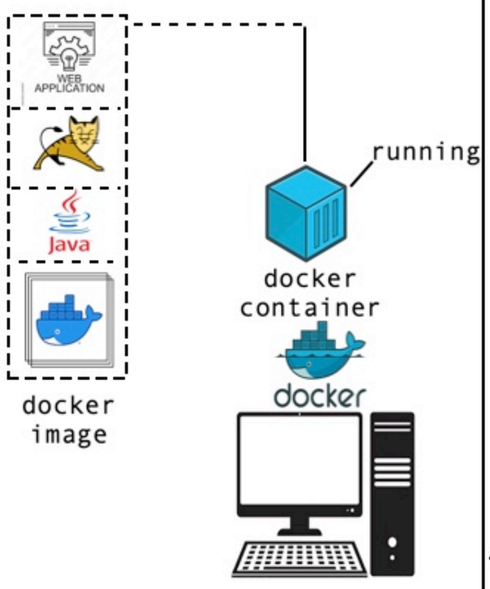

The developer has to implement in every application an HEALTHCHECK endpoint, that performs pre-liminary or basic checks on verifying the application and its dependent resources are healthy or not like
1. database
2. backend servers
3. any external services or resources on which our application is dependent on
and reports the status of the application with statuscode: 200 or ERRORCODE if it is an web application

The running status of the container doesnt guarantee the application that is packaged inside the container is HEALTHY. There could be several reasons where the container might be reported as RUNNING, but still the underlying application that is packaged inside the container may not be running.

Incase when we are packaging and deploying an web application on a web server like tomcat etc, since these servers are long-running processes and runs in daemon/background processes, even the underlying applications that are deployed on these web servers fails also still the container reports its status as running like

1. the deployment of the war application might have been failed during the startup of the server
2. that tomcat startup itself might have been failed but the container would be blocked for running at TTY
3. during the execution the underlying application might have been crashed due to various reasons

In any of the above cases still the container would be running and reports the status: RUNNING only which is false. This leads to unnecessary consumption of cpu/memory and system resources that leads DEAD application containers and in-availability of the application

To rescue us from these problems in identifying such UN-HEALTH application containers, the docker has introduced HEALTHCHECK Directive. which checks the HEALTH of the underlying containerized application and reports the HEALTH status of it.

HEALTHCHECK directive

syntax:-
HEALTHCHECK COMMAND

The docker engine upon launching the container would periodically execute the HEALTHCHECK COMMAND we supplied and capture the exitcode of the COMMAND
if the exitcode returned by the COMMAND is
0 - success
non-zero = is considered a failed
and reports the status of the container accordingly

We need to configure HEALTHCHECK directive in Dockerfile written with appropriate COMMANDS that needs to be executed for checking the HEALTH of the application and capture the exitcode to report the status

OPTIONS in HEALTHCHECK directive
1. --start-period=MS
we can configure the interval delay upon the startup of container, to perform HEALTHCHECK requests.

docker container run airtel2:1.0
if the docker engine starts performing healthchecks immediately after starting the container, before the underlying application is running, all the HEALTHCHECK COMMANDS results in faiure and eventually leads to TERMINATION of the container. To avoid this problem we can configure dealyed-interval indicating how long docker engine has to wait upon the container has been reported as RUNNING to begin performing HEALTHCHECKS.

2. --interval=DURATION
how frequently the HEALTHCHECK COMMAND to be executed to verify the HEALTH

3. --retries=N
retry indicates how many times the docker engine should execute the HEALTHCHECK COMMAND before report the application status as: failed

4. --timeout=MS
The time in milliseconds the docker engine should wait for response from the HEALTHCHECK command before it marks as failed.

HEALTHCHECK [--OPTIONS] COMMAND