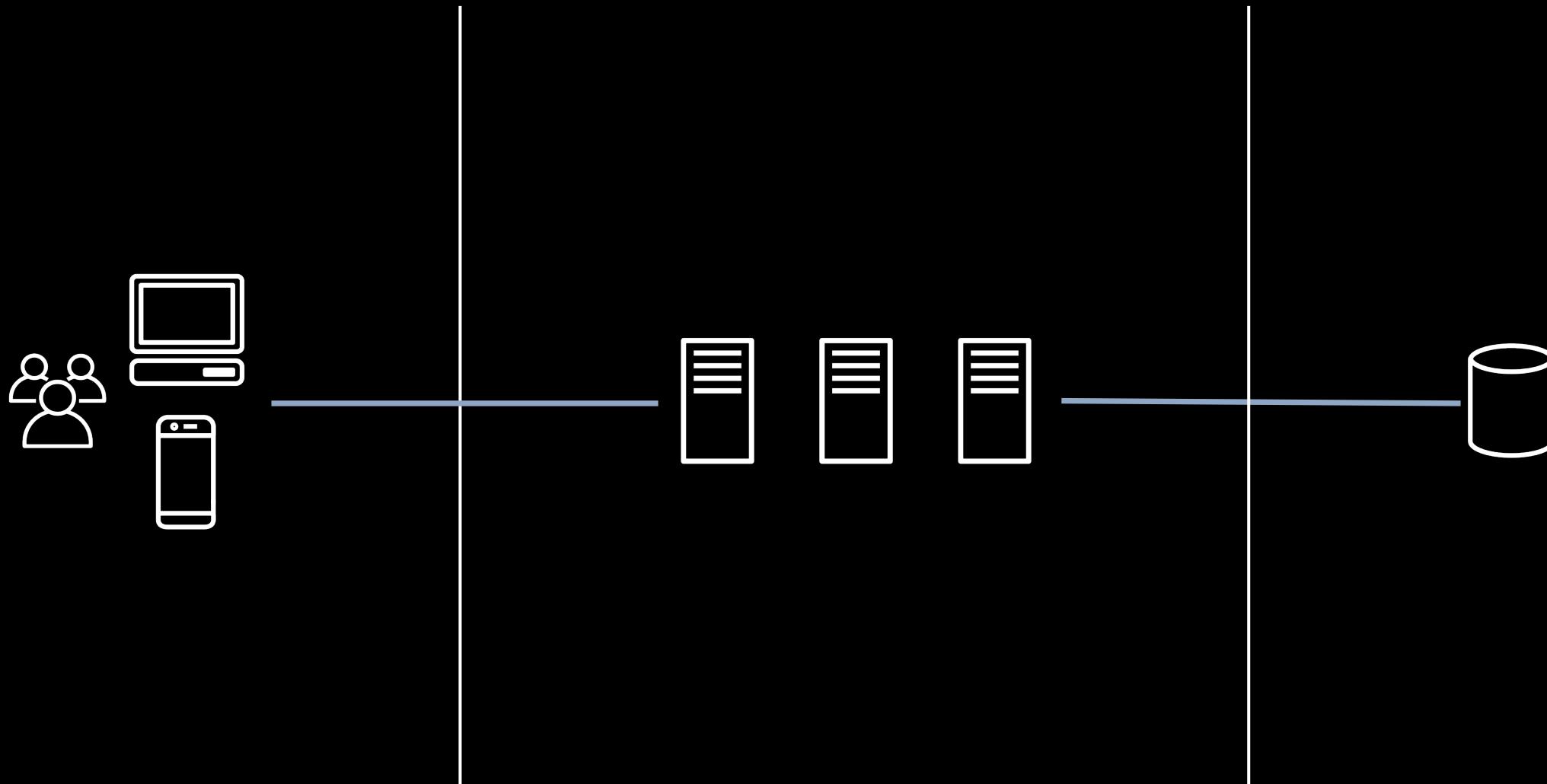
(障害が発生する前提の設計)

- Q. 時間のないシード期のスタートアップにとって、Design for Failure を真面目に全て向き合うのは難しいのでは？
- まずは、**Design for Failure** な設計が組み込まれたサービスを最初から選定できないか考える
 - スピードとコストとのトレードオフではない
 - むしろ対応に追われる工数がなくなりスピードアップ
- 実装やアーキテクチャに依存するところはその先で考える

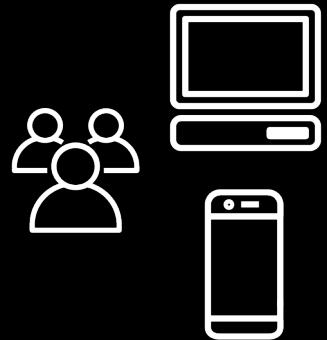
プロダクト開発に専念できる AWSの使い方



オーソドックスな技術スタック



オーソドックスな技術スタック



フロントエンド、
モバイルアプリ

言語・フレームワーク

React, Vue, Swift,
Kotlin, Java etc

インフラ

Web Hosting サービス、
BaaS etc



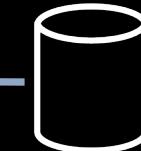
バックエンド
Webサーバー

言語・フレームワーク

Ruby on Rails, Laravel,
Django, Express, Spring etc

インフラ

仮想サーバー、コンテナ、
サーバーレス etc



バックエンド
DBサーバー

DB 種類

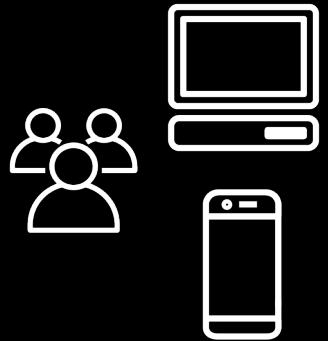
RDBMS, NoSQL,
Fulltext Search, etc

インフラ

マネージドDBサービス、
仮想サーバー etc



負担をオフロードできる AWS のサービス



フロントエンド、
モバイルアプリ



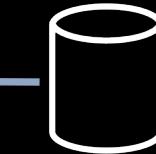
Amplify Hosting



バックエンド
Webサーバー



AWS App Runner



バックエンド
DBサーバー



Amazon Aurora
Serverless v2

Amplify Hosting



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Amplify の 4 つのコンポーネント



Amplify CLI

Web やモバイルアプリケーションを一般的なユースケースベースのガイド付きワークフローでバックエンドを簡単に作成、管理するツール



Amplify Libraries

Web やモバイルアプリケーションと AWS を統合するためのユースケース中心のライブラリ



Amplify Hosting

継続的デプロイメントを管理し、モダンな Web アプリケーションをビルド、テスト、デプロイ、そしてホスティングするための AWS サービス



Amplify Studio

AWS 上に最小限のコーディングでフロントからバックまでのアプリケーションを作成できるビジュアルな開発環境

AWS Amplify の 4 つのコンポーネント



Amplify CLI

Web やモバイルアプリケーションを一般的なユースケースベースのガイド付きワークフローでバックエンドを簡単に作成、管理するツール



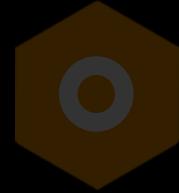
Amplify Libraries

Web やモバイルアプリケーションと AWS を統合するためのユースケース中心のライブラリ



Amplify Hosting

継続的デプロイメントを管理し、モダンな Web アプリケーションをビルド、テスト、デプロイ、そしてホスティングするための AWS サービス



Amplify Studio

AWS 上に最小限のコーディングでフロントからバックまでのアプリケーションを作成できるビジュアルな開発環境

AWS Amplify Hosting

ウェブアプリや静的ウェブサイトのためのフルマネージドなホスティングサービス
簡単なカスタムドメイン設定も可能

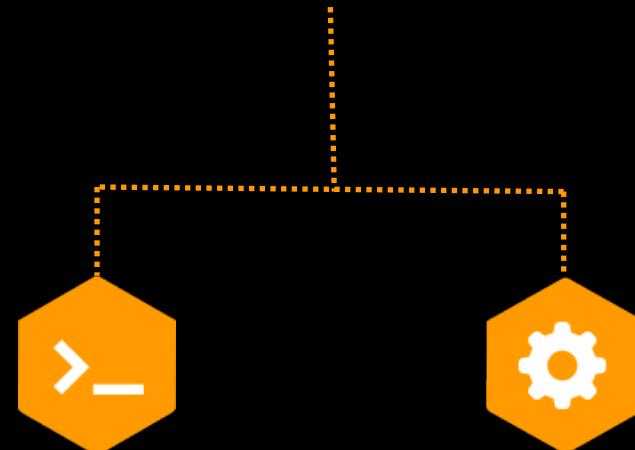


アプリのフロントエンドと
バックエンドのデプロイ

継続的デプロイメント

Amplify CLI で Infrastructure as Code (IaC)
を生成

Amplify CLI やコンソールからのデプロイ

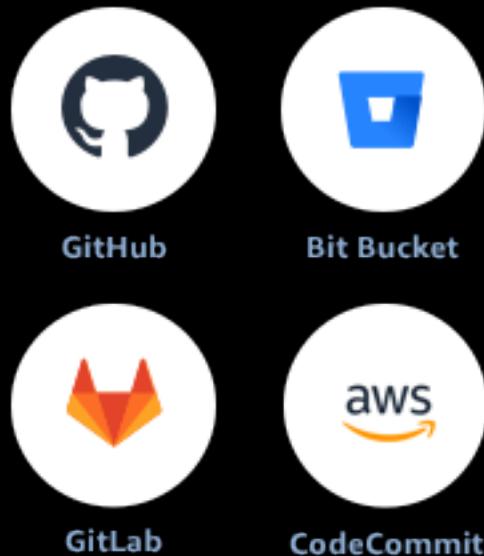


AWS Amplify CLI

AWS Amplify Console



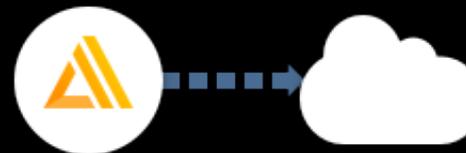
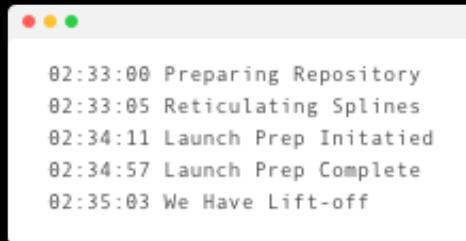
Amplify Hosting の利用フロー



1. リポジトリを接続



2. ビルド設定



3. デプロイ

Amplify Hosting の主な機能



Amplify Hosting - 環境の分割

接続したブランチ毎に
ホスティングされ、URL
が払い出される

This tab lists all connected branches, select a branch to view build details.

main

Continuous deploys set up (Edit)

<https://main...amplifyapp.com>

プロビジョン ビルド デプロイ 検証

前回のデプロイ
8/9/2021, 4:10:48 PM

最終コミット
This is an autogenerated message | Auto-build | GitHub - main

Previews
Disabled

dev

Continuous deploys set up (Edit)

<https://dev...amplifyapp.com>

プロビジョン ビルド デプロイ 検証

前回のデプロイ
8/10/2021, 10:23:09 AM

最終コミット
これは自動生成されたメッセージです | Auto-build | GitHub - dev

Previews
Disabled



Amplify Hosting - Pull Request Previews

Previews

Previews offer a way to preview changes before merging a pull request. [Learn more](#)

The screenshot shows the Amplify Hosting Pull Request Previews interface. At the top, there's a diagram illustrating the workflow: 'Create a branch' (with a person icon), 'Add commits' (with a tree icon), 'Open a pull request' (with a link to https://pr1.amplifyapp.com), 'Discuss and review' (with a speech bubble icon), and 'Merge and deploy' (with a merge icon). Below the diagram, it says 'PR preview is deleted'. Under 'Pull requests', there's a search bar and a table with columns: Name, Description, Preview URL, Status, and Branch. One row is shown: 'pr-2' with 'GitHub - Update README.md' description, 'https://pr-2.d19ab8t30yq0qc.amplifyapp.com' Preview URL, 'In progress' status, and 'master' branch. An orange 'Preview settings' button is at the top right of the table.

The screenshot shows a GitHub Pull Request interface. On the left, there's a green checkmark icon with a wrench symbol. The main area displays a summary of build checks: 'All checks have passed' (1 successful check), 'AWS Amplify Console Web Preview' (green checkmark), and 'This branch has no conflicts with the base branch' (green checkmark). Below this, there's a 'Merge pull request' button and a note: 'You can also open this in GitHub Desktop or view command line instructions.' A red box highlights the 'Details' link next to the AWS Amplify Console Web Preview section.

<https://docs.aws.amazon.com/amplify/latest/userguide/pr-previews.html>



Amplify Hosting - ベーシック認証

Access control

Restrict access to your branches with a username and password. [Learn more](#)

Access control settings

Apply a global password - OFF

Branch name	Access setting	Username	Password
develop	Restricted - password required	jaga	***** Password must be at least 7 characters
master	Publicly viewable		

[Cancel](#) [Save](#)

接続したブランチ毎に
Username/Password を設定して
ベーシック認証をかけることが可能

<https://docs.aws.amazon.com/amplify/latest/userguide/access-control.html>

もっと AWS Amplify を知りたい方へ

公式ドキュメント

<https://docs.amplify.aws>

AWS Summit 2021 「Web・モバイルアプリ開発を加速させる AWS Amplify」

https://d1.awsstatic.com/events/jp/2021/summit-online/AWS-47_AWS_Summit_Online_2021_FWM01.pdf

ワークショップ

<https://amplify-sns.workshop.aws/ja/>

Amplify 学習リソース集

<https://aws-amplify-jp.github.io/resources>

Amplify Japan User Group Slack

<https://github.com/aws-amplify-jp/awesome-aws-amplify-ja#slack>

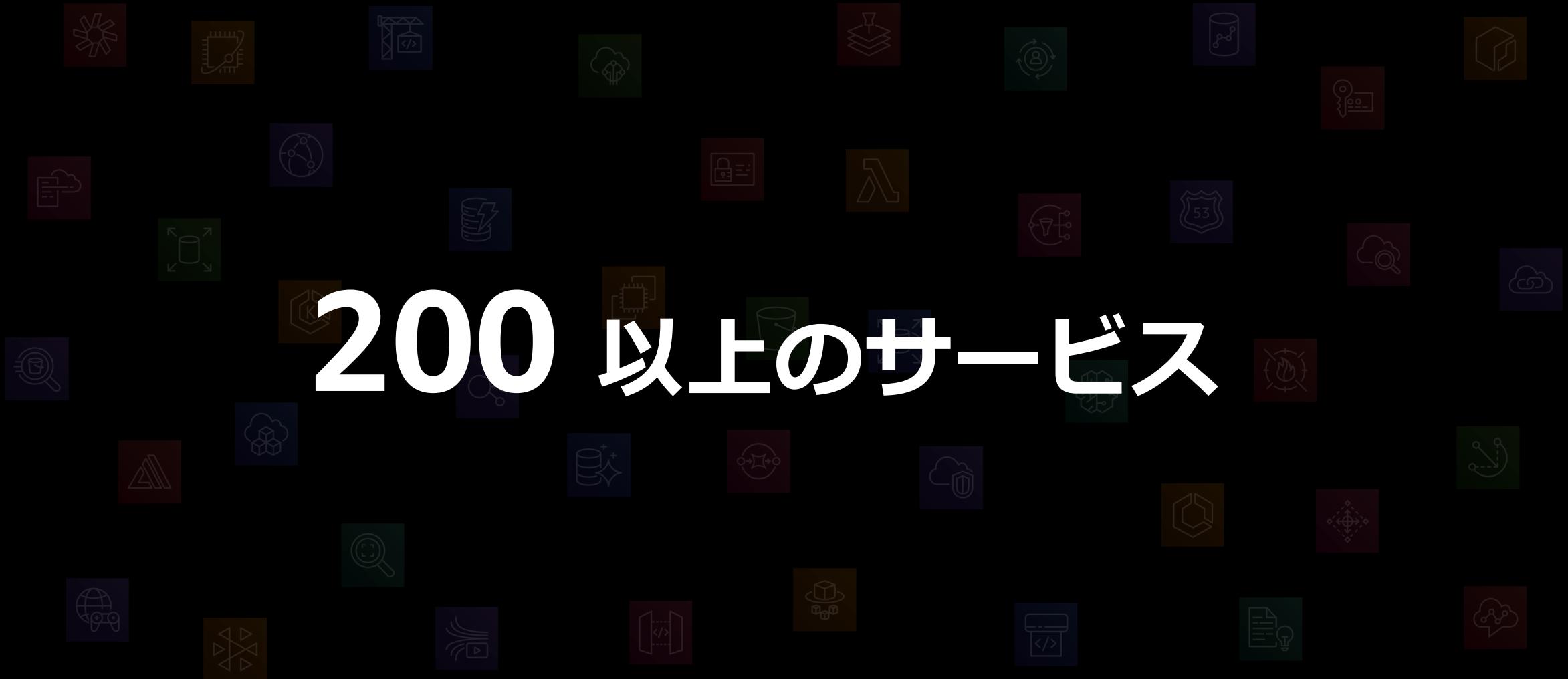


AWS App Runner



AWS は幅広い選択肢を提供

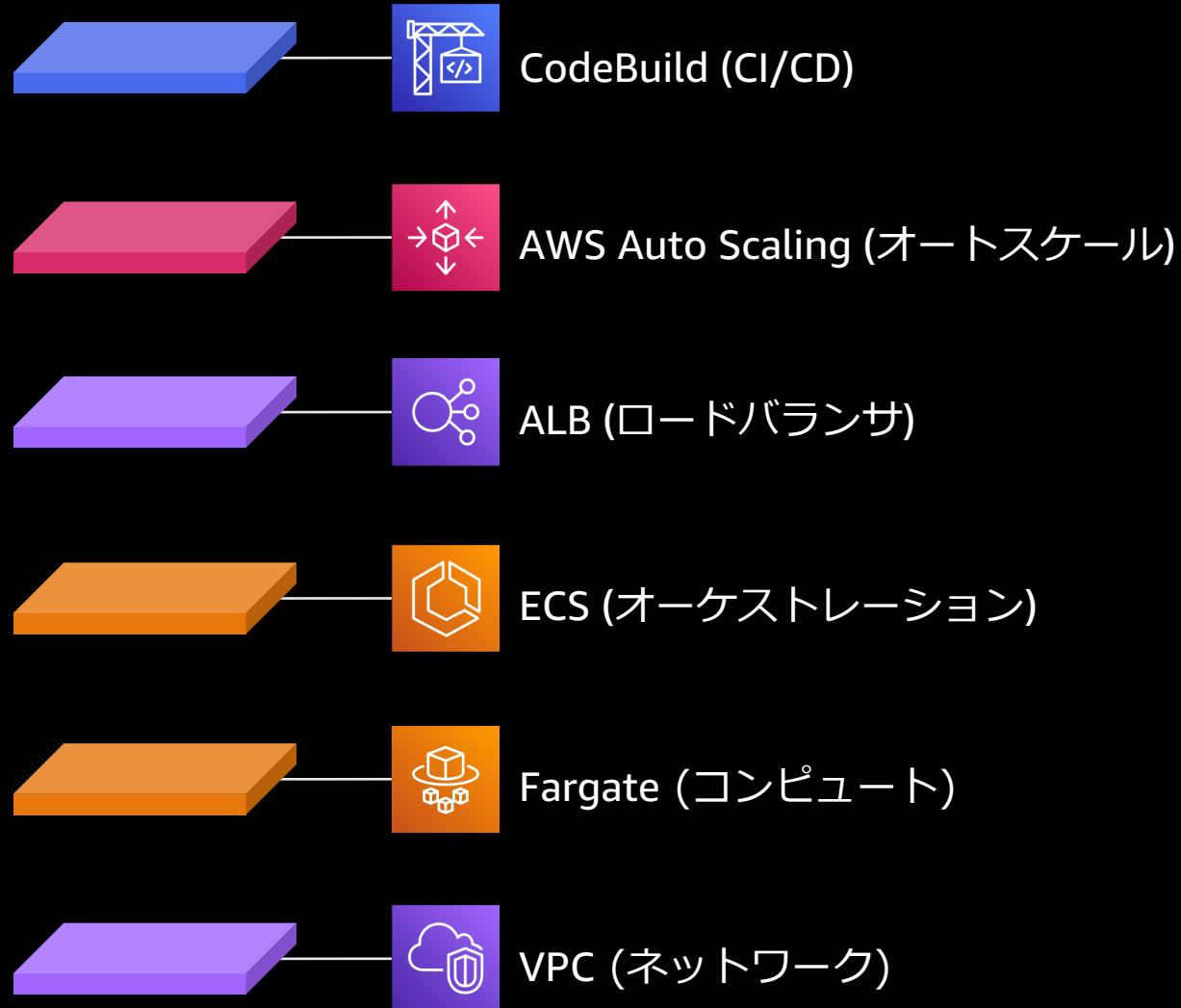
200 以上のサービス



ウェブアプリケーションを AWS 上で動かしたい場合

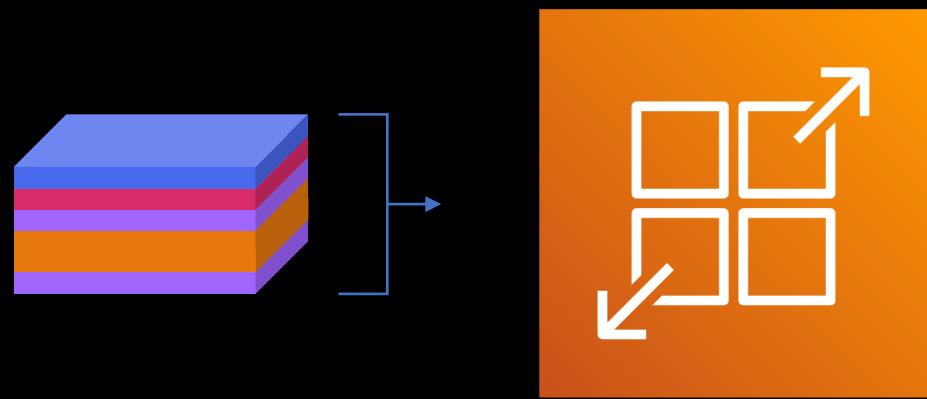


AWS サービスを組み合わせてインフラを構築



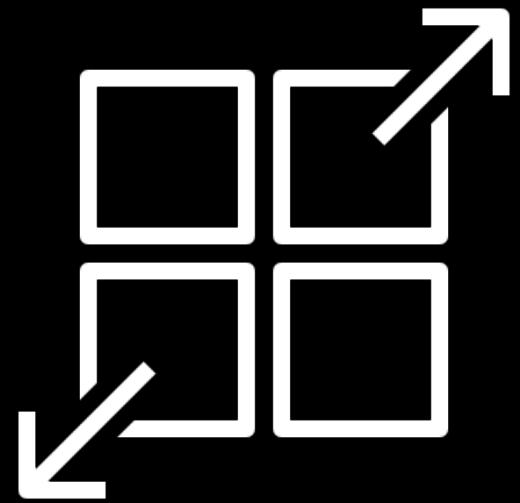
柔軟性が高い

手間がかかる



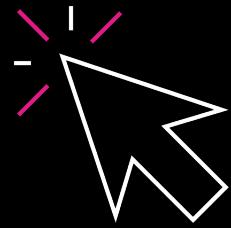
AWS App Runner

App Runner とは

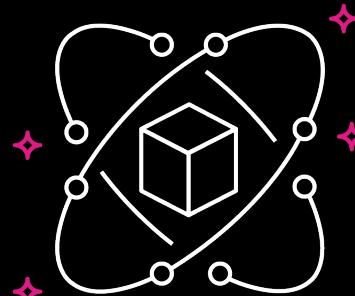


インフラや AWS についての経験がなくても、
コンテナ化されたウェブアプリケーションや API を
簡単かつ迅速に AWS 上で動かせるサービス

AWS App Runner の主な特徴



シンプルな
セットアップ

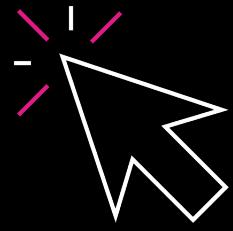


シンプルな
オートスケーリング

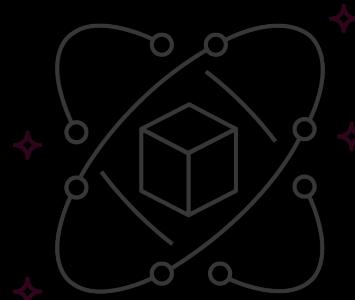


統合された
メトリクス・ロギング機能

AWS App Runner の主な特徴



シンプルな
セットアップ



シンプルな
オートスケーリング



統合された
メトリクス・ロギング機能

App Runner 上にウェブアプリケーションをデプロイ



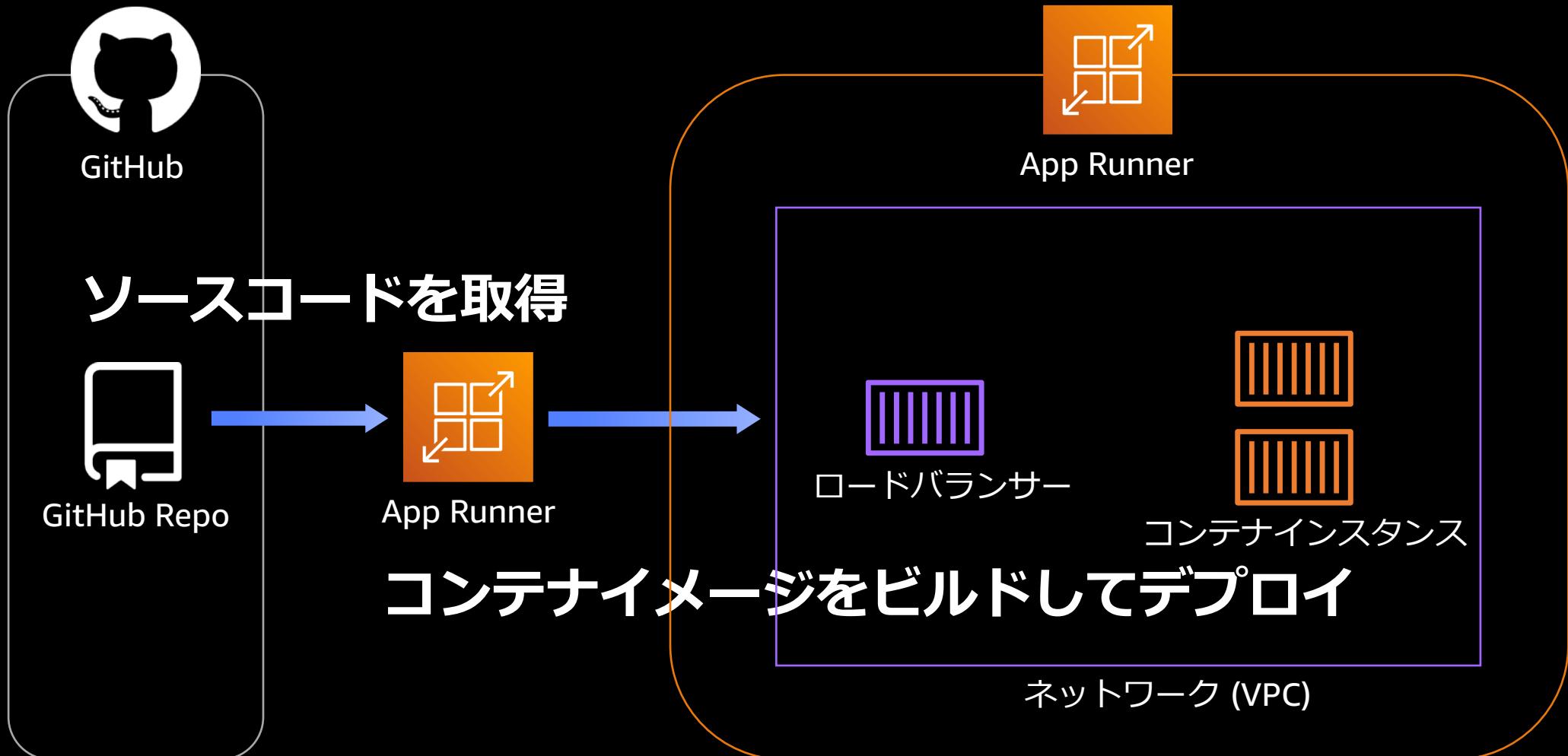
動かしたいウェブアプリケーションを指定する

App Runner 上にウェブアプリケーションをデプロイ

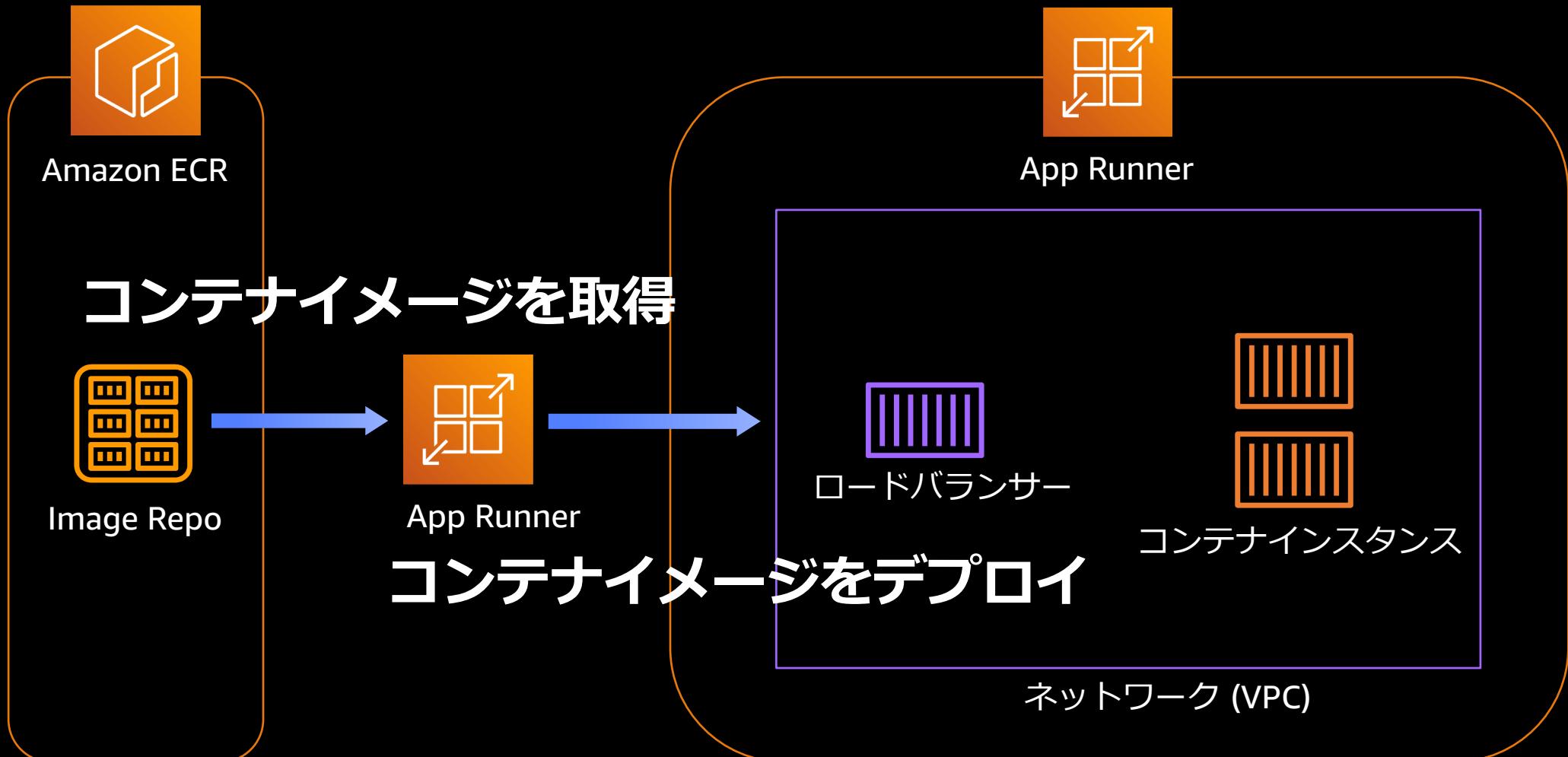


動かしたいウェブアプリケーションを指定する

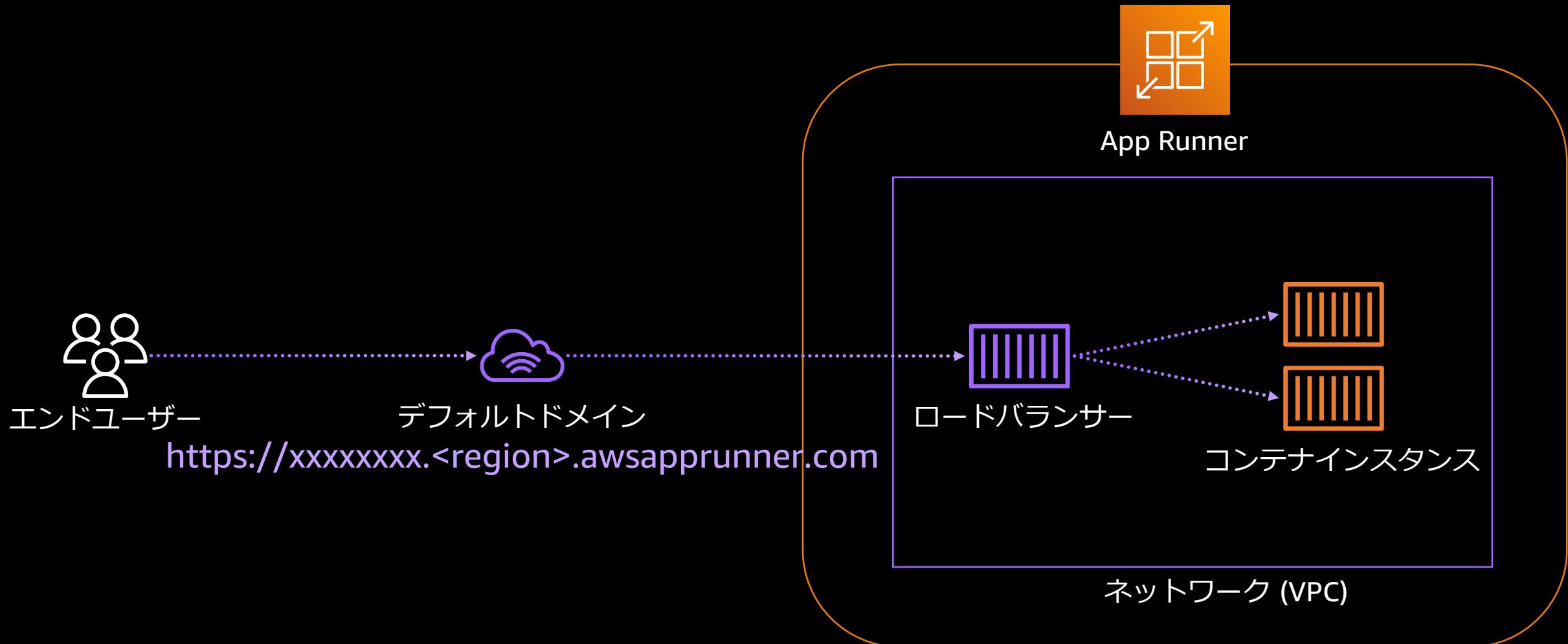
App Runner 上にウェブアプリケーションをデプロイ



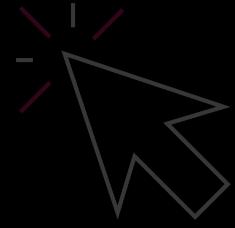
App Runner 上にウェブアプリケーションをデプロイ



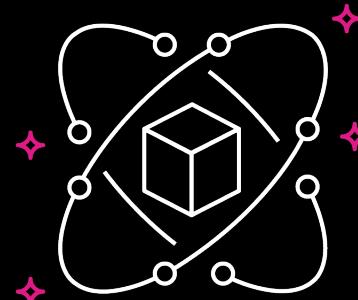
App Runner 上にウェブアプリケーションをデプロイ



AWS App Runner の主な特徴



シンプルな
セットアップ



シンプルな
オートスケーリング



統合された
メトリクス・ロギング機能

App Runner - シンプルなオートスケーリングの設定

Add custom auto scaling configuration

Configuration name
DefaultConfiguration

Concurrency
App Runner scales your service when the number of simultaneous requests per instance exceeds this limit.
100 requests per instance

Minimum size
The minimum number of provisioned instances App Runner maintains with low incoming request traffic.
1 instances
1 - 25

Maximum size
The maximum number of instances your service scales to.
25 instances
1 - 25

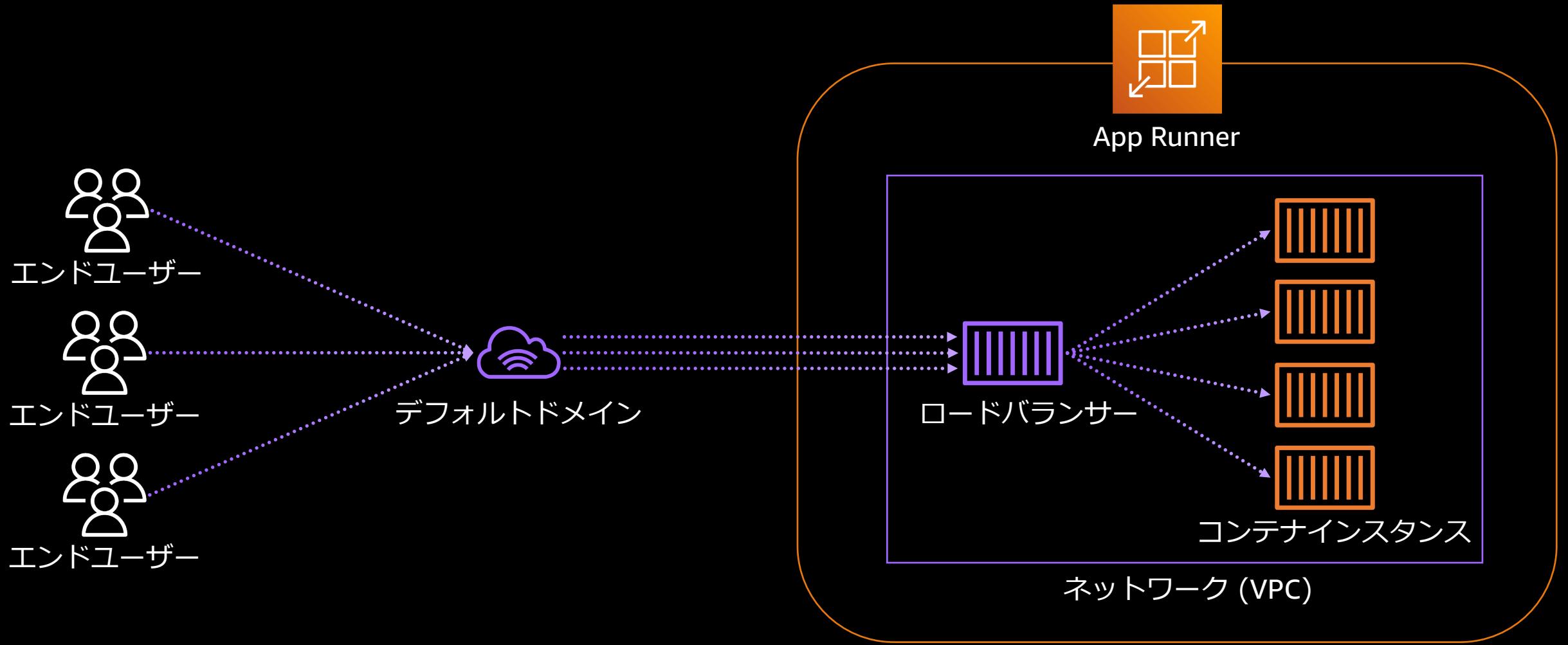
Cancel Save

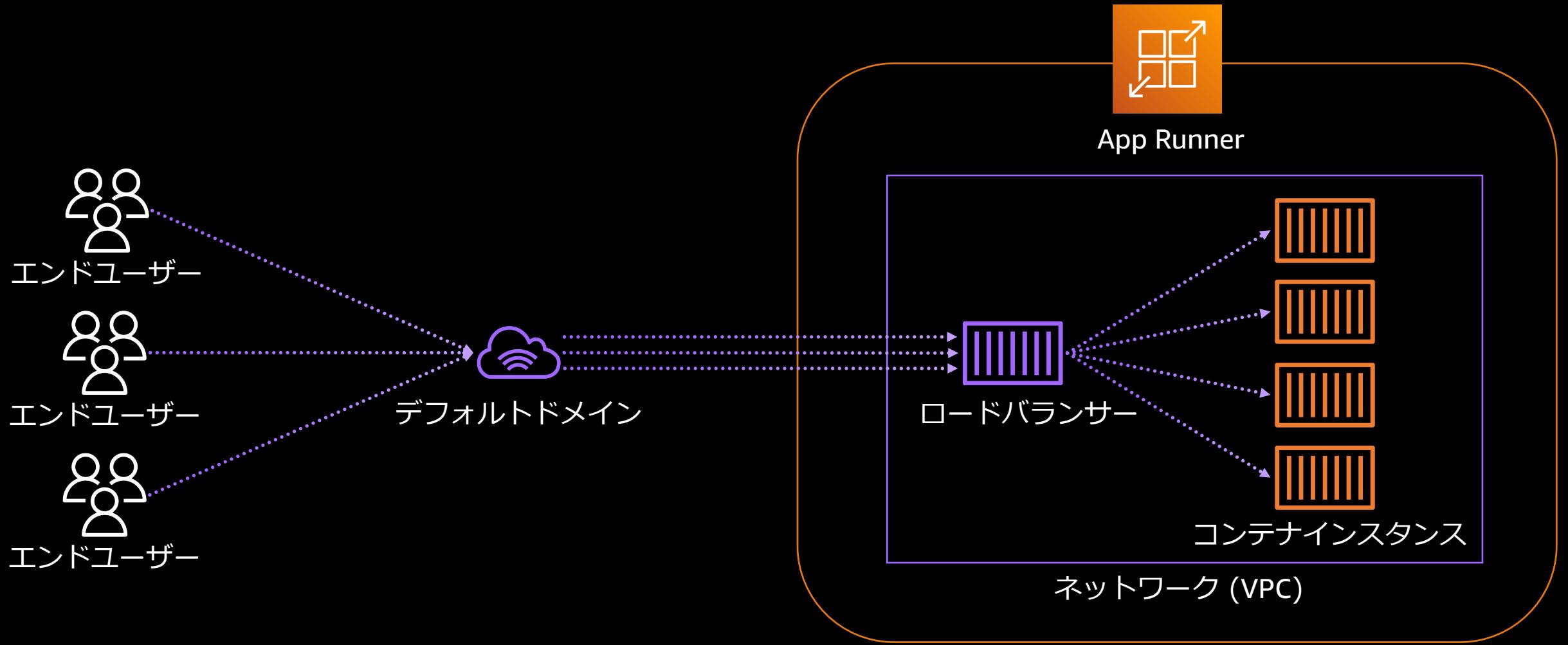
デフォルト
の設定値

オートスケールのしきい値
となるリクエスト数

コンテナインスタンスの
最小数

コンテナインスタンスの
最大数

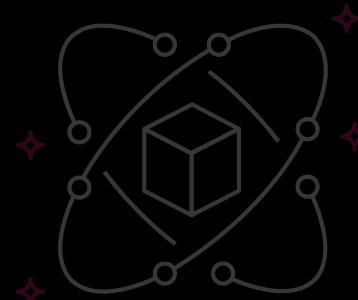




AWS App Runner の主な特徴



シンプルな
セットアップ



シンプルな
オートスケーリング



統合された
メトリクス・ロギング機能

ログとメトリクスを自動で収集



App Runner - Console でのログの閲覧

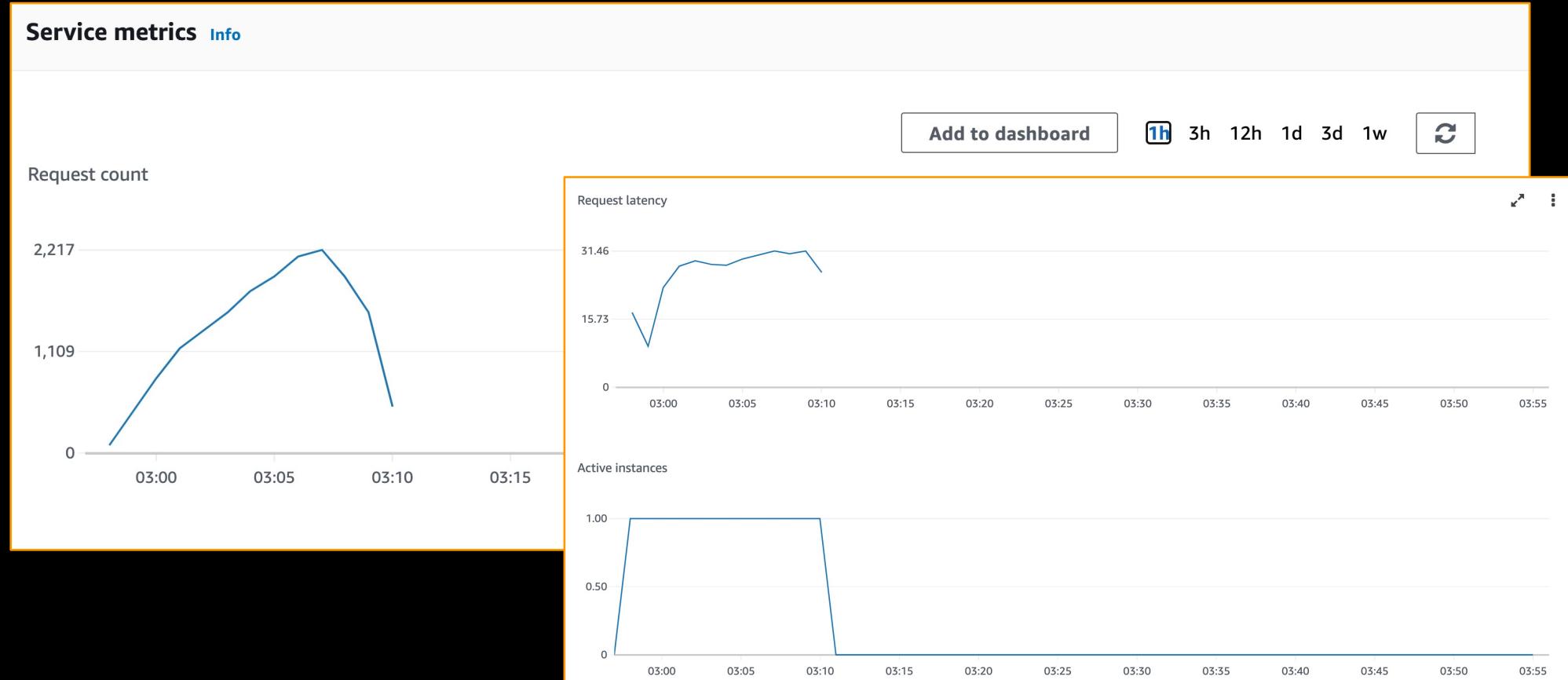
Application logs

Search

1 08-04-2021 11:11 AM ready - started server on 0.0.0.0:3000, url: http://localhost:3000
2 08-04-2021 11:11 AM \$ next start
3 08-04-2021 11:11 AM yarn run v1.22.10

自分でログエージェントを実装する必要なし

App Runner - Console でのメトリクスの閲覧



リクエスト数、応答のレイテンシ、HTTP 2XX 4XX 5XX それぞれの数
を App Runner のコンソール画面で閲覧可能

もっと AWS App Runner を知りたい方へ

<https://docs.aws.amazon.com/apprunner/latest/dg/what-is-apprunner.html>

<https://aws.amazon.com/jp/blogs/news/introducing-aws-app-runner/>

<https://www.apprunnerworkshop.com/>

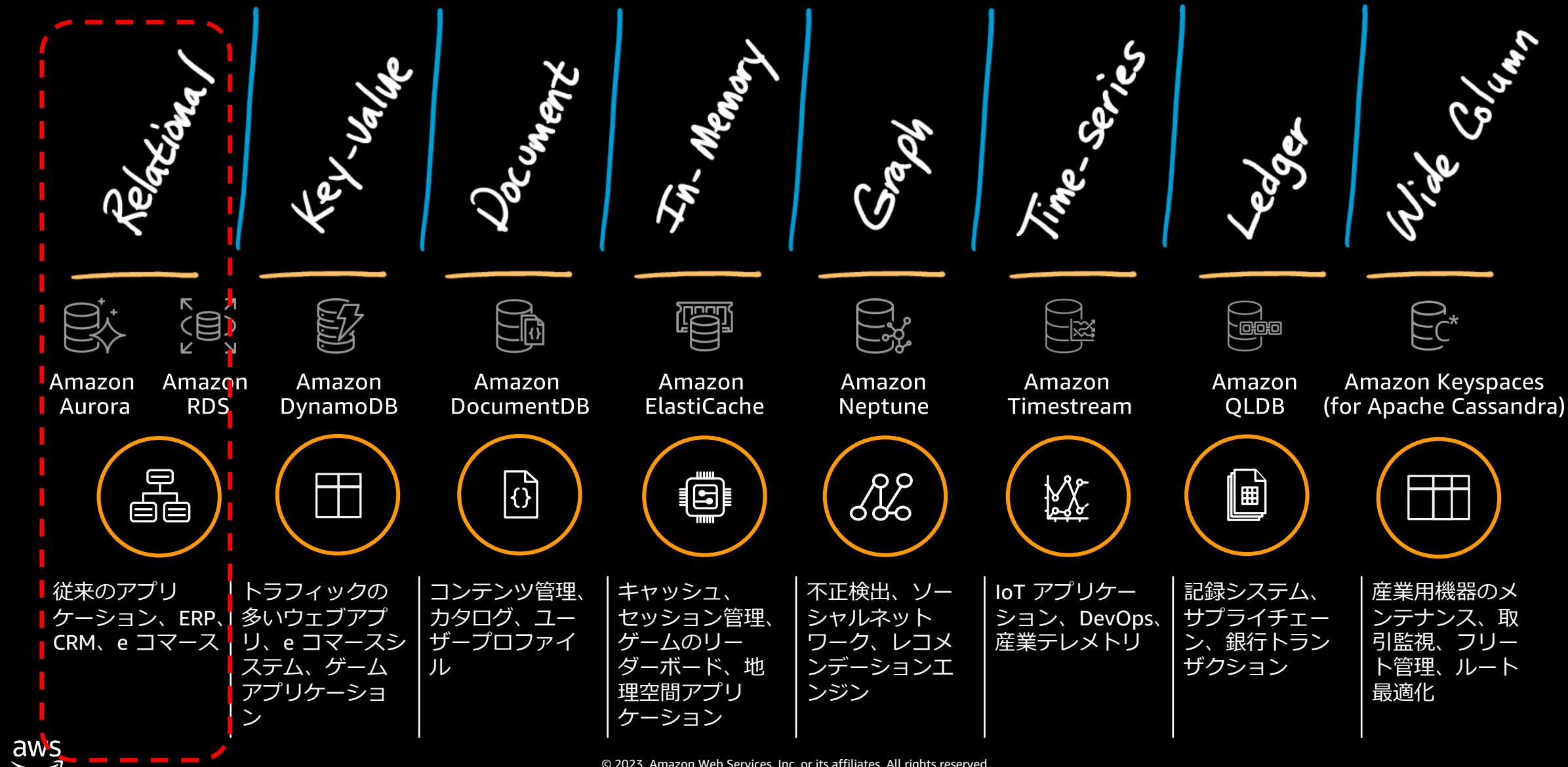
<https://github.com/aws/apprunner-roadmap/projects/1>



Amazon Aurora Serverless v2



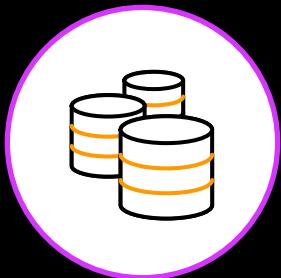
AWS purpose-built database



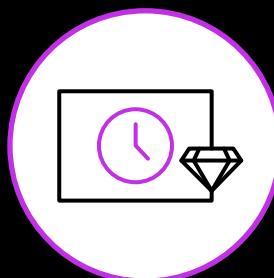
Amazon Aurora

クラウド向けに再設計された MySQL, PostgreSQL と互換性のある RDBMS
商用データベースの性能と可用性を 1/10 のコストで

優れた性能と拡張性



高可用性と耐久性



高い安全性



フルマネージド



Amazon Aurora のアーキテクチャ

データベース用に設計された
専用の分散ストレージシステム

データは3箇所に分散された数百
のストレージノードに6つのコ
ピーとして保存



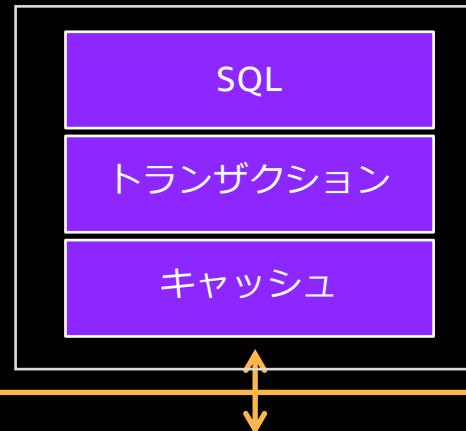
Amazon Aurora のアーキテクチャ

データベース用に設計された
専用の分散ストレージシステム

データは3箇所に分散された数百
のストレージノードに6つのコ
ピーとして保存

SQL を実行するインターフェース
は分散ストレージを共有するこ
とで、責務が分離

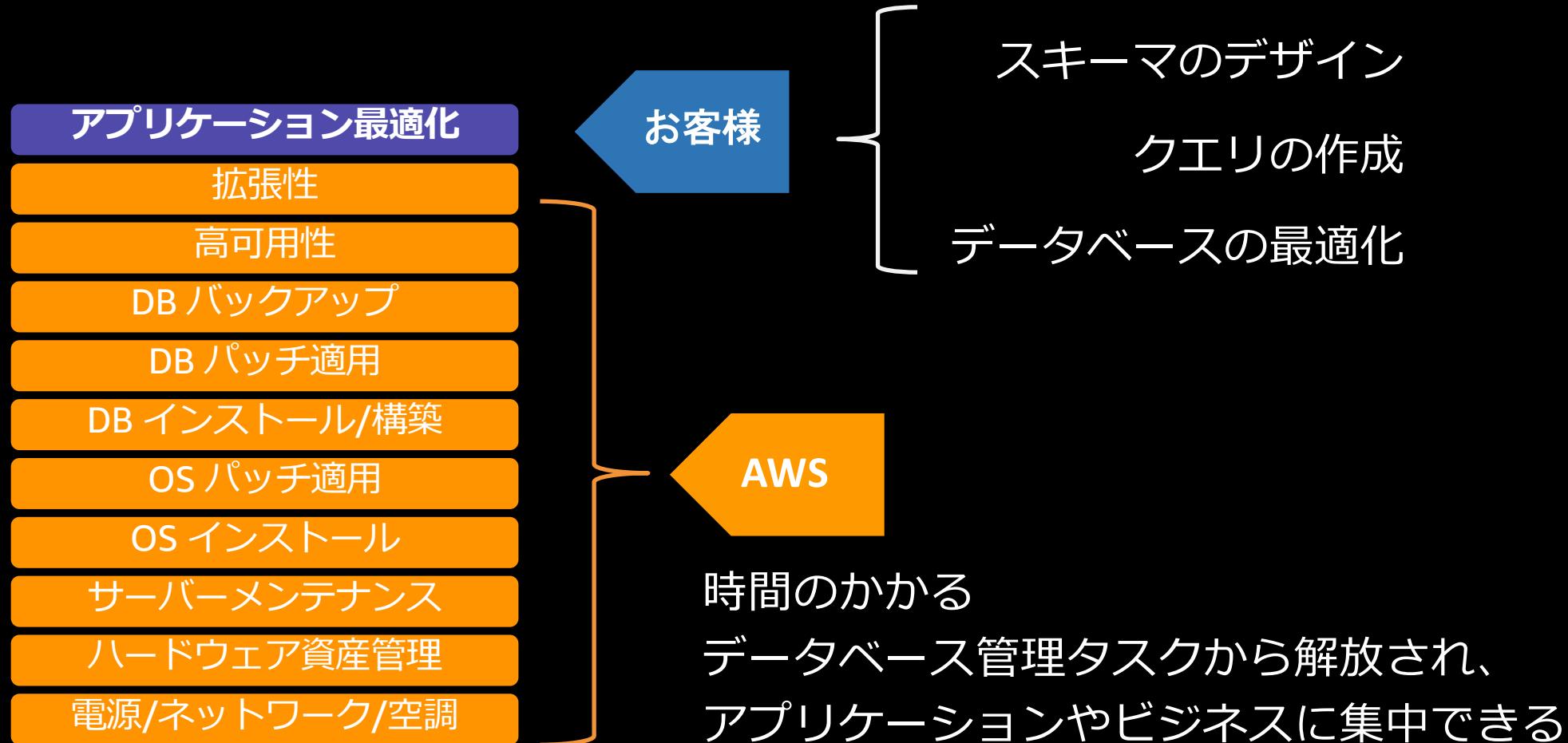
コンピュートノード



分散ストレージシステム



フルマネージド化による管理負荷の軽減



Amazon Aurora Serverless v2



アプリケーションのニーズに応じて自動的に容量を拡張

インスタンスタイプの一つとして簡単なセットアップ

秒単位のシンプルな従量課金

瞬時に拡張し、要求の厳しいアプリケーションをサポート

データベースのキャパシティ管理の心配からの解放

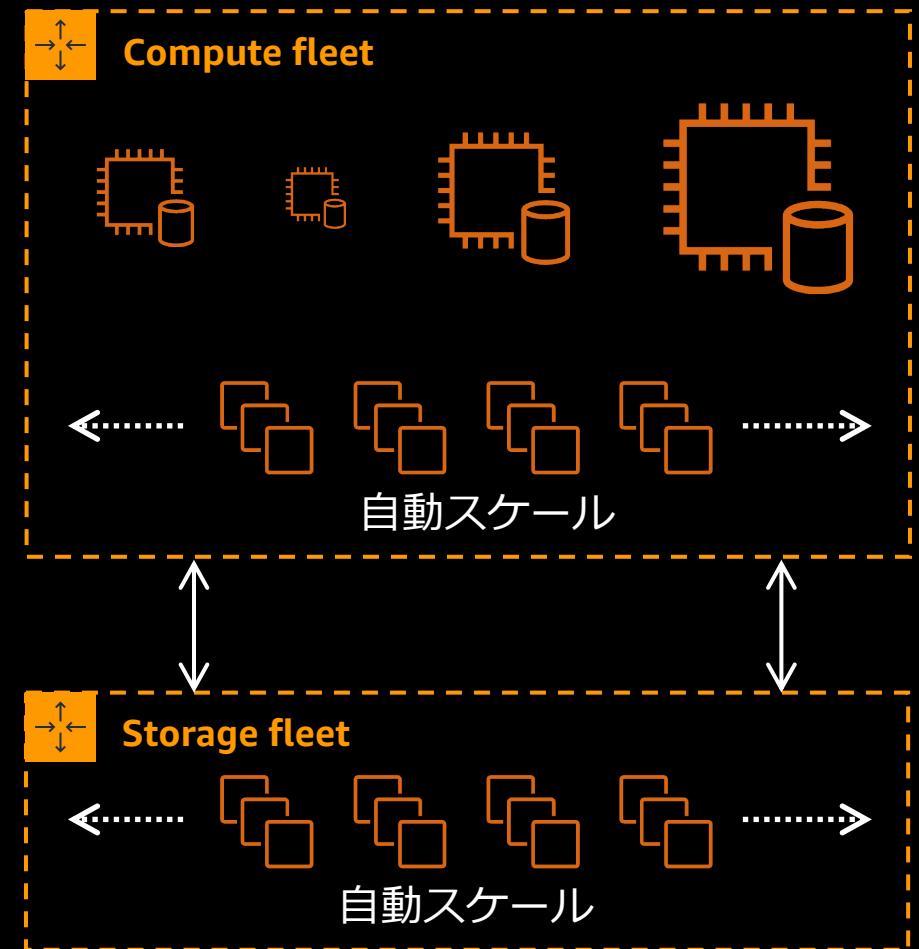
急激なアクセス増減対応の選択肢

Amazon Aurora Serverless v2



急激なアクセス増減対応の選択肢

Amazon Aurora Serverless v2

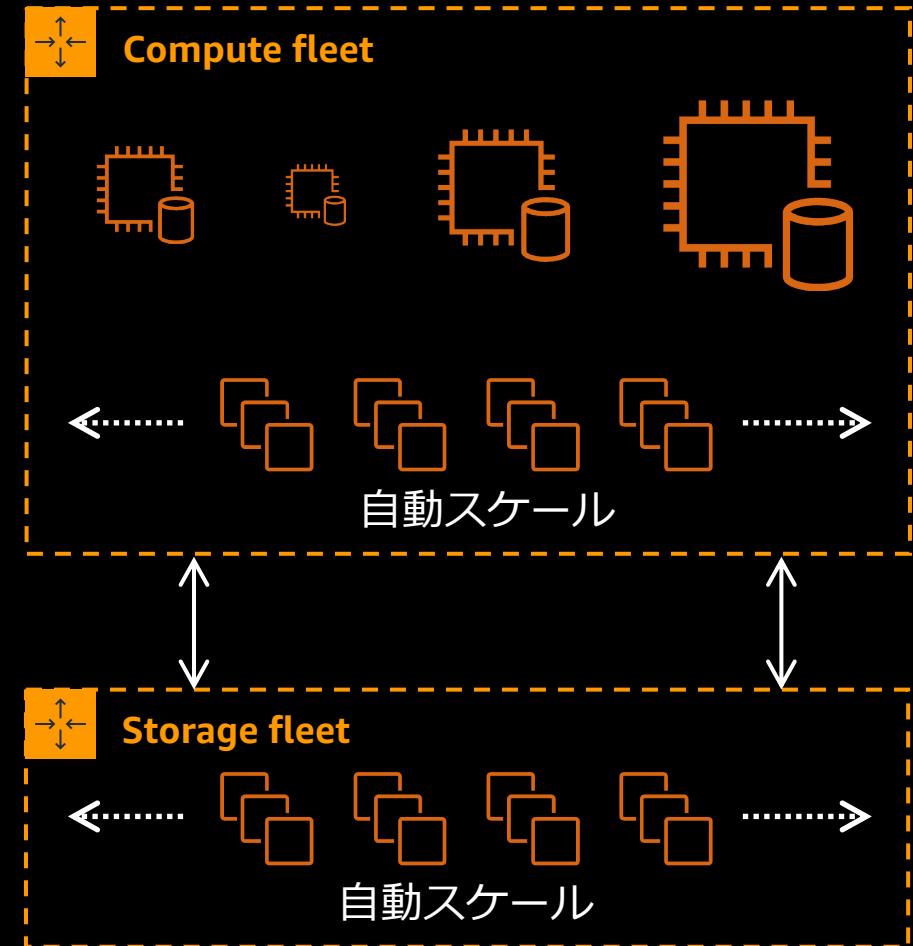


急激なアクセス増減対応の選択肢

Amazon Aurora Serverless v2

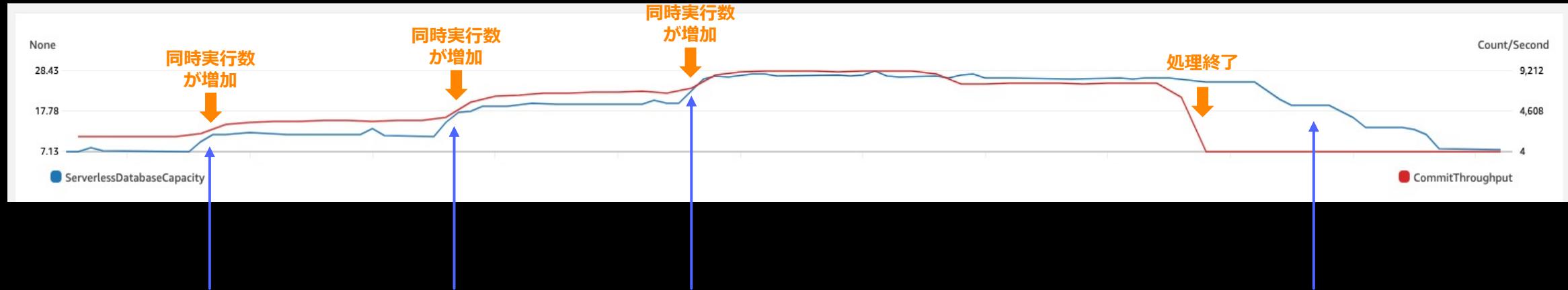
- ・ **インプレーススケール** : CPUやメモリのリソースなどを動的に追加することで、1秒以内にスケーリングが可能
- ・ **パフォーマンス影響なし** : 数十万トランザクションを実行中でも、スケーリングによる影響はない

<https://aws.amazon.com/jp/blogs/news/amazon-aurora-serverless-v2-is-generally-available-instant-scaling-for-demanding-workloads/>



Aurora Serverless v2 のシームレスなスケーリング

Aurora Serverless v2 のスケーリング例 (定期的に同時実行数を上げながら OLTP 処理を実施)



同時実行数が増加して、必要なリソースが
増加した時点で、Aurora Serverless v2の
キャパシティが増加(青線)
また、スケール時にトランザクション
(赤線のCommitThroughput)を阻害しない

処理が終了して、リソースが不要
になると徐々にキャパシティが
減少(青線)

もっと Amazon Aurora Serverless v2 を知りたい方へ

公式ドキュメント

<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-serverless-v2.html>

ブログ

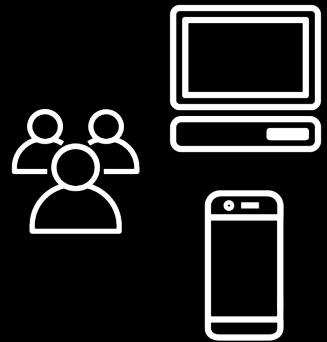
<https://aws.amazon.com/jp/blogs/news/amazon-aurora-serverless-v2-is-generally-available-instant-scaling-for-demanding-workloads/>

Deep Dive

https://www.youtube.com/watch?time_continue=350&v=b2Tl6SsWC-M&feature=emb_title



負担をオフロードできる AWS のサービス



フロントエンド、
モバイルアプリ



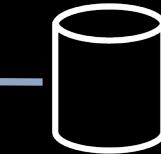
Amplify Hosting



バックエンド
Webサーバー



AWS App Runner



バックエンド
DBサーバー



Amazon Aurora
Serverless v2

負担をオフロードできる AWS のサービス

- ・ リポジトリ連携でCI/CD
- ・ 複数の環境を簡単構築

フロントエンド、
モバイルアプリ



Amplify Hosting

- ・ 簡単デプロイ
- ・ シンプルな設定でオートスケール

バックエンド
Webサーバー



AWS App Runner

- ・ RDBMSでありながら
高速スケールアップ

バックエンド
DBサーバー



Amazon Aurora
Serverless v2

さいごに - お伝えしたこと -

- スタートアップにとって大事なことは
ビジネスの成長にフォーカスすること
- スタートアップが技術選定の際に考えるべきこと
 - Undifferentiated Heavy Lifting (差別化に繋がらない重労働)の排除
 - 目の前の判断が Two-way Door (後からの変更が難しくないもの)
なものではないか見極める
- まずは、Design for Failure (障害が発生する前提の設計)
な設計が組み込まれたサービスを選定できいか考える

AWS TRAINING & CERTIFICATION

AWS Skill Builder の 500+ の 無料デジタルコースで学ぼう

30以上のAWSソリューションの中から、自分に最も関係のあるクラウドスキルとサービスにフォーカスし、自習用のデジタル学習プランとRamp-Upガイドで学ぶことができます。

- 自分のペースでAWSクラウド上を活用した未来を切り開く
- 学習プランでスキルや知識を向上
- AWS認定資格でクラウドの専門知識を証明する



**自分に合ったスキルアップ方法をで学びましょう
[EXPLORE.SKILLBUILDER.AWS](#) »**



AWS Builders Online Series に ご参加いただきありがとうございます

楽しんでいただけましたか? ぜひアンケートにご協力ください。
本日のイベントに関するご意見/ご感想や今後のイベントについての
ご希望や改善のご提案などがございましたら、ぜひお聞かせください。



aws-apj-marketing@amazon.com



twitter.com/awscloud_jp



facebook.com/600986860012140



<https://www.youtube.com/user/AmazonWebServicesJP>



<https://www.linkedin.com/showcase/aws-careers/>



twitch.tv/aws



Thank you!

Shingo Noguchi

Startup Solutions Architect

