

IVS CTO Night & Day

[re:Cap] Containers & Microservices

2018.12.18 Kazuki Matsuda



Kazuki Matsuda

Solutions Architect - Startups
Amazon Web Services Japan

《 前職 》

AdTech系のBigData的なスタートアップ

《 好きな AWS のサービス 》

Amazon Timestream (New!!), AWS Lambda, **Chalice**



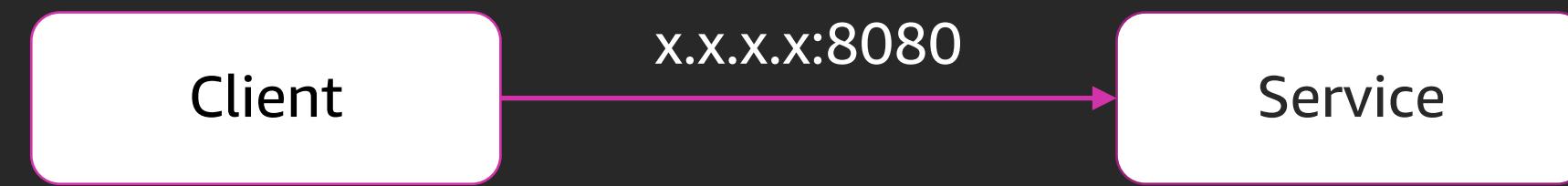
Agenda

- New Launch: AWS Cloud Map
- New Launch: AWS App Mesh
- Amazon ECS 関連アップデート
- Amazon EKS 関連アップデート
- Amazon ECR 関連アップデート
- その他のアップデート

AWS Cloud Map

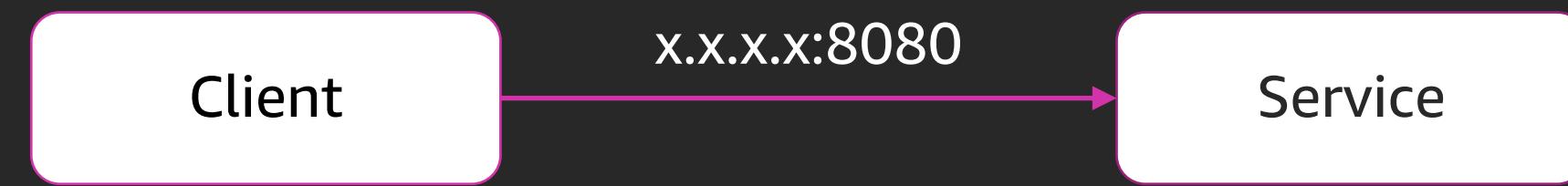
クラウドリソースのサービスディスカバリを実現するマネージドサービス

依存リソースへの一貫したアクセス



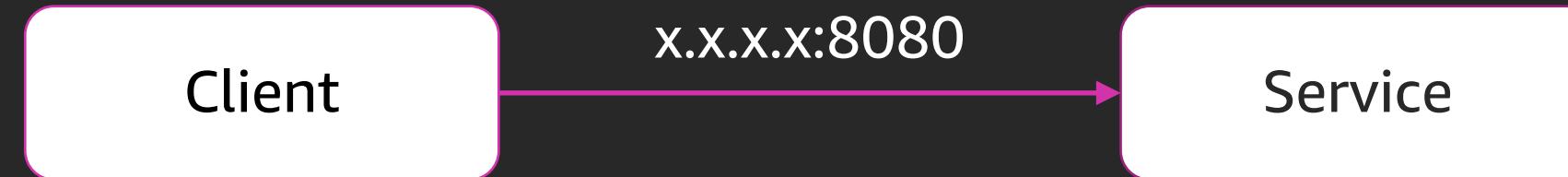
Host: x.x.x.x
Port: 8080

依存リソースへの一貫したアクセス



Host: **y.y.y.y**
Port: 8080

依存リソースへの一貫したアクセス



- 解決策



ロードバランサを利用したサーバサイドディスカバリ、ヘルスチェック



DNS を利用したディスカバリ、クライアントサイドディスカバリ、ヘルスチェック

依存リソースへの一貫したアクセス



これらの手段がサポートしていないリソースは？

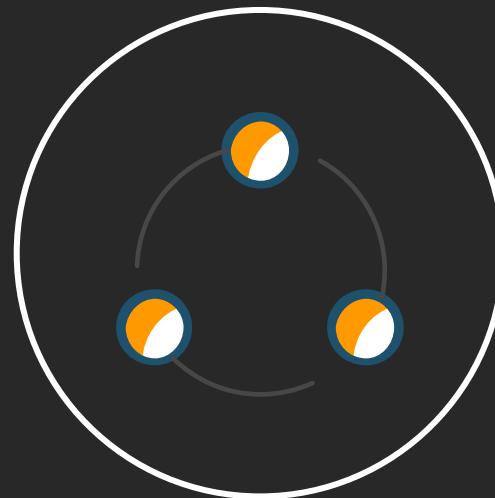
依存リソースへの一貫したアクセス

- dev-myservice.mydomain.local
- stg-myservice.mydomain.local
- prod-myservice.mydomain.local
- etc., etc.

↑ アプリケーションから見た依存サービスの論理名がデプロイメントステージごとに影響を受けて一貫性を保てていない例

Introducing AWS Cloud Map

クラウドリソースの動的な「地図」を管理する



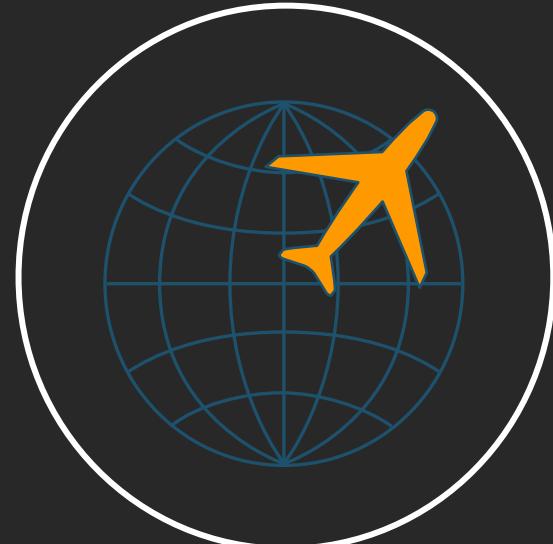
任意の“論理名”による
ディスカバリ



特定の属性からの
ディスカバリ



正常状態なリソースの
ディスカバリ



高可用なリージョナル API と
グローバル DNS

AWS Cloud Map Key Features



- 全てのクラウド リソースのためのレジストリ
- 高速かつセキュアな名前解決
- 属性情報を利用したディスカバリ
- Amazon Route 53 ヘルスチェックを利用した対向システムの状態以上への対応
- AWS プラットフォームやオープンソースソリューションとのインテグレーション

AWS Cloud Map: Benefits

	高速かつセキュア	<ul style="list-style-type: none">5秒未満での高速な変更の伝播認証認可が可能なサービスディスカバリ平均2msの低レイテンシなディスカバリ
	シンプルなサービスディスカバリ	<ul style="list-style-type: none">全コンテナサービスに共通のサービスディスカバリを提供属性ベースのディスカバリ論理名や親和性のある名前によるサービスディスカバリ
	名前解決	<ul style="list-style-type: none">ELB の CNAME と A レコードをサポート最大8アドレスまでを名前解決の結果として返すことが可能IP ベースのリソースに対しては DNS クエリによるディスカバリも可能
	管理容易性	<ul style="list-style-type: none">IP ベースのリソースに対する Amazon Route 53 ヘルスチェックAWS プラットフォームやオープンソースソリューションとのインテグレーション

AWS Cloud Map: Ecosystem

AWS Integrations

- Amazon Elastic Container Services (ECS)
- AWS Fargate
- Amazon Elastic Container Services for Kubernetes (EKS)

Open Source Integrations

- Kubernetes (ExternalDNS)
- Istio (Pilot) – Tetrate.io
 - <https://www.tetrate.io/blog/istio-cloud-map-operator/>
- Consul – HashiCorp
 - <https://www.hashicorp.com/blog/enabling-service-discovery-consul-cloud-map>

AWS Cloud Map 上の Namespace の作成

選択できる Discovery 手段

- API calls
- API calls and DNS in VPCs
- API calls and public DNS

1. `aws servicediscovery create-http-namespace --name mydata.aws`

Create namespace [Info](#)

A namespace typically contains the services for one application.

Namespace configuration
The namespace configuration determines how your application discovers service instances

Namespace name
A friendly name lets you easily find a namespace on the dashboard.
 The namespace name can have up to 1,024 characters, and must start and end with a letter. Valid characters: a-z, A-Z, 0-9, _ (underscore), and - (hyphen)

Namespace description - optional
This description appears on your dashboard. It can help you quickly identify what your namespace is used for.
 The description can have up to 1,024 characters.

Instance discovery
Instance discovery determines how your application discovers registered instances.

API calls
Your application makes API calls to discover registered instances.

API calls and DNS queries in VPCs
Your application makes API calls or submits DNS queries in VPCs to discover registered instances. Additional charges apply.

API calls and public DNS queries
Your application makes API calls or submits public DNS queries to discover registered instances. Additional charges apply.

[Cancel](#) [Create namespace](#)

AWS Cloud Map へのサービス登録

1. aws servicediscovery create-service
--name mydynamodb
--http-config "NamespaceId=%namespace_id%"
2. aws servicediscovery register-instance
--service-id %service_id%
--instance-id instance-1
--attributes
ARN=arn:aws:dynamodb:us-west-
2:123456789012:table/users,
STAGE=beta,
VERSION=1.0,
READ_ONLY=false

Register service instance [Info](#)

Service instance information
AWS Cloud Map uses these settings in the specified service to register one service instance.

Instance type
When your application requests a service instance, Cloud Map returns the information that is required to access the instance.

Identifying information for another resource based on CNAME
This option is for resources that have service configured for CNAME

Service instance ID
You can use this value to update an existing service instance.
my-aurora-writer

Standard attributes

CNAME
cluster-identifier.cluster-xxxxxxxxx123.us-west-2.rds.amazonaws.com

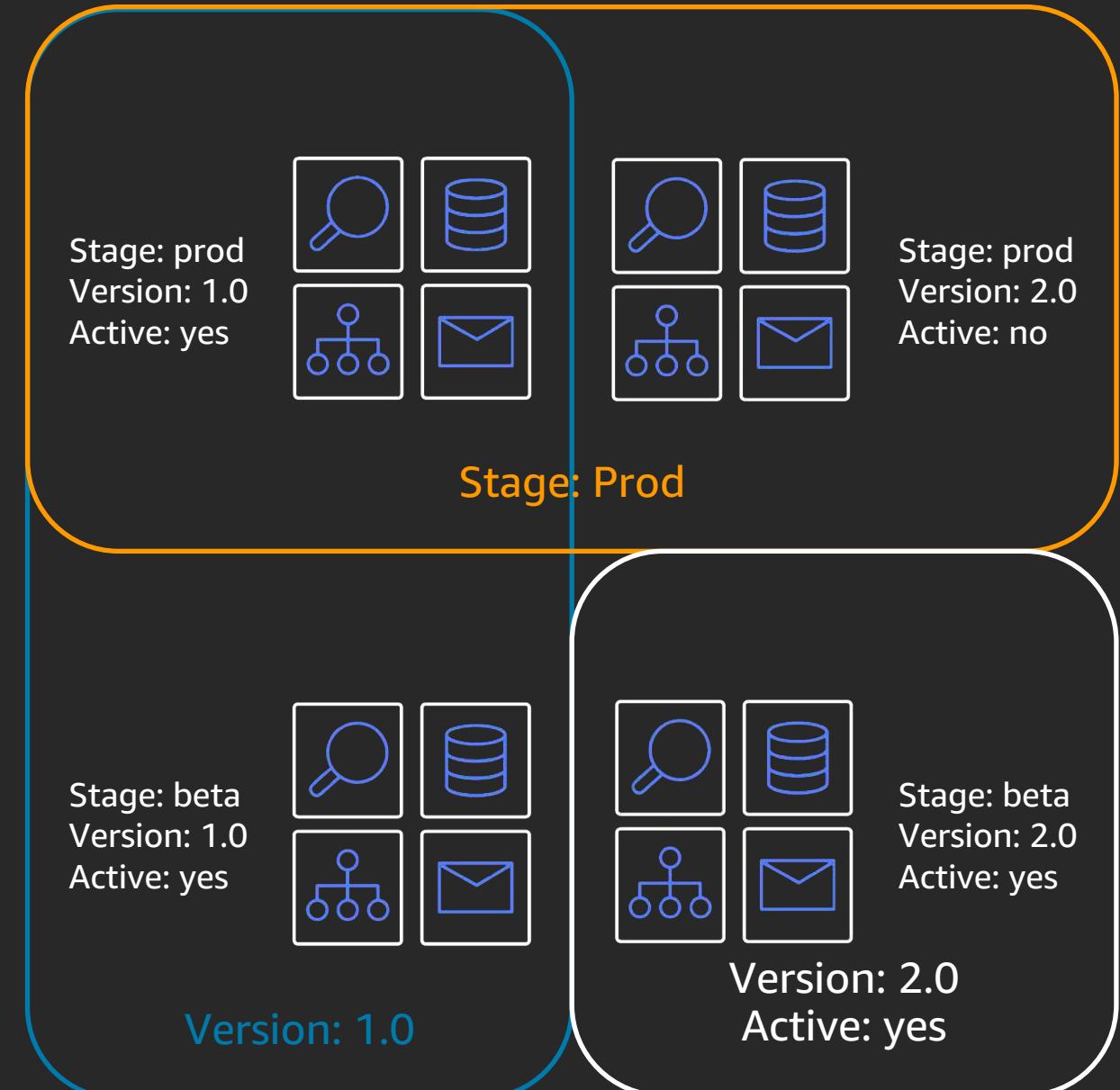
Custom attributes - optional
The key-value pairs that you specify here are associated with the service instance.

Key	Value
STAGE	demo

属性ベースのサービスディスカバリ

```
1. aws servicediscovery discover-instances  
  --namespace-name mydata.aws  
  --service-name mydynamodb  
  --query-parameters VERSION=1.0,  
  STAGE=beta  
--> {ARN=arn:aws:dynamodb:us-west-  
2:123456789012:table/users,  
  STAGE=beta, VERSION=1.0, READ_ONLY=false}
```

```
2. aws servicediscovery discover-instances  
  --namespace-name mydata.aws  
  --service-name mydynamodb  
  --query-parameters VERSION=2.0  
--> {}
```



AWS Cloud Map: Regions Supported

US East

- N. Virginia
- Ohio

US West

- N. California
- Oregon

Canada

- Central

Europe

- Frankfurt
- Ireland
- London
- Paris

Asia Pacific

- Mumbai
- Seoul
- Singapore
- Sydney
- **Tokyo**

AWS Cloud Map: 価格

シンプルかつ予測しやすい価格モデル

- 登録リソース: 1つにつき \$0.10 / 月
- ディスカバリ API コール: \$1.00 / 100万回

DNS を有効にしたネームスペースの作成・利用と

DNS クエリ、ヘルスチェックには Amazon Route 53 の価格モデルが適用されます。参照:

<https://aws.amazon.com/route53/pricing/>

価格の例

- 10 サービス、平均して月間 75 EC2 インスタンス

サービスレジストリ料金

- EC2 インスタンス: $75 \times \$0.10 = \7.50 / 月
- DynamoDB テーブル: $10 \times \$0.10 = \1.00 / 月

ディスカバリ API 呼び出し

- $75 \times \$1.00 \times 175,000 / 1,000,000 = \13.12 / 月

合計

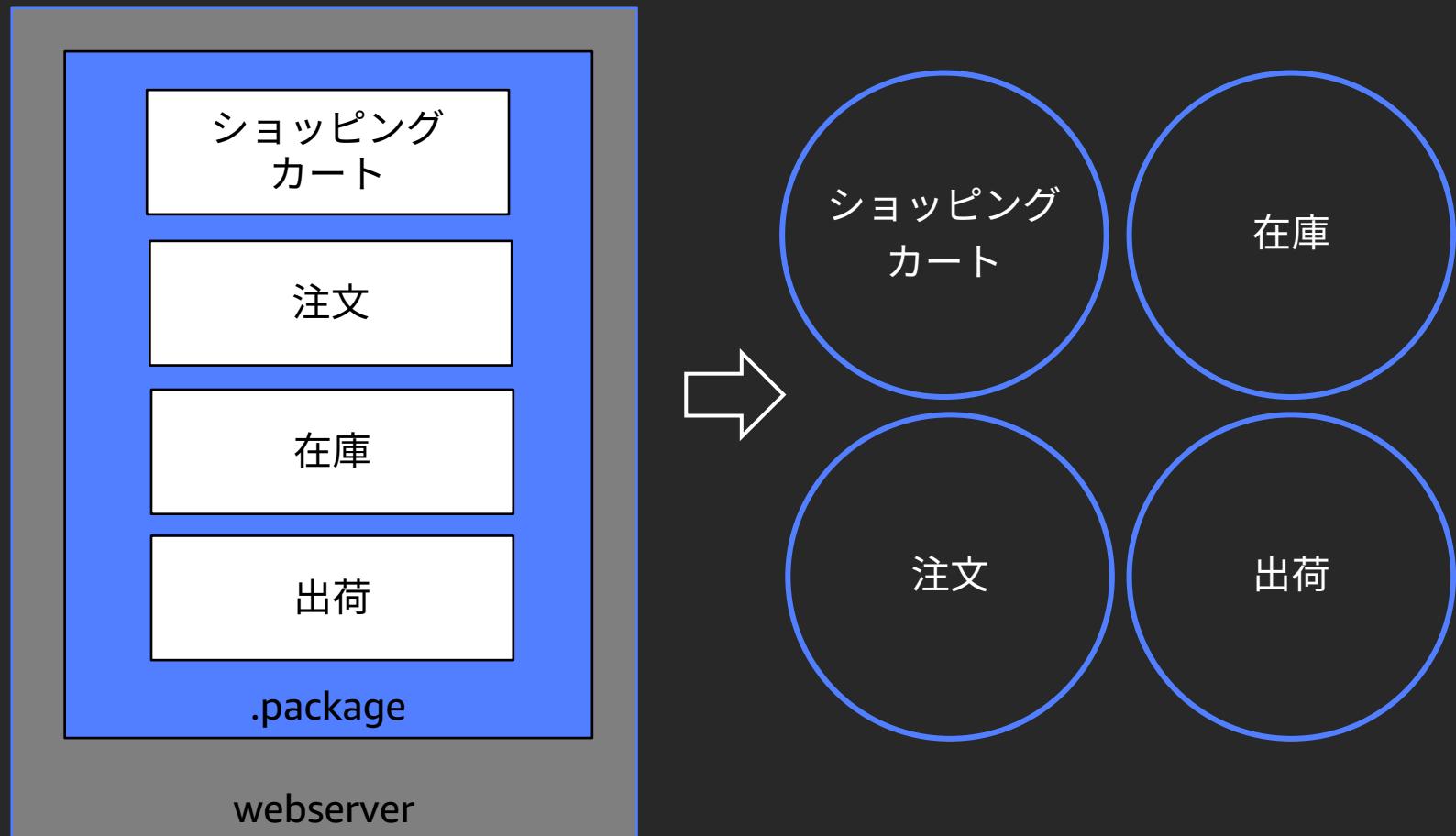
- $\$7.50 + \$1.00 + \$13.12 = \21.62 / 月

AWS App Mesh

マネージドサービスメッシュ

Microservices

- 自律的なチームによる開発・運用
- 一つのことを上手にこなす
- Polyglot(-able)
- 独立してスケール
- API による外部サービスと連携
- コンテナとの親和性

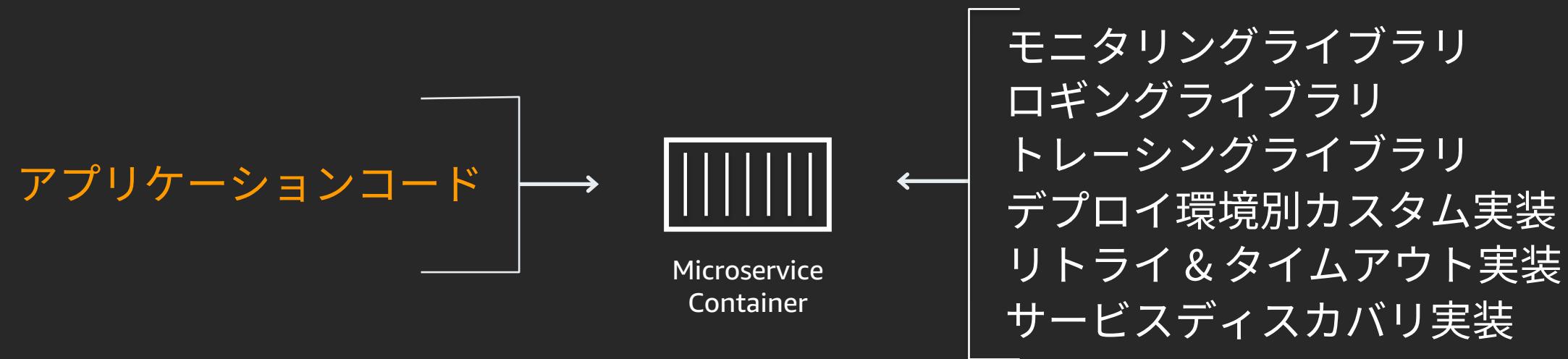


Microservices - 課題 -

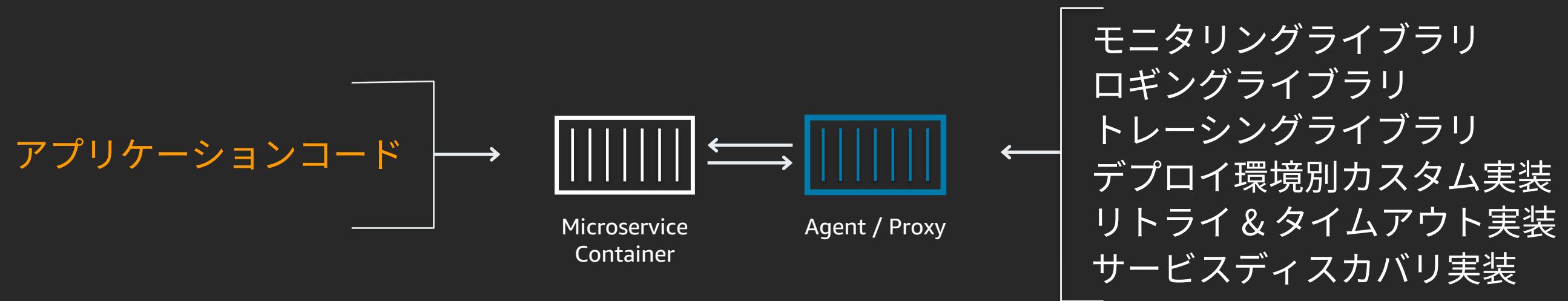
- ・複雑に連携するシステムからどのようにして問題の発生箇所を見つけるか？
- ・問題の発生したサービスが他のサービスに影響を与えないようにするには？
- ・各サービスが正しく他のサービスと通信できるように設定するには？
- ・サービス間の通信を可視化するには？
- ・Etc., etc.,

プログラミング言語や通信プロトコル、サービスの違いに関係なく
これらを一貫した方法で実現するには？

Option1 - 実装する -



Option2 - プロキシへのオフロード -



AWS App Mesh uses the Envoy Proxy



- Lyft 社で 2016 年に誕生
- 現在は CNCF 管理プロジェクト、先日 “Graduated” に
- 安定して動作し、多くの大規模プロダクション利用事例
 - Airbnb, Booking.com, eBay, Netflix, Pinterest, Square, Uber, Yelp

App Mesh による容易な連携:

- Amazon CloudWatch
- AWS X-Ray
- Envoy と連携可能なサードパーティのモニタリング・トレーシングツール群

Availability

現在は Public Preview のステータスで、以下のリージョンで利用可能

- N. Virginia
- Ohio
- Oregon
- Ireland

Preview v/s GA (capabilities)

Preview

API ready
For use with sample apps
not production
HTTP path based routing
Logs, metrics integrations

GA

Console
Integrations
Traffic management
AWS Cloud Map
Cross Account
Amazon EC2

Post GA

TLS
Ingress

Examples and Roadmap available on GitHub
<https://github.com/awslabs/aws-app-mesh-examples>

Amazon ECS

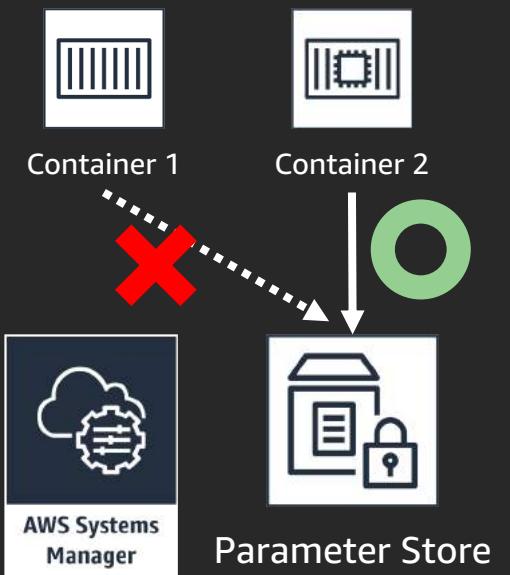
Amazon ECS, AWS Fargate がリソースのタグ付けに対応

- Amazon ECS と AWS Fargate の各種リソースにタグ付け可能に。コストの把握やリソースの用途把握が容易になった
 - ✓ サービス、タスク定義、タスク、クラスタ、コンテナインスタンスなどがタグ付けの対象となる
 - ✓ タグ付けには新形式の ARN とリソース ID が必要。オプトインすることで有効化できる
- クラスター/コンテナインスタンス/サービス/タスク/タスク定義それぞれにタグ付けが可能



Amazon ECS でタスクに対してより安全に秘密情報を渡せるように

- タスク定義に Systems Manager Parameter Store に格納されたパラメーターのフル ARN もしくは名前をセットすることで起動時にパラメーターが環境変数としてコンテナ内に展開される
 - ✓ Secrets Manager に格納されたパラメーターも参照可能
- これまでにはタスク定義にパラメーターそのものを環境変数として記述するか、コンテナの起動時に S3 や DynamoDB から取得してくるなどのワークアラウンドが必要だった
- EC2 モードと Fargate(v1.3) で共に利用可能



Amazon ECS で Blue/Green デプロイメント が可能に

- ロードバランサと連携したトラフィックのフリップ
 - 新たにプロビジョンされた Green タスク群にトラフィックを向ける
- Blue タスク群への高速なロールバック
 - 手動キャンセル
 - Lambda フック処理の失敗
 - CloudWatch アラーム連携
- コンソールや API でのデプロイメントステータスや履歴の確認
- Amazon SNS による通知や CloudWatch Events との連携
- "CodeDeploy-ECS" デプロイアクションを CodePipeline から利用可能
- "aws ecs deploy" コマンドによるデプロイの実行も可能



Amazon ECS で Blue/Green デプロイメントが可能に

Configure deployments

Deployment type* Blue/Green deployment (powered by CodeDeploy)

Application name* AppECS-canary-codedeploy-bg

Deployment group name* DgpECS-canary-codedeploy-bg

BeforeInstall hook

AfterInstall hook

BeforeAllowTraffic hook

AfterAllowTraffic hook

デプロイを停止 デプロイを停止してロールバック Terminate original task set

デプロイのステータス

ステップ 1:
Deploying replacement task set Completed 成功

ステップ 2:
Rerouting production traffic to replacement task set 100% traffic shifted 成功

ステップ 3:
Wait 1 時間 0 分 Waiting 進行中

ステップ 4:
Terminate original task set Not started

トラフィック移行の進行状況

オリジナル	置換
0%	100%
Original task set not serving traffic	Replacement task set serving traffic

Amazon ECS で ARM アーキテクチャがサポート

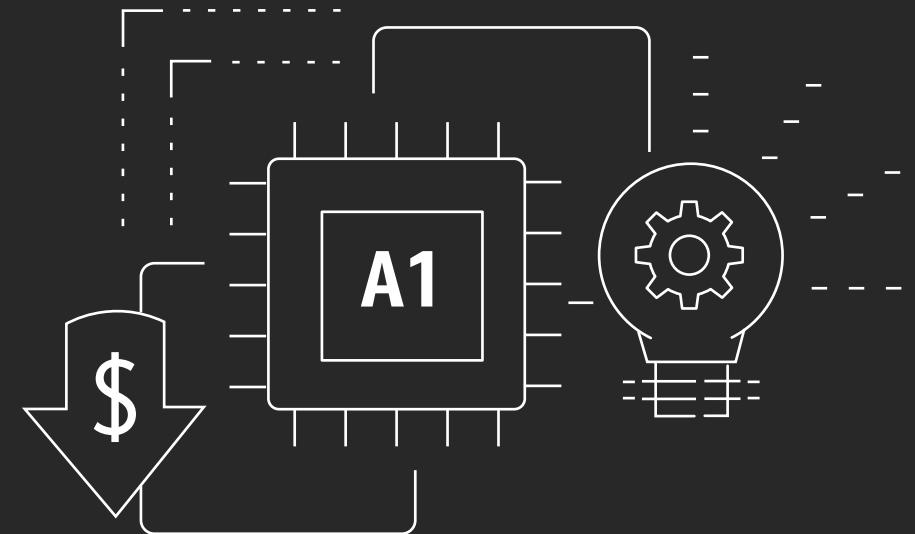
- EC2 モードで利用可能 (非 Fargate)
- ECS optimized AMI for ARM が利用可能
- 多くの OSS パッケージを ARM アーキテクチャ向けにコンパイル済み
- Python, Node.js, Ruby, Java などは大体動く
- コンテナを ARM アーキテクチャ上で再ビルドするだけで、すぐにコスト削減することが可能
- Golang はクロスコンパイルが必要だが、CI/CDのパイプライン内で容易に行うことが可能

例 : **GOARCH=amd64 GOOS=linux go build my_package**

A1インスタンス

ARM ベースの AWS Graviton プロセッサ搭載 A1インスタンス

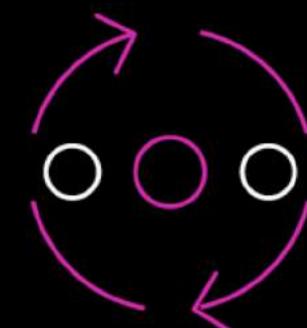
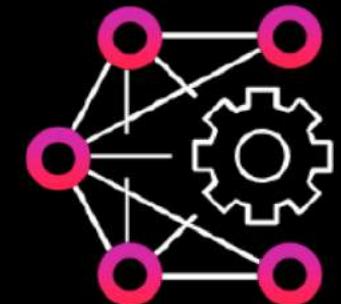
- **64-bit ARMアーキテクチャのEC2インスタンス**
- マイクロサービスや、ウェブサーバ、キヤツシユサーバなど、小規模インスタンスを多用する用途に最適
- 他ファミリと比較して最大45%のコスト削減を期待できる
- Amazon Linux 2, RHEL, Ubuntu のAMI がすでに利用可能で他も近日サポート開始予定
- バージニア、オレゴン、アイルランド、オハイオの各リージョンで利用可



a1ファミリ	vCPU	メモリ (GiB)	EBS帯域 (Gbps)	NW帯域 (Gbps)	コスト (\$/時)
a1.medium	1	2	Max 3.5	Max 10	0.0255
a1.large	2	4	Max 3.5	Max 10	0.0510
a1.xlarge	4	8	Max 3.5	Max 10	0.1020
a1.2xlarge	8	16	Max 3.5	Max 10	0.2040
a1.4xlarge	16	32	3.5	Max 10	0.4080

※コストはバージニアのものです。リージョン毎に金額は異なります

AWS Graviton プロセッサ



64ビットArm Neoverseコア搭載のAWSカスタム
シリコン

ターゲットとするワークLOADの最適化

カスタマーのために継続的なイノベーション・
ビルドを繰り返す

Amazon EC2 汎用インスタンス

T3/T3a
instances



時々高いCPU使用率を必要とする多くのワークロード

M5/M5a
instances



CPU、メモリおよびネットワークリソースそれぞれを
バランスよく使用するワークロード

A1 instances



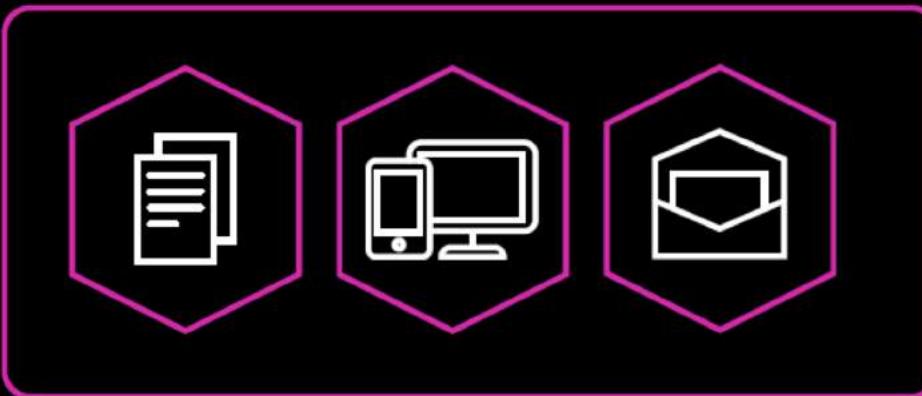
複数のCPUコアを複数インスタンスに跨ってスケール
アウトし、広範なARMエコシステムによってサポート
されるワークロード

Amazon EC2 A1のターゲットアプリケーション

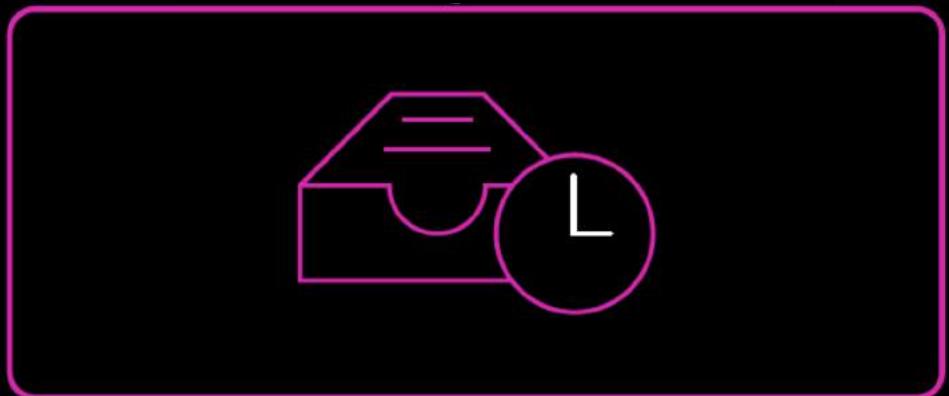
ウェブサーバー



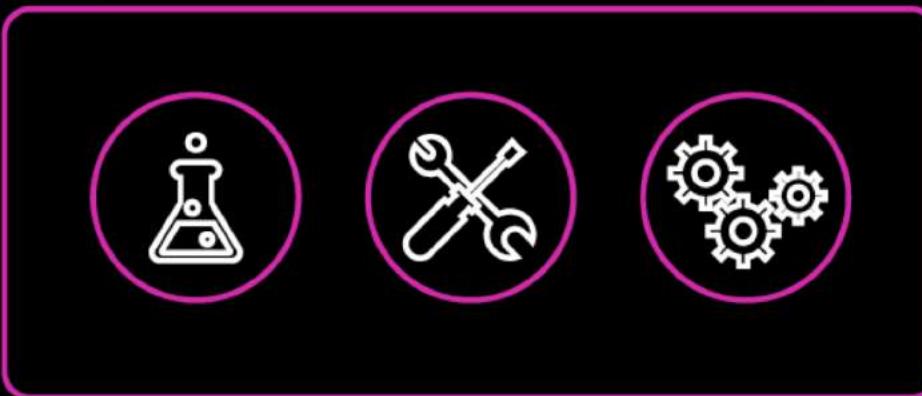
コンテナ マイクロサービス



キャッシュサーバー



開発/テスト環境



ソフトウェアエコシステム

OSVs and ISVs

Amazon Linux 2



Ubuntu 16.04 and newer



Red Hat Enterprise Linux

7.6 and newer



More coming soon

Containers

Most Docker official images support arm64



ECS

Available today

EKS

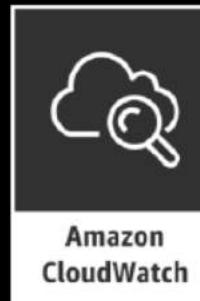
Coming soon



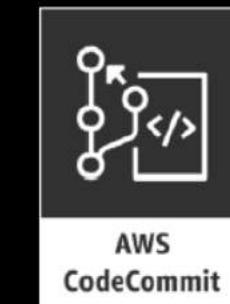
AWS Marketplace



AWS Systems Manager



Amazon CloudWatch



AWS CodeCommit



AWS Cloud9



AWS CodePipeline

Coming Soon: Amazon Corretto

AWS Fargate

AWS Fargate Platform Version 1.3 Adds Secrets Support

Posted On: Dec 17, 2018

[AWS Fargate](#) Platform Version 1.3.0 is now available. This update adds Secrets support when using Fargate launch type with [Amazon Elastic Container Service \(ECS\)](#).

As customers build applications, they need to reference sensitive information such as database credentials, tokens, configuration variables or SSH keys. Previously, customers had to directly reference this sensitive information in the task definition or manage your own run-time secrets with custom solutions to decouple secrets from core application logic stored in container images.

Now, you have new task definition conventions for accessing sensitive information stored in AWS Systems Manager Parameter Store . You can still set, get, update and delete secrets via existing AWS Systems Manager Parameter Store APIs. However, task definitions can now be used to designate which of those secrets should be exposed to what containers.

You can learn more about the feature in [ECS documentation](#). Also, please review the [Fargate documentation](#) to understand potential impact to tasks in cases of certain security updates to the underlying infrastructure.

Platform version 1.3* is available in all regions that AWS Fargate is available. See the [global region table](#) for more information on AWS regions and services.

Note: This feature is available via CLI/SDK and will be available in console for all Fargate regions by Friday, December 21, 2018. Additionally, if you are launching your tasks or services before this date, please use version 1.3 directly instead of the LATEST label.

Amazon EKS

Amazon EKS Adds ALB Support

Amazon EKS Adds ALB Support with AWS ALB Ingress Controller

Posted On: Nov 20, 2018

The AWS [ALB ingress controller project](#) is now generally available in version 1.0.0 and supported with [Amazon Elastic Container Service for Kubernetes \(EKS\)](#). This project allows you to use the [Elastic Load Balancing Application Load Balancer \(ALB\)](#) with your Kubernetes cluster managed by Amazon EKS.

Kubernetes allows you to configure an ingress controller in order to route traffic to containers, a management resource that authorizes external traffic to be routed within the Kubernetes application. Previously, support for the ALB ingress controller was a community driven effort that wasn't formally supported by AWS.

Now, the AWS ALB ingress controller is formally maintained and supported by AWS and is in production use by multiple customers. You can use the ingress controller with Amazon EKS or any Kubernetes cluster running on AWS.

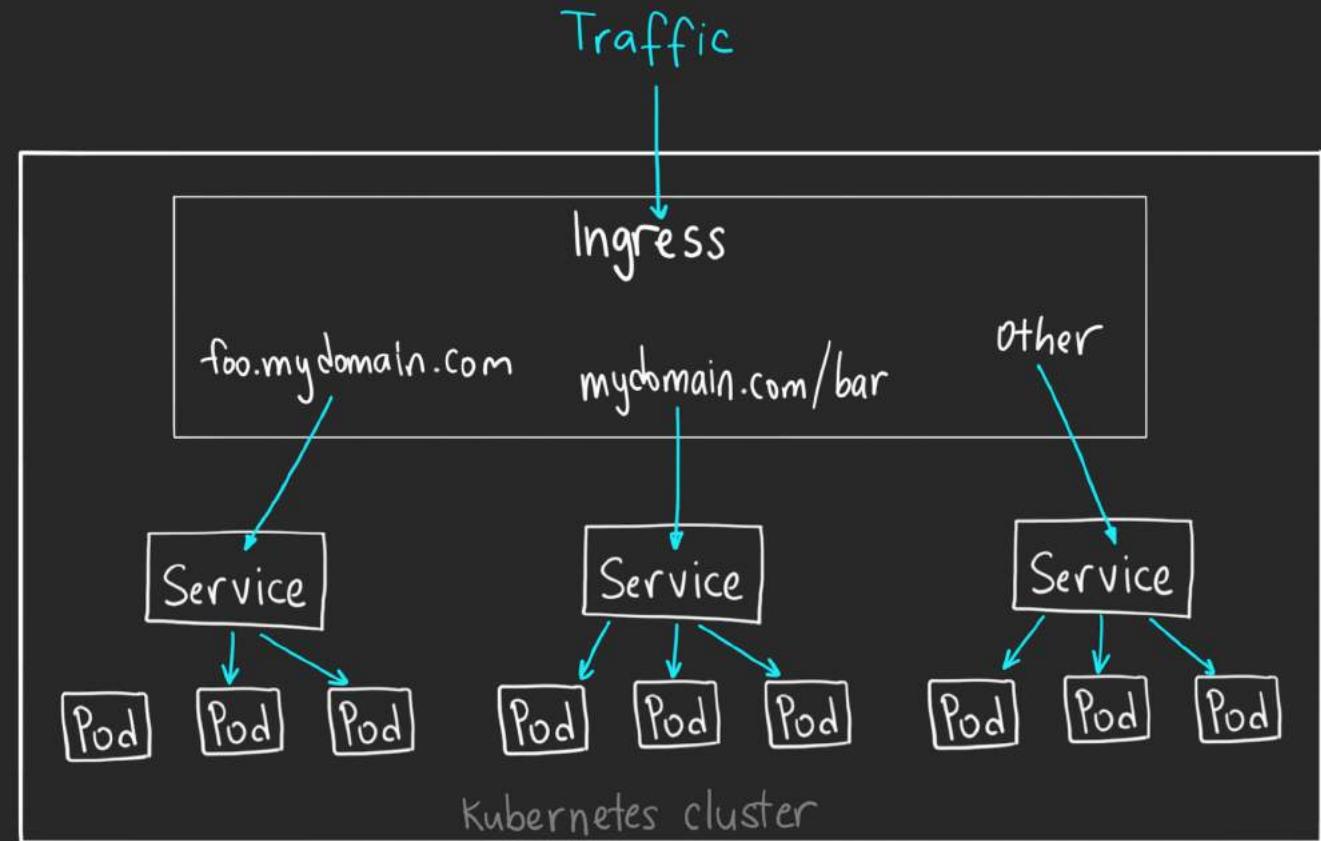
To learn more, read our [blog](#).

For implementation details, visit the AWS ALB ingress controller [GitHub project](#).

Please visit the [AWS region table](#) to see all AWS regions where Amazon EKS is available.

Amazon EKS Adds ALB Support

- Kubernetes クラスタ上でホスト、パスベースのルーティングを利用可能
- Kubernetes クラスタから ALB の展開と管理が可能
- ALB は複数のサービスのフロントでスマートルータあるいはエントリーポイントとして機能する
- L7 の柔軟なルーティング機能を利用可能



Kubernetes 1.11 のサポート

- 1.11 が利用可能 (CVE-2018-1002105 対応済み)
- EKS は最大3つのバージョンの Kubernetes をサポート
 - Deprecation となったバージョンで新規のクラスタを作成することは出来ない
- プラットフォームバージョンは、API サーバ構成の変更または Kubernetes パッチバージョンを表す
- 同一の Kubernetes バージョン内でのみプラットフォームバージョンはインクリメントされる

K8s 1.10

eks.1

eks.2

eks.3

K8s 1.11

eks.1

eks.2

K8s 1.12

eks.1

EKS Kubernetes Version Updates

- 新しい `UpdateClusterVersion` API は Kubernetes のインプレース アップデートをサポート
- EKS API に "update" オブジェクトが追加
- `ListUpdates` や `DescribeUpdate` API を利用して特定の更新プログラムの状態を可視化できるよう

Amazon ECR

CodePipeline で ECR を Source として利用可能に

- ・ 多くのお客様は、ベースイメージやアプリケーションにバンドルされるサイドカーアイメージを管理されています
- ・ 今まで、これらのイメージが更新されたときに CodePipeline をトリガーする直接的な方法はありませんでした
- ・ CodePipeline は ECR をパイプラインソースとして使用できるようになり、自動的にリリースを開始し、最新のソースコードを取得します
- ・ アプリケーションソースコードまたはベースイメージのいずれかが変更されると、CodePipeline はリリースを開始し、新しいアプリケーションイメージを構築します

ECR Console v2

Compute

Amazon Elastic Container Registry

Easily store, manage, and deploy container images

Amazon Elastic Container Registry (ECR) is a fully-managed container registry that makes it easy for developers to store, manage, and deploy container images.

How it works

The diagram illustrates the ECR workflow:

- Write code and package code into a Docker image.
- Push the Docker image to Amazon ECR.
- Compress, encrypt, and commit access to images.
- Version, tag, and manage image versions.
- Run containers by pulling images and running containers unhosted or using Amazon ECS or AWS Lambda.

Pricing (US)

You pay only for the amount of data you store in your repositories and data transferred to the Internet.

[Learn more](#)

Create a repository

[Get Started](#)

ECR > Repositories

Repository name	URI	Created at
firebase-load-testing	310511981941.dkr.ecr.ap-northeast-1.amazonaws.com.firebaseio-load-testing	07/06/18, 11:31:11 PM
mu-25-ecsdemo-frontend	310511981941.dkr.ecr.ap-northeast-1.amazonaws.com/mu-25-ecsdemo-frontend	10/23/18, 8:26:37 AM
scorekeep-api	310511981941.dkr.ecr.ap-northeast-1.amazonaws.com/scorekeep-api	08/28/18, 11:28:38 AM
scorekeep-frontend	310511981941.dkr.ecr.ap-northeast-1.amazonaws.com/scorekeep-frontend	08/28/18, 11:28:39 AM
slash-test/scorekeep-api	310511981941.dkr.ecr.ap-northeast-1.amazonaws.com/slash-test/scorekeep-api	12/06/18, 2:44:52 PM

Other Notable Announcements



AWS Marketplace for Containers

- ECS コンソール、AWS マーケットプレイスから ISV のコンテナプロダクトの検索と購入が可能
- ECS, EKS, Fargate に対応
- 様々なカテゴリのプロダクトがすでに公開されている。
 - e.g. HPC、セキュリティ、開発者ツール
- SaaS プロダクトも
- 無料、BYOL、月額タイプや使った分だけ課金などのタイプがある
- コンテナマーケットプレイスのプロダクトについての請求は毎月の AWS からの請求に合算される



Firecracker

- ・ マルチテナントなコンテナワークロードを実現するための KVM ベースの新しい仮想化技術
- ・ トライディショナルな仮想マシン同様のセキュアなアイソレーションを実現
- ・ OSS として公開 <https://firecracker-microvm.github.io/>
- ・ <125ms でユーザー空間にてアプリケーションコードを起動可能
- ・ 1ホストあたり 150 microVM/sec 起動をサポート
- ・ AWS 上ではベアメタルインスタンスで実行可能。オンプレミス、ラップトップでも。仮想マシン上で動かす場合は Nested Virtualization のサポートが必要。
- ・ 現時点では Intel プロセッサのみをサポート。AMD と ARM サポートは 2019 年予定。

[Experimental] Public Roadmap (Github)

 [aws / containers-roadmap](#)

[Watch](#) 112 [Star](#) 500 [Fork](#) 11

[Code](#) [Issues 51](#) [Pull requests 0](#) [Projects 1](#) [Insights](#)

This is the public roadmap for AWS container services (ECS, ECR, Fargate, and EKS). <https://aws.amazon.com/containers/new>

23 commits 2 branches 0 releases 5 contributors Apache-2.0

Branch: [master](#) ▾ [New pull request](#) [Find file](#) [Clone or download](#) ▾

Commit	Message	Time
 abby-fuller	Add contributing instructions to README	Latest commit 38f7577 2 days ago
 .github	Update community-request-.md	3 days ago
 CODE_OF_CONDUCT.md	Creating initial file from template	20 days ago
 CONTRIBUTING.md	Update contributing guidelines	2 days ago
 LICENSE	Creating initial file from template	20 days ago
 NOTICE	Creating initial file from template	20 days ago
 README.md	Add contributing instructions to README	2 days ago

 README.md

Containers Roadmap

This is the public roadmap for AWS container services (ECS, ECR, Fargate, and EKS).

Experimental Public Roadmap - AWS Container Services

Thank you!

Kazuki Matsuda