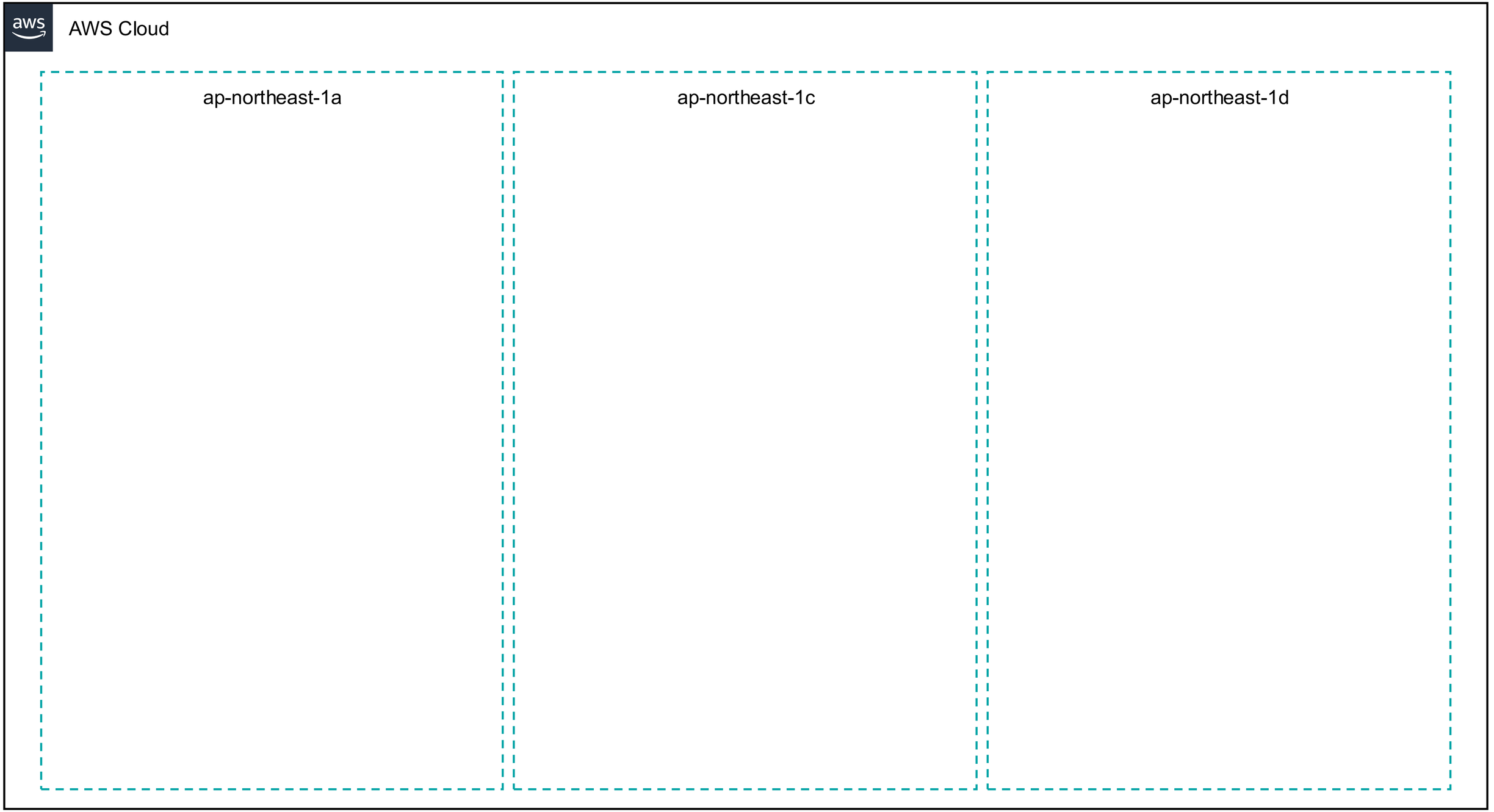
Virtual Private Cloud

Amazon VPCで始めるネットワーク入門

2024/08/03


AWS Cloud
ap-northeast-1a
ap-northeast-1c
ap-northeast-1d

Virtual Private Cloud

Amazon VPC

- VPCはAWS上に構築する
  仮想ネットワークのこと

- **リージョンを跨ぐことはできないが、**
  AZを跨いで構築できる

- 仮想ネットワークはリージョン内に構築され、
  それぞれのVPCは独立している

- デフォルトを使うではなく、自分で作ることで
  セキュリティを向上させられる

IP Address

IPアドレス

- IPアドレスは**全てのインターネットに接続する端末へ割り当て**られる
- IPアドレスには現在、**IPv4とIPv6の2種類が存在**する
- **IPv4**には0.0.0.0 ~ 255.255.255.255の約43億(2の32乗)個があるが、**インターネットの普及によって枯渇**してきた
- **IPv6**は 0000:0000:0000:0000:0000:0000:0000:0000 ~ ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff の約128澗(2の128乗)個あり、**ほとんど枯渇の恐れがない**

IP Address

IPアドレス

Column

- IPv4枯渇の影響により、AWSでは2024年2月から
  パブリックIPv4に対し、課金を開始
  - パブリックIPv4には1時間あたり $0.005 のコストが発生
  - 月間 $3.72 (31日/月で計算)ものコストが発生するため、
    できるだけIPv4を減らした方が良い
- できるだけIPv6を使いたいが、ロードバランサーはIPv4ベースの
  ルートをするため、**完全な移行は難しい**

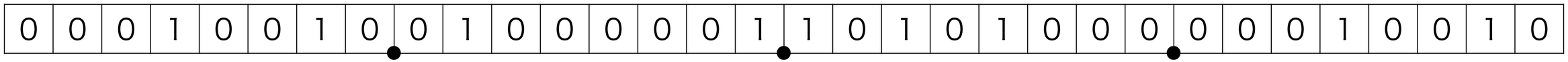
IPv4

IPv4の構造

例えば、

18.65.168.18

'.' で分割して ⬇ 8ビットの2進数に変換



Private IPv4

プライベート（ローカル）IPv4の構造

- プライベートIPは**アドレス前半にネットワーク部、後半にホスト部**が指定されている

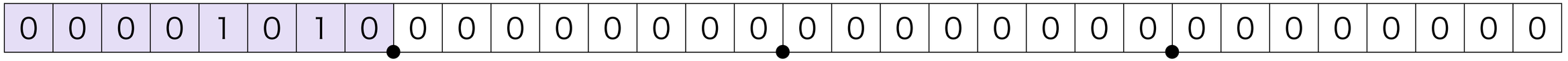
- **クラスによって指定されている範囲が異なり、クラスに沿ってアドレス設計**されることが多い

- クラスはネットワーク部、ホスト部に指定できる段階が少なく、クラスを変えると**一気にIPアドレスの数が変化してしまう**問題がある

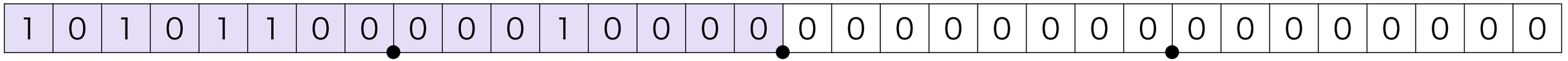
Private IPv4

プライベート（ローカル）IPv4の構造

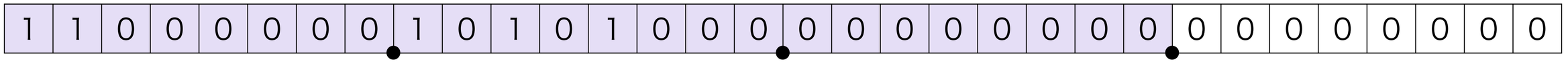
クラスA: 10.0.0.0 ~ 10.255.255.255（約1600万台）



クラスB: 172.16.0.0 ~ 172.31.255.255（約6.5万台）



クラスC: 192.168.0.0 ~ 192.168.255.255（256台）



塗りつぶしがネットワーク部を示す

Classless Inter-Domain Routing

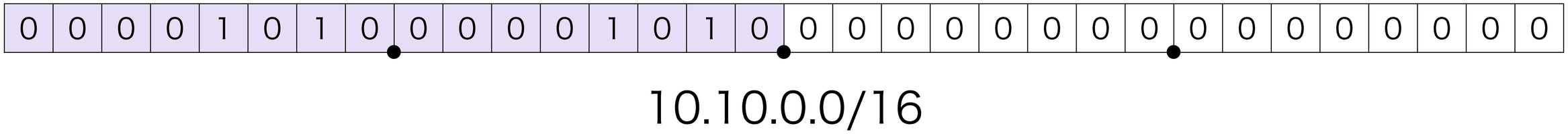
CIDRとは

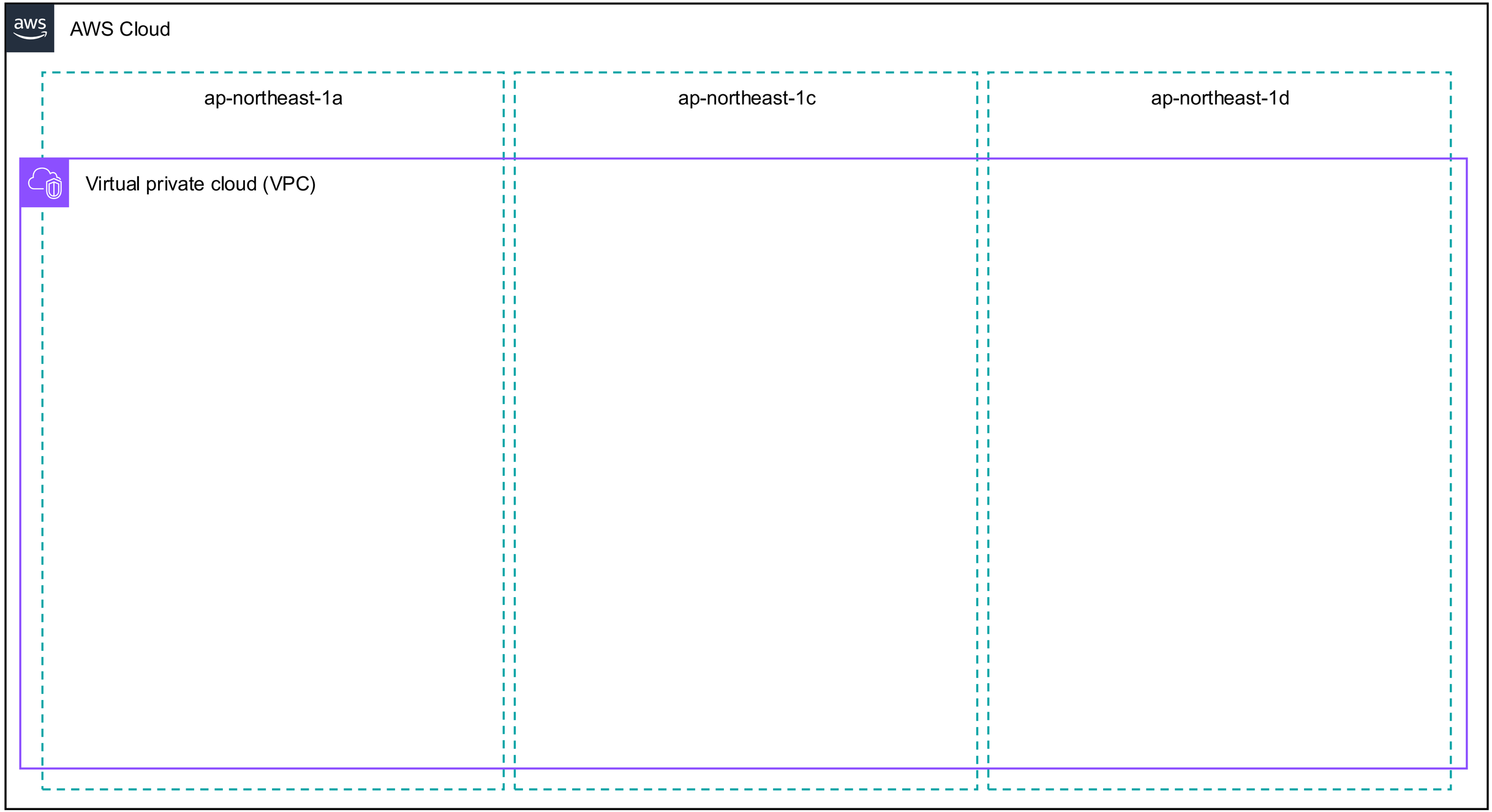
- CIDRとは**クラスを使用せずに**ネットワーク部、ホスト部を分割する方法
- クラスを使ったものと比較して、より**自由な設計**ができる
- AWSなどの**クラウドではCIDRを使用**する

VPC

VPCにおけるCIDR設計

- VPCにおける **IPv4 CIDR範囲は /16 ~ /28** と決まっている
- /後の数字は上から何ビットネットワーク部にするかを表す (**マスク**)
- **基本的には10.0.0.0/16を使用すれば良い**が、
  別のVPCとピアリングする際に同じだとできないため、
  10.10.0.0/16にすることもある




AWS Cloud
ap-northeast-1a
ap-northeast-1c
ap-northeast-1d
Virtual private cloud (VPC)

Subnet

サブネットとは

- サブネットは**VPC内をさらに分割**するネットワーク
- **AZを跨いだサブネットは作成できず、1AZごとに1つは最低必要**となる
  （使用するAZのみでも良いが、全て作ることを推奨）
- サービスごとにサブネットを分けると、
  サブネット単位でパブリックかプライベートかを選べるようになる
- サービスは今後増える可能性があるので、サブネットマスクの**上位ビット**
  AZは増える可能性が低いので、サブネットマスクの**下位ビット**に配置する
- サブネットCIDRは**アドレス範囲の開始場所**を表す

Subnet

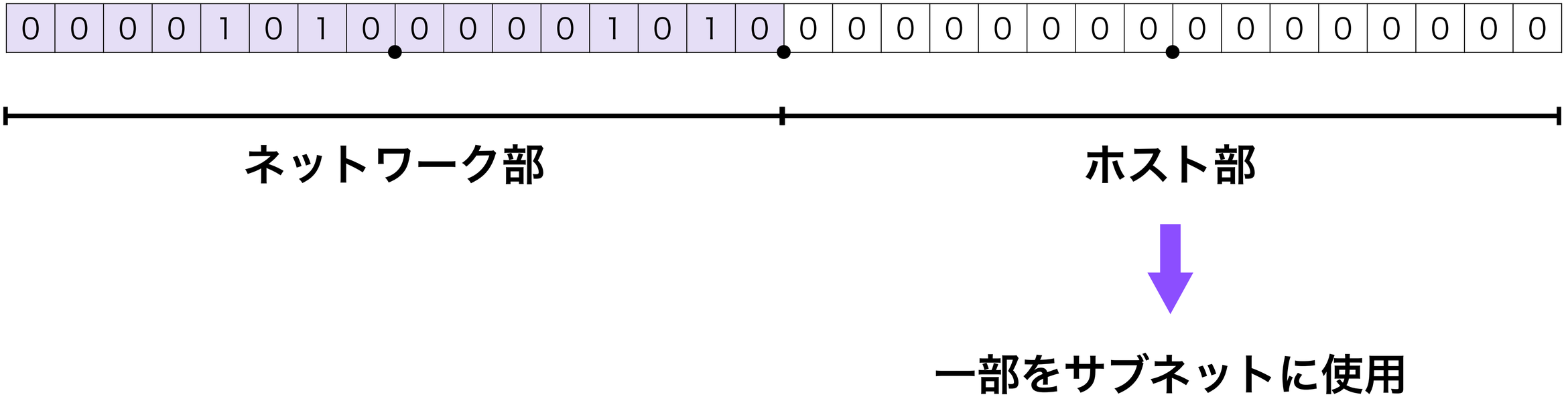
サブネットとは

- AZを識別するアドレス部には**AZの識別ができるビット数を渡せば良い**
  - **AZ数 < 2^n** の不等式が成立する最小の n が必要なビット数
    東京リージョンの場合、3AZ < 2^2 = 4 なので、**2ビット必要**
- サービスを識別するアドレス部には**サービスの個数**を考えれば良いが、
  今後、サービスが増えることも考えて、**多めに見積もる**必要がある
  - サービスのアドレス部にも**AZと同様の不等式**を用いて考える
- また、**ホスト部のアドレスが過剰であれば、サービス部を増やして調整**する

CIDR

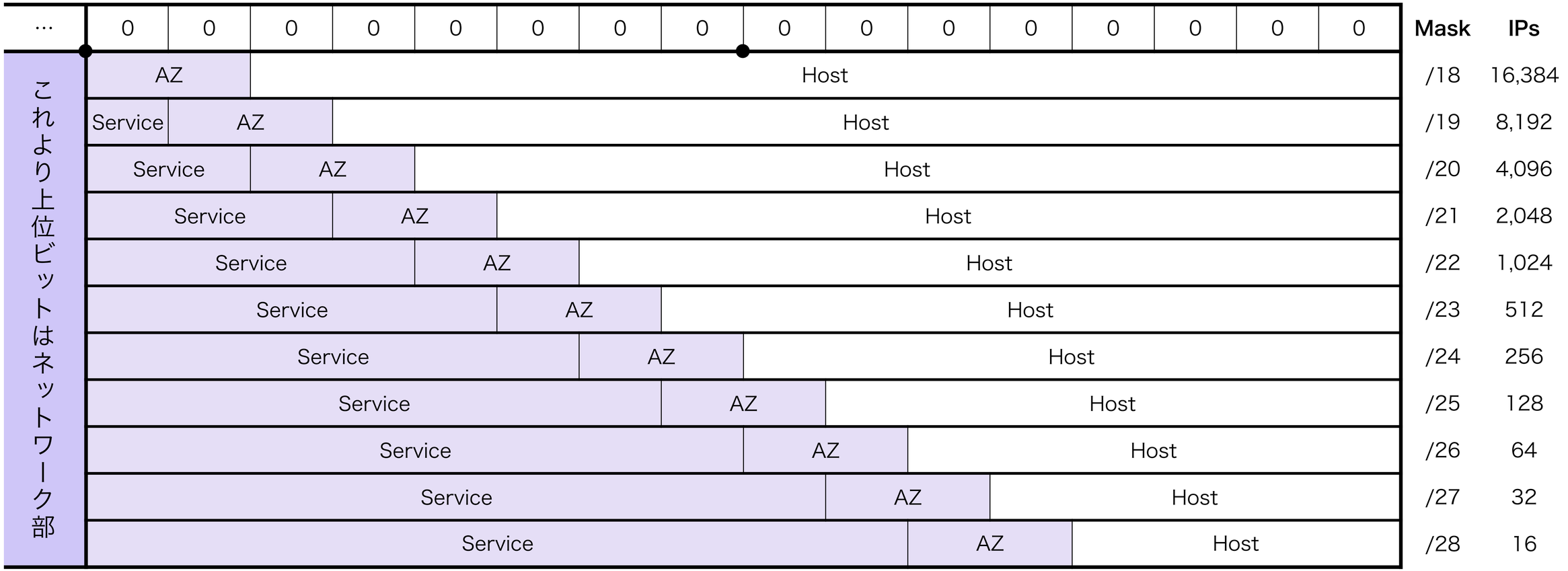
CIDR設計

現在の VPC CIDR 設計



CIDR

CIDR設計



塗りつぶしがサブネットマスクを示す

CIDR Example

CIDR設計例

- 要件
  - VPCのCIDR: **10.10.0.0/16**
  - リージョン: ap-northeast-1 (**3AZ**)
  - 必要なサービス: Web, App, DB (**3 Services**)

この要件が定義されている場合、どのようにCIDR設計をすれば良いか

CIDR Example

CIDR設計例

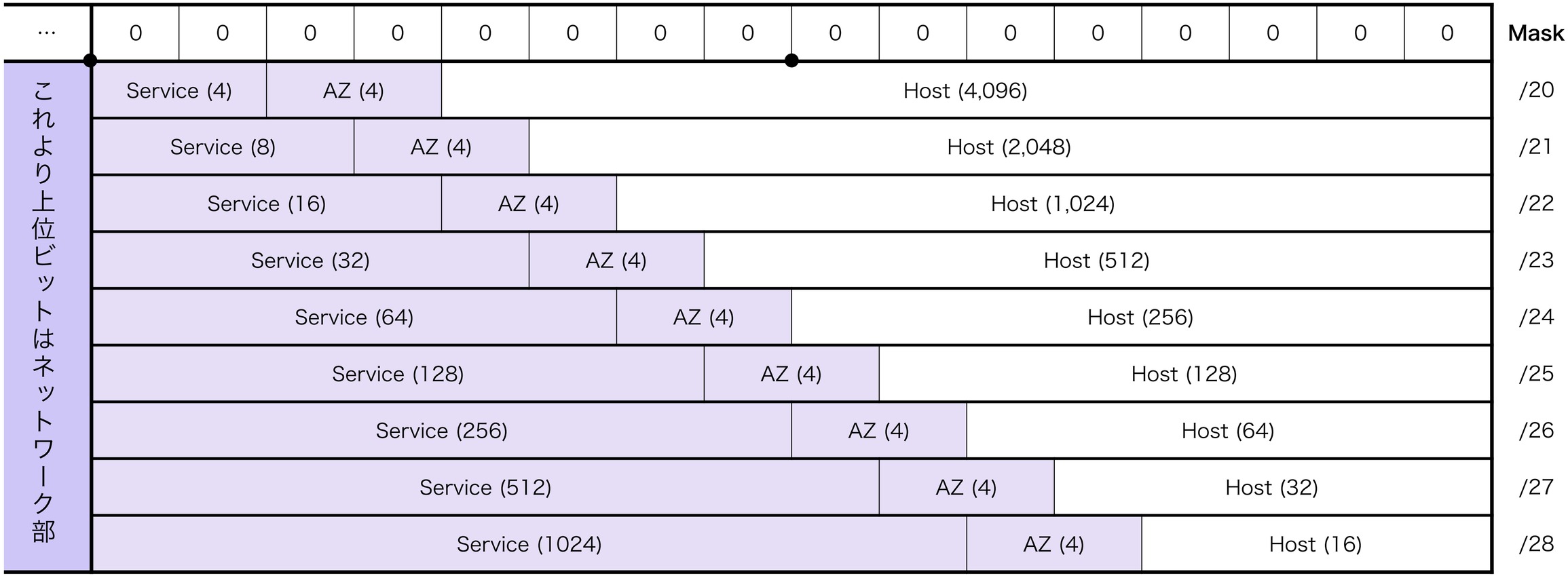
- リージョン内 3AZ分のサブネットを作成する場合、
  3[AZ] < 2^n[ビット] より、**AZの識別に最低2ビットが必要**

- 3 Services 分のサブネットを作成する場合、
  3[Service] < 2^n[ビット] より、**サービスの識別に最低2ビット必要**

- サービスが今後増える可能性やサブネット内のホスト数を加味して、
  サービス部を増やす

CIDR Example

CIDR設計例



塗りつぶしがサブネットマスクを示す

CIDR Example

CIDR設計例

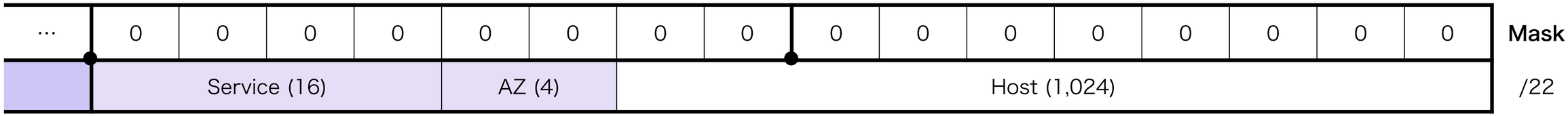
- リージョンとサービスのビットはそれぞれ最低2ビットだった
  → **VPC CIDRと足して、最低/20**となる

- サービスが今後増えることを加味し、サービス部を増やしておく
  → **サービス数を最大8(2^3)個や16(2^4)個にし、/21や/22**とする

- 今回は/22で設計する

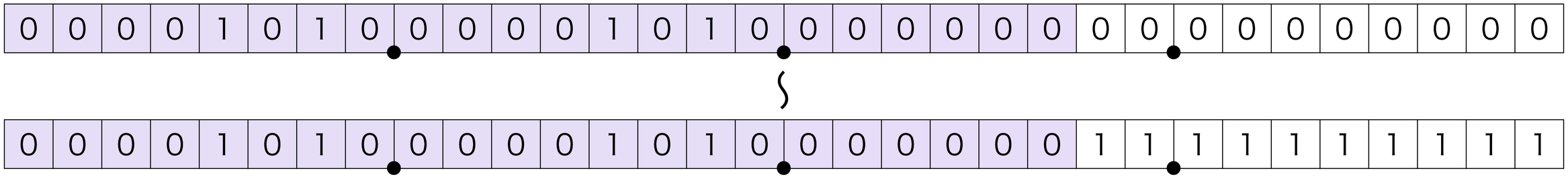
CIDR Example

CIDR設計例



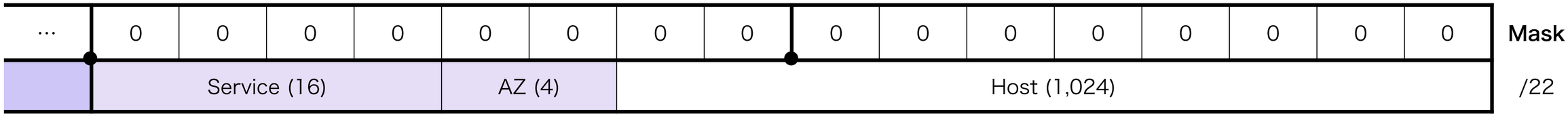
CIDRはアドレス範囲の開始場所を表す

今回の場合、使用できる最初のサブネットの範囲はVPC CIDRと組み合わせて、
10.10.0.0 ~ 10.10.3.255となる (塗りつぶしはサブネット内で固定)



CIDR Example

CIDR設計例



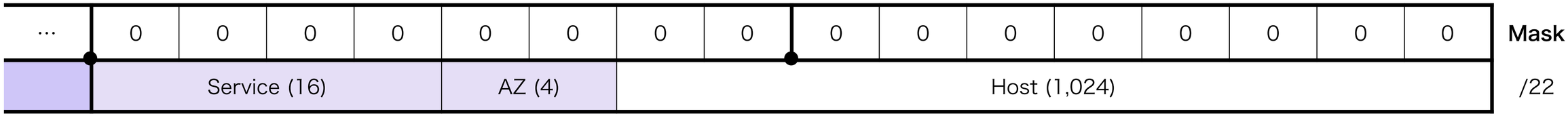
今回は下記のように設定し、識別を行う

Service	ビット
Web	0000
App	0001
DB	0010

AZ	ビット
ap-northeast-1a	00
ap-northeast-1c	01
ap-northeast-1d	10

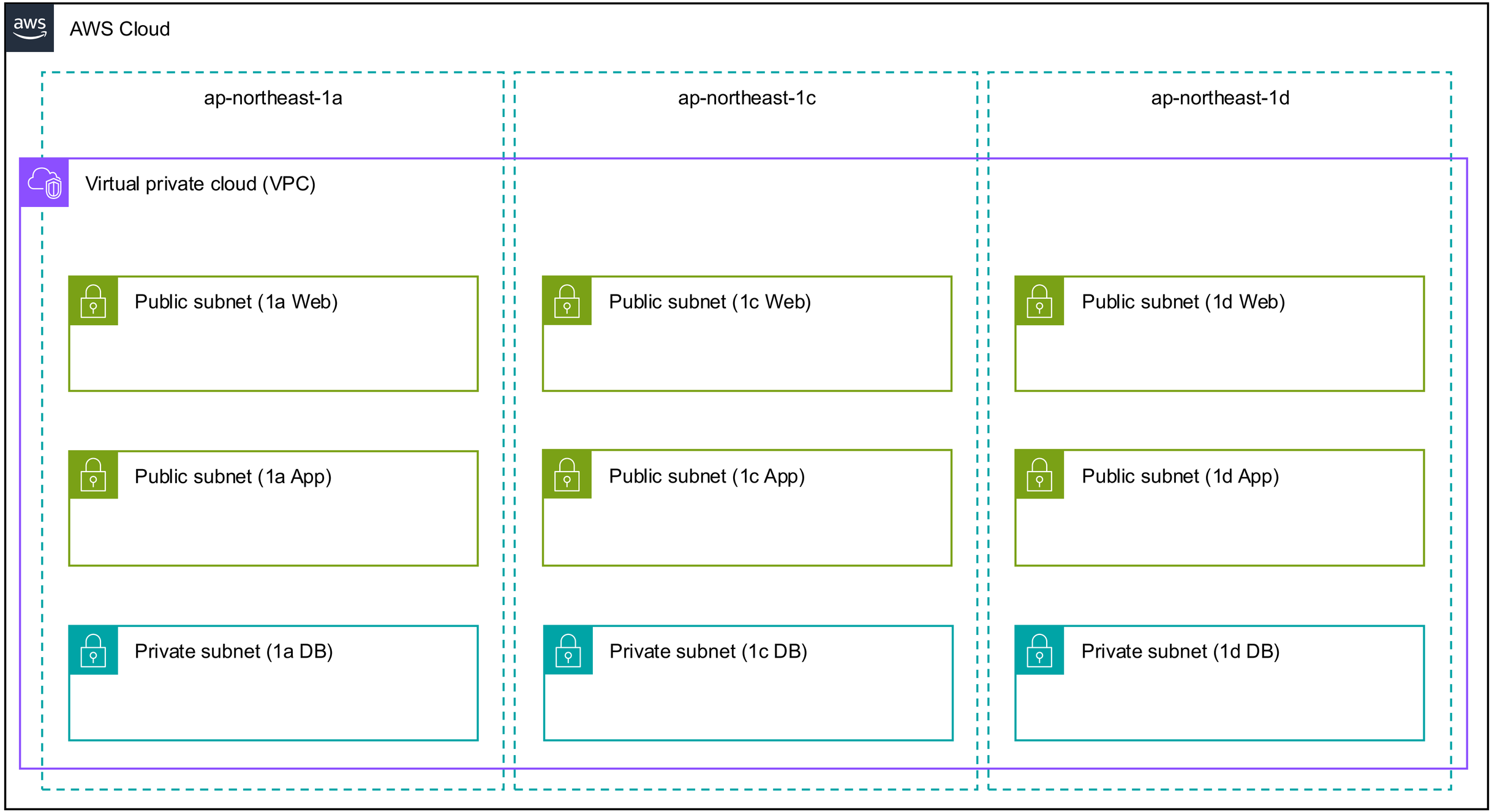
CIDR Example

CIDR設計例



ServiceとAZを組み合わせると、サブネットCIDRは下記のようになる

Service \ AZ	1a	1c	1d
Web	10.10.0.0/22	10.10.4.0/22	10.10.8.0/22
App	10.10.16.0/22	10.10.20.0/22	10.10.24.0/22
DB	10.10.32.0/22	10.10.36.0/22	10.10.40.0/22


AWS Cloud
ap-northeast-1a
ap-northeast-1c
ap-northeast-1d
Virtual private cloud (VPC)
Public subnet (1a Web)
Public subnet (1c Web)
Public subnet (1d Web)
Public subnet (1a App)
Public subnet (1c App)
Public subnet (1d App)
Private subnet (1a DB)
Private subnet (1c DB)
Private subnet (1d DB)

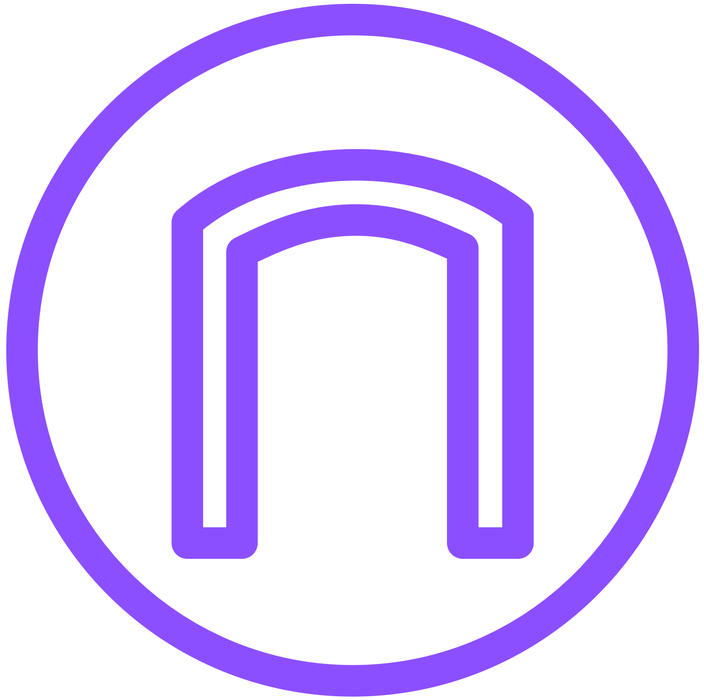
Internet Gateway

インターネットゲートウェイ

- インターネットゲートウェイは
  VPC内部と外部インターネットが
  相方向通信できるようにするゲートウェイ

- IGWを接続すれば、サーバーや
  ロードバランサーに**外部インターネットから**
  アクセスできるようになる

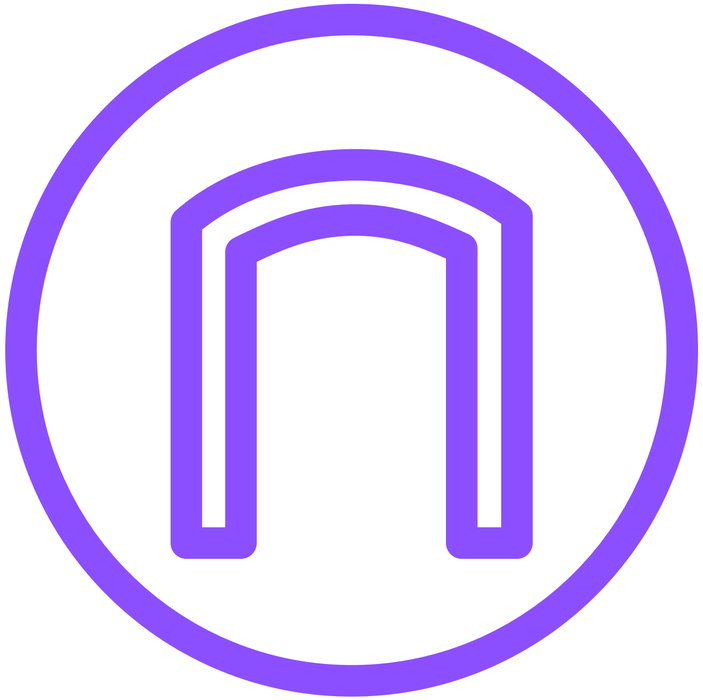
- **パブリックサブネットが**
  外部ネットワークと通信する際に必要



Internet Gateway

インターネットゲートウェイ

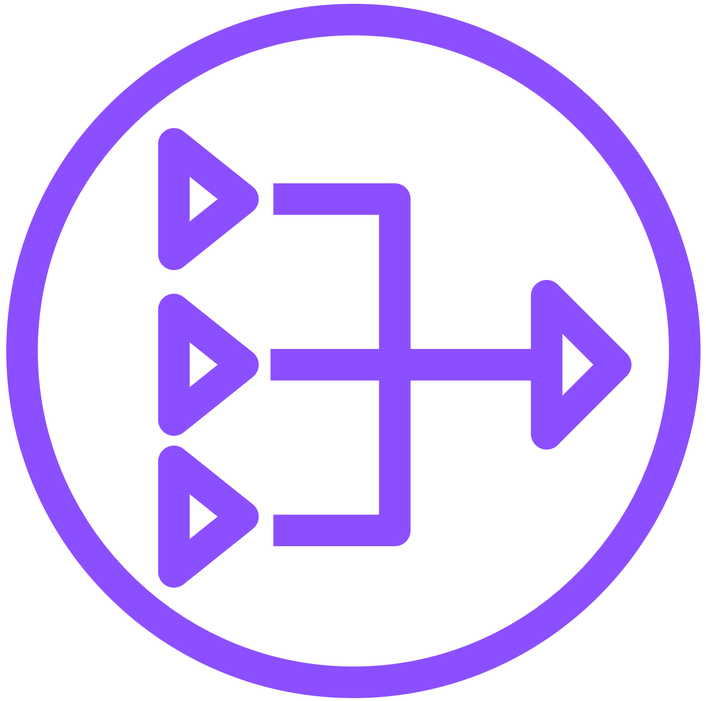
- それぞれのインスタンスに
  パブリックIPが付与されている必要がある
  - パブリックIPv4料金が発生
- VPCにアタッチする必要がある



NAT Gateway

NATゲートウェイ

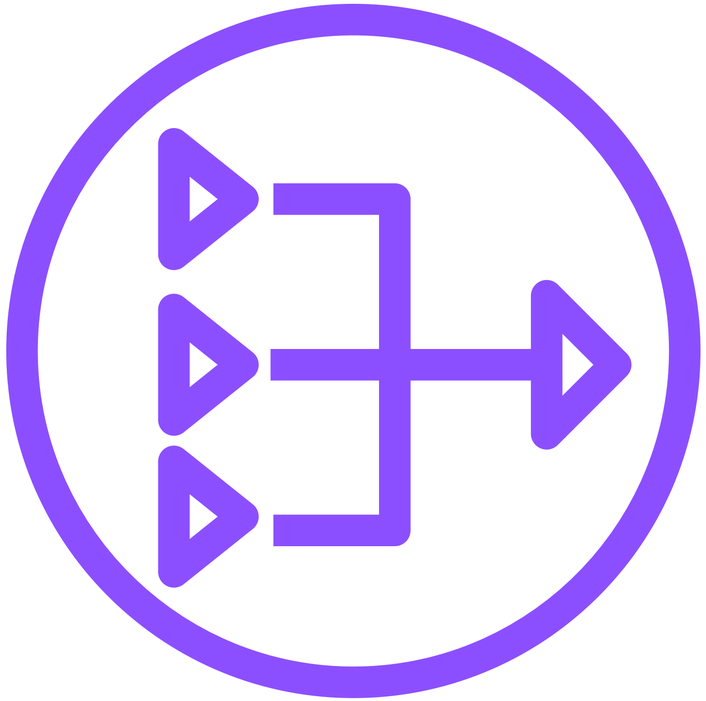
- NATゲートウェイは
  VPC内部から外部インターネットへの
  外向き通信をできるようにするゲートウェイ
- **プライベートサブネットにあるインスタンスの**
  ソフトウェアアップデートなどに必要
- **プライベートサブネットが**
  外部ネットワークと通信する際に必要



NAT Gateway

NATゲートウェイ

- 外部から直接アクセスできず、
  プライベートサブネットの安全性を保てる
- 外部インターネットへアクセスする際に
  プライベートIPをパブリックIPに一時的に変換
- プライベートIPで運用するため、
  パブリックIPv4料金が発生しない特徴がある

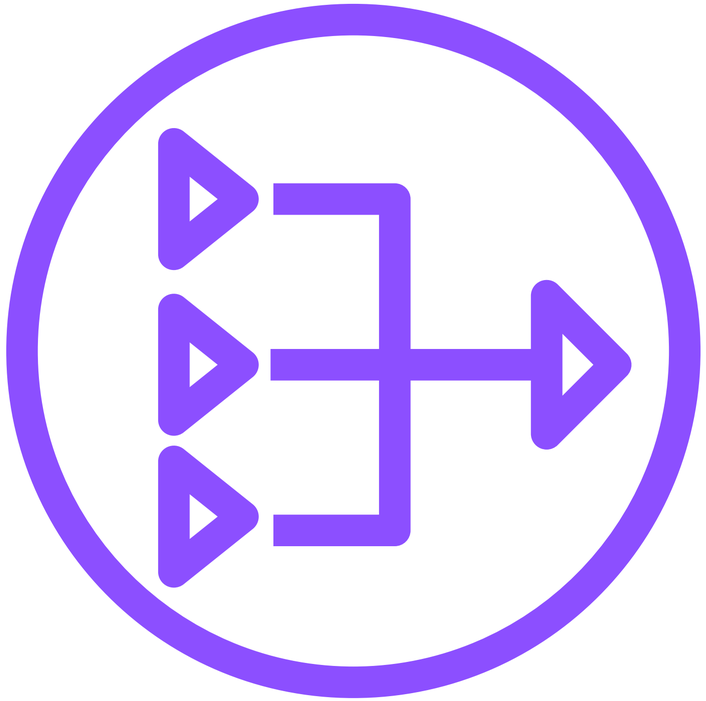


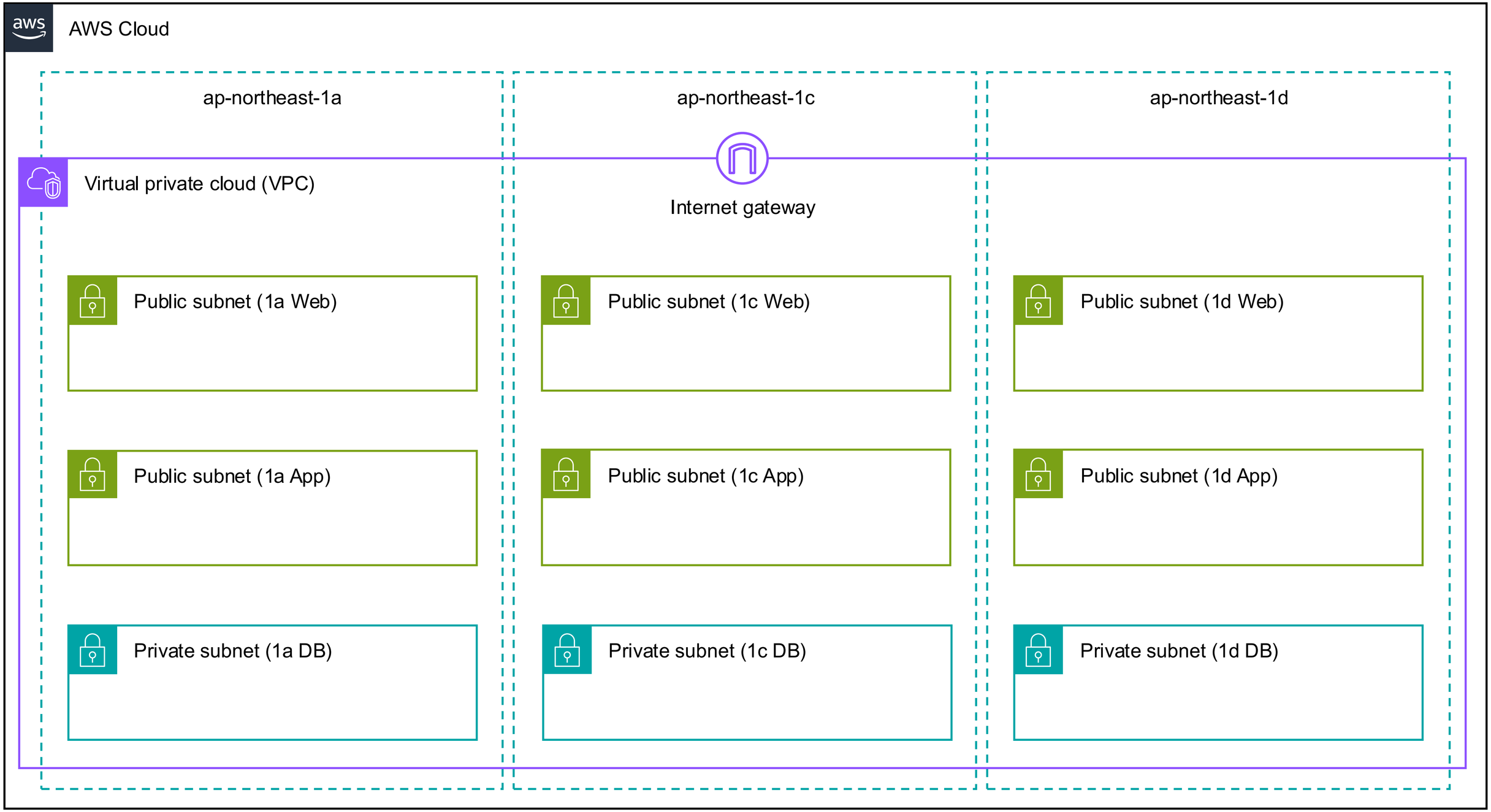
NAT Gateway

NATゲートウェイ

- NATゲートウェイは $0.062/h と $0.062/GBの
  合計金額になっており、**高価なサービス**のため、
  商用サービスなどでは使用するが、
  私用サービスなどでは使用しないことが多い

- 通信がない場合でも、
  月間 $46.128 (31日/月で計算)もの
  コストが発生する




AWS Cloud
ap-northeast-1a
ap-northeast-1c
ap-northeast-1d
Virtual private cloud (VPC)
Internet gateway
Public subnet (1a Web)
Public subnet (1c Web)
Public subnet (1d Web)
Public subnet (1a App)
Public subnet (1c App)
Public subnet (1d App)
Private subnet (1a DB)
Private subnet (1c DB)
Private subnet (1d DB)

Route Table

ルートテーブル

- ルートテーブルはその**サブネットからどこへアクセスできるかを指定する**もの
- 通信がたどる**ルートを書かれた案内標識**のようなもの
- ルートテーブルは「**行き先**」を指定するものであり、
  受け入れる**送信元を指定するものではない**
- 通常は自分のVPC範囲内である、
  送信先: 10.10.0.0/16 の ターゲット: localと
  送信先: 0.0.0.0/0 の ターゲット: インターネットゲートウェイを指定する

Network ACL

ネットワークACL

- ネットワークACLはその**サブネットへどこからのインバウンド、サブネットからどこへのアウトバウンドを許可/拒否するかを指定する**もの
- **ホワイトリスト・ブラックリスト方式**で設定
- 作成時には**双方向全てのトラフィックが許可**されている
- **ルール番号が小さい方から順に当てはまるものをチェック**し、当てはまった時点で許可/拒否が決まる

Security Group

セキュリティグループ

- セキュリティグループはその**インスタンスへどこからのインバウンド、サブネットからどこへのアウトバウンドを許可/拒否するかを指定する**もの
- **ホワイトリスト方式**で設定
- 作成時には**双方向全てのトラフィックが拒否**されている
- ネットワークACLの制限の方が厳しい場合、**トラフィックを許可していても、ネットワークACLの設定に従う**

Network ACL vs Security Group

ネットワークACL vs セキュリティグループ

- どちらもアクセス拒否などの機能を持ち、**ファイアウォール的に使用可能**
- 違いはネットワークACLは**サブネットに対して**行う設定に対し、
  セキュリティグループは**インスタンスに対して**行う設定
- 細かいトラフィックの制御には**セキュリティグループを使う**
- ネットワークACLはデフォルトのまま運用することもある

Network ACL vs Security Group

ネットワークACL vs セキュリティグループ

- **なぜセキュリティグループで細かい制御を行うのか**
  - ネットワークACLはステートレスで、
    行き通信と戻り通信が別に評価される
  - セキュリティグループはステートフルで、
    行き通信が許可されていれば、自動的に戻り通信も許可される
    - ◉ セキュリティグループは戻り通信が受信できなくならないので、
      設定ミスを防げる

Network ACL vs Security Group

ネットワークACL vs セキュリティグループ

- **なぜセキュリティグループで細かい制御を行うのか**
  - 同一ネットワーク内の通信はネットワークACLを通らないため、
    同一ネットワーク内の通信は制御できない
  - セキュリティグループは**送信元/先にセキュリティグループを指定**できる

このような理由から一般的なユースケースでは**ネットワークACLを緩く、セキュリティグループを厳しく**することが多い

Network ACL Example

ネットワークACLの設定例

- 要件
  - デフォルトのまま使う

Port

ポート

- それぞれのアプリケーションでは
  デフォルトで使用するポートが指定されている

サービス名 / タイプ	デフォルトポート
HTTP	80
HTTPS	443
SSH	22
SMTP	25

サービス名 / タイプ	デフォルトポート
IMAP	143
POP3	110
MySQL / Aurora	3306
PostgreSQL	5432

Protocol

プロトコル

- 通信には**プロトコルというやりとりの方式**が決められている
- トランスポート層には主に2つのプロトコルがある
  - **TCP**
    - 長所: 接続指向であり、信頼性が高い
    - 短所: UDPと比較してリアルタイム性に欠ける
  - **UDP**
    - 長所: 接続レスで、高速なデータ転送を優先する
    - 短所: データの到達を保証しないため、信頼性が低い

Security Group Example

セキュリティグループの設定例

- 要件 (インバウンド)

▸ **Web**

- TCP 80 From All
- TCP 443 From All

▸ **App**

- TCP 80 From Web
- TCP 22 From All

▸ **DB**

- TCP 5432 From App

Security Group Example

セキュリティグループの設定例

- 要件 (アウトバウンド)

▸ **Web**

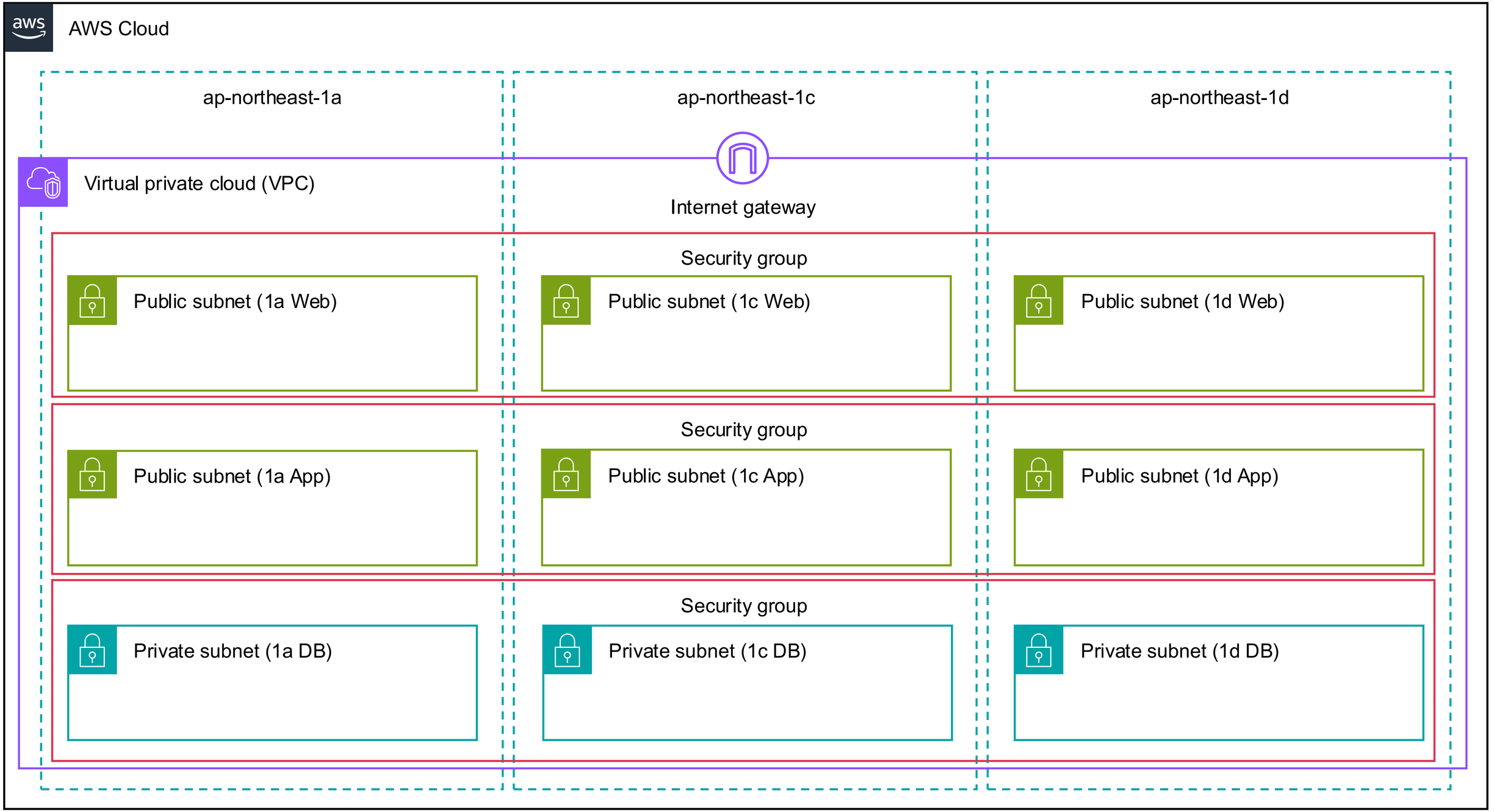
- All To All

▸ **App**

- All To All

▸ **DB**

- All To All


AWS Cloud
ap-northeast-1a
ap-northeast-1c
ap-northeast-1d
Virtual private cloud (VPC)
Internet gateway
Security group
Public subnet (1a Web)
Public subnet (1c Web)
Public subnet (1d Web)
Security group
Public subnet (1a App)
Public subnet (1c App)
Public subnet (1d App)
Security group
Private subnet (1a DB)
Private subnet (1c DB)
Private subnet (1d DB)