Software Development Lifecycle (SDLC)

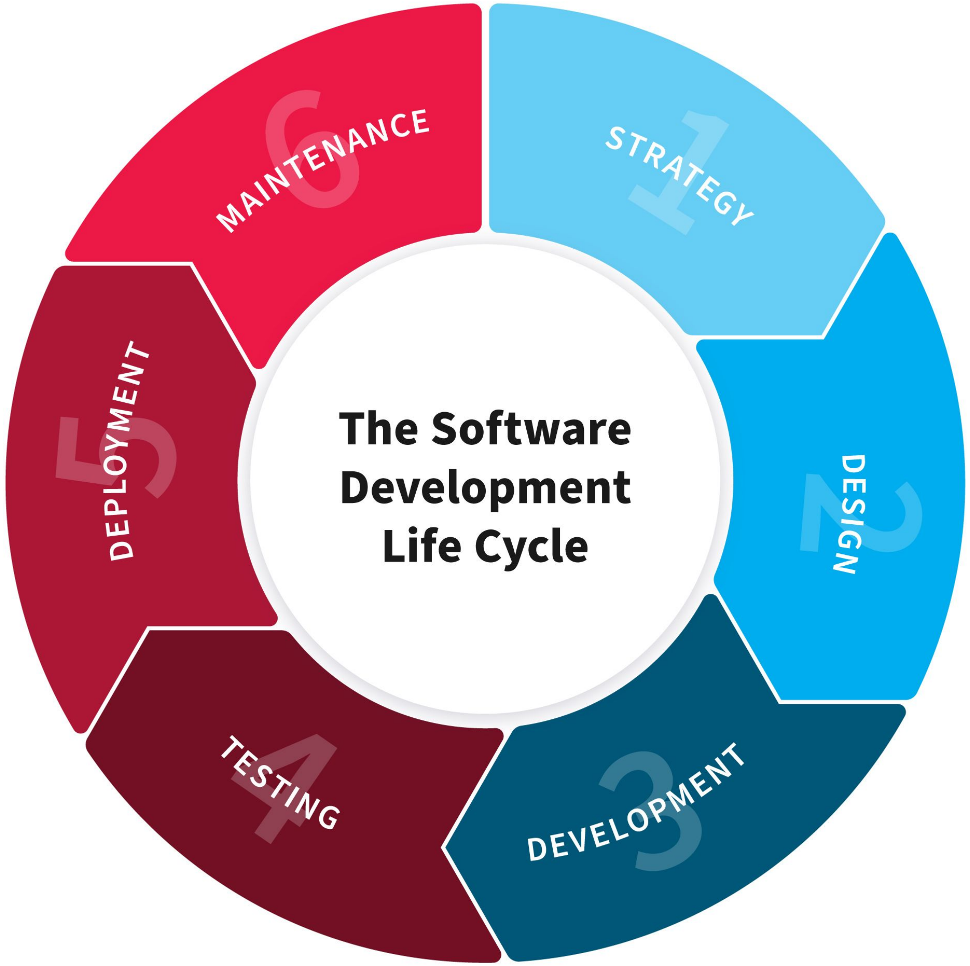
What is SDLC?

Your everyday software process/workflow

- Creating git branches, commits, merges, …

People usually contribute two types of code:

- Feature
- Bug fixes


The Software Development Life Cycle
1 STRATEGY
2 DESIGN
3 DEVELOPMENT
4 TESTING
5 DEPLOYMENT
6 MAINTENANCE

Where does DevOps come?

Strategy, Design, Development, & Testing

- Even you might not be writing code during those stages, you should agree with the dev team(s) on certain practices like
  - Writing tests
  - Writing logs (prints) and send them to you
  - What happens when a bug is discovered
  - What happens when the app goes down/becomes unavailable
  - Anything you think will come up in production.

Where does DevOps come?

Strategy, Design, Development, & Testing

- It can go deeper into ensuring availability (for example 99.99% uptime), latency, … .
  Although that might get into Site-Reliability Engineering (SRE); which is a little bit more advanced than a DevOps role.
  https://cloud.google.com/blog/products/devops-sre/sre-fundamentals-slis-slas-and-slos

Where does DevOps come?

Deployment & Monitoring

- You’ll work more on this part.
- Making sure you make the app/project available to clients
- Making sure you have a proper way of collecting the logs and metrics
- Ensuring you know the health status of your app/website/service/…

So, what can be used to do this?

- Version Control Service (github, gitlab, azure repos, …)
- Gitflow

How can Gitflow help?

It defines a standard way for your software development workflow

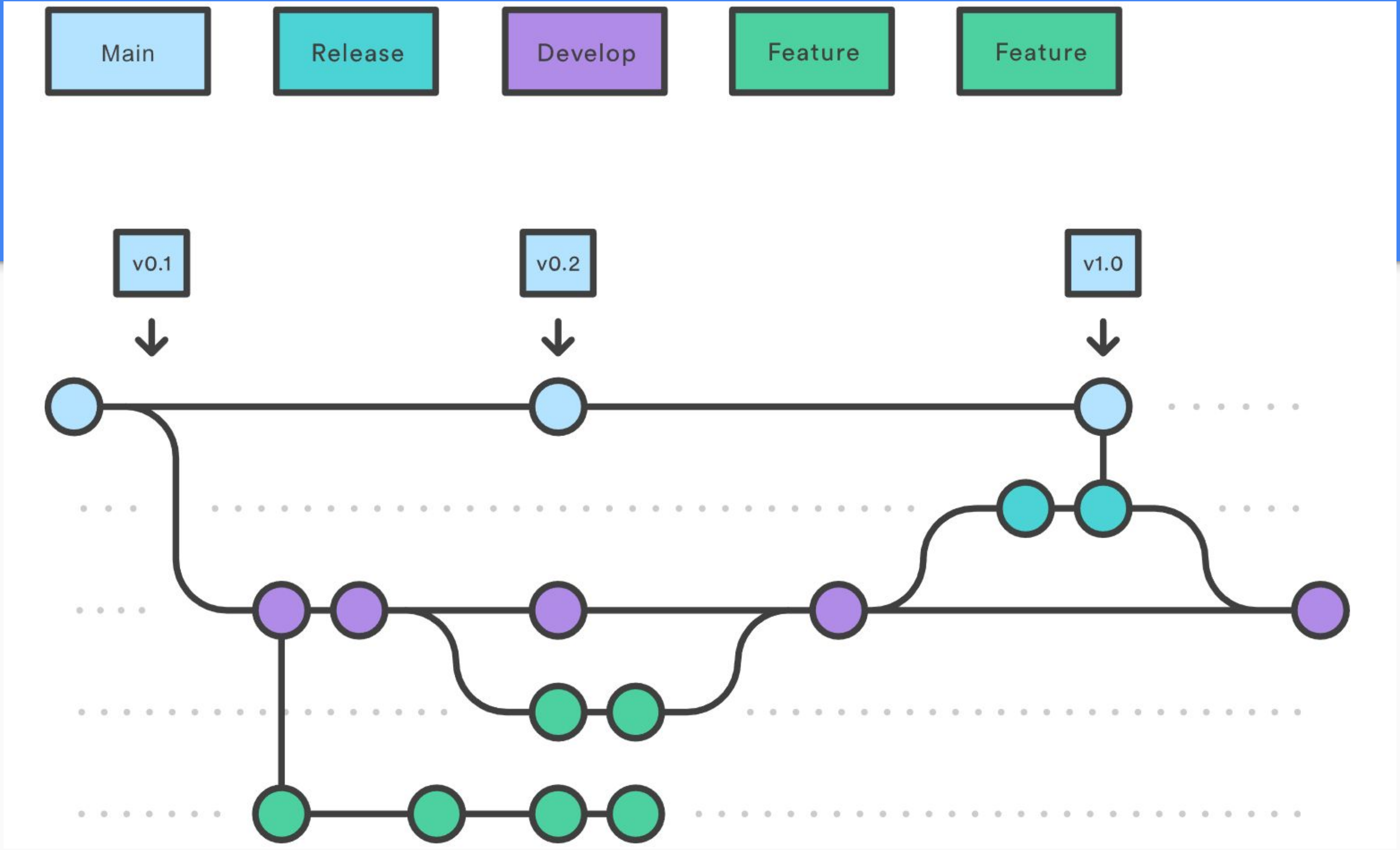
- Production code comes from branch `main`
- Next-Release code comes from branch `develop`, which is the default branch and normal development is done on it.
- Any new feature gets merged into `develop`
- Any hotfix gets merged into both `main` and `develop`
- When it’s time for a new release, we create a new release branch, make sure it’s good, then merge it into `main`.
- Git tags are used to point out versions in version control

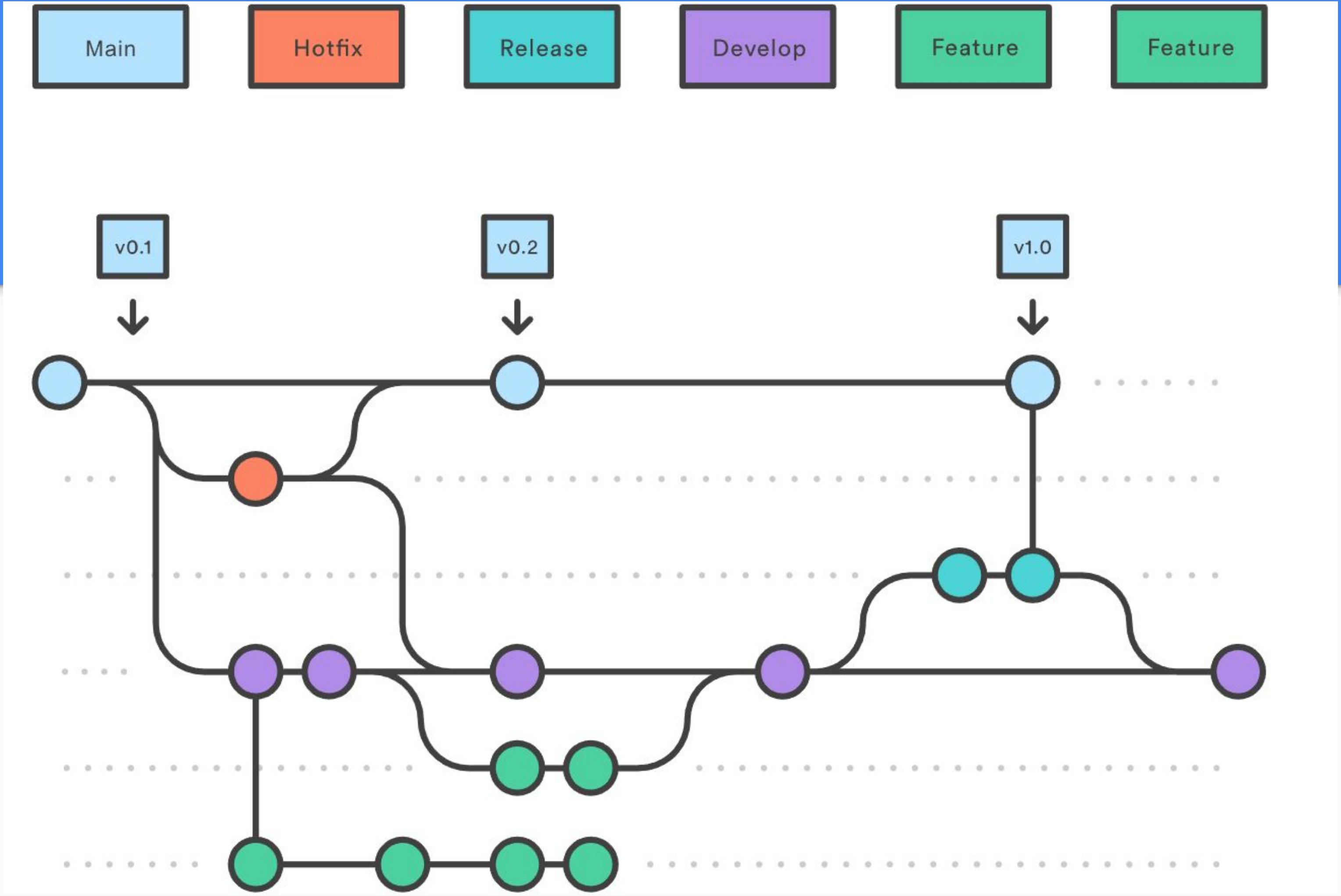
How to version?
Semantic Versioning 2.0.0 (SemVer)

Helps you standardize the app/project versioning;

- MAJOR.MINOR.PATCH (for example, 1.7.3)
- You only increment MAJOR if backwards compatibility is broken
- You increment MINOR for backward-compatible changes/updates
- You increment PATCH for backward-compatible bug fixes

More details: https://semver.org/spec/v2.0.0.html


Main
Release
Develop
Feature
Feature
v0.1
v0.2
v1.0


Main
Hotfix
Release
Develop
Feature
Feature
v0.1
v0.2
v1.0

More about gitflow

https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Github, Gitlab, …
Branch Protection

- Disallow direct push to main or develop
- Require a Pull Request for any code contribution (feature or bug fixes)
- Minimum Number of Approvals to allow Merge
- Passing the existing testing suite to be able to Merge