Version control strategies

1. **No branches** Your team works only from the main source tree
2. **Branch for release**. Your team creates a branch before release time to stabilize the release. Then merge changes back into main source
3. **Branches for maintenance**. Your team creates a branch to maintain an old build – a released version that will need maintenance.
4. **Branch for feature**. You can create separate branches for work on specific features to be completed in parallel
5. **Branch for team**. You branch to isolate sub-teams so they can work without being subject to breaking changes.

What to choose

- If producing release versions that you'll sell & maintain
  - Use strategy 3 and always have that branch to maintain
- If you push out updates all the time
  - Use strategy 2 to pull releases off to the side, but eventually forget them
  - Continue to develop in the Head

Continuous integration

- All code changes go directly into mainline of version control
  - Automatically builds complete software as submitted
  - Automatically release-test all new contributions
- Always have a working mainline in your version control
  - Smart servers can rebuilt only bits that are changed.
- Supports TDD – all submissions need to integrate successfully
- Integration [and testing] is figured out from the start
- It identifies bugs quickly, because integration happens sooner

Build configurations

- Configuration scripts – specify how different versions are built
- Automatic build servers run these scripts to create working versions of the software
- Configuration for a new version includes
  - Versions of components included
  - Which platform-specific libraries to bundle in
  - Which test suite to perform
  - Where to deploy the build
- A **release** (or several) is then build based on this configuration