Editor: **Diomidis Spinellis**
Athens University of Economics and Business, dds@aueb.gr

Service Orchestration with Rundeck

Diomidis Spinellis

INCREASINGLY, SOFTWARE IS provided as a service. Managing and controlling the service's provision is tricky, but tools for service orchestration, such as Rundeck, can make our lives easier.

Take software deployment as an example. A well-run IT shop will have automated both the building of its software using tools like make, Ant, and Maven and the configuration of the hosts the software runs on with CFEngine, Chef, or Puppet (see "Don't Install Software by Hand," *IEEE Software*, vol. 29, no. 4, 2012, pp. 86–87). Furthermore, version control tools and continuous integration will manage the software source code and the configuration recipes, handling developer contributions, reviews, traceability, branches, logging, and sophisticated workflows. However, these tools still leave a gap between the software that has been built and is ready to deploy, and the server that has been configured with the appropriate components and libraries and is ready to run the software.

> Jobs are used to codify, record, and automate the processes associated with specific operations.

Extending build or configuration tools to handle deployment is problematic. One way to bridge the gap is to add rules to the build system. Many Makefiles, for instance, feature a "make install" rule that installs the built software. However, this approach is simplistic because it assumes that the build environment is the same as the deployment one. On production systems, this is rarely the case. Alternatively, the system configuration tool can be given the new software version as a package for deployment. This method is also unsatisfactory because it will install the software across the specified nodes in a "big bang" fashion with minimal control over the process.

Rundeck, a workflow management and service orchestration tool, helps us move from built software to a configured system by letting us define tasks to deploy the software or configure its operation. (Other tools in the same area are Capistrano, Fabric, Func, Knife, and MCollective.) When the software is ready for deployment, a suitably authorized administrator can log into the Rundeck Web interface and run the corresponding Rundeck job. The job will fetch the built software artifacts and gradually deploy them as specified, for example, first on the test infrastructure, then on a small subset of the nodes, and, finally, across all appropriate nodes. The job can be written to handle things like database migration, cache flushing, and restarting related services. The job invocation will also select the nodes where the job will be executed, log the operation and its progress, and verify that the deployment succeeded. And software deployment

Post your comments online by visiting the column's blog:
www.spinellis.gr/tools

0740-7459/14/$31.00 © 2014 IEEE

is just one service orchestration scenario that Rundeck can manage.

Configuration

Rundeck, open source software written in Java, is available as a package for many popular platforms. It's installed on a single server, interacting with other hosts via mechanisms such as SSH. Its architecture is plugin-based, which allows it to source its environment data from diverse cloud providers, such as AWS-EC2 and Rightscale, and to send notifications to systems like Jira, log4j, Jabber, PagerDuty, HipChat, and IRC. Plugins can also be used to define jobs, although it's often easier to use a system's shell commands for this purpose.

After installing Rundeck, administrators typically define the characteristics of the computing nodes (hosts) where the jobs will run, as well as the jobs themselves. Rundeck offers many ways to identify nodes, including the use of tags, which are supported by major cloud infrastructures. Thus nodes can be identified through their metadata, rather than their host names. This allows, for example, an administrator to specify that one job will only run on nodes designated as "memory cache servers" and that another should only run on nodes with more than four cores.

Defining a job involves specifying its options and its workflow. Options are associated with a name, a description, and allowed values. These can be supplied as a list, or they can be retrieved from a URL. Further restrictions on them can be specified in the form of a regular expression. When a job is run, Rundeck displays the job's available options and verifies that their given input matches the specified correct values.

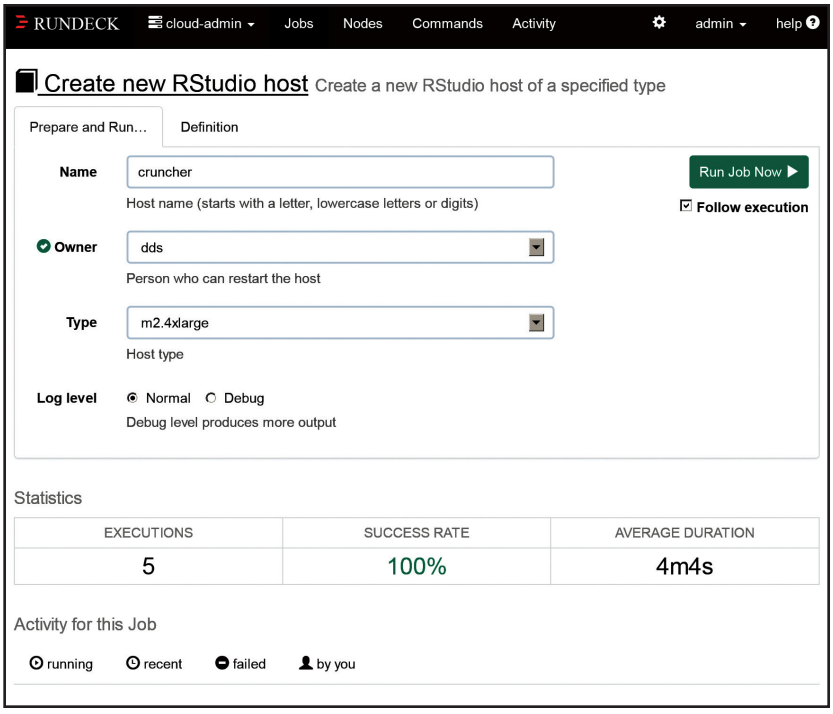

FIGURE 1. Specifying job options through Rundeck's GUI.

Administrators define workflows in terms of node steps, which run on each node, or steps that execute once for the entire workflow. A typical node step would be to restart a specified process, whereas a workflow step would be to send a notification message to the end-user support group. Each job step can be a command, a Rundeck script (specified inline or obtained from a URL), a reference to another job, or a plugin (which can be a shell script). Nested jobs can orchestrate complex services whose operation depends on processes running on multiple hosts. In addition, scheduled jobs can help bring programs that would be executed by the Unix cron daemon or the Windows Task Scheduler under the same single roof.

Jobs can be specified to run to completion on each node before being run on another node, or to have each step execute concurrently on all applicable nodes. This last method dramatically speeds up jobs that run on hundreds of nodes. In addition, administrators can specify what to do in case of a failure, whether a job should be run repeatedly, and whether multiple concurrent executions are allowed, as well as logging and notifications. Thus, jobs are used to codify, record, and automate the processes associated with specific operations (such as deploying a new version of a Web application).

Usage

When a job is run, administrators have to enter its options (see Figure 1). Administrators can also manually specify the nodes where it is to run. Rundeck then provides an estimate of the time required to finish the job, a monitor of the step executing at each node, a report of the job's out-

come per node, and a log of the job's output, which can be displayed by time or by node. At the end, a summary shows the number of nodes where the job run was complete, incomplete, failed, or not started. If a job failed on some nodes, there's even a button to retry it.

All job activity reports are stored in a database that can be browsed and filtered by job name, user, completion status, and time. If all operations on a system get run as Rundeck jobs, the activity report provides a precise indication of what was executed on each system's node, when, and by whom. This can be very useful for debugging, postmortem examinations, and compliance monitoring. Some even recommend executing all remote commands as Rundeck jobs (created through its built-in editor). This practice creates a central job repository and can aid knowledge sharing and automation.

Although Rundeck provides a command-line interface, an API, and a (rather strict) textual representation for specifying a job, its shiny graphical interface is what makes it appeal to a wide user base. This allows administrators to delegate certain actions to operators who might be uncomfortable with command-line tools. Nevertheless, the other interfaces are also useful, because monitoring tools can automate service recovery through the API, while the textual representation of jobs coupled with commands to dump and load their specifications makes it easy to put jobs under version control.

Jobs are organized together into projects (these can correspond to an organization's functional units) and into smaller subdivisions. Sophisticated access control lets administrators define groups and access control policies by project, group, or job. For example, developers could be allowed to deploy the software on the development and testing hosts, whereas site operators could be authorized to deploy the software on the production hosts. Logging and authorization provide transparency in the organization's operations and promote trust between the various teams.

None of Rundeck's features is difficult to implement from scratch via clever shell programming or a scripting language and its assorted libraries. In fact, many administrators (including this column's author) have done so, multiple times. However, these home-brew implementations will always lack the features, organization, centralization, and polish that a framework like Rundeck can offer. Not investing in learning and deploying such frameworks is a disservice to our clients and to our profession.

DIOMIDIS SPINELLIS is a professor in the Department of Management Science and Technology at the Athens University of Economics and Business and the author of the books *Code Reading* and *Code Quality: The Open Source Perspective* (Addison-Wesley, 2003 and 2006). Contact him at dds@aueb.gr.