For those who used to work with the docker image build process, we've migrated docker image build job from Jenkins to CircleCI.

Current config

The config of docker image now live at https://github.com/pytorch/pytorch/tree/master/.circleci/docker and you can find existing images (both permanent and weekly) at http://docker.pytorch.org which will be updated hourly by this job: https://github.com/pytorch/pytorch/blob/988ef190e3df0b48ef59dcf8007258ed79bb7946/.circleci/config.yml#L2038-L2093

Q & A

How to trigger new build process if I don't want to wait for a week?

New images are automatically built when any changes are committed to `.circleci/docker`. Developer images are retained for 2 weeks, so if your builds are older than 2 weeks you will need to rebuild your images if you intend to merge the PR associated with docker image changes.

New images will automatically be passed down through to dependent jobs.

Where to find tags for newly created/pushed images?

http://docker.pytorch.org which will be updated hourly

How to docker pull the docker images?

Note that the docker images are built within a private AWS ECR repository, the detailed instruction of how to pull those images can be found here: https://github.com/pytorch/ossci-job-dsl#ci-failed-but-my-local-build-is-fine-what-should-i-do.

How to reproduce build and test with the local docker image?

Right now, since CI has so many different configuration and environment variables settings, it's hard to have 100% docker reproducible build and tests, so we still recommend you rely on CI first (as of 08/12/2021). If you really want to reproduce the docker based `build` and `test`, here's the steps you can follow. Make sure you have the ECR credentials (see instruction above), and you have already pulled the base docker images, you can try to reproduce what the GitHub Action CI does.

For example:

- build step: https://github.com/pytorch/pytorch/blob/219ba6575b682a9b61476da041c2220142d20e3b/.github/workflows/generated-linux-xenial-py3.6-gcc5.4.yml#L140-L159
- test step: https://github.com/pytorch/pytorch/blob/219ba6575b682a9b61476da041c2220142d20e3b/.github/workflows/generated-linux-xenial-py3.6-gcc5.4.yml#L327-L351

How do we purge old images / what's the retention policy?

We have ecr_gc_job job (you can search for it in config.yml) that runs every hour to purge old images. Currently, we need temporary images for 1 day, and weekly builds for 2 weeks. And we will keep image with tags defined in https://github.com/pytorch/pytorch/blob/master/.circleci/verbatim-sources/workflows-ecr-gc.yml#L13 forever. code for the purge job is https://github.com/pytorch/pytorch/blob/master/.circleci/ecr_gc_docker/gc.py

The production images disappeared, what should I do?

Sometimes there is a bug in ecr_gc_job and it deletes Docker images it shouldn't. All Docker images are also saved to S3 with a one-month retention period, so there's a chance they may still be there. You can use this script to recover in that case:

```
import yaml
import boto3
import subprocess
import os
import requests
import argparse
import multiprocessing

parser = argparse.ArgumentParser(description='Recover Docker images from S3 to ECR')
parser.add_argument('tag', metavar='TAG', help='tag to recover, something like 07597f23-fa81-474c-8bef-
5c8a91b50595')
parser.add_argument('-j', metavar='N', type=int, default=8, help='number of jobs to run in parallel')
args = parser.parse_args()
recover_id = args.tag

r =
yaml.safe_load(requests.get('https://raw.githubusercontent.com/pytorch/pytorch/master/.circleci/config.yml').text)

builds = [b['docker_build_job']['image_name'] for b in r['workflows']['docker_build']['jobs'] if isinstance(b,
```

```
dict)]

def image_name(b):
    return "{}:{}".format(b, recover_id)

def s3_url(b):
    return "pytorch/base/{}.tar".format(image_name(b))

def upload_to_ecr(b):
    print(b)
    s3 = boto3.client('s3')
    print("s3 url: {}".format(s3_url(b)))
    tmp_file = '/tmp/{}.tar'.format(image_name(b))
    s3.download_file('ossci-linux-build', s3_url(b), tmp_file)
    print("loading...")
    subprocess.check_call(('docker', 'load', '--input', tmp_file))
    print("pushing...")
    subprocess.check_call(('docker', 'push', '308535385114.dkr.ecr.us-east-
1.amazonaws.com/pytorch/{}'.format(image_name(b))))
    print("done.")
    print()

pool = multiprocessing.Pool(processes=args.j)
pool.map(upload_to_ecr, builds)
```

How to add a new base docker image?

Note this instruction provides guidance to add a new **base** docker image. If you could reuse one of available docker images listed in https://github.com/pytorch/pytorch/blob/master/.circleci/docker/build.sh#L80, please do so and not adding new ones.

- Add an entry in https://github.com/pytorch/pytorch/blob/master/.circleci/docker/build.sh#L80 and make changes to Dockerfiles accordingly.
- Test your image by building it locally.
- Add a repo in AWS ECR to hold your image. It requires access to PyTorch's AWS account to do this step.
- Trigger a new build process as described above.
- **WAIT UNTIL ALL BUILD JOBS TO FINISH** and make sure all new images have been uploaded. Save the tag of the new images.
- Run regenerated.sh with the new tag and update your PR.

See an example PR https://github.com/pytorch/pytorch/pull/36187