Continuous Integration (CI) Pipeline with Git, Jenkins, JFrog Artifactory, and ONTAP (ONTAP 9, ONTAP Select, ONTAP Cloud)
CI Workflow Enabled by NetApp Technologies

Bikash Roy Choudhury, NetApp
Akshay Patil, NetApp

August 2017 | TR-4547

Abstract

A massive evolution is under way that is transforming traditional forms of application development to more agile processes that drive faster time to market. Organizations are adopting continuous integration (CI) and continuous delivery (CD) workflows for these more agile processes. These new DevOps workflows offer more value and enable greater innovation around the applications developed. CI workflows allow applications to be tested in a high-velocity iterative manner. In addition to speeding the pace and volume of development, there are substantial improvements in quality because bugs are identified earlier in the code development lifecycle.

Although there is a lot of focus on changing workflows and automating infrastructure and development tools, data is created, stored, and processed during the entire application lifecycle. Enterprise-class NetApp® storage provides several data services that use native technologies that integrate with CI tools such as Git, Jenkins, and JFrog Artifactory. These technologies use Representational State Transfer (RESTful) APIs to provide automation for developers to write, test, build, stage, and deploy their applications without any knowledge of the underlying storage, yet maintaining full control over their data.
TABLE OF CONTENTS

1 Introduction ........................................................................................................................................... 3

2 Continuous Integration with Git, Jenkins, Artifactory, and ONTAP ................................................. 3
   2.1 Data Management .......................................................................................................................... 4
   2.2 RESTful APIs ................................................................................................................................. 5
   2.3 Binary Artifact Management .......................................................................................................... 5
   2.4 Software Control Management (SCM) .......................................................................................... 5
   2.5 Continuous Integration .................................................................................................................. 6
   2.6 Container Orchestration ................................................................................................................ 6

3 Zero Storage Touch with NetApp Jenkins Framework ........................................................................... 6
   3.1 NetApp Jenkins Framework Modules ............................................................................................. 7
   3.2 NetApp Jenkins Workflow ............................................................................................................... 7
   3.3 NetApp Jenkins Framework: Scalable Architecture ....................................................................... 12

4 Conclusion ............................................................................................................................................. 13

Version History ........................................................................................................................................ 14

LIST OF FIGURES
Figure 1) CI/CD workflow using Jenkins pipeline. .................................................................................. 4
Figure 2) CI workflow enabled with NetApp technologies. ................................................................. 4
Figure 3) NetApp Jenkins framework. .................................................................................................... 7
Figure 4) NetApp Jenkins framework pipelines. ..................................................................................... 8
Figure 5) NetApp Jenkins workflow. ....................................................................................................... 8
Figure 6) CI environment with NetApp Jenkins framework. ................................................................ 9
Figure 7) Builds in Docker Registry managed by JFrog Artifactory. ...................................................... 11
Figure 8) NetApp Jenkins architecture. ................................................................................................. 12
1 Introduction

In the ongoing digital transformation, the software business is driving the economy. Every form of consumer-driven business that has a digital footprint is developed and consumed through applications. Business owners are constantly thinking about innovation and the value they can provide through their products, services, and features. This innovation drives faster time to market and breeds healthy competition for organizations in the software business.

Organizations are rapidly moving away from traditional forms of software development to agile workflows that can drive faster time to market. Continuous integration (CI) and continuous delivery (CD) are two of the most common forms of agile workflows in the application development cycle.

Most businesses face multiple challenges during the transformation in the DevOps process. The challenges focus on three main areas:

- Standardization and consolidation of development and operational tools
- Cultural impediments
- Transitioning into continuous workflow for software development and delivery

During the application development cycle, data is created, stored, processed, shared, managed, and protected. The volume of the data may not be high for the actual code repository, but a lot of data is generated from multiple source code versions, builds from continuous integrations, binaries and packages, user acceptance testing (UAT) or staging, and post deployment of the application in production. High availability, scalability, and predictable performance of data are strong requirements during the different phases of the DevOps process.

Data in any form (active, in motion, or at rest) is extremely important to application developers and business owners. However, managing and maintaining the infrastructure that stores the data are low priorities for most developers. Integrating infrastructure as code into workflows and development tools enables developers to have greater control over the data without requiring them to know a lot about operations.

NetApp, with ONTAP® as the data management software, has a significant storage footprint for application code development, build, and test environments. ONTAP provides scalability, data protection, data portability, and persistence for data access on premises, in public cloud, or in hybrid cloud. It also provides storage efficiencies to enable more workloads with lower storage costs.

This technical report addresses some of the critical challenges that application owners and developers often face, such as reducing build times, improving developer productivity, and reducing overall infrastructure costs to provide better return on investment (ROI) by using ONTAP technologies. Integrating development tools like Git, Jenkins, JFrog Artifactory, and ONTAP is a step in moving to a more consistent data management platform for the CI pipeline using RESTful APIs.

2 Continuous Integration with Git, Jenkins, Artifactory, and ONTAP

Continuous integration (CI) is an automated and iterative test-and-build process that enables teams to find bugs in code in the early stages of development. Continuous delivery (CD), in contrast, is a completely automated process, from successful builds through application deployment and release into production. CI and CD workflows enable better code quality by testing code iteratively and automating the software delivery pipeline.

As shown in Figure 1, the workflow starts with the development cycle that includes the CI process until the application is released into production in a continuous manner. The flowchart in Figure 1 shows how the developer checks out the code from the master code base by using the Software Control Management (SCM) tool, which manages the different code revisions. The developer modifies the code to add new features and functionality, test, build, and merge the changes into the main codebase. The final successful build is then promoted to be deployed for user acceptance testing or staging and is finally released into production.
This report focuses primarily on some of the common development tools used during the continuous integration process, as shown in Figure 2. The development cycle consists of use cases and design documents, source code management, build and test, and binary artifact management. For the purpose of this document and integration with ONTAP, development tools like GitLab Community Edition, Jenkins, and JFrog Artifactory are ingrated with ONTAP by using RESTful APIs on Docker containers. Business organizations may use different development tools to accomplish the CI/CD process in their environment.

2.1 Data Management

NetApp ONTAP data management software is used to store, manage, and protect data. Thin provisioned NetApp FlexVol® volumes, Snapshot™ copies, and FlexClone® volumes provide the flexibility to provision, grow, and shrink storage on demand and allow more workloads to scale during the CI/CD process.

Apart from significant improvements and efficiencies identified in the CI pipeline for developers, every organization wants a formally managed set of processes for data governance. Those processes include data security, compliance, and protection.
As shown in Figure 2, NetApp meets all the data governance requirements mentioned earlier that are embedded and integrated natively into the storage layer, along with service automation analytics and secure multitenant technologies:

- **Data compliance.** NetApp offers different forms of compliance, such as Federal Information Processing Standards (FIPS) 140-2, for both active data and data at rest.
- **Data security.** NetApp provides data security with full encryption and also provides security at the protocol layers (Network File System [NFS] and server message block [SMB]). No matter how it is accessed from the development workflows, data is secure while in motion, in active state, and at rest.
- **Service automation analytics.** NetApp offers several levels of basic and compound APIs that can report, monitor, and provision data. Service-level objective (SLO) based APIs for storage can also provide performance headroom on different storage controllers by using APIs, which allows handling of data growth, scalability, and load balancing. These APIs can be provisioned or consumed as “infrastructure as code” or as “configuration as code” in agile development environments.
- **Secure multitenancy.** NetApp storage can provide secure tenants that can run in the same cluster. Sales, marketing, and finance can each have their own tenants and coexist in the same cluster. This keeps each tenant secure and also allows better data management capabilities.
- **Storage efficiencies.** Thin-provisioned volumes and Snapshot copies use space very efficiently. Snapshot copies are crucial to enabling consistent checkpoints or recovery points for data. FlexClone volumes are used to create near-instantaneous workspaces for development workflows that take up very little capacity relative to the dataset being cloned. Inline deduplication allows build QA test copies to use storage space efficiently because most of the build files are full copies and not delta copies. Many build files are small in size (<4k). The compaction feature in All Flash FAS further optimizes the storage space. All of these space savings add up to reduced storage cost and therefore improved ROI.
- **Integrated data protection.** NetApp SnapMirror® replication technology makes it possible to easily replicate data between different environments, including cloud instances, without requiring lock-in to any one provider. This technology also enables data to move into different availability zones in the cloud, even spanning different geographic locations for disaster recovery. NetApp SnapVault® backup software can also be used to archive files for data at rest. Data can also be moved in object stores with NetApp StorageGRID® for less costly and denser archiving.

### 2.2 RESTful APIs

NetApp Service Level Manager (NSLM) 1.0 is a service level objective based ONTAP RESTful API integration with Jenkins in this plugin. The RESTful APIs are used to create volumes, Snapshot copies, and FlexClone volumes in ONTAP. These ONTAP APIs automatically enable load balancing and scalability of the FlexVol and FlexClone volumes in a cluster namespace based on controller headroom.

### 2.3 Binary Artifact Management

JFrog Artifactory is a universal artifact repository manager that fully supports software packages created with any language or technology. It provides a central hub for distributed repository management for all artifacts required for binaries like Python, Java, and C++. It works as an intermediate layer between the different development teams across an organization and external repositories. Artifactory supports multiple build packages like Maven, Docker, Gradle, and NPM.

### 2.4 Software Control Management (SCM)

SCM is different from the binary artifact manager described in section 2.3. Git is a popular open source and distributed source code management tool. It is a version control system that tracks the changes made by different developers to the source code files or blobs. Different flavors of Git include Bitbucket from Atlassian, GitLab CE or Enterprise Edition (EE), and Helix4Git from Perforce.
2.5 Continuous Integration

Jenkins is a commonly used open source continuous integration (CI) tool that enables developers to build and test code to identify bugs quickly in an automated manner. Jenkins follows a distributed architecture with a master and slave configuration. The Jenkins master is responsible for scheduling, managing, dispatching, and monitoring build jobs. Each of these jobs represents a slave. The slaves run different jobs as requested by the master as pipelines in a distributed manner.

2.6 Container Orchestration

Docker containers are used to run the Jenkins master and the slaves in the CI workflow. Containers provide modularity and portability of source code and binaries during application development. Resiliency for the Jenkins master is an important requirement during the software build process. The resiliency for the Jenkins master is set up in two ways, which complement each other:

- The Jenkins home directory is configured on shared storage like ONTAP and mounted over Network Filesystem (NFSv3) using the NetApp Docker Volume plugin (netappdvp).
- Jenkins master and slaves run as a service in a Docker Swarm multilhost cluster. If the Jenkins master fails, a new Docker service for the Jenkins is immediately spun up on a different node pointing to the home directory on the NFS share.

NetApp offers a Docker Volume plugin called netappdvp that mounts persistent data storage over NFS and iSCSI.

3 Zero Storage Touch with NetApp Jenkins Framework

Jenkins is primarily used to handle the “Dev” part of the common DevOps workflow, while most of the “Ops” part is done by DevOps administrators and infrastructure engineers. The NetApp Jenkins framework focuses on bringing both the Dev and Ops parts closer together by seamlessly integrating and automating the tasks performed at the storage layer. The NetApp Jenkins framework gives developers a zero storage touch experience, making them more productive in writing, testing, and building code.

The CI pipeline integration with Jenkins and ONTAP 9 using NetApp Service Level Manager APIs offers developers and business owners the following benefits:

- Reduces developer build time by more than 50%, leading to faster time to market:
  - Mostly incremental builds with limited full builds required
  - Build artifacts not in the same location as the source code repository
- Improves developer productivity and efficiency by more than 60% to achieve development at scale:
  - Instantaneous prepackaged user workspaces
  - Mitigates risk for code changes and reduced merge conflicts
  - Quick recovery from test (unit, smoke) failures
- Reduces infrastructure cost (compute, network, and storage) in development and deployment environments by up to 40% though thin provisioning and storage efficiencies like compaction and inline deduplication and compression.
  About 50% space savings can be achieved by using inline compression and deduplication because the build environment uses the same binaries and dependencies over and over again.

The NetApp Jenkins framework offers some exciting integrations for developers who are using tools like GitLab CE, Jenkins, and JFrog Artifactory to establish a reliable and consistent CI pipeline. Integrating Docker containers with the storage-persistent Docker Volume plug-in and Docker Swarm provides scalability, agility, and resiliency in the CI environment. Native ONTAP technologies such as Snapshot copies and FlexClone volumes integrate with Jenkins by using NetApp Service Level Manager, giving much-needed transparency to developers, who may not know much about storage.
Note: The NetApp Jenkins framework does not have any direct dependency on a specific Linux version or physical hosts or virtual machines. The aim of this integration is to provide developers with all the storage-related features they need without having to know NetApp technologies. This framework can run on ONTAP 9 (FAS), ONTAP Select on virtual machines, and ONTAP Cloud in Amazon Web Services and Azure.

3.1 NetApp Jenkins Framework Modules

The scope of the NetApp Jenkins integration is to provide a framework from the time the local source code repository is set up with GitLab all the way up to having a successful Docker image and build file zipped and pushed into JFrog Artifactory. As shown in Figure 3, all the modules of the framework—GitLab, Jenkins, and developer workspaces—run on Docker containers that mount data-persistent volumes from ONTAP by using netappdvp. Docker containers provide a modular form of architecture for developing and deploying cloud-native applications.

![NetApp Jenkins framework](image)

Figure 3) NetApp Jenkins framework.

In this framework, Jenkins master runs as a Docker service spanning a swarm cluster, while the rest of the CI jobs run on a lightweight Jenkins slave. GitLab, all the provisioned Jenkins jobs (CI or integrated builds, developer or private builds) and JFrog Artifactory run as Jenkins slaves. Jenkins uses various plugins to communicate with GitLab and JFrog Artifactory. These Jenkins slaves run as Docker services. This framework also provides the ability to automatically fail over and recover from Jenkins master failures, as described in section 2.6. Section 3.3 explains how slaves communicate with Jenkins master in the NetApp Jenkins architecture.

3.2 NetApp Jenkins Workflow

The workflow for the NetApp Jenkins framework is targeted to two different roles—DevOps administrators and developers. DevOps admins are responsible for the following pipeline operations to set up the CI environment, as shown in Figure 4:

- Source code management
- Continous integration
- Build artifact management

Because all development sites and projects are different, in setting up these pipelines the DevOps admin follows a set of one-time ONTAP configuration steps in the Jenkins environment.

Developers normally use the developer workspace pipeline to create instantaneous workspaces that are prepackaged with the source code, prebuild artifacts, and binaries. Developers can access all of the pipelines, depending on the policy and the permissions set up by the DevOps admin.
The NetApp Jenkins framework consists primarily of five workflow components (see Figure 5):

- **Version control.** Source code management pipeline, SCM snapshots
- **CI environment.** Continuous integration pipeline, build snapshots
- **Developer environment.** Developer workspace pipeline
- **Code management.** Purge policy, list build snapshots, list SCM snapshots
- **Building artifacts.** Build artifact management pipeline

Jenkins runs all tasks as *jobs* through pipelines and controls and monitors all jobs. Each of these phases offers a direct advantage to the CI and the developer workflow, as discussed later in this section. ONTAP provides a shared infrastructure in which the volumes are exported and mounted on the Docker containers running on physical nodes or on VMs over NFSv3. All of the NetApp technologies and integrations are transparent to the developers and DevOps admins.

**Source Code Repository**

This Jenkins job allows you to create a local source code repository in ONTAP. It creates a volume or a partition and mounts it on a Docker container using netappdvp. A migration or *git clone* of the source code from the GitHub location populates the local repository volume on NetApp storage.

**Note:** By default, the framework uses GitLab CE as the SCM tool. Any SCM tool that has a Docker image can be replaced and used in this framework.
There are several reasons for creating a local repository in the NetApp storage:

- The `git clone` from a public or private repository for different CI jobs can use up compute resources and network bandwidth when the main codebase is cloned multiple times to different user workspaces. A local code repository reduces network traffic every time code is pulled from the GitHub location. Operations to access the source code are offloaded to ONTAP when creating multiple CI environments (see section 3.2.2).

- All changes in the source code repository can be managed locally before pushing the final updates to GitHub, affording better control and ownership of code.

- Automatic checkpoints or Snapshot copies on the SCM volume are created on every successful source code check in as shown in Figure 6. This helps in a couple of ways:
  - Periodic consistent backups of the code repository and its changes offer better data protection
  - Collaboration of source code with remote sites where local users can access the source code at that local site at scale
  - Quick data recovery from any failure or data corruption

Figure 6) CI environment with NetApp Jenkins framework.

![CI environment diagram](image)

**Continuous Integration Environment**

NetApp recommends creating different development or CI code branches from the main codebase after the developer begins to create new code and to work on new features, as shown in Figure 6. These development or CI code branches can be organized on different ONTAP volumes. If the application that is developed does not have a large codebase, then a single development branch may suffice. This allows more control to introduce new features, identify and fix bugs quickly, and run tests independently and iteratively on different development branches in parallel.

ONTAP provides a stable CI environment by creating volumes for every development branch and mounts them automatically on Docker services, which are checked out from the main codebase. These volumes are synced with the right version of the tools, compilers, RPM, and libraries required for the application developed using a specific programming language (Java, .NET, C, PHP, Python, etc.). This is followed by a full build process. All subsequent builds can be made incrementally. This approach significantly reduces build times.
Upon successful completion of the full or incremental build, a Snapshot copy or a checkpoint is automatically created on the development branch volume. This volume becomes the baseline volume for that particular development branch. This volume is completely prepackaged with the source code, prebuild artifacts, and binaries.

Having detached baseline volumes for every developer or CI code branch has the following benefits:

- The build artifacts are separate from the source code, reducing network traffic on the SCM server.
- The developer builds that are run on the baseline volume for every development branch are incremental in nature. Because this baseline volume has precompiled artifacts, the builds don't take much time. For example, if the branch has 1,000 lines of code and only 4 lines are changed, then the build happens for those changes only. This reduces the build time significantly.
- Parallel builds can run on multiple development branch volumes simultaneously and can finally be merged to the main codebase in the SCM repository. Parallel builds reduce the use of compute and network resources due to the incremental nature of the builds.

**Note:** The NetApp Jenkins framework allows DevOps admins and build administrators to set up global prepush hooks based on organizational code standards. These hooks can include prechecks, syntax validations, and so on. The hooks also minimize bad code check-ins and prevent unnecessary CI triggers.

After every change submitted by the developer to the local SCM or GIT repository, `git push` updates the respective CI code branch, and an incremental build is triggered immediately. Upon successful completion of the developer build, a Snapshot copy is created. This is an iterative process for every code change submitted to the local Git repository and updating of the CI volumes.

Depending on the scheduled CI tests, full nightly builds are performed. Upon successful completion of the build, the stable build is pushed into the correct repository managed by JFrog Artifactory.

**Developer Environment**

When developers want to check out code, they don't have to access the main codebase in the repository. They can select a checkpoint of a successful build and create a workspace from the Jenkins UI for themselves. These workspaces are prepackaged clones that are created instantaneously and mounted on a Docker container, as shown in Figure 6. The developer can then perform unit tests, CI tests, or precheck-in analysis/ Gerrit on the code changes in their respective workspaces before committing the final code changes to the main source code repository.

This approach improves developer productivity because there is no wait time for the entire codebase to be individually cloned (`git clone`) from the original source code repository, sync all the dependencies, and then build it every time. This saves time and compute and network resources in the infrastructure. Regardless of the codebase size, the ownership of the files and directories in the workspace also changes instantly.

Using thin-provisioned FlexClone volumes to clone user workspaces address another challenge with `git clone` and `git push`. If `git push` and `git clone` are used to push and clone many user workspaces, the Git server will probably run out of CPU and memory resources. Every `git clone` creates a pack file on the Git server before the cloning operation. Creating the pack file consumes a lot of CPU, and copy operations take up a lot of memory. In this scenario, the GIT servers should have a lot of CPU and memory, which still cannot scale as the number of developers and the code size grow.

NetApp recommends having a single `git clone` operation to the local SCM on ONTAP and then leveraging the FlexClone feature to create the user workspace. FlexClone volumes scale with the number of user workspaces created for every developer.

**Code Management**

This is essentially a clean-up phase in which only the code that needs to be retained is kept, and Snapshot copies or checkpoints and unused user workspaces are purged (deleted). Not all the Snapshot copies or checkpoints created from all the changes submitted to the source code repository...
and the CI volumes need to be retained. A full build on the CI volume is not a frequent requirement during the build process.

NetApp recommends periodically listing and checking the Snapshot copies generated from successful incremental and CI builds. This procedure allows the CI admin to retain the most recent Snapshot copies that resulted in a successful build (incremental, CI, or full nightly). The rest of the Snapshot copies can be deleted to free up space.

The NetApp Jenkins framework offers a purge policy to clean up the unused Snapshot copies and retain only a specific number of copies. The DevOps admin can specify the number of Snapshot copies to retain when configuring the Jenkins environment.

The purge policy also gives a warning if the number of “busy Snapshot copies” (Snapshot copies acting as parent snapshots for FlexClone volumes) exceeds a specified number. This makes it easy for the DevOps admins to maintain the limit of 255 Snapshot copies per FlexVol volume. After the developers have finished using the workspaces, or any of those workspaces are marked offline, those can be eliminated for better manageability and control.

Managing Build Artifacts and Binaries

Managing and maintaining build artifacts is one of the most challenging tasks that build engineers face. In the NetApp Jenkins framework, all the build dependencies are pulled from JFrog Artifactory, which functions as a central repository for artifacts and binaries. Along with the build dependencies, all of the artifacts are stored in JFrog Artifactory with proper versioning. DevOps admins also have the option to keep zipped copies of their CI builds and preserve the complete build environment as a Docker image through the build artifact management pipeline.

Figure 7) Builds in Docker Registry managed by JFrog Artifactory.

There are several advantages to keeping zip files and Docker images of the entire build environment:

- **If** the developer wants to test a bug reported by users, it is possible to re-create the build environment with all of its dependencies on a Docker container.
- Storage efficiencies like inline compression and deduplication offer better storage efficiency than binary compression for delta change in the binaries.
- On-demand data protection by using a NetApp tool like NetApp SnapCenter® software from JFrog Artifactory protects the stable builds.
- The NetApp StorageGRID object-based storage solution enables collaboration of the artifacts and binaries across different geographic locations to give users local access.
3.3 NetApp Jenkins Framework: Scalable Architecture

Microservices are frequently chosen as the preferred architecture for distributed development, and containers are the unit of deployment. Orchestrators like Docker Swarm and Kubernetes use container services like Docker containers and pods in a cluster for scalability and resiliency. Containers are ephemeral in nature; the data in the container is lost as soon as the container completes its task. There is no persistency, protection, or availability to the data that is degenerated during writing code, test, and build.

The NetApp Jenkins framework architecture benefits the CI workflow and data generated during this process by using Docker Swarm, netappdvp, NSLM APIs, and NFS.

- **Data persistence.** NetApp provides a NetApp Docker Volume plugin (netappdvp) that supports NFS and iSCSI drivers to mount volumes or partitions created in ONTAP by using NSLM APIs. The source code in GitLab, builds in CI environments, and artifacts and binaries in JFrog Artifactory require persistent data. Netappdvp removes the dependency to physically mount shared storage over NFS on specific UNIX nodes for the and enables the containers to seamlessly mount persistent storage from any node in the Swarm cluster.

- **Resiliency.** Docker Swarm provides the resiliency to the Jenkins master that runs as a Docker service in a multinode cluster. If there is any disruption to the master, then a new Jenkins master is respawned on any of the nodes in the cluster so that the surviving slaves can connect with the master. If any slave dies, then the Jenkins master restarts the task.

- **Scalability.** All of the persistent data volumes for JFrog Artifactory, GitLab, and all of the CI partitions and developer workspaces are mounted over NFS, including the Jenkins home directory. Although the docker services can respawn at the compute layer by Docker Swarm, ONTAP allows the services to connect with the home directory and the relevant dataset from the shared storage to complete the task. NFS and netappdvp provide vertical scalability of containers with data persistence.

- **Performance.** NSLM APIs enable provisioning of partitions or volumes for GitLab, Artifactory, CI workloads, and developer workspaces based on service level offerings (SLO)s. The APIs can use predefined SLOs or create custom SLOs based on business requirements. The NSLM APIs also enable load balancing and provide scalability in the data management layer for different workloads tied to specific SLOs during the CI/CD process.

Figure 8) NetApp Jenkins architecture.
As shown in Figure 8, the NetApp Jenkins scalable architecture is based on Docker Swarm, Jenkins, GitLab, and JFrog Artifactory on ONTAP, providing a standard data management platform for on-premises and public cloud environments. Each of these tools and the developer workspaces run in their respective Docker containers, mounting persistent data storage by using netappdvp. The Jenkins masker runs as a Docker service that manages all the jobs in the different pipelines, as described in section 3.2. Each Docker container communicates on port 50000 over Java Network Launch Protocol (JNLP).

All of this integration is complete and is available in a Docker image to be downloaded from netapp.io. The preliminary configuration for this framework needs to be done, because every environment is different. The README file contains step-by-step documentation to set up and configure the NetApp Jenkins framework.

4 Conclusion

The CI pipeline with Jenkins and Docker using NSLM APIs abstracts storage technologies (for example, volume creation, Snapshot copies, and FlexClone volumes) and provides a modular architecture for cloud-native application development. NetApp Docker Volume plugin (netappdvp) makes the storage persistent for all the containers spun up by the Jenkins pipelines. This integration provides scalability in the compute layer with containers, and also instantly creates thin-provisioned user workspaces and multiple copies of builds that are required during QA and testing or deployment in production using FlexClone volumes in the storage layer. The overall workflow and architecture with Jenkins enable greatly reduced build times, improved developer productivity, and reduced storage costs.
Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Document Version History</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 2.0</td>
<td>August 2017</td>
<td>Jenkins Pipeline 2.0 - Bikash Roy Choudhury &amp; Akshay Patil</td>
</tr>
<tr>
<td>Version 1.0</td>
<td>September 2016</td>
<td>Jenkins Free Style Jobs - Bikash Roy Choudhury</td>
</tr>
</tbody>
</table>

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer’s installation in accordance with published specifications.

Copyright Information

Copyright © 2016–2017 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark Information

NETAPP, the NETAPP logo, and the marks listed at http://www.netapp.com/TM are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.