Troubleshooting in Docker

Troubleshooting frequently-occurring issues in Docker

We'll cover the following

If you have followed all the instructions in every chapter, you will definitely have a running app now. However, you should keep in mind that sometimes, not going by the book will make you learn better. So, if you have been giving it a try and trying out different things so far, you may face some issues and that is okay.

In this lesson, we will see the troubleshooting of frequently-occurring issues.

> For system or compatibility issues, check out Docker system support.

Frequently-faced issues

Here, we will see the errors and associated troubleshooting.

Wrong build path

```
$ docker build .
unable to prepare context: unable to evaluate symlinks in Dockerfile path: lst
at /Users/venkateshachintalwar/Documents/Online_Projects/Dockerfile: no
such file or directory
```

If you are accidentally not in the directory where Dockerfile is located, Docker will throw an error. `cd` into the directory where Dockerfile is located and that should solve the error.

solve the error.

Permission issues

```
WARNING: Error loading config file: /home/user/.docker/config.json -
stat /home/user/.docker/config.json: permission denied
```

If you face this kind of permission issues, you should either run the command with `sudo` or do the following:

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

This will add the current user to the Docker group which gives the current user the entire super-user level access for Docker.

Port issues

```
$ docker run -p 5000:5000 flask_app:1.0
docker: Error response from daemon: driver failed programming external connect
ivity on endpoint wizardly_ramanujan (d51f0d6f47ed3d559e767191adcd3b71fc364d3d
4ae3433787e0e2555948dcc8): Bind for 0.0.0.0:5000 failed: port is already alloc
ated.
```

This happens when you try to map an already-occupied host port to a container port.

There are two ways to solve this:

1. Type `docker ps` which will list all the running containers on the machine.

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
             STATUS              PORTS                    NAMES
a15ad5a56847        flask_app:1.0       "flask run"         14 hours ag
o        Up 14 hours         0.0.0.0:5000->5000/tcp   festive_ganguly
```

If you notice the `ports` column, you can see the 5000 port is already in use. Stop the container using `docker stop <CONTAINER ID>` and then retry.

2. When you don’t see the port in `docker ps` command output, it means the port is used by some other application. Use `lsof -i TCP:<PORT>`

This will print all the services using the specified port, then, you can kill one of them using `pkill` utility, but take precautions before killing a process.

them using `pkill` utility, but take precautions before killing a process.

```
$ lsof -i TCP:5000
COMMAND     PID                 USER   FD   TYPE             DEVICE SIZE/OFF N
ODE NAME
com.docke   999 venkateshachintalwar   43u  IPv6 0xa23249e6f915f25b      0t0
TCP *:commplex-main (LISTEN)
Python    43184 venkateshachintalwar    9u  IPv4 0xa23249e6f91699a3      0t0
TCP *:commplex-main (LISTEN)
```

Space issues (critical)

This issue is very critical and you will face it in production systems frequently if you are building images on production machines.

> When building an image, Docker pulls a lot of prebuilt images from the Docker registry and stores them on the local machine. Building images frequently makes Docker pull a lot of images and can result in space issues on the disk.

There are a couple of ways to tackle this problem:

1. Figure out how much space is used by Docker. Type `docker system df`. This will print space used by Docker in a human-readable format.

```
$ docker system df
TYPE                TOTAL               ACTIVE              SIZE
                RECLAIMABLE
Images              36                  6                   12.19GB
             11.48GB (94%)
Containers          18                  1                   28.31MB
             28.06MB (99%)
Local Volumes       33                  0                   1.818GB
             1.818GB (100%)
Build Cache         0                   0                   0B
                  0B
```

This is my system's stat. Notice the `total reclaimable` column. We can reclaim that much space for Docker.

2. Type `docker system prune`. This will prompt you to notify what it should remove

remove:

```
$ docker system prune
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all dangling images
  - all dangling build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
bd4fb9779be4c7a4306312a24ad9b3ff0e8eaf6bb3adccfc6611c9d4a44c716b
2db0f9c68e21897b4238392fe2947749f8b57d52227c80e29293a5ebcb39a180
43f3d18d03f5c5616e9b261163599695d90c52034411f688cc4d086b51aa08e7
69388164e1840c400cda8e3f09b4ab18843fc86d9ced93a8af5733d07a9f71f4
fbebcb5207d53e8c139d9201b033b5299e8279d70b841afb9e3fa95f76b613a0
a15ad5a568479526eafacd4cfc3e81479694364313e05c17665e314c0612326a
204df32b95048bc2ebd575215b3d731612160cd96fb28957f6208558ebf928a7
452b46853a194e81a6508765d07e2155e25b074da027802c774f7bf285a79a8c
64dcb356b5e4ac39938dd1f261b7121a5b187ba0fd512002bb8db0cfdf2c30db
6c10020040853cc659a6fa26b3d22645eed956e089bcfb26c8f96d4faa4a4dac
fc37723918bb0b6a94fc7fbc786534d0843963201bfd03ff0cd6699af8c417de
5f1e8417ed223219f0b0111323e3d213c07e4f55f5adaa189442c9d9a71eef04
0f7870b13f801c7a7e3a350ca0d0afe65d264fad5363d22fded4ff24d7d3f500
ba08f6839d59648dd3d539e6d1c3b1d787626fd8efe687f9e986fd7c428ff1d2
83ccef4e7abe255e5a2eb83cc97f3e336b4f8b22f249b75247e587a10dddfb4e
9f3d3d0d923fb518fc2d3d871ecdad2490e02f35c74cddd036aac5514c8418e0
84c4e3304c6f9d54b570c3a94da2bfb8125452eb9265c8ce9d8d6fae5a67894e

Deleted Networks:
flask_app_default
frontend_default
backend_default

Deleted Images:
deleted: sha256:de8d20e77a70ab59f09b4759f3b7476f2bc14f78a3d213c23451f903cdb88b
55
deleted: sha256:f4fc298e9bb0571cef34c9ce1de942fc7e961f3a599cdffdf98587a9da8436
d5
deleted: sha256:2f5311bdf41c3f6066909657cdd20bd176323dc92456dfebdf3912508ec3e1
9a
deleted: sha256:10970d749df66bb9aa8fbe690cb2afe9f02c9634bd4f2981c68d3a96526216
0b
deleted: sha256:92bb211bea7b830ef840e71d713c228899c5e3410f9c42680b9b8ac20f2768
c4
deleted: sha256:87abf541af47de9df2dcb2ea24d809e4588ee66e4962cc9ac9be80dd47485b
```

```
74
deleted: sha256:707ec6b5f54e3797b9916c7bc654e7f3eee4a556feb0d8f02a8c80b42e0548
0b
deleted: sha256:083a31255590627b3f64364f128aa9b90ad0edc4b69e6a2e6c8a4b0f3cb5ae
1b
deleted: sha256:b57b59de4d92dc7817f9b9230362e8332ac1ced766d42b13d70eddb414bc8b
ab
deleted: sha256:523ad4323258b87b8bfb5c585aa79d9283ab70032ff98b05ac898a9869caed
ff
deleted: sha256:b534b45d73504d93eabdc10540d8b7c6781bd8d64077b09639290d19e2f600
1c
deleted: sha256:7b2fdc416c1e55101dc4f5a9ec0a3e35c408379007138d1dffc7a97dd7db50
44
deleted: sha256:3edb8369bc83bd9a9d5700eacdd02282cf14c4f19973a096ff4b35531a3ccd
93
deleted: sha256:7d7b1f45aa5f36c47dc589b30ab0d3370b038700040c1f3598512e8f5744b2
4d
deleted: sha256:713884aa1af94fc2fc3b4af347db41874717902ee14098901c6e9722136773
21
deleted: sha256:f7635a5ae625be3c4d171ac1c9d154be6be257a283f10512faffcf93973b0d
ec
deleted: sha256:fe76f6800d31e5234eb9e7cea8d0205da005ee9b57a1c4878c043389dedb4b
aa
deleted: sha256:1b87706e8a423c21bbbca5b57fb538009519fefca8a00746e783c3fcdc81df
d7
deleted: sha256:f5ee99cd316966014e42cdf25cfeb4d77cea2fe245c2e7639ad0d2d9062aa7
e9
deleted: sha256:c78f6d2f8c7d325a29e3aa798581c4bdcd87d48a9fd7af1f2cf830cc1fe1f6
23
deleted: sha256:e68f780397b50ef63c1fcc1ec835df7e5e93a139d825eb941b1f248816a5a5
fa

Total reclaimed space: 1.356GB
```

This is housekeeping stuff. It will remove all useless things and create a space. You can also prune at the granular level if you want to clean specific things like:

`docker container prune`, `docker images prune`, `docker network prune`

There are a lot of other utilities to manage the disk, like `df -h`. You can make use of them also.

Container inspection

One of the best utilities is the `docker inspect` command. This is a little bit
advanced for beginners, but it's good to know what Docker consists of, so you can

advanced for beginners, but it's good to know what Docker consists of, so you can go through this command once.

You can get all configurations of a container, an image or a network. Type `docker inspect <Container/Image/Network ID>`

Below is the sample output for a container:

```
$ docker inspect 48a7ce046abe
[
    {
        "Id": "48a7ce046abeb2dffab2bf59545078048ba1c853658d79b33dd62ec8dc746aa3",
        "Created": "2020-04-04T05:54:10.8406701Z",
        "Path": "flask",
        "Args": [
            "run"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 4087,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2020-04-04T05:54:11.7320125Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:3507ec2e185181554a5614ca4cd76abaa2014c9b82b5c70f7d450e489fb98a8e",
        "ResolvConfPath": "/var/lib/docker/containers/48a7ce046abeb2dffab2bf59545078048ba1c853658d
        "HostnamePath": "/var/lib/docker/containers/48a7ce046abeb2dffab2bf59545078048ba1c853658d79
        "HostsPath": "/var/lib/docker/containers/48a7ce046abeb2dffab2bf59545078048ba1c853658d79b33
        "LogPath": "/var/lib/docker/containers/48a7ce046abeb2dffab2bf59545078048ba1c853658d79b33dd
53658d79b33dd62ec8dc746aa3-json.log",
        "Name": "/condescending_roentgen",
        "RestartCount": 0,
        "Driver": "overlay2",
        "Platform": "linux",
        "MountLabel": "",
        "ProcessLabel": "",
        "AppArmorProfile": "",
        "ExecIDs": null,
        "HostConfig": {
            "Binds": null,
            "ContainerIDFile": "",
            "LogConfig": {
                "Type": "json-file",
                "Config": {}
            },
            "NetworkMode": "default",
            "PortBindings": {
                "5000/tcp": [
                    {
                        "HostIp": "",
                        "HostPort": "5000"
                    }
                ]
            },
            "RestartPolicy": {
                "Name": "no",

                "MaximumRetryCount": 0
            },
            "AutoRemove": false,
            "VolumeDriver": "",
            "VolumesFrom": null,
            "CapAdd": null,
            "CapDrop": null,
            "Capabilities": null,
            "Dns": [],
            "DnsOptions": [],
            "DnsSearch": [],
            "ExtraHosts": null,
            "GroupAdd": null,
            "IpcMode": "private",
            "Cgroup": "",
            "Links": null,
            "OomScoreAdj": 0,
            "PidMode": "",
            "Privileged": false,
            "PublishAllPorts": false,
            "ReadonlyRootfs": false,
            "SecurityOpt": null,
            "UTSMode": "",
            "UsernsMode": "",
            "ShmSize": 67108864,
            "Runtime": "runc",
            "ConsoleSize": [
                0,
                0
            ],
            "Isolation": "",
            "CpuShares": 0,
            "Memory": 0,
            "NanoCpus": 0,
            "CgroupParent": "",
            "BlkioWeight": 0,
            "BlkioWeightDevice": [],
            "BlkioDeviceReadBps": null,
            "BlkioDeviceWriteBps": null,
            "BlkioDeviceReadIOps": null,
            "BlkioDeviceWriteIOps": null,
            "CpuPeriod": 0,
            "CpuQuota": 0,
            "CpuRealtimePeriod": 0,
            "CpuRealtimeRuntime": 0,
            "CpusetCpus": "",
            "CpusetMems": "",
            "Devices": [],
            "DeviceCgroupRules": null,
            "DeviceRequests": null,
            "KernelMemory": 0,
            "KernelMemoryTCP": 0,
            "MemoryReservation": 0,
            "MemorySwap": 0,
            "MemorySwappiness": null,
            "OomKillDisable": false,
            "PidsLimit": null,
            "Ulimits": null,
            "CpuCount": 0,
            "CpuPercent": 0,
            "IOMaximumIOps": 0,
            "IOMaximumBandwidth": 0,
            "MaskedPaths": [
                "/proc/asound",
                "/proc/acpi",
                "/proc/kcore",
                "/proc/keys",
                "/proc/latency_stats",
                "/proc/timer_list",
                "/proc/timer_stats",
                "/proc/sched_debug",
                "/proc/scsi",
                "/sys/firmware"
            ],
            "ReadonlyPaths": [
                "/proc/bus",
                "/proc/fs",
                "/proc/irq",
                "/proc/sys",
                "/proc/sysrq-trigger"
            ]
        },
        "GraphDriver": {
            "Data": {
                "LowerDir": "/var/lib/docker/overlay2/c1d40b060b8196e89151565c0cbcbb729479120aa578
                "MergedDir": "/var/lib/docker/overlay2/c1d40b060b8196e89151565c0cbcbb729479120aa57
                "UpperDir": "/var/lib/docker/overlay2/c1d40b060b8196e89151565c0cbcbb729479120aa578
                "WorkDir": "/var/lib/docker/overlay2/c1d40b060b8196e89151565c0cbcbb729479120aa5782
            },
            "Name": "overlay2"
        },
        "Mounts": [],
        "Config": {
            "Hostname": "48a7ce046abe",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": true,
            "AttachStderr": true,
            "ExposedPorts": {
                "5000/tcp": {}
            },
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
                "LANG=C.UTF-8",
                "GPG_KEY=E3FF2839C048B25C084DEBE9B26995E310250568",
                "PYTHON_VERSION=3.9.0a4",
                "PYTHON_PIP_VERSION=20.0.2",
                "PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/d59197a3c169cef378a22428a3
                "PYTHON_GET_PIP_SHA256=421ac1d44c0cf9730a088e337867d974b91bdce4ea2636099275071878c
                "FLASK_APP=app.py",
                "FLASK_RUN_HOST=0.0.0.0"
            ],
            "Cmd": [
                "flask",
                "run"
            ],
            "Image": "flask_app:1.0",
            "Volumes": null,
            "WorkingDir": "/code",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": {}
        },
        "NetworkSettings": {
            "Bridge": "",
            "SandboxID": "279d0393d5209ae1e7957ec4fd9e0a524115c597f01dc13b51273f47dc2d6e58",
            "HairpinMode": false,
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "Ports": {
                "5000/tcp": [
                    {
                        "HostIp": "0.0.0.0",
                        "HostPort": "5000"
                    }
                ]
            },
            "SandboxKey": "/var/run/docker/netns/279d0393d520",
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "EndpointID": "4b0f123e09c14ecc60586bbd9219b50bbb11a9fb297caf9cdb9bf3702b83621c",
            "Gateway": "172.17.0.1",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "MacAddress": "02:42:ac:11:00:02",
            "Networks": {
                "bridge": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "NetworkID": "915fe26ffbc81b05e2e313bd32d8794062e53590e870260feb8dfb9bfb45e8d3
                    "EndpointID": "4b0f123e09c14ecc60586bbd9219b50bbb11a9fb297caf9cdb9bf3702b83621
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:11:00:02",
                    "DriverOpts": null
                }
            }
        }
    }
]
```

Don’t worry about the output for now. Once you get used to Docker, you’ll understand most of these.