Programming historian en español, 2021.

Creación de sitios estáticos con Jekyll y GitHub Pages.

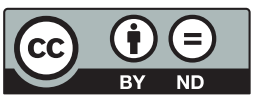
del Rio Riande, Gimena, Hernández, Nidia, de León, Romina y Calarco, Gabriel.

Cita:

del Rio Riande, Gimena, Hernández, Nidia, de León, Romina y Calarco, Gabriel (2021). *Creación de sitios estáticos con Jekyll y GitHub Pages. Programming historian en español,.*

Dirección estable: https://www.aacademica.org/gabriel.calarco/12

ARK: https://n2t.net/ark:/13683/prgr/5DK



Creación de sitios estáticos con Jekyll y GitHub Pages

Amanda Visconti

Esta lección te ayudará a crear un sitio web seguro completamente gratuito, fácil de mantener y sobre el que tendrás control total, como un blog académico, un sitio web de proyectos o un portafolio en línea.

Disponible en: EN (original) | ES

Esta lección es para ti si deseas tener un sitio web de proyectos totalmente gratuito, seguro, fácil de actualizar y de preservar sobre el que tengas control total, como un blog académico, un sitio web de proyecto, o un portfolio en línea.

Al final de esta lección, tendrás un sitio web básico en funcionamiento donde podrás publicar contenido que otras personas podrán visitar -¡se verá así!- y también tendrás algunos recursos para explorar, si deseas personalizar aún más el sitio.

Requisitos: una computadora (Mac/Windows/Linux funcionan, aunque esta lección no cubre algunos aspectos del uso de Linux), que permita la descarga e instalación de software y conexión a Internet que soporte la descarga de software. Según los usuarios, se necesitan entre 1 y 3 horas para completar toda la lección.

Nivel de dificultad: Intermedio (esta lección no incluye el uso de línea de comandos y git, pero te ofrece todo lo necesario para que la completes).

Contenidoscontenidos

- ¿Qué son los sitios estáticos, Jekyll, etc. y por qué deberían importarme?
  - Sitios dinámicos, sitios estáticos y Jekyll
  - GitHub & GitHub Pages
  - ¿Por qué usar sitios estáticos?
- Antes de la instalación
  - Sistemas operativos
  - Cuenta de usuario de GitHub
  - Aplicación GitHub Desktop
  - Editor de texto
  - Línea de comandos
- Instalación de dependencias
  - En Mac
  - Herramientas de línea de comandos
  - Homebrew

    - Ruby y Ruby Gems
    - NodeJS
    - Jekyll
    - En Windows
- Configuración de Jekyll
- Ejecutar un sitio web localmente
    - Mini ayudamemoria
- Modificar la configuración del sitio
    - Configuración básica del sitio con _config.yml
    - ¿Dónde está (y qué es) cada cosa?
- Redacción de páginas y entradas de blog
    - Escritura en Markdown
    - Creación de páginas
    - Creación de entradas
- "Hosting" en GitHub Pages
- Poniéndonos elegantes
    - Diseño visual
    - Funcionalidad
- Guía
- Ayuda, créditos y lecturas
    - Ayuda
    - Creditos
    - Lecturas

¿Qué son los sitios estáticos, Jekyll, etc. y por qué deberían importarme? qué-son-los-sitios-estáticos-jekyll-etc-y-por-qué-deberían-importarme-

Este tutorial se basa en la Documentación oficial de Jekyll escrita por la comunidad de Jekyll. Revisa la sección "Leer más" al final de la lección si deseas profundizar más sobre estos temas.

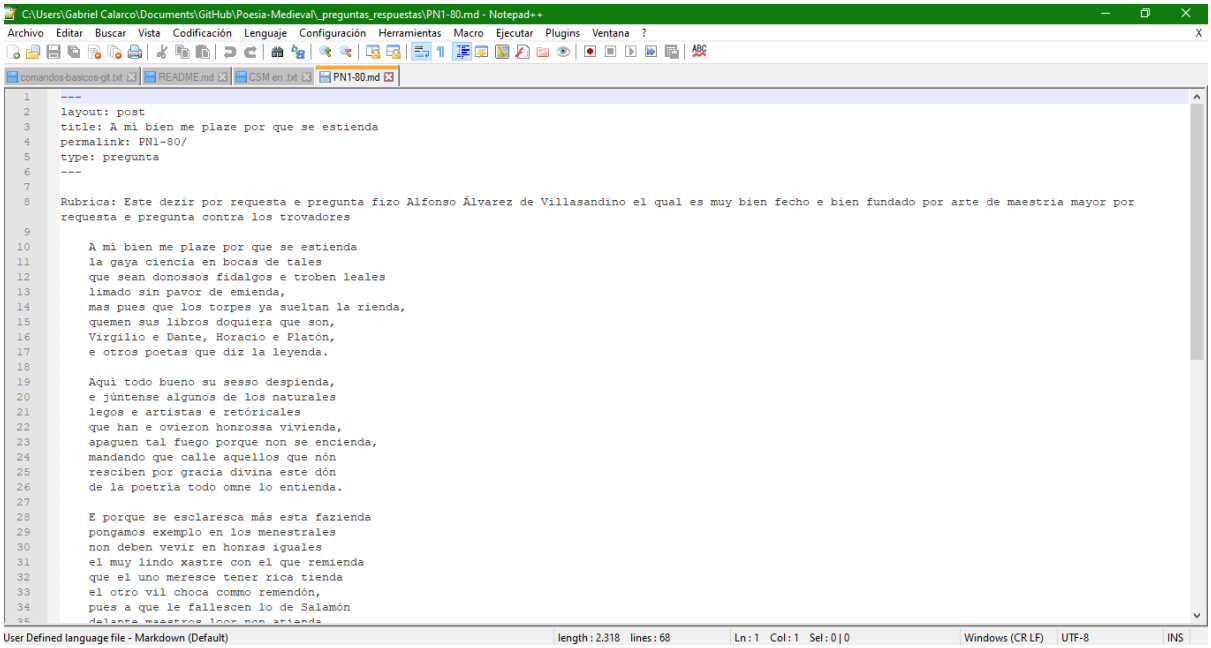
Sitios dinámicos, sitios estáticos y Jekyll sitios-dinámicos-sitios-estáticos-y-jekyll-

Tanto los *sitios web dinámicos*, como los que son creados y administrados por gestores de contenidos tales como Drupal, WordPress y Omeka extraen información de una base de datos para completar el contenido de una página web. Cuando buscamos un libro en Amazon.com, por ejemplo, la página de resultados de búsqueda no existe como una página HTML completa; en cambio, Amazon.com tiene una plantilla para la página de resultados de búsqueda, que incluye elementos que comparten todas las páginas de resultados (como el menú principal y el logotipo de Amazon) y consulta una base de datos para insertar en esa plantilla los resultados de la búsqueda que nosotros realizamos.

Los *sitios web estáticos* no usan una base de datos para almacenar información; por el contrario, toda la información que se muestra en cada página web ya está contenida en tu

correspondiente archivo HTML. Las páginas HTML que componen un sitio estático se pueden escribir completamente a mano o se puede delegar parte de este trabajo usando una herramienta como Jekyll.

Jekyll es un software que nos ayuda a “generar” o crear un *sitio web estático* (Jekyll muchas veces es caracterizado como un “generador de sitios estáticos”). Jekyll utiliza plantillas de página para aquellos elementos como menús principales y pies de página que normalmente se repiten en todas las páginas que componen un sitio y por ende consumen mucho tiempo si escribimos manualmente el HTML para incluirlos en cada página web. Con Jekyll, estas plantillas se combinan con otros archivos con información específica (por ejemplo, un archivo para cada post de un blog) para generar páginas HTML completas para los visitantes de un sitio. Jekyll no necesita consultar bases de datos ni crear una nueva página HTML (o completar una parcial) cuando los usuarios visitan una página web, sino que ya cuenta con las páginas HTML completas y solo las actualiza cuando/si alguna vez cambian.



Código de una página de Jekyll en formato md

Código de la misma página de Jekyll pero en formato html, tal como puede verse en el explorador

Hay que tener en cuenta que cuando alguien se refiere a un “sitio web de Jekyll”, en realidad se refiere a un sitio web estático (HTML puro) que se ha creado utilizando Jekyll. Jekyll es un software que crea sitios web. Jekyll no está “ejecutando” el sitio en vivo, sino que es un “generador de sitios estáticos”: es una herramienta que crea los archivos del sitio estático que luego son alojados en un servidor web, como se hace con cualquier otro sitio web HTML.

Dado que los sitios estáticos no son más que archivos de texto (sin una base de datos que complique las cosas), es posible *versionarlos* fácilmente, es decir, usar una herramienta para llevar un registro de las diferentes versiones del sitio a lo largo del tiempo rastreando los cambios en los archivos de texto que lo componen. El control de versiones es muy útil cuando deseamos combinar ambas versiones (por ejemplo, dos estudiantes escriben una publicación de blog juntos y deseamos combinar sus dos versiones) o cuando queremos comparar archivos para buscar diferencias entre ellos (por ejemplo, “¿Cómo se describía el proyecto en la página ‘Acerca de’ original?”). El control de versiones es muy útil cuando se trabaja en equipo (por ejemplo, permite combinar y rastrear el trabajo de diferentes personas), pero también es útil al crear o ejecutar un sitio web por nuestra propia cuenta.

Puedes leer más acerca de Jekyll o generadores de sitios estáticos (en inglés).

GitHub & GitHub Pages github--github-pages-

GitHub Pages es un espacio gratuito para almacenar los archivos que ejecutan un sitio web y alojar ese sitio para que las personas lo visiten (solo funciona para tipos particulares de sitios web, como sitios HTML básicos o sitios Jekyll; no aloja bases de datos).

GitHub es una plataforma visual para utilizar *git*, un sistema de *versionado* o, en otras palabras, de registro de cambios realizados en los archivos (código y documentos de texto,

entre otros) a través del tiempo (como ya explicamos más arriba). Si tienes curiosidad, puedes explorar este minitutorial de GitHub (en inglés).

¿Por qué usar sitios estáticos?por-qué-usar-sitios-estáticos

Opciones como Drupal, WordPress y Omeka son últiles para la creación de sitios web complejos e interactivos como Amazon o una edición digital interactiva de una novela, pero para muchos blogs, sitios web de proyectos y portafolios en línea, un sitio web estático (como un sitio web creado con Jekyll) puede hacer todo lo que se necesita al mismo tiempo que proporciona algunas ventajas:

- **Mantenimiento:** Las actualizaciones y el mantenimiento necesitan realizarse con mucha menos frecuencia (menos de una vez al año).

- **Preservación:** Al no emplear base de datos, los archivos de texto que componen tu sitio son todo lo que necesitas guardar para preservar y replicarlo. Resulta sencillo hacer un respaldo del sitio o enviarlo a un repositorio institucional.

- **Aprendizaje:** Debido a la ausencia de base de datos y a que no hay una gran cantidad de archivos de código que brinden funciones que quizás ni siquiera necesites, hay muchos menos componentes en tu sitio web; es más fácil revisarlos y saber lo que hace cada uno. Es mucho más fácil convertirse en un usuario de Jekyll básico y avanzado.

- **Más personalización**: Dado que aprender a dominar tu sitio web es fácil, cosas que definitivamente querrás hacer, como cambiar el aspecto (el “tema”) de un sitio creado por Jekyll, resultan mucho más sencillas que alterar el aspecto de un sitio en WordPress o Drupal.

- **Alojamiento gratuito:** Si bien muchas herramientas de sitios web como Drupal, WordPress y Omeka son gratuitas, alojarlas (pagar a alguien para que muestre los archivos de tu sitio web a los visitantes del sitio) puede costar dinero.

- **Control de versiones:** Hospedar en GitHub Pages significa que tu sitio está vinculado a la interfaz visual de GitHub para el control de versiones de git, por lo que puede realizarse un seguimiento de los cambios en tu sitio y, si fuera necesario, volver al estado anterior de cualquier publicación de blog, o página. Esto incluye archivos cargados que tal vez desees almacenar en el sitio, como programas de estudio y publicaciones antiguas (el control de versiones se explicó con más detalle anteriormente).

- **Seguridad:** No hay una base de datos a la que haya que proteger de posibles ataques maliciosos.

- **Velocidad:** Los archivos mínimos del sitio web y la inexistencia de una base de datos para consultar resultan en un tiempo de carga de página más rápida.

La creación de un sitio web estático con Jekyll ofrece aún más ventajas, sin perder las de un sitio web estático HTML codificado a mano:

- **Aprendizaje:** Es más fácil comenzar a personalizar tu sitio y escribir tu contenido, ya que no necesitarás aprender o usar HTML.

- **Creado para bloguear:** Jekyll fue creado para permitir publicaciones de blog, por lo que es fácil bloguear (agregar contenido nuevo, ordenado por fecha) y realizar tareas relacionadas, como mostrar un archivo de todas las publicaciones de blog por mes, o incluir un enlace a las tres publicaciones de blog más recientes al final de cada publicación.

- **La plantilla automatiza las tareas repetitivas:** Jekyll facilita la automatización de las tareas repetitivas del sitio web a través de su sistema de “plantillas”: puedes crear contenido que, por ejemplo, debe aparecer en el encabezado y pie de cada página (por ejemplo, el logotipo o el menú principal), o repetir información en cada publicación de blog (por ejemplo, nombre del autor y fecha de publicación). Esta información de la plantilla se repetirá automáticamente en las páginas web que desees, en lugar de obligarte a reescribir manualmente esa información. Esto no solo ahorra mucho tiempo de copiar y pegar si alguna vez deseas cambiar algo que aparece en cada página de tu sitio web (por ejemplo, un nuevo logotipo o un nuevo elemento en el menú principal), ya que si lo cambias una vez en una plantilla, lo cambiarás en cada lugar que aparece en tu sitio web.

Antes de la instalación antes-de-la-instalación-

¡Estamos listos, manos a la obra! En el resto de esta lección, vamos a instalar algunos programas en nuestras computadoras, usar la línea de comandos para instalar algunas cosas que solo se pueden instalar de esa manera, ver y personalizar una versión privada de tu sitio web y finalmente hacer que tu sitio web sea accesible públicamente en la web. Si tienes problemas en algún momento de esta lección, consulta la sección de ayuda sobre cómo hacer preguntas o informar problemas

En esta sección vamos a asegurarnos de tener todo lo necesario para crear un sitio web estático con Jekyll y GitHub Pages. Para eso, vamos a abordar:

- qué sistema operativo es posible usar (es decir, Mac / Windows / Linux)
- crear una cuenta de GitHub
- por qué es necesario usar un “editor de texto” para trabajar en nuestro sitio web
- cómo usar la línea de comandos

Todos los elementos que vamos a instalar son herramientas de desarrollo web estándar. Se trata de herramientas confiables, por lo que no es indispensable saber exactamente qué hace cada una de ellas. Brindaremos una breve explicación de los elementos que hay que comprender en profundidad y dejaremos enlaces en caso de desear saber más sobre lo que se está instalando.

Sistemas operativos sistemas-operativos-

Este tutorial está destinado a usuarios de Windows y Mac. Jekyll también funciona en Linux; sin embargo, para fines pedagógicos, este tutorial utiliza el software GitHub Desktop (disponible para Windows y Mac únicamente); los usuarios de Linux tienen que usar git para ello ((algo que este tutorial no aborda)).

Jekyll no es oficialmente compatible con Windows, lo que significa que la documentación oficial de Jekyll (las páginas que guían a través de la configuración y que explican su funcionamiento) no aborda el uso de Windows. Este tutorial se basa en las instrucciones de Windows de David Burela para las partes de la sección Instalación de dependencias en las que los usuarios de Windows deben hacer algo diferente; sin embargo, como parte de esta traducción al español hemos revisado el proceso de instalación en Windows.

Cuenta de usuario de GitHub cuenta-de-usuario-de-github-

La cuenta de usuario de GitHub nos permite alojar nuestro sitio web (ponerlo a disposición para que otros lo visiten) de forma gratuita en esa plataforma. Como beneficio adicional, también nos permite llevar un registro de las versiones de nuestro sitio y tu escritura a medida que crece o cambia con el tiempo.

1. Visita GitHub.com y haz clic en el botón verde “Sign up” (Registrarse).
2. En la página siguiente, ingresa el nombre de usuario deseado. El nombre de usuario es visible para otros usuarios, nos identifica en GitHub y también es parte de la URL de nuestro sitio. Por ejemplo, si el nombre de usuario de GitHub es *hdcaicyt*, la URL del sitio será http://hdcaicyt.github.io/. (Ten en cuenta que uno también puede comprar su propio nombre de dominio y usarlo para este sitio, pero eso no se tratará en este tutorial). Escribe una dirección de correo electrónico de uso habitual y añade una contraseña que contenga al menos un número y una letra minúscula.
3. En el recuadro “Verify your account”, presiona el botón “Verify” (Verificar). Usa las flechas para poner la imagen en el sentido correcto. Finalmente, haz clic en “Select a plan” (Seleccionar un plan).
4. En la página siguiente, haz clic en el botón “Choose Free” (Seleccionar gratis).
5. Baja hasta el final de la siguiente página y haz clic en “Complete Setup” (Completar configuración).
6. Ve a tu servicio de email y abre el correo de GitHub (si no aparece en la bandeja de entrada, búscalo en correo no deseado) y haz clic en el botón “Verify email address” (Verificar dirección de email).
7. *Opcional*: puedes visitar https://github.com/settings/profile para agregar un nombre completo (puede ser tu nombre real, nombre de usuario de GitHub u otra cosa) y más información de perfil público, si lo deseas.

Aplicación GitHub Desktop aplicación-github-desktop-

La aplicación GitHub Desktop facilita la actualización del sitio web luego de haberlo configurado. En lugar de usar la línea de comandos cada vez que queramos actualizar nuestro sitio, es posible usar este herramienta visual.

1. Visita el sitio de GitHub Desktop y haz clic en el botón “Download” para descargar GitHub Desktop en tu computadora (Mac y Windows solamente; los usuarios de Linux pueden usar git solo a través de la línea de comandos).
2. Una vez que el archivo se haya descargado, haz doble clic en él y sigue las siguientes instrucciones de instalación.
3. Ingresa el nombre de usuario y la contraseña para la cuenta de GitHub que creaste en el punto anterior y haz clic en “Continuar”.
4. Ingresa el nombre y el correo electrónico a los que deseas asociar tu sitio (probablemente tu nombre público y tu email de trabajo).
5. En la misma página, haz clic en el botón “Instalar herramientas de línea de comandos” e ingresa el nombre de usuario y contraseña de tu computadora, si te lo solicita (luego haz clic en el botón “Instalar ayudante” en el indicador). Cuando recibas un mensaje de que todas las herramientas de línea de comandos se han instalado correctamente, haz clic en continuar.
6. La última página te preguntará “¿Qué repositorios deseas usar?”. Ignórala y haz clic en el botón “Listo”.
7. *Opcional:* puedes hacer el tutorial de uso de GitHub Desktop si lo deseas, pero en esta lección cubriremos todo lo que necesitas saber sobre GitHub).

Editor de texto editor-de-texto-

Es necesario descargar e instalar un editor de texto para realizar pequeñas personalizaciones al código de nuestro sitio Jekyll. Algunas buenas opciones gratuitas incluyen jEdit, Atom, SublimeText, Notepad ++ para Windows o BBedit para Mac. Los procesadores de texto, como Microsoft Word o WordPad, no son una buena opción porque es fácil olvidar cómo formatear y guardar el archivo; es posible agregar accidentalmente formatos y caracteres extra y/o invisibles que pueden generar problemas en el sitio. Por eso es mejor usar programas que puedan guardar lo que escribimos como texto plano (por ejemplo, HTML o Markdown).

Opcional: Consulta la sección “Creación en Markdown” más abajo, para más información sobre un programa de edición específico en Markdown, que también puedes instalar cuando ya estemos en la etapa de crear páginas web y/o publicaciones (posts) de blog.

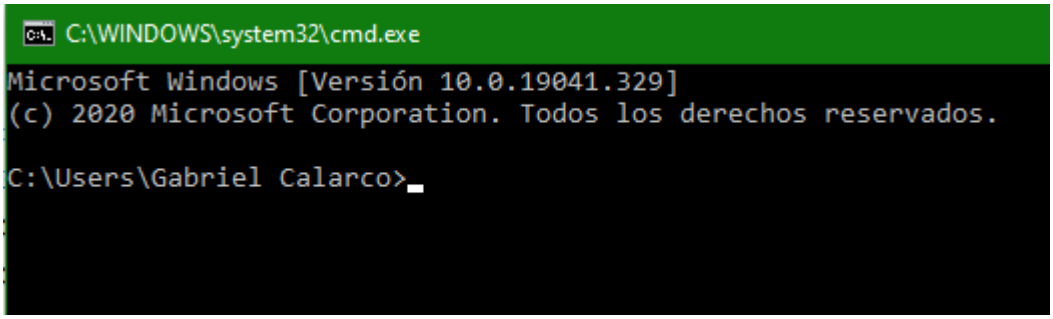
Línea de comandos línea-de-comandos-

La línea de comandos puede ser definida como una forma de interactuar con la computadora mediante texto: permite escribir comandos para llevar a cabo acciones sencillas (como “mostrar una lista de los archivos en este directorio” o “cambiar quién tiene permiso para acceder a este archivo”), así como para realizar acciones más complejas. No obstante, existen buenas alternativas visuales para efectuar acciones en la computadora (por ejemplo, la aplicación GitHub Desktop que instalamos arriba) y otras veces tendremos que usar la línea de comandos para indicarle qué hacer a la computadora. Si deseas más información de la que se proporciona en este tutorial, The Programming Historian tiene una lección que explora en profundidad la línea de comandos, pero aquí cubriremos todo lo necesario para completar la creación de nuestro sitio web y solo usaremos la línea de comandos cuando sea necesario o más sencillo que una interfaz visual.

Mientras que la línea de comandos usa comandos de texto, la mayoría de los usuarios utilizan una “interfaz gráfica de usuario” (también conocida como GUI, “graphical user interface”). Cualquier programa en el que las interacciones usuario-computadora se dan a través de una interfaz visual que contiene íconos, imágenes, funciones de clic con el mouse, etc. es una GUI. ¿Por qué usaríamos la línea de comandos si existen las GUI? Muchas veces es más simple y rápido escribir (o cortar y pegar de un tutorial) una serie de comandos en la línea de comandos que hacer lo mismo usando una GUI. Otras veces, hay cosas para las cuales nadie ha creado una GUI y solo es posible hacerlas a través de la línea de comandos.

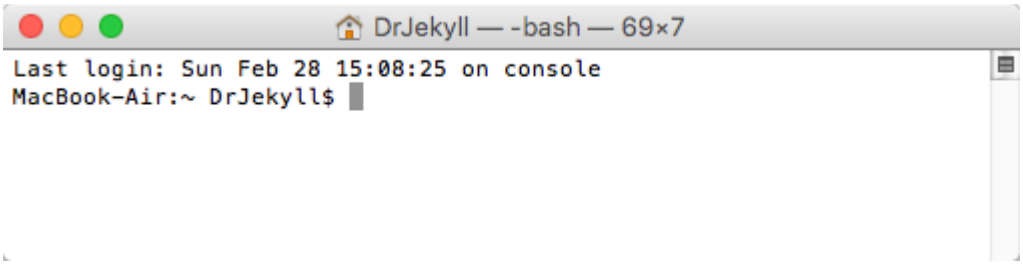
El programa de línea de comandos estándar se llama “Terminal” en Mac (ubicado en *Aplicaciones* > *Utilidades*) y “Cmd” (o “Símbolo del sistema”), “Windows Power Shell” o “Git Bash” en Windows (estas son tres opciones diferentes que difieren cada una en el tipo de comandos que aceptan).

A continuación, se muestra cómo se ve una ventana de línea de comandos en Windows (usando Cmd). Verás algo como *C:\Users\Gabriel Calarco>*. Ese texto se llama “prompt” (“orden” o “solicitud”, porque solicita que el usuario ingrese comandos obligatoriamente para poder continuar). En esta captura de pantalla, *C:* es el nombre de la unidad de disco y *Gabriel Calarco* es la cuenta de usuario actualmente conectada (el prompt será diferente en tu computadora; mostrará tu nombre de usuario).



Prompt de la línea de comandos en Windows.

La línea de comandos en sistemas Unix (Mac y Linux) es ligeramente diferente:

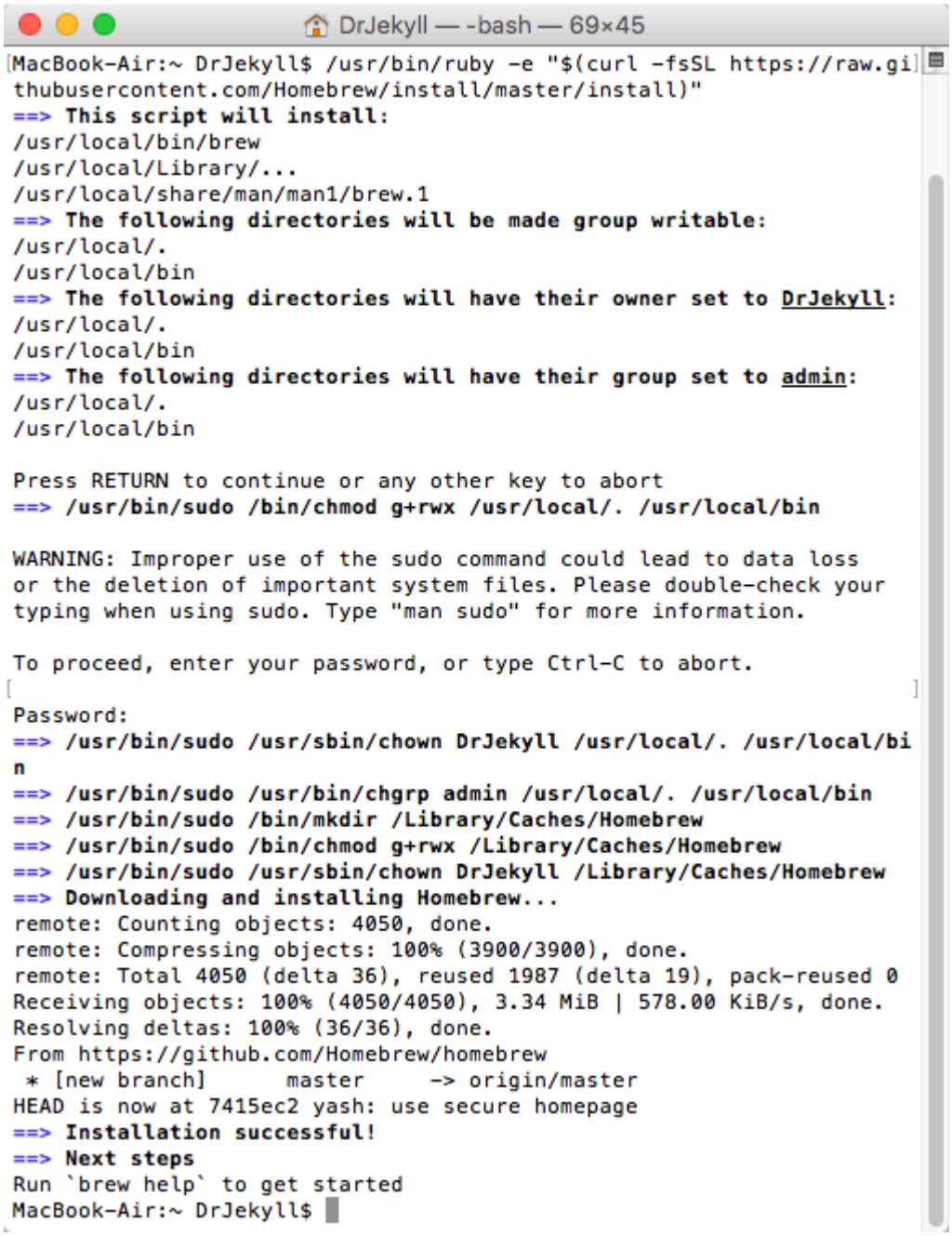


Captura de pantalla de la línea de comandos.

Siempre que en este tutorial pidamos abrir una ventana de línea de comandos e ingresar comandos, ten en cuenta lo siguiente:

1. **Los comandos que debes escribir (o copiar/pegar) en la línea de comandos tienen el siguiente formato:** ejemplo de formato de código. Cada fragmento de código formateado debe copiarse y pegarse en la línea de comandos, seguido de “Enter”.

2. **Debes dejar que los procesos de instalación se ejecuten *completamente* antes de ingresar nuevos comandos.** A veces, escribir un comando y presionar “Enter” produce un resultado instantáneo; otras veces, una gran cantidad de texto comenzará a llenar la ventana de la línea de comandos o parecerá que la línea de comandos no está haciendo nada pero algo está sucediendo detrás de escena, como descargar un archivo. Por eso es importante que **al escribir un comando y presionar Enter, esperemos que ese comando termine por completo *antes de escribir otra cosa***, de lo contrario, podríamos detener un proceso por la mitad y generar problemas. {0}. ¿Cómo saber cuándo se ha completado un comando? Cuando la línea de comandos emite nuevamente el [prompt] (por ejemplo, *Macbook-Air:~DrJekyll$* en la computadora de la autora de este tutorial). La captura de pantalla a continuación muestra un ejemplo de ingreso de un comando, seguido de un texto que muestra lo que estaba sucediendo mientras se procesaba ese comando (y a veces pedía hacer algo, como ingresar la contraseña) y finalmente la reaparición del prompt para hacer saber que ya se puede escribir algo más.



Captura de pantalla de la línea de comandos.

Si necesitamos hacer otra cosa en la línea de comandos y no queremos esperar, podemos abrir una nueva ventana de línea de comandos (en una Mac, presiona ⌘-N o anda a *Shell > Nueva ventana > Nueva ventana con Configuración-Básica*) y trabaja allí mientras esperamos que finalice el proceso en la otra ventana de línea de comandos.

1. Algo muy útil cuando escribimos los mismos comandos muchas veces o queremos recordar algo que escribimos antes: podemos presionar ↑ (flecha hacia arriba) en la

línea de comandos para desplazarnos por los comandos recientemente escritos y presionar “Enter” después de que aparezca el que deseamos usar.

Instalación de dependencias instalación-de-dependencias-

A continuación, vamos a instalar algunas dependencias de software (es decir, programas de los que depende Jekyll para poder trabajar) usando la línea de comandos ya que no hay una interfaz visual para hacerlo. Esta sección se divide en instrucciones para Mac e instrucciones para Windows, así que puedes ir a la sección de instalación de dependencias en Mac si estás usando Mac, o a la sección de instalación de dependencias en Windows si estás usando Windows.

En Mac en-mac-

Si estás utilizando una computadora Mac, sigue las instrucciones que se encuentran a continuación.

Abre una ventana de línea de comandos (*Aplicaciones* > *Utilidades* > *Terminal*) e ingresa el código que se muestra en los pasos a continuación (el código es el texto que aparece formateado así) siguiendo las sugerencias de uso de la línea de comandos detalladas más arriba.

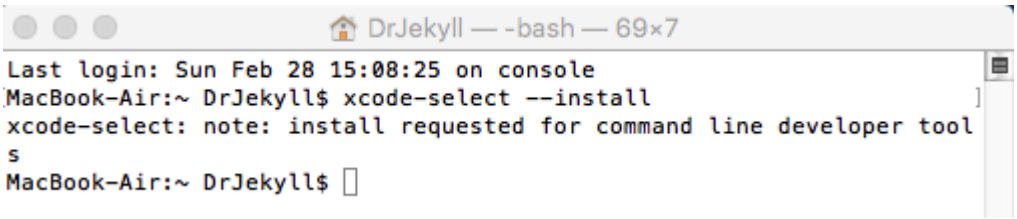
Herramientas de línea de comandos herramientas-de-línea-de-comandos-

Primero vamos a instalar las “herramientas de línea de comandos” de Mac para poder usar Homebrew (que instalaremos a continuación). Homebrew permite descargar e instalar desde la línea de comandos software de código abierto (es un “administrador de paquetes”), lo que facilitará la instalación de Ruby (el lenguaje en el que se basa Jekyll).

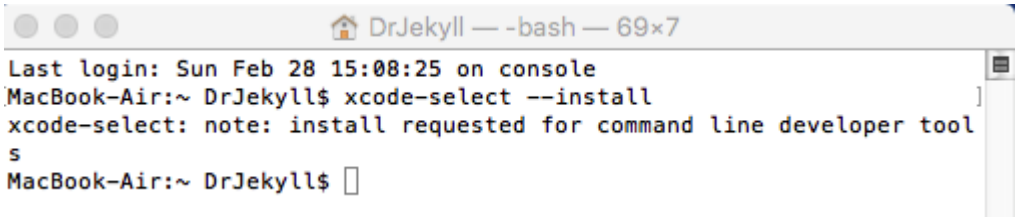
En el Terminal, pega el siguiente código y presiona Enter:

xcode-select --install

Va a aparecer un mensaje como el siguiente, seguido de una ventana emergente:



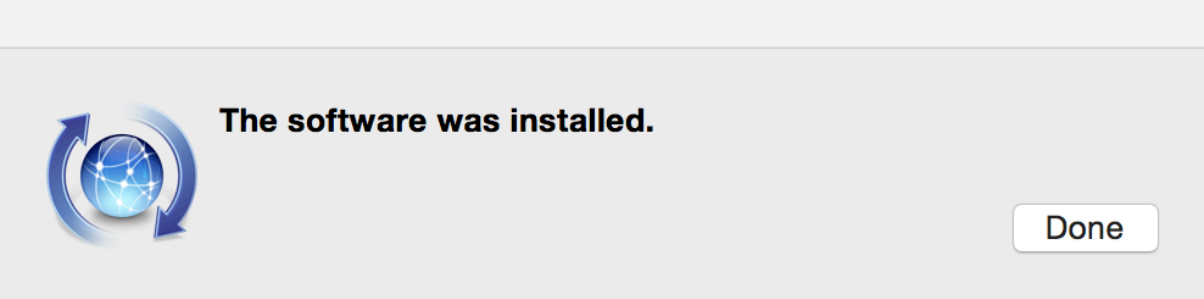
Captura de pantalla de la ventana emergente.



Captura de pantalla de la ventana emergente para instalar.

En la ventana emergente, haz clic en “Install”.

Una vez que termine la instalación, va a aparecer un mensaje de instalación exitosa:



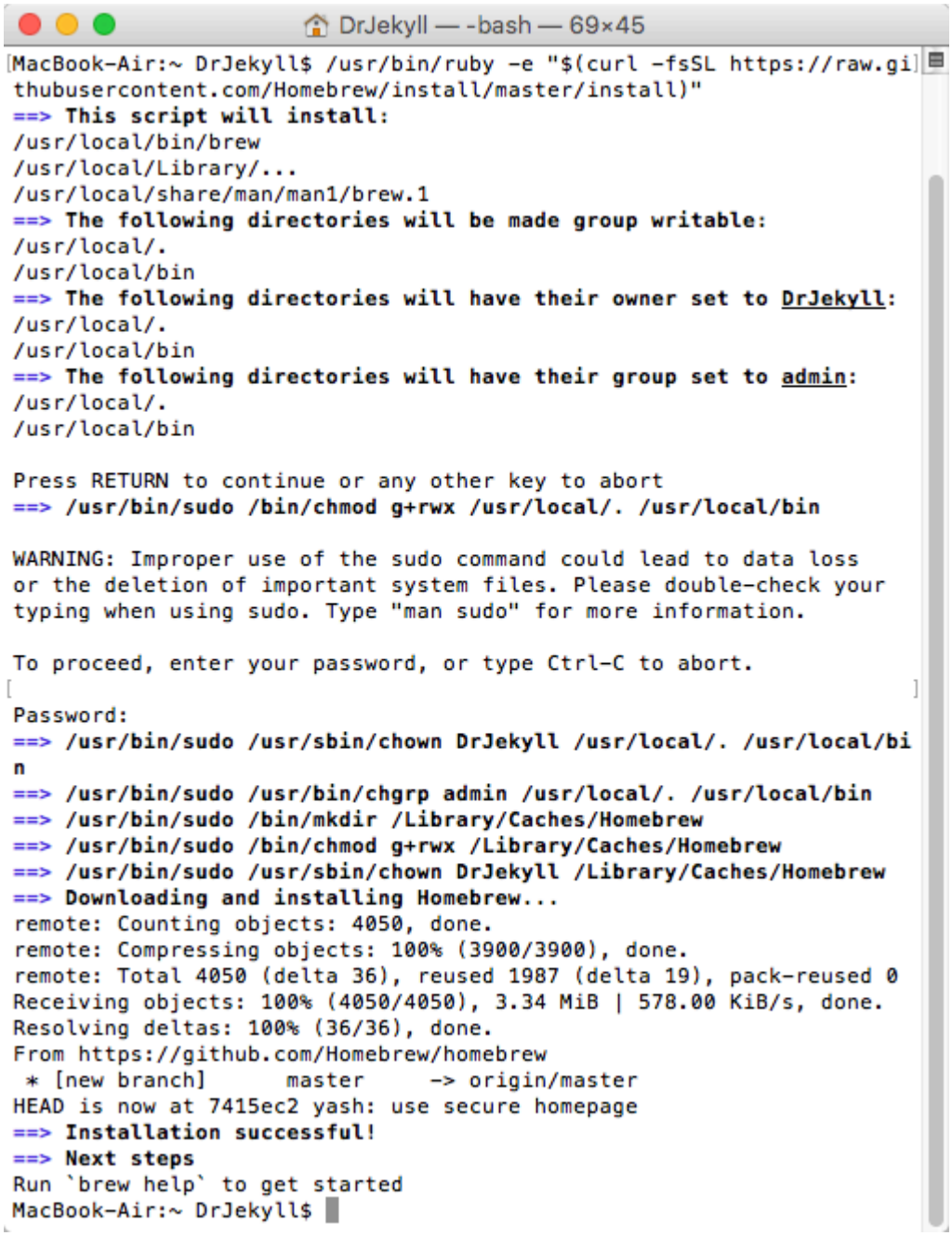
Captura de pantalla de la ventana emergente luego de la instalación exitosa.

Homebrew homebrew-

Al terminar la instalación de las herramientas de la línea de comandos, regresa a la ventana de la línea de comandos y copia el siguiente texto para instalar Homebrew:

/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"

Presiona “Enter” cuando sea necesario, e ingresa la contraseña de tu computadora cuando se solicite. A modo de referencia, mostramos una captura de pantalla del comando ingresado en la línea de comandos de la autora del tutorial, seguido de todo el texto que apareció (incluido el mensaje para presionar “Enter” e ingresar la contraseña).



Captura de pantalla del proceso de instalación de Homebrew.

Ruby y Ruby Gems ruby-y-ruby-gems-

Jekyll está construido a partir del lenguaje de programación Ruby. Ruby Gems es un administrador de paquetes que facilita la configuración de programas Ruby tales como Jekyll (Ruby Gems agrega algunas cosas para simplificar las instalaciones de Ruby).

En la línea de comandos, ingresa:

brew install ruby

Espera hasta que el prompt vuelva a aparecer para ingresar el siguiente comando:

gem install rubygems-update

En caso de recibir un error de permisos luego de instalar Ruby, incluye la dirección de Gems en tu sistema. Prueba lo siguiente:

echo '# Install Ruby Gems to ~/gems' >> ~/.bashrc echo 'export GEM_HOME=$HOME/gems' >> ~/.bashrc echo 'export PATH=$HOME/gems/bin:$PATH' >> ~/.bashrc seguido de source ~/.bashrc.

Algunas personas que utilizan macOS Catalina y macOS Big Sur han reportado dificultades para instalar Ruby y Ruby Gems. Si bien esta lección fue publicada antes de la aparición de esos sistemas operativos, el código compartido aquí ha sido adaptado para ofrecer una posible solución. Si sigues encontrando problemas, estas indicaciones (en inglés) podrían ser de utilidad.

NodeJS nodejs-

NodeJS (o Node.js) es una plataforma de desarrollo (específicamente, es un “entorno de ejecución”) que, por ejemplo, ayuda a que Javascript se ejecute más rápido.

En la línea de comandos, ingresa:

brew install node

Jekyll jekyll-

Jekyll es el programa que crea nuestro sitio web, simplificando ciertas tareas comunes, como usar la misma plantilla (el mismo logotipo, menú, información del autor, etc.) en todas las páginas de nuestro blog. Puedes ver más información sobre Jekyll en Sitios dinámicos, sitios estáticos & Jekyll y en ¿Por qué usar sitios estáticos?, más arriba.

En la línea de comandos, ingresa:

gem install jekyll

¡Felicitaciones, hemos terminado de instalar todo lo necesario para crear nuestro sitio web! Omite los siguientes pasos (que son solo para usuarios de Windows).

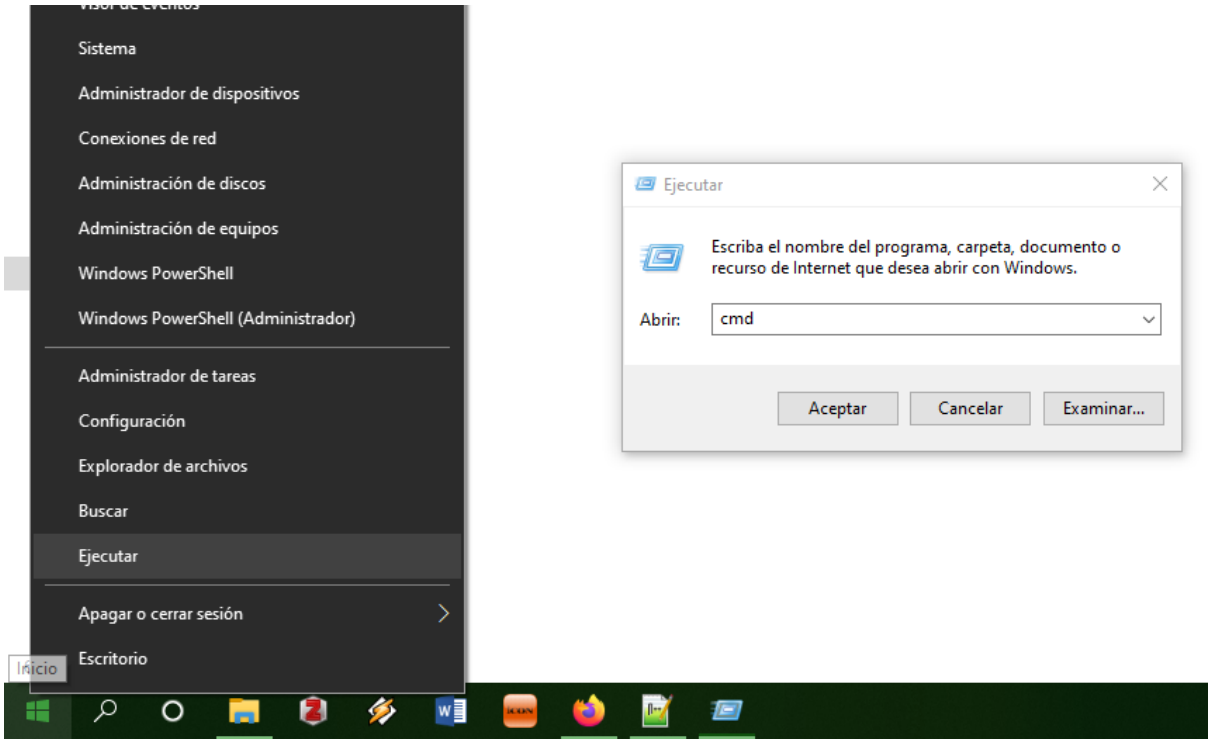
En Windows en-windows-

En esta sección, las instrucciones para usuarios de Windows difieren de las de los usuarios de Mac. Debes hacer estos pasos únicamente si estás utilizando Windows.

1. Antes que nada, necesitamos una herramienta de línea de comandos que reconozca los mismos comandos que las computadoras Mac y Linux (es decir, los sistemas operativos Unix). Visita https://git-scm.com/downloads y haz clic en el enlace Windows. Una vez que hayas terminado la descarga, haz doble clic en el archivo descargado y sigue los pasos para instalar Git Bash (deja todas las opciones como

están).

2. Abre Cmd (abre el Menú de inicio y busca “Cmd” y aparecerá una aplicación que puedes abrir). Otra forma sencilla de abrir el Cmd en Windows, es hacer clic derecho sobre el botón de inicio que se encuentra en la barra de tareas y seleccionar la opción “Ejecutar”, lo que abrirá una ventana emergente en la que debes escribir “cmd” y presionar “Aceptar”.



Ejecutar Cmd.

1. En primer lugar debes instalar Ruby. Ve a https://rubyinstaller.org/downloads/ y descarga la versión más completa, que es Ruby+Devkit 2.6.6-1 (x64) (la tercera de las opciones de la columna WITH DEVKIT). Una vez instalado, la consola se abrirá automáticamente y te pedirá que indiques qué componentes deseas instalar, presiona “Enter” para instalar todo. Este proceso se repetirá dos veces. La segunda vez la consola se cerrará automáticamente.

Para comprobar que la instalación de Ruby se realizó correctamente, vuelve a abrir la consola y escribe:

ruby -v

1. A continaución instalaremos Jekyll a través de la consola, para eso debes escribir los siguientes comandos, uno por uno, haciendo “Enter” y esperando hasta que se descarguen todas las “gemas” de Jekyll. El primer comando a ejecutar es:

gem install jekyll

Este proceso puede demorar un buen rato. La última frase que debe aparecer en la consola es “26 gems installed”. Una vez que la instalación se haya completado volverá a aparecer el prompt.

1. Finalmente, para comprobar que Jekyll se haya instalado correctamente, escribe el siguiente comando y presiona “Enter”:

```
jekyll -v
```

Si recibes la respuesta “jekyll x.x.x” (la última versión de Jekyll actualmente es la 4.1.1, pero este número puede variar), es que Jekyll se instaló correctamente.

¡Felicitaciones, hemos terminado de instalar todo lo necesario para crear nuestro sitio web! De aquí en adelante, las instrucciones son iguales para Windows y Mac.

Configuración de Jekyll configuración-de-jekyll-

Ya hemos instalado todo lo necesario para crear un sitio web. En esta sección utilizaremos Jekyll para generar una nueva carpeta con los archivos que conforman el sitio web. También ubicaremos esta carpeta en un lugar accesible para la aplicación GitHub Desktop para que estén en el lugar correcto cuando deseemos publicarlos como un sitio web público más adelante en la lección.

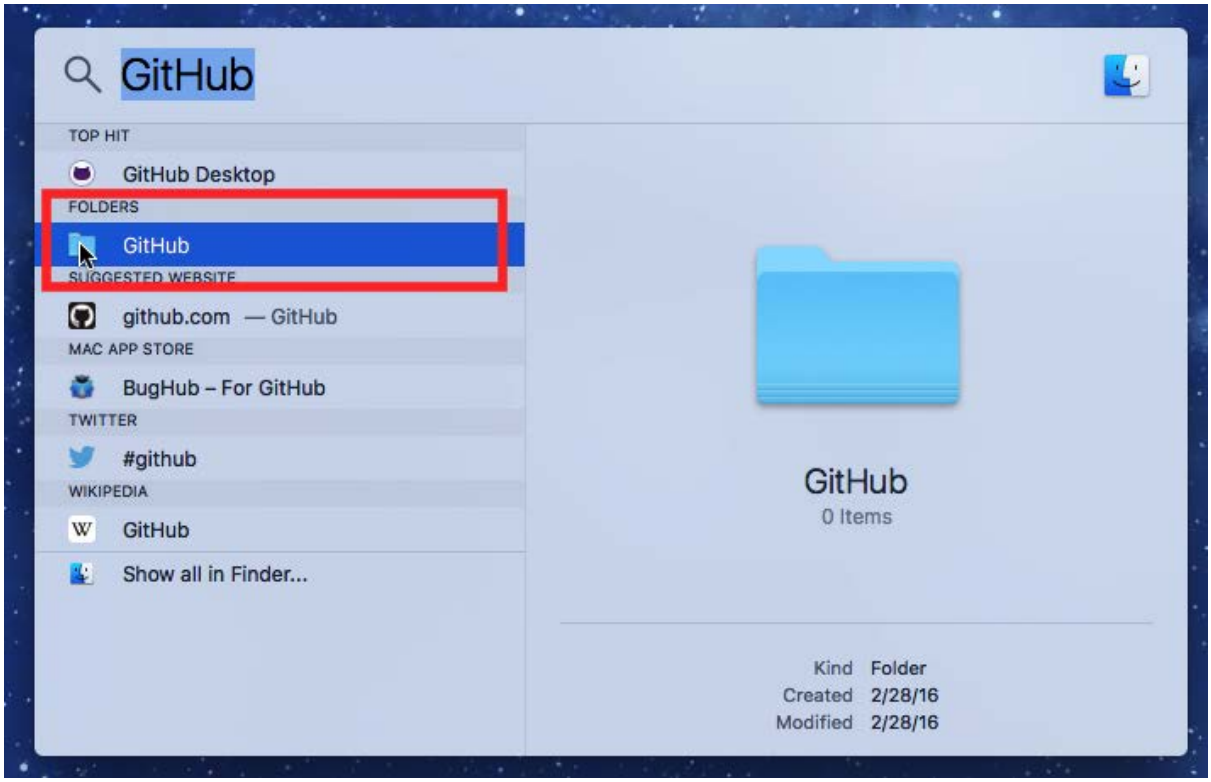
1. Es necesario conocer la ruta de la carpeta GitHub creada por la instalación de GitHub Desktop (la ruta es el texto que indica la ubicación de cierta carpeta o archivo en el árbol de carpetas de una computadora, por ejemplo /Desktop/MyRecipes/Spaghetti.doc). Si no conoces la ruta de la carpeta GitHub, haz clic en el campo de búsqueda del Menú de inicio (en Windows).

<img src="/images/building-static-sites-with-jekyll-github-pages/sitios-estaticos-con-jekyll-y-github-pages_5.png" alt="En Windows puedes encontrar la carpeta con los archivos de tu repositorio de GitHub seleccionando la pestaña Repository del menú superior de GitHub Desktop y en el recuadro que se despliega al seleccionar "show in explorer".">

En Windows puedes encontrar la carpeta con los archivos de tu repositorio de GitHub seleccionando la pestaña Repository del menú superior de GitHub Desktop y en el recuadro que se despliega al seleccionar “show in explorer”.

En Mac, presiona “⌘ + espacio” y aparecerá un cuadro de búsqueda en el medio de la pantalla; escribe “GitHub”, luego haz doble clic en la opción “GitHub” que aparece debajo de “Folders” (Carpetas) para abrir la carpeta GitHub en Finder.

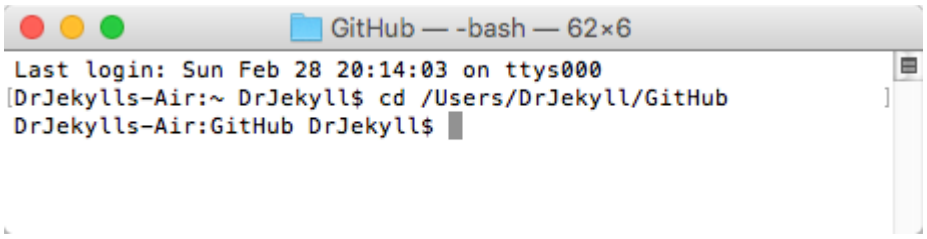
Ten en cuenta que en algunas computadoras esta carpeta está etiquetada como “GitHub para Mac” y puede que no aparezca en una búsqueda. Si los pasos anteriores no ubicaron una carpeta de GitHub, ve a Biblioteca > Soporte de aplicaciones en Finder y verifica si la carpeta “GitHub para Mac” está allí.



Captura de pantalla de la carpeta GitHub.

Haz clic derecho en la carpeta “GitHub” y elige “Copiar GitHub”. La ruta de la carpeta GitHub ha sido copiada.

1. En la línea de comandos, escribe cd, seguido de espacio, seguido de la ruta a la carpeta GitHub (⌘-v para pegar la ruta copiada en el paso previo). En la computadora de la autora (logeada como *DrJekyll*), esto se ve de la siguiente manera:



Captura de pantalla de la linea de comandos.

El comando *cd* (**c**hange **d**irectory) le indica a la computadora que ubique la ruta indicada, en este caso, la ruta a la carpeta GitHub creada por la instalación de GitHub Desktop.

1. En la línea de comandos, escribe el siguiente comando seguido de “Enter”:

   gem install jekyll bundler

Es necesario esperar a que vuelva a aparecer el prompt para continuar con el siguiente paso.

2. La URL pública de tu sitio tendrá la siguiente forma: http://amandavisconti.github.io/JekyllDemo/ (*amandavisconti* es el usuario de GitHub de la autora y *JekyllDemo* el nombre del sitio que ingresamos en este paso (*es posible pagar y usar tu propia URL personalizada, pero no lo cubriremos en este tutorial*). **Los sitios en mayúsculas y minúsculas *no* dirigen al mismo sitio web**, así que a diferencia del ejemplo **JekyllDemo** es recomendable elegir un nombre todo en minúsculas para asegurarse de que la gente lo escriba correctamente.

   En la línea de comandos, escribe lo siguiente (reemplaza *JekyllDemo* con el nombre que desees para tu sitio):

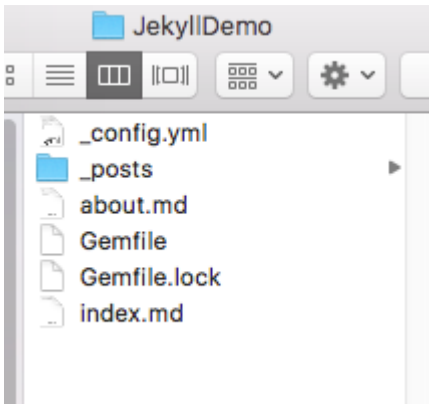
   jekyll new JekyllDemo

   Este comando le indica a *jekyll* que cree un *nuevo* (new) sitio instalando los archivos necesarios en la carpeta llamada *JekyllDemo*. **De ahora en adelante, llamaremos "carpeta del sitio web" a la carpeta creada en este paso (por ej., *JekyllDemo*).**

3. En la línea de comandos, escribe lo siguiente para ir a la carpeta del sitio web (en el resto del tutorial, reemplaza *JekyllDemo* por el nombre elegido en el paso previo):

   cd JekyllDemo

   Si miras en la carpeta *GitHub* > *JekyllDemo* en el explorador de archivos, verás una serie de archivos nuevos -los archivos que ejecutarán tu sitio web- que han sido instalados (más abajo explicaremos qué hace cada uno):



Captura de pantalla de la carpeta creada.

Ejecutar un sitio web localmente ejecutar-un-sitio-web-localmente-

Esta sección describe cómo ejecutar un sitio web **localmente**. Esto significa que podrás ver cómo se ve tu sitio web en un navegador, pero únicamente en tu computadora (a eso se refiere lo de “localmente”). Trabajar en una versión local de un sitio web quiere decir que el sitio es privado, nadie puede verlo todavía (el sitio no es público, nadie puede escribir la URL y verlo en su computadora).

Esto te permite experimentar todo lo que desees y publicar el sitio al mundo cuando esté listo. Además, una vez que el sitio esté publicado, puedes seguir experimentando localmente con nuevos contenidos, diseños, etc. y agregar estos cambios una vez que estés conforme con cómo se ven en el sitio local.

1. En la línea de comandos, escribe

   bundle exec jekyll serve --watch

   Este es el comando que debes ejecutar cada vez que quieras visualizar tu sitio localmente.

   jekyll serve le indica a la computadora que ejecute Jekyll localmente.

   –*watch* precedido de *bundle exec* le indica a Jekyll que busque cambios en los archivos del sitio web (por ejemplo, nuevos posts o páginas) y que los muestre al actualizar el navegador. **Una excepción** es el archivo _config.yml, que será explicado en detalle en la próxima sección (los cambios realizados en este archivo solo se muestran luego de detener y reiniciar Jekyll).

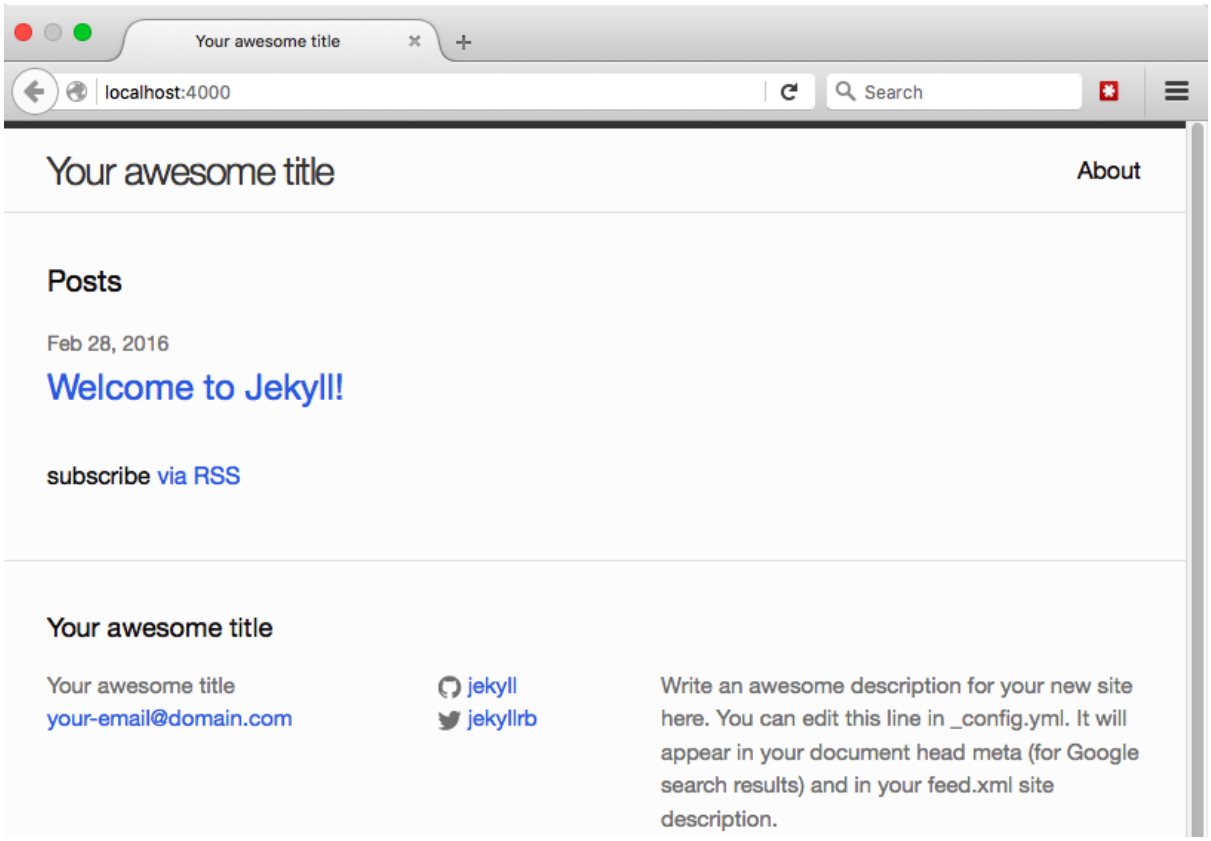
2. Luego de escribir el comando previo, aparecerá en el terminal un proceso que no se detiene. ¿Recuerdas que te contamos que si escribías algo en la línea de comandos mientras este todavía está ejecutando el comando previo se pueden ocasionar problemas? Ahora Jekyll está corriendo en esta línea de comandos, de manera que si deseas ejecutar comandos mientras visualizas tu sitio local, deberás abrir una nueva ventana de línea de comandos (ver la sección acerca del uso de la línea de comandos)

```
Amandas-MacBook-Air:GitHubGeneral amanda$ cd JekyllDemo
Amandas-MacBook-Air:JekyllDemo amanda$ bundle exec jekyll serve --watch
Configuration file: /Volumes/Vault101/Documents/GitHubGeneral/JekyllDemo/_config.yml
Configuration file: /Volumes/Vault101/Documents/GitHubGeneral/JekyllDemo/_config.yml
            Source: /Volumes/Vault101/Documents/GitHubGeneral/JekyllDemo
       Destination: /Volumes/Vault101/Documents/GitHubGeneral/JekyllDemo/_site
 Incremental build: disabled. Enable with --incremental
      Generating...
                    done in 0.658 seconds.
 Auto-regeneration: enabled for '/Volumes/Vault101/Documents/GitHubGeneral/JekyllDemo'
Configuration file: /Volumes/Vault101/Documents/GitHubGeneral/JekyllDemo/_config.yml
    Server address: http://127.0.0.1:4000/
  Server running... press ctrl-c to stop.
```

Captura de pantalla de el terminal ejecutando localmente el sitio.

1. Para detener la ejecución local de nuestro sitio, debemos presionar **control-c** (esto libera la línea de comandos para usarla nuevamente). Basta con ingresar bundle exec jekyll serve --watch nuevamente para volver a ejecutar el sitio localmente.

2. Para visualizar el sitio que se está ejecutando localmente, visita **localhost:4000** en tu navegador. Verás un sitio web Jekyll básico con texto predefinido:



Vista del sitio web en el navegador

Mini ayudamemoria mini-ayudamemoria-

- Escribe bundle exec jekyll serve --watch en la línea de comandos para ejecutar el sitio web localmente. Visita **localhost:4000** en un navegador para visualizar el sitio localmente. En la próxima sección haremos modificaciones que nos obligarán a visitar **localhost:4000/JekyllDemo/** para poder visualizar el sitio (ingresando el nombre de la carpeta de tu sitio web en lugar de *JekyllDemo* y asegurándote de incluir la barra final */*).

- Presiona **control-c** en la línea de comandos para detener la ejecución local del sitio web.

- Cuando hagas cambios en los archivos del sitio web mientras este se está ejecutando, asegúrate de guardar los archivos y refrescar la página (F5 o ⌘+R) en el

navegador para poder ver las modificaciones. Sin embargo, si realizas cambios en _config.yml, deberás detener la ejecución del sitio y reiniciarla para poder verlos aplicados.

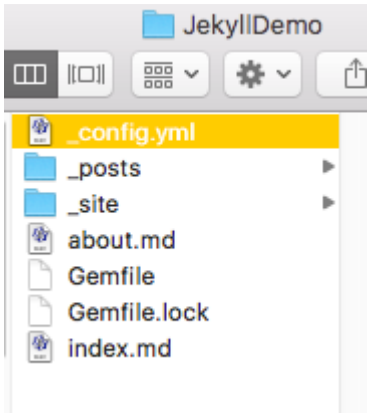
- ¿Escribes, copias o pegas mucho bundle exec jekyll serve --watch? Puedes presionar la tecla ↑ (flecha hacia arriba) en la línea de comandos para hacer desfilar los comandos ingresados recientemente. Presiona “Enter” cuando aparezca el comando que deseas ejecutar.

Modificar la configuración del sitio modificar-la-configuración-del-sitio-

Ya tenemos un sitio web básico privado, accesible únicamente en nuestra computadora. En esta sección, vamos a personalizar el sitio cambiando el título y el autor. También vamos a dar un panorama de lo que hacen los diferentes archivos del sitio web.

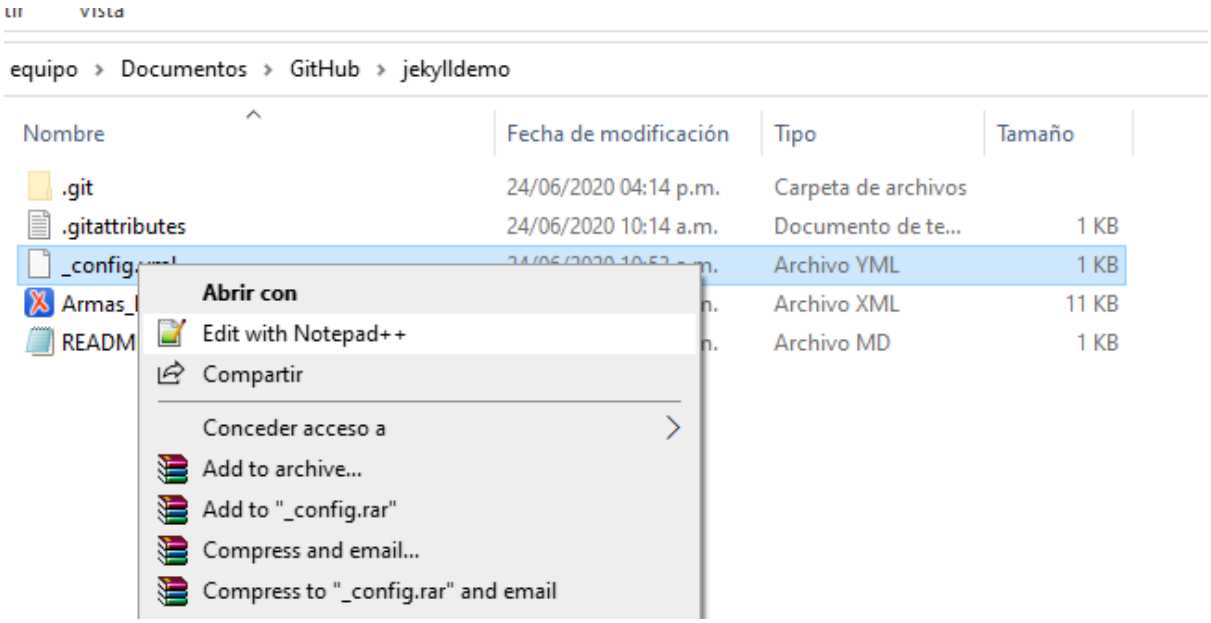
Configuración básica del sitio con _config.yml configuración-básica-del-sitio-con-_configyml-

1. Abre la carpeta de tu sitio web en el explorador de archivos. El sitio de la autora del tutorial se encuentra en /Users/DrJekyll/GitHub/JekyllDemo (*DrJekyll* es el nombre de usuario de la autora y *JekyllDemo* es el nombre de la carpeta del sitio web de este tutorial). Visita la sección “Configuración de Jekyll” si necesitas ayuda para encontrar la carpeta de tu sitio web.

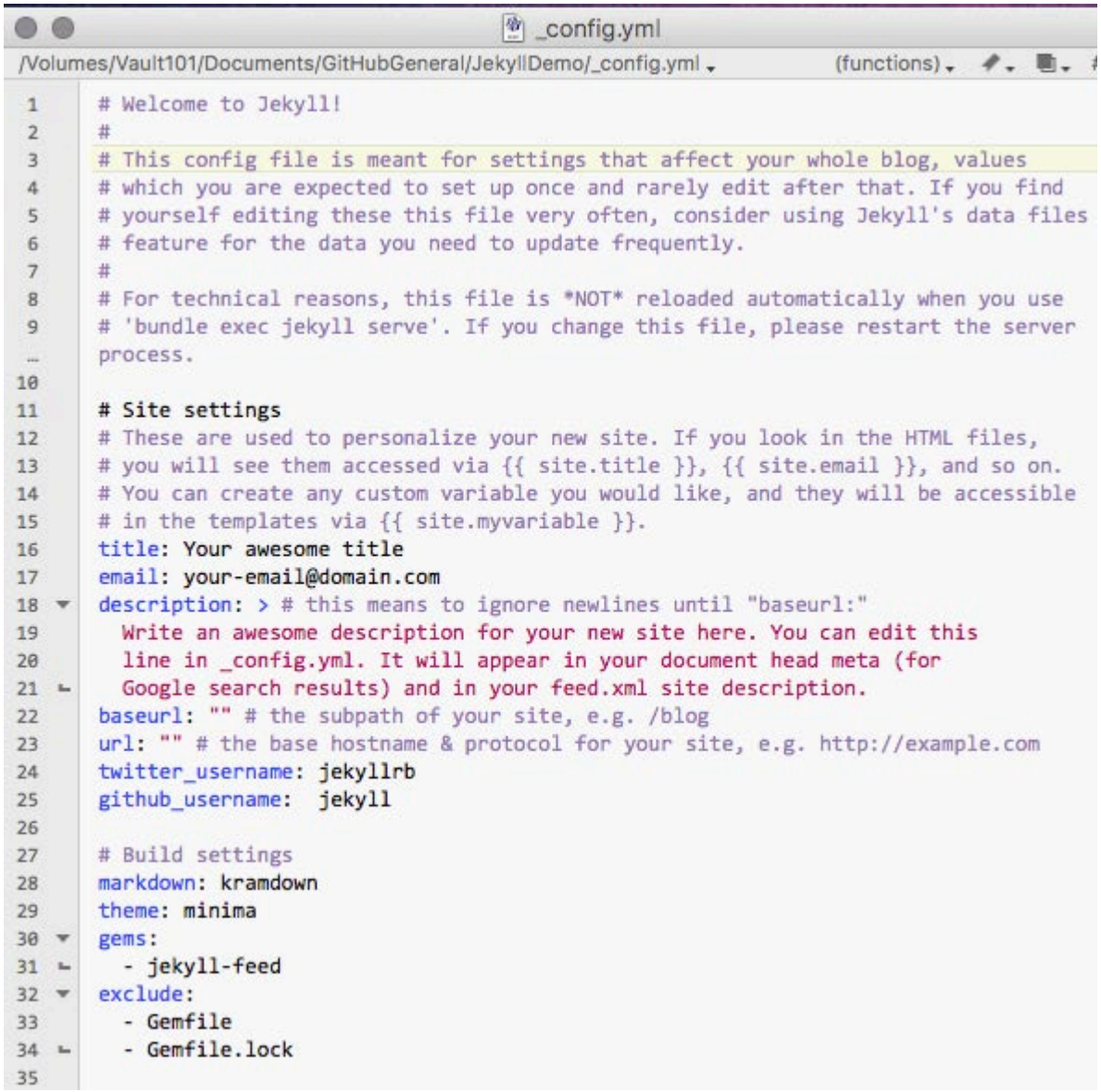


Captura de pantalla de la carpeta que contiene los archivos del sitio web.

1. Comenzaremos por personalizar el archivo de configuración principal _config.yml. Deberás abrir este archivo y los demás archivos del sitio web usando un editor de texto (por ej., Notepad++ en Windows o BBedit en Mac).



En Windows, al hacer clic derecho sobre el archivo .yml puede aparecer directamente la opción de editar el documento con Notepad++; en caso contrario debe elegirse la opción abrir con y seleccionar el editor de texto de una lista de programas.



El archivo _config.yml

El archivo _config.yml es un archivo “destinado a configuraciones que afectan a todo tu blog, valores que se espera que configures una sola vez y rara vez necesites volver editar más tarde” (como dice en el archivo). _config.yml es donde se puede definir el nombre del sitio web y compartir información como la dirección de email que queremos asociar al sitio y otras configuraciones básicas que desees que estén disponibles en todo el sitio web (cuentas de redes sociales, por ejemplo).

La extensión .yml refiere a cómo fue escrito el archivo usando YAML (acrónimo de *YAML Ain’t Markup Language*, “YAML no es un lenguaje de marcado”). YAML es un modo de escribir datos que es a la vez fácil de escribir y de leer para los humanos y fácil de interpretar para las máquinas. No es necesario profundizar en YAML aquí, pero es importante dejar el formato del archivo _config.yml en su estado original aunque modifiquemos el contenido (por ej., el título del sitio debe quedar en una línea diferente del email).

1. Puedes cambiar el texto en este archivo, guardarlo y luego ver tu sitio web local en un navegador para ver los cambios. **Es necesario tener en cuenta que los cambios en _config.yml**, a diferencia del resto de los archivos del sitio web, no se mostrarán si se realizan mientras el sitio web se está ejecutando. Para ver los cambios implementados en este archivo en particular, debes realizarlos mientras el sitio web no se está ejecutando *o* después de realizar cambios en _config.yml, detener y luego ejecutar el sitio de vuelta. (Los cambios en el archivo _config.yml quedan fuera de la capacidad de refrescar el sitio porque este archivo se puede usar para declarar la estructura de los enlaces del sitio y alterarlos mientras el sitio se está ejecutando podría provocar daños.)

   Si se produce algún error, hacer pequeños cambios en los archivos del sitio (comenzar por uno solo), guardar y refrescar el navegador para visualizar los efectos en el sitio permite identificar la causa con más claridad.

   - Las líneas que comienzan con el signo # son *comentarios*: los comentarios no son interpretados como código sino que sirven para dejar notas sobre cómo hacer algo o sobre modificaciones realizadas en el código.

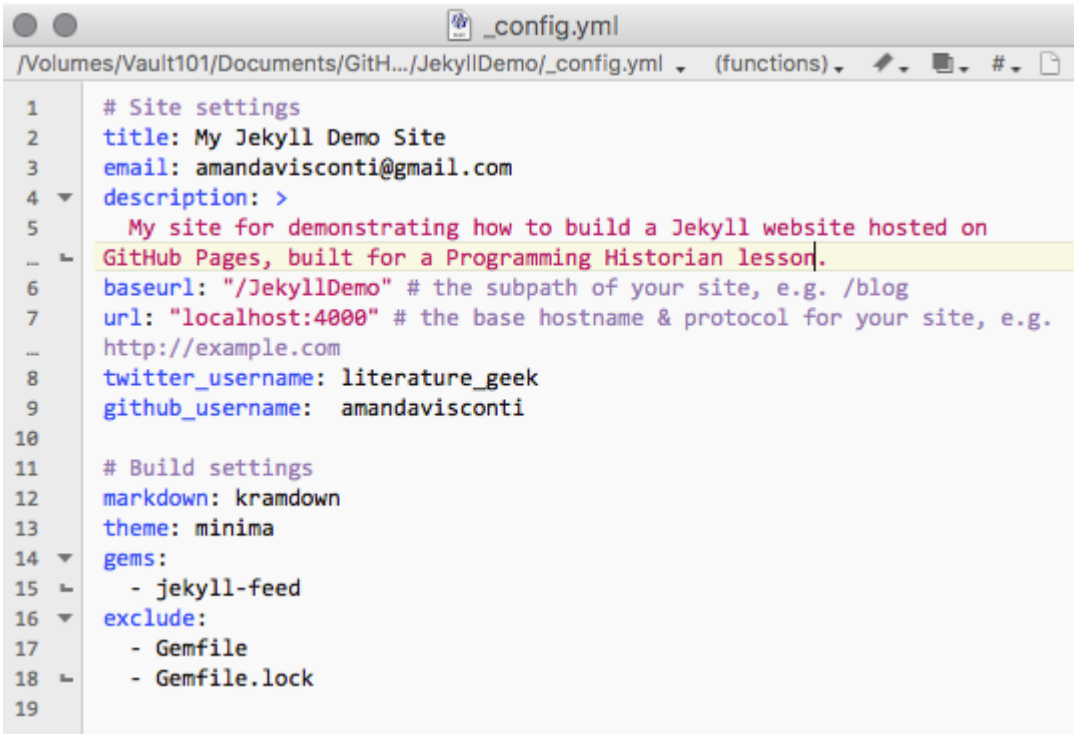
   - Es posible borrar los comentarios sin consecuencias para el sitio web (por ejemplo, puedes borrar las líneas comentadas 1-6 en _config.yml si no deseas ver esa información acerca del uso de Jekyll).

2. Modifica el archivo _config.yml siguiendo estas instrucciones:

   - **title**: el título de tu sitio web como deseas que aparezca en el encabezado del sitio web.
   - **email**: tu dirección de email.
   - **description**: la descripción del sitio web que será usada por los motores de búsqueda y que será utilizada por RSS.
   - **baseurl**: completa entre las comillas con una barra oblicua ***/*** seguida del nombre de la carpeta de tu sitio web (por ej., “/JekyllDemo”) para que el sitio tome la URL correcta.
   - **url**: reemplaza “http://yourdomain.com” por “localhost:4000” para que el navegador tome la versión local de tu sitio en la URL correcta.
   - **twitter_username**: tu nombre de usuario de Twitter (no incluir @).
   - **github_username**: tu nombre de usuario de GitHub.
3. Los cambios realizados en las líneas *baseurl* y *url* permitirán que tu sitio se ejecute desde los mismos archivos, tanto localmente en tu computadora como en vivo en la web, pero **al hacer esto ha cambiado la URL en donde puedes visualizar tu sitio localmente**. De ahora en adelante, en lugar de ser localhost:4000, es localhost:4000/JekyllDemo/ (debes cambiar *JekyllDemo* por el nombre de la carpeta de tu sitio web y no olvides incluir la última barra oblicua ***/***)

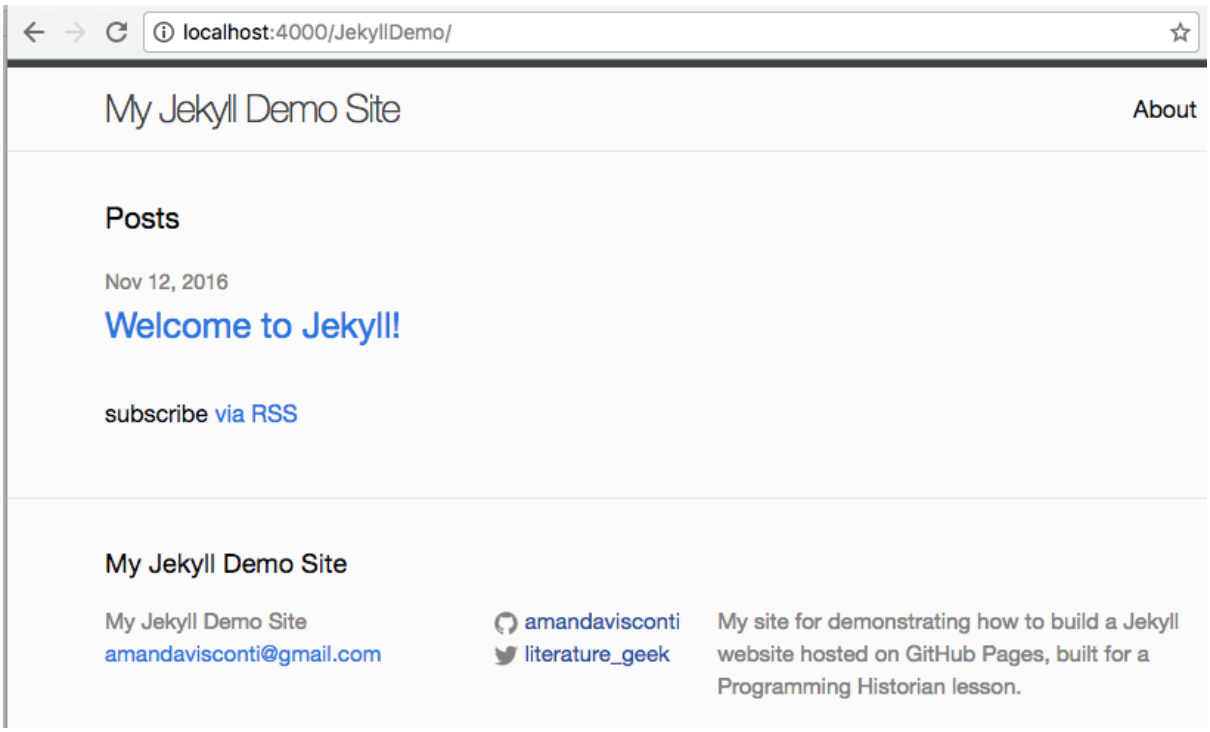
   En la captura de pantalla que está debajo, la autora ha borrado las líneas comentadas 1-6 y el comentario que explicaba lo que hace “description” (no es obligatorio, es solo para demostrar que es posible borrar los comentarios). También

modificó el resto del archivo según los cambios antes mencionados:



El archivo _config.yml modificado

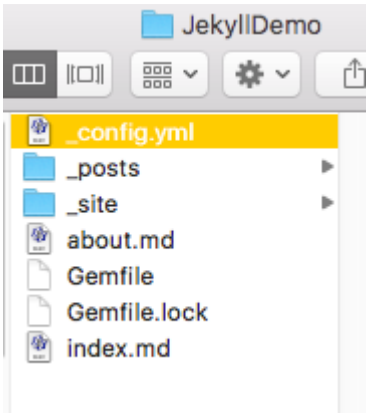
1. Guarda el archivo y ejecuta el sitio web (o detenlo y vuelve a ejecutarlo si estaba en ejecución) y luego visita localhost:4000/JekyllDemo/ en tu navegador (cambiando *JekyllDemo* por el nombre de la carpeta de tu sitio web e incluyendo la barra oblicua final) para ver localmente los cambios en tu sitio:



Ejecución local del sitio web

¿Dónde está (y qué es) cada cosa? dónde-está-y-qué-es-cada-cosa-

Para tener una idea de cómo funciona el sitio y con qué archivos se puede experimentar para hacer cosas más avanzadas, aquí hay algunas notas sobre lo que hace cada carpeta o archivo de tu sitio web. Recuerda siempre abrir y editar cualquier archivo con un editor de texto (por ejemplo, Notepad++) y no con un procesador de textos (no utilices Microsoft Word ni nada que permita agregar formato como cursiva y negrita). Es muy importante no usar Word o procesadores de texto porque estos programas agregan caracteres invisibles que si se guardan en los archivos de nuestro sitio web pueden dañarlo. Si ya deseas comenzar a agregar contenido a tu sitio y hacerlo público, puedes saltar a la siguiente sección.



Carpeta con los archivos de nuestro sitio

- **_config.yml** fue explicado más arriba; contiene información básica de la configuración del sitio, como el título y otras posibilidades que no abordaremos aquí (por ej., cómo estructurar los links)
- la carpeta **_includes** contiene archivos que son incluidos en todas o varias páginas (por ej., el código para que el encabezado del sitio tenga el título y el menú principal en todas las páginas del sitio)
- la carpeta **_layouts** contiene código que controla cómo se ven las páginas de nuestro sitio web (default.html), así como también modificaciones de ese código para darle un estilo más específico a las entradas (post.html) y las páginas (page.html)
- la carpeta **_posts** contiene los archivos que representan cada una de las entradas de nuestro sitio web. Si creamos un nuevo archivo en esta carpeta aparecerá una nueva entrada de blog en el sitio web en orden cronológico inverso (de la más reciente a la más vieja). Detallaremos cómo crear entradas de blog en la próxima sección
- la carpeta **_sass** contiene archivos SCSS que controlan el diseño visual del sitio web
- la carpeta **_site** almacena las páginas HTML que aparecen en Internet (por ej., nuestras entradas de blog serán escritas como archivos Markdown pero Jekyll las convertirá a HTML para mostrarlas en Internet)
- **about.md** es un ejemplo de *página de Jekyll*. Ya se encuentra linkeada en el encabezado de nuestro sitio web y podemos cambiar el contenido de esta página abriendo el archivo about.md y modificando el texto. Detallaremos cómo crear nuevas páginas en la próxima sección

- la carpeta **css** contiene CSS obtenido a partir del SCSS que controla el diseño visual del sitio
- **feed.xml** permite que el público siga el feed RSS de las entradas de nuestro blog
- **index.html** controla la estructura de la página de inicio del sitio

Redacción de páginas y entradas de blog redacción-de-páginas-y-entradas-de-blog-

Esta sección describirá cómo crear páginas o entradas de blog en tu sitio web.

Páginas y **entradas de blog** son dos tipos de contenido escrito, pero con estilos diferentes. Las páginas (como “Acerca de”) no están organizadas ni se muestran cronológicamente; sin embargo, pueden ser incluidas en el menú principal de tu sitio web. Las entradas de blog están pensadas para ser utilizadas como contenido organizado por fecha de publicación. Las URLs para páginas y entradas también son diferentes por defecto (pero tú puedes cambiar eso): las URLs de página se ven como MySite.com/about/, mientras que las URLs de entradas se ven como MySite.com/2016/02/29/my-post-title.html.

Escritura en Markdown escritura-en-markdown-

Markdown es un lenguaje de marcado para dar formato a tus escritos para que puedan ser leídos en la web: es un conjunto de símbolos, fáciles de recordar, que muestran dónde debe añadirse el formato del texto (por ejemplo, un # delante del texto significa que se le da formato como encabezado, mientras que un * significa que tendrá formato como elemento de lista con viñetas). Para Jekyll en particular, Markdown permite escribir páginas web y entradas de blog de una manera cómoda para los autores (por ejemplo, no es necesario buscar/añadir etiquetas HTML mientras se intenta escribir un ensayo), y que el escrito aparezca con un buen formato en la web (es decir, convertido de texto a HTML).

En esta lección no cubriremos Markdown; si no estás familiarizado con él, puedes crear entradas y páginas sin formato (es decir, sin negrita / cursiva, encabezados, listas enumeradas o viñetas). Pero es sencillo aprender a agregarlos: aquí hay una guía de referencias de markdown en inglés, también puedes consultar esta guía en español, así como la lección en Programming Historian de Sarah Simpkin sobre el cómo y porque escribir con Markdown. Consulta estos enlaces si quieres dar formato al texto (cursiva, negrita, encabezados, listas enumeradas o viñetas), añadir hipervínculos, incrustar imágenes u otros archivos.

Asegúrate que la guía de referencias de Markdown que consultes sea similar a “kramdown”, porque es lo que admite GitHub Pages (donde alojaremos nuestro sitio web). (Hay varios “tipos” de Markdown con sutiles diferencias en lo que respecta a símbolos, pero en su mayoría los que se usan más frecuentemente, como los que crean el formato de encabezados, son iguales. Por lo tanto, puedes utilizar una hoja de referencia Markdown que no especifique que se trate de kramdown, pero si recibes errores en tu sitio web usando símbolos que no están incluidos en kramdown, este podría ser el motivo).

Si te interesa un editor de Markdown, puedes utilizar uno como Typora (OS X y Windows; de descarga gratuita), que te permitirá utilizar atajos de teclado (por ejemplo, resaltar texto y presionar cmd-B o Ctrl-B para ponerlo en negrita) y/o hacer que se muestre tal y cómo se verá en la web (ver los encabezados con el estilo de los encabezados, en lugar del texto normal con un # delante de ellos).

Creación de páginas creación-de-páginas-

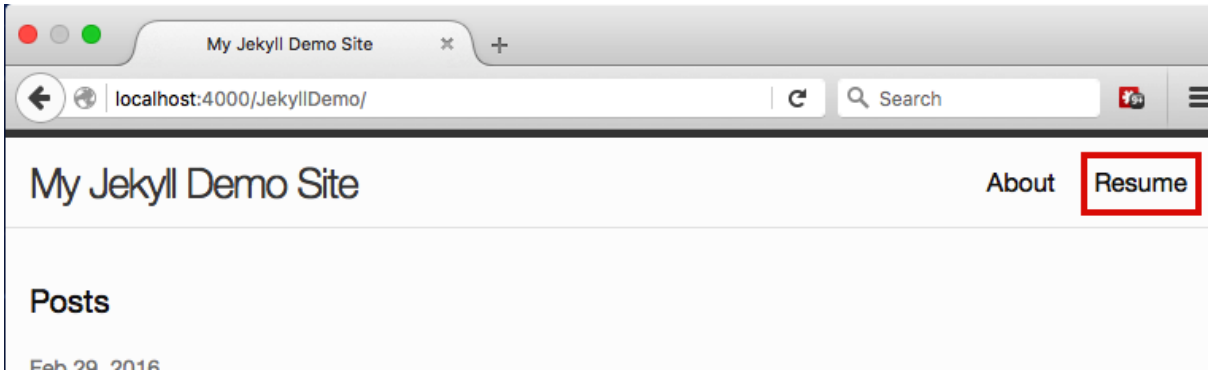
1. Para ver una página existente en tu sitio web (creada por defecto en tu sitio web de Jekyll con el resto de los archivos), navega hasta la carpeta de tu sitio web y abre el archivo about.md en un editor de texto (por ej. TextWrangler) o en un editor de Markdown (p. ej. Typora). Allí verás el archivo creado como "Acerca de (About)". Puedes hacer clic en el enlace "Acerca de", situado en la parte superior derecha de la página web, y podrás observar cómo aparece la página web que crea el archivo en un navegador.

2. El material entre guiones -– se llama "front matter" (*al abrir el archivo en un editor de Markdown este puede aparecer sobre un fondo gris en lugar de entre guiones*). Este apartado le dice a tu sitio si el contenido posterior debe formatearse como página o entrada de blog, el título de la entrada, la fecha y la hora en que fue publicada, y cualquier categoría que quieras que aparezca en la entrada o la página.

   Puedes realizar cambios en el texto preliminar (front matter) de una página:

   - **layout:** Mantén esto tal como está (debería decir "page").
   - **title:** Cámbialo al título deseado (a diferencia de las entradas, no hay comillas alrededor del título). En la siguiente captura de pantalla, se ha agregado una página con el título "Resume".
   - **permalink:** Cambia el texto entre las dos barras diagonales por la palabra (o frase, ¡pero necesitarás usar guiones y no espacios!) que desees que continúe la URL principal de tu sitio para llegar a la página. Por ejemplo, **enlace permanente:/about/** ubica la página en localhost:4000/yourwebsitefoldername/about/.
3. El espacio debajo del segundo guión del texto preliminar (o debajo del recuadro gris si usa un editor Markdown) es donde debes escribir el contenido de tu página, usando el formato Markdown descrito anteriormente

4. Para crear una nueva página además de la existente "Acerca de (About)" (que puede ser personalizada o eliminada), crea una copia del archivo about.md en la misma carpeta (la principal del sitio web) y cambia el nombre al título que desees, utilizando guiones en lugar de espacios (por ejemplo, resume.md o contact-me.md). También cambia el título, el enlace permanente en el texto preliminar del archivo y el contenido. La nueva página debe aparecer automáticamente en el menú principal en el encabezado del sitio:



La nueva página en nuestro sitio aparece en el menú

Como referencia, puedes consultar un ejemplo de página en mi sitio de demostración, o ver el archivo que está detrás de esa página.

Creación de entradas creación-de-entradas-

1. En Finder (en macOS, en Windows en *Explorador de archivos*), navega hasta la carpeta de tu sitio web (por ejemplo, *JekyllDemo*) y luego dentro de ella, ingresa a la carpeta _posts. Abre el archivo que se encuentra allí con un editor de texto (p. ej. TextWrangler) o un editor de Markdown (por ej. Typora). El archivo se llamará algo así como 2016-02-28-welcome-to-jekyll.markdown (la fecha coincidirá con la de la creación del sitio de Jekyll).

Un ejemplo de post abierto en un editor de textos

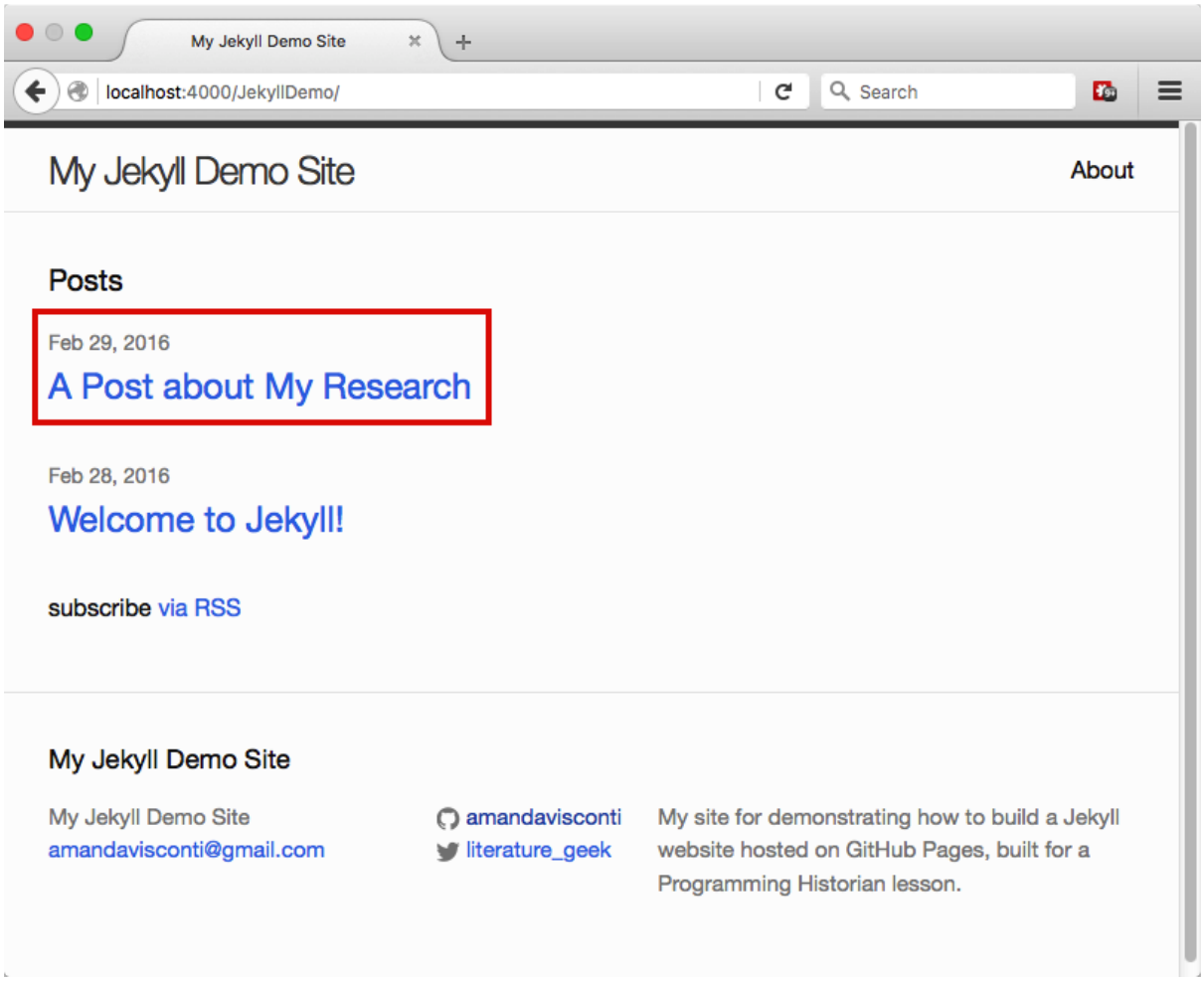
Al igual que en las páginas, el material entre las líneas -– de las entradas se llama texto preliminar (al abrir el archivo en un editor de Markdown este puede aparecer sobre un fondo gris en lugar de entre guiones). Este apartado le dice a tu sitio si debe formatear el contenido posterior como página o entrada de blog, el título de la entrada, la fecha y la hora en que fue publicada, y cualquier categoría que quieras que aparezca en la entrada o la página.

1. Vamos a escribir una segunda entrada para que tengas una noción de la apariencia de múltiples entradas en tu sitio. Cierra el archivo 20xx-xx-xx-welcome-to-jekyll.markdown que estaba abierto, luego haz clic con el botón derecho del mouse en ese archivo en Finder (en macOS) y elige “Duplicar” (en Windows sería copiar y pegar). Un segundo archivo llamado 20xx-xx-xx-welcome-to-jekyll copy.markdown aparecerá en en la carpeta _sites.

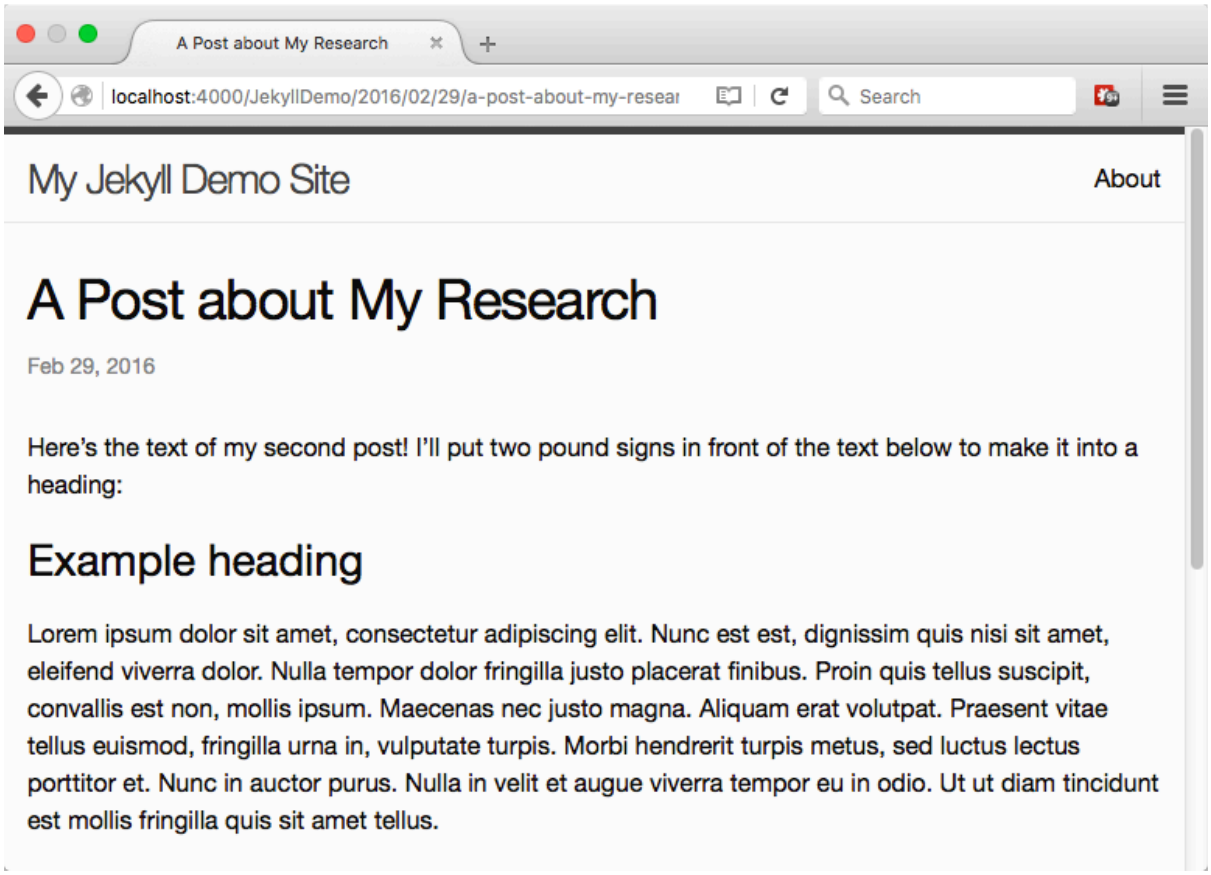
2. Haz clic una vez en el archivo 20xx-xx-xx-welcome-to-jekyll copy.markdown para poder editar el nombre del mismo, y luego modifícalo para que muestre la fecha de hoy y contenga un título diferente, como 2016-02-29-a-post-about-my-research.markdown (utiliza guiones entre las palabras, y **no utilices espacios**).

3. Ahora abre este último archivo en tu editor de texto o de Markdown y personaliza lo siguiente:

    - **layout:** Mantén esto tal como está, no modifiques nada (debería decir *post*).
    - **title:** Cambia “Welcome to Jekyll!” a cualquier título que desees para tu nueva entrada (manteniendo las comillas alrededor del título). La norma es hacer que el título sea igual que las palabras en el nombre del archivo (excepto con espacios añadidos y mayúsculas). Así es como aparecerá el título en la página web de la publicación.
    - **date:** Cambia esto cuando desees que la publicación muestre fecha y hora de publicación, asegurándote que coincida con la fecha que forma parte del nombre del archivo. (La fecha *y* hora deben ser pasadas, para que tu publicación aparezca).
    - **categories:** Elimina, por ahora, las palabras “jekyll update (actualización de jekyll)”, y no agregues aquí nada más, ya que el tema actual no las utiliza y desordena las URL de las publicaciones. (*Otros temas pueden usar este campo para ordenar las publicaciones de blog por categorías*.)
    - **El espacio debajo del segundo -– (o debajo del recuadro gris si usa un editor Markdown)** es donde debes escribir el contenido de tu post, usando el formato Markdown descrito anteriormente
4. Después de guardar el archivo, deberías poder ver tu segunda entrada en la página principal de tu sitio, y al hacer clic en el enlace, debería ir a la página de la entrada:



La nueva entrada ahora aparece en la página principal



La nueva página web

Ten en cuenta que **la URL de la publicación** es la URL de tu sitio web local (por ejemplo, localhost:4000/JekyllDemo/) seguido del año/mes/fecha de publicación, del título tal como está escrito en tu archivo y finaliza con .html (por ejemplo, localhost:4000/JekyllDemo/2016/02/29/a-post-about-my-research.html). Jekyll convierte el archivo Markdown que creaste en la carpeta _posts en esta página web HTML.

Eliminar un archivo de la carpeta _posts lo elimina de tu sitio web (puedes intentarlo con la publicación de muestra "Welcome to Jekyll!!").

Para crear nuevos posts, duplica un archivo existente. Recuerda cambiar el texto preliminar, el contenido dentro de la entrada, así como el nombre del archivo (fecha y título).

Como referencia, puedes consultar el siguiente ejemplo de entrada en mi sitio de demostración, o acceder al código que ejecuta esa entrada.

"Hosting" en GitHub Pages hosting-en-github-pages-

Ahora que ya sabes cómo añadir páginas y publicaciones, en esta sección moveremos tu sitio local a la web, para que otros puedan visitarlo.* **En este punto, estaremos haciendo una versión pública de tu sitio** *(tanto para motores de búsqueda como para cualquiera que conozca o encuentre casualmente el enlace).

Anteriormente en esta lección, instalamos la aplicación GitHub Desktop. Ahora la utilizaremos para mover los archivos de tu sitio a un servidor que los presentará como páginas web (GitHub Pages), que el público podrá visitar en línea. Esta será la primera vez en la que subiremos todos los archivos de tu sitio a la web. En el futuro, utilizarás esta aplicación siempre que hayas realizado cambios en los archivos de tu sitio local y desees que esos cambios se vean reflejados en la versión pública del sitio (al final de esta sección encontrarás una guía con información útil para realizar esta tarea).

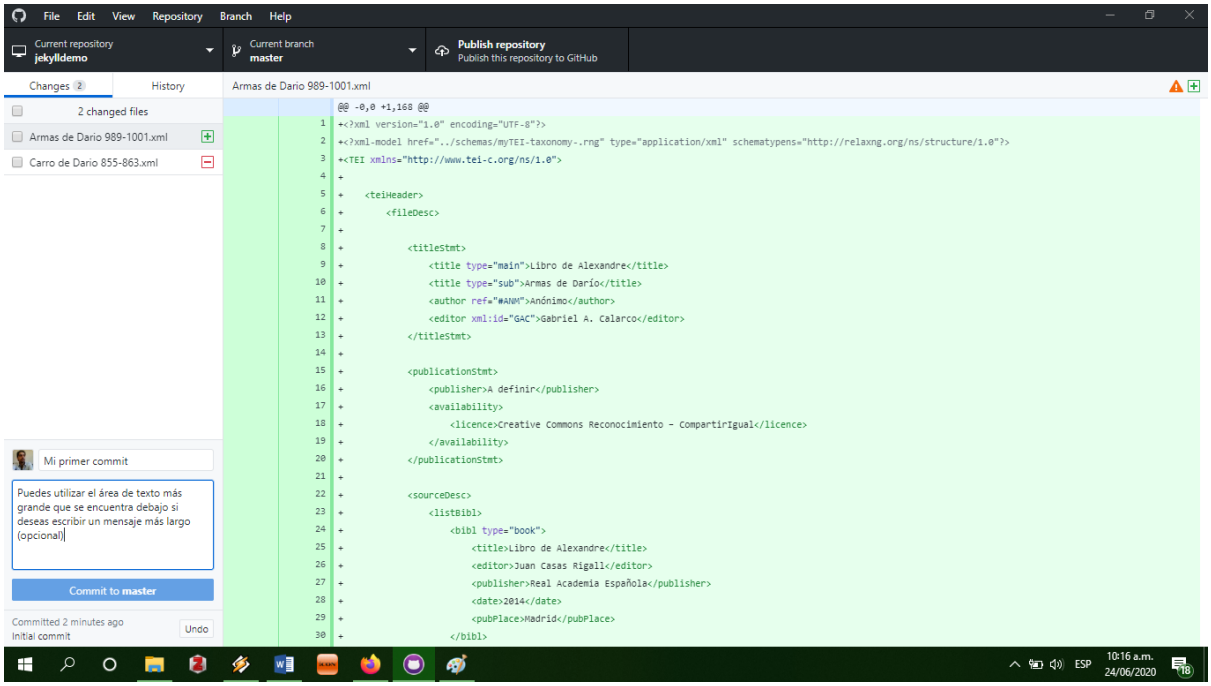
1. Abre la aplicación GitHub Desktop y haz clic en el signo “+” (Mac) o en la pestaña “File” (Windows) que se encuentra en la esquina superior izquierda. Después, haz clic en la opción “Add” (o “Add local repository…”) que aparece en el menú desplegable.

2. Haz clic en el botón “Choose…” y selecciona la carpeta (*JekyllDemo* en nuestro ejemplo) que contiene los archivos de tu sitio local (si estás utilizando Mac y te resulta imposible encontrar esta carpeta, es posible que sea porque se halla oculta; usa estas indicaciones para hacerla visible y que, de esta forma, GitHub Desktop pueda ingresar a ella).

3. Luego, haz clic en el botón “Create & Add Repository” (Mac) o “Add Repository” (Windows). Ahora aparecerá una lista de los archivos en los que has realizado cambios (ya sean adiciones o sustracciones de archivos o en ellos) desde la última vez en la que copiaste el código de tu sitio a GitHub (en el caso de que todavía no hayas realizado este proceso, todos los archivos del repositorio aparecerán listados como nuevas adiciones).

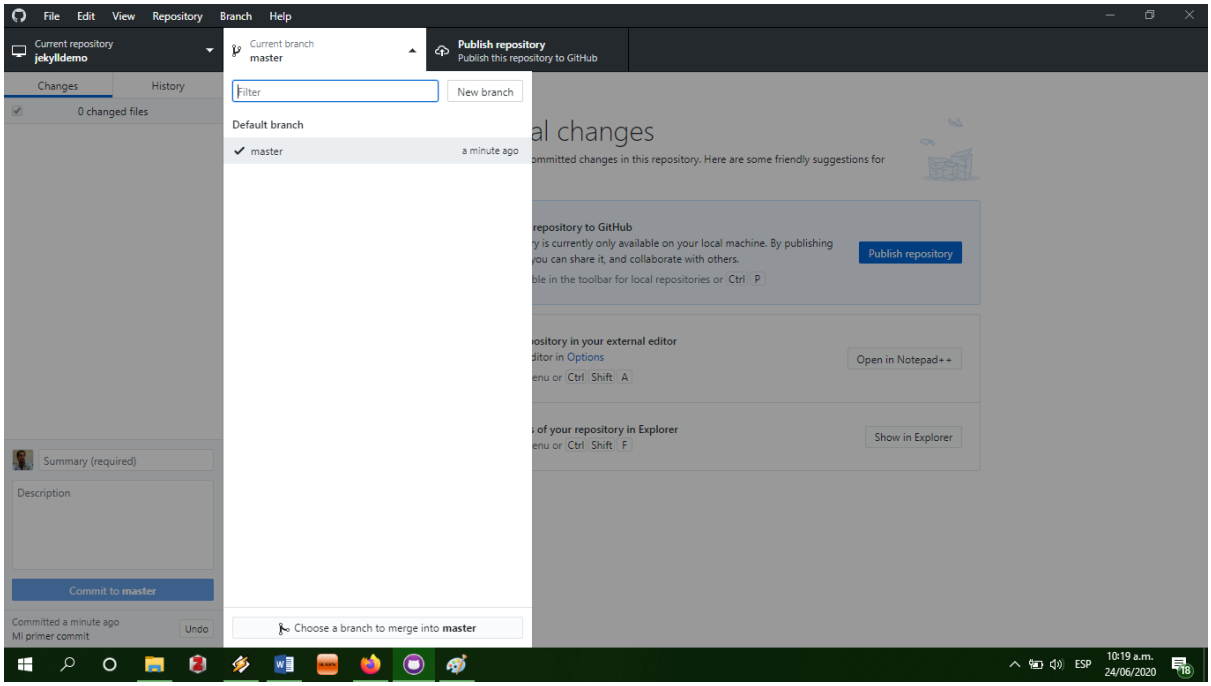
4. Completa el primer campo con una descripción de los cambios que hayas realizado en tu sitio desde la última vez en que hayas subido tus archivos a GitHub (ten en cuenta que el espacio es limitado). Para este primer caso, un comentario breve como “Mi primer commit” será suficiente; en el futuro, es posible que desees ser más descriptivo para ayudarte a localizar cuándo fue realizado un determinado cambio; por ejemplo, escribiendo “Se añade nueva página de contacto”.

   Puedes utilizar el área de texto más grande que se encuentra debajo, si deseas escribir un mensaje más largo (*opcional*).



Captura de pantalla de GitHub Desktop en Windows. En la columna izquierda se pueden observar los cambios realizados, los campos de textos para completar y el botón azul que confirma el commit.

1. En la sección superior de la ventana de GitHub Desktop, haz clic en el tercer ícono desde la izquierda (el mensaje “Add a branch” debería aparecer si colocas el cursor encima de él) (Mac), o en el recuadro “current branch” y luego el botón “New branch” (Windows). Luego, escribe *gh-pages* en el campo “Name” y haz clic en el botón “Create branch”.

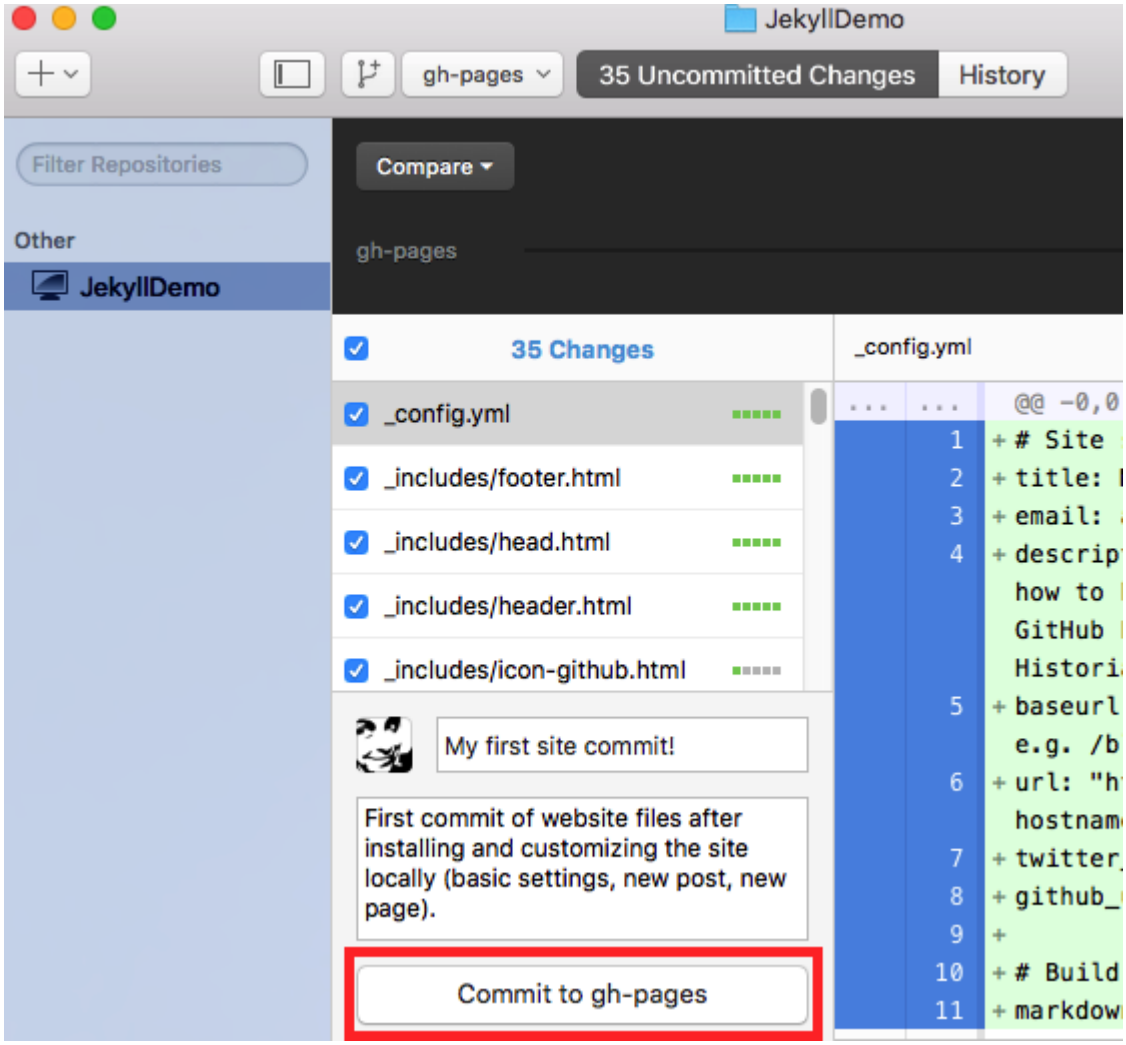


Captura de pantalla de GitHub Desktop en Windows.

A partir de este punto el proceso para publicar nuestro sitio en GitHub Pages difiere entre Windows y Mac

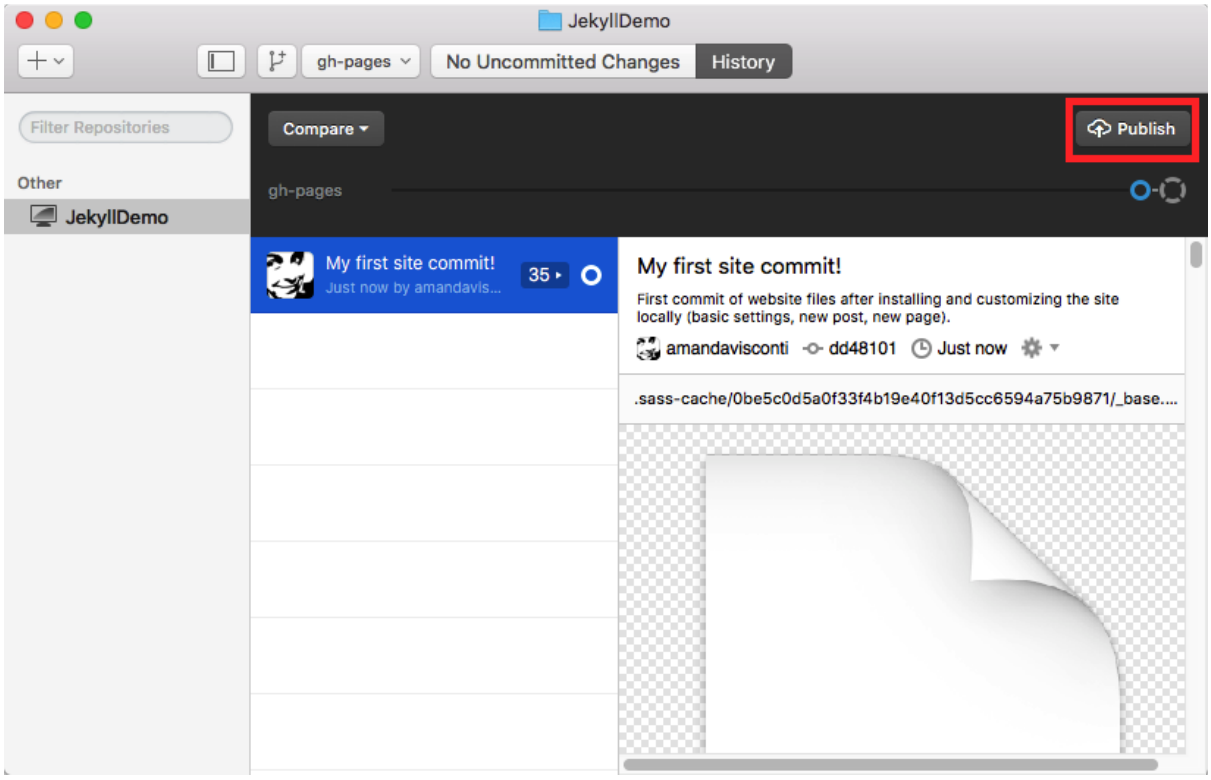
Para usuarios de Mac

1. Haz clic en el botón “Commit to gh-pages” en la sección inferior izquierda de la ventana de la aplicación.



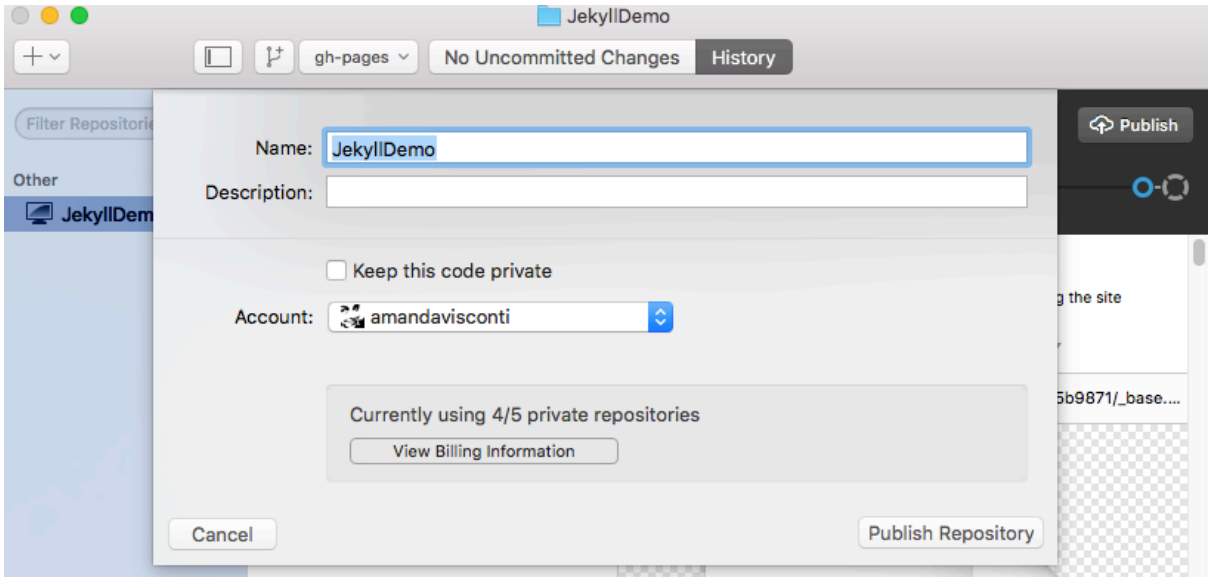
Haz clic en el botón *Commit to gh-pages*

1. Haz clic en el botón “Publish” en la sección superior derecha.



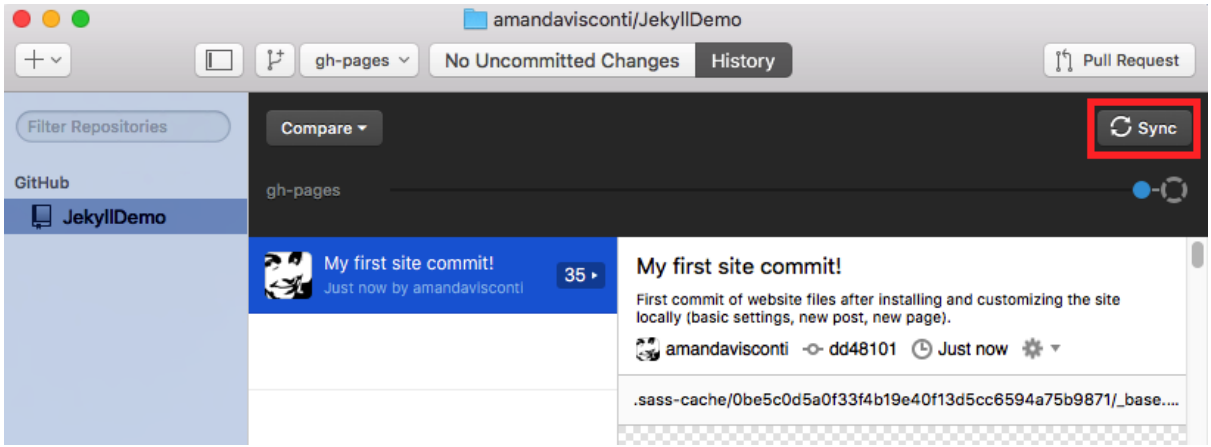
Haz clic en el botón *Publish*

1. En la ventana emergente, deja todo como está y haz clic en el botón “Publish repository” en la sección inferior derecha (puede que tu ventana no muestre las opciones relativas a repositorios privados que se muestran en la captura de pantalla).



La ventana emergente

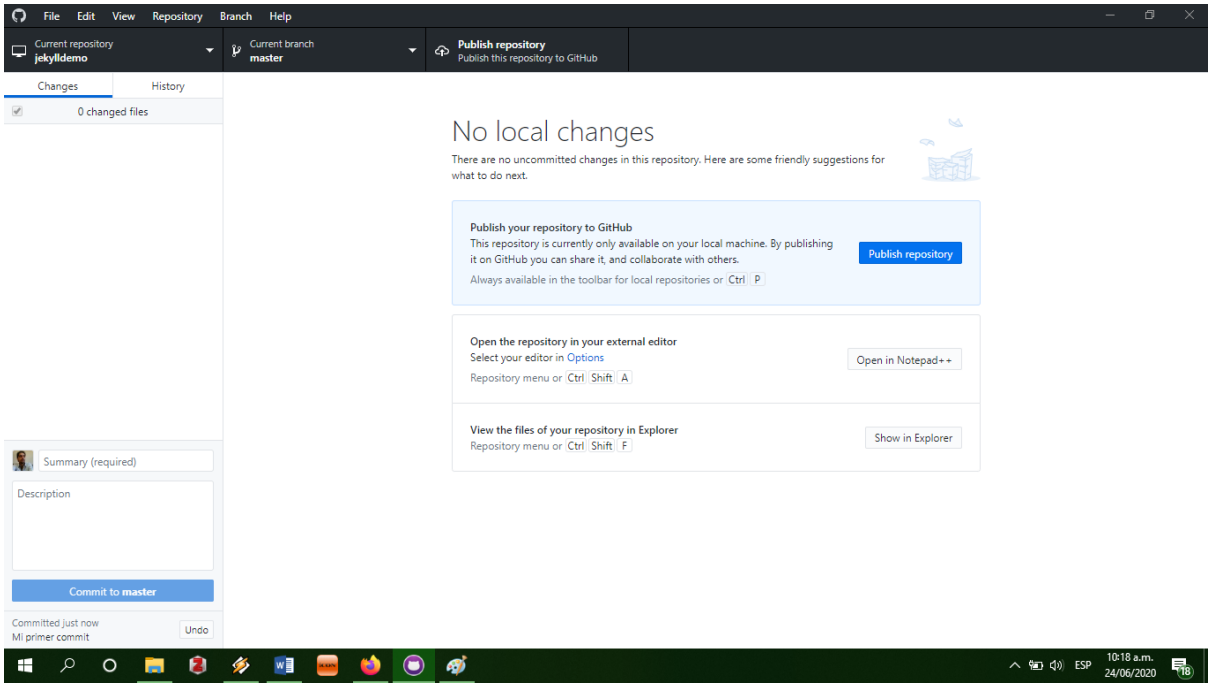
1. Haz clic en el botón “Sync” que se encuentra en la sección superior derecha.



Haz clic en el botón *sync*

Para usuarios de Windows

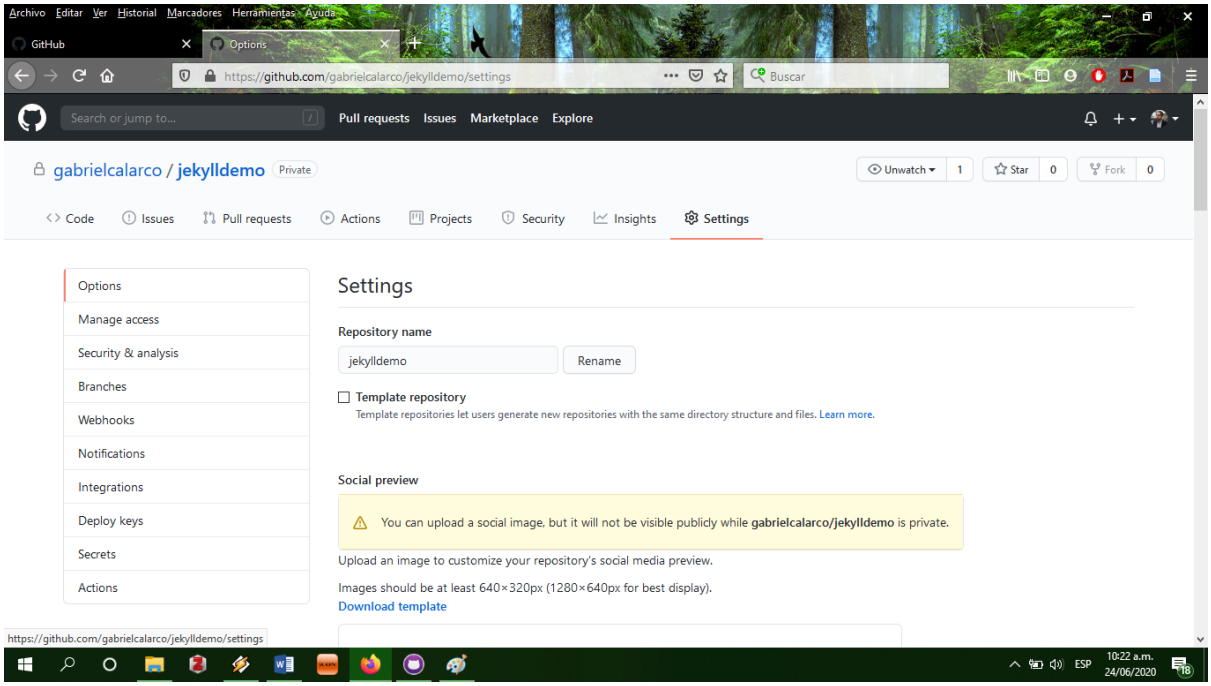
1. Haz clic en el botón “Publish repository” que aparece destacado en azul.



Captura de pantalla de GitHub Desktop en Windows.

1. Haz clic en el botón “View on GitHub” que aparece en la parte central de la ventana, en tercer lugar.

2. Ya en GitHub, debes cambiar las opciones de privacidad de tu repositorio para hacerlo público. Para esto, accede a la sección de “Settings” y baja hasta el recuadro titulado “Danger Zone”. Haz clic en el botón “Change visibility” y selecciona la opción “Make public”.

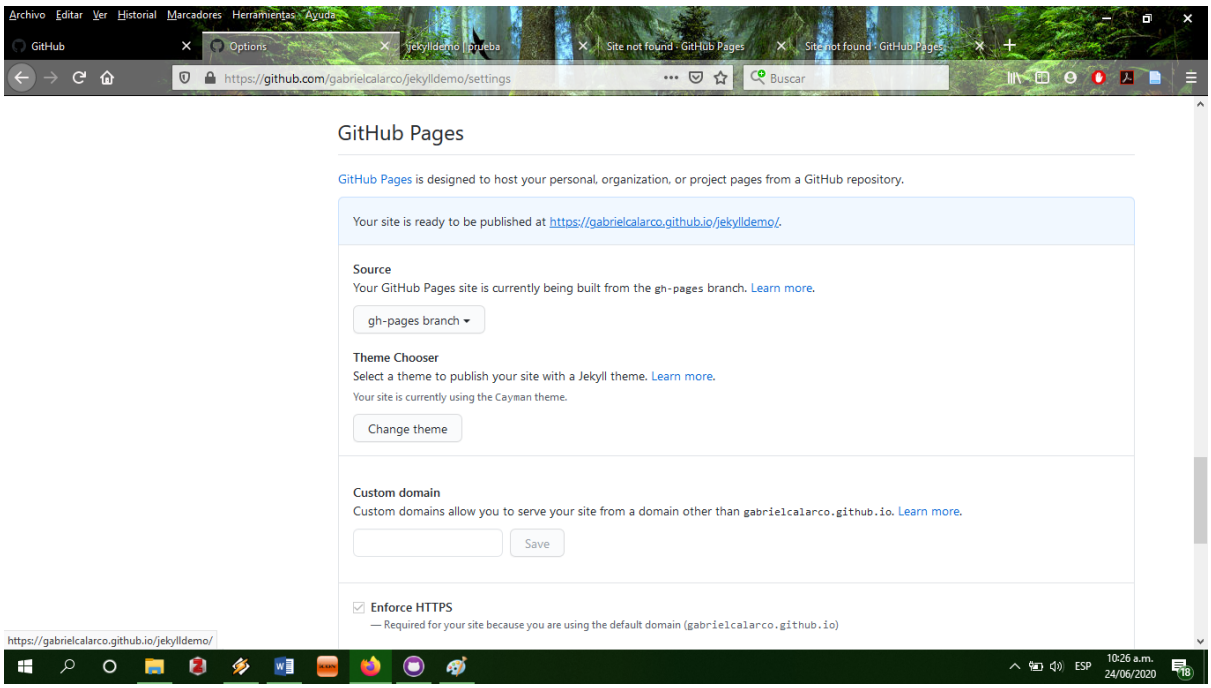


Ubicación de la opciones (settings) del repositorio en GitHub.

<img src="/images/building-static-sites-with-jekyll-github-pages/sitios-estaticos-con-jekyll-y-github-pages_11.png" alt=""Danger zone" de las opciones de GitHub.">

“Danger zone” de las opciones de GitHub.

1. Arriba del recuadro de Danger Zone se encuentran las opciones de GitHub Pages. Allí debes cambiar la opción “Source” y seleccionar la rama (branch) “gh-pages”.



Sección de GitHub Pages en las opciones de GitHub.

Para Mac y Windows

1. Ahora ya puedes visitar tu sitio web (¡y compartir el enlace para que otros lo exploren!). La URL sigue la estructura de tu nombre de usuario de GitHub PUNTO github.io BARRA nombre de tu sitio web BARRA. (por ejemplo, la URL del sitio de ejemplo de la autora es amandavisconti.github.io/JekyllDemo/).

Poniéndonos elegantes poniéndonos-elegantes-

Esta lección no cubre el trabajo avanzado para personalizar la apariencia de tu sitio web ni la adición de nuevas funcionalidades; sin embargo, aquí compartimos algo de información para que puedas comenzar a investigar por tu cuenta.

Diseño visual diseño-visual-

El diseño visual de un sitio web es referido usualmente como el *tema* (aunque propiamente, un tema es el conjunto de código y archivos de imagen que generan un cambio importante en la apariencia de un sitio web).

Puedes personalizar el tema de tu sitio realizando cambios en los archivos que se encuentran en las carpetas _sass y css. (lamentablemente, en sus versiones más recientes, Jekyll comenzó a usar SASS en lugar de CSS, lo que hace que sea más difícil aprender a personalizarlas para los no-diseñadores). También puedes añadir (y luego personalizar, si lo deseas) un tema creado por alguien más, a los que puedes acceder realizando una búsqueda con el término “Jekyll themes” en alguno de los siguientes recursos:

- Tema “Ed” para ediciones digitales mínimas, de Alex Gil (gratis)
- Tema “Digital Edition”, de Rebecca Sutton Koese (gratis)
- El directorio de Jekyll Themes (gratis)
- JekyllThemes.io (gratis y pago)

Funcionalidad funcionalidad-

- Los plugins de Jekyll te permiten añadir pequeños segmentos de código que permiten sumar funcionalidades a tu sitio, tales como realizar búsquedas de texto, permitir el uso de emojis, o crear nubes de palabras.

- Si deseas alojar tu sitio en GitHub Pages, como lo hicimos en esta lección, solo podrás utilizar los plugins de Jekyll que ya están incluidos en las *gems* de GitHub Pages que instalamos (aquí tienes una lista completa de lo que hemos instalado, cuando añadimos la *gem* de GitHub Pages a nuestro Gemfile).

- Si decides alojar tu sitio de Jekyll en otro servidor que no sea GitHub Pages, puedes utilizar cualquier plugin de Jekyll (las instrucciones para alojar tu sitio varían entre diferentes proveedores de hosting web y no las desarrollaremos en esta lección, pero aquí tienes una página que explica cómo instalar plugins, una vez que poseas tu sitio con hosting propio). Puedes realizar una búsqueda utilizando “Jekyll plugin” y añadir la funcionalidad que necesites para explorar si hay una herramienta

apropiada disponible, o revisar la documentación sobre plugins en el sitio oficial de Jekyll.

- También puedes mantener GitHub Pages como hosting gratuito para tu sitio, pero darle un **nombre de dominio personalizado** (los dominios pueden ser adquiridos por un costo razonable -que suele rondar los 10 dólares anuales- a través de un registrador de dominios como NearlyFreeSpeech.net). Por ejemplo, el blog de la autora de este tutorial, LiteratureGeek.com, fue hecho con Jekyll y está alojado en GitHub Pages, al igual que el sitio que creaste en esta lección, pero utiliza un dominio personalizado que la autora compró y configuró para que condujera a su sitio web. Las instrucciones para establecer un dominio personalizado pueden ser encontradas aquí.

- Además, puedes **migrar un blog existente** desde otras plataformas, incluyendo WordPress, Blogger, Drupal y Tumblr, para lo cual debes seguir el enlace que se encuentra en el sector derecho de esta página. Cuando migres un sitio, asegúrate de tener una copia de seguridad de tu sitio original, en caso de que necesites realizar más de un intento para que las publicaciones del sitio queden en la misma URL que antes (y que de esta forma el sitio se mantenga en los resultados de los buscadores y en los marcadores).

Guía guía-

Para realizar pruebas en el sitio de forma local (nuevos plugins, temas, o explorar cómo luce una nueva publicación):

- *Correr el sitio en forma local*: Escribe bundle exec jekyll serve --watch en la línea de comandos.
- *Visitar el sitio local*: Abrir **localhost:4000/yourwebfoldername/** en un explorador (por ejemplo: *localhost:4000/JekyllDemo/*). ¡No olvides la barra inclinada al final!
- *Ver los cambios en el sitio local a medida que los vas realizando*: Para visualizar los cambios en los archivos mientras el sitio está abierto en el explorador, debes guardar los archivos modificados y refrescar la página en el explorador; a menos que el cambio se haya realizado en el archivo _config.yml, en cuyo caso debes cerrar el sitio en el navegador y luego volver a entrar para ver los cambios.
- *Detener el sitio local*: Presiona **control-c** en la línea de comandos.

Para trasladar los cambios que realizaste en tu sitio local a la versión en línea:

- Abre GitHub Desktop y escribe una breve descripción de los cambios realizados (de forma opcional, también puedes escribir una descripción más larga en el segundo cuadro de texto).
- Haz clic en el botón “commit” que se encuentra debajo del recuadro de texto.
- Una vez que el commit haya finalizado, haz clic en el botón “Sync” en la sección superior derecha de la pantalla (Mac) o en el botón “Push origin” que aparece destacado en azul (Windows).

- Espera un poco a que GitHub reciba los cambios (usualmente entre 10 a 90 segundos) y refresca tu sitio online para ver los cambios allí reflejados.

Ayuda, créditos y lecturas ayuda-créditos-y-lecturas-

Ayuda ayuda-

Si encuentras algún problema, Jekyll tiene una página para problemas, conocidos como troubleshooting, que te puede ayudar. Si estás trabajando en la línea de comandos y recibes un mensaje de error, no te olvides de buscar más acerca del error en la web. Más allá de los motores de búsqueda tradicionales, el sitio StackExchange es un buen lugar para encontrar preguntas y respuestas de gente que tuvo este tipo de problemas.

Creditos creditos-

Gracias a Fred Gibbs, editor del *Programming Historian* por editar, debatir y revisar la lección original. A Paige Morgan por revisar la lección; a Scott Weingart y sus estudiantes por poner en práctica y testear esta lección en Windows; a Tod Robbins y Matthew Lincoln por sugerencias en DH Slack sobre lo que debería enseñar esta lección. Asimismo, agradecemos a Marc Bria por su revisión y sugerencias con respecto a la traducción de esta lección al español.

Lecturas lecturas-

Puedes visitar estos sitios para más documentación, inspiración y para aprender más sobre Jekyll:

- Documentación oficial de Jekyll
- Jekyll tiene links a recursos “no oficiales” sobre su funcionamiento en Windows: https://jekyll-windows.juthilo.com/ y https://davidburela.wordpress.com/2015/11/28/easily-install-jekyll-on-windows-with-3-command-prompt-entries-and-chocolatey/
- https://help.github.com/articles/using-jekyll-with-pages/
- Amanda Visconti, “Introducing Static Sites for Digital Humanities Projects (why & what are Jekyll, GitHub, etc.?)”
- Alex Gil, “How (and Why) to Generate a Static Website Using Jekyll, Part 1”
- Eduardo Bouças, “An Introduction to Static Site Generators”
- Guía de estilo de Jekyll
- Prose: editor de contenido (creado en Jekyll)
- Únete al Slack sobre humanidades digitales (cualquiera puede sumarse; no se necesita saber sobre humanidades digitales) y súmate a los debates acerca de Jekyll y otras plataformas de publicación en el canal #publishing.

Acerca del autor

Amanda Visconti es Directora general del centro de Humanidades Digitales Scholars' Lab en la Universidad de Virginia.

Cita sugerida

Amanda Visconti, "Creación de sitios estáticos con Jekyll y GitHub Pages", traducido por HD CAICYT Lab team, Gimena del Rio Riande, Nidia Hernández, Romina De León, Gabriel Calarco, y Raffaele Viglianti, *Programming Historian en español* 5 (2021), https://doi.org/10.46430/phes0050.