# Environment strategy

# 

|  |  |
| --- | --- |
| **Environment** | **Description** |
| **Feature Development Environments** | Individual Dev sandboxes or scratch orgs. If multiple developers work in the same environment, they should align between themselves to avoid conflicts and overwritten code. |
| **DevInt** | Dev sandbox used for validations and checking that all features being implemented do not have conflict among themselves. |
| **Testing** | Partial sandbox to retain data even after being refreshed. User for QAs and testers to validate if features are working properly. |
| **Staging** | Full sandbox where features approved by QAs and testers are deployed. This environment will contain at all times features that are ready to go to Production. It will also be used for training and UAT. |
| **Production** | This is the live environment. |
| **Hotfix** | Dev Pro sandbox. Will always be a copy of Production and will be used for fixing and testing of critical bugs n Production. |

**Scratch Orgs/Individual Sandboxes**

There is one scratch org/sandbox assigned to every developer. If multiple developers work in the same environment, they should align between themselves to avoid conflicts and overwritten code.

It is strongly recommended to keep a local copy of the ongoing development on VSCode as well as the Feature branch associated update in a regular basic (ideally several times a day) to avoid losing any ongoing work.

Every time there is a successful merge request on the “main” branch from any individual sandbox, the latest code is automatically deployed into all individual sandboxes.

Squad Tester supports Squad Developers in this environment to perform feature testing before merge.

**DevInt (Development/Deployment Checks)**

Dev sandbox used for developments integrations to perform validations and checking that all features being implemented do not have conflict among themselves.

Code gets deploy into this environment automatically every time a merge request has been approved on “main” branch.

This environment is not mean to be used by Developers or QA.

**Testing (SIT and regression testing)**

Partial sandbox to retain data even after being refreshed. User for QAs and testers to validate if features are working properly.

Code is deployed after every successful deployment in DevInt to allow testers to work on it as soon as it is ready.

Full run of local test at 00:00 CET.

*\*\*There is a possibility to just once a day so QA may not be confused with the features available on the environment and its instability.*

**Staging**

Full sandbox where features approved by QAs and testers are deployed. This environment will contain at all times features that are ready to go to Production. It will also be used for training and UAT.

Deployments into this environment happen when there is a new Release candidate (rc) branch, or new changes are added to the last one.

**Production**

Deployment on demand when release has been approved to Go Live.

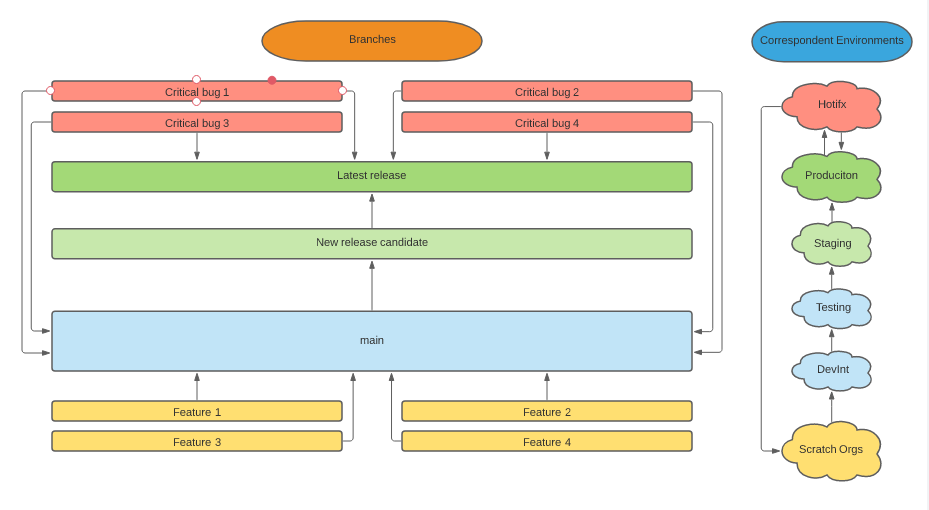
**Hotfix**

Copy of Production and will be used for fixing and testing of critical bugs in Production.

Fixes on this environment will be back promote into the “main” branch to ensure consistency of the code.

|  |  |
| --- | --- |
| **Branch** | **Description** |
| **main** | Default branch. Protected. Every developer should always checkout from the most updated version of this branch in order to create their feature branches before starting a new development. When a development is finished and have passed proper quality gates, it will be merged into this branch and automatically deployed to DevInt. Content of this branch will also be deployed automatically to Testing after successful deployment to DevInt. |
| **feature/** | Branches created by developers out of main for developments. Individual sandboxes or scratch orgs are used to work on these branches. |
| **bugfix/** | Branches created by developers out of main for bug fixes. Individual sandboxes or scratch orgs are used to work on these branches. |
| **patch/** | Protected. Branches created out of the latest tag when a hotfix is required. Once hotfixes are received here, the branch is deployed to Production, rebased into main, and a new tag is created. |
| **hotfix/** | Branches created by developers out of latest patch branch for critical bugs in production. Hotfix environment is used to work on these branches. They are validated and merged into the latest patch branch. |
| **rc/ (release candidate)** | Protected. Branches created out of main, at a particular point in time, where all features until then have been properly tested, and approved. It is deployed to Staging for UAT and performance testing. Any defects found during that phase should be fixed with a bugfix branch to be validated and merged into the latest rc branch and directly deployed to the Staging environment. Once UAT and performance testing is approved, the branch is deployed to Production, rebased into main, and a new tag is created.  If a feature already merged in main is not tested and approved, features added later, even if tested and approved, will not be deployed to next environments until all previous features are tested and approved as well. This restriction is to avoid dependencies on untested features and maintain robustness, however it is subjected to exceptions. |

# Branching strategy



*How should developers use branches and source control?*

*Feature development:*

1. Every time a new feature development is going to start, developers should create a **feature branch from main.**
   1. Make sure you create your feature branch named after your US out of latest version of main.
   2. Make sure you push your development into the feature branch frequently (ideally few times per day).
   3. Make sure you keep a local copy (VSCode) and the feature branch update to avoid losing work when automatic back promotion happens on individual sandboxes.
2. Feature branches must have a sort life time (no more than few days) to avoid conflicts when merging the code into main branch.
   1. If this is not possible you should at least rebase your feature branch every few days to avoid having conflicts with other developments already merged into main.
3. When your feature (US) is ready and tested by the Squad Tester in your individual sandbox or scratch org, it is time to merge it to “main” branch.
   1. Please review the developing [guides](https://github.com/Nakama-Partnering-Services/guides) to ensure that the right formatting, code coverage, etc., has been taken into account to ensure your merge request is compliance and can be merge into main.
   2. Make sure you include all the relevant manual steps associated to your US on the deployment runbook.
   3. Remember, we enforce rebase in this project, you must rebase your local repository before your final commit and merge request creation.

*Bugfix development:*

1. Bugfix branches should be created for US of bugfix type. These US are common few days before a release to production, when UAT and final E2E testing is happening in Staging. After feature freeze, a new release candidate (rc) branch is created and only bug fixes are allowed. These ones should be created from and merged to this new release candidate branch.
2. Bugfix get promoted across environments in the same way that Features when they are part of the minor release and are validated and merged into main and deployed to Testing and lower environments.
3. If bugfixes are developed as part of the UAT testing after code freeze, they are validated and merged against latest release candidate branch and directly deployed into Staging. Later, they are rebased into main branch and deployed into Testing and lower environments.

*Hotfix development:*

1. If new release candidate is already into Staging, a new patch/ branch should be created out of the latest tag. Else, a new rc branch should be created out of the latest tag instead. Hotfix branches should be created out of the resulting branch, which contains production code.
2. Hotfix will only be used when critical/urgent bugs have been identified in production and require an immediate fix.
3. Fixes are implemented in Hotfix. If using a patch branch, they will not go to Staging first. They will be directly validated against Production and then, manually deployed there.
4. Hotfix branch is rebased back into main to ensure code consistency on the next releases.