AWS Black Belt Online Seminar

Elastic Load Balancing

Yosuke Okumura

Network Solutions Architect
2023/05

自己紹介

名前：奥村 洋介

所属：ネットワークソリューション部
ネットワークソリューションアーキテクト

経歴：回線キャリアやISPで10年ほどエンジニア

好きなAWSサービス：Amazon Route 53, AWS Transit Gateway

本セミナーの対象者

Elastic Load Balancingのうち、Application Load BalancerとNetwork Load Balancerを一から勉強したい方。

アジェンダ

- Elastic Load Balancing(ELB)の基本
- 高可用性と負荷分散
- コネクションの設計
- ヘルスチェック/モニタリング
- セキュリティ
- IPv6
- ALB固有機能
- NLB固有機能
- 他サービス連携
- 負荷テスト
- 料金

Elastic Load Balancer(ELB)の基本

Elastic Load Balancing (ELB) とは

Amazon VPC上のロードバランサー

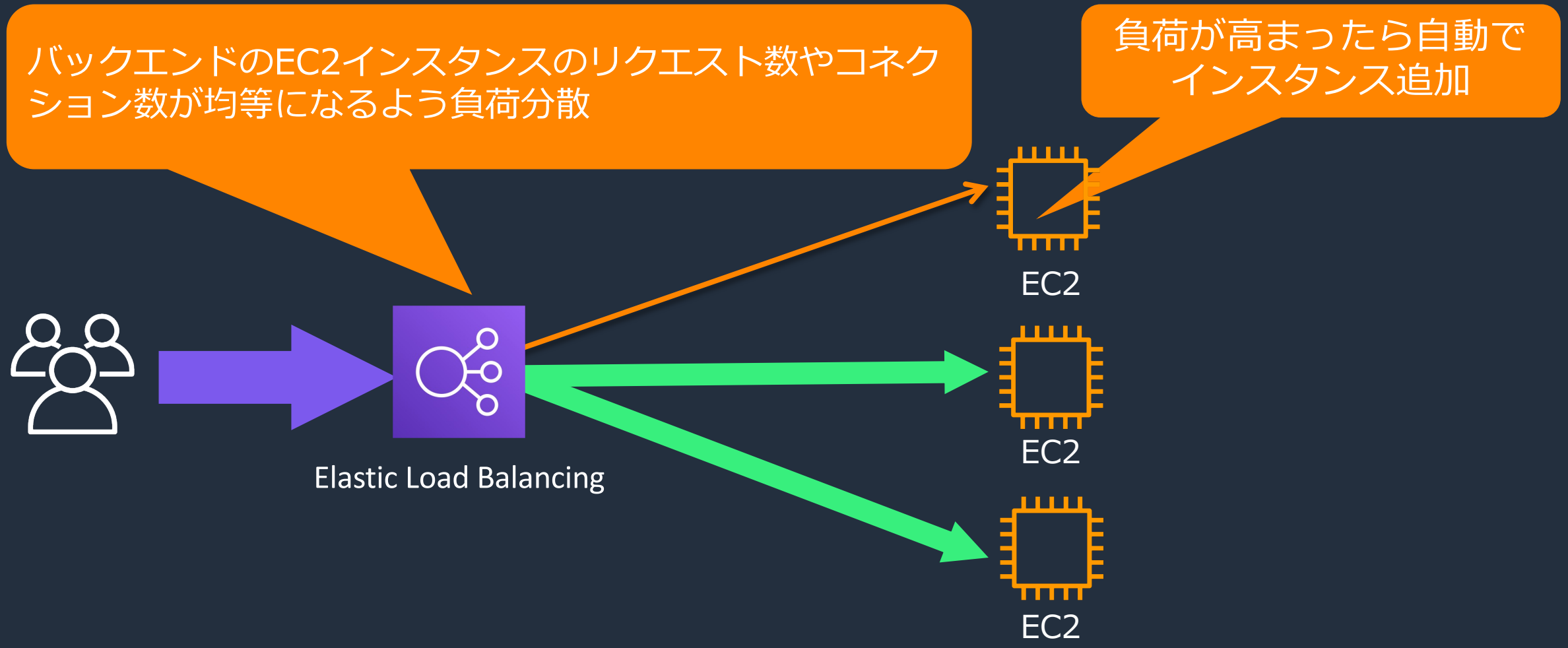
ELBを使う目的

- **スケーラビリティ**：複数のEC2インスタンス/ECSコンテナ..etc（ターゲット）に負荷分散
- **アベイラビリティ**：複数のアベイラビリティゾーンにある複数のターゲットの中から正常なターゲットにのみ振り分け

ELBの特徴

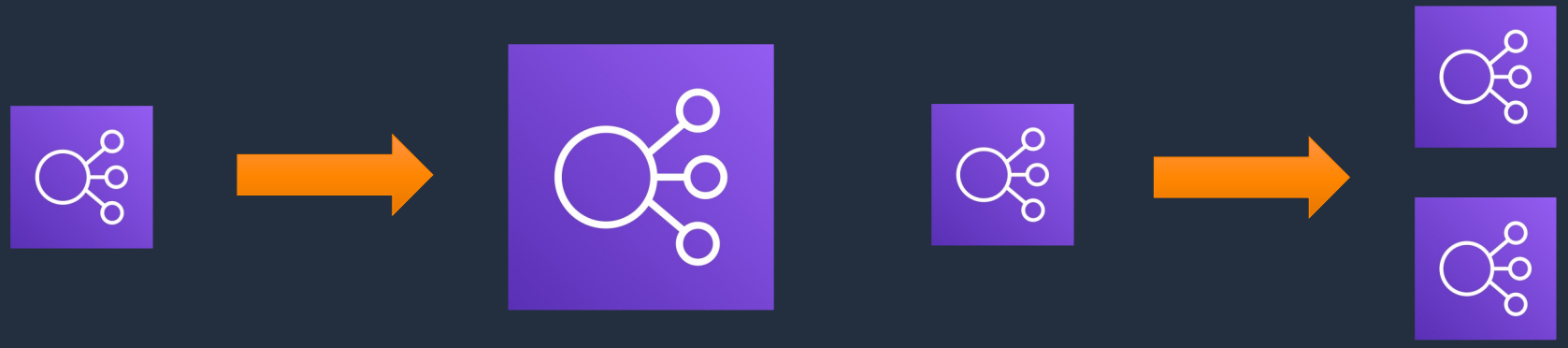
- **オートスケール**：ELB自体やターゲットの負荷に応じてキャパシティを自動的に増減
- **従量課金**：従量課金で利用可能
- **マネージド**：マネージドサービスなので細かい運用が不要
- **豊富なサービス連携**：Auto Scaling, Route 53, CloudFront等

負荷分散してスケーラブルなシステムを



ELB自体もスケーラブル

ELB自体も負荷の増減に応じて自動でスケールします。
（キャパシティが自動で増減する）



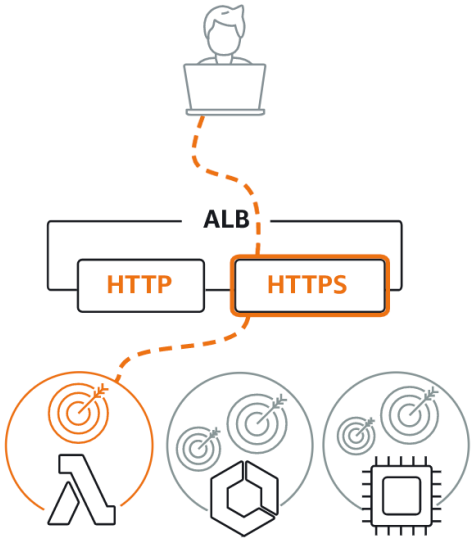
スケールアップ　　スケールアウト

[Tips]
ALBはスケールアウトするとき、IPアドレスが増減する。アクセスするときには必ずDNS名で！独自ドメインでのアクセスも可能。**Route 53でAliasレコードを利用すれば、Zone Apexも利用できる。**

ELBの種類

このセッションの対象

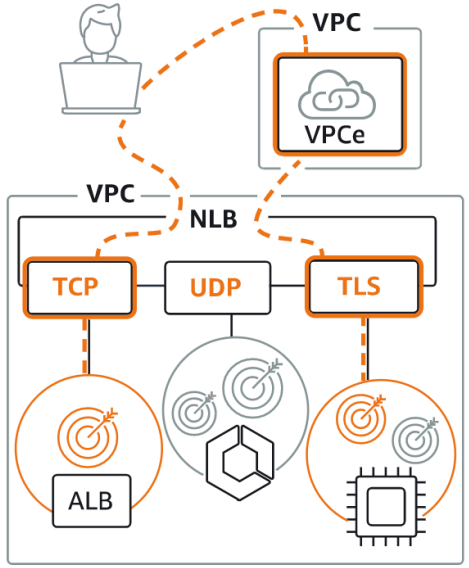
Application Load Balancer 情報



HTTP および HTTPS トラフィックを使用するウェブアプリケーション用に柔軟性の高い機能セットが必要な場合は、Application Load Balancer を選択します。Application Load Balancer はリクエストレベルで動作し、マイクロサービスとコンテナを含む、アプリケーションアーキテクチャを対象とした高度なルーティングおよび可視性機能を提供します。

作成

Network Load Balancer 情報



非常に高いパフォーマンス、大規模な TLS のオフロード、証明書のデプロイの一元管理、UDP のサポート、およびアプリケーションの静的 IP アドレスが必要な場合は、Network Load Balancer を選択します。Network Load Balancer は接続レベルで動作し、非常に低いレイテンシーを維持しながら、1 秒あたり数百万のリクエストを確実に処理することができます。

作成

Gateway Load Balancer 情報

GENEVE をサポートするサードパーティーの仮想アプライアンスのフリートをデプロイおよび管理する必要がある場合は、Gateway Load Balancer を選択します。これらのアプライアンスを使用すると、セキュリティ、コンプライアンス、ポリシー制御を向上させることができます。

作成

＋Classic Load Balancer
（前の世代）

ALBとNLBの位置付け

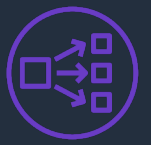
Elastic Load Balancing
(ELB)

以後、このセッションでは、ELBと表記した場合ALBとNLBを指します。

Application Load Balancer
(ALB)

HTTP, HTTPS, HTTP/2

L7 のコンテントベースのロードバランサー

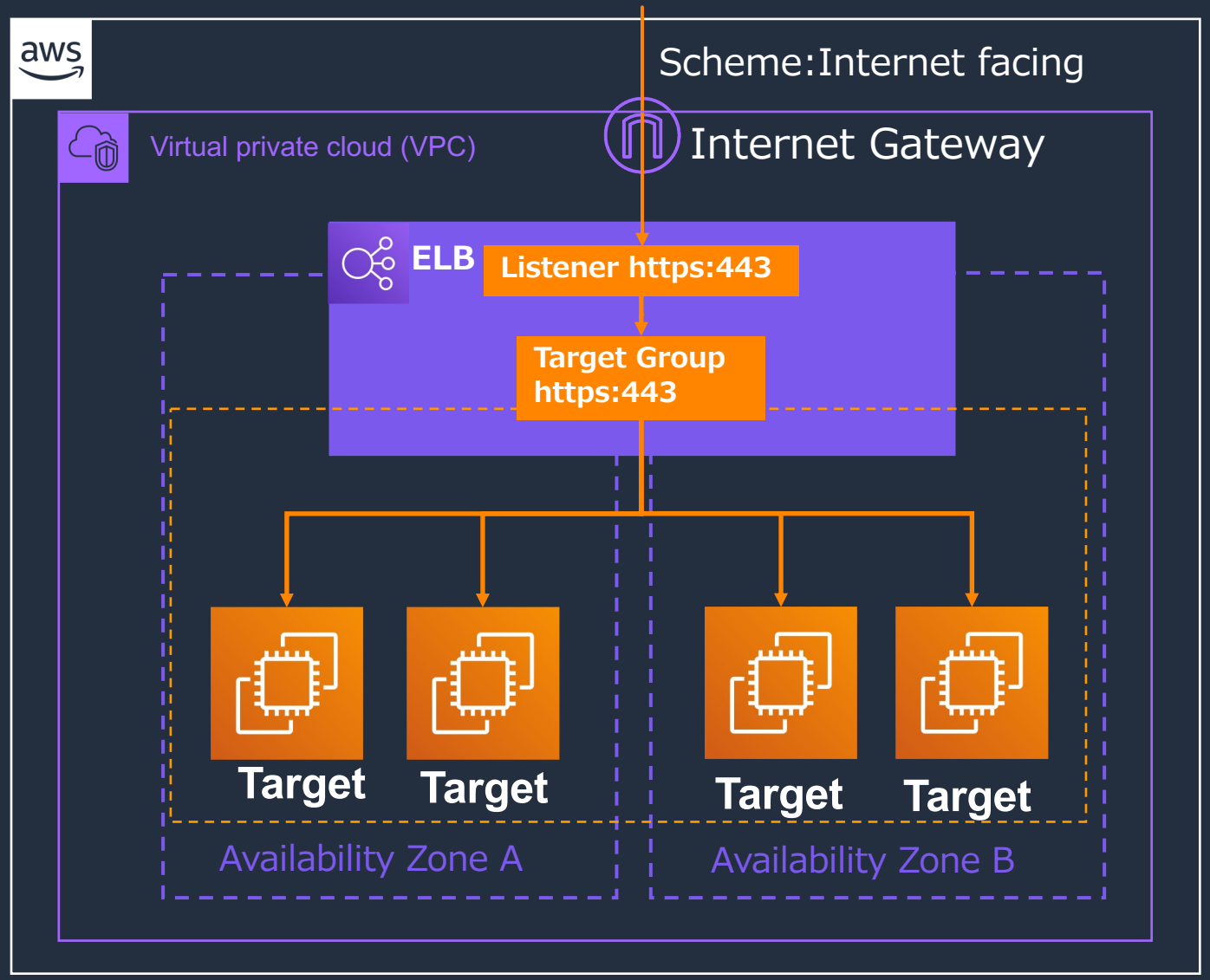


Network Load Balancer
(NLB)

TCP, UDP, TLS

L4+TLSオフロードを提供するロードバランサー

ELBのコンポーネント



・スキーム

パブリックIPを設定してインターネットに公開するか、VPC内部のみに公開するか選択できる。内部のみの場合は、Direct ConnectやTransit Gateway経由でも利用できる

・リスナー

ELBがListenするプロトコルとポート番号を決定する。一つのELBに複数設定可能

・ターゲットグループ

トラフィックをルーティング先ターゲットをグループ化したもので、この単位でELBはトラフィックを転送・分散する。
リスナーに対して設定。条件により複数設定可能

・ターゲット

ELBがトラフィックを転送するEC2インスタンスなどのリソースやエンドポイント

ターゲットタイプ

ロードバランス先の設定で、ターゲットグループ作成時に設定。以下の種類があります

インスタンス

- EC2インスタンスを指定するタイプ。ELBと同じVPCにインスタンスが存在する必要がある。
  EC2のAuto Scalingと併用可能

IPアドレス

- IPアドレスを直接指定するタイプ。EC2インスタンス以外や、VPCの外にターゲットが存在する場合など。IPv6にも対応。Fargateを利用している場合もこちら

Lambda関数

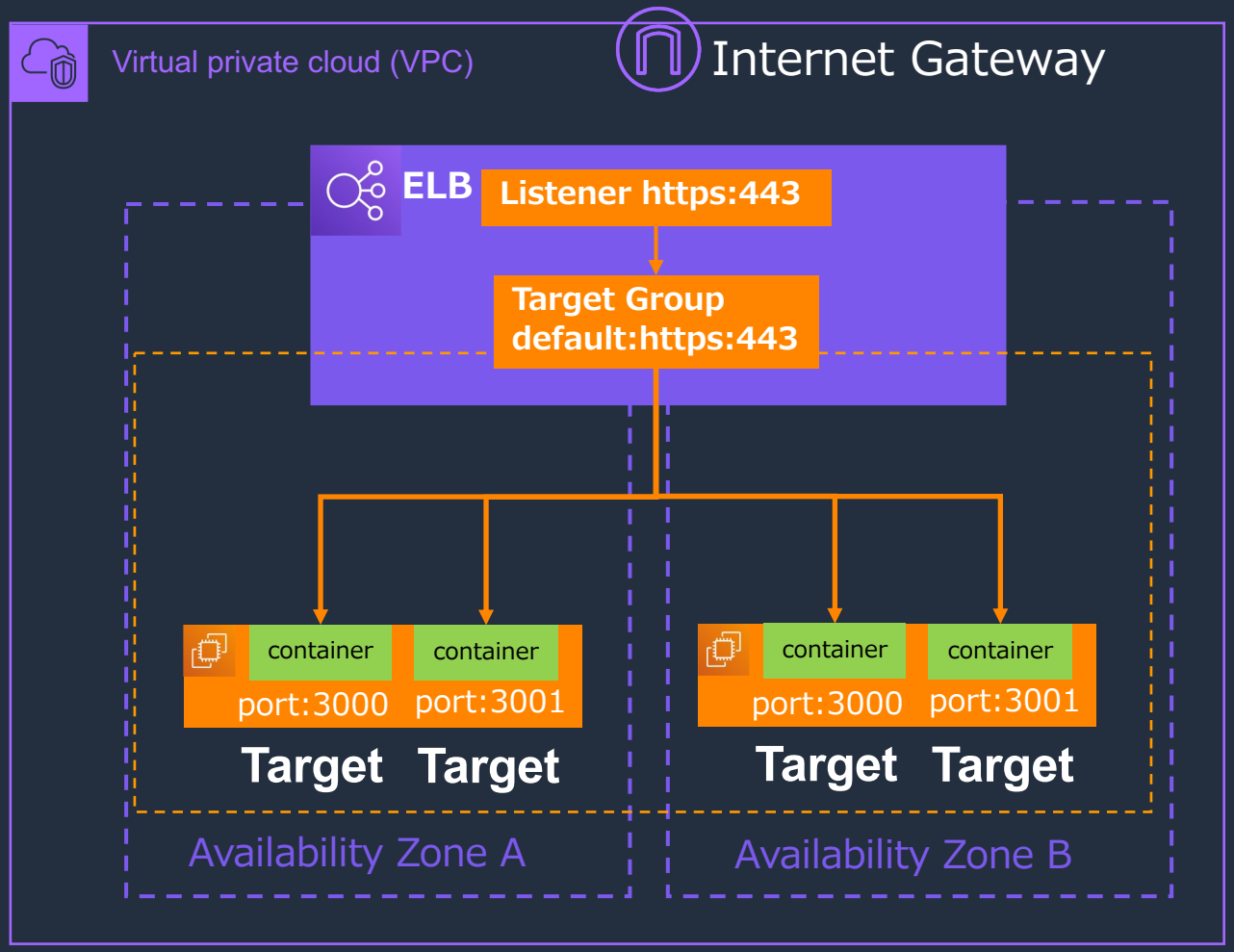
- ALBのみ対応。Lamda関数を呼び出し、Lamda関数から返されたコンテンツをクライアントに返す

Application Load Balancer

- NLBのターゲットにALBを指定したい場合に選択

インスタンスのListen port override

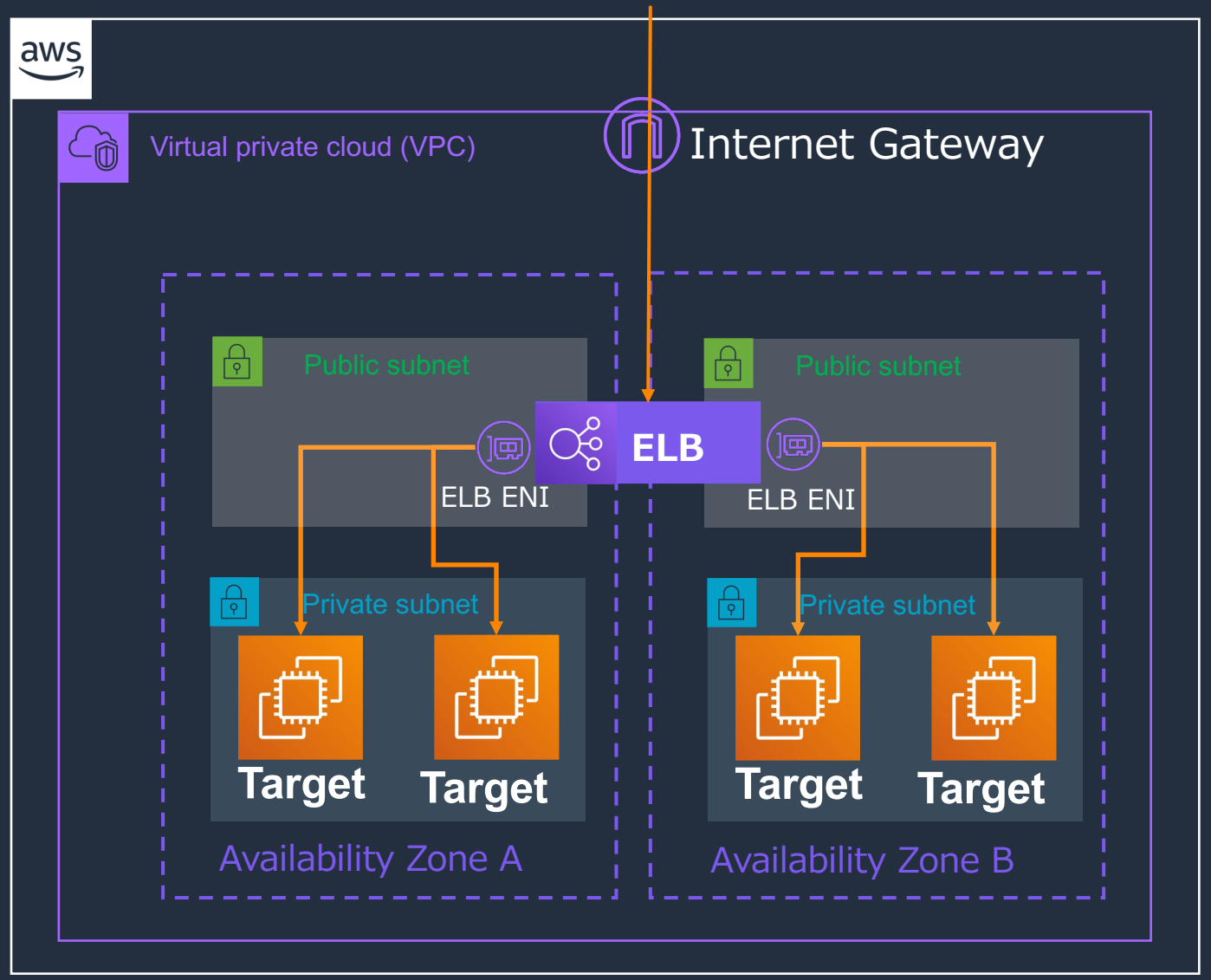
ターゲットグループにターゲットを設定する際、ポートを変更できます。



一つのインスタンスを、IPアドレスは同じでポートを変えて複数のターゲットとして登録可能。コンテナなど、一つのインスタンスの中でロードバランスしたい場合に活用できる。

ターゲットグループ全体に設定するポート番号はデフォルトの意味であり、ターゲットごとに変更可能。

VPCへの設置



VPCへ設置

- AZごとに 1 つのサブネットを指定
- ALBは2つ以上のAZを必ず利用
- ELB用のElastice Network Interface(ENI)が作成される(ALBではスケールに伴って増減)

セキュリティグループ

- ALBはセキュリティグループを指定できる
- ALBはICMP Echo Request/Replyを許可すれば、pingに応答する
- ALBの場合、バックエンドのEC2インスタンスはELBからのみリクエストを受け付ける設定を推奨

クライアントからのアクセス

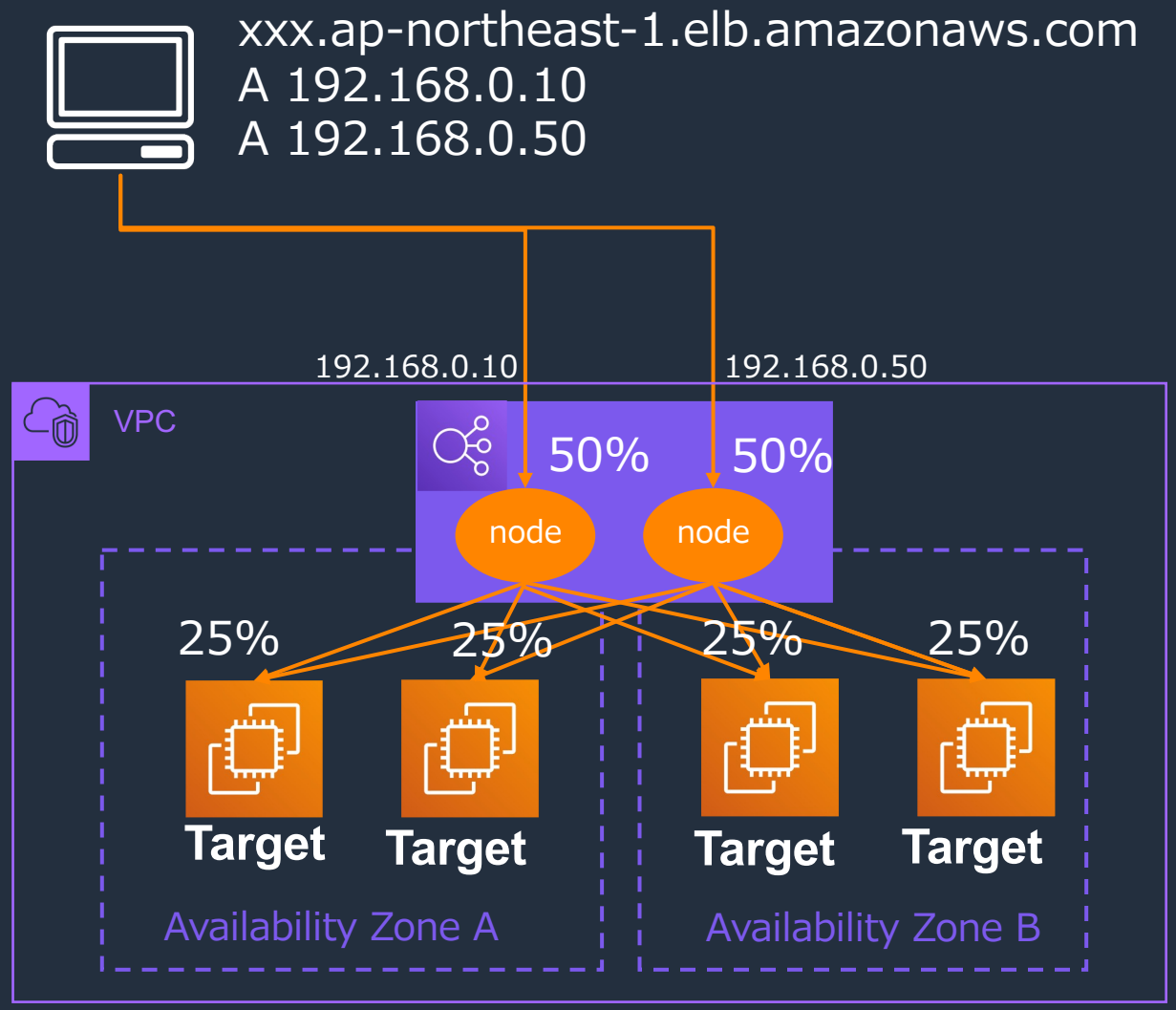
- FQDNが付与されるので、それを利用する
- Route 53との連携で自由なドメインを利用可能
- **IPを直接指定するのは非推奨。特にALBはIPアドレスがスケールアウトに伴って変動するので、現実的ではない**

高可用性と負荷分散

ELB自体のスケーリング

- ELBは負荷に応じて自動でスケールする
- ALBはトラフィックが瞬間的に急増したために、スケーリングが間に合わない場合がある。その場合、応答が遅くなったりタイムアウトしたりする
  - 新サービス開始
  - TVやメディアによるサービス紹介
- 事前にスケールさせておくことで対応できる
  - **Pre-Warming（暖気運転）の申請をサポートケースにて行う**
    ※Business/Enterpriseサポートが必要
  - イベント前に負荷を段階的にかけておき、スケールさせておく
- NLBは暖気不要で瞬間的な数百万リクエスト/秒のトラフィックも捌ける

負荷分散の基本の二つ



DNSラウンドロビン

ELBは複数のノードで構成される。AZに一つ以上ノードがあり、ノードごとにIPアドレスを持つ(NLBはAZにつき一つで固定、ALBは負荷により増減)。ELBのFQDNを名前解決すると、複数のAレコードが返却される。クライアントはこのどれかを利用する。ここでELB自体の負荷が分散される

ルーティング時の分散

バックエンドにリクエストをルーティングする際に、指定のアルゴリズムに従ってターゲットグループ内のサーバにリクエストを分散。複数のAZに分散することにより高い可用性を実現

ルーティングアルゴリズム

ターゲットグループにルーティングする際の分散の考え方

ALBの場合、以下から選択可能

・**ラウンドロビン**

- デフォルトのアルゴリズム。均等にルーティングする

・**最小未処理リクエストルーティング**

- リクエストが来た時点で、処理中のリクエストがもっとも少ないターゲットにルーティングする
- バックエンドのインスタンスの性能にバラツキがある場合などに有効
- 性能の高いインスタンスはそれだけ早くリクエストを処理する。結果的に、性能の高いインスタンスに多くのリクエストを流すことができる

NLBの場合

・**フローハッシュアルゴリズム**

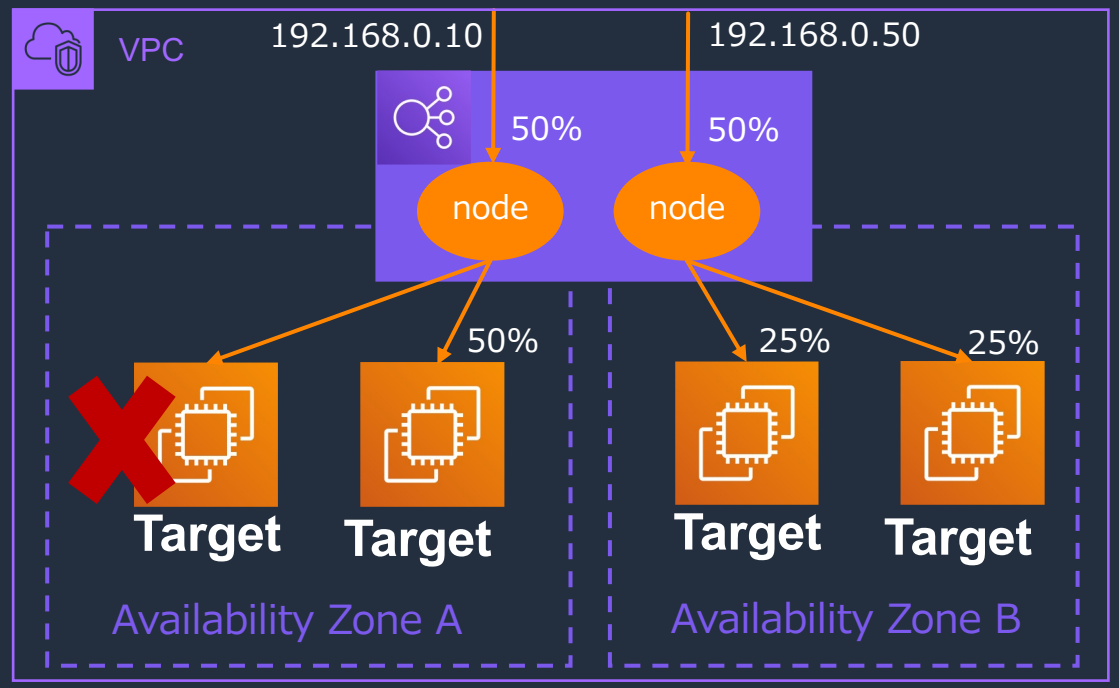
- プロトコル、送信先/送信元アドレス、送信先/送信元ポート、TCP シーケンス番号が一致している通信を一つのフローとし、これらが一致している場合は同じターゲットにルーティングする
- このフロー単位で均等にルーティングする

ルーティングアルゴリズム

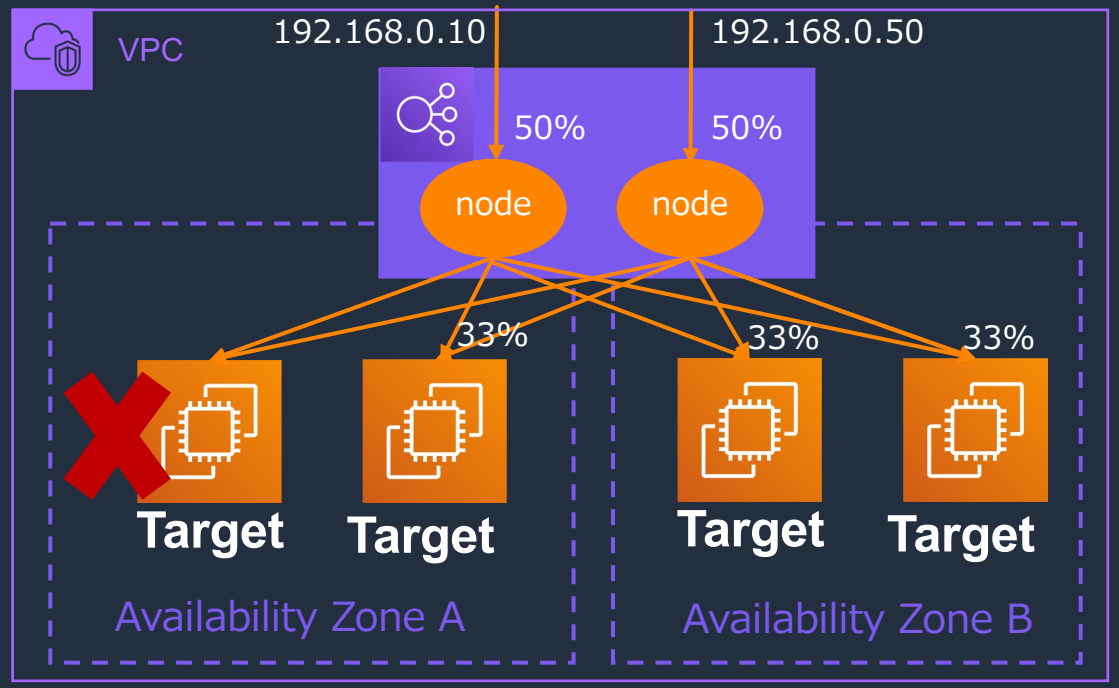
クロスゾーン負荷分散

ELBの名前解決を行うと、複数のAレコードが返る。それぞれのIPアドレスは、ELBのノードを示しており、それらはELBをデプロイしたAZのどれかに属している。クロスゾーン負荷分散を有効にすると、AZ Aに属しているノードが、別のAZにトラフィックをルーティングするようになり、全体の負荷をより均等にすることができる。ALBはデフォルトで有効、NLBはデフォルトで無効で、変更可能。

クロスゾーン負荷分散無効



クロスゾーン負荷分散有効



ターゲットグループの正常性

正常ターゲット数の設定

ターゲットグループ内で、最低いくつの正常なターゲットが存在していれば、そのターゲットグループは正常とみなすかの設定。絶対値(台数）か割合(%)で指定できる。

正常ターゲット数を下回ったときの動きの設定

DNSフェイルオーバー：AZ内でしきい値を下回ると、そのAZのIPアドレスを返さなくなり、リクエストが来なくなる。

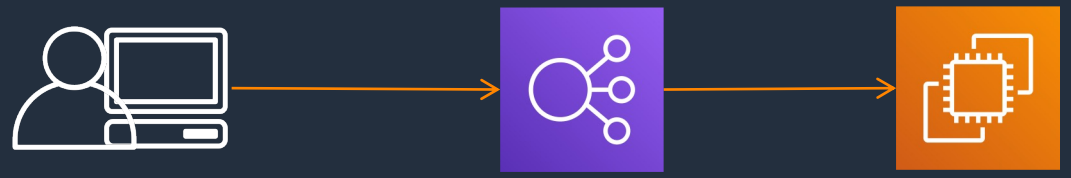
ルーティングフェイルオーバー： AZ内でしきい値を下回ると、異常なターゲットにもリクエストをルーティングするようにする。これにより、一時的にヘルスチェックに不合格になったターゲットが存在したような場合、正常なターゲットの過負荷を防ぎ、少しでもリクエストを処理できるようになる。

https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/application/target-group-health.html

コネクションの設計

コネクションタイムアウト

- 無通信状態が続くとそのコネクションを自動で切断する
- ALBのコネクションタイムアウト値は変更可能
  - 1～4,000秒の間で自由に設定可能
  - NLBは2023年5月時点で350秒固定
- バックエンドのサーバのタイムアウト値は、ELBの値より長くしておくこと



https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/application/application-load-balancers.html#connection-idle-timeout

Connection Draining (登録解除の遅延)

バックエンドのEC2インスタンスをELBから登録解除したり、ヘルスチェックが失敗した時に、新規リクエストの割り振りは中止して、処理中のリクエストは終わるまで一定期間待ってからターゲットを切り離すこと

- デフォルトで有効、タイムアウト 300秒
- タイムアウト最大 3600秒
- Connection Draining 動作中はターゲットのヘルスステータスに「draining」と表示される

https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/classic/config-conn-drain.html

スティッキー セッション (stickiness)

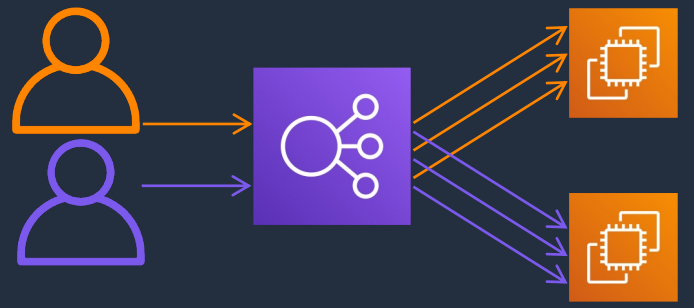
同じクライアントからのリクエストを全て同じEC2インスタンスに送信する仕組み

- ALBではCookieベースで制御する。デフォルトで無効
- NLBでは送信元IPアドレスベースで制御する。こちらもデフォルトで無効。NAT経由だったり、送信元IPが限定されている時はうまく分散されなくなるので注意

ALBのスティッキーセッションの有効期限

- ロードバランサーによって生成されるクッキーを使用したスティキーセッション
  - セッション開始からの有効期間を指定してALBで制御
  - 無期限にする事も可（無期限でもブラウザを閉じれば終了）
- アプリケーション生成のクッキーを使用したスティッキーセッション
  - アプリケーションが作成したCookieにあわせる
  - アプリケーションが作成するCookie名を指定

EC2インスタンスの増減を柔軟にできるように、セッション情報などは別のDBサーバやキャッシュサーバに持たせるのが望ましい。
この場合スティッキー セッションは不要。



ヘルスチェック/モニタリング

ヘルスチェック

- ELBは正常なターゲットにのみトラフィックをルーティングする
- ELBは設定値に基づき、ターゲットに対してヘルスチェックを定期的に行い、正常なターゲットかを判定する
- 正常判定が厳しすぎるとインスタンスが使えるまでに時間がかかり、逆に異常との判定が厳しすぎても、 過負荷時に処理できるインスタンスを減らしてしまうことにもなる

設定	説明	設定例
プロトコル (HealthCheckProtocol)	ターゲットでヘルスチェックを実行するときにロードバランサーが使用するプロトコル (ELBの種類によってHTTP/HTTPS, TCPなど)	HTTP (Status 200が返るのを確認)
ポート (HealthCheckPort)	ターゲットでヘルスチェックを実行するときにロードバランサーが使用するポート デフォルトは、各ターゲットがロードバランサーからトラフィックを受信するポート	トラフィックポート
パス (HealthCheckPath)	ヘルスチェックのターゲットの送信先である ping パス デフォルトは /	/index.html
タイムアウト時間	ヘルスチェックを失敗と見なす、ターゲットからレスポンスがない時間 (秒単位) 範囲は 2～120 秒	5秒
正常のしきい値 (HealthyThresholdCount)	異常なターゲットが正常であると見なされるまでに必要なヘルスチェックの連続成功回数	5
非正常のしきい値 (UnhealthyThresholdCount)	ターゲットが異常であると見なされるまでに必要なヘルスチェックの連続失敗回数	2
間隔	個々のターゲットのヘルスチェックの概算間隔 (秒単位)	30秒
成功コード (Matcher)	ターゲットからの正常なレスポンスを確認するために使用する HTTP コード	200

モニタリング

- CloudWatchによりメトリクスを60 秒間隔で取得可能。ALBかNLBかで項目は異なる
- 各項目で平均(average)、合計(sum)、最大値(max)、最小値(min)等を表示できる。ドキュメントに各メトリクスでどの統計を利用するが推奨なのか以下ガイドあり

<ALBメトリクスの例>

項目	内容
HealthyHostCount	正常なバックエンドのホスト数
UnHealthyHostCount	異常なバックエンドのホスト数
RequestCount	リクエスト数
HTTPCode_ELB_4XX HTTPCode_ELB_5XX	ELBが返した4xx, 5xxのレスポンス数
HTTPCode_Backend_2XX、 HTTPCode_Backend_3XX、 HTTPCode_Backend_4XX、 HTTPCode_Backend_5XX	バックエンドが返した2xx,3xx,4xx,5xxレスポンス数

https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/application/load-balancer-cloudwatch-metrics.html

アクセスログ

- 最短5分間隔でELBのアクセスログを取得可能(NLBはTLSリスナー利用時のみ)
- 指定したS3バケットに簡単にログを自動保管
- ELBの種類によってアクセスログの出力フィールドも異なる

例) ALBのHTTPSリスナーのログ

```
https 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188 192.168.131.39:2817
10.0.0.1:80 0.086 0.048 0.037 200 200 0 57 "GET https://www.example.com:443/ HTTP/1.1" "curl/7.46.0"
ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 arn:aws:elasticloadbalancing:us-east-
2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 "Root=1-58337281-
1d84f3d73c47ec4e58577259" "www.example.com" "arn:aws:acm:us-east-
2:123456789012:certificate/12345678-1234-1234-1234-123456789012" 1 2018-07-02T22:22:48.364000Z
"authenticate,forward" "-" "-"
```

[リクエストタイプ] [timestamp] [elbのリソースID] [クライアントのIPとポート番号] [ターゲットのIPとポート番号] [ELBがリクエストを受け取った時点からターゲットに送信するまでの合計経過時間] [ELBがターゲットにリクエストを送信した時点から、そのターゲットが応答ヘッダーの送信を開始した時点までの合計経過時間] [ELBがターゲットから応答ヘッダーを受け取った時点から、クライアントへの応答の送信を開始した時点までの合計経過時間] .....

https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/application/load-balancer-access-logs.html
https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/network/load-balancer-access-logs.html
http://docs.aws.amazon.com/ja_jp/ElasticLoadBalancing/latest/DeveloperGuide/access-log-collection.html

セキュリティ

SSL / TLS Termination

ELB側でSSL / TLS 終端ができる (基本的にはTLSプロトコルを使用)
以下の通信パターンが考えられる

1.ELBでSSL終端し、ターゲットとの通信はSSLなし
- バックエンドのEC2インスタンスでSSL処理せずに済むため負荷をオフロードできる

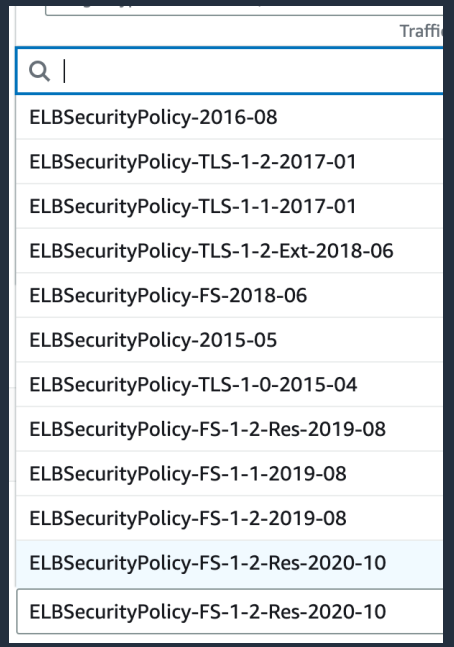
2.ELBでSSL終端し、ターゲットとは別途SSL

3.ELBではSSL終端せずターゲットにTCPリスナーで送信、バックエンドでSSL
- NLBのみ
- クライアント証明書認証などを利用する場合はこちら

http://docs.aws.amazon.com/ja_jp/ElasticLoadBalancing/latest/DeveloperGuide/elb-listener-config.html#using-elb-listenerconfig-quickref

セキュリティポリシー

- SSL/TLS利用時には事前定義されたセキュリティポリシーを利用する
- 事前定義されたセキュリティポリシーはSSL/TLSプロトコルの設定+暗号スイート(Cypher)の設定が定義されている
- TLS v1.0, v1.1, v1.2をサポート。2023年2月現在では、NLBでTLSオフロードした場合のみTLS1.3をサポート
- Perfect Forward Secrecy (PFS) のサポート
- Server Order Preference のサポート
- ALPNポリシーのサポート（NLBのみ）
- 新しく作ったELBではELBSecurityPolicy-2016-08 がデフォルトだが準拠したいガイドライン等によって選択を

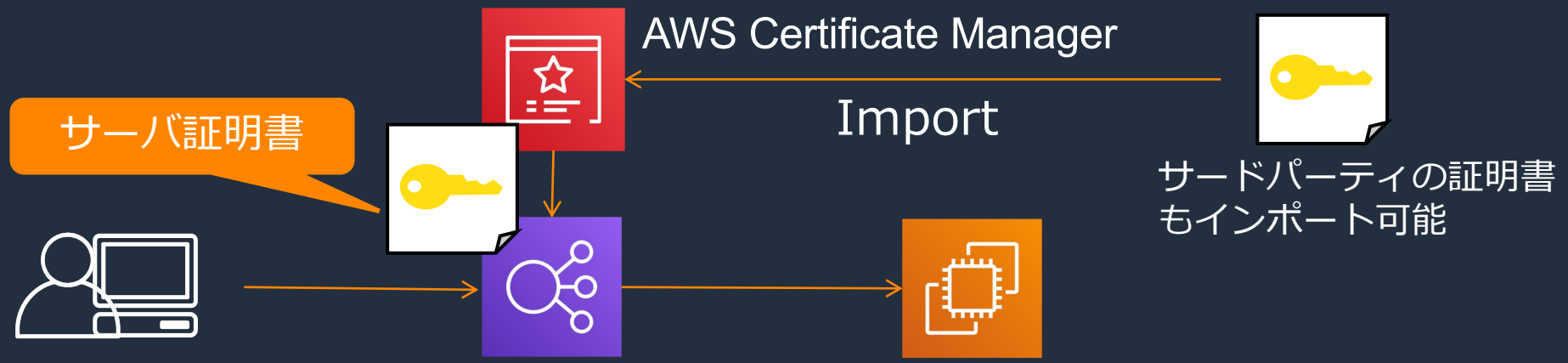


https://docs.aws.amazon.com/ja_jp/ElasticLoadBalancing/latest/DeveloperGuide/ssl-config-update.html

HTTPS/SSL利用時のTLSサーバ証明書

AWS Certificate Manager(ACM)を使用すれば証明書のリクエスト、管理、更新、プロビジョニングが容易に実行可能

- 無料で証明書を利用可能（ACMと統合されているELB, Amazon CloudFront , Amazon API Gatewayに対してのみ）
- ELB に対する証明書の設定を数クリックで完了
- 証明書は自動更新されるので、失効の心配がない
- ACMで発行される証明書はドメイン認証タイプ(DV)の証明書なのでより上位の証明書(OV, EV)を利用する場合はサードパーティの証明書を取得してインポートして利用



http://docs.aws.amazon.com/ja_jp/ElasticLoadBalancing/latest/DeveloperGuide/ssl-server-cert.html

複数TLS証明書を設定（SNIによるスマートセレクション）

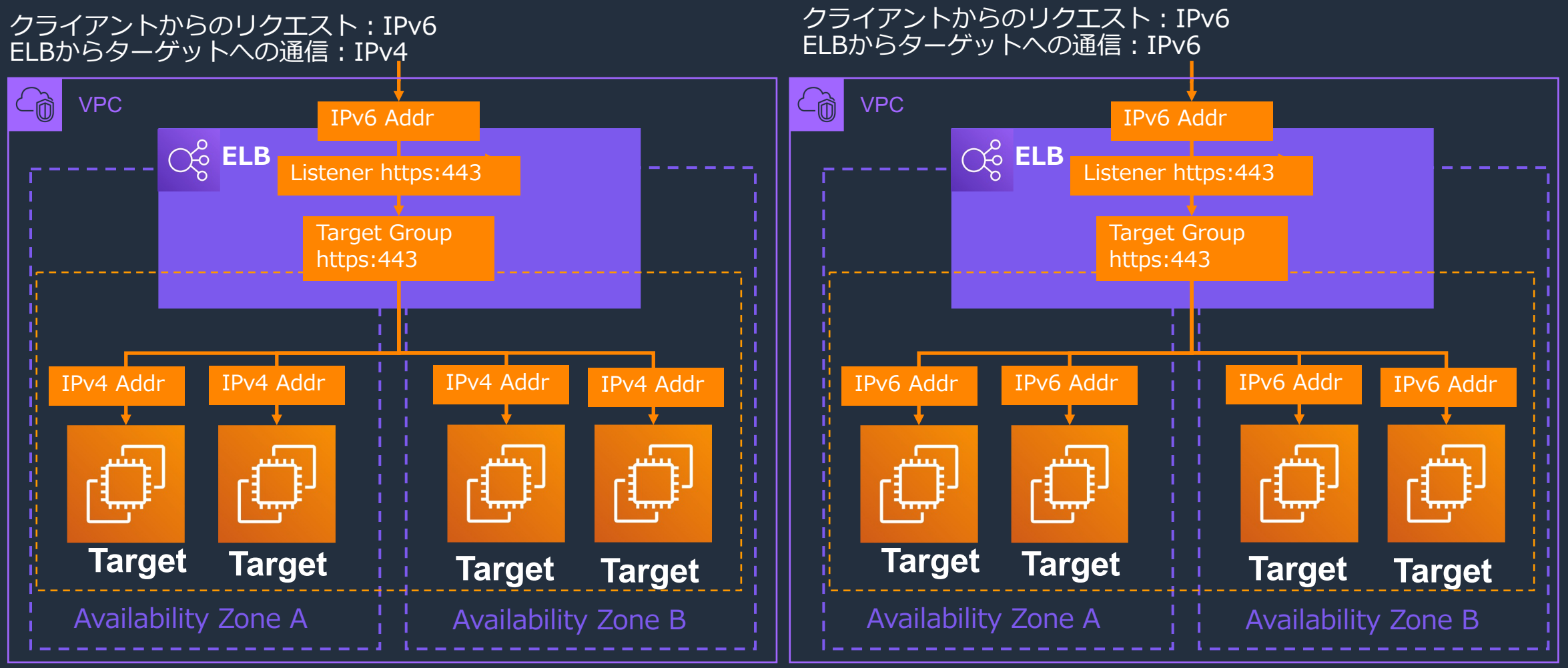
- 複数のTLS証明書を1つのALB/NLBのリスナーに設定可能
  - 複数のドメインを使う場合などに有効
  - SNIをサポートするクライアントには、適切な証明書を選択してTLSで通信をできる
    - SNI非サポートのクライアントにはデフォルト証明書が使われる
  - ドメインはもちろんサポートする鍵交換方式や暗号、署名アルゴリズムを元に証明書を選択するスマートセレクション
- 最大25証明書まで (デフォルト証明書を除く)
  - ACMまたはIAMの全ての証明書が利用可能

https://aws.amazon.com/jp/blogs/news/new-application-load-balancer-sni/
https://aws.amazon.com/about-aws/whats-new/2017/10/elastic-load-balancing-application-load-balancers-now-support-multiple-ssl-certificates-and-smart-certificate-selection-using-server-name-indication-sni/

IPv6

IPv6対応

構成パターンを選択できる。ただし、IPv6 Single Stackは不可。UDPはIPv4のみ(NLB)。
インターネット向け、内部向け双方に対応。



ALB固有機能

ALBの特徴と固有の機能

- L7ロードバランサー
- HTTP/HTTPS(+Websocket,gRPC) に対応
- リスナールールによるコンテンツベースのルーティング (高度なリクエストルーティング)
- ユーザー認証機能
- トラフィックのスパイク時には暖気申請が必要
- その他
  - ネイティブ HTTP/2 対応
  - ターゲットとしての Lambda 関数
  - x-forwarded-forヘッダでクライアントのIPアドレスを記録
  - AWS WAFとの連携

高度なリクエストルーティング（リスナールール）

リスナールール

ALBの場合、図のようなエディタを使って
リスナールールを設定可能。
「パスが/testだったら固定レスポンスを返す」
「送信元IPがxxx.xxx.xxx.xxxだったらリダイレクトする」
「それ以外だったら（デフォルト）ターゲットグループに転送する」といった柔軟な制御が可能。

条件に使えるの要素の例

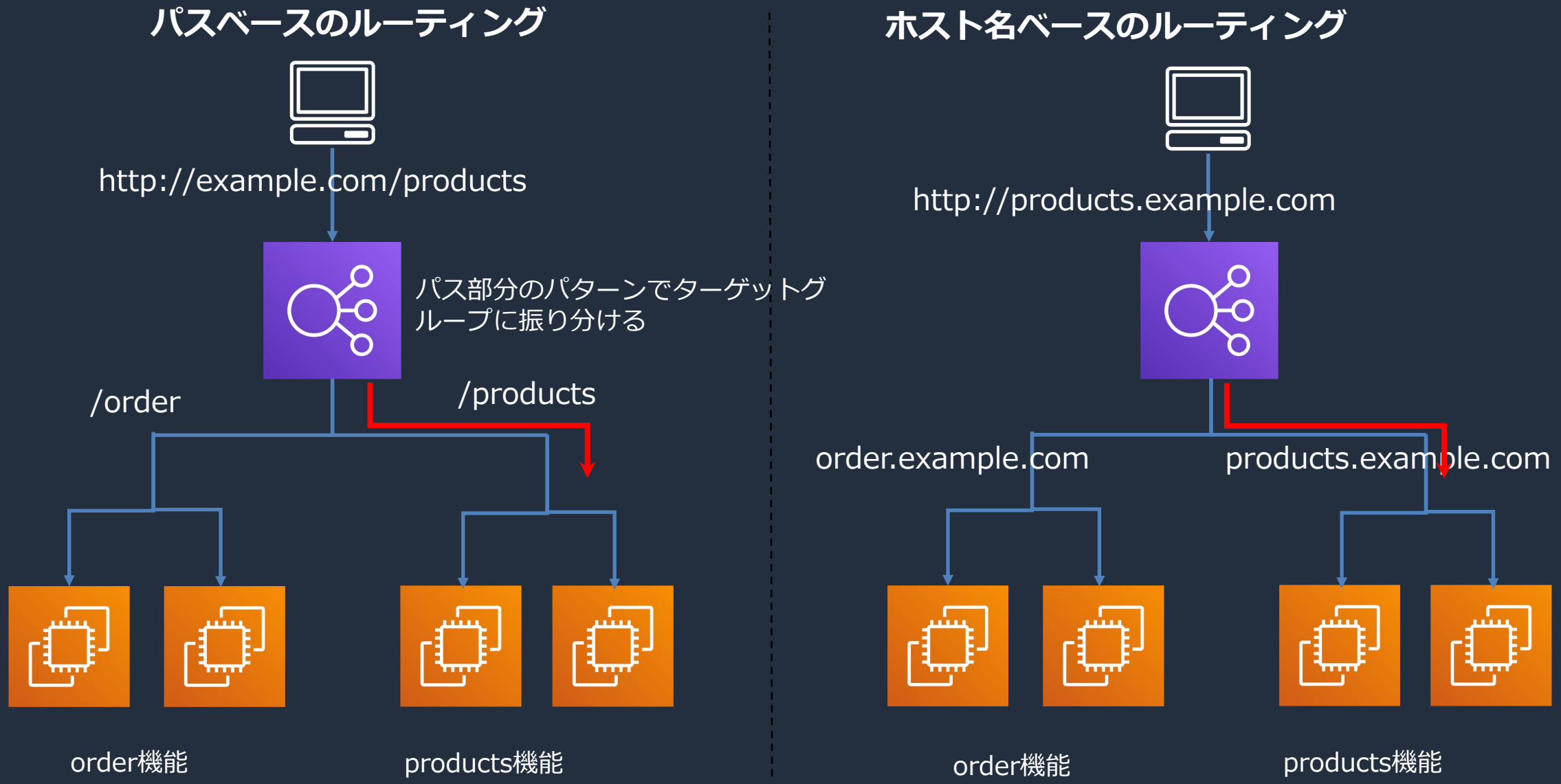
- ホストヘッダ
- HTTPヘッダ
- パス
- HTTPリクエストメソッド
- 文字列のクエリ
- 送信元IP

条件に対応するアクションで選択できるものの例

- ターゲットグループに転送
- リダイレクト
- 固定レスポンスを返す
- 認証する

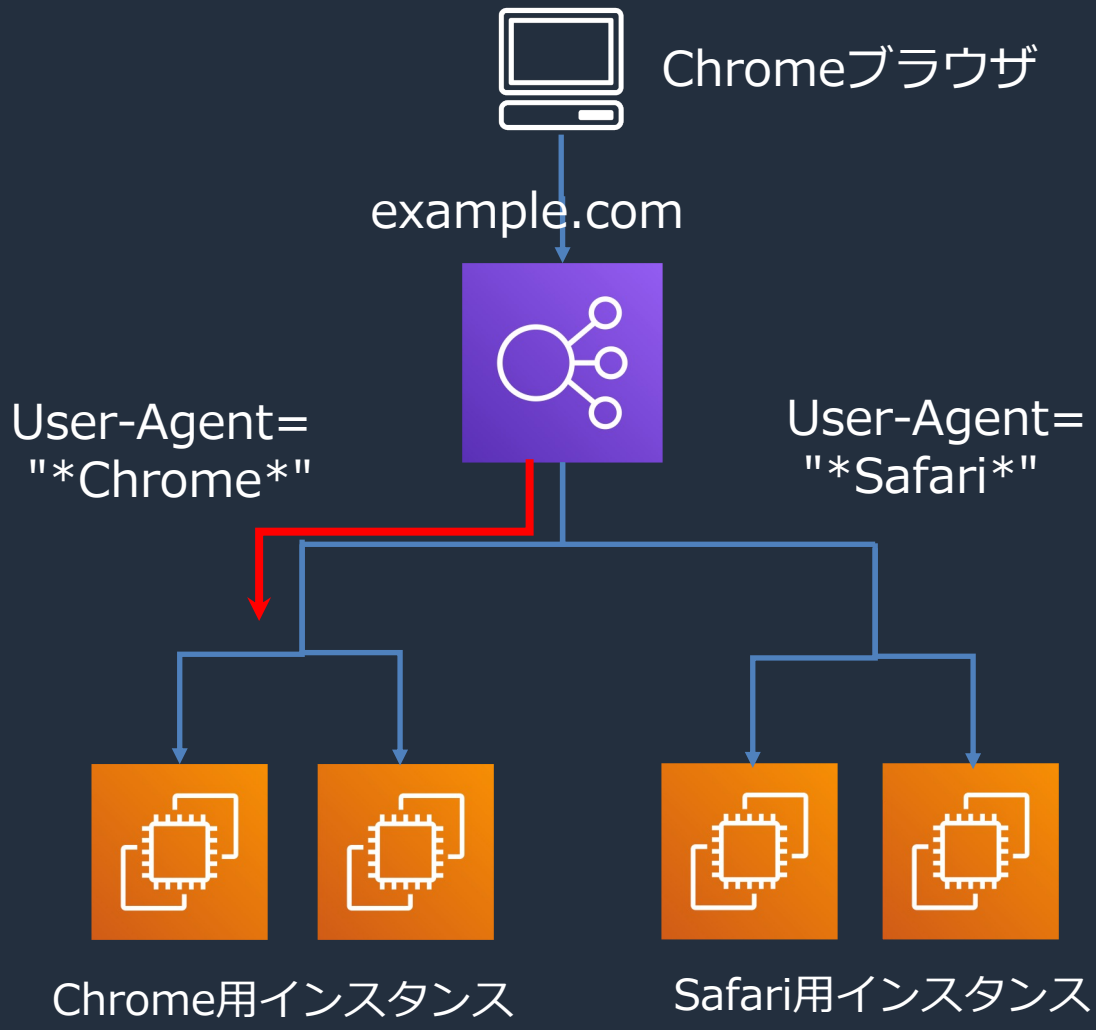
https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/application/load-balancer-listeners.html#rule-condition-types

コンテンツベースのルーティング

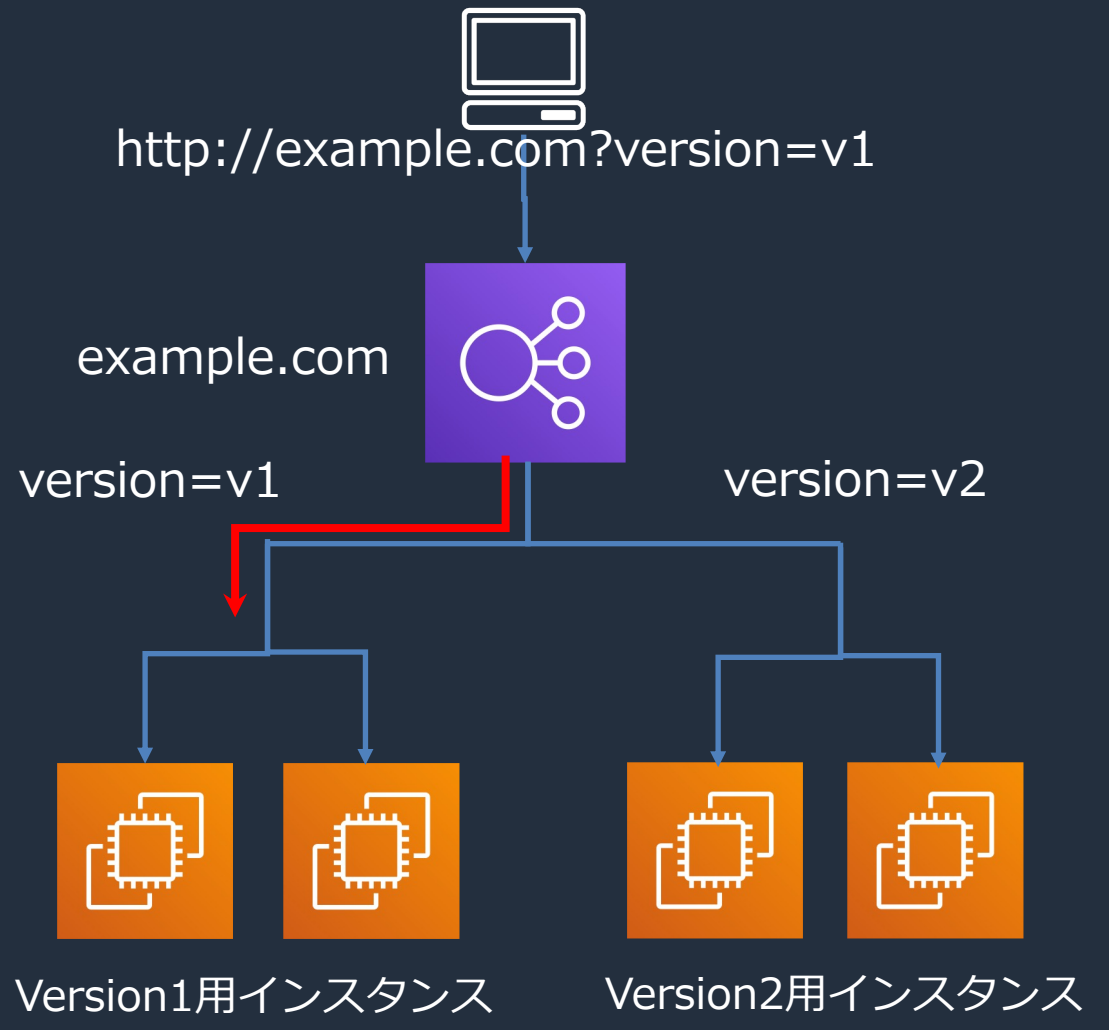


コンテンツベースのルーティング

HTTPヘッダーベースのルーティング

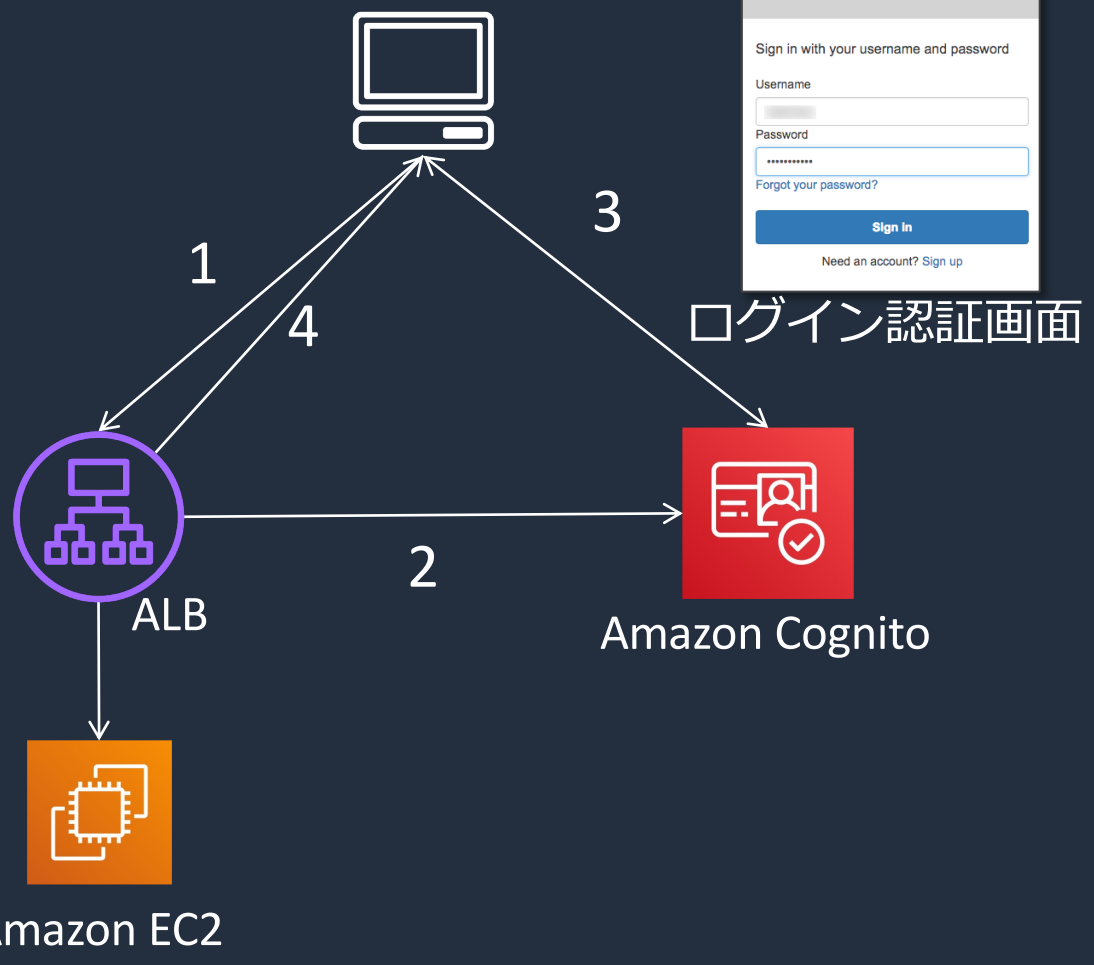


クエリ文字列ベースのルーティング

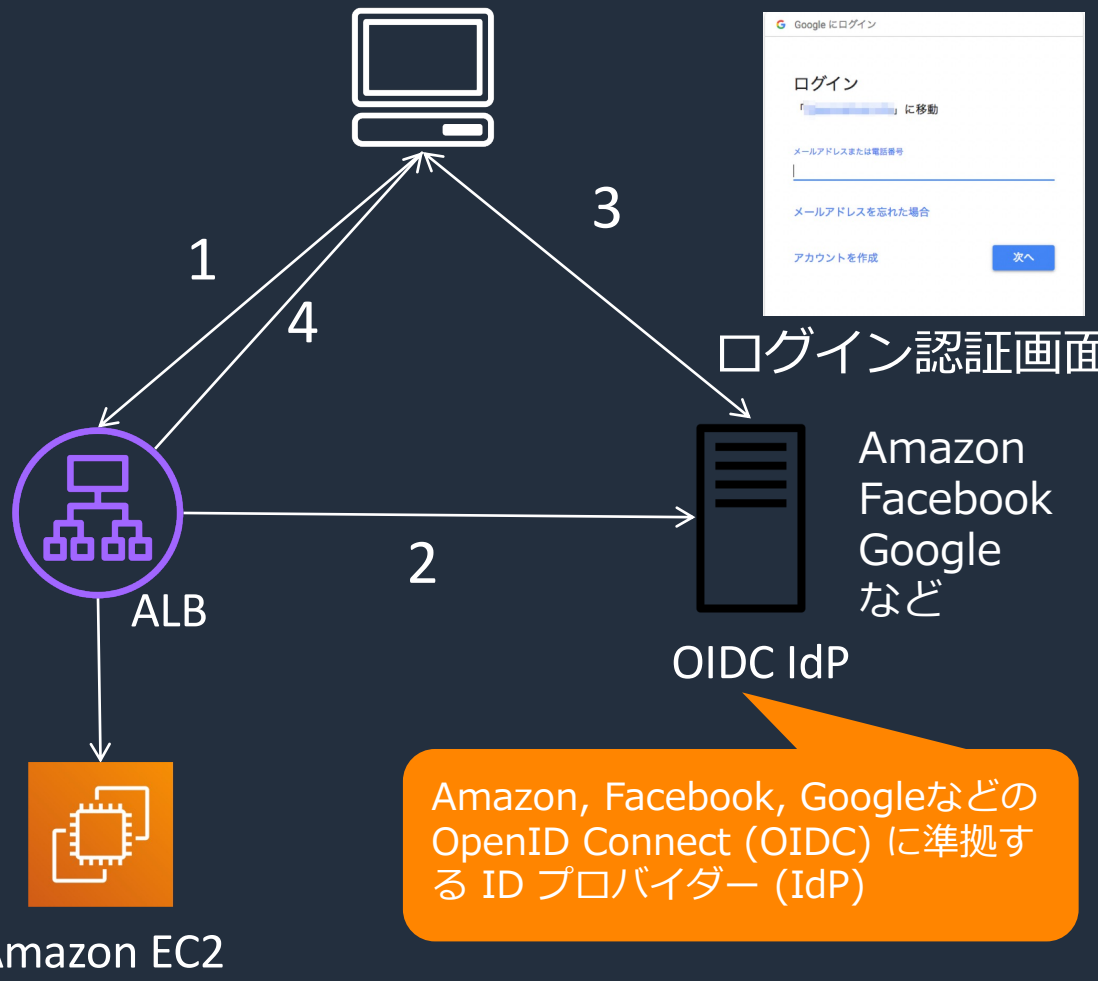


ALBのユーザー認証機能

Cognitoによる認証



OIDC IdPによる認証



最初にALBにアクセスするとCognitoやIdPの認証画面にリダイレクトされる

クライアントのIPアドレスの取得

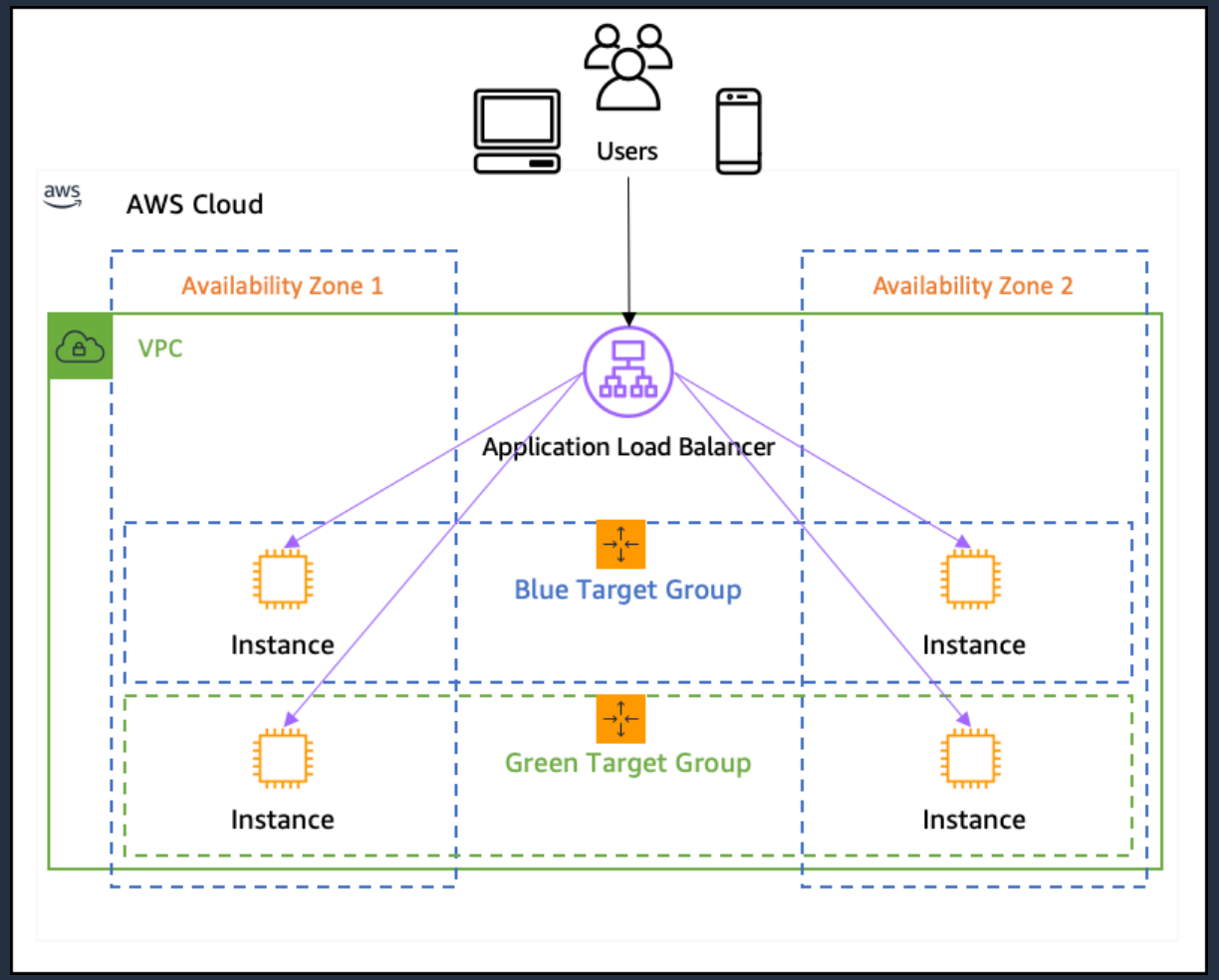
- HTTP/HTTPS リスナーを使用するALBの場合、クライアントの IP アドレスをキャプチャするには X-Forwarded-For ヘッダーを使用する必要あり
  - ターゲット側のアクセスログにはALBのIPアドレスが記録される

- NLBの場合は、透過的にルーティングするため、送信元IPアドレスがクライアントのIP アドレスとなる。ただし、送信元がNLBのIPアドレスになるケースがある。その場合はProxy Protocolを利用できる（後述）

https://aws.amazon.com/jp/premiumsupport/knowledge-center/elb-capture-client-ip-addresses/

加重ターゲットグループ

一つのリスナーに複数のターゲットグループを設定し、割合を設定できる



Blue-Greenデプロイをしたい場合に活用できる

たとえば、新しいバージョンのアプリケーションをデプロイする際、規模の大きい本番トラフィックを一度に切り替えるのは不安がある。そのようなとき、少しずつ新しい環境にトラフィックを移すことができる。

最初は10%だけGreenのターゲットグループにルーティングされるように設定し、問題なければ20%..30%...と少しずつ割合を変えていく、といった運用が可能。

そのほかの機能

- ネイティブ HTTP/2 対応
  HTTP 1.1プロトコルから多数の改善が行われ、1つのHTTP/2コネクションで最大 128 のリクエストを並行して送信可能
- ターゲットとして Lambda 関数を指定可能
- Websocketに対応
- gRPCに対応
- Outpostに対応
- Desync軽減モードに対応
  HTTP Desync Attacksのような攻撃を対策。HTTP Desync GurdianというAWSのOSSライブラリを利用しており、脅威レベルを判定とブロックが可能
- AWS WAFに対応

NLB固有機能

NLBの特徴と機能

- L4(TCP/UDP)のロードバランサー
- 高スループット、低レイテンシ
- 固定IPアドレス: AZ毎に1つ設定。インターネット向けの場合EIastic IPを設定
- 送信元IPアドレス/Portの保持
- 暖機なしに急激なスパイクにも対応可能
- AWS PrivateLink で他のVPCにエンドポイントを作成可能
- ALBをターゲットに指定できる
- セキュリティーグループなし

https://aws.amazon.com/jp/blogs/news/new-network-load-balancer-effortless-scaling-to-millions-of-requests-per-second/
https://aws.amazon.com/about-aws/whats-new/2017/09/announcing-network-load-balancer-for-elastic-load-balancing/

Source IP/Portの保持

- クライアントのSource IPとPortを、そのままTargetまで届けることができる
  - Targetはクライアントと直接通信しているかの様に見える
  - 実際は、行きも戻りもNLBを通っている
  - IP Targetの場合は、リスナーでUDPまたはTCP_UDPの場合のみ透過
  - IP TargetでリスナーがTCPまたはTLSの場合や、TargetがNLBがあるVPCの外にある場合は保持されず、NLBのIPとなる
  - PrivateLink経由の場合も保持されず、NLBのIPとなる
  - 保持されない場合などはProxy Protocolを活用できる
- TargetのSecurity GroupでクライアントIPの接続を許可する必要あり
  - インターネット向けに広く公開する場合は0.0.0.0/0で公開が必要
  - ある程度制限をする場合は加えて、Health checkのためにVPC CIDRかNLB ENIからのアクセスも許可する必要あり
  - VPC内からのアクセスの場合でもターゲットへのアクセス許可はセキュリティグループ IDの指定ではなくクライアントIPの指定が必要

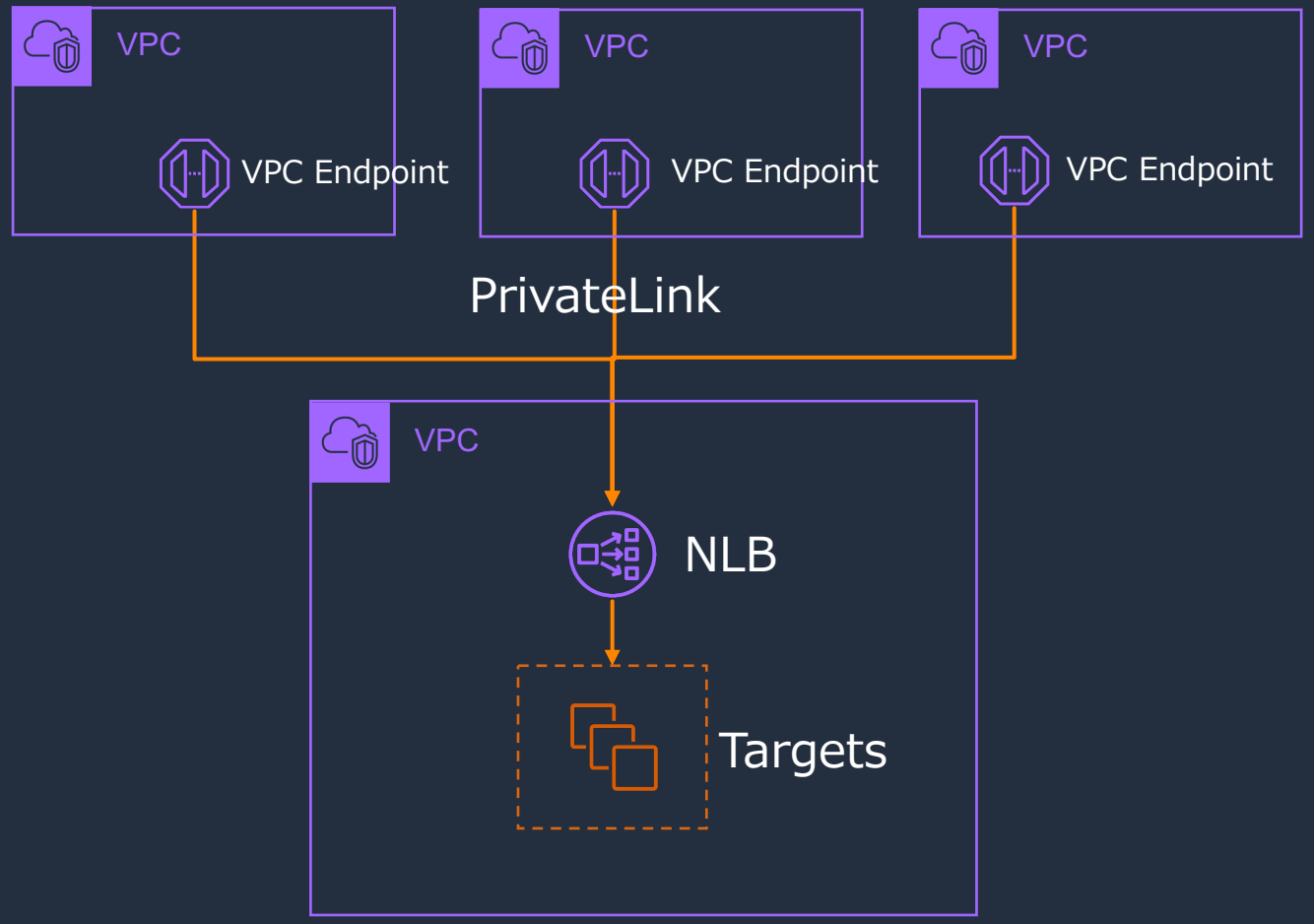
固定IPアドレス

- Internet-facing、Internal共にIPアドレスが固定
  - AZ毎に1つのIPアドレスを利用、DNSはAレコードでも設定可能
    - IPアドレス直指定でアクセスすると、そのAZが障害になった際に切り替わらないので注意
  - ALB, CLBではIPアドレスは不定（DNSで同定可能）

- NLB作成時に自動割当されたIPアドレス、又はNLB作成時に指定した自分が持っているElastic IPのいずれか
  - 自動割り当てされたIPアドレス以外の自前のElastic IPを使う際にはNLB作成前にあらかじめElastic IPを用意しておく必要あり
  - NLB作成後に変更は不可能

- よくあるユースケース
  - Firewallの制約等で、ELBのIPアドレスの固定が必要な時

PrivateLinkで別のVPCにアプリケーションを提供

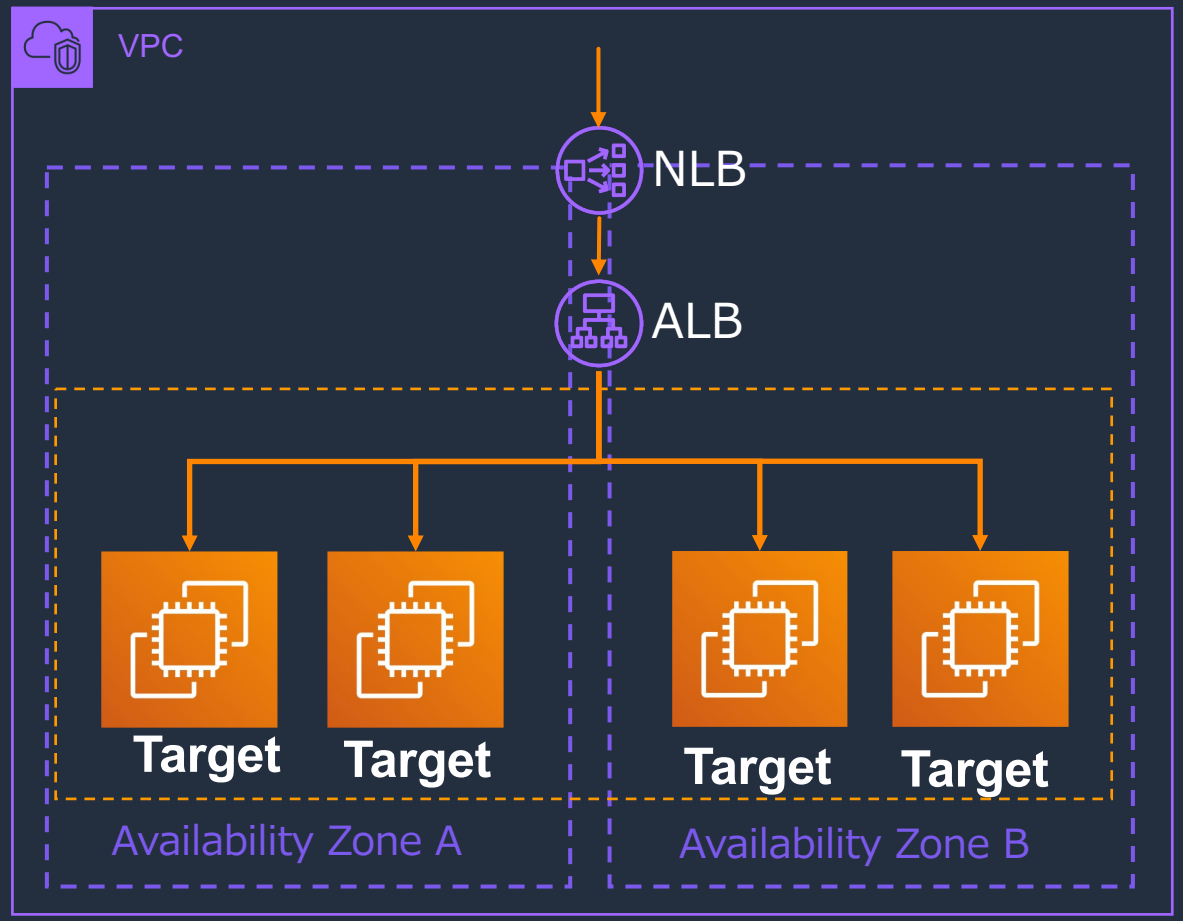


PrivateLink

- NLBのエンドポイントを別のVPCに作ることができる
- 複数のVPCに作ることができる
- **VPC同士のCIDRが重複していても問題なく利用できる**
- 各VPCからのEC2インスタンスなどから、VPCエンドポイントに対して通信を行うと、NLBの配下のサーバに通信が転送される
- Proxy Protocolを利用するとサーバ側でPrivatelinkのIDを取得できる

ターゲットタイプALB

NLBは、ターゲットタイプにALBを指定可能



用途

・ALBに対して固定IPアドレスでアクセスしたい

・ALB配下のアプリケーションサービスをPrivateLinkで別VPCに提供したい

・HTTP/HTTPSと他のプロトコルを併用するようなアプリケーションを構成したい

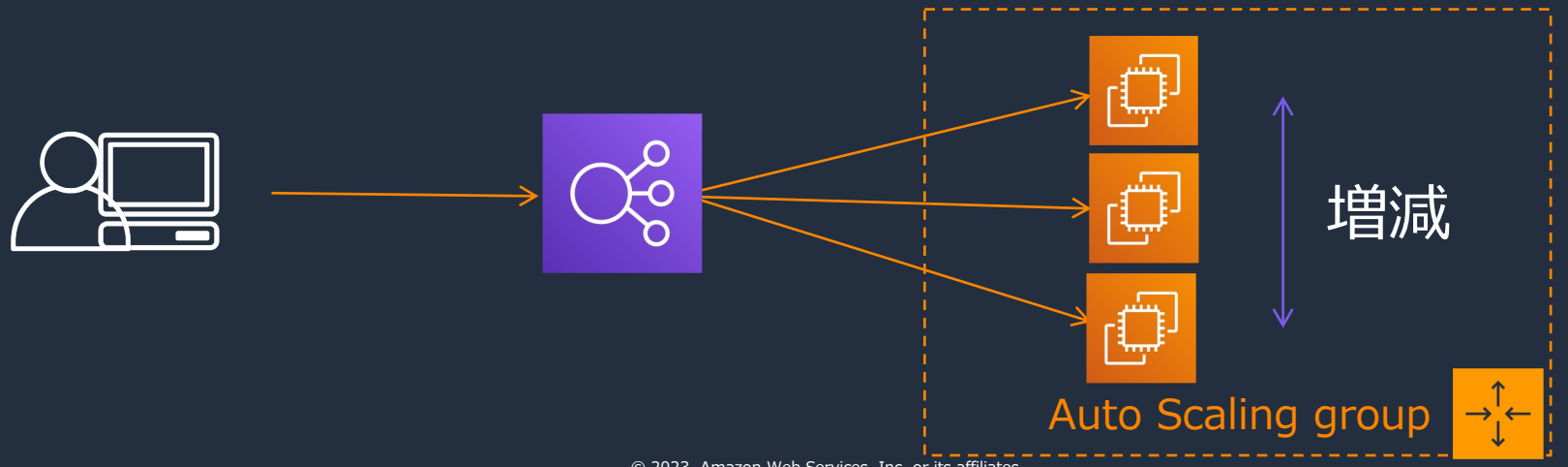
注意すること

- Instance Targetには一部古い世代が利用不可
  - C1, CC1, CC2, CG1, CG2, CR1, CS1, G1, G2, HI1, HS1, M1, M2, M3, T1
  - まとめるとI2,C3を除く2013年以前のインスタンスタイプ
- Idle Connection Timeoutは350秒固定
  - アイドルタイムアウト期間の経過後にクライアントまたはターゲットがデータを送信した場合、TCP RST パケットが返されて接続が無効になったことを示す
- TLSリスナーではアクセスログが取得可能だが、TCPの場合はVPC Flow Logで代替
- NLB自体にセキュリティーグループの設定はない

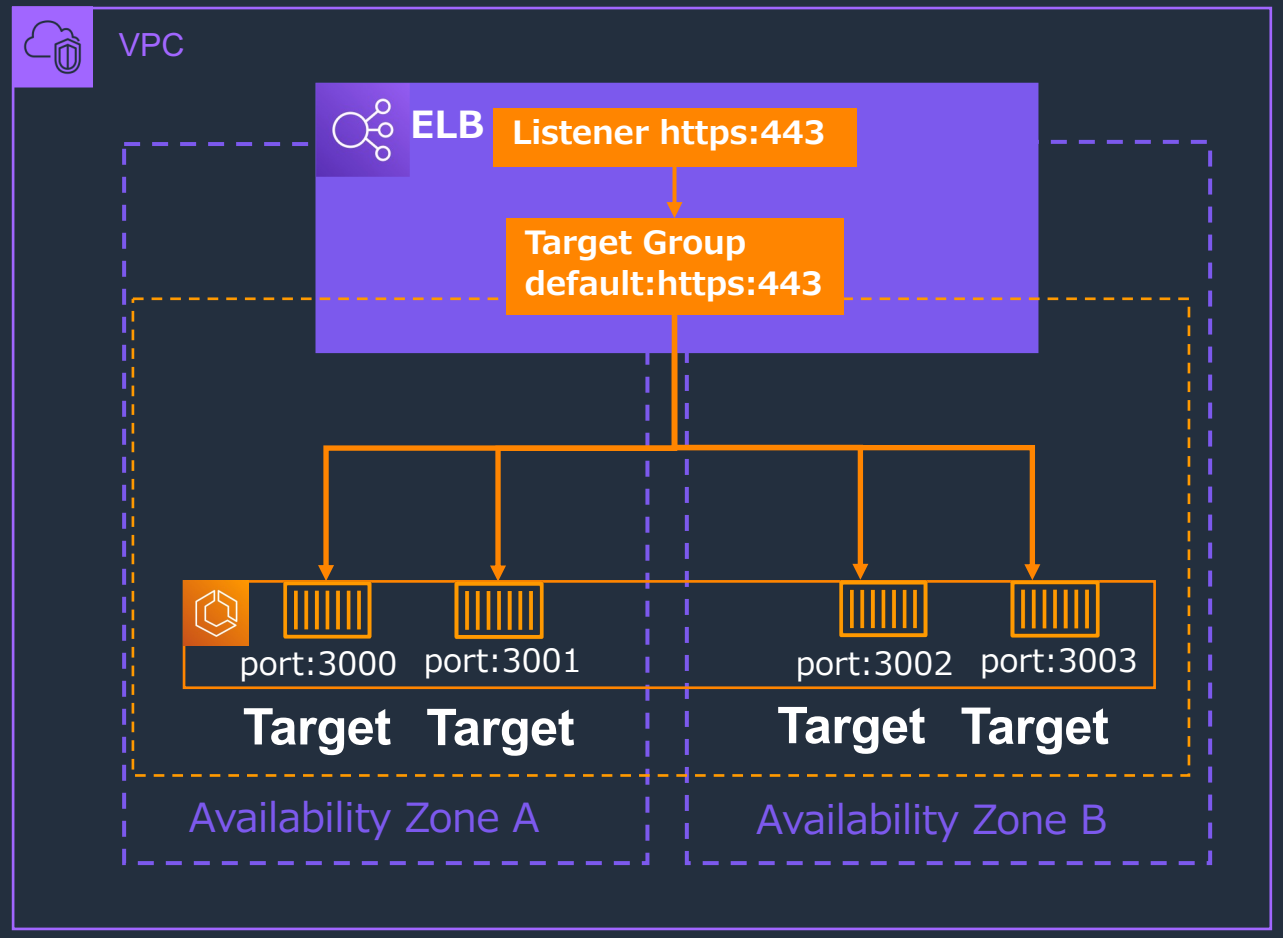
他サービスとの連携

Auto Scalingとの連携

- Auto Scalingによるインスタンス増減時にELBへの追加・削除が可能
- ELBのヘルスチェックの結果をAuto Scalingに反映可能
- インスタンス削減時は、Connection Drainingによる処理中の接続を待つ
- 利用例
  - 一定間隔でレスポンスをチェックし、遅延が増加したらインスタンスを自動追加
  - ELBのヘルスチェックが成功したEC2インスタンスを常にX台以上



ECSとの連携例



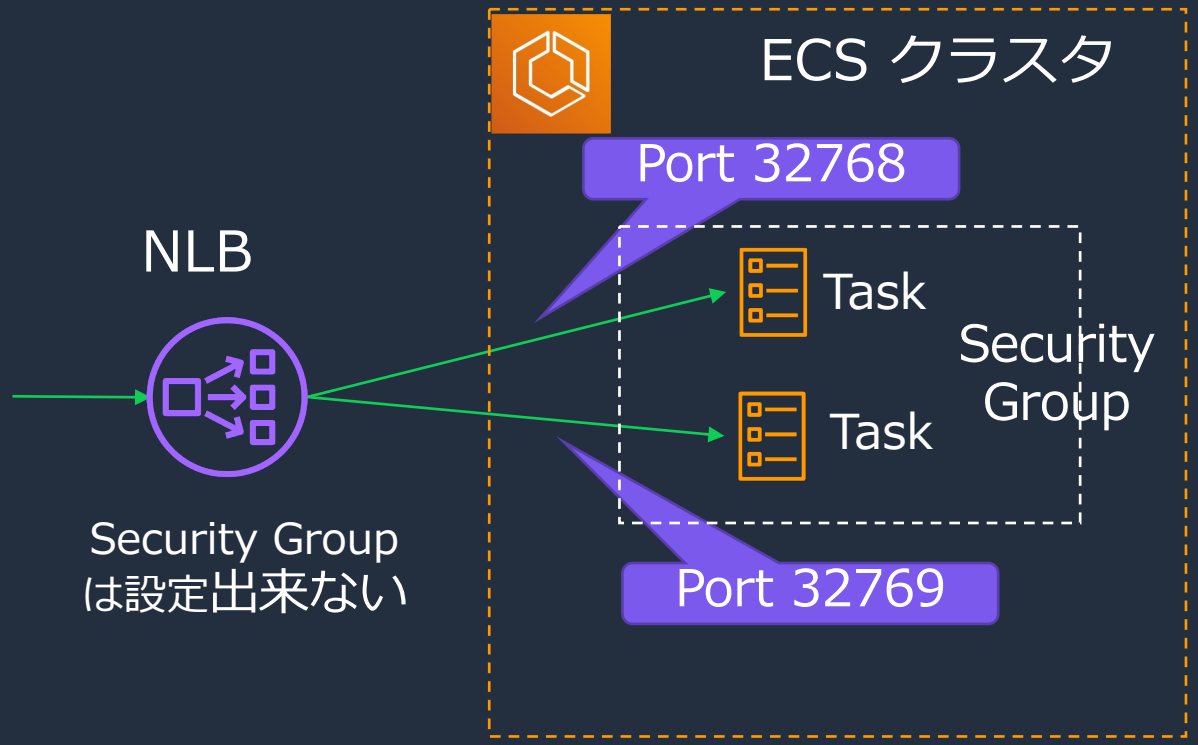
動的ポートマッピング

ネットワークモードがbridgeの場合、動的ポートマッピングにより各タスクに割り振られたECSコンテナのポートをターゲットグループに自動的に登録

ECSのコンソール画面でALBとの連携を設定可能。

Security Group と動的ポートマッピング

NLB には Security Group が設定できないため、ECS コンテナインスタンス側で Security Group の設定を行う。ECS タスクに動的に設定されるポートの範囲を意識する必要がある。

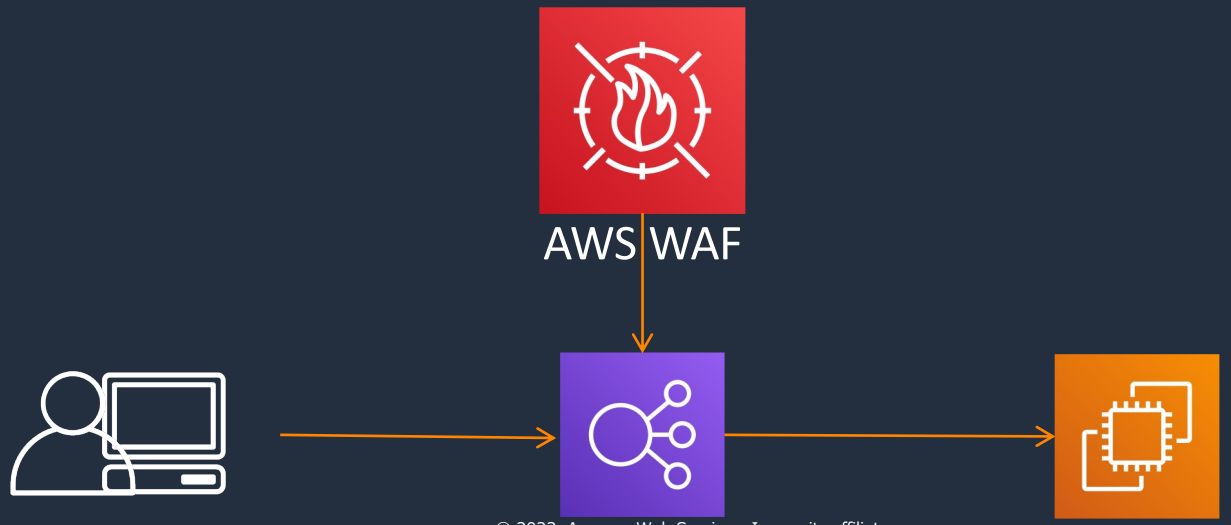


- ECS コンテナインスタンス側で Security Group を設定
- ECS の動的ホストポートマッピングの範囲は /proc/sys/net/ipv4/ip_local_port_range で設定

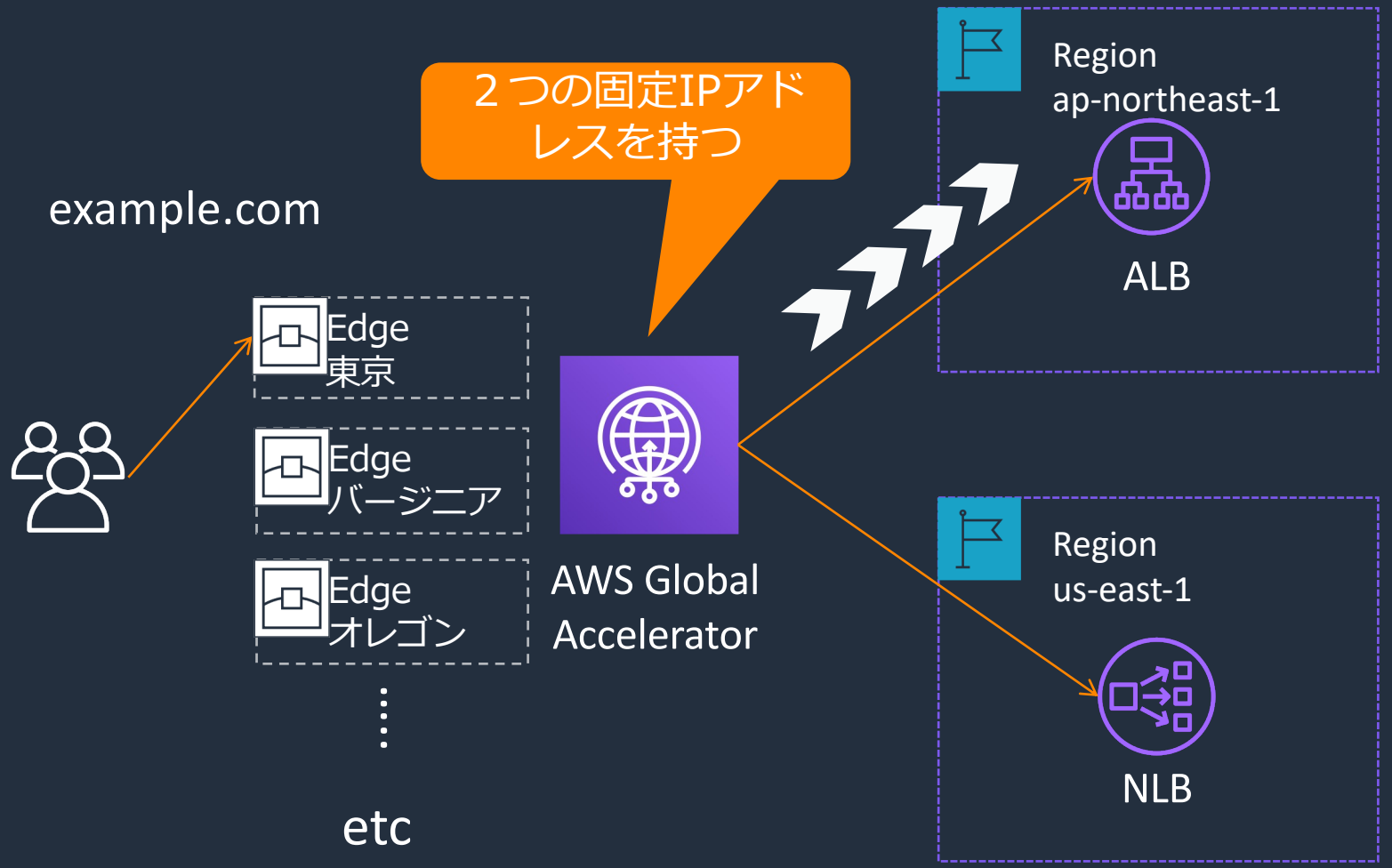
ALBとAWS WAFの連携

- ALBに対してAWS WAFを適用可能

- AWS WAFを用いると以下の条件を利用してALBを保護可能
  - リクエストレートによるアクセス権限(Rate Limit)
  - 特定のIPアドレスや地域からのアクセスを制限
  - クロスサイトスクリプティングやSQLインジェクションからの保護
  - HTTP ヘッダー、HTTP 本文、URI 文字列に対するサイズ制約や正規表現でのマッチング
- ALBの設定で、フェイルオープンを設定可能

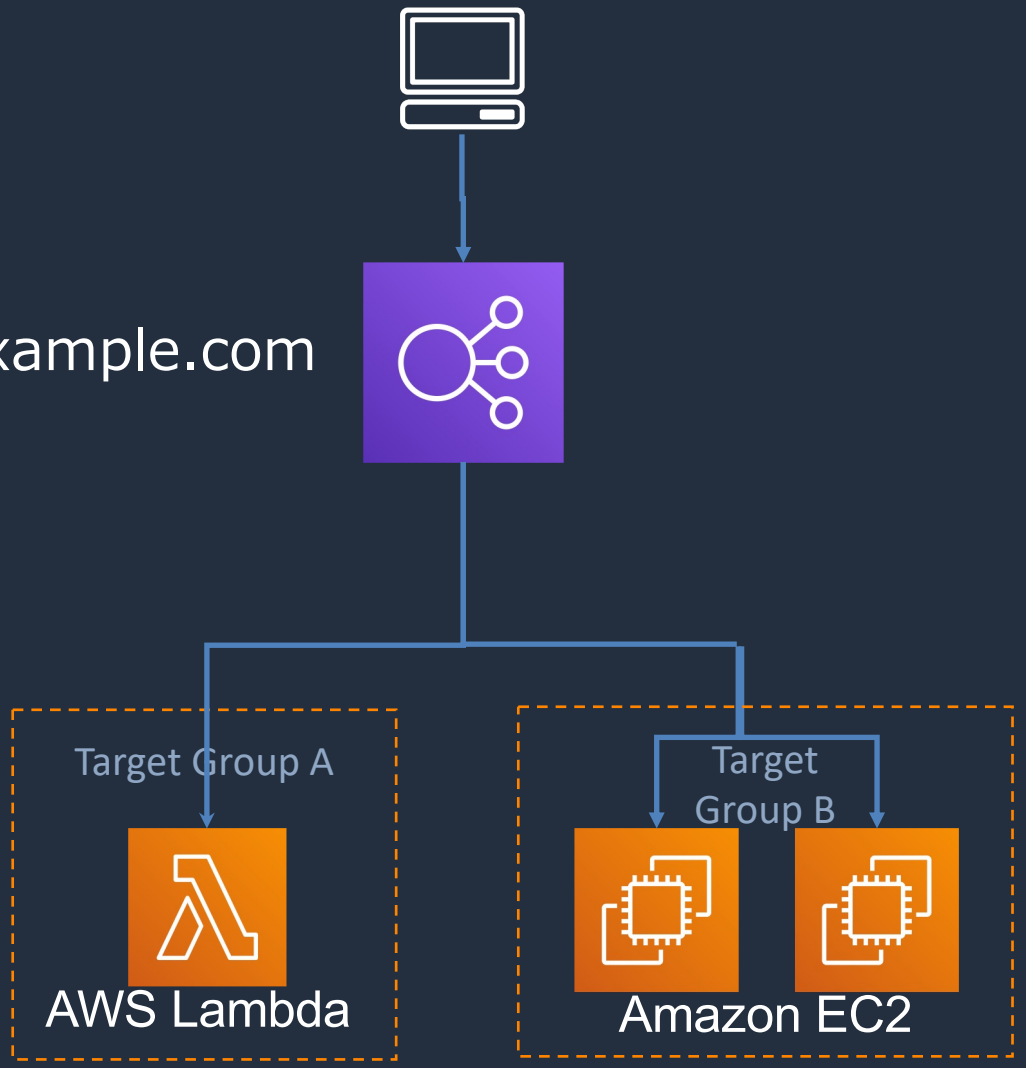


AWS Global Acceleratorとの連携



- Global AcceleratorのエンドポイントにALB, NLB を指定できる
- ユーザーは近いエッジロケーションからAmazon Global Networkを経由して最も近いリージョンにアクセス可能
- ユーザーはマルチリージョンアプリケーションへのアクセスに対して固定IPでアクセス可能
- ALBのIPを固定にできる

Lambda 関数をターゲットにする(ALBのみ)

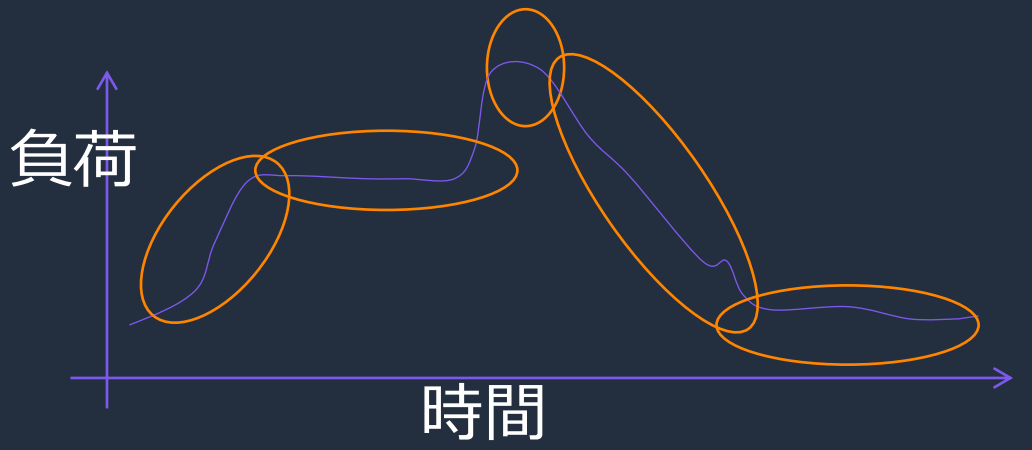


- Lambda 関数をターゲットとするターゲットグループを作成し、リクエストを転送するリスナールールを設定可能
- ターゲットグループにリクエストを転送すると、Lambda 関数を呼び出し、リクエストのコンテンツを JSON 形式で Lambda 関数に渡す
- VPCが同一であれば、１つのALBに対してLambdaのターゲトグループにもEC2またはECSのターゲットグループにもリクエストを転送可能

負荷テスト

推奨テストアプローチ

- 想定する最大負荷のテスト
- 通常のトラフィック時のテスト
  - トラフィックの多い時
  - トラフィックの少ない時
  - トラフィックの傾向に変化がある時（朝や昼の時間帯など）
- 短い時間でトラフィックが大きく変化する場合のテスト



ELB以外にも負荷生成クライアント、バックエンドEC2インスタンスも監視すべき

- アプリケーション内部の動作も要確認
- どこかボトルネックになっているか把握しておく

Amazon EC2 Testing Policy

- ネットワーク負荷テストついてのガイドライン
  - 詳しくは：https://aws.amazon.com/jp/ec2/testing/
- ALBの負荷試験を行う際は、同時にEC2にも負荷をかけることになる
- DDoSのようなトラフィックとみなされるものは禁止されている
- 必ずガイドラインを参照すること

負荷テストの注意事項

- ALBのスケーリングに関わる留意事項
  - ALBではスケールが間に合わないと、遅延やタイムアウトが発生する
  - たとえば、1回目の負荷試験で発生した問題が2回目以降に再現しない場合、1回目のみスケーリングが間にあわなかった可能性がある
  - そのため、テストの際は徐々にトラフィックを増加させるように構成する必要がある。それができない場合や、**典型的ではない極端な負荷が発生した場合**の動作を評価したい場合は、AWSサポートに連絡し、暖気運転(Pre-Warming）の相談を
  - 徐々にスケールさせる際は、トラフィックが5分間隔で50%以上増加しないようにすることを推奨

- 名前解決に注意
  - ELBのDNSレコードはTTLが60秒となっている。クライアント側で少なくとも60秒に1回DNSの再解決をすることを想定している、テストの際も注意を

- 分散するようなトラフィックでテストする
  - 同じCookieでリクエストを続けた場合などは振り分けに偏りが発生
  - NLBのstickinessは送信元IPベースなので、同じクライアントから大量に送ると偏りが発生

https://aws.amazon.com/jp/articles/best-practices-in-evaluating-elastic-load-balancing/

料金

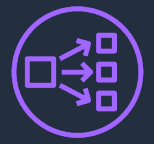
料金

ALB

- **ALB毎の料金＋LCU消費の料金が1時間単位**
  - ALB 1つにつき1時間毎に$0.0243 (東京リージョン)
    - 参考: NLBも$0.0243
  - 1 LCU 1時間毎に$0.008 (全リージョン)

ALBの1 LCUの単位

- 新規接続数: 25/sec
- Active接続数: 3,000/min
- 帯域: 2.22 Mbps (1 GB/hour)
- Ruleの評価数 1,000/sec



NLB

- **NLB毎の料金＋LCU消費の料金が1時間単位**
  - NLB 1つにつき1時間毎に$0.0243 (東京リージョン)
  - 1 LCU 1時間毎に$0.006 (全リージョン)
    - 参考: ALBは$0.008
    - LCUの定義が異なるのでALB/NLBどちらが安いかは比較すること。

NLBの1 LCUの単位

- 新規TCP接続又はフロー数: 800/sec
- Active接続又はフロー数: 10,000/min
- 帯域: 2.22 Mbps (1 GB/hour)

https://aws.amazon.com/jp/elasticloadbalancing/pricing/

本資料に関するお問い合わせ・ご感想

技術的な内容に関しましては、有料のAWSサポート窓口へ
お問い合わせください

https://aws.amazon.com/jp/premiumsupport/

料金面でのお問い合わせに関しましては、カスタマーサポート窓口へ
お問い合わせください（マネジメントコンソールへのログインが必要です）

https://console.aws.amazon.com/support/home#/case/create?issueType=customer-service

具体的な案件に対する構成相談は、後述する個別相談会をご活用ください

ご感想はTwitterへ！ハッシュタグは以下をご利用ください
#awsblackbelt

その他コンテンツのご紹介

ウェビナーなど、AWSのイベントスケジュールをご参照いただけます

https://aws.amazon.com/jp/events/

ハンズオンコンテンツ

https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-hands-on/

AWS 個別相談会

AWSのソリューションアーキテクトと直接会話いただけます

https://pages.awscloud.com/JAPAN-event-SP-Weekly-Sales-Consulting-Seminar-2021-reg-event.html

Thank you!