Proposal: Create a Release Candidate for every release

Summary: Have every release go through a RC process for testing and evaluation of untestable requirements

Background

We have had several point releases have significant errors discovered after the release has been cut and announced. The most recent of these caused errors in being able to sync mainnet on a fresh box, which were not found through the normal testing and evaluation cycle.

Proposal

- Every release gets an RC which needs to be formally evaluated before being accepted as a release
- Only bugfixes to the RC should be accepted into the release branch for the RC
- Bugfixes are to be merged to the master branch and cherry-picked onto the release branch
- Any new commit other than the release commit on the release branch invalidates any testing done on the release already

Test Process

- To be done from the next release and following:
  - Upgrade a node already in-sync on each of the supported testnets and mainnet(s)
    - Ensure it stays in-sync
    - Ensure it maintains a full peer-count
  - Fast-sync the networks from 0
    - Ensure we get to HEAD and stay in sync
    - Ensure we are able to maintain our peers
    - Ensure we can do so within recommended system requirements (CPU and RAM)
- Engineering work required
  - Create an IBFT-based testnet
    - 1 long running
    - 1 automated setup for testing initialization
  - Create private testnet with known number of peers (cross-client and cross-version) < --max-peers
    - If we cannot maintain the known number of peers, fail
    - If we cannot sync to HEAD, fail
    - If we cannot maintain sync for ~24h, fail
  - Include testing of mining and stratum-integration
  - Ideally these "private" test networks will be managed by Terraform and given a test load by Caliper