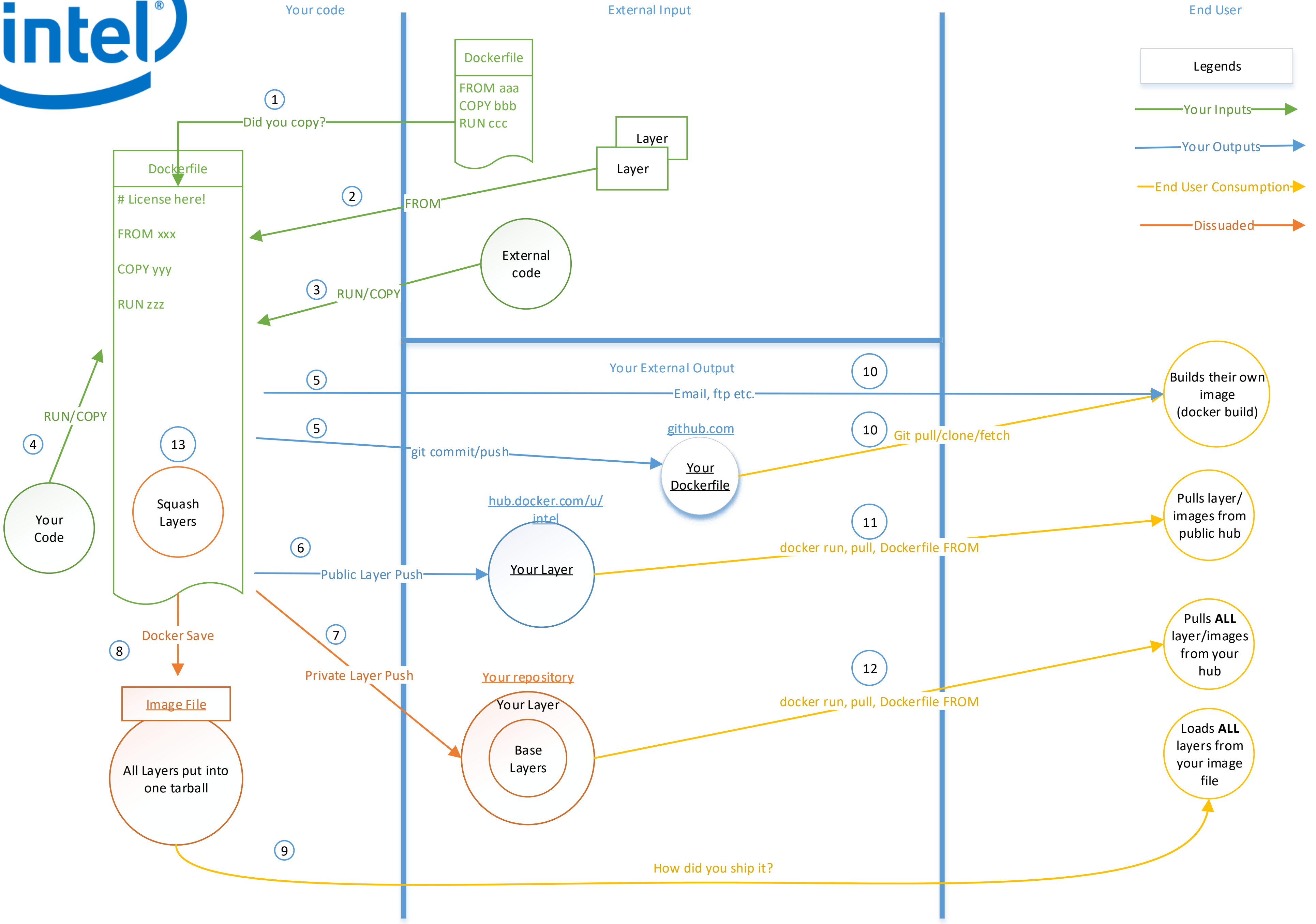

Your code
External Input
End User
Legends
Your Inputs
Your Outputs
End User Consumption
Dissuaded
1
Did you copy?
Dockerfile
FROM aaa
COPY bbb
RUN ccc
Layer
Layer
2
FROM
Dockerfile
# License here!
FROM xxx
COPY yyy
RUN zzz
3
RUN/COPY
External code
4
RUN/COPY
Your Code
13
Squash Layers
5
Your External Output
Email, ftp etc.
10
Builds their own image (docker build)
5
git commit/push
github.com
Your Dockerfile
10
Git pull/clone/fetch
hub.docker.com/u/intel
11
docker run, pull, Dockerfile FROM
Pulls layer/ images from public hub
6
Public Layer Push
Your Layer
7
Private Layer Push
8
Docker Save
Image File
All Layers put into one tarball
Your repository
Your Layer
Base Layers
12
docker run, pull, Dockerfile FROM
Pulls ALL layer/images from your hub
Loads ALL layers from your image file
9
How did you ship it?

(1) Dockerfile provenance
Did you create your DockerFile from scratch, or did you reference or copy an existing DockerFile?

If you referenced or copied an existing DockerFile, did that DockerFile container a licence?

Ensure you know the provenance of your DockerFile contents.
Ensure you comply with any inbound license obligations.
License your Dockerfile, especially if you are shipping it.

(2) FROM base layer/images
Understand where your base image layers come from.
Understand what base distro your base image is derived from.
Confirm that your base layers are acceptable to the OSPDT and security teams.
Check that your base layers are ideally from Docker Official Images.
Does your base layer come with any obligations, such as attributions or providing source code?

(3) Importing code from outside Intel?

Did any of your FROM or COPY lines import code that originates outside of Intel?

Ensure you understand what code you imported, from where.
Ensure you understand any license implications of that code.
Ensure you list that code in your IP Plan if applicable
Ensure you list that code in your OS PDT slide deck if applicable.
Ensure if you download code into your layers, you use a secure channel to download and/or check the validity of SHA/checksums if available.

(4) Importing code from Inside Intel

Ensure you understand if all the imported code is from your project, or imported from other parts of intel as well.
Ensure all of the code appears in your OS PDT slide deck, as appropriate.
If you are importing code from another project, ensure that project already has an OSPDT approval, or be prepared to roll that code into your project for approval.

(5) Are you releasing your DockerFile

Are you shipping your DockerFile (for instance, by pushing to GitHub, or supplying to end users via any other method such as email, ftp etc.)?

Ensure your DockerFile containers a license.
If you imported or copied any of your Dockerfile from elsewhere (see (1)), ensure that your (re)license is compatible with all such artifacts.
(NB: Call out here what you need for 'full approval')

(6) Publishing your layer to an existing image hub/registry

Are you publishing your layer to an existing public repository, e.g. `docker push` to `hub.docker.com`.

Ensure you understand everything that is in **your layers** (the layers you authored in your DockerFile).

(legal point, `docker push` actually pushes *all* the layers of your image to the registry to prove it has access to them – if the layers already exist on the registry, after verification, the data is then discarded. Thus, you are not really 'publishing' the base layers, per se)
(NB: Call out here what you need for 'full approval')

(7) Publishing your layer to your own image hub/registry (dissuaded)

Similar to (6), but in this case you **are** publishing all the layers in your image, including all layers obtained from outside sources (such as in (2) and (3)).

Ensure you have the legal rights to publish these layers.
Ensure you meet any legal obligations that come with these layers.
(NB: Call out here what you need for 'full approval')

(8) Publishing a (full) docker image (dissuaded)

If you are publishing a full docker image (not just a docker layer) – for instance via 'docker save', to a tarfile, then that image file will contain **all** the layers of your image, including those obtained in (2) and (3). You are therefore publishing and shipping those layers.

Ensure you have the legal rights to publish these layers.
Ensure you meet any legal obligations that come with these layers.
(NB: Call out here what you need for 'full approval')

(9) Customer imports your image

When a customer imports your image (as created in (8)), they will import all data for all layers will have come **from you**.
Ensure all obligations in (8) have been met.

(10) Customer uses your DockerFile

When a customer obtains your DockerFile (no matter where from), and creates their local layers/image, their local build will re-create these from the original sources and layers.

As per (5):
Ensure your DockerFile has a license.
Do not supply DockerFiles that create layers or images that may contravene licenses and copyrights.

(11) Customer uses your layer from public registry

All dependant layers will be pulled from the public repository. You are neither publishing nor modifying these layers.
(6) should have ensured your layer is OK for publication.

(12) Customer uses your layer from your registry

All dependant layers will be pulled from your repository. You are effectively publishing and shipping all of these layers.

You will have responsibilities to maintain (e.g. security updates) for all of these layers, as per (7).

(13) Compressing layers

Do not compress layers you have generated with layers obtained from base images. Do not compress base layers together.

Compressing layers that are not 'yours' may bring extra components into your layers (such as a complete distribution), or have licensing implications and obligations.

You can compress layers you have generated.