发布到 NPM 的软件包和 WordPress 核心更新

目录

同步 Gutenberg 插件
WordPress 发布

独立 bug 修复包发布
开发版本发布

Gutenberg 仓库遵循 WordPress SVN 仓库 的分支策略，适用于每个主要的 WordPress
版本。除此之外，它还包含另外两个控制 npm 发布流程的特殊分支：

wp/latest 分支包含与使用 latest 分发标签发布到 npm
的软件包相同的版本。此分支的目标是与最新的 Gutenberg
插件版本保持同步，唯一的例外是计划外的 错误修复发布 。
wp/next 分支包含与使用 next 分发标签发布到 npm 的软件包相同的版本。它始终与 trunk
分支保持同步。项目应仅将这些软件包用于开发或测试目的。
针对特定 WordPress 主要版本（包括其后续次要版本）的 Gutenberg 分支 wp/X.Y（例如 wp/6.2
）会在计划包含在下一个主要 WordPress 版本中的最后一次发布后不久，基于当前的 Gutenberg
插件发布分支 release/X.Y（例如 release/15.1）创建。

发布类型及其时间表：

同步 Gutenberg 插件 （ latest 分发标签） ——基于新创建的 release/X.Y（例如 release/12.8
）分支，每两周自动发布一次，使用 Gutenberg 插件的 RC1 版本。
WordPress 发布 （ wp-X.Y 分发标签，例如 wp-6.2）——根据需求从 wp/X.Y（例如 wp/6.2
）分支触发发布。一旦我们到达 WordPress 主要发布周期中仅从 Gutenberg 仓库精选提交到
WordPress 核心的阶段（在 Beta 1 之前不久），我们使用 wp/X.Y 分支（从 release/X.Y
分支创建，例如 release/15.1）进行 npm 发布，并使用 wp-X.Y
分发标签。也可以使用旧分支将错误或安全修复反向移植到相应旧版本的 WordPress 核心。
开发发布 （ next 分发标签） ——也可以在需要测试即将到来的更改时随时执行开发发布。

还可以选择随时执行 独立错误修复软件包发布 。这应仅保留用于必须在常规周期之外发布到 npm 的关键错误修复或安全发布。

同步 Gutenberg 插件
针对每个 Gutenberg 插件版本，我们也会在 npm 上发布更新后的 WordPress 软件包版本。这是通过处理

Gutenberg 插件发布的 发布工具 自动完成的。成功的 RC1 发布会触发 npm 发布任务，这需要 Gutenberg

核心团队成员批准。找到新版本的 "构建 Gutenberg 插件 Zip"工作流 ，并 批准 它。

我们特意在 Gutenberg RC1 发布时，用 Gutenberg 发布分支 release/X.Y（例如 release/12.7
）的内容更新 Gutenberg 仓库内的 wp/latest 分支。这样做是为了确保 wp/latest 分支尽可能接近
Gutenberg 插件的最新版本。这实际上也消除了在向 trunk 回移植提交时，因发布期间应用到 package.json 和 CHANGELOG.md 文件的更新而产生冲突的可能性。过去，当我们在常规 Gutenberg 发布一周后进行 npm 发布时，在这方面遇到了很多问题。在将新软件包版本发布到 npm 时，我们至少选择 minor
版本升级，以便为未来的错误修复或安全发布留出空间。

在幕后，所有步骤都通过 ./bin/plugin/cli.js npm-latest
命令自动完成。作为记录，手动过程将非常接近以下步骤：

1. 确保 WordPress 的 trunk 分支已开放用于功能增强。
2. 使用 git fetch 获取最新发布的 Gutenberg 发布分支。
3. 检出 wp/latest 分支。
4. 从当前分支移除所有文件： git rm -r .。
5. 从发布分支检出所有文件： git checkout release/x.x -- . 。
6. 将所有更改提交到 wp/latest 分支： git commit -m "Merge changes published in the Gutenberg plugin vX.X release" 并推送到仓库。
7. 使用计算出的新发布版本更新包的 CHANGELOG.md 文件，并提交到 wp/latest
分支。假设包版本使用 major.minor.patch 格式，确保至少应用 minor
版本号提升。例如，如果待发布包的 CHANGELOG 显示下一个未发布版本是 5.6.1，则在 minor
版本提升时选择 5.7.0。这很重要，因为补丁版本号应保留，以备 WordPress
小版本需要错误修复时使用（见下文）。
8. 通过控制台登录 npm： npm login。注意，您应已启用双因素认证（ 2FA）。
9. 在 wp/latest 分支上，使用 npm ci 安装 npm 依赖项。
10. 运行脚本 npx lerna publish --no-private。
当被问及每个包的版本号时，选择更新后的 CHANGELOG 文件中的值。
您将被多次要求输入一次性密码（ OTP）。这是您使用的 2FA
验证器应用生成的代码。根据要发布的包数量，您可能需要输入多次 OTP
，因为它们往往在所有包发布前就会过期。
如果发布过程未完成（可能因为超时或输入了错误的 OTP），您可以通过 npx lerna publish from-package 恢复发布。
11. 最后，在 npm 包发布完成后，将 lerna 创建的提交（ "Publish" 和 CHANGELOG 更新）拣选到 Gutenberg 的 trunk 分支中。

WordPress 发布

当需要将错误修复或安全修复反向移植到 WordPress
核心时，需要遵循以下工作流程。这可能在几种使用场景下发生：

在 WordPress 发布周期的 beta 和 RC 阶段，此时该版本的 wp/X.Y 分支（例如 wp/5.7
）已经存在。
对于 WordPress 次要版本和安全版本发布（例如 5.1.1）。

1. 检出相关的 WordPress 主版本分支（如果次要版本是 5.2.1，则检出 wp/5.2）。
2. 从该分支创建一个功能分支，并将所需错误修复的合并提交拣选到该分支上。拣选过程可以使用 npm run other:cherry-pick 脚本自动化。
3. 从此分支创建一个拉取请求，目标分支是上面使用的 WordPress 主版本分支。
4. 使用 "Rebase and Merge" 按钮合并拉取请求，以保留提交历史。

现在， wp/X.Y 分支已准备好发布 npm 包。要启动此过程，请转到 Gutenberg 的 GitHub 仓库的 Actions

选项卡，找到 "Publish npm packages" action。注意写着 "This workflow has a workflow_dispatch
event trigger." 的蓝色横幅，并展开其右侧的 "Run workflow" 下拉菜单。

npm 发布的运行工作流下拉菜单

要为 WordPress 主版本发布将包发布到 npm，请选择 trunk
作为运行工作流的分支（这意味着用于运行工作流的脚本来自 trunk 分支，不过只要在下面选择了正确的
"Release type"，包本身将从发布分支发布），然后从 "Release type" 下拉菜单中选择 wp，并在
"WordPress major release" 输入字段中输入 X.Y（例如 5.2）。最后，按下绿色的 "Run workflow"

按钮。这会触发 npm 发布任务，并且需要由一位 Gutenberg 核心团队成员批准。找到当前发布的 "Publish

npm packages" action，并让其获得 批准 。

作为记录，手动过程如下所示：

1. 检出之前使用的 WordPress 分支（例如 wp/5.2）。
2. git pull。
3. 运行 npx lerna publish patch --no-private --dist-tag wp-5.2 命令（更多信息请参见 [
包发布流程 ]），但当被问及为每个包选择版本号时（假设包版本使用 major.minor.patch
格式），请确保仅提升 patch 版本号。例如，如果此 WordPress 分支上次发布的包版本是 5.6.0
，则选择 5.6.1 作为版本。

注意： 对于 WordPress 5.0 和 WordPress 5.1
，使用了不同的发布流程。这意味着当选择针对这两个版本的 npm 包版本时，你将无法使用下一个 patch
版本号，因为它可能已被使用。对于这些版本，你应该使用 “元数据 ”修饰符。例如，如果此 WordPress
分支最后发布的包版本是 5.6.1，则选择 5.6.1+patch.1 作为版本。

3. 可选地，使用新发布的版本更新已发布包的 CHANGELOG.md 文件，并提交到相应的分支（例如
wp/5.2）。
4. 如果有的话，将 CHANGELOG 更新提交拣选到 Gutenberg 的 trunk 分支中。

现在， npm 包应该已准备就绪，可以创建补丁并提交到相应的 WordPress SVN 分支。

# 独立 bug 修复包发布

当软件包需要在常规发布周期之外向 *npm* 发布错误修复或安全更新时，需要遵循以下工作流程。

注意：`trunk` 和 `wp/latest` 分支均受限制，仅 Gutenberg 核心团队可以 推送 。

识别需要从仓库 `trunk` 分支移植到 `wp/latest` 分支的拉取请求的提交哈希值。

现在需要准备 `wp/latest` 分支以发布软件包到 *npm* 。

打开终端并执行以下步骤：

1. `git checkout trunk`
2. `git pull`
3. `git checkout wp/latest`
4. `git pull`

在移植提交之前，检查 `wp/latest` 分支是否没有任何待发布的软件包：

1. `git checkout wp/latest`
2. `npx lerna updated`

现在将提交从 `trunk` *cherry-pick* 到 `wp/latest`，如果提交是拉取请求的合并提交，请使用 `-m 1` `commithash`：

1. `git cherry-pick -m 1 cb150a2`
2. `git push`

在等待 GitHub Actions 构建 `wp/latest` 分支通过 的同时，识别并开始更新 `CHANGELOG.md` 文件：

1. `git checkout wp/latest`
2. `npx lerna updated` 示例：

```
npx lerna updated
@wordpress/e2e-tests
@wordpress/jest-preset-default
@wordpress/scripts
lerna success found 3 packages ready to publish
```

检查当前 `CHANGELOG.md` 文件中列出的版本，浏览软件包的提交历史，例如 @wordpress/scripts，并留意 "*chore(release): publish*" 和 "*Update changelogs*" 提交以确定最近的版本更新，然后查看自最近一次发布以来的提交，这有助于发现自上次发布以来发生了哪些更改。

注意：您可能会发现每个软件包的当前版本不是最新的，如果是这样，更新先前发布的版本将不胜感激。

现在，`wp/latest` 分支已准备好发布 npm 软件包。要启动此过程，请转到 Gutenberg 的 GitHub 仓库的 Actions 选项卡，找到 "Publish npm packages" action。注意写着 "This workflow has a `workflow_dispatch` event trigger." 的蓝色横幅，并展开其右侧的 "Run workflow" 下拉菜单。

要将包含错误修复的包发布到 npm，请从 “发布类型 ”下拉菜单中选择 `bugfix`，并留空 “WordPress 主版本 ” 输入字段。最后，按下绿色的 “运行工作流 ”按钮。这会触发 npm 发布任务，需要由 Gutenberg 核心团队成员批准。找到当前发布的 “发布 npm 包 ”操作，并 批准 它。

在后台，其余流程通过 `./bin/plugin/cli.js npm-bugfix` 命令自动完成。作为记录，手动流程将非常接近以下步骤：

1. 检出 `wp/latest` 分支。
2. 使用计算出的新发布版本更新包的 `CHANGELOG.md` 文件，并提交到 `wp/latest` 分支。
3. 通过控制台登录 npm：`npm login`。注意，您应启用双因素认证（2FA）。
4. 在 `wp/latest` 分支上，使用 `npm ci` 安装 npm 依赖项。
5. 运行脚本 `npx lerna publish --no-private`。
   - 当询问每个包的版本号时，选择更新后的 CHANGELOG 文件中的值。
   - 您将被多次要求输入一次性密码（OTP）。这是您使用的 2FA 验证器应用中的代码。根据要发布的包数量，您可能需要输入多个 OTP，因为它们往往在所有包发布前就会过期。
   - 如果发布过程未完成（可能因为超时或输入了错误的 OTP），您可以通过 `npx lerna publish from-package` 恢复它。
6. 最后，既然 npm 包已发布，将 lerna 创建的提交（“发布 ”和 CHANGELOG 更新）拣选到 Gutenberg 的 `trunk` 分支中。

# 开发版本发布

如 同步 Gutenberg 插件 章节所述，软件包发布每两周从 `wp/latest` 分支进行一次。开发者也可以随时使用开发版本来测试 `trunk` 分支中即将到来的变更。我们利用了 npm 软件包分发标签功能，使得根据 npm 指南使用代码库的未来版本成为可能：

> 默认情况下，npm 使用 `latest` 标签来标识软件包的当前版本，`npm install <pkg>`（不带任何 `@<version>` 或 `@<tag>` 说明符）会安装 `latest` 标签。通常，项目仅将 `latest` 标签用于稳定发布版本，而将其他标签用于预发布版等不稳定版本。

在我们的场景中，我们使用 `next` 分发标签来标识开发代码。想要安装此版本软件包的开发者需要输入：

```
npm install @wordpress/components@next
```

要启动 npm 软件包开发版本的发布流程，请前往 Gutenberg 的 GitHub 仓库的 Actions 选项卡，找到 "发布 npm 软件包" 操作。注意蓝色横幅上写着 "此工作流有一个 `workflow_dispatch` 事件触发器。" ，并展开其右侧的 "运行工作流" 下拉菜单。

要将开发包发布到 npm，请从 "发布类型" 下拉菜单中选择 `development`，并留空 "WordPress 主版本" 输入字段。最后，按下绿色的 "运行工作流" 按钮。这会触发 npm 发布任务，并且需要得到 Gutenberg 核心团队成员的批准。找到当前发布的 "发布 npm 软件包" 操作，并批准它。

在后台，发布流程通过 `./bin/plugin/cli.js npm-next` 命令完全自动化。它确保 `wp/next` 分支与为 Gutenberg 插件创建的最新发布分支 (`release/X.Y`) 保持同步。为了避免软件包版本冲突，我们总是包含最新提交的 `sha`，例如 `@wordpress/block-editor@5.2.10-next.645224df70.0`。

---

版本号 #2
由 壳壳虫 创建于 2026-03-27 01:08:43 UTC
由 壳壳虫 更新于 2026-03-27 01:08:43 UTC