PyTorch Enterprise Support Program

Certification Guide

Last Updated: May 6, 2021

This document sets forth the conformance testing instructions and requirements for the PyTorch Enterprise Support Program

Capitalized terms used herein and not otherwise defined shall have the same meanings set forth in the PyTorch Enterprise Support Program Terms.

Participant can provide input for improvements of this Certification Guide to Facebook. Facebook will be the only one who can make changes to this Certification Guide as appropriate and after consulting with Participants and providing 30 days' notice.

Any communication among Participants and with Facebook is encouraged preferably through the Point of Contact for PTE Matters named by the Participant in the PyTorch Enterprise Participation Form.

Tests

Create, maintain, and submit to PyTorch Maintainers, for approval, a representative set of PyTorch workload regression tests

a. PyTorch workload regression test should exercise end-to-end model training, inferencing and/or deployment process for the models that Participant uses or supports in production.
b. The workload regression test should capture failures in the areas of API compatibility, performance regression and model training stability, model inference latency.
c. Participants are required to establish a Continuous Integration (CI) system that:
    i. Runs when a pull request is submitted to the Pytorch Github repository
    ii. Validates the pull request against the latest master branch of Pytorch by running the internal workload.
    iii. Publishes the results of that pull request to a publicly visible location.
d. Participants are required to do one of the following for one or more of the Pytorch domain libraries (like torchvision, torchtext, or torchaudio) and community libraries (like Huggingface or Torch Lightning):
    i. Establish a CI system similar to the one for the core Pytorch library (as mentioned in 1c above) OR

        ii. Commit to contributing new code or tests to the domain libraries every release.
    e. In addition to required tests above, participants may create optional tests to improve coverage over time.

Publish

1. Publish results of testing Qualifying Offerings against Participant workloads.
    a. The CI coverage should be implemented with open source and test execution that can be run by other developers outside the Participant organization. As an example, GitHub Actions can be used to execute the tests on AWS or Azure or GCP runners and developers can reproduce the results independently on a development environment created on one of those clouds.
    b. Alternatively, if the results are internal and cannot be run in open source, Participant can implement a build status indicator for workload regression test in PyTorch github. E.g. similar to "Facebook Internal CI passed" on current pull requests to PyTorch.
2. Fully automate CI/CD of PyTorch releases made as part of a Qualifying Offering
3. Create and maintain a web presence declaring their Qualifying Offerings and asserting compliance with the Support Program requirements.

Subsequent Testing

1. Participant will commit to run daily test workloads on at least all the following configurations:
    a. Two versions of CUDA
    b. Two versions of Python
    c. Windows and Linux
2. Every time a failure is detected by the internal workload regression test, Participant will create standalone isolated tests that reproduce it and submit PRs with these standalone tests to the PyTorch master repo. This will ensure that over time OSS coverage of PyTorch grows to capture scenarios important for Enterprise workloads.

Distribution

1. All PyTorch distributions made available as part of a Qualifying Offering must be derived from the then-current PyTorch release.
2. Participant may submit Hotfixes from time to time, after triaging issues and determining which issues they deem important enough to fix in Qualifying Offering, based on (1) the severity of the issue; (2) the scope of the change required; (3) input from community

including Facebook PyTorch teams; and (4) input from their commercial customers who may request to expedite fixes for issues that are impacting business objectives.. A “Hotfix” for purposes of this Certification Guide, is a code change that addresses a high priority issue in PyTorch (e.g. a bug fix) but does not introduce any functionality not available in the then current PyTorch release. Participant will do as follows for Hotfix approval:

a. Participant creates a set of PRs for the Hotfix. One PR must be produced for each affected supported version of Qualifying Offering. Participant then requests approval from the Facebook PyTorch team.
b. Facebook will provide a Point of Contact for release activities.
c. Facebook has 3 US business days to review the Qualifying Offering PR and either approve or reject it. By the end of the 3 day period, in the absence of a decision from Facebook, the PR is deemed to be approved. Facebook can refuse the PR only for one of the following reasons and in that case it will provide a justification why one of the conditions applies:
    i. The PR doesn’t match the scope of the issue being fixed.
    ii. The hotfix introduces regression in functionality or breaks backwards compatibility.
    iii. The scope of the hotfix is too large or nature of the fix can introduce significant risks to other components.
    iv. The hotfix introduces a new feature in PyTorch.
d. In the cases where Facebook needs more than 3 US business days to ensure that the PR does not meet one of the above conditions, Facebook will notify the PR submitter that an extension is needed, explaining which of the above four conditions may apply. Facebook may issue extensions for 5 business days at a time and work expeditiously to either invalidate or approve the PR. Participant may not introduce any functionality in PyTorch Enterprise distributions beyond Hotfixes.