LightHouseIT

第5章1節
バージョン管理

- Git基本原理　Gitの基本理論
- Git命令行工具　Gitコマンドラインツール
- Markdown　マークダウン

© LightHouseIT. All rights reserved.

LightHouseIT

© LightHouseIT. All rights reserved.

版本控制工具

- 什么是版本控制工具
- 版本控制工具提供完备的版本管理功能，用于存储、追踪目录（文件夹）和文件的修改历史，是软件开发者的必备工具，是软件公司的基础设施。
- 版本控制最主要的功能就是追踪文件的变更。它将什么时候、什么人更改了文件的什么内容等信息忠实地了已录下来。每一次文件的改变，文件的版本号都将增加。除了记录版本变更外，版本控制的另一个重要功能是并行开发。软件开发往往是多人协同作业，版本控制可以有效地解决版本的同步以及不同开发者之间的开发通信问题，提高协同开发的效率。并行开发中最常见的不同版本软件的错误(Bug)修正问题也可以通过版本控制中分支与合并的方法有效地解决。

© LightHouseIT. All rights reserved.

版本控制工具的作用

- 1 协同开发：团队协作共同完成同一个项目
- 2 版本管理：以不断提升项目版本的方式逐步完成项目。
- 3 数据备份：开发中以版本控制的形式保存每一个历史版本。
- 4 权限控制：对团队开发人员进行不同的权限分配。
- 5 分支管理：允许开发团队在工作过程中多条生产线同时推进任务，进一步提高效率。

© LightHouseIT. All rights reserved.

Git基本概念

- Git 是一个开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。
- Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。
- Git 与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持。

https://git-scm.com

© LightHouseIT. All rights reserved.

Git工作流程（远程与本地的关系）

Git 工作流程

1 克隆 Git 资源作为工作目录。

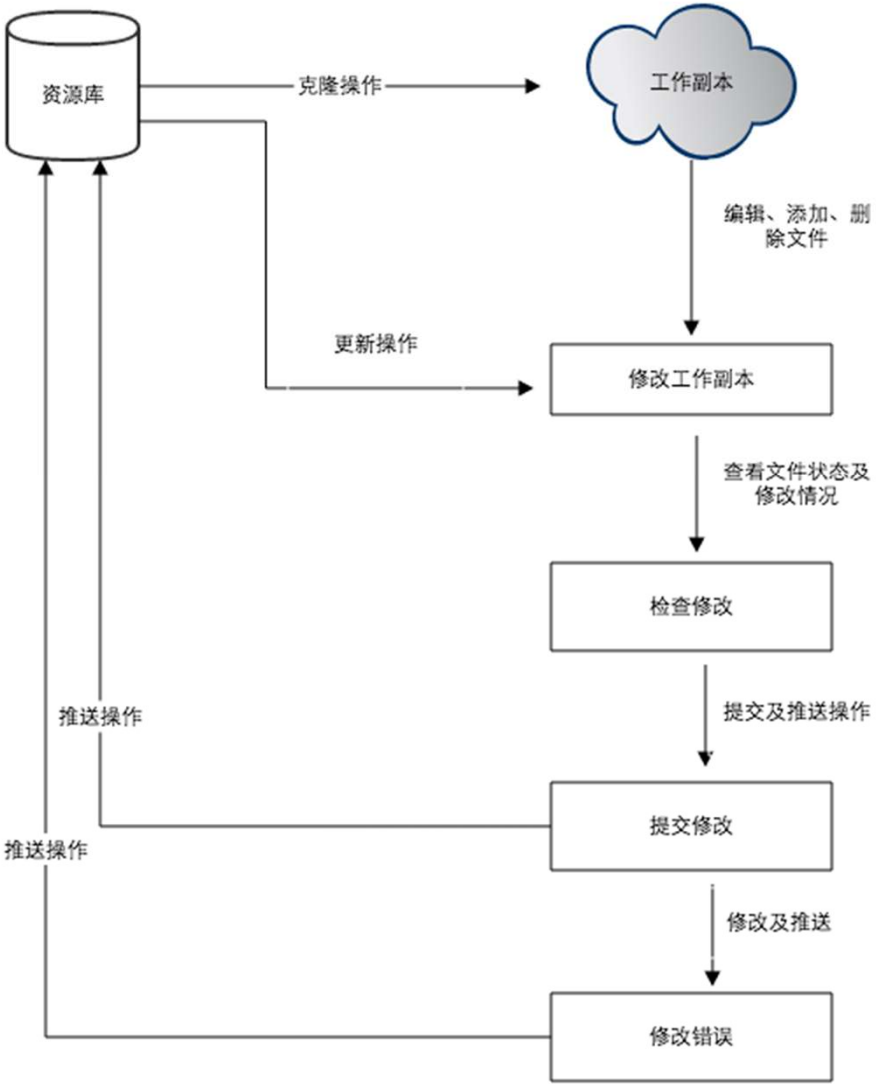
2 在克隆的资源上添加或修改文件。

3 如果其他人修改了，你可以更新资源。

4 在提交前查看修改。

5 提交修改。

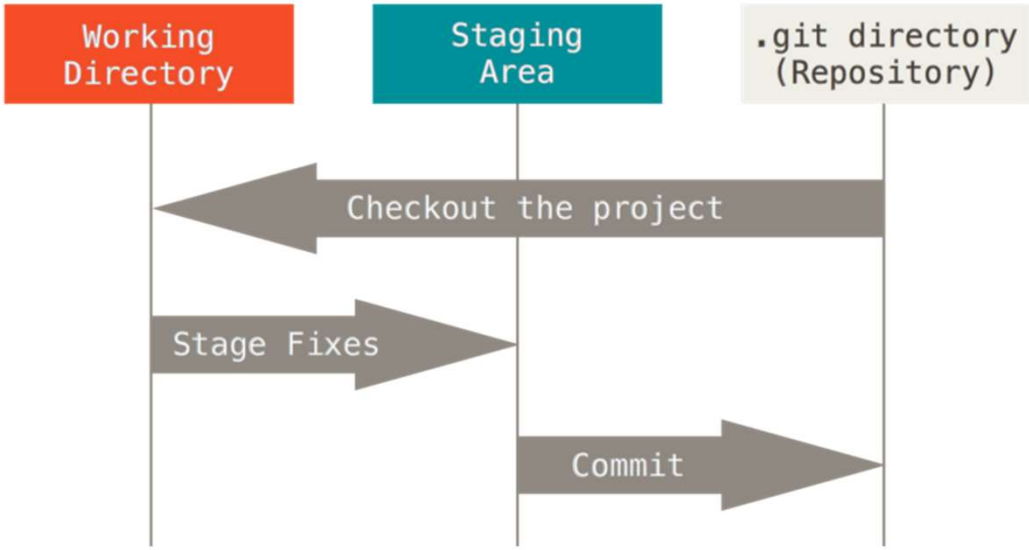
6 在修改完成后，如果发现错误，可以撤回提交并再次修改并提交。



© LightHouseIT. All rights reserved.

Git工作流程（本地仓库）

Git有三种状态，你的文件可能处于其中之一：已提交(committed)、已修改(modified)和已暂存(staged)。 已提交表示数据已经安全的保存在本地数据库中。 已修改表示修改了文件，但还没保存到数据库中。 已暂存表示对一个已修改文件的当前版本做了标记，使之包含在下次提交的快照中。



由此引入 Git项目的三个工作区域的概念：Git仓库、工作目录以及暂存区域。

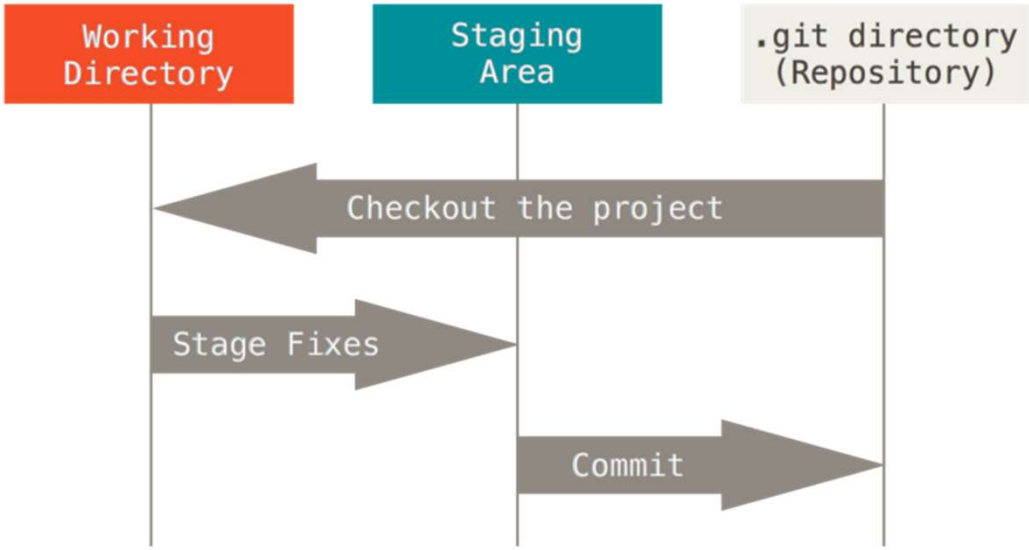
© LightHouseIT. All rights reserved.

Git工作流程（本地仓库）

Git仓库目录是 Git用来保存项目的元数据和对象数据库的地方。 这是 Git中最重要的部分，从其它计算机克隆仓库时，拷贝的就是这里的数据。

工作目录是对项目的某个版本独立提取出来的内容。 这些从Git仓库的压缩数据库中提取出来的文件，放在磁盘上供你使用或修改。

暂存区域是一个文件，保存了下次将提交的文件列表信息，一般在Git仓库目录中。 有时候也被称作‘索引’，不过一般说法还是叫暂存区域。

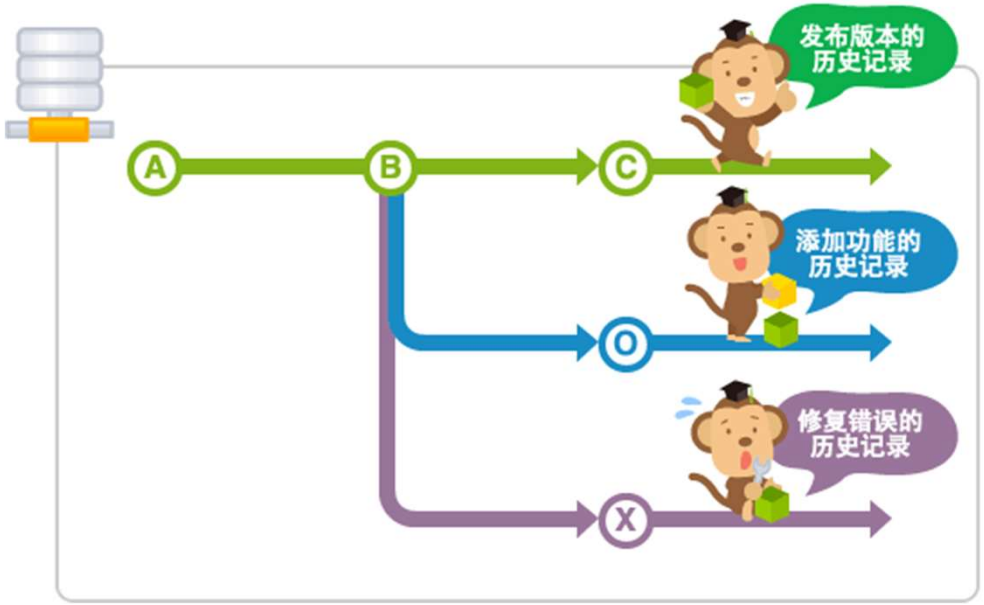


© LightHouseIT. All rights reserved.

Git分支（branch）

在开发软件时，可能有多人同时为同一个软件开发功能或修复BUG，可能存在多个Release版本，并且需要对各个版本进行维护。

所幸，Git的分支功能可以支持同时进行多个功能的开发和版本管理。



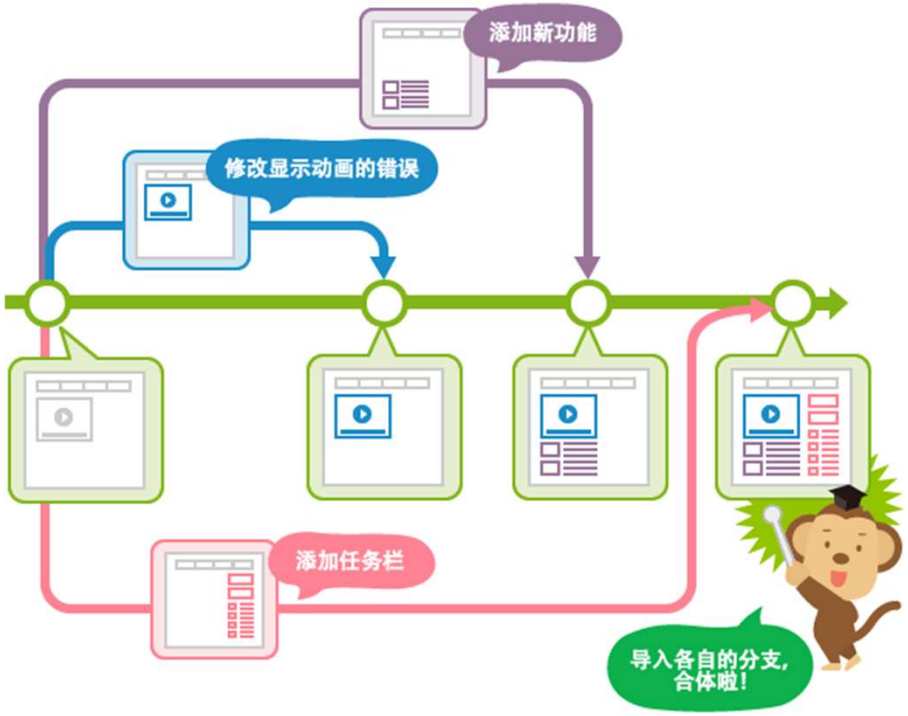
分支是为了将修改记录的整体流程分叉保存。分叉后的分支不受其他分支的影响，所以在同一个数据库里可以同时进行多个修改。

© LightHouseIT. All rights reserved.

Git分支

分叉的分支可以合并。

为了不受其他开发人员的影响，可以在主分支（master）上建立自己专用的分支。完成工作后，将自己分支上的修改合并到主分支。因为每一次提交的历史记录都会被保存，所以当发生问题时，定位和修改造成问题的提交就容易多了。



© LightHouseIT. All rights reserved.

© LightHouseIT. All rights reserved.

LightHouseIT

© LightHouseIT. All rights reserved.

Git安装

windows：https://gitforwindows.org

mac：（在terminal中安装）

1.安装包管理工具：/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

2.安装git： brew install git

确认安装：

在cmd或者terminal中输入：git --version

© LightHouseIT. All rights reserved.

仓库克隆

克隆仓库的命令格式为：

git clone [url]

比如，要克隆 Git 代码仓库 Grit，可以用下面的命令：

$ git clone git://github.com/schacon/grit.git

执行该命令后，会在当前目录下创建一个名为grit的目录，其中包含一个 .git 的目录，用于保存下载下来的所有版本记录。

如果要自己定义要新建的项目目录名称，可以在上面的命令末尾指定新的名字：

$ git clone git://github.com/schacon/grit.git mygrit

© LightHouseIT. All rights reserved.

仓库克隆

克隆仓库的命令格式为：

git clone [url]

比如，要克隆 Git 代码仓库 Grit，可以用下面的命令：

$ git clone git://github.com/schacon/grit.git

执行该命令后，会在当前目录下创建一个名为grit的目录，其中包含一个 .git 的目录，用于保存下载下来的所有版本记录。

如果要自己定义要新建的项目目录名称，可以在上面的命令末尾指定新的名字：

$ git clone git://github.com/schacon/grit.git mygrit

使用git pull可以将远程仓库的最新修改下拉到本地仓库

© LightHouseIT. All rights reserved.

Git分支管理

列出分支基本命令：

$ git branch

如果我们要手动创建一个分支。执行 git branch (branchname) 即可。

$ git branch testing

用 git checkout (branch) 切换到我们要修改的分支。

$ git checkout testing

也可以使用 git checkout -b (branchname) 命令来创建新分支并立即切换到该分支下，从而在该分支中操作。

$ git checkout -b newtest

删除分支命令：

git branch -d (branchname)

© LightHouseIT. All rights reserved.

Git执行变更

当你在当前的仓库进行修改之后

- **使用git status命令查看当前工作区状态:**

$ git status

- **Git添加操作(git add命令)将文件添加到暂存区域。**

$ git add [所需要修改的文件名或者.表示添加全部修改]

在执行上命令后，已将文件添加到存储区域，git status命令将显示临时区域中存在的文件。

- **要提交更改，可使用了git commit命令**

后跟-m选项。 如果忽略了-m选项。 Git将打开一个文本编辑器，我们可以在其中编写多行的提交备注消息。在执行 git commit命令之前，一定要先执行 git add 命令。

$ git commit -m commit信息

- **将当前分支推送到origin主机的对应分支**

$ git push

© LightHouseIT. All rights reserved.

Git分支合并

在执行完git push之后，你的修改已经成功被反映到远程仓库了，但是并没有被反映到master分支上。所以此时需要执行分支合并操作。

- 如果你是这个项目的拥有者，你可以使用git merge在本地合并之后再将合并好的master分支推送到远程仓库。
- 如果你只是这个项目的贡献者，你没有权利独自修改master，可以使用github的pull request来向你的项目管理员申请一次合并。

© LightHouseIT. All rights reserved.

Git分支合并

本地合并

首先将分支切换到你的目标分支，然后再合并需要合并进来的分支

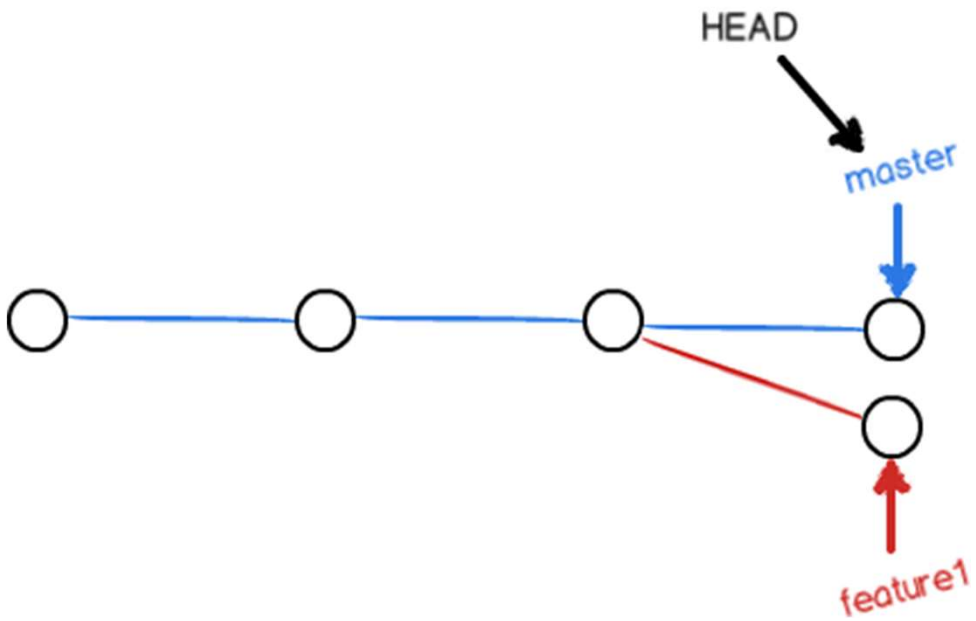
$ git checkout master

$ git merge test

通过这两个命令我们做到来将test分支合并到master上

© LightHouseIT. All rights reserved.

Git解决冲突



在上图的情况中，feature1在想要合并到master时发现，在feature1分支发生修改的时候，可能有别的同事也对master发生了修改。

这时候再想要合并回master分支的时候依据代码的修改情况（比如两个人都修改了同一行代码）可能会发生冲突。

© LightHouseIT. All rights reserved.

Git解决冲突

$ git merge feature1

Auto-merging readme.txt

CONFLICT (content): Merge conflict in readme.txt

Automatic merge failed; fix conflicts and then commit the result.

Git告诉我们，readme.txt文件存在冲突，必须手动解决冲突后再提交。git status也可以告诉我们冲突的文件。

Git用<<<<<<<，=======，>>>>>>>在冲突的文件中标记出不同分支的内容。

© LightHouseIT. All rights reserved.

Git解决冲突

Git 会在有冲突的文件中加入标准的冲突解决标记，这样你可以打开这些包含冲突的文件然后手动解决冲突。 出现冲突的文件会包含一些特殊区段，看起来像下面这个样子：

```
<<<<<<< HEAD
Creating a new branch is quick & simple.
=======
Creating a new branch is quick AND simple.
>>>>>>> feature1
```

这是当前分支的最新commit里发生冲突的代码

这是需要合并的分支里发生冲突的代码

© LightHouseIT. All rights reserved.

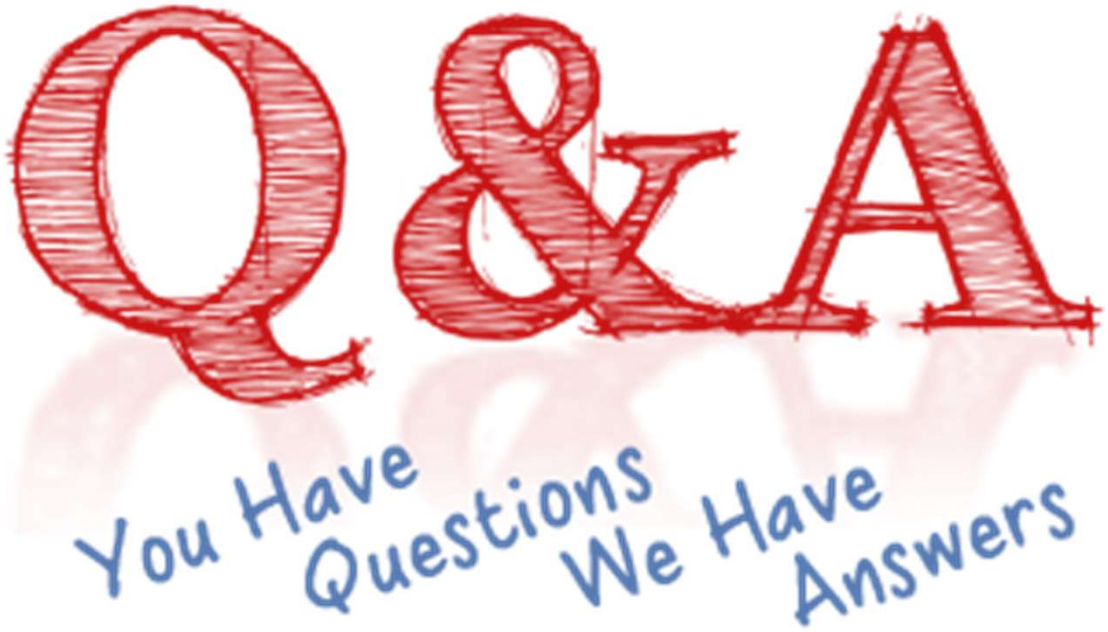
Git解决冲突

我们需要将<<<<<<<，=======，>>>>>>>里的内容手动修改后保存。

Creating a new branch is quick and simple.

上述的冲突解决方案仅保留了其中一个分支的修改，并且 <<<<<<< ，======= , 和 >>>>>>> 这些行被完全删除了。 在你解决了所有文件里的冲突之后，对每个文件使用 git add 命令来将其标记为冲突已解决。 一旦暂存这些原本有冲突的文件，Git 就会将它们标记为冲突已解决。

© LightHouseIT. All rights reserved.



© LightHouseIT. All rights reserved.

LightHouseIT

© LightHouseIT. All rights reserved.

Markdown

Markdown 是一种轻量级标记语言，它允许人们使用易读易写的纯文本格式编写文档。

Markdown 语言在 2004 由约翰·格鲁伯（英语：John Gruber）创建。

Markdown 编写的文档可以导出 HTML 、Word、图像、PDF、Epub 等多种格式的文档。

Markdown 编写的文档后缀为 .md, .markdown。

具体写法参考：

https://github.com/zhongtaoaki/lighthouse-april-java/blob/main/markdown.md

（本班专用github链接）

© LightHouseIT. All rights reserved.

© LightHouseIT. All rights reserved.

© LightHouseIT. All rights reserved.