Gerrit is a free, web-based team code collaboration tool. Software developers in a team can review each other’s modifications on their source code using a Web browser and approve or reject those changes. Gerrit is a fork of Rietveld, another code review tool.

### Feature Comparison

<table>
<thead>
<tr>
<th>FEATURES</th>
<th>Community Support</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CORE</td>
</tr>
</tbody>
</table>

**Manage**
- Subrepos
- Audit Events
- Audit Reports
- Compliance Management
- Code Analytics
- Device Reports
- Value Stream Management
- Insights

**Plan**
- Issue Tracking
- Kanban Boards
- Time Tracking
- Epics
- Roadmaps
- Service Desk
- Requirements Management
- Quality Management
- Web IDE
- Live Integration
- Snippets
- Design Management
- Web Performance
- Usability Testing
- Accessibility Testing
- Unique Tests

**Create**
- Source Code Management
- Code Reviews
- Wiki
- Static Site Generator
- Continuous Integration
- Code Quality
- Code Testing and Coverage
- Load Testing
- Container Registry
- Docker Registry
- Dependency Prone
- Container Scanning
- License Compliance
- Secret Detection
- Vulnerability Management
- Feature Flags
- Release Orchestration
- Release Guidance
- Secrets Management

**Verify**
- SAST
- DAST
- Fuzz Testing
- Dependency Scanning
- Continuous Delivery
- Orchestration
- Advanced Deployments
- Containerized Orchestration
- Orchestration
- Static Analysis
- Code Coverage
- Containerized Security
- Container Security
- Traffic Monitoring
- User Tracking
- Product Analytics
- Synthetic Monitoring

**Package**
- Package Registry
- Container Registry
- Helm Chart Registry
- Dependency Prone
- Docker Registry
- Containerized Orchestration
- Orchestration
- Static Analysis
- Code Coverage
- Containerized Security
- Container Security
- Traffic Monitoring
- User Tracking
- Product Analytics
- Synthetic Monitoring

**Secure**
- Container Security
- Containerized Orchestration
- Orchestration
- Static Analysis
- Code Coverage
- Containerized Security
- Container Security
- Traffic Monitoring
- User Tracking
- Product Analytics
- Synthetic Monitoring

**Release**
- Package Integration
- Advanced Deployments
- Containerized Orchestration
- Orchestration
- Static Analysis
- Code Coverage
- Containerized Security
- Container Security
- Traffic Monitoring
- User Tracking
- Product Analytics
- Synthetic Monitoring

**Configure**
- Auto Deploy
- Rollout Orchestration
- Orchestration
- Static Analysis
- Code Coverage
- Containerized Security
- Container Security
- Traffic Monitoring
- User Tracking
- Product Analytics
- Synthetic Monitoring

**Monitor**
- Metrics
- Alert Management
- Incident Management
- Logging
- Traffic Monitoring
- User Tracking
- Product Analytics
- Synthetic Monitoring

**Defend**
- Web Application Firewall
- Container Host Security
- Container Network Security
- Traffic Monitoring
- User Tracking
- Product Analytics
- Synthetic Monitoring
The GitLab Community Forum is an active and vibrant place for all GitLab users to share and seek support.

Visit the GitLab Community Forum

**Required Merge Request Approvals**

<table>
<thead>
<tr>
<th>Core</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>Bronze</td>
<td>Silver</td>
<td>Gold</td>
</tr>
</tbody>
</table>

When a project needs multiple sign-offs, you can require every merge request to be approved before merging. With Required Merge Request Approvals you can set the number of necessary approvals and predefine a list of specific approvers. In turn, guarantee the quality and the standards of your code.

Learn more about merge request approvals

**Multiple approvers in code review**

<table>
<thead>
<tr>
<th>Core</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>Bronze</td>
<td>Silver</td>
<td>Gold</td>
</tr>
</tbody>
</table>

In GitLab, to ensure strict code review, you can require a minimum number of users to approve of a merge request before it is able to be merged. You can undo an approval by removing it after the fact.

Approvals Documentation

**Approval rules for code review**

<table>
<thead>
<tr>
<th>Core</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>Bronze</td>
<td>Silver</td>
<td>Gold</td>
</tr>
</tbody>
</table>

Make sure the right people review merge requests with approval rules by specifying lists of eligible approvers, the minimum number of approvals for each, and which target branches they protect. This makes it easy to request review from different teams like Engineering, UX and Product.

Approvals Documentation

**Optional Merge Request Approvals**

<table>
<thead>
<tr>
<th>Core</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>Bronze</td>
<td>Silver</td>
<td>Gold</td>
</tr>
</tbody>
</table>

Code review is an essential practice of every successful project, and giving your approval once a merge request is in good shape is an important part of the review process, as it clearly communicates the ability to merge the change.

Learn more about optional merge request approvals

**Code Owners**

<table>
<thead>
<tr>
<th>Core</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>Bronze</td>
<td>Silver</td>
<td>Gold</td>
</tr>
</tbody>
</table>

Assign Code Owners to files to indicate the team members responsible for code in your project using a [CODEOWNERS](#) file. Code owners are assigned automatically as merge request approvers, can be set as required and shown when viewing files.

Learn more about Code Owners

**Code Owners Sections**

<table>
<thead>
<tr>
<th>Core</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>Bronze</td>
<td>Silver</td>
<td>Gold</td>
</tr>
</tbody>
</table>

Code Owner Sections allow each team to configure their own code owners configuration independently, allowing multiple teams to look after common parts of the codebase.

Learn more about Code Owners Sections

**Image Discussions**

Within a commit view or a merge request diff view, and with respect to a specific location of an image, you can have a resolvable discussion. Have multiple discussions specifying different areas of an image.

Image Discussions

**Merge Request Commit Discussions**
Comment on a commit within the context of a merge request itself

**Merge Request Commit Discussions**

**Inline commenting and discussion resolution**

Code or text review is faster and more effective with inline comments in merge requests. Leave comments and resolve discussions on specific lines of code. In GitLab, Merge Request Inline comments are interpreted as a discussion and can be left on any line, changed or unchanged. You can configure your project to only accept merge requests when all discussions are resolved.

Learn more about resolving discussions

**Git protocol v2 support**

Git's wire protocol defines how clones, fetches and pushes are communicated between the client and server. Git protocol v2 improves performance of fetch commands and enables future protocol improvements.

Learn more about Git protocol v2

**Code review dashboards**

Dashboards with a filterable set of code reviews (could be by project, by user, by branch, by status, or a combination of those). Dashboards includes code review status and links to get to them. This makes it easy to see what is going on with code reviews for a desired subset.

Code review with GitLab

**Contributor agreements**

Users can be required to sign one or more contributor agreements before being able to submit a change in a project.

Read more on the issue

**Robot comments**

Support for inline comments that are generated by automated third-party systems, for example robot comments can be used to represent the results of code analyzers.

GitLab merge requests store results

**Works with multiple repository types**

Supports more than one repository type, such as Git, Subversion, Perforce, CVS, Mercurial.

Learn about migrating from other SCMs