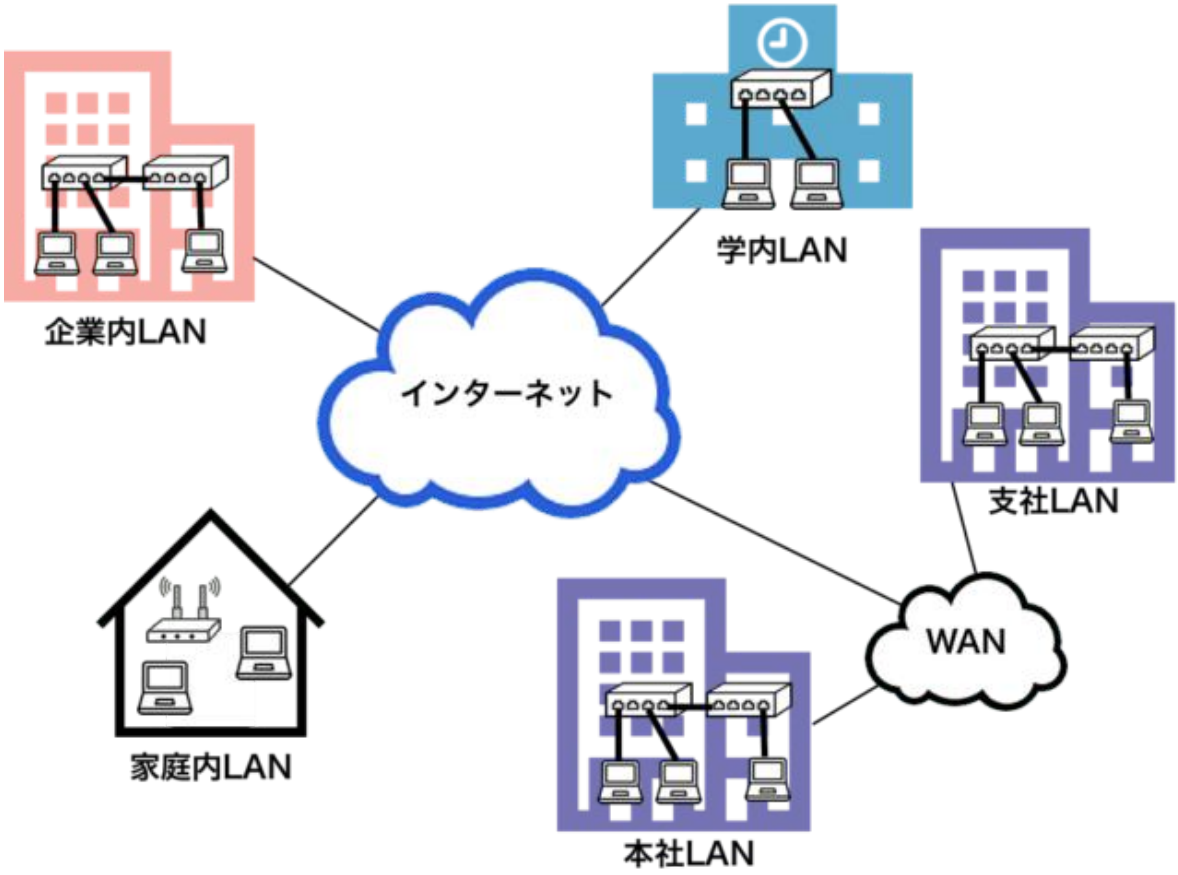
# 01. ネットワークの全体像

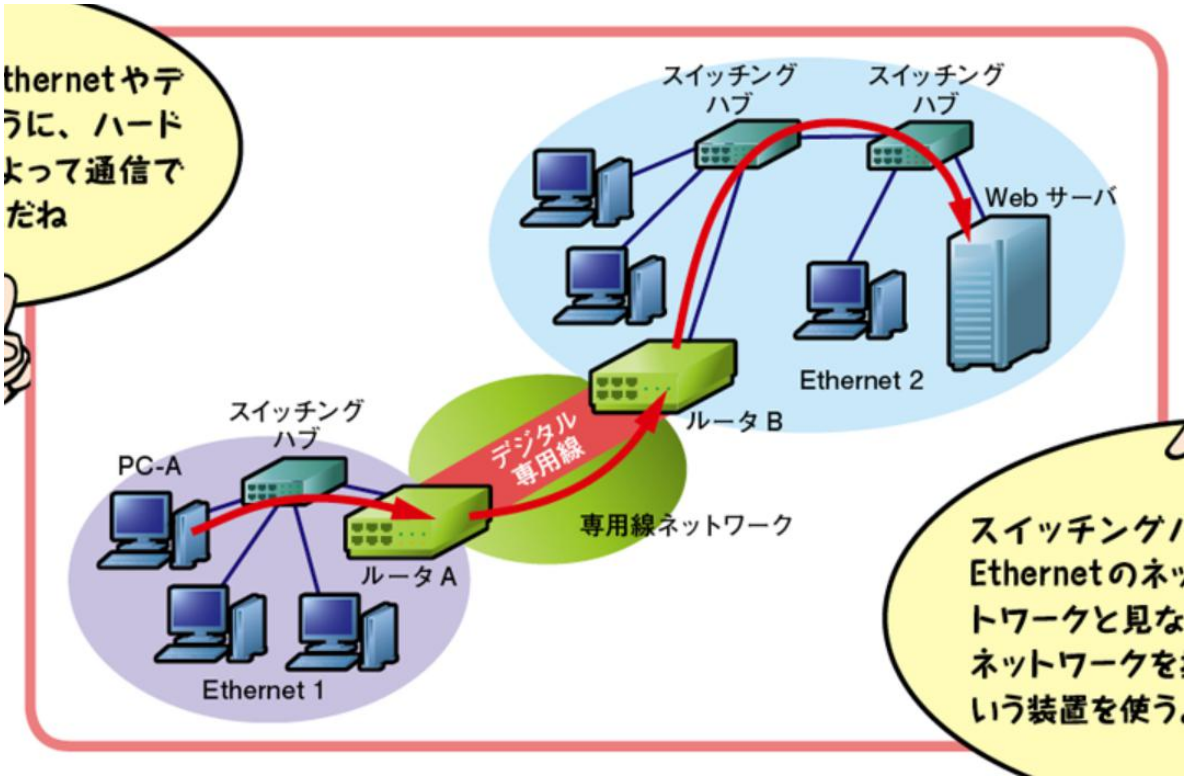
## ◆ インターネット、WAN、LAN

ネットワークには、『インターネット』『WAN』『LAN』がある。家庭内LAN、学内LAN、企業内LAN、企業WANなど、さまざまなネットワークがあり、インターネットは、それぞれのネットワークを互いに接続しているネットワークである。




## ◆ WAN、LANの具体例

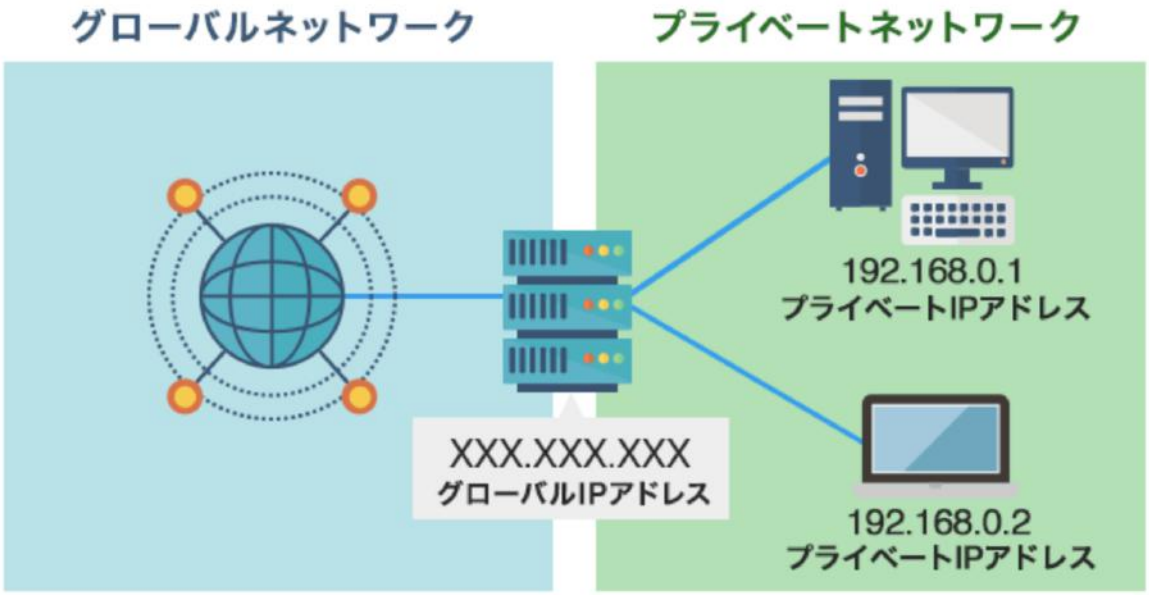
例えば、LANとしてEthernet、WANとしてデジタル専用線を用いる。




## ◆ WANの種類と歴史


## ◆ グローバルネットワークとプライベートネットワーク

ルータを境に、プライベートネットワークとグローバルネットワークに分けられる。ややこしいが、ルータにはグローバルIPアドレスが割り当てられている。




## ◆ プライベートネットワークにおけるセグメント

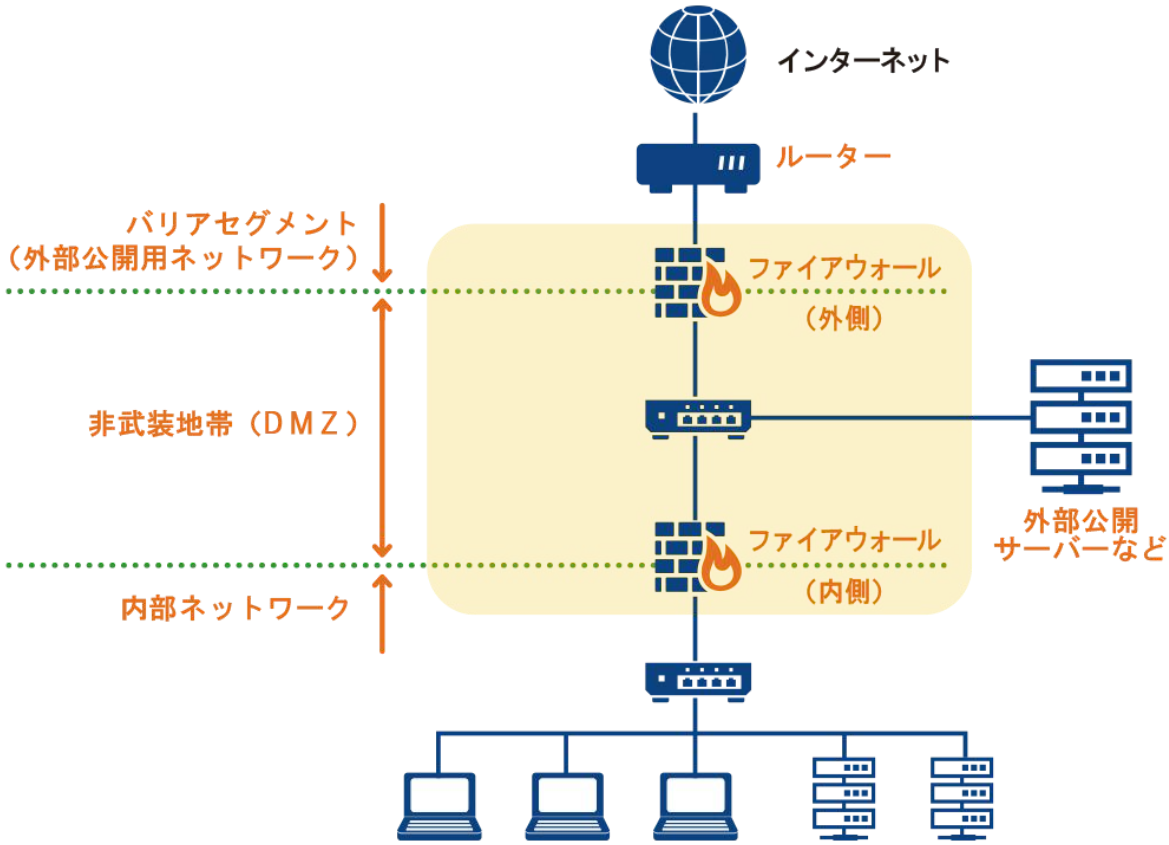
プライベートネットワークは、外部ネットワーク、非武装地帯、内部ネットワークに分類される。

- **非武装地帯に設置するべきサーバの種類**

攻撃の影響が内部ネットワークに広がる可能性を防ぐために、外部から直接リクエストを受ける、『DNSサーバ』、『Webサーバ』、『メールサーバ』、『Proxyサーバ』は非武装地帯に設置するべき。

- **内部ネットワークに設置するべきサーバの種類**

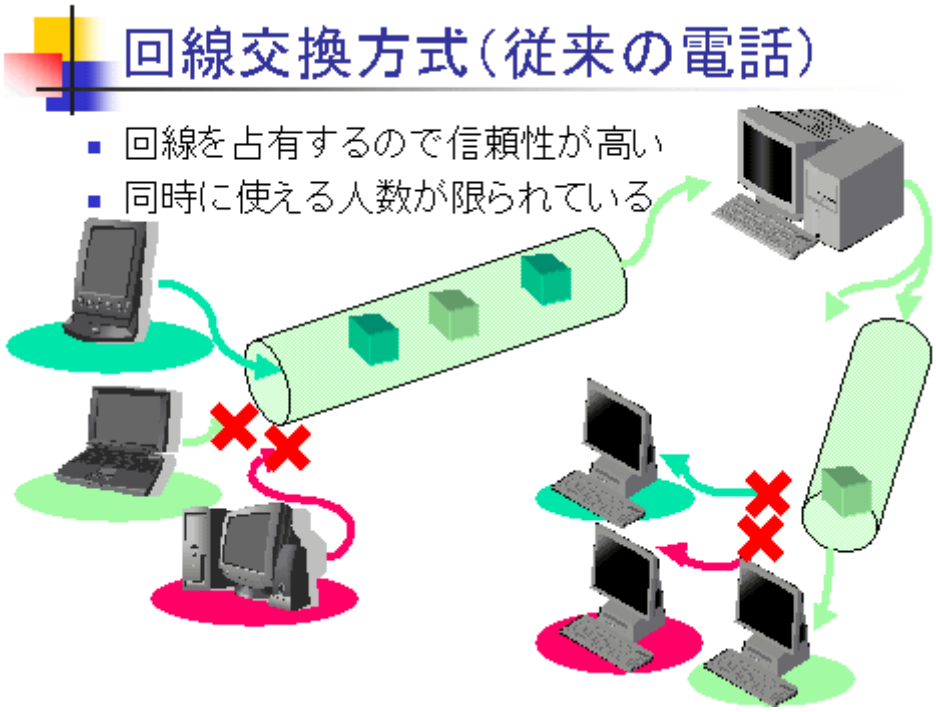
外部からリクエストを受けないデータベースサーバは、内部ネットワークに設置するべき。



### ◆ ネットワークにおけるデータ通信方法の種類

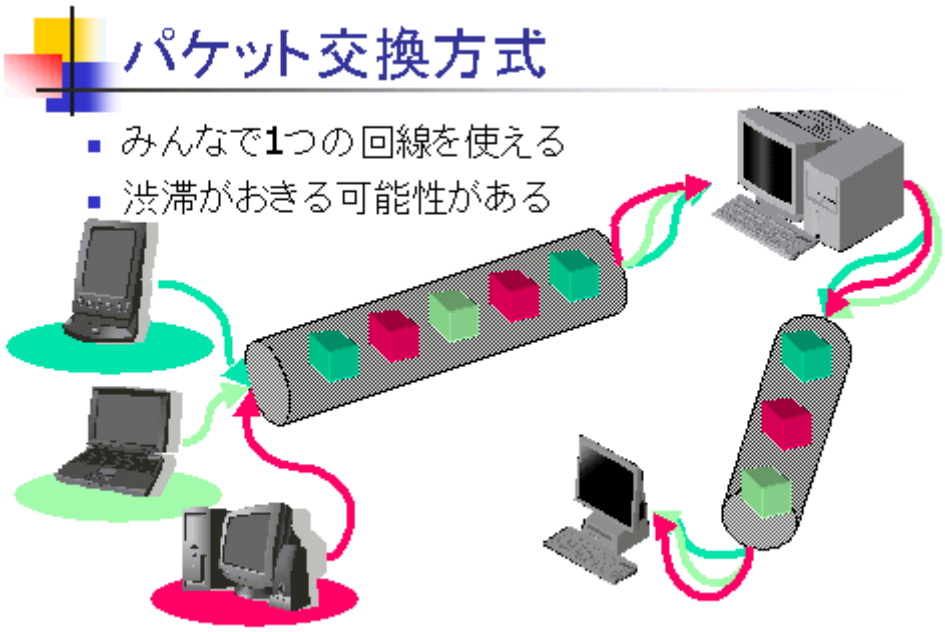
- **回線交換方式**

  少数対少数でデータ通信を行うため、送信時に、送信者と受信者の宛先情報は必要ない。




- **パケット交換方式**

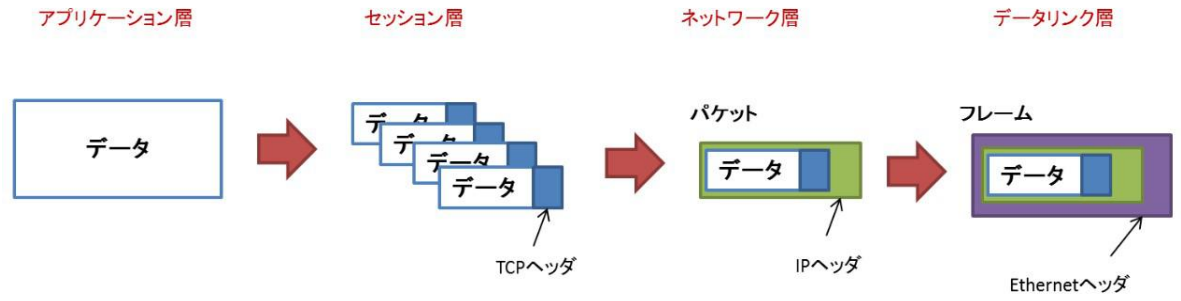
  通信するデータをパケット化する。多数対多数でデータ通信を行うため、送信時に、送信者と受信者の宛先情報が必要になる。




# 02-01. OSI参照モデルとTCP階層モデル

### ◆ データへのヘッダ情報追加とカプセル化

パケット交換方式におけるパケットのヘッダ情報は、パソコンの各概念層のプロトコルによって追加されていく。

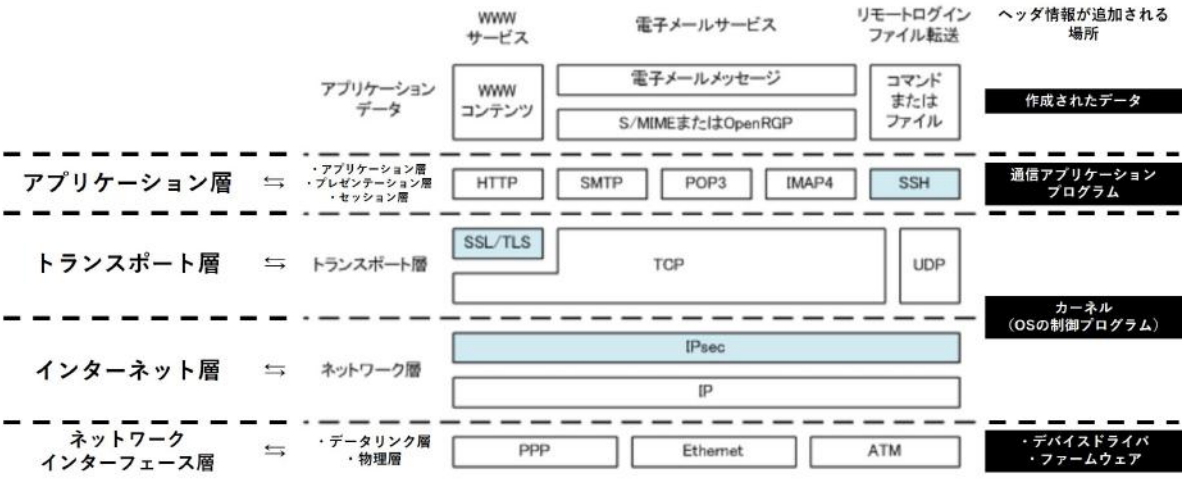



## ◆ OSI参照モデルにおいて各概念層で追加されるヘッダ情報の内容

| OSI参照モデル | | | LAN接続機器 |
|---|---|---|---|
| 第7層 | アプリケーション層 | 通信サービス | ゲートウェイ |
| 第6層 | プレゼンテーション層 | データの表現形式の変換 | |
| 第5層 | セッション層 | 同期制御 | |
| 第4層 | トランスポート層 | システム間のデータ転送 | |
| 第3層 | ネットワーク層 | データの伝送経路を選択 | ルータ |
| 第2層 | データリンク層 | 伝送制御 | ブリッジ |
| 第1層 | 物理層 | 物理的な伝送媒体 | リピータ |

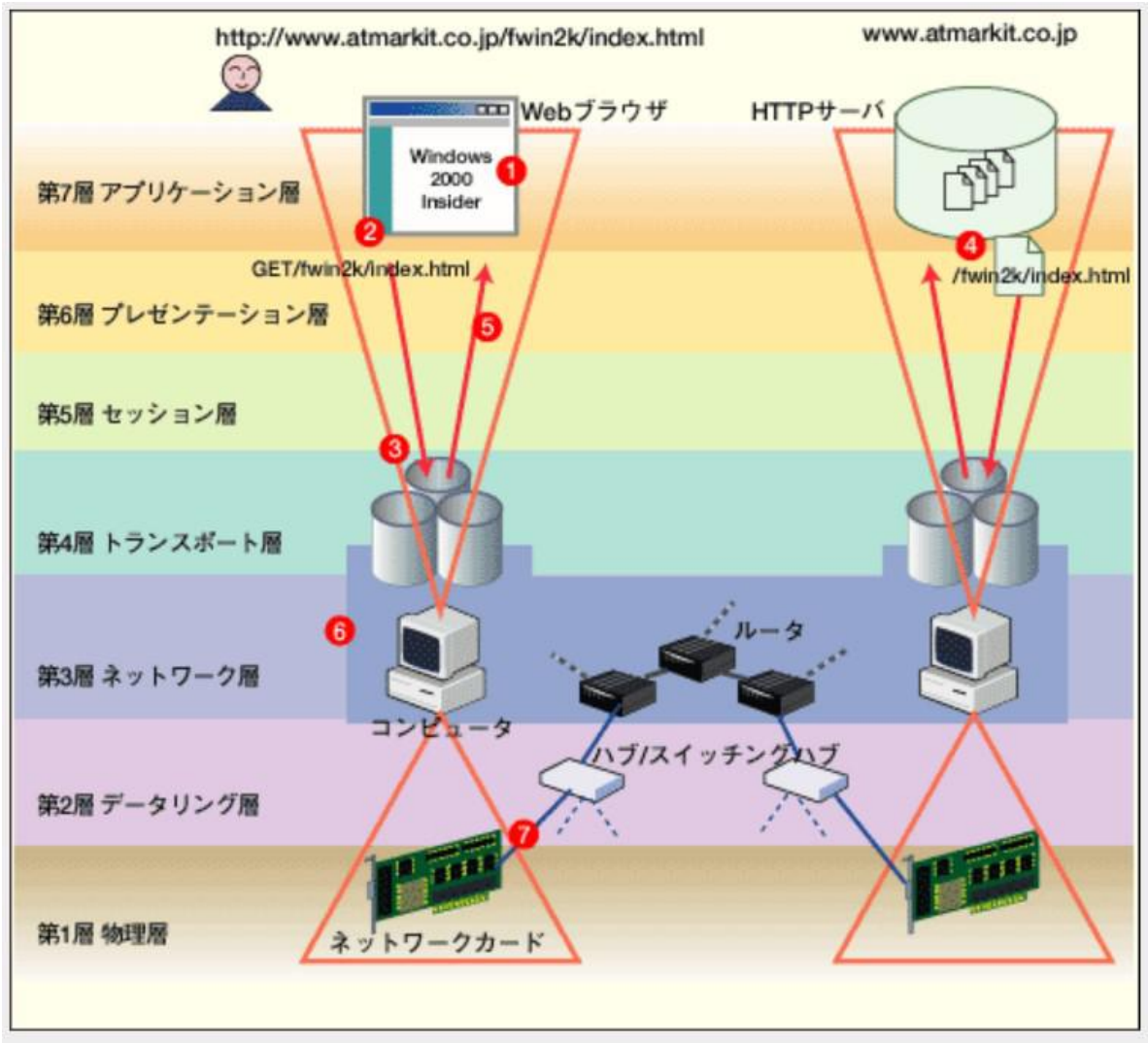
## ◆ TCP/IP階層モデルとの対応関係と、ヘッダ情報追加プロトコルの分類

TCP/IPモデルで用いられるプロトコルのうち、最も代表的な「TCP」と「IP」から名前をとって「TCP/IP」と名付けられた。



# 02-02. 【OSI】通信機器におけるヘッダ情報認識

## ◆ 各概念層の実際の通信機器の対応関係

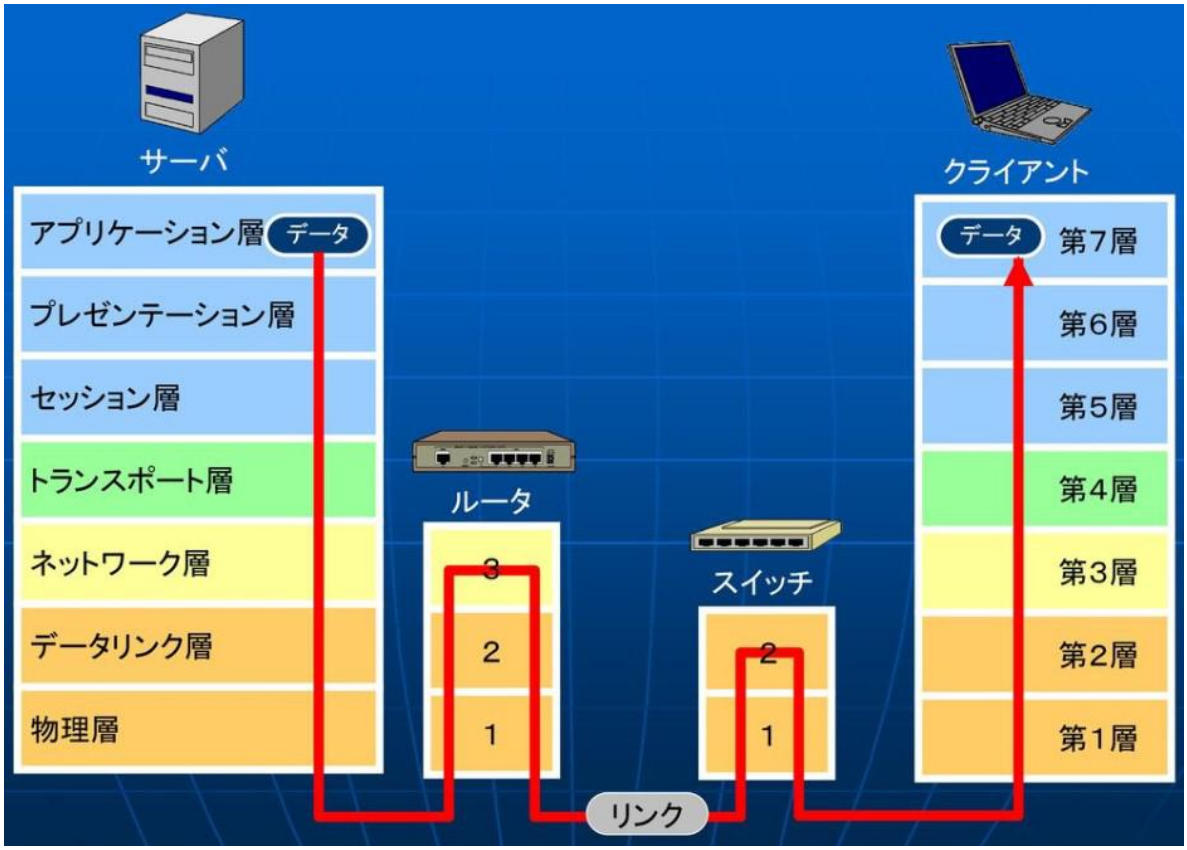



- **ネットワーク層**
- **データリンク層**
- **物理層**

Network Interface Card（（例）LANアダプタ、LANボード、LANカード）、リピータ、LANケーブル

## ◆ 通信機器における各層のヘッダ情報の認識

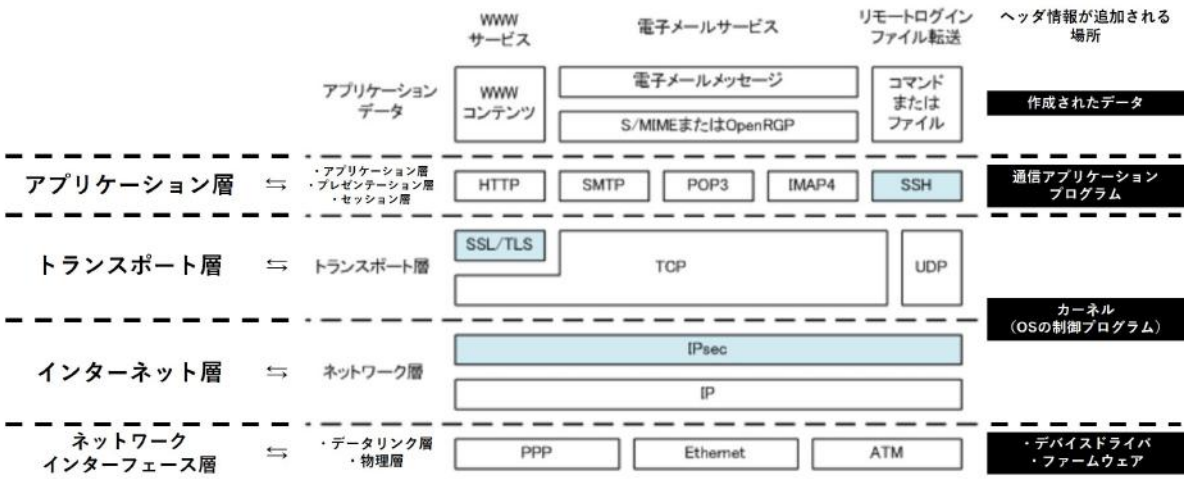
送信元で作成されたパケットは、非カプセル化されながら、通信機器に認識される。




# 02-03. 【TCP/IP】アプリケーション層の『HTTP』によるヘッダ情報追加

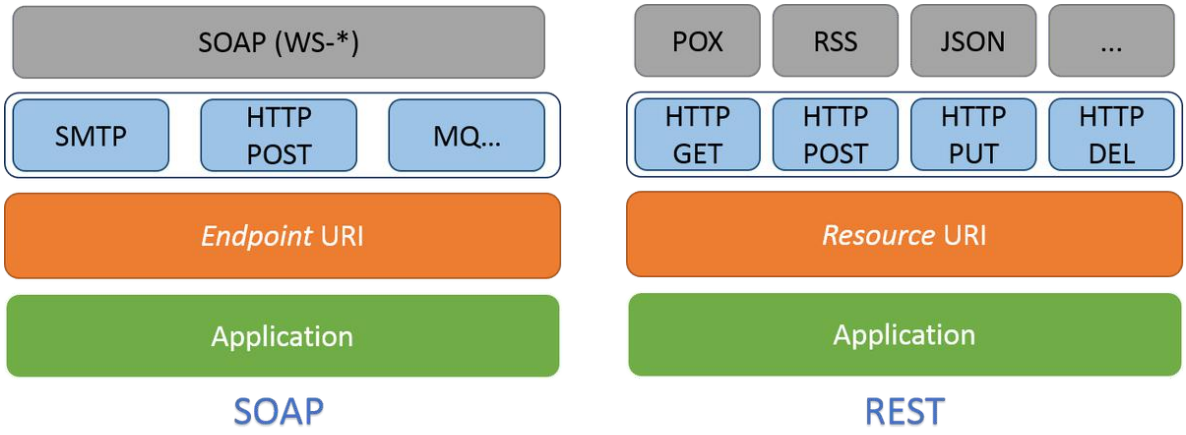
## ◆ ヘッダ情報追加プロトコルの分類と追加される場所（再掲）

パケット交換方式におけるパケットのヘッダ情報は、パソコンの各概念層のプロトコルによって追加されていく。




## ◆ WebAPIの種類

HTTPの構造は、WebAPIの種類によって決まる。WebAPIには、『SOAP』と『REST』がある。

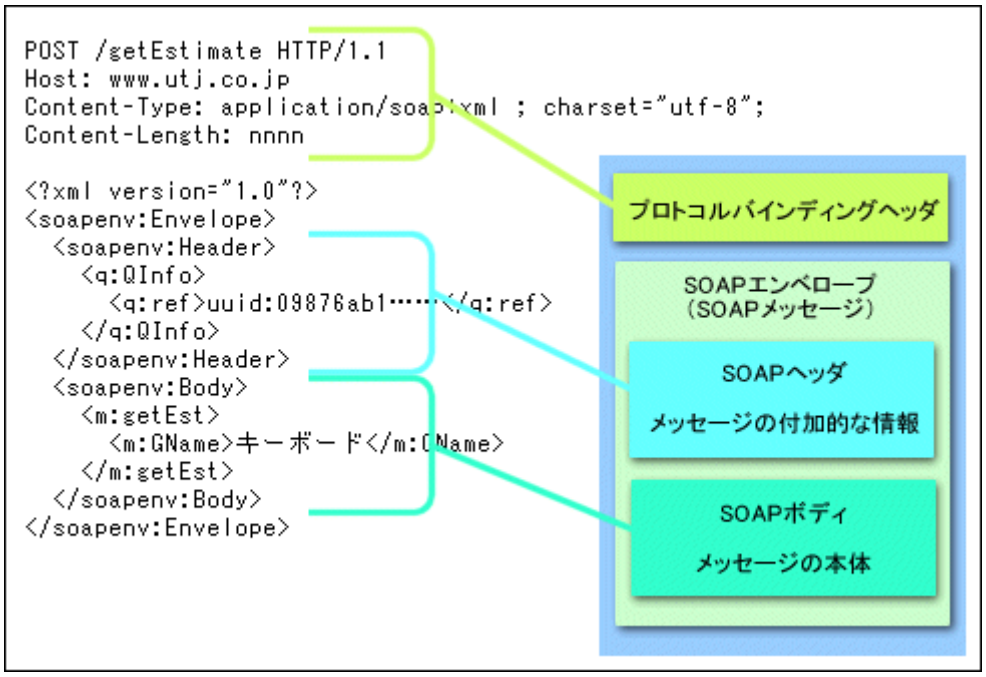



## ◆ SOAPにおけるメソッド

1999年にMicrosoft社などによって発表された。2000年前半には企業システムのWebサービス化とともに普及するかに思われたが、複雑な構造であったため次第に敬遠されるようになった。その後、簡単な構造のREST APIが好まれるようになった。

**【具体例】**

POSTメソッドによるリクエスト

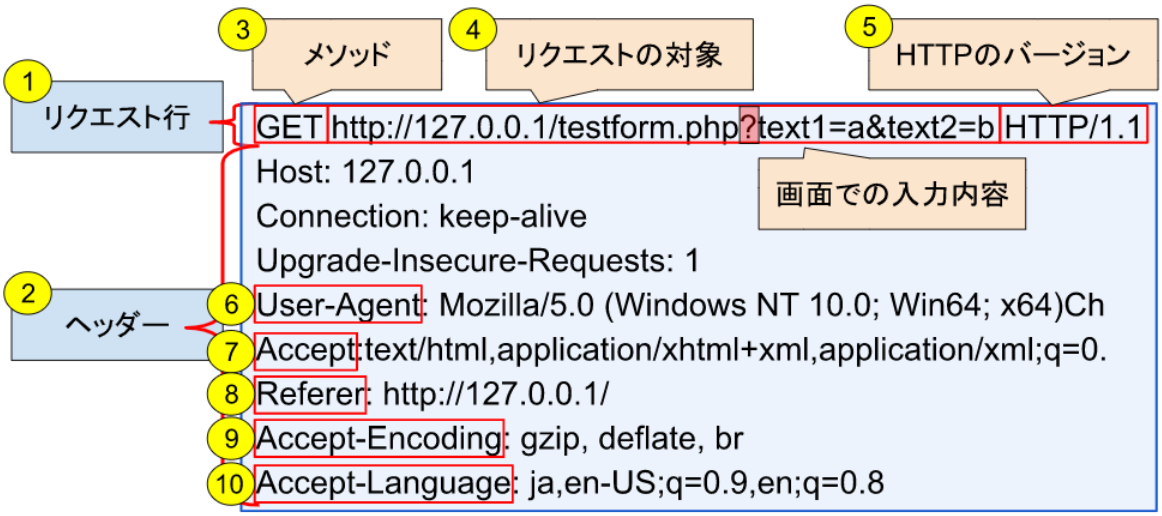



## ◆ RESTにおけるメソッド

| メソッド | 意味 |
|---|---|
| GET | リソースの取得 |
| POST | 子リソースの作成、リソースへのデータ追加、その他処理 |
| PUT | リソースの更新、リソースの作成 |
| DELETE | リソースの削除 |
| HEAD | リソースのヘッダ(メタデータの取得) |
| OPTIONS | リソースがサポートしているメソッドの取得 |
| TRACE | プロキシ動作の確認 |
| CONNECT | プロキシ動作のトンネル接続への変更 |

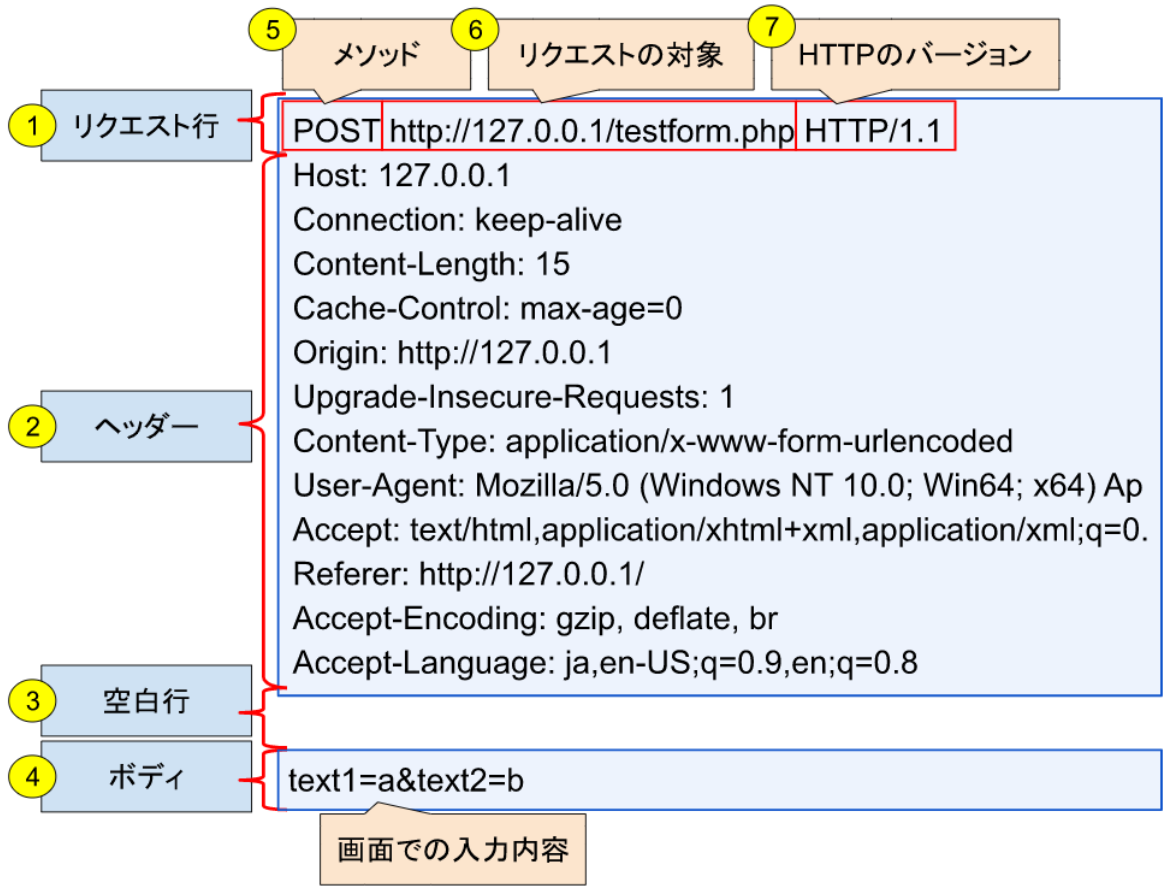
- **GETメソッドによるリクエスト情報**
  URL以降に、『/？～』付加してHTTPリクエストを送る方法。URLに情報が記述されるため、履歴で確認できてしまう。リクエスト情報は、以下の要素に分類できる。

1. リクエスト行
2. リクエストヘッダー：リクエストの詳細情報
3. メソッド：GET
4. リクエストの対象：ファイルへのパス
5. HTTPプロトコルのバージョン
6. User-Agent：ブラウザのバージョン情報等
7. Accept：ブラウザが想定する(利用可能な)MIMEのタイプ
8. Referer：遷移元のページ
9. Accept-Encoding：ブラウザがデコードできるエンコーディング形式
10. Accept-Language：ブラウザが想定する(利用可能な)言語




- **POSTメソッドによるリクエスト情報**
  メッセージボディに記述してHTTPリクエストを送る方法。メッセージボディに情報が記述されるため、履歴では確認できない。リクエスト情報は、以下の要素に分類できる。

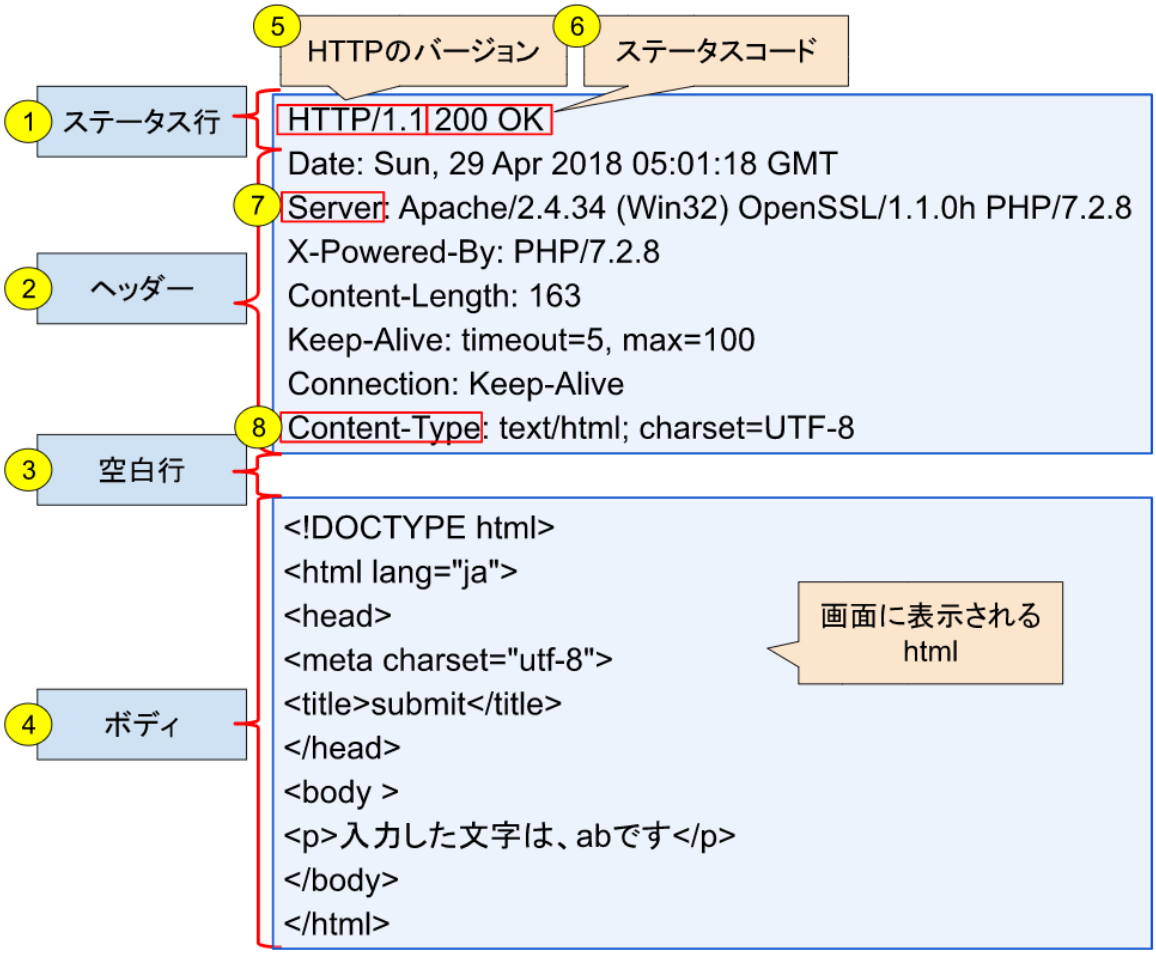
1. リクエスト行
2. リクエストヘッダー：リクエストの詳細情報
3. 空白行：ヘッダーとボディを分ける
4. ボディ：画面での入力内容が入っている
5. メソッド：POST
6. リクエストの対象：ファイルへのパス
7. HTTPプロトコルのバージョン




- **レスポンス情報**

リクエスト情報は、以下の要素に分類できる。

1. HTTPステータス
2. レスポンスヘッダー：レスポンスの詳細情報がある
3. 空白行：ヘッダーとボディを分ける
4. ボディ：HTMLや画像等が入る
5. HTTPプロトコルのバージョン
6. ステータスコード：200はサーバーのWebシステム処理が成功したことを表す
7. Server：サーバーの名前とバージョン等です。
8. Content-Type：出力するMIMEタイプ



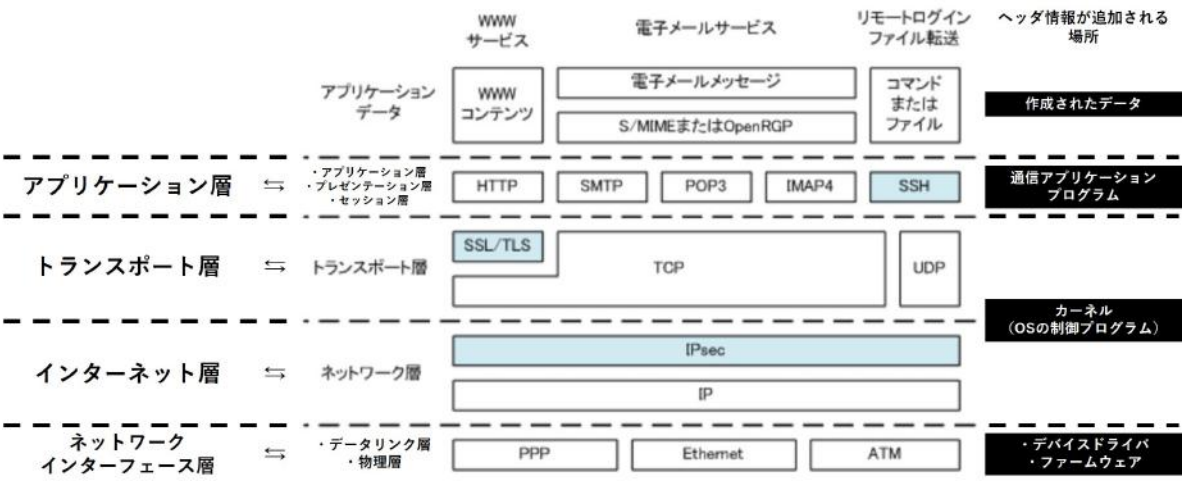

### ◆ HTTPステータスの種類

- **100番台：継続**
- **200番台：リクエスト成功**
- **300番台：リダイレクトに関するステータス**
- **400番台：リクエスト失敗**
- **500番台：サーバーエラー**

# 02-04. 【TCP/IP】アプリケーション層の『SMTP』『POP3』『IMAP』によるヘッダ情報追加

### ◆ ヘッダ情報追加プロトコルの分類と追加される場所（再掲）

パケット交換方式におけるパケットのヘッダ情報は、パソコンの各概念層のプロトコルによって追加されていく。




## ◆ メール送信用プロトコル

- **SMTP AUTH：Simple Mail Transfer Protocol AUTHentication**

  メール送信にあたってユーザ認証の仕組みがないSMTPを拡張し、ユーザ認証機能を追加した仕様。

## ◆ メール受信用プロトコル

- **POP3：Post Official Protocol version 3**

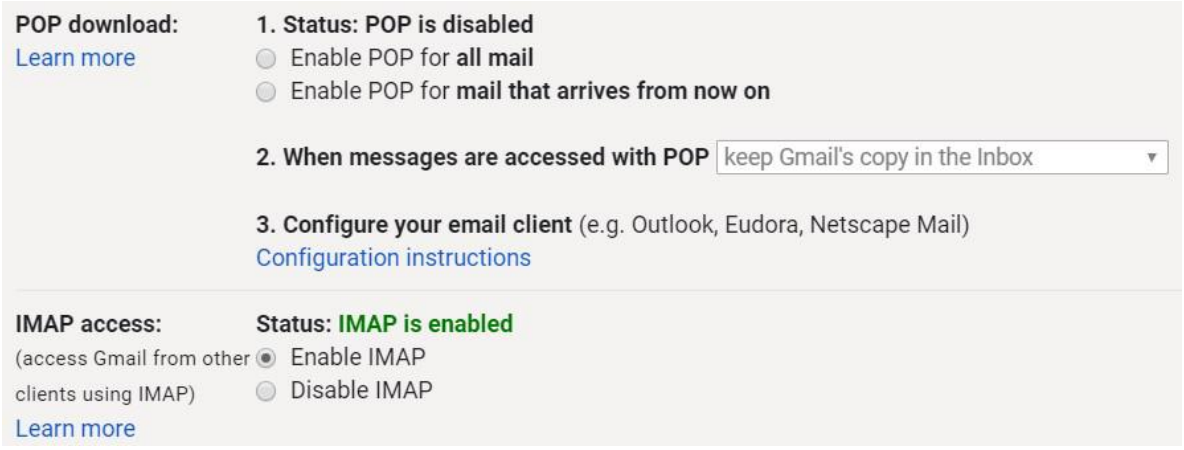
  メールサーバに届いたメールを、受信機器にダウンロードし、受信機器で閲覧するプロトコル。メールの既読未読状況は、他の受信機器と共有される。

- **IMAP4：Internet Message Access Protocol version 3**

  メールサーバに届いたメールを、受信機器にダウンロードせず、メールサーバに置いたまま閲覧するプロトコル。メールの既読未読状況は、他の受信機器と共有されない。

  **【具体例】**

  GmailでPOPかIMAPを設定可能



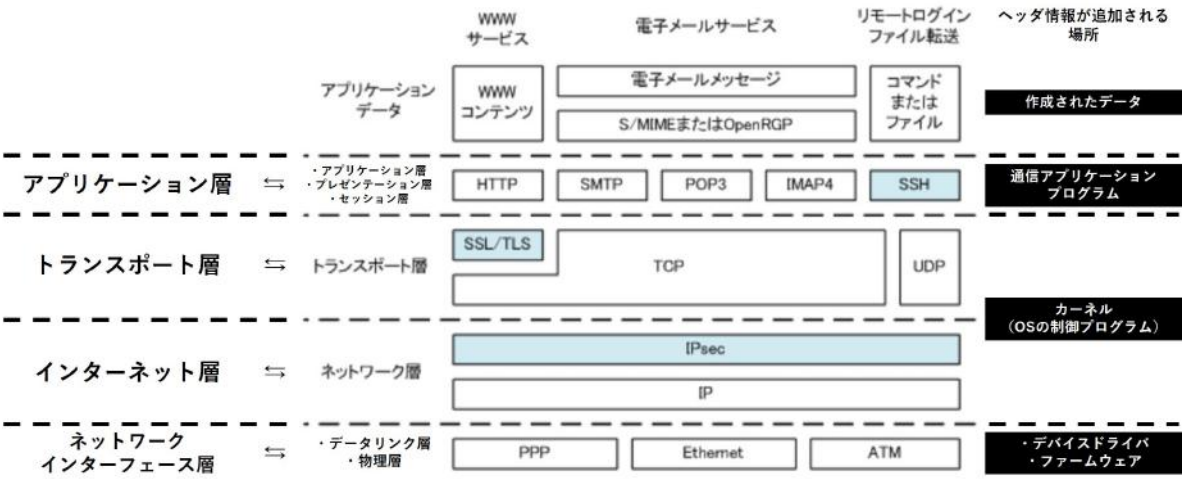

- **APOP：Authenticated POP**

  メール受信の際に、チャレンジレスポンス方式の認証を行うことで平文の認証情報がネットワークに流れるのを防止するプロトコル

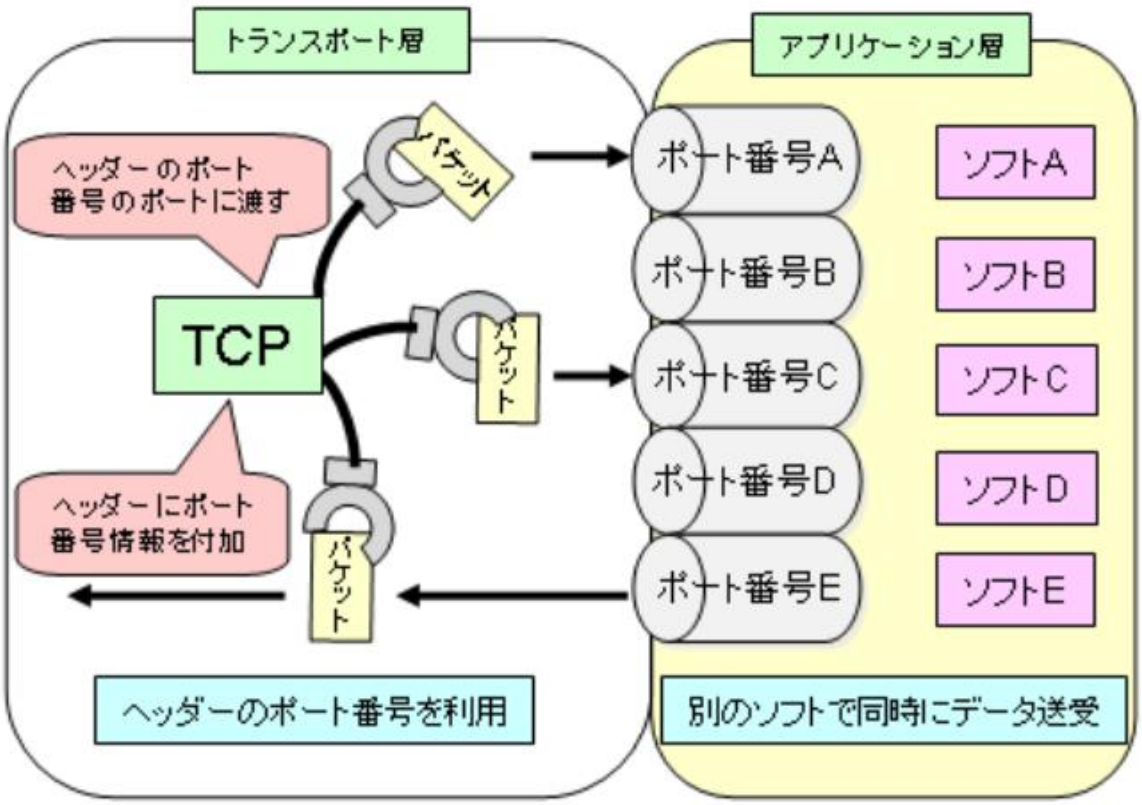
# 02-05. 【TCP/IP】トランスポート層『TCP』によるヘッダ情報追加

## ◆ ヘッダ情報追加プロトコルの分類と追加される場所（再掲）

パケット交換方式におけるパケットのヘッダ情報は、パソコンの各概念層のプロトコルによって追加されていく。




## ◆ レスポンスにおけるポート番号の識別の仕組み

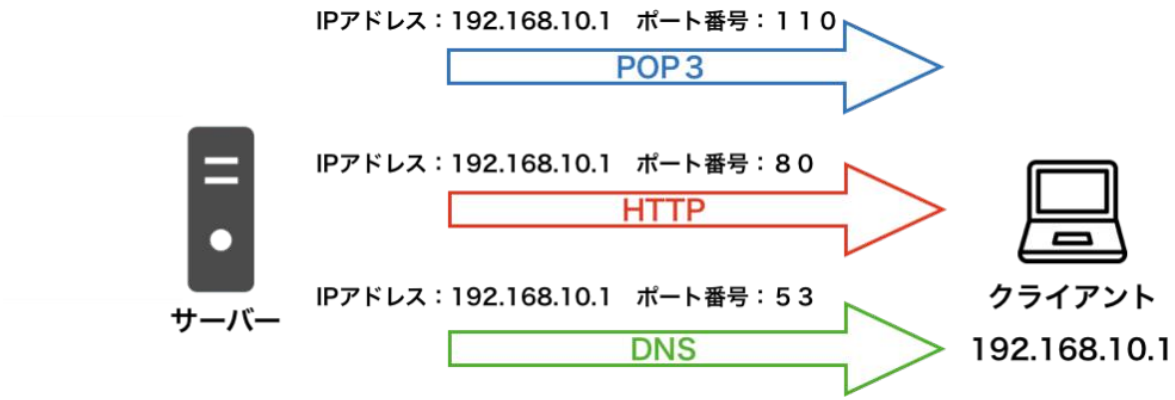



まず、プライベートIPアドレスを用いて、レスポンス先のパソコンを識別する。その後、リクエスト時のポート番号を元にして、特定のアプリにレスポンスする。

**【具体例】**

Webページを見ながらメールアプリを起動している場合...

1. プロトコル『POP3』とポート番号『110』を指定して、メールアプリにレスポンス。
2. プロトコル『HTTP』とポート番号『80』を指定して、Webフロントにレスポンス。
3. プロトコル『DNS』とポート番号『53』を指定して、Webフロントにレスポンス。




## ◆ ポートスキャナ

ポートスキャナを用いることによって、各ポートにアクセスし、応答があるかどうかや、どのようなソフトウェアが応答するかを調べ、一覧表示することができる。

## ◆ ポート番号の種類

アプリには、種類ごとにポート番号が割り当てられている。

- **Well known ポート番号（0 ～ 1023）**

  IANA：Internet Assigned Numbers Authority（インターネット割当番号公社）によって管理されているポート番号。Webサーバがリクエストを受信する時、またレスポンスを送信する時に使用される。

| ポート番号 | アプリケーションプロトコル | トランスポートプロトコル |
|---|---|---|
| 20 | FTP Data | TCP |
| 21 | FTP | TCP |
| 22 | SSH | TCP |
| 23 | telnet | TCP |
| 25 | SMTP | TCP |
| 53 | DNS | UDP/TCP |
| 67 | DHCP（Bootstrap Protocol Server） | UDP |
| 68 | DHCP（Bootstrap Protocol Client） | UDP |
| 69 | TFTP | UDP |
| 80 | HTTP | TCP |
| 110 | POP3 | TCP |
| 123 | NTP | UDP |
| 161 | SNMP | UDP |
| 162 | SNMP（TRAP） | UDP |
| 443 | HTTPS | TCP |
| 520 | RIP | UDP |

- **登録済みポート番号（1024 ～ 49151）**

  IANAが登録申請を受けて公開しているポート番号。企業が作成した独自のアプリなどに対して割り当てられる。クライアントがリクエストを送信する時、またレスポンスを受信する時に使用される。

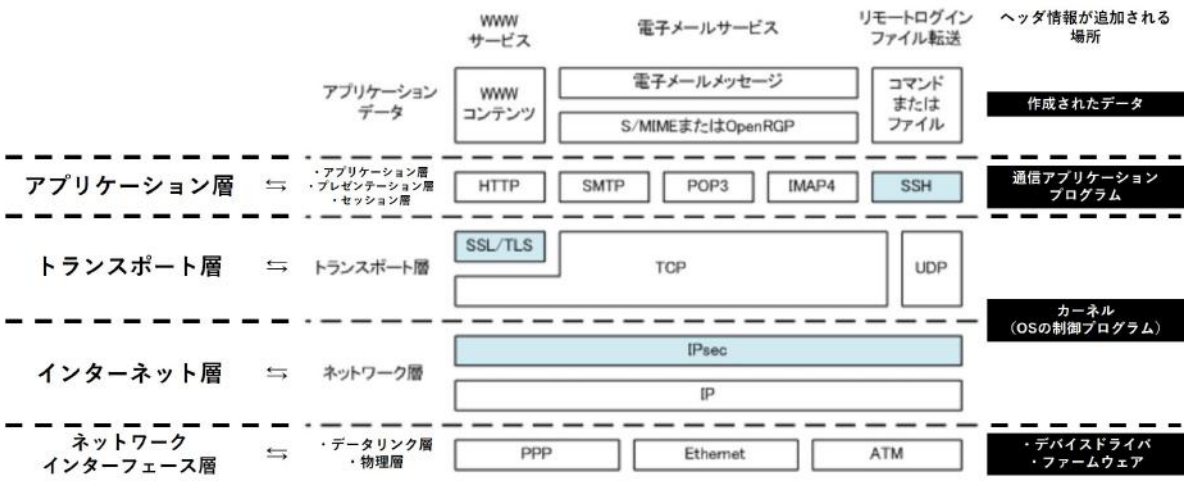
- **動的／非公式ポート番号（49152 ～ 65535）**

  自由に使用できるポート番号。クライアントがリクエストを送信する時、またレスポンスを受信する時に使用される。

# 02-06. 【TCP/IP】インターネット層『IPv4』によるヘッダ情報追加

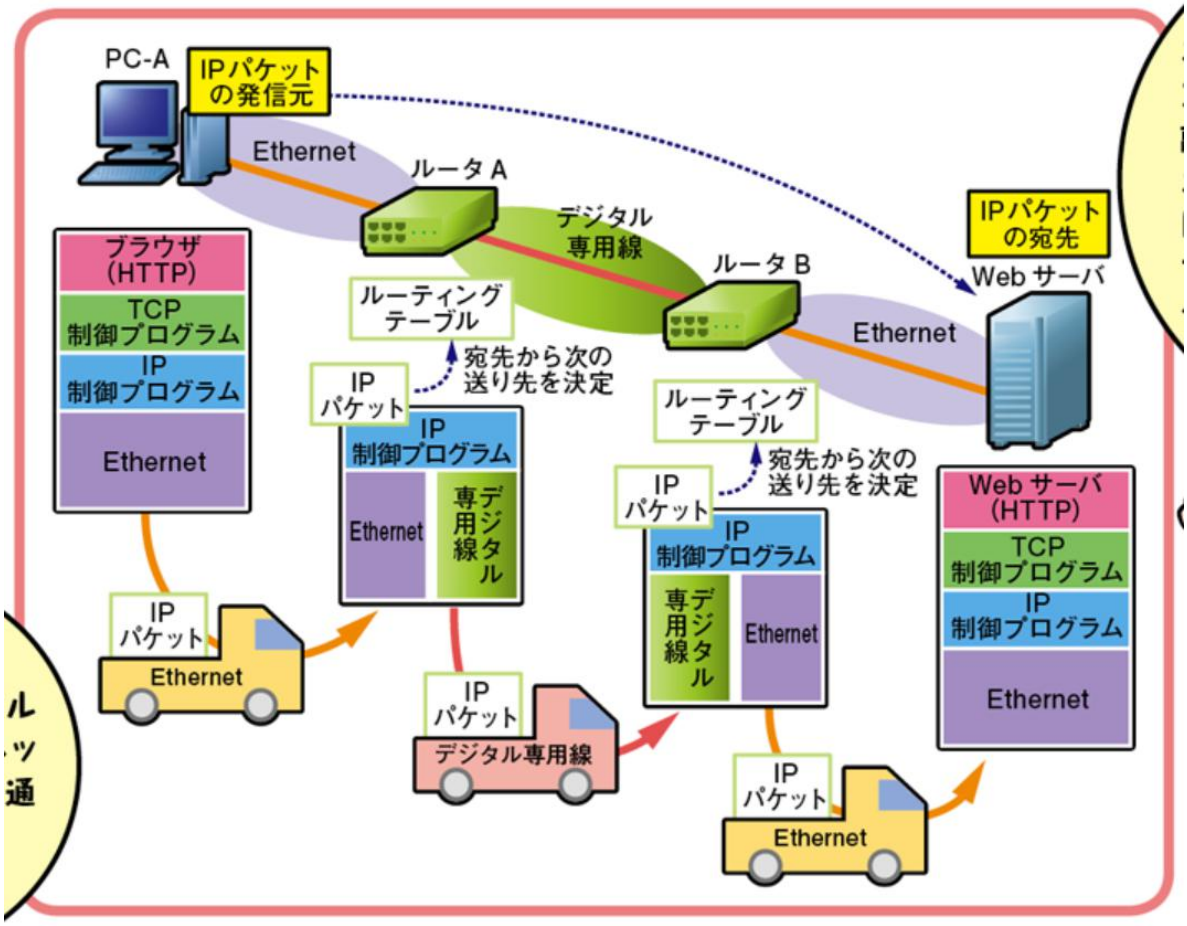
## ◆ ヘッダ情報追加プロトコルの分類と追加される場所（再掲）

パケット交換方式におけるパケットのヘッダ情報は、パソコンの各概念層のプロトコルによって追加されていく。




## ◆ IPパケットのヘッダ情報を用いた宛先認識

1. PC-Aは、構成したIPパケットをEthernetに乗せて、ルータAに送信。
2. ルータAは、IPパケットをデジタル専用線に乗せて、ルータBに送信。
3. ルータBは、構成したIPパケットをEthernetに乗せて、Webサーバに送信。



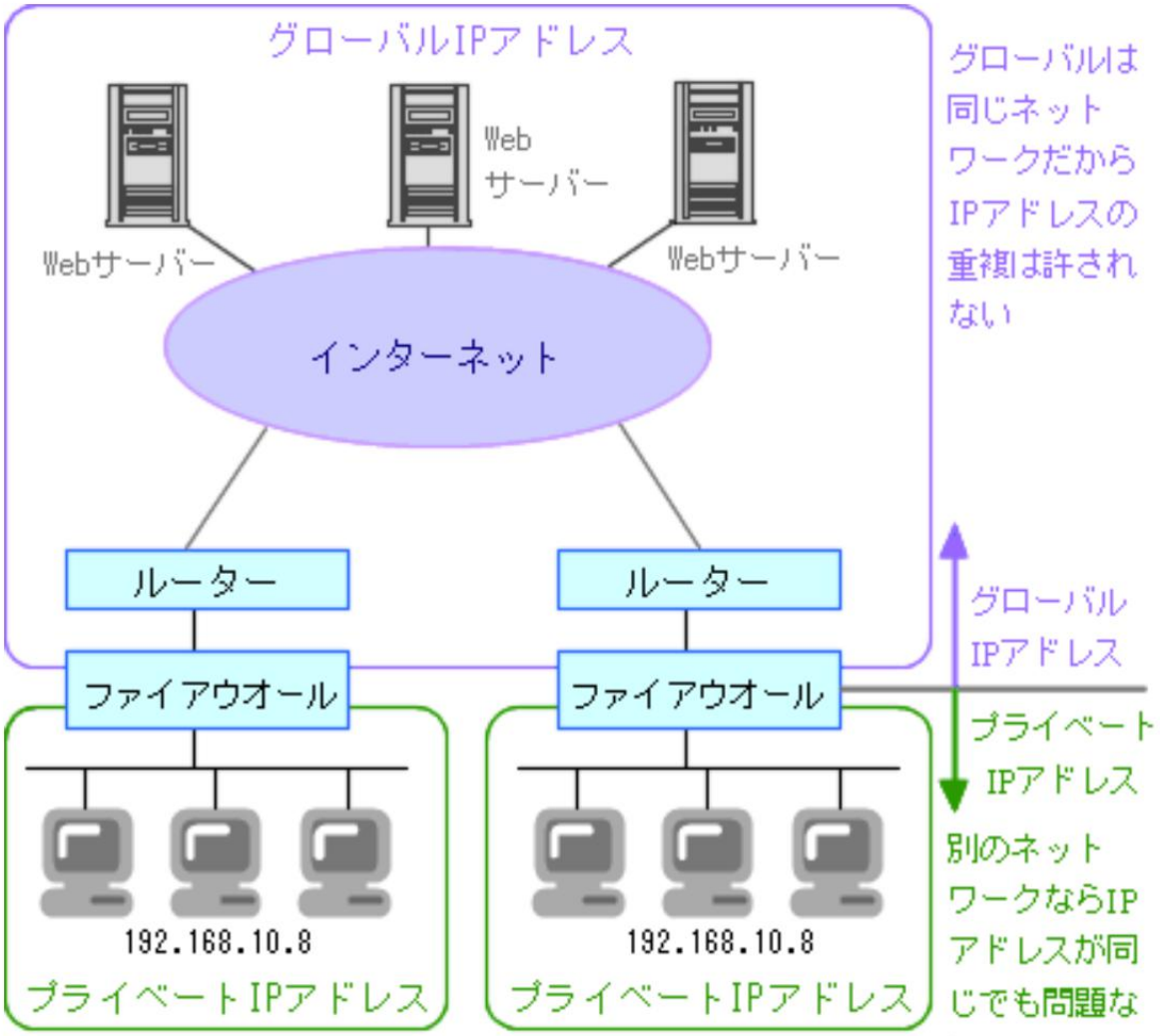

## ◆ IPv4アドレスの種類

- **プライベートIPアドレス**

  LAN内で使用される。異なるプライベートネットワーク間では、同じIPv4アドレスが存在する。
  プライベートｆは、『10.0.0.0 ～ 10.255.255.255』、『172.16.0.0 ～ 172.31.255.255』、
  『192.168.0.0 ～ 192.168.255.255』で表される。

- **グローバルIPアドレス**

  プロバイダが提供するIPv4アドレス。パブリックネットワーク内に同じIPv4アドレスは存在せず、Network Information Centerへの使用申請が必要。プライベートIPアドレスの番号でなければ、グローバルIPアドレスである。ルータには、グローバルIPアドレスが割り当てられている。



プライベート・IPアドレスとグローバル・IPアドレス

## ◆ IPv4アドレスのクラス

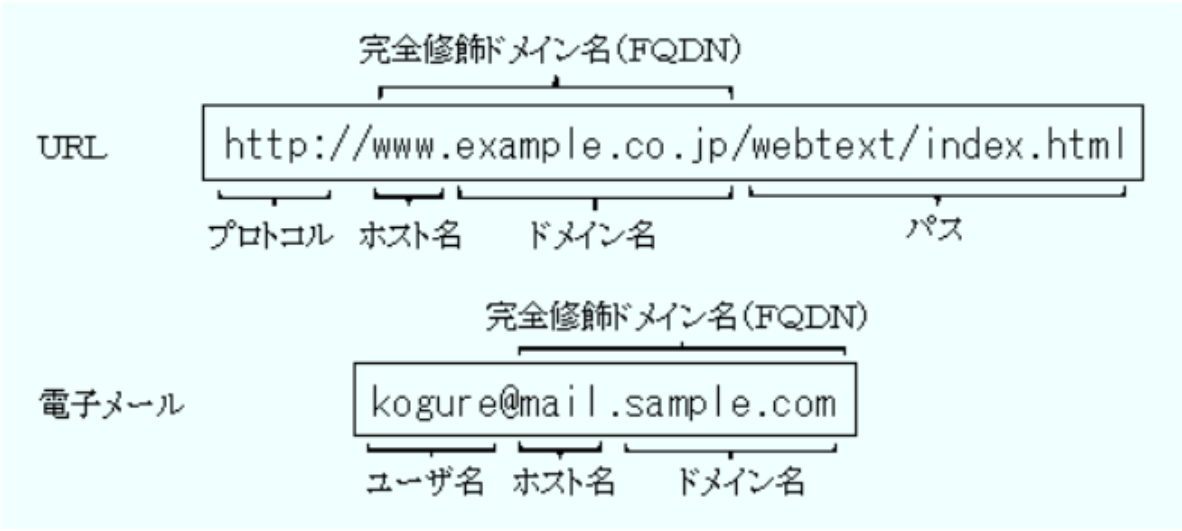
**【具体例】**

128.0.0.0は、クラス B

- **クラスA**
  上位8ビット部分が、10進表記で、0～127
- **クラスB**
  上位8ビット部分が、10進表記で、128～191
- **クラスC**
  上位8ビット部分が、10進表記で、192～223
- **クラスD**
  上位8ビット部分が、10進表記で、224～239

## ◆ URLとメールアドレスの構造

- 各部品の名称




- 各部品の様々な組み合わせ

| | | |
|---|---|---|
| http://www.example.co.jp | ホスト名 | www |
| | ドメイン | example.co.jp |
| | FQND | www.example.co.jp |
| http://news.headline.example.jp | ホスト名 | news.headline |
| | ドメイン | example.jp |
| | FQND | news.headline.example.jp |
| http://example.com | ホスト名 | - |
| | ドメイン | example.com |
| | FQND | example.com |
| http://www.日本.jp | ホスト名 | www |
| | ドメイン | 日本.jp |
| | FQND | www.日本.jp |
| tarou@example.jp | ホスト名 | - |
| | ドメイン | example.jp |
| | FQND | example.jp |
| tarou@abc.example.ne.jp | ホスト名 | abc |
| | ドメイン | example.ne.jp |
| | FQND | abc.example.ne.jp |

## ◆ 完全修飾ドメイン名とグローバルIPアドレスのマッピング

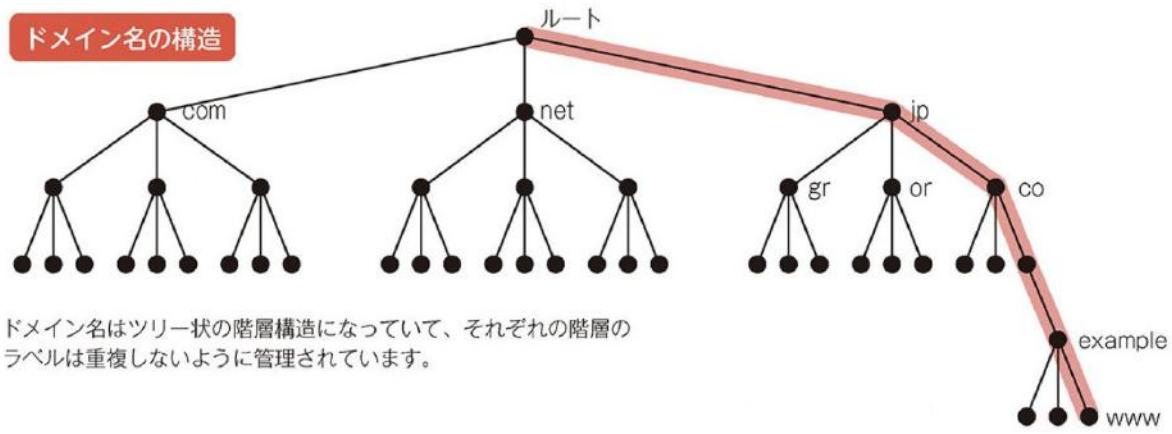
http://**www.kagoya.com**/

**||**

http://**203.142.205.139**/

例えば、外部Webサーバのグローバルipアドレスが『203.142.205.139』であると知っている場合、URLのプロトコル部分以下を『203.142.205.139』としてリクエストすれば、外部Webサーバが提供するウェブサイトにアクセスできる。しかし、グローバルipアドレスは数字の羅列であるため、人間には覚えにくい。そこで、グローバルipアドレスの代わりに、完全修飾ドメイン名をURLの一部として用いる。

- **完全修飾ドメイン名の構造**

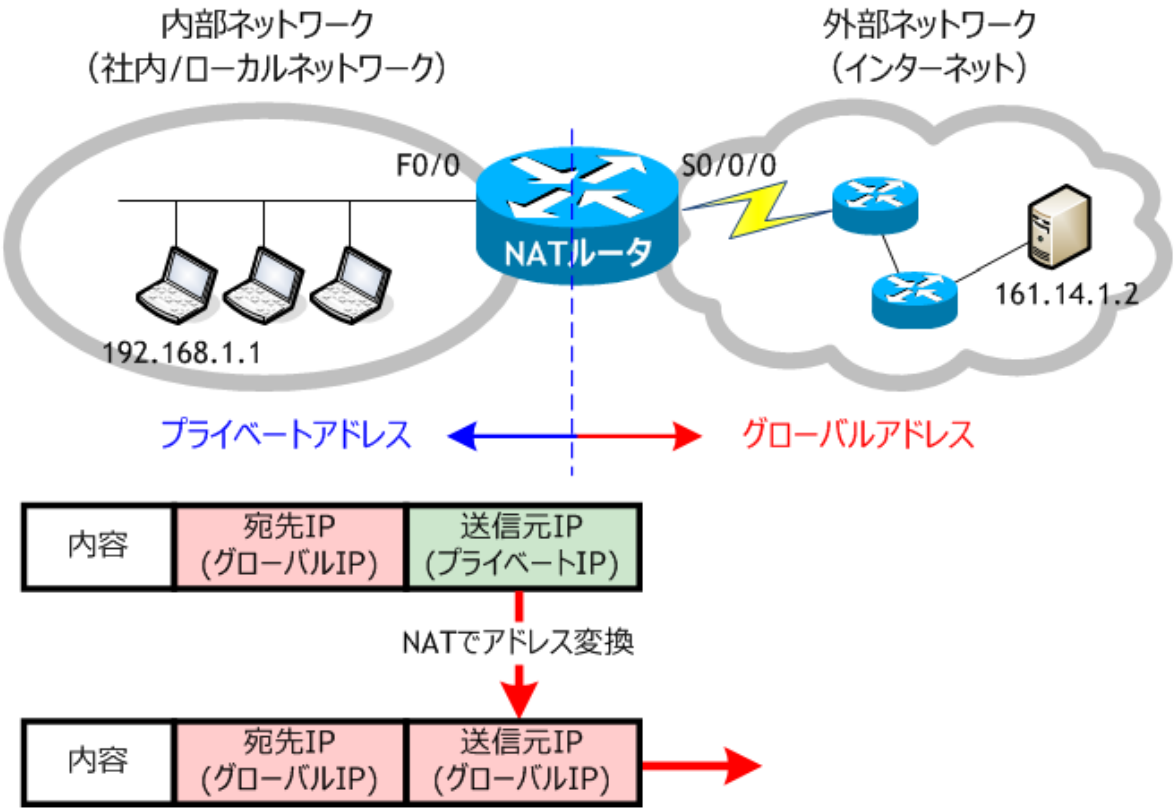



# 02-07. ルータの種類

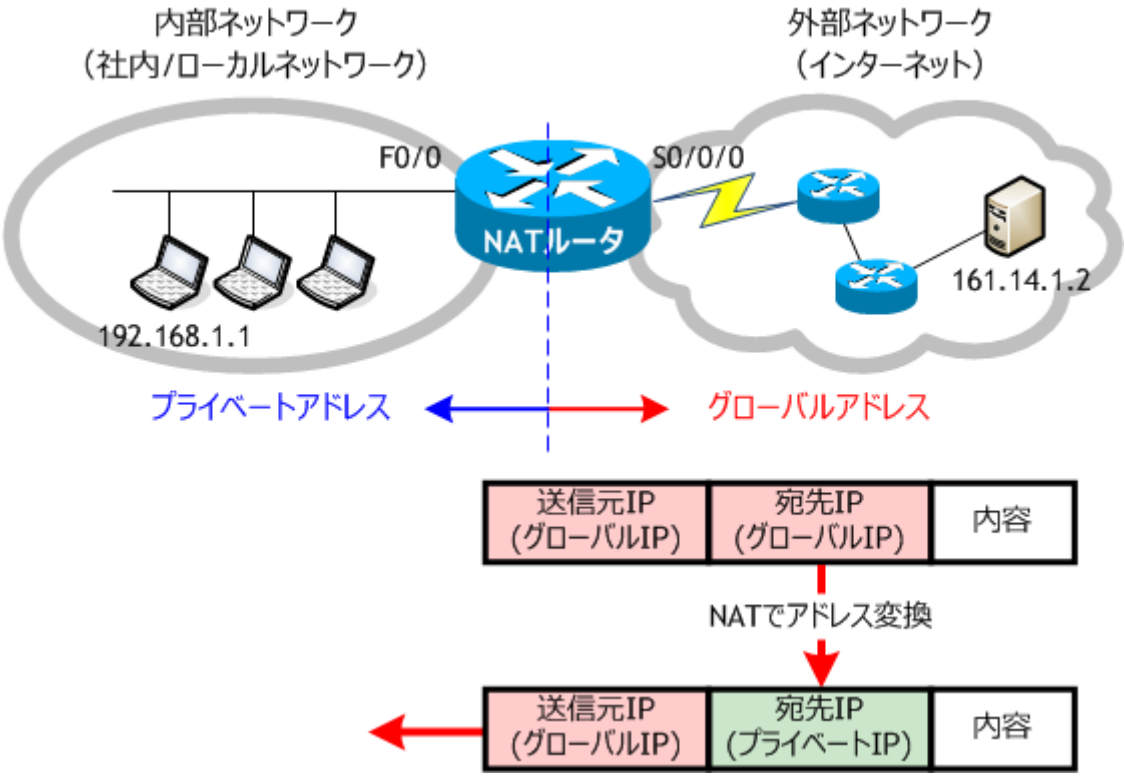
## ◆ NAT：Network Address TranslationによるIPアドレスv4の変換

パケットのヘッダ情報におけるプライベートipアドレスとグローバルipアドレスを相互変換する機能。

- **プライベートIPアドレスからグローバルIPアドレスへの変換**

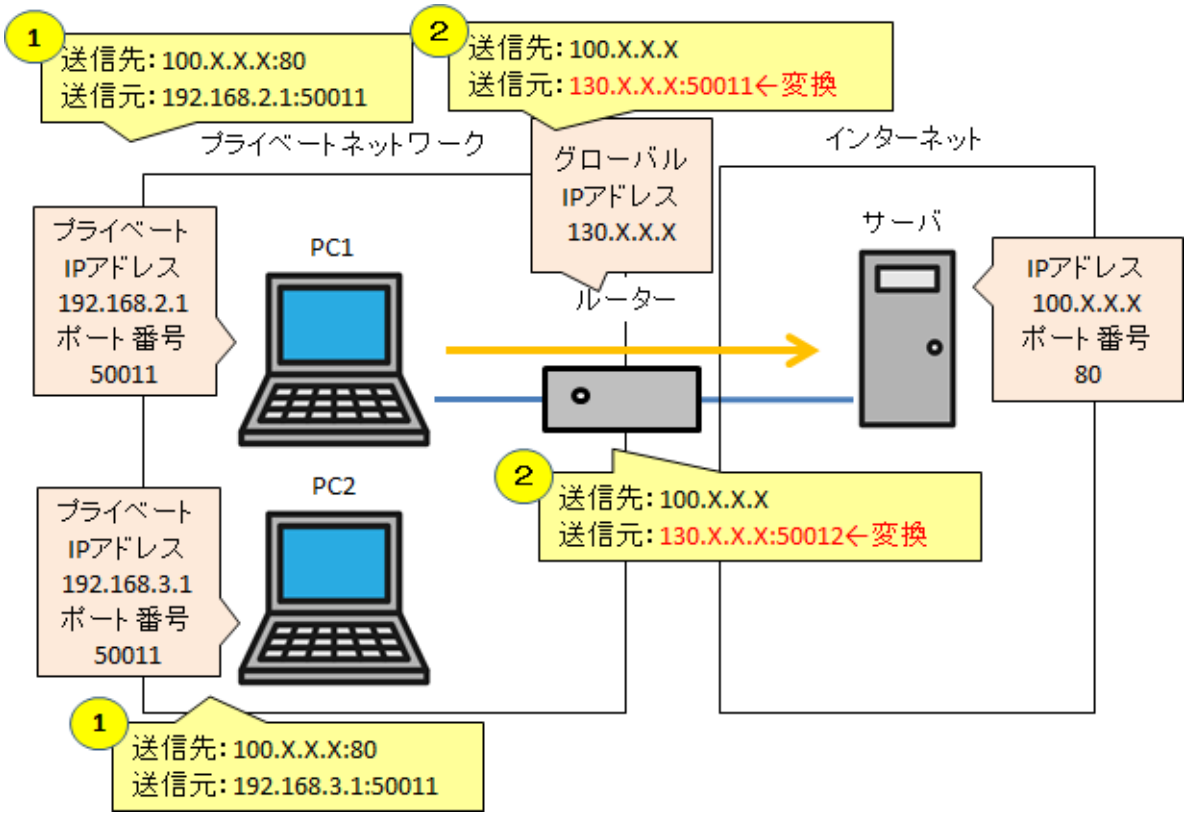



- **グローバルIPアドレスからプライベートIPアドレスへの変換**




## ◆ NAPT：Network Address Port Translationによるポート番号の変換

NATと同様にプライベートIPアドレスとグローバルIPアドレスの間を相互変換するだけでなく、ポート番号も変換する。

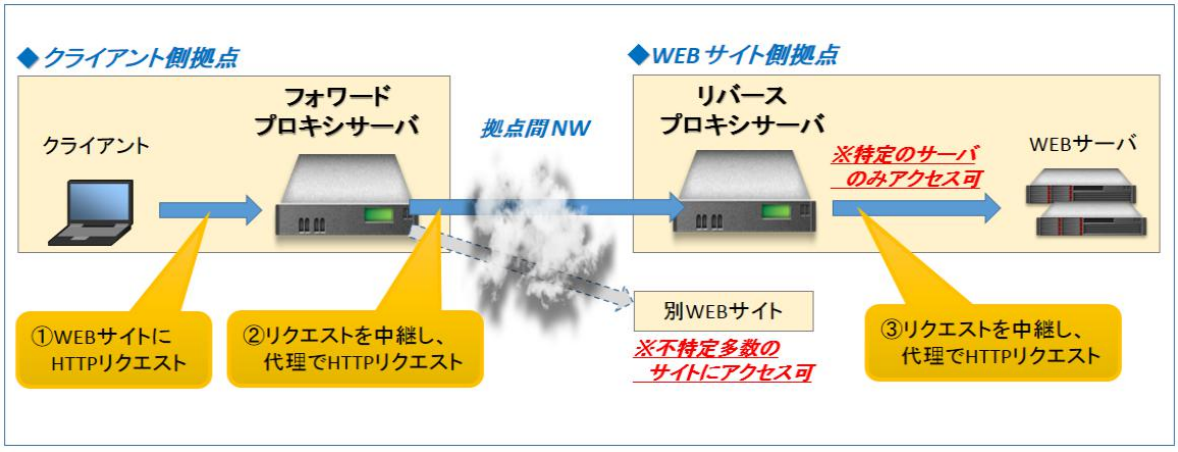



# 02-08. 名前解決の仕組み

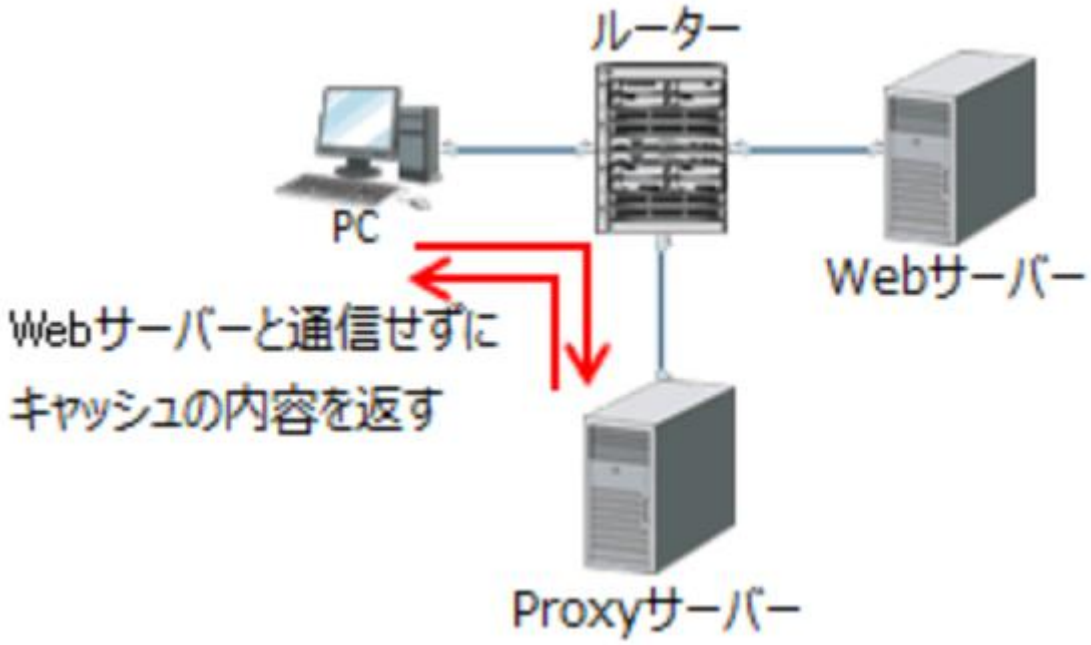
## ◆ Proxyサーバの機能

- **代理リクエスト機能（セキュリティのノートも参照）**

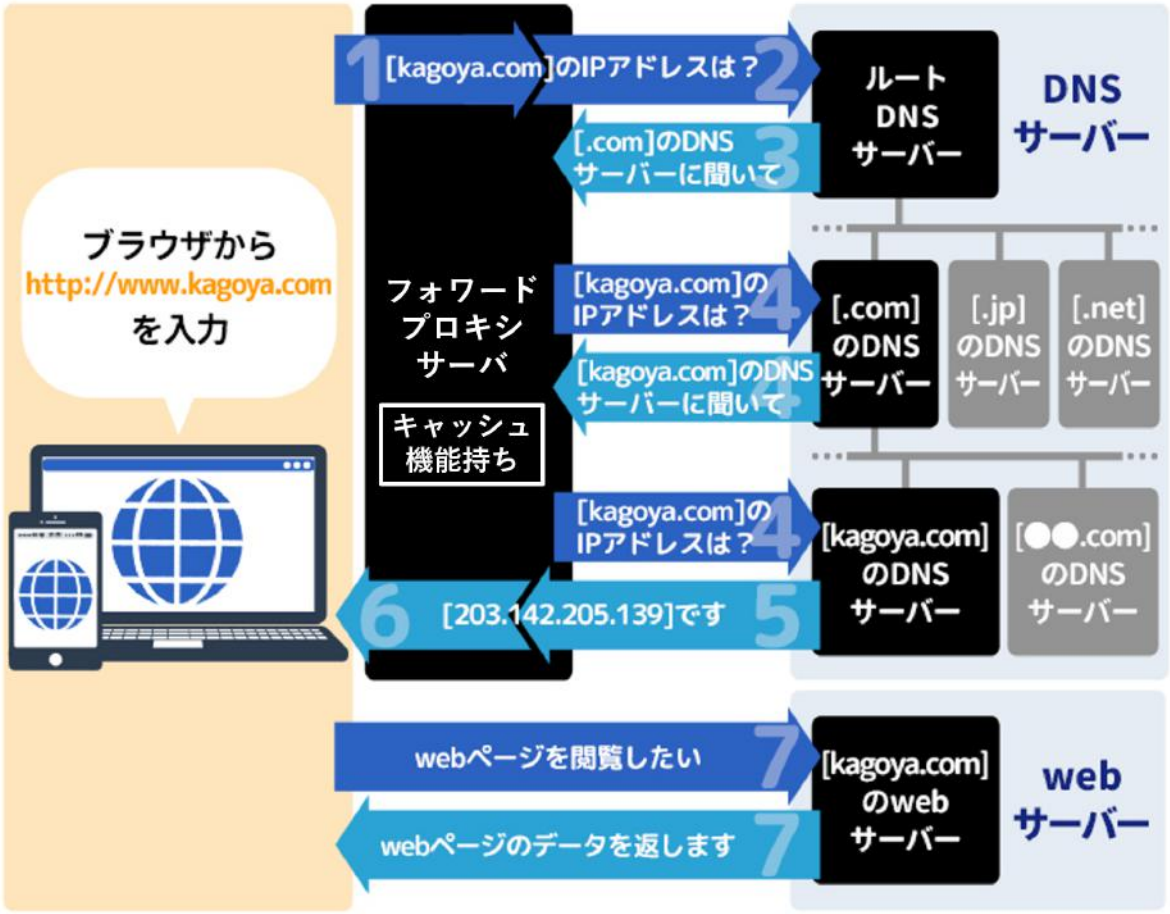
代理でリクエストを送るフォワードプロキシサーバと、レスポンスを送るプロキシサーバに分類できる。




- **キャッシュ機能**




## ◆ Proxyサーバ（FwとRe）、DNSサーバ、Webサーバによる名前解決




- **（1）完全修飾ドメイン名に対応するIPアドレスのレスポンス**

1. クライアントPCは、完全修飾ドメイン名を、フォワードProxyサーバにリクエスト。
2. フォワードProxyサーバは、完全修飾ドメイン名を、リバースProxyサーバに代理リクエスト。
3. リバースProxyサーバは、完全修飾ドメイン名を、DNSサーバに代理リクエスト。

4. DNSサーバは、完全修飾ドメインにマッピングされるIPv4アドレスを取得し、リバースProxyサーバにレスポンス。
5. リバースProxyサーバは、IPv4アドレスを、フォワードProxyサーバに代理レスポンス。
   （※NATによるIPv4アドレスのネットワーク間変換が起こる）
6. フォワードProxyサーバは、IPv4アドレスを、クライアントPCに代理レスポンス。

- **（2）IPアドレスに対応するWebページのレスポンス**

1. クライアントPCは、レスポンスされたIPv4アドレスを基に、Webページを、リバースProxyサーバにリクエスト。
2. リバースProxyサーバは、Webページを、Webサーバに代理リクエスト。
3. Webサーバは、Webページを、リバースProxyサーバにレスポンス。
4. リバースProxyサーバは、Webページを、クライアントPCに代理レスポンス。

※上記の様に、リクエストとレスポンスが繰り返されるが、データ通信は光速なので、１秒もかからない。

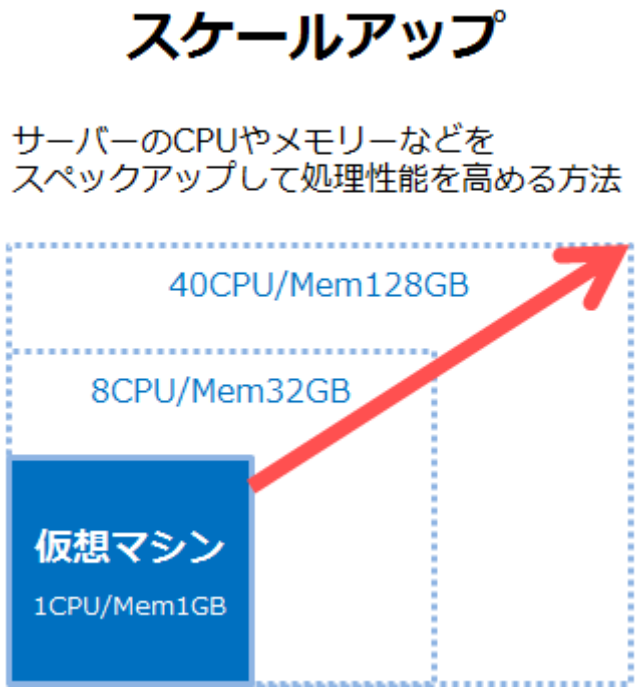
## ◆ 外部Webサーバとのデータ通信に要する時間

ネットワークのデータグローバル光回線が用いられている。光は、30万km／秒であり、一周4万kmの地球を7.5周／秒できることからもわかるように、遠く離れた国にあるWebサーバであっても、データ通信に要する時間は、１秒にも満たない。

## ◆ サーバの処理能力向上の方法

- **スケールアップ ⇔ スケールダウン**

  サーバ自体のスペックをより高くすることで、サーバ当たりの処理能力を向上させる。その逆は、スケールダウン。



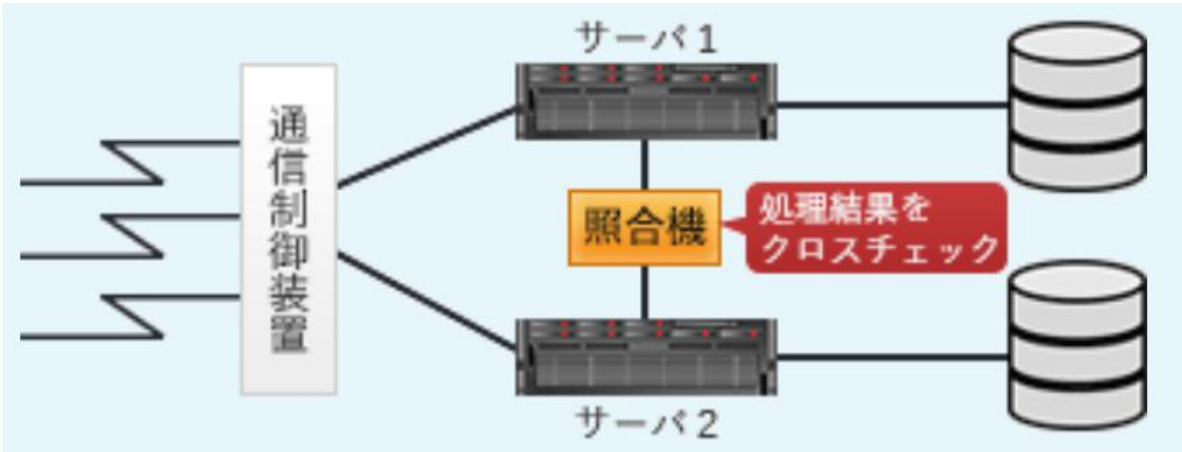

- **スケールアウト ⇔ スケールイン**

  サーバの台数を増やすことで、サーバ全体の処理能力を向上させる。その逆は、スケールイン。


# 02-09. システムの構成方法

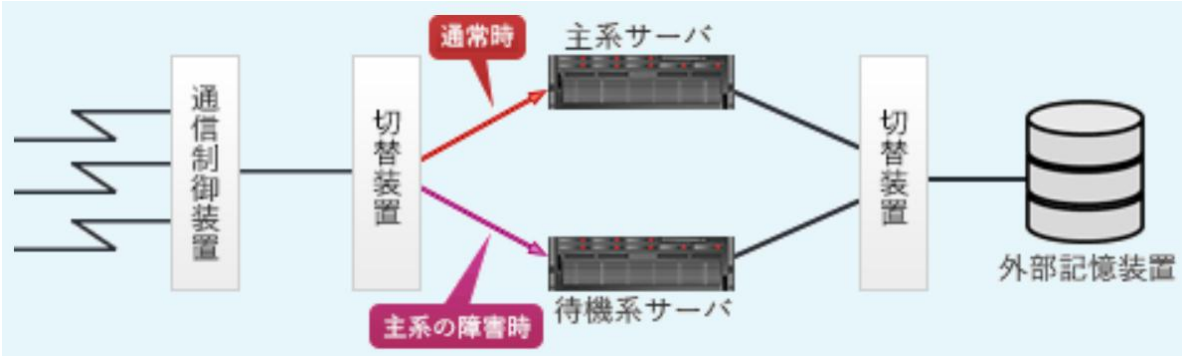
## ◆ Dualシステム

同じ処理を行う2つのシステムからなるシステム構成のこと。随時、処理結果を照合する。いずれかが故障した場合、異常が発生したシステムを切り離し、残る片方で処理を続けることによって、故障を乗り切る。



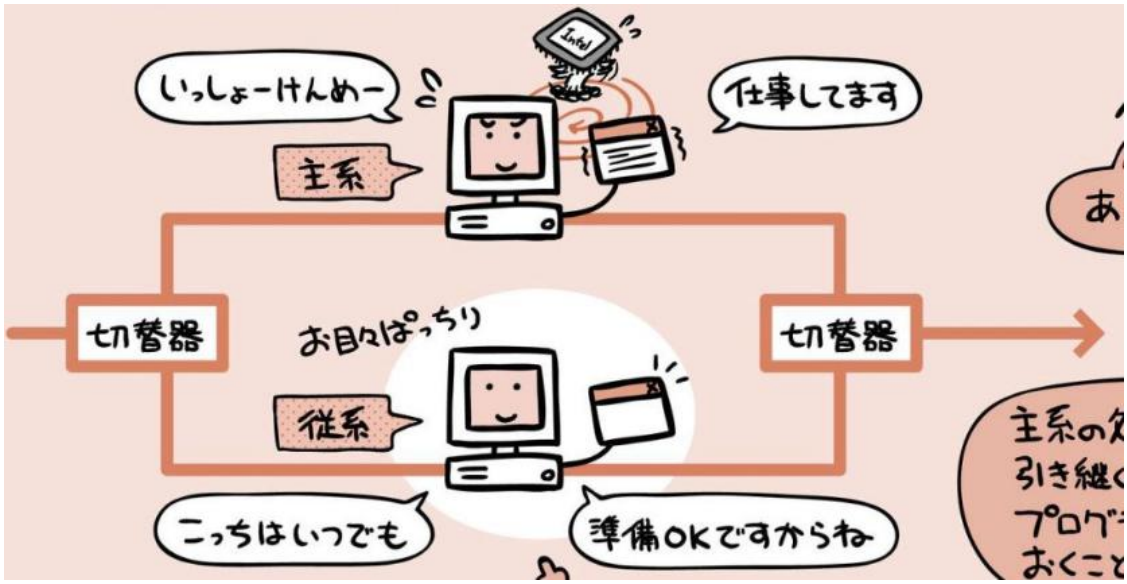

## ◆ Duplexシステム

オンライン処理を行う主系システムと、バッチ処理を行う従系システムからなるシステム構成のこと。主系システムが故障した場合、主系システムのオンライン処理を従系システムに引き継ぎ、処理を続けることによって、故障を乗り切る。

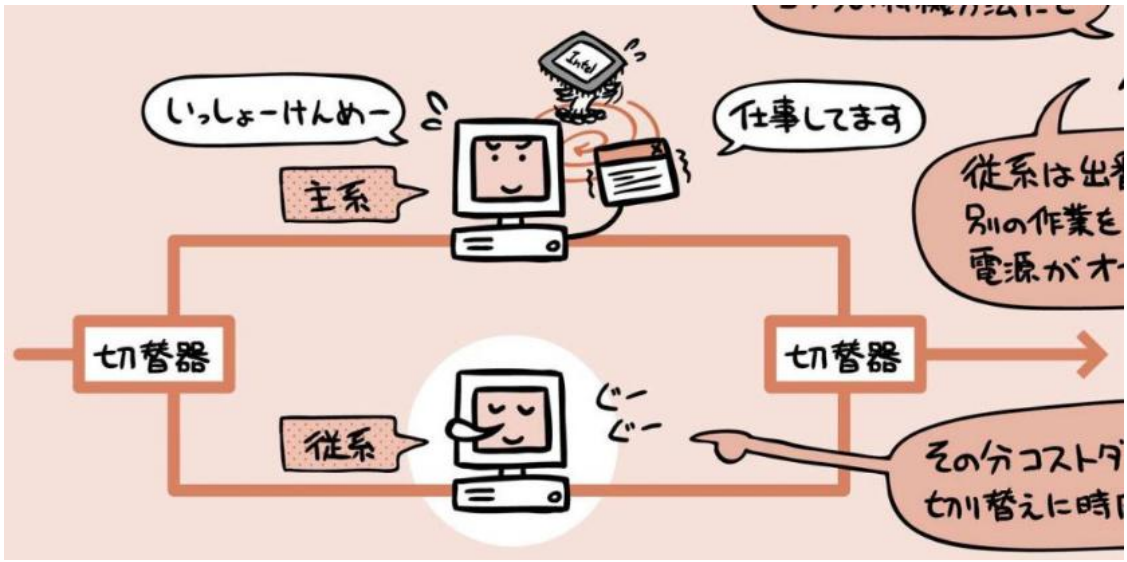



従系システムの待機方法には２つの種類がある。

- **ホットスタンバイ**




- **コールドスタンバイ**

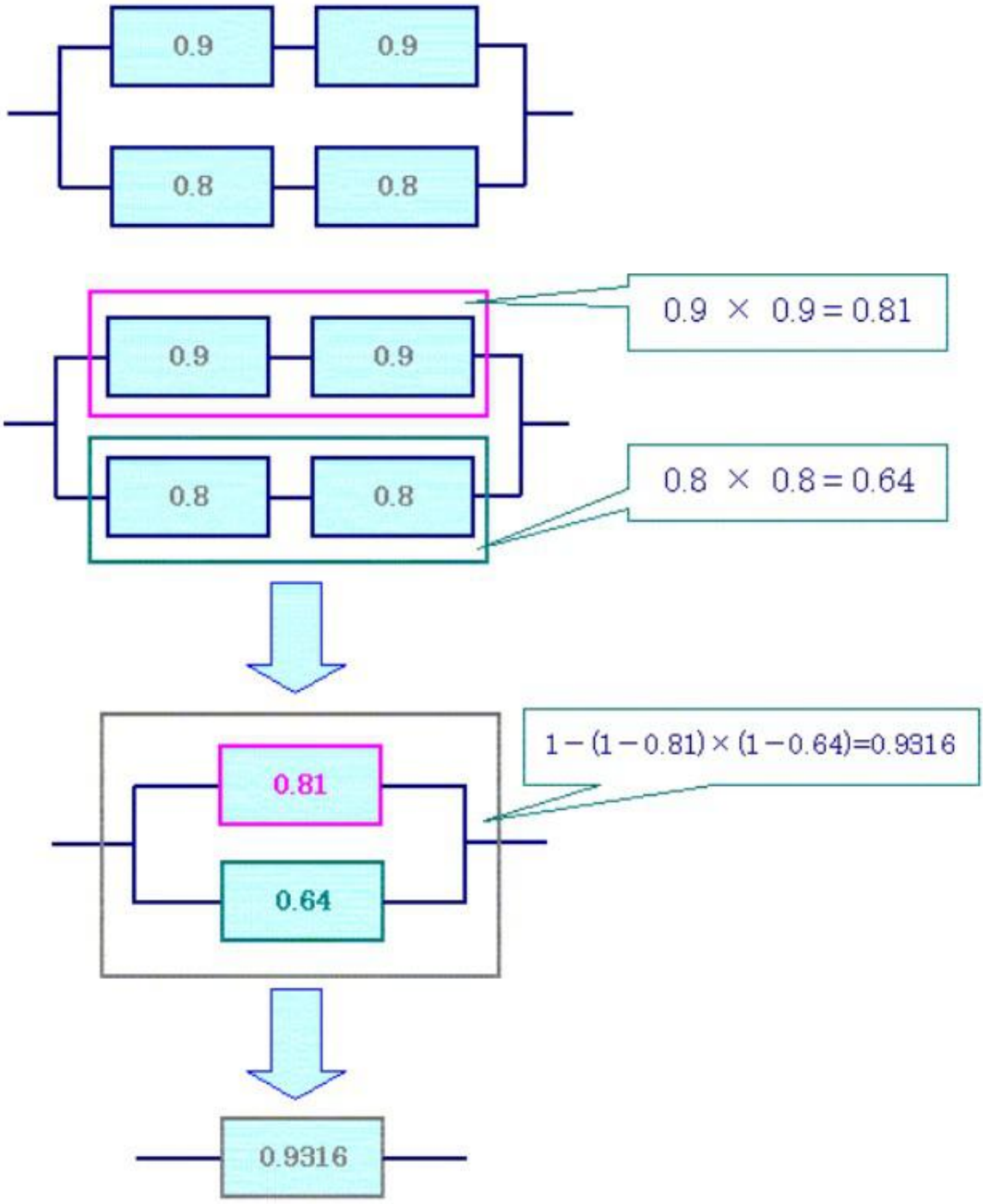



## ◆ システムの稼働率

並列システムの場合、両方の非稼働率をかけて、全体から引く。

**【具体例】**

$1 - (1 - 0.81) \times (1 - 0.64) = 0.9316$




# 03-01. サーバ仮想化とコンテナ仮想化

自身の開発環境でWebアプリを動かしたい場合、パソコン内にLinux環境のWebサーバを仮想的に構築し、そこWebアプリを設置する。そして、自身のパソコンをユーザ、また仮想Webサーバをクライアントに見立てて、SSHを用いてデータ通信を行う。仮想環境の構築方法にはいくつか種類がある。

## ◆ ホスト型仮想化

ホストOS上で、サーバを仮想的に構築する。

**【Provider例】**

VMware Workstation、Oracle VM VirtualBox


## ◆ ハイパーバイザー型仮想化

BIOSから起動したハイパーバイザー上で、サーバを仮想的に構築する（※ホストOSは用いない）。

**【Provider例】**

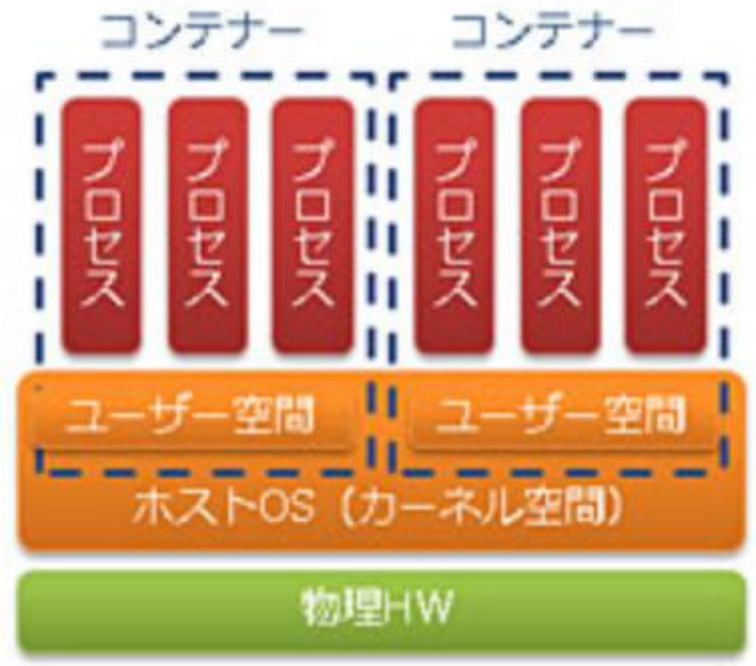
VMware vSphere Hypervisor、Xen、KVM

### ◆ コンテナ型仮想化

ホストOS上で、サーバではなく、コンテナを仮想的に構築する。カーネルのリソースを分割できるNamespace（PID namespace、Network namespace、UID namespace）とControl Groupsを用いて、単一のOS上に独立したコンテナを構築する。

**【Provider例】**

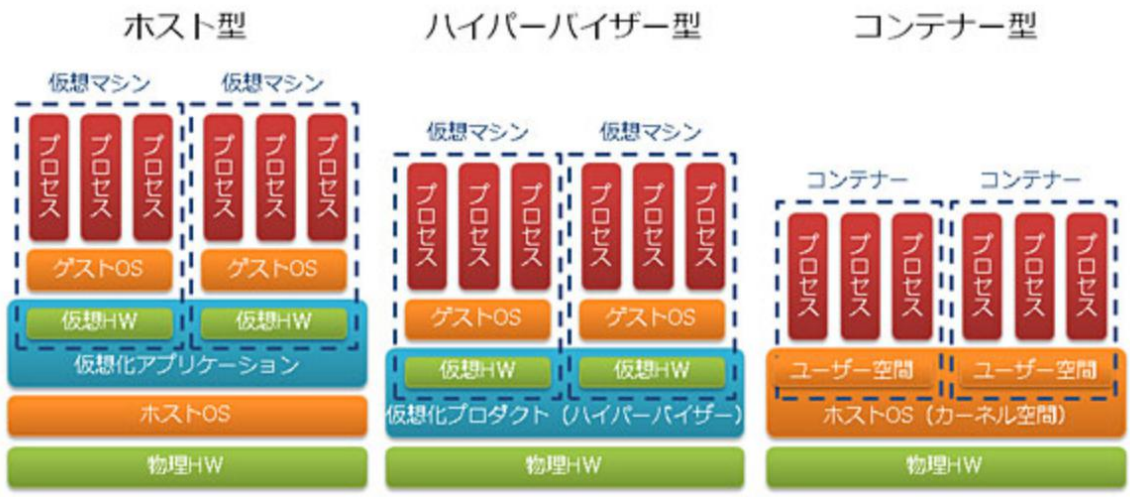
Docker、LXC、OpenVZ




# 03-02. 各仮想化のパフォーマンスの比較

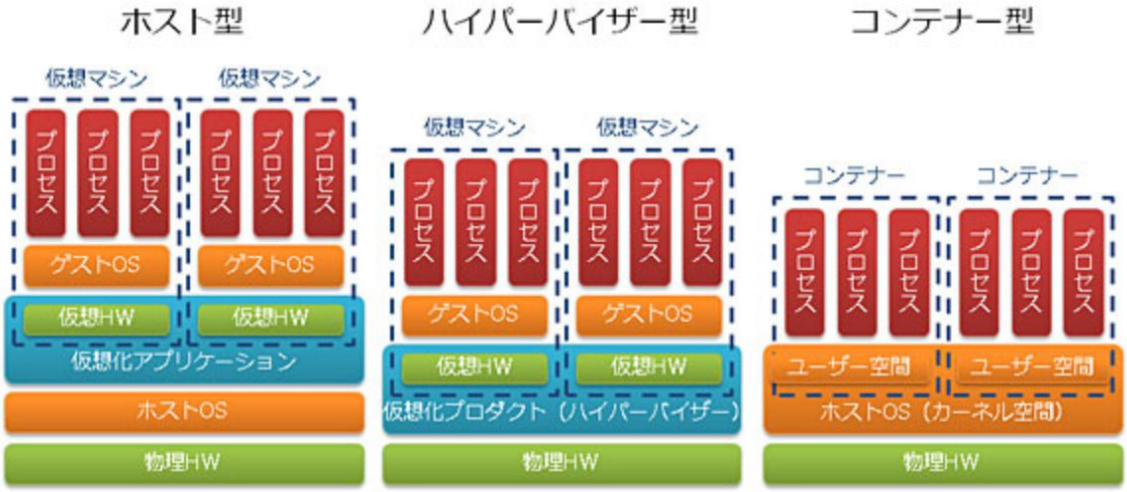
### ◆ 起動速度の違い

ホスト型とハイパーバイザ型では、ハードウェア（CPU、メモリ、ハードディスク）とゲストOSを仮想化することが必要である。一方で、コンテナ型では、ハードウェアとゲストOSの仮想化は行わず、namespaceを用いてコンテナを構成するため、その分起動が速い。



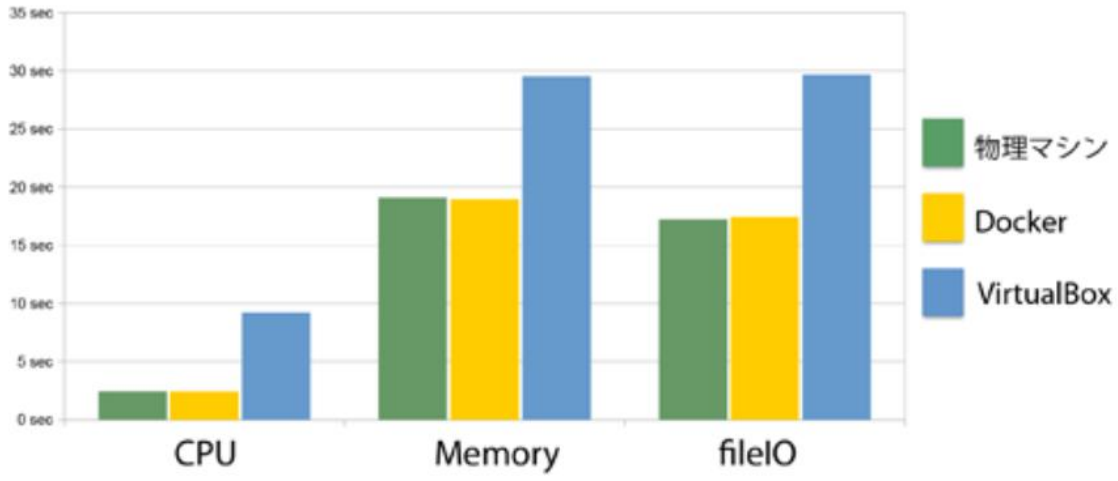
### ◆ 処理速度の違い

ゲストOS上のアプリを操作する場合、ホスト型とハイパーバイザ型では、ハードウェアやハイパーバイザーを経由する必要がある。この分だけ、時間（Overhead）を要する。一方で、コンテナ型では、各コンテナがホストOSとカーネルを共有するため、Overheadが小さい。

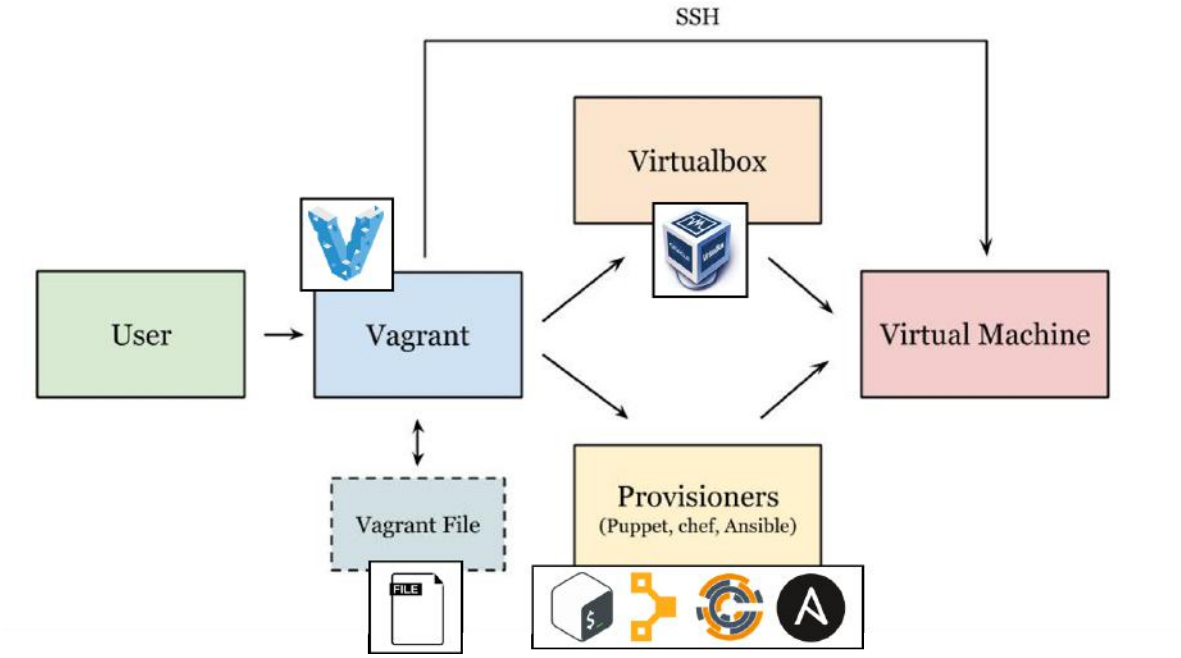



- **Overheadの比較**

sysbenchというベンチマークツールを用いて、CPU・メモリ・ファイルI/Oに着目し、物理マシン・コンテナ型仮想化（Docker）・ホスト型仮想化（VirtualBox）のパフォーマンスを比較。




# 03-03. Provider、Provisioner、Vagrantを用いた仮想環境の構築・環境設定・操作




### ◆ Provider

基本ソフトウェアにおける制御プログラムや一連のハードウェアを仮想的に構築する。構築方法の違いによって、『ホスト型』、『ハイパーバイザ型』、『コンテナ型』に分類できる。

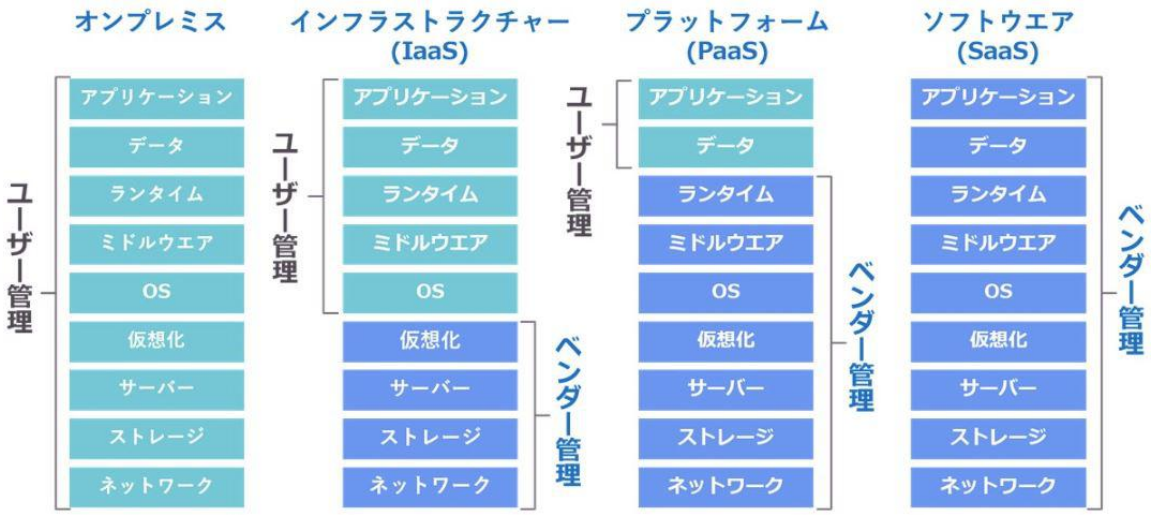
### ◆ Provisioner

Providerによって構築された仮想サーバやコンテナに、Web開発のためのソフトウェアをインストールするアプリ。具体的には、基本ソフトウェアにおける汎用言語プログラムやファイアウォールをインストールする。

### ◆ Vagrant

ProviderとProvisionerによる仮想サーバやコンテナの構築・環境設定・操作を自動化するプログラム。チームメンバーが別々に仮想サーバを構築する場合、ProviderとProvisionerの環境設定に違いが生じてしまう。Vagrantを使う場合、仮想サーバやコンテナの環境設定はVagrantfileに記述されている。また、仮想サーバやコンテナの構築・操作はvagrant経由で行える。これらのために、Vagrantを用いれば、チームメンバーが同じ環境設定の下で、仮想サーバやコンテナを構築・操作することができる。

# 04-01. オンプレミスとクラウドコンピューティング




## ◆ On premise

自社の設備によって、システムを運用すること。

## ◆ Iaas：Infrastructure as a Service

**【Iaasアプリ例】**

Amazon Web Service、Google Cloud Platform、Microsoft Azure、IBM Cloud

## ◆ Paas：Platform as a Service

**【Paasアプリ例】**

Google App Engine、Windows Azure

## ◆ Saas：Software as a Service

従来はパッケージとして提供していたアプリケーションを、Webアプリケーションとして提供するサービスのこと。

**【Saasアプリ例】**

Google Apps（Google Map、Google Cloud、Google Calender など）

# 04-02. masterブランチ変更後の自動テスト

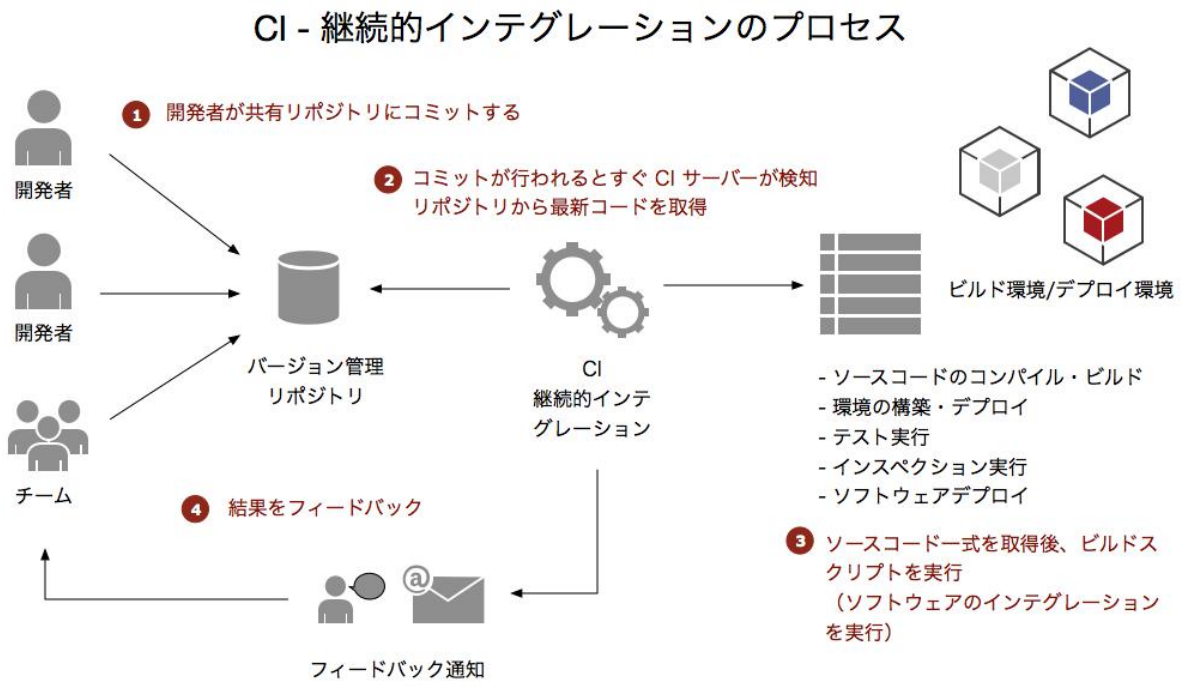
## ◆ 継続的インテグレーションにおける自動テスト（※テスト駆動開発も参照）

1. テスト駆動開発の基、テストディレクトリでテストプログラムを実装したうえで、新機能を設計実装する。
2. ツールが、GituHubから、テストプログラムを含むmasterブランチの状態を取得する。

3. ツールによって、テストサーバの仮想化やコンパイルを行い、テストプログラムを自動で実行する。
4. 結果を通知することも可能。

**【自動化ツールアプリ例】**

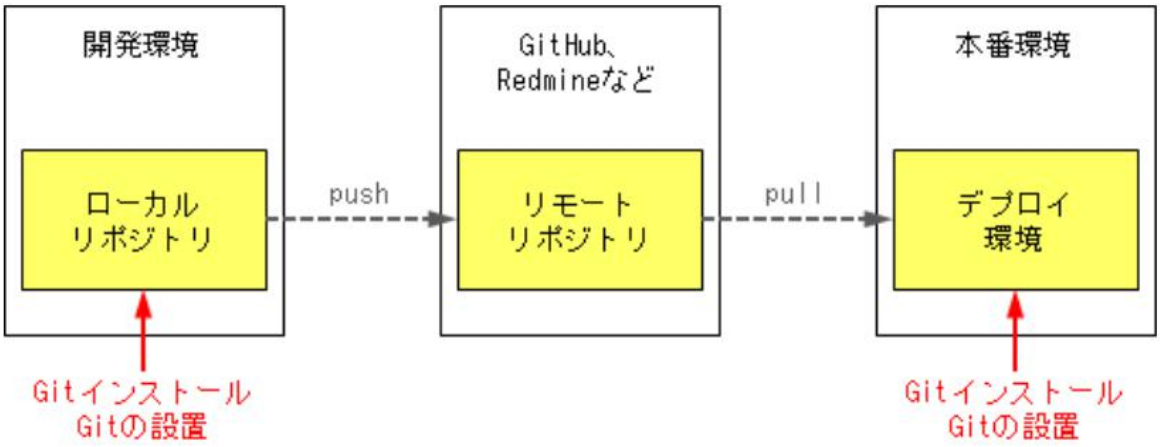
Circle CI、Jenkins



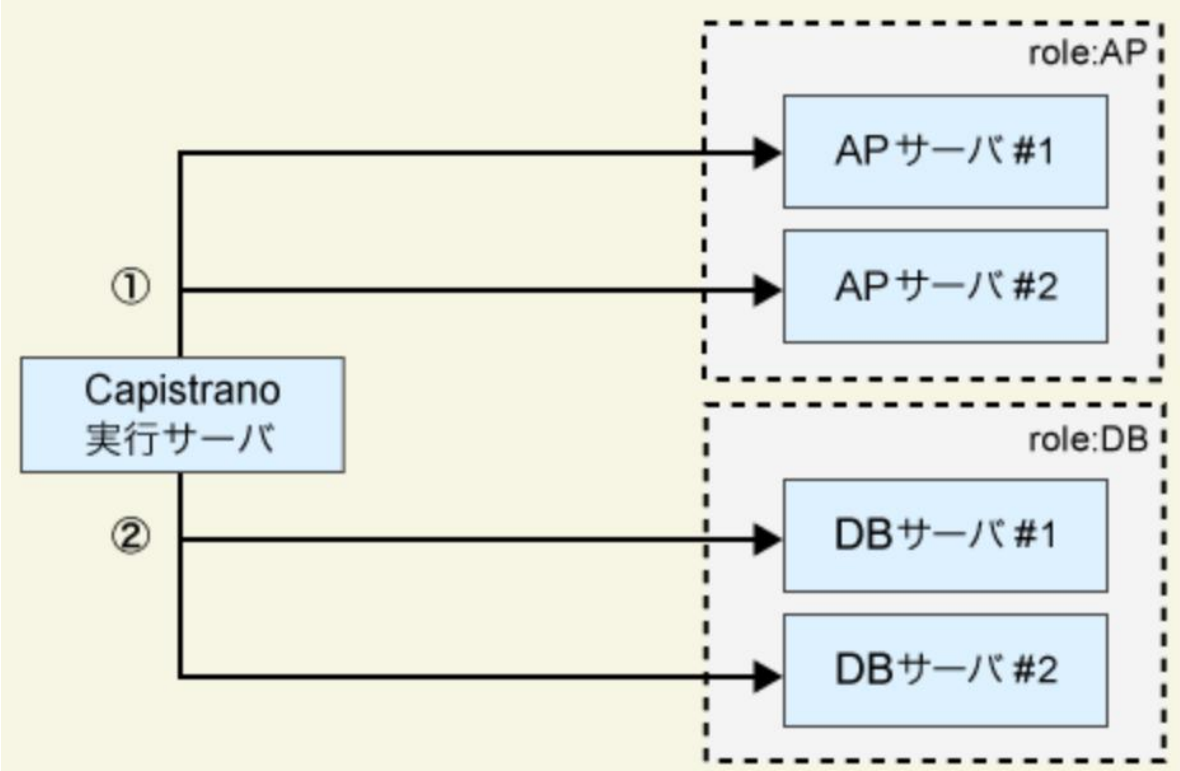

# 04-03. 自動テスト後のAWS仮想サーバへのデプロイ

## ◆ インスタンスへのデプロイにデプロイツールを用いる場合

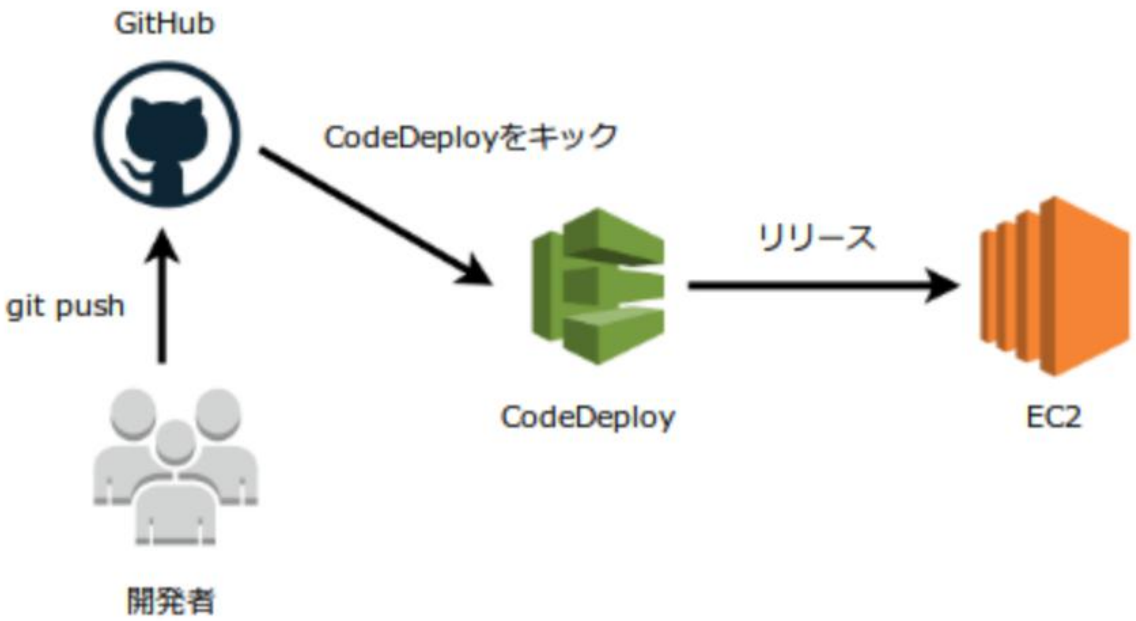
本番環境（ここではクラウドWebサーバ）にGitをインストールして`pull`を行い、GitHubからデプロイサーバにmasterブランチの状態を取り込む。




デプロイ自動化ツール（例：Capistrano）をキックすることによって、デプロイサーバからAWSにおけるインスタンスやデータベースへデプロイが行われる。




### ◆ インスタンスへのデプロイにAWS CodeDeployを用いる場合

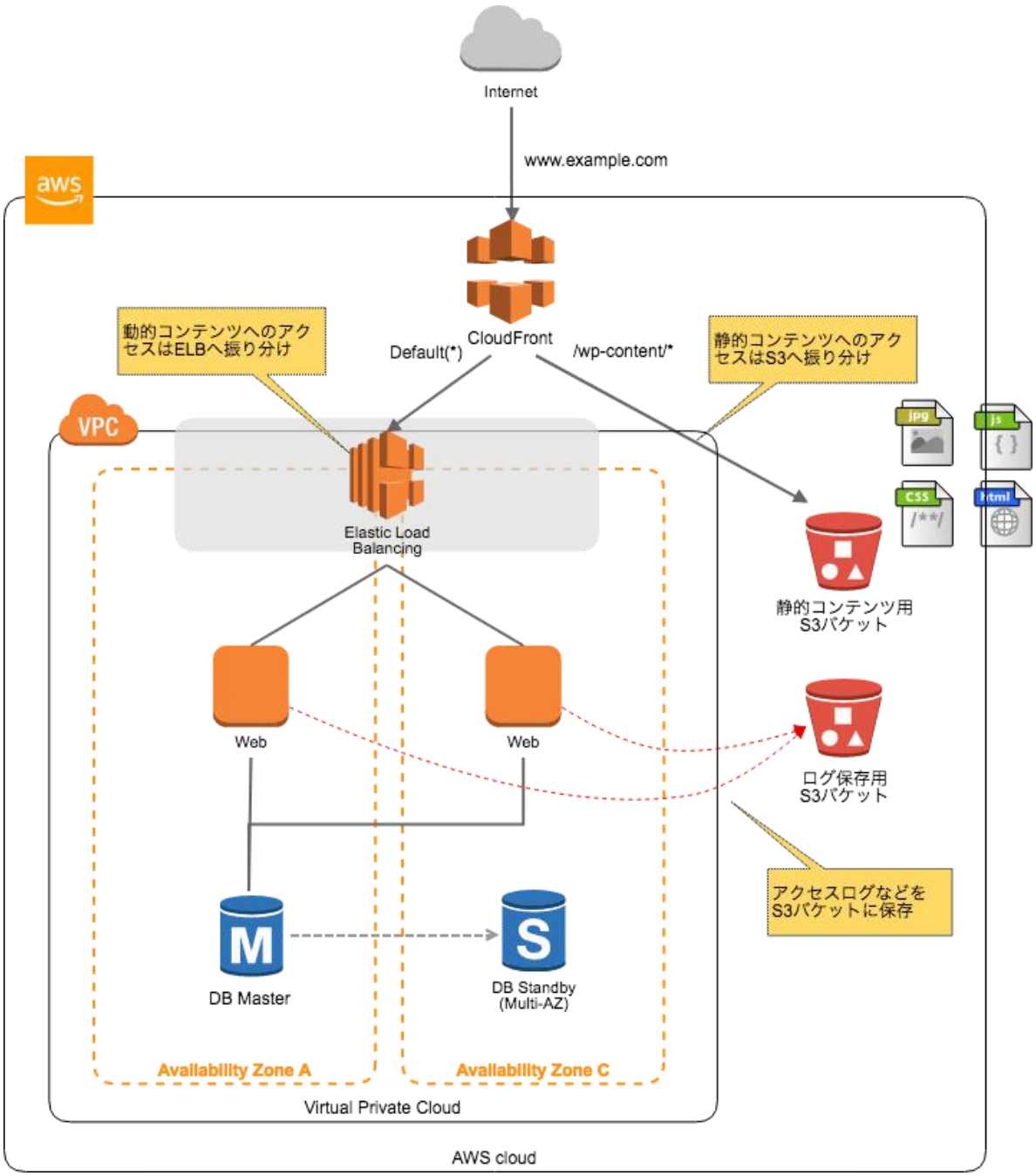



# 04-04. AWSによるWebサービスのリリース

AWSから、グローバルIPアドレスと完全修飾ドメイン名が提供され、Webサービスがリリースされる。

**【クラウドデザイン例】**

以下のデザイン例では、Dualシステムが採用されている。




## ◆ RegionとAvailability Zone

2016年1月6日時点では、以下のRegionにデータセンター群がある。各データセンターには、別系統の電源、空調、ネットワーク機器によって運用されているAvailability Zoneが設置されている。東京Regionには、3つのAvailability Zoneがある。


## ◆ Route 53

クラウドDNSサーバーとして働く。リクエストされた完全修飾ドメイン名とインスタンスのグローバルIPアドレスをマッピングしている。

## ◆ CloudFront

クラウドプロキシサーバとして働く。Bucketへのアクセスをキャッシングする。

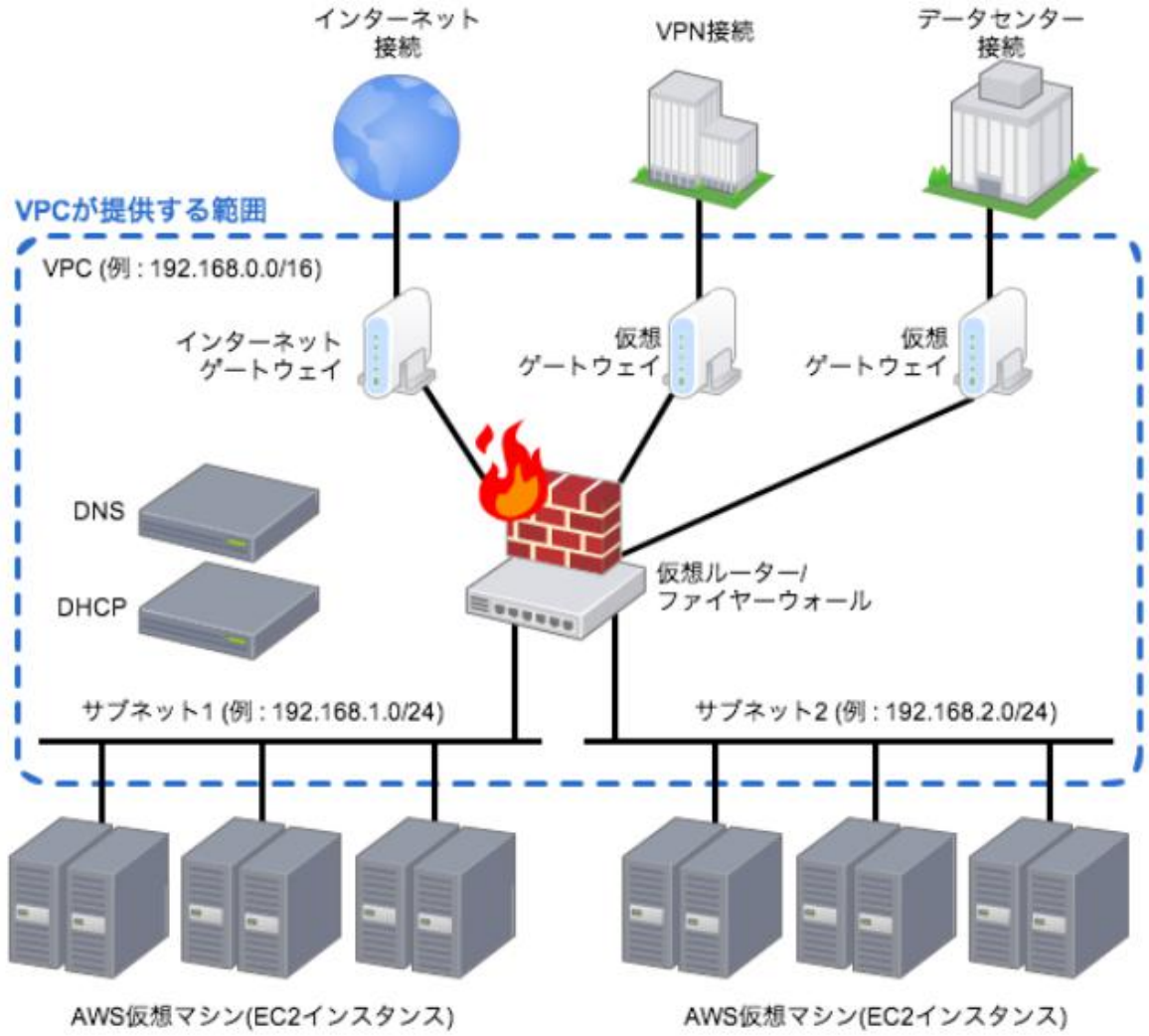
## ◆ S3：Simple Storage Service

- **Bucket**

クラウドストレージとして働く。Amazon S3に保存するCSSファイルや画像ファイルを管理できる。

## ◆ VPC：Virtual Private Cloud

クラウドプライベートネットワークとして働く。プライベートIPアドレスが割り当てられた、VPCと呼ばれるプライベートネットワークを仮想的に構築することができる。異なるAvailability Zoneに渡って仮想サーバを立ち上げることによって、仮想サーバをデュアル化することできる。




- **AZ：Availability Zone**

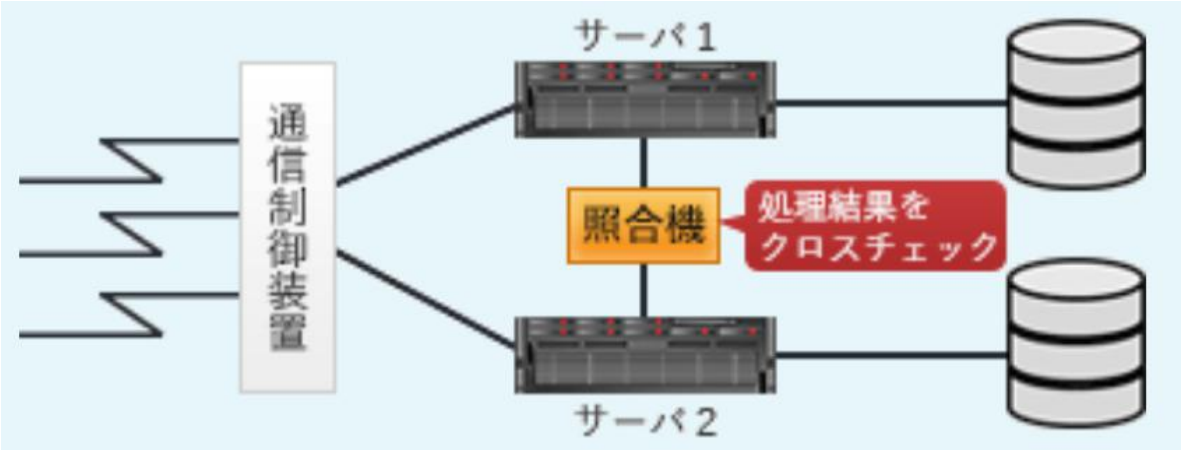
VPCの中に、AZと呼ばれる物理的に離れたデータセンターがある。AZの中に、VPC subnetを作ることができ、そこにインスタンス（クラウドWebサーバ）を構築できる。

- **ELB：Elastic Load Balancing**

デュアル化させたインスタンスへのアクセスを自動的に分配し、サーバへの負荷を緩和するサービス。

## ◆ Dualシステム（再掲）

同じ処理を行う2つのシステムからなるシステム構成のこと。随時、処理結果を照合する。いずれかが故障した場合、異常が発生したシステムを切り離し、残る片方で処理を続けることによって、故障を乗り切る。

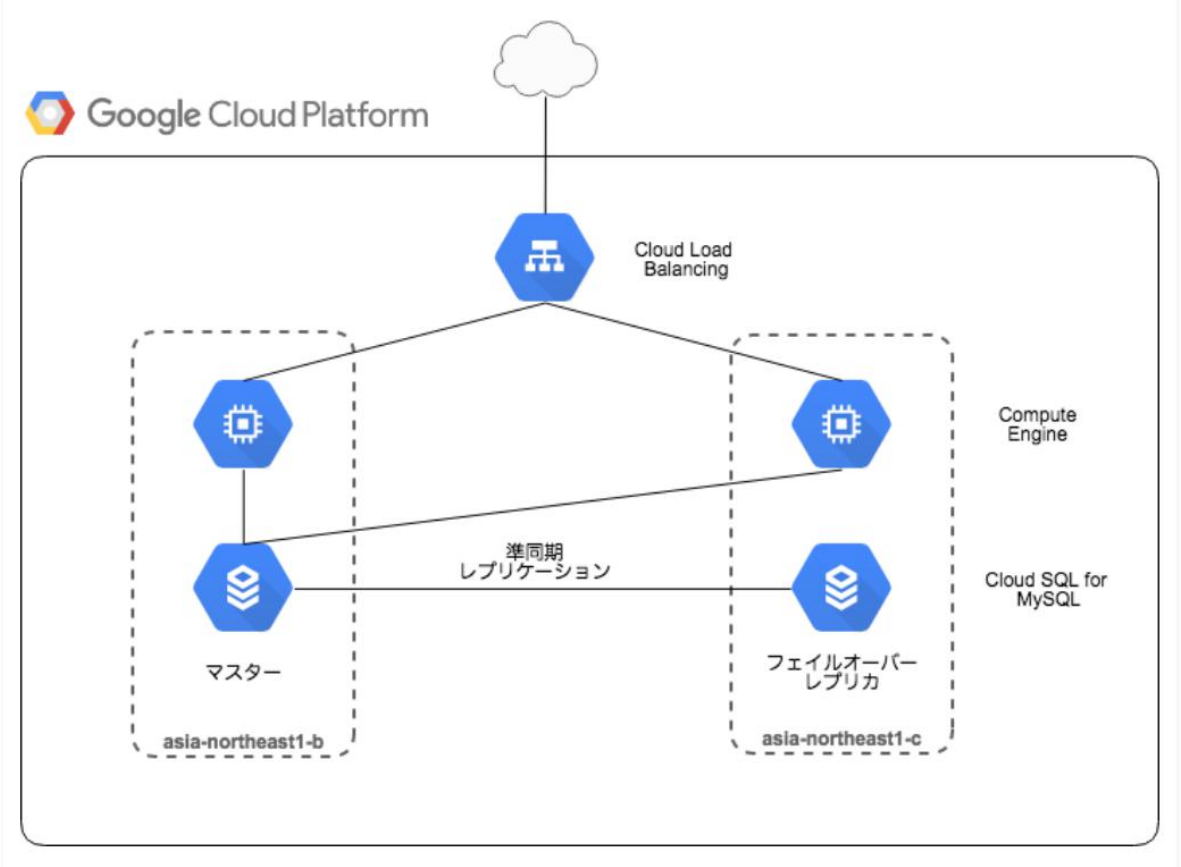



# 05-01. GCPによるWebサービスのリリース

GCPから、グローバルIPアドレスと完全修飾ドメイン名が提供され、Webサービスがリリースされる。

**【クラウドデザイン例】**

以下のデザイン例では、Dualシステムが採用されている。




## ◆ GAE：Google App Engine：GAE

クラウドデプロイサーバとして働く。AWSにおけるElastic Beanstalkに相当する。

## ◆ GCE：Google Compute Engine

クラウドWebサーバとして働く。AWSにおけるEC2に相当する。