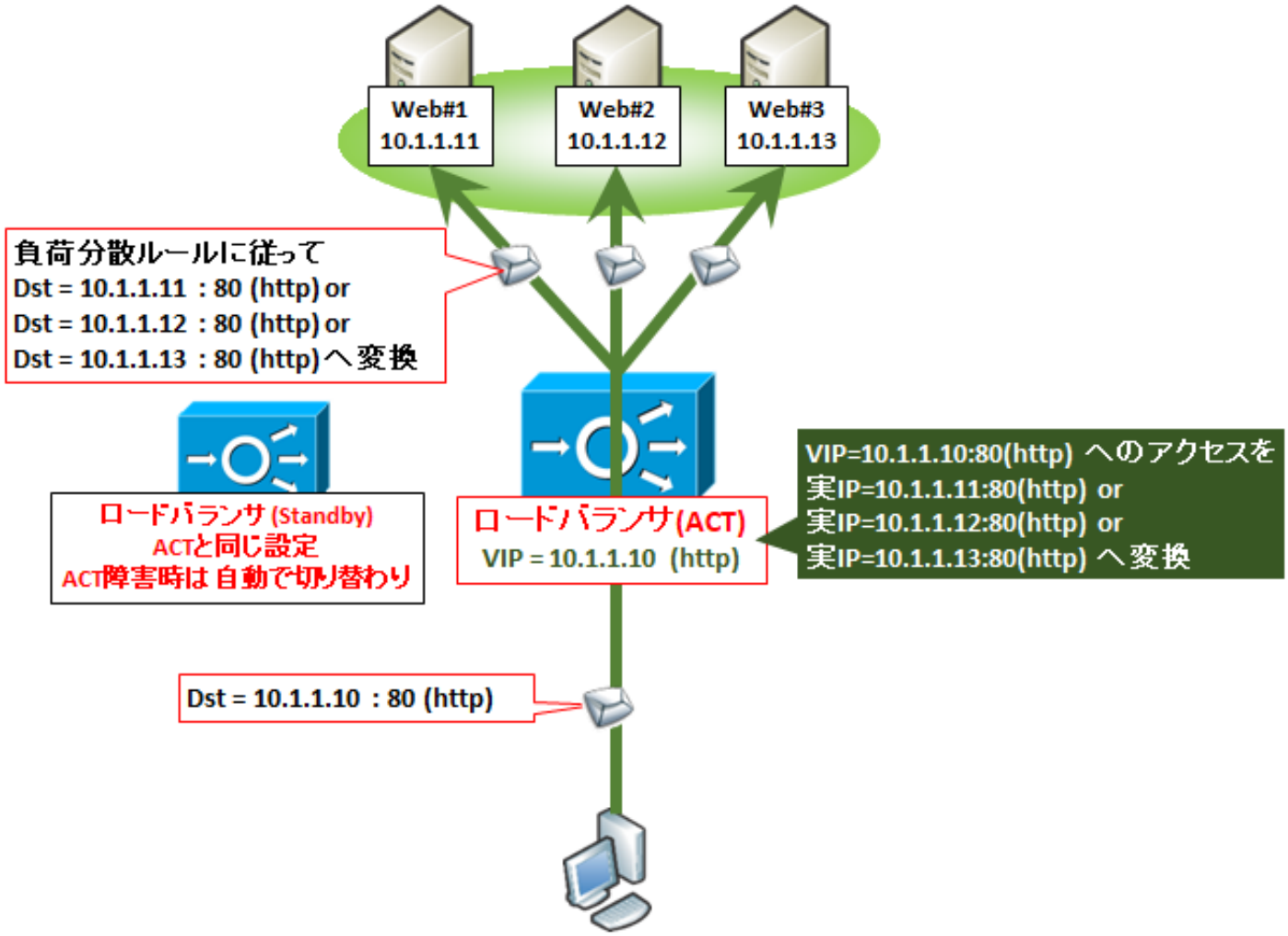
図解】ロードバランサ(L7)の仕組み～セッション維持方式、SSL証明書やリバースプロキシ等の構成例～

2020.03.10　2017.07.12

ロードバランサとは

ロードバランサとは、サーバの**負荷分散**を行いつつ、**冗長化**を実現するアプライアンス機器です。http(s)サーバを負荷分散するイメージが強いですが、DNSやIMAP、RDP(リモートデスクトップ)等の**様々なプロトコルに対応**しているものが多いです。また、サーバの信頼性を上げる機器でもありますので、大抵、このロードバランサ自体もVRRPのような自動切り替わり可能な冗長化機能を持っています。

仕組みとしては、ロードバランサ上の1つの**VIP(Virtual IP: 仮想IPアドレス)**に、負荷分散させたいサーバ群の実IPアドレスたちを紐付けます。そして、VIPへのアクセスがあった際に、**設定された負荷分散ルール(負荷分散方式、および、パーシステンス方式)を元に**、実IPアドレスに変換させます。この動作は、実質リバースプロキシと同じ動作になるため、ロードバランサをリバースプロキシとして構成することも稀にあります。(Windowsサーバを直接DMZに晒したくない時等)



補足ですが、L4 レベル (TCP/UDP 送信元宛先ポート) の情報を使ってNextHopを決める機器を **L4 スイッチ**、http や RDP 等のアプリケーションの中身を見て NextHop を決める機器を **L7 スイッチ**と呼びます。なので**ロードバランサのことを L7 スイッチと呼ぶこともあります**。

製品の種類

製品としては Citrix の Netscaler、F5 の BIG-IP 等が有名です。安価なものだと、セイコーソリューションズの Netwiser が挙がります。

AWS の ELB (Elastic Load Balancer) も仕組みとしては大枠は同じですので、本記事がある程度参考になるかと思います。

また、Windows サーバの場合はハードウェアを追加購入しなくても、NLB (Network Load Balancing) という機能が無料で使えますが、こちらは仕組みがだいぶ異なります。（紛らわしいことに AWS の ELB の種類にも NLB：Network Load Balancer が存在しますが、全くの別物です。）

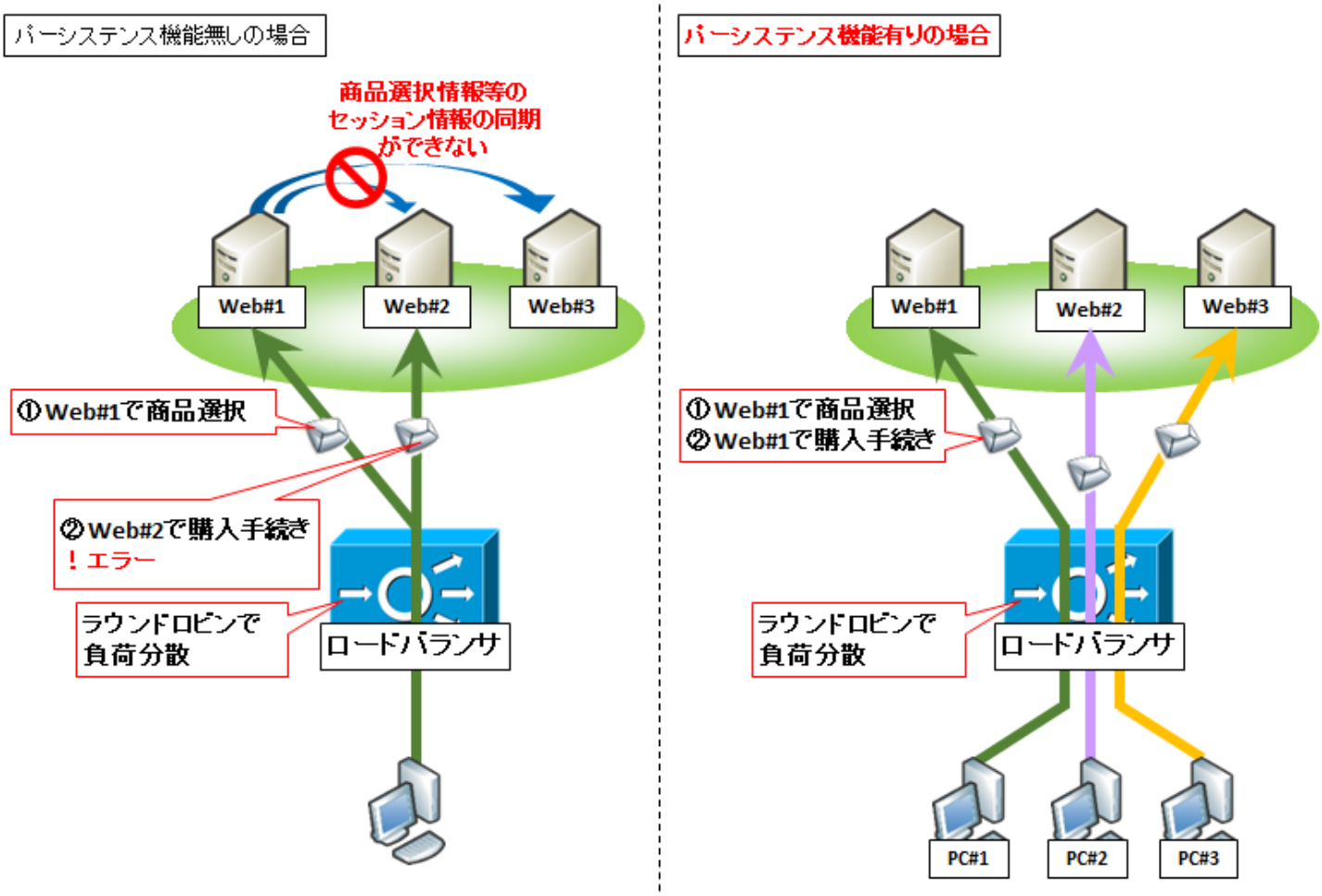
負荷分散方式

負荷分散方式の代表的な例を以下に挙げます。

> ラウンドロビン・・・最初に来た通信を Web サーバ#1へ、次に来た通信を Web サーバ#2へ、と順番に割り当てていく
>
> 最小レスポンス時間・・・死活監視を行った際のレスポンス時間が一番短いサーバを使う
>
> 最小コネクション数・・・一番接続数の少ないサーバを使う
>
> 最小利用帯域・・・一番トラフィック流量の少ないサーバを使う

パーシステンス方式

ロードバランサは通信を割り振りますが、**サーバ間の同期までは面倒を見てくれません**。また、サーバ側も『1 つの Web サーバでのセッション情報を他のサーバに同期する』ということは出来ません。なので、ロードバランサ側で、『**あるユーザが色々な Web サーバを行ったり来たりせず、1 台の Web サーバを見続けさせる機能**』を用意しています。それが、**パーシステンス機能**です。



パーシステンス方式の代表的な例を以下に示します。

> クッキー・・・**http のクッキー情報**をロードバランサが挿入し、ユーザを識別する。同じクッキを持つ通信は必ず同じ Web サーバを使わせる
>
> 資格情報・・・**RDP のユーザ ID 情報**を識別する。同じユーザ情報わ持つ通信は必ず同じ Terminal Server (TS) を使わせる
>
> 送信元 IP・・・同一 IP の通信を同一ユーザと見なす。同じ送信元 IP の通信は必ず同じサーバを使わせる

ロードバランサでの https 通信対応 (TLS/SSL暗号化および復号化)

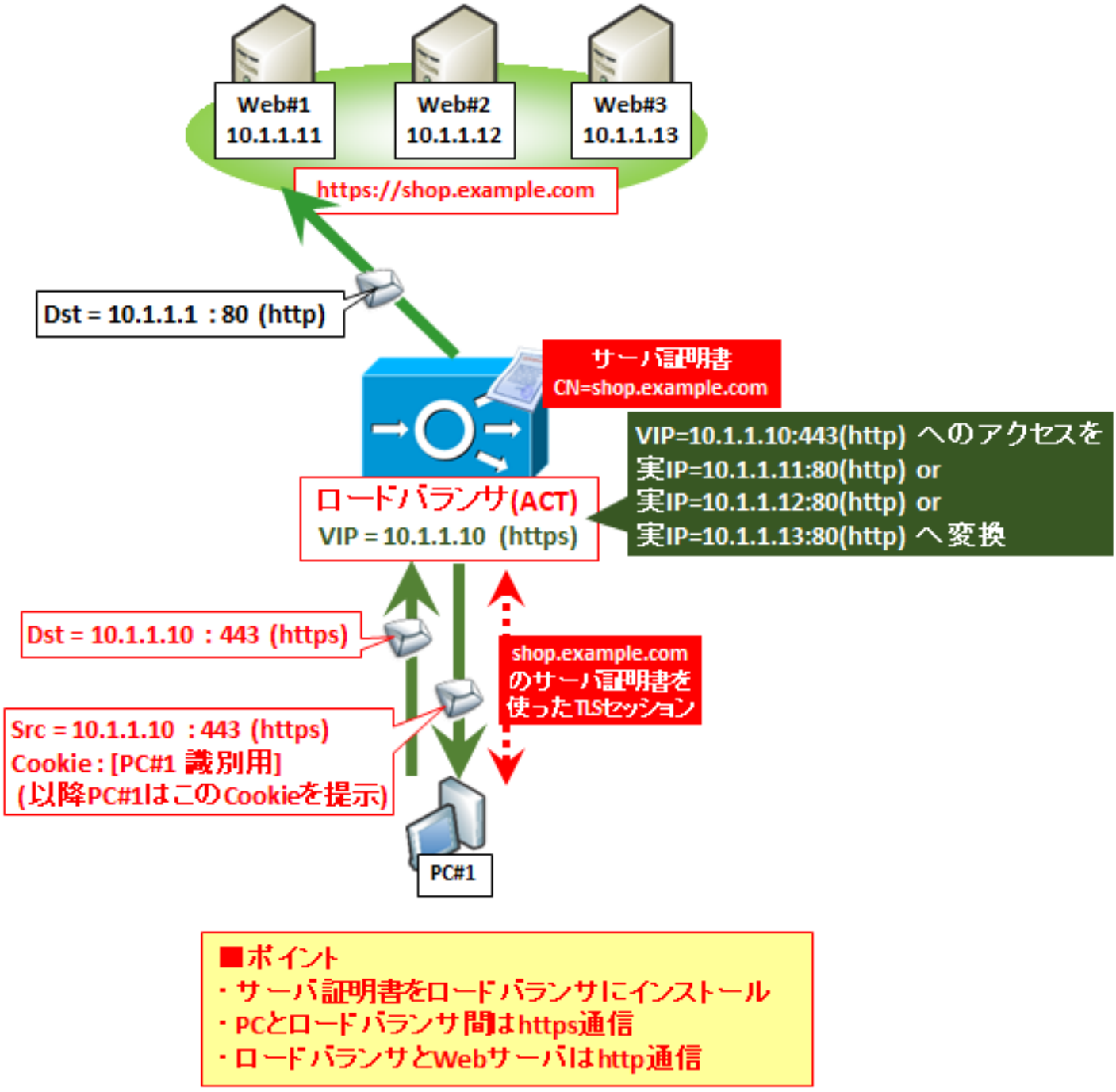
https 等の暗号化された通信でも、この L7 のスイッチングを実現したい、という需要が多くあります。

これを実現するためには、暗号化されている https 通信等を**復号化**させて http の中身を見る必要があります。

つまり、TLS 終端を『クライアント⇔サーバ間』ではなく『クライアント⇔負荷分散装置間』にします。

その関係で、ロードバランサには **SSL アクセラレータ**(通常 CPU に処理させる TLS 暗号化・復号化を、専用 ASIC によりハードウェア処理させる)機能を持っていることが多いです。

実装としては、**本来サーバに配置すべきデジタル証明書および秘密鍵をロードバランサに配置**し、**PC とロードバランサ間**は、そのデジタル証明書および秘密鍵を使って **https 通信**を行い、ロードバランサはそれを http に復号化し、http のクッキー情報を挿入・閲覧し、クッキーに応じた Web サーバへ http 通信を送ります。

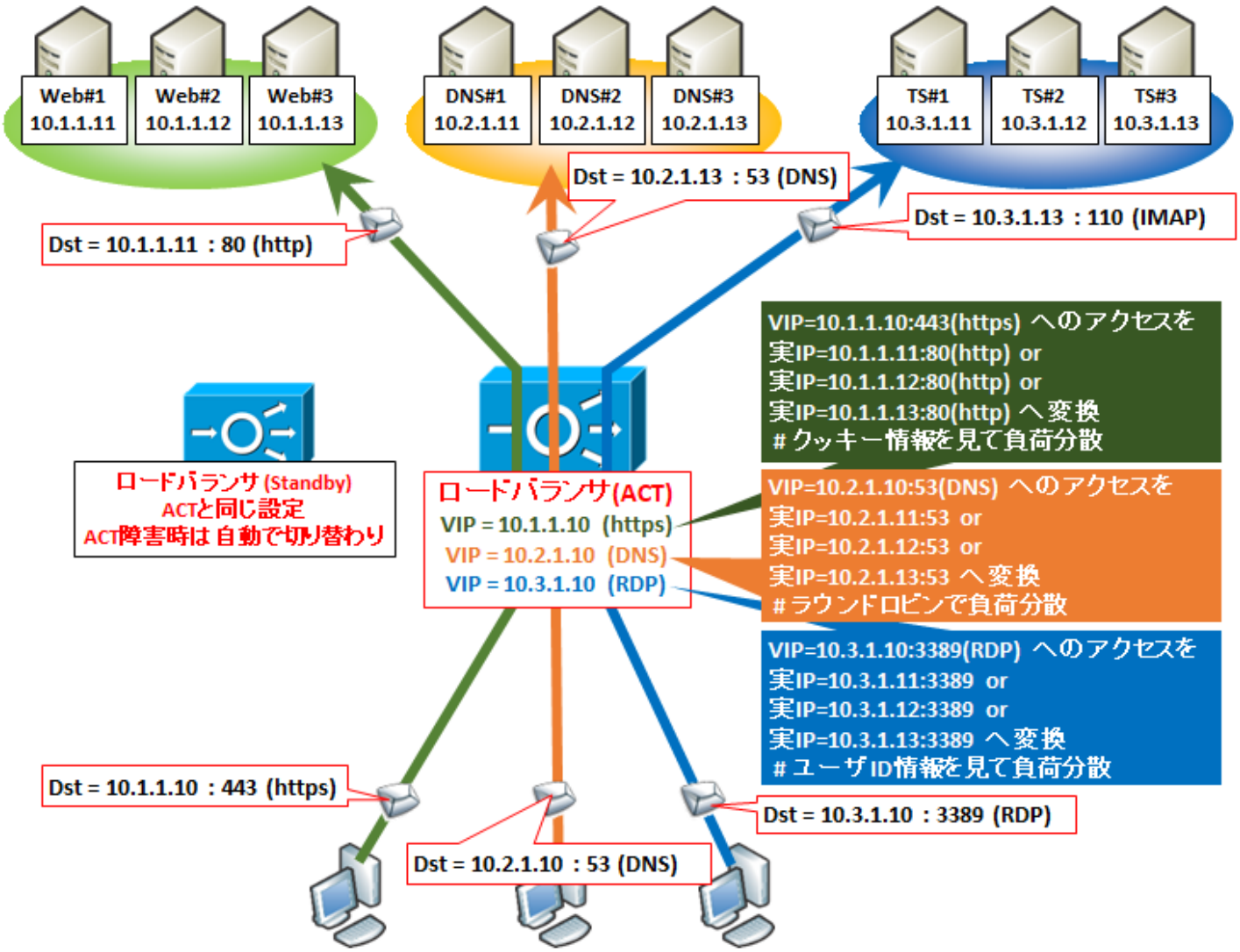


ロードバランサと Web サーバの間はセキュリティが下がりますが、**2 つの機器の間が距離的に近くにあり、その間であれば盗聴(パケットキャプチャ)はされない**、という前提であれば問題はありません。

ですので、Web サーバにはデジタル証明書を配置する必要はありません。ただ、有っても害は無いので、**デジタル証明書のライセンス的に問題ないのであれば**（昔は配置する機器1台につき 1 ライセンスでしたが、最近は同じ証明書であれば OK のものも多い）配置しておいたほうが万が一の対応としてロードバランサを経由させず直接 Web サーバに https 通信させることができるので良いかもしれません。

ロードバランサの通信イメージ

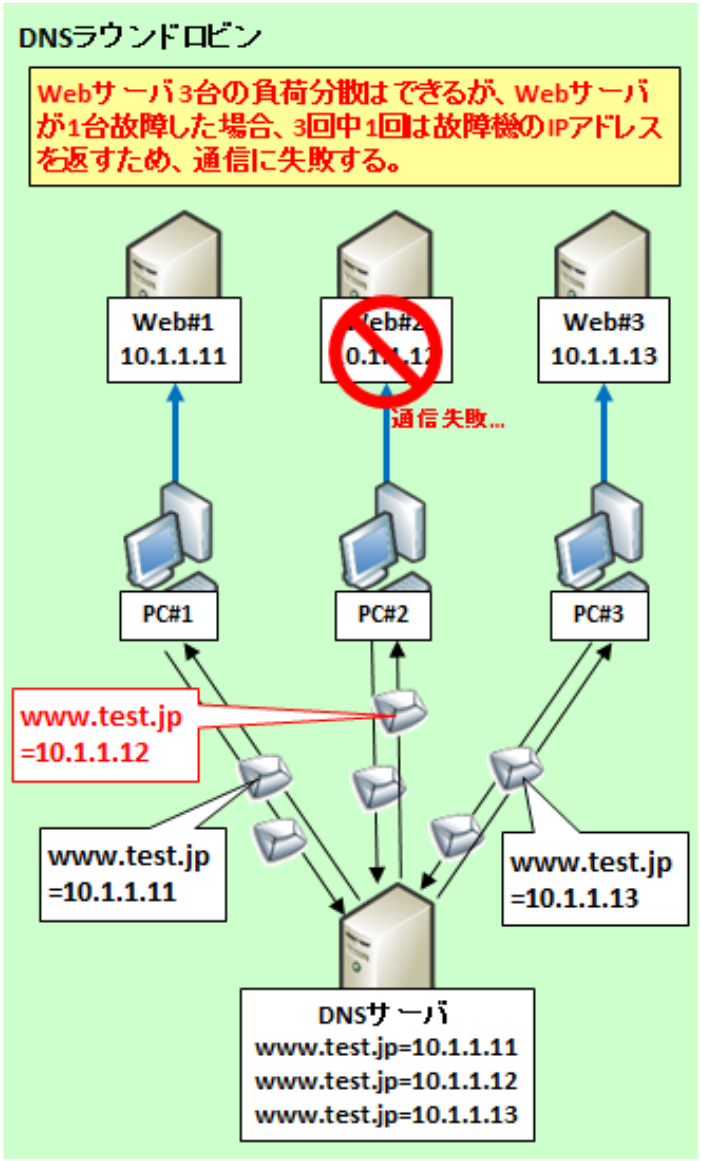
ロードバランサの通信イメージを以下に示します。

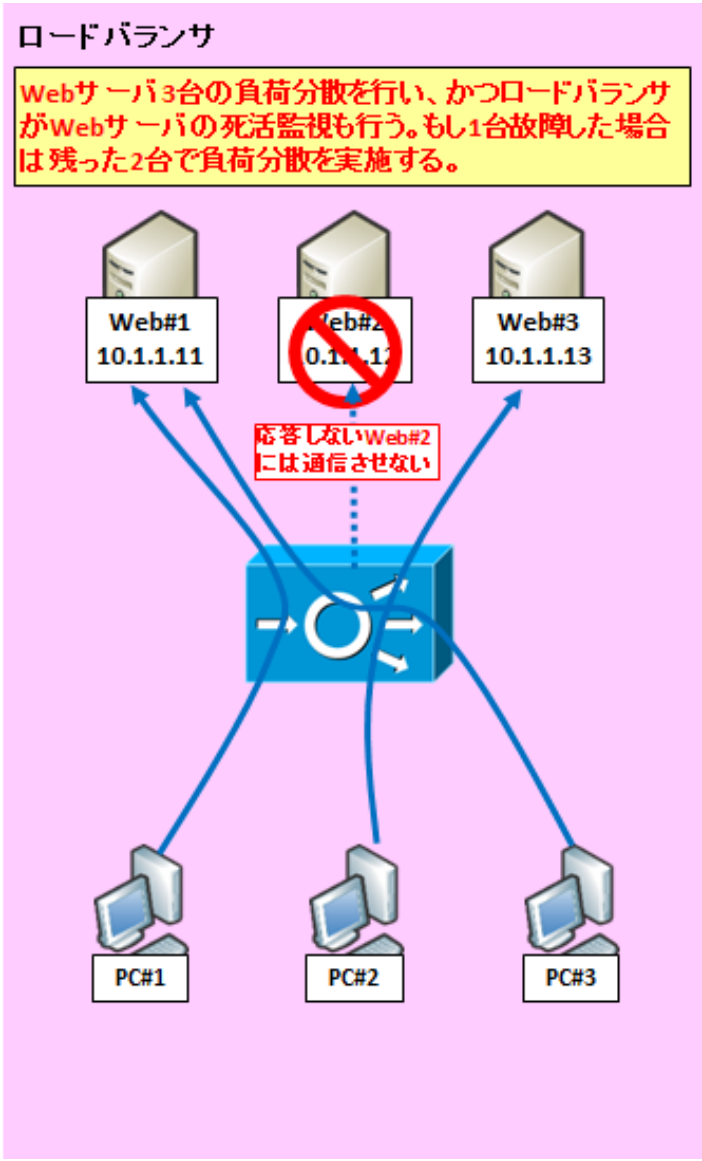


ロードバランサとDNSラウンドロビンの決定的な違い　～負荷分散と冗長化はイコールではない～

冒頭で、『**負荷分散と冗長化を実現する**』と書きました。もしかしたら「**負荷分散出来ているなら、冗長化は自然と出来ているでしょ**」と思ったかもしれませんが、厳密には異なります。

その一番分かり易い例は、**DNS ラウンドロビン**です。これは負荷分散は出来ますが、冗長化は出来ません。**DNS ラウンドロビンでは、ロードバランサと違い、死活監視を行わないため**です。





ロードバランサの構成パターン

ロードバランサの構成パターンは大きく分けて2つあります。

1つが **Destination-NAT (DstNAT) 構成**、もう1つが **Source & Destination NAT (Src&DstNAT) 構成**です。

DstNAT 構成

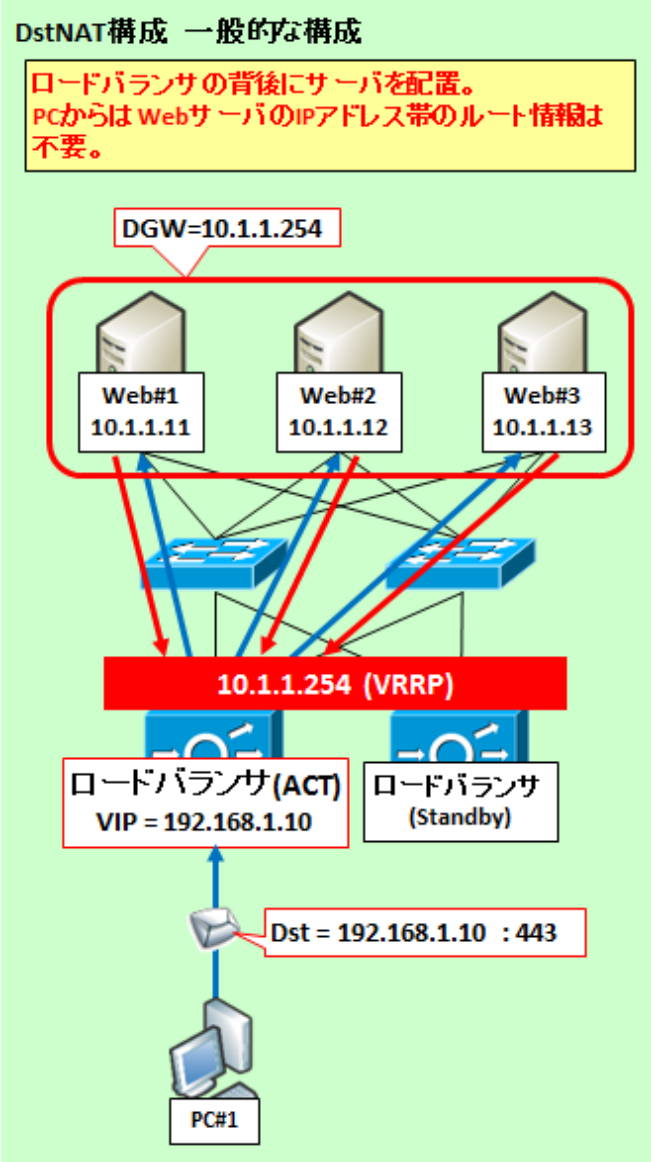
VIP をサーバの実 IP に DstNAT 変換します。

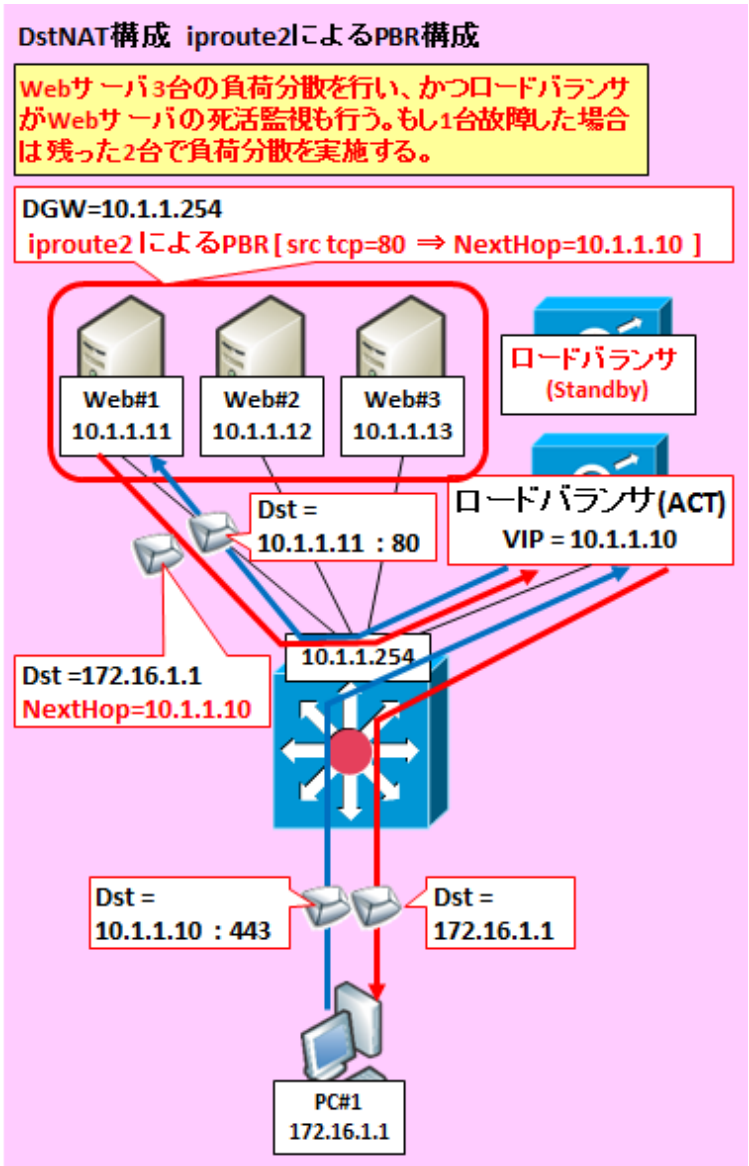
この構成のメリットは、送信元 IP が変換されないため、**サーバ側で送信元 IP のログが残せること**です。

一方、**デメリット**は、**戻りの通信がロードバランサを経由するようにしっかりと設計する必要がある**点です。

構成例として一番素直なのは『2アーム構成』と呼ばれ（インライン方式とも呼ばれます）、**ロードバランサの背後にサーバ専用のセグメントを作る**ことです。ロードバランサから 2 本の足（アームだから腕?!）が出ます。

また、ちょっとトリッキーな構成にはなりますが、Linux サーバであれば、**IPROUTE2 による PBRを行うことで、ロードバランスするプロトコルだけ NextHop をロードバランサに向ける**、というやり方もあります。この構成は 1 アーム構成と呼ばれます。

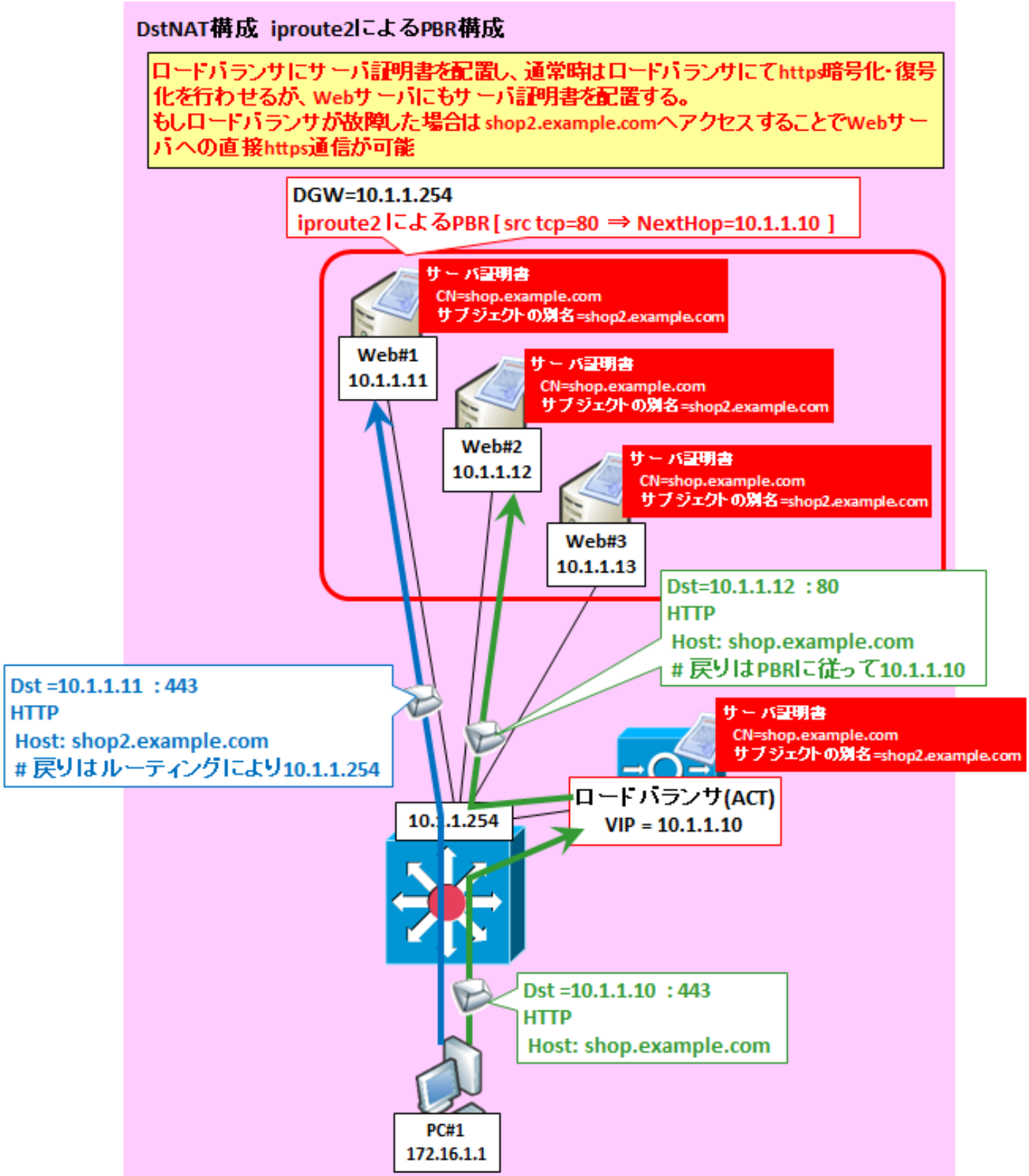




また、この IPROUTE2 を使った実装ではさらにトリッキーな構成として、**ロードバランサの VIP でも Web サーバの実 IP でも両方 https 通信できる状態にする**ことができます。

これにより、高価なロードバランサの購入を 1 台に抑えつつ、ロードバランサ障害時には DNS のレコード変更（もしくはクライアントの hosts ファイル追記）のみの簡易な手動切り替えで暫定対応ができる構成を組めます。

さらに発展させ、証明書のエイリアス（サブジェクトの別名、サブジェクト代替名）を使えば、ロードバランサ障害時はクライアント側で URL を変えるだけで対応できます。



Src&DstNAT構成

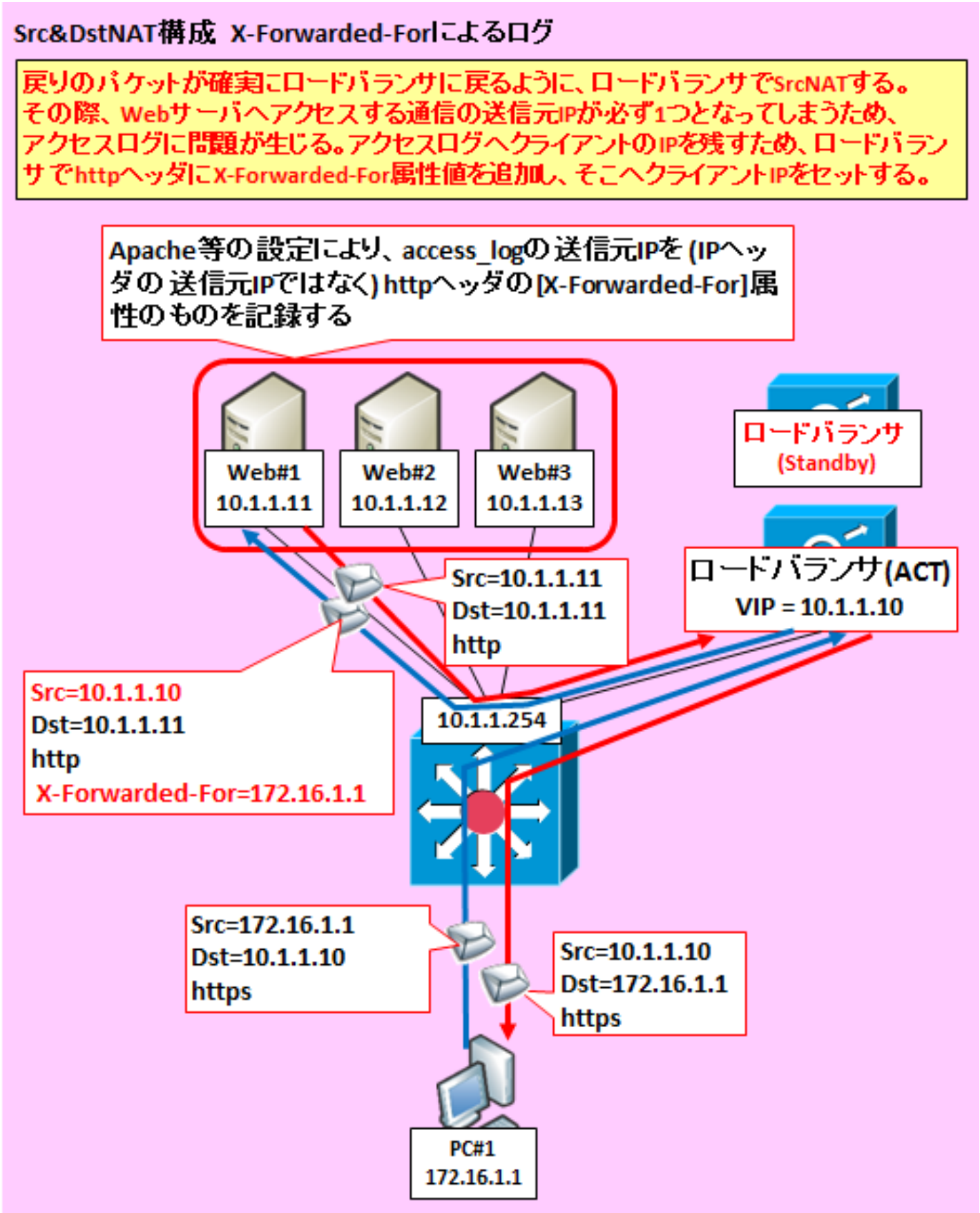
VIP から実 IP に Destination NAT するタイミングで、一緒に **Source NAT すれば**、**戻りの通信は必ずロードバランサ経由になる**ので、ネットワークの考慮は不要になります。

ただし、デメリットとして**サーバ側から見たらアクセス元の IP が全て同じになってしまいます**。

ですが Web サーバへのアクセスログについては、http の **X-Forwarded-For (XFF) 属性**や、Proxy protocol 使うことでこの問題は解決できます。

ロードバランサにて http の中身に **X-Forwarded-For の属性や Proxy protocol の情報を追加**し、その属性値として Source NAT 変換前のIPアドレス(つまり**クライアントの IP アドレス**)を記載します。

Web サーバ側では、アクセスログの送信元 IP として「**X-Forwarded-For 属性がある場合はこちらをログに残す**」という設定を入れておきます。構成例を下記に示します。


Src&DstNAT構成 X-Forwarded-Forによるログ
戻りのパケットが確実にロードバランサに戻るように、ロードバランサでSrcNATする。
その際、Webサーバへアクセスする通信の送信元IPが必ず1つとなってしまうため、
アクセスログに問題が生じる。アクセスログへクライアントのIPを残すため、ロードバランサでhttpヘッダにX-Forwarded-For属性値を追加し、そこへクライアントIPをセットする。
Apache等の設定により、access_logの送信元IPを(IPヘッダの送信元IPではなく)httpヘッダの[X-Forwarded-For]属性のものを記録する
Web#1
10.1.1.11
Web#2
10.1.1.12
Web#3
10.1.1.13
ロードバランサ
(Standby)
ロードバランサ(ACT)
VIP = 10.1.1.10
Src=10.1.1.11
Dst=10.1.1.11
http
Src=10.1.1.10
Dst=10.1.1.11
http
X-Forwarded-For=172.16.1.1
10.1.1.254
Src=172.16.1.1
Dst=10.1.1.10
https
Src=10.1.1.10
Dst=172.16.1.1
https
PC#1
172.16.1.1