Docker deep dive How we leverage the Docker stack

Ezri Zhu tzhu22@stevens.edu https://ezrizhu.com

Bluepring @ Stevens Institute of Technology

October 1, 2024

10-01 2024-1 Docker deep dive

Docker deep dive How we leverage the Docker stack

> Ezri Zhu tzhu22@stevens.edu https://ezrizhu.com

October 1, 2024

- 1. Hello, my name is Ezri Zhu I am a second year computer science undergrad at the Stevens Institute of Technology
- 2. I have been the VP of tech at blueprint since last year. There will be a Q&A at the end, tho feel free to interrupt me during the presentation for questions.

Ezri Zhu (Blueprint@Stevens)

Docker deep dive

October 1, 2024

メロト メタト メヨト メヨト



- Dockerfile, Docker Images
- Docker Container
- Docker Registry
- Docker-compose

Logo credit: https://github.com/Aikoyori/ProgrammingVTuberLogos/

10-01 2024-1 Docker deep dive



1. In this talk I will do a overview of the different components of a typical docker deployment, then we'll get into how exactly we're leveraging Docker at Blueprint

October 1, 2024

*ロト *個ト *注ト *注ト

What's Docker for? - The Bigger Picture

Docker simplifies software development and deployment by packaging applications and their dependencies into portable containers, that their behavior are the same across different environment.

- Developer develops their application
- Developer writes a Dockerfile, defining how to package their application in a docker container
- Developer builds the Dockerfile into an image, pushes to a registry
- Developer pulls the image from the registry on the deployment server
- At the same time, developer 2 can also pull the same image and test it on their machine

Docker deep dive

0-01

2024

What's Docker for? - The Bigger Picture

What's Docker for? - The Bigger Picture

- Docker simplifies software development and deployment by packaging applications and their dependencies into portable containers, that their behavior are the same across different environment.
- Developer develops their application
- Developer writes a Dockerfile, defining how to package their application in a docker container
- Developer builds the Dockerfile into an image, pushes to a registry
 Developer pulls the image from the registry on the deployment serve
- Developer pulls the image from the registry on the deployment serving At the same time, developer 2 can also pull the same image and tes it on their marhine
- 1. I will first give you a quick overview of how docker is used in a typical development cycle
- 2. We're mostly software developers here, and I am sure most of you have ran into the issue where someone's application doesn't work on someone elses computer
- 3. Those issues are usually solved by dependency issues, a certain python application may rely on a bunch of other python dependencies, and your systems package manager may also provide these packages but under different versions, thus also breaking seeemly working deployments
- 4. With docker, you define exactly what the environment is, from the base OS image that the container is built from, to unpacking your application

Linux Namespace via unshare

This is your computer, a program usually have access to all of these system resources provided by the Kernel.

By default, processes you call will inherit all of your namespaces



4/22

1. This is your computer, a program usually have access to all of these system resources provided by the Kernel.

- 2. By default, processes you call will inherit all of your namespaces, just like how when you prefix a command with sudo, it will inherit all of root's permissions
- 3. Here, you will notice three programs, a nginx reverse proxy, a postgres database, and my amazing webapp written in rust
- 4. As you can see, nginx is able to reverse proxy the webapp because they both share a network namespace
- 5. Then, the webapp is able to communicate with the postgres database via a socket, so that's done in the IPC namespace
- 6. There are other namespaces (PID, filesystem mounts, control groups, users), but we will pretend they don't exist for now for the sake of simplicity.

Linux Namespace via unshar



Linux Namespace via unshare



Docker deep dive

Linux Namespace via unshare

Linux Namespace via unshare

TD's Computer (he's a bit paranoid)

- 1. Here is an example of linux namespaces being used in the real world
- 2. TD has a computer and he really doesn't want to be tracked by ad companies and other organizations on the internet.
- 3. So he has three VPN setup in three separate linux network namespaces.
- 4. He then spawns a browser in each of the three namespaces, where he will work on different things.
- 5. This allows him to have three different IP addresses on the browser.

Docker deep dive

October 1, 2024

5/22

(日) (四) (日) (日) (日)

How overlayfs works



https://docs.kernel.org/filesystems/overlayfs.html

Docker	deep dive

10-01

2024-

Overlay's Hore readings under a discription where the stand discription where the stan

- 1. Overlayfs at minimum uses four directories, a lowerdir, a upperdir, a workdir, and a directry to mount the merged view of everything
- 2. lowerdir contains everything that were already in the system, and overlayfs will not write to it
- 3. the merged directory, which we labled as overlay view on the diagram, is where we will be interacting with overlayfs
- 4. upperdir is where overlayfs will write changes to when they are made in the merged directory.
- 5. workdir is where overlay stage changes to as it is copying files up from lowerdir to upperdir

Ezri Zhu	(Blueprint@Stevens)	
----------	---------------------	--

Docker deep dive

October 1, 2024

6/22

メロト 不得下 イヨト イヨト 二日

Beginning state

overlay (file 1) (file	2 File 3 File 4
upperdir	
lowerdin File 1 File	a)
lowerdir	File 3 File 4

	Docker	deep	dive
-01			
-10			
24-		└_Ov	erlayt
20			5

Overlayfs



- 1. For example, we have two directories, lowerdir 1, and lowerdir 2, and they each contains two files
- 2. You don't need to have two lower-directories, and when you first start overlayfs with two lowerdirs, it will just merge the two
- 3. You will have to give overlayfs three empty directories, one for the upperdir, one for the workdir, and one to mount the overlay view to
- 4. after overlayfs is mounted to the overlay view, you are able to see the four files from the two lower directories in from the overlay view

Ezri Zhu (Blueprint@Stevens)

Docker deep dive

7/22

October 1, 2024

< □ > < 同 > < 回 > < 回 >

Makes file 5, edits file 1

overlay (File 1) (File 2)	File 3 File 4	File 5
upperdir (File 1 (updates)		File 5
lowerdin (File 1) (File 2)		
lowerdir	File 3 File 4	

	Docker deep dive
-01	
-10	
024	└─Overlayfs
Ñ	

overlay elet H	43 (H45) (H	e V Re S
epentr [fig.t		Pie s
lowerster (Ha 1) (B)	4.8	
lovenin	Pla.5	la 4

1. Here we'll make a new file called file 5, and we will edit file 1

2. As you can see, overlayfs writes the updated file 1, and creates file 5 in upperdir, leaving lowerdir alone. However, the changes are reflected in the overlay view

October 1, 2024

8/22

メロト メロト メヨト メヨ

Deletes file 2

overlay (File 1)	File 3 File 4	file 5
upperdir (File 1 (updates) (whiteout file)		File 5
lowerdin (file 1) (file 2)		
lowerdir	File 3 File 4	

	Docker
Ξ	
4	
H	
4	
8	
Ñ	

└─Overlayfs

deep dive

Deletes file 2			
evenoy ele t	16.5	164	Ple
upperdir Rig 1 Rig 2 (unitarret file)			Pile
lowerdr (Hall) (Hall)			
loventr	(#6.5	04.4	

- 1. Here, we will delete file two. For file deletions, overlayfs will write a character device at the location of the file that it is removing, and the file will disappear from the overlay view
- 2. You probably didn't understand all of that, but the main takeaway is that overlayfs allows us to stack filesystems on top of each other like a hamburger, and if we want to add some files, we just add a lettuce on the top with more files, if we want to remove some files, we add a lettuce on the top with a hole at where the file is
- 3. And that's how docker images are built, its just a big hamburger of layers of filesystem states

October 1, 2024

9/22

Docker deep dive 2024-10-01 └─Overlayfs Overlayfs

And this is how you start an overlay mount

mount -t overlay overlay \ -o "lowerdir=\$lowerdir1,\$lowerdir2,upperdir=\$upperdir,workdir=\$workdir" "\$overlay"

Ezri Zhu (Blueprint@Stevens)

Docker deep dive

▲□▶▲圖▶▲≣▶▲≣▶ = ● ● ● October 1, 2024

\$ docker pull postgres Using default tag: latest latest: Pulling from library/postgres a803e7c4b030: Pull complete 009c876521a0: Pull complete 9c412905cca2: Pull complete 6463d4bf467a: Pull complete bd8b983728ed: Pull complete febc167f3560: Pull complete d73c81c4ade3: Pull complete 34b3b0ac6e9e: Pull complete 9bd86d074f4e: Pull complete 406f63329750: Pull complete ec40772694b7: Pull complete 7d3dfa1637e9: Pull complete e217ca41159f: Pull complete Digest: sha256:f1aaf6f8be5552bef66c5580efbd2942c37d7277cd0416ef4939fa34bf0baf31 Status: Downloaded newer image for postgres: latest docker.io/library/postgres:latest

Docker deep dive

10-01

2024-1

Overlayfs

Overlayfs I dasher pull pasagras Daing default sag: Latent Latent: Polling from likeary a020-2016/2018 Poll complete 6402/2018/2018 Poll complete 6402/2018/2018 Poll complete fabb/0372264: Poll complete fabb/037266: Poll complete fabb/037266: Poll complete fabb/037266: Poll complete Digent - sha256 (TaaHilkasUU2)so4665UU2sof642042s3 Status - Develoaded newer image for postgres lates desker - in/likeary/postgres latest

- 1. And this is also how docker works, when you build a container, each line in the dockerfile is being written to it's own upperdir, with everything preceding are just lowerdirs in a merged view
- 2. So when you pull a docker image and run it, each layer is just a lowerdir, and your container is mounted on the overlay view

October 1, 2024

11/22

*ロト *個ト *注ト *注ト - 注:

Dockerfile

A Quick Example

FROM node:20 WORKDIR / app COPY . .

RUN npm i

RUN npm run build

EXPOSE 3000

CMD ["npm", "run", "start"]

Docker deep dive 10-01 2024-1

-Dockerfile

A Quick Example FROM node:20 WORKDIR /app COPY RUN nom i RUN npm run build EXPOSE 3000 CMD ["npm", "run", "start"]

Dockerfile

- 1. This is a pretty standard Dockerfile for a node application, packed by npm
- 2. First, we pull the node:20 image, maintianed by the nice people over at nodejs, this allows us to pin our node version to be at 20
- 3. Second, we set the working directory to /app, this is the same as doing cd in the container

Ezri Zhu (Blueprint@Stevens)

Docker deep dive

October 1, 2024

*ロト *個ト *注ト *注ト - 注:

Building a docker image

docker build -t ghcr.io/stevensblueprint/project:ezri-latest .

2024-10-01

Building a docker image

Docker deep dive

- 1. In the same directory as the Dockerfile, we can now build the docker image
- 2. Docker images are named via labels, a image can have mutlple labels
- 3. Each image has a hash, it's basically the image's unique ID
- 4. But we can also give it a more human readable name, consisted of a path and a tag
- 5. usually a path denotes what the application is, such as the name of the repository
- 6. and a tag denotes the version that the application is

Building a docker image

October 1, 2024

13/22

A D F A B F A B F

Building a docker image

docker build -t app .

docker tag app ghcr.io/stevensblueprint/project:ezri-latest docker tag app ghcr.io/stevensblueprint/project:latest docker tag app ghcr.io/stevensblueprint/project:staging 2024-10-01

Building a docker image

Docker deep dive

- 1. In the same directory as the Dockerfile, we can now build the docker image
- 2. Docker images are named via labels, a image can have mutlple labels
- 3. Each image has a hash, it's basically the image's unique ID
- 4. But we can also give it a more human readable name, consisted of a path and a tag
- 5. usually a path denotes what the application is, such as the name of the repository
- 6. and a tag denotes the version that the application is

Building a docker image

focker build -t app

docker tag app ghcr.io/stevensblueprint/project:ezri-latest

focker tag app ghcr.io/stevensblueprint/project:latest focker tag app ghcr.io/stevensblueprint/project:staging

Ezri Zhu (Blueprint@Stevens)

Docker deep dive

October 1, 2024

14 / 22

イロト 不得下 イヨト イヨト

Pushing a docker image

docker push ghcr.io/stevensblueprint/project:ezri-latest docker push ghcr.io/stevensblueprint/project:latest docker push ghcr.io/stevensblueprint/project:staging Docker deep dive

Pushing a docker image

docker push ghcr.io/stevensblueprint/project:ezri-latest docker push ghcr.io/stevensblueprint/project:latest docker push ghcr.io/stevensblueprint/project:staging

Pushing a docker image

- 1. Now, we can use docker push to push it to the docker registery
- 2. In our case, we're using ghcr.io, which is the github container registry
- 3. That's basically a place where we store our container images, and we can pull them from other places

Ezri Zhu (Blueprint@Stevens)

Docker deep dive

October 1, 2024

A D N A B N A B N A B N

Launching a docker container

Docker deep dive

-Launching a docker container

Launching a docker container

docker run -name app-test -p 8080:8080 ghcr.is/stevensblueprint/project:ezri-latest Full options here: https://docs.docker.com/reference/cli/docker/container/run,

docker run –name app-test -p 8080:8080 ghcr.io/stevensblueprint/project:ezri-latest Full options here:

https://docs.docker.com/reference/cli/docker/container/run/

October 1, 2024

イロト 不問 トイヨト イヨト 二三

Writing a docker compose file

```
services:
  redis :
    image: redis:latest
    restart: always
    ports:
      - "6379:6379"
  api:
    build :
      dockerfile: Dockerfile
      context: .
    restart: always
    ports:
      - "8080:8080"
    depends_on:
      – redis
                                      Ezri Zhu (Blueprint@Stevens)
                         Docker deep dive
                                            October 1, 2024
```

Docker deep dive 2024-10-01 Writing a docker compose file

17 / 22

Writing a docker compose file

services redis: image: redis:latest restart: always ports: - "6379:6379" api: build: dockerfile: Dockerfile context: restart: always ports: - "8080:8080" depends.on: - redis

Launching a docker compose

Docker deep dive

Launching a docker compose

Launching a docker compose

docker compose up (-d) docker compose down docker compose ps docker compose logs

docker compose up (-d) docker compose down docker compose ps docker compose logs

 < □ > < ⊡ > < ⊡ > < ⊡ > < ⊡ > < ⊡ >
 ⊇

 October 1, 2024

Recap

Docker deep dive

Dockerfile: Delines a docker image Docker limage: A portable runtime environment Docker registry: A place to tocker images Docker container: A running instance of a docker image Docker compose: A collection of docker containers and its desendencies

Recap

Dockerfile: Defines a docker image Docker Image: A portable runtime environment Docker registry: A place to store docker images Docker container: A running instance of a docker image Docker compose: A collection of docker containers and its dependencies

October 1, 2024

19/22

イロト 不問 とくほと 不良とう 良

nuances missed

OCI Containers vs Docker contianers Look into: github.com/containers/crun Also see also: mobyproject.org/ Docker deep dive

10-01

2024-1

OCI Containers vs Docker contianers Look into: github.com/containers/crum Also see also: mobyproject.org/

nuances missed

- 1. docker is not the only way to do linux contianers, but for the sake of simplicity this ppt only talked ab dockers
- 2. Feel free to also ask about how we're using github actions to build and test, and deploy docker containers in our CI/CD workflow

October 1, 2024

20 / 22

*ロト *個ト *注ト *注ト - 注

related tech

firecracker MVM (orig): firecracker-microvm.github.io firecracker MVM (flyio): fly.io/blog/sandboxing-and-workload-isolation v8 isolates (cf talk) www.infoq.com/presentations/cloudflare-v8 v8 isolates (cf security) developers.cloudflare.com/workers/reference/security-model v8 isolates (deno edition) deno.com/blog/anatomy-isolate-cloud v8 isolates (deno edition) blog/anatomy-isolate-cloud

v8 isolates (deno but diff) blog.val.town/blog/first-four-val-town-runtimes Nix, Flakes, NixOS

Docker deep dive

10-01

2024-

Firscacker MVM (orig): Firscacker-microvm github in firscacker MVM (flyio): Firsio Jobg (and boxing and-workbad-indiction of biotexis (cf 1 and biow midle) com presentation (choculture-v8 v8 indices (cf 1 and bio); for an one primeration (choculture-v8 v8 indices (cf 1 and bio); deno com (biog (antomy-indices - load v8 indices (deno bac dRT) biog val Loavy-biog (First-four-val-tour-nutrime Nr.; Falues, NoCS

related tech

- 1. basically other ways we run/scale applications
- 2. happy to talk ab them after the talk

October 1, 2024

21/22

イロト 不問 とくほと 不良とう 良

Thank you!

Docker deep dive Teak you booker deep dive Contraction Contractio

Questions?

October 1, 2024

★白▶★御▶★臣▶★臣▶ 三臣