

CS 188

Scalable Internet Services

Andrew Mutz

October 11, 2016



Today's Agenda

Motivation

High Availability

- HA datacenter design
- HA on AWS

Client-side Caching

For Next Time



Motivation

As we've discussed, more and more of our daily lives depend on software.

If software is indeed eating the world, what happens when software systems fail?



Motivation

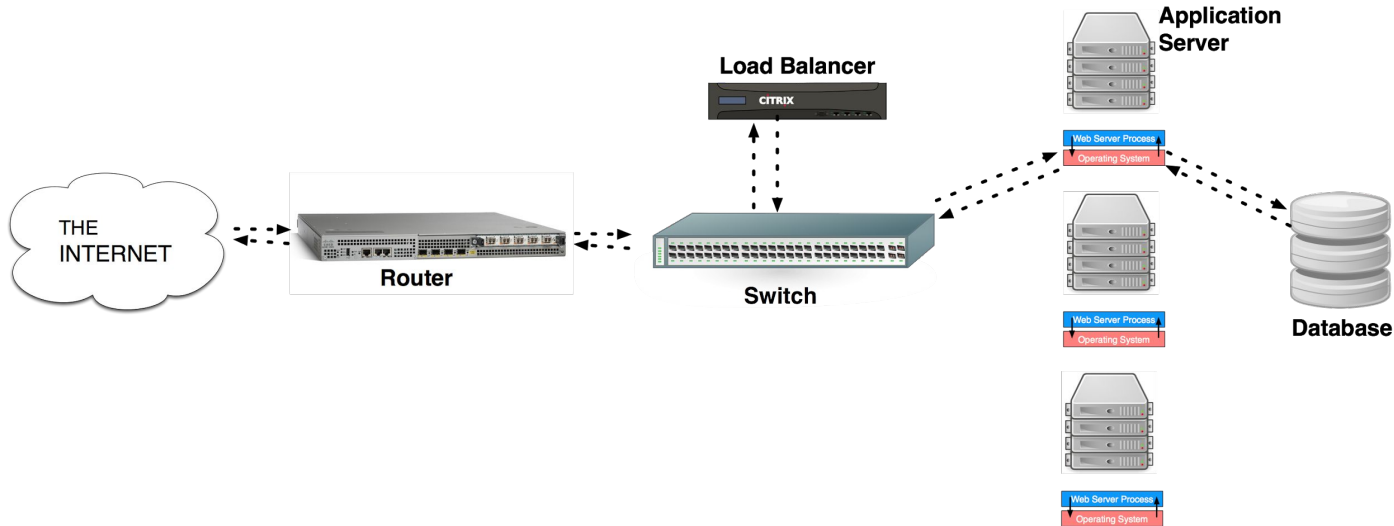
Modern web applications power some very important parts of our lives

- Banking, Medical, Telephony (increasingly), etc.
- High availability is increasingly important.
- A common phrase targeted by businesses is “X nines”
 - Three nines = 99.9% uptime = ~ 45 minutes a month down
 - Four nines = 99.99% uptime = ~ 5 minutes a month down
 - Business applications
 - Five nines = 99.999% uptime = ~ five minutes a year down
 - Communications companies



High Availability

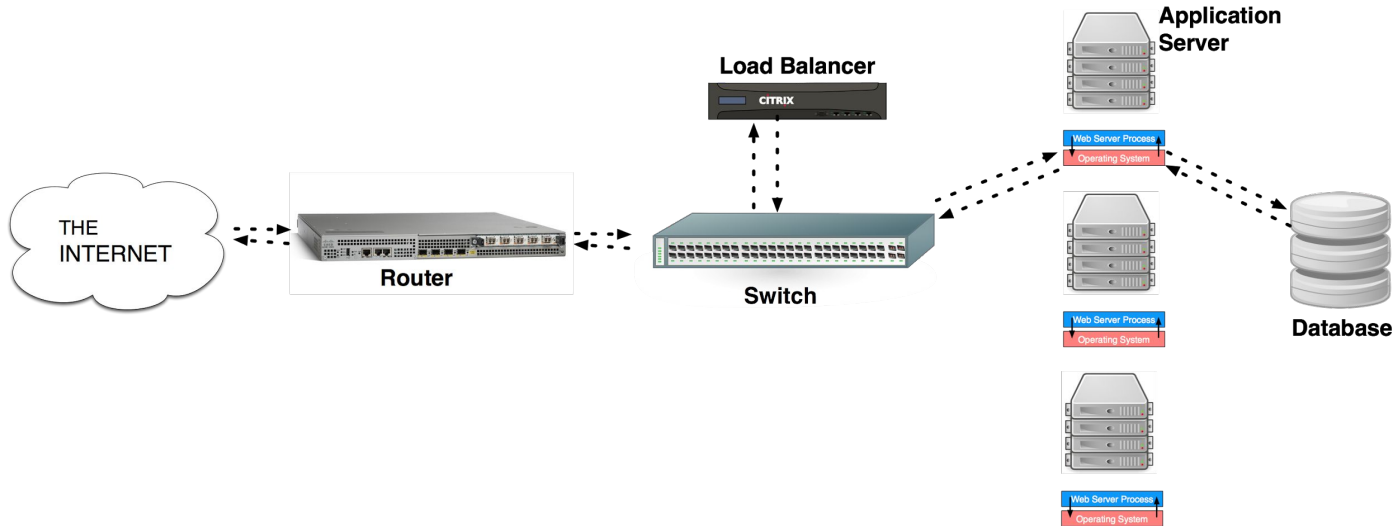
What are possible causes of failures?



High Availability

What are possible causes of failures?

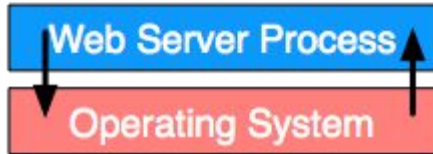
- Server process dies?
- Application server fails?
- Load balancer fails?
- Switch fails?
- Internet fails?
- Database fails?
- Entire datacenter fails?



High Availability



Application Server Fails?



High Availability

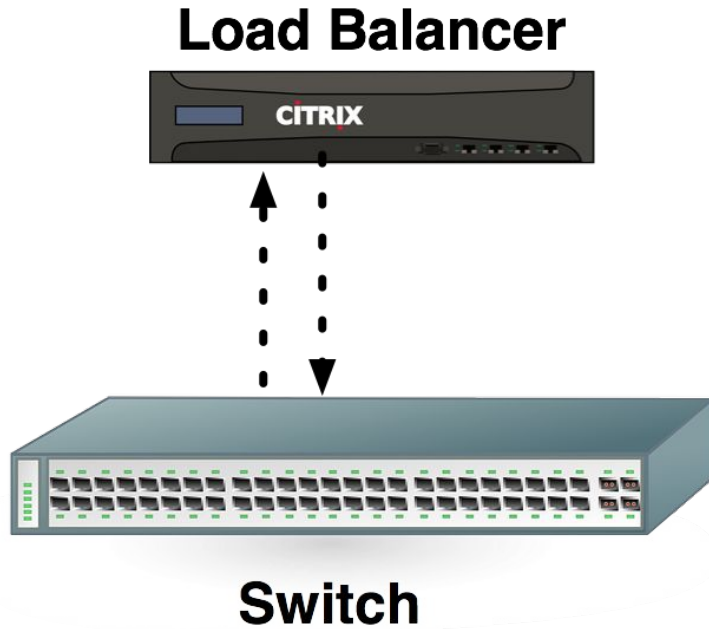


Application Server Fails?

- We've already done a lot here
- Having process-level isolation reduces disruptions to a single process failure
- Our load-balanced configuration means any single app server can go down and we can direct load elsewhere



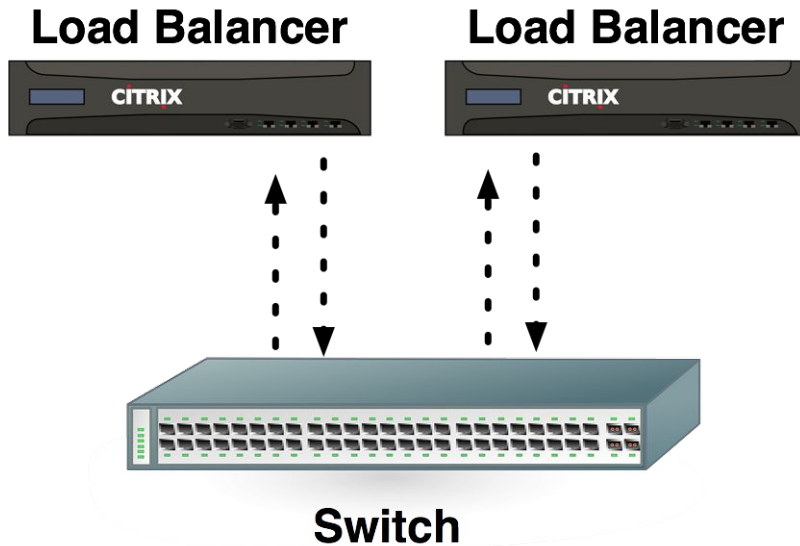
High Availability



Load balancer fails?



High Availability

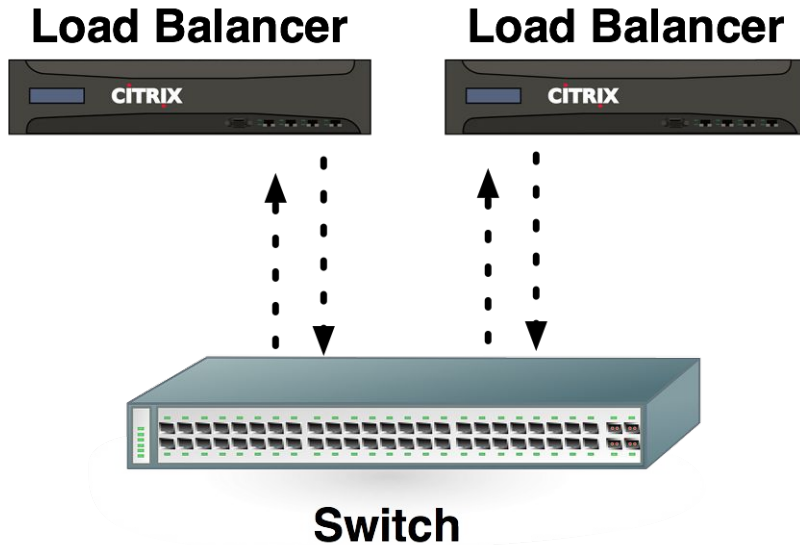


Load balancer fails?

- Lets buy two: primary & failover
- How do we detect when failure has occurred?



High Availability

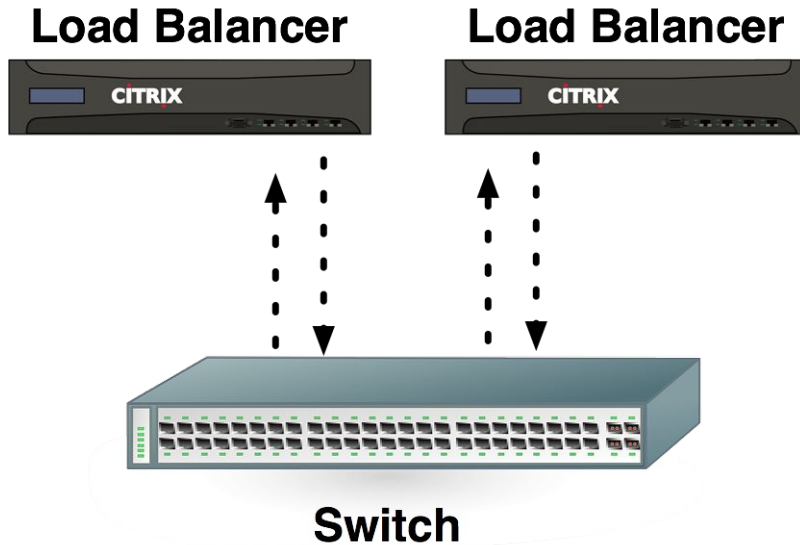


Load balancer fails?

- Load balancers use heartbeats to determine health
- During failover, what happens to
 - Established flows?
 - IP address?



High Availability

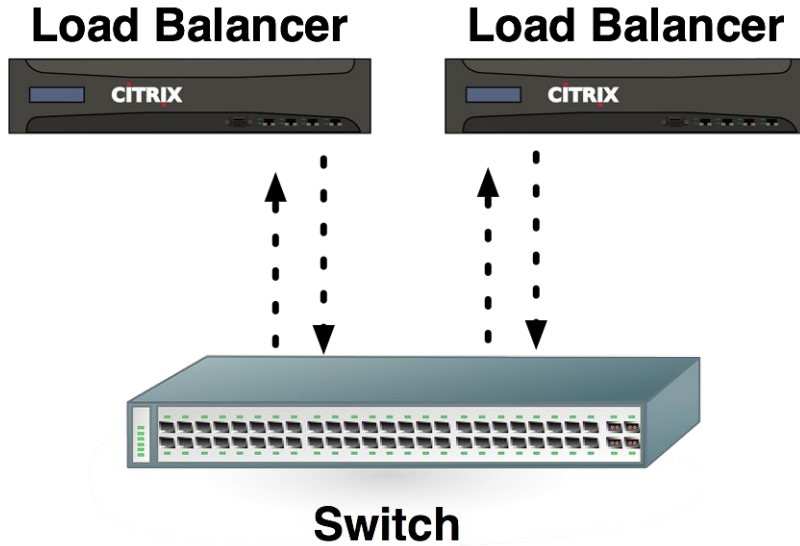


Load balancer fails?

- Established flows & IP address
 - When secondary determines it needs to step in, it issues a Gratuitous ARP
 - Other devices on the network that were communicating with the primary cleanly switch over to secondary
 - Established flows can be supported



High Availability

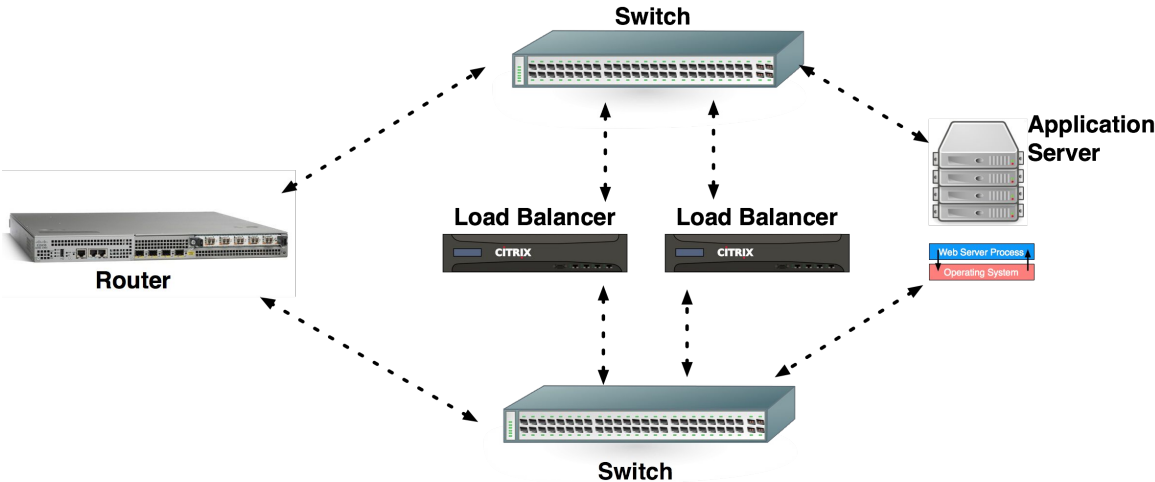


Switch fails?



High Availability

Switch fails?



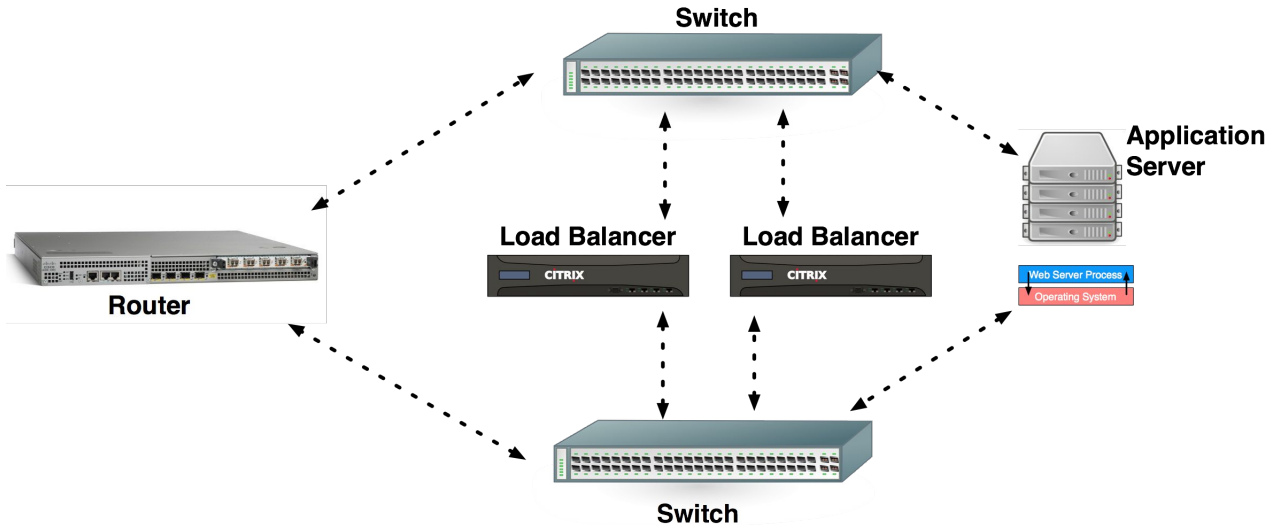
Lets buy two!

- Link aggregation allows multiple interfaces to share a MAC address
- MC-LAG, is Multi-Chassis Link Aggregation
- Link Aggregation Control Protocol handles failure detection
- Failure is simple: no sessions to maintain.



High Availability

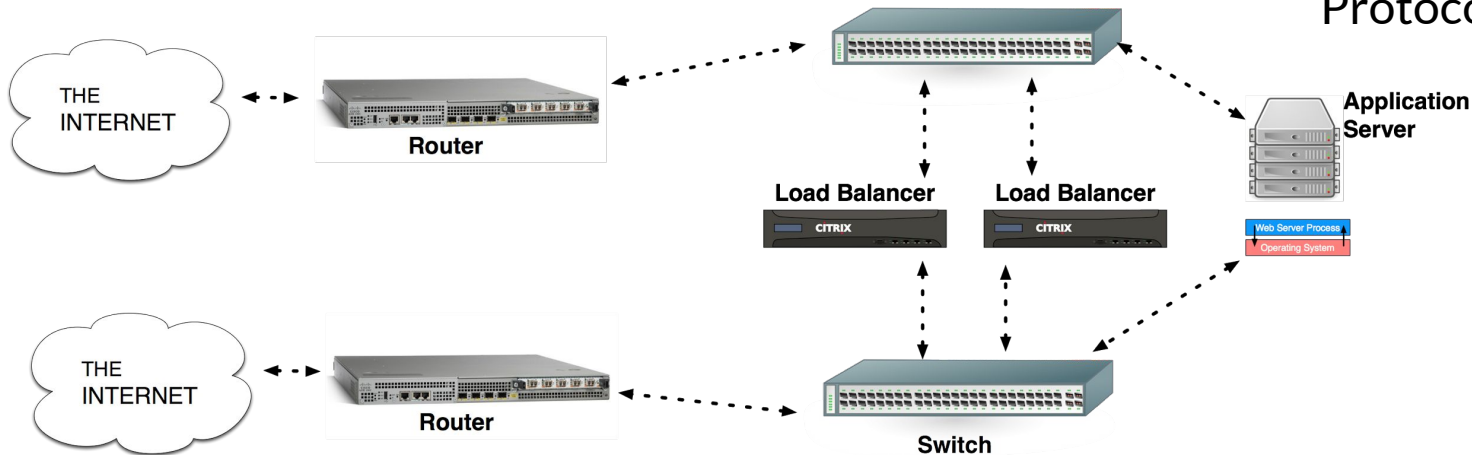
Router fails?



High Availability

Router fails?

- Lets buy two!
- Similar heartbeat and failover system (Hot Standby Routing Protocol)



High Availability

Internet fails?



High Availability

Internet fails?

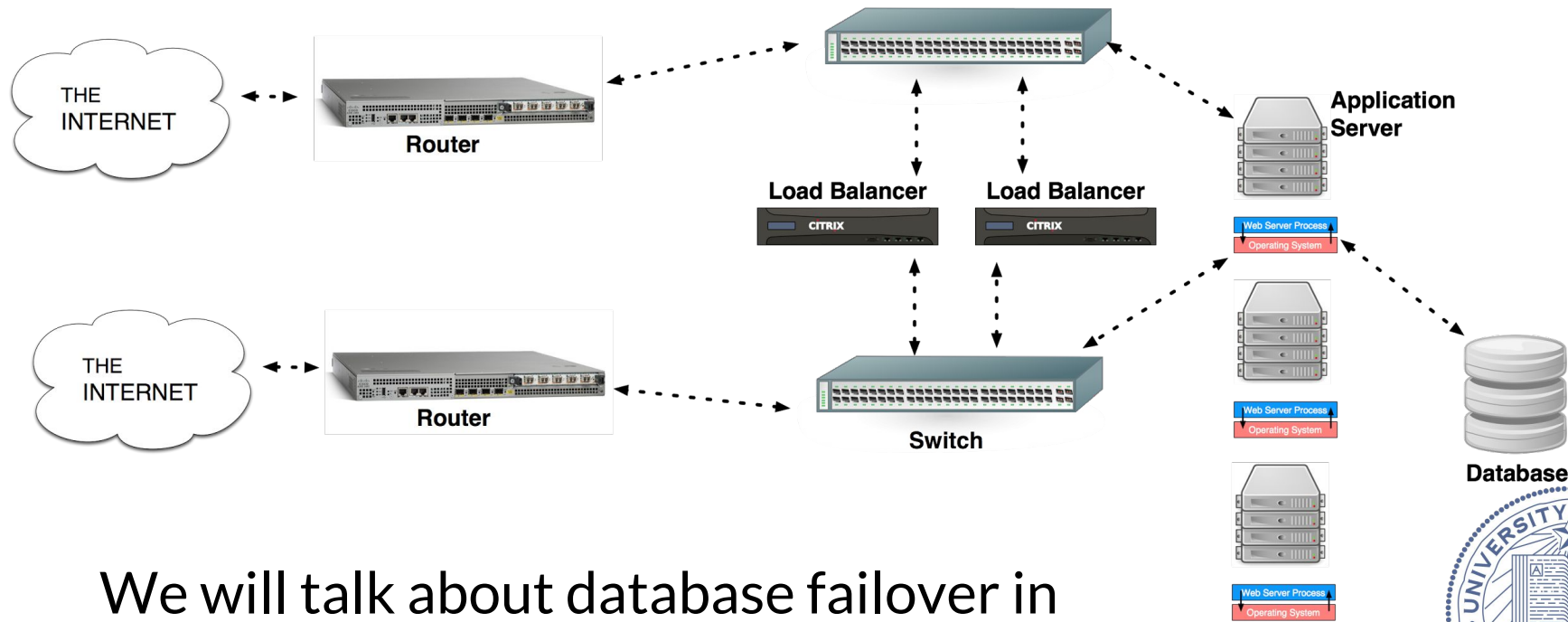
- Lets have two ISPs.
 - One link to, say, Sprint and another to Internap.

How do we handle routing when we have two ISPs?

- Outgoing traffic is easy, since we control these decisions.
 - Pick the cheapest or most reliable link
 - Pick the “closer” link
- Incoming traffic is hard
 - We can't directly tell clients how to reach our web app
 - We need to use BGP to persuade clients
 - Prepending, community strings



High Availability



We will talk about database failover in future lectures



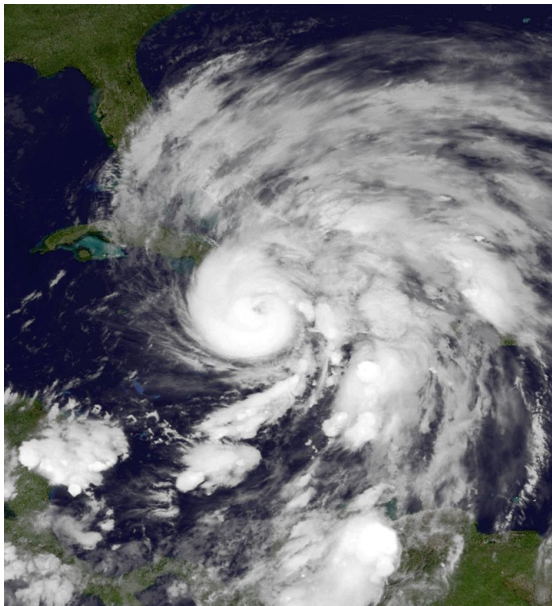
High Availability

Ok so we're good right? Nothing can go wrong?



High Availability

Ok so we're good right? Nothing can go wrong?



High Availability

Hurricane Sandy takes data centers offline with flooding, power outages

Hosting customers stranded as generators in NY data centers run out of fuel.

by Jon Brodtkin - Oct 30 2012, 9:25am PDT

 Share

 Tweet

100



High Availability

Availability Axiom (Pete Tenereillo):

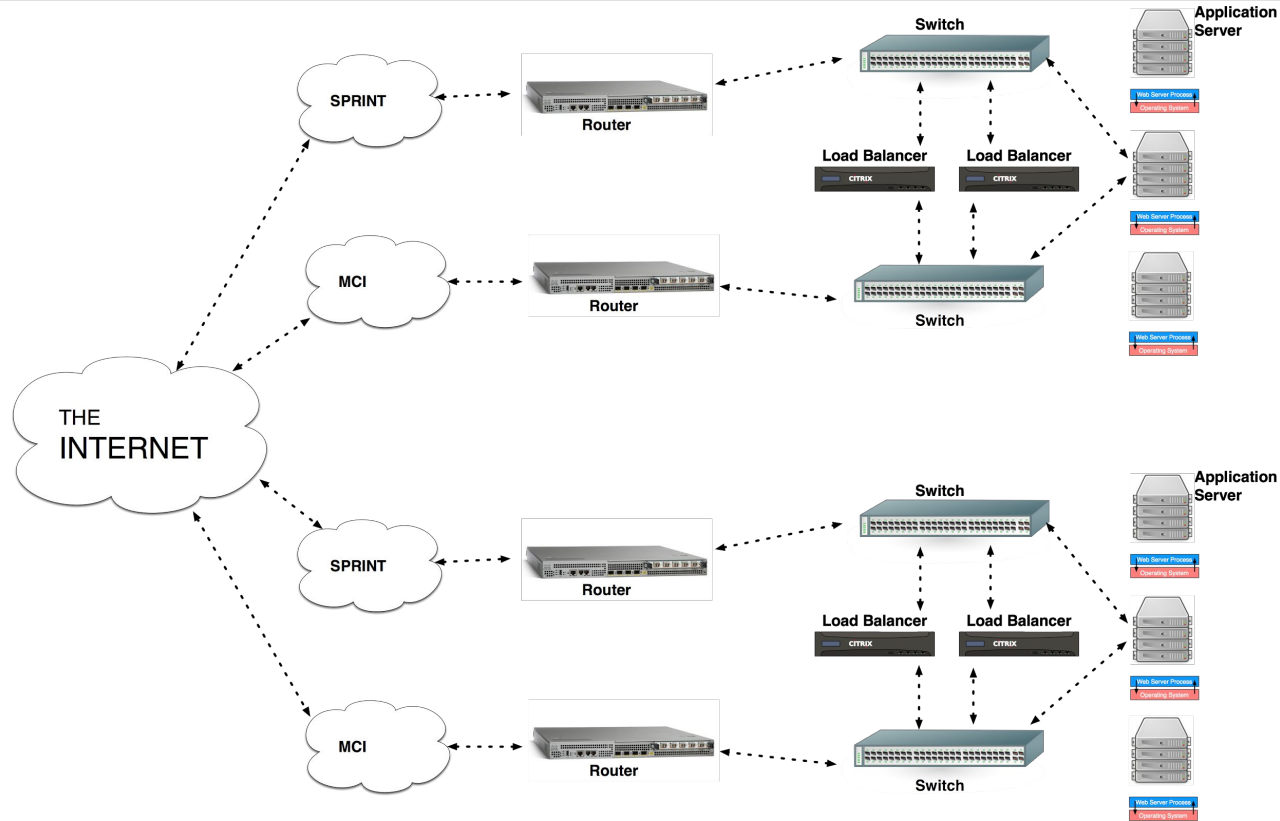
- The only way to achieve high-availability for browser based clients is the include the use of multiple A-records (DNS).

Result:

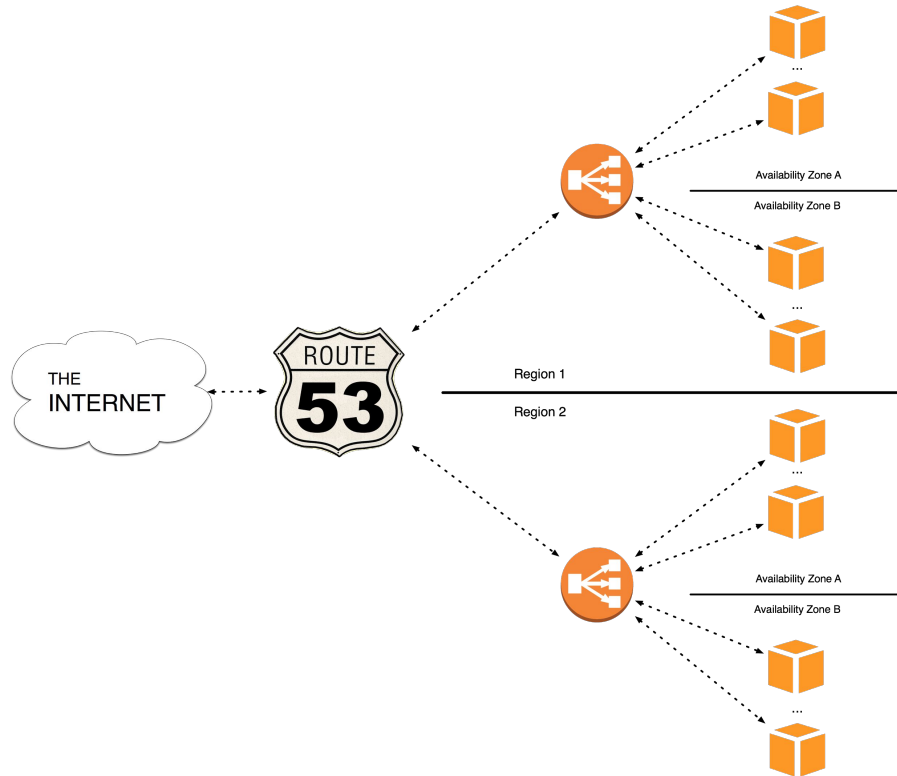
- For performance, we want to send the browser to one datacenter.
- For availability, we want to send the browser multiple A records.
- We end up having to make a choice between performance and availability.



High Availability



High Availability on AWS



AWS has regions and availability zones

- **Region:** think a city
- **Availability zone:** think a data center

Failures between availability zones are not correlated*.



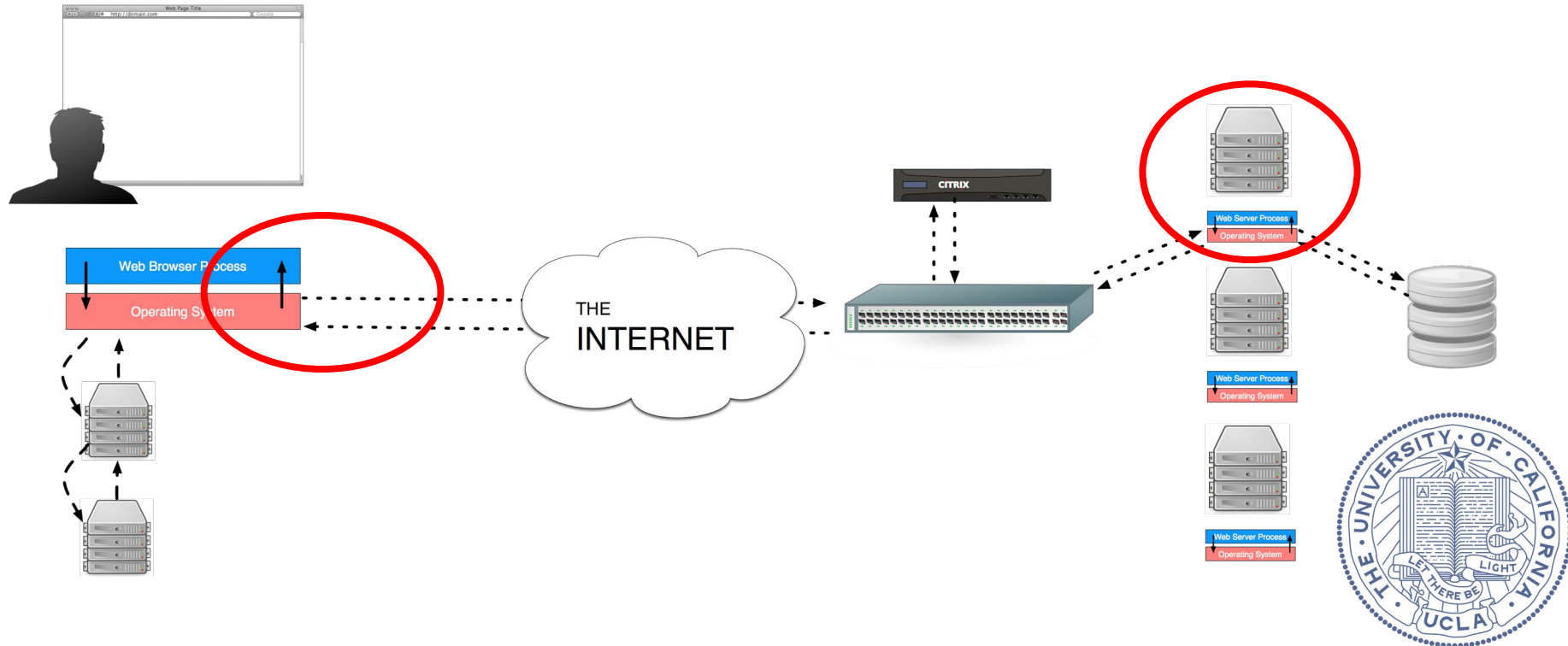
For your projects

The cloudformation templates I provide you emphasize testing scaling over High Availability.

Advanced students can still do HA, but it will take more configuration work on your part.



Client-side Caching



Motivation

We want our important application data persisted safely in our data center.

And it needs to be regularly read and updated by geographically distributed clients.

And it needs to be fast.



Motivation

Performance Matters!

| Delay | User Reaction |
|---------------|-------------------------------|
| 0 - 100 ms | Instant |
| 100 - 300 ms | Slight perceptible delay |
| 300 - 1000 ms | Task focus, perceptible delay |
| 1 second+ | Mental context switch |
| 10 seconds+ | I'll come back later... |

Source: Ilya Grigorik (igvita.com)



Motivation

But there are challenges:

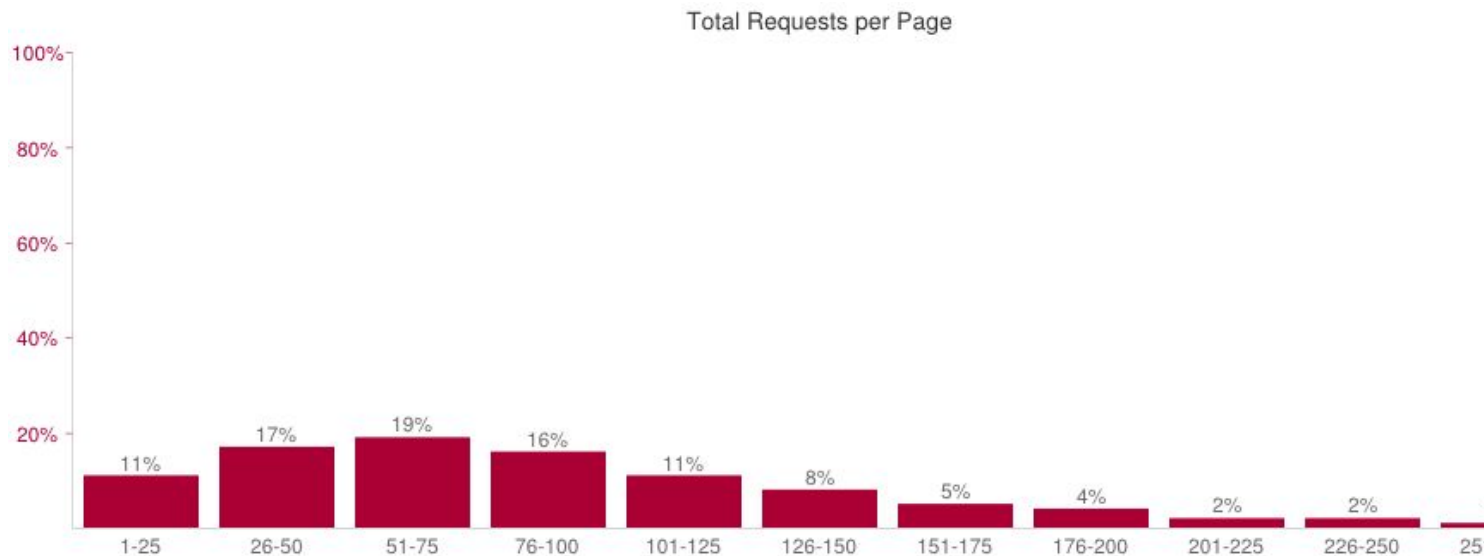
| Route | Distance | Time, light in vacuum | Time, light in fiber |
|---------------|-----------|-----------------------|----------------------|
| NYC to SF | 4,148 km | 14 ms | 21 ms |
| NYC to London | 5,585 km | 19 ms | 28 ms |
| NYC to Sydney | 15,993 km | 53 ms | 80 ms |
| Equator | 40,075 km | 133 ms | 200 ms |

Source: High Performance Browser Networking, Ilya Grigorik



Motivation

A page is more than a single request:



Source: <http://httparchive.org/>



Motivation

The fastest request is the one that never happens!

Cache: a component that transparently stores data so that future requests for that data can be served faster.

Where to introduce caching?

- Inside the browser
- In front of the server (CDNs, etc.)
- Inside the application server
- Inside the database (query cache)



Client-side Caching

How does the browser cache data? How does it know when it can safely present previously seen data as current?

The building blocks are all HTTP headers:

- etag
- cache-control
 - max-age
 - no-cache
 - no-store
 - public | private
- if-modified-since
- if-none-match



Client-side Caching

cache-control : no-store

When accompanying a response, the browser (or intermediate proxy) is instructed to not reuse this data under any circumstances.

This can also used for sensitive information.



Client-side Caching

cache-control : no-cache

When accompanying a response, the browser (or intermediate proxy) is instructed to revalidate before reusing it.

Without this, the browser can use recently seen versions safely.



Client-side Caching

cache-control : private

When accompanying a response, the browser (or intermediate proxy) is instructed that the data is specific to the requesting user.

Intermediate proxies should discard such data, but a single-user browser can reuse it.

The opposite of this is cache-control : public



Client-side Caching

`cache-control : max-age=120`

When accompanying a response, the browser (or intermediate proxy) should consider this copy stale if the specified number of seconds has passed.

The more modern version of the `expires` and `date` headers.



Client-side Caching

etag: “5bf444d26f9f1c74”

When accompanying a response, the browser will keep this “entity tag” along with saved copies of the resource.

When requesting the same resource in the future, this tag can be presented to indicate the version it had previously seen.

This isn't necessarily a digest of the resource that was served up, but can be thought of as such.



Client-side Caching

`if-modified-since: Sun, 19 Oct 2014 19:43:31`

When accompanying a request, this indicates that the client already has a copy that was fresh as of the specified date.

If the server's copy is newer than the specified date, it will be served to the client.

If the server's copy hasn't changed since the specified date, the server will return 304 (not modified).



Client-side Caching

`if-none-match: “5bf444d26f9f1c74”`

When accompanying a request, this indicates that the client has a cached copy with the associated tag. Multiple etags can be provided.

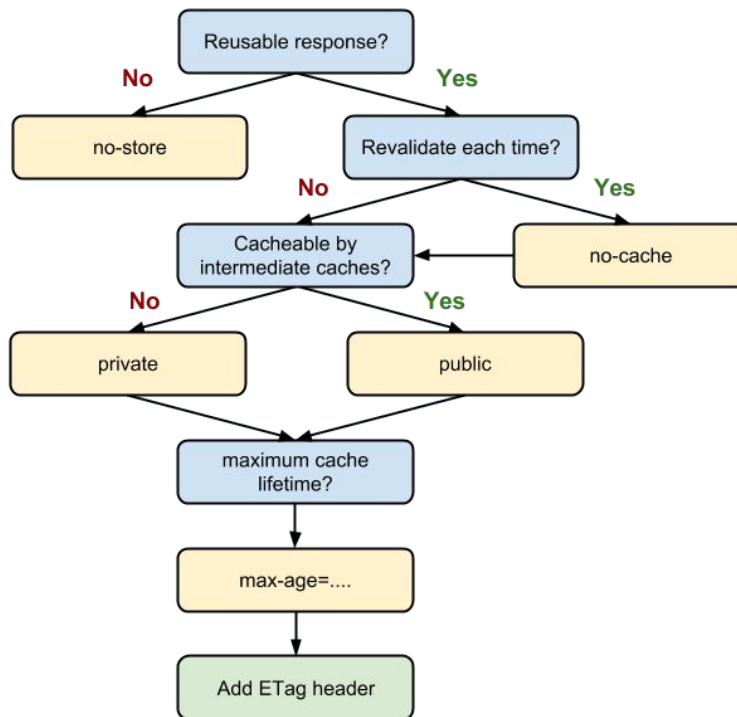
If the server's current version has one of the etags listed, the server will return 304 (not modified) with the etag of the current resource included.

If the server's version has a non-matching etag, then the result will be returned as normal.



Client-side Caching

In Summary...



Client-side Caching

Let's pull this together and apply what we've seen.

Let's say we are serving up some javascript that won't change over the next day, but does have some user-specific code in it.

What headers should the response include?



Client-side Caching

We want it reusable, but private:

Cache-control: private, max-age=86400



Client-side Caching

Let's say we are serving up an image that may be changing in the future, and we never want a stale version shown. The image is not specific to the requestor.

What headers should the response include?



Client-side Caching

We want it reusable with revalidation and public:

Cache-control: public, no-cache

ETag: “4d7a6ca05b5df656”

Clients will request the resource with:

if-none-match: “4d7a6ca05b5df656”



Client-side Caching

Let's say we are serving up an image with the user's social security and credit card numbers.

What headers should the response include?



Client-side Caching

We want it reusable, but private:

Cache-control: private, no-store



Client-side Caching

Demo App

New Submission

Title

Url

Community

Eos non et at cumque. ▾










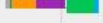


Create Submission

Lets try this out on the demo app

Lets implement HTTP caching for this page in the UI.















Client-side Caching

| Name Path | Method | Status Text | Type | Size Content | Time Latenc | Timeline | 200 ms |
|--|--------|----------------|------------------|--------------------|------------------|---|--------|
|  new /submissions | GET | 200 OK | text/html | 3.1 KB 2.3 KB | 92 ms 53 ms |  | |
|  application-f3e64a74b7ab4dff6d9d96f8038945dd.css /assets | GET | 200 OK | text/css | 651 B 837 B | 94 ms 54 ms |  | |
|  application-f221e1ec5ab975eeadbb83b9e995b0d3.js /assets | GET | 200 OK | application/j... | 39.0 KB 114 KB | 152 ... 50 ms |  | |
|  bootstrap.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css | GET | 200 OK | text/css | 24.4 KB 114 KB | 80 ms 78 ms |  | |
|  bootstrap-theme.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css | GET | 200 OK | text/css | 3.2 KB 19.5 KB | 61 ms 61 ms |  | |
|  bootstrap.min.js maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js | GET | 200 OK | text/javascript | 11.6 KB 34.6 KB | 63 ms 62 ms |  | |

Initial page load gets every resource



Client-side Caching

| Name Path | Method | Status Text | Type | Size Content | Time Latenc | Timeline | 100 ms | 150 ms | 200 ms |
|--|--------|---------------------|------------------|------------------|------------------|---|--------|--------|--------|
|  new /submissions | GET | 200 OK | text/html | 3.1 KB 2.3 KB | 106 ... 63 ms |  | | | |
|  application-f3e64a74b7ab4dff6d9d96f8038945dd.css /assets | GET | 304 Not Modified | text/css | 209 B 837 B | 60 ms 60 ms |  | | | |
|  application-f221e1ec5ab975eeadbb83b9e995b0d3.js /assets | GET | 304 Not Modified | application/j... | 223 B 114 KB | 57 ms 57 ms |  | | | |
|  bootstrap.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css | GET | 304 Not Modified | text/css | 454 B 114 KB | 28 ms 27 ms |  | | | |
|  bootstrap-theme.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css | GET | 304 Not Modified | text/css | 454 B 19.5 KB | 56 ms 55 ms |  | | | |
|  bootstrap.min.js maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js | GET | 304 Not Modified | text/javascript | 454 B 34.6 KB | 75 ms 74 ms |  | | | |

Refreshing the page doesn't re-download the assets, but it does redownload /submissions/new



Client-side Caching

Demo App

New Submission

Title

Url

Community

Eos non et at cumque.

Create Submission

When can this page
be out of date?



Client-side Caching

```
class SubmissionsController < ApplicationController
  ...

  def new
    @submission = Submission.new
  end

  ...
end
```



Client-side Caching

```
class SubmissionsController < ApplicationController
  ...

  def new
    @submission = Submission.new if stale?(Community.all)
  end

  ...
end
```



Client-side Caching

What is this actually doing?

- `if stale?(Community.all)`















Client-side Caching

What is this actually doing?

- `if stale?(Community.all)`
- This tells Rails to base the etag on all communities
- You can think of this as taking the digest of all the concatenated `updated_at` fields of all communities.
- Whenever this controller is invoked, Rails compares the etag presented in the request to the etag that would be generated
 - If they are the same, it returns a 304



Client-side Caching

| Name Path | Method | Status Text | Type | Size Content | Time Latenc | Timeline | 100 ms | 150 ms |
|--|--------|---------------------|------------------|------------------|----------------|---|---|--------|
|  new /submissions | GET | 304 Not Modified | text/html | 732 B 2.4 KB | 54 ms 53 ms |  | | |
|  application-f3e64a74b7ab4dff6d9d96f8038945dd.css /assets | GET | 304 Not Modified | text/css | 209 B 837 B | 52 ms 51 ms | |  | |
|  application-f221e1ec5ab975eeadbb83b9e995b0d3.js /assets | GET | 304 Not Modified | application/j... | 223 B 114 KB | 54 ms 53 ms | |  | |
|  bootstrap.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css | GET | 304 Not Modified | text/css | 454 B 114 KB | 23 ms 22 ms | |  | |
|  bootstrap-theme.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css | GET | 304 Not Modified | text/css | 454 B 19.5 KB | 23 ms 23 ms | |  | |
|  bootstrap.min.js maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js | GET | 304 Not Modified | text/javascript | 454 B 34.6 KB | 24 ms 23 ms | |  | |

The web console indicates we are successful. Adding a new Community causes a 200 response.



Client-side Caching

Demo App

Title: Amet aliquid facere doloribus ut.

Url: <http://bergstrom.net/rahsaan.bechtelar>

Community: Eos non et at cumque.

Comment on this submission

Comments:

Voluptas est repellendus sed. Illum nam quia. Magnam itaque nostrum sed quae asperiores eligendi.

Reply

Tempora ipsum laudantium. Eum et provident eos voluptas. Ex consequatur numquam commodi. Molestiae aspernatur laborum. Dolores voluptates provident exercitationem quo eos voluptatem fugiat.

Reply

Quam beatae qui sed aut. Amet harum architecto ratione quaerat. Fuga quia maxime quis ipsum dolor. Cum molestiae quia sed illum et.

Reply

Vel molestiae nemo ullam enim eveniet aut. Quis eos nulla eaque dolor et et. Neque voluptas saepe. Cumque ea atque numquam incidunt aut. Facilis est quia et veniam totam ipsam voluptatem.

Reply

Lets try this same technique for this part of the UI.

What can make this page stale?



Client-side Caching

```
class SubmissionsController < ApplicationController
  before_action :set_submission, only: [:show, :edit, :update, :destroy]

  ...

  def show
  end

  ...

  def set_submission
    @submission = Submission.find(params[:id])
  end
end
```



Client-side Caching

```
class SubmissionsController < ApplicationController
  before_action :set_submission, only: [:show, :edit, :update, :destroy]

  ...













  def show
    fresh_when([@submission, @submission.community, @submission.comments])
  end

  ...

  def set_submission
    @submission = Submission.find(params[:id])
  end
end
```



Client-side Caching

| Name Path | Method | Status Text | Type | Size Content | Time Latenc | Timeline | 100 ms | 150 ms | 200 ms |
|---|--------|---------------------|------------------|------------------|----------------|---|---|--------|--------|
|  1 /submissions | GET | 304 Not Modified | text/html | 732 B 9.1 KB | 64 ms 63 ms |  | | | |
|  application-f3e64a74b7ab4dff6d9d96f8038945dd.css /assets | GET | 304 Not Modified | text/css | 209 B 837 B | 56 ms 55 ms | |  | | |
|  application-f221e1ec5ab975eeadb83b9e995b0d3.js /assets | GET | 304 Not Modified | application/j... | 223 B 114 KB | 49 ms 48 ms | |  | | |
|  bootstrap.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css | GET | 304 Not Modified | text/css | 454 B 114 KB | 20 ms 19 ms | |  | | |
|  bootstrap-theme.min.css maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css | GET | 304 Not Modified | text/css | 454 B 19.5 KB | 21 ms 19 ms | |  | | |
|  bootstrap.min.js maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js | GET | 304 Not Modified | text/javascript | 454 B 34.6 KB | 21 ms 20 ms | |  | | |

The web console indicates we are successful. Adding a new Comment or modifying the community causes a 200 response.



Client-side Caching

Demo App

Submissions

| Title | Url | Community |
|--|---|--|
| Amet aliquid facere doloribus ut. | http://bergstrom.net/rahsaan.bechtelar | Eos non et at cumque. 25 comments |
| Sed quasi ut occaecati nulla provident. | http://daniel.info/nayeli.turner | Eos non et at cumque. 20 comments |
| Dolor quas non ut animi quaerat nobis voluptas sit. | http://schultz.org/savanna | Eos non et at cumque. 20 comments |
| Sit cum nulla et labore. | http://millsmarquardt.info/wallace | Eos non et at cumque. 20 comments |
| Et reiciendis necessitatibus eos quis ut enim quae qui. | http://kovacek.biz/genoveva | Eos non et at cumque. 20 comments |

How about the index
page?

What can make this
page stale?



Client-side Caching

```
class SubmissionsController < ApplicationController
  ...

  def index
    @submissions = Submission.all
  end

  ...

end
```



Client-side Caching

```
class SubmissionsController < ApplicationController
  ...

  def index
    if stale?([Submission.all, Community.all, Comments.all])
      @submissions = Submission.all
    end
  end
  ...

end
```



For Next Time...

You should be working on your first sprint's worth of stories.

Please update your git repos with a README file. Please include:

- Project name
- Project description
- For each student your name and a photo

