Understanding the Docker Internals

Nitin AGARWAL · Follow
3 min read · Jan 5, 2017

936 3

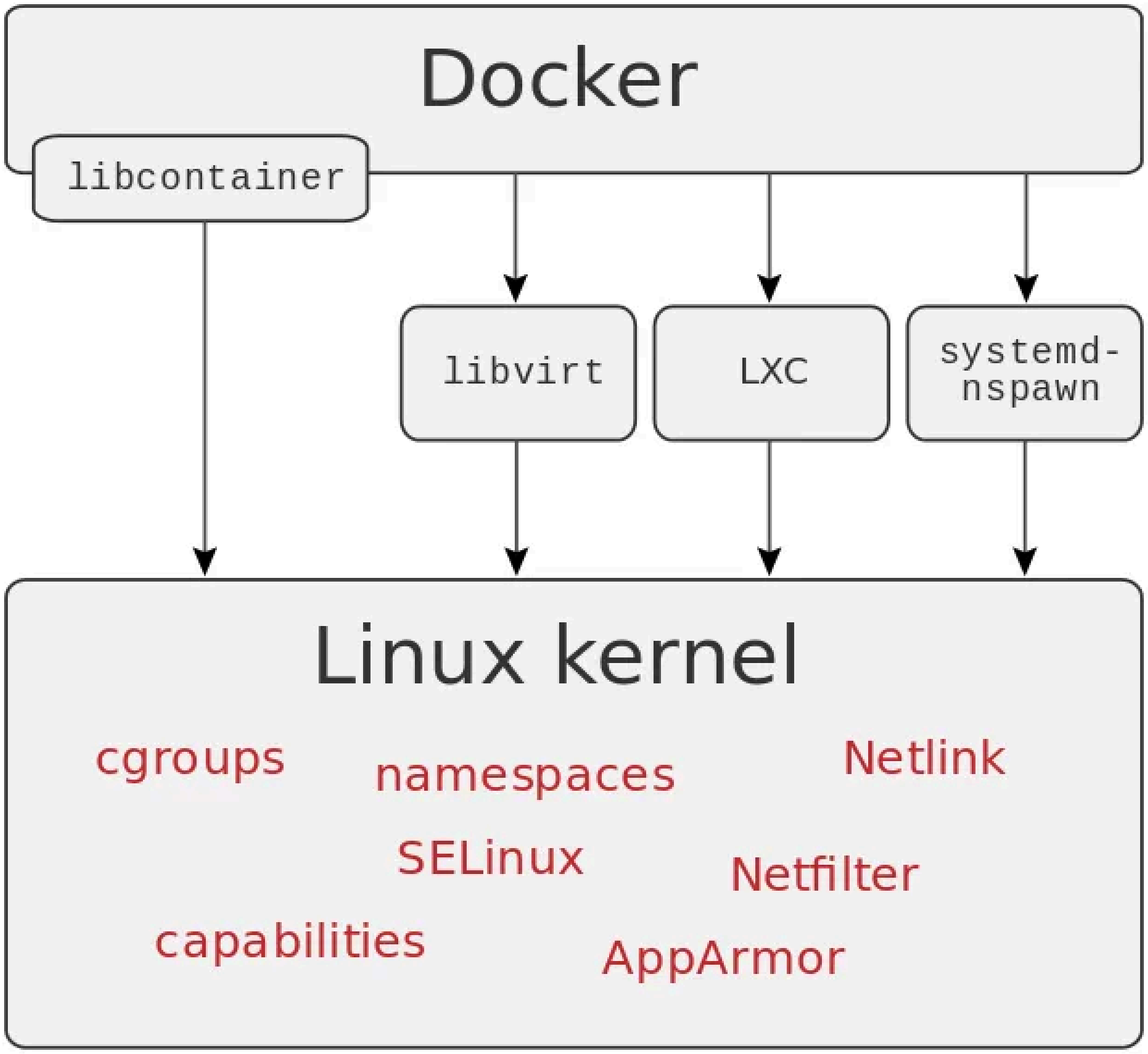


Image Source and Credits: https://delftswa.github.io/chapters/docker/

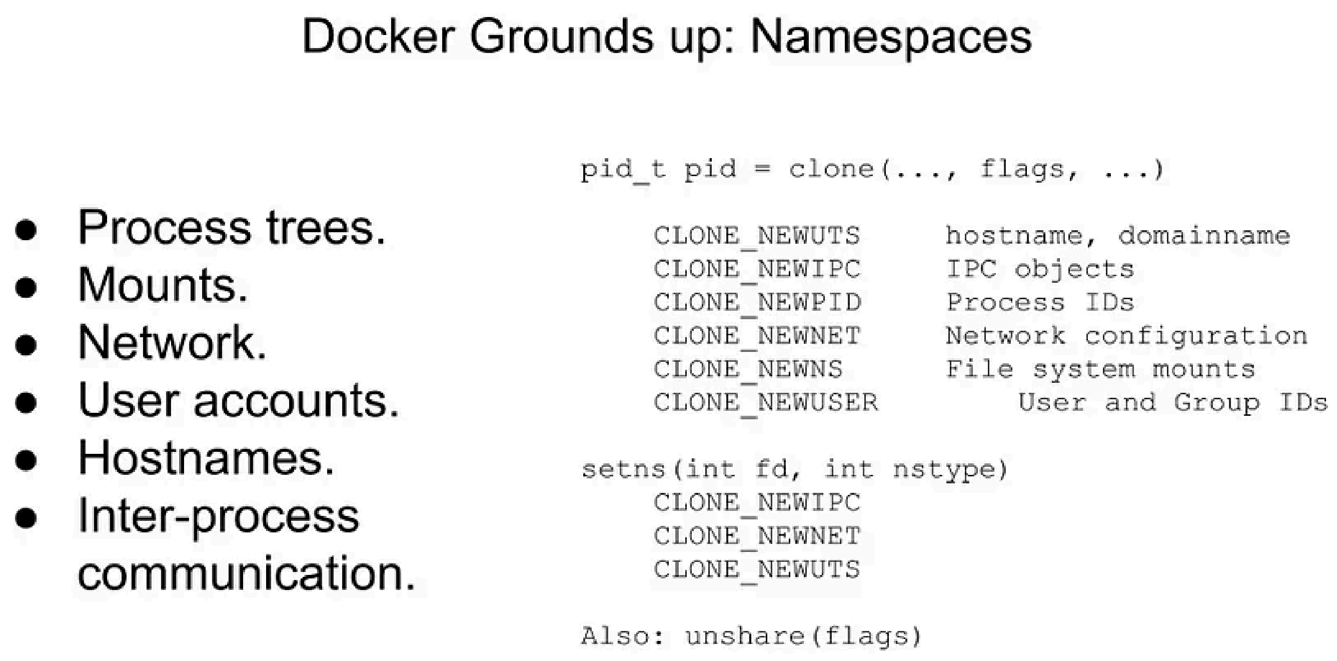
Docker takes advantage of several features of the Linux kernel to deliver its functionality.

Namespaces

Docker makes use of kernel namespaces to provide the isolated workspace called the *container*. When you run a container, Docker creates a set of *namespaces* for that container. These namespaces provide a layer of isolation. Each aspect of a container runs in a separate namespace and its access is limited to that namespace.

Docker Engine uses the following namespaces on Linux:

- **PID namespace** for process isolation.
- **NET namespace** for managing network interfaces.
- **IPC namespace** for managing access to IPC resources.
- **MNT namespace** for managing filesystem mount points.
- **UTS namespace** for isolating kernel and version identifiers.



Docker Grounds up: Namespaces

- Process trees.
- Mounts.
- Network.
- User accounts.
- Hostnames.
- Inter-process communication.

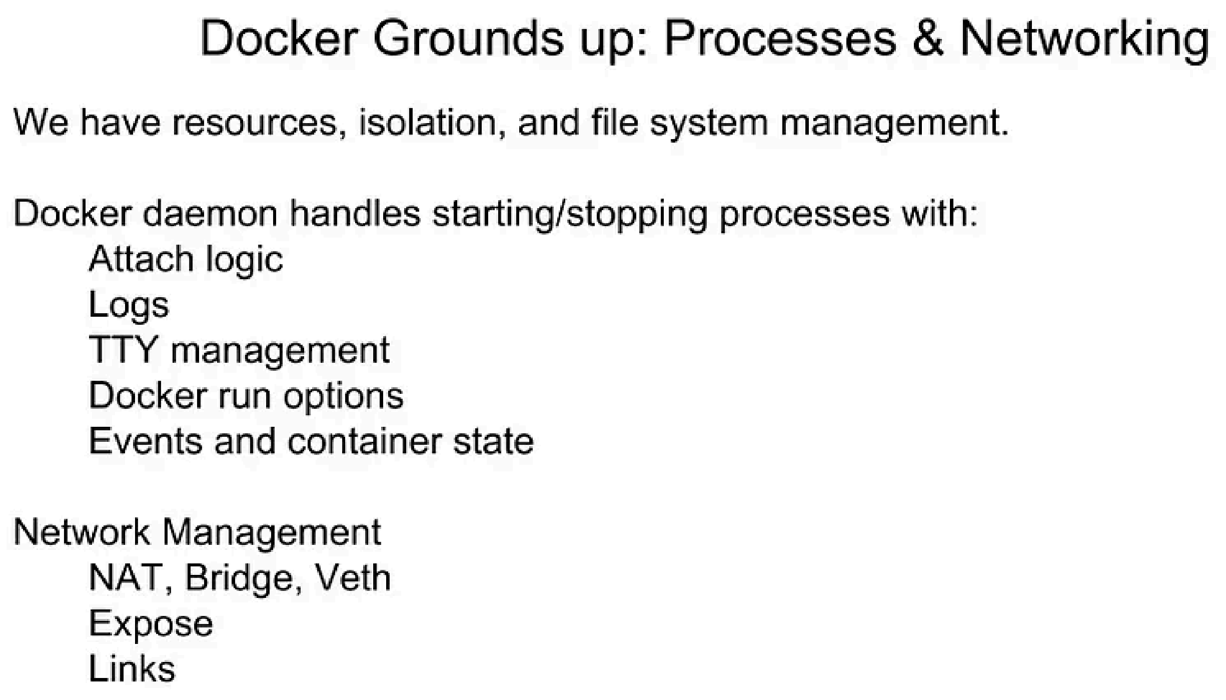
```
pid_t pid = clone(..., flags, ...)

    CLONE_NEWUTS     hostname, domainname
    CLONE_NEWIPC     IPC objects
    CLONE_NEWPID     Process IDs
    CLONE_NEWNET     Network configuration
    CLONE_NEWNS      File system mounts
    CLONE_NEWUSER        User and Group IDs

setns(int fd, int nstype)
    CLONE_NEWIPC
    CLONE_NEWNET
    CLONE_NEWUTS

Also: unshare(flags)
```

Image Source and Credits (Rohit Jnagal): http://www.slideshare.net/RohitJnagal/docker-internals


Docker Grounds up: Processes & Networking

We have resources, isolation, and file system management.

Docker daemon handles starting/stopping processes with:
- Attach logic
- Logs
- TTY management
- Docker run options
- Events and container state

Network Management
- NAT, Bridge, Veth
- Expose
- Links

Image Source and Credits (Rohit Jnagal): https://docs.google.com/presentation/d/1juVgXiLTM-ZmAmYBOshNwhBABkUqwIxVodHZwq-0eGg/edit?usp=sharing

Cgroups

Docker also makes use of kernel control groups for resource allocation and isolation. A cgroup limits an application to a specific set of resources. Control groups allow Docker Engine to share available hardware resources to containers and optionally enforce limits and constraints.

Docker Engine uses the following cgroups:

- **Memory cgroup** for managing accounting, limits and notifications.
- **HugeTBL cgroup** for accounting usage of huge pages by process group.
- **CPU group** for managing user / system CPU time and usage.
- **CPUSet cgroup** for binding a group to specific CPU. Useful for real time applications and NUMA systems with localized memory per CPU.
- **BlkIO cgroup** for measuring & limiting amount of blckIO by group.

- **net_cls** and **net_prio cgroup** for tagging the traffic control.
- **Devices cgroup** for reading / writing access devices.
- **Freezer cgroup** for freezing a group. Useful for cluster batch scheduling, process migration and debugging without affecting prtrace.

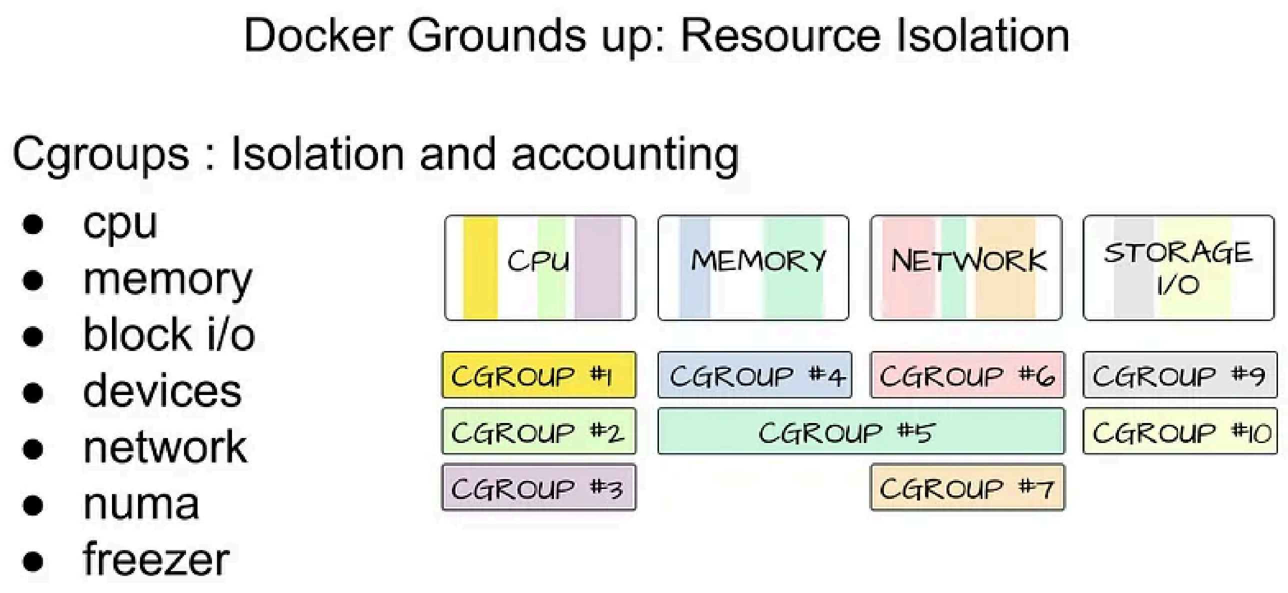


Image Source and Credits (Rohit Jnagal & Mairin): http://www.slideshare.net/RohitJnagal/docker-internals and https://mairin.wordpress.com/2011/05/13/ideas-for-a-cgroups-ui/

Union File Systems

Union file systems operate by creating layers, making them very lightweight and fast. Docker Engine uses UnionFS to provide the building blocks for containers. Docker Engine can use multiple UnionFS variants, including AUFS, btrfs, vfs, and devicemapper.

Docker Grounds up: Filesystem

File-system Isolation:

- Building a rootfs dir and chroot into it.
- With mount namespace, use pivot-root.

Features:

- Layering, CoW, Caching, Diffing

Solutions:

- UnionFS, Snapshotting FS, CoW block devices

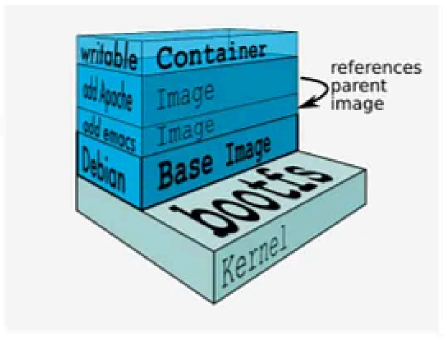
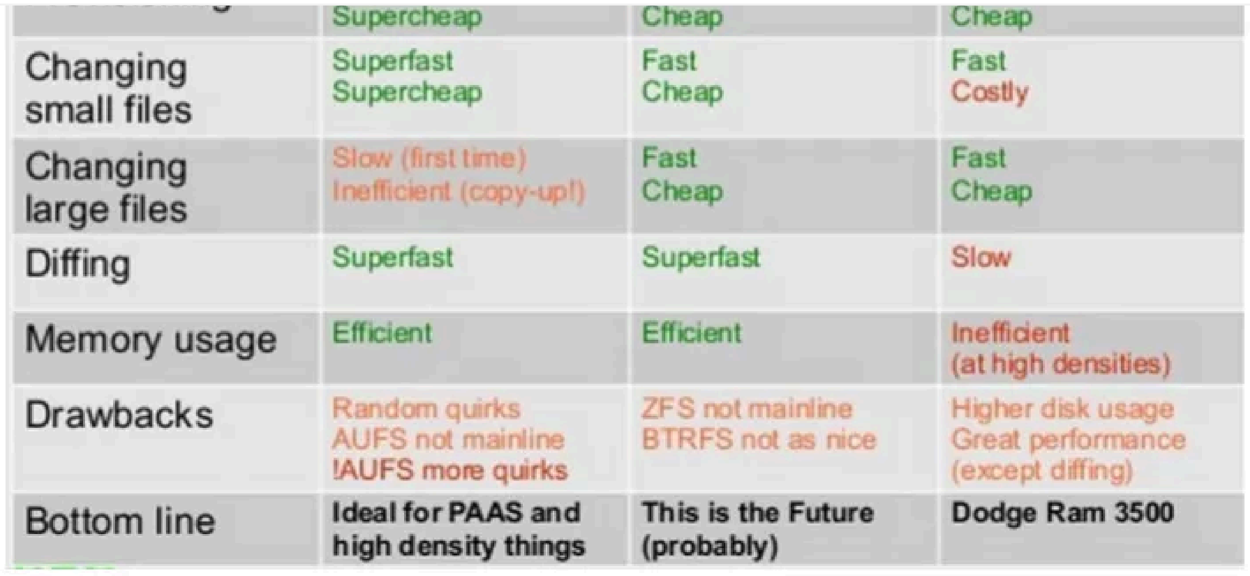


Image Source and Credits (Rohit Jnagal): http://www.slideshare.net/RohitJnagal/docker-internals

Docker Grounds up: Filesystem



	Supercheap	Cheap	Cheap
Changing small files	Superfast Supercheap	Fast Cheap	Fast Costly
Changing large files	Slow (first time) Inefficient (copy-up!)	Fast Cheap	Fast Cheap
Diffing	Superfast	Superfast	Slow
Memory usage	Efficient	Efficient	Inefficient (at high densities)
Drawbacks	Random quirks AUFS not mainline !AUFS more quirks	ZFS not mainline BTRFS not as nice	Higher disk usage Great performance (except diffing)
Bottom line	**Ideal for PAAS and high density things**	**This is the Future (probably)**	**Dodge Ram 3500**

Image Source and Credits (Jerome Petazzoni): http://www.slideshare.net/RohitJnagal/docker-internals

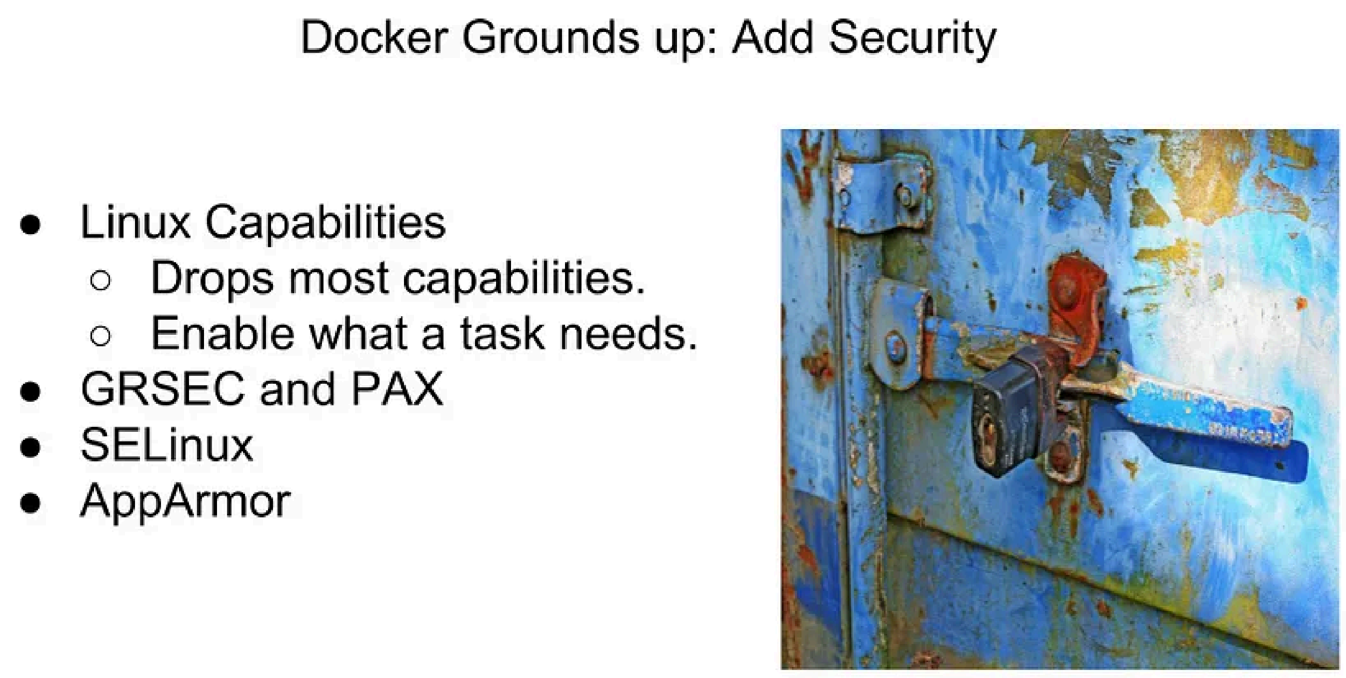
Container Format

Docker Engine combines the namespaces, control groups and UnionFS into a wrapper called a container format. The default container format is libcontainer.

Security

Docker Engine makes use of AppArmor, Seccomp, Capabilities kernel features for security purposes.

- **AppArmor** allows to restrict programs capabilities with per-program profiles.
- **Seccomp** used for filtering syscalls issued by a program.
- **Capabilties** for performing permission checks.



Docker Grounds up: Add Security

- Linux Capabilities
  - Drops most capabilities.
  - Enable what a task needs.
- GRSEC and PAX
- SELinux
- AppArmor

Image Source and Credits (Leo Reynolds): http://www.slideshare.net/RohitJnagal/docker-internals

. . .

Source: ~ http://docker-saigon.github.io/post/Docker-Internals/

Disclaimer: Content and Image source has been mentioned. Special credit to concerned folks.

Docker DevOps Cloud Computing Linux Cloud

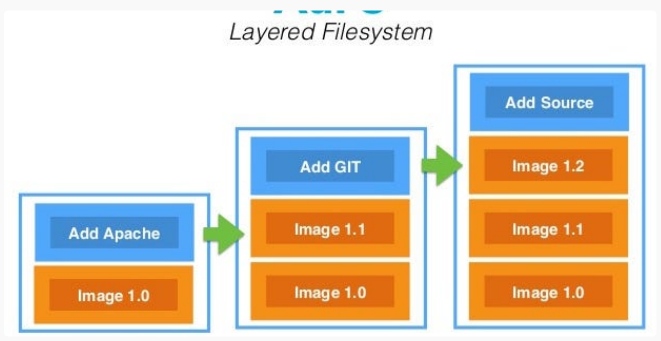
Written by Nitin AGARWAL

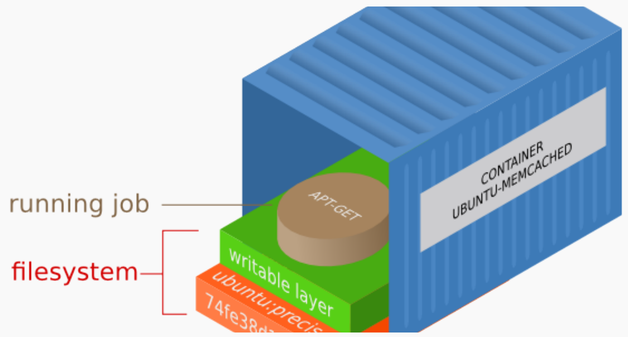
Follow

838 Followers

Senior Software Engineer — Cloud Native and Distributed Systems

More from Nitin AGARWAL





Nitin AGARWAL Nitin AGARWAL

Docker Container's Filesystem Demystified

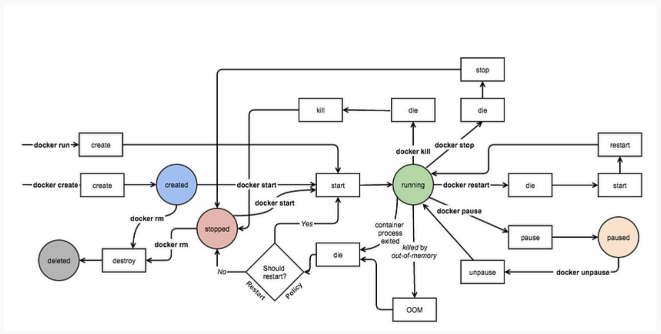
In this blog, we'll discuss about the Filesystem of Docker Containers in particular. Docker…

Jan 22, 2017 394 3

Best Practices for working with Dockerfiles

Dockerfiles allow users to define the exact actions needed to create a new container…

Jan 4, 2017 372 2

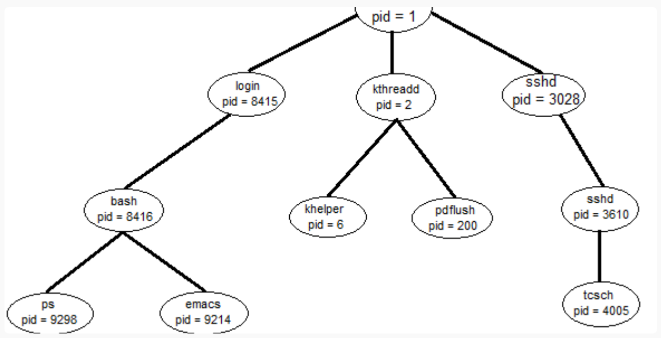


Nitin AGARWAL

Lifecycle of Docker Container

Though the above pic is self explanatory, we'll just discuss the basics / CLI commands for…

Jan 5, 2017 920 10



Nitin AGARWAL

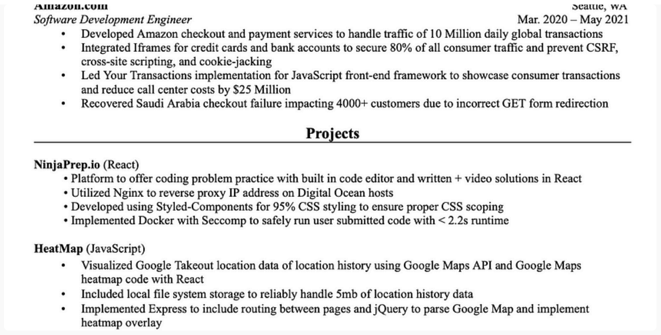
An Init System inside the Docker Container

In this blog, we'll understand the PID 1 zombie reaping problem and how it can be solved…

Jan 5, 2017 251 2

See all from Nitin AGARWAL

Recommended from Medium


Amazon.com Seattle, WA
Software Development Engineer Mar. 2020 – May 2021
- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)
- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)
- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

Alexander Nguyen in Level Up Coding

The resume that got a software engineer a $300,000 job at Google.

1-page. Well-formatted.

Jun 1 23K 458

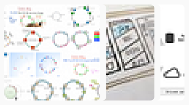
Furkan Türkal

How does Docker actually work? The Hard Way: A Technical Deep...

Unveiling the power of Docker: What is Docker? How does Docker work? Explore th...

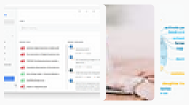
Jun 4 559 6

Lists

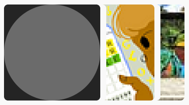


General Coding Knowledge
20 stories · 1625 saves

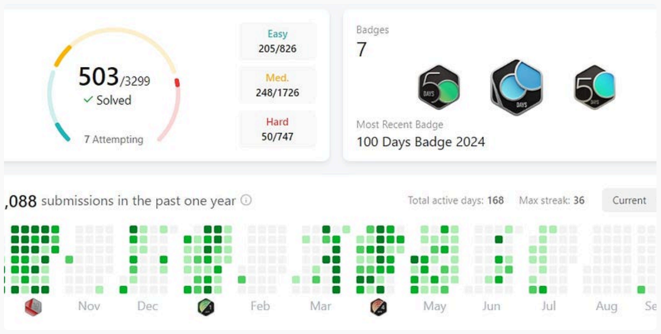
Coding & Development
11 stories · 836 saves



Productivity
242 stories · 569 saves



Staff Picks
745 stories · 1352 saves



Surabhi Gupta in Code Like A Girl

Akhilesh Mishra

Why 500 LeetCode Problems Changed My Life

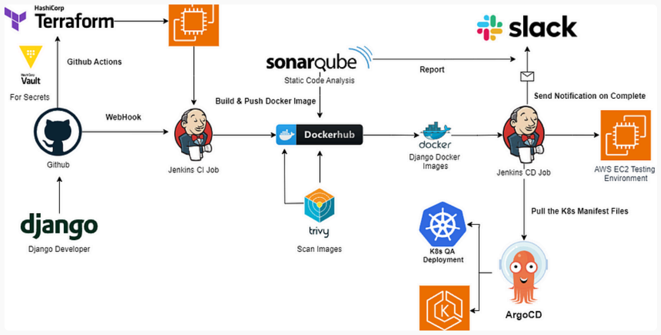
How I Prepared for DSA and Secured a Role at Microsoft

Sep 26 1.5K 34

Everything You Need To Know About Docker

Devops zero to hero #3—A Complete Guide to start using Docker in your devops workflow

Jan 23 522 3



Sachinthana Buddhika

CI/CD Pipeline with GitHub, Jenkins and Docker

A CI/CD pipeline automates code deployment using GitHub, Jenkins, and Docker.

Jun 16 14 1

Joel Wembo in Django Unleashed

Technical Guide: End-to-End CI/CD DevOps with Jenkins, Docker,...

Building an end-to-end CI/CD pipeline for Django applications using Jenkins, Docker,...

Apr 12 815 16

See more recommendations