Vidyalankar Institute of Technology
Accredited A+ by NAAC

DEPARTMENT OF INFORMATION TECHNOLOGY Experiment No3

Semester	Semester VIII	
Subject	DevOps Lab	
Subject Professor In-charge	Prof. Yash Shah	
Laboratory	L11B	
Student Name	Ashwini Jadhav	
Roll Number	17101B0038	
Grade and Subject Teacher's Signature		

Experiment Number	3	
Experiment Title	To install and configure Docker for creating Containers	
Resources / Apparatus Required	Hardware: Compatible Computer System	Kali Linux, Docker
Objectives	Explore and implement containerization.	
Theory	**What is containerization?** It involves encapsulating or packaging up software code and all its dependencies so that it can run uniformly and consistently on any infrastructure. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. Container images become containers at runtime and in the case of Docker containers - images become containers when they run on Docker Engine. Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging. **Need of containerization:** ➢ Containerization reduces wasted resources because each container only holds the application and related binaries or libraries. ➢ By allowing more containers in the environment without the need for more servers, containerization increases scalability anywhere from 10 to 100 times that of traditional VM environments.	

- The ability to rapidly spin up new containers also increases the capacity to handle website traffic load seamlessly.
- Using containerization helps your cloud environment efficiency; by deploying multiple containerized applications on to a single cloud instance, you get much closer to achieving 100% utilization.
- Improved security by isolating applications from the host system and from each other.
- Faster app start-up and easier scaling.
- Flexibility to work on virtualized infrastructures or on bare metal servers
- Easier management since install, upgrade, and rollback processes are built into the Kubernetes platform.

How to carry out containerization using Docker:

Create a docker image by pulling from docker:
docker pull ubuntu (different OS have their own image)

Build a container by running an image:
docker run -it -d *image_name*
(-it, makes container interactive,
-d, stands for daemon, container will work in background)

Developer has to enter container to put project files. In order to do so, we execute the container:
docker exec -it *container_id* bash

Container id can be obtained from:
docker ps

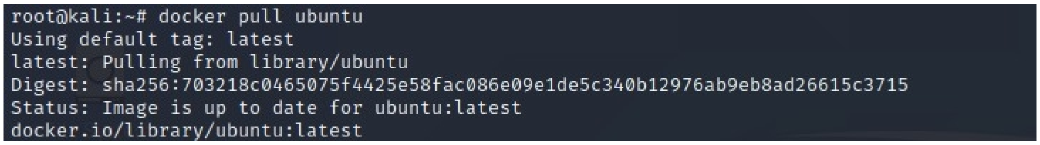
Add project files within container.

To exit container:
exit

Developer can send only image to tester. To create an image:
docker commit *container_id name*

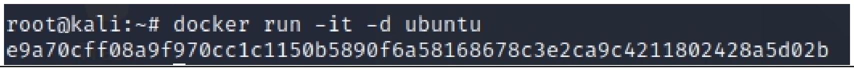
Output

Creating docker image:



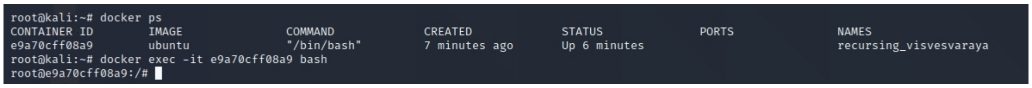
```
root@kali:~# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:703218c0465075f4425e58fac086e09e1de5c340b12976ab9eb8ad26615c3715
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```

Building a container:



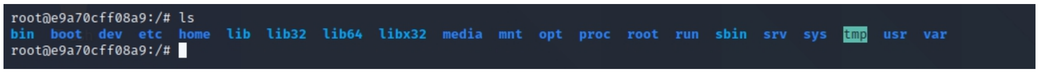
```
root@kali:~# docker run -it -d ubuntu
e9a70cff08a9f970cc1c1150b5890f6a58168678c3e2ca9c4211802428a5d02b
```

Obtaining container ID and entering container:



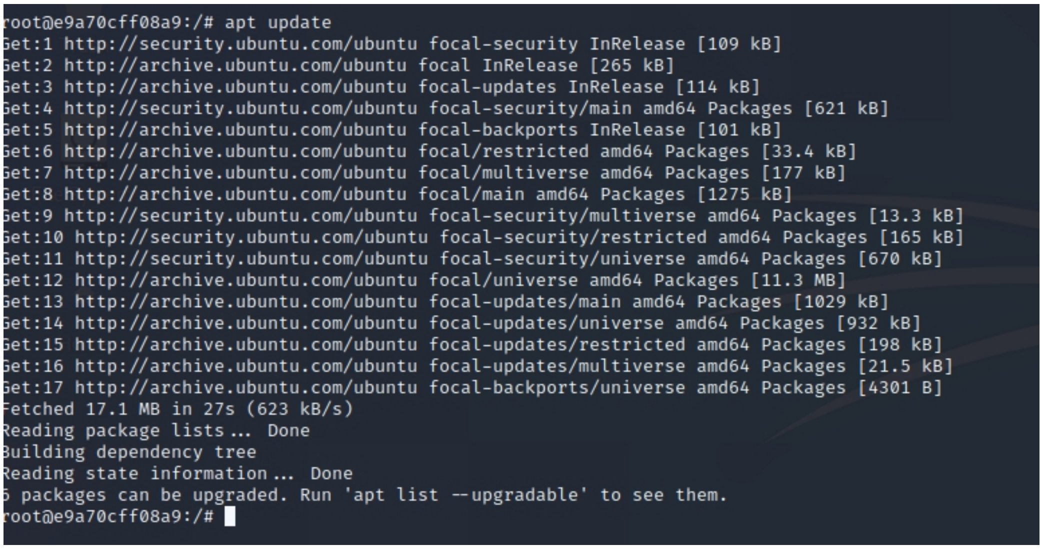
```
root@kali:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
e9a70cff08a9        ubuntu              "/bin/bash"         7 minutes ago       Up 6 minutes                            recursing_visvesvaraya
root@kali:~# docker exec -it e9a70cff08a9 bash
root@e9a70cff08a9:/# 
```

Only Basic OS in container:



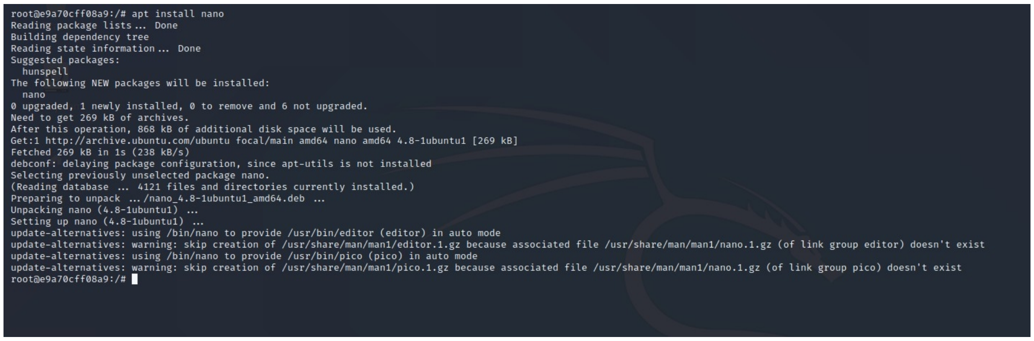
```
root@e9a70cff08a9:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@e9a70cff08a9:/# 
```

Updating apt within container:



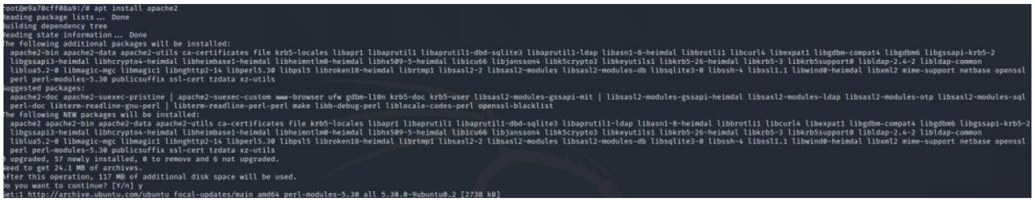
```
root@e9a70cff08a9:/# apt update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [109 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [621 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [13.3 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [165 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [670 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1029 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [932 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [198 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [21.5 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [4301 B]
Fetched 17.1 MB in 27s (623 kB/s)
Reading package lists ... Done
Building dependency tree
Reading state information ... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@e9a70cff08a9:/# 
```

Installing text editor 'nano' within container:



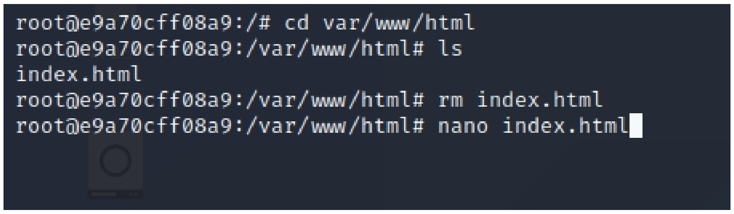
```
root@e9a70cff08a9:/# apt install nano
Reading package lists ... Done
Building dependency tree
Reading state information ... Done
Suggested packages:
  hunspell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 269 kB of archives.
After this operation, 868 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 nano amd64 4.8-1ubuntu1 [269 kB]
Fetched 269 kB in 1s (238 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package nano.
(Reading database ... 4121 files and directories currently installed.)
Preparing to unpack .../nano_4.8-1ubuntu1_amd64.deb ...
Unpacking nano (4.8-1ubuntu1) ...
Setting up nano (4.8-1ubuntu1) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/editor.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group editor) doesn't exist
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/pico.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group pico) doesn't exist
root@e9a70cff08a9:/# 
```

Installing apache within container:



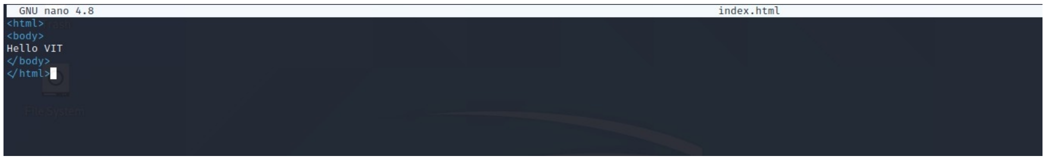
```
root@e9a70cff08a9:/# apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils ca-certificates file krb5-locales libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libasn1-8-heimdal libbrotli1 libcurl4 libexpat1 libgdbm-compat4 libgdbm6 libgssapi-krb5-2
  libgssapi3-heimdal libhcrypto4-heimdal libheimbase1-heimdal libheimntlm0-heimdal libhx509-5-heimdal libicu66 libjansson4 libk5crypto3 libkeyutils1 libkrb5-26-heimdal libkrb5-3 libkrb5support0 libldap-2.4-2 libldap-common
  liblua5.2-0 libmagic-mgc libmagic1 libnghttp2-14 libperl5.30 libpsl5 libroken18-heimdal librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libsqlite3-0 libssh-4 libssl1.1 libwind0-heimdal libxml2 mime-support netbase openssl
  perl perl-modules-5.30 publicsuffix ssl-cert tzdata xz-utils
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser ufw gdbm-l10n krb5-doc krb5-user libsasl2-modules-gssapi-mit | libsasl2-modules-gssapi-heimdal libsasl2-modules-ldap libsasl2-modules-otp libsasl2-modules-sql
  perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl make libb-debug-perl liblocale-codes-perl openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils ca-certificates file krb5-locales libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libasn1-8-heimdal libbrotli1 libcurl4 libexpat1 libgdbm-compat4 libgdbm6 libgssapi-krb5-2
  libgssapi3-heimdal libhcrypto4-heimdal libheimbase1-heimdal libheimntlm0-heimdal libhx509-5-heimdal libicu66 libjansson4 libk5crypto3 libkeyutils1 libkrb5-26-heimdal libkrb5-3 libkrb5support0 libldap-2.4-2 libldap-common
  liblua5.2-0 libmagic-mgc libmagic1 libnghttp2-14 libperl5.30 libpsl5 libroken18-heimdal librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libsqlite3-0 libssh-4 libssl1.1 libwind0-heimdal libxml2 mime-support netbase openssl
  perl perl-modules-5.30 publicsuffix ssl-cert tzdata xz-utils
0 upgraded, 57 newly installed, 0 to remove and 6 not upgraded.
Need to get 24.1 MB of archives.
After this operation, 117 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 perl-modules-5.30 all 5.30.0-9ubuntu0.2 [2738 kB]
```

Creating 'sample project' within container:



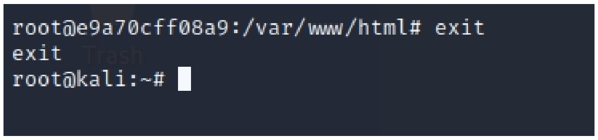
```
root@e9a70cff08a9:/# cd var/www/html
root@e9a70cff08a9:/var/www/html# ls
index.html
root@e9a70cff08a9:/var/www/html# rm index.html
root@e9a70cff08a9:/var/www/html# nano index.html
```

Sample Index file:



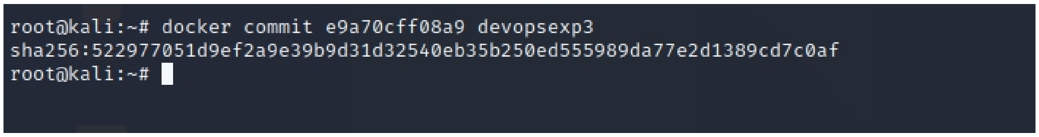
```
<html>
<body>
Hello VIT
</body>
</html>
```

To go out of the container:



```
root@e9a70cff08a9:/var/www/html# exit
exit
root@kali:~#
```

Creating an image of the container:



```
root@kali:~# docker commit e9a70cff08a9 devopsexp3
sha256:522977051d9ef2a9e39b9d31d32540eb35b250ed555989da77e2d1389cd7c0af
root@kali:~#
```

Conclusion	Thus, we have implemented containerization using Docker.