Moving Ballista to a separate repo

GitHub Issue: https://github.com/apache/arrow-datafusion/issues/2502

Goal

The goal is to move the Ballista project to a new top-level **arrow-ballista** repository

Rationale

- Decouple the release process for DataFusion and Ballista
- Allow each project to have top-level documentation and user guides that are targeting the appropriate audience
- Reduce issue tracking and PR review burden for DataFusion maintainers who are not as interested in Ballista
- Help avoid accidental circular dependencies being introduced between the projects (such as datafusion-cli crate has circular dependency #2433)
- Helps formalize the public API for DataFusion that other query engines should be using

Challenges

Catching DataFusion changes that break compatibility with Ballista

One of the main reasons for keeping the two projects in the same repository is that it prevents DataFusion changes from being committed if they cause regressions in the Ballista test suite.

Common examples of this are:

- Changes to logical plans and expressions that prevent them from being serialized to protobuf
  - This is maybe less of an issue now that we have the datafusion-proto crate?
  - We should review the test coverage here and maybe add additional tests for Ballista use cases
- Changes to ExecutionPlan trait (especially async changes)
- Scheduler changes

Proposal for addressing this

arrow-ballista repo

- arrow-ballista dependencies on arrow-datafusion change over time
  - Start out pointing at 8.0.0 crates.io release
  - Switch to git dependency with sha when we need to pick up a new feature from arrow-datafusion
  - Keep updating sha to pick up new features as needed
  - When there is a new arrow-datafusion release, point to the official release again and decide whether to release a new arrow-ballista version at that time
  - Repeat

arrow-datafusion repo

- Add CI check that pulls arrow-ballista master branch locally and builds it using path dependencies on arrow-datafusion so that we know if changes in the current PR would break compatibility with arrow-ballista
  - We can build arrow-ballista with a feature that overrides the dependencies on arrow-datafusion to be based on paths instead of git dependencies
  - This should be an optional check so that we can choose to merge if needed (with co-ordination with Ballista devs)

Defining a public API

For projects like Ballista that are building on DataFusion, it would be good to have a known public API that can be considered stable and unsupported.

Perhaps the best way to do this is to make more of an effort to keep unsupported internal functions and data structures private so that the rustdoc is the public API.

Open Questions

- Tpc-h benchmarks can run against both DataFusion and Ballista so do we just duplicate this in both repos?

    - Duplicate is easiest and each project can evolve the code as needed
- How will Ballista reference the datafusion.proto file?
    - Use the new binary serde API to delegate to DataFusion for serde?

Steps to implement the plan

- Pre-cutover
    - Do a dry run on a branch or PR
    - Agree on CI approach
    - Hold a formal vote
- Cutover
    - Create repo via ASF tools
    - Push the 8.0.0 arrow-datafusion history to the new repo
    - Change the ballista Cargo.toml files to depend on DataFusion 8.0.0
    - Delete datafusion crates & confirm tests pass
    - Delete ballista crates from DataFusion repo & confirm tests pass
- Post-cutover
    - Move Ballista issues to new repo if possible
    - Update DataFusion README and user guide as needed
    - Create new Ballista README page

Earlier discussion on options for addressing the issue of breaking changes

Option 1: Ballista builds against specific DataFusion git sha

We change the dependencies to git dependencies, pointing at a specific sha. When we first create the repo, this could be the `8.0.0` tag but we then point to later code as needed.

We could choose to automate creating PRs to update the sha and this would allow us to catch breaking changes early (as long as someone is paying attention to these upgrade PRs in Ballista).

When we are ready to release a new version of Ballista, we would change the dependencies to regular crate dependencies.

Option 2: Ballista builds against DataFusion master branch

This would catch breaking changes much earlier but could be disruptive to Ballista developers.

Option 3: Run latest ballista tests in DataFusion CI

This is roughly the approach taken by arrow-rs to ensure that the integration tests with other arrow implementations do not break. The idea is that each DataFusion PR also checks out and runs the ballista CI tests.

This would ensure that datafusion doesn't break ballista, but it would require careful coordination with ballista is/when we did any breaking API changes. It works well for arrow because the format is quite stable and we have minimal breaking changes