Container repositories provide several key benefits, including:

- **Collaboration**—organizations can use private repositories to share their own container images with internal teams, and public repositories to share images with an external community. This helps accelerate development pipelines and ensure standards across teams and projects, while maintaining privacy for more sensitive software assets.
- **Convenience**—a container repository makes it easy for developers to access their own images, images created by their colleagues, and a large variety of images created by the cloud native community.
- **Integration**—when orchestrating containers in production with platforms like Docker and Kubernetes, it is possible to pull images from the repository and deploy them in an automated way that is conducive to DevOps methodologies.

Deploying a private repository in an organization can provide many benefits, including:

- **Performance**—large images can cause performance and latency issues when pulled over the Internet. Instead, images can be pulled from a private repository deployed on-premises within an organization, to facilitate development workflows.
- **Availability**—when pulling images over the Internet from a remote repository, this service becomes an essential part of an organization's cloud native application ecosystem. If the repository is compromised or goes down, the images are not available during that time. Private container repositories give organizations greater flexibility and control over the availability and accessibility of the container image repository.
- **Security**—images retrieved from public or external repositories may contain security risks (e.g., vulnerabilities, malware), may have been tampered with, or may have been created with a malicious intent. Using a private container repository makes it possible to control security risk exposure to unknown or untrusted assets and ensure that any remediation of risks is reflected in the images stored in the container repository.

Top 3 Container Registry Services

A container repository is a software component used to store and access container images. A container registry is a collection of container repositories, including mechanisms like authentication, authorization, and storage management. Below we cover the top container registry services you can use to access existing container repositories or create your own.

Docker Hub

Docker Hub is the world's largest public container image repository, maintained by the makers of the Docker container engine. Key features include:

- **Repositories** – allow developers and DevOps to pull container images or push new images.
- **Organizations** – let teams set up private repos accessible only to designated workgroups.
- **Official Images** – provide access to high-quality, official Docker images.
- **Publisher Images** – provide a range of verified container images offered by external vendors.
- **Builds** – integrate with GitHub and BitBucket, letting developers and DevOps to automatically build images and push them to Docker Hub.
- **Webhooks** – let developers and DevOps run pre-determined, automated actions after an image is pushed to a repository.

Amazon Elastic Container Registry (ECR)

Amazon ECR is a managed service that provides a secure, scalable container registry, which supports both public and private image repositories, and provides granular access permissions using Amazon Identity and Access Management (IAM). Key features of Amazon ECR include:

- **Public Gallery** – a portal that lists public image repositories hosted on Amazon ECR.

- **Registry** – each Amazon account can create a public registry, which can contain public image repositories.
- **Authorization** – requires clients to authenticate before being permitted to push images to public registries. Amazon ECR supports both anonymous and authorized calls for image pulls.
- **Repository** – the ECR entity that stores Docker images, and Open Container Initiative (OCI) compatible artifacts.
- **Repository policy** – centralizes control over repository access and the specific actions permitted. This can be applied to users or roles across the organization.
- **Images** – container images within Amazon ECR can be used in a variety of ways, such as in a local development environment, to create Amazon ECS tasks, or with Amazon EKS pods.

Azure Container Registry (ACR)

ACR is a managed Docker registry service in the Azure cloud, based on Docker Registry 2.0. Key features include:

- **Registry service tiers** – enable creation of container repositories in the same Azure location as the resources that use those images. This can yield better performance. Premium registries provide geo-replication for enhanced resilience.
- **Security and access** – container images are transferred over HTTPS, client connections are secured using TLS, and Azure role-based access control (Azure RBAC) is used to control access. The Premium service tier also provides content trust, virtual networks and firewalls.
- **Image and artifact** support – ACR supports both Windows and Linux images, as well as the use of standard Docker CLI commands. It also stores Helm charts and OCI-compliant images.
- **Automated image builds** – ACR Tasks the build, test, and deployment of images automatically in Azure. Users can configure build tasks to automate a container build pipeline and accelerate pushing images into the container registry based on code commits.

How to Choose a Container Repository

Here are several aspects to consider when selecting a container repository for your team or organization.

Resilience

An image repository can significantly affect the availability of your workloads. If you are attempting to architect a highly available cloud-based system, one option is to use multiple repository providers, to ensure maximum resilience. Another option is to use cloud-based registries, like AWS ECR or Google Container Registry, and provision your repository in multiple regions.

Integration

The cloud provider you choose will significantly affect the registry you are able to use, because each cloud provider enforces its own limitation and offers different terms.

For example, Alibaba Cloud’s container registry and Azure Container Registry offer the use of fully-managed private container registries, which are hosted within the cloud infrastructure. They also support core Docker APIs. AWS and Google Cloud also offer managed Docker container registries, which integrate with the providers’ CI/CD tools as well as with commonly used tools like Bitbucket and GitHub.

License

Docker offers a free, cloud-based image repository called Docker Hub, which enables entities worldwide to create and share container images. To use this repository, users must sign a license agreement. There are licenses available for open source projects and there are also enterprise-level options that come with commercial considerations regarding intellectual property. It is important to consider which agreement is most appropriate for your use case before selecting a service.

Cost

Each image repository is priced differently. Some repositories are offered for free—these tend to be public repositories, shared under an open source license. Private repositories and managed registries often have a range of pricing options, each suitable for a distinct level of use and tailored to meet a variety of influential factors.

A third-party, private Docker repository service typically costs around $15 per month per repository. Most products bill according to the number of repositories and active users. When used in conjunction with cloud services, organizations can benefit from competitive pricing for hosting, because cloud providers control the infrastructure upon which the repositories function.

It is also important to account for additional costs, such fees associated with transferring data out of, and across, cloud accounts. Additionally, when setting up a multi-region infrastructure for image repositories, it is important to take into account the costs of copying and storing images in multiple repositories.

Security Features

Each repository offers distinct security measures and capabilities. Consider your own organization's security requirements, as well as any regulatory or compliance standards to which your organization must adhere.

Docker Hub, for example, offers subscription accounts that allow basic organizational control and permissions. Enterprise-ready Docker repositories, on the other hand, offer additional security management features and granular security controls. Granular security enables restricted read/write access as well as administrative permissions for specific images, typically based on labels.

Most cloud providers allow integration of the container repository with the organization's identity provider, which enables integrated authentication to enforce security standards without impeding operations.

Most services also offer built-in image vulnerability scanning to detect and identify vulnerabilities which may put images at risk when deployed into production. Docker Enterprise Edition, for example, uses Aqua to scan images for common vulnerabilities and exposures (CVEs) within Docker layers and to identify images which have been tampered with or compromised.

Security Risks of Container Image Repositories

Many DevOps professionals consider container image repositories to be an integral part of modern development cycles and CI/CD release pipelines. Yet container images, and the repositories which store them, may pose potential security risks. It is important to raise awareness of these risks across the organization, and to ensure steps are taken to address, avoid, or mitigate risks before they can be abused by malicious actors.

Common container image repository security risks include:

- **Limited control in the cloud**—the majority of cloud providers offer the use of native registries, which can be used to access images in customer repositories. For example, Amazon Web Services (AWS) offers the use of Elastic Container Registry (ECR), Google Cloud offers the use of the Google Container Registry (GCR), and Microsoft Azure offers the Azure Container Registry (ACR). While these registries are designed with security in mind, they may also limit the control organizations have over the creation and configuration of their own container images. Depending on your organization's workflows and supporting technologies, these restrictions may require modifications to established security-related practices.
- **Compromised public images**—open source images enable quick development. Container images that come from unknown sources may, however, contain vulnerabilities, misconfigurations, or malware that can compromise the organization's IT assets or sensitive data. When using public or third-party repositories, organizations should scan for security vulnerabilities and malicious packages prior to using the images.
- **Unsigned images**—technologies like Docker Content Trust make it possible to verify the author of an image, and ensure it has not been tampered with, via cryptographic signatures. This type of measure can be beneficial, as many public

repositories do not enforce signed images, ultimately obliging the user to ensure that the image they are pulling is authentic.

In general, the use of community repositories for commercial or enterprise purposes is not considered secure. Despite this, many developers or engineering teams may leverage such resources on a smaller scale for internal projects or collaboration.

With these risks in mind, organizations should consider their current and planned usage of public and private repositories and registries, and ensure they implement appropriate security measures to safeguard container images. This includes container vulnerability scanning and integrity verification within repositories, registries, and before deployment into production environments.

Related content: Read our guide to Docker security best practices

Container Image Security with Aqua

The Aqua Platform enables Security and DevOps teams to scan container images for vulnerabilities and security risks that could put cloud native applications and sensitive data at risk. Aqua's deep integration with CI/CD pipelines, image repositories, and container registries means teams don't need to slow down workflows or introduce additional burden to ensure security. Automate scans when new commits are made, accelerate remediation and ticketing, or monitor containers in runtime and prevent unauthorized actions in container environments. To learn more about the risks that can threaten containers, check out Aqua's annual Cloud Native Threat Report. For recommendations on how to prepare for these threats, check out 10 Things DevOps Need to Do for Container Security.