IaC (Infrastructure as Code)

The best way to think of Infrastructure as Code is in the literal sense. There is a long history of configuration languages that manage infrastructure. When I worked at Caltech in 2000, I used tools like radmind[120], CFEngine[121] and [.

Newer generation tools include Terraform[122] and Pulumi[123]. The general concept is that the application software and the deployment environment benefit from automation. Humans make mistakes, but automation is forever.

Learn what IAC is in the following screencast.

Video Link: https://www.youtube.com/watch?v=rfZWRpN6Da4[124]

Learn what IAC in the real world is in the following screencast.

Video Link: https://www.youtube.com/watch?v=nrCYVyBuOIw[125]

Launch a VM with Terraform in the following screencast.

Video Link: https://www.youtube.com/watch?v=mh4qf0MS0F4[126]

What is Continuous Delivery and Continuous Deployment?

Let's build on the knowledge of Continuous Delivery from Chapter one. It is a technique that leverages several powerful tools continuous integration, IaC, and Cloud Computing. Continuous Delivery lets the cloud infrastructure be defined as code and allows for near real-time changes of both code and new environments.

Continuous Delivery for Hugo Static Site from Zero

Hugo[127] is a popular static site generator. This tutorial will guide you through using AWS Cloud9[128] to create a Hugo website and develop against it using the cloud development environment. The final step will be the set up a continuous integration pipeline using AWS Code Pipeline[129].

[120] http://www.radmind.org
[121] https://en.wikipedia.org/wiki/CFEngine
[122] https://www.ansible.com
[123] https://www.pulumi.com
[124] https://www.youtube.com/watch?v=rfZWRpN6Da4
[125] https://www.youtube.com/watch?v=nrCYVyBuOIw
[126] https://www.youtube.com/watch?v=mh4qf0MS0F4
[127] https://gohugo.io/
[128] https://aws.amazon.com/cloud9/
[129] https://aws.amazon.com/codepipeline/

Note these steps will be similar for other cloud environments or your OS X laptop, but this particular tutorial targets AWS Cloud9.

The steps described next appear in this screencast AWS Hugo Continuous Delivery.

Video Link: https://www.youtube.com/watch?v=xiodvLdPnvI[130]

- Step 1: Launch an AWS Cloud9 Environment

Use the AWS Free Tier and a Cloud9 Environment with the defaults.

- Step2: Download the `hugo` binary and put it in your Cloud9 path.

Go to the latest releases of `hugo` https://github.com/gohugoio/hugo/releases[131]. Download the latest release using the `wget` command. It should look similar to the following:

```
wget https://github.com/gohugoio/hugo/releases/download/v0.79.1/hugo_0.79.1_Linux-64\
bit.tar.gz
```

Note that you shouldn't just blindly cut and paste the code above! Make sure you get the latest release or if not on Cloud9, use the appropriate version

Now put this file in your `~/.bin` directory using these commands (again make sure you put your version of hugo here: i.e. `hugo_0.99.x_Linux-32bit.tar.gz`):

```
tar xzvf hugo_<VERSION>.tar.gz
mkdir -p ~/bin
mv ~/environment/hugo . #assuming that you download this into ~/environment
which hugo              #this shows the `path` to hugo
```

The output of `which hugo` should be something like:

```
ec2-user:~/environment $ which hugo
~/bin/hugo
```

Finally, check to see that the version flag works as a basic sanity check. This output is what it looks like on my cloud9 machine (*your version number will likely be different*)

[130] https://www.youtube.com/watch?v=xiodvLdPnvI
[131] https://github.com/gohugoio/hugo/releases

```
ec2-user:~/environment $ hugo version
Hugo Static Site Generator v0.79.1-EDB9248D linux/386 BuildDate: 2020-12-19T15:41:12Z
```

These steps get you access to `hugo`, and you can run it like any other command-line tool. If you cannot or get stuck, refer to the screencast later and look at the quickstart guide[132].

- Step3: Make a `hugo` website locally and test it in Cloud9

One great thing about `hugo` is that it just a `go` binary. It makes it simple to both develop and deploy `hugo` sites. The following section derives from the official `hugo` quickstart guide[133].

A. Create a new repo in Github and clone it into your environment. Change into it via the "cd" command. Add a `.gitignore` file with the word `public` in it. This step will stop the `public` directory from checking into the repo.
B. Create a new site using the following command: `hugo new site quickstart`
C. Add a theme (you could swap this part with any theme[134] you want).

```
cd quickstart
git submodule add https://github.com/budparr/gohugo-theme-ananke.git themes/ananke
echo 'theme = "ananke"' >> config.toml
```

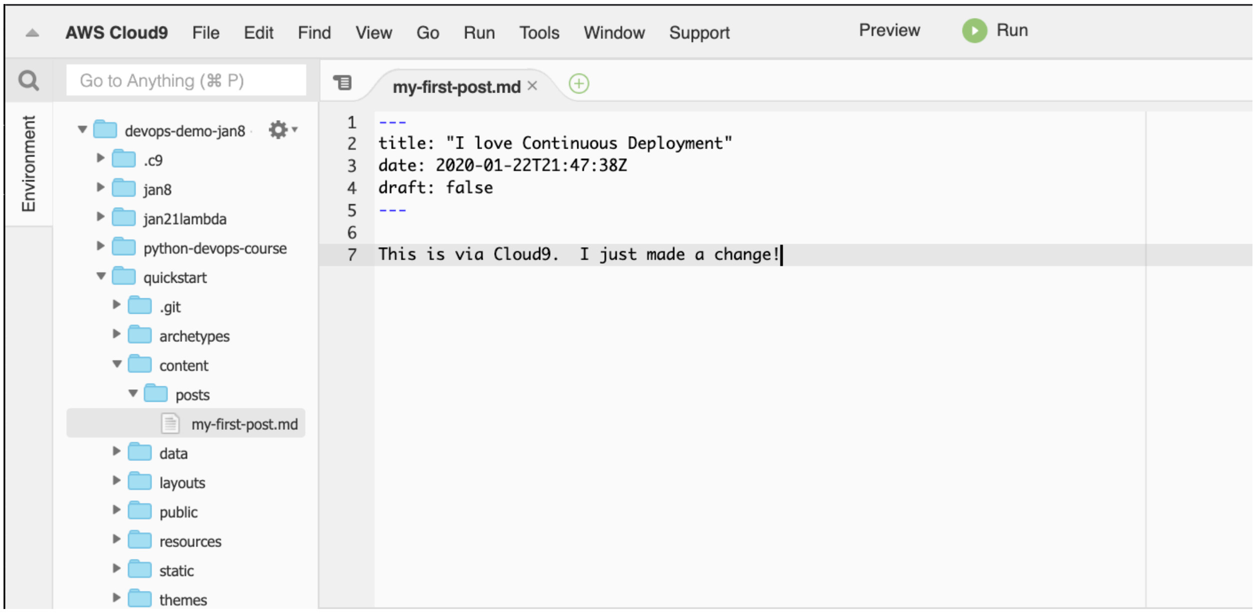
- Step4: Create a post

To create a new blog post, type the following command.

```
hugo new posts/my-first-post.md
```

This post is easily editable inside of AWS Cloud9, as shown in the following screenshot.

[132] https://gohugo.io/getting-started/installing#step-2-download-the-tarball
[133] https://gohugo.io/getting-started/quick-start/
[134] https://themes.gohugo.io/



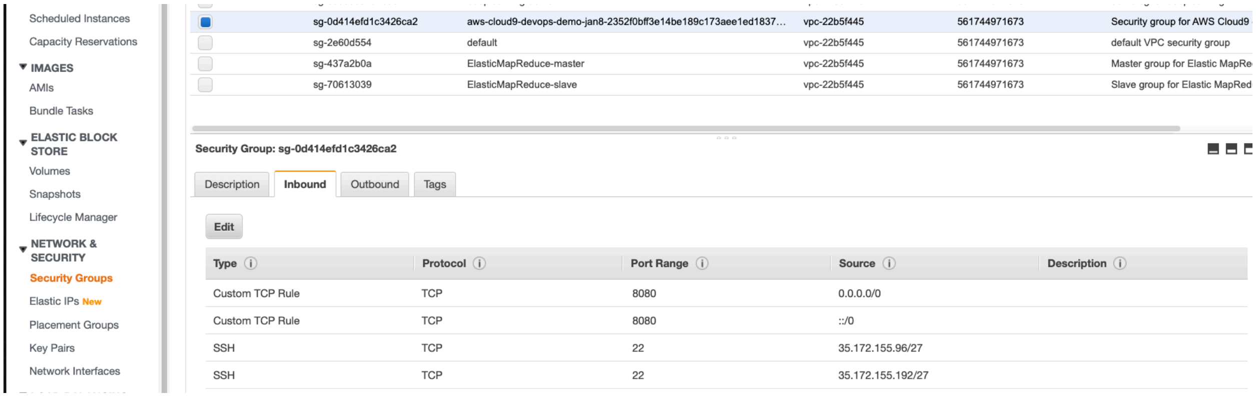
```
---
title: "I love Continuous Deployment"
date: 2020-01-22T21:47:38Z
draft: false
---

This is via Cloud9.  I just made a change!
```

aws cloud9 edit Hugo post

- Step5: Run Hugo locally in Cloud9

Up to this point, things have been relatively straightforward. In this section, we are going to run `hugo` as a development server. This step will require us to open up a port on EC2 security groups. To do this step, proceed with the following tasks.

A. Open a new tab on the AWS Console and type in `EC2` and scroll down to security groups and look for the security group with the same name as your AWS Cloud9 environment as shown:



☑	sg-0d414efd1c3426ca2	aws-cloud9-devops-demo-jan8-2352f0bff3e14be189c173aee1ed1837...	vpc-22b5f445	561744971673	Security group for AWS Cloud9
□	sg-2e60d554	default	vpc-22b5f445	561744971673	default VPC security group
□	sg-437a2b0a	ElasticMapReduce-master	vpc-22b5f445	561744971673	Master group for Elastic MapRe
□	sg-70613039	ElasticMapReduce-slave	vpc-22b5f445	561744971673	Slave group for Elastic MapRed

Security Group: sg-0d414efd1c3426ca2

Type	Protocol	Port Range	Source	Description
Custom TCP Rule	TCP	8080	0.0.0.0/0	
Custom TCP Rule	TCP	8080	::/0	
SSH	TCP	22	35.172.155.96/27	
SSH	TCP	22	35.172.155.192/27	

AWS Cloud9 environment

.

B. Open up via new TCP rule port `8080` and the `edit` button. This step will allow us to browse to port `8080` to preview our website as we develop it locally on AWS Cloud9.

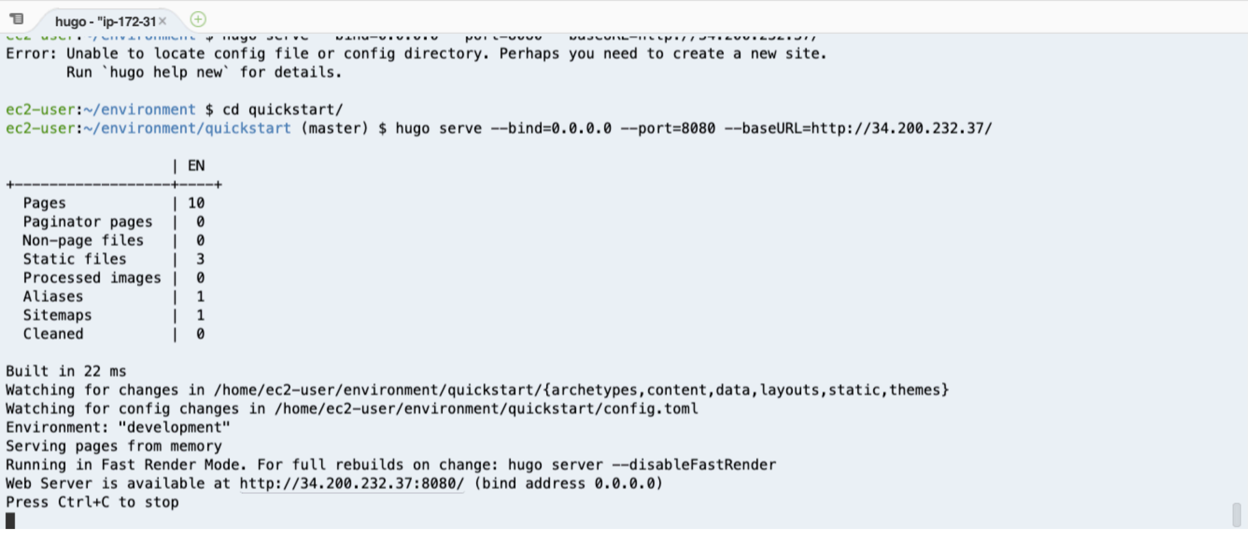
C. Navigate back to AWS Cloud9 and run the following command to find out the IP Address (we will use this IP Address when we run `hugo`). *Note you can also find your IP Address from the AWS Console for EC2)*

~~~
curl ipinfo.io
````

You should see something like this (*but with a different IP Address*).
~~~

```
ec2-user:~/environment $ curl ipinfo.io
{
“ip”: “34.200.232.37”,
“hostname”: “ec2-34-200-232-37.compute-1.amazonaws.com”,
“city”: “Virginia Beach”,
“region”: “Virginia”,
“country”: “US”,
“loc”: “36.8512,-76.1692”,
“org”: “AS14618 Amazon.com, Inc.”,
“postal”: “23465”,
“timezone”: “America/New_York”,
“readme”: “https://ipinfo.io/missingauth”
```

~~~
D.  Run `hugo` with the following options; you will need to swap this IP Address out\
 with the one you generated earlier.  Notice that the `baseURL` is essential so you \
can test navigation.

```bash
hugo serve --bind=0.0.0.0 --port=8080 --baseURL=http://34.200.232.37/
~~~

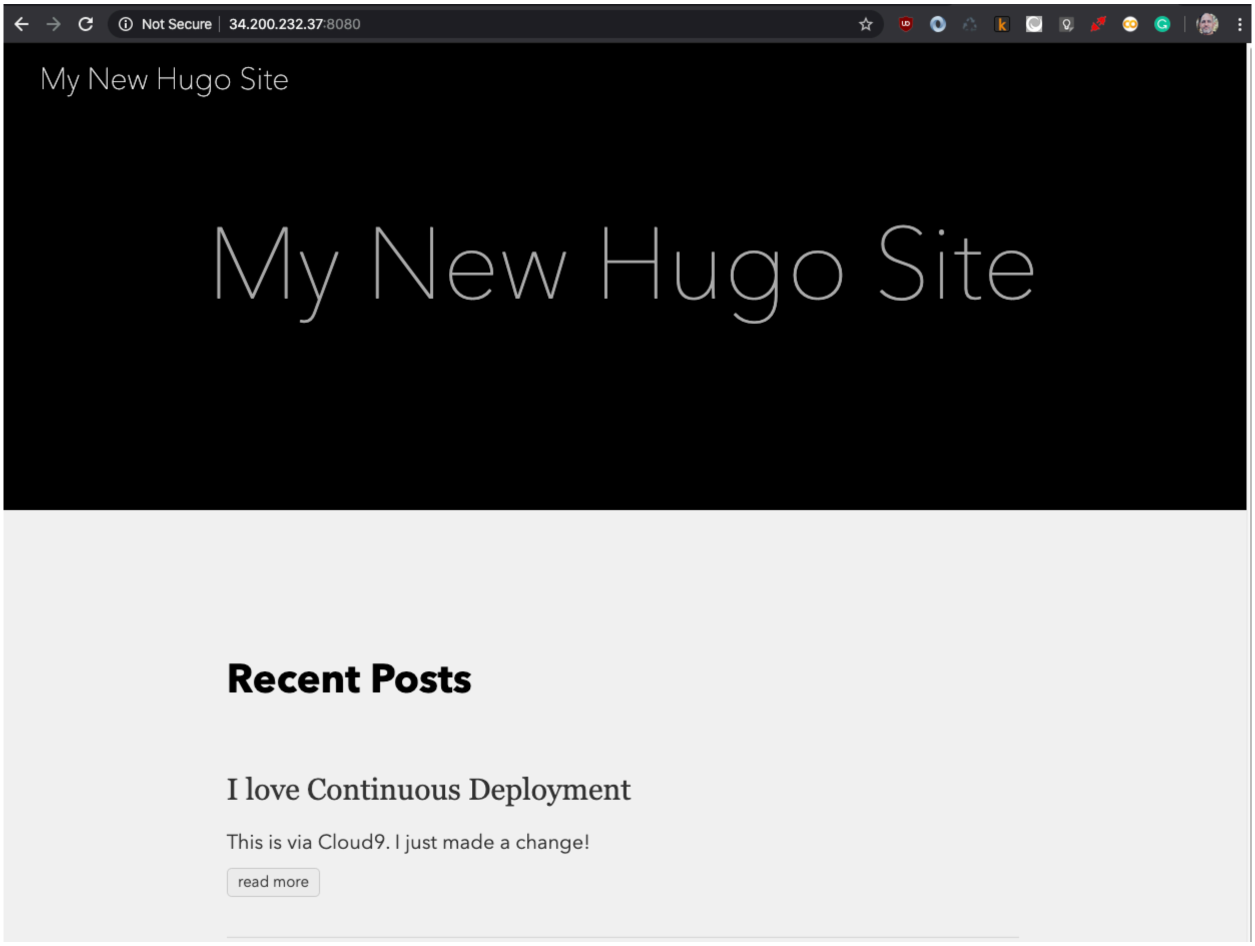
If this was successful, you should get something similar to the following output.



```
ec2-user:~/environment $ hugo serve --bind=0.0.0.0 --port=8080 --baseURL=http://34.200.232.37/
Error: Unable to locate config file or config directory. Perhaps you need to create a new site.
       Run `hugo help new` for details.

ec2-user:~/environment $ cd quickstart/
ec2-user:~/environment/quickstart (master) $ hugo serve --bind=0.0.0.0 --port=8080 --baseURL=http://34.200.232.37/

                   | EN
+------------------+----+
  Pages            | 10
  Paginator pages  |  0
  Non-page files   |  0
  Static files     |  3
  Processed images |  0
  Aliases          |  1
  Sitemaps         |  1
  Cleaned          |  0

Built in 22 ms
Watching for changes in /home/ec2-user/environment/quickstart/{archetypes,content,data,layouts,static,themes}
Watching for config changes in /home/ec2-user/environment/quickstart/config.toml
Environment: "development"
Serving pages from memory
Running in Fast Render Mode. For full rebuilds on change: hugo server --disableFastRender
Web Server is available at http://34.200.232.37:8080/ (bind address 0.0.0.0)
Press Ctrl+C to stop
```

hugo local

E. Open a new tab in your browser and type paste in the URL in the output. In my production, it is `http://34.200.232.37:8080/`, but it will be *different for you*.



My New Hugo Site

My New Hugo Site

Recent Posts

I love Continuous Deployment

This is via Cloud9. I just made a change!

hugo website

If you edit the markdown file, it will render out the changes live. This step allows for an interactive development workflow.

- Step6: Create Static Hosted Amazon S3 website and deploy to the bucket.

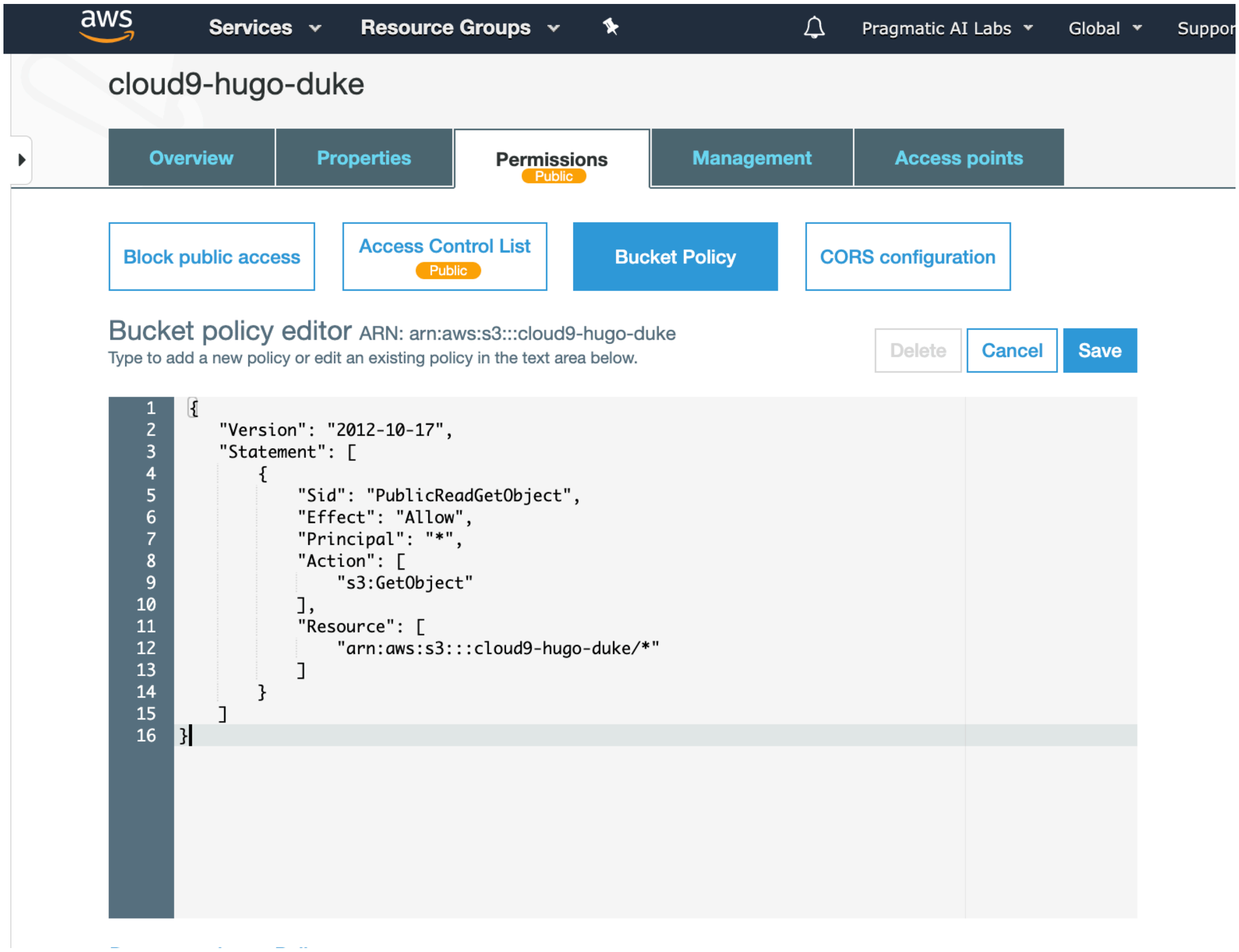
The next thing to do is to deploy this website directory to an AWS S3 bucket. You can follow the instructions here on creating an s3 bucket and set it up for hosting[135].

This step also means setting a `bucket policy` via the bucket policy editor, as shown below. The name of your bucket *WILL NOT BE `cloud9-hugo-duke`* you must change this.

[135] https://docs.aws.amazon.com/AmazonS3/latest/user-guide/static-website-hosting.html

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::cloud9-hugo-duke/*"
            ]
        }
    ]
}
```

The bucket policy editor workflow looks as follows.



Bucket policy editor ARN: arn:aws:s3:::cloud9-hugo-duke
Type to add a new policy or edit an existing policy in the text area below.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::cloud9-hugo-duke/*"
            ]
        }
    ]
}
```

bucket policy editor

- Step7: Deploy the website manually before it becomes fully automated

With automation, it is essential to manually write down the steps for a workflow before fully automating it. The following items will need confirmation:

A. The `config.toml` will need to be edited, as shown below. Note that your s3 bucket URL will be different.

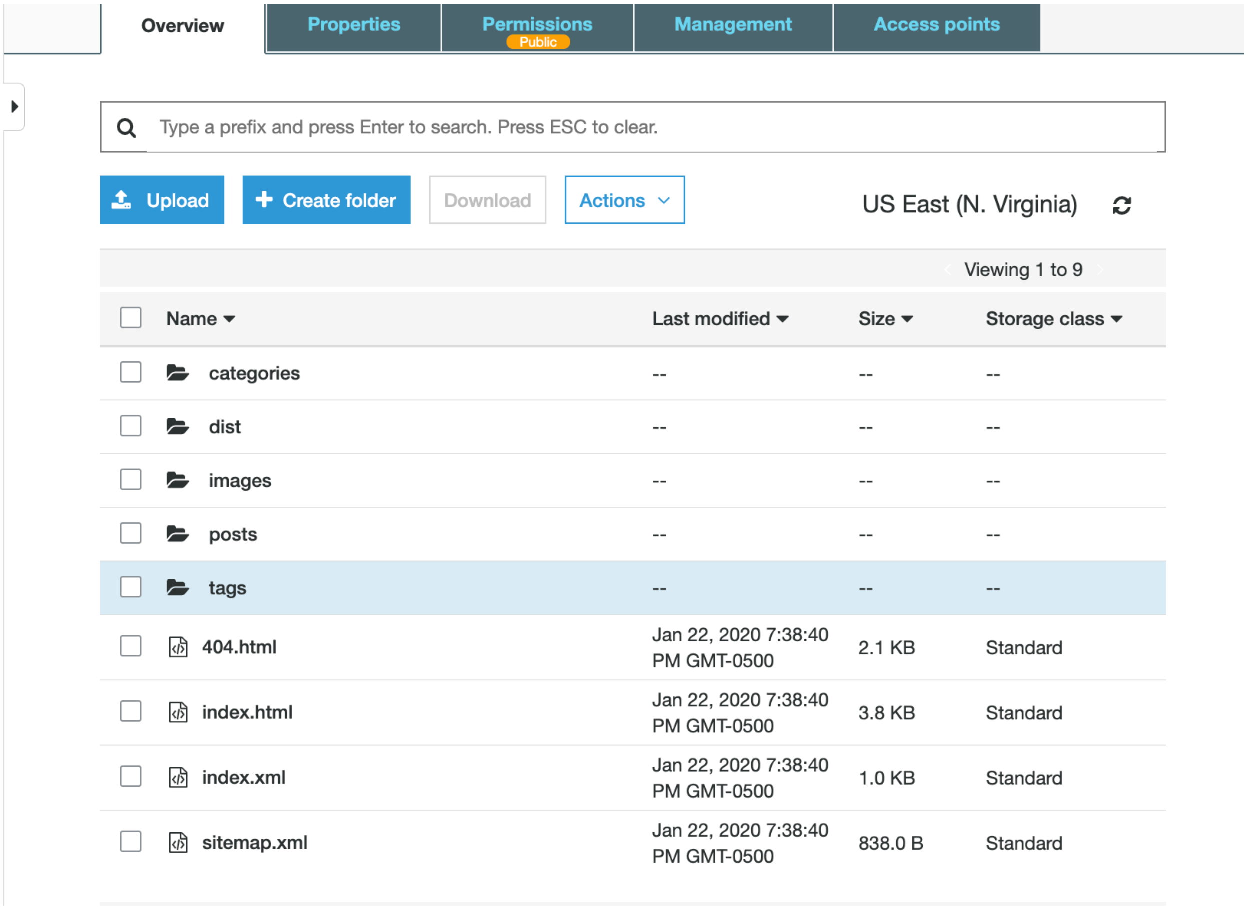
```
baseURL = "http://cloud9-hugo-duke.s3-website-us-east-1.amazonaws.com"
languageCode = "en-us"
title = "My New Hugo Sit via AWS Cloud9"
theme = "ananke"

[[deployment.targets]]
# An arbitrary name for this target.
name = "awsbucket"
URL = "s3://cloud9-hugo-duke/?region=us-east-1" #your bucket here
```

B. Now, you can deploy by using the built-in `hugo deploy` command. The deployment command output should look like this after you run `hugo deploy`. You can read more about the `deploy` command in the official docs[136].

```
ec2-user:~/environment/quickstart (master) $ hugo deploy                   \

Deploying to target "awsbucket" (s3://cloud9-hugo-duke/?region=us-east-1)  \

Identified 15 file(s) to upload, totaling 393 kB, and 0 file(s) to delete. \

Success!
```

The contents of the AWS S3 bucket should look similar to this.

[136] https://gohugo.io/hosting-and-deployment/hugo-deploy/



Name	Last modified	Size	Storage class
categories	--	--	--
dist	--	--	--
images	--	--	--
posts	--	--	--
tags	--	--	--
404.html	Jan 22, 2020 7:38:40 PM GMT-0500	2.1 KB	Standard
index.html	Jan 22, 2020 7:38:40 PM GMT-0500	3.8 KB	Standard
index.xml	Jan 22, 2020 7:38:40 PM GMT-0500	1.0 KB	Standard
sitemap.xml	Jan 22, 2020 7:38:40 PM GMT-0500	838.0 B	Standard

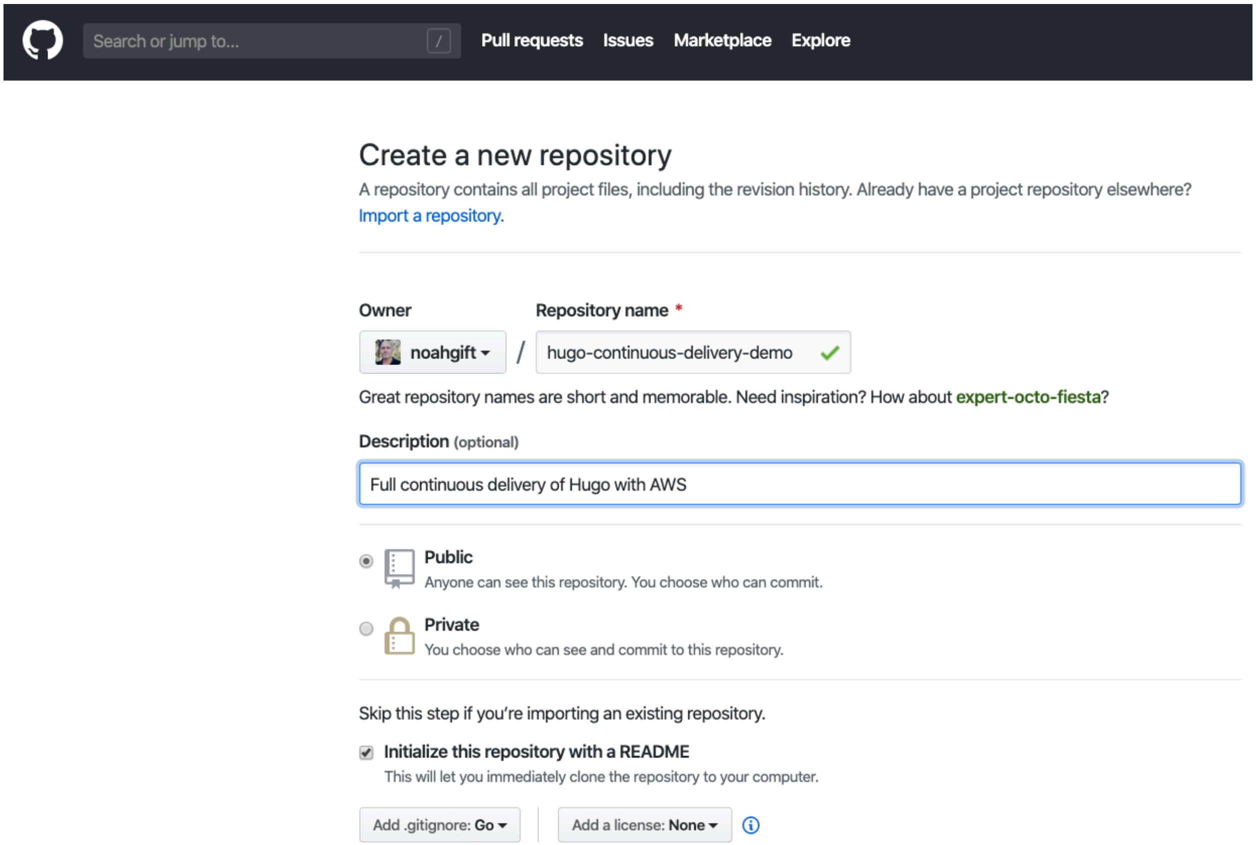
bucket contents

The website demonstrated in this tutorial is visible here: http://cloud9-hugo-duke.s3-website-us-east-1.amazonaws.com/[137]

- Step8: Check into Github

A. Create a new Github repo (and add `.gitignore`)

[137] http://cloud9-hugo-duke.s3-website-us-east-1.amazonaws.com/



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: noahgift / Repository name *: hugo-continuous-delivery-demo ✓

Great repository names are short and memorable. Need inspiration? How about expert-octo-fiesta?

Description (optional): Full continuous delivery of Hugo with AWS

● Public
Anyone can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☑ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

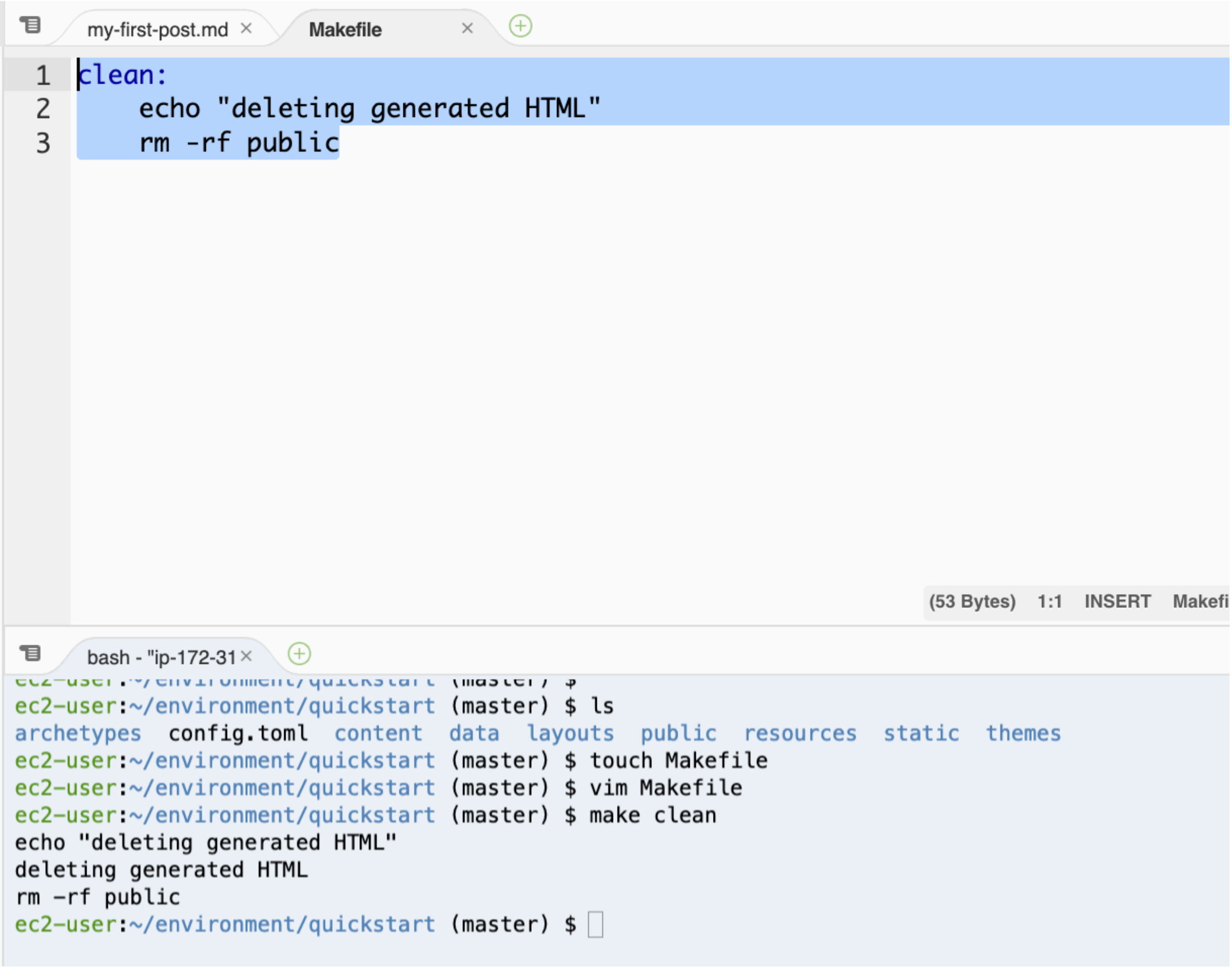
Add .gitignore: Go | Add a license: None

add git repo

(Remember to double check you added a .gitignore[138] *and added* `public` *to ‘.gitignore)*

B. In AWS Cloud9, in the quickstart directory, create a `Makefile` with a `clean` command. This will `rm -rf` the public HTML directory that `hugo` creates. You don't want to check this into source control.

[138] https://github.com/noahgift/hugo-continuous-delivery-demo/blob/master/.gitignore



```
clean:
    echo "deleting generated HTML"
    rm -rf public
```

```
ec2-user:~/environment/quickstart (master) $
ec2-user:~/environment/quickstart (master) $ ls
archetypes  config.toml  content  data  layouts  public  resources  static  themes
ec2-user:~/environment/quickstart (master) $ touch Makefile
ec2-user:~/environment/quickstart (master) $ vim Makefile
ec2-user:~/environment/quickstart (master) $ make clean
echo "deleting generated HTML"
deleting generated HTML
rm -rf public
ec2-user:~/environment/quickstart (master) $ 
```

create Makefile

```
clean:
        echo "deleting generated HTML"
        rm -rf public
```

C. Now run `make clean` to delete the `public` directory and all of the source code `hugo` generated (don't worry, it regenerates HTML anytime you run `hugo`).

5. Add the source code and push to Github.

Typically I get the "lay of the land" before I commit. I do this by running `git status`. Here is my output in the next section. *You can see that I need to `Makefile` `archetypes` `config.toml` and `content/`.*

```
ec2-user:~/environment/quickstart (master) $ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitmodules
        new file:   themes/ananke

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Makefile
        archetypes/
        config.toml
        content/
```

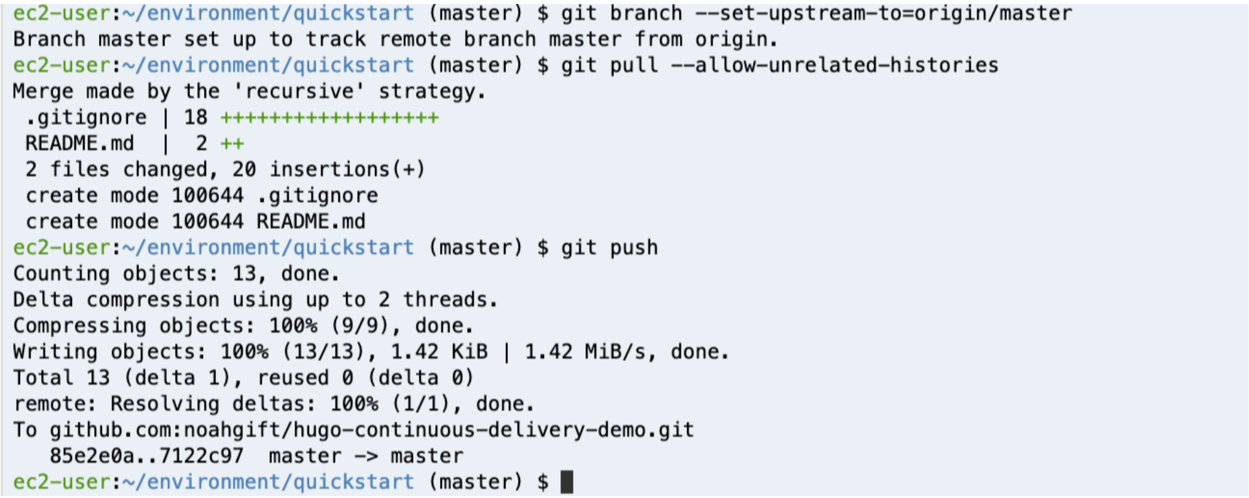
I add them by typing the command `git add *`. You can see below that this will add all of those files and directories:

```
ec2-user:~/environment/quickstart (master) $ git add *
ec2-user:~/environment/quickstart (master) $ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitmodules
        new file:   Makefile
        new file:   archetypes/default.md
        new file:   config.toml
        new file:   content/posts/my-first-post.md
        new file:   themes/ananke
```

Now push these files by doing the following command.

```
git push
```

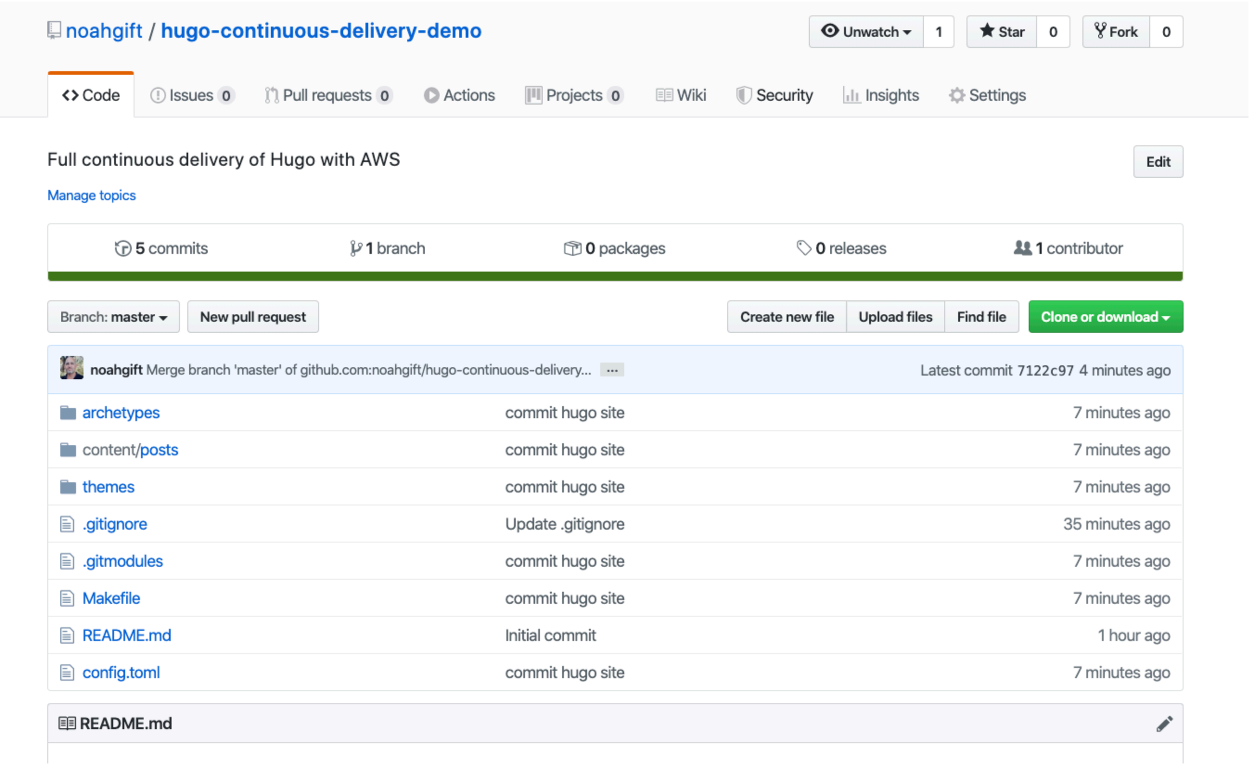
You can see what this looks like below:



```
ec2-user:~/environment/quickstart (master) $ git branch --set-upstream-to=origin/master
Branch master set up to track remote branch master from origin.
ec2-user:~/environment/quickstart (master) $ git pull --allow-unrelated-histories
Merge made by the 'recursive' strategy.
 .gitignore | 18 ++++++++++++++++++
 README.md  |  2 ++
 2 files changed, 20 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
ec2-user:~/environment/quickstart (master) $ git push
Counting objects: 13, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (13/13), 1.42 KiB | 1.42 MiB/s, done.
Total 13 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To github.com:noahgift/hugo-continuous-delivery-demo.git
   85e2e0a..7122c97  master -> master
ec2-user:~/environment/quickstart (master) $ █
```

git push hugo

The Github repo looks like this now:



Full continuous delivery of Hugo with AWS

noahgift Merge branch 'master' of github.com:noahgift/hugo-continuous-delivery... … Latest commit 7122c97 4 minutes ago

archetypes	commit hugo site	7 minutes ago
content/posts	commit hugo site	7 minutes ago
themes	commit hugo site	7 minutes ago
.gitignore	Update .gitignore	35 minutes ago
.gitmodules	commit hugo site	7 minutes ago
Makefile	commit hugo site	7 minutes ago
README.md	Initial commit	1 hour ago
config.toml	commit hugo site	7 minutes ago

README.md

github repo

NOTE: Using `git` *can be very challenging in edge cases. If this workflow doesn't work, you can also start over from scratch and clone your GitHub repo and manually add* `hugo` *into it*

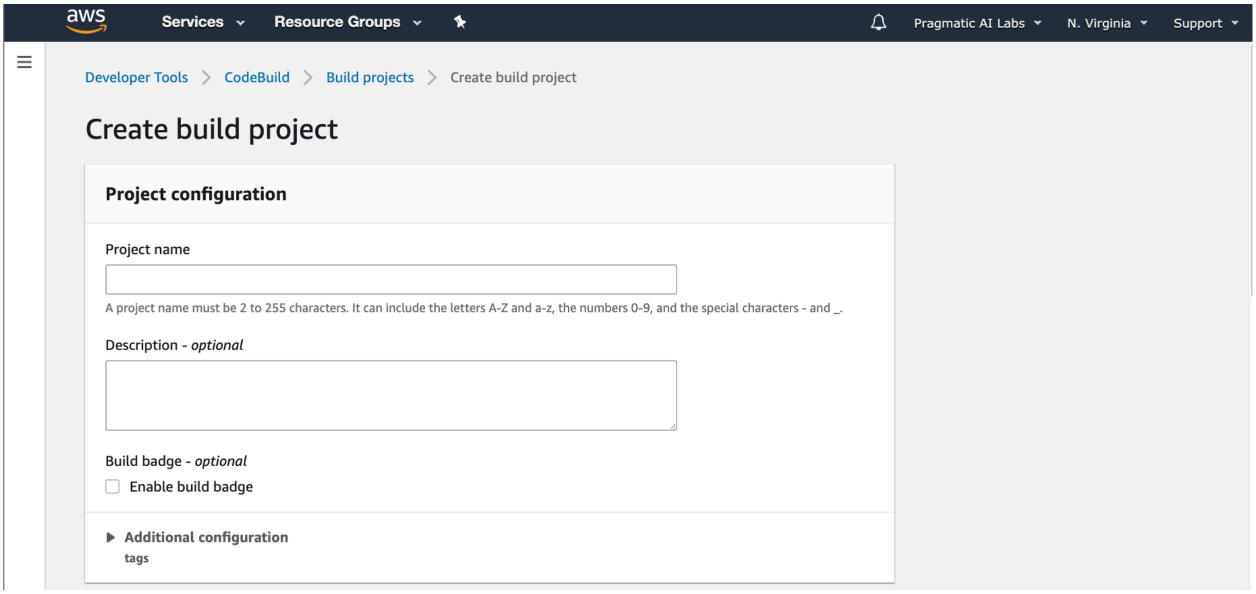
(Optional step: If you want to verify your `hugo` *site, check out this project on your laptop or another*

AWS Cloud9 instance and run `hugo`.)

- Step9: Continuous Delivery with AWS CodeBuild

Now it is time for the final part. Let's continuously setup delivery using AWS CodeBuild. This step will allow changes that get pushed to Github to deploy automatically.

A. Go to AWS CodeBuild[139] and create a new project. It is should look like this:


aws
Services
Resource Groups
Pragmatic AI Labs
N. Virginia
Support
Developer Tools > CodeBuild > Build projects > Create build project

Create build project

Project configuration

Project name

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.

Description - *optional*

Build badge - *optional*

□ Enable build badge

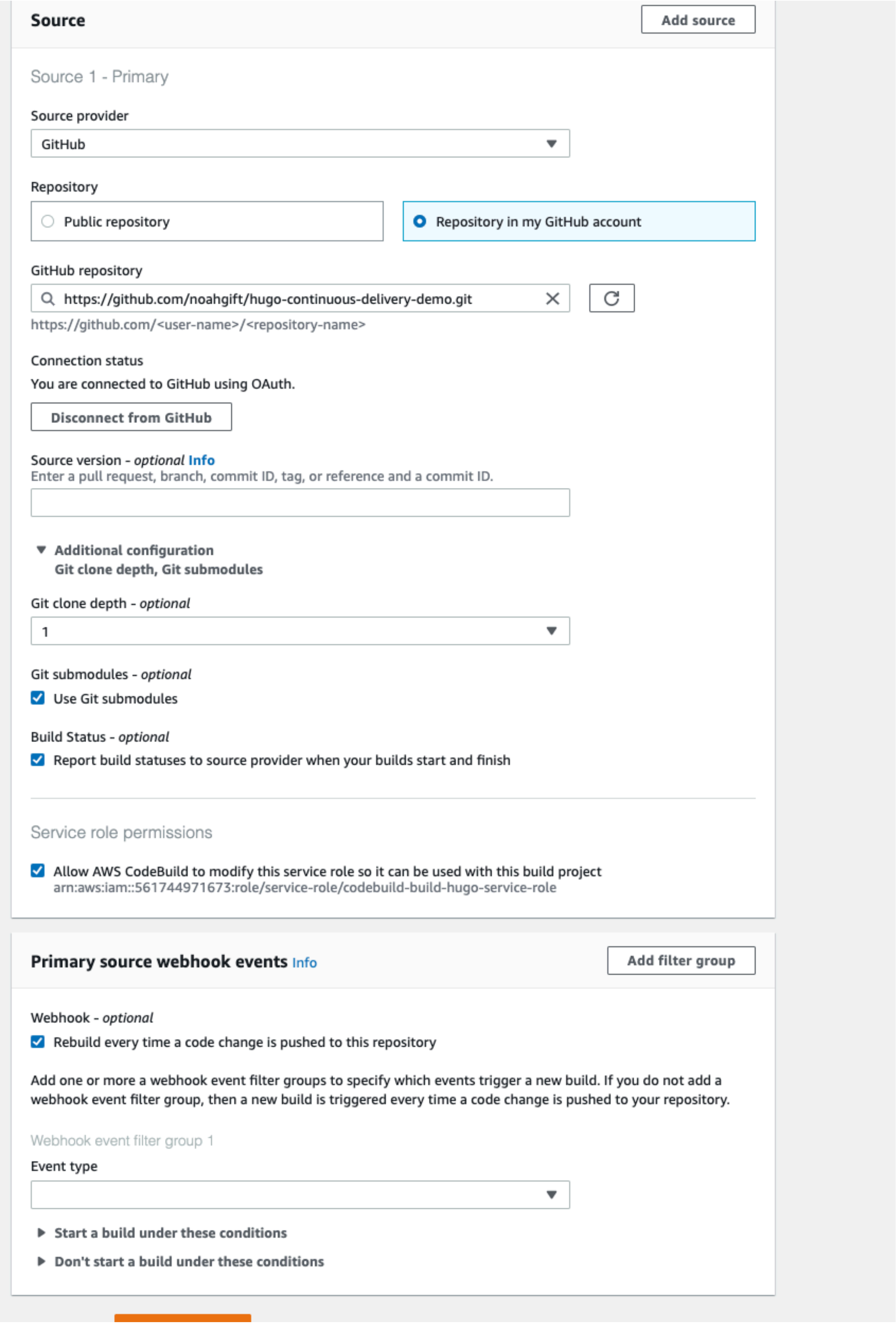
▶ Additional configuration
tags

code build

Note create a build in the same region you made your bucket: i.e., N. Virginia!

B. The source code section should look similar to this screenshot. *Note the `webhook`. This step will do continuous delivery on changes*

[139] https://aws.amazon.com/codebuild/



Source

Source 1 - Primary

Source provider

GitHub ▼

Repository

○ Public repository

● Repository in my GitHub account

GitHub repository

https://github.com/noahgift/hugo-continuous-delivery-demo.git

https://github.com/<user-name>/<repository-name>

Connection status

You are connected to GitHub using OAuth.

Source version - *optional* Info

Enter a pull request, branch, commit ID, tag, or reference and a commit ID.

▼ **Additional configuration**
Git clone depth, Git submodules

Git clone depth - *optional*

1 ▼

Git submodules - *optional*

☑ Use Git submodules

Build Status - *optional*

☑ Report build statuses to source provider when your builds start and finish

Service role permissions

☑ Allow AWS CodeBuild to modify this service role so it can be used with this build project
arn:aws:iam::561744971673:role/service-role/codebuild-build-hugo-service-role

Primary source webhook events Info

Webhook - *optional*

☑ Rebuild every time a code change is pushed to this repository

Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

Webhook event filter group 1

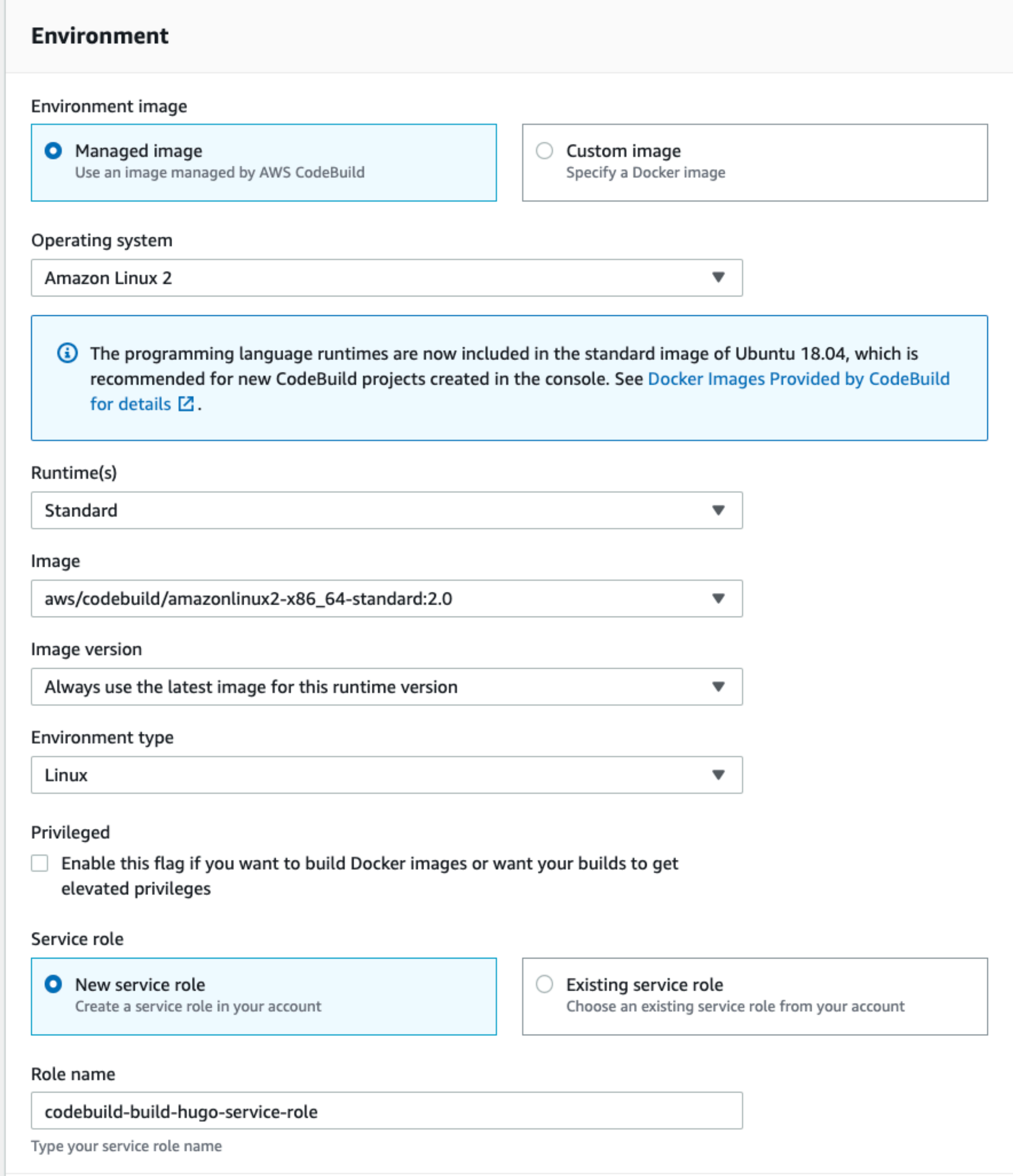
Event type

▼

▶ **Start a build under these conditions**

▶ **Don't start a build under these conditions**

C. The AWS Code Build environment should look similar to this. Click the "create build" button:



Environment

Environment image

● Managed image
Use an image managed by AWS CodeBuild

○ Custom image
Specify a Docker image

Operating system

Amazon Linux 2 ▼

The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See Docker Images Provided by CodeBuild for details.

Runtime(s)

Standard ▼

Image

aws/codebuild/amazonlinux2-x86_64-standard:2.0 ▼

Image version

Always use the latest image for this runtime version ▼

Environment type

Linux ▼

Privileged

□ Enable this flag if you want to build Docker images or want your builds to get elevated privileges

Service role

● New service role
Create a service role in your account

○ Existing service role
Choose an existing service role from your account

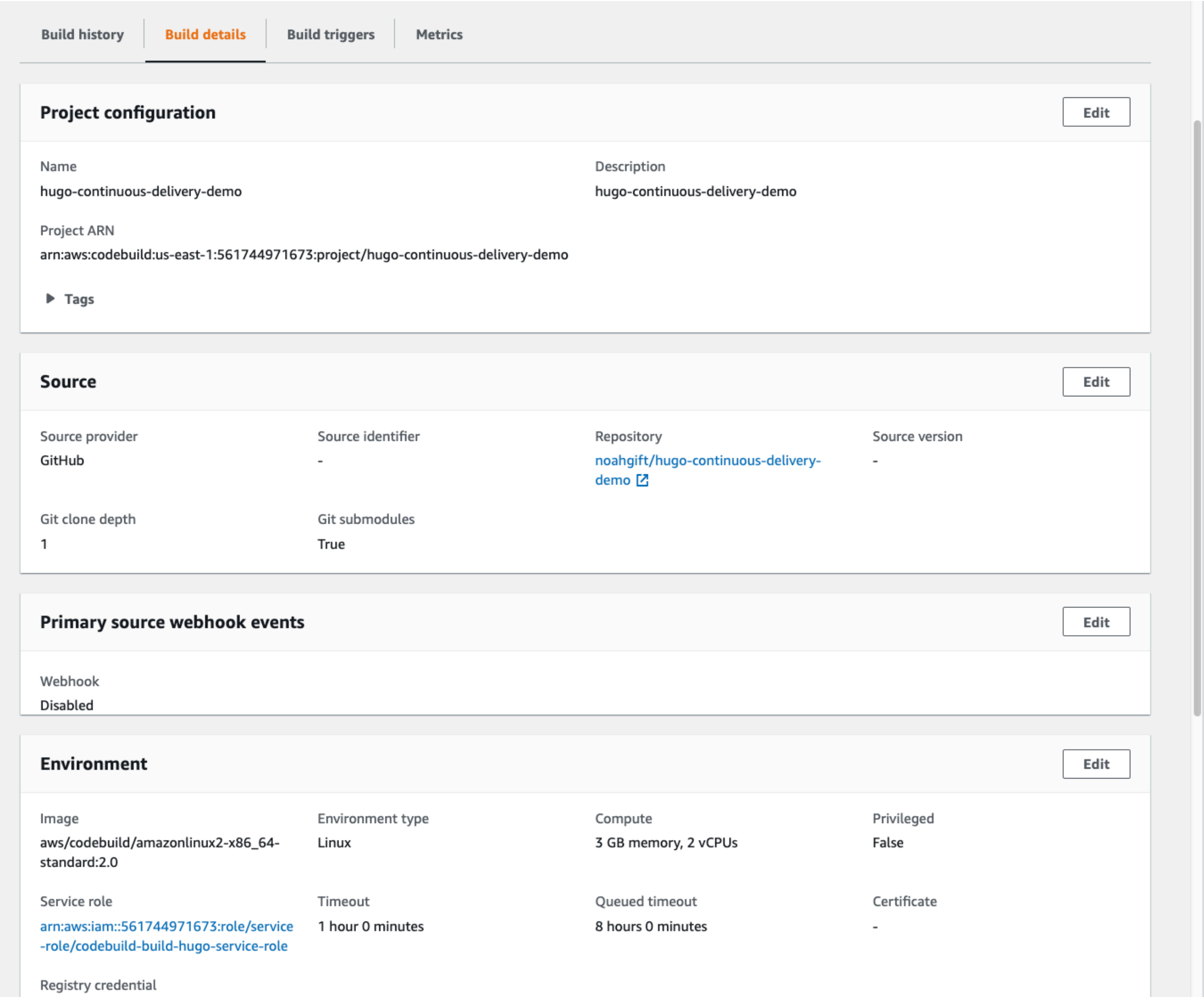
Role name

codebuild-build-hugo-service-role

Type your service role name

codebuild environment

D. After you create the build navigate to the "Build details" section and select the service role. This where the privileges to deploy to S3 will be setup:



Project configuration

Name
hugo-continuous-delivery-demo

Description
hugo-continuous-delivery-demo

Project ARN
arn:aws:codebuild:us-east-1:561744971673:project/hugo-continuous-delivery-demo

▶ Tags

Source

Source provider	Source identifier	Repository	Source version
GitHub	-	noahgift/hugo-continuous-delivery-demo	-

Git clone depth	Git submodules
1	True

Primary source webhook events

Webhook
Disabled

Environment

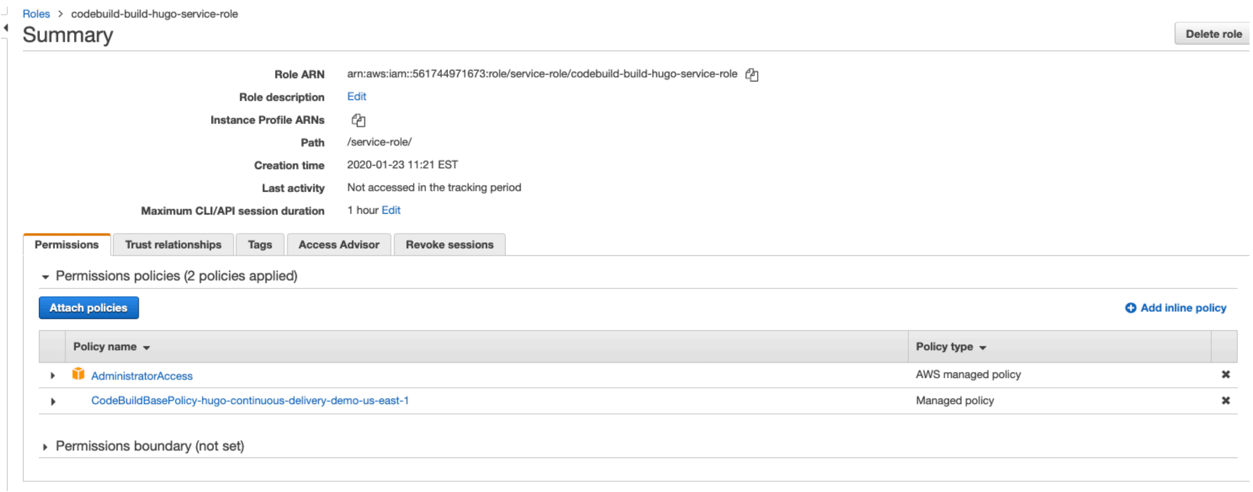
Image	Environment type	Compute	Privileged
aws/codebuild/amazonlinux2-x86_64-standard:2.0	Linux	3 GB memory, 2 vCPUs	False

Service role	Timeout	Queued timeout	Certificate
arn:aws:iam::561744971673:role/service-role/codebuild-build-hugo-service-role	1 hour 0 minutes	8 hours 0 minutes	-

Registry credential

codebuild service role

You will add an “admin” policy that looks like this:



Role ARN arn:aws:iam::561744971673:role/service-role/codebuild-build-hugo-service-role

Role description Edit

Instance Profile ARNs

Path /service-role/

Creation time 2020-01-23 11:21 EST

Last activity Not accessed in the tracking period

Maximum CLI/API session duration 1 hour Edit

Permissions policies (2 policies applied)

Policy name	Policy type	
AdministratorAccess	AWS managed policy	✖
CodeBuildBasePolicy-hugo-continuous-delivery-demo-us-east-1	Managed policy	✖

Permissions boundary (not set)

admin policy

Now, in AWS Cloud9, go back and create the final step.

The following is a `buildspec.yml` file you can paste it. You create the file with AWS Cloud9 by typing: `touch buildspec.yml` then editing.

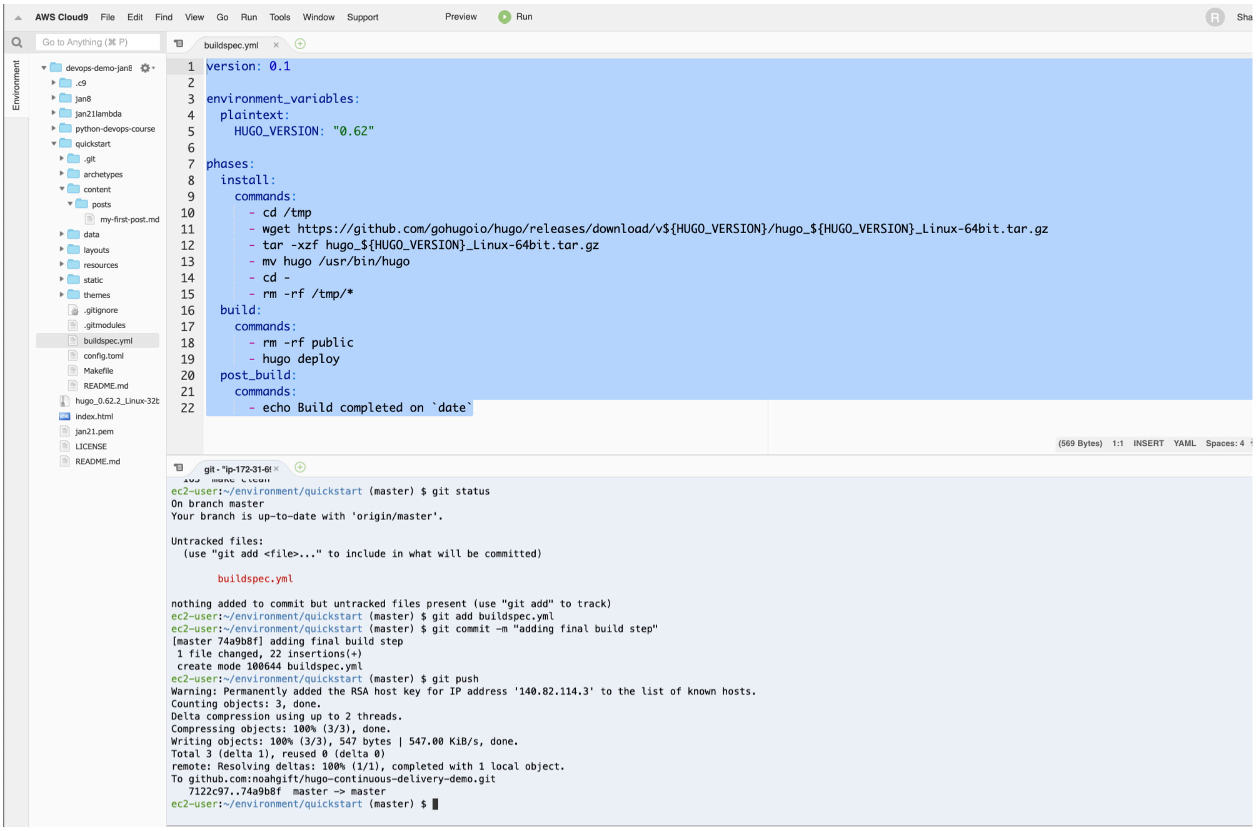
NOTE: Something like the following `aws s3 sync public/ s3://hugo-duke-jan23/ --region us-east-1 --delete` is an effective and explicit way to deploy if `hugo deploy` is not working correctly

```
version: 0.2

environment_variables:
  plaintext:
    HUGO_VERSION: "0.79.1"

phases:
  install:
    runtime-versions:
      docker: 18
    commands:
      - cd /tmp
      - wget https://github.com/gohugoio/hugo/releases/download/v${HUGO_VERSION}/hug\
o_${HUGO_VERSION}_Linux-64bit.tar.gz
      - tar -xzf hugo_${HUGO_VERSION}_Linux-64bit.tar.gz
      - mv hugo /usr/bin/hugo
      - cd -
      - rm -rf /tmp/*
  build:
    commands:
      - rm -rf public
      - hugo
      - aws s3 sync public/ s3://hugo-duke-jan23/ --region us-east-1 --delete
  post_build:
    commands:
      - echo Build completed on `date`
```

Now check this file into git and push:

```
git add buildspec.yml
git commit -m "adding final build step."
git push
```

It should look like this:

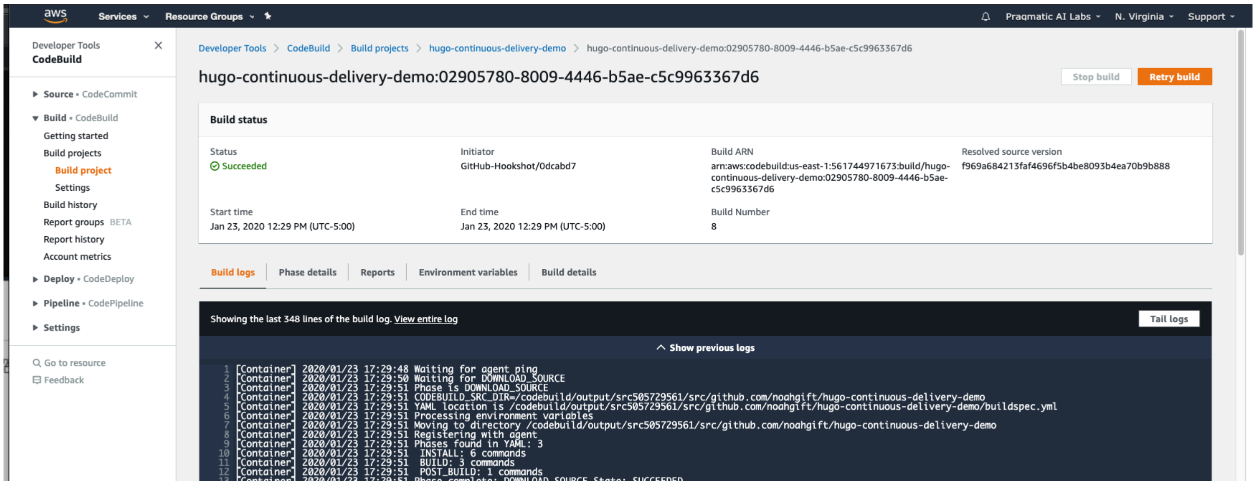


```
version: 0.1

environment_variables:
  plaintext:
    HUGO_VERSION: "0.62"

phases:
  install:
    commands:
      - cd /tmp
      - wget https://github.com/gohugoio/hugo/releases/download/v${HUGO_VERSION}/hugo_${HUGO_VERSION}_Linux-64bit.tar.gz
      - tar -xzf hugo_${HUGO_VERSION}_Linux-64bit.tar.gz
      - mv hugo /usr/bin/hugo
      - cd -
      - rm -rf /tmp/*
  build:
    commands:
      - rm -rf public
      - hugo deploy
  post_build:
    commands:
      - echo Build completed on `date`
```

```
ec2-user:~/environment/quickstart (master) $ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        buildspec.yml

nothing added to commit but untracked files present (use "git add" to track)
ec2-user:~/environment/quickstart (master) $ git add buildspec.yml
ec2-user:~/environment/quickstart (master) $ git commit -m "adding final build step"
[master 74a9b8f] adding final build step
 1 file changed, 22 insertions(+)
 create mode 100644 buildspec.yml
ec2-user:~/environment/quickstart (master) $ git push
Warning: Permanently added the RSA host key for IP address '140.82.114.3' to the list of known hosts.
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 547 bytes | 547.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:noahgift/hugo-continuous-delivery-demo.git
   7122c97..74a9b8f  master -> master
ec2-user:~/environment/quickstart (master) $
```

buildspec push

Now every time you make changes to the content directory, it will "auto-deploy" as shown.



hugo-continuous-delivery-demo:02905780-8009-4446-b5ae-c5c9963367d6

Build status

Status: Succeeded

Initiator: GitHub-Hookshot/0dcabd7

Build ARN: arn:aws:codebuild:us-east-1:561744971673:build/hugo-continuous-delivery-demo:02905780-8009-4446-b5ae-c5c9963367d6

Resolved source version: f969a684213faf4696f5b4be8093b4ea70b9b888

Start time: Jan 23, 2020 12:29 PM (UTC-5:00)

End time: Jan 23, 2020 12:29 PM (UTC-5:00)

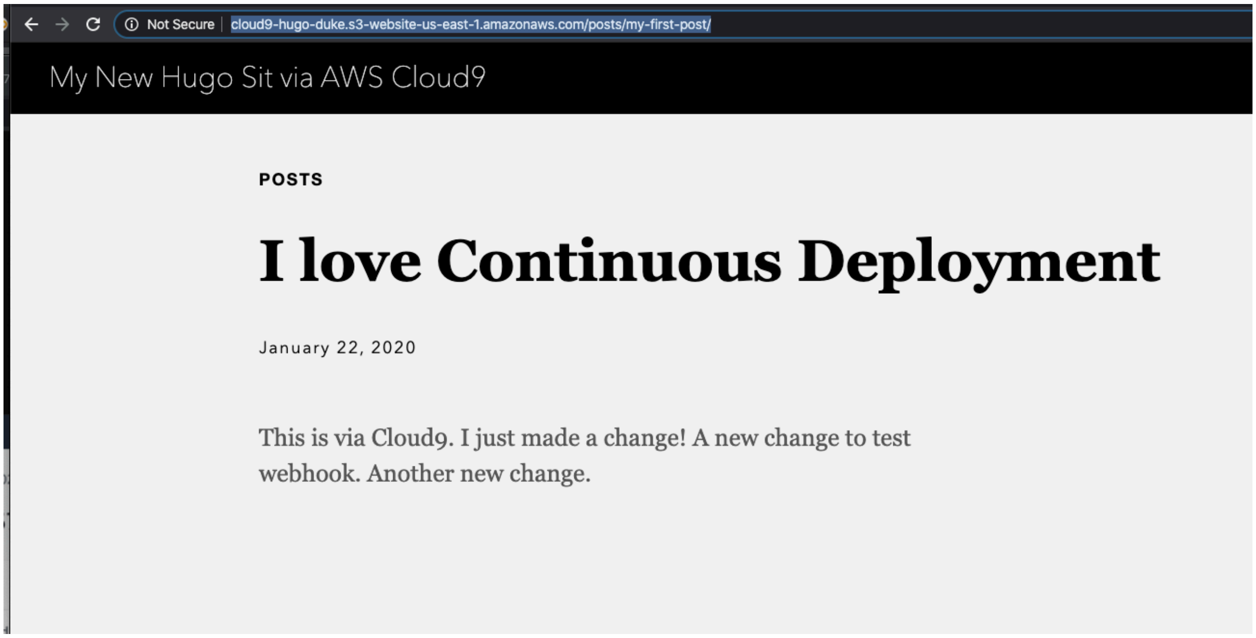
Build Number: 8

Showing the last 348 lines of the build log. View entire log

```
[Container] 2020/01/23 17:29:48 Waiting for agent ping
[Container] 2020/01/23 17:29:50 Waiting for DOWNLOAD_SOURCE
[Container] 2020/01/23 17:29:51 Phase is DOWNLOAD_SOURCE
[Container] 2020/01/23 17:29:51 CODEBUILD_SRC_DIR=/codebuild/output/src505729561/src/github.com/noahgift/hugo-continuous-delivery-demo
[Container] 2020/01/23 17:29:51 YAML location is /codebuild/output/src505729561/src/github.com/noahgift/hugo-continuous-delivery-demo/buildspec.yml
[Container] 2020/01/23 17:29:51 Processing environment variables
[Container] 2020/01/23 17:29:51 Moving to directory /codebuild/output/src505729561/src/github.com/noahgift/hugo-continuous-delivery-demo
[Container] 2020/01/23 17:29:51 Registering with agent
[Container] 2020/01/23 17:29:51 Phases found in YAML: 3
[Container] 2020/01/23 17:29:51  INSTALL: 6 commands
[Container] 2020/01/23 17:29:51  BUILD: 3 commands
[Container] 2020/01/23 17:29:51  POST_BUILD: 1 commands
[Container] 2020/01/23 17:29:51 Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED
```

auto-build

As you create new posts, etc., it will deploy.


Not Secure
cloud9-hugo-duke.s3-website-us-east-1.amazonaws.com/posts/my-first-post/
My New Hugo Sit via AWS Cloud9
POSTS

I love Continuous Deployment

January 22, 2020

This is via Cloud9. I just made a change! A new change to test webhook. Another new change.

auto deploy

Hugo AWS Continuous Delivery Conclusion

Continuous Delivery is a powerful technique to master. In this situation, it could immediately be useful to build a portfolio website for a Data Scientist or a new website like the New York Times or Wall Street Journal.

- Example Hugo AWS Repository[140]

[140] https://github.com/noahgift/hugo-duke-jan23

Post Setup (Optional Advanced Configurations & Notes)

The following are additional notes on how to do more advanced setup actions for Hugo.

Setting up SSL for CloudFront

Go to AWS Certificate Manager and click **Request a certificate** button.
First, we need to add domain names, in our case (example.com). When you enter the domain name as `*.example.com`, click **Add another name to this certificate** button and add the bare domain `example.com` too. Next step, select the **DNS validation** option and click the **Confirm and request** button in Review.
To use DNS validation, you add a CNAME record to the DNS configuration for your domain. Add CNAME record created on ACM to the DNS configuration for your domain on **Route 53**.

CloudFront configurations

Create a web distribution in the CloudFront section. In the **Origin Domain Name** field, select Endpoint of your bucket. Select "Redirect HTTP to HTTPS" from the **Viewer Protocol Policy**. Add your domain names in the **Alternate Domain Name** filed and select the SSL certificate you have created in the ACM. In the **Default Root Object** type `index.html`. Once done, please proceed and complete the distribution.

Integrating Route53 with CloudFront distribution:

Copy the domain name from the CloudFront distribution and edit A record in your Route53. Select **Alias**, in **Alias Target**, enter your CloudFront domain URL which is ******.cloudfront.net. Click **Save Record Set**. Now that you have created A record. The domain name example.com will route to your **CloudFront distribution**.
We need to create a CNAME record to point other sub-domains like `www.example.com` to map to the created **A record**
Click **Create Record Set**, enter * in name textbox. Select **CNAME** from Type. In value, type the A record; in our case, it will be example.com. Click **Save Record Set**. Now even www.example.com will forward to example.com, which in turn will forward to CloudFront distribution.

Building Hugo Sites Automatically Using AWS CodeBuild

The first thing that we need is a set of instructions for building the Hugo site. Since the build server starts cleaning every time up push event, this step includes downloading Hugo and all the dependencies required. One of the options that CodeBuild has for specifying the build instruction is the `buildspec.yaml` file.

Navigate to the CodeBuild console and create a new project using settings similar to this or that meet your project's demands:

* **Project name:** `somename-hugo-build-deploy`
* **Source provider:** `GitHub`
* **Repository:** `Use a repository in my account.`
* **Choose a repository:** `Choose your GitHub repository`
* Click on **Webhook** checkbox for rebuilding project every time a code change pushes to this repository
* **Environment image:** `Use an image managed by AWS CodeBuild`
* **Operating System:** `Ubuntu`
* **Runtime:** `Base`
* **Runtime version:** `Choose a runtime environment version`
* **Buildspec name:** `buildspec.yml`
* **Artifact type:** `No artifact`
* **Cache:** `No cache`
* **Service role:** `Create a service role in your account`

Creating IAM Role

For building a project, deploy to S3 and enable CloudFront Invalidation, we need to create an individual IAM role. Add IAM role and attach **CloudFrontFullAccess** and **AmazonS3FullAccess** policies. After that, click **Add permissions** button again, select “Attach existing policies directly,” and click the **Create policy** button. Select “JSON” and paste the following user policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "cloudfront:CreateInvalidation",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::s3-<bucket-name>",
                "arn:aws:s3:::s3-<bucket-name>/*"
            ]
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::s3-<bucket-name>",
                "arn:aws:s3:::s3-<bucket-name>/*"
            ]
        }
    ]
}
```

Case Studies-Hugo-Continuous-Deploy

What are some logical next steps you could improve?

- Setup the build server to have a more granular security policy.
- Create an SSL certificate via AWS (for free).
- Publish your content to the AWS Cloudfront CDN.
- Enhance the `Makefile` to use a `deploy` command you also use in the build server instead of the verbose `aws sync` command.
- Try to “deploy” from many spots: Laptop, editing Github pages directly, a different cloud.
- Can you use the built-in `hugo` deployment commands[141] to simplify this setup?

Take some or all of these case study items and complete them.

Summary

This chapter covers foundational topics in Cloud Computing, including Economics of Cloud Computing, What is Cloud Computing, Cloud Computing Service models, and several hands-on approaches to building Cloud Computing applications and services.

If you enjoyed this book, consider buying a copy

- Buy a copy of the Cloud Computing for Data on Lean Pub[142]
- Buy the bundle Master Python on Lean Pub[143]

[141] https://gohugo.io/hosting-and-deployment/hugo-deploy/
[142] http://leanpub.com/cloud4data/c/WbJPdnkotEr6
[143] https://leanpub.com/b/masterpython