Examples using the Docker Engine SDKs and Docker API

Estimated reading time: 22 minutes

After you install Docker (https://docs.docker.com/install/), you can install the Go and Python SDKs (https://docs.docker.com/develop/sdk/#install-the-sdks) and also try out the Docker Engine API.

Each of these examples show how to perform a given Docker operation using the Go and Python SDKs and the HTTP API using `curl` .

Run a container

This first example shows how to run a container using the Docker API. On the command line, you would use the `docker run` command, but this is just as easy to do from your own apps too.

This is the equivalent of typing `docker run alpine echo hello world` at the command prompt:

Go | Python | HTTP

```
import docker
client = docker.from_env()
print client.containers.run("alpine", ["echo", "hello", "world"])
```

Run a container in the background

You can also run containers in the background, the equivalent of typing `docker run -d bfirsh/reticulate-splines` :

Go | Python | HTTP

```
import docker
client = docker.from_env()
container = client.containers.run("bfirsh/reticulate-splines", detach=True)
print container.id
```

List and manage containers

You can use the API to list containers that are running, just like using `docker ps`:

Go | Python | HTTP

```
import docker
client = docker.from_env()
for container in client.containers.list():
  print container.id
```

Stop all running containers

Now that you know what containers exist, you can perform operations on them. This example stops all running containers.

> **Note**: Don't run this on a production server. Also, if you are using swarm services, the containers stop, but Docker creates new ones to keep the service running in its configured state.

Go | Python | HTTP

```
import docker
client = docker.from_env()
for container in client.containers.list():
  container.stop()
```

Print the logs of a specific container

You can also perform actions on individual containers. This example prints the logs of a container given its ID. You need to modify the code before running it to change the hard-coded ID of the container to print the logs for.

Go | Python | HTTP

```
import docker
client = docker.from_env()
container = client.containers.get('f1064a8a4c82')
print container.logs()
```

List all images

List the images on your Engine, similar to `docker image ls`:

Go | Python | HTTP

```
import docker
client = docker.from_env()
for image in client.images.list():
  print image.id
```

Pull an image

Pull an image, like `docker pull`:

Go | Python | HTTP

```
import docker
client = docker.from_env()
image = client.images.pull("alpine")
print image.id
```

Pull an image with authentication

Pull an image, like `docker pull`, with authentication:

> **Note**: Credentials are sent in the clear. Docker's official registries use HTTPS. Private registries should also be configured to use HTTPS.

Go | Python | HTTP

The Python SDK retrieves authentication information from the credentials store (https://docs.docker.com/engine/reference/commandline/login/#credentials-store) file and integrates with credential helpers (https://github.com/docker/docker-credential-helpers). It is possible to override these credentials, but that is out of scope for this Getting Started guide. After using `docker login`, the Python SDK uses these credentials automatically.

```
import docker
client = docker.from_env()
image = client.images.pull("alpine")
print image.id
```

Commit a container

Commit a container to create an image from its contents:

Go | Python | HTTP

```
import docker
client = docker.from_env()
container = client.containers.run("alpine", ["touch", "/helloworld"], detach=T
container.wait()
image = container.commit("helloworld")
print image.id
```

developing (https://docs.docker.com/glossary/?term=developing), api (https://docs.docker.com/glossary/?term=api), sdk (https://docs.docker.com/glossary/?term=sdk), developers (https://docs.docker.com/glossary/?term=developers), rest (https://docs.docker.com/glossary/?term=rest), curl (https://docs.docker.com/glossary/?term=curl), python (https://docs.docker.com/glossary/?term=python), go (https://docs.docker.com/glossary/?term=go)