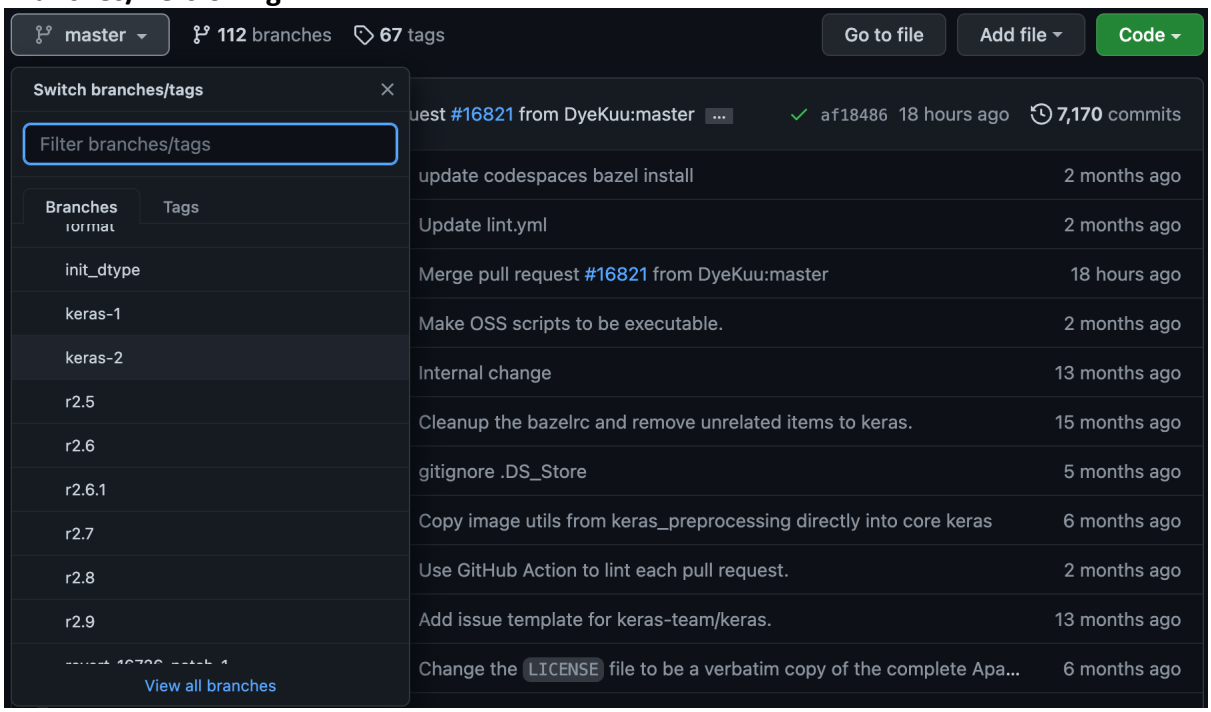
PROBLEM 4

You are part of an e-commerce product team, planning a black Friday event as part of that, checking the beta version of the same by deploying the last patch to it. Noticed after applying the patch, major features get broken in the beta version, which affects the release majorly. The sales team wants to revert the last patch fix applied to the beta environment.
The last patch includes:
Front-end MS code change
Back-end MS code change
MongoDB schema changes
k8s cron (scheduler) changes
Environment: k8s (MS, dbs)
We need to roll back to the old state by roll backing all the above items in the patch
Bonus:
If the team given the solution for the above without any data loss, by retaining the new data ingested
after the patch updates

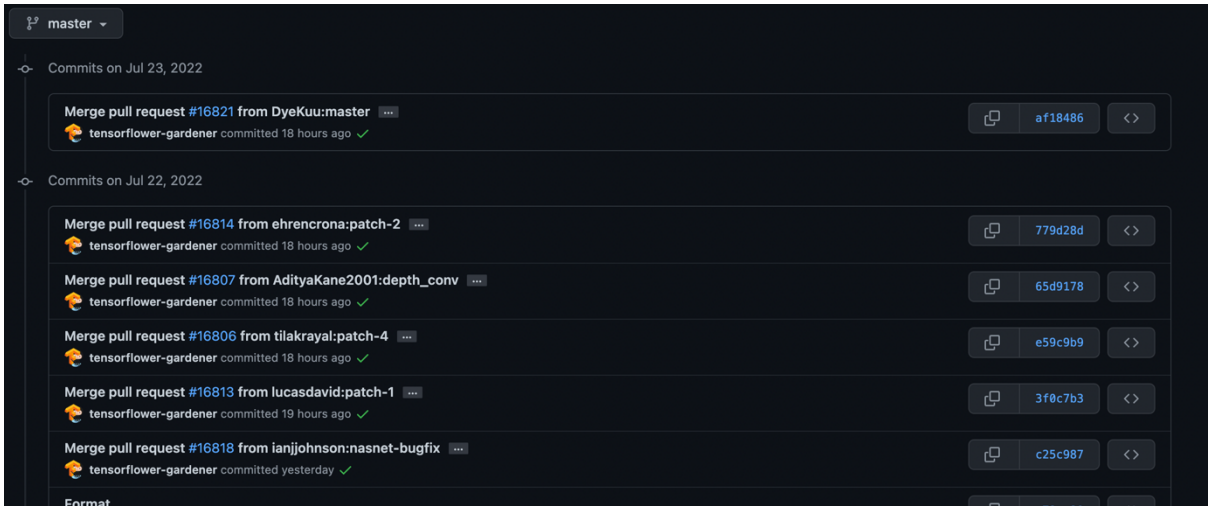
SOLUTION:

GitHub is a code hosting platform for version control and collaboration.

Branches/Versioning:

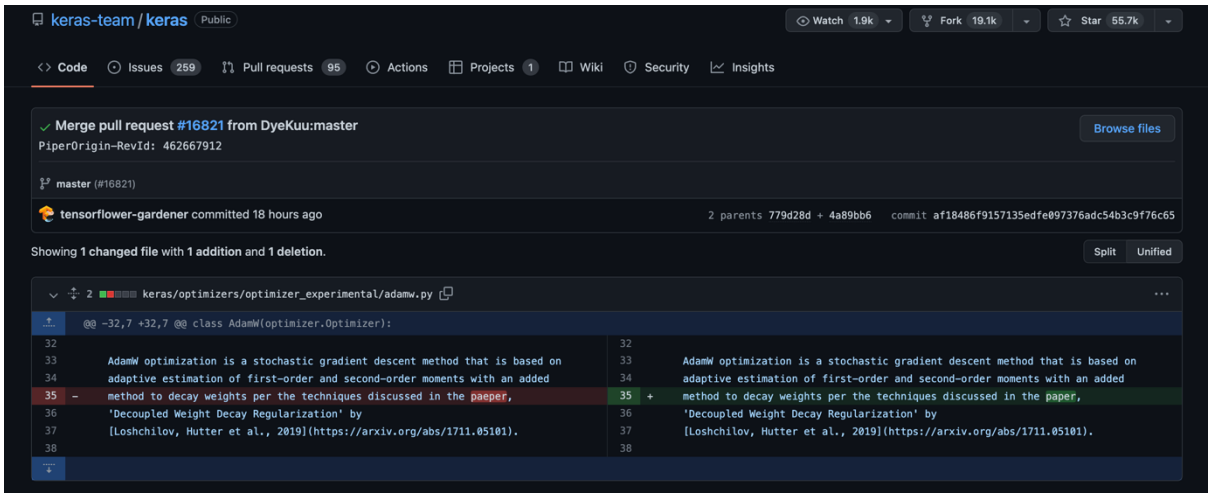


The above screenshot is a public repository of tensorflow. Github Branches allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository.



The above image is called commits. When you make a commit to save your work, Git creates a unique ID (ie. the "SHA" or "hash") that allows you to keep record of the specific changes committed along with who made them and when.

1. FrontEnd or BackEnd Version Rollback



   - You can roll back the changes from the GitHub Website by *pushing the older version branch into the master/main branch.*
   - If you use CI/CD Pipelines, then downgrading the version of that product or a service is very easy. You will just have to follow the above step and roll back the updated version to production.
2. MongoDB schema changes
   - **Write an upgrade script.** The easiest strategy is to write an upgrade script. There is effectively no difference to this method between a relational database (SQL Server, Oracle) and MongoDB. Identify the documents that need to be changed and update them.
   - **Incrementally update your documents** as they are used.
     - First, queries against a schema where half the documents are version 1 and half the documents are version 2 could go awry. For instance, if you rename an element, then your query will need to test both the old element name and the new element name to get all the results.
     - Second, any incremental upgrade code must stay in the code-base until all the documents have been upgraded. For instance, if there have been 3

versions of a document, [1, 2, and 3] and we remove the upgrade code from version 1 to version 2, any documents that still exist as version 1 are un-upgradeable.

3. k8s cron (scheduler) changes

```
$ kubectl create job my-one-time-job --from=cronjobs/my-cronjob
```

This is how a k8s cron job is run. We can reupdate the code for each cronjob and run the code again.