Shell 助手

目录
你应该知道
搜索你的 bash 历史记录
组合 WP-CLI 命令
定义别名，常用命令的快捷宏
保存命令输出
代码片段
查看插件更新日志
以当前目录的 ABSPATH 和当前目录所有者身份启动 wp-cli
使用 WP-CLI 安装和配置 WordPress
列出所有文章中的图片 URL
从文件创建页面并用文件名标记
Change to a certain WordPress installation's directory from a menu
从 CSV 中丢弃标题行
Sort plugins or themes by certain column(s)

随着你对 WP-CLI 的深入使用，你会发现掌握一些命令行知识能极大提升工作效率。以下是我们最推荐的几款
Shell 辅助工具。

你应该知道

搜索你的 bash 历史记录
你知道吗？你在 shell 中运行的每个命令都会被保存到历史记录中。使用 CTRL + R 来搜索你的历史记录：

$ wp core download --version=nightly --force
bck-i-search: wp

当出现 'bck-i-search' 时，你的按键将用于搜索保存在 bash 历史记录中的命令。按 return 键来运行当前选中的命令。

或者，另一种搜索历史记录的方法是对 `history` 命令的输出进行 grep 过滤，例如：

```
$ history | grep wp
```

在该列表中找到的任何命令都可以通过感叹号加数字重新执行，例如，如果输出显示你想要的命令是 #218，你只需输入 `!218` 即可。

组合 WP-CLI 命令

在许多情况下，能够将一个命令的结果传递给另一个命令会非常强大。可组合性是 WP-CLI 的核心理念，组合命令有两种常见方法。

命令替换 将一个命令的输出传递给另一个命令，不对输出进行任何转换。

`wp post list` 仅列出文章；它不对文章执行任何操作。在此示例中，该命令以空格分隔值的形式列出页面 ID。

```
$ wp post list --post_type='page' --format=ids
1164 1186
```

将 `wp post list` 与 `wp post delete` 结合使用，可以轻松删除所有文章。在此示例中，`$()` 允许我们将空格分隔的页面 ID 传递给 `wp post delete`。

```
$ wp post delete $(wp post list --post_type='page' --format=ids)
Success: Trashed post 1164.
Success: Trashed post 1186.
```

如果需要更多灵活性，*xargs* 允许将一个命令的输出传递给另一个命令，同时对输出进行轻微转换。

您可能希望将所有编辑者权限分配给作者角色。但是，`wp cap list` 以换行符分隔权限，而 `wp cap add` 只接受空格分隔的权限。这时可以使用 `xargs`，其默认行为是将换行输出拆分为空格分隔的列表。请注意 `|` shell 运算符，它将 `wp cap list` 的结果传递给 `xargs`。没有 `|`，您会看到 WP-CLI 错误。

```
$ wp cap list 'editor' | xargs wp cap add 'author'
Success: Added 24 capabilities to 'author' role.
```

`wp user generate` 仅生成用户；它不执行补充操作。在此示例中，`wp user generate` 将用户 ID 传递给 `xargs`，后者将空格分隔的 ID 拆分为列表，并为每个 ID 调用 `wp user meta add`。

```
$ wp user generate --count=5 --format=ids | xargs -0 -d ' ' -I % wp user meta add % foo bar
Success: Added custom field.
Success: Added custom field.
```

```
Success: Added custom field.
Success: Added custom field.
Success: Added custom field.
```

□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□ `check-all` □□□□□□□□□□□□□□□□□□□□□□□□□□□□ `alias` □□□□ shell □□□□□□ `check-all` □□□□□□□□□□□□□□□ `~/.bashrc` □ `~/.zshrc` □□□□□□□□□□

```
$ alias check-all='wp core check-update && wp plugin list --update=available && wp theme list
--update=available'
$ check-all
+-----------------+-------------+-----------------------------------------------------------+
| version         | update_type | package_url                                               |
+-----------------+-------------+-----------------------------------------------------------+
| 4.7-beta4-39322 | minor       | https://wordpress.org/nightly-builds/wordpress-latest.zip |
+-----------------+-------------+-----------------------------------------------------------+
+-------------------------+----------+-----------+--------------+
| name                    | status   | update    | version      |
+-------------------------+----------+-----------+--------------+
| akismet                 | inactive | available | 3.1.8        |
| co-authors-plus         | inactive | available | 3.1.1        |
| wp-redis                | inactive | available | 0.2.2        |
| rest-api                | active   | available | 2.0-beta13.1 |
| wp-api-oauth1           | inactive | available | 0.2          |
+-------------------------+----------+-----------+--------------+
+----------------------+----------+-----------+---------+
| name                 | status   | update    | version |
+----------------------+----------+-----------+---------+
| p2                   | inactive | available | 1.4.2   |
+----------------------+----------+-----------+---------+
```

□□□□□

WP-CLI □□□□□□□□□□□□□□□□ `STDOUT` □ `STDERR` □□□□□□□□□□□□□□□□□□□□□□ WP-CLI □□□

仅使用 `>` 会将命令的 STDOUT 捕获到文件中。

```
$ wp import wordpress.wxr --authors=create > import.log
```

使用 `2>&1` 将 STDERR 重定向到 STDOUT，然后将 STDOUT 捕获到日志文件中。

```
$ wp import wordpress.wxr --authors=create > import.log 2>&1
```

当您将输出捕获到文件时，您在当前终端会话中将看不到输出。但是，您可以打开第二个终端会话并运行 `tail -f <file>` 来实时查看添加到文件中的输出。

或者，您可以使用 `tee` 命令，它同时写入标准输出和文件。您只需将输出通过管道传递给该命令并指定文件名即可。

```
$ wp import wordpress.wxr --authors=create | tee import.log
```

这将在当前终端屏幕上显示输出，同时将其写入日志文件。

请注意，如果文件已存在，它将被覆盖。如果您想将输出追加到文件中，请使用 `-a` 选项。

```
$ wp import wordpress.wxr --authors=create | tee -a import.log
```

代码片段

你是 bash 或 zsh 的专家吗？在这里分享你的技巧。

查看插件更新日志

需要 `elinks` 来浏览 HTML。

```
wget -qO- http://api.wordpress.org/plugins/info/1.0/akismet|php -r
'$seri=unserialize(stream_get_contents(STDIN)); echo $seri->sections["changelog"];'|elinks -
force-html
```

解释

- 设置 wget 静默模式并查询 WordPress.org 插件 API
- 使用 php 反序列化，`stream_get_contents(STDIN)` 意为 “从 标准输入 获取全部内容 ”
- 输出 API 回复中的更新日志部分
- 启动 elinks（一个控制台浏览器）来查看更新日志

以当前目录的 ABSPATH 和当前目录所有者身份启动 wp-cli

```
#!/bin/bash

sudo -u $(stat . -c %U) -- wp --path="$PWD" "$@"
```

说明

`stat` 命令返回当前目录（ WordPress 根目录）的所有者。

使用 WP-CLI 安装和配置 WordPress

```
wp_install ()
{
    wp core download --path=$1;
    cd $1;
    read -p 'name the database:' dbname;
    wp config create --dbname=$dbname --dbuser=root --dbpass=awoods --dbhost=localhost;
    wp db create;
    wp core install --prompt
}

$ source ~/.bashrc
$ wp_install new-site
```

说明

将此函数添加到您的 ~/.bashrc 文件中，然后重新加载您的 shell（或打开一个新的 shell）。您需要将这些数据库凭据替换为您自己的。当您需要创建一个新的 WordPress 站点时，调用此函数并指定您希望创建站点的目录名称。这模拟了基于 Web 的安装过程。

列出所有文章中的图片 URL

```
wp post list --field=ID|xargs -I % wp post get % --field=post_content|sed -ne
's;.*\(https\?\S\+\(jpe\?g\|png\|gif\)\).*;\1;gp'
```

说明

- □□□□□□ ID
- □□□□□□□□□ (xargs)
- □□□□□ URL (sed)

□□□□□□□□□□□□□

```
wp post create new_page.html --post_type=page --post_title="New Page" --porcelain | xargs -I %
wp post meta add % imported_from new_page.html
```

□□

- □□□□□□□ --porcelain □□□□□□□□□□ ID□
- □□ xargs □□□□□□□□□□ "-I %" □□□□□ ID □□□□□□

Change to a certain WordPress installation's directory from a menu

```
#!/bin/bash

WP_TOP_PATH="/home/"
MENU_TEXT="Choose an installation"
GAUGE_TEXT="Searching for WordPress"

declare -a MENU
WPS="$(wp --allow-root find "$WP_TOP_PATH" --field=version_path)"
WP_TOTAL="$(wc -l <<< "$WPS")"
WP_COUNT="0"

while read -r WP; do
    WP_LOCAL="${WP%wp-includes/version.php}"

    NAME="$(cd "$WP_LOCAL"; sudo -u "$(stat . -c %U)" -- wp --no-debug --quiet option get
blogname)"
    if [ -z "$NAME" ]; then
        NAME="(unknown)"
    fi
    MENU+=( "$WP_LOCAL" "$NAME" )
```

```
    echo "$((++WP_COUNT * 100 / WP_TOTAL))".
done <<< "$WPS" > >(whiptail --gauge "$GAUGE_TEXT" 7 74 0)

WP_LOCAL="$(whiptail --title "WordPress" --menu "$MENU_TEXT"  $((${#MENU[*]} / 2 + 7)) 74 10
"${MENU[@]}" 3>&1 1>&2 2>&3)"

if [ $? -ne 0 ] || [ ! -d "$WP_LOCAL" ]; then
    echo "Cannot find '${WP_LOCAL}'" 1>&2
    exit 100
fi

echo "cd ${WP_LOCAL}"
```

Explanation

- Needs `wp-cli/find-command` and `whiptail`
- Find all WordPress installations below `$WP_TOP_PATH` - must be run as root
- Display a progress bar while getting `blogname` of each installation
- Choose one installation from a nice menu and display `cd` command for it

□ CSV □□□□□

□□□□□□ `--format=csv` □□□□□□□□□

```
$ wp user list --format=csv
ID,user_login,display_name,user_email,user_registered,roles
1,daniel,daniel,daniel@handbuilt.co,"2022-12-21 23:05:16",administrator
```

□□□□□□□□□□ `tail` □□□□□□□□

```
$ wp user list --format=csv | tail -n +2
1,daniel,daniel,daniel@handbuilt.co,"2022-12-21 23:05:16",administrator
```

Sort plugins or themes by certain column(s)

- **Challenge**: wp-cli's `<plugin|theme> list` commands have no option by which column(s) to sort.
- **Solution**: You can output as `--format=csv` and then simply pipe into a CLI app which has sorting functions built-in.

- E.g. if you want to sort by status and then by name with miller, short form `mlr`:

```
$ wp plugin list --format=csv | mlr --icsv --opprint sort -f status,name
```

- Explanation:
  - Manpage says: `sort -f {comma-separated field names} Lexical ascending`
  - In our example this means: Sort the plugins lexically ascending, first by status, then by name.

□□□ #2
□ □□□ □□□ 2026-03-26 17:08:46 UTC
□ □□□ □□□ 2026-03-26 17:08:46 UTC