

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 – داکر image
- 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 آن 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 می کنید 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 می گرفت. 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 اپلیکیشن 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**

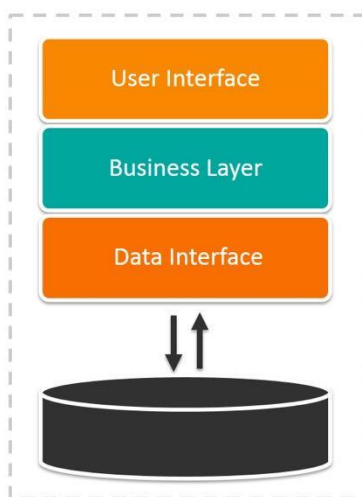
مزایای معماری **microservices** نسبت به **monolithic** :

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 کنید ولی در معماری **microservices** به دلیل جدا سازی سریع تر انجام خواهد شد.

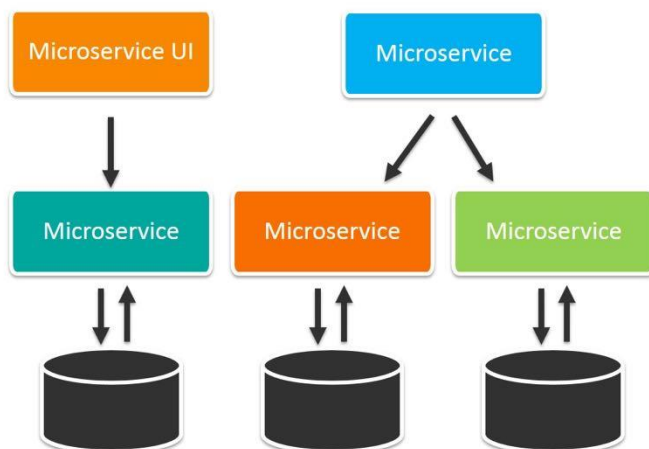
۲ در معماری **monolithic** یک اشتباه کل اپلیکیشن را از دسترس خارج خواهد ولی در **microservices** به دلیل طراحی درست میتوان به سرعت آن مشکل را برطرف کرد

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

Monolithic Architecture

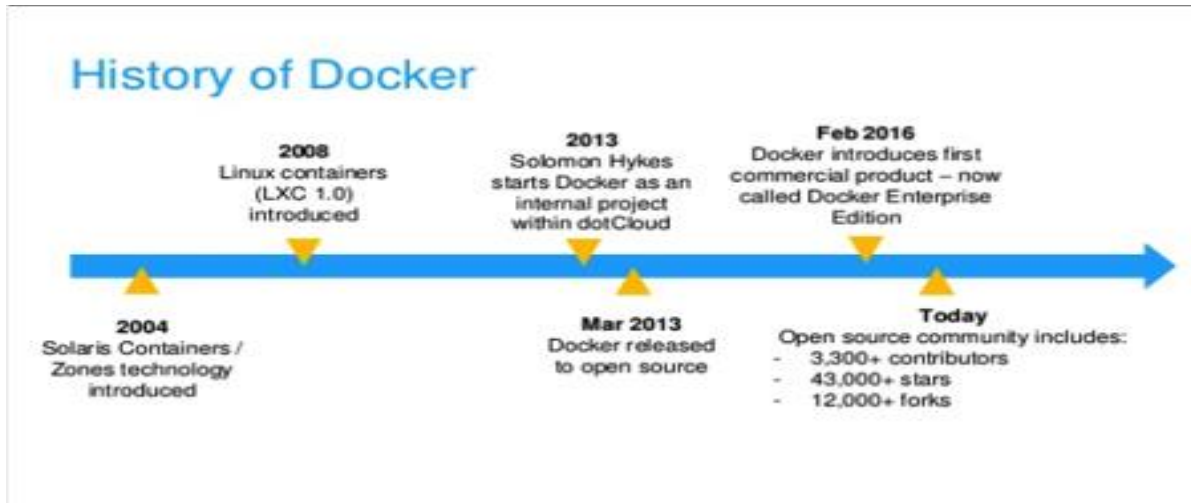


Microservices Architecture



فصل دوم

تاریخچه ی داکر



در سال ۲۰۰۴ ابتدا سولاریس کانتاینر توسط شرکت zone technology معرفی شد در سال ۲۰۰۸ لینوکس کانتاینر و سپس در سال ۲۰۱۳ آقای 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 آمد 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 آن ۱۹.۰۳.۱۳ است.

فصل سوم

داکر چیست؟

Docker Container as a Service Platform



Docker
Toolbox

- “docker push” with image signing



Docker
Trusted Registry

- Search/browse repos
- Teams-based RBAC
- View signed images
- Deleting tags



Docker Universal
Control Plane

- Authentication
- Deploy and scale-out app
- Monitor stats
- Secrets management



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

Build any app

Ship any where

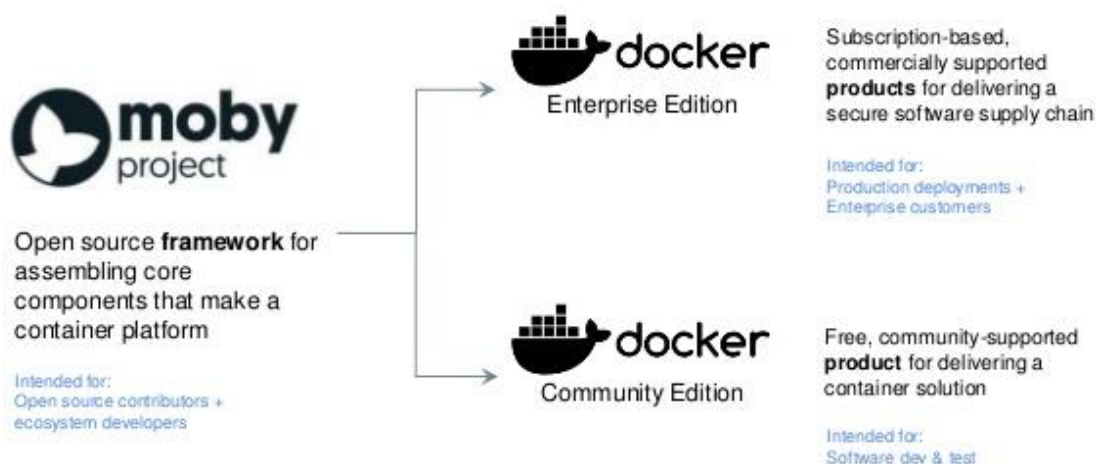
Run in any where

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

The Docker Family Tree

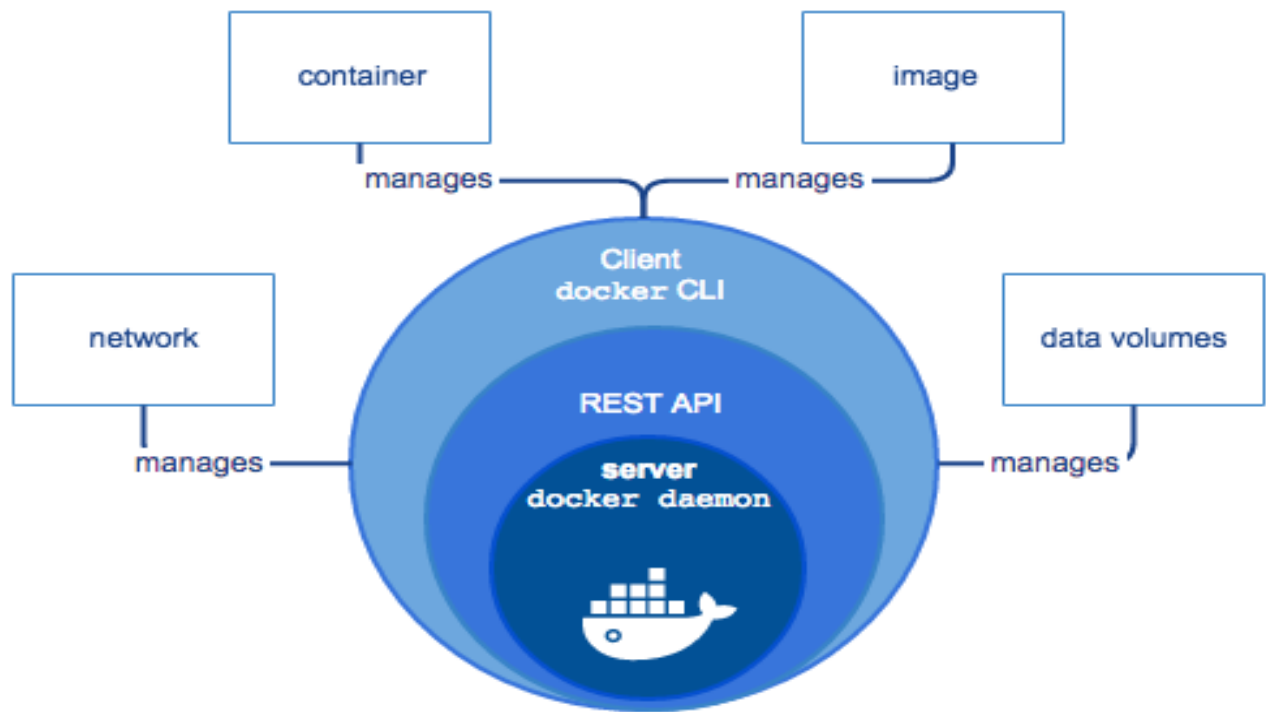


پروژه ی موبی تشکیل شده از دو نسخه ی داکر که یکی **docker enterprise edition** که همان نسخه ی پولی داکر است که ویژگی های کمی پیشرفته تر نسبت به نسخه متن باز آن دارد و نسخه ی **community** که متن باز است و برنامه نویس های از سراسر دنیا بر روی این پروژه کار میکنند. برای مطالعه بیشتر راجب تفاوت این دو نسخه از لینک زیر استفاده کنید¹

فصل پنجم

کلیات داکر

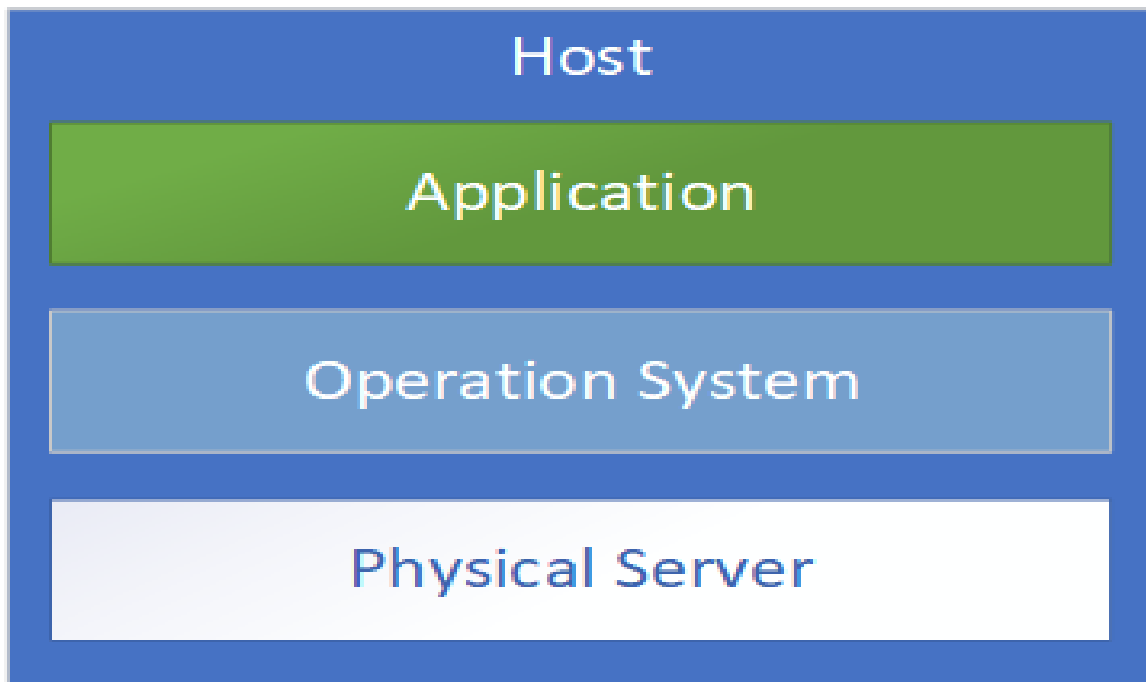
¹ <https://stackoverflow.com/questions/45018786/what-is-the-exact-difference-between-docker-ee-enterprise-edition-docker-ce>



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

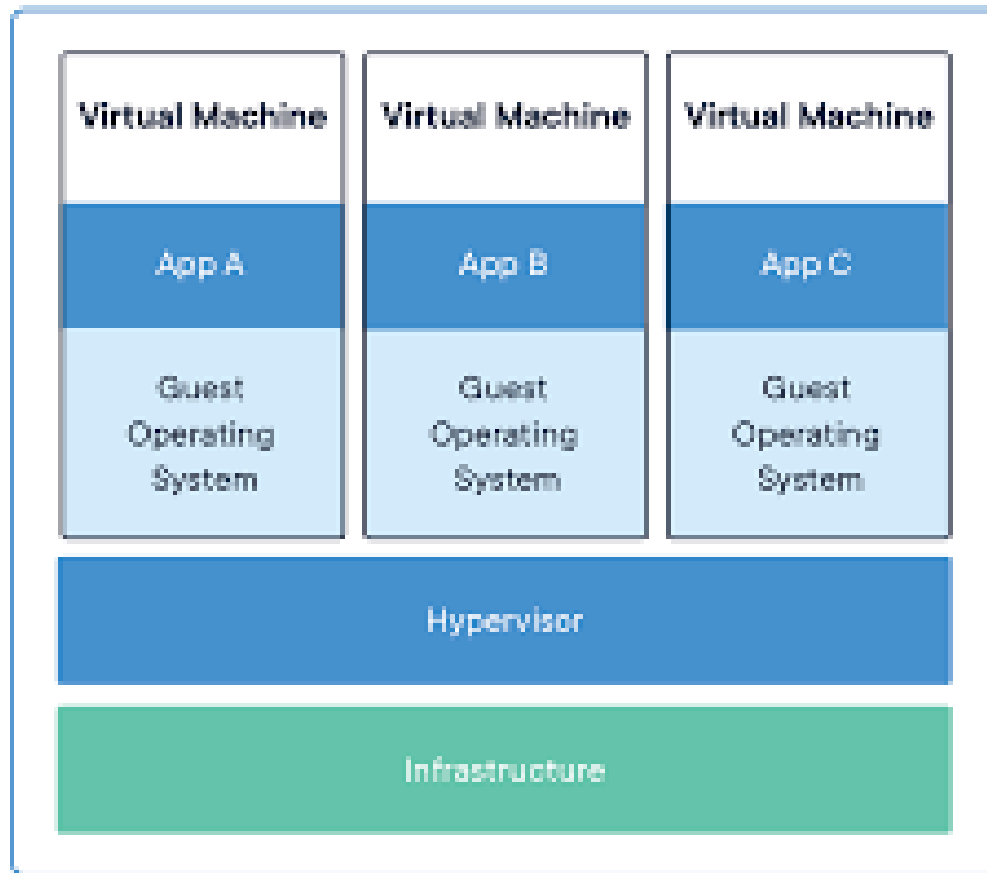
مشکلات نصب اپلیکیشن در زمان پیاده سازی



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

- ۱ زمان پیاده سازی بسیار کند بود
- ۲ هزینه 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 فیزیکی میتوانستیم چندین سیستم عامل را اجرا کنیم که مفهوم virtualization یا مجازی سازی به میدان آمد. اپلیکیشن های بسیاری برای این منظور ساخته شد که vmware از مشهورترین پلتفرم ها بود که سیستم عامل ESXI روی سرور فیزیکی نصب می شد. از اپلیکیشن 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 و مفهوم های جدیدی که در این زمان به وجود آمده بود به سختی در این معماری جای می گرفت مانند مفهوم CI/CD یا

Continuous integration and continuous development(delivery)

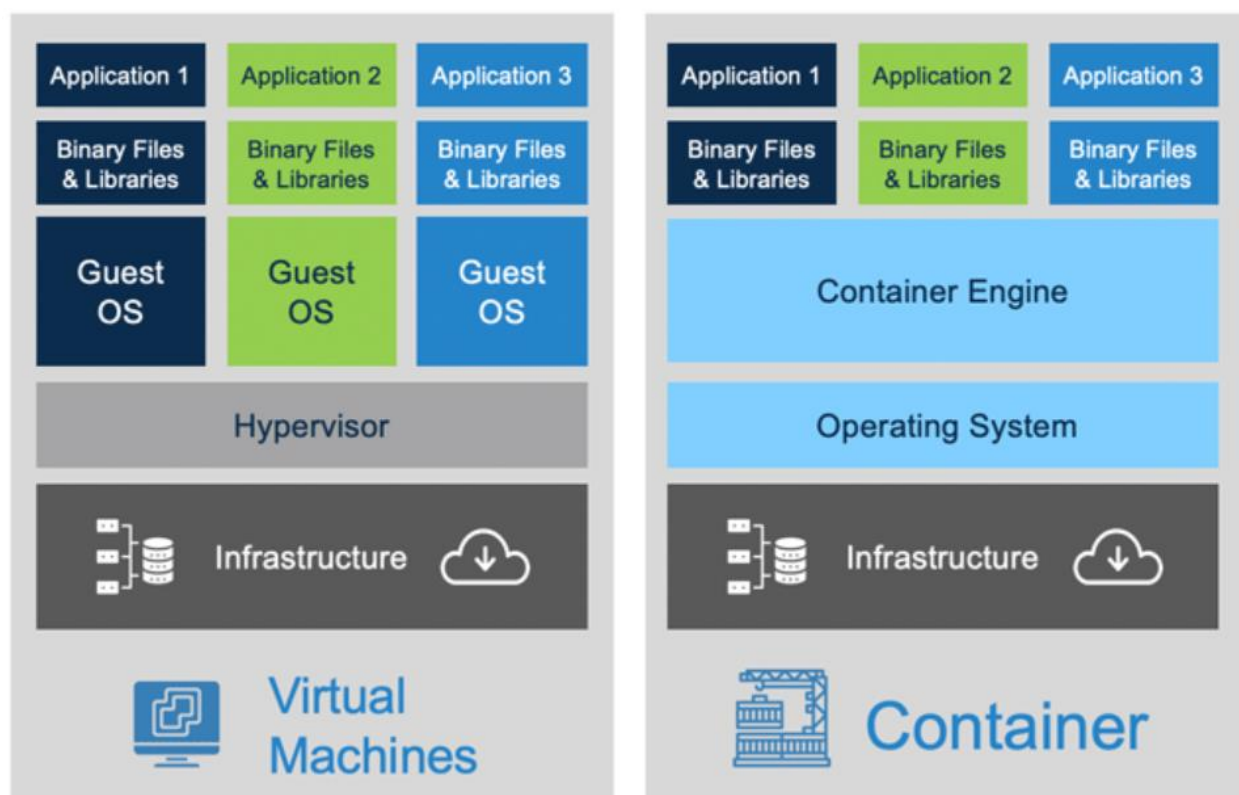
فصل ۷

کانتاینر چیست؟

آخرین نسل معماری پیاده سازی اپلیکیشن که از معماری microservices پیروی میکند. کانتاینر (Container) به صورت استاندارد 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 کنید²

تفاوت بین container and hyper visor



فصل ۸

مزیت های استفاده از داکر

۱ سرعت

۲ قابلیت جا به جا ی اپلیکیشن در انواع سیستم عامل

۳ کارایی

² <https://www.docker.com/resources/what-container>

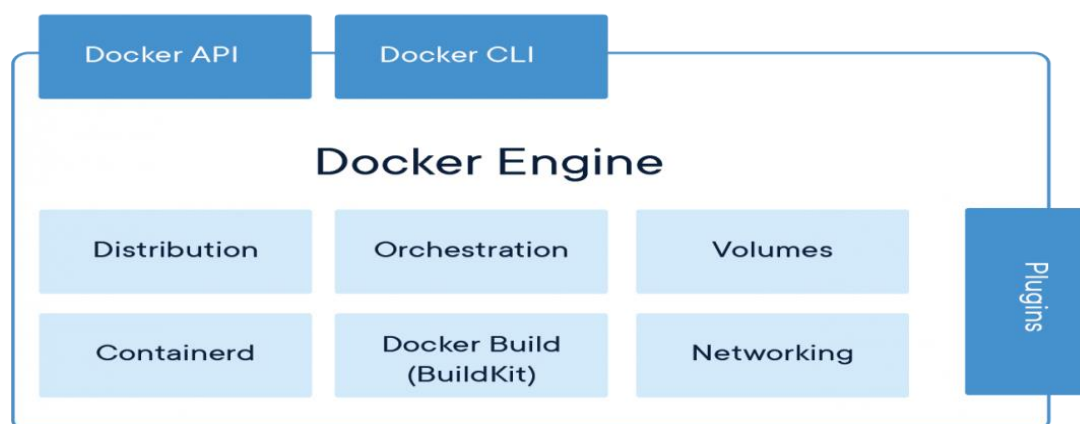
۴ امنیت

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 `docker` `<= command line interface`

برای اطلاعات بیشتر به این لینک مراجعه کنید³

³ <https://docs.docker.com/engine/>

فصل ۱۰

نصب داکر

نصب داکر بر روی سرور CENTOS 7

ابتدا 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>

سپیس

```
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 آخرین کامند باید ببینیم که پروسس داکر در حال اجرا است یا نه.

برای نصب در سیستم عامل ویندوز از این لینک استفاده کنید⁴. 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 آن اضافه میشود

۴ یک network bridge interface در اختیار کانتاینر قرار میگیرد

۵ یک IP به کانتاینر اختصاص داده می شود

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

۷ خروجی اپلیکیشن نمایش داده می شود

به طور مثال یک ایمج را از docker hub دریافت میکنیم `docker pull imagename` و حالا میخواهیم از آن داکر ایمج یک کانتاینر اجرا کنیم خود آن داکر از قبل لایه های و تنظیمات مورد نیاز آن اپلیکیشن را دارد و بعد از آن یک لایه نتورک در اختیار کانتاینر قرار میگیرد و سپس یک ای پی در اختیار آن کانتاینر قرار میگیرد و بسته به شل ی که در آن ایمج 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 آن ارتباط برقرار کرد (در این آموزش از `power shell` ویندوز استفاده شده و داکر را در ویندوز نصب کرده ام)

⁴ <https://www.docker.com/get-started>

```

PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> docker info
Client:
 Debug Mode: false

Server:
 Containers: 13
  Running: 9
  Paused: 0
  Stopped: 4
 Images: 29
 Server Version: 19.03.12
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd
 init version: fec3683
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 4.19.76-linuxkit
 Operating System: Docker Desktop
 OSType: linux
 Architecture: x86_64
 CPUs: 12
 Total Memory: 1.943GiB
 Name: docker-desktop
 ID: UJDI:Z7PQ:XTBL:EGQB:TGGE:NBYO:HB77:2GFO:3ASG:RGUR:TCOM:Z2HZ
 Docker Root Dir: /var/lib/docker
 Debug Mode: true
 File Descriptors: 102
 Goroutines: 128
 System Time: 2020-10-05T12:37:32.941161275Z

```

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 `image` 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 کنید⁵

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 ایمج 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 تنظیم کردن دی ان اس ازاد برای گرفتن ایمج ها استفاده شده بعضی شرکت ها

⁵ <https://docs.docker.com/engine/reference/commandline/info/>

مثل شکن این امر را آسان کرده اند برای این منظور ابتدا ای پی های شکن را در DNS خود تنظیم کنید آموزش تنظیم کردن DNS در ویندوز⁶ و سایت شکن⁷

کامند بعدی docker run است

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

Docker run hello-world

```
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:4cf9c47f86df71d48364001ede3a4fcd85ae80ce02ebad74156906caff5378bc
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

ابتدا ایمج 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

⁶ <https://www.zoomit.ir/2019/7/7/337508/dns-server-change-settings-windows10/>

⁷ <https://shecan.ir/>

```

PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> 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
  --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)
  -d, --detach             Run container in background and
                           print container ID
  --detach-keys string     Override the key sequence for
                           detaching a container
  --device list            Add a host device to the container
  --device-cgroup-rule list
                           Add a rule to the cgroup allowed
                           devices list
  --device-read-bps list   Limit read rate (bytes per second)
                           from a device (default [])
  --device-read-iops list  Limit read rate (IO per second)
                           from a device (default [])
  --device-write-bps list  Limit write rate (bytes per
                           second) to a device (default [])
  --device-write-iops list Limit write rate (IO per second)

```

برای مطالعه بیشتر راجب کامند ⁸run

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.

⁸ <https://docs.docker.com/engine/reference/commandline/run/>

for more examples and ideas, visit:
<https://docs.docker.com/get-started/>

```
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
172.16.107.252:8083/receiver  alpine-new         46d3123c9826       2 days ago         59.8MB
172.16.107.252:8083/sender    alpine-new         9ca6d0f33033       2 days ago         59.8MB
<none>                <none>             74b5dd8417bd       2 days ago         59.8MB
172.16.107.252:8083/fv        dev                aecfb19f7aed       9 days ago         7.37GB
172.16.107.252:8083/receiver  alpine-dev         68e104c6a9f5       9 days ago         59.8MB
172.16.107.252:8083/sender    alpine-dev         423457a67a82       9 days ago         59.8MB
172.16.107.252:8083/receiver  <none>             621d546eea5f       9 days ago         59.8MB
172.16.107.252:8083/sender    <none>             621d546eea5f       9 days ago         59.8MB
python                 alpine             ff6233d0ceb9       11 days ago        42.9MB
172.16.107.252:8083/fv        latest            137dc62205bf       3 weeks ago        6.72GB
rancher/rancher          latest            8524c0d7aecc       4 weeks ago        871MB
prom/blackbox-exporter     master           f422538de024       4 weeks ago        20.9MB
hello-world              latest           bf756fb1ae65       9 months ago       13.3kB

PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
172.16.107.252:8083/receiver  alpine-new         46d3123c9826       2 days ago         59.8MB
172.16.107.252:8083/sender    alpine-new         9ca6d0f33033       2 days ago         59.8MB
<none>                <none>             74b5dd8417bd       2 days ago         59.8MB
172.16.107.252:8083/fv        dev                aecfb19f7aed       9 days ago         7.37GB
172.16.107.252:8083/receiver  alpine-dev         68e104c6a9f5       9 days ago         59.8MB
172.16.107.252:8083/sender    alpine-dev         423457a67a82       9 days ago         59.8MB
172.16.107.252:8083/receiver  <none>             621d546eea5f       9 days ago         59.8MB
172.16.107.252:8083/sender    <none>             621d546eea5f       9 days ago         59.8MB
python                 alpine             ff6233d0ceb9       11 days ago        42.9MB
172.16.107.252:8083/fv        latest            137dc62205bf       3 weeks ago        6.72GB
rancher/rancher          latest            8524c0d7aecc       4 weeks ago        871MB
prom/blackbox-exporter     master           f422538de024       4 weeks ago        20.9MB
hello-world              latest           bf756fb1ae65       9 months ago       13.3kB
```

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 میخوایم صحبت کنیم -i و -t که همراه این کامند اجرا می شوند

Docker run -it image:tag

-i به معنای interactive 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.

```
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> docker run -it centos:latest
Unable to find image 'centos:latest' locally
latest: Pulling from library/centos
3c72a8ed6814: Downloading [==>] 1 5.393MB/74.87MB
```

با اجرای این کامند سرور داکر از داکر هاب centos image را pull کرده و اجرا می کند.


```
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> docker run -it centos:latest
Unable to find image 'centos:latest' locally
latest: Pulling from library/centos
3c72a8ed6814: Pull complete
Digest: sha256:76d24f3ba3317fa945743bb3746fbaf3a0b752f10b10376960de01da70685fbd
Status: Downloaded newer image for centos:latest
[root@1959c96dc8cc /]#
[root@1959c96dc8cc /]#
[root@1959c96dc8cc /]#
```

الان کانتاینر **centos** ما اجرا شده و یک شل در اختیار ما قرار داده و **c96dc8cc1959** 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

```
PS C:\Users\b.hasanbeygi> docker ps
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS
1959c96dc8cc	centos:latest	kind_kare	"/bin/bash"	5 minutes ago	Up 5 minutes

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 در حال 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 اگر 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 دیگر **-q** است

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

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

-qa Docker ps ایدی کلیه کانتاینر های در حال اجرا و متوقف شده یا نگه داشته شده را می توان دید. برا مطالعه بیشتر⁹

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

⁹ <https://docs.docker.com/engine/reference/commandline/ps/>

```
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> docker run -it centos:latest
Unable to find image 'centos:latest' locally
latest: Pulling from library/centos
3c72a8ed6814: Pull complete
Digest: sha256:76d24f3ba3317fa945743bb3746fbaf3a0b752f10b10376960de01da70685fbd
Status: Downloaded newer image for centos:latest
[root@1959c96dc8cc /]#
[root@1959c96dc8cc /]#
[root@1959c96dc8cc /]# exit
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver>
```

در ادامه با flag یا سویچ -d آشنا 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

```
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver>
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver>
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> docker run -itd centos:latest
c4e69a25f91adb97cd2982293b3e30f79686956c1119a9873f2d4bc3770068b0
PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver>
```

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

```
PS PS C:\Users\b.hasanbeygi\Desktop\khatam\receiver>
CONPS C:\Users\b.hasanbeygi\Desktop\khatam\receiver> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
195MES					
c4e69a25f91a	centos:latest	"/bin/bash"	2 minutes ago	Up 2 minutes	

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 -q docker ps

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 آورده و سپس ایدی آن را در جلوی کامند قرار می دهیم به طور مثال:

```
PS C:\Users\h.hasanbeygi\Desktop\khatam\receiver> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
c4e69a25f91a	centos:latest	"/bin/bash"	29 minutes ago	Exited (137) 3 seconds ago
nervous_bose				

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

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

با سوییچ `-f` کانتاینر را ابتدا متوقف و سپس پاک میکنید `Force`

تمرین: یک کانتاینر `centos` را بالا بیارید سپس آن 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

اسم کانتاینر `-f Docker rm`

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 کنید¹⁰

کامند بعدی کامند بسیار کاربردی است

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

¹⁰ <https://docs.docker.com/engine/reference/commandline/rm/>

روش کار به این صورت است:

ابتدا یک فایل متنی را بسازید 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.

```
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd centos
7548e9c4ebdc2c232e506a479b33e9f6dae9306dc5e29e4f84177d712e8000fb
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker ps
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS
7548e9c4ebdc	centos	stupefied_hawking	"/bin/bash"	22 seconds ago	Up 21 seconds

بعد از ساخت کانتاینر حالا از کامند `docker cp` استفاده میکنیم:

```
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker cp .\example.txt 7548e9c4ebdc:/root
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
```

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 آن را چک کنید:

```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker exec -it 7548e9c4ebdc bash
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]# ls -l /root
total 12
-rw----- 1 root root 2366 Aug  9 21:40 anaconda-ks.cfg
-rw-r--r-- 1 root root  435 Aug  9 21:40 anaconda-post.log
-rwxr-xr-x 1 root root    0 Oct  6 18:54 example.txt
-rw----- 1 root root 2026 Aug  9 21:40 original-ks.cfg
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]#

```

می توان در دایرکتوری /root فایل 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

```

PS C:\Users\b.hasanbeygi\Desktop\khatam> docker stop 7548e9c4ebdc
7548e9c4ebdc
PS C:\Users\b.hasanbeygi\Desktop\khatam>

```

که در این کانتاینر متوقف میشه

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

Docker start 7548e9c4ebdc

```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker start 7548e9c4ebdc
7548e9c4ebdc
PS C:\Users\b.hasanbeygi\Desktop\khatam>

```

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]

```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker attach 7548e9c4ebdc
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]# ls -l /root
total 12
-rw----- 1 root root 2366 Aug  9 21:40 anaconda-ks.cfg
-rw-r--r-- 1 root root  435 Aug  9 21:40 anaconda-post.log
-rwxr-xr-x 1 root root   0 Oct  6 18:54 example.txt
-rw----- 1 root root 2026 Aug  9 21:40 original-ks.cfg
[root@7548e9c4ebdc /]# exit
PS C:\Users\b.hasanbeygi\Desktop\khatam>

```

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:

```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker cp .\behnam-mlops.txt 7548e9c4ebdc:/opt
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker start 7548e9c4ebdc
7548e9c4ebdc
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker attach 7548e9c4ebdc
[root@7548e9c4ebdc /]# ls -la /opt/
total 8
drwxr-xr-x 1 root root 4096 Oct  6 19:22 .
drwxr-xr-x 1 root root 4096 Oct  6 19:22 ..
-rwxr-xr-x 1 root root   0 Oct  6 19:21 behnam-mlops.txt
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]#

```

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
```

```
behtnam
```

```
#[/ root@0ee0762da564]
```

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

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

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

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd --name behtnam centos:latest
85e46ec7a4b8360084623abb5860c78700060836f93baf7186969d55108ef1e5
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
85e46ec7a4b8	centos:latest	"/bin/bash"	3 seconds ago	Up 2 seconds
02caa0bb0-22	centos:latest	"/bin/bash"	About an hour ago	Exited (130)

کامند بعدی docker exec است. این کامند برای وصل شدن به کانتینر و ارتباط با آن است. با سویچ های t - که همان tty است و i - که به معنای interactive است می آید. ساختار این کامند به شکل زیر است :

Docker exec -it [containerID] bash

آن bash آخری مربوط به شل موجود در کانتینر است که در زمان ساخت image در آن تعبیه شده میتوان به عنوان مثال از sh نیز استفاده کرد /bin/bash و ¹¹/bin/sh

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker exec -it 7548e9c4ebdc bash
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]# ls -l /root
total 12
-rw----- 1 root root 2366 Aug 9 21:40 anaconda-ks.cfg
-rw-r--r-- 1 root root 435 Aug 9 21:40 anaconda-post.log
-rwxr-xr-x 1 root root 0 Oct 6 18:54 example.txt
-rw----- 1 root root 2026 Aug 9 21:40 original-ks.cfg
[root@7548e9c4ebdc /]#
[root@7548e9c4ebdc /]#
```

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

¹¹ <https://docs.docker.com/engine/reference/commandline/exec/>

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 سوچ - -
hostname

```
Docker run -itd - - hostname beny centos:latest
```

```
Docker exec -it [containerID] bash
```

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd --hostname beny --name behnam centos:latest
87632a838eff1e0096761b244994fa860e979d8b3fcc823db2711a18aacfbb939
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
87632a838eff	centos:latest	"/bin/bash"	5 seconds ago
behn			

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker exec -it 87632a838eff bash
[root@beny /]#
[root@beny /]#
[root@beny /]#
```

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
```



```

root@beny /J# exit
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd --name topproc centos:latest /usr/bin/top -b
d72b0b32629545a87aa5a00a6179914ca73ae55596b5a0a73b56d6d657d22afe
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker attach d72b

top - 21:04:03 up 7 days, 14:32, 0 users, load average: 0.09, 0.09, 0.07
Tasks: 1 total, 1 running, 0 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.5 us, 0.2 sy, 0.0 ni, 99.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1989.6 total, 666.0 free, 609.3 used, 714.3 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used, 1261.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
    1 root        20   0   48924    3812   3280 R   0.0   0.2   0:00.02 top

top - 21:04:03 up 7 days, 14:32, 0 users, load average: 0.09, 0.09, 0.07
Tasks: 1 total, 1 running, 0 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1989.6 total, 666.0 free, 609.3 used, 714.3 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used, 1261.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
    1 root        20   0   48924    3812   3280 R   0.0   0.2   0:00.02 top

```

که اگر مشاهده کنید در زمان اجرایی کانتاینر کامند top طور متناوب اجرا می شود
 سوییچ -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 متغیر به
 آن 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 مسئله را متوجه شوید.

```

FROM ubuntu:18.04
COPY . /app
RUN make /app

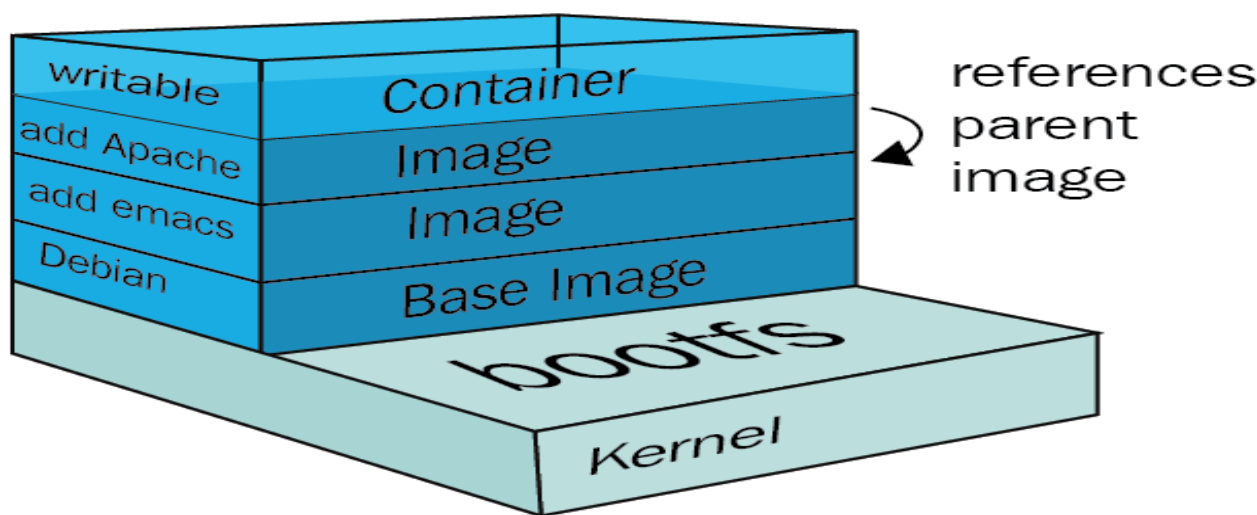
```

در این Dockerfile در لایه اول گفته شده که از ubuntu:18.04 image 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 ایمج 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/WRITE هستند.

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

برای مطالعه بیشتر راجب image و لایه های ان به لینک زیر مراجعه کنید¹²



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.

¹² <https://docs.docker.com/storage/storagedriver/>

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 image از سرور داکر خود می توان از کامند زیر استفاده کنید.

Docker rmi [imageID]

یا

Docker rmi IMAGE NAME

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
172.16.107.252:8083/receiver  alpine-new         46d3123c9826       4 days ago
172.16.107.252:8083/sender    alpine-new         9ca6d0f33033       4 days ago
<none>                  <none>             74b5dd8417bd       4 days ago
172.16.107.252:8083/fv        dev                aecfb19f7aed       10 days ago
172.16.107.252:8083/receiver  alpine-dev         68e104c6a9f5       10 days ago
172.16.107.252:8083/sender    alpine-dev         423457a67a82       10 days ago
172.16.107.252:8083/receiver  <none>             621d546eea5f       10 days ago
172.16.107.252:8083/sender    <none>             621d546eea5f       10 days ago
python                  alpine             ff6233d0ceb9       12 days ago
172.16.107.252:8083/fv        latest            137dc62205bf       3 weeks ago
rancher/rancher          latest            8524c0d7aecc       4 weeks ago
prom/blackbox-exporter     master           f422538de024       5 weeks ago
centos                    latest            0d120b6ccaa8       8 weeks ago
hello-world               latest            bf756fb1ae65       9 months ago
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker rmi -f hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:4cf9c47f86df71d48364001ede3a4fcd85ae80ce02ebad74156906caf
Deleted: sha256:bf756fb1ae65adf866bd8c456593cd24beb6a0a061dedf42b26a993176745f6b
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
```

اگر در زمان حذف image برای شما error اینکه از image یک کانتاینر بالا آمده و در حال استفاده است امد میتوانید با سوییچ -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 میخواهیم 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 میخواهید 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`

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd centos:latest
7eeba0e633d2fcb6c0be79ba94301209e75024c2d1aa0121daf362b5b72a87e2
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker exec -it 7ee bash
[root@7eeba0e633d2 /]# touch behnam /opt
[root@7eeba0e633d2 /]# exit
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker commit 7ee centos:behnam
sha256:7caca70cc67bf062e64f3861087e5f476e03ee803f7b31de66bf15480c7a552c
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
centos               behnam             7caca70cc67b       13 seco
172.16.102.252:8083/receiver  alpine-new        46d3123c9826       4 days
```

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

کامند بعدی `docker save` است

با کمک این کامند می‌توان 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 [imageID]`

```
ktop\khatam>
ktop\khatam>
ktop\khatam>
ktop\khatam> docker save -o centos-behnam.tar.gz 7ca
ktop\khatam> ls
```

sanbeygi\Desktop\khatam

Time	Length	Name
06 PM		FU
09 PM		fu-dev
12 AM		receiver
02 PM		sender
51 PM		sender - test
51 PM	0	behnam-mlops.txt
17 AM	222371840	centos-behnam.tar.gz
24 PM	0	example.txt
27 AM	2339778560	fu-receiver.tar.gz
33 AM	2339781632	fu-sender.tar.gz
10 AM	6765634048	fu.tar.gz
10 PM	2454447	processor.log.json

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.

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker load -i .\centos-behnam.tar.gz
Loaded image ID: sha256:7caca70cc67bf062e64f3861087e5f476e03ee803f7b31de66bf15480c7a552c
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos               behnam             7caca70cc67b       10 minutes ago     215
122.16.102.252:8083/receiver alpine-new         46d3123c9826       4 days ago         59.
```

برای مطالعه بیش تر به این لینک ها مراجعه کنید.¹⁴¹³

پروژه :

یک سرور `centos` را بسازید در آن پکیج `vim` را نصب کنید. سپس از آن کانتاینر یک `commit` بگیر به اسم `centos:vim` و بعد از آن یک کانتاینر از آن `image` که ساخته اید `UP` کنید.

حل: `docker run -itd centos`

بعد : `docker exec -it [containerID] bash`

بعد `yum install -y vim`

بعد `docker commit [containerID] centos:vim`

بعد `docker run -itd centos:vim`

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

`Docker pull mariadb:latest`

`Docker run -itd -e MYSQL_ROOT_PASSWORD1234 mariadb:latest`

¹³ <https://docs.docker.com/engine/reference/commandline/save/>

¹⁴ <https://docs.docker.com/engine/reference/commandline/load/>

حتما باید پارامتر MYSQL_ROOT_PASSWORD به همین شکل به کانتاینری که میخواهیم بسازیم پاس داده شود و پسورد داده شود که من پسورد ۱۲۳۴ را انتخاب کرده ام.

```
Using default tag: latest
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker pull mariadb
Using default tag: latest
latest: Pulling from library/mariadb
d72e567cc804: Downloading 1.768MB/28.56MB
0f3630e5ff08: Download complete
b6a83d81d1f4: Download complete
4bf2111ecf0e: Download complete
9572d64978a0: Downloading 1.146MB/5.489MB
bcc9953bffb3: Waiting
de429570dda5: Waiting
3652bc6ea9f9: Waiting
6e4bf87041c8: Waiting
96e489d6af27: Waiting
baaf018282fc: Waiting
1cca1ca0b2da: Waiting
```

سپس یک کانتاینر از mariadb image ایجاد میکنیم:

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd -e MYSQL_ROOT_PASSWORD=1234 mariadb:latest
b45aedca9c27edadb014ef895787ce99997715454dc7125b4b465f245969067f3
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
b45aedca9c27	mariadb:latest	"docker-entrypoint.s?"	4 seconds ago	Up 3 sec

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

```

PS C:\Users\b.hasanbeygi\Desktop\khatam> docker exec -it b45 bash
root@b45aedca9c27:/# mysql -uroot -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.5.6-MariaDB-1:10.5.6+maria~focal mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database behnam;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| behnam   |
| information_schema |
| mysql    |
| performance_schema |
+-----+
4 rows in set (0.000 sec)

MariaDB [(none)]>

```

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.

```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker ps
CONTAINER ID        IMAGE               COMMAND
b45aedca9c27       mariadb:latest     "docker-
74cdd90c2fdd       mariadb:latest     "docker-
asser              centos:latest      "/bin/ba
tern              centos:latest      "/bin/ba
87632a838eff       172.16.107.252:8083/receiver:alpine-new "python
b9867a19ce4d       172.16.107.252:8083/receiver:alpine-new "python
8ffecb40bb06       172.16.107.252:8083/receiver:alpine-new "python
g                 172.16.107.252:8083/receiver:alpine-new "python
a8550bd7d315       172.16.107.252:8083/sender:alpine-new   "python
tonelli          172.16.107.252:8083/sender:alpine-new   "python
0ec1c336ebf5     172.16.107.252:8083/sender:alpine-new   "python
ach              172.16.107.252:8083/sender:alpine-new   "python
1e304de973af     172.16.107.252:8083/sender:alpine-new   "python
fe8a62516151     172.16.107.252:8083/sender:alpine-new   "python
80c6ba05be87     172.16.107.252:8083/sender:alpine-new   "python
nd              172.16.107.252:8083/sender:alpine-new   "python
4ca9da5d5156     prom/blackbox-exporter:master           "/bin/b
f53053fc6f35
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker kill b45aedca9c27

```

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

کامند docker import and export :

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

Docker export [containerID] > MYCONTAINER.TAR

و در سرور داکر دیگری :

Docker import MYCONTAINER.TAR

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker export b45aedca9c27 > mymariadb.tar
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker import mymariadb.tar
```

کامند docker restart :

با زدن این کامند کانتاینر restart می شود.

Docker restart [containerID]

فصل ۱۴

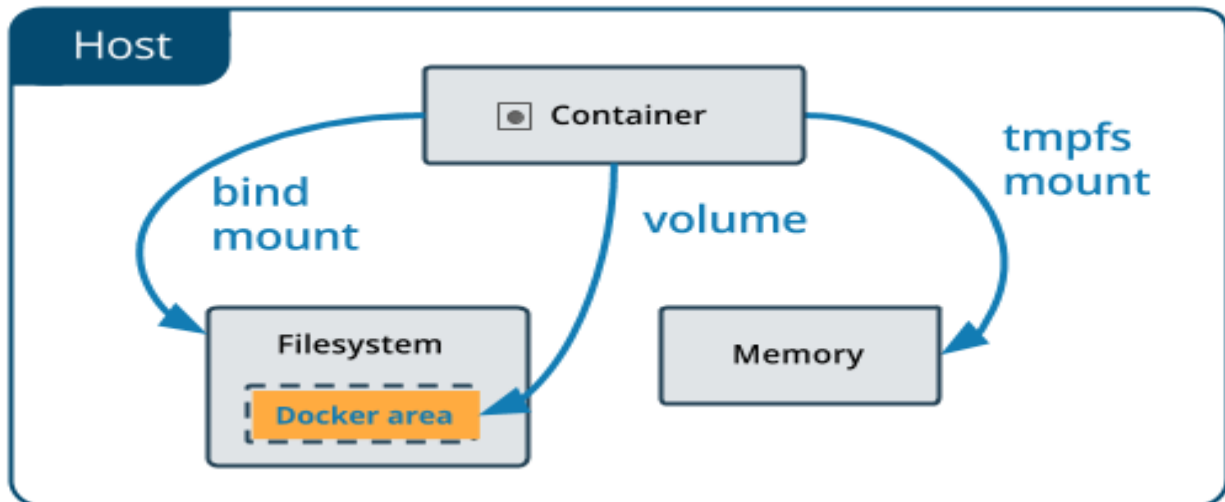
Docker volumes

زمانی که شما یک کانتاینر را اجرا می کنید خود ان کانتاینر نمی تواند اطلاعات شما را ذخیره (persist) کند یعنی به صورت دائمی 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

¹⁵ <https://docs.docker.com/engine/reference/commandline/kill/>

مزایای استفاده از volume:

- ۱ از volume می توان به سادگی یک آپ 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 volume¹⁶

با flag یا سویچ -v در زمان اجرای docker run میتوان به کانتاینر فهماند که 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

```
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker volume create --name behnam
behnam
```

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

¹⁶ <https://docs.docker.com/storage/volumes/>

```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker volume ls
DRIVER          VOLUME NAME
local           0e9772aa7d2749435460f799b25dba72bd02253f6594fe13fe6c7dd6bab1d74c
local           37c7dae33298628b3b94894ec2b372c009c4fb8bd2a7f0954d5d5fcab696f07
local           70d4a98d345210383a6bd6b4a983be2a2b39867304a023463c56fd2a57fa168a
local           92a7969b6bcc0e9098057a83f58aeb4375854921345b73c14677f205b4a901f0
local           234e181af535bad4ef4fa47af5dba68bb7f7d2ec44be5a8036c728db53f412e3
local           49090d404dadd21a17fd08b18ba26e8a74c6cafbc30254927c89a416c824893d
local           bd5cfa7db74f98f3be7c1c99c8d1f9fec280701ad4cd338ee17447e7a75e184c
local           behnam
PS C:\Users\b.hasanbeygi\Desktop\khatam>

```

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 تغییری در /root کانتاینر اتفاق بیافتد در volume بهنام همان اتفاق خواهد افتاد:

```

local           behnam
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd -v behnam:/root --name behnam-volume centos:latest
a9248062604b001bdcdb32b829e28eb22aeefc249eb17a208c1c7afbe34469f2
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS
a9248062604b   centos:latest  "/bin/bash"             2 seconds ago   Up
b4f5e0e0a9e27   centos:latest  "docker-entrypoint.sh"  2 hours ago     Up

```

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

```

bash: clear: command not found
[root@a9248062604b ~]# exit
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker exec -it a9248062604b bash
[root@a9248062604b /]# cd /root/
[root@a9248062604b ~]# ls
anaconda-ks.cfg  anaconda-post.log  original-ks.cfg
[root@a9248062604b ~]# touch behnam.txt
[root@a9248062604b ~]# echo docker is very easy > behnam.txt
[root@a9248062604b ~]# cat behnam.txt
docker is very easy
[root@a9248062604b ~]# ls
bash: $'\331\205ls': command not found
[root@a9248062604b ~]# ls
anaconda-ks.cfg  anaconda-post.log  behnam.txt  original-ks.cfg
[root@a9248062604b ~]#
[root@a9248062604b ~]#

```

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 مسئله 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.

```
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -it --rm --privileged --pid=host justincormack/nsenter1
Unable to find image 'justincormack/nsenter1:latest' locally
latest: Pulling from justincormack/nsenter1
5bc638ae6f98: Pull complete
Digest: sha256:5af0be5e42ebd55eea2c593e4622f810065c3f45bb805eaacf43f08f3d06ffd8
Status: Downloaded newer image for justincormack/nsenter1:latest
/ # ls
EFI          boot         dev          home         lib          mnt          proc         run          srv          top
bin          containers   etc          init         media        opt          root        sbin        sys        usr
/ #
/ #
/ #
/ # cd /var/lib/
/var/lib # cd docker/
/var/lib/docker # ls
builder      buildkit     containerd   containers   image        network      overlay2     plugins      runtimes     swarm
/var/lib/docker #
/var/lib/docker #
/var/lib/docker # cd volumes/
/var/lib/docker/volumes # ls
be97722aa7d2249435460f799b25dba72bd02253f6594fe13fe6c7dd6bah1d74c  92a7969b6bcc0e9098057a83f58aeb4375854921345b73c146
234e181af535bad4ef4fa47af5dba68bb7f7d2ec44be5a8036c728db53f412e3    bd5cfa7db74f98f3be7c1c99c8d1f9fec280701ad4cd338ee1
37c7dae33298628b3b94894ec2b372c009c4fb8bd2a7f0954d5d5fcab696f07    behnam
49090d404dadd21a17fd00b18ba26e8a74c6cafbcb30254927c89a416c824893d    metadata.db
70d4a98d345210383a6bd6b4a983be2a2b39867304a023463c56fd2a57fa168a
/var/lib/docker/volumes # cd behnam/
/var/lib/docker/volumes/behnam # ls
_data
/var/lib/docker/volumes/behnam #
/var/lib/docker/volumes/behnam #
/var/lib/docker/volumes/behnam # cd _data/
/var/lib/docker/volumes/behnam/_data # ls
anaconda-ks.cfg  anaconda-post.log  behnam.txt        original-ks.cfg
/var/lib/docker/volumes/behnam/_data #
/var/lib/docker/volumes/behnam/_data #
```

با کماند `docker inspect` میتوانید کلیه مشخصات کانتینری که ایجاد کرده اید را ببینید.

`Docker inspect [containerID]`

```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED
f53053fc6f35       prom/blackbox-exporter:master          "/bin/blackbox_expor?"  12 days ag
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker inspect f53053fc6f35
[
  {
    "Id": "f53053fc6f35bacee9b77628b399b12e634853a606f33e9bc828e9481d9b6d5",
    "Created": "2020-09-28T19:30:04.127087804Z",
    "Path": "/bin/blackbox_exporter",
    "Args": [
      "--config.file=/etc/blackbox_exporter/config.yml"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 2703,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2020-10-11T17:52:41.6379075Z",
      "FinishedAt": "2020-10-11T17:46:08.3591892Z"
    },
    "Image": "sha256:f422538de024ca742f1c3064fa834de48c10a0b3097beabaf39c44e7d20786",
    "ResolvConfPath": "/var/lib/docker/containers/f53053fc6f35bacee9b77628b399b12e63",
    "HostnamePath": "/var/lib/docker/containers/f53053fc6f35bacee9b77628b399b12e63",
    "HostsPath": "/var/lib/docker/containers/f53053fc6f35bacee9b77628b399b12e63485",
    "LogPath": "/var/lib/docker/containers/f53053fc6f35bacee9b77628b399b12e634853a",
    "Name": "/angry_blackburn",
  }
]

```










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 آن 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.یعنی اینکه در قسمت اول دایرکتوری سیستم عامل و در قسمت بعدی کانتاینتر HOST:CONTAINER

به طور مثال شما یک elastic search برای لاگ خوانی اپلیکیشن خود دارید به 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 آن استفاده کنید به طور مثال:

 .pip	9/10/2020 11:55 PM	File folder
 config	9/10/2020 11:53 PM	File folder
 io_module	9/10/2020 11:53 PM	File folder
 logs	9/10/2020 11:53 PM	File folder
 utils	9/20/2020 12:36 AM	File folder
 conf	10/3/2020 2:02 PM	Python Source File
 Dockerfile	10/3/2020 3:09 PM	File
 requirements	9/23/2020 12:32 PM	Text Document
 sender_app	10/3/2020 3:15 PM	Python Source File

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 آمده 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 آورد

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

```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
c83d03316c7b        bridge             bridge             local
6ba1e1566342        host               host               local
d7d12e181ebc        none              null               local
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker network inspect c83d03316c7b
[
  {
    "Name": "bridge",
    "Id": "c83d03316c7b435ee422ead54c884b71293702377e89bf840608f16add60daaf",
    "Created": "2020-10-11T17:46:09.8750627Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "f53053fc6f35bacebea77628b399b12e634853a606f33e9bc828e9481d9b6d5": {
        "Name": "angry_blackburn",
        "EndpointID": "a0820406c5d06753768afd64afdaa4967ad80f3c089cb99e930de101f533b4fa",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      }
    }
  }
]

```

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

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

Network bridge 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 آن 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 forwarding 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` جزئیات کانتاینر را نگاه کنید
اگر به قسمت `port` نگاه کنید پورت ۳۳۰۶ که پورت مخصوص سرور `mysql` است
را می توانید ببینید.

```
{
  "Cmd": [
    "mysqld"
  ],
  "Image": "mariadb:latest",
  "Volumes": {
    "/var/lib/mysql": {}
  },
  "WorkingDir": "",
  "Entrypoint": [
    "docker-entrypoint.sh"
  ],
  "OnBuild": null,
  "Labels": {}
},
"NetworkSettings": {
  "Bridge": "",
  "SandboxID": "fe5bffff688667aef8682549da4d8346516adc8e1",
  "HairpinMode": false,
  "LinkLocalIPv6Address": "",
  "LinkLocalIPv6PrefixLen": 0,
  "Ports": {
    "3306/tcp": null
  },
  "SandboxKey": "/var/run/docker/netns/fe5bffff688666"
```

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 اصطلاحاً `port forwarding` می گویند.

برای این منظور از `flag` یا سوییچ `-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.

```
C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd -p 3306:3306 -e MYSQL_ROOT_PASSWORD=1234 mariadb:latest
5c0271ce4710d31c77fe1acd704d0249018c2bf7589c188e202df15d0003cd15
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5c0271ce4710	mariadb:latest	"docker-entrypoint.s?"	3 seconds ago	Up 2 seconds	0.0.0.0:3306->3306/tcp	amazing_wiles
28d2a6ce625c	mariadb:latest	"docker-entrypoint.s?"	14 minutes ago	Up 14 minutes	3306/tcp	eager_keldush

اگر در قسمت پورت نگاه کنید به این صورت است 3306 0.0.0.0:3306 □ 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 اگر پورت 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 یک اند خود اختصاص دهید به طور مثال ۳۰۰۲ و به فرانت خود بگویید که یک اند در پورت ۳۰۰۲ فعالیت میکند به حالت معمول ما به تعداد ۶۵۵۳۵ پورت داریم که می توانیم از آن ها استفاده کنیم.

اگر میخواهیم 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
```

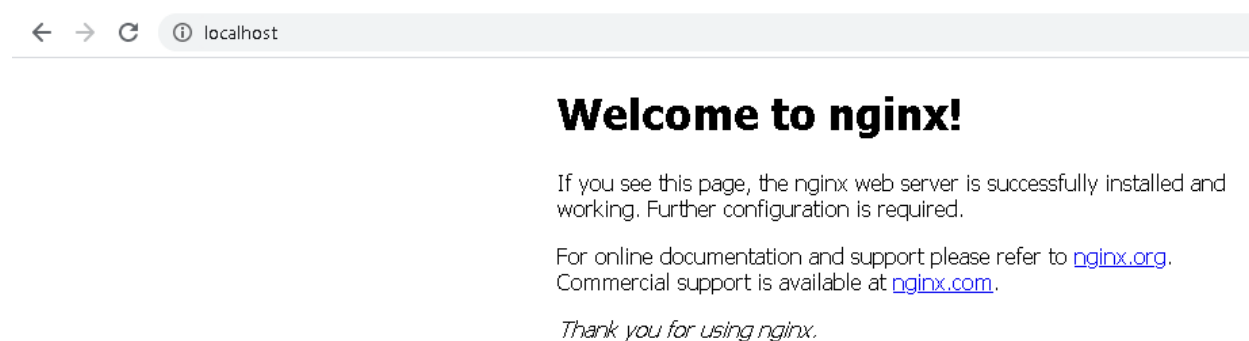
مثال: nginx یک وب سرور است. ابتدا آن را از هاب داکر در سیستم خود pull کنید سپس با زدن این کامند آن را اجرا کنید


```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd -p 80:80 nginx
d5ffe21b49cfc26e08099113ee1e51d0b1884c83133d49e760f837b0d0c54476
PS C:\Users\b.hasanbeygi\Desktop\khatam>

```

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



خب حالا بیاید `port` هاست را تغییر دهیم.

```

PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam> docker run -itd -p 8080:80 nginx
8b7d0dac8e6f29f081c1de8058d91c82396a427795cb6801fe7ab29072fff3dc
PS C:\Users\b.hasanbeygi\Desktop\khatam>
PS C:\Users\b.hasanbeygi\Desktop\khatam>

```

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

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

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

فصل ۱۶

ساختن Docker image

در hub.docker.com بی شمار image وجود دارد آن هم به دلیل اجتماع قوی از برنامه نویس ها و سیستم ادمن 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 آن پیروی کرد و در بعضی از اپلیکیشن ها به دلیل متفاوت بودن نوع نصب وابستگی های اپلیکیشن و یا نوع اجرا شدن آن 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 آن وجود داشته باشد.

★	.pip	9/10/2020 11:55 PM	File folder
★	config	9/10/2020 11:53 PM	File folder
★	io_module	9/10/2020 11:53 PM	File folder
★	logs	9/10/2020 11:53 PM	File folder
★	utils	9/20/2020 12:36 AM	File folder
	conf	10/3/2020 2:02 PM	Python Source File
	Dockerfile	10/3/2020 3:09 PM	File
	requirements	9/23/2020 12:32 PM	Text Document
	sender_app	10/3/2020 3:15 PM	Python Source File

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

Dockerfile به این شکل است:

```

Dockerfile X
C: > Users > b.hasanbeygi > Desktop > khatam > sender > Dockerfile > FROM
1 FROM python:alpine
2 WORKDIR /sender
3 COPY requirements.txt .
4 ADD ../.pip/pip.conf /etc/pip.conf
5 RUN pip3 install -r requirements.txt
6 COPY . .
7 CMD [ "python","./sender_app.py" ]

```

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 اپلیکیشنی برای ساخته شدن نیاز به یک base image یا ایمج 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 ایمج 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 آن 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 آماده 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 ایمج را بساز و به مرحله بعد برو

CMD : همان کامند است . اگر این قسمت را در داکر ایمج 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 اپلیکیشن python است به این شکل آن 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 آن به صورت default به کانتاینر دستور داد که هر زمان UP شد کامند 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 بیشتر¹⁷

بعد از اینکه Dockerfile را ایجاد کرده اید حالا باید آن را بسازید با زدن این کامند 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 .

¹⁷ <https://docs.docker.com/engine/reference/builder/>

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

```
PS C:\Users\b.hasanbeygi\Desktop\khatam\sender> ls

Directory: C:\Users\b.hasanbeygi\Desktop\khatam\sender

Mode                LastWriteTime         Length Name
----                -
d-----          9/10/2020   11:55 PM             .pip
d-----          9/10/2020   11:53 PM            config
d-----          9/10/2020   11:53 PM          io_module
d-----          9/10/2020   11:53 PM             logs
d-----          9/20/2020   12:36 AM            utils
-a-----         10/3/2020    2:02 PM              0 conf.py
-a-----         10/3/2020    3:09 PM            179 Dockerfile
-a-----          9/23/2020   12:32 PM            84 requirements.txt
-a-----         10/3/2020    3:15 PM           6525 sender_app.py

PS C:\Users\b.hasanbeygi\Desktop\khatam\sender> docker build -t 172.16.107.252:8083/sender:alpine .
Sending build context to Docker daemon 61.44kB
Step 1/7 : FROM python:alpine
--> ff6233d0ceb9
Step 2/7 : WORKDIR /sender
--> Using cache
--> 41b1cc2993fd
Step 3/7 : COPY requirements.txt .
--> Using cache
--> 284f8360399a
Step 4/7 : ADD ./pip/pip.conf /etc/pip.conf
--> Using cache
--> 495dde646723
Step 5/7 : RUN pip3 install -r requirements.txt
--> Using cache
--> 3c83a5c29e40
Step 6/7 : COPY . .
--> 0929f55b814e
Step 7/7 : CMD [ "python", "./sender_app.py" ]
--> Running in 2772d23e1a99
Removing intermediate container 2772d23e1a99
--> c021b675cf37
Successfully built c021b675cf37
Successfully tagged 172.16.107.252:8083/sender:alpine
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and
/ permissions. It is recommended to double check and reset permissions for sensitive files and directories.
PS C:\Users\b.hasanbeygi\Desktop\khatam\sender>
```

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 ایمج یکم زمان ساخت آن طول بکشد ولی در زمان های بعدی ساخت به دلیل اینکه cache می شود سرعت ساخت ایمج خیلی زیاد است.

فصل ۱۷

Docker compose

Compose به معنای قطب نماست. و واقعا قطب نماست 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

¹⁸ <https://docs.docker.com/compose/install/>

نصب docker compose در لینوکس:

آخرین ورژن (در زمان نوشتن این آموزش) 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 کامند به داکر کامپوز `permission` اجرای بدهید.

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

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

`Docker-compose --version`

که باید این خروجی به شما نمایش داده شود:

```
PS C:\Users\b.hasanbeygi>  
PS C:\Users\b.hasanbeygi> docker-compose --version  
docker-compose version 1.27.4, build 40524192  
PS C:\Users\b.hasanbeygi>
```

داکر را اگر در ویندوز استفاده می کنید داکر کامپوز همراه با `docker desktop` نصب شده و نیازی به نصب آن 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 کافیه 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 آن 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.

```

docker-compose.yml > Cloud Code > {} networks > {} gitlab > external
1  version: '3.5'
2
3  services:
4
5      gitlab:
6
7          image: gitlab/gitlab-ce:latest
8
9          hostname: 172.16.107.252
10
11         restart: unless-stopped
12
13         environment:
14
15             GITLAB_OMNIBUS_CONFIG: |
16                 gitlab_rails['gitlab_shell_ssh_port'] = 8822
17
18
19                 registry_external_url 'http://localhost:8083'
20
21                 registry['enable'] = true

```

Ln 69, Col 31 Spaces: 2 UTF-8 CRLF

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 آن 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 متغیر های که نیاز است برای ارتباط با سرور gitlab را می نویسیم

ادامه:

```
23 docker-compose.yml
24
25
26     ports:
27
28     - "80:80"
29
30     - "8822:22"
31
32     volumes:
33
34     - ./config/gitlab:/etc/gitlab
35
36     - ./data/gitlab:/var/opt/gitlab
37
38     - ./logs:/var/log/gitlab
39
40     networks:
41
42     - gitlab
```

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 آخر `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:


```
gitlab-runner:

  image: gitlab/gitlab-runner:latest

  restart: unless-stopped

  depends_on:

    - gitlab

  volumes:

    - ./config/gitlab-runner:/etc/gitlab-runner

    - /var/run/docker.sock:/var/run/docker.sock

  networks:

    - gitlab
```

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.

```

58     - /var/run/docker.sock:/var/run/docker.sock
59
60     networks:
61
62     - gitlab
63
64
65
66     networks:
67
68         gitlab:
69             external: true

```

در آخر باید شبکه‌ی که می‌خواهیم در سرور داکر اجرا شود تعریف کنیم که معنی `external : true` این است که از شبکه‌ی 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 ایمج 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 می‌توان از کامند زیر استفاده کرد

Docker-compose ps

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

Docker-compose down سرویس ها را پایین می‌آورد

Docker-compose restart gitlab سرور گیت لب شما را ریستارت می‌کند

Docker-compose logs -f gitlab لاگ های که سرور `gitlab` شما می‌سازد را به شما نشان می‌دهد.

برای مطالعه بیش تر راجب ¹⁹ docker-compose

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 آموزش 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

```
> OPEN EDITO... 1 UNSAVED
GITLAB
docker-compose.yml

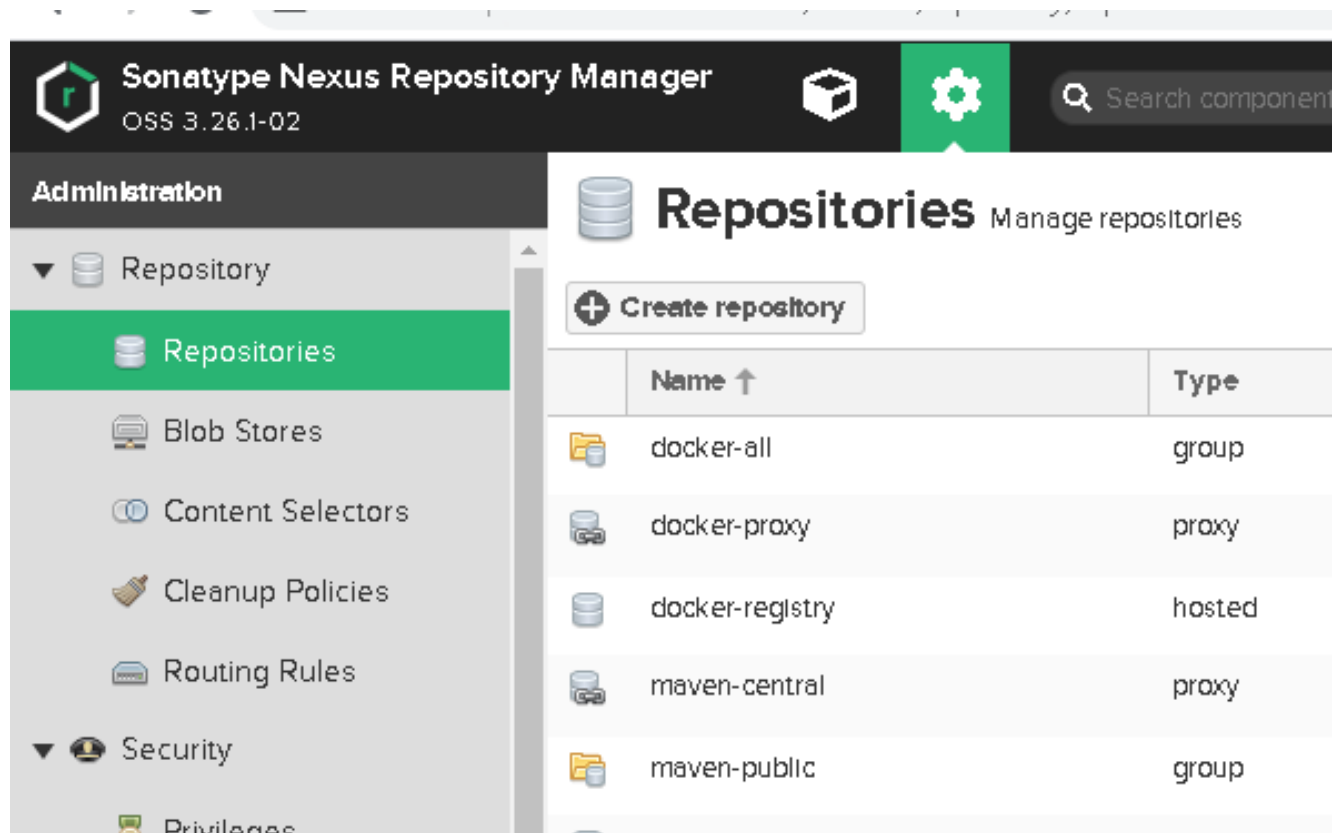
1  version: "2"
2
3  services:
4    nexus:
5      image: sonatype/nexus3
6      environment:
7        - HTTP_PROXY=http://172.16.107.134:3128/
8        - HTTPS_PROXY=http://172.16.107.134:3128/
9      volumes:
10     - "nexus-data:/nexus-data"
11     ports:
12       - "8081:8081"
13       - "8082:8082"
14       - "8083:8083"
15       - "8084:8084"
16     restart: always
17
18   volumes:
19     nexus-data: {}
20
```

¹⁹ <https://docs.docker.com/compose/>

قسمت HTTP_PROXY را نادیده بگیرید به دلیل اینکه سرور های من از طریق پراکسی اینترنت می گیرند این 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 میخوانیم nexus را در آن 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.



	Name ↑	Type
	docker-all	group
	docker-proxy	proxy
	docker-registry	hosted
	maven-central	proxy
	maven-public	group

در قسمت blob stores ابتدا باید registry خود را بسازید

03/03/2017 02:10

Administration

- Repository
 - Repositories
 - Blob Stores**
 - Content Selectors
 - Cleanup Policies
 - Routing Rules
- Security
 - Privileges
 - Roles
 - Users
 - Anonymous Access
 - LDAP
 - Realms
 - SSL Certificates
- IQ Server

Blob Stores / docker-registry

This blob store is in use by 1 repository and 0 other blob stores and cannot be deleted

Updating blob store configuration will cause it to be temporarily unavailable for a short when changing values.

Delete blob store

Settings

Type:
File

Name:
docker-registry

State:
Started

☐ Enable Soft Quota

Path:
docker-registry

Save Discard

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

Administration

Repository

Repositories

Blob Stores

Content Selectors

Cleanup Policies

Routing Rules

Security

Privileges

Roles

Users

Anonymous Access

LDAP

Realms

SSL Certificates

IQ Server

Repositories / docker-registry

Delete repository

Rebuild Index

Settings

Name:

docker-registry

Format:

docker

Type:

hosted

URL:

http://172.16.107.252:8081/repository/docker-registry/

Online:

☒ If checked, the repository accepts incoming requests

Repository Connectors

Connectors allow Docker clients to connect directly to hosted registries, but are not always required. Consult our [documentation](#) for which connector is appropriate for your use case. For information on scaling the repositories see our [scaling documentation](#).

HTTP:

Create an HTTP connector at specified port. Normally used if the server is behind a secure proxy.

☒ 8083

HTTPS:

Create an HTTPS connector at specified port. Normally used if the server is configured for https.

☐

Allow anonymous docker pull:

ادامه تنظیمات

HTTPS:

Create an HTTPS connector at specified port. Normally used if the server is configured for https.

☐

Allow anonymous docker pull:

☐ Allow anonymous docker pull (Docker Bearer Token Realm required)

Docker Registry API Support

Enable Docker V1 API:

☒ Allow clients to use the V1 API to interact with this repository

Storage

Blob store:

docker-registry

Strict Content Type Validation:

☒ Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

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

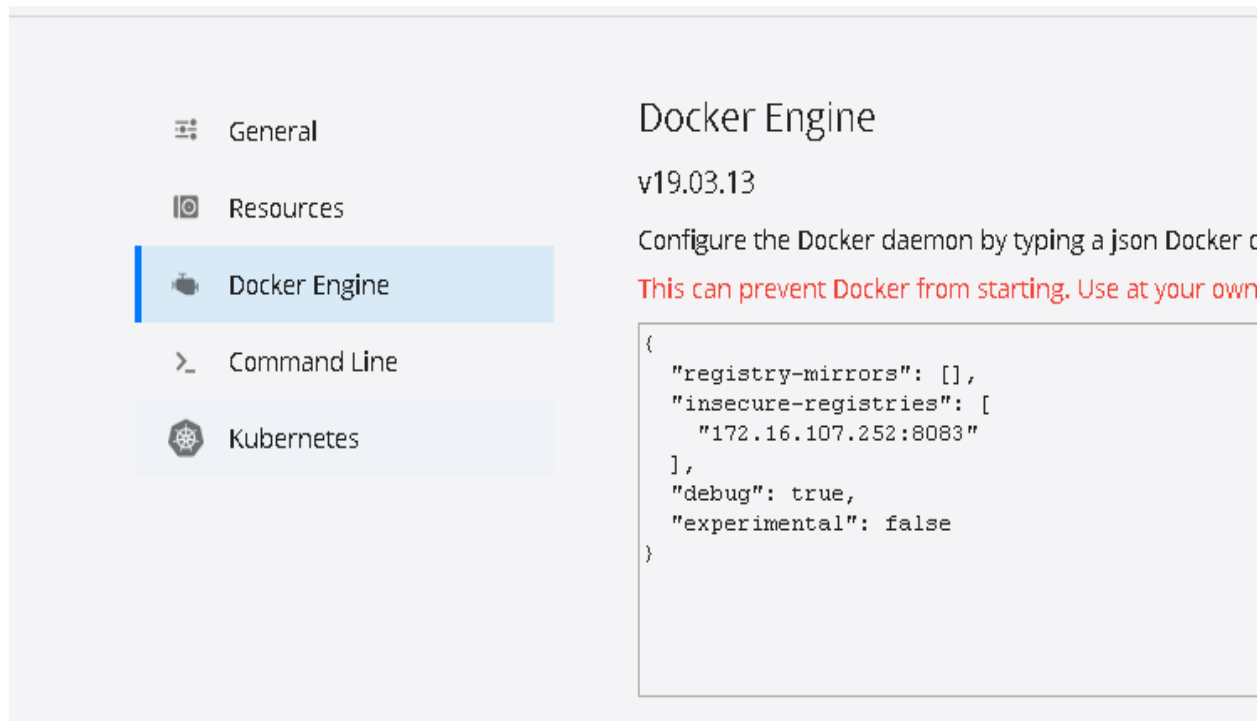
به دلیل اینکه registry را که اجرا کردید از طرف داکر غیر امن خوانده می شود شما باید به سرور داکر خود تنظیماتی بدهید که مشکل insecure-registry را برطرف کنید

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 آن کامند `Docker log ip:8083` را زده و با یوزر نیم و پسورد 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 کنید به همان شکل قبلی لاگین کرده

برای اینکه ایمج 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

ایمج **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 :

```
PS C:\Users\b.hasanbeygi>
PS C:\Users\b.hasanbeygi>
PS C:\Users\b.hasanbeygi> docker tag nginx:latest 172.16.107.252:8083/nginx:latest
PS C:\Users\b.hasanbeygi> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
172.16.107.252:8083/sender	alpine	c021b675cf37	2 hours ago	59.8MB
mariadb	latest	4e7e0dfceed8	4 days ago	406MB
nginx	latest	992e3b7be046	6 days ago	133MB
172.16.107.252:8083/nginx	latest	992e3b7be046	6 days ago	133MB
172.16.107.252:8083/receiver	alpine-new	46d3123c9826	8 days ago	59.8MB

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

Docker push IP:8083/nginx:latest


```

PS C:\Users\b.hasanbeygi>
PS C:\Users\b.hasanbeygi>
PS C:\Users\b.hasanbeygi> docker push 172.16.107.252:8083/nginx
The push refers to repository [172.16.107.252:8083/nginx]
8032102adebe: Pushed
8eb80f066de2: Pushed
7230cfe05cc1: Pushed
822ae9fef1d8: Pushed
07cab4339852: Layer already exists
latest: digest: sha256:416d511ffa63777489af47f250b70d1570e428b67666567085f2bece3571ad83 size: 1362
PS C:\Users\b.hasanbeygi>

```

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



و با کامند زیر می توانید این ایمج را در هر سروری که insecure-registries را برای آن 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

[Linkedin.com/behnam-hasanbeygi](https://www.linkedin.com/company/behnam-hasanbeygi)