

How to Migrate AWS Elastic Container Registry Artifacts to Vultr Container Registry

Learn the step-by-step process to migrate your container images from AWS Elastic Container Registry to Vultr Container Registry with our comprehensive guide.

Contents

01	Introduction	3
02	Prerequisites	3
03	Prepare the Source AWS Elastic Container Registry	4
04	Set Up the Vultr Container Registry	6
05	Migrate Repositories From the Elastic Container Registry to the Vultr Container Registry	8
06	Migrate Container Registries Using Skopeo	10
07	Verify and Test the Vultr Container Registry Images	14
08	Cut Over Containers to the Vultr Container Registry	16
09	Conclusion	19

Introduction

A container registry is a centralized storage and distribution system for container images, that enables users to store, manage, and retrieve images for deployment across different environments. Vultr Container Registry provides a reliable and scalable solution for storing container images and artifacts. It offers cost efficiency with a predictable pricing model and a user-friendly interface with straightforward configurations.

Migrating container registries from AWS Elastic Container Registry (ECR) to Vultr Container Registry requires a well-structured approach to ensure data integrity, uphold security, and reduce downtime for your active applications.

Follow this guide to migrate from AWS Elastic Container Registries to Vultr Container Registry.

Prerequisites

Before you begin, you need to:

- Have access to an [Ubuntu 24.04 instance](#) as a non-root sudo user to use as the migration workstation.
- [Install Docker](#) on your migration workstation to push and pull images between providers.
- Provision a [Vultr Container Registry to use as the destination registry](#).
- Have access to an [AWS IAM user](#) with the necessary permissions to access AWS Elastic Container Registry (ECR) and a configured [AWS CLI](#) for a successful migration.

Prepare the Source AWS Elastic Container Registry

Before migrating, ensure that the AWS CLI is installed and configured on your migration workstation and you have enough permissions to pull images from AWS Elastic Container Registry (ECR).

1. Log in to AWS Elastic Container Registry.

CONSOLE

```
$ aws ecr get-login-password --region <aws_region> | docker  
login --username AWS --password-stdin  
<aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com
```

Replace `<aws_account_id>` with your AWS Account ID and `<aws_region>` with the AWS Region where your ECR is located.

Your output should be similar to the one below:

```
WARNING! Your password will be stored unencrypted in /home/cloudshell-  
user/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/go/credential-store/  
  
Login Succeeded
```

2. List all the available ECR Repositories.

CONSOLE

```
$ aws ecr describe-repositories --query  
'repositories[*].repositoryName' --output text
```

Your output should be similar to the one below:

```
golf    bravo    delta    foxtrot alpha    charlie echo
```

3. List all the image tags across all repositories.

CONSOLE

```
$ for repo in $(aws ecr describe-repositories --query 'repositories[*].repositoryName' --output text); do \
aws ecr list-images --repository-name $repo --query 'imageIds[*].imageTag' --output text; done
```

The above command returns a list of all the image tags present in your AWS Elastic Container Registry (ECR) repositories.

4. Create a new `pull_images_script.sh` script using a text editor such as `vim`.

CONSOLE

```
$ vim pull_images_script.sh
```

5. Add the following code to the `pull_images_script.sh` file. Replace `<aws_account_id>` with your AWS Account ID and `<aws_region>` with the AWS Region where your ECR is located.

BASH

```
#!/bin/bash

AWS_ACCOUNT_ID="<aws_account_id>"
AWS_REGION="<aws_region>"

for repo in $(aws ecr describe-repositories --query 'repositories[*].repositoryName' --output text); do
  for tag in $(aws ecr list-images --repository-name $repo --query 'imageIds[*].imageTag' --output text); do
    docker pull $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_REGION.amazonaws.com/$repo:$tag
  done
done
```

Press Esc then `: + W + Q` and press enter to save and close the file.

The above script automates the process of pulling all the images from AWS Elastic Container Registry (ECR) along with their associated tags.

6. Grant the execution permission to the script.

CONSOLE

```
$ chmod +x pull_images_script.sh
```

7. Execute the script to pull all the images from ECR to your workstation.

CONSOLE

```
$ ./pull_images_script.sh
```

Your output should be similar to the one below:

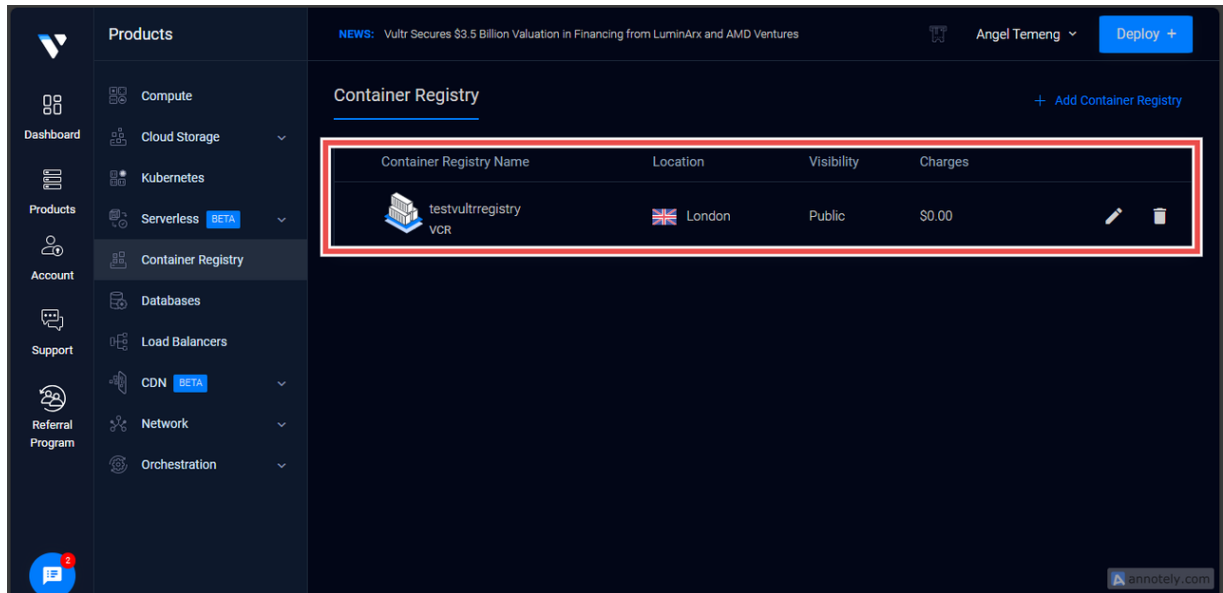
```
latest: Pulling from golf
f0a3fe58120b: Pull complete
Digest: sha256:0f17f65e7a050f35355941bcb375b63867827dd1ab7eb2c949730c195cf0bc25
Status: Downloaded newer image for
<aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/golf:latest
<aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/golf:latest
latest: Pulling from bravo
0d653a9bda6a: Pull complete
b55ba9745e01: Pull complete
...
```

Set Up the Vultr Container Registry

Set up a Vultr Container Registry to store and manage your container images. Follow the steps below to create and configure your registry if not available in your account.

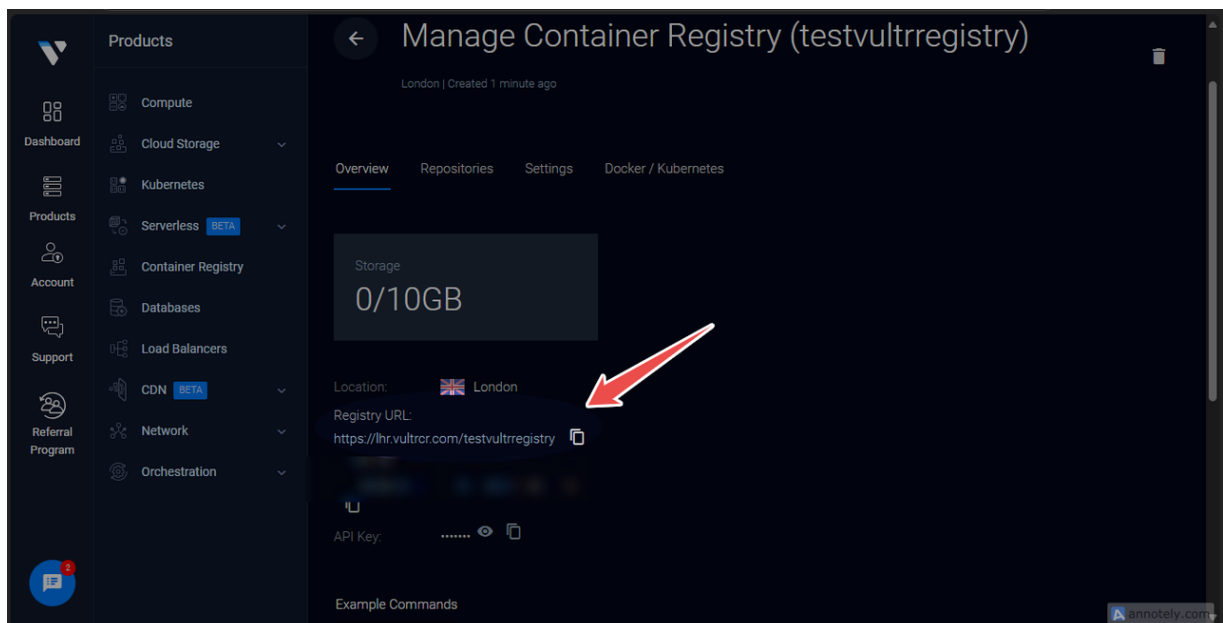
1. Log in to your [Vultr Customer Portal](#).

2. Visit the [Vultr Container Registry Documentation](#) to provision a new container registry.
3. Select the newly created Vultr Container Registry.



In this example, the Vultr Registry's name is `testvultrregistry`.

4. Copy the registry URL from the Registry Details page.



5. Log in to Vultr Container Registry.

CONSOLE

```
$ docker login <vultr_registry_url> --username  
<your_vultr_username> --password <your_vultr_password>
```

Replace `<vultr_registry_url>` with the URL of your Vultr Container Registry, `<your_vultr_username>` with your Vultr Registry username, and `<your_vultr_password>` with the Vultr Registry API Key.

Your output should be similar to the one below:

```
WARNING! Your credentials are stored unencrypted in '/home/cloudshell-  
user/.docker/config.json'.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/go/credential-store/  
  
Login Succeeded
```

Migrate Repositories From the Elastic Container Registry to the Vultr Container Registry

You can migrate individual container images or entire repositories from AWS Elastic Container Registry (ECR) to Vultr Container Registry. Follow the steps below to migrate all repositories, including all tagged images, to Vultr Container Registry.

1. Create a new `migrate_to_vultr.sh` script using a text editor such as `vim`.

CONSOLE

```
$ vim migrate_to_vultr.sh
```

2. Add the following code to the `migrate_to_vultr.sh` file. Replace `<aws_account_id>` with your AWS Account ID, `<aws_region>` with the AWS Region and `<vultr_registry_url>` with your Vultr Registry URL.

```
BASH

#!/bin/bash

AWS_REGION="<aws_region>"
AWS_ACCOUNT_ID="<aws_account_id>"
VULTR_REGISTRY_URL="<vultr_registry_url>"

for repo in $(aws ecr describe-repositories --query
'repositories[*].repositoryName' --output text); do
  for tag in $(aws ecr list-images --repository-name $repo --
query 'imageIds[*].imageTag' --output text); do
    echo "Tagging image for Vultr Container Registry..."
    docker tag $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_REGION.amazonaws.com/$repo:$tag $VULTR_REGISTRY_URL/
$repo:$tag

    echo "Pushing image $repo:$tag to Vultr Container
Registry..."
    docker push $VULTR_REGISTRY_URL/$repo:$tag
  done
done

echo "Migration completed successfully."
```

Press Esc then `:` + W + Q and press enter to save and close the file.

The above script automates the process of pushing all the images to Vultr Container Registry along with their associated tags.

Note

Replace `<vultr_registry_url>` with your actual Vultr Container Registry URL, eliminating the preceding `https://`.

3. Grant the execution permission to the script.

```
CONSOLE

$ chmod +x migrate_to_vultr.sh
```

4. Execute the migration script to push all the images to Vultr Container Registry.

```
CONSOLE
```

```
$ ./migrate_to_vultr.sh
```

Your output should be similar to the one below:

```
...
Tagging image for Vultr Container Registry...
Pushing image echo:busy-latest to Vultr Container Registry...
The push refers to repository [lhr.vultrcr.com/testvultrregistry/echo]
59654b79daad: Mounted from testvultrregistry/charlie
busy-latest: digest:
sha256:9307d4acaeeed71c4c6ba84494aef0464ffc446a5d91f0a7c196fa8eda1bf0590 size: 527
Migration completed successfully.
```

Migrate Container Registries Using Skopeo

Skopeo simplifies the migration process by allowing you to copy container images directly between AWS Elastic Container Registry (ECR) and Vultr Container Registry without pulling them to your local machine. This approach reduces disk space usage and speeds up the migration. Follow the steps below to install `Skopeo` on your workstation and migrate your repositories.

Install Skopeo

Before migrating your repositories, you need to install Skopeo. It's a lightweight tool that makes image transfers seamless.

1. Update the server's APT package index.

```
CONSOLE
```

```
$ sudo apt update
```

2. Install Skopeo.

CONSOLE

```
$ sudo apt install skopeo -y
```

3. View the installed Skopeo version.

CONSOLE

```
$ skopeo --version
```

Your output should be similar to the one below:

```
skopeo version 1.4.1
```

Authenticate with AWS Elastic Container Registry and Vultr Container Registry

To migrate repositories, you need to authenticate with both AWS ECR and Vultr Container Registry. Follow the steps below to log in to the container registries.

1. Log in to AWS Elastic Container Registry.

CONSOLE

```
$ aws ecr get-login-password --region <aws_region> | docker  
login --username AWS --password-stdin  
<aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com
```

Replace `<aws_account_id>` with your AWS Account ID and `<aws_region>` with the AWS Region where your ECR is located.

Your output should be similar to the one below:

```
WARNING! Your password will be stored unencrypted in /home/cloudshell-  
user/.docker/config.json.  
Configure a credential helper to remove this warning. See
```

```
https://docs.docker.com/go/credential-store/
```

```
Login Succeeded
```

2. Log in to Vultr Container Registry.

CONSOLE

```
$ docker login <vultr_registry_url> --username  
<your_vultr_username> --password <your_vultr_password>
```

Replace `<vultr_registry_url>` with the URL of your Vultr Container Registry, `<your_vultr_username>` with your Vultr Registry username, and `<your_vultr_password>` with the Vultr Registry API Key.

Your output should be similar to the one below:

```
WARNING! Your credentials are stored unencrypted in '/home/cloudshell-  
user/.docker/config.json'.
```

```
Configure a credential helper to remove this warning. See  
https://docs.docker.com/go/credential-store/
```

```
Login Succeeded
```

Migrate Container Images Using Skopeo

Follow the steps below to migrate the container images using Skopeo and automate it using a script.

- Skopeo Command Syntax:

```
$ skopeo copy docker://<aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/  
<repo_name>:<tag> \  
docker://<vultr_registry_url>/<repo_name>:<tag>
```

Replace:

- `<aws_account_id>`: Your AWS account ID.
- `<aws_region>`: Your AWS region.
- `<repo_name>`: The repository name.

- `<tag>`: The image tag.
- `<vultr_registry_url>`: Your Vultr Container Registry URL.

1. Create a `migrate_with_skopeo.sh` script to automate the migration.

CONSOLE

```
$ vim migrate_with_skopeo.sh
```

2. Add the following code to the `migrate_with_skopeo.sh` file. Replace `<aws_account_id>` with your AWS Account ID, `<aws_region>` with the AWS Region and `<vultr_registry_url>` with your Vultr Registry URL.

BASH

```
#!/bin/bash

AWS_REGION="<aws_region>"
AWS_ACCOUNT_ID="<aws_account_id>"
VULTR_REGISTRY_URL="<vultr_registry_url>"

for repo in $(aws ecr describe-repositories --query 'repositories[*].repositoryName' --output text); do
  for tag in $(aws ecr list-images --repository-name $repo --query 'imageIds[*].imageTag' --output text); do
    echo "Migrating $repo:$tag with Skopeo..."
    skopeo copy docker://$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$repo:$tag \
      docker://$VULTR_REGISTRY_URL/$repo:$tag
  done
done

echo "Migration completed successfully."
```

Press Esc then `:` + W + Q and press enter to save and close the file.

3. Grant the execution permission to the script.

CONSOLE

```
$ chmod +x migrate_with_skopeo.sh
```

- Execute the migration script to push all the images to Vultr Container Registry.

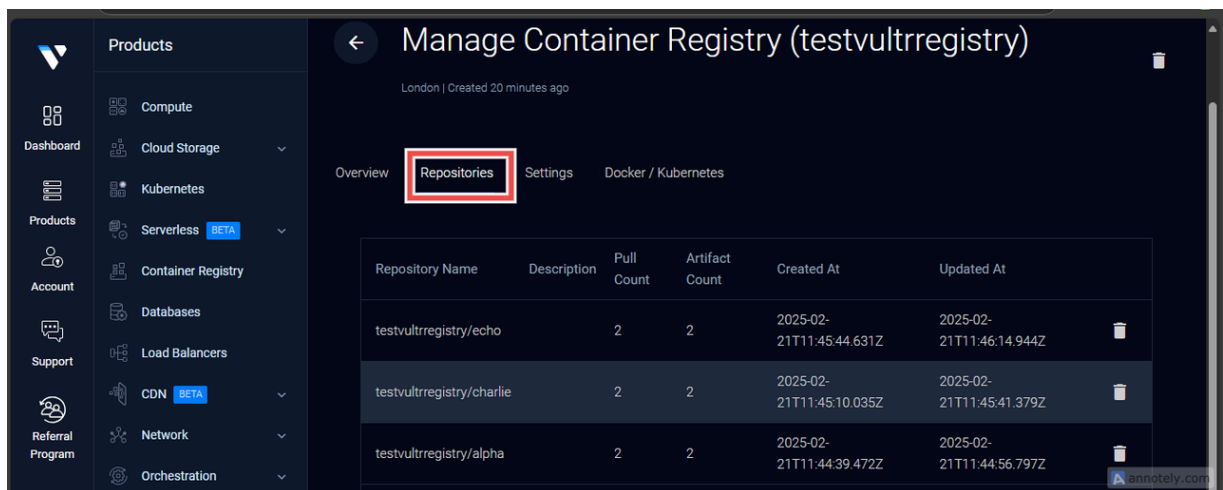
```
CONSOLE
```

```
$ ./migrate_with_skopeo.sh
```

Verify and Test the Vultr Container Registry Images

Verifying the integrity of migrated container images is essential to ensure a successful migration and prevent corruption during the migration process. Follow the steps below to verify that your images were transferred to the Vultr Container Registry. This includes listing the container images, checking their digests, running test containers to confirm functionality.

- Visit the [Vultr Container Registry](#) page.
- Select your registry.
- Navigate to the **Repositories** tab.



In the above image, you can view all your container images that have been migrated to the Vultr Container Registry.

- Verify image integrity with digests.

Each image has a unique SHA-256 digest, which you can compare between AWS ECR and the Vultr Container Registry to confirm they match. Follow the steps below to perform this check.

1. Retrieve the digest of an image stored in AWS ECR.

CONSOLE

```
$ aws ecr describe-images --repository-name <repo_name>
--query 'imageDetails[*].[imageDigest]' --output text
```

Your output should be similar to the one below:

```
sha256:ec1b05d1eac264d9204a57f4ad9d4dc35e9e756e9fedaea0674aefc7edb1d6a4
```

2. Pull the same image from the Vultr Container Registry using the correct tag.

CONSOLE

```
$ docker pull <vultr_registry_url>/<repo_name>:<tag>
```

3. Retrieve the digest of the same image stored in Vultr Container Registry.

CONSOLE

```
$ docker inspect <vultr_registry_url>/<repo_name>:<tag>
--format='{{index .RepoDigests 0}}' | awk -F'@' '{print
$2}'
```

Your output should be similar to the one below:

```
sha256:ec1b05d1eac264d9204a57f4ad9d4dc35e9e756e9fedaea0674aefc7edb1d6a4
```

In the example above, the image integrity has been preserved, meaning both digests are the same, indicating a successful transfer.

5. Run a test container to ensure the image functions as expected.

CONSOLE

```
$ docker run --rm <vultr_registry_url>/<repo_name>:<tag>  
echo "Test successful!"
```

Your output should be similar to the one below:

```
Test Successful
```

Cut Over Containers to the Vultr Container Registry

Cutting over to Vultr Container Registry after migrating your existing images allows you to seamlessly transition your applications and services. Follow the recommendations below after a successful migration to cut over your containers to Vultr Container Registry.

1. Update your Dockerfiles to use the Vultr Container Registry.

- Before:

DOCKERFILE

```
FROM  
<aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/  
<repo_name>:<tag>
```

- After:

DOCKERFILE

```
FROM <vultr_registry_url>/<repo_name>:<tag>
```

Replace `<vultr_registry_url>` with the actual URL of your Vultr registry.

2. Update CI/CD pipelines to use the Vultr Container Registry.

- Before (AWS ECR in GitHub Actions):

YAML

```
- name: Log in to AWS ECR
  run: aws ecr get-login-password --region <aws_region>
| docker login --username AWS --password-stdin
<aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com

- name: Pull and run the image
  run: |
    docker pull
    <aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/
    <repo_name>:<tag>
    docker run --rm
    <aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/
    <repo_name>:<tag> echo "Test successful!"
```

- After (Vultr Container Registry in GitHub Actions):

YAML

```
- name: Login to Vultr Container Registry
  run: docker login <vultr_registry_url> --username
<your_vultr_username> --password <your_vultr_password>

- name: Pull and run the new image
  run: |
    docker pull <vultr_registry_url>/<repo_name>:<tag>
    docker run --rm <vultr_registry_url>/
    <repo_name>:<tag> echo "Test successful!"
```

Replace `<vultr_registry_url>` with the actual URL of your Vultr registry, and `<your_vultr_username>` and `<your_vultr_password>` with your actual Vultr credentials.

3. Redeploy your services to use the Vultr Container Registry.

- Docker Compose.
 - Before: Update the app image URL.

YAML

```
services:
  app:
    image:
      <aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/
      <repo_name>:<tag>
```

■ After:

YAML

```
services:
  app:
    image: <vultr_registry_url>/<repo_name>:<tag>
```

1. Redeploy the docker compose services.

CONSOLE

```
$ docker-compose down && docker-compose up -d
```

- Kubernetes.

1. Update the image under the container section in the definition to use Vultr Container Registry.

■ Before:

YAML

```
containers:
- name: app
  image:
    <aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/
    <repo_name>:<tag>
```

■ After:

YAML

```
containers:  
- name: app  
  image: <vultr_registry_url>/<repo_name>:<tag>
```

2. Apply the changes made in the definition file.

CONSOLE

```
$ kubectl apply -f <definition-file>.yaml
```

Conclusion

You have migrated container images from AWS Elastic Container Registry (ECR) to Vultr Container Registry. This guide helps you with updating Dockerfiles, configuring CI/CD pipelines, and redeploying services using Docker Compose and Kubernetes. The migration process minimizes downtime, reduces costs, and streamlines your container management. For more information on managing your Vultr Container Registry, visit the [Vultr Container Registry documentation](#).



VULTR

