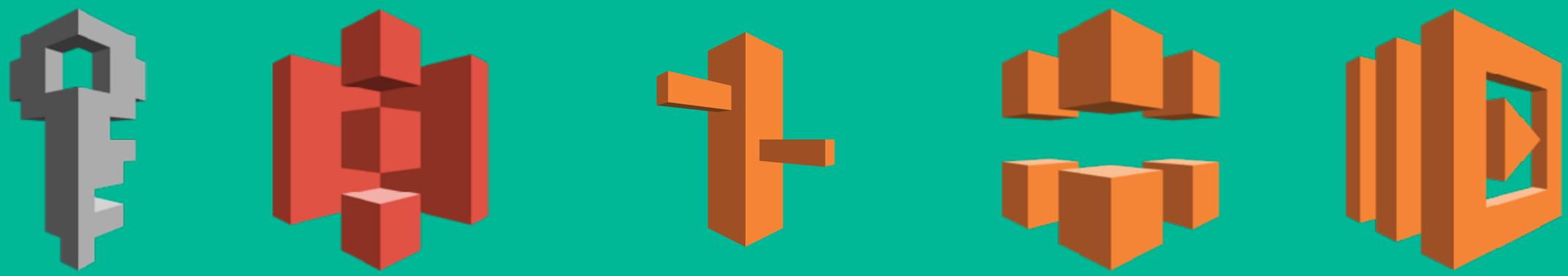
Real World Client-Side Deployment on AWS

Steve Kinney



This workshop covers some of the approaches we've taken (or *are taking*) at SendGrid for our large, client-side application.

Chicago - Google Maps

Steve

https://www.google.com/maps/place/Chicago,+IL/@31.9048243,-51.7672423,2.77z/data=!4m5!3m4!1s0x880e2c3cd0f4cbed:0xafe0a6ad09...

The Original State

- Our client-side code was hosted out of our Rails app.
- The Rails application was located in Chicago.
- Users from around the world had to reach Chicago in order to get the application.
- We had to deploy Rails every time we wanted to make a change to our React application.

	Before	After	Difference
Denver	5.682s	1.419s	133.26%
Frankfurt	11.026s	1.869s	489.94%
Mumbai	9.641s	1.373s	602.18%
Sydney	12.31s	0.87s	1314.94%

Noted

Steve

Secure | https://mysuperfunwebsite.com/notes/1

Noted

Things to Do

To Do - Buy a winning lottery ticket - Buy some vegan spam - Clean out from under the bed

Very Important Link

[Do not click here] (https://bit.ly/very-secret).

Hello World

I shredded your linens for you run in circles run up and down stairs. Tuxedo cats always looking dap...

A Picture of Prince

![Prince] (/prince-1.jpg)

Close

Things to Do

To Do

- Buy a winning lottery ticket
- Buy some vegan spam
- Clean out from under the bed

Noted

- A hollowed out version of our production application.
- Based on the webpack configuration we use.
- Includes modern techniques like lazy-loading and client-side routing.
- Notable differences: smaller, not TypeScript.

We’re going to build this infrastructure *today*.

What are we going to cover?

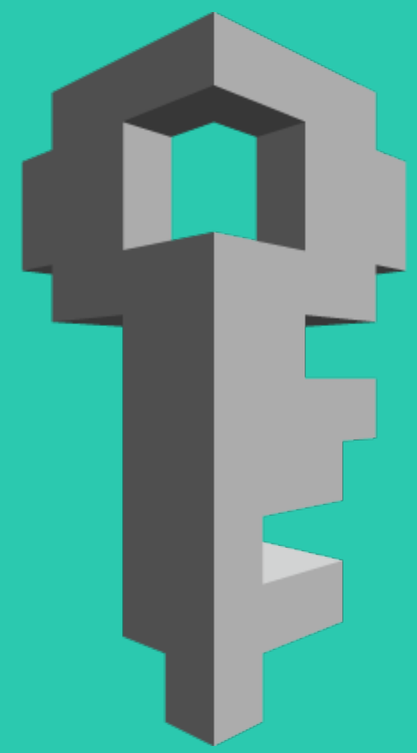
- We'll be looking at getting a single-page application:
    - Hosted on AWS.
    - Distributed globally.
    - Secured with SSL.
    - Automatically deployed with CI/CD.
    - Dynamically responding to requests.

What are we *not* going to cover.

- **Servers**. This course is focused on the high-performance distribution of your client-side, JavaScript application.
- **Serverless**. Scott Moss has an amazing workshop that came out recently and you should totally go watch it—*after* this one, of course.

Together

- Let’s take a quick tour of the free tier.
- We’ll also set up billing notification warning us if we blow past the free tier.



Identity **A**ccess and **M**anagement

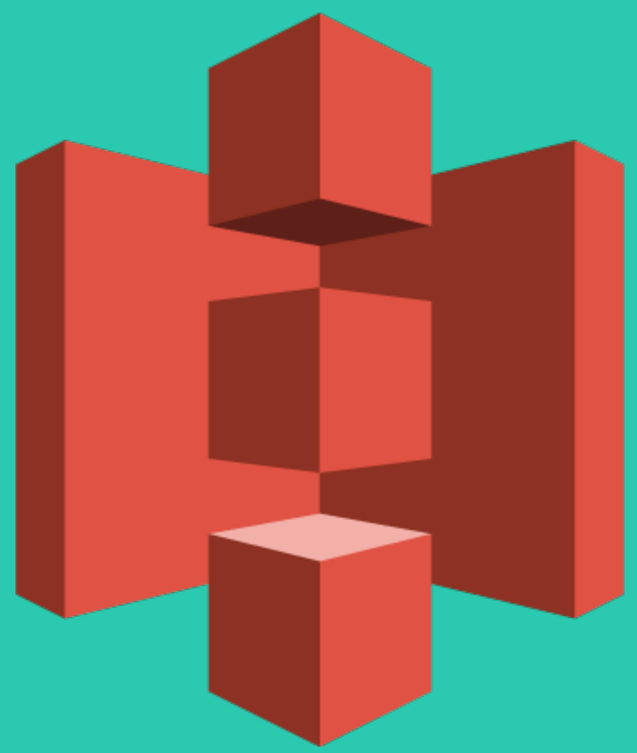
TL;DR: It’s how you manage sub accounts in AWS.

Generally speaking, it's a *bad* idea to use your root account for anything.

Principle of Least Access

Together

- We’re going to turn on MFA for our root account.
- We’re going to make a new user (e.g. *not* our root account).
- We’re going to turn on MFA for that one too.



Simple **S**torage **S**ervice

A High Level

- In our S3 account, you have your buckets.
- You can put objects (read: *files*) in your buckets.
- You can read from your buckets as well.
- You can host web pages out of your buckets.

A Slightly Lower, But Still High Level

- Infinitely scalable.
- Files can be as small as zero bytes or as large as 5 terabytes.
- 99.9% availability (built for 99.99%).
- 99.999999999% durability.

Features

- Lifecycle Management
- Versioning
- Encryption
- Security

Storage Tiers

- **Standard**—this is what we'll be using today.
- **Infrequently accessed**.
- **Reduced redundancy**.
- **Glacier**—this technically isn't S3, but whatever.

S3 is effectively a *key/value* store.

Data Consistency Model

- Putting new objects in S3 is immediate. You'll get back a 200 response.
- Updating and removing objects is eventually consistent. Users might get an old version. (This has literally never happened to me.)

So, what's this going to cost me?

- Uploading to S3 is free.
- You get charged for storage.
- You get charged for requests. (We'll learn how to mitigate this later.)

An Aside on Registering a Domain Name with AWS

Exercise

- The is **totally optional**. If you don’t want to purchase a domain name, you’ll skip a few steps as we go along, but you’ll be able to come along.

- We’re going to register a domain name now so that we know we have one that’s unique.

- Also: registration takes some time, so—let’s get it started now.

console.aws.amazon.com/route53/home?#

aws Services Resource Groups deploy @ fem-dryrun Global Support

Amazon Route 53

You can use Amazon Route 53 to register new domains, transfer existing domains, route traffic for your domains to your AWS and external resources, and monitor the health of your resources.

DNS management

If you already have a domain name, such as example.com, Route 53 can tell the Domain Name System (DNS) where on the Internet to find web servers, mail servers, and other resources for your domain.
Learn More

Get started now

Traffic management

Route 53 traffic flow provides a visual tool that you can use to create and update sophisticated routing policies to route end users to multiple endpoints for your application.
Learn More

Get started now

Availability monitoring

Route 53 can monitor the health and performance of your application as well as your web servers and other resources. Route 53 can also redirect traffic to healthy resources.
Learn More

Get started now

Domain registration

If you need a domain name, you can find available name and register it by using Route 53. You can also make Route 53 th registrar for existing domains that you registered with other registrars.
Learn More

Get started now

console.aws.amazon.com/route53/home?#DomainListing:

aws | Services | Resource Groups | deploy @ fem-dryrun | Global | Support

- Dashboard
- Hosted zones
- Health checks

Traffic flow
- Traffic policies
- Policy records

Domains
- **Registered domains**
- Pending requests

Register Domain | **Transfer Domain** | **Domain Billing Report**

Search ains by prefix

No domains to display

Domain Nam	Privacy Protection	Expiration Date	Auto Renew	Transfer Lock

No domains to display

console.aws.amazon.com/route53/home?#DomainRegistration:

aws | Services | Resource Groups | deploy @ fem-dryrun | Global | Support

1: Domain Search

2: Contact Details

3: Verify & Purchase

Choose a domain name

superfunwebsite | .com - $12.00 | Check

Availability for 'superfunwebsite.com'

Domain Name		Status	Price /1 Year	Action
superfunwebsite.com	✗	Unavailable		

Related domain suggestions

Domain Name		Status	Price /1 Year	Action
mysuperfunwebsite.com	✓	Available	$12.00	Add to cart
sufuwe.com	✓	Available	$12.00	Add to cart
sufuwe.net	✓	Available	$11.00	Add to cart
superfunwebsite.info	✓	Available	$12.00	Add to cart
superfunwebsite.me	✓	Available	$17.00	Add to cart
superfunwebsite.net	✓	Available	$11.00	Add to cart
superfunwebsite.org	✓	Available	$12.00	Add to cart
superfunwebsite.tv	✓	Available	$32.00	Add to cart

Shopping cart

Feedback | English (US)

Privacy Policy | Terms of Use

console.aws.amazon.com/route53/home?#DomainRegistration:

aws | Services | Resource Groups | deploy @ fem-dryrun | Global | Support

Related domain suggestions

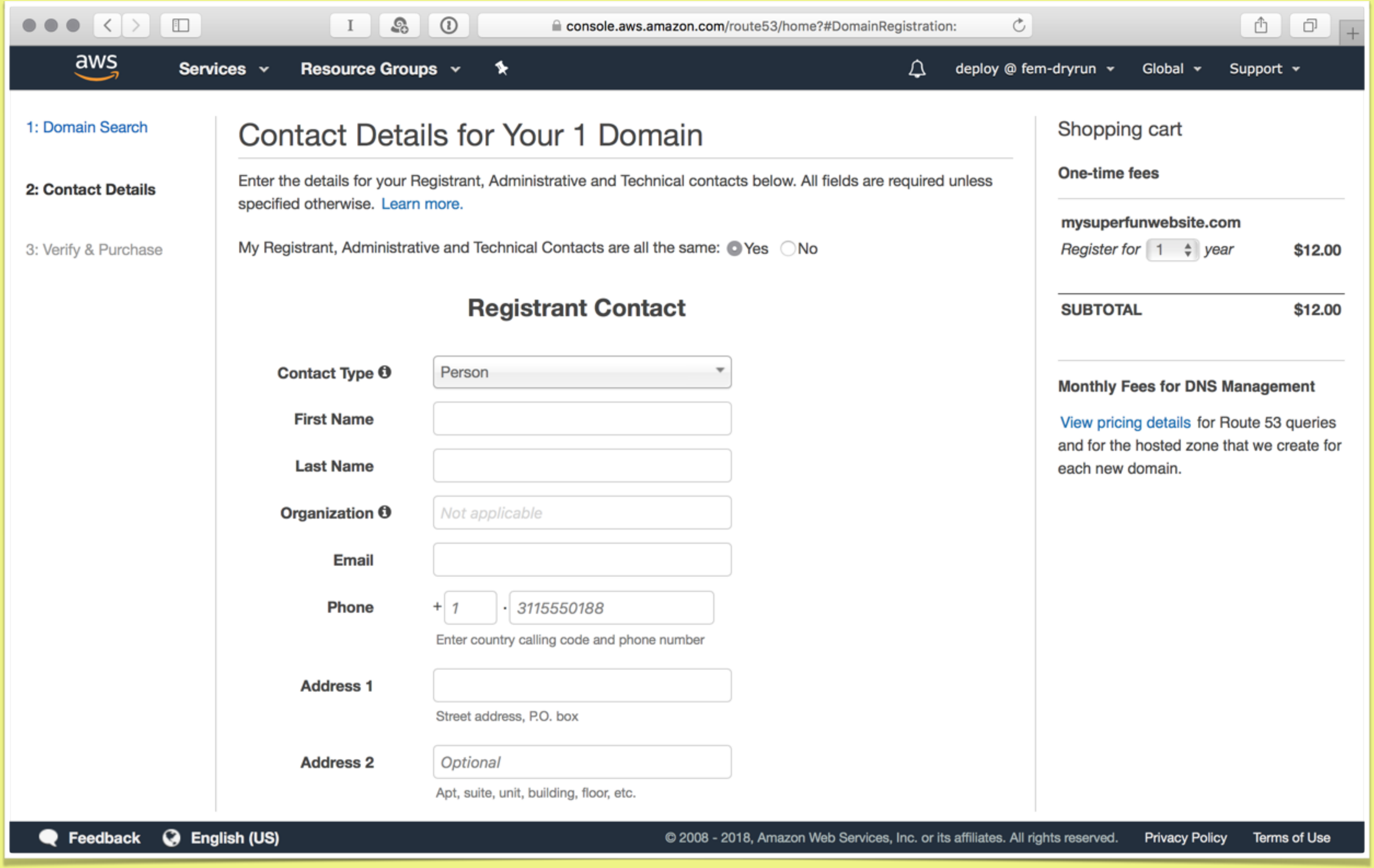
Domain Name		Status	Price /1 Year	Action
mysuperfunwebsite.com	✓	Available - In Cart	$12.00	Add to cart
sufuwe.com	✓	Available	$12.00	Add to cart
sufuwe.net	✓	Available	$11.00	Add to cart
superfunwebsite.info	✓	Available	$12.00	Add to cart
superfunwebsite.me	✓	Available	$17.00	Add to cart
superfunwebsite.net	✓	Available	$11.00	Add to cart
superfunwebsite.org	✓	Available	$12.00	Add to cart
superfunwebsite.tv	✓	Available	$32.00	Add to cart
superplaywebsite.com	✓	Available	$12.00	Add to cart
superplaywebsite.net	✓	Available	$11.00	Add to cart
thesuperfunwebsite.com	✓	Available	$12.00	Add to cart

SUBTOTAL **$12.00**

Monthly Fees for DNS Management

View pricing details for Route 53 queries and for the hosted zone that we create for each new domain.

Cancel | Continue


console.aws.amazon.com/route53/home?#DomainRegistration:
aws
Services
Resource Groups
deploy @ fem-dryrun
Global
Support
1: Domain Search
2: Contact Details
3: Verify & Purchase
Contact Details for Your 1 Domain
Enter the details for your Registrant, Administrative and Technical contacts below. All fields are required unless specified otherwise. Learn more.
My Registrant, Administrative and Technical Contacts are all the same: Yes No
Registrant Contact
Contact Type
Person
First Name
Last Name
Organization
Not applicable
Email
Phone
+ 1 · 3115550188
Enter country calling code and phone number
Address 1
Street address, P.O. box
Address 2
Optional
Apt, suite, unit, building, floor, etc.
Shopping cart
One-time fees
mysuperfunwebsite.com
Register for 1 year
$12.00
SUBTOTAL
$12.00
Monthly Fees for DNS Management
View pricing details for Route 53 queries and for the hosted zone that we create for each new domain.
Feedback
English (US)
© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

console.aws.amazon.com/route53/home?#DomainRegistration:

aws Services Resource Groups deploy @ fem-dryrun Global Support

Organization	Not applicable
Email	[redacted]
Phone	+ 1 · [redacted]
	Enter country calling code and phone number
Address 1	[redacted]
	Street address, P.O. box
Address 2	Optional
	Apt, suite, unit, building, floor, etc.
Country	United States
State	Colorado
City	Denver
Postal/Zip Code	[redacted]
Privacy Protection	When the contact type is Person:

- Privacy protection hides **some** contact details for .com domains.

Enable Disable

Cancel Back Continue

1: Domain Search

2: Contact Details

3: Verify & Purchase

Check your contact details

Confirm that the following contact information is correct. When you complete your purchase, we'll use this information for all of the domains in your shopping cart.

Registrant Contact

Administrative Contact

Technical Contact

Managing DNS for Your New Domain

To make it easier for you to use Route 53 as the DNS service for your new domain, we'll automatically create a hosted zone. That's where you store information about how to route traffic for your domain, for example, to an Amazon EC2 instance. If you won't use your domain right now, you can delete the hosted zone. If you will use your domain, Route 53 charges for the hosted zone and for the DNS queries that we receive for your domain. For more information, see Amazon Route 53 Pricing.

Terms and Conditions

Amazon Route 53 enables you to register and transfer domain names using your AWS account. However, AWS is not a domain name registrar, so we use registrar associates to perform registration and transfer services. When you purchase domain names through AWS, you are registering your domain with one of our registrar associates. The registrar for your domain will periodically contact the registrant contact that you specified to verify the contact details and renew registration.

Shopping cart

One-time fees

mysuperfunwebsite.com

Register for 1 *year* **$12.00**

SUBTOTAL **$12.00**

Monthly Fees for DNS Management

View pricing details for Route 53 queries and for the hosted zone that we create for each new domain.

console.aws.amazon.com/route53/home?#DomainRequests:

aws Services Resource Groups deploy @ fem-dryrun Global Support

- Dashboard
- Hosted zones
- Health checks

Traffic flow

- Traffic policies
- Policy records

Domains

- Registered domains
- **Pending requests**

Status of new domain registrations and domain transfers

Domains that we're registering or transferring for you are listed below. When the registration or transfer is complete, the domain appears on the Registered domains page.

Displaying 1 to 1 out of 1 domains

Domain Name	Status	Timestamp
mysuperfunwebsite.com	Domain registration in progress	August 11, 2018 14:07 UTC-6

Exercise

- This is **totally optional**. If you don’t want to purchase a domain name, you’ll skip a few steps as we go along, but you’ll be able to come along.

- Register a domain name now so that we know we have one that’s unique.

- Also: registration takes some time, so—let’s get it started now.

A Quick Word on Policies

You’re about to see me use a bucket policy, let’s talk about it first.

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

Policy Terms

- **Principal** — *who* can do the thing?
- **Action** — *what* can they do?
- **Resource** — to *which* things?

Let’s upload our application.

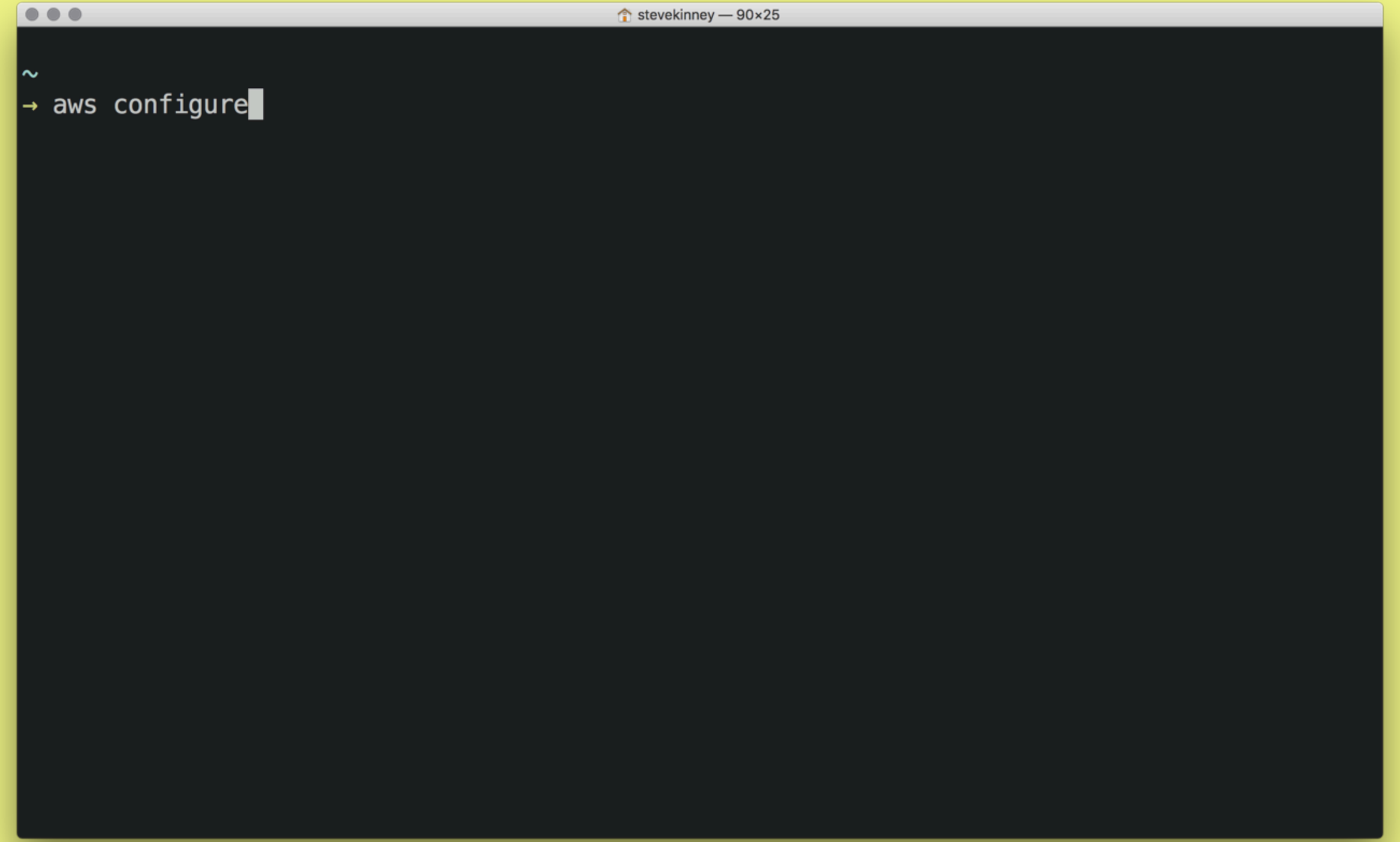
Together

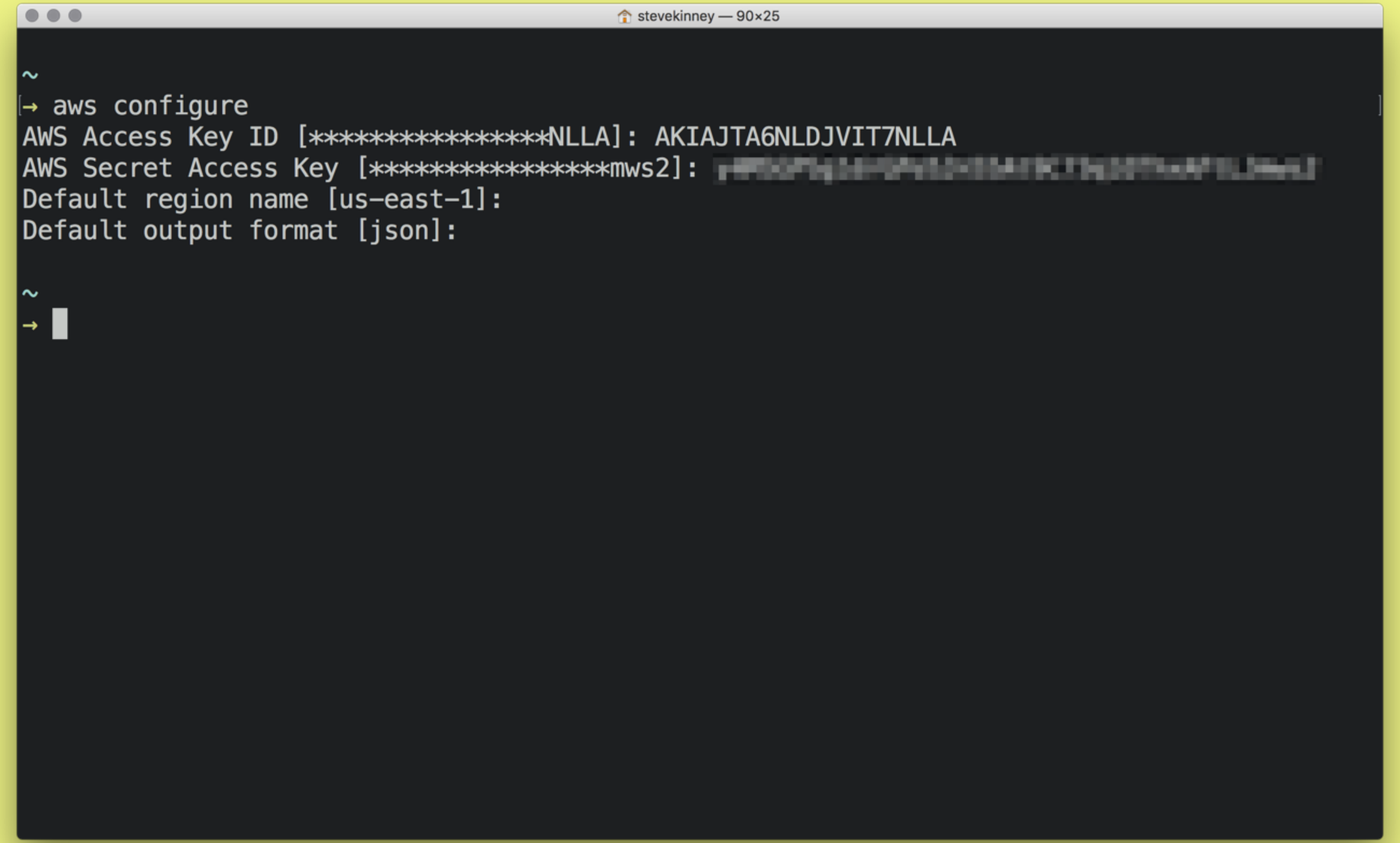
- We’re going to set up an S3 bucket.
- We’re going to set a policy on the bucket.
- We’re going to configure the S3 bucket for static website hosting.
- We’re going to upload a simple React application via the command line.

Areas of Improvement

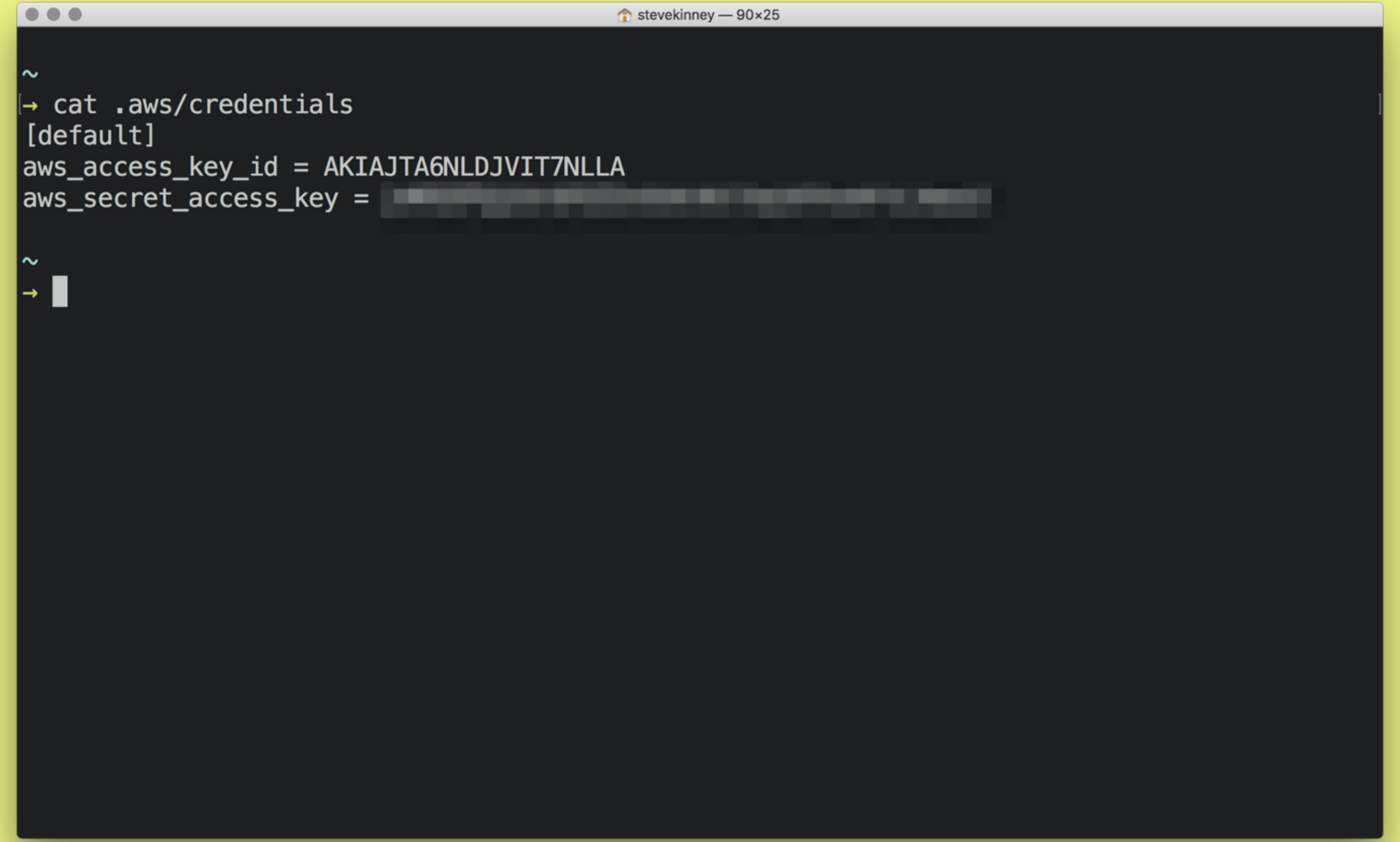
- The URL isn’t great.
- Doing this manually can get tedious.
- It’s hosted in Virginia. (No offense to Virginia.)
- Routing is kind of breaking the web.

We’ll fix all of these things in due time.


stevekinney — 90×25
~
→ aws configure


stevekinney — 90×25
~
→ aws configure
AWS Access Key ID [**********NLLA]: AKIAJTA6NLDJVIT7NLLA
AWS Secret Access Key [**********mws2]:
Default region name [us-east-1]:
Default output format [json]:
~
→

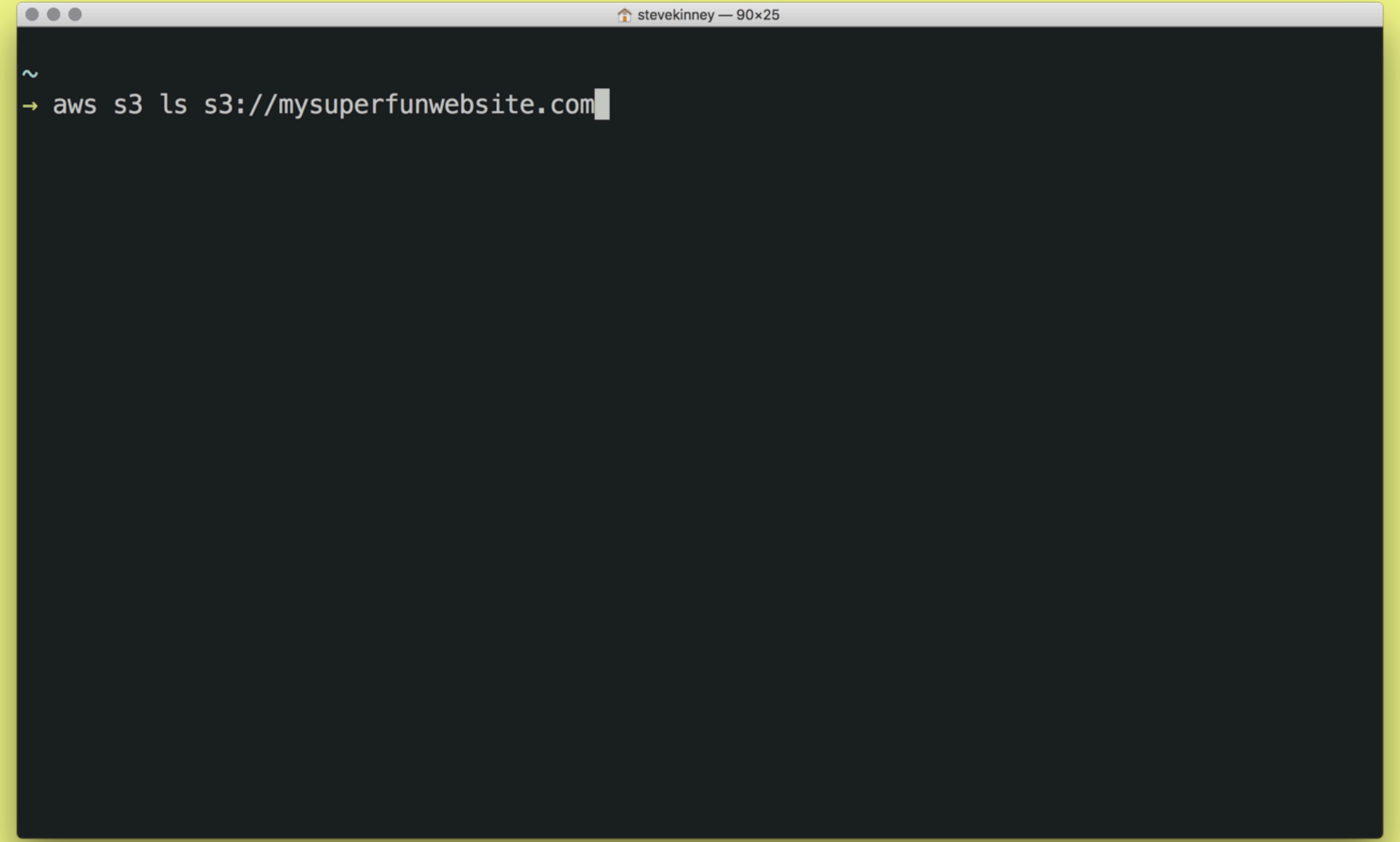
You can also support multiple profiles with the AWS CLI.

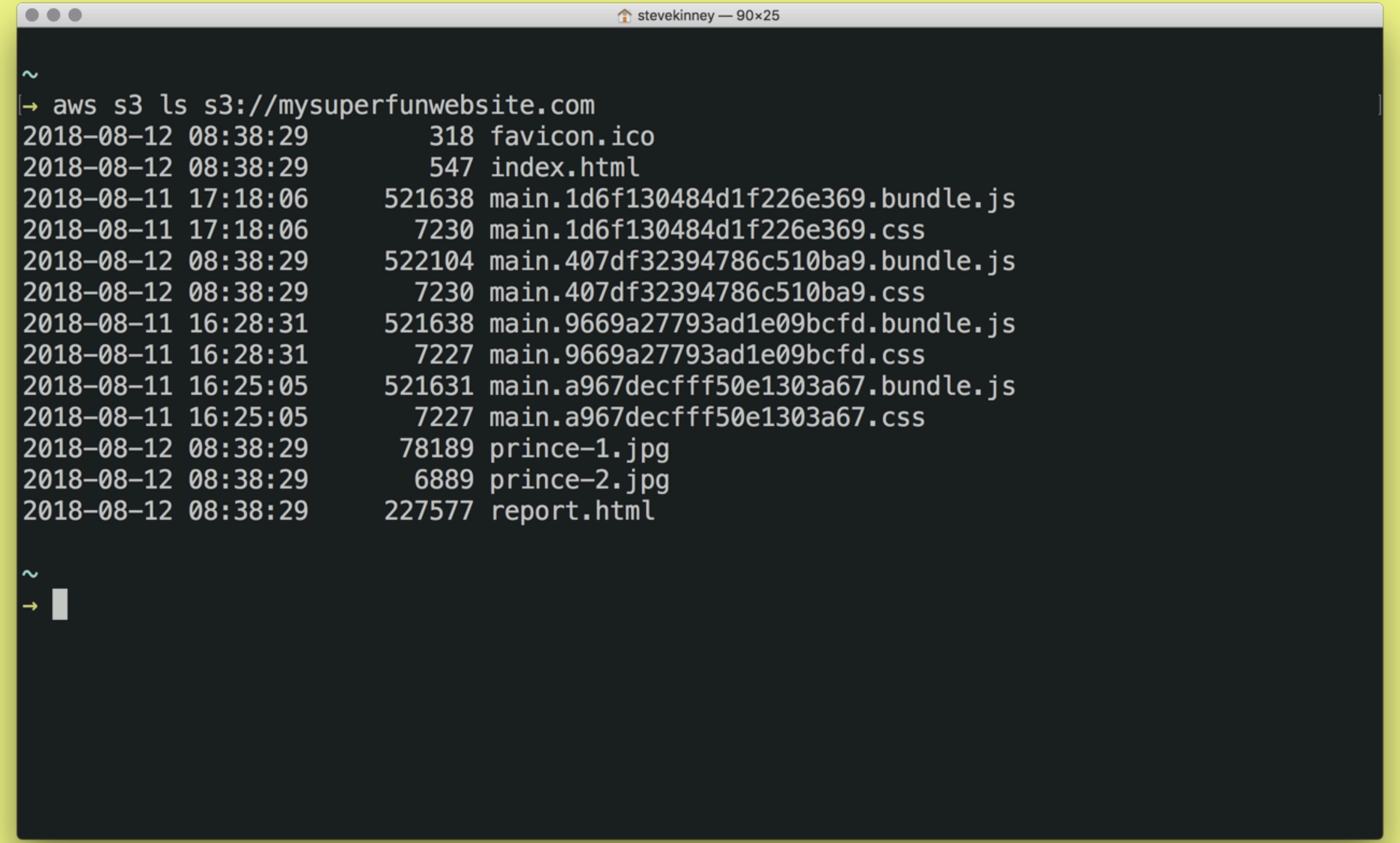

stevekinney — 90×25
~
→ cat .aws/credentials
[default]
aws_access_key_id = AKIAJTA6NLDJVIT7NLLA
aws_secret_access_key =
~
→

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMxIbNOTREAL

[sideproject]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbTOTALLYFAKE
```

Anything you can do in the console you can do with the CLI.


stevekinney — 90×25
~
→ aws s3 ls s3://mysuperfunwebsite.com

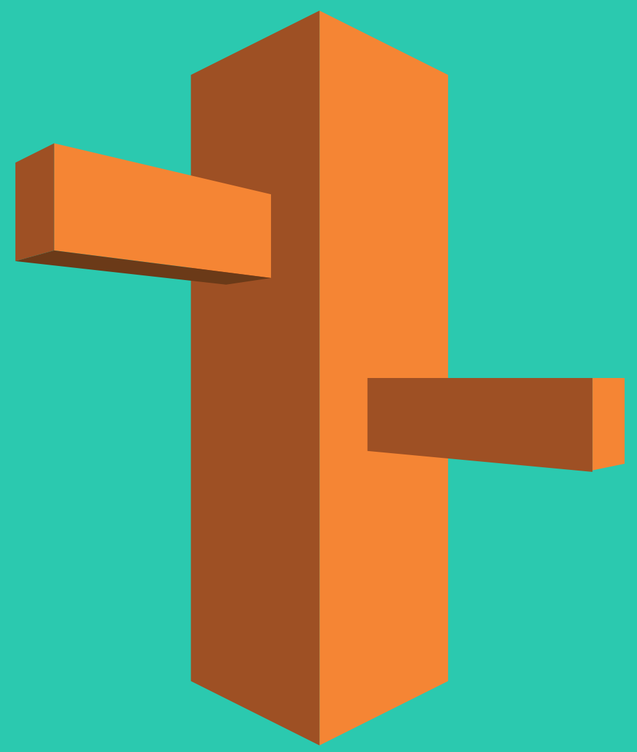

stevekinney — 90×25
~
→ aws s3 ls s3://mysuperfunwebsite.com
2018-08-12 08:38:29   318 favicon.ico
2018-08-12 08:38:29   547 index.html
2018-08-11 17:18:06   521638 main.1d6f130484d1f226e369.bundle.js
2018-08-11 17:18:06   7230 main.1d6f130484d1f226e369.css
2018-08-12 08:38:29   522104 main.407df32394786c510ba9.bundle.js
2018-08-12 08:38:29   7230 main.407df32394786c510ba9.css
2018-08-11 16:28:31   521638 main.9669a27793ad1e09bcfd.bundle.js
2018-08-11 16:28:31   7227 main.9669a27793ad1e09bcfd.css
2018-08-11 16:25:05   521631 main.a967decfff50e1303a67.bundle.js
2018-08-11 16:25:05   7227 main.a967decfff50e1303a67.css
2018-08-12 08:38:29   78189 prince-1.jpg
2018-08-12 08:38:29   6889 prince-2.jpg
2018-08-12 08:38:29   227577 report.html
~
→

Copying a Directory to S3

```
aws s3 cp dist/ s3://mysuperfunwebsite.com/ --recursive
```

Exercise

- Take a few minutes to configure your the AWS CLI for your account.



Route 53 is a scalable DNS service.

Domain **N**ame **S**ervice

awesomeapp.io → 52.84.82.66

Together

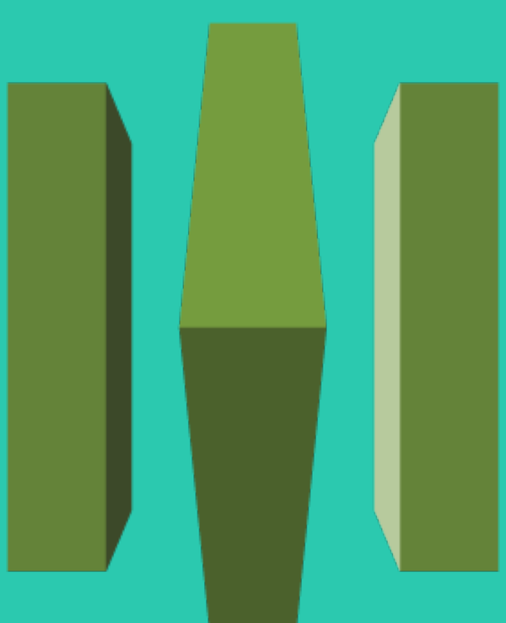
- We’re going to create an A record that aliases to our base bucket.

Exercise

- Create a second alias for your “www” bucket.
- Same process as before, but this time you want to add a subdomain and point to the bucket that has the “www” prefix.

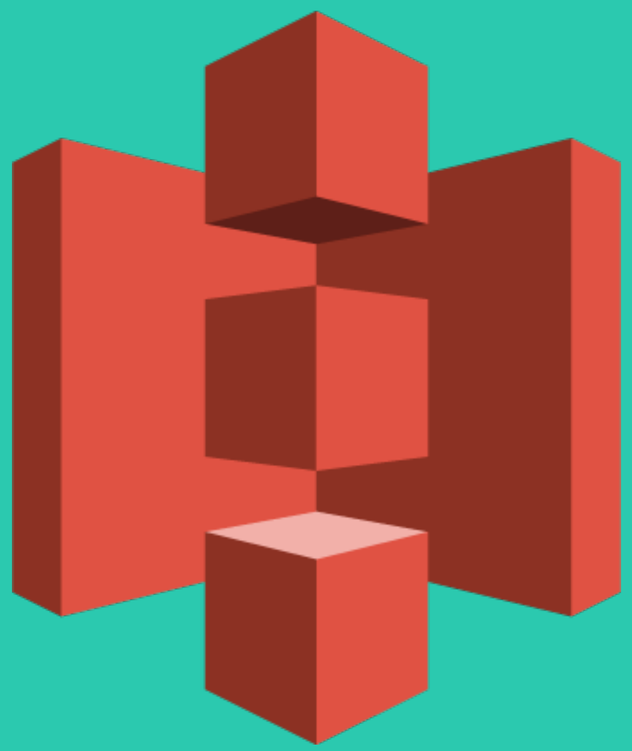
Areas of Improvement

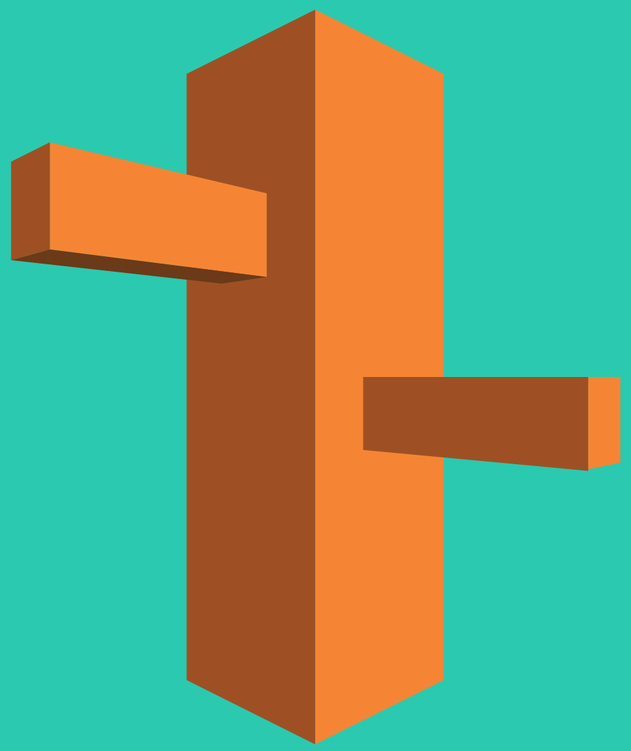
- The URL isn’t ~~great~~ over HTTPS.
- Doing this manually can get tedious.
- It’s hosted in Virginia. (No offense to Virginia.)
- Routing is still breaking the web.



Together

- We’re going to generate a free SSL certificate for our fancy new domain name.
- This will allow us to support HTTPS when we set up CloudFront.





Navigating to a note works fine—but what if we go directly to that URL?

Together

- Let’s see where we can set up a better error page.
- Let’s not even use that error page.
- We’ll just act like nothing ever went wrong.

Let's get www.<your-domain> working.

Exercise

- Make a new bucket called `www.<your-original-bucket-name>`.
- Go to Static Web Hosting.
- Set it up to redirect to your main bucket.
- Verify that it works.
- (We'll do this together shortly.)

Areas of Improvement

- ~~The URL isn’t great.~~ (It’s great, just not secure.)
- Doing this manually can get tedious.
- It’s hosted in Virginia. (No offense to Virginia.)
- Routing is **still** kind of breaking the web.

We’re going to do this out of order.

- As my co-worker, Steven, likes to say CloudFront puts the “eventual” in eventual consistency.
- Everything in CloudFront takes a while, so we’re going to just set it up now and then we’ll talk about it while it’s cooking.

Together

- We’re going to create a new CloudFront distribution.
- We’re going to point it to our static website on S3.
- We’ll add our domain names.
- We’ll set up gzipping for our assets.
- We’ll set a default root object.

So, as I was saying...

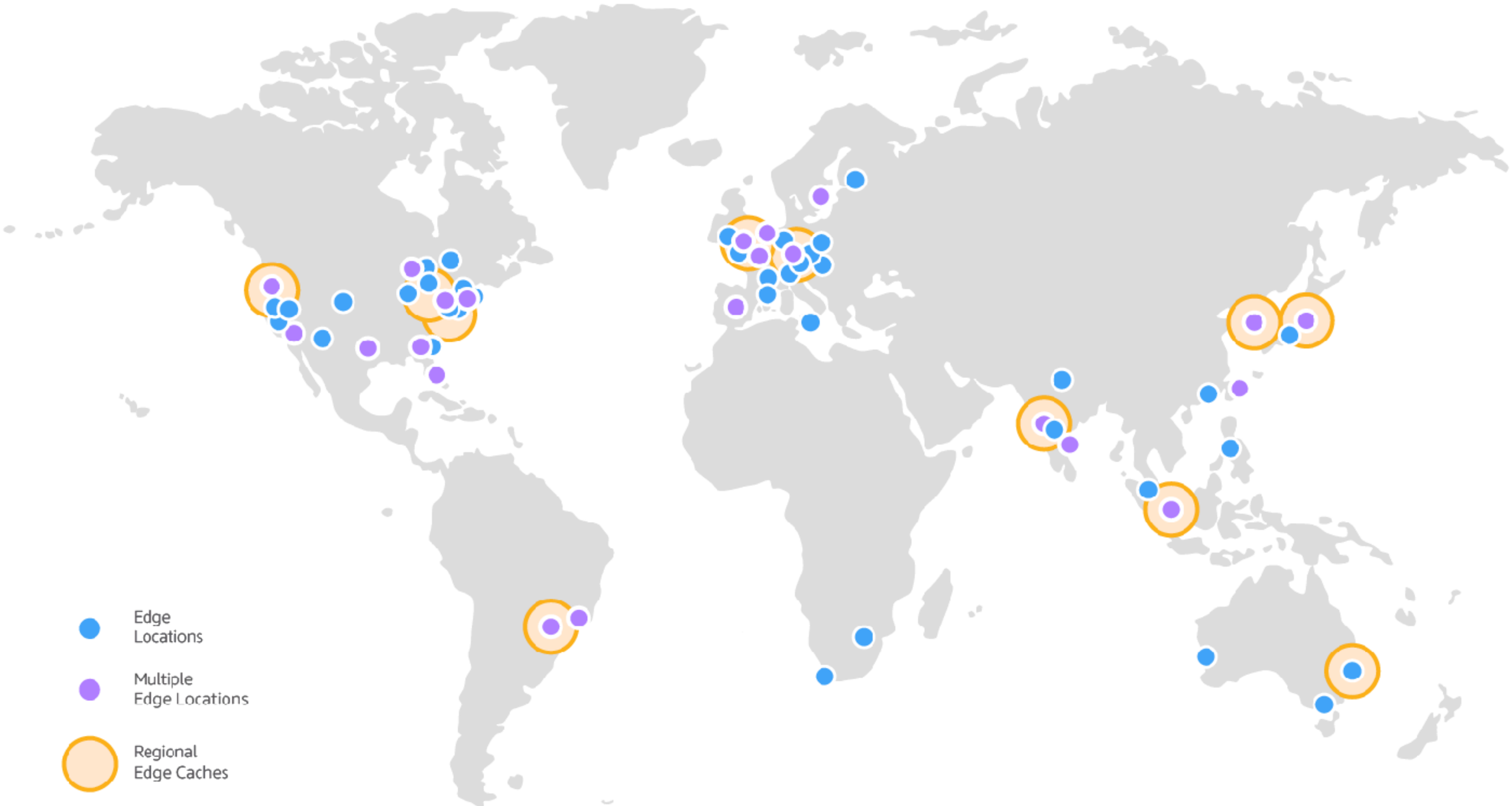
Our application is currently hosted in Virginia.

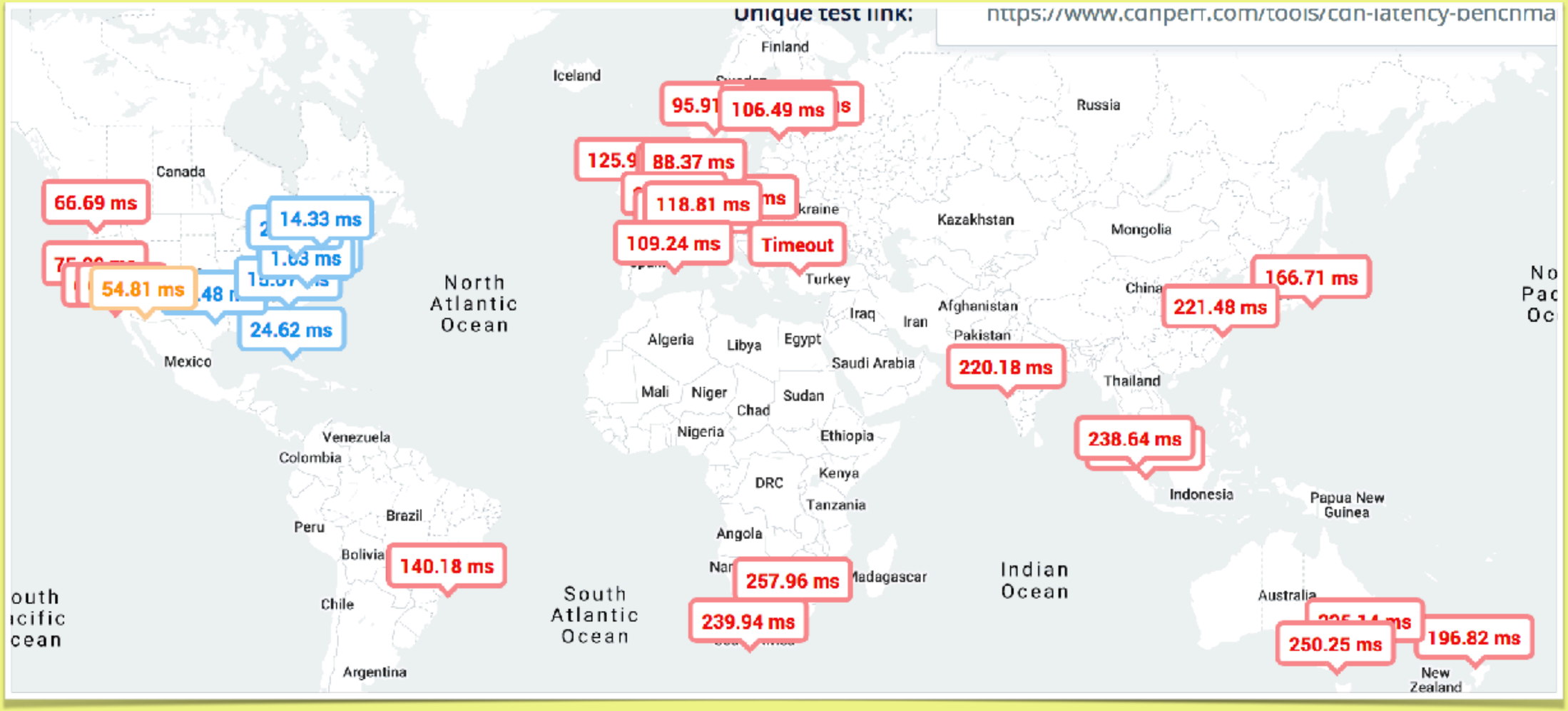
Fun fact: Some places are further from Virginia than others.

It will take those people longer to get the application because of physics.

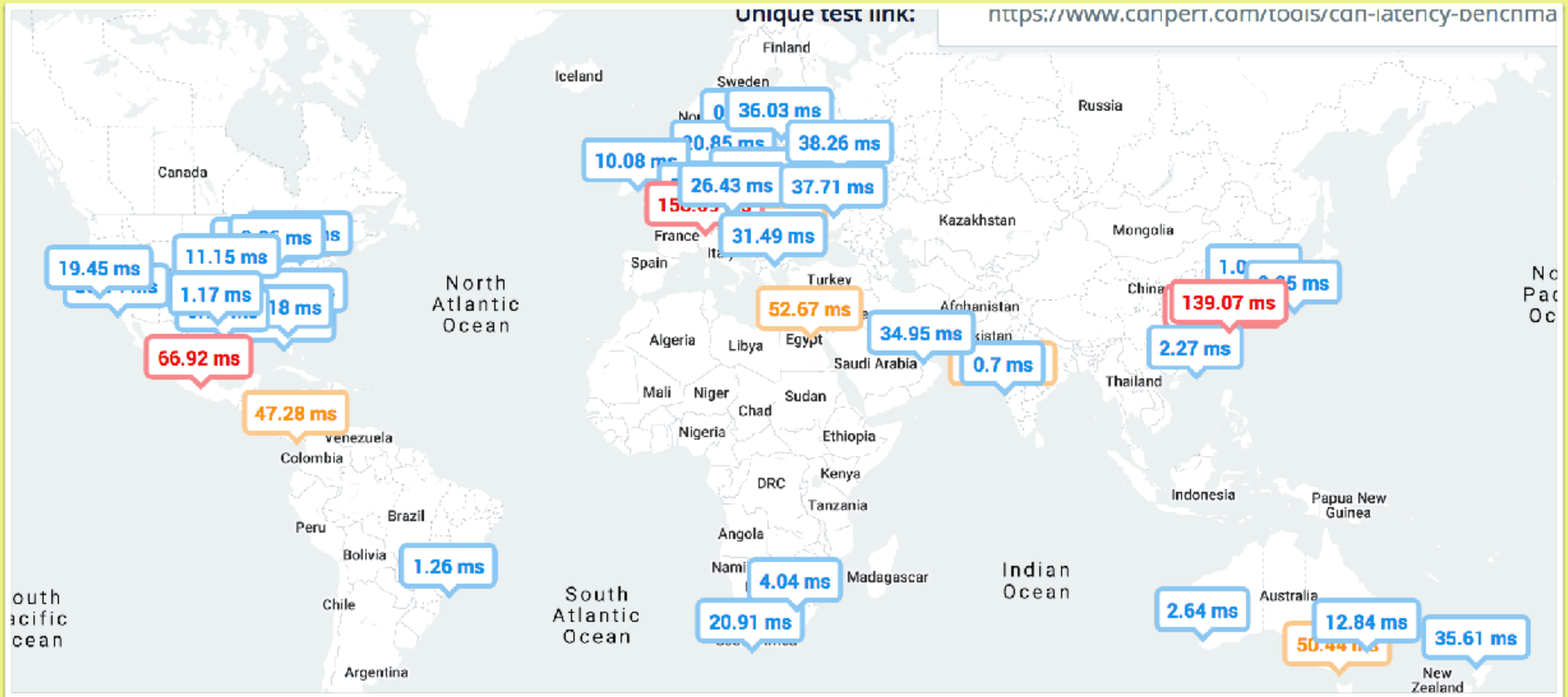
We *could* try to figure out where most of our users are and then put it as close to them as possible.

Or, we could just put it *everywhere.*


Edge
Locations
Multiple
Edge Locations
Regional
Edge Caches

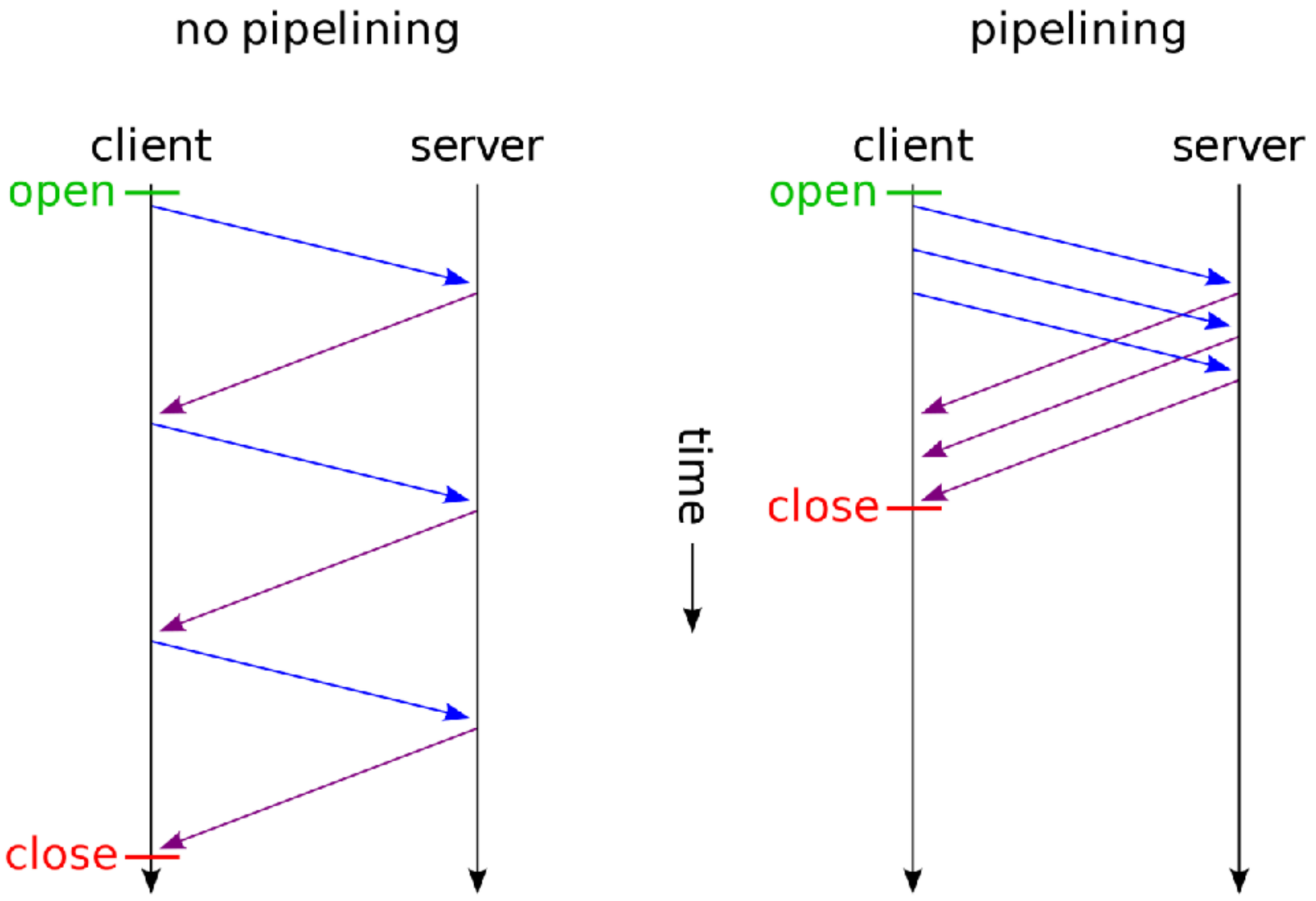


https://www.cdnperf.com/tools/cdn-latency-benchmark/?id=b0a01b2cb7466a0af3bf45855ea23358



https://www.cdnperf.com/tools/cdn-latency-benchmark/?id=24c4ade324bc752e2e561dc1ace88078

CloudFront also supports HTTP/2 out of the box.



Bonus: CloudFront will cache our assets in S3, which reduces the number of requests.

This is nice because it reduces our costs. 

A Word on Headers

By default, CloudFront ignores request headers.

Some CloudFront Custom Headers

- CloudFront-Is-Desktop-Viewer
- CloudFront-Is-Mobile-Viewer
- CloudFront-Is-SmartTV-Viewer
- CloudFront-Is-Tablet-Viewer
- CloudFront-Viewer-Country

Exercise

- We'll visit our CloudFront distribution and verify that it's up and running.

- Make a change to application (e.g. make a background color something ridiculous).

- Deploy the application.

Cache Invalidation: One of the three hard problems in computer science.

Strategies

- **Best when possible**: Don’t do it. Use unique names.

- **Alternative**: Bust the cache when needed.

Areas of Improvement

- ~~The URL isn’t great or secure.~~
- Doing this manually can get tedious.
- ~~It’s hosting in Virginia. (No offense to Virginia, but it’s no New Jersey.)~~
- Yea, routing is still weird. ¯_(ツ)_/¯

Disclaimer

- I opted to use Travis CI because it's free for public repositories and easy to set up.
- You can do this with *any* CI tool. (We used to do it with Jenkins at **SendGrid**, now we use Buildkite.)

Together

- We’ll add our repository to Travis CI.
- We’ll create a configuration for Travis CI.
- We’ll deploy our application and witness the magic and glory.

Exercise

- Make another obnoxious change to your application.
- Merge it into master and push it up.
- Behold the glory of your creation.

Tasting Notes

- You eventually need to bust the cache on CloudFront.
- You can do this with a relatively low TTL on the cache header *or* you can do this with invalidations.
- You get up 1,000 invalidations per month for free.
- You can have your CI/CD tool take care of all of this when merging to master.

Areas of Improvement

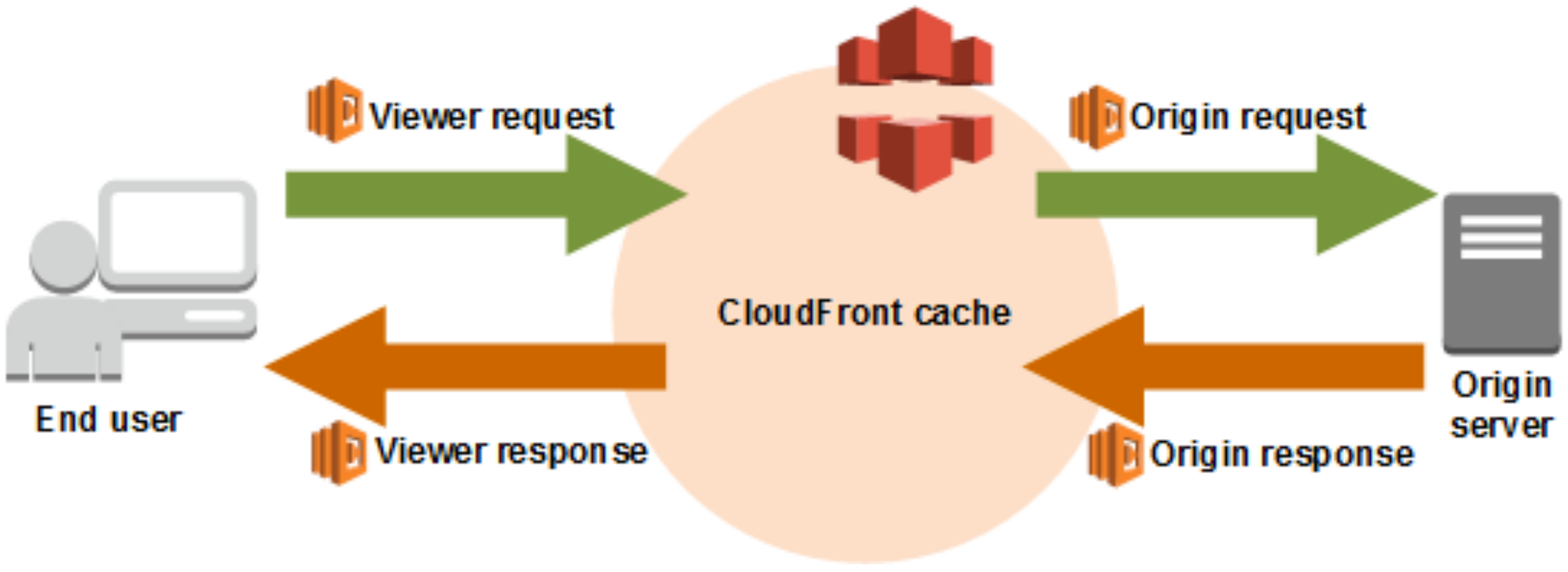
- ~~The URL isn’t great.~~
- ~~Doing this manually can get tedious.~~
- ~~It’s hosting in Virginia. (No offense to Virginia.)~~
- Oh yea—routing.

Lambda functions allow you to execute code in response to events.

Lambda@Edge allows you to deploy functions to your CloudFront edge nodes.

These functions can intercept and modify requests and responses.

Lambda@Edge


Viewer request
Origin request
CloudFront cache
End user
Origin server
Viewer response
Origin response

Viewer Request

- Executed on every request before CloudFront’s cache is checked.
- Modify the cache key (URL, cookies, headers, query string).
- Perform authentication and authorization checks.
- Make additional network calls.
- Generate responses (uncached).

Origin Request

- Executed on a cache miss, before the request is forwarded to the origin.
- You can make external network calls.
- Dynamically set an origin based on the headers.
- Re-write URLs (pretty URLs).
- Cool for stuff like internationalization.

Origin Response

- Responses from the origin that haven’t been cached yet.
- Executed on cache miss, after a response is received from the origin.
- Make external network calls.
- Modify response headers prior to caching.

Viewer Response

- Executed on all requests, after a response is received from the origin or cache.
- Modify the response headers without caching the result.
- Make external network calls.

One drawback: It's unclear where your logs are going to be.

Lambda@Edge functions send their logs to the nearest region—not the region they're replicated from.

A quick aside about a Friday night argument with one of the architects at SendGrid.

Together

- Let's fix routing: If the viewer is looking for a known client-side route (e.g. `/notes/1`), we'll get them the index.html instead.
- Let's adjust CloudFront to go back to treating 404s as 404s.

Exercise

- We want to look at each *Viewer Request*. We're going to add some more to our previous Lambda.

- If the `request.uri` is `/prince-1.jpg`, we want to change it to `/prince-2.jpg`.

Areas of Improvement

- ~~The URL isn’t great.~~
- ~~Doing this manually can get tedious.~~
- ~~It’s hosting in Virginia. (No offense to Virginia.)~~
- ~~Oh yea—routing.~~

Additional Things

- Implement A/B testing.
- URL redirection (pretty URLs).
- Header normalization.
- Redirecting unauthenticated users to the login page.

Conclusion

Conclusion

- The needs of your particular application will probably differ slightly.
- But, there is probably a solution using AWS.
- Some of this grunt work can be further automated via tools like Terraform or CloudFormation, but there is a cost/benefit analysis that needs to happen there.

```
provider "aws" {
  access_key = "${var.aws_access_key}"
  secret_key = "${var.aws_secret_key}"
  region     = "us-east-1"
}

# Create a web server
resource "aws_instance" "web" {
  # ...
}
```

Advantages

- Infrastructure as code means that infrastructure changes can go through code review.
- It means they can be checked into Git.
- It means it *can* be made reusable.

Fin.