WP-CLI: Remote WP management

One thing that has always bugged me about WordPress is that it has a GUI only administrator interface and it made it more cumbersome than it needs to be, even for simple publishing workflows like mine where I copy the Markdown text, spell check it, preview it, fix any problems in the preview and scheduled publication.

WordPress CLI (`wp-cli`) adds some flexibility for command line operations for WordPress installations. It also allows you to manage remote installations of WordPress. It's not a perfect solution but it beats having to do everything through the GUI hands down.

Installing

As a Mac user I will install the CLI via Homebrew because it makes it easier and because sooner rather than later I'll forget that I installed it manually and and will run it manually anyways.

Instructions for other operating systems are in the Installation section of the CLI handbook.

Creating a global configuration file

```
# Global parameter defaults
user: carlos
disabled_commands:
  - db drop
# Aliases to other WordPress installs
# An alias can include 'user', 'url', 'path', 'ssh', or 'http'
@local:
    path: <path removed for security>
@production:
    ssh: <ssh removed for security>
    path: <path removed for security>
```

Using in a local dev environment

The basic installation will allow you to work in your local development environment. For example the following commands will create different type of posts. These are some of the things I'm looking to use as part of my workflow.

The first example creates a post and schedules for a future date.

```
# Create post and schedule for future
wp @local post create --post_title='A future post'\
  --post_status=future \
  --post_date='2020-12-01 07:00:00'
```

The second example takes the content of the file as the body of the postand assigns it to categories indicated by the numbers in the field.

```
# Create post with content from given file
# and assign it to pre-defined categories
wp @local post create ./post-content.txt \
  --post_category=201,345 \
  --post_title='Post from file'\
'Post from file'
```

The last example combines future posting with using the content of a file as the body of the post.

This gives you the most flexibility to review and make changes before the date of the post; It allows you to preview in the admin screen but it requires some discipline working far enough ahead of time that you won't be with your back against the wall with deadlines.

```
wp @local post create ./post-content.txt \
  --post_title='A future post'\
  --post_status=future \
  --post_date='2020-12-01 07:00:00'
```

The next step is to run the same commands on our remote production server.

Using in a remote installation

In order for WP-CLI to work on the remote server is to install it on your server. The installation requires the following items

- Shell access to your production server
- Write access to the wordpress installation

If you don't meet the pre-requisites you will not be able to run these scripts.

> These examples duplicate those in the previous sections.
>
> The first one creates a post and schedules it in the future.

```
# Create post and schedule for future
wp @production post create\
  --post_title='A future post'\
  --post_status=future \
  --post_date='2020-12-01 07:00:00'
```

The second example takes the content of a file, assigns it to categories and gives it a title.

```
# Create post with content from given file
wp @production post create \
  --post_content="$(cat post-content.txt)" \
  --post_category=201,345\
  --post_title='Post from file'\
```

The final example expands the previous example by removing the categories, changing the status from draft to future and assigning the date when the post will become available.

```
# Create post with content from file
# and schedule it into the future
wp @production post create \
  --post_content="$(cat xquery.md)" \
```

```
  --post_title='A future post'\
  --post_status=f$(cat xquery.md)"$(cat xquery.md)"ate='2019-02-07 07:00:0
```

The posts may take a long time to appear in the target blog so be ready and publish ahead of time, longer than you normally would.

Duplicating my workflow using WP-CLI

So now that I have working examples, I can duplicate the workflow I currently use with WP-CLI.

> I want to be able to do two things:
>
> - Upload draft posts to my blog so I can proofread them and make any changes in the admin post editor
> - Schedule posts when I feel confident that they are ready to go

The first script takes advantage of positional parameters to feed the data we need into the script. The first parameter is the file that we want to publish and the second one is the title.

> This will get us a draft post ready for review

```
#!/usr/bin/env bash
# Positional parameters
# $1 is the file we want to publish
# $2 is the title of the post.
wp @production post create \
  --post_content="$(cat $1)" \
  --post_title=$2
$(cat $1)"$(cat $1)"
```

The second use case is a little more complicated both in the number of parameters and how we build the script to handle them.

```
#!/usr/bin/env bash

wp @production post create \
  --post_content="$(cat $1)" \
  --post_title=$2\
  --post_status=future \
  --post_date="$3 $4"
```

The script has additional parameters. They are listed and explained below

- $1 Name of the file
- $2 Title
- $3 Date for the post. Format 2019-02-07
- $4 Time for post. Format 07:00:00

Using this script would let you run it from the command line and, if needed, include it as part of a build automation process.

Conclusion

Using WP-CLI you can autommte a lot of the management tasks that would normally require acccess to the administrator UI and give you a more direct way of handling posting to your blog or blogs.

We haven't covered all the functionality available in WP-CLI. There is a lot more you can do and it's definitely worth exploring.

Links and Resources

- WP-CLI Handbook
- WP-CLI Common issues and their fixes
- WP-CLI – Advanced WordPress Management
- WP-CLI Tutorial
- Running Commands Remotely
- Making WP-CLI accessible on a remote server