ahana

OSS Presto Development & Release

Ahana DevOps & Performance Engineering

Background

1. Since project inception, the presto team at Facebook/Meta has been the only one responsible for building, testing, and publishing Presto
2. Facebook/Meta testing is a big factor in the reliability, scalability, and performance of presto releases, giving confidence to the community
3. That said, the project is tied very closely to the internal build infra which can cause friction
4. During testing if something breaks internally, there is little visibility into what broke
5. The goal of Meta donating the presto project was also for the Presto Foundation community to pick up building Presto

Proposed Model Moving Forward

1. Presto Foundation member Ahana to increase responsibility for build and release process as service for the community
2. Invite the community to contribute to a “query bank” and datasets
3. Initially community testing will be very limited in comparison to the large users like Meta
4. PF members will continue to do their own internal testing and report back
5. Over time, the community builds up shared query bank and workload to increase validation coverage

Scope of the Process

1. Product source code repository prestodb/presto

2. DevOps pipelines/tools/scripts prestodb/presto-release (?)

Two Release Trains

1. Scheduled monthly release train, targeting **Stable** releases

2. Scheduled Weekly release train, targeting **Edge** releases

Branches and Release Versions

1. Only a Stable release increments the build version of the master branch
2. Each release (Edge or Stable) creates a new branch off master branch
3. Stable
    a. version: 0.281.0
    b. branch: release/0.281
4. Edge
    a. version: 0.281-edge1, 0.281-edge2, …
    b. branch: edge/0.281-edge1, edge/0.281-edge2, …

Stable Release Schedule

Stable release is scheduled monthly on the first Thursday, standard Maven release process is performed, and

1. A new release branch, say release/0.281 (version=0.281), is created from maven release tag on the master branch
2. Commits into branch release/0.281 is limited to big fixes
3. Community announcement of the release branch

Stable Release Schedule (cont.)

In the next three (?) weeks

1. Key community members work on validation and fixes on the release branch (release/0.281)
2. Repeat till it is certified
3. Release (0.281.0) is announced with build artifacts and release notes

Edge Release Schedule

An Edge releases is performed on every Thursday after the corresponding Stable release, and before next Stable release.

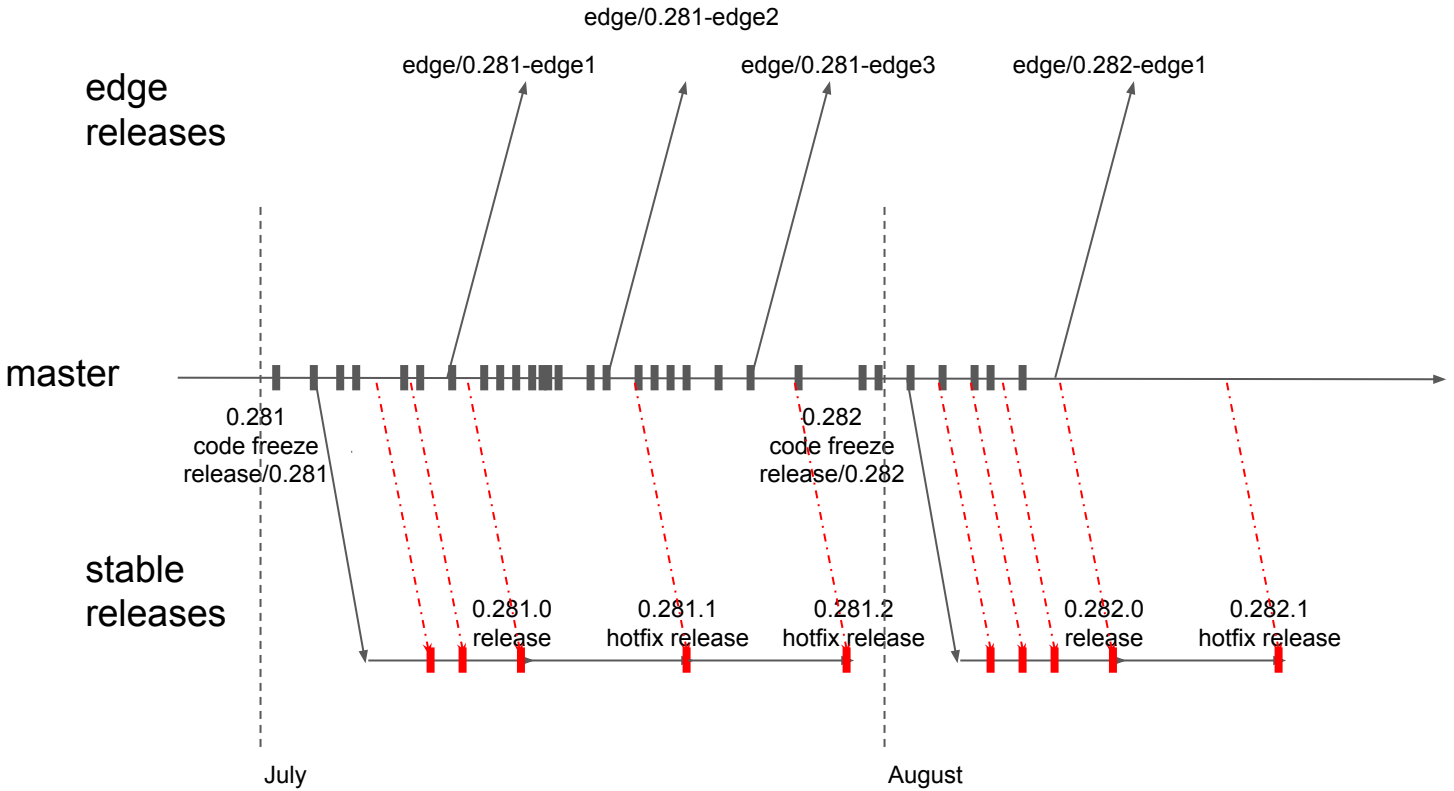
1. The Edge branch is cut at the head of master branch, say edge/0.281-edge1
2. The version is updated in this new branch accordingly, say 0.281-edge1
3. Community announcement of the Edge release branch and build artifacts
4. No release notes are needed (or just commit messages?)

Hotfixes (or Patches)

Hotfixes will not be supported after next new stable releases.

1. Community reports a bug,
2. Key community members decide that a hotfix is needed
3. Develop and validate the hotfix on the master branch
4. Backport the hotfix to corresponding Stable and Edge branches
5. Increment the version
    a. 0.281.1 for the first hotfix of the Stable release
    b. 0.281.1-edge1
6. Community announcement of a hotfix and build artifacts
7. Update release notes for the Stable release

Possible Release History



Feature Development / Bug Fix

1. Create a GitHub Issue, and provide related documentation
2. Create a new branch from master
    a. feature/{issue-id}-{description}
    b. bug/{issue-id}-{description}
    c. forked repository (?)
3. Development and test
4. Create a PR for review, target branch (master, or release/)
5. Receive approvals and merge the PR

Verification Pipelines

1. Build Verification Pipeline (bvp)

2. Deployment Verification Pipeline**s** (dvp)

3. System Verification Pipeline**s** (svp)

Build Verification Pipeline

1. Unit tests and integration tests in repository prestodb/presto

2. Run on every commit to every branch via GitHub Actions (?)
   a. may skip for non-code commits
   b. may skip for intermediate commits to a feature development branch or a PR

3. Required when merging a PR into the **master** branch

4. Required when backporting a fix into a **stable release** branch

Deployment Verification Pipelines

1. Jenkins pipelines defined in repository prestodb/presto-release

2. Build artifacts (jar/docker image) and deploy presto clusters, then run
   a. benchto
   b. verifier
   c. other queries on different workloads

3. Optional when merging a PR into the **master** branch

4. Required when releasing build artifacts to the community public

System Verification Pipelines

1. Jenkins pipelines defined in prestodb/presto-release

2. Deploy presto clusters using existing build artifacts, then run
   a. load and stability testing
   b. chaotic/recoverability testing
   c. scalability testing
   d. …

3. Some are required when releasing build artifacts to the community public

Naming Conventions

1. Branches
    a. feature/18078-presto-native-execution
    b. bug/18264-native-macos-failure
    c. release/0.281

2. Tags
    a. edge-0.281-edge1
    b. release-0.281.0

Release Infrastructure

1. GitHub actions are used for unit and integration tests hosted in prestodb/presto
2. Ahana-managed Jenkins pipelines are used for deployment verification, system verification and automated releases
3. https://ci.prestodb.io, read access to all and write access to some members
4. Community members to contribute more test cases and test data (TBD)