In this scenario, we'll cover how to launch a private Docker Registry with TLS via SSL.

A private Registry enables you to distribute Docker Images without being dependent on external providers or the public cloud. This allows you to increase security and confidence of your image sources and versioning.

TLS (Transport Layer Security) is a protocol that provides secure communication over the internet. It is the successor to SSL (Secure Sockets Layer) and is used to encrypt data transmitted between a client and a server to protect it from interception and tampering. TLS is commonly used in web browsing, email, messaging, and other online applications.

To use TLS, both the client and server must have a valid SSL/TLS certificate. The SSL/TLS certificate is issued by a trusted third-party certificate authority (CA), and it contains information about the identity of the certificate holder, such as their name and domain name.

When a client establishes a connection with a server, the server presents its SSL/TLS certificate to the client. The client verifies the certificate against a list of trusted CAs to ensure that it is valid and has not been tampered with. If the certificate is valid, the client and server negotiate a shared encryption key and use it to encrypt all data transmitted between them.

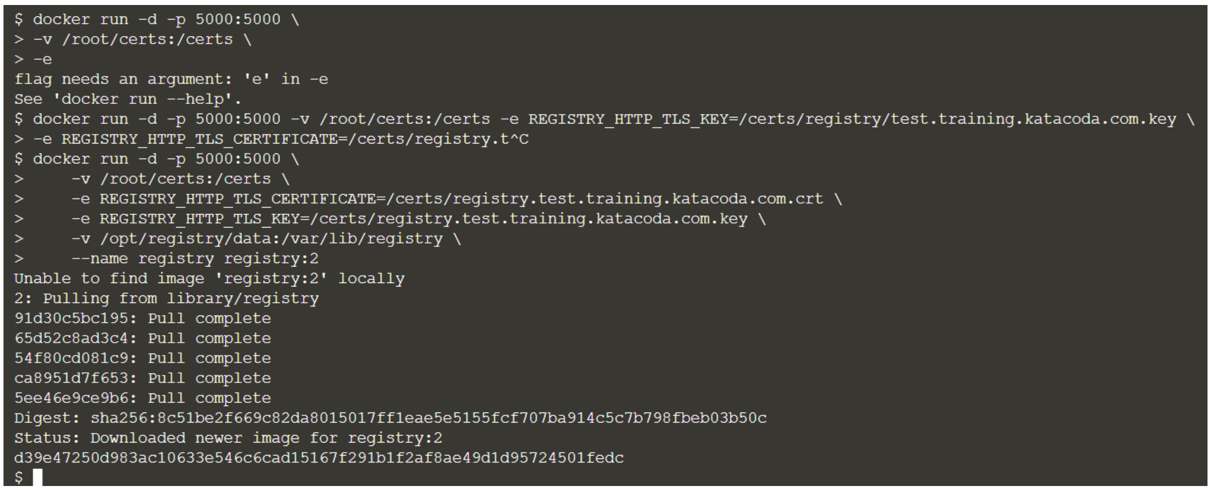
To enable TLS on a server, you will need to obtain a valid SSL/TLS certificate from a trusted CA, install it on the server, and configure the server to use TLS. In addition, you may need to configure client applications to use TLS when communicating with the server.

It is important to note that TLS is not foolproof, and it is still possible for attackers to intercept and tamper with encrypted data under certain circumstances. Therefore, it is important to use other security measures, such as firewalls and intrusion detection systems, to protect against attacks.

Step 1 - Starting Registry
The Registry is deployed as a container and accessible via port 5000. Docker clients will use this domain to access the registry and push/pull images. By specifying a domain, a client can access multiple registries.

In this example our Docker registry is located at registry.test.training.katacoda.com.

```
docker run -d -p 5000:5000 \
   -v /root/certs:/certs \
   -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/registry.test.training.katacoda.com.crt \
   -e REGISTRY_HTTP_TLS_KEY=/certs/registry.test.training.katacoda.com.key \
   -v /opt/registry/data:/var/lib/registry \
   --name registry registry:2
```
Mounting the volume /var/lib/registry is important. This is where the Registry will store all of the pushed images. Mounting the directory will allow you to restart and upgrade the container in future.



```
$ docker run -d -p 5000:5000 \
> -v /root/certs:/certs \
> -e
flag needs an argument: 'e' in -e
See 'docker run --help'.
$ docker run -d -p 5000:5000 -v /root/certs:/certs -e REGISTRY_HTTP_TLS_KEY=/certs/registry/test.training.katacoda.com.key \
> -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/registry.t^C
$ docker run -d -p 5000:5000 \
>     -v /root/certs:/certs \
>     -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/registry.test.training.katacoda.com.crt \
>     -e REGISTRY_HTTP_TLS_KEY=/certs/registry.test.training.katacoda.com.key \
>     -v /opt/registry/data:/var/lib/registry \
>     --name registry registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
91d30c5bc195: Pull complete
65d52c8ad3c4: Pull complete
54f80cd081c9: Pull complete
ca8951d7f653: Pull complete
5ee46e9ce9b6: Pull complete
Digest: sha256:8c51be2f669c82da8015017ff1eae5e5155fcf707ba914c5c7b798fbeb03b50c
Status: Downloaded newer image for registry:2
d39e47250d983ac10633e546c6cad15167f291b1f2af8ae49d1d95724501fedc
$
```

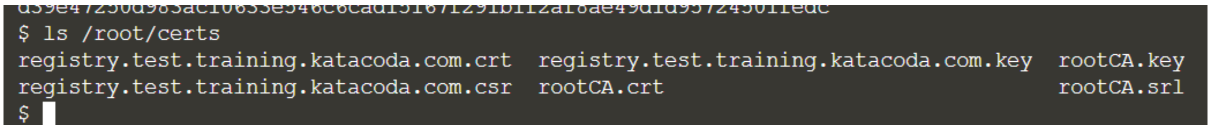
Step 2 - SSL
Securing access to the Registry via TLS is important. If the Registry is insecure, then you'll need to configure every Docker daemon accessing the Registry to allow access.

To secure the Registry, we'll use SSL certificates combined with NGINX to manage the SSL termination. We've added the certificate and key to the certs directory on the client.

ls /root/certs/

When creating the Registry container, the certs where mounted in with the correct environment variables set for the Registry to pickup the certificates.

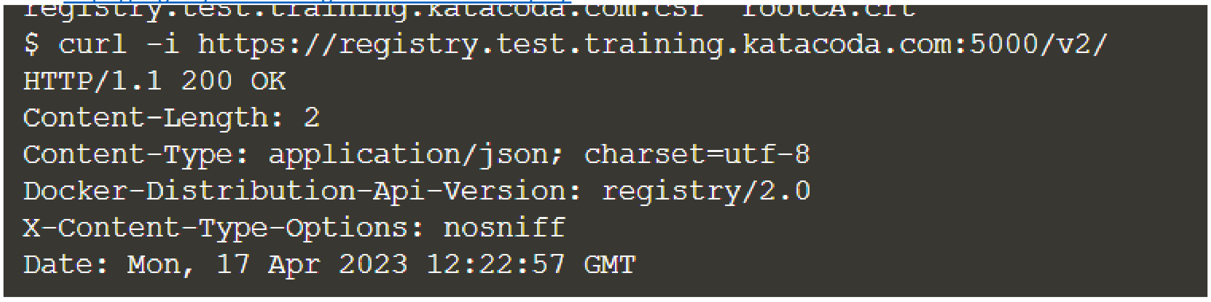
-v /root/certs:/certs \
-e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/registry.test.training.katacoda.com.crt \
-e REGISTRY_HTTP_TLS_KEY=/certs/registry.test.training.katacoda.com.key \
Important
These certificates are only for test purposes and not production use. Visit letsencrypt.org to obtain a certificate for your domain.



```
d59e47250d983ac10633e546c6cad15187129lbf12af8ae49d1d95724501fedc
$ ls /root/certs
registry.test.training.katacoda.com.crt  registry.test.training.katacoda.com.key  rootCA.key
registry.test.training.katacoda.com.csr  rootCA.crt                               rootCA.srl
$
```

Step 3 - Testing
We can test access to the registry using curl. The response should provide headers, for example Docker-Distribution-API-Version, indicating the request was processed by the Registry server.

curl -i https://registry.test.training.katacoda.com:5000/v2/



```
registry.test.training.katacoda.com.csr  rootCA.crt
$ curl -i https://registry.test.training.katacoda.com:5000/v2/
HTTP/1.1 200 OK
Content-Length: 2
Content-Type: application/json; charset=utf-8
Docker-Distribution-Api-Version: registry/2.0
X-Content-Type-Options: nosniff
Date: Mon, 17 Apr 2023 12:22:57 GMT
```

Step 4 - Pushing Images
The Registry is now running.

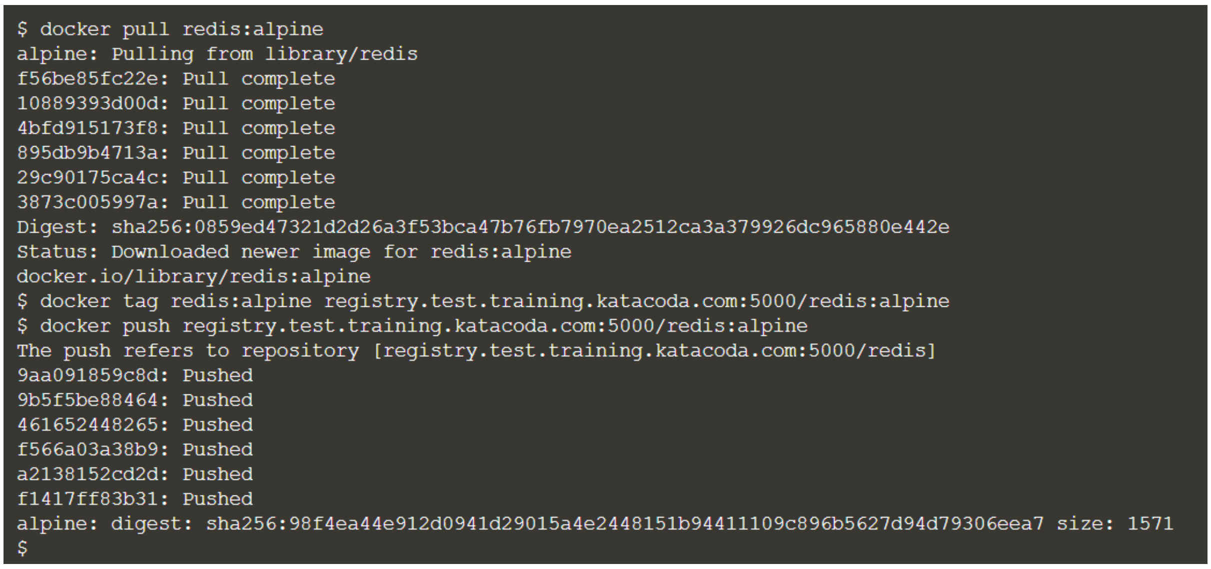
We now need to push images to our new Registry. To push/pull images from non-default Registries we need to include the URL in the image name. Generally, an image follows a <name>:<tag> format as Docker defaults to the public registry. The full format is <registry-url>:<name>:<tag>.

We can use the docker tag to add additional tags to existing images. In this case the Redis image.

docker pull redis:alpine; docker tag redis:alpine registry.test.training.katacoda.com:5000/redis:alpine

Once tagged we can push the image. The different layers of the image will be pushed.

docker push registry.test.training.katacoda.com:5000/redis:alpine



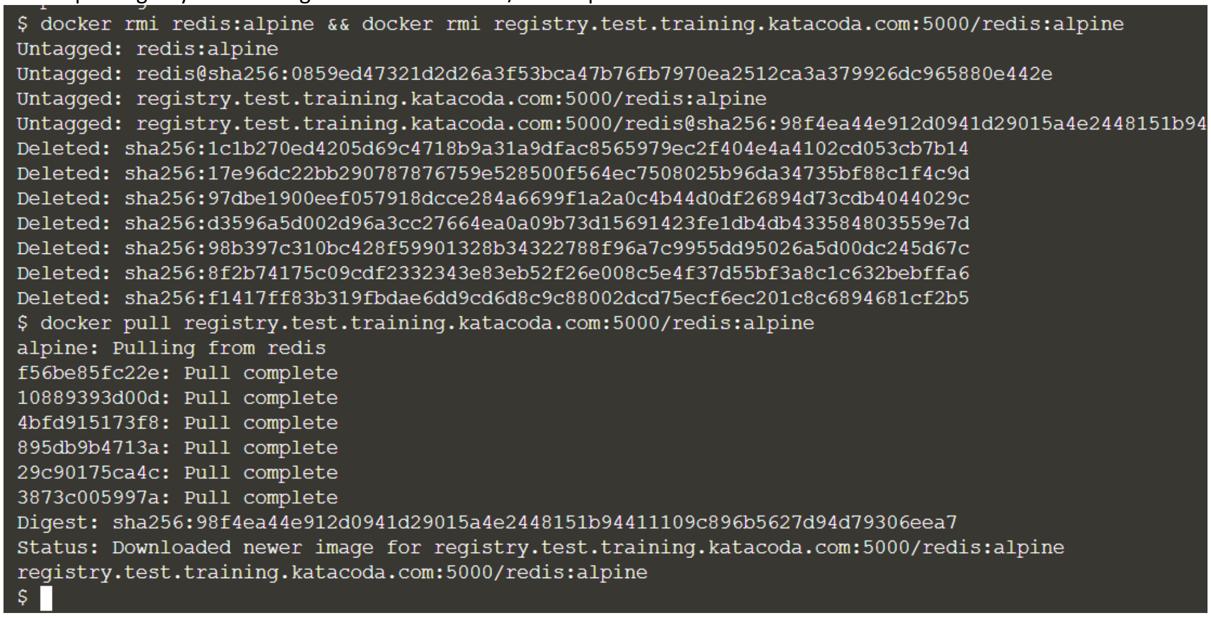
```
$ docker pull redis:alpine
alpine: Pulling from library/redis
f56be85fc22e: Pull complete
10889393d00d: Pull complete
4bfd915173f8: Pull complete
895db9b4713a: Pull complete
29c90175ca4c: Pull complete
3873c005997a: Pull complete
Digest: sha256:0859ed47321d2d26a3f53bca47b76fb7970ea2512ca3a379926dc965880e442e
Status: Downloaded newer image for redis:alpine
docker.io/library/redis:alpine
$ docker tag redis:alpine registry.test.training.katacoda.com:5000/redis:alpine
$ docker push registry.test.training.katacoda.com:5000/redis:alpine
The push refers to repository [registry.test.training.katacoda.com:5000/redis]
9aa091859c8d: Pushed
9b5f5be88464: Pushed
461652448265: Pushed
f566a03a38b9: Pushed
a2138152cd2d: Pushed
f1417ff83b31: Pushed
alpine: digest: sha256:98f4ea44e912d0941d29015a4e2448151b94411109c896b5627d94d79306eea7 size: 1571
$
```

Step 5 - Pulling Images
First, remove images so they need to be pulled again

docker rmi redis:alpine && docker rmi registry.test.training.katacoda.com:5000/redis:alpine

As with push, pulling also includes the URL of our target Registry.

docker pull registry.test.training.katacoda.com:5000/redis:alpine



```
$ docker rmi redis:alpine && docker rmi registry.test.training.katacoda.com:5000/redis:alpine
Untagged: redis:alpine
Untagged: redis@sha256:0859ed47321d2d26a3f53bca47b76fb7970ea2512ca3a379926dc965880e442e
Untagged: registry.test.training.katacoda.com:5000/redis:alpine
Untagged: registry.test.training.katacoda.com:5000/redis@sha256:98f4ea44e912d0941d29015a4e2448151b94
Deleted: sha256:1c1b270ed4205d69c4718b9a31a9dfac8565979ec2f404e4a4102cd053cb7b14
Deleted: sha256:17e96dc22bb290787876759e528500f564ec7508025b96da34735bf88c1f4c9d
Deleted: sha256:97dbe1900eef057918dcce284a6699f1a2a0c4b44d0df26894d73cdb4044029c
Deleted: sha256:d3596a5d002d96a3cc27664ea0a09b73d15691423fe1db4db433584803559e7d
Deleted: sha256:98b397c310bc428f59901328b34322788f96a7c9955dd95026a5d00dc245d67c
Deleted: sha256:8f2b74175c09cdf2332343e83eb52f26e008c5e4f37d55bf3a8c1c632bebffa6
Deleted: sha256:f1417ff83b319fbdae6dd9cd6d8c9c88002dcd75ecf6ec201c8c6894681cf2b5
$ docker pull registry.test.training.katacoda.com:5000/redis:alpine
alpine: Pulling from redis
f56be85fc22e: Pull complete
10889393d00d: Pull complete
4bfd915173f8: Pull complete
895db9b4713a: Pull complete
29c90175ca4c: Pull complete
3873c005997a: Pull complete
Digest: sha256:98f4ea44e912d0941d29015a4e2448151b94411109c896b5627d94d79306eea7
Status: Downloaded newer image for registry.test.training.katacoda.com:5000/redis:alpine
registry.test.training.katacoda.com:5000/redis:alpine
$
```