Enterprise Architecture

Enterprise Architecture Technical Board

Multiple Applications on a Compute Instance

Editor: Greg Charest

ETAB Workgroup Members:

Greg Charest
Colin Murtaugh
Ben Rota
Tom Vachon

Version:	1.0
Last Revised:	August 8, 2017
Status:	Final
Document Type:	Guidance

Audience Level: Management, Technical

Topic Statement:

Installing multiple small applications on a single EC2 instance may result in cost savings but this practice has significant serviceability and resilience implications.

Complete topic description:
https://confluence.huit.harvard.edu/display/ETAB/Maximum+production+applications+on+a+compute+instance

Executive Summary:

This document outlines the potential problems associated with multiple applications running on a single instance, describes the current best practice for isolating production applications, and makes recommendations that should be applied to mitigate potential problems when the best practice is not followed and multiple applications share a single EC2 instance.

Introduction:

Some existing on-premises environments have been designed to run multiple applications on a single server instance due to the low level of resources required by each application. This leads to additional complexity and lack of flexibility in a cloud environment. As we migrate to the cloud, we would like to take advantage of the opportunity to isolate production environments onto dedicated EC2 instances.

Discussion:

Shared environments create or contribute to the following problems:

- **Single point of failure.** The shared instance represents a single point of failure for multiple applications. Although auto scaling may provide some resilience, individual application level failures may not be caught by the automated health check.
- **Lack of maintainability.** The ease with which an individual application in a shared environment can be modified, improved, or adapted to a changed environment is greatly reduced. Upgrades, patches and other activities may entail a service outage for multiple applications.
- **Increased change/incident management.** An instance level failure or change may result in multiple service incidents that must be managed and resolved. In addition, patch/release schedules will need to be coordinated and changes will likely take longer to execute.

- **Reduced ability to utilize certain cloud cost saving techniques**. Unlike on-premises physical servers, cloud based instances can be shut down or even deleted when they not needed (for example, applications only used at certain times of the year). Implementing this cost savings strategy is much more difficult when a single instance supports multiple applications.
- **Software library dependencies.** Applications may rely on the implicit existence of different system-wide support packages. Collisions between differing versions of the libraries required for different applications greatly complicate patching and upgrade processes.

Although the above discussion is focused on multiple applications on an instance, it should be noted that similar issues arise when multiple environments (dev/stage/test) are hosted on a single instance.

Recommendations

- The present 'best practice' is to isolate production applications on their own instance. This architectural pattern, while potentially costlier, eliminates the above problems and provides additional flexibility.
- In the event that budgetary resources do not permit this approach, the issues described above should be carefully considered prior to a decision to place multiple applications on a single instance. An appropriate exception process should be used, the decision should be properly documented, and a re-evaluation should be scheduled at an appropriate time.
- The issues related to maintainability and dependencies can be mitigated to a certain degree. Potential tools and approaches should be evaluated during the development/installation/migration process. These might include:
    - dependency declaration and dependency isolation. Example include Python/Pip/virtualenv, Ruby Gemfile/bundle exec.
    - The use of application containerization technology such as Docker containers.
- Application databases should be independent. In relational terms, the tables, fields, relationships, views, etc. should be organized into individual schemas that can be managed using the relevant relational database management tools. If a database acts as a data store for multiple **related** applications (an integration database), a single schema can be used. It is important to note however, that this may result in deep coupling that significantly increases the risk involved in changing the supported applications.