クラウド基盤構築演習
第二部: Eucalyptusによるクラウド基盤構築

第11回: Eucalyptusの仕組み（前編）

ver1.0 2012/02/28

目次

- マシンイメージ管理機能の仕組み
- ユーザ管理機能の仕組み
- キーペア管理機能の仕組み
- インスタンス管理機能の仕組み

本講で学ぶこと、実施すること -1-

- マシンイメージ管理機能の仕組みを理解しましょう
  - マシンイメージがWalrusによってどのように管理・構成されているかが理解できます
- ユーザ管理機能の仕組みを理解しましょう
  - ユーザがCLCにてどのように管理されているかが理解できます

本講で学ぶこと、実施すること -2-

- キーペア管理機能を理解しましょう
  - インスタンスにキーペアの公開鍵が設定されるまでの処理の流れが理解できます
- インスタンス管理機能を理解しましょう
  - インスタンスが起動するまでに各コンポーネントでどのような処理が発生しているかを理解します
  - Eucalyptusの機能の大半がこの機能によるものであることが理解できるでしょう

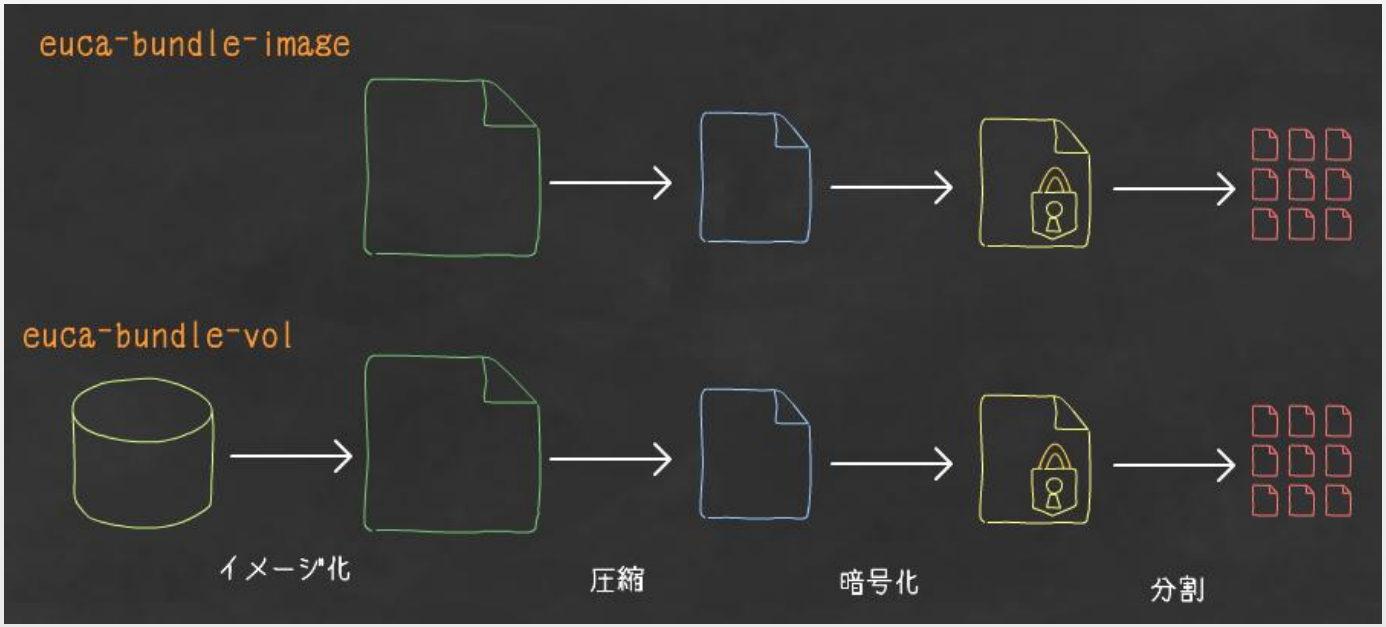
マシンイメージ管理機能の仕組み

イメージの登録 -1-

- Eucalyptusでインスタンスを起動するためには、まずマシンイメージを作成して登録する必要があります
  - 一般ユーザはカーネルイメージやRAMディスクイメージを登録できないため、一般的には管理者が登録したそれらのイメージを利用します
- 登録するためのマシンイメージを用意する方法は色々な方法がありますが、手っ取り早い方法は以下の2つのサイトから入手する方法があります
  - Eucalyptusの公式サイトから入手
    http://open.eucalyptus.com/wiki/EucalyptusUserImageCreatorGuide
  - マシンイメージ工房から入手
    http://eucalyptus.machine-image.com
- 入手したマシンイメージはEuca2oolsのコマンドeuca-bundle-imageとeuca-upload-bundleとeuca-registerを実行して登録します
  - もちろんEuca2oolsを使わずに他のツールやAPIを直接叩いて登録することも可能です
- ただし、v3.1からは状況が変化し、EuStoreと呼ばれるイメージストアからマシンイメージをダウンロードして登録するためにeustore-系コマンドが提供されており、それで登録するのが一番簡単になっています。

イメージの登録 -2-

- マシンイメージのバンドルの仕組み
「バンドル」にはイメージファイルをバンドルするeuca-bundle-imageコマンドと、稼動しているLinuxをバンドルする――つまりインスタンスを保存するeuca-bundle-volコマンドの2つがあります



イメージの登録 -3-

- euca-bundle-imageはイメージファイルをtar+gzで圧縮し、X509証明書で暗号化し、10MB毎に分割し、マシンイメージに関する情報を記述したマニフェストファイルを生成します
  - euca-bundle-volは空のイメージファイル(ext3フォーマット)を作成し、物理ディスクのデータをイメージファイルにコピーし、euca-bundle-imageと同じ処理を行ないます
  - 作成できるイメージファイルは最大で10GBまでです
  - Euca-bundle-volは--no-inheritオプションが指定されない限り、メタデータからインスタンス情報を取得します。これはマニフェストを生成する際にインスタンスの情報を継承するためです。そのためインスタンスでない環境、たとえば物理マシン上でeuca-bundle-volを実行する場合は--no-inheritオプションを指定します
  - 暗号化処理では圧縮されたイメージファイルをAES-128-CBCという方式で暗号化します
  - バンドルではインスタンスがメタデータを取得する場合を除き、Eucalyptusにはアクセスしません

イメージの登録 -4-

- マシンイメージのアップロードの仕組み

  マシンイメージをeuca-upload-bundleでWalrusにアップロードする際は、まずeuca-upload-bundleのオプションで指定されたバケット名が既にWalrusに存在するか否かをチェックします。指定されたバケットが既に存在し、アクセス権限があれば処理を継続しますが、アクセス権限がない場合はエラーになり処理が中断します。なお、指定されたバケットが存在しない場合はバケットを作成します。

  バケットの準備が整ったら、まずマニフェストファイルをアップロードし次に分割されたファイルをアップロードします。

  ちなみに、マシンイメージのアップロードではEC2 APIは使わず、S3 APIのみ使用します。

イメージの登録 -5-

- マシンイメージの登録の仕組み

  マシンイメージの登録にはeuca-registerコマンドを使用しEC2 APIのRegisterImageを発呼します。Eucalyptusは指定されたマニフェストに従って、アップロードされている分割マシンイメージの整合性チェックを行ないます。チェックの結果に問題がなければ分割マシンイメージを結合→複号→展開という処理を行ない、キャッシュファイルとして素のマシンイメージをWalrusに保持します。

  なお、このキャッシュファイルを作成する際に、Web管理画面の「Walrus Configuration」の項目「Space reserved for unbundling images (MB)」に設定されている値（デフォルト値は30720MB）をチェックし、キャッシュファイルが消費しているディスクサイズがその値以上であれば古いキャッシュファイルを削除します。

  ちなみに、Walrusがキャッシュファイルを作成/保持するタイミングには、このRegisterImageが発呼された場合やもしくはインスタンス起動時にWalrusにキャッシュファイルが存在しない場合などがあります。

ユーザ管理機能の仕組み

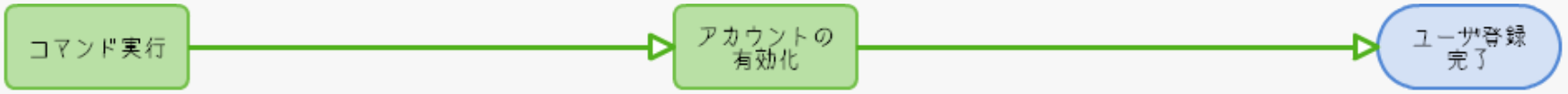
- バージョン2.X系まで -

ユーザの登録 -1-

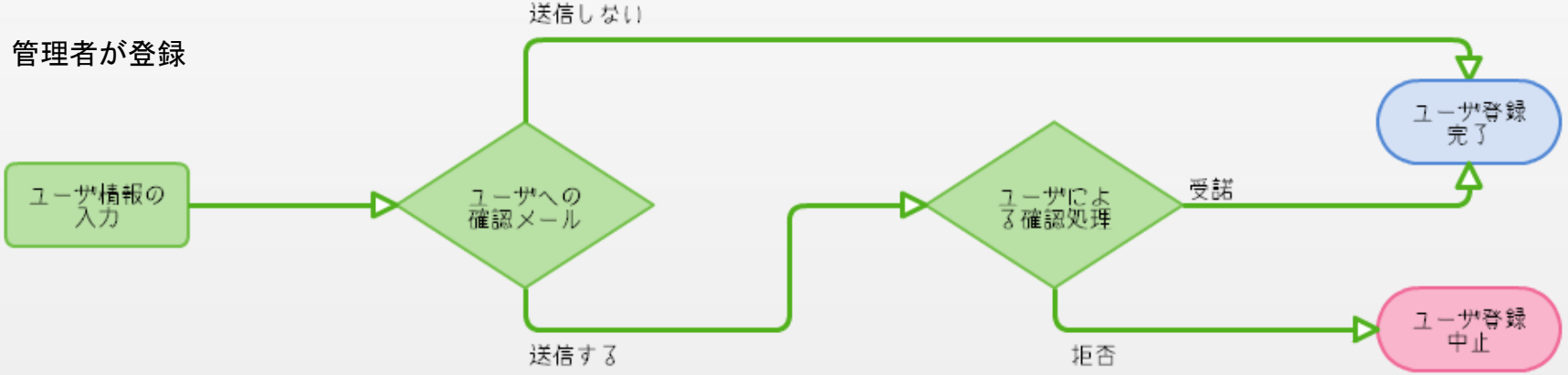
- Eucalyptusを利用するためには何よりも最初にユーザを登録しないといけません
  - もちろん、最初にシステムが作成するユーザ「admin」で利用し続けることも可能ですが、複数の利用者で利用する場合に単一ユーザを共有するのは好ましくありません
- ユーザを登録するには以下の3つの方法があります
  - CLC上で管理コマンド「euca-add-user」で登録
  - Eucalyptusの管理者がWeb管理画面で登録
  - Eucalyptusを利用したいユーザがWeb管理画面でユーザ登録を申請
    - つまり前述したECCの利用申請です

ユーザの登録 -2-

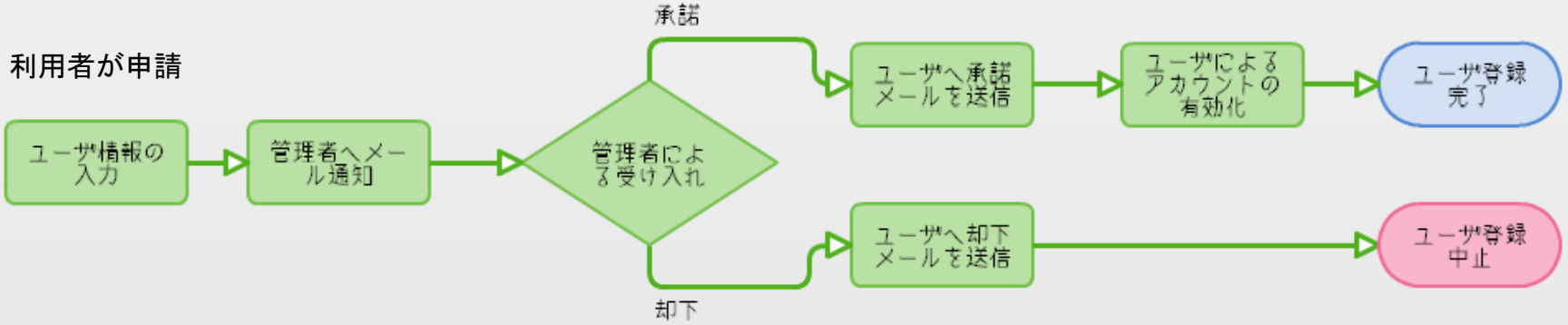
コマンドで登録



管理者が登録



利用者が申請



キーペア管理機能の仕組み

キーペアの処理

- ユーザがキーペアを作成した時の処理は以下のようになっています
  - sshの公開鍵をCLCのDBに登録
  - sshの秘密鍵をユーザに出力
    - euca2oolsは標準出力に出る
- ユーザがインスタンスを起動した際にEucalyptusはキーペアのssh公開鍵をインスタンスの/root/.ssh/authorized_keysに追加します
  - このときマシンイメージがext3以外でフォーマットされている場合は、この処理は失敗します

インスタンス管理機能の仕組み

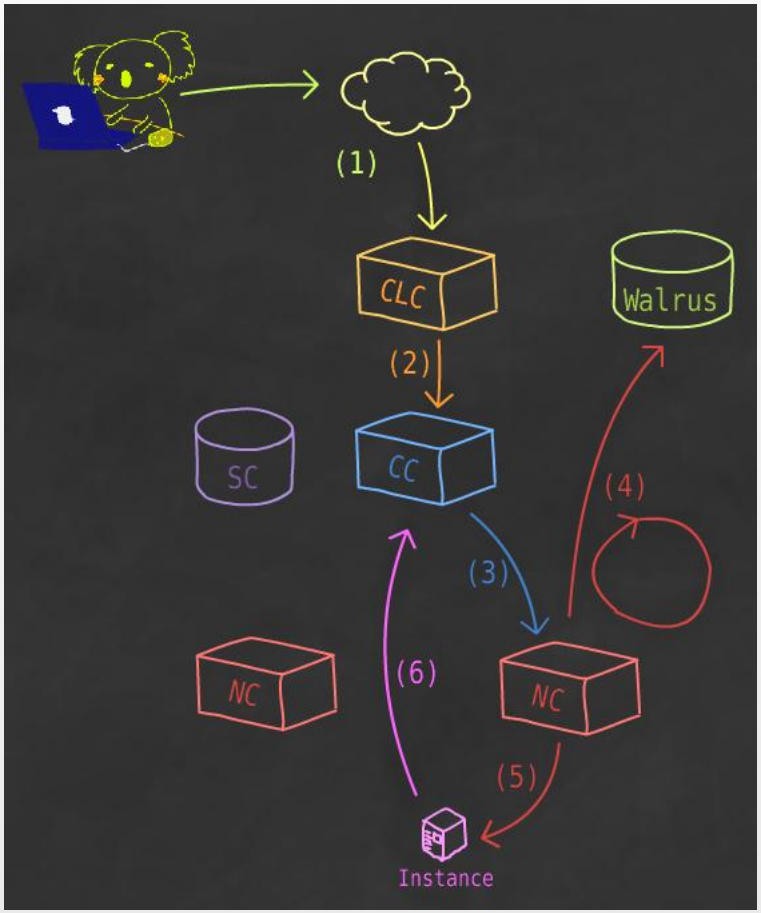
インスタンスの起動

- インスタンスの起動処理の流れ

  ユーザがインスタンスを起動した際の各コンポーネント間の処理のおおまかな流れは以下のようになっています。

  1. ユーザがCLCにインスタンスの起動を要求
  2. CLCがCCにインスタンスの起動を要求
  3. CCは起動スケジュールと空きリソース具合によってインスタンスを起動するNCを選定し、選定したNCにインスタンスの起動を要求
  4. NCは自身のキャッシュを確認し、キャッシュが存在しない場合はWalrusからマシンイメージを取得。キャッシュが存在する場合はキャッシュをコピー
  5. NCはマシンイメージからインスタンスを起動
  6. 起動したインスタンスはCCで起動しているDHCPサーバからIPアドレスを取得

  この流れに沿って、各コンポーネント上での処理を説明していきます。



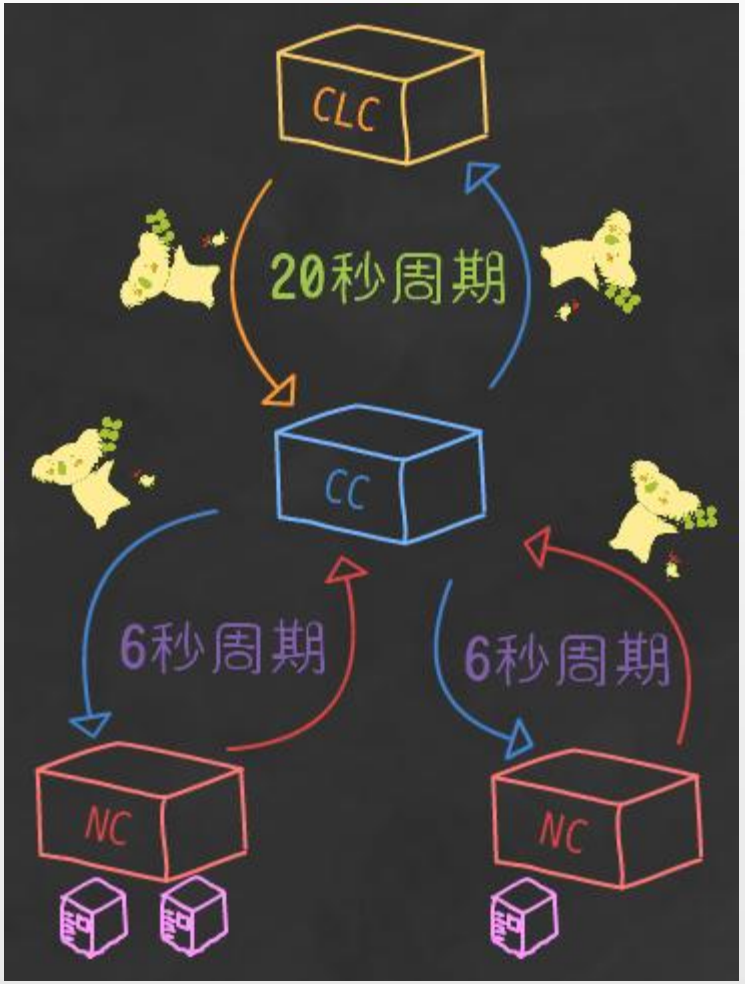
インスタンスの起動処理 -1-

- ユーザがCLCにインスタンスの起動を要求

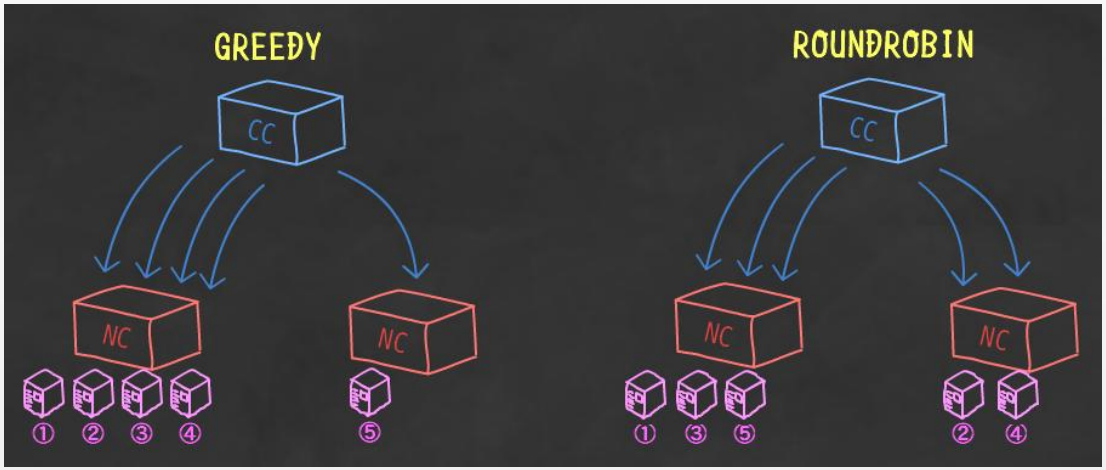
  CLCはユーザからのインスタンス起動の要求を受けつけると以下のチェックを行ないます。
  - ユーザの正当性をチェック
  - 起動要求の内容の正当性をチェック
  - 起動するインスタンスに見合ったリソースの空きに関するチェック

  リソースの空きは、指定されたVM Typesの空きがあるか、利用可能なPublicIPの空きがあるかなどをチェックします。チェックした結果がエラーの場合はユーザにエラーを返し、問題がなければ次の処理へと進みます。

  CLCは自身のクラウドにおけるリソースの空き状況を把握するために20秒毎に各CCにリソース情報の問い合わせを行っており、CCは自身のクラスタにおけるリソースの空き状況を把握するために6秒毎に各NCに対してリソース状況の確認を行なっています。



インスタンスの起動処理 -2-



- CLCがCCにインスタンスの起動を要求
  CLCがCCにインスタンスの起動を要求する際に以下の情報を渡します。
  - マシンイメージなどの情報
  - インスタンスタイプ
  - キーペアの情報
  - セキュリティグループなどの情報

  CCは各NCのリソース状況やEucalyptusの設定値に従いインスタンスを起動するNCを選択し、CLCにはプライベートIPとMACアドレスとNCの情報などを返します。CCから情報を受け取ったCLCはパブリックIPの割り当てをCCに要求します。

  なお、CCがNCを選択する際にはeucalyptus.confに設定されているSCHEDPOLICYに従います。SCHEDPOLICYに設定できる値はGREEDYとROUNDROBINとPOWERSAVEがあります。

  GREEDYは1つめのNCのリソースが枯渇するまでそのNCでインスタンスを起動し、リソースが枯渇したら次のNCを使うようなスケジュールポリシーです。ROUNDROBINは各NCで順番にインスタンスを起動していくスケジュールポリシーで、Eucalyptusではこれがデフォルト値です。POWERSAVEはインスタンスが起動していないNCを停止させ、リソースが必要になったらWake-on-LANでNCを起動させるスケジュールポリシーですが、これはUbuntuでのみ動作します。

インスタンスの起動処理 -3-

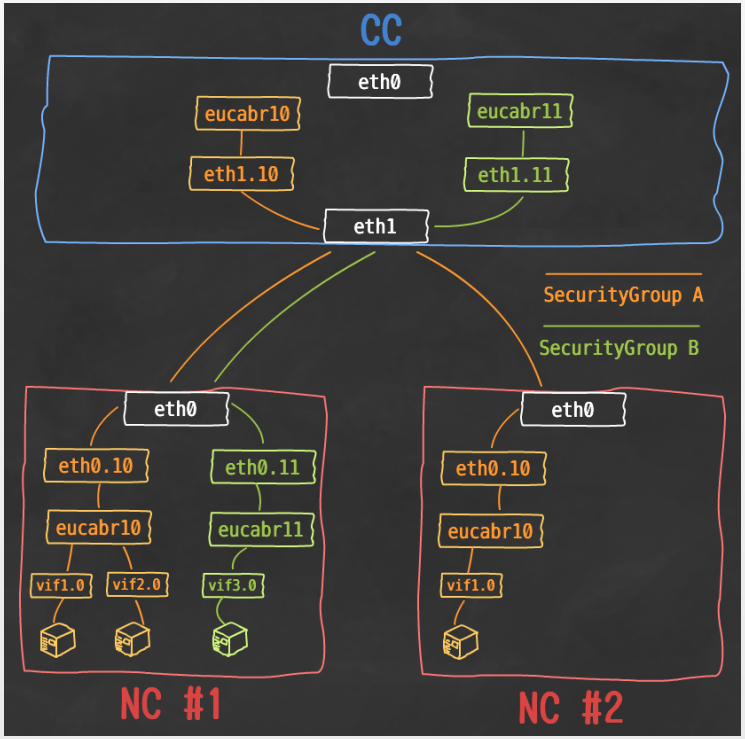
- CCがNCにインスタンスの起動を要求

  CCはNCにインスタンスの起動要求を行なう前処理として、当該セキュリティグループに関する仮想ネットワークが存在するか否かをチェックし、もし該当する仮想ネットワークが存在しない場合は以下のように仮想ブリッジやVLANデバイスを作成します。これらについては本セミナーの第二部で詳しく説明致します。

  仮想ネットワークを作成し、CCがNCにインスタンスの起動を要求する際には以下の情報を渡します。

  - マシンイメージに関する情報
  - インスタンスタイプ
  - キーペアの情報
  - タグVLANのIDとPrivateIP用のMACアドレス

  NCにインスタンス起動要求が受け入れられたら、CCはCLCから受け取ったPublicIPをPrivateIPと関連付けるルールをiptablesに設定します。一方、命令を受け取ったNCは起動するインスタンス分のリソースを自身の空きリソースから減じて次の処理に進みます。



インスタンスの起動処理 -4-

- NC上でのマシンイメージ処理

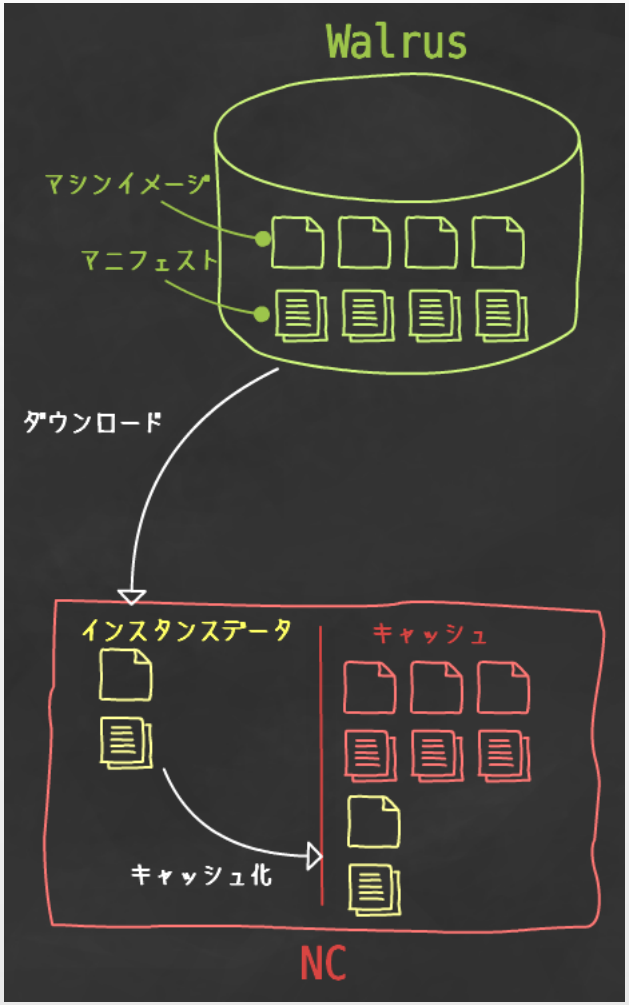
  NCはインスタンスの起動要求を受け付けたら、要求されたマシンイメージが自身のキャッシュに存在するかチェックします。マシンイメージがキャッシュ上に存在しない場合はWalrusから以下のデータをダウンロードします。

  - マシンイメージ(EMI,EKI,ERI)のマニフェストファイル
  - マシンイメージ(EMI,EKI,ERI)

  Walrusからマシンイメージをダウンロードする際、Walrus上に復号されたマシンイメージがキャッシュされていない場合はWalrus上で復号処理が実施され、NCは復号処理が終るまでダウンロード処理を(10回まで)リトライし続けます。

  マシンイメージがダウンロードされたら、NCはインスタンスを起動する前にダウンロードしたマシンイメージをNC上のキャッシュに登録します。もしNC上のキャッシュがeucalyptus.confのNC_CACHE_SIZEで指定した値に達した場合は古いキャッシュファイルから削除され、新しいマシンイメージがキャッシュに加えられます。

  もしマシンイメージがNCのキャッシュ上に存在する場合は、Walrusからはマニフェストファイルのみダウンロードされ、NC上のキャッシュファイルの完全性をチェックします。キャッシュファイルが破損および改竄されていない場合はキャッシュからインスタンスデータの格納場所にコピーされます。



インスタンスの起動処理 -5-

- NCでのインスタンス起動処理

  NCはマシンイメージの処理が終ったのち、キーペアのSSH公開鍵をマシンイメージ(EMI)の/root/.ssh/authorized_keysに追加します。

  次に、NCはswapディスクとephemeralディスクの準備を行ないます。このときNCはswapおよびephemeralディスクをddコマンドでraw形式なファイルとして作成を行ないます。なお、このときにephemeralディスクは数GBから数十GBのサイズになるため、処理に時間がかかります。

  インスタンスデータの格納場所にこれら各種ファイルが準備できたら、NCはlibvirtで起動するためのxmlファイル(libvirt.xml)を生成し、それを以ってlibvirtにインスタンスの起動を命令します。libvirtおよびハイパーバイザーがインスタンスを起動する仕組みについては本セミナーの第二部にて詳しく説明致します。
- インスタンス上での起動処理

  インスタンス上で起動処理が始まる頃にはCLCは既にインスタンスの状態をpendingからrunningに遷移させています

  なお、このときインスタンス自身はまだ起動処理中ですので、CLC上でrunningに遷移した直後にインスタンスに接続しようとするとコネクションエラーなどになる場合があります。

  インスタンスは自身の起動処理の中でCCのDHCPサーバからPrivateIPのアドレスを取得します。

インスタンスの停止処理

- S3インスタンスを停止した際の処理

  インスタンスを停止すると、CLC→CC→NCの順番で停止命令が発呼されます。

  CCは当該インスタンスが起動しているNCに対して停止命令が発呼できなかったり、発呼した停止命令に対するレスポンスが得られないなどの場合には管理下の全NCに対して当該インスタンスの停止命令を発呼し、停止命令を受け取ったNCはCCに対して停止命令を受け付けたことを返信します。

  そのあとNCはハイパーバイザーに停止命令を送り、インスタンスを停止します。ここまでの間、インスタンスのステータスはrunningからshutting-downに遷移します。

  NCはインスタンスが停止したのち、インスタンスデータ(/var/lib/eucalyptus/instances/ユーザID/インスタンスID)を削除します。

  その後、NCはCCからの6秒毎のリソース問い合わせでインスタンスの情報を返し、CCは20秒毎のCLCからのリソース問い合わせでインスタンスの情報を返し、インスタンスが停止したことをCLCが把握した時点でインスタンスのステータスはshutting-downからterminatedに遷移します。

  インスタンスのステータスがterminatedに遷移したあとは一般利用者に対するインスタンスのリスト上では当該インスタンスは消えますが、CLC上ではそれから10分後に内部情報をTERMINATEDからBURIEDに遷移させ、完全にインスタンス情報を消去します。