Continuous Integration and Deployment (CI/CD)

WHITEPAPER | OCT 2015

APCERA.COM

Table of contents

Chapter 1: Introduction

Apcera Support Team | October 2015

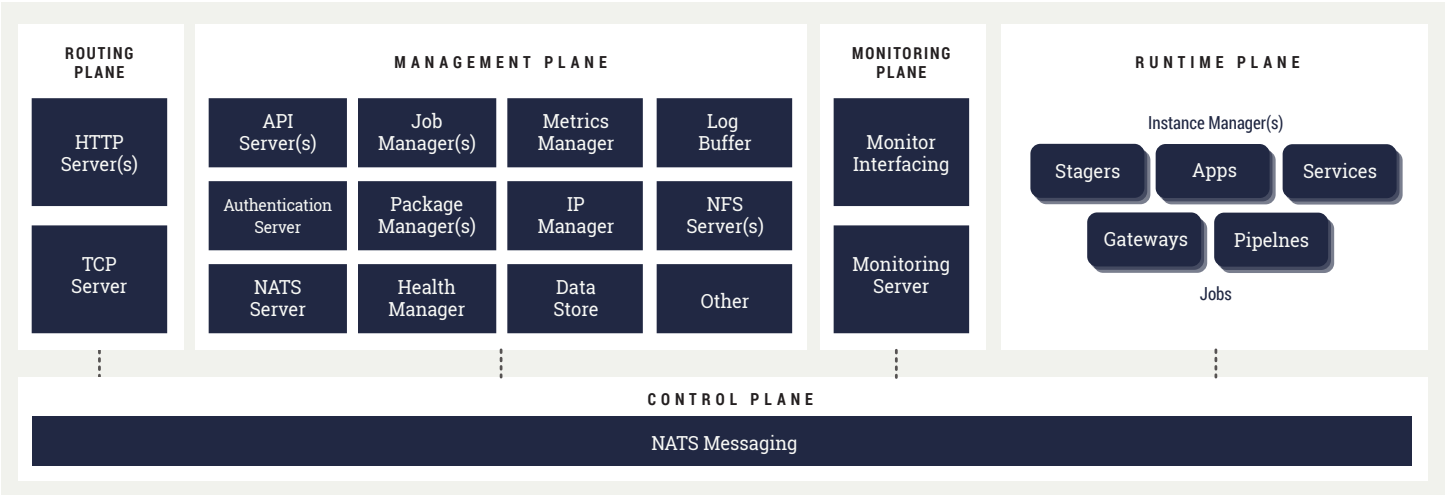
Organizations are always looking for solutions to decrease costs and increase productivity when developing and deploying software. Continuous integration and deployment systems have become the industry standard for testing applications and running regression tests. By spending less time trying to figure out how to deploy, test and scale an application, productivity will increase and operations costs will decrease. The intent of this paper is to explain how developers can use continuous integration systems such as Jenkins combined with the Apcera Platform to enhance the development and deployment lifecycle.

The Apcera Platform is the world's first policy-driven, trusted cloud application platform supporting deployment, orchestration and governance for diverse workloads across multiple clouds (hybrid, public and private). The platform enables users to deploy applications, containerized operating systems and Docker containers.

Deploying applications such as Java, Ruby, Go, and Node JS are made possible by intelligence in the platform with staging pipelines. The staging process determines what is needed to make a runnable package from the source code and will install the necessary modules and dependencies to properly deploy and run the code in a container.

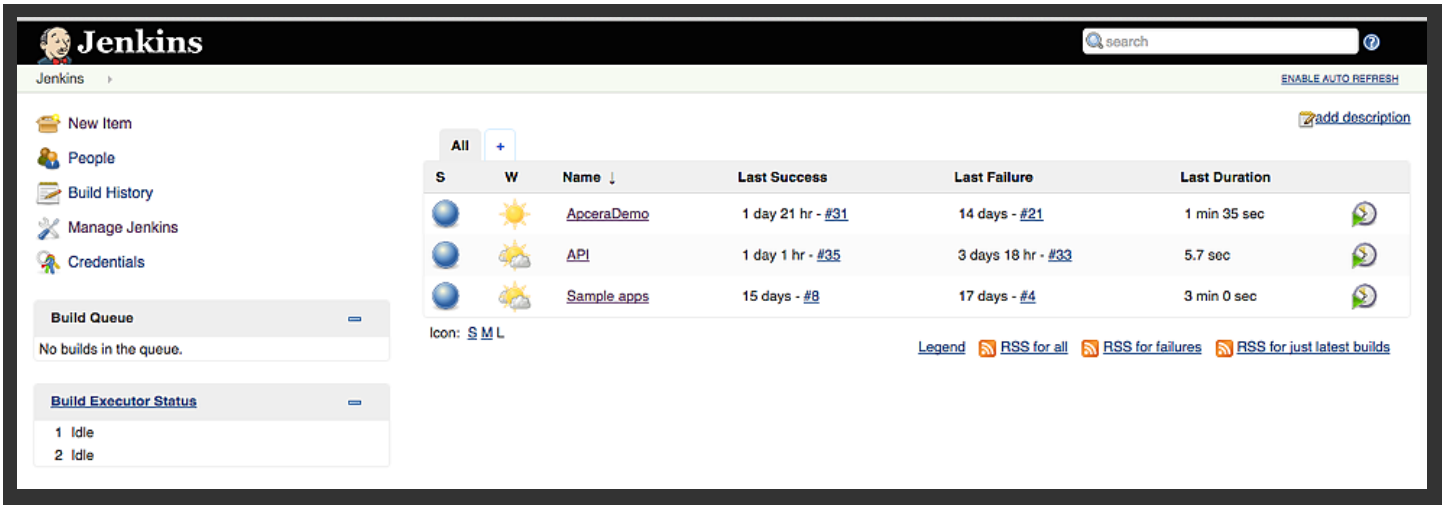
Jenkins is an open source and cross-platform continuous integration and delivery application. Since Jenkins is written in Java, any computer installed with Java can run the application with a simple command. Jenkins features an intuitive web and command line interface for interacting with the system along with hundreds of plugins such as GitHub or a MySQL connector to extend its capabilities. By integrating Jenkins with the Apcera Platform, developers can think of unique ways to test and deploy their code with a simple click of the mouse.

FIGURE 1 | Continuous Integration on the Apcera Platform



Chapter 2: Continuous Integration

FIGURE 2 | Jenkins Web Interface

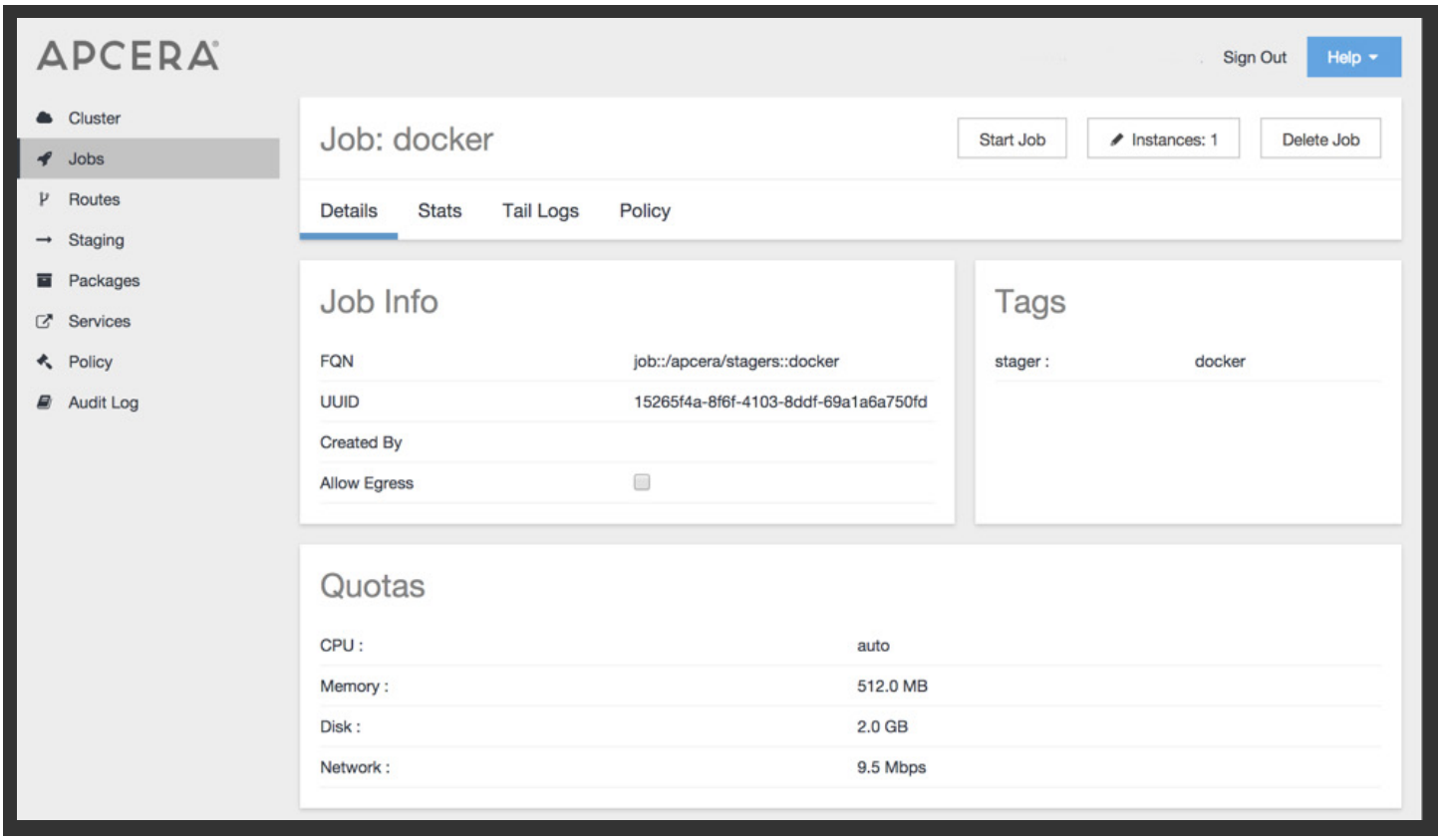


Continuous Integration is a common practice amongst engineering teams where code is merged into a code repository such as Git or SVN on a regular basis. After the code is checked into the repository, a build of the application will be made. Software teams must then figure out a way to run tests for each build and share the results. Results are often compiled with shell scripts and reports for each build are created manually. This has been a problem in the past because it is a time-consuming task to accomplish this on a regular basis for each build.

For many years, continuous integration systems such as Jenkins have been used to solve this problem by automating the integration process. Jenkins features a plugin for version control systems such as Github and Subversion (SVN). In Jenkins, developers can install the GitHub plugin and add their GIT repository information for the code branch that needs to be tested. Next, after the code is retrieved from the GIT repository, the developer needs to tell the build system what to do by using custom scripts or by adding shell commands in the project properties.

By executing scripts to control the build process, the possibilities are endless on what you can do. For example, after the source code is pulled from GIT, Jenkins can be instructed how to compile the code with a script. After the code is compiled, Jenkins can call additional scripts or shell commands to run tests on the build. All of the results will be available live in the console log window after a build is started. If errors are detected, the build will be marked as a failure.

FIGURE 3 | A Jenkins job running from the Apcera Web Console



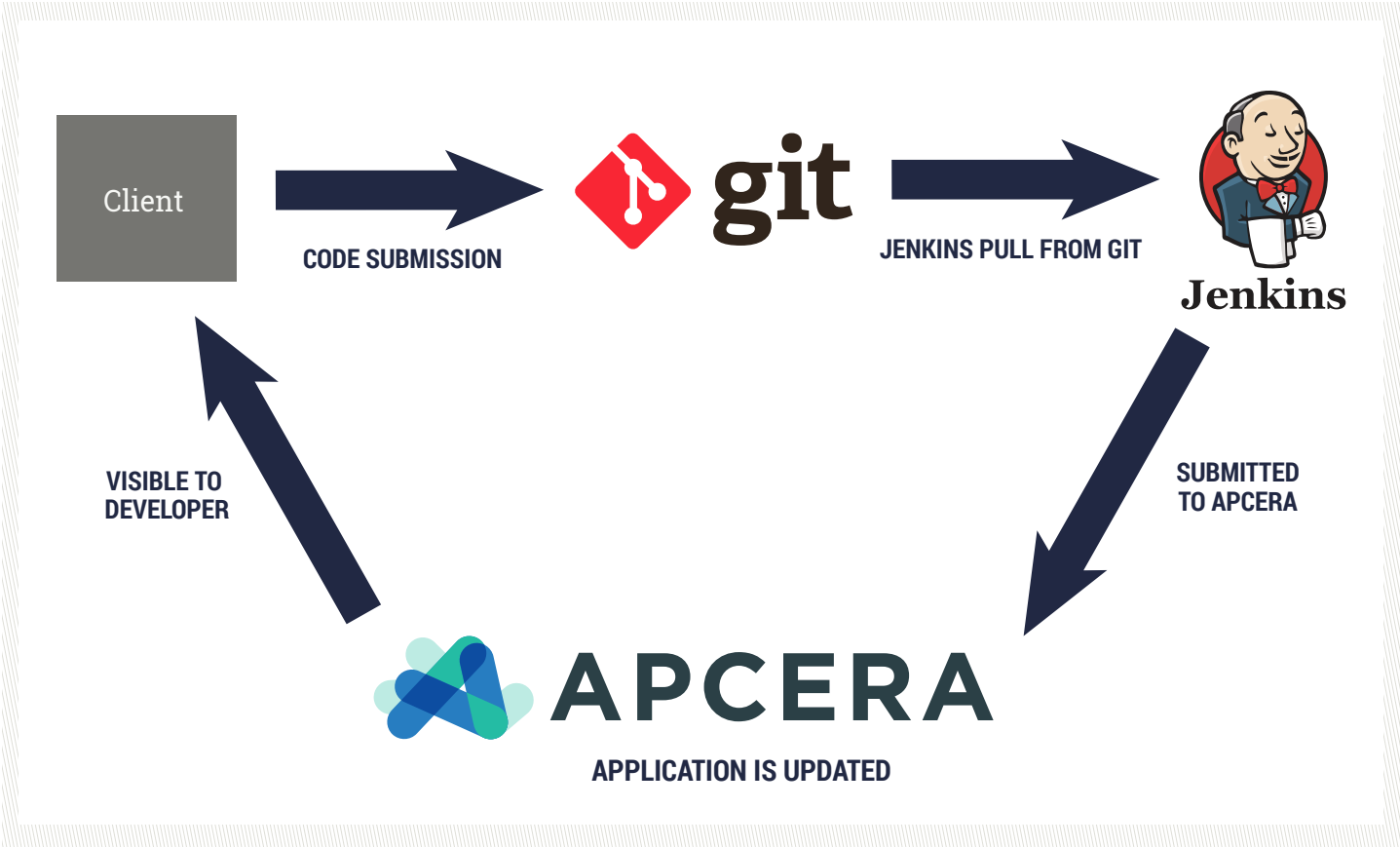
Continuous Integration (cont'd)

If user input is required for building or testing, Jenkins projects can be built with parameters which allow customized user input fields to be passed to scripts as strings. If the application being built contains a restful API or command line arguments, developers can simplify testing by parsing the build parameters to their application. For example, with an application that has a restful API, developers can run a batch set of curl commands for each supported API call, see the response in the console log and share it with their organization automatically.

By combining Jenkins with the Apcera Platform, developers can deploy their code automatically and then run functional tests. To take advantage of the Apcera Platform, you must install and run both Jenkins and the command line client APC in the same container. After Jenkins is running inside a container, it can make calls to the Apcera Platform with scripts. By integrating this with a version control repository, Jenkins can be configured to automatically create a build every time code is pushed to the repository, and execute scripts containing APC calls to deploy the application on the Apcera Platform without user intervention.

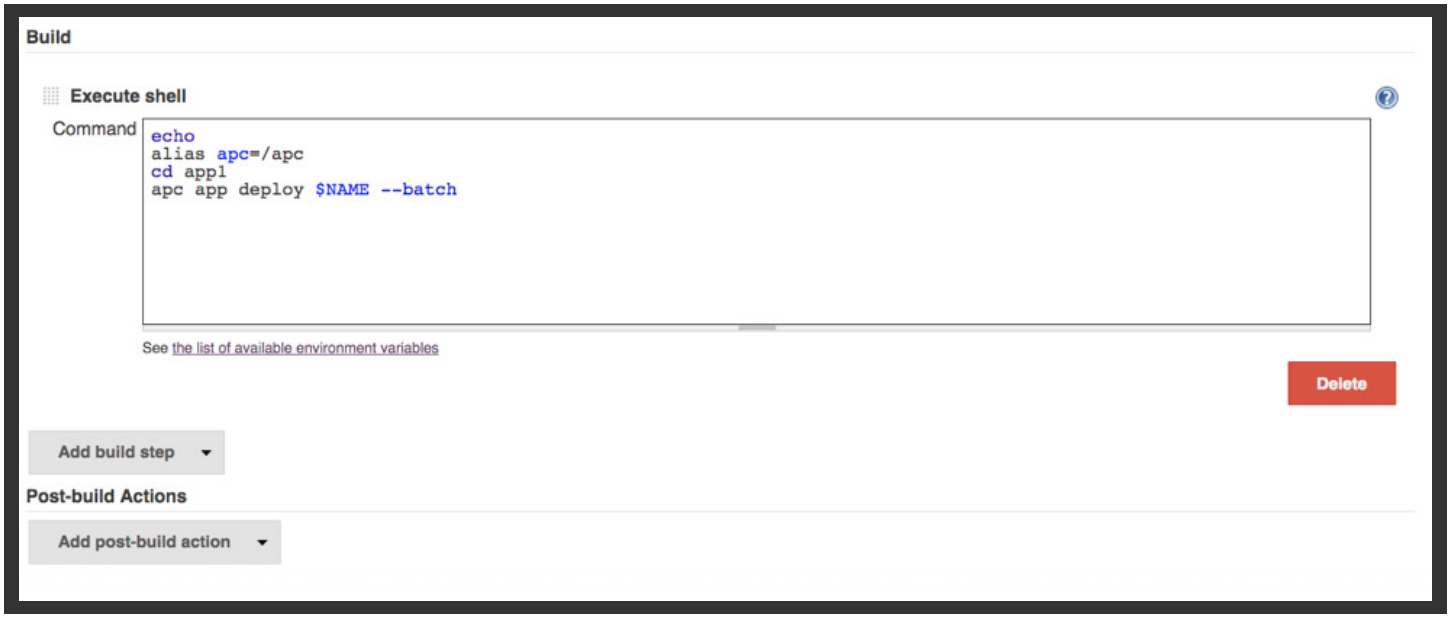
Chapter 3: Continuous Deployment

FIGURE 4 | Continuous Deployment



Writing software is considered by many an art form; however, a work of art is never complete in the eyes of an artist. Developers needs tools such as Jenkins to continuously deploy applications with new features and improvements to the code base. The Apcera Platform has this concept in mind with the ability to redeploy an application with updated code on the cluster. For continuous deployment with Apcera and Jenkins, the developer can submit code on his or her laptop and then push the changes to the Git repository. After Jenkins detects the code changes in GIT, the package build script in Jenkins can make APC calls to redeploy the application. After completion, the code will be available in production.

FIGURE 5 | Redeploying an application in Jenkins via script



Continuous Deployment (cont'd)

Having the ability to allow developers to continuously deploy applications requires protection mechanisms to ensure that cluster resources are not exhausted. The Apcera Platform provides policy that sets user restrictions on memory, CPU, network, and I/O. If a developer using Jenkins is deploying too many projects to the Apcera Platform or the build tests are exceeding set limits, the requests will be denied by policy.

The policy framework has been the core of the Apcera Platform since inception. Policy governs authentication to the cluster, policy modification, system resources, networking, and service and data usage. Policy is set cluster wide and is assigned to users and namespaces. This functionality provides users with pervasive controls to help secure the system and determine the network and service potential of the application.

A software development group uses a diverse set of languages, libraries and versions to develop software and the Apcera Platform has policy to help control this. By setting a policy for package resolution, administrators have the ability to create rules to restrict specific application and library versions such as Java 7 or 8 by default. This policy is useful to address security concerns and restrict application and library versions to developers in a heterogeneous environment.

The Apcera Platform allows administrators and developers to connect their applications to different services internally or externally, such as MySQL or NFS permitted by policy. For application testing and deployment, the Jenkins build script can be changed at any time with a new service provider. For example, if the developer needs to test their application for a different version of the MySQL, a new instance of the application can be deployed and updated with the new database service.

Continuous Deployment (cont'd)

Many organizations need a scalable solution for their build system. Jenkins has the ability to do distributed builds across multiple instances in a master and slave configuration. Large build environments can be made to support a large number of projects or tests. Since the Apcera Platform can run containerized operating systems and Docker images, developers have the ability to run single or multiple instances of Jenkins publically or privately from containers permitted by policy. If a large cluster of Jenkins instances are required, the administrator or developer can create as many instances as they need and add them to the cluster straight from Apcera.

Running Jenkins in a container on the Apcera Platform is the most efficient way to create and deploy applications. This approach creates a single platform for application development, deployment and testing. By completing all theses tasks locally, users will be more productive because the code and tests will be on-premise and will not travel outside the corporate network. With built-in application scaling on the Apcera Platform, the Jenkins build scripts that make APC calls have the ability to control the number of application instances needed to scale upon creation or after deployment.

About Apcera

Apcera delivers seamless workload mobility and pervasive policy and governance, across any cloud, public or private. Built on a foundation of trust, global 2000 companies use Apcera to securely develop, deploy, orchestrate and govern diverse workloads across multiple cloud providers, resulting in lower cost, simplified operations and mitigated risk. Apcera is headquartered in San Francisco.
Learn more at www.apcera.com.