サービスディスカバリと AWS Cloud Map

AWS Black Belt Online Seminar

AWS Container Specialist
Principal Solution Architect
荒木靖宏

2021-July

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

このセッションで扱うこと

- なぜサービスディスカバリが必要か
- サービスディスカバリ/サービスレジストリパターン
- AWS Cloud Map

Monolith
すべてを実行

Microservices
ひとつのことを実行

aws

Monolith
すべてを実行

唯一のサービス
死ぬ時は全てが死ぬ

aws

個々のサービスは個々に死ぬ
使えるサービスを呼ぶ必要がある

Microservices
ひとつのことを実行

aws

個々のサービスは個々に死ぬ
使えるサービスを呼ぶ必要がある
↓
個々のサービスの生死を知る
「サービスレジストリ」を使って
「サービスディスカバリ」する

Microservices
ひとつのことを実行

aws

サービス ディスカバリ/サービス レジストリパターン

aws

サービス ディスカバリ/サービス レジストリパターンとは

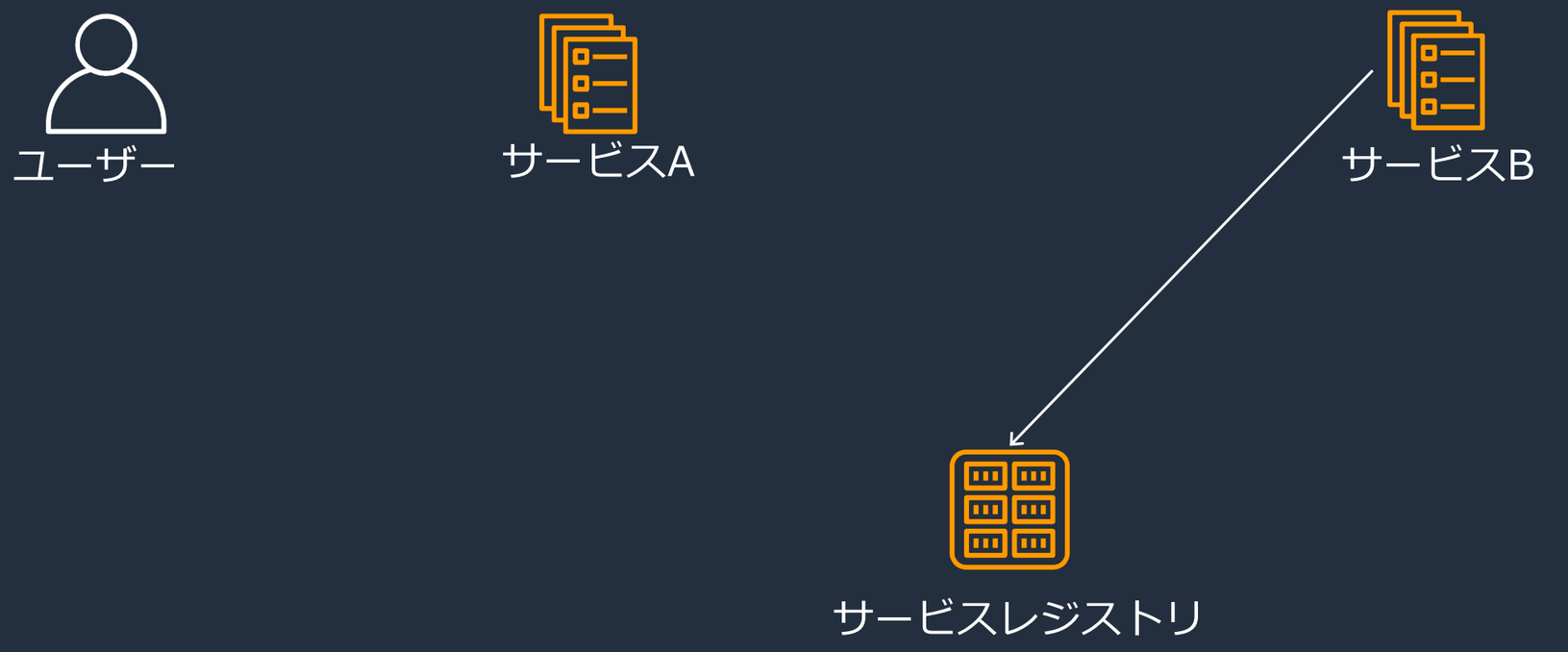
- サービスが他のサービスを利用する際の課題を解決するパターン

- **サービスレジストリ**は、サービス情報の登録場所
  - 個々のコンテナやインスタンスの起動時に、呼び出されるサービスがそのサービス自体の情報を登録する

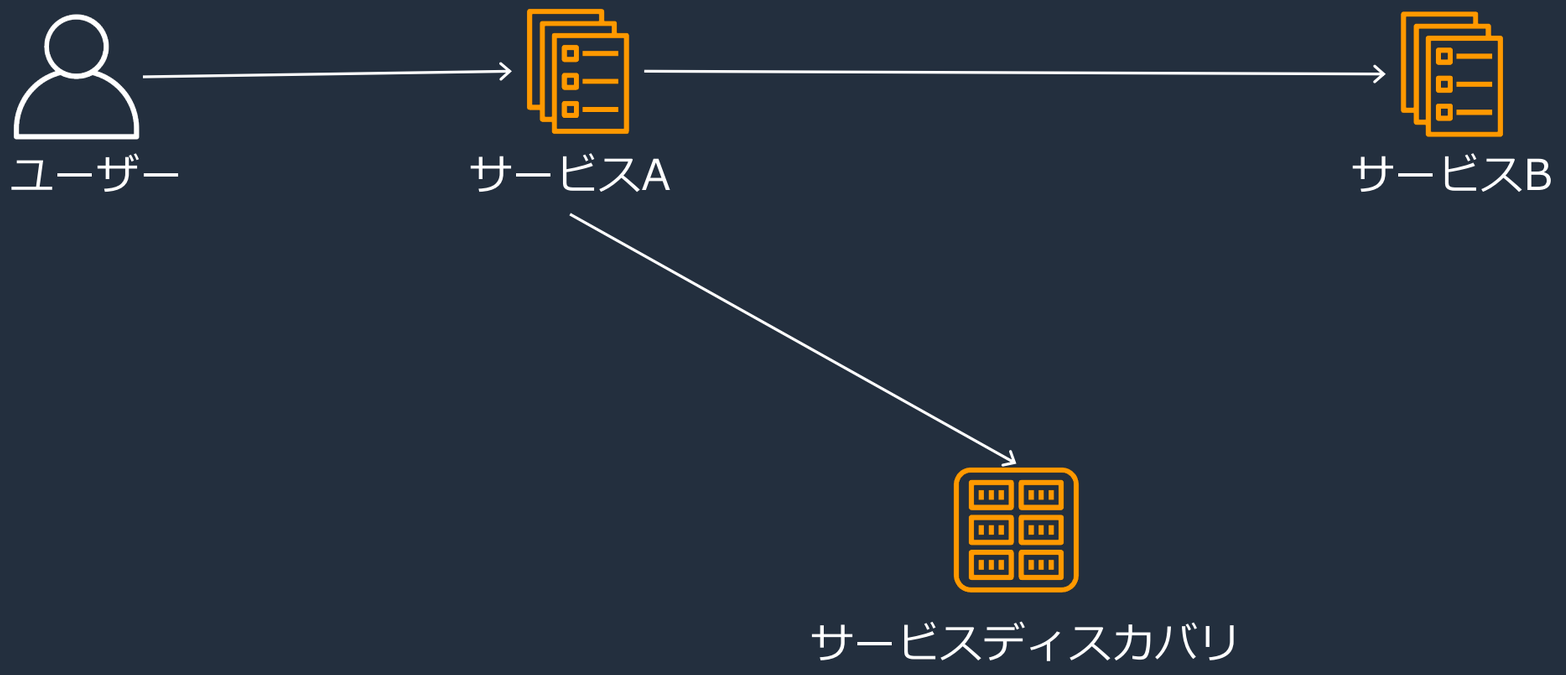
- **サービスディスカバリ**は、動的に変更するサービス情報の取得

aws

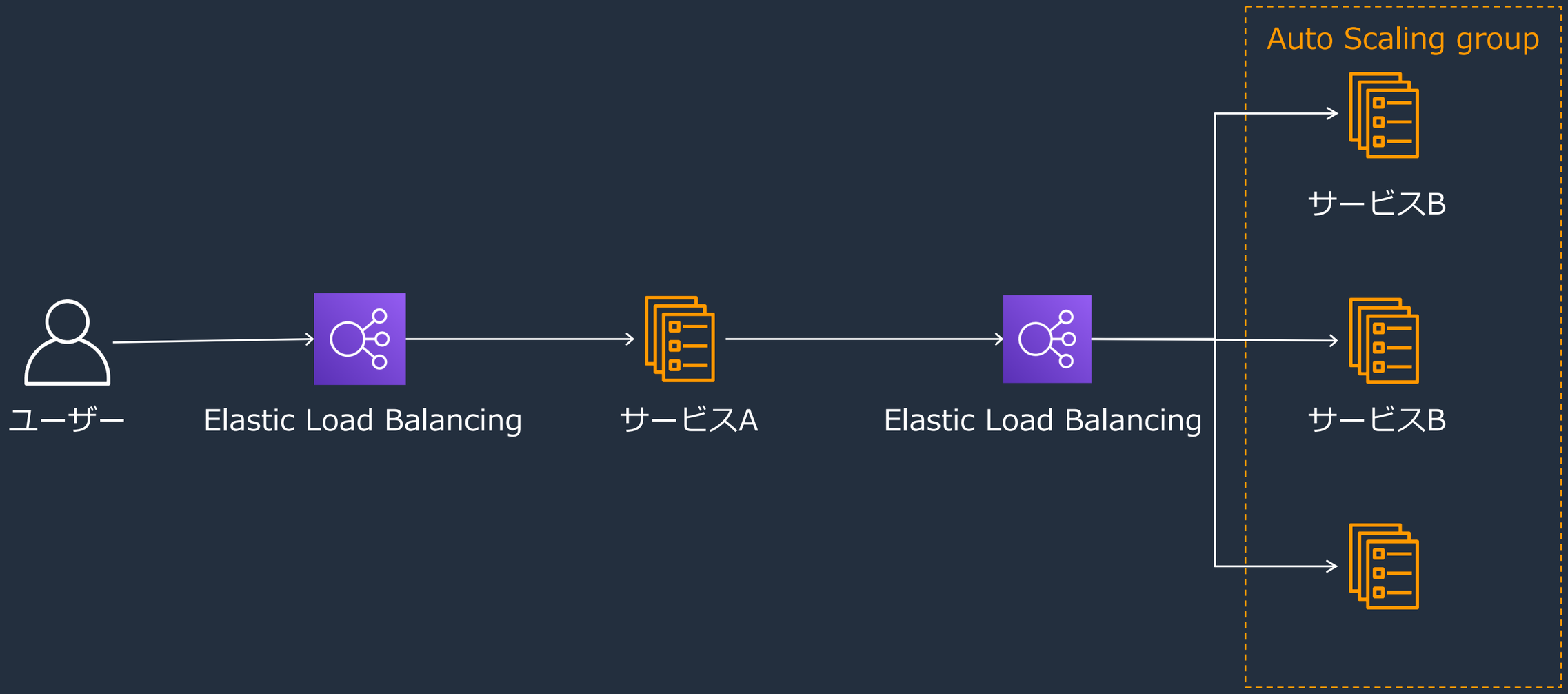
サービス レジストリパターン



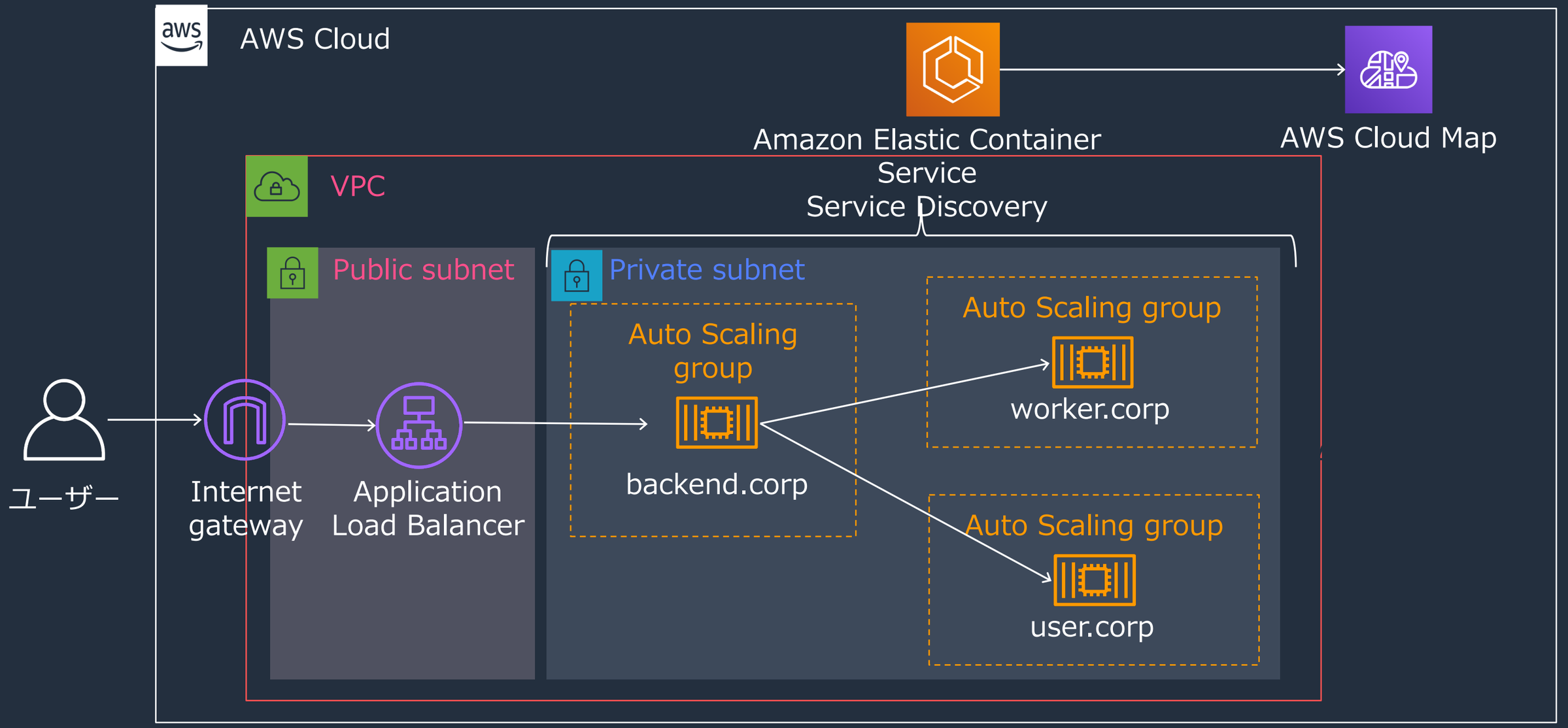
サービス ディスカバリパターン



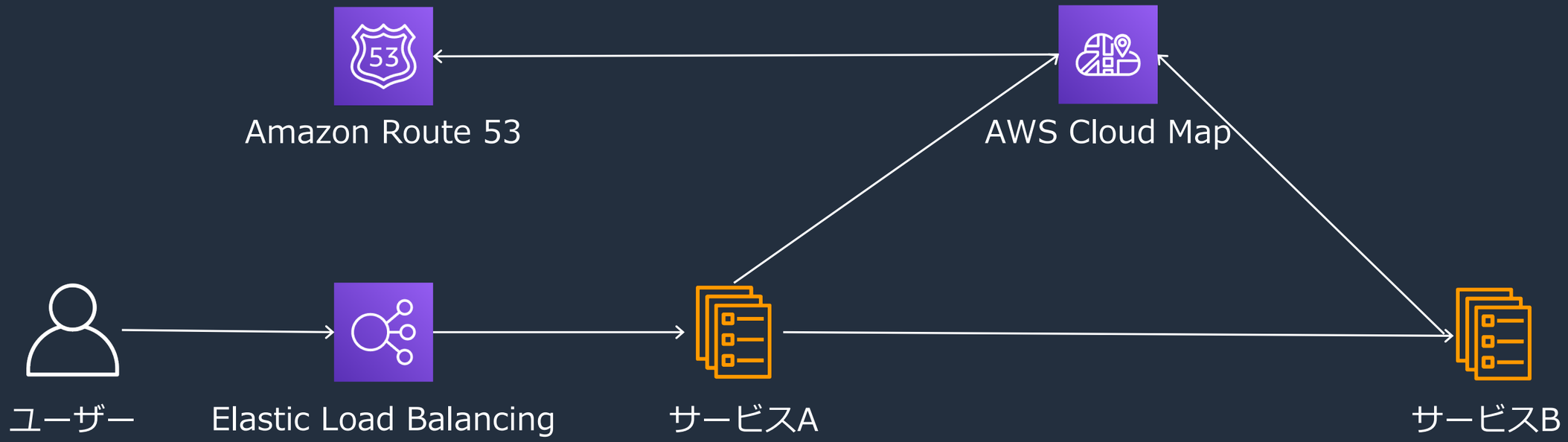
実装例１：Elastic Load Balancing を利用



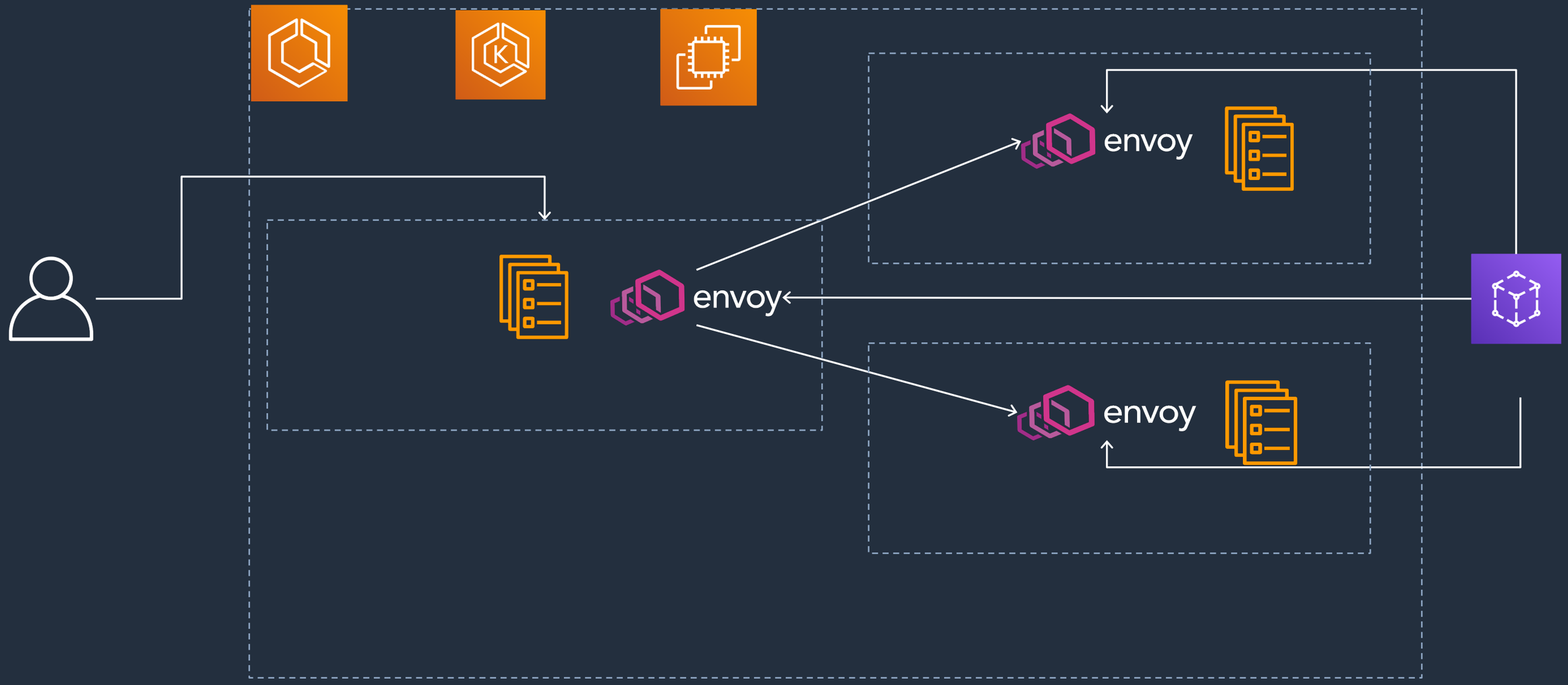
実装例2： Amazon ECSのサービスディスカバリを利用



実装例3： AWS Cloud Map の利用



実装例４： AWS App Meshのサービスメッシュのサービスディスカバリ



AWS Cloud Mapの主な機能の紹介

AWS Cloud Map

すべてのクラウドリソースのレジストリ

高速で安全な名前解決

属性ベースの検出

Amazon Route 53 ヘルスチェックを使った一時的な障害の処理

AWS クラウドおよびオープンソースソリューションとの統合

aws

AWS Cloud Map:利点

高速かつ安全	• 99.99% のSLA可用性を持つリージョナルサービス • 高速な変更 — 5 秒未満 • 認証されたサービスディスカバリコール • 低レイテンシーディスカバリコール — 平均 2 ms
シンプルなサービス検出	• すべてのコンテナサービスでの統合サービス検出 • 属性ベースのサービス検出 • 論理的でわかりやすい名前でサービスの場所と構成パラメータを検出する
名前解決	• ELB の CNAME および A レコードのサポート • 複数値の応答ルーティング（最大 8 つのアドレスを返します） • IP ベースリソースの DNS ディスカバリモード（オプション）
容易な管理性	• リージョン間でアプリケーションを簡単に拡張可能 • IP ベースのリソースの Amazon Route 53 ヘルスチェックを設定する • AWS およびオープンソースソリューションとの統合

AWS Cloud Map:エコシステム

AWS インテグレーション

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

オープンソースインテグレーション

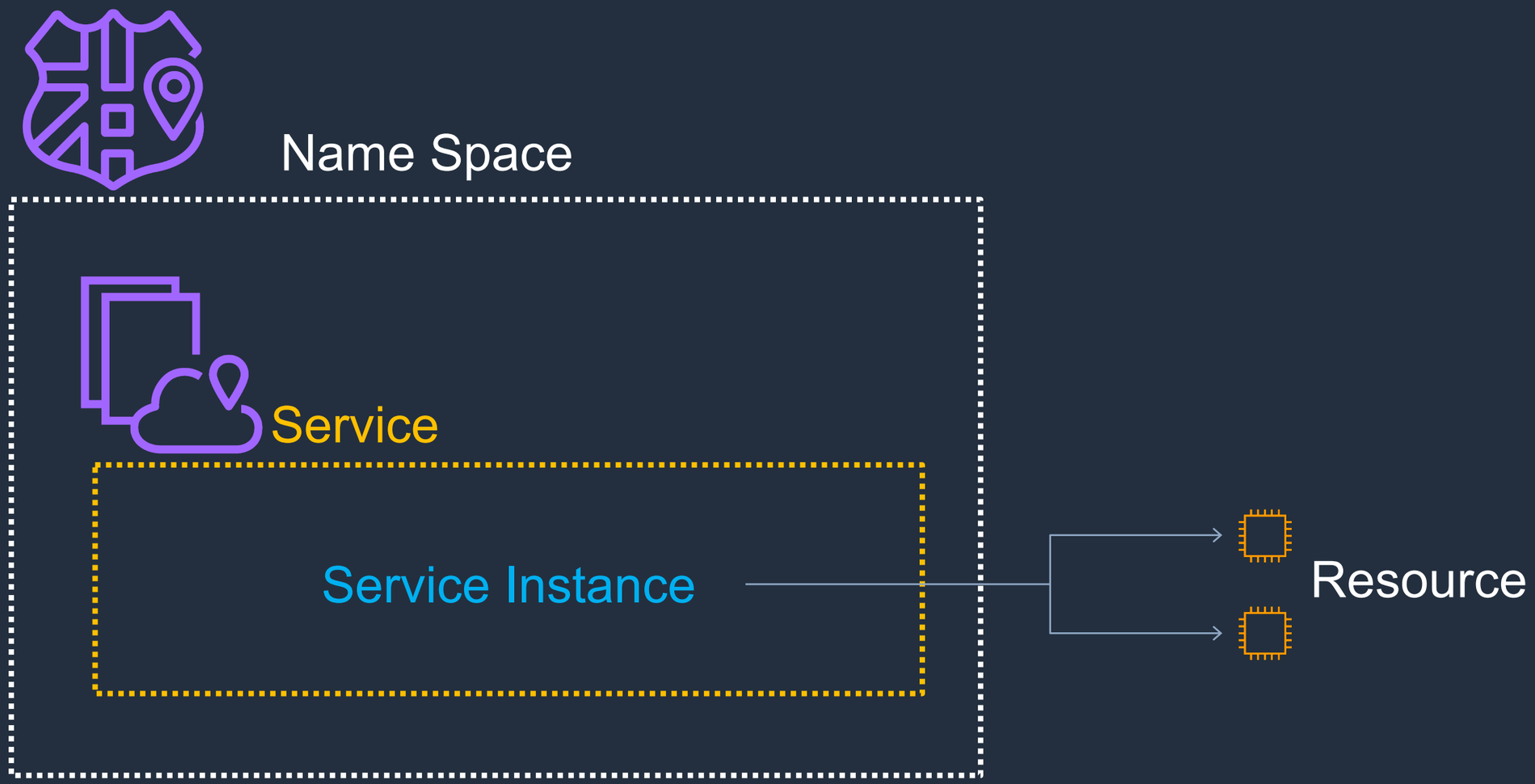
- Kubernetes （via ExternalDNS）
- Istio (via Pilot) — Tetrate.io
- Consul.io — HashiCorp

aws

AWS Cloud Map: 用語

- Name Space
  - サービスの論理グループ
  - 使用方法を指定：APIのみ、APIとVPCのDNS、APIとパブリックDNS
- サービス
  - サービスインスタンスを登録するためのテンプレート
  - DNSレコードタイプや、ヘルスチェック方法を指定する
- サービスインスタンス
  - 実際のリソースにアクセスする方法がある

aws



aws

API + パブリックDNS ディスカバリモードのリソース登録

```
1. aws servicediscovery create-public-dns-namespace --name
cmtest.araki.net

2. aws servicediscovery create-service --name frontend
   --dns-config “NamespaceId=${NAMESPACE_ID} DnsRecords=[{Type=A,
TTL=60}]”

3. aws servicediscovery register-instance --service-id
${SERVICE_ID} --instance-id ${SERVICE_INSTANCE}
   --attributes
   AWS_INSTANCE_IPV4=10.7.20.100,
   AWS_INSAANCE_PORT=8080
```

デモ

API ディスカバリのクラウドリソース登録

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

2. aws servicediscovery create-service --name logs --namespace-id
%namespace_id%

3. aws servicediscovery register-instance --service-id
%service_id% --instance-id %id%
   --attributes
   ARN=arn:aws:s3:::cloudmapdemoservicelogsbeta1,
   stage=beta,
   shard=s_1,
   read_only=no,
   path=/mylogs
```


aws

APIコールによるセキュアな名前解決

```
aws servicediscovery discover-instances --namespace-name shared --service-
name logs
-->
{    "Instances": [
        {
            "InstanceId": "i1",
            "NamespaceName": "shared",
            "ServiceName": "logs",
            "HealthStatus": "UNKNOWN",
            "Attributes": {
                "read_only": "no",
                "path": "/mylogs",
                "shard": "s_1",
                "ARN": "arn:aws:s3:::cloudmapdemoservicelogsbeta1",
                "stage": "beta”
            }
        }
    ]
}
```

Python SDKを利用したサービスインスタンスの登録例

```
def register_service():
    client = boto3.client('servicediscovery')

    response = client.register_instance(
        ServiceId = SERVICE_ID,
        InstanceId = INSTANCE_ID,
        Attributes = {
            'version': '1.0',
            'stage': 'prod',
            'AWS_INSTANCE_IPV4': '34.209.47.250',
            'AWS_INSTANCE_PORT': '80'
        }
    )
    return response
```


aws

Python SDKを利用したサービスインスタンスの取得例

```
def discover_services():
    client = boto3.client('servicediscovery')

    response = client.discover_instances(
        NamespaceName = NAMESPACE_NAME,
        ServiceName = SERVICE_NAME,
        QueryParameters = {
            "version": "1.0",
            "stage": "prod"
        }
    )
    return response
```


aws

このセッションで扱ったこと

- なぜサービスディスカバリが必要か
  → サービスが他のサービスを利用する際の課題を解決
- サービスディスカバリ/サービスレジストリパターン
  → 分散システムの前提となる考え方とその実装
- AWS Cloud Map
  → アプリケーションのリソースに論理名をマッピングできるマネージドサービス。サービスディスカバリにはAWS SDK、RESTful API コール、または DNS クエリを使用可能。