**1. Run the container as a non-root user**

**2. Remove unnecessary packages/software from the image**

**3. Scan and rebuild images to include security patches**

Whether you are building your Docker image from scratch or you are building them on top of the third-party image, it is very important to have your images scanned for vulnerabilities as part of your continuous integration process. This includes scanning included packages, binaries, libraries, files, etc. against one or more well-known vulnerabilities databases.

**4. Enable Docker Content Trust (DCT)**

If DCT is enabled pull, run, and build instructions will only work with the trusted (signed) images.

echo $DOCKER\_CONTENT\_TRUST

**5. Use COPY instead of ADD in Dockerfile**

Using ADD instruction in the Dockerfile introduces the risk of adding malicious files from

**6. Do not store any secret in Dockerfile**

**7. Install verified packages and use trusted base images**

**8. Remove setuid and setgid permission from the image**

**9. Carefully use update instruction in Dockerfile**

If you use package update instruction like apt-get update or apk update (depending on the base image and package manager) in your Docker file, make sure that they are never present alone in a single Dockerfile line.

If the package update instruction is present in a single line in the Dockerfile, then the same cached update layer will be used. This will prevent any fresh update to be part of subsequent builds.

**10. Add HEALTHCHECK to Docker image**

Based on the health of the containers Docker can provision new containers to replace the unhealthy containers.

HEALTHCHECK --interval=5m --timeout=3s \  
 CMD curl <https://localhost:8443/version> -k || exit 1