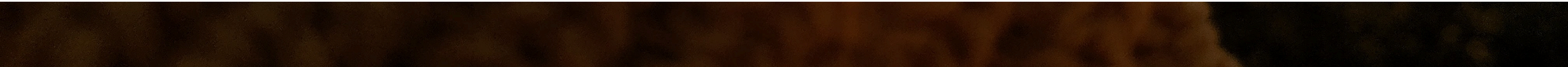
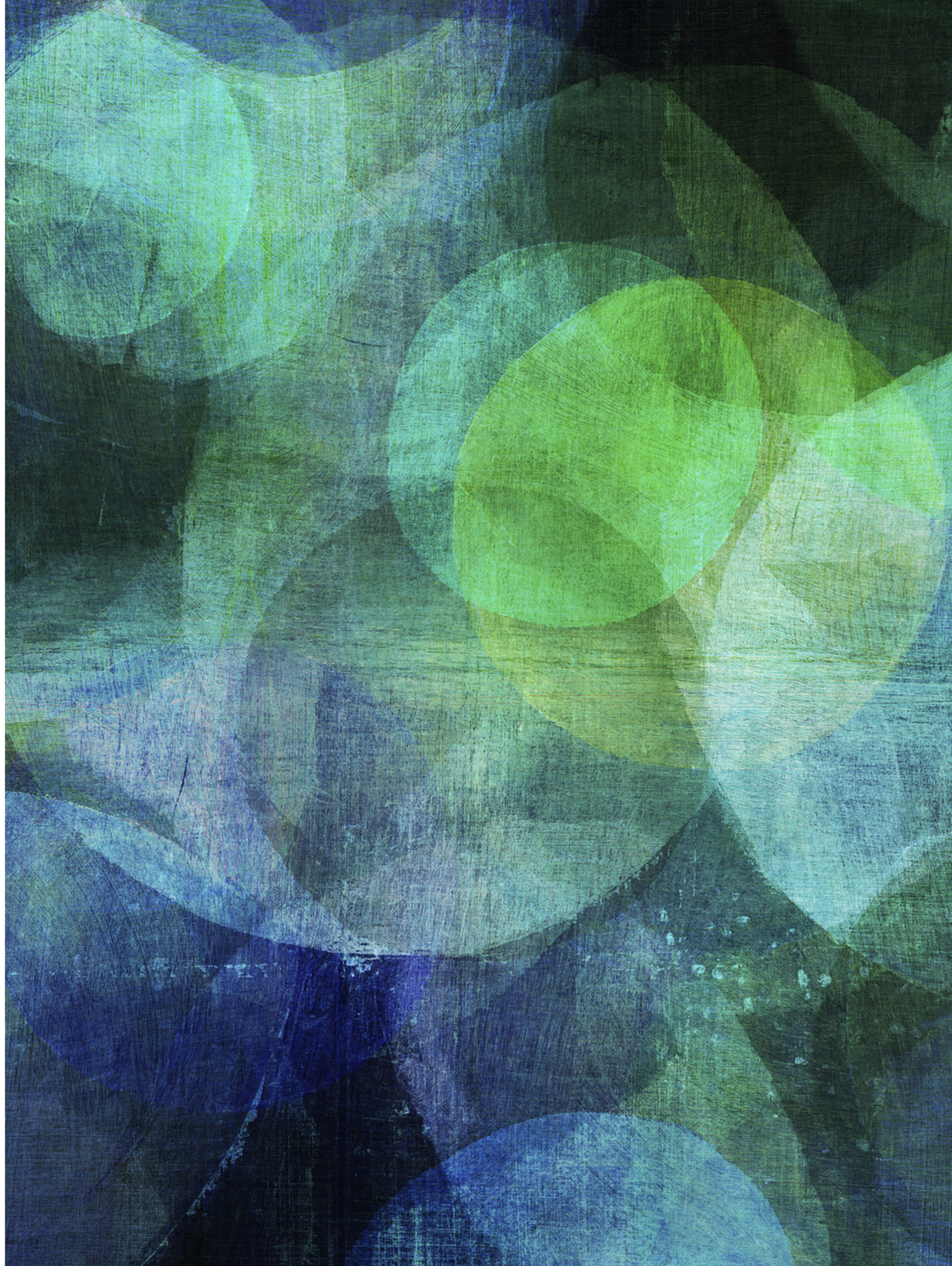
INTRODUCTION TO DOCKER

INTRODUCTION TO DOCKER

Docker Image, Container and Registry

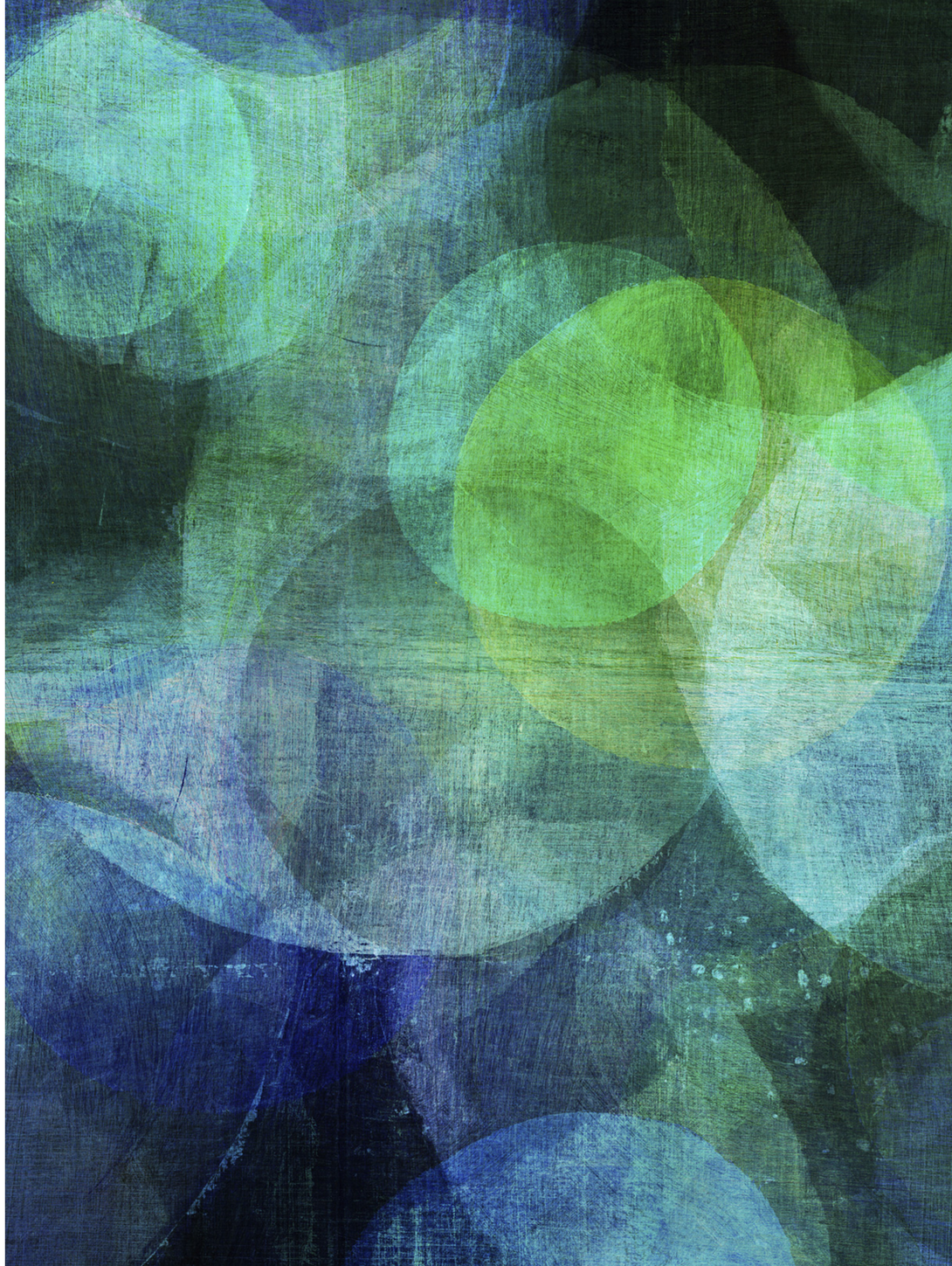


AGENDA



AGENDA

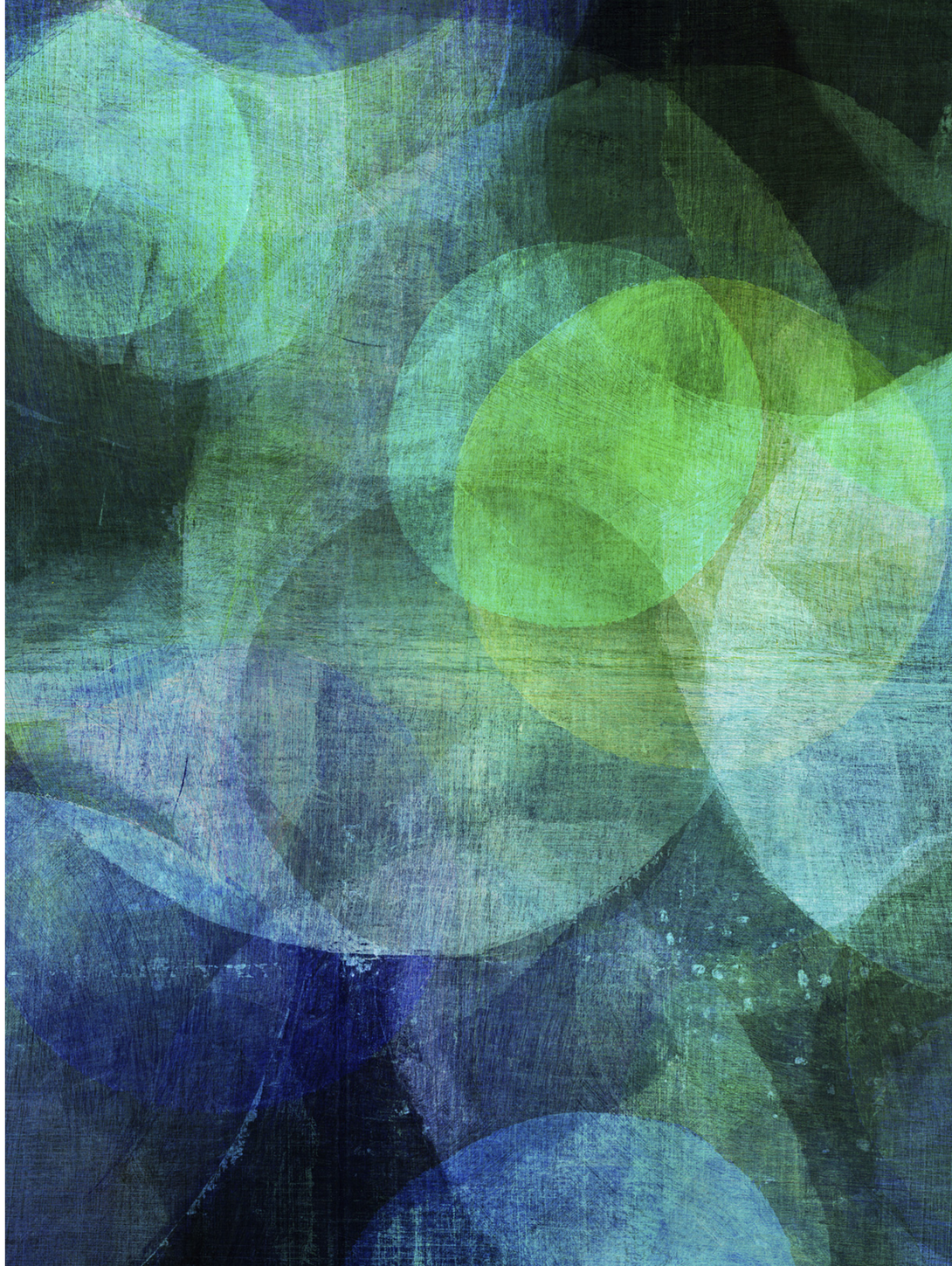
- Hello Docker



AGENDA

- Hello Docker

-Docker-CLI

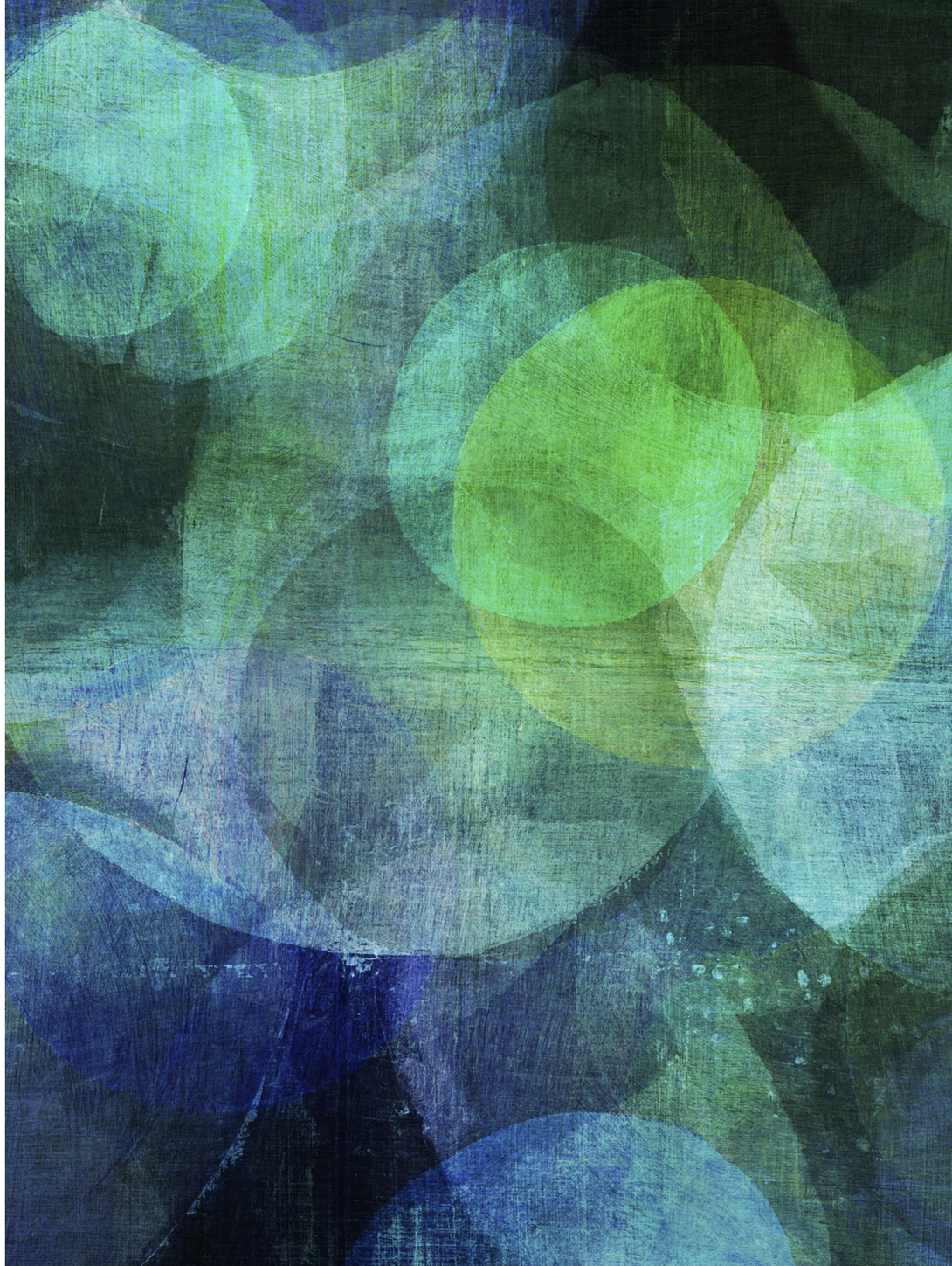


AGENDA

- Hello Docker

-Docker-CLI

-Docker Engine



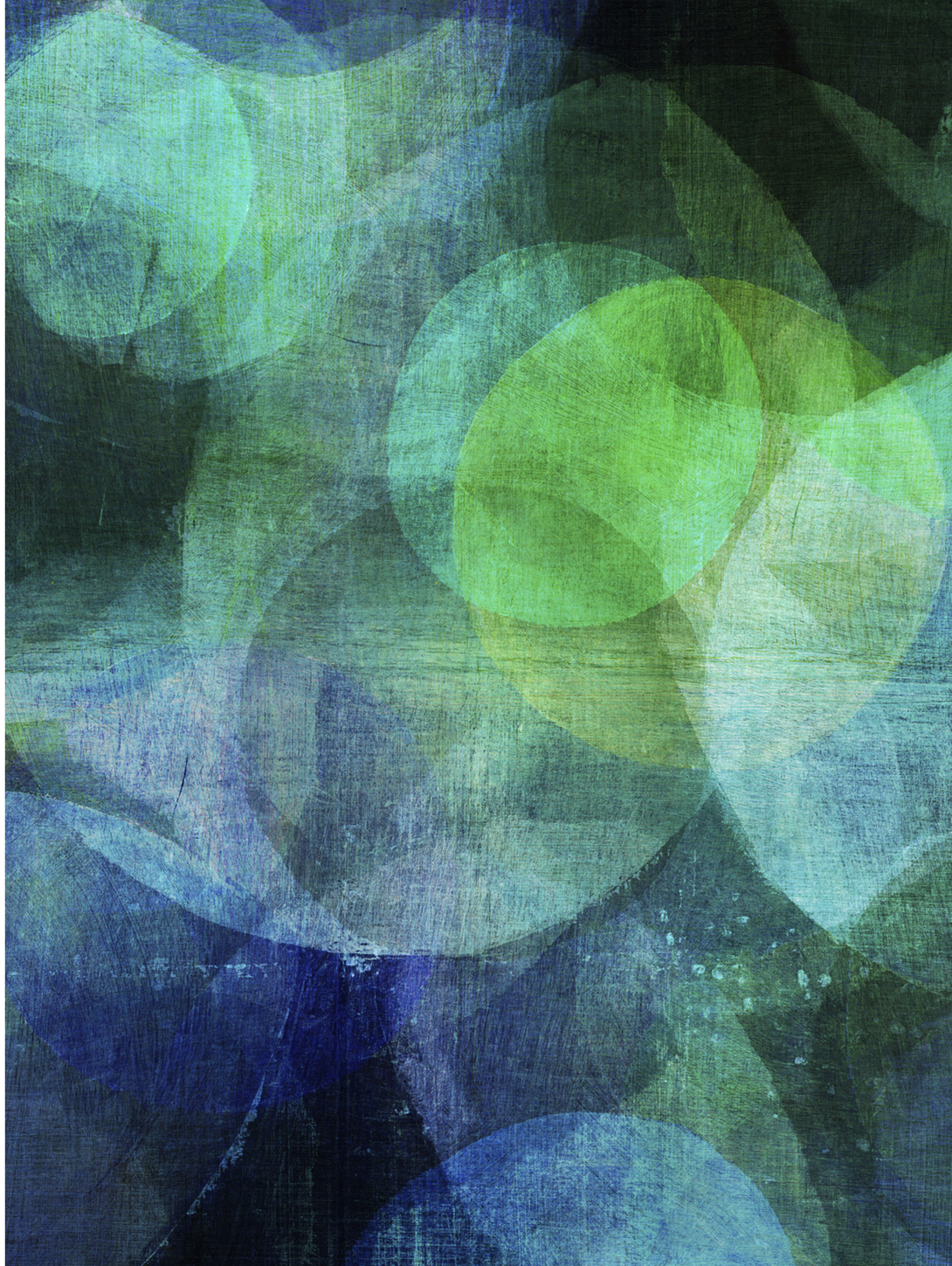
AGENDA

- Hello Docker

-Docker-CLI

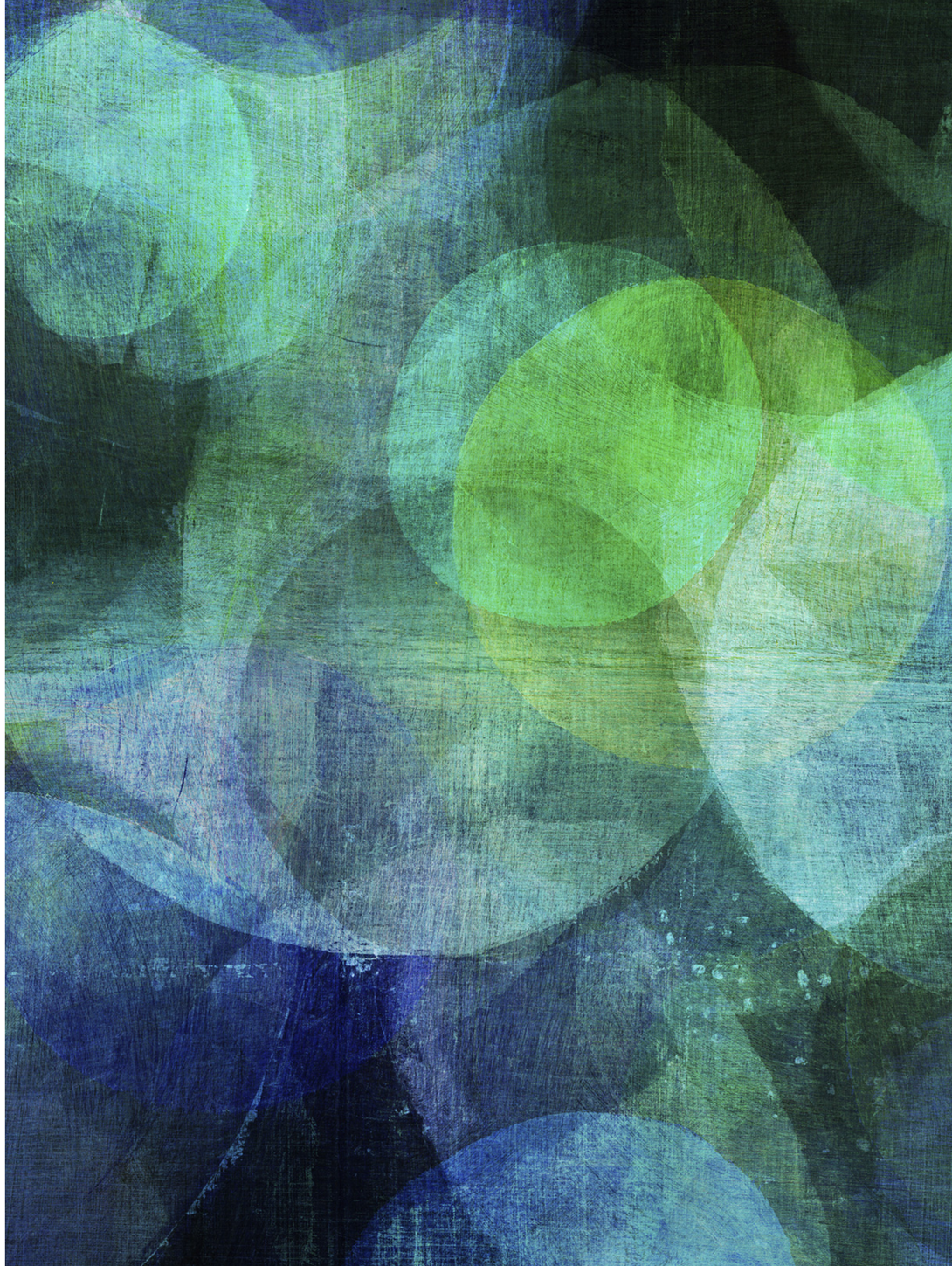
-Docker Engine

- Docker Image



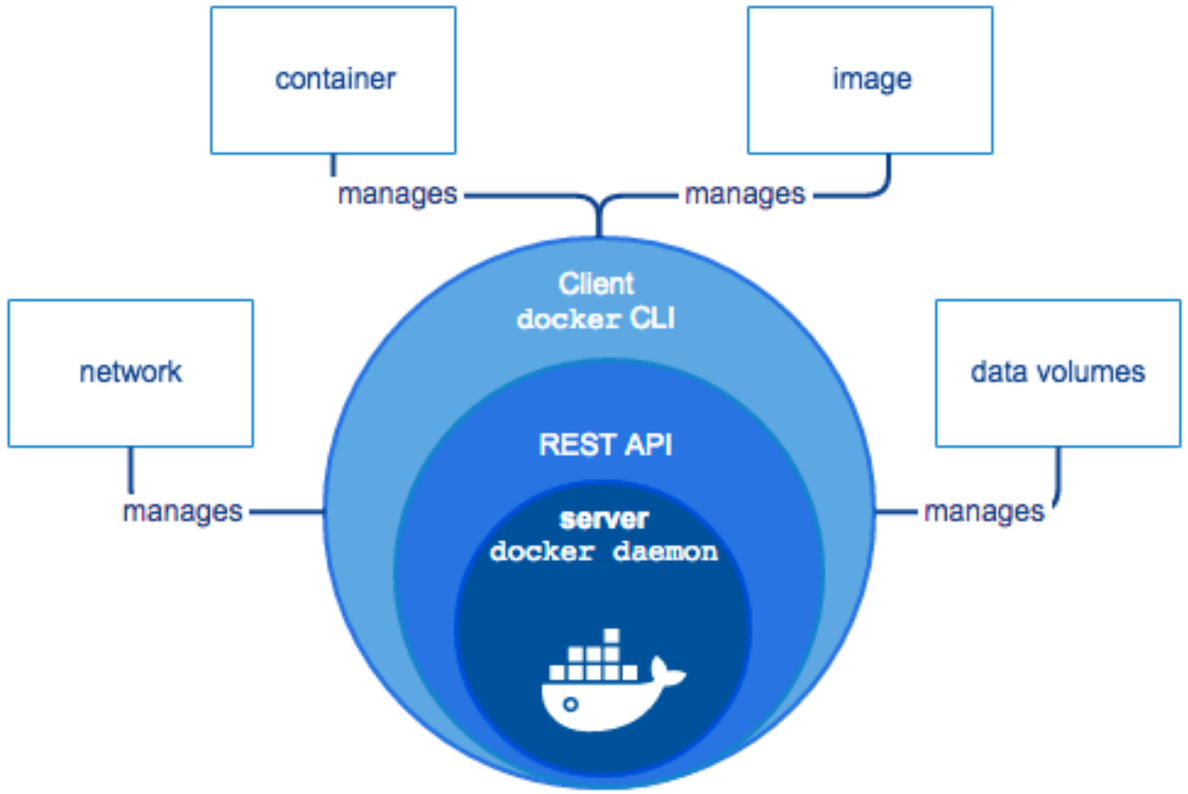
AGENDA

- Hello Docker
- Docker-CLI
- Docker Engine
- Docker Image
- Docker Container



DOCKER CLI
(COMMAND LINE INTERFACE)

DOCKER ENGINE



Docker Engine is a client-server application with:

- *A server* which is a type of long-running program called a daemon process (the `dockerd` command).
- *A REST API* which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
- *A command line interface (CLI)* client (the `docker` command).

DOCKER-CLI: 'DOCKER' COMMAND

```
C:\Users\karan>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default
                           "C:\\Users\\karan\\.docker")
  -c, --context string     Name of the context to use to connect to the
                           daemon (overrides DOCKER_HOST env var and
                           default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal")
                           (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "C:\\Users\\karan\\.docker\\ca.pem")
```

DOCKER-CLI: DEFAULT COMMAND

```
Commands:
  attach      Attach local standard input, output, and error streams to a running
container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
```

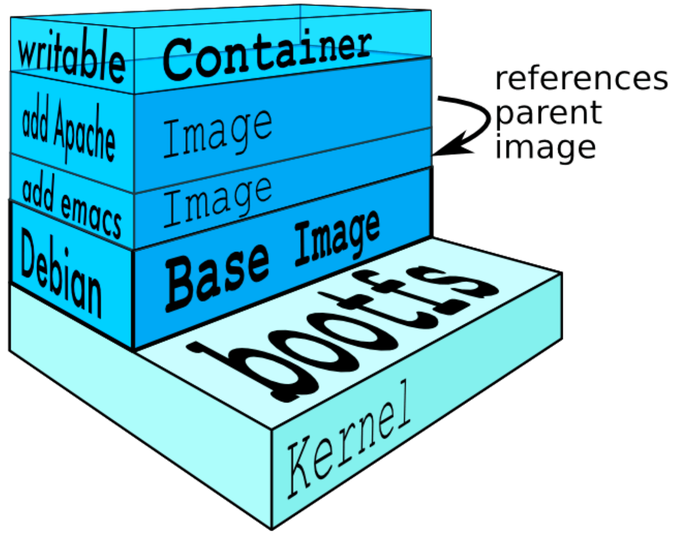
DOCKER-CLI: MANAGEMENT COMMAND

```
Management Commands:
  app*        Docker App (Docker Inc., v0.9.1-beta3)
  builder     Manage builds
  buildx*     Build with BuildKit (Docker Inc., v0.5.1-docker)
  compose*    Docker Compose (Docker Inc., 2.0.0-beta.1)
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  scan*       Docker Scan (Docker Inc., v0.8.0)
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes
```

DOCKER IMAGE

DOCKER IMAGE

An *image* is a read-only template with instructions for creating a Docker container. Often, an image is *based on* another image, with some additional customization. For example, you may build an image which is based on the `Debian` image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.



RUN YOUR FIRST CONTAINER

```
$ docker run hello-world
```

RUN YOUR FIRST CONTAINER

IMAGES

```
$ docker run hello-world
```

RUN YOUR FIRST CONTAINER

docker

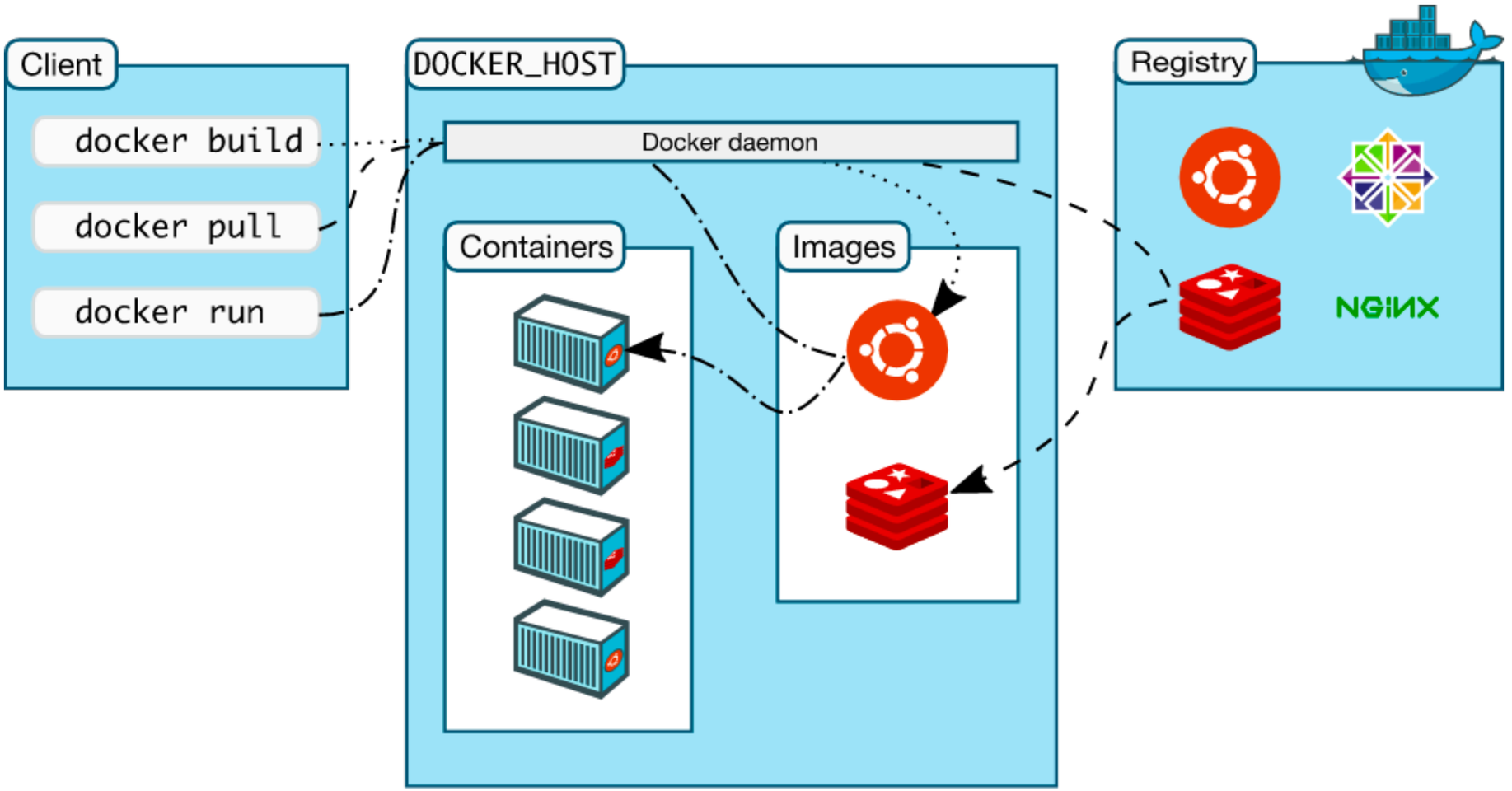
IMAGES


```
$ docker run hello-world
```

RUN YOUR FIRST CONTAINER


```
$ docker run hello-world
```

DOCKER IMAGE



DOCKER-CLI: IMAGE COMMAND

```
Usage:  docker image COMMAND

Manage images

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by defa
ult)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
```

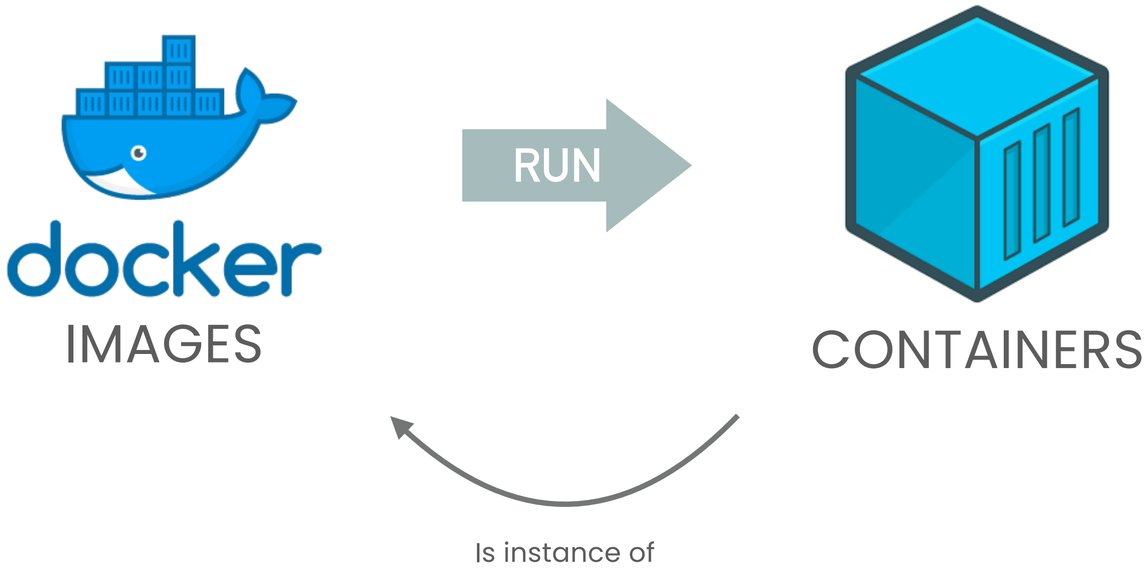
DOCKER CONTAINER

DOCKER CONTAINER

A *container* is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

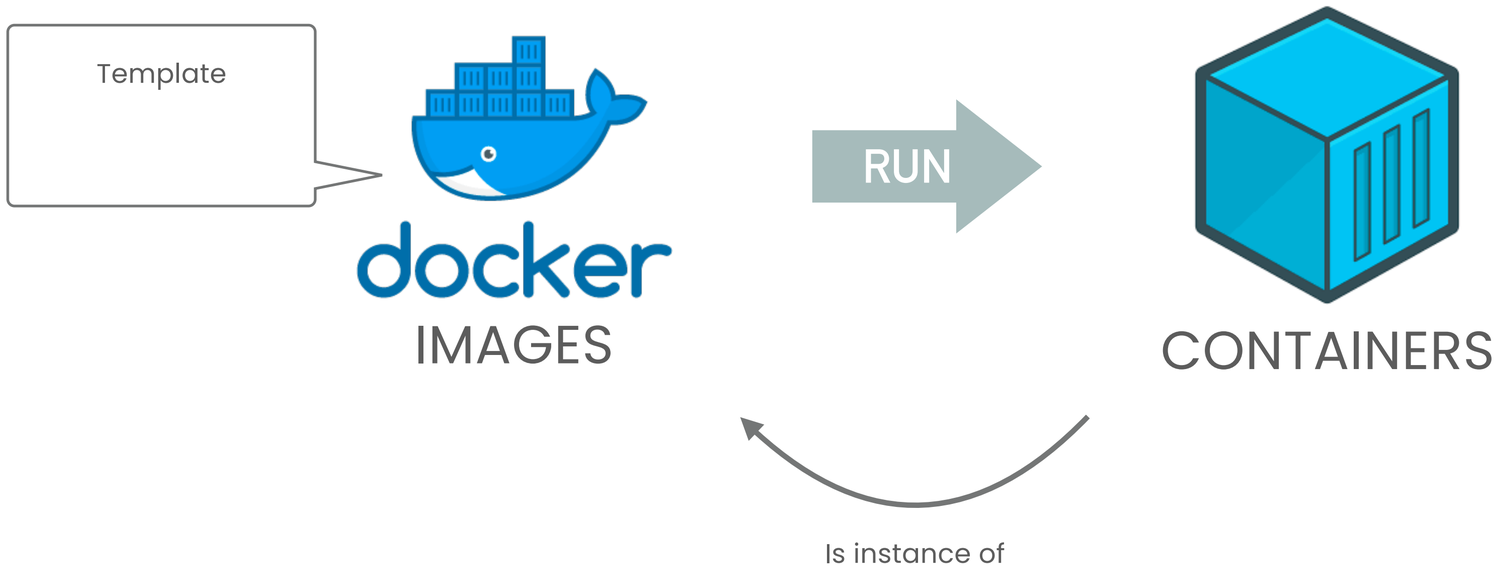
DOCKER CONTAINER

A *container* is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.



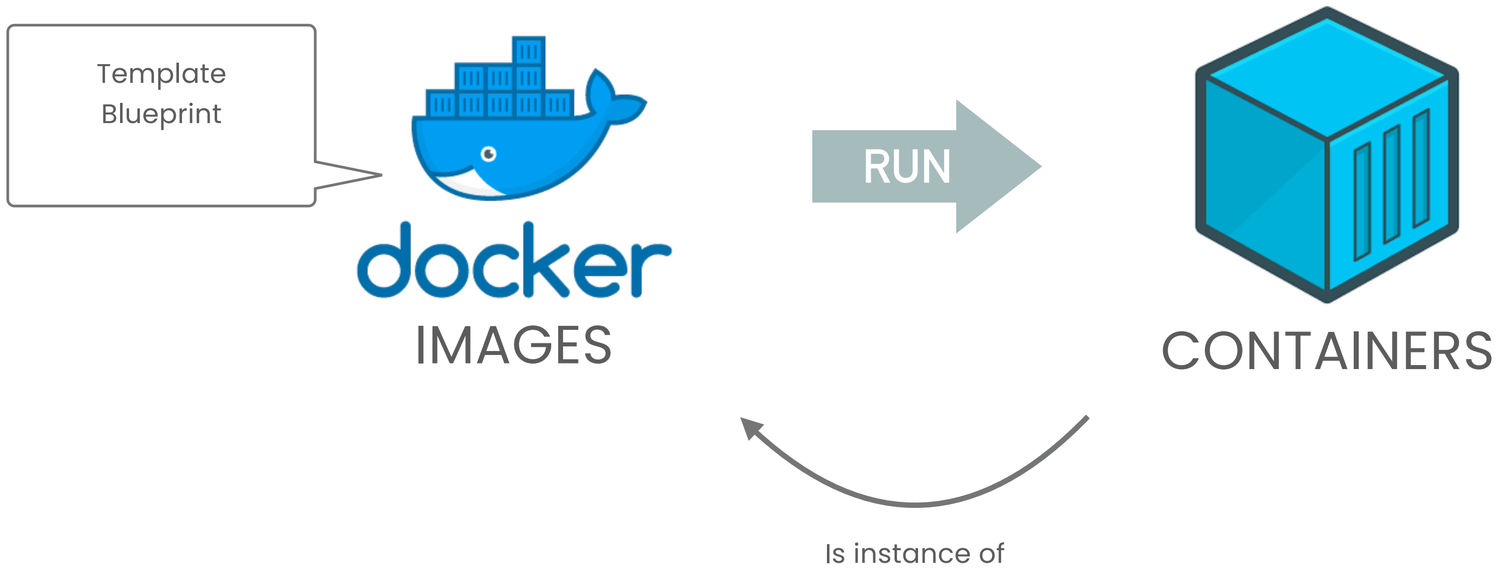
DOCKER CONTAINER

A *container* is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.



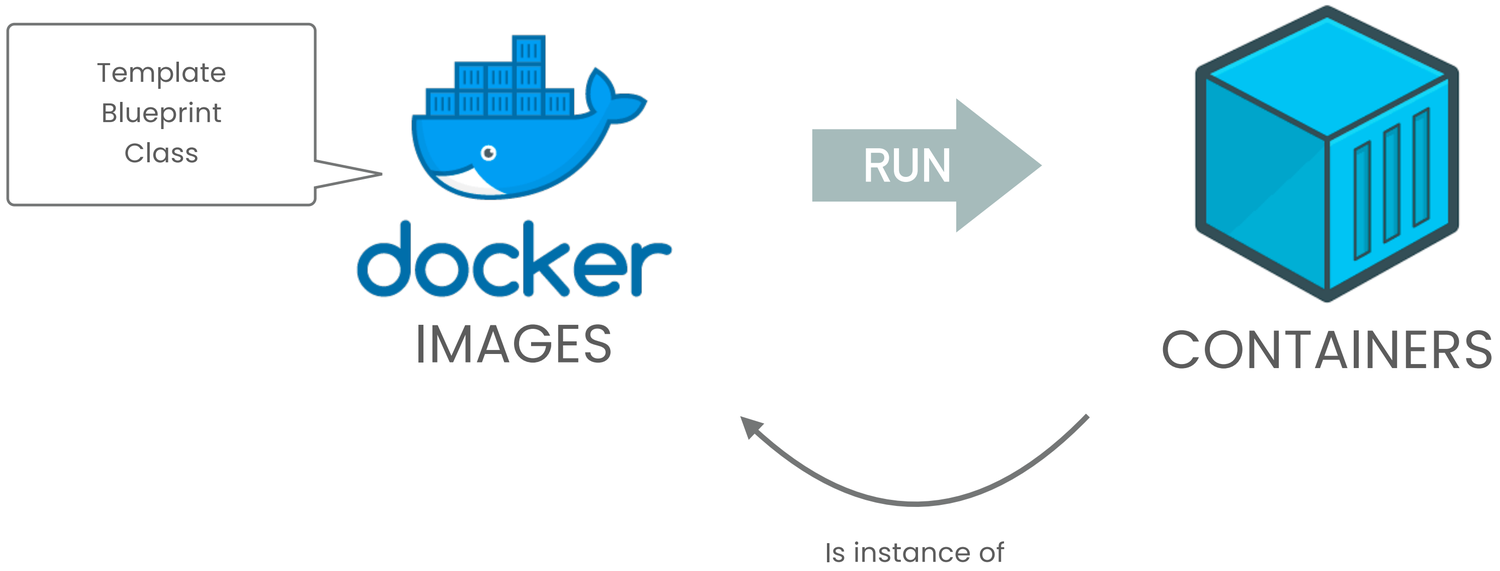
DOCKER CONTAINER

A *container* is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.



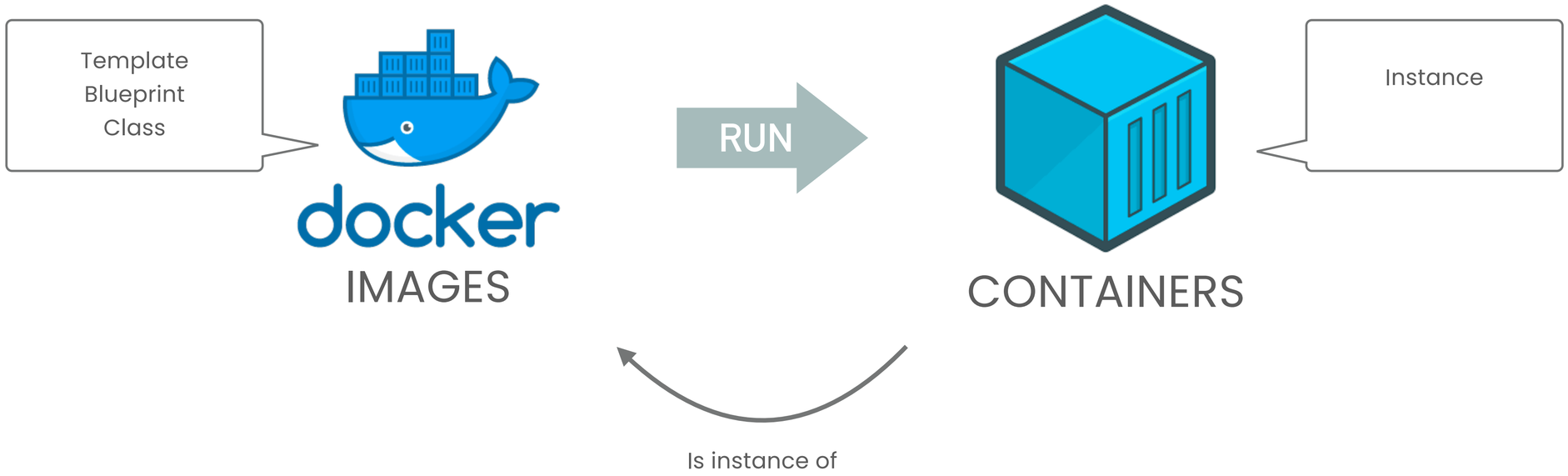
DOCKER CONTAINER

A *container* is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.



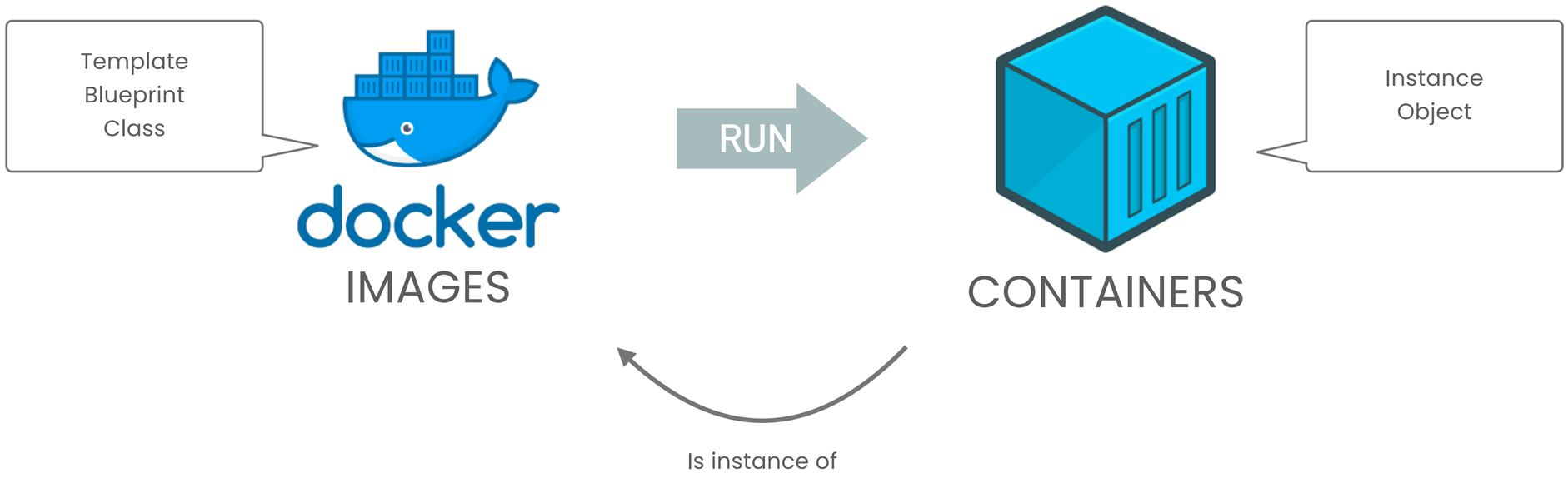
DOCKER CONTAINER

A *container* is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.



DOCKER CONTAINER

A *container* is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

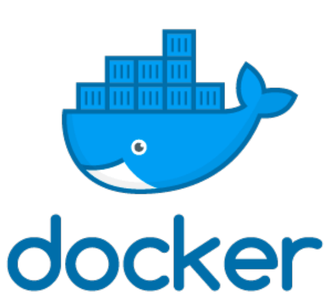


DOCKER IMAGE – CONTAINERS

DOCKER IMAGE - CONTAINERS

DOCKER IMAGE - CONTAINERS

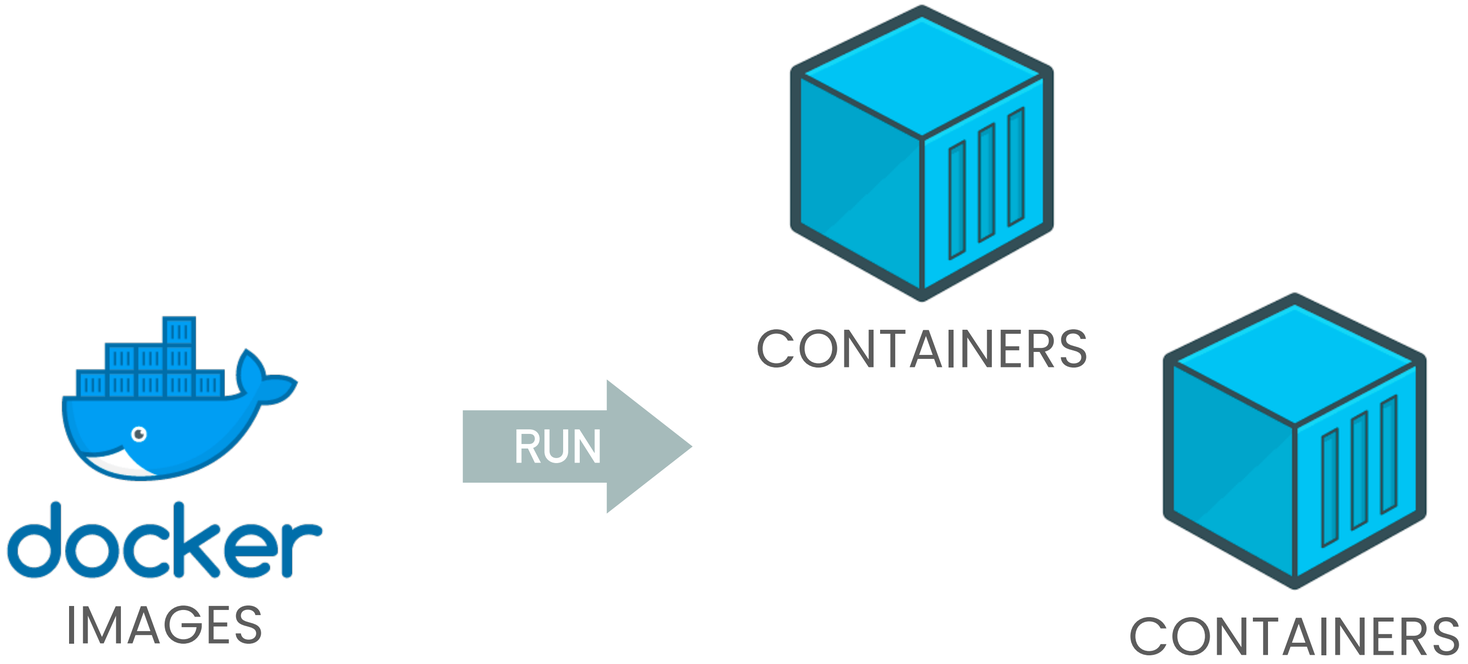
DOCKER IMAGE - CONTAINERS



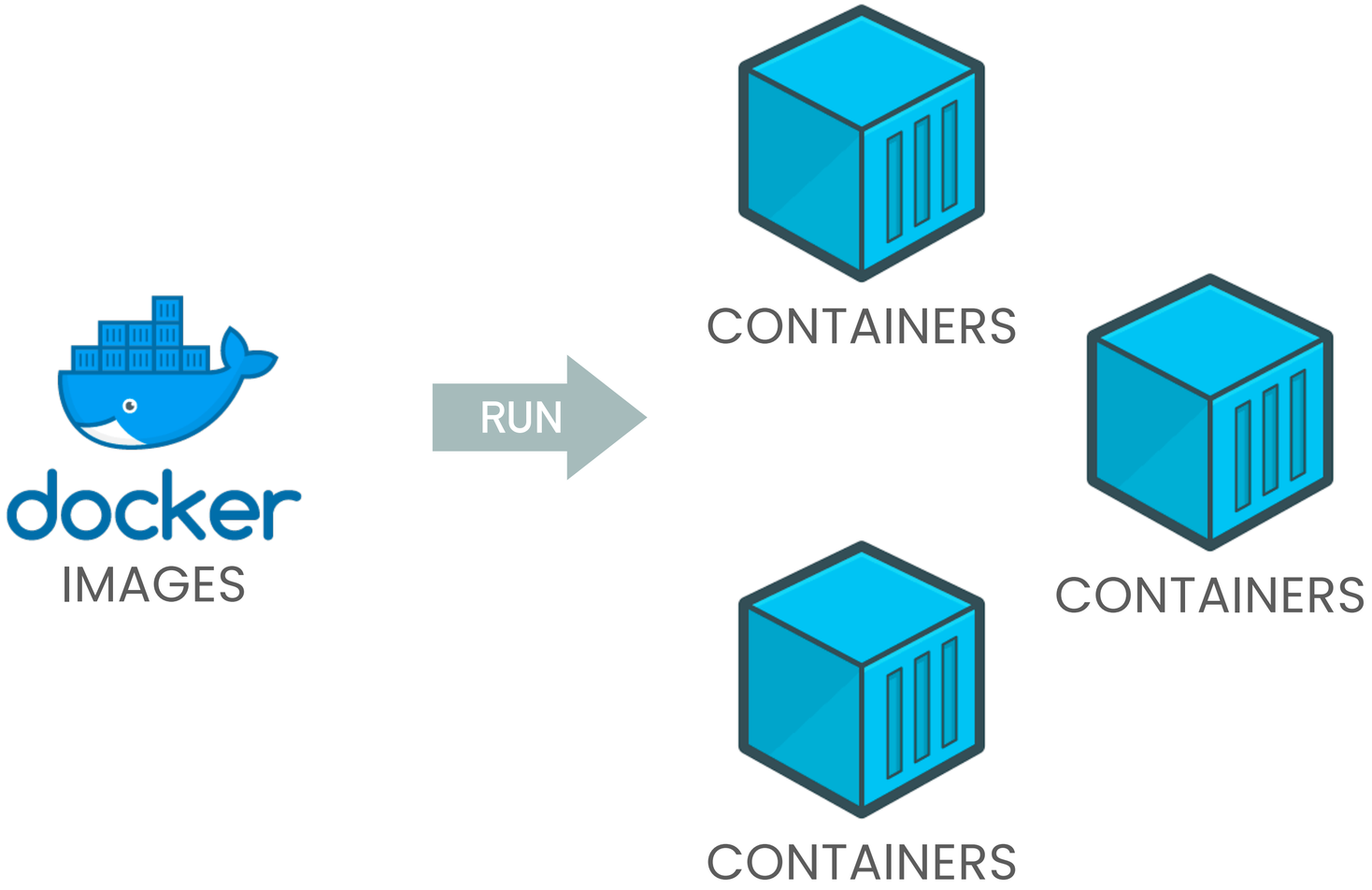
IMAGES

CONTAINERS

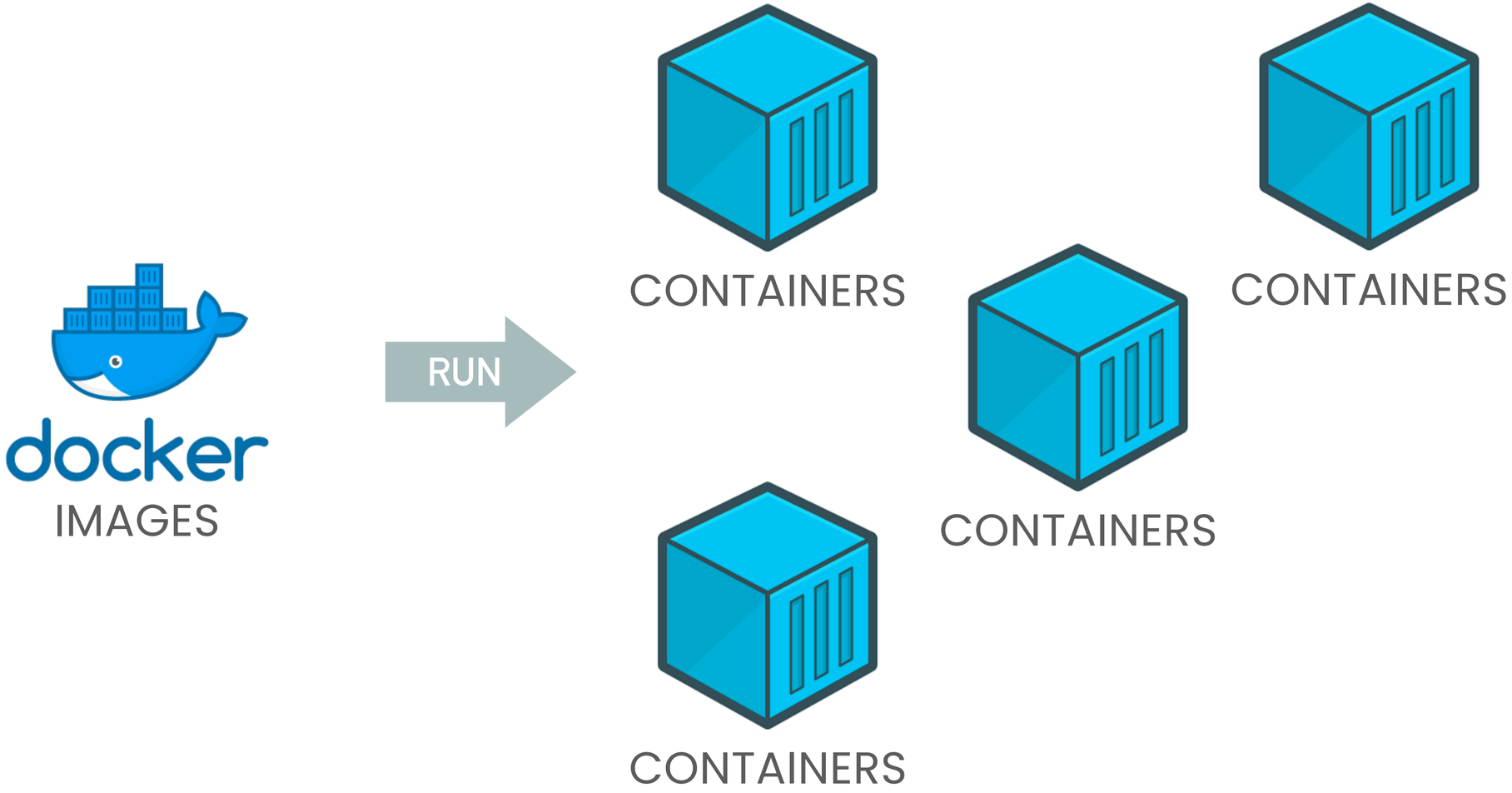
DOCKER IMAGE - CONTAINERS



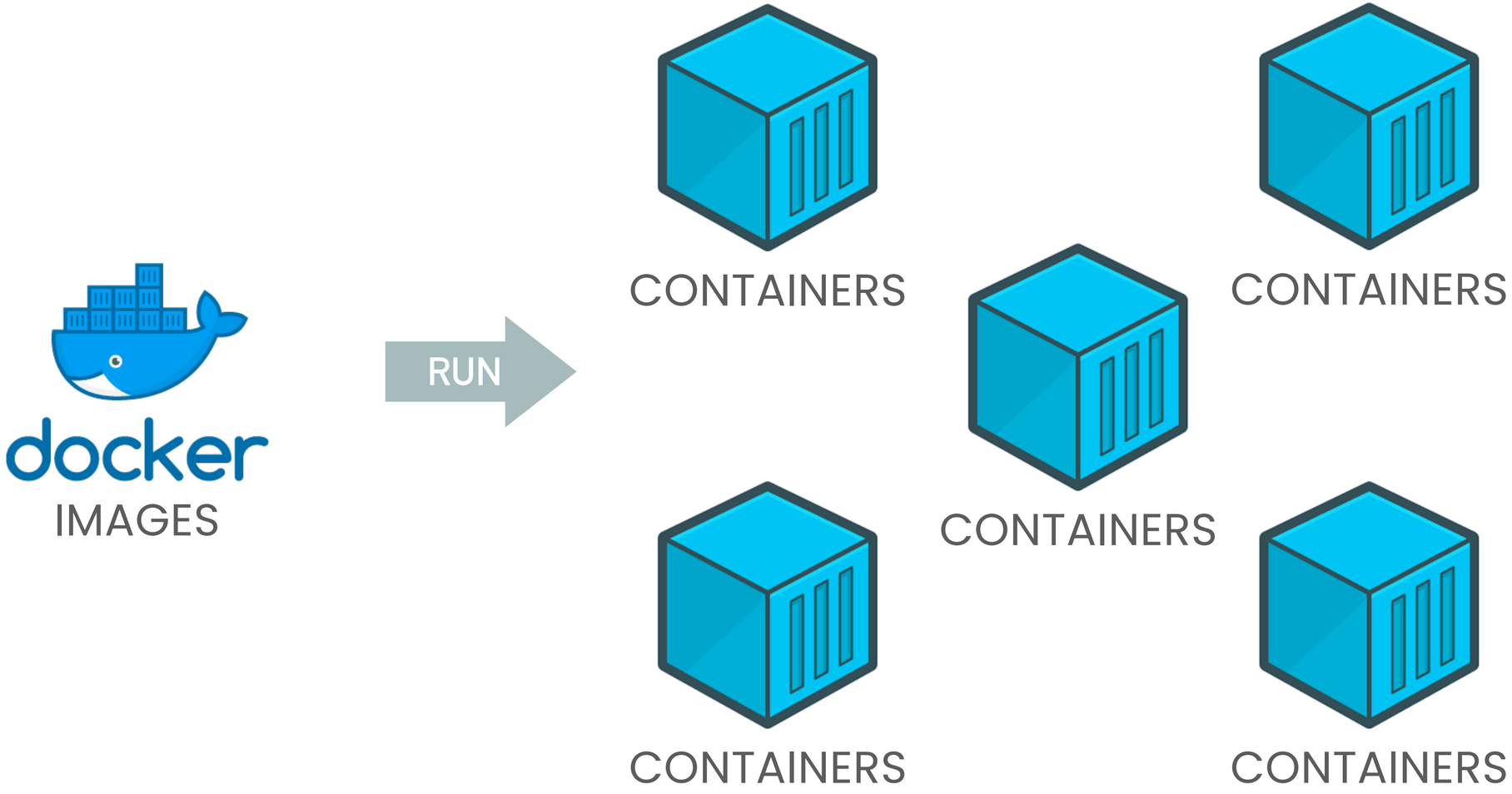
DOCKER IMAGE - CONTAINERS



DOCKER IMAGE - CONTAINERS



DOCKER IMAGE - CONTAINERS



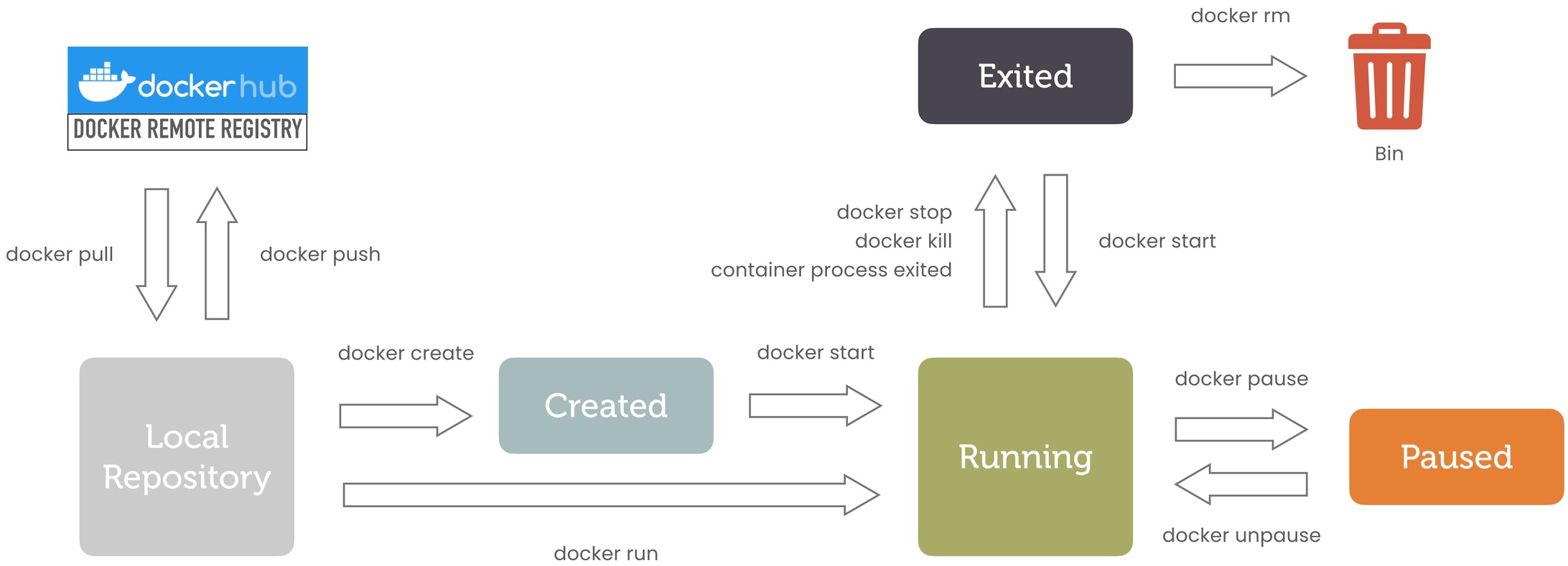
DOCKER-CLI: CONTAINER COMMAND

```
Usage:  docker container COMMAND

Manage containers

Commands:
  attach      Attach local standard input, output, and error streams to a running
container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  inspect     Display detailed information on one or more containers
  kill        Kill one or more running containers
  logs        Fetch the logs of a container
  ls          List containers
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  prune       Remove all stopped containers
```

DOCKER CONTAINER - LIFECYCLE



THANK YOU