Reduce deployments runtime with GitHub Actions Cache

Here’s to incremental deployments 🥂.

About Me

- Head of Platform Engineering @Baania
- Often known as DevSecMLFinDataOps
- Faster deployments -> Faster iterations

kahnwong

Karnsiree Wong

karnwong.me

CI/CD Workflow

CI/CD Breakdown

- Image build
- Image push
- Image pull (during deployment)

Dockerfile

```
FROM node:18

WORKDIR /opt/build

COPY package.json .
COPY yarn.lock .
RUN yarn install

COPY . .

RUN yarn build

EXPOSE 3000
CMD [ "yarn", "start", "-H", "0.0.0.0" ]
```

GitHub Actions Buildx

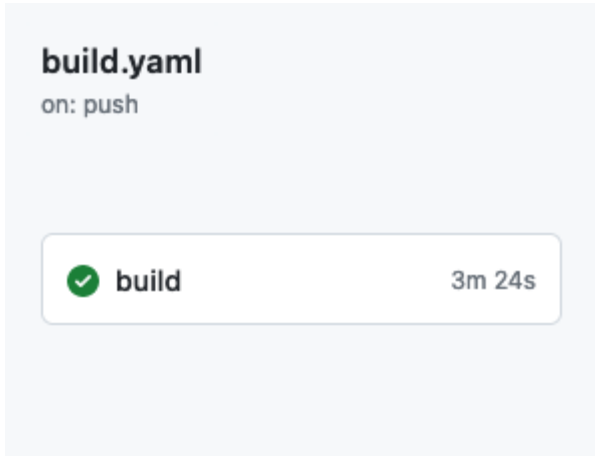
```yaml
- name: Build and tag image
  uses: docker/build-push-action@v5
  with:
    context: .
    builder: ${{ steps.buildx.outputs.name }}
    file: Dockerfile
    push: true
    tags: ${{ steps.meta.outputs.tags }}
    provenance: false
```

But we can cache docker layers

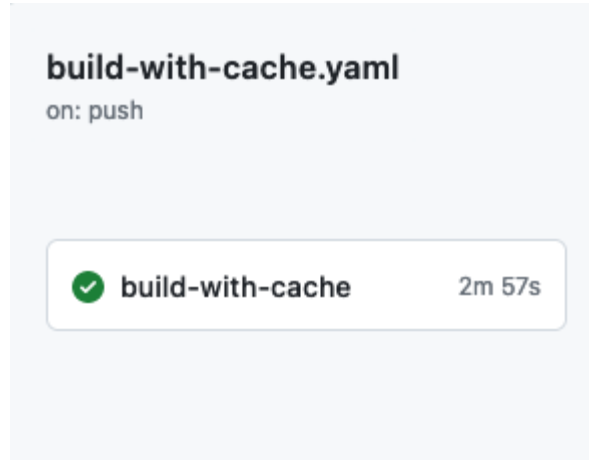
GitHub Actions Buildx with Cache

```yaml
- name: Build and tag image
  uses: docker/build-push-action@v5
  with:
    context: .
    builder: ${{ steps.buildx.outputs.name }}
    file: Dockerfile
    push: true
    cache-from: type=gha # add this
    cache-to: type=gha,mode=max # add this
    tags: ${{ steps.meta.outputs.tags }}
    provenance: false
```

Buildx



Buildx with Cache



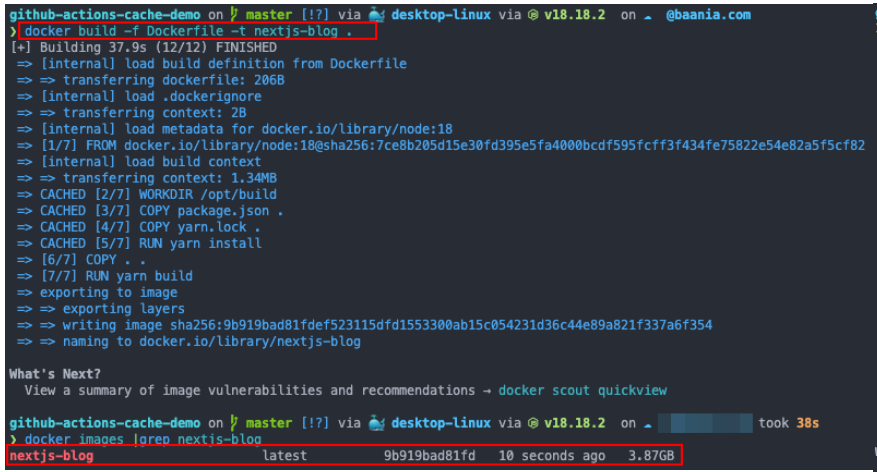
What about image size?

Dockerfile with multi-stage build

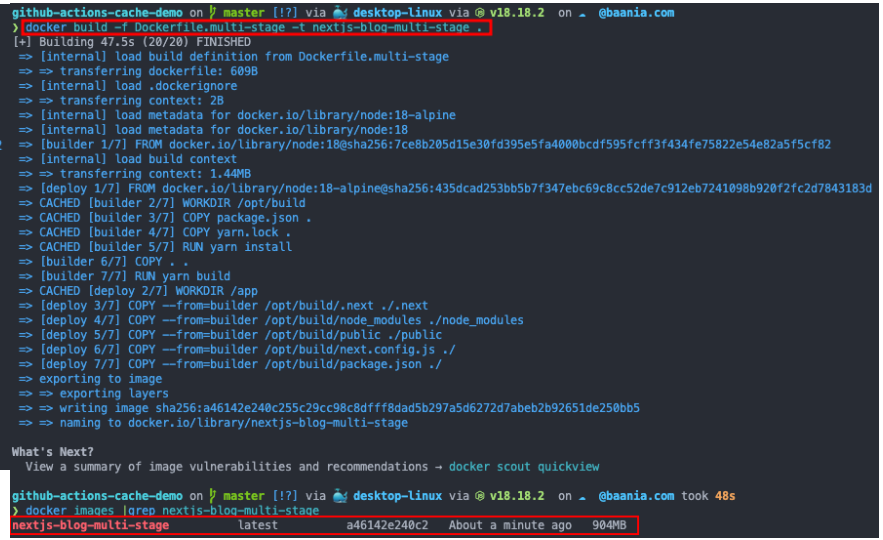
```dockerfile
# --------------- builder --------------- #
FROM node:18 AS builder

WORKDIR /opt/build

COPY package.json .
COPY yarn.lock .
RUN yarn install

COPY . .
RUN yarn build

# --------------- package --------------- #
FROM node:18-alpine AS deploy

WORKDIR /app
COPY --from=builder /opt/build/.next ./.next
COPY --from=builder /opt/build/node_modules ./node_modules
COPY --from=builder /opt/build/public ./public
COPY --from=builder /opt/build/next.config.js ./
COPY --from=builder /opt/build/package.json ./

EXPOSE 3000
CMD [ "yarn", "start", "-H", "0.0.0.0" ]
```

Let’s guess the image size!

Normal build



Multi-stage build



How much can we save?

Build time

- 30s per build (from 3m24s to 2m57s)

Image storage (compressed)

- 300MB per image (from 450MB to 150MB)

Ready for cost reduction?

If we deploy 150 times / month

Cost breakdown

Type	Actions Runtime	Image storage	AWS ECR Cost
No cache	3m24s * 150 = 510m	450MB * 150 = 66GB	66GB * 0.10 USD = 6.6 USD
With cache	2m57s * 150 = 442.5m	150MB * 150 = 22GB	22GB * 0.10 USD = 2.2 USD

In total, we can save 67.5m and 4.4 USD per month.

Check out the slides and repo!