Branch-based Pros

Development Strategies

- Parallel development of several features
- "Each feature is self-contained in a branch"
- Facilitating parallel and flexible development of features
- "The same repository being worked on by different teams."
- Flexibility in development and collaboration
- "The division of work and modularization of tasks becomes natural"
- Organization and Synchronization of Development
- "One of the main factors is the organization of the code and the division of tasks"
- Management of Development Phases
- "a branch based model to clearly divide what is still under development, what is in each release, what will be a correction and from which release"

Organization, Control and Security

- More centralized control over changes and additions
- "The flow of changes is extremely organized, in addition to isolating each part in a specific branch."
- Code Assurance and Quality Control
- "Security in a productive environment, shared work by feature, CI/CD Tool (TeamCity), Code review"
- Review Process and Change History
- "Changes are reviewed before being integrated. Change history is richer as comments and discussions are recorded in the pull request."
- Team size and defined processes
- "It is most used in large teams, which have a very well-defined and automated structure of software quality processes."
- Code Testing and Validation
- "Testing and validating the code before it is merged into the main branch of the project facilitates the detection of errors and bugs"

Environment Management and Deployment

- Deployment and QA Management
- "When deployments do not need to be frequent or there are extremely rigid windows"
- Understanding and Managing Code
- "It makes it viable to maintain code that has already been released, without changing the flow of the current and future release"
- CI/CD
- "The CI/CD system allows you to view the behavior of each branch in an isolated production environment, which does not affect the main one."

Branch-based Cons

Project Complexity

- Complexity and overhead in repository maintenance due to excessive segmentation and complex product history
- "Too many branches make it difficult to manage the project as a whole, it is necessary to constantly 'map' the flow, in addition to the amount of time
- Delivery Time
- "Maintaining epic branches that take a long time tends to require either rebases or merges with updates"

Team Size and Maturity

- Team Size
- "According to the number of teams working on it, the effort in merge activities increases"
- Very large teams where requirements change a lot
- "more documentation and steps for releasing features, and for switching between project contexts"
- Solo Devs and Prototyping/MVP
- "Small teams (1-3 people) physically working together can coordinate well without needing this model"

Organizational and Process Factors

- Organizational Factors
- "Multiplication of branches. If branches are too long-lived, they become outdated and lead to conflicts.""
- "After development, merge and/or rebase processes. Inexperienced programmers have difficulty/fear of merging"
- "Low Production Code Security"
- "demands more documentation and steps to exchange between project contexts and to launch features"

Ease of Use & Complexity

- Simplicity, Clarity, Fast Feedback and Focus on Delivery
- "For those who are not familiar with the standard operation, the learning curve of the flow can be substantial"
- Complexity and Risks of Loss of Information
- "Complexity, loss of information if the developer forgets to send the branch to the remote repository."

I am not sure

Trunk-based Pros

Simplicity and Ease of Use

- "few people maintaining the project, ease, easy maintenance"
- "Speed, less complexity, use in personal projects"
- "Fewer risk factors and fewer merge failures"
- "Immediate application of the code."

Code Stability and Control

- "It's easier to have the team aligned on the flows to follow when there are fewer branches and steps in a remote work context."
- "Greater chance of keeping the code stable and immutable, with fewer risk factors and merge failures."
- "In this case, it only makes sense if there are very few developers. A maximum of 3. Otherwise, there could be a lot of problems with code conflicts and the occurrence of poorly mapped systemic errors"

Agility and Speed in Development

- "Speed in software delivery"
- "Development occurs faster. Good for applications that require fast feedback."

Team Experience and Maturity

- "Culture of continuous delivery, aggressive use of feature flags, good application monitoring structure, atomic deployments and using rollouts, experienced team, smaller applications or those that require quick feedback."
- "Baseline for new project setup"
- "More experienced teams, fast deliveries"
- "Smaller teams that do not share the git repository with other teams"
- "Small projects, stabilized codebase, high maturity of the development team"

Project Specificity

- "When the code is deployed directly into production, either for ease of correction and use of feature flags. Another case is when there is only 1 dev working on the project"

Trunk-based Cons

Project Scalability and Complexity

- "With several members, you don't have much control when several developers use just one branch."
- "Large teams where the main branch version changes multiple times over the course of a single day."
- "If there are a lot of people making commits at the same time, it becomes difficult to keep the code updated."
- "Inexperienced teams. Large team, complex code base"

Quality Management and Control

- "Difficult quality control, prone to frequent breaks and interference between commits/pushes."
- "Once a wrong commit can be very difficult to revert in just one branch."
- "Low team maturity, rapid functionality increments."
- "When there is code review in geographically distributed teams. Larger teams (>3 people)"

Code Development and Maintenance

- "More steps and greater complexity for code changes."
- "Complexity in orchestration and rebasing when there are differences with the master."
- "Incomplete or bugged code can be more easily generated in the project trunk."
- "Interference between commits/pushes."
- "Extensive testing, greater complexity in the code, use of toggle to enable features, this generates code duplication"
- "Incomplete or buggy code can be generated more easily in the project trunk."

Project Requirements and Needs

- "Applications with strict deployment windows, time-consuming continuous integration processes, teams with little CD experience, lack of application monitoring and feature toggles, as well as complex deployments with many dependencies."
- "Need to have multiple versions, as in the case of mobile applications."
- "Lack of feature toggle."

I am not sure

- "Prone to frequent breakages"
- "Difficulty in quality control"
- "Difficulty controlling versions"
- "Test Security"
- ""Difficult to monitor the application"