Manage Signed Images with Docker Content Trust in Azure Container Registry - Azure Container Registry

By rayoef

Archived: 2026-04-05 17:02:17 UTC

Azure Container Registry implements the Docker Content Trust (DCT) model to enable pushing and pulling signed images. This article gets you started with enabling DCT in your container registries. DCT is a feature of the Premium service tier of Container Registry.

Important

Starting May 31, 2026, Docker Content Trust cannot be enabled on new container registries or registries that haven't enabled it previously.

DCT will be deprecated and completely removed on March 31, 2028. For details and transition guidance, refer to Transition from Docker Content Trust to Notary Project.

Limitations

A token that has repository-scoped permissions doesn't currently support Docker push and pull of signed images.

How DCT works

Important to any distributed system that's designed with security in mind is verifying both the *source* and the *integrity* of data entering the system. Consumers of the data need to be able to verify the publisher (source) of the data and ensure that the data wasn't modified after it was published (integrity).

As an image publisher, you can use DCT to sign the images that you push to your registry. Consumers of your images (people or systems pulling images from your registry) can configure their clients to pull only signed images. When an image consumer pulls a signed image, their Docker client verifies the integrity of the image. In this model, consumers are assured that you published the signed images in your registry, and that the images weren't modified after publication.

Note

Container Registry does not support `acr import` to import images signed with DCT. By design, the signatures aren't visible after the import. Notary v2 stores these signatures as artifacts.

Trusted images

DCT works with the tags in a repository. Image repositories can contain images that have both signed and unsigned tags. For example, you might sign only the `myimage:stable` and `myimage:latest` images, but not

`myimage:dev`.

Signing keys

You manage DCT by using a set of cryptographic signing keys. These keys are associated with a specific repository in a registry.

There are several types of signing keys that Docker clients and your registry use in managing trust for the tags in a repository. When you enable DCT and integrate it into your pipeline for container publishing and consumption, you must manage these keys carefully. For more information, see Manage keys later in this article and Manage keys for content trust in the Docker documentation.

Enable DCT for the registry

Your first step is to enable DCT at the registry level. After you enable DCT, clients (users or services) can push signed images to your registry.

Enabling DCT for your registry doesn't restrict registry usage to only consumers who have DCT enabled. Consumers who don't have DCT enabled can continue to use your registry as normal. Consumers who enabled DCT in their clients, however, can see *only* signed images in your registry.

To enable DCT for your registry by using the Azure portal, go to the registry. Under **Policies**, select **Content trust**. Select **Enabled**, then select **Save**. You can also use the az acr config content-trust update command in the Azure CLI.

Enable DCT for the client

To work with trusted images, both image publishers and consumers need to enable DCT for their Docker clients. As a publisher, you can sign the images that you push to a DCT-enabled registry. As a consumer, enabling DCT limits your view of a registry to signed images only. DCT is disabled by default in Docker clients, but you can enable it per shell session or per command.

To enable DCT for a shell session, set the `DOCKER_CONTENT_TRUST` environment variable to `1`. For example, in the Bash shell, use this code:

```
 # Enable DCT for a shell session
export DOCKER_CONTENT_TRUST=1
```

If you want to enable or disable DCT for a single command, several Docker commands support the `--disable-content-trust` argument.

To enable DCT for a single command, use this code:

```
 # Enable DCT for a single command
docker build --disable-content-trust=false -t myacr.azurecr.io/myimage:v1 .
```

If you enabled DCT for your shell session and want to disable it for a single command, use this code:

```
 # Disable DCT for a single command
docker build --disable-content-trust -t myacr.azurecr.io/myimage:v1 .
```

Grant image signing permissions

Only the users or systems to which you grant permission can push trusted images to your registry. To grant this push permission to a user (or a system by using a service principal), assign the `AcrImageSigner` role to the user's Microsoft Entra identity. This permission is in addition to the `AcrPush` (or equivalent) role that's required for pushing images to the registry. For more information, see Azure Container Registry permissions and role assignments overview.

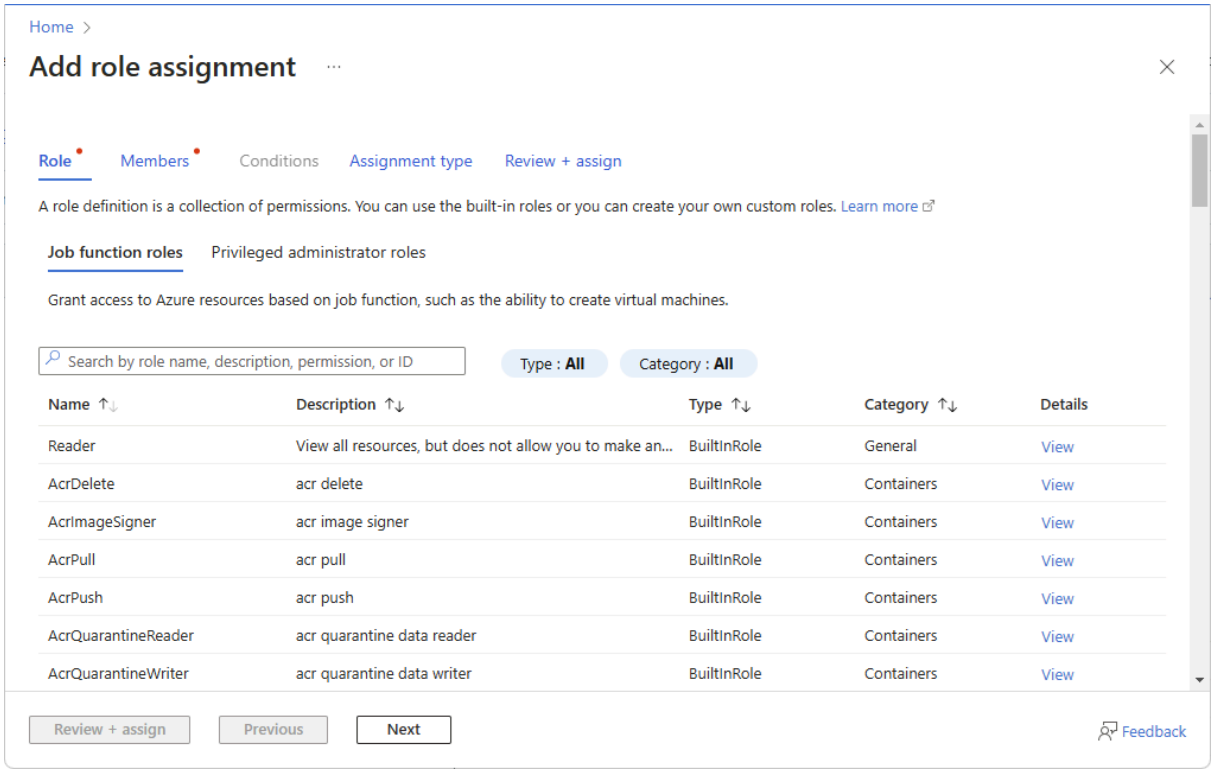
The `AcrImageSigner` role includes both the `Microsoft.ContainerRegistry/registries/sign/write` action and the `Microsoft.ContainerRegistry/registries/trustedCollections/write` data action.

Azure portal

To grant the `AcrImageSigner` role by using the Azure portal:

1. Select **Access control (IAM)**.
2. Select **Add** > **Add role assignment** to open the **Add role assignment** pane.
3. Assign the following role. In this example, the role is assigned to an individual user. For detailed steps, see Assign Azure roles by using the Azure portal.

Setting	Value
Role	**AcrImageSigner**
Assign access to	**User**
Members	**Alain**



Azure CLI

To grant signing permissions to a user by using the Azure CLI, assign the `AcrImageSigner` role to the user, scoped to your registry. The format of the command is:

```
az role assignment create --scope <registry ID> --role AcrImageSigner --assignee <user name>
```

For example, to assign the role to a non-administrative user, you can run the following commands in an authenticated Azure CLI session. Modify the `REGISTRY` value to reflect the name of your Azure container registry.

```
 # Grant signing permissions to an authenticated Azure CLI user
REGISTRY=myregistry
REGISTRY_ID=$(az acr show --name $REGISTRY --query id --output tsv)
```

```
az role assignment create --scope $REGISTRY_ID --role AcrImageSigner --assignee azureuser@contoso.com
```

You can also grant a service principal the rights to push trusted images to your registry. Using a service principal is useful for build systems and other unattended systems that need to push trusted images to your registry. The format is similar to granting a user permission, but you specify a service principal ID for the `--assignee` value:

```
az role assignment create --scope $REGISTRY_ID --role AcrImageSigner --assignee <service principal ID>
```

The `<service principal ID>` value can be the service principal's `appId` value, its `objectId` value, or one of its `servicePrincipalNames` values. For more information about working with service principals and Azure Container Registry, see Azure Container Registry authentication with service principals.

Push a trusted image

To push a trusted image tag to your container registry, enable DCT and use `docker push`. After pushing with a signed tag finishes the first time, you're asked to create a passphrase for both a root signing key and a repository signing key. Both the root and repository keys are generated and stored locally on your machine.

```
 $ docker push myregistry.azurecr.io/myimage:v1
[...]
The push refers to repository [myregistry.azurecr.io/myimage]
ee83fc5847cb: Pushed
v1: digest: sha256:aca41a608e5eb015f1ec6755f490f3be26b48010b178e78c00eac21ffbe246f1 size: 524
Signing and pushing trust metadata
You are about to create a new root signing key passphrase. This passphrase
will be used to protect the most sensitive key in your signing system. Please
choose a long, complex passphrase and be careful to keep the password and the
key file itself secure and backed up. It is highly recommended that you use a
password manager to generate the passphrase and keep it safe. There will be no
way to recover this key. You can find the key in your config directory.
Enter passphrase for new root key with ID 4c6c56a:
Repeat passphrase for new root key with ID 4c6c56a:
Enter passphrase for new repository key with ID bcd6d98:
Repeat passphrase for new repository key with ID bcd6d98:
Finished initializing "myregistry.azurecr.io/myimage"
Successfully signed myregistry.azurecr.io/myimage:v1
```

After your first `docker push` action with DCT enabled, the Docker client uses the same root key for subsequent pushes. On each subsequent push to the same repository, you're asked only for the repository key. Each time you push a trusted image to a new repository, you're asked to supply a passphrase for a new repository key.

Pull a trusted image

To pull a trusted image, enable DCT and run the `docker pull` command as normal. To pull trusted images, the `AcrPull` role is enough for normal users. No additional roles (like an `AcrImageSigner` role) are required. Consumers who have DCT enabled can pull only images that have signed tags. Here's an example of pulling a signed tag:

```
 $ docker pull myregistry.azurecr.io/myimage:signed
Pull (1 of 1): myregistry.azurecr.io/myimage:signed@sha256:0800d17e37fb4f8194495b1a188f121e5b54efb52b5d93dc9e0ec
sha256:0800d17e37fb4f8194495b1a188f121e5b54efb52b5d93dc9e0ed97fce49564b: Pulling from myimage
8e3ba11ec2a2: Pull complete
Digest: sha256:0800d17e37fb4f8194495b1a188f121e5b54efb52b5d93dc9e0ed97fce49564b
```

```
Status: Downloaded newer image for myregistry.azurecr.io/myimage@sha256:0800d17e37fb4f8194495b1a188f121e5b54efb5
Tagging myregistry.azurecr.io/myimage@sha256:0800d17e37fb4f8194495b1a188f121e5b54efb52b5d93dc9e0ed97fce49564b as
```

If a client that has DCT enabled tries to pull an unsigned tag, the operation fails with an error similar to the following example:

```
 $ docker pull myregistry.azurecr.io/myimage:unsigned
Error: remote trust data does not exist
```

Behind the scenes

When you run `docker pull`, the Docker client uses the same library as in the Notary CLI to request the tag-to-SHA-256 digest mapping for the tag that you're pulling. After the client validates the signatures on the trust data, it instructs Docker Engine to do a "pull by digest." During the pull, the engine uses the SHA-256 checksum as a content address to request and validate the image manifest from the Azure container registry.

Note

Azure Container Registry doesn't officially support the Notary CLI but is compatible with the Notary Server API. The Notary Server API is included with Docker Desktop. Currently, we recommend Notary version 0.6.0.

Manage keys

As stated in the `docker push` output when you push your first trusted image, the root key is the most sensitive. By default, the Docker client stores signing keys in the following directory:

```
~/.docker/trust/private
```

Back up your root and repository keys by compressing them in an archive and storing the archive in a secure location. For example, use this command in Bash:

```
umask 077; tar -zcvf docker_private_keys_backup.tar.gz ~/.docker/trust/private; umask 022
```

Along with the locally generated root and repository keys, Container Registry generates and stores several other keys when you push a trusted image. For a detailed discussion of the various keys in the DCT implementation, including management guidance, see Manage keys for content trust in the Docker documentation.

Lost root key

If you lose access to your root key, you lose access to the signed tags in any repository whose tags that key signed. Container Registry can't restore access to image tags signed with a lost root key. To remove all trust data (signatures) for your registry, disable and then re-enable DCT for the registry.

Warning

Disabling and re-enabling DCT in your registry *deletes all trust data for all signed tags in every repository in your registry*. This action is irreversible. Container Registry can't recover deleted trust data. Disabling DCT does not delete the images themselves.

To disable DCT for your registry, go to the registry in the Azure portal. Under **Policies**, select **Content Trust**. Select **Disabled**, then select **Save**. You're warned of the loss of all signatures in the registry. Select **OK** to permanently delete all signatures in your registry.

Related content

- For more information about DCT, including `docker trust` commands and trust delegations, see Content trust in Docker.
- For an example of using DCT when you build and push a Docker image, see the Azure Pipelines documentation.
