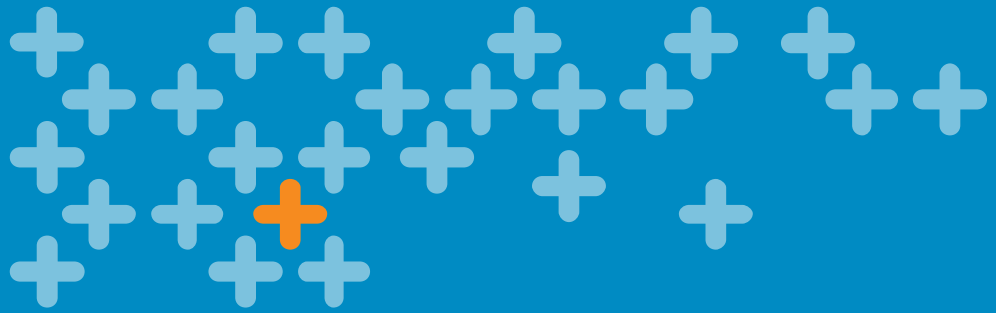


Keeping your builds Green using Docker

Jakob Ehn
@jakobehn
https://blog.ehn.nu

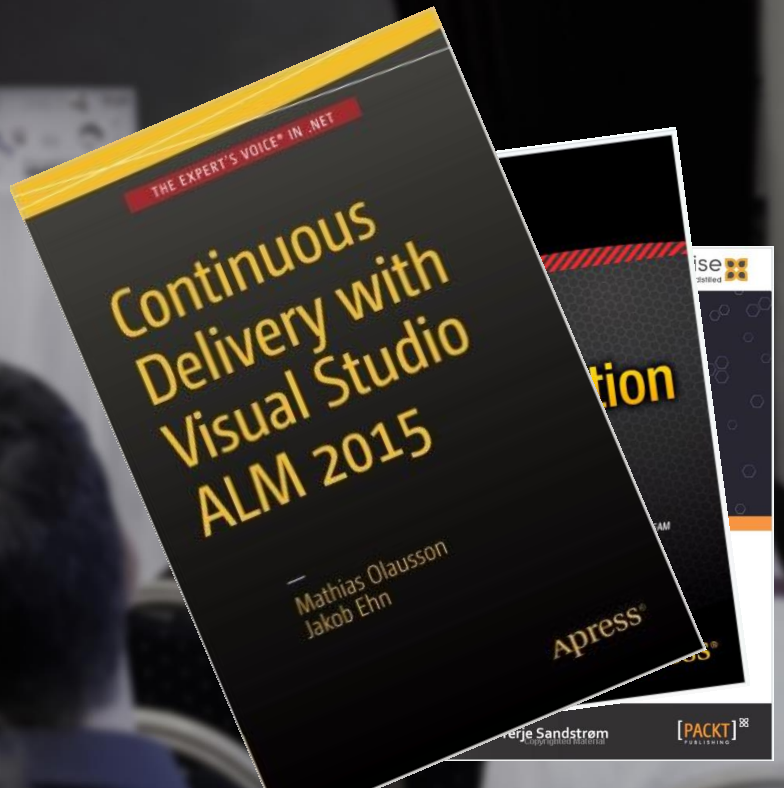
active SOLUTION

active SOLUTION

Microsoft Azure MVP

https://blog.ehn.nu

@jakobehn



Microsoft® Most Valuable Professional

Why do builds fail?

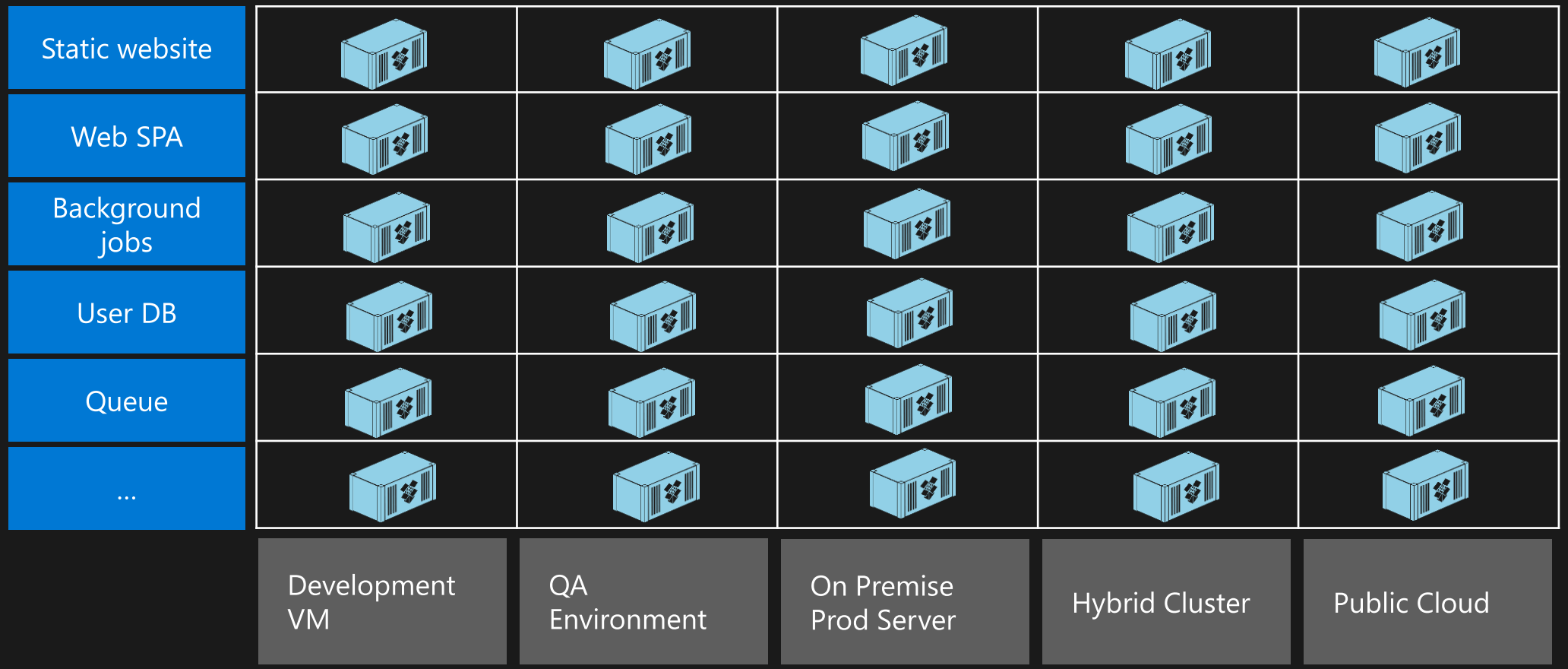

DEPENDENCIES
DEPENDENCIES
EVERYWHERE

Why do Builds Fail?

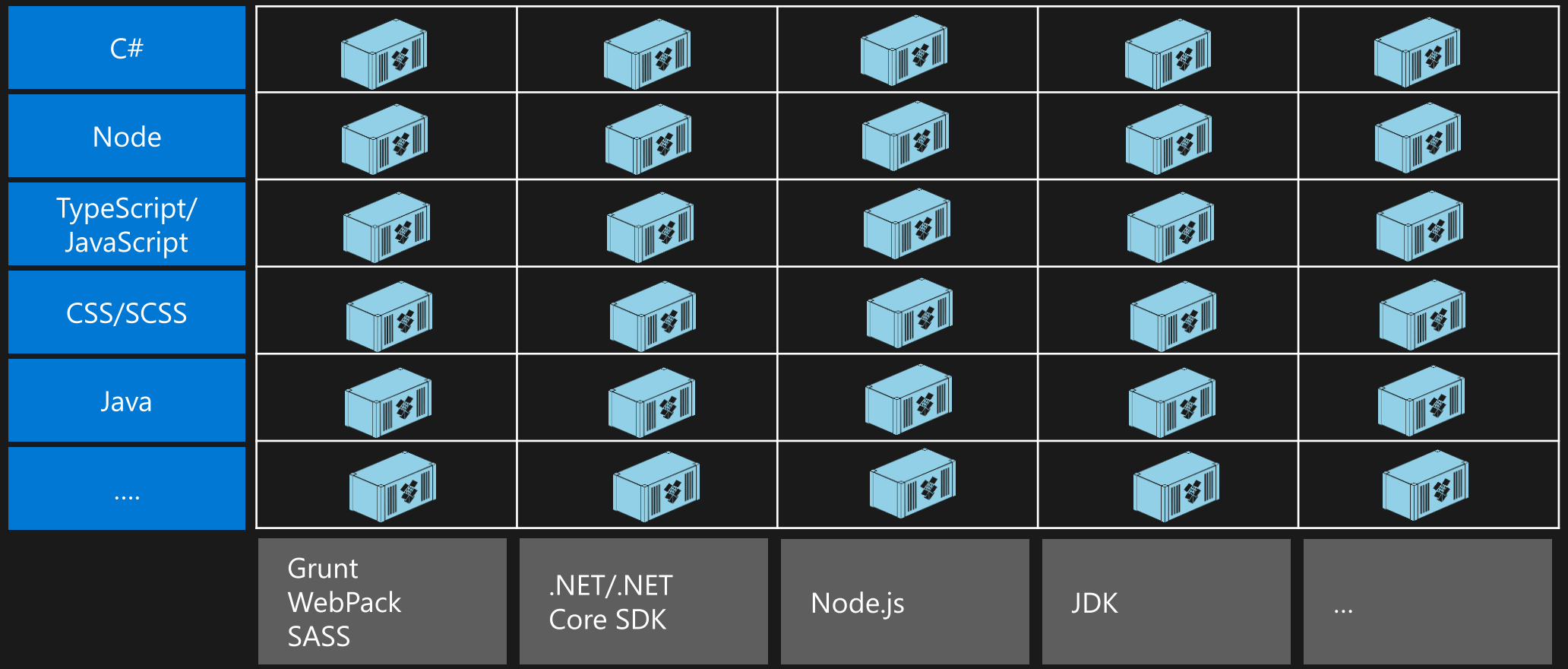
- Apps have many different dependencies (that changes all the time..)
- Developer machines installed/upgraded manually
- Build servers installed/upgraded manually
- CI builds often != Local builds

- Need consistency
- Run builds in a controlled, isolated environment

Dependency Hell

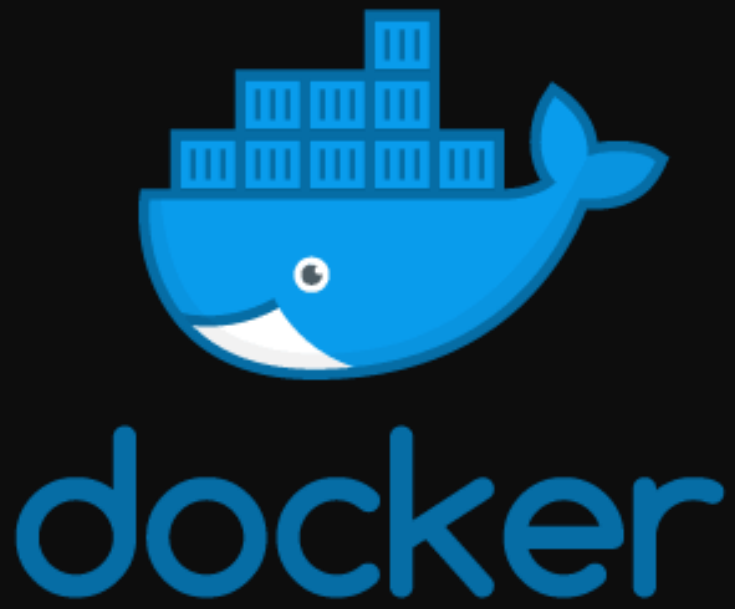


(Build) Dependency Hell

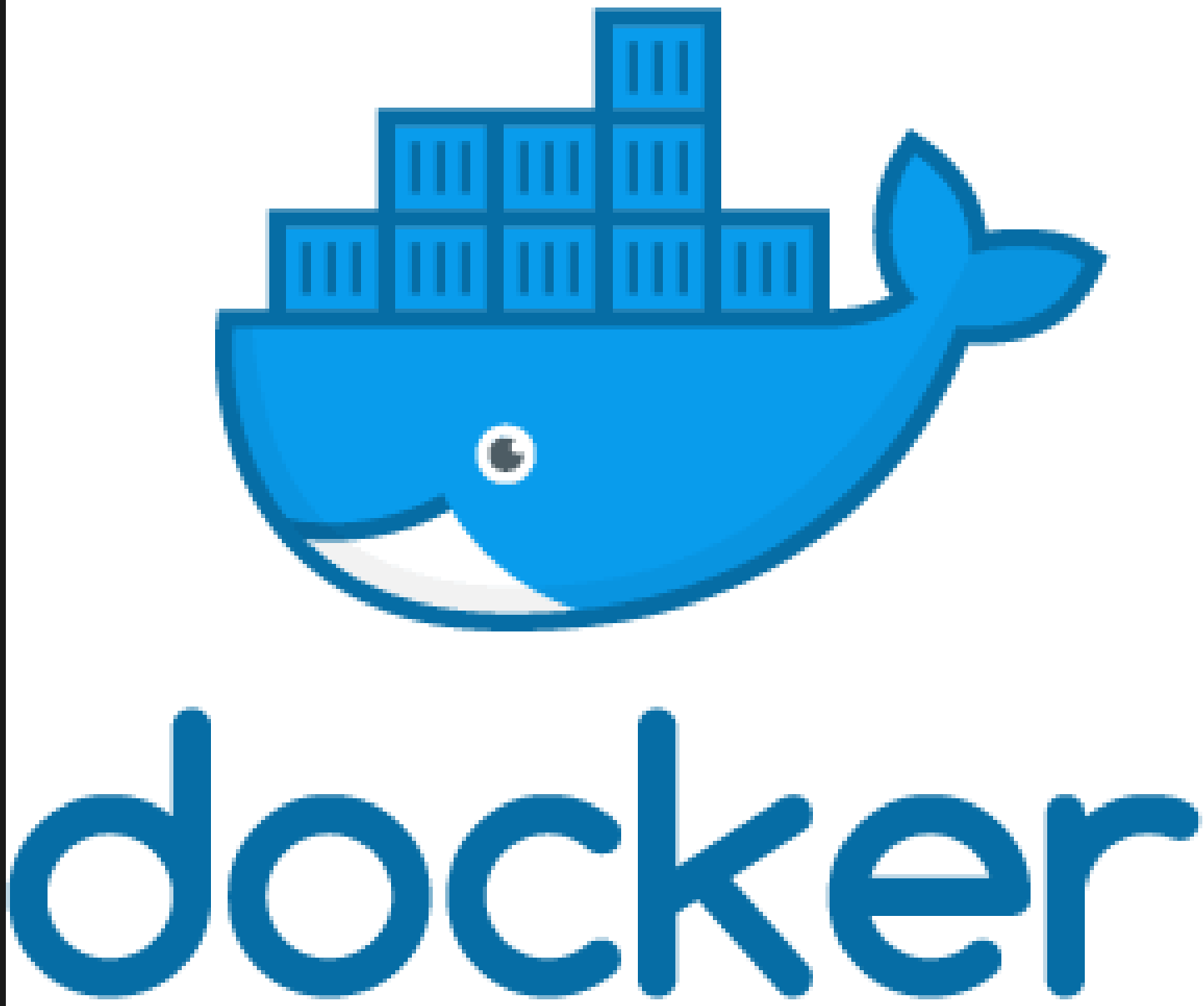


Two Approaches using Docker

- Build apps inside a container
  - The “build container” pattern
- Containerize the build environment
  - Builds can use any technology



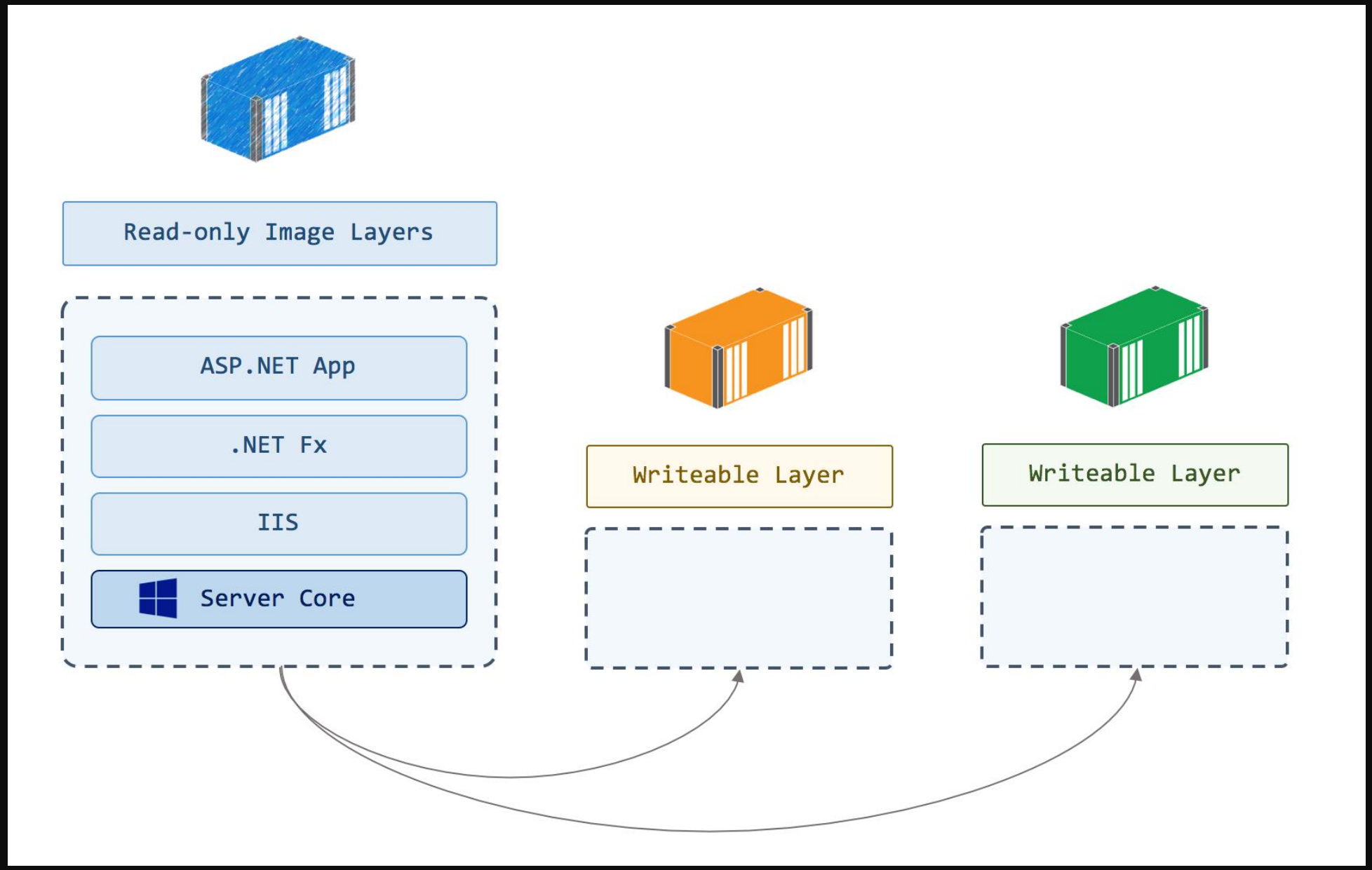
Dockerfile Builds



Dockerfile example

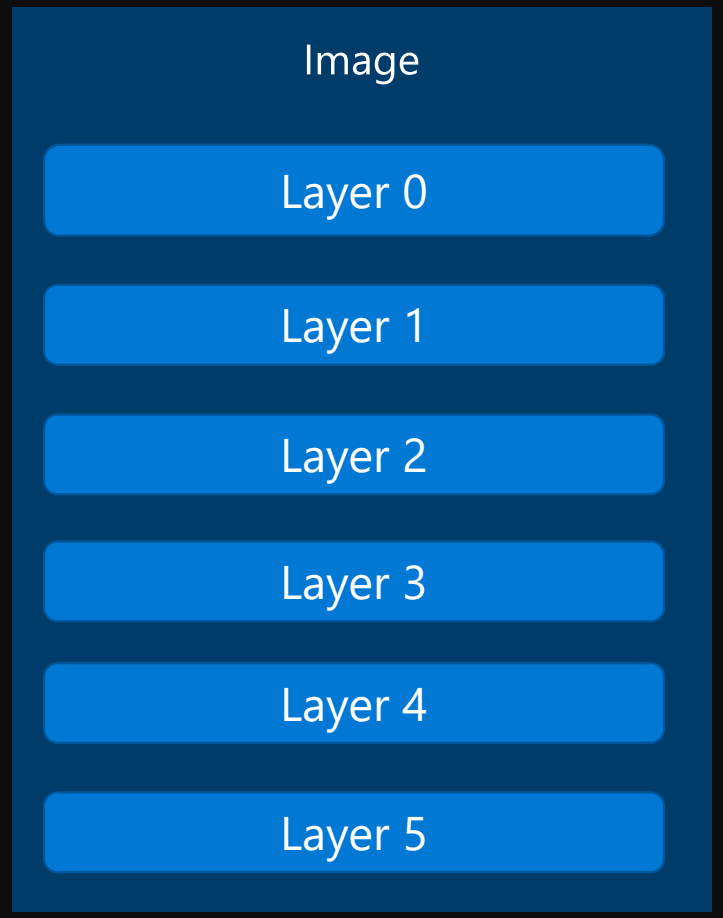
```
FROM node:8

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 8080

CMD [ "npm", "start" ]
```

Docker Image Layers



Docker Image Layers

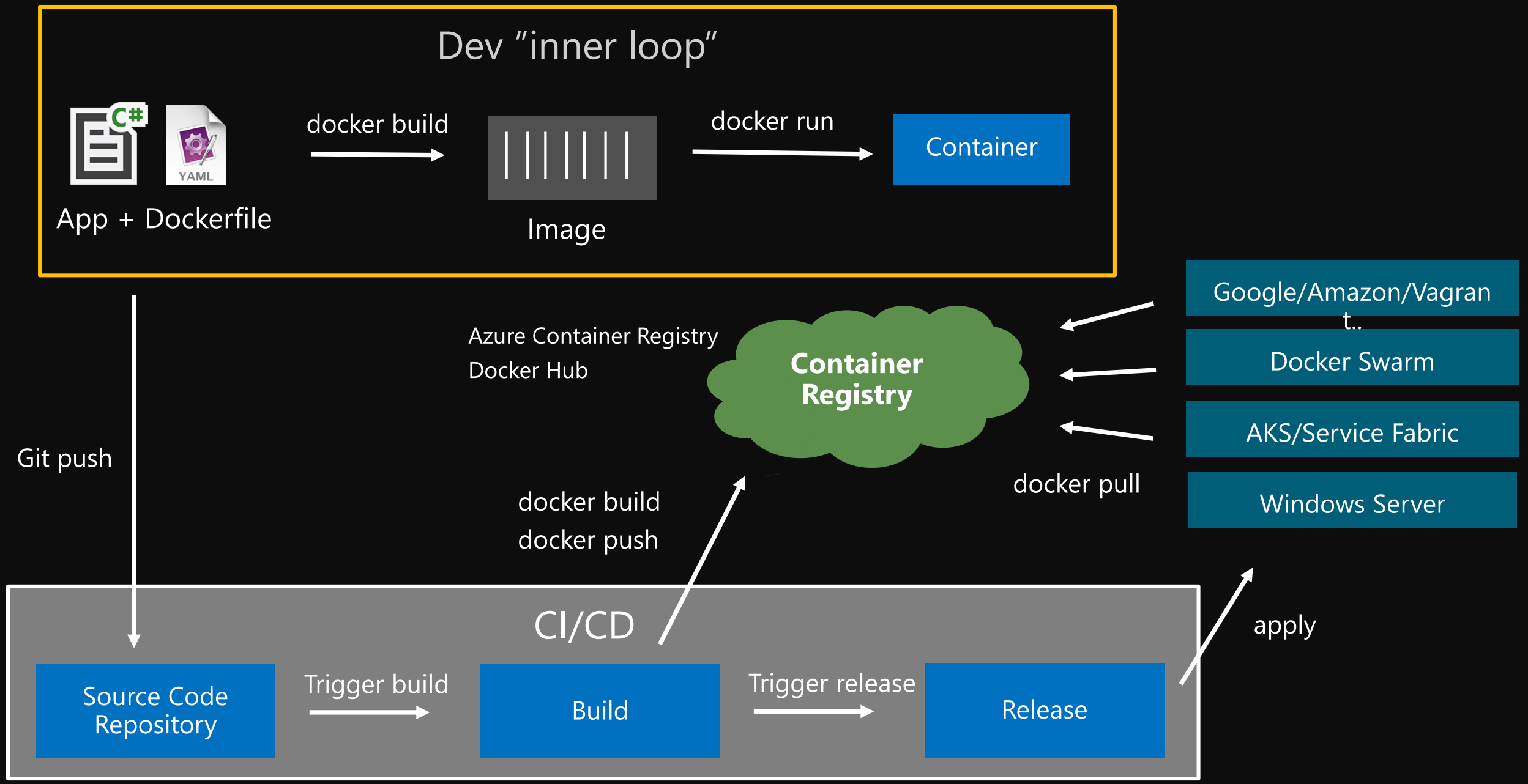
```
FROM node:8

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 8080

CMD [ "npm", "start" ]
```



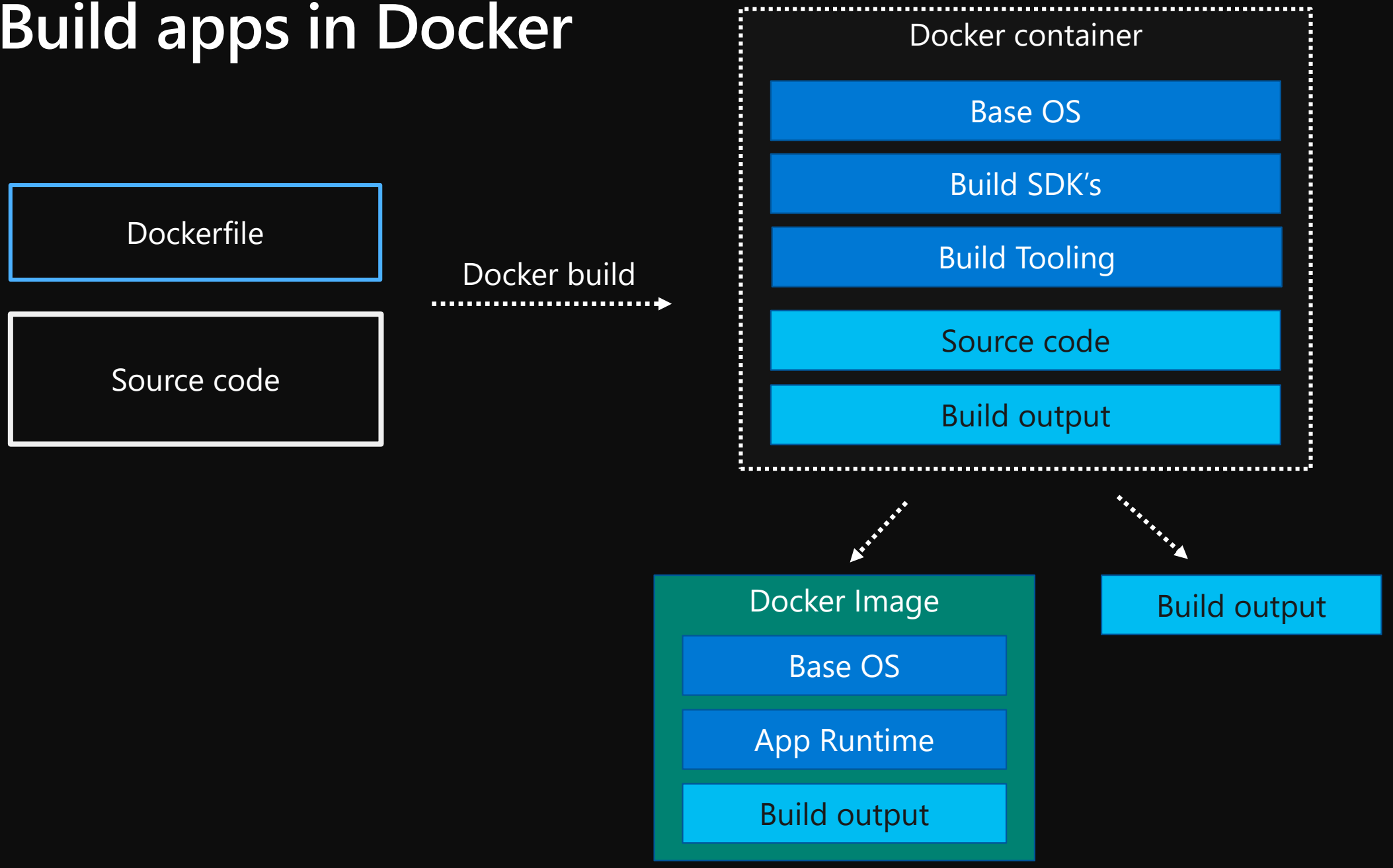
Docker multistage builds

```
FROM microsoft/dotnet/code/sdk:2.2 AS build
WORKDIR /src
COPY ["WebApplication1/WebApplication1.csproj", "WebApplication1/"]
RUN dotnet restore "WebApplication1/WebApplication1.csproj"
COPY . .
WORKDIR "/src/WebApplication1"
RUN dotnet build "WebApplication1.csproj" -c Release -o /app
RUN dotnet publish "WebApplication1.csproj" -c Release -o /app

FROM microsoft/dotnet/core/runtime:2.2
WORKDIR /app
COPY --from=build /app .
ENTRYPOINT ["dotnet", "WebApplication1.dll"]
```

Docker Workflow



Build apps in Docker



Demo – Docker builds

Jakob Ehn

Dockerfile Best Practices

- Optimize for size
  - Use multistage builds to minimize size
  - Minimize number of layers
  - Group related instructions into one command
  - Remove setup files
- Optimize for build duration
  - Ordering is important
  - Use a .dockerignore file
- Beware of caching

Dockerfile Builds Summary

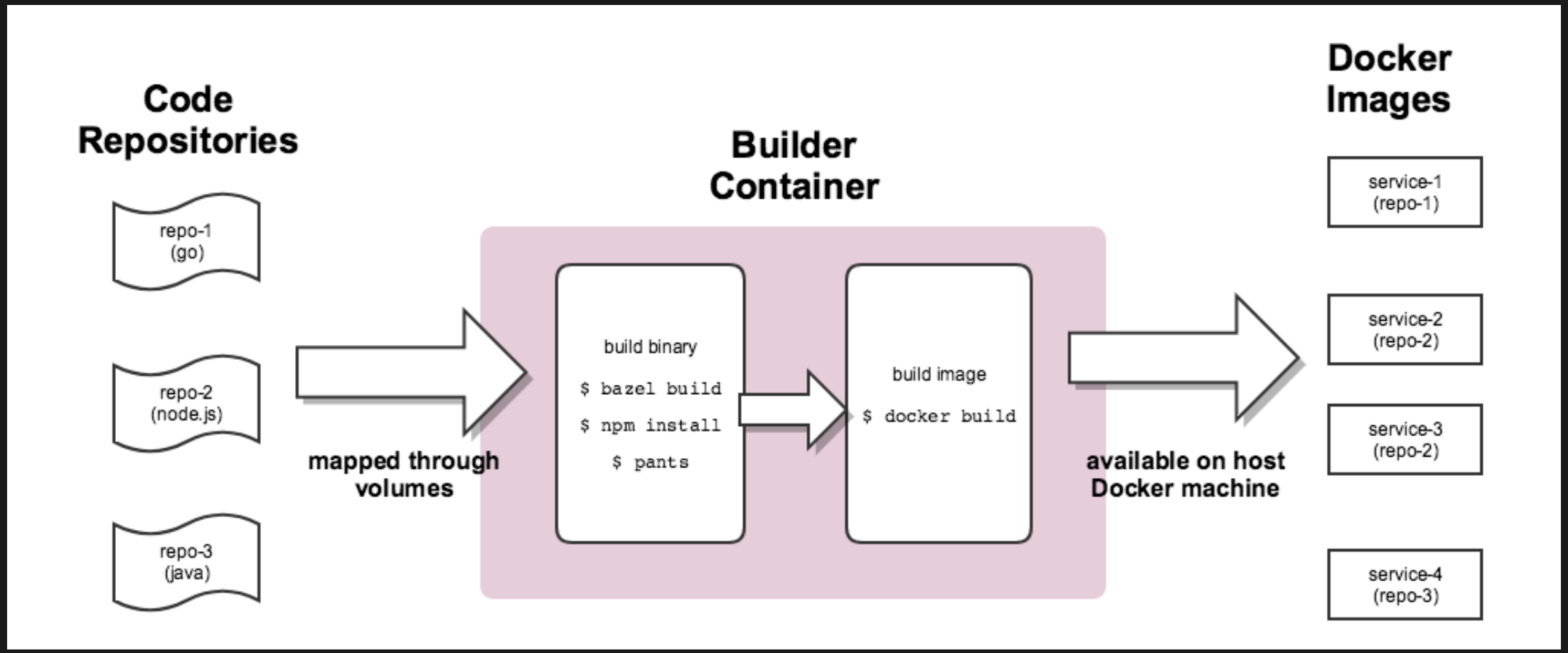
Pro's

- Full consistency
- Run everywhere with only Docker installed
- Can improve build times with Docker caching

Con's

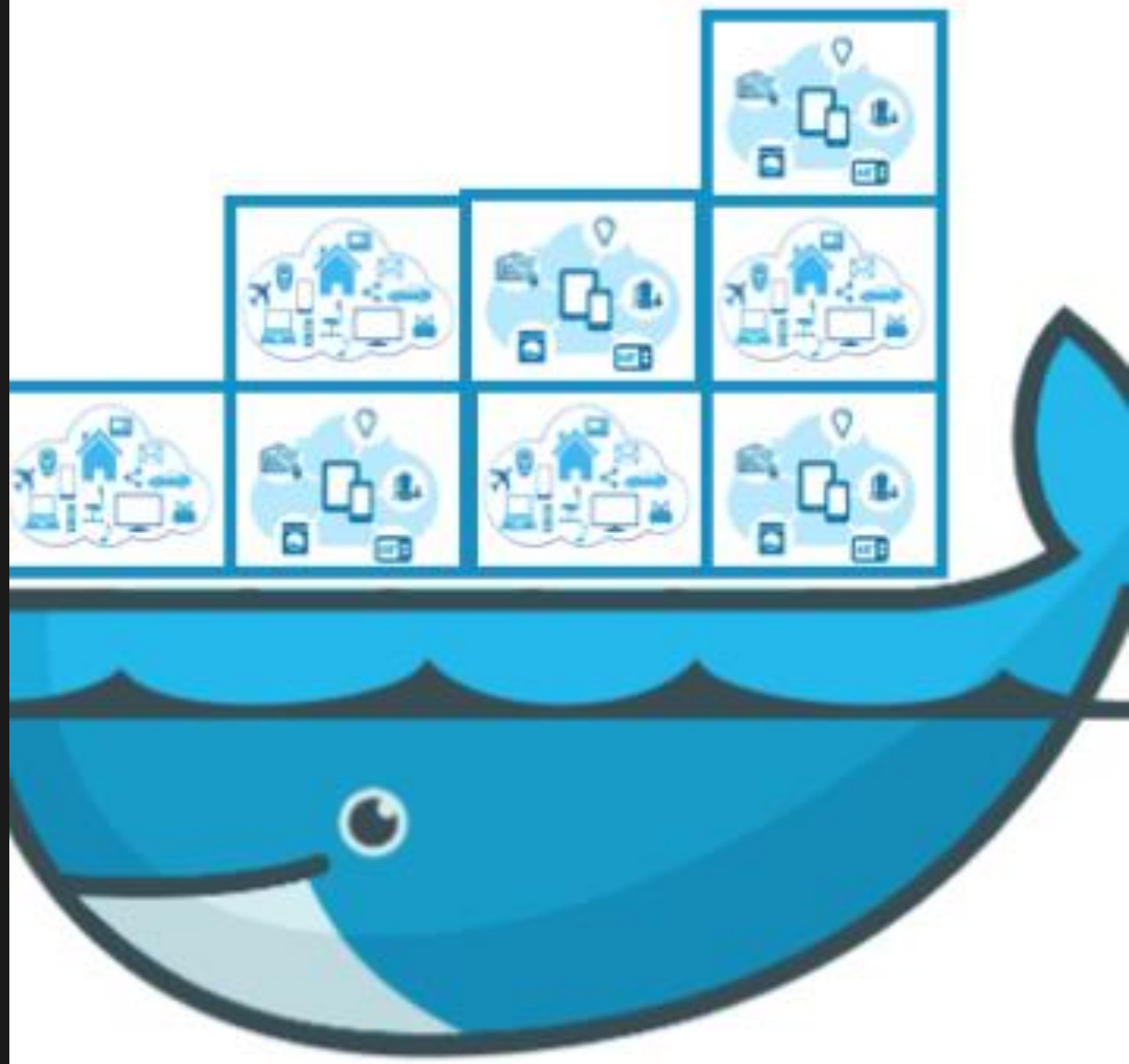
- All-in on Docker
- Need to extract output
- Hard to reuse existing build tasks
- Managing auth credentials

Example: Tinder

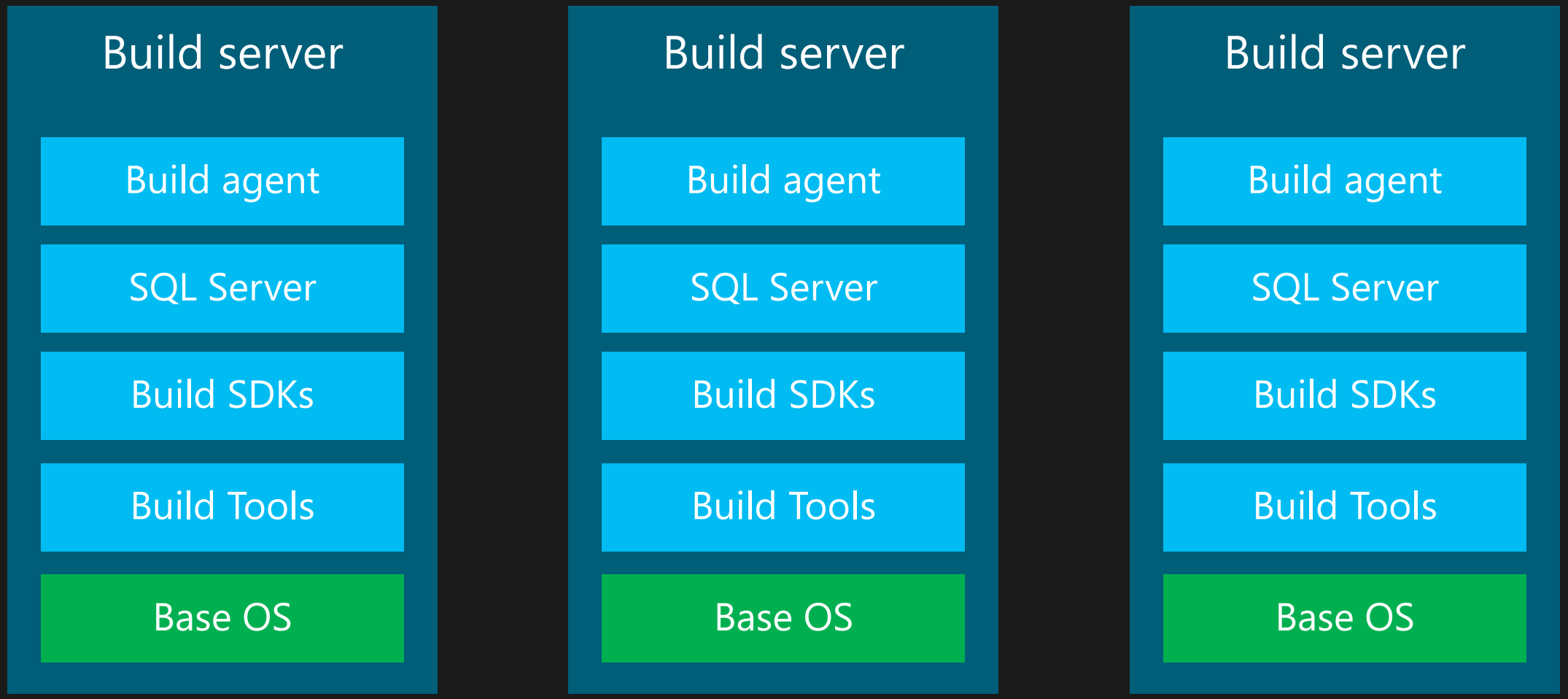


https://medium.com/@tinder.engineering/tinders-move-to-kubernetes-cda2a6372f44

Containerized Build Environments



Build Environments



Common Issues

- Build servers installed/upgraded manually
- Inconsistent state (snowflake servers)
- Hard to scale out
- Underutilized build servers
- Low build throughput

Containerized Build Environment

FROM microsoft/dotnet-framework:4.7.2-sdk-...

- SQL Server Express
- Visual Studio 2017 Build Tools
- .NET Core SDK
- SQL Server Data Tools
- ....
- Azure DevOps Build Agent

Dockerfile

docker build

Build Agent Image

Build agent

SDK's

SQL Server

docker pull
docker run

Build server

Build Agent Container

Build Agent Container

Build Agent Container

Running the Dockerized Build Agent

```
docker run -d
           -m 4GB
           --name <name>
           --storage-opt "size=50GB”
           --restart always

           -e TFS_URL=<url>
           -e TFS_PAT=<pat>
           -e TFS_POOL_NAME=<poolname>
           -e TFS_AGENT_NAME=<name>
           -e SA_PASSWORD=<sqlpwd>

           buildagent:1.0
```

Run multiple instances using Docker Compose

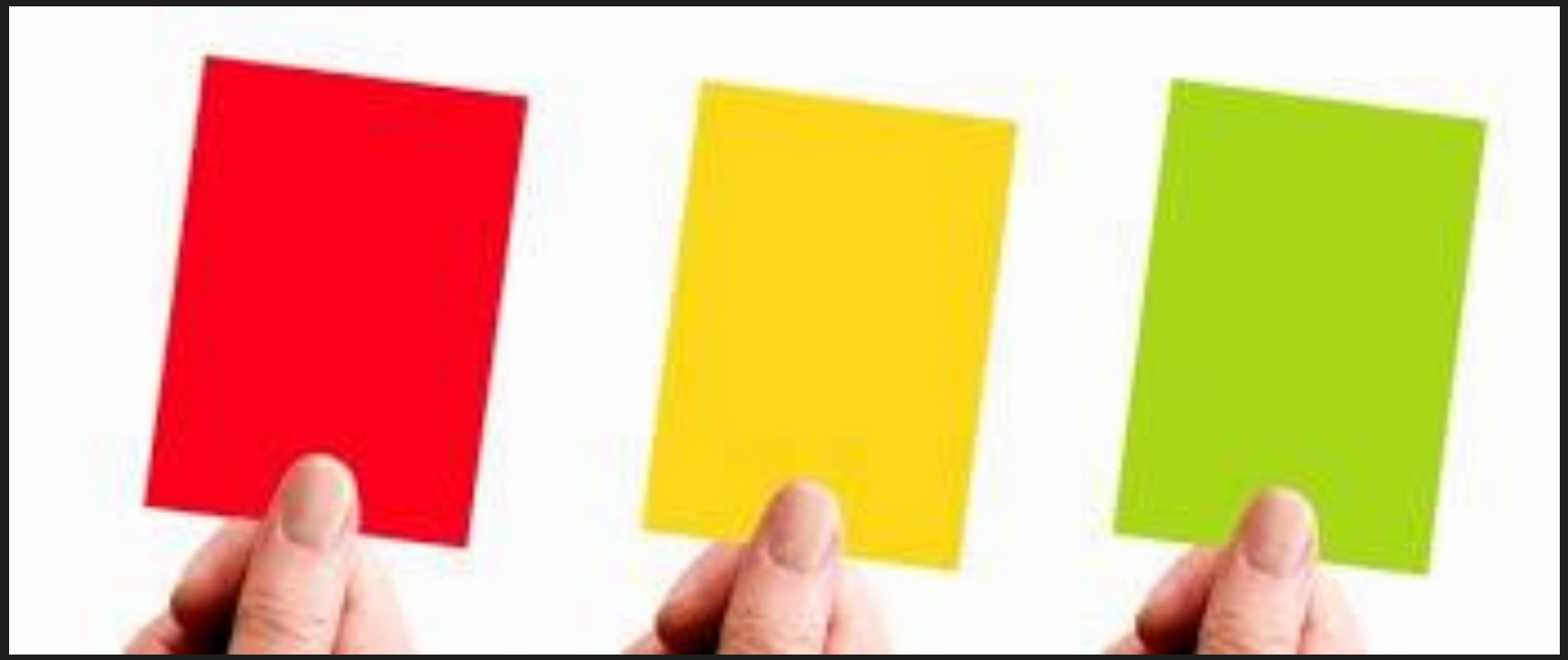
```yaml
services:

  agent1:
    image: ${IMAGE}:${VERSION}
    environment:
      TFS_AGENT_NAME: ${AGENTNAME}-1
    storage_opt:
      size: '${STORAGESIZE}'
    restart: always

 agent2:
    image: ${IMAGE}:${VERSION}
    environment:
      TFS_AGENT_NAME: ${AGENTNAME}-2
    storage_opt:
      size: '${STORAGESIZE}'
    restart: always
```

docker-compose up -d

Demo – Containerized Build Environments

Jakob Ehn

Please Evaluate The Session



Thank you!

Jakob Ehn
@jakobehn
https://blog.ehn.nu

active SOLUTION