Developer Guide

Ways of working

Kamran Zafar

Agenda

- Development environment
- Developing, building and testing code
- Branching model
- Promotion and release
- Definition of Done

Development Environment

- Core tools
    - IDE (**IntelliJ**, **Eclipse**, Netbeans etc.)
    - Java (**JDK 1.8**)
    - Build tools (**Maven**, Gradle etc.)
    - Version control software (**Git**)
- Project specific tools (and for local integration testing)
    - Docker (for containerised microservices)
    - IBM Websphere (Legacy applications)
    - AWS CLI (For cloud based services)
    - Postgres, Oracle etc. etc.

Development Environment Setup 1

Install and setup both core and project specific tools on host machine.

Advantages

- Easy setup

Disadvantages

- Environment Clutter
- Configuration management overhead
- Configuration conflicts
- Tooling incompatibility
- Unnecessary resource utilisation
- Not so easy to test

Development Environment Setup 2

Install only core tools on host machine and virtualise the rest.

Advantages

- Clean host development environment
- Isolated project test pods
- Spin up environment and tooling when needed
- Optimum resource utilisation
- No configuration/tooling conflicts and incompatibilities

Disadvantages

- Learning curve
- Initial setup using virtualisation tools like **Vagrant**

Develop, build and test code locally

- Code must build locally and all checks (static code analysis, unit tests etc.) should pass

- Integration and deployment testing should be done locally using relevant services and containers (databases, app server, docker etc.)

- Unit test cases must be written for each newly created code-unit (class, method etc.)

- Existing unit test cases must be kept up-to-date in case of changes to existing code

Branching Model

- Two primary branches

    1. Master

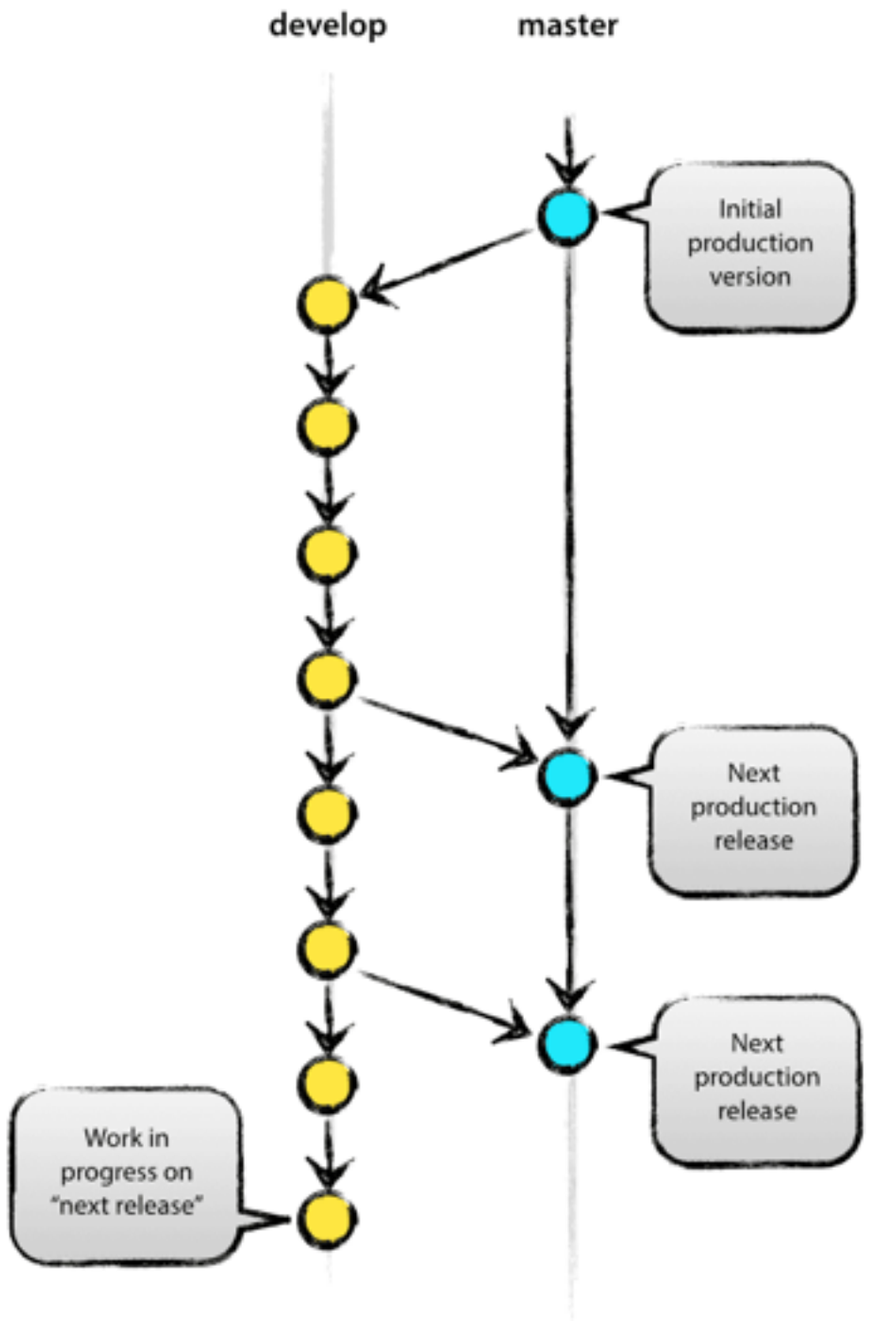
        - In-Sync with Production

        - Stable releases

    2. Development

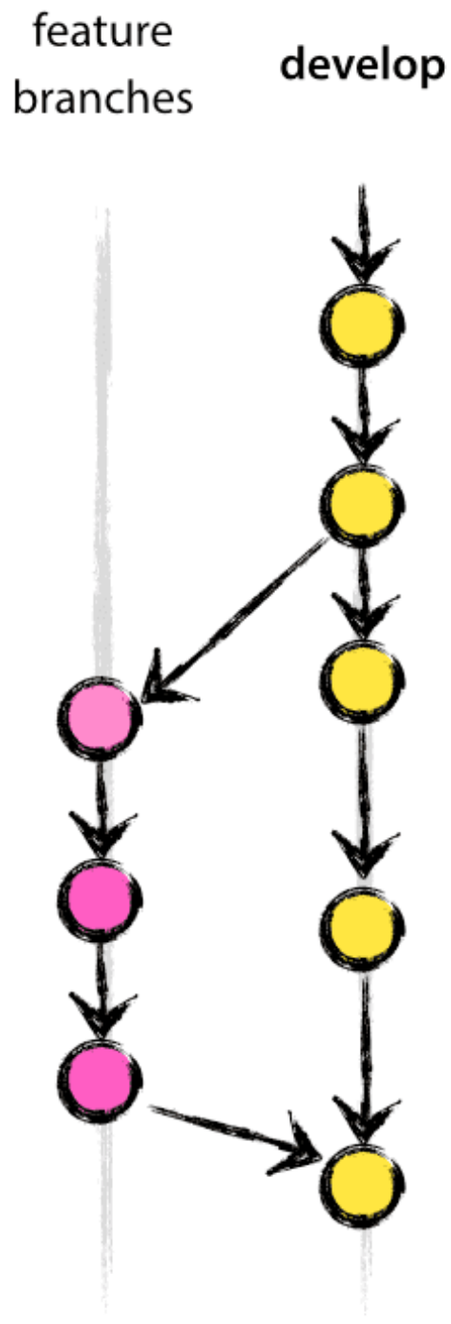
        - Working branch

        - Snapshot releases



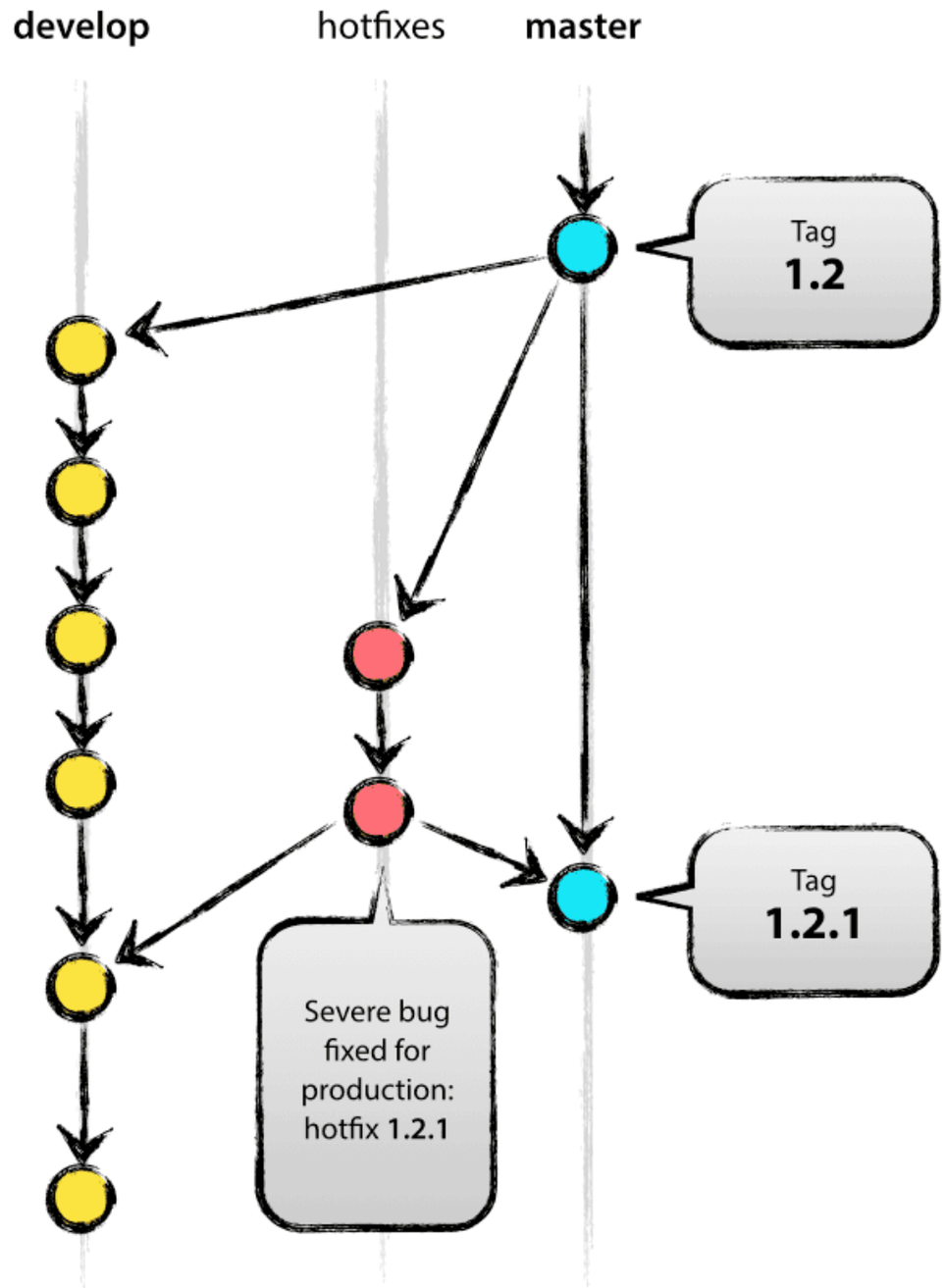
Branching Model

- Multiple Feature branches
  - Branched off development
  - Merged to development once feature is ready to be functionally tested
  - The merge to development builds and deploys snapshot
  - The merge-commit is tagged as “Release candidate”



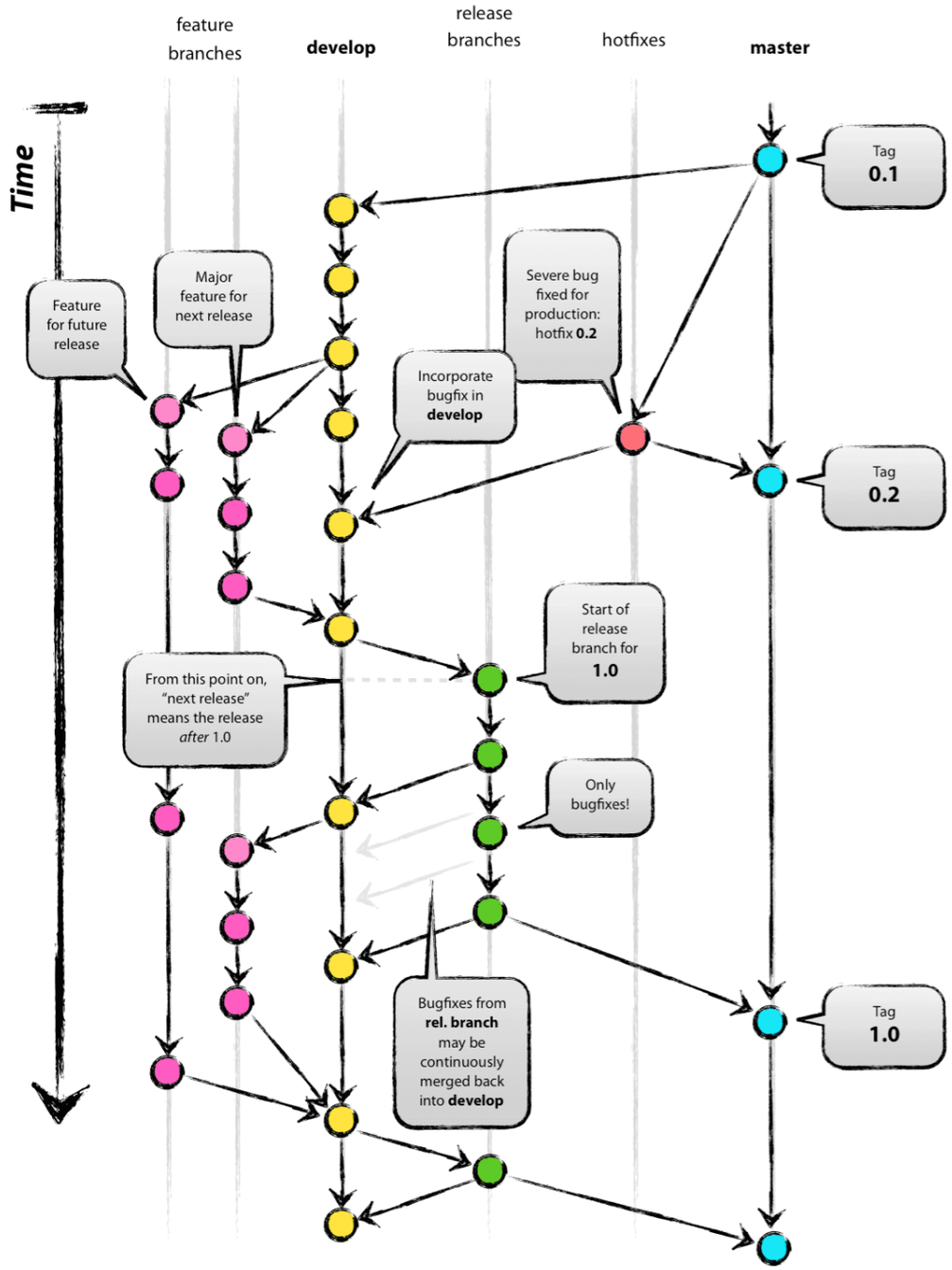
Branching Model

- Hotfix branch
  - Branched off master
  - Deploys to test environment once fixed
  - Merges to master after promotion to production
  - Merges to development



Branching Model

- Release branch
  - Branched off development when features need to be released
  - Release build promoted and tested on SIT and PPTE
  - Released to production
  - Merged to master
  - Merged to development
  - Commit to master is tagged as “Release” with version



Promotion and Release

- All snapshot artefacts are published to “build” repository and are not promoted beyond the test environment.

- All release artefacts can be promoted to “verify” repository for SIT/PPTE deployment and testing.

- Once a release artefact is stable, it can then be promoted to “release” repository for deployment to production.

Definition of Done

1. Story kickoff
   - Requirements
   - Acceptance criteria
2. Development completed
3. Code unit tested
4. CI updates (Jenkins pipelines)
5. Sonar passed (Owasp, Claire scan, findbug, checkstyle, code coverage)
6. Code walkthrough and peer-review
7. Functional tests passed (acceptance criteria met)
8. Regression tests pass
9. NFRs met
   - Performance SLAs
   - Security compliance (CSAM requirements)
10. Documentation