آموزش داکر

نویسنده: بهنام حسن بیگی

mlops fanapsoft/AI team

سرفصل ها:

۱ – مدرن شدن اپلیکیشن ها

۲ – تاریخچه ی داکر

۳ – داکر چیست؟!

۴ – نسخه های داکر

۵ – کلیات داکر

۶ – مشکلات نصب اپلیکیشن در زمان پیاده سازی

۷ -کانتاینر چیست؟

۸ – مزایایی استفاده از داکر

۹ – پایه و اساس موتور داکر docker engine

۱۰ – نصب داکر

۱۱ – مراحل ساخت یک کانتاینر در داکر

۱۲ – دستورات یا کامند های داکر

۱۳ – داکرimage

۱۴ – داکر volume

۱۵ – شبکه داکر

۱۶ – ساختن image به کمک Dockerfile

۱۷ – docker-compose

۱۸ – docker registry

فصل اول

مدرن شدن اپلیکیشن ها

در روش قدیمی پیاده سازی اپلیکیشن ها تمام کد ها و فانکشن های مربوط به ان اپلیکیشن در یک محیط اجرا می شد.فرض کنید یک وب اپلیکیشن فلسک را باید راه اندازی کنید در روش قدیمی دیتابیس و کد اصلی و همچنین وب سرویسی که استفاده میکتید و user interface که طراحی کرده اید همه در یک محیط اجرا می شد در صورت وجود باگ یا هر مشکلی در dependence های موجود اپلیکیشن شما خارج از دسترس می شد و پیدا کردن این که مشکل از کدام قسمت است به سختی صورت میگرفت. به این نوع معماری قدیمی monolithic می گویند.

با پیشرفت تکنولوژی به خصوص container runtime management ها مثل داکر و RKT و LXC معماری اجرا (deployment) اپلیکیشن ها تغییر پیدا کرد.

در این نوع معماری تمام سرویس های مورد نیاز اپیلیکشن از هم جدا و در محیط های متفاوتی اجرا می شوند.در این زمان بود که نوع جدیدی از معماری ساخت اپلیکیشن معرفی شد به اسم microservices

مزایای معماری microservices نسبت به monolithic :

۱ اگر در اپلیکیشن باگ و یا نیاز به ورژن جدید وجود داشته باشد معماری monolithic باید از ابتدا کد را کامپایل و دوباره تست کتید ولی در معماری microservices به دلیل جدا سازی سریع تر انجام خواهد شد.

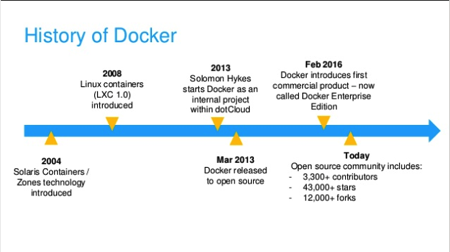
۲ در معماری monolithic یک اشتباه کل اپلیکیشن را از دسترس خارج خواهد ولی در microservices به دلیل طراحی درست میتوان به سرعت ان مشکل را برطرف کرد

۳ در معماری monolithic به سختی میتوان high available را اجرا کرد ولی در microservices میتوان از یک سرویس چندین جایگزین مهیا کرد.



فصل دوم

تاریخچه ی داکر

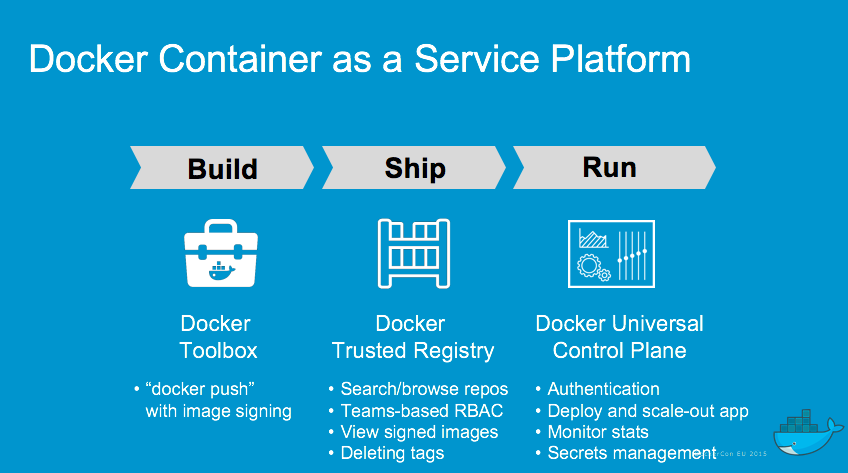


در سال ۲۰۰۴ ایتدا سولاریس کانتاینر توسط شرکت zone technology معرفی شد در سال ۲۰۰۸ لینوکس کانتاینر و سپس در سال ۲۰۱۳ اقای سالمان هایکس پروژ ی داکر رو شروع کرد در همان سال هم به صورت پروژ ی متن باز در اختیار همه قرار گرفت در سال ۲۰۱۶ یک نسخه از داکر به بازار امد به اسم docker enterprise edition که داکر را به صورت سازمانی در اختیار شرکت ها قرار می داد (نسخه ی پولی ولی تحت لیسانس و پشتیبانی شرکت داکر)

به زبان Go نوشته شده و در زمان تهیه این سند جدیدترین نسخه ی ان ۱۹.۰۳.۱۳ است.

فصل سوم

داکر چیست؟



داکر یک پلتفرم متن باز است که شعار معروفش این است:

Build any app

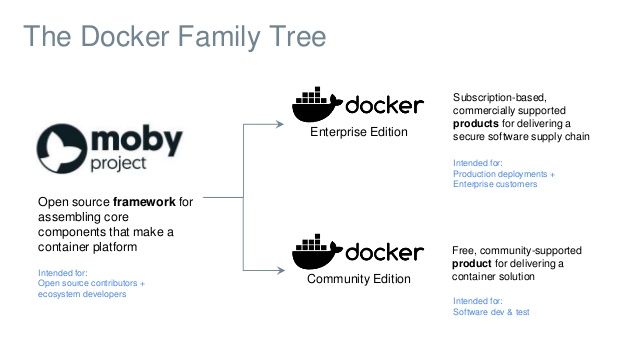
Ship any where

Run in any where

شما می توانید هر اپلیکیشی را در داکر پیاده سازی کنید ذخیره کنید و سپس در هر سیستم عاملی اجرا کنید.

فصل چهارم

خانواده ی داکر



پروژه ی موبی تشکیل شده از دو نسخه ی داکر که یکی docker enterprise edition که همان نسخه ی پولی داکر است که ویژگی های کمی پیشرفته تر نسبت به نسخه متن باز آن دارد و نسخه ی community که متن باز است و برنامه نویس های از سراسر دنیا بر روی این پروژه کار میکنند.برای مطالعه بیشتر راجب تفاوت این دو نسخه از لینک زیر استفاده کنید[[1]](#footnote-1)

فصل پنجم

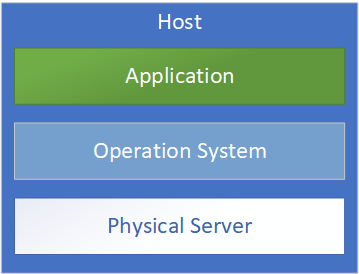
کلیات داکر



داکر تشکیل شده از سه لایه که لایه درونی تر daemon یا همان سرویس داکر است لایه بعدی یک RESTFUL API است که رابط بین command line interface(CLI) و server daemon است .مدیریت شبکه و کانتاینر ها همچنین image ها و data volume به عهده ی CLI است.

فصل ۶

مشکلات نصب اپلیکیشن در زمان پیاده سازی



در معماری قدیمی پیاده سازی اپلیکیشن ها برتامه به صورت مستقیم روی سرور نصب می شد که سختی های را به دنبال داشت:

۱ زمان پیاده سازی بسیار کند بود

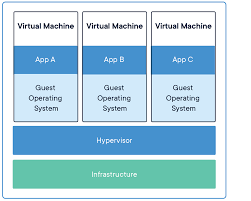
۲ هزیته اضافی و زیادی را بر روی دوش شرکت می گذاشت

۳ منابع سخت افزاری زیادی مصرف میکرد

۴ به سختی می شد اون رو گسترش داد

۵ مهاجرت به نسخه ی جدید تر به سختی انجام میگرفت

نسل بعدی HYPER VISOR ها بودند که در یک سرور فیزیکال میتوانستیم چندین سیستم عامل را اجرا کنیم که مفهوم virtualization یا مجازی سازی به میدان امد.اپلیکیشن های بسیاری برای این منظور ساخته شد که vmware از مشهورترین پلتفرم ها بود که که سیستم عامل ESXI روی سرور فیزیکی نصب می شد.از اپایکیشن های دیگر می توان virtualbox و qemu KVM را نام برد



در این معماری نیز مصرف بیش از اندازه ی منابع سخت افزاری داشتیم همچنین هزینه ها بسیار بالا بود و معنا ومفهوم های جدیدی که در این زمان به وجود امده بود به سختی در این معماری جای می گرفت مانند مفهوم CI/CD یا

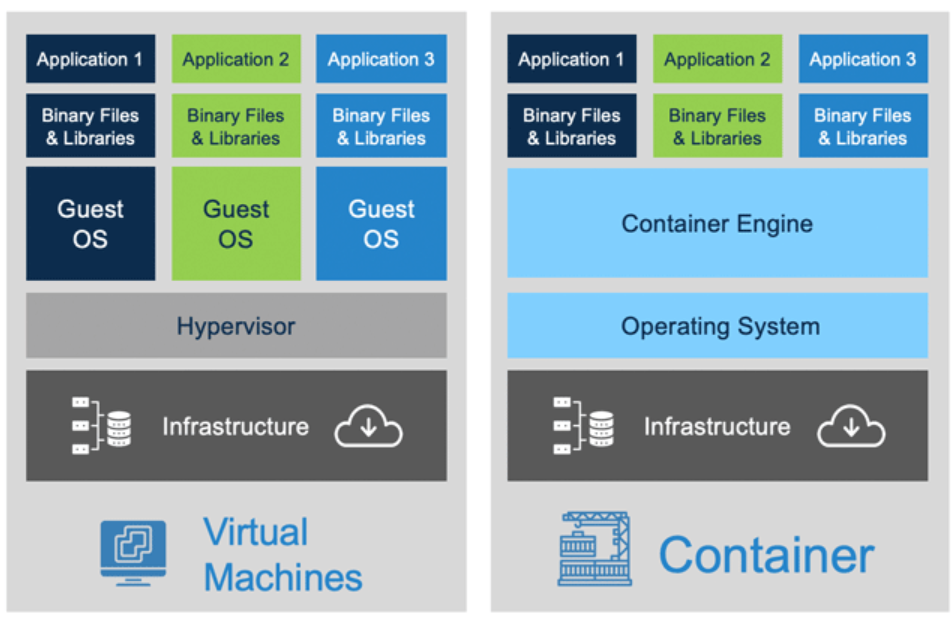
**Continuous integration and continuous development(delivery)**

فصل ۷

کانتاینر چیست؟

اخرین نسل معماری پیاده سازی اپلیکیشن که از معماری microservices پیروی میکنه.کانتاینر(Container) به صورت استاندار اپلیکیشن شما رو بسته بندی یا پکیج(package) میکند با همه وابستگی ها (dependencies).اپلیکیشن ها در کانتاینر های جدا ایزوله هستند و با هم تداخلی ندارند.همه ی اپلیکیشن ها از یک کرنل(kernel) استفاده میکنند.کانتاینر ها در همه ی سرور های linux یا windows server کارایی دارند.برای مطالعه ی بیشتر از لینک زیر استفاده کنید[[2]](#footnote-2)

تفاوت بین container and hyper visor



فصل ۸

مزیت های استفاده از داکر

۱ سرعت

۲ قابلیت جا به جای اپلیکیشن در انواع سیستم عامل

۳ کارایی

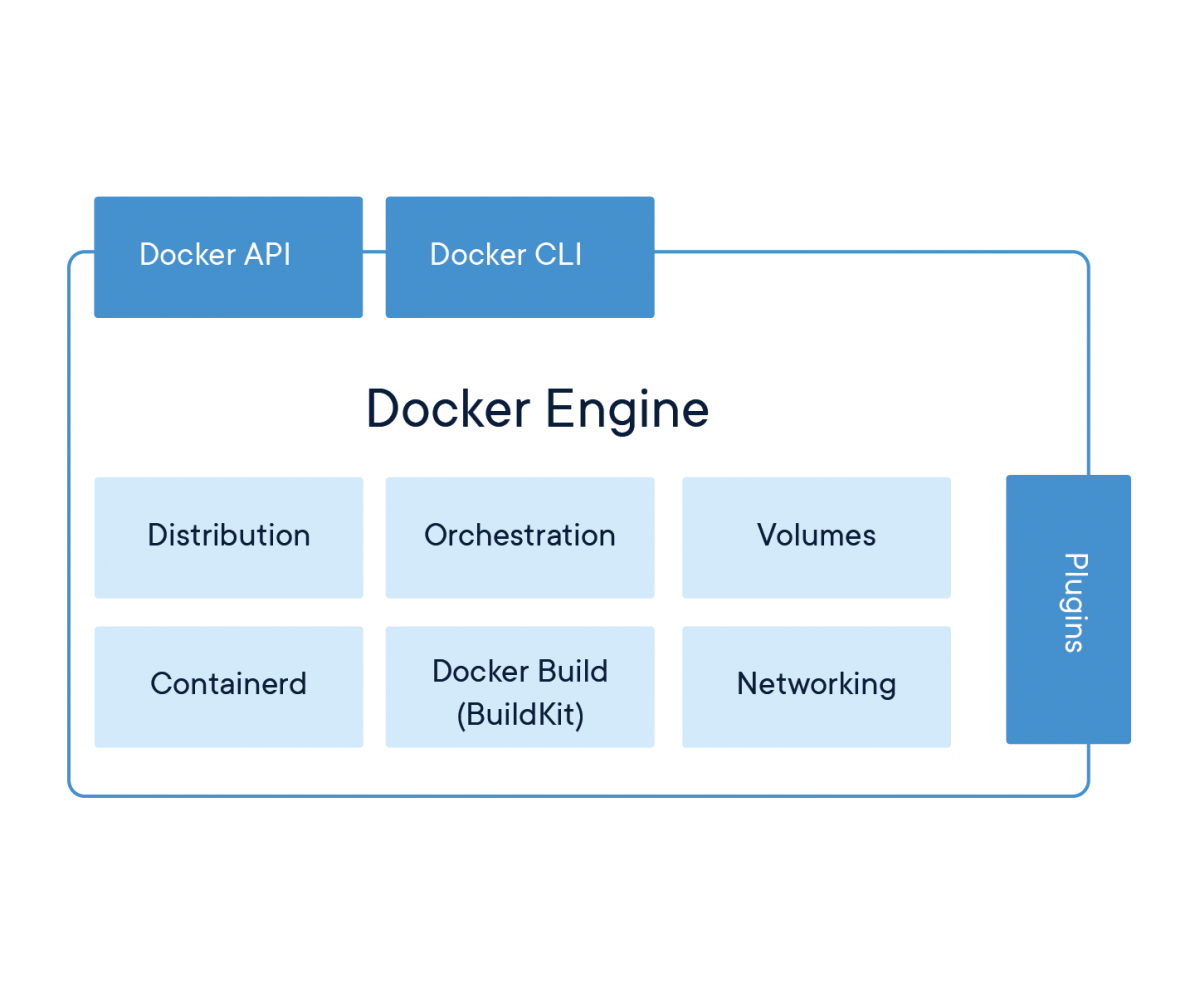
۴ امنیت

۵ متن باز بودن

۶ اسان سازی معنا و مفهوم دواپس

فصل ۹

پایه و اساس موتور داکر docker engine



موتور داکر یک تکنولوژی متن باز برای ساختن کانتاینر و همچنین پیاده سازی اون هاست شبیه به کلاینت سرور عمل میکنه که از اجزای زیر ساخته شده:

یک پروسس سرور برای اجرای اون dockerd

یک API برای ارتباط داشتن با daemon داکر

یک command line interface => docker

برای اطلاعات بیشتر به این لینک مراجعه کنید[[3]](#footnote-3)

فصل ۱۰

نصب داکر

نصب داکر بر روی سرور CENTOS 7

ایتدا باید selinux را غیر فعال کنید.و اگر قبلا داکر را نصب کرده اید ان را پاک کنید.

Yum remove docker

Yum remove docker-common

Yum remove docker-selinux

Yum remove docker-engine

مرحله بعد نصب ابزار های yum است و اضافه کردن repository داکر برای نصب است:

yum install -y yum-utils

yum-config-manager --add-repo\ <https://download.docker.com/linux/centos/docker-ce.repo>

سپس

yum-config-manager --disable docker-ce-edge

yum makecache fast

yum install docker-ce

systemctl start docker

systemctl enable docker

ps -ef | grep docker

با زدن اخرین کامند باید ببینیم که پروسس داکر در حال اجرا است یا نه.

برای نصب در سیستم عامل ویندوز از این لینک استفاده کنید[[4]](#footnote-4).داکر در ویندوز به این صورت عمل می کند که یک سرور لیتوکسی در ویندوز اجرا میکند پس به صورت مستقیم از سیستم عامل ویندوز استفاده نمی شود.

فصل ۱۱

مراحل ساخت یک کانتاینر در داکر

۱ ابتدا باید یک image از docker hub یا ریجستری که برای داکر ایجاد کرده ایم بگیریم به این عمل docker pull میگویند

۲ یک کانتاینر جدید ساخته می شود

۳ یک فایل سیستم جدید ساخته می شود و لایه برای نوشتن و خواندن بر روی ان اضافه میشود

۴ یک network bridage interface در اختیار کانتاینر قرار میگیرد

۵ یک IP به کانتاینر اختصاص داده می شود

۶ یک پروسس مانند شل در لینوکس در ان کانتاینر اجرا می شود /bin/bash or /bin/sh

`۷ خروجی اپلیکیشن نمایش داده می شود

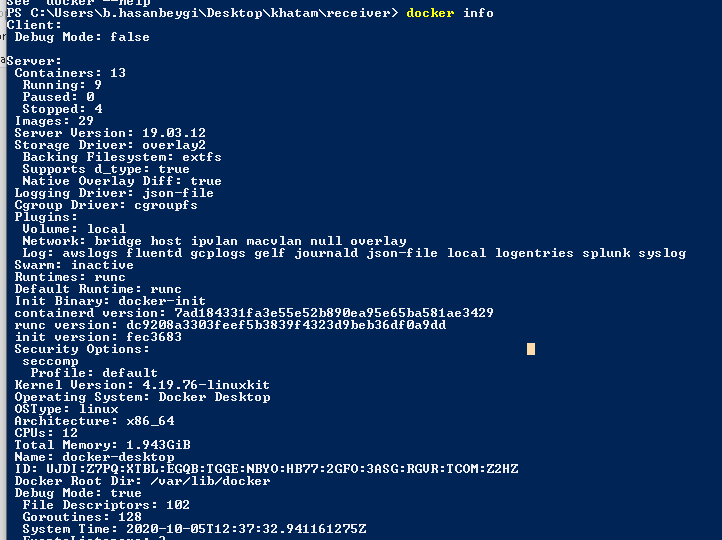
به طور مثال یک ایمیج را از docker hub دریافت میکنیم docker pull imagename و حالا میخواهیم از ان داکر ایمیج یک کانتاینر اجرا کنیم خود ان داکر از قبل لایه های و تنظیمات مورد نیاز ان اپلیکیشن را دارد و بعد از ان یک لایه نتورک در اختیار کانتاینر قرار میگیرد و سپس یک ای پی در اختیار ان کانتاینر قرار میگیرد و بسته به شل ی که در ان ایمیج از قبل وجود

داشته می توان با کانتاینر در تعامل بود.

فصل ۱۲

دستورات یا کامند های داکر

اولین کامند در داکر کامند docker info است که اگر daemon داکر اجرا شده باشد میتوان با ترمینال لینوکس یا powershell با ان ارتباط برقرار کرد (در این اموزش از power shell ویندوز استفاده شده و داکر را در ویندوز نصب کرده ام)



این کامند اطلاعات مفیدی را به ما می دهد. در قسمت server میزان کانتاینر های را میتوان دید containers و تعداد کانتاینر های در حال اجرا running و تعداد نگه داشته شده ها paused و تعداد متوقف شده ها stopped در اخر کل iamge های که pull شده اند.ورژن سرور ۱۹.۰۳.۱۲ است که از storage driver overlay2 استفاده می کند.برای اطلاعات بیش تر به این لینک مراجعه ت کنید[[5]](#footnote-5)

در لینوکس در این مسیر اطلاعات و زیر لایه های کانتاینر ذخیره می شود /var/lib/docker

مدل نوشتن کامند در داکر به این صورت است

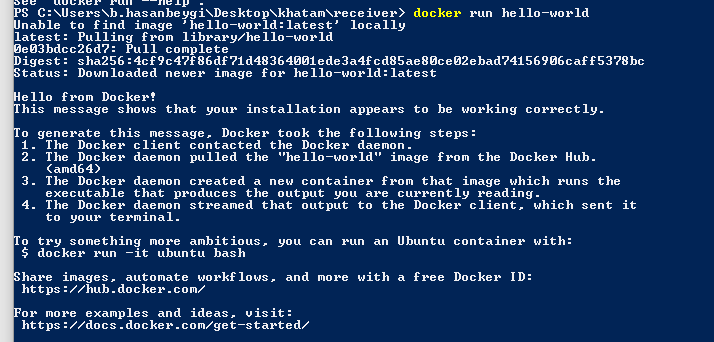
docker [option] [command] [arguments]

داکر هاب dockerhub یک داکر ریجستری است برای گرفتن ایمیج ها .به دلیل تحریم بودن کشور ایران و کره شمالی و سوریه و سودان دسترسی به داکر هاب از طریق اینترنت ایران امکان پذیر نیست و باید از فیلتر شکن یا تنطیم کردن دی ان اس ازاد برای گرفتن ایمیج ها استفاده شده بعضی شرکت ها مثل شکن این امر را اسان کرده اند برای این منظور ابتدا ای پی های شکن را در DNS خود تنظیم کنید اموزش تنظیم کردن DNS در ویندوز[[6]](#footnote-6) و سایت شکن[[7]](#footnote-7)

کامند بعدی docker run است

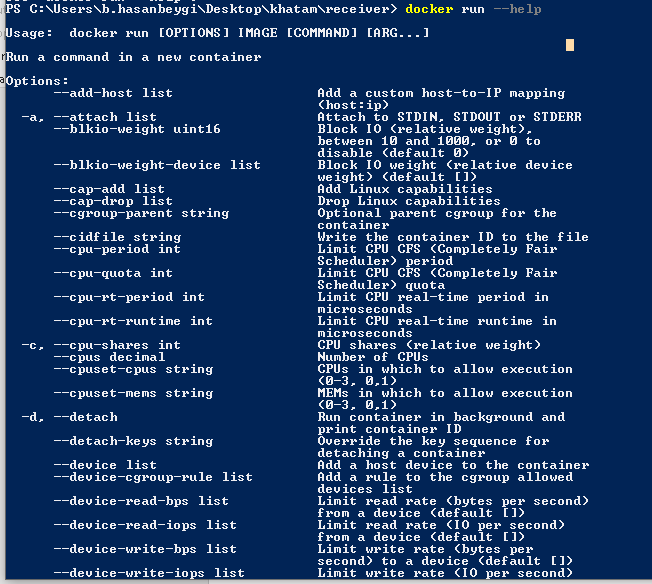
این کامند را اجرا کنید تا یک ایمیج از داکر هاب گرفته شده و سپس کانتاینر اجرا شود

Docker run hello-world



ابتدا ایمیج را از داکر هاب گرفته و سپس به شما خروجی بالا را نشان خواهد داد.

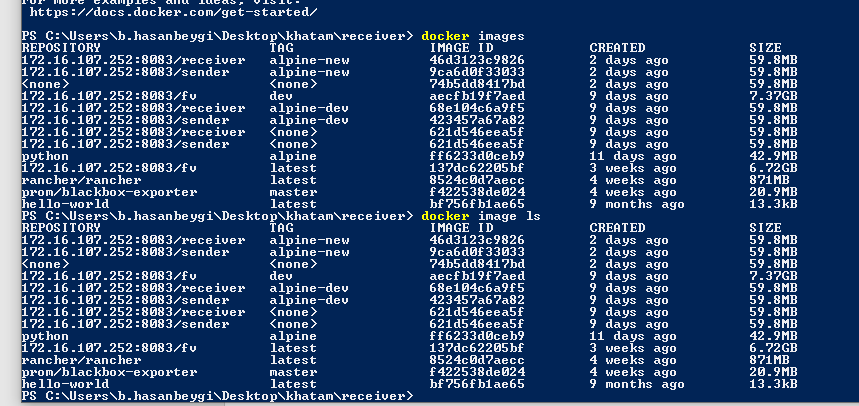
با زدن کامند docker run –help میتوانید flag ها یا سویچ های که همراه کامند run است را مشاهده کنید



برای مطالعه بیشر راجب کامند run[[8]](#footnote-8)

کامند بعدی docker images or docker image ls است.

این کامند لیست image های که بر روی سرور داکر شما وجود دارد را نشان می دهد.



این کامند اگر با flag یا سویچ -a باشد تمام image ها را نشان می دهد و اگر با -q باشد فقط ایدی image را نشان می دهد.

Docker images -a

Docker images -q

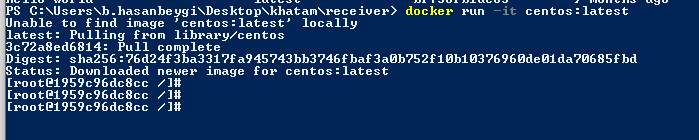
در این قسمت راجب دو flag یا سویچ میخواهیم صحبت کنیم -i و -t که همراه این کامند اجرا می شوند

Docker run -it image:tag

-i به معنایinteractive است یعنی با کانتاینر می تواند تعامل داشته باشد. -t به معنایی tty است که در مفهوم لینوکسی هر شخص زمانی ترمینال را اجرا میکند و شل اجرا می شود به ان یک tty اختصاص داده می شود.



با اجرای این کامند سرور داکر از داکر هاب image centos را pull کرد ه واجرا می کند.



الان کانتاینر centos ما اجرا شده و یک شل در اختیار ما قرار داده و1959c96dc8cc اسم هاست ما هست یا همان hostname. اگر یک پاور شل دیگر اجرا کنیم یا ترمینال دیگری و این کامند را اجرا کنیم می توان اجرا بودن کانتاینر را مشاهده کرد.

Docker ps



کامند بعدی راجب docker ps است

ما در لینوکس یک کامند داریم به صورت ps که کلیه پروسس ها یا برنامه ها یا سرویس های درحال اجرا را به ما نشان می دهد.

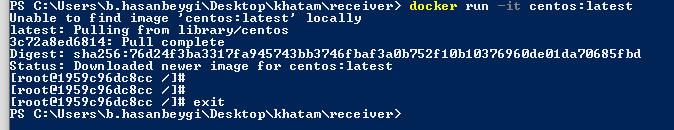
در سرور داکر این کامند نیز کانتاینر های در حال اجرا را به ما نشان می دهد.یک زمان نیاز است که کلیه کانتاینر های در حال اجرا و متوقف شده و همچنین نگه داشته شده را ببینم در عکس بالا اگر مشاهده کنید یک قسمت مربوط به status است که وضعیت کانتاینر را نشان می دهد اکر در حال اجرا باشد up اگر متوقف یا نگه داشته شده باشد Exited را نشان می دهد.flag و یا سویچ های زیادی را میتوان همراه docker ps اجرا کرد ولی دو سویچ معروف یکی -a و دگیر -q است

Docker ps -a کلیه کانتاینر های در حال اجرا یا متوقف شده یا نگه داشته شده را به ما نشان می دهد.

Docker ps -q ایدی کانتاینر های در حال اجرا(فقط در حال اجرا) را به ما نشان میدهد.

Docker ps -qa ایدی کلیه کانتاینر های در حال اجرا و متوقف شده یا نگه داشته شده را می توان دید.برا مطالعه بیشتر[[9]](#footnote-9)

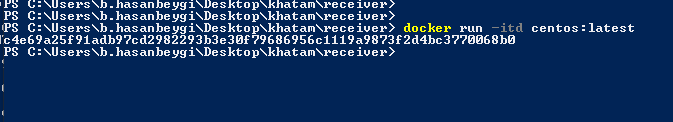
با زدن ctrl+d در پاورشل یا ترمینال در همان ترمینالی که image centos را اجرا کرده بودیم کانتاینر متوقف می شود.



در ادامه با flag یا سویچ -d اشنا می شویم در لینوکس مفهومی داریم به اسم foreground و background این دو مفهوم به این معناست که پروسس در پشت ترمینال یا tty اجرا شود و با بستن شدن ترمینال از بین نرود(background) در foreground پروسس در ترمینال اجرا می شود با بستن tty یا ترمینال پروسس از بین می رود.

در سرور داکر سویچ -d همان کار background را انجام می دهد.-d == detach

Docker run -itd centos:latest



و اگر کامند docker ps را اجرا کنید به شما خروجی زیر را می دهد.



تمرین : ۵ عدد کانتاینر centos را اجرا کنید که همگی در حالت background باشند سپس فقط ایدی کانتاینر ها را نشان بدهید.

حل :

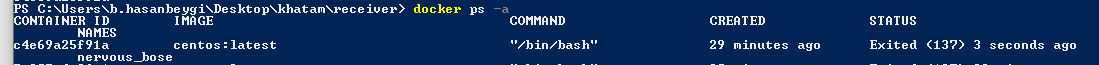
۵ بار کامند زیر را اجرا کنید

Docker run -itd centos:latest

و سپس با کامند رو به رو فقط ایدی را مشاهده کنید docker ps -q

کامند بعدی docker rm است.

برای پاک کردن کانتاینر های ایجاد شده سپس متوقف شده یا status == Exited از این کامند استفاده میکنیم به طور مثل یک centos را بالا اورده و سپس ایدی ان را در جلوی کامند قرار می دهیم به طور مثال:



با کامند رو به رو کانتاینر را می توان حذف کرد docker rm c4e69a25f91a

اگر کانتاینر UP باشد از کامند رو به رو استفاده می کنیم که کانتاینر را متوقف کنیم docker rm -f [containerID]

با سویچ -f کانتاینر را ابتدا متوقف و سپس پاک میکنیدForce

تمرین: یک کانتاینر centos را بالا بیارید سپس ان را پاک کنید

جواب:

Docker run -itd centos:latest

Docker ps

Docker rm -f [containerID]

به دلیل اینکه هر کانتاینر اسم مخصوص به خود را میگیرد(یعنی دو کانتاینر اسم مشابه ندارند)برای همین می توان از این روش هم کانتاینر را پاک کرد

Docker rm -f اسم کانتاینر

برای پاک کردن همه ی کانتاینر های UP AND EXITED

از این کامند استفاده میکنند:docker rm -f $(docker ps -a)

برای مطالعه بیش تر این لینک را بررسی کنید[[10]](#footnote-10)

کامند بعدی کامند بسیار کاربردی است

Docker cp با این کامند می توان از سروری که در ان داکر را اجرا کردید به کانتاینر خود فایل انتقال دهید

روش کار به این صورت است:

ابتدا یک فایل متنی را بسیازید مثل example.txt

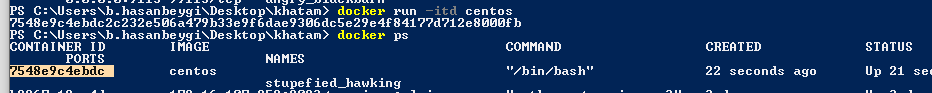
سپس با این کامند فایل را به کانتاینر خود انتقال دهید.

Docker run -itd centos:latest

کانتاینر را ایجاد کرده سپس با docker ps ایدی کانتاینر را به دست بیارید سپس :

Docker cp example.txt [containerID]:/path/

در عکس زیر به صورت کامل توضیح داده خواهد شد.



بعد از ساخت کانتاینر حالا از کامندdocker cp استفاده میکنیم:

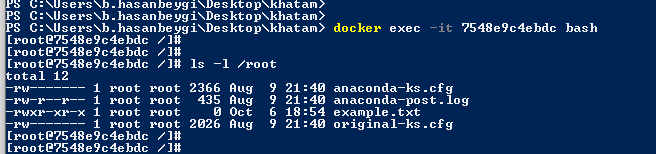


با این کامند فایل مورد نظر در /root کانتاینر کپی می شود.

برای چک کردن درست بودن این موضوع از یک کامند استفاده میکنیم تا بتوانیم وارد کانتاینر شده و بررسی را انجام دهیم این کامند به صورت مفصل بعدا توضیح داده می شود:

Docker exec -it 7548e9c4ebdc bash

سپس به /root بروید و ان را چک کنید:



می توان در دایرکتوری /root قایل example.txt را مشاهده کرد

سه کامند stop و start و attach

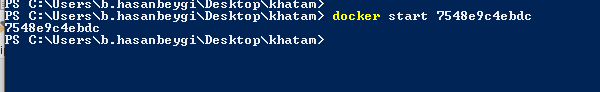
با زدن کامند docker stop میتوان کانتاینر در حال اجرا را متوقف کرد



که در این کانتاینر متوقف میشه

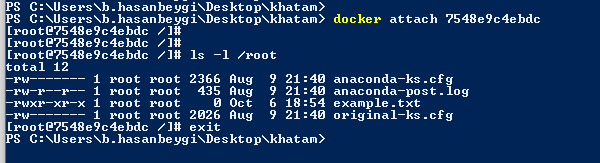
با این کامند میتوان کانتاینر را اجرا کرد docker start [containerID]

Docker start 7548e9c4ebdc



با اجرای کامند docker attach میتوان به کانتاینر وصل شد ولی بعد از زدن دکمه های ctrl+D کانتاینر متوقف می شود

Docker attach[containerID]

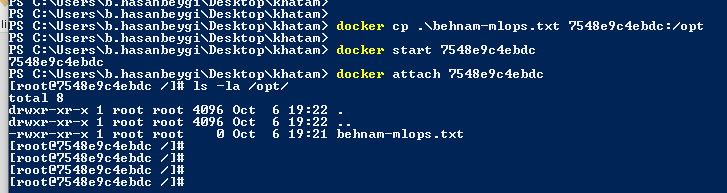


چون که کامند attach کانتاینر را در foreground اجرا میکند نه background برای همین بعد از زدن ctrl + D کانتاینر متوقف شده .

پروژه:

یک فایل بسازید سپس ان را به یک کانتاینر متوقف شده انتقال دهید و در اخر چک کنید که ان فایل در کانتاینر وجود داشته باشد.

حل:



کامند بعدی چگونگی اجرای متغییر ها در کانتاینر است . با سویچ -e میتوان در زمان ایجاد کانتاینر به ان متغییر را پاس داد

Docker run -it -e VAR IMAGE:TAG

به طور مثال :

export var=behnam

docker run -it -e var centos:latest

[root@0ee0762da564 /]# echo $var

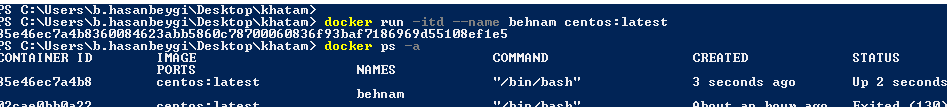
behnam

[root@0ee0762da564 /]#

در داکر یک سویچ نیز وجود دارد که با ان می توان ENV فایل را به کانتاینر پاس داد

به طور مثال یک فایل .env داریم که می خواهیم ان را به کانتاینر خود پاس داده با سویچ - -env-file .env میتوان ان را به کانتاینر پاس داد.

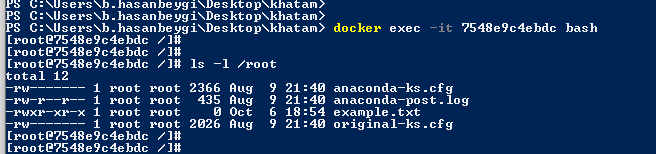
کامند بعدی راجب سویچ - - name میخواهیم صحبت کنیم با این سویچ میتوان برای کانتاینر خود اسم انتخاب کرد به طور مثال:



کامند بعدی docker exec است.این کامند برای وصل شدن به کانتاینر و ارتباط با ان است.با سویچ های – t که همان tty است و – i که به معنای interactive است می اید.ساختار این کامند به شکل زیر است :

Docker exec -it [containerID] bash

ان bash اخری مربوط به شل موجود در کانتاینر است که در زمان ساخت image در ان تعبیه شده میتوان به عنوان مثال از sh نیز استفاده کرد /bin/bash و [[11]](#footnote-11)/bin/sh



اگر در این زمان با دستور کیبرد ترکیبی ctrl+d از کانتاینر خارج شوید کانتاینر UP خواهند ما و در background در حال اجرا می ماند.

تمرین : یک کانتاینر با اسم خود بسازید وارد آن شوید و سپس خارج شده و ببینید که کانتاینر UP است یا نه!

docker run -itd - - name YOURNAME centos:latest

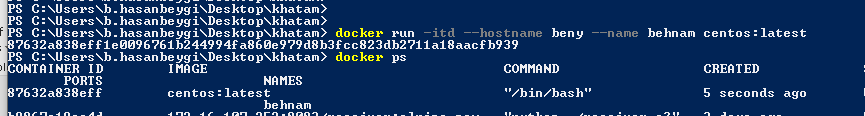
ctrl+d

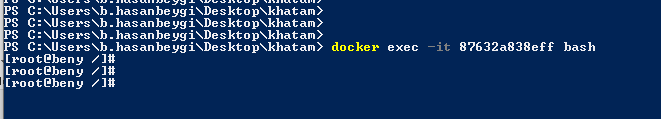
docker ps

کامند بعدی در رابطه با برقراری HOSTNAME در کانتاینر مورد نظر است با سویچ - - hostname

Docker run -itd - - hostname beny centos:latest

Docker exec -it [containerID] bash

 \

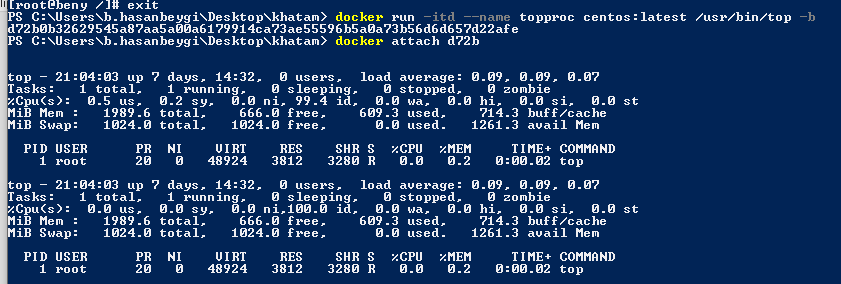


می بینید که در جای HOSTNAME nickname من قرار گرفته که beny است.

میتوان به یک کانتاینر در زمان START ان یک کامند خاص داد که به ان Entrypoint گفته می شود.

به طور مثال در زمان تشکیل یک کانتاینر میتوان کامند top که یک کامند لینوکسی است را اجرا کرد.

Docker run -itd - - name topproc centos:latest /usr/bin/top -b



که اگر مشاهده کنید در زمان اجرایی کانتاینر کامند top طور متناوب اجرا می شود

سویچ -b در حالت batchmode باعث اجرای کانتاینر می شود.

پروژه:

یک کانتاینر centos بسازید که اسمش اسم کوچک شما hostname نام فامیلی شما و یک متغییر به ان پاس دهید که اسم شما باشد.

حل: export var1=behnam

docker run-itd - - name behnam - -hostname hasanbeygi -e var1 centos:latest

فصل ۱۳

Docker image

داکر ایمیج از چندین لایه تشکیل شده که هر لایه نشان دهنده یک قسمت ساخته شده توسط Dockerfile (بعدا توضیح داده می شود) است.

یک مثال از Dockerfile را نشان می دهم تا این مسله را متوجه شوید.

FROM ubuntu:18.04

COPY . /app

RUN make /app

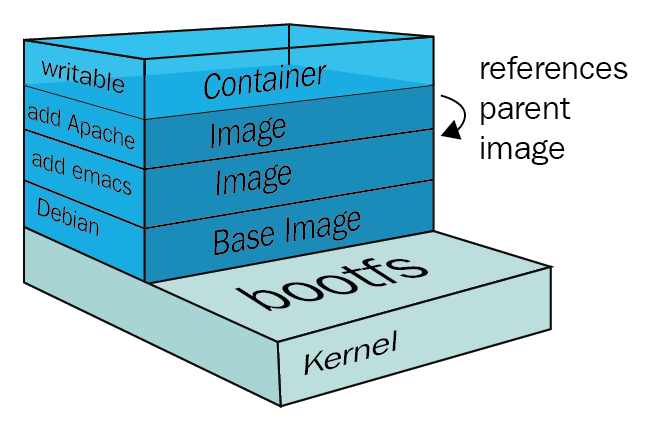
CMD python /app/app.py

در این Dockerfile در لایه اول گفته شده که از image ubuntu:18.04 استفاده شود یعنی پایه و به اصطلاح base ما ubuntu است در لایه بعدی گفته شده که کد های که در دایرکتوری جاری ان است در ایمیج کپی شود هر لایه که شروع به اجرا می شود یک کانتاینر موقت تشکیل شده و عمل دستور داده شده انجام می شود سپس کانتاینر موقتی پاک شده و در لایه image قرار میگیرد. در لایه بعدی گفته شده که کامند make را در پوشه ی که قبلا کد را کپی کرده ایم اجرا شود و در لایه اخر به image یک کامند ثابت را داده ایم که در صورت UP شدن یک کانتاینر از این image کد پایتونی که در دایرکتوری app ساخته ایم اجرا شود.

متوجه باشید در زمان تشکیل این image این لایه های READ ONLY هستند و بعدا نمیتوان بدون ساخت دوباره ان ها را تغییر داد ولی کانتاینر ها READ/WIRTE هستند.

اگر در زمان گرفتن image از docker hub دقت کرده باشید چندین لایه برای گرفتن بعضی image ها استفاده می شود که معنا مفهومش همین چیزی است که در بالای متن ذکر شد.

برای مطالعه بیشتر راجب image و لایه های ان به لینک زیر مراجعه کنید[[12]](#footnote-12)



در این شکل گفته شده که kernel linux در زیر لایه قرار دارد و در لایه بعدی filesystem های مربوط به boot پروسس ها قرار دارد و بعد از ان base image ما قرار گرفته که بعد از ما میتوانیم image های مربوط به پروژه ی خود را بسازیم.

برای متصل شدن به داکر هاب از کامند زیر می توان استفاده کرد.

Docker login hub.docker.com

که یک ریپوزیتوری بسیار بزرگ برای ذخیره image ساخته شده برای پروژه های متفاوت است.هر سرویسی که شما فکرش را بکنید در docker hub وجود دارد.

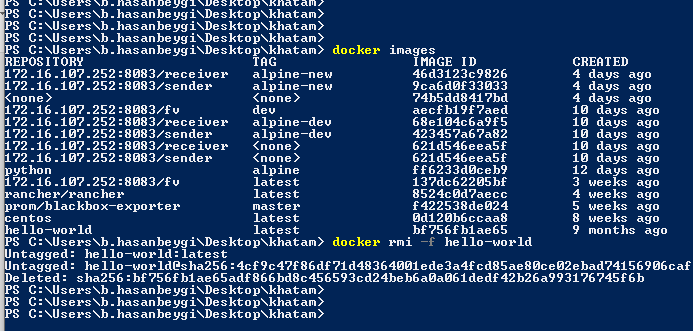
با کامند docker images می توانید image های را که ساخته اید یا اینکه از docker hub گرفته اید را مشاهده کنید.

برای حذف کردن یک iamge از سرور داکر خود می توان از کامند زیر استفاده کنید.

Docker rmi [imageID]

یا

Docker rmi IMAGE NAME



اگر در زمان حذف image برای شما error اینکه از image یک کانتاینر بالا امده و در حال استفاده است امد میتوانید با سویچ -f ان را force کنید ولی بدانید به دلیل اینکه کانتاینر وابسته به image است اگر image را پاک کنید کانتاینر نیز پاک می شود.

کامند بعدی که میخواهییم استفاده کنیم docker commit است .

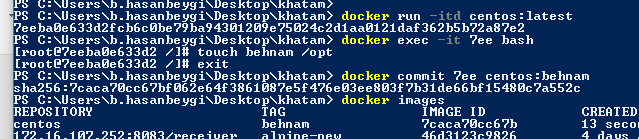
این کامند به شما کمک می کند در کانتاینر که ایجاد کرده این تغییرات دلخواه خود را اعمال کنید سپس تغییرات را در image که شما میخواهیید اسم گذاری کنید ذخیره یا اعمال کنید به طور مثال :

یک کانتاینر ایجاد کنید docker run -itd centos:latest

سپس به ان کانتاینر متصل شود docker exec -it [containerID] bash

سپس در یکی از دایرکتوری ها یک فایل ایجاد کنید touch behnam /opt

بعد از ان این کامند را بزنید docker commit [containerID] centos:behnam

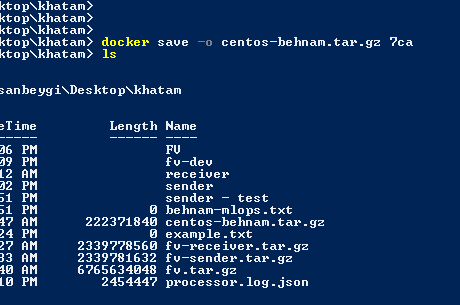


سپس اگر با iamge که خودتان ساخته اید یک کانتاینر را اجرا کنید در فایل /opt میتوانید فایل ساخته شده را ببینید.

کامند بعدی dokcer save است

با کمک این کامند می توان از image ساخته شده یا image که در سرور داکر خود دارید یک فایل تهیه کرده و سپس جا به جا کنید و در سرور دیگر این image را اجرا کنید.

Docker save -o filename.tar.gz [iamgeID]



کامند بعدی docker load است که همان فایل .tar.gz که ساخته اید را دوباره در سرور داکر به صورت یک image بالا بیاورید.

Docker load -I centos-behnam.tar.gz

برای دیدن کارای این کامند ابتدا باید image را که قبلا در همان سرور ساخته اید پاک کنید سپس با زدن این کامند image را دوباره در سرور بالا بیاورید.



برای مطالعه بیش تر به این لینک ها مراجعه کنید.[[13]](#footnote-13)[[14]](#footnote-14)

پروژه :

یک سرور centos را بسازید در ان پکیج vim را تصب کنید.سپس از ان کانتاینر یک commit بگیر به اسم centos:vim و بعد از ان یک کانتاینر از ان image که ساخته ایدUP کنید.

حل: docker run -itd centos

بعد : docker exec -it [containerID] bash

بعد yum install -y vim

بعد docker commit [containerID] centos:vim

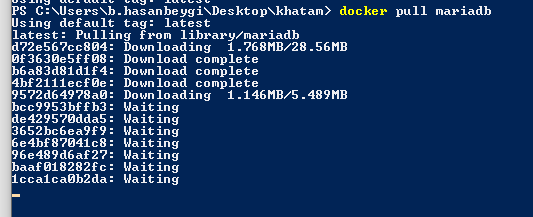
بعدdocker run -itd centos:vim

در این قسمت می خواهیم درباره image mysql or mariadb صحبت کنیم که برای اجرای ان در داکر باید یک پارامتر به صورت env به ان داده شود تا بتوان از کانتاینر ان استفاده کرد:

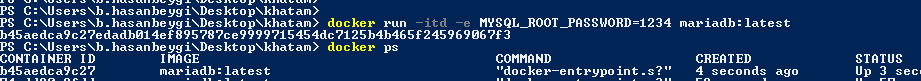
Docker pull mariadb:latest

Docker run -itd -e MYSQL\_ROOT\_PASSWORD1234 mariadb:latest

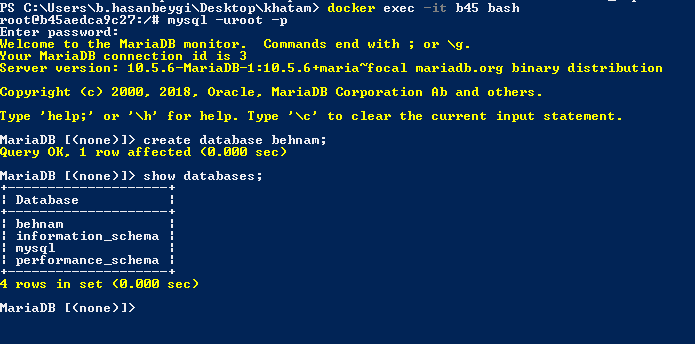
حتما باید پارامتر MYSQL\_ROOT\_PASSWORD به همین شکل به کانتاینری که میخواهیم بسازیم پاس داده شود و پسوورد داده شود که من پسوورد ۱۲۳۴ را انتخاب کرده ام.



سپس یک کانتاینر از image mariadb ایجاد میکنیم:

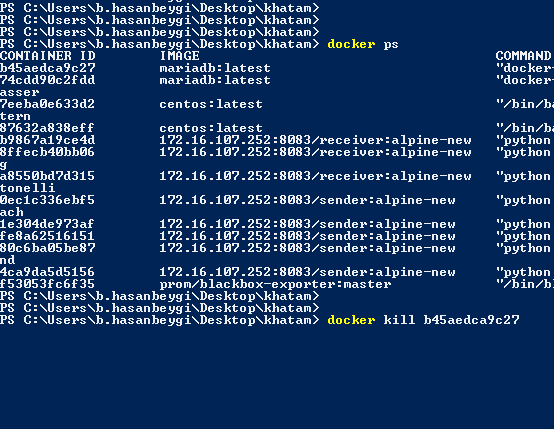


با کامند exec وارد کانتاینر بشید و یک دیتبابیس در کانتاینر بسازید:



کامند بعدی کامند docker kill :

در لینوکس ما برای مدیریت پروسس ها یک کامند داریم به نام kill که می توان با این کامند به process ها signal فرستاد که یا متوقف شود یا از بین بروند. کامند docker kill شبیه به kill است با این تفاوت که فقط پروسس را از بین میبرد یعنی باعث می شود کانتاینر به صورت force پایین بیاید.



Docker kill [containerID] و کانتاینر متوقف می شود ولی حذف نمی شود.برای مطالعه بیش تر راجب docker kill [[15]](#footnote-15)

کامند docker import and export :

با کمک این کامند های می توان از یک کانتاینر یک فایل tar ساخت و در سرورداکر دیگری ان را بالا اورده و شروع به کار کرد.

Docker export [containerID] > MYCONTAINER.TAR

و در سرور داکر دیگری :

Docker import MYCONTAINER.TAR



کامند docker restart :

با زدن این کامند کانتاینر restart می شود.

Docker restart [containerID]

فصل ۱۴

Docker volumes

زمانی که شما یک کانتاینر را اجرا می کنید خود ان کانتاینر نمی تواند اطلاعات شما را ذخیره (persist) کند یعنی به صورت دایمی اطلاعات را داشته باشید و با از دست رفتن کانتاینر شما کلیه اطلاعاتان پاک می شود. به طور مثال فرض کنید یک کانتاینر mariadb برای دیتابیس خود در سرور داکر اجرا کرده باشید زمانی که کانتاینر شما kill شود و سپس remove کلیه اطلاعاتی که درون کانتاینر دارید از بین می رود.برای همین منظور استفاده از docker volume ضروری است

مزایای استفاده از volume:

۱ از volume می توان به سادگی بک اپ گرفت و از محیطی به محیط دیگر انتقال داد.

۲ میتوان از طریق docker CLI volume ها را مدیریت کرد

۳ volume ها هم در کانتاینر های لینوکسی قابل استفاده اند هم در ویندوزی

۴ volume ها رو میتوان بین چندین کانتاینر share کرد.



کانتاینر برای ذخیره اطلاعات از فضای host استفاده میکند.برای اطلاعات بیش تر راجب docker volume[[16]](#footnote-16)

با flag یا سویچ -v در زمان اجرایdocker run میتوان به کانتاینر فهماندکه میتواند به host ماونت(mount) شود.

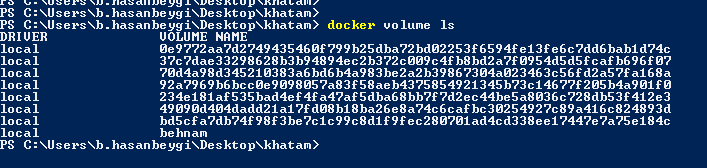
در ابتدا با این کامند یک volume در سرور داکر خود ایجاد کنید( :(best practice

Docker volume create - - behnam

با این کامند شما یک volume در سرور داکر خود ایجاد کرده اید که میتواند اطلاعات را در خود ذخیره کند



سپس با کامند docker volume ls میتوان volume های که وجود دارد را لیست کنیم یا ببینیم:

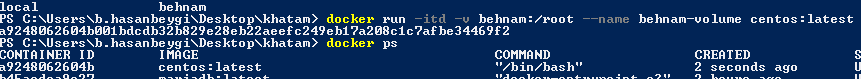


در این مرحله با زدن این کامند می توان از volume که در سرور داکر ایجاد کرده ایم استفاده کنیم:

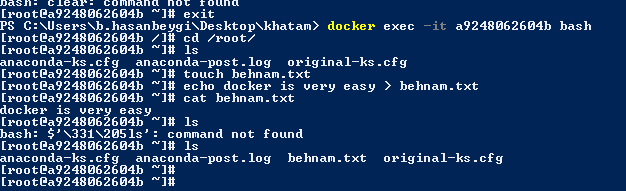
به طور مثال:

Docker run -itd -v behnam:/root centos:lates

با زدن این کامند دایرکتوری /root کانتاینر به هاست سرور mount می شود یعنی هر تغییراتی در /root کانتاینر اتفاق بیافتد در volume بهنام همان اتفاق خواهد افتاد:



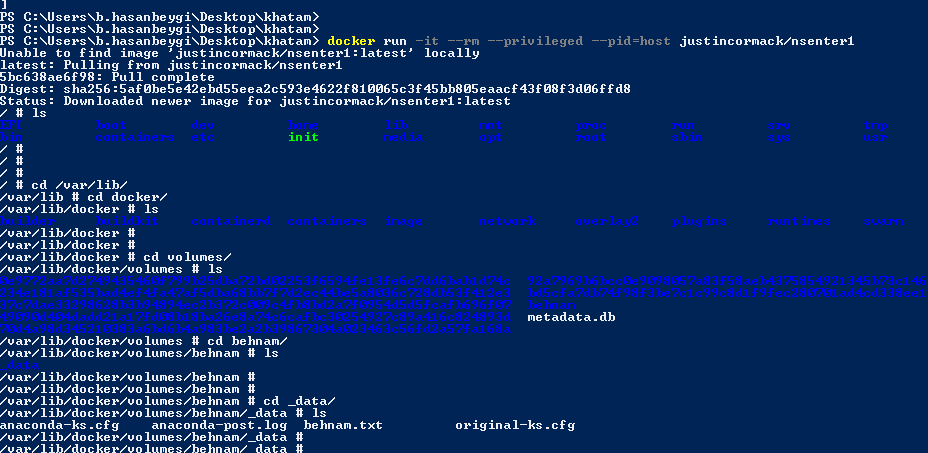
و برای تست کردن به کانتاینر وصل شید و در دایرکتوری /root یک فایل برای تست بسازید:



حالا برای تست این که در هاست شما این فایل وجود دارد یا نه چک میکنید(در لینوکس):

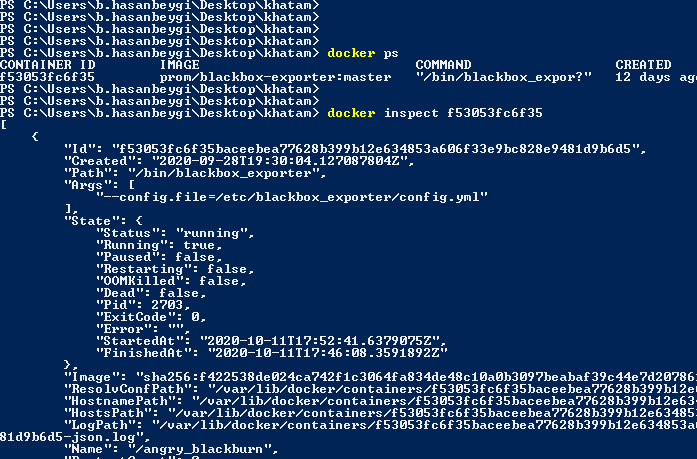
به دایرکتوری /var/lib/docker می رود و در انجا volume که ساخته اید(behnam for example) را می توانید ببینیدو در دایرکتوری -data میتوانید فایل behnam.txt را ببینید.

در ویندوز این مسله به دلیل اینکه سرور داکر در یک hyper v اجرا می شود نمیتوان به مسیری که گفته شد مراجعه کرد برای دیدن نتیجه.برای همین باید به سرور داکر در hyper v وصل شد و سپس نتایج را دید.



با کامند docker inspect میتوانید کلیه مشخصات کانتاینری که ایجاد کرده اید را ببینید.

Docker inspect [containerID)



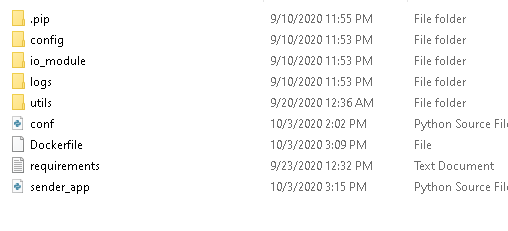
بعد از زدن این کامند کل مشخصات کانتاینری که ایجاد کرده اید را می توان دید.در مشخصاتی که وجود دارد قسمت IP کانتاینر و همچنین volume که به ان mount شده است برای ما اهمیت دارد.

در سرور های لینوکسی به دلیل اینکه می توان مستقیما از host سرور استفاده کرده میتوان بدون ساختن volume برای ذخیره اطلاعات مستقیما دایرکتوری مد نظر را به کانتاینر اختصاص داد اگر با سیستم عامل لینوکسی کار میکنید می توان با زدن این کامند دایرکتوری مد نظر را اختصاص داد:

Docker run -itd -v /opt/:/root centos:latest

توجه: در قسمت -v /opt:/root شما دارید دایرکتوری opt سیستم عامل را به root کانتاینر mount میکنید.یعتی اینکه در قسمت اول دایرکتوری سیستم عامل و درقسمت بعدی کانتاینتر HOST:CONTAINER

به طور مثال شما یک elastic search برای لاگ خوانی اپلیکیشن خود داریدبه دلیل اینکه container ایزوله است شما دسترسی به لاگ های درون کانتاینر ندارید و باید فایل یا دایرکتوری مربوط به کانتاینر خود را به سیستم عامل متصل کرده و سپس از ان استفاده کنید به طور مثال:



این اپلیکیشن یک app پایتون flask است که در ما در کد اصلی نوشته ایم که در فایل logs لاگ های مربوط به اپلیکیشن را قرار دهد. در زمان اجرای کانتاینر ما باید برای اینکه elastic search سرور بتواند این لاگ ها را بررسی کند به کانتاینر دستور دهیم که دایرکتوری logs درون کانتاینر را به host مانت(mount) کند برای همین به این شکل عمل میکنیم.

Docker run -itd -v /opt/applogs:/logs MYAPP

که در این حالت لاگ ها در فایل /opt/applogs هاست ما قرار میگیرد.

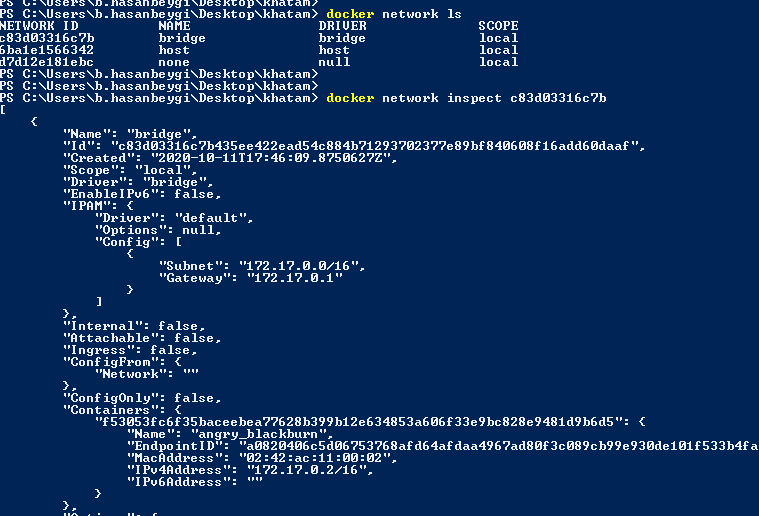
فصل ۱۵

Docker network

بعد از نصب سرور داکر یک network به شبکه سیستم عامل شما یا سرور شما اضافه می شود با range ۱۷۲ که اگر با

سیستم عامل های لینوکس کار می کنید در ترمینال کامند ifconfig یا ip a را بزنید میتوانید interface نتورکی که برای داکر به وجود امده را مشاهده کنید ولی در ویندوز به دلیل اینکه سرور داکر در hyper v اجرا می شود خود hyper v نتورکی را برای سرور داکر به وجود می اورد

با زدن کامند docker network ls می توانید network های را که در سرور داکر به وجود امده را مشاهده کنید.



در قسمت های قبل راجب کامند inspect صحبت شد کلا زمانی که از inspect استفاده می شود منظور این است که میخواهیم راجب ان سرویس اطلاعات کامل را به دست اوریم.

در شکل بالا مشخص است که network bridage و network host و network null سه network است که به صورت default در network سرور داکر وجود دارد.

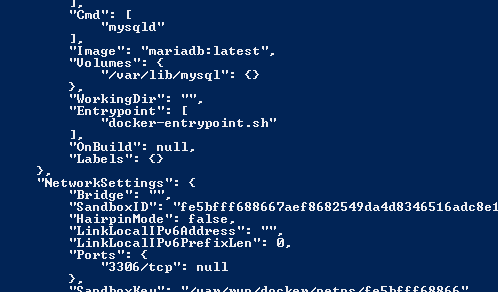
Network bridage با رنج ۱۷۲ در سرور داکر فعال است اگر کانتاینر خود را با کامند docker inspect چک کنید رنج ip که گرفته ۱۷۲ است.

می توان در داکر network های مجزا از هم تشکیل داد که بر اساس سیاست های تعریف شده بتوان در ان network ها اپلیکیشن ها با یکدیگر در ارتباط باشند مانند vlan در cisco . این مطلب را در docker-compose مفصل توضیح خواهم داد.

قدرتمند بود سرور داکر در مبحث network مربوط به port fowrading است .استفاده از port های هاست برای publish شدن اپلیکیشن به بیرون از محیط کانتاینر .به طور مثال اگر شما یک اپلیکیشن node js در بک اند خود داشته باشید برای ارتباط با react.js که فرانت شما است باید یک پورت را به بک اند خود اختصاص دهید مانند پورت ۳۰۰۰ و فرانت شما با ip هاست شما با پورت ۳۰۰۰ به بک اند وصل می شود.IP:PORT

به عنوان مثال : ( در لینوکس)

ابتدا یک کانتاینر mariadb بسازید سپس با کامند docker inspect جزییات کانتاینر را نگاه کنید اگر به قسمت port نگاه کنید پورت ۳۳۰۶ که پورت مخصوص سرور mysql است را می توانید ببینید.

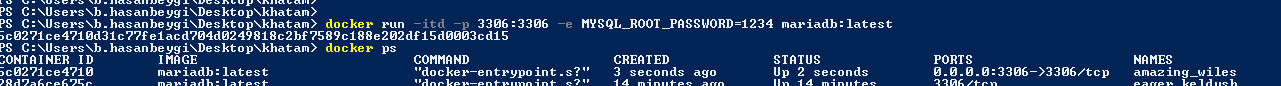


اگر مشاهده کنید پورت ۳۳۰۶ را مشخص کرده که برای ارتباط داشتن این کانتاینر با اپلیکیشن های دیگر باید این پورت به پورت host سرور شما forward شود که اصطلاحا port forwarding می گویند.

برای این منظور از flag یا سویچ -p استفاده می شود که به این صورت است.

Docker run -itd -p HOSTPORT:CONTAINERPORT image:tag

در ابتدا پورت هاست را می دهیم در قسمت بعدی کانتاینر.



اگر در قسمت پورت نگاه کنید به این صورت است 0.0.0.0:3306  3306 یعنی اینکه در iptables سرور یک role تعریف کرده که از هر ip اگر port 3306 هاست یا ip شما را صدا زدن یا call کردن کانتاینر جواب دهد.

اگر با کامند telnet در ترمینال لینوکس خود با ip کانتاینر و پورت 3306 را اجرا کنید باید به سرور mysql وصل شود.

telnet 172.7.0.2 3306

Trying 172.7.0.2...

Connected to 172.7.0.2.

برای تست این سناریو میتوان یک کانتاینر سرور mysql در یک سرور داکر جدا ایجاد کرد و در یک هاست دیگر اما در همان network با زدن این کامند به سرور mysql وصل شد

Mysql -uroot -p -h 192.168.x.x -p 3306

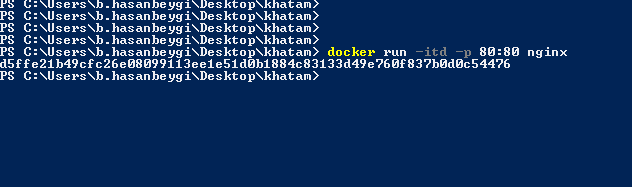
که به ip:port داده شده همان کانتاینر سرور mysql است متصل می شود.

توجه : هر کانتاینر به یک پورت وصل می شود و نمیتوان همزمان دو کانتاینر را به یک پورت وصل کرد. به طور مثال اگرپورت ۳۰۰ هاست شما درگیر یک اپلیکیشن دیگر باشد و شما یک اپ nodejs داشته باشید که کانتاینر در پورت ۳۰۰۰ می تواند فعالیت کند برای اینکه بتوانید با react که فرانت شما است در ارتباط باشید باید یک پورت دیگر را به بک اند خود اختصاص دهید به طور مثال ۳۰۰۲ و به فرانت خود بگویید که بک اند در پورت ۳۰۰۲ فعالیت میکند به حالت معمول ما به تعداد ۶۵۵۳۵ پورت داریم که می توانیم از ان ها استفاده کنیم.

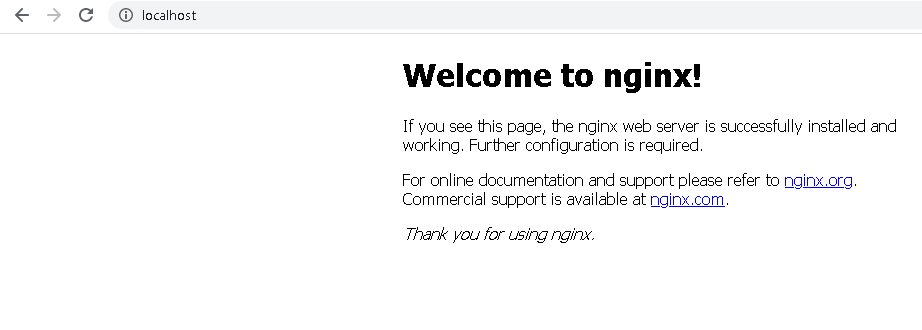
اگر میخواهییم پورت ۳۰۰۲ هاست را به پورت ۳۰۰۰ nodejs خود اختصاص دهیم با زدن این کامند باید این عمل انجام شود.

Docker run -itd -p 3002:3000 nodejs

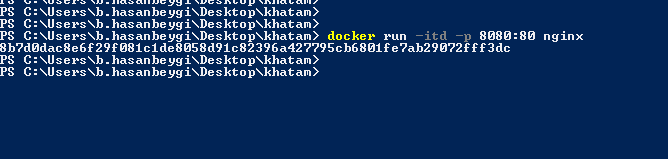
مثال: nginx یک وب سرور است.ابتدا ان را از هاب داکر در سیستم خود pull کنید سپس با زدن این کامند ان را اجرا کنید



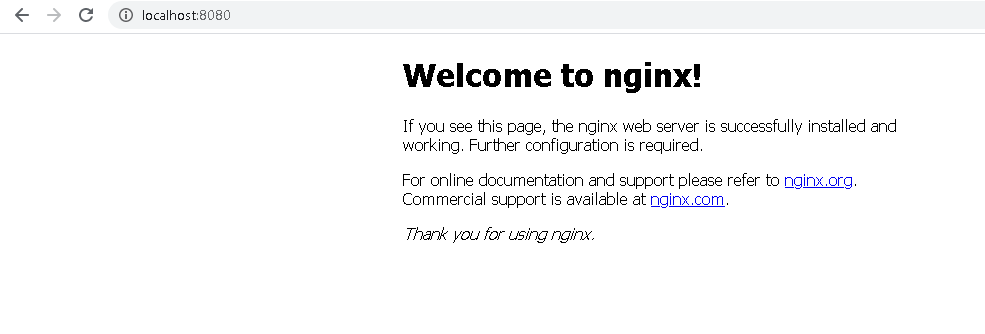
سپس به مرورگر خود بروید ودر قسمت ادرس بار ip:80 یا localhost:80 را سرچ کنید تا نتیجه را ببیند.منظور از ip همان ip هاست است که سرور داکر را در ان deploy کرده اید



خب حالا بیاید port هاست را تغییر دهید.



همان طور که می بینید پورت اولی که مربوط به هاست تغییر پیدا کرده و ۸۰۸۰ شده حالا دوباره ip:8080 یا localhost:8080 را سرچ کنید



پس می توان با تغییر پورت ها سرویس های متفاوتی را در سرور داکر ایجاد و با ان ها تعامل کرد.

فصل ۱۶

ساختن Docker image

در hub.docker.com بی شمار image وجود دارد ان هم به دلیل اجتماع قوی از برنامه نویس ها و سیستم ادمین های که به docker علاقمند هستند و شما هر سرویسی را که تصور کنید در docker میتوانید پیدا کنید.

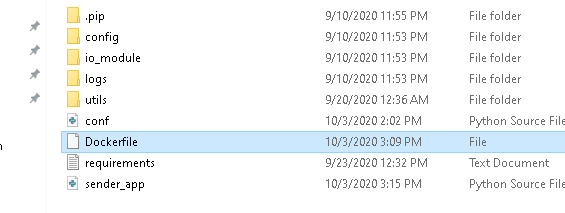
یک سناریو:

شرکت از شما درخواست کرده که اپلیکیشن خود را بر روی سرور داکر پیاده سازی کنید و شما هیچ تفکری نسبت به ساخت image برای اپلیکیشن خود ندارید.

ابتدا باید بدانید که ساختن image در داکر یک قواعد و اصولی دارد که باید از ان پیروی کرد و در بعضی از اپلیکیشن ها به دلیل متفاوت بودن نوع نصب وابستگی های اپلیکیشن و یا نوع اجرا شدن ان نوشتن Docker image قدری متفاوت می شود ولی ساختار همان است.

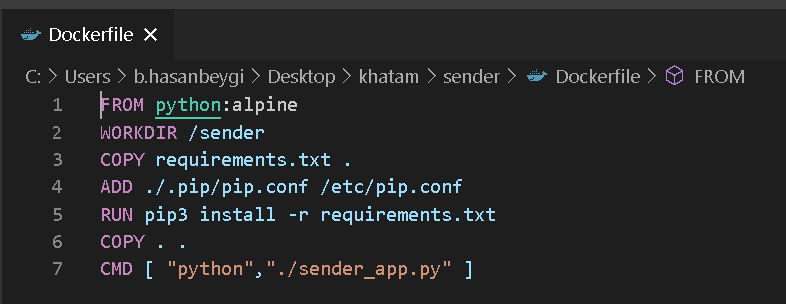
به طور مثال یک Dockerfile را با هم بررسی می کنیم:

ابتدا باید یک دایرکتوری بسازید که کلیه قسمت ها و وابستگی های اپلیکیشن شما در ان وجود داشته باشد.



بعد در همان دایرکتوری که ساخته این یک فایل ایجاد کنید به اسم Dockerfile که حتما باید D بزرگ نوشته شده باشد.

Dockerfile به این شکل است:



هر کدام از قسمت ها را به طور کامل توضیح خواهم داد: (با کلمات بزرگ شبیه به کلمات بنفش نوشته شود)

FROM : به معنی این است که از کدام image پایه می خواهید استفاده کنید .هر اپیلیکیشنی برای ساخته شدن نیاز به یک base image یا ایمیج پایه دارد که من در اینجا برای ساخت این اپلیکیشن از python:alpine استفاده کردم که یک image سبک و با حجم پایین است.

WORKDIR : به معنی این است که می خواهم در یک دایرکتوری مشغول به انجام کاری بشوم ان محیط را برای شما سرور داکر محیا می کند

توجه: هر کدام از این قسمت ها خود یک layer یا لایه است که در ان یک کانتاینر به صورت موقتی ایجاد شده تغییرات اعمال می شود و بعد commit و یک ایمیج تشکیل می شود برای مرحله ی بعد

COPY : requirements.txt را در دایرکتوری /sender کپی کن

ADD : شبیه به قسمت copy است و فایل مربوطه را در دایرکتوری مورد نظر کپی میکند.این قسمت که می بینید من یک nexus repository را راه اندازی کردم و در ان وابستگی های python خود را نگه داری میکنم به این دلیل که بعدا نیاز شد برای نصب در جای دیگر دوباره از اینترنت استفاده نکنم و از همین nexus استفاده کنم.

RUN : در این قسمت کامندی که مد نظر است برای اینکه اپلیکیشن را اماده سازی میکند اجرا می شود. در اینجا به طور مثال اپلیکیشن python ما وابستگی باید نصب شود این کامند را گذاشته ایم

COPY : . . این دو نقطه با فاصله از هم به این معنی است که محتویات درون فایلی که داریم را در محیط کانتاینر کپی کن و سپس commit بگیر و ایمیج را بساز و به مرحله بعد برو

CMD : همان کامند است . اگر این قسمت را در داکر ایمیج خود قرار دهیم میتوانیم به کانتاینر بفهمانیم که هر زمان UP شد کامندی که مد نظر ماست را اجرا کنید.یعنی اینکه اجرا شدن کانتاینر منوط می شود به اجرا شدن اپلیکیشن که در این جا چون اپیلیکیشن python است به این شکل ان را اجرا کرده ایم.

حالا بقیه قسمت های Dockerfile را توضیح خواهیم داد:

MAINTAINER : به معنی نویسنده یا کسی که docker image را ساخته است و به صورت اختیاری میتوان به Dockerfile اضافه کرد

ENV : می توان با این قسمت به docker image خود variable اختصاص داد

ENTRYPOINT: شبیه به CMD است که می توان با ان به صورت default به کانتاینر دستور داد که هر زمان UP شد کامتد مد نظر را اجرا کند

USER : میتوان با این قسمت به image UID خاصی را اختصاص داد

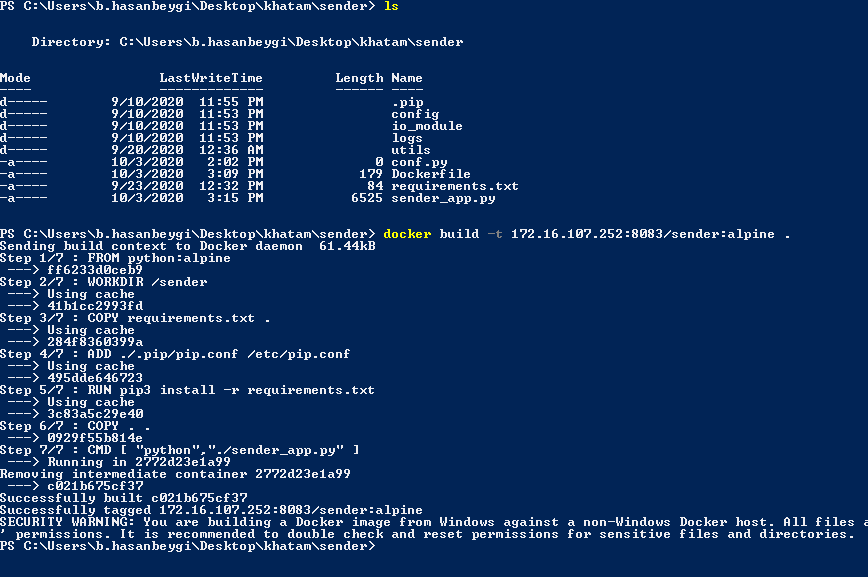
VOLUME: با این قسمت در image مشخص میکنید که کدام قسمت از کانتاینر با کدام دایرکتوری یا فایل از هاست شما mount باشد.

برای توضیحات بیشتر[[17]](#footnote-17)

بعد از اینکه Dockerfile را ایجاد کرده اید حالا باید ان را بسازید با زدن این کامند ایمیج شما ساخته خواهد شد.این کامند را باید در دایرکتوری که dockerfile را ایجاد کرده اید اجرا کنید powershell or linux shell

Docker build -t imageName:tag .

در اخر کامند یک نقطه است به معنی اینکه dockerfile شما در این دایرکتوری قرار دارد و ان را شناسایی کند.



اگر مشاهده کنید هر قسمتی را که در Dockerfile مشخص کرده ایم به صورت یک step یا قدم حساب کرده سپس به مرحله بعد می رود.

شاید در ابتدای ساخت این ایمیج یکم زمان ساخت ان طول بکشد ولی در زمان های بعدی ساخت به دلیل اینکه cache می شود سرعت ساخت ایمیج خیلی زیاد است.

فصل ۱۷

Docker compose

Compose به معنای قطب نماست. و واقعا قطب نمای است برای رسیدن به اپلیکیشن های خود . داکر کومپوز یک کامپوننت یا یکی از ابزار های داکر است که در لینوکس جداگانه باید setup شود.[[18]](#footnote-18)

نصب docker compose در لینوکس:

اخرین ورژن (در زمان نوشتن این اموزش ) را در سرور لینوکس خود دانلود کنید

sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

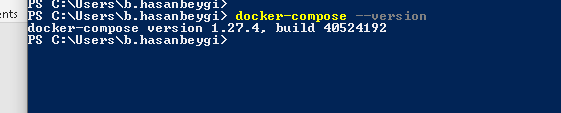
با زدن این کامند ورژن ۱.۲۷.۴ داکر کامپوز در usr/local/bin/docker-compose دانلود شده. سپس باید با زدن این کامتد به داکر کامپوز permission اجرای بدهید.

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

و در اخرین با زدن این کامند ورژن داکر کامپوز خود را چک کنید

Docker-compose –version

که باید این خروجی به شما نمایش داده شود:

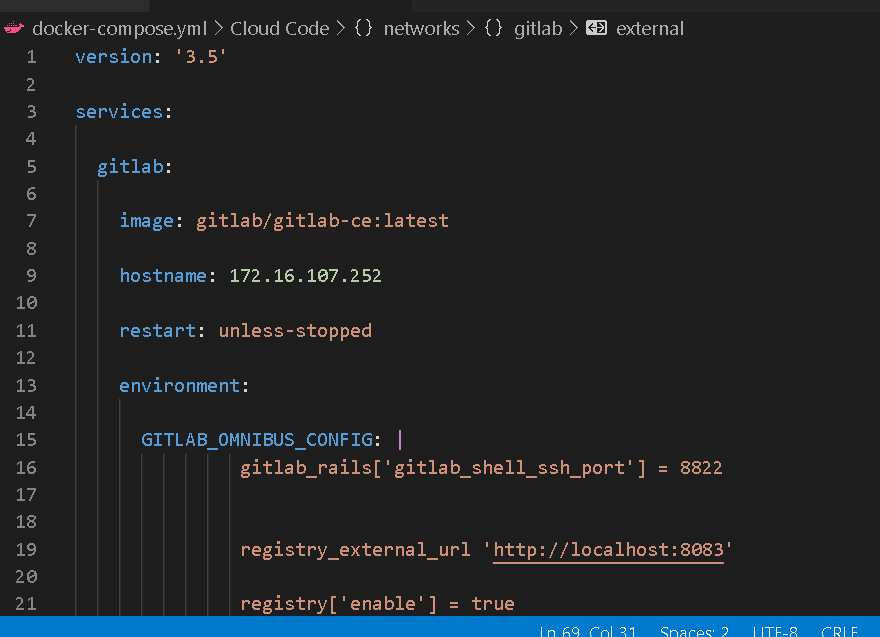


داکر را اگر در ویندوز استفاده می کنید داکر کامپوز همراه با docker destktop نصب شده و نیازی به نصب ان ندارید

Docker-compose به شما قابلیتی می دهد که چندین کانتاینر را فقط با نوشتن یک فایل اجرا کنید و نیازی نیست کامندی را اجرا کنید فقط کافیست یک فایل داکر کامپوز بنویسید.

برای مثال یک سرور gitlab که یک repository برای نگه داری کد ها و همچنین برای مبحث CI/CD است را در اینجا نوشته و راجب ان توضیح خواهیم داد:

ابتدا باید یک فایل به اسم docker-compose.yml ایجاد کنید و به زبان yaml داکرکامپوز را بنویسید .در زبان yaml اینتند ها (intend ) بسیار مهم هستند یعنی فاصله ها و قرار گیری ها بسیار مهم است.



Version : ‘3.5’ : به معنای این است که از ورژن ۳.۵ داکر کامپوز استفاده می شود.

در قسمت services ما کانتاینر های که نیاز داریم UP را می نویسیم

اولین سرویس gitlab نام دارد که اسم سرویس ماست

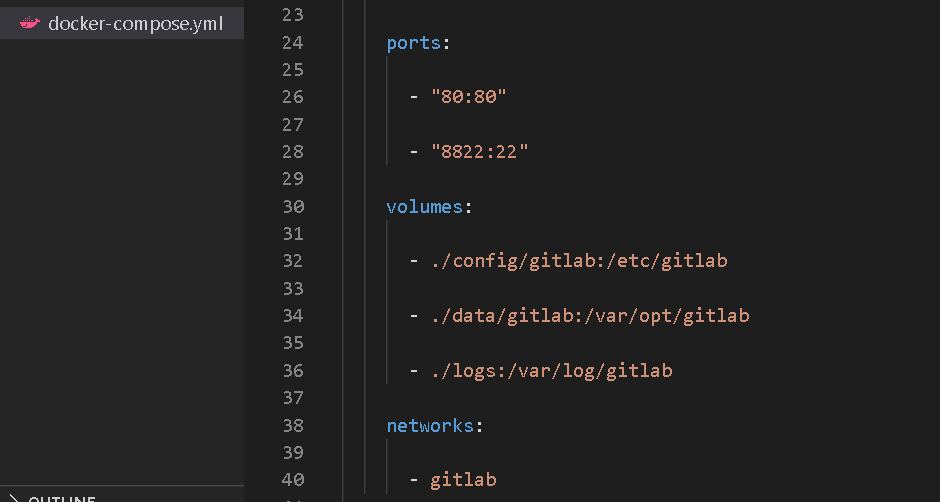
در زیر قسمت gitlab از image گیت لبی که خود کمپانی gitlab در اختیار ما قرار داده استفاده میکنیم.

Hostname را ip سروری که میخواهیم gitlab را در ان نصب کنیم می نویسیم.

در قسمت restart ما یک سیاست یا policy تعریف کرده ایم که هر موقع سرور gitlab ما متوقف شد سرور را ریستارت کرده تا دوباره سرور در دسترس قرار بگیرد.

در قسمت environment ما متغییر های که نیاز است برای ارتباط با سرور gitlab را می نویسیم

ادامه:

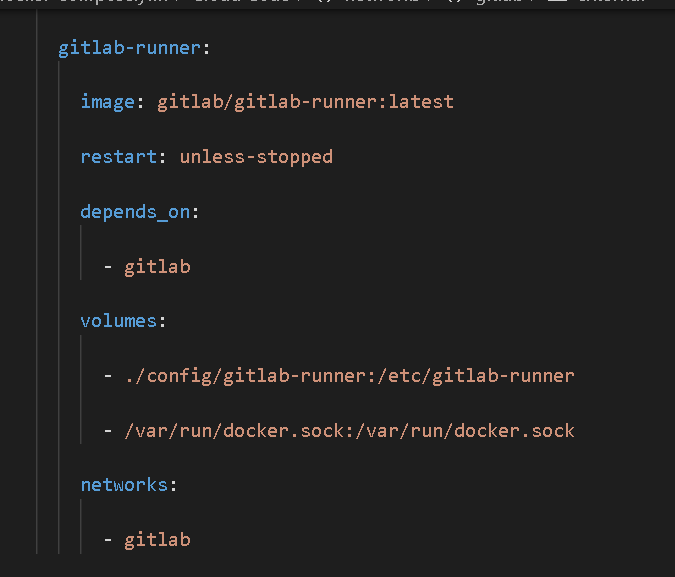


در قسمت ports ما پورت های که نیاز است برای اجرای سرور را نوشته ایم هم ردیف environment در شکل بالا

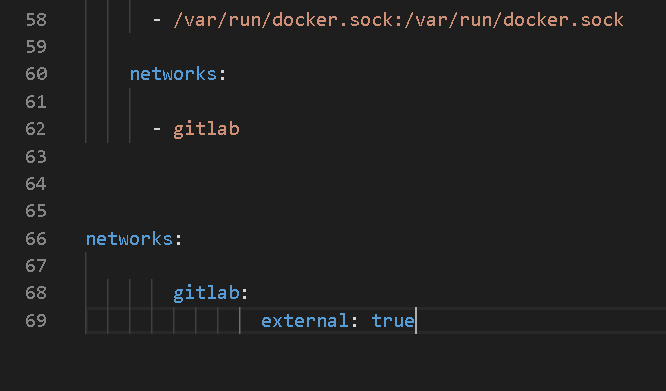
سپس volume های که نیاز است به هاست ما mount شوند را مشخص کرده و می نویسیم

در اخر network را نوشته که یک شبکه اختصاصی در سرور داکر ایجاد کرده برای ارتباط داشتن بین کانتاینر های مختلف در این شبکه که هم دیگر را بتوانند پیدا کنند اگر قسمت network را مشخص نکرده بودیم نیز خود داکر کامپوز یک شبکه به صورت default می ساخت و در اختیار ما قرار می داد.

در ادامه:



سرویس دوم که gitlab runner است که مربوط به CI/CD است را نوشته فقط همان قسمت depends\_on را توضیح می دهم یعنی اینکه به سرور gitlab ما وابستگی دارد و تا کامل سرور gitlab ما بالا نیاید runner ما اجرا نخواهد شد.



در اخر باید شبکه ی که میخواهییم در سرور داکر اجرا شود تعریف کنیم که معنی external : true این است که از شبکه ی دیگر نیز بتوانند سرور gitlab ما را ببینند

بعد از نوشتن این docker-compose.yml فایل ابتدا با این کامند image های مورد نیاز را از docker hub گرفته ( کامند را در همان دایرکتوری که فایل را ساخته اید اجرا کنید)

Docker-compose pull

بعد از گرفتن ایمیج ها این کامند را اجرا کنید تا کلیه سرویس ها UP شوند

Docker-compose up -d

برای بررسی اینکه سرویس ها بالا هستند یا نه می توان از کامند زیر استفاده کرد

Docker-compose ps

کلا هر کامندی که مربوط به داکر کامپوزی که میخواهیید سرویس ها را در ان اجرا کنید باید در همان دایرکتوری که فایل docker-compose.yml را ایجاد کرده اید بنویسید.

Docker-compose down سرویس ها را پایین می اورد

Docker-compose restart gitlab سرور گیت لب شما را ریستارت می کند

Docker-compose logs -f gitlab لاگ های که سرور gitlab شما می سازد را به شما نشان می دهد.

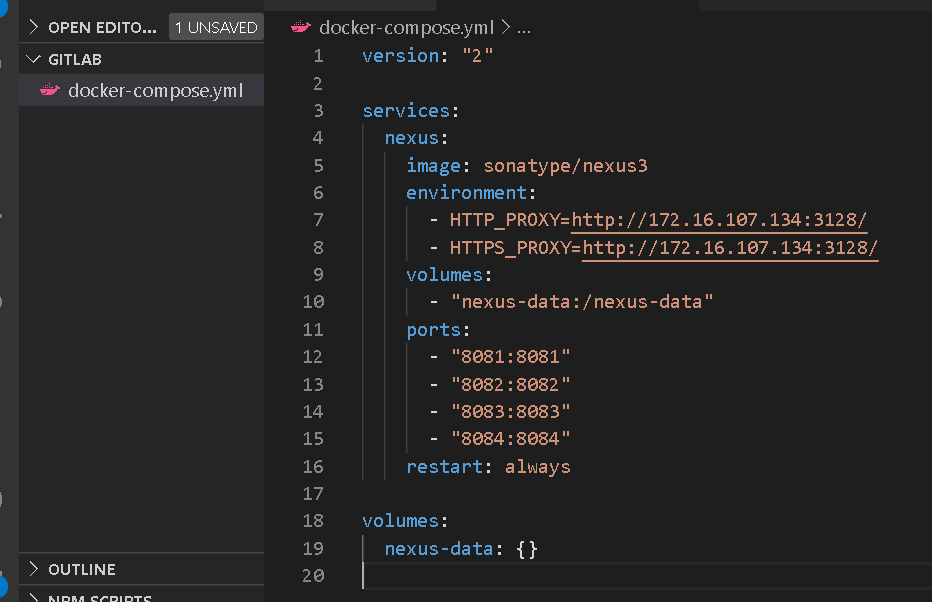
برای مطالعه بیش تر راجب docker-compose [[19]](#footnote-19)

فصل ۱۸

Docker registry

Registry به معنای یک مخزن یا repository برای ذخیره image های است که می سازید.در اینجا من قصد دارم ریجستری sonatype nexus را برای شما اموزش بدهم که یکی از قوی ترین registry ها برای ذخیره image های داکر پکیج های پایتون یا هر اپلیکیشن مد نظر شما

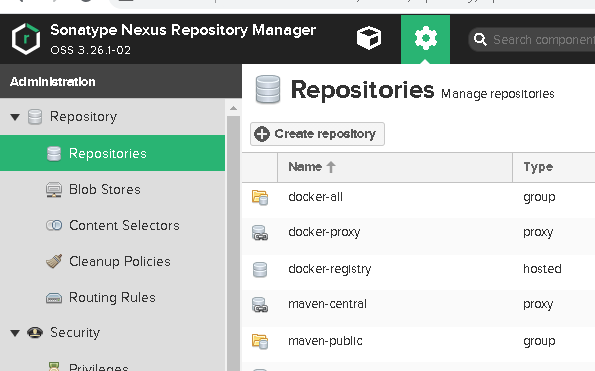
ابتدا یک فایل docker-compose.yml میسازیم برای اجرای سرور nexus



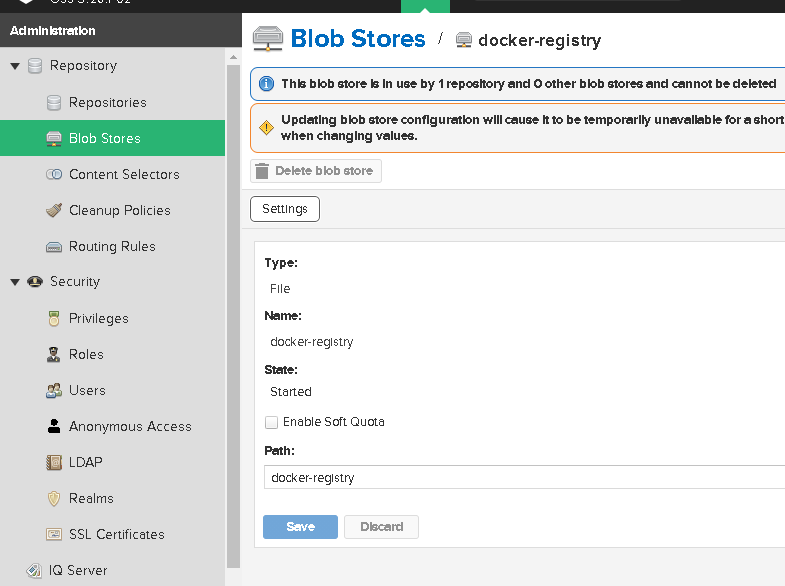
قسمت HTTP\_PROXY را نادیده بگیرید به دلیل اینکه سرور های من از طریق پراکسی اینترنت می گیرنداین environment را تعریف کرده ام و شما اگر از اینترنت به صورت مستقیم استفاده میکنید نیازی به این بخش ندارید

فقط قسمت port های برای ما مهم است زمانی که 8081 برای web-ui یا صفحه ی که میخواییم nexus را در ان مشاهده کنیم و 8082 برای پراکسی و ذخیره کردن image های در nexus اس و ۸۰۸۳ برای insecure registry است.

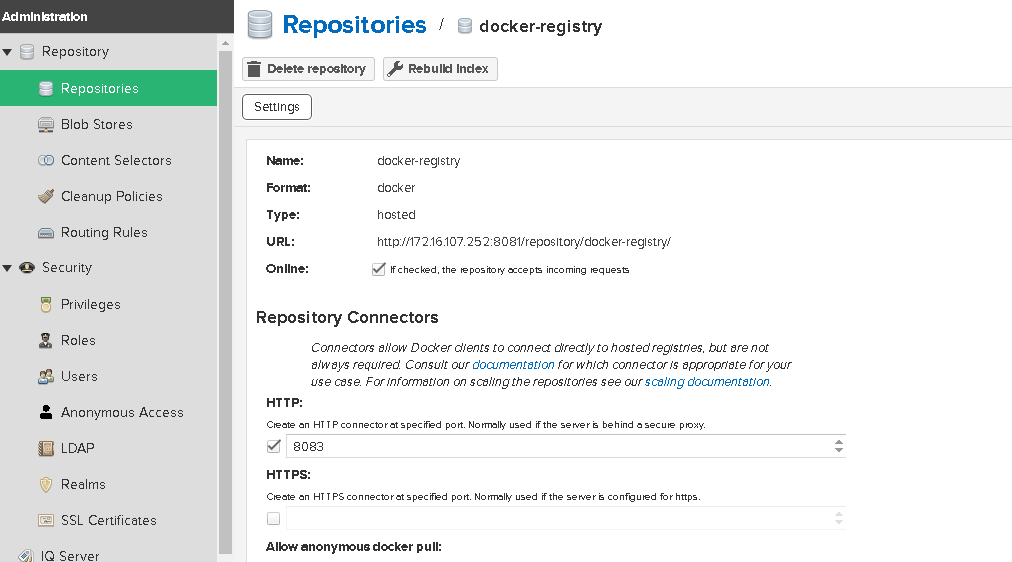
زمانی که شما کامند docker-compose up -d را اجرا کنید سرور برای شما اجرا می شود و باید صفحه ی لاگین در سرور nexus را مشاهده کنید.



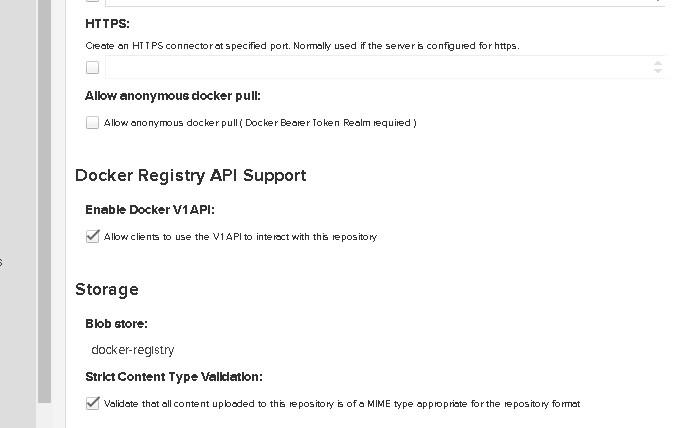
در قسمت blob stores ابتدا باید registry خود را بسازید



بعد از ساخته شدن به قسمت repository بروید و بر روی docker-registry که ساخته کلیک کنید.



ادامه تنظیمات



و بعد از رو گزینه save کلیک کرده و داکر رجستری شما ساخته خواهد شد.

به دلیل اینکه registry را که اجرا کردید از طرف داکر غیر امن خوانده می شود شما باید به سرور داکر خود تنظماتی بدهید که مشکل insecure-registry را برطرف کنید

در لینوکس به مسیر /etc/docker/daemon.json بروید ان را vim کنید سپس در ان فایل این تنظیم را قرار دهید.

}

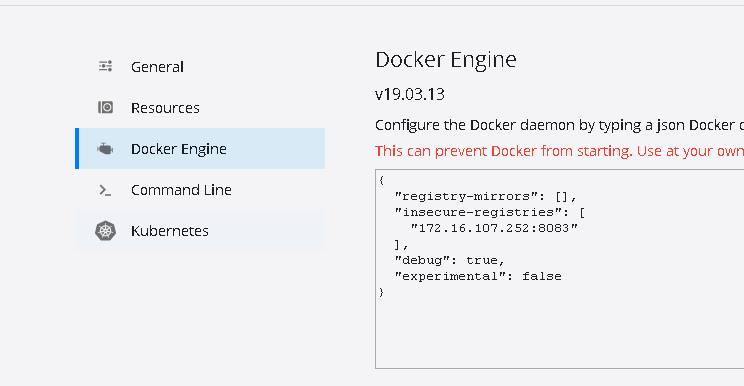
“insecure-registries” : [ “IP:8083“ ]

}

سپس سرور را ریستارت کرده systemctl restart docker و بعد از ان کامند

Docker log ip:8083 را زده و با یوزر نیم و پسوورد مورد نظر لاگین می کنید

در ویندوز:

در setting داکر دسکتاپ و سپس در قسمت docker engine

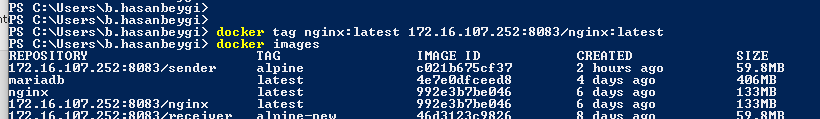
Insecure -registries را به همین شکل نوشته و داکر را ریستارت می کنیدو به همان شکل قبلی لاگین کرده

برای اینکه ایمیج هایتان را در سرور nexus یا هر registry که راه اندازی کرده اید pull یا push کنید باید این قواعد یا سبک مدل را رعایت کنید

ایمیج nginx را که از قبل pull کرده اید با این کامند تغییر اسم دهید

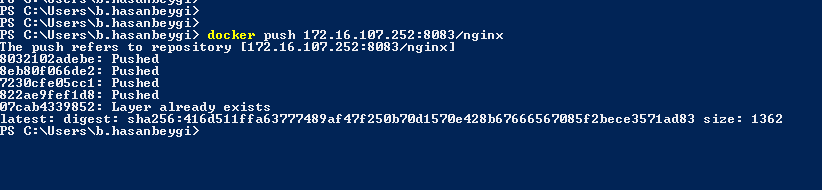
Docker tag nginx:latest IP:8083/nginx:latest

به جای ip باید ip سرور nexus خود را قرار دهید :

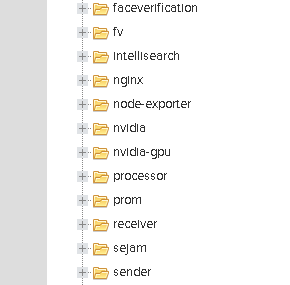


و با زدن این کامند image nginx را به registry ارسال کنید

Docker push IP:8083/nginx:latest



اگر به سرور nexus بروید می توانید ایمیجی را که ارسال کرده اید را ببینید



و با کامند زیر می توانید این ایمیج را در هر سروری که insecure-registries را برای ان تعریف کرده pull کنید

Docker pull IP:8083/nginx:latest

تمام

ورژن ۰.۰۹

اگر سوالی راجب هر نوع tools متن باز داشتید می تونید از من بپرسید

[Behnam.hasanbeygi@gmail.com](mailto:Behnam.hasanbeygi@gmail.com)

Linkedin.com/behnam-hasanbeygi

1. https://stackoverflow.com/questions/45018786/what-is-the-exact-difference-between-docker-ee-enterprise-edition-docker-ce [↑](#footnote-ref-1)
2. https://www.docker.com/resources/what-container [↑](#footnote-ref-2)
3. https://docs.docker.com/engine/ [↑](#footnote-ref-3)
4. https://www.docker.com/get-started [↑](#footnote-ref-4)
5. https://docs.docker.com/engine/reference/commandline/info/ [↑](#footnote-ref-5)
6. https://www.zoomit.ir/2019/7/7/337508/dns-server-change-settings-windows10/ [↑](#footnote-ref-6)
7. https://shecan.ir/ [↑](#footnote-ref-7)
8. https://docs.docker.com/engine/reference/commandline/run/ [↑](#footnote-ref-8)
9. https://docs.docker.com/engine/reference/commandline/ps/ [↑](#footnote-ref-9)
10. https://docs.docker.com/engine/reference/commandline/rm/ [↑](#footnote-ref-10)
11. https://docs.docker.com/engine/reference/commandline/exec/ [↑](#footnote-ref-11)
12. https://docs.docker.com/storage/storagedriver/ [↑](#footnote-ref-12)
13. https://docs.docker.com/engine/reference/commandline/save/ [↑](#footnote-ref-13)
14. https://docs.docker.com/engine/reference/commandline/load/ [↑](#footnote-ref-14)
15. https://docs.docker.com/engine/reference/commandline/kill/ [↑](#footnote-ref-15)
16. https://docs.docker.com/storage/volumes/ [↑](#footnote-ref-16)
17. https://docs.docker.com/engine/reference/builder/ [↑](#footnote-ref-17)
18. https://docs.docker.com/compose/install/ [↑](#footnote-ref-18)
19. https://docs.docker.com/compose/ [↑](#footnote-ref-19)