**Cache busting**

Whenever an image is build from a dockerfile, docker reads its memory and checks which instructions were already executed.

These steps will not be re executed.

It will execute only the latest instructions. This is a time saving mechanism provided by docker.

But, the disadvantage is, we can end up installing software packages from a repository which is updated long time back.

# mkdir docker

# cd docker

# vim dockerfile

Lets just add one more instruction

FROM ubuntu

MAINTAINER logiclabs

RUN apt-get update

RUN apt-get install -y git

RUN apt-get install -y tree #added extra command

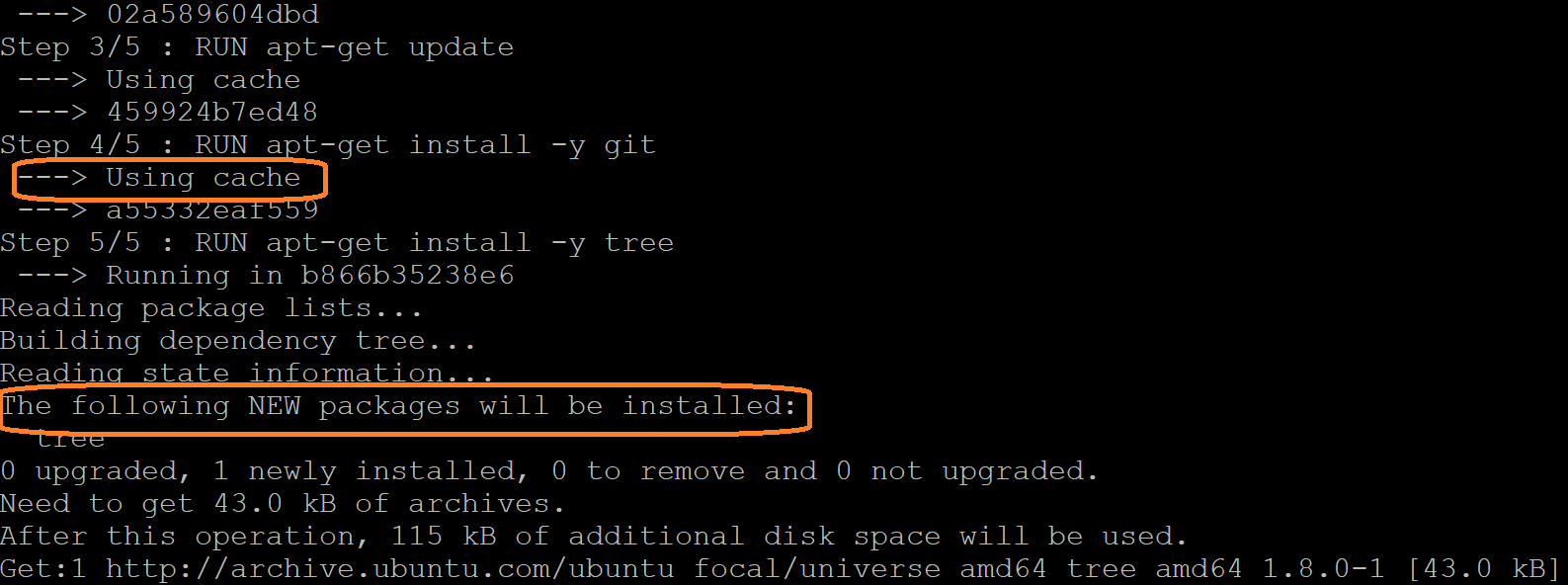
:wq

Lets build an image

# **docker build –t <Any image-name> .**

# **docker build -t myubuntu** .

Observe the output, Step 2, 3, 4 is using cache. Only step 5 is executed freshly



**Advantage:** time saving mechanism

**Disadvantage** : Lets say, you are running after 4 months, We are installing tree from apt which is updated long time back. )

**To avoid this disadvantage we use cache busting**

**Note**: cache busting is implemented **using && symbol**.

Which ever statement in the docker file **has && will be re-executed**.

# vim dockerfile

FROM ubuntu

MAINTAINER logiclabs

**RUN apt-get update && apt-get install -y git tree**

:wq

Lets build an image

**# docker build -t myubuntu .**

**Observe** the output, step 3 - It is not using cache