Project Review

Olivier Liechti
TWEB

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Yeoman, Grunt, heroku and git

Yeoman, grunt, heroku and git

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- In the project, one objective is to be able to deploy our application in the **cloud**.
- For that purpose, **heroku** offers a convenient and free (well, at least during development...). We had already tested heroku in a previous lab assignment.
- We have made the choice to use **a particular yeoman generator** to bootstrap our project. Remember the notion of **scaffolding** (échaffaudage) that we have presented in a previous lecture.
- This generator is called **generator-angular-fullstack**.
- As it is always the case with scaffolding and automation tools, **complex tasks work (almost) out of the box**. But at some point, you need to understand what is happening **behind the scenes**...

Yeoman, grunt, heroku and git

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- For instance, from the generator documentation, you know that in order to **initially setup heroku** for your application, you need to:

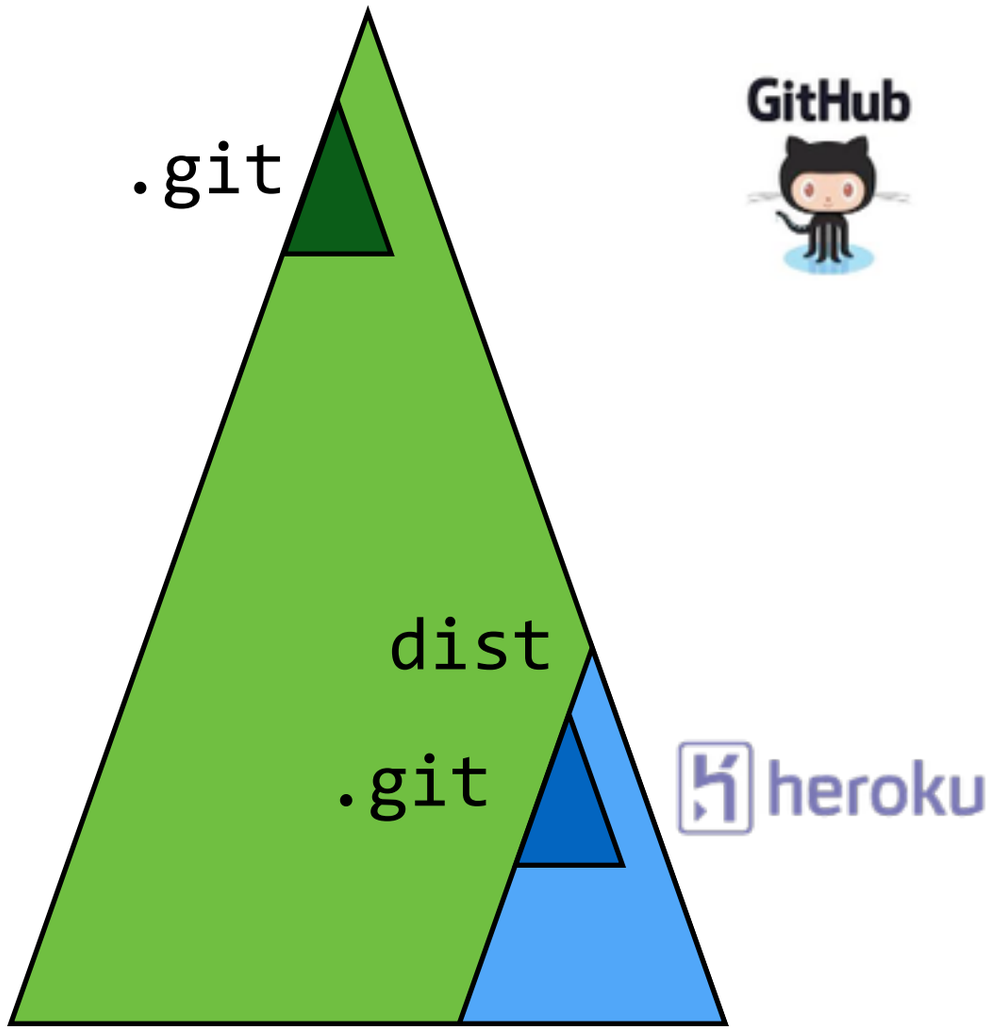
```
cd dist
yo angular-fullstack:heroku
```

- To understand what happens when you type the last command, you have to:
  - Remember the notion of "**sub-generator**" offered by yeoman. A sub-generator allows you to do scaffolding-related operations *after* a project skeleton has been generated.
  - For instance, when you add an API end-point to your project, you use a specific sub-generator. Same thing for a controller on the client side.
  - **:heroku** is a sub-generator, and you have access to its source code in the git repo of the angular fullstack. (https://github.com/DaftMonk/generator-angular-fullstack/blob/master/heroku/index.js).
  - If you look at the source code, you will see that the sub-generator is using the **heroku toolbelt** (i.e. the commands and tools that you install on your machine when you setup heroku) and is providing **wrappers** around them.

Yeoman, grunt, heroku and git

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

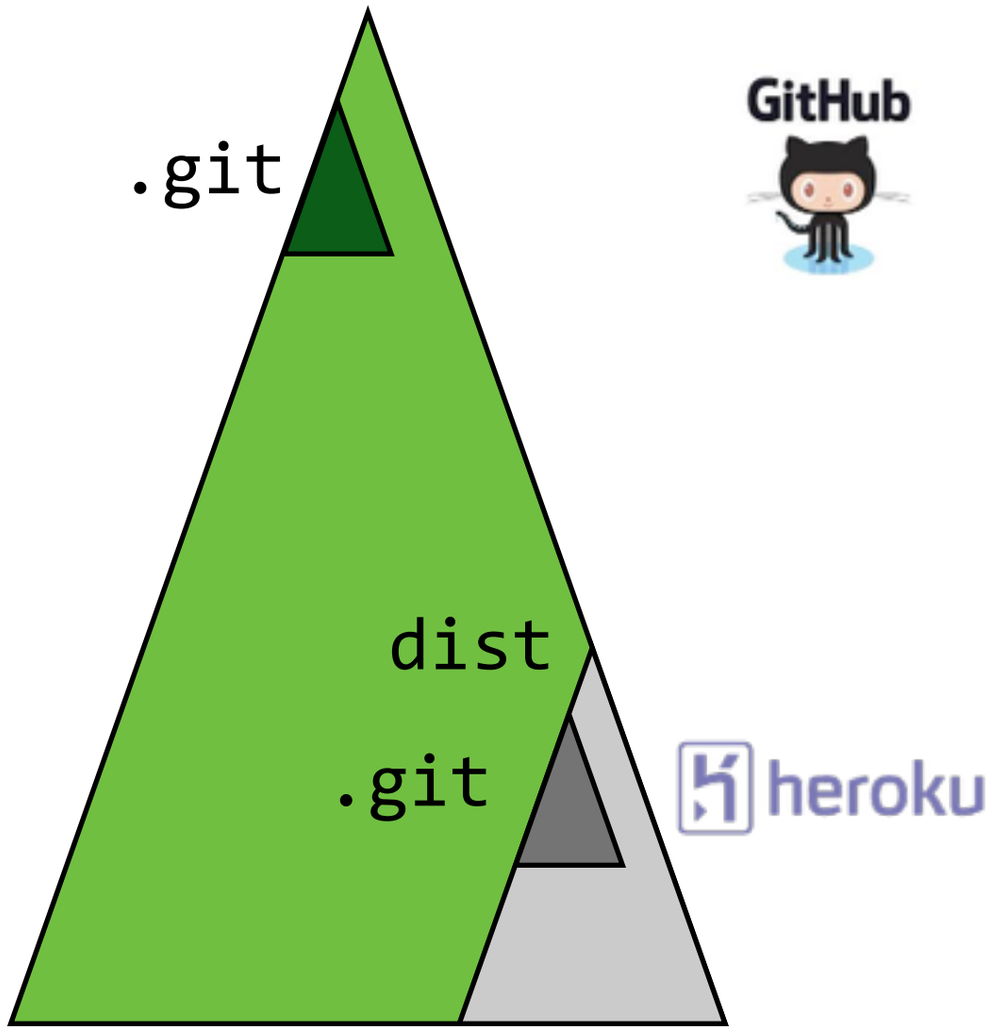
- Next, you need to understand that in our development workflow, **git** is used for **two different purposes**:
  - to **manage our source code** artifacts (sharing and versioning). We use GitHub for that purpose.
  - to **deploy our application into production**. This is a choice that heroku has made: it is their way to handle deployments.
- Because of that, you have to understand that you are dealing with **two different git clones in your project folder**.



Yeoman, grunt, heroku and git

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

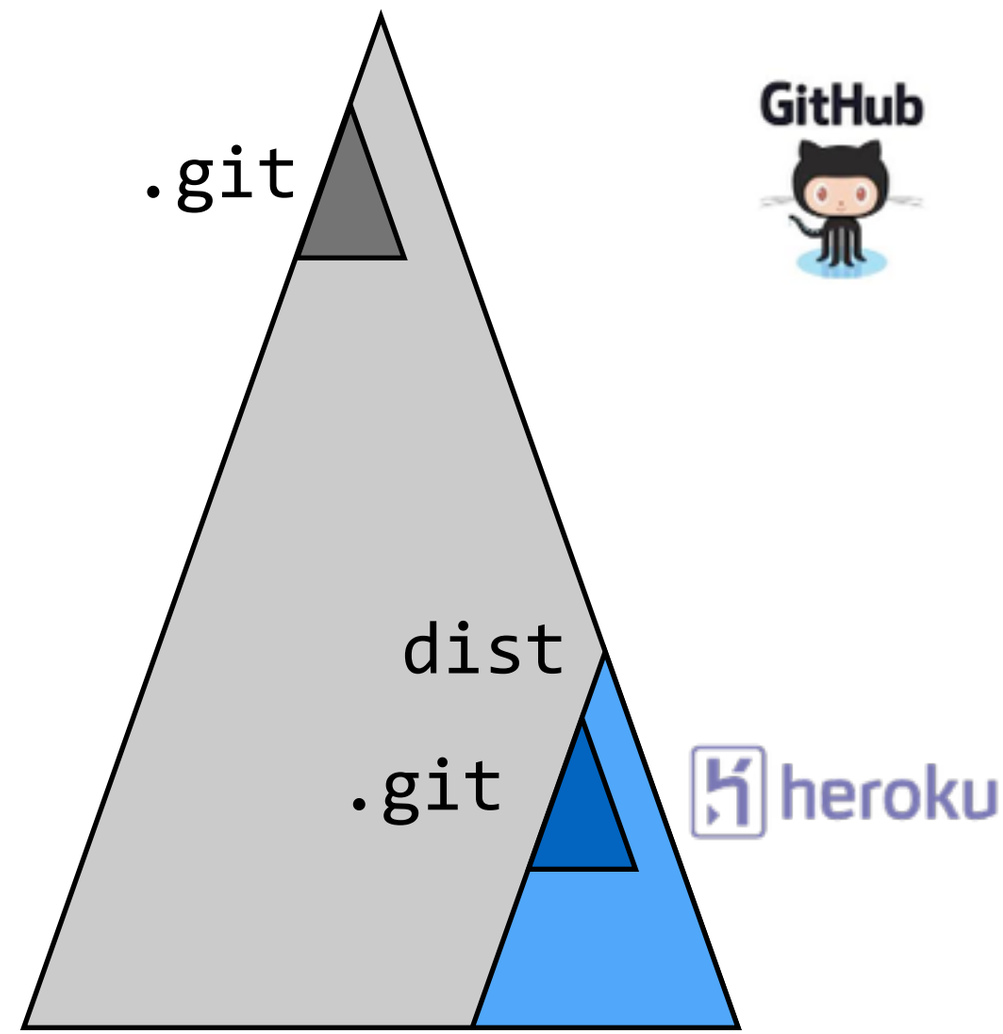
- The **first git repo** is what you have at the root of your project directory:
  - You have `.git` folder at the root and if you look at the `.git/config` file, you will find a remote definition that points to a GitHub repo.
  - This is what you use to manage your source code. Have a look at the `.gitignore` file and you will see that the `dist` directory (among others) is not versioned.
- So, when you modify your source files and want to push them to GitHub, you do a **git push origin master** from the project folder.



Yeoman, grunt, heroku and git

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- The **second git repo** is what you have at the root of `dist` directory:
  - If you look at the corresponding `.git/config` file, you will find a remote definition that points to a heroku repo.
  - This is what you use (indirectly) to deploy your application.
- When you deploy your app, you do not do directly an git push heroku master. But in fact, when you do a `grunt buildcontrol:heroku`, this is what happens.
- To understand the details, you need to look at the source code of this Grunt plugin (https://github.com/robwierzbowski/grunt-build-control).



heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Bower, wiredep and grunt-wiredep

Bower, wiredep and grunt-wiredep

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- Like npm, bower is a **package manager**.
- In other words, it is a tool that is used to **create "packages"** ("bundles", "archives") of various files, to **search** them, to **install** them, to manage **dependencies**, etc.
- Whereas npm is used mostly on the server side, **bower is used on the client side**. Files in bower packages include HTML, CSS, javascript files.
- When you install bower, you have access to **command line tools** (bower install, bower list, bower update, etc.).
- You also have access to a **JavaScript API** that you could use in your (Node.js) scripts.
- The **most common way** to use bower, however, is to use a module that integrates bower with a **build management tool** (such as **Grunt** or Gulp).

stephenplusplus / **grunt-wiredep**

uses

taptapship / **wiredep**

uses

bower

Bower, wiredep and grunt-wiredep

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- **wiredep** is a npm package that injects your bower dependencies into an HTML source file.

```
<html>
<head>
  <!-- bower:css -->
  <!-- endbower -->
</head>
<body>
  <!-- bower:js -->
  <script src="bower_components/jquery/dist/jquery.js"></script>
  <!-- endbower -->
</body>
</html>
```

automatically injected

stephenplusplus / **grunt-wiredep**

uses

taptapship / **wiredep**

uses

bower

- **grunt-wiredep** is grunt plugin that uses wiredep.

Bower, wiredep and grunt-wiredep

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- The **angular-fullstack generator** uses this feature. Have a look at the Gruntfile.js and index.html files to see how.

stephenplusplus / **grunt-wiredep**

uses

```
// Automatically inject Bower components into the app
wiredep: {
  target: {
    src: '<%= yeoman.client %>/index.html',
    ignorePath: '<%= yeoman.client %>/',
    exclude: [/bootstrap-sass-official/, /bootstrap.js/, '/json3/', '/es5-shim/
  }
},
```

taptapship / **wiredep**

e.css/]

uses

```
<!-- bower:js -->
<script src="bower_components/jquery/dist/jquery.js"></script>
<script src="bower_components/angular/angular.js"></script>
<script src="bower_components/angular-resource/angular-resource.js"></script>
<script src="bower_components/angular-cookies/angular-cookies.js"></script>
<script src="bower_components/angular-sanitize/angular-sanitize.js"></script>
<script src="bower_components/angular-bootstrap/ui-bootstrap-tpls.js"></script>
<script src="bower_components/lodash/dist/lodash.compat.js"></script>
<script src="bower_components/angular-socket-io/socket.js"></script>
<script src="bower_components/angular-ui-router/release/angular-ui-router.js"></script>
<script src="bower_components/pdfjs-dist/build/pdf.js"></script>
<script src="bower_components/pdfjs-dist/build/pdf.worker.js"></script>
<script src="bower_components/ng-file-upload/angular-file-upload.js"></script>
<script src="bower_components/ng-file-upload-shim/angular-file-upload-shim.js"></script>
<!-- endbower -->
```

bower

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Dealing with PDFs (storing, accessing, etc.)

Where/how to store PDF files?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- When testing locally, there is no issue to store the uploaded PDF files on the file system.
- When deploying on heroku, there are some aspects to consider:
  - With the free tier, you have **limited space** (no issue for the project, but be aware of large PDF files for your demo).
  - Heroku as a notion of **ephemeral file system**. You can store files on your dynos, but as soon as they are restarted, you will lose them (no long-term persistence).
  - You can use **third-party persistence solutions** (RDBMs, NoSQL, cloud storage) in conjunction with your MongoHQ setup.
- The **Amazon S3** service is a good candidate.

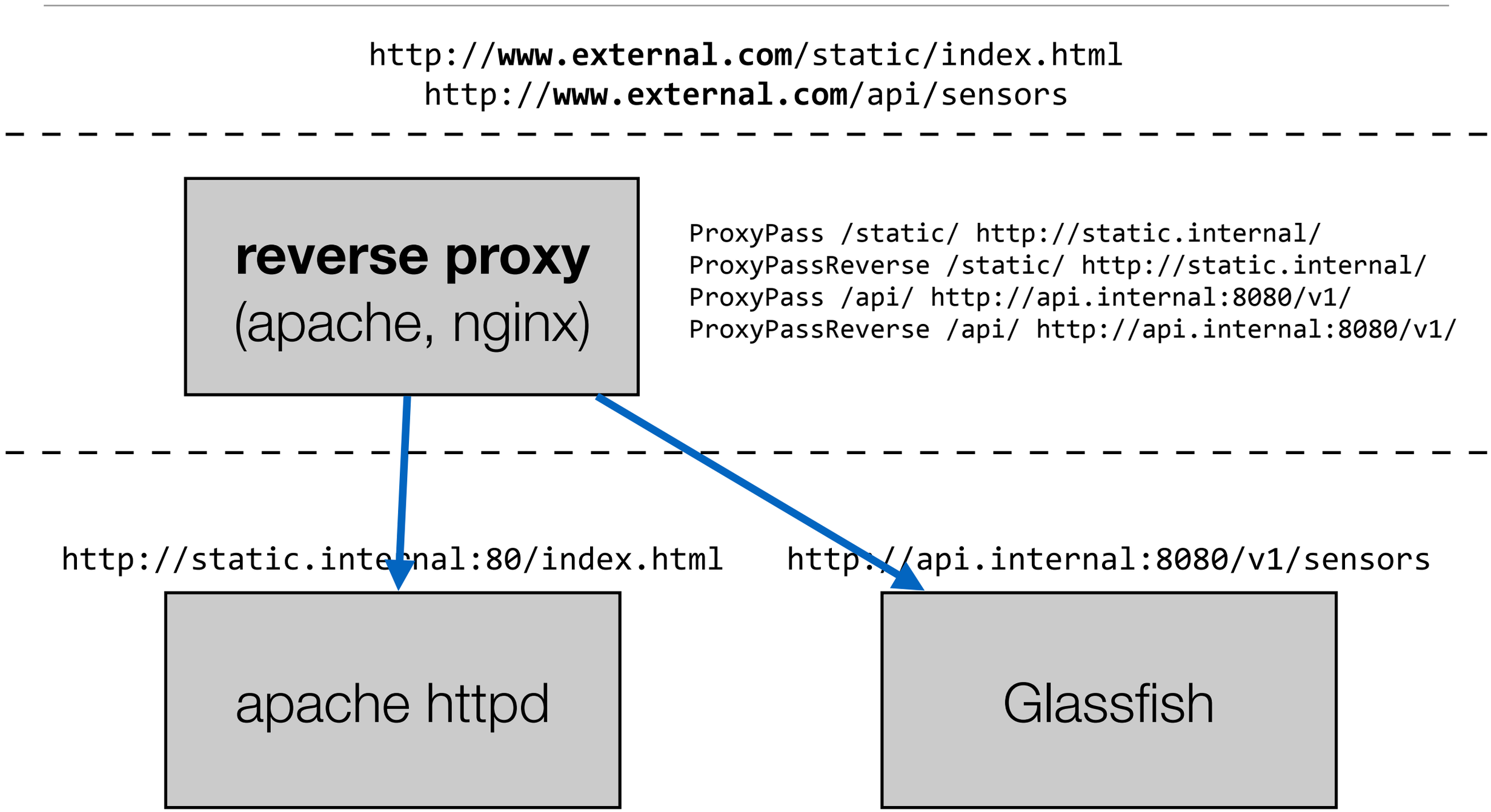
Accessing remote PDFs (cross domain)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- For security reasons, a script running in the browser is only allowed to make AJAX requests to the origin server (**same origin policy**).
- In other words, if the script is loaded from an HTML page served by `http://www.demo.com/test.html`, it will **NOT** be allowed to make AJAX requests to an URL such as `http://www.other.com/api/v1/students/`.
- Last year (in the RES course), we saw one way do deal with this issue when some documents (HTML and Javascript) are served by an **apache server** and when data is served by a **glassfish server**.
- In this solution, we used a **reverse proxy** to expose a single logical hostname to the outside world, although we had two physical hostnames.

Accessing remote PDFs (cross domain)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



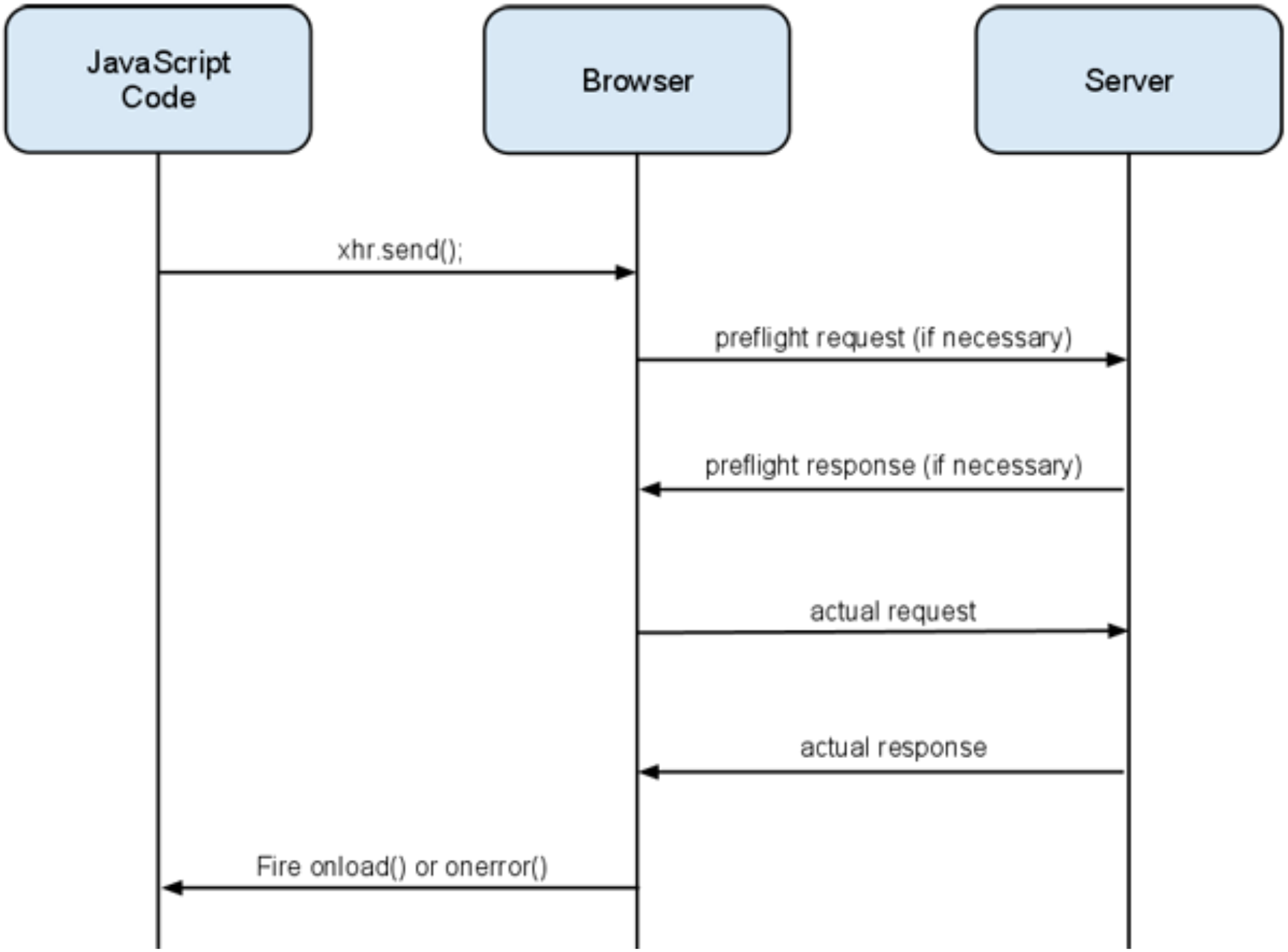
Accessing remote PDFs (cross domain)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- **Cross-Origin Resource Sharing** (CORS) provides an alternative.
- This mechanism is a recommendation from the **W3C** (http://www.w3.org/TR/cors/).
- At a high level, CORS works like this:
  - A page served by **http://www.frontend.com/index.html** executes a script. This script makes an AJAX call to **http://api.backend.com/resources/123**.
  - The developer of **api.backend.com** decides that it is legitimate for www.frontend.com to make AJAX calls towards the API. He uses CORS to express this fact and sends back special headers in the HTTP responses.
  - (In most cases), when the client-side script interacts with the API, a "**pre-flight**" HTTP request is sent in a first step to check whether a certain HTTP request will be accepted or not (depending on the HTTP method, content type, etc.).
  - The pre-flight request uses the **OPTIONS** HTTP verb. If access is authorized by the pre-flight request, then the follow-up request is sent to the server.

Accessing remote PDFs (cross domain)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



http://www.html5rocks.com/en/tutorials/cors/

Accessing remote PDFs (cross domain)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Preflight Request:

the client page comes from here

```
OPTIONS /cors HTTP/1.1
Origin: http://api.bob.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: X-Custom-Header
Host: api.alice.com
Accept-Language: en-US
Connection: keep-alive
User-Agent: Mozilla/5.0...
```

the script wants to access a resource here...

... and do an HTTP PUT

Preflight Response:

```
Access-Control-Allow-Origin: http://api.bob.com
Access-Control-Allow-Methods: GET, POST, PUT
Access-Control-Allow-Headers: X-Custom-Header
Content-Type: text/html; charset=utf-8
```

http://www.html5rocks.com/en/tutorials/cors/

this is who I authorize and how

Accessing remote PDFs (cross domain)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Actual Request:

```
PUT /cors HTTP/1.1
Origin: http://api.bob.com
Host: api.alice.com
X-Custom-Header: value
Accept-Language: en-US
Connection: keep-alive
User-Agent: Mozilla/5.0...
```

Actual Response:

```
Access-Control-Allow-Origin: http://api.bob.com
Content-Type: text/html; charset=utf-8
```

http://www.html5rocks.com/en/tutorials/cors/

Using Amazon S3 to store PDF files

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- One idea for **storing PDF files in your application** is to use the Amazon Web Services S3 service.
- It is essentially a service that **allows you to store files** and that provides a **REST API** for managing them. There is a free tier.
- If you want to use S3 with your project, you will need to:
  - setup an AWS account (there is a free tier, but you need a credit card # to register)
  - configure **CORS** (http://docs.aws.amazon.com/AmazonS3/latest/dev/cors.html)
  - decide whether you want to upload files directly from the browser to S3 (more efficient) or go via heroku (the upload would then be implemented on your server-side code).