References for Docker Installation:
https://www.starwaredesign.com/index.php/blog/64-fpga-meets-devops-xilinx-vivado-and-jenkins-with-docker

https://www.starwaredesign.com/index.php/blog/63-fpga-meets-devops-xilinx-vivado-and-git

https://github.com/starwaredesign/vivado-docker

References for Docker Commands:
https://www.tutorialspoint.com/docker/docker_hub.htm
https://www.tutorialspoint.com/docker/index.htm
https://docs.docker.com/engine/reference/commandline/cli/
https://docs.docker.com/engine/reference/commandline/exec/

Introduction to Docker:

In this second blog post of the series "FPGA meets DevOps" I am going show you how to integrate Xilinx Vivado with Docker and Jenkins.

Docker provides a lightweight operating system level virtualisation. It allows developers to package up an application with all the parts it needs in a container, and then ship it out as one package. A container image is described by a file (Dockerfile) which contains a sequence of commands to create the image itself (i.e.: packages to install, configuration tasks, etc) and it is all you need to replicate the exact build environment on another machine.

The objective is to create a container that will run Vivado in headless mode (without user interface) to build the FPGA image.

First install Docker for your Linux distribution, i.e. for Ubuntu:
https://docs.docker.com/install/linux/docker-ce/ubuntu/

Pre-installation Steps:
– Download a full Xilinx Vivado image (not the web installer) from Xilinx(owned by AMD) website
– The tar.gz file needs to be provided during the container image creation via a web-server. If you don't have a web-server you can i.e. launch a very simple one written in Python 3, just run this command line from the directory were the Vivado tar.gz file is:

```
$ python3 -m http.server
```

Response: `Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...`
This will run a web-server listening on port 8000 and providing the content of the directory where the script is run over HTTP.

– Add the tar.gz file to the vivado-docker folder which will be cloned from git in the below mentioned Step 1

Installation Steps:

Step1:
Clone or download the repository containing the Dockerfile

```
Git clone https://github.com/starwaredesign/vivado-docker.git
```

```
Response:
```

```
Cloning into 'vivado-docker'...
```

```
remote: Enumerating objects: 54, done.
```

```
remote: Total 54 (delta 0), reused 0 (delta 0), pack-reused 54
```

```
Unpacking objects: 100% (54/54), done.
```

```
Checking connectivity... done.
```

Step2:
Get a license for Vivado from the Xilinx website and copy Xilinx.lic inside the vivado-docker directory
<--attach example of Xilinx.lic in repo>

Step3:
Create the docker image:

```
$ sudo docker build --build-arg VIVADO_TAR_HOST=192.168.1.100:8000
--build-arg VIVADO_TAR_FILE=Xilinx_Vivado_SDK_2018.3_1207_2324 -t
vivado:2018.3 --build-arg VIVADO_VERSION=2018.3 .
```

In this example I am creating a Vivado 2018.3 image. You might need to adjust the VIVADO_TAR_HOST and VIVADO_TAR_FILE arguments based on your system configuration.

Response:

```
Sending build context to Docker daemon  140.3kB
```

```
Step 1/17 : FROM ubuntu:16.04

…

Step 17/17 : COPY Xilinx.lic /home/vivado/.Xilinx/

---> 77e6b40ccf04

Successfully built 77e6b40ccf04

Successfully tagged vivado:2018.3
```

Note:
2 script files are primarily involved:

- Install-config.sh
- Dockerfile.sh

If problems are encountered during any of the 17 steps of image creation, change the individual commands in the script file accordingly.
“chmod 777” is a command that can be used to change access mode to ‘read, write and execute’. This command maybe required in order to change the commands in the script file.

You can list the available docker images:

```
$ sudo docker image ls

REPOSITORY            TAG IMAGE ID            CREATED SIZE

vivado                2018.3 77e6b40ccf04        10 minutes ago 20.2GB
```

Step4:

We can now install Jenkins (https://jenkins.io/). Jenkins is an open source automation tools that helps to automate the build, test and deployment process. Jenkins processes can be triggered by a source version control commit, scheduled i.e. every day or from the completion of another process.

The objective is to get Jenkins to build the FPGA bitstream using the Docker image we've built earlier.

Jenkins itself can run inside a Docker container (https://jenkins.io/doc/book/installing/)

```
$ sudo docker run \
 -u root \
 --rm \
 -d \
 -p 8080:8080 \
 -p 50000:50000 \
 -v jenkins-data:/var/jenkins_home \
 -v /var/run/docker.sock:/var/run/docker.sock \
 jenkinsci/blueocean
```

This will download the Jenkins Docker container and run it. Then get access to Jenkins through a web browser on port 8080. When accessing Jenkins the first time, use the automatically generated admin password that can be found in the docker image logs

```
$ sudo docker ps
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
3f99cea6c114        jenkinsci/blueocean  "/sbin/tini -- /usr/…"   4 minutes ago       Up 4 minutes        0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp   musing_haibt
```

Copy the container ID (in this case 3f99cea6c114)

```
$ sudo docker logs 3f99cea6c114
```

```
Step5:
```

Create a first user account. This is a very simple setup that doesn't take into account security of the Jenkins server. Please look at the official documentation. (https://jenkins.io/doc/book/system-administration/security/) on how to create users, set permissions, use https, etc.

Jenkins is an example of a CICD Software
(Continuous Integration and Continuous Development)

—Continued here from Koteshwar sir–

Note:
Once the docker container is created.. Keep a backup in dockerhub

Few useful Commands:

- docker ps: To display the list of existing running containers
- docker images: To list the images present
- docker run –help: Explore few available features related to docker containers and images

Giving access to USB ports:
–tty

-t: tty- lib-modules(related to USB ports) will be given access to, for docker..basically to enable remote working on FPGA Boards

-i: interactive

–cap -add: to add linux capabilities
-a: to display those containers that aren't running

Running a present application in docker-
https://docs.docker.com/get-started/02_our_app/#:~:text=Now%20that%20you%20have%20an,use%20the%20docker%20run%20command.&text=You%20use%20the%20%2Dd%20flag,to%20the%20container's%20port%203000.

LXC Container(Alternative to docker)-
http://threespeedlogic.com/running-vivado-on-lxc.html

USB Access-

Finally the programming tool that can transfer the bitstream to your FPGA. This task is still not straightforward to run on a container in Windows and Mac since Docker Desktop runs in a virtual machine and is not able to access your locally connected USB devices. I recommend getting the pre-built OpenOCD binaries on https://github.com/xpack-dev-tools/openocd-xpack/releases or using `brew install open-ocd` if using a Mac.

Running Vivado on Docker-
https://carlosedp.medium.com/xilinx-open-source-fpga-toolchain-on-docker-containers-932026650a615; contains bitstream information
http://phwl.org/2020/xilinx-vivado-on-ubuntu-using-docker/
https://hackaday.com/2021/05/05/using-docker-to-sail-through-open-source-xilinx-fpga-development/

Latest command to type in terminal to run the docker image:
root@iitdh-OptiPlex-3070:/home/iitdh# docker run -e DISPLAY=`hostname`:0.0 -it --rm -v $PWD:/home/user/work -w /home/user vivado_latest

Response from terminal: vivado@3532e819fc09:/home/user$
What command to type here to open vivado?
Solution:
https://ez.analog.com/fpga/f/q-a/51646/bin-sh-vivado-command-not-found
https://wiki.analog.com/resources/fpga/docs/build