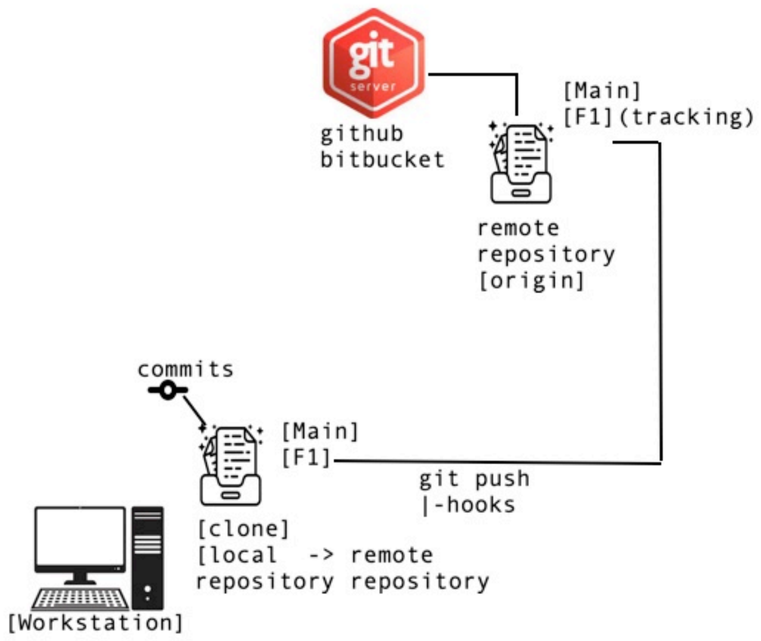

Branching Strategies
#1. Feature-based Release
1.1 For each user story the developer has to work on, he has to create an Feature branch corresponding to that user story.

The developer works on the feature on within the local repo of the workstation and commits all the code changes within the feature branch.
upon completing the code changes the developer has to push the changes in the local repo of the f1 branch into remote repo of the git server

Here to push the code into the remote repo tracking branch and to enforce quality gating we use #2 mechanims

1.1.1 projects that are build from scratch based on Agile Methodology
when the developer does an git push:
1. The code must and should pass through the quality gating process upon certifying then only the code will be pushed on to the F1 (Tracking branch) of remote repo server
Here comes the Git Hooks, through which we run the
 1.1 build on the local repo code
 1.2 run the unittest cases
 1.3 capture testcoverage using jacoco
 1.4 do code review using sonarqube
if the code meets the quality criteria then the code will be pushed onto the Tracking Branch f1 of the remote repo.
Otherwise git push fails in pushing the local code on to remote repo branch

since most of the code has been tested through unit test automation, we may have less or 0% dependency on qa engineers to test and certify the code.

1.1.2 projects migrating to Agile methodology
There are projects that are not being developed on other methodologies and are being migrated to Agile Methodology. In this case they may not be unit tests in place and may not enforce any quality gating tools like sonarqube or jacaco etc. On such projects if we enforce quality gating checks in-first place when we do git push, then the release will be blocked, since it takes huge time/efforts in building the unit tests, test coverages and code reivews ontop of the code. So to help the team in building the quality code over the time we implement a different gating process here.

1.1.2.1 when developer does an git push to the remote repo tracking branch:
Here on the local workstation of the developer, we only build the code, execute the unit testcases and verify all the tests are passed or not. If the tests fails we abort the push otherwise the code will be pushed without verifying quality checks.

The devops team has to setup an automated job (could be a pipeline) that executes based on scheduled intervals or when a push happens onto the remote repo branch, that runs through the testcases, testcoverage and codecoverage and published the coverage report to the development team asking them to fix the issues reported.

In this case the team has to greatly rely on manually testing carried by the qa engineers in releasing the code to the production.

#2. upon completing the feature developement, the code should be released to the qa
merge the feature code with the other features that are worked upon by the developers in current sprint by merging into develop branch. Which means each feature branch upon completing the Feature should be merged into develop branch and deliver it for qa

F1 (completed) -> merge -> develop
F2 (completed) -> merge -> develop (+F1) (continous integration)
F3 (completed) -> merge -> develop (+F1)(+F2) —> deliver to qa for testing

Now the developer has to create an Pull Request (PR) to merge the code from Feature branch to Develop and would be send for code review to the whole team. At this stage #2 things takes place
1. a code pipeline will be triggered (before merge after creating the Pull Request) that ensures the unit tests on the integrated or consolidated code is passing through and the quality gating criteria is met)
2. the PR needs to be approved by the members of the team.
Then only the PR will be allowed to merge onto the develop

#3. release the integrated code to qa
upon merging the code onto the develop branch. On the develop branch code an continous deployment pipeline will be triggered.
1. builds the code
2. provisions the qa infra through iac tools
3. install softwares and configures them using software configuration management tools
4. deploys the application on to the infra
5. runs the integration-tests on that code

if the integration-tests or e2e tests passed then the code will be released to qa for further testing otherwise the qa env will be rolled back with previous good state of the code.

#4. upon the qa verifies and certifies all the user stories of the sprint in develop, then these features will be demoed to the product owner based on the acceptancy the code will be merged onto the Main and released to the production.

Here all the features are socked in lower environments like qa, uat, performance environment and at later point of time the code will be delivered to the production which is called “Feature-based” release.
So that the customers aspart of that release sees quite a number of new features in their system.