Cloud and Machine Learning

CSCI-GA.3033-085 Spring 2024

Prof. Hao Yu

Prof. I-Hsin Chung

Lecture 7-2: Containers, Docker, Kubernetes

Agenda

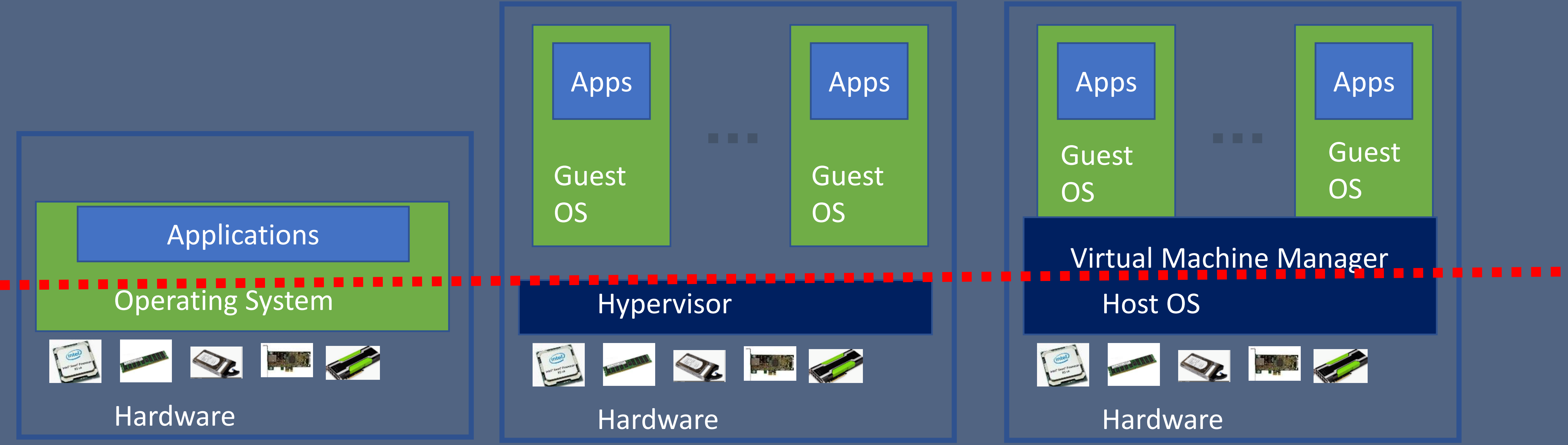
- Lecture
  - **AI workflows**
  - Introduction to Containers
  - Docker and ecosystem
  - Container orchestration
  - Kubernetes
- Lab
  - Use Docker on your virtual machine
  - Use Container to run a web application on your laptop

What is propelling current generation AI

- Data
- Compute infrastructure
- Algorithms

Equally important how these capabilities are packaged and distributed

Different VM architectures



Baremetal or Non-virtualized system

Native VM using Hypervisor

Hosted VM using VMM

AI Workflow: critical steps in ML

Data preparation

Workflow of steps (e.g., remove hate and profanity, deduplicate)

Hours to days

10-2000+ low to mid-end CPU cores

10+ low to mid-end GPUs per

10-100+ concurrent jobs

on-prem **Public clouds**

Distributed training

Long-running job on massive infrastructure

weeks to months

10-500+ high-end GPUs (per job)

10+ concurrent jobs

on-prem **Public clouds**

Model adaptation

Model tuning with custom data set for downstream tasks

minutes to hours

1+ mid to high-end GPU (per job)

100+ concurrent jobs

on-prem **Public clouds**

Inference

May have sensitivity to latency, throughput, power

sub-second API request

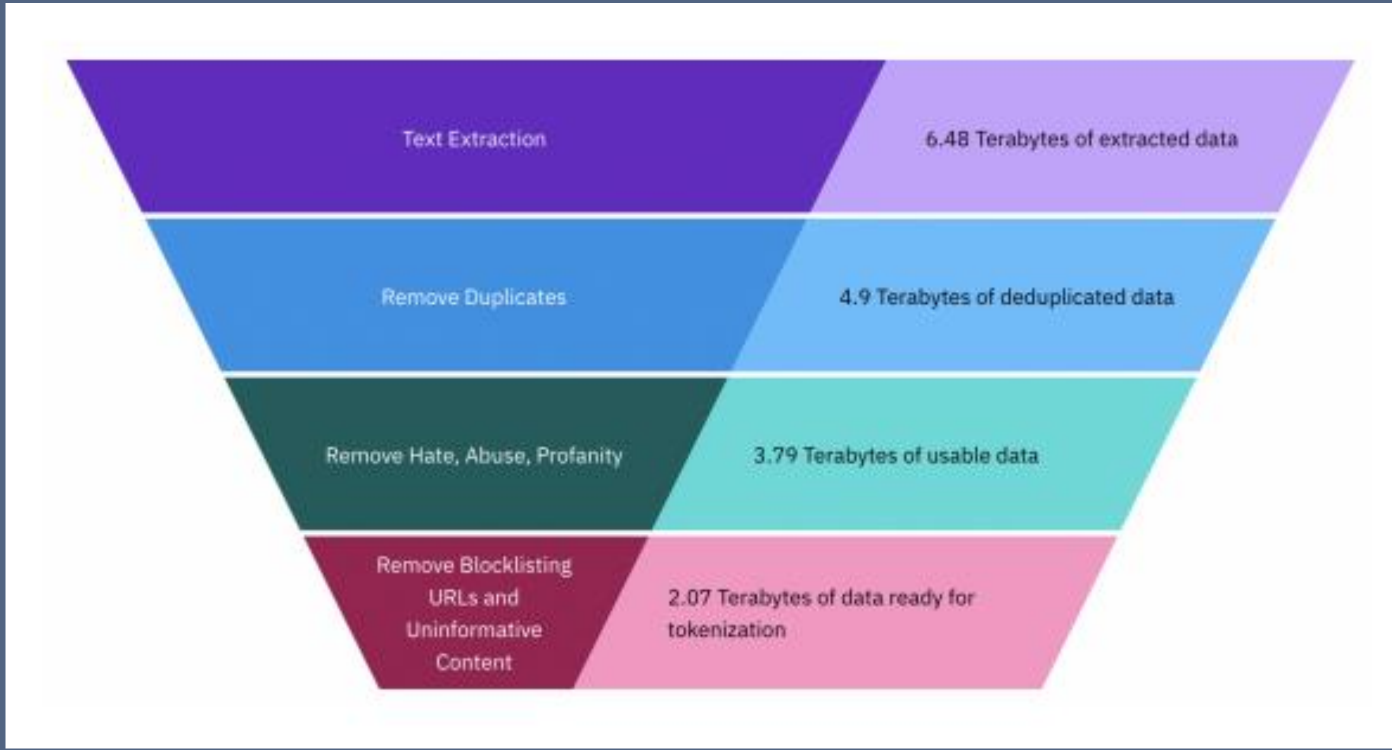
Single low-end GPU per fine tuning task

Fraction to multiple GPUs per inference, or specialized accelerator

Thousands of API requests

on-prem **Public clouds** **Edge**

Example data processing



IBM Research data governance summary stats

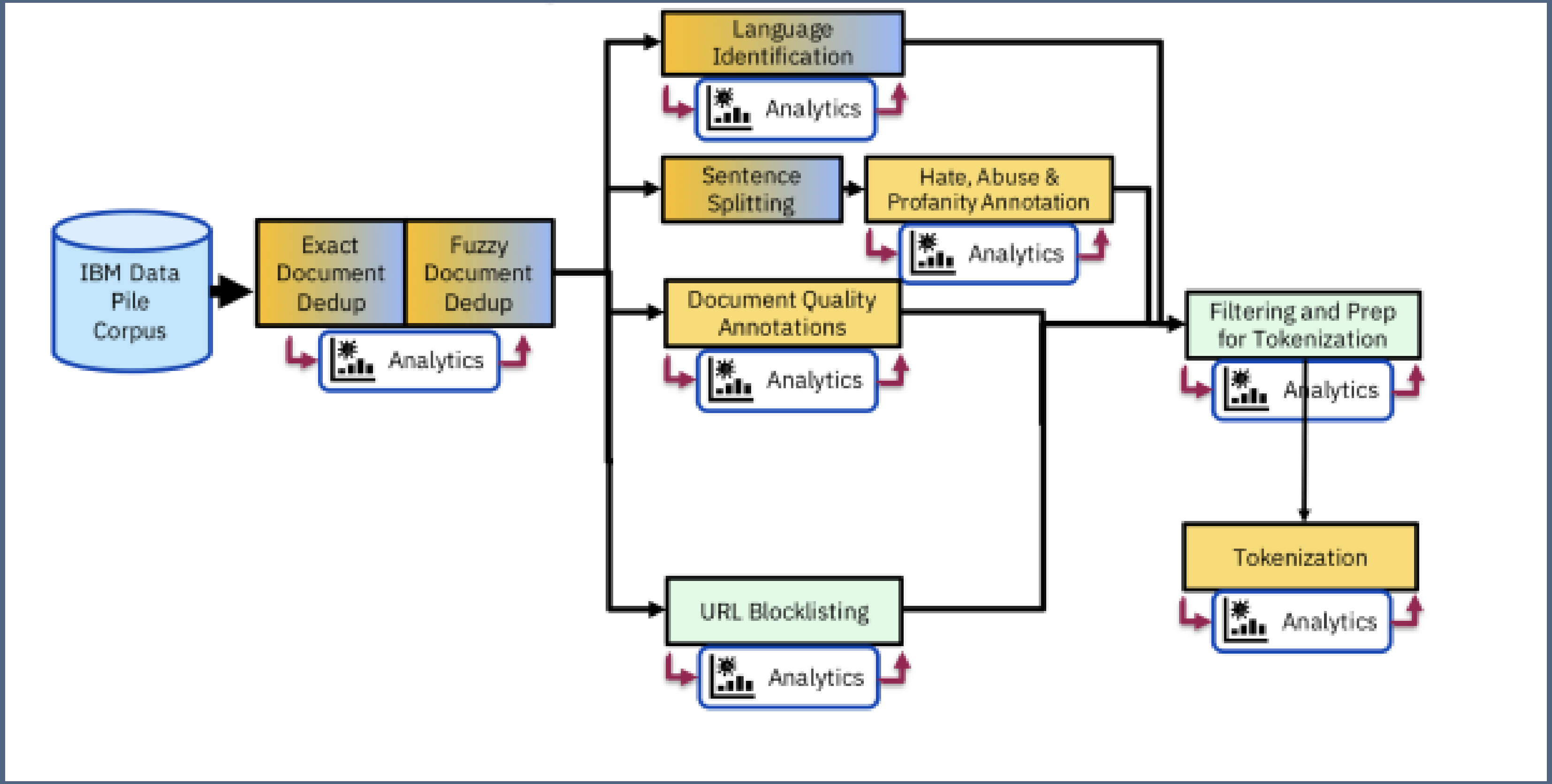
TABLE 2: Statistics of commonly-used data sources.

Corpora	Size	Source	Latest Update Time
BookCorpus [153]	5GB	Books	Dec-2015
Gutenberg [154]	-	Books	Dec-2021
C4 [82]	800GB	CommonCrawl	Apr-2019
CC-Stories-R [155]	31GB	CommonCrawl	Sep-2019
CC-NEWS [27]	78GB	CommonCrawl	Feb-2019
REALNEWs [156]	120GB	CommonCrawl	Apr-2019
OpenWebText [157]	38GB	Reddit links	Mar-2023
Pushift.io [158]	2TB	Reddit links	Mar-2023
Wikipedia [159]	21GB	Wikipedia	Mar-2023
BigQuery [160]	-	Codes	Mar-2023
the Pile [161]	800GB	Other	Dec-2020
ROOTS [162]	1.6TB	Other	Jun-2022

https://arxiv.org/pdf/2303.18223.pdf

https://medium.com/@manavg/thoughts-on-ibm-granite-models-80cda8e37a7b

Typical data pre-processing pipeline



https://medium.com/@manavg/thoughts-on-ibm-granite-models-80cda8e37a7b

Typical data pre-processing pipeline

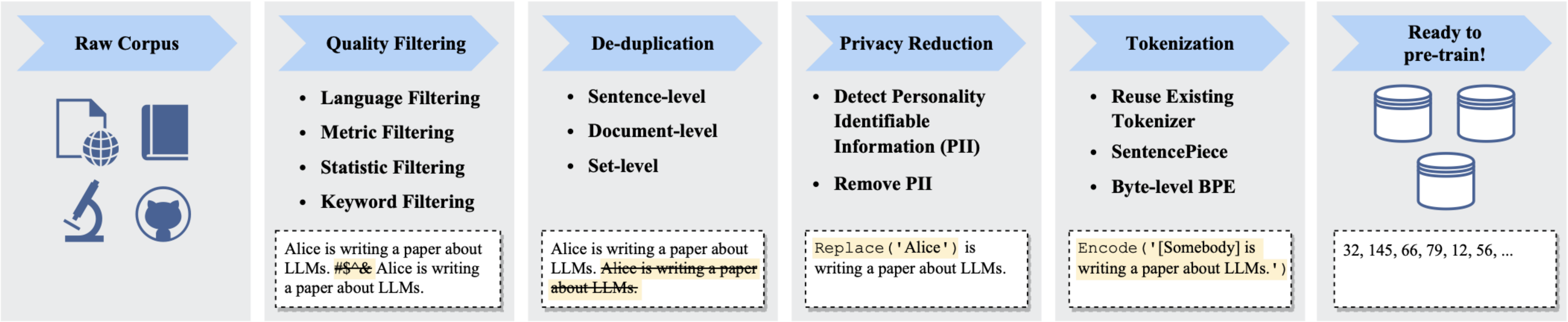


Fig. 7: An illustration of a typical data preprocessing pipeline for pre-training large language models.

https://arxiv.org/pdf/2303.18223.pdf

Statistics of public models

	Model	Release Time	Size (B)	Base Model	Adaptation IT	Adaptation RLHF	Pre-train Data Scale	Latest Data Timestamp	Hardware (GPUs / TPUs)	Training Time	Evaluation ICL	Evaluation CoT
	T5 [82]	Oct-2019	11	-	-	-	1T tokens	Apr-2019	1024 TPU v3	-	✓	-
	mT5 [83]	Oct-2020	13	-	-	-	1T tokens	-	-	-	✓	-
	PanGu-α [84]	Apr-2021	13*	-	-	-	1.1TB	-	2048 Ascend 910	-	✓	-
	CPM-2 [85]	Jun-2021	198	-	-	-	2.6TB	-	-	-	-	-
	T0 [28]	Oct-2021	11	T5	✓	-	-	-	512 TPU v3	27 h	✓	-
	CodeGen [86]	Mar-2022	16	-	-	-	577B tokens	-	-	-	✓	-
	GPT-NeoX-20B [87]	Apr-2022	20	-	-	-	825GB	-	96 40G A100	-	✓	-
	Tk-Instruct [88]	Apr-2022	11	T5	✓	-	-	-	256 TPU v3	4 h	✓	-
	UL2 [89]	May-2022	20	-	-	-	1T tokens	Apr-2019	512 TPU v4	-	✓	✓
	OPT [90]	May-2022	175	-	-	-	180B tokens	-	992 80G A100	-	✓	-
	NLLB [91]	Jul-2022	54.5	-	-	-	-	-	-	-	✓	-
	CodeGeeX [92]	Sep-2022	13	-	-	-	850B tokens	-	1536 Ascend 910	60 d	✓	-
	GLM [93]	Oct-2022	130	-	-	-	400B tokens	-	768 40G A100	60 d	✓	-
	Flan-T5 [69]	Oct-2022	11	T5	✓	-	-	-	-	-	✓	✓
	BLOOM [78]	Nov-2022	176	-	-	-	366B tokens	-	384 80G A100	105 d	✓	-
	mT0 [94]	Nov-2022	13	mT5	✓	-	-	-	-	-	✓	-
	Galactica [35]	Nov-2022	120	-	-	-	106B tokens	-	-	-	✓	✓
	BLOOMZ [94]	Nov-2022	176	BLOOM	✓	-	-	-	-	-	✓	-
Publicly	OPT-IML [95]	Dec-2022	175	OPT	✓	-	-	-	128 40G A100	-	✓	✓
Available	LLaMA [57]	Feb-2023	65	-	-	-	1.4T tokens	-	2048 80G A100	21 d	✓	-
	Pythia [96]	Apr-2023	12	-	-	-	300B tokens	-	256 40G A100	-	✓	-
	CodeGen2 [97]	May-2023	16	-	-	-	400B tokens	-	-	-	✓	-
	StarCoder [98]	May-2023	15.5	-	-	-	1T tokens	-	512 40G A100	-	✓	✓
	LLaMA2 [99]	Jul-2023	70	-	✓	✓	2T tokens	-	2000 80G A100	-	✓	-
	Baichuan2 [100]	Sep-2023	13	-	✓	✓	2.6T tokens	-	1024 A800	-	✓	-
	QWEN [101]	Sep-2023	14	-	✓	✓	3T tokens	-	-	-	✓	-
	FLM [102]	Sep-2023	101	-	✓	-	311B tokens	-	192 A800	22 d	✓	-
	Skywork [103]	Oct-2023	13	-	-	-	3.2T tokens	-	512 80G A800	-	✓	-

https://arxiv.org/pdf/2303.18223.pdf

Statistics of closed models

	Model	Release Time	Size (B)	Base Model	Adaptation IT	Adaptation RLHF	Pre-train Data Scale	Latest Data Timestamp	Hardware (GPUs / TPUs)	Training Time	Evaluation ICL	Evaluation CoT
Closed Source	GPT-3 [55]	May-2020	175	-	-	-	300B tokens	-	-	-	✓	-
	GShard [104]	Jun-2020	600	-	-	-	1T tokens	-	2048 TPU v3	4 d	-	-
	Codex [105]	Jul-2021	12	GPT-3	-	-	100B tokens	May-2020	-	-	✓	-
	ERNIE 3.0 [106]	Jul-2021	10	-	-	-	375B tokens	-	384 V100	-	✓	-
	Jurassic-1 [107]	Aug-2021	178	-	-	-	300B tokens	-	800 GPU	-	✓	-
	HyperCLOVA [108]	Sep-2021	82	-	-	-	300B tokens	-	1024 A100	13.4 d	✓	-
	FLAN [67]	Sep-2021	137	LaMDA-PT	✓	-	-	-	128 TPU v3	60 h	✓	-
	Yuan 1.0 [109]	Oct-2021	245	-	-	-	180B tokens	-	2128 GPU	-	✓	-
	Anthropic [110]	Dec-2021	52	-	-	-	400B tokens	-	-	-	✓	-
	WebGPT [81]	Dec-2021	175	GPT-3	-	✓	-	-	-	-	✓	-
	Gopher [64]	Dec-2021	280	-	-	-	300B tokens	-	4096 TPU v3	920 h	✓	-
	ERNIE 3.0 Titan [111]	Dec-2021	260	-	-	-	-	-	-	-	✓	-
	GLaM [112]	Dec-2021	1200	-	-	-	280B tokens	-	1024 TPU v4	574 h	✓	-
	LaMDA [68]	Jan-2022	137	-	-	-	768B tokens	-	1024 TPU v3	57.7 d	-	-
	MT-NLG [113]	Jan-2022	530	-	-	-	270B tokens	-	4480 80G A100	-	✓	-
	AlphaCode [114]	Feb-2022	41	-	-	-	967B tokens	Jul-2021	-	-	-	-
	InstructGPT [66]	Mar-2022	175	GPT-3	✓	✓	-	-	-	-	✓	-
	Chinchilla [34]	Mar-2022	70	-	-	-	1.4T tokens	-	-	-	✓	-
	PaLM [56]	Apr-2022	540	-	-	-	780B tokens	-	6144 TPU v4	-	✓	✓
	AlexaTM [115]	Aug-2022	20	-	-	-	1.3T tokens	-	128 A100	120 d	✓	✓
	Sparrow [116]	Sep-2022	70	-	-	✓	-	-	64 TPU v3	-	✓	-
	WeLM [117]	Sep-2022	10	-	-	-	300B tokens	-	128 A100 40G	24 d	✓	-
	U-PaLM [118]	Oct-2022	540	PaLM	-	-	-	-	512 TPU v4	5 d	✓	✓
	Flan-PaLM [69]	Oct-2022	540	PaLM	✓	-	-	-	512 TPU v4	37 h	✓	✓
	Flan-U-PaLM [69]	Oct-2022	540	U-PaLM	✓	-	-	-	-	-	✓	✓
	GPT-4 [46]	Mar-2023	-	-	✓	✓	-	-	-	-	✓	✓
	PanGu-Σ [119]	Mar-2023	1085	PanGu-α	-	-	329B tokens	-	512 Ascend 910	100 d	✓	-
	PaLM2 [120]	May-2023	16	-	✓	-	100B tokens	-	-	-	✓	✓

https://arxiv.org/pdf/2303.18223.pdf

AI Workflow: critical steps in ML

Data preparation

Workflow of steps (e.g., remove hate and profanity, deduplicate)

Hours to days

10-2000+ low to mid-end CPU cores

10+ low to mid-end GPUs per

10-100+ concurrent jobs

on-prem **Public clouds**

Distributed training

Long-running job on massive infrastructure

weeks to months

10-500+ high-end GPUs (per job)

10+ concurrent jobs

on-prem **Public clouds**

Model adaptation

Model tuning with custom data set for downstream tasks

minutes to hours

1+ mid to high-end GPU (per job)

100+ concurrent jobs

on-prem **Public clouds**

Inference

May have sensitivity to latency, throughput, power

sub-second API request

Single low-end GPU per fine tuning task

Fraction to multiple GPUs per inference, or specialized accelerator

Thousands of API requests

on-prem **Public clouds** **Edge**

Statistics of public models

	Model	Release Time	Size (B)	Base Model	Adaptation IT	Adaptation RLHF	Pre-train Data Scale	Latest Data Timestamp	Hardware (GPUs / TPUs)	Training Time	Evaluation ICL	Evaluation CoT
	T5 [82]	Oct-2019	11	-	-	-	1T tokens	Apr-2019	1024 TPU v3	-	✓	-
	mT5 [83]	Oct-2020	13	-	-	-	1T tokens	-	-	-	✓	-
	PanGu-α [84]	Apr-2021	13*	-	-	-	1.1TB	-	2048 Ascend 910	-	✓	-
	CPM-2 [85]	Jun-2021	198	-	-	-	2.6TB	-	-	-	-	-
	T0 [28]	Oct-2021	11	T5	✓	-	-	-	512 TPU v3	27 h	✓	-
	CodeGen [86]	Mar-2022	16	-	-	-	577B tokens	-	-	-	✓	-
	GPT-NeoX-20B [87]	Apr-2022	20	-	-	-	825GB	-	96 40G A100	-	✓	-
	Tk-Instruct [88]	Apr-2022	11	T5	✓	-	-	-	256 TPU v3	4 h	✓	-
	UL2 [89]	May-2022	20	-	-	-	1T tokens	Apr-2019	512 TPU v4	-	✓	✓
	OPT [90]	May-2022	175	-	-	-	180B tokens	-	992 80G A100	-	✓	-
	NLLB [91]	Jul-2022	54.5	-	-	-	-	-	-	-	✓	-
	CodeGeeX [92]	Sep-2022	13	-	-	-	850B tokens	-	1536 Ascend 910	60 d	✓	-
	GLM [93]	Oct-2022	130	-	-	-	400B tokens	-	768 40G A100	60 d	✓	-
	Flan-T5 [69]	Oct-2022	11	T5	✓	-	-	-	-	-	✓	✓
	BLOOM [78]	Nov-2022	176	-	-	-	366B tokens	-	384 80G A100	105 d	✓	-
	mT0 [94]	Nov-2022	13	mT5	✓	-	-	-	-	-	✓	-
	Galactica [35]	Nov-2022	120	-	-	-	106B tokens	-	-	-	✓	✓
	BLOOMZ [94]	Nov-2022	176	BLOOM	✓	-	-	-	-	-	✓	-
Publicly	OPT-IML [95]	Dec-2022	175	OPT	✓	-	-	-	128 40G A100	-	✓	✓
Available	LLaMA [57]	Feb-2023	65	-	-	-	1.4T tokens	-	2048 80G A100	21 d	✓	-
	Pythia [96]	Apr-2023	12	-	-	-	300B tokens	-	256 40G A100	-	✓	-
	CodeGen2 [97]	May-2023	16	-	-	-	400B tokens	-	-	-	✓	-
	StarCoder [98]	May-2023	15.5	-	-	-	1T tokens	-	512 40G A100	-	✓	✓
	LLaMA2 [99]	Jul-2023	70	-	✓	✓	2T tokens	-	2000 80G A100	-	✓	-
	Baichuan2 [100]	Sep-2023	13	-	✓	✓	2.6T tokens	-	1024 A800	-	✓	-
	QWEN [101]	Sep-2023	14	-	✓	✓	3T tokens	-	-	-	✓	-
	FLM [102]	Sep-2023	101	-	✓	-	311B tokens	-	192 A800	22 d	✓	-
	Skywork [103]	Oct-2023	13	-	-	-	3.2T tokens	-	512 80G A800	-	✓	-

https://arxiv.org/pdf/2303.18223.pdf

Statistics of closed models

	Model	Release Time	Size (B)	Base Model	Adaptation IT	Adaptation RLHF	Pre-train Data Scale	Latest Data Timestamp	Hardware (GPUs / TPUs)	Training Time	Evaluation ICL	Evaluation CoT
Closed Source	GPT-3 [55]	May-2020	175	-	-	-	300B tokens	-	-	-	✓	-
	GShard [104]	Jun-2020	600	-	-	-	1T tokens	-	2048 TPU v3	4 d	-	-
	Codex [105]	Jul-2021	12	GPT-3	-	-	100B tokens	May-2020	-	-	✓	-
	ERNIE 3.0 [106]	Jul-2021	10	-	-	-	375B tokens	-	384 V100	-	✓	-
	Jurassic-1 [107]	Aug-2021	178	-	-	-	300B tokens	-	800 GPU	-	✓	-
	HyperCLOVA [108]	Sep-2021	82	-	-	-	300B tokens	-	1024 A100	13.4 d	✓	-
	FLAN [67]	Sep-2021	137	LaMDA-PT	✓	-	-	-	128 TPU v3	60 h	✓	-
	Yuan 1.0 [109]	Oct-2021	245	-	-	-	180B tokens	-	2128 GPU	-	✓	-
	Anthropic [110]	Dec-2021	52	-	-	-	400B tokens	-	-	-	✓	-
	WebGPT [81]	Dec-2021	175	GPT-3	-	✓	-	-	-	-	✓	-
	Gopher [64]	Dec-2021	280	-	-	-	300B tokens	-	4096 TPU v3	920 h	✓	-
	ERNIE 3.0 Titan [111]	Dec-2021	260	-	-	-	-	-	-	-	✓	-
	GLaM [112]	Dec-2021	1200	-	-	-	280B tokens	-	1024 TPU v4	574 h	✓	-
	LaMDA [68]	Jan-2022	137	-	-	-	768B tokens	-	1024 TPU v3	57.7 d	-	-
	MT-NLG [113]	Jan-2022	530	-	-	-	270B tokens	-	4480 80G A100	-	✓	-
	AlphaCode [114]	Feb-2022	41	-	-	-	967B tokens	Jul-2021	-	-	-	-
	InstructGPT [66]	Mar-2022	175	GPT-3	✓	✓	-	-	-	-	✓	-
	Chinchilla [34]	Mar-2022	70	-	-	-	1.4T tokens	-	-	-	✓	-
	PaLM [56]	Apr-2022	540	-	-	-	780B tokens	-	6144 TPU v4	-	✓	✓
	AlexaTM [115]	Aug-2022	20	-	-	-	1.3T tokens	-	128 A100	120 d	✓	✓
	Sparrow [116]	Sep-2022	70	-	-	✓	-	-	64 TPU v3	-	✓	-
	WeLM [117]	Sep-2022	10	-	-	-	300B tokens	-	128 A100 40G	24 d	✓	-
	U-PaLM [118]	Oct-2022	540	PaLM	-	-	-	-	512 TPU v4	5 d	✓	✓
	Flan-PaLM [69]	Oct-2022	540	PaLM	✓	-	-	-	512 TPU v4	37 h	✓	✓
	Flan-U-PaLM [69]	Oct-2022	540	U-PaLM	✓	-	-	-	-	-	✓	✓
	GPT-4 [46]	Mar-2023	-	-	✓	✓	-	-	-	-	✓	✓
	PanGu-Σ [119]	Mar-2023	1085	PanGu-α	-	-	329B tokens	-	512 Ascend 910	100 d	✓	-
	PaLM2 [120]	May-2023	16	-	✓	-	100B tokens	-	-	-	✓	✓

https://arxiv.org/pdf/2303.18223.pdf

AI Workflow: critical steps in ML

Data preparation

Workflow of steps (e.g., remove hate and profanity, deduplicate)

Hours to days

10-2000+ low to mid-end CPU cores

10+ low to mid-end GPUs per

10-100+ concurrent jobs

on-prem **Public clouds**

Distributed training

Long-running job on massive infrastructure

weeks to months

10-500+ high-end GPUs (per job)

10+ concurrent jobs

on-prem **Public clouds**

Model adaptation

Model tuning with custom data set for downstream tasks

minutes to hours

1+ mid to high-end GPU (per job)

100+ concurrent jobs

on-prem **Public clouds**

Inference

May have sensitivity to latency, throughput, power

sub-second API request

Single low-end GPU per fine tuning task

Fraction to multiple GPUs per inference, or specialized accelerator

Thousands of API requests

on-prem **Public clouds** **Edge**

Example resources for model adaptation

Models	A800 Full Training			A800 LoRA Training			A800 Inference (16-bit)		3090 Inference (16-bit)		3090 Inference (8-bit)	
	#GPU	BS	Time	#GPU	BS	Time	#GPU	#Token/s	#GPU	#Token/s	#GPU	#Token/s
LLaMA (7B)	2	8	3.0h	1	80	3.5h	1	36.6	1	24.3	1	7.5
LLaMA (13B)	4	8	3.1h	1	48	5.1h	1	26.8	2	9.9	1	4.5
LLaMA (30B)	8	4	6.1h	1	24	14.3h	1	17.7	4	3.8	2	2.6
LLaMA (65B)	16	2	11.2h	1	4	60.6h	2	8.8	8	2.0	4	1.5

See section 5-7: https://arxiv.org/pdf/2303.18223.pdf

AI Workflow: critical steps in ML

Data preparation

Workflow of steps (e.g., remove hate and profanity, deduplicate)

Hours to days

10-2000+ low to mid-end CPU cores

10+ low to mid-end GPUs per

10-100+ concurrent jobs

on-prem **Public clouds**

Distributed training

Long-running job on massive infrastructure

weeks to months

10-500+ high-end GPUs (per job)

10+ concurrent jobs

on-prem **Public clouds**

Model adaptation

Model tuning with custom data set for downstream tasks

minutes to hours

1+ mid to high-end GPU (per job)

100+ concurrent jobs

on-prem **Public clouds**

Inference

May have sensitivity to latency, throughput, power

sub-second API request

Single low-end GPU per fine tuning task

Fraction to multiple GPUs per inference, or specialized accelerator

Thousands of API requests

on-prem **Public clouds** **Edge**

AI Workflow: summary of job types

Data preparation	*Distributed training*	*Model adaptation*	*Inference*
Workflow of steps (e.g., remove hate and profanity, deduplicate)	Long-running job on massive infrastructure	Model tuning with custom data set for downstream tasks	May have sensitivity to latency, throughput, power

Bag of tasks

10-2000+ low to mid-end CPU cores

10+ low to mid-end GPUs per

10-100+ concurrent jobs

on-prem **Public clouds**

Batch jobs

10-500+ high-end GPUs (per job)

10+ concurrent jobs

on-prem **Public clouds**

Bag of tasks + batch

1+ mid to high-end GPU (per job)

100+ concurrent jobs

on-prem **Public clouds**

Web services

Single low-end GPU per fine tuning task

Fraction to multiple GPUs per inference, or specialized accelerator

Thousands of API requests

on-prem **Public clouds** **Edge**

AI Workflow: Platform requirements

Data preparation

Workflow of steps (e.g., remove hate and profanity, deduplicate)

Distributed training

Long-running job on massive infrastructure

Model adaptation

Model tuning with custom data set for downstream tasks

Inference

May have sensitivity to latency, throughput, power

Bag of tasks

High throughput

Pack for efficiency

on-prem **Public clouds**

Batch jobs

Optimize for performance

Communication intensive

Efficient I/O Performance:

Input, Checkpoint

on-prem **Public clouds**

Bag of tasks + batch

High throughput

High performance

on-prem **Public clouds**

Web services

Horizontal scaling

Distributed load balancing

High availability

on-prem **Public clouds** **Edge**

AI Workflow: Platform requirements

Data preparation

Workflow of steps (e.g., remove hate and profanity, deduplicate, etc.)

Distributed training

Long-running job on massive infrastructure

Model adaptation

Model tuning with custom data set for downstream tasks

Inference

May have sensitivity to latency, throughput, power,

Hours to days	weeks to months	minutes to hours	sub-second API request
10-2000+ low to mid-end CPU cores 10+ low to mid-end GPUs per 10-100+ concurrent jobs	10-500+ high-end GPUs (per job) 10+ concurrent jobs	1+ mid to high-end GPU (per job) 100+ concurrent jobs	Single low-end GPU per fine tuning task **Fraction** to multiple GPUs per inference, or specialized accelerator Thousands of API requests
on-prem **Public clouds**	**on-prem** **Public clouds**	**on-prem** **Public clouds**	**on-prem** **Public clouds** **Edge**

AI Workflow summary

- AI workloads cover a range of job types
  - Bag of tasks – pre-processing tools
  - Batch jobs – training software
  - Model adaptation – various tools
  - Inference – web services

- Supporting these workloads require a cloud platform

- Traditional HPC systems are good at supporting bag of tasks and batch but not the others

Agenda

- Lecture
  - AI workflows
  - **Introduction to Containers**
  - **Docker and ecosystem**
  - Container orchestration
  - Kubernetes
- Lab
  - Use Docker on your virtual machine
  - Use Container to run a web application on your laptop

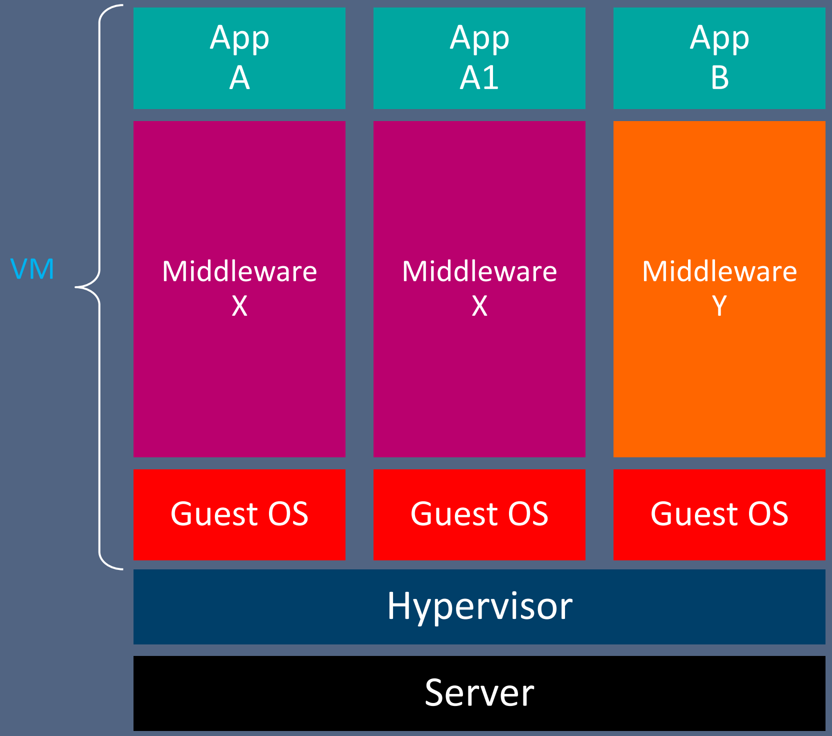
Use case for Containers

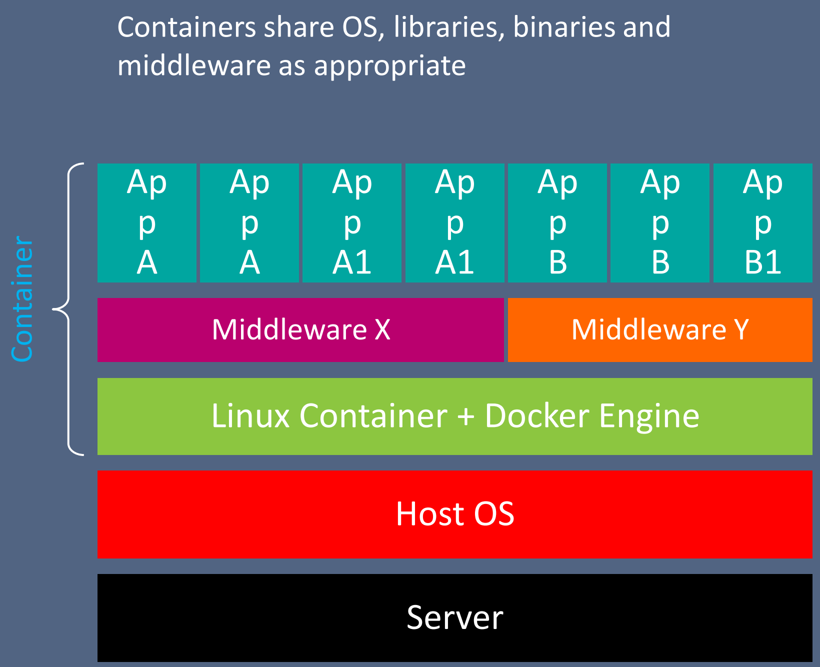
If you look at Cloudera's first attempt at a cloud offering, Altus, it was based on deployment via virtual machines (VMs), a process that typically took about 8 minutes to spin up clusters. With Docker and Kubernetes on CDP, that goes down to 30 seconds.

https://www.zdnet.com/article/where-does-cloudera-go-from-here/

What's a Container and how it differs from a VM?

- The concept of containers emerged a decade ago (e.g. Sun Solaris Zones and IBM AIX's WPARs). **Docker** is built on open source container capabilities inside Linux kernel (cgroups, namespaces, selinux, etc)
- A container encapsulates an application and its dependencies which run in an isolated process on the host's operating system (all application share the same OS)
- Traditional hardware virtualization creates an entire virtual machine. Each VM contains not only the application (which may only be 10's of MB) but must include and an entire Guest operating System (which may measure in 1-10s of GB).

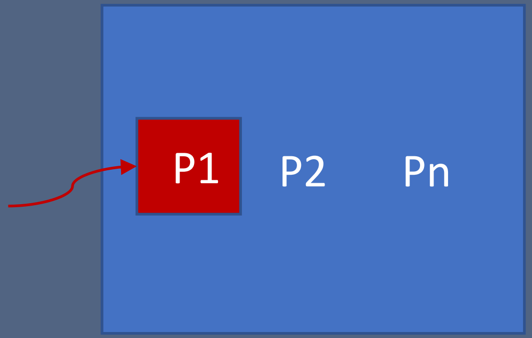




What is a container?

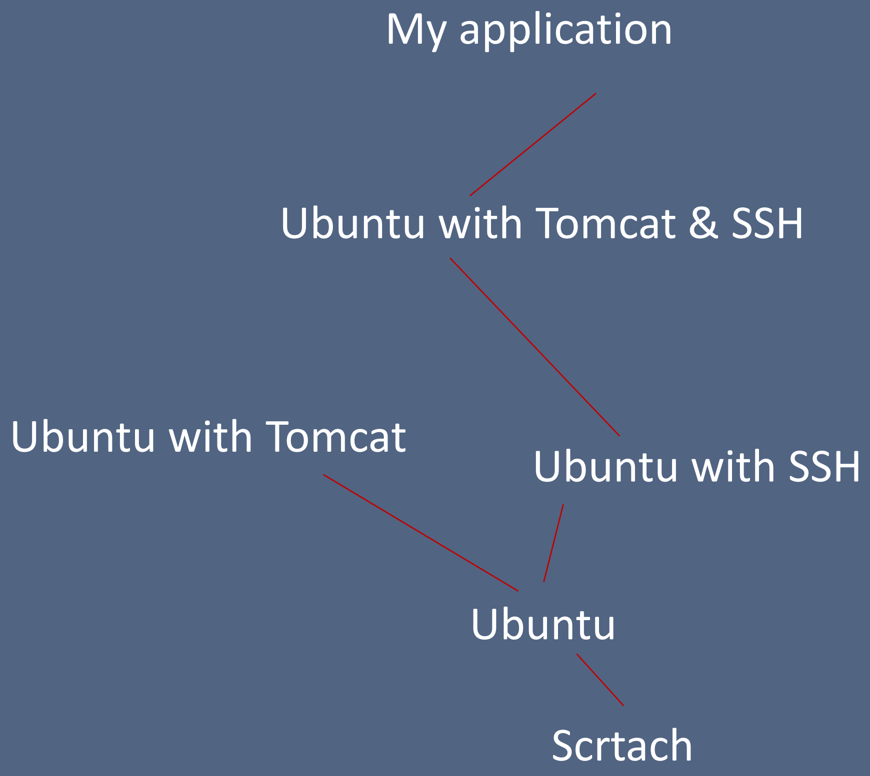
Runtime: A sandbox for a process

Name space (pid, network)
Cgroups (Memory, CPU, GPU)



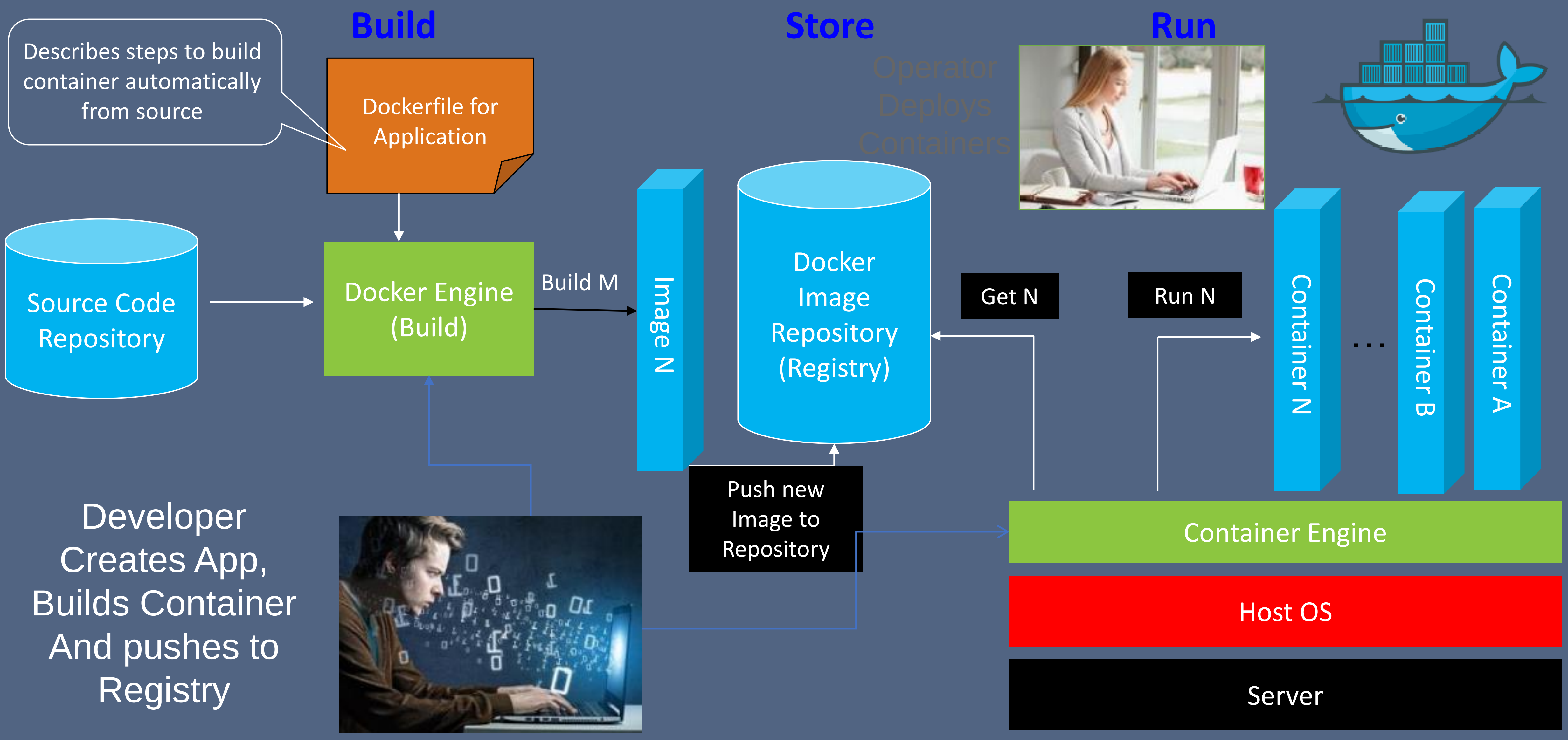
Host: Linux

Image

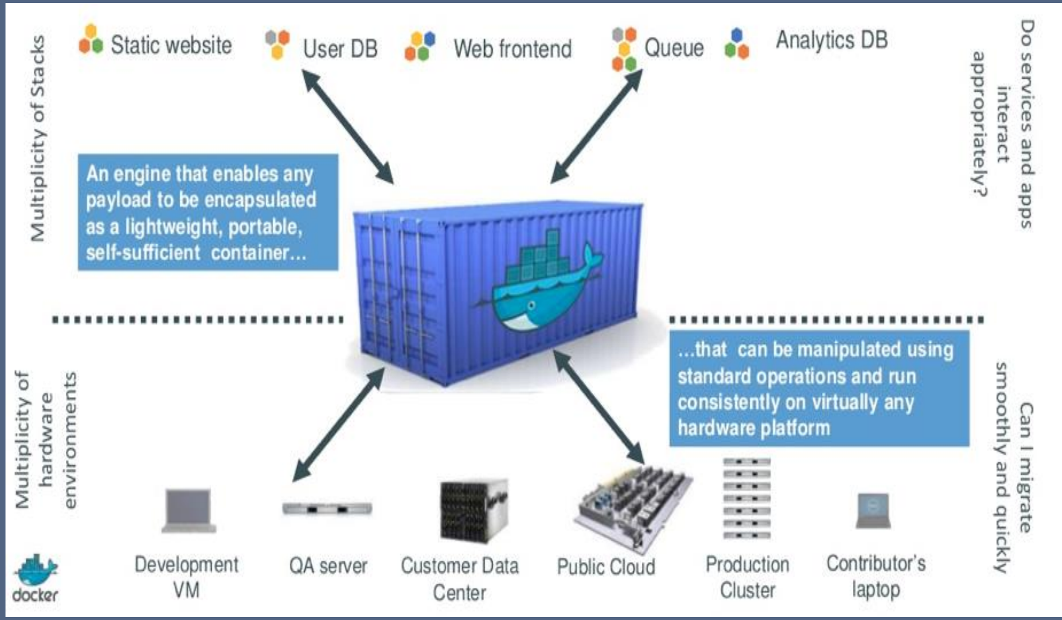


Dockerfile

What are the Basic Functions of Containers



Docker is an evolution of the Container Technology



Dev ⟷ Test ⟷ Production

Light weight virtualization container
- Portable
- Fast start
- Lightweight
- Tool and container ecosystem

Potential impact
- DevTest
- Platform as a Service
- Cloud Platform Services
- Software delivery

- Provision in seconds / milliseconds
- Near bare metal runtime performance
- VM-like agility – it's still "virtualization"
- Flexibility
  - Containerize "application(s)"
  - Deliver Polyglot apps
  - Repeatability
- Lightweight
- Open source – free – lower TCO
- Supported OOTB modern Linux kernel
- Runs on baremetal
- Growing set of tools and ecosystem
- Versioning and Portability
- Others: containerd, cri-o, lmctfy, rkt, wpar, Solaris zones

Containers vs VMs

Containers	VMs
Share the host operating system	Each VM has an operating system
Light weight	Heavy weight
Native performance	Some overhead
Memory and CPU can be changed flexibly	Change in Memory/CPU changes require reboot
Isolation at the process	Fully isolated
Virtually no startup time	Startup time in minutes
Support for HW like GPUs/FPGA not fully mature	Mature support for HW

Containers and VMs will co-exist for a long time

Docker by the numbers

80B
Container downloads

32,000+
GitHub Stars

200+
Meetups Around the Globe

650+
Commercial Customers

2M
Dockerized Applications in Hub

100K+
Third-party projects using Docker

Feb 2019

Docker by the numbers

105B
Container downloads

750+
Docker Enterprise Customers

200+
Meetups around the Globe

32,000+
GitHub Stars

5.8M
Dockerized Apps on Hub

100K+
3rd-party projects using Docker

Oct 2019

Docker Layered Filesystem

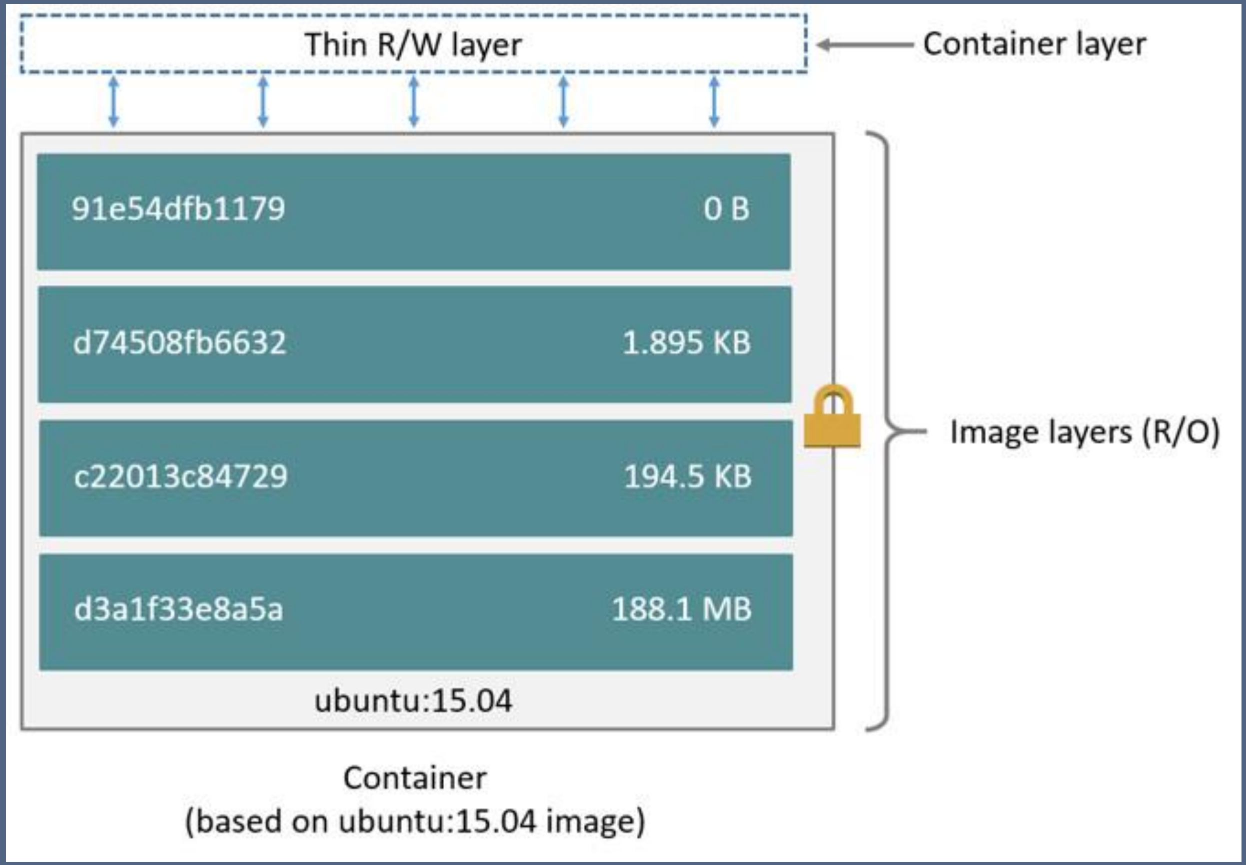
- Docker uses a Copy-On-Write layered filesystem
  - Only changes from the read-only layers are copied
- You can see the layers when you pull or push an image

```
$ docker pull ubuntu:15.04

15.04: Pulling from library/ubuntu
1ba8ac955b97: Pull complete
f157c4e5ede7: Pull complete
0b7e98f84c4c: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:5e279a9df07990286cce22e1b0f5b0490629ca6d187698746ae5e28e604a640e
Status: Downloaded newer image for ubuntu:15.04
```

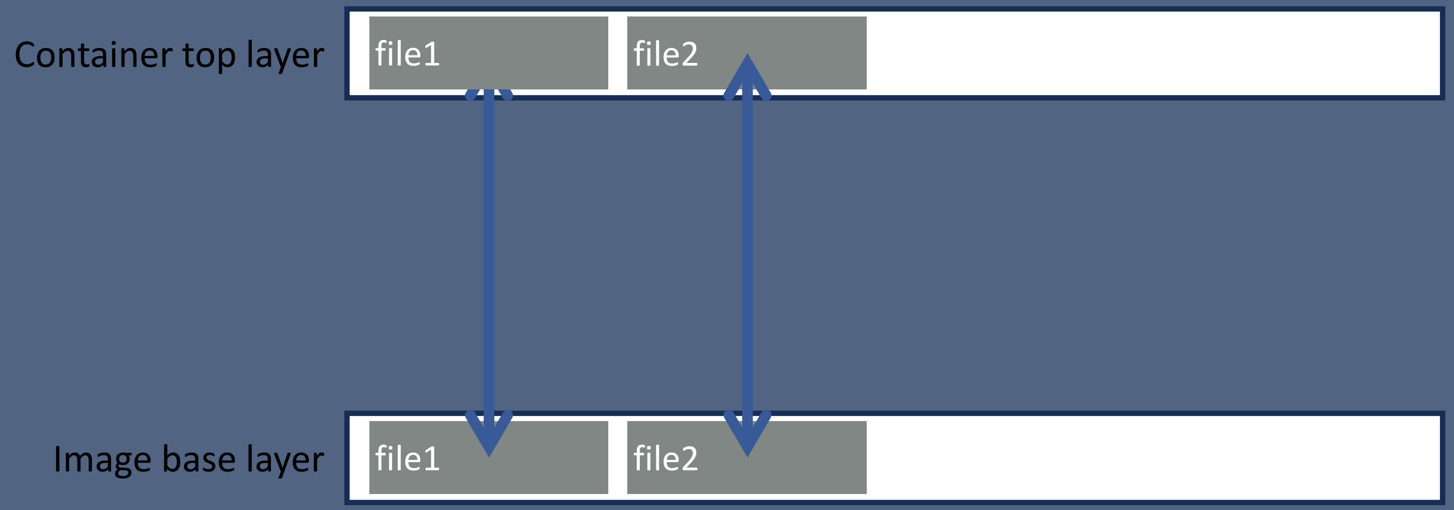
Images and Layers

- Each Docker image references a list of read-only layers that represent filesystem differences
- Layers are stacked on top of each other to form a base for a container's root filesystem
- When you create a new container, you add a new, thin, writable layer on top of the underlying stack
- All changes made to the running container - such as writing new files, modifying existing files, and deleting files - are written to this thin writable container layer



Layers are exposed to Top

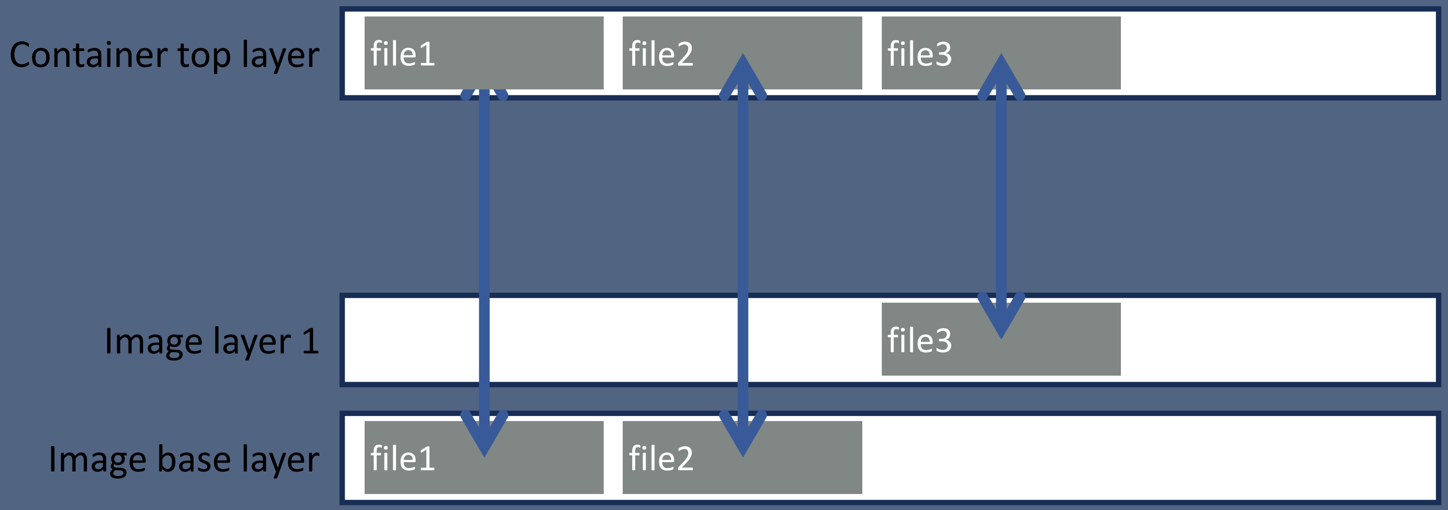
- Files from the read-only layers below are visible to the top layer



Docker container
(AUFS storage-driver demonstrating whiteout file)

Add Layers with more Files

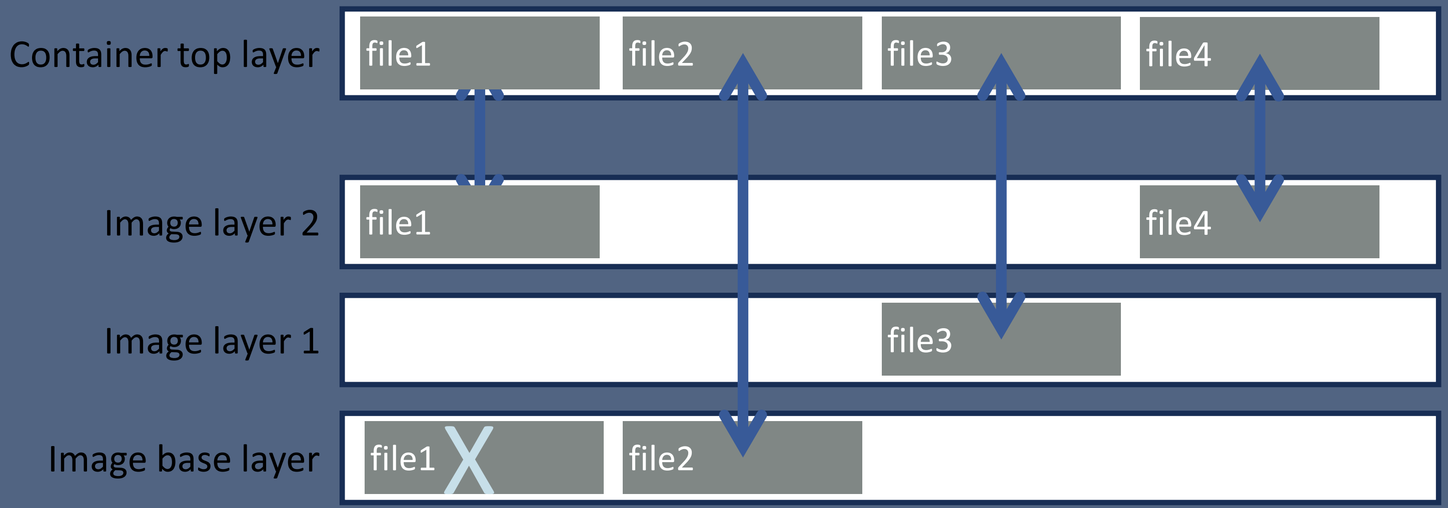
- As you add layers more files become visible at the top layer



Docker container
(AUFS storage-driver demonstrating whiteout file)

New Versions

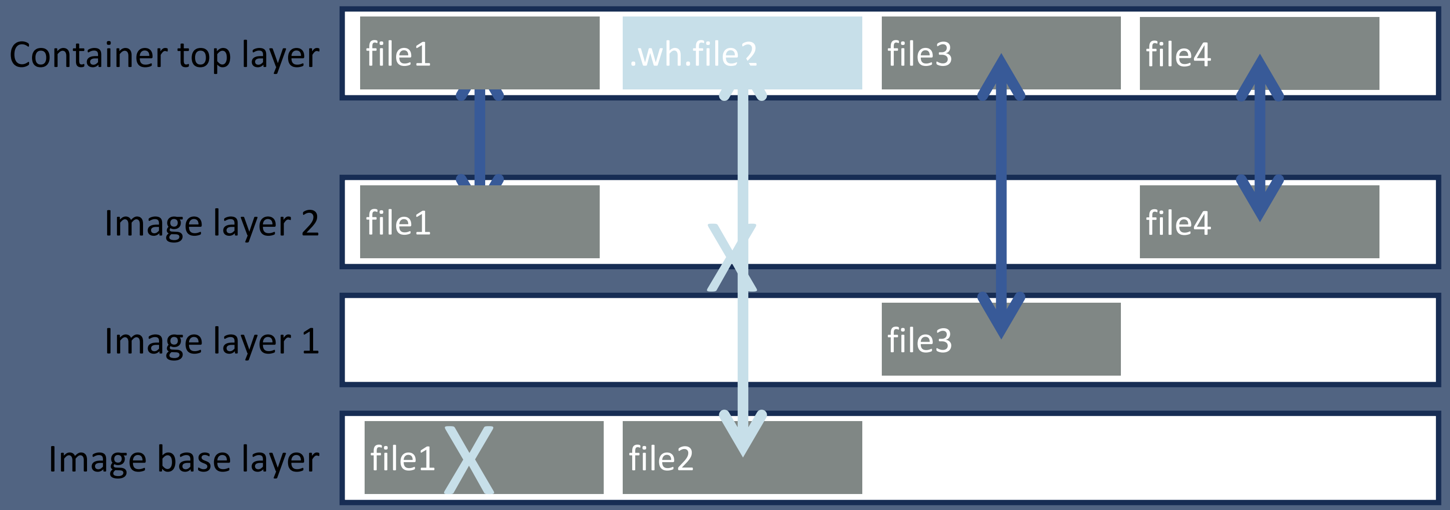
- A new version of `file1` is added and it hides the old `file1` version



Docker container
(AUFS storage-driver demonstrating whiteout file)

White-out Files

- Special files called "white-out" files are used to make files appear to be deleted



Docker container
(AUFS storage-driver demonstrating whiteout file)

How Layers are Reused

- Note when you provision with Vagrant how Redis:Alpine reuses Alpine layer:

Alpine layer is being reused by Redis:Alpine

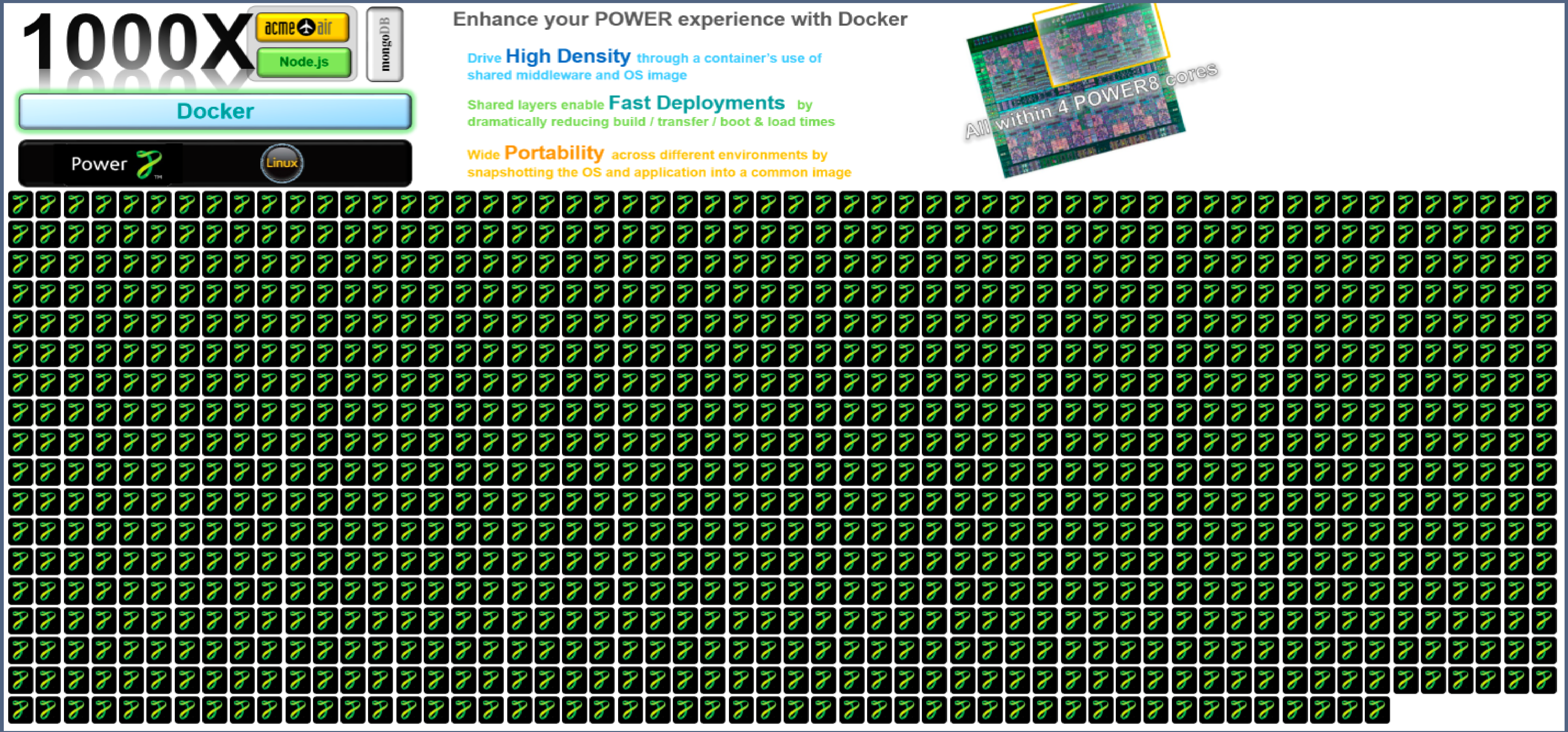
```
==> default: Running provisioner: docker...
    default: Installing Docker onto machine...
==> default: Pulling Docker images...
==> default: -- Image: alpine:latest
==> default: stdin: is not a tty
==> default: latest: Pulling from library/alpine
==> default: 3690ec4760f9: Pulling fs layer
==> default: 3690ec4760f9: Verifying Checksum
==> default: 3690ec4760f9: Download complete
==> default: 3690ec4760f9: Pull complete
==> default: Digest: sha256:1354db23ff5478120c98[illegible]eca1611a51c9f2b88b61f24283ee8200bf9a54f2e5c
==> default: Status: Downloaded newer image for alpine:latest
==> default: -- Image: redis:alpine
==> default: stdin: is not a tty
==> default: alpine: Pulling from library/redis
==> default: 3690ec4760f9: Already exists
==> default: 5e231f7bdf9d: Pulling fs layer
==> default: 5a74fb2950f8: Pulling fs layer
. . .
```

What Can you Do with Docker?

- You can run **Containers** from the Images in the Docker Registry (e.g., Docker Hub)
- You can build **Docker Images** that hold your applications and their dependancies
- You can create **Docker Containers** from those Docker images to run your applications
- You can share those **Docker images** via Docker Hub or your own Docker registry
- You can pull those **images** from the Docker registry to **deploy them as Containers** on a server running Docker Engine

Docker ecosystem

- Docker github:
  - https://github.com/docker
- Docker registry
  - https://hub.docker.com
- Orchestration
  - Docker swarm, Kubernetes, Mesos, OpenStack, etc



1000K docker contianers on a single POWER system and 2Million on a single Z system

Docker at insane scale on IBM Power Systems
https://www.ibm.com/blog/docker-insane-scale-on-ibm-power-systems/

Lab objectives

- Use Docker on your virtual machine

- Use Container to run an application on your laptop

4 ways to work with Docker

- In this class we are going to use Vagrant but I want you to be aware of all of your options:

  1. Docker for macOS — Mac only

  2. Docker for Windows — Windows only

  3. Docker Toolbox — Older Mac and Windows

  4. Vagrant and Virtualbox — Mac, Windows, and Linux

Sample Docker file for the exercise

- Clone this repo: https://github.com/nyufall2019/vg-ibmcloud
- git clone https://github.com/nyufall2019/vg-ibmcloud.git
- Move to the directory and fire up your VM (vagrant up, vagrant ssh)

Now deploy the application with the Docker

- Notice the Dockerfile in ` vg-ibmcloud`
- Git pull to sync
- Build the application docker image
  - docker build -t hello-app .
- Run the application:
  - docker run -p 8001:8001 -it hello-app

Suggested Study Material

- Docker adoption from Data dog
  - https://www.datadoghq.com/docker-adoption/
- History of containers
- Download and try different docker containers
- Study how the layer file system works in Docker:
  - https://www.slideshare.net/DmitrySkaredov/the-docker-ecosystem