# WordPress

# 主机托管手册

WordPress 主机托管手册，面向主机商与运维。


- WordPress 兼容性
- Contributor Day
- 文档
- 参与进来
- WordPress 托管团队手册
- 学习主机托管
- 性能
- 可靠性
- 服务器环境
- 主机的可持续性
- 托管项目
- 团队代表
- Monthly Hosting Team Updates
- 测试
- 升级 WordPress

# WordPress 兼容性

WordPress、PHP 与 MySQL / MariaDB 之间的关系非常紧密，这些技术版本的匹配对于正常运行至关重要。以下矩阵主要面向托管公司及系统管理员。

[tip]如果您是开发者，您可能也会对核心团队发布的 PHP 兼容性与 WordPress 版本文档感兴趣。[/tip]

# WordPress、PHP、MySQL / MariaDB 版本

此表显示了 WordPress 发布时可用的（且受安全支持的）版本。这并不意味着 WordPress 对这些版本提供了 100% 的完全支持（但通常确实支持）。

| WordPress | PHP | MySQL | MariaDB | 发布日期 |
|---|---|---|---|---|
| WordPress 6.8 | 8.1 - 8.4 | 8.0 / 8.4 / 9.1 | 10.5 - 10.6 / 10.11 / 11.4 - 11.5 | 2025-04-15 |
| WordPress 6.7 | 8.1 - 8.4 | 8.0 / 8.4 / 9.1 | 10.5 - 10.6 / 10.11 / 11.4 - 11.5 | 2024-11-12 |
| WordPress 6.6 | 8.1 - 8.3 | 8.0 / 8.2 - 8.4 | 10.5 - 10.6 / 10.11 / 11.1 - 11.2 / 11.4 | 2024-07-16 |
| WordPress 6.5 | 8.1 - 8.3 | 8.0 - 8.3 | 10.4 - 10.6 / 10.11 / 11.0 - 11.3 | 2024-04-02 |
| WordPress 6.4 | 8.0 - 8.2 | 8.0 - 8.2 | 10.4 - 10.6 / 10.10 - 11.1 | 2023-11-07 |
| WordPress 6.3 | 8.0 - 8.2 | 5.7 - 8.1 | 10.4 - 10.6 / 10.9 - 11.0 | 2023-08-08 |
| WordPress 6.2 | 8.0 - 8.2 | 5.7 - 8.0 | 10.3 - 10.11 | 2023-03-28 |
| WordPress 6.1 | 7.4 - 8.1 | 5.7 - 8.0 | 10.3 - 10.6 | 2022-11-01 |
| WordPress 6.0 | 7.4 - 8.1 | 5.7 - 8.0 | 10.3 - 10.6 | 2022-05-24 |
| WordPress 5.9 | 7.4 - 8.1 | 5.7 - 8.0 | 10.2 - 10.6 | 2022-01-25 |
| WordPress 5.8 | 7.3 - 8.0 | 5.7 - 8.0 | 10.2 - 10.6 | 2021-07-20 |
| WordPress 5.7 | 7.3 - 8.0 | 5.7 - 8.0 | 10.2 - 10.5 | 2021-03-09 |

| WordPress | PHP | MySQL | MariaDB | □□□□ |
|---|---|---|---|---|
| WordPress 5.6 | 7.3 - 8.0 | 5.7 - 8.0 | 10.2 - 10.5 | 2020-12-08 |
| WordPress 5.5 | 7.2 - 7.4 | 5.7 - 8.0 | 10.2 - 10.5 | 2020-08-11 |
| WordPress 5.4 | 7.2 - 7.4 | 5.6 - 8.0 | 10.2 - 10.4 | 2020-03-31 |
| WordPress 5.3 | 7.2 - 7.4 | 5.6 - 8.0 | 10.2 - 10.4 | 2019-11-12 |
| WordPress 5.2 | 7.1 - 7.3 | 5.6 - 8.0 | 10.1 - 10.3 | 2019-05-07 |
| WordPress 5.1 | 7.1 - 7.3 | 5.6 - 8.0 | 10.1 - 10.3 | 2019-02-21 |
| WordPress 5.0 | 7.1 - 7.3 | 5.6 - 8.0 | 10.1 - 10.3 | 2018-12-06 |
| WordPress 4.9 | 7.0 - 7.2 | 5.5 - 5.7 | 10.0 - 10.2 | 2017-11-15 |
| WordPress 4.8 | 5.6 - 7.1 | 5.5 - 5.7 | 10.0 - 10.2 | 2017-06-08 |
| WordPress 4.7 | 5.6 - 7.1 | 5.5 - 5.7 | 5.5 - 10.1 | 2016-12-06 |
| WordPress 4.6 | 5.6 - 7.0 | 5.5 - 5.7 | 5.5 - 10.1 | 2016-08-16 |
| WordPress 4.5 | 5.6 - 7.0 | 5.5 - 5.7 | 5.5 - 10.1 | 2016-04-12 |
| WordPress 4.4 | 5.5 - 7.0 | 5.5 - 5.7 | 5.5 - 10.1 | 2015-12-08 |
| WordPress 4.3 | 5.5 - 5.6 | 5.5 - 5.6 | 5.5 - 10.0 | 2015-08-18 |
| WordPress 4.2 | 5.4 - 5.6 | 5.5 - 5.6 | 5.5 - 10.0 | 2015-04-23 |
| WordPress 4.1 | 5.4 - 5.6 | 5.5 - 5.6 | 5.5 - 10.0 | 2014-12-17 |
| WordPress 4.0 | 5.4 - 5.6 | 5.5 - 5.6 | 5.5 - 10.0 | 2014-09-04 |
| WordPress 3.9 | 5.3 - 5.5 | 5.5 - 5.6 | 5.5 - 10.0 | 2014-04-16 |
| WordPress 3.8 | 5.3 - 5.5 | 5.5 - 5.6 | 5.5 | 2013-12-12 |
| WordPress 3.7 | 5.3 - 5.5 | 5.5 - 5.6 | 5.5 | 2013-10-24 |

## □□□□□

□□□□ WordPress □□□□□□□□□□□□□□□□□□□□□□ PHP □□□□□□□□□

| WordPress | PHP | MySQL | MariaDB |
|---|---|---|---|
| WordPress 6.6+ | 7.2.24+ | 5.5.5+ | 5.5.5+ |
| WordPress 6.5+ | 7.0+ | 5.5.5+ | 5.5.5+ |
| WordPress 6.3+ | 7.0+ | 5.0.15+ | 5.5+ |
| WordPress 5.2+ | 5.6.20+ | 5.0.15+ | 5.5+ |
| WordPress 3.2+ | 5.2.4+ | 5.0.15+ | 5.5+ |

| WordPress | PHP | MySQL | MariaDB |
|---|---|---|---|
| WordPress 2.9+ | 4.3+ | 4.1.2+ | |
| WordPress 2.5+ | 4.3+ | 4.0+ | |
| WordPress 2.1+ | 4.2+ | 4.0+ | |
| WordPress 2.0+ | 4.2+ | 3.23.23+ | |

# Contributor Day

The Contributor Day is an event, usually parallel to a WordCamp, focused on contributing to the WordPress Community in any of its teams.

To participate in a Contributor Day it is only necessary to have a WordPress.org account, and, if possible, access to a Slack team channel.

# User account

## WordPress.org user account *(required)*

**If you already have a user account**, you can access the Login screen and log in. If you don't remember your password, you can get an email and change it.

**If you do not have a user account**, you can access the Registry of a new account and enter the desired username and an email account where you will receive the emails. Once completed, you will receive an email to validate the account, and you can access with your username and password.

To check your WordPress profile, visit your Profile Page.

## WordPress Slack user account *(recommended)*

When you create a WordPress.org user account, an email account is automatically created like your_wordpress.org_username@chat.wordpress.org, where *your_wordpress.org_username* is the chosen user.

To access the WordPress Slack, you can register from the SignUp page and enter your username in the email account field. At that time you will receive the instructions by email to continue. If you had previously logged in, you just have to visit the WordPress Slack home page and enter the Slack username and password.

Once you are there, please visit the #hosting channel.

# GitHub user account *(recommended)*

Some of the projects are developed through GitHub, so you may need an account.

**If you already have a GitHub account**, you can access the Login screen and log in.

**If you do not have a user account**, you can access the Registry of a new account and follow the steps.

## Sync your WordPress.org and GitHub profile

In order for your contributions on GitHub to appear on your WordPress profile, you need to relate the account to your profile. To do this, visit the Edit your WordPress Profile, and in the **GitHub Username** section add your GitHub account.

# Hosting Team projects

The projects of the Hosting team can vary, in addition to being able to cross with those of other teams.

Please, check our Hosting Projects.

文档
在此，您将找到一系列网站、文档或通用资料，需要定期查阅以保持 WordPress、 PHP、 MySQL、MariaDB 版本文档或任何与托管团队相关的内容处于最新状态。
理想情况下，至少每月一次，并在每次 WordPress 版本发布时，所有这些内容都应被审查并保持更新。

WordPress 托管团队手册

环境

服务器环境
本页面展示关于各版本 WordPress 理想环境的信息。
它记录了 Web 服务器的稳定版本、 PHP 与各主要 WordPress 版本的兼容性及可能的例外情况、 PHP
的生命周期、推荐的 PHP 扩展以及兼容的数据库。
这些信息可能与核心团队的信息不完全一致，因为它专门针对托管公司及其系统，而非开发环境。

兼容性

WordPress 兼容性
此页面展示新 WordPress 版本发布时可用服务器的兼容性。这样，如果您需要设置特定版本的 WordPress ，可以了解当时可用的稳定技术。
此外，该页面还包含一份随时间推移存在的最低要求列表，作为一种变更日志。

升级指南

升级 WordPress
本页详细说明了将 WordPress
从旧版本更新至新版本的步骤。该流程旨在确保每一步技术切换都不会丢失信息，力求实现最大程度的兼容性

。

WordPress 项目

系统要求

系统要求
系统要求页面包含两个独立的信息块。
第一个信息块列出了当前 WordPress 版本推荐的 PHP、 MySQL 和 MariaDB 版本。
第二个信息块是一个段落，记录了 PHP、 MySQL 和 MariaDB 的最低支持版本。

WordPress 与 PHP

PHP 兼容性与 WordPress 版本
此页面由核心团队负责维护，并针对每个 WordPress 版本进行更新。它包含一个表格，详细说明了与 PHP 版本的兼容性、可能的例外情况，以及是否包含 “测试版 ”或“存在例外兼容 ”的状态。

托管团队项目

项目

托管项目
此处列出了托管团队的所有项目。主要包括工具和文档。
此列表需要保持最新，以便个人能够通过 “未来五计划 ”参与其中。

测试

测试
测试页面包含有关测试工具的信息。

请注意，这些工具正常运行有一些非常规要求（例如 NodeJS），这些需要与测试团队同步。

学习

学习主机托管
WordPress 拥有大量与主机托管和系统管理相关的视频、课程和手册。
除了能够添加来自 Learn WordPress、WordPress.tv 等不同来源以及 WordPress
网络内各博客文章的新材料外，还需确保内容保持最新。

基本信息

团队代表

团队代表
关于团队代表的信息，应每年或在变更发生时更新，同时可能调整入职和离职流程，例如工具访问权限、 Slack 频道或任何其他持续发展事项。

参与进来

托管团队是做什么的？

为了提升托管体验并教育人们关于 WordPress 托管的最佳实践，托管团队正在推进 多个项目 。

托管手册 中有一套托管最佳实践，包括 性能 、可靠性 、安全性 和服务器环境 方面的建议。

该团队还运行并维护一套 自动化托管测试
，这些测试在许多托管公司的托管环境中运行。您可以设置测试在您的托管环境中运行，并通过修复错误、添
加功能或改进测试报告页面的设计来帮助完善这些工具。

改进托管手册

本手册包含有关托管团队的信息，以及运行 WordPress 的托管建议。
这些建议由团队整理，并作为 WordPress 中站点健康建议 的基础。它们既可作为学习托管 WordPress 的参考资料，也可作为帮助 WordPress 和托管服务商共同改进的途径。

自动化托管测试开发的改进

本手册正在接受审核和改进。您可以通过 Github查看进度并参与贡献。

您可以在 PHPUnit 测试运行器（问题） 和 PHPUnit 测试报告器（问题） 中提出改进建议或解决现有问题。
运行器仓库包含在主机上运行的托管测试部分，报告器仓库包含在 WordPress.org
上运行以接收和显示测试结果的插件。

新贡献者入门指南

Slack

你可以在 wordpress.slack.com找到全球 WordPress Slack。注册和入门指南可在 chat.wordpress.org

获取。加入后，请找到 #hosting频道。

你可以使用 WordPress.org 用户名 注册 。若没有用户名，请 创建 WordPress 个人资料 ，然后即可使用 MyWordPressUsername@chat.wordpress.org格式创建账户（将 MyWordPressUsername替换为你的 WordPress.org 用户名）。

会议内容概览

会议通常旨在交流本周内与托管相关的 WordPress 动态，并同步团队当前项目的进展状态。

议程通常包含以下主题：

问候环节
WordPress 社区动态
WordPress 托管团队事务
自由讨论

在问候环节 ，我们会进行 “挥手示意 ”，以了解与会人员情况并问候大家近况。欢迎新参与者进行自我介绍。

在WordPress 社区动态 部分，团队将讨论 WordPress 领域的最新动态及其对托管商的影响。例如新的 WordPress 版本发布、新提案或 WordCamp 活动。

会议的 WordPress 托管团队事务
部分主要关注托管相关项目（如手册或自动化托管测试）中的开放任务或问题。此外，我们也会在此处理来自
其他团队的请求。

其他所有待讨论事项将在 自由讨论
环节处理。每位与会者均可在此提出议题进行讨论，或仅向托管商们通报信息。

会议：记录笔记

记录会议笔记是提供帮助的好方法！

每次会议期间，我们都会创建一篇总结讨论内容的帖子，以便无法参会的人员了解情况。你可以在 托管团队的

Make 站点 上查看以往会议的笔记示例。

我们始终需要更多人加入！如果你有兴趣帮助记录笔记，请与团队代表之一交流，以获取团队 Make 站点的访问权限。

分步指南：如何创建会议记录

1. 申请访问权限

在 #hosting频道中申请访问 Hosting Make 网站的权限。
团队代表可以授予你所需的角色，以便你可以在 https://make.wordpress.org/hosting/wp-admin/ 上创建和编辑文章。
2. 从现有模板开始
在 WordPress 管理后台，转到 文章 → 所有文章 。
找到一篇最近的 "Hosting Team meeting summary" 或 "Hosting Meeting Notes" 文章。
使用 "Duplicate"/"Copy to new draft"
选项，基于该文章创建一个新的草稿。这会为你保留基本结构和标题。
3. 更新文章详情
将 标题 更改为遵循当前格式，例如： Hosting Team meeting summary 2025-11-26。
更新引言中的 日期和时间 ，并添加 Slack 存档的链接（如果有的话）。
检查 标签 并保持与之前的记录一致，通常包括： meetings、 summary 和 weekly-hosting-chat 。
4. 撰写记录
使用议程中的标题（例如： 问候 、关于 WordPress 社区 、关于 WordPress 托管团队 、
自由讨论 ），并为每个讨论的主题添加简短摘要。

记录中应包含的内容

撰写内容时，请尽量包含以下信息：

会议背景 – 日期、时间以及举行会议的 Slack 频道。
议程链接 ，如果已发布。
每个议程要点的关键讨论内容 ，以简短、中立的摘要形式呈现。
做出的决定 （如有），例如 “团队同意...”。
行动项 ，最好注明跟进人员以及（若已知）截止时间。
会议中提及的相关工单、 PR 和帖子的链接 。
下次会议信息 – 日期、时间和频道。
致谢 – 感谢协助准备议程、记录或具体工作的人员。

请保持语气中立、客观（不掺杂个人观点），并避免使用听起来像是代表 WordPress.org 官方声明的措辞，除非团队已明确同意。

审阅与发布

1. 当您对草稿满意后，将 文章状态 设置为 **“待审阅 ”**。

2. 在 #hosting频道分享草稿链接并请求快速审阅。
3. 团队代表或其他经验丰富的贡献者将进行审阅，必要时进行小幅调整，然后发布记录。

您可以在 此核心手册页面 查看关于写作风格的额外建议。由于这是来自核心团队的指南，并非所有建议都适用
— — 最重要的是注意不要无意中代表 WordPress.org 或团队发声（如果这不是 WordPress.org
或团队的决定），并在发布前进行同行审阅。

此外，您可以在 Slack 上访问 #hosting频道，查看并开始参与。团队总有事情可做！

贡献者日笔记

如果你想了解与托管团队一起的贡献者日可能是什么样子，不妨看看团队在以往贡献者日的工作笔记！

WordCamp US 2025 贡献者日
WordCamp Europe 2025 贡献者日
WordCamp Asia 2025 贡献者日
WordCamp Malaysia 2024 贡献者日
WordCamp Europe 2024 贡献者日
WordCamp Europe 2023 贡献者日
WordCamp US 2022 贡献者日
WordCamp Europe 2022 贡献者日
WordCamp Europe 2020 贡献者日
WordCamp US 2019 贡献者日

相关 WordPress 团队

如果您对 托管团队 感兴趣，您可能也会对这些团队感兴趣：

核心团队 ：核心团队负责构建 WordPress。无论您是经验丰富的 PHP、 HTML、JavaScript 或 CSS
开发人员，还是刚刚开始学习编码，核心团队都欢迎您的加入。您可以编写代码、修复错误、参与决
策讨论并协助开发。
核心性能团队 ：核心性能团队致力于监控、提升和推广 WordPress 核心及其周边生态系统的性能。
CLI 团队 ：WP-CLI 是用于与您的 WordPress 站点交互和管理的官方命令行工具。
文档团队
：好的文档能让人们在遇到困难时自助解决问题。文档团队负责创建文档，并一直在寻找写作者。团
队博客会讨论当前项目。
支持团队 ：在支持论坛或 IRC
中回答问题是最简单的贡献方式之一。每个人都知道一些问题的答案！此博客是讨论支持相关问题的
场所。
测试团队 ：测试团队使用我们手头的所有设备，巡查整个 WordPress
生态系统的流程。我们测试、记录并报告 WordPress 用户体验。通过持续的 "吃自家狗粮 "
和视觉记录，我们不仅了解问题所在，也明白什么是对的。我们沉浸在我们所创造产品的语境中，并
倡导用户体验。
Tide 团队 ：Tide 是一系列针对目录中每个插件和主题运行的自动化测试，然后在目录中显示 PHP 兼容性以及测试错误 /警告。

培训团队 ：WordPress
培训团队通过同步和异步学习，以及为讲师在实时环境中使用的可下载课程计划（通过
learn.wordpress.org），帮助人们学习使用、扩展 WordPress 并为 WordPress 做出贡献。

如果您喜欢教人们如何使用 WordPress 并为它构建东西，请立即停下您手头的事情，加入我们的团队！

团队徽章
以下是志愿者目前可以获得托管贡献者徽章的方式：

被采纳的分布式测试 PR
对最佳实践文档的贡献
协助在主机上设置自动化测试
帮助主持会议或撰写会议记录
定期积极参与会议并对讨论事项提供反馈
在贡献者日提供帮助

如果您已做出贡献但尚未获得徽章，我们深表歉意！ 请访问托管贡献者页面并提交申请
，详细说明您的贡献。如有疑问，请随时在 Slack 上联系任何团队代表。

[info]如果您有兴趣改进本手册，请查看 Github 手册仓库 ，或在官方 WordPress Slack的 #hosting 频道中留言。 [/info]

WordPress 托管团队手册
欢迎查阅 WordPress 托管手册 ！这里汇集了托管团队协作编写的 WordPress 托管最佳实践文档。请使用左侧的 "目录 "菜单浏览主题。

关于托管团队
WordPress 托管团队由经验丰富的专业人士组成，致力于弥合 WordPress 与托管环境之间的鸿沟。我们的成员包括系统管理员、托管服务提供商、插件开发者和服务器基础设施专家，
他们都热衷于优化生产环境中 WordPress 的性能和可靠性。

托管团队使命

托管团队成员通常是网络托管行业的专业人士；系统管理员、技术支持工程师、软件工程师、插件开发者等，
他们致力于支持 WordPress 与 WordPress 网络托管服务商之间的关系。

团队的目标和项目专注于促进 WordPress
开源项目与网络托管服务商之间的协作，通过在托管系统上进行分布式测试，并记录适用于托管 WordPress 的最佳实践。

托管团队相信，卓越的托管体验能让 WordPress 蓬勃发展。团队致力于：

维护测试软件 ，供托管服务商测试服务器与新版本 WordPress 的兼容性
发布权威文档 ，涵盖托管最佳实践、服务器配置以及 WordPress 部署的环境建议
维护兼容性指南 ，帮助托管服务商和网站所有者了解哪些服务器环境最适合每个 WordPress 版本
监控和评估 新的 PHP 版本、数据库（ MySQL/MariaDB）和 Web 服务器，以确保 WordPress
的兼容性和性能
提供技术资源 ，帮助 WordPress 社区就其托管基础设施做出明智决策
在托管行业内倡导 WordPress，并在更广泛的 WordPress 项目中代表托管需求

托管团队职责

托管团队专注于端到端的分布式托管测试，提供 WordPress 托管最佳实践文档，以及关于 WordPress 托管、可用工具和服务器环境建议（如 PHP、数据库和 Web 服务器）的指导。

参与方式： 如果您想了解更多并加入我们的工作，请随时访问托管手册的 参与页面 。

WordPress 托管团队是什么？

托管团队 致力于通过用户协作与教育，提升 WordPress 在各种托管环境下的用户体验。

我们欢迎所有人参与！这可能包括托管 WordPress 的人员、有兴趣了解更多关于 WordPress
托管的人员，或直接与 WordPress 托管服务商合作的人员，例如系统工程师、系统架构师、 DevOps
工程师以及从事核心、插件或主题开发的开发者。

托管团队何时开会？

托管团队在 WordPress Slack 的 #hosting频道中开会。讨论使用英语进行。请查看 WordPress 会议日历了解当前日程安排。

[info]如果您有兴趣改进本手册，请查看 Github 手册仓库 ，或在官方 WordPress Slack的 #hosting 频道中留言。 [/info]

学习主机托管
WordPress 拥有大量与主机托管和系统管理相关的视频、课程和手册。本页面为您精选了一些最出色的资源。

学习 WordPress

教程

PHP

测试插件对 PHP 版本的兼容性 (英文)

Performance

Improving website performance with caching (English) Image Optimization (English)
Improving Site Performance: An overview (English)

Security

7 Tips to improve website security (English)
Managing Updates (English)
Seguridad: formularios, acceso y contraseña (Spanish)

Development / Debug

Installing Node.js and npm for local WordPress development (English) Debugging in WordPress (English)
Tools: Site Health (English)
Local WordPress Installations (For Beginners) (English)
WordPress Troubleshooting Basics: Part 1 (English)

... and More

- Migrating your WordPress website to a different host and domain *(English)*
- How to choose the right hosting for your WordPress website *(English)*
- How to Install WordPress *(English)*

# Lesson Plans

- WebP images in WordPress *(English)*
- What Is W3 Total Cache *(English)*
- Improving Site Performance *(English)*
- How To Install WordPress On A Server *(English)*
- Local Install *(English)*
- Troubleshooting Basics *(English)*
- Backing Up Your WordPress Site *(English)*
- Regenerate Thumbnails *(English)*

# WordPress TV

- 2FA is coming... login de WordPress a este lado del muro *(Spanish)*
- What is the Domain Name System? *(English)*
- How DevOps Principles are Transforming IT Operations and Business Agility *(English)*
- From Fresh Grad to Professional: A Guide to Choosing Between Freelance and Employment in IT *(English)*
- Improving website performance with caching *(English)*
- WordPress database optimization and maintenance *(English)*
- Testing your products for PHP version compatibility *(English)*
- Installing Node.js and npm for local WordPress development *(English)*
- The WordPress Database *(English)*
- Debugging in WordPress *(English)*
- WP-CLI for PHP developers *(English)*
- 7 Tips to improve website security *(English)*
- Managing a WordPress Multisite Network *(English)*
- From PHP 5.3 to 8.2: what are you missing? *(English)*
- Front-end Performance and WordPress *(English)*

- Working with Version Control *(English)*
- WordPress Through the Terminal *(English)*
- Passkeys -The Future of Authentication *(English)*
- WordPress Data Privacy *(English)*
- Safeguarding Your Website – Smart WordPress Security *(English)*
- Entén i millora la velocitat del teu web WordPress *(Catalan)*
- Caballo de troya: vulnerabilidades en WordPress *(Spanish)*
- Administra WordPress des de la línia d'ordres (CLI) *(Catalan)*
- Secure the emails of your WordPress site with SPF, DKIM and DMARC *(English)*

性能
本节将介绍为 WordPress 配置服务以提升性能的基础知识。

缓存
WordPress 具有丰富的动态功能，但这需要付出代价。处理 PHP、查询数据库以及从外部 API 收集信息等任务都会消耗资源和时间。

缓存通过重用先前计算的结果来节省潜在繁重任务的时间，而不是为每个页面视图重新计 “
算。

缓存通常在一定时间后过期并重新生成，以显示最新内容。当项目从缓存中提供时，响应时间更快（通常来自
内存），并减轻服务器负载。

在典型的页面加载过程中，可能会按以下顺序检查各种缓存：

1. 本地浏览器缓存 / 本地存储 / Web 应用清单
2. 内容分发网络 (CDN)
3. 全页缓存 （反向代理 - Varnish 或 NGINX）
4. 全页缓存 （基于插件的文件式全页缓存）
5. 静态缓存 （JS、CSS、图像，使用静态缓存服务、 NGINX tryfiles 等）
6. 操作码缓存
7. 对象缓存 （wp_options、瞬态 API）
8. 片段缓存 （数据库、静态文件、瞬态 API）

每个缓存实例可能位于不同的域、服务器或计算实例上，由本地机器、远程服务器集群或请求链中的任何中间
计算机路由。建议咨询您的主机提供商，了解服务器端解决方案（如 Varnish 和 NGINX
），这些方案通常比插件和基于文件的全页缓存更快。

每个缓存可能具有以下任意配置：
数据存储： RAM、SSD 或机械硬盘。物理连接或网络数据传输。
输入 /输出延迟：从本地或远程服务器主板到 RAM、数据 I/O 或网络 I/O 的连接。

对于任何给定的页面加载，速度和用户体验将取决于所有服务的综合延迟，以及用户与 Web 应用程序交互时这些服务的处理顺序。

（示例 ：CSS；由 JavaScript、PHP 或预处理器生成。通过网络发送。最早显示：在第一个 HTML 响应的第一个 HTTP
数据包中的内联脚本。典型情况：由许多插件在多个请求中加载多个文件。折中方案：合并文件，缓存在本地
、CDN 或服务器 RAM 或 SSD 中。）

内容分发网络（ CDN）缓存

内容分发网络旨在优化服务器与来自不同地理位置的访客之间的网络延迟。数据分布在各个端点之间，访客会
从距离他们最近的端点获取服务。

除了优化网络延迟外， CDN 还可以作为另一层静态缓存和 /或全页缓存，运行在所有端点上。

确保 CDN
与所有其他缓存系统良好协作，并且当您从主服务器请求时，它能清除所有端点的缓存，这一点非常重要。否
则，某些地区的用户可能会得到过时的结果，这通常是一个难以排查的问题。

请注意，如果在 CDN 边缘节点上没有启用全页缓存，可能会增加 TTFB
（首字节时间），因为节点必须先从源服务器获取数据，然后才能将其提供给最终用户。

Full Page Cache
In order to display your content, WordPress does a lot of work under the hood, and all those calculations require server resources and time to complete. For starters, the PHP service on the server has to process the request, load WordPress core, your theme PHP files and all PHP scripts coming from your plugins. The majority of those PHP files make requests to your database, too, which adds to the overall resource footprint of your site.

The best way to cache these requests is to use a reverse proxy like NGINX or Varnish which stores the output directly into the server memory or hard disk. That saves a lot of processing power because cached content is served straight out of the reverse proxy without hitting your web server, the PHP service, or your database service at all. If a reverse proxy is not available on your current server setup, you can fallback to storing cached content into your file system. It's slower than reverse proxies and a hit reaches your web server and your PHP service at least once, so it can direct the request to the proper cached file but still - it's much faster than doing all the computing for every request.

Full Page Caching stores the HTML output of a request, but all the CSS, JS, images and font files will have to be loaded separately too. They are handled separately and optimizing them is worth investing the time and effort. Static caching can have great effect on those resources. You can often use the same reverse proxy to static resources in the server memory - CSS, JS, Fonts, Images and serve them directly.

It's important to have the ability to expire caches when necessary to avoid serving visitors old data. When available, selective caching is preferred over purging the entire cache, to avoid the cost of

WordPress regenerating every page for the site. Furthermore, it's good practice to exclude certain types of pages from your full page caching completely because they are different for each user. For example, if you have an online store, it's imperative that your cart, checkout and profile pages are completely dynamic. In general, it’s a good idea to exclude all logged in users from the cache because they are supposed to see personalized content. Another important aspect is the default caching period, which can be different for each website depending on how often data is changed.

Full Page Cache Example



> 2005 □□ WordPress □□□□□□□□□□ ——
> □□□□□□□□□□□□□□□□□□□□□□□□□□□□ PHP
> □□□□□□□□□□□□□□□□□□□□□□□□□□□□ WordPress
> □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
> □□□□□□□ Redis □ Memcached
> □□□□□□□□□□□□□□□□□□□□□□□□ drop-ins□□□□ WordPress
> □□□□□□□

source: scalewp.io

What does this mean? Think of a standard WordPress homepage displaying the most recent posts. Each of these posts has quite a bit of information associated with it WordPress must look up such as the author, categories, tags, and excerpt.

Support for a persistent object cache gives WordPress, plugins, and themes, a place to store that data for reuse. While these items are cached, PHP execution time is improved while lessening the load on the database. It's particularly helpful in situations where much of the page is difficult to cache from the front-end, like for authenticated traffic or e-commerce applications.

For these reasons, persistent object caching support is commonly offered with managed WordPress hosting.

Object Cache Example

> Transients are inherently sped up by caching plugins, where normal Options are not. A memcached plugin, for example, would make WordPress store transient values in fast memory instead of in the database. For this reason, transients should be used to store any data that is expected to expire, or which can expire at any time. Transients should also never be assumed to be in the database, since they may not be stored there at all.

来源： WordPress 通用 API 手册

操作码缓存

Web 服务器必须读取、编译并运行每个 PHP 脚本。操作码缓存会将每个 PHP
脚本的编译副本存储在内存或磁盘中。当 Web 服务器开始处理 WordPress 的 PHP
脚本时，它会检查操作码缓存中是否有该 PHP 脚本的缓存副本。如果存在缓存副本， Web
服务器可以直接使用缓存副本来运行 PHP 脚本，而无需再次读取和编译脚本。跳过 PHP 脚本的读取和编译过程可以显著提升 Web 服务器的资源利用率，并使 WordPress
能够处理比原本多得多的请求。这对于动态内容和认证流量尤其有帮助，因为全页面缓存在这些情况下效果不
佳。

与任何缓存一样，操作码缓存可能会阻止更改生效，直到缓存过期或被清除。具体到操作码缓存，这意味着会
加载旧版本的编译 PHP 代码。在更新插件、主题或 WordPress
核心时，应从缓存中清除相应的文件，以避免继续加载旧版本。

片段缓存

这种缓存方法允许保存原本不可缓存的动态网站内容片段。它特别适用于页面主体为静态但包含特定动态元素
（如购物车）的网站，或会员制网站。

在 WordPress 环境中，开发者通常使用 WordPress 瞬态 /对象缓存 API
来存储页面部分内容。这种情况下，提供持久化对象缓存可使这些内容在数据库外部实现缓存。

WordPress
本身不支持在前端缓存中单独存储这些片段，这意味着需要手动配置页面的可缓存部分，并配置前端缓存（无
论是 Nginx、Varnish
或其他方案）以支持片段缓存。这通常属于高级技术，仅适用于动态流量需求极高的网站或托管平台。

缓存清理 /清除 /清空
缓存清理与缓存存储同等重要。必须确保在必要时清除所有缓存层级。

片段缓存是指将高成本或耗时的服务器端操作临时存储，以避免给 Web
服务器带来负担并延迟向访客交付内容。对于生成菜单标记、小部件标记以及缓慢的 MySQL 或 HTTP
响应等操作，这已成为常见做法。核心目前使用瞬态来缓存对 WordPress.org API 的 HTTP
调用，以获取更新和事件信息。

当片段缓存与全页缓存适当搭配时，效果尤为显著。例如，每个页面都显示的统一 <footer>
标记可以临时存储。当服务器需要重建静态缓存文件并找到片段时，可以避免服务器在每个页面上运行菜单 / 小部件查询来生成页脚标记。

对核心资源进行全局缓存时应格外谨慎。如果网站菜单依赖于动态的 .current-menu-item
类，将菜单标记存储在片段中会 “固化 ”该类，导致用户导航时不再高亮显示正确的页面。任何 WordPress 核心资源的缓存都应选择加入，并集成适当的刷新机制，以便用户在修改资源时进行更新。

片段缓存应始终使用瞬态 API，而不是直接使用 wp_cache_* 函数。在没有持久对象缓存的环境中，
set_transient() 会将缓存值存储在数据库的 wp_options 表中。然而，当启用对象缓存时，
set_transient() 将包装 wp_cache_set()。

PHP

请参阅 PHP 优化 。

可靠性
当您的网站上线后，如何保持其稳定运行？如果网站出现故障，您可以采取哪些措施？在大多数情况下，
WordPress 与其他 Web 应用程序的最佳实践相同，但此处详细说明了一些差异和建议。

备份

请参阅 备份 。如果您需要更具体的文档，请参考 备份数据库 和备份 WordPress 文件 。

监控

请参阅 监控 。

版本控制

请参阅 版本控制 。

服务器环境
尽管 WordPress 几乎能在任何环境中运行，但某些环境在功能和性能方面更为优化。以下是针对 WordPress 高效运行的最低服务器环境配置建议，同时考虑了使用第三方插件和主题的 WordPress
网站，这些插件和主题通常会引入额外的服务器级要求。

WordPress 环境推荐
快速推荐：

WordPress 6.8 服务器兼容性
WordPress 6.7 服务器兼容性
WordPress 6.6 服务器兼容性
WordPress 6.5 PHP 兼容性
WordPress 6.4 PHP 兼容性
WordPress 6.3 PHP 兼容性

所有已发布的兼容性文章可在以下位置查看：

版本兼容性

Web Server

Web
服务器是一种软件，它接收并处理来自网站访问者计算机的网络请求，然后将相应的网页数据返回给用户。存
在许多不同类型的 Web 服务器软件，它们运行在不同的操作系统上。通常，只要您的 Web 服务器支持并执行 PHP 文件，它就应该能够运行 WordPress。

两种最常见的、也是 WordPress 推荐的 Web 服务器软件是：

Apache HTTPD 2.4
nginx 1.26 & 1.27

已知与 WordPress 兼容良好的、由网络托管公司和开发者使用的其他软件包括：

Angie 1.7
LiteSpeed Web Server 6.3 / 6.2 / 6.1 / 6.0 / 5.4
OpenLiteSpeed 1.8 / 1.7

以上是撰写本文档时（针对 WordPress 6.8）的最新版本。请务必保持您的 Web
服务器为最新版本，以确保最佳性能！

PHP

PHP 是 WordPress
代码所基于的编程语言。该语言在服务器上运行，为了安全性和功能性，保持其更新至关重要。

WordPress 支持许多 PHP 版本，其中一些甚至已过时（ 参见 PHP 兼容性与 WordPress 版本
）。对于托管公司，建议如下：

WordPress 版本
以下是特定 WordPress 版本的详细信息、该版本的 PHP 兼容性以及发布时与 PHP 兼容性相关的开发工单。与 PHP 兼容性相关的工单可随时通过 搜索 WordPress Trac查找。

WordPress 6.8

PHP 8.1 (安全支持)
PHP 8.2 (活跃支持)
PHP 8.3 (活跃支持)
PHP 8.4 (候选支持)

重要提示： WordPress 6.8 完全兼容 PHP 7.2 (1)、 7.3 (1)、7.4 (1)、8.0(1)、8.1 和 8.2，并且 Beta 兼容 PHP 8.3 和 PHP 8.4。 自 2025 年 4 月发布的 WordPress 6.8 起，不再使用 “附带例外兼容 ”这一术语。

“Beta 兼容 ”或 “Beta 支持 ”是什么意思？

“Beta 兼容 ”或 “Beta 支持 ”意味着 WordPress 核心正在积极努力实现与该 PHP
版本的完全兼容，但可能仍有一些问题正在解决过程中。以下是概述 PHP 8.3 和 8.4 Beta
支持已知问题的工单。关于 PHP 版本何时从 Beta 转为完全支持 的更多信息，由核心团队记录。

PHP 8.3
当使用 “Beta 兼容 ”的 PHP 版本时，可能会在错误日志、 wp-admin
或页面上看到弃用通知。弃用通知不是错误，而是表明通知中引用的代码将在未来的 PHP
版本中更改。出现弃用通知时， PHP 代码将继续工作，没有任何问题。

#59231: 为 PHP 8.3 做准备。 。注意：已关闭 /已修复
#59232: 引入 #[Override] 属性来标记重载方法
此属性通过明确标记重载方法来帮助防止编码错误。它还有助于重构、调试以及捕获父类中潜
在的破坏性更改。 注意：已有补丁，但移至未来版本。
#59233: 改进 unserialize() 的错误处理 。maybe_unserialize()
函数仍可能遇到带有尾随字节的数据。 注意：移至未来版本。
PHP 8.4
弃用通知。弃用通知不是错误，而是表明通知中引用的代码将在未来的 PHP
版本中更改。出现弃用通知时， PHP 代码将继续工作，没有任何问题。
#62061: 为 PHP 8.4 做准备。 。注意：已关闭 /已修复
Other PHP Related Tickets

#51525: Add new functions apply_filters_single_type() and apply_filters_ref_array_single_type().. Note: Moved to Future Release.
#54183: Tests: decide on how to handle deprecations in PHPUnit. Note: Moved to Future Release.

#54537: Tests: Enable PHP version check once PHP 8.0 compatibility is achieved..
Note: Moved to Future Release.

#58874: Code Modernization: Consider using the null coalescing operator.. Note:
Moved to Future Release.

#59234: Introduce a wp_json_decode() function, including validation when available.
Note: This ticket has been closed and won’t be moving forward.

WordPress 6.7

PHP 8.1
PHP 8.2
PHP 8.3

IMPORTANT: WordPress 6.7 is compatible with exceptions with PHP 8.0, PHP 8.1, PHP 8.2, and beta compatible with PHP 8.3, and PHP 8.4.

What does 'compatible with exceptions' mean?

PHP 8.0

#48689: Filesystem WP_Filesystem_FTPext and WP_Filesystem_SSH2 when connect
fails. An investigation is underway as to why on some occasions the access to the
files returns some type of error. NOTE: Has a patch.
#49728: Prepare for the internal functions throwing TypeError or ValueError
exceptions on unexpected types/values. Internal functions will throw an exception if the function call arguments are of a type that is not expected. NOTE: Has a patch.

#51019: convert_smilies() fails on large tags. The function fails when dealing with large HTML tags, particularly when an image with a large data URL is included in the post content. NOTE: Has a patch, but moved to WordPress 6.8.
#55121: classic widgets with no settings won't show up in 5.9. Classic widgets with no settings do not appear correctly in WordPress 5.9 and above. This is due to changes in how widgets are handled in the block editor, causing compatibility problems with older widget setups. NOTE: Has a patch, but moved to WordPress 6.3.

#55257: map_deep() function incompatibility with incomplete objects in PHP 8.0+. The function becomes incompatible with incomplete objects when running on PHP 8.0 or higher. NOTE: Has a patch, but moved to Future Release.
#59649: Named parameters. WordPress does not support named parameters. PHP 8.0 supports optionally calling functions and class methods by specifying the parameter name, instead of calling them on the order of parameters that they are declared. PHP, and many other programming languages, support positional parameters: The caller passes the parameters in the same order the
function/method declares its parameters. NOTE: Moved to WordPress 6.8.

#60745: WP_Query::parse_query() does not handle invalid query arg values. The function does not properly handle invalid query argument values. This results in PHP fatal errors when unintended data types, like arrays, are passed where scalars are expected. NOTE: Has a patch, but moved to WordPress 6.8.
PHP 8.1
并非所有 “向不可为空参数传递 null”的问题都已被发现。在 PHP
中，你可以明确告知函数应接受何种类型的信息。如果你告诉函数期望某种特定类型的信息，
却完全没有提供任何信息（ null 类似于表示 “无”），那么 PHP
会感到困惑并报错。当有人意外地向一个未设计为处理 “无 ”的函数传递 “无 ”
时，就会出现此问题。

#53465: htmlentities() 需要显式设置 flags 参数的默认值。 根据 htmlentities()，PHP 8.1 中 flags 的默认值 “已从 ENT_COMPAT 更改为 ENT_QUOTES | ENT_SUBSTITUTE | ENT_HTML401 ” 。正在调查核心中此功能的所有用例。 注意：已有补丁。

#57579: 将大多数 strip_tags() 替换为 wp_strip_tags()。
提议通过最小化加载过程中的不必要文件系统调用来优化块模板的处理方式。
注意：已有补丁，但移至未来版本。

#57580: 避免 add_submenu_page() 中 null 参数导致的错误。 当 null
作为参数传递时，由 wp_normalize_path() 函数引起的已弃用错误。此问题主要由于插件在
add_submenu_page() 函数中错误地传递了 null。注意：已有补丁，但移至未来版本。

#61179: 关于在 widgets.php 中传递 null 的已弃用消息 解决 widgets.php
中的一个已弃用消息问题。 注意：已有补丁，但移至 WordPress 6.7。
PHP 8.2

#55603: utf8_{encode|decode} deprecation with pending decision on requiring a PHP extension. NOTE: Has a patch, but moved to WordPress 6.8.
#56034: Unknown dynamic properties' deprecation. NOTE: Moved to Future Release.

- **#57304**: Add SensitiveParameter attribute to DB connection and login variables. This enhancement aims to protect sensitive data in case of errors, making it less likely to be exposed in logs or bug reports. *NOTE: Moved to WordPress 6.7.*
- **#60875**: Handler proposal for known dynamic properties that are initialized and set late only when getting its value. Handling dynamic properties that are initialized only when accessed. Since PHP 8.2 deprecates dynamic (non-declared) properties, the proposal aims to pre-declare these properties while retaining the current design where they are lazily initialized. *NOTE: Moved to WordPress 6.8.*
- **#61154**: Fix the 'attributes' dynamic property in WP_Block. Fixing the 'attributes' dynamic property in the `WP_Block` class. *NOTE: Has a patch, but moved to WordPress 6.8.*
- **#61890**: Handle WP_Term dynamic properties for PHP 8.2. Handling of dynamic properties in the `WP_Term` class to ensure compatibility. *NOTE: Has a patch, but moved to WordPress 6.8.*

*What does "beta" mean?*

- PHP 8.3
  - Deprecation notices. A deprecation notice is not an error, but is an indicator that the code being cited in the notice will be changed in future php versions. With a deprecation notice, the PHP code will continue to work and nothing is broken.
  - **#59231**: Prepare for PHP 8.3.. *NOTE: Has a patch, but moved to WordPress 6.7.*
  - **#59232**: Introduce #[Override] attribute to mark overloaded methods This attribute helps prevent coding errors by making it clear when a method is overloaded. It also assists with refactoring, debugging, and catching potential breaking changes in the parent class. *NOTE: Has a patch, but moved to Future Release.*
  - **#59233**: Improve error handling for unserialize(). `maybe_unserialize()` function could still be confronted by data with trailing bytes. *NOTE: Moved to Future Release.*
  - **#59654**: PHP 8.x: various compatibility fixes for WordPress 6.7. This ticket acts as a central hub for smaller patches that fix specific PHP 8.x failures. It continues the work from previous releases, ensuring that WordPress maintains compatibility with newer PHP versions like PHP 8.0, 8.1, 8.2, and upcoming versions like PHP 8.3. *NOTE: Moved to WordPress 6.7.*
- PHP 8.4
  - □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ PHP □□□□□□□□□□□□□□□ PHP □□□□□□□□□□□□□□□□□
  - **#62061**: □ PHP 8.4 □□□□ □□□ □ □□□□□□□□

□□□□□□

- PHP

#51525: 新增函数 apply_filters_single_type() 和 apply_filters_ref_array_single_type()
。
#54183: 测试：决定如何处理 PHPUnit 中的弃用问题
#54537: 测试：一旦实现 PHP 8.0 兼容性，就启用 PHP 版本检查。
#58874: 代码现代化：考虑使用空值合并运算符。
#59234: 引入 wp_json_decode() 函数，包括在可用时进行验证

WordPress 6.6

PHP 8.1
PHP 8.2
PHP 8.3

重要提示： WordPress 6.6 与 PHP 8.1 和 PHP 8.2 存在例外兼容 ，与 PHP 8.3 为 Beta 兼容 。

“存在例外兼容 ”是什么意思？

PHP 8.1
并非所有 “向不可为 null 的参数传递 null”的问题都已被发现。 在 PHP
中，你可以明确告诉函数它应该接受什么类型的信息。如果你告诉函数期望某种类型的信息，
而你却什么都没给它（ null 就像是说 “什么都没有 ”），那么 PHP
会感到困惑并报错。当有人不小心给一个函数传递了 “什么都没有 ”，而该函数并未设计为处理 “
什么都没有 ”时，就会发生此问题。
_ htmlentities() 需要显式设置 flags 参数的默认值 _。根据 htmlentities()，PHP 8.1 中
flags 的默认值 “已从 ENT_COMPAT 更改为 ENT_QUOTES | ENT_SUBSTITUTE |
ENT_HTML401”。正在调查 WordPress
核心中此功能的所有使用案例。注意：已有补丁，但移至 WordPress 6.7。
将大多数 strip_tags() 替换为 wp_strip_tags()。在极少数情况下， strip_tags()
函数会接收到 null 值，这会生成一个关于字符串已弃用的警告。注意：已有补丁。
更新 is_serialized 函数以接受 Enums。 Enums 与旧版 PHP
不向后兼容。注意：已有补丁，但移至 WordPress 6.7。
PHP 8.2

utf8_{encode|decode} 弃用 ，关于是否需要 PHP 扩展的决定待定。注意：已有补丁，但移至
WordPress 6.7。
“未知动态属性 ”弃用。注意：移至 WordPress 6.7。

“Beta”是什么意思？

PHP 8.3
弃用通知 。弃用通知不是错误，而是表明通知中引用的代码将在未来的 PHP
版本中更改。出现弃用通知时， PHP 代码将继续工作，没有任何问题。

改进 unserialize() 的错误处理 。 maybe_unserialize()
函数仍可能遇到带有尾随字节的数据。注意：移至 WordPress 6.7。

WordPress 6.5

PHP 8.1
PHP 8.2
PHP 8.3

重要提示： WordPress 6.5 与 PHP 8.0、PHP 8.1 和 PHP 8.2 存在例外兼容 ，与 PHP 8.3 为 Beta 兼容 。

“存在例外兼容 ”是什么意思？

PHP 8.0
命名参数 。WordPress 不支持命名参数。
连接失败时的文件系统 WP_Filesystem_FTPext 和 WP_Filesystem_SSH2。
PHP 8.1
并非所有 “向不可为 null 的参数传递 null”的问题都已被发现。
htmlentities() 等函数需要显式设置 flags 参数的默认值 。
将大多数 strip_tags() 替换为 wp_strip_tags()。
PHP 8.2

utf8_{encode|decode} 弃用 ，关于是否需要 PHP 扩展的决定待定。
未知动态属性 弃用。

“Beta”是什么意思？

PHP 8.3
弃用通知：弃用通知不是错误，而是表明通知中引用的代码将在未来的 PHP
版本中更改。出现弃用通知时， PHP 代码将继续工作，没有任何功能损坏。

WordPress 6.4

PHP 8.1
PHP 8.2

重要提示： WordPress 6.4 与 PHP 8.0、PHP 8.1 和 PHP 8.2 存在例外兼容 ，与 PHP 8.3 为 Beta 兼容 。

“存在例外兼容 ”是什么意思？

PHP 8.0
命名参数 。WordPress 不支持命名参数。
连接失败时的文件系统 WP_Filesystem_FTPext 和 WP_Filesystem_SSH2。

PHP 8.1
并非所有 “向不可为 null 的参数传递 null”的问题都已被发现。 htmlentities() 等函数需要显式设置 flags 参数的默认值 。
将大多数 strip_tags() 替换为 wp_strip_tags()。
PHP 8.2

utf8_{encode|decode} 弃用 ，关于是否需要 PHP 扩展的决定待定。
未知动态属性 弃用。

“Beta”是什么意思？

PHP 8.3
弃用通知：弃用通知不是错误，而是表明通知中引用的代码将在未来的 PHP
版本中更改。出现弃用通知时， PHP 代码将继续工作，没有任何问题。

WordPress 6.3

PHP 8.1
PHP 8.2

重要提示： WordPress 6.3 与 PHP 8.0 和 PHP 8.1 存在例外兼容 ，与 PHP 8.2 为 Beta 兼容 。

“存在例外兼容 ”是什么意思？

PHP 8.0
命名参数。 WordPress 不支持命名参数。
连接失败时的文件系统 WP_Filesystem_FTPext 和 WP_Filesystem_SSH2。
PHP 8.1

htmlentities() 等函数需要显式设置 flags 参数的默认值 。
将大多数 strip_tags() 替换为 wp_strip_tags()。
对未知设置的 unregister_setting()。

“Beta”是什么意思？

PHP 8.2
弃用通知：弃用通知不是错误，而是表示通知中引用的代码将在未来的 PHP
版本中更改。出现弃用通知时， PHP 代码将继续工作，没有任何问题。

关于 PHP

自 2022 年 11 月 26 日起， PHP 社区仅以 仅安全修复 模式维护 PHP 8.1。将 PHP 保持在最新的稳定版本对于 WordPress 的速度和安全性至关重要。

PHP 社区不再维护 PHP 8.1 之前的版本，尽管它们可能会从操作系统发行版获得安全更新。

已终止支持的 PHP 版本：

PHP 8.3：2027 年 12 月 31 日
PHP 8.2：2026 年 12 月 31 日
PHP 8.1：2025 年 12 月 31 日
PHP 8.0：2023 年 11 月 26 日 最后版本： 8.0.30
PHP 7.4：2022 年 11 月 28 日 最后版本： 7.4.33
PHP 7.3：2021 年 12 月 6 日 最后版本： 7.3.33
PHP 7.2：2020 年 11 月 30 日 最后版本： 7.2.34
PHP 7.1：2019 年 12 月 1 日 最后版本： 7.1.33
PHP 7.0：2019 年 1 月 10 日 最后版本： 7.0.33
PHP 5.6：2018 年 12 月 31 日 最后版本： 5.6.40
PHP 5.5：2016 年 7 月 21 日 最后版本： 5.5.38
PHP 5.4：2015 年 9 月 3 日 最后版本： 5.4.45
PHP 5.3：2014 年 8 月 14 日 最后版本： 5.3.29
PHP 5.2：2011 年 1 月 6 日 最后版本： 5.2.17
PHP 5.1：2006 年 8 月 24 日 最后版本： 5.1.6
PHP 5.0：2005 年 9 月 5 日 最后版本： 5.0.5
PHP 4.4：2008 年 8 月 7 日 最后版本： 4.4.9
PHP 4.3：2005 年 3 月 31 日 最后版本： 4.3.11
PHP 4.2：2002 年 9 月 6 日 最后版本： 4.2.3
PHP 4.1：2002 年 3 月 12 日 最后版本： 4.1.2
PHP 4.0：2001 年 6 月 23 日 最后版本： 4.0.6

PHP 扩展

当可用时， WordPress 核心会利用各种 PHP 扩展。如果缺少首选扩展， WordPress 要么需要做更多工作来完成该模块辅助的任务，要么在最坏情况下，会移除相关功能。

说明：

一些扩展已内置在 PHP 中（或默认启用），在典型的托管环境中无需特殊操作。
其他扩展是可选的，可能需要根据 PHP 的打包方式来安装 /启用。

以下列表假设 PHP 版本 >= 7.2（撰写本文时 WordPress 支持的最低 PHP 版本）。

内置扩展（无需托管操作）

以下扩展是 PHP 自身的组成部分，在支持的 PHP 版本（ >= 7.2）中始终启用。无需进行托管配置：

pcre - 在 PHP 和 WordPress 中广泛使用的正则表达式引擎。

必需扩展

□□□□□ PHP □□□ WordPress □□□□□ □□ □□

- json - □□□□□□□□□□□□□□ JSON □□□□□□□□ PHP >= 8.0 □□□□□□□ PHP 7.x □□□□□□□□□□□□□□□□□□□□
- mysqli - □□□ MySQL/MariaDB □□□□□□□□□□□ PHP □□□□ mysqlnd □□□□□□□□□□ `mysqlnd` □□□□□□□ `mysqli` □

□□□□□ PHP □□□ □□□□ □□□□□ WordPress □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

- curl (PHP >= 7.3 requires libcurl >= 7.15.5; PHP >= 8.0 requires libcurl >= 7.29.0 ; PHP >= 8.4 requires libcurl >= 7.61.0) - Performs remote request operations.
- dom (requires libxml) - Used to validate Text Widget content and to automatically configure IIS7+.
- exif - Works with metadata stored in images.
- fileinfo - Used to detect MIME types of file uploads.
- hash - Used for hashing, including passwords and update packages. (Always enabled in PHP >= 7.4; on PHP 7.2-7.3 it is commonly enabled but not guaranteed.)
- igbinary - Increases performance as a drop in replacement for the standard PHP serializer.
- imagick (requires ImageMagick >= 6.2.4) - Provides better image quality for media uploads. See WP_Image_Editor for details. Smarter image resizing (for smaller images), and PDF thumbnail support when Ghost Script is available.
- intl (PHP >= 7.4.0 requires ICU >= 50.1) - Enable to perform locale-aware operations including but not limited to formatting, transliteration, encoding conversion, calendar operations, conformant collation, locating text boundaries and working with locale identifiers, timezones and graphemes.
- mbstring - Used to properly handle UTF8 text.
- openssl (PHP 7.1-8.0 requires OpenSSL >= 1.0.1 / < 3.0; PHP >= 8.1 requires OpenSSL >= 1.0.2 / < 4.0; PHP >= 8.4 requires OpenSSL >= 1.1.1 / < 4.0) - SSL-based connections to other hosts.
- xml (requires libxml) - Used for XML parsing, such as from a third-party site.
- zip (requires libzip >= 0.11; recommended libzip >= 1.6) - Used for decompressing Plugins, Themes, and WordPress update packages.

The PHP extensions listed below are *recommended* to allow some WordPress cache (if necessary). APCu, Memcached, and Redis are alternatives of which only one needs to be used.

- apcu – In-memory key-value store for PHP (former APC stripped of opcode caching).
- memcached (requires libmemcached >= 1.0.0) - memcached is a high-performance, distributed memory object caching system, generic in nature, but intended for use in speeding up dynamic web applications by alleviating database load.

- opcache - PHP can be configured to preload scripts into the opcache when the engine starts.
- redis - PHP extension for interfacing with Redis.

The PHP extensions listed below are *optional* to improve some WordPress functionality.

- timezonedb - Timezone Database to be used with PHP's date and time functions

For the sake of completeness, below is a list of the remaining PHP modules WordPress *may* use in certain situations or if other modules are unavailable. These are fallbacks or optional and not necessarily needed in an optimal environment, but installing them won't hurt.

- bc - For arbitrary precision mathematics, which supports numbers of any size and precision up to 2147483647 decimal digits.
- filter - Used for securely filtering user input.
- image (requires libgd >= 2.1.0; requires zlib >= 1.2.0.4; optional freetype2) - If Imagick isn't installed, the GD Graphics Library is used as a functionally limited fallback for image manipulation.
- iconv (requires libiconv/POSIX) - Used to convert between character sets.
- shmop - Shmop is an easy to use set of functions that allows PHP to read, write, create and delete Unix shared memory segments.
- simplexml (requires libxml) - Used for XML parsing.
- sodium (bundled with PHP as of 7.2.0; requires libsodium >= 1.0.8) - Validates signatures and provides securely random bytes.
- xmlreader (requires libxml) - Used for XML parsing.
- zlib (requires zlib >= 1.2.0.4) - Gzip compression and decompression.

These extensions are used for file changes, such as updates and plugin/theme installation, when files aren't writeable on the server.

- ssh2 (requires OpenSSL and libssh >= 1.2; recommended libssh >= 1.2.9) - Provide access to resources (shell, remote exec, tunneling, file transfer) on a remote machine using a secure cryptographic transport.
- ftp - Implement client access to files servers speaking the File Transfer Protocol (FTP).
- sockets - Implements a low-level interface to the socket communication functions based on the popular BSD sockets.

The priority of the transports are Direct file IO, SSH2, FTP PHP Extension, FTP implemented with Sockets, and FTP implemented through PHP alone.

# System Packages

- curl (recommended >= 8.4)
- Ghost Script (recommended Ghost Script >= 10.0)- Enables Imagick/ImageMagick to generate PDF thumbnails for the media library. See Enhanced PDF Support in WordPress 4.7 for details.
- ImageMagick (recommended ImageMagick >= 7.1) - Required by Imagick extension.
- OpenSSL (recommended >= 3.0)
- WebP
- AVIF

## □□□

WordPress □□□□ MySQL □□□□□□□□□□

## □□□□□

□□□ MySQL □ MariaDB □□□□□□□□□□□□□

| □□ | □□ | □□□□□□□□ |
|---|---|---|
| MySQL | 8.4 LTS | 2032□ 4□ |
| MySQL | 8.0 LTS | 2026□ 4□ |
| MariaDB | 11.8 LTS | 2030□ 6□ |
| MariaDB | 11.4 LTS | 2029□ 5□ 29□ |
| MariaDB | 10.11 LTS | 2028□ 2□ 16□ |
| MariaDB | 10.6 LTS | 2026□ 7□ 6□ |

# End of life MySQL versions

The following versions of MySQL have reached their end of life (EOL) and are no longer supported by the MySQL community. **It is highly recommended to upgrade to a supported version for security and performance reasons.**

| Version | Initial GA Release | Final Release | EOL Date |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 9.2* | January 21, 2025 | 9.2.0 (January 21, 2025) | April 15, 2025 |
| 9.1* | October 15, 2024 | 9.1.0 (October 15, 2024) | January 21, 2025 |
| 9.0* | July 1, 2024 | 9.0.1 (July 23, 2024) | October 15, 2024 |
| 8.3* | January 16, 2024 | 8.3.0 (January 16, 2024) | July 1, 2024 |
| 8.2* | October 25, 2023 | 8.2.0 (October 25, 2024) | January 16, 2024 |
| 8.1* | July 18, 2023 | 8.1.0 (July 18, 2023) | October 25, 2023 |
| 5.7 | October 21, 2015 | 5.7.44 (October 25, 2023) | October 21, 2023 |
| 5.6 | February 5, 2013 | 5.6.51 (January 20, 2021) | February 5, 2021 |
| 5.5 | December 3, 2010 | 5.5.62 (October 22, 2018) | December 3, 2018 |
| 5.1 | November 14, 2008 | 5.1.73 (December 3, 2013) | December 31, 2013 |
| 5.0 | October 19, 2005 | 5.0.96 (March 21, 2012) | March 21, 2012 |

# End of life MariaDB versions

The following versions of MariaDB have reached their end of life (EOL) and are no longer supported by the MariaDB community. **It is highly recommended to upgrade to a supported version for security and performance reasons.**

| Version | Initial GA Release | Final Release | EOL Date |
|---|---|---|---|
| 11.5 | August 15, 2024 | 11.5.2 (August 15, 2024) | November 21, 2024 |
| 11.3 | February 19, 2024 | 11.3.2 (February 19, 2024) | May 16, 2024 |
| 11.2 | November 21, 2023 | 11.2.6 (November 4, 2024) | November 21, 2024 |
| 11.1 | August 21, 2023 | 11.1.6 (August 14, 2024) | August 21, 2024 |
| 11.0 | June 7, 2023 | 11.0.6 (May 17, 2024) | June 6, 2024 |
| 10.10 | November 17, 2022 | 10.10.7 (November 14, 2023) | November 17, 2023 |
| 10.9 | August 22, 2022 | 10.9.8 (August 14, 2023) | August 22, 2023 |
| 10.8 | May 21, 2022 | 10.8.8 (May 10, 2023) | May 20, 2023 |
| 10.7 | February 14, 2022 | 10.7.8 (February 6, 2023) | February 9, 2023 |
| 10.5 | June 24, 2020 | 10.8.8 (May 10, 2023) | June 24, 2025 |
| 10.4 | June 16, 2019 | 10.4.34 (May 17, 2024) | June 18, 2024 |
| 10.3 | May 25, 2018 | 10.3.39 (May 10, 2023) | May 25, 2023 |
| 10.2 | May 23, 2017 | 10.2.44 (May 21, 2022) | May 23, 2022 |

| Version | Initial GA Release | Final Release | EOL Date |
|---|---|---|---|
| 10.1 | October 17, 2015 | 10.1.48 (November 4, 2020) | October 17, 2020 |
| 10.0 | March 31, 2014 | 10.0.38 (January 31, 2019) | March 31, 2019 |
| 5.5 | April 11, 2012 | 5.5.68 (May 12, 2020) | April 11, 2020 |
| 5.3 | February 29, 2012 | 5.3.12 (January 30, 2013) | March 1, 2017 |
| 5.2 | November 10, 2010 | 5.2.14 (January 30, 2013) | November 10, 2015 |
| 5.1 | February 1, 2010 | 5.1.67 (January 30, 2013) | February 1, 2015 |



MariaDB  MySQL "" "" 2-6

WordPress

## MySQL Innovation Releases

| Version | Initial GA Release | Final Release | EOL Date |
|---|---|---|---|
| **9.3*** | April 15, 2025 | | July 2025 |
| 9.2 | January 21, 2025 | 9.2.0 (January 21, 2025) | April 15, 2025 |
| 9.1 | October 15, 2024 | 9.1.0 (October 15, 2024) | January 21, 2025 |
| 9.0 | July 1, 2024 | 9.0.1 (July 23, 2024) | October 15, 2024 |
| 8.3 | January 16, 2024 | 8.3.0 (January 16, 2024) | July 1, 2024 |
| 8.2 | October 25, 2023 | 8.2.0 (October 25, 2024) | January 16, 2024 |
| 8.1 | July 18, 2023 | 8.1.0 (July 18, 2023) | October 25, 2023 |

* Indicates the current innovation release.

## MariaDB Innovation/Short-Term Releases

The MariaDB project has decided to retire the concept of innovation releases, replacing them with the concept of "rolling GA releases". This will start with the 12.0 release.

Previously MariaDB referred to these releases as "short-term releases".

| Version | Release Type | Initial GA Release | Final Release | EOL Date |
|---|---|---|---|---|
| 11.5 | Innovation | August 15, 2024 | 11.5.2 (August 15, 2024) | November 21, 2024 |
| 11.3 | Innovation | February 19, 2024 | 11.3.2 (February 19, 2024) | May 16, 2024 |
| 11.2 | Short-term | November 21, 2023 | 11.2.6 (November 4, 2024) | November 21, 2024 |
| 11.1 | Short-term | August 21, 2023 | 11.1.6 (August 14, 2024) | August 21, 2024 |
| 11.0 | Short-term | June 7, 2023 | 11.0.6 (May 17, 2024) | June 6, 2024 |
| 10.10 | Short-term | November 17, 2022 | 10.10.7 (November 14, 2023) | November 17, 2023 |
| 10.9 | Short-term | August 22, 2022 | 10.9.8 (August 14, 2023) | August 22, 2023 |
| 10.8 | Short-term | May 21, 2022 | 10.8.8 (May 10, 2023) | May 20, 2023 |
| 10.7 | Short-term | February 14, 2022 | 10.7.8 (February 6, 2023) | February 9, 2023 |

# Other MySQL servers

While WordPress does not regularly test against these MySQL servers, they are known to perform well.

- Percona MySQL Server
- Amazon Aurora
- Amazon RDS for MariaDB
- Amazon RDS for MySQL
- Azure Database for MySQL
- Google Cloud MySQL
- DigitalOcean MySQL
- IBM Cloud Databases for MySQL
- MySQL HeatWave

Although WordPress may run on older versions, it is recommended to use these or newer ones for security and performance reasons.

实验性功能： SQLite（功能插件）

注意： WordPress 核心在生产环境中正式支持 MySQL/MariaDB。SQLite 是通过社区 SQLite 数据库集成
功能插件实现的 可选且实验性 功能。请将其视为主机的测试 /评估路径，而非全面替代方案。

征求主机反馈
如果您对此功能进行群体试用，请分享：流量概况、缓存情况、写入速率、数据库文件大小、错误日志以及任
何插件 /主题不兼容情况。如果未来提议合并到核心，这些反馈将非常有用。

如何查看当前版本？
若您使用的是 WordPress 5.2+ 版本，管理后台的 站点健康 功能（位于菜单栏 工具
项下）已内置相关信息查看工具。

若您使用较旧版本，可通过安装名为 Health Check & Troubleshooting的社区插件来激活 站点健康 功能（更多

插件使用帮助 ）。

[info]若您有意完善本手册，请访问 Github 手册仓库 ，或在官方 WordPress Slack的 #hosting 频道 留言。 [/info]

主机的可持续性
随着互联网的发展，托管网站对环境的影响也在增加。可持续的网络托管有助于减少碳排放并提高能源效率。以下是选择可持续托管提供商时需要考虑的一些关键因素，以及减少网络托管环境足迹的最佳实践。

能源效率
采用节能服务器与冷却系统，并整合利用率不足的服务器，对主机的可持续发展至关重要。过去十年间，托管服务的最大改进在于利用虚拟化技术减少物理服务器需求。使用固态硬盘替代机械硬盘也能提升能源效率。为进一步降低能耗，托管服务商可实施云托管和无服务器架构，实现按需动态分配资源。

可再生能源使用
托管服务提供商可以通过使用可再生能源为数据中心供电或运营来减少环境影响。投资太阳能、风能或水力发电等可再生能源可以显著改善环境影响。

绿色数据中心
获得绿色认证（如 LEED、 ISO 50001
）的数据中心针对能效进行了优化。提供商还可选择冰岛等气候凉爽地区的数据中心，以最大限度地减少冷却
需求。

区域数据中心
在可行的情况下，提供特定区域的托管选项以减少数据传输。引导客户理解本地数据中心的优势。添加 CDN 以高效应对国际流量。

员工与办公场所可持续性

可持续性举措不应局限于数据中心和服务器。若使用办公场所，请确保办公楼宇绿色节能。在办公室内推行回收计划，采用节能照明并鼓励无纸化办公。推广远程工作以减少日常通勤和办公能耗带来的整体碳足迹。

可持续数据管理
有效的数据管理实践有助于减少网络托管存储占用。采用增量备份可避免不必要的数据存储。定期归档或删除
未使用数据，并优化数据库和存储系统，能最大限度减少冗余。实施数据精简和图像压缩技术可改善数据管理
，降低单个网站的存储占用。

优化性能
优化性能对于降低服务器负载和能耗至关重要。托管服务商可实施服务器端缓存（如 Redis 或 Memcached ）来减少数据库查询和 CPU 使用率。可为静态和动态内容（如 HTML 和 PHP
）添加静态与动态缓存以降低服务器资源占用，从而减少 RAM 和 CPU 消耗。提供 WordPress
内置性能优化功能，包括代码压缩、数据清理和图像压缩。

安全至关重要
提高客户对安全影响一切的认识。黑客和恶意机器人可能使服务器过载，并产生不必要的能耗与数据影响。建
议至少使用 WP Scan 或 Pathstack 等工具，确保您能识别漏洞并及时应对。

使用长寿命硬件
应负责任地管理服务器和硬件：尽可能延长使用时间、尽可能高效利用、确保获得认证，并采购长寿命产品。
当硬件达到使用寿命终点时，务必进行回收、循环利用和升级再造。

减少未使用资源的数量
测试和预发布环境固然重要且实用 ——
前提是您的客户确实在使用它们。请确保客户能够轻松创建和销毁这些环境，让他们放心地不必长期保留。每个未使用的环境仍在消耗存储和计算空间，即使应用程序没有流量。

对碳与环境项目的再投资

托管公司有多种方式改善其环境影响。考虑到首要任务应是避免和减少影响，像 https://onetreeplanted.org 和 https://theoceancleanup.com这样的慈善机构和企业提供了很好的补偿途径。

教育您的客户
向客户普及可持续网络托管的重要性，以及他们可以采取哪些措施来改善自身网站的碳足迹。提供教程、网络
研讨会和互动工具等教育资源，用于测量网站足迹并提供提升网站效率的建议。

跟踪进展
务必检查初始基准，以便随时查看进展并进行比较。

持续改进与审计
可持续性是一项持续承诺。托管服务提供商应定期审计其能源、水资源和废弃物管理实践，并持续关注最新技
术以提升环境绩效。

可再生能源证书（ RECs）
托管服务提供商还可以通过购买可再生能源证书（ RECs
）来补充其可持续发展努力，这些证书通过投资可再生能源来抵消能源消耗。例如： 可再生能源证书（ RECs）

生态友好的插件与主题开发
鼓励插件和主题开发者采用可持续的编码实践。轻量级代码、最少的外部依赖和高效的数据库查询，能够降低
运行在共享基础设施上的网站的整体能耗。 WordPress 特定的可持续性指南可帮助开发者构建时兼顾效率。

托管项目
托管团队的项目可能多种多样，并且能够与其他团队的项目交叉协作。
目前可供贡献的项目列表包括：

WordPress 托管团队手册 ，专注于文档编写。
自动化托管测试 ，专注于 PHP 兼容性。
高级管理手册 ，专注于文档编写，与 文档团队 合作完成。

WordPress 托管团队手册

内容简介

本手册包含关于托管团队的信息，以及运行 WordPress 的托管建议。

这些建议由团队共同制定，并作为 WordPress 中站点健康建议 的基础。它们既可作为学习托管 WordPress 的参考资料，也可作为帮助 WordPress 与托管服务商共同改进的途径。

技能要求

技术文档撰写
系统管理
这些技能并非必需，但会有所帮助。

参与方式

本手册正处于审核与改进阶段。您可以通过 GitHub查看进度并参与贡献。

首先请 访问代码仓库页面 ，所有相关信息均在此处。建议先 查看议题列表，确认您的建议是否已被收录或待处理；若尚未收录，请 创建新议题 。

提交议题后，我们将共同讨论建议的修改内容。

提交拉取请求（ PR）后需经过审核流程。默认情况下，提交的 PR
需要托管团队两名成员审核批准。若涉及修正内容或内部操作元素的 PR 由托管团队成员创建，则仅需一次审核即可。

工具

参与本项目需要准备：

GitHub 账户
文本 / Markdown 编辑器

帮助

如需协助，请在 #hosting频道提问或联系托管团队代表。

自动化托管测试

概述

Runner 代码库 (PHPUnit 测试运行器 )包含在主机上运行的托管测试部分，而 Reporter 代码库 (PHPUnit 测试报告器 )包含在 WordPress.org 上运行以接收和显示测试结果的插件。

技能要求

PHP 编程
系统管理

这些技能并非必需，但会有所帮助。

参与方式

您可以在 PHPUnit 测试运行器（ 问题列表 ）和 PHPUnit 测试报告器（ 问题列表）中提出改进建议或解决现有问题。

工具

参与本项目需要准备：
GitHub 账户
文本 / PHP 编辑器

帮助

如需帮助，请在 #hosting频道提问或联系托管团队代表。

高级管理员手册

内容定位

本文档旨在作为现有终端用户文档（ 用户中心 / 支持
）的进阶补充。其目标是：用户文档保持简洁，不包含高级技术内容；而开发者文档（ 开发中心
）则明确区分基础开发、高级开发及系统配置。

技能要求

技术文档撰写
这些技能并非必需，但会有所帮助。

参与方式

虽然我们仍处于起步阶段，但如果您知道其他可能适合纳入本手册的 WordPress 内容，请 提交议题并与文档团队讨论。请使用标签 "advanced administration"。

若您希望帮助尽快发布本手册， 请认领议题并参与贡献
。如有任何疑问、建议或改进意见，您可以直接在网站上留言。

工具

参与本项目需要准备：

GitHub 账户
文本 / Markdown 编辑器

帮助

如需帮助，请在 #docs或 #hosting频道提问，或联系托管团队代表。

团队代表
WordPress
项目拥有来自全球的贡献者，他们承担着多种角色，包括编码、支持、翻译、活动组织、文档撰写以及插件和
主题审核。这些多样化的角色对项目的持续成功和发展做出了重要贡献。
这些参与者被组织成特定的项目团队，每个团队在 make.wordpress.org
网络上都有一个专门的空间，用于讨论和分享他们贡献的最新动态。
每个团队必须至少选出两名团队代表。这些代表充当各自团队与更广泛的 WordPress 项目之间的联络人，确保问题得到适当的表达和解决，并促进团队之间的协作。

更多关于 团队代表：头衔、职责 ...的信息

托管团队

托管团队代表

托管团队（ 2025年）的代表是：

@crixu
@zunaid321

项目负责人

文档项目（托管手册与高级管理手册）由以下人员领导：
默认为团队代表 /管理员。团队集体管理这些项目
PHPUnit 测试项目（ PHPUnit 测试运行器与 PHPUnit 测试报告）由以下人员领导：
默认为团队代表 /管理员。团队集体管理这些项目

Hosting Team Admins

@zunaid321 (Team Rep 2025 - present)

@jessibelle (Team Rep 2023)
@crixu (Team Rep 2021-present)
@amykamala (Team Rep 2019-2023)
@chaion07 (Team admin since 2021)

Past Hosting Team Reps and Admins

@javiercasares (Team Rep 2021 - 2024)
@jadonn (Team Rep 2018 - 2022)
@pfefferle
@kirasong (Started the team in 2017, Team Rep 2017 - 2021)
@danielbachhuber (Started the team in 2017)

托管团队组织架构

团队代表

在广阔的 WordPress
生态系统中，不同团队协同工作以确保平台的持续发展与成功。每个团队至少设有一名代表，通常称为 “ 团队代表 ”。对于托管团队而言，团队代表扮演着至关重要的角色。

托管团队代表的职责

托管团队的团队代表，在更广泛的 WordPress 社区中担任托管团队的代言人和大使。其职责是多方面的：

1. 代表职能： 团队代表作为托管团队的主要联络人，向更广泛的 WordPress
社区传达团队的利益、挑战和成就。
2. 信息交流： 团队代表充当沟通的桥梁，促进与其他团队代表及 WordPress 项目领导层之间关键信息（如项目更新和战略决策）的交流。
3. 会议协调：
团队代表的职责包括确保每周团队会议按计划进行。其责任涉及鼓励团队成员参与议程制定、收集讨
论议题并主持会议。重要的是，会议协调者的角色不仅限于团队代表；任何团队成员都有资格主持会
议。

团队代表职责

职责轮换：
托管团队确保至少有两名团队代表。理想情况下，一名代表在任期的前半段承担主要职责，而另一名
代表则担任后备角色。在任期的后半段，角色互换。这种结构确保了连续性，因为后备代表在承担主

- Make/Updates
  3
- 

## 

make.wordpress.org/hosting "" Slack

- 
- make.wordpress.org/hosting

make.wordpress.org/hosting

# 

WordPress WordPress

## 

## 

1. 
   - 
     - 

高级管理手册
所需技能：
精通 GitHub 仓库的管理与组织。
最好具备技术写作经验，因为这类项目的核心是产出清晰、简洁且信息丰富的文档。
2. 软件项目：
此类项目包括：
PHPUnit 测试运行器
PHPUnit 测试报告
所需技能：
熟练管理 GitHub 仓库。
深入理解 PHP 单元测试及 PHP 语言。
拥有或至少熟悉 WordPress 插件的创建与编程经验者优先。

托管团队入职 /离职
站点
访问 Make Hosting
访问 Make (议程)
访问 Make Updates
访问 托管贡献者个人资料徽章
访问 托管团队个人资料徽章
沟通
托管邮箱 ( hosting -at- wordpress.org ) 使用 HelpScout Slack #hosting频道的 "/here" 访问权限
加入 #team-reps Slack 频道
加入 #props Slack 频道
加入 #github-teams-collab Slack 频道
加入 #contributor-day Slack 频道， 推荐
加入 #core-php Slack 频道， 推荐
加入 #core-site-health Slack 频道， 推荐
加入 #core-test Slack 频道， 推荐
加入 #core-upgrade-install Slack 频道， 推荐
加入 #tide Slack 频道， 推荐
加入 #wcasia Slack 频道， 推荐
加入 #wceu Slack 频道， 推荐
加入 #wcus Slack 频道， 推荐
GitHub 团队 注意：必须是 GitHub WordPress 组织 的成员才能查看。

托管团队
托管手册团队
托管文档团队
托管测试团队
GitHub 仓库
托管手册 Github 仓库
高级管理手册 Github 仓库
WordPress PHPUnit 测试套件 Github 仓库 (又名 test-runner) WordPress PHPUnit 测试报告器 Github 仓库 (又名 test-reporter)

# Monthly Hosting Team Updates

The Hosting Team is publishing monthly updates to make.wordpress.org/hosting and https://make.wordpress.org/updates/ with the team activity currently measured in:

- Activity on the Hosting Tests
- GitHub Activities on Hosting Team maintained Repos

As an example you can view the monthly update for July 2025, which was the first one published.

In order to publish those reports you need to have Administrator access to make.wordpress.org/hosting and Editor access to make.wordpress.org/updates.

# How to generate the reports

## Activity on Hosting Tests

For Hosting Tests the team reports the newly shared reports by hosts for that month. This data can be gathered from the WordPress Backend of make.wordpress.org/hosting. The results page can be accessed directly https://make.wordpress.org/hosting/wp-admin/edit.php?s&post_status=all&post_type=result&action=-1&m=202507, with having the last number being the year and month you want to get results for. While on this page you can see the total amount of all published "posts" as number of all test results ever shared.

Last but not least the number of test reporters and recent activity can be retrieved from the WordPress Backend too. The number of recent active reporters can be counted on https://make.wordpress.org/hosting/test-results/.

## GitHub Report

We prepared a script that collects stats from GitHub through the API and outputs the table and the contributors. Right now the script is located in this repo.

Follow the readme, create your own .env file and run the script. Once it is finished it will output all the required stats in the `report` folder.

# Publishing on make.wordpress.org

Once the data is collected go ahead and copy the report of last month from the WordPress Backend. Update all the **data and dates** and ask the team for a review before publishing.

The Crosspost to `/updates` won't include the content of the post. In order to fix this you need to login to make.wordpress.org/updates and edit the post to add the content.

测试
为提升与 WordPress 的兼容性， WordPress 托管团队为托管公司提供工具，使其能在自身基础设施上运行
WordPress 自动化测试以改善兼容性。测试结果可发布至 托管测试结果信息页面 ，向 WordPress 社区展示其兼容状态。

该方案包含两个工具：运行器负责在主机上执行核心 PHPUnit 测试，并可选择 将信息发送至结果页面
；报告器则是运行于 托管页面 并展示测试结果的插件。

这是什么

运行器

托管公司可能托管着数百万个 WordPress 网站，因此确保其配置尽可能与软件兼容至关重要。

为了验证这种兼容性， WordPress 社区提供了一系列 PHPUnit测试，用于检查 WordPress 在任何环境中的运行情况。

报告器
Runner
测试会生成一份报告，其中包含与关联于网络主机的机器人用户相关的测试结果，这些结果会被捕获并显示在
主机测试结果 页面上。

试用 PHPUnit 测试运行器

什么是 phpunit-test-runner

phpunit-test-runner是一个旨在让托管公司更轻松地运行 WordPress 项目自动化测试的工具。

关于此工具有 完整的文档 。此外，如果您愿意，您可以让您的测试结果出现在 WordPress 的托管测试结果页面上。

该工具可以手动运行，也可以通过像 Travis 这样的自动化系统运行。本文档的目的是展示如何手动运行测试，以说明其工作原理。

要求

要使用 Runner，需要满足以下条件：

一台具有常规配置的服务器 /托管（基础设施）
一个可供测试的数据库（该数据库将被多次创建和销毁）
NodeJS 20.x

NodeJS 安装

如果您使用 Debian / Ubuntu 系统，请使用以下命令安装或更新 NodeJS：

curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash - sudo apt -y install nodejs
node -v

如果您使用 RHEL / CentOS 系统，请使用以下命令安装或更新 NodeJS：

curl -fsSL https://rpm.nodesource.com/setup_20.x | sudo -E bash - sudo yum install -y nodejs
node -v

安装运行器

首先，下载软件。此示例使用 /home/wptestrunner/ 文件夹，但请根据环境设置为最佳路径。

cd /home/wptestrunner/
git clone https://github.com/WordPress/phpunit-test-runner.git cd phpunit-test-runner/

下一步是配置环境。为此，复制示例文件并进行配置。

cp .env.default .env
vim .env

内容（摘要形式）可以如下所示：

```
# 文件在交付到环境之前可以准备的目录路径。
export WPT_PREPARE_DIR=/home/wptestrunner/wordpress

# 可以复制 `wordpress-develop` 仓库并从中运行测试的目录路径。在同一环境中运行测试时，将 WPT_TEST_DIR 设置为 WPT_PREPARE_DIR
export WPT_TEST_DIR=$WPT_PREPARE_DIR

# 用于向报告服务进行身份验证的 API 密钥，格式为 'username:password'。请查看"创建你的机器人"部分以获取身份验证信息。
export WPT_REPORT_API_KEY=
#export WPT_REPORT_API_KEY=examplehostingcompanybot:XXXX XXXX XXXX XXXX

# （可选）定义替代的报告 URL
export WPT_REPORT_URL=

# 可写入和重置的数据库凭据。警告！！！此数据库将在测试之间被销毁。仅使用安全的数据库凭据。
export WPT_DB_NAME=wordpress
export WPT_DB_USER=wordpress
export WPT_DB_PASSWORD=__PASSWORD__
export WPT_DB_HOST=localhost

# （可选）设置自定义表前缀以允许对同一数据库进行并发操作。
export WPT_TABLE_PREFIX=${WPT_TABLE_PREFIX-wptests_}

# （可选）定义要调用的 PHP 可执行文件
export WPT_PHP_EXECUTABLE=${WPT_PHP_EXECUTABLE-php}

# （可选）定义 PHPUnit 命令执行调用。如果由于某些原因无法直接调用 `php phpunit.phar`，请使用此选项。
export WPT_PHPUNIT_CMD=

# （可选）定义删除测试目录的命令执行。如果由于某些原因无法直接调用 `rm -r`，请使用此选项。
export WPT_RM_TEST_DIR_CMD=

# SSH 连接字符串（也可以是别名）。如果测试计划在同一环境中运行，请留空。
export WPT_SSH_CONNECT=

# 传递给 SSH 连接的任何选项。默认为 '-o StrictHostKeyChecking=no'
export WPT_SSH_OPTIONS=

# SSH 私钥，base64 编码。
export WPT_SSH_PRIVATE_KEY_BASE64=
```

```
# 输出日志记录。使用 'verbose' 增加详细程度
export WPT_DEBUG=
```

配置用于下载 WordPress 软件和进行数据库访问以准备测试的文件夹。

## 环境准备

在执行首次测试前，让我们更新所有组件。如果您希望保持环境最新，可以在每次测试前运行此流程，但这更多取决于是否处于生产环境。

```
cd /home/wptestrunner/phpunit-test-runner/
git pull
source .env
```

## 准备测试

现在环境已就绪，运行测试准备工作。

```
php prepare.php
```

系统将运行一系列漫长的安装、配置和不同组件的编译过程来准备测试。如果出现警告信息，不必过于担心，这完全正常。在流程结束时，如果缺少某些必要组件，系统会发出提示。如果一切顺利，最终应该会看到类似这样的输出：

```
完成。
正在替换 wp-tests-config.php 中的变量
成功：环境准备就绪。
```

环境准备完成后，下一步就是首次运行测试。

[info]每次运行测试都必须执行全部 4 个步骤。这包括每次都要通过运行 `prepare.php` 来准备环境，即使环境配置没有发生任何更改。[/info]

## 运行测试

现在环境已准备就绪，让我们运行测试。为此，请执行将执行测试的文件。

```
php test.php
```

这些符号代表什么？

. → 每个点表示测试已正确通过。

S → 表示测试已被跳过。这通常是因为这些测试仅在特定配置下有效。

F → 表示测试失败。失败原因的信息会在末尾显示。

E → 表示测试因 PHP 错误而失败，可能是错误、警告或通知。

I → 表示测试被标记为未完成。

如果遵循这些步骤，一切应该都能完美运行，并且不会出现任何错误。如果在遵循这些步骤后遇到错误，可能
是由于某些缺失的调整或 PHP
扩展等原因。我们建议您调整配置，直到其正常工作。毕竟，此工具旨在帮助您优化该基础设施中 WordPress 的最佳配置。

创建报告

即使测试失败，也会生成报告。第一份报告显示环境信息。其中最重要的元素包括 WordPress
中常用的扩展以及一些通常很有用的工具。

cat /home/wptestrunner/wordpress/env.json

该文件内容大致如下：

{
"php_version": "7.4.5",
"php_modules": {
"bcmath": false,
"curl": "7.4.5",
"filter": "7.4.5",
"gd": false,
"libsodium": false,
"mcrypt": false,
"mod_xml": false,
"mysqli": "7.4.5",
"imagick": false,
"pcre": "7.4.5",
"xml": "7.4.5",
"xmlreader": "7.4.5",
"zlib": "7.4.5"
},
"system_utils": {

```
    "curl": "7.58.0 (x86_64-pc-linux-gnu) libcurl\/7.58.0 OpenSSL\/1.1.1g zlib\/1.2.11
libidn2\/2.3.0 libpsl\/0.19.1 (+libidn2\/2.0.4) nghttp2\/1.30.0 librtmp\/2.3",
    "ghostscript": "",
    "imagemagick": false,
    "openssl": "1.1.1g 21 Apr 2020"
  },
  "mysql_version": "mysql Ver 15.1 Distrib 10.4.12-MariaDB, for debian-linux-gnu (x86_64)
using readline 5.2",
  "os_name": "Linux",
  "os_version": "4.15.0-20-generic"
}
```


除了这份报告，还会提供一个包含测试过程中所有信息的最终文件。该文件包含所有执行的测试（超过 10,000 项），提供每项测试的执行时间以及可能出现的任何问题。

```
cat /home/wptestrunner/wordpress/junit.xml
```

此时，如果需要，我们可以通过将报告发送到 WordPress.org 来生成报告。即使您尚未包含 WordPress 用户（创建方法见下文），仍然可以运行此文件。

```
php report.php
```

## 清理环境以便进行其他测试

测试运行成功后，剩下的就是删除所有已创建的文件，以便重新开始。为此，请执行以下命令：

```
php cleanup.php
```

## 自动化执行

完成首次手动测试后，请将所有步骤编写为自动化脚本，因为每次测试执行都需要按顺序完成这些步骤。

该脚本应在 WordPress 主分支发生变更 /提交时运行。许多托管服务商会通过 cron 定时任务（每隔数小时 / 数天）或在代码变更时运行此脚本，以执行相应的检查。

## 优化配置

请勿忘记，此工具旨在验证环境和基础设施是否为 WordPress 运行的最佳配置。因此，遵循示例，您可以进行多项改进，例如安装 `bcmath`、`gd`、`libsodium`、`mcrypt`、`mod_xml`

和 imagick 等扩展，或 ghostscript 和 imagemagick 等实用工具。

目标何在？实现零错误，并获得完美配置的绿灯。

[alert]某些测试可能会被跳过，或者可能存在 PHPUnit 认为 “有风险 ”的测试。 [/alert]

如何报告：为 WordPress.org

创建你的机器人

如果你 /你的公司希望 测试结果能显示在 WordPress.org 页面上 ，请为此创建一个用户。

首先要做的是 在 WordPress.org 上创建一个用户 。例如，如果你的公司名为 ExampleHostingCompany, Inc ，可以将你的用户名命名为类似 examplehostingcompanybot
。请注意，应经常检查关联的电子邮件账户，因为会收到关于测试可能运行的邮件。

在测试页面创建一个议题 ，请求将该机器人作为 测试报告者
包含在结果页面中，并注明你为该用户使用的电子邮件账户。

[tip]此用户的头像必须是贵公司的徽标，且名称和 URL 必须明确显示是哪家公司。 [/tip]

[info]
托管团队中的成员将审核该请求，并为你添加用户，或要求提供更多信息。团队会尽快回复，但由于此步骤是
手动的，请耐心等待。 [/info]

一旦用户在系统中创建完成，你将通过电子邮件收到加入邀请。然后，你可以登录
make.wordpress.org/hosting，并在 用户 -> 你的个人资料 中创建一个应用程序密码。

为了使报告正常工作，请将机器人的用户名以及应用程序密码放在 .env 文件中，内容大致如下： export WPT_REPORT_API_KEY='examplehostingcompanybot:ABCD 1234 abcd 4567 EFGH efgh' 。

[info]如果你有兴趣改进本手册，请查看 Github 手册仓库 ，或在官方 WordPress Slack的 #hosting 频道
中留言。 [/info]

# 升级 WordPress

如果 WordPress 能够持续维护更新，并根据每个主要版本的 当前要求 进行调整，那么升级过程应该很简单。

旧版本会怎样?

出于善意， WordPress 项目会为从 4.7 版到最新版本提供安全支持。然而，向后移植补丁并非总是可行，因此，除最新版本外的其他 WordPress 版本 不被官方支持 。

每个版本都有其支持的数据库版本、 PHP 版本以及一系列需要随时间更新的兼容性要求。

如果不更新会怎样?

本页提供了一些关于如何升级的思路和建议。在许多情况下，可能无法升级操作系统或引入新的 PHP 版本，因此您可能需要将 WordPress 迁移到已更新的托管环境。

本文档会做出一些假设，因为每个案例都不同。在进行任何升级之前，请备份所有网站数据: WordPress 文件、数据库、证书以及操作系统 /服务配置。

重要提示: 这是一个非常手动的过程，而非批量操作。升级后很可能会有损失，并且需要进行一些后续更新。

[tip]如果只指明了主要版本号 (X.X)，请使用其可用的最新次要版本 (X.X.x)。[/tip]

# 使用 WP-CLI 进行升级

如果您的托管环境中已安装 WP-CLI ，升级过程的某些环节可以得到简化。请务必在运行任何命令前，对文件和数据库进行完整备份。

有关 `wp core` 命令和选项的最新完整列表，请参阅官方 WP-CLI 文档:

https://developer.wordpress.org/cli/commands/core/

# 从 WordPress 0.7 - 3.6 升级（通过迁移）

目标:

- WordPress: 升级到 WordPress 6.2
- PHP: 升级到 PHP 7.4
- SQL: 升级到 MySQL 8.0 / MariaDB 10.11

损失:

- 内容: 无
- 插件: 全部
- 主题: 全部

这些是最旧版本的 WordPress，并且已经多年不受支持。通常，必须假设会有一些损失，虽然不是内容损失，但可能是主题和插件的某些功能损失。

考虑到目标是保留内容并假设其余元素会损失，有以下步骤。

与任何升级一样，首先要做的是备份。从 WP < 3.6 升级的最佳方法是执行内容迁移。

1. 创建一个全新的 WordPress，不包含数据库。
2. 将旧 WordPress 文件中的 "/wp-content/uploads/" 内容复制到新站点。
3. 使用旧数据库信息创建一个新数据库。最佳方法是使用 "mysqldump"。
4. 使用所有新数据配置 wp-config.php。
5. 访问 "/wp-admin/" 页面，并按照升级流程操作。

这样，WordPress 将能够维护和更新数据库中的内容，并能在更新版本的 WordPress 中使用这些内容。

现在应该可以使用一个带有默认主题和所有内容的 WordPress。

在恢复数据库时，字符编码通常会出现技术问题。备份数据可能不是以 UTF-8 编码，而是可能采用 ISO 或 ASCII 等 "已弃用 "的格式。确保在恢复数据库时正确更新字符编码！更多关于在 WordPress 数据库中转换字符集的信息可以在这里找到。

---

# 从 WordPress 3.7 - 4.0 升级

目标:

- WordPress: 升级到 WordPress 4.1
- PHP: 升级到 PHP 5.6.20+
- SQL: 升级到 MySQL 5.6 / MariaDB 10.0

可能的损失:

- 内容: 无

插件：很可能有
主题：很可能有

WordPress 版本 <= 4.0 兼容的 PHP 版本如今已很难获得。这些版本可能从 PHP 5.2（甚至更早）到 PHP 5.5。

WP-CLI 在 PHP 5.6.20 以下无法工作，因此此更新必须手动完成。

请遵循以下步骤：

备份所有内容
移除未使用的主题
启用 Twenty Ten 主题
删除已停用的插件
停用所有活跃的插件

然后用 WordPress 4.1 覆盖 WordPress 核心文件。

将 PHP 升级到 5.6.20+，并将 SQL 升级到 MySQL 5.6 / MariaDB 10.0。

访问 /wp-admin/ 并完成升级。

接下来进行 WordPress 4.9 的升级。

从 WordPress 4.1 - 4.8 升级
目标：

WordPress：升级到 WordPress 4.9
PHP：升级到 PHP 7.2
SQL：保持或升级到 MySQL 5.6 / MariaDB 10.0

损失：

内容：无
插件：很可能有
主题：很可能有

再次执行清理流程：

移除未使用的主题
激活 Twenty Ten 主题
删除未使用的插件

□ WordPress 4.9 □□ WordPress □□□□□
□ PHP □□□ 7.2□

□□□□□ WordPress 5.3 □□□□

---

# WordPress 4.9 - 5.2

Goals:

- WordPress: upgrade to WordPress 5.3
- PHP: upgrade to PHP 7.4
- SQL: upgrade to MySQL 8.0 / MariaDB 10.3

Losses:

- Content: none
- Plugins: probably no
- Themes: probably no

Major notes:

- WordPress 4.9 was the last Classic Editor version.
- WordPress 5.0+ fully supports Classic Editor content.

Follow the same cleanup steps:

- Activate Twenty Ten
- Deactivate plugins
- Replace core with WordPress 5.3

Then upgrade PHP to 7.4 and SQL to MySQL 8.0 / MariaDB 10.3.

---

# WordPress 5.3 - 6.2

□□□

- WordPress□□□□ WordPress 6.2
- PHP□□□□ PHP 7.4
- SQL□□□□ MySQL 8.0 LTS / MariaDB 10.11 LTS

□□□

内容：无
插件：无
主题：无

正常升级。一切应能正常工作。

WordPress 6.3 - 6.8

目标：

WordPress：升级至 WordPress 6.8
PHP：升级至 PHP 8.1
SQL：升级至 MySQL 8.0 LTS / MariaDB 10.11 LTS

损失：

内容：无
插件：无
主题：无

PHP 变更：

WP 6.3 弃用了 PHP 5.6
WP 6.6 弃用了 PHP 7.0 和 7.1 WP 6.6 要求 PHP 7.2.24 或更高版本

正常升级即可。一切应能正常运行。