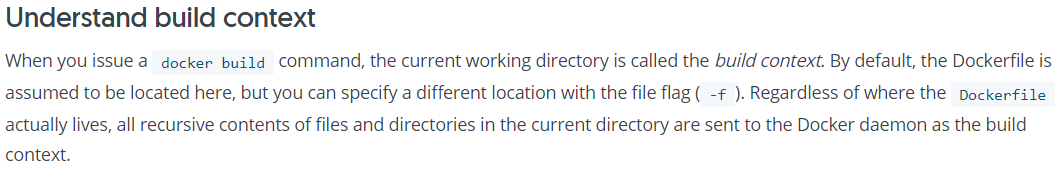
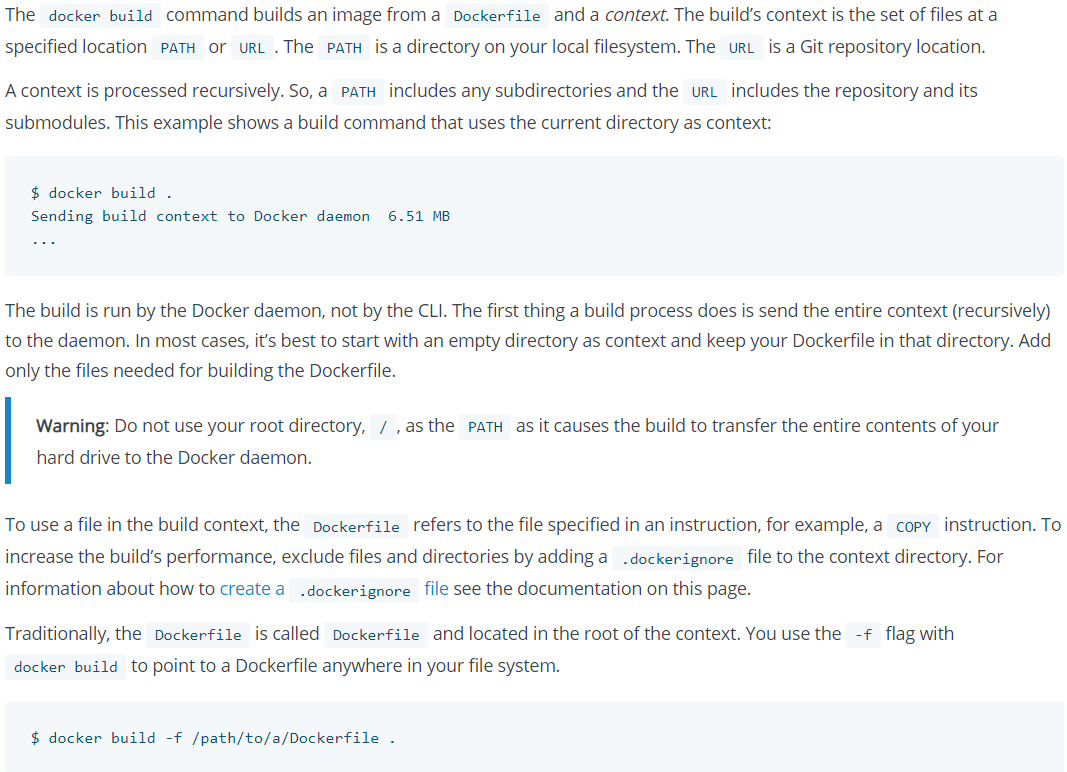
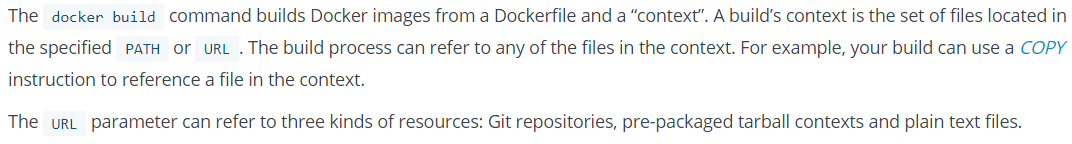
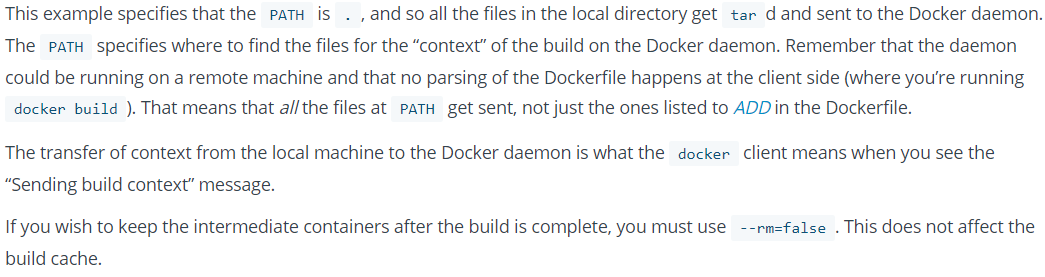
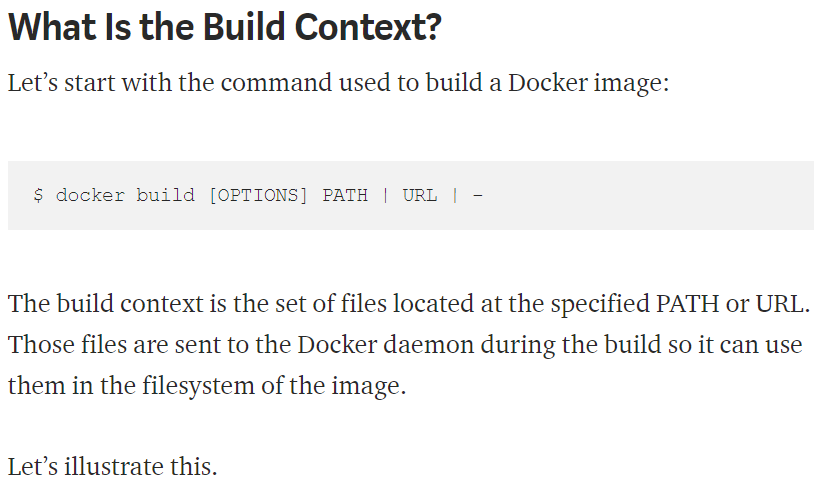
**Docker Build Context:**

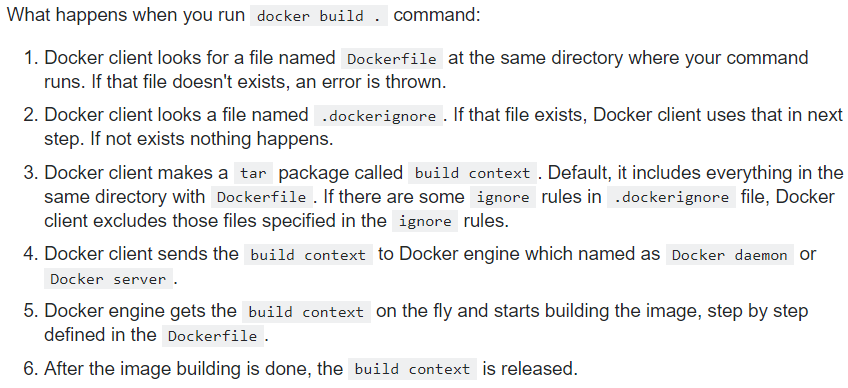


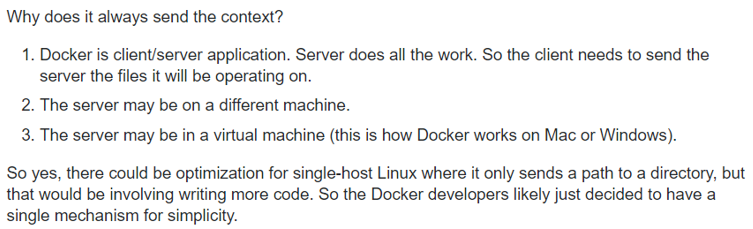
****



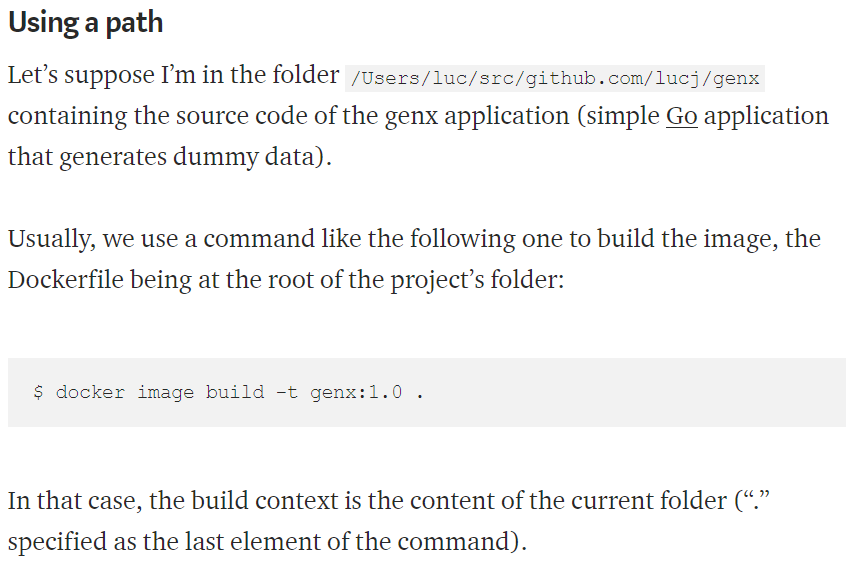




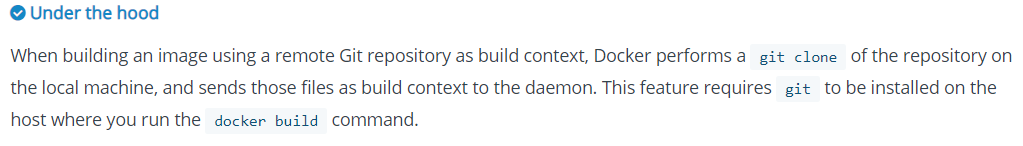




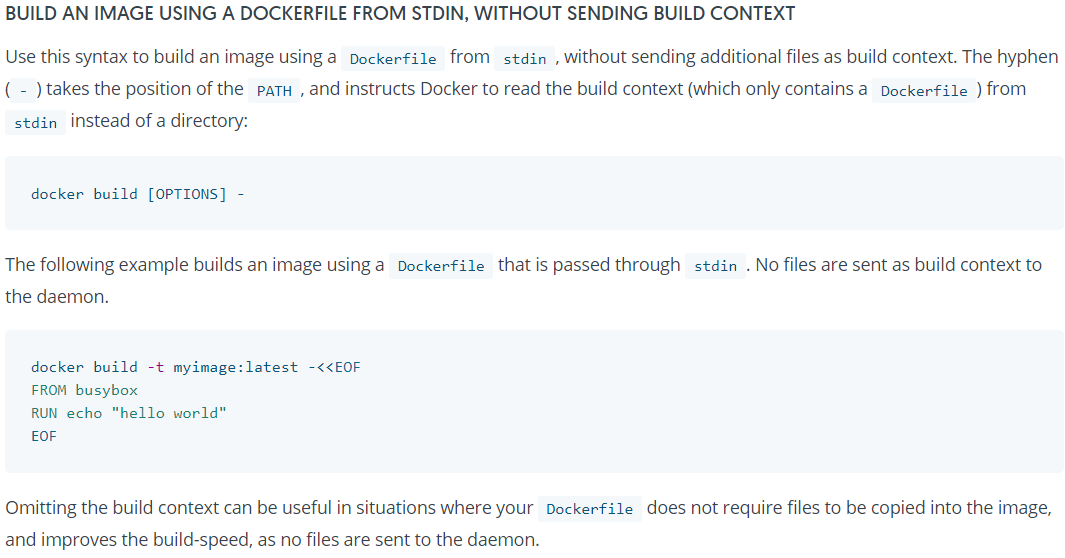
**Differenct ways or building docker image with context:**







**Building image without context using STDIN:**



**We can’t use COPY or ADD command in dockerfile if the same context is not present in docker daemon**

**Example:**



**Build using dockerfile without sending the context:**

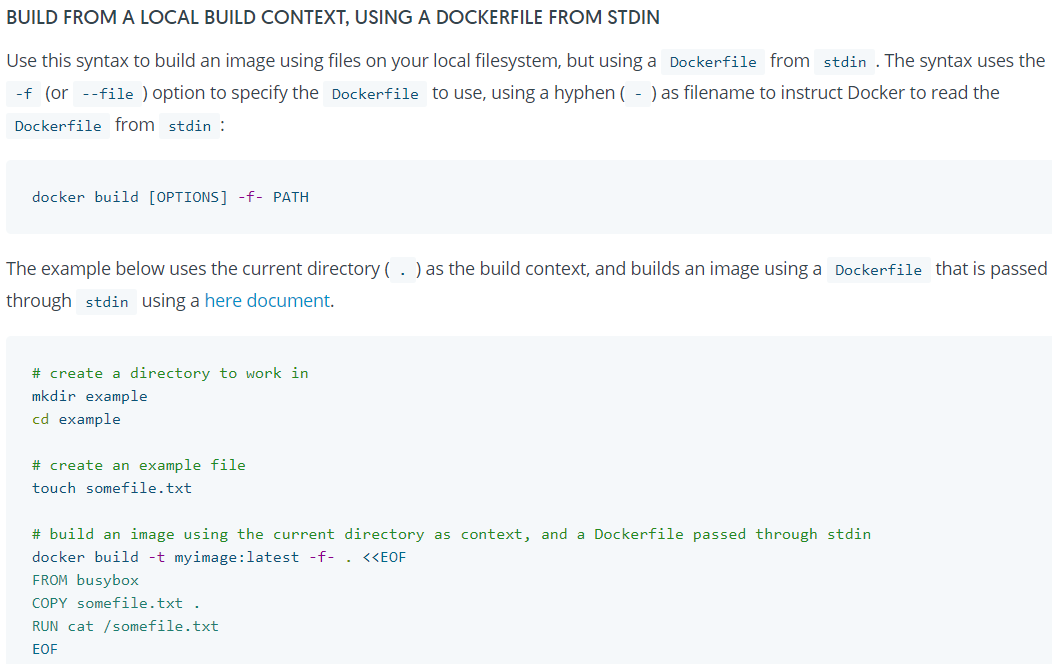
* **Docker build - < (dockerfile name)**

With this you can also give non-default dockerfile name

Here you can’t use (.) to build the image. You can either use full path of dockerfile with the name or you can give the path from current directory

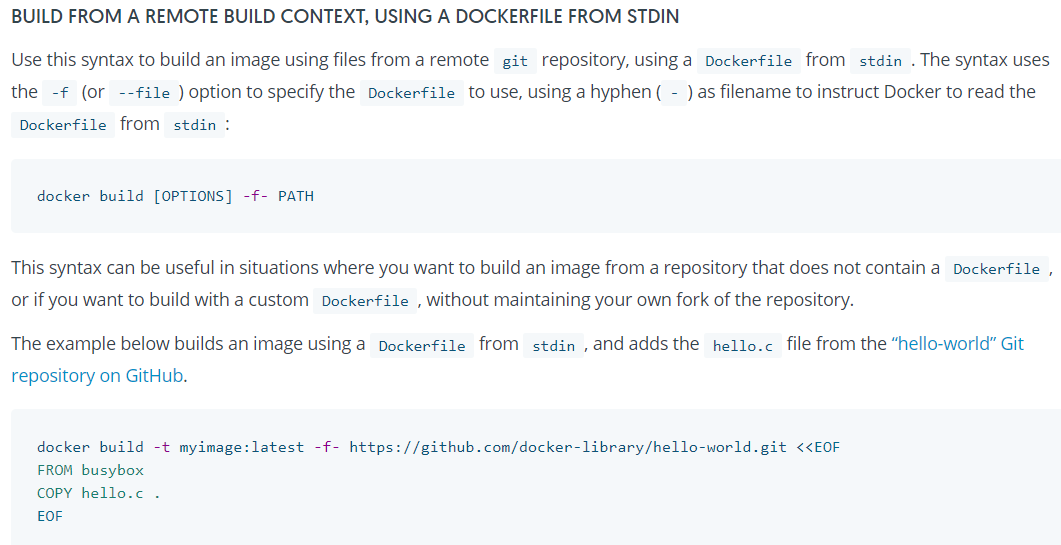
**Even if we build the image with “docker build –“ command, docker still sends some 2KB data to the daemon process which could be some meta data**

**Build from a local context using docker file in STDIN:**



As above, the (.) is the path of the local context

**Build from remote context using dockerfile from STDIN:**



Building the image with non-default docker file or using the different build context path:

* **Docker build -t <tag> -f <dockerfile path> <context path>**

With the above. We need to pass the nondefault dockerfile name or full path and the context path details

This feature was added in docker 18.03.0-ce

**Note:**

If the context path is in the current working directory, then we can use (.) but we can’t use (.) for dockerfile even if it’s in current working directory. We need to give either full path with dockerfile name or at least the docker file name.

**Example:**

If ~/Akhil is the current directory where the dockerfile exits. Then you can use any of the below commands

* **Docker build -t <tagname> -f ~/Akhil/<dockerfile name> <context path>**
* **Docker build -t <tag name> -f <dockerfile name> <context path>**

**Note:**

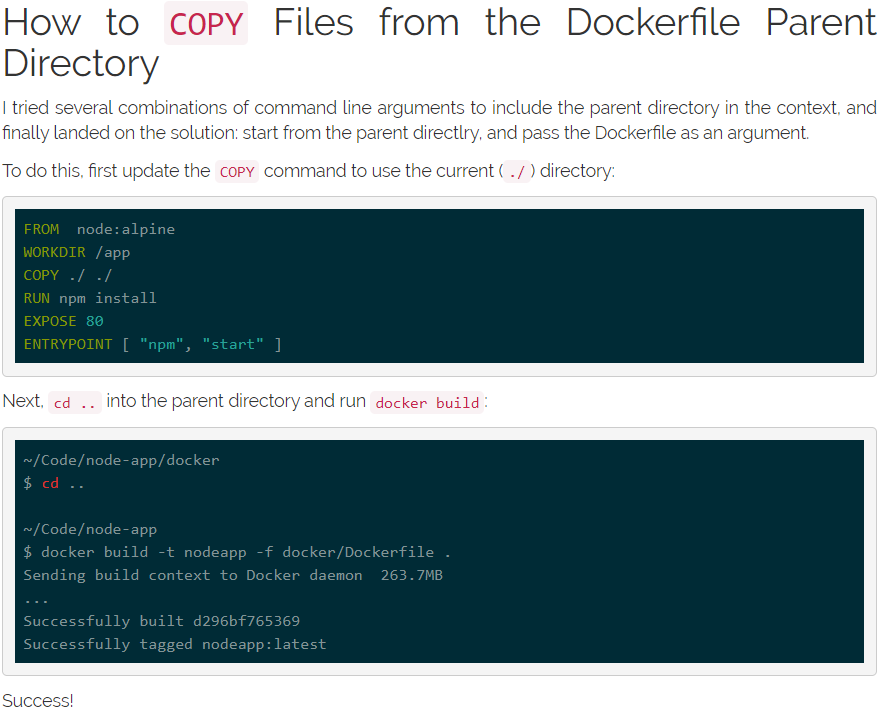
this method we can also use if we have dockerfile with different name

**Example:**

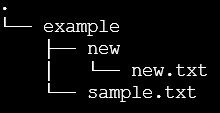
* **docker build -t new -f akhil/dockerfile.txt ~/example**
* **docker build -t new -f ~/akhil/dockerfile.txt ~/example**
* **docker build -t image -f dockerfile.txt ../example/**

as above commands, if we are in ~ directory, we can either give the dockerfile path from the current path as shown in first command or you can use the full path as second command

the same way it works for context path also as per the third command. You can either use (.) or the full path or the path from the current directory



**context path in docker daemon:**

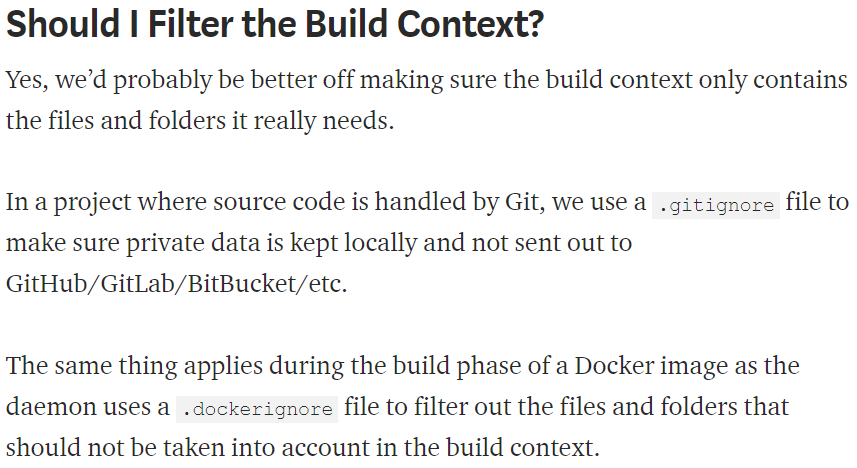


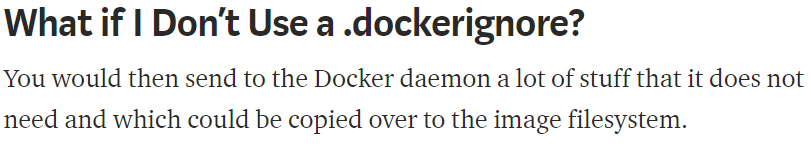
Let’s say this the folder structure we have, and if we push ~/example/new context to the docker daemon. Then the docker daemon would be containing only “new.txt” file. Id doesn’t contain the folder structure

So, if we want to use COPY or ADD command to use the “new.txt” file, we need to just give the filename in dockerfile not the full path of the file in our local system which docker doesn’t have

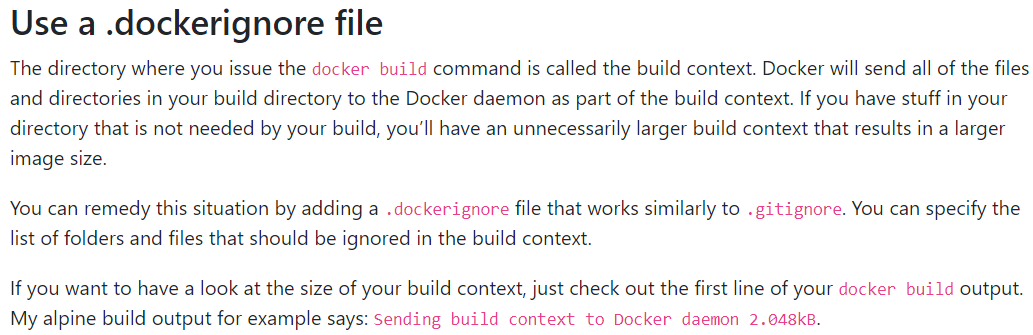
**Filter the build context:**

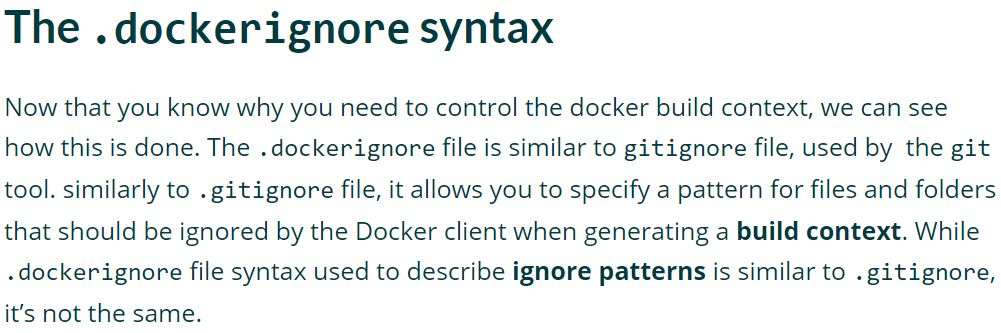
If you want to improve the build-speed by excluding some files from the build- context, we can also use .dockerignore file

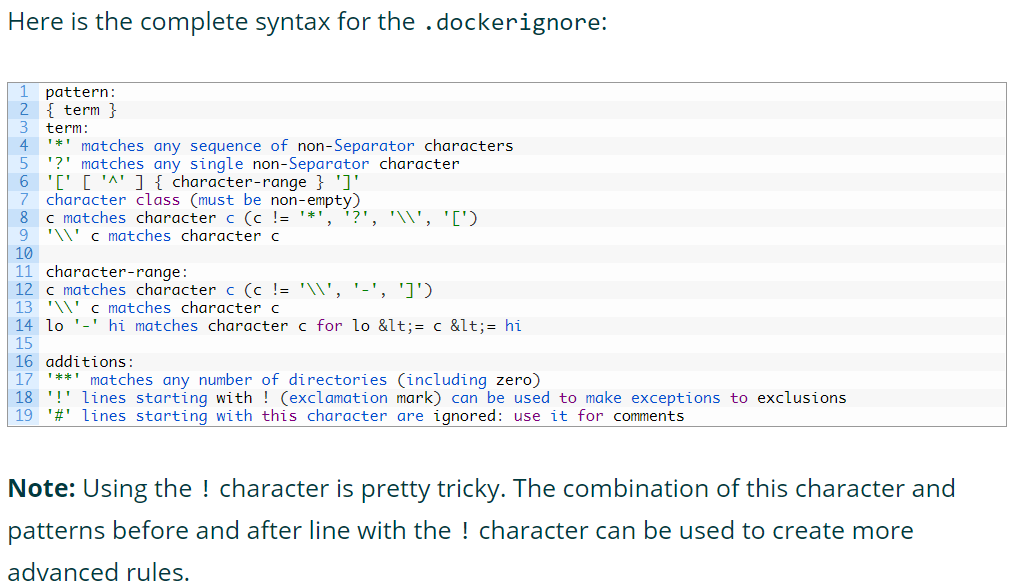


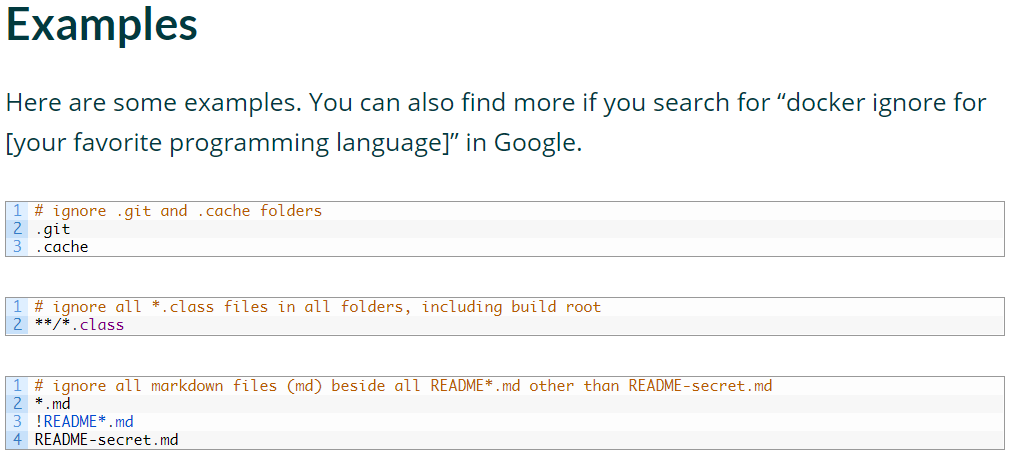


**Dockerignore file:**



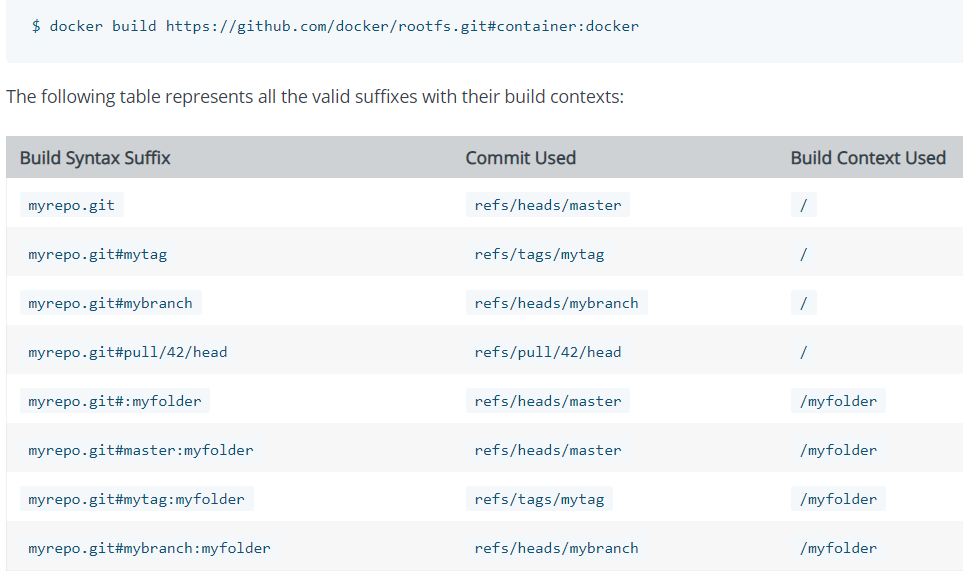






**Build options:**

* Below are the options we can use while building the image from git URL.



* More options in build command as below.

|  |  |  |
| --- | --- | --- |
| --file , -f |  | Name of the Dockerfile (Default is ‘PATH/Dockerfile’) |
| --force-rm |  | Always remove intermediate containers |
| --label |  | Set metadata for an image |
| --memory , -m |  | Memory limit |
| --rm | true | Remove intermediate containers after a successful build |
| --tag , -t |  | Name and optionally a tag in the ‘name:tag’ format |
| --target |  | Set the target build stage to build. |