Solution proposal for CI/CD + coordinator repo scenario

I would use GH Actions because of native permissions management, stability and easy integration for this solution.

Coordinator repo setup

- Repository `main` branch is protected and only mergeable via the Github Actions CI pipeline operated by GH App.
- Directory `changesets` with manifests holding compatible branches from each repository to be merged together.
- Directory `scripts` holding the `run_integration_tests.sh` script.

Coordinator mechanism

1. A user creates a PR with a manifest file containing **one or more changesets** from the example file where each contains **one or more repos with branch refs**.

2. The PR triggers an asynchronous pipeline run described in section **Integration Pipeline**.

3. The PR has a check for completion of **Integration Pipeline** run and PR is merged with changes for the status of respective changesets to either 'failed' or 'success'.

4. Changesets that got tested successfully are merged to `main` branch in respective repos.

Integration Pipeline

Integration pipelines are in a single concurrency group to run only one at a time.

1. Run a matrix-job for each of the changeset in uploaded manifest (parallel to make it faster):

   1. Checkout all repositories by looping through `repositories` key in the changeset spec.
   2. Run `run_integrations_tests.sh` with all downloaded branches.
   3. Wait and check the result of tests:
      - **(FAILED result)** logs from failed tests are outputted in the PR as comment, team is notified and this particular changeset result is saved as 'failed'.
      - **(SUCCESS result)** all branches defined in the changeset are merged to `main` branch in respective repos and this particular changeset result is saved as 'success'.
   4. Commit for the manifest is created in the PR with status update of respective changeset to either 'failed' or 'success'.

2. After all jobs are completed, PR is **merged**.

Permissions and security

- Protected `main` branches on all repos that take part in the system (GH setup managed from Terraform and team do not have Admin privileges to change it) - this creates immutable infrastructure.
- Set only GH App `ci-bot` can merge branches through a pipeline.
- `ci-bot` credentials are securely stored and cannot be retrieved by non-administrator user.

Trade-offs and assumptions

- I assume we can run multiple E2E test in ephemeral environment (e.g. using Docker or K8s) to speed up the process.
- For faster total-time-to-merge we can run **Integration Pipeline** in parallel, this assumes all changesets in a single batch are non-interfering.
- We could make **manual approval step** between E2E tests succeded and merging repos to main branches to manually review the environment before merging, but it would slow down the process. The end goal should be to aim for so comprehensive E2E tests that changeset should be tested and merges should go automatically after successfull result.

Additional enhancements

- We have implemented batching by running all changesets together in the same pipeline run.
- I cannot find a reason for implementing dynamic queue management with knowledge about the problem that I have and with my assumptions above. HOWEVER, if we have **limited runners** or we expect to have **a lot of changesets** to be run at the same time or need **a lot of flexibility**, we could improve the solution and introduce new `priority` key to the changeset spec to define the order of execution.
- But going further at **this level of complexity**, for the user of this system, in my opinion it would be easier to have a **simple Web APP with DB backend to create and manage changesets visually**, ordering in the queue and dynamic reorder, pausing, resuming or skipping. This would be **better solution for both the user of this system and its maintainers**.

Final Thoughts

For basic use case, I would go the **coordinator repo route**, but if we expect to have a lot of changesets to be run at the same time or need a lot of flexibility, I would go 100% for the **Web APP route** as I've already seen similar solutions in the past in my career for this kind of problems.