ジョブ管理の技 17

スケールイン・スケールアウトするインスタンスに対するジョブ運用を行いたい

現状の課題

システムのオートスケールにはさまざまなメリットがある。特に、ビジネスの状況に応じ、システム基盤の増強や変更が柔軟に行えることが大きなメリットである。例えば、従来であれば基盤の設計・構築からアプリケーションの設計・構築まで 3 カ月掛かったものが、すでに構築されているクラウド基盤の上にアプリケーションを稼働させるだけで利用可能になるため、数週間でリリースできるようなケースがある。特に、利用者の増減などによるトランザクションの変化には柔軟かつ効率的に対応できるようになる。

システム変更のスピードが上がり、柔軟性が向上する一方、ジョブ管理の面では課題もある。ジョブは各インスタンスで実行されるため、ジョブ定義をインスタンスごとに行う必要がある。つまり、インスタンスの増減に比例してジョブ定義登録・変更作業も発生する。

他のインスタンスで動かしているジョブと全く同じであるにもかかわらず、インスタンスごとにジョブを追加登録しなければならない。逆にインスタンスが減る際には、該当インスタンスのジョブ定義を削除する必要がある。スケーラブルなシステムを構築しようとしても、ジョブ定義の再構成がボトルネックとなってスピーディーな構築が妨げられてしまう。

解決策

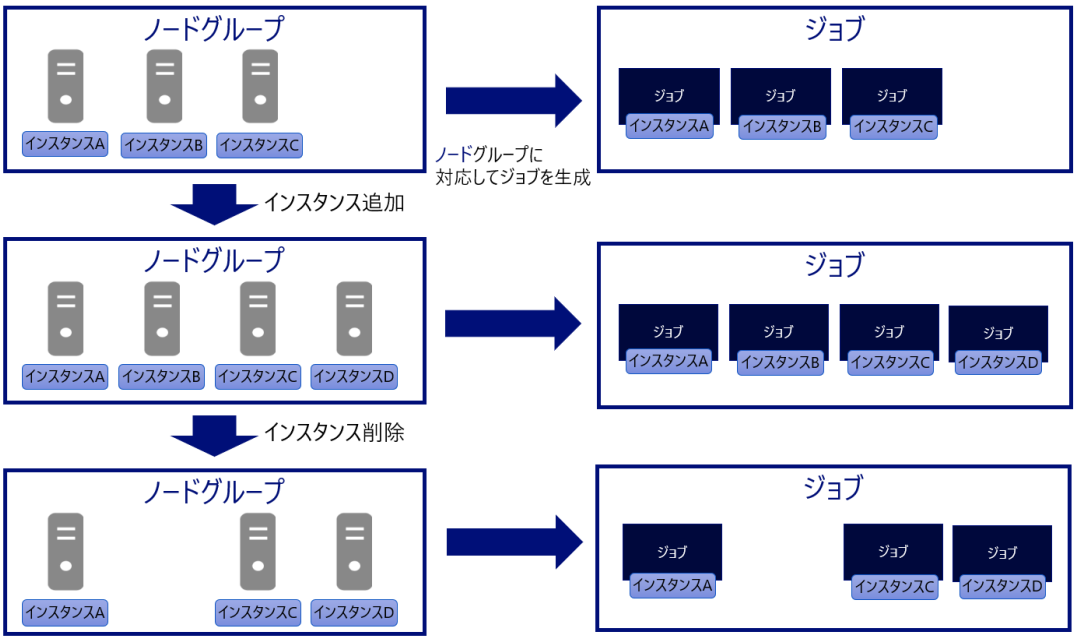
頻繁に変化するインスタンス数に応じて、自動的にジョブ定義を再構成できればスピーディーなシステム変更が可能になる。インスタンスの追加時には追加インスタンスに自動的にジョブを定義し、インスタンス削除時にはジョブの実行をやめるのである。

スケーラブルなシステムの場合、トランザクション増加などのイベントをトリガーにしてインスタンスを増減させたい場合があるが、そのような動的なシステム変更に対しては、プロビジョニング作業に組み込むことにより効率化できる。

Senju Family での実践方法

Senju DevOperation Conductor では指定したグループに所属するインスタンス内でジョブを実行する方法が 2 種類存在する。一つ目はノードグループ内のインスタンス全てに対して、同一のジョブを実行するグループネット機能を提供している。また、もう一つはラウンドロビン方式で、ノードグループ内の 1 インスタンスでジョブを実行する機能を提供している。これにより、インスタンス間で負荷を分散してジョブを実行することができる。また、負荷分散を行うインスタンスも後から自由にノードグループに追加・削除が可能である。

ノードグループにジョブを定義しておけば、ノードグループに後から加えたインスタンスでもそのジョブが実行されるようになる。ノードグループからインスタンスを外せば、そのインスタ

ンスではジョブが実行されなくなる。

ノードグループへのインスタンスの追加・削除は、コマンドで動的に制御することも可能である。メッセージアクション機能やランブックオートメーション機能と組み合わせ、インスタンスのリソース状況(CPU やメモリー使用率)が異常になったらインスタンスを追加し、さらにジョブ定義やジョブ実行まで自動化することができる。