クラウド実践演習

演習の流れ

1. クラウド上での環境構築
   - 掲示板サービスの本番環境作成
   - Puppet(+MCollective)による自動化
   - Nagios + Gangliaを用いた監視の設定
2. 掲示板サービスのスケールアウトを実現する
3. より高度なアプリケーションの配布
4. インスタンスがシャットダウンしてしまう問題に対応する

チーム分け

- 小さなグループに分けて、
  掲示板サービスの開発から運用まで全てをこなす
  チームを編成する
- チームでやること
  - 開発と運用
    - （ただし、本演習にはプログラムの開発は含まれていない）
  - 議論
    - 課題に対するグループディスカッション
    - 設計・実演
    - 結果のまとめ
  - 発表
    - 他チームとの意見交換を行う

演習課題の流れ

掲示板サービスの本番環境作成

- 背景
  - あなたのチームは、これから掲示板サービスのリリースとサービスインを控えています。
- 目的
  - 掲示板サービスを構築する
    - edubase Cloud上で掲示板サービスの本番環境を構築する
    - チーム内で手分けして掲示板サービスの構築を行う
- 備考
  - ロードバランサ(LB)、アプリケーションサーバ(Web)、データベースサーバ(DB)は、それぞれ仮想マシンイメージ(EMI)が用意されているので、これを利用してください。

Puppet(+MCollective)による自動化

- 背景
  - 手作業が多かった環境構築を、できるだけスクリプト化し、作業効率の向上を図る必要がある。
- 目的
  - 先ほど行った掲示板サービスの構築作業を自動化する
    - 自動化のためのツールとして下記の2つを用いる
      - Puppet
      - MCollective
  - 作成済みのイメージファイルを用いて、掲示板サービスを再度構成してみる。
    - 「掲示板アプリケーションの更新」部分について、Puppetのマニュフェストファイルを追加する。

NAGIOS + GANGLIAを用いた監視の設定

- 背景
  - サービスがダウンしている時間を短くするため、
    監視の仕組みを取り入れて
    掲示板サービスの障害を素早く検知したい。
- 目的
  - 稼働中の掲示板サービスへ監視を行う
    - 監視のためのツールとして下記の2つを用いる
      - Nagios
      - Ganglia
  - 監視用インスタンスの起動と構成を行う
    - 掲示板サービスを走らせているインスタンス群を
      Nagios, Gangliaの監視画面へ追加する

掲示板サービスのスケールアウトを実現する

- 背景
  - 掲示板サービスで短期のPR活動としてキャンペーンを行うこととした。大手ポータルサイトへの広告出稿を計画しており、数日後にはその広告から大量の一般ユーザがやってくるものと想定されている。
- 目的
  - 掲示板サービスの性能が低下しない対策を実施する
    - 今回の演習では、アプリケーションサーバ(Web)のスケールアウトを実演する
    - また、スケールアウトの逆であるシュリンクインについても同様に検討したい
- 備考
  - Shell Script, Puppet, Mcollectiveを組み合わせ、1コマンドラインの入力で掲示板サービスをスケールアウト/シュリンクインさせられる仕組みについて設計すること

より高度なアプリケーションの配布

- 背景
  - 一般ユーザからのフィードバックを受けて、掲示板サービスは随時バージョンアップの開発がされることになった。チームではこれを素早く本番環境へデプロイしなければならない。
- 目的
  - 素早く、正確に実施したいので、手作業をなるべく減らして実現する必要がある
  - 一般ユーザがいつも通り掲示板サービスを利用してもらいながら、新しいバージョンの掲示板サービスを使えるようにする
- 備考
  - 現在のアプリケーションファイル(warファイル)をコピーするだけのデプロイ方式は対応が不十分である。よりユーザに影響の少ない方式を検討する。

インスタンスがシャットダウンしてしまう問題に対応する

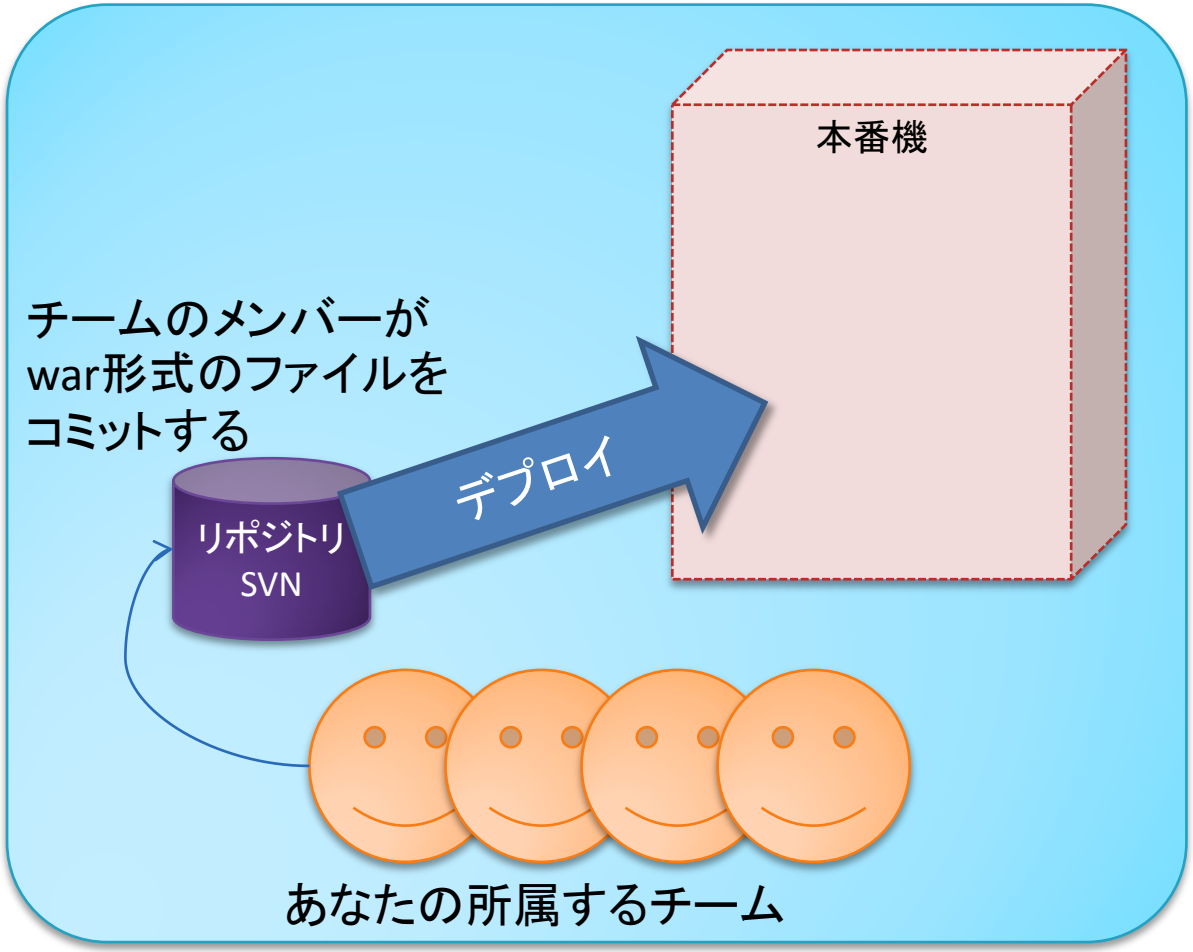
- 背景
  - IaaSを利用していると、ある日突然インスタンスがシャットダウンすると言う現象にあたった。Nagiosはそれを検知したものの、人手の対応までに時間を要してしまった。これを改善したい。
- 目的
  - 障害の検知と同時に、できる対応はスクリプトで実行し、人手の対応を待たなくて済むようにしたい
    - 可能であれば、人手を介さず、復旧するところまで完了させ、対応業務の効率化を図る
- 備考
  - 監視機能と組み合わせ、インスタンスの停止障害から復旧までをスクリプト化できるように考え実装する。

本演習の前提事項

開発ルール

- アプリケーションコードはwar形式で
  リポジトリにコミットすることとする。
  - リポジトリはSubversionを用いる
  - 今回の演習では、実際に開発は行わない
    - テストがグリーンになったアプリケーションがwar形式で既に用意されているので、これをコミットすることで、新しいバージョンの掲示板サービスの開発が完了したことにする

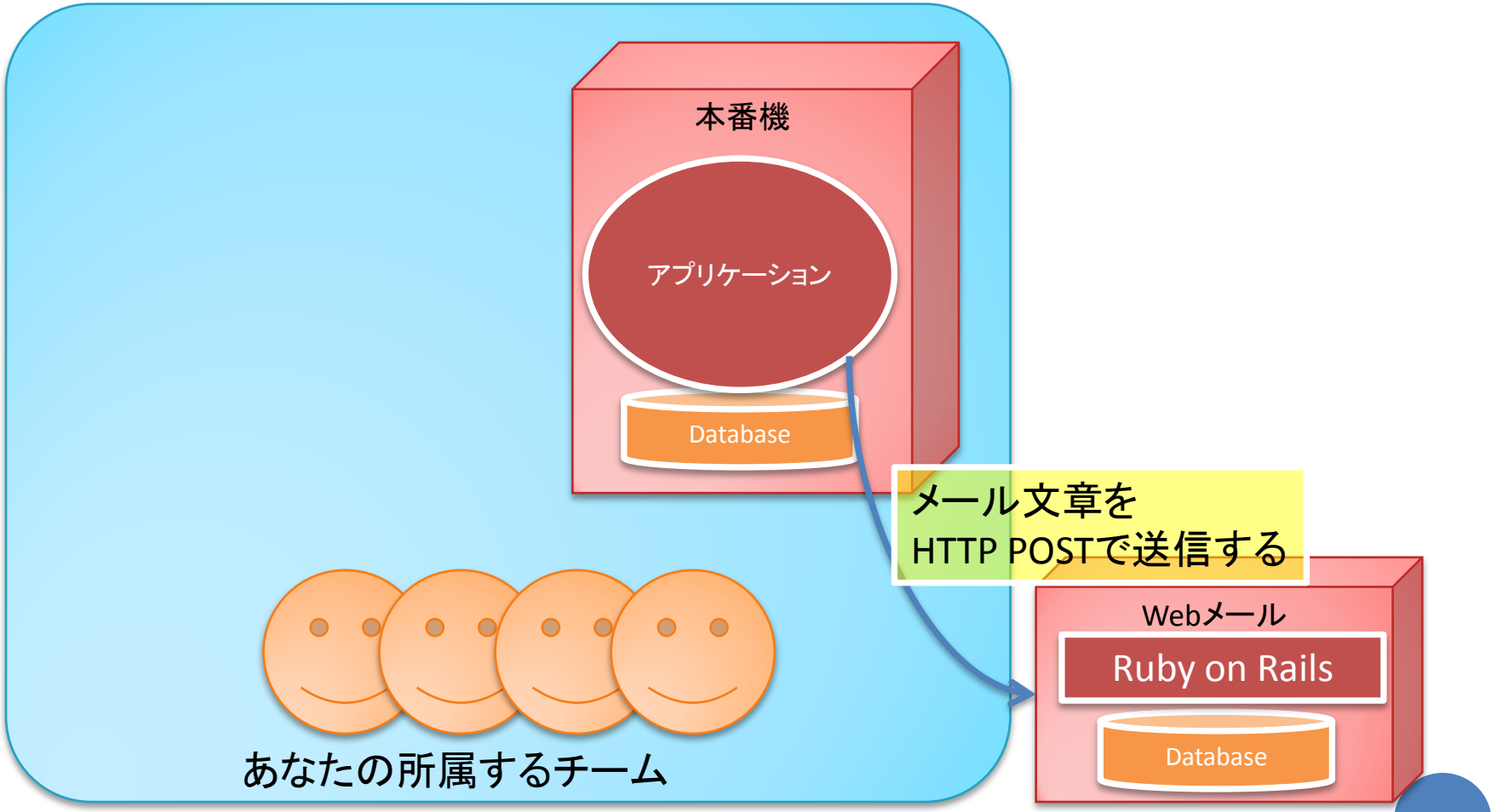
アプリケーションコードはWAR形式でリポジトリにコミットする



メールの環境

- 掲示板サービスはユーザ登録で、該当ユーザに向けてメールを送信する仕組みを備える
  - 今回の演習では一般的なemailシステムは利用しない
    - 環境が用意できないため、代替の仕組みを用意する
    - Webメールを模したサーバへ、直接掲示板サービスからメールデータをHTTP POSTすることとしている

Webメールを模したサーバへ、
掲示板サービスからメールデータをHTTP POSTする



掲示板サービスの本番環境作成

掲示板サービスの本番環境作成

Puppet(+MCollective)による自動化

Nagios + Gangliaを用いた監視の設定

掲示板サービスのスケールアウトを実現する

より高度なアプリケーションの配布

インスタンスが
シャットダウンしてしまう問題に対応する

あなたのチームは、
これから掲示板サービスの
リリースとサービスインを
控えています。

演習の内容

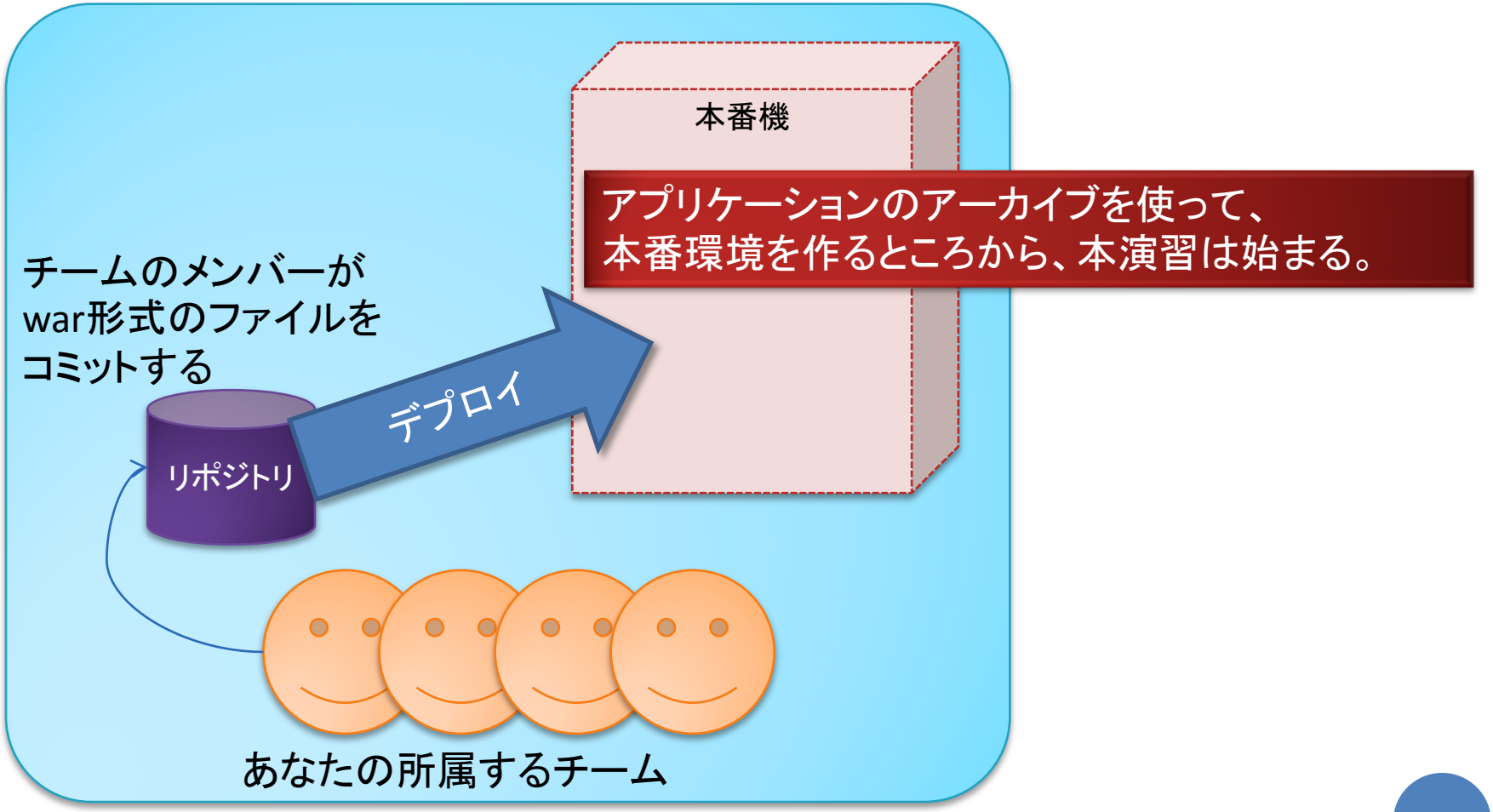
- 時間割

  0:00～0:15　前提説明

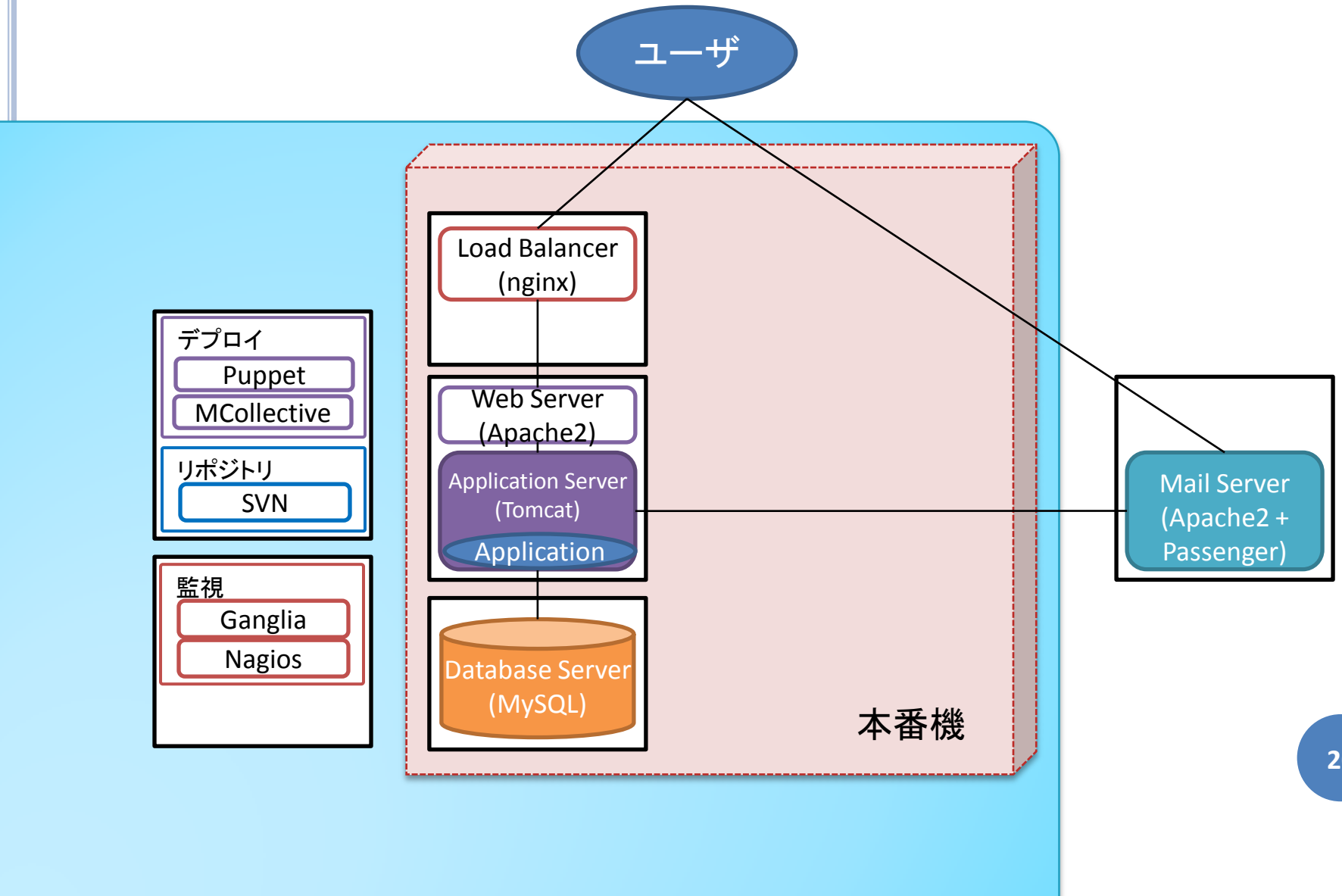
  0:15～1:00　構築作業

- 目的
  - 掲示板サービスを構築する
    - edubaseCloud上で掲示板サービスの本番環境を構築する
    - チーム内で手分けして掲示板サービスの構築を行う
- 備考
  - ロードバランサ(LB)、アプリケーションサーバ(Web)、データベースサーバ(DB)は、それぞれ仮想マシンイメージ(EMI)が用意されているので、これを利用してください。
- 使用するツール
  - ssh

演習の開始

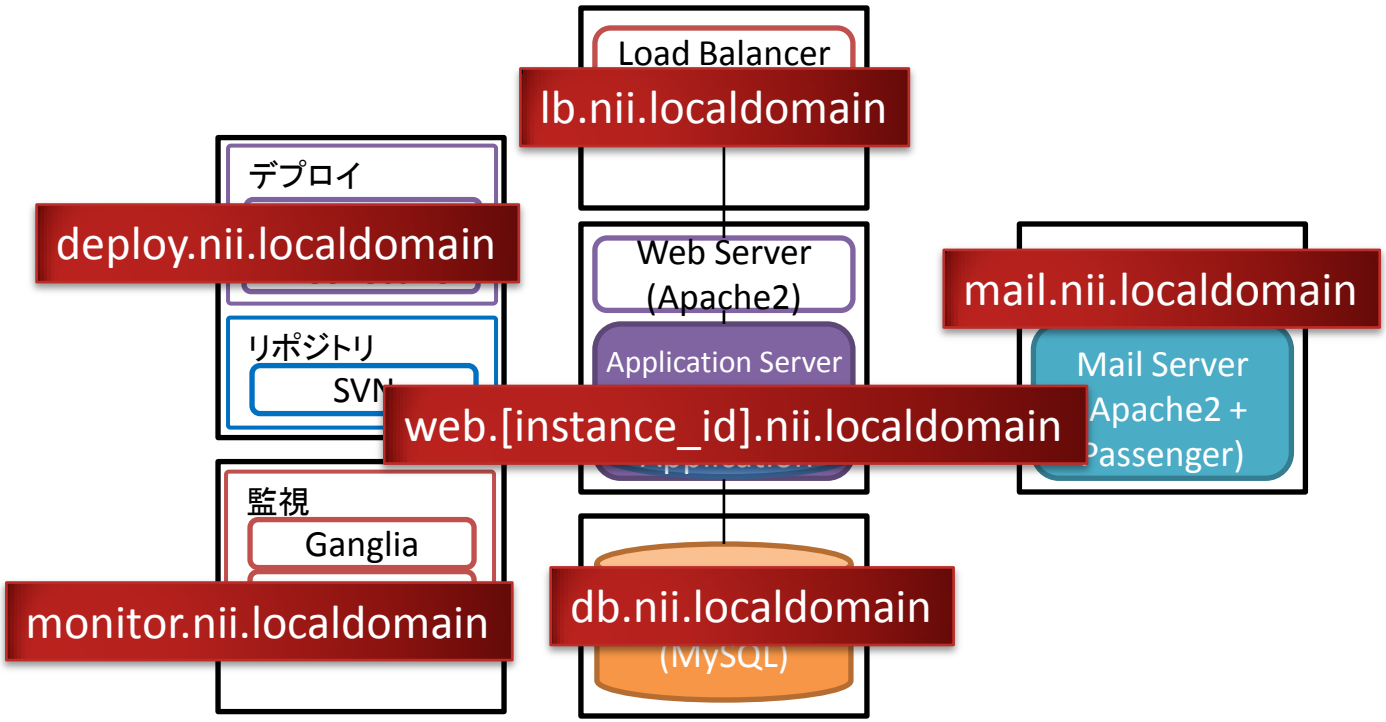

本番機
アプリケーションのアーカイブを使って、
本番環境を作るところから、本演習は始まる。
チームのメンバーが
war形式のファイルを
コミットする
デプロイ
リポジトリ
あなたの所属するチーム

掲示板サービス本番環境の構築



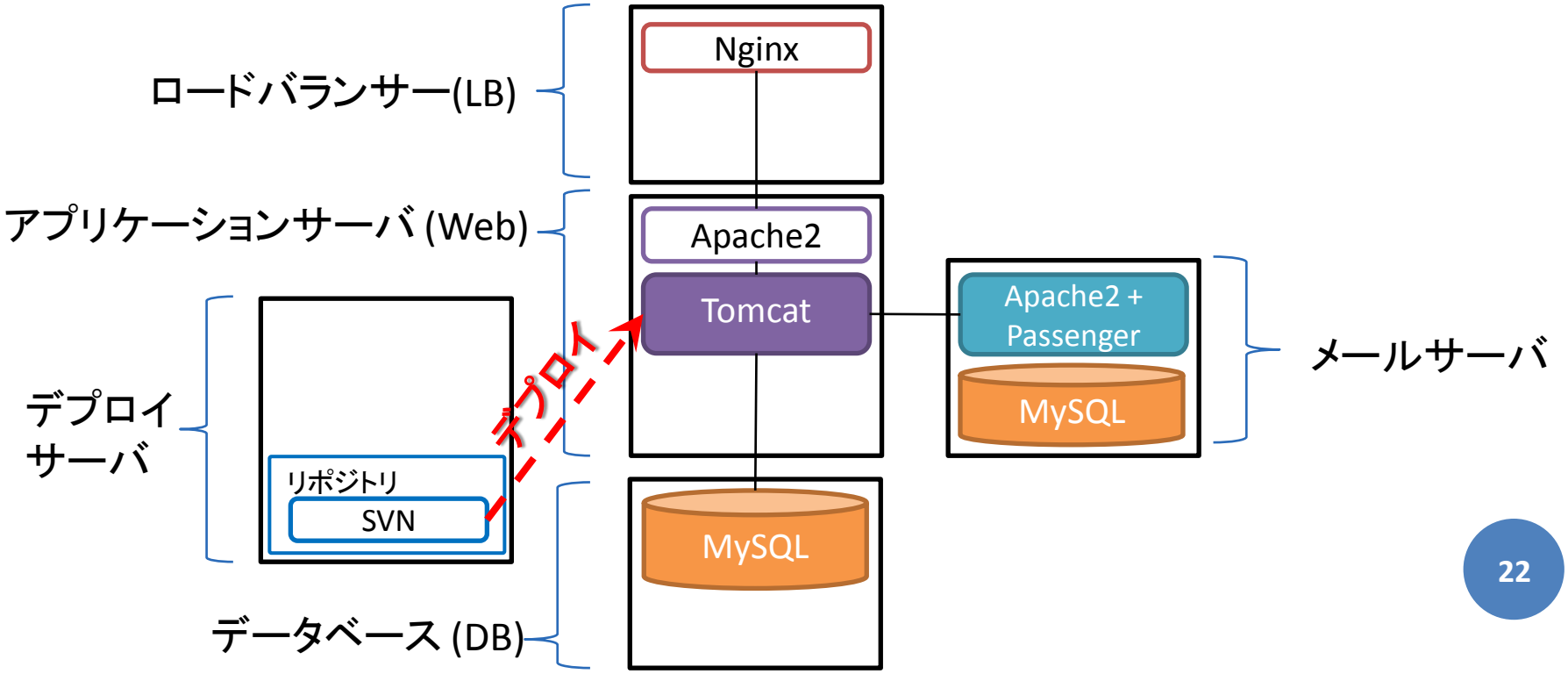
ホスト名

- 起動時にホスト名を設定するスクリプトを実行している
- Webサーバは複数台起動するため、
  ホスト名にInstance IDを含めて区別する



演習課題:環境構築手順

- 掲示板サービスを構成する
  デプロイサーバ、LB、Web、DB、メールサーバの
  各サーバを起動、セットアップをして、サービスを開始する



Puppet(+MCollective)による自動化

掲示板サービスの本番環境作成

Puppet(+MCollective)による自動化

Nagios + Gangliaを用いた監視の設定

掲示板サービスのスケールアウトを実現する

より高度なアプリケーションの配布

**インスタンスが
シャットダウンしてしまう問題に対応する**

手作業が多かった
環境構築を、
できるだけスクリプト化し、
作業効率の向上を図る
必要がある

演習の内容

- 時間割

  0:00～0:20　Puppet+MCollectiveの詳細と本演習の使い方

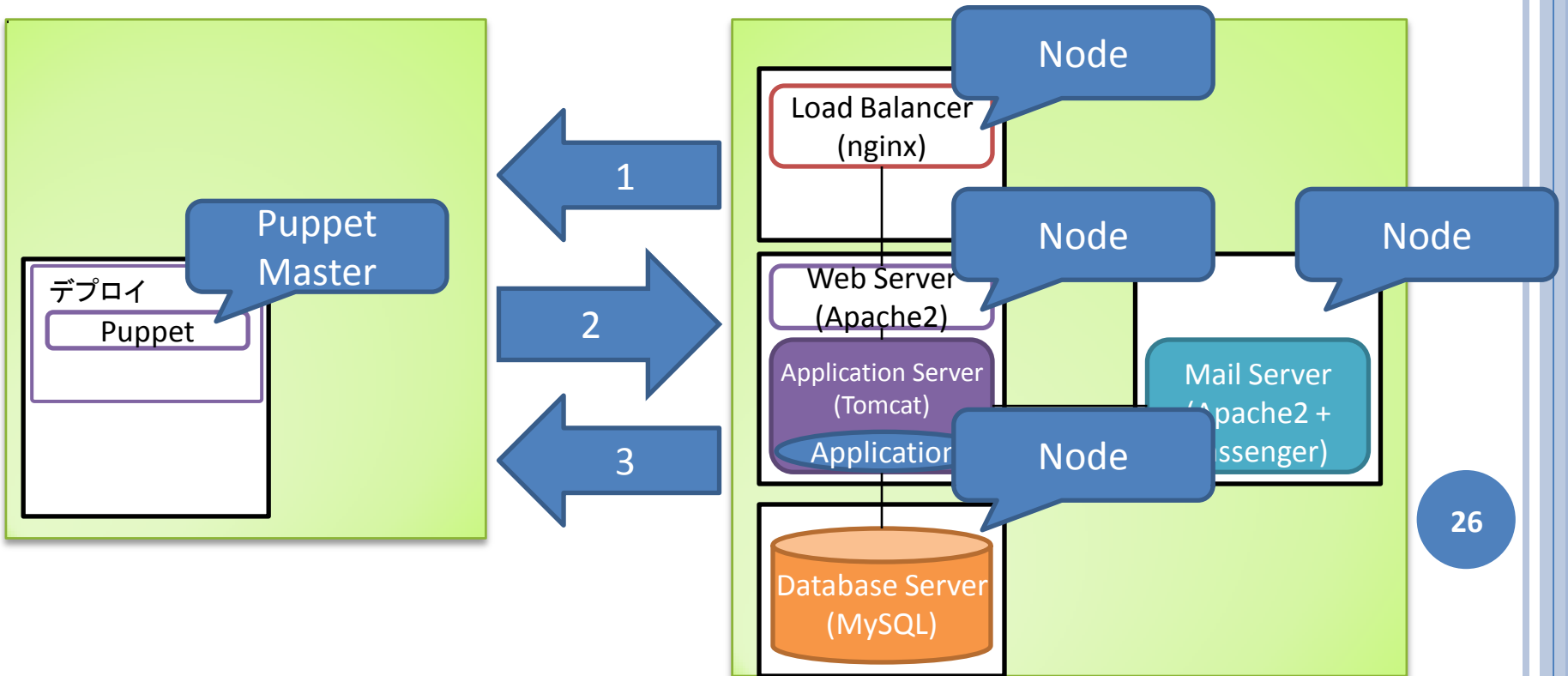
  0:20～1:00　ツールを使って自動化した環境構築手順を実施する

  1:00～1:15　Puppet+MCollectiveの設定ウォークスルー

- 目的
  - 先ほど行った掲示板サービスの構築作業を自動化する
    - 自動化のためのツールとして下記の2つを用いる。
      - Puppet:サーバの構成管理を行うツール
      - MCollective: RPCを行うためのツール
  - 作成済みのイメージファイルを用いて、掲示板サービスを再度構成してみる。
    - 「掲示板アプリケーションの更新」部分について、PuppetのRecipeファイルを追加する。

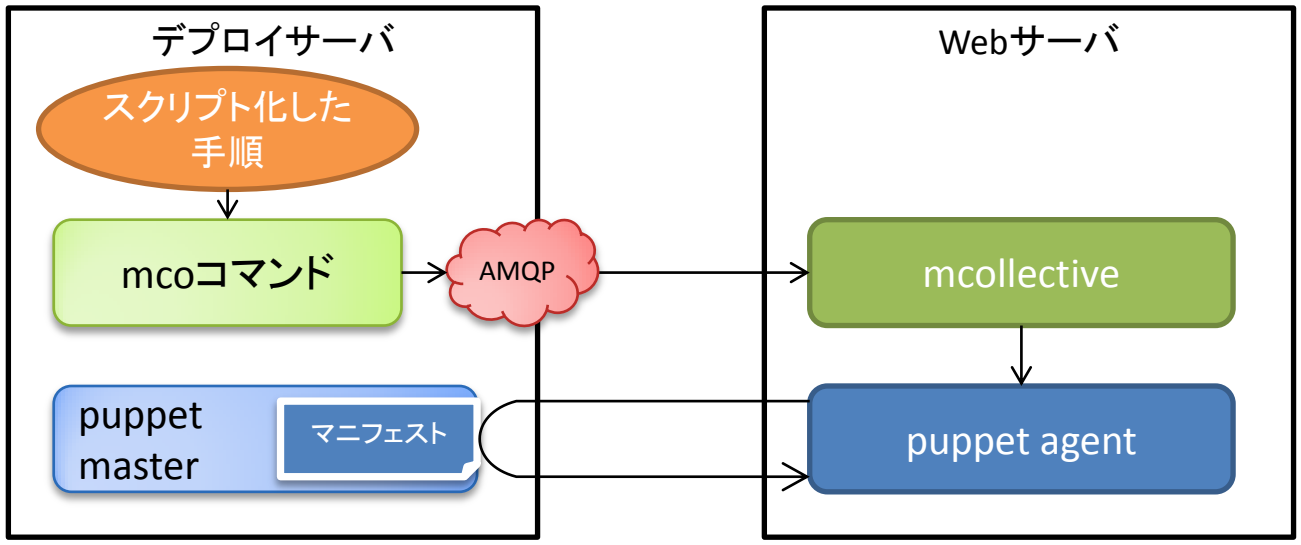
本演習でPUPPETが構成管理をするフロー

1. Puppet Masterに、構成情報の取得を要求する
2. Puppet Masterはカタログから、構成情報をNodeに送る
   - この時、Nodeのホスト名を利用して構成情報を決定する
3. 各サーバでは、Puppet Masterから受け取った構成情報を元に、設定を反映し、結果をPuppet Masterに送信する



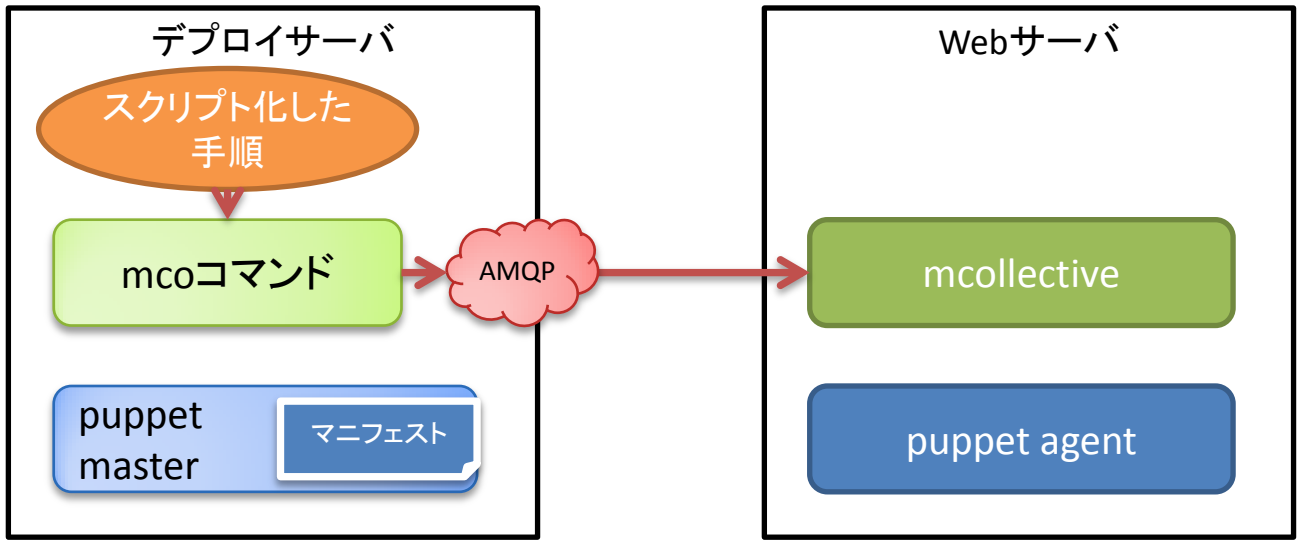
スクリプト化するために PUPPETとMCOLLECTIVEを使用する

- PuppetとMCollectiveを組み合わせて、以下の役割分担を行う
  - Puppetで構成管理をする
  - MCollectiveで実行するサーバを特定する



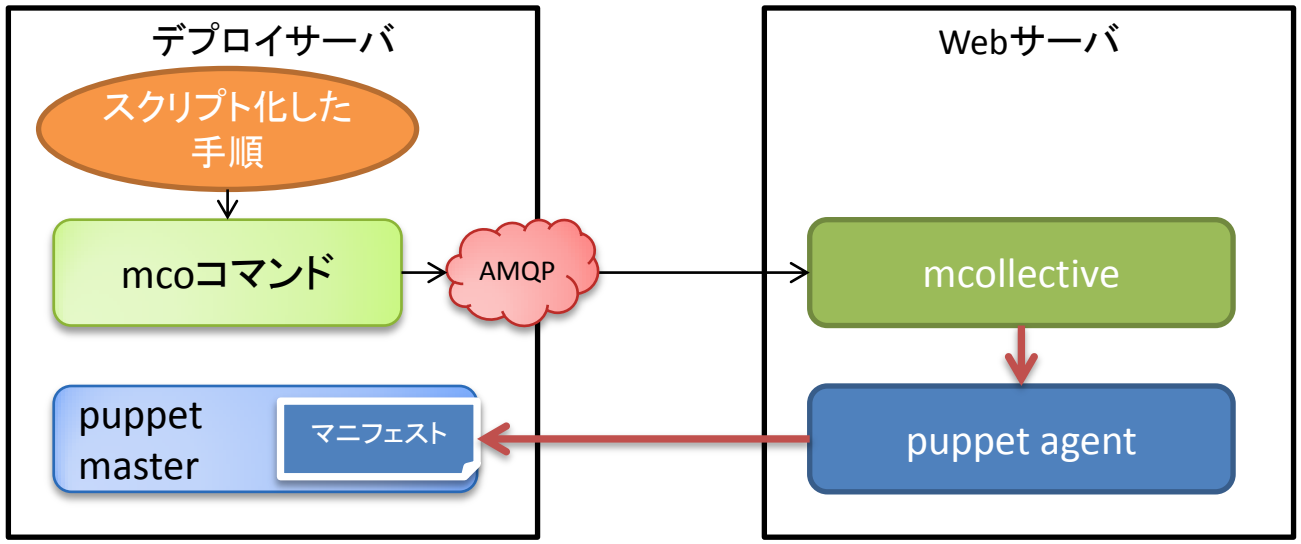
MCollectiveを使用して Puppetを起動するフロー(1/3)

- スクリプト化した手順からWebサーバを指定して mcollective-clientを実行する
- mcollective-clientはAMQPネットワークにメッセージを送り、AMQPネットワークに接続しているサーバ全てに送信される
- Webサーバがメッセージを受け取り、メッセージに従って処理を実行する



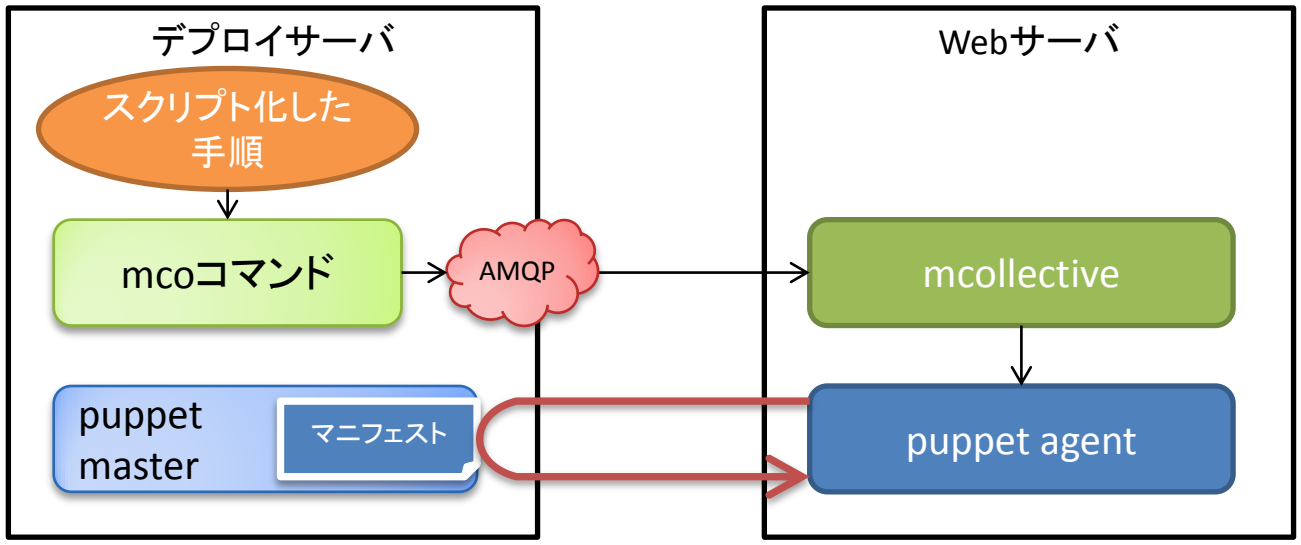
MCollectiveを使用して Puppetを起動するフロー(2/3)

- mcollectiveは指示されたメッセージに従い、Puppet agentを実行する
- Puppet agentはPuppet masterに構成情報の送信を要求する



MCollectiveを使用して Puppetを起動するフロー(3/3)

- Puppet masterはマニフェストを元にWebサーバ用の構成情報を作成し、送信する
- Puppet agentは構成情報を元にWebサーバの構成管理を行う



使用するアプリケーションのコマンドおよび用意したスクリプト

- mcoコマンド
  - mcollective-clientを実行するコマンド
  - 詳細は「MCollectiveのコマンド」を参照する
- puppetcaコマンド
  - puppet master が動作しているサーバで実行する
  - puppet agentが接続している状況の確認と承認/破棄を行う

  puppetca -l

  　接続状況の確認

  puppetca -s

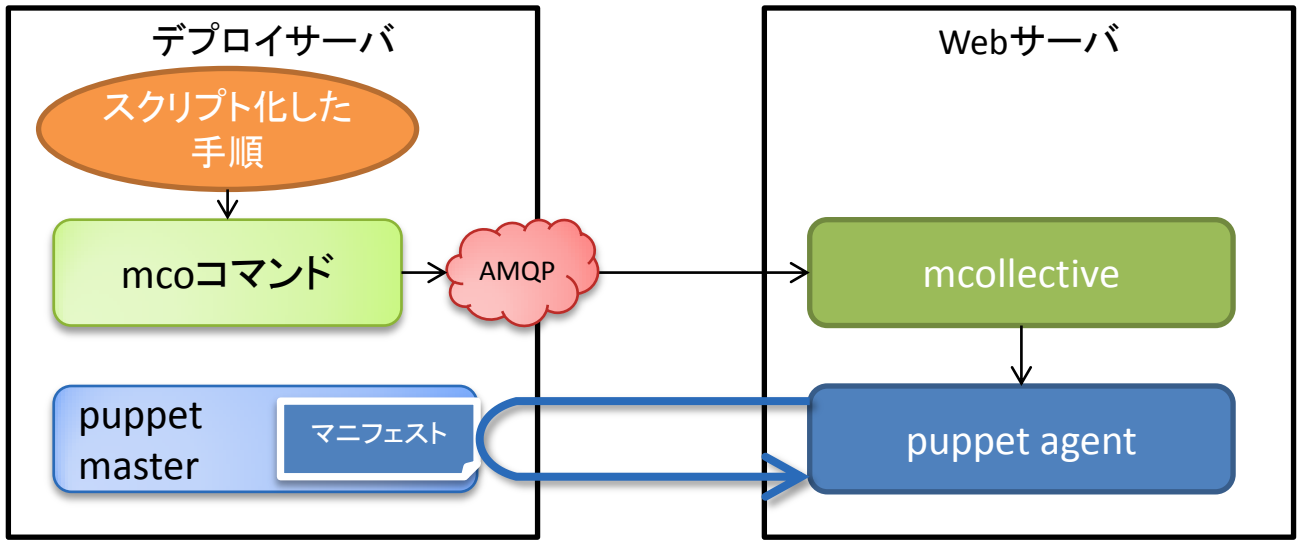
  　接続の承認
- 本演習のために用意したスクリプト
  - 「提供スクリプト」を参照する

演習課題:PUPPET+MCOLLECTIVEを用いて環境を構築する

- 「掲示板サービスの本番環境作成」において
  手動で構築した環境をPuppetとMCollectiveを利用して
  再度構築する
- 「掲示板サービスの本番環境作成」で
  起動したサーバのうち、**LB**、**Web**、**DB**、**Mail**のサーバを
  シャットダウンする
  - デプロイサーバとブラウザ確認用インスタンスを残す
  - EBS起動(LB、DB、Mail)の場合は、一覧に残さずに
    ボリュームも削除する
- 「Puppet+MCollectiveによる自動化」を実施する
  - この手順は、デプロイサーバ上だけでコマンドを実行し
    掲示板サービスの環境構築作業行う手順である。

PuppetおよびMCollective設定のウォークスルー

- デプロイサーバと各サーバの設定を確認し、PuppetとMCollectiveがどのように連携しているか理解する
  - 「PuppetMCollective説明」を参考に一通り設定を確認すること



Nagios + Gangliaを用いた監視の設定

掲示板サービスの本番環境作成

Puppet(+MCollective)による自動化

Nagios + Gangliaを用いた監視の設定

掲示板サービスのスケールアウトを実現する

より高度なアプリケーションの配布

インスタンスが
シャットダウンしてしまう問題に対応する

サービスがダウンしている時間を短くするため、監視の仕組みを取り入れて掲示板サービスの障害を素早く検知したい

演習の内容

- 時間割

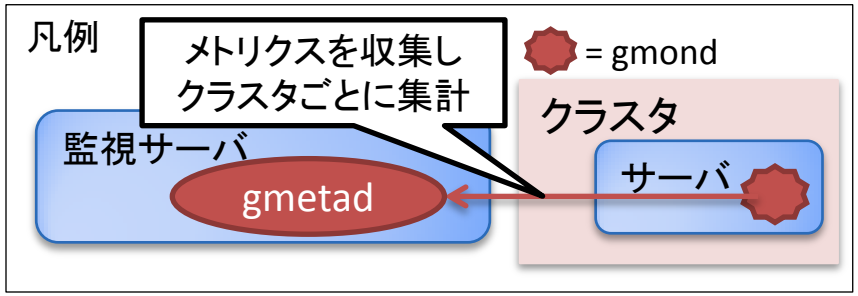
  0:00～0:20　Nagios、Gangliaの詳細説明

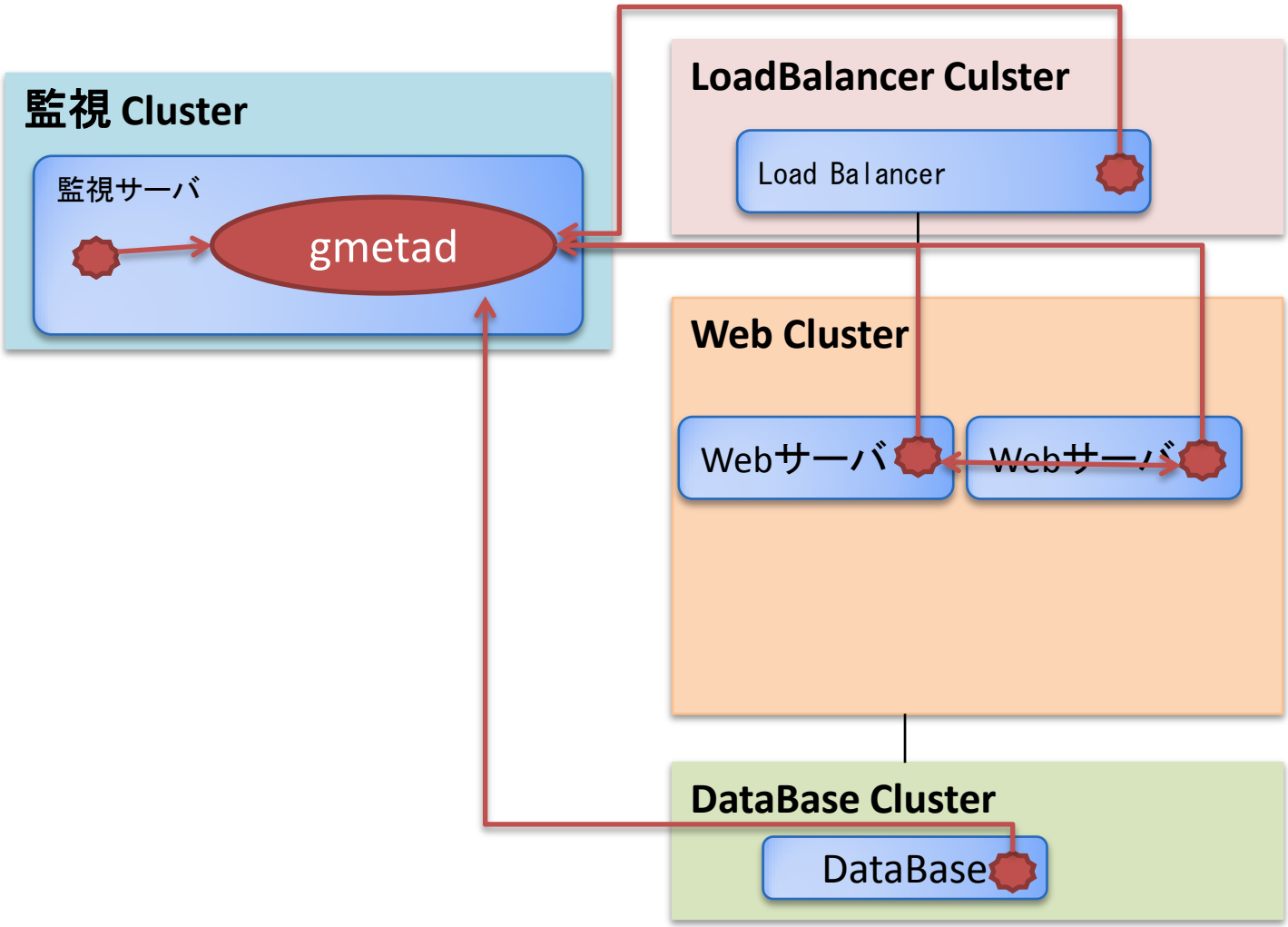
  0:20～0:50　監視サーバの起動と構成を行う

  0:50～1:15　監視設定のウォークスルー

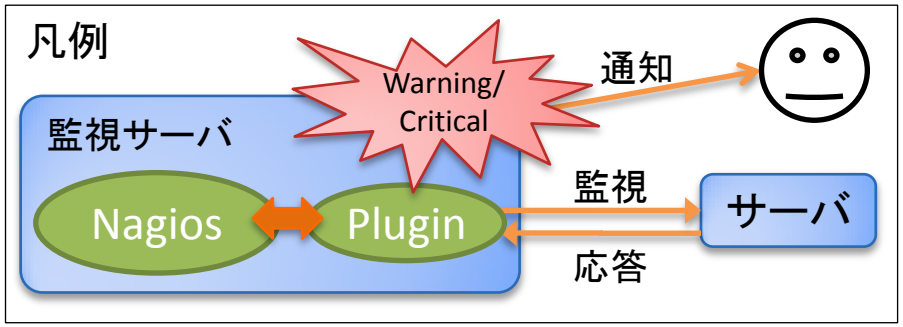
- 目的
  - 稼働中の掲示板サービスへ監視を行う
    - 監視のためのツールとして下記の2つを用いる。
      - Nagios
      - Ganglia
  - 監視用インスタンスの起動と構成を行う
    - 掲示板サービスを走らせているインスタンス群をNagios, Gangliaの監視画面へ追加する

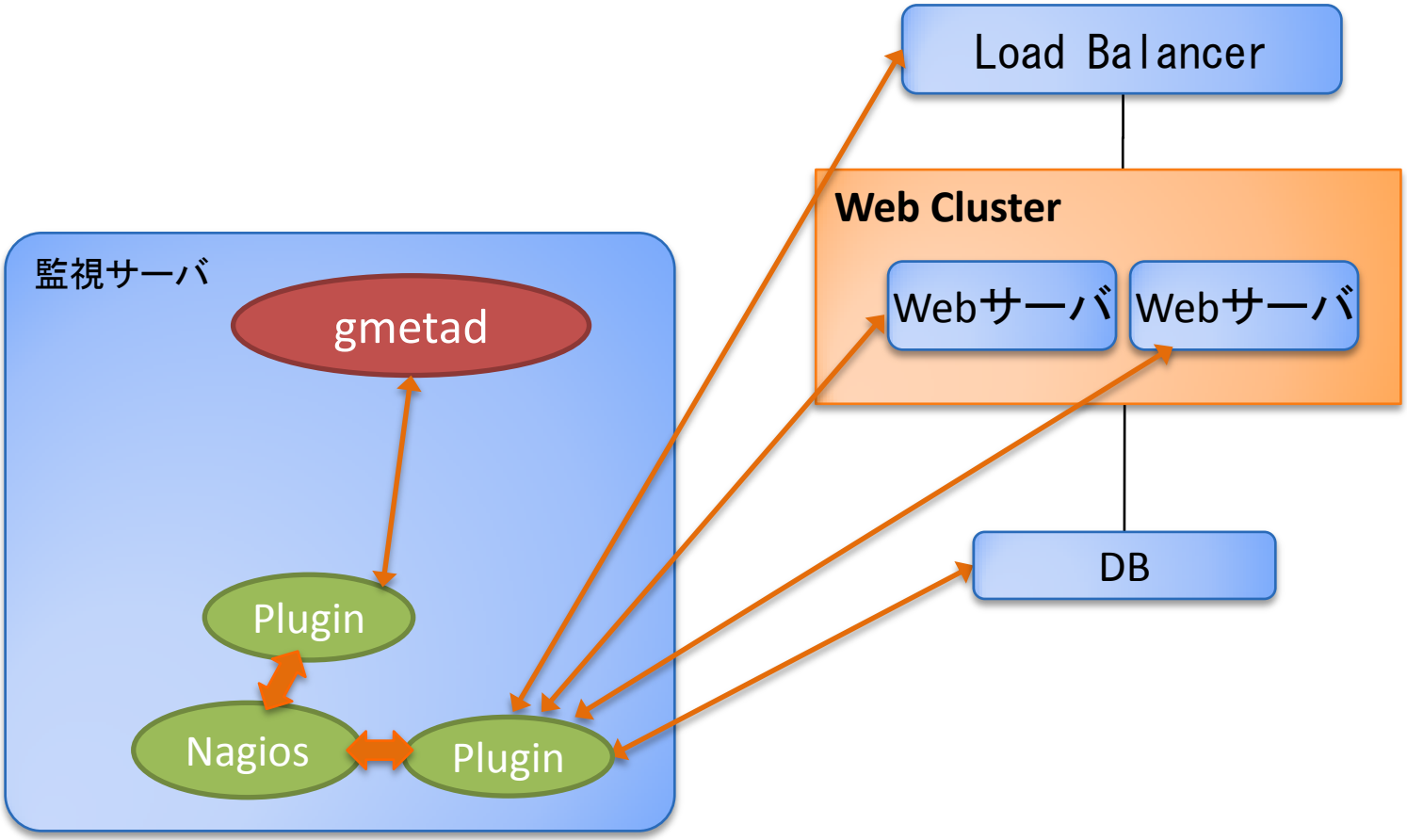
GANGLIAによる監視





NAGIOSによる監視





NagiosとGangliaの使い方

- Gangliaが収集する
  - 各サーバのメトリクスをクラスタに分けて収集する
  - RRDに保存する
- Nagiosが異常を検知をする
  - pingなどの外形監視をする
  - Gangliaの収集した結果を監視する

演習課題:構築した環境に監視を追加する

- Puppet+MCollectiveを用いて構築した環境に
  監視サーバを追加し、各サーバの監視を開始する
- 「Nagios + Gangliaを用いた監視の設定」
  の手順を実施すること

監視設定の確認をする

- 監視サーバと各サーバの設定を確認し、GangliaとNagiosが実施している監視を確認する
  - 「NagiosGanglia説明」資料を参考に一通り設定を確認する
  - Nagios、Gangliaの監視結果をブラウザで確認する

本番環境の構築完了

- 構築の自動化、監視の設定が終わったため、
  本番環境の構築は完了となる
- これより、掲示板サービスを公開する
  - こちらでJMeterを用意したので、
    実際のユーザを模倣したアクセスをする
    - 演習の時間が終わるまで実行したままにする
  - 大きな負荷にはならないが、
    なるべくユーザ側でエラーとならないように
    運用をすること

掲示板サービスのスケールアウトを実現する

掲示板サービスの本番環境作成

Puppet(+MCollective)による自動化

Nagios + Gangliaを用いた監視の設定

掲示板サービスのスケールアウトを実現する

より高度なアプリケーションの配布

インスタンスが
シャットダウンしてしまう問題に対応する

掲示板サービスで短期のPR活動としてキャンペーンを行うこととした。
大手ポータルサイトへの広告出稿を計画しており、
数日後にはその広告から大量の一般ユーザがやってくるものと想定されている。

演習の内容

- 時間割

  0:00～0:15　Webサーバをスケールアウトする手順を検討する

  0:15～1:00　Webサーバをスケールアウトする手順を自動化する

  1:00～1:10　Webサーバをシュリンクインする手順を検討する

  1:10～1:30　Webサーバをシュリンクインする手順を自動化する

- 目的
  - 掲示板サービスの性能が低下しない対策を実施する
    - 今回の演習では、アプリケーションサーバ(Web)の
      スケールアウトを実演する
    - また、スケールアウトの逆であるシュリンクインについても
      同様に検討したい
- 備考
  - Shell Script, Puppet, Mcollectiveを組み合わせ、1コマンドラインの
    入力で掲示板サービスをスケールアウト/シュリンクインさせられる
    仕組みについて設計すること

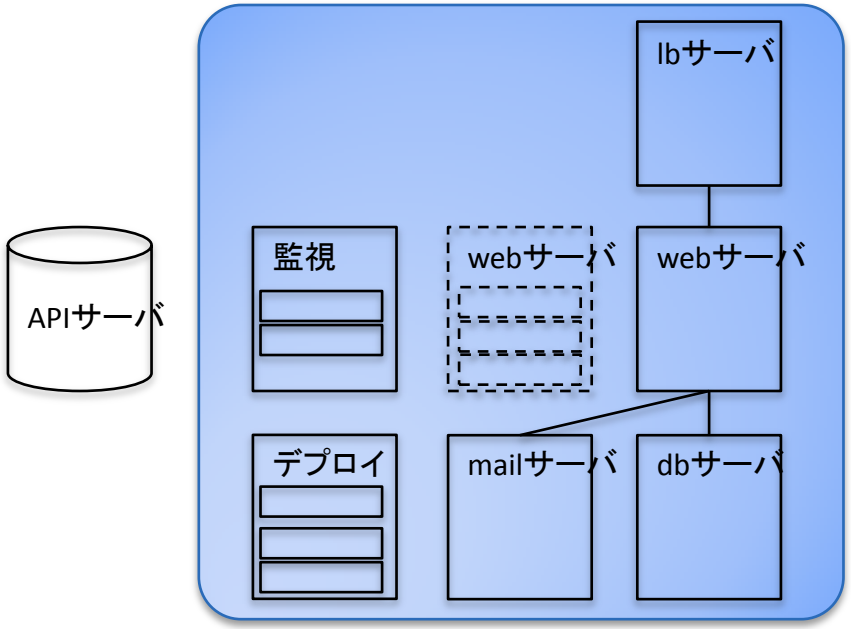
Webサーバをスケールアウトする

- キャンペーンを実施したいので、Webサーバの台数を増やしたり減らしたりする仕組みを作ること

- 台数を増やす、減らす作業を1ラインのコマンドで実施できるようになる必要がある
  - それぞれ実施に必要な手順を明らかにした後に自動化を検討する

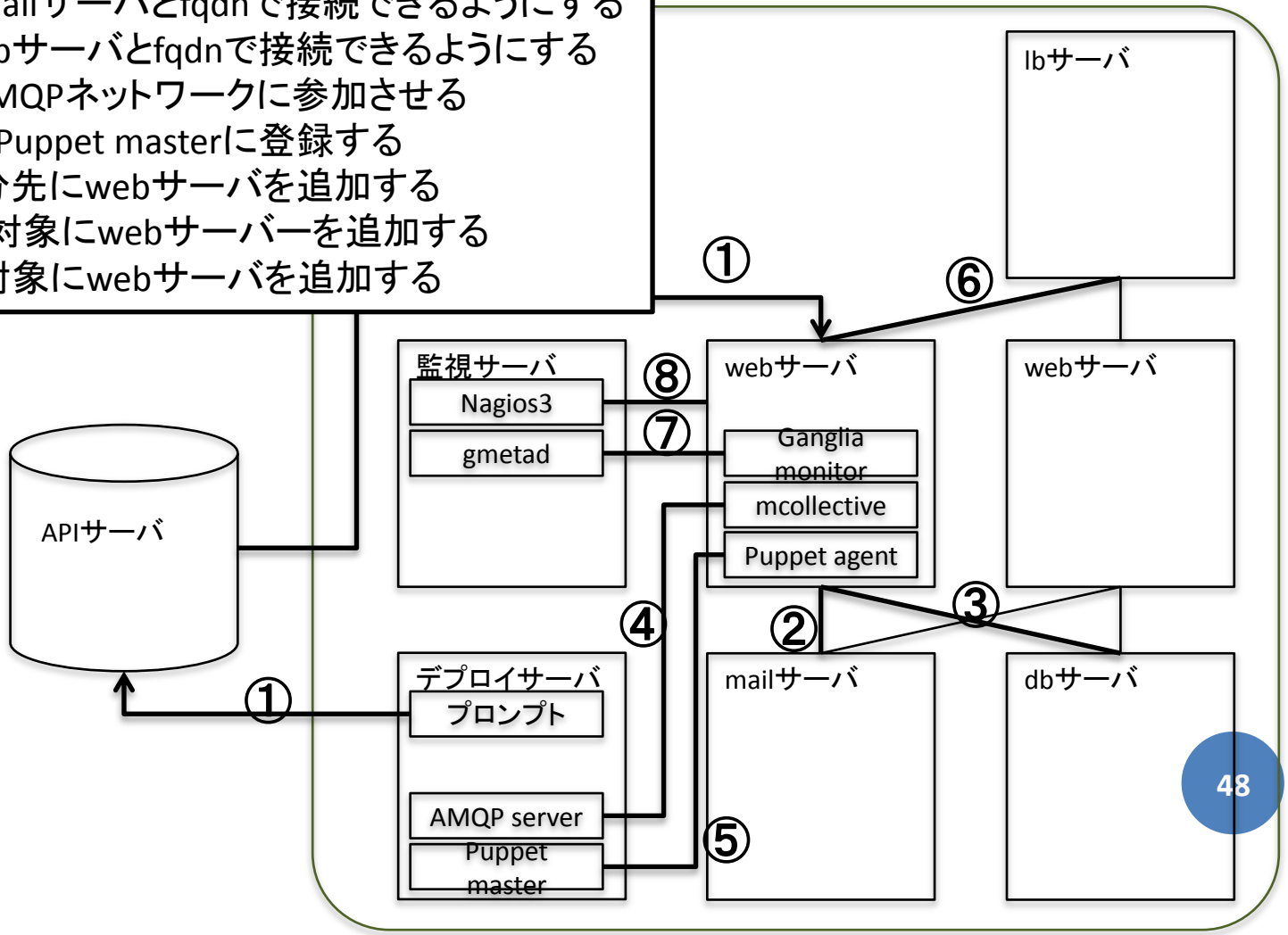
検討課題:Webサーバを追加する手順

- Webサーバを追加する手順を考えること
  - Webサーバを追加するためのヒントは、
    環境構築手順にある
  - チェックポイントを用意したので、
    考え方がわからないグループは参考にすること



解答例:WEBサーバを追加する手順

①webサーバを起動する
②webサーバがmailサーバとfqdnで接続できるようにする
③webサーバがdbサーバとfqdnで接続できるようにする
④mcollectiveをAMQPネットワークに参加させる
⑤Puppet agentをPuppet masterに登録する
⑥lbサーバの振分先にwebサーバを追加する
⑦Gangliaの監視対象にwebサーバーを追加する
⑧Nagiosの監視対象にwebサーバを追加する

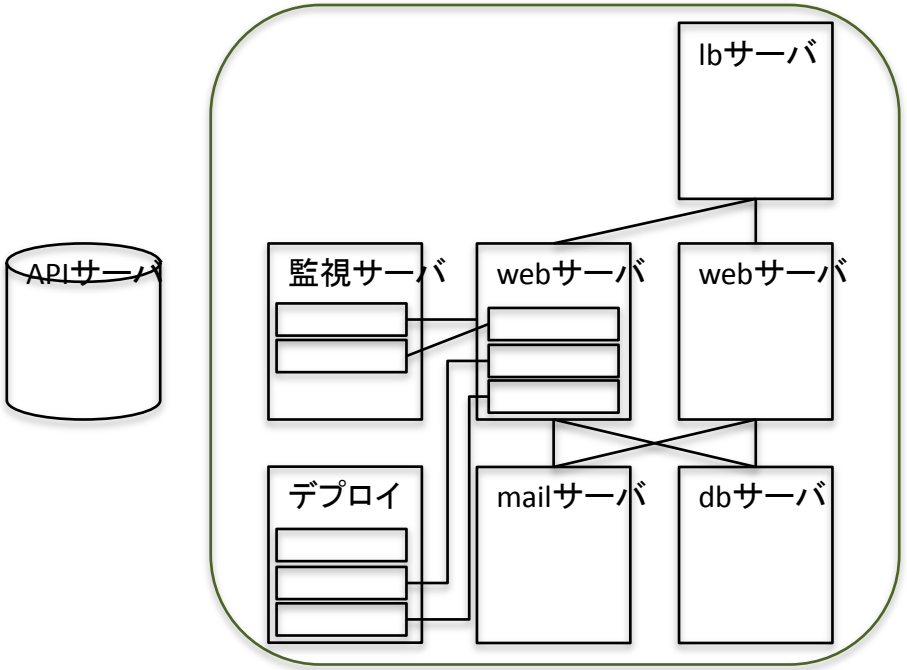


演習課題:Webサーバの追加を実行する

- 検討した手順に従ってコマンドを実行して
  Webサーバを追加しなさい
  - 解答例の手順に従った手順書が
    「04a_掲示板アプリケーションのスケールアウト」です。
- 手動で実行できたら、シェルスクリプトを作成し、
  Webサーバの追加を自動化しなさい
  - 解答例の手順に従ったスクリプトが
    「task/add_webnode.sh」です。

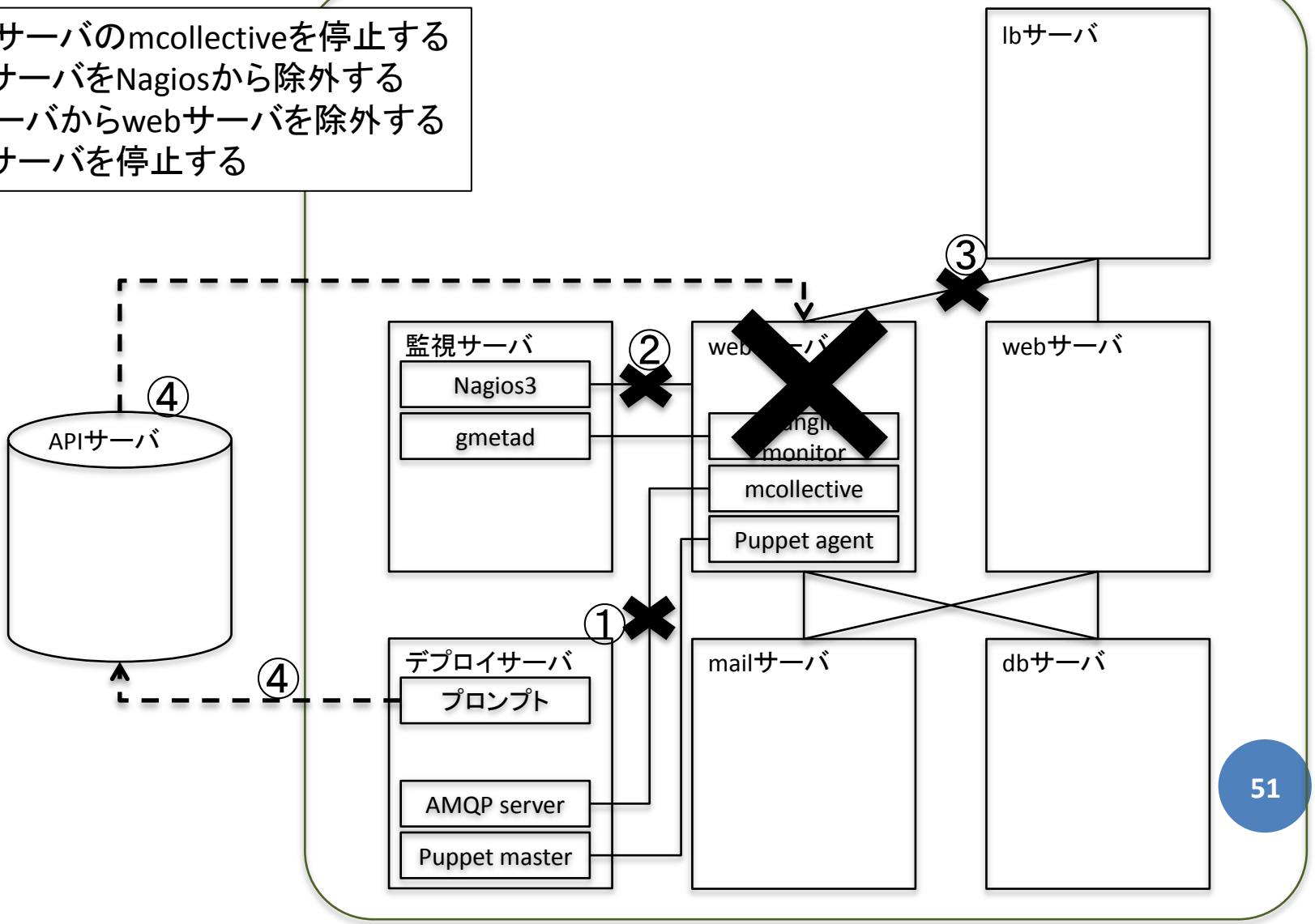
検討課題:Webサーバを削除する手順

- Webサーバを削除する手順を考えること
  - Webサーバをシャットダウンして影響がある接続を終了してからシャットダウンを行う。
  - チェックポイントを用意したので、考え方がわからないグループは参考にすること



解答例:WEBサーバを削除する手順

① webサーバのmcollectiveを停止する
②webサーバをNagiosから除外する
③ lbサーバからwebサーバを除外する
④webサーバを停止する



演習課題:Webサーバの削除を実行する

- 検討した手順に従ってコマンドを実行し、Webサーバを削除すること。
  - 解答例の手順に従った手順書が「04b_掲示板アプリケーションのシュリンクイン」である
- 手動で実行できたら、シェルスクリプトを作成し、Webサーバの削除を自動化すること
  - 解答例の手順に従ったスクリプトが「task/delete_webnode.sh」である

より高度なアプリケーションの配布

掲示板サービスの本番環境作成

Puppet(+MCollective)による自動化

Nagios + Gangliaを用いた監視の設定

掲示板サービスのスケールアウトを実現する

より高度なアプリケーションの配布

インスタンスが
シャットダウンしてしまう問題に対応する

一般ユーザからのフィードバックを受けて、
掲示板サービスは
随時バージョンアップの開発がされることになった。
チームではこれを素早く
本番環境へデプロイしなければならない。

演習の内容

- 時間割

  0:00～0:05　背景説明

  0:05～0:20　warファイルの単純デプロイスクリプト実行

  0:20～0:50　デプロイ時に発生するエラーの対応検討

  0:50～1:50　デプロイ時に発生するエラー対応実装

- 目的
  - 素早く、正確に実施したいので、手作業をなるべく減らして実現する必要がある
  - 一般ユーザがいつも通り掲示板サービスを利用してもらいながら、新しいバージョンの掲示板サービスを使えるようにする
- 備考
  - 現在のアプリケーションファイル(warファイル)をコピーするだけのデプロイ方式は対応が不十分である。よりユーザに影響の少ない方式を検討する。

アプリケーションをデプロイする

- Subversion とPuppetの復習
  - 開発担当はwarファイルをSubversionにコミットする
  - 運用担当は連絡を受けて最新版を
    本番環境にデプロイする
    - デプロイサーバからPuppetを用いて配布する

演習課題:WARファイルをデプロイする

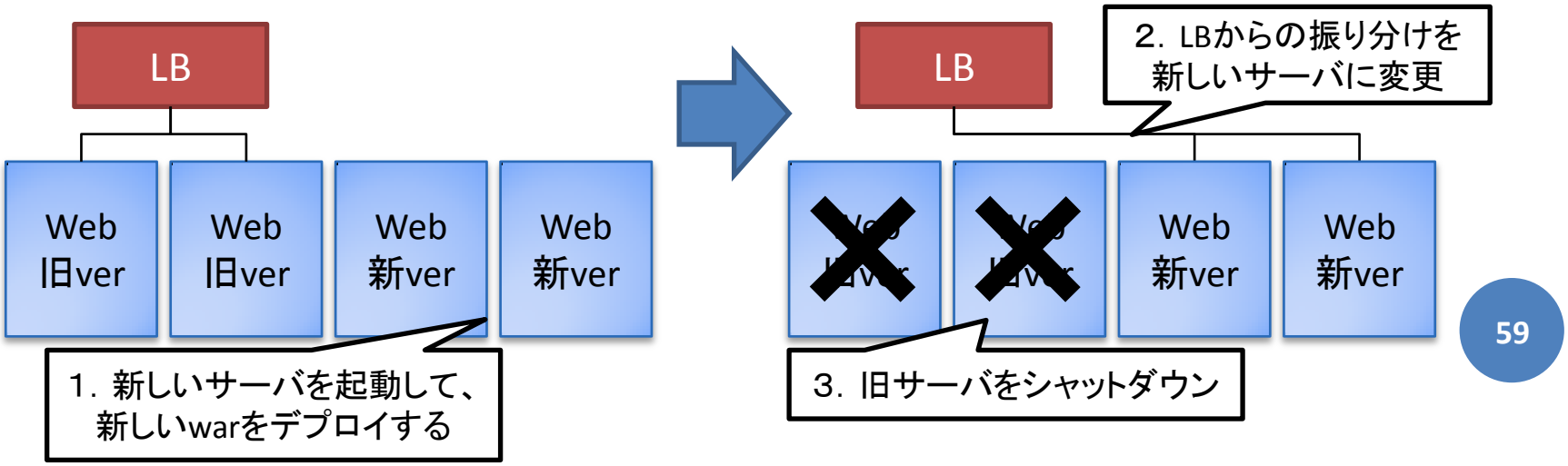
- 単純にwarファイルの配布を行うこと
  - Subversionからwarファイルを取得し、
    デプロイする方法を検討すること
  - 「アプリケーションの配布手順書」を参考に
    検討した方法で実行すること

検討課題:WARファイル配布時に404エラーが発生する事象の回避

- ログを確認するとwarを配布した後、404エラーを返している
  - TOMCATがwarファイルを展開している最中にリクエストが届くと発生している
- このままでは、サービスを行いながら本番環境にデプロイすることができないので、回避する方法を検討すること

解答例:WARファイルをデプロイした時に発生するエラーを回避する

- 起動しているWebサーバ同じ台数のWebサーバを構築しロードバランサを切り替える
  - サービスレベルを変えずにデプロイすることができる
  - サーバの起動、設定を行うため、既存サーバを使うより時間がかかる
  - 課金があるクラウドを利用している場合、一時的に倍の課金が発生する



演習課題:404エラーが発生しないWARの配布

- 検討した手順で404エラーが発生しないwarの配布を行うこと
  - 「より高度なアプリケーションの配布手順書」を実施すると解答例で示した方法を試すことができる
- 手動で実行できたら自動化すること
  - 下記は解答例で示した方法の自動化スクリプトである

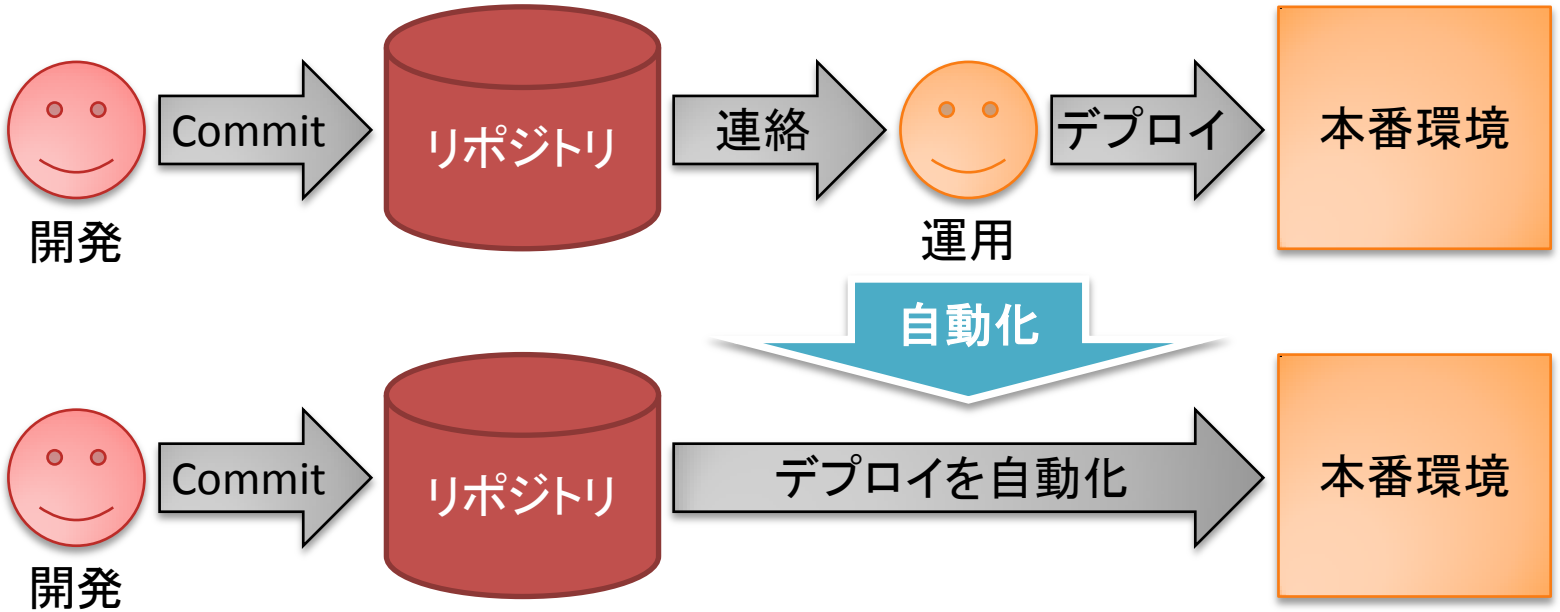
    /root/work/deploy/task/graceful_deploy_war.sh

発展課題:デプロイを自動化する

- 人がコマンドを実行するする運用となっているため、ミスや時間のロスが発生することがある
  - 自動化する方法はないか

解答例

- リポジトリへのコミットが完了したらデプロイする
- 解答の実施
  - 「デプロイを自動化する」手順を参照して実施する



DBのマイグレート

- DBのマイグレートとは以下の要素を持つ
  - スキーマの操作
  - データの操作(マスタデータレコードの挿入/削除等)
- アプリケーションのバージョンに合わせて実施する必要がある
  - 本番環境を運用していく中で、DB定義を変更することがある
  - ロールバック可能なマイグレート方法を取る必要がある場合もある
  - ルールはプロジェクトごとに検討する必要がある
    - オンラインで行ってよいのか
    - 適用の手順は何が適しているのか
- DBマイグレーションツールを使う方法がある
  - ツールが本番のDB状態をバージョン番号で管理する
  - 適用とロールバックをスクリプトで記述し、実行順番を規定する
  - 例）db:migrate(Ruby on Rails), Doctrin(PHP)など

インスタンスが
シャットダウンしてしまう問題に対応

掲示板サービスの本番環境作成

Puppet(+MCollective)による自動化

Nagios + Gangliaを用いた監視の設定

掲示板サービスのスケールアウトを実現する

より高度なアプリケーションの配布

インスタンスが
シャットダウンしてしまう問題に対応する

IaaSを利用していると、
ある日突然インスタンスが
シャットダウンする
と言う現象にあたった。
Nagiosはそれを検知したもの
の、人手の対応までに
時間を要してしまった。
これを改善したい。

演習の内容

- 時間割

  0:00～0:10　インスタンスがシャットダウンしてしまう現象

  0:10～0:25　インスタンスがシャットダウンする

  0:25～1:00　インスタンスのシャットダウンと自動復旧を実装する

  1:00～2:00　自動復旧をセットアップする

- 目的
  - 掲示板サービスの性能が低下しない対策を実施する
    - 今回の演習では、アプリケーションサーバ(Web)の
      スケールアウトを実演する
    - また、スケールアウトの逆であるシュリンクインについても
      同様に検討したい
- 備考
  - Shell Script, Puppet, Mcollectiveを組み合わせ、1コマンドラインの
    入力で掲示板サービスをスケールアウト/シュリンクインさせられる
    仕組みについて設計すること

IaaSでは不意にインスタンスにアクセスできなくなる

- クラウドを利用していると不意にインスタンスにアクセスできなくなることがある
  - シャットダウン
  - ネットワークの問題
  - サーバ過負荷
  - ハード故障　など

アクセスできない時の対処方法

- クラウドでは原因をインスタンスから調査することはできないため、対症療法を検討する
  - 冗長化して1台にアクセスできなくてもサービスが継続するようにする
    - 同時に、シャットダウン時に別のインスタンスを補充する
    - 今回のWebサーバ
  - 冗長化しない/できない場合
    - EBS起動のインスタンスは、Volumeから再起動を行う
      - S3インスタンスより起動は速いがアクセスできない時間がある
      - 今回のLB、DBサーバ
    - S3インスタンスは別インスタンスを起動し設定を行う
      - 監視サーバ、デプロイサーバ

本演習ではWEBサーバを対象にする

- Webサーバを追加することで対応できる
- 監視サーバがアクセスできなくなった場合に、Webサーバを追加する仕組みがあれば障害に自動対処できる
  - 起動するスクリプトはデプロイサーバにあるため、デプロイサーバで実行する

演習課題:Webサーバを1台シャットダウンする

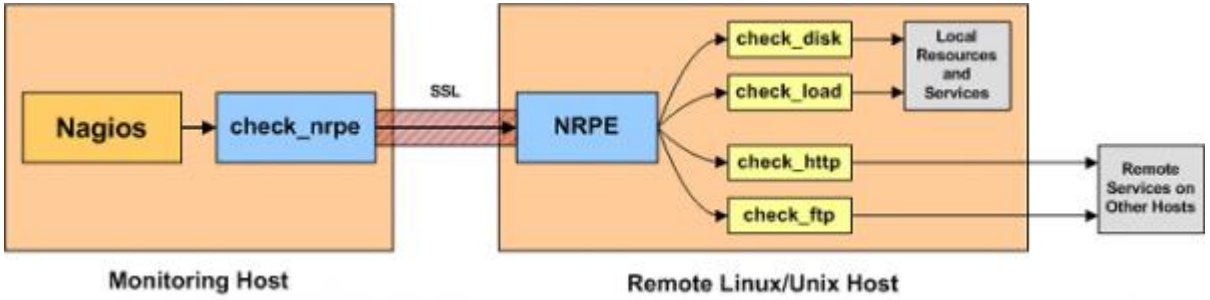
- Webサーバが2台以上立ちあがっていない場合は、Webサーバの追加を行う
- Cloud ClientからWebサーバを1台シャットダウンし、Nagiosの画面で確認する

Webサーバがシャットダウンした時に起こせること

- 現状はNagiosが検知した結果を、ブラウザで確認できる
- 通知方法の色々
  - ブラウザで確認する
  - メールで通知する
  - Nagios Pluginを実行する

監視サーバから、デプロイサーバ上でスクリプトを実行する

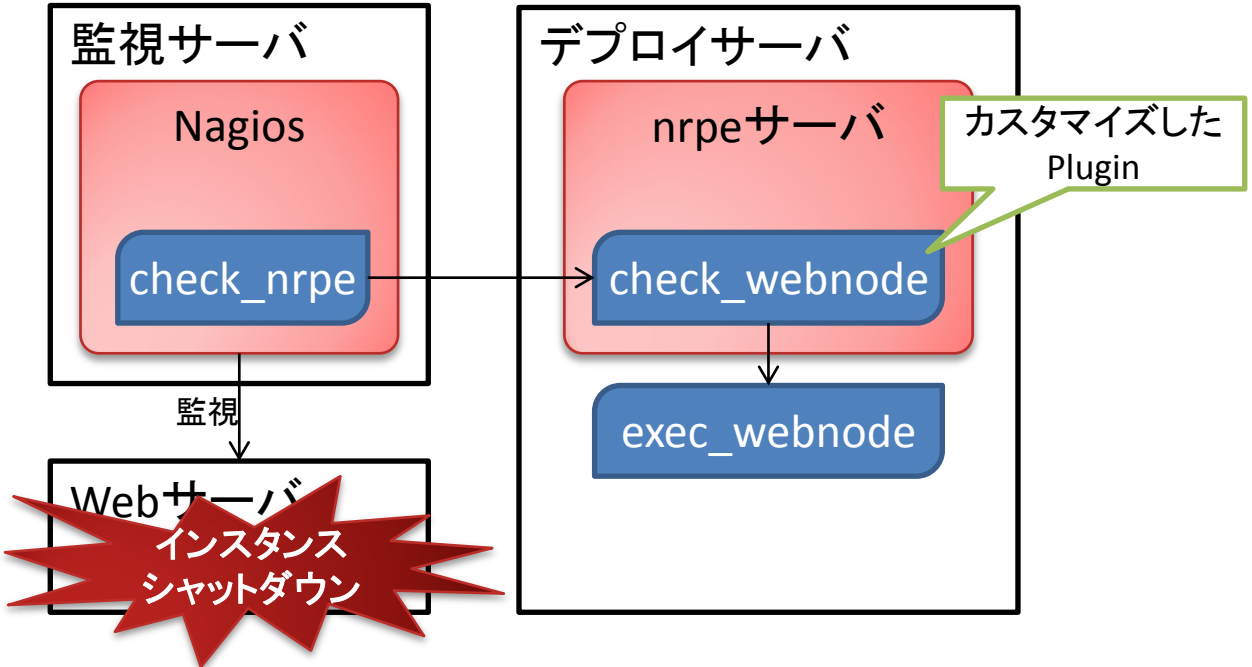
- 監視の結果により発生するイベントでデプロイサーバのスクリプトを実行する
- Nagios Remote Plugin Executor (NRPE)
  - Nagiosが動作する監視サーバが、リモートホスト上でNagiosのPluginを実行する
    - リモートホストにはデプロイサーバを指定する
    - カスタマイズしたPluginを使用する



引用：http://exchange.nagios.org/directory/image/93

本演習における
シャットダウン自動復旧の流れ

- Webサーバのインスタンスがシャットダウンすると、Nagiosが検知する。
- 設定にしたがってNagiosがcheck_nrpeを実行する
  - 実行時にデプロイサーバのcheck_webnodeを指定する
- デプロイサーバでは、MCollectiveを経由してexec_webnodeを実行する
  - exec_webnodeに復旧するためのスクリプトを記述する



検討課題:シャットダウンが起きた時の対処

- Webサーバがシャットダウンした時に
  どのような対処を行うのがよいか検討すること

解答例:シャットダウンが起きた時の対処

- Webサーバがシャットダウンした時には
  サーバを1台追加して元の状態に戻す
  - add_webnodeスクリプトを呼び出す

実技課題:シャットダウン対応の実施

- 「 インスタンスがシャットダウンしてしまう問題に対応する」にしたがって自動復旧のための設定を行う
  - exec_webnodeに復旧するためのスクリプトを記述すること
- 再度Cloud Clientからシャットダウンする
- 意図的にWebサーバを1台減らしても、自動復旧が実行されないことを確認する

成果発表会

アクセス結果発表

- 先ほどユーザを模して、掲示板サービスにアクセスしていたJMeterを停止しました。
- 集計結果を発表します。

報告会資料の作成(10分)

- 10分で終わる内容にしてください
- 下記トピックから、特に興味のある課題を1つ以上選択して、発表の題材としてください
  - スケールアウトの実現
  - アプリケーションの配布（デプロイ）
  - 予期せぬシャットダウンへの対処
  - 計画的にスケールアウトする
  - ユーザ登録が出来なくなる問題に対応する
- 共通的な発表内容として、下記する観点でまとめてみてください
  - 実際に苦労をしたところ
  - 演習課題で実施したことから、さらに実践的にするため、どのような工夫が考えられるか？
  - 身の周りのプロジェクトへ適用するために、演習で実施した内容に加えて、どのような検討が必要か？

提出課題

提出課題目次

- どちらかを選択して提出すること

1. デプロイを自動化する
2. 計画的にスケールアウトする

デプロイを自動化する

提出課題：デプロイを自動化する

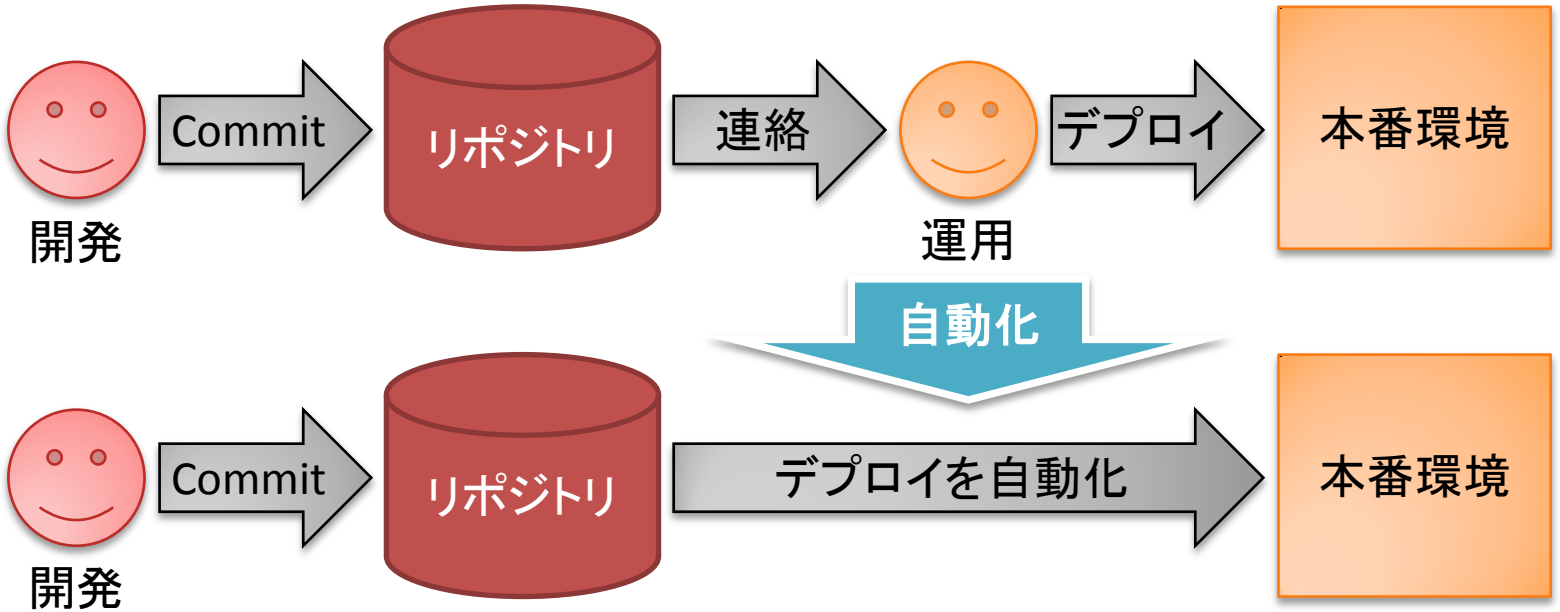
- デプロイのミスを減らすため、デプロイを自動化したい。
  スライドP61の「発展課題：デプロイを自動化する」を
  実施しなさい。
  また、使用したスクリプトおよび設定を提出しなさい。

（再掲）発展課題:デプロイを自動化する

- 人がコマンドを実行するする運用となっているため、ミスや時間のロスが発生することがある
  - 自動化する方法はないか

解答例

- リポジトリへのコミットが完了したらデプロイする
- 解答の実施
  - 「デプロイを自動化する」手順を参照して実施する



計画的にスケールアウトする

提出課題:計画的スケールアウトの実施

- Webサーバの台数を下記のように時間帯に合わせて変化させたい。必要なスクリプトの実装や設定を行い、実現しなさい。
  また、その際に使用したスクリプトおよび設定を提出しなさい。
  - 9:00～21:00　　6台
  - 21:00～翌朝9:00　2台

計画的にスケールアウトする

- 背景
  - アクセスの統計をみていると、
    日中は負荷が高く夜になると負荷が下がることが分かった
  - スケールアウトを使って負荷が増えても今まで通りサービス
    を継続したい
    - 例）昼は6台で稼働し、夜には2台に縮小したい
- 目的
  - 起動するサーバの台数を指定するスクリプトによって
    スケールアウト/シュリンクインを自動的に制御する
- 内容
  - 計画的スケールアウトの実現方法検討
  - 計画的スケールアウトの実装

計画的スケールアウトの必要性

- キャンペーンを実施するために、
  マシン台数を大幅に増やす必要がある
  - 1台増やす作業を5分で終わるように自動化しても、
    100台増やすためには500分(8時間20分)以上かかる

検討課題:計画的スケールアウトの実装方法

- 昼間だけのキャンペーンを行うため、
  昼は100台で稼働し夜は15台で稼働する環境を作る
- どのように実装すべきか検討する
  - 9:00～21:00　　100台
  - 21:00～翌朝9:00　15台

解答:計画的スケールアウトの実装方法

- 指定した台数を引数にして、
  追加と削除を組み合わせるスクリプトを実装する
  - 増減の台数を指定するスクリプトを作る。
  - 21時に75台増やして、9時に75台減らす方式はNG
  - インスタンスがシャットダウンするなどの原因で、
    台数がずれても修正できないため
- 上記スクリプトを指定時刻に起動するようcrontabで
  設定する