GIT Internals :-

-> objects :- Git stores data as objects.
Identified by SHA-1 hash.

blob -> stores raw content of the file,

Tree -> represents the directory, it only maps the filename to hash of blob
creates the snapshot of directory.

Commit -> points to tree objects, contains meta data (author name, data, message etc).

-> staging area (Index) :- add command moves -> WA to staging area.

-> References :- pointers to specific commit hash

Branch -> always points latest commit
changes mode moves to updated.

HEAD -> indicates current branch working on

Tag -> immutable pointer to specific commit,
used for marking versions.

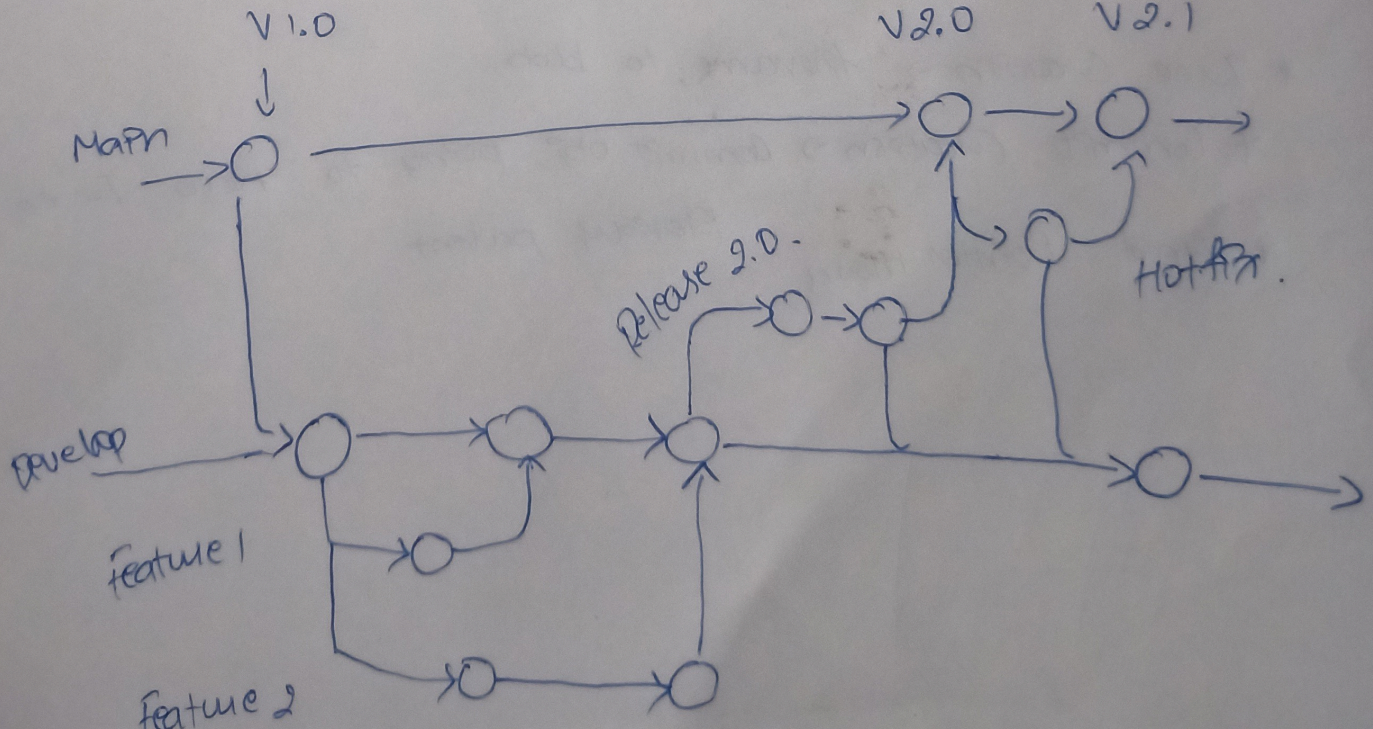
Nuances in behavior and operation:-

Snapshot based storage ->
-> makes snapshot of entire project in each commit.
-> if files haven't changed, links to previous, no copying.

Directed Acyclic graph:- The history of commit pointing to parent.

Fast forward merge -> if no commit on a branch, it moves branch forward.

Git workflow for Agile Development teams :-



operational mechanisms:-

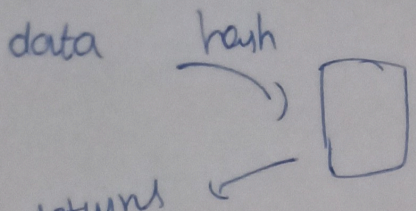
when we add file to staging process.

1. * Hashing of the file content using SHA-1
   * Blob Creation -> Compress content and write in object db.
   * Index updation -> update staging area index.

when file Commit:-

* Tree Creation -> filename to blob.
* Commit Creation -> Commit obj points to new Tree, Previous parent
* Head Pointer moves.

CAS → Content Addressable Storage:-



deduplication → two identical / same file have same hash, so stored only once in db.

Immutability → once object created, cannot be changed, editing content creates new hash, new blob

DAG graphs:-

always single direction.

Merging capabilities → merging sees the lowest common ancestor and merge them.

Cryptographic chain → Commit object → hash of parent → DAG forms Merkle tree.