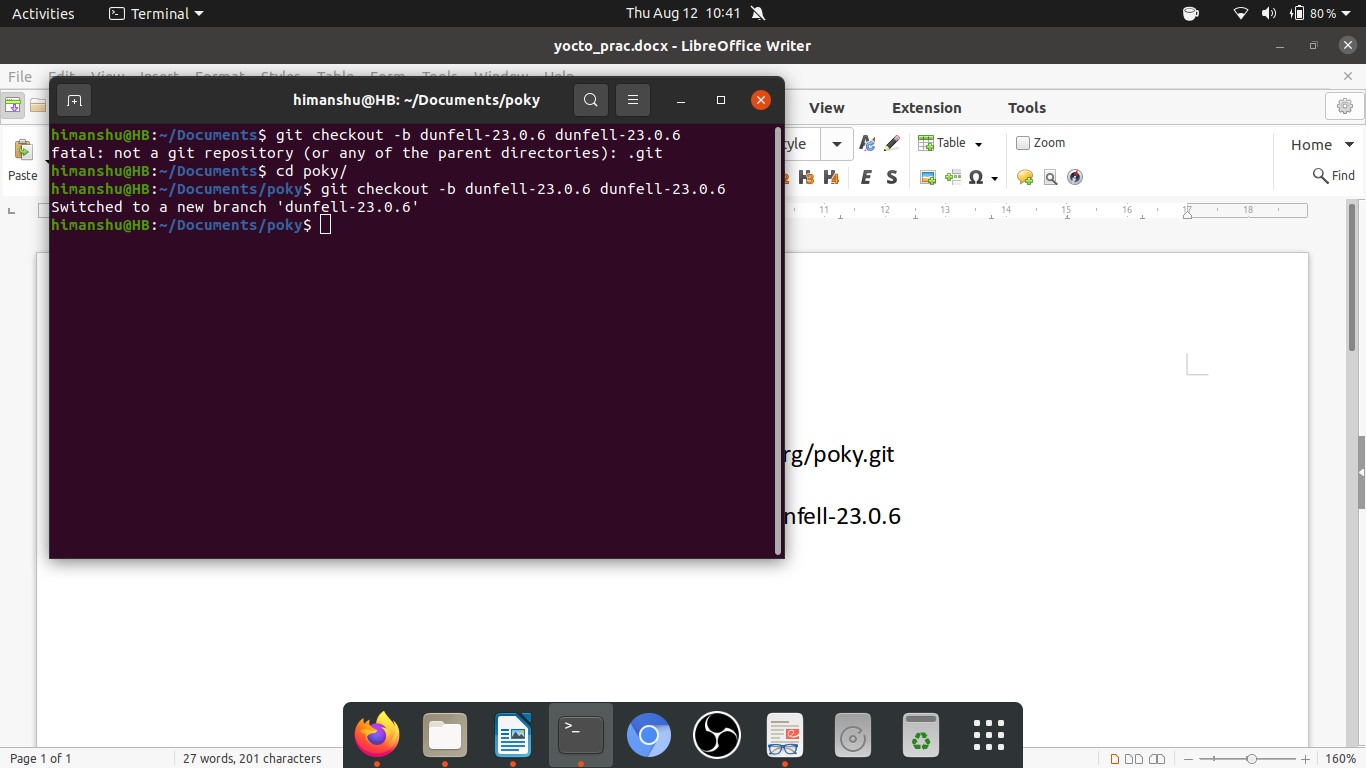
Doing the yocto build with bootlin lab pdf

1. install the yocto

**git clone git://git.yoctoproject.org/poky.git**

2. checkout the requireed version/branch of yocto

**git checkout -b dunfell-23.0.6 dunfell-23.0.6**



3. install the meta layer for arm

**git clone git://git.yoctoproject.org/meta-arm.git**

**git clone git://git.yoctoproject.org/meta-ti.git**

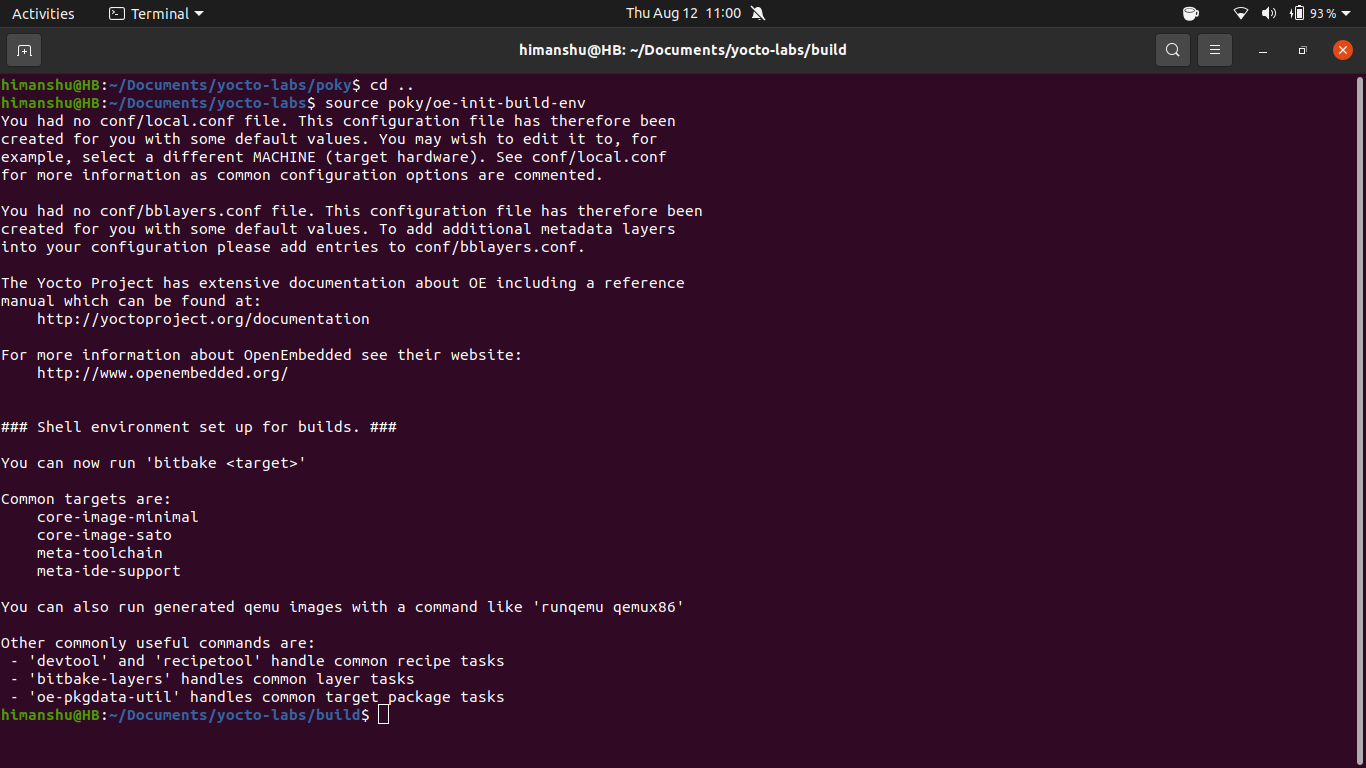
**cd $HOME/yocto-labs/meta-ti**

**git checkout -b dunfell-labs 07.03.00.005**

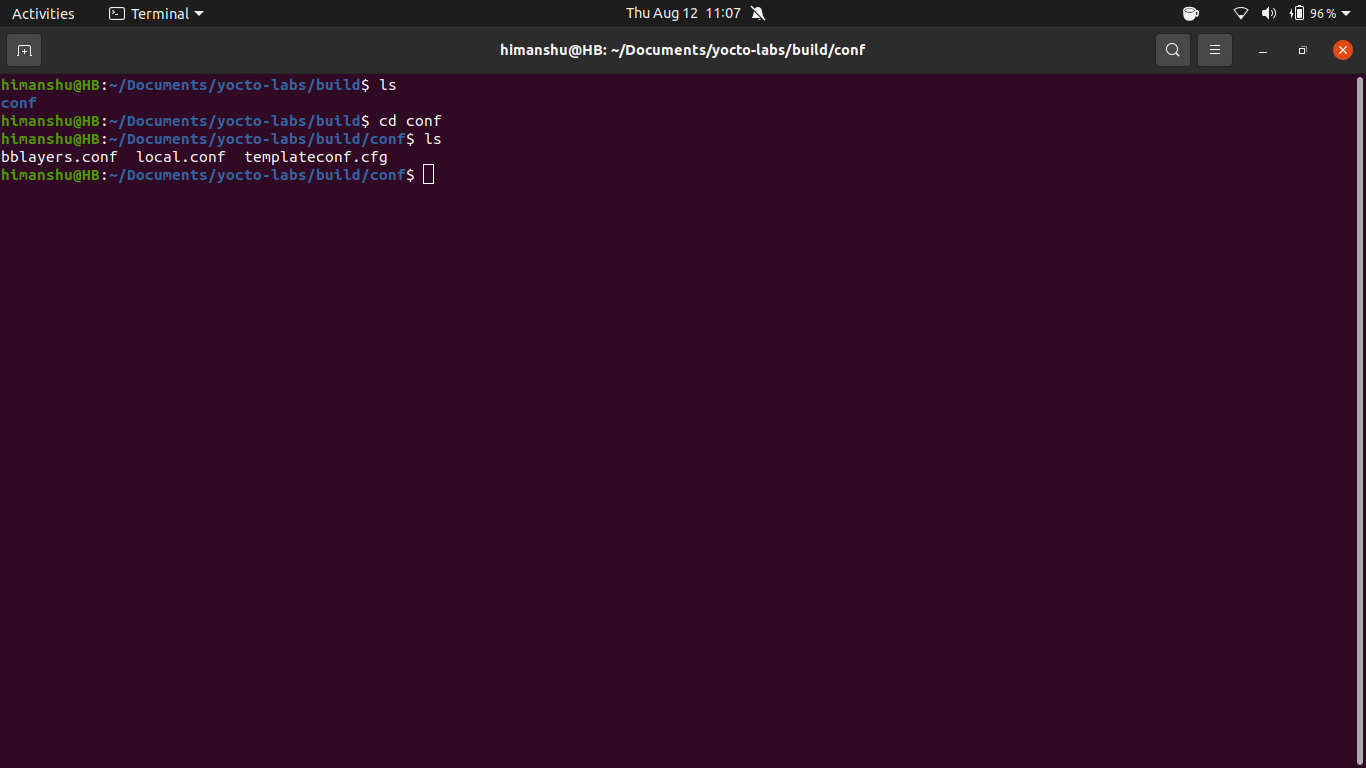
**cd $HOME/yocto-labs/meta-arm**

**git checkout 1cf8b975e1c40bf8e8c0bf315db5d4cddcb01a7b**

4. set up the build environment



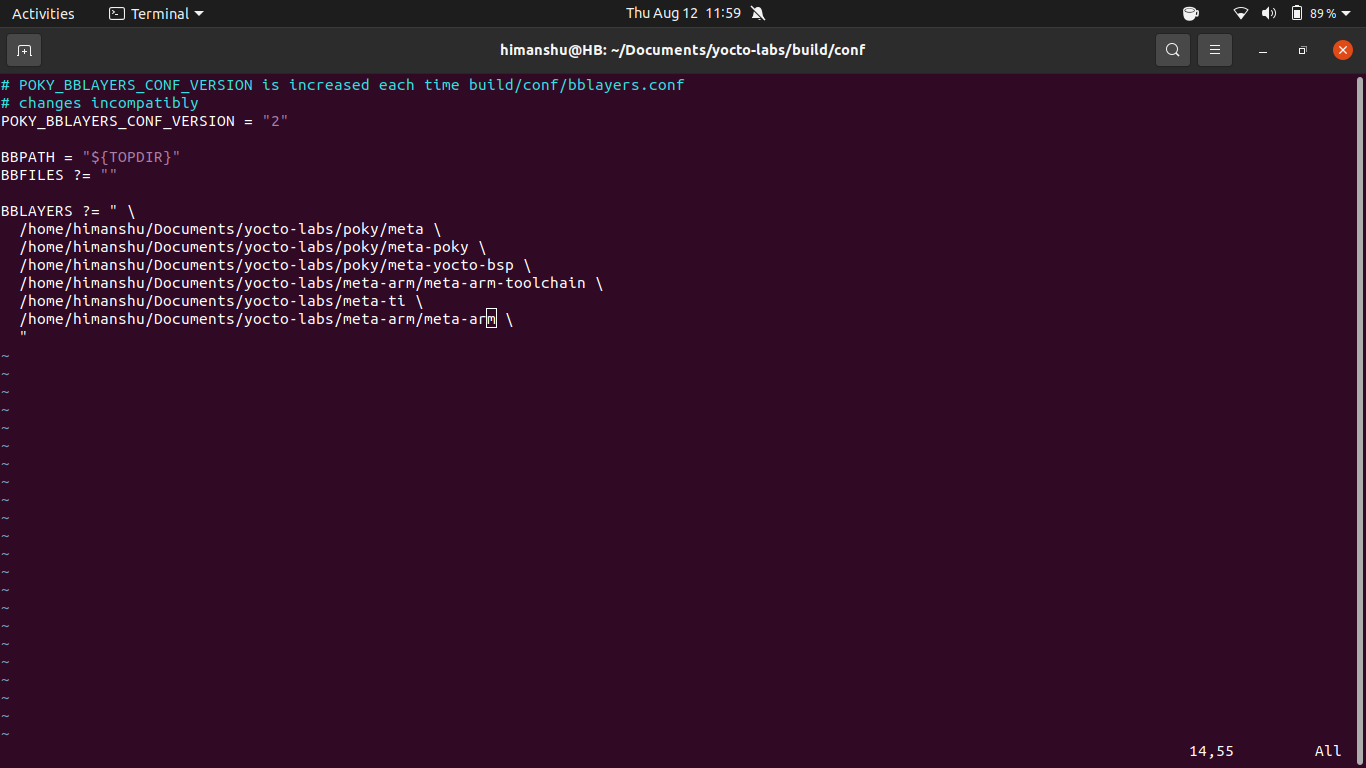
after this above setting up environment there will be a build folder generated containing all the required file along with the recipe file



Don’t forget to make the configuration aware of the ARM and TI layers. Edit the layer configura-

tion file ($BUILDDIR/conf/bblayers.conf) and append the full path to the meta-arm-toolchain,

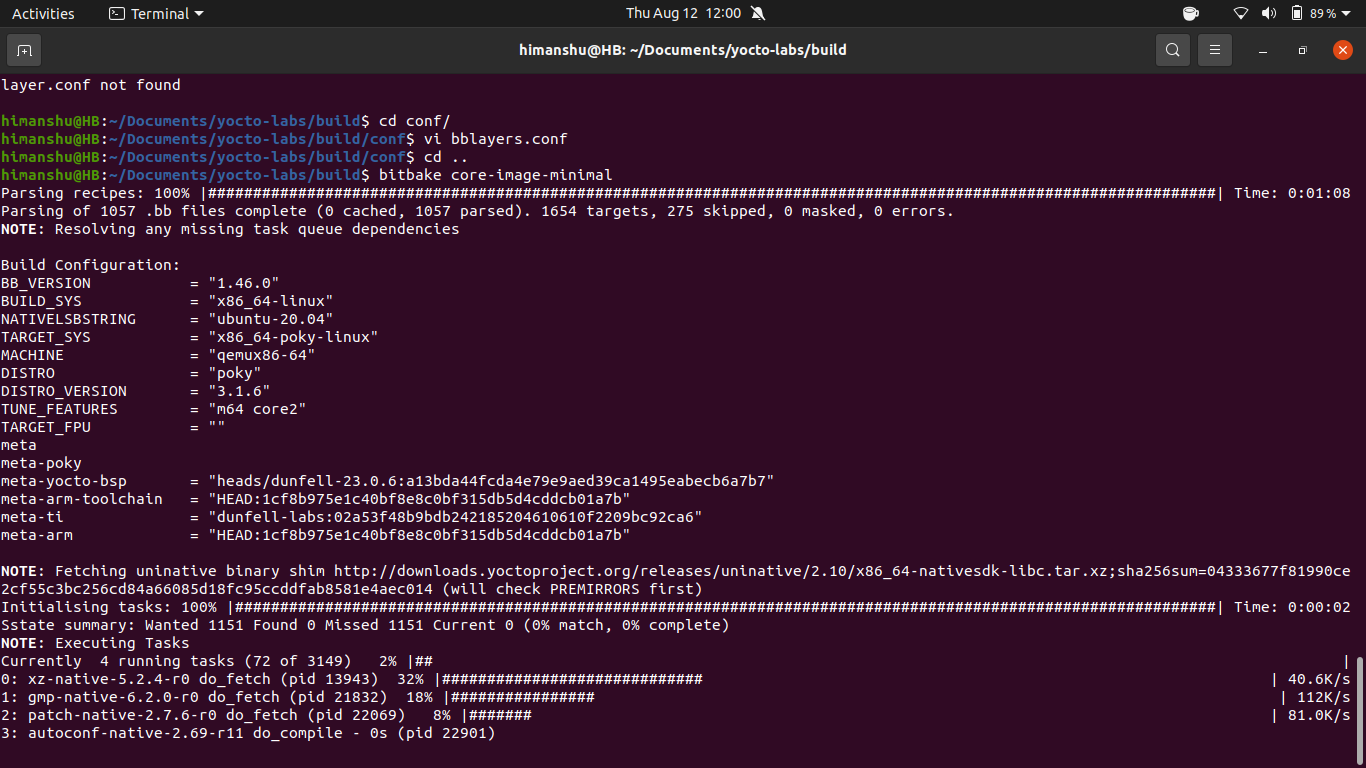
meta-arm, meta-ti directories to the BBLAYERS variable.



5. install the first image

**bitbake core-image-minimal**

by deafualt its a qemu-arm image that is going to be compiled



**Adding package to build build**

1. open the https://layers.openembedded.org/layerindex/branch/master/layers/

<https://layers.openembedded.org/layerindex/branch/master/recipes/>

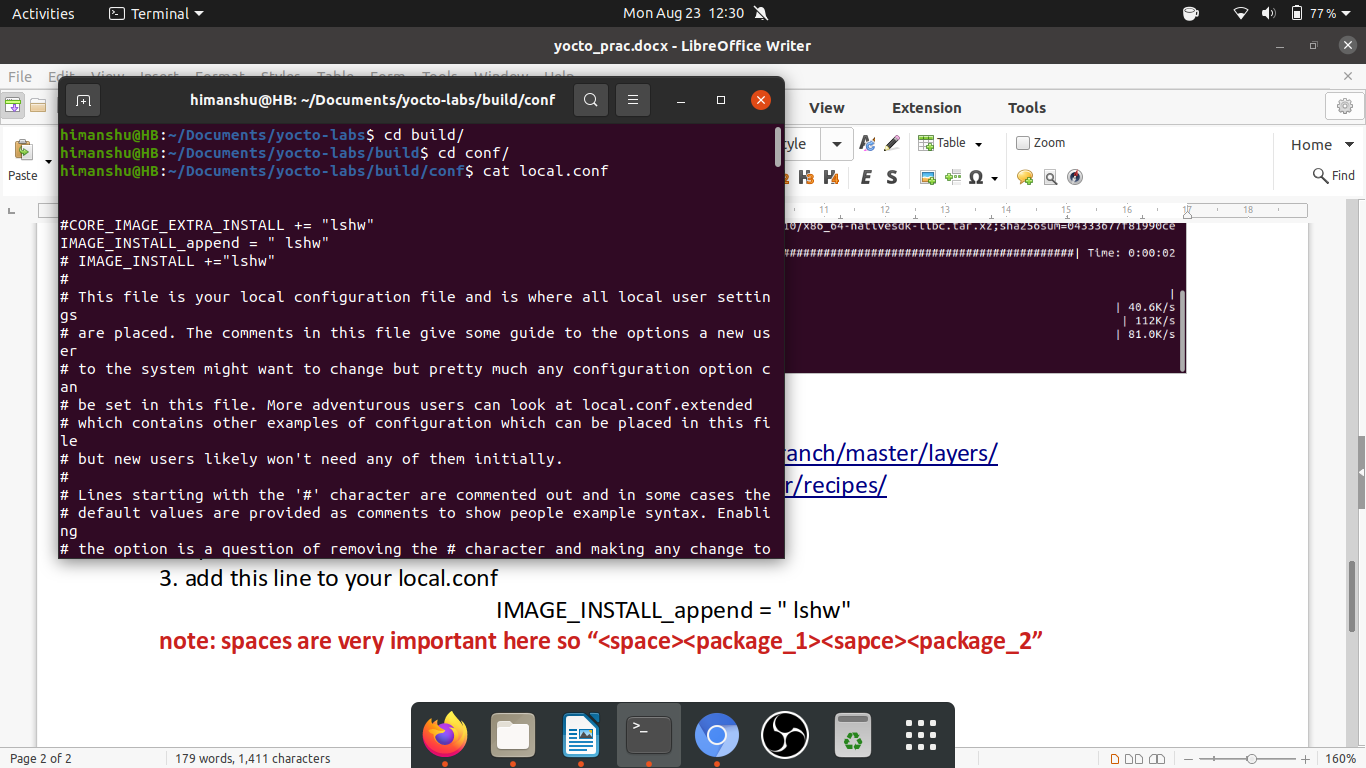
site and check what package you wanted in your build

2. open the build/conf/local.conf file

3. add this line to your local.conf

IMAGE\_INSTALL\_append = " lshw"

**note: spaces are very important here so “<space><package\_1><sapce><package\_2”**



4. then do bitbake core-image-minimal

23rd august – watching a tutorial for yocto layers

<https://www.youtube.com/watch?v=nqHylLP2NmA>-simple layer, custom image and devtool

1. create a folder and name it anything – this folder will be yout meta-layer

2. use bitbake command to make that folder as layer

**bitbake-layers create-layer <folder name>**

2. add path of your own meta layer into the conf/bblayer.conf

3. inside that meta folder go

meta-collins/recipes-example/

create a new folder name images inside that recipe-example

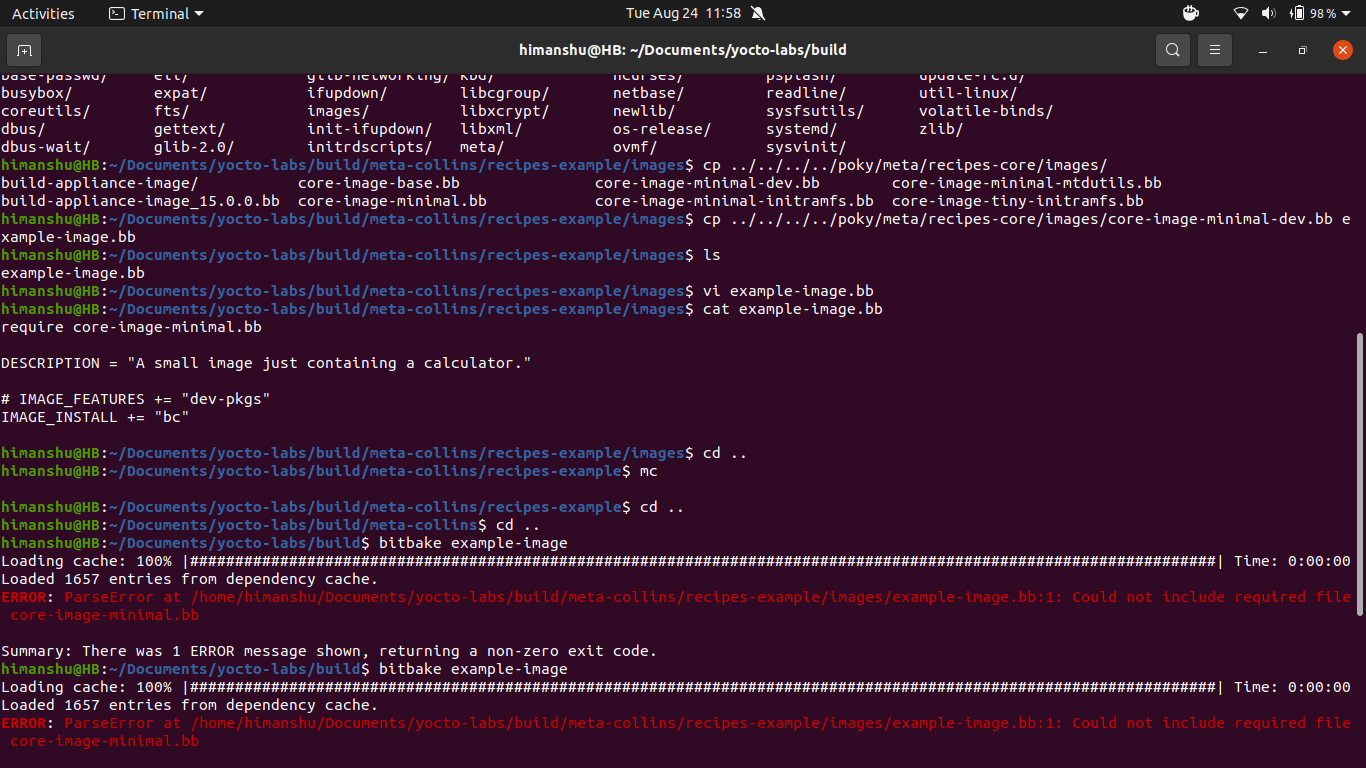
4. for the sake of simplicity copy some already present layer config file inside that images folder

note: copy the file but you must change the name of that bb file otherwise error will occur

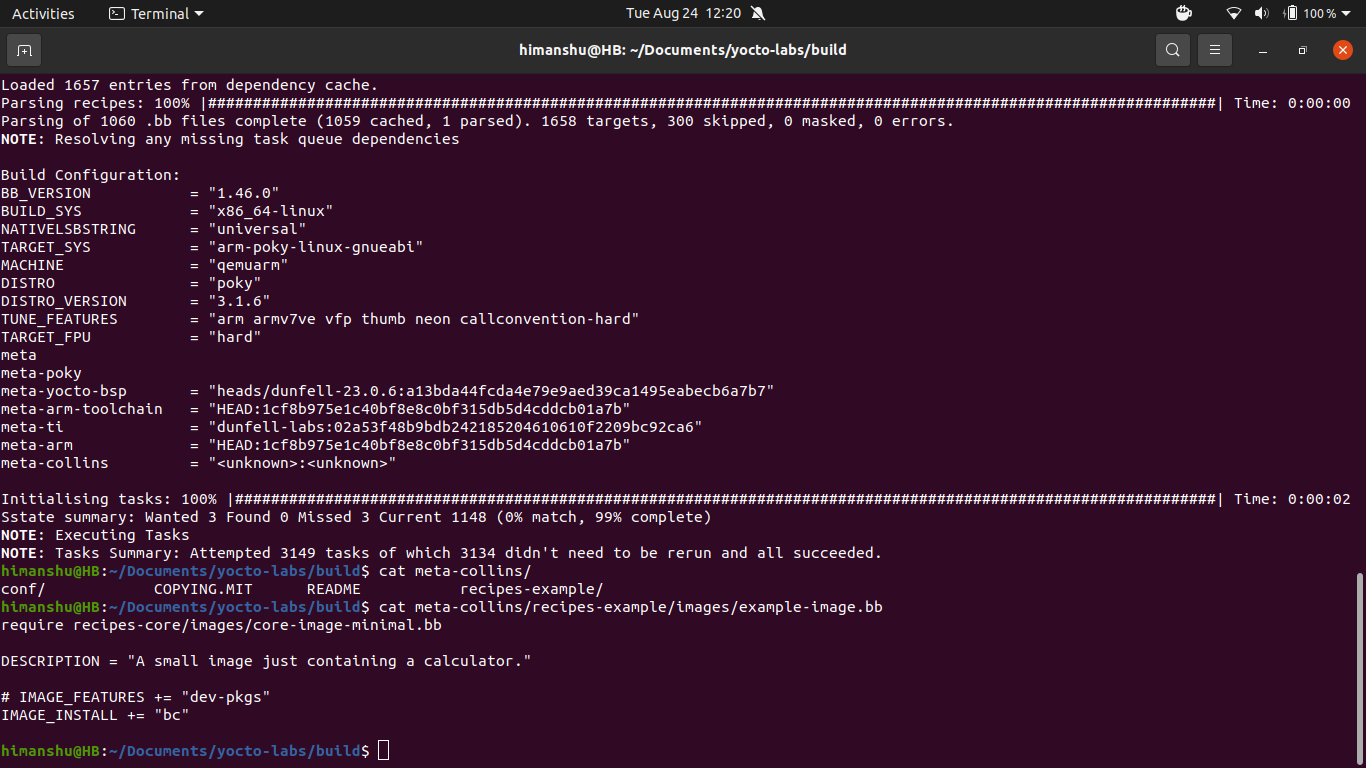
for example: i have named that file as example-image.bb

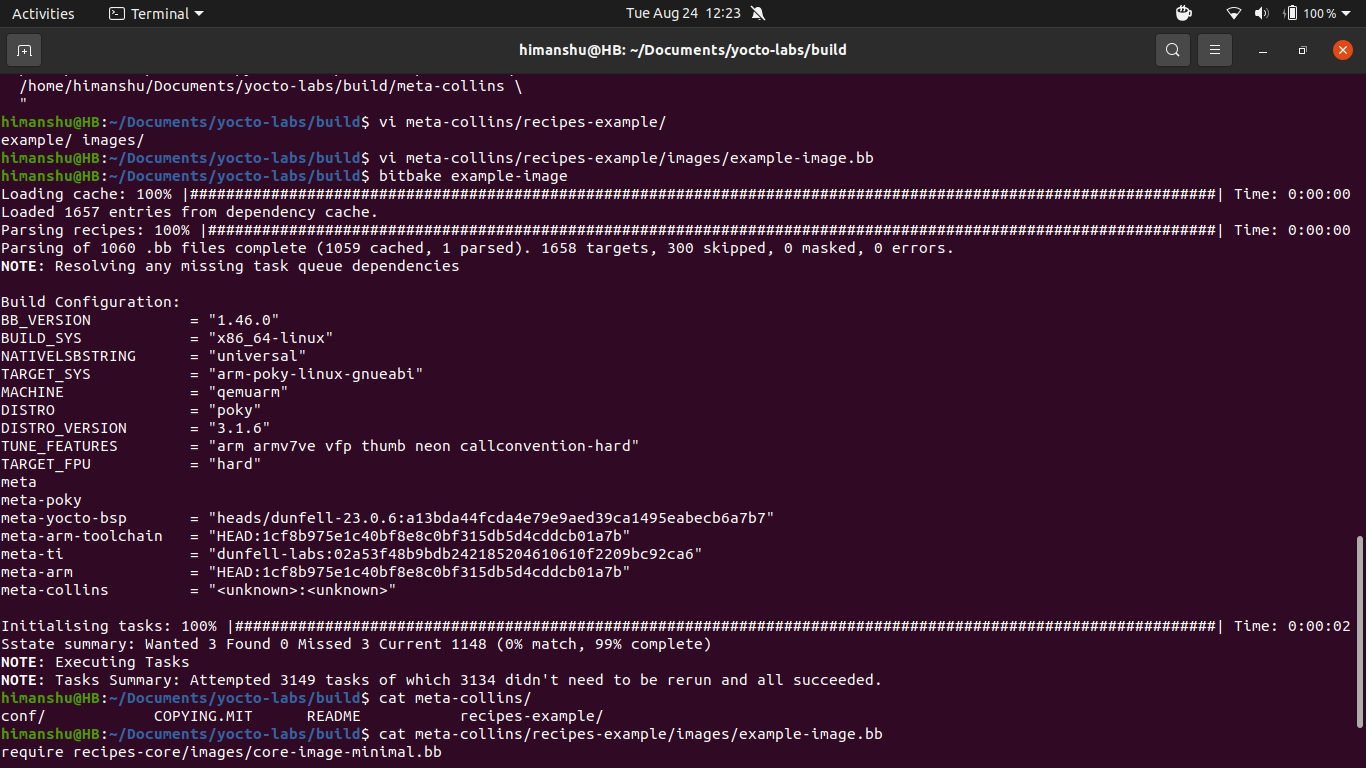
5. **bitbake example-images** {this will build the image}

6. note: this will give an error- this error is due to the fact that the .bb file that we wanted to create

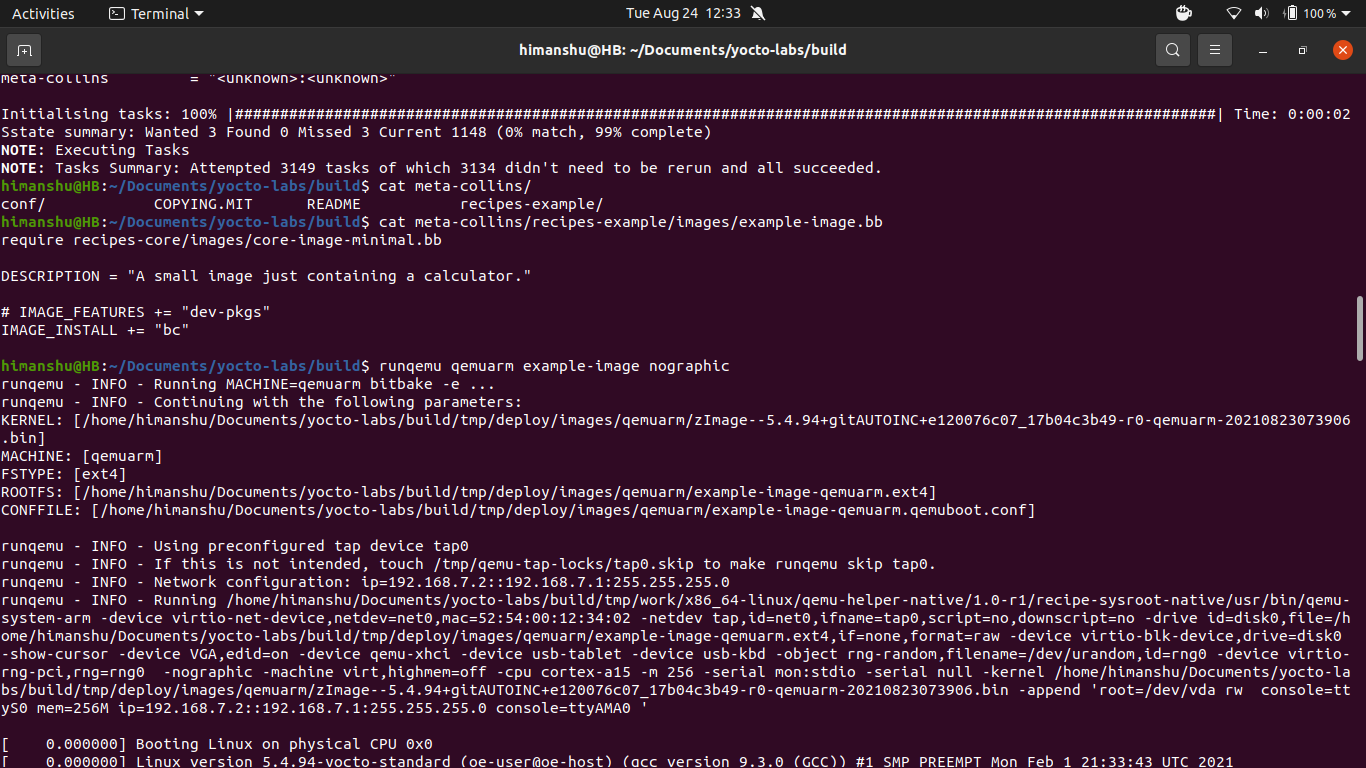


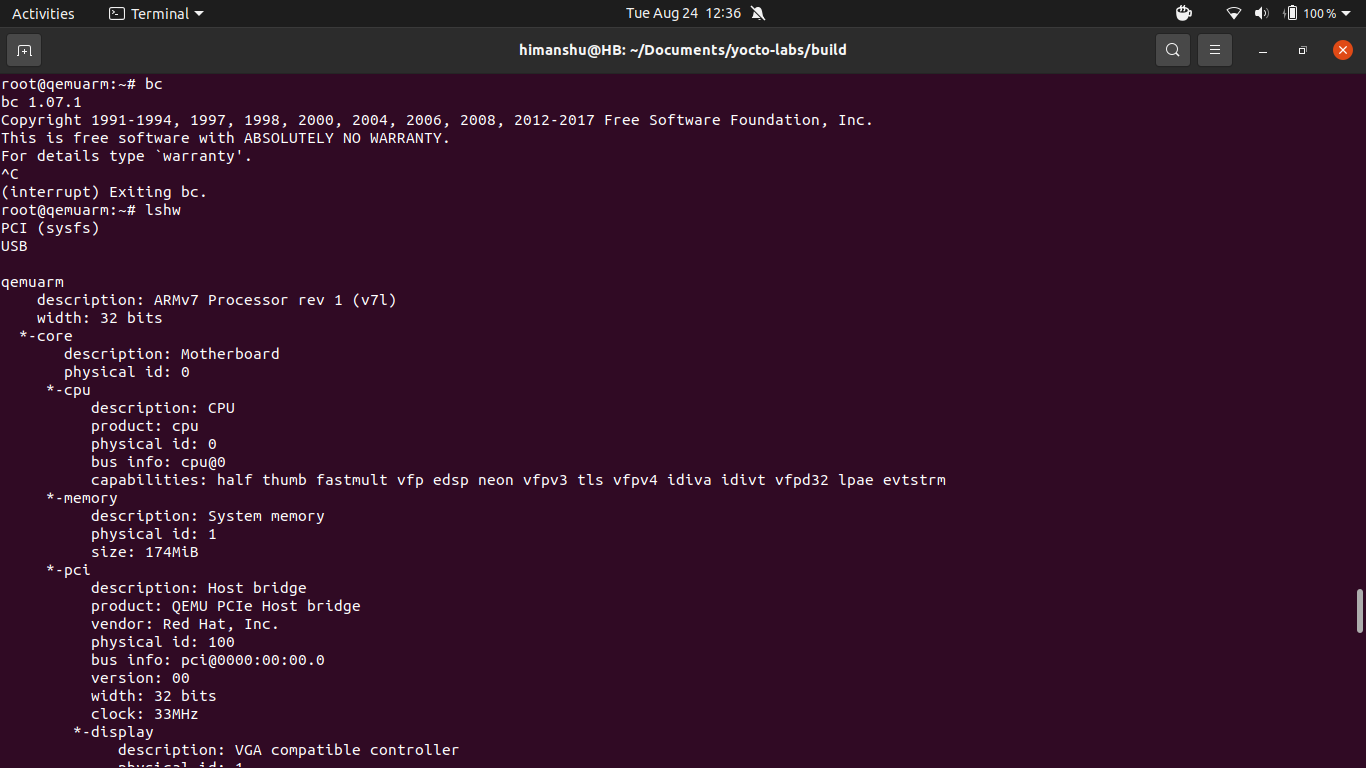
require other layers and we need to give path to those requireed file or we can copy those required files in the same folder



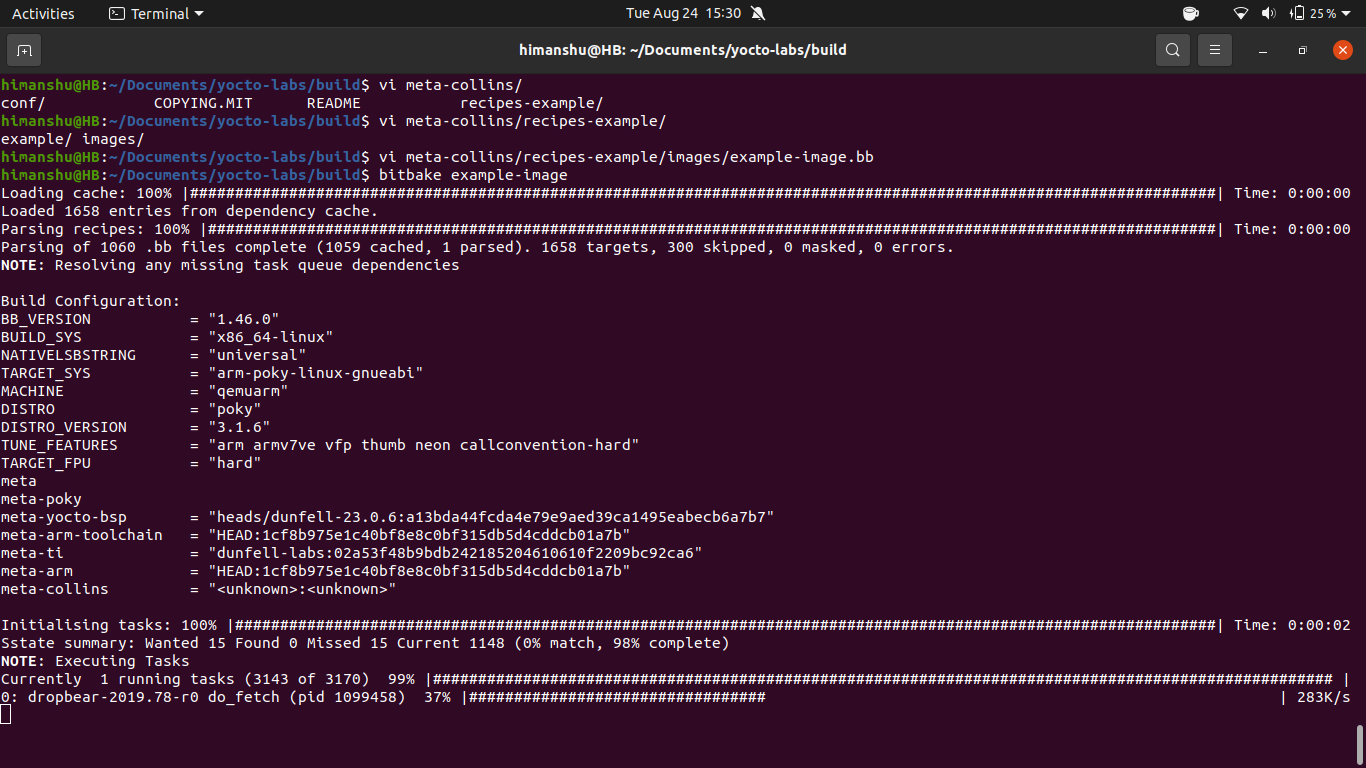


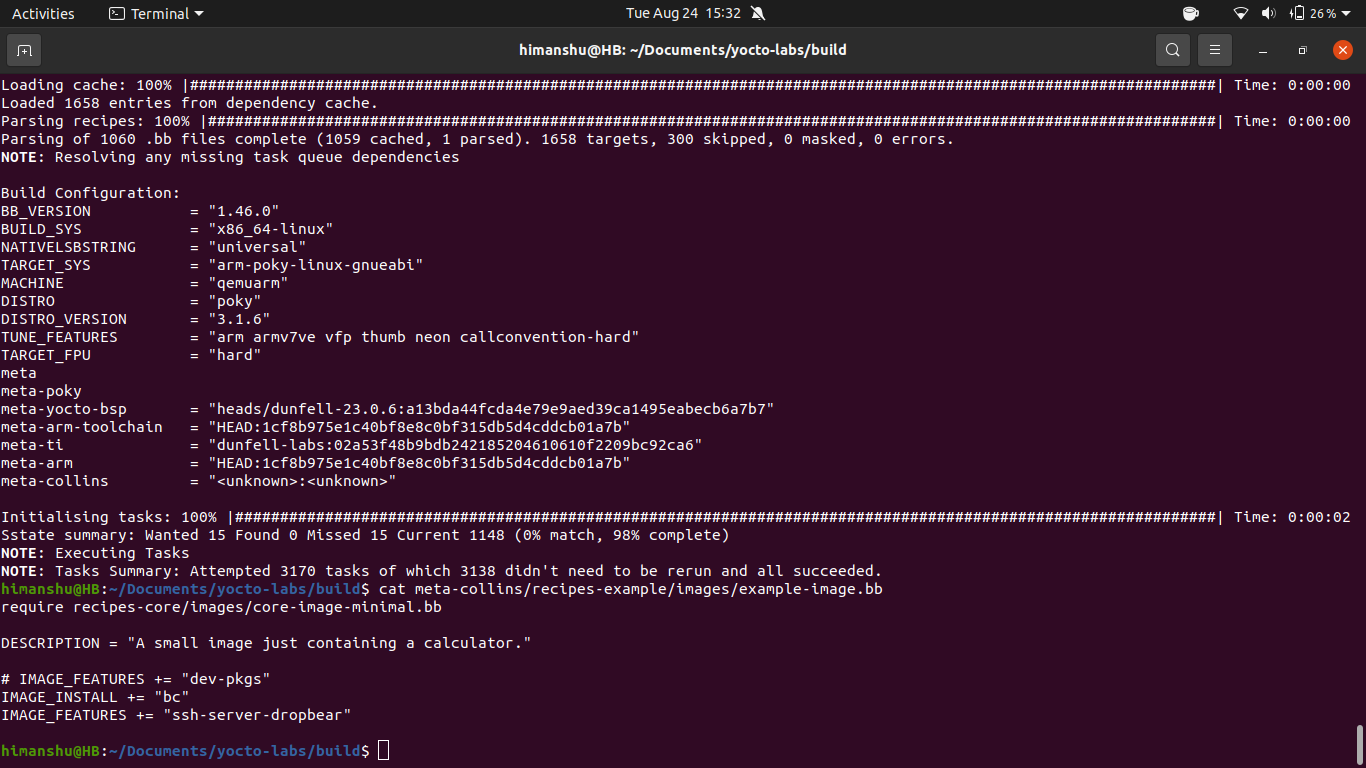
7. run the build





8. you can also add packages by adding the package name in your example-image.bb

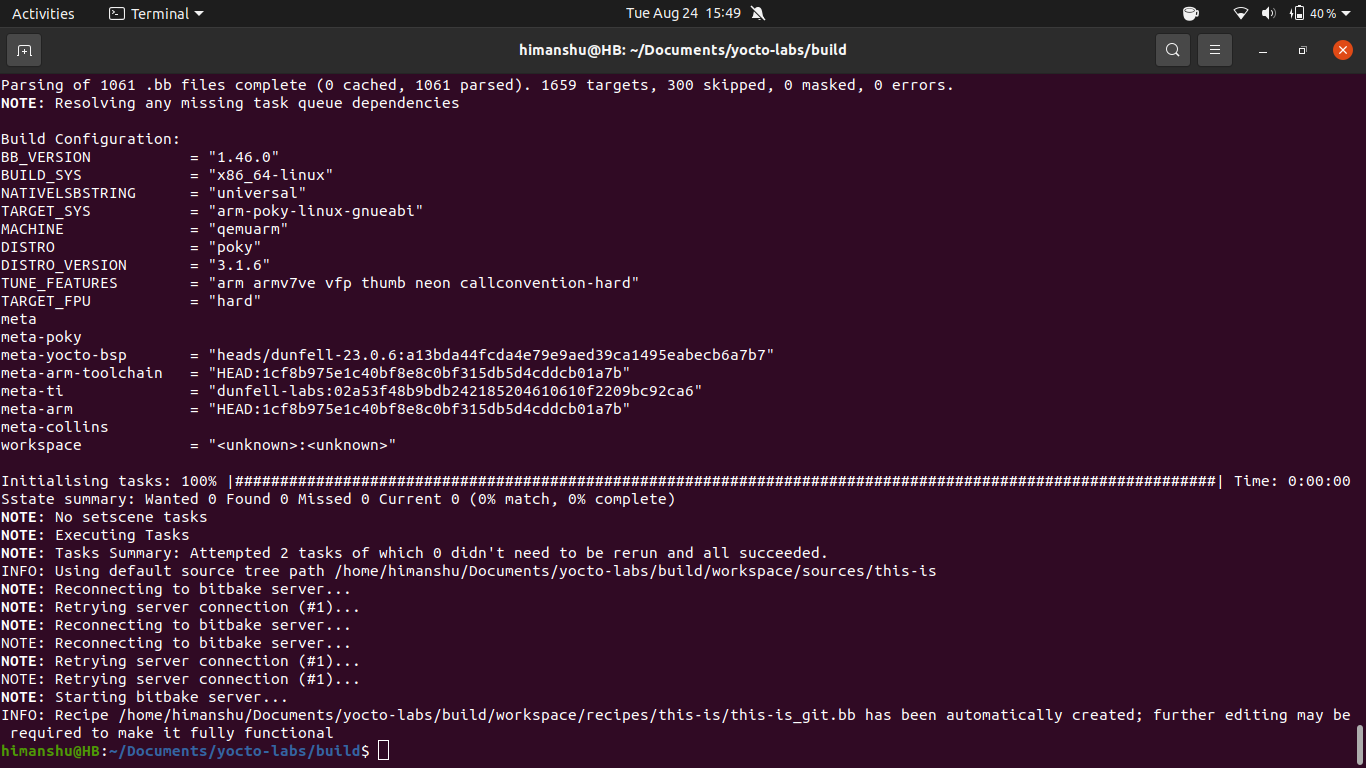




In the later section of the video he is demonstarting the recipe building techniques

devtool

devtool add <https://github.com/LetoThe2nd/this_is.git>



now build this recipe that you have git cloned

bitbake this-is

note: this is actually the manual building of the recipe so that doesnt mean it will be included in any of the image

the one thing that you can do with manual compilation of the recipe is you can check whether they are porperly compiled or not before adding them to your custom image

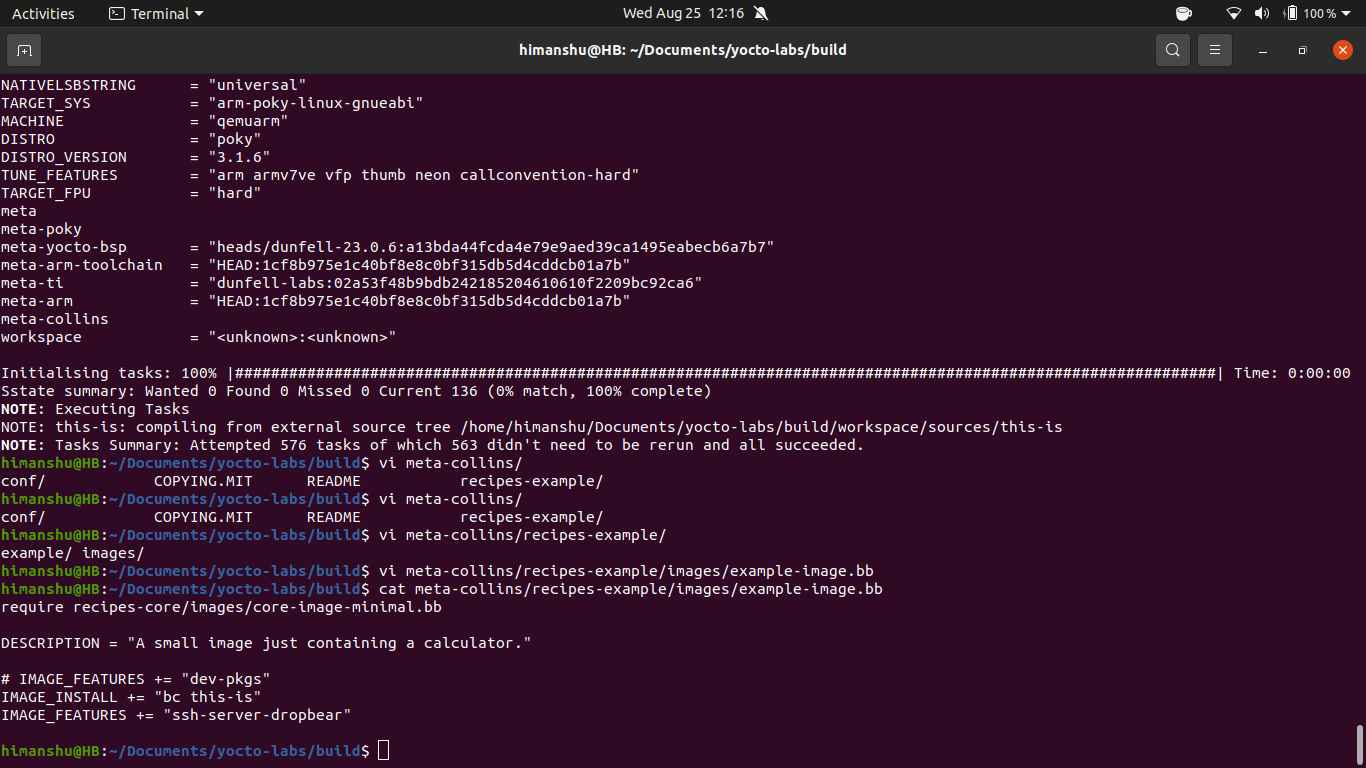
build/temp/work/<recipe\_name>/<recipe\_version>/package/usr/bin/\*<recipe\_name>

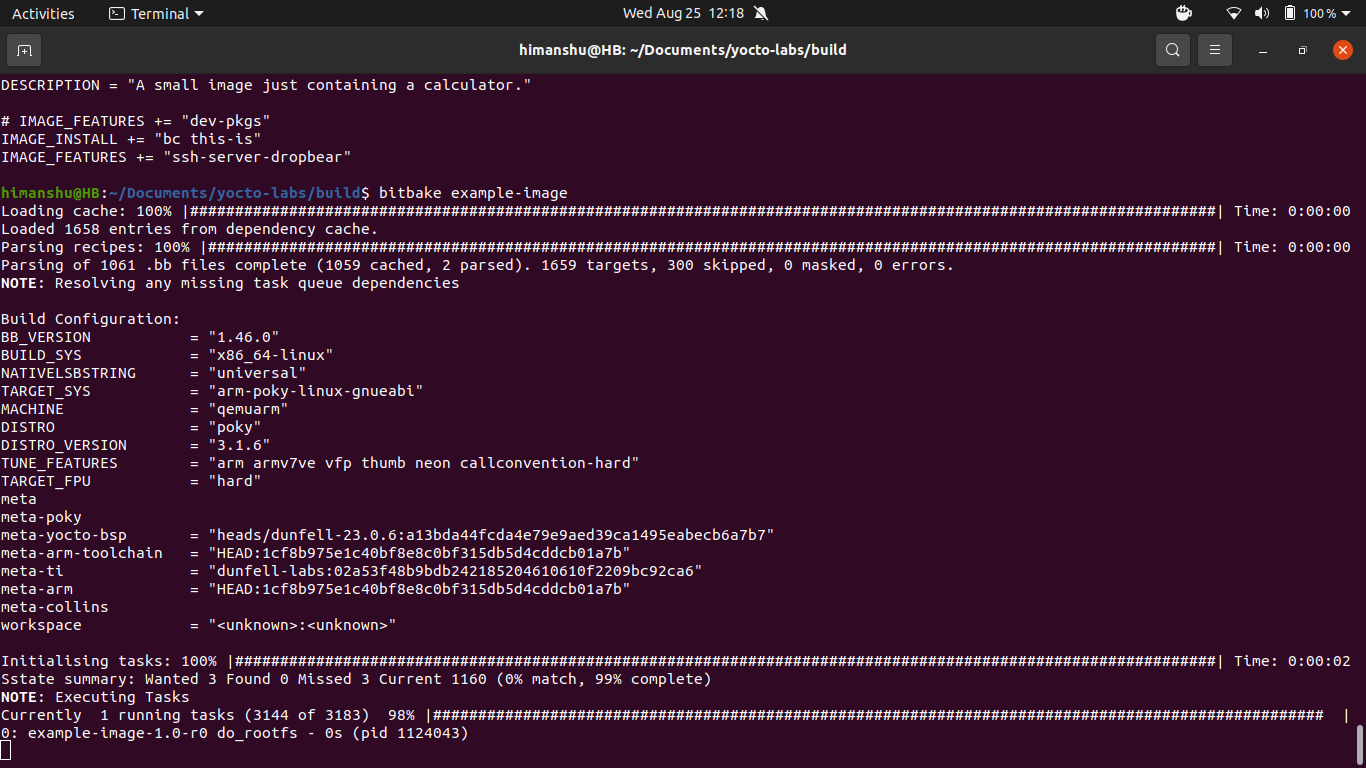
command - file \*<recipe\_name>

9. now in order to add that recipe to your image what we can do is

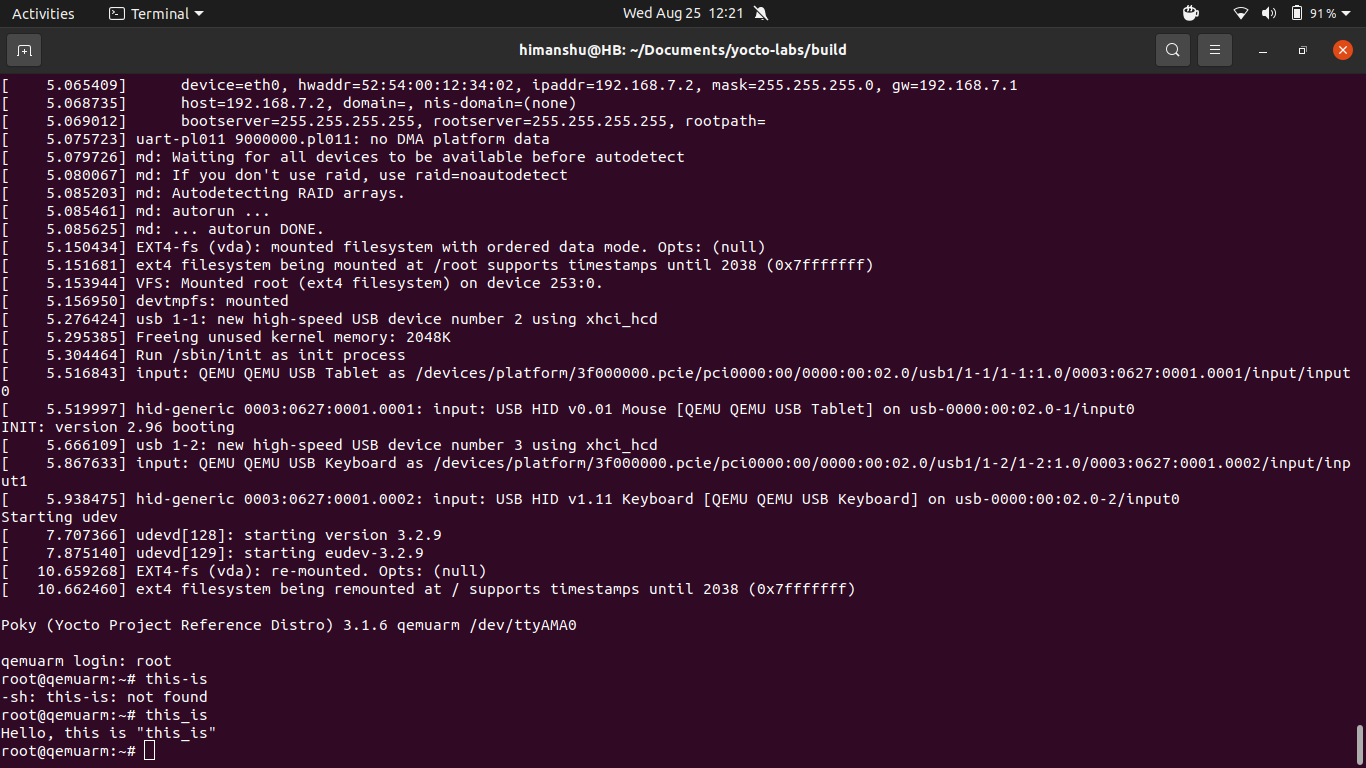
- open your own meta layer folder and edit that .bb file

meta-collins/recipe-example/images/meta-collins.bb





you can check that the recipe has been added to the image that was build



10. in order to edit any recipe

**devtool edit-recipe <recipe\_name>**

this above command will automatically find the recipe file(.bb) from the meta folders that you have included in the bblayer.conf (build/conf/bblayer.conf)

#3 lecture <https://www.youtube.com/watch?v=IehnEC3GOGU> package dependencies and spiltting

1. add the library

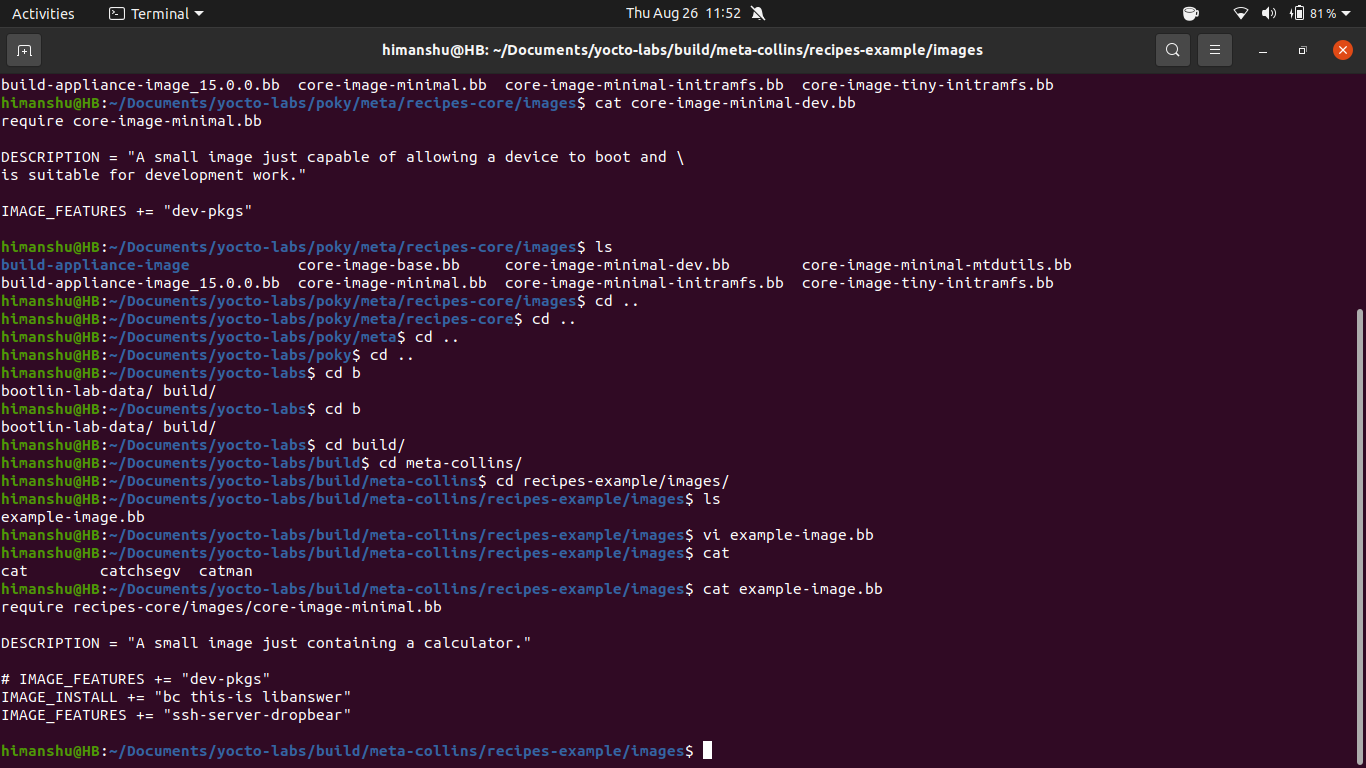
devtool add libanswer **https://github.com/LetoThe2nd/libanswer**

2. build the library

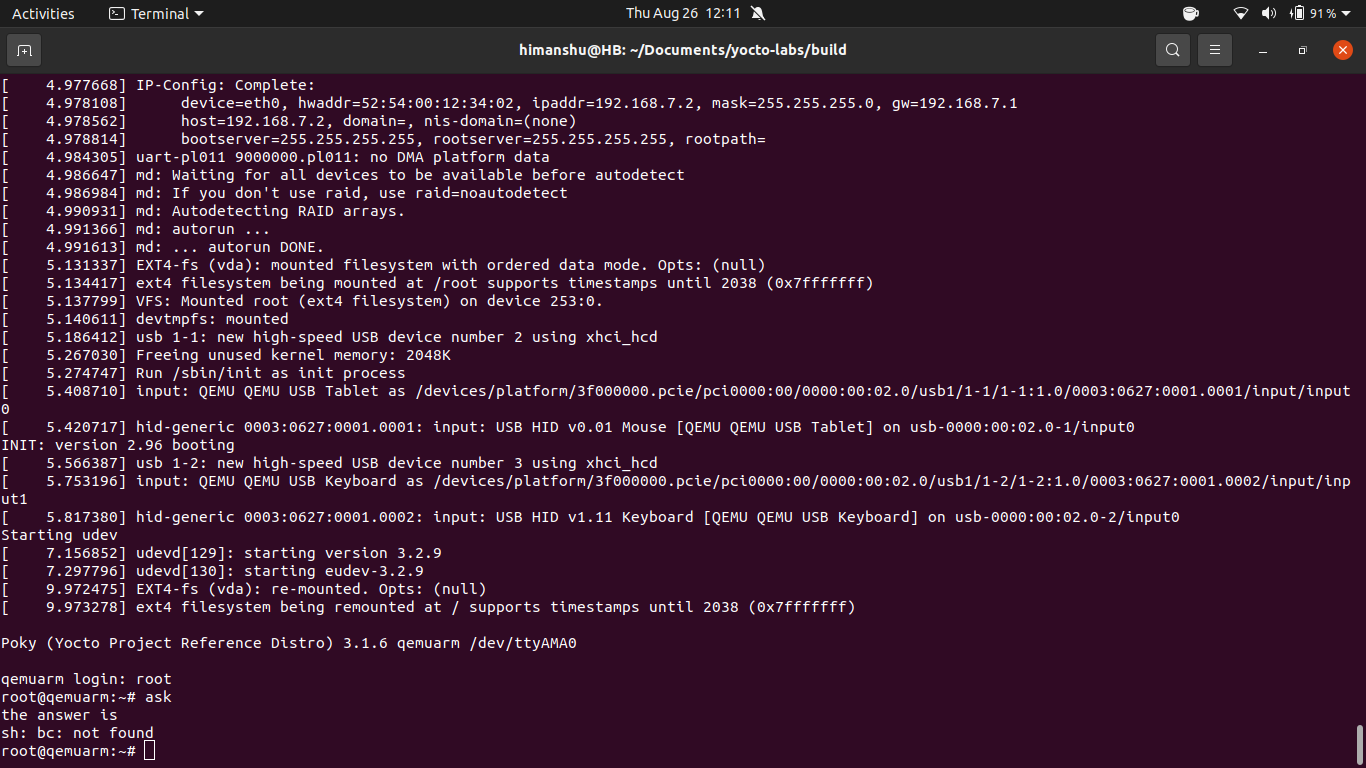
bitbake libanswer

3. add the package to your build by making changes to your recipe (exmaple-image.bb)

IMAGE\_INSTALL += “libanswer”

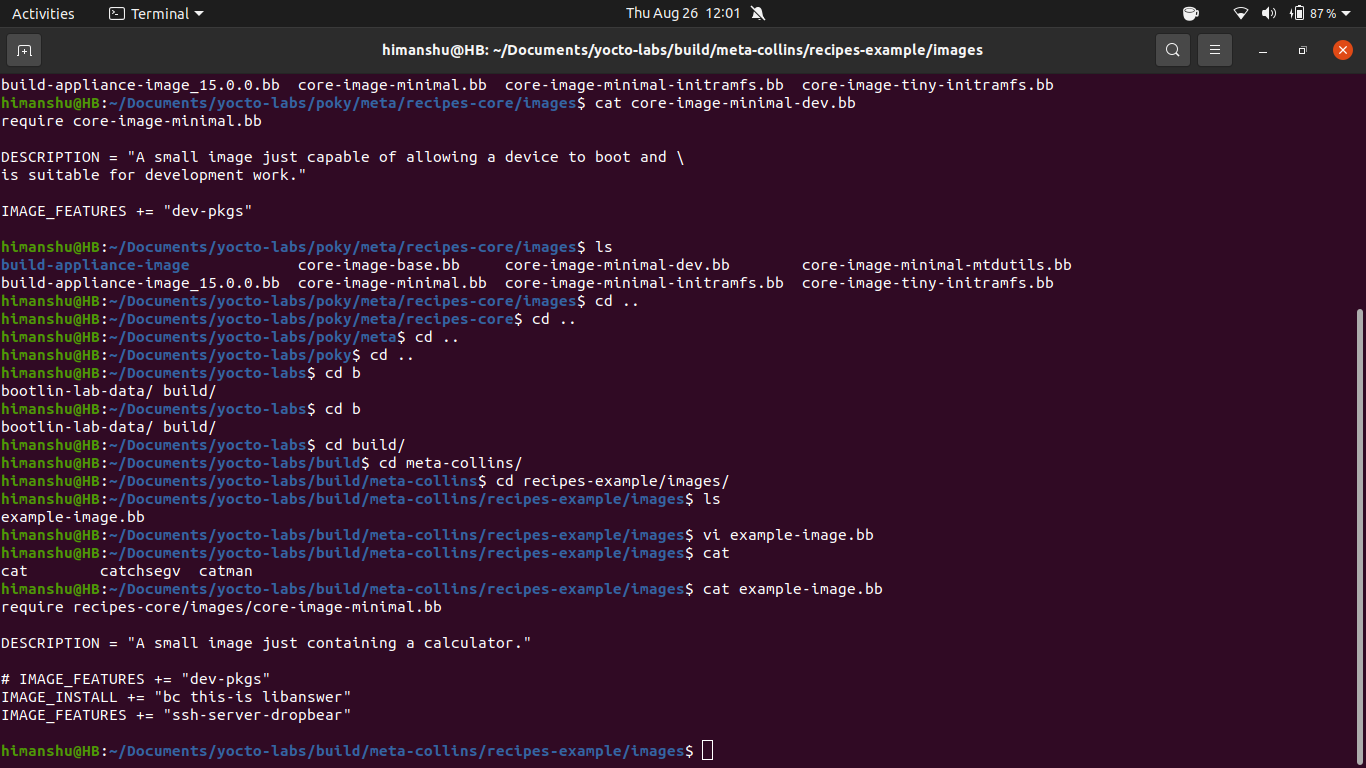


4. after compiling and running the compiled image type “ask” this will give you an run dependecnie error since it need “bc” package



there are two ways to solve this problem

a)you either I**MAGE\_INSTALL += “bc libanswer”** in your main recipe config(example here example-image.bb is our main recipe/image and we wanted the libanswer attached to this image)

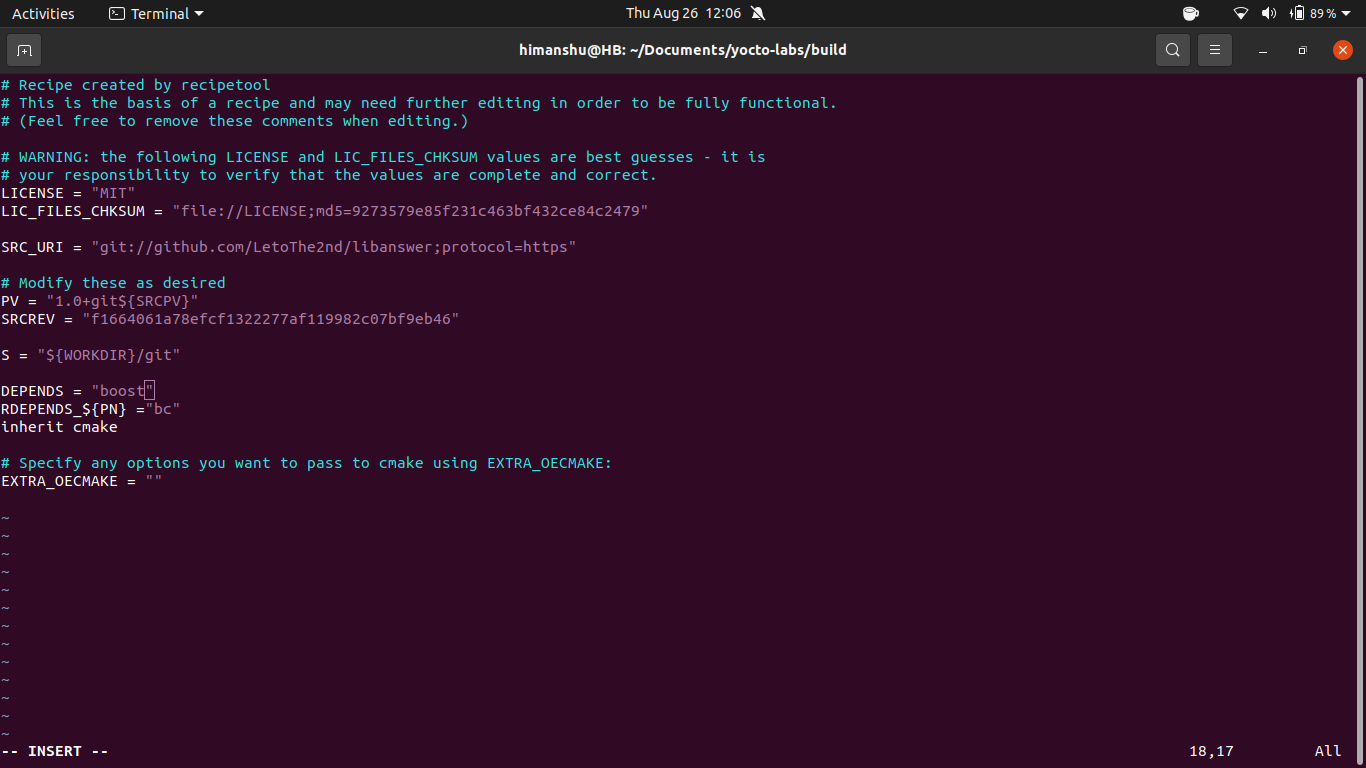


here you can see the bc package is included within the main image

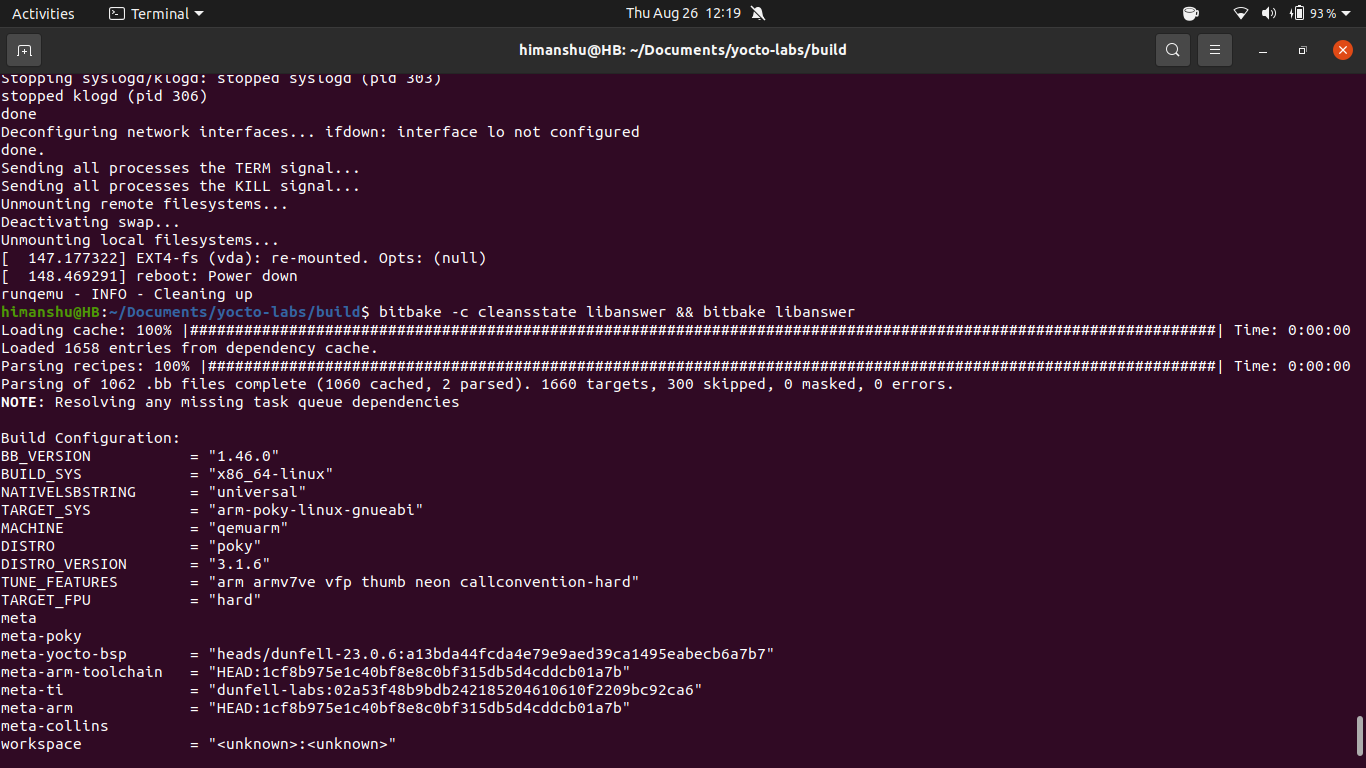
b) you can edit your libanswer config file and add that pc package within the libanswer by using

**devtool edit-recipe libanswer**

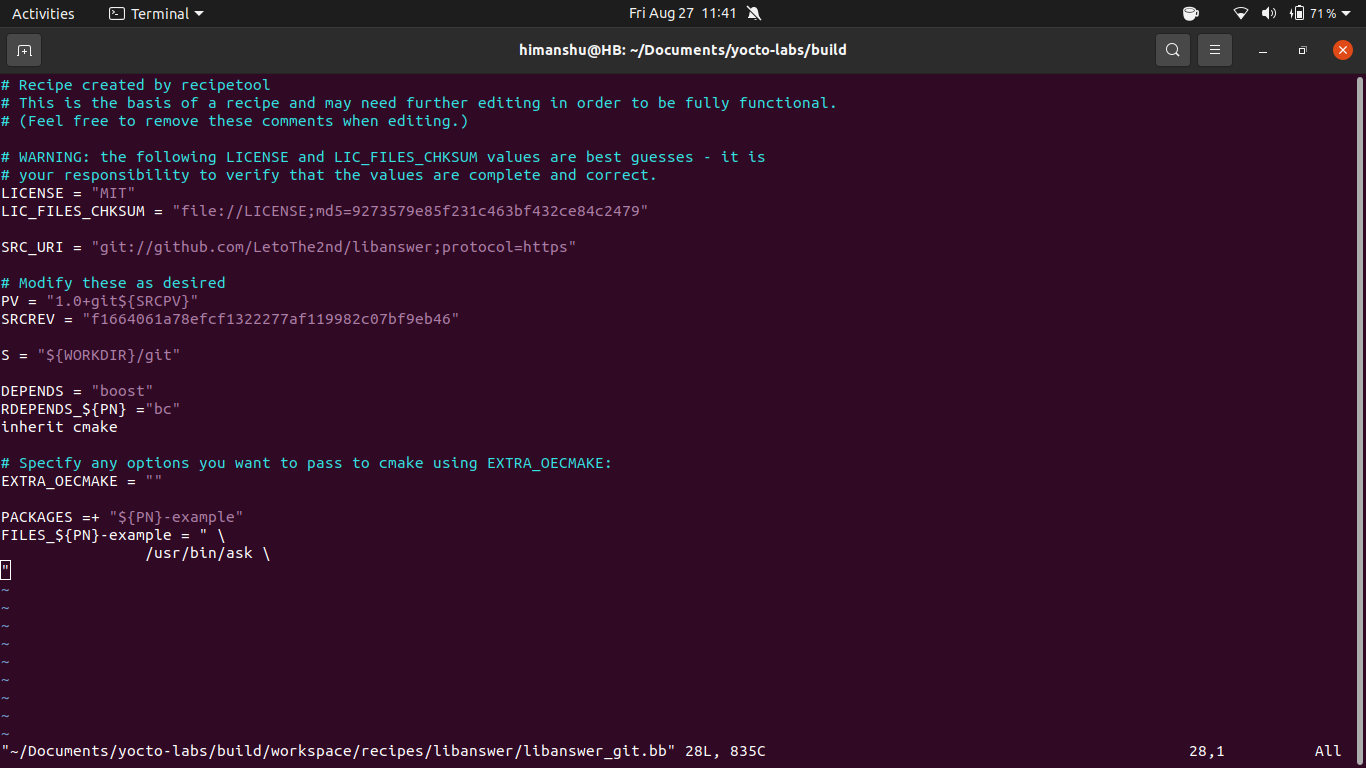
add the line **RDEPENDS = “bc”**



5. try to clean build

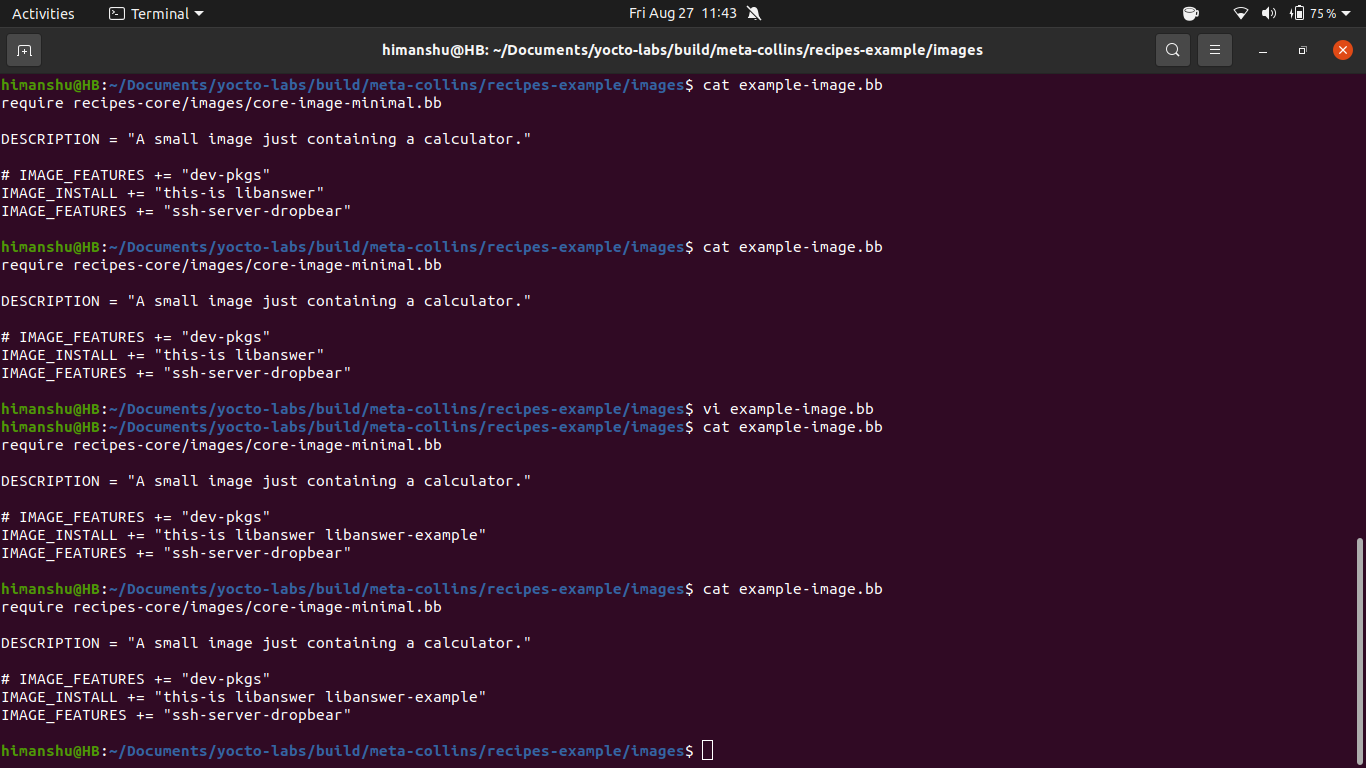
**bitbake -c cleansstate libanswer && bitbake libanswer**

6. split the package



this is canonical splitting

7. since you have splitted the package that means your have to add both the package name in your own recipe (example-image.bb)

**IMAGE\_INSTALL += “libanswer libanswer-example”**

**new task**

**this section will be about the 3 commonly used buses (I2c,pci,Uart)**

**https://www.youtube.com/watch?v=c10wAKWpjts**