UbuCon Korea 2023

닷넷과 함께하는 경량 컨테이너

부제: 얼마까지 알아보셨어요?

Table of Contents

Introduce the speaker

김정우 @CLOUDMATE

Jungwoo (Jayden) Kim

jungwoo.kim1230

dev-jungwookim

rokag3@gmail.com

SW Developer

Service Development Team Lead of CLOUDMATE

주로 .NET, Go, Python을 사용하여 web, server, client app을 개발하고 있으며,
개발한 서비스가 서비스 목적을 달성하는지에 집중하고 있습니다.

2005 ~ 2009 : SI 스타트업에서 커리어 시작. 주로 영림원, LG CNS 프로젝트에 투입.

2009 ~ 2020 : 모두투어 ERP 신규 개발 및 유지 보수.

2020 ~ 현재 : 클라우드메이트 서비스개발팀에서 다양한 B2B SaaS 개발

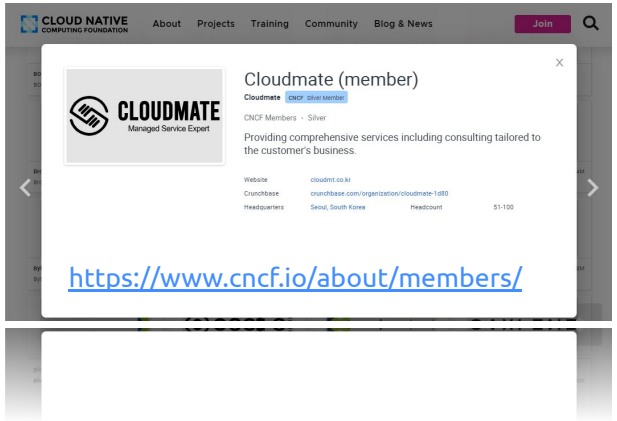
Introduce our company CLOUDMATE

CLOUDMATE
Managed Service Expert

https://cloudmt.co.kr/

매니지드 서비스 전문 기업 (MSP) 으로서 고객이 클라우드의 이점을 최대화하여 사용할 수 있도록 하는 클라우드 네이티브를 미션으로 가지고 있습니다.

비즈니스 모델 개발, 운영 프로세스 변화 등 클라우드에 대한 다양한 고객 니즈를 이해하고 성공적인 클라우드 도입을 위해 컨설팅과 마이그레이션을 수행하며, 이후 안정적인 클라우드 운영을 위한 매니지드 서비스와 교육 및 기술지원 서비스를 제공하고 있습니다.



The technical level for this session is

Beginner level

2. What is Lightweight Container ?

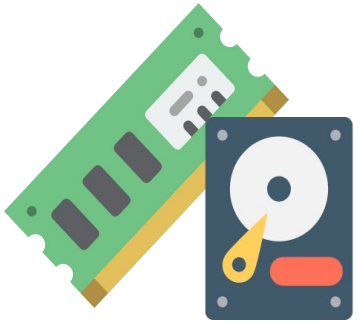
2. What is Lightweight Container ?

컨테이너 크기가 경량이면 얻을 수 있는 장점은

엄청 많다.

2. What is Lightweight Container ?

빠른 배포
빠른 시작

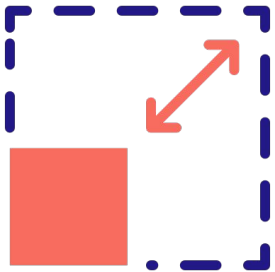


자원 효율성

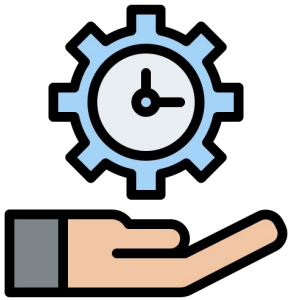
클라우드 비용 절감

2. What is Lightweight Container ?

보안
(공격 벡터 최소화)



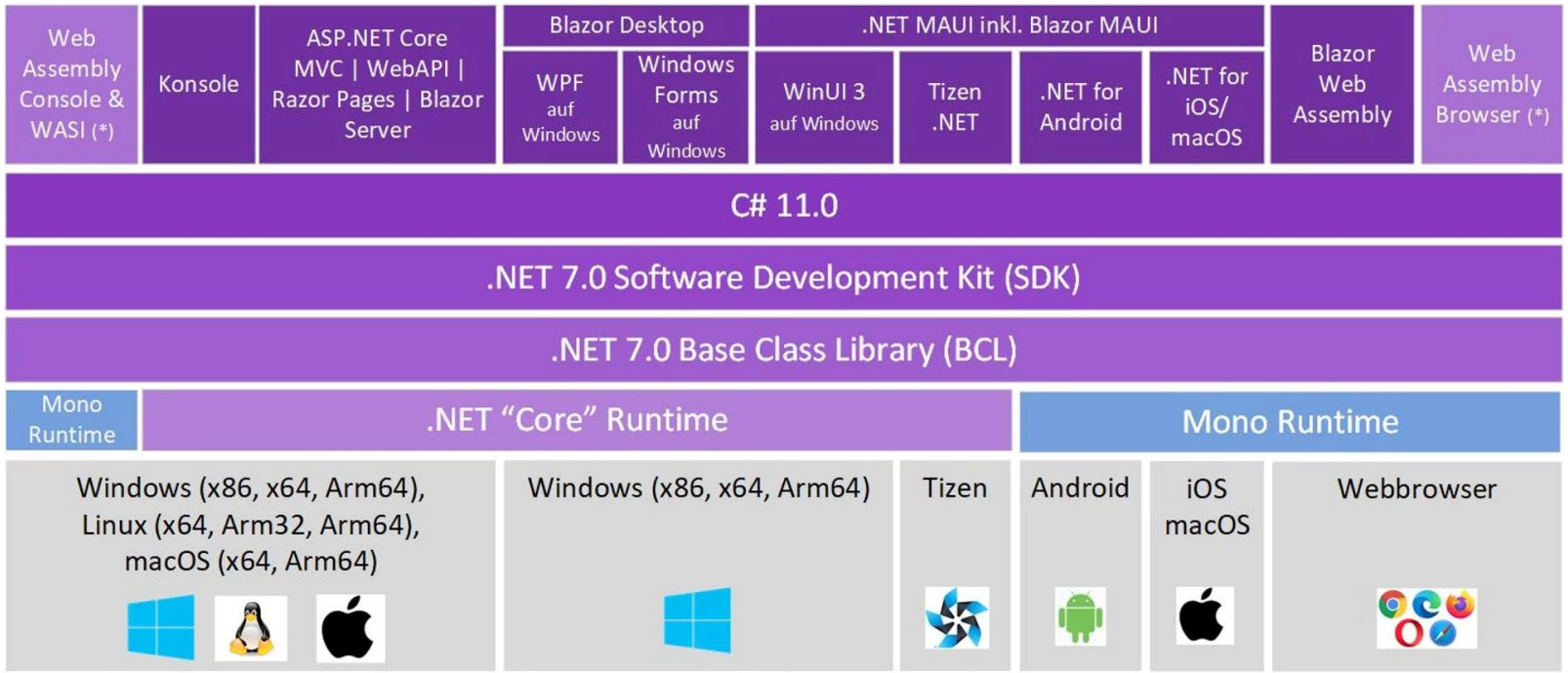
빠른 스케일링



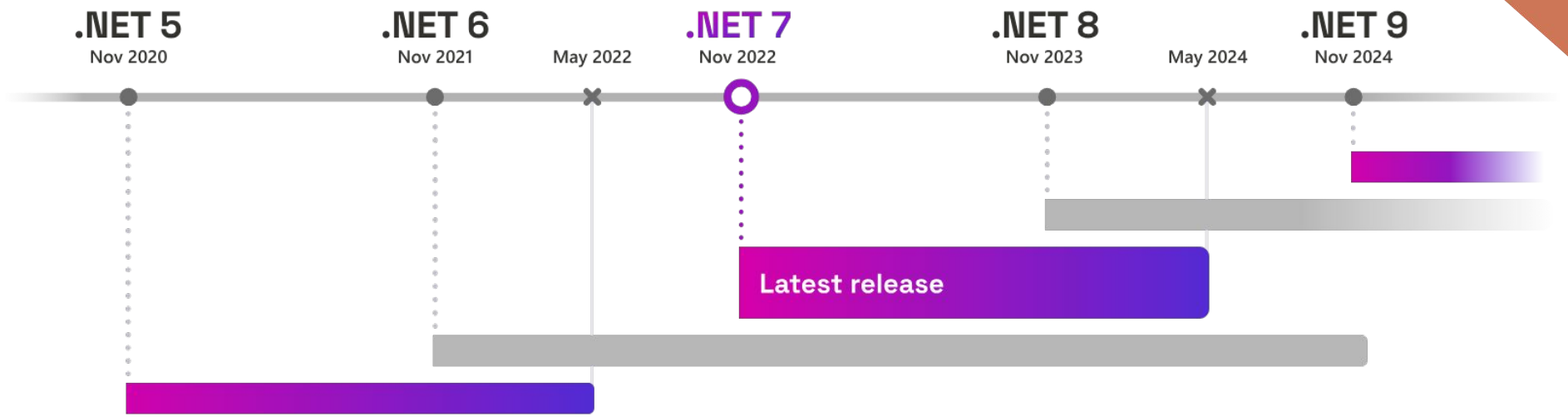
관리 용이성

3. Introduction to .NET 7

3. Introduction to .NET 7



3. Introduction to .NET 7



4. ASP.NET 7.0 image containerization

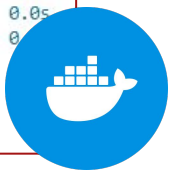
4. ASP.NET 7.0 image containerization

```dockerfile
# Docker hub - ASP.NET Core Runtime 6.0 / 7.0 -> https://hub.docker.com/_/microsoft-dotnet-aspnet
FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 8080
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
WORKDIR /src
COPY ["WebAPI20230803.csproj", "."]
RUN dotnet restore "./WebAPI20230803.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "WebAPI20230803.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "WebAPI20230803.csproj" -c Release -o /app/publish /p:UseAppHost=false

FROM base AS final
ENV TZ=Asia/Seoul
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "WebAPI20230803.dll"]
```

Dockerfile

4. ASP.NET 7.0 image containerization

```
PS C:\BizData\source\WebAPI20230803\WebAPI20230803> docker build -f Dockerfile --force-rm --no-cache -t webapi-aspnet .
[+] Building 14.1s (18/18) FINISHED
 => [internal] load .dockerignore                                                                                    0.0s
 => => transferring context: 2B                                                                                      0.0s
 => [internal] load build definition from Dockerfile                                                                 0.0s
 => => transferring dockerfile: 729B                                                                                 0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0                                                    0.2s
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0                                                 0.2s
 => [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:bdcfb498261ca18f023ac67615d814ea743aa3288eb880855fa2eb86c6313ccc  0.0s
 => [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:7.0@sha256:b236eb1df512af4d8034ce8f65f9e8740f1845d894b7906a4b1b66890946c03f  0.0s
 => [internal] load build context                                                                                    0.0s
 => => transferring context: 37.72kB                                                                                 0.0s
 => CACHED [build 2/7] WORKDIR /src                                                                                  0.0s
 => CACHED [base 2/2] WORKDIR /app                                                                                   0.0s
 => CACHED [final 1/2] WORKDIR /app                                                                                  0.0s
 => [build 3/7] COPY [WebAPI20230803.csproj, .]                                                                      0.0s
 => [build 4/7] RUN dotnet restore "./WebAPI20230803.csproj"                                                         6.3s
 => [build 5/7] COPY . .                                                                                             0.5s
 => [build 6/7] WORKDIR /src/.                                                                                       0.0s
 => [build 7/7] RUN dotnet build "WebAPI20230803.csproj" -c Release -o /app/build                                    4.7s
 => [publish 1/1] RUN dotnet publish "WebAPI20230803.csproj" -c Release -o /app/publish /p:UseAppHost=false         2.0s
 => [final 2/2] COPY --from=publish /app/publish .                                                                   0.0s
 => exporting to image                                                                                               0.1s
 => => exporting layers                                                                                              0.0s
 => => writing image sha256:e0467842bb26bca75d5f1353df52898341f9271701bef54f6de23bbe0b3aa5e8                         0.0s
 => => naming to docker.io/library/webapi-aspnet                                                                     0

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
```



4. ASP.NET 7.0 image containerization

```
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
PS C:\BizData\source\WebAPI20230803\WebAPI20230803> docker run -p 8080:8080 webapi-aspnet
warn: Microsoft.AspNetCore.Server.Kestrel[0]
      Overriding address(es) 'http://0.0.0.0:8080'. Binding to endpoints defined via IConfiguration and/or UseKestrel() instead.
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:8080
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
```

5. Alpine Linux containerization

5. Alpine Linux containerization

- musl, busybox 기반의 비상업적 범용 Linux 배포판
- 패키지 관리자 apk 통해서 다양한 패키지 선택적 설치 가능
- 경량이고 리소스 효율성 높음

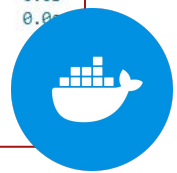
5. Alpine Linux containerization

```
1  # 베이스 이미지 설정
2  # FROM mcr.microsoft.com/dotnet/nightly/runtime:7.0-alpine AS base
3  FROM mcr.microsoft.com/dotnet/aspnet:7.0-alpine AS base
4  RUN apk add --no-cache icu-libs
5
6  # icu-libs를 넣으니까 SqlConnection.Open()에서 에러가 나지 않음.
7
8  WORKDIR /app
9  # EXPOSE 80
10 EXPOSE 8080
11 # EXPOSE 443
12
13 # 빌드 이미지 설정
14 FROM mcr.microsoft.com/dotnet/sdk:7.0-alpine AS build
15 # RUN apk add --no-cache dotnet-sdk-7.0
16
```

Dockerfile

5. Alpine Linux containerization

```
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
PS C:\BizData\source\WebAPI20230803\WebAPI20230803> docker build -f Dockerfile_Alpine --force-rm --no-cache -t webapi-alpine .
[+] Building 13.9s (19/19) FINISHED
 => [internal] load .dockerignore                                                                                    0.0s
 => => transferring context: 2B                                                                                      0.0s
 => [internal] load build definition from Dockerfile_Alpine                                                          0.0s
 => => transferring dockerfile: 1.21kB                                                                               0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0-alpine                                             0.1s
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0-alpine                                          0.1s
 => [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0-alpine@sha256:e3b051cbad561cec1b1ce3586aaf1279aeda72c2416f41c909d293cddf21c011  0.0s
 => [internal] load build context                                                                                    0.0s
 => => transferring context: 37.72kB                                                                                 0.0s
 => CACHED [base 1/3] FROM mcr.microsoft.com/dotnet/aspnet:7.0-alpine@sha256:f3d99f54d504a21d38e4cc2f13ff47d67235efeeb85c109d3d1ff1808  0.0s
 => CACHED [build 2/7] WORKDIR /src                                                                                  0.0s
 => [base 2/3] RUN apk add --no-cache icu-libs                                                                       1.2s
 => [build 3/7] COPY [WebAPI20230803.csproj, .]                                                                      0.0s
 => [build 4/7] RUN dotnet restore "./WebAPI20230803.csproj"                                                         6.3s
 => [base 3/3] WORKDIR /app                                                                                          0.0s
 => [final 1/2] WORKDIR /app                                                                                         0.0s
 => [build 5/7] COPY . .                                                                                             0.5s
 => [build 6/7] WORKDIR /src/.                                                                                       0.0s
 => [build 7/7] RUN dotnet build "WebAPI20230803.csproj" -c Release -o /app/build                                    4.5s
 => [publish 1/1] RUN dotnet publish "WebAPI20230803.csproj" -c Release -o /app/publish                              2.2s
 => [final 2/2] COPY --from=publish /app/publish .                                                                   0.0s
 => exporting to image                                                                                               0.1s
 => => exporting layers                                                                                              0.1s
 => => writing image sha256:d19b369f29a630e10f82d62e765c611d5e89e1aad74b697538b321f804ee5b02                         0.0s
 => => naming to docker.io/library/webapi-alpine                                                                     0.0s

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
```



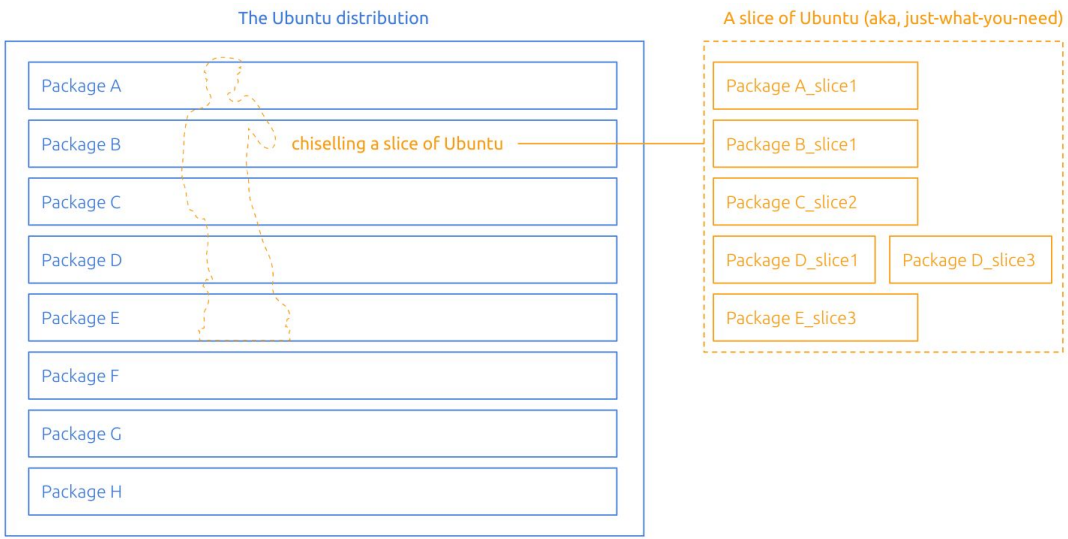
5. Alpine Linux containerization

```
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
PS C:\BizData\source\WebAPI20230803\WebAPI20230803> docker run -p 8080:8080 webapi-alpine
warn: Microsoft.AspNetCore.Server.Kestrel[0]
      Overriding address(es) 'http://0.0.0.0:8080'. Binding to endpoints defined via IConfiguration and/or UseKestrel() instead.
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:8080
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
```

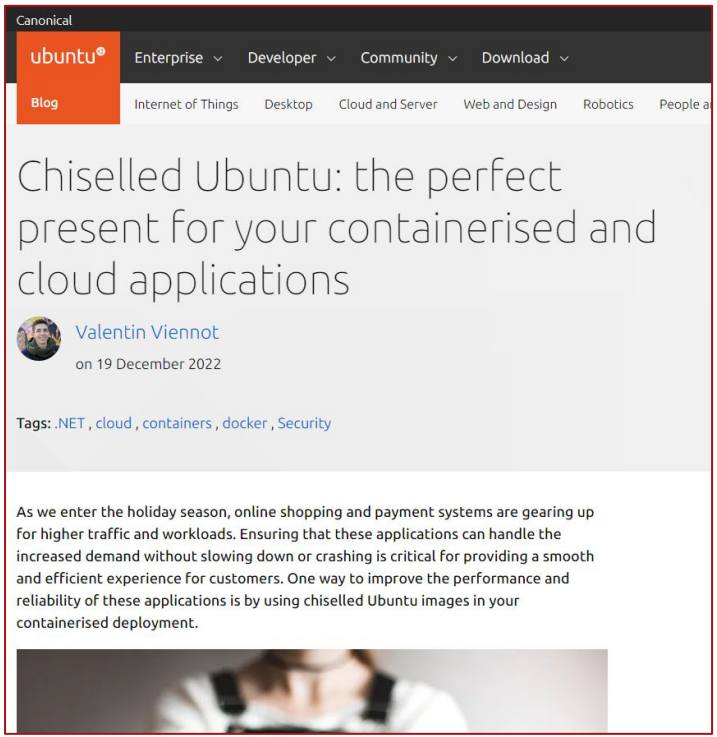
6. Chiseled Ubuntu containerization

6. Chiseled Ubuntu containerization

- Debian package의 하위 집합 (package slice)
- Canonical에서 준 2022년 Christmas gift(?)



6. Chiseled Ubuntu containerization



Canonical

ubuntu® Enterprise Developer Community Download

Blog Internet of Things Desktop Cloud and Server Web and Design Robotics People a

Chiselled Ubuntu: the perfect present for your containerised and cloud applications

Valentin Viennot

on 19 December 2022

Tags: .NET , cloud , containers , docker , Security

As we enter the holiday season, online shopping and payment systems are gearing up for higher traffic and workloads. Ensuring that these applications can handle the increased demand without slowing down or crashing is critical for providing a smooth and efficient experience for customers. One way to improve the performance and reliability of these applications is by using chiselled Ubuntu images in your containerised deployment.

Microsoft and Canonical announce native .NET availability in Ubuntu 22.04 hosts and containers

Canonical

on 16 August 2022

Tags: .NET , containers , Microsoft , Open source , Security

ⓘ This article is more than 1 year old.

.NET

Ubuntu

6. Chiseled Ubuntu containerization

6. Chiseled Ubuntu containerization

```
1  # golang:1.20 이미지를 chisel 이름으로 새로운 단계 시작
2  FROM golang:1.20 as chisel
3
4  LABEL maintainer="rokag3@gmail.com"
5  LABEL version="1.0.0"
6  LABEL description="dockerfile test"
7
8  # GitHub 저장소에서 Chisel repo 클론하고 /opt/chisel 으로 이동,
9  RUN git clone --depth 1 -b main https://github.com/canonical/chisel /opt/chisel
10 WORKDIR /opt/chisel
11 RUN go build ./cmd/chisel
12
13 # https://hub.docker.com/_/microsoft-dotnet
14 FROM mcr.microsoft.com/dotnet/sdk:7.0-jammy AS build
15
16 # 앞서 빌드한 Chisel 실행 파일을 /usr/bin/ 으로 복사
17 COPY --from=chisel /opt/chisel/chisel /usr/bin/
18
19 # /rootfs 디렉토리 생성하고 chisel cut 으로 Ubuntu 22.04 릴리즈에 필요한 라이브러리를 추출해서 /
   rootfs 디렉토리에 저장
20 RUN mkdir /rootfs
21 RUN chisel cut --release "ubuntu-22.04" --root /rootfs/ libicu70_libs
22 # libssl3_libs libc6_libs dotnet-runtime-6.0_libs dotnet-hostfxr-6.0_libs libgcc-s1_libs
   liblttng-ust1_libs libstdc++6_libs libunwind-13_libs libunwind8_libs zlib1g_libs
23 # && apt-get update && apt-get install -y bash
```

Dockerfile

6. Chiseled Ubuntu containerization

```
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
PS C:\BizData\source\WebAPI20230803\WebAPI20230803> docker build -f Dockerfile_Chiseled --force-rm --no-cache -t webapi-chiseled .
[+] Building 107.9s (24/24) FINISHED                                                                  docker:default
 => [internal] load .dockerignore                                                                               0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [internal] load build definition from Dockerfile_Chiseled                                                   0.1s
 => => transferring dockerfile: 1.92kB                                                                          0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0-jammy                                         0.2s
 => [internal] load metadata for docker.io/library/golang:1.20                                                 10.4s
 => [internal] load metadata for mcr.microsoft.com/dotnet/nightly/aspnet:7.0-jammy-chiseled                     1.1s
 => [auth] library/golang:pull token for registry-1.docker.io                                                   0.0s
 => [chisel 1/4] FROM docker.io/library/golang:1.20@sha256:2e558eddb06645eb1909979c225ce4475b072b6f5b5390ddeba3da75afd6d9bd  26.2s
 => => resolve docker.io/library/golang:1.20@sha256:2e558eddb06645eb1909979c225ce4475b072b6f5b5390ddeba3da75afd6d9bd  0.1s
 => => sha256:efe38cb419e2b2012f66d1782d2efe2fd8884c71d9f342581e1697ba9047b5f8 1.58kB / 1.58kB                  0.0s
 => => sha256:00046d1e755ea94fa55a700ca9a10597e4fac7c47be19d970a359b0267a51fbf 24.03MB / 24.03MB                3.0s
 => => sha256:2e558eddb06645eb1909979c225ce4475b072b6f5b5390ddeba3da75afd6d9bd 2.36kB / 2.36kB                  0.0s
 => => sha256:f0b37d626a9957eea46dc65ffb5b063d8fc86ee17ea859c76dc7e34374b2cc84 6.86kB / 6.86kB                  0.0s
 => => sha256:012c0b3e998c1a0c0bedcf712eaaafb188580529dd026a04aa1ce13fdb39e42b 49.56MB / 49.56MB                5.4s
 => => sha256:9f13f5a53d118643c1f1ff294867c09f224d00edca21f56caa71c2321f8ca004 64.11MB / 64.11MB                7.0s
```

syncopation..

```
 => => transferring context: 37.72kB                                                                            0.1s
 => [chisel 2/4] RUN git clone --depth 1 -b main https://github.com/canonical/chisel /opt/chisel                2.2s
 => [chisel 3/4] WORKDIR /opt/chisel                                                                            0.0s
 => [chisel 4/4] RUN go build ./cmd/chisel                                                                     13.4s
 => [build 2/8] COPY --from=chisel /opt/chisel/chisel /usr/bin/                                                 0.1s
 => [build 3/8] RUN mkdir /rootfs                                                                               0.4s
 => [build 4/8] RUN chisel cut --release "ubuntu-22.04" --root /rootfs/ libicu70_libs                          41.4s
 => [build 5/8] WORKDIR /source                                                                                 0.0s
 => [build 6/8] COPY . .                                                                                        0.5s
 => [build 7/8] RUN dotnet restore "WebAPI20230803.csproj"                                                      6.4s
 => [build 8/8] RUN dotnet publish -c Release -o /app --os linux --arch x64                                     5.6s
 => [stage-2 2/4] COPY --from=build /rootfs /                                                                   0.1s
 => [stage-2 3/4] WORKDIR /app                                                                                  0.0s
 => [stage-2 4/4] COPY --from=build /app .                                                                      0.0s
 => exporting to image
 => => exporting layers
 => => writing image sha256:db796ef2212ca893c3047bb7c036211359625b5ec27a28cd1c3537683afc2374
 => => naming to docker.io/library/webapi-chiseled

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
```

6. Chiseled Ubuntu containerization

```
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
PS C:\BizData\source\WebAPI20230803\WebAPI20230803> docker run -p 8080:8080 webapi-chiseled
warn: Microsoft.AspNetCore.Server.Kestrel[0]
      Overriding address(es) 'http://0.0.0.0:8080'. Binding to endpoints defined via IConfiguration and/or UseKestrel() instead.
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:8080
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
```

7. Distroless containerization

7. Distroless containerization

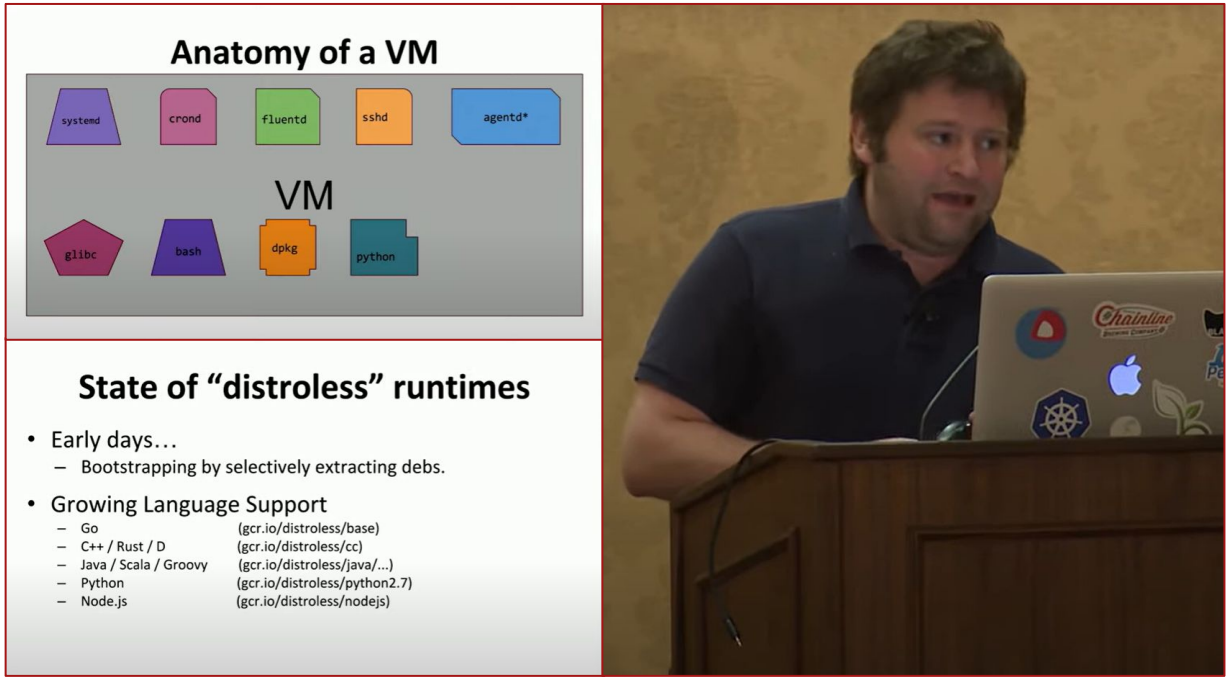
"Distroless" Container Images.

CI passing

"Distroless" images contain only your application and its runtime dependencies. They do not contain package managers, shells or any other programs you would expect to find in a standard Linux distribution.

- Google에서 개발 시작.
- 가능한 최소한의 컨테이너 이미지를 생성하자.
- 패키지 관리자, 쉘 등 이 없음.
- OSS Bazel로 빌드.

7. Distroless containerization



2017 swampUP Sessions | Distroless Docker: Containerizing Apps, not VMs - Matthew Moore

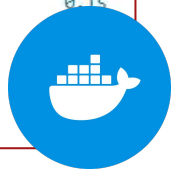
7. Distroless containerization

```
FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
# FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS build

WORKDIR /app

COPY . .

EXPOSE 8080

RUN dotnet build WebAPI20230803.csproj -c Release
RUN dotnet publish WebAPI20230803.csproj -c Release -o /app/publish /p:self-contained=true
# --os linux --arch x64

## Distroless 베이스 이미지를 사용하여 최종 이미지 빌드
# FROM gcr.io/distroless/dotnet-7.0-sdk
# FROM gcr.io/distroless/dotnet-7.0-runtime
# FROM asia.gcr.io/distroless/dotnet-7.0
FROM gcr.io/distroless/cc
# FROM gcr.io/distroless/base
# FROM gcr.io/distroless/base-nossl

COPY --from=build /app/publish /app

# ENTRYPOINT ["dotnet", "run", "--project", "WebAPI20230803.csproj"]
# ENTRYPOINT ["dotnet", "WebAPI20230803.dll"]
ENTRYPOINT ["/app/WebAPI20230803"]
# CMD ["/app/WebAPI20230803"]
```

Dockerfile

7. Distroless containerization

```
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
PS C:\BizData\source\WebAPI20230803\WebAPI20230803> docker build -f Dockerfile_Distroless --force-rm --no-cache -t webapi-distroless .
[+] Building 12.5s (13/13) FINISHED
 => [internal] load .dockerignore                                                                                    0.0s
 => => transferring context: 2B                                                                                      0.0s
 => [internal] load build definition from Dockerfile_Distroless                                                      0.0s
 => => transferring dockerfile: 4.20kB                                                                               0.0s
 => [internal] load metadata for gcr.io/distroless/cc:latest                                                         0.8s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0                                                    0.2s
 => [build 1/5] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:bdcfb498261ca18f023ac67615d814ea743aa3288eb880855fa2eb86c6313ccc  0.0s
 => [internal] load build context                                                                                    0.1s
 => => transferring context: 37.72kB                                                                                 0.1s
 => CACHED [stage-1 1/2] FROM gcr.io/distroless/cc@sha256:3603adbdee2906dc3b7a18d7c0424a40633231c61dcd82196ae15de1282a5822  0.0s
 => CACHED [build 2/5] WORKDIR /app                                                                                  0.0s
 => [build 3/5] COPY . .                                                                                             0.8s
 => [build 4/5] RUN dotnet build WebAPI20230803.csproj -c Release                                                    8.3s
 => [build 5/5] RUN dotnet publish WebAPI20230803.csproj -c Release -o /app/publish /p:self-contained=true          2.0s
 => [stage-1 2/2] COPY --from=build /app/publish /app                                                                0.1s
 => exporting to image                                                                                               0.1s
 => => exporting layers                                                                                              0.1s
 => => writing image sha256:98cee4b4613764ffc23f8c1d8c796f3ec5a26790097f9f9722a935519ec6cf11                         0.0s
 => => naming to docker.io/library/webapi-distroless                                                                 0.1s

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\BizData\source\WebAPI20230803\WebAPI20230803>
```



Q&A

Thank you

References

Chiselled Ubuntu: the perfect present for your containerised and cloud applications (2022.12.19 Valentin Viennot) - https://canonical.com/blog/chiselled-containers-perfect-gift-cloud-applications

The benefits of Chiselled Ubuntu images in action with an ASP.NET shop demo (2022.12.20 Valentin Viennot) - https://ubuntu.com/blog/benefits-chiselled-ubuntu-images-aspnet-eshop-demo

Canonical의 Chisel GitHub repo - https://github.com/canonical/chisel

Canonical의 Chisel slice releases - https://github.com/canonical/chisel-releases/tree/ubuntu-22.04/slices

Microsoft and Canonical announce native .NET availability in Ubuntu 22.04 hosts and containers - https://ubuntu.com/blog/install-dotnet-on-ubuntu

Using .NET with Cheseled Ubuntu Containers | .NET Conf 2022 - https://youtu.be/FLGFzlWF4Gs

.NET 6 is now in Ubuntu 22.04 - https://devblogs.microsoft.com/dotnet/dotnet-6-is-now-in-ubuntu-2204/

.NET 7 SDK built-in container support and Ubuntu Chiseled - https://laurentkempe.com/2022/11/14/dotnet-7-sdk-built-in-container-support-and-ubuntu-chiseled/

Which Container Images To Use Distroless Or Alpine? - https://itnext.io/which-container-images-to-use-distroless-or-alpine-96e3dab43a22

Alpine Linux에서 SqlClient 배포 이슈 - https://github.com/dotnet/SqlClient/issues/81#issuecomment-399375323

References

Secure your .NET cloud apps with rootless Linux Containers - https://devblogs.microsoft.com/dotnet/securing-containers-with-rootless/

GoogleContainerTools GitHub repo > Distroless - https://github.com/GoogleContainerTools/distroless

.NET 7 is Available Today (2022.11.08) - https://devblogs.microsoft.com/dotnet/announcing-dotnet-7/

Docker hub - ASP.NET Core Runtime 6.0 / 7.0 - https://hub.docker.com/_/microsoft-dotnet-aspnet

Alpine Linux - https://www.alpinelinux.org/

Distroless Docker: Containerizing Apps, not VMs - Matthew Moore (2017 swampUP Sessions) - https://youtu.be/lviLZFciDv4

Bazel - https://bazel.build/?hl=ko