"Jobs-as-Code" Approach Infuses DevOps Focus into Job Scheduling, Workload Automation

Julie Craig
Research Director
Application Management
Enterprise Management Associates
Read Julie's blog
Follow Julie on Twitter @julie_craig

Overview

"Jobs-as-Code" is an emerging industry concept that has the potential to change the way IT Operations organizations operate.

Job Scheduling and Workload Automation center on standard operational processes designed to ensure that orchestrated job streams—batch jobs, in particular—execute reliably, in the correct sequence, and with enough reporting to provide feedback on whether jobs complete normally and on time. Such automation is a major improvement on the manual processes of the past, in which Operations staff manually ran commands, processes, and scripts in a specific sequence to drive prescribed operational processes for a given company.

At the same time, as the workloads and services supported by the average IT organization have become increasingly complex, a need for new ways to control job runs of all types has become glaringly apparent. Particularly with the rise of DevOps and Continuous Delivery practices that have dramatically accelerated the speed with which new and modified software services are introduced to production, automation of job streams supporting Workload Automation solutions can rapidly drift out of sync with the actual jobs being run. Too often, job run sequences must be patched with new or modified scripts by Operations personnel who may or may not be intimately familiar with the overall process.

At the same time, "DevOps" practices—centered on teams of senior personnel with cross-functional skill sets working collaboratively across development-focused and operationally focused stages of the application lifecycle—have now become the norm in the majority of companies. The role of these teams is to provide the expertise necessary to support the development, deployment, monitoring, and ongoing management of complex IT applications and services. Enterprise Management Associates (EMA) research studies have repeatedly found that those companies most successful at promoting high-quality collaboration between Development and Operations are those most likely to see exceptional revenue growth.

Still, it is too often the case that, although these cross-functional DevOps teams are in place, the Development and Operations organizations themselves still operate separately. New applications and services are signed off and "tossed over the wall" for Operations to deal with and maintain. At the same time the complexity of these services demands an increasingly granular level of knowledge about the actual steps required to run a given application, job, or process.

The Jobs-as-Code concept allows Development to encapsulate granular knowledge about applications and services into the programming environments they utilize as part of their day-to-day work. This "shift left" in terms of workstream specification means that the steps required to execute a given job become part of the code itself.

The Jobs-as-Code concept allows Development to encapsulate granular knowledge about applications and services into the programming environments they utilize as part of their day-to-day work. This "shift left" in terms of workstream specification means that the steps required to execute a given job become part of the code itself.

the steps required to execute a given job become part of the code itself. This allows automation details to be versioned and encapsulated within standard deployment packages and replaces traditional, one-off scripts with purpose-built code addressing the requirements of each specific workload.

This EMA Impact Brief explores these concepts in greater depth and highlights the value propositions of adopting a "shift left" mentality and approach to workload processing.

Jobs-as-Code: What Does this Term Mean?

Many in the industry are starting to recognize the value proposition of the Jobs-as-Code concept. For example, the Jenkins "Pipeline as Code" concept also proposes an alternative to the User Interface (UI) typically utilized to create jobs.

Jenkins defines the concept as follows: "Users now can implement a project's entire build/test/deploy pipeline in a Jenkins file and store that alongside their code, treating their pipeline as another piece of code checked into source control."[1]

Some leading-edge vendors have come to a similar conclusion and incorporated similar concepts into their product lines. BMC, for example, has added Jobs-as-Code to its Control-M product, enabling many of the tasks that have traditionally consumed time and resources on the Operations side to be addressed earlier in the lifecycle by the developers who truly understand the application (see Figure 1).

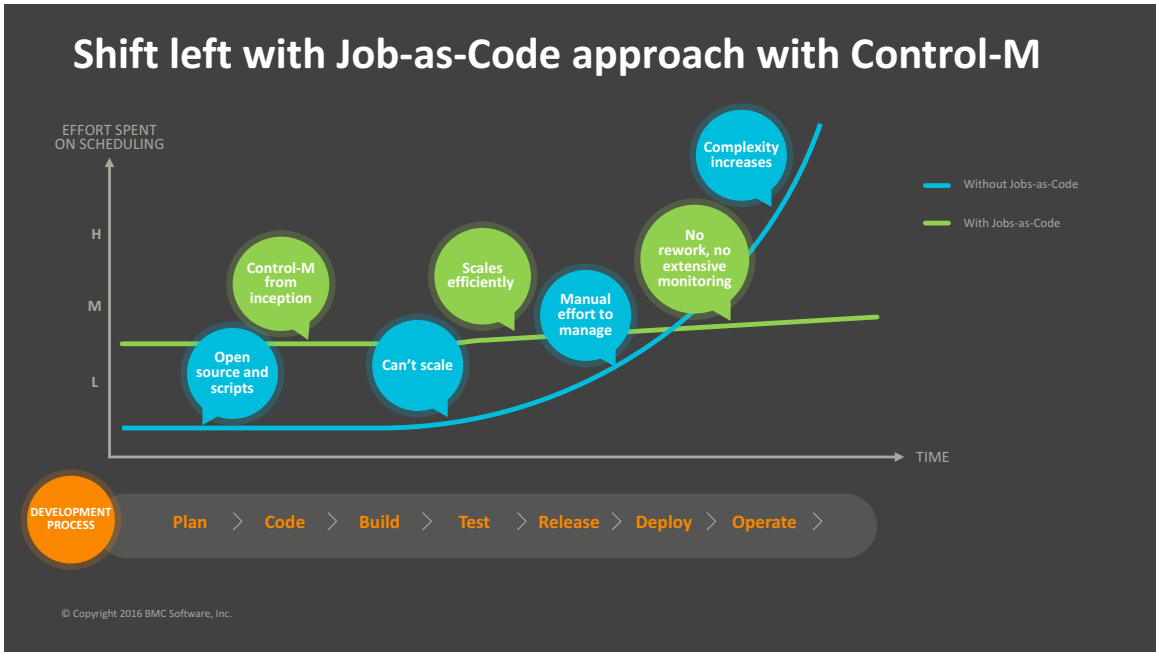

Figure 1. Jobs-as-Code with Control-M shifts job scheduling specification tasks to earlier stages of the lifecycle

Automating the Service Lifecycle

DevOps concepts have been incorporated into a broad swath of enterprise management toolsets, most notably those addressing application management, software testing, software deployment, and Release Automation. However Workload Automation is one category of tooling that has largely been ignored in the context of the DevOps revolution. At the same time, while software has become increasingly critical to business success, the tasks associated with running production workloads have become increasingly complex.

[1] Additional information available at jenkins.io/solutions/pipeline

Spanning multiple processes and steps, batch jobs often leverage scripts developed on an ad hoc basis by the Operations teams tasked with managing job scheduling and execution. Increasingly, products such BMC's Control-M Workload Automation are replacing manual processes and enabling complex, multi-step jobs to be automated. While this brings a great deal of efficiency to the process, many companies running complex software processes eventually discover that simply automating an existing process may not be enough. Often, the process itself must be changed to modify the job stream and/or the software supporting it.

Once software hits production, Operations personnel who are unfamiliar with the code underlying the software are often tasked with writing scripts to modify job stream processing or recover from a failure. Creating, adding, and maintaining scripts over time consumes staff resources and intensifies workload complexity as new scripts are added to already complex job streams. In addition, ad hoc script modification can disrupt business as usual, force Operations specialists to take time out from scheduled work to rectify faulty job streams. Over time, batch runs become increasingly complex, unwieldy, and difficult to maintain.

Traditional Workload Automation: Both a Boon and a Burden

Traditionally, Graphical User Interfaces (GUIs) have enabled Operations teams to issue commands, run scripts, and otherwise interact with Workload Automation tools. GUIs simplify the process of running standard workloads, particularly when those workloads are tested, predictable, and normally error-free.

However, there are multiple problems with this approach:

- **"Megascripts"** – Operations teams complain that they receive "megascripts" from Development and that when these scripts fail, they are a "disaster to debug." Operations specialists—who are often unfamiliar with the code behind the job being run—are forced to troubleshoot and, too often, to create additional scripts to address that problem in future runs. This creates "script sprawl," takes time away from scheduled tasks, interrupts the batch schedule, and often results in long-running operational processes that can adversely impact production processing.
- **Time-consuming scripting** – Scripting job runs requires significant amounts of time, and maintaining scripts as jobs change takes even more time. Since most job runs require multiple scripts to execute, the process of keeping track of which script goes with which job can be mindboggling, particularly for new employees or those with less on-the-job experience.
- **Silo versus lifecycle-focused approach** – In most cases, DevOps is not even a part of the workload automation conversation. While DevOps teams are now in place at most companies and DevOps practices have permeated the lifecycle, batch jobs themselves have often not been updated.
- **Limited access to GUI-enabled toolsets** – Due to cost and licensing-related factors, only production-facing Operations teams typically have access to Workload Automation tools. So although developers often write the scripts supporting job runs, they can't test the code in the actual tool that will be used to run it.

Jobs-as-Code: A Revolutionary Concept and Better Way

BMC launched its first foray into Jobs-as-Code in June 2016 with the Control-M Automation API offering. The June 2017 announcement built on the same concept with a new Workbench component. Workbench enables developers to test and debug Jobs-as-Code on their personal computers or at a workgroup level without having to contact Operations for access to the Control-M Automation GUI.

Intended to further adoption of the Jobs-as-Code approach, these capabilities answer the needs of modern IT by providing a simple way to enable those most familiar with the software jobs being run—the developers—to design, code, and specify Job Scheduling and workload execution processes.

BMC has exposed a large swath of Control-M commands and processes via APIs. It has also provided a simple JavaScript Object Notation (JSON)–based methodology to build job streams via code. The intent is to eliminate the need to do scripting, since the most popular Control-M commands—and the entire job stream—are now accessible via code.

A new developer-focused Workbench provides an Integrated Development Environment (IDE)–like approach to building jobs. It allows developers to build jobs locally with no network connection required and to iterate and test as they normally would with IDEs.

BMC has exposed a large swath of Control-M commands and processes via APIs. It has also provided a simple JavaScript Object Notation (JSON)–based methodology to build job streams via code. The intent is to eliminate the need to do scripting, since the most popular Control-M commands—and the entire job stream—are now accessible via code.

Control-M has included, for example, support for debugging using breakpoints and similar features that a developer would expect when developing in Java or C. By shifting left in terms of job stream development, this approach enables development teams to incorporate standard DevOps and lifecycle practices into Workload Automation processes via code versus GUI access.

The Control-M Jobs-as-Code concept includes:

- **Developer access to Control-M functions** – With the latest release, developers now have access to the functionality underlying Control-M without having to evoke the GUI controlling Control-M in the data center. Developers access this functionality in a familiar, IDE-like interface that is similar to the development environments they are used to.
- **New notation** – A new standard notation provides developer access and eliminates the need to access GUIs to take advantage of Control-M capabilities. Code can be incorporated into GIT repositories and software packages, versioned, and promoted across the lifecycle along with application code.
- **RESTful APIs** – BMC's modern RESTful APIs, natively speaking JSON, provide the interaction between code and Control-M. While developers must still understand the syntax, the language structure is familiar. This reduces learning curves and replaces script-based job streams with API-controlled services whose development becomes part of the DevOps lifecycle.
- **Workbench** – Developers can verify coding via a built-in function that checks and validates JSON code. The Workbench can be downloaded locally and operated without a network connection. Developers have full control, autonomy, and ownership of all components without having to rely on licenses or direct access to the production instance of Control-M. Code can be submitted to builds within code management systems and can then be promoted to subsequent lifecycle stages as part of standard application code packages. The Workbench includes multiple features designed to assist developers in building job streams that address the requirements of a given batch job (see Figure 2).
- **Script replacement***:* Jobs-as-Code can replace scripting with API-driven code. Programs can be invoked, environments can be created, and other functions completed, all via the notation system.

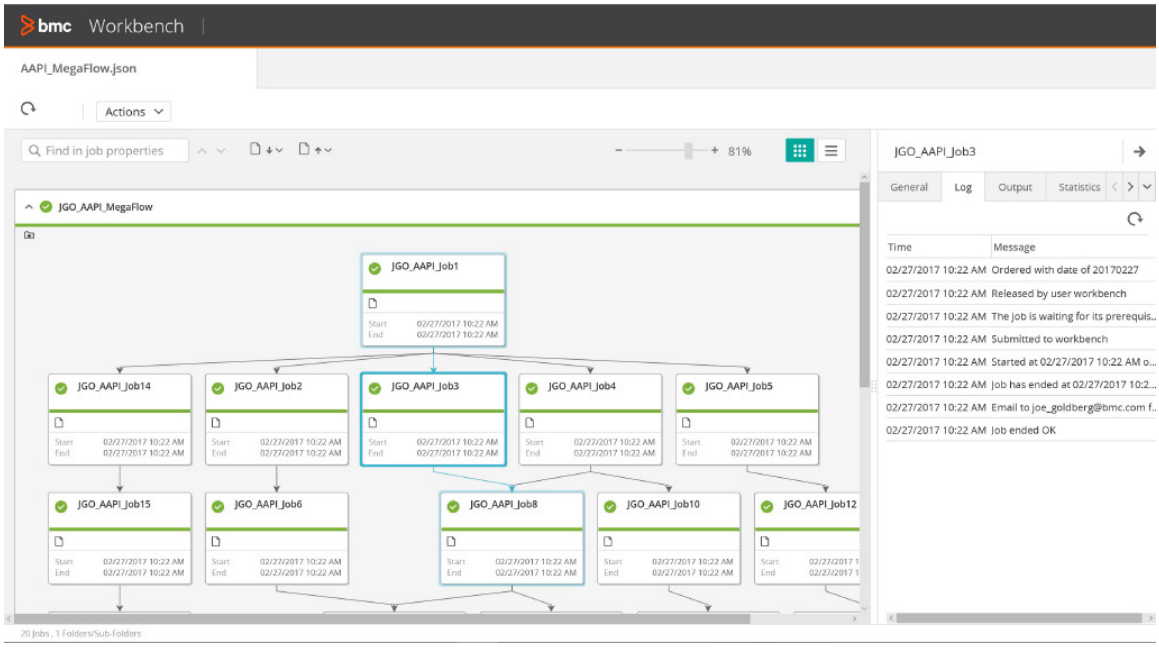

Figure 2. Workbench Megaflow diagram

EMA Perspective

Batch jobs are a daily reality for the vast majority of enterprise-sized companies. Many have invested in Workload Automation solutions, but even after such investments may find themselves mired in the script-based processes of the past. "Megascripts," runbooks, execution errors, and changing software requirements can make batch processes unpredictable at best and, in the worst case scenario, detrimental to business as usual for IT's customers. Maintaining scripts and recovering from failed job sequences requires time and effort from IT Operations personnel whose skills could be better used to address new business-differentiating projects.

With its latest Jobs-as-Code functions, BMC has introduced significant new features to the ControlM solution that benefit both current customers and prospective buyers. A product formerly aimed primarily at Operations can now be used—essentially free of charge—by Development as well. These new capabilities provide a basis for achieving the true purpose of DevOps, which should be cross-functional collaboration, wherever it is needed and at any stage of the application lifecycle.

With its latest Jobs-as-Code functions, BMC has introduced significant new features to the Control M solution that benefit both current customers and prospective buyers. A product formerly aimed primarily at Operations can now be used—essentially free of charge—by Development as well. These new capabilities provide a basis for achieving the true purpose of DevOps, which should be cross-functional collaboration, wherever it is needed and at any stage of the application lifecycle.

Additional Reading

"Jenkins jobs as code: the Job DSL plugin" is available at developer.epages.com/blog/2016/01/28/jenkins-job-dsl-plugin.html.

About EMA