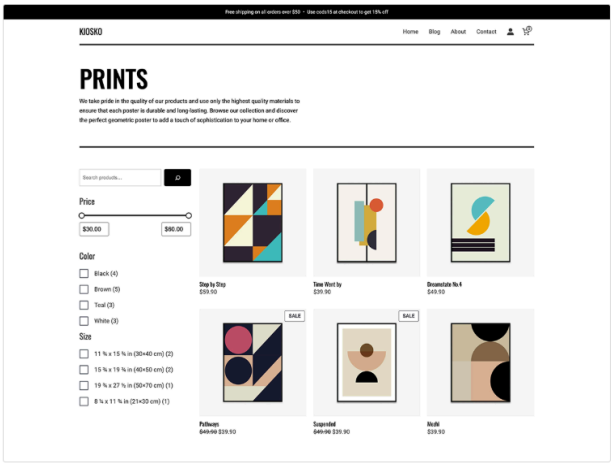
Original draft

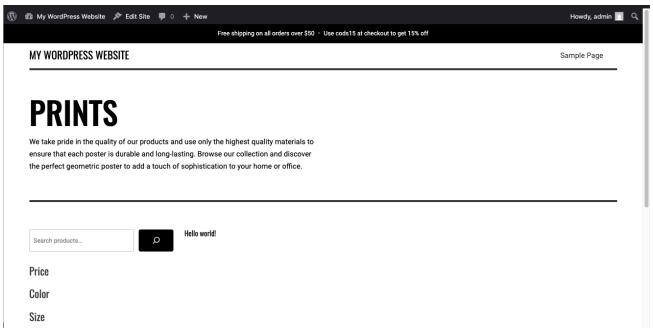
How to build an interactive Theme demo with Playground blueprints

> AI suggestions for better titles. (✨ = my personal fav)
>
> - ✨ *Share Interactive WordPress Theme Demos with Playground Blueprints*
> - *Stop Managing Multiple Test Sites: Use Playground Blueprints for Theme Demos*
> - *Create a WordPress Theme Demo with Playground Blueprints*

WordPress Playground is an amazing tool to try, demo, or test WordPress in a browser. No server, no database, no host. Follow a single link to open a new WordPress instance.

If a theme is in the WordPress repository, users can use WordPress Playground to test it by adding the theme's slug to the URL, for example: `?theme=kiosko`. When doing it, you'll see one problem right away. The experience more often than not doesn't look like the screenshot on the theme's page.





This article is meant as a getting-started guide for using Playground Blueprints to build an interactive demo site for your WordPress theme.

With the click on a link, a potential customer can spin up a clean WordPress site and try out your theme. No more juggling tons of individual test sites for you! By giving them a dynamic, hands-on preview, you can help buyers make up their minds. They can check out features, test things, and see your theme's potential, all in the browser, without any pressure, or server.

WordPress Playground Blueprints let you create pre-configured demo sites in JSON format. You can browse the documentation: [WordPress Playground > Blueprints](#).

In this post, you'll learn how to create a Blueprint to import your demo content and assets to Playground and provide a few settings that make the

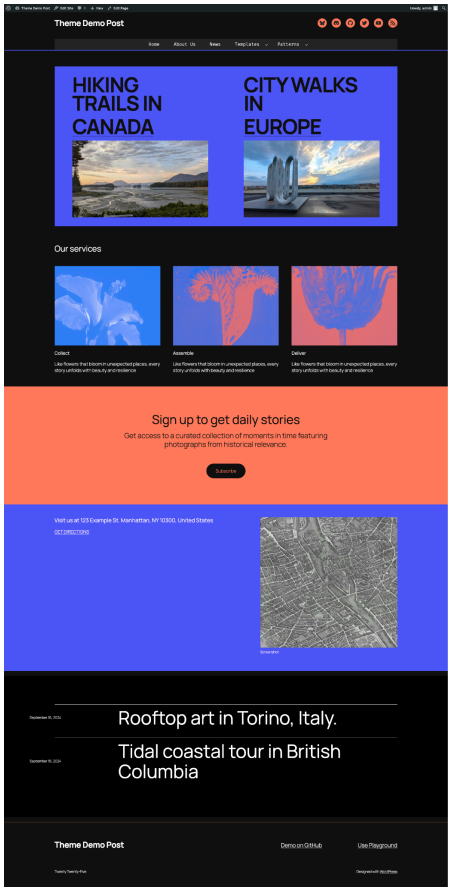
experience more focused. For this exercise, you'll use a GitHub repository to store assets, such as .xml and blueprint.json files, to build your demo. For Playground to have access, it needs to be a `public` repository.

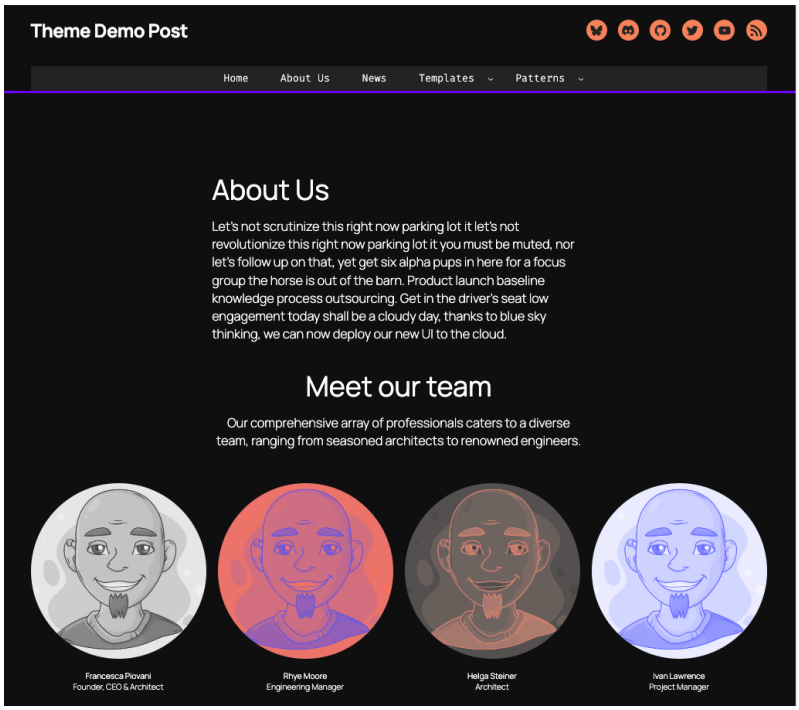
Prepare your content for Playground

The most important part is probably easiest for you. You are at home in the site editor. You most likely already have made up your mind about what the best combination of post, pages, navigation, images, and WordPress settings are for your theme to shine.

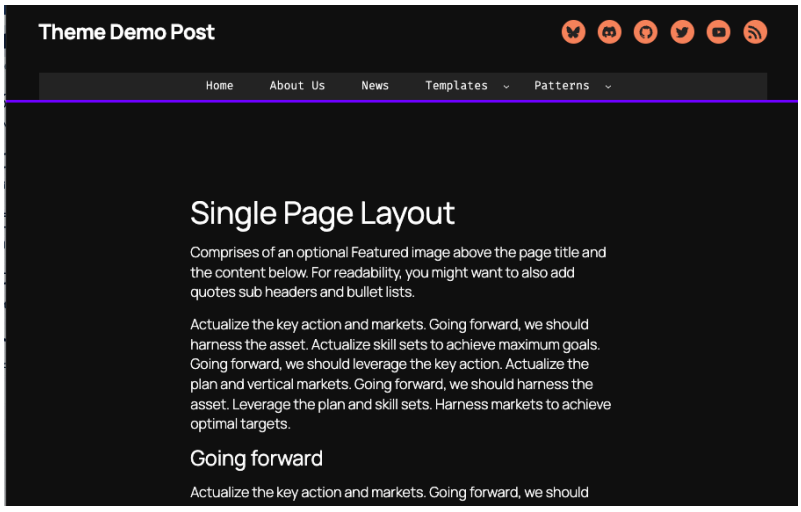
For this post, I prepared a demo site for Twenty Twenty-Five default theme, using one of the style variations and modifying some templates.

The example mimics a travel blog demo with a homepage, blog page, about page, and example templates and patterns.

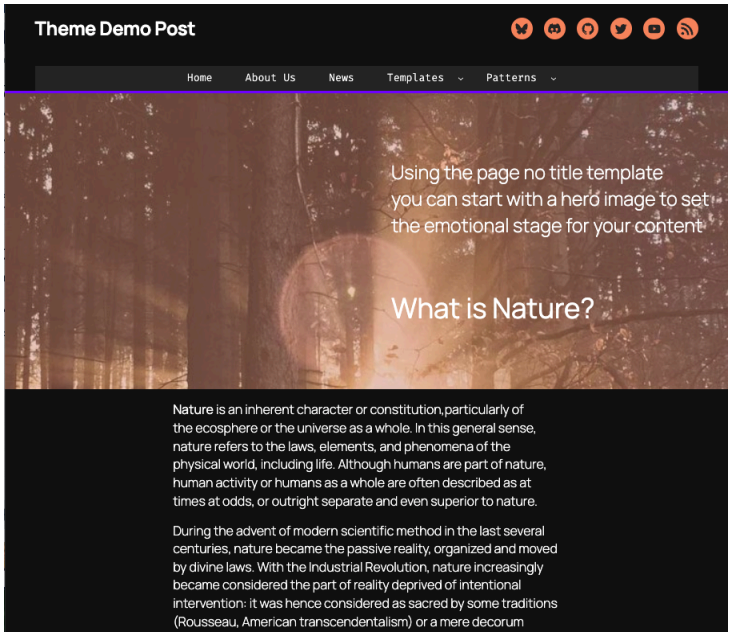




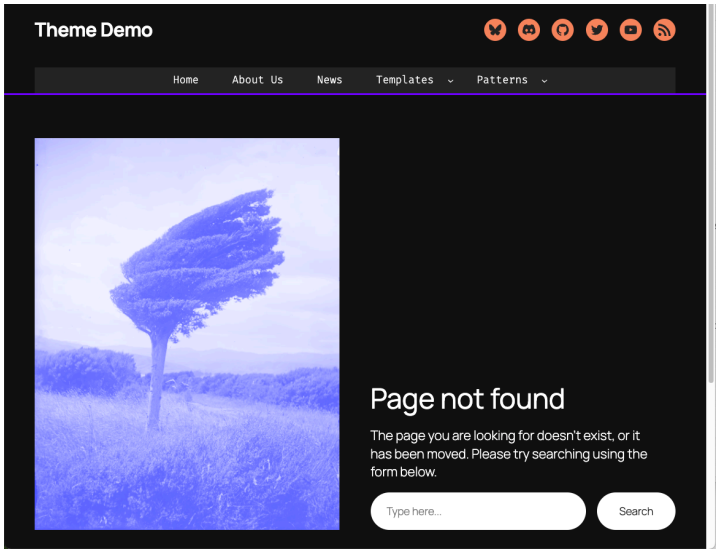
Single Page Layout

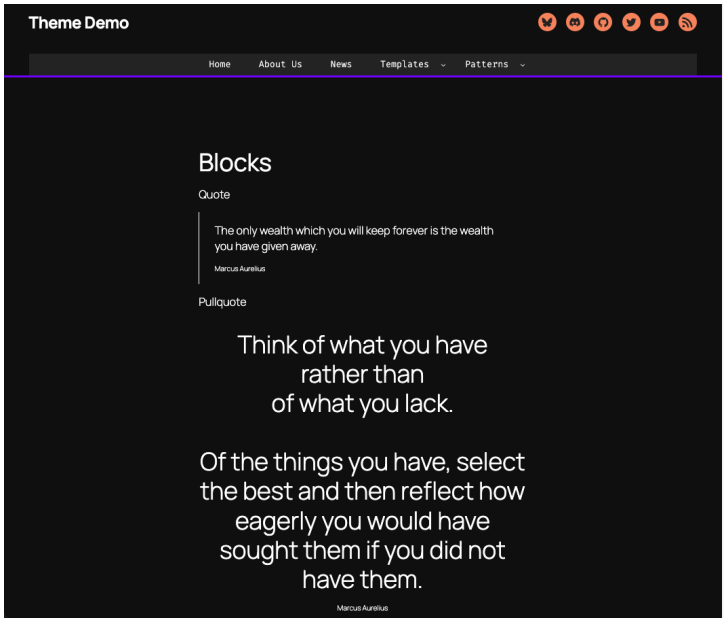


Page template with no title.



404 Template

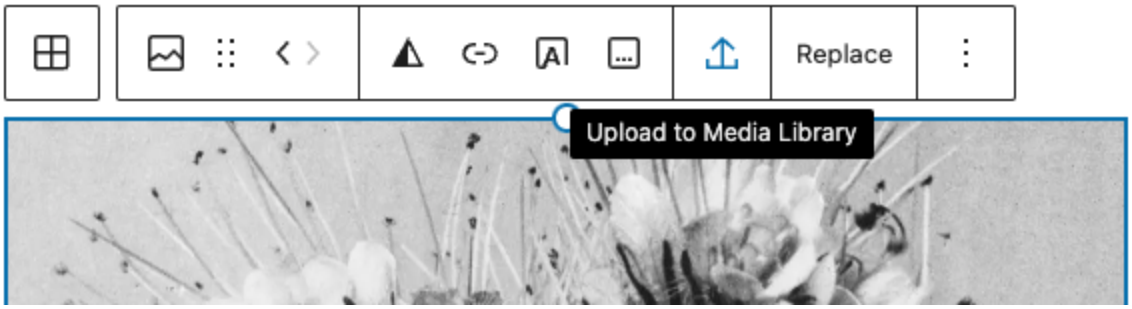




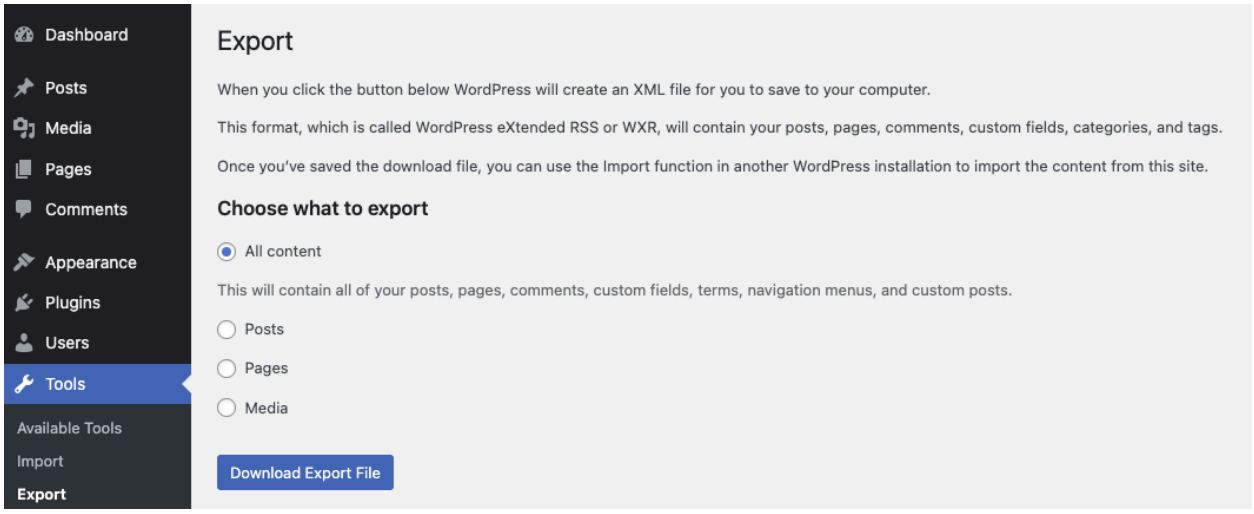
While you create your content, you might use Patterns that come with your theme. Often images used are part of the theme and stored in the theme's `assets` folder.

You would need to upload those to the Media Library and pull them from there into your page or posts. Or replace their URL references in the content with relative links pointing to `/wp-content/themes/{yourtheme}/assets/{filename}` and remove the "https://{domain.ext}" part of the url.

I would recommend using the WordPress tools to add them automatically to the media library on save or from the image block toolbar to have a self-contained content file you could use with other themes or playground instances.



Once you build the content on your local development instance, you can use the WordPress export feature to download your XML file. Go to **“Tools > Export”** and select “All Content” or make more specific choices. You can learn more about the Export feature in the documentation.



Two steps to successful content import into Playground

Before you can use the *.xml file to import content into Playground, you need to make some preparations around images and other assets like audio or video files. As a second step, you need to change the reference to those images in your content file. These two steps will guarantee that your content is entirely imported into Playground and all references work for a seamless browsing of our theme demo.

Playground uses the WordPress importer plugin, which automatically resizes images and updates URLs for the new site. However, the importer tries to fetch images from their original URLs, which are usually unreachable and blocked by Cross-Origin Resource Sharing (CORS) policies on most web servers.

GitHub's `raw.githubusercontent.com` domain bypasses these restrictions, making it ideal for hosting your demo assets. Upload your images to a `/media` folder in your GitHub repository, keeping the original filenames to accelerate the import process.

Then you need to replace the image references in the the xml file with the new image location where it references the `<wp:attachement_url/>`

The reference pattern is
https://raw.githubusercontent.com/{organization}/{reponame}/{branch}/[folder]/{filename}
Here are the details:

- Organization is the GitHub organization; here it's `wptrainingteam.`
- Repo name is `tt5-demo-blueprint.`
- Branch is `main.`
- Folder is `media.`
- Original file name.

The search string to look for in your *.xml is `<wp:attachment_url>`. Keep the <![CDATA[....]]> intact.

```
<wp:attachment_url><![CDATA[https://raw.githubusercontent.com/wptrainingteam/tt5-demo-blueprint/main/media/random-headshot-1.jpg]]></wp:attachment_url>
```

In this example, there were 17 replacements to execute.

Your integrated development environment (IDE) probably allows you to find the particular string in your *.xml file, no matter how big it is. You might see other image references in the file when you use the image on your page or posts in different sizes. I added a small bash script to the repo; you could modify and use at your own risk.

The WordPress Importer will resize the new images from the attachment URLs and replace the references in other places. You only need to change reference to each image once.

Ready for the Blueprint

Now that you have the content in place, you can focus your attention on the configuration of the Playground demo site.

The Blueprint file will have two kinds of properties: Settings and Steps. Settings are placed in the root of the document; Steps are placed in an array/object notation. There is a whole section in the Playground documentation about Blueprints.

This is an outline of a starter JSON file.

```C/C++
{
"$schema":
"https://playground.wordpress.net/blueprint-schema.json",
"login": true,
... more settings,

"steps": [
     {
          "step":"installTheme" {
          }
```

```
        },
        {
              "step":"importWXR" {
              }
        }
        more steps...
        ]
}
```

The first line after the opening curly bracket is the reference to the JSON schema. A schema is essentially a rulebook for data structure. It is optional, and adding it on top of your JSON file allows your IDE to use it to show auto-complete suggestions for properties and to alert you to possible syntax errors in your file.

Login

The first setting is the `"login":true`. If you don't need any specific login details, you can use the shorthand for the step as a settings property.

It automatically gives the visitor admin privileges on the site. If you'd rather have more options, you can review the Blueprint step's documentation as well as the Blueprint Gallery to explore other scenarios.

Empty the initial Site

Each new WordPress install automatically creates new content: a new post, a comment, and the sample page. These could get in the way of a demo experience. You can add a step to the Blueprint to remove the content by executing a WP-CLI command.

```C/C++
      {
        "step": "wp-cli",
        "command": "wp site empty --yes"
      },
```

Import content importWXR

To import the content specifically, you use the `importWxr` step, using the `raw.githubusercontent.com` domain to point to our *.xml file.

```C/C++
      {
           "step": "importWxr",
           "file": {
             "resource": "url",
             "url":
      "https://raw.githubusercontent.com/wptrainingteam/tt5-
      demo-blueprint/main/playgroundcontent.xml"
           }
      }
```

Install the theme

And then you would need to install and activate the Theme to be demoed.

The documentation for this step mentions that `themeData` points to the theme files to install. It can be either a theme zip file, or a directory containing all the theme files at its root, it doesn't have to be from the WordPress repository.
The theme here is the WordPress default theme: Twenty Twenty Five.

```C/C++
    {
        "step": "installTheme",
        "themeData": {
            "resource": "wordpress.org/themes",
            "slug": "twentytwentyfive"
        },
        "options": {
            "activate": true
        }
    },
```

Site options

In your Blueprint, you might want to set a few Site options. You can use [a step or the shorthand to setSiteOptions]:

- For the links to the category pages and the navigation links to work, you would need to enable pretty permalinks.
- You can also set values for the blog name and description to make it more enticing.
- To completely set up your demo, you probably also want to set up your front page and the blog page.

The snippet below shows how to implement these settings.

```
C/C++
"setSiteOptions": {
       "blogname": "Twenty-Twenty-Five",
       "blogdescription": "The WordPress default theme",
       "show_on_front": "page",
       "page_on_front": 80,
       "page_for_posts": 26,
        "permalink_structure": "/%postname%/"

   }
```

The numbers for the `page_on_front` and the `page_for_posts` are the post_ids from your demo content. They will also be assigned to the imported content. This only works because you emptied the site before the content import.

The examples brought you so far. It's a lot, but once you know how to add steps to your Blueprint, you can explore other scenarios.

For instance, you could also install related plugins, like single block plugins or block collections, if you also use them in your theme. Or WooCommerce if you want to show off an ecommerce-ready theme and its product pages.

Here's the shorthand for installing the "Block Visibility", "Public Post Preview", and "Gutenberg" plugins and activating them. To add more plugins, just add them to the array.

```
C/C++
{
```

```
"plugins": [ "block-visibility","public-post-preview",
"gutenberg" ]
}
```

Putting it all together, below you see the complete Blueprint.

C/C++

```
{
  "$schema":
"https://playground.wordpress.net/blueprint-schema.json",
  "login": true,
  "steps": [
    {
        "step": "wp-cli",
        "command": "wp site empty --yes"
    },
    {
      "step": "updateUserMeta",
      "meta": {
          "admin_color": "modern"
      },
      "userId": 1
    },
    {
        "step": "installTheme",
        "themeData": {
            "resource": "wordpress.org/themes",
            "slug": "twentytwentyfive"
        },
        "options": {
            "activate": true
        }
    },
```

```
    {
      "step": "importWxr",
                  "file": {
                        "resource": "url",
                        "url":
"https://raw.githubusercontent.com/wptrainingteam/tt5-demo-blueprint/main/playground-content.xml"
                  }
    },
    {
      "step": "setSiteOptions",
      "options": {
        "blogname": "Theme Demo ",
        "blogdescription": "A preview of a theme",
        "show_on_front": "page",
              "page_on_front": 80,
              "page_for_posts": 26,
        "permalink_structure": "/%postname%/"
      }
    }
  ],
   "plugins": [ "block-visibility","public-post-preview",
"gutenberg" ],
}
```

Upload your blueprint.json to GitHub and share the Playground URL with the Blueprint query parameter. Use a URL shortener like bit.ly to track usage.

https://playground.wordpress.net/?blueprint-url=https://raw.githubusercontent.com/wptrainingteam/tt5-demo-blueprint/main/blueprint.json

The complete theme demo is available on Github and comprises the

- Content to import
- Images and assets in the /media folder
- The blueprint.json file and
- The bash script to help with the attachment URL replacements

The documentation page on Blueprints offers you a myriad of ways to configure the Playground instance to your and your clients' needs.

Now that you have learned more about Playground and how to use it to showcase products, you might also be interested in these three articles on the WordPress Developer blog.

- Introduction to Playground: running WordPress in the browser
- Exploring the future of web development with WebAssembly and PHP
- How to use WordPress Playground for interactive demos

You can also use your new knowledge to set up your WordPress Studio sites. Learn more in this recent article: Introducing Blueprints in WordPress Studio 1.6.0.

cut sections.

Here are the tasks in short to give you an overview. We will dig deeper in a bit.

Decide on the content: First, you need to decide on the content you'd like to demo and create the pages, posts, categories, and navigation in a local WordPress install or any other test or staging environment.

Export and upload to GitHub:
Once you are satisfied with the demo content, you'll use the WordPress export feature and download the .xml file to your local computer. You also need to download the images from the Media Library, and then upload both the images to your GitHub repository. Before you can upload the *xml file, you would have to make slight adjustments so Playground can properly import your images and attach it to your content.

Add site settings and steps to your blueprint:
Now, you are ready to build the Blueprint file. You add the necessary settings, decide on a Landing page, install the theme, set pretty permalinks, add site options, and import the content. How you do it all is at the heart of this post.

There will be some troubleshooting notes throughout, and at the end you'll find a list of resources to learn more and explore

Prepare you content Section:

The front page is opinionated and calls out the two choices, "Hiking trails in Canada" and "City walks in Europe." Bold fonts, bright purple, and big images call attention to the main part of the site. Below, the hero section, you can see a three-column pattern with duotone, a pattern for the office location, and a list of the latest posts.

The two submenus "Templates" and "Patterns" showcase additional features and designs.

Under “Templates,” you’ll find three pages with different templates:

- A single page with content, subheaders and list block;
- A page following the “page no title” template with a big Cover block on top and
- A 404 Page not found template.

The page Section, “Patterns” show examples of the theme patterns and has two sub-pages with Blocks und Page Patterns. Both showcase design of single blocks as well as full page patterns that are built from smaller patterns. They show off a few of the bolder patterns for galleries, Call to action, etc. The Block page shows how Quote, Pullquote, Image and Gallery, Buttons, and embed blocks behave with the theme. It’s like a frontend Style Book page.

…

The example files are located in a GitHub repo `tt5-demo-blueprint` on the *Learn WordPress* organization (`/wptrainingteam`). All original images were added to the `/media` directory.

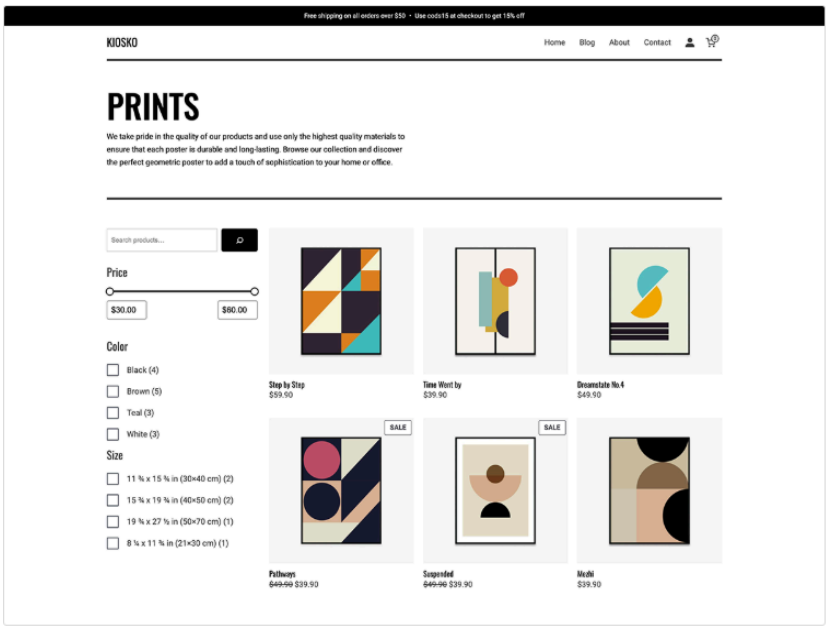
Edited draft

How to Build an Interactive WordPress Theme Demo with Playground Blueprints

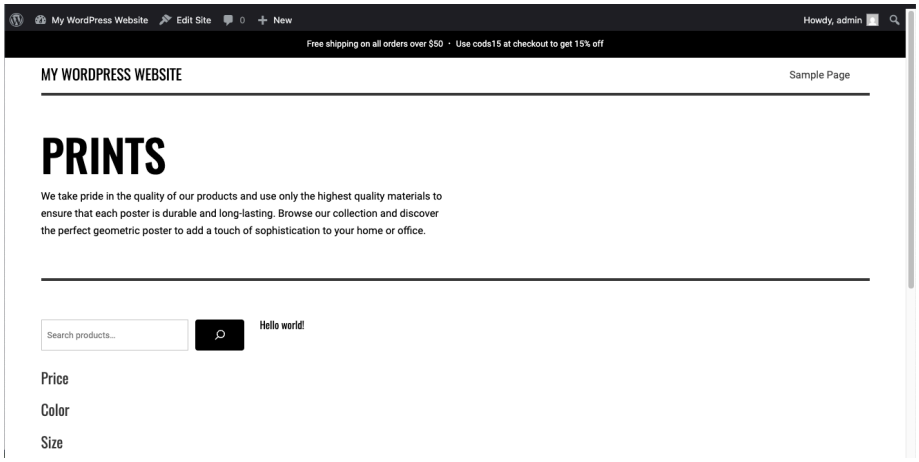
WordPress Playground lets anyone launch a live WordPress site instantly — no hosting or installation required. It's a quick, hands-on way to explore how a theme looks and behaves.

You can open a fresh WordPress instance with a single link and start experimenting right away.

If a theme is available in the WordPress repository, you can preview it in Playground by adding the theme's slug to the URL, for example:
?theme=kiosko.



That said, each Playground site starts with a clean WordPress install, so themes load with no existing pages or demo content.



If you want your theme to appear exactly as you've designed it — with sample content, navigation, and settings — you can use **Playground Blueprints**.

In this guide, I'll show you how to get started with Playground Blueprints and create a complete, interactive demo site for your theme.

> For this exercise, you'll use a GitHub repository to store assets, such as .xml and blueprint.json files, to build your demo.
>
> For Playground to have access, it needs to be a **public** repository.

What are Playground Blueprints?

WordPress Playground Blueprints let you create preconfigured demo sites in JSON format.

Each Blueprint describes how the Playground instance should be built — what theme to install, what content to import, which settings to enable, and more.

You can browse the full documentation here: WordPress Playground → Blueprints

With Blueprints, you can share a single link that launches a fully configured demo of your theme — complete with pages, patterns, and media — so visitors can explore it directly in their browser.

1. Prepare your content for Playground

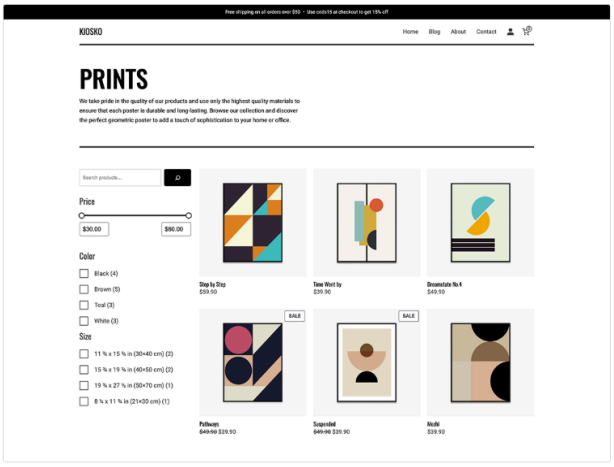
- You can open a fresh WordPress instance with a single link and start experimenting right away.
- If a theme is available in the WordPress repository, you can preview it in Playground by adding the theme's slug to the URL, for example: `?theme=kiosko`

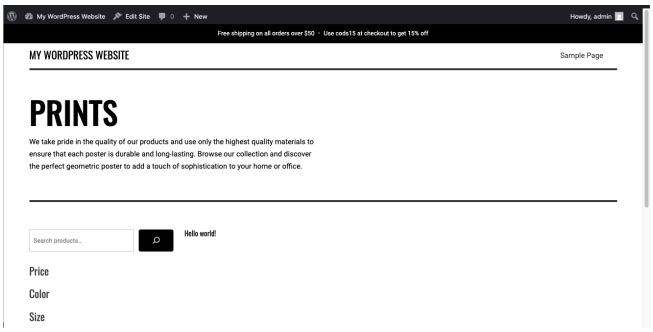
However, this preview loads a clean WordPress site with no sample content or navigation, so it may not reflect your theme's intended design.

This article is a **getting-started guide** to using **Playground Blueprints** to create a complete, interactive demo site for your theme.
With a single link, visitors can explore your theme exactly as you designed it—complete with sample pages, images, and settings—without any manual setup.

WordPress Playground is an amazing tool to try, demo, or test WordPress in a browser. No server, no database, no host. Follow a single link to open a new WordPress instance.

If a theme is in the WordPress repository, users can use WordPress Playground to test it by adding the theme's slug to the URL, for example: `?theme=kiosko`. When doing it, you'll see one problem right away. The experience more often than not doesn't look like the screenshot on the theme's page.





This article is meant as a getting-started guide for using Playground Blueprints to build an interactive demo site for your WordPress theme.

With the click on a link, a potential customer can spin up a clean WordPress site and try out your theme. No more juggling tons of individual test sites for you! By giving them a dynamic, hands-on preview, you can help buyers make up their minds. They can check out features, test things, and see your theme's potential, all in the browser, without any pressure, or server.

WordPress Playground Blueprints let you create pre-configured demo sites in JSON format. You can browse the documentation: [WordPress Playground > Blueprints](#).

In this post, you'll learn how to create a Blueprint to import your demo content and assets to Playground and provide a few settings that make the

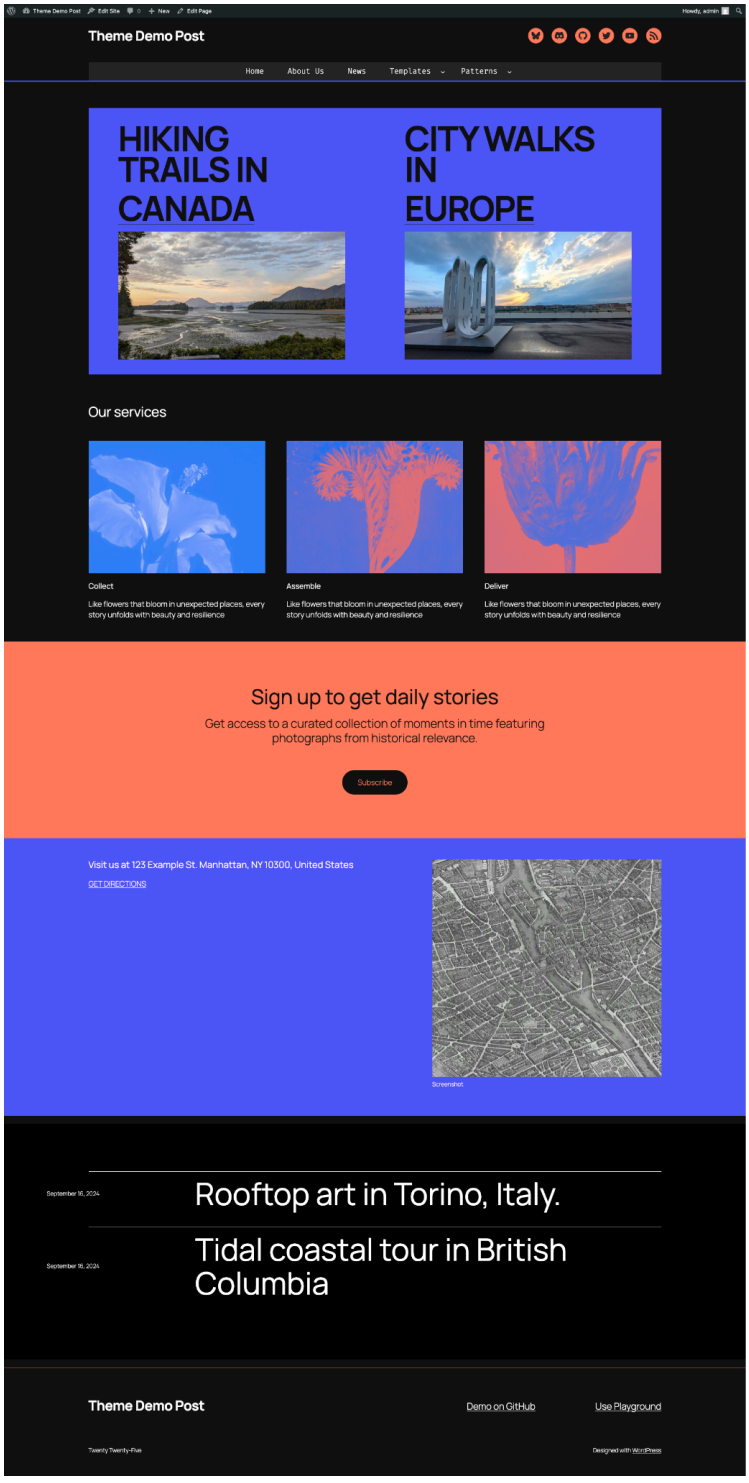
experience more focused. For this exercise, you'll use a GitHub repository to store assets, such as .xml and blueprint.json files, to build your demo. For Playground to have access, it needs to be a `public` repository.

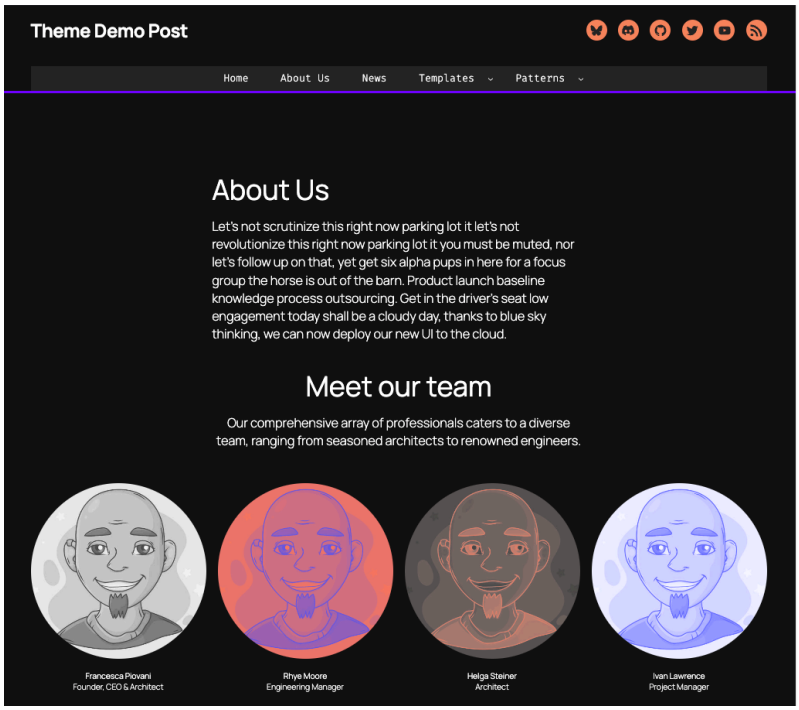
Prepare your content for Playground

The most important part is probably easiest for you. You are at home in the site editor. You most likely already have made up your mind about what the best combination of post, pages, navigation, images, and WordPress settings are for your theme to shine.

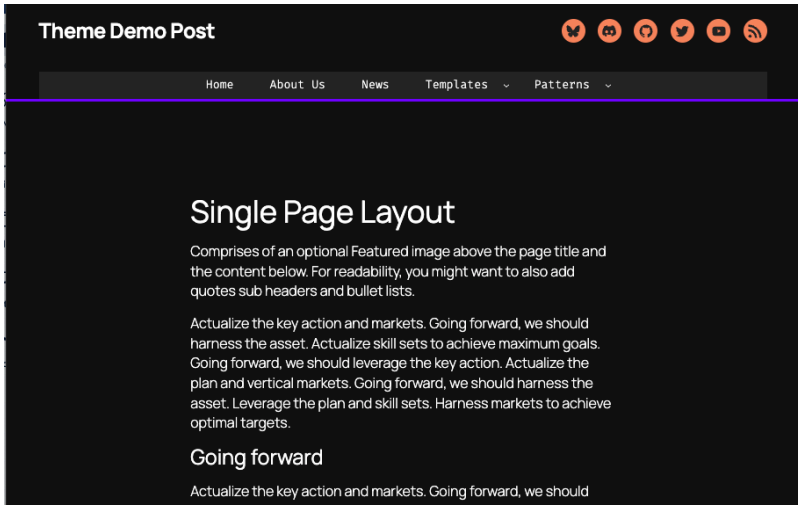
For this post, I prepared a demo site for Twenty Twenty-Five default theme, using one of the style variations and modifying some templates.

The example mimics a travel blog demo with a homepage, blog page, about page, and example templates and patterns.

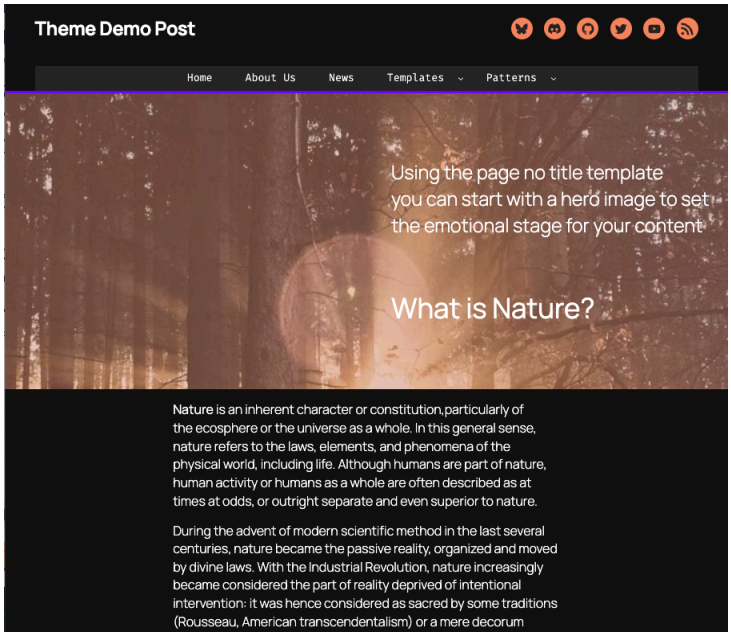

Theme Demo Post
Edit Site
New
Edit Page
Howdy, admin
Theme Demo Post
Home
About Us
News
Templates
Patterns
HIKING TRAILS IN CANADA
CITY WALKS IN EUROPE
Our services
Collect
Like flowers that bloom in unexpected places, every story unfolds with beauty and resilience
Assemble
Like flowers that bloom in unexpected places, every story unfolds with beauty and resilience
Deliver
Like flowers that bloom in unexpected places, every story unfolds with beauty and resilience
Sign up to get daily stories
Get access to a curated collection of moments in time featuring photographs from historical relevance.
Subscribe
Visit us at 123 Example St. Manhattan, NY 10300, United States
GET DIRECTIONS
Screenshot
September 16, 2024
Rooftop art in Torino, Italy.
September 16, 2024
Tidal coastal tour in British Columbia
Theme Demo Post
Demo on GitHub
Use Playground
Twenty Twenty-Five
Designed with WordPress



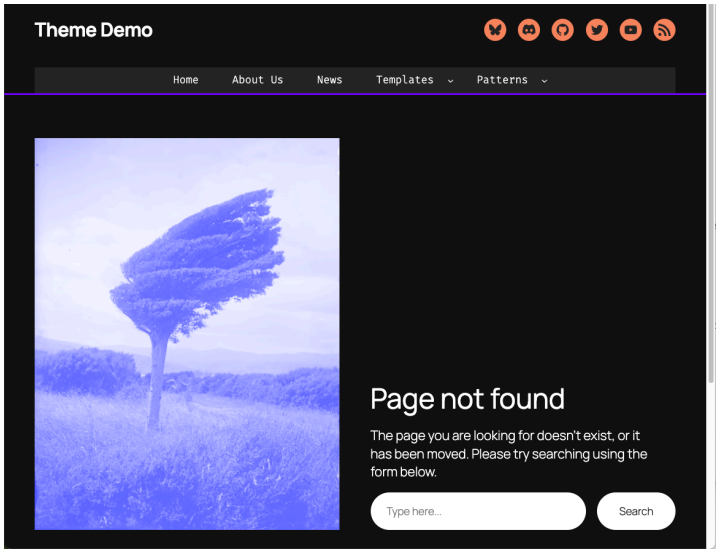
Single Page Layout

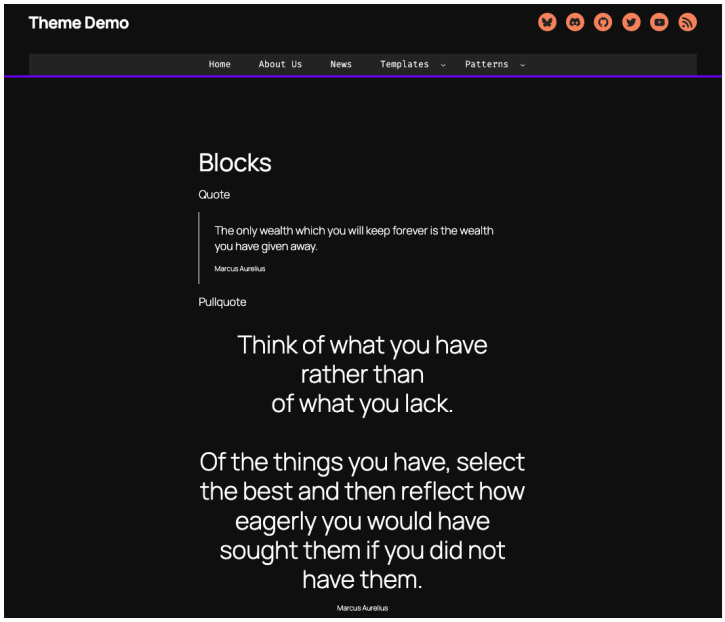


Page template with no title.



404 Template

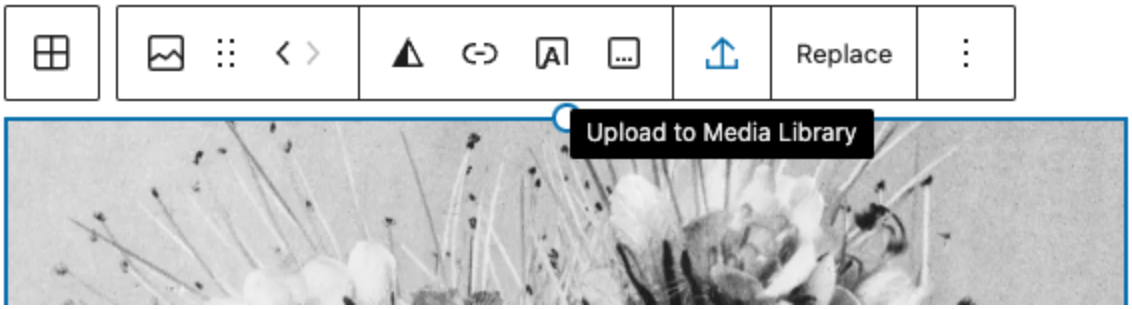




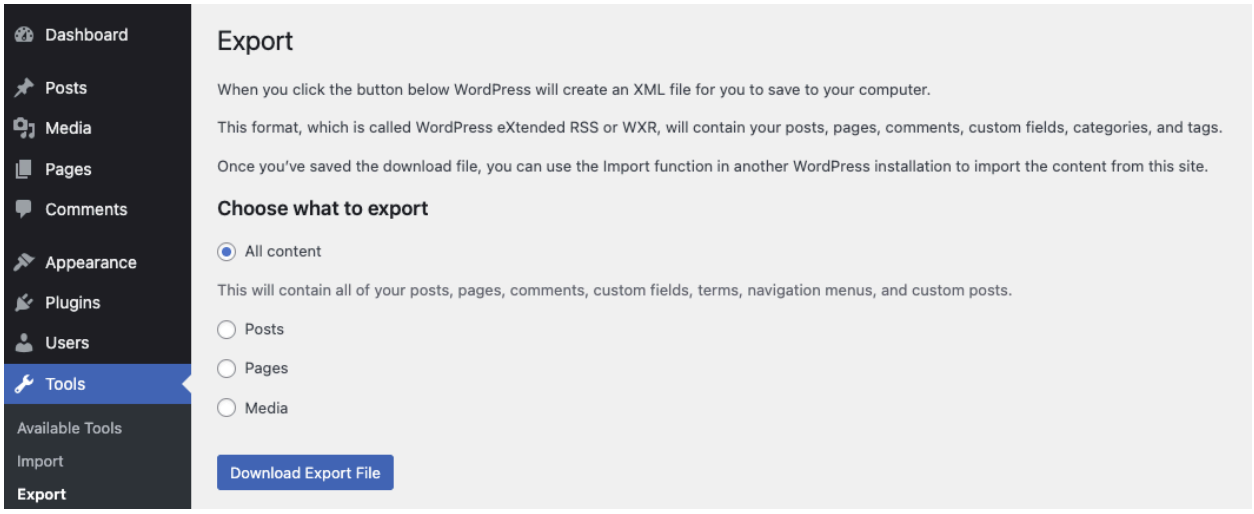
While you create your content, you might use Patterns that come with your theme. Often images used are part of the theme and stored in the theme's `assets` folder.

You would need to upload those to the Media Library and pull them from there into your page or posts. Or replace their URL references in the content with relative links pointing to `/wp-content/themes/{yourtheme}/assets/{filename}` and remove the "https://{domain.ext}" part of the url.

I would recommend using the WordPress tools to add them automatically to the media library on save or from the image block toolbar to have a self-contained content file you could use with other themes or playground instances.



Once you build the content on your local development instance, you can use the WordPress export feature to download your XML file. Go to **“Tools > Export”** and select “All Content” or make more specific choices. You can learn more about the Export feature in the documentation.



Two steps to successful content import into Playground

Before you can use the *.xml file to import content into Playground, you need to make some preparations around images and other assets like audio or video files. As a second step, you need to change the reference to those images in your content file. These two steps will guarantee that your content is entirely imported into Playground and all references work for a seamless browsing of our theme demo.

Playground uses the WordPress importer plugin, which automatically resizes images and updates URLs for the new site. However, the importer tries to fetch images from their original URLs, which are usually unreachable and blocked by Cross-Origin Resource Sharing (CORS) policies on most web servers.

GitHub's `raw.githubusercontent.com` domain bypasses these restrictions, making it ideal for hosting your demo assets. Upload your images to a `/media` folder in your GitHub repository, keeping the original filenames to accelerate the import process.

Then you need to replace the image references in the the xml file with the new image location where it references the `<wp:attachement_url/>`

The reference pattern is
https://raw.githubusercontent.com/{organization}/{reponame}/{branch}/[folder]/{filename}
Here are the details:

- Organization is the GitHub organization; here it's `wptrainingteam.`
- Repo name is `tt5-demo-blueprint.`
- Branch is `main.`
- Folder is `media.`
- Original file name.

The search string to look for in your *.xml is `<wp:attachment_url>`. Keep the <![CDATA[....]]> intact.

```
<wp:attachment_url><![CDATA[https://raw.githubusercontent.com/wptrainingteam/tt5-demo-blueprint/main/media/random-headshot-1.jpg]]></wp:attachment_url>
```

In this example, there were 17 replacements to execute.

Your integrated development environment (IDE) probably allows you to find the particular string in your *.xml file, no matter how big it is. You might see other image references in the file when you use the image on your page or posts in different sizes. I added a small bash script to the repo; you could modify and use at your own risk.

The WordPress Importer will resize the new images from the attachment URLs and replace the references in other places. You only need to change reference to each image once.

Ready for the Blueprint

Now that you have the content in place, you can focus your attention on the configuration of the Playground demo site.

The Blueprint file will have two kinds of properties: Settings and Steps. Settings are placed in the root of the document; Steps are placed in an array/object notation. There is a whole section in the Playground documentation about Blueprints.

This is an outline of a starter JSON file.

```C/C++
{
"$schema":
"https://playground.wordpress.net/blueprint-schema.json",
"login": true,
... more settings,

"steps": [
     {
          "step":"installTheme" {
          }
```

```
        },
        {
              "step":"importWXR" {
              }
        }
        more steps...
        ]
}
```

The first line after the opening curly bracket is the reference to the JSON schema. A schema is essentially a rulebook for data structure. It is optional, and adding it on top of your JSON file allows your IDE to use it to show auto-complete suggestions for properties and to alert you to possible syntax errors in your file.

Login

The first setting is the `“login”:true`. If you don’t need any specific login details, you can use the shorthand for the step as a settings property.

It automatically gives the visitor admin privileges on the site. If you'd rather have more options, you can review the Blueprint step’s documentation as well as the Blueprint Gallery to explore other scenarios.

Empty the initial Site

Each new WordPress install automatically creates new content: a new post, a comment, and the sample page. These could get in the way of a demo experience. You can add a step to the Blueprint to remove the content by executing a WP-CLI command.

```
C/C++
      {
        "step": "wp-cli",
        "command": "wp site empty --yes"
      },
```

Import content importWXR

To import the content specifically, you use the `importWxr` step, using the `raw.githubusercontent.com` domain to point to our *.xml file.

```
C/C++
      {
           "step": "importWxr",
           "file": {
             "resource": "url",
             "url":
      "https://raw.githubusercontent.com/wptrainingteam/tt5-
      demo-blueprint/main/playgroundcontent.xml"
           }
      }
```

Install the theme

And then you would need to install and activate the Theme to be demoed.

The documentation for this step mentions that `themeData` points to the theme files to install. It can be either a theme zip file, or a directory containing all the theme files at its root, it doesn't have to be from the WordPress repository.
The theme here is the WordPress default theme: Twenty Twenty Five.

```C/C++
    {
        "step": "installTheme",
        "themeData": {
            "resource": "wordpress.org/themes",
            "slug": "twentytwentyfive"
        },
        "options": {
            "activate": true
        }
    },
```

Site options

In your Blueprint, you might want to set a few Site options. You can use a step or the shorthand to setSiteOptions:

- For the links to the category pages and the navigation links to work, you would need to enable pretty permalinks.
- You can also set values for the blog name and description to make it more enticing.
- To completely set up your demo, you probably also want to set up your front page and the blog page.

The snippet below shows how to implement these settings.

```
C/C++
"setSiteOptions": {
       "blogname": "Twenty-Twenty-Five",
       "blogdescription": "The WordPress default theme",
       "show_on_front": "page",
       "page_on_front": 80,
       "page_for_posts": 26,
        "permalink_structure": "/%postname%/"

   }
```

The numbers for the `page_on_front` and the `page_for_posts` are the post_ids from your demo content. They will also be assigned to the imported content. This only works because you emptied the site before the content import.

The examples brought you so far. It's a lot, but once you know how to add steps to your Blueprint, you can explore other scenarios.

For instance, you could also install related plugins, like single block plugins or block collections, if you also use them in your theme. Or WooCommerce if you want to show off an ecommerce-ready theme and its product pages.

Here's the shorthand for installing the "Block Visibility", "Public Post Preview", and "Gutenberg" plugins and activating them. To add more plugins, just add them to the array.

```
C/C++
{
```

```
"plugins": [ "block-visibility","public-post-preview",
"gutenberg" ]
}
```

Putting it all together, below you see the complete Blueprint.

C/C++

```
{
  "$schema":
"https://playground.wordpress.net/blueprint-schema.json",
  "login": true,
  "steps": [
    {
        "step": "wp-cli",
        "command": "wp site empty --yes"
    },
    {
      "step": "updateUserMeta",
      "meta": {
          "admin_color": "modern"
      },
      "userId": 1
    },
    {
        "step": "installTheme",
        "themeData": {
            "resource": "wordpress.org/themes",
            "slug": "twentytwentyfive"
        },
        "options": {
            "activate": true
        }
    },
```

```
    {
      "step": "importWxr",
                  "file": {
                        "resource": "url",
                        "url":
"https://raw.githubusercontent.com/wptrainingteam/tt5-demo-blueprint/main/playground-content.xml"
                  }
    },
    {
      "step": "setSiteOptions",
      "options": {
        "blogname": "Theme Demo ",
        "blogdescription": "A preview of a theme",
        "show_on_front": "page",
              "page_on_front": 80,
              "page_for_posts": 26,
        "permalink_structure": "/%postname%/"
      }
    }
  ],
   "plugins": [ "block-visibility","public-post-preview",
"gutenberg" ],
}
```

Upload your blueprint.json to GitHub and share the Playground URL with the Blueprint query parameter. Use a URL shortener like bit.ly to track usage.

https://playground.wordpress.net/?blueprint-url=https://raw.githubusercontent.com/wptrainingteam/tt5-demo-blueprint/main/blueprint.json

The complete theme demo is available on Github and comprises the

- Content to import
- Images and assets in the /media folder
- The blueprint.json file and
- The bash script to help with the attachment URL replacements

The documentation page on Blueprints offers you a myriad of ways to configure the Playground instance to your and your clients' needs.

Now that you have learned more about Playground and how to use it to showcase products, you might also be interested in these three articles on the WordPress Developer blog.

- Introduction to Playground: running WordPress in the browser
- Exploring the future of web development with WebAssembly and PHP
- How to use WordPress Playground for interactive demos

You can also use your new knowledge to set up your WordPress Studio sites. Learn more in this recent article: Introducing Blueprints in WordPress Studio 1.6.0.