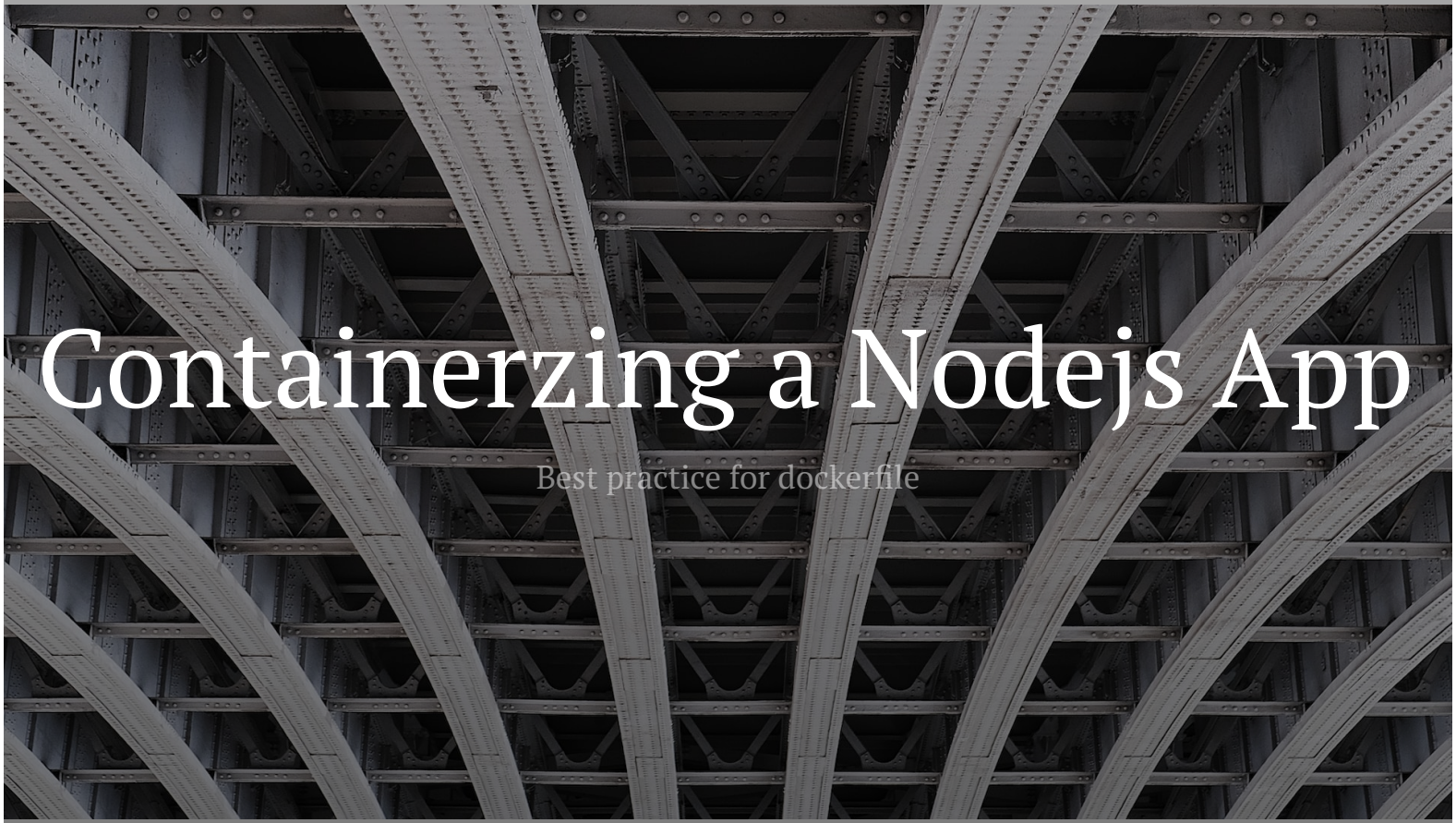


Containerzing a Nodejs App

Best practice for dockerfile

Jain Ramchurn

Linux and Open-source enthusiast

Platform Engineer at Ringier South Africa

Admin of Linux Mirrors in Mauritius

🌐 zain.mu

🐧 EdogawaZain

server.js

```
const express = require('express');
const app = express();
const port = 8080;

app.get('/', (_, res) => {
  res.send('Hello World!');
})

app.listen(port, () => {
  console.log(`Listening on port ${port}`);
})
```

Directory Structure

```
.git
.gitignore
.README.md
node_modules
package.json
package-lock.json
server.js
```

DockerFile

```
FROM node:20

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies and bundle app source
COPY . .

RUN npm ci
EXPOSE 8080
CMD ["node", "server.js"]
```

.dockerignore

```
.git
.gitignore
README.md
node_modules
```

Seperating modules and app code

```
FROM node:20

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
COPY packages*.json .

RUN npm ci

COPY server.js .
EXPOSE 8080
CMD ["node", "server.js"]
```

Run as non-root user

```
FROM node:20
USER node

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
COPY --chown=node:node packages*.json .

RUN npm ci

COPY --chown=node:node server.js .
EXPOSE 8080
CMD ["node", "server.js"]
```

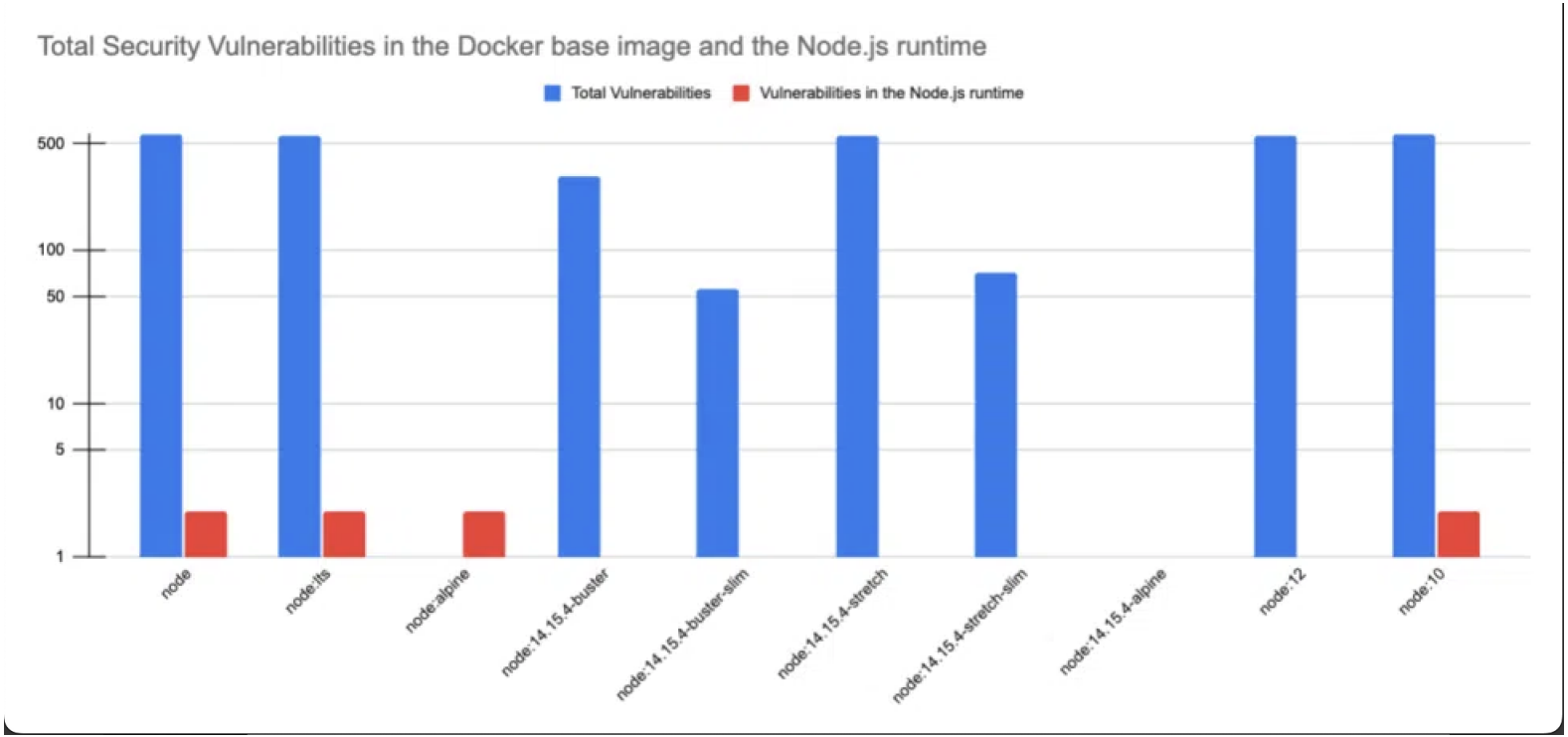
Node.js Image

Node.js Docker image is based on Debian.

For e.g: `` `node:bookworm` ``

Two other variants:

- slim - provides a functional NodeJs env and nothing more.
- alpine - based on Alpine Linux distro, but is an unofficial and is experimental.


Total Security Vulnerabilities in the Docker base image and the Node.js runtime
Total Vulnerabilities
Vulnerabilities in the Node.js runtime
500
100
50
10
5
1
node
node:its
node:alpine
node:14.15.4-buster
node:14.15.4-buster-slim
node:14.15.4-stretch
node:14.15.4-stretch-slim
node:14.15.4-alpine
node:12
node:10

Deterministic tag

`node:alpine`

`node:20.4.0-alpine3.17`

Minimize image size

```
FROM node:20.4.0-alpine3.17
USER node

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
COPY --chown=node:node packages*.json .

RUN npm ci

COPY --chown=node:node server.js .
EXPOSE 8080
CMD ["node", "server.js"]
```

Minimize image size

```
REPOSITORY          TAG       IMAGE ID       CREATED              SIZE
node-20-alpine      latest    ae76c5b808a2   12 seconds ago       185MB
node-20             latest    3abe3becfb39   14 minutes ago       1.1GB
```

Tini (short for Tiny Init)

- init for container
- Tiny helps address the "PID 1 problem" in Docker containers
- Docker expects the main process within a container to run as PID 1, but traditional processes don’t handle signals correctly in this role.
- Tiny ensures that signals (such as SIGTERM) sent to the main Tini process are properly propagated to the child processes, allowing them to gracefully terminate when needed

Tini

```
FROM node:20.4.0-alpine3.17
RUN apk add --no-cache tini
USER node

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
COPY --chown=node:node packages*.json .

RUN npm ci

COPY --chown=node:node server.js .
EXPOSE 8080
ENTRYPOINT ["/sbin/tini", "-"]
CMD ["node", "server.js"]
```

Multi-stage build

```
FROM node:20.4.0-alpine3.17 AS base
RUN apk add --no-cache tini
USER node

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
COPY --chown=node:node packages*.json .

RUN npm ci

COPY --chown=node:node server.js .

FROM base AS APP
EXPOSE 8080
ENTRYPOINT ["/sbin/tini", "-"]
CMD ["node", "server.js"]
```

Multi-stage build: better example

```
# syntax=docker/dockerfile:1
FROM golang:1.21 as build
WORKDIR /src
COPY <<EOF /src/main.go
package main

import "fmt"

func main() {
  fmt.Println("hello, world")
}
EOF
RUN go build -o /bin/hello ./main.go

FROM scratch
COPY --from=build /bin/hello /bin/hello
CMD ["/bin/hello"]
```

Buildkit

- Buildkit is a builder backend
- default builder in Docker Engine as of version 23.0
- BuildKit efficiently skips unused stages and builds stages concurrently when possible.
- Enable Buildkit:

```
DOCKER_BUILDKIT=1 docker build .
```

- Alternatively

```
docker buildx build .
```

Initial DockerFile

```
FROM node:20

WORKDIR /usr/src/app
COPY . .

RUN npm ci
EXPOSE 8080
CMD ["node", "server.js"]
```

Improved DockerFile

```
FROM node:20.4.0-alpine3.17 AS
RUN apk add --no-cache tini
USER node

WORKDIR /usr/src/app

COPY --chown=node:node packages

RUN npm ci
COPY --chown=node:node server.j

FROM base AS APP
EXPOSE 8080
ENTRYPOINT ["/sbin/tini", "-"]
CMD ["node", "server.js"]
```

Other tips

- Use cache mounts
- Leverage HEALTHCHECK
- Use linter

Thank you!

END