GitHub Pages

Web sites for you and your projects

Hosted directly from your [GitHub repository](https://github.com/). Just edit, push, and your changes are live.

Roll vanilla, or generate a site for your project to quickly get started.

You get one site per GitHub account and organization,   
and unlimited project sites. Ready? Let’s get started.

**User or organization site**

#### Create a repository

Head over to [GitHub](https://github.com) and [create a new repository](https://github.com/new) named username.github.io, where username is your username (or organization name) on GitHub.

If the first part of the repository doesn’t exactly match your username, it won’t work, so make sure to get it right.

#### Clone the repository

Go to the folder where you want to store your project, and clone the new repository:

1. git clone <https://github.com/username/username.github.io>

#### Hello World

Enter the project folder and add an index.html file:

cd username.github.io

echo "Hello World" > index.html

#### Push it

Add, commit, and push your changes:

git add --all

git commit -m "Initial commit"

git push -u origin master

#### …and you're done!

Fire up a browser and go to **http://username.github.io**.

## Using Jekyll as a static site generator with GitHub Pages

If you use Jekyll as a static site generator with GitHub Pages, you benefit from more support with setting up, updating, and troubleshooting your site.

###### [Customizing GitHub Pages](https://help.github.com/categories/customizing-github-pages) / About GitHub Pages and Jekyll

Top of Form



Bottom of Form

Top of Form

Bottom of Form

## About GitHub Pages and Jekyll

In addition to supporting regular HTML content, GitHub Pages supports [Jekyll](https://github.com/jekyll/jekyll), a popular static site generator.

GitHub Pages is deeply integrated with [Jekyll](https://github.com/jekyll/jekyll), a popular static site generator designed for blogging and software documentation, but used for much more. Jekyll makes it easy to create site-wide headers and footers without having to copy them across every page. It also offers [some other advanced templating features](http://jekyllrb.com/docs/templates/).

Although GitHub Pages was designed to work with any static site generator, using Jekyll offers a lot of built-in support and is the only static site generator GitHub officially documents in detail. The main advantages of Jekyll are:

* You can use [Markdown](https://help.github.com/articles/markdown-basics) instead of HTML. Markdown is simpler to read and write.
* You can use common templates, such as headers and footers, that are shared across files.
* You can use a simplified build process to build your site with GitHub Pages.

###### [Customizing GitHub Pages](https://help.github.com/categories/customizing-github-pages) / About GitHub Pages and Jekyll

Top of Form



Bottom of Form

Top of Form

Bottom of Form

### Jekyll's build process

Jekyll's simplified build process with GitHub Pages is one of the biggest advantages of using Jekyll [instead of other static site generators](https://help.github.com/articles/using-a-static-site-generator-other-than-jekyll). GitHub Pages manages your site's build process with a single push to your site's publishing branch. This is Jekyll's build process for managing your site:

1. Push file changes to your pages publishing branch
2. GitHub Pages publishes your site.

**Note:** The publishing branch you use depends on the type of GitHub Pages site you have.

* For [User or Organization pages](https://help.github.com/articles/user-organization-and-project-pages/#user--organization-pages), use the "master" branch.
* For [Project pages](https://help.github.com/articles/user-organization-and-project-pages/#project-pages), use the "gh-pages" branch.

Jekyll is a simple, blog-aware, static site generator perfect for personal, project, or organization sites. Think of it like a file-based CMS, without all the complexity. Jekyll takes your content, renders Markdown and Liquid templates, and spits out a complete, static website ready to be served by Apache, Nginx or another web server. Jekyll is the engine behind [GitHub Pages](https://pages.github.com), which you can use to host sites right from your GitHub repositories.

Jekyll does what you tell it to do — no more, no less. It doesn't try to outsmart users by making bold assumptions, nor does it burden them with needless complexity and configuration. Put simply, Jekyll gets out of your way and allows you to concentrate on what truly matters: your content.

1. Install the Jekyll gem

The Jekyll gem makes a jekyll executable available to you in your Terminal window. You can use this command in a number of ways:

1. $ jekyll build
2. # => The current folder will be generated into ./\_site
3. $ jekyll build --destination <destination>
4. # => The current folder will be generated into <destination>
5. $ jekyll build --source <source> --destination <destination>
6. # => The <source> folder will be generated into <destination>
7. $ jekyll build --watch
8. # => The current folder will be generated into ./\_site,
9. # watched for changes, and regenerated automatically.

The \_config.yml master configuration file contains global configurations and variable definitions that are read once at execution time. Changes made to \_config.yml during automatic regeneration are not loaded until the next execution.

Note [Data Files](https://jekyllrb.com/docs/datafiles) are included and reloaded during automatic regeneration.

##### Destination folders are cleaned on site builds

The contents of <destination> are automatically cleaned, by default, when the site is built. Files or folders that are not created by your site will be removed. Files and folders you wish to retain in <destination> may be specified within the <keep\_files> configuration directive.

Do not use an important location for <destination>; instead, use it as a staging area and copy files from there to your web server.

Jekyll also comes with a built-in development server that will allow you to preview what the generated site will look like in your browser locally.

$ jekyll serve

# => A development server will run at http://localhost:4000/

# Auto-regeneration: enabled. Use `--no-watch` to disable.

$ jekyll serve --detach

# => Same as `jekyll serve` but will detach from the current terminal.

# If you need to kill the server, you can `kill -9 1234` where "1234" is the PID.

# If you cannot find the PID, then do, `ps aux | grep jekyll` and kill the instance. [Read more](http://unixhelp.ed.ac.uk/shell/jobz5.html).

##### Be aware of default behavior

As of version 2.4, the serve command will watch for changes automatically. To disable this, you can use jekyll serve --no-watch, which preserves the old behavior.

$ jekyll serve --no-watch

# => Same as `jekyll serve` but will not watch for changes.

These are just a few of the available [configuration options](https://jekyllrb.com/docs/configuration/). Many configuration options can either be specified as flags on the command line, or alternatively (and more commonly) they can be specified in a \_config.yml file at the root of the source directory. Jekyll will automatically use the options from this file when run. For example, if you place the following lines in your \_config.yml file:

source: \_source

destination: \_deploy

Then the following two commands will be equivalent:

$ jekyll build

$ jekyll build --source \_source --destination \_deploy

For more about the possible configuration options, see the [configuration](https://jekyllrb.com/docs/configuration/) page.

If you’re interested in browsing these docs on-the-go, install the jekyll-docs gem and run jekyll docs in your terminal.

[Back](https://jekyllrb.com/docs/installation/)

[Next](https://jekyllrb.com/docs/structure/)

#### Getting Started

* [Welcome](https://jekyllrb.com/docs/home/)
* [Quick-start guide](https://jekyllrb.com/docs/quickstart/)
* [Installation](https://jekyllrb.com/docs/installation/)
* [Basic Usage](https://jekyllrb.com/docs/usage/)
* [Directory structure](https://jekyllrb.com/docs/structure/)
* [Configuration](https://jekyllrb.com/docs/configuration/)

#### Your Content

* [Front Matter](https://jekyllrb.com/docs/frontmatter/)
* [Writing posts](https://jekyllrb.com/docs/posts/)
* [Working with drafts](https://jekyllrb.com/docs/drafts/)
* [Creating pages](https://jekyllrb.com/docs/pages/)
* [Static Files](https://jekyllrb.com/docs/static-files/)
* [Variables](https://jekyllrb.com/docs/variables/)
* [Collections](https://jekyllrb.com/docs/collections/)
* [Data Files](https://jekyllrb.com/docs/datafiles/)
* [Assets](https://jekyllrb.com/docs/assets/)
* [Blog migrations](https://jekyllrb.com/docs/migrations/)

#### Customization

* [Templates](https://jekyllrb.com/docs/templates/)
* [Permalinks](https://jekyllrb.com/docs/permalinks/)
* [Pagination](https://jekyllrb.com/docs/pagination/)
* [Plugins](https://jekyllrb.com/docs/plugins/)
* [Extras](https://jekyllrb.com/docs/extras/)

#### Deployment

* [GitHub Pages](https://jekyllrb.com/docs/github-pages/)
* [Deployment methods](https://jekyllrb.com/docs/deployment-methods/)
* [Continuous Integration](https://jekyllrb.com/docs/continuous-integration/)

#### Miscellaneous

* [Troubleshooting](https://jekyllrb.com/docs/troubleshooting/)
* [Sites using Jekyll](https://jekyllrb.com/docs/sites/)
* [Resources](https://jekyllrb.com/docs/resources/)
* [Upgrading from 0.x to 2.x](https://jekyllrb.com/docs/upgrading/0-to-2/)
* [Upgrading from 2.x to 3.x](https://jekyllrb.com/docs/upgrading/2-to-3/)

#### Meta

* [Contributing](https://jekyllrb.com/docs/contributing/)
* [Code of Conduct](https://jekyllrb.com/docs/conduct/)
* [History](https://jekyllrb.com/docs/history/)

CNAME

## Used for Using a custom domain with GitHub Pages

You can customize the domain name of your GitHub Pages site.

### [Quick start: Setting up a custom domain](https://help.github.com/articles/quick-start-setting-up-a-custom-domain)

There are three main stages to setting up a custom domain for your GitHub Pages site: choosing your custom domain and registering it with a DNS provider, setting up your pages site repository, and configuring your domain with your DNS provider.

### [About supported custom domains](https://help.github.com/articles/about-supported-custom-domains)

If you're setting up a custom domain for your GitHub Pages site, choose a supported custom domain for the easiest setup and more support. GitHub Pages is designed to work with two types of custom domains: apex domains and subdomains.

### [Custom domain redirects for GitHub Pages sites](https://help.github.com/articles/custom-domain-redirects-for-github-pages-sites)

The type of pages site you're using determines how your site redirects custom domains.

LExers

A lexer is a software program that performs lexical analysis.  Lexical analysis is the process of separating a stream of characters into different words, which in computer science we call 'tokens' .  When you read my answer you are performing the lexical operation of breaking the string of text at the space characters into multiple words.  
  
A parser goes one level further than the lexer and takes the tokens produced by the lexer and tries to determine if proper sentences have been formed.  Parsers work at the grammatical level, lexers work at the word level.

The VMware controller is needed as CT-MG component.  
Therefore provide a controller for VMware services in product level quality.

**Business Motivation**

* VMware vsphere adapter is needed by Uni Wismar, NOW IT, ZF, BT-CIT, EST's internal ESS environment.
* VMware is widely used among Japanese customers, required from almost all opportunities in Japan.
* FJJ is unable to promote FCSM business without VMware adapter.

[UI for Controller configuration settings (with CUSTOM\_BRANDING\_URL)](http://wwwi.est.fujitsu.com/cb/issue/6503)

**Precondition:**  
The APP administrator has to assign a "controller administrator" to each registered controller by specifying

* an organization ID of a technology provider organization, and
* the credentials of a user (uid and pwd) of a technology manager in that organization.  
  This technology manager is the "owner/administrator" of the controller and only this technology manager has access to the controller configuration page.

The controller configuration page is accessible with the URL: *<APPserver>:<APPport>/(color: #ff0000;)oscm-app-<controllerId>*

**Layout:**

The page allows the technology manager to insert/modify:

* General settings: according to [General controller configuration settings](http://wwwi.est.fujitsu.com/cb/issue/6502)http://wwwi.est.fujitsu.com/cb/images/issuetypes/requirement.gif
* Controller specific settings, depending on the IaaS to be integrated.

**Note:**   
**this picture is only a snapshot of the prototype UI! The UI must follow the CT-MG UI guidelines!**