**Docker কি?**

[ডেভ অপসের](https://w3programmers.com/bangla/devops-%e0%a6%95%e0%a6%bf/) এবং [Micro Service Development](https://aws.amazon.com/microservices/) এ খুব গুরুত্বপূর্ণ টুল হল Docker. [ডকার](https://www.docker.com/" \t "_blank)কি এবং ডকার কিভাবে কাজ করে, সেটি আলোচনার পূর্বে আমি আমাদের বাস্তব জীবনের কিছু বিষয় নিয়ে আলোচনা করব।



**ডকার**

আমরা আমাদের ডেভেলপ করা এপ্লিকেশন কে যখন অন্য কোনো সার্ভারে রান করতে যাই , তখন আমরা প্রায়শই একটি কথা শুনি বা বলি : “**আমার পিসি তে তো সব ঠিক মত চলল বা আমার পি সি তে তো সব ঠিকই ছিল**” তার মানে আপনি নতুন যে জায়গায় আপনার এপ্লিকেশন টি রান করতে চাচ্ছেন সেটির এবং আপনি আপনার যে পিসি তে এপ্লিকেশন টি ডেভেলপ করেছেন তার কনফিগারেশন এবং এনভায়রনমেন্ট এক না।

এই বিষয়টা যে শুধুমাত্র এপ্লিকেশন ডেভেলপমেন্ট এর বেলায় ঘটে শুধু তাই নয়। বরং আমাদের বাস্তব জীবনেও আপনি দেখবেন , আমরা যেখানে বড় হই, যখন আমরা নতুন কোথাও যাই , তখন সেখানকার এনভায়রনমেন্ট এবং কালচার আমাদের জন্য নতুন হওয়ায় আমাদের জন্য নতুন জায়গার সাথে খাপ খাওয়ানো একটু কঠিনই হয়।

কেমন হতো, আমরা যে এনভায়রনমেন্ট এবং যে কালচারে বড় হয়েছি , সেখান থেকে নতুন জায়গায় যাওয়ার সময় যদি আমাদের নিজেদের বেড়ে উঠার এনভাইরোনমনেট এবং কালচার ও সাথে নিয়ে যেতে পারতাম?

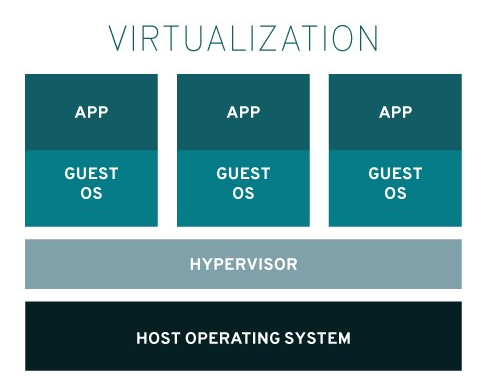
আমাদের বাস্তব জীবনে এই ব্যাপার গুলো অনেক কঠিন হলেও ওয়েব এপ্লিকেশন ডেভেলপমেন্ট, ট্রান্সফার এবং ডেপ্লয় এর বেলায় Docker এবং Kubernet এর মতো কন্টেইনার নির্ভর ভার্চুয়ালাইজেশন প্রযুক্তি গুলো আমাদেরকে এই ধরণের কাজ অনেক সহজ করে দিয়েছে।

ডকার কি সেটা বুঝার আগে , প্রথমে আমাদেরকে বুঝতে হবে Virtualization এবং Container কি?

**Virtualization কি?**

ধরুন আপনার একটা Hosting ব্যবসা আছে। এখন আপনার ৩ জন ক্লায়েন্ট আপনার কাছে ৩ টি হোস্টিং চেয়েছে। যাদের প্রত্যেকের 3 Core Processor এবং 8 GB করে RAM লাগবে। এখন যদি আপনার একটি 24 GB RAM এবং 9 Core Processor থেকে থাকে , তাহলে অনায়াসেই VM Ware, Virtual Box অথবা Microsoft এর Hyper V এই তিনটির যে কোনো একটির মাধ্যমে , আপনি আপনার কম্পিউটার টিকে Hardware Level এ Virtually ভাগ করে দিতে পারেন। এতে আপনার ৩ ক্লায়েন্ট তাদের প্রত্যেকে নিজেদের ভাগের হার্ডওয়্যার কে একটি স্বয়ং সম্পূর্ণ কম্পিউটার হিসেবে ব্যবহার করতে পারবে। এবং নিজেদের ইচ্ছামতো যেকোনো সফটওয়্যার ইনস্টল-আনইনস্টল এবং রান করতে পারবে। আর এই ব্যাপারটিকে বলা হয় Virtualization বা Hardware Level Virtualization.

আরো ভালো ভাবে বুঝার জন্য নিচের ছবিটির দিখে লক্ষ্য করুন :



Computer Virtualization

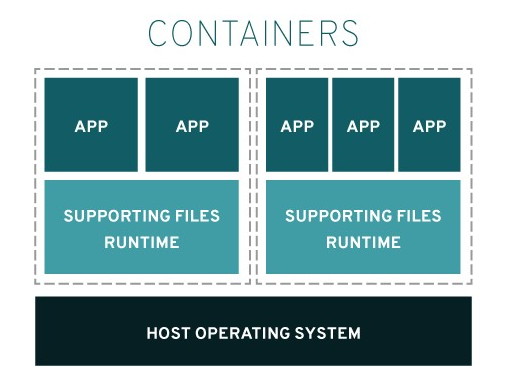
এইটা তো গেলো একটা পি সি কে আপনি Hardware Level Virtualization এর মাধ্যমে ভাগ করা। এখন ধরুন আপনার এইরকম একটি ভার্চুয়াল বা ফিজিক্যাল পি সি আছে। সেখানে আপনি NodeJS সফটওয়্যার ব্যবহার হয়েছে এই রকম তিনটি সফটওয়্যার ইনস্টল করতে চান। যেখানে তিনটি সফটওয়্যার এ যথাক্রমে NodeJS এর ভার্সন ১৪, ১৫ এবং ১৬ ব্যবহৃত হয়েছে। অথবা PHP Laravel Framework ব্যবহার হয়েছে এই রকম তিনটি সফটওয়্যার ইনস্টল করতে চান। যেখানে তিনটি সফটওয়্যার এ যথাক্রমে Laravel এর ভার্সন 5, 7 এবং 9 ব্যবহৃত হয়েছে। যেগুলো এই মুহূর্তে আপডেট করা খুবই সময় এবং ব্যয় সাপেক্ষ। এক্ষেত্রে আপনি বলতে পারেন VM Ware, Virtual Box অথবা Microsoft এর Hyper V ব্যবহার করে তিনটি Virtual Machine করে নিলেই হয়? আসলে শুধু সফটওয়্যার এর ভার্সন সমস্যা সমাধানের জন্য নতুন নতুন ভার্চুয়াল বা ফিজিক্যাল পিসি ব্যবহার করাও বুদ্ধিদীপ্ত নয়। কারণ এতে আপনার সফটওয়্যার পরিচালনা খরচ অনেক বেড়ে যাবে। এছাড়া এইরকম ভার্সন গত সমস্যা আপনার জীবনে শত সহস্র বার আসতেই পারে। তাই বলে কি আপনি শত সহস্র পি সি বিল্ড করবেন?

আর এই সমস্যাটি সমাধান করা যায় , যদি আমরা আমাদের Operating System এর Kernel কে Virtually ভাগ করতে পারি এবং সিস্টেমের একটি অংশের অ্যাপ্লিকেশন প্রক্রিয়াগুলি থেকে অন্য অংশের অ্যাপ্লিকেশন প্রক্রিয়াগুলিকে বিচ্ছিন্ন রাখতে পারি।

**Container কি?**

তাহলে আমরা বলতে পারি , Container হলো এমন একটি Tool বা API যা একটি Operating System এর Kernel কে Virtually ভাগ করে এবং সিস্টেমের একটি অংশের অ্যাপ্লিকেশন প্রক্রিয়াগুলি থেকে অন্য অংশের অ্যাপ্লিকেশন প্রক্রিয়াগুলিকে বিচ্ছিন্ন রাখতে পারে। আর লিনাক্স অপারেটিং সিস্টেমে এই কাজটি করা হয়[LXC API](https://linuxcontainers.org/lxc/introduction/" \t "_blank) এর মাধ্যমে। যাকে বলা হয় Linux Container. আর এই ব্যাপারটিকে বলা হয় Application/OS Level Virtualization.

আরো ভালো ভাবে বুঝার জন্য নিচের ছবিটির দিখে লক্ষ্য করুন :



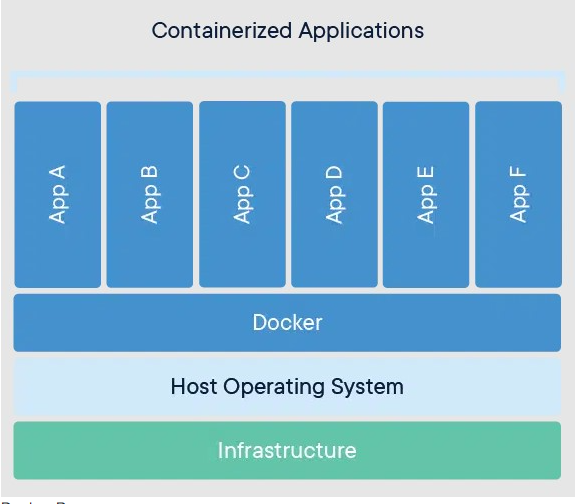
Linux Container

**Docker কি?**

ডকার হল একধরনের Container Engine বা এপ্লিকেশন সিস্টেম যা একধরণের কন্টেইনার নির্ভর ভার্চুয়ালাইজেশন এর সুবিধা দিয়ে থাকে। যার কাজ হল ডেভেলপমেন্ট এর জন্য আপনার Application Layer কে System Layer থেকে আলাদা করা এবং প্রয়োজনীয় সব ধরণের ডিপেন্ডেন্সিস দিয়ে সাহায্য করা। । এটা একটা Cross Platform অর্থাৎ সব ধরণের অপারেটিং সিস্টেমে চলে এবং একটি Open Source Tool, যা ডেভেলপারদেরকে কনটেইনার তৈরি, ডেপ্লয়, রান , আপডেট এবং ম্যানেজ করতে সুযোগ দেয় এবং প্রয়োজনীয় সব ধরণের টেকনোলজি, টুলস, কনফিগারেশন এবং এনভায়রনমেন্টাল সাপোর্ট প্রদান করে।

উদাহরণস্বরূপ, আপনার যদি একটি জাভা jar ফাইল থাকে তবে আপনি জাভা ইনস্টল করা যেকোনো সার্ভারে এটি চালাতে পারেন। একইভাবে, একবার আপনি ডকার ব্যবহার করে প্রয়োজনীয় অ্যাপ্লিকেশন সহ একটি container package তৈরি করলে, আপনি ডকার ইনস্টল করা অন্য যে কোনও হোস্টে এটি চালাতে পারেন।

আরো ভালো ভাবে বুঝার জন্য নিচের ছবিটির দিখে লক্ষ্য করুন :



Docker Process

**Docker এবং Container এর মধ্যে পার্থক্য**

Docker একটি technology বা tool যা দক্ষতার সাথে অপারেটিং সিস্টেমের container গুলো রান করার জন্য তৈরি করা হয়েছে।

সুতরাং, আমি কি ডকার ছাড়া একটি কন্টেইনার চালাতে পারি?

হ্যাঁ! অবশ্যই. আপনি Linux সার্ভারে কন্টেইনার গুলো চালানোর জন্য LXC technology ব্যবহার করতে পারেন। এছাড়াও, [Podman](https://podman.io/" \t "_blank)এর মতো সর্বশেষ tool গুলি ডকারের মতো একই ধরণের ওয়ার্কফ্লো অফার করে।

তবে Docker সম্পর্কে আপনার নিম্নোক্ত বিষয়গুলো জানা থাকা উচিত:

1. ডকার LXC নয়
2. ডকার একটি Virtual Machine Solution নয়।
3. ডকার কোনো configuration management system নয় এবং Chef, Puppet, Ansible ইত্যাদির বিকল্প নয়।
4. ডকার একটি platform as a service technology নয়।
5. ডকার কোনো container নয়।

**কিসে ডকার কে এত দুর্দান্ত করে তুললো ?**

ডকারের রয়েছে এমন একটি efficient workflow, যা দিয়ে আপনি যেকোনো অ্যাপ্লিকেশনকে ডেভেলপারের ল্যাপটপ থেকে test environment এ, এবং test environment থেকে production খুব সহজে নিয়ে যেতে পারেন। আর এই ব্যাপারটি আপনি তখনি বুঝতে পারবেন যখন আপনি একটি Docker image কে একটি অ্যাপ্লিকেশন প্যাকেজ করার একটি practical example দেখবেন।

আপনি কি জানেন যে একটি docker container শুরু করতে **এক সেকেন্ডেরও কম সময়** লাগে?

এটি অবিশ্বাস্যভাবে দ্রুত, এবং এটি লিনাক্স কার্নেলের সাথে কমপ্যাটিবল হওয়ায় এটি যেকোনো হোস্টে চলতে পারে। (একই সাথে এটি উইন্ডোজকে সাপোর্ট করে)

ডকার তার ইমেজ স্টোরেজের জন্য একটি [Copy-on-write union file system](https://en.wikipedia.org/wiki/UnionFS) ব্যবহার করে। অতএব, যখন একটি container পরিবর্তন করা হয়, শুধুমাত্র চেঞ্জেস গুলি write মডেলের কপি ব্যবহার করে ডিস্কে লেখা হবে।

[Copy on write](https://en.wikipedia.org/wiki/Copy-on-write) এর সাথে, আপনার সমস্ত কন্টেইনারের জন্য অপ্টিমাইজ করা শেয়ার্ড স্টোরেজ লেয়ার থাকবে।

**Docker Core Architecture**

ডকার তৈরি হওয়ার পর থেকে তার আর্কিটেকচারটি কয়েকবার পরিবর্তিত হয়েছে। নিম্নে ডকারের কিছু উল্লেখযোগ্য architectural পরিবর্তন নিয়ে আলোচনা করা হলো :

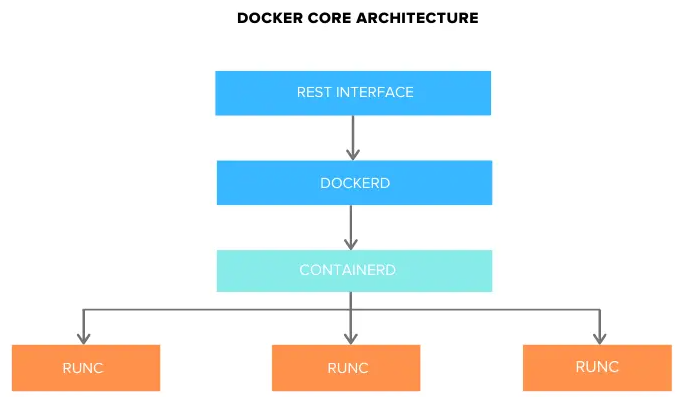
* ডকার প্রথমে 2014 সালে LXC থেকে libcontainer এ মুভ করে।
* [runc](https://github.com/opencontainers/runc)– এটি একটি CLI যা সমস্ত [OCI](https://opencontainers.org/) specification গুলোকে অনুসরণ করে এমন কন্টেইনার গুলোকে spinning করার জন্য ব্যবহৃত হয়।
* containerd – ডকার 2016 সালে তার container management component কে [containerd](https://github.com/containerd/containerd" \t "_blank)এ আলাদা করেছে

ডকার যখন প্রাথমিকভাবে চালু করা হয়েছিল, তখন এটির ছিল monolithic architecture. এখন এটি নিম্নলিখিত তিনটি ভিন্ন component বিভক্ত:

* Docker Engine (dockerd)
* docker-containerd (containerd)
* docker-runc (runc)

ডকার এবং অন্যান্য বড় অর্গানাইজেশন গুলি একটি standard container runtime এবং management layer গুলি তৈরিতে কন্ট্রিবিউট করেছিল। তাই এখন সব অর্গানাইজেশন গুলো এবং তাদের কান্ট্রিবিউটররা **containerd** এবং **runc** এখন Cloud Native Foundation এর অংশ ।

নিচের ছবিতে Docker Core Architecture দেওয়া হলো:



Docker Core Architecture

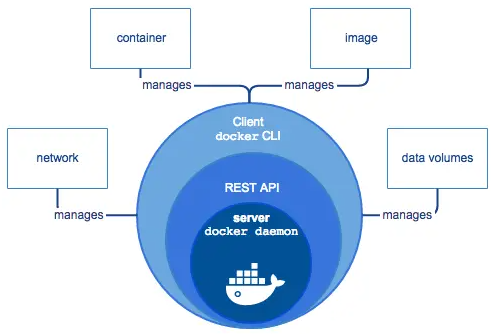
এবার প্রতিটি ডকার Component নিয়ে আলোচনা করা যাক:

**Docker Engine**

Docker Engine মূলতঃ docker daemon, একটি API interface এবং Docker CLI রয়েছে। docker daemon (dockerd) এটি মূলতঃ dockerd systemd service হিসাবে নিরবিচ্ছিন্নভাবে চলে। এটি ডকার image গুলো তৈরির দায়িত্বে থাকে।

Image গুলো ম্যানেজ করতে এবং container গুলো চালানোর জন্য, dockerd docker-containerd API গুলিকে কল করে।

নিচের ছবিটি লক্ষ্য করুন :



Docker Engine Components Flow

**docker-containerd (containerd)**

containerd হল আরেকটি system daemon service যা ডকার ইমেজগুলি ডাউনলোড করার এবং একটি container হিসাবে চালানোর দায়িত্বে থাকে। এটি dockerd service থেকে নির্দেশাবলী পাওয়ার জন্য এর API প্রদান করে।

**docker-runc**

runc হল একটি container যা রানটাইমে একটি কন্টেইনারের জন্য প্রয়োজনীয় namespaces গুলো এবং cgroups তৈরি করার কাজটি করে থাকে। এটি তারপর সেই namespaces গুলোর ভিতরে container command গুলো চালায়। runc রানটাইম OCI স্পেসিফিকেশন অনুযায়ী প্রয়োগ করা হয়।

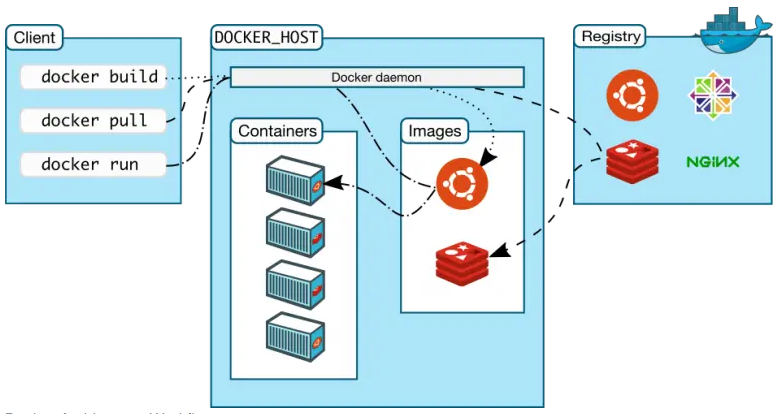
**ডকার কিভাবে কাজ করে?**

আমরা ডকারের core building block গুলি দেখেছি।

এখন ডকার component গুলি ব্যবহার করে Docker workflow বুঝার চেষ্টা করব।

**Docker Components**

নিম্নলিখিত অফিসিয়াল high-level docker architecture diagram টি common Docker workflow দেখায়।

Docker Architecture Workflow

ডকার ইকোসিস্টেম নিম্নলিখিত পাঁচটি components নিয়ে গঠিত

* Docker Daemon (dockerd)
* Docker Client
* Docker Images
* Docker Registries
* Docker Containers

এবার উপরের পাঁচটি Components এর প্রত্যেকটি নিয়ে আলোচনা করা যাক :

**Docker Daemon কি?**

ডকারের রয়েছে একটি ক্লায়েন্ট-সার্ভার আর্কিটেকচার । Docker Daemon (dockerd) বা server যা containers গুলো সম্পর্কিত সমস্ত কাজের দায়িত্বে থাকে।

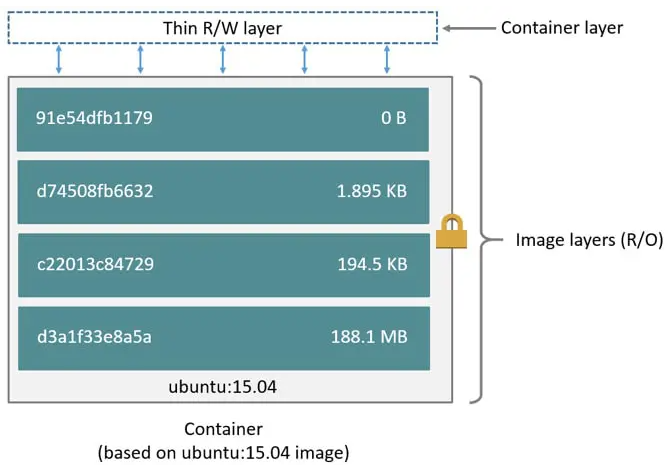
daemon এর কাজ হচ্ছে CLI বা REST API এর মাধ্যমে ডকার ক্লায়েন্ট থেকে কমান্ড গ্রহণ করে। ডকার ক্লায়েন্ট daemon এর মতো একই হোস্টে থাকতে পারে বা অন্য কোনও হোস্টে উপস্থিত থাকতে পারে।

ডিফল্টরূপে, docker daemon docker.sock UNIX সকেট এর মাধ্যমে কাজ করে । আপনার যদি দূরবর্তীভাবে ডকার API অ্যাক্সেস করার জন্য কোনও ব্যবহারের ক্ষেত্রে থাকে তবে আপনাকে এটি একটি হোস্ট পোর্টের মাধ্যমে প্রকাশ করতে হবে।

**Docker Image কি?**

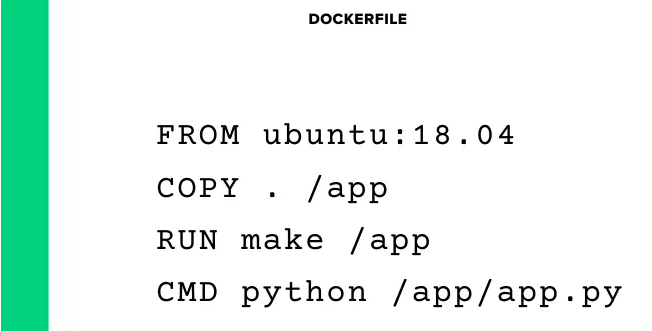
Images গুলি ডকারের মৌলিক বিল্ডিং ব্লক। এটিতে একটি অ্যাপ্লিকেশন চালানোর জন্য OS library গুলো , dependency গুলো এবং tools গুলো রয়েছে।

container গুলো তৈরির জন্য application dependenc গুলো সহ Images গুলি prebuilt (পূর্বনির্মাণ) করা যেতে পারে। উদাহরণস্বরূপ, আপনি যদি একটি Ubuntu container হিসাবে একটি Nginx ওয়েব সার্ভার চালাতে চান তবে আপনাকে Nginx বাইনারি এবং Nginx চালানোর জন্য প্রয়োজনীয় সমস্ত OS লাইব্রেরি সহ একটি ডকার Image তৈরি করতে হবে।



Dockerfile ডকারের image তৈরির জন্য ব্যবহৃত হয়। একটি Dockerfile একটি text file যাতে প্রতি লাইনে একটি command (instructions) থাকে।

নিম্নে একটি ডকারফাইলের উদাহরণ দেওয়া হলো।

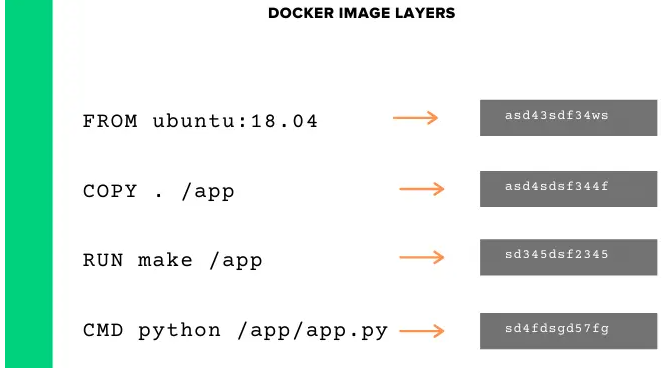
dockerfile example

একটি docker ডকার image একটি layered ফ্যাশনে অর্গানাইজড হয়। একটি Dockerfile এর প্রতিটি instruction একটি image একটি একটি layer যুক্ত করা হয়। image এর সবার উপরে লেখার writable layer টি একটি container।

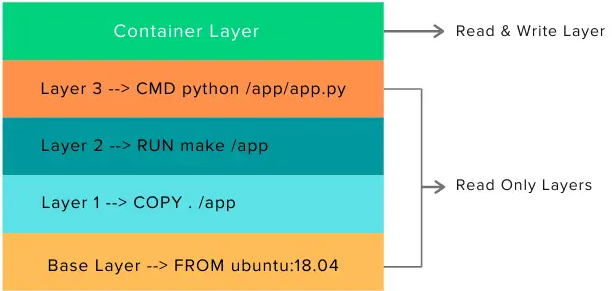
প্রতিটি image একটি base image থেকে তৈরি করা হয়।

উদাহরণস্বরূপ, যদি আপনি উবুন্টুর একটি বেস ইমেজ ব্যবহার করতে পারেন এবং Nginx অ্যাপ্লিকেশনের সাথে অন্য একটি ইমেজ তৈরি করতে পারেন। একটি বেস ইমেজ একটি প্যারেন্ট ইমেজ বা একটি প্যারেন্ট ইমেজ থেকে নির্মিত একটি ইমেজ হতে পারে।

আপনি জিজ্ঞাসা করতে পারেন এই বেস ইমেজ (প্যারেন্ট ইমেজ) কোথা থেকে এসেছে? প্রাথমিক প্যারেন্ট বেস ইমেজ তৈরি করতে ডকার ইউটিলিটি আছে। এটি প্রয়োজনীয় OS লাইব্রেরিগুলি নেয় এবং সেগুলিকে একটি বেস ইমেজে তৈরি করে৷ আপনাকে এটি করতে হবে না কারণ আপনি লিনাক্স ডিস্ট্রোসের জন্য অফিসিয়াল বেস ইমেজ পাবেন।

docker image layers

একটি image এর top layer টি যদি writable এবং running container দ্বারা ব্যবহৃত হয়। image এর অন্যান্য layers গুলি তখন read-only থাকে। নিচের ছবিটি লক্ষ্য করুন :

docker image writable

**Docker Registry কি?**

এটি ডকার image গুলির জন্য একটি repository (storage)।

একটি রেজিস্ট্রি public বা private হতে পারে। উদাহরণস্বরূপ, ডকার Inc Docker Hub নামে একটি hosted registry service সরবরাহ করে। এটি আপনাকে একটি central location থেকে image গুলি আপলোড এবং ডাউনলোড করতে দেয়।

নোট: ডিফল্টরূপে, আপনি যখন ডকার ইনস্টল করেন, তখন এটি public Docker hub থেকে image গুলি সন্ধান করে যদি না আপনি ডকার সেটিংসে একটি কাস্টম রেজিস্ট্রি নির্দিষ্ট করেন৷

আপনার repository public হলে অন্যান্য Docker hub ব্যবহারকারীরা আপনার সমস্ত image এ অ্যাক্সেস করতে পারবেন। আপনি ডকার হাবে একটি ব্যক্তিগত রেজিস্ট্রিও তৈরি করতে পারেন।

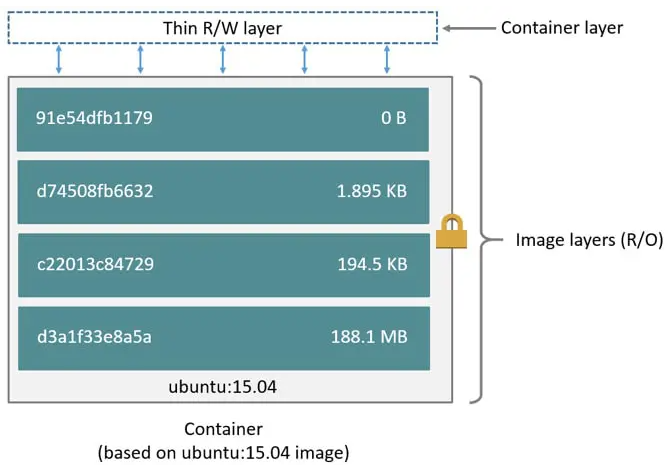
ডকার হাব গিটের মতো কাজ করে, যেখানে আপনি আপনার ল্যাপটপে স্থানীয়ভাবে আপনার image গুলি তৈরি করতে পারেন, এটি কমিট করতে পারেন এবং তারপরে ডকার হাবে push দিতে পারেন।

টিপ: enterprise networks/project এ ডকার ব্যবহার করার সময়, public docker hub ব্যবহার না করে আপনার নিজস্ব docker registry গুলো সেট আপ করুন। সমস্ত cloud provider এর নিজস্ব container registry service গুলো রয়েছে৷

**Docker Container কি?**

Docker Container সমূহ বিদ্যমান image গুলো থেকে তৈরি করা হয়. এটি image এর একটি writable layer.

আপনি যদি image layer গুলো এবং একটি container কে মেলানোর করার চেষ্টা করেন তবে এটি একটি ubuntu-based image এর জন্য কীভাবে দেখায় তা নিম্নে দেখানো হলো :

container layers

আপনি আপনার অ্যাপ্লিকেশনগুলিকে একটি container এ package করতে পারেন, এটিকে commit করতে পারেন এবং এটি থেকে আরও কন্টেইনার তৈরি করতে এটিকে একটি golden image হিসেবে তৈরি করতে পারেন৷

কন্টেইনারগুলি start, stop, commit এবং terminate করা যেতে পারে। আপনি যদি একটি কন্টেইনারকে commit না দিয়ে stop করে দেন, তাহলে সমস্ত কন্টেইনার changes গুলি হারিয়ে যাবে৷

Ideally, container গুলিকে অপরিবর্তনীয় objects হিসাবে বিবেচনা করা হয় এবং running container এ পরিবর্তন করার recommend দেওয়া হয় না। পরিবর্তে, শুধুমাত্র testing purpose এ একটি running container এ পরিবর্তন করুন।