Docker

فهرست

۱. مقدمه ۵
2. منظور از Container چیست؟ ۵
3. Dockerچیست؟ ٦
- ❖ Dockerfile ٦
- ❖ Docker image ۷
- ❖ Docker run ۷
- ❖ Docker Hub ۷
- ❖ Docker Engine ۷
4. داکر چطور کار می کند؟ ۸
5. داکر (Docker) چیست؟ ۸
- ❖ مزیت داکر ۱۲
- ❖ درباره گریز از تحریم ۱٤
6. آموزش نصب داکر بر روی سیستم‌عامل لینوکس ۱۵
- ❖ نحوه نصب داکر توسط خودم ۱۸
7. آشنایی با ابزارهای خط فرمان داکرDocker ۲٤
- ❖ ۱-Engine CLI: ۲۵
- ❖ ۲-Compose CLI : ۲۵
- ❖ ۳-Machine CLI : ۲۵
- ❖ ۴-UCP CLI : ۲٦
- ❖ ۵-DTR CLI : ۲۸
۸. ابزارهای جانبی کنار داکر ۲۸
- ❖ ۱-Kitematic ۲۸
- ❖ ۲-Portainer ۲۹
- ❖ ۳-Docker Machine ۳۱
- ❖ ۴-Docker Toolbox ۳۲
- ❖ ۵-Docker Compose ۳۲
- ❖ ۶-VirtualBox ۳۳
9. نصب Docker ToolBox بر روی ویندوز ۳٤
۱۰. نصب و استفاده ازKitematic ۳۷
۱۱. داکر فایل(DockerFile) ٤۲
12. اشتراک گذاری داکر امیج(Docker Image) ٤۷
۱۳. داکر کامپوز(Docker Compose) ٤۷
14. نسخه‌های مختلف داکر که خوب است بدانیم ۵۰
- ❖ نسخه‌یEnterprise Edition ۵۱

- نسخه‌ی Community Edition ٥٢
- ١٥. دستورات مهم داکر ٥٤
- دستور docker ps ٥٤
- دستور docker info ٥٥
- دستور docker pull ٥٦
- دستور docker images ٥٧
- دستور docker create ٥٨
- دستور docker run ٥٩
- دستور docker rename ٦٢
- دستور docker rm ٦٣
- دستور docker rmi ٦٤
- دستور docker tag ٦٥
- دستور docker search ٦٦
- دستور docker build ٧٠
- دستور docker commit ٧٣
- دستور docker cp ٧٤
- دستور docker events ٧٥
- دستور docker exec ٧٨
- دستور docker export ٨٠
- دستور docker import ٨١
- دستور docker save ٨٣
- دستور docker load ٨٤
- دستور docker restart ٨٦
- دستور docker kill ٨٧
- دستور docker inspect ٨٩
- دستور docker history ٩١
- دستور docker login ٩٣
- دستور docker logout ٩٥
- دستور docker logs ٩٦
- دستور docker port ٩٨
- دستور docker diff ٩٩
- دستور docker push ١٠٠
- دستور docker start ١٠١
- دستور docker stop ١٠٢
- دستور docker pause ١٠٤

- دستور docker unpause ۱۰٤
- دستور docker top ۱۰٥
- دستور docker stats ۱۰٦
- دستور docker update ۱۰۹
- دستور docker container ۱۱۰
- دستور docker image ۱۱۲

16. توضیح کوتاه اما بسیار مفید در مورد شبکه‌ی داکر ۱۱۳
- شبکه‌ی bridge ۱۱۳
- شبکه‌ی none ۱۱٤
- شبکه‌ی host ۱۱٥

17. داکر و iptables (docker and iptables) ۱۱٦

18. استفاده از DNS سرور داخلی داکر ۱۱۸

19. دستورات مدیریتی داکر– docker network ۱۲۱
- دستور docker network connect ۱۲۲
- دستور docker network create ۱۲۳
- دستور docker network disconnect ۱۲٥
- دستور docker network inspect ۱۲٦
- دستور docker network ls ۱۲۷
- دستور docker network prune ۱۲۹
- دستور docker network rm ۱۳۰

20. دستورات مدیریتی داکر–docker secret ۱۳۰
- دستور docker secret ۱۳۱

21. دستور مدیریتی docker node ۱۳٤
- دستورls docker node ۱۳٤
- دستور docker node demote ۱۳٥
- دستور docker node promote ۱۳٥
- دستور docker node inspect ۱۳٦
- دستور docker node ps ۱۳٦
- دستور docker node rm ۱۳۷
- دستور docker node update ۱۳۷

22. ایجاد ریجیستری خصوصی ۱۳۸

۲۳. راکت چت ۱٤۱

۱. مقدمه

داکر یک پلتفرم نرم افزاری برای ساخت اپلیکیشن های مبتنی بر Container است. محیط های اجرایی کوچک و سبک که به طور مشترک از هسته سیستم عامل استفاده می‌کنند اما در عین حال در یک محیط ایزوله و کاملا جدا از هم قرار دارند. هر چند مفهوم container یا نگهدارنده از مدت‌ها قبل در حوزه IT مطرح بود اما داکر به عنوان یک پروژه متن باز در سال ۲۰۱۳ معرفی و عرضه شد. در واقع داکر باعث شد container جان تازه ای بگیرد و دوباره محبوب شود. توسعه نرم افزار به سمت استفاده از Container و میکرو سرویس‌ها رفت و بعدها به عنوان توسعه ابری یا Cloud-native Development شناخته شد. شاید بعضی از مفاهیم که تا اینجا گفته شد را نشناسید اما اصلا نگران نباشید چون تمام این موارد را تا انتهای مطلب با هم مرور خواهیم کرد و در پایان متوجه خواهیم شد که داکر چیست و چه کاربردهایی دارد.

۲. منظور از Container چیست؟

یکی از اهدافی که سیستم های جدید توسعه نرم افزار دنبال می کنند، این است که برنامه ها در یک محیط، اما به صورت ایزوله و جدا از هم نگهداری شوند. به این ترتیب فعالیت آنها بر روی یکدیگر تاثیر نداشته و جدا از هم کار می کنند. البته اجرای این فرآیند به خاطر استفاده از پکیج ها، کتابخانه ها و دیگر کامپونتت های نرم افزاری می تواند پیچیده شود.

یکی از راه های پیاده کردن این تکنولوژی استفاده از ماشین مجازی (Virtual Machine) است که برنامه ها را روی یک سخت افزار اما کاملا جدا از هم نگه می دارد. پس در این حالت کامپوننت های ما تداخل خاصی با هم نداشته و رقابت برای استفاده از منابع سخت افزاری هم به حداقل می رسد. اما ماشین های مجازی مشکلاتی هم دارند. اول از همه اینکه نرم افزارهای سنگینی بوده و سخت افزار نسبتا قدرتمندی می خواهند. همینطور هر برنامه نیاز به سیستم عامل جداگانه دارد که ممکن است این سیستم عامل ها حجم های چند گیگابایتی داشته باشند. و اینکه ممکن است نگهداری و بروزرسانی آنها دشوار شود.

بررسی عمیق تر Container به همراه یک مثال

در مقابل Container قرار دارد که می‌تواند جایگزین مناسبی برای ماشین‌های مجازی باشد. Container محیط‌های اجرایی را جدا کرده و هسته سیستم عامل را به اشتراک می گذارد. حجم آنها معمولا به مگابایت بوده و نسبت به ماشین های مجازی از

منابع کمتری استفاده می‌کند. همینطور برخلاف ماشین‌های مجازی که برای اجرا نیاز به زمان نسبتا زیادی دارند، **Container**ها بلافاصله اجرا می شوند.

زمانی که **Container** را با ماشین مجازی مقایسه می‌کنیم یعنی با یک شبیه ساز طرف حساب هستیم. اما دقیقا چه چیزی را شبیه سازی می کنیم؟ برای درک بهتر موضوع بهتر است از یک مثال استفاده کنیم. فرض کنید در شرکتی مشغول به کار هستید و ناهار خود را هر روز در خانه درست کرده و آن را داخل یک ظرف به شرکت می برید تا آنجا میل بفرمایید. دیگر لازم نیست داخل شرکت شروع به پختن غذا کنید چون احتمالا زمان زیادی را از شما می گیرد. کار **Container** هم تا حدودی شبیه به این است. شما پروژه خود را (غذا) داخل **Container** (ظرف غذا) قرار داده و آن را هر کجا که دوست داشتید (مثلا شرکت) می برید.

۳. Dockerچیست؟

داکر یک پروژه متن باز است که اجازه می‌دهد **Container** یا برنامه‌های مبتنی بر **Container** بسازید. با اینکه داکر در شروع کار برای لینوکس ساخته شد اما امروزه به خوبی در ویندوز و **MacOS** هم اجرا می‌شود. برای ساخت یک برنامه با داکر باید از کامپوننت‌های مختلفی استفاده کنیم. حالا برای درک بهتر مفهوم داکر بهتر است سراغ این کامپوننت‌ها رفته و آن‌ها را بررسی کنیم:

❖ Dockerfile

هر **Container** داکر با یک فایل داکر شروع به کار می‌کند. **Dockerfile** یک فایل متنی بوده که داخل آن با یک سینتکس ساده و قابل فهم دستورالعمل‌های ساخت **Docker Image** قرار داده شده است (کمی جلوتر این مفهوم را بررسی خواهیم کرد) این فایل اطلاعات بسیار مهمی را در برمی‌گیرد که برای راه اندازی داکر استفاده از آنها ضروری است. در واقع **Dockerfile** مشخص می‌کند که پشت **Container** ما چه سیستم عاملی قرار بگیرد، همینطور از چه زبان‌ها، متغیرهای محلی، پورت‌های شبکه یا غیره استفاده شود. و مهم تر از همه اینکه مشخص کند **Container** ما بعد از اینکه واقعا اجرا شد قرار است چه کاری انجام دهد.

❖ Docker image

در واقع زمانی که کار نوشتن Dockerfile را تمام کردید، یک قابلیت به اسم Docker Build را فراخوانی می‌کنید که وظیفه دارد یک Image بر اساس محتویات Dockerfile شما بسازد. Dockerfile شامل یک سری دستورالعمل برای ساختن یک Image است، در حالی که Docker Image یک فایل قابل حمل است که شامل یک سری دستورالعمل بوده که مشخص می‌کند Containerکدام کامپوننت‌های نرم افزاری را اجرا کند و اینکه چطور آنها اجرا شوند. به احتمال زیاد Dockerfile بخواهد تعدادی فایل را از مخزن‌های مختلف (Repository) دانلود کند و اینجا باید به طور واضح مشخص کنید که کدام نسخه‌ها دریافت شوند. همینطور Image ساخته شده استاتیک می‌باشد، یعنی یک بار ساختن آن کافی بوده و نیازی به تغییر آن ندارید. همانطور که از اسم آنها می‌توانید حدس بزنید، Image یک تصویر از سیستم عامل اصلی می‌باشد.

❖ Docker run

قابلیت Docker run در واقع یک دستور است که Container را راه اندازی می‌کند. می‌دانیم که هر Container یک instance یا نمونه از Image است و ماهیت Container به صورت موقتی می‌باشد. اما با این حال می توان آنها را متوقف (Stop) یا راه اندازی مجدد (Restart) کرد. هر Image می‌تواند تعداد زیادی Container داشته باشد (تا زمانی که هر کدام نام منحصر به فردی داشته باشند)

❖ Docker Hub

هر چند ساختContainer ها کار راحتی است، اما لزومی ندارد برای هر Image از اول Container بسازید. Docker Hub یک مخزن Saas برای به اشتراک گذاری و مدیریتContainer ها است. در آنجا می‌توانید Imageهای رسمی داکر که معمولا به صورت متن باز هستند را پیدا کنید. همینطور عموم مردم هم پروژه‌های خود را به این Repository اضافه می‌کنند.

❖ Docker Engine

مغز متفکر و هسته اصلی داکر به حساب می‌آید. یک تکنولوژی **کلاینت/سرور** که Container ها را ساخته و آنها را اجرا می‌کند. در واقع اگر کسی درباره داکر صحبت کرده و منظورش شرکت داکر یا پروژه داکر نباشد، یعنی دارد درباره **Docker Engine** حرف می‌زند. این موتور در دو نسخه Enterprise یا Community عرضه می شود. نسخه Community به طور متن باز و کاملا

رایگان قابل استفاده است در حالی که نسخه Enterprise یک سری قابلیت‌های اضافه داشته و سالانه برای هر Node نزدیک به ۱۵۰۰ دلار هزینه دارد.

۴. داکر چطور کار می کند؟

تمام این نکات را گفتیم تا به این بخش رسیده و بفهمیم داکر چیست و چطور کار می‌کند. داکر را می‌توانیم یک مجازی ساز در نظر بگیریم، اما نه به شکلی که ماشین‌های مجازی کار می‌کنند. این سرویس به شما کمک می‌کند یک محیط را به چند بخش تقسیم کرده و و در هر بخش یک برنامه مجزا اجرا کنید. برای درک بهتر موضوع یک لحظه داکر را شبیه به ماشین مجازی در نظر بگیرید، می‌دانیم که ماشین‌های مجازی با Hypervisor عملیات شبیه سازی را انجام می‌دهند اما در داکر این لایه حذف شده و ما مستقیم با هسته سیستم عامل کار می‌کنیم. این کار بازدهی سیستم را بسیار بالا می‌برد.

ما به جای اینکه چند سیستم عامل را داخل یک شبیه ساز نصب کنیم یک بار داکر را نصب می‌کنیم. این فرآیند را شبیه به نصب یک نرم افزار عادی در نظر بگیرید. بعد داکر محیط‌های مستقلی به نام Container ایجاد می‌کند که هر Container می‌تواند شامل بسته‌های نرم افزاری مختلفی باشد. حالا می‌توانید داخل داکر Container های مختلفی ایجاد کنید و پروژه‌های خود را به طور مستقل روی هر کدام اجرا نمایید.

نتیجه گیری

داکر یک ابزار اوپن سورس برای طراحی و ساخت برنامه‌ها به کمک Container است. Container به برنامه نویس کمک می‌کند که پروژه خود را با تمام بخش‌هایی که دارد (مثل کتابخانه ها، وابستگی ها و غیره) به صورت یک پکیج دربیاورد. به لطف این تکنیک، برنامه نویس خیالش راحت است که برنامه او می‌تواند در سیستم‌های دیگر بدون نیاز به تنظیمات خاص یا ابزارهای جانبی اجرا شود. داکر را شبیه به ماشین مجازی در نظر بگیرید، با این تفاوت که بازدهی بسیار بالاتری دارد.

۵. داکر (Docker) چیست؟

داکر برترین پلتفرم مجازی سازی سطح سیستم عامل جهان است.

داکر نام کمپانی توسعه دهنده تنها پلتفرم مجازی سازی سطح سیستم عامل با پشتیبانی از امکانات ابری است. امروزه کسب و کارها تحت فشار لزوم مهاجرت به دنیای دیجیتال قرار دارند, اما توسط اپیلکیشن ها و زیرساخت‌های سابق احاطه شده‌اند. داکر، استقلال واقعی نرم افزار از زیرساخت و استقلال توسعه دهندگان از متخصصان IT را فراهم آورده تا پتانسیل آنها را آزاد کرده و شرایط را برای همکاری و نوآوری بهتر فراهم بیاورد.

در ادامه به ویژگی های برجسته داکر میپردازیم

۱-قابلیت حمل (PORTABILITY) :

خداحافظی با فضای کار محدود به کامپیوتر شخصی. دستیابی به استقلال در همه محیط های ابری و پیشرو

۲-چالاکی (Agility) :

شتابدهی به توسعه نرم افزار تا ۱۳ برابر و پاسخدهی بلادرنگ به نیاز های کاربر

۳-کاهش هزینه (COST SAVINGS) :

بهینه سازی مصرف منابع زیرساخت و تسهیل عملیات‌های مختلف در نهایتا کاهش ۵۰ درصدی هزینه‌ها

۴-امنیت (SECURITY) :

به مقصد رساندن توسعه اپلیکیشن به صورت امن در تمام روند توسعه با ظرفیت‌های امنیتی درون نهاده و پیکربندی‌های خارجی

پلتفرم داکر تنها پلتفرم مبتنی بر مجازی سازی سطح سیستم عامل می‌باشد که بستر مناسبی فراهم کرده است برای ایجاد, ایمن سازی, و مدیریت آرایه گسترده‌ای از اپلیکیشن‌ها, از مرحله توسعه تا ارائه بر روی بستر سازمانی یا ابری.

نسخه Community داکر (CE) با ابزارهایی که دارد, به توسعه دهندگان این امکان را می‌دهد تا اپلیکیشن‌های خود را در بستری مناسب توسعه دهند. و نسخه شرکتی داکر , (EE) بستر آی تی با معماری چندگانه را عملیات‌های مقیاس گسترده فراهم می‌کند.

معرفی داکر (Docker)

داکر یا Docker یک پلتفرم اوپن سورس فوق‌العاده برای مدیریت مجازی سازی سطح سیستم عامل است, در پست قبل گفتیم که این تکنیک مجازی سازی یا بهتر است بگوییم ایزولیشن Containerization است. داکر یا Docker به کشتی‌های بزرگ حمل کانتینر گفته می‌شود, پس فلسفه نام گذاری داکر, توانایی آن در مدیریت و هندل تعداد زیادی محیط مجازی یا کانتینر است.

برای ایجاد یک کانیتنر وردپرس روی داکر تنها کافیست در خط فرمان تایپ کنید:

```
Docker pull worpress

Docker run wordpress
```

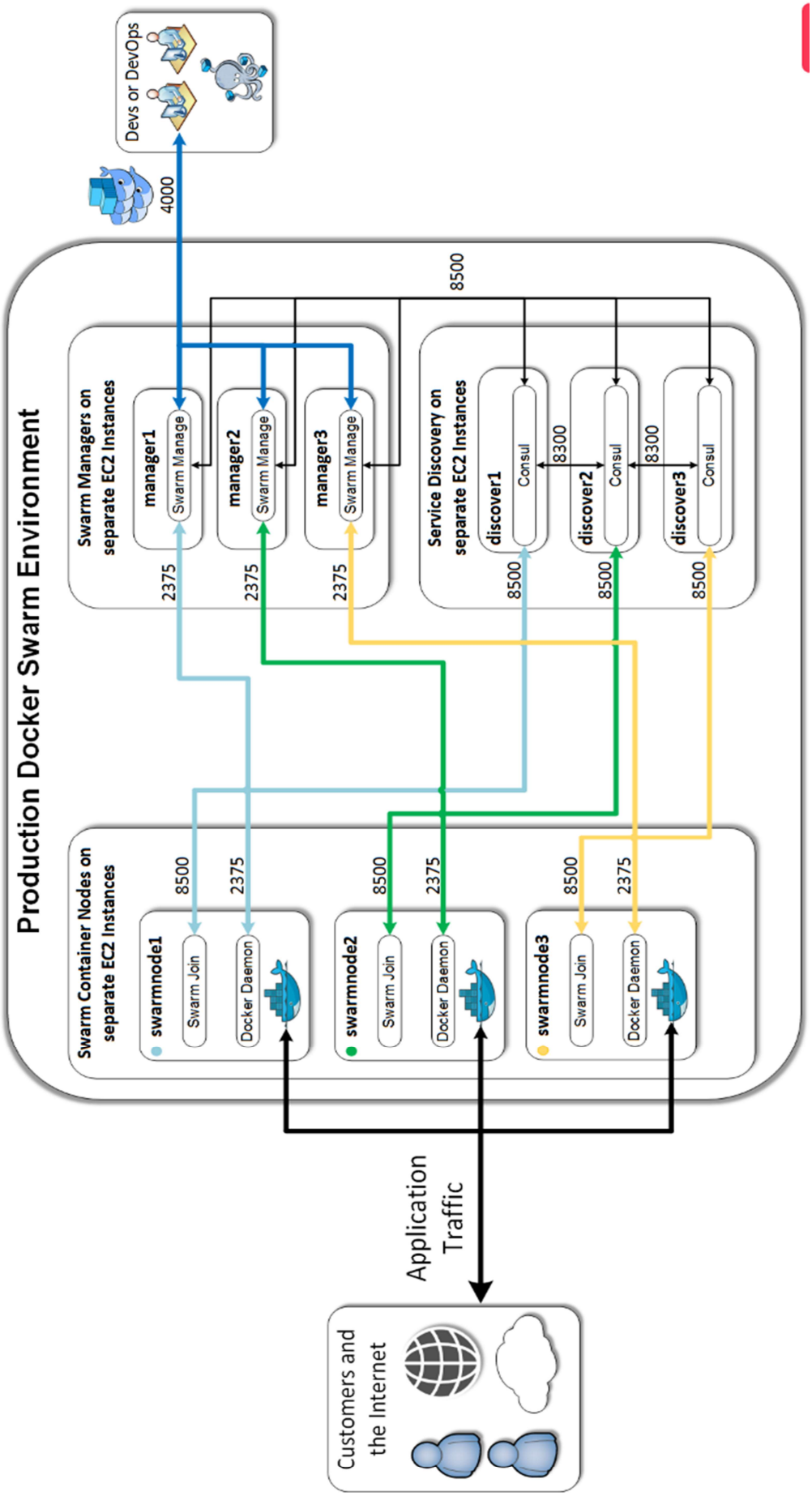
تنها با این کار یک کانتینر آماده وردپرس برای شما فراهم می‌شود, اگر می‌خواهید از طریق وب به وردپرس دسترسی داشته باشید تنها کافیست پورت ۸۰ کانتینر را به یک پورت دیگر map کنید مثلا:

```
docker run wordpress -p 8080:80
```

پورت ۸۰ کانتینر را روی پورت ۸۰۸۰ سرور اصلی شما پورت می‌کند و شما از هر کجا اینترنت می‌توانید از طریق این پورت به کانتینر وردپرس خود وصل شوید و مثل یک سرور مجازی واقعی با آن کار کنید.

شما به سادگی می‌توانید محیط برنامه نویسی خود را به صورت یک کانتینر با سایر اعضای تیم به اشتراک بگذارید, و حتی دستورهای اولیه برای راه اندازی کانتینر را تعیین کنید و پورت‌های خروجی آنرا مشخص کنید. با کمک Swarm می‌توانید شبکه گسترده‌ای از کانتینرهای متصل به هم را در قالب یک کانتینر واحد بسته‌بندی کنید. و با کمک Docker Machine فرآیند ایجاد کانتینرها را خودکار کنید.

Docker HUB مجموعه‌ای کامل از ایمیج‌های داکر را فراهم آورده است تا به سادگی از tomcat جاوا تا وب سرور Caddy را Pull کرده و در چند ثانیه آماده بهره برداری کنید.

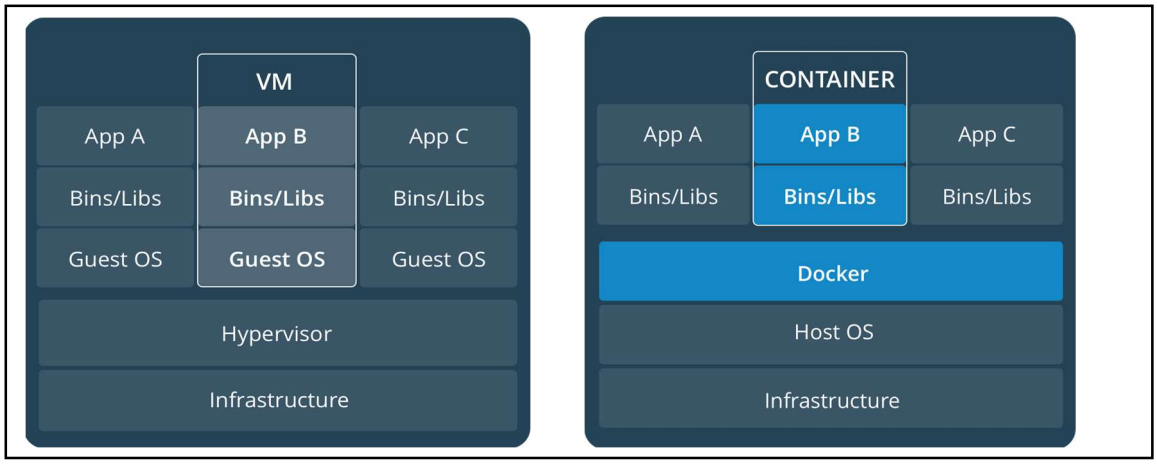

Devs or DevOps
4000
Production Docker Swarm Environment
Swarm Managers on separate EC2 Instances
manager1
Swarm Manage
manager2
Swarm Manage
manager3
Swarm Manage
2375
2375
2375
8500
Service Discovery on separate EC2 Instances
discover1
Consul
discover2
Consul
discover3
Consul
8300
8300
8500
8500
8500
Swarm Container Nodes on separate EC2 Instances
swarmnode1
Swarm Join
Docker Daemon
swarmnode2
Swarm Join
Docker Daemon
swarmnode3
Swarm Join
Docker Daemon
8500
2375
8500
2375
8500
2375
Application Traffic
Customers and the Internet

هدف اصلی داکر ساخت یک محیط ایزوله برای اپلیکیشن‌ها هست تا بهترین بازده رو داشته باشند. حالا چطوری اینکارو میکنه؟ مجازی سازی یعنی یه فضا رو کامل جدا می‌کند (که بعدا بهش می‌گیم**container**) تا محیط ایزوله شود.

اولین چیزی که به ذهن خودم رسید این بود خب چه فرقی با **vmware** داره اونم کارش همینه که! دقیقا همینطوری و دقیقا نه - در واقع هر دوتاشون میتونن برای اینکار استفاده بشن اما اینکار رو از روش های مختلفی انجام میدن.

وقتی از **vmware** استفاده می‌کنیم منابع سخت افزاری رو برای محیط های مختلف تقسیم می‌کنیم و در نهایت برایشون سیستم عامل نصب می‌کنیم. و اپلیکیشن رو توی اون سیستم عامل نصب می‌کنیم.

(**Hypervisor** ها مجازی سازها هستند مثل **vmware, virtualbox, vSphere** و ...



اما داکر به جای اینکه از همون اول منابع سخت افزاری رو تقسیم کنه و سرور مجبور باشه چندین سیستم عامل رو همزمان اجرا کنه روی سیستم عامل نصب میشه (مثل اوپن سوزه) و حالا محیط‌های ایزوله روی خود داکر اجرا میشه که از طریق **image** های اماده درست می‌شه و درخواست هاشون رو به سیستم عامل میده و سیستم عامل تحویل کرنل میده.

❖ مزیت داکر

۱ - بخاطر اینکه برای یه محیط ایزوله یه سیستم عامل جدا نصب نمیکنیم یعنی هدر رفت منابعمون خیلی کمتر میشه.

۲ - در حالت اول(ماشین مجازی) از همون اول منابع سخت افزاری تقسیم شده و اگه بار یه سرویس خیلی کم باشه نمیشه تواناییش رو به سرورهای دیگ که درگیرن بده ولی توی داکر بخاطر اینکه تو سطح سیستم عامل هستیم اینکار خیلی راحت انجام میشه (البته توانایی اینکه محدودیت استفاده از منابع هم میتونیم بزاریم)

۳ - وقتی یه سرویس رو توی داکر راه اندازی می‌کنیم می‌توانیم image ساخته شده رو ذخیره کنیم و روی هر سیستمی خیلی سریع راه اندازی کنیم.

در ماشین‌های مجازی هزینه‌ها بالا هستند. هزینه لایسنس و آپدیت سیستم عامل‌ها و خود هایپروایزرها و همچنین منابع زیادی(از جمله حافظه) درگیر هستند اما با این وجود محیط کاملا ایزوله به وجود می‌آورند. و اگر به سیستم عامل مجازی مشکلی پیش بیاید در این صورت سیستم عامل اصلی خراب نمی‌شود.

(معایب:) در داکر هم استفاده از نرم افزارها به سادگی استفاده از نرم افزارها در ماشین‌های مجازی نیست و هر نرم افزار باید به Container تبدیل شود سپس استفاده شود هر چند که تعداد این Container بسیار زیاد موجود است.

داکر (Dockerme.ir) یک پلتفرم متن باز است که بر مبنای سیستم‌عامل لینوکس راه‌اندازی شده است. اما اگر از سیستم‌عامل دیگری همانند ویندوز و مک استفاده کنید می‌توانید از داکر و قابلیت‌های آن نیز بهرمند بشید.

داکر با استفاده از دو قابلیت مهم موجود در کرنل لینوکس به نام‌های Cgroups و Namespace امکانی را فراهم آورده است که پروسه‌ها و نرم‌افزارها به صورت مجزا در محیط کاملا ایزوله‌ای بر روی کرنل لینوکس راه‌اندازی شود که به این محیط و بسته‌ی ایزوله، کانتینر (Container) می‌گویند. البته تا این‌جای کار سرویس داکر کار خاصی انجام نداده و این موضوع رو تمامی Linux Containers ها نیز دارا هستند. اما داکر امکانی را فراهم آورد که بتوان به راحتی با این قابلیت ارتباط برقرار کرده و از آن نهایت استفاده را کرد.

پس مشخص شد که داکر، امکانی را فراهم می‌کند که بتوان به راحتی با کانتینرها کار کرد. داکر این امکان را فراهم می‌کند که بتوان نرم‌افزار و کدهایی که آماده شده است را با سرعت خیلی زیادی تست و در محیط عملیاتی (Production) استفاده کرد. داکر این زمان را به شدت کاهش می‌دهد و البته که کیفیت آن را نیز بالا خواهد برد. با استفاده از داکر به خوبی می‌توان مواردی

که برای یک پروژه نیاز است را در کنار هم جمع‌آوری کرد و به صورت کامل آنها را در یک پکیج قرار داد. یعنی به اختصار هر آن چیزی که یک نرم‌افزار نیاز خواهد داشت اعم از پکیج‌های وابسته (Dependency Package) و کتابخانه‌ها (library) مورد نیاز در یک کانتینر آماده خواهد شد و همواره همراه نرم‌افزار در هر محیط که نیاز به راه‌اندازی دارد منتقل خواهد شد. در کل داکر یک سرویس بسیار مناسب و کارآمد برای استفاده در محیط‌های راه‌اندازی، آزمایشگاهی و عملیاتی می‌باشد.

اما نکته‌ی مهمی که نیاز است که حتما به آن توجه شود این است که برای استفاده از آن باید دانش کافی در زمینه‌ی داکر و میکروسرویس (Micro Service) را داشت.

❖ **درباره گریز از تحریم**

متاسفانه گروه داکر اجازه‌ی استفاده از سرویس‌های خود را بر روی برخی از کشورها من جمله ایران مسدود کرده است. این موضوع در نصب سرویس داکر، دانلود ایمیج‌های داخل Docker Hub و برخی از دستورات داکر که با این سایت ارتباط برقرار می‌کند موثر است. گریز از تحریم داکر با چند روش می‌توانیم آن را دور بزنیم که یکی از آنها استفاده از فیلتر شکن‌ها می‌باشد و دیگری استفاده از سرویس(سایت) شکن است که DNS سیستم خود را به آدرس آی پی قرار داده شده در شکن تنظیم می‌کنیم و بدون مشکل کارها را انجام می‌دهیم.

تاریخچه‌ی داکر:

در این متن به صورت اختصار تاریخچه پیدایش داکر رو با همدیگر بررسی می‌کنم.

سال ۲۰۰۶ : در این سال پروسس کانتینر (Process Container) ارائه شد.

سال ۲۰۰۸ : در این سال LXC لینوکس کانتینر (Linux Containers) ارائه شد که این موضوع مقدمه‌ای مناسب و خوبی برای استفاده از کانتینرها بود.

سال ۲۰۱۳ : سرویس داکر (Docker) در این سال ارائه شد. در همین سال شرکت بزرگ Redhat اعلام کرد که پروژه‌های Fedora, Red Hat Enterprise Linux, and OpenShift به سمت این پروژه خواهد برد و از این پروژه حمایت کرد.

سال ۲۰۱۴ : در این سال شرکت ماکروسافت اعلام کرد که نسخه‌ی آینده ویندوز سرور خودش رو به سمتی خواهد برد که بتواند داکر کلاینت رو پشتیبانی کند. شرکت IBM نیز در همین سال اعلام کرد که برنامه‌ریزی خواهد کرد تا در IBM Cloud از این سرویس استفاده خواهد کرد و در همین سال شراکت (Partnership) خودش را با سرویس داکر اعلام کرد.

سال ۲۰۱۶ : شرکت داکر سرویس تجاری (Commercial) خود را در این سال ارائه کرد.

سال ۲۰۱۸ : در حال حاضر سرویس داکر دارای جامعه‌ی متن باز بسیار پویا و کاملی می‌باشد که بسیار فعال بوده و تقریبا همواره در حال رشد و شکوفایی بیشتر می‌باشد.

نکته‌ی آخر : در حال حاضر سرویس‌های بزرگ و معتبری از داکر حمایت و استفاده می‌کنند که در ادامه لیست برخی از آنها آورده می شود:

- Google
- IBM
- Redhat
- Openstack
- Canonical
- Openshift
- Kubernetes
- Microsoft
- Rancher
- Rackspace

۶. آموزش نصب داکر بر روی سیستم‌عامل لینوکس

برای نصب داکر بر روی سیستم‌عامل لینوکس می‌توان از روش‌ها مختلفی استفاده کرد اما ما از راحت‌ترین روش نصب داکر استفاده می‌کنیم.

در ابتدا که در جریان هستید داکر ریپوهای خود رو به روی ایران مسدود کرده و خطای ۴۰۳ به ما می‌دهد. برای گریز از این مشکل راه‌های مختلفی وجود دارد که یکی از راحت‌ترین راه‌ها استفاده از سرویس شکن می‌باشد. سایت داکر بر روی یکی از دامنه‌های زیرمجموعه خود در اینجا اسکریپتی قرار داده که همواره آخرین نسخه‌ی پایدار داکر را بر روی سیستم‌عامل‌های زیر نصب می‌کند:

- Centos
- Fedora
- Debian
- Ubuntu

البته که در همون آدرس اسکریپتی برای نصب نسخه‌ی تستی خود نیز ارائه کرده است که اصلا توصیه نمی‌شود تا نسخه‌ی تستی سرویس داکر نصب شود. اگر شما از یکی از این ۴ تا سیستم‌عامل استفاده می‌کنید اصلا نگران نباشید به راحتی می‌توانید بر روی سیستم‌عامل خود سرویس داکر را نصب کنید.

خوب ابتدا DNS سیستم‌عامل خود رو به آدرس سایت شکن تغییر بدید تا بدون مشکل از ریپوی داکر بتونید استفاده کنید. این موضوع برای گریز از تحریم داکر است و اگر خودتون سرویس پروکسی دارید نیازی به این کار نخواهید داشت. و البته که اگر سرور شما خارج از ایران باشد نیز نیاز به این کار نخواهد بود.

در ابتدا از کانفیگ قبلی خود پشتیبان تهیه کنید و بعد آدرس جدید رو وارد کنید.

گرفتن پشتیبان از کانفیگ قبلی با دستور زیر انجام می‌شود.

```
cat /etc/resolv.conf > resolv.conf.old
```

تغییر کانفیگ و استفاده از DNS Server جدید برای گریز از تحریم داکر

```
echo "nameserver 94.232.174.194" > /etc/resolv.conf
```

بعد از این مرحله شما می‌توانید به راحتی و بدون مشکل داکر رو بر روی سیستم‌عامل خود نصب کنید.

برای نصب داکر از این آدرس اسکریپت مربوط به نصب داکر رو دانلود کنید و بعد آن اسکریپت را ران کنید. دستورات زیر اقدامات لازم برای نصب را انجام خواهند داد.

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
```

بسته به سرعت اینترنتی که دارید یه زمانی به طول می‌انجامد و بعد از آن نصب به پایان می‌رسد. پس از اتمام نصب نسخه‌ی نصب شده سرویس داکر را به شما نمایش می‌دهد. با دستور زیر نیز می‌توانید نسخه‌ی نصب شده رو مشاهده کنید.

docker version / docker -version / docker --version

سرویس داکر تنها با دسترسی کاربر **root** راه‌اندازی و استفاده می‌شود. برای اینکه از کاربرهای با دسترسی پایین‌تر برای کار با سرویس داکر استفاده کنید دستور زیر را وارد کنید.

sudo usermod -aG docker your-user

نکته‌ی مهم اینکه بعد از اعمال دستور بالا کاربر با سطح دسترسی پایین‌تر امکان راه‌اندازی و مدیریت کانتینرها و سرویس‌های راه‌اندازی شده با داکر را خواهد داشت و باید دقت کرد که این امکان فقط برای کاربرهایی فراهم شود که مورد تایید هستند. البته این موضوع به صورت کلی در سرورهای عملیاتی مطرح هست. در این صفحه در مورد مخاطراتی که ممکن است این تغییر دسترسی ایجاد کند صحبت شده است.

بر روی توزیع‌های مبتنی بر **deb** بعد از نصب سرویس داکر راه‌اندازی می‌شود اما در سرویس‌های مبتی بر **rpm** نیاز است که سرویس داکر را راه‌اندازی کنید. برای این کار از دستورات زیر می‌توانید استفاده کنید.

systemctl start docker

systemctl status docker

با دستور زیر همواره بعد از راه‌اندازی سیستم عامل سرویس داکر راه‌اندازی خواهد شد.

```
systemctl enable docker
```

برای به‌روز رسانی دیگر نیاز به راه‌اندازی این اسکریپت نیست و ریپوی داکر داخل سیستم‌عامل شما قرار داده شده است و با به روزرسانی داکر یا سیستم‌عامل خودتون اگر به روز رسانی برای داکر آماده شده باشد آن را نیز دریافت خواهید کرد.

نصب بر روی سیستم‌عامل Arch Linux

اسکریپتی که در بالا در مورد آن توضیح داده شده بر روی توزیع لینوکس **Arch** کارایی ندارد اما ریپوی این سیستم‌عامل همواره به روز بوده و تقریبا آخرین نسخه‌ی پایدار سرویس داکر رو می‌توان با دستور زیر نصب کرد. در این لینک نیز توضیحات کاملی برای نصب سرویس داکر در این سیستم‌عامل قرار داده شده است.

```
pacman -Syyu docker
```

همانند دیگر توزیع‌های لینوکس مبتنی بر **systemd** سرویس داکر راه‌اندازی خواهد شد.

```
systemctl start docker
systemctl status docker
```

❖ **نحوه نصب داکر توسط خودم**

مراحل نصب

مراحل نصب داکر بر روی لینوکس اوبونتو بصورت زیر می‌باشد.

در ابتدا که در جریان هستید داکر ریپوهای خود رو به روی ایران مسدود کرده و خطای ۴۰۳ به ما می‌دهد. برای گریز از این مشکل راه‌های مختلفی وجود دارد که یکی از راحت‌ترین راه‌ها استفاده از سرویس شکن (سایت شکن) می‌باشد.

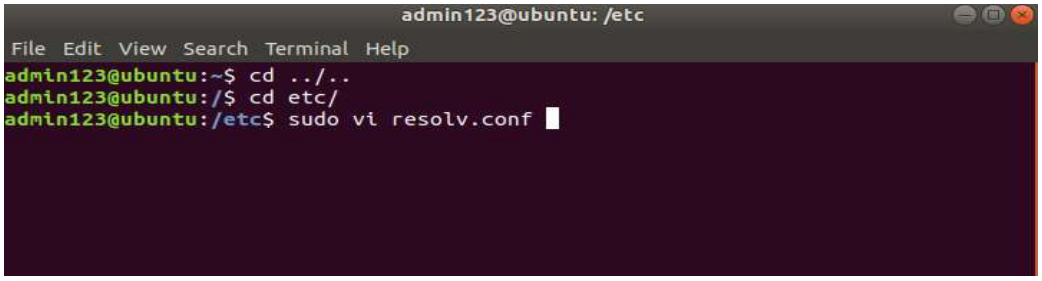
خوب ابتدا DNS سیستم‌عامل خود رو به آدرس سایت شکن تغییر بدید تا بدون مشکل از ریپوی داکر بتونید استفاده کنید. این موضوع برای گریز از تحریم داکر است و اگر خودتون سرویس پروکسی دارید نیازی به این کار نخواهید داشت. و البته که اگر سرور شما خارج از ایران باشید نیز نیاز به این کار نخواهد بود.

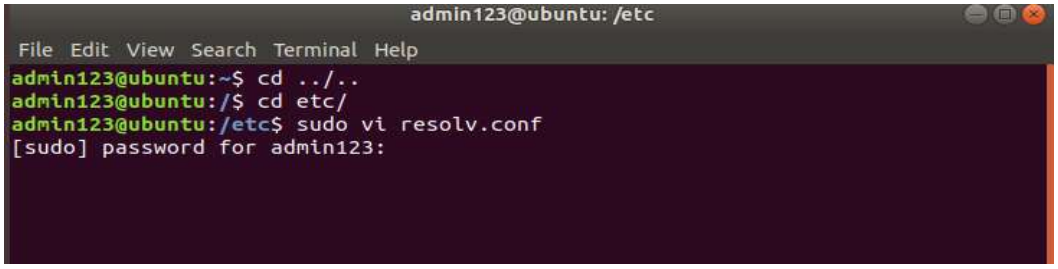
تغییر DNS در لینوکس

در توزیع اوبونتو، تغییر DNS در این محیط به گونه‌ای است که متن موجود در یک فایل کانفیگ را تغییر داده و ذخیره می‌کنیم.

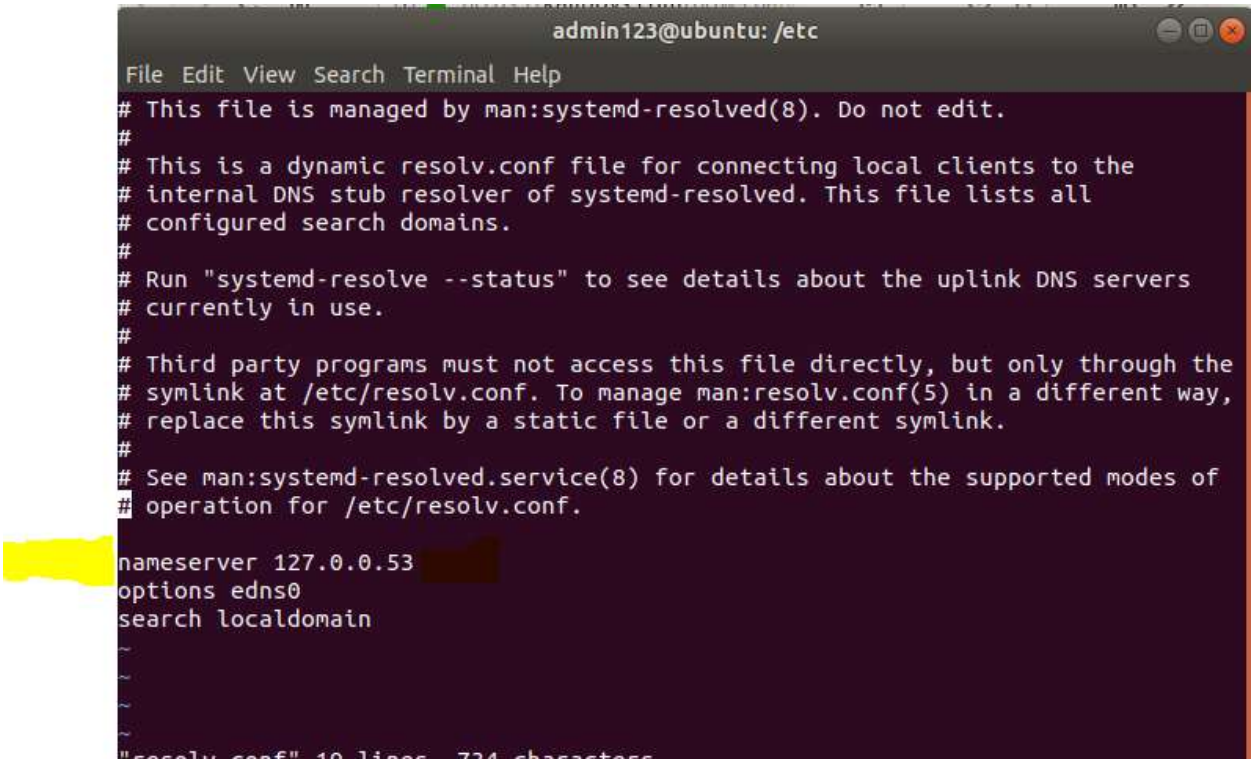
برای این کار باید فایل resolve.conf را ویرایش کنیم که در مسیر /etc/resolve.conf قرار دارد.

و برای ویرایش فایل موردنظر با استفاده از خط فرمان یا همان Terminal از ویرایشگر vi یا nano یا سایر ویرایشگر ها استفاده می کنیم و بعد از اتمام تغییرات، فایل را ذخیره می کنیم.





این فایل که باز شد



ما باید به قسمت مشخص شده مشخصات DNS مربوط به شکن را نیز به آن اضافه کنیم(برای استفاده از شکن در هر شبکه‌ای، کافیه DNS را روی این IP ها تنظیم کنید۱۷۸,۲۲,۱۲۲,۱۰۰ و ۱۸۵,۵۱,۲۰۰,۲ که مربوط به شکن است)

در سایت docker.ir نیز گفته بود Acquire::http::Proxy "http://Proxy.Docker.ir:5555"; را می‌توان اضافه کرد. من اضافه کردم.

کار با ویرایشگر vi

با دستور vi filename اگر فایلی به نام filename در دایرکتوری فعلی موجود باشد ، ادیتور آن را برای شما باز می کند و اگر فایلی با این نام موجود نباشد ، با زدن این دستور فایل مورد نظر برای شما ساخته می شود و امکان ویرایش آن را دارید.

اگر میخواهید فایل را در حالت Read Only باز کنید می‌توانید از سویچ R در دستور بالا استفاده کنید ، یعنی به این صورت:

vi -R filename

ویرایش فایل در vi :

پس از باز شدن محیط ویرایشگر vi شما با دستورات زیر می‌توانید تغییرات بدهید، کافی است در محیط ادیتور vi کلیدهای زیر را بزنید:

* با زدن دکمه i روی صفحه کلید وارد insert mode می شوید و شما می توانید نوشته را ویرایش کنید.

* با زدن x شما می‌توانید کارکتر را حذف کنید.

* با زدن escape از حالت insert mode خارج می شوید.

خارج شدن از محیط ویرایشگر vi :

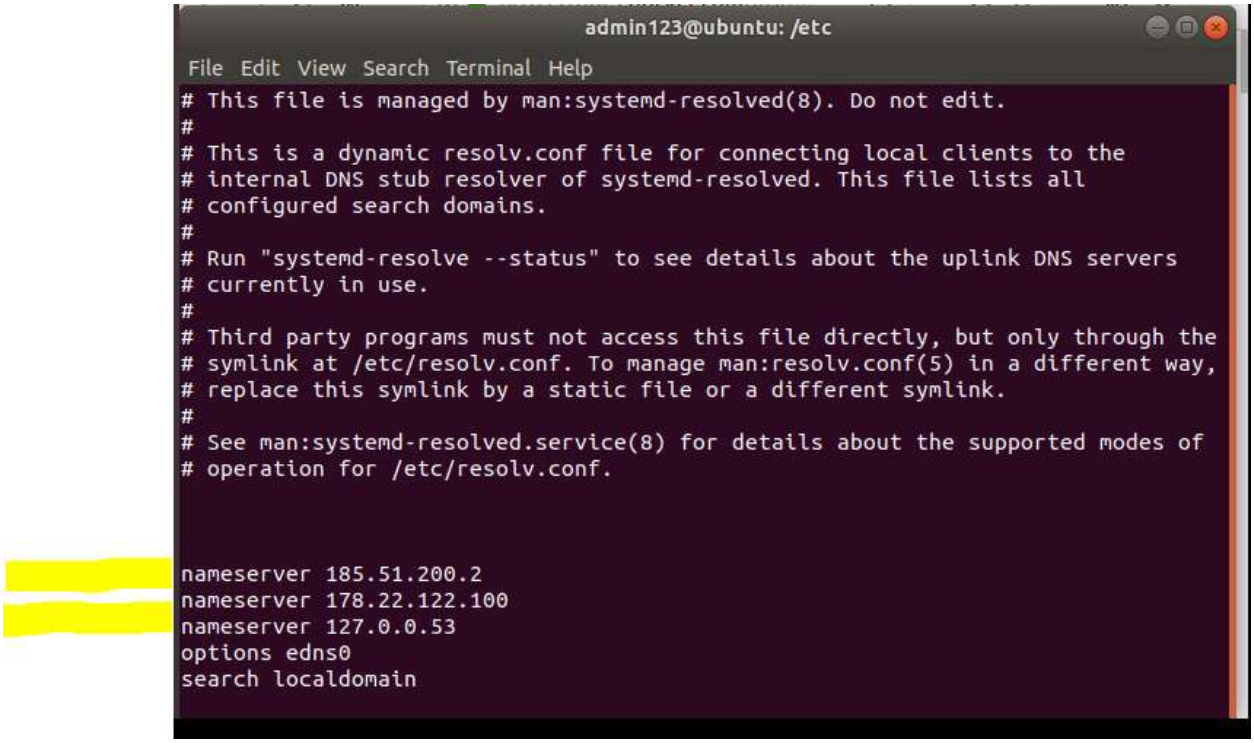
باید ابتدا دکمه esc را بزنیم تا از حالت insert mode خارج شود و سپس :

برای خروج از ویرایشگر vi بدون ذخیره کردن تغییرات از دستور زیر استفاده کنید

q!:

برای خروج و ذخیره تغییرات از دستور زیر استفاده کنید

wq:



سپس دستورات مربوط به نصب داکر که در سایت

https://phoenixnap.com/kb/how-to-install-docker-on-ubuntu-18-04

موجود است که بصورت زیر می‌باشد:

Prerequisites

- Ubuntu 18.04 64-bit operating system
- A user account with **sudo** privileges
- Command line / terminal (**CTRL-ALT-T or Applications menu > Accessories > Terminal**)
- Docker software repositories (optional)

Install Docker on Ubuntu Using Default Repositories

Step 1: Update Software Repositories

As usual, it's a good idea to update the local database of software to make sure you've got access to the latest revisions.

Therefore, open a terminal window and type:

```
sudo apt-get update
```

Allow the operation to complete.

Step 2: Uninstall Old Versions of Docker

Next, it's recommended to uninstall any old Docker software before proceeding.

Use the command:

```
sudo apt-get remove docker docker-engine docker.io
```

Step 3: Install Docker

To install Docker on Ubuntu, in the terminal window enter the command:

```
sudo apt install docker.io
```

Step 4: Start and Automate Docker

The Docker service needs to be setup to run at startup. To do so, type in each command followed by enter:

```
sudo systemctl start docker
sudo systemctl enable docker
```

```
example@example: ~
File  Edit  View  Search  Terminal  Help
example@example:~$ sudo systemctl start docker
example@example:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd
/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
example@example:~$
```

Step 5 (Optional): Check Docker Version

To verify the installed Docker version number, enter:

```
docker --version
```

```
example@example: ~
File  Edit  View  Search  Terminal  Help
example@example:~$ docker --version
Docker version 18.09.5, build e8ff056
```

Alternative: Install Docker from Official Repository

نصب به دو صورت **Default Repositories** و **Official Repository** است که ما مورد اول را نصب کردیم و نحوه نصب مورد دوم نیز در سایت قرار دارد که یکسری مراحل اضافه‌تری دارد.

از دستورات زیر برای اینکه ببینیم داکر نصب شده است استفاده می‌کنیم

Docker version

Docker ps

پس از اینکه داکر نصب شد باید برای استفاده از ایمیج‌ها مراحل زیر را انجام دهیم تا بتوانیم image ها را pull کنیم وگرنه بخاطر تحریم بودن اینکارها انجام نمی‌شود.

ابتدا توسط دستور زیر دایرکتوری را ایجاد نمایید :

mkdir -p /etc/systemd/system/docker.service.d/

و یا به دایرکتوری cd /etc/systemd/system/ رفته و در آنجا دستور زیر را اجرا می‌کنیم تا دایرکتوری جدید ایجاد شود.

mkdir -p docker.service.d

برای استفاده از این سرویس لازم است به مسیر زیر رفته و

Cd /etc/systemd/system/docker.service.d/

در اینجا یک فایل با نام http-proxy.conf بصورت زیر درست کنیم:

```
Sudo vi http-proxy.conf
```

حال فایل خالی ایجاد شده و با استفاده از دستورات vi قسمت زیر را به فایل اضافه می‌کنیم

```
[Service]
Environment="HTTP_PROXY=http://Proxy.Docker.ir:5555"
```

(این قسمت یک پراکسی در سایت docker.ir است. احتمالا در اینجا هم می توان از آی پی_های شکن استفاده کرد.)

این فایل را ذخیره می‌کنیم(:wq)
(لازم به ذکر است که این فایل احتمالاً از پیش وجود نداشته و باید فایل جدید با محتویات بالا ایجاد شود.)

سپس سرویس داکر را ریستارت می‌نماییم:

```
systemctl daemon-reload
systemctl restart docker
```

برای تست می‌توانید از ایمیج زیر استفاده نمایید:

```
docker run hello-world
```

اگر این ایمیج وجود نداشت در این صورت بصورت اتومات دانلود می‌شود.

ما ایمیج portainer را نیز بعد از انجام این مراحل بصورت زیر نصب کردیم

```
Sudo docker volume create portainer_data

Sudo docker run -d -p 9000:9000 -p 8000:8000 --name portainer --restart always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer
```

این دستورات باعث می‌شود که ایمیج portainer دانلود شود و بصورت container در داکر اجرا شود.

ما اگر قبل و بعد از نصب portainer دستور sudo docker ps را اجرا کنیم مشاهده می‌کنیم که بعد از نصب پوینتر در حال اجرا است.

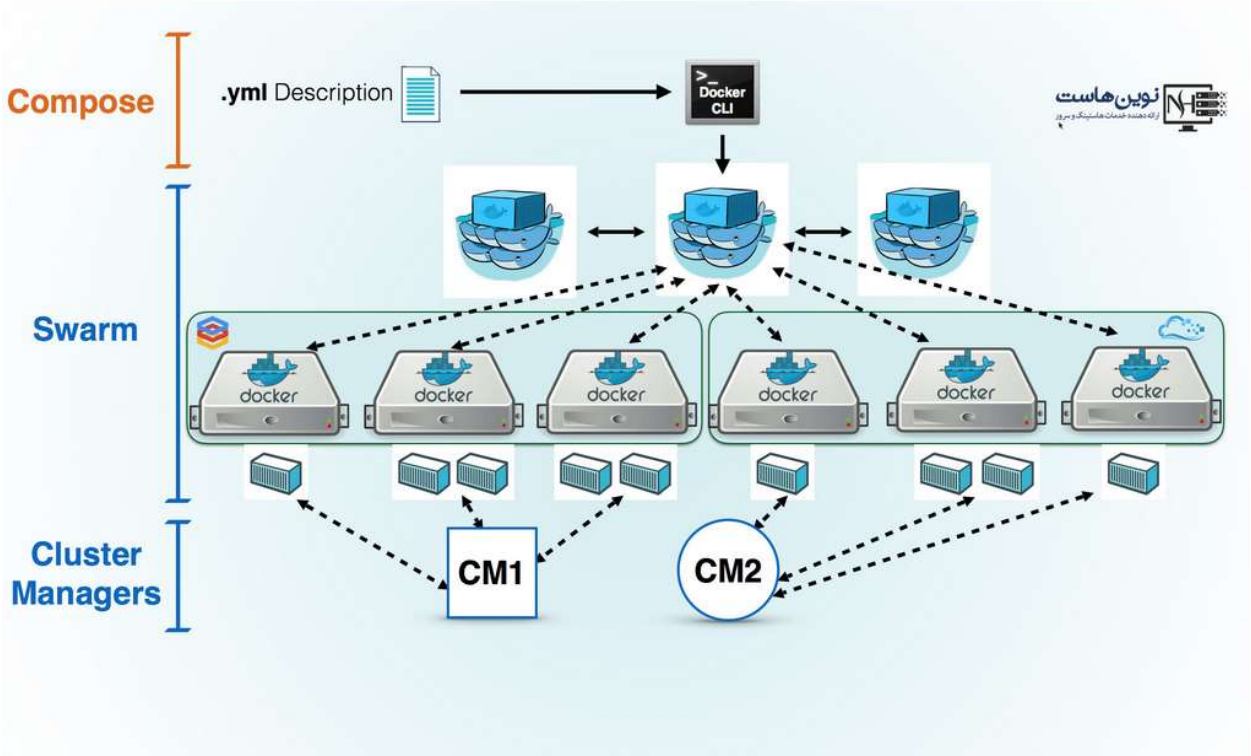
(احتمالا دستورات دیگری نیز برای تبدیل شدن portainer به container وجود دارد

sudo docker run -d -p 9000:9000 portainer/portainer

)

سپس مرورگر فایرفاکس را باز می‌کنیم و به ادرس آی پی http://127.0.0.1:9000 می‌رویم و ثبت نام می‌کنیم و بصورت گرافیکی برخی از کنترل‌ها را روی container ها انجام می‌دهیم.

۷. آشنایی با ابزارهای خط فرمان داکر Docker



داکر Docker چند ابزار خط فرمان (CLI) مختلف معرفی کرده که ادمین می‌تواند با کمک این ابزارها نهایت استفاده بهینه از پلتفرم داکر را تجربه کند.

این ابزارها شامل Engine CLI یا خط فرمان اصلی داکر، Compose CLI که برای ساخت اپلیکیشن‌های متشکل از چند کانتینر به کار میرود، Machine CLI که برای ایجاد و مدیریت ماشین های مجازی (virtual box, vmware) جهت پیاده سازی داکر ساخته شده، UCP CLI یا Universal Control Plane که برای مدیریت یکپارچه کلاسترها و swarm ساخته شده و نهایتا DTR CLI که برای مدیریت رجیستری داکر (Docker Trusted Registry) طراحی شده است.

در ادامه به بررسی اجمالی این ابزارها میپردازیم:

❖ ۱-Engine CLI:

دستوراتی که در خط فرمان با **docker** شروع می‌شود جزو این ابزار هستند, سرویس **dockerd** نیز برای سرویس دهی به این ابزار اجرا می‌شود.

دستور **docker help** راهنمای کل دستورات ممکن این ابزار را برای شما چاپ خواهد کرد.

معمولا لازم است دستورات داکر را با **sudo** شروع کنید, دستورات **docker pull** و **docker run** مهم ترین و کاربردی ترین دستورات داکر هستند که به ترتیب برای دانلود ایمیج سیستم عامل یا محیط مربوطه, و اجرای آن تحت یک کانتینر مورد استفاده قرار دارند.

دستور **docker build** نیز کمک می‌کند دایکتوری جاری را به یک ایمیج کانتینتر قابل انتقال تبدیل نمایید، که البته لازم است یک فایل کانفیگ تحت عنوان **DockerFile** ایجاد کنید.

دستورات **Docker pause** و **Docker unpause** نیز مشابه وضعیت **hibernate** در ویندوز یا **suspend** در **vmware** است که کانتینر را با حفظ وضعیت جاری تعلیق می‌کند.

دستور جالب **docker diff CONTAINER** تمام تغییرات اعمال شده روی فایل سیستم کانتینر از زمان ایجاد را برای شما لیست می‌کند که برای تحلیل, مهندسی معکوس و عیب‌یابی برنامه‌ها بسیار بسیار یاری رسان خواهد بود.

❖ ۲-Compose CLI :

ابزار **Compose** به شما این امکان را می‌دهد که اپلیکیشن‌هایی متشکل از چند کانتینر و **Volumes** را به یکدیگر متصل کنید.

به طور مثال می‌توانید وردپرس را روی یک کانتینر و دیتابیس **MySQL** را روی سرور دیگر نصب کنید و به کمک شبکه داخلی داکر به یکدیگر متصل کنید.

کافیست کانفیگ شبکه مورد نظر را در فایل **docker-compose.yml** تنظیم کرده و سپس با دستور

docker-compose up مجموعه چند کانتینری را برپا کنید.

❖ ۳-Machine CLI :

و اینجاست که جذابیت داکر دوچندان می‌شود; نیازی نیست برای هر سرور مجازی یک به یک داکر نصب کنید و کانفیگ‌های لازم را انجام دهید

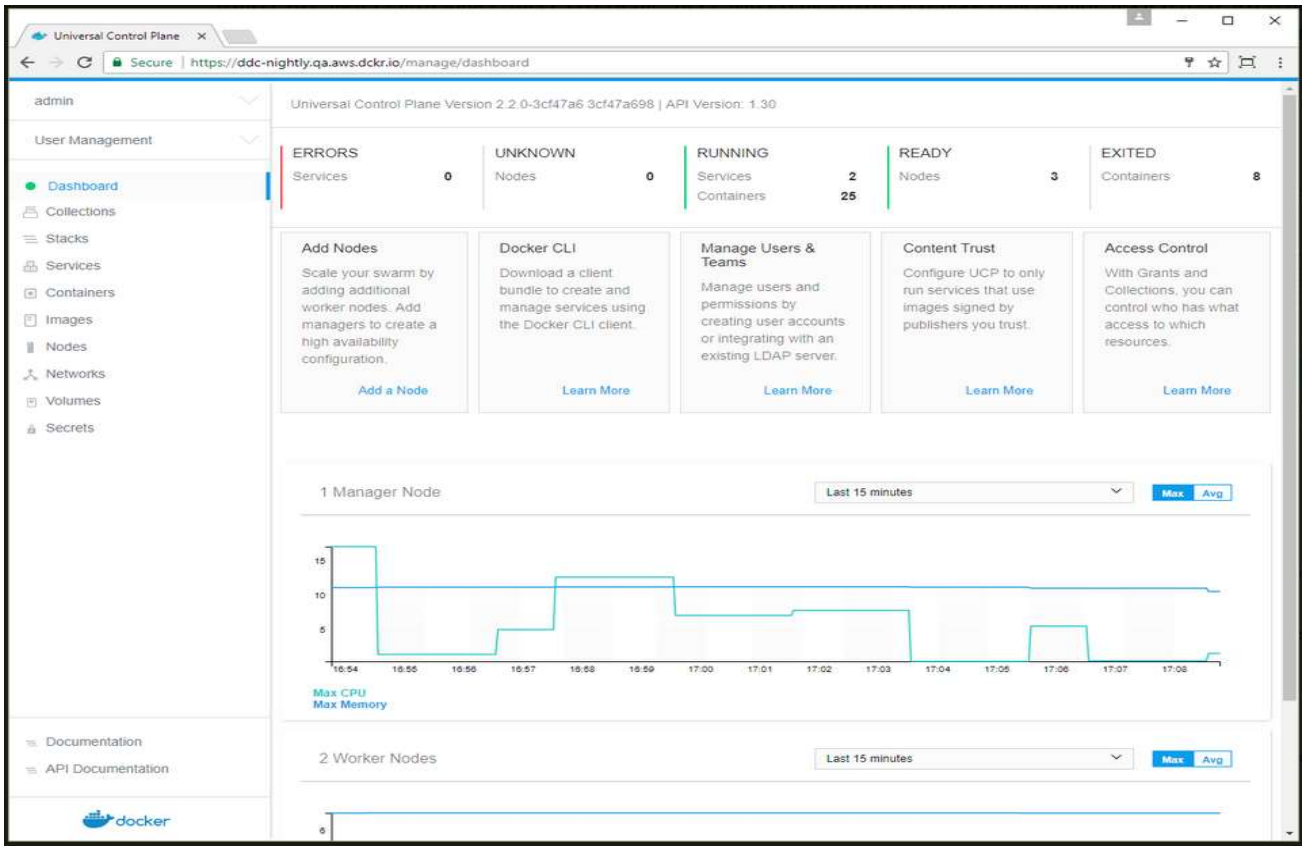
کافیست Docker Machine را به پلتفرم مجازی ساز (Digital ocean, Virtualbox, Amazon, Vmware) متصل کنید و با دستور Docker create دستور ماشین مجازی را صادر کنید, داکر ماشین را برای شما نصب و آماده سازی کرده و دسترسی SSH را در اختیارتان قرار خواهد داد, دستورات SSH, Start, Stop, SCP نیز برای مدیریت و دسترسی به ماشین ایجاد شده در دسترس خواهد بود.

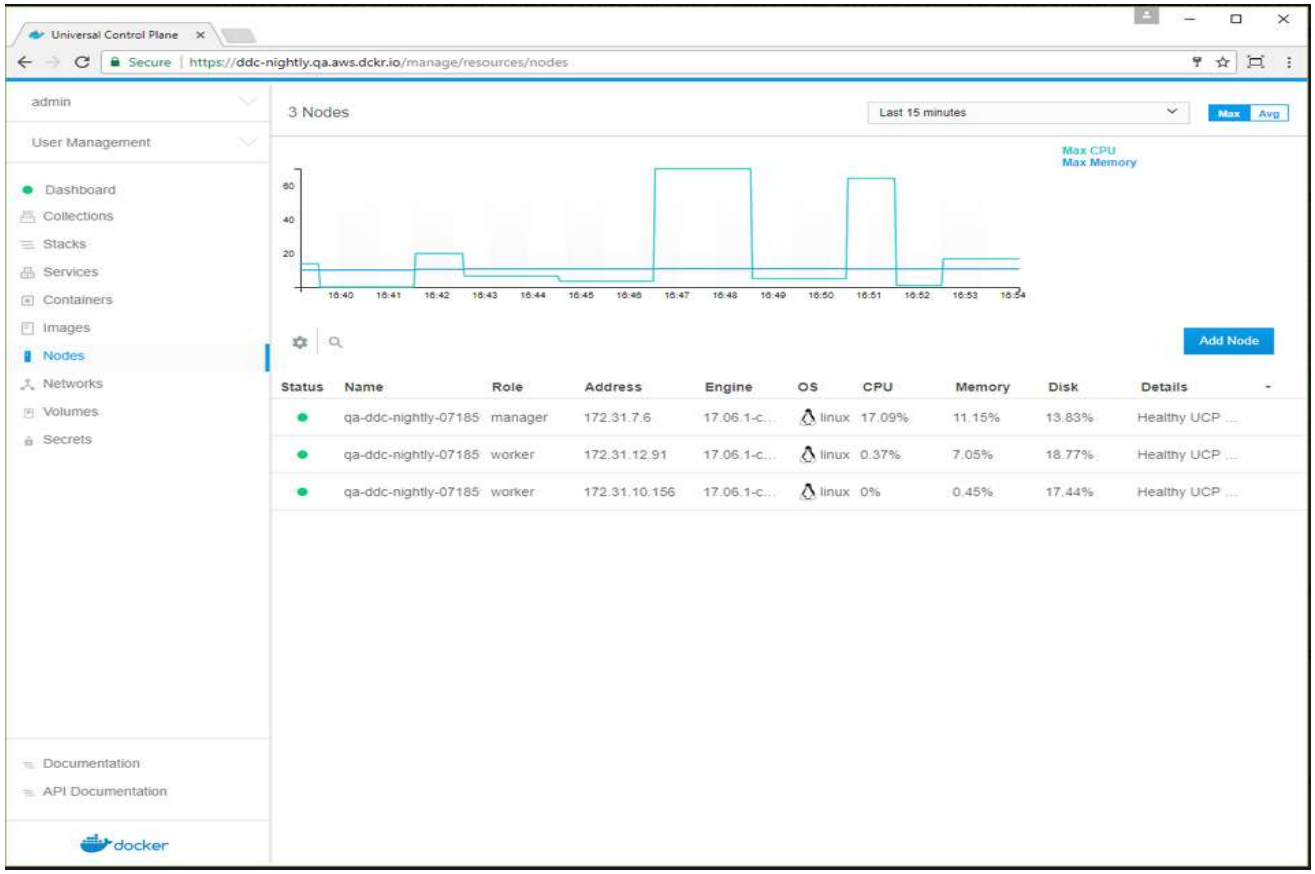
❖ ۴-UCP CLI :

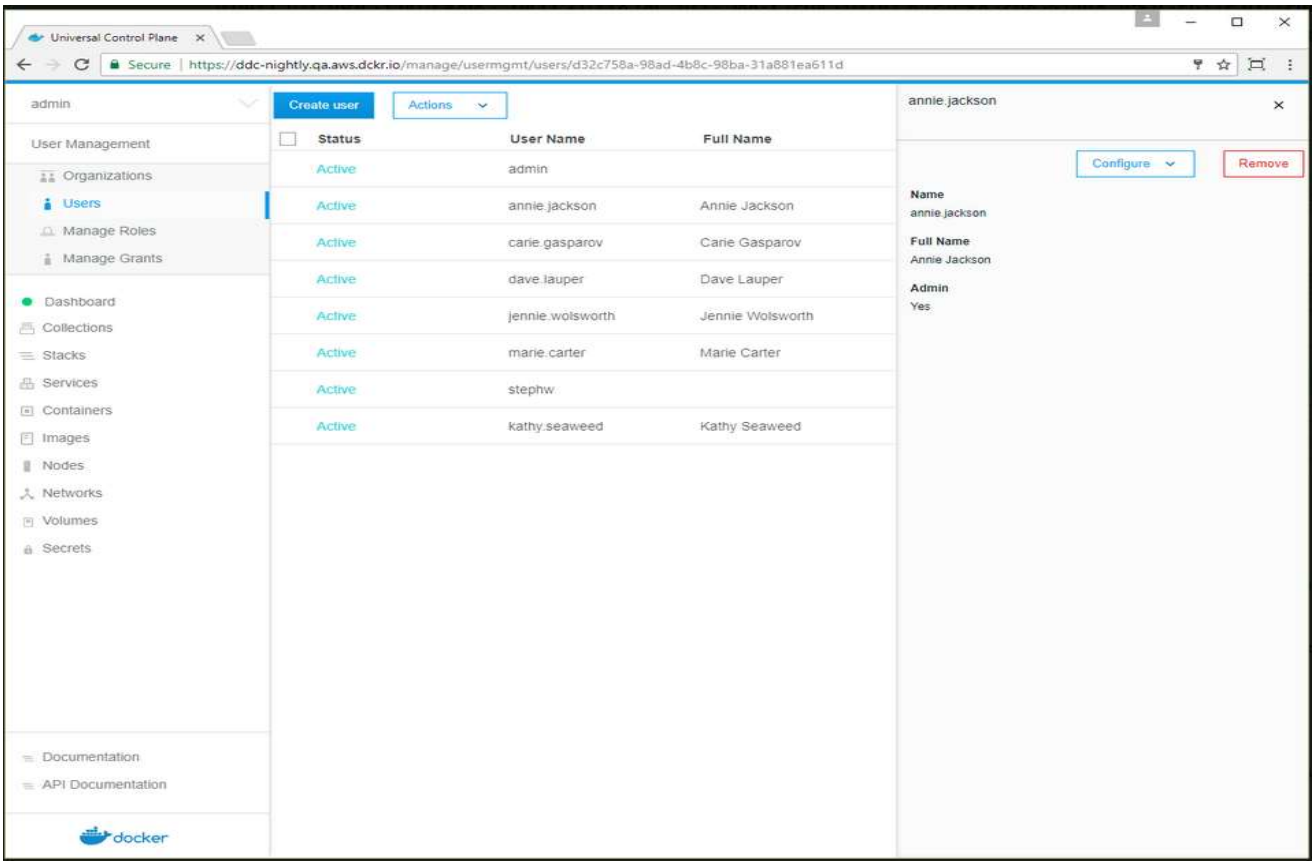
تا اینجا متوجه شدیم که داکر ابزارهای زیادی برای ایجاد ماشین‌های مجازی و کانتینترهای گروهی دارد, اما مدیریت دستی همه این‌ها در ابعاد بزرگ بسیار طاقت فرسا خواهد بود, لذا داکر Universal Control Plane یا UCP را معرفی کرده که یک پنل تحت وب عالی برای مدیریت swarm ها یا دسته‌های بزرگ کانتینر فراهم می‌کند.

همچنین می‌توانید وضعیت عملکرد کانتینترها، مصرف منابع و تغییرات ایجاد شده را به سادگی مانیتور کنید.

ابزار Docker Swarm: وقتی اپلیکیشن شما بزرگ میشه شما ممکنه بخواین که چند تا container داشته باشین تا traffic شما رو هندل کنند. این ابزار باعث میشه مدیریت گروهی از containerها (clusters) در یک زمان، کار ساده تری بشه.




Universal Control Plane
Secure | https://ddc-nightly.qa.aws.dckr.io/manage/resources/nodes
admin
User Management
Dashboard
Collections
Stacks
Services
Containers
Images
Nodes
Networks
Volumes
Secrets
Documentation
API Documentation
docker
3 Nodes
Last 15 minutes
Max
Avg
Max CPU
Max Memory
Add Node
Status Name Role Address Engine OS CPU Memory Disk Details
qa-ddc-nightly-07185 manager 172.31.7.6 17.06.1-c... linux 17.09% 11.15% 13.83% Healthy UCP ...
qa-ddc-nightly-07185 worker 172.31.12.91 17.06.1-c... linux 0.37% 7.05% 18.77% Healthy UCP ...
qa-ddc-nightly-07185 worker 172.31.10.156 17.06.1-c... linux 0% 0.45% 17.44% Healthy UCP ...


Universal Control Plane
Secure | https://ddc-nightly.qa.aws.dckr.io/manage/usermgmt/users/d32c758a-98ad-4b8c-98ba-31a881ea611d
admin
User Management
Organizations
Users
Manage Roles
Manage Grants
Dashboard
Collections
Stacks
Services
Containers
Images
Nodes
Networks
Volumes
Secrets
Documentation
API Documentation
docker
Create user
Actions
Status User Name Full Name
Active admin
Active annie.jackson Annie Jackson
Active carie.gasparov Carie Gasparov
Active dave.lauper Dave Lauper
Active jennie.wolsworth Jennie Wolsworth
Active marie.carter Marie Carter
Active stephw
Active kathy.seaweed Kathy Seaweed
annie.jackson
Configure
Remove
Name
annie.jackson
Full Name
Annie Jackson
Admin
Yes

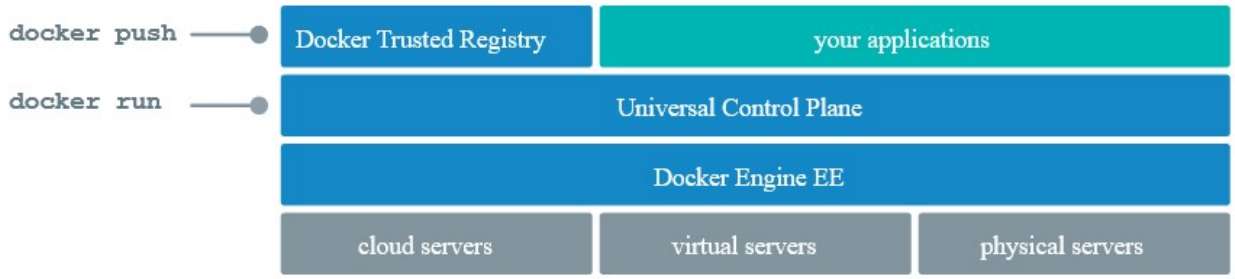
❖ ۵-DTR CLI :

DTR یا Docker Trusted Registry یکی از جذاب‌ترین ابزارهای داکر برای شرکت‌هاست.

با راه اندازی DTR می‌توانید ایمیج‌هایی از کانتینرهای مورد نظر را بر روی سرور محلی قرار داده و دسترسی سرورها را به این رجیستری محدود کنید.

عملیات تهیه بکاپ, و رپلیکا (نسخه موازی) توسط DTR CLI قابل مدیریت است و دسترسی‌ها و ارتباط‌ها همه با certificate های امنیتی انجام می‌پذیرد.

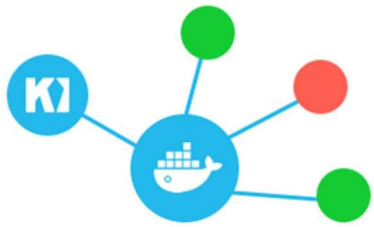
ایجاد کش نیز از قابلیت‌های جالب رجیستری داکر است که کمک می‌کند ایمیج ها را از نزدیک‌ترین سرور دریافت کنید.



۸. ابزارهای جانبی کنار داکر

❖ ۱-Kitematic

با استفاده از این ابزار به راحتی می‌توان با سرویس‌دهنده‌ی داکر ارتباط گرفت و تمام مواردی که در Docker Client با استفاده از دستور (Command) به سرویس‌دهنده‌ی داکر ارسال می‌شد با استفاده از GUI بسیار زیبای این نرم‌افزار انجام داد. بسیار رابط کاربری عالی داشته و به خوبی می‌شود با این نرم‌افزار تمام اتفاقاتی که سمت Client باید انجام شود را بررسی و مدیریت کرد. این ابزار ابتدا برای استفاده در سیستم عامل مک توسعه داده شده بود که بعد از آن برای سیستم‌عامل‌های ویندوز و لینوکس نیز توسعه داده شد. به صورت پیش‌فرض به مخزن Docker Hub نیز متصل می‌شود و با استفاده از رابط کاربری آن می‌توان در این مخزن ایمیج‌های مورد نظر خود را بررسی و از آنها استفاده کرد. برخی از قابلیت‌های مهم برای اجرا کردن کانتینرها که شما باید در رابط خط فرمان (CLI) باید به آنها توجه کنید در این ابزار به خوبی و سادگی با استفاده از واسط گرافیکی کاربر (GUI) برای شما محیا شده است. به عنوان مثال تغییر و تنظیم والیوم‌های داخل کانتینر، مشاهده‌ی لاگ کانتینر به صورت زنده، ایجاد و تغییر پورت‌ها و. ...



پیشنهاد من این است که در ابتدا برای یادگیری بهتر و آشنایی بیشتر با سرویس داکر و دستورات مورد استفاده در آن از این ابزار استفاده کنید تا مشاهده‌ی دقیق‌تری از اتفاقات صورت گرفته داشته باشید و بعد از اینکه به دستورات سرویس داکر مسلط شدید دیگر نیازی به استفاده از Kitematic ندارید.

❖ ۲-Portainer

این ابزار به خوبی یک GUI کامل در اختیار ما قرار می‌دهد. اما با این تفاوت که دیگر نیاز نیست حتما بر روی همان کامپیوتر سرویس دهنده داکر نصب شود. بلکه می‌تواند به سرویس دهنده‌های مختلف داکر متصل شود. این ابزار این امکان را دارد که به چند تا سرویس دهنده داکر متصل شود. با استفاده از این ابزار می‌توان به خوبی سرویس‌های بزرگ را پیاده سازی و از آنها استفاده کرد. در ضمن Portainer بر روی وب در دسترس می‌باشد از این رو می‌توان آن را بر روی سرورهایی که دارای UI و

Desktop نیستند به خوبی کاربرد دارد.

همانند Kitematic اصلا توصیه نمی‌شود که در سرویس پروداکشن از پنل استفاده شود.

portainer یک محیط تحت وب بر مبنای خود داکر است که برای مدیریت سرورهای داکر طراحی شده, پرتینر خروجی متنی محیط خط فرمان داکر را به قالب گرافیکی و قابل کلیک کردن تبدیل می‌کند. ایجاد ارتباط بین شبکه‌ها, مدیریت Volume ها و کانتینرها, مدیریت و نظارت بر مصرف منابع, و گرافیکی کردن امور دریافت ایمیج‌ها و راه اندازی کانتینرها از قبلیت‌های عمده این ابزار است.

آموزش نصب portainer برای مدیریت تحت وب داکر

نصب portainer از طریق خود داکر انجام می‌پذیرد, در واقع نیازی به نصب نیست بلکه کافیست پورتینر را pull و اجرا کنید:

```
sudo docker run -d -p 9000:9000 portainer/portainer
```

با اینکار ایمیج portainer از dockerhub دریافت شده و از طریق پورت ۹۰۰۰ سرور در دسترس خواهد بود.

من خودم بصورت زیر نصب کردم:

پس از اینکه داکر نصب شد باید برای استفاده از ایمیج‌ها مراحل زیر را انجام دهیم تا بتوانیم image ها را pull کنیم وگرنه بخاطر تحریم بودن اینکارها انجام نمی‌شود.

ابتدا توسط دستور زیر دایرکتوری را ایجاد نمایید:

```
mkdir -p /etc/systemd/system/docker.service.d/
```

و یا به دایرکتوری cd /etc/systemd/system/ رفته و در آنجا دستور زیر را اجرا می‌کنیم تا دایرکتوری جدید ایجاد شود.

```
mkdir -p  docker.service.d
```

برای استفاده از این سرویس لازم است به مسیر زیر رفته و

```
Cd /etc/systemd/system/docker.service.d/
```

در اینجا یک فایل با نام http-proxy.conf بصورت زیر درست کنیم:

```
Sudo vi http-proxy.conf
```

حال فایل خالی ایجاد شده و با استفاده از دستورات vi قسمت زیر را به فایل اضافه می‌کنیم

```
[Service]
Environment="HTTP_PROXY=http://Proxy.Docker.ir:5555"
```

(این قسمت یک پراکسی در سایت docker.ir است. احتمالا در اینجا هم می توان از آی پی_های شکن استفاده کرد.)

این فایل را ذخیره می‌کنیم(:wq)

(لازم به ذکر است که این فایل احتمالاً از پیش وجود نداشته و باید فایل جدید با محتویات بالا ایجاد شود.)

سپس سرویس داکر را ریستارت می‌نماییم:

```
systemctl daemon-reload
systemctl restart docker
```

برای تست می‌توانید از ایمیج زیر استفاده نمایید:

```
docker run hello-world
```

اگر این ایمیج وجود نداشت در این صورت بصورت اتومات دانلود می‌شود.

ما ایمیج portainer را نیز بعد از انجام این مراحل بصورت زیر نصب کردیم

```
Sudo docker volume create portainer_data

Sudo docker run -d -p 9000:9000 -p 8000:8000 --name portainer --restart always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer
```

این دستورات باعث می‌شود که ایمیج portainer دانلود شود و بصورت container در داکر اجرا شود.

ما اگر قبل و بعد از نصب portainer دستور sudo docker ps را اجرا کنیم مشاهده می‌کنیم که بعد از نصب پوینتر در حال اجرا است.

(احتمالا دستورات دیگری نیز برای اجرای portainer وجود دارد

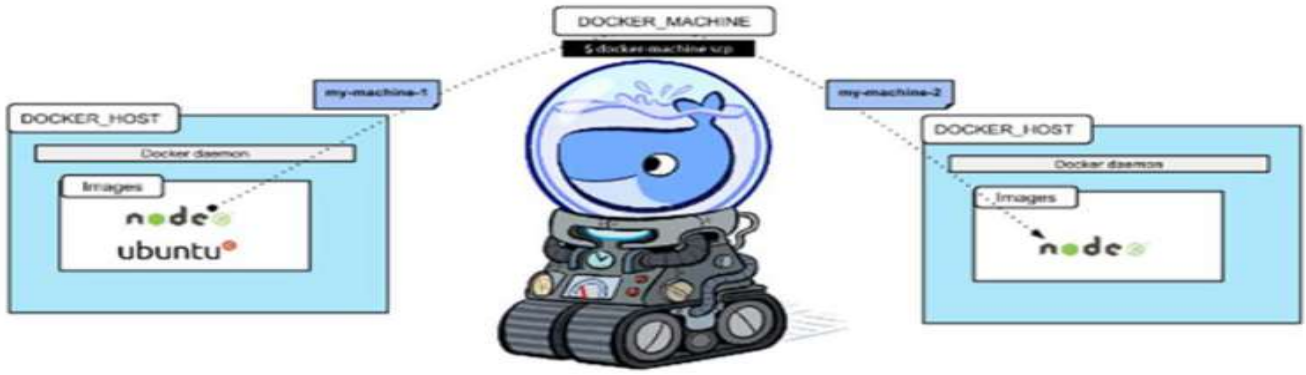
```
sudo docker run -d -p 9000:9000 portainer/portainer
```

)

سپس مرورگر فایرفاکس را باز می‌کنیم و به ادرس آی پی http://127.0.0.1:9000 می‌رویم و ثبت نام می‌کنیم.

❖ ۳-Docker Machine

این ابزار برای راه‌اندازی یه ماشین مخصوص کار با داکر می‌باشد. با استفاده از این ابزار شما یک ماشین مجازی VPS راه‌اندازی خواهد کرد و تمام موارد مربوط به داکر را در آن خواهید داشت. با استفاده از این ابزار می‌توانید ماشین‌های مختلف را مدیریت کرده و انواع اقدامات اعم از به روز رسانی، استارت، استوپ، شروع مجدد و اقدامات دیگر را بر روی ماشین‌ها انجام دهید. این ابزار واسطی بین داکر سرویس‌دهنده و محیط نصب آن با محیط کاری شما می‌باشد. برای زمان‌هایی که نیاز دارید با سرویس داکر کار کنید اما سیستم‌عاملی که دارید ویندوز یا مک می‌باشد این امکان را فراهم می‌کند که بتوانید سرویس داکر را داشته باشید. این ابزار همواره با docker toolbox نصب می‌شود. این ابزار برای انجام کارهای خود دارای یک خط فرمان مخصوص می‌باشد که با استفاده از آن می‌توان تمام عملکردهای لازم را انجام داد.

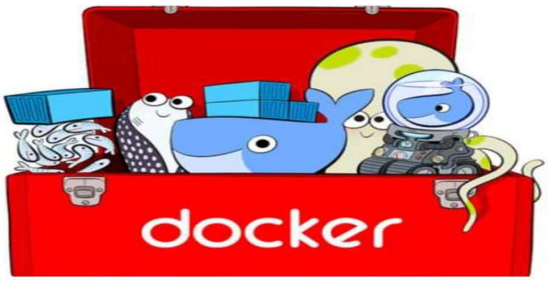


❖ Docker Toolbox-۴

این ابزار تجمیع تمام ابزارهایی که توضیح دادیم می‌باشد و تمام آنها را در خود دارد و با نصب آن بر روی ویندوز یا مک می‌توانید تمام ابزارها را یک‌جا داشته باشید. مواردی که در ابزار وجود دارد شامل:

- Docker Engine
- Docker Machine
- Docker Compose
- Kitematic
- VirtualBox

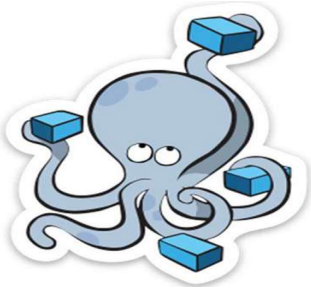
البته در نسخه‌های جدید به جای استفاده از Docker Toolbox از Docker Desktop استفاده می‌شود که با استفاده از Hyperv و ارتباط با آن امکان استفاده از داکر را در اختیار ما قرار می‌دهد.



❖ Docker Compose-۵

این ابزار که بسیار پر استفاده می‌باشد به شما کمک می‌کند که یک یا تعدادی از کانتینرها رو در قالب سرویس‌های مختلف راه‌اندازی کنید. با استفاده از این ابزار شما تمام کانفیگ‌ها و موارد مد نظر خود را در یک فایل متنی با فرمت Yaml قرار می‌دهید که باید به نحوه‌ی قرارگیری عبارات با فرمت مناسب آنها در کنار هم توجه کنید، سپس با استفاده از یک دستور تمام آن کانتیرها را راه‌اندازی و اجرا خواهید کرد. از مزایای استفاده از این ابزار اینه که شما یک بار تمام موارد مد نظر خود در راه‌اندازی سرویس

را مشخص می‌کنید سپس این فایل را در محل‌های مختلف اجرا می‌کنید. معمولا کامپوز فایل ایجاد شده امکان این را دارد که در ساختارهای مختلف مورد استفاده قرار گیرد.



نحوه نصب داکر کامپوز

امکان نصب داکر کامپوز بر روی سیستم عاملهای لینوکس، ویندوز و مک وجود دارد و این کار از سه طریق ممکن است: نصب مستقیم فایل باینری، نصب از طریق داکر (در ویندوز و مک) و یا نصب از طریق یک بسته نرم افزاری Python Pip .

برای نصب داکر کامپوز در لینوکس باید فایل باینری آن را از گیت هاب دانلود کرده و سپس به یک فایل قابل اجرا تبدیل کنید. در تکه کد زیر نحوه انجام این کار با استفاده از دستور curl نمایش داده شده است. توجه کنید که داکر کامپوز تنها بر روی نسخه های ۶۴ بیتی لینوکس قابل اجراست:

```
$ sudo curl -L https://github.com/docker/compose/releases/
download/1.8.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose
```

با دستورات بالا نتونستم نصب کنم.

فقط یک دایرکتوری بنام docker-compose در مسیر /usr/local/bin/ ایجاد کردم و دستور زیر را

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

اجرا کردم که مربوط حق دسترسی‌ها می‌باشد. سپس در این دایرکتوری یک فایل متنی با دستور vi با تمپلت داکر کامپوز به نام composeTest ایجاد کردم. حالا نمی‌دونم روال رو درست رفتم یا خیر.

❖ ۶-VirtualBox

این ابزار به ما کمک می‌کند که داخل سیستم‌عامل خودمون به صورت مجازی و کاملا مجزا یک یا چند سیستم‌عامل دیگر داشته باشید. این ابزار همانند Vmware Workstation فعالیت می‌کند. بسیار سبک‌تر بوده و با منابع کمتری این امکان را در اختیار ما قرار می‌دهد. به صورت متن باز و کاملا رایگان می باشد و در تقریبا در بیشتر سیستم‌عامل‌ها پشتیبانی می شود.

۹. نصب Docker ToolBox بر روی ویندوز

قبلا در این لینک در مورد نصب داکر با استفاده از پکیج بر روی ویندوز صحبت کردیم. در اینجا می‌خواهیم با استفاده از **Docker ToolBox** بر روی ویندوز داکر را نصب و از آن استفاده کنیم. نکته اینکه این روش در بیشتر نسخه‌های ویندوز پشتیبانی می‌شود.

نصب داکر بر روی ویندوز با استفاده از Docker ToolBox

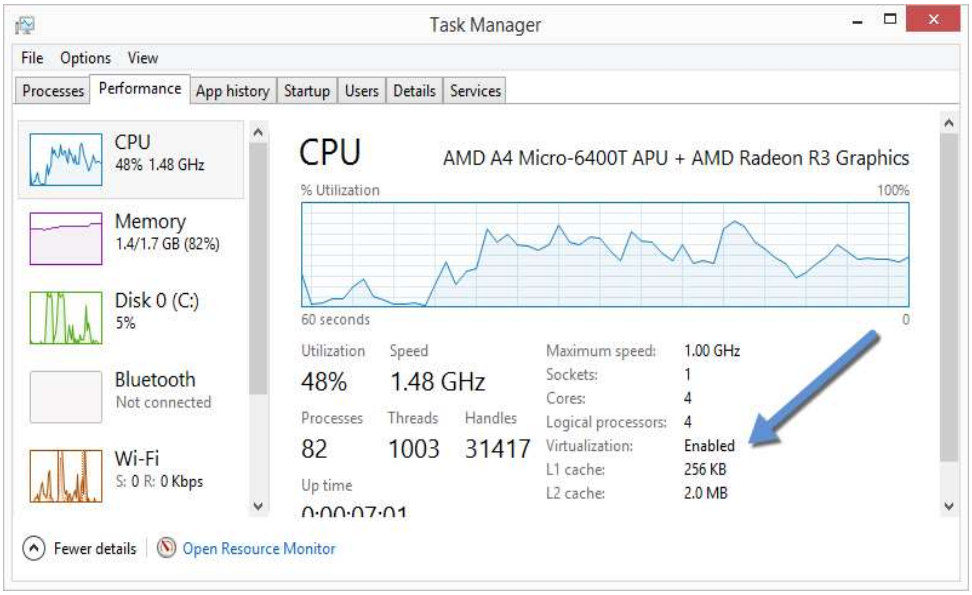
با این روش می‌توان بر روی سیستم‌عامل‌های ویندوز و مک سرویس داکر را نصب و از آن استفاده کرد. با نصب **ToolBox** تمام موارد زیر بر روی سیستم‌عامل شما نصب می‌شود:

- Docker Client
- Docker Machine
- Docker Compose
- Kitematic
- Oracle VirtualBox

با استفاده از داکر ماشین(**Docker Machine**) یک لینوکس ماشین سبک و کوچیک راه‌اندازی می‌شود که سرویس‌دهنده‌ی داکر درون آن قرار داره و با ارتباط با آن می‌توانید از سرویس داکر استفاده کنید.

داکر **toolbox** بر روی ویندوز ۷ و جدیدتر از آن و تنها نسخه‌ی ۶۴ بیتی راه‌اندازی می‌شود. یه نکته‌ی مهم دیگه اینکه باید حتما در بایوس کامپیوتر خودتون قابلیت مجازی‌سازی رو فعال کرده باشید تا بتونید از لینوکس مجازی که قبلا صحبت کردیم استفاده کنید.

به تصویر زیر دقت کنید. در این تصویر نشان می‌دهد که اگر قابلیت مجازی‌سازی روی کامپیوتر شما فعال باشد در task manager می‌توانید فعال بودن این امکان را مشاهده کنید.



بعد از فعال کردن قابلیت مجازی‌سازی، از اینجا ToolBox رو دانلود کنید. البته همان‌طور که قبلا توضیح داده شد داکر بر روی ایران لینک‌های خود را مسدود کرده است از این رو از تحریم گریز شکن استفاده کنید که نحوه‌ی استفاده از آن در اینجا کامل توضیح داده شده است.

{در ابتدا لازم است تا پکیج مورد نظر را دانلود کنید. همانطور که در جریان هستید داکر کلا روی ایران لینک‌های خود را مسدود کرده است. برای همین شما برای دانلود با خطای ۴۰۳ مواجه خواهید شد. برای رفع این موضوع می‌توان از روش‌های مختلفی استفاده کرد که در این لینک توضیح داده شده اما در ادامه از سایت شکن و سرویس رفع تحریم این سایت استفاده می‌کنیم. برای این کار اقدامات زیر را انجام دهید.

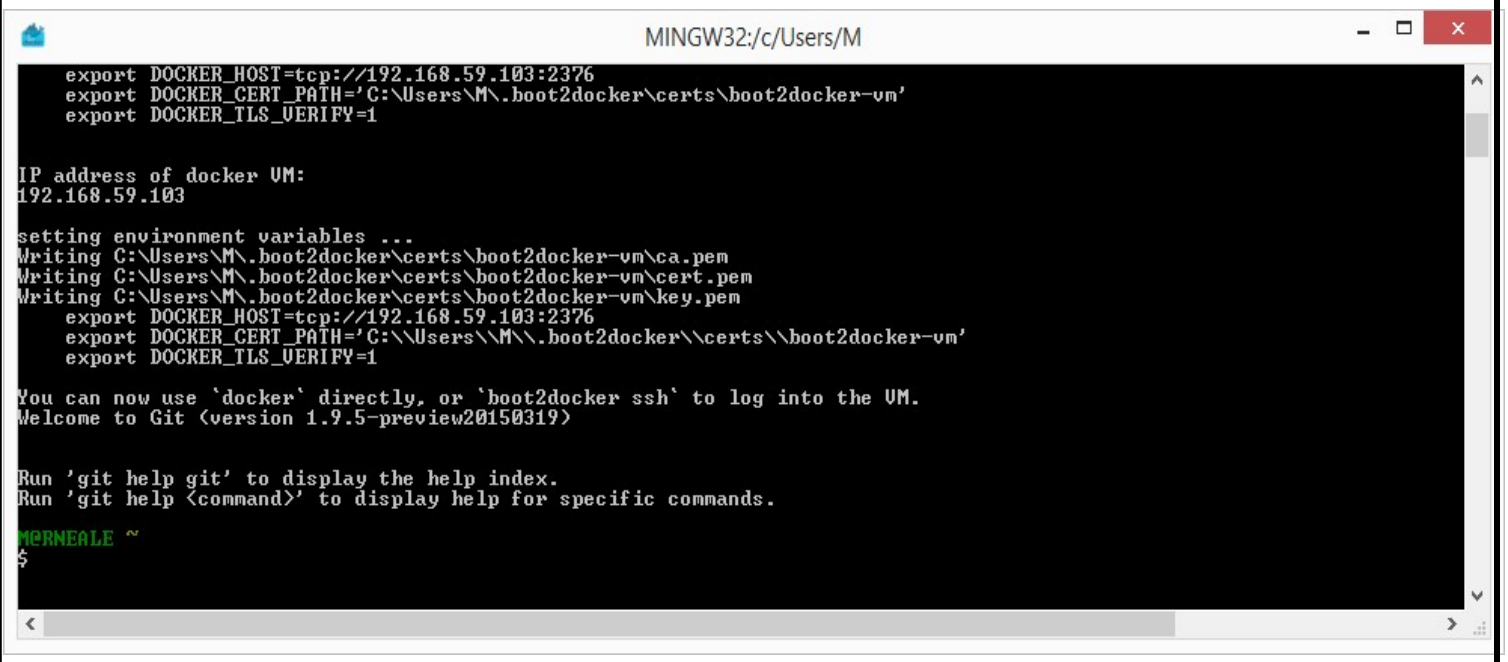
- ابتدا Control Panel را باز کنید.
- بر روی Network and Sharing Center کلیک کنید.
- در قسمت گوشه سمت چپ صفحه بر روی Change adapter settings کلیک کنید.
- سپس بر روی کارت شبکه خود راست کلیک کرده و گزینه‌ی Properties را انتخاب کنید.
- بعد از آن Internet Protocol Version 4 (TCP/IPv4) را از لیست باز شده انتخاب کرده و بر روی گزینه‌ی Properties کلیک کنید.
- بر روی گزینه‌ی Use the following DNS server addresses کلیک کنید و این IP را در خانه‌های آن وارد نمایید. ۹۴,۲۳۲,۱۷۴,۱۹۴ (آدرس آی‌پی مربوط به شکن که در سایتش موجود است)

- بعد از وارد کردن IP جدید بر روی گزینه‌ی OK کلیک کرده و سپس Close را انتخاب نمایید.

انتظار می‌رود بعد از این مراحل و تغییر DNS Server بتوانید از سایت داکر پکیج مورد نظر را دانلود کنید. } از فیلتر شکن هم می‌شود استفاده کرد.

پس از دانلود بر روی پکیج دانلود شده دوبار کلیک کرده تا نصب شود. موارد نصب را تا به انتها پیش ببرید. در انتها بعد از نصب کامل بر روی Desktop شما همانند تصویر زیر باید آیکن هر ۳ نرم‌افزار نصب شده قرار داشته باشد.

بعد از آن بر روی Docker QuickStart کلید کرده تا مراحل کانفیگ کردن Docker ToolBox طی شود. اگر قابلیت مجازی سازی بر روی رایانه‌ی شما فعال نشده باشد از این مرحله جلوتر نخواهید رفت و به شما پیغام خطا می‌دهد. در ادامه در جواب قسمت User Account Control نیز گزینه‌ی Yes را ارسال کنید. بعد از آن با تصویر زیر مواجه خواهید شد.



بعد از این مرحله ToolBox شما به صورت کامل نصب شده است. همانطور که مشاهده کردید Kitematic و docker-compose نیز کنار سرویس داکر برای شما نصب شد که این می‌توانید از آن نیز استفاده کنید.

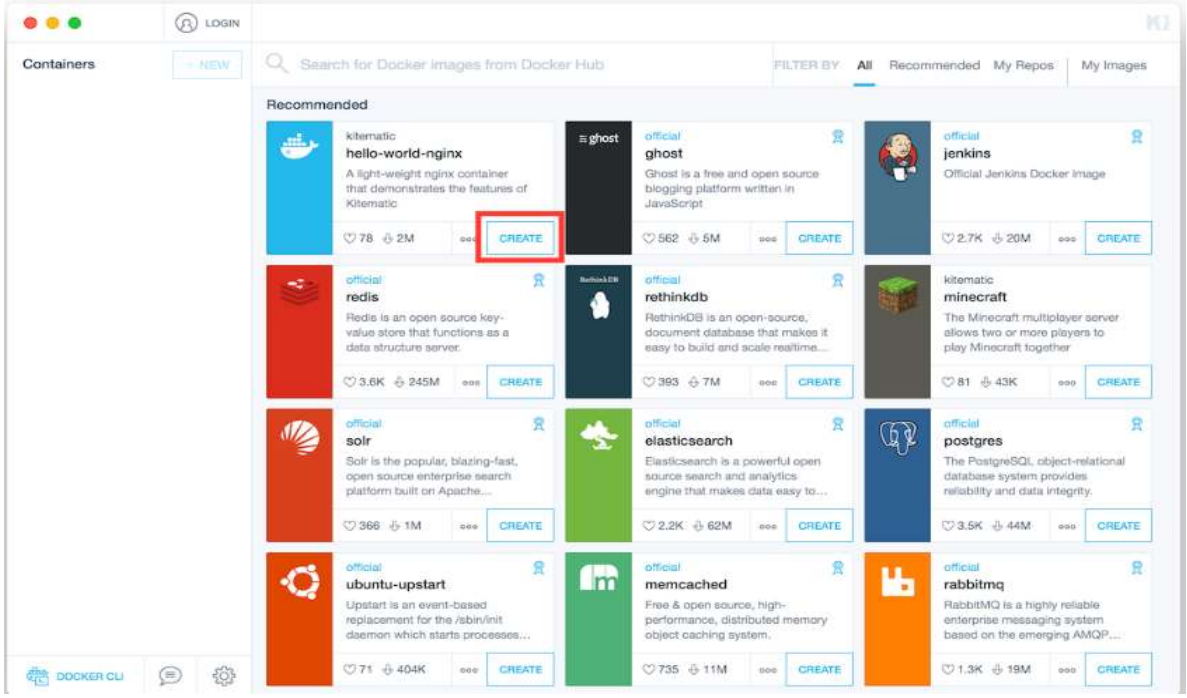
۱۰. نصب و استفاده از Kitematic

Kitematic کارش اینه که برای شما یک داشبورد عالی ایجاد می‌کنه تا بتونید به خوبی با سرویس داکر کار کنید. به شدت ساده و عالی می‌تونید تمام اقداماتی که با استفاده از کامند و Docker Client انجام می‌دهید را اینجا با استفاده از پنل خوب و دوست‌داشتنی Kitematic انجام بدید .

Kitematic برای سیستم‌عامل‌های مک، ویندوز و اوبونتو لینوکس پکیج آماده قرار داده است.

برای نصب Kitematic مخصوص سیستم‌عامل خودتون را دانلود کنید. توجه کنید به جز مواردی که ذکر شده سیستم‌عامل Arch Linux نیز در ریپوزیتوری AUR خودش این پکیج را دارد که این از این آدرس می‌توانید این پکیج را دریافت و آن را نصب کنید. برای نصب در سیستم‌عامل‌های دیگر با دریافت Source code می‌توانید نصب را انجام و از آن استفاده کنید.

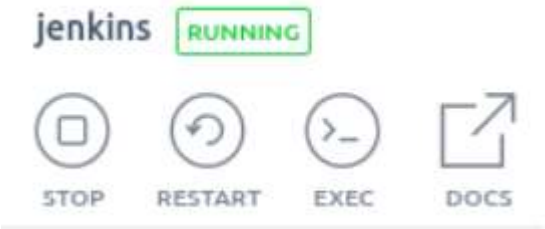
پس از نصب، بر روی ایکون Kitematic کلیک کنید تا برنامه باز شود. این برنامه با کاربر سطح پایین اجرا می‌شود از این رو نیاز دارد که دسترسی این کاربر برای سرویس داکر فراهم شده باشد یعنی باید همون کاربری که با اون برنامه را اجرا می‌کنید به سرویس داکر دسترسی داشته باشد. برای ایجاد دسترسی کاربر به سرویس داکر در سیستم‌عامل لینوکس قبلا در اینجا توضیح داده شده بود. بعد از اینکه نرم‌افزار Kitematic توانست با سرویس‌دهنده‌ی داکر ارتباط برقرار کند از شما حساب کاربری Docker Hub را می‌خواهد که این مرحله را می‌توانید Skip کنید و اگر حساب کاربری داکر هاب خود را وارد کنید با آن ریجیستری ارتباط برقرار می‌کند و به خوبی لیست imageهای موجود را به شما به صورت گرافیکی همانند تصویر زیر به شما نمایش می‌دهد.



در قسمت سمت چپ تصویر بالا که **container** نوشته شده. در این محل تمامی کانتینرهای ایجاد شده قرار دارد که شما می‌توانید لیست آنها را مشاهده کنید و تمامی اقدامات لازم را با انتخاب هر کدام از آنها انجام دهید.

برای ایجاد یک کانتینر می‌توانید بر روی هر کدام از ایمیج‌های که در وسط تصویر مشاهده می‌کنید روی گزینه‌ی **CREATE** کلیک کنید تا از روی اون ایمیج یک کانتینر برای شما ایجاد کند. پس از ایجاد کانتیر آن را در منوی سمت چپ خود خواهید داشت. با کلیک بر روی کانتیر می‌توانید تمام مشخصات آن را مشاهده کنید و در صورت نیاز آنها را تغییر دهید.

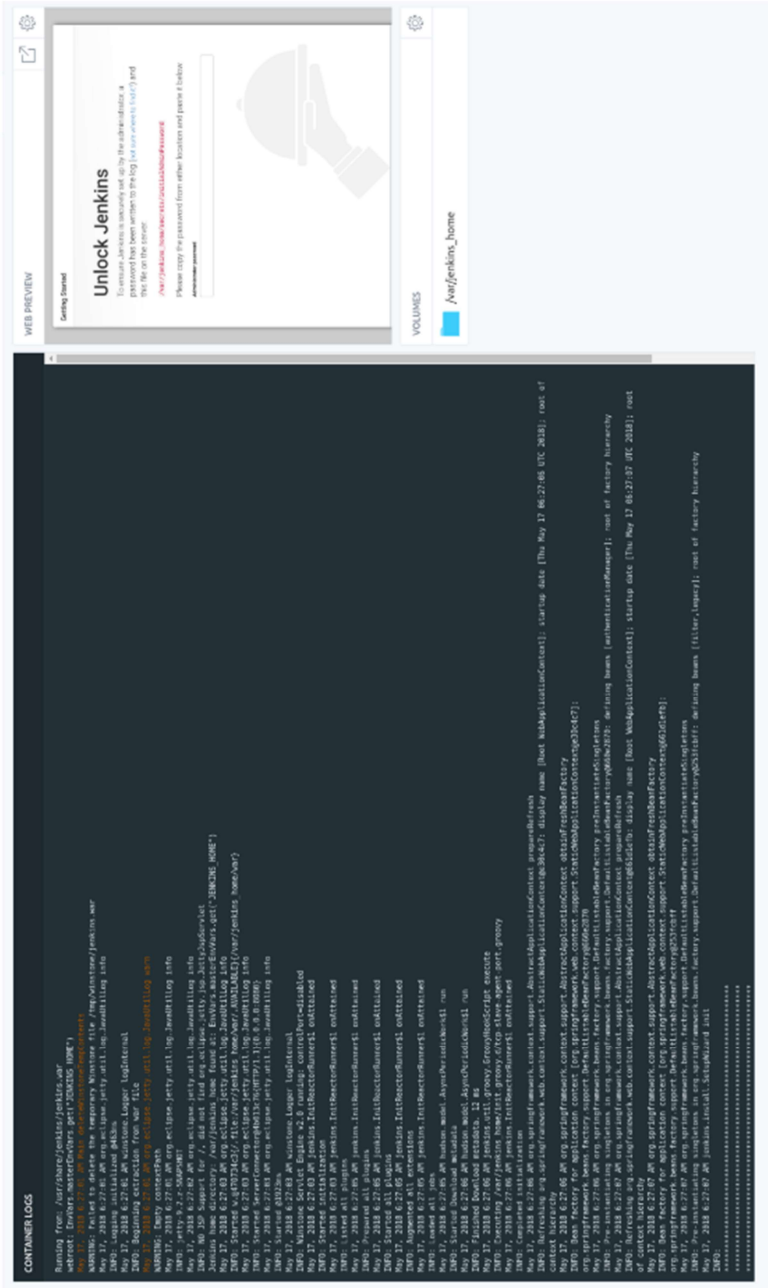
بر روی هر کانتینر که کلیک کنید به صورت کلی با دو تا منو مواجه هستید قسمت اول که با نام **Home** قرار دارد و شما یک سری اطلاعات از کانتینر خواهید داشت و در قسمت دوم که نام **Settings** می‌باشد شما می‌توانید تمام قسمت‌های مربوط به کانتینر را تغییر دهید. البته در کنار این دو منو یک قسمت وجود دارد که همواره در بالای صفحه در دسترس خواهد بود. برای توضیح تنظیمات، بنده یک کانتینر **Jenkins** راه‌اندازی کردیم که تمام توضیحات بر روی همین کانتینر داده می‌شود.



همانطور که ملاحضه می‌کنید در این تصویر وضعیت حال حاضر کانتینر به همراه نام آن را به شما نشان می‌دهد که به صورت **Running** می‌باشد. در زیر آن قسمتی برای **Stop** کردن کانتینر و یا **Restart** کردن آن وجود دارد. قسمت **Exec** به شما این امکان را می‌دهد که وارد کانتینر شده و بتوانید روی کانتینر دستورات خود را اعمال کنید. قسمت **Docs** هم مربوط به داکیومنت‌های ایمیج می‌باشد که صفحه‌ی داک مربوط به ایمیج در داکر هاب را به شما نمایش می‌دهد.

منوی Home

در منوی **Home** شما می‌توانید لاگ کانتینر را به همراه **Web Preview** و **Volume**های آن مشاهده کنید.

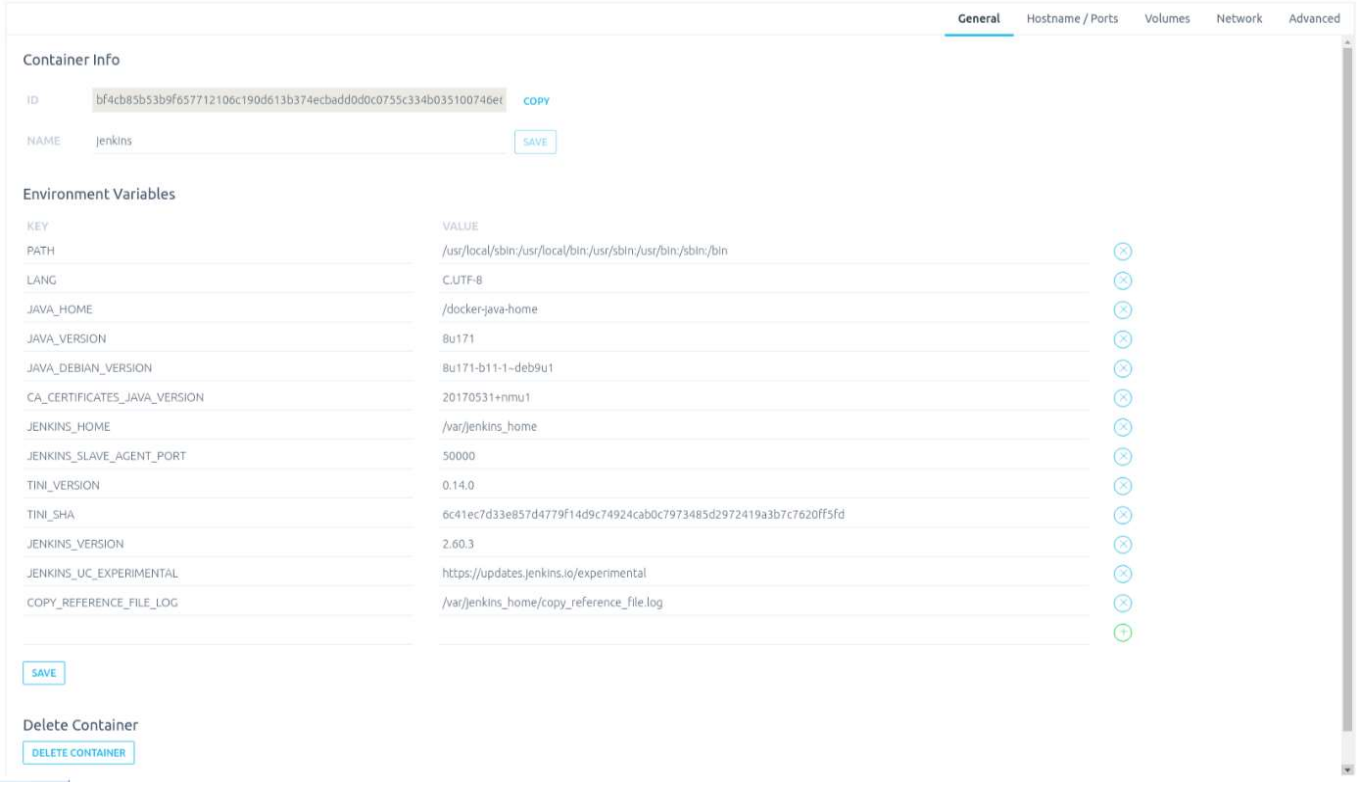


منوی Settings

در منوی Settings موارد زیر در اختیار شما قرار می‌گیرد.

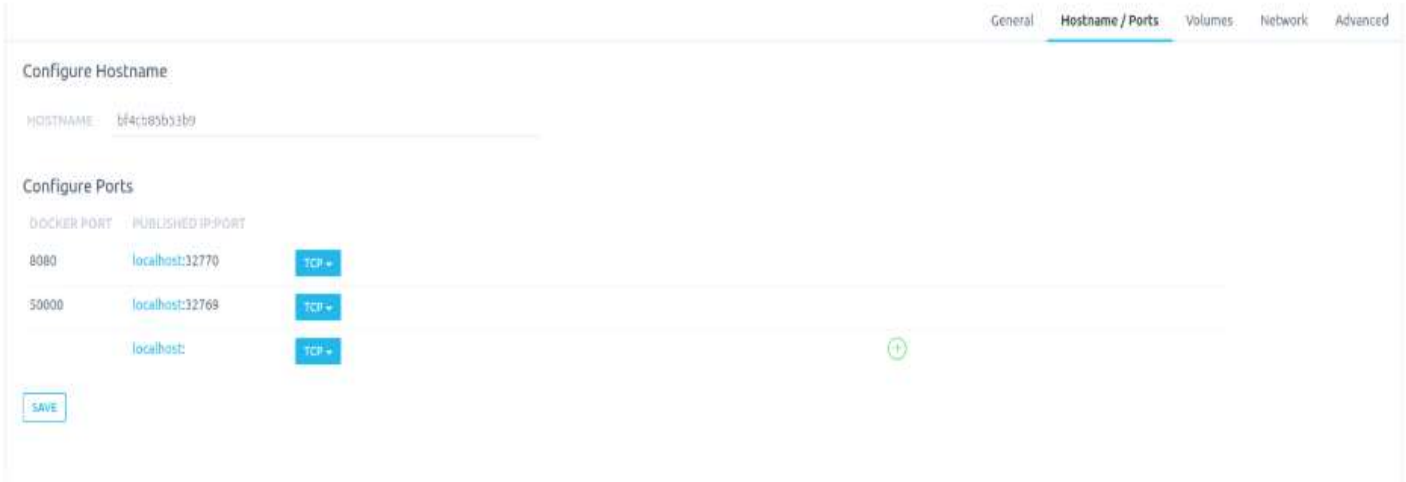
زیرمنوی General

در این قسمت شما می‌توانید نام کانتینر را تغییر داده و متغیرهای مختلف را به صورت کاملا پویا به کانتینر خود اضافه و یا کم کنید. امکان اینکه کانتینر را به صورت کامل پاک کنید نیز در این قسمت فراهم است.



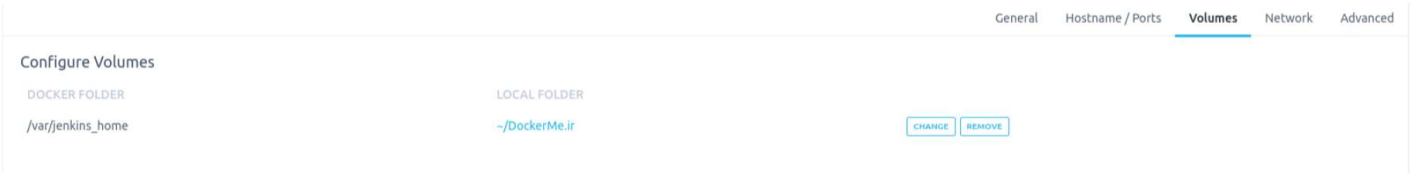
زیر منوی Hostname / Ports

در این منو شما امکان این را دارید که Hostname و Ports مد نظر خودتون را تغییر دهید.



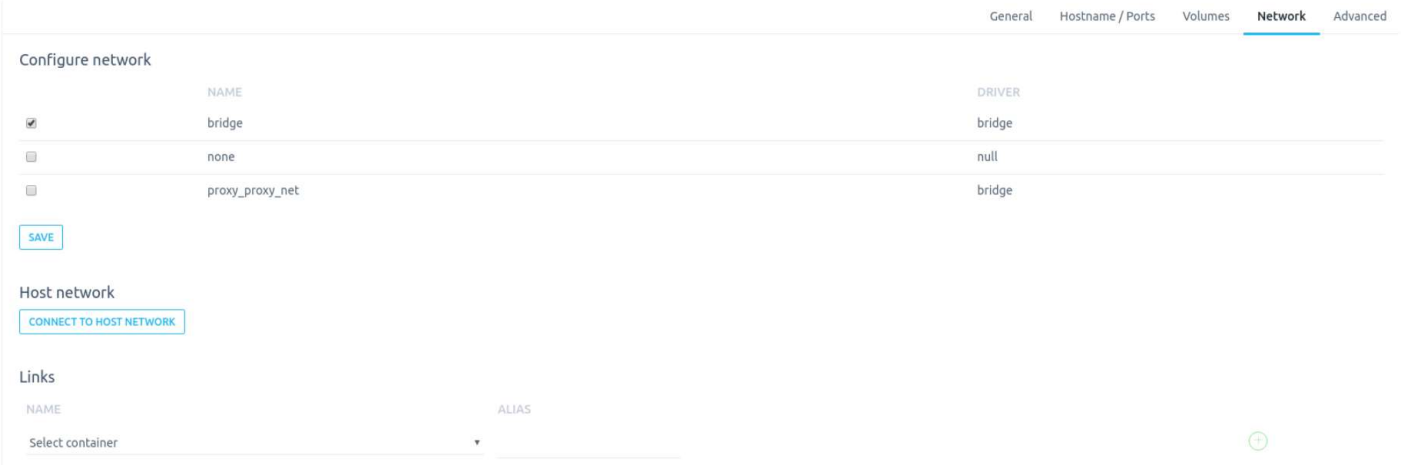
زیر منوی Volumes

در این منو شما می‌توانید به کانتیر خود والیوم اختصاص دهید که با استفاده از آن اطلاعات مهم داخل کانتینر خود را بر روی دیسک حفظ کنید.



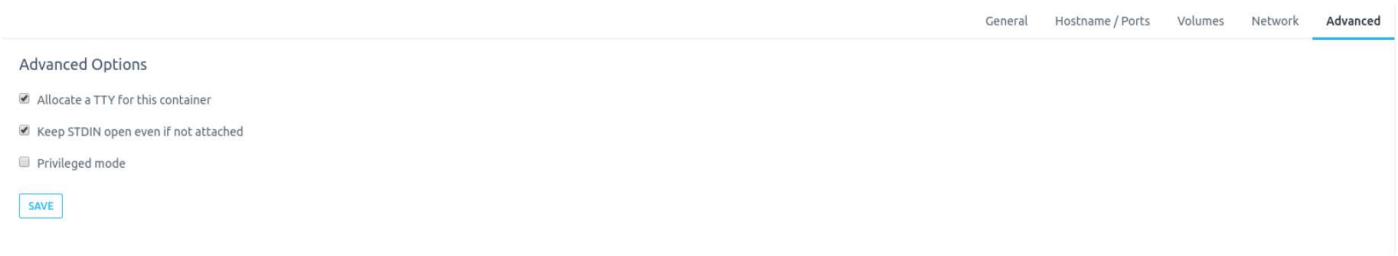
زیرمنوی Network

در این منو تنظیمات اصلی شبکه‌ی کانتینر خود را انجام خواهید داد. از چه کارت شبکه‌ای استفاده شود و یا اینکه به صورت خاص شبکه‌ی این کانتینر همان شبکه‌ی هاست باشد که این تنظیمات به شدت کارساز خواهد بود. از امکانات دیگر این زیرمنو این است که امکان لینک شدن با کانتینرهای دیگه را برای شما فراهم می‌کند.



زیرمنوی Advanced

در این زیرمنو شما امکان این را دارید که کانتینر را با دسترسی بالا راه‌اندازی کنید و برای کانتینر خود TTY در نظر بگیرید و اینکه این موارد را به صورت تعاملی در اختیار داشته باشید.



نکته‌ی مهمی که حتما به آن توجه کنید اینکه تمام این موارد با استفاده از دستورات داکر قابل انجام است و خود این نرم‌افزار با استفاده از این دستورات با سرویس‌دهنده‌ی داکر ارتباط برقرار می‌کند. برای ابتدای امر و درک بهتر مفاهیم توصیه می‌شود از این نرم‌افزار استفاده کرد اما بعد از گذشت زمان بهتر است که به صورت کامل با خود دستورات داکر تمام اقدامات را انجام داد.

۱۱. داکر فایل(DockerFile)

dockerfile ها در واقع فایل‌های تنظیمات داکر می‌باشند که با استفاده ار آنها می‌توانیم به داکر بگوییم که یک **container** را چگونه بالا بیاورد و آن را تنظیم کند. مثلا چه سرویس‌هایی روی آن بالا بیاید و چه پورت‌هایی را باز کند و چه اجازه دسترسی به آنها دهد.

در زیر جدولی آورده ایم که کلمات کلیدی مورد استفاده در این فایل را توضیح داده است.

دستور مورد استفاده	توضیح
FROM	مشخص کردن image مورد استفاده در Container
MAINTAINER	سازنده و توصیحاتContainer
RUN	اجرا کردن دستور هنگام ساخته شدن Container
CMD	اجرا کردن یک دستور بعد از ساخته شدن Container
ENTERYPOINT	اجرا کردن یک دستور بعد از ساخته شدن Container
LABEL	اضافه کردن متادیتا به Container
EXPOSE	مشخص کردن پورت هایی که Container بعد بالا آمدن اجازه دسترسی به آن را می دهد
ENV	اضافه کردن متغییرهای محلی به Container

ADD	اضافه یا کپی یا لینک کردن فایل یا دایرکتوری به Container
COPY	اضافه یا کپی فایل یا دایرکتوری به Container بدون لینک کردن
VOLUME	اضافه کردن یک mount point در ماشین میزبان یا Containerهای دیگر
USER	ست کردن یک user یا UID در Container
WORKDIR	ست کردن دایرکتوری اجرایی در Container

در اینجا ما با مثالی درک این موضوع را برای شما آسان‌تر می‌کنیم.

در مثال ما می‌خواهیم یک **container** ایجاد کنیم که روی آن سرویس **httpd** بالا بیاید و همچنین یک فایل **index.html** در آن باشد و شماره پورت**httpd**، شماره ۸۰ باشد.

برای این منظور ابتدا یک فایل ایجاد می‌کنیم

```
vi nowlearn.ir-container
```

سپس اطلاعاتی مانند زیر را در آن کپی می‌کنیم

```
FROM centos
MAINTAINER nowlearn <admin@nowlearn.ir>
RUN yum -y install httpd
RUN echo “Hello DockerFile” > /var/www/html/index.html
EXPOSE 80
CMD [“-D”, “FOREGROUND”]
ENTRYPOINT [“/usr/sbin/httpd”]
```

نکته: شما می‌توانید با توجه به جدول و نیاز خودتان **dockerfile** خودتان را بسازید

سپس برای بالا آوردن **Container** دستوری مانند زیر را اجرا می‌کنیم

```
docker build -t web_server:latest
```

نکته: توجه کنید که باید در دایرکتوری که **dockerfile** خود را ساختیم باشیم تا عملیات با موفقیت انجام شود.

در صورتی که همه کارها را درست انجام داده باشیم خروجی مانند زیر می‌بینیم

```
Sending build context to Docker daemon 10.24 kB
Step 0 : FROM centos
—> 7322fbe74aa5
Step 1 : MAINTAINER nowlearn <admin@nowlearn.ir>
```

```
—> Running in fa5364b3d41f
—> 57d8fd36b7f7
.....
.....
Removing intermediate container 3efa8e1dcae9
Successfully built 7c39aaa338b4
```

حالا اگر ما docker image بگیریم باید خروجی مانند زیر ببینیم

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
web_server	latest	7c39aaa338b4	24 seconds ago	283.9 MB
docker.io/centos	latest	ce20c473cd8a	8 weeks ago	172.3 MB

برای اینکه ببینیم آیا container ما درست بالا آمده ااست یا نه دستور زیر را اجرا می‌کنیم

```
curl http://localhost/
```

اگر خروجی مانند زیر دیدید کارهای انجام شده درست بوده است

```
Hello DockerFile
```

نکته: در دفعه‌های بعدی برای اینکهcontainer ی که ساختیم سریع بالا بیاوریم بصورت زیر عمل می‌کنیم

```
docker run -d -p 80:80 web_server
```

مورد دیگر:

در داکر فایل (Docker File) آنچه پروژه به آن نیاز دارد را تعریف می‌کنیم.

در این محیط مجازی دسترسی به منابع سیستم از بقیه سیستم جدا شده است. پس باید پورت ۸۰ پروژه برای اتصال از بیرون آماده باشد.

مزیت استفاده:

با استفاده از این سیستم می توانید پروژه خود را در هر کجا اجرا کنید.

چگونه یک داکر فایل (Docker File) بسازیم؟

یک پوشه می‌سازیم و به داخل آن (cd) آن رفته و فایل به نام DockerFile می‌سازیم.

دقت کنید که دقیقا نام این فایل DockerFile باشد نه DockerFile.txt و...

```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

پروژه را می نویسیم

دو فایل دیگر با نام های **requirements.txt** و **app.py** در همان پوشه ایجاد می‌کنیم.

requirements.txt

```
Flask
Redis
```

app.py

```
from flask import Flask
from redis import Redis, RedisError
import os
import socket

redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visits = redis.incr("counter")
    except RedisError:
        visits = "<i>cannot connect to Redis, counter disabled</i>"

    html = "<h3>Hello {name}!</h3>" \
           "<b>Hostname:</b> {hostname}<br/>" \
           "<b>Visits:</b> {visits}"
    return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname(), visits=visits)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)
```

همانطور که متوجه شدید ما پکیج های لازم را در requirements.txt تعریف کردیم و بعد از نصب الزامات در داکر فایل (Docker File) این فایل صدا زده و پکیج های ذکر شده در آن را در پروژه نصب می کند.

پروژه را می سازیم

در پوشه ای که بودیم دستور ls را وارد میکنیم تا محتویات پوشه را ببینیم.

```
$ ls
```

محتویات باید به صورت زیر باشد.

```
Dockerfile    app.py    requirements.txt
```

دستور زیر را اجرا می کنیم تا داکر ایمیج (Docker Image) خود را بسازیم.

```
docker build --tag=friendlyhello .
```

یا

```
docker build -t=friendlyhello .
```

با tag=friendlyhello:v0.0.1 این روش نوشتن در دستور بالا می توانیم ورژن نرم افزار خود را مشخص کنیم.

دستور زیر را می‌زنیم تا داکر ایمیج‌های (Docker Images) خود را ببینیم.

```
$ docker image ls
```

خروجی:

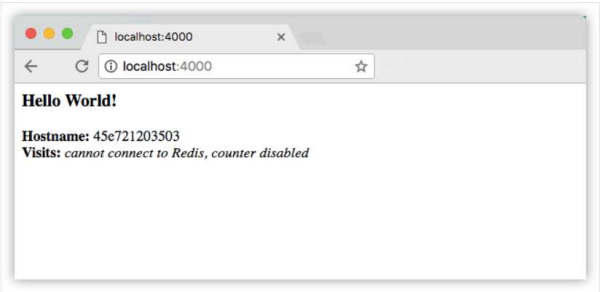
```
REPOSITORY      TAG         IMAGE ID
friendlyhello   latest      326387cea398
```

همانطور که مشاهده می کنید داکر ایمیج (Docker Image) ما ساخته شده است.

با دستور زیر پروژه خود را اجرا می کنیم.

```
docker run -p 4000:80 friendlyhello
```

حالا میتوانیم در مرورگر خود پروژه را ببینیم با ادرس http://localhost:4000



۱۲.اشتراک گذاری داکر امیج(Docker Image)

برای اینکار شما نیاز به ساخت یک اکانت در سایت داکر هاب دارید.

بعد از ساخت اکانت دستور زیر را میزنیم.

```
$ docker login
```

نیاز به علامت گذاری داکر امیج **(Docker Image)** خود داریم به صورت زیر

```
$ docker tag image username/repository:tag
```

برای مثال

```
$ docker tag friendlyhello gordon/get-started:part2
```

برای مشاهده ایمیجی که الان نشانه گذاری کردیم دستور زیر را وارد می کنیم.

```
$ docker image ls
```

خروجی:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
friendlyhello	latest	d9e555c53008	3 minutes ago	195MB
gordon/get-started	part2	d9e555c53008	3 minutes ago	195MB
python	2.7-slim	1c7128a655f6	5 days ago	183MB

...

آپلود ایمیج

```
$ docker push username/repository:tag
```

اگر وارد اکانت خود در سایت داکر هاب شوید متوجه خواهید شد که پروژه شما در آن آپلود شده است

بحث داکر فایل تمام شد.

۱۳. داکر کامپوز(Docker Compose)

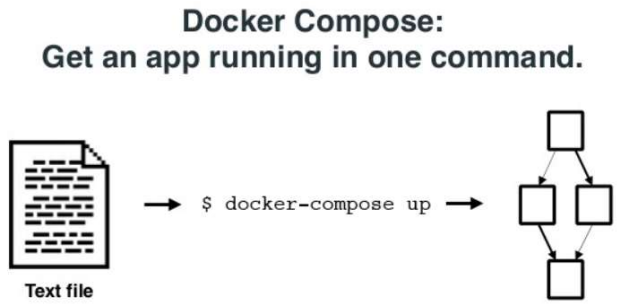
Docker Compose گرداننده و یا راهنمای کانتینر داکر است.

ابزار **Docker Compose** (با نام قبلی**Fig**) ابزاری متن باز برای خودکارسازی کار با کانتینرهاست که ابتدا توسط تیم **Orchard** ایجاد و سپس در سال ۲۰۱۴ توسط شرکت داکر خریداری شد.

داکر کامپوز برای تعریف و اجرای برنامه هایی که از چند کانتینر تشکیل شده‌اند بکار میرود. با استفاده از این ابزار می‌توان یک یا چند کانتینر را به همراه تمام پارامترهای لازم برای اجرای آنها در یک فایل YAML به نام کامپوزفایل (docker-compose.yml) تعریف کرد و سپس همه آنها را با استفاده از تنها یک دستور ایجاد و راه اندازی نمود. در این فایل هر یک از این کانتینرها یک سرویس نامیده می‌شود که تعریف دقیق آن به این صورت است : هر سرویس یک کانتینر است که به شکلی با کانتینرهای دیگر در تعامل است و مشخصات زمان اجرای خاص خود را دارد.

غالبا استفاده از داکر کامپوز یک فرآیند سه مرحله ایست:

- تعریف محیط اجرای برنامه‌ها با داکرفایل (Dockerfile) که در نتیجه در هر جایی امکان اجرا دارد.
- تعریف سرویس‌های معادل هر برنامه در کامپوزفایل که در نتیجه امکان اجرای آنها در یک محیط ایزوله در کنار یکدیگر بوجود می‌آید.
- اجرای دستور docker-compose up که در نتیجه برنامه شما با همه سرویس‌هایش اجرا می‌شود.



در واقع ابزار داکر کامپوز دستوراتی برای مدیریت چرخه عمر کانتینرها دارد که امکانات زیر را فراهم می‌کند:

- اجرا، توقف و بازتولید سرویسها
- مشاهده وضعیت سرویسهای در حال اجرا
- مشاهده خروجی (لاگ) سرویسهای در حال اجرا

کامپوز فایل(Compose File)

تمام داستان، نوشتن یک فایل به نام docker-compose.yaml است که در روت پروژه قرار میگیرد و توی این فایل ما توصیف می‌کنیم که چی می‌خواهیم و چجوری می‌خواهیم .

من یک دیتابیس می‌خواهم با این ویژگی‌ها، و همینطور به PHP نسخه ۷٫۴ هم برای اجرا نیازه، به Nodejs ورژن ۱۰ و مثلا شاید Redis . یعنی تمام این تکنولوژی‌ها در یک فایل (فعلا شما در نظر بگیرید یک فایل وجود دارد به این نام بعدا می‌توانیم

این فایل را با نام‌های مختلف ذخیره کنیم و نحوه اجرای آن فرق دارد) لیست می‌شوند و نحوه تنظیمات آنها نیز بیان می‌شود. نکته زیبایی که وجود دارد این است که در این فایل می‌توانید نحوه رفتار کانتینر داکر خود را در حین خطا تعیین کنید یعنی مثلا اگر دیتابیس MySQL به خطا خورد به هر دلیلی، داکر ریستارتش کن یا یک کانتینتر جدید اجرا کن می‌توانید تمام این اپلیکیشن‌ها را در یک محیط ایزوله در شبکه‌های گوناگون قرار دهید و یا اینکه داده‌های کانتینر در چه Volume ای ذخیره شود (مسیر دیسک).

اجازه دهید یک مثال ببینیم:

```
1  version: '3'
2
3  services:
4    db:
5      image: mysql:5.7
6      volumes:
7        - /home/ali/backup/mysql_data :/var/lib/mysql
8      restart: always
9      environment:
10       MYSQL_ROOT_PASSWORD: wordpress
11       MYSQL_DATABASE: wordpress
12       MYSQL_USER: wordpress
13       MYSQL_PASSWORD: wordpress
14
15   wordpress:
16     depends_on:
17       - db
18     image: wordpress:latest
19     ports:
20       - "8000:80"
21     restart: always
22     environment:
23       WORDPRESS_DB_HOST: db:3306
24       WORDPRESS_DB_USER: wordpress
25       WORDPRESS_DB_PASSWORD: wordpress
```

متغیرهای محیطی که با تگ environment مشخص می‌شوند برای هر کانتینری تفاوت دارد و شما باید بر اساس کانتینری که استفاده می‌کنید این متغیرها را مقداردهی کنید. تمام این موارد در سایت https://hub.docker.com/ وجود دارد و وقتی به صفحه ایمیج مد نظرتان بروید این موارد را خواهید دید.

در خط اول ورژن docker-compose را مشخص می‌کنید. سرویس‌ها را مشخص می‌کنید که تمام سرویس‌های استفاده شده توسط شما را شناسایی می‌کند.

در این مثال از دو سرویس ساده استفاده کرده‌ایم. یک دیتابیس که نام آن را db و دیگری وردپرس که نام آن را wordpress قرار دادیم. این اسامی دلخواه بوده و هر اسمی مایل باشید می‌توانید بر روی آنها قرار دهید. در سرویس db مشخص کرده‌ایم داده‌های موجود برروی کانتینر به مسیر /home/ali/backup/mysql_data بایند باشند. هر داده‌ای در مسیر گفته شده در سمت راست در کانتینر قرار گیرد بلافاصله در مسیر گفته شده در سمت چپ نیز قرار می‌گیرد. باید این کار را انجام دهید چون

اگر دیتای خود را بایند نکنید بعد از خاموش شدن کانتینر بلافاصله تمام دیتای شما از بین خواهد رفت. چون کانتینرها اگر خاموش شوند ناپایدار هستند و اگر به آنها نگویید موردی را ذخیره نمی‌کنند.

در خط بعد **reastart:always** را قرار داده‌ایم که به این معنی است که اگر خطا خورد یا کانتینتر استاپ شد بلافاصله آن را ریستارت کن. متغیرهای محیطی که مشخص هستند چه مواردی هستند و تعریف آنها واضح است.

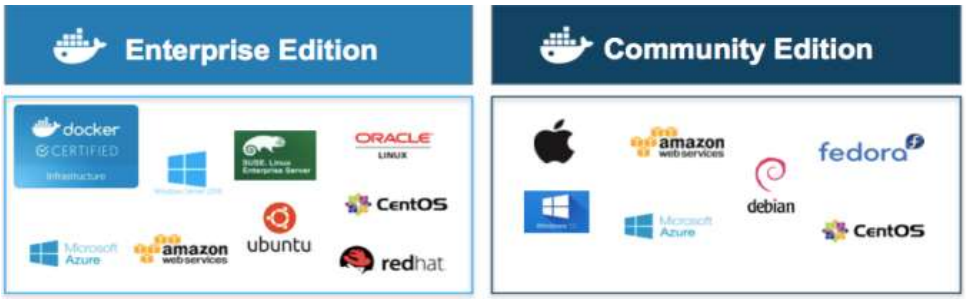
فقط در کانتینر وردپرس وقتی مشخص می‌کنید که به چه کانتینری وابسته است **(depend_on)** باید نام کانتینر خود را که در قبل از این کانتینر تعریف کرده‌اید عنوان کنید و حواستان باشد باید قبل از این کانتینر تعریف شده باشد وگرنه خطای عدم شناسایی کانتینر برمی‌گرداند.

حال شما با داشتن این فایل و قرار دادن آن در هر سرور و مسیری قادر خواهید بود اپلیکیشن خود را به درستی و بدون کوچکترین کار اضافه و همچنین کار یک نیروی انسانی دیگر اجرا کنید

با استفاده از دستور **docker-composer up -d** فایل اجرا و کانتینرها ساخته خواهند شد.

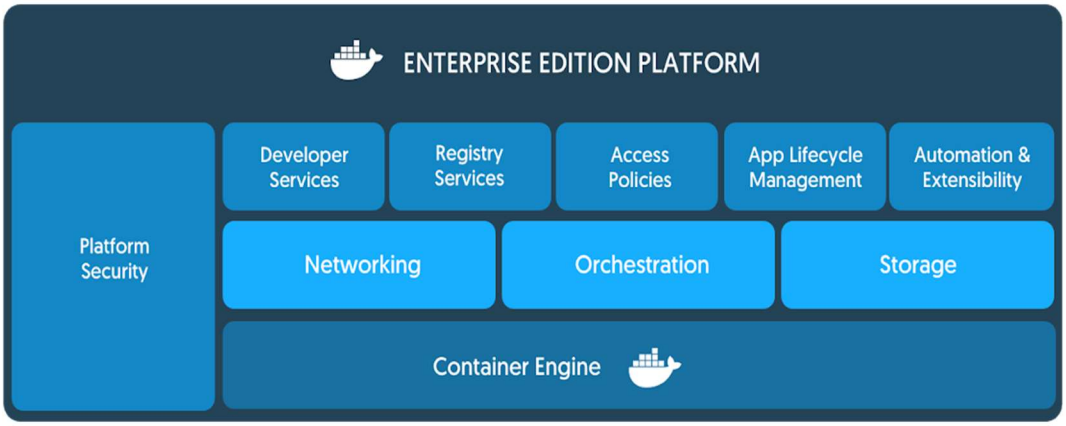
به جای ساختن ماشین‌های مجازی متفاوت و کنترل آنها شما فقط با این فایل کار خواهید کرد. مابقی کار را داکر برای شما انجام میدهد.

۱۴.نسخه‌های مختلف داکر که خوب است بدانیم



داکر به صورت کلی دارای دو نسخه‌ی اصلی می‌باشد که در ادامه توضیح داده می‌شود:

❖ نسخه‌ی Enterprise Edition



این نسخه نیاز به اشتراک دارد و برای استفاده از آن باید هزینه کرد. یک راهکار کامل حرفه‌ای است که دارای پشتیبانی کاملی می‌باشد. در کل یک راه‌کار کاملا خودکار برای نگهداری و راه‌اندازی کانتیرها به صورت کاملا حرفه‌ای است. تمام آن چیزی که شما به آن نیاز دارید و خودتون باید ایجاد کنید را به صورت یک‌پارچه در اختیار شما قرار می‌دهد.

- بر روی سیستم‌عامل و پلتفرم‌های مختلف در دسترس خواهد بود.
- دارای مانیتورینگ و احراز هویت دیجیتال به همراه اسکن امنیتی کانتینرها می‌باشد.
- به خوبی با اپلیکیشن‌های مختلف یکپارچه شده و می‌تواند فرآیندهای مختلف را به صورت خودکار راه‌اندازی کرد.
- همواره دارای آخرین نسخه‌ی پایدار داکر می‌باشد.
- دارای پشتیبانی به همراه SLA است.
- دارای قابلیت پشتیبانی از نسخ مختلف و قدیمی و ارائه‌ی پچ‌های امنیتی و به روزرسانی‌های لازم می‌باشد.
- مجموعه کاملی از خدمات و آموزش‌ها برای سرعت بخشیدن به سرویس‌دهی است.
- برای استفاده از آن نیاز است تا ثبت نام کرده و هزینه پرداخت کرد.

در تصویر زیر فهرستی از قابلیت‌های این نسخه که در سه مسیر Basic, Standard, Advanced ارائه می‌شود فهرست شده است.

	EE BASIC SUPPORTED CONTAINER RUNTIME	EE STANDARD ENTERPRISE MANAGEMENT FEATURES FOR SINGLE TEAM USE	EE ADVANCED ADDITIONAL CAPABILITIES FOR MULTI-TEAM USE, POLICY-BASED AUTOMATION
Secure container engine with networking, security, and storage	✓	✓	✓
Docker Certified infrastructure, plugins and containers	✓	✓	✓
Private image registry with caching		✓	✓
Integrated app and cluster management across Swarm and Kubernetes		✓	✓
Enhanced RBAC, LDAP / AD support		✓	✓
Integrated secrets management, image signing policy		✓	✓
Secure multi-tenancy with node-based isolation			✓
Policy-based, automated image promotions			✓
Image mirroring across registries			✓
Image security scanning and continuous vulnerability scanning			✓

❖ نسخه‌ی Community Edition

این نسخه به صورت رایگان ارائه می‌شود. برای تیم‌های کوچیک و محققین بسیار توصیه می‌شود و همواره دو مسیر برای ارائه دارد که به قرار زیر می‌باشد:

- **نسخه‌ی Stable**: هر ۳ ماه یک‌بار به روزرسانی می‌شود و قابل اعتماد است. البته ممکن است به وصله‌ای (patch) نیاز داشته باشد که در بین این زمان ارائه می‌شود. البته این به روز رسانی‌ها تا ۴ ماه بعد از ارائه نسخه‌ی ابتدایی ارائه می‌شود.
- **نسخه‌ی Edge**: هر ماه ویژگی‌های جدیدی را به شما اضافه می‌کند.

نسخه‌ی docker CE دارای قابلیت‌های زیر می‌باشد.

- بر روی سیستم‌عامل و پلتفرم‌های مختلف در دسترس خواهد بود.

- همواره نسخه‌ی آخر داکر خواهد بود که بهینه‌سازی شده برای سرویس‌های زیرساختی
- همواره آخرین قابلیت‌های پایدار داکر را در خود دارد.
- ریپوهای عمومی و خصوصی را بدون محدودیت در اختیار قرار می‌دهد.
- به صورت خودکار بیلد از سرویس دارد.
- برنامه‌ی منظم برای انتشار نسخه‌ی پایدار **(Stable)** دارد.
- دارای پشتیبانی جامعه‌ی آزاد است**(Community support)** .
- کاملا رایگان است.

پشتیبانی داکر:

در مورد پشتیبانی داکر صحبت شد که در نسخه به چه صورت است اما در تصویر زیر این موضوع به صورت خیلی شفاف و دقیق‌تری بررسی شده است.

	COMMUNITY EDITION	ENTERPRISE EDITION	
		BUSINESS DAY	BUSINESS CRITICAL
Coverage Hours	Help available anytime from the online community of docker users. plus non-technical customer service inquiries handled by Docker directly.	9:00am - 6:00 Local Time Monday - Friday	Business Day + 24 x 7 x 365 for P1
Initial Response Time		From 2 business hours, depending on severity	From 1 hour, depending on severity
Contact Methods	Docker Community Forums Customer Service Form	Docker Success Center Phone	Docker Success Center Phone
Number of Designated Support Admins	N/A	4	8
Number of Incidents	N/A	Unlimited	Unlimited

نکته‌ی مهم اینکه اگر از نسخه‌ی رایگان داکر استفاده کنید تمام امکانات نسخه‌ی حرفه‌ای آن را می‌توانید ایجاد کنید اما در نسخه‌ی حرفه‌ای داکر تمام این موارد به صورت کامل در اختیار شما قرار می‌گیرد.

۱۵. دستورات مهم داکر

❖ دستور docker ps

```
e: $ docker ps -h
d -h has been deprecated, please use --help

- ps [OPTIONS]

-s


            Show all containers (default shows just running)
- filter    Filter output based on conditions provided
t string    Pretty-print containers using a Go template
int         Show n last created containers (includes all states)
t           Show the latest created container (includes all stat
unc         Don't truncate output
            Only display numeric IDs
            Display total file sizes
e: $
```

❖ دستور docker info

```
ıd@DockerMe: $ docker info -h
ı shorthand -h has been deprecated, please use --help

ıe:  docker info [OPTIONS]

ılay system-wide information

.ons:
ˈ, --format string   Format the output using the given Go templ
ıd@DockerMe: $
```

❖ دستور docker pull

```
ockerMe: $ docker pull -h
orthand -h has been deprecated, please use --help

 docker pull [OPTIONS] NAME[:TAG|@DIGEST]

 image or a repository from a registry

:
-all-tags                Download all tagged images in the repc
-disable-content-trust   Skip image verification (default true)
ockerMe: $
```

❖ دستور docker images

```
ckerMe: $ docker images -h
rthand -h has been deprecated, please use --help

 docker images [OPTIONS] [REPOSITORY[:TAG]]

ges

:
-all             Show all images (default hides intermediate im
-digests         Show digests
-filter filter   Filter output based on conditions provided
-format string   Pretty-print images using a Go template
-no-trunc        Don't truncate output
-quiet           Only show numeric IDs
ckerMe: $
```

❖ دستور docker create

با استفاده از این دستور می‌توانید یک کانتینر ایجاد کنید. این دستور دارای آپشن‌های بسیار زیادی می‌باشد که با استفاده از آنها می‌توان یک کانتینر ایجاد کرد و تمام مواردی که مد نظر برای ران شدن آن است را در آن لحاظ کرد. بعد از ایجاد کانتینر با تمام موارد مد نظر می‌توان آن را **start** کرد. آپشن‌های این دستور با آپشن‌های دستور **run** یکسان می‌باشد و تنها تفاوت در این است که در دستور **run** بعد از اجرای دستور کانتینر شروع به کار می‌کند اما در اینجا کانتینر ایجاد شده است و بعد نیاز است که آن را **start** کنید. موارد اصلی و مهم این دستور به همراه دستور **run** توضیح داده می‌شود.

```
ahmad@DockerMe:~$ docker create --help

Usage:  docker create [OPTIONS] IMAGE [COMMAND] [ARG...]

Create a new container

Options:
      --add-host list                  Add a custom host-to-IP mapping (host:ip)
  -a, --attach list                    Attach to STDIN, STDOUT or STDERR
      --blkio-weight uint16            Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
      --blkio-weight-device list       Block IO weight (relative device weight) (default [])
      --cap-add list                   Add Linux capabilities
      --cap-drop list                  Drop Linux capabilities
      --cgroup-parent string           Optional parent cgroup for the container
      --cidfile string                 Write the container ID to the file
      --cpu-period int                 Limit CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int                  Limit CPU CFS (Completely Fair Scheduler) quota
      --cpu-rt-period int              Limit CPU real-time period in microseconds
      --cpu-rt-runtime int             Limit CPU real-time runtime in microseconds
  -c, --cpu-shares int                 CPU shares (relative weight)
      --cpus decimal                   Number of CPUs
```

❖ دستور docker run

با استفاده از این دستور می‌توان از روی ایمیج‌های موجود یا ایجاد شده کانتینر راه‌اندازی کرد. این دستور آپشن‌های خیلی زیادی دارد که با استفاده از آنها می‌توان هر آن چیزی که لازم است تا روی یک کانتینر اعمال شود انجام گیرد. نکته‌ی مهمی که هست اینکه توضیح تمام آپشن‌های این دستور هم از حوصله این مستند خارج هست و هم تمام این موارد کاربردی نخواهد بود. سعی بر آن است که تمام آپشن‌های پر کاربرد توضیح داده شود و مابقی موارد اگر نیاز بود در سناریوهایی که برای تست پیاده‌سازی می‌کنیم بررسی می‌شود.

```
ahmad@DockerMe:~$ docker run --help

Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

Options:
      --add-host list                  Add a custom host-to-IP mapping (host:ip)
  -a, --attach list                    Attach to STDIN, STDOUT or STDERR
      --blkio-weight uint16            Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
      --blkio-weight-device list       Block IO weight (relative device weight) (default [])
      --cap-add list                   Add Linux capabilities
      --cap-drop list                  Drop Linux capabilities
      --cgroup-parent string           Optional parent cgroup for the container
      --cidfile string                 Write the container ID to the file
      --cpu-period int                 Limit CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int                  Limit CPU CFS (Completely Fair Scheduler) quota
      --cpu-rt-period int              Limit CPU real-time period in microseconds
      --cpu-rt-runtime int             Limit CPU real-time runtime in microseconds
  -c, --cpu-shares int                 CPU shares (relative weight)
      --cpus decimal                   Number of CPUs
```

تعدادی آپشن وجود دارد برای ایجاد محدودیت روی پردازنده و میزان استفاده از رم که حسب نیاز سناریوها توضیح داده می‌شود.

آپشن **d-** : کانتینر را در **background** راه‌اندازی می‌کند و تنها **id** آن را نمایش می‌دهد.

آپشن **dns** : می‌توان **dns server** انتخابی به کانتینر اضافه کرد.

آپشن **entrypoint** : با استفاده از این آپشن می‌توان **entrypoint** به کانتینر داد در غیر این صورت از **entrypoint** ایمیج استفاده می‌کند.

آپشن **e-** : متغیرهای محلی به کانتینر اضافه می‌کند.

آپشن **env-file** : متغیرهای محلی را از روی یک فایل دریافت می‌کند.

آپشن **expose** : یک پورت یا تعدادی پورت را در معرض **(expose)** قرار می‌دهد.

آپشن **hostname** : برای کانتینر **hostname** تنظیم می‌کند.

آپشن **i –** : به صورت تعاملی کانتینر را راه‌اندازی می‌کند.

آپشن **ip** : به کانتینر **ip** اختصاص می‌دهد.

آپشن **link** : بین کانتینر با کانتینرهای دیگه لینک برقرار می‌کند.

آپشن **l –** : برای کانتینر لیبل قرار می‌دهد.

آپشن **device** : با استفاده از این آپشن می‌توان دیوایس‌های سرور هاست را به کانتینر منتقل می‌کند.

آپشن **name** : به کانتینر اسم می‌دهد.

آپشن **net** : مشخص می‌کند که این کانتینر از چه شبکه‌ای استفاده کند.

آپشن **p-** : با استفاده از این آپشن پورت کانتینر را در هاست منتشر **(Publish)** می‌کند.

آپشن **P-** : تمام پورت‌هایی که در کانتینر وجود دارد را در هاست منتشر می‌کند.

آپشن **privileged** : کانتینر را با دسترسی بالا اجرا می‌کند.

آپشن **restart** : با استفاده از این آپشن می‌توان سیاست‌های مختلفی برای ریستارت کانتینر در نظر گرفت.

آپشن **rm** : به صورت خودکار بعد از متوقف شدن کانتینر آن را پاک خواهد کرد.

آپشن **t-** : به کانتینر یک **tty** اختصاص می‌دهد.

آپشن **v -** : با استفاده از این آپشن می‌توان والیوم به کانتینر اضافه کرد و آن را روی هاست مپ کرد.

آپشن **volume-driver**: با استفاده از این آپشن می‌توان درایور والیوم داکر به خصوصی را برای کانتینر مشخص کرد.

آپشن **volumes-from**: این آپشن امکان مپ کردن والیوم از یک کانتینر دیگه رو برای ما مهیا می‌کنه.

آپشن **user**: با استفاده از این آپشن می‌توان **name** یا **uid** و **group** یا **gid** به کانتینر اختصاص داد.

آپشن **sysctl**: می‌توان آپشن **sysctl** به کانتینر اعمال کرد.

آپشن **w-** : محلی که دایرکتوری کاری هنگام شروع به کار کانتینر باشد را مشخص می‌کند.

خوب دستور **run** یکی از مهمترین دستورات داکر است که تمام راه‌اندازی کانتینرها با استفاده از آن انجام می‌شود. در ادامه برای مشخص‌تر موضوع از چند مثال استفاده می‌شود. در مثال‌های زیر بیشتر آپشن‌های معرفی شده استفاده شده است. به نحوه‌ی استفاده از این آپشن‌ها دقت کنید.

```
docker run --name test -it debian
docker run -t -i --privileged ubuntu bash
docker  run -w /path/to/dir/ -i -t  ubuntu pwd
docker run -p 127.0.0.1:80:8080/tcp ubuntu bash
docker run --expose 80 ubuntu bash
docker run -e MYVAR1 --env MYVAR2=foo --env-file ./env.list ubuntu bash
docker run -l my-label --label com.example.foo=bar ubuntu bash
docker run -itd --network=my-net --ip=10.10.9.75 busybox
docker run --volumes-from 777f7dc92da7 --volumes-from ba8c0c54f0f2:ro -i -t ubuntu pwd
docker run --device=/dev/sdc:/dev/xvdc --device=/dev/sdd --device=/dev/zero:/dev/nulo -i -t ubuntu ls -l /dev/{xvdc,sdd,nulo}
docker run --restart=always redis
docker run --sysctl net.ipv4.ip_forward=1 someimage
```

در این لینک {https://docs.docker.com/engine/reference/run/} به خوبی تمام آپشن‌های دستور **run** را قسمت بندی کرده و هر قسمت را به صورت جداگانه بررسی کرده است. آپشن‌های این دستور دارای اهمیت است و خیلی مهمه که در سناریوهای تمرینی از تمام این موارد استفاده کرد.

❖ دستور docker rename

با استفاده از این دستور شما می‌توانید کانتینر خودتون رو **rename** کنید و اسم جدید برای اون قرار بدید. این دستور آپشن اضافی ندارد. نکته اینکه تفاوتی بین اینکه کانتینر در حال کار باشد یا نه وجود ندارد و در هر حالتی که باشد شما می‌توانید نام آن را تغییر دهید.

```
ahmad@DockerMe: $ docker rename --help

Usage:  docker rename CONTAINER NEW_NAME

Rename a container

Options:

ahmad@DockerMe: $
```

برای تغییر نام کانتینر به مثال زیر توجه کنید همانند مثال زیر می‌توانید نام کانتینر را تغییر دهید.

```
$ docker ps
CONTAINER ID    IMAGE    COMMAND    CREATED        STATUS              PORTS        NAMES
4ce8adaf33b3    busybox  "sh"       2 hours ago    Up  About a minute                 test1
$ docker rename test1 salam
$ docker ps
CONTAINER ID    IMAGE    COMMAND    CREATED        STATUS          PORTS        NAMES
4ce8adaf33b3    busybox  "sh"       2 hours ago    Up 3 minutes                 salam
```

❖ دستور docker rm

با استفاده از این دستور شما می‌توانید کانتینرهای خود را پاک کنید. همواره می‌توانید یک یا تعدادی کانتینر را پاک کنید. این دستور ۳ عدد آپشن دارد که در ادامه آنها را بررسی می‌کنیم.

```
ahmad@DockerMe: $ docker rm --help

Usage:  docker rm [OPTIONS] CONTAINER [CONTAINER...]

Remove one or more containers

Options:
  -f, --force     Force the removal of a running container (uses SIGKILL)
  -l, --link      Remove the specified link
  -v, --volumes   Remove the volumes associated with the container
ahmad@DockerMe: $
```

آپشن **f-** : این آپشن به اجبار و با اولیوت بالا کانتینر در حال کار را نیز پاک می‌کند. به صورت پیش فرض و بدون استفاده از این آپشن شما نمی‌توانید کانتینر در حال کار را پاک کنید اما اگر از این آپشن استفاده کنید این امکان برای شما مهیا می‌شود.

آپشن **l-** : این آپشن لینک‌هایی که بین کانتینرها در نتورک ایجاد شده است را نیز پاک می‌کند.

آپشن **v-** : هم خود کانتینر و هم والیومی که به آن متصل شده است را پاک می‌کند.

همان‌طور که قبلا نیز گفته شده بود می‌توانید دستورات داکر را به صورت ترکیبی نیز استفاده کنید. یکی از دستوراتی که ممکن است زیاد استفاده شود این است که کل کانتینرهای داخل یک سرور را پاک کنید که برای این کار از دستور زیر استفاده می‌کنید.

```
docker rm $(docker ps -a -q)
```

در این دستور لیستی از تمامی کانتینرهای داخل سرور تهیه می‌شود که با استفاده از دستور **q -** فقط **id** آنها فهرست شده است سپس با استفاده از دستور **rm** تمامی آنها پاک می‌شود.

❖ دستور docker rmi

با استفاده از این دستور می‌توانید یک یا تعدادی از ایمیج‌ها را پاک کنید. دو تا آپشن برای این دستور وجود دارد که در ادامه توضیح داده می‌شود.

```
ahmad@DockerMe: $ docker rmi --help

Usage:  docker rmi [OPTIONS] IMAGE [IMAGE...]

Remove one or more images

Options:
  -f, --force      Force removal of the image
      --no-prune   Do not delete untagged parents
ahmad@DockerMe: $
```

بدون آپشن: تنها ایمیج‌هایی که وابستگی هیچ کانتینری به آنها نباشد را پاک می‌کند. یعنی اگر شما از روی ایمیج خود کانتینری راه‌اندازی کرده باشید آن را پاک نخواهد کرد. در مثال زیر از روی ایمیج **busybox:latest** کانتینر ۴ **ce8adaf33b3** راه‌اندازی شده و در حال کار می باشد البته در صورتی که کانتینر وجود داشته باشد و **stop** شده باشد نیز باز ایمیج را پاک نمی‌کند.

```
$ docker rmi busybox:latest

Error response from daemon: conflict: unable to remove repository reference "busybox:latest" (must force) - container 4ce8adaf33b
3 is using its referenced image 8c811b4aec35
```

آپشن**-f** : با این آپشن ایمیج را پاک می‌کند و اگر از روی اون ایمیج کانتینری هم باشد آن ایمیج را آنتگ می‌کنند اما کماکان آی‌دی اون ایمیج باقی خواهد ماند. در هیچ صورتی اگر کانتینری در حال ران داشته باشید نمی‌توانید ایمیجی که از روی اون کانتینر را ایجاد کردید را پاک کنید.

آپشن **no-prune**: ایمیج‌های که از روی آنها این ایمیج ایجاد شده است (والدین این ایمیج هستند) را پاک نمی‌کند.

❖ دستور docker tag

با استفاده از این دستور تگ (tag= برچسب) مربوط به ایمیج را تغییر نام می‌دهید و می‌توانید برای ایمیج خود نام و تگ جدید قرار بدید. این دستور آپشن اضافی ندارد و به صورت زیر می‌توانید تگ هر ایمیج را تغییر دهید و از آن یک ایمیج جدید با تگ جدید داشته باشید. این آپشن‌ها بسیار کاربردی بوده و استفاده می‌شود.

```
ahmad@DockerMe: $ docker tag --help

Usage:  docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Options:

ahmad@DockerMe: $
```

برای استفاده همانند مثال زیر در ابتدا ایمیج اصلی که می‌خواهید تگ یا نام آن را تغییر دهید و بعد ایمیج جدید با تگ و نام جدید را قرار می‌دهید. البته همانطور که در مثال هم مشخص است با این دستور می‌توانید ایمیج خود را به صورت کامل تغییر نام دهید و rename کنید. به جای نام و تگ برای ایمیج اصلی می‌توانید از id آن نیز استفاده کنید.

```
# docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
$ docker tag busybox:latest test:2
```

همواره برای اینکه ایمیج را در ریجیستری push کنند ابتدای امر آن را تغییر نام داده و به نام ریجیستری و ریپوی مد نظر تغییر داده سپس آن را push می‌کنند. به مثال زیر توجه کنید.

```
$ docker tag busybox:latest docker.dockerme.ir/test/test:2
```

همان‌طور که ملاحضه می‌کنید ایمیج جدید با نام test و تگ ۲ و ریپوی test در ریجیستری docker.dockerme.ir آماده شده است و الان آماده‌ی push کردن می‌باشد. البته قبلش باید به ریجیستری مورد نظر login کرده باشید و دسترسی آن مهیا شده باشد که در دستور login کامل توضیح داده می‌شود.

❖ دستور docker search

با این دستور در **docker hub** که ریجیستری اصلی داکر می‌باشد می‌توانید جستجو کنید و ایمیج مد نظر خودتون رو پیدا کنید.

خطای ۴۰۳ هنگام استفاده از دستور **docker search**:

همان‌طور که قبلا هم گفته شد داکر هاب بر روی ایران و چند تا کشور دیگه مسدود می‌باشد برای دانلود از آن روش‌های مختلفی پیشنهاد شد که در این لینک به صورت کامل توضیح داده شده است. اگر شما از روش **mirror registry** استفاده می‌کنید می‌توانید ایمیج دانلود کنید اما دستور **search** برای شما کماکان خطای ۴۰۳ را خواهد داد. برای اینکه این مشکل نیز برطرف شود می‌توانید از **proxy** یا از روش سایت شکن استفاده کنید که تمام این موارد در همان لینک توضیح داده شده است.

```
ahmad@DockerMe: $ docker search --help

Usage:  docker search [OPTIONS] TERM

Search the Docker Hub for images

Options:
  -f, --filter filter   Filter output based on conditions provided
      --format string   Pretty-print search using a Go template
      --limit int       Max number of search results (default 25)
      --no-trunc        Don't truncate output
ahmad@DockerMe: $
```

بدون آپشن: در داکر هاب جستجو کرده و هر موردی را که پیدا کند به شما نمایش می‌دهد. در این نمایش چند آیتم مشخص است که می‌توان بر اساس همان موارد فیلتر نوشت و جستجوی بهتری داشت.

- نام ایمیج: اسم هر ایمیج می‌باشد.
- توضیحات: توضیح خیلی مختصری از هر ایمیج قرار دارد.
- تعداد ستاره‌های هر ایمیج می‌باشد.
- رسمی بودن ایمیج را مشخص می‌کند.
- اتوماتیک بودن ایمیج را مشخص می‌کند.

آپشن **filter** : برای جستجویی که انجام می‌دهید می‌توانید فیلتر قرار دهید و بر اساس فیلتر شما خروجی دستور نمایش داده شود.

مواردی که می‌توان در فیلتر لحاظ کرد همانند تصویر زیر می‌باشد.

- stars (int - number of stars the image has)
- is-automated (boolean - true or false) - is the image automated or not
- is-official (boolean - true or false) - is the image official or not

در ادامه نحوه استفاده از این آپشن در مثالی مشخص شده است. در این درخواست مشخص دنبال نسخه‌ای از ایمیج **busybox** هستیم که رسمی **(official)** باشد و تعداد ستاره‌های آن ۲۰ به بالا باشد. در جواب ایمیجی را به ما نشان می‌دهد که ۱۲۶۸ تا ستاره دارد و به صورت رسمی می‌باشد. حالا به دستور دوم توجه کنید که قسمت **automated** را نیز وارد فیلتر کردیم که نتیجه آن هیچ ایمیجی نیست که هم به صورت رسمی باشد و هم حداقل ۲۰ تا ستاره داشته باشد و هم اتوماتیک باشد.

```
$ docker search --filter "is-official=true" --filter "stars=20" busybox
NAME        DESCRIPTION        STARS        OFFICIAL        AUTOMATED
busybox     Busybox base image.   1268      [OK]
docker search --filter "is-official=true" --filter "stars=20" --filter "is-automated=true"busybox
NAME        DESCRIPTION        STARS        OFFICIAL        AUTOMATED
```

آپشن **format**: بر اساس این آپشن شما می‌توانید خروجی **search** را همانند چیزی که خودتان می‌پسندید تنظیم کنید. مواردی که می‌توانید در فیلتر لحاظ کنید به قرار زیر است.

Placeholder	Description
`.Name`	Image Name
`.Description`	Image description
`.StarCount`	Number of stars for the image
`.IsOfficial`	"OK" if image is official
`.IsAutomated`	"OK" if image build was automated

دقیقا تمام ۵ موردی که در هر ایمیج قرار دارد را می‌توانید بر اساس چیزی که خودتان در نظر دارید تنظیم کنید. به مثال زیر توجه کنید. در این مثال تنها در خروجی نام ایمیج به همراه : و بعد از آن تعداد ستاره‌های هر ایمیج و بعد از آن عبارت **DockerMe.ir +++** قرار داده شده است. یعنی هر طوری که مد نظر ما باشد می‌توانیم این خروجی را تنظیم کنیم.

```
$ docker search --format "{{.Name}}: {{.StarCount}} +++ DockerMe.ir" nginx
nginx: 8564 +++ DockerMe.ir
jwilder/nginx-proxy: 1340 +++ DockerMe.ir
richarvey/nginx-php-fpm: 547 +++ DockerMe.ir
jrcs/letsencrypt-nginx-proxy-companion: 374 +++ DockerMe.ir
kong: 190 +++ DockerMe.ir
webdevops/php-nginx: 103 +++ DockerMe.ir
kitematic/hello-world-nginx: 99 +++ DockerMe.ir
bitnami/nginx: 52 +++ DockerMe.ir
zabbix/zabbix-web-nginx-mysql: 52 +++ DockerMe.ir
linuxserver/nginx: 35 +++ DockerMe.ir
1and1internet/ubuntu-16-nginx-php-phpmyadmin-mysql-5: 35 +++ DockerMe.ir
tobi312/rpi-nginx: 19 +++ DockerMe.ir
```

آپشن **limit**: تعداد خروجی‌های جستجو را می‌توانید مشخص کنید. به صورت پیش‌فرض ۲۵ عدد می‌باشد. شما می‌توانید فقط تعدادی که مد نظر خودتون هست رو مشخص کنید تا نمایش دهد. در مثال زیر ۳ تا مورد از جستجو رو بنده در نظر گرفتم.

```
$ docker search --limit 3 --format "{{.Name}}: {{.StarCount}} +++ DockerMe.ir" nginx
nginx: 8564 +++ DockerMe.ir
jwilder/nginx-proxy: 1340 +++ DockerMe.ir
jrcs/letsencrypt-nginx-proxy-companion: 374 +++ DockerMe.ir
```

آپشن **no-trunc**: خروجی نمایش داده شده را مختصر نمی‌کند.

❖ دستور docker build

با استفاده از این دستور می‌توان از روی داکرفایل ایمیج داکر ایجاد کرد. این دستور آپشن‌های زیادی دارد که در تصویر زیر لیست آنها را می‌توانید مشاهده کنید.

```
ahmad@DockerMe: $ docker build --help

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile

Options:
      --add-host list          Add a custom host-to-IP mapping (host:ip)
      --build-arg list         Set build-time variables
      --cache-from strings     Images to consider as cache sources
      --cgroup-parent string   Optional parent cgroup for the container
      --compress               Compress the build context using gzip
      --cpu-period int         Limit the CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int          Limit the CPU CFS (Completely Fair Scheduler) quota
  -c, --cpu-shares int         CPU shares (relative weight)
      --cpuset-cpus string     CPUs in which to allow execution (0-3, 0,1)
      --cpuset-mems string     MEMs in which to allow execution (0-3, 0,1)
      --disable-content-trust  Skip image verification (default true)
  -f, --file string            Name of the Dockerfile (Default is 'PATH/Dockerfile')
      --force-rm               Always remove intermediate containers
      --iidfile string         Write the image ID to the file
      --isolation string       Container isolation technology
      --label list             Set metadata for an image
  -m, --memory bytes           Memory limit
      --memory-swap bytes      Swap limit equal to memory plus swap: '-1' to enable unlimited swap
      --network string         Set the networking mode for the RUN instructions during build (default "default")
      --no-cache               Do not use cache when building the image
      --pull                   Always attempt to pull a newer version of the image
  -q, --quiet                  Suppress the build output and print image ID on success
      --rm                     Remove intermediate containers after a successful build (default true)
      --security-opt strings   Security options
      --shm-size bytes         Size of /dev/shm
  -t, --tag list               Name and optionally a tag in the 'name:tag' format
      --target string          Set the target build stage to build.
      --ulimit ulimit          Ulimit options (default [])
ahmad@DockerMe: $
```

توضیح مختصری درباره‌ی فرآیند**build**:

فرآیند **build** به این صورت است که از روی یک ایمیج **Base** مابقی تغییرات در آن داده می‌شود تا ایمیج مد نظر شما ایجاد شود. نکته این که در این مسیر ممکن چندین ایمیج ایجاد شود تا به ایمیج نهایی برسیم. این قابلیت لایه‌ای بودن بسیار به کم حجم شدن ایمیج نهایی می‌تواند کمک کند. ممکن است در لایه‌ی آخر ایمیج شما مشکلی رخ دهد که مجبور باشید بعد از اصلاح

آن از ابتدا فرآیند **build** را انجام دهید. قابلیت کش که از همین روش لایه‌ای بودن ایمیج‌ها فراهم شده است به شما کمک می‌کند که تمام لایه‌های درست و صحیح را مجدد دریافت نکنید و تنها از همان لایه‌ای که خطا داشته است ادامه دهید.

به صورت کلی این دستور به این صورت استفاده می‌شود، در جایی که داکرفایل وجود دارد اجرا شده تا از روی اون ایمیج مورد نظر شما را ایجاد کند البته نام ایمیج را به صورت رندم انتخاب می‌کند. این روش ساده‌ترین راه استفاده از این دستور می‌باشد. حالا با کمک آپشن‌های این دستور می‌توان به حالات مختلف از این دستور استفاده کرد. در ضمن لزومی ندارد حتما داکرفایل در پوشه‌ی جاری که دستور اجرا می‌شود باشد می‌تواند آن را از مسیر مشخص دیگری دریافت کرد. با توجه به اهمیت و میزان استفاده، برخی از آپشن‌ها توضیح داده می‌شود و مابقی آپشن‌ها حسب نیاز سناریوهایی که بعدا پیاده‌سازی می‌شود بررسی و توضیح داده خواهد شد.

آپشن **t-** : با استفاده از آن می‌توانید به ایمیجی که ایجاد می‌شود اسم و تگ بدهید.

آپشن **f-** : اگر داکر فایل شما در مسیر دیگه‌ای قرار دارد با استفاده از این آپشن می‌توانید مسیردهی کنید تا به اون داکرفایل برسید. به صورت پیش‌فرض و بدون استفاده از این آپشن باید حتما در مسیر داکرفایل باشید.

آپشن **force-rm** :کانتینرهای میانی که ایجاد می‌شود در لایه‌های مختلف تا ایمیج نهایی ایجاد شود را پاک می‌کند. اگر از این آپشن استفاده کنید کانتینرهای میانی را پاک می‌کند در این صورت اگر مشکلی در لایه‌ای ایجاد شود باید از ابتدا تمام لایه‌ها را ایجاد کرد.

آپشن **rm** : بعد از نهایی شدن و ایجاد کامل ایمیج، کانتینرهایی که برای لایه‌های مختلف ایجاد کرده است را پاک می‌کند. این آپشن به صورت پیش‌فرض فعال می‌باشد. تفاوت این آپشن با آپشن **force-rm** در این است که در این آپشن باید حتما ایمیج با موفقیت ایجاد شود تا تمام کانتینرها را پاک کند.

آپشن: **pull** : همواره آخرین نسخه‌ی ایمیج بیس را دریافت کرده و بعد فرآیند **build** را انجام خواهد داد.

آپشن **disable-content-trust** : بررسی‌های مربوط به چک کردن ایمیج را انجام نمی‌دهد.

آپشن **no-cache**: از کش استفاده نمی‌کند و ایمیج را از ابتدا به صورت کامل **build** می‌کند. معمولا یکی از بهترین قابلیت‌های داکر استفاده از کش و لایه‌ای بودن ایمیج‌ها می‌باشد. اگر این آپشن را فعال کنید هیچ‌گاه در هنگام **build** از کش استفاده نکرده و همواره سعی بر آن دارد که خودش ایمیج را از ابتدا ایجاد کند.

آپشن **quiet** : خروجی هنگام **build** را به شما نمایش نمی‌دهد و تنها **id** ایمیج‌هایی که ایجاد شده است را نمایش می‌دهد. توصیه می‌شود از این آپشن استفاده نکنید زیرا این اطلاعات به شما کمک می‌کند.

آپشن **security-opt**: با استفاده از این دستور می‌توانید آپشن‌های امنیتی در هنگام **build** خود اضافه کنید.

❖ دستور docker commit

با استفاده از این دستور از کانتینر در حال کار می‌توانیم یک ایمیج ایجاد کنیم. این موضوع زمانی کاربرد دارد که تغییراتی در کانتینر ایجاد شده و نیاز است تا این تغییرات در ایمیج جدیدی ثبت شود.

```
ahmad@DockerMe: $ docker commit --help

Usage:  docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]

Create a new image from a container's changes

Options:
  -a, --author string    Author (e.g., "John Hannibal Smith <hannibal@a-team.com>")
  -c, --change list      Apply Dockerfile instruction to the created image
  -m, --message string   Commit message
  -p, --pause            Pause container during commit (default true)
ahmad@DockerMe: $
```

آپشن **a-** : با استفاده از این آپشن در هنگام **commit** می‌توانید به ایمیج یک نام نویسنده جدید اضافه کنید. این موضوع مرسوم است که در هر ایمیج نام فردی که داکرفایل را ایجاد کرده و ایمیج را آماده کرده است قرار داده می‌شود.

آپشن **c-** : می‌توانید در هنگام **commit** ایمیج، برخی از دستورات داکرفایل را در آن اضافه کنید. در مثال نمونه‌های استفاده از این آپشن آورده شده است. لیست مواردی که می‌توان از آنها استفاده کرد در ادامه فهرست شده است.

CMD | ENTRYPOINT | ENV | EXPOSE | LABEL | ONBUILD | USER | VOLUME | WORKDIR

آپشن **m-** : در هنگام **commit** ایمیج پیغامی نوشته شود که این پیغام می‌تواند توضیح این باشد که چرا این ایمیج ایجاد شده است و اینکه چه تغییراتی در آن انجام شده است.

آپشن **p-** : این آپشن که به صورت پیش‌فرض فعال می‌باشد در هنگام **commit** ایمیج آن را **pause** می‌کند که دیگر هیچ تغییری در هنگام **commit** انجام نشود.

❖ دستور docker cp

با استفاده از این دستور می‌توان فایل یا دایرکتوری رو بین کانتینر و هاست جابه‌جا کرد. هم می‌توان از هاست به کانتینر و هم از کانتینر به هاست منتقل کرد.

```
ahmad@DockerMe: $ docker cp --help

Usage:  docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-
        docker cp [OPTIONS] SRC_PATH|- CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem

Options:
  -a, --archive       Archive mode (copy all uid/gid information)
  -L, --follow-link   Always follow symbol link in SRC_PATH
ahmad@DockerMe: $
```

همان‌طور که در ادامه مشاهده می‌کنید می‌توانید فایل و فولدر را از کانتینر به هاست و یا اینکه از هاست به کانتینر منتقل کنید.

```
docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-
docker cp [OPTIONS] SRC_PATH|- CONTAINER:DEST_PATH
```

آپشن **a-** : با استفاده از این آپشن اطلاعات تمام موارد مربوط به فایل یا فولدر همانند **uid/gid** را نیز کامل منتقل می‌کند.

آپشن **L-** : با استفاده از این آپشن لینک‌های نرم‌افزاری () که در فایل‌ها و فولدرها هست را دنبال می‌کند و فایلی که منتقل می‌کند دارای داده‌های درست می‌باشد و به لینک قبلی خود وابسته نیست.

❖ دستور docker events

با استفاده از این دستور می‌توانید events های سرویس داکر خودتون رو بررسی کنید. هر اتفاقی در سرویس دهنده‌ی داکر در تمامی کامپوننت‌های آن رخ دهد می‌توانید لاگ آن را مشاهده کنید.

```
ahmad@DockerMe:~$ docker events --help

Usage:  docker events [OPTIONS]

Get real time events from the server

Options:
  -f, --filter filter   Filter output based on conditions provided
      --format string   Format the output using the given Go template
      --since string    Show all events created since timestamp
      --until string    Stream events until this timestamp
ahmad@DockerMe:~$
```

آپشن filter : با استفاده از این آپشن می‌توانید برای خروجی ارائه شده فیلتر بنویسید. تغییرات در هر آبجکت داکر همانند ایمیج و کانتینر و ... را در این لاگ اضافه می‌کند. لیست این موارد در این لینک فهرست شده است. این لیست تقریبا تمام مواردی که مورد نیاز است را شامل می‌شود. به خاطر گسترده بودن لاگ‌های خروجی، این آپشن اهمیت خیلی زیادی دارد زیرا می‌توان با استفاده از آن لاگ را متناسب با هدفی که داریم جدا کنیم و مشاهده کنیم. در ادامه لیست مواردی که می‌توان در فیلتر قرار داده آورده شده است.

- config (`config=<name or id>`)
- container (`container=<name or id>`)
- daemon (`daemon=<name or id>`)
- event (`event=<event action>`)
- image (`image=<repository or tag>`)
- label (`label=<key>` or `label=<key>=<value>`)
- network (`network=<name or id>`)
- node (`node=<id>`)
- plugin (`plugin=<name or id>`)
- scope (`scope=<local or swarm>`)
- secret (`secret=<name or id>`)
- service (`service=<name or id>`)
- type
  (`type=<container or image or volume or network or daemon or plugin or service or node or secret or config>`)
- volume (`volume=<name>`)

آپشن format : با استفاده از این آپشن می‌توانید فرمت خروجی را متناسب با نظر خودتون تغییر بدهید. قالبی که از آن پیروی می‌کند همان قالب پیش‌فرض Go می‌باشد که در این لینک می‌توانید آن را بررسی کنید. یکی از فرمت‌های اصلی ارائه خروجی با قالب json است.

آپشن since : برای این است که رخدادها را از چه زمانی به شما نمایش دهد.

آپشن until : برای این است که رخدادها را تا چه زمانی به شما نمایش دهد.

در مثال‌های زیر به صورت کامل این موضوع مشخص می‌باشد.

```
$ docker events --format '{{json .}}'
{"status":"create","id":"196016a57679bf42424484918746a9474cd905dd993c4d0f4..  {"status":"attach","id":"196016a57679bf424
24484918746a9474cd905dd993c4d0f4..

$ docker events --since '10m'
2017-01-05T00:35:41.241772953+08:00 volume create testVol (driver=local)
2017-01-05T00:35:58.859401177+08:00 container create d9cd...4d70 (image=alpine:latest, name=test)

$ docker events --filter 'event=stop'
2017-01-05T00:40:22.880175420+08:00 container stop 0fdb...ff37 (image=alpine:latest, name=test)
2017-01-05T00:41:17.888104182+08:00 container stop 2a8f...4e78 (image=alpine, name=kickass_brattain)
```

```
$ docker events --filter 'image=alpine'
2017-01-05T00:41:55.784240236+08:00 container create d9cd...4d70 (image=alpine, name=happy_meitner)
2017-01-05T00:41:55.913156783+08:00 container start d9cd...4d70 (image=alpine, name=happy_meitner)
```

```
123$ docker events --filter 'container=test'
2017-01-05T00:43:00.139719934+08:00 container start 0fdb...ff37 (image=alpine:latest, name=test)
2017-01-05T00:43:09.259951086+08:00 container kill 0fdb...ff37 (image=alpine:latest, name=test, signal=15)
```

```
123$ docker events --filter 'container=test' --filter 'container=d9cdb1525ea8'
2017-01-05T00:44:11.517071981+08:00 container start 0fdb...ff37 (image=alpine:latest, name=test)
2017-01-05T00:44:17.685870901+08:00 container start d9cd...4d70 (image=alpine, name=happy_meitner)
```

این دستور می‌تواند در فرآیند رفع مشکل بسیار موثر باشد در این لینک مثال‌های مختلفی از این موضوع زده شده است که می‌تواند کارگشا باشد.

❖ دستور docker exec

با استفاده از این دستور می‌توانید در کانتینرهای در حال کار دستور اجرا کنید.

```
ahmad@DockerMe:~$ docker exec --help

Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container

Options:
  -d, --detach               Detached mode: run command in the background
      --detach-keys string   Override the key sequence for detaching a container
  -e, --env list             Set environment variables
  -i, --interactive          Keep STDIN open even if not attached
      --privileged           Give extended privileges to the command
  -t, --tty                  Allocate a pseudo-TTY
  -u, --user string          Username or UID (format: <name|uid>[:<group|gid>])
  -w, --workdir string       Working directory inside the container
ahmad@DockerMe:~$
```

آپشن detach : با استفاده از این آپشن دستور شما در background کانتینر اجرا می‌شود.

آپشن detach-keys : با این آپشن کلیدهای مربوط به جدا شدن از کانتینر را نادیده می‌گیرد.

آپشن env : می‌توان متغییرهای محیطی (environment variables) برای آن تنظیم کرد.

آپشن interactive : با استفاده از این دستور می‌توانید به صورت تعاملی با کانتینر ارتباط داشته باشید.

آپشن privileged : دستور را با دسترسی بالا اجرا می‌کند.

آپشن tty : با این دستور می‌توانید از کانتینر خود یک tty داشته باشید.

آپشن user : می‌توان user یا uid و group یا gid برای آن تنظیم کرد.

آپشن workdir : می‌توانید دستور مد نظر خود را در دایرکتوری که مشخص می‌کنید اجرا شود.

این دستور بسیار کاربردی بوده و ممکن است در طی کار با کانتینرها بارها به این دستور نیاز پیدا کنید. به مثال‌های زیر توجه کنید.

```
$ docker exec -d my_container touch /tmp/execWorks
```

```
$ docker exec -it my_container bash
```

```
$ docker exec -it -e VAR=1 my_container bash
```

```
$ docker exec -it my_container pwd/
```

❖ دستور docker export

با استفاده از این دستور می‌توانید فایل سیستم کانتینر خود را به صورت آرشیو برای خودتون نگه دارید. این دستور تنها یک آپشن دارد که برای مشخص کردن خروجی است.

```
ahmad@DockerMe:~$ docker export --help

Usage:  docker export [OPTIONS] CONTAINER

Export a container's filesystem as a tar archive

Options:
  -o, --output string   Write to a file, instead of STDOUT
ahmad@DockerMe:~$
```

آپشن o- : با استفاده از این آپشن می‌توان مسیر و نام فایلی که ایجاد می‌شود را مشخص کرد.

به مثال‌های زیر توجه کنید.

```
$ docker export my_container > latest.tar
```

```
$ docker export --output="latest.tar" my_container
```

❖ دستور docker import

با استفاده از این دستور فایل‌سیستمی که قبلا export کرده بودیم و آرشیو شده بود را می‌توانید به صورت فایل‌سیستم یک ایمیج جدید آن را وارد کنیم.

```
ahmad@DockerMe:~$ docker import --help

Usage:  docker import [OPTIONS] file|URL|- [REPOSITORY[:TAG]]

Import the contents from a tarball to create a filesystem image

Options:
  -c, --change list      Apply Dockerfile instruction to the created image
  -m, --message string   Set commit message for imported image
ahmad@DockerMe:~$
```

بدون آپشن می‌توانید فایل tar را از مسیری یا از یک url فراخوانی کنید و آن را با نام یک ایمیج جدید ذخیره نمایید.

آپشن change : با استفاده از این آپشن می‌توانید در هنگام ایجاد ایمیج از دستورات داکرفایل در آن استفاده کنید و در ایمیج نهایی آنها را قرار دهید. دستورالعمل‌هایی که از داکرفایل می‌توانید اینجا استفاده کنید در ادامه لیست شده است.

```
CMD | ENTRYPOINT | ENV | EXPOSE | ONBUILD | USER | VOLUME | WORKDIR
```

آپشن message : هنگام ایجاد ایمیج پیغامی باهاش ذخیره کنید که به شما بعدا در بررسی آن کمک کند.

به مثال‌های زیر توجه کنید.

```
$ sudo tar -c . | docker import --change "ENV DEBUG true" - exampleimagedir
```

```
$ sudo tar -c . | docker import - exampleimagedir
```

```
$ docker import /path/to/exampleimage.tgz
```

```
$ cat exampleimage.tgz | docker import --message "New image imported from tarball" - exampleimagelocal:new
```

```
$ docker import http://example.com/exampleimage.tgz
```

❖ دستور docker save

با استفاده از این دستور می‌توانید یک یا تعدادی ایمیج را در یک فایل tar ذخیره کرده و آنها را منتقل کنید. از این روش برای backup از ایمیج‌ها هم استفاده می‌شود. این دستور برای مواقعی که در سرویس دسترسی به اینترنت ندارید یا اینکه در اون سرور پروکسی برای دریافت ایمیج ندارید می‌تواند کمک کند.

```
ahmad@DockerMe: $ docker save --help

Usage:  docker save [OPTIONS] IMAGE [IMAGE...]

Save one or more images to a tar archive (streamed to STDOUT by default)

Options:
  -o, --output string   Write to a file, instead of STDOUT
ahmad@DockerMe: $
```

تنها آپشن این دستور -o می‌باشد که output را مشخص می‌کند. در مثال‌های زیر انواع روش‌های استفاده از این دستور مشخص شده است.

```
$ docker save busybox > busybox.tar
$ ls -sh busybox.tar
2.7M busybox.tar
```

```
$ docker save --output busybox.tar busybox
$ ls -sh busybox.tar
2.7M busybox.tar
```

```
$ docker save -o fedora-all.tar fedora
```

```
$ docker save -o fedora-latest.tar fedora:latest
```

❖ دستور docker load

با استفاده از این دستور می‌توانید ایمیج‌هایی که توسط دستور save به صورت آرشیو کردید را در سرویس‌دهنده‌ی داکر سیستم خود قرار دهید. با استفاده از این دستور می‌توانید در سرور مقصد از ایمیج‌هایی که backup کردید استفاده کنید. این دستور دارای آپشن i- می‌باشد که فایل آرشیو ورودی را مشخص می‌کند.

```
ahmad@DockerMe: $ docker save --help

Usage:  docker save [OPTIONS] IMAGE [IMAGE...]

Save one or more images to a tar archive (streamed to STDOUT by default)

Options:
  -o, --output string   Write to a file, instead of STDOUT
ahmad@DockerMe: $
```

در مثال‌های زیر نحوه‌ی استفاده از این دستور مشخص شده است.

```
$ docker docker image ls
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE


$ docker load < busybox.tar.gz
Loaded image: busybox:latest


$ docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
busybox         latest       769b9341d937   7 weeks ago    2.489 MB


$ docker load --input fedora.tar
Loaded image: fedora:rawhide
Loaded image: fedora:20


$ docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
busybox         latest       769b9341d937   7 weeks ago    2.489 MB
```

fedora	rawhide	0d20aec6529d	7 weeks ago	387 MB
fedora	20	58394af37342	7 weeks ago	385.5 MB
fedora	heisenbug	58394af37342	7 weeks ago	385.5 MB
fedora	latest	58394af37342	7 weeks ago	385.5 MB

❖ دستور docker restart

با استفاده از این دستور یک یا تعدادی کانتینر را می‌توانید ریستارت کنید. این دستور تنها یک آپشن دارد به نام -t که با استفاده از آن می‌توانید برای ریستارت زمان تعیین کنید.

```
ahmad@DockerMe: $ docker restart --help

Usage:  docker restart [OPTIONS] CONTAINER [CONTAINER...]

Restart one or more containers

Options:
  -t, --time int   Seconds to wait for stop before killing the container (default 10)
ahmad@DockerMe: $
```

در مثال زیر نحوه‌ی استفاده از دستور نمایش داده شده است.

```
$ docker restart my_container

$ docker restart -t 10 my_container
```

❖ دستور docker kill

با استفاده از این دستور یک تا تعدادی از کانتینرها را می‌توان kill کرد. تنها آپشن این دستور s- می‌باشد که با استفاده از آن می‌توان سیگنال مد نظر خود را به کانتینر ارسال کرد.

```
ahmad@DockerMe: $ docker kill --help

Usage:  docker kill [OPTIONS] CONTAINER [CONTAINER...]

Kill one or more running containers

Options:
  -s, --signal string   Signal to send to the container (default "KILL")
ahmad@DockerMe: $
```

برخی از استفاده‌های این دستور در ادامه قرار داده شده است.

$ docker kill my_container

با استفاده از دستور kill -l در لینوکس می‌توانید لیست سیگنال‌های دستور kill را مشاهده کنید. شما هم می‌توانید عدد سیگنال مد نظرتون یا اختصار آن را قرار بدید. دقیقا همانند مثال زیر که SIGHUP همان HUP بوده که عدد سیگنال آن ۱ می‌باشد. هر سه دستور پایین دقیقا یک کار را انجام می‌دهند.

$ docker kill --signal=SIGHUP my_container

$ docker kill --signal=HUP my_container

$ docker kill --signal=1 my_container

در تصویر زیر فهرست تمام سیگنال‌های مربوط به دستور kill را مشاهده می‌کنید.

```
ahmad@DockerMe: $ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
ahmad@DockerMe: $
```

منبع برای دستور kill و منبع برای دستورdocker kill .

❖ دستور docker inspect

با استفاده از این دستور شما می‌توانید اطلاعات زیادی در مورد هر کدام از آبجکت‌های داکر داشته باشید. به عنوان مثال می‌توانید اطلاعات در مورد کانتینر و ایمیج و والیوم و یا نتورک‌های که ایجاد شده است داشته باشید.

```
ahmad@DockerMe: $ docker inspect --help

Usage:  docker inspect [OPTIONS] NAME|ID [NAME|ID...]

Return low-level information on Docker objects

Options:
  -f, --format string   Format the output using the given Go template
  -s, --size            Display total file sizes if the type is container
      --type string     Return JSON for specified type
ahmad@DockerMe: $
```

آپشن‌های مربوط به این دستور رو در تصویر بالا مشاهده می‌کنید. همانند دستورات قبل با استفاده از آپشن format می‌توانید خروجی دستور را تغییر داده و همانند چیزی که خودتون می‌خواهید آن را تنظیم کنید. اما بر خلاف دستورات قبلی جدول مشخص از موارد مربوط به این آپشن پیدا نکردم که در اختیار شما قرار بدم که دلیل اصلی آن نیز تعدد و تنوع این تنظیمات می‌تواند باشد. اما الگوی مشخص آن به این صورت است که اگر نام هر قسمت را در فرمت مشخص زیر قرار بدهید تمام اون قسمت را به شما نمایش می‌دهد. اما اگر مد نظر شما قسمتی مشخصی است تمام شاخه‌های آن را نیز فهرست نمایید. به مثال زیر توجه کنید.در دستور اول کل تنظیمات مربوط به شبکه این کانتینر را در خروجی قرار داده‌ام.

```
$docker inspect -f '{{.NetworkSettings}}' my_container {{
1e652cd4275553a99e5b8e51fab5740c37aee1df52f71e016c21fbe00d211d78 false  0 map[] /var/run/docker/netns/1e652cd42755
[] []} {3f93fe4ec940f332ce5e9ee25ba4866a3bdbb8d598338e72a4a59d2f0cfbdc5f 172.17.0.1  0 172.17.0.2 16  02:42:ac:11:00:02}
map[bridge:0xc4205220c0]}
```

با دستور زیر می‌توانید دستور بالا را به صورت json خروجی بگیرید.

```
$ docker inspect -f '{{json .NetworkSettings}}' my_container

{"Bridge":"","SandboxID":"1e652cd4275553a99e5b8e51fab5740c37aee1df52f71e016c21fbe00d211d78","HairpinMode":false,"Lin
kLocalIPv6Address":"","LinkLocalIPv6PrefixLen":0,"Ports":{},"SandboxKey":"/var/run/docker/netns/1e652cd42755","SecondaryIPA
ddresses":null,"SecondaryIPv6Addresses":null,"EndpointID":"3f93fe4ec940f332ce5e9ee25ba4866a3bdbb8d598338e72a4a59d2f0
cfbdc5f","Gateway":"172.17.0.1","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"IPAddress":"172.17.0.2","IPPrefixLen":16,"IPv6
Gateway":"","MacAddress":"02:42:ac:11:00:02","Networks":{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,
"NetworkID":"0f1ddf9690bbf50632242487d790a2b1e22f51e9e98b99c34ded3136c994fef1","EndpointID":"3f93fe4ec940f332ce5e
9ee25ba4866a3bdbb8d598338e72a4a59d2f0cfbdc5f","Gateway":"172.17.0.1","IPAddress":"172.17.0.2","IPPrefixLen":16,"IPv6Gat
eway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"MacAddress":"02:42:ac:11:00:02","DriverOpts":null}}}
```

در دستور دوم نیاز داشتم که ip کانتینر را به صورت مشخص جدا کنم و نمایش فقط بر اساس آن باشد. اگر از دستور inspect استفاده کنید تمام این قسمت‌ها به صورت زیرمجموعه یکدیگر قرار دارد که می‌توانید آنها به این صورت در کنار هم قرار دهید و

استفاده کنید. البته می‌توانید به اختصار فقط شاخه‌ی اصلی که همان NetworkSettings می‌باشد و سپس IPAddress را قرار دهید.

```
$ docker inspect -f '{{.NetworkSettings.Networks.bridge.IPAddress}}' my_container 172.17.0.2
$ docker inspect -f '{{.NetworkSettings.IPAddress}}' my_container 172.17.0.2
```

در دستور سوم برخی از موارد دیگه همانند اسم کانتینر را در ابتدا قرار داده و در انتهای فرمت نیز عبارت مد نظر خودم رو اضافه کردم.

```
$ docker inspect -f '{{.Name}} + {{.NetworkSettings.Networks.bridge.IPAddress}} +++ DockerMe.ir'  my_container
/my_container + 172.17.0.2 +++ DockerMe.ir
```

در این لینک و این لینک توضیحات خوبی برای انواع فرمت‌های خروجی ارائه شده است. شما می‌توانید مشخص کنید که این خروجی به چه صورت به شما نمایش داده شود. توضیح بیشتر این موضوع به نظرم از حوصله‌ی این متن خارج است شاید بعدا بر حسب نیاز بیشتر در موردش صحبت کنیم.

❖ دستور docker history

این دستور تاریخچه ایمیج را در اختیار ما قرار می‌دهد.

```
ahmad@DockerMe: $ docker history  --help

Usage:  docker history [OPTIONS] IMAGE

Show the history of an image

Options:
      --format string   Pretty-print images using a Go template
  -H, --human           Print sizes and dates in human readable format (default true)
      --no-trunc        Don't truncate output
  -q, --quiet           Only show numeric IDs
ahmad@DockerMe: $
```

این دستور دارای آپشن‌های زیر می‌باشد که در ادامه توضیح داده می‌شود.

آپشن q-: این آپشن تنها id ایمیج‌های واقع در تاریخچه را نمایش می‌دهد.

آپشنH- : این آپشن خروجی را متناسب با انسان نمایش می‌دهد که به صورت پیش‌فرض فعال است. در خروجی حجم ایمیج‌ها را به صورت صحیح و با MB یا GB نمایش می‌دهد در صورتی که این آپشن غیر فعال باشد تمام موارد بر اساس بایت نمایش می‌دهد.

آپشن no-trune : دیگر برخی از موارد را مختصر نمی‌کند و کامل نمایش می‌دهد.

آپشن format : با این آپشن شما می‌توانید خروجی را طبق نظر خودتون تنظیم کنید. مواردی که می‌توانید در خروجی آنها را قرار دهید به قرار زیر است.

Placeholder	Description
`.ID`	Image ID
`.CreatedSince`	Elapsed time since the image was created if `--human=true`, otherwise timestamp of when image was created
`.CreatedAt`	Timestamp of when image was created
`.CreatedBy`	Command that was used to create the image
`.Size`	Image disk size
`.Comment`	Comment for image

برای درک بهتر این دستور به مثال‌های زیر توجه کنید.

```
$ docker history busybox:latest

IMAGE          CREATED        CREATED BY                          SIZE          COMMENT
8c811b4aec35   12 days ago    /bin/sh -c #(nop)  CMD ["sh"]        0B            <missing>      12 days ago
/bin/sh -c #(nop) ADD file:5f0439d8328ab58c0…   1.15MB


$ docker history --format "{{.ID}}: {{.Size}} +++ DockerMe.ir" busybox:latest

8c811b4aec35: 0B +++ DockerMe.ir

<missing>: 1.15MB +++ DockerMe.ir
```

❖ دستور docker login

با استفاده از این دستور می‌توانید داخل ریجیستری لاگین کنید. اطلاعات مربوط به لاگین شما به صورت رمزنگاری شده در داکر کلاینت شما ذخیره می‌شود و تا زمانی که از آن ریجیستری logout نکنید این اطلاعات برقرار می‌باشد. نکته اینکه می‌توانید به چندین ریجیستری به صورت همزمان نیز لاگین کنید. می‌توانید از روش‌های مختلفی نیز برای قرار دادن پسورد و اطلاعات مربوط به لاگین خود نیز استفاده کنید. این موضوع شامل ریجیستری‌های عمومی و خصوصی می‌شود.

اطلاعات ورود و دسترسی شما به ریجیستری معمولا در مسیر زیر ذخیره می‌شود.

```
$HOME/.docker/config.json
```

آپشن -p : با استفاده از آن می‌توانید پسورد را در دستور قرار بدهید. این موضوع از نظر امنیتی مشکل داشته و پسورد در historyسرور باقی خواهد ماند.

آپشن password-stdin : پسورد را از روی یک فایل دریافت می‌کند.

آپشن-u : آپشن مربوط به حساب کاربری می‌باشد.

برای مشخص‌تر شدن موضوع به مثال‌های زیر توجه کنید. دقت کنید اگر اطلاعات کاربری را در دستور وارد شده قرار ندهید بعد از اعمال دستور این موارد که شامل user و pass می‌باشد را از شما سوال می‌کند.

```
$ docker login https://docker.dockerme.ir
$ docker login --username admin https://docker.dockerme.ir
$ cat test.txt | docker login --username admin --password-stdin https://docker.dockerme.ir
```

```
cat /home/ahmad/.docker/config.json
{
        "auths": {
                "docker.dockerme.ir": {
                "auth": "YWRtaW46WWFAQWJiYXM="
                }
        }
}
```

اگر لینک ریجیستری را نیز وارد نکنید به صورت پیش‌فرض به https://hub.docker.com وصل خواهد شد که یک ریجیستری عمومی است و برای لاگین به آن و قرار دادن ایمیج‌های خود در آن از طریق این لینک باید ثبت‌نام انجام دهید.

```
$ docker login

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.


$ cat test.txt | docker login --username ahmadrafiee --password-stdin

Login Succeeded


$ cat /home/ahmad/.docker/config.json

{
        "auths": {
                "docker.dockerme.ir": {
                "auth": "YWRtaW46WWFAQWJiYXM="
                },
                "https://index.docker.io/v1/": {
                "auth": "YWhtYWRyYWZpZWU6WWFAQWJiYXM="
                }
}
```

همانطور که ملاحضه می‌کنید در حال حاضر در دو تا ریجیستری لاگین هستم و اطلاعات ورود هر دو تا ریجیستری موجود می‌باشد.

❖ دستور docker logout

با این دستور از ریجیستری که لاگین کرده بودیم قبلا خارج می‌شود و اطلاعات ارتباط با آن برداشته می‌شود.

```
ahmad@DockerMe: $ docker logout --help

Usage:  docker logout [SERVER]

Log out from a Docker registry

Options:

ahmad@DockerMe: $
```

این دستور آپشن دیگری ندارد. اگر بدون لینک ریجیستری دستور را وارد کنید به صورت پیشفرض اگر اطلاعات مربوط به لاگین در ریجیستری داکر وجود داشته باشد آن را پاک می‌کند. به مثال زیر توجه کنید در ابتدا اطلاعات مربوط به لاگین وجود دارد و بعد از logout اون اطلاعات برداشته می‌شود.

```
$ cat /home/ahmad/.docker/config.json
{
        "auths":
        {
        "docker.dockerme.ir": {
        "auth": "YWRtaW46WWFAQWJiYXM=" },
        "https://index.docker.io/v1/": {
        "auth": "YWhtYWRyYWZpZWU6WWFAQWJiYXM=" }
        }
}

$ docker logout
Removing login credentials for https://index.docker.io/v1/

cat /home/ahmad/.docker/config.json
{
        "auths":
        {
        "docker.dockerme.ir": {
        "auth": "YWRtaW46WWFAQWJiYXM=" }
        }
}
```

❖ دستور docker logs

با استفاده از این دستور لاگ کانتینر را مشاهده می‌کنید. به صورت پیش فرض تمام لاگ کانتیری که در مقابل دستور باشد را به شما نمایش می‌دهد.

```
ahmad@DockerMe: $ docker logs --help

Usage:  docker logs [OPTIONS] CONTAINER

Fetch the logs of a container

Options:
      --details        Show extra details provided to logs
  -f, --follow         Follow log output
      --since string   Show logs since timestamp (e.g. 2013-01-02T13:23:37) or relative (e.g. 42m for 42 minutes)
      --tail string    Number of lines to show from the end of the logs (default "all")
  -t, --timestamps     Show timestamps
      --until string   Show logs before a timestamp (e.g. 2013-01-02T13:23:37) or relative (e.g. 42m for 42 minutes)
ahmad@DockerMe: $
```

نکته‌ی مهم اینکه اگر لاگ درایور سرویس داکر خود را تغییر داده باشید ممکن است این دستور برای شما کار نکند. این دستور تنها در زمان استفاده از لاگ درایور json-file یا journald کار می‌کند. در این لینک در مورد درایور لاگ داکر بیشتر صحبت شده است.

آپشن details : موارد اضافه‌تری را در لاگ به شما نمایش می‌دهد.

آپشن f– : با استفاده از این آپشن لاگ کانتینر را فالو می‌کند و همواره آن را نمایش می‌دهد تا زمانی که شما خودتون از این دستور خارج شوید.

آپشن since : با این آپشن می‌توان زمان تعیین کرد به این صورت که از این زمان به بعد لاگ را در اختیار من قرار بده.

آپشن tail : این دستور تعداد خط‌های آخر را مشخص می‌کند مثلا ۱۰۰ خط آخر را نمایش بده یا اینکه ۱۰ خط آخر لاگ را به من نمایش بده. حال ترکیب این آپشن با آپشن -f همواره ۱۰ خط آخر لاگ را به شما نمایش می‌دهد. اگر از این آپشن استفاده نکنید به صورت پیش‌فرض تمام خط‌های لاگ را به شما نمایش می‌دهد.

آپشن t- : زمان اتفاقات را نیز در لاگ به شما نمایش می‌دهد.

آپشن until : با این آپشن می‌توان زمان تعیین کرد به این صورت که تا این زمان لاگ را به من نشان بده.

نکته‌ی مهم اینکه این دستور بسیار مهم می‌باشد و به شدت به فرآیند رفع مشکل کمک می‌کند. برخی از ترکیب‌های خوب این دستور در ادامه آورده شده است.

```
$ docker logs -f my-container
```

```
$ docker logs -f --tail 100 my-container
$ docker logs -f --tail 100 -t my-container
```

❖ دستور docker port

لیست پورت‌های که از سرور به کانتینر مپ شده است را نمایش می‌دهد. هیچ‌گونه آپشنی هم ندارد.

```
ahmad@DockerMe: $ docker port --help

Usage:  docker port CONTAINER [PRIVATE_PORT[/PROTO]]

List port mappings or a specific mapping for the container

Options:

ahmad@DockerMe: $
```

به مثال‌های زیر توجه کنید.

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
b650456536c7        busybox:latest      top                 54 minutes ago      Up 54 minutes       0.0.0.0:1234->
```

```
$ docker port test
7890/tcp -> 0.0.0.0:4321
9876/tcp -> 0.0.0.0:1234
```

```
$ docker port test 7890/tcp
0.0.0.0:4321
```

```
$ docker port test 7890/udp
2014/06/24 11:53:36 Error: No public port '7890/udp' published for test
```

```
$ docker port test 7890
0.0.0.0:4321
```

❖ دستور docker diff

توضیحی از تغییرات داخل فایل یا دایرکتوری‌های داخل کانتینر در اختیار ما قرار می‌دهد. این دستور نیز هیچ‌گونه آپشنی ندارد.

```
ahmad@DockerMe: $ docker diff --help

Usage:  docker diff CONTAINER

Inspect changes to files or directories on a container's filesystem

Options:

ahmad@DockerMe: $
```

برای نمایش تغییرات از سمبلیک‌هایی استفاده می‌کند که در ادامه توضیح داده می‌شود.

سمبلA : یعنی اینکه یک فایل یا دایرکتوری اضافه شده است.

سمبلD : یعنی اینکه فایل یا دایرکتوری پاک شده است.

سمبلC : یعنی اینکه فایل یا دایرکتوری تغییر کرده است.

به صورت زیر از این دستور استفاده می‌شود.

```
$ docker diff my-container
C /tmp
D /tmp/hsperfdata_jenkins
A /tmp/hsperfdata_jenkins/8
D /tmp/jetty-0.0.0.0-8080-war-_-any-4465523861556992604.dir
A /tmp/jffi8867713240318490067.tmp
D /tmp/jna--1712433994
A /tmp/winstone1616795870189931273.jar
```

❖ دستور docker push

با استفاده از این دستور می‌توان ایمیج‌ها را داخل ریجیستری قرار داد.

```
ahmad@DockerMe: $ docker push -h
Flag shorthand -h has been deprecated, please use --help

Usage:  docker push [OPTIONS] NAME[:TAG]

Push an image or a repository to a registry

Options:
      --disable-content-trust   Skip image signing (default true)
ahmad@DockerMe: $
```

تنها آپشنی که این دستور دارد disable-content-trust می‌باشد که مربوط به چک و بررسی ایمیج می‌باشد.

نکته‌ی مهم اینکه قبلا باید حتما داخل یک ریجیستری وارد شده باشید تا بتوانید که ایمیج‌های خود را داخل آن قرار دهید.

❖ دستور docker start

با این دستور می‌توان کانتینر و یا کانتینرهایی که stop شده است و یا اینکه ایجاد شده اما run نباشد را start کرد.

```
ahmad@DockerMe: $ docker start -h
Flag shorthand -h has been deprecated, please use --help

Usage:  docker start [OPTIONS] CONTAINER [CONTAINER...]

Start one or more stopped containers

Options:
  -a, --attach              Attach STDOUT/STDERR and forward signals
      --detach-keys string  Override the key sequence for detaching a container
  -i, --interactive         Attach container's STDIN
ahmad@DockerMe: $
```

آپشن a- : با استفاده از این آپشن بعد از start کانتینر به آن متصل شده و می‌توان در کانتینر دستور اجرا کرد.

آپشن i- : با استفاده از این آپشن به کانتینر متصل می‌شود.

❖ دستور docker stop

با استفاده از این دستور می‌توان کانتینر و یا کانتینرهای در حال کار را stop کرد. این دستور تنها یک آپشن دارد و آن هم مربوط به t- می‌باشد که با استفاده از آن می‌توان برای متوقف کردن کانتینر زمان در نظر گرفت که بعد از طی شدن آن زمان کانتینر را stop می‌شود. به صورت پیش‌فرض این زمان ۱۰ ثانیه در نظر گرفته شده است.

```
ahmad@DockerMe: $ docker stop -h
Flag shorthand -h has been deprecated, please use --help

Usage:  docker stop [OPTIONS] CONTAINER [CONTAINER...]

Stop one or more running containers

Options:
  -t, --time int   Seconds to wait for stop before killing it (default 10)
ahmad@DockerMe: $
```

دستور docker version

با استفاده از این دستور شما نسخه‌ی نصب شده‌ی داکر را می‌توانید مشاهده کنید. این دستور تنها یک آپشن دارد که با استفاده از آن می‌توان فرمت خروجی پیش‌فرض را تغییر داد.

```
ahmad@DockerMe: $ docker version -h
Flag shorthand -h has been deprecated, please use --help

Usage:  docker version [OPTIONS]

Show the Docker version information

Options:
  -f, --format string   Format the output using the given Go template
ahmad@DockerMe: $
```

از این دستور می‌توان به حالت‌های دیگه هم استفاده کرد. به مثال‌های زیر توجه کنید. در دستور اول شما تنها نسخه‌ی سرور را خواهید داشت و هیچ توضیح دیگه‌ای داده نخواهید داشت و در دستور دوم تمام توضیحات در قالب یک فایل json در اختیار شما قرار می‌گیرد.

```
docker version --format '{{.Server.Version}}'
docker version --format '{{json .}}'
```

❖ دستور docker pause

با استفاده از این دستور شما می‌توانید یک یا تعدادی از کانتینرها را pause کنید. این دستور هیچ‌گونه آپشنی ندارد.

```
ahmad@DockerMe: $ docker pause --help

Usage:  docker pause CONTAINER [CONTAINER...]

Pause all processes within one or more containers

Options:
```

❖ دستور docker unpause

با استفاده از این دستور می‌تواند یک یا تعدادی از کانتینر را unpause کنید. این دستور نیز هیچ‌گونه آپشنی ندارد.

```
ahmad@DockerMe: $ docker unpause --help

Usage:  docker unpause CONTAINER [CONTAINER...]

Unpause all processes within one or more containers

Options:
```

❖ دستور docker top

با استفاده از این دستور شما می‌توانید پروسه‌های جاری هر کانتینر را مشاهده کنید. این دستور آپشنی ندارد و به صورت زیر استفاده می‌شود. اما می‌توانید آپشن‌های مربوط به ps را در مقابل نام کانتینر قرار دهید تا در خروجی مورد نظر اعمال شود. همانطور که مشاهده می‌کنید در دستور دوم اطلاعات بیشتر قرار داده شده است.

```
$docker top test2
UID     PID     PPID    C    STIME    TTY     TIME      CMD
root    27975   2795    0    17:51    pts/0   00:00:00  sh

$ docker top test2 -aux
USER    PID     %CPU  %MEM  VSZ   RSS  TTY    STAT  START  TIME  COMMAND
root    27975   0.0   0.0   1244  4    pts/0  Ss+   17:51  0:00  sh
```

```
ahmad@DockerMe: $ docker top --help
Usage:  docker top CONTAINER [ps OPTIONS]
Display the running processes of a container
Options:
```

❖ دستور docker stats

با استفاده از این دستور به صورت live stream می‌تونیم میزان منابع مورد استفاده هر کانتینر را مشاهده کنیم. این دستور برای مانیتورینگ منابع استفاده شده سرور توسط کانتینرها بسیار مناسب است. سیستم‌های مانیتورینگ با ارتباط با سرویس‌دهنده‌ی داکر و استفاده از این دستور وضعیت در لحظه‌ی تمام کانتینرها را بدست می‌آورند.

```
ahmad@DockerMe:~$ docker stats --help

Usage:  docker stats [OPTIONS] [CONTAINER...]

Display a live stream of container(s) resource usage statistics

Options:
  -a, --all            Show all containers (default shows just running)
      --format string  Pretty-print images using a Go template
      --no-stream      Disable streaming stats and only pull the first result
      --no-trunc       Do not truncate output
ahmad@DockerMe:~$
```

در جدول زیر توضیح هر قسمت خروجی دستور stats به همراه توضیح آن مشخص شده است.

Column name	Description
`CONTAINER ID` and `Name`	the ID and name of the container
`CPU %` and `MEM %`	the percentage of the host's CPU and memory the container is using
`MEM USAGE / LIMIT`	the total memory the container is using, and the total amount of memory it is allowed to use
`NET I/O`	The amount of data the container has sent and received over its network interface
`BLOCK I/O`	The amount of data the container has read to and written from block devices on the host
`PIDs`	the number of processes or threads the container has created

بدون آپشن: به صورت مداوم stats تمام کانتینرها را نمایش می‌دهد و اگر جلوی دستور نام کانتینری را وارد کنید تنها stats همان کانتینر را نمایش می‌دهد.

آپشن no-trunc : این آپشن تمام موارد همانندid ها و ... را به صورت کامل بدون اینکه خلاصه کند نمایش می‌دهد.

آپشن no-stream : بعد از استفاده از این آپشن فقط یک بار stats را نمایش می‌دهد و دیگر به صورت مداوم نمایش نمی‌دهد.

آپشن all : موارد مربوط به تمام کانتینرها را نمایش می‌دهد. این آپشن به صورت پیش‌فرض فعال است.

آپشن format : فرمت خروجی را همانطور که مد نظر خودمان است کانفیگ می‌کنیم. برای ایجاد فرمت مد نظر می‌توانیم از موارد جدول زیر استفاده کنیم.

Placeholder	Description
`.Container`	Container name or ID (user input)
`.Name`	Container name
`.ID`	Container ID
`.CPUPerc`	CPU percentage
`.MemUsage`	Memory usage
`.NetIO`	Network IO
`.BlockIO`	Block IO
`.MemPerc`	Memory percentage (Not available on Windows)
`.PIDs`	Number of PIDs (Not available on Windows)

برای بهتر مشخص شدن موضوع به مثال زیر توجه کنید. در این مثال نام و میزان مصرف پردازنده و رم کانتینر test در فرمت خروجی در نظر گرفته شده است.

```
docker stats --format "table {{.Container}}{{.CPUPerc}}t{{.MemUsage}}" test
CONTAINER        CPU %        MEM USAGE / LIMIT
  test              0.20%        383.3MiB / 7.734GiB
```

فرمت پیش‌فرض دستور stats در لینوکس به صورت زیر می‌باشد.

```
"table{{.ID}}t{{.Name}}t{{.CPUPerc}}t{{.MemUsage}}t{{.MemPerc}}t{{.NetIO}}t{{.BlockIO}}t{{.PIDs}}"
```

❖ دستور docker update

با استفاده از این دستور می‌توان برخی از کانفیگ‌های یک یا تعدادی از کانتینرها را تغییر داد.

```
ahmad@DockerMe:~$ docker update --help

Usage:  docker update [OPTIONS] CONTAINER [CONTAINER...]

Update configuration of one or more containers

Options:
      --blkio-weight uint16        Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
      --cpu-period int             Limit CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int              Limit CPU CFS (Completely Fair Scheduler) quota
      --cpu-rt-period int          Limit the CPU real-time period in microseconds
      --cpu-rt-runtime int         Limit the CPU real-time runtime in microseconds
  -c, --cpu-shares int             CPU shares (relative weight)
      --cpus decimal               Number of CPUs
      --cpuset-cpus string         CPUs in which to allow execution (0-3, 0,1)
      --cpuset-mems string         MEMs in which to allow execution (0-3, 0,1)
      --kernel-memory bytes        Kernel memory limit
  -m, --memory bytes               Memory limit
      --memory-reservation bytes   Memory soft limit
      --memory-swap bytes          Swap limit equal to memory plus swap: '-1' to enable unlimited swap
      --restart string             Restart policy to apply when a container exits
ahmad@DockerMe:~$
```

با استفاده از این دستور می‌توان آپشن‌های مربوط به پردازنده و رم هر کانتینر را تغییر داد و سیاست ریستارت کانتینر را نیز می‌شود به روز کرد. توی چند تا مثال نحوه‌ی استفاده از این دستور را بررسی می‌کنیم.

```
$ docker update --kernel-memory 80M my_container
```

```
$ docker update --cpu-shares 512 -m 300M my_container
```

```
$ docker update --cpu-shares 512 my_container
```

❖ دستور docker container

در این قسمت در مورد دستور docker container صحبت می‌کنیم که به واسطه‌ی آن می‌توان کانتینرها را مدیریت کرد.

```
ahmad@DockerMe:~$ docker container --help

Usage:  docker container COMMAND

Manage containers

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  inspect     Display detailed information on one or more containers
  kill        Kill one or more running containers
  logs        Fetch the logs of a container
  ls          List containers
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  prune       Remove all stopped containers
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  run         Run a command in a new container
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes

Run 'docker container COMMAND --help' for more information on a command.
```

همانطور که مشاهده می‌کنید این دستور دارای امکانات زیادی می‌باشد که می‌توان با استفاده از آن کانتینرها را مدیریت کرد. این دستورات هم‌پوشانی کامل با دستورات توضیح داده شده‌ی قبلی دارد و دقیقا همان کارها را انجام می‌دهد. تمام دستورات به صورت جداگانه توضیح داده شده است و برخی از آپشن‌ها که قبلا توضیح داده نشده همانند ls یک alias از دستور ps می‌باشد و همان کار را خواهد کرد.

به عنوان مثال دستور docker container ps با دستور docker ps کاملا برابر و یکسان می‌باشند.

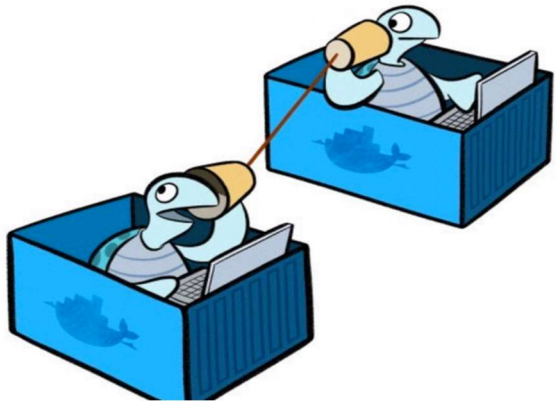
❖ دستور docker image

این دستور نیز کاملا با دستورات مربوط به ایمیج داکر یکسان می‌باشد.

```
ahmad@DockerMe:~$ docker image --help

Usage:  docker image COMMAND

Manage images

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
```

به عنوان مثال دستور docker image rm یک alias از دستور docker rmi می‌باشد.

۱۶.توضیح کوتاه اما بسیار مفید در مورد شبکه‌ی داکر



یکی از مواردی که معمولا کمتر بررسی می‌شود اما در سرویس‌های عملیاتی با داکر از مهمترین موارد است شبکه‌ی داکر می‌باشد که شامل ارتباطات بین کانتینرها در یک هاست، ارتباطات بین کانتیرها با هاست خود و ارتباطات بین کانتینرها با چندین هاست و کانتینرهای آن می‌باشد. البته که این مبحث به شدت پیچیده و گسترده است و زمانی که شما در حجم بالا بخواهید از آن استفاده کنید نیاز است تا چندین کتاب را در مورد آن بررسی کنید. به عنوان مثال این کتاب به خوبی و با حجم کم به این موضوع پرداخته است.

داکر به صورت پیش‌فرض بعد از نصب ۳ شبکه برای خود ایجاد می‌کنه که همانند تصویری زیر می‌باشد.

docker network ls

```
$ docker network ls

NETWORK ID          NAME                DRIVER
7fca4eb8c647        bridge              bridge
9f904ee27bf5        none                null
cf03ee007fb4        host                host
```

شما هر زمان نیاز بود که کانتینر راه‌اندازی کنید با آپشن network– می‌توانید کانتینر خود را به هر کدام از این شبکه‌ها متصل نمایید.

❖ شبکه‌ی bridge

داکر به صورت پیش‌فرض شبکه‌ی مورد نیاز برای ارتباط کانتینر با هاست را بر روی درایور bridge پیکربندی می‌کند و در دسترس قرار می‌دهد. اگر برای کانتینر خود شبکه‌ای پیکربندی نکنید به صورت پیش‌فرض به شبکه‌ای به نام docker0 متصل

می‌شود. این شبکه همواره بعد از نصب داکر بر روی سیستم‌عامل شما ایجاد می‌شود که مربوط به سرویس‌دهنده‌ی داکر می‌باشد و معمولا هم subnet آن ۱۷۲,۱۷,۰,۰ می‌باشد. البته این موارد به صورت کامل قابل پیکربندی است و شما می‌توانید سرویس داکر خود را به گونه‌ای راه‌اندازی کنید که شبکه‌ی bridge آن دارای تنظیمات دیگری باشد.

به صورت پیش‌فرض این bridge بر روی سیستم‌عامل شما دارای ip با شماره‌ی ۱۷۲,۱۷,۰,۱ می‌باشد که کانتینرهای شما در همین رنج ip خواهند گرفت. به تصویر زیر توجه کنید:

```
$ ip addr show

docker0   Link encap:Ethernet  HWaddr 02:42:47:bc:3a:eb
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:47ff:febc:3aeb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9001  Metric:1
          RX packets:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1100 (1.1 KB)  TX bytes:648 (648.0 B)
```

❖ شبکه‌ی none

این شبکه برای زمانی است که شما نیاز ندارید داخل کانتینر خود کارت شبکه‌ای داشته باشید. این موضوع بسیار کم کاربرد می‌باشد اما این امکان فراهم است که شما کانتینری راه‌اندازی کنید که هیچ‌گونه ارتباط شبکه‌ای حتی با هاست خودش هم نداشته باشد. در تصویر زیر داخل یک کانتینر را نشان می‌دهد که با شبکه‌ی none راه‌اندازی شده است.

```
$ docker attach nonenetcontainer

root@0cb243cd1293:/# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

root@0cb243cd1293:/# ip -4 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever

root@0cb243cd1293:/#
```

❖ شبکه‌ی host

در این حالت کانتینر از شبکه‌ی هاست استفاده می‌کند و در این حالت هیچ ایزولیشنی بین هاست و کانتینر وجود نخواهد داشت. به عنوان مثال اگر پورت ۸۰ یا ۳۳۰۶ از کانتینر در دسترس باشد به صورت کامل همان پورت هاست خواهد بود و تفاوتی بین این دو نیست و به گونه‌ای به صورت کامل کانتینر تمام مواردش بر روی هاست قرار خواهد گرفت.

نکته‌ی مهم : به صورت پیش‌فرض شبکه‌ی bridge به کانتینرها متصل می‌گردد و اگر بخواهید از دو شبکه‌ی دیگر استفاده کنید حتما باید آنها را در راه‌اندازی کانتینر به صورت آپشن وارد کنید. این موضوع به صورت کامل قابل پیکربندی است و می‌توانید همانند تمام موارد دیگر پیش‌فرض شبکه را نیز تغییر دهید.

```
docker info | grep Network  Network: bridge host macvlan null overlay
```

داکر و port های کانتینرها:

برای port ها در داکر دو مکانیزم وجود دارد یکی exposing و دیگری publishing می‌باشد.

همواره با استفاده از آپشن expose در داکرفایل می‌توان مشخص کرد چه پورت‌هایی از این ایمیج در حال سرویس‌دهی می‌باشند. این آپشن در دستور docker run نیز وجود دارد و قابل استفاده می‌باشد. اما دقت کنید که expose هیچ پورتی را در دسترس قرار نمی‌دهد.

آپشن publish برای در دسترس قرار دادن پورت‌های کانتینر در هاست و یا بیرون آن می‌باشد. با استفاده از این آپشن که با p- نیز در دسترس است می‌توان پورت‌های داخل کانتینر را بر روی پورت‌های داخل هاست map کرد و بر اساس نیازی که وجود دارد آنها را به ip های مختلف bind نمود. به عنوان مثال پورت ۳۳۰۶ تنها بر روی شبکه لوکال هاست bind شود و در دسترس باشد اما پورت ۸۰ برای تمامیip ها و شبکه‌ها در دسترس باشد. به این مثال‌ها توجه کنید.

در مثال اول پورت ۸۰ را به یک رندوم پورت که اینجا ۳۲۷۶۸ می‌باشد map کرده است که دسترسی آن برای تمامی شبکه‌ها می‌باشد و بر روی ۰,۰,۰,۰ bind شده است.

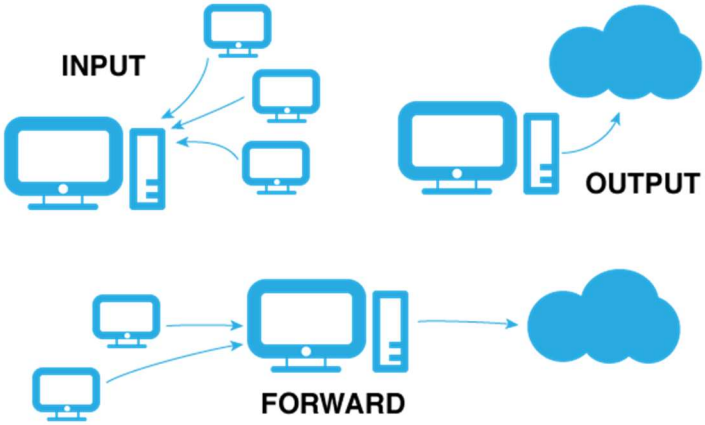
```
"nginx -g 'daemon ..."   43 hours ago       Up About a minute   443/tcp, 0.0.0.0:32768->80/tcp   nginx
```

در مثال دوم علاوه بر اینکه برای پورت ۸۰ پورت ۸۰۸۰ را در نظر گرفته و دیگر رندوم پورت نیست تنها بر روی شبکه لوکال سرور هم در دسترس قرار داده شده است.

```
"nginx -g 'daemon ..."   43 hours ago       Up 3 seconds      443/tcp, 127.0.0.1:8080->80/tcp   nginx
```

در سناریوهای پیاده‌سازی داکر بیشتر در این خصوص باهم صحبت خواهیم کرد.

۱۷. داکر و iptables (docker and iptables)



یک نکته‌ی مهم دیگه که خیلی باید به آن توجه ویژه‌ای داشت تعامل داکر با iptables است که سرویس داکر برای سهولت کار با آن دسترسی‌های مورد نیاز را روی سرویس iptables که فایروال لینوکس می‌باشد ایجاد می‌کند و برای این کار یک chain مخصوص برای خود ایجاد کرده و دسترسی‌های لازم را در آن قرار می‌دهد که ممکن است این موضوع با سیاست‌های امنیتی شما بر روی سرور متفاوت باشد از این رو باید همواره به این موضوع توجه ویژه‌ای داشته باشید. داکر به صورت پویا همواره با اضافه و کم شدن کانتینرها و سرویس‌ها و ... بر روی iptables این دسترسی‌ها را ایجاد می‌کند.

نکته : در این سایت آموزش کامل لینوکس به زبان فارسی و البته کاربردی داده می‌شود که پیشنهاد می‌کنم حتما قسمت iptables آن را بررسی کنید تا با شناخت بیشتر این سرویس مهم و حیاتی بتوانید به خوبی از آن استفاده کنید.

مرور برخی از قابلیت‌های مهم:

- می‌توانید هر تعدادی کارت شبکه که نیاز دارید ایجاد کنید.
- می‌توان کانتینر در حال کار را از شبکه‌ای جدا و به شبکه‌ی دیگری متصل کرد.
- می‌توان یک کانتینر را به تعدادی از شبکه‌ها متصل کنید.
- می‌توانید هر شبکه‌ای که ایجاد می‌کنید طبق نیاز خودتون به صورت کامل پیکربندی کنید.
- می‌توانید به هر کانتینر static ip بدهید.
- با استفاده از آپشن link– می‌توانید بین کانتیرها ارتباط برقرار کرده به صورتی که آنها با اسم با یکدیگر ارتباط داشته باشند.
- شما می‌توانید برای سرویس داکر و کانتینرها پروکسی تنظیم کنید و در صورت نیاز از آنها استفاده کنید.

برای داکر درایورهای مختلفی برای شبکه وجود دارد که در پست دیگری به تفصیل توضیح داده می‌شود.

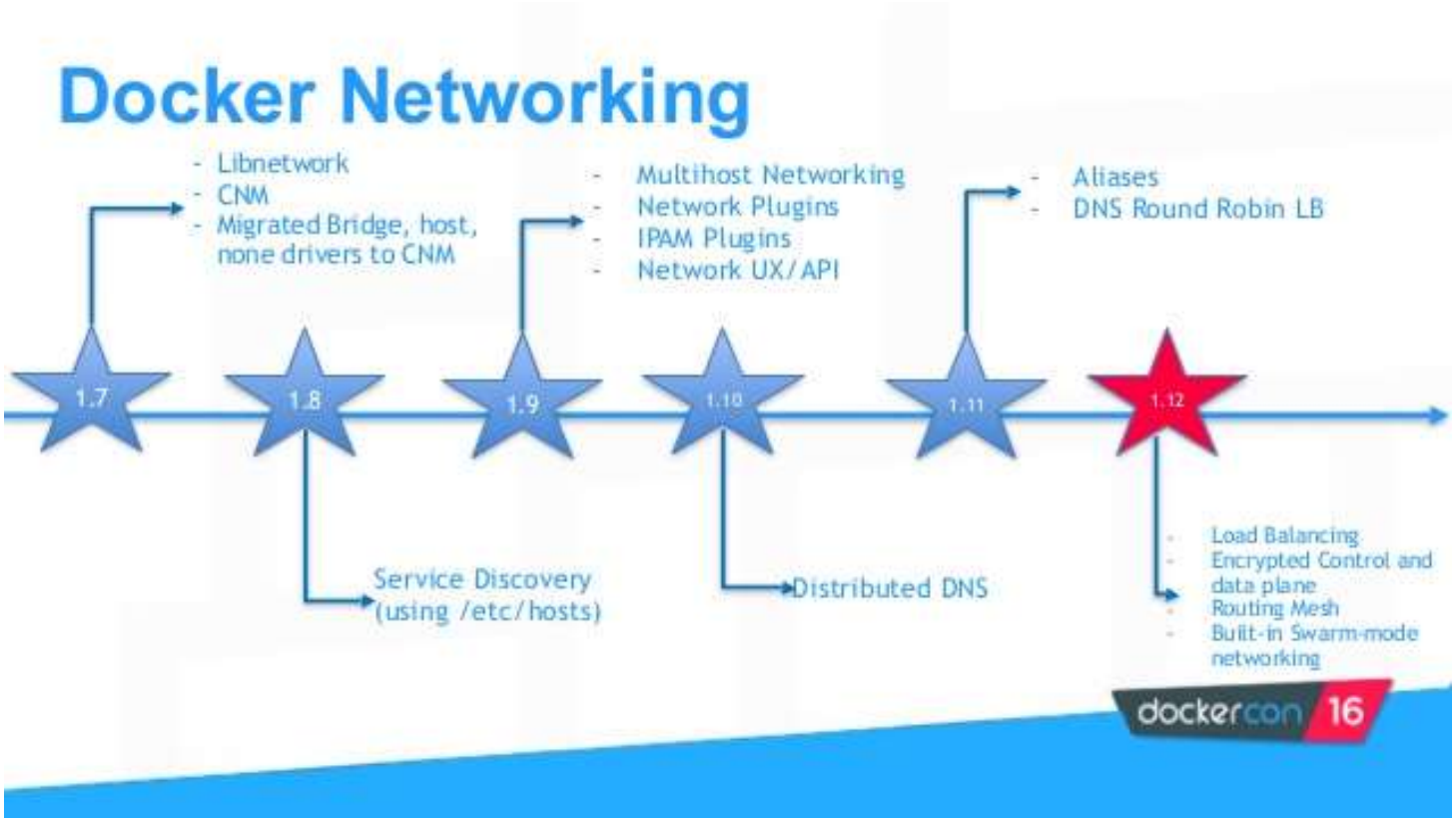
۱۸. استفاده از DNS سرور داخلی داکر

طرح مشکل:

همواره حین راه‌اندازی سرویس و یک محصول نیاز است تا بین ماژول‌های مختلف ارتباط برقرار شود و نتیجه‌ی این ارتباط می‌تواند محصول نهایی را شکل دهد. به عنوان مثال یک سایت وردپرس که شامل سرویس وردپرس و یک سرویس دیتابیس می‌باشد. ارتباط بین این دو تا سرویس محصول نهایی که همان سایت مد نظر است را شکل می‌دهد. این ارتباط می بایست همواره سریع و پایدار و امن باشد که یکی از مشکلات اصلی resolve شدن این سرویس‌ها است. حال اگر برای هر سرویس یک name در DNS Server تنظیم شود این ارتباط بهتر برقرار می‌شود. خیلی مهم است که این ارتباط امن نیز باشد و کسی از بیرون سرویس به آن دسترسی نداشته باشد.

حالا شما در نظر داشته باشید با استفاده از داکر تمام پروسه‌های خود را جدا کرده و آنها را ایزوله کردید این مشکل بیشتر خودش را نشان خواهد داد که برای این کار نیاز است تا ارتباطات خود را مدیریت کنید و بین کانتینرهای مختلف ارتباط درست و سریع برقرار کنید. سرویس داکر از نسخه‌ی ۱,۱۰ به بعد با ایجاد یک DNS Server توزیع شده به خوبی این امکان را فراهم کرده است که در ادامه به توضیح نحوه‌ی استفاده از آن می‌پردازیم.

در تصویر زیر سیر تکامل شبکه‌ی داکر را به ازای نسخه‌های مختلف سرویس داکر مشاهده می‌کنید.



استفاده از DNS سرور داخلی داکر:

از نسخه‌ی ۱,۱۰ داکر به بعد این امکان فراهم شد که dns server به صورت داخلی برای داکر وجود داشته باشد که با استفاده از آن برخی از موارد تسهیل می‌شود. بعد از این نسخه این امکان فراهم بود که به صورت داخلی service discovery برای تمامی کانتینرها با توجه به اسم کانتینر و یا net-alias و یا لینک‌های آن وجود داشته باشد.

سرویس DNS داخلی داکر با ip به شماره‌ی ۱۲۷,۰,۰,۱۱ در دسترس می‌باشد و همواره داخل فایل resolv.conf کانتینرها این nameserver پیکربندی شده است. اگر داخل یک کانتینر بخواهید کانتینر دیگه‌ای که در همان شبکه هست را صدا کنید تنها نیاز است که نام یا ID آن را استفاده کنید این DNS Server ip آن را پیدا کرده و در اختیار شما قرار می‌دهد. در کل این Dns Serverاطلاعات تمامی کانتینرهای داخل یک شبکه را دارد و به راحتی می‌توانید از آن استفاده کنید.

راه‌های مختلفی برای ارتباط با این DNS سرور داخلی داکر وجود دارد و می‌توان آن را کانفیگ کرد که در ادامه به برخی از این آپشن‌ها اشاره می‌شود.

نام کانتینر : با استفاده از این آپشن شما برای کانتینر خود یک نام انتخاب می‌کنید که به صورت کامل می‌توانید با همان نام در شبکه‌ی خود کانتینر آن را صدا کنید. البته به صورت خودکار نام برای هر کانتینر ایجاد می‌شود که آن نام نیز معتبر است. با ID هر کانتینر هم می‌توان آن را صدا کرد و بررسی نمود.

–name=CONTAINER-NAME

نام alias : این نام در شبکه‌ی کانتینر معتبر بوده و آن کانتینر علاوه بر نام خود با این نام نیز شناخته می‌شود. شما با اضافه کردن این آپشن می‌توانید برای کانتینر خود یک نام دیگر در DNS قرار دهید.

–network-alias=ALIAS or –alias=ALIAS

ایجاد لینک بین کانتینرها : با استفاده از این آپشن نیز می‌توانید به کانتینر که ایجاد کردید ارتباط برقرار کنید. با این آپشن می‌توان به کانتینرهایی که در یک شبکه با کانتینر شما نیست ارتباط برقرار کرد.

–link=CONTAINER_NAME:ALIAS

تنظیم ip برای DNS سرور : با استفاده از این آپشن شما می‌توانید برای DNS سرور کانتینر خود IP قرار دهید. این ip در فایل resolv.conf به روز می‌شود.

–dns=[IP_ADDRESS…]

تنظیم Domain برای DNS سرور : با استفاده از این آپشن می‌توانید برای DNS سرور کانتینر خود domain قرار دهید و با استفاده از آن می‌توانید تغییراتی در DNS سرور کانتینر خود ایجاد کنید.

–dns-search=DOMAIN…

قرار دادن آپشن‌های دیگر در DNS سرور : با استفاده از این آپشن شما می‌توانید داخل کانتینر خود و فایل resolv.conf آپشن‌هایی که مجاز است را قرار دهید.

–dns-opt=OPTION…

در کل این موضوع خیلی مهم است که بتوانید به خوبی با DNS Server ارتباط برقرار کنید و از آن استفاده کنیم. با دقت مناسب در استفاده از این آپشن‌ها به خوبی می‌توان بین کانتینرها ارتباط برقرار کرد.

۱۹.دستورات مدیریتی داکر– docker network

این قسمت در مورد docker network می‌خواهیم صحبت کنیم. این دستور به ما کمک خواهد کرد که شبکه‌ی ارتباطی docker و کانتیرهای آن را مدیریت کنیم و ارتباط کانتینرها با یکدیگر و هاست خود و دیگر هاست‌ها را پیکربندی نماییم.

```
ahmad@DockerMe:~$ docker network --help

Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.
ahmad@DockerMe:~$
```

این دستور ۷ تا آپشن دارد که تعدادی از آنها خیلی اهمیت دارد و بسیار پر مصرف می‌باشد.

❖ دستور docker network connect

با استفاده از این آپشن می‌توانید یک کانتینر را به یک شبکه که ایجاد کرده‌اید متصل نمایید. این دستور خودش دارای آپشن‌هایی هست که با استفاده از آن می‌توان کانتینر را به شبکه متصل کرد.

```
ahmad@DockerMe:~$ docker network connect --help

Usage:  docker network connect [OPTIONS] NETWORK CONTAINER

Connect a container to a network

Options:
      --alias strings           Add network-scoped alias for the container
      --ip string               IPv4 address (e.g., 172.30.100.104)
      --ip6 string              IPv6 address (e.g., 2001:db8::33)
      --link list               Add link to another container
      --link-local-ip strings   Add a link-local address for the container
ahmad@DockerMe:~$
```

با استفاده از آپشن‌ها می‌توان به کانتینر alias داد و برای آن ip نسخه‌ی ۴ و ۶ قرار داد و بین کانتینرها لینک ایجاد کرد. با استفاده از این آپشن می‌توان کانتینرهایی که ایجاد شده است را به شبکه‌های مختلف که ساخته شده است متصل کرد. به مثال‌های زیر توجه کنید:

```
docker network connect network1 container1 docker network connect --ip 10.10.36.122 network1 container1 docker network conne
ct --link container2:c2 network1 container2 docker network connect --alias db --alias mysql network1 container2
```

در مثال‌های بالا مشاهده می‌کنید که کانتینر در حال کار را به یک شبکه متصل کرده که در این راستا کانفیگ‌های دیگری اعم از IP و لینک و موارد این‌چنینی را انجام داده است.

❖ دستور docker network create

با استفاده از این آپشن شما می‌توانید یک network ایجاد کنید. این دستور نیز خودش دارای آپشن‌های زیادی می‌باشد که با استفاده از آنها می‌توانید network را با جزئیات بیشتری ایجاد کنید.

```
ahmad@DockerMe:~$ docker network create --help

Usage:  docker network create [OPTIONS] NETWORK

Create a network

Options:
      --attachable           Enable manual container attachment
      --aux-address map      Auxiliary IPv4 or IPv6 addresses used by Network driver (default map[])
      --config-from string   The network from which copying the configuration
      --config-only          Create a configuration only network
  -d, --driver string        Driver to manage the Network (default "bridge")
      --gateway strings      IPv4 or IPv6 Gateway for the master subnet
      --ingress              Create swarm routing-mesh network
      --internal             Restrict external access to the network
      --ip-range strings     Allocate container ip from a sub-range
      --ipam-driver string   IP Address Management Driver (default "default")
      --ipam-opt map         Set IPAM driver specific options (default map[])
      --ipv6                 Enable IPv6 networking
      --label list           Set metadata on a network
  -o, --opt map              Set driver specific options (default map[])
      --scope string         Control the network's scope
      --subnet strings       Subnet in CIDR format that represents a network segment
```

معمولا تمام این آپشن‌ها مورد استفاده نیست و بیشتر در سناریوها این موارد را بررسی می‌کنیم. در اینجا به بیان چند مثال اکتفا می‌کنیم.

در این مثال یک network با درایور bridge ایجاد شده است که نام آن my-bridge-network می‌باشد.

```
docker network create -d bridge my-bridge-network
```

در این مثال یک network با subnet مشخص ip ایجاد شده است که نام آن multi-host-network می‌باشد.

```
docker network create –subnet 172.20.0.0/16 –ip-range 172.20.240.0/20 multi-host-network
```

در این مثال یک network با subnet مشخص ایجاد شده است که درایور آن را نیز bridge قرار داده است که برای درایور bridgeآپشن mtu را برابر ۹۲۱۶ قرار داده است.

```
docker network create –driver=bridge –subnet=172.28.0.0/16 –ip-range=172.28.5.0/24 –gateway=172.28.5.254 –opt
com.docker.network.driver.mtu=9216 br0
```

همان‌طور که در مثال‌های بالا مشاهده کردید تقریبا تمام موارد مربوط به پیکربندی network را می‌توانید با آپشن‌های موجود در هنگام ایجاد network بر روی آن اعمال کنید. توضیح تمام این موارد را می‌توانید در این لینک مشاهده کنید و اینکه بنا بر نیاز برخی از این آپشن‌ها در سناریوهای پیاده‌سازی بررسی خواهد شد.

❖ دستور docker network disconnect

این دستور برعکس دستور connect کانتینر شما را از network شما قطع خواهد کرد و به network پیش‌فرض متصل می‌کند. تنها یک آپشن دارد که مربوط به force می‌باشد. به الگو استفاده‌ی آن توجه کنید.

```
ahmad@DockerMe:~$ docker network disconnect --help

Usage:  docker network disconnect [OPTIONS] NETWORK CONTAINER

Disconnect a container from a network

Options:
  -f, --force   Force the container to disconnect from a network
ahmad@DockerMe:~$
```

به مثال زیر توجه کنید:

docker network disconnect multi-host-network container1

❖ دستور docker network inspect

با استفاده از این دستور می‌توانید جزئیات کامل و تمامی اطلاعات موجود در مورد network که ایجاد شده است را داشته باشید.

```
ahmad@DockerMe:~$ docker network inspect --help

Usage:  docker network inspect [OPTIONS] NETWORK [NETWORK...]

Display detailed information on one or more networks

Options:
  -f, --format string   Format the output using the given Go template
  -v, --verbose         Verbose output for diagnostics
ahmad@DockerMe:~$
```

به صورت پیش‌فرض خروجی که به شما نمایش می‌دهد به صورت JSON می‌باشد. با استفاده از آپشن f- می‌توانید فرمت مد نظر خودتون در خروجی این دستور را داشته باشید.

هر زمانی که یک network ایجاد کردید با استفاده از این دستور می‌توانید بررسی کنید که مواردی که اعمال کردید به درستی ایجاد شده است و یا اینکه چه کانتینرهایی در حال حاضر با این network در ارتباط هستند.

❖ دستور docker network ls

با استفاده از این دستور شما می‌توانید لیست network هایی که ایجاد کردید را مشاهده کنید. این دستور دارای چند تا آپشن می‌باشد که با توجه به توضیحات دستورات قبلی تمام موارد آن مشخص می‌باشد.

```
ahmad@DockerMe:~$ docker network ls --help

Usage:  docker network ls [OPTIONS]

List networks

Aliases:
  ls, list

Options:
  -f, --filter filter   Provide filter values (e.g. 'driver=bridge')
      --format string   Pretty-print networks using a Go template
      --no-trunc        Do not truncate the output
  -q, --quiet           Only display network IDs
ahmad@DockerMe:~$
```

به مثال‌های زیر توجه کنید:

در مثال زیر لیست network هایی را نمایش می‌دهد که درایور آنها bridge باشد.

docker network ls –filter driver=bridge

مواردی که بر اساس آن می‌توان در این دستور filter نوشت تا خروجی دلخواه را داشت به قرار زیر است.

The currently supported filters are:

- driver
- id (network's id)
- label (`label=<key>` or `label=<key>=<value>`)
- name (network's name)
- scope (`swarm|global|local`)
- type (`custom|builtin`)

یکی دیگر از آپشن‌هایی که مورد استفاده دارد format است که بر اساس آن می‌توان خروجی را متناسب با دلخواه و مورد استفاده خود تغییر داد و آن را ایجاد کرد. مواردی که در فرمت می‌توان آنها را داشت به قرار زیر است:

Valid placeholders for the Go template are listed below:

Placeholder	Description
`.ID`	Network ID
`.Name`	Network name
`.Driver`	Network driver
`.Scope`	Network scope (local, global)
`.IPv6`	Whether IPv6 is enabled on the network or not.
`.Internal`	Whether the network is internal or not.
`.Labels`	All labels assigned to the network.
`.Label`	Value of a specific label for this network. For example `{{.Label "project.version"}}`
`.CreatedAt`	Time when the network was created

برای مثال زمانی که بخواهید در خروجی فقط id و driver مربوط به هر network را داشته باشید از دستور زیر استفاده می‌کنید.

```
docker network ls --format "{{.ID}}: {{.Driver}}"
```

❖ دستور docker network prune

با استفاده از این دستور شما می‌توانید تمام network هایی که بی استفاده است را پاک کنید.

```
ahmad@DockerMe:~$ docker network prune --help

Usage:  docker network prune [OPTIONS]

Remove all unused networks

Options:
      --filter filter   Provide filter values (e.g. 'until=<timestamp>')
  -f, --force           Do not prompt for confirmation
ahmad@DockerMe:~$
```

آپشن‌های این دستور مشخص می‌باشد که با یکی از آنها فیلتری برای مشخص کردن اینکه کدام یک از آپشن‌ها پاک شود مشخص خواهید کرد و با یکی دیگر برای پاک کردن اجبار ایجاد خواهید کرد.

بر اساس موارد زیر می‌توانید برای این دستور فیلتر مشخص بنویسید.

The currently supported filters are:

- until (`<timestamp>`) - only remove networks created before given timestamp
- label (`label=<key>` , `label=<key>=<value>` , `label!=<key>` , or `label!=<key>=<value>`) - only remove networks with (or without, in case `label!=...` is used) the specified labels.

❖ دستور docker network rm

با استفاده از این دستور نیز می‌توانید network ایجاد شده را پاک نمایید.

```
docker network rm my-network
```

۲۰.دستورات مدیریتی داکر–docker secret

این محتوا تنها در زمانی که از داکر به صورت swarm mode استفاده شود کاربرد دارد. آن هم برای زمانی که بخواهیم دیتای حساسی رو در شبکه منتقل کنیم که از جمله‌ی آنها می‌توان به موارد زیر اشاره کرد:

- Usernames and passwords
- TLS certificates and keys
- SSH keys
- Generic strings or binary content (up to 500 kb in size)
- Other important data such as the name of a database or internal server

به صورت کلی به جای اینکه دیتای مهم و حساس رو در داخل ایمیج یا سورس کد به صورت غیر رمزنگاری شده نگهداری کنند از secret استفاده می‌کنند تا امنیت لازم روی دیتا رعایت شود.

گاهی هم برای احراض هویت از docker secret استفاده می‌شود که بتوان با استفاده از آن محیط‌های Deployment را مدیریت و کنترل کرد.

البته می‌توان با استفاده از docker secret موارد دیگه همانند کانفیگ‌ها را نیز مدیریت کرد.

چطوری کار می‌کند؟

به این صورت عمل می‌کند که در تمامی نودهای Swarm سرویس داکر Secret خود را در بستر کاملا TLS به سمت نود Swarm Manager ارسال می‌کند و این Secret ها به صورت غیر رمزنگاری شده در تمامی Manager های Swarm قرار دارند و همواره این اطلاعات بین آنها sync می‌باشد. بعد از اجرای کانتینر درون آن Secret به صورت decrypted شده در مسیر زیر Mount می‌شود.

Linux Container: /run/secrets/<secret_name>

Windows Container: C:ProgramDataDockersecrets

نودها تنها دسترسی به encrypted secrets ها دارند در صورتی که Manager نود باشد یا اون سرویس را اجرا کرده باشد به خود Secret دسترسی دارد. زمانی که یک کانتینر در حال کار متوقف می‌شود مجدد Secret رمزنگاری شده و مواردی که در رم سرور Mount شده بود پاک می‌شود.

به زبان ساده‌تر تنها در زمانی secret ها به صورت غیر رمزنگاری در دسترس هستند که توسط خود swarm سرویس آنها راه‌اندازی شده باشد. در صورتی که کانتینر متوقف شود تمام سکرت‌ها مجدد رمزگذاری خواهند شد.

❖ دستور docker secret

با استفاده از دستور docker secret می‌توان secret ها را مدیریت کرد.

```
[  5:01 ]  [ root@DockerMe:~ ]
 $ docker secret --help

Usage:  docker secret COMMAND

Manage Docker secrets

Commands:
  create      Create a secret from a file or STDIN as content
  inspect     Display detailed information on one or more secrets
  ls          List secrets
  rm          Remove one or more secrets

Run 'docker secret COMMAND --help' for more information on a command.
[  5:01 ]  [ root@DockerMe:~ ]
```

همانطور که مشاهده می‌کنید با استفاده از این دستور و آپشن‌های آن می‌توانید secret ایجاد و یا اینکه آن‌ها را پاک کنید.

در ادامه با توضیح یک مثال نحوه‌ی استفاده از secret و دستور docker secret توضیح داده می‌شود.

با استفاده از دستور زیر می‌توانید یک secret با نام my_secret_data ایجاد کنید.

```
printf "This is a secret" | docker secret create my_secret_data -
```

با استفاده از دستور زیر یک سرویس با نام redis ایجاد می‌کنیم که از این secret استفاده می‌کند.

```
docker service  create --name redis --secret my_secret_data redis:alpine
```

بعد از راه‌اندازی کانتینر redis حتما اطمینان پیدا کنید که بدون مشکل راه‌اندازی شده باشد.

دقت کنید secret ایجاد شده که در کانتینر redis در مسیر /run/secrets قرار دارد. با استفاده از دستورات زیر می‌توانید نام secret و سپس محتوای آن را مشاهده کنید.

```
docker ps --filter name=redis -q
5cb1c2348a59
docker container exec $(docker ps --filter name=redis -q) ls -l /run/secrets
-r--r--r--    1 root     root           17 Dec 13 22:48 my_secret_data
docker container exec $(docker ps --filter name=redis -q) cat /run/secrets/my_secret_data
This is a secret
```

در صورتی که الان از این کانتینر یک ایمیج تهیه کنید و آن را commit کنید و از ایمیج جدید هم یک کانتینر ایجاد کنید داخل کانتینر جدید دیگر secret وجود ندارد و secret داخل ایمیج نیست. این نکته بسیار اهمیت دارد.

تا زمانی که سرویس redis برقرار باشد امکان اینکه این secret را پاک کنید میسر نمی‌باشد. می‌توانید با دستور زیر secret مربوط به هر سرویس در حال کار را پاک کنید.

```
1docker service update --secret-rm my_secret_data redis
```

حال اگر بعد از این دستور داخل دایرکتوری مد نظر را بگردید دیگه secret قبلی رو مشاهده نخواهید کرد.

```
docker container exec -it $(docker ps --filter name=redis -q) cat /run/secrets/my_secret_data
cat: can't open '/run/secrets/my_secret_data': No such file or directory
```

با استفاده از دستور زیر نیز می‌توانید secret ایجاد شده را پاک کنید.

```
1docker secret rm my_secret_data
```

طی یک مثال ساده نحوه‌ی استفاده از secret رو بررسی کردیم. بعدا در زمان توضیح swarm و استفاده از آن بیشتر با موضوع secretآشنا خواهیم شد.

۲۱.دستور مدیریتی docker node

با استفاده از این دستور می‌توانید نودهای داخل کلاستر swarm را مدیریت کنید. این دستور تنها بر روی نودهای swarm manager در دسترس می‌باشد. با استفاده از این دستور می‌توان یک نود را به نود manager یا اینکه یک نود manager را به نود worker تبدیل کرد. در ادامه خیلی مختصر آپشن‌های این دستور را باهم بررسی خواهیم کرد.

```
[root@n4.dockerme.ir ~]# docker node --help

Usage:  docker node COMMAND

Manage Swarm nodes

Commands:
  demote      Demote one or more nodes from manager in the swarm
  inspect     Display detailed information on one or more nodes
  ls          List nodes in the swarm
  promote     Promote one or more nodes to manager in the swarm
  ps          List tasks running on one or more nodes, defaults to current node
  rm          Remove one or more nodes from the swarm
  update      Update a node

Run 'docker node COMMAND --help' for more information on a command.
[root@n4.dockerme.ir ~]#
```

❖ دستور docker node ls

با استفاده از این دستور لیست نودها نمایش داده می‌شود. همانطور که ملاحظه می‌کنید این دستور دارای آپشن‌هایی نیز می‌باشد که قبلا در دستورات قبلی تمام این موارد توضیح داده شده است.

```
[root@n4.dockerme.ir ~]# docker node ls --help

Usage:  docker node ls [OPTIONS]

List nodes in the swarm

Aliases:
  ls, list

Options:
  -f, --filter filter   Filter output based on conditions provided
      --format string   Pretty-print nodes using a Go template
  -q, --quiet           Only display IDs
[root@n4.dockerme.ir ~]#
```

❖ دستور docker node demote

با استفاده از این دستور یک یا چند نودی که به صورت manager راه‌اندازی شده است را به worker مبدل می‌کند.

```
[root@n4.dockerme.ir ~]# docker node demote --help

Usage:  docker node demote NODE [NODE...]

Demote one or more nodes from manager in the swarm
[root@n4.dockerme.ir ~]#
```

❖ دستور docker node promote

با استفاده از این دستور یک یا چند نودی که به صورت worker می‌باشند را می‌توان به نود manager تبدیل کرد.

```
[root@n4.dockerme.ir ~]# docker node promote --help

Usage:  docker node promote NODE [NODE...]

Promote one or more nodes to manager in the swarm
[root@n4.dockerme.ir ~]#
```

❖ دستور docker node inspect

با استفاده از این دستور اطلاعات و جزئیات بسیار زیادی در مورد هر نود می‌توان بدست آورد. این اطلاعات شامل نسخه‌ی داکر که بر روی سرور نصب شده و پلاگین‌ها و ... می‌باشد.

```
[root@n4.dockerme.ir ~]# docker node inspect --help

Usage:  docker node inspect [OPTIONS] self|NODE [NODE...]

Display detailed information on one or more nodes

Options:
  -f, --format string   Format the output using the given Go template
      --pretty          Print the information in a human friendly format
[root@n4.dockerme.ir ~]#
```

❖ دستور docker node ps

با استفاده از این دستور می‌توان وظایف جاری بر روی هر نود را مشاهده کرد. به صورت پیش‌فرض وظایف نودی که بر روی آن دستور ران می‌شود را نمایش می‌دهد. اما می‌توان در انتهای دستور نود مد نظر را مشخص کرد تا وظایف جاری آن نود را به شما نمایش دهد. با استفاده از این دستور می‌توان همزمان وظایف جاری روی چند تا نود را نیز مشاهده کرد.

```
[root@n4.dockerme.ir ~]# docker node ps --help

Usage:  docker node ps [OPTIONS] [NODE...]

List tasks running on one or more nodes, defaults to current node

Options:
  -f, --filter filter   Filter output based on conditions provided
      --format string   Pretty-print tasks using a Go template
      --no-resolve      Do not map IDs to Names
      --no-trunc        Do not truncate output
  -q, --quiet           Only display task IDs
[root@n4.dockerme.ir ~]#
```

❖ دستور docker node rm

با استفاده از این دستور می‌توان یک یا چند تا نود رو کلا از swarm بیرون انداخت به گونه‌ای که دیگر در کلاستر حضور نداشته باشند. اگر نودی به صورت manager در حال کار باشد ابتدا باید آن را به worker و سپس آن را پاک نمود. نودهای worker در حال کار را نیز ابتدا باید down کرد و سپس آنها را پاک نمود ولی با استفاده از آپشن -f می‌توان نودهای worker در حال کار را نیز پاک کرد.

```
[root@n4.dockerme.ir ~]# docker node rm --help

Usage:  docker node rm [OPTIONS] NODE [NODE...]

Remove one or more nodes from the swarm

Aliases:
  rm, remove

Options:
  -f, --force   Force remove a node from the swarm
[root@n4.dockerme.ir ~]#
```

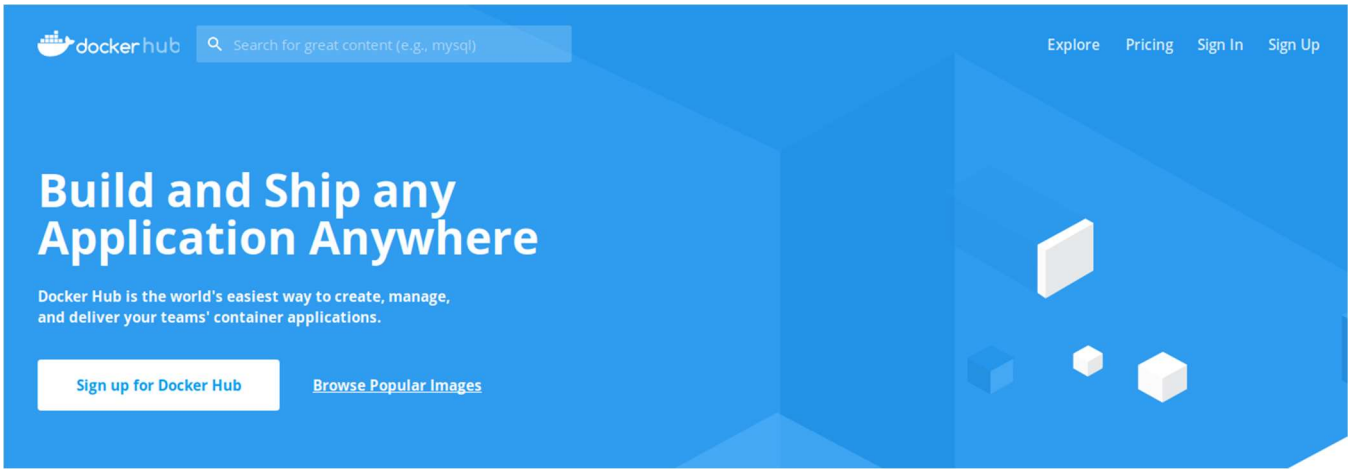
❖ دستور docker node update

با استفاده از این دستور می‌توان آپشن‌های مربوط به هر نود را به روز کرد. به عنوان مثال می‌توان role نودها را تغییر داده و از worker به manager و از manager به worker تبدیل کرد.

```
[root@n4.dockerme.ir ~]# docker node update --help

Usage:  docker node update [OPTIONS] NODE

Update a node

Options:
      --availability string   Availability of the node ("active"|"pause"|"drain")
      --label-add list        Add or update a node label (key=value)
      --label-rm list         Remove a node label if exists
      --role string           Role of the node ("worker"|"manager")
[root@n4.dockerme.ir ~]#
```

۲۲. ایجاد ریجیستری خصوصی

داکر ریجیستری از اجزای اصلی سرویس داکر می‌باشد. ریجیستری یک مخزن است که ایمیج‌های داکر در آن قرار داده می‌شود. معمولا به دو صورت عمومی و خصوصی ارائه می‌شود. یکی از بهترین و معروف‌ترین رجیستری‌های داکر، مربوط به همین کمپانی به نام داکر هاب است که به صورت پیش‌فرض برای سرویس داکر این ریجیستری تنظیم شده است. نکته‌ی مهم اینکه شما می‌توانید در این رجیستری ثبت‌نام کنید و برای خودتون ریپوزیتوری مخصوص خود را داشته باشید. قبلا تنها امکان ایجاد رجیستری عمومی در این سایت بود اما در حال حاضر می‌توانید ریپوزیتوری به صورت خصوصی داشته باشید. یعنی قبلا تمام ایمیج‌هایی که در این رجیستری قرار می‌دادید در اختیار عموم قرار می‌گرفت اما در حال حاضر این امکان وجود دارد که ریپوزیتوری خصوصی نیز داشته باشید. حتما بهتون پیشنهاد می‌کنم که در این سایت ثبت‌نام کنید و برای خودتون اینجا رجیستری داشته باشید.



اما به صورت کلی هر شرکت یا هر پروژه‌ای برای خودش می‌تواند یک رجیستری خصوصی و محلی داشته باشد. در این مستند سعی بر آن شده تا یک رجیستری کامل خصوصی ایجاد کنیم تا در پروژه‌های خودمون از آن استفاده کنیم. برای شروع می‌توان به همین سادگی یک رجیستری با کمترین کانفیگ و ساده‌ترین حالت ایجاد کرد.

```
docker run -d -p 5000:5000 --restart=always --name registry registry:2
```

حالا بر روی پورت ۵۰۰۰ روی سرور خودمون یک ریجیستری محلی داریم. با استفاده از دستورات زیر می‌توانیم یک ایمیج را داخل این رجیستری قرار بدیم. ابتدا ایمیج ubuntu رو دریافت کرده و سپس آن را tag می‌کنیم و بعد از آن در رجیستری خودمون قرار می‌دهیم.

```
docker pull ubuntu:16.04
docker tag ubuntu:16.04 localhost:5000/my-ubuntu
docker push localhost:5000/my-ubuntu
```

اما این ریجیستری برای یک پروژه‌ی محلی مناسب است و کفایت می‌کند اما نیاز است که ما ایمیج‌های خود را در جاهای مختلف و در محیط‌های مختلف داشته باشیم. از این رو نیاز است تا رجیستری خود را به صورت public اما با دسترسی محدود داشته باشیم. به ادامه این مطلب توجه کنید. اول یک‌سری آپشن مهم و کاربردی به دستور docker run خود اضافه می‌کنیم. این کانفیگ‌ها بسیار مهم و کارگشا می‌باشد. انتخاب :volume خیلی مهم است که همواره دیتای داخل کانتینر به بیرون منتقل شود و بر روی استوریج مطمئنی نگهداری شود -v /mnt/registry:/var/lib/registry . با استفاده از این آپشن شما می‌توانید دیتای داخل رجیستری خود را بر روی یک استوریج مجزا از کانتینر ذخیره نمایید. استفاده از آدرس مخصوص: با استفاده از این آپشن می‌توانیم پورت رجیستری را روی IP مشخصی که مد نظر خودمان است bind کنیم.

```
-e REGISTRY_HTTP_ADDR=0.0.0.0:5001
```

راه‌اندازی رجیستری به صورت TLS: این نکته خیلی مهم است که رجیستری ما به صورت TLS راه‌اندازی شود. برای این کار نیاز است تا از آپشن‌های زیر استفاده شود.

```
-v "$(pwd)"/certs:/certs
-e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt
-e REGISTRY_HTTP_TLS_KEY=/certs/domain.key
```

احراض هویت : یکی از موارد مهم در ایجاد رجیستری خصوصی، ایجاد سطح دسترسی و کاربر می‌باشد. نیاز است تا حتما این امکان را بر روی رجیستری خود نیز داشته باشیم. برای این موضوع ابتدا فایل htpasswd را ایجاد کرد سپس آن را به ایمیج رجیستری وارد کرده تا از آن استفاده کند. در ادامه نحوه‌ی استفاده از آن نمایش داده می‌شود.

```
mkdir auth
docker run --entrypoint htpasswd registry:2 -Bbn testuser testpassword > auth/htpasswd
```

ابتدا فایل مورد نظر را ایجاد کرده و سپس user و pass مد نظر خود را که در این مثال testuser و testpassword می‌باشد را در فایل مد نظر قرار می‌دهیم. بعد از آن ایمیج مد نظر خود را با آپشن‌های زیر راه‌اندازی کرده و در آن نوع authentication را htpasswd قرار داده و مسیر آن را مشخص و از آن استفاده می‌کنیم.

```
docker run -d -p 5000:5000
--restart=always --name registry
-v "$(pwd)"/auth:/auth
-e "REGISTRY_AUTH=htpasswd"
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm"
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd
-v "$(pwd)"/certs:/certs
-e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt
-e REGISTRY_HTTP_TLS_KEY=/certs/domain.key
registry:2
```

اگر خواسته باشیم که رجیستری خود را با docker compose راه‌اندازی کنیم می‌توانیم از این فایل استفاده کنیم.

```
registry:
  restart: always
  image: registry:2
  ports:
    - 5000:5000
  environment:
    REGISTRY_HTTP_TLS_CERTIFICATE: /certs/domain.crt
    REGISTRY_HTTP_TLS_KEY: /certs/domain.key
    REGISTRY_AUTH: htpasswd
    REGISTRY_AUTH_HTPASSWD_PATH: /auth/htpasswd
    REGISTRY_AUTH_HTPASSWD_REALM: Registry Realm
  volumes:
    - /path/data:/var/lib/registry
```

```
- /path/certs:/certs
- /path/auth:/auth
```

در برخی از پروژه‌ها هم TLS و Authentication رو بر عهده‌ی سرویس‌های دیگه همانند nginx قرار می‌دهند که این سرویس اون موارد رو هندل می‌کند.البته رجیستری‌های خیلی خوبی وجود دارند که به شما همزمان که CLI می‌دهند می‌توانید از UI آنها نیز استفاده کنید. از این رو پیشنهاد می‌کنم که این موارد را نیز بررسی نمایید. به عنوان مثال Jfrog یکی از بهترین‌ها می‌باشد که می‌توانید از آن استفاده کنید.

۲۳. راکت چت

راکت چت جایگزین خوبی برای اسلک

این چند روز اخیر این شرکت دیگه ارتباط با ایران و کاربران ایرانی رو محدود کرد و مشکلاتی را برای شرکت‌های ایرانی که از محصولات آن استفاده می‌کردند ایجاد کرده است. Slack یه اپلیکیشن چت گروهی می‌باشد که خیلی امکانات خوبی در اختیار شرکت‌ها قرار می‌دهد. این اپلیکیشن بر روی زیرساخت ابری پیاده‌سازی شده که این امکان به کیفیت این سرویس نیز کمک عمده‌ای کرده است.

خوب دیگه این امکان فراهم نیست تا شرکت‌ها و افراد ایرانی از اسلک استفاده کنند از این رو باید یک گزینه‌ی جایگزین برای این نرم‌افزار فراهم شود تا امکانات خوبی را اختیار آنها قرار دهد. پیشنهاد من استفاده از اپلیکیشن متن‌باز Rocket.Chat می‌باشد. این نرم‌افزار به صورت کامل رایگان، بدون محدودیت و متن باز می‌باشد. یک راهکار خوب برای جایگزین کردن با اسلک می‌باشد. البته این سرویس نسخه‌ی ابری هم دارد که به صورت پولی می‌باشد و می‌بایست برای آن پرداخت انجام دهید. از امکانات دیگر این نرم‌افزار می‌توان به سازگاری آن با تمام پلتفرم‌ها می‌باشد که انواع مختلفی از clientها را پشتیبانی می‌کند و در کل خیلی کار راه‌انداز هست و توصیه می‌کنم در کنار سرویس‌های داخلی شرکت خودتون از این چت سرور هم استفاده کنید.

از مزایای استفاده از راکت چت گفتم حالا می‌خوام در مورد نصبش صحبت کنم. نصب این ابزار روش‌های مختلفی دارد که خوب داکیومنت خوبی هم برای نصب اون وجود دارد که در این لینک (داکیومنت‌های سایت خود راکت چت) می‌توانید آنها را مشاهده کنید. اما روشی که من می‌خوام به شما پیشنهاد بدم استفاده از داکر کامپوز می‌باشد.

فایل داکر کامپوز:

فایل داکر کامپوز را می‌توانید از اینجا دریافت کنید اما برای راحتی بیشتر شما عزیزان در این پست هم تمام موارد قرار داده شده است. دقت کنید که حتما موارد که داخل [متغیرهای پیاده‌سازی] می‌باشد را مخصوص سرویس خود تکمیل نمایید. این متغیرها شامل مواردی همچون پورت‌های سرویس بر روی هاست و دایرکتوری مربوط به ذخیره اطلاعات داخل کانتینرها می‌باشد.

```
1  version: '2'
2  services:
3    mongo:
4      restart: unless-stopped
5      image: mongo
6      volumes:
7        - [data directory path]:/data/db
8        - [backup directory path]:/backups
9      command: --smallfiles
10   rocketchat-oeru:
11     restart: unless-stopped
12     image: rocketchat/rocket.chat
13     ports:
14       - "127.0.0.1:[port number]:3000" # should be a free port above 1024
15     depends_on:
16       - mongo
17     environment:
18       - MONGO_URL=mongodb://mongo/rocket
19       - ROOT_URL=[domain name (including schema, e.g. http://)]
20     volumes:
21       - [upload directory path]:/var/www/rocket.chat/uploads
```

- Edit the ROOT_URL value to be your FQDN.
- Edit the ROCKETCHAT_URL to be your *public* IP address. Keep the port (3000) the same.

- Edit ROCKETCHAT_USER, ROCKETCHAT_PASSWORD, and BOT_NAME.

پس از دریافت و تغییر پارامترهای داکر کامپوز و اصلاح آن براس استفاده خودتون مراحل زیر را طی کنید.

در سایت خودش کامل توضیح داده که کدوم پارامترها رو تغییر بدیم. Composefile در سایت هم با این یخورده متفاوت است.(https://rocket.chat/docs/installation/docker-containers/)

نحوه‌ی اجرای سرویس:

با استفاده از دستورات زیر ایمیج‌های مورد نیاز را دانلود می‌کنیم البته که می‌توان این دستورات رو هم نزد و هر زمانی که فایل کامپوز را اجرا کنید قبل از اجرا ایمیج‌ها را ابتدا دریافت می‌کند اما می‌توان جداگانه ایمیج‌ها را دریافت کرد چون معمولا تو ایران دریافت ایمیج خیلی زمان‌بر است از این رو خوبه که اول ایمیج‌ها را دریافت کنید.

```
docker pull mongo
docker pull rocket.chat
```

بعد از اتمام دریافت ایمیج‌ها می‌توانید با استفاده از دستور زیر کامپوزفایل مورد نظر را اجرا کرده تا سرویس‌های مد نظر راه‌اندازی شود. (فقط باید به دایرکتوریی که این compose file قرار دارد برویم و حواسمان باشد که اسم compose file بصورت استاندارد باشد.)

```
docker-compose up -d
```

با استفاده از دستور زیر می‌توانید لاگ کانتینرهای داخل کامپوز فایل را به صورت کامل مشاهده کنید.

```
docker-compose logs -f
```

و با این دستور ۱۰۰ خط آخر لاگ تمام کانتینرها را مشاهده کنید.

```
docker-compose logs -f --tail 100
```

استفاده از nginx به عنوان reverse proxy:

خوب الان سرویس چت آماده هست اما روی پورت داخلی سرور در حال سرویس‌دهی می‌باشد. البته این پورت به فراخور تنظیمات شما می‌تواند متفاوت باشد اما به صورت پیش‌فرض پورت داخل کانتینر ۳۰۰۰ می‌باشد. برای این که بتوانیم از این سرویس استفاده کنیم پیشنهاد می‌کنم که از nginx به عنوان reverse proxy استفاده کنیم تا تمام درخواست‌ها به سمت آن آمده و nginx آنها را به بـَکِـند خودش هدایت کند.

برای این کار از کانفیگ زیر استفاده می‌کنیم. البته این کانفیگ را می‌توانید از اینجا (گیت لب احمد رفیعی) نیز دریافت کنید. ابتدا مقادیر مورد نظر خود را جایگزین می‌کنید سپس برای دامنه‌ی مد نظر خود گواهی‌نامه‌ی امن دریافت کنید. بعد از آن سرویس nginx را تست کرده و بعد از آن را reload نمایید. تمام این مراحل را در ادامه می‌توانید مشاهده کنید.

{گیت لب احمد رفیعی:

```
server
{
        listen 443 ssl http2;
        server_name [domain name];


        # SSL
        ssl_certificate /etc/letsencrypt/live/[domain name]/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/[domain name]/privkey.pem;




        # SSL Settings
        ##
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_prefer_server_ciphers on;
        ssl_dhparam /etc/ssl/certs/dhparam.pem;
        ssl_ciphers "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA256
      EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES
      !MD5 !EXP !PSK !SRP !DSS !MEDIUM";
        ssl_session_timeout 1d;
        ssl_session_cache shared:SSL:50m;
        ssl_stapling on;
        ssl_stapling_verify on;
        add_header Strict-Transport-Security max-age=15768000;
```

```
    location ~ /.well-known {
                allow all;
        }


  location / {
  proxy_pass                          http://127.0.0.1:[your rocketchat port];
  proxy_set_header  Host              $http_host;   # required for docker client's sake
  proxy_set_header  X-Real-IP         $remote_addr; # pass on real client's IP
  proxy_set_header  X-Forwarded-For   $proxy_add_x_forwarded_for;
  proxy_set_header  X-Forwarded-Proto $scheme;
  proxy_read_timeout                  900;
  proxy_http_version 1.1;
  proxy_set_header Upgrade $http_upgrade;
  proxy_set_header Connection "upgrade";
  proxy_set_header X-Forwarded-Host $host;
  proxy_set_header X-Nginx-Proxy true;
  proxy_redirect off;
}


}
server {
    listen 80;
    server_name [domain name];
    return 301 https://$host$request_uri;
}
```

{

دریافت ssl certificate از letsencrypt:

دریافت Certificate از letsencrypt روش‌های متعددی دارد که خیلی کار را باهاش را راحت کرده است. در یک سری پست و ویدیو مفصل که بعدا آماده می‌کنم تمام روش‌های دریافت Certificate را بررسی می‌کنیم اما اینجا با استفاده از روش standalone گواهی مد نظر خودمون رو دریافت خواهیم کرد. حتما قبل از دریافت certificate نیاز است تا certbot را نصب کرده باشید که برای نصب می‌توانید از این لینک استفاده کنید.

```
certbot certonly --standalone --preferred-challenges http -d [domain name]
```

بعد از استفاده از این دستور certificate مورد نظر در مسیرهای که نمایش می‌دهد ذخیره می‌شود.

برای کیفیت بیشتر و امن‌تر کردن سرویس از dhparam.pem نیز استفاده شده است که با دستور زیر ایجاد می‌شود.

```
openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

بعد از ایجاد تمام این موارد می‌توانید با استفاده از دستور زیر کانفیگ nginx را بررسی و تست نمایید.

```
nginx -t
```

اگر نتیجه اوکی بود با استفاده از دستور زیر می‌توانید سرویس nginx خود را مجدد راه‌اندازی و یا reload نمایید.

```
systemctl reload nginx
systemctl status nginx
```

بعد از راه‌اندازی کامل nginx و سرویس‌های دیگه می‌تونید به آدرس domain name خود مراجعه کرده و ادامه‌ی کانفیگ‌های مربوط به rocket chat رو انجام دهید.

می‌شد به جای استفاده از nginx از Traefik استفاده کرد یا اینکه خود nginx رو هم به صورت کانتینر راه‌اندازی کرد اما معمولا روی سرورها یک nginx برای مصارف دیگه هست که از همین استفاده کنیم. در کل Traefik گزینه‌ی مناسبی برای این گونه استفاده ها می‌باشد. داکیومنت و موارد اون رو هم می‌تونید اینجا بررسی کنید.