Cognizant

Using Containers to More Effectively Manage DevOps Continuous Integration

Extending the concept of containers to continuous integration (CI) provides IT organizations with faster and more scalable ways to reduce the cost and maintenance of new digital infrastructure.

Executive Summary

As digital waves wash over the enterprise, IT organizations must embrace Agile and DevOps methodologies as competitive necessities rather than nice-to-have luxuries. Agile engineering practices are now mainstream, opening IT to the virtues of continuous integration (CI) and continuous delivery as ways to better deliver on the promise of distributed technology stacks. Jenkins, one of the most widely used CI servers, requires the establishment of CI farms, deployed and managed in a master/multi-slave architecture. CI aims to accelerate software deployment. However, the substantial time and effort that it takes to set up, maintain and onboard new projects into a Jenkins Build Farm and the inability of the CI farms to process the build loads can delay developer feedback, thus negating CI's important benefits.

Emerging technologies such as containers (i.e., CI capabilities in a box) offer a viable solution to Jenkins' CI challenges. The ability to run Jenkins within containers enables optimizing the build infrastructure and accelerating the CI process.

This white paper analyzes the use of containers to provide self-contained and autonomous CI infrastructure that scales on demand through the use of the container platform and container management software like Docker and Kubernetes.

Since containers are lightweight, they can be spawned on demand, depending on the build load for each developer.

CONTINUOUS INTEGRATION ECOSYSTEM CHALLENGES

IT organizations seeking to set up a typical Jenkins-based CI ecosystem face the following challenges:

- **Low infrastructure utilization:** The build infrastructure is typically suboptimal. For example, during release times, when there are huge spikes of development activities, multiple developers attempt to commit codes to an ill-equipped infrastructure, resulting in substandard performance. During the rest of the year, there may a lower load on the build systems. Despite proper forecasting that is based on projected release volumes, these idle times are unavoidable. This results in low utilization of the build infrastructure and poor ROI.
- **Limited scalability of build infrastructure:** Even with a CI system in place, the ecosystem is unable to scale to meet peak build loads; this is frustrating for developers who find their builds placed on queues. It often defeats the essence of DevOps where developer feedback needs to be instantaneous and continuous. This limited scalability necessitates advance forecasting and configuration of additional slaves while onboarding new projects.
- **Time-consuming setup and configuration:** Setting up a Jenkins-based CI ecosystem involves multiple activities such as provisioning the server-based specifications, configuring a secure connection between the Jenkins master and slave, and installing the necessary software and plug-ins. This process includes: providing access to various users; analyzing the suitability of templates for jobs and recreating them if necessary; testing the templates and plug-ins and establishing connectivity to the deployment environments from created nodes. It can take up to three to four weeks to set up a Jenkins Slave ecosystem and onboard a new application, which includes the time required for approvals and resolutions from enterprise IT and network teams.
- **Increased vulnerability due to un-versioned pipelines:** Today, developers define build steps as configurable items inside the CI systems. They must navigate through the Jenkins user interface to define these build steps. The build steps become fragile due to the inability of developers to control the versions of pipeline changes, resulting in configuration drift and errors.

USING CONTAINERS FOR FASTER AND EFFECTIVE CI

The ability to run applications inside containers can be extended to continuous integration, where the CI server is run within the Docker container. This provides significant opportunities for optimizing the build infrastructure. Since containers are lightweight, they can be spawned on demand, depending on the build load for each developer. A container orchestration software like Kubernetes or Apache Mesos can be used to manage the entire lifecycle of the containers within the CI server. We have developed a solution that takes this approach to provide developers with a CI infrastructure that scales on demand.

Solution Overview

The solution encompasses a central framework that listens to the developer commits that occur with the source code repository (SCR) that is monitored. The trigger is exerted when a commit happens to the repo and it invokes the framework endpoint configured as a web hook. The SCR passes the commit metadata through the payload to the framework. The framework processes the metadata into critical parameters for CI such as developer ID, commit ID, repo name and branch, along with other information. The framework also parses the pipeline file and maps the Docker images that must be pulled for this specific request.

A Docker image specific to the application technology that can be used for the build needs to be mapped. For instance, in order to build a J2EE application, a Docker image – along with the necessary build time dependencies of JDK and Maven – needs to be composed and built as a composite image. The framework pulls the image mentioned in the pipeline code from the Docker registry and applies it to the application build.

The framework then initiates Kubernetes to produce Docker containers based on a specific image with Jenkins, and prepares it for the CI. The application context is loaded into the Jenkins running inside the Docker, and the jobs are created along with the necessary pipeline components. Pipeline jobs are executed inside the container, and the respective entities are passed between them within the pipeline.

For instance, a typical pipeline may involve code checkout, application build, unit tests triggered and static code analysis, followed by archiving binaries to a repository. The pipeline can also be extended for test automation and environment provisioning. The only prerequisite is that the containers should be able to access the respective DevOps tools and environments.

Figure 1 offers a representation of how our solution operates.

A Container-Based CI Solution

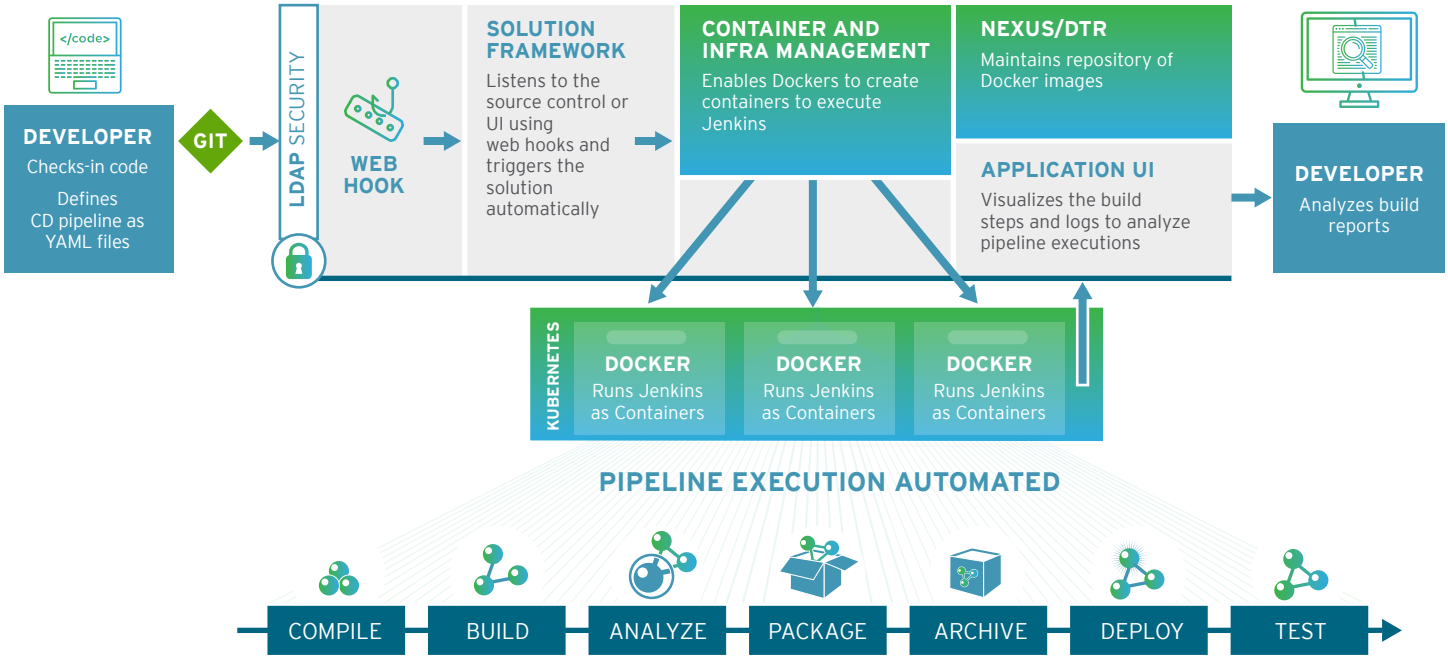


Figure 1

Based on the volume of the build requests (or commits), a new instance of Docker is provisioned for each developer by granting an exclusive CI system. This becomes a powerful proposition making the build farms more scalable.

Kubernetes confers the dual benefits of container orchestration as well as infrastructure abstraction. Based on the volume of the build requests (or commits), a new instance of Docker is provisioned for each developer by granting an exclusive CI system. This becomes a powerful proposition making the build farms more scalable. The logs from Jenkins and associated plug-ins can be redirected to a log management stack like Elastic search, Logstash and Kibana (ELK) for more complex log analytics such as build analysis and real-time progress of the pipeline.

Solution Benefits

Initial deployments of our solution have resulted in the following benefits:

- **Optimized build infrastructure utilization:** The abilities to spawn Jenkins containers on demand and tear them down upon execution ensure that the compute resources are optimized, thus leading to a better utilization of the build infrastructure.
- **Scalable build farms:** The container-based autonomous CI ecosystem provides the capacity to scale up or scale down based on the load (i.e., code commits). This elastic nature of the CI ecosystem will ensure instant feedback once a commit is made to the repository.
- **Reduction of time to onboard new applications into CI ecosystem:** New applications can be brought into the CI ecosystem more quickly. The time needed for onboarding new projects is shortened by the "templatized" Docker build images, the pluggable Kubernetes slave that can accommodate new loads and the reusable pipeline as code modules that can be shared across projects.
- **Increases in developer productivity:** Developers do not need to wait in queues to receive feedback on builds. Pipeline as code provides significant convenience and productivity to developers enabling them to define CI configurations alongside code in the same repository.

Quick Take

Large U.S.-Based Healthcare Outfit Containerizes CI

We partnered with a leading U.S. healthcare solution provider to address challenges that were present in its build and CI processes. The company's technology landscape had 456 applications built in Java technology that leveraged dedicated Jenkins-based nodes for CI. The enterprise wanted to achieve scalability of build infrastructure using containers and optimize its CI build investments. Based on the customer's need and leveraging their existing Jenkins Master infrastructure, our solution was tailored to provide elastic build infrastructure that focused on spawning Jenkins slaves on demand using Kubernetes and Docker.

The solution's envisioned benefits include:

- **A 55% reduction in time to set up CI ecosystem.**
- **More than 12x increase in infrastructure utilization.**
- **Over 80% reduction in compute infrastructure.**
- **A 40% reduction in infrastructure costs.**

LOOKING FORWARD

Our solution can be enhanced further by connecting it with enterprise bot channels. While the most natural way to trigger the solution is by means of source code repository web hooks, another way this can occur is via a simple web application or a bot configured to an enterprise chat channel. A bot-based trigger could blend the solution into the value proposition of conversational DevOps, where the developers can trigger CI build manually through conversations with a bot inside an enterprise chat channel.

ABOUT THE AUTHORS

Karthikeyan Vedagiri

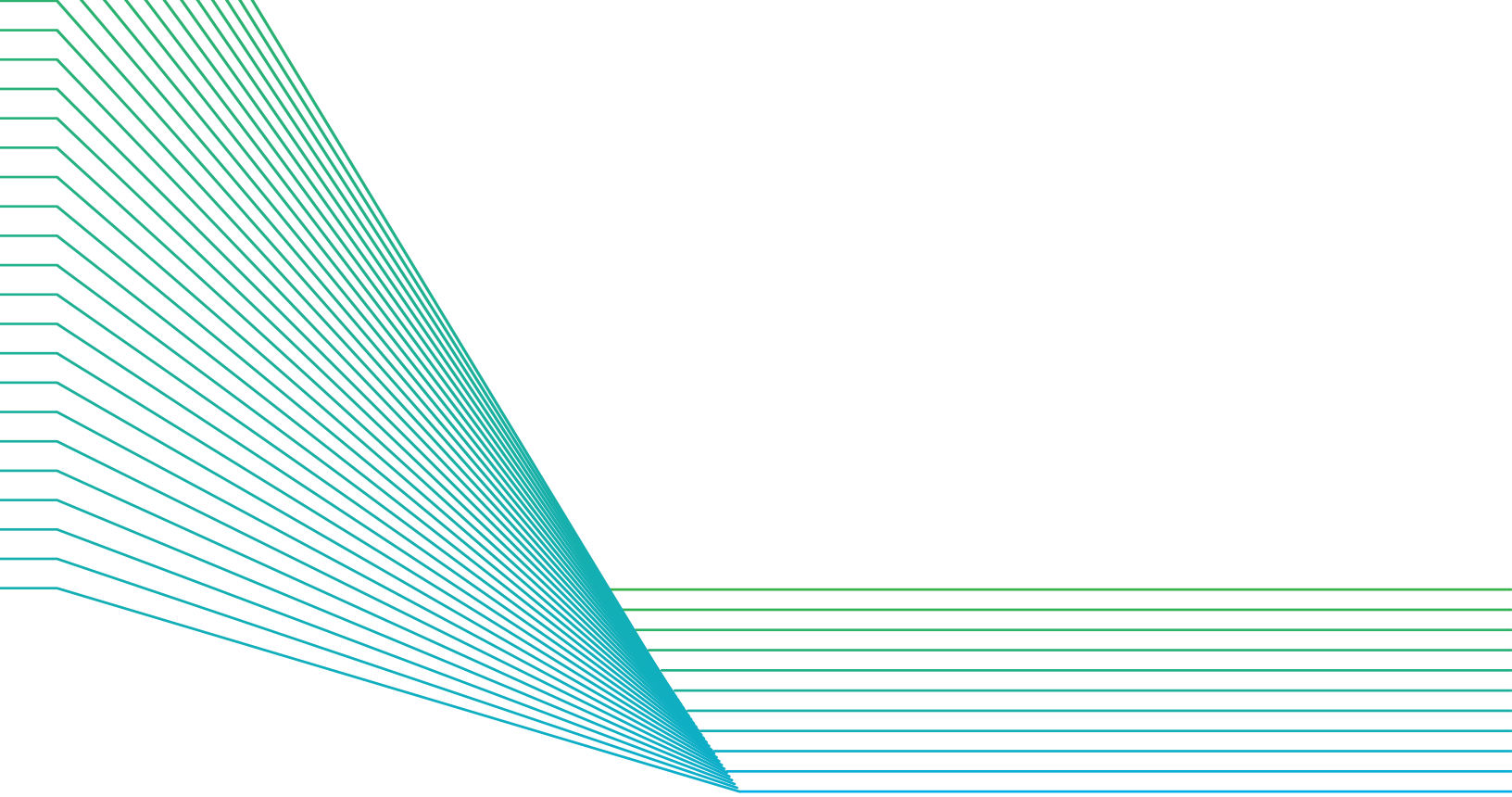
Senior Manager, Cognizant's DevOps Practice

Karthikeyan Vedagiri is Senior Manager, Projects within Cognizant's DevOps Practice where he focuses on the development of products and accelerators for Cognizant OneDevOps™. He has more than 13 years of experience in product engineering and quality assurance, and he specializes in design and architecture of test automation frameworks, deployment pipelines and DevOps platforms for emerging technologies such as cloud, container and platform as a service (PaaS). Karthikeyan can be reached at Karthikeyan.Vedagiri@cognizant.com.

Rangarajan Rajamani

Senior Manager, Cognizant's DevOps Practice

Rangarajan Rajamani is Senior Manager, Business Development within Cognizant's DevOps Practice. He has over 14 years of expertise in multiple industries such as global life sciences, manufacturing and construction, performing domain consulting, process consulting, presales and product evangelization. Rangarajan focuses on marketing and evangelizing digital technologies in areas such as IoT, cloud, service virtualization, functional QA automation and DevOps. He can be reached at Rangarajan.Rajamani@cognizant.com.



ABOUT COGNIZANT

Cognizant (NASDAQ-100: CTSH) is one of the world's leading professional services companies, transforming clients' business, operating and technology models for the digital era. Our unique industry-based, consultative approach helps clients envision, build and run more innovative and efficient businesses. Headquartered in the U.S., Cognizant is ranked 205 on the Fortune 500 and is consistently listed among the most admired companies in the world. Learn how Cognizant helps clients lead with digital at www.cognizant.com or follow us @Cognizant.

Cognizant

World Headquarters
500 Frank W. Burr Blvd.
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Fax: +1 201 801 0243
Toll Free: +1 888 937 3277

European Headquarters
1 Kingdom Street
Paddington Central
London W2 6BD England
Phone: +44 (0) 20 7297 7600
Fax: +44 (0) 20 7121 0102

India Operations Headquarters
#5/535 Old Mahabalipuram Road
Okkiyam Pettai, Thoraipakkam
Chennai, 600 096 India
Phone: +91 (0) 44 4209 6000
Fax: +91 (0) 44 4209 6060

TL Codex 3033