クラウド基盤構築演習

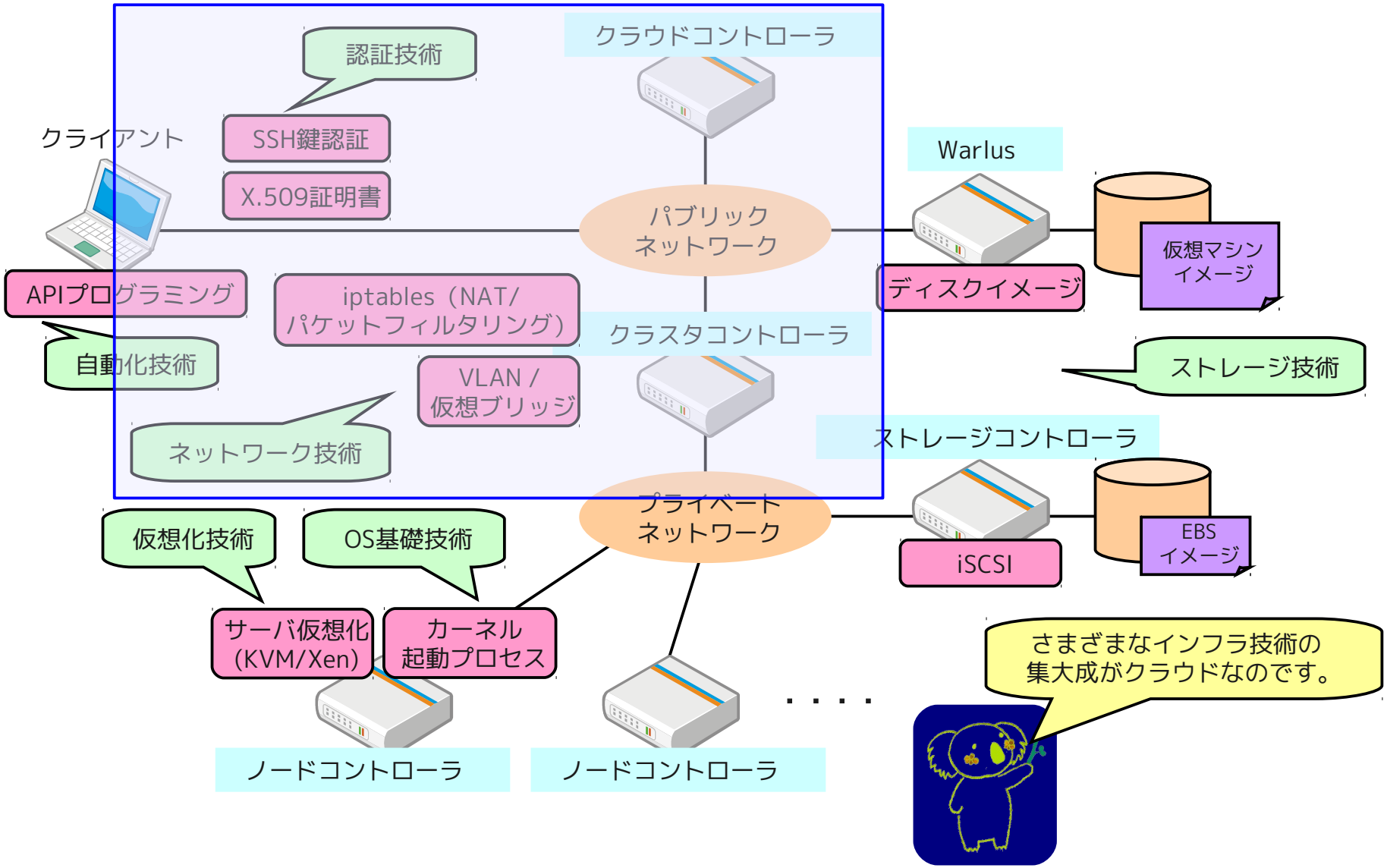
第一部
クラウド基盤を支えるインフラ技術
～ 第3回 IPネットワークと認証技術の基礎

ver1.1 2012/05/01

目次

- IPネットワークとVLANの基礎
- iptablesの基礎とパケットフィルタリング
- iptablesによるNAT処理
- 鍵ペアによるSSH認証/SOAP API認証
- 参考資料: iptablesのパケットマッチング条件

「第３回～第４回」の対象範囲



メモとしてお使いください

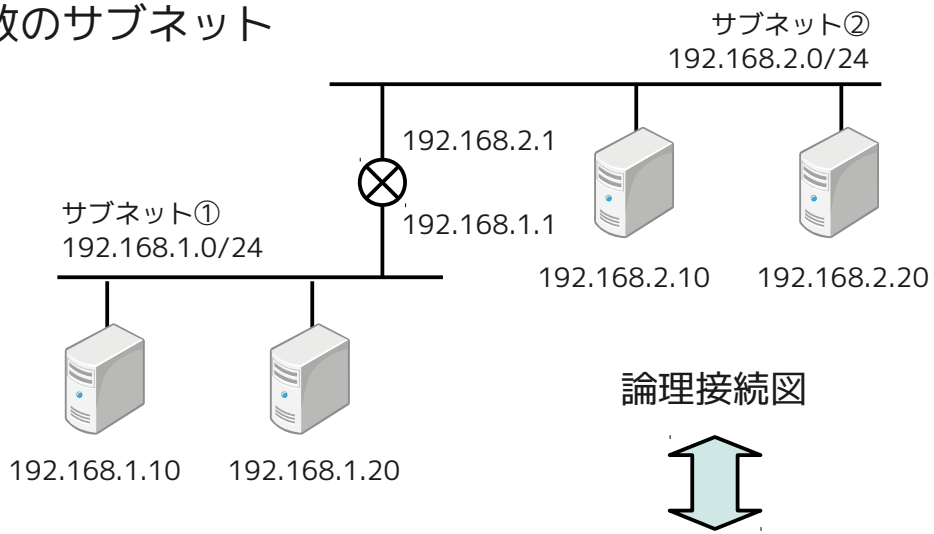
メモとしてお使いください

メモとしてお使いください

IPネットワークとVLANの基礎

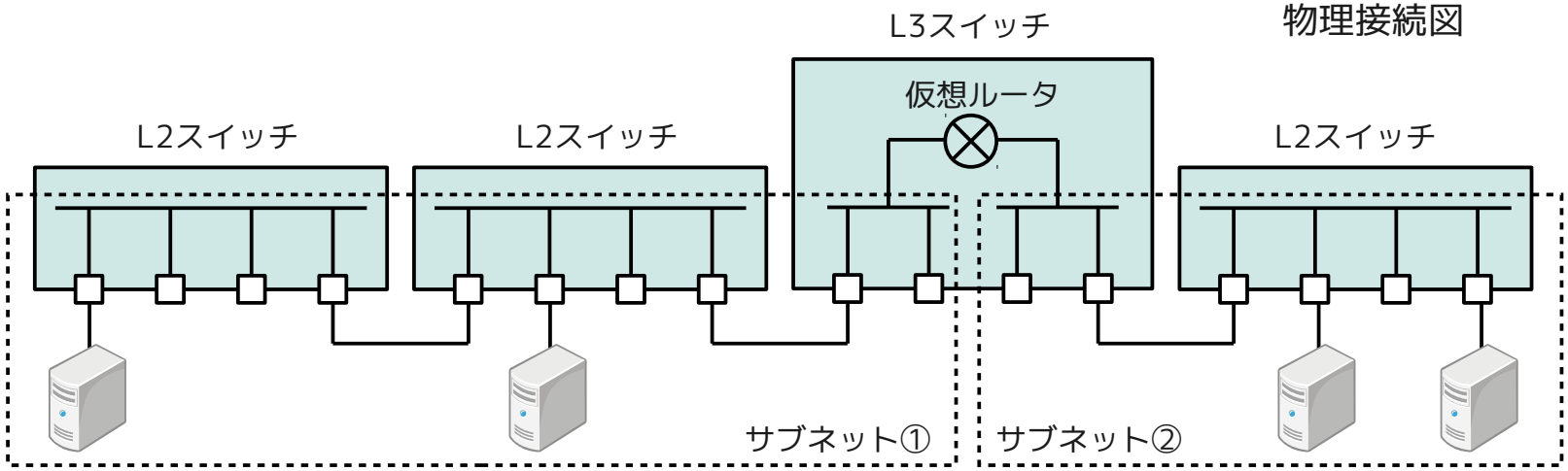
IPネットワークの基本構造

- IPネットワークの基本単位は、L2スイッチで相互接続されたひと続きの「サブネット」です。複数のサブネットがルータによって相互接続されます。
  - 同じサブネットに接続されたサーバは、ルータを介さずに直接、パケットを交換します。
  - 異なるサブネットに向けたパケットは、ルータが中継していきます。
  - L3スイッチは、内部に「仮想ルータ」を持つネットワークスイッチです。



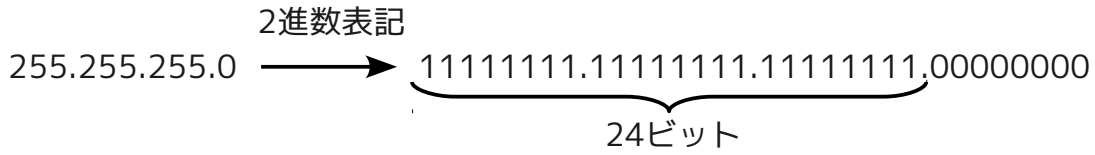
論理接続図

物理接続図



ネットワークアドレスの表記方法

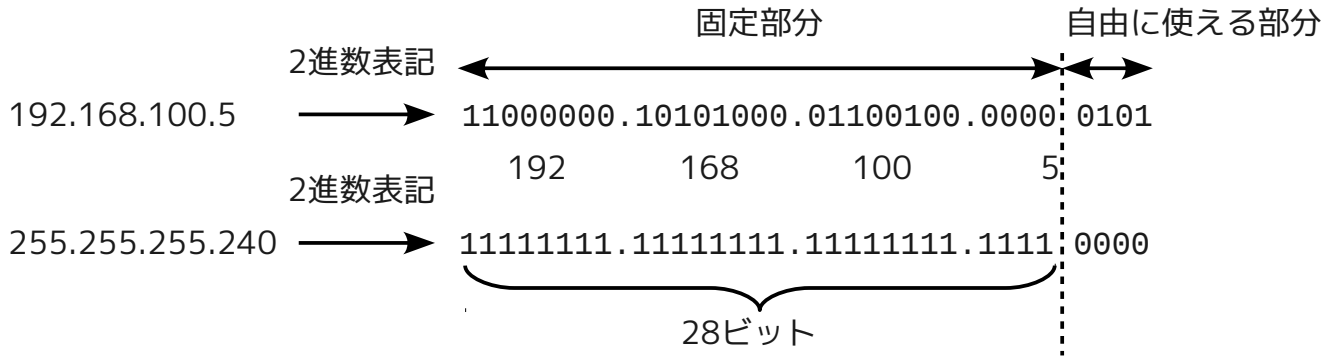
- 各サブネットには、利用できるIPアドレスの範囲を示す「ネットワークアドレス」が割り当てられます。
  - ネットワークアドレス「192.168.1.0/24」は、「192.168.1.0」の左から24ビット分が固定で、その後ろを自由に使えるという意味です。
    - IPアドレスは、「8ビット.8ビット.8ビット.8ビット」という数字の並びですので、今の場合は、「192.168.1.」までが固定で、最後の8ビット（0～255）を自由に使えることになります。
    - この範囲の最初と最後にあたる「192.168.1.0」と「192.168.1.255」は特別な意味を持つので、サーバに割り当てることはできません。（最初の「192.168.1.0」はネットワークそのものを表すアドレスで、最後の「192.168.1.255」はブロードキャストアドレスになります。）
  - 「192.168.1.0/255.255.255.0」は、「192.168.1.0/24」と同じ意味になります。
    - これらの関係は下図を参照してください。「/255.255.255.0」や「/24」を「ネットマスク」と呼びます。



- IPアドレスを表記する際は、ネットワークアドレスを明示するためにネットマスクを併記することがあります。
  - 例えば、「192.168.1.10/24」は、ネットワークアドレス「192.168.1.0/24」のサブネットに属することがわかります。
  - 1つのサブネット内に異なるネットワークアドレスのIPアドレスを混在することはできません。

ネットワークアドレスの計算例

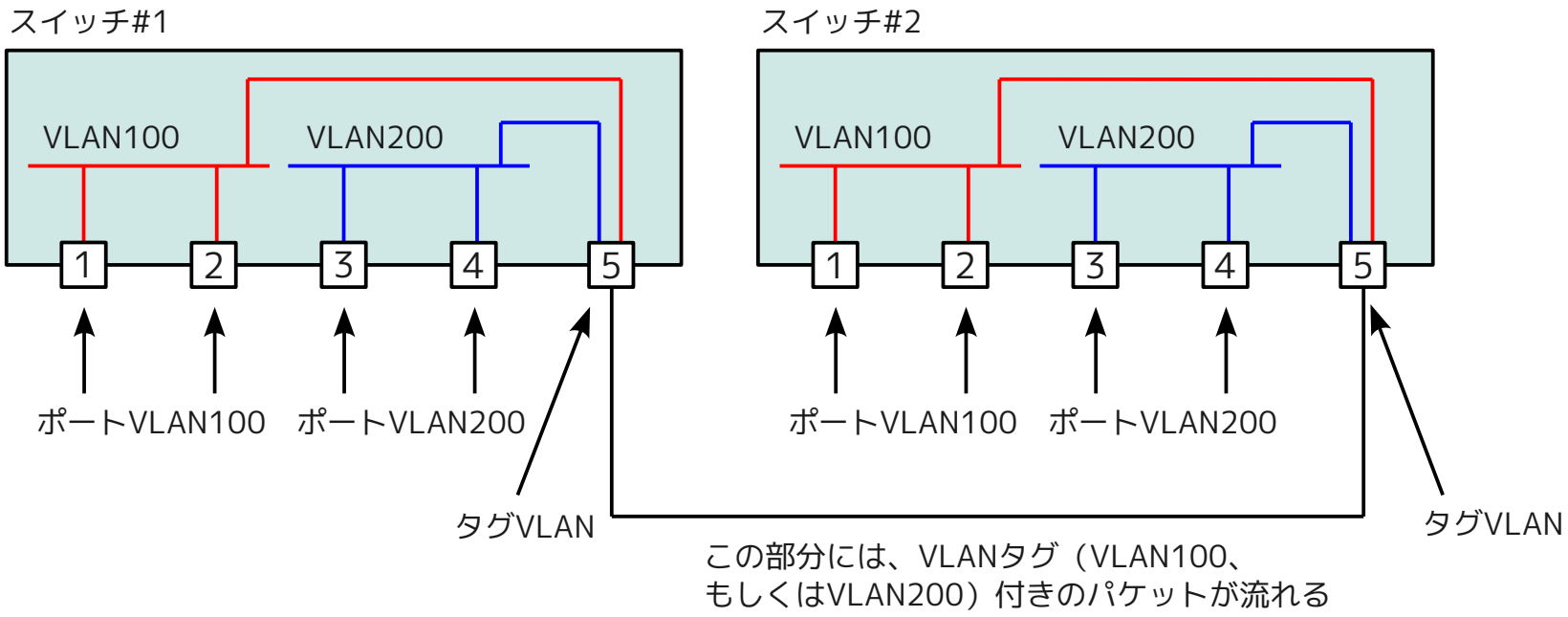
- IPアドレス「192.168.100.5/255.255.255.240」が属するサブネットのネットワークアドレスを計算してみます。
  - 下図より、自由に使える部分は末尾の4ビットで、IPアドレスの範囲は「192.168.100.0～192.168.100.15」とわかります。
  - 自由に使える部分に0を入れたものがネットワークアドレスになるので、「192.168.100.0/28」あるいは「192.168.100.0/255.255.255.240」が答えになります。



- 同様に、IPアドレス「192.168.100.24/255.255.255.240」が属するサブネットのネットワークアドレスを計算するとどうなるでしょうか？
  - 答えは「192.168.100.16/28」です。

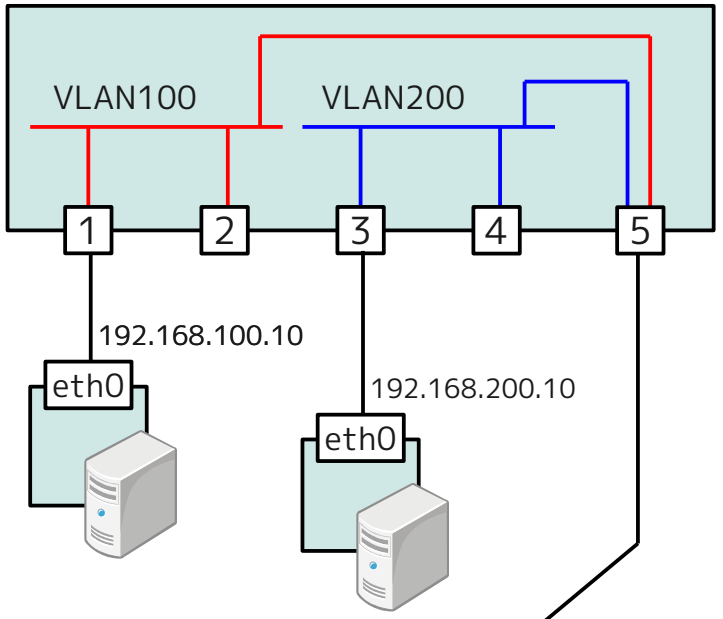
VLANの基本的な仕組み

- VLANは、L2スイッチ内部に複数のネットワークセグメントを作成する機能です。
  - ポートごとに、内部でどのVLANに接続するかを指定するのが「ポートVLAN」です。
  - 「ポートVLAN」に設定されたポートに接続する機器は、VLAN用の特別な設定は不要です。
- スイッチ間をカスケード接続する際は、接続ポートを「タグVLAN」に設定します。
  - このポートを出入りするパケットには、どのVLANに行くのかを示す「VLANタグ」が付けられます。



LinuxサーバのVLAN接続方法

- LinuxサーバをVLANに接続するには、2種類の方法があります。
  - ポートVLANを設定したスイッチポートに接続する。
    - Linuxサーバ側では特別な設定は不要です。
    - ポートVLANで設定されたVLANのみと通信を行います。
  - タグVLANを設定したスイッチポートに接続する。
    - Linuxサーバ側で、タグの付け外しを行う「VLANデバイス」を構成する必要があります。
    - 複数のVLANデバイスを構成することで、複数のVLANと通信することも可能です。



ifcfg-eth0

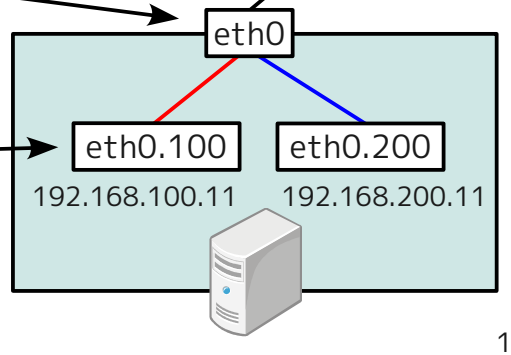
```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
```

物理NIC

ifcfg-eth0.100

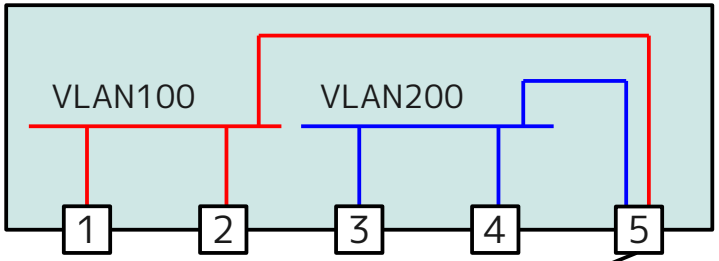
```
DEVICE=eth0.100
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.100.11
NETMASK=255.255.255.0
```

VLANデバイス



サーバ仮想化におけるVLAN接続

- サーバ仮想化環境では、仮想ブリッジ（仮想スイッチ）を経由して、仮想マシンをVLANデバイスに接続することで、仮想マシンをVLANに接続します。
  - 仮想マシンのゲストOS上で「仮想NIC」に対して、IPアドレスを設定します。
  - VLANタグの付け外しはVLANデバイスが行うので、ゲストOSでは、VLANの存在を意識する必要はありません。



ifcfg-eth0　物理NIC

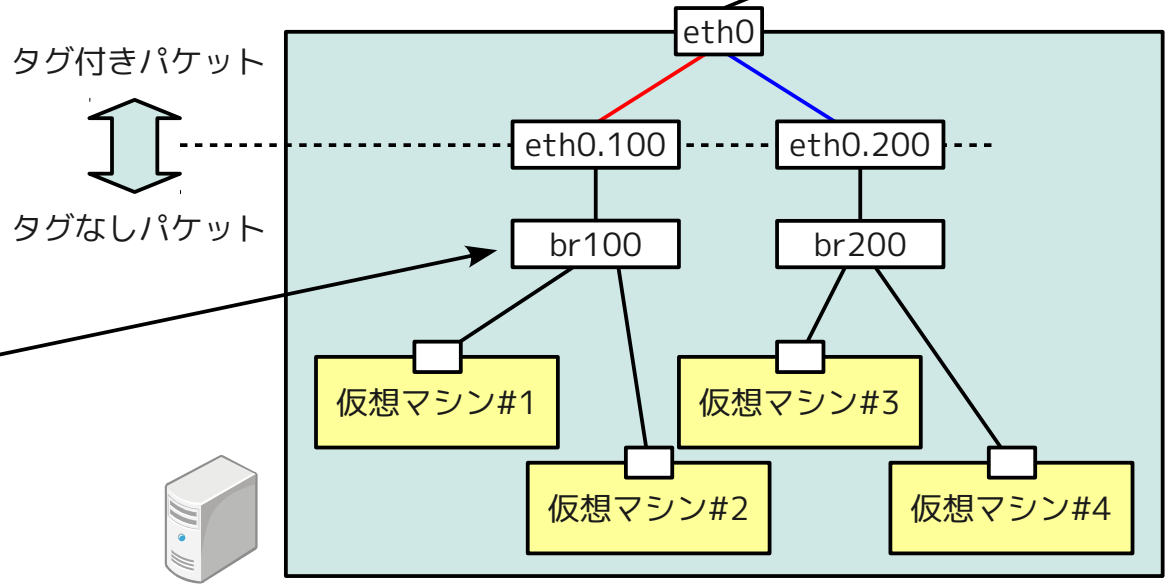
```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
```

ifcfg-eth0.100　VLANデバイス

```
DEVICE=eth0.100
ONBOOT=yes
BOOTPROTO=none
BRIDGE=br0
```

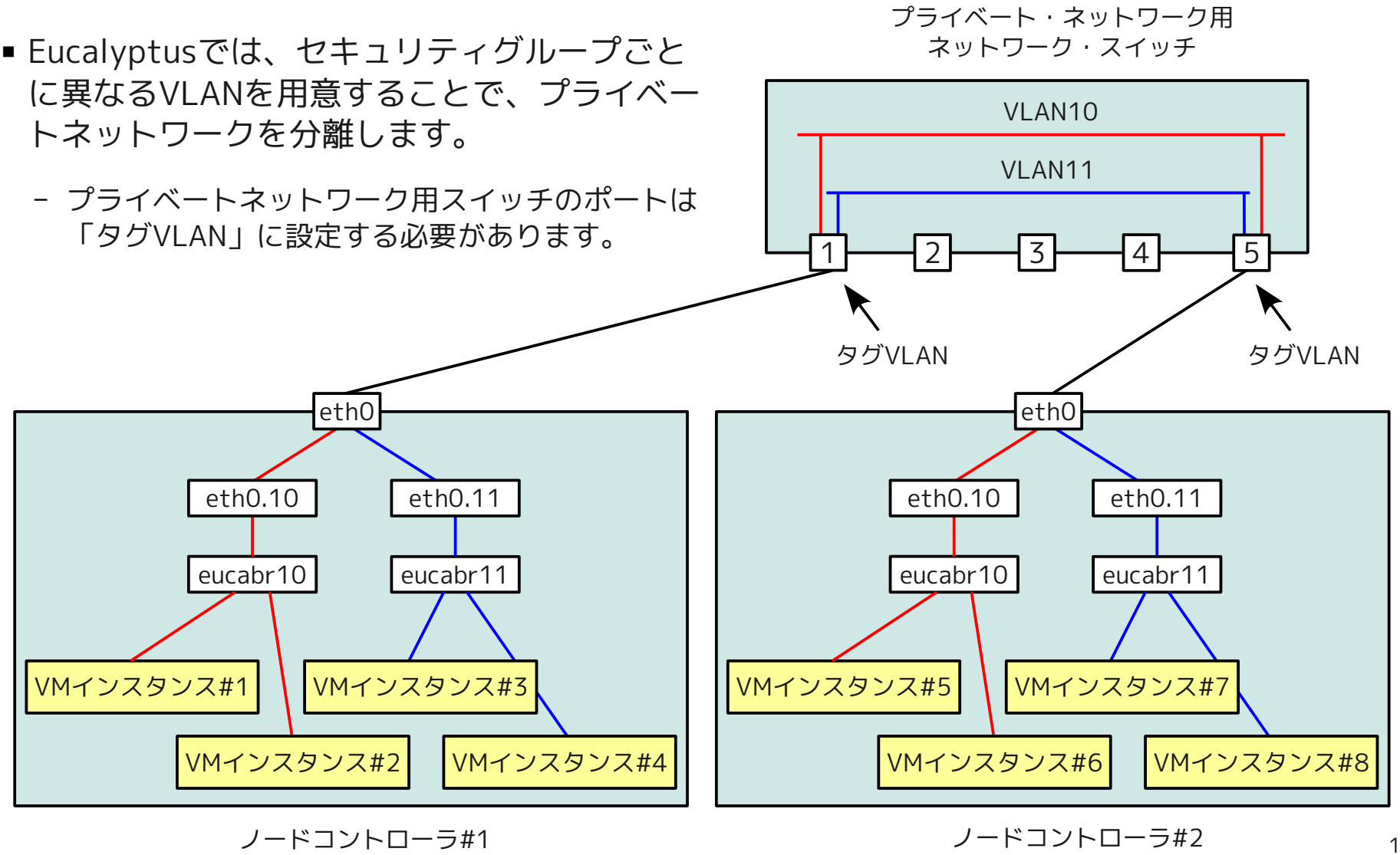
ifcfg-br100　仮想ブリッジ

```
DEVICE=br100
ONBOOT=yes
TYPE=Bridge
BOOTPROTO=none
```



（参考）EucalyptusにおけるVLANの利用法

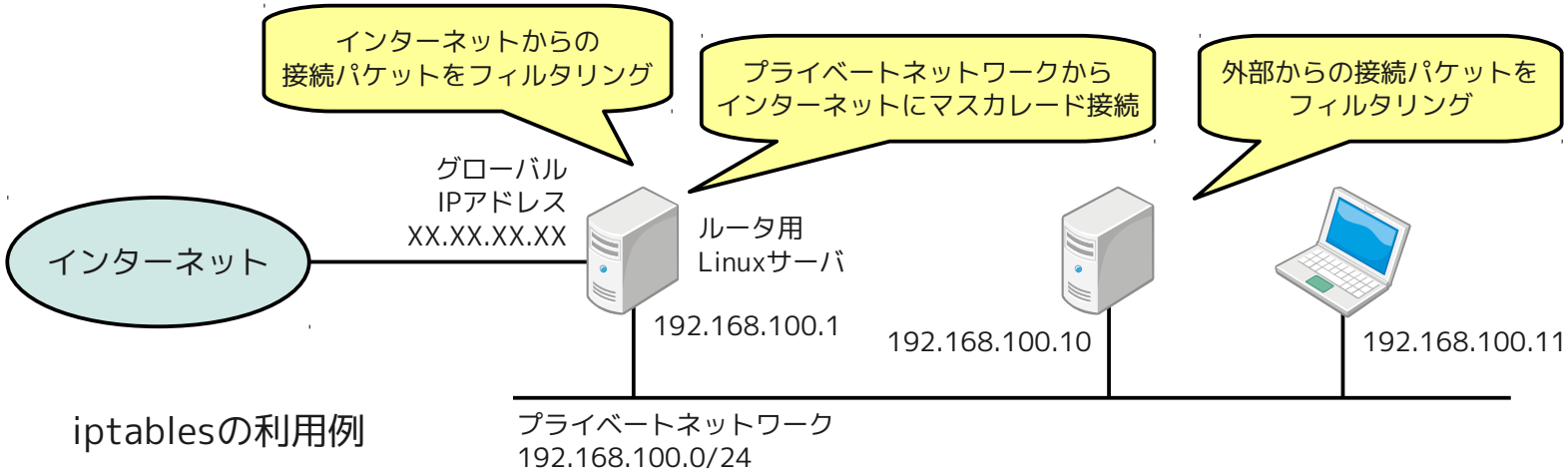
- Eucalyptusでは、セキュリティグループごとに異なるVLANを用意することで、プライベートネットワークを分離します。
  - プライベートネットワーク用スイッチのポートは「タグVLAN」に設定する必要があります。



iptablesの基礎とパケットフィルタリング

iptablesの機能

- iptablesの機能は、大きくは「パケットフィルタリング」と「NAT (Network Address Translation)」に分かれます。
  - パケットフィルタリングは、Linux上にファイアウォール機能を提供するもので、IPパケットの送信、受信、転送のそれぞれについて、許可条件を設定します。
  - NATは、Linuxサーバをルータとして使用する際に利用する機能で、パケットの転送時に、パケットに含まれる送信元、もしくは宛先IPアドレスを変換します。
    - 正確には、DNAT（宛先アドレス変換）、SNAT（送信元アドレス変換）、マスカレード（SNATの特別版）の3種類があります。それぞれの詳細はこの後で説明します。

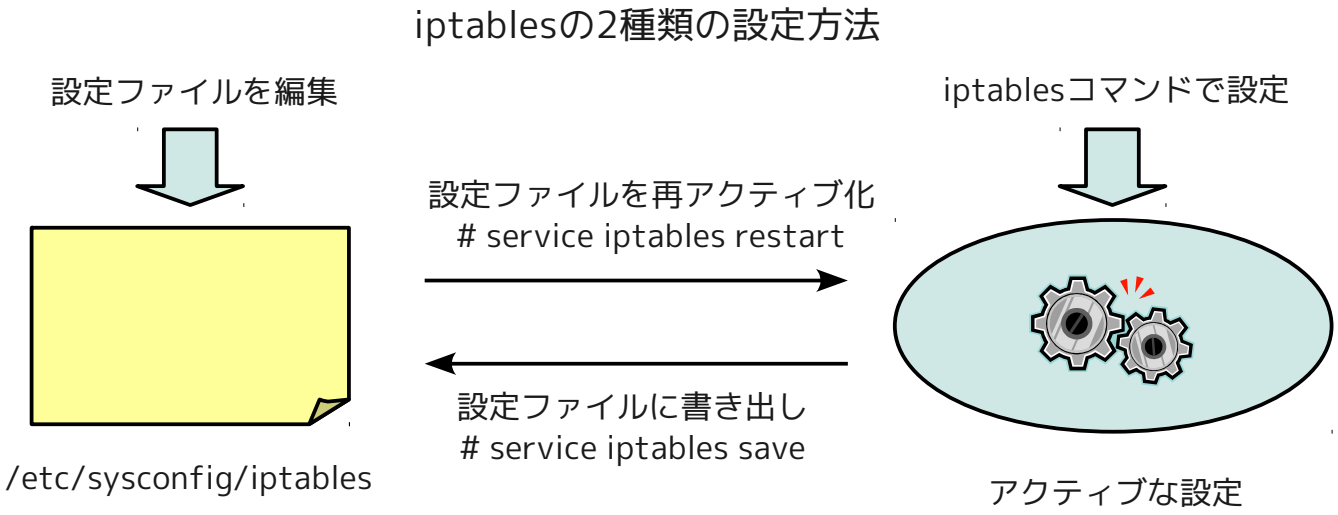


iptablesの利用例

iptablesの設定方法

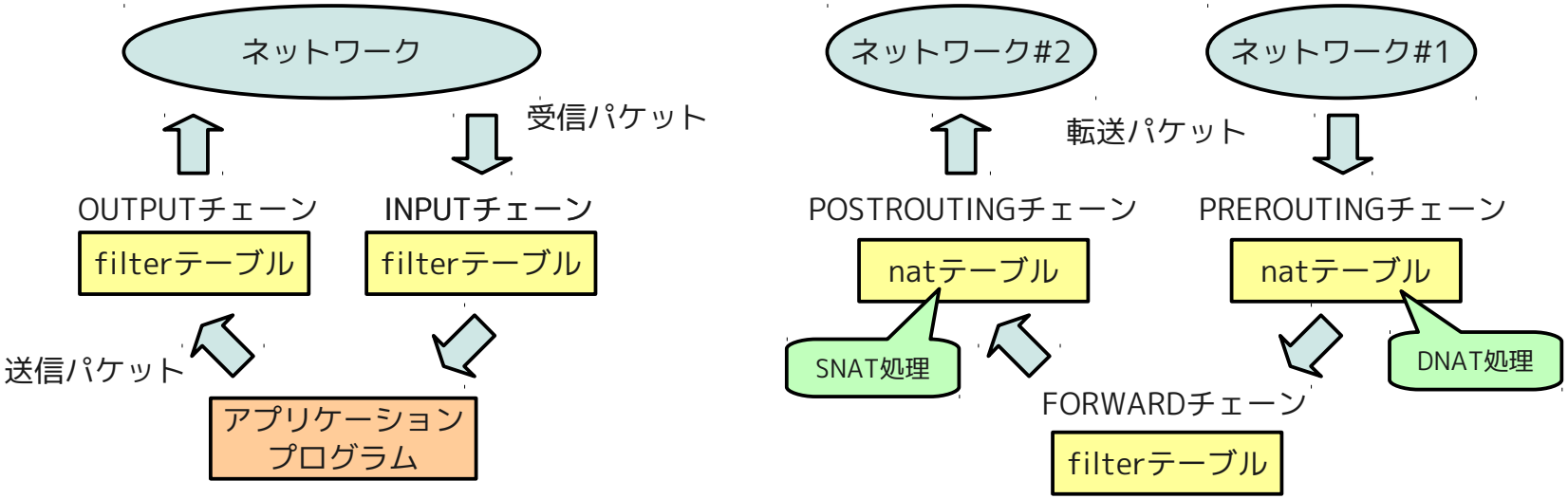
- iptablesの有効化／無効化は、iptablesサービスの起動・停止で行います。
  - # service iptables start：設定ファイル「/etc/sysconfig/iptables」をアクティブ化します。
  - # service iptables stop：アクティブな設定がすべて無効になります。

- iptablesの設定は、設定ファイルを編集して再アクティブ化する方法と、iptablesコマンドでアクティブな設定を直接変更する方法があります。
  - アクティブな設定を直接変更した場合は、別途、設定ファイルに内容を書き出しておかないと、再起動後に変更が失われます。



iptablesにおける処理の流れ

- Linuxサーバを通過するパケットは、いくつかの「チェーン」を通過していきます。
  - INPUTチェーン：受信パケットが通過します。
  - OUTPUTチェーン：送信パケットが通過します。
  - PREROUTING、FORWARD、POSTROUTINGチェーン：転送パケットが通過します。
- それぞれのチェーンでは、次のテーブルの設定に従ってパケットの処理が行われます。
  - filterテーブル：パケットフィルタリングの処理を定義します。
  - natテーブル：NATの処理を定義します。



パケットフィルタリングの設定例

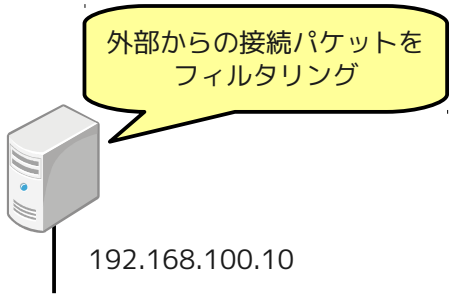
- パケットフィルタリング設定の基本コマンドは次の通りです。
  - 指定条件にマッチするパケットに対するターゲットの定義
    - iptables -A <チェーン> <パケット条件> -j <ターゲット>
  - デフォルトターゲットの定義
    - iptables -P <チェーン> <ターゲット>

ターゲット	説明
ACCEPT	パケットの送受信を許可
DROP	パケットを破棄する
REJECT	パケットの拒否をICMPで通知
LOG	パケット情報をSyslogに出力

  - -Aオプションで設定した順にパケット条件の評価が行われて、最初にマッチした条件に対するターゲットが実行されます。LOGアクション以外はそこで評価が終了します。
  - どの条件にもマッチしなかった場合は、-Pオプションで指定したアクションが実行されます。
  - 現在のアクティブな設定は次のコマンドで表示します。
    - iptables [-v][-n] -L <チェーン>
    - -vオプションは詳細情報を表示します。-nオプションはIPアドレスやTCP/UDPポート番号を数値で表示します。（指定しない場合は名前解決を行います。）

受信パケットに対するフィルタリング設定手順の例

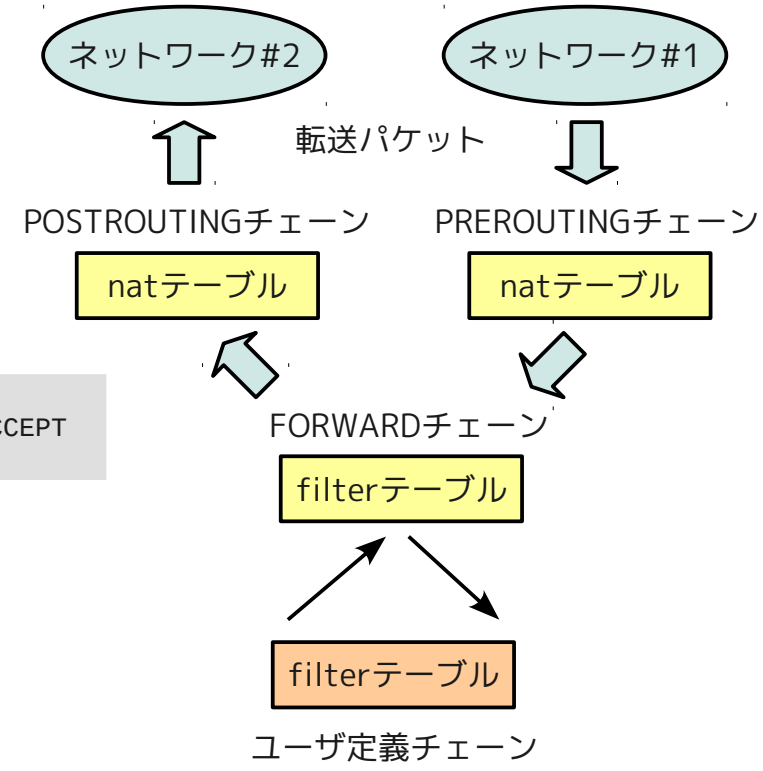
```
# service iptables stop
# iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
# iptables -A INPUT -i lo -j ACCEPT
# iptables -A INPUT -d 192.168.100.10 -p tcp -m tcp --dport 22 -j ACCEPT
# iptables -A INPUT -j LOG -m limit --log-prefix "[INPUT Dropped] "
# iptables -P INPUT DROP
# service iptables save
# service iptables start
```



（参考）ユーザ定義チェーンの利用

- デフォルトで用意されたチェーンの他に任意の名前のチェーンを追加で定義することができます。これを「ユーザ定義チェーン」と呼びます。
  - -jオプションのターゲット名にユーザ定義チェーンを指定すると、そのパケットは該当のユーザ定義チェーンに指定した処理が行われます。
    - これは「サブルーチン」を呼び出すイメージになります。ユーザ定義チェーン内の評価にマッチしなかったパケットは、呼び出し元のチェーンに戻って評価を継続します。
  - 次は、SSH接続を許可する条件を設定した「SSH_OK」チェーンを定義して、FORWARDチェーンから呼び出す例です。

```
# iptables -N SSH_OK
# iptables -A SSH_OK -p tcp -p tcp -m tcp --dport 22 -j ACCEPT
# iptables -A FORWARD -j SSH_OK
```



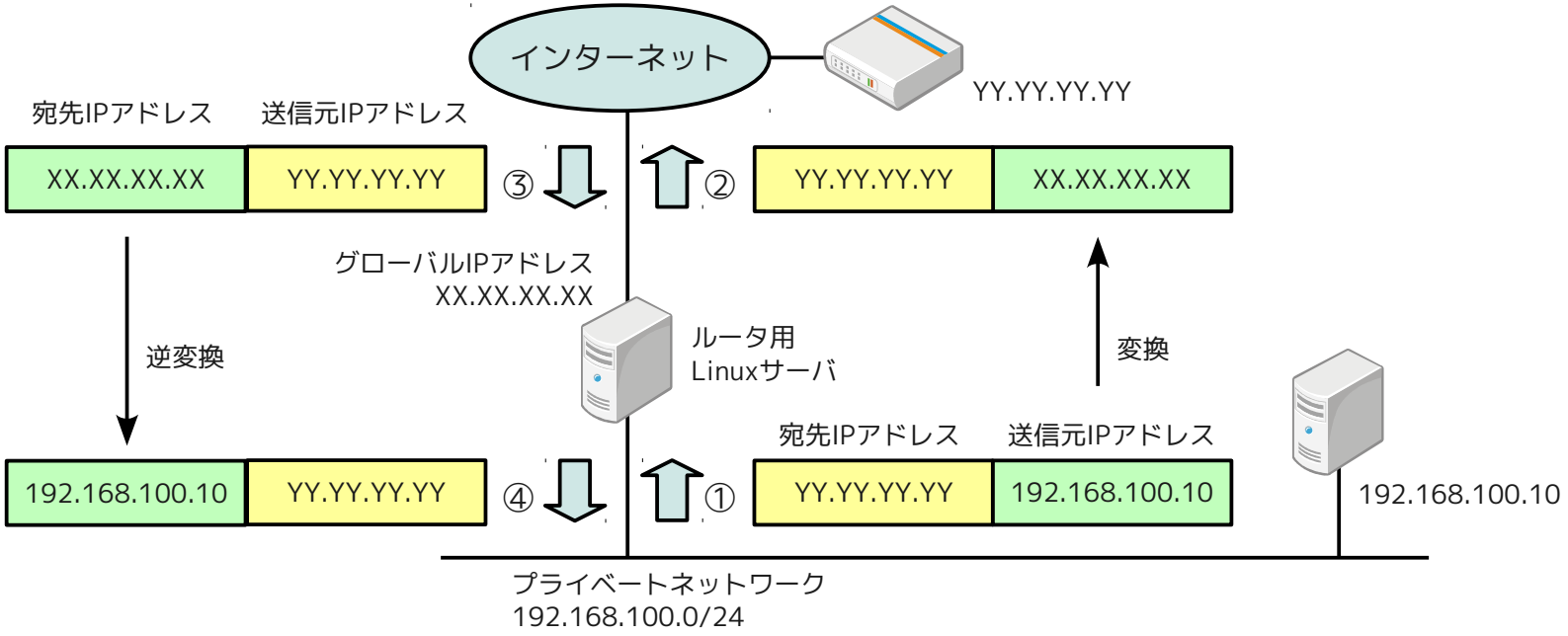
メモとしてお使いください

メモとしてお使いください

iptablesによるNAT処理

SNAT/マスカレードの仕組み

- 下図は、SNAT/マスカレードによる送信元IPアドレス変換の例です。
  - プライベートネットワークのIPアドレス「192.168.100.10」のサーバからインターネットのIPアドレス「YY.YY.YY.YY」のサーバにパケットが送信されます（①）。
  - ルータ用Linuxは、送信元IPアドレスを自身のグローバルIP「XX.XX.XX.XX」に書き換えて、インターネットにパケットを転送します（②）。
  - インターネットから返送されたパケット（③）は、宛先IPアドレスを元の「192.168.100.10」に逆変換して、プライベートネットワークのサーバに送り返します（④）。

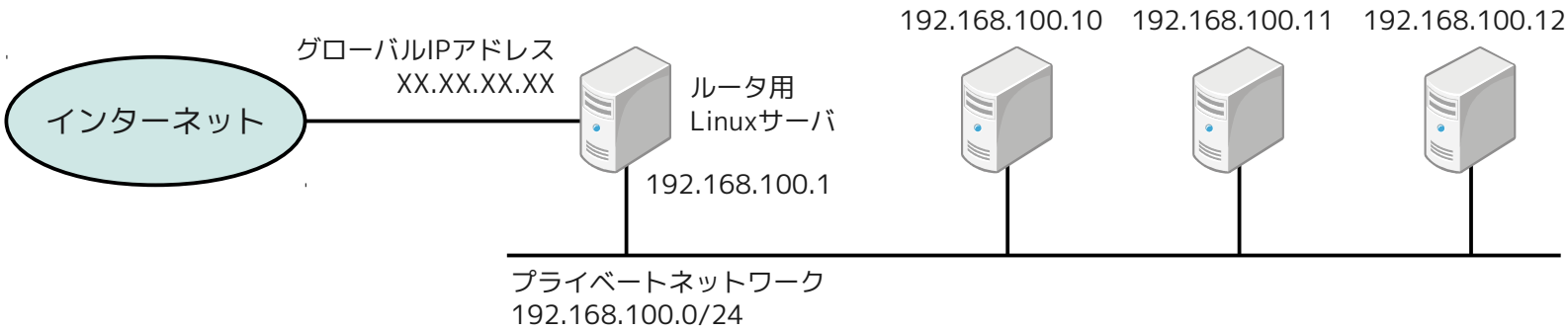


複数サーバのSNAT処理について

- 複数サーバのプライベートIPアドレスを共通のグローバルIPに変換する際は、逆変換の際に、変換先のプライベートIPアドレスを正しく判別する必要があります。
- iptablesは、送信元IPアドレスと送信元ポート番号のペアを記憶しておき、返送パケットの送信先ポート番号から、逆変換先のIPアドレスを決定します。
  - 複数サーバが同じ送信元ポート番号を使った場合、SNAT処理において、送信元ポート番号も変換します。
  - ただし、大量のサーバが同時にSNATを利用すると、変換先の送信元ポート番号が不足して、SNAT処理ができなくなる場合があります。

SNAT記憶テーブルの例

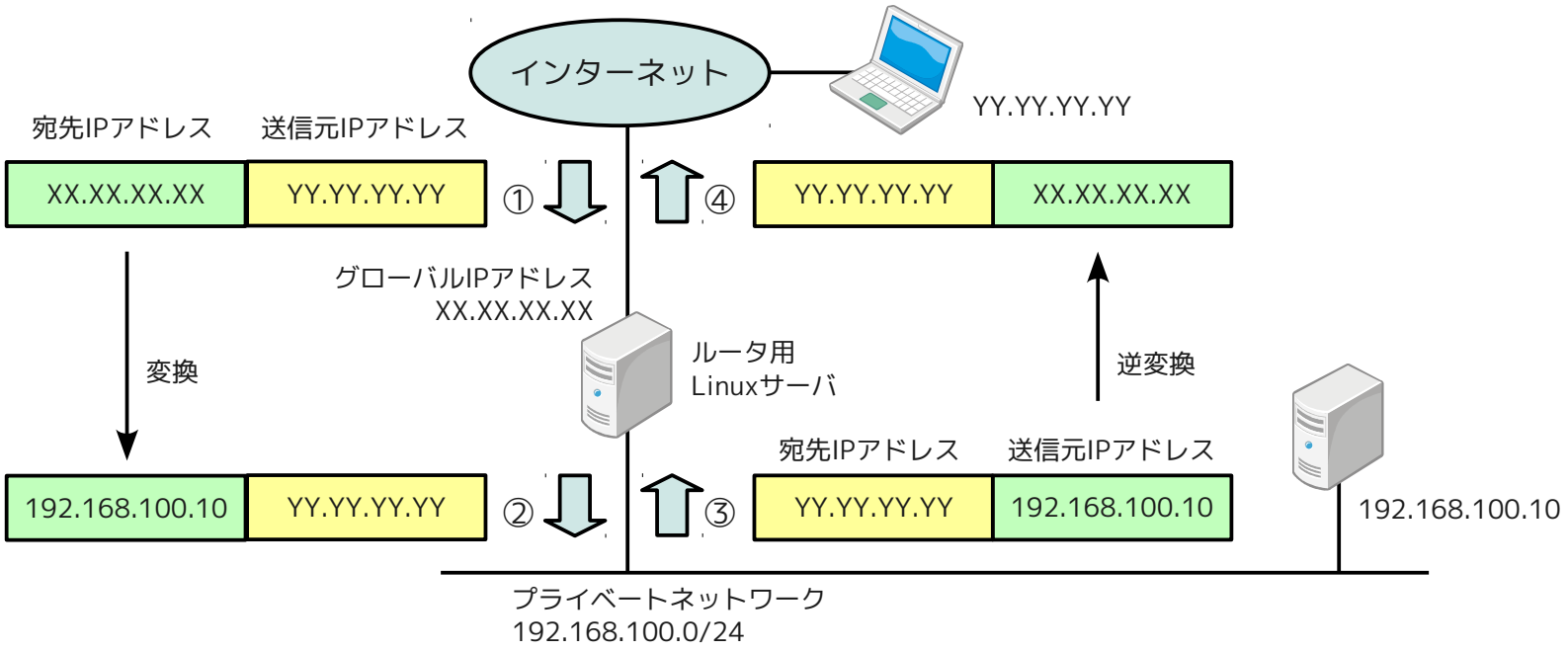
送信元IPアドレス	送信元ポート番号	変換後の送信元ポート番号
192.168.100.10	3021	3021
192.168.100.11	4008	4008
192.168.100.12	3021	9000



- SNAT対象のサーバごとに、個別のグローバルIPアドレスを割り当てて使用することも可能です。

DNATの仕組み

- 下図は、DNATによる宛先IPアドレス変換の例です。
  - インターネットのIPアドレス「YY.YY.YY.YY」のPCからルータ用Linuxサーバのグローバルアドレス「XX.XX.XX.XX」にパケットが送信されます（①）。
  - ルータ用Linuxは、宛先IPアドレスをプライベートネットワークのサーバのIPアドレス「192.168.100.10」に書き換えて、パケットを転送します（②）。
  - プライベートネットワークのサーバから返送されたパケット（③）は、送信元IPアドレスを「XX.XX.XX.XX」に逆変換して、インターネットのPCに送り返します（④）。



NATの設定例 (1)

- NATの設定は、「-t nat」オプションでnatテーブルを指定します。SNAT/マスカレードとDNATで使用するチェーンが異なります。
  - SNAT/マスカレードはPOSTROUTINGチェーンを使用します。マスカレードでは、変換IPアドレスには、ルータのNICのIPアドレスが自動的に使用されます。
    - iptables -t nat -A POSTROUTING <パケット条件> -j SNAT --to-source <変換IPアドレス>
    - iptables -t nat -A POSTROUTING <パケット条件> -j MASQUERADE
  - DNATは、PREROUTIGNチェーンを使用します。
    - iptables -t nat -A PREROUTING <パケット条件> -j DNAT --to-destination <変換IPアドレス>

- SNAT/マスカレードの設定例
  - 変換元と変換先のIPを1対1で指定して、SNAT変換を行います。

```
# iptables -t nat -A POSTROUTING -s 192.168.100.10 -j SNAT --to-source XX.XX.XX.10
# iptables -t nat -A POSTROUTING -s 192.168.100.11 -j SNAT --to-source XX.XX.XX.11
# iptables -t nat -A POSTROUTING -s 192.168.100.12 -j SNAT --to-source XX.XX.XX.12
```

  - プライベートネットワーク「192.168.100.0/24」から、他のネットワークへの通信をマスカレードで変換します。

```
# iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -j MASQUERADE
```

    - マスカレード処理では変換先のIPアドレスは指定しません。パケットを送出するNICのIPアドレスが自動的に選択されます。

NATの設定例 (2)

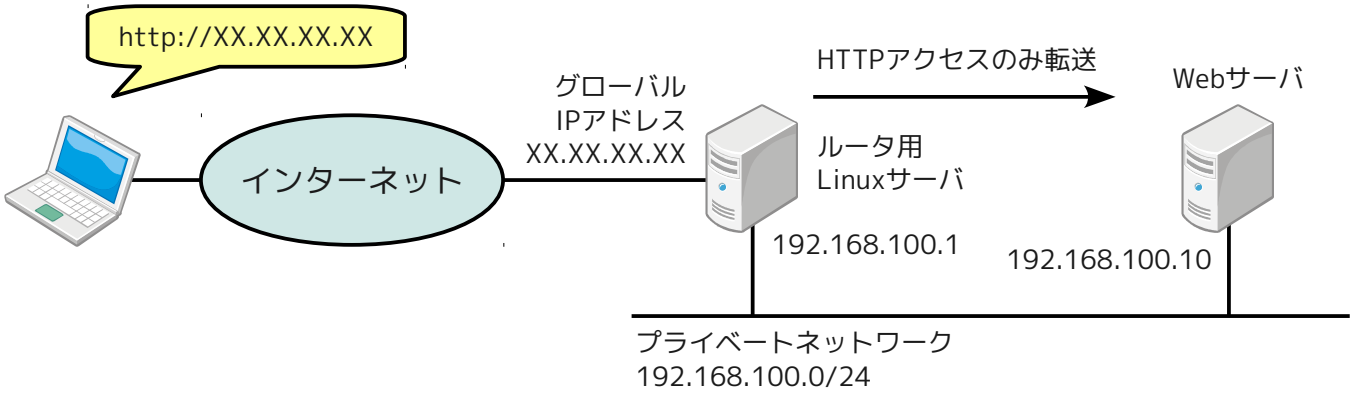
- DNATの設定例
  - 前ページのSNAT変換表と同じ変換をDNATでも行います。

```
# iptables -t nat -A PREROUTING -d XX.XX.XX.10 -j DNAT --to-destination 192.168.100.10
# iptables -t nat -A PREROUTING -d XX.XX.XX.11 -j DNAT --to-destination 192.168.100.11
# iptables -t nat -A PREROUTING -d XX.XX.XX.12 -j DNAT --to-destination 192.168.100.12
```

    - SNATだけの場合、プライベートネットワークのサーバからインターネットへの接続はできますが、インターネットからプライベートネットワークのサーバへの接続はできません。前ページのSNATと合わせて、DNATを設定することで、プライベートネットワークとインターネットの双方向通信が可能になります。
  - インターネットからグローバルIPアドレス「XX.XX.XX.XX」にHTTPでアクセスすると、プライベートネットワークのサーバ「192.168.100.10」に転送して応答を返します。

```
# iptables -t nat -A PREROUTING -d XX.XX.XX.XX -p tcp --dport 80 -j DNAT --to-destination 192.168.100.10
```

    - 宛先ポートがTCP80番のパケットだけを選択して、「192.168.100.10」のWebサーバに転送しています。



メモとしてお使いください

メモとしてお使いください

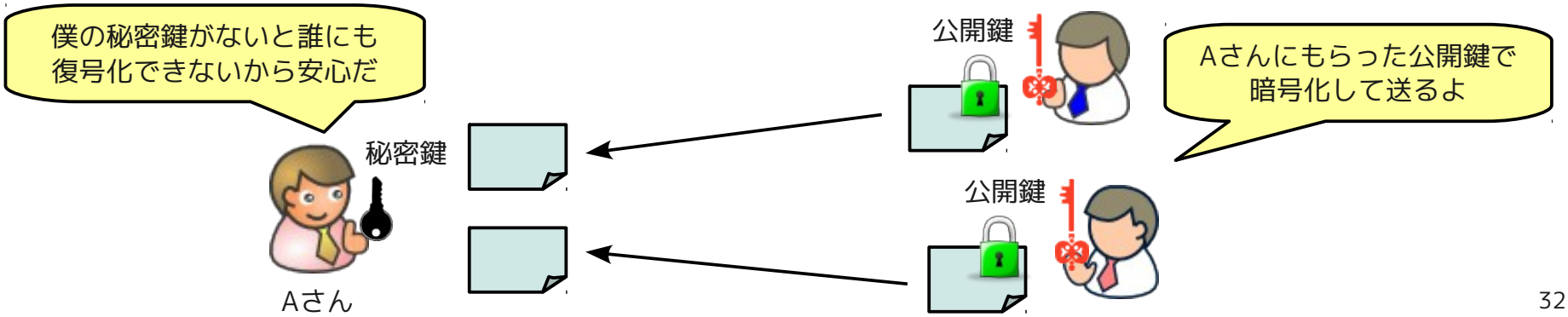
鍵ペアによるSSH認証

鍵ペアによる認証の基礎

- ネットワーク上の認証には、ユーザ認証とサーバ認証があります。
  - ユーザ認証：接続を許可するサーバから見て、本物のユーザであることを確認する。
  - サーバ認証：接続するユーザから見て、本物のサーバであることを確認する。

- これらの認証には、主に非対称鍵暗号を利用した方法が用いられます。
  - 非対称鍵暗号では、暗号化に使う鍵と復号化に使う鍵が異なります。
  - 非対称鍵暗号では、鍵の所有者だけが保管するべき秘密鍵と、万人に配布しても構わない公開鍵を利用します。秘密鍵は誤って他人の手に渡らないように管理する必要があります。

- 下図は、事前に配布した公開鍵で暗号化したメッセージを送ってもらう例です。このメッセージは、秘密鍵を持つ自分だけが復号化して読むことができます。

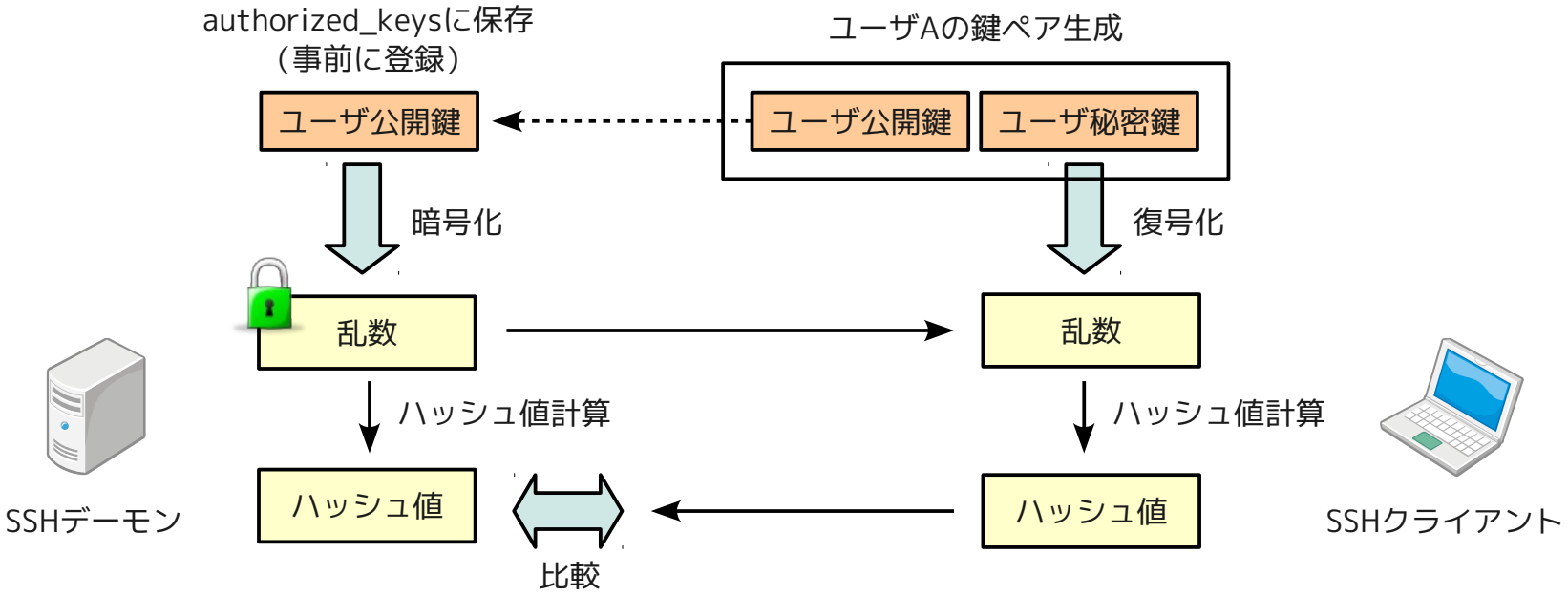


対称鍵暗号について

- 暗号化と復号化に同じ鍵（秘密鍵）を使う対象鍵暗号は、非対象鍵暗号と比較して、次のような特徴があります。
  - 暗号化／復号化の計算量が少なく、高速な処理が可能です。
  - 秘密鍵を共有する必要があるので、相手側から鍵の情報がもれないための仕組みが必要です。
- SSH接続では、ユーザ認証／サーバ認証には非対称鍵暗号を用いますが、通信内容の暗号化には、対象鍵暗号を使用します。
  - 非対称鍵による認証が成功すると、その後の通信を暗号化するための対象鍵（セッション鍵）を生成して利用します。SSH接続が終了するとセッション鍵は破棄されます。
- Kerberos認証、Windows AD認証では、対象鍵暗号のみを使用します。
  - 秘密鍵を集中管理するための専用の「認証局」を用いることで、ユーザとサーバで鍵を共有することなく安全に認証を行います。
  - 歴史的に非対称鍵暗号化の研究が進んでいなかった時代に開発された技術ですが、ネットワーク的に閉じた環境での認証方式として、現在でも使用されています。

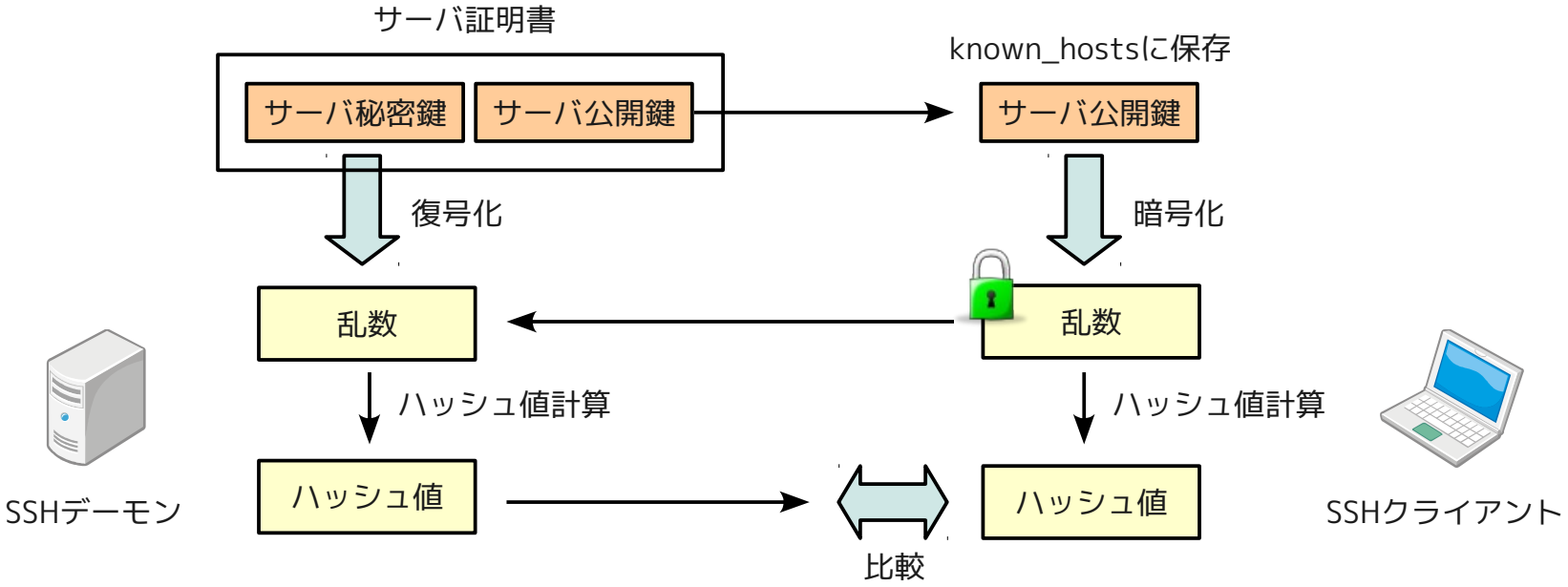
鍵ペアによるSSHユーザ認証の仕組み

- 公開鍵認証でSSH接続する際は、次の流れでユーザ認証が行われます。
  - SSH接続するユーザの鍵ペアを事前に生成して、公開鍵を接続先サーバに登録します。
  - SSH接続を受け付けたサーバ（SSHデーモン）は、乱数を発生してユーザの公開鍵で暗号化したものをクライアント（SSHクライアント）に送信します。
  - クライアントは対応する秘密鍵で復号化して、そのハッシュ値を返答します。
  - サーバは元の乱数のハッシュ値を計算して、クライアントの返答と一致することを確認します。



鍵ペアによるSSHサーバ認証の仕組み

- SSH接続の際は、ユーザから見て接続先のサーバが本物であることを確認する、サーバ認証も行われます。
  - クライアントはサーバからサーバ証明書の公開鍵を受け取って保存します。
    - 同じサーバから以前と異なる公開鍵を受け取った場合は、そこで処理を中断します。
  - クライアントは乱数をサーバの公開鍵で暗号化してサーバに送り、サーバは対応する秘密鍵で復号化して、そのハッシュ値を返送します。
  - クライアントは元の乱数のハッシュ値を計算して、サーバの返答と一致することを確認します。



SSH公開鍵認証の設定手順

- 次は、SSHクライアント上でユーザの鍵ペアを作成して、サーバ「server01」にユーザ「user01」としてログインするための公開鍵登録を行う例です。

```
# ssh-keygen -N ""
# ssh-copy-id user01@server01
```

  - ssh-keygenコマンドで鍵ペアを作成して、クライアントの「~/.ssh/id_rsa」（秘密鍵）および「~/.ssh/id_rsa.pub」（公開鍵）に保存します。
    - -Nオプションは秘密鍵を使用する際に入力するパスフレーズを指定します。これは、秘密鍵が盗難にあった場合などの安全対策となります。シェルスクリプトによるバッチ処理などで自動接続を行う場合は、パスフレーズなしの鍵ペアが必要です。
  - ssh-copy-idコマンドで、接続先サーバの「~/.ssh/authorized_keys」ファイルに公開鍵を登録します。同時に、接続先サーバのサーバ証明書（サーバ公開鍵）をダウンロードして、クライアント側の「~/.ssh/known_hosts」に登録します。
    - どちらも1行に1つの公開鍵が記載されたテキストファイルですので、テキストエディタで編集可能です。

~/.ssh/known_hostsの例

```
server01 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAnakSHxbAkQ0oDML1aYNyKS/2aqIkVDSKfpr+KKR/VoICon9HlX
0TS2oks1oqt/VYt8EFsRjyQgkXjd+hoU71rea+9Wvbya5/54nQeqtyqnlNKFkNMG6r4ewrpjOv1DccYLF5bFouXig1SuYM5
168QiSW7pjBDFNRKKBMGa3koGpaX1PCBybUwH6aCZe3MYF4BR9zYzwePI3/d6w0Y+PX/3TWEadsdXdncrbLguVqyVbD5QJ2
ZQIHV9d8zO1IxjjeSL15/0Jgm+VtUkt5PntZ9UAm/RDXe8lHmd6S05ieTqfEUpqiiblSwg08HnpgCDrjn/74pi/XJw7mD3w
AaYR3xQ==
10.7.7.55 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA3K0ppAgND9dVJxQmvLUCaSBblwSDr7xHa4ePuUtdBVXRdjIiY
o9EVNEv0oQDfzrQICJH6HKwf06etJogYiNBc4UOv4OWLEQqoXES56BceD+Sx3uenel7N8T1bjIQfW4gpH1nwC5BHyaR71dM
Bpr9nfbhp2NlXZS6Ar9Sd8PiXw/3u6jPa0UiwFB48smLpKxWxqQotWsvWSiUtGPEQBz8FoiDKY2ENECeW8i1LBKJA44rxyB
VD9JTFB4WwZTbvyO9dCCpF6tOVT9vptFTVALm5m2cpKRYg2AU8gRAZo3tmP1QaFZhlSVaJSYoaVj8X+bPJ4bA462B83PHdm
XrDh/tkw==
```

SSH公開鍵認証を使用する際の注意点

- 新しいサーバに初めて接続する際は、サーバ証明書の受け入れ確認メッセージが表示される場合があります。

新しいサーバ証明書の受け入れ確認メッセージ

```
# ssh root@server01
The authenticity of host 'server01 (XXX.XXX.XXX.XXX)' can't be established.
RSA key fingerprint is 7c:e6:a3:78:a6:54:a8:4b:f2:87:0b:d5:5d:66:9c:d9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'server01' (RSA) to the list of known hosts.
```

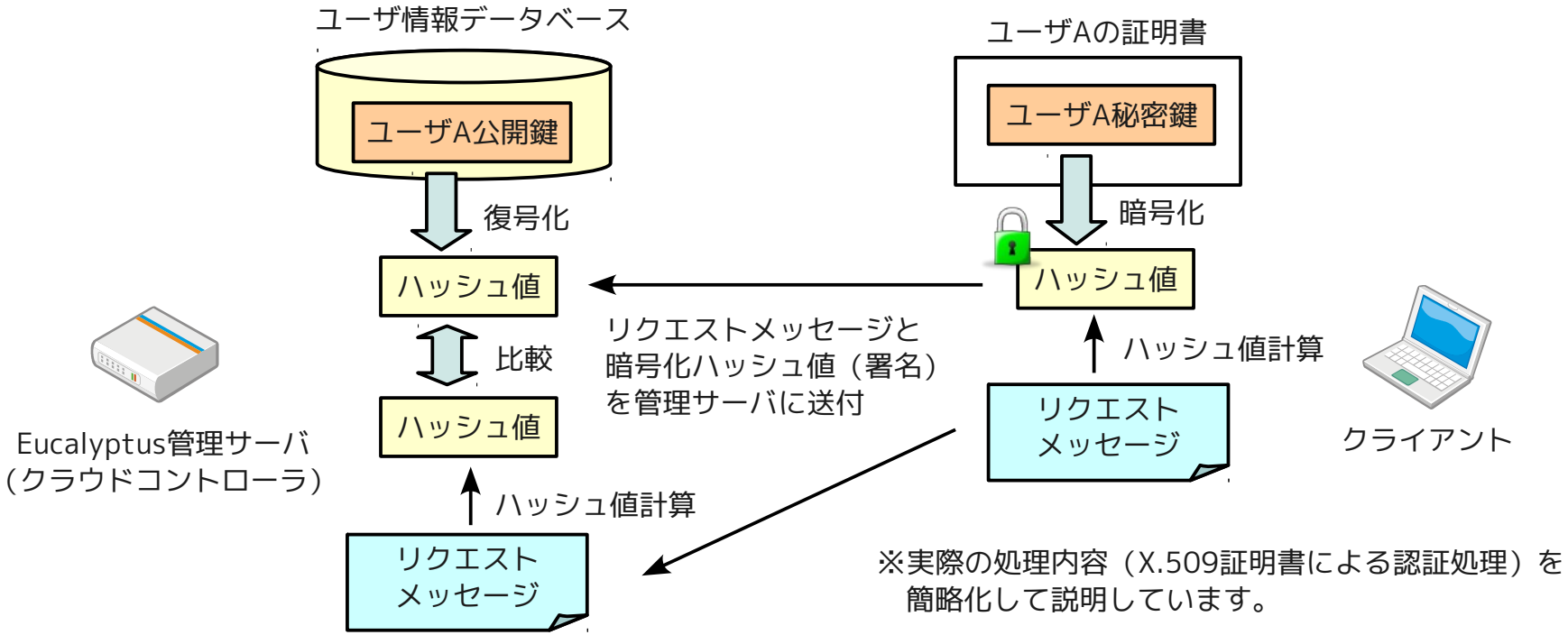
- 以前に接続したことのあるサーバで、前回と異なるサーバ証明書（サーバ公開鍵）を受け取った場合、偽者のサーバの可能性があるためにクライアントは接続を拒否します。
  - 何らかの理由で、公開鍵が変更されたことがわかっている場合は、「~/.ssh/known_hosts」の該当のエントリを削除して、再度、SSH接続します。

サーバ証明書が前回と異なる場合の警告メッセージ

```
# ssh root@server01
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
e3:d4:87:e2:5b:34:21:8b:7b:40:7a:41:93:91:06:d0.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending key in /root/.ssh/known_hosts:1
RSA host key for server01 has changed and you have requested strict checking.
Host key verification failed.
```

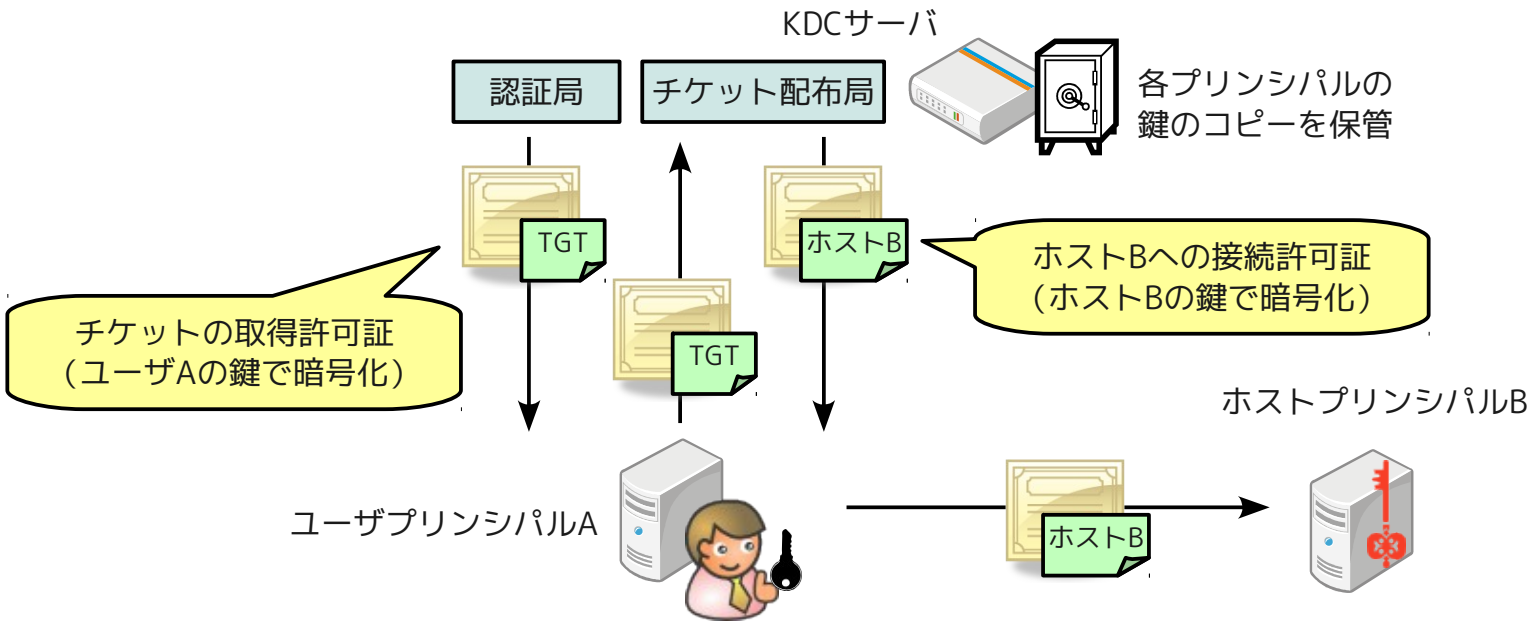
EucalyptusのSOAP APIユーザ認証

- クライアントは、リクエストメッセージのハッシュ値を秘密鍵で暗号化して送ります。サーバは、対応する公開鍵で復号化して正しいハッシュ値であることを確認します。
  - ユーザAの公開鍵で復号化できる暗号を生成できるのは、対応する秘密鍵を持ったユーザAだけなので、メッセージの送信元が本物のユーザAであることが分かります。
  - メッセージの暗号化ハッシュ値を添付することを「メッセージに署名する」と言います。メッセージが改竄されるとハッシュ値が合わなくなるので、メッセージ改竄を防止する効果もあります。



（参考）Kerberosによるサーバ接続認証

- Kerberosでは、対象鍵暗号のみを使用します。ユーザAがホストBに接続する際の認証はつぎのようになります。
  - ユーザAは、「TGT（チケット取得許可証）」をKDCサーバから取得します。
    - TGTはユーザAの鍵で暗号化されており、本物のユーザAしか使用できません。
  - ユーザAは、TGTをKDCに提示して、ホストBへの接続チケット（接続許可証）を取得します。
    - 接続チケットはホストBの鍵で暗号化されており、本物のホストBしか内容を確認できません。
  - ユーザAは、接続チケットをホストBに提示して、ホストBへの接続を開始します。



メモとしてお使いください

メモとしてお使いください

メモとしてお使いください

参考資料: iptablesのパケットマッチング条件

プロトコルに依存しないパケットマッチング (1)

- 右表は、TCP/UDP/ICMPなどのプロトコルの種類に依存しないマッチング条件です。
  - 「オプション」に複数の表記が記載されているものは、すべて同じ意味です。

オプション	マッチング条件
-s, --src, --source	送信元IPアドレス
-d, --dst, --destination	宛先IPアドレス
-i, --in-interface	パケットを受信したインターフェース
-o, --out-interface	パケットを送信するインターフェース
-p, --protocol	通信プロトコル（TCP/UDP/ICMP）
--state	コネクション・トラッキングの状態（「-m state」の直後に指定）

- 送信元IPアドレス/宛先IPアドレスによるマッチング条件
  - 特定のIPアドレスを指定すると、そのIPアドレスにマッチします。
    - -s 192.168.1.20　　⇒ 送信元IPアドレスが192.168.1.20のパケットにマッチ
  - ネットワークアドレスを表記すると、その範囲のIPアドレスにマッチします。
    - -s 192.168.1.0/24 (-s 192.168.1.0/255.255.255.0)

      ⇒ 送信元IPアドレスが192.168.1.0～192.168.1.255のパケットにマッチ
  - 否定条件の例です。
    - ! -s 192.168.1.20　　⇒ 送信元IPアドレスが192.168.1.20以外のパケットにマッチ
    - ! -s 192.168.1.0/24　　⇒ 送信元IPアドレスが192.168.1.0～192.168.1.255以外のパケットにマッチ
  - 上記の例はすべて「-s」を「-d」に変えると宛先IPアドレスに対する条件になります。

プロトコルに依存しないパケットマッチング (2)

- パケットを送受信するインターフェースによるマッチング条件
  - インターフェースのデバイス名を指定します。
    - -i eth0　　⇒ eth0で受信したパケットにマッチ
  - VLANを使用する場合は、VLANインターフェースを明示的に指定します。
    - -i eth0.101 ⇒ eth0.101（VLAN101）で受信したパケットにマッチ
  - 「+」は、任意の末尾を表します。これにより、任意のVLAN番号のVLANインターフェースにマッチすることが可能です。
    - -i eth0.+　　⇒ eth0の任意のVLANインターフェースで受信したパケットにマッチ
  - 否定条件の例です。
    - ! -i eth0　　⇒ eth0以外で受信したパケットにマッチ
  - 上記の例はすべて「-i」を「-o」に変えると送信インターフェースに対する条件になります。

プロトコルに依存しないパケットマッチング (3)

- 通信プロトコル（TCP/UDP/ICMP）によるマッチング条件
  - TCP/UDP/ICMPのプロトコルによるマッチングを行います。
    - -p tcp ⇒ TCPパケットにマッチ
    - -p udp ⇒ UDPパケットにマッチ
    - -p icmp ⇒ ICMPパケットにマッチ
  - この後で説明する特定プロトコルに固有のマッチング条件は、これらと同時に指定する必要があります。

- コネクショントラッキングの状態によるマッチング条件
  - stateモジュールによる拡張マッチング機能のため、先に「-m state」オプションを指定する必要があります。
    - -m state --state NEW ⇒ ステータスがNEWのパケットにマッチ
    - -m state --state ESTABLISHED ⇒ ステータスがESTABLISHEDのパケットにマッチ
    - -m state --state RELATED ⇒ ステータスがRELATEDのパケットにマッチ
    - -m state --state INVALID ⇒ ステータスがINVALIDのパケットにマッチ
  - 複数条件の指定例です。
    - -m state --state NEW,RELATED ⇒ ステータスがNEW、もしくはRELATEDのパケットにマッチ

プロトコルに依存するパケットマッチング (1)

- 下表は、特定のプロトコルに固有のマッチング条件です。
  - 対応するプロトコルを指定した-pオプションと同時に使用する必要があります。

オプション	マッチング条件
--icmp-type	ICMPタイプ/コード 「-p icmp」と同時に指定
--sport, --source-port	送信元ポート番号 「-p tcp」もしくは「-p udp」と同時に指定
--dport, --destination-port	宛先ポート番号 「-p tcp」もしくは「-p udp」と同時に指定
--tcp-flags	TCPセッション・フラグ 「-p tcp」と同時に指定

プロトコルに依存するパケットマッチング (2)

- ICMPタイプ/コードによるマッチング条件
  - ICMPパケットについて、特定のタイプ、およびコードによるマッチングを行います。icmpパケットの拡張マッチング機能を使用するため「-m icmp」オプションを指定します。
    - -p icmp -m icmp --icmp-type 8

      ⇒ タイプ8のICMPパケット（Echo Request）にマッチ
    - -p icmp -m icmp --icmp-type 3

      ⇒ タイプ3のICMPパケット（Destination Unreachable）にマッチ（コードは任意）
    - -p icmp -m icmp --icmp-type 3/2

      ⇒ タイプ3/コード2のICMPパケット（Destination Unreachable/Port Unreachable）にマッチ
  - これらは、それぞれ、次のように書いても同じ意味になります。
    - -p icmp -m icmp --icmp-type echo-request
    - -p icmp -m icmp --icmp-type destination-unreachable
    - -p icmp -m icmp --icmp-type port-unreachable
  - タイプとコードを指定する場合、数値では「タイプ番号/コード番号」で指定するのに対して、文字列では、コード部分の文字列のみになります。指定可能な文字列は、次のコマンドで確認します。
    - iptables -p icmp -h

プロトコルに依存するパケットマッチング (3)

- 送信元ポート番号/宛先ポート番号によるマッチング条件
  - TCPおよびUDPパケットについて、送信元ポート番号、および宛先ポート番号でマッチングを行います。TCP/UDPパケットの拡張マッチング機能を使用するため「-m tcp」もしくは「-m udp」オプションを指定します。
    - -p tcp -m tcp --dport 80 ⇒ 宛先ポート番号がTCP80番のパケットにマッチ
    - -p udp -m udp --dport 53 ⇒ 宛先ポート番号がUDP53番のパケットにマッチ
  - ポート番号の範囲で指定することもできます。
    - -p tcp -m tcp --sport 0:1023 ⇒ 送信元ポート番号がTCP0番～TCP1023番のパケットにマッチ
  - 否定条件の例です。
    - -p tcp -m tcp ! --dport 22 ⇒ 宛先ポート番号がTCP22番以外のTCPパケットにマッチ

プロトコルに依存するパケットマッチング (4)

- TCPセッションフラグによるマッチング条件
  - TCPパケットについて「URG、ACK、PSH、RST、SYN、FIN」のフラグでマッチングを行います。TCPパケットの拡張マッチング機能を使用するため「-m tcp」オプションを指定します。
  - 評価対象のフラグと、その中で立っているべきフラグを指定します。
    - -p tcp -m tcp --tcp-flags SYN,ACK,RST SYN,ACK

      ⇒ SYN、ACK、RSTの中でSYNフラグとACKフラグのみが立っているパケットにマッチ
    - この例では、RSTフラグが立っているパケットにはマッチしませんが、その他のフラグの状態はマッチングの結果には影響しません。
  - 評価対象のフラグがすべて立っていないことを示すには「NONE」を用います。
    - -p tcp -m tcp --tcp-flags FIN,RST NONE　⇒ FINフラグとRSTフラグが共に立っていないパケットにマッチ
  - 「--syn」オプションは「TCPセッションを開始するSYNパケット」を示します。次はどちらも「FIN,SYN,RST,ACKの中でSYNのみが立っているパケット」という意味になります。
    - -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN
    - -p tcp -m tcp --syn
  - 否定条件の例です。次はどちらも「FIN,SYN,RST,ACKの中でSYNのみが立っているパケット」以外のTCPパケットにマッチします。
    - -p tcp -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN
    - -p tcp -m tcp ! --syn

リミットマッチオプションについて (1)

- リミットマッチオプションを使用すると、マッチング条件にパケットをマッチさせる回数を制限することができます。
  - LOGターゲットで情報を記録する際に、大量のログ出力が発生して問題になる場合があります。このような時にログ出力を制限することができます。
  - マッチした回数を記録するカウンタによってコントロールします。パケットがマッチすると、カウンタが1づつ増加していき、指定した値に達すると、それ以上はマッチしなくなります。ただし、一定時間ごとにカウンタの値が1づつ減少します。
  - limitモジュールの拡張マッチング機能のため「-m limit」オプションを指定します。

- 次の例で説明します。
  - -m limit --limit 5/min --limit-burst 10
  - カウンタの最大値が10「--limit-burst 10」で、1分間に5回（正確には12秒に1回）カウンタの値が減少します「--limit 5/min」。その結果、条件にマッチするパケットが連続した場合、最初の10個はすべてマッチして、その後は12秒に1個の割合でマッチします。
  - 「--limit」オプションは、単位時間に何回カウンタを減少するかを指定します。「/sec」（秒）、「/min」（分）、「/hour」（時間）、「/day」（日）が使用可能です。

リミットマッチオプションについて (2)

- LOGターゲット以外でもリミットマッチを使用することができます。
  - 次の例では、10回以上連続するpingを受けると、その後は、1時間に6回だけ（正確には10分に1回だけ）pingに応答します。

```
# iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 6/hour --limit-burst 10 -j ACCEPT
# iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

- 各オプションのデフォルト値は「--limit 3/hour」および「--limit-burst 5」です。
  - 「-m limit」のみを指定した場合は、それぞれのデフォルト値が適用されます。

メモとしてお使いください

メモとしてお使いください