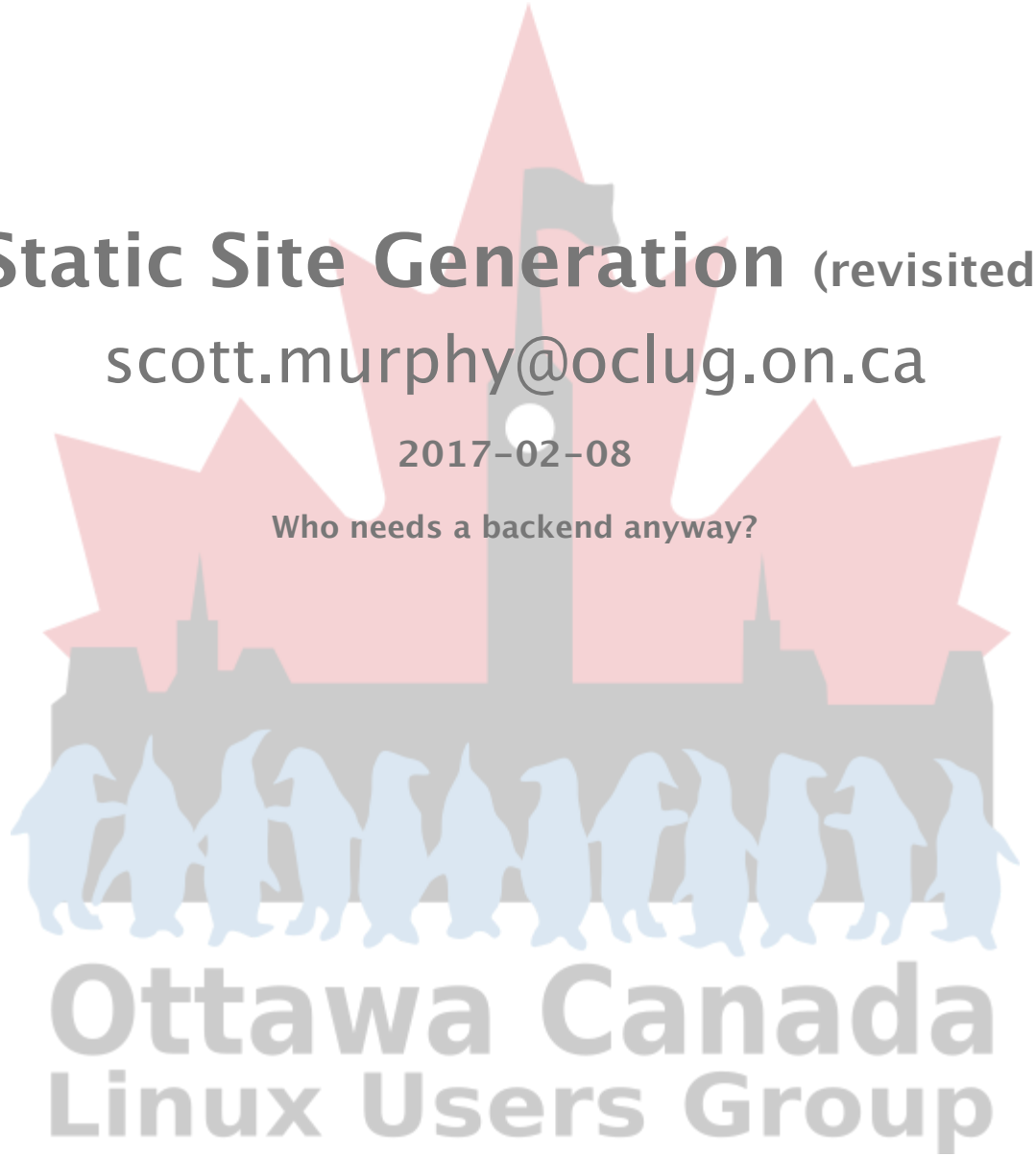
# Static Site Generation (revisited)

scott.murphy@oclug.on.ca

2017-02-08

Who needs a backend anyway?



# Some Basic Information

# Who am I?

For the most part, a systems administrator concentrating on Unix and Unix like operating systems. I started playing with Unix back in 1982 and have not looked back since.

# About this Talk

This talk is an updated and expanded talk based on the original short talk I did for OCLUG in June of 2015. That was a 20 minute talk and did not cover any information beyond the basics and a demo. Hopefully this one has some information that you will be able to use.

Static site generators have been a surprisingly active area over the past couple of years and there has been a lot of developments since 2015.

# What is a Static Site?

A static site is typically:

- A site that has no back end system to provide dynamic content
- Just a collection of HTML, CSS, and possible some javascript
  - There may even be media files
- It is usually served by a lightweight webserver

Ottawa Canada
Linux Users Group

# What is a Static Site Generator?

A static site generator (SSG) is:

- A tool that transforms content from one format, such as Markdown or AsciiDoc, into another format, usually HTML
- It lays out pages so that the site's look and feel is consistent across all pages
- Static sites can be deployed to any web server.
- It may have a plugin ecosystem that you can use to offer enhanced functionality
  - Embed external systems such as disqus
  - Add a photo gallery lightbox
  - The list goes on...

# Is a Static Site for You?

Before making any decision, examine:

- Your requirements, size, complexity, etc.
- Your target audience
- Your development skills
- Hosting/deployment factors

# Various Reasons Why You Should Not Have a Static Site

# You are on Your Own

Without some experience with software development, you will probably struggle. The vast bulk of static systems will require some intervention on your part to get them operating in a fashion that you will be happy with. As with a lot of open source software, having the technical ability will smooth over many rough edges. This is not like a content management system (CMS) and you will likely need some knowledge beyond “install & run”.

# A Myriad of Choices

There are many static site generators out there. I have a list of 168 tools for static site generation. While that is a lot, not many people use the vast majority of them. You can check the list of static generators provided in the references to get a rough count of usage. Given that, there are not really a lot of people using static sites, so you need to spend time on research and evaluation.

# Initial Setup Time

Your first site will take more time than you think. The initial generating of content will probably go well but then you will probably want to theme it, which will take more time.

You will also have to develop a workflow for your deployment - it typically does not end up on the web site without some intervention. There is probably an existing build process that you will have to tailor.

# Management Interface?

You think that a CMS interface is daunting to a non technical person? Try no interface. Everything happens from a command line (or a script with a couple of buttons - if you build it). You are editing a pile of markdown files and then following a work flow or running a script. I hope you like the command line.

# Consistency

Your standard CMS enforces control over the author's output. Since it stores elements in a database and these elements are usually fields such as title, body, excerpt, etc., the system drops the correct element in the right place. The theme then sets the appearance of such things and there is typically little writer control over how this is displayed. The theme is in control.

Static sites on the other hand, have no such protections. While there are standard ways to tag these stylable elements, it is not enforced. Anything you can enter in markdown or HTML will be displayed.

# Larger Site Considerations

Your website contains hundreds or thousands of pages composed of regular features, items of immediate interest, and special features. There may be multiple authors or other considerations – maybe you have a news aggregator.

While it would be possible to manage content using a static site generator, you need to consider that:

- Editing and publishing is more awkward. Editors may require access to the Git repo or shared folders rather than a simple web or app interface.
- Real-time updates would be delayed because the site must be rebuilt, tested and deployed build times could increase rapidly and deployment would become cumbersome.
- Conventional wisdom dictates that static site generators are perhaps best suited to sites containing no more than a few hundred pages with a couple of new posts every week
- Automated build and deploy processes will be required, and you may reach a point where a CMS becomes a more practical option

# Server-side Limitations

A true static site does not offer interactive facilities such as user logins, form filling, search or discussion forums. There are some add-ons that offer functionality such as search, or comments but forms are right out unless you can find a hosted form system.

If you need something like this, there are some build steps that let you shim in php code to create content, however there is no database and the result is still HTML.

# Other Considerations

- Not SEO friendly enough
- Hard to update
- Difficult to syndicate
- Less functionality
- Organizational bottlenecks
- Static sites are ugly
- Static websites are going obsolete

Ottawa Canada
Linux Users Group

# Why have a Static Site?

# First a Reality Check

Most websites are:

- Less than 50 pages
- Receive infrequent updates
- Depend on a someone to make changes

A CMS is often overkill for many use cases and static site generation could simplify development/deployment and can certainly reduce operating costs and maintenance.

# Why Static?

There is no reason to have a massive CMS with a database and attendant systems when you could have a fast, secure and (with the right CSS theming) beautiful static site.

How can you decide if a static site is the right system for you?

# Security

With a static site, you don't have to worry code being injected into your site when users visit it.

Static sites are built on a production machine by static site generators, which take your master code and spit out flat HTML files with CSS and JavaScript. When a user requests a page from your site, the server just sends them the file for that page, instead of building that page from various assets each time.

No assembly process means standard hacking attacks like scripting or database security exploits just don't work. There is no db query, no dynamic update, just an HTML page being returned

# You Feel the Need for Speed

- Your browser (no matter what one you use) renders renders the HTML, CSS, and javascript into a hopefully pleasant human reading experience.
- The server, regardless of the platform or OS, sends HTML, CSS, and javascript.

So, if all that is all that is being sent, then just having that pre-rendered would speed up page serving by a large percentage. Think of all of the cumulative overhead saved on generating content every time a browser hits a page.

# Improved Resource Utilization

Closely related to the previous slide, this is not the generation of the content, but the amount of resources consumed to generate that content.

Your basic site built with many popular CMS offerings start out as a generic solution that is then customized by adding plugins. This tends to grow to many plugins over time. I know that in many implementations, each plugin is effectively loaded by each page viewed “just in case” the functionality is needed. This is slow and has memory overhead associated with it.

With a static site, you can get lean and mean, with a tool that does exactly what you need it to (probably with some tweaking) and it will be fast!

# Smaller Footprint

A normal CMS install is a semi-organized collection of software components and hardware, which resembles:

- A LAMP stack (or WAMP)
- A server running your preferred distro of Linux or Windows
- A web server running Nginx or Apache
- PHP with required extensions and web server tweaks
- MySQL (typically)
- CMS of choice
- All the plugins!
- Your theme and template code

# Smaller Footprint (continued)

A static site, once generated only requires any web server that can return HTML files.

That's it.

# Reliability

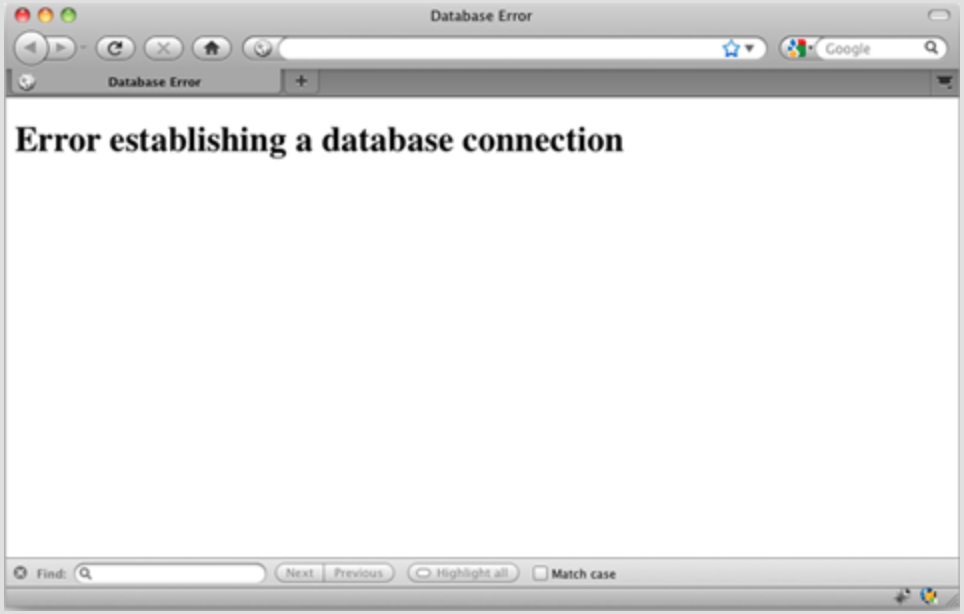



Image from: http://www.jasonbobich.com/2010/08/01/moving-wordpress-to-a-new-server/

Ever seen an error like this one? If you've managed to avoid it, you are in the minority. Obviously, working without a database you won't see that, but static site reliability goes beyond database errors.

Ottawa Canada
Linux Users Group

# Reliability (continued)

The beauty of serving up flat HTML files is that they can be hosted anywhere and everywhere, like on a Content Delivery Network (CDN). What if there is a DDoS attack on the server where your non-static site is hosted?

Chances are you will be down for at least a few hours, maybe even days. If that attack hits a node where your static site is hosted? If you are on a CDN, then your site just gets served up from the next closest node. Chances are nobody will even notice that there's a problem. Even if it is not a CDN, all you need to be back up and running is a basic webserver. That can be stood up in minutes.

# Revision/Version Control

You can design a site on your desktop/laptop machine and then upload it to a host and you are done. The majority of developer types will end up using some sort of version control system like Git. You already know how much of a lifesaver this can be on projects, but it's useful when it comes to developing websites as well. Static sites are text files, so they work very well with version control.

If the unimaginable happens and you mess up your source, you can roll back to an earlier version easily. Better yet, unless you published, your site has not even been touched yet.

# Expected Output

One of the joys of using a static website generator is in the build and testing process. The build process will output your site to a particular directory on your build machine, and most build systems include a local web server.

This allows you to check your progress as you go. You will know that your site looks exactly the same to your visitors as it does to you as the designer/developer.

# Can we say Scalable?

Congratulations, your site has gone viral.

You are seeing insane page hits and it is growing. Did you take this into consideration when planning? Did you overprovision or was just enough resource the way you went?

With a static sites, scalability is barely an issue. Requests mean increased pages served, but no increased overhead building those pages. With some providers, scale is built in, while with others like Amazon's S3, all you have to pay for is the increased bandwidth.

# Hosting Costs

If all the work of building a static site is done on your development machine, then what exactly is it that you are paying for?

## Data storage!

Your static files take up little space, and so your service is minimal, and hopefully reflected in your cost.

# Flexibility

Most CMSs normally limit your options because they're tied to a database with specific fields. If you want to add a widget to some pages, you'll normally require a plugin, a shortcode or some custom functionality.

In a static site, the widget can simply be inserted into a file directly or using a partial/snippet. There are few limits, because you no longer have the constraints imposed by a CMS.

# Demo Time

# What are we going to see?

Well, I don't have a Windows box with me, so I'll use my Macbook as the source machine and I have a server set up to be the target for the tools. You should be able to reach it via the local WiFi connection. If not, I'll use my phone as a tether and we'll do it that way.

- MkDocs
  - I'm always looking for better ways to keep my information organized and since my preferred document storage format is ASCII text and my preferred method of writing is asciidoc or markdown, this fits the bill nicely.
- Jekyll
  - I'm new to this one and just dusted it off for this demo. Hopefully it lives up to it's reputation. I'm going to use the default and then a modified default to show how the styling changes everything

# Server - Digital Ocean Droplet

We are using a cloud provider called Digital Ocean for the demo. Much easier than setting up a physical machine and running demo code and quite inexpensive.

If I was going to be running this for an extended period, I'd probably use Amazon Web Services or some thing like that to keep it inexpensive. This server costs $0.007/hour to run, so at the end of the talk, it will have cost me less than $0.68US to have this running.

The server is running Ubuntu 16.04 LTS, but there are a variety of other options.

Information for the demo:

- The demo URL was: http://138.197.148.102
- The site was turned off, as it is no longer necessary

If you are interested in checking out the offering, you can get a $10.00 credit to play with creating your own systems by following this link: Digital Ocean

# MkDocs Demo

MkDocs a very easy one to work with. It is designed to create a collection of documents and can be themed to have many different appearances. We will just look at a couple, the MkDocs theme and the readthedocs theme. I have a few markdown documents placed in the directory already, so we do not need to create content.

We will view the site with the built in viewer and then push to an external site, as external access works anyway.

# MkDocs Demo (continued)

Once you have it installed (there are excellent install instructions for most platforms), you have a simple workflow:

1. Create documents in markdown format
2. Edit the mkdocs YAML file to describe the site layout
3. Run the built in server
    1. Review your work
    2. Make changes
    3. Repeat as necessary
4. Build the site
5. Publish the site to your platform of choice

# Jekyll Demo

This is a more complicated one to work with. It is designed around creating a blog or dated publication. The pages can be themed and just a few tweaks can make a large difference in the look of the site. As I am not a Jekyll adept, we will use the default site that gets generated.

# Jekyll Demo (continued)

The workflow is similar, but different:

1. Create documents in markdown format
2. At the top of the file, provide the meta data that describes the post
3. Run the built in server
    1. Review your work
    2. Make changes
    3. Repeat as necessary
4. Build the site
5. Publish the site to your platform of choice

# Some things to Consider

You probably want support for something beyond basic markdown. I'd suggest a system that handles extended markdown or asciidoc as a markup language of choice. They both translate well to a variety of formats and don't have the limitations of base markdown.

ReStructured Text (RST) is also an option, so if you do much python, you are probably an old hand with it.

# Summary

No matter what system you prefer to work with, there's probably a static tool for you. There is more information available at staticgen.com, probably the most comprehensive source of information on static website generators available at this time.

It is a great time to take advantage of the speed, security and reliability of static web generators and the development environment around them. All you need to do is install one and give it a whirl. Who knows, you might even like it.

# References

- StaticGen: Top Open-Source SSGs
- Seven reasons not to use a static site generator
- Seven reasons to use a static site generator
- 6 reasons why your Static HTML site is bad for your Non-profit

# References (continued)

- Markdown Reference
- Markdown Extra
- Markdown Extensions
- AsciiDoc & AsciiDoctor
- MkDocs Site
- Jekyll Site
- Pelican Site
- Make a Static Website with Jekyll
- Yes We Jekyll

# Other Items

## What was used to create the slides?

The slides are all written in markdown with a very tiny amount of HTML to handle things basic markdown does not do, such as mixed font sizes and justification and they were generated with a python package called landslide

## How do I do this with Windows?

- You can install much if not all of this with Chocolaty. The project calls itself “The sane way to manage software on Windows.” Apparently it works with installations based on MSI, NSO=IS, InnoSetup, tc. It also works with runtime binaries and zip archives. No, I have not used it, so no personal experience here.
- You can use cygwin to manage all of this as well. It installs a bunch of posix tools and provides a bash shell to call them from. If you haven’t used it and you need something to run unix tools in, this is a pretty good place to start.