Mục lục

[Chương 1 .GITHUB 7](#_Toc42032973)

[1.Github là gì? 7](#_Toc42032974)

[1.1 Phân loại GitHub 13](#_Toc42032980)

[2 . Git là gì? 7](#_Toc42032975)

[2.1 Tại sao nên sử dụng Git? 13](#_Toc42032980)

[2.2 Quy trình làm việc trên Git là gì ? 13](#_Toc42032980)

[3 . Cách làm việc trên GitHub 7](#_Toc42032975)

[3.1 Thao tác với repositỏy ở local 13](#_Toc42032980)

[3.2 Làm việc với repository ở server GitHub 13](#_Toc42032980)

[4 . Các khái niệm cần biết trên Git/GitHub là gì? 7](#_Toc42032975)

[4.1 GitHub Repository (kho lưu trữ) là gì? 13](#_Toc42032980)

[4.2 GitHub Snapshot là gì? 13](#_Toc42032980)

[4.3 GitHub Commit là gì? 13](#_Toc42032980)

[4.4 GitHub Clone là gì? 13](#_Toc42032980)

[4.5 Push 13](#_Toc42032980)

[4.6 Fetch 13](#_Toc42032980)

[4.7. Pull 13](#_Toc42032980)

[4.8 GitHub Branch là gì? 13](#_Toc42032980)

[4.9 GitHub Fork Repository là gì? 13](#_Toc42032980)

[5 . Tính năng API của GitHub là gì? 7](#_Toc42032975)

[6 . Hướng dẫn sử dụng GitHub cơ bản 7](#_Toc42032975)

[6.1 Cách tạo một GitHub Repository 13](#_Toc42032980)

[6.2 Tạo branch trên GitHub 13](#_Toc42032980)

[6.3 Cách tạo GitHub Commit Command 13](#_Toc42032980)

[6.4 Pull Command 13](#_Toc42032980)

[6.5 Merge command 13](#_Toc42032980)

[6.6 Cách tải dự án từ GitHub về máy tính 13](#_Toc42032980)

[Chương 2 .REDMINE 11](#_Toc42032977)

[1. Redmine là gì ? 7](#_Toc42032975)

[2. Cách cài đặt Redmine nhanh nhất 7](#_Toc42032975)

[2.1 Cách cài đặt Redmine nhanh nhất cho bạn 13](#_Toc42032980)

[2.2 Ưu và nhược điểm của Redmine 13](#_Toc42032980)

[3. Hướng dẫn bạn cách sử dụng công cụ trên Redmine 11](#_Toc42032978)

[3.1 Cách tạo một GitHub Repository 13](#_Toc42032980)

[3.2 Cách tạo một GitHub Repository 13](#_Toc42032980)

[3.3 Cách tạo một GitHub Repository 13](#_Toc42032980)

[3.4 Cách tạo một GitHub Repository 13](#_Toc42032980)

[3.5 Cách tạo một GitHub Repository 13](#_Toc42032980)

[3.6 Cách tạo một GitHub Repository 13](#_Toc42032980)

[3.7 Cách tạo một GitHub Repository 13](#_Toc42032980)

**GitHub là dịch vụ server quản lý nhiều phiên bản code, giúp lưu trữ chúng và cả nội dung dự án của hai hay nhiều tài khoản của lập trình viên**. Đây cũng là nơi giúp các lập trình viên post các code, kế hoạch của mình lên để các thành viên khác trong nhóm dự án theo dõi, copy các đoạn code đó về và tiếp tục phần việc của mình. Vậy ***nền tảng xây dựng GitHub là gì?*** Cùng ***Mắt Bão*** tìm hiểu nhé!

GitHub là một server quản lý nhiều phiên bản kho lưu trữ

Với khả năng lưu trữ và độ bảo mật cao của mình, **GitHub**được xem là nguồn lưu dữ liệu mở phổ biến nhất hiện nay. ***GitHub*** là sự kết hợp giữa 2 từ:

* ***Git – hệ thống quản lý dự án và phiên bản code***
* ***Hub – nơi biến những dòng lệnh trên Git thành mạng xã hội cho lập trình viên***

**GitHub** được sử dụng chủ yếu cho dự án có nhiều người cùng hợp tác và cần giám sát toàn bộ thay đổi của dự án. Bên cạnh đó, **GitHub** còn có khả năng khôi phục code khi cần thiết. Khi sử dụng **GitHub**, ngoài các công việc chính như tạo **Branch**, tạo **Pull Request** và **Fork** một **Repository**, bạn có thể theo dõi, tương tác với người khác như một mạng xã hội thông thường.

### Phân loại GitHub

Các máy tính có thể **clone** lại mã nguồn từ một **repository** và **GitHub** chính là một **dịch vụ máy chủ repository** công cộng. Mỗi người có thể tạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có thể làm việc. GitHub có 2 loại: miễn phí và tính phí.

Với **GitHub phiên bản có phí** thường được các doanh nghiệp sử dụng để tăng khả năng quản lý team cũng như phân quyền bảo mật dự án.  
Còn lại thì phần lớn chúng ta đều sử dụng **Github với tài khoản miễn phí** để lưu trữ source code.

Tính đến tháng 9 năm 2019, ***giá của GitHub*** cụ thể như sau:

* **GitHub Individuals**: Đây là bản cá nhân có giá từ 0 – 7$. Với gói Pro 7$ bạn sẽ có thêm nhiều tính năng hơn so với miễn phí. Ví dụ như: Draft pull requests, Code owners, Pages and wikis, Repository insights… Và nhiều tính năng khác.
* **GitHub Team**: Có giá từ 9$ trở lên. Mức giá cao hơn tùy thuộc vào mô hình doanh nghiệp của bạn (enterprise). Họ sẽ liên lạc để làm báo giá cho bạn. Đúng theo thành ngữ “tiền nào của đó”, giá càng cao thì tính năng GitHub mang lại càng toàn diện.

**GitHub** cung cấp các tính năng social networking như feeds, followers, và network graph để các developer học hỏi kinh nghiệm của nhau thông qua lịch sử commit.

Nếu một comment để mô tả và giải thích một đoạn code. Thì với GitHub, commit message chính là phần mô tả hành động mà bạn thực hiện trên source code.

## ****Git là gì?****

Git là hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay

**Git là hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) được nhiều người ưa chuộng**. Git giúp máy tính lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản từ kho lưu trữ. Tất cả những code đã được update và thông tin về người sửa đổi đều được lưu lại với Git.

### ****Tại sao nên sử dụng Git?****

Git rất dễ để sử dụng, bảo đảm tính an toàn và thực hiện các hoạt động truy xuất nhanh chóng

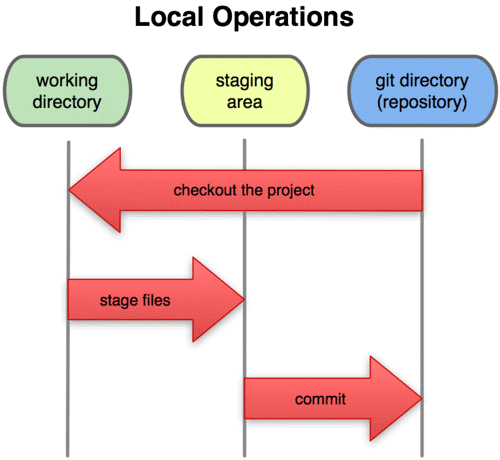
Git ra đời để phục vụ các lập trình viên chia sẻ code với nhau. Bởi thế Git mang đến rất nhiều lợi ích cho lập trình bất kể ngôn ngữ nào, như:

* **Git rất dễ sử dụng và có độ bảo mật an toàn thông tin cao**
* **Quy trình làm việc code** theo nhóm sẽ **đơn giản và dễ hiểu** hơn so với việc **kết hợp branch**
* Bạn sẽ **không còn lo ngại thay đổi làm mất mã nguồn** từ kho chứa bởi hệ thống có thể khôi phục hoàn toàn các phiên bản cũ khi cần.
* **Dễ dàng trong việc deploy sản phẩm**

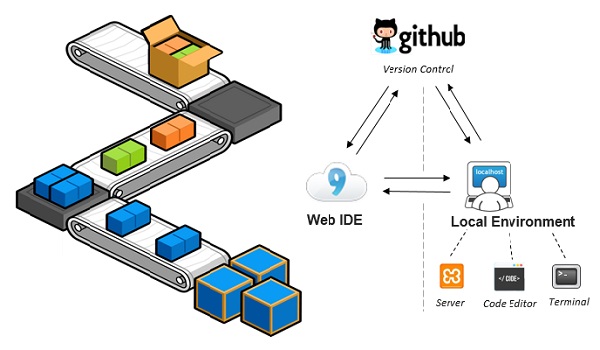
Theo đó, mọi thành viên của sự án có thể làm việc mọi lúc mọi nơi. Chỉ cần clone mã nguồn kho chứa hoặc một phiên bản clone bất kỳ, nhờ đó, việc phát triển các dự án, sản phẩm sẽ nhanh chóng, dễ dàng hơn.

### ****Quy trình làm việc trên Git là gì?****

Cùng nhắc lại **tính năng của Git là gì**? ***Git giúp bạn quản lý phiên bản mã nguồn***, bạn không thể chỉnh code trong Git. Bạn vẫn phải thực hiện các công việc của mình trên IDE của ngôn ngữ lập trình của bạn. Thông thường, ***quy trình xử lý công việc trên Git*** sẽ như sau:

Mô tả quy trình làm việc trên Git

## ****Cách làm việc trên GitHub****

Có hai nền tảng để bạn làm việc với GitHub là local workflow và server workflow

Biết được ***GitHub là gì*** thôi chưa đủ. Bạn còn cần phải nắm được **quy tắc làm việc trên GitHub**. Có hai nền tảng để bạn làm việc với **GitHub** là **local workflow** và **server workflow**.

Ở **local workflow**, bạn có thể thay đổi source code. Sau đó xác nhận các thay đổi đó tại **server workflow**. Lưu ý, bản xác nhận trên server phải chứa tính năng hoàn chỉnh hoặc có thể chạy được. Việc xác nhận các đoạn code chưa hoàn chỉnh, còn dở dang sẽ ảnh hưởng các thành viên khác khi họ cùng sử dụng kho lưu trữ.

Từ kho lưu trữ của **GitHub**, bạn có thể tạo bản build cho production site bằng cách gửi các ***source code*** thay đổi lên đó. Khi sử dụng kho lưu trữ của server, người dùng sẽ phải cung cấp mã chứng nhận, hệ thống sẽ so sánh **SSH key** ở **local** của bạn và **SSH key** trên **server** tương ứng với **account** mà đã đăng ký trước đó.

### Thao tác với repository ở local

Với 2 command thường dùng là git add và **git commit**

* **git add**: add file đã thay đổi vào stage
* **git commit commit** các file đã add vào stage lên repository ở local Ngoài ra bạn xem một số command khác

### ****Làm việc với repository ở server GitHub****

Cuối cùng khi có một bản ổn định và hoàn tất ta sẽ quyết định cập nhật nó lên **repository server** với:

* **push**: push thay đổi từ **repository local** lên **repository server**
* **fetch**: cập nhật thay đổi từ **repository server** về **repository local**
* **pull/rebase**: sao chép source code từ **server** về **local workspace** (tương đương checkout của SVN)

Để hiểu thêm về Source Code, mời bạn xem bài viết: “[***Source Code là gì***](https://wiki.matbao.net/kb/source-code-la-gi-tong-hop-day-du-nhung-kien-thuc-ve-source-code/#source-code-la-gi)“

## ****Các khái niệm cần biết trên Git/GitHub là gì?****

**Github**tồn tại khá nhiều những khái niệm quan trọng. Tuy nhiên, dưới đây là một số những khái niệm cơ bản mà bắt buộc người dùng phải nắm để có thể sử dụng Github một cách hiệu quả nhất:

* GitHub Repository (kho lưu trữ)
* GitHub Snapshot
* GitHub Commit
* GitHub Clone
* Push
* Fetch
* Pull
* GitHub Branch
* GitHub Fork Repository

Các khái niệm cơ bản trên Git và Github rất gần gũi với lập trình viên

### GitHub Repository (kho lưu trữ) là gì?

Trong **GitHub**, **repository là kho lưu trữ**, chứa toàn bộ dữ liệu thông tin, hình ảnh, video, bảng biểu… và các thay đổi trong quá trình thực hiện dự án. Có nhiều hình thức khác nhau để bạn tự tổ chức kho lưu trữ của mình, **GitHub** sẽ cho phép lập trình viên chọn một trong hai loại kho lưu trữ là **Local Repository** hoặc **Remote Repository**.

**Repository** ở **local** là kho lưu trữ để chia sẻ giữa nhiều người và bố trí trên server chuyên dụng. **Repository** ở server là kho lưu trữ trên máy tính cá nhân, dành cho một người dùng.

### GitHub Snapshot là gì?

**Snapshot** là ảnh chụp các bước commit của bạn trên kho lưu trữ nhằm lưu lại nội dung tập tin, thư mục để tham chiếu. Để hiệu quả hơn, nếu tập tin không có thay đổi, Git không lưu trữ tập tin đó lại. Mà nó chỉ tạo liên kết tới tập tin gốc đã tồn tại trước đó. Sau đó khi cần bạn hoàn toàn có thể khôi phục và sử dụng lại một snapshot. Đây cũng chính là lợi thế của Git khi nó không lưu dữ liệu mà sẽ lưu dạng snapshot. Công cụ này sẽ giúp người dùng tiết kiệm khá nhiều không gian lưu trữ.

### GitHub Commit là gì?

**Commit** là thao tác ghi lại việc thêm/thay đổi file hay thư mục vào kho lưu trữ. Theo đó, kho sẽ tạo **thư mục commit** hoặc **revision** để ghi nhận những thay đổi này. Các **commit** nối tiếp nhau theo thứ tự thời gian chỉnh sửa. Vì vậy, chỉ cần nhìn vào đây, bạn sẽ biết được lịch sử chỉnh sửa và thay đổi các file. Mỗi **commit** đều yêu cầu phải có **commit** message giúp ghi nhận sự thay đổi theo tiến trình update của lập trình viên.

### GitHub Clone là gì?

**Clone** là bản sao của một kho chứa Git có sẵn. **Clone** tạo ra bản sao hoàn chỉnh dữ liệu của kho lưu chứa trên máy chủ và tất cả lịch sử trên kho. Với nó, bạn có thể undo bất kỳ bước nào dù đã **commit**. Và dù ổ cứng máy chủ có bị hư hỏng và không sử dụng được, bạn vẫn có thể sử dụng **Clone** của bất kỳ máy khách nào để khôi phục lại dữ liệu máy chủ.

Lưu ý, ***GitHub*** có một cách khác để sao chép kho từ người khác. Đó là bạn **thực hiện fork** trên **repository** bạn cần. Điểm khác của fork là bạn có thể đóng góp thêm vào repository gốc bằng cách thực hiện pull request. Khi chủ sở hữu của repository nơi bạn fork nhận được yêu cầu sẽ xem xét chỉnh sửa của bạn, nếu thấy hay sẽ tiến hành merge nội dung chỉnh sửa của bạn vào source gốc.

### Push

**Push** là lệnh đưa nội dung mà bạn commit từ kho lưu trữ ở local lên kho lưu trữ server.

### Fetch

**Fetch** là lệnh sử dụng trên kho lưu trữ server, giúp bạn di chuyển toàn bộ dữ liệu trên kho này về máy tính để tích hợp dữ liệu vào branch.

### Pull

**Pull** là lệnh lấy dữ liệu trên kho lưu trữ server để tích hợp vào branch.

### GitHub Branch là gì?

**Branch** là một tính năng cho phép bạn tách riêng các phần của dự án. Dùng Branch để thử nghiệm các tính năng mới hoặc điều chỉnh, sửa lỗi project. Khi khởi tạo kho lưu trữ hoặc Clone, bạn sẽ được tạo lập một branch riêng. Branch riêng sẽ chứa toàn bộ mã nguồn trong kho. Như vậy mọi thành viên đều có thể phát triển nên các nội dung mới mà không sợ ảnh hưởng đến phần code hiện tại.

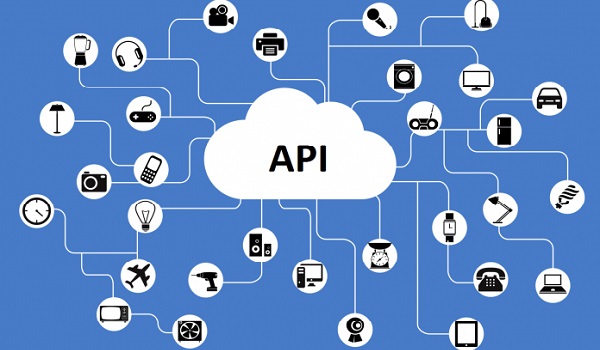
**Branch master** là nhánh “mặc định” khi bạn tạo một kho lưu trữ và là nhánh chính của ứng dụng.

### ****GitHub Fork Repository là gì?****

Fork Repository là bản sao của kho lưu trữ trên GitHub

**Kho lưu trữ Fork** là một bản copy của kho chứa **source code** trên **GitHub**. Tạo một **Fork repository** sẽ giúp bạn dễ dàng chỉnh sửa, thay đổi **source code** mà không ảnh hưởng kho lưu trữ gốc.

## ****Tính năng API của GitHub là gì?****

API là tính năng quản lý khá toàn diện của Github

Ngoài **Git**, **GitHub** còn hỗ trợ tính năng API cho lập trình viên và người dùng khác, nhằm quản lý nguồn dữ liệu dễ dàng và khoa học hơn. Sau đây là một số tính năng cơ bản của API:

* **API giúp bạn cập nhật kho lưu trữ thông qua web browser**. Cho phép chỉnh sửa các file source code thông qua http-post.
* **API hỗ trợ người dùng so sánh code của dự án qua các commit, comments**. Đồng thời bạn cũng có thể đưa ra nhận xét thông qua web browser.
* **API cho phép người dùng đăng ký một đường link cho các kho lưu trữ**.

## Hướng dẫn sử dụng GitHub cơ bản

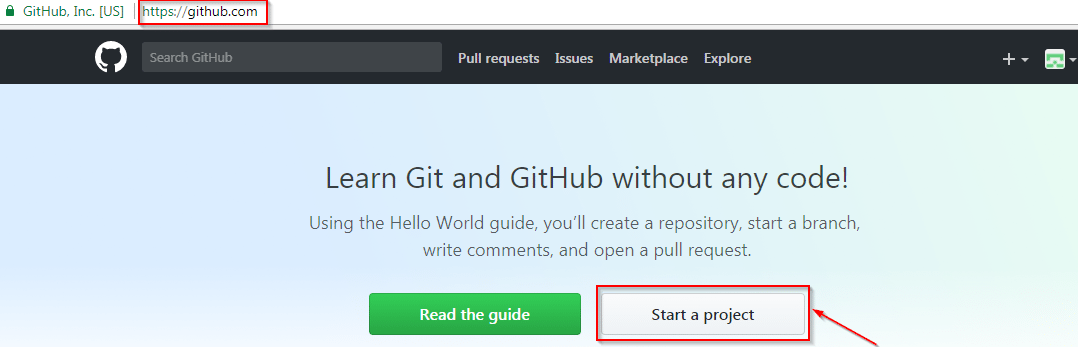
Dưới đây, Mắt Bão sẽ hướng dẫn các bạn một số thao tác sử dụng Github cơ bản sau đây:

* Cách tạo một GitHub Repository
* Tạo branch trên GitHub
* Cách tạo GitHub Commit Command
* Pull Command
* Merge command
* Cách tải dự án từ GitHub về máy tính

### ****Cách tạo một GitHub Repository****

Để tạo một repository trên GitHub bạn làm như sau:

* Đăng ký một tài khoản GitHub bằng cách click vào “***Sign up for GitHub***”.
* Sau khi đăng ký và kích hoạt thành công. Bạn bắt đầu tạo mới một project với “***Start a new project***”.

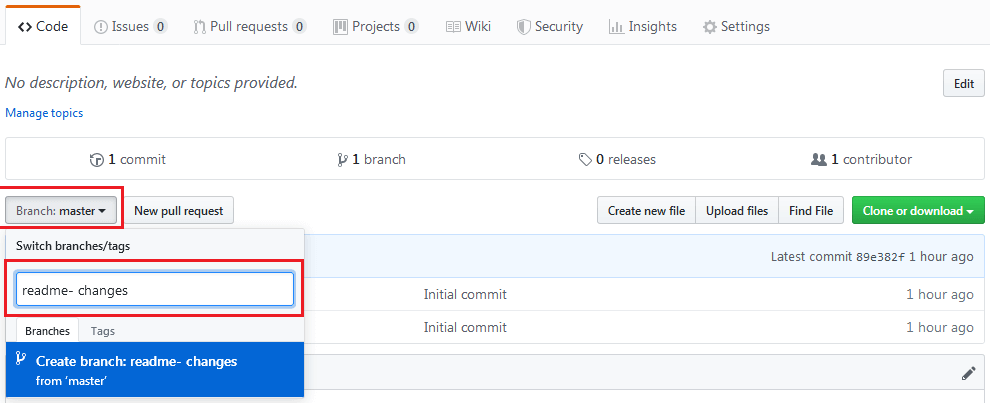
Tạo Repository trên Github

* Nhập tên **Repositoty** và nhấn nút “**Create Repository**”.
* Trong đó, bạn lưu ý 2 options sau:
  + Theo mặc định thì repository để là public. Tức là ai cũng có thể xem được repo này. Muốn quản lý nội bộ thì chọn Private.
  + Bạn có thêm một README file để giới thiệu repo kèm với một file .gitignore. Github đã có sẵn template .gitignore cho bạn, cứ chọn một template phù hợp với mã nguồn dự án là được.
* Khi đã có repository, bạn có thể clone, pull, push… source code của mình.

### ****Tạo branch trên GitHub****

Để tạo một branch trên GitHub, bạn làm như sau:

* Click vào ***dropdown branch***=> ***Branch: master***
* Khi danh sách branch xổ ra, bạn có thể tạo thêm một branch mới.

Đây là các lưu ý tạo Branch GitHub

### ****Cách tạo GitHub Commit Command****

***GitHub Commit command là gì***? Nó cho phép bạn lưu lại những thay đổi của file. Khi bạn commit, nên viết mô tả rõ ràng trong commit message. Điều này sẽ giúp cho quản lý dự án tốt hơn. Bạn có thể theo dõi, review những thay đổi source code sau này.  
Để tạo commit, bạn làm như sau:

* Chọn file muốn sửa
* Chọn “Edit” để sửa file.
* Sau khi sửa xong thì điền thông tin message và nhấn Commit.

### ****Pull Command****

***Pull Command GitHub là gì***? Lệnh PULL request là lệnh quan trọng nhất trên GitHub. Nó cho biết những thay đổi trong source code. Nó yêu cầu owner của source code xem xét và merge vào master branch.

Tính năng này rất hay cho các dự án ***mã nguồn mở***. Khi mà bất kì cũng có thể đóng góp công sức cho dự án. Tất nhiên, mọi sự thay đổi đều phải được sự đồng ý của owner. Chi tiết lệnh Pull như sau:

* Lệnh **pull request** : Là lệnh yêu cầu chủ owner dự án xem xét một thay đổi nào đó trước khi merge vào master branch.
* Lệnh **Pull**: đây là lệnh của git. Đây là lệnh update source code từ server về local. Nếu có bất kì sự xung đột code nào (conflict) thì bạn cần phải resolve nó.

### ****Merge command****

Merge command GitHub là gì? Lệnh merge này cho phép bạn hợp nhất những thay đổi vào một branch.

* Click vào “**Merge pull request**” để hợp nhất những thay đổi vào master branch.
* Click vào “**Confirm merge**”.

### ****Cách tải dự án từ GitHub về máy tính****

Có 2 cách để tải dự án từ GitHub:

* Một là bạn chọn Zip toàn bộ dự án và tải về
* Hai là bạn có thể clone dự án về bằng lệnh git. Bạn cũng click vào “Clone or Download”. Sau đó copy đường dẫn và gõ trong cửa sổ terminal trên máy tính như sau

git clone git@github.com:duong-dan-ma-ban-da-copy.git

**REDMINE**

**1. Redmine là gì bạn biết chưa?**

Redmine là gì? Bạn đã bao giờ nghe đến ứng dụng này chưa? Redmine là một công cụ được tạo ra nhằm mục đích theo dõi các vấn đề xảy ra của một dự án và quản lý dự án đó. Redmine hoạt động trên web miễn phí và mã nguồn mở để người dùng dễ dàng sử dụng. Đây là phần mềm hỗ trợ cho người sử dụng quản lý nhiều dự án cùng một lúc hoặc các tiểu dự án có liên quan đến nhau và liên quan đến dự án chính, Redmine có tính năng cho mỗi một dự án khác nhau và diễn đàn khác nhau. Redmine giúp theo dõi thời gian, kiểm soát các truy cập với vai trò linh hoạt của nó trong từng dự án.

**Redmine là gì?**

Redmine bao gồm biểu đồ thanh và biểu đồ lịch để hỗ trợ một cách thức quan nhất trong khoảng thời gian thực hiện dự án, Redmine là sự tích hợp từ nhiều hệ thống phiên bản khác nhau và trình duyệt kho lưu trữ, Redmine coa vai trò trong việc quản lý các vấn đề như nhiệm vụ, lỗi và yêu cầu hỗ trợ. Thông qua việc sử dụng phần mềm quản lý dự án sẽ dễ dàng hơn cho bạn trong việc báo cáo kết quả hàng ngày với các bộ phần có chức trách cao hơn trong doanh nghiệp.

**\* Các tính năng của Redmine bao gồm như sau:**

+ Cho phép việc theo dõi nhiều dự án cùng một lúc

+ Hỗ trợ cho việc kiểm soát về truy cập với vai trò linh hoạt

+ Theo dõi các vấn đề của dự án

+ Tạo biểu đồ

+ Quản lý các dữ liệu như: Tin tức, tài liệu, tập tin có liên quan đến dự án

+ Cho phép truy cập dữ liệu web và gửi thông báo về email của bạn

+ Hỗ trợ việc quản lý cho các dự án, và diễn đàn cho mỗi dự án đó

+ Cho phép người dùng theo dõi được thời gian một cách đơn giản

+ Bạn có thể tùy chỉnh cho các vấn đề của dự án và mực thời gian của dự án cũng như người dùng

+ Hỗ trợ một loạt các tích hợp của quản lý cấu hình phần mềm như: hệ thống kiểm soát sửa đổi, hệ thống đồng thời, hệ thống kiểm soát để theo dõi những thay đổi trong mã nguồn, công cụ kiểm soát sửa đổi phân tán, hệ thống sửa đỗi máy chủ phân tán và máy khách,..

+ Hỗ trợ việc xác thực nhiều nhà cung cấp trung lập, và tiêu chuẩn công nghệ ứng dụng để truy cập và duy trì các dịch vụ về thông tin, thư mục sử dụng trên giao thức internet.

+ Hỗ trợ tính năng giúp người dùng tự đăng ký cài đặt

+ Hỗ trợ sử dụng với 49 ngôn ngữ trên thế giới. Trong đó có cả Redmine tiếng Việt.

+ Cho phép truy cập nhiều cơ sở dữ liệu khác nhau

+ Redmine cho phép việc bổ trợ trong các dự án

+ Cung cấp một giao diện về lập trình ứng dụng để hỗ trợ truyền thông và giúp chuyển trạng thái của đại diện.

Quản lý dự án Redmine sẽ giúp bạn dễ dàng hơn trong việc quản lý các vấn đề của dự án trên web và được viết trên Ruby Rails. Với việc sử dụng miễn phí phần mềm này đã giúp bạn quản lý dự án hiệu quả và khoa học hơn.