Technical Validation

Red Hat OpenShift Container Storage

Simplifying Persistent Container Storage for the Open Hybrid Cloud

By Kerry Dolan, Senior IT Validation Analyst

September 2020

This ESG Technical Validation was commissioned by Red Hat and is distributed under license from ESG.

Contents

ESG Technical Validations

The goal of ESG Technical Validations is to educate IT professionals about information technology solutions for companies of all types and sizes. ESG Technical Validations are not meant to replace the evaluation process that should be conducted before making purchasing decisions, but rather to provide insight into these emerging technologies. Our objectives are to explore some of the more valuable features and functions of IT solutions, show how they can be used to solve real customer problems, and identify any areas needing improvement. The ESG Validation Team's expert third-party perspective is based on our own hands-on testing as well as on interviews with customers who use these products in production environments.

Introduction

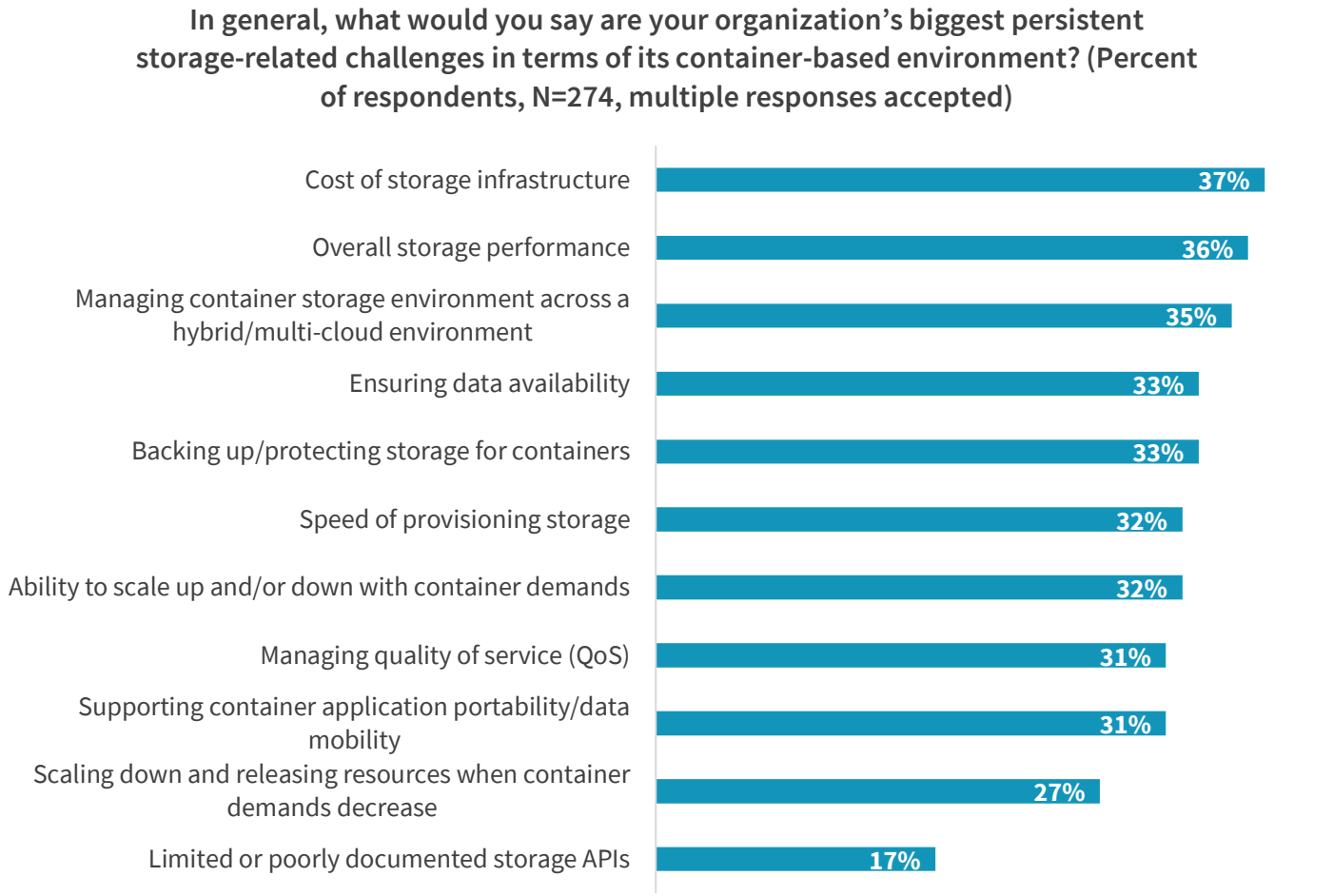
This ESG Technical Validation documents remote testing of Red Hat OpenShift Container Storage with a focus on the ease of use and breadth of data services.

Background

Containers have become an important part of data center modernization. They simplify building, packaging, and deploying applications, and are hardware agnostic and designed for agility—they can run on physical, virtual, or cloud infrastructure and can be moved around as needed. According to recent ESG research, 74% of respondents are currently using containers for production applications or plan to in the next 12 months, and 54% of them have or will have container-based applications deployed in a combination of public cloud platforms and private data centers.[1]

As containers are increasingly used for stateful applications, they need persistent storage. However, storage solutions designed for legacy and virtualized applications don't work well for containers, limiting agility, adding sprawling storage silos, and increasing complexity. According to the same research, among the top persistent storage challenge of containers are managing container storage across a hybrid/multi-cloud environment, speed of provisioning storage, scaling up and down, and data services such as availability, protection, and QoS.

Figure 1. Top Persistent Storage Challenges for Containers



Source: Enterprise Strategy Group

[1] Source: ESG Research Report, *Data Storage Trends in an Increasingly Hybrid Cloud World*, March 2020.

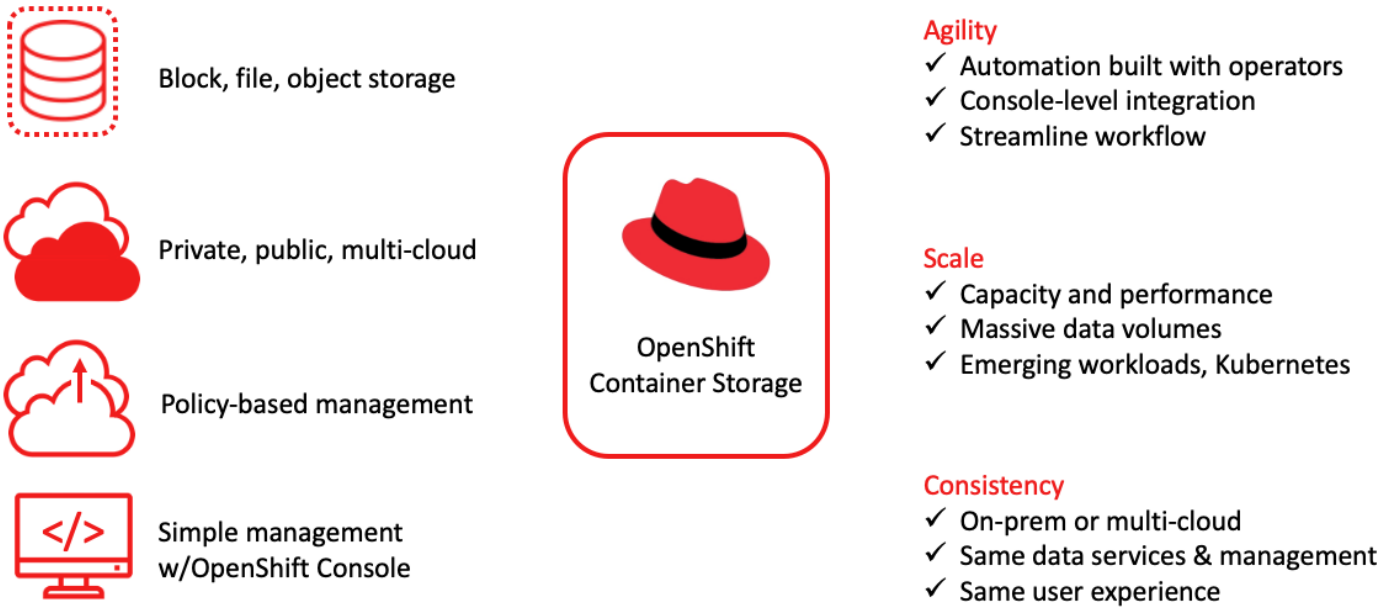
Red Hat OpenShift Container Storage

Red Hat OpenShift Container Storage is a software-defined storage solution that is fully integrated with Red Hat OpenShift. OpenShift Container Storage provides complete data services for the open, hybrid cloud. It runs on-premises, in virtual machines, or in multiple public clouds, including Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). It offers dynamic, stateful, highly available, container-native storage with a consistent experience across infrastructures and storage types—block, file, and object storage. OpenShift Container Storage's rapid, flexible deployment, scalability, seamless upgrades, and automation reduce the complexity that storage can add to application development processes and increases agility to support modern applications such as continuous integration/continuous deployment (CD/CD) pipelines, structured databases, unstructured data lakes, and data analytics and AI/ML applications. It is highly scalable, delivering hundreds of petabytes of storage while maintaining orchestration and automation.

OpenShift Container Storage gives developers the ability to not only access data quickly and easily, but also to provision and manage application storage needs without involving infrastructure teams. By providing API-driven storage with the click of a button, it reduces delays and increases productivity. Developers and data engineers can access all the data services they need, when they need them, while remaining focused on their own jobs, not worrying about where or how data is stored. In addition, the ability to virtualize the S3 namespace enables multi-cloud data services. For example, organizations can set up an on-premises S3 endpoint using OpenShift Container Storage that points to an S3 bucket in AWS or Azure. The user writes to the OpenShift Container Storage endpoint without knowing where the data is stored or having to manage it. This gives organizations the ability to change or add public cloud services without having to re-code.

Installation is easy and fast, and OpenShift Container Storage is integrated with Red Hat OpenShift Container Platform, enabling developers and data scientists to monitor and manage storage in the same interface as application resources using the Red Hat OpenShift administrator console. Developers can create persistent volumes attached to multiple pods in parallel, add storage when needed, and scale automatically. Automation is delivered through operators that are accessible on the Operator Hub and Red Hat Marketplace.

Figure 2. OpenShift Container Storage



Source: Enterprise Strategy Group

Recent feature additions include:

- Expansion of persistent volume claims (PVCs).
- Disconnected mode and proxy mode.
- External storage cluster connectivity. This enables a cluster to connect to external Red Hat Ceph storage, as well as the standard internal mode, where the storage cluster runs inside OpenShift Container Platform.

ESG Technical Validation

ESG viewed remote demos of OpenShift Container Storage that showed the ease of installation and management, as well as the consistent experience across infrastructure and storage types. The test cluster was created in the AWS cloud US East 2 region, using three m5.2xlarge master nodes and three m5.4xlarge worker nodes.

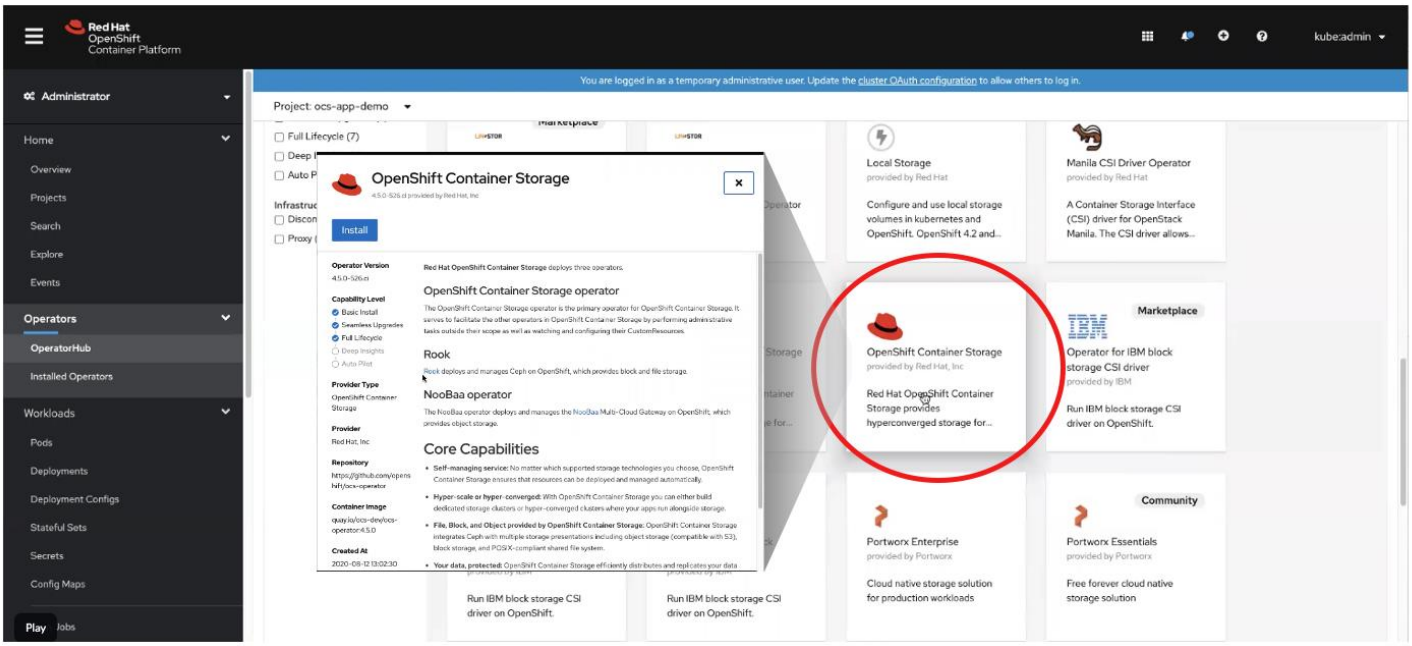
Installation

Speed of installation is a key feature of OpenShift Container Storage. It enables users to get fast access to the type of storage they need for any project, without the typical storage requisition delays. "Operators," available from the OperatorHub in the OpenShift Container Platform GUI, are used to automate installation and management tasks, freeing administrators from having to deal with storage requests. The OpenShift Container Storage Operator is a certified operator, but community-developed operators are also available.

ESG Testing

To begin installation, we navigated to the ***Storage*** category on the ***OperatorHub***, selected ***OpenShift Container Storage***, and clicked ***Install*** (see Figure 3). This deployed the OpenShift Container Storage operator with the ability to deploy and manage file storage, block storage, and a Multi Cloud Gateway to provide object storage.

Figure 3. OpenShift Container Storage Installation



Source: Enterprise Strategy Group

Next, we were shown an ***Update Channel*** listing several OpenShift Container Storage versions, where we selected version 4.5 and subscribed to this channel for automatic updates. Updates are executed over the air, similar to how mobile phones are updated automatically. We selected a specific namespace on the cluster, *openshift-storage*, selected the automatic approval strategy, and clicked ***Install***. This launched a catalog-service version, which then tracked the version of the components being installed. Once the operator was installed, we viewed tiles for three APIs now available:

- ***Storage Cluster***, representing all the storage resources needed to store data.
- ***Backing Store***, which enables creation of storage targets. These are used by the Multi Cloud Gateway to create custom Bucket Classes (see below) along with custom protection policies. Multi Cloud Gateway Backing Stores can be configured to use local storage via PVCs or object storage (e.g., AWS S3, S3-compatible, and Azure blob).
- ***Bucket Class***, which defines storage policies such as tiering, mirroring, and spreading that are delivered by combinations of Backing Stores. Users select a Bucket Class based on the storage features they need.

Organizations can have multiple Backing Stores. For example, one Backing Store may be configured to keep data in a local pool on the OpenShift Container Platform cluster, while another might store data in an S3-compatible object store. Administrators create Bucket Classes with which developers interact, simplifying the process for them. For example, one developer might select the Bucket Class offering a single, local copy for temporary data, while another might use a different Bucket Class designed to mirror data between S3 and an Azure blob.

Create Storage Cluster

Next, from the ***Operator Details*** page we deployed the Storage Cluster by clicking ***Create Instance***. We selected internal mode, which runs the cluster inside the OpenShift Container Platform; next, we selected three nodes to serve as Availability Zones. Data is replicated to all three nodes to ensure high availability (see Figure 4). Should one node fail and lose all the operating pods, the cluster would still function, and applications would still run on the other nodes.

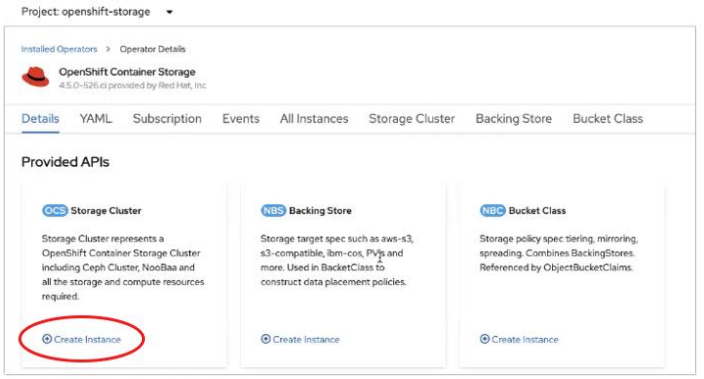
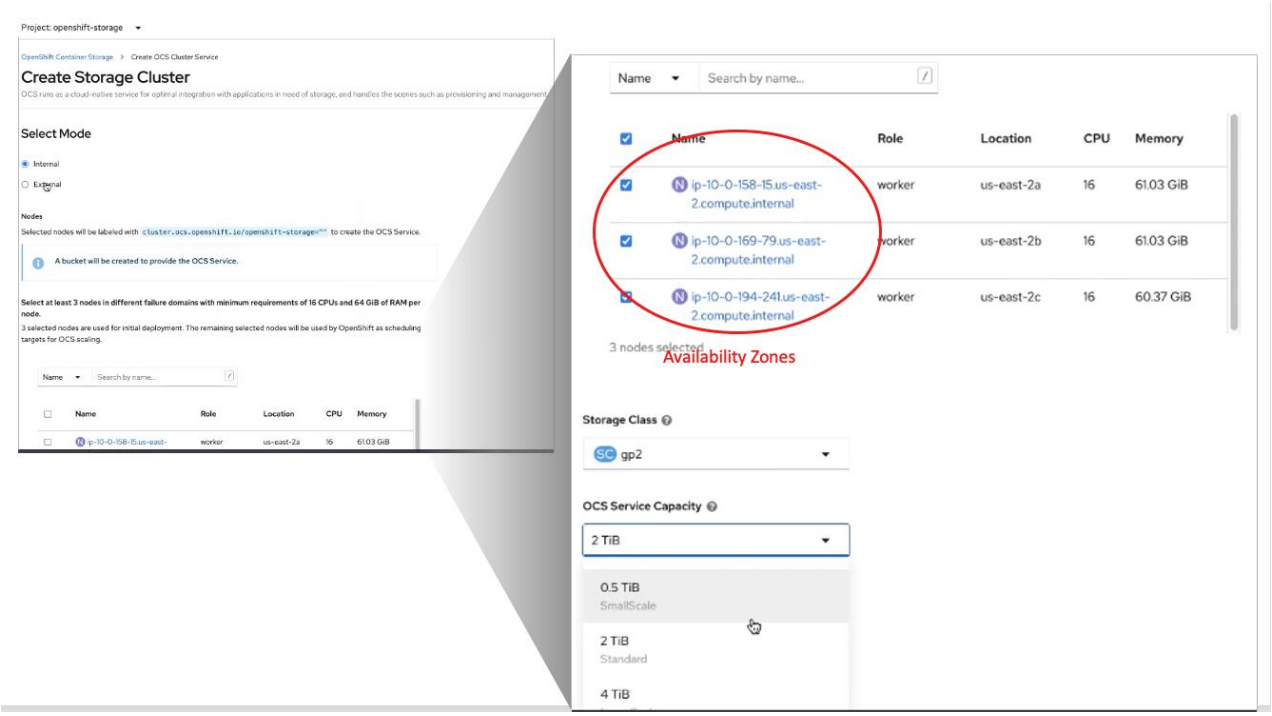


Figure 4. Create Storage Cluster



Source: Enterprise Strategy Group

ESG selected the gp2 storage class (the default Amazon Elastic Block Storage (EBS) general purpose SSD storage) to provision volumes. We selected the 2TiB disk size from the ***OpenShift Container Storage Service Capacity*** drop-down menu; larger and smaller disk sizes are also available. We clicked ***Create*** to start deploying the cluster. This took only a few minutes, during which we were able to view progress, such as creating and starting up containers and pods, through various dashboard tabs. This information is available for administrators who may be interested, but not necessary for users who may not.

Cluster Created

In less than 10 minutes, the cluster was created and ready to use. When the cluster is created, OpenShift Container Storage creates custom dashboards; users can view their persistent storage and object services on those tabs on the ***Overview*** page. The ***Persistent Storage*** tab delivers and tracks use of RWO and RWX persistent volume claims:

- RWO – Read/write once. These volumes can be mounted for read and write from a single node, such as for databases or low-latency application requirements.
- RWX – Read/write multiple times, for data sharing. The volumes can be mounted for read and write from multiple nodes, such as for traditional file workloads.

To support the object service, the OpenShift Container Storage Operator deploys a Multi Cloud Gateway that provides deduplication and encryption to deliver on different storage policies. That database and the Backing Store must be initialized before the service is available, and a test bucket is created to validate connectivity.

Why This Matters

Developers need persistent storage for many container applications, but it can take a long time to request and deploy it. Accessing different types of storage for different types of data and infrastructure causes complications and delays.

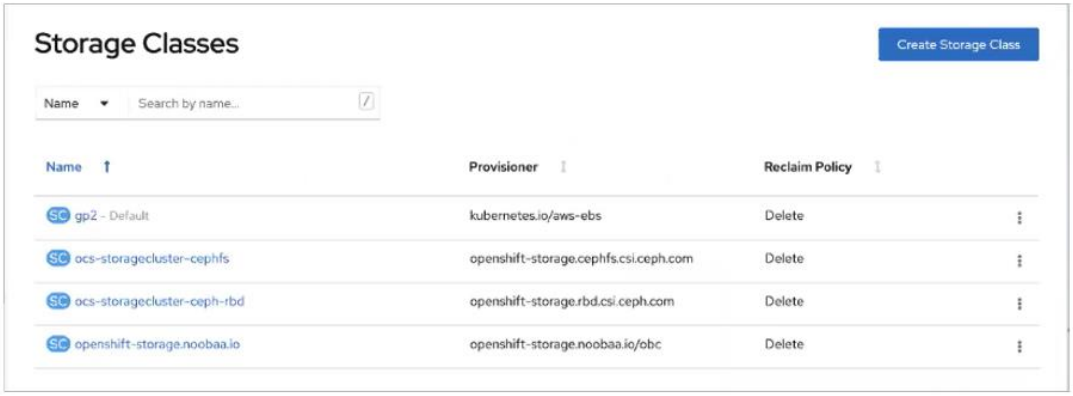
ESG validated the ease and speed of installing Red Hat OpenShift Container Storage. In less than 10 minutes, from a single pane of glass, we installed OpenShift Container Storage and created an AWS cloud cluster with data resiliency across multiple AWS availability zones that would be updated automatically and could be used for file, block, or object storage. This contrasts dramatically with the traditional method of setting up individual storage environments to support different data types, configuring different storage for cloud, bare metal, or VMware environments, and individually configuring the data services required for protection and availability. OpenShift Container Storage can help developers be more autonomous and productive by getting the data services they need without distracting their focus to storage tasks.

Management

With OpenShift Container Storage, administrators, developers, and other users have a simple, graphical way to get the storage and data services they need. These capabilities ensure greater productivity for container-based application development and usage. Users who are more comfortable can continue to execute tasks using the CLI, but the GUI-based management extends storage management capabilities to more users. Definitions and instructions are included throughout the GUI.

ESG Testing

In addition to exploring the GUI, ESG tested several container storage management tasks including cluster expansion, use of templates, and accessing object storage.



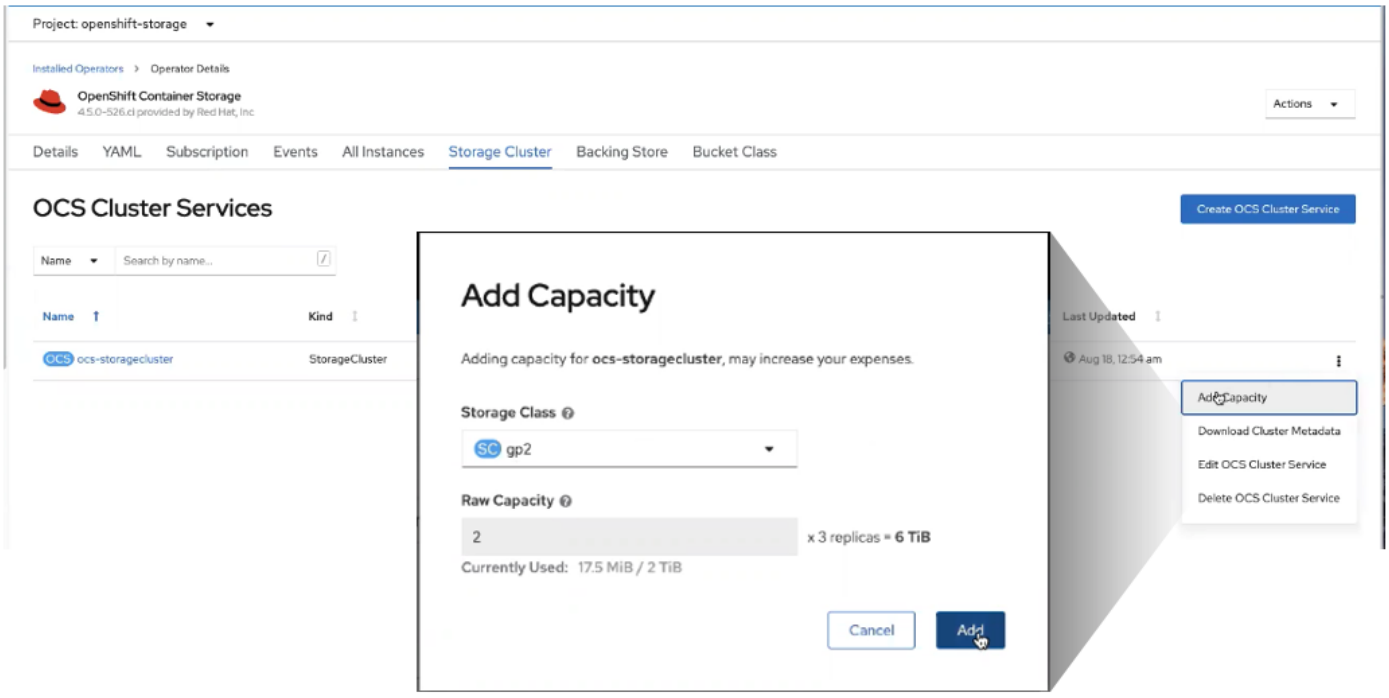
All tasks can be done in the GUI, and details are available for every project, storage class, etc. Under the ***Storage*** tab in the left nav, we could view Persistent Volumes (PVs), Persistent Volume Claims, Storage Classes, Object Buckets, and Object Bucket Claims; in each section, we could search with filters. The ***Storage Classes*** page showed each storage class with its provisioner: the default gp2, provisioned with AWS EBS and Kubernetes, plus typical file, block, and object storage classes. The *ocs-storagecluster-cephfs* provided file data access for both RWO and RWX; *ocs-storagecluster-ceph-rbd* provided block-only RWO PVCs, for which OpenShift Container Storage creates a filesystem right away so it can be consumed by pods. The *openshift-storage-noobaa.io* class is for object storage. Administrators can simply tell developers which class will provide which type of storage so they can provision on their own.

The GUI includes alerts to warn users when resources may run out so that they can be expanded. Storage metrics are also available and can be pulled into dashboards or graphs as needed.

Expand Cluster

Our first task was to expand the cluster to add storage capacity. OpenShift Container Storage enabled us to dynamically allocate new volumes from the AWS gp2 storage class without interrupting applications. This took less than 2 minutes with just a few mouse clicks. To configure additional storage, we returned to the ***Installed Operators*** page where we could find the operator and clicked on the ***Storage Cluster*** API on the right side of the page. This opened the ***OpenShift Container Storage Clusters Services*** page, from which we clicked the drop-down box next to our cluster and selected ***Add Capacity*** (see Figure 5). The disk size defaulted to the 2TiB disk initially configured, showing 6 TiB total to configure the three availability zones, and we simply clicked ***Add***.

Figure 5. Expand Cluster



Source: Enterprise Strategy Group

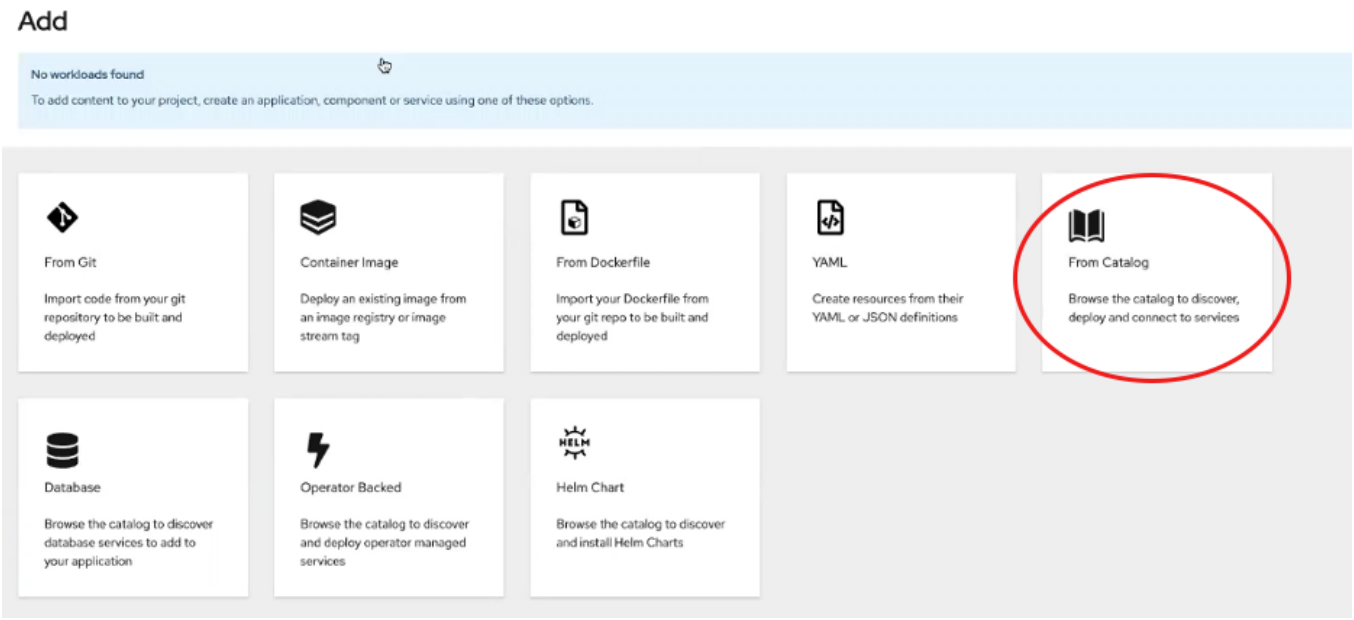
Returning to the ***Cluster Overview*** tab, we noted an alert that the storage was degraded; similarly, in the Persistent Storage tab, the Data Resiliency status changed to ***Progressing***. This occurred because the storage already in use needed to be reallocated across the expanded cluster. OpenShift Container Storage moves the data around in the background after the expansion. Once this was complete, the alerts returned to green, and the cluster showed 3.4 TiB usable capacity.

ESG also demonstrated expansion of a PVC, an equally simple process of using a drop-down menu and selecting the volume size. Once the volume was expanded in the cluster, the file system was resized automatically on top of that volume.

Adding Workloads with Templates

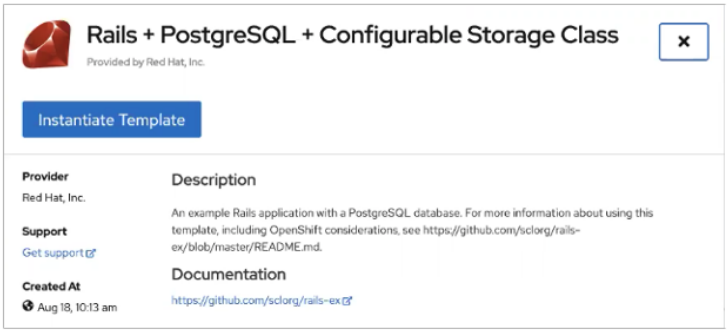
To add content to a project or create an application, users simply click on the appropriate tile and follow instructions. For this cluster, workloads could be added from Git repositories, container images, docker files, or Helm charts; created from YAML or JSON definitions; or selected from catalogs.

Figure 6. Adding Workloads



Source: Enterprise Strategy Group

Using Templates



ESG clicked on the ***From Catalog*** tile to access the developer catalog, selected ***Template***, and searched for "Rails," which brought up tiles for three templates. By clicking on a tile, we selected the template that would create a PostgreSQL database with Rails as the front-end and include configurable storage. Once we clicked ***Instantiate Template***, we could configure the namespace, memory limit, volume capacity, and storage class from which to allocate the volume. This template was preset for block storage for a database, using an RWO PVC, but we could have selected a different class. Administrators have the option to make the storage class non-configurable. We retained all the other database and Rails defaults and clicked ***Create***; the short process of creating the application could be viewed from the ***Workloads*** page. Under the ***Persistent Volume Claims*** page, we could see the correct storage class being provisioned for the application.

This process makes it easy for developers to launch applications with the right storage with little effort. If developers want to compare application performance or experiment with different storage classes, OpenShift Container Storage is set up to make that simple as well.

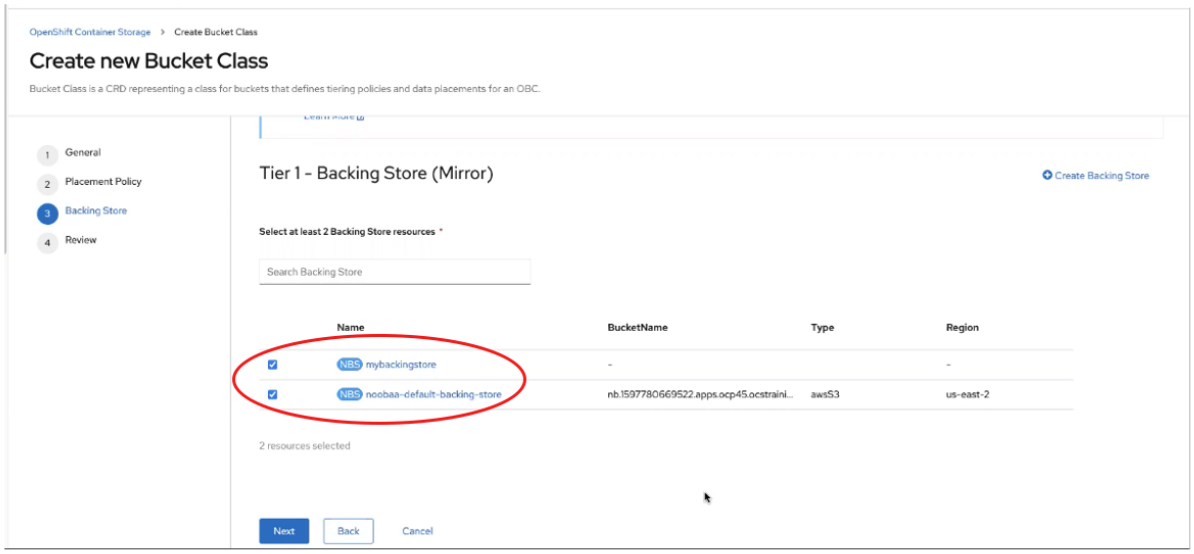
Object Bucket Claim

OpenShift Container Storage provides a range of storage options from a single pane of glass, all with the same user experience. To demonstrate object storage, we created a custom Backing Store with a custom Bucket Class configured with a data mirroring policy between an AWS S3 Backing Store and a local storage-backed Backing Store; these tasks demonstrate how easy it is to both create and consume OpenShift Container Storage.

From the ***Installed Operators*** page, we clicked on ***Backing Store/Create Backing Store***, named the new one *mybackingstore*, and selected ***PVC*** to create a local Backing Store. We used the same 2TiB disks as before, configured a 250GB volume size, and selected the *ocs-storagecluster-ceph-rbd* storage class. Note that we could also view the Multi Cloud Gateway using S3 that was created by default when we started our cluster in AWS.

Next, we clicked on ***Bucket Class/Create Bucket Class***, named it *mybucketclass*, chose the Tier-1 Mirror policy to duplicate data across resources for high availability, and clicked next. On the next screen, we chose multiple Backing Stores on which to create the mirrored copies; we chose both the local *mybackingstore* and the default cloud Backing Store on AWS S3 (*noobaa-default-backing-store*). We clicked next, reviewed the settings, and clicked ***Create Bucket Class***. Then both showed up on the Bucket Class page. Creating the Bucket Class configures the S3 environment as desired, including the AWS region, access key, and secret key. The user doesn't have to deal with any of this back-end AWS configuration; it is all done automatically.

Figure 7. Creating a Bucket Class



Source: Enterprise Strategy Group

Next, we returned to the CLI and configured a single batch job (*webinar3*) to upload data as defined by the Bucket Class we created. This workload was initiated using a YAML file. We launched the job from the CLI, and as it was creating the container, we could view the running pods and all the details of the project from ***Workloads/Pods/webinar3***, including memory and CPU usage, the YAML file, the container environment with all the variables we set, logs, and events. The logs and events screens showed the batch job running, using the Backing Store we created. No volume was attached to the pod because it was using the RESTful S3 protocol—the application sends a request to the HTTP endpoint, which does the job. Back on the ***Overview*** page, we could see the persistent storage and object service capacity breakdowns of the projects running.

Why This Matters

Anything that helps developers work faster will accelerate time to market and time to revenue. Faster application development and faster data insights can increase profitability.

ESG validated how easy it is to create and expand a storage cluster with the right data services. Storage tasks are simplified in the GUI, including the ability to create workloads from preset templates or from typical development resources such as YAML files and Git repositories. It was simple and fast to create the storage resources and policies required for object storage as well. All developers have to do is select a storage class, and OpenShift Container Storage does the provisioning, attaching, and mounting. In addition, with just a few clicks, we were able to create multiple replicas across local and cloud platforms. The powerful combination of GUI and CLI that is easily scriptable makes OpenShift Container Storage simple for developers to use, hiding the complications of persistent storage and allowing them to easily consume what they need.

The Bigger Truth

Containers offer a flexible way to build and deploy workloads and applications. They can operate on physical, virtual, or cloud infrastructure, are lighter weight than virtual machines, and are extremely portable. Container-based applications and microservices architectures orchestrated with Kubernetes need storage and data services that can adapt in the same way; without that, storage becomes a hindrance.

OpenShift Container Storage provides developers, data scientists, and other users with all the storage and data services their applications need—persistence, high availability, and protection—without the complexity of dealing with it. They simply select the storage class desired, and the storage provisioner takes care of the rest. A key feature is that the experience is the same regardless of what infrastructure you are using (on-premises or in the cloud, and bare metal or in VMs) or what type of storage you need (block, file, or object). It's the same experience for administrators or for users, and for test/dev or for production applications.

While performance and scale are key features of OpenShift Container Storage, this report was focused on usability. All installation and management functions are available in the GUI, making it easy for users without storage management skills to get what they need when they need it.

ESG validated the following results:

- 10-minute, GUI-based installation.
- Integration with OpenShift Container Platform.
- Easy access to block, file, or object storage resources from any infrastructure.
- Data services including persistence, high availability, and data protection.
- Simple management including capacity expansion.
- Consistent experience across infrastructures and data types.

Some organizations have struggled to use containers to avoid adding complexity. OpenShift Container Storage can solve that problem, providing persistent, container-native storage and data services that are simple to install and use. ESG believes that it is well worth checking out OpenShift Container Storage to see if it can add value to your organization.