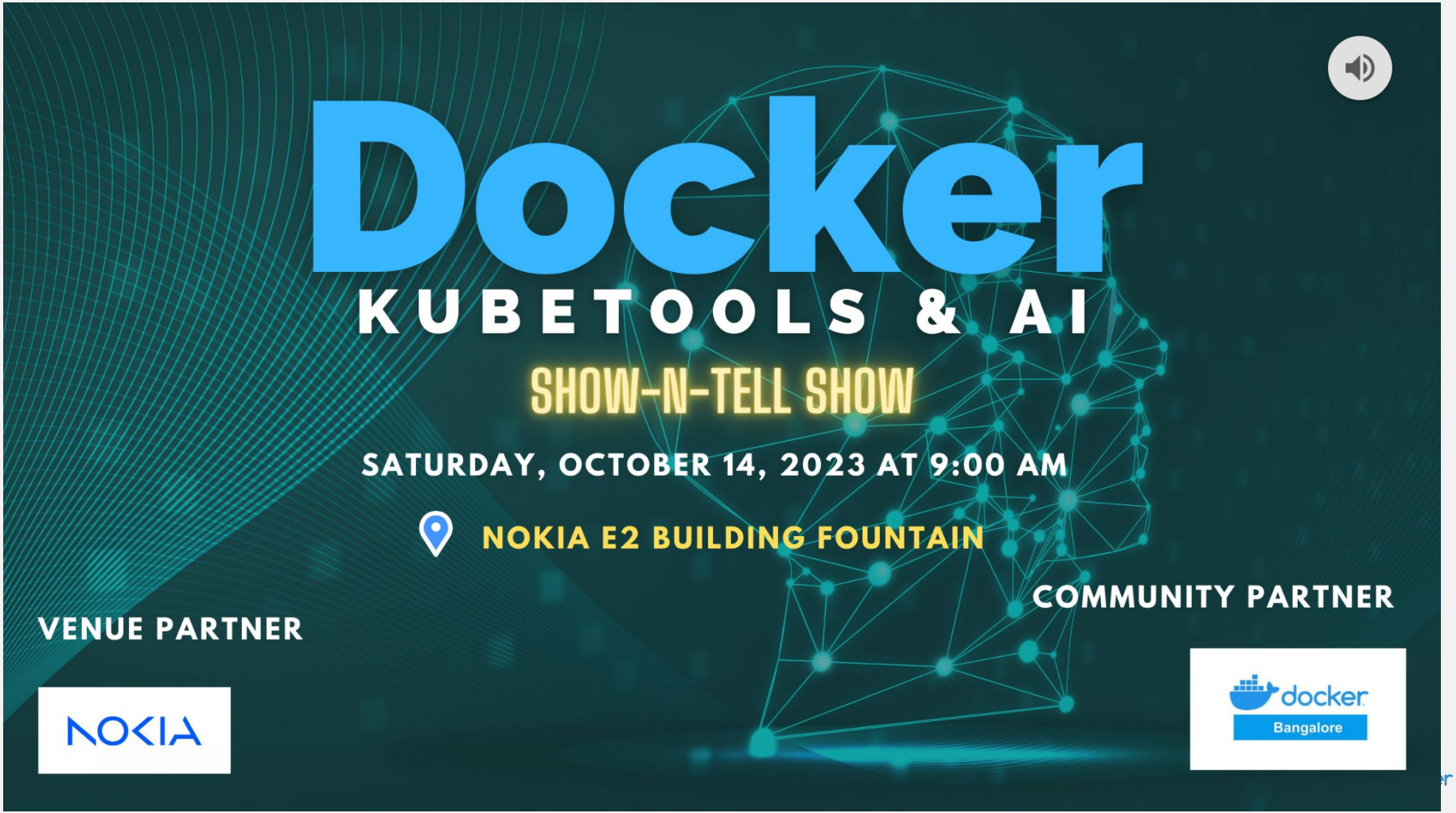
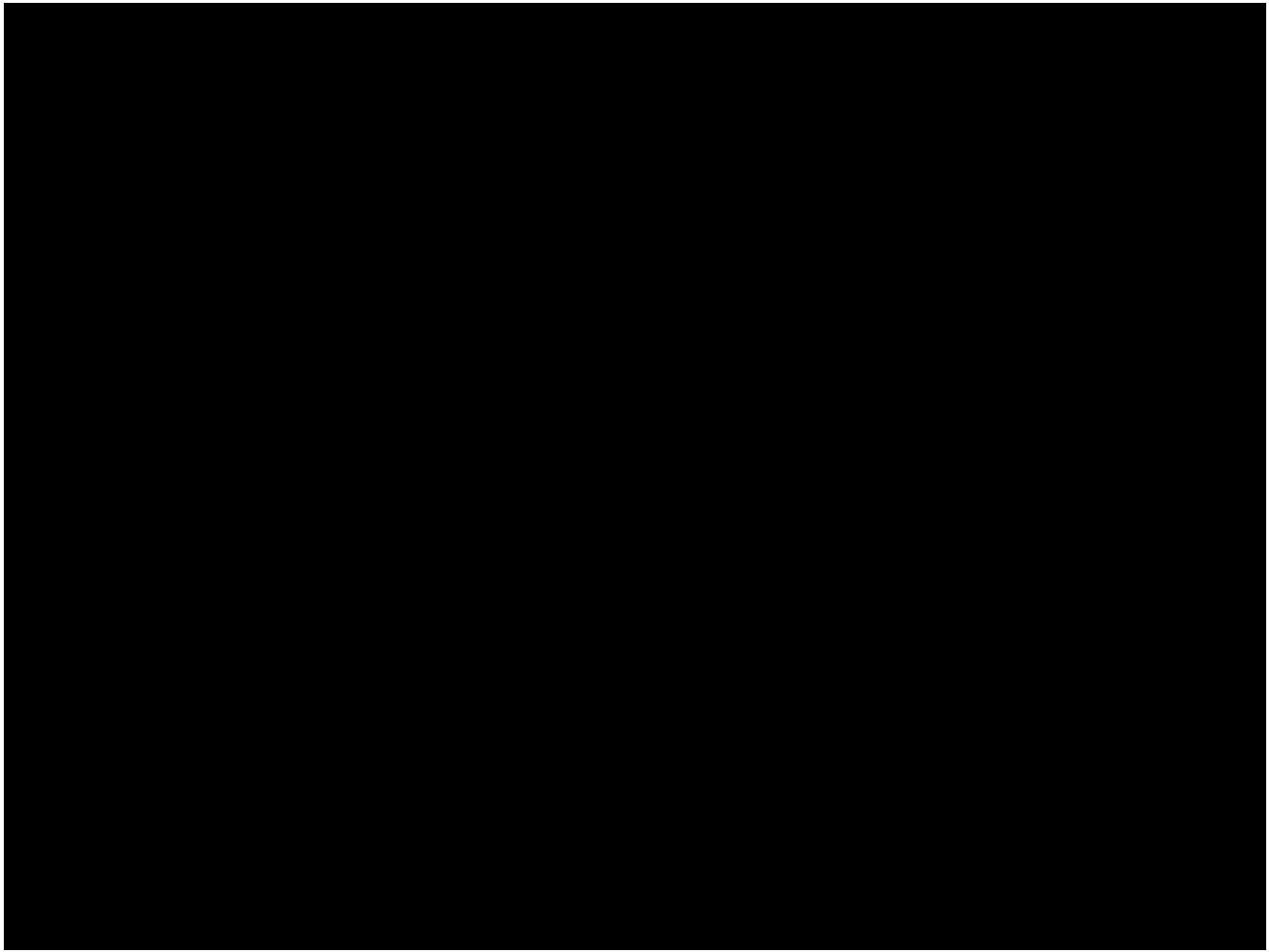
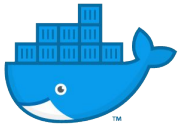
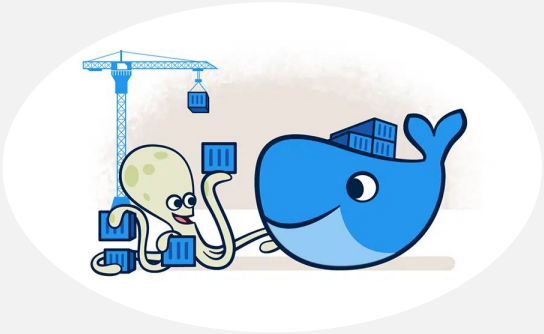

Docker
KUBETOOLS & AI
SHOW-N-TELL SHOW
SATURDAY, OCTOBER 14, 2023 AT 9:00 AM
NOKIA E2 BUILDING FOUNTAIN
VENUE PARTNER
NOKIA
COMMUNITY PARTNER
docker
Bangalore

Docker in 100 seconds





What's New in Docker?



docker

Speaker

- Developer Advocate at Docker
- Author @ https://collabnix.com
- Former Docker Captain
- Docker Community Leader, Bangalore
- ARM Ambassador
- Worked @ Dell EMC, VMware, Redis & CGI

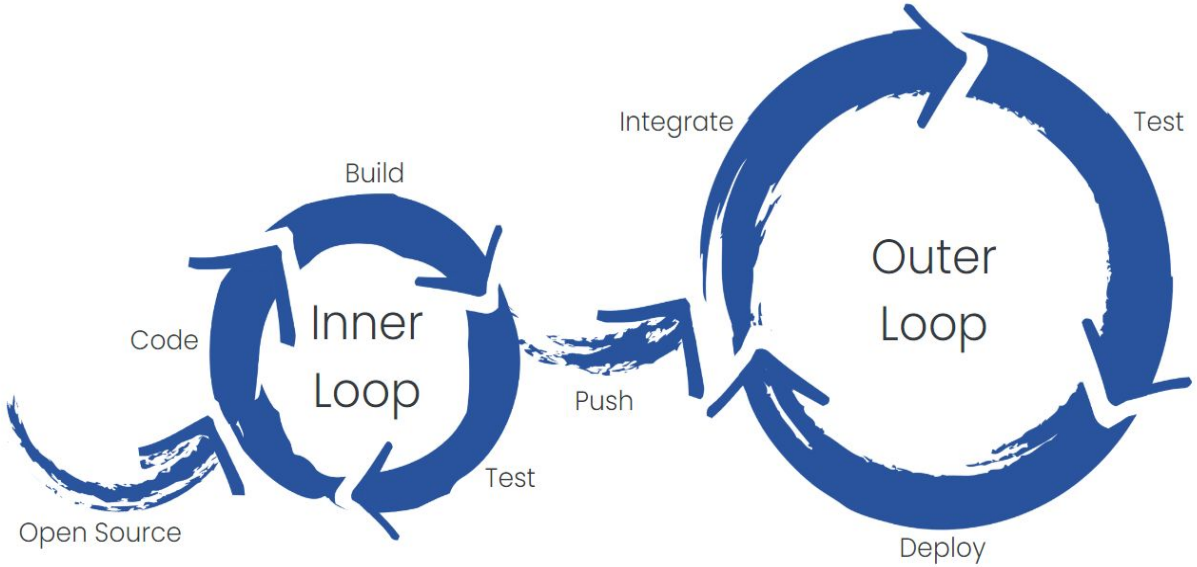
redis

vmware

DELL EMC

docker

Docker is Uniquely Focused on Developer Success



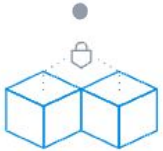
Trusted Images | Docker Desktop | **Docker Ecosystem** | Delivery Platforms

20M+ Active Developers | 450+ Trusted Partners

Built for Developers, by Developers

Speed

- Docker init
- Compose File Watch
- Compose Include
- VirtioFS Support
- VPNKit => gVisor



Security

- Docker Scout
- Attestations

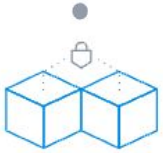
Choice

- Docker Extensions
- Orchestration
- Docker Sponsored Open Source Projects
- Rosetta 2
- WebAssembly

New Dockercon Announcements

Speed

- Compose File Watch went GA
- Next-Gen Docker Build
- Docker Debug



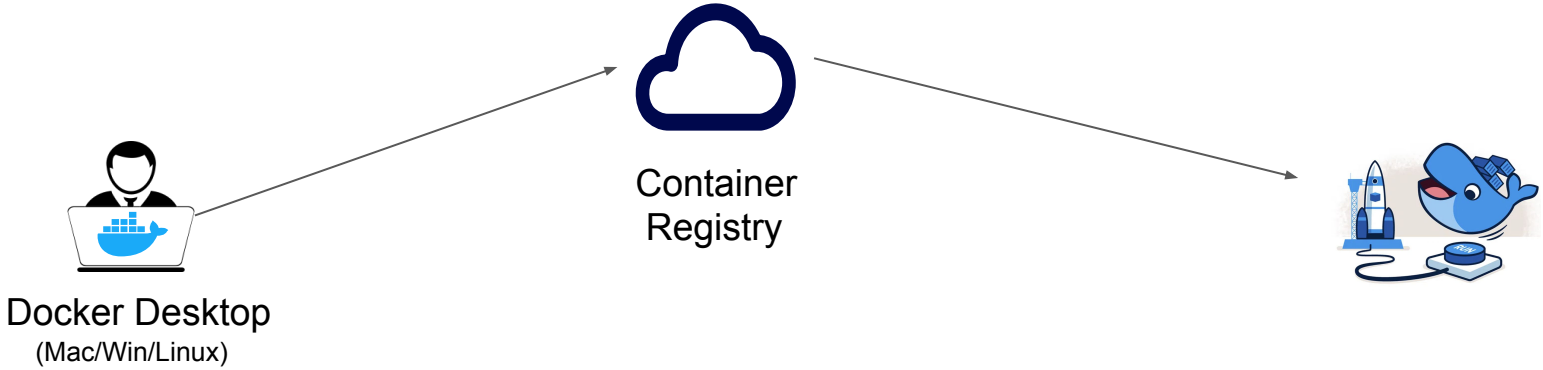
Security

- Docker Scout goes GA

Choice

- GenAI Stack
- Docker AI

A 30,000 ft View



BUILD

- Package applications as portable container images
- Create Multi-container apps using Docker Compose

`$ docker build`

SHARE

- Collaborate and distribute via Registry
- Shareable application with clear interface for operators

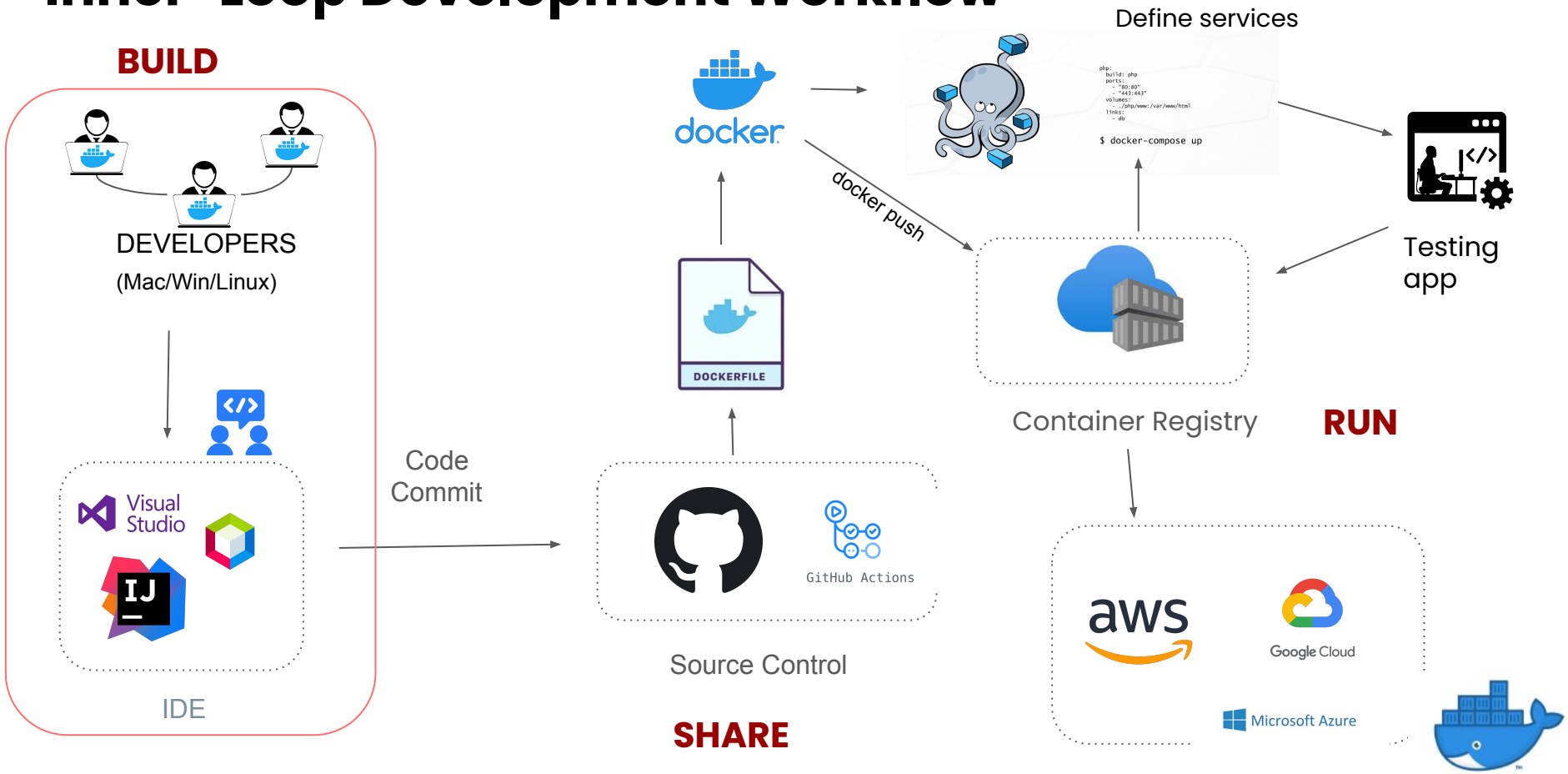
`$ docker push`

RUN

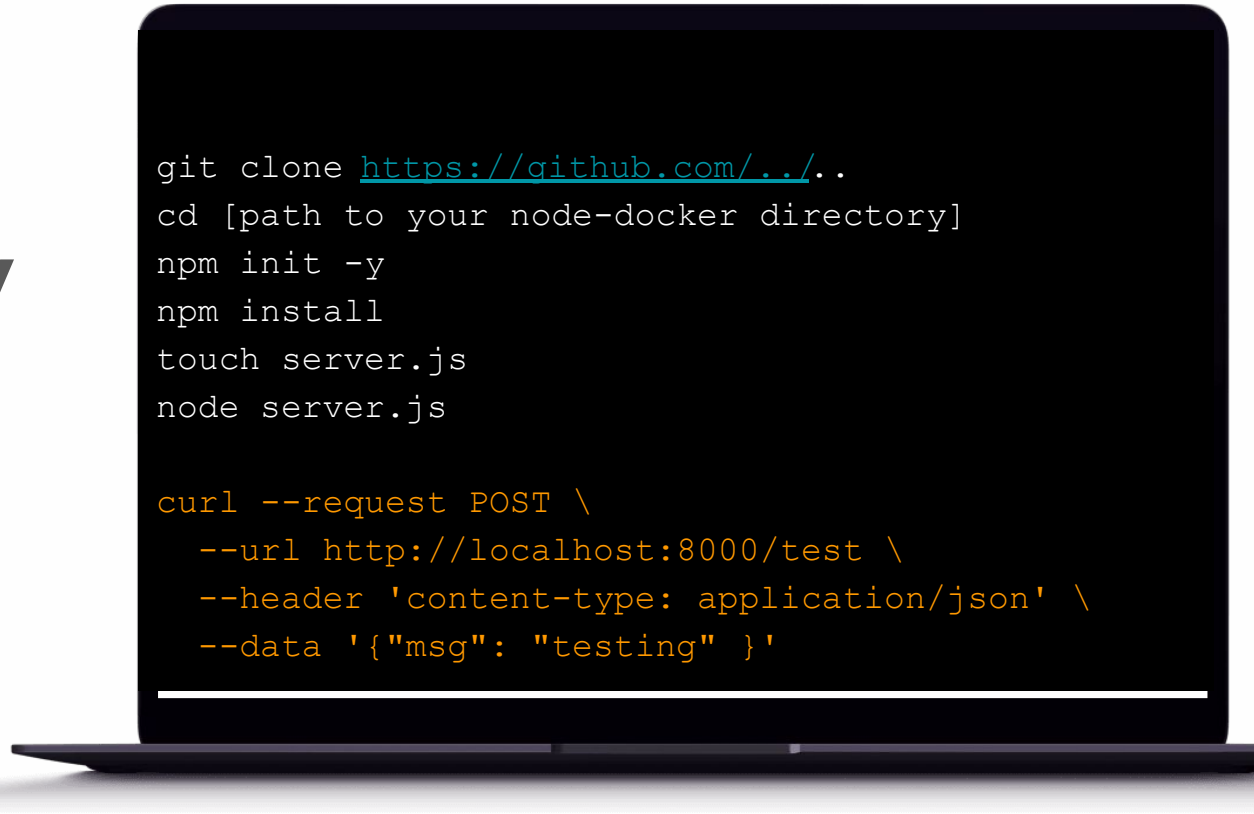
- Run multiple versions of the same application and manage pre-environment settings
- Launch your applications locally and on the cloud with AWS ECS and Azure ACI.

`$ docker run`

Inner-Loop Development Workflow



Build and Deploy a Simple Node Application



Application

```
git clone https://github.com/../..
cd [path to your node-docker directory]
npm init -y
npm install
touch server.js
node server.js

curl --request POST \
  --url http://localhost:8000/test \
  --header 'content-type: application/json'
\
  --data '{"msg": "testing" }'
```

Dockerfile

```
FROM node

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

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

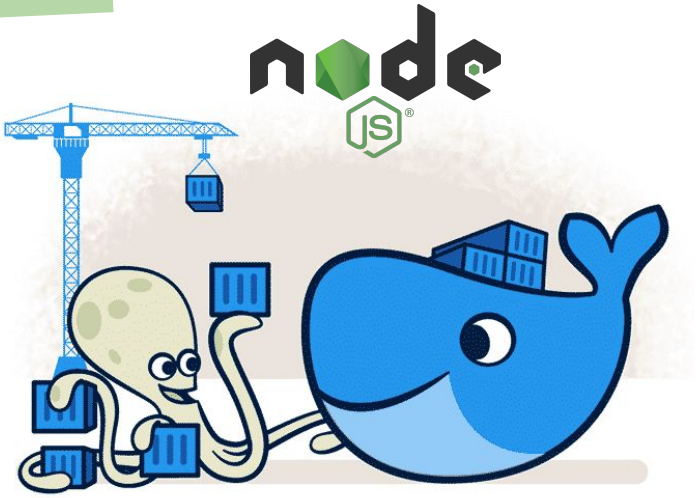
```
$ docker build -t node-docker .
$ docker run -d -p 8080:800 node-docker
```

Do's

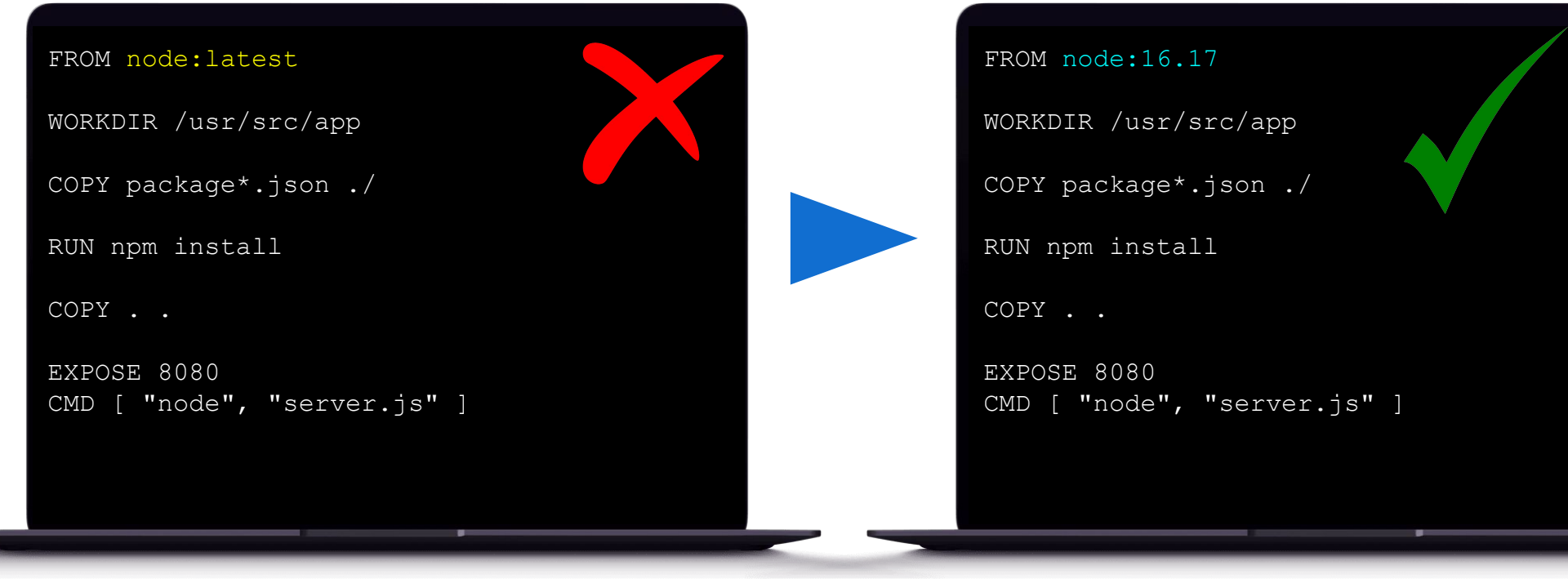
- node:<tag>
- USER node
- MEM LIMIT
- HEALTHCHECK
- COPY
- SCANNING
- Multi-Stage
- .dockerignore
- buildx –platform

Don't

- node:latest
- ADD
- USER root
- EXPOSE db_port
- node_modules
- SIGTERM

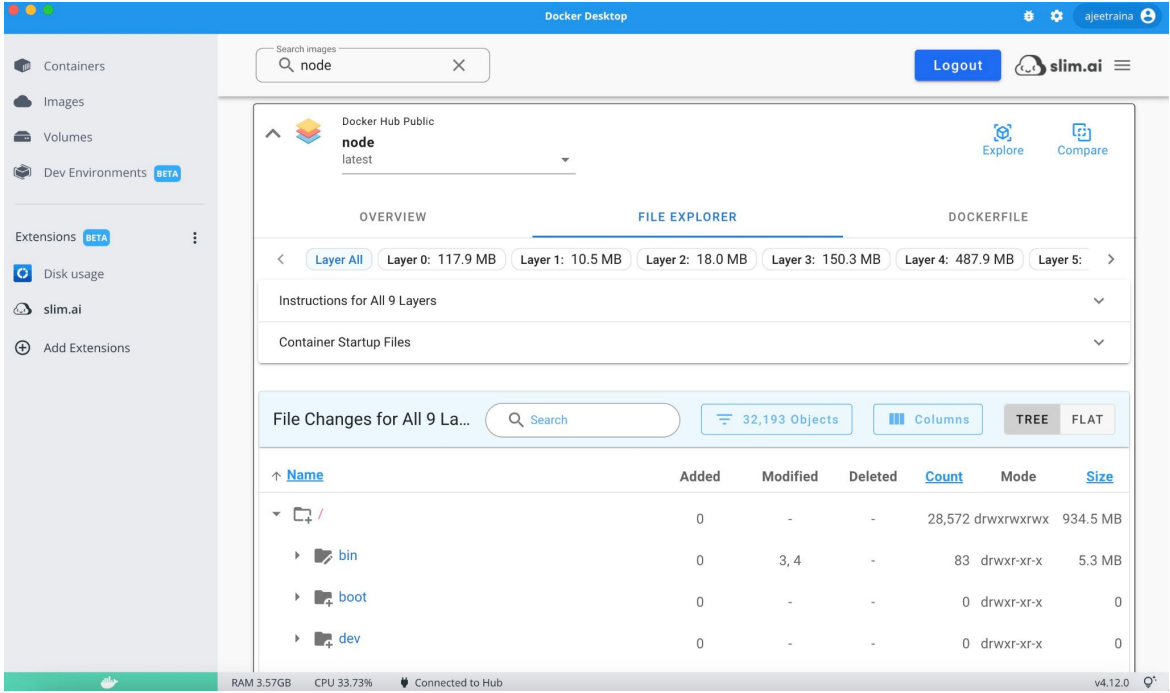


#1 Use explicit Base Image reference instead of latest

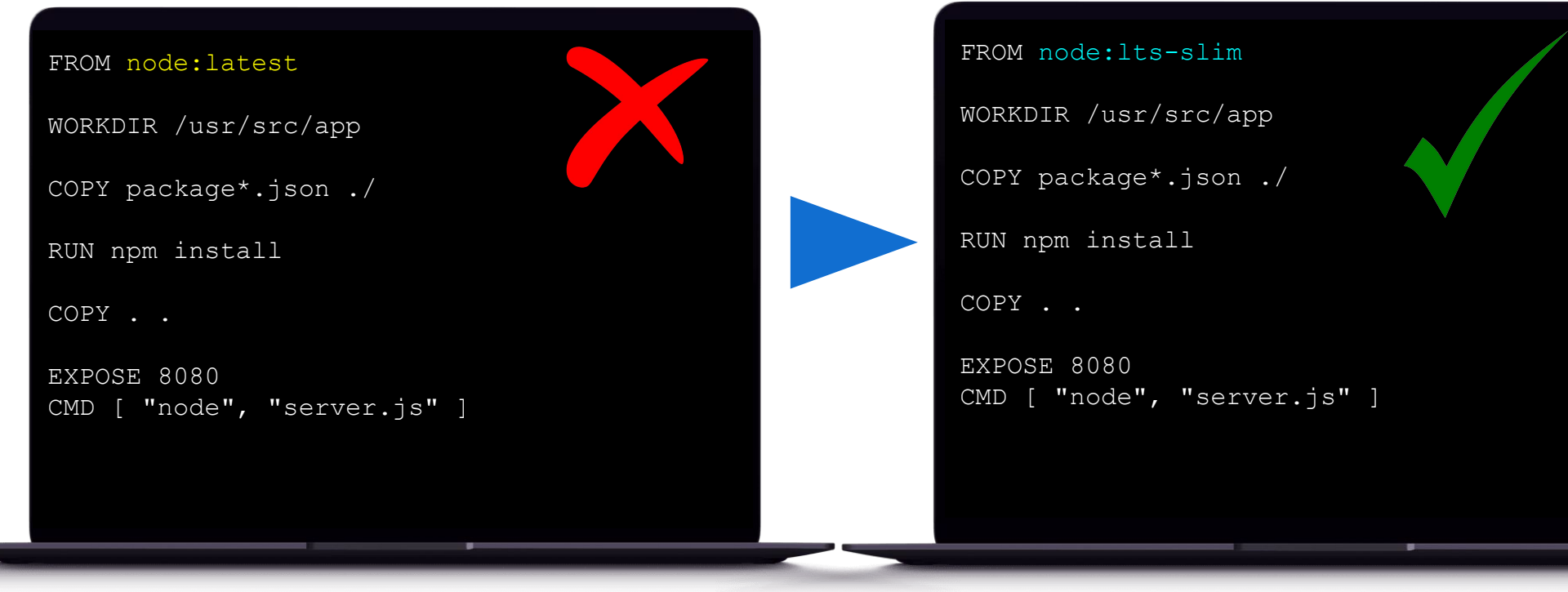


Downsides of latest Images

- Inconsistent Docker Image Builds
- Non-deterministic Behavior

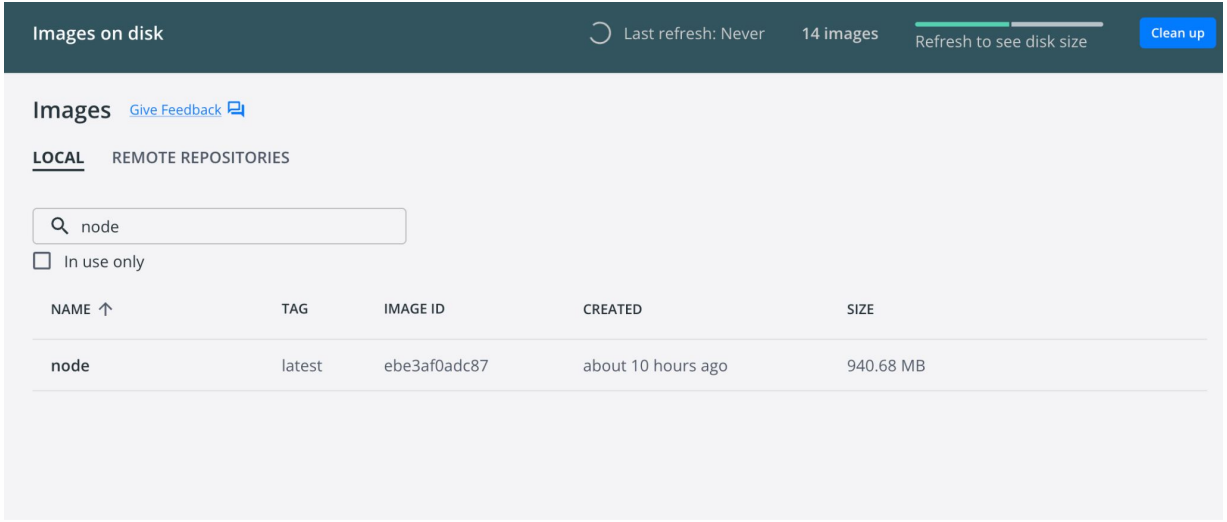


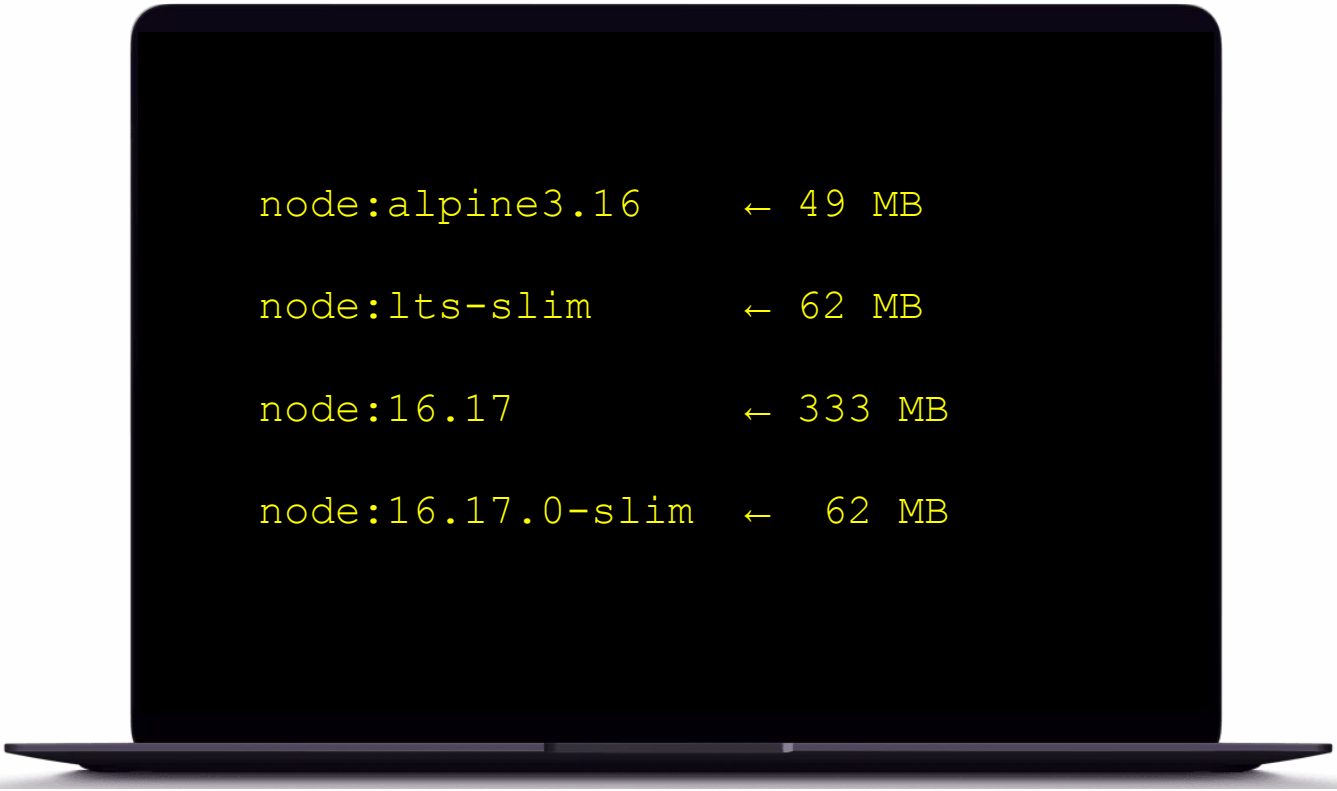
#2 Prefer Leaner Docker Images



Downsides of Bigger Image

- Larger Download Size
- Higher Exposure to Vulnerabilities
- Increases Resource Consumption

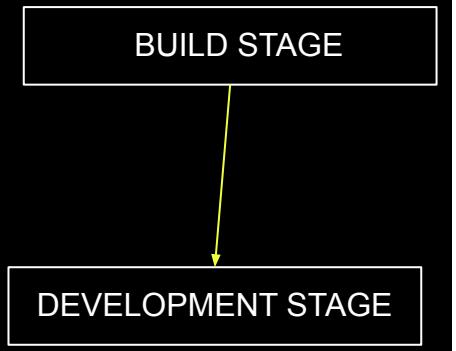



node:alpine3.16 ← 49 MB
node:lts-slim ← 62 MB
node:16.17 ← 333 MB
node:16.17.0-slim ← 62 MB

#3

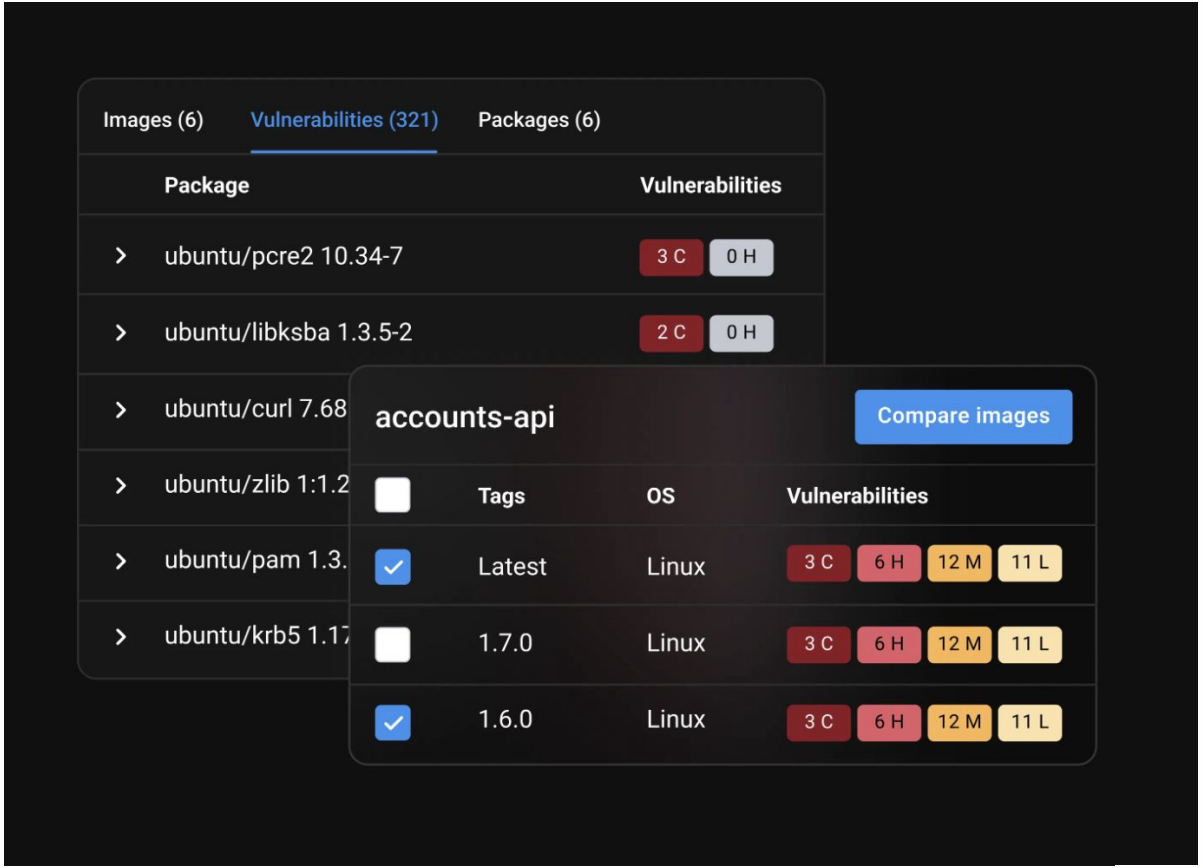
Use Multi-Stage Build

```
FROM node:lts-buster-slim AS builder

WORKDIR /usr/src/app

COPY package.json /usr/src/app/package.json
COPY package-lock.json /usr/src/app/package-lock.json
RUN npm ci

COPY . /usr/src/app

EXPOSE 3000

CMD [ "npm", "run", "dev" ]

FROM builder as dev-envs
RUN <<EOF
apt-get update
apt-get install -y --no-install-recommends git
EOF


# install Docker tools (cli, buildx, compose)
```



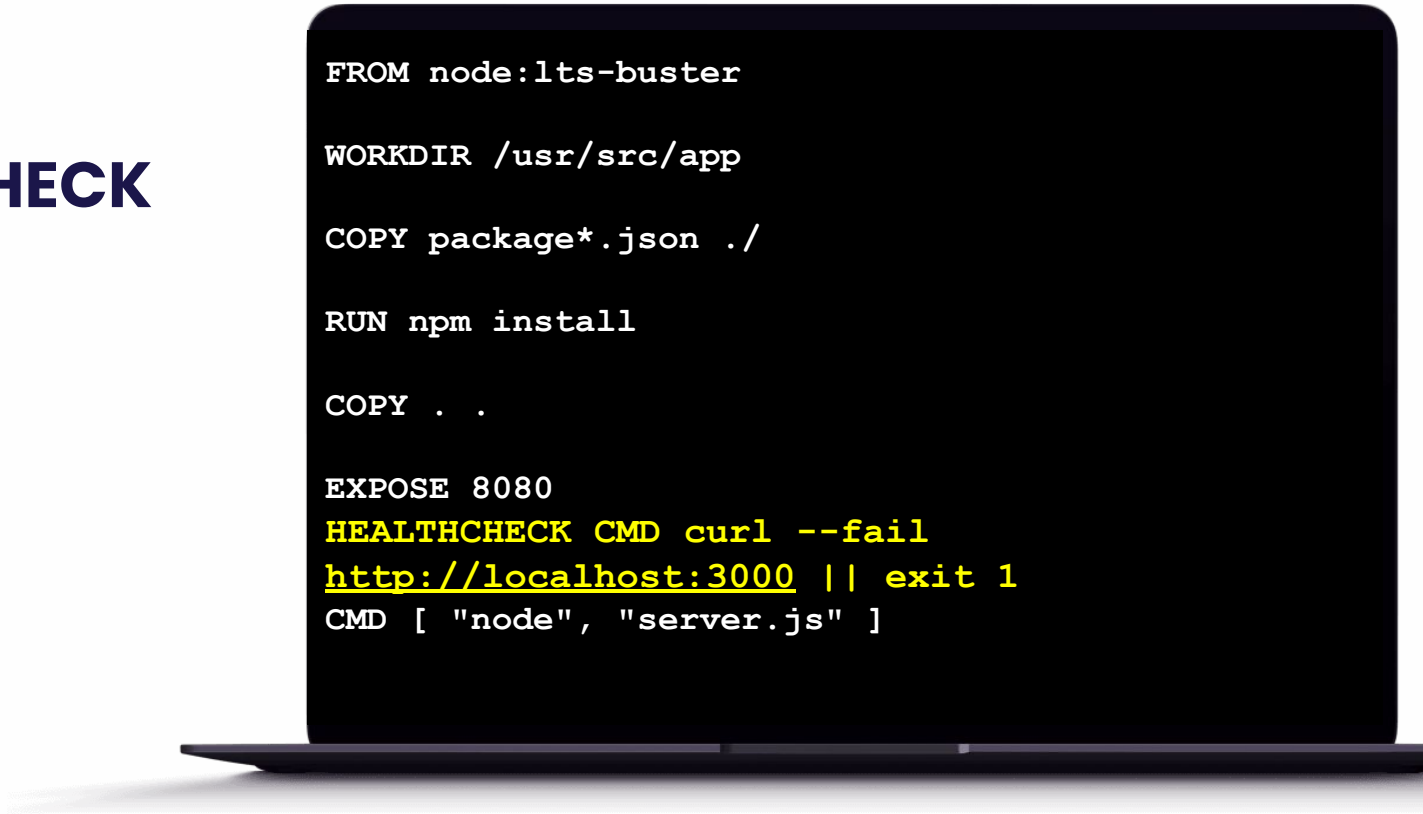
#4

Quickly identify and fix vulnerabilities during the build time using Scout



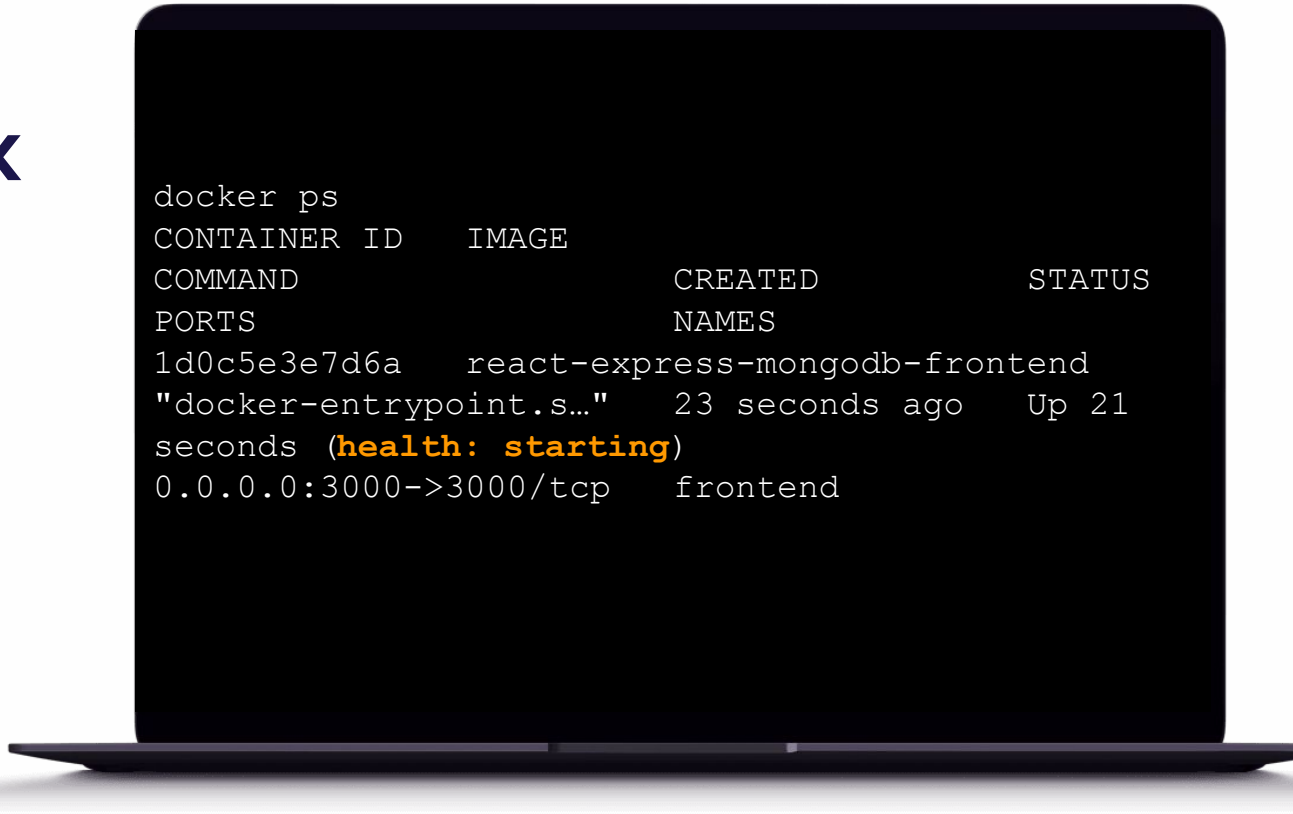
#5

Add HEALTHCHECK



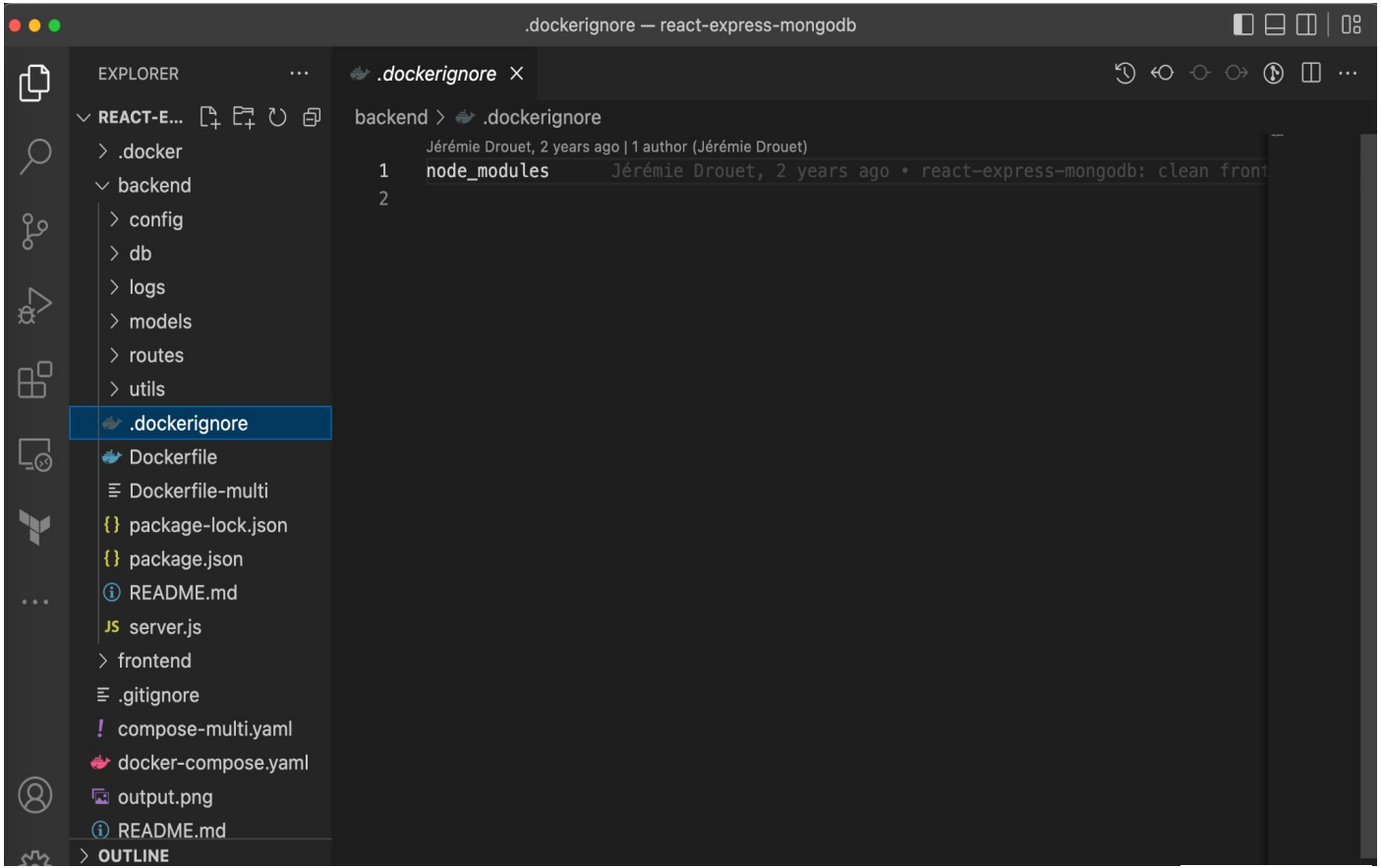
#5

Add HEALTHCHECK



#6

Use .dockerignore



#7

Run as non-root user for security purpose

```
FROM node:lts-buster AS development
WORKDIR /usr/src/app
COPY package.json /usr/src/app
COPY package-lock.json /usr/src/app
RUN npm ci
COPY . /usr/src/app
EXPOSE 3000
CMD ["npm", "start"]

FROM development as dev-envs
RUN <<EOF
apt-get update
apt-get install -y --no-install-recommends git
EOF

RUN <<EOF
useradd -s /bin/bash -m vscode
groupadd docker
usermod -aG docker vscode
EOF
# install Docker tools (cli, buildx, compose)
COPY --from=gloursdocker/docker / /
```

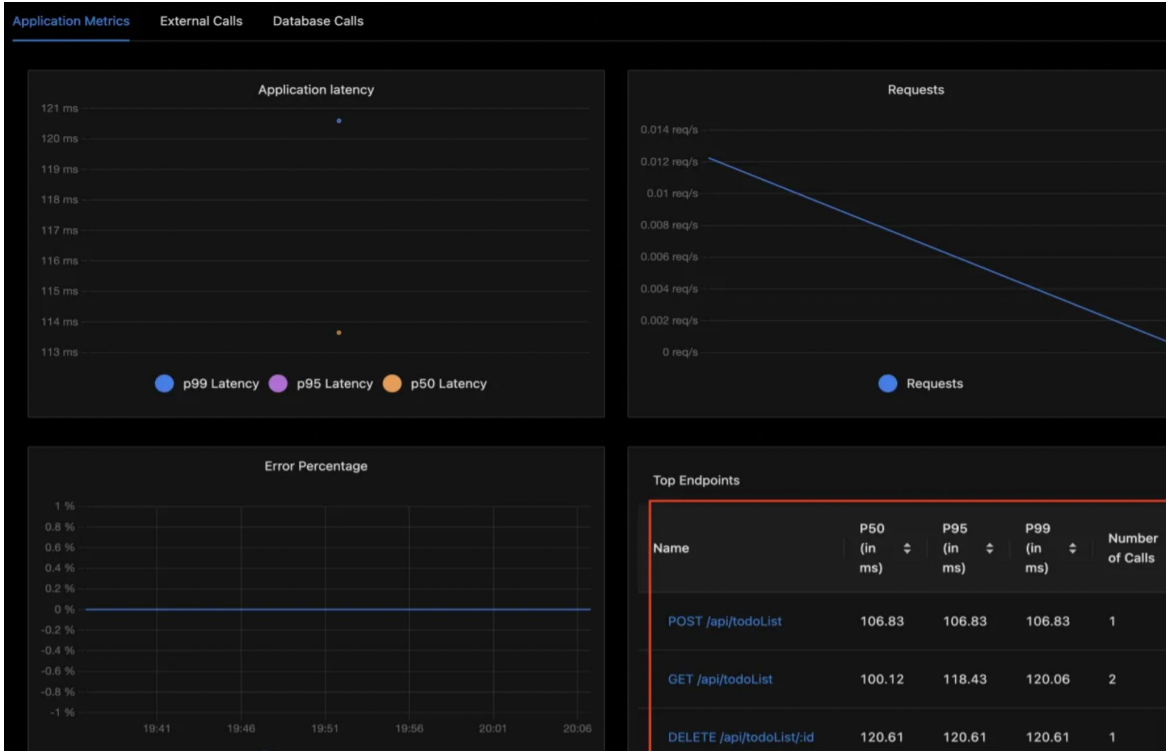
#8

Favour Multi-Architecture Docker Images

```
docker run --rm mplatform/mquery node:lts-buster
Unable to find image 'mplatform/mquery:latest'
locally
d0989420b6f0: Download complete
af74e063fc6e: Download complete
3441ed415baf: Download complete
a0c6ee298a93: Download complete
894bcacb16df: Downloading
[==========================================>
]  3.146MB/3.452MB
Image: node:lts-buster (digest:
sha256:a5d9200d3b8c17f0f3d7717034a9c215015b7aae70cb2
a9d5e5dae7ff8aa6ca8)
 * Manifest List: Yes (Image type:
application/vnd.docker.distribution.manifest.list.v2
+json)
 * Supported platforms:
   - linux/amd64
   - linux/arm/v7
   - linux/arm64/v8
```

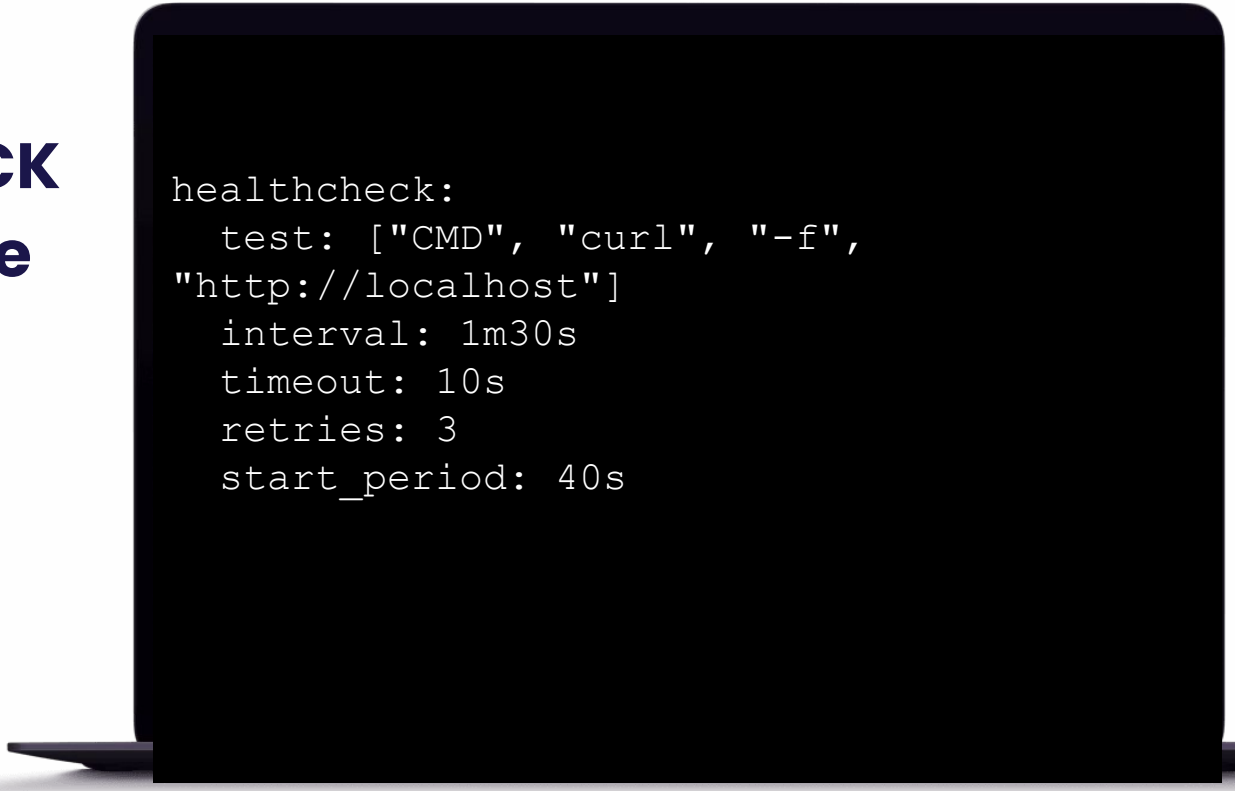
#9 Use Open Telemetry API to measure Node performance

- Metrics(what)
- Logs(why)
- Trace(how)



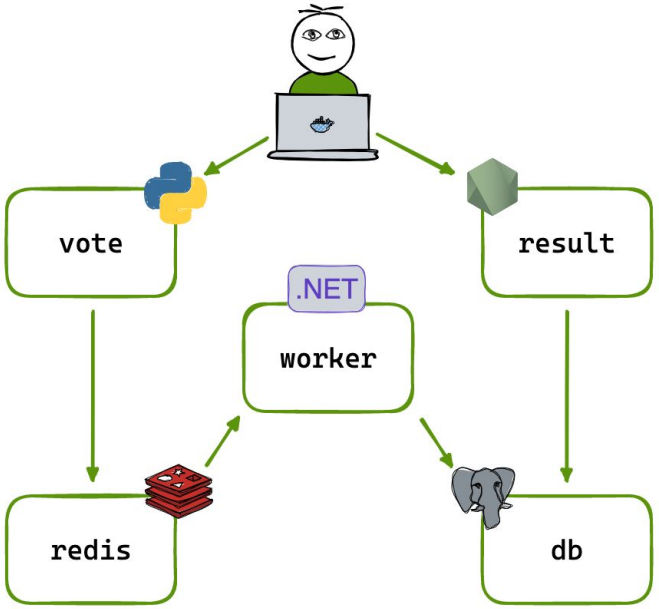
#10

Using HEALTHCHECK in Docker Compose

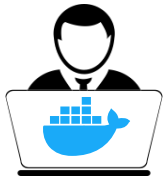


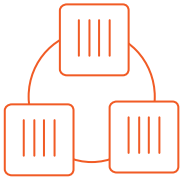
Build and Deploy a Voting App

- A front-end web app(**Vote**) written in Python Flask framework which allows the users to cast their votes.
- A Redis which collects votes casted by the users
- A service written in .NET(**worker**) that retrieves the votes data from Redis and store it into Postgres DB service
- A Postgres database used as a persistent storage database, backed by a Docker volume
- A Node.js web app(**result**) which shows the results of the voting in real time



Building Prod-based Containerised Application is Painful

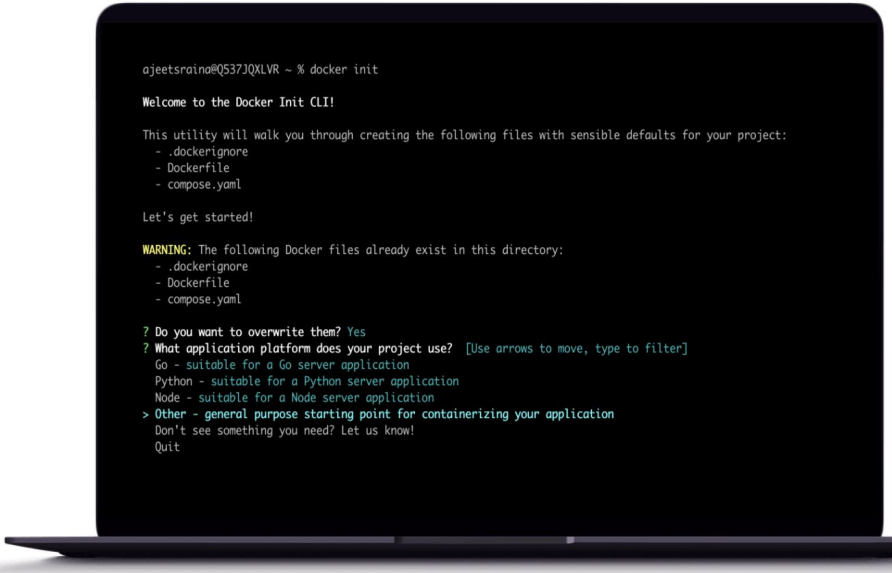




Introducing

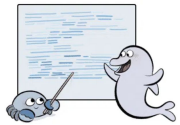
$ docker init

- Introduced for the first time in [Docker Desktop 4.18]
- Generates Docker assets for projects
- Allows you to choose application platform
- Makes it easier to create Docker images and containers

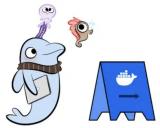


$ docker init

Simplified Docker Assets Creation



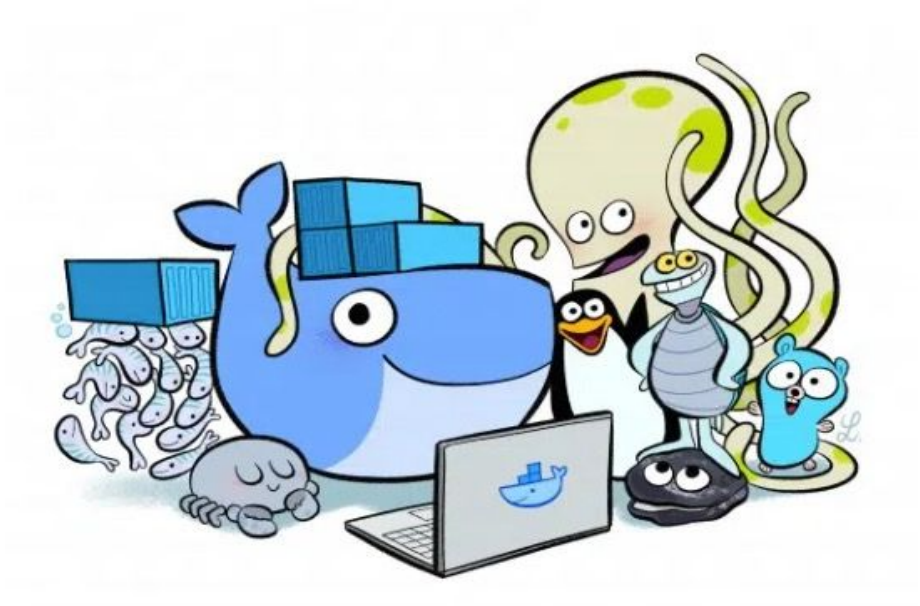
Saves Time and Effort



Better Project Organization

Enhanced Portability

It’s Demo Time



https://github.com/dockersamples/docker-init-demos

Compose Watch

Compose Watch

- No Longer Experiment Features, it's now GA
- Automatically updates your compose service containers while you work
- Blazing-fast file synchronization supporting live update

```
services:
  web:
    build: .
    command: npm start
    x-develop:
      watch:
        - action: sync
          path: ./web
          target: /src/web
          ignore:
            - node_modules/
        - action: rebuild
          path: package.json
```

How it works?

- Automatically builds a new image with BuildKit and replaces the running service container
- Add an x-develop section to your services in the compose.yaml file
- Configure it with a list of paths to watch and actions to take
- Watch rules allow ignoring specific files or entire directories within the watched tree.

Compose Watch Actions

Sync

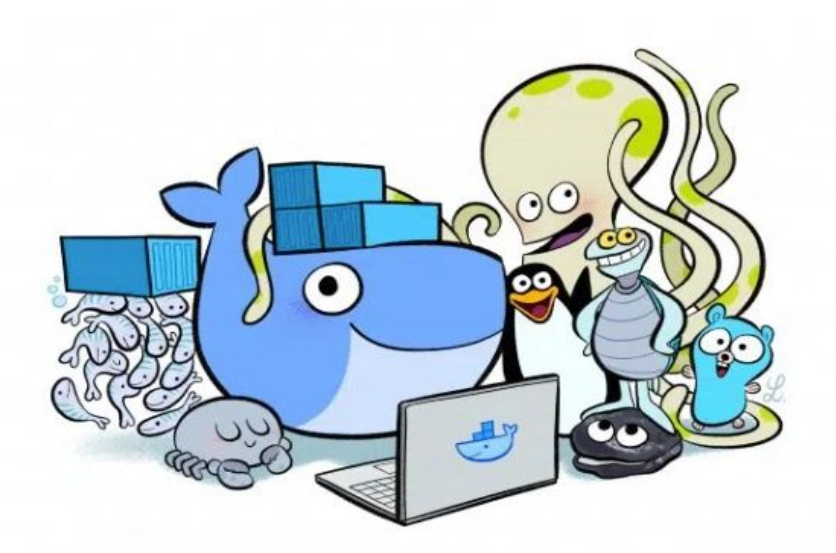
Specifies a path to watch for changes in the host file system, and a corresponding target path inside the container to synchronize changes to.

Rebuild

The "rebuild" action specifies a path to watch for changes in the host file system, and triggers a rebuild of the container when changes are detected.

```
services:
  app:
    build: .
    x-develop:
      watch:
        - action: rebuild
          path: .
    command: sh -c "yarn install && yarn run dev"
    ports:
      - 3000:3000
    working_dir: /app
    volumes:
      - ./:/app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: secret
      MYSQL_DB: todos

  mysql:
    image: mysql:8.0
    ports:
      - 3306:3306
    volumes:
      - todo-mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: todos

volumes:
  todo-mysql-data:
```

It’s Demo Time



https://github.com/dockersamples/avatars

$ docker compose watch

Compose Include

Compose Include

- Introduced in Docker Compose 2.20
- Makes it easier to modularize complex applications into sub-compose files
- Ability to reuse other Compose files

web.yaml

```yaml
version: '2'
include:
  - postgresql.yaml
services:
  web:
    image: nginx:latest
    depends_on:
      - database

```

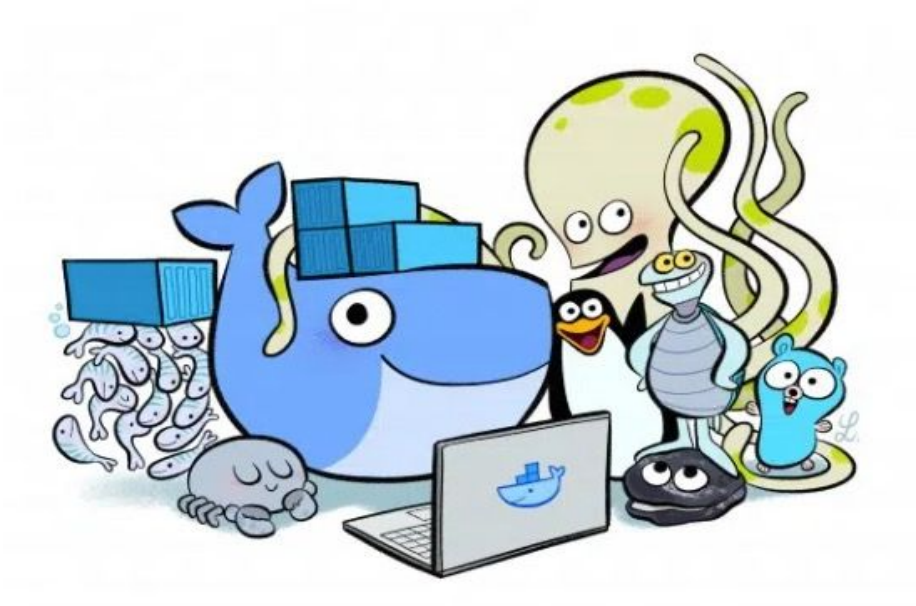
How it works?

- Once the included Compose app is loaded, all resource definitions are copied into the current Compose application model
- Any volumes, network or other resources pulled in from the included Compose file can be used by the current Compose app for cross-service references.

postgresql.yaml

```yaml
services:
  database:
    image: postgres:12
    environment:
      POSTGRES_PASSWORD: password
```

It’s Demo Time



https://github.com/dockersamples/compose-include-demo

Container Security Monitoring for Developers

The “Software Supply Chain” in Practice

- Software is built from **multiple parts**, dependencies and **open source components**
- All software contains **vulnerabilities** - both known and unknown
- Software development involves **multiple developers**, teams and system
- Components are created by **contributors both inside and outside the organization**
- Attackers target components with vulnerabilities to gain access & insert **malicious code** into an organization’s software

Software Supply Chain Security Challenges

96%

of applications include open source components

70–90%

of code in an application is from open source

300%

increase in supply chain attacks in 2021

9%

of dev teams know how to safely use open source components

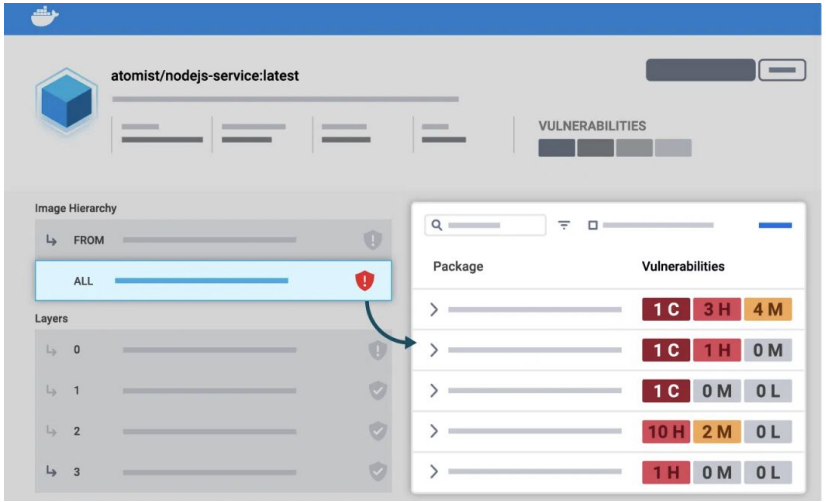
Source ~ [The State of Software Supply Chain Report 2023 by Sonatype]

Given All That, Every Company Needs to..

Know what software they're running

Know what risks that software has

Fix those risks quickly



Empowering Developers with Docker Scout

Understand your applications' **dependencies**

Analyze the vulnerabilities in your dependencies

Act quickly with suggested remediation options

How is Open Source Software (OSS) Vulnerable?

Malicious actors try to exploit old OSS code

- Malicious actors **search for weaknesses in Open Source** software, and prey on dependant codebases
- **They exploit these weaknesses** to enter the software Supply Chain
- The aim is to **use other trusted software as a host** from which to **deploy malicious code** into a target's environment

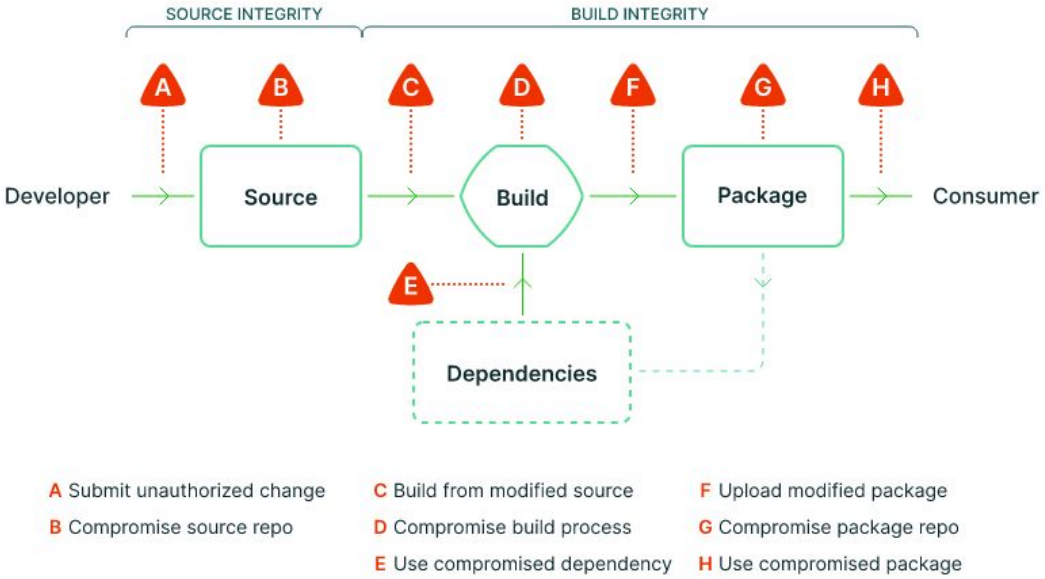
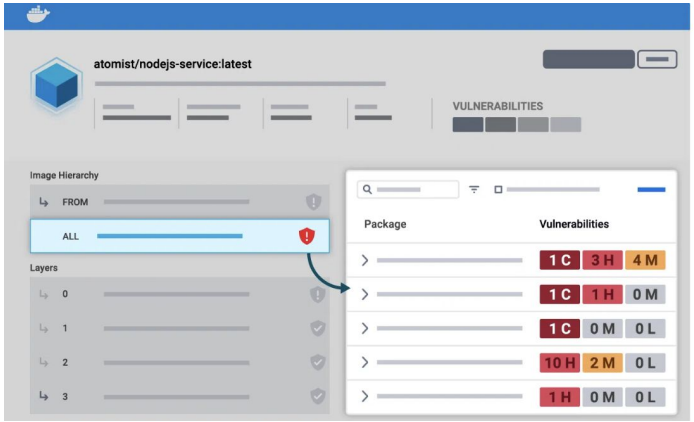


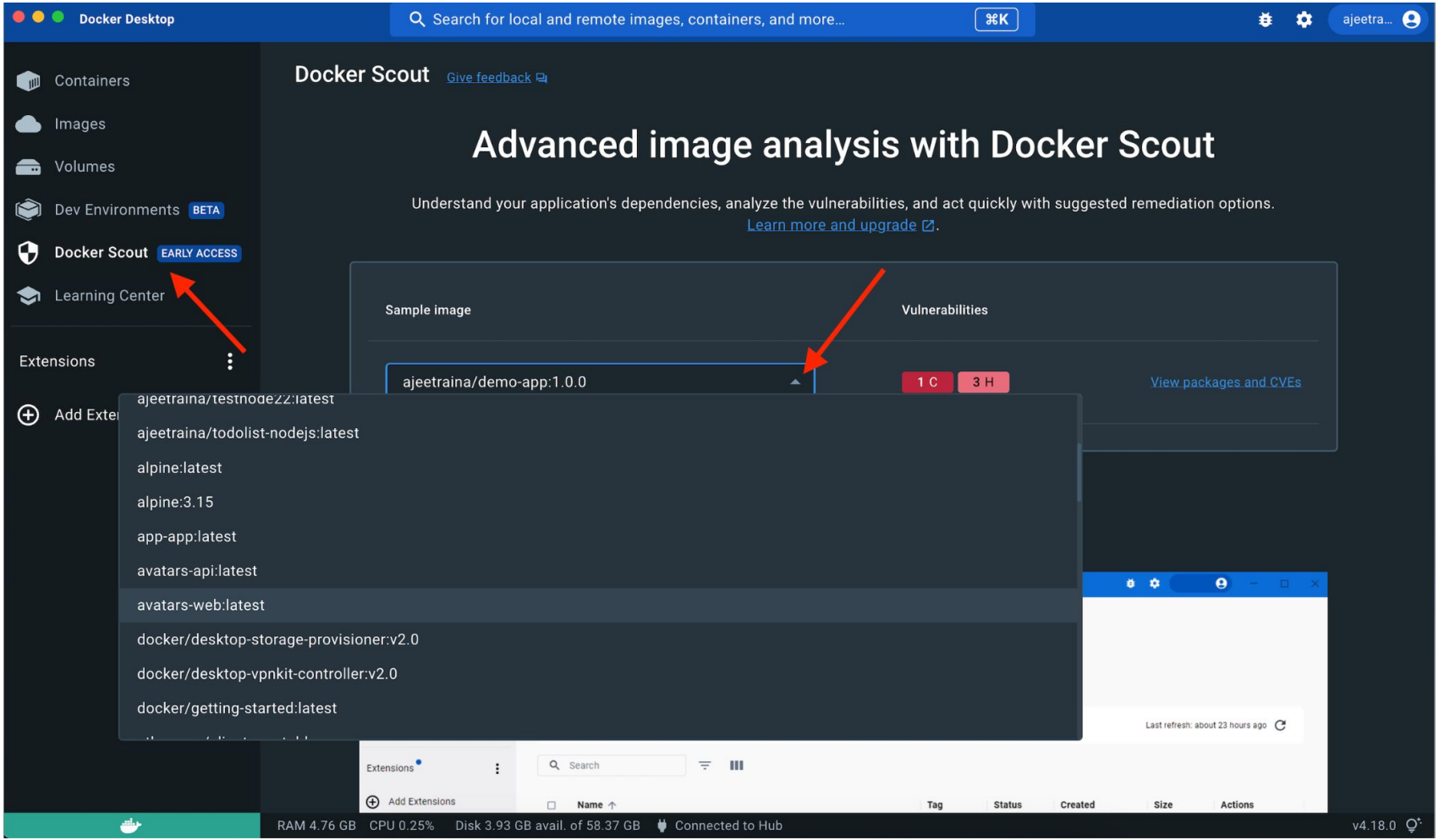
Image source: https://slsa.dev/spec/v0.1/#supply-chain-threats

Secure Your Supply Chain at each Layer

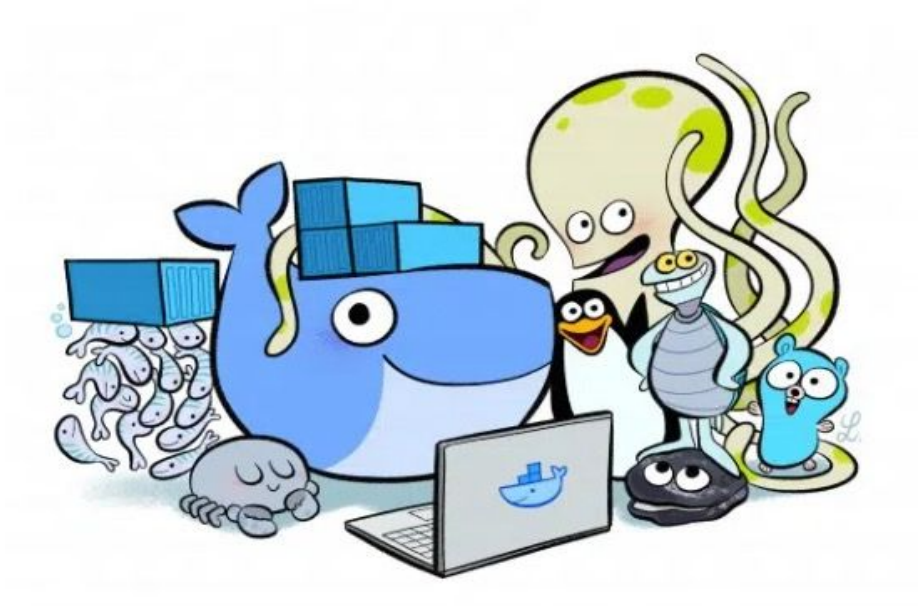
- Unified view of securing your container development
- Includes a layer-by-layer view of dependencies, their known vulnerabilities, and recommended remediation paths.
- Designed with developers in mind
- 1st Class Citizen(integrated directly into Docker)
- Sits as a layer on top of the Docker ecosystem to help developers build and maintain a secure software supply chain



"Spend less time search for and fixing vulnerabilities"


Docker Desktop
Search for local and remote images, containers, and more...
⌘K
ajeetra...
Containers
Images
Volumes
Dev Environments BETA
Docker Scout EARLY ACCESS
Learning Center
Extensions
Add Exte
Docker Scout
Give feedback
Advanced image analysis with Docker Scout
Understand your application's dependencies, analyze the vulnerabilities, and act quickly with suggested remediation options.
Learn more and upgrade.
Sample image
Vulnerabilities
ajeetraina/demo-app:1.0.0
1 C
3 H
View packages and CVEs
ajeetraina/testnode22:latest
ajeetraina/todolist-nodejs:latest
alpine:latest
alpine:3.15
app-app:latest
avatars-api:latest
avatars-web:latest
docker/desktop-storage-provisioner:v2.0
docker/desktop-vpnkit-controller:v2.0
docker/getting-started:latest
Last refresh: about 23 hours ago
Extensions
Add Extensions
Search
Name
Tag
Status
Created
Size
Actions
RAM 4.76 GB
CPU 0.25%
Disk 3.93 GB avail. of 58.37 GB
Connected to Hub
v4.18.0

It’s Demo Time



https://github.com/docker/scout-demo-service

Scout CLI Integration with CI Environments

GitHub Actions

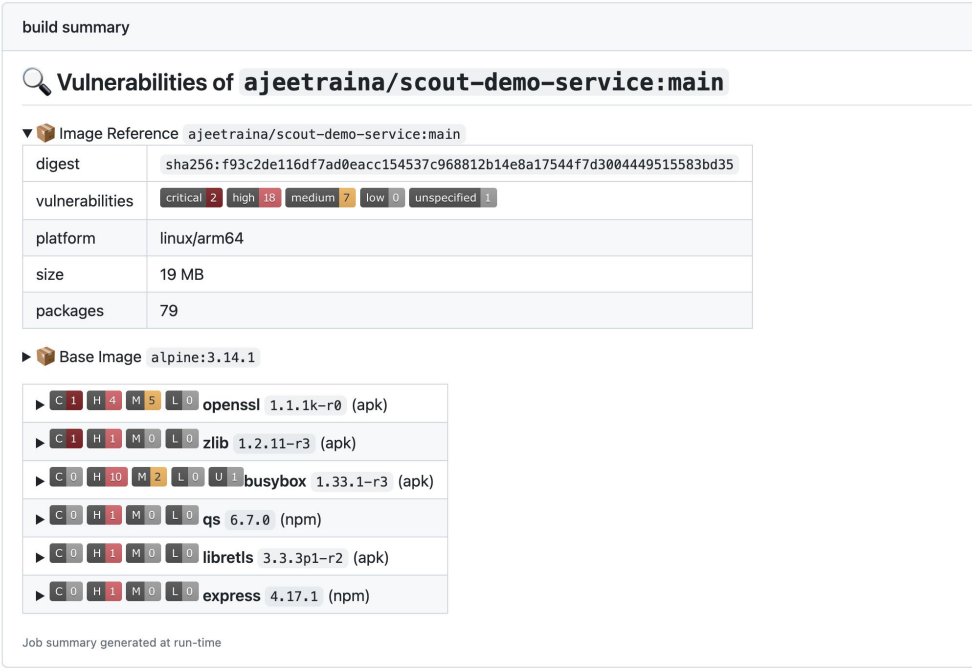
GitLab

Jenkins

Azure Pipelines

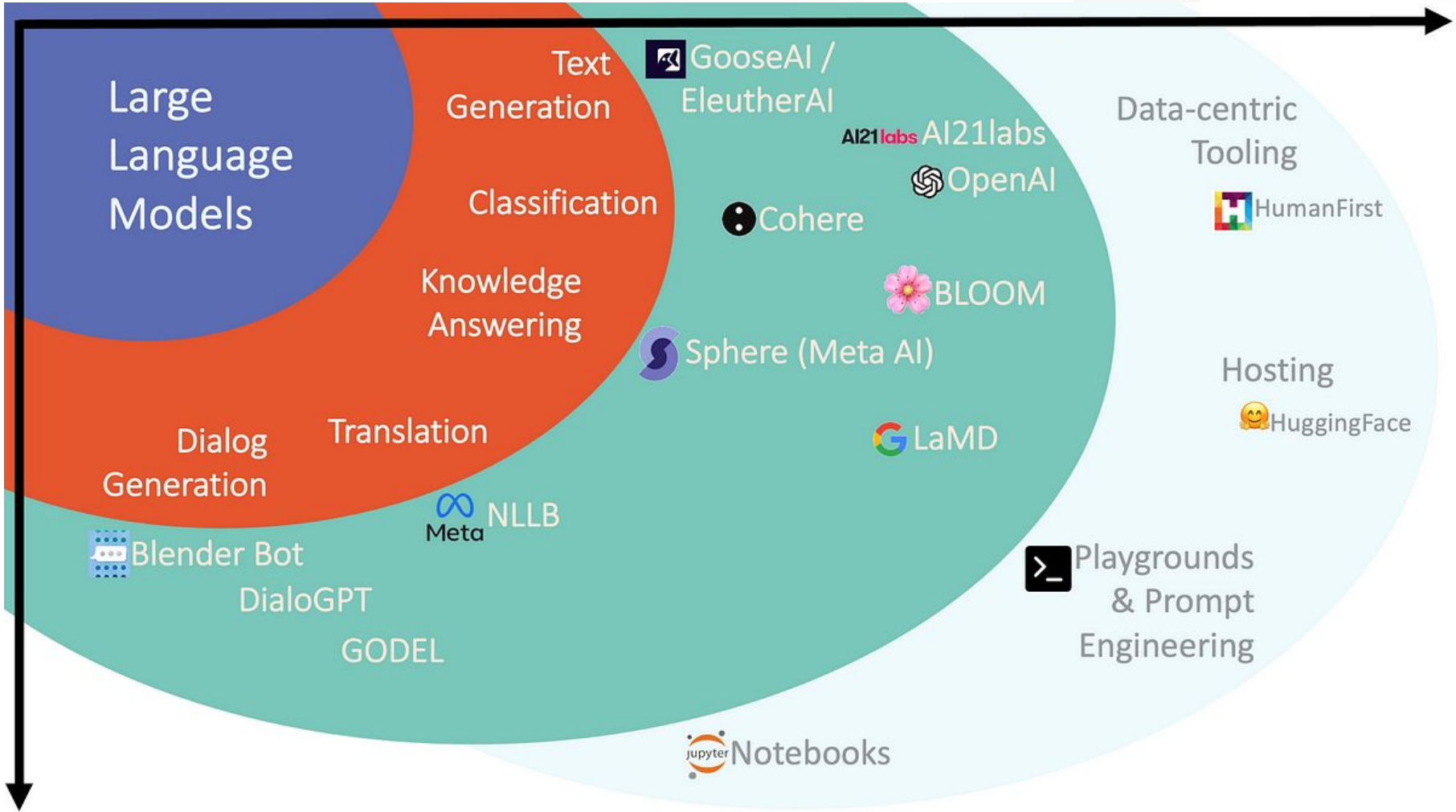
circleci

ATLASSIAN Bitbucket



The GenAI Stack

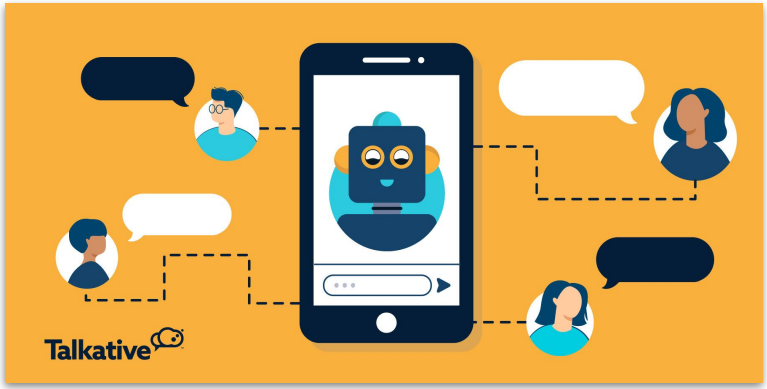

LLM! LLM!
LIGHTYEAR
EVERYWHERE
imgflip.com


Large Language Models
Text Generation
Classification
Knowledge Answering
Dialog Generation
Translation
GooseAI / EleutherAI
AI21labs AI21labs
OpenAI
Cohere
BLOOM
Sphere (Meta AI)
LaMD
Meta NLLB
Blender Bot
DialoGPT
GODEL
Data-centric Tooling
HumanFirst
Hosting
HuggingFace
Playgrounds & Prompt Engineering
jupyter Notebooks

Why are LLMs so popular?

- Automate data retrieval tasks such as finding and summarizing info from large datasets
- Improve customer service experiences - Creating Chatbots
- Expedite reading, understanding, and summarizing
- Content & code generation

... save time and money, improve growth and retention



Top 5 GenAI Challenges

1. Mostly trained to generate human-like language, not to make accurate inferences with that language
2. Training data comes from publically available corpuses which can contain false, bias, or contradicting information
3. Knowledge cut-off due to resource intensive training (i.e. ChatGPT only trained on data up to Sep 2021)
4. Lack of enterprise domain knowledge
5. Inability to verify or attribute sources

Result:

- Great language understanding
- Issues with factual accuracy and consistency

Therefore:

Companies cannot rely on LLMs alone for mission critical data and decisions, instead, they must rely on private, factual information.

	Parrot	ChatGPT
		
Learns random sentences from random people	✓	✓
Talks like a person but doesn't really understand what it's saying	✓	✓
Occasionally speaks absolute non sense	✓	✓
Is a cute little bird	✓	✗

LLM Hallucinations

Definition: Language models generate text that is incorrect, nonsensical, or unreal.

- Appear to answer questions confidently even if they don't have facts
- May provide contradicting or inconsistent responses to similar prompts

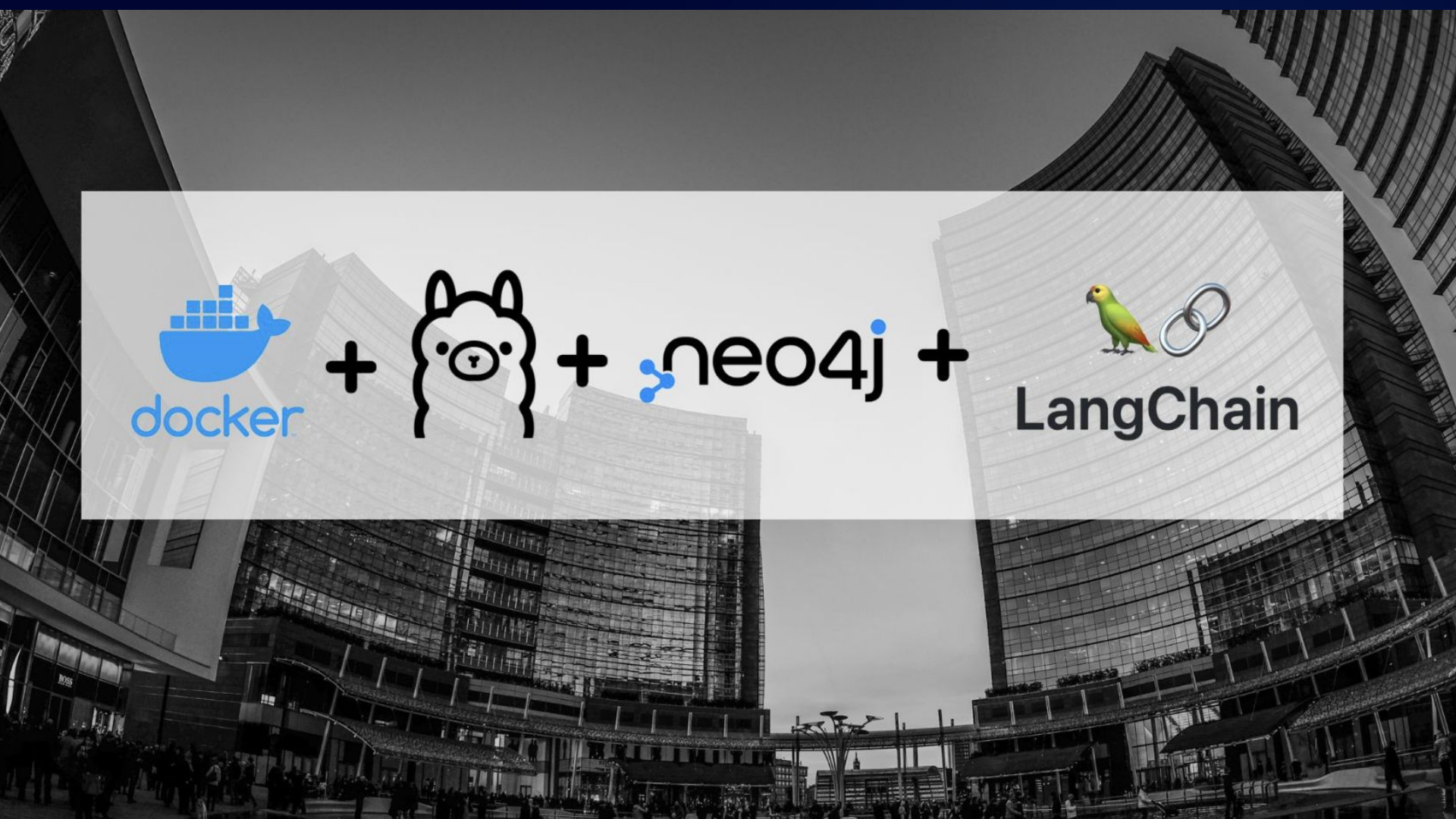
Further LLM Limitations

- Knowledge Cutoff: Not trained on the latest data but instead only up to a cutoff date which can be years in the past (GPT is only trained on pre Sep 2021 data)
- Lack of enterprise domain knowledge
- Lack of explainability and inability to verify or attribute sources for answers
- Ethical and data bias concerns
- Sensitive to prompt (input) phrasing
- Vulnerable to prompt injection

How to Help LLMs Do Better?

All of these are useful, but grounding is where **data** adds value

Fine-Tuning	Few-Shot Learning	Grounding
Provide additional training data to better tune GenAI to your use case	Provide completed examples “shots” to the AI as context in prompts. *a.k.a In-Context Learning*	Provide AI with the information to use for generating responses


docker
neo4j
LangChain

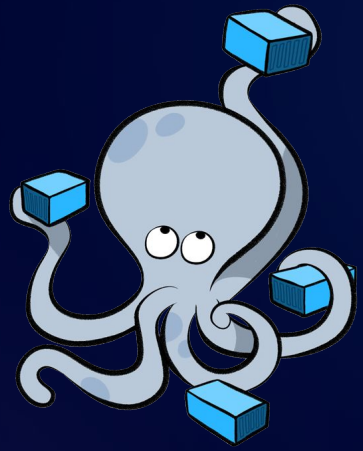
Components of GenAI Stack

- The stack is a set of Docker containers that make it easy to experiment with building and running Generative AI apps.
- The containers provide a dev environment of a pre-built, support agent app with data import and response generation use-cases.

Includes:

1. Ollama - A management tool for local LLMs (Ollama)
2. Neo4j - A database for grounding
3. GenAI apps based on LangChain
4. Pre-configured LLMs - A preconfigured Large Language Models such as Llama2, GPT-3.5, and GPT-4, to jumpstart your AI projects.

GenAI Stack - Docker Compose



FROM ollama	FROM langchain	FROM neo4j
local LLM management	GenAI apps in Python	Vector- & Graph Database

It’s Demo Time

https://github.com/docker/genai-stack

References

- Docker init Demo - https://github.com/dockersamples/docker-init-demos
- Docker Scout Demo - https://github.com/dockersamples/scout-demo-service
- Docker Scout CLI - https://github.com/docker/scout-cli
- Compose File Watch - https://github.com/dockersamples/avatars

simplicity  choice