IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
MIDLAND DIVISION

INTELLECTUAL VENTURES I LLC and INTELLECTUAL VENTURES II LLC,	)	
Plaintiffs,	)	C.A. No. 7:24-cv-00277-ADA
v.	)	
	)	**JURY TRIAL DEMANDED**
SOUTHWEST AIRLINES CO.,	)	
Defendant.	)	

PLAINTIFFS' DISCLOSURE OF
PRELIMINARY INFRINGEMENT CONTENTIONS

Pursuant to the Court's Standing Order Governing Proceedings ("OGP"), Plaintiffs Intellectual Ventures I LLC and Intellectual Ventures II LLC (collectively, "Intellectual Ventures," "IV," or "Plaintiffs"), hereby provides its Initial Infringement Contentions to Defendant Southwest Airlines Co. ("Southwest" or "Defendant") for United States Patent No. 8,332,844 ("the '844 Patent"), United States Patent No. 8,407,722 ("the '722 Patent"), United States Patent No. 7,949,785 ("the '785 Patent"), United States Patent No. 8,027,326 ("the '326 Patent), United States Patent No. 7,324,469 ("the '469 Patent"), and United States Patent No. 7,257,582 ("the '582 Patent") (collectively, the "Asserted Patents" or "Patents-in-Suit").

IV makes this disclosure based on the information presently available to it. Discovery in this case has not started, and IV reserves its right to amend or supplement these disclosures as permitted by the Federal Rules of Civil Procedure, by the local rules of the Western District of Texas, and by order of the Court.

For each Asserted Patent, Plaintiffs identify the following Accused Instrumentalities of which it is currently aware. The identification of Accused Instrumentalities is based on Plaintiffs' research and analysis to date, without the benefit of discovery from the Defendant. Plaintiffs

reserve the right to add, delete, substitute or otherwise amend this list of Accused Instrumentalities based on discovery or other circumstances, in a manner consistent with the Federal Rules of Civil Procedures, local rules, and standing order.

The Accused Instrumentalities include, without limitation, the following:

- ’844 Patent. The Accused Instrumentalities include, without limitation, Southwest systems that utilize Docker, and all past, current and future Southwest products that operate in the same or substantially similar manner as the specifically identified products and components, and all past, current and future Southwest products that have the same or substantially similar features as the specifically identified products and components provided in Exhibit 1.

- ’722 Patent. The Accused Instrumentalities include, without limitation, Southwest systems that utilize Kafka, and all past, current and future Southwest products that operate in the same or substantially similar manner as the specifically identified products and components, and all past, current, and future Southwest products that have the same or substantially similar features as the specifically identified products and components provided in Exhibit 2.

- ’785 Patent. The Accused Instrumentalities include, without limitation, Southwest systems that utilize Kubernetes, and all past, current and future Southwest products that operate in the same or substantially similar manner as the specifically identified products and components, and all past, current and future Southwest products that have the same or substantially similar features as the specifically identified products and components provided in Exhibit 3.

- ’326 Patent. The Accused Instrumentalities include, without limitation, Southwest systems that provide Wi-Fi Access Points that support at least IEEE 802.11n and/or 802.11ac, and all past, current, and future Southwest products that operate in the same or substantially similar manner as the specifically identified products and components, and all past, current and future Southwest products that have the same or substantially similar features as the specifically identified products and components. Exhibit 4.

- ’469 Patent. The Accused Instrumentalities include, without limitation, Southwest systems that provide satellite-based onboard WiFi, and all past, current and future Southwest products that operate in the same or substantially similar manner as the specifically identified products and components, and all past, current and future Southwest products that have the same or substantially similar features as the specifically identified products and components. Exhibit 5.

- ’582 Patent. The Accused Instrumentalities include, without limitation, Southwest systems that utilize Hadoop and/or Spark, and all past, current and future Southwest products that operate in the same or substantially similar manner as the specifically identified products and components, and all past, current and future Southwest products that have the same or substantially similar features as the specifically identified products and components provided in Exhibits 6 and 7.

- All past, current and future Southwest products and services that operate in the same or substantially similar manner as the specifically identified products and services above and

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 2

described in Exhibits 1-7 (claim charts identifying specifically where each element of each asserted claim is found within each Accused Instrumentality).

- All past, current and future Southwest products and services that have the same or substantially similar features as the specifically identified products and services above and described in Exhibits 1-7, (claim charts identifying specifically where each element of each asserted claim is found within each Accused Instrumentality).

Plaintiffs' infringement contentions apply to the Accused Instrumentalities as well as all other past, current and future hardware and software products and services developed, made, used, offered for sale, sold, imported, and provided by Southwest that contain or make use of the Patented Technology.[1] This identification of Accused Instrumentalities is based upon Plaintiffs' diligent investigation to date, and without the benefit of any discovery from the Defendant and without the benefit of the Court's claim construction. Plaintiffs expressly reserve the right to add, delete, substitute or otherwise amend this list of Accused Instrumentalities based on discovery, the Court's claim construction, or other circumstances, in a manner consistent with the Federal Rules of Civil Procedure and the Court's rules, including the local Patent Rules.

These preliminary infringement contentions are based upon publicly available information and Plaintiffs' diligent investigation to date, and without the benefit of any substantial discovery from the Defendant and without the benefit of the Court's claim construction. Plaintiffs anticipate that discovery may provide additional evidence of Defendant's infringement, may lead to the discovery of additional instances of infringement, and may also enable identification of additional claims that are infringed by Defendant.

Based upon currently available information, IV asserts that Southwest has infringed and/or continues to infringe the patents and claims identified in the attached claim charts (the "Asserted Claims" of the Patents-in-Suit). Infringement claim charts evidencing the correspondence between (i) the elements of the Asserted Claims, and (ii) the corresponding items of the Accused

[1] "Patented Technology" means all technologies described in the claims of the Patents-in-Suit.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 3

Instrumentalities are attached hereto. Further, Exhibits 1-7, which are attached hereto and incorporated by reference, are exemplary infringement claim charts identifying specifically where each limitation of each Asserted Claim is found within each Accused Instrumentality or practiced by each Accused Instrumentality.

Plaintiffs assert that Defendant has directly infringed and continues to directly infringe the Asserted Claims literally through the Accused Instrumentalities by making, using, offering for sale, selling, and/or importing into the United States the Accused Instrumentalities. To the extent that Defendant alleges that one or more limitations of the Asserted Claims are not literally found in the Accused Instrumentalities, Plaintiffs allege that such limitations are found in or practiced by the Accused Instrumentalities under the doctrine of equivalents. Any differences alleged to exist between any of the Asserted Claims and any of the Accused Instrumentalities are insubstantial and that each Accused Instrumentality also meets each limitation under the doctrine of equivalents as the identified features of the Accused Instrumentality performs substantially the same function in substantially the same way to achieve substantially the same result as the corresponding claim limitation. IV reserves the right to assert infringement solely under the doctrine of equivalents with respect to any particular claim element(s), if warranted by discovery, further analysis, and/or claim constructions in this case.

These assertions are based upon Plaintiffs' diligent investigation to date, and without the benefit of any substantial discovery from the Defendant and without the benefit of the Court's claim construction. Plaintiffs reserve the right to add, delete, substitute or otherwise amend these infringement allegations based on discovery, the Court's claim construction, or other circumstances, in a manner consistent with the Federal Rules of Civil Procedure and the Court's rules, including the local Patent Rules.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 4

Plaintiffs further assert that Defendant has indirectly infringed and continues to indirectly infringe by actively inducing infringement of one or more of the claims of the Asserted Patents through the Accused Instrumentalities. *See* Exhibits 1-7.

Plaintiffs also assert that these third parties directly infringe at least one or more of the claims of the Asserted Patents through the manufacture, use, sale, offer to sell, or importation of the Accused Instrumentalities. *See* Exhibits 1-7.

For example, Defendant has actively induced infringement by encouraging the use of the Accused Instrumentalities in ways that infringe each Asserted Claim. Defendant knew or should have known that such encouragement would induce infringement. Defendant has taken active steps with the specific intent to encourage and cause others to use each Accused Instrumentality in ways that infringes each Asserted Claim. Such active steps by Defendant with specific intent to induce infringement have included, among other things, advertising, promoting, marketing, making available for use, offering to sell, and/or selling the Accused Instrumentalities to others; encouraging and influencing others to import, offer to sell, and/or sell the Accused Instrumentalities; directing and instructing others to use the Accused Instrumentalities in infringing ways, including providing and promoting the described hardware and/or software components and features that when used in their normal and customary ways as intended and designed; and by providing the Accused Instrumentalities to others. Southwest has performed the aforementioned steps with the knowledge of the Asserted Patents through receipt of a letter dated September 30, 2024, and received by Defendant on the same date, and also as of the date when the complaint in this case was filed. Southwest has known or should have known that the acts it has induced constitute infringement because, for instance, it has been aware that its customers, employees, and/or third parties, such as vendors, will use the Accused Instrumentalities, resulting in direct infringement.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 5

IV further reserves the right to supplement or revise its infringement contentions and/or charts. Such supplement may be based upon, for example, further discovery. Further, IV reserves the right to supplement or revise its infringement contentions and/or chart(s), including identification of additional asserted claims, based on, for example, new versions or variations of one or more of the accused products that are later discovered.

The Asserted Claims of the Patents in Suit are entitled to at least the priority dates identified below:

- Each of the Asserted Claims of the ’844 Patent is entitled to a priority date of no later than December 30, 2004.
- Each of the Asserted Claims of the ’722 Patent is entitled to a priority date of no later than December 18, 2000.
- Each of the Asserted Claims of the ’785 Patent is entitled to a priority date of no later than March 21, 2003.
- Each of the Asserted Claims of the ’326 Patent is entitled to a priority date of no later than January 12, 2004.
- Each of the Asserted Claims of the ’469 Patent is entitled to a priority date of no later than September 29, 2003.
- Each of the Asserted Claims of the ’582 Patent is entitled to a priority date of no later than March 13, 2002.

The subject matter described by the Asserted Claims may have been conceived and reduced to practice prior to these dates. Plaintiffs’ research and analysis is ongoing, and Plaintiffs reserve the right to assert that the claims are entitled to a priority date that is earlier than the above dates.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 6

Dated: February 10, 2025

Respectfully submitted,

By: */s/ Mark D. Siegmund*

Mark D. Siegmund (TX Bar No. 24117055)
msiegmund@cjsjlaw.com
CHERRY JOHNSON SIEGMUND JAMES PLLC
7901 Fish Pond Road
Ste 2nd Floor
Waco, TX 76710
254-732-2242
Fax: 866-627-3509
Email: msiegmund@cjsjlaw.com

Jonathan K. Waldrop (CA Bar No. 297903)
(Admitted in this District)
jwaldrop@kasowitz.com
Darcy L. Jones (CA Bar No. 309474)
(Admitted in this District)
djones@kasowitz.com
Marcus A. Barber (CA Bar No. 307361)
(Admitted in this District)
mbarber@kasowitz.com
John W. Downing (CA Bar No. 252850)
(Admitted in this District)
jdowning@kasowitz.com
Heather S. Kim (CA Bar No. 277686)
(Admitted in this District)
hkim@kasowitz.com
ThucMinh Nguyen (CA Bar No. 304382)
(Admitted in this District)
tnguyen@kasowitz.com
Jonathan Hicks (CA Bar No. 274634)
(Admitted in this District)
jhicks@kasowitz.com
KASOWITZ BENSON TORRES LLP
333 Twin Dolphin Drive, Suite 200
Redwood Shores, California 94065
Telephone: (650) 453-5170
Facsimile: (650) 453-5171

Attorneys for Plaintiffs
INTELLECTUAL VENTURES I LLC and
INTELLECTUAL VENTURES II LLC

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 7

CERTIFICATE OF SERVICE

I hereby certify that a true and correct copy of the foregoing instrument was served or delivered electronically, via email, to all counsel of record, on this 10th day of February 2025.

/s/ Jonathan K. Waldrop
Jonathan K. Waldrop

Exhibit 1 to
Intellectual Ventures I LLC's and Intellectual Ventures II LLC's
Preliminary Infringement Contentions

Infringement Claim Chart of
U.S. Patent No. 8,332,844 ("the '844 Patent")

The Accused Systems and Services include without limitation Southwest systems and services that utilize Docker; all past, current, and future systems and services that operate in the same or substantially similar manner as the specifically identified systems and services; and all past, current, and future Southwest systems and services that have the same or substantially similar features as the specifically identified systems and services ("Example Southwest Count 1 Systems and Services" or "Southwest Systems and Services").[1]

On information and belief, Southwest's Systems and Services directly infringe the asserted claims of the '844 patent, either literally or under the doctrine of equivalents, through at least using Docker in its various systems. In addition, Southwest directly infringes the '844 patent by testing the Southwest Systems and Services and its various systems that use Docker. On information and belief, Southwest, with knowledge at least since the time of filing the complaint, also indirectly infringes the '844 Patent by inducing its employees, customers, and other third parties (such as vendors) to use Southwest Systems and Services, and provides documents that include instructions regarding how to use Southwest products and services that use Docker in the Southwest Systems and Services in an infringing manner. On information and belief, Southwest supplies its employees, customers, and other third parties with access to the Southwest Systems and Services through various products and services that allows its employees, customers, and other third parties to use the claimed method. On information and belief, Southwest provides instructions on how to use its various products and services that use Docker in the Southwest Systems and Services in an infringing manner, the use of which results in infringement of the '844 Patent claims through performance of the claimed methods.

Consistent with this Court's Standing Order Governing Proceedings (OGP), IV identifies Docker as used in Southwest's systems and offerings as the Accused Instrumentality. Information regarding how specifically Southwest implements Docker, including for which products and offerings, is non-public. IV intends to obtain discovery regarding how the Docker is used in Southwest products and services and will update its infringement positions once this discovery is completed.

IV does not intend this exemplary claim chart to be limiting, and IV reserves its rights to pursue other accused instrumentalities, patent claims, evidence, and infringement arguments in this case. Discovery has yet to begin, and this case is still in its initial stages. Accordingly, IV reserves the right to amend and/or supplement these contentions to the full extent allowed by the Court, including but not limited to, incorporating additional information, claims, theories, and / or accused products.

[1] For the avoidance of doubt, Plaintiffs do not accuse public clouds of Southwest if those services are provided by a cloud provider with a license to Plaintiffs' patents that covers Southwest's activities. IV will provide relevant license agreements for cloud providers in discovery. To the extent any of these licenses are relevant to Southwest's activities, Plaintiffs will meet and confer with Southwest about the impact of such license(s).

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 10

On information and belief, the Southwest Systems and Services use Docker in Southwest's private cloud(s). For example, Southwest posts, or has posted, job opportunities that require familiarity with Docker containerization concepts.

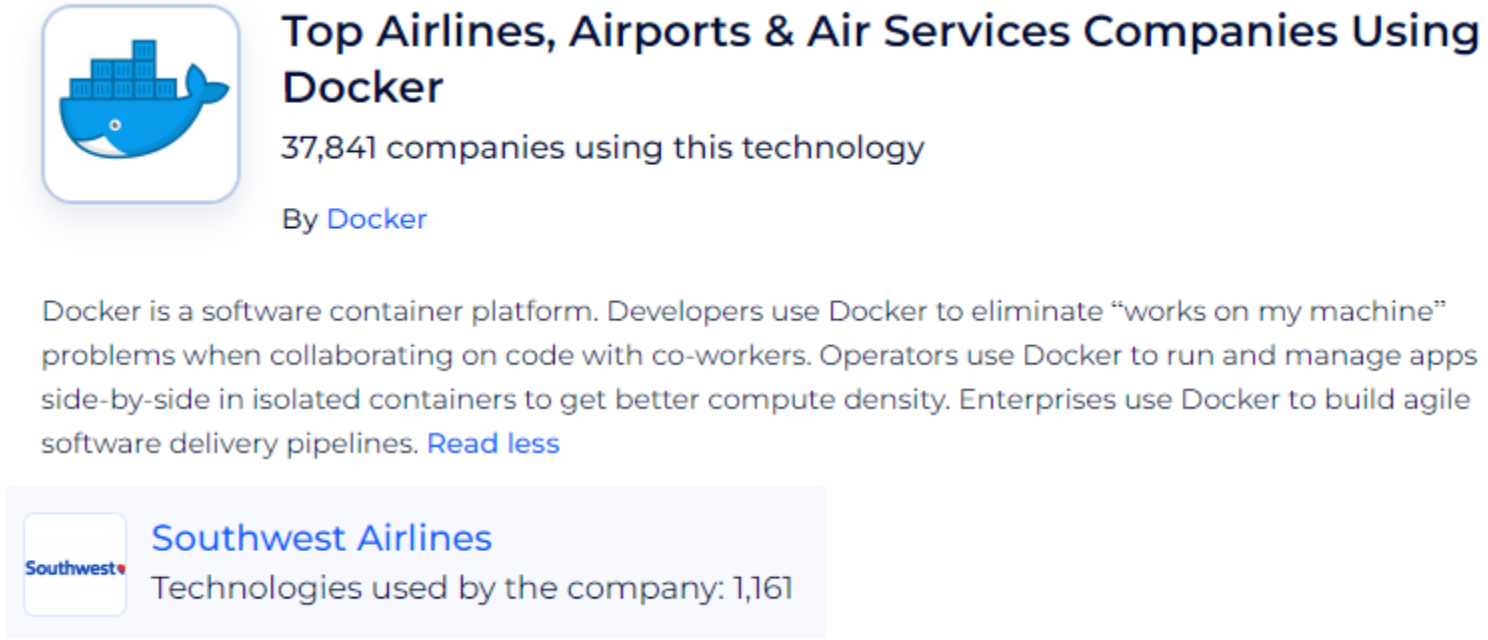
See https://www.linkedin.com/in/charlesmarshall-eth/, the job profile of an associate software engineer listing Docker as a skill. (last accessed 9/23/2024).

See https://www.linkedin.com/in/bhaveshkar-kongari/, the job profile of a senior DevOps Engineer listing usage of Docker. (last accessed 9/24/24).

As another example, Southwest has stated that it is investing in cloud technology and has "moved about 50% of its technology" to the cloud and has indicated cloud migration is one of its areas of focus for 2024 and beyond. Source: https://www.phocuswire.com/southwest-airlines-cio-tech-investment.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 11

On information and belief, other information confirms Southwest uses Docker technology.



Source: https://www.zoominfo.com/tech/23717/docker-tech-from-transportation-airline-industry-in-us-by-revenue.[2]

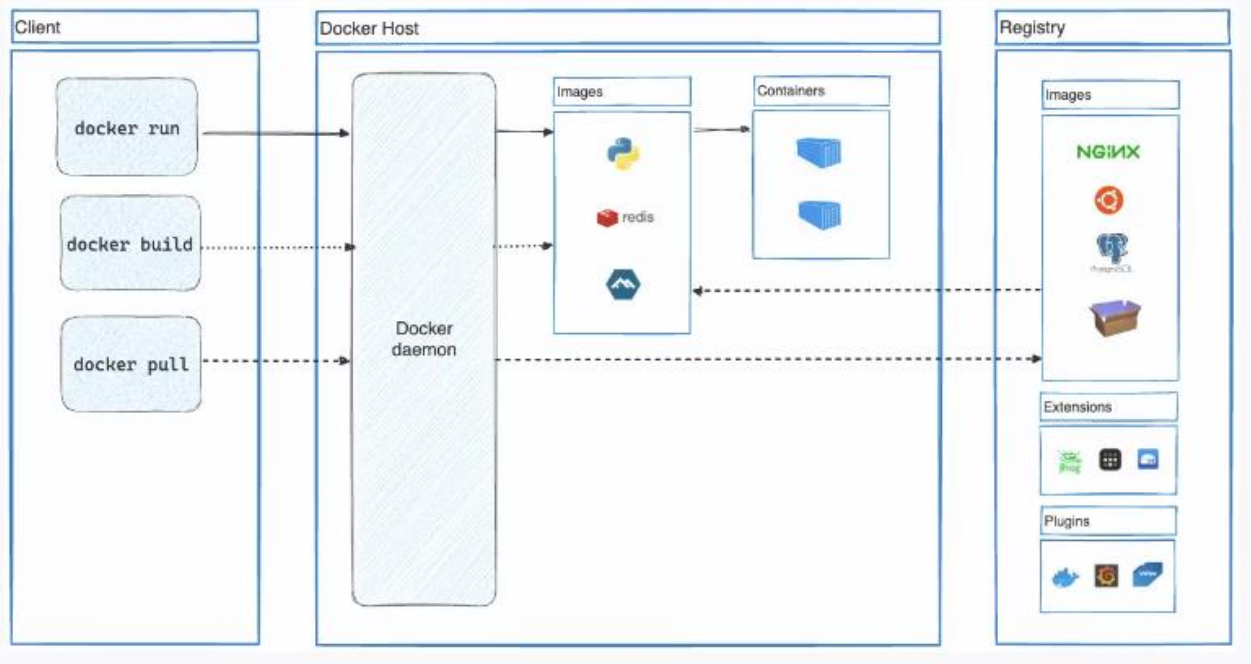
[2] All sources cited in this document were publicly accessible as of the filing date of the Complaint.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 12

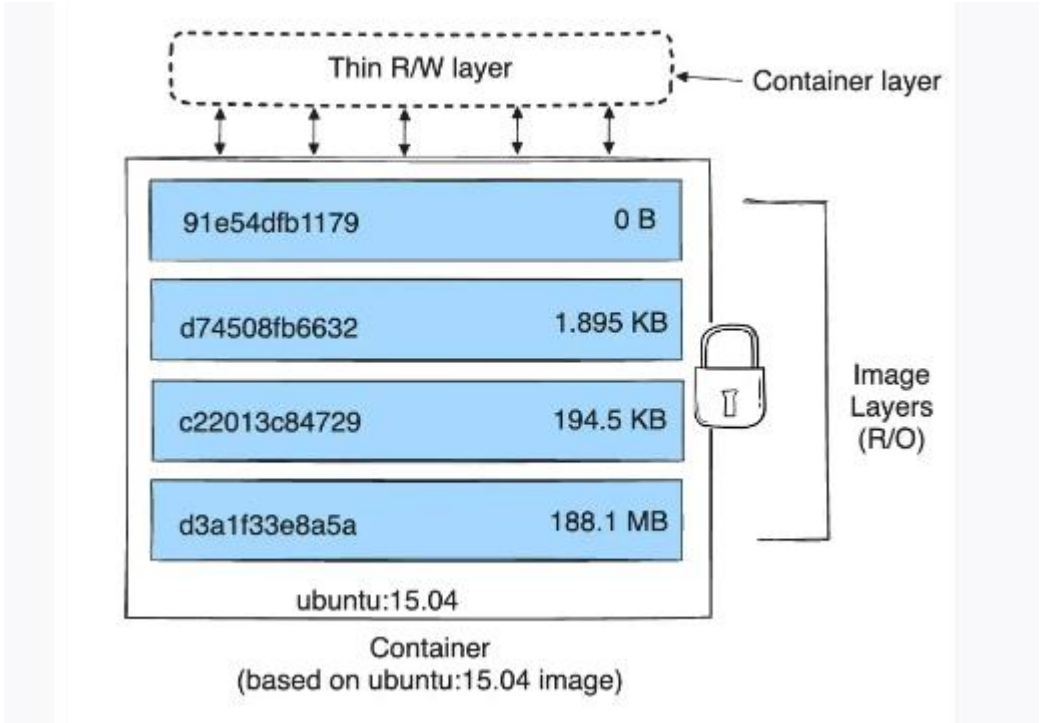
U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
[7.pre]. A method for providing data to a plurality of compute nodes, comprising:	To the extent this preamble is limiting, on information and belief, the Southwest Count 1 Systems and Services practice a method for providing data to a plurality of compute nodes. For example, on information and belief, Southwest's use of Docker provides "a method for providing data to a plurality of compute nodes." Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code, you can significantly reduce the delay between writing code and running it in production. Source: https://docs.docker.com/get-started/overview/. The Docker platform Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security lets you run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you don't need to rely on what's installed on the host. You can share containers while you work, and be sure that everyone you share with gets the same container that works in the same way. Source: https://docs.docker.com/get-started/overview/.

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	Docker provides tooling and a platform to manage the lifecycle of your containers: • Develop your application and its supporting components using containers. • The container becomes the unit for distributing and testing your application. • When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two. Source: https://docs.docker.com/get-started/overview/. On information and belief, Docker uses a client-server architecture. For example, the Docker architecture shown below shows a client, a Docker host, and a registry. Docker clients communicate with a Docker daemon, which builds, runs, and distributes Docker containers. Docker images are run within containers.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 14

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	 Source: https://docs.docker.com/get-started/overview/.
[7.a] storing blocks of a root image of said compute nodes on a first storage unit;	On information and belief, the Southwest Count 1 Systems and Services practice storing blocks of a root image of said compute nodes on a first storage unit. For example, on information and belief, Docker stores blocks of a root image including a collection of read-only layers in a Docker image in the Docker registry. The layers are stacked on top of each other. When you create a new container, you add a new writable layer on top of the underlying layers. This layer is often called the "container layer". 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. The diagram below shows a container based on an `ubuntu:15.04` image.

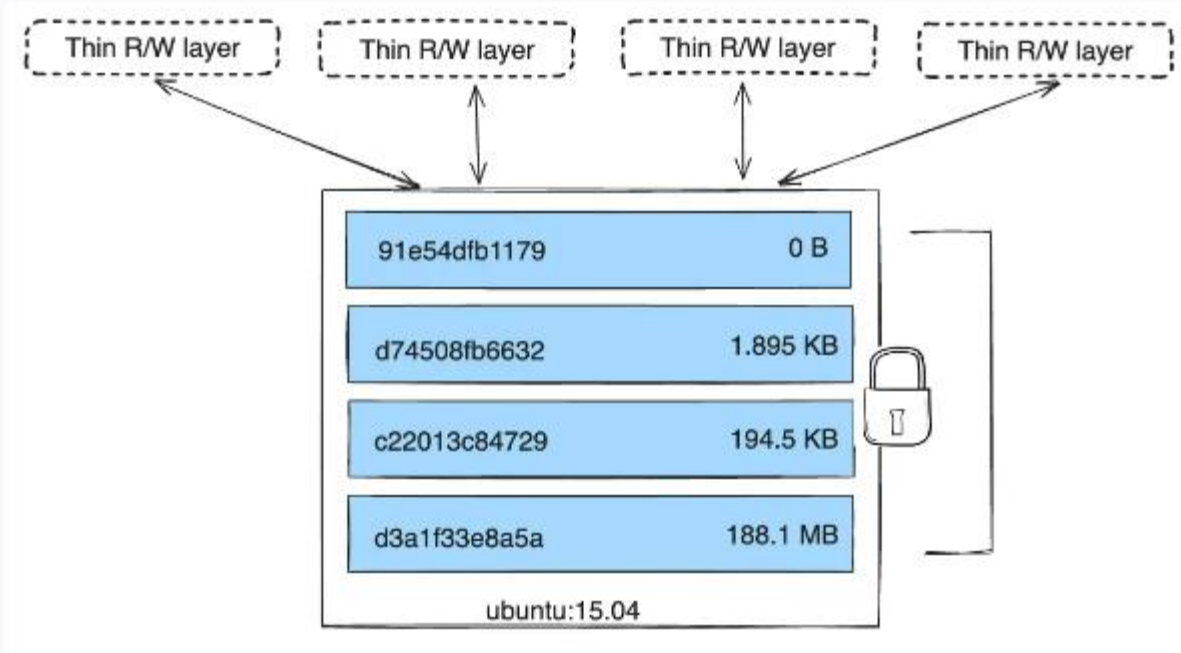
PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 15

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	Source: https://docs.docker.com/storage/storagedriver/.  Source: https://docs.docker.com/storage/storagedriver/. **Distribution Registry** **What it is** The Registry is a stateless, highly scalable server side application that stores and lets you distribute container images and other content. The Registry is open-source, under the permissive Apache license.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 16

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	Source: https://docs.docker.com/registry/. **Why use it** You should use the Registry if you want to: • tightly control where your images are being stored • fully own your images distribution pipeline • integrate image storage and distribution tightly into your in-house development workflow Source: https://docs.docker.com/registry/. On information and belief, the image below shows blocks of a root image (associated with ubuntu version 15.04) that is associated with a compute node.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 17

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	 Source: https://docs.docker.com/storage/storagedriver/. On information and belief, an image stores blocks of a root image, such as an ubuntu image or other operating systems software and other applications and software. An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization. For example, you may build an image which is based on the `ubuntu` image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run. Source: https://docs.docker.com/get-started/overview/.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 18

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	Docker registries A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker looks for images on Docker Hub by default. You can even run your own private registry. Source: https://docs.docker.com/get-started/overview/. On information and belief, Docker includes images that are built up from a series of layers, where each layer represents an instruction in the image's Dockerfile. Blocks of a root image are stored in a memory associated with a computer such as a server (node).

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 19

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**

Images and layers

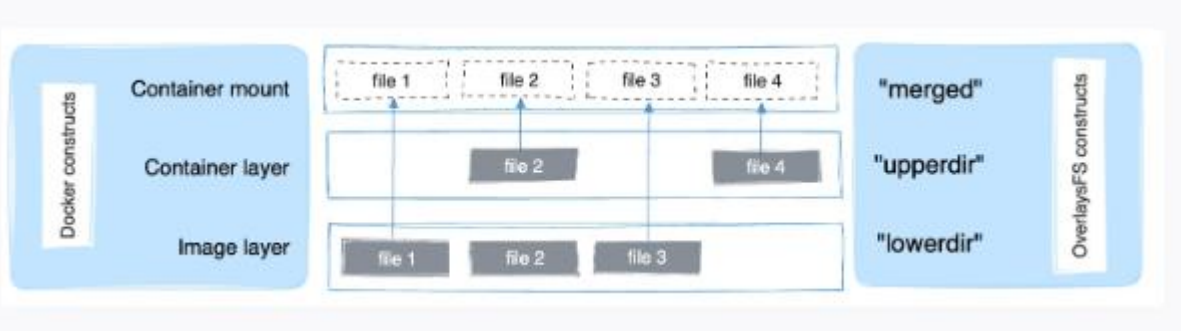
A Docker image is built up from a series of layers. Each layer represents an instruction in the image's Dockerfile. Each layer except the very last one is read-only. Consider the following Dockerfile:

```
# syntax=docker/dockerfile:1

FROM ubuntu:22.04
LABEL org.opencontainers.image.authors="org@example.com"
COPY . /app
RUN make /app
RUN rm -r $HOME/.cache
CMD python /app/app.py
```

This Dockerfile contains four commands. Commands that modify the filesystem create a layer. The `FROM` statement starts out by creating a layer from the `ubuntu:22.04` image. The `LABEL` command only modifies the image's metadata, and doesn't produce a new layer. The `COPY` command adds some files from your Docker client's current directory. The first `RUN` command builds your application using the `make` command, and writes the result to a new layer. The second `RUN` command removes a cache directory, and writes the result to a new layer. Finally, the `CMD` instruction specifies what command to run within the container, which only modifies the image's metadata, which doesn't produce an image layer.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 20

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	Source: https://docs.docker.com/engine/storage/drivers/. The major difference between a container and an image is the top writable layer. All writes to the container that add new or modify existing data are stored in this writable layer. When the container is deleted, the writable layer is also deleted. The underlying image remains unchanged. Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image. Source: https://docs.docker.com/engine/storage/drivers/. On information and belief, a Docker image and a Docker container can be layered in a manner as shown in the image below.  Source: https://docs.docker.com/engine/storage/drivers/overlayfs-driver/. *See also* preamble [7.pre].

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 21

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
[7.b] storing leaf images for respective compute nodes on respective second storage units,	On information and belief, the Southwest Count 1 Systems and Services practice storing leaf images for respective compute nodes on respective second storage units. For example, on information and belief, Docker stores leaf images including thin R/W layer in a runnable container for respective compute nodes on respective second storage units including Docker's local storage area. The layers are stacked on top of each other. When you create a new container, you add a new writable layer on top of the underlying layers. This layer is often called the "container layer". 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. The diagram below shows a container based on an `ubuntu:15.04` image. Source: https://docs.docker.com/engine/storage/drivers/.

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**

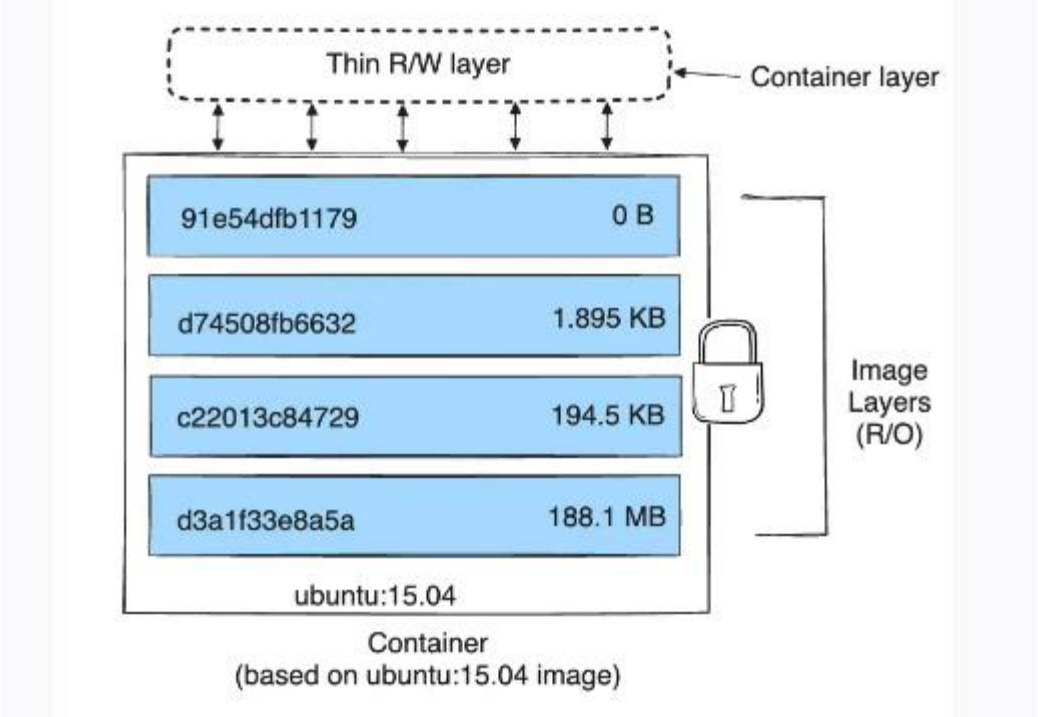
	When you use `docker pull` to pull down an image from a repository, or when you create a container from an image that doesn't yet exist locally, each layer is pulled down separately, and stored in Docker's local storage area, which is usually `/var/lib/docker/` on Linux hosts. You can see these layers being pulled in this example:
	
	`$ docker pull ubuntu:22.04`
	`22.04: Pulling from library/ubuntu`
	`f476d66f5408: Pull complete`
	`8882c27f669e: Pull complete`
	`d9af21273955: Pull complete`
	`f5029279ec12: Pull complete`
	`Digest: sha256:6120be6a2b7ce665d0cbddc3ce6eae60fe94637c6a66985312d11`
	`Status: Downloaded newer image for ubuntu:22.04`
	`docker.io/library/ubuntu:22.04`
	
	Source: https://docs.docker.com/engine/storage/drivers/.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 23

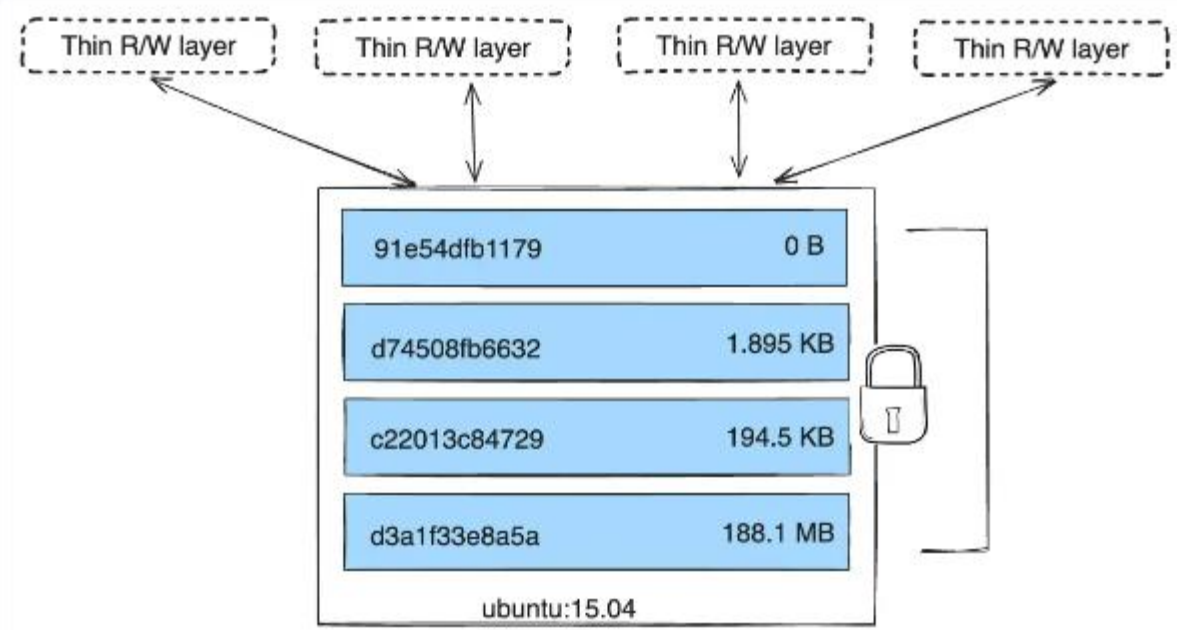
U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	# Images and layers A Docker image is built up from a series of layers. Each layer represents an instruction in the image's Dockerfile. Each layer except the very last one is read-only. Consider the following Dockerfile: ``` # syntax=docker/dockerfile:1 FROM ubuntu:22.04 LABEL org.opencontainers.image.authors="org@example.com" COPY . /app RUN make /app RUN rm -r $HOME/.cache CMD python /app/app.py ``` This Dockerfile contains four commands. Commands that modify the filesystem create a layer. The `FROM` statement starts out by creating a layer from the `ubuntu:22.04` image. The `LABEL` command only modifies the image's metadata, and doesn't produce a new layer. The `COPY` command adds some files from your Docker client's current directory. The first `RUN` command builds your application using the `make` command, and writes the result to a new layer. The second `RUN` command removes a cache directory, and writes the result to a new layer. Finally, the `CMD` instruction specifies what command to run within the container, which only modifies the image's metadata, which doesn't produce an image layer.

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	Source: https://docs.docker.com/engine/storage/drivers/. The major difference between a container and an image is the top writable layer. All writes to the container that add new or modify existing data are stored in this writable layer. When the container is deleted, the writable layer is also deleted. The underlying image remains unchanged. Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image. Source: https://docs.docker.com/engine/storage/drivers/.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 25

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	 Source: https://docs.docker.com/engine/storage/drivers/. On information and belief, the image below shows a thin r/w layer for each of four separate containers that are each stored in memory and are associated with a a root image that is further associated with a compute node.

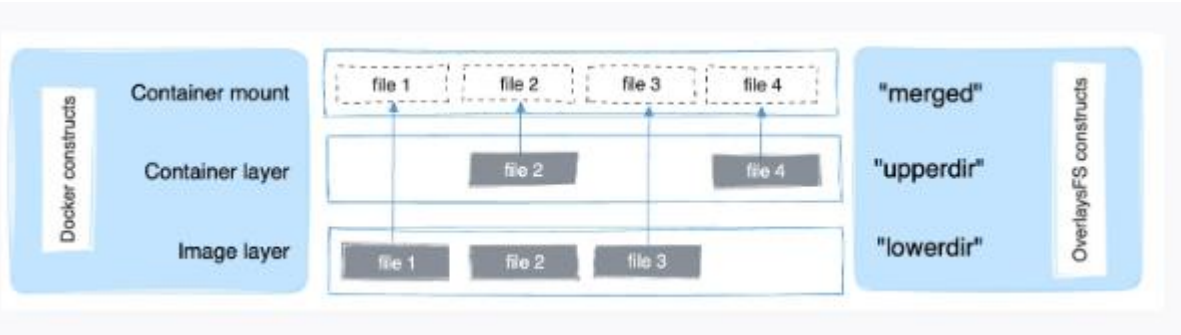
PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 26

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	 Source: https://docs.docker.com/engine/storage/drivers/. *See also* description below of containers. *See* https://docs.docker.com/engine/docker-overview/. On information and belief, a Docker container is a runnable instance of an image and includes configuration options when it is created or started. When an image is run and a container is generated, a writable layer is created (on top of other layer(s)) and includes all changes to the running container.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 27

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	Containers A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state. By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine. A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that aren't stored in persistent storage disappear. Source: https://docs.docker.com/get-started/docker-overview/.

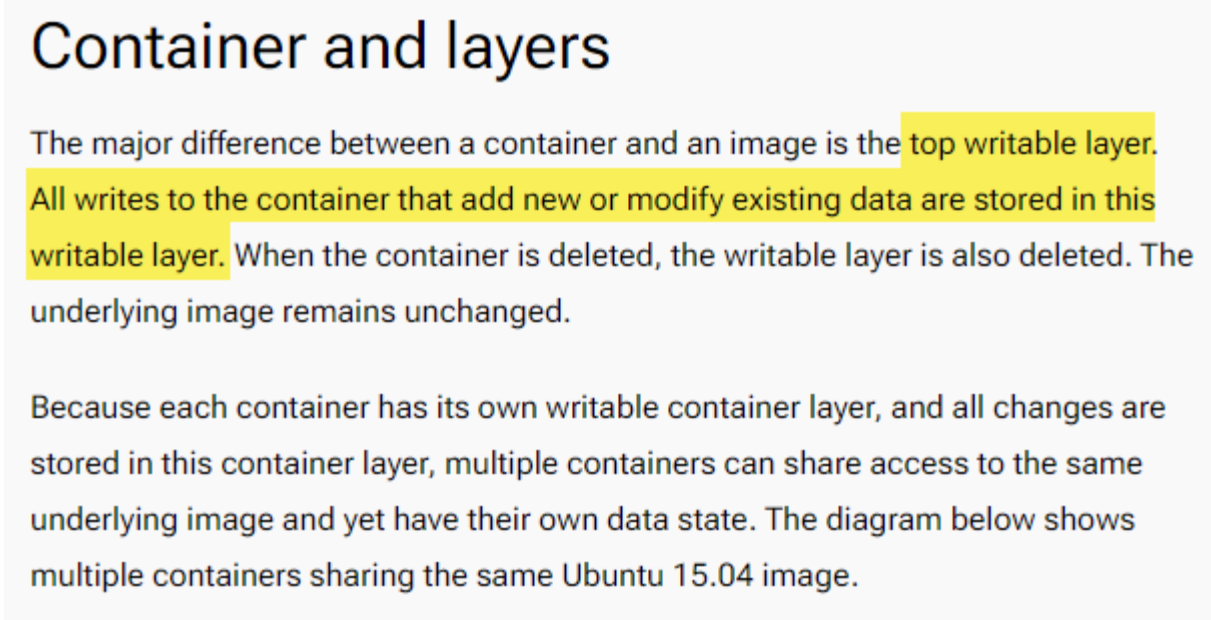
PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 28

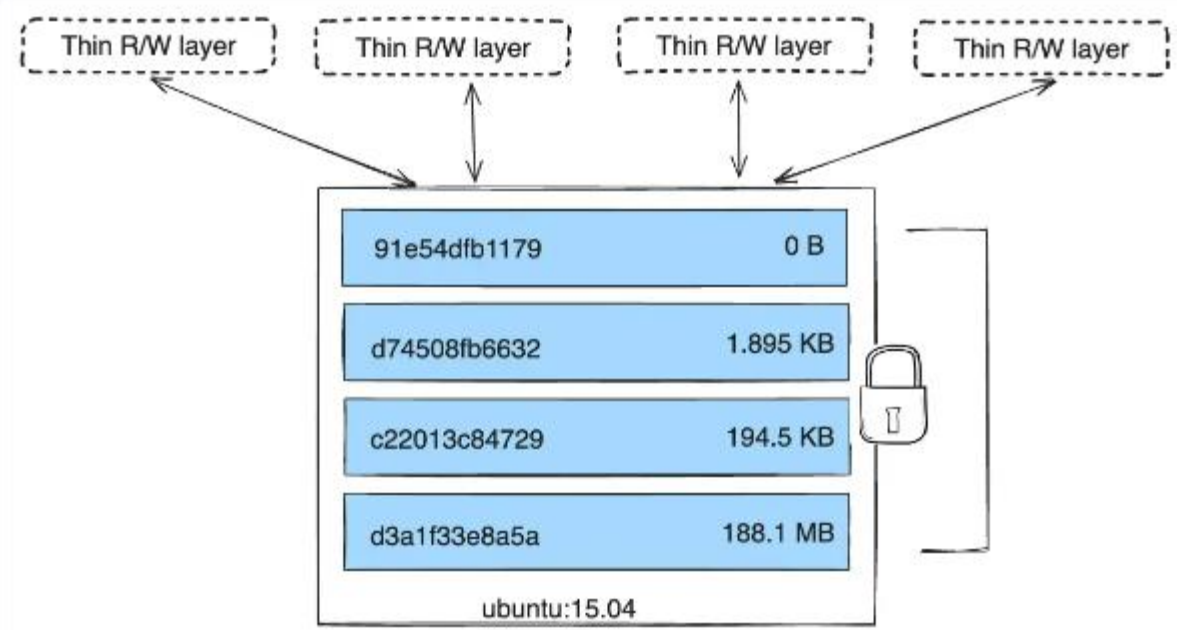
U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	Docker uses storage drivers to store image layers, and to store data in the writable layer of a container. The container's writable layer doesn't persist after the container is deleted, but is suitable for storing ephemeral data that is generated at runtime. Storage drivers are optimized for space efficiency, but (depending on the storage driver) write speeds are lower than native file system performance, especially for storage drivers that use a copy-on-write filesystem. Write-intensive applications, such as database storage, are impacted by a performance overhead, particularly if pre-existing data exists in the read-only layer. On information and belief, a Docker image and a Docker container can be layered in a manner as shown in the image below.  Source: https://docs.docker.com/engine/storage/drivers/overlayfs-driver/.
[7.b.i] said leaf images including only additional data blocks not previously contained in said root image and changes made by	On information and belief, the Southwest Count 1 Systems and Services practice storing leaf images, where said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image.

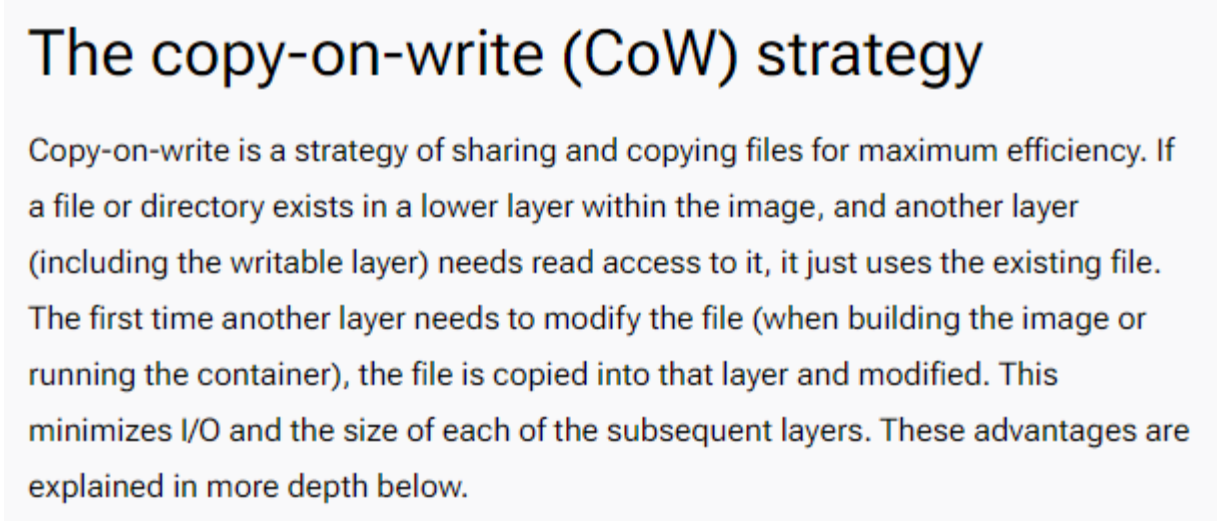
PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 29

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
respective compute nodes to the blocks of the root image,	For example, on information and belief, Docker includes a writable runnable container layer for new data such as new files and including a copy on write strategy for changes made to the blocks of the root image. The layers are stacked on top of each other. When you create a new container, you add a new writable layer on top of the underlying layers. This layer is often called the "container layer". 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. The diagram below shows a container based on an `ubuntu:15.04` image. Source: https://docs.docker.com/engine/storage/drivers/.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 30

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	 Source: https://docs.docker.com/engine/storage/drivers/.

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	 Source: https://docs.docker.com/engine/storage/drivers/. Docker uses storage drivers to manage the contents of the image layers and the writable container layer. Each storage driver handles the implementation differently, but all drivers use stackable image layers and the copy-on-write (CoW) strategy. Source: https://docs.docker.com/engine/storage/drivers/.

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	 Source: https://docs.docker.com/engine/storage/drivers/. On information and belief, when a container is created or started, a thin writable container layer is added on top of the other layers, and any changes the container makes to the filesystem are stored in this layer, whereas any files that the container does not change do not get copied into this layer. Multiple containers can share access to the same image and maintain their own data state using their container layer.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 33

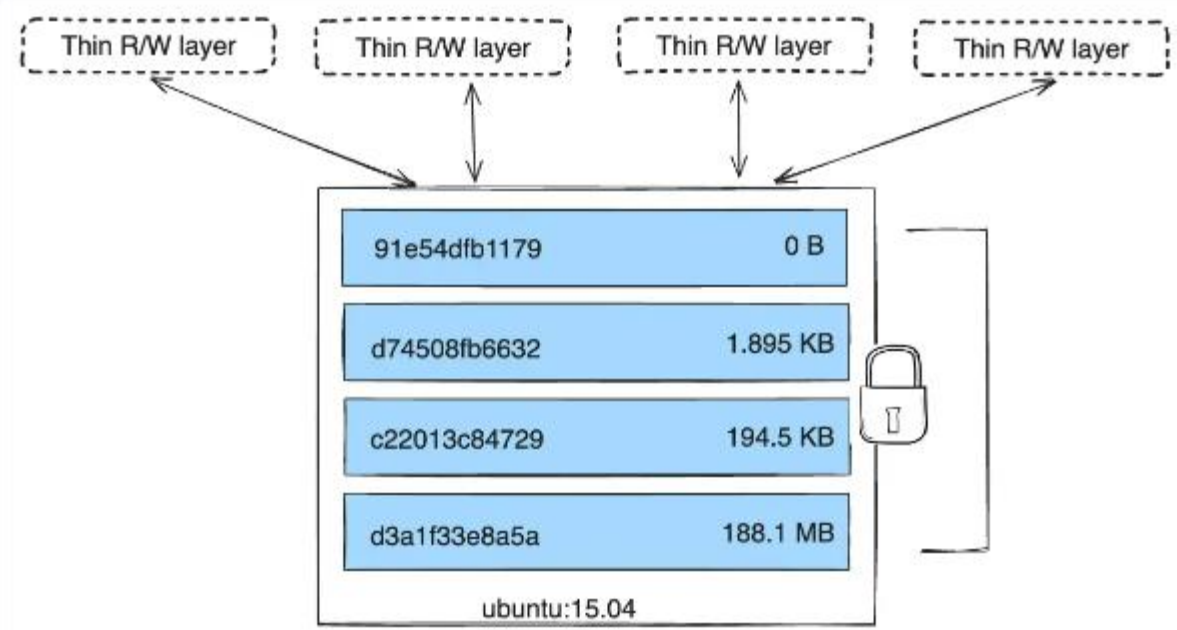
U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	## Copying makes containers efficient When you start a container, a thin writable container layer is added on top of the other layers. Any changes the container makes to the filesystem are stored here. Any files the container doesn't change don't get copied to this writable layer. This means that the writable layer is as small as possible. Source: https://docs.docker.com/engine/storage/drivers/. Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image. Source: https://docs.docker.com/engine/storage/drivers/. *See also* claim limitations [7.a] and [7.b].

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
[7.b.ii] wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	On information and belief, the Southwest Count 1 Systems and Services practice said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes. For example, on information and belief, Docker contains a writable layer using copy on write, leaving unchanged image data in their original image layers. Docker uses storage drivers to manage the contents of the image layers and the writable container layer. Each storage driver handles the implementation differently, but all drivers use stackable image layers and the copy-on-write (CoW) strategy. Source: https://docs.docker.com/engine/storage/drivers/. The copy-on-write (CoW) strategy Copy-on-write is a strategy of sharing and copying files for maximum efficiency. If a file or directory exists in a lower layer within the image, and another layer (including the writable layer) needs read access to it, it just uses the existing file. The first time another layer needs to modify the file (when building the image or running the container), the file is copied into that layer and modified. This minimizes I/O and the size of each of the subsequent layers. These advantages are explained in more depth below. Source: https://docs.docker.com/engine/storage/drivers/.

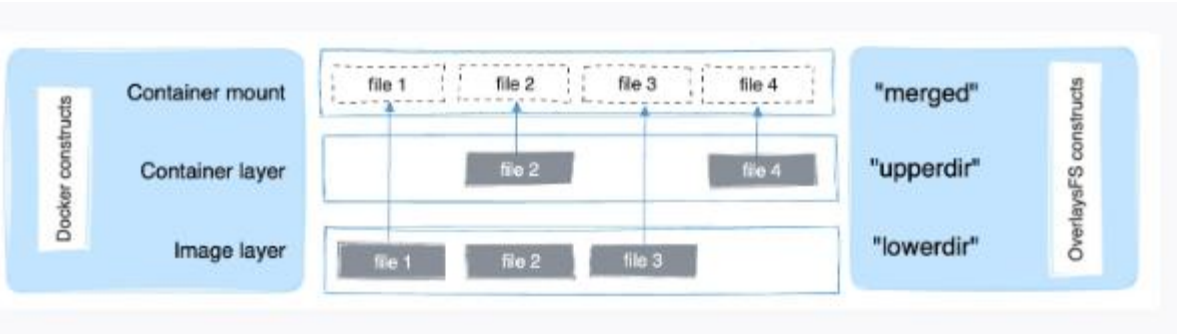
PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 35

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	On information and belief, when a container is created or started, a thin writable container layer is added on top of the other layers, and any changes the container makes to the filesystem are stored in this layer, whereas any files that the container does not change do not get copied into this layer. Multiple containers can share access to the same image and maintain their own data state using their container layer. Container and layers The major difference between a container and an image is the top writable layer. All writes to the container that add new or modify existing data are stored in this writable layer. When the container is deleted, the writable layer is also deleted. The underlying image remains unchanged. Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image. Source: https://docs.docker.com/engine/storage/drivers/.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 36

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	 Source: https://docs.docker.com/engine/storage/drivers/. **Copying makes containers efficient** When you start a container, a thin writable container layer is added on top of the other layers. Any changes the container makes to the filesystem are stored here. Any files the container doesn't change don't get copied to this writable layer. This means that the writable layer is as small as possible. Source: https://docs.docker.com/engine/storage/drivers/.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 37

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image. Source: https://docs.docker.com/engine/storage/drivers/. On information and belief, in the example below, the image layer includes file 1, file 2, and file 3, and the container layer includes file 2 and file 4. The container mount provides a unified view of the image and container layers. In this example, the container layer does not include file 1 or file 3, and includes file 2 and file 4, which have either been modified or created.  Source: https://docs.docker.com/engine/storage/drivers/overlayfs-driver/. *See also* claim limitations [7.a], [7.b], and [7.b.i].
[7.c] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	On information and belief, the Southwest Count 1 Systems and Services practice caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory. Docker caches unchanged image layers from previous commands.

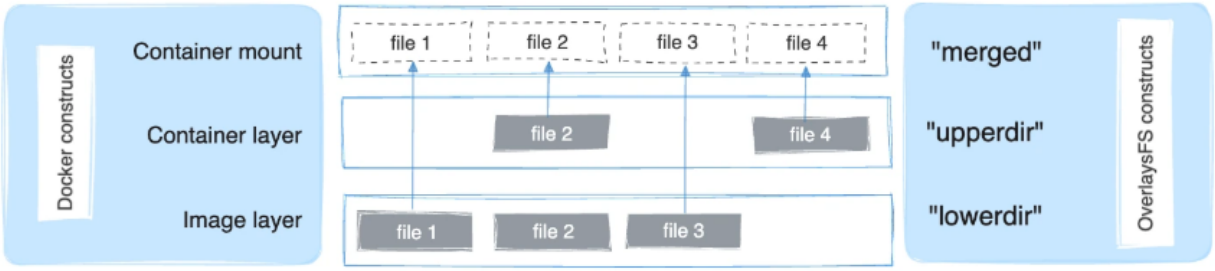
PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 38

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	# Optimize cache usage in builds When building with Docker, a layer is reused from the build cache if the instruction and the files it depends on hasn't changed since it was previously built. Reusing layers from the cache speeds up the build process because Docker doesn't have to rebuild the layer again. Source: https://docs.docker.com/build/cache/optimize/. ## Understanding Docker Cache Before we dive into the cleaning process, it's important to understand how Docker cache works. When you build a Docker image, Docker uses a build cache to speed up the build process. The build cache stores intermediate layers of the image, which are the layers that don't change frequently. This allows Docker to reuse these layers when building a new image, saving time and resources. Source: https://collabnix.com/how-to-clear-docker-cache/.

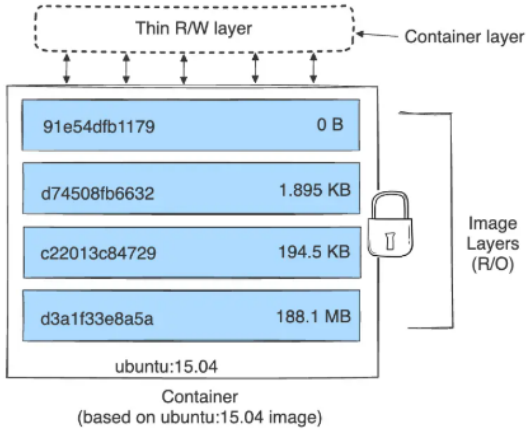
PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 39

U.S. Patent No. 8,332,844 (Claim 7)	
Claim 7	**Example Southwest Count 1 Systems and Services**
	At each occurrence of a RUN command in the Dockerfile, Docker will create and commit a new layer to the image, which is just a set of tightly-coupled directories full of various file structure that comprise a Docker image. In a default install, these are located in /var/lib/docker. During a new build, all of these file structures have to be created and written to disk — this is where Docker stores base images. Once created, the container (and subsequent new ones) will be stored in the folder in this same area. What makes the cache important? If the objects on the file system that Docker is about to produce are unchanged between builds, reusing a cache of a previous build on the host is a great time-saver. It makes building a new container really, really fast. None of those file structures have to be created and written to disk this time — the reference to them is sufficient to locate and reuse the previously built structures. Source: https://thenewstack.io/understanding-the-docker-cache-for-faster-builds/.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 40

U.S. Patent No. 8,332,844 (Claim 11)	
Claim 11	**Example Southwest Count 1 Systems and Services**
11] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	On information and belief, the Southwest Count 1 Systems and Services practice the method of claim 7. *See* claim 7. On information and belief, the Southwest Count 1 Systems and Services further practice merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments. For example, on information and belief, Docker uses each of an overlay feature, merge feature, and add a new layer feature that facilitates performance of this limitation. The following diagram shows how a Docker image and a Docker container are layered. The image layer is the `lowerdir` and the container layer is the `upperdir`. If the image has multiple layers, multiple `lowerdir` directories are used. The unified view is exposed through a directory called `merged` which is effectively the containers mount point.  Source: https://docs.docker.com/engine/storage/drivers/overlayfs-driver/. Where the image layer and the container layer contain the same files, the container layer (`upperdir`) takes precedence and obscures the existence of the same files in the image layer. To create a container, the `overlay2` driver combines the directory representing the image's top layer plus a new directory for the container. The image's layers are the `lowerdirs` in the overlay and are read-only. The new directory for the container is the `upperdir` and is writable. Source: https://docs.docker.com/engine/storage/drivers/overlayfs-driver/.

PETITIONERS - AA/SW EXHIBIT 1028
American Airlines, et al. v. Intellectual Ventures II
IPR2025-00931, Page 41

U.S. Patent No. 8,332,844 (Claim 11)	
Claim 11	**Example Southwest Count 1 Systems and Services**
	The layers are stacked on top of each other. When you create a new container, you add a new writable layer on top of the underlying layers. This layer is often called the "container layer". 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. The diagram below shows a container based on an `ubuntu:15.04` image.  Source: https://docs.docker.com/engine/storage/drivers/. Containers A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state. By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine. A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that aren't stored in persistent storage disappear. Source: https://docs.docker.com/get-started/docker-overview/.