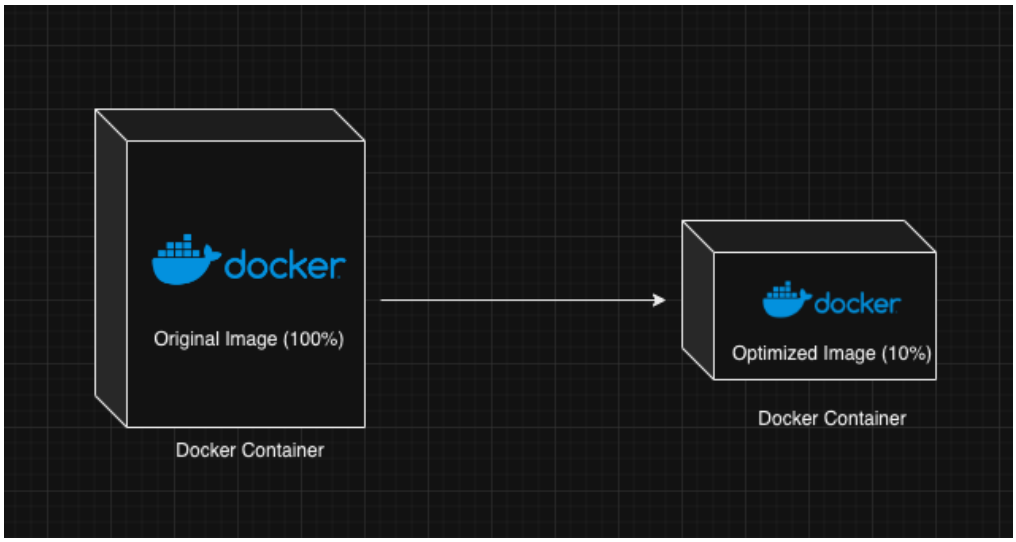
Image optimization



Optimizing Docker images is crucial for improving build times, reducing storage requirements, and enhancing security. Two effective strategies for achieving this are:

Multi-Stage Builds

Slim Base Images

Multi-Stage Builds

Multi-stage builds allow you to use multiple FROM statements in a Docker file, enabling you to separate the build environment from the runtime environment. This approach ensures that only the necessary artefacts are included in the final image, excluding build tools and unnecessary files.

Dockerfile Copy Edit

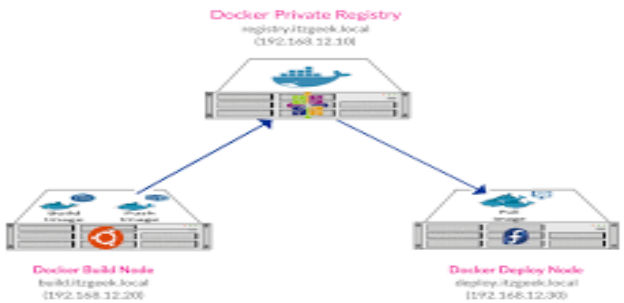
```
# Build stage
FROM golang:1.17 AS builder
WORKDIR /app
COPY . .
RUN go build -o myapp

# Production stage
FROM alpine:latest
WORKDIR /app
COPY --from=builder /app/myapp .
CMD ["./myapp"]
```

the Go application is built in the first stage using the golang:1.17 image. In the second stage, the compiled binary is copied into a minimal alpine: latest image, resulting in a smaller final image size.

Slim Base Images

Selecting a minimal base image can significantly reduce the size of your Docker images. Alpine Linux, for instance, is a lightweight distribution that serves as an excellent base for many applications.

Private docker hub



Setting Up a Private Docker Registry

1. Using Docker Hub's Private Repositories

Docker Hub offers private repositories where you can store your images securely.

Create a Private Repository: When creating a new repository on Docker Hub, you can set its visibility to private. This ensures that the repository is not publicly accessible and can only be accessed by you and the collaborators you specify.

Setting Up a Self-Hosted Private Registry

prefer to host your registry, Docker provides the official registry image.

Run the Registry: Use the following command to start a basic registry on port 5000:

docker run -d -p 5000:5000 --name registry registry:2

Using the Private Registry: Tagging and Pushing Images

Tag an Image: Before pushing an image to your private registry, tag it with the registry's address:

docker tag my-image localhost:5000/my-image

Push the Image: Push the tagged image to your registry:

docker push localhost:5000/my-image

Pulling Images:

docker pull localhost:5000/my-image