# Notary Trust

## Using the MTR Notary

Assuming you pushed content to the registry already, or you want to access existing content, log in to the registry using your credentials:

```
$ docker login -u $USERNAME -p $PASSWORD mtr.external.otc.telekomcloud.com
```

The following environment variables have to be set:

```
$ export DOCKER_CONTENT_TRUST=1
$ export DOCKER_CONTENT_TRUST_SERVER=https://notary.external.otc.telekomcloud.com
```

Check if the notary works for you:

```
$ notary -d ~/.docker/trust -s $DOCKER_CONTENT_TRUST_SERVER list mtr.external.otc.telekomcloud.com/e2etest
/trustedrepo
NAME    DIGEST                                                              SIZE (BYTES)    ROLE
----    ------                                                              ------------    ----
1.2     a15790640a6690aa1730c38cf0a440e2aa44aaca9b0e8931a9f2b0d7cc90fd65    528             targets
```

In the following example, you pull a sample container image, e.g. security/redis:latest with DOCKER_CONTENT_TRUST unset for now. If you try to pull the (so far) untrusted container image with TRUST set, you will get an error message:

```
$ DOCKER_CONTENT_TRUST=1
$ docker pull mtr.external.otc.telekomcloud.com/e2etest/redis
Using default tag: latest
Error: remote trust data does not exist for mtr.external.otc.telekomcloud.com/e2etest/redis: notary.external.
otc.telekomcloud.com does not have trust data for mtr.external.otc.telekomcloud.com/e2etest/redis
```

OK, so you need to pull the container image with trust disabled:

```
$ docker pull --disable-content-trust mtr.external.otc.telekomcloud.com/e2etest/redis
Using default tag: latest
latest: Pulling from e2etest/redis
afb6ec6fdc1c: Pull complete
...
618fdf7d0dec: Pull complete
Digest: sha256:322db967e8e590510e50b03892cebc81a58ceda69326fb1231e4a27a8bb0d28e
Status: Downloaded newer image for mtr.external.otc.telekomcloud.com/e2etest/redis:latest
mtr.external.otc.telekomcloud.com/e2etest/redis:latest
```

Now you can modify / commit the pulled container image and push a trusted version with a new tag, e.g. when your container image is production ready for everyone else. If you do this for the first time, docker will set up trust for you by creating a root key for the docker trust folder. The root key is protected by a passphrase, which docker asks for. Then for every repository a target (or repo) key is created which as well needs to be protected by a passphrase. The keys are stored in the docker trust directory under subfolder `private/root_keys` and `private/tuf_keys`. It is good practice to keep the keys in a safe place (make backups!), and to use a password manager like KeePass to create long random passphrases for you. Lost keys or passphrases are extremely hard to recover, in particular the root key.

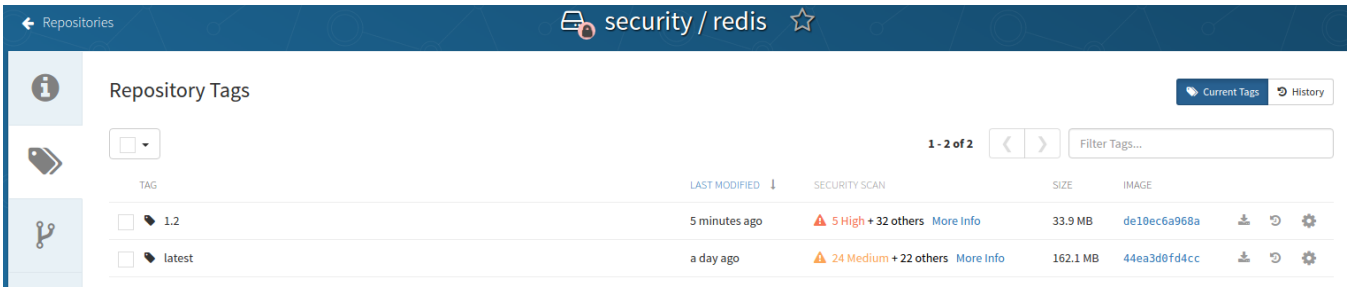
```
$ docker commit -m "New Redis build" -a "Joe User"  271cdaf1d774 mtr.external.otc.telekomcloud.com/e2etest
/redis:1.2
sha256:cfd09e0364bc3626da6ac06babd46d1cb59d37856d5e3b7ceff2759e017ca9b6

$ docker push  mtr.external.otc.telekomcloud.com/e2etest/redis:1.2
The push refers to repository [mtr.external.otc.telekomcloud.com/e2etest/redis]
6ef422d19214: Layer already exists
..
ffc9b21953f4: Layer already exists
1.2: digest: sha256:0dc07c996e496d039f91be9b54c88261276c973c155a22890a58ea6b01d954ba size: 1572
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 4581e9f:
Passphrase is too short. Please use a password manager to generate and store a good random passphrase.
Enter passphrase for new root key with ID 4581e9f:
Repeat passphrase for new root key with ID 4581e9f:
Enter passphrase for new repository key with ID afe600c:
Repeat passphrase for new repository key with ID afe600c:
Finished initializing "mtr.external.otc.telekomcloud.com/e2etest/redis"
Successfully signed mtr.external.otc.telekomcloud.com/e2etest/redis:1.2
```

Check the trust for security/redis:

```
$ notary -d ~/.docker/trust -s $DOCKER_CONTENT_TRUST_SERVER list mtr.external.otc.telekomcloud.com/e2etest/redis
NAME    DIGEST                                                              SIZE (BYTES)    ROLE
----    ------                                                              ------------    ----
1.2     0dc07c996e496d039f91be9b54c88261276c973c155a22890a58ea6b01d954ba    1572            targets
```

After this step, you will find the created root and repository keys in `~/.docker/trust`.

Currently if you check the repository in Quay, you will find that Quay knows nothing about trust. In the next release of Quay the trust will be shown (expected in September). This is how it looks like in Quay now:




Please note that it is the tag that is signed and trusted. It is possible to push another version of the container image with the same tag to the registry with DOCKER_CONTENT_TRUST disabled. This will push a separate version of the container image without trust. Both are separate entities in the registry.

This said, it is important that you keep track about your current trust settings (i.e. the DOCKER_CONTENT_TRUST value) yourself when pulling trusted content in order to not inadvertently pull untrusted (fake? malicious?) content instead.

### Transfer trust

There are two ways to enable a third party to deliver trusted images; either export the target key or establish a delegation:

#### Export target key

```
user1$ export DOCKER_CONTENT_TRUST_SERVER=https://notary.external.otc.telekomcloud.com
user1$ export TRUSTDIR=~/.docker/trust/
user1$ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER key list | grep ibrain2/frontend
  targets   ...d.com/ibrain2/frontend   <keyid>  ...(/home/xubuntu/.docker/trust/private)
user1$ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER key export <keyid> /tmp/ibrain2_frontend.pem
```

### Import target key

```
user2@puno:~$ export DOCKER_CONTENT_TRUST_SERVER=https://notary.external.otc.telekomcloud.com
user2@puno:~$ export TRUSTDIR=~/.docker/trust/
user2@puno:~$ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER -g mtr.external.otc.telekomcloud.com/ibrain2
/frontend key import /tmp/ibrain2_frontend.pem
Enter passphrase for imported targets key:
Enter passphrase for new targets key with ID c912a67 (mtr.external.otc.telekomcloud.com/ibrain2/frontend):
Repeat passphrase for new targets key with ID c912a67 (mtr.external.otc.telekomcloud.com/ibrain2/frontend):
user2@puno:~$ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER key list | grep ibrain2/frontend
  targets   ...d.com/ibrain2/frontend   c912a6727ee15c6684cc46905d7dc82df456a4a15c4b4279e4cda5f3d7f77540   ...
/home/user2/.docker/trust/private)
```

Now user2 can also push a signed image to the repository.

### Establish delegation

In case the trust has to be revoked by a third party, the usage of delegations has an advantage.

The supplier has to provide a certificate which is added to the repository as a delegation.

```
user1$ export GUN=mtr.external.otc.telekomcloud.com/ibrain2/frontend

user1$ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER delegation add $GUN targets/releases keys/delegation.
crt --all-paths
Addition of delegation role targets/releases with keys
[c300caa0d09937c7e658b61a06f0f9938a1c8b51ec31538ba87761208a37d171], with paths ["" <all paths>], to repository
"mtr.external.otc.telekomcloud.com/ibrain2/frontend" staged for next publish.

user1$ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER publish $GUN
Pushing changes to mtr.external.otc.telekomcloud.com/ibrain2/frontend
Enter passphrase for targets key with ID c912a67 (mtr.external.otc.telekomcloud.com/ibrain2/frontend):

user1$ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER  delegation list  $GUN

        ROLE             PATHS                                    KEY IDS
THRESHOLD
-----------------------------------------------------------------------------------------------------------------
----
  targets/releases   "" <all paths>   c300caa0d09937c7e658b61a06f0f9938a1c8b51ec31538ba87761208a37d171   1
```

Without having the target key the trusted delivery fails:

```
user3$ docker push mtr.external.otc.telekomcloud.com/ibrain2/frontend:1.31138
The push refers to a repository [mtr.external.otc.telekomcloud.com/ibrain2/frontend]
...
Signing and pushing trust metadata
Failed to sign "mtr.external.otc.telekomcloud.com/ibrain2/frontend":1.31138 - no valid signing keys for
delegation roles
no valid signing keys for delegation roles
```

After an import of the private key, it works:

```
user3 $ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER  key import keys/delegation.key --role user
Enter passphrase for new user key with ID c300caa:
Repeat passphrase for new user key with ID c300caa:

evernote@puno:~$ docker push mtr.external.otc.telekomcloud.com/ibrain2/frontend:1.31138
The push refers to a repository [mtr.external.otc.telekomcloud.com/ibrain2/frontend]
...

Signing and pushing trust metadata
Enter passphrase for user key with ID c300caa:
Successfully signed "mtr.external.otc.telekomcloud.com/ibrain2/frontend":1.31138
```

### Revoke delegation

Now it is possible just to revoke one delegation without rotating the target key:

```
user1 $ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER delegation remove $GUN targets/releases
Are you sure you want to remove all data for this delegation? (yes/no)
yes
Forced removal (including all keys and paths) of delegation role targets/releases to repository "mtr.external.
otc.telekomcloud.com/ibrain2/frontend" staged for next publish.

user1 $ notary -d $TRUSTDIR -s $DOCKER_CONTENT_TRUST_SERVER publish $GUN
Pushing changes to mtr.external.otc.telekomcloud.com/ibrain2/frontend
Enter passphrase for targets key with ID c912a67 (mtr.external.otc.telekomcloud.com/ibrain2/frontend):
```

Consequently, user3 can not deliver trusted anymore.

```
user3 $ docker push mtr.external.otc.telekomcloud.com/ibrain2/frontend:1.31138
The push refers to a repository [mtr.external.otc.telekomcloud.com/ibrain2/frontend]
...
Signing and pushing trust metadata
ERRO[0007] couldn't add target to targets: could not find necessary signing keys, at least one of these keys
must be available: c912a6727ee15c6684cc46905d7dc82df456a4a15c4b4279e4cda5f3d7f77540
Failed to sign "mtr.external.otc.telekomcloud.com/ibrain2/frontend":1.31138 - could not find necessary signing
keys, at least one of these keys must be available:
c912a6727ee15c6684cc46905d7dc82df456a4a15c4b4279e4cda5f3d7f77540
Error: could not find signing keys for remote repository mtr.external.otc.telekomcloud.com/ibrain2/frontend, or
could not decrypt signing key: could not find necessary signing keys, at least one of these keys must be
available: c912a6727ee15c6684cc46905d7dc82df456a4a15c4b4279e4cda5f3d7f77540
```

## Automating trust handling

To automate the trust handling, the following environment variables can be set to bypass passphrase requests:

- `DOCKER_CONTENT_TRUST_ROOT_PASSPHRASE`
- `DOCKER_CONTENT_TRUST_REPOSITORY_PASSPHRASE`

More info: https://docs.docker.com/engine/security/trust/content_trust/#content-trust-operations-and-keys, https://docs.docker.com/engine/reference/commandline/cli/#notary,