Branching strategies 101 for data professionals

Abstract

Branching strategies 101 for data professionals

Branching strategies are essential for any data professional - whether you're a DBA, DEV or BI professional.

Even as a data professionals you can create and manage different versions of your code, data, and models without affecting the main branch/definition of your code, data, and models.

Branching also enables easy collaboration, experimentation, and testing in a safe and organized way.

In this session, we will introduce the basic concepts of branching, such as branch creation, merging, and deletion.

We will also discuss some common branching strategies, such as feature branches, release branches, and hotfix branches.

Finally, we will put this into practice using Git as well as try and provide you with some best practices and tips for using branching effectively in your data projects.

Branching strategies 101 for data professionals

Karim Ourtani

hello@karimourtani.com

karimourtani

karimourtani

karimourtani

Developer

Flemish Government
Departement of Care

Vlaanderen
is zorgzaam en
gezond samenleven

Board member
dataMinds – Belgian Data Platform Usergroup

Track owner Data, BI, Power BI, AI
Techorama (BE)

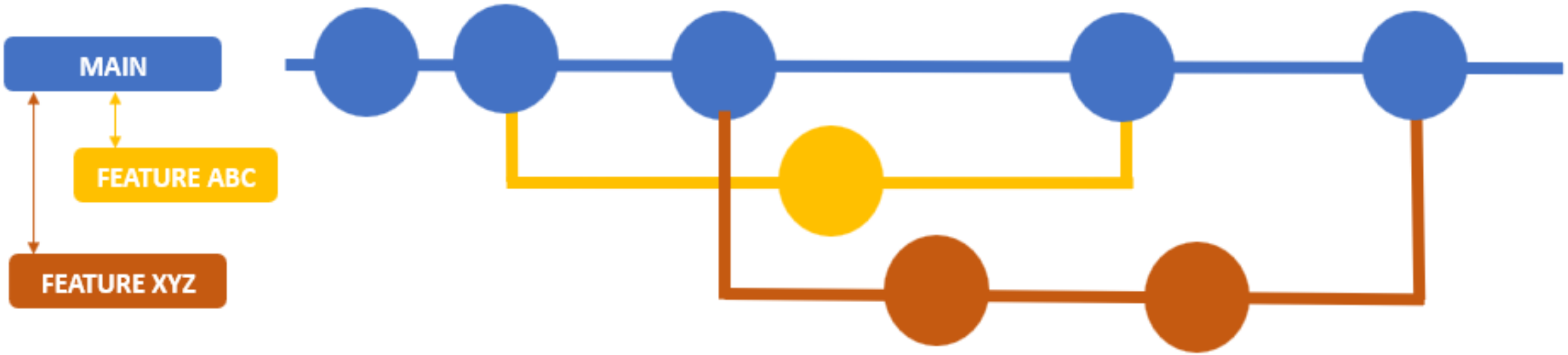
Geek
TRS-80 / GW-Basic - SQL 6.5 / VB 3.0

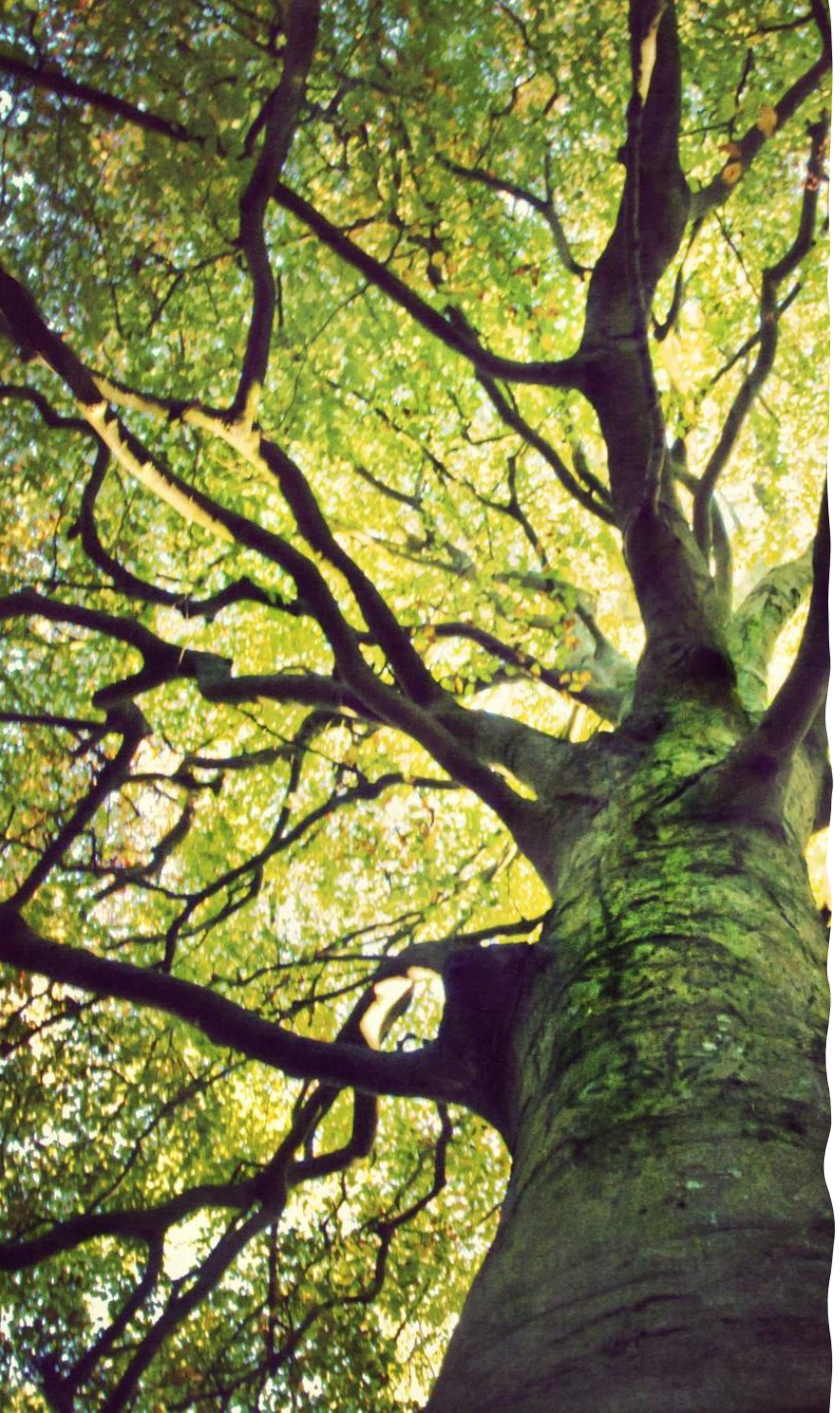
Hobbies
Travel - Thrifting - Gardening - Lego

Agenda

- What is branching?
- What types of branches are there?
- Best practices?
- Two most common strategies

What is branching?





Common types of branches

- Main Branch:
  Serves as the official release history.
- Develop Branch:
  Acts as an integration branch for features.
- Feature Branches:
  Used to develop new features that will eventually be merged back into the develop branch.
- Release Branches: Support preparation of a new production release.
- Hotfix Branches: Allow for quick fixes to the released version.

What are best practices?

Keep Your Branch Strategy Simple

Feature Branch Workflow

Main Branch Quality

Collaboration and Review

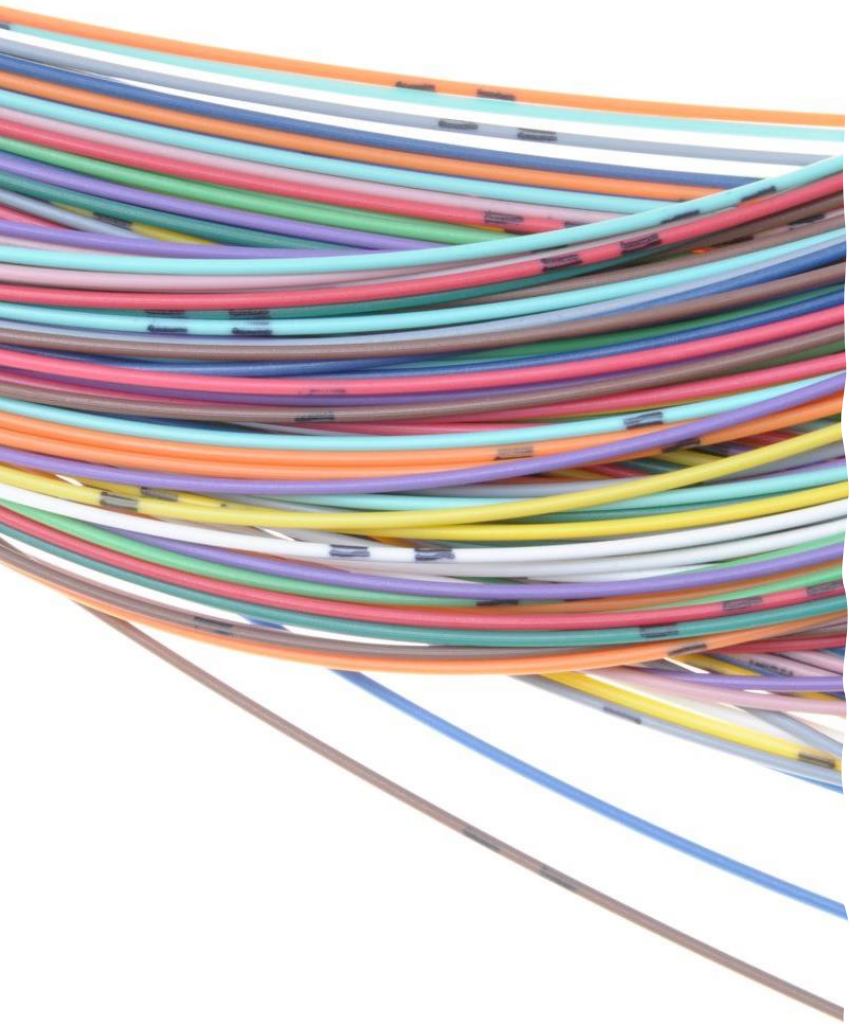
Branch Naming Conventions

Keep Your Branch Strategy Simple

- Use feature branches for new features and bug fixes.
- Merge feature branches into the main branch using pull requests.
- Maintain a high-quality, up-to-date main branch.

Feature Branch Workflow

- Develop features and fix bugs in feature branches off the main branch.
- Isolate work in progress from the completed work in the main branch.
- Name feature branches by convention for easy identification.



Main Branch Quality

- Ensure the code in the main branch passes tests and builds cleanly.
- Feature branches should start from a known good version of the code.
- Implement a branch policy that requires pull requests to merge code into the main branch.

Collaboration and Review

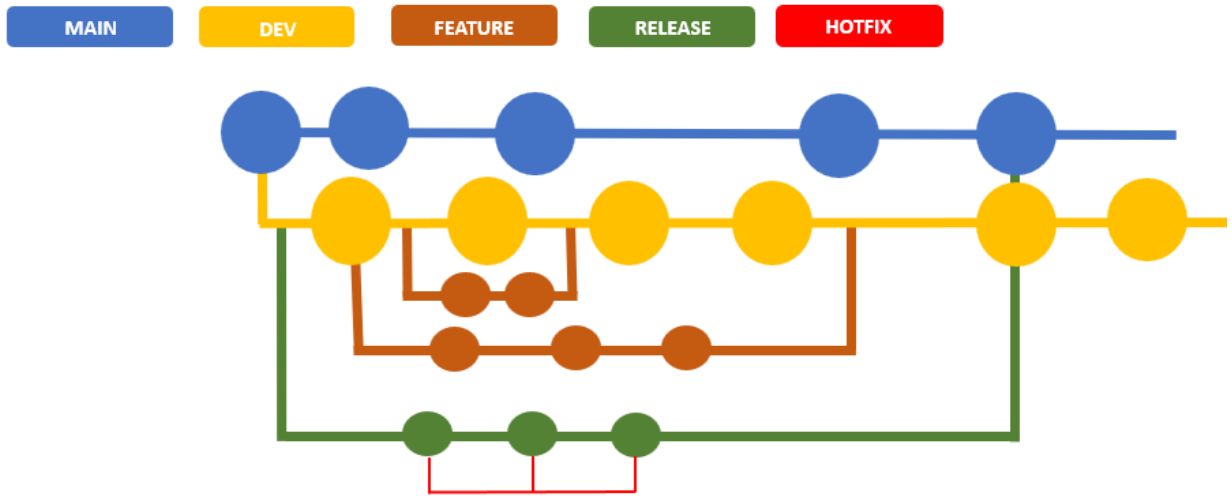
- Use pull requests for code review to improve code quality.
- Distribute review responsibilities to spread knowledge across the team.

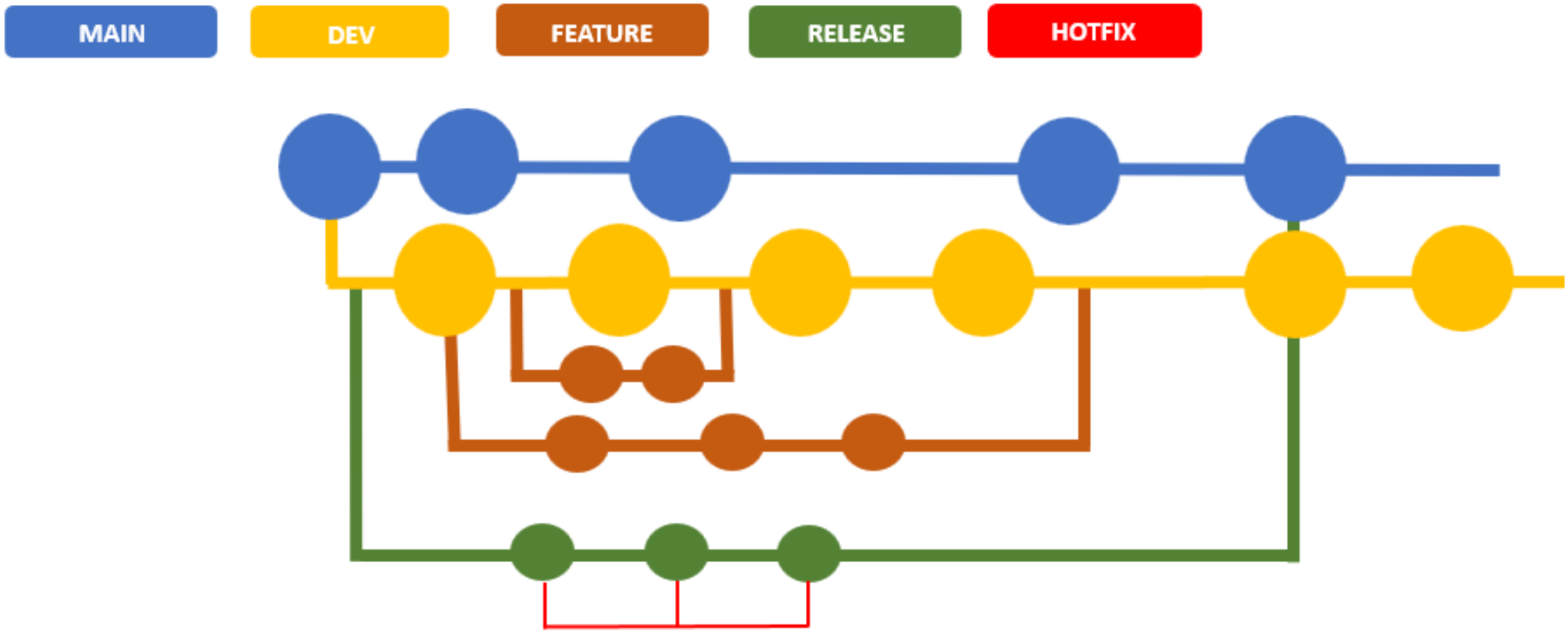
Branch Naming Conventions

- Adopt a consistent naming convention for branches.
- Example formats:
  - feature/feature-name
  - bugfix/description
  - users/username/workitem
  - feature/featurid-feature-name

Strategies: GitFlow Branching Model

- **Advantages**
  - usually ensures a clean state of branches at any given moment in the life cycle of the project
  - branch development is totally separated from the production (main branch)
  - the main branch is always or almost ready for release
  - the branches naming follows a systematic pattern making it easier to comprehend
  - gitflow offers a dedicated channel for hotfixes to production.
- **Disadvantages**
  - Many branches can become difficult to manage
  - features can take days to merge because a lot of changes are committed to the development branch without being integrated, tested previously.
  - long-lived branches is a large maintenance headache for a team
  - complex work process complexity
  - slow feedback
  - heavy branch overhead

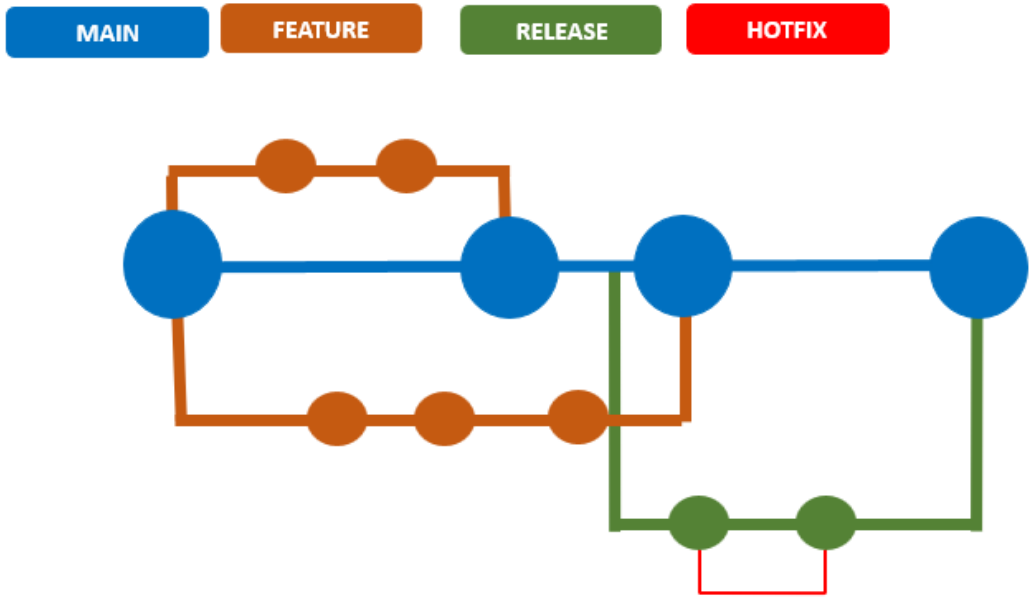


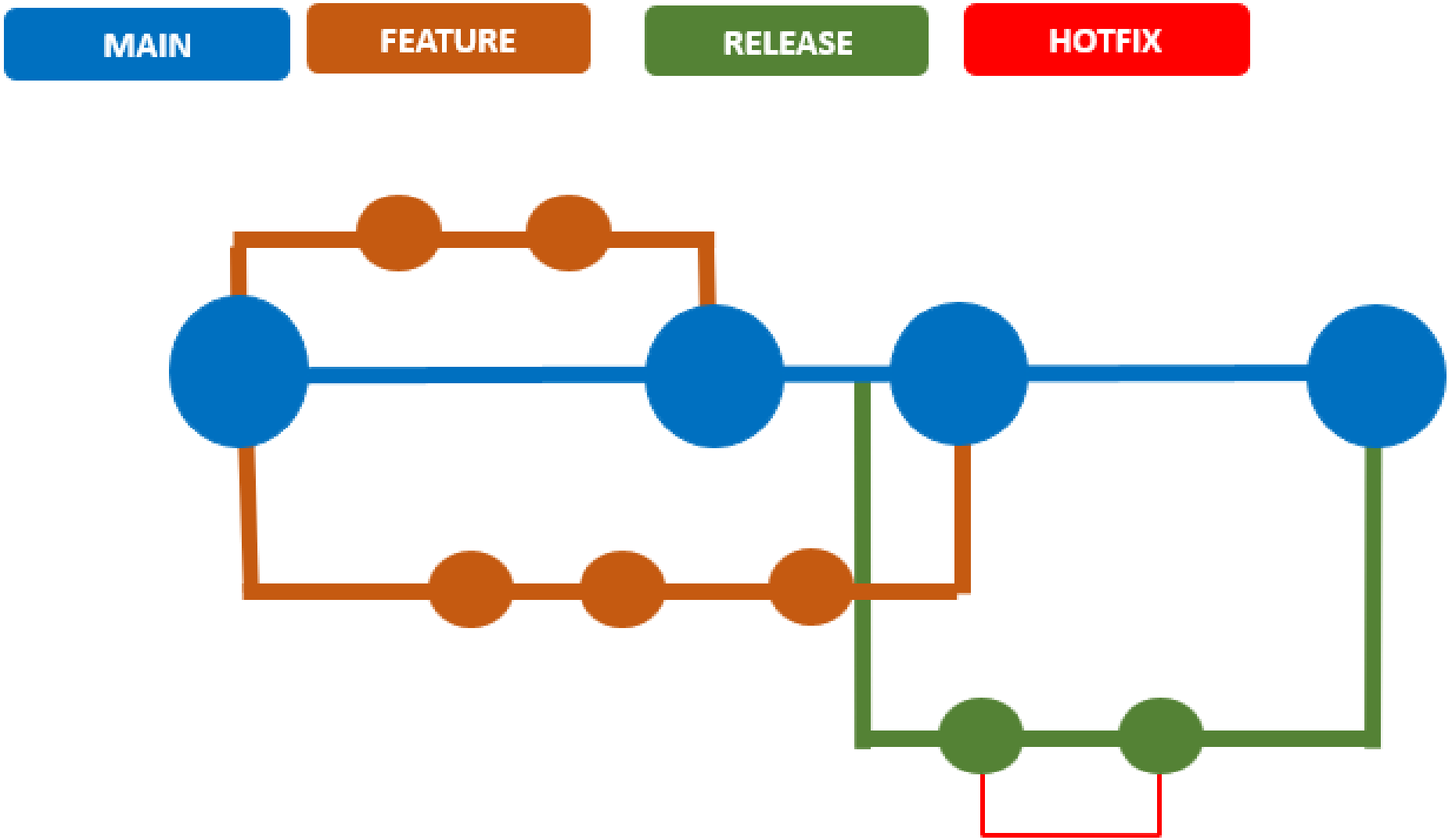

MAIN
DEV
FEATURE
RELEASE
HOTFIX

Strategies: Trunk Based Development Model

- **Advantages**
  - truly Continuous Integrations
  - small changes
  - feature flags
  - branch by abstraction

- **Disadvantages**
  - contention collision
  - Can be more complex at times
  - desired High team maturity
  - strong automation
  - test a lot before push




MAIN
FEATURE
RELEASE
HOTFIX


DEMO

hello@karimourtani.com

karimourtani

karimourtani

karimourtani