# Development Workflow using Git

Case Study

# Contents

## ENVIRONMENTS

In managing the projects we build and maintain, we will have the following set of environment or server tiers, in order:

1. **Development (DEV)**: where all the development code is committed and kept internally, and it is the first environment after the local machines
2. **Integration (INT)**: to make sure that environmental concerns are fully considered in the application
3. **Staging & Trial:** the pre-production or production-simulation environments, and the last before going on production
4. **Production/Release**: the final version of source code, files, environment variables, and similar stuff

The source code will be merged into the mentioned environments in the order they are listed.

# ENVIRONMENT BRANCHING

In our project environments, the methodology we will follow to separate environments from each other per each project is setting a multi-branches environment (repository). We are gonna have one branch for each environment and one more branch as a temporary branch that will be created and removed for a special purpose. These special purposes will be when for example merging a heavy change into which it is a matter of “make it or break it”. In this situation, many tests and maybe environmental conditions are set to this temp branch which will be an extra step of making sure everything is working as wanted. Note that in most cases this branch will not be needed or just be created as a buffer between the two environments and the merge from it into its master branch (env branch) will be immediate.

The following flow diagram shows our method of making a temp branch for each environment before merging into it. Note that the next sections will explain the flow of the process depicted in the diagram.

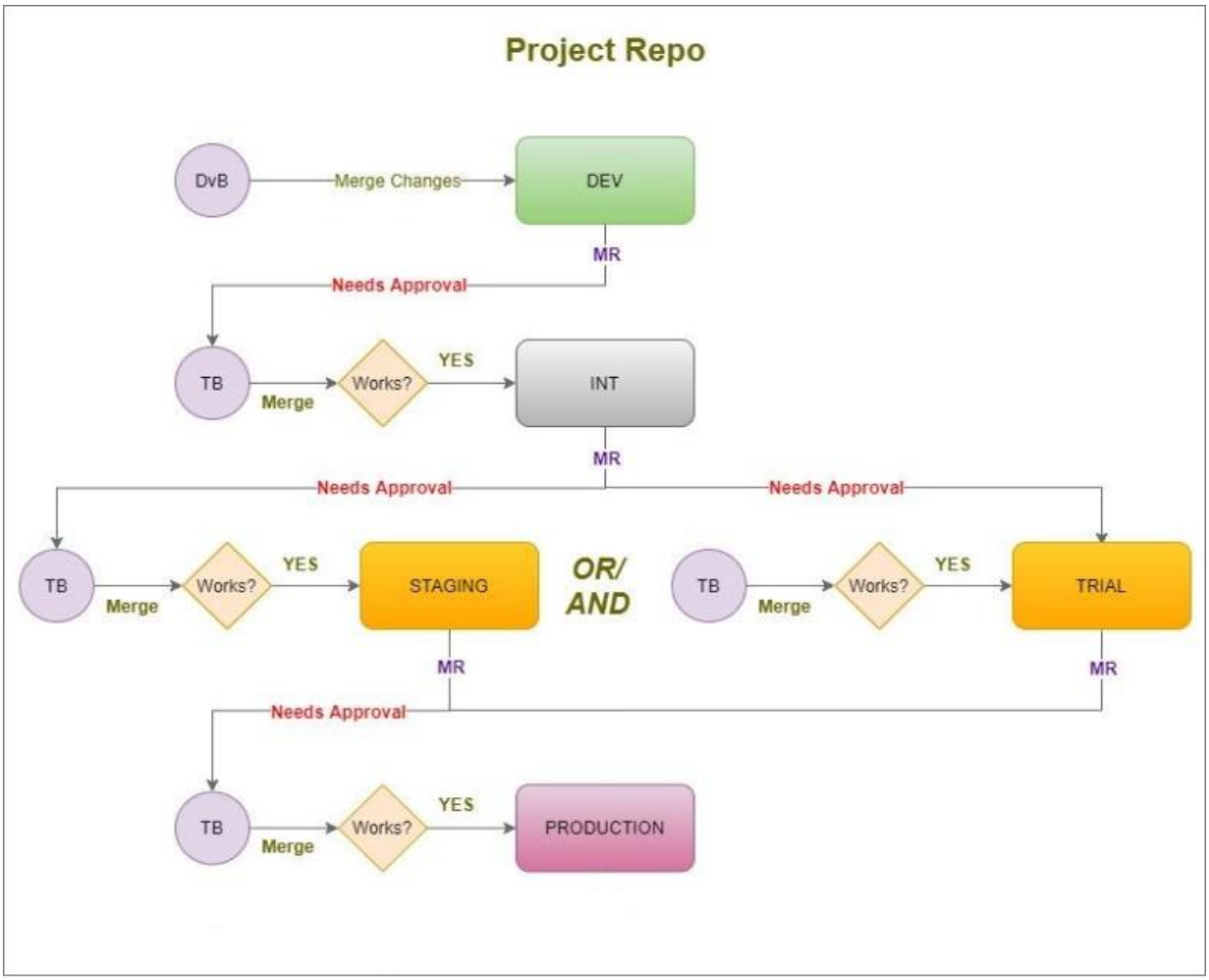


*Figure 1: Merging changes through different environments.*

## TEAM AND PERMISSIONS

Not all team members have the same permissions to the Git resources of the project. In the following table, permissions are shown per each team resource level, in the most general or normal way.

| Level | Permitted Actions | Needs approval/review? |
|---|---|---|
| **Developer** | • Create branches from DEV master B<br>• Commit Changes to DEV master B | • -<br>• Sr. Dev / Lead Eng. |
| **Sr. Developer** | • Create branches from DEV master B<br>• Commit Changes to DEV master B<br>• Create / Merge into INT/Staging/Trial TB<br>• Merge TB into INT/Staging/Trial | • -<br>• -<br>• Lead Dev.<br>• If ok? -<br>not ok? Lead Dev. |
| **Ops / DevOps** | • Create branches from DEV master B<br>• Commit Changes to DEV master B<br>• Create / Merge into INT/Staging/Trial TB<br>• Merge TB into INT/Staging/Trial<br>• Create / Merge into Prod/Release TB<br>• Merge TB into Prod/Release | • -<br>• Sr. Dev / Lead Eng.<br>• Tech. Mngr<br>• If ok? -<br>not ok? Tech. Mngr<br>• Tech. Mngr + Client<br>• If ok? -<br>not ok? Tech. Mngr |
| **Lead Dev /** | • Create branches from DEV master B<br>• Commit Changes to DEV master B<br>• Create / Merge to INT/Staging/Trial TB<br>• Merge TB into INT/Staging/Trial | • -<br>• -<br>• Tech. Mngr<br>• If ok? -<br>not ok? Tech. Mngr |
| **Lead Ops / Lead DevOps** | • Create branches from DEV master B<br>• Commit Changes to DEV master B<br>• Create / Merge into INT/Staging/Trial TB<br>• Merge TB into INT/Staging/Trial<br>• Create / Merge into Prod/Release TB<br>• Merge TB into Prod/Release | • -<br>• -<br>• Tech. Mngr<br>• If ok? -<br>not ok? Tech. Mngr<br>• Tech. Mngr + Client<br>• If ok? -<br>not ok? Tech. Mngr |
| **Tech. Manager** | • Create branches from DEV master B<br>• Commit Changes to DEV master B<br>• Create / Merge into INT/Staging/Trial TB<br>• Merge TB into INT/Staging/Trial<br>• Create / Merge into Prod/Release TB<br>• Merge TB into Prod/Release | • -<br>• -<br>• -<br>• -<br>• Client<br>• - |

# WORKFLOW

## Making code changes

The first step of the project delivery workflow is a developer, ops engineer, or a DevOps engineer assigned to a ticket, and then creating a branch from the DEV master branch to work on the ticket, or, as default, each team member will have his/her branch. All environments here are assumed to be ready as repository branches per each project. After that, a senior engineer (dev, ops, DevOps) or the team leader reviews the changes the engineer made and approves him/her to merge them into the DEV master branch.

## Deployment on Integration, and start testing

The next step is that the team lead moving from DEV to INT by first creating a temp branch from the INT environment, merge changes into the temp branch, merge the temp branch into the INT master branch, and deploy them. This is done if the change is not serious or there is a chance it will affect the INT environment. What to do next is to start applying different QA and automated tests on the INT environment to make sure the new version of code is comfortable with the environmental conditions and works well.

## Moving to Staging or/and Trial

After the testing on INT resulted in success, now it is time to move to staging or/and trial. Here the team lead returns to the technical manager asking for approval to start merging into and deploying on the staging or/and trail environments. When the manager confirms it is time to do so and approves that, the team lead does the same that was done to INT starting with creating a temp branch and ending with testing the deployed changes by the QA team.

## Production

When everything works fine so far, the technical manager(s) makes a demo of the new version of the application to the client on staging or trial. If the client accepts the application behavior, they ask about moving it to production. After approval, the operations / DevOps team starts the same cycle of creating a temp branch of the environment, which is the production here, and if there are any needed tests, they will be applied to that branch, if not, the branch gets merged into and deployed on production.

## The official release

The release is basically the same as production, with the same steps done in production, including the client, the technical managers, and the ops / DevOps team. What is different in release than production is that it is the business state of revealing a new feature(s) or/and resolving an old issue(s), which is done by the vendor and client agreement. The release involves some other main aspects like the products or service’s version, release notes, and so on.

# SPECIAL CASES

## Big Changes

When making significant changes that may affect the environment to be deployed on, it is better to create a temporary branch from that environment and then merge those changes and deploy them to that branch. After that, testing those changes on that branch to make sure the changes are compatible with the new environment.

## Temp Branch test failed

When the test on the temp branch of an environment to be deployed on results in an issue or system-break, the team member that was responsible for merging to the temp branch will refer to the technical manager to decide between reverting changes or resolving the issue. After the manager makes the decision, the workflow of either reverting changes or moving on by resolving the issue will be normal.

Reverting changes will be done by the team member responsible for merging to the temp branch, whereas resolving the issue will be dependent on which team is responsible for it.

## Hotfixes

Hotfixes are issues that happen to the production or release environment. In this case, a new branch called “Hotfix” is created. The reason for creating the hotfix branch is to take the last version of the code of that environment and then work on it to resolve the issue. When the issue comes as an urgent requirement, if the operations of DevOps team can resolve it, they will do that and immediately merge and deploy the hotfix branch with the committed solution to the issued environment, and the technical manager will be kept posted with that procedure. If the issue is to be resolved by one of the development teams, the technical manager will be notified to decide the procedure or the responsible team to resolve it, merge the resolved code, and deploy it.