

Deploying on Heroku

Olivier Liechti & Simon Oulevay
COMEM Web Services 2016

heig-vd

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

Heroku

heig-vd

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



What is Heroku?

Heroku is a **cloud platform** that lets companies **build, deliver, monitor and scale apps** — we're the fastest way to go from idea to URL, bypassing all those infrastructure headaches.

Heroku focuses relentlessly on **apps** and the **developer experience** around apps. Heroku lets companies of all sizes embrace the value of apps, not the distraction of hardware, nor the distraction of servers - virtual or otherwise.

Heroku and Git

GitHub repo



Heroku repo



origin

heroku

Heroku integrates with your **Git** workflow. When you register your app on Heroku, it will add a **heroku remote** to your repository.



Local repo

To **deploy** the latest version of your app, make sure your changes are committed, and **push** your changes to the heroku remote:

```
git push heroku master
```

Set up Heroku

Create a free account and install the **Heroku toolbelt**.

<https://signup.heroku.com/>

<https://toolbelt.heroku.com/>

Make sure you have an SSH key

```
$> ls -la ~/.ssh  
id_rsa  
id_rsa.pub
```

If you have those two files, then you already have an SSH key. **id_rsa** is the private key, **id_rsa.pub** is the public key.

```
$> ssh-keygen -t rsa -b 4096  
Generating public/private rsa key pair.  
Enter file in which to save the key (/Users/adam/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /Users/adam/.ssh/id_rsa.  
Your public key has been saved in /Users/adam/.ssh/id_rsa.pub.
```

If you don't have a key, generate one with **ssh-keygen**.

Make sure that the path to the **.ssh** directory is in your **user's home folder** (in this example, /Users/adam).

```
$> cd ~  
$> pwd  
/Users/adam
```

If you're not sure where your **home folder** is, use these commands.

<https://devcenter.heroku.com/articles/keys>

Add your app to Heroku

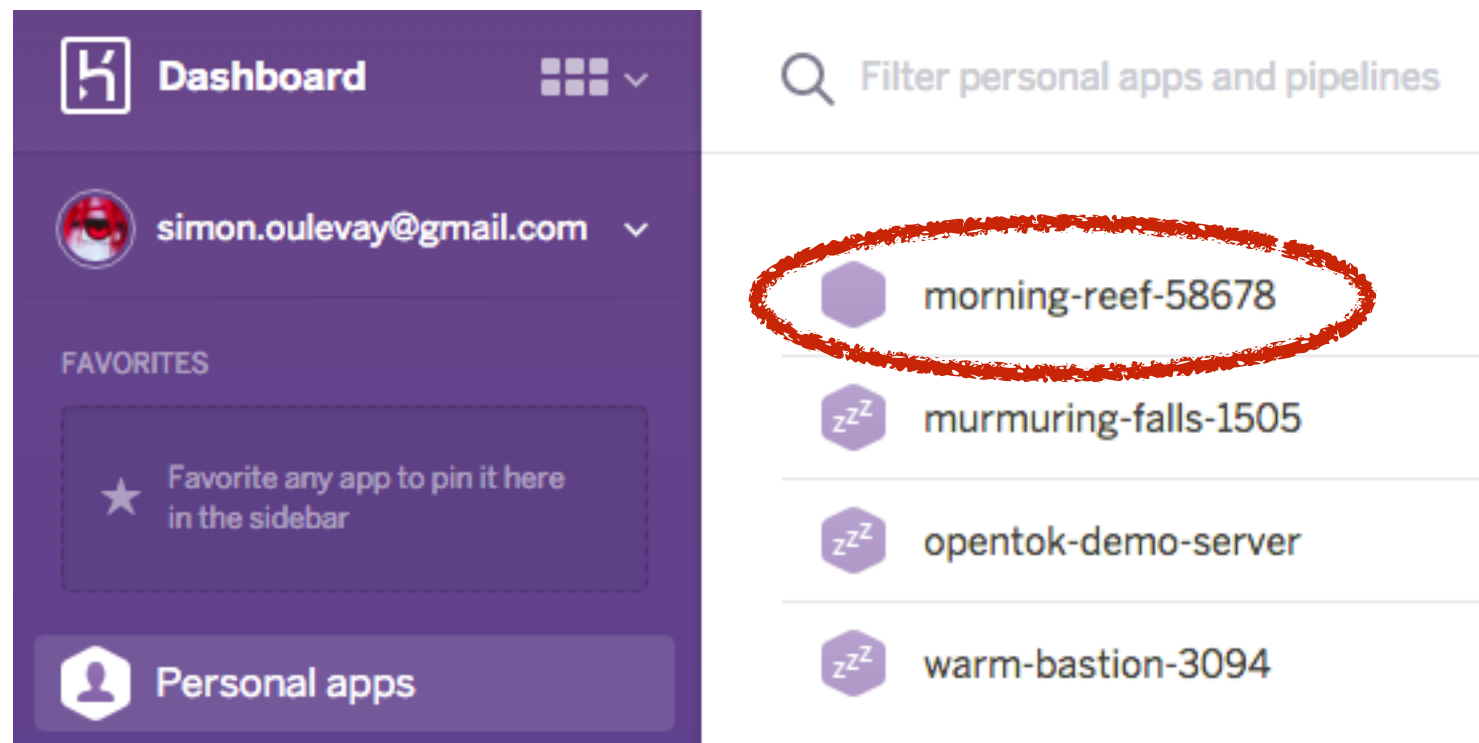
Run **heroku create** in your project's directory:

```
$> cd /path/to/repo
$> heroku create
Enter your Heroku credentials.
Email: john.doe@example.com
Password (typing will be hidden):
Logged in as john.doe@example.com
Creating app... done, stack is cedar-14
https://morning-reef-58678.herokuapp.com/ | https://git.heroku.com/morning-reef-58678.git
```

Find your application in Heroku

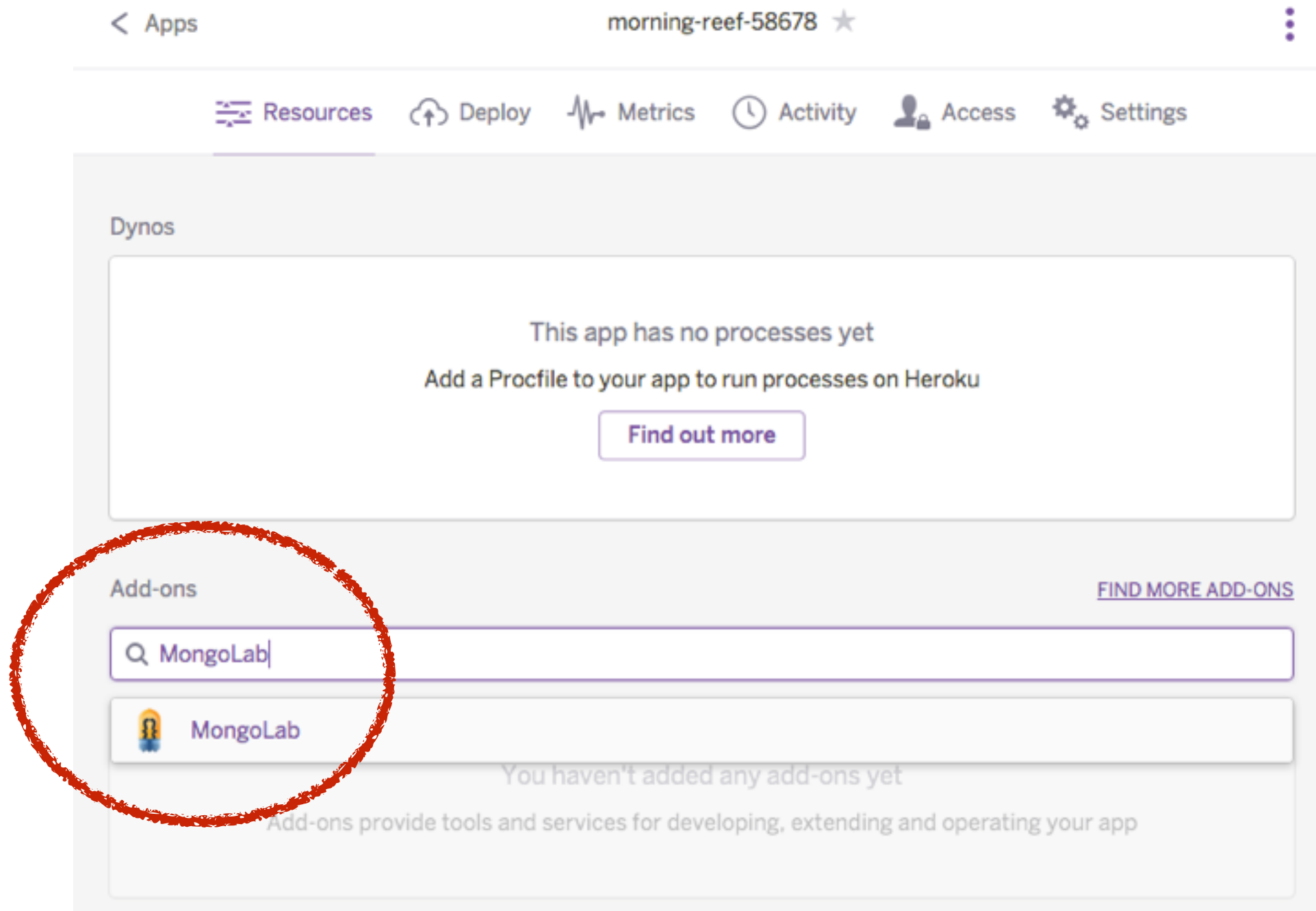
heig-vd

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



Adding MongoDB

Add the free version of the **MongoLab** plugin to your app:



When you develop your Express application on your local machine, you are in what is called a **development environment**. You want code to be automatically reloaded, full error stack traces, and speed is not a major concern.

Heroku is a **production environment**. Here you want your code and assets to be cached for speed, and you do not want to reveal sensitive information to the user.

Express environment

config/config.js

```
var path = require('path'),
    rootPath = path.normalize(__dirname + '/../..'),
    env = process.env.NODE_ENV || 'development';

var config = {
  development: {
    root: rootPath,
    app: {
      name: 'webs-2016-example'
    },
    port: 3000,
    db: 'mongodb://localhost/webs-2016-example-development'
  },
  // test environment

  production: {
    root: rootPath,
    app: {
      name: 'webs-2016-example'
    },
    port: 3000,
    db: 'mongodb://localhost/webs-2016-example-production'
  }
};

module.exports = config[env];
```

Your Express app already has what it needs to be deployed in multiple **environments**.

A different section of the **config** object is loaded depending on the environment. This allows you to use a different port and database connection string for different environments.

Which environment is applied depends on the **\$NODE_ENV** environment variable, which defaults to "development".

Express environment

config/express.js

```
if(app.get('env') === 'development'){  
  app.use(function (err, req, res, next) {  
    res.status(err.status || 500);  
    res.render('error', {  
      message: err.message,  
      error: err,  
      title: 'error'  
    });  
  });  
}  
  
app.use(function (err, req, res, next) {  
  res.status(err.status || 500);  
  res.render('error', {  
    message: err.message,  
    error: {},  
    title: 'error'  
  });  
});
```

Parts of your app will **behave differently** depending on the **environment**.

For example, error **stack traces** are shown to the user in the **development** environment, because you want to be able to debug the problem.

In the **production** environment, only the error message is shown, because you don't want to reveal sensitive information.

Update your production configuration

Heroku hosts your app on a shared server with other apps. Therefore, it will not use port 3000, but choose a random port number instead. This port number will be provided in the **\$PORT** environment variable.

Similarly, the MongoDB connection string will be provided by the MongoLab plugin in the **\$MONGODB_URI** environment variable.

Update the configuration of your **production** environment to use these **environment variables** if they are available.

config/config.js

```
production: {  
  root: rootPath,  
  app: {  
    name: 'webs-2016-example'  
  },  
  port: process.env.PORT || 3000,  
  db: process.env.MONGODB_URI || 'mongodb://localhost/webs-2016-example-production'  
}
```

This syntax:

```
var port = process.env.PORT || 3000;
```

Is equivalent to this:

```
var port;  
if (process.env.PORT) {  
    port = process.env.PORT;  
} else {  
    port = 3000;  
}
```

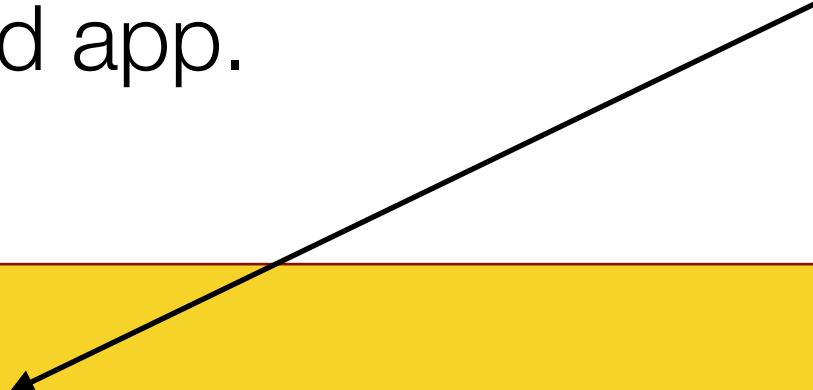
Push your app to Heroku

To deploy the latest version of your app to Heroku, simply push your **master** branch to the **heroku** remote.

```
$> git push heroku master
Counting objects: 44, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (42/42), done.
Writing objects: 100% (44/44), 6.70 KiB | 0 bytes/s, done.
Total 44 (delta 8), reused 24 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Node.js app detected
remote: -----> Creating runtime environment
remote: -----> Installing binaries
remote: -----> Restoring cache
remote: -----> Building dependencies
remote: -----> Caching build
remote: -----> Build succeeded!
remote: -----> Discovering process types
remote: -----> Compressing...
remote: -----> Launching...
remote:          Released v4
remote:          https://morning-reef-58678.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/morning-reef-58678.git
* [new branch]      master -> master
```


Check it out

When the deployment is done, Heroku gives you the link to your deployed app.



```
remote: -----> Launching...
remote:           Released v4
remote:           https://morning-reef-58678.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/morning-reef-58678.git
 * [new branch]      master -> master
```

Not working?

Use **heroku logs** to check your app's logs on Heroku.

```
$> heroku logs
2016-02-24T18:58:53.725938+00:00 heroku[api]: Deploy 35191e8 by john.doe@example.com
2016-02-24T18:58:53.725938+00:00 heroku[api]: Release v6 created by john.doe@example.com
2016-02-24T18:58:55.207645+00:00 heroku[web.1]: Starting process with command `npm start`
2016-02-24T18:58:57.914290+00:00 app[web.1]:
2016-02-24T18:58:57.914288+00:00 app[web.1]: > webs-2016-example@0.0.1 start /app
2016-02-24T18:58:57.914290+00:00 app[web.1]: > node app.js
2016-02-24T18:58:57.914267+00:00 app[web.1]:
2016-02-24T18:58:53.844899+00:00 heroku[web.1]: State changed from crashed to starting
2016-02-24T18:58:58.676213+00:00 app[web.1]: Express server listening on port 34405
2016-02-24T18:58:58.891893+00:00 heroku[web.1]: State changed from starting to up
2016-02-24T18:59:00.193676+00:00 app[web.1]: GET / 200 24.217 ms - -
2016-02-24T18:59:01.205618+00:00 app[web.1]: GET /favicon.ico 404 552.827 ms - 240
```

MongoLab database console

heig-vd

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

Add-ons [FIND MORE ADD-ONS](#)

🔍 Quickly add add-ons from Elements

	MongoLab	Sandbox	Free	⋮
--	----------	---------	------	---

Collections Users Stats Backups Tools

Collections


✖ Delete all collections + Add collection

NAME	DOCUMENTS	CAPPED?	SIZE ⓘ	
people	1	false	8.09 KB	✕



MongoLab database console


heig-vd

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


Collection: people 


Documents Indexes Stats Tools

Documents  Delete all documents in collection  Add document


--- Start new search --- 



All Documents

Display mode: ☒ list ☐ table ([edit table view](#)) 

records / page 10  [1 - 1 of 1]

```
{
  "_id": {
    "$oid": "56cc9597528648110002ddb0"
  },
  "age": 24,
  "name": {
    "first": "John"
  }
}
```

records / page 10  [1 - 1 of 1]

Heroku sleeps

Apps using the **free dyno type** have a sleep and a recharge behavior. Free dynos will **sleep** when a web dyno receives **no web traffic** for a period of time. In addition, if a free dyno exceeds a quota of 18 hours of activity during a 24 hour window, it will be forced to recharge.

<https://devcenter.heroku.com/articles/dyno-sleeping>

In other words: if you haven't used your Heroku app for a while, it's normal if it takes a few seconds to respond the first time.



Deploy your documentation

Single-page documentation

If your documentation is a **single HTML page** without separate assets (CSS/JS files), like **grunt-raml2html** produces, generate it into the **public** directory, and update the **home** controller to send it.

Where to save the generated documentation

Location of your RAML file(s)

Gruntfile.js

```
raml2html: {  
  all: {  
    files: {  
      'public/api.html': ['api.raml']  
    }  
  }  
},
```

app/controllers/home.js

```
router.get('/', function (req, res, next) {  
  res.sendFile('api.html', {  
    root: 'public'  
  });  
});
```

Documentation folder

If your documentation is a **directory with multiple files**, like **grunt-apidoc** produces, generate it into the **public** directory, and update the **home** controller to **redirect** to it.

Location of your API controllers

Where to save the generated documentation

Gruntfile.js

```
apidoc: {  
  all: {  
    src: "app/",  
    dest: "public/apidoc/"  
  },  
},
```

app/controllers/home.js

```
router.get('/', function (req, res, next) {  
  res.redirect('/apidoc');  
});
```