LU11a - Environments & Branches

Umgebungen

Bislang war es möglich nur auf dem Master/Main-Branch zu arbeiten und diesen Stand auch für das Deployment zu nutzen. Bei grösseren Projekten nutzt man in der Regel verschiede Umgebungen.

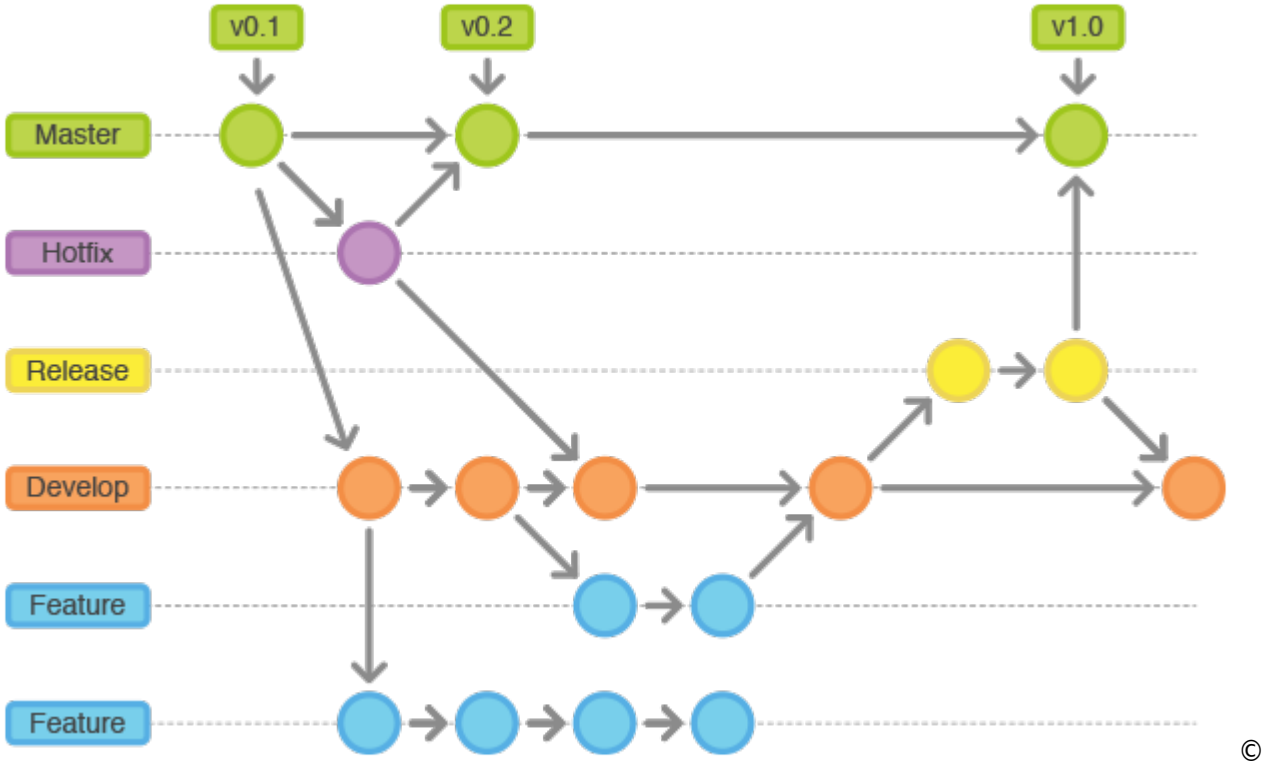
Die genaue Benennung und Aufstellung variiert natürlich je nach Vorgehen, Team und Projekt. Die grundlegendsten Umgebungen sind jedoch Dev/Develop, Test/UAT/QA und Prod.

© Miami University - Dev, Test, Prod: Oh, My!

Diese Umgebungen haben eigene Git-Branches und separate Host-Umgebungen. So ist sichergestellt, dass Entwicklung und das Testing die bereits veröffentlichte Applikation nicht beeinträchtigt. Dabei durchläuft neu entwickelter Code die Umgebungen in der Reihenfolge DEV → TEST → PROD. Bei Daten ist es durchaus üblich, dass man (gewisse) produktive Daten zu Test oder Debug-Zwecken auf Test- oder Entwicklungsumgebungen kopiert. Die Daten „fliessen“ in dem Fall von PROD → TEST → DEV.

Branches

Für die produktive Umgebung wird oft der Master/Main-Branch verwendet. Die Test-Umgebung kann auf dem neusten Release-Branch basieren oder auf einem kontinuierlichen Test-Branch. Die Development-Umgebung ist wiederum mit dem Dev-Branch verknüpft.



©

Seibert - Git-Workflows: Der Gitflow-Workflow

Darüber hinaus wird jedoch zusätzlich oft auch noch mit sogenannten Feature-Branches gearbeitet. Beim Beginn einer Story wird ein neuer Git-Branch abgezweigt und nach lokaler Implementation wieder in den Develop-Branch „gemerged“ (beziehungsweise „gerebased“). Dadurch kann zum Beispiel sichergestellt werden, dass sich die einzelnen Entwickler sich nicht gegenseitig stören und dass nur syntaktisch korrekter und somit lauffähiger Code auf der Development-Umgebung landet.

Hotfix-Branches werden jeweils bei einem Ausliefern eines Releases erstellt und dienen dazu, kleine Änderungen mit dem produktiven Codestand zu testen, ohne von den den neu entwickelten Features beeinflusst zu werden.

An dieser Stelle möchte ich nochmals das Semantic Versioning in Erinnerung rufen. Hotfixes erhöhen wie Bugfixes die Patch-Version und durchlaufen in der Regel nie die Development-Umgebung. Reguläre Änderungen wie neue Features durchlaufen diese Umgebungen und sorgen für eine Anpassungen der Minor- oder gar Major-Version.

Nächtliche Builds oder (teils ungetestete) Pre-Release-Versionen werden mittels Suffix angegeben.

©

Baeldung - A Guide to Semantic Versioning