A New CI System for Apache MXNet (incubating)

Currently, MXNet uses Apache's Jenkins build service where Jenkins master runs in Apache Infrastructure and is connected to slaves running in AWS account donated and managed by Amazon and MXNet's contributors at Amazon.

In this document I have listed the basic and advanced features that would be useful in a new CI system for MXNet

Problems with the Current Setup

1. MXNet has a complex build pipeline which is only expected to get more involved with time.
2. Over the past few months, builds have not been stable due to multiple reasons including both Infrastructure issues as well as flaky/incorrect tests.
3. Apache Jenkins master is overloaded with over 200 projects.
4. There have been issues such as unable to trigger builds, unable to access builds.apache.org pages, connectivity issues from master to slaves.
5. These and other issues have led to too many requests to the Apache Infra Team -
    - Addition/deletion of new slaves: Due to scaling activity, recycling, troubleshooting or any actions leading to change of slave Machines.
    - Change in Plugins / other Jenkins Master configurations.
    - Experimentation on CI pipelines including changes to scripts requiring approvals.
6. Hard to debug and resolve issues - Since access to master and slave is not with the same community, it requires Infra and community to dive deep together on all action items.
7. Several Tests rely on external datasets which has led to issues

On the dev list we considered the following -

1. Set up a separate CI build system for Apache MXNet outside Apache Infra
2. Have a separate Jenkins Master in Apache Infra for MXNet
3. Review design of current setup, refine and fill the gaps.

As a unanimous choice it was decided to move forward with ***Option #1*** and in this document I outline some guidelines which would help us in moving forward with a new system.

Basic requirements of a CI for MXNet

1. **Multiplatform support** is the primary requirement of MXNet CI including but not limited to - Ubuntu, Windows, MacOS, Android, Raspberry Pi, Jetson TX2 etc
2. It should easily **integrate with Github** and be able able to trigger builds on PR updates, merges (even comments in the future!) etc.
3. The builds page and status should be available to the **community for viewing**.
4. The **MXNet committers** should be able to login (using Apache credentials) - there should be some mechanism to whitelist access. In addition MXNet community might decide to provide login access to MXNet contributors as administrators through lazy vote.
5. There must be a **structured pipeline** -
   a. The pipeline should consist of multiple ***stages*** for example 'builds', 'unit tests',

'Integration Tests' etc which run sequentially.
b. Different ***tasks*** in a stage should be scheduled across multiple agents and run in parallel (ideally using docker containers)
c. Results from one stage should be ***stashed and reused*** by the next stage (even across agents). There must be a mechanism to ensure this happens cleanly across multiple branches and PRs that may be building at the same time.
d. The current MXNet pipeline does not need to deploy anything.
e. Every task running on a slave must clean the workspace and any downloaded data from it.

6. Small build and unit test times (the community must decide on an appropriate time limit). The current unit tests need some refactoring into smaller tasks or migration to the nightly job.
7. There must be a separate pipeline/job for long running tests that run ***nightly***.
8. In the future, the plan is to ***release artifacts*** from this CI infrastructure including docker images and pip wheels. There must be a secure way of doing this.
9. Timeouts should be set for the time a build runs on an executor. A task should not timeout in the queue. This is causing many problems currently.
10. Datasets for tests should be available to nodes locally (For example using S3 buckets, shared cache)
11. Current Tests need to be revisited for correctness and validity.
12. Need to reconsider having a separate stage for Integration tests and deploy. These are very short stages that get stuck at the back of the build queue and considerably increase the overall build time. (We could also consider having a dedicated node for these short stages so they do not become the bottleneck)
13. More diverse tests need to be added for different devices - RPi, Mobile devices in future.

Additional Features

1. **High Availability:**
   a. The build agents/ slaves should be distributed across multiple regions
   b. Use ***EIPs*** even in a separate setup to make rebooting/replacing slaves easier
   c. In the future have a second identical setup which can act as a backup.
2. **Scalability:**
   a. Use ***autoscaling groups*** to increase or decrease the number of instances as needed.
   b. Have an additional fleet of slaves approved and ready to be deployed when needed.
3. **Automation:**
   a. Master/Slave management should be automated as much as possible.
4. The setup should make testing during ***Releases*** easier -
   a. There should be a quick way to copy all the jobs running on the master branch to run on the release branch.
   b. The ability to ***prioritize builds*** on this branch over other running builds (without having to manually kill running PR jobs as is currently done)
5. Run only one PR build at a time → ***Kill old Builds of the same PR***
6. Ability to reorder the build queue and prioritize in general (example releases)
7. Easier debugging: A good ***dashboard*** for build results, health metrics and code coverage in tests.
8. Tests/Jobs to identify ***Flaky tests***

9. Ability to replicate/duplicate the whole CI setup (for example for a backup or to the Berlin team.) using terraform templates for infrastructure, and a mixture of shell and python scripts.
10. The CI system should regularly go through extensive failure-testing of various scenarios.
11. Unit tests should be parallelized in a more structured manner, longer running tests should start first so that they do not become the stragglers in the end while all the short tests have completed.

Different CI Frameworks :

	PROS	CONS
JENKINS	1. Free 2. Should be quickest to set up - Jobs and scripts only need to be copied. 3. From experience it is known to handle MXNet's build pipeline complexity 4. No ramp-up period required 5. A separate Jenkins setup will be highly customizable due to lack of restrictions on usage of groovy scripts. 6. Since this is opensource, plugins can be modified/developed as and when necessary 7. Supports multiple platforms including edge devices and macOS. 8. There are many aws plugins for jenkins which can be explored to automate processes (s3, ec2 etc)	1. There are some plugins that are unstable because these are open source and untested. 2. As of now, there is no '***perfect***' plugin to modify the build queue to prioritize urgent builds. 3. Some investigation and additional test dumps needed to generate appropriate metrics dashboard.
AWS CodeBuild Plugin with Jenkins	1. Using the aws codebuild jenkins plugin, one can send jenkins build jobs to aws code build. 2. This eliminates the need for you to provision, configure, and manage Jenkins build nodes. 3. CodeBuild provides autoscaling of nodes	1. AWS CodeBuild currently only supports Ubuntu slaves. (But windows slaves can use EIPs)
TRAVIS	1. Other Apache projects are using it - so examples exist 2. Scripts used to configure builds so it is highly customizable.	1. Limited control since it is a hosted service 2. Lacks multiplatform support (does not support windows, CUDA/GPU specific testing.)
TEAMCITY	1. Stable and well maintained by Jetbrains 2. Good reports and metrics 3. Integrates very well with EC2 and	1. Proprietary (but open source projects can request free usage) 2. Limited examples for reference

	AMIs 4. You can reorder build queues with drag and drop 5. Concept of build chain similar to jenkins pipelines 6. It now supports scripted configs	in the apache/open source community (Apache Ant is one project that uses it) 3. This will require some ramp-up time, experimentation, change of build scripts etc
GITLAB CI	1. It looks like a complete CI framework providing all the features discussed	1. It does not support direct integration with github. Mirroring the Github repo to Gitlab is will be required.
BUILDBOT		

Conclusion

Based on the above discussion there are two different ways to go as follows:

a. Why Jenkins -

Most issues with the current setup are due to restricted access and not Jenkins itself.
The current MXNet CI is very unstable and the need for migration from Apache Jenkins is urgent.
Due to familiarity with Jenkins and already existing reusable pieces, it seems to be the logical and fastest choice.

b. A different framework? -

On the other hand a stable and well maintained framework like Teamcity could be a good solution too. It has some interesting features like good integration with EC2 instances and 'drag and drop' build queues.
However, due to lack of experience with these new frameworks, there are some unknowns - a) The initial Ramp-up and setup time; b) The necessary experimentation which might not be as successful as expected

So, to conclude,

In the short term, it might be a good idea to build a separate Jenkins based Infrastructure which is dedicated to MXNet.
As a long term project, it could be a good idea to experiment with Teamcity. If it is found to be a good fit, the CI may be gradually migrated to Teamcity.

Additonal Questions/comments

1. Do we want to test on MAC? Who pays for it?

2. What is the best way to connect the slaves to the master such that the replacement/rebooting process can be automated.
3. Be careful with passwords/keys when scripting the setup
4. Use EFS ?

References / more to read:

[1] https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software
[2] https://www.keycdn.com/blog/jenkins-vs-travis/
[3] https://lustforge.com/2014/08/21/jenkins-vs-teamcity-the-better-ci-tool/
[4] https://www.slant.co/versus/2477/2478/~jenkins_vs_teamcity
[5] https://blog.jetbrains.com/teamcity/2016/03/teamcity-take-on-build-pipelines/
[6]https://lists.apache.org/thread.html/13e5ba08acd2f28a49f791c23f4baf97ba9e5bcdabe43aec598d1e19@%3Cdev.mxnet.apache.org%3E
[7]https://lists.apache.org/thread.html/1038929de22ec9b8e0d1e60372913170a3330108457ab788b87eb066@%3Cdev.mxnet.apache.org%3E
[8]https://lists.apache.org/thread.html/a17974458be4130c2e27e2586a876ccb02eaa445e4811fa60e77a5c8@%3Cdev.mxnet.apache.org%3E
[9]https://lists.apache.org/thread.html/84a94be51e25ea780338e70feb9c1d4fb9084b2be6937f2b9f2ad8f7@%3Cdev.mxnet.apache.org%3E
[10]https://lists.apache.org/thread.html/4322a367c4891adadc26584338722b8e2d2e6dff37dd0fd1340a42ea@%3Cdev.mxnet.apache.org%3E
[11]https://lists.apache.org/thread.html/350228fae24f2c85a82e29dfb57587a27cf906b1ebe3586278f368fe@%3Cdev.mxnet.apache.org%3E
[12]https://d0.awsstatic.com/whitepapers/DevOps/Jenkins_on_AWS.pdf