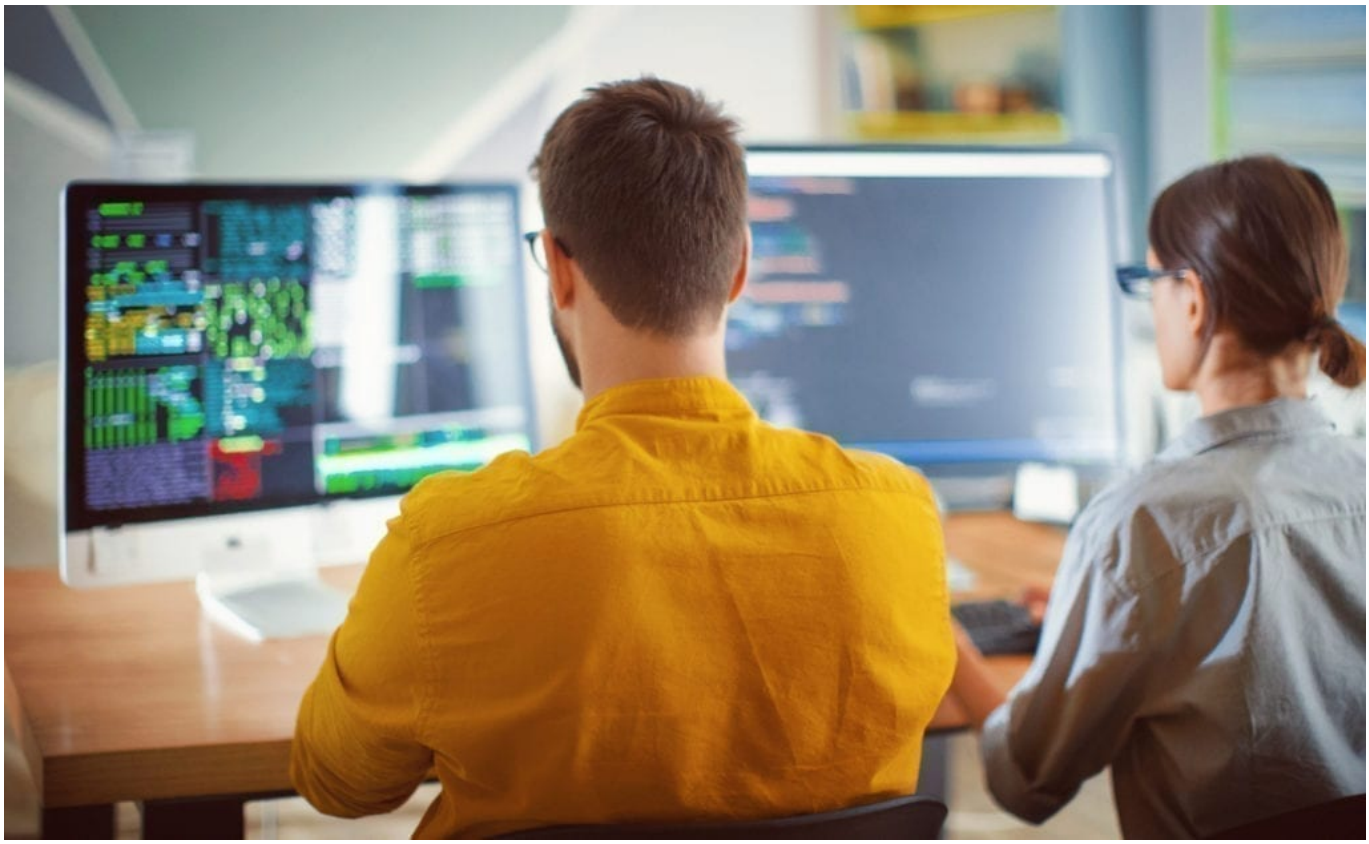
# Storing a Private Docker Registry on FlashBlade™ S3



*This is the second post in a multi-part series by Bikash Roy Choudhury and Emily Watkins, where we discuss how to configure a Kubernetes-based AI Data Hub for data scientists.*

---

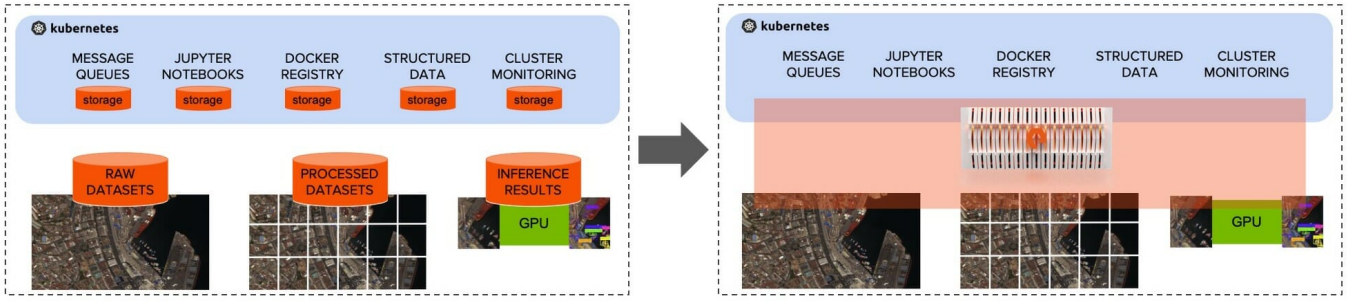
For data science experiments, there's always a new library–or a new version of an existing library–to test, and data scientists are most likely juggling six other libraries they want or need specific versions of. A common solution is to constantly create development environments by saving new Docker images. However, the total number of Docker images owned by a team can balloon quickly.

As teams work toward repeatable, scalable tests, it becomes increasingly important to manage Docker images centrally. Doing so simplifies sharing between team members as well as across compute nodes, like GPU servers.

You may want to set up a private Docker registry for environments that do not have internet connection or do not use DockerHub.

It can also help simplify your Artificial Intelligence (AI) deployment if you store your Docker registry on the same machine where your data scientist's training datasets are saved. Many components of an AI data hub require storage. Storing more components of your AI platform on centralized storage, such as FlashBlade S3, results in simpler management and operations. It also lowers overall costs by avoiding silos of unused

capacity and by sharing hardware across performance-sensitive and colder jobs.




In this blog post, we describe how to configure a private Docker registry on FlashBlade S3.

# Configuration

Note: In this post, we set up an "insecure" Private Docker Registry since our cluster is on a tightly-controlled network. There are three options for securing a registry:

1. Use HTTP ("insecure-registry" mode) – method followed bellow
2. Issue a self-signed certificate
3. Obtain a TLS certificate from a 3rd-party certificate authority – official recommendation from Docker

Each of these options require some additional configuration. An insecure registry is a quick way to configure a registry in a lab environment that's on a secure private network.

At a high level, the configuration steps include: setting up an S3 bucket on FlashBlade, configuring the node that hosts the registry server, and launching the server. We'll also provide example usage of the registry.

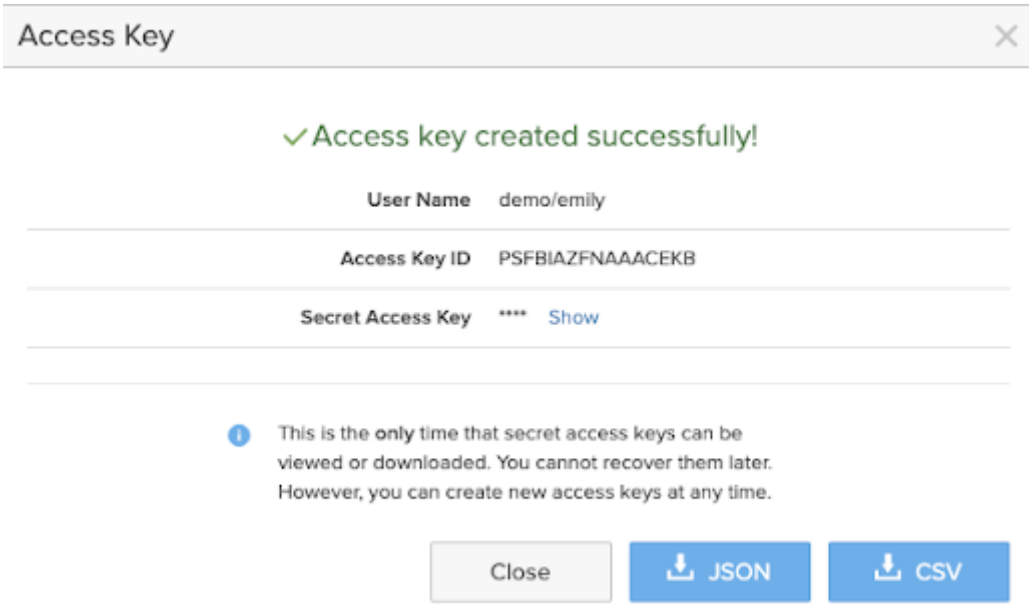
# Configure FlashBlade

First, we'll discuss how to set up an S3 bucket on FlashBlade object store to be the backend for the registry.

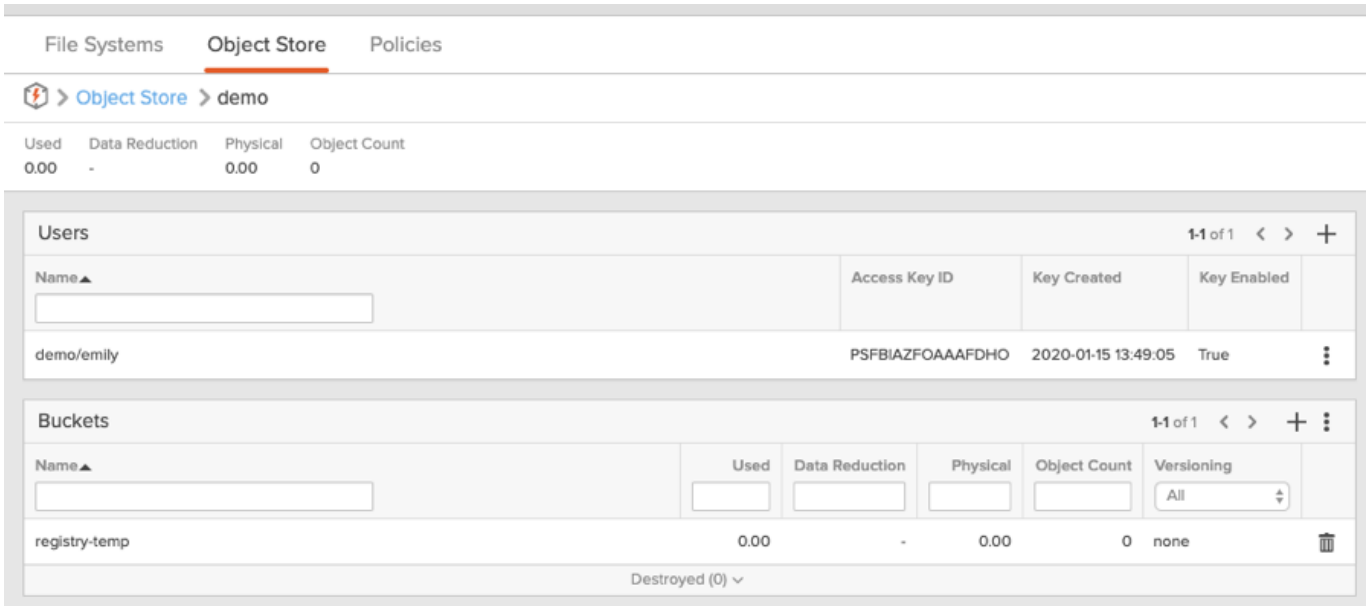
*Here, we'll demonstrate how to do the setup in the FlashBlade GUI, but our colleague Joshua Robinson wrote a python script that automates the creation of S3 users, keys, and buckets.*

In the FlashBlade GUI, navigate to Storage > Object Store and create a new Account. You can then add a "registry" bucket and a new user for this Account. Then create an Access Key for the user.

Note: when you create the Access Key, you will be provided with a Secret Access Key, which is only accessible at the time of Access Key creation. Considering that the Secret Access Key will be needed to configure the registry, it is recommended that it be downloaded and saved as a JSON or CSV file.




Creating a new user for your S3 Account provides an Access Key and Secret Access Key that you will need later in this setup.




The FlashBlade bucket is now ready to use.

# Configure Registry Server Node

Before we can launch the registry, we need to edit three files **on the machine that's going to run our registry server.**

- /etc/docker/daemon.json
- /etc/default/docker
- /etc/docker/config.yml

In this demo, our registry server is 10.61.169.83 (running Ubuntu 18.04).

**Configure /etc/docker/daemon.json**

If the file does not exist, create it and add the insecure registry.

- Use the IP address of the current machine
- It's common to use port 5000 for Docker registries, but you can customize the published port if desired.

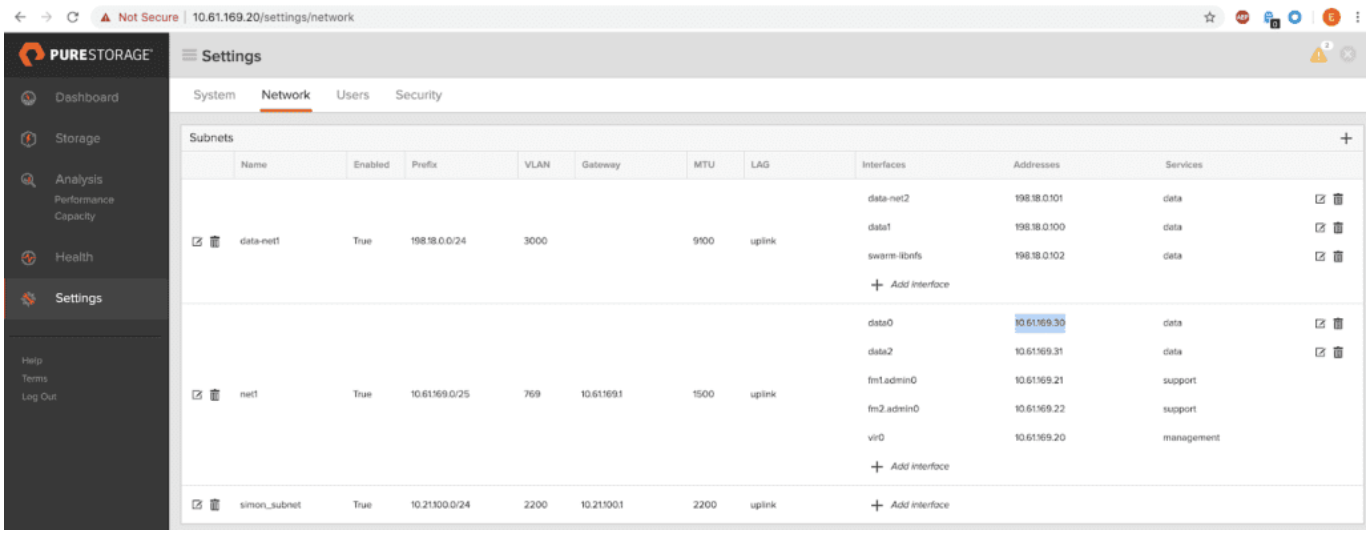
[crayon-6515baa2a1f17069755242/]
The registry can be seen in docker info now.
[crayon-6515baa2a1f21193669615/]
**Add the registry information to /etc/default/docker**
[crayon-6515baa2a1f23079556397/]
**Restart the Docker service**

systemctl restart docker

Confirm that Docker is up again after restart:
[crayon-6515baa2a1f24083896749/]
**Add S3 info to /etc/docker/config.yml**

We need the following info from our S3 configuration on FlashBlade: S3 user's access key and secret key, bucket name, and FlashBlade data VIP.

The FlashBlade data VIP can be found under Settings -> Network.




We use this information to fill in the "s3" fields in the Docker config file:
[crayon-6515baa2a1f26801361333/]

- Deploy the registry server with this command:

"–name registry": the server will run in a container named "registry"

"registry:2": the command downloads this image from Docker to use for the registry

[crayon-6515baa2a1f27843084704/]
The following confirms that the server is running:
[crayon-6515baa2a1f28737197504/]

(You can also check the registry containers' logs with docker logs registry).

### Configure all other nodes in the cluster.

Once that is successful, update all nodes participating in the Kubernetes cluster with the registry's information, and then restart their Docker daemon.

In this example, the registry known as "registry-ai-projects.dev.purestorage.com:5000" is being added to our DGX-1.

Here's what /etc/docker/daemon.json looks like on one of our DGX-1s:
[crayon-6515baa2a1f29093159799/]
Whenever you edit the docker daemon.json, you should restart Docker so the changes take effect.

Confirm that docker is "active (running)" via systemctl status docker.

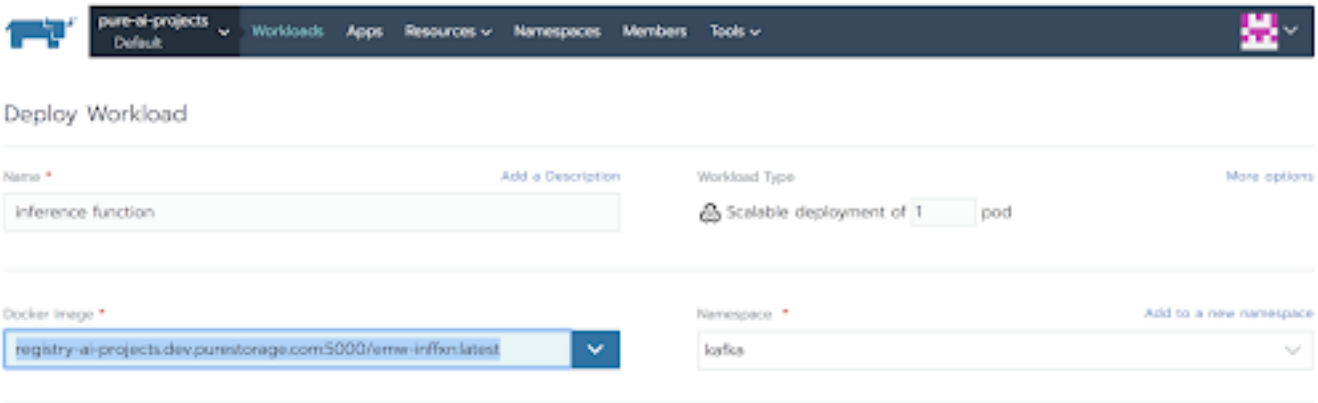
## Use the registry

Example of pushing an image from a node in the cluster (here, a DGX-1) to the registry:
[crayon-6515baa2a1f2a810897993/]
By default, your registry will present a catalog of its images (called "repositories") at http://<registryIP>:5000/v2/_catalog

Here's ours:


As you can see, the "python" image is shown here (tags are hidden in this view). Success!

Images in the registry are available when launching new workloads in the cluster. There's complete continuity for the applications, which can utilize the Docker images regardless of whether they're saved as files or objects.



In the Rancher GUI for a cluster: Workloads -> Deploy -> select an image from the registry for the Docker Image field

## Optional Modifications

Here are some of our favorite registry enhancements.

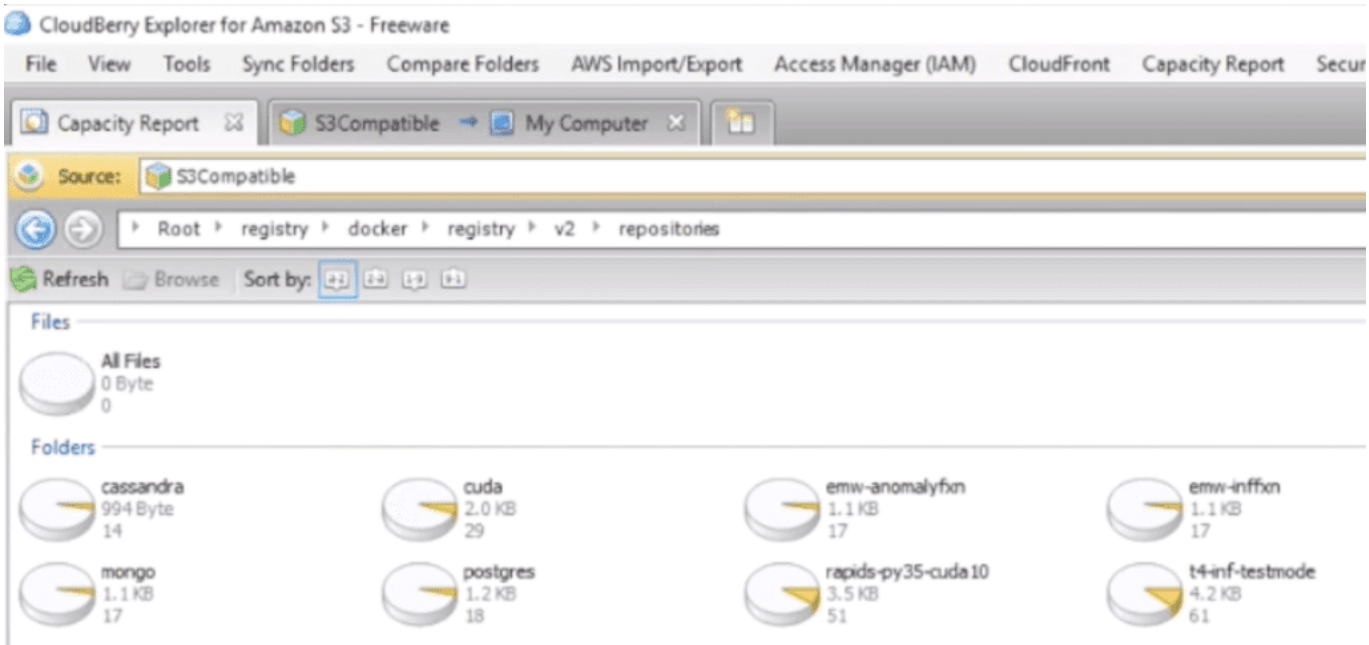
**Modification 1:** You may notice we navigated to "registry-ai-projects.dev.purestorage.com:5000..." instead of "10.61.169.83:5000...". The /etc/hosts file on the node hosting the registry server can be used to map to map the node's IP address to a new domain name. Users can now push images to the registry at <domain-name>:5000.

**Modification 2:** As previously mentioned, it's possible to make your registry more secure by including a self-signed certificate. You can find example configuration steps in Pure's Registry-as-a-Service white paper.

**Modification 3:** In this example, we configured a Docker registry outside Kubernetes so that the registry can be shared across multiple clusters. Alternatively, it's possible to launch the registry as a Kubernetes pod and use an ingress service to forward traffic to the registry. Here's an example.

**Modification 4:** If your team is highly visual, you can use an object storage explorer tools like MSP360 (fka. CloudBerry) to get a file-system like view of the registry bucket.

Here's a snippet of ours:




## Conclusion

Setting up a private Docker Registry on Flashblade S3 allows team members to share environments and better collaborate on various AI projects.

As the storage for an AI Data Hub, FlashBlade can provide scalable, performant storage for not only the hot tier of datasets that data scientists are training on, but also for ancillary components like a Docker registry.

Stay tuned for the next installment in our blog series on how to configure a Kubernetes-based AI Data Hub for data scientists: *Hosting Jupyter-as-a service on FlashBlade, followed by:*

- Scraping FlashBlade metrics using a Prometheus exporter
- Visualizing Prometheus data with Grafana dashboard for FlashBlade
- Automating an inference pipeline in a Kubernetes Cluster
- Tuning networking configuration of a Kubernetes-based AI Data Hub
- Integrating Pure RapidFile Toolkit into Jupyter notebooks