
Cookiecutter Static Site Generator Documentation

Release 0.1

Andrew Pinkham

October 25, 2016

1	Table of Contents	3
1.1	Introduction	3
1.2	Installing Project Prerequisites	4
1.3	Prologue: Signing up for Amazon Web Services	5
1.4	Generating a Static Website Project	13
1.5	Obtaining Security Certificates	16
1.6	Deploying your Static Site to the Cloud	20
1.7	Updating Your Static Website	21
1.8	Contributing to the Cookiecutter Static Site Project	21
2	Indices and tables	23

This is the documentation for the Cookiecutter static site template, hosted on [GitHub](#). The template may be used by [Cookiecutter](#) to generate a project that will allow for easy and rapid deployment of a static website (a website comprised of only HTML, CSS, and JavaScript).

This documentation is written like a walkthrough, intended for beginners with little to no programming experience. While this template relies on programming tools, using the template requires no programming.

If you are a developer, the [Read Me](#) document in the [GitHub project](#) is likely enough for you.

Table of Contents

1.1 Introduction

This project may seem silly. After all, [Amazon provides a full tutorial on how to set up a static website in the cloud using S3 buckets and a CloudFront distribution](#). (You can also watch a [YouTube video with similar content](#).)

So why use this template to generate a template? Well, the difficulty with the tutorial is that it focuses entirely on the visual web console. The net effect is that it takes time, is easily forgotten, and is not reproducible with code.

Far worse, however, is that the official Amazon tutorial does not follow best practices:

- user setup is not covered
- the website is unsecured (no TLS certificate)
- DNS speed is ignored in favor of a bucket redirection trick

The `cookiecutter-static-site` template project aims to:

- be fast
- be easily reproducible
- generate the cloud infrastructure (with two commands)
- use a TLS security certificate to secure the website
- optimize DNS lookups for the website domain
- be forgettable (the template provides a `Makefile` with all of the commands you'll need)

If the template were to recreate the entire Amazon tutorial, only two commands would be necessary.

- `aws cloudformation create-stack`
- `aws s3 sync`

This walkthrough is about *mise-en-scène*. Getting setup is a little involved. You'll have to get setup once, but you'll be off to the races for repeat performances.

This walkthrough is aimed at people who have dabbled with the commandline, but who may not be comfortable with it yet. We'll walk through each and every step needed to get your site up in the cloud. Your next step will be to *install the necessary prerequisites*.

If you are a developer, the [Read Me document](#) in the [GitHub project](#) is likely enough for you.

1.2 Installing Project Prerequisites

This project relies on several programming tools, but can be used without knowledge of programming languages.

Note: Manually downloading and installing software—as indicated by the instructions in this walkthrough—is fine if you are not doing this very often. If you are learning to develop or code, I urge you to look into package management software, as it will save you time in the long run. If you are on BSD or Linux, your system comes with a package manager. If you are on a Mac, I recommend [MacPorts](#) or [Homebrew](#). If you're on Windows, I hear good things about [Chocolatey](#).

1.2.1 Section Goals

This section prepares you for:

1. Using the `cookiecutter-static-site` template to generate a project
2. Generating security certificates to secure your website
3. Using the provided scripts in your new project to upload (deploy) your static website to the cloud

This section lists all of the tools required to perform the three steps above. Rather than provide detailed instructions for each installation, this section refers you instead to installation instructions of each utility.

The tools below are listed in the order they will be used.

1.2.2 Python and *pip*

Python version 2.7, 3.3, 3.4, or 3.5 must be installed for both project generation and site deployment. If you're on a modern Mac, Python 2.7 is already installed for you (but you are encouraged to [upgrade Python](#)). If you're on Windows, you will need to [install Python](#). If you're on BSD or Linux, you don't need my help.

Python is installed with `pip`, a package manager for Python packages/applications.

1.2.3 *virtualenvwrapper*

`virtualenvwrapper` is an optional tool that allows for the logical separation of installed packages. I highly recommend using it.

If you are on Windows, you may wish to use either `virtualenvwrapper-win` or `virtualenvwrapper-powershell`. The [installation guide for virtualenvwrapper](#) has more information.

1.2.4 OpenSSL

Named after the now-deprecated Secure Sockets Layer security protocol, OpenSSL is an open-source utility that allows for many cryptographic and security applications. In this walkthrough, you will use it to obtain a TLS security certificate for your website.

If you're on Windows, you will need to google installations instructions (Sorry, I cannot help you here.) If you're on a Mac, then OpenSSL is already installed. On Linux and BSD, if OpenSSL is not on your system, then your package manager should quickly fix that problem for you.

1.2.5 jq

Whereas Python, *pip*, and *virtualenvwrapper* are tools for generating the static site deployment project, *jq* is a tool necessary for the deployment of the site to the cloud. You will not need to use the tool yourself, but the deployment scripts we will use do rely on the tool.

Downloads and installation instructions may be found [here](#).

1.3 Prologue: Signing up for Amazon Web Services

This project will give you the tools to deploy your website to the [Amazon Web Services \(AWS\)](#) Cloud. For this to happen, however, you need to have an account with AWS. The process is very straightforward, and the actual signup is not detailed here.

Once you're done, you'll be greeted by the overwhelming AWS console.

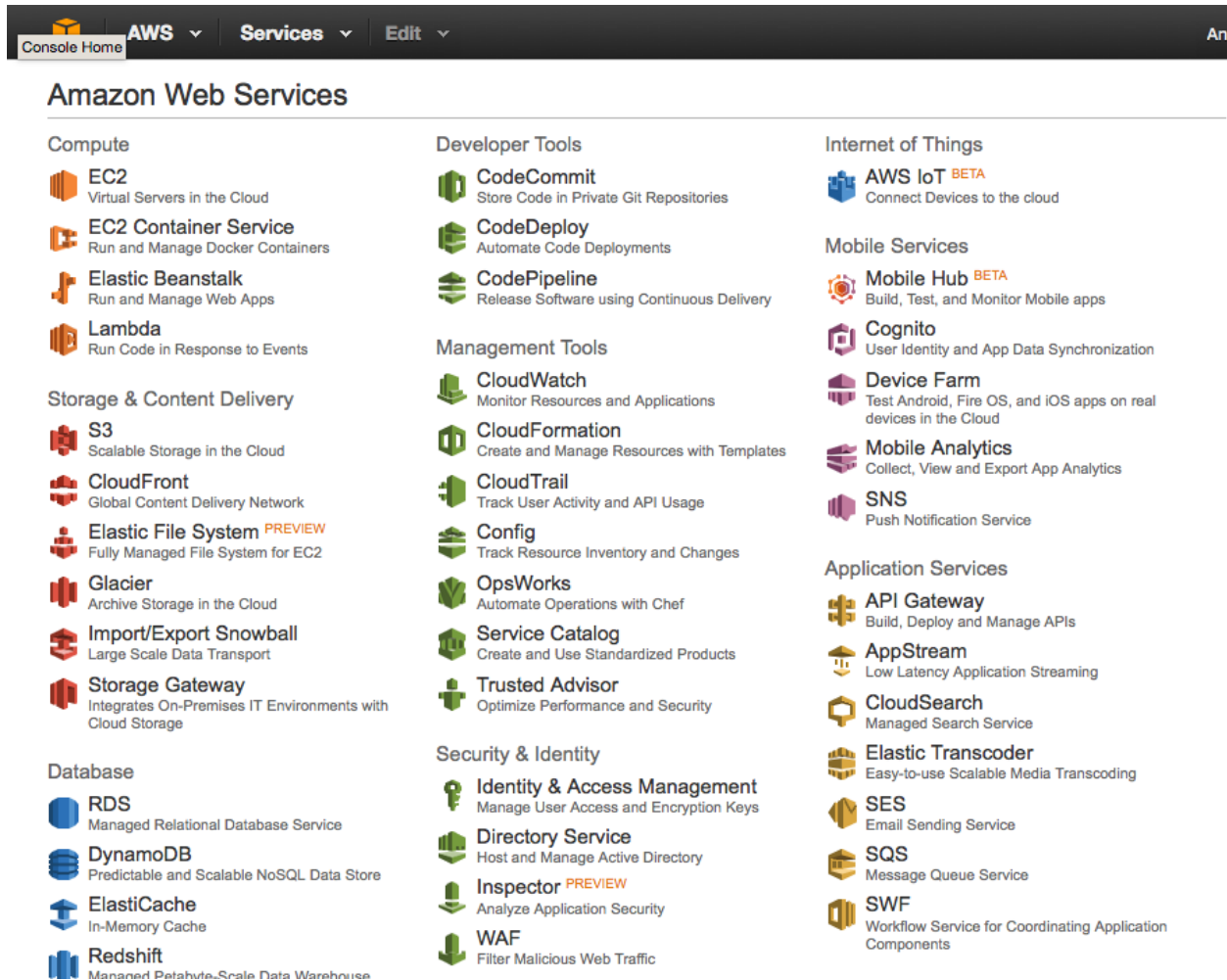


Fig. 1.1: AWS Console, accessible by clicking the top left icon.

1.3.1 Section Goals

Once you have an account, your goal will be to create a user to allow command-line access to the cloud. However, before we can create the user, we will have to secure the cloud.

Both securing an account and creating a user happen in the Identity and Access Management (IAM) section. You can find it on the AWS console under Security & Identity, or you can [click here](#) if you're currently logged in.

1.3.2 Securing your Account

When you are greeted by the IAM Management Console, you will discover a list of actions to take to secure your cloud resources.

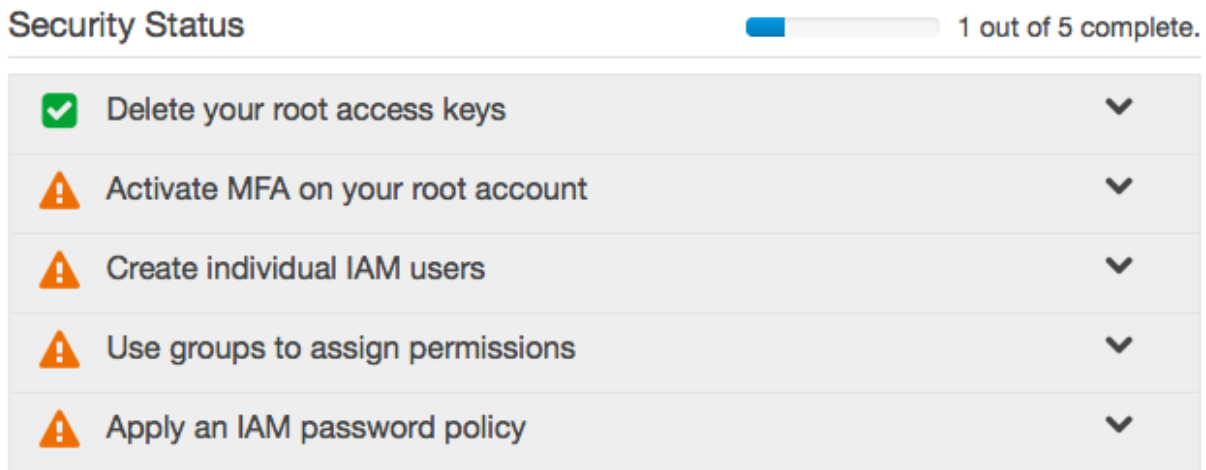


Fig. 1.2: Security Task List in the AWS IAM Management Console

By default on new accounts, the first item will have been completed for you (if your account is older, this may not have been done for you).

The second item, [Multi-Factor Authentication \(MFA\)](#), will allow us to setup [Two-Factor Authentication](#) using a smartphone. This is certainly not required, but I heartily recommend it. Click on the *Activate MFA on your root account* item in the list, then on *Manage MFA*, and then follow the instructions. You will need to install Google Authenticator on your smartphone to properly setup the service.

You may have to hit the *Dashboard* link towards the top left of your window to get back to the IAM console. You'll be greeted by a green checkmark in your task list.

We can then skip to the end of the task list, and setup password restrictions for any future users. Click on *Apply an IAM password policy* and then on *Manage Password Policy*. You'll be greeted with a page with a number of choices. My usual selection consists of the following (but feel free to pick your own):

Once you successfully apply a password policy, the task list will appear with three checkmarks:

1.3.3 Creating a User and applying permissions with a Group

With user management comes the idea of permissions: we want to restrict what a user can do on our system for security purposes. The best way to do this is to create a group, and assign permissions to that group. We then add users to that group, rather than adding permissions directly to the user.

Security Status 2 out of 5 complete.

<input checked="" type="checkbox"/>	Delete your root access keys	▼
<input checked="" type="checkbox"/>	Activate MFA on your root account	▼
<input type="checkbox"/>	Create individual IAM users	▼
<input type="checkbox"/>	Use groups to assign permissions	▼
<input type="checkbox"/>	Apply an IAM password policy	▼

Fig. 1.3: Security Task List with MFA Setup Complete

Minimum password length:

- ☐ Require at least one uppercase letter ⓘ
- ☐ Require at least one lowercase letter ⓘ
- ☐ Require at least one number ⓘ
- ☐ Require at least one non-alphanumeric character ⓘ
- ☒ Allow users to change their own password ⓘ
- ☒ Enable password expiration ⓘ

Password expiration period (in days):
- ☒ Prevent password reuse ⓘ

Number of passwords to remember:
- ☐ Password expiration requires administrator reset ⓘ

Apply password policy
Delete password policy

Fig. 1.4: Andrew's Password Policy Settings.

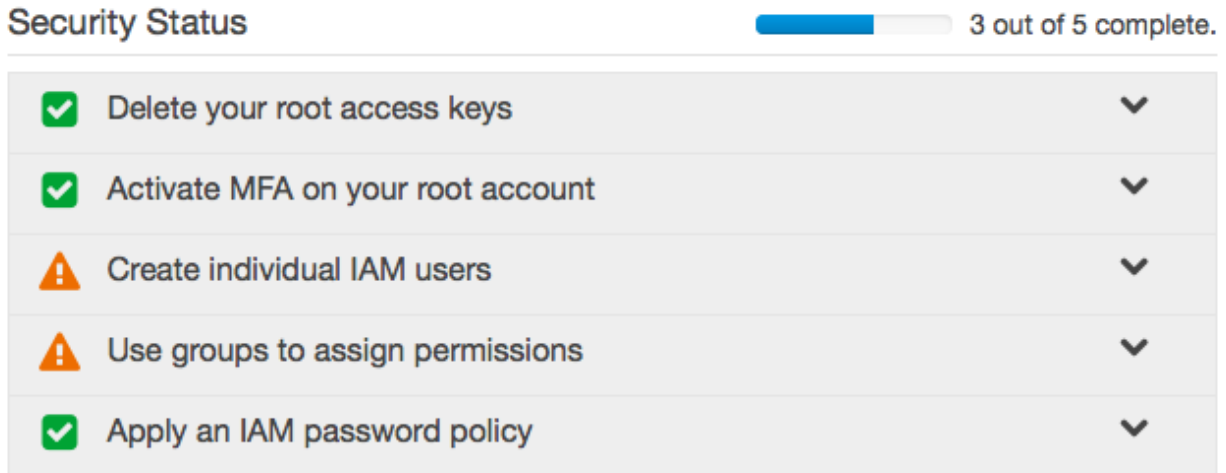


Fig. 1.5: Security Task List with Password Policy Complete

Creating a Group with Permissions

Our first task is thus to create a group. In the IAM console, in our security task list, click on *Use groups to assign permissions* and then on *Manage Groups*. Alternatively, you can simply click on *Groups* in the sidebar on the left.

On this new page, click on the *Create New Group* button. This will bring you to the Create New Group Wizard. I will call the new group **administrators** and then click *Next Step* at the bottom right of the screen. This brings us to the long list of permissions we can apply to our new group.

As we're creating a user policy for our own command-line access, we want to give ourselves full access to the cloud. We therefore click the checkbox next to *AdministratorAccess*, and click *Next Step*.

AWS will give us an overview of our new group, allowing us to click *Create Group* to actually finish the task.

Creating a User

Finally, we can create a user, and then add the user to our **administrators** group.

Much like with groups, we can either click on *Create individual IAM users* in the task list and then click on *Manage Users*, or use the *Users* link in the sidebar. On the new page, click *Create New Users*.

We only need a single user, so I will only fill in the top text field. Make sure that *Generate an access key for each user* checkbox at the bottom of the textfields is checked (the default).

With this new user created, you will be prompted to download the authentication credentials for the new user (the button at the bottom right of the screen on the new page). You will need this information when generating the static site project with our template. I recommend saving this information in a safe location, such as a password manager, as this information gives full access to your AWS account.

After you've downloaded and saved your credentials, your new user will appear in the list of users.

If you click on the new user, the full profile will be displayed.

Warning: The Amazon Resource Numbers (ARNs) associated with various cloud resources are unique, and should be kept semi-secret.

Use the *Add User to Group* button to add the new user to the **administrators** group.

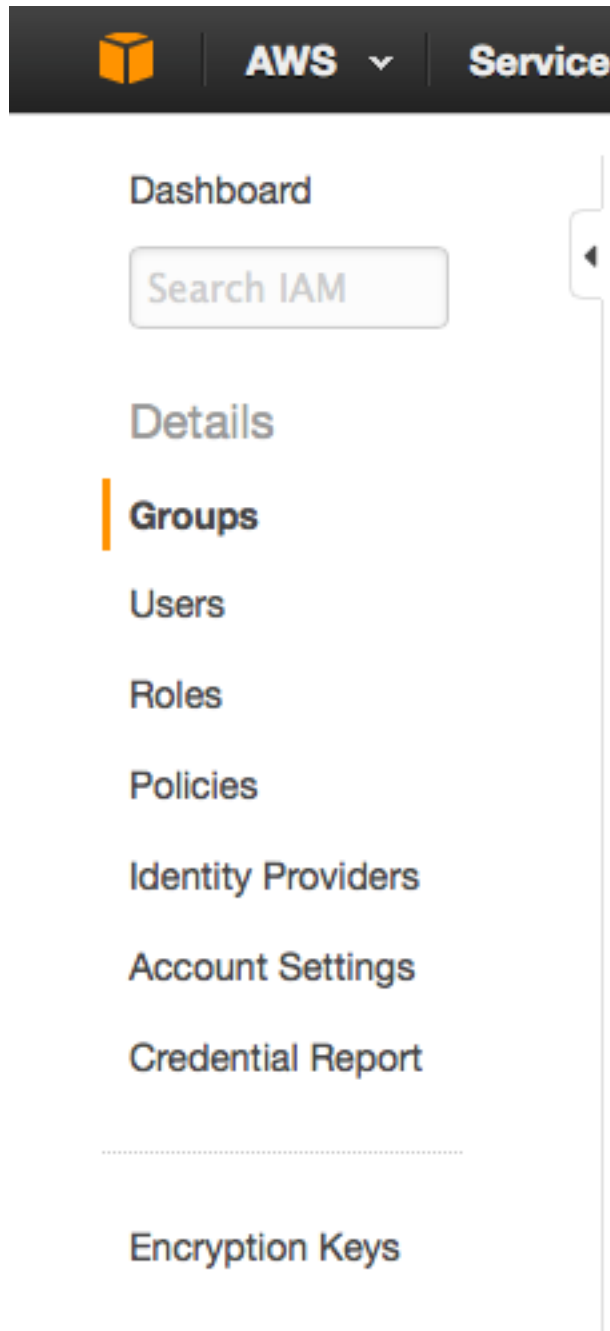


Fig. 1.6: Groups button in the IAM Console Sidebar

Attach Policy

Select one or more policies to attach. Each group can have up to 10 policies attached.











Filter: Policy Type ▾		Filter	Showing 187 results		
		Policy Name ⇅	Attached Entities ⇅	Creation Time ⇅	Edited Time ⇅
<input type="checkbox"/>		AdministratorAccess	1	2015-02-06 13:39 EST	2015-02-06 13:3...
<input type="checkbox"/>		AmazonAPIGatewa...	0	2015-07-09 13:34 EST	2015-07-09 13:3...
<input type="checkbox"/>		AmazonAPIGatewa...	0	2015-07-09 13:36 EST	2015-07-09 13:3...
<input type="checkbox"/>		AmazonAPIGatewa...	0	2015-11-11 18:41 EST	2015-11-11 18:4...
<input type="checkbox"/>		AmazonAppStream...	0	2015-02-06 13:40 EST	2015-02-06 13:4...
<input type="checkbox"/>		AmazonAppStream...	0	2015-02-06 13:40 EST	2015-02-06 13:4...
<input type="checkbox"/>		AmazonCognitoDev...	0	2015-03-24 13:22 EST	2015-03-24 13:2...
<input type="checkbox"/>		AmazonCognitoPo...	0	2015-03-24 13:14 EST	2015-03-24 13:1...
<input type="checkbox"/>		AmazonCognitoRea...	0	2015-03-24 13:06 EST	2015-03-24 13:0...
<input type="checkbox"/>		AmazonDMSCloud...	0	2016-01-07 18:44 EST	2016-01-07 18:4...
<div> Cancel Previous Next Step </div>					

Fig. 1.7: Groups button in the IAM Console Sidebar




Filter: Policy Type ▾		Filter	Showing 187 results		
		Policy Name ⇅	Attached Entities ⇅	Creation Time ⇅	Edited Time ⇅
<input checked="" type="checkbox"/>		AdministratorAccess	1	2015-02-06 13:39 EST	2015-02-06 13:3...
<input type="checkbox"/>		AmazonAPIGatewa...	0	2015-07-09 13:34 EST	2015-07-09 13:3...
<input type="checkbox"/>		AmazonAPIGatewa...	0	2015-07-09 13:36 EST	2015-07-09 13:3...

Fig. 1.8: AdministratorAccess Policy in the AWS IAM Create New Group Wizard

Review

Review the following information, then click **Create Group** to proceed.

Group Name	administrators	Edit Group Name
Policies	arn:aws:iam::aws:policy/AdministratorAccess	Edit Policies

Fig. 1.9: AdministratorAccess Policy Review in the AWS IAM Create New Group Wizard

Enter User Names:

1.
2.
3.
4.
5.

Maximum 64 characters each

☒ **Generate an access key for each user**

Fig. 1.10: Create User in AWS IAM Console

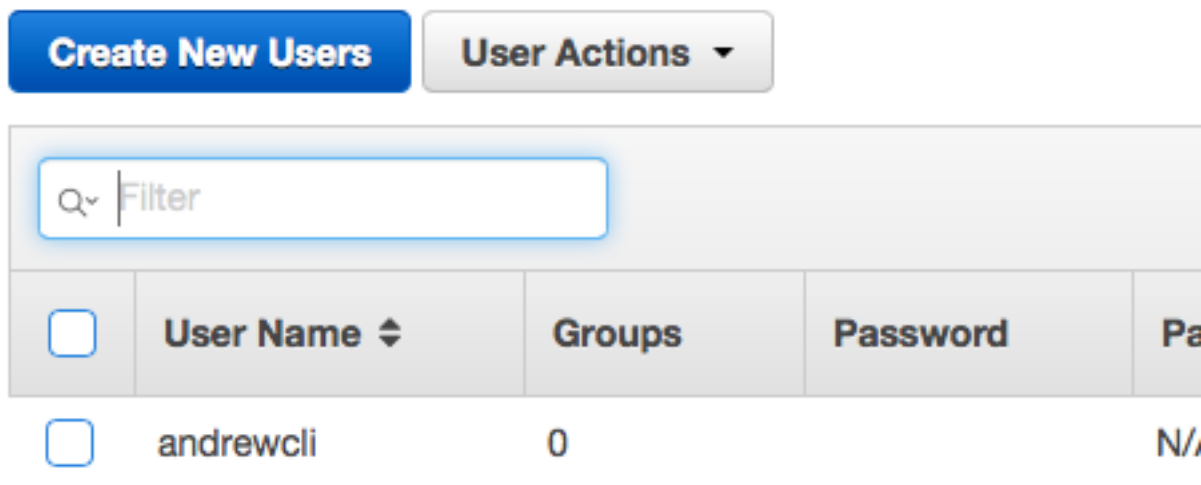


Fig. 1.11: New User in User List of AWS IAM Console

[IAM](#) > [Users](#) > **andrewcli**

▼ Summary

User ARN: arn:aws:iam::**Secret Number**:user/andrewcli
Has Password: No
Groups (for this user): 0
Path: /
Creation Time: 2015-12-05 10:32 EST

Groups

Permissions

Security Credentials

This user does not belong to any groups.

Add User to Groups

Fig. 1.12: New User Profile

Having created a user and assigned it a group with permissions, the task list for securing your account is now complete. Congratulations!

Security Status

5 out of 5 complete.

<input checked="" type="checkbox"/>	Delete your root access keys	▼
<input checked="" type="checkbox"/>	Activate MFA on your root account	▼
<input checked="" type="checkbox"/>	Create individual IAM users	▼
<input checked="" type="checkbox"/>	Use groups to assign permissions	▼
<input checked="" type="checkbox"/>	Apply an IAM password policy	▼

1.4 Generating a Static Website Project

With all of your *Prerequisites* installed, and an *AWS account with IAM user account*, we can actually generate the project that will quickly deploy your website.

1.4.1 Section Goals

In all cases, our goal is to use `cookiecutter` to download the `cookiecutter-static-site` project (this project!) to jump start our work by generating a static website project. The project will make deploying the static website to the cloud as easy as possible.

In the *first walkthrough* below, we will do this without any complications. In the *second walkthrough*, we will use `virtualenvwrapper` to separate our project tools from other projects.

1.4.2 `virtualenvwrapper` : to use, or not to use?

If you're planning to learn Python, or use any other Python projects, I strongly encourage you to use `virtualenvwrapper` (discussed and installed in the *Prerequisites section*).

If you have no intention of using multiple Python projects or programming Python, you do not need to use `virtualenvwrapper`. Even if you plan to generate multiple static websites using this project template, you will not need `virtualenvwrapper`.

1.4.3 Generating the project without `virtualenvwrapper`

Our first task is to install `cookiecutter`. In your terminal, write the following code, without the dollar sign (which is shown to designate the fact that this is a terminal).

```
$ pip install cookiecutter
```

Now that you have `cookiecutter` installed, we can use `cookiecutter` to use the `cookiecutter-static-site` template to create our new project.

```
$ cookiecutter gh:jambonsw/cookiecutter-static-site
```

`cookiecutter` will prompt you for information to customize your project. Below is an example of the questions asked (but may not reflect the most current questions!).

```
project_name [My Static Website]: JamBon Software
project_slug [jambon-software]: jambonsw
domain_name [example.com]: jambonsw.com
aws_access_key_id [XXXXXXXXXXXXXXXXXXXX]:
aws_secret_access_key [XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX]:
```

The project slug will be used to create a new directory (folder) with the project code in it. You can use the code below to navigate into directory using the terminal (you'll need to replace *jambonsw* with the slug of your own project).

```
$ cd jambonsw # replace with the value you specified when prompted by cookiecutter
```

Note: Your new project will automatically initiate a `git` project. If you do not want to use version control, or wish to use another version control system, you can easily delete the `git` project with the code below.

```
$ rm -rf .git # Warning: deletes with impunity
$ rm -f .gitignore # Warning: deletes with impunity
```

We then install all of the smaller Python tools (not listed in the *prerequisites section*!) necessary for our project.

```
$ pip install -r requirements.txt
```

Please note that whenever you work on this project, the scripts provided will need to have specific information available: specifically, the scripts expect to have several environment variables available in the environment. To make your life easier, all of these variables are specified in `env.sh`. You will need to enter the command below whenever you start working on this project (one of the major advantages of `virtualenvwrapper` is that we only need to specify the variables once).

```
$ source env.sh
```

Warning: Note that `env.sh` contains information that would allow someone to hack your website. Keep it safe and secure!

The `unenv.sh` script removes these secret variables from your environment. Use the command below when you stop working on your project (or you can simply close your terminal window).

```
$ source unenv.sh
```

Your project is ready to go! You can either read below to see how to do the same with `virtualenvwrapper`, or jump directly to instructions about *how to obtain a security certificate*.

1.4.4 Generating the project with *virtualenvwrapper*

Much like in the *first walkthrough above*, the goal is to use `cookiecutter` to generate a project from the `cookiecutter-static-site` project. However, this time we will separate the project logically according to terminal environment, making it easy to run other code projects.

Make sure that you have properly installed `virtualenvwrapper`, and that you've created the necessary environment variables, as specified in the [installation guide](#).

To start, we want to create a new virtual environment. In the code below, I name the project `jambonsw`, in anticipation of the fact that I will name the project slug the same. The project slug is primarily used to specify the name of the directory that contains the project. You may follow this convention, or name the environment as you please.

Warning: Do not copy the dollar signs or anything before the dollar signs in the code!

```
$ mkvirtualenv jambonsw
```

Your terminal will change to show you that you are now working in a virtual environment by printing the name of the environment in parentheses before the dollar sign.

Use the code below if you wish to leave the environment.

```
(jambonsw) $ deactivate
```

To enable an existing environment, use the `workon` command with the name of the environment.

```
$ workon jambonsw
```

In this environment, we then install `cookiecutter`. Note that `cookiecutter` will only be available if we're in the environment!

```
(jambonsw) $ cookiecutter gh:jambonsw/cookiecutter-static-site
```

As show in the *[example code in the first walkthrough above](#)*, `cookiecutter` will prompt you with various questions to get you started.

We use the project slug (specified by you when prompted) to enter the project directory (folder). Replace the `jambonsw` directory name in the code below with the slug of your own project.

```
(jambonsw) $ cd jambonsw
```

Note: Your new project will automatically initiate a `git` project. If you do not want to use version control, or wish to use another version control system, you can easily delete the `git` project with the code below.

```
$ rm -rf .git # Warning: deletes with impunity
$ rm -f .gitignore # Warning: deletes with impunity
```

To make our life easier in the long run, we can now associate this directory with the environment. We do so with the `setvirtualenvproject` command.

```
(jambonsw) $ setvirtualenvproject
```

If you use the `workon` command while in another directory, the command will automatically bring you to this directory.

We then install all of the smaller Python tools (not listed in the *[prerequisites section!](#)*) necessary for our project.

```
(jambonsw) $ pip install -r requirements.txt
```

One of the key advantages of using `virtualenvwrapper` is that we don't need to source `env.sh` whenever we choose to work on the project. Instead, we can get `virtualenvwrapper` to add the needed environment variables to the environment for us. We simply copy the file to a place where

Warning: The command below will replace the existing `postactivate` and `postdeactivate` file. If you edited it (perhaps while reading documentation elsewhere), please back it up or combine the files yourself.

```
(jambonsw) $ mv env.sh $WORKON_HOME/$(basename $VIRTUAL_ENV)/bin/postactivate
(jambonsw) $ mv unenv.sh $WORKON_HOME/$(basename $VIRTUAL_ENV)/bin/postdeactivate
```

Your project is ready to go! Before you can deploy the website, however, you'll need to *obtain a security certificate*.

1.5 Obtaining Security Certificates

Unsecured HTTP (the protocol that computers use to communicate over the web) is being phased out. Securing your website is no longer a recommendation; encrypting communication is becoming a modern requirement for websites.

1.5.1 Section Goals

The goal of this section is to upload a security certificate for AWS to use to encrypt communication to and from your website.

To obtain a certificate, we must first buy the right to a certificate from a Certificate Authority (CA (Certificate Authority)). We then use OpenSSL to generate cryptographic keys, which in turn allows us to generate a Certificate Signing Request (CSR (Certificate Signing Request)). We return to the CA with our CSR to obtain (generate and activate) our certificate.

We will then upload three things to AWS:

1. a private key
2. a security certificate
3. a certificate chain proving the validity of our certificate

Using a classical CA will cost you \$10-20/year. If you're feeling daring (and comfortable with the command line) you may be interested in [Let's Encrypt](#) (LE (Let's Encrypt)). LE is a free, open-source CA. However, LE is currently in beta, and can be a bit finicky. Once I think LE is ready for you, I will update this section with instructions on how to use it. In the meantime, you're on your own!

1.5.2 Obtaining a Security Certificate from a Certificate Authority

Warning: At the end of the section, you will need to be able to access the email associated with your domain on whois. Please take a moment to ensure that you are able to do so. If you don't know what this means, please reach out to your domain registrar's support team.

Security certificates rely on [Public-Key Cryptography](#). To obtain a security certificate, we first need to generate a set of keys. We will then create a Certificate Signing Request (CSR) for the keys, which will be the information a Certificate Authority needs to generate a certificate for us.

Before any of that, however, we need to buy the ability to request a certificate from a Certificate Authority (CA).

Buying a Certificate

When you buy a certificate from a CA, you're not actually buying the certificate; you're buying the right to generate and activate a certificate using a CSR in the future.

There are many places to buy a certificate. Among them are (alphabetically):

- [Comodo](#)
- [DigiCert](#)
- [GeoTrust](#)
- [RapidSSL](#)
- [Thawte](#)

Generally, you can find better deals (price-wise) when buying through a domain registrar. I currently use [NameCheap's Certificate options](#), and would recommend either [Comodo PositiveSSL](#) for \$9/yr or [RapidSSL](#) for \$10.95/yr. If you're looking to secure multiple domains, you'll want to evaluate [other options](#), as AWS only allows a single certificate per website. Of course, if you already have a domain registrar (other than NameCheap), it's worth seeing what they have to offer. I furthermore recommend reaching out to your registrar's support to discuss your options.

Once you've bought the ability to activate a certificate, you are ready to proceed.

Generating a Certificate Signing Request

This section requires the use of OpenSSL, discussed in the *Prerequisites section*.

All of this work will occur in the certificates directory of the *project you generated last section*. You may optionally be in the virtual environment you created.

```
$ cd certificates
```

To start, we generate a private key of length 2048 bits using the RSA algorithm. We store it in a file called `private-key.pem`.

```
$ openssl genrsa 2048 > private-key.pem
```

Now that we have a private key, we can generate a CSR, creatively named `csr.pem`.

```
$ openssl req -new -key private-key.pem -out csr.pem
```

You will be prompted for the values of your certificate. The Common Name field must be the main domain name for the website. Not all of the fields need be specified. In many cases, the CA will ask that the challenge password be left blank. When answering the questions for JamBon Software, I entered the following values:

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:MA
Locality Name (eg, city) []:Boston
Organization Name (eg, company) [Internet Widgits Pty Ltd]:JamBon Software
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:jambonsw.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Now that you have a CSR, you will have to go to the website where you bought your certificate, and activate the certificate. At some point in the process, they will ask you for the certificate request: you will either upload the `csr.pem` file, or else copy the contents of the file into a textfield.

At the end of the activation process, you'll have the option of picking how the CA will verify your ownership of the domain and how to send you your certificate. The simplest of these options will be to

Once you have given the CSR to a CA, the CA will begin the process of generating your certificate.

Preparing the Certificate for AWS

It would be too simple if your CA simply sent you what you needed. In this section we'll take a look at what your CA might send you, and what to do with it in that case.

In all cases, the goal will be to have three files:

1. a private key
2. a security certificate
3. a certificate chain proving the validity of our certificate

AWS specifies that all of these files must be in the PEM format.

Note: The text in the sections below will refer to certificates with the prefix `jambonsw_com`, as this walkthrough is based on JamBon Software's own website. You will wish to name your certificates according to your domain. If your domain is `andrew.pinkham.com`, you'd want to name the certificates with the prefix `andrew_pinkham_com`.

You likely only need to read one of the sections below, depending on what your CA sends you.

Receiving a Certificate and Bundle

If you receive two files, one will be the certificate and the other will be the chain bundle. When I ordered the [Comodo PositiveSSL](#), the two files I received were:

- `jambonsw_com.crt`
- `jambonsw_com.ca-bundle`

Save both files to the `certificates` directory of your generated project. In my case, both of the files I received were already in PEM format. However, just to make sure, we can use OpenSSL to convert them.

```
$ openssl x509 -in jambonsw_com.crt -outform pem -out jambonsw_com_cert.pem
$ openssl x509 -in jambonsw_com.ca-bundle -outform pem -out jambonsw_com_ca_chain.pem
```

You could then delete the originals.

```
$ rm jambonsw_com.crt
$ rm jambonsw_com.ca-bundle
```

In the `certificates` directory, you should now have:

- `csr.pem`
- `jambonsw_com_ca_chain.pem`
- `jambonsw_com_cert.pem`
- `private-key.pem`

You are now ready to *upload your certificate to AWS*.

Receiving a Certificate, Multiple Chain Certs, and a Root

CAs (Certificate Authorities) will frequently send the CA chain as separate files. Your goal is to concatenate them into a single file, from least to most important. If the email you receive does not tell you what the order is, you will have to read the documentation on your CA's site, or contact their support, to determine what that order is.

One of the bundles I received from Comodo was:

- AddTrustExternalCARoot.crt (Root CA Certificate)
- COMODORSAAAddTrustCA.crt (Intermediate CA Certificate)
- COMODORSADomainValidationSecureServerCA.crt (Intermediate CA Certificate)
- jambonsw_com.crt

We start by ensuring that all of the files are in PEM format.

```
$ openssl x509 -in COMODORSADomainValidationSecureServerCA.crt -outform pem -out COMODORSADomainValidationSecureServerCA.pem
$ openssl x509 -in COMODORSAAAddTrustCA.crt -outform pem -out COMODORSAAAddTrustCA.pem
$ openssl x509 -in AddTrustExternalCARoot.crt -outform pem -out AddTrustExternalCARoot.pem
$ openssl x509 -in jambonsw_com.crt -outform pem -out jambonsw_com_cert.pem
```

Now that all our files are in the right format, we can simply concatenate the files into a single CA chain.

```
$ cat COMODORSADomainValidationSecureServerCA.pem > jambonsw_com_ca_chain.pem
$ cat COMODORSAAAddTrustCA.pem >> jambonsw_com_ca_chain.pem
$ cat AddTrustExternalCARoot.pem >> jambonsw_com_ca_chain.pem
```

Finally, I opt to delete all of the unnecessary files (having backed up the email from Comodo).

```
$ rm AddTrustExternalCARoot.crt
$ rm AddTrustExternalCARoot.pem
$ rm COMODORSAAAddTrustCA.crt
$ rm COMODORSAAAddTrustCA.pem
$ rm COMODORSADomainValidationSecureServerCA.crt
$ rm COMODORSADomainValidationSecureServerCA.pem
$ rm jambonsw_com.crt
```

In the `certificates` directory, you should now have:

- csr.pem
- jambonsw_com_ca_chain.pem
- jambonsw_com_cert.pem
- private-key.pem

You are now ready to *upload your certificate to AWS*.

Uploading your Certificate

Now that you have a private key, a certificate, and a certificate chain, you can upload the certificate to AWS to secure your website.

To do so, simply move to the root directory of your project (where the `Makefile` is), and enter the command below.

```
$ make upload-cert
```

Ta-da! Your certificate has been uploaded to AWS. You are now ready to *deploy your website*.

1.6 Deploying your Static Site to the Cloud

You have the tools. You have the accounts. You have security. Let's put some content in the cloud.

1.6.1 Section Goals

We will first use the scripts to create a Cloud Formation stack. This will create all of the cloud resources we need to store our website.

We will then upload the website to the cloud.

1.6.2 Preparing for Work

If you're using `virtualenvwrapper`, simply activate the environment.

```
$ workon jambonsw # Replace jambonsw with the name of your own env
$ # if you don't remember the name of your env
$ # type workon, and then tap the tab button on your keyboard twice
```

If you're not using `virtualenvwrapper`, you need to load the appropriate environment variables.

```
$ cd Path/To/Your/Project
$ source env.sh
```

1.6.3 Creating a CloudFormation Stack

To create a CloudFormation stack, we need a script (currently in `your_project_dir/certificates/cloudformation_build`) and parameters for that script (will be created at `your_project_dir/certificates/cloudformation_parameters.json`).

To make your life as simple as possible, a script has been supplied to generate the parameters for you (this works as long as you've already uploaded a security certificate).

```
$ ./generate-params.sh
```

If the script succeeds, it will delete itself after creating `your_project_dir/certificates/cloudformation_parameters.json`.

Note: If you're using `git`, I recommend adding the new file to your repository.

```
$ git add certificates/cloudformation_parameters.json
$ git commit -m "Generated CloudFormation parameters."
```

With all the right files in place, we can now create the CloudFormation stack with the simple command listed below.

```
$ make create-stack
```

This can take up to 20 minutes to complete.

1.6.4 Pointing your Domain to Amazon's Nameservers

Now that you have a set of resources in the cloud, you need to point your domain name at the cloud. You are setting the nameservers of your domain.

To get the Amazon Nameservers for your website, simply use the command below.


```
$ make dns
```

You will need to go to you domain's registrar and specify these nameservers. If you don't know how to do this, look through the FAQ section of your registrar's website, or contact support.

1.6.5 Uploading Content to the Cloud

We now have the infrastructure for a website, but we've not actually put any content in the cloud yet. The command below changes that, by uploading all of the content in the *your_project_dir/content/* directory. By default, the template includes an extremely basic webpage.

```
$ make
```

And with that last command issued, you have successfully deployed your website to the cloud.

Congratulations!

1.7 Updating Your Static Website

Notes!

update content and then `make sync`

update `cloudformation_build.json` and `make stack`

Example case: add MX record

1.8 Contributing to the Cookiecutter Static Site Project

Notes!

Fork project on Github

```
git co -m issue_101
```

```
git co -m new_feature
```

Issue PR from branch!

Indices and tables

- `genindex`
- `modindex`
- `search`