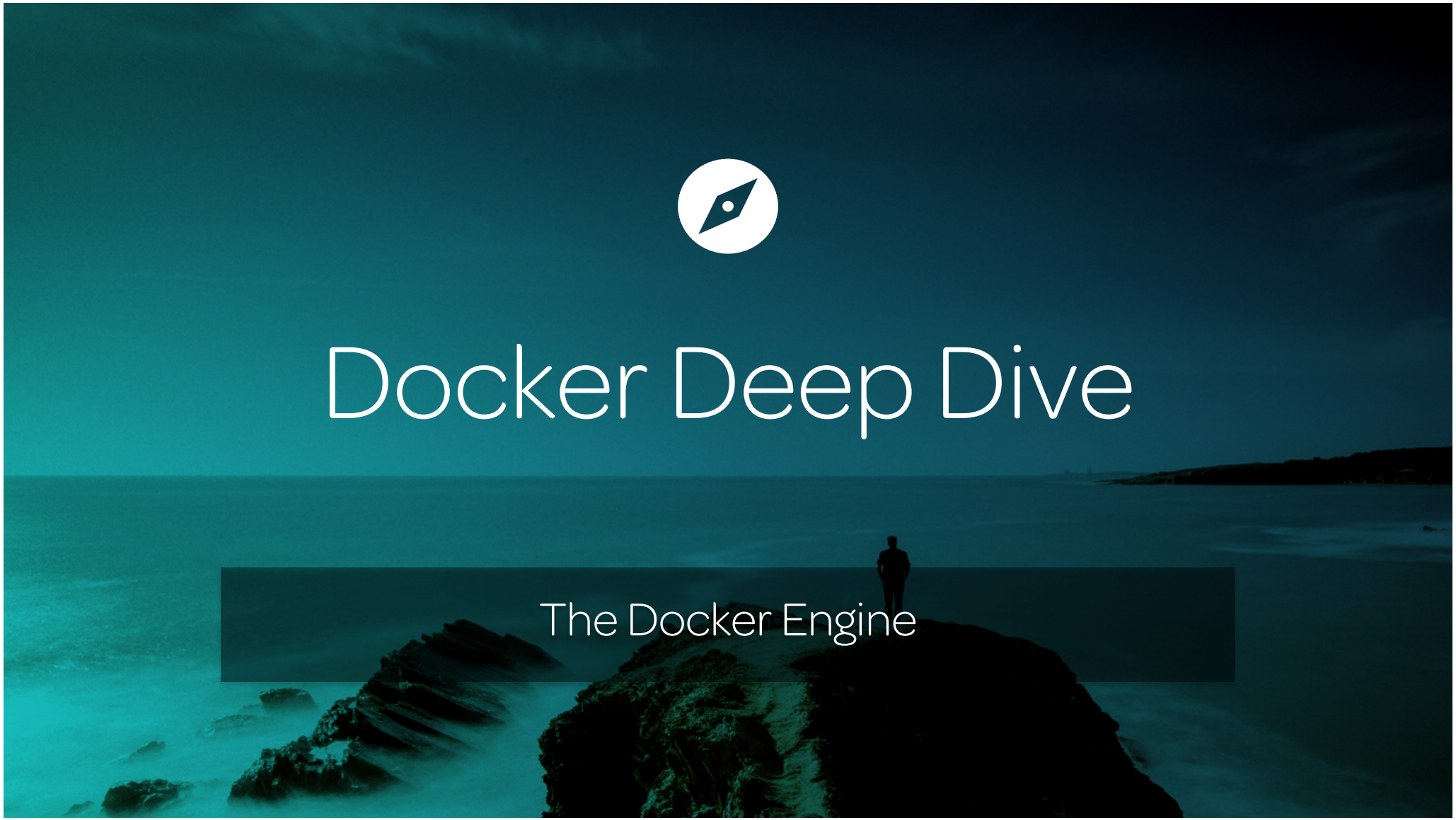
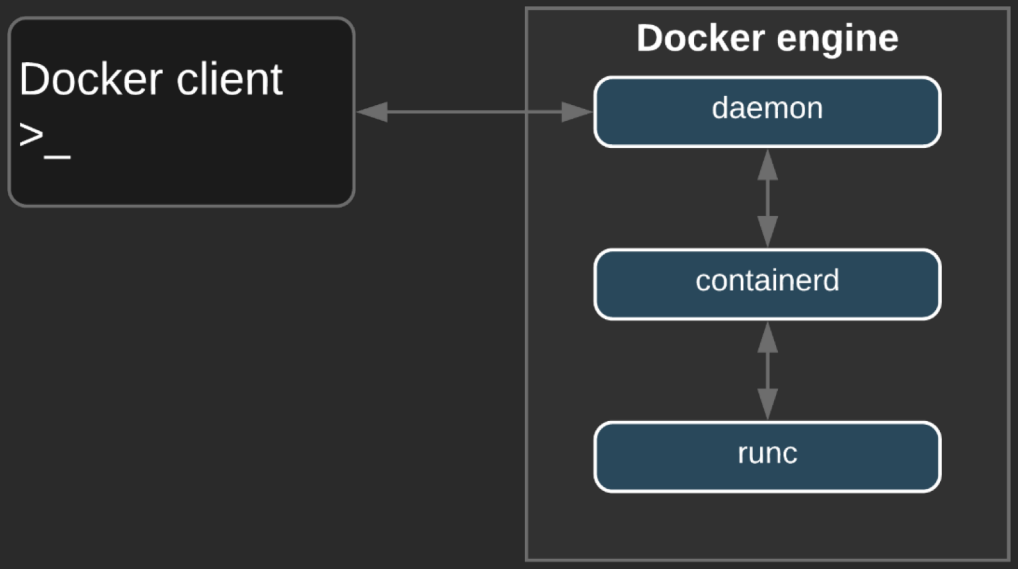

Docker Deep Dive
The Docker Engine

Under The Hood

Docker engine :

- Modular in design:
  - Batteries included but replaceable
- Based on an open-standards outline by the Open Container Initiative
- The major components:
  - Docker client
  - Docker daemon
  - `containerd`
  - `runc`
- The components work together to create and run containers

Under The Hood (cont.)

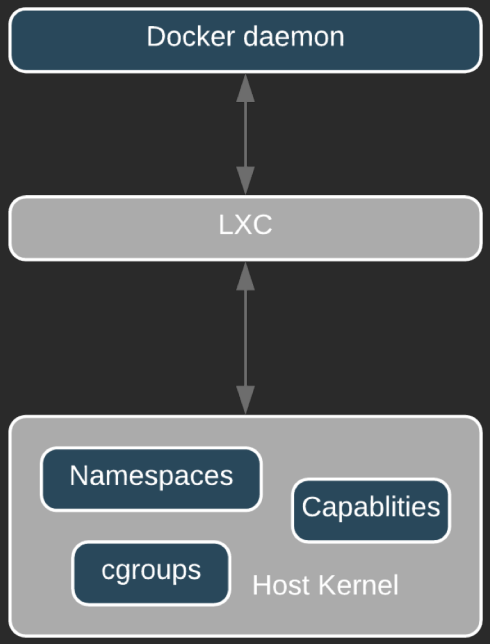


A Brief History of the Docker Engine

The first release of Docker

- The Docker daemon
  - Docker client
  - Docker daemon
  - Monolithic binary
  - Docker client
  - Docker API
  - Container runtime
  - Image builds
  - Much more...
- LXC
  - Namespaces
  - Control groups (cgroups)
  - Linux-specific

A Brief History of the Docker Engine (cont.)



Refactoring of the Docker Engine

LXC was later replaced with libcontainer:

- Docker 0.9
- Platform agnostic

Issues with the monolithic Docker daemon:

- Harder to innovate
- Slow
- Not what the ecosystem wanted

Docker became more modular:

- Smaller more specialized tools
- Pluggable architecture

Refactoring of the Docker Engine (cont.)

Open Container Initiative:

- Image spec
- Container runtime spec
- Version 1.0 released in 2017
- Docker, Inc. heavily contributed
- Docker 1.11 (2016) used the specification as much as possible

Refactoring of the Docker Engine (cont.)

`runc`

- Implementation of the OCI container-runtime-spec
- Lightweight CLI wrapper for libcontainer
- Create containers

`containerd`

- Manage container lifecycle:
  - Start
  - Stop
  - Pause
  - Delete
- Image management
- Part of the 1.11 release

Refactoring of the Docker Engine (cont.)

`shim`

- Implementation of daemonless Containers
- `containerd` forks an instance of `runc` for each new container
- `runc` process exits after the container is created
- `shim` process becomes the container parent
- Responsible for:
  - STDIN and STDOUT
  - Reporting exit status to the Docker daemon

Running Containers

```
docker container run -it --name <NAME> <IMAGE>:<TAG>
```

Creating a container:

- Use the CLI to execute a command.
- Docker client uses the appropriate API payload.
- The command POSTs to the correct API endpoint.
- The Docker daemon receives instructions.
- The Docker daemon calls `containerd` to start a new container.
- The Docker daemon uses `gRPC`, a CRUD style API.
- `containerd` creates an OCI bundle from the Docker image.
- Tells `runc` to create a container using the OCI bundle
- `runc` interfaces with the OS kernal to get the constructs needed to create a container

Running Containers (cont.)

Creating a container:

- This includes namespaces, `cgroups`, etc.
- The container process is started as a child process
- Once the container starts `runc` will exit
- Which completes the process and the container is now running

Process Summary