

WHITE PAPER

# REGISTRY-AS-A-SERVICE ON FLASHBLADE

HOSTING PRIVATE DOCKER REGISTRIES

# TABLE OF CONTENTS

- INTRODUCTION** ..... 3
- GOALS AND OBJECTIVES** ..... 4
- CONTAINER ADOPTION IN THE DEVOPS PROCESS** ..... 4
- DOCKER REGISTRY IN CI/CD PROCESS ON FLASHBLADE** ..... 5
- PRIVATE DOCKER REGISTRY** ..... 7
  - Importance of Private Docker Registry ..... 7
- REGISTRY-AS-A-SERVICE ON FLASHBLADE** ..... 9
- ENVIRONMENT DETAILS** ..... 10
  - Linux Nodes ..... 10
  - Network Configuration ..... 10
  - Docker Setup ..... 12
- SECURE DOCKER REGISTRY V2 SETUP ON FLASHBLADE OVER NFS** ..... 19
- DOCKER SWARM CLUSTER SETUP** ..... 32
- CONCLUSION** ..... 33

## INTRODUCTION

The drive toward designing cloud-native applications is enabling and introducing new and modern processes that follow the 12factor guidelines to developing and deploying applications in production. This new generation of application development is all about virtualizing resources and abstracting underlying infrastructure in order to consume it as code.

The elastic nature of the cloud has made microservices the preferred architecture for distributed development, with containers as the unit of deployment. While hypervisors like VMware, Hyper-V, and KVM provide server virtualization, [containers](#) provide virtualization at the application layer. Containers bundle the binaries and libraries that enable the application to run on any platform – including Linux, Windows, and MAC OS.

Cloud-native environments hosted in private datacenters require a standard data management flash platform to provide more predictable performance for heterogeneous workloads, support for parallel builds during the Continuous Integration (CI) process, and hosting for source code and build repositories. Horizontal scalability and availability with high resiliency for applications and their associated datasets are also important requirements.

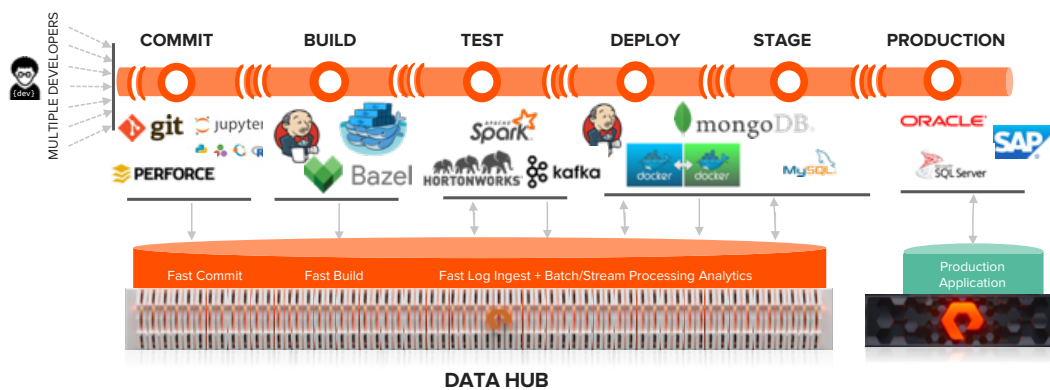


FIGURE 1. Data hub and the Continuous Integration process

The Pure [FlashBlade™](#) product is one of the most advanced scale-out storage solutions ever built. It's a true data hub, purpose-built to handle all the various workloads generated during the CI process. A data hub is a data-centric architecture for storage that powers analytics and artificial intelligence (AI) through agile software development. It enables enterprises to consolidate data silos and share data in today's rapidly-evolving, data-first world. As shown in Figure 1 above, a data hub takes the key strengths of each silo and integrates them into a single, unified platform that includes four must-have qualities: high throughput for file & object, native scale-out, multi-dimensional performance, and massively parallel architecture.

Apart from providing unprecedented performance, availability, and scalability, FlashBlade has the ability to self-heal from any process failure or data store breakdown – without impacting the overall functioning of the application development and deployment process. Finally, by delivering data reduction for all your CI workloads, FlashBlade is cost-efficient.

## GOALS AND OBJECTIVES

In this paper we provide details for configuring and hosting secured private Docker Registries on [FlashBlade](#) over NFS on a single data platform. In doing so, we further demonstrate that FlashBlade can serve as shared infrastructure to enable data availability, scalability, protection, and performance for all workloads in the application development, testing, and delivery workstream.

## CONTAINER ADOPTION IN THE DEVOPS PROCESS

The recent DevOps Research and Analysis (DORA) report indicates that there is big increase in the use of [containers](#) in development and production environments.

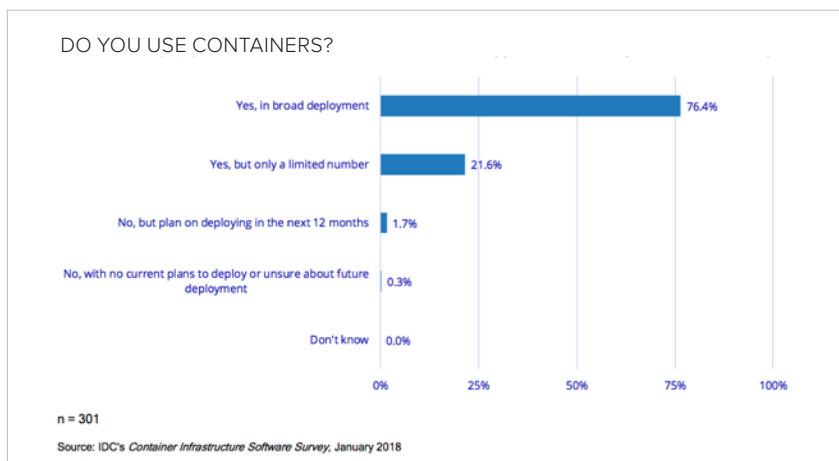


FIGURE 2. State of container adoption

Use of containers during the software development lifecycle (SDLC) provides modularity – enabling agile delivery pipelines for new features and dramatically faster time-to-market. With the increase in the use of containers depicted in Figure 2, Docker images are becoming a more common form of consumption versus Open Virtualization Format (OVF)/ Open Virtualization Appliance (OVA) images. Containers can run on Virtual Machines (VMs) and/or physical machines.

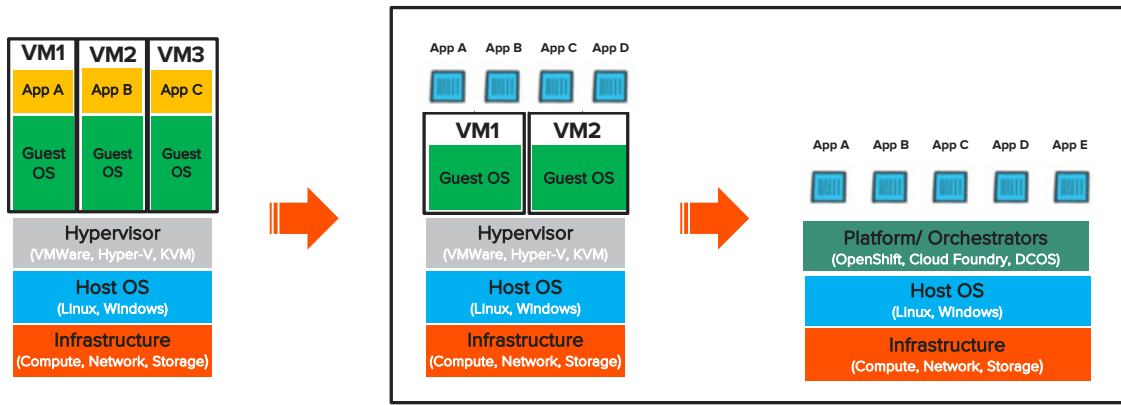


FIGURE 3. Virtual Machines vs. Containers

Figure 3 indicates that applications running on VMs are stateful and resource-constrained compared to containers. If a VM or host goes down, the application is lost. Containers like Docker provide application virtualization that's lightweight and can be packaged and scaled in any run time environment. Containers are traditionally stateless but require persistent storage for databases, credentials, keys, configuration, and password data that needs to be shared and reused by applications, source code, binary repositories, etc.

### DOCKER REGISTRY IN CI/CD PROCESS ON FLASHBLADE

Docker Registry is primarily used for hosting, packaging, and distributing docker images among users, teams, and lines of business. Unlike other source code repositories, Docker is a shared repository. Figure 4 illustrates that images and packaging in Docker are becoming more popular in the software development lifecycle. But higher volumes of Docker images and packages require scalable capacity and performance in cost-efficient storage.

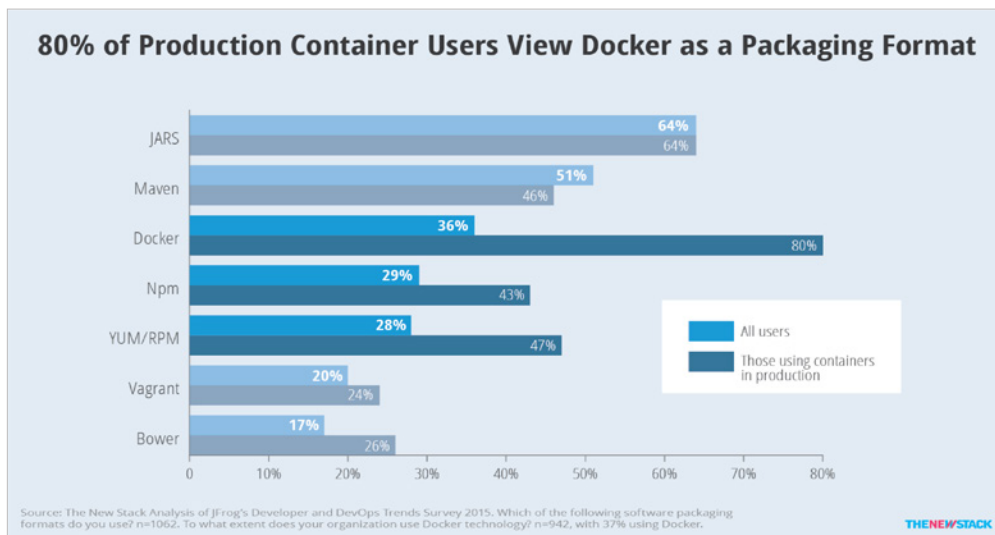


FIGURE 4. Higher use of Docker packaging format over others

Docker Registry is an integral part of the Continuous Integration (CI) and Continuous Deployment (CD) process for developing and deploying cloud-native applications. A build tool like Jenkins can be automated to run parallel builds, at any given time, and push the resulting images to the Docker Registry upon successful completion of CI tests. These Docker images are generated and packaged at the end of the build process during the development cycle.

The Docker images are then automatically promoted to Continuous Deployment (CD) and later released to production using different CD tools like Jenkins and UrbanCode, or Platform-as-a Service (PaaS) tools like RedHat OpenShift, CloudFoundry, and Datacenter Operating System (DCOS) from Mesosphere.

Figure 5 shows that FlashBlade provides a standard data platform for the various development and deployment workloads comprising the software development life cycle (SDLC). FlashBlade is a single, cost-efficient platform that delivers scalable performance and capacity alongside high bandwidth, and is well-suited to handling software builds, which are compute-intensive workloads with high metadata operations. Configuring builds and Docker Registry over NFS on [FlashBlade](#) allows continuity in the development process with scalability and a reduced data footprint.

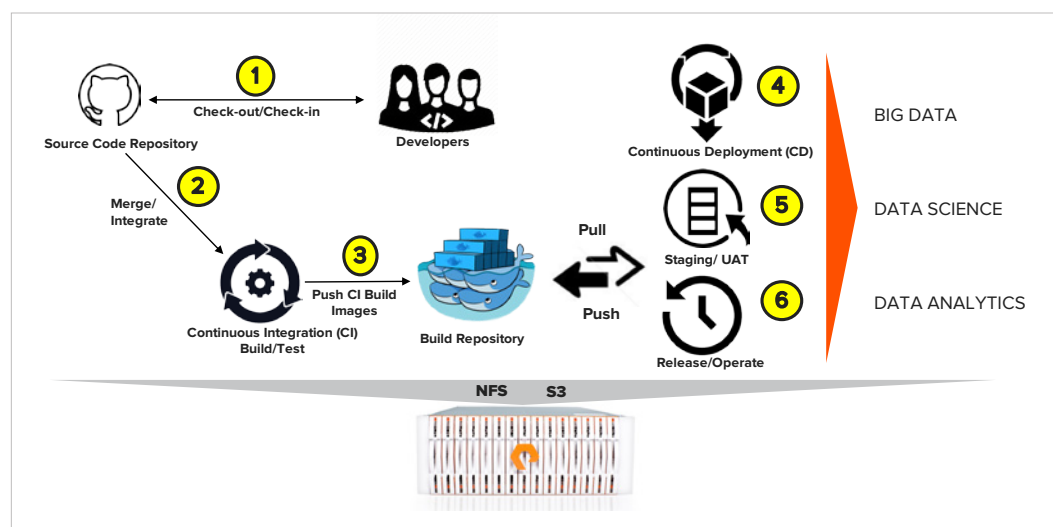


FIGURE 5. Docker Registry in the CI/CD process

While Docker containers are ephemeral in nature, Docker images are immutable pieces of binaries that include all the tools – binaries, configuration files, etc. – they need to run or execute. The Dockerfile is the recipe to build these Docker images.

Dockerfile helps to bootstrap a filesystem in user space. That means Ubuntu, CentOS, or Debian can run as a container (user space) on a RedHat Enterprise Linux (RHEL) host. Unlike source code versions, the Docker images have “tags” for different versions, as shown in Figure 6 below. These Docker images are packaged with the right set of tools, patches, etc., in layers on top of the base image.

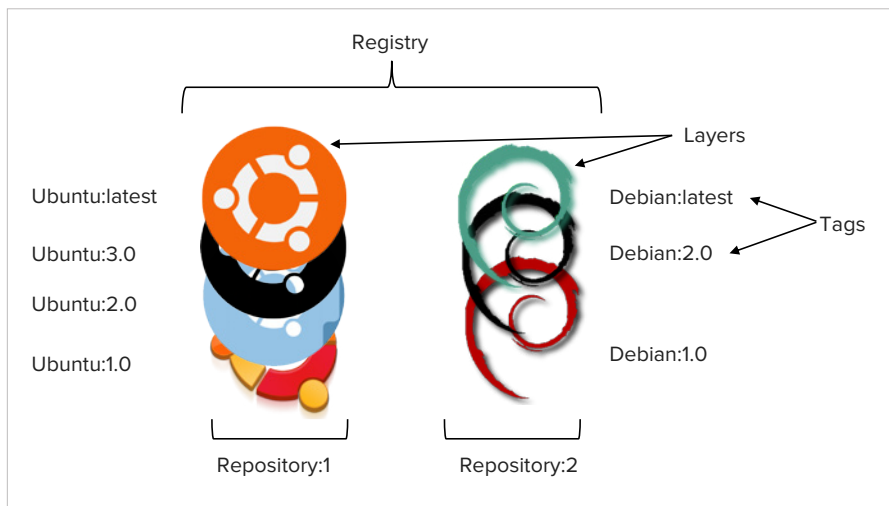


FIGURE 6. Docker Repositories vs. Registry

Docker Registries contain different repositories for each of Docker image; there may, however, be different versions (tags) of the images in a single repository. Large enterprises that use [containers](#) in a CI process may experience up to 250,000 push/pull operations of Docker images in development and deployment in a day – operations that are shared by different teams and lines of businesses.

### PRIVATE DOCKER REGISTRY

The Docker Registry service can be hosted in public as well as in private. One of the most common Registry services is provided by DockerHub, and it includes some free offerings to individuals and small businesses. When confronted with a higher number of Docker image repositories, DockerHub and other publicly hosted registries often develop performance bottlenecks due to less flexibility to integrate with other development tools, security challenges, and limited capacity scaling.

Private Docker Registry is a private hosting service that can be configured using the open source Docker distribution that supports Docker Registry v2. (Other configuration choices include Docker Trusted Registry (DTR) from Docker Datacenter and JFrog Artifactory.) Private Docker Registry can be hosted in secured and unsecured modes internally within a private datacenter.

### Importance of Private Docker Registry

While many public clouds and commercial off-the-shelf (COTS) products are available to distribute Docker images using Docker Registry, private Docker registries are still relevant and used by organizations. A recent survey indicates that self-hosting is the most common form of hosting and distributing Docker images as adoption of containers increases.

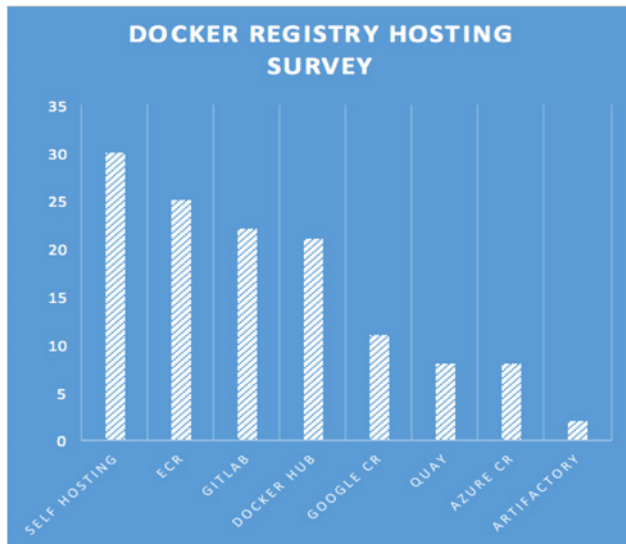


FIGURE 7. Self-hosting Docker Registry leading over other hosting options<sup>1</sup>

There are many reasons why organizations choose to use private Docker Registry over publically hosted registry services. Apart from being open-source software and free to download, Docker Registry is more popular than others because it offers:

1. Security and privacy for proprietary code. Private Docker registries provide granular access and integration with LDAP and OAuth, etc., in a native production environment for user authorization and authentication, along with audit logs.
2. A disconnected environment where there is no internet, and the environment lives behind a firewall. By contrast, mirroring a publically hosted registry between remotes sites is slow and resource consuming.
3. Freedom from network congestion and performance bottlenecks that can lead to outages with a large number of push/pull operations for Docker images on public cloud-hosted services.
4. Cost-effective capacity scaling, as opposed to public cloud-hosted registry services that charge by the storage capacity used – and the \$/GB/month increases with the size and number of the repositories.
5. No limitations on development tools and integrations in the SDLC for a CI/CD workflow, as compared to the limitations inherent in publically hosted options.

Docker recommends using flash as the form of shared storage for private Docker Registries. The Pure FlashBlade product is the perfect fit: persistent storage that provides an advanced data hub with scalable performance, storage efficiency, and reduced cost to store, share, and collaborate among developers. As mentioned in the earlier section, private Docker Registry can be configured over NFS along with other phases of the CI/CD, like source code repository, build/test, [Blue/Green deployments](#), [A/B testing](#), and release to production. However, a private Docker Registry can also be configured over S3 natively on FlashBlade for better scalability and performance. This latter option is not within the scope of the current document.

<sup>1</sup> Source: [https://www.reddit.com/r/docker/comments/5owr9b/poll\\_where\\_is\\_your\\_docker\\_registry\\_hosted/#bottom-comments](https://www.reddit.com/r/docker/comments/5owr9b/poll_where_is_your_docker_registry_hosted/#bottom-comments)



## REGISTRY-AS-A-SERVICE ON FLASHBLADE

A secured private Docker Registry can be configured in many different ways on [FlashBlade](#) over NFS for hosting and collaboration. Multiple private Docker Registries can be configured on a single data platform like FlashBlade for scalable performance and capacity. However, this section provides a very simple and detailed step by step process to install and configure Docker Registry version 2 on FlashBlade.

We illustrate the setup and functioning of the private Docker Registry with a Linux node that has an NFS mount from FlashBlade, as shown in Figure 8. Docker images are pushed in to the Docker Registry from a build server after successfully merging and testing the code changes in the CI phase. Multiple images are eventually pulled from the Registry to be deployed by end-users for staging, and eventually released into production. FlashBlade provides continuous availability and eliminates performance bottlenecks when users perform Docker push/pull operations at scale.

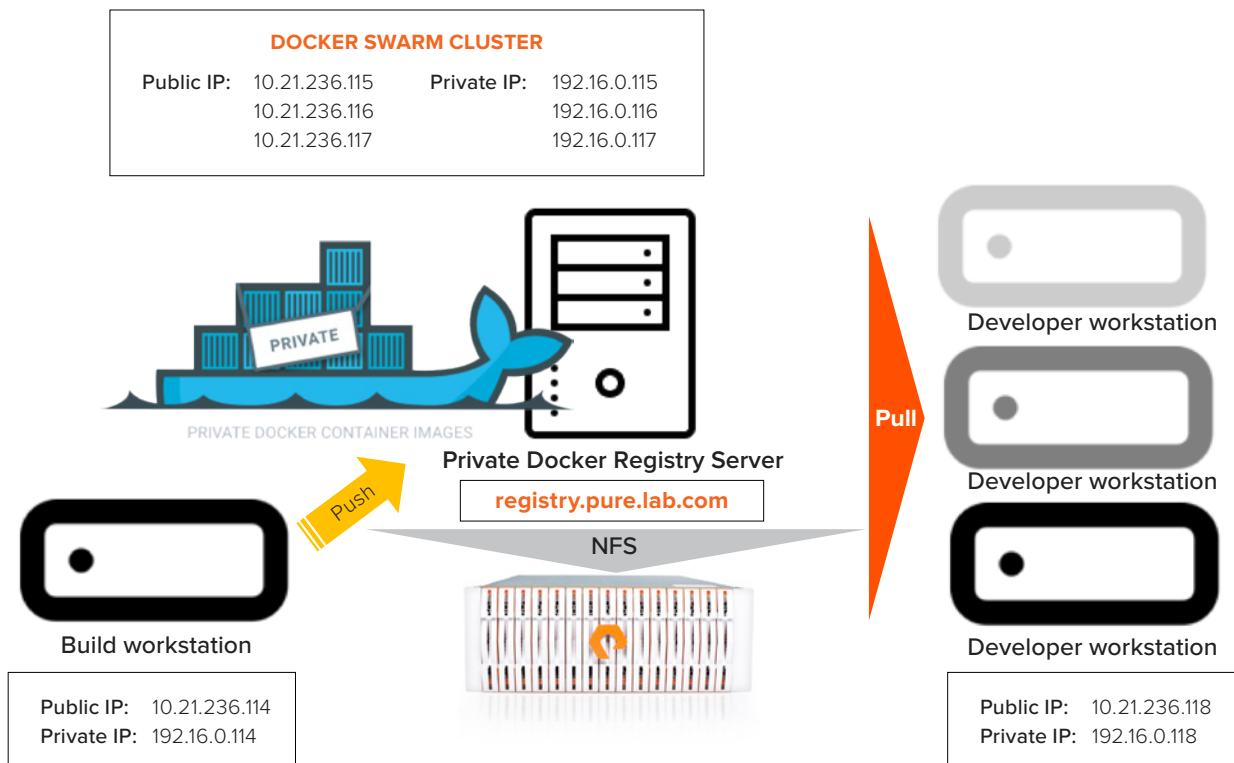


FIGURE 8. Docker operation flow to and from the Docker Registry

## ENVIRONMENT DETAILS

The secured private Registry set up for purposes of this documentation had the following configuration:

### Linux Nodes

Clusters can be configured using Kubernetes, Openshift, and Docker Swarm orchestrator tools. Docker registry in OpenShift can also be configured to run on [FlashBlade](#). For the purposes of this paper, Docker Swarm was used to create a cluster. All the Linux nodes are part of the Docker swarm cluster and build server, and all the user workstations that perform the Docker push/pull operations are configured with the following version of CentOS Linux. It is recommended to use CentOS/RHEL version 6.5 and later for this configuration. It is also recommended to have at least 50GB of local HDD, 8GB RAM, and two core processors on each of the Linux nodes.

```
[root@sn1-r720-g09-17 ~]# uname -m && cat /etc/redhat-release
x86_64
CentOS Linux release 7.5.1804 (Core)
```

### Network Configuration

The network configuration is the critical part of this setup. For a very basic secured private Registry setup, the following network, docker daemon, and name resolution have to be configured.

- **Public IP network** for Docker Swarm and applications that run in docker containers for external communication – for example, 10.21.236.115 – 10.21.236.117
- **Private IP network** in a separate VLAN from the public network for the secure Docker Registry to communicate – 192.16.0.115 – 192.16.0.117. This is not routable to the public network. All the nodes that access this private Registry will have a similar network configuration. If a node without these network settings tries to perform “Docker push/pull”, we will end up with the following error:

```
[root@sn1-r720-g09-23 ~]# docker push registry.pure.lab.com:5000/node
The push refers to repository [registry.pure.lab.com:5000/node]
Get https://registry.pure.lab.com:5000/v2/: dial tcp 185.53.178.9:5000: connect: connection refused
```

**Connection refused** means there is no communication happening from the node on which the Docker push/pull operation is happening to the Registry nodes. If the private IP (192.16.0.x) is not **pingable** between the nodes, **no route to host**, another common network error, will occur while trying to push/pull Docker images from hosts with incorrect network settings.

- In this sample setup, the Registry name is **registry.pure.lab.com** and an A-record is created in the **pure.lab.com** domain with the IP address 192.16.0.115. Configuring DNS is beyond the scope of this document.

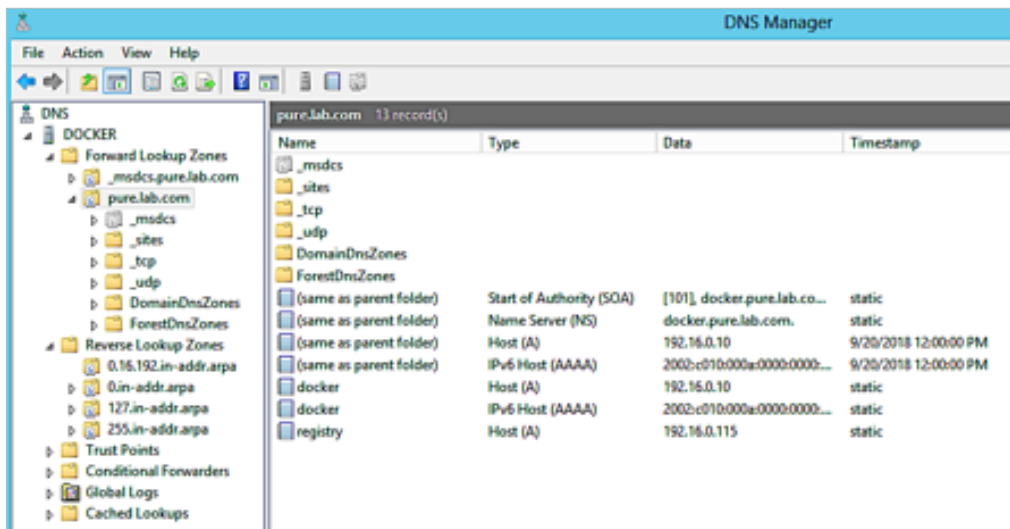


FIGURE 9. Registry record included in the pure.lab.com domain

Include the IP address of the nameserver pure.lab.com domain in the `/etc/resolv.conf` file in each of the Linux hosts.

```
[root@sn1-r720-g09-17 ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search puretec.purestorage.com
search pure.lab.com
nameserver 10.21.93.16
nameserver 192.16.0.10
```

If there is no DNS in the environment, the local `/etc/hosts` file can be used for name resolution. This Fully Qualified Domain Name (FQDN) has to be added to the local hosts file (`/etc/hosts`) on all the Linux hosts on that part of the Docker swarm cluster and the nodes that will perform the Docker push/pull operations.

```
[root@sn1-r720-g09-17 ~]# cat /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
192.16.0.115 registry.pure.lab.com registry
```

Confirm that the `registry.pure.lab.com` can be pinged with the correct name resolution from any host that is configured to use the private Docker Registry.

```
[root@sn1-r720-g09-15 ~]# ping registry.pure.lab.com
PING registry.pure.lab.com (192.16.0.115) 56(84) bytes of data.
64 bytes from registry.pure.lab.com (192.16.0.115): icmp_seq=1 ttl=64 time=0.211 ms
```

## Docker Setup

The following steps list how **docker-ce** (community edition) can be installed and configured on each of the swarm nodes and the rest of the Linux nodes that would perform the Docker push/pull operation to the private Docker Registry.

- Disable SELinux on all the Linux nodes.

```
[root@sn1-r720-g09-17 ~]# setenforce 0
setenforce: SELinux is disabled
[root@sn1-r720-g09-17 ~]# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/
sysconfig/selinux
```

- Enable the bridge for the Docker – **br\_netfilter** kernel module

```
[root@sn1-r720-g09-17 ~]# modprobe br_netfilter
[root@sn1-r720-g09-17 ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

- Disable swap on all the Linux nodes on which Docker is configured.

```
[root@sn1-r720-g09-17 ~]# swapoff -a
```

- Comment out the swap entry in **/etc/fstab** to make the change persistent across reboots.

```
[root@sn1-r720-g09-17 ~]# cat /etc/fstab |grep swap
#/dev/mapper/vg0-lv_swap swap swap defaults 0 0
```

- Install the package dependencies for **docker-ce**

```
[root@sn1-r720-g09-17 ~]# yum install -y yum-utils device-mapper-persistent-data lvm2
```

- Add the docker repository on the Linux nodes and install **docker-ce** on each of them.

```
[root@sn1-r720-g09-17 ~]# yum-config-manager --add-repo https://download.docker.com/linux/centos/
docker-ce.repo
[root@sn1-r720-g09-17 ~]# yum install -y docker-ce
```

- Reboot the Linux nodes.

- After the reboot, enable and start Docker on the Linux nodes.

```
[root@sn1-r720-g09-17 ~]# systemctl start docker && systemctl enable docker
```

- Check the Docker version after the install on the Linux nodes.

```
[root@sn1-r720-g09-17 ~]# docker --version
Docker version 18.06.1-ce, build e68fc7a
```

Note: Docker 18.06.1-c2 was used for this validation.

- A private IP network: 172.17.0.5 is configured to the default bridge network that is used by the Docker engine. A `/etc/docker/daemon.json` file has to be manually created after Docker is installed in the all the Linux nodes including Swarm cluster that performs the Docker pull/push operations to the private Docker Registry.

The private Docker Registry also used port 5000. As `registry.pure.lab.com` is used as a secured Registry, the localhosts on port 5000 should be called unsecured, and those entries must be included in the `/etc/docker/daemon.json` file.

```
[root@sn1-r720-g09-17 ~]# cat /etc/docker/daemon.json
{
    "bip": "172.17.0.5/16",
    "insecure-registries" : ["localhost:5000",
                            "127.0.0.1:5000"
    ],
    "storage-driver": "overlay2",
    "storage-opts": [
        "overlay2.override_kernel_check=true"
    ]
}
```

- After the `/etc/docker/daemon.json` file is created and configured, Docker has to be restarted.

```
[root@sn1-r720-g09-17 ~]# systemctl restart docker
```

- A `docker info` command should yield something similar to the output listed below.

```
[root@sn1-r720-g09-17 ~]# docker info
Containers: 2
Running: 2
Paused: 0
Stopped: 0
Images: 1
Server Version: 18.06.1-ce
Storage Driver: overlay2
Backing Filesystem: extfs
Supports d_type: true
Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
Volume: local
```

Network: bridge host macvlan null overlay  
Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog  
Swarm: active  
NodeID: 4z5leyoscu1cdvh8rf24wuxnu  
Is Manager: true  
ClusterID: wu19o3qoechognglazrk4rlbh  
Managers: 1  
Nodes: 3  
Orchestration:  
  Task History Retention Limit: 1  
Raft:  
  Snapshot Interval: 10000  
  Number of Old Snapshots to Retain: 0  
  Heartbeat Tick: 1  
  Election Tick: 10  
Dispatcher:  
  Heartbeat Period: 5 seconds  
CA Configuration:  
  Expiry Duration: 3 months  
  Force Rotate: 0  
Autolock Managers: false  
Root Rotation In Progress: false  
Node Address: 10.21.236.115  
Manager Addresses:  
  10.21.236.115:2377  
Runtimes: runc  
Default Runtime: runc  
Init Binary: docker-init  
containerd version: 468a545b9edcd5932818eb9de8e72413e616e86e  
runc version: 69663f0bd4b60df09991c08812a60108003fa340  
init version: fec3683  
Security Options:  
  seccomp  
    Profile: default  
Kernel Version: 3.10.0-862.11.6.el7.x86\_64  
Operating System: CentOS Linux 7 (Core)  
OSType: linux  
Architecture: x86\_64

```
CPU: 48
Total Memory: 503.7GiB
Name: sn1-r720-g09-17.puretec.purestorage.com
ID: 2LKE:UNAS:AHPU:74TB:Y65T:DDKO:NJOJ:KXY6:57AT:DACO:07JE:7FMZ
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
  localhost:5000
  127.0.0.1:5000
  127.0.0.0/8
Live Restore Enabled: false
```

- The network should look something like the following:

```
root@sn1-r720-g09-17 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eno3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether ec:f4:bb:d1:f7:7c brd ff:ff:ff:ff:ff:ff
3: eno4: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether ec:f4:bb:d1:f7:7d brd ff:ff:ff:ff:ff:ff
4: eno1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether ec:f4:bb:d1:f7:78 brd ff:ff:ff:ff:ff:ff
5: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether ec:f4:bb:d1:f7:7a brd ff:ff:ff:ff:ff:ff
6: enp66s0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 9000 qdisc mq master bond0 state UP group
default qlen 1000
    link/ether f4:52:14:97:1c:41 brd ff:ff:ff:ff:ff:ff
7: enp66s0d1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether f4:52:14:97:1c:42 brd ff:ff:ff:ff:ff:ff
8: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 9000 qdisc noqueue state UP group default qlen
1000
```

```

link/ether f4:52:14:97:1c:41 brd ff:ff:ff:ff:ff:ff
inet6 fe80::f652:14ff:fe97:1c41/64 scope link
    valid_lft forever preferred_lft forever
9: bond0.2236@bond0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UP group default
qlen 1000
    link/ether f4:52:14:97:1c:41 brd ff:ff:ff:ff:ff:ff
    inet 10.21.236.115/24 brd 10.21.236.255 scope global noprefixroute bond0.2236
        valid_lft forever preferred_lft forever
    inet6 fe80::f652:14ff:fe97:1c41/64 scope link
        valid_lft forever preferred_lft forever
10: bond0.1116@bond0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UP group default
qlen 1000
    link/ether f4:52:14:97:1c:41 brd ff:ff:ff:ff:ff:ff
    inet 192.16.0.115/24 brd 192.16.0.255 scope global noprefixroute bond0.1116
        valid_lft forever preferred_lft forever
    inet6 fe80::f652:14ff:fe97:1c41/64 scope link
        valid_lft forever preferred_lft forever
11: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:d4:b1:7a:b5 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.5/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever

```

- Install and enable/start the **httpd** on all the Linux nodes.

```
[root@sn1-r720-g09-11 ~]# yum install -y httpd
```

```
[root@sn1-r720-g09-11 ~]# systemctl enable httpd && systemctl start httpd
```

```
[root@sn1-r720-g09-11 ~]# systemctl status httpd
```

```
□ httpd.service - The Apache HTTP Server
```

```
Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
```

```
Active: active (running) since Mon 2018-09-10 22:05:32 PDT; 1 weeks 5 days ago
```

```
Docs: man:httpd(8)
```

```
man:apachectl(8)
```

```
Process: 27703 ExecReload=/usr/sbin/httpd $OPTIONS -k graceful (code=exited, status=0/SUCCESS)
```

```
Main PID: 1779 (httpd)
```

```
Status: "Total requests: 0; Current requests/sec: 0; Current traffic: 0 B/sec"
```

```
Tasks: 7
```

```
Memory: 8.3M
```

```
CGroup: /system.slice/httpd.service
```



```
├─ 1779 /usr/sbin/httpd -DFOREGROUND
├─27719 /usr/sbin/httpd -DFOREGROUND
├─27720 /usr/sbin/httpd -DFOREGROUND
├─27721 /usr/sbin/httpd -DFOREGROUND
├─27722 /usr/sbin/httpd -DFOREGROUND
├─27723 /usr/sbin/httpd -DFOREGROUND
└─27724 /usr/sbin/httpd -DFOREGROUND
```

```
Sep 10 22:05:32 sn1-r720-g09-11.puretec.purestorage.com systemd[1]: Starting The Apache HTTP Server...
Sep 10 22:05:32 sn1-r720-g09-11.puretec.purestorage.com systemd[1]: Started The Apache HTTP Server.
Sep 14 03:10:01 sn1-r720-g09-11.puretec.purestorage.com systemd[1]: Reloaded The Apache HTTP Server.
Sep 16 03:19:02 sn1-r720-g09-11.puretec.purestorage.com systemd[1]: Reloaded The Apache HTTP Server.
Sep 23 03:47:02 sn1-r720-g09-11.puretec.purestorage.com systemd[1]: Reloaded The Apache HTTP Server.
[root@sn1-r720-g09-11 ~]#
```

- Configure a self-signed certificate to secure private Docker Registry for the **registry.pure.lab.com** node after creating a new **/certs** directory.

```
[root@sn1-r720-g09-17 ~]# mkdir -p /certs
```

```
[root@sn1-r720-g09-17 ~]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout /certs/ca.key -x509
-days 365 -out /certs/ca.crt
```

```
Generating a 4096 bit RSA private key
```

```
.....
.....++
.....++
```

```
writing new private key to '/certs/ca.key'
```

```
-----
```

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
```

```
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
```

```
-----
```

```
Country Name (2 letter code) [XX]:US
```

```
State or Province Name (full name) []:CA
```

```
Locality Name (eg, city) [Default City]:MV
```

```
Organization Name (eg, company) [Default Company Ltd]:Pure Storage
```

```
Organizational Unit Name (eg, section) []:FlashBlade
Common Name (eg, your name or your server's hostname) []:registry.pure.lab.com
Email Address []:bikash@purestorage.com
[root@sn1-r720-g09-17 ~]#
```

```
[root@sn1-r720-g09-17 ~]# ls -al /certs/
total 16
drwxr-xr-x  2 root root 4096 Sep 15 16:21 .
dr-xr-xr-x. 19 root root 4096 Sep 15 23:47 ..
-rw-r--r--  1 root root 2155 Sep 15 16:21 ca.crt
-rw-r--r--  1 root root 3272 Sep 15 16:21 ca.key
```

- Restart Docker.

```
[root@sn1-r720-g09-17 ~]# systemctl restart docker
```

```
[root@sn1-r720-g09-17 ~]# systemctl status docker
```

```
• docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2018-09-10 22:05:28 PDT; 1 weeks 3 days ago
     Docs: https://docs.docker.com
  Main PID: 1833 (dockerd)
    Tasks: 154
   Memory: 2.5G
    CGroup: /system.slice/docker.service
            └─1833 /usr/bin/dockerd
                └─2058 docker-containerd --config /var/run/docker/containerd/containerd.toml

Sep 10 22:05:28 sn1-r720-g09-17.puretec.purestorage.com dockerd[1833]: time="2018-09-10T22:05:28.639964096-07:00" level=info msg="Loading containers: done."
Sep 10 22:05:28 sn1-r720-g09-17.puretec.purestorage.com dockerd[1833]: time="2018-09-10T22:05:28.684068144-07:00" level=info msg="Docker daemon" commit=e68fc7a graphdriver(s)=overlay2 version=18.06.1-ce
Sep 10 22:05:28 sn1-r720-g09-17.puretec.purestorage.com dockerd[1833]: time="2018-09-10T22:05:28.684826304-07:00" level=info msg="Daemon has completed initialization"
Sep 10 22:05:28 sn1-r720-g09-17.puretec.purestorage.com dockerd[1833]: time="2018-09-10T22:05:28.714378495-07:00" level=info msg="API listen on /var/run/docker.sock"
Sep 10 22:05:28 sn1-r720-g09-17.puretec.purestorage.com systemd[1]: Started Docker Application Container Engine.
```

## SECURE DOCKER REGISTRY V2 SETUP ON FLASHBLADE OVER NFS

Setup and configure Docker Registry version v2. **docker-registry** is now deprecated and **docker-distribution** has replaced **docker-registry**.

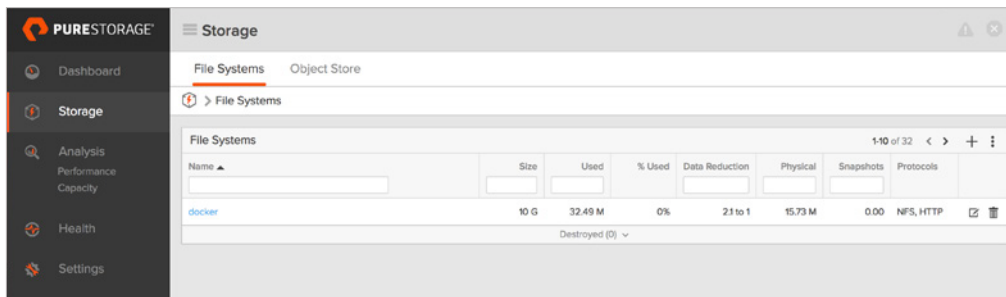
```
[root@sn1-r720-g09-21 ~]# yum install -y docker-distribution
```

- After docker-distribution is installed, a new **config.yml** file will be created under **/etc/docker-distribution/registry**. The **config.yml** file has the environment information of the private Docker Registry. For the secure private Docker Registry, we are using a file share from FlashBlade over NFS.

```
[root@sn1-r720-g09-17 ~]# cat /etc/docker-distribution/registry/config.yml
```

```
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
http:
  addr: :5000
  tls:
    certificate: /certs/ca.crt
    key: /certs/ca.key
  headers:
    X-Content-Type-Options: [nosniff]
```

- Before the docker-distribution service is started, the default registry path **/var/lib/registry** on the Linux host has to mount the NFS share from the FlashBlade. The following NFS share **docker** is created on the FlashBlade.



The screenshot shows the Pure Storage FlashBlade management console. The left sidebar contains navigation options: Dashboard, Storage, Analysis (Performance, Capacity), Health, and Settings. The main area is titled 'Storage' and shows 'File Systems' and 'Object Store' tabs. Under 'File Systems', there is a table listing the 'docker' share. The table has columns for Name, Size, Used, % Used, Data Reduction, Physical, Snapshots, and Protocols. The 'docker' share is listed with a size of 10 G, 32.49 M used, 0% usage, 2:1 data reduction, 15.73 M physical size, 0.00 snapshots, and NFS, HTTP protocols.

Name	Size	Used	% Used	Data Reduction	Physical	Snapshots	Protocols
docker	10 G	32.49 M	0%	2:1	15.73 M	0.00	NFS, HTTP

FIGURE 10. NFS share **docker** created on FlashBlade

- On the **registry.pure.lab.com** Linux node, the **/var/lib/registry** is mounted to the NFS share **docker** created on FlashBlade. All three nodes in the swarm cluster mount to the same NFS share **docker** from their respective nodes for load-balancing.

```
[root@sn1-r720-g09-17 ~]# cat /etc/fstab

10.21.236.202:/docker          /var/lib/registry nfs      hard,rw,bg,vers=3,tcp,nolock,timeo=600

10.21.236.202:/docker          /var/lib/registry nfs      hard,rw,bg,vers=3,tcp,nolock,timeo=600

10.21.236.202:/docker          /var/lib/registry nfs      hard,rw,bg,vers=3,tcp,nolock,timeo=600

[root@sn1-r720-g09-17 ~]# ls -al /var/lib/registry/
total 4
drwxr-xr-x  1 root root    0 Sep 10 10:41 .
drwxr-xr-x. 39 root root 4096 Sep 15 15:10 ..
drwxr-xr-x  1 root root    0 Sep 16 08:16 docker
drwxr-xr-x  1 root root    0 Sep 10 10:41 .fast-remove
drwxr-xr-x  1 root root    0 Sep 10 10:41 .snapshot
```

- Enable and start **docker-distribution** on the **registry.pure.lab.com** node.

```
[root@sn1-r720-g09-17 ~]# systemctl enable docker-distribution.service

[root@sn1-r720-g09-17 ~]# systemctl start docker-distribution.service

[root@sn1-r720-g09-17 ~]# systemctl status docker-distribution.service
□ docker-distribution.service - v2 Registry server for Docker
   Loaded: loaded (/usr/lib/systemd/system/docker-distribution.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2018-09-21 10:45:38 PDT; 2s ago
 Main PID: 10210 (registry)
    Tasks: 12
   Memory: 7.0M
    CGroup: /system.slice/docker-distribution.service
           └─10210 /usr/bin/registry serve /etc/docker-distribution/registry/config.yml

Sep 21 10:45:38 sn1-r720-g09-17.puretec.purestorage.com systemd[1]: Started v2 Registry server for Docker.
```

```

Sep 21 10:45:38 sn1-r720-g09-17.puretec.purestorage.com systemd[1]: Starting v2 Registry server for
Docker...
Sep 21 10:45:38 sn1-r720-g09-17.puretec.purestorage.com registry[10210]: time="2018-09-
21T10:45:38-07:00" level=warning msg="No HTTP secret provided - generated random secret. This may
cause problems with uploads if multiple registries are behind a load-bala...
Sep 21 10:45:38 sn1-r720-g09-17.puretec.purestorage.com registry[10210]: time="2018-09-
21T10:45:38-07:00" level=info msg="redis not configured" go.version=go1.9.4 instance.id=be517d44-2938-
4214-92cd-d6dac5cf4725 version="v2.6.2+unknown"
Sep 21 10:45:38 sn1-r720-g09-17.puretec.purestorage.com registry[10210]: time="2018-09-
21T10:45:38-07:00" level=info msg="Starting upload purge in 45m0s" go.version=go1.9.4 instance.
id=be517d44-2938-4214-92cd-d6dac5cf4725 version="v2.6.2+unknown"
Sep 21 10:45:38 sn1-r720-g09-17.puretec.purestorage.com registry[10210]: time="2018-09-
21T10:45:38-07:00" level=info msg="using inmemory blob descriptor cache" go.version=go1.9.4 instance.
id=be517d44-2938-4214-92cd-d6dac5cf4725 version="v2.6.2+unknown"
Sep 21 10:45:38 sn1-r720-g09-17.puretec.purestorage.com registry[10210]: time="2018-09-
21T10:45:38-07:00" level=info msg="listening on [::]:5000, tls" go.version=go1.9.4 instance.id=be517d44-
2938-4214-92cd-d6dac5cf4725 version="v2.6.2+unknown"

```

Hint: Some lines were ellipsized, use `-l` to show in full.

- Configure secure private Docker Registry as a service in the Docker swarm cluster.

```

[root@sn1-r720-g09-17 ~]# docker pull registry:2
2: Pulling from library/registry
Digest: sha256:5a156ff125e5a12ac7fdec2b90b7e2ae5120fa249cf62248337b6d04abc574c8
Status: Downloaded newer image for registry:2

```

```

[root@sn1-r720-g09-17 ~]# docker service create --name registry -p 5000 -e DOCKER_REGISTRY_CONFIG=/
etc/docker-distribution/registry/config.yml registry:2
2ffjngr0w29awk25dskpzhs8
overall progress: 1 out of 1 tasks
1/1: running [=====]
verify: Service converged

```

```

[root@sn1-r720-g09-17 ~]# docker service ls

```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
2ffjngr0w29a	registry	replicated	1/1	registry:2	*:3000->5000/tcp

- Scale the **registry** service to three replicas that can load-balance on three swarm cluster nodes.

```
root@sn1-r720-g09-17 ~]# docker service scale registry=3
registry scaled to 3
overall progress: 3 out of 3 tasks
1/3: running [=====>]
2/3: running [=====>]
3/3: running [=====>]
verify: Service converged
```

```
[root@sn1-r720-g09-17 ~]# docker service ps registry
```

ID	NAME	IMAGE	NODE
0hdg9dnocd3q	registry.1	registry:2	sn1-r720-g09-19.puretec.purestorage.com
Running	Running 10 minutes ago		
mnmptowakqf	registry.2	registry:2	sn1-r720-g09-17.puretec.purestorage.com
Running	Running 53 seconds ago		
vwi8low0ynku	registry.3	registry:2	sn1-r720-g09-21.puretec.purestorage.com
Running	Running 54 seconds ago		

- After the Docker, Docker swarm cluster, and Docker Registry configuration, the network has now created **Docker\_gwbridge** and **veth0** interfaces for the Registry containers.

```
[root@sn1-r720-g09-17 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eno3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether ec:f4:bb:d1:f7:7c brd ff:ff:ff:ff:ff:ff
3: eno4: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether ec:f4:bb:d1:f7:7d brd ff:ff:ff:ff:ff:ff
4: eno1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether ec:f4:bb:d1:f7:78 brd ff:ff:ff:ff:ff:ff
5: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether ec:f4:bb:d1:f7:7a brd ff:ff:ff:ff:ff:ff
```

```

6: enp66s0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 9000 qdisc mq master bond0 state UP group
default qlen 1000
    link/ether f4:52:14:97:1c:41 brd ff:ff:ff:ff:ff:ff
7: enp66s0d1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether f4:52:14:97:1c:42 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::2f23:a2f9:cd62:4b3a/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
8: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 9000 qdisc noqueue state UP group default qlen
1000
    link/ether f4:52:14:97:1c:41 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::f652:14ff:fe97:1c41/64 scope link
        valid_lft forever preferred_lft forever
9: bond0.2236@bond0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UP group default
qlen 1000
    link/ether f4:52:14:97:1c:41 brd ff:ff:ff:ff:ff:ff
    inet 10.21.236.115/24 brd 10.21.236.255 scope global noprefixroute bond0.2236
        valid_lft forever preferred_lft forever
    inet6 fe80::f652:14ff:fe97:1c41/64 scope link
        valid_lft forever preferred_lft forever
10: bond0.1116@bond0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UP group default
qlen 1000
    link/ether f4:52:14:97:1c:41 brd ff:ff:ff:ff:ff:ff
    inet 192.16.0.115/24 brd 192.16.0.255 scope global noprefixroute bond0.1116
        valid_lft forever preferred_lft forever
    inet6 fe80::f652:14ff:fe97:1c41/64 scope link
        valid_lft forever preferred_lft forever
11: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:dd:f0:7a:22 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.5/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
12: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:8b:76:da:80 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker_gwbridge
        valid_lft forever preferred_lft forever
    inet6 fe80::42:8bff:fe76:da80/64 scope link
        valid_lft forever preferred_lft forever
18: veth42e215e@if17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge
state UP group default

```

```

link/ether 32:d3:2c:d3:b2:75 brd ff:ff:ff:ff:ff:ff link-netnsid 1
inet6 fe80::30d3:2cff:fed3:b275/64 scope link
    valid_lft forever preferred_lft forever
22: veth3d4600b@if21: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge
state UP group default
    link/ether 96:69:d6:6b:4f:0e brd ff:ff:ff:ff:ff:ff link-netnsid 2
    inet6 fe80::9469:d6ff:fe6b:4f0e/64 scope link
        valid_lft forever preferred_lft forever
26: vetha0ec353@if25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge
state UP group default
    link/ether 92:ad:48:04:ed:3e brd ff:ff:ff:ff:ff:ff link-netnsid 3
    inet6 fe80::90ad:48ff:fe04:ed3e/64 scope link
        valid_lft forever preferred_lft forever

```

- The above virtual interfaces and bridge gateway listing checks out that the Registry configuration is good. Now the secure private Docker Registry server configuration is complete.

Configure the build node for **docker push**.

- As mentioned in the previous section, Docker is installed on the build node where the Docker push operation is performed, in order to push the different Docker images to the Registry server **registry.pure.lab.com**.

- Create a new directory with the Registry server name and port 5000.

```
[root@sn1-r720-g09-15 ~]# mkdir -p /etc/docker/certs.d/registry.pure.lab.com:5000
```

- Copy the **ca.crt** from the private Docker Registry server – **registry.pure.lab.com**.

```
[root@sn1-r720-g09-15 ~]# scp -rp root@10.21.236.107:/certs/ca.crt /etc/docker/certs.d/registry.pure.lab.com/:5000/.
```

- Make sure the **/etc/docker/daemon.json** file has the same entries as mentioned in the previous section.
- Restart Docker service

```
[root@sn1-r720-g09-15 ~]# systemctl restart docker
```

- New images can be generated on the build server as part of the CI process and then pushed to the private Docker Registry automatically as part of the build pipeline. For the purpose of this paper, a few Docker images were pulled from the dockerhub library to demonstrate the **docker push** operation.

```

[root@sn1-r720-g09-15 ~]# docker pull debian
Using default tag: latest
latest: Pulling from library/debian

```



```
05d1a5232b46: Pull complete
Digest: sha256:07fe888a6090482fc6e930c1282d1edf67998a39a09a0b339242fbfa2b602fff
Status: Downloaded newer image for debian:latest
```

```
[root@sn1-r720-g09-15 ~]# docker pull centos
Using default tag: latest
latest: Pulling from library/centos
256b176beaff: Pull complete
Digest: sha256:6f6d986d425aeabdc3a02cb61c02abb2e78e57357e92417d6d58332856024faf
Status: Downloaded newer image for centos:latest
```

- Tag the images with the Registry server name – **registry.pure.lab.com** – and port **5000**.

```
[root@sn1-r720-g09-15 ~]# docker tag centos registry.pure.lab.com:5000/centos
[root@sn1-r720-g09-15 ~]# docker tag debian registry.pure.lab.com:5000/Debian
```

- List the tags of the Docker images.

```
[root@sn1-r720-g09-15 ~]# docker images|grep debian
debian                                latest                                f2aae6ff5d89                        2 weeks
ago                                  101MB
registry.pure.lab.com:5000/debian    latest                                f2aae6ff5d89                        2 weeks
ago                                  101MB
```

```
[root@sn1-r720-g09-15 ~]# docker images|grep centos
centos                                latest                                5182e96772bf                        6 weeks
ago                                  200MB
registry.pure.lab.com:5000/centos    latest                                5182e96772bf                        6 weeks
ago                                  200MB
[root@sn1-r720-g09-15 ~]#
```

- Push the newly tagged images to the secure private Registry – **registry.pure.lab.com**.

```
[root@sn1-r720-g09-15 ~]# docker push registry.pure.lab.com:5000/debian
The push refers to repository [registry.pure.lab.com:5000/debian]
b28ef0b6fef8: Pushed
latest: digest: sha256:00c5748eb465a0139063c544de181177da504dfa4e545ac3c0ecd13b7363e70f size: 529
```

```
[root@sn1-r720-g09-15 ~]# docker push registry.pure.lab.com:5000/centos
The push refers to repository [registry.pure.lab.com:5000/centos]
1d31b5806ba4: Pushed
```

latest: digest: sha256:fc2476ccae2a5186313f2d1dadb4a969d6d2d4c6b23fa98b6c7b0a1faad67685 size: 529

- Check the Registry server to confirm the images are now available.

```
[root@sn1-r720-g09-15 ~]# curl --cacert /etc/docker/certs.d/registry.pure.lab.com:5000/ca.crt https://
registry.pure.lab.com:5000/v2/_catalog
{"repositories":["centos","debian","node","wordpress"]}
[root@sn1-r720-g09-15 ~]#
```

- The Registry server – **registry.pure.lab.com** – that mounts the NFS share **docker** from FlashBlade lists all the repositories.

```
[root@sn1-r720-g09-17 registry]# pwd
/var/lib/registry
[root@sn1-r720-g09-17 registry]#
[root@sn1-r720-g09-17 registry]# ls -al
total 4
drwxr-xr-x  1 root root   0 Sep 10 10:41 .
drwxr-xr-x 39 root root 4096 Sep 15 15:10 ..
drwxr-xr-x  1 root root   0 Sep 16 08:16 docker
drwxr-xr-x  1 root root   0 Sep 10 10:41 .fast-remove
drwxr-xr-x  1 root root   0 Sep 10 10:41 .snapshot
[root@sn1-r720-g09-17 registry]#
```

```
[root@sn1-r720-g09-17 registry]# tree -d
.
├── docker
│   └── registry
│       └── v2
│           ├── blobs
│           │   └── sha256
│           │       ├── 00
│           │       │   └── 00c5748eb465a0139063c544de181177da504dfa4e545ac3c0ecd13b7363e70f
│           │       ├── 05
│           │       │   └── 05d1a5232b461a4b35424129580054caa878cd56f100e34282510bd4b4082e4d
│           │       ├── 15
│           │       │   └── 15bc7736db11de5eddd0f13bb1c28ebe5612a4fcf398c7c1077f446abdbfb935
│           │       ├── 23
│           │       │   └── 23d9db872f7e06bc37f2a2c704c042fdd90c872a130ce7ef2272f0f9d03afdbc
│           │       └── 25
```

| | └─ 256b176beaff7815db2a93ee2071621ae88f451bb1e198ca73010ed5bba79b65  
| | └─ 41  
| | └─ 41e689eea0cdaab1280ed184f117171f0897a9353b3bd4cc321f0839d4027511  
| | └─ 46  
| | └─ 46dde23c37b3419122bb597461c1a48bdea1842aaae7dbe728dfa20a9aabe11b  
| | └─ 48  
| | └─ 480d183b9ecf7a67b386cc7d1cf2919efbba591dab9e8c7384bbf323a183686  
| | └─ 4c  
| | └─ 4cb905a2737c60594ee948e4d6e7ae5375b571990c02fef97b1b5825d2275633  
| | └─ 51  
| | └─ 5182e96772bf11f4b912658e265dfe0db8bd314475443b6434ea708784192892  
| | └─ 54  
| | └─ 543b133149a6507f2c2362aa42deda94d023bba156e6b8238c6cb3b71ec15dd3  
| | └─ 65  
| | └─ 65632e89c5f4ef102bcd13b6e86baf954e0b902f688a46961d5f0a36dddfebe  
| | └─ 69  
| | └─ 69a25f7e493029f541fc3c7ac66fdffd5f8c4b9b33346031523d053177bb365  
| | └─ 6e  
| | └─ 6ebaeb0745895220f609d4aa703e4563c39de239a2d00b85bece23a3ca3ac735  
| | └─ 70  
| | └─ 708410509e8f13fd6433384965b4a40f9f7a2863e516d3257e06ec0912247cb8  
| | └─ 81  
| | └─ 8178629708ff26a93a1f3b0cc54217949a6f9f8d30f72f9665d4043f4757b24f  
| | └─ 81bc8dc9a8659a169f8187092eb2037bef64645ea74103b9a7d573ef4fc07496  
| | └─ 8a  
| | └─ 8ac9c275c1a51f9f692adaeba47bb7f8471113cfb984723b21b83e4d0e48ee71  
| | └─ 90  
| | └─ 904e670dd387d970b3d51bc93c54ecaf3e2f1614f1dcc30085dbd001ff6128d5  
| | └─ a0  
| | └─ a06f9e60b0808ffb7b1aa9dd498f894fe6d6663ad37ba09726638677c7970389  
| | └─ a3  
| | └─ a3ed406e3c880fbafac0ae7ab1a889a46f9ef17e86e3efd898158a3241a0518b  
| | └─ ae  
| | └─ ae70e16ef0357e55da00ce408666de68104a907cfdec4fac79eb281ee23ed4b5  
| | └─ b2  
| | └─ b2c40cef4807e3464b2859ebb5e4ac179cfbc253a212ce725f3a5d27388f79fe  
| | └─ b9  
| | └─ b9501b07bdf16f01651117cbfbe7deeed95c4c58331b4b164c8876e30328ec59

```

|   ├── be
|   |   └── be8881be8156e4068e611fe956aba2b9593ebd953be14fb7feea6d0659aa3abe
|   ├── c0
|   |   └── c0cfd94445383406b8a0935e92616ab11a87307e702d4dfaabf2b25542f49a40
|   ├── c4
|   |   ├── c41139f0d4f55571bdfd90615889964aa74ed2878bd8811c906033642e3b9770
|   |   └── c44709db38b42671c256080936a6e578adeb806a580750f56e9ccb2388292176
|   ├── cd
|   |   └── cd75fa32da8fd946b82c0447feac1f3c24330594492e3be74a516b18437d5306
|   ├── d6
|   |   └── d660b1f15b9bfb8142f50b518156f2d364d9642fe05854538b060498e2f7928d
|   ├── e6
|   |   └── e6006cdfa16b487a3a92269f59ecf33b936311fb9934fd4a5b7775b46933fdfe
|   ├── e7
|   |   └── e7428f935583e84197ae834885e62b69922f1ce7e8672a3746295555b3853fc7
|   ├── ed
|   |   └── edde81479afc60a26d95e7f657cceac56aa8a169fd4691b98da919f00e097d3d
|   ├── f1
|   |   └── f180ba12d718e7cecbefe48d0a552bb27932480b35bb0130acf61194c8b4acf8
|   ├── f2
|   |   └── f2aae6ff5d896839bfb8609cb1510bcf36efcb6950683c3bcfb760668b0eefbe
|   ├── f3
|   |   └── f3507e55e5eba49288cb3c8ff469e5a772b31fe8d0b5d2dae06faff4a4d34318
|   ├── f9
|   |   └── f9ecb48a3c125fb88cd60f765a8be522c55cb897a2670721060a98835b4a42b0
|   └── fc
|       └── fc2476ccae2a5186313f2d1dad4a969d6d2d4c6b23fa98b6c7b0a1faad67685

```

└── **repositories**

└── **centos**

└── **\_layers**

└── **sha256**

└── 256b176beaff7815db2a93ee2071621ae88f451bb1e198ca73010ed5bba79b65

└── 5182e96772bf11f4b912658e265dfe0db8bd314475443b6434ea708784192892

└── **\_manifests**

└── **revisions**

└── **sha256**

└──

fc2476ccae2a5186313f2d1dad4a969d6d2d4c6b23fa98b6c7b0a1faad67685

```

| | └─ tags
| |   └─ latest
| |     └─ current
| |       └─ index
| |         └─ sha256
| |           └─
fc2476ccae2a5186313f2d1dadb4a969d6d2d4c6b23fa98b6c7b0a1faad67685
| └─ _uploads
└─ debian
| └─ _layers
| | └─ sha256
| |   └─ 05d1a5232b461a4b35424129580054caa878cd56f100e34282510bd4b4082e4d
| |     └─ f2aae6ff5d896839bfb8609cb1510bcf36efcb6950683c3bcfb760668b0eefbe
| └─ _manifests
| | └─ revisions
| | | └─ sha256
| | |   └─
00c5748eb465a0139063c544de181177da504dfa4e545ac3c0ecd13b7363e70f
| | └─ tags
| |   └─ latest
| |     └─ current
| |       └─ index
| |         └─ sha256
| |           └─
00c5748eb465a0139063c544de181177da504dfa4e545ac3c0ecd13b7363e70f
| └─ _uploads
└─ node
| └─ _layers
| | └─ sha256
| |   └─ 46dde23c37b3419122bb597461c1a48bdea1842aaae7dbe728dfa20a9aabe11b
| |     └─ 480d183b9ecf7a67b386cc7d1cf2919effbaa591dab9e8c7384bbf323a183686
| |       └─ 6ebaeb0745895220f609d4aa703e4563c39de239a2d00b85bece23a3ca3ac735
| |         └─ 8ac9c275c1a51f9f692adaeba47bb7f8471113cfb984723b21b83e4d0e48ee71
| |           └─ c0cfd94445383406b8a0935e92616ab11a87307e702d4dfaabf2b25542f49a40
| |             └─ c44709db38b42671c256080936a6e578adeb806a580750f56e9ccb2388292176
| |               └─ d660b1f15b9bfb8142f50b518156f2d364d9642fe05854538b060498e2f7928d
| |                 └─ e7428f935583e84197ae834885e62b69922f1ce7e8672a374629555b3853fc7
| |                   └─ edde81479afc60a26d95e7f657cceac56aa8a169fd4691b98da919f00e097d3d
| └─ _manifests

```

```

| | | | --- revisions
| | | |   └─ sha256
| | | |     └─ 81bc8dc9a8659a169f8187092eb2037bef64645ea74103b9a7d573ef4fc07496
| | | └─ tags
| | |   └─ latest
| | |     └─ current
| | |       └─ index
| | |         └─ sha256
| | |           └─

```

81bc8dc9a8659a169f8187092eb2037bef64645ea74103b9a7d573ef4fc07496

```

|   └─ _uploads
└─ wordpress
  └─ _layers
    └─ sha256
      └─ 15bc7736db11de5eddd0f13bb1c28ebe5612a4fcf398c7c1077f446abdbdfb935
      └─ 23d9db872f7e06bc37f2a2c704c042fdd90c872a130ce7ef2272f0f9d03afdbc
      └─ 41e689eea0cdaab1280ed184f117171f0897a9353b3bd4cc321f0839d4027511
      └─ 4cb905a2737c60594ee948e4d6e7ae5375b571990c02fef97b1b5825d2275633
      └─ 543b133149a6507f2c2362aa42deda94d023bba156e6b8238c6cb3b71ec15dd3
      └─ 65632e89c5f4ef102bcd13b6e86baf954e0b902f688a46961d5ff0a36dddfebe
      └─ 69a25f7e493029f541fc3c7ac66fdffdd5f8c4b9b33346031523d053177b365
      └─ 8178629708ff26a93a1f3b0cc54217949a6f9f8d30f72f9665d4043f4757b24f
      └─ 904e670dd387d970b3d51bc93c54ecaf3e2f1614f1dcc30085dbd001ff6128d5
      └─ a06f9e60b0808ffb7b1aa9dd498f894fe6d6663ad37ba09726638677c7970389
      └─ a3ed406e3c880fbafac0ae7ab1a889a46f9ef17e86e3efd898158a3241a0518b
      └─ ae70e16ef0357e55da00ce408666de68104a907cfddec4fac79eb281ee23ed4b5
      └─ b2c40cef4807e3464b2859ebb5e4ac179cfbc253a212ce725f3a5d27388f79fe
      └─ b9501b07bdf16f01651117cbf7e7deeed95c4c58331b4b164c8876e30328ec59
      └─ be8881be8156e4068e611fe956aba2b9593ebd953be14fb7feea6d0659aa3abe
      └─ c41139f0d4f55571bdfd90615889964aa74ed2878bd8811c906033642e3b9770
      └─ cd75fa32da8fd946b82c0447feac1f3c24330594492e3be74a516b18437d5306
      └─ e6006cdfa16b487a3a92269f59ecf33b936311fb9934fd4a5b7775b46933fdfe
      └─ f180ba12d718e7cecebebe48d0a552bb27932480b35bb0130acf61194c8b4acf8
      └─ f3507e55e5eba49288cb3c8ff469e5a772b31fe8d0b5d2dae06faff4a4d34318
      └─ f9ecb48a3c125fb88cd60f765a8be522c55cb897a2670721060a98835b4a42b0
    └─ _manifests
      └─ revisions
      └─ sha256

```



```
[root@sn1-r720-g09-23 ~]# docker pull registry.pure.lab.com:5000/centos
Using default tag: latest
latest: Pulling from centos
256b176beaff: Pull complete
Digest: sha256:fc2476ccae2a5186313f2d1dad4a969d6d2d4c6b23fa98b6c7b0a1faad67685
Status: Downloaded newer image for registry.pure.lab.com:5000/centos:latest
```

- Check the new images pulled from the secure Registry server – **registry.pure.lab.com**.

```
[root@sn1-r720-g09-23 ~]# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
registry.pure.lab.com:5000/debian latest       f2aae6ff5d89     2 weeks ago     101MB
registry.pure.lab.com:5000/centos latest       5182e96772bf     6 weeks ago     200MB
[root@sn1-r720-g09-23 ~]#
```

## DOCKER SWARM CLUSTER SETUP

- Configure Docker swarm cluster with three nodes, as illustrated in Figure 8 above. Run the **docker swarm init** command on the **registry.pure.lab.com** node.

```
[root@sn1-r720-g09-17 ~]# docker swarm init --advertise-addr 10.21.236.115
Swarm initialized: current node (4z5leyosculcdvh8rf24wuxnu) is now a manager.
```

To add a worker to this swarm, run the following command:

```
docker swarm join --token SWMTKN-1-2b7ee6dvoveg0e9mntntc6n2fagmm27fg4vzmu7pcpvs1e5nz26-e8r1kjj56v1xkueankxrgj9fu 10.21.236.115:2377
```

To add a manager to this swarm, run **docker swarm join-token manager** and follow the instructions.

- Add two more nodes to the cluster.

```
[root@sn1-r720-g09-19 ~]# docker swarm join --token SWMTKN-1-2b7ee6dvoveg0e9mntntc6n2fagmm27fg4vzmu7pcpvs1e5nz26-e8r1kjj56v1xkueankxrgj9fu 10.21.236.115:2377
This node joined a swarm as a worker.
```

```
[root@sn1-r720-g09-17 ~]# docker node ls
ID          HOSTNAME          STATUS  AVAILABILITY
MANAGER STATUS  ENGINE VERSION
4z5leyosculcdvh8rf24wuxnu * sn1-r720-g09-17.puretec.purestorage.com Ready  Active
Leader      18.06.1-ce
```



```

5d2sovxbiarna0sg1m2epm3b    sn1-r720-g09-19.puretec.purestorage.com    Ready    Active
18.06.1-ce
tzjs9gxodvmd2mm7dud4fvdi    sn1-r720-g09-21.puretec.purestorage.com    Ready    Active
18.06.1-ce

```

```

[root@sn1-r720-g09-17 ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
5ffa4fe40942       bridge             bridge              local
bd82739a9bde       docker_gwbridge    bridge              local
7a78bdc85e9d       host               host                local
28d49m1d0q89       ingress            overlay             swarm
f85b7ef40497       none               null                local

```

## CONCLUSION

Microservices and [containers](#) have transformed the way applications are developed in organizations that follow modern DevOps principles. Private Docker Registries are used and accessed by different tools in data pipelines via a simple URL in the CI/CD process. A private Docker Registry can be configured and consumed as part of a CI pipeline where privacy, better development tools, integration, and cost efficiency for proprietary code and binaries are desired.

As the sole purpose of the Docker Registry is to host and collaborate in disconnected and private environments, NFS on FlashBlade allows not only performance and capacity scaling with a reduced data footprint, but also enables faster data recovery in the event of data loss and disasters. FlashBlade provides a data hub for different workloads within the CI/CD process.

Optimizing source code repositories and parallel builds on FlashBlade is a discussion for a separate paper. Here, we've shown that [FlashBlade](#) has the ability to store, deliver, and scale binary repositories like Docker images in a cost-efficient manner with high data availability and faster data recovery capabilities.

## ABOUT THE AUTHOR

As a Technical Director for DevOps/EDA, Bikash Roy Choudhury is responsible for designing and architecting solutions to address customer business requirements in their transition to agile development and application workflows across industry verticals that include EDA/high tech, financial services, gaming, social media, and web-based development organizations. Bikash has also worked on validating solutions – with Red Hat OpenStack Platform (IaaS), Apprenda (PaaS), Kubernetes/Docker, Jenkins, JFrog Artifactory, Ansible, IBM Private Cloud (PaaS), and Perforce Helix – using RESTful APIs and integrating them with data platforms that provide persistent data storage in private, hybrid, and public clouds. With over 26 years of experience, Bikash has a deep understanding of workloads and workflows that have transitioned from spinning disks to flash over the years, via NFS/blocks and S3 object store, and of automation and data management strategies for end users and business owners.



© 2018 Pure Storage, Inc. All rights reserved.

Pure Storage, the “P” Logo, and FlashBlade are trademarks or registered trademarks of Pure Storage, Inc. in the U.S. and other countries. Docker is a registered trademark of Docker, Inc. in the U.S. and other countries.

The Pure Storage product described in this documentation is distributed under a license agreement and may be used only in accordance with the terms of the agreement. The license agreement restricts its use, copying, distribution, decompilation, and reverse engineering. No part of this documentation may be reproduced in any form by any means without prior written authorization from Pure Storage, Inc. and its licensors, if any.

THE DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. PURE STORAGE SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

ps\_wp34p\_registry-as-a-service-with-flashblade\_itr\_01

SALES@PURESTORAGE.COM | 800-379-PURE | @PURESTORAGE