Integration Manager Workflow in Git

Integration Manager Workflow

- Suited for team-based projects, especially large scale and open-source projects.
- Commonly used in open-source and large-scale projects where contributions come from many developers, and a central authority (integration manager) oversees the quality and integrity of the main project branch.

Key Concepts of Integration Manager Workflow

- Integration Manager Role
- Branching Strategy
- Code Review and Quality Control
- Pull Requests and Merging
- Main Repository and Forks

Role of the Integration Manager

- Reviews and merges code changes
- Acts as the gatekeeper for the main project branch
- Contributors submit changes for review

Branching Strategy

- Contributors work on their own branches (feature/bug-fix)
- Only Integration Manager has write access to the main branch

Code Review and Quality Control

- Integration Manager reviews the code
- Tests the contributions
- Provides feedback and ensures quality and consistency

Pull Requests and Merging

- Contributors submit Pull Requests (PRs)
- Integration Manager reviews, discusses, tests, and merges the PRs

Main Repository and Forks

- Central repository maintained by the Integration Manager
- Contributors work on forked copies to avoid impacting the main project

Benefits

- Quality Assurance: The integration manager checks each contribution for bugs and inconsistencies before merging, helping maintain high-quality code.
- Scalability: This workflow supports multiple contributors by isolating each person’s work until it’s reviewed, making it ideal for open-source and large projects.
- Reduced Merge Conflicts: By requiring contributors to fork the repository and submit pull requests, merge conflicts are minimized in the main repository.

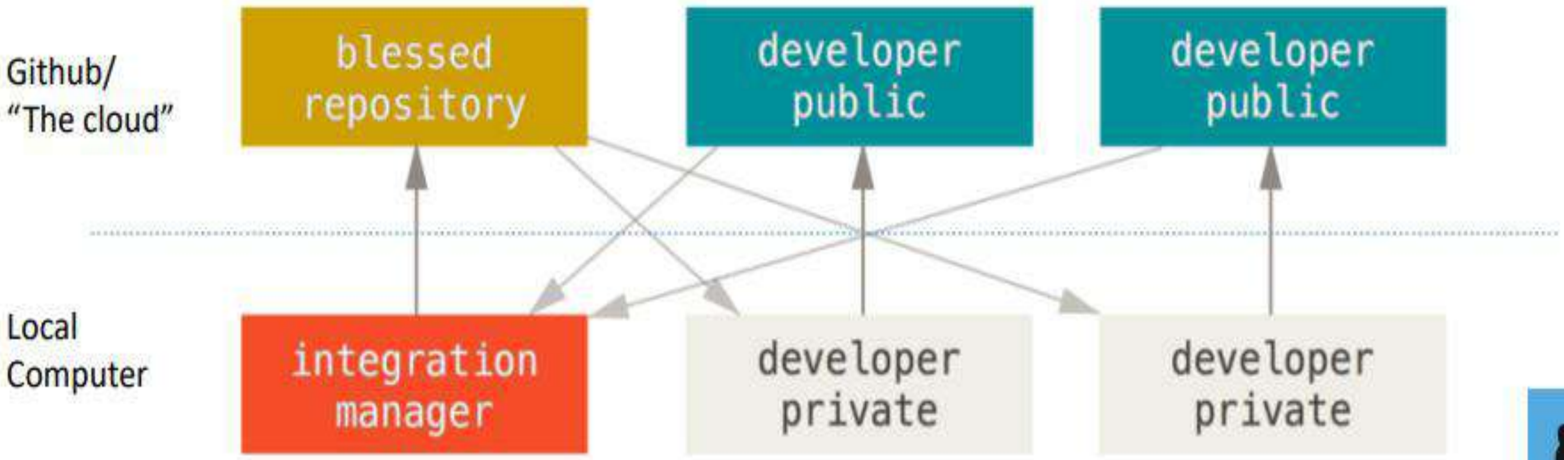
Typical Use Cases

- Open-Source Projects: Many contributors working on different features or fixes.
- Enterprise Projects: Managed by lead developers
- Controlled Environments: Where code quality and consistency are priorities, requiring strict review processes before integration.

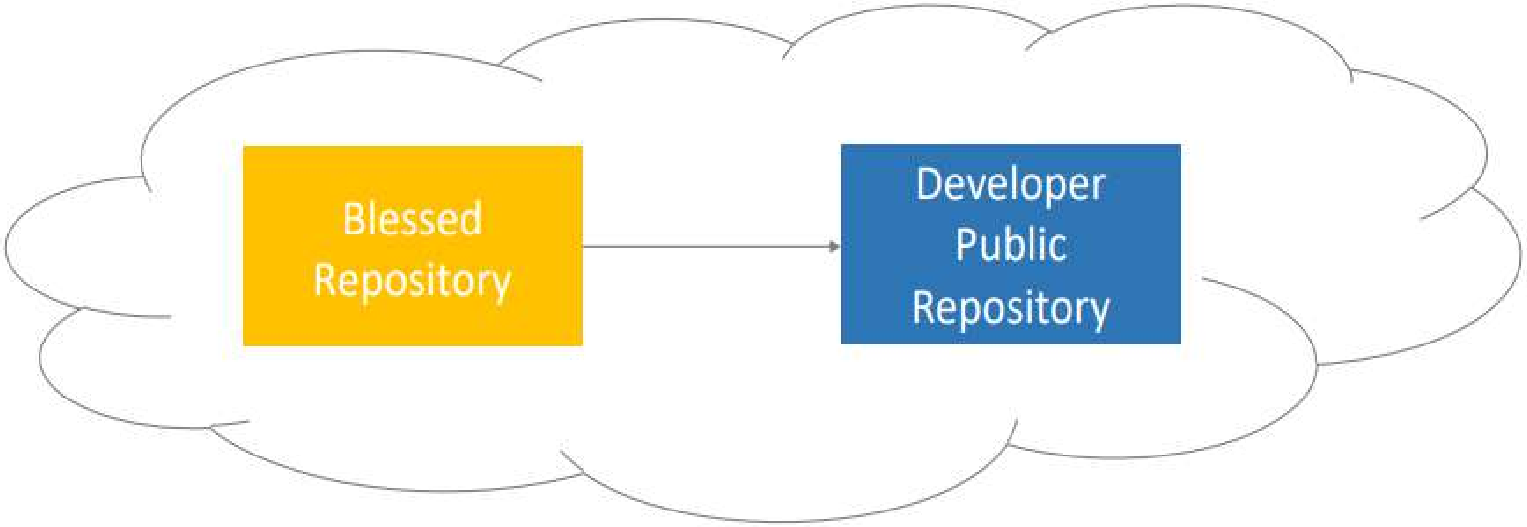
Example Workflow

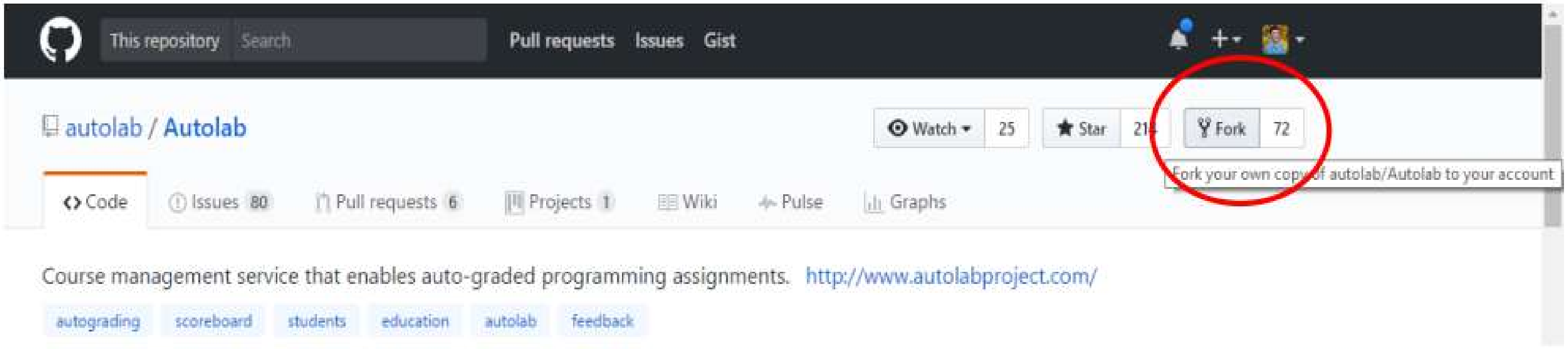
- Step 1: Integration Manager maintains the main repository
- Step 2: Contributors create forks for feature/fix development
- Step 3: Contributors submit Pull Requests(PRs) after changes
- Step 4: Integration Manager reviews, provides feedback, and merges PR into main branch.

Integration-Manager Workflow



Step 1. **Fork** the public repository





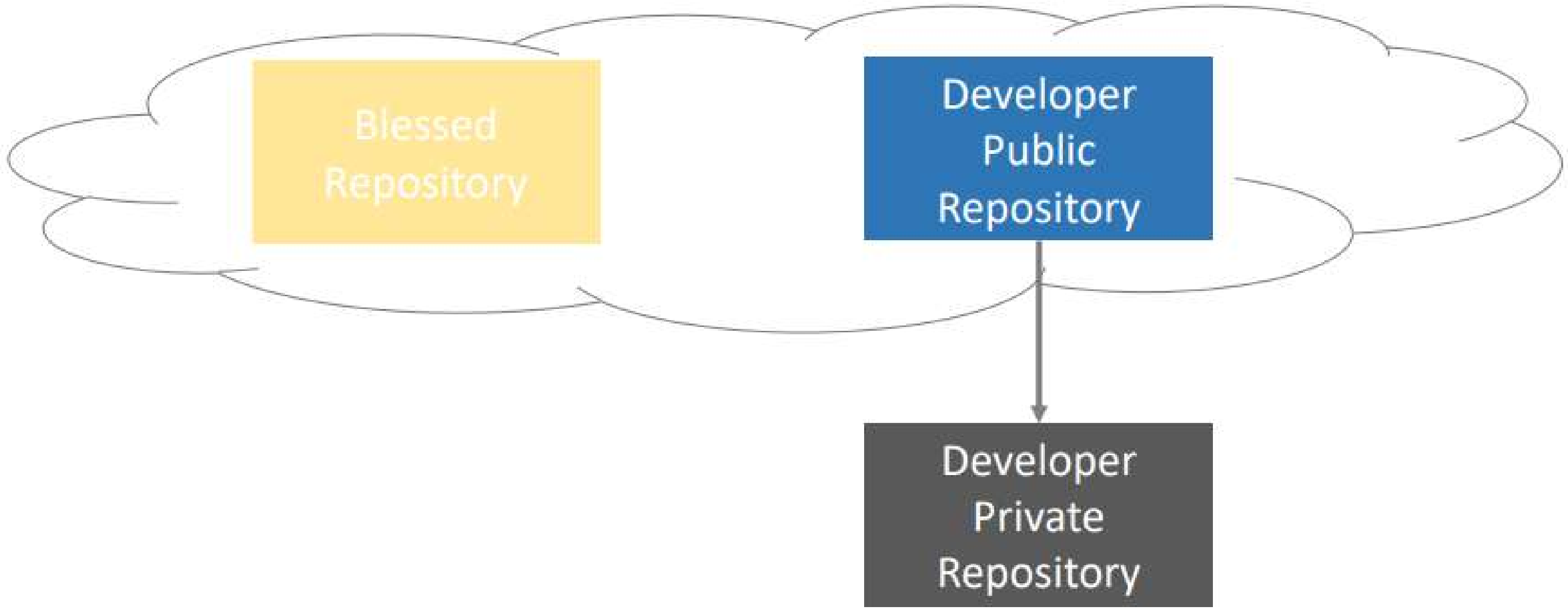
Fork your own copy of autolab/Autolab to your account

Course management service that enables auto-graded programming assignments. http://www.autolabproject.com/

autograding scoreboard students education autolab feedback

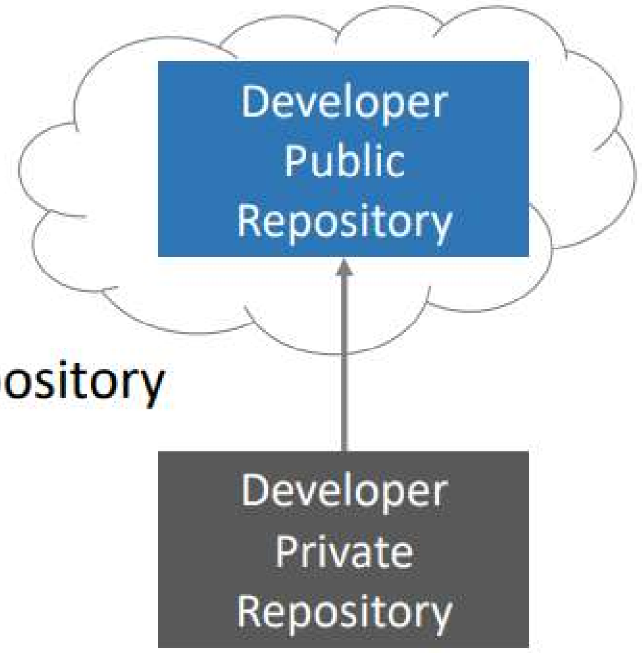
Step 2. Clone your public repository

```
$ git clone https://github.com/aperley/Autolab.git
```



Step 3. Create a **feature branch** and make some commits

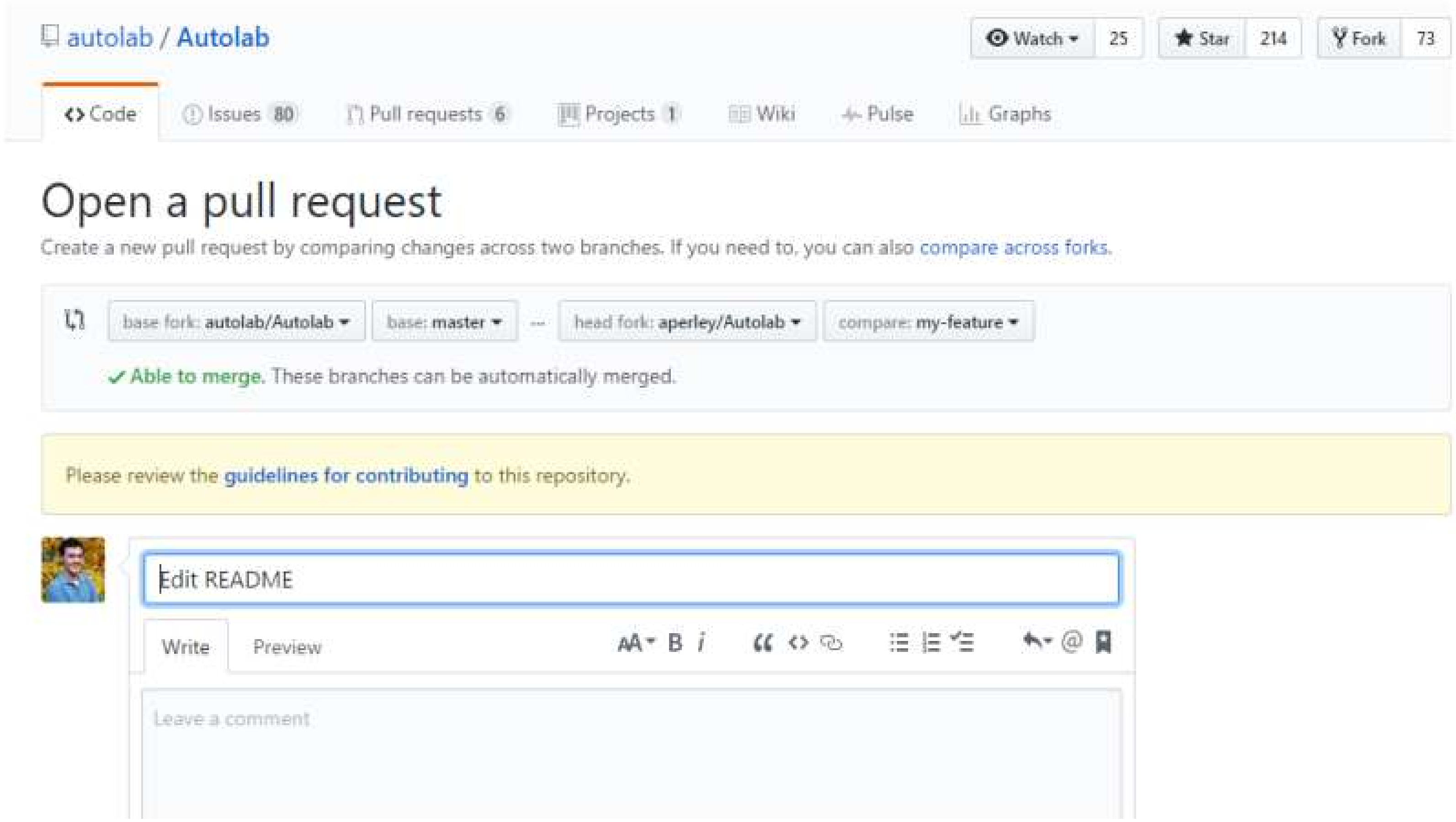
```
$ git checkout -b my-feature
$ <do some work>
$ git commit -am "add my feature"
```



Then **push** your feature branch to your public repository

```
$ git push origin my-feature
```

Step 4. Create a **pull request**



autolab / **Autolab**

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base fork: autolab/Autolab ▾ base: master ▾ ... head fork: aperley/Autolab ▾ compare: my-feature ▾

✓ Able to merge. These branches can be automatically merged.

Please review the **guidelines for contributing** to this repository.

Edit README

The integration manager can inspect and **pull in** your changes

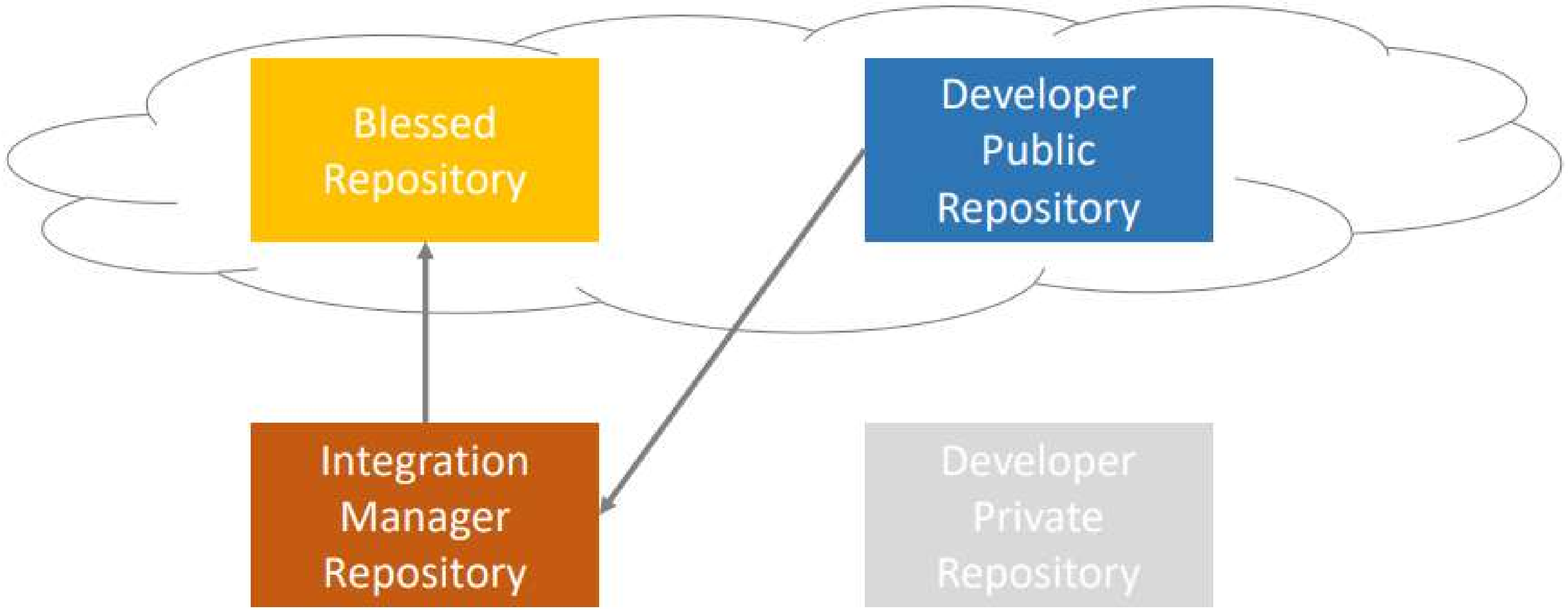
As the integration manager:

```
$ git remote add aperleys-fork
https://github.com/aperley/Autolab.git
$ git checkout aperleys-fork/my-feature
```

If it looks good:

```
$ git checkout master
$ git merge aperleys-fork/my-feature
$ git push origin master
```

The integration manager can inspect and **pull in** your changes

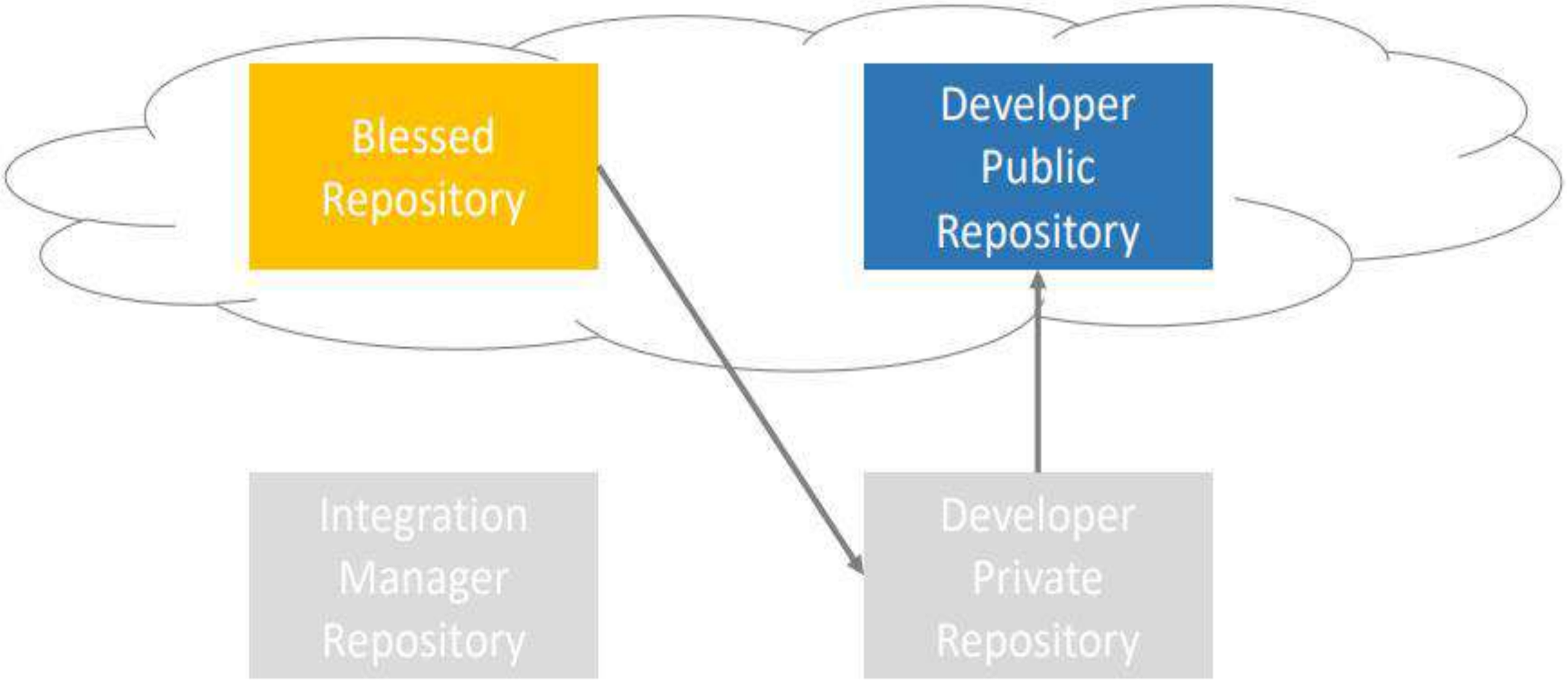

Blessed Repository
Developer Public Repository
Integration Manager Repository
Developer Private Repository

You need to keep your fork up to date

In the private developer repo

```
$ git remote add upstream
https://github.com/autolab/Autolab.git
$ git fetch upstream
$ git checkout master
$ git merge upstream/master
$ git push origin master
```

You need to keep your fork up to date


Blessed Repository
Developer Public Repository
Integration Manager Repository
Developer Private Repository