Creating WordPress staging environments

Although I've always thought about staging environments as clones of production systems, I understand the need to do things differently in production than in staging or development.

This would be no different than using Gulp to only concatenate and minify scripts in production or use compressed method when transpiling SASS but until version 5.5 there was no native way to do it.

WordPress 5.5 introduced the wp_get_environment_type() function that allows developers to branch code based on the type of server we're running.

First set the `WP_ENVIRONMENT_TYPE` variable in `wp-config.php`

```
define( 'WP_ENVIRONMENT_TYPE', 'staging' );
```

Note

When `wp_get_environment_type()` is development, WordPress will set `WP_DEBUG` to true if it is not defined in `wp-config.php`.

Once we set up the environment type, we can use it to provide different headers or other type of content for different types of server.

For example, we use different functions to display the header for a page or post type. `show_production_header()` could include the identification strings required for Google and Blink to work, staging and development headers could include debug intstrumentation beyond `WP_DEBUG`.

```
<?php
switch ( wp_get_environment_type() ) {
```

```
  case 'local':
  case 'development':
    show_development_header();
    break;
  case 'staging':
    show_staging_header();
    break;
  case 'production':
  default:
    show_production_header();
    break;
}
```

Just like the headers example, we can use a similar switch statement to create environment-specific branches in our code.

Your plugins may want to disable features when working in Development or staging modes or make features work differently in Production sites.

For an in-depth discussion of why this is necessary, see issue 33161 in WordPress core Trac.