Transcoding videos in Ruby: A story

Michał Matyas

@nerdblogpl

https://nerdblog.pl

UNTITLED KINGDOM

twitter: @untitledknd

https://untitledkingdom.com/jobs

Every piece of code presented here may not work properly or at all (sorry)

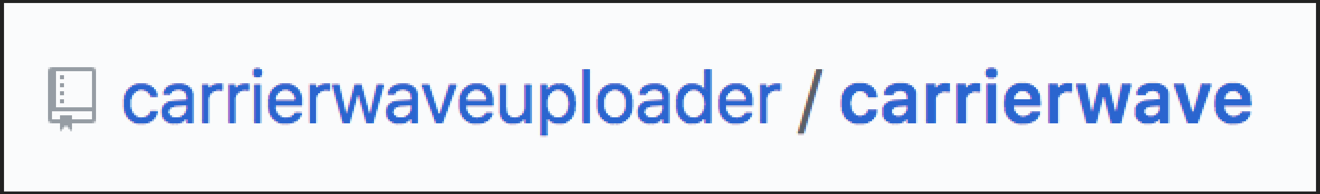
Introduction

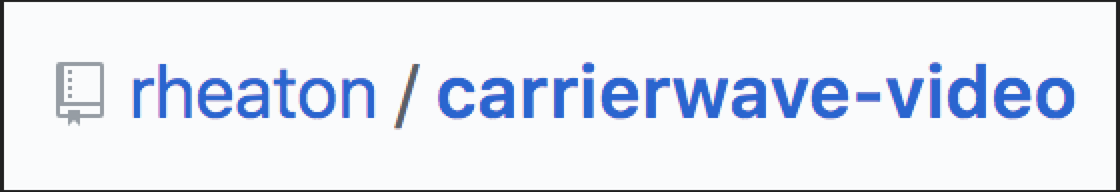
There was once a project...

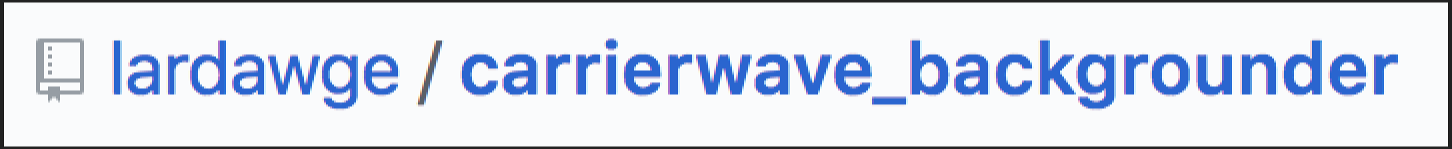
carrierwaveuploader / carrierwave

carrierwaveuploader / **carrierwave**

rheaton / **carrierwave-video**


carrierwaveuploader / **carrierwave**


rheaton / **carrierwave-video**


lardawge / **carrierwave_backgrounder**

Lesson #1
Hindsight 20/20

It's easy to look back and think "this should've been obvious" but you usually lack all the knowledge and experience at the time.

```
class Video < ActiveRecord::Base
class Video < ActiveRecord::Base

before_save                                    mount_uploader VideoUploader
```

```
class Video < ActiveRecord::Base

before_save                        mount_uploader VideoUploader

                                   process_in_background UploadWorker
```

```
class Video < ActiveRecord::Base

before_save           mount_uploader VideoUploader

                      process_in_background UploadWorker

before_processing     Video.status = processing
```

```
class Video < ActiveRecord::Base

before_save                 mount_uploader VideoUploader

                            process_in_background UploadWorker

before_processing           Video.status = processing

                            carrierwave-video (ffmpeg)
```

```
class Video < ActiveRecord::Base

before_save                  mount_uploader VideoUploader

                             process_in_background UploadWorker

before_processing            Video.status = processing

                             carrierwave-video (ffmpeg)

after_processing_success     Video.status = ready
```

class Video < ActiveRecord::Base

before_save	mount_uploader VideoUploader
	process_in_background UploadWorker
before_processing	Video.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Video.status = ready
after_processing_failure	Video.status = failed

CALLBACK ALL THINGS!!!

CALLBACK ALL THINGS!!!

(please don't)

Lesson #2
Avoid callbacks

Callbacks make your code harder to reason and harder to isolate, slowly turning everything into a tightly coupled co-dependent mess.

Carrierwave versions

```
include CarrierWave::Video
include ::CarrierWave::Backgrounder::Delay

storage :file

version :mp4 do
  process :encode_video => [:mp4]
end

version :webm do
  process :encode_video => [:webm]
end
```

Carrierwave versions

```
support_format :mp4_1080p, {
  resolution: '1920x1080',
  progress: :on_progress_1080p,
  if: :allow_1080p?
}

support_format :mp4_720p, {
  resolution: '1920x1080',
  progress: :on_progress_720p
}

support_format :mp4_480p, {
  resolution: '852x480',
  custom: '-preset ultrafast -g 5',
  progress: :on_progress_480p
}
```

Lesson #3

Don't use Carrierwave versions

Or at least don't use them for anything more complex. They are good for other things (probably).

Consider using Shrine instead.

Video

Video

Version	Version	Version
kind: "720p"	kind: "480p"	kind: nil

Video

Version
kind: "720p"

Version
kind: "480p"

Version
kind: nil

Version
kind: "720p"
special: true

Version
kind: "480p"
special: true

Version
kind: nil
special: true

class Video < ActiveRecord::Base

before_save	mount_uploader VideoUploader
	process_in_background UploadWorker
before_processing	Video.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Video.status = ready
after_processing_failure	Video.status = failed

```
class Video < ActiveRecord::Base
Video.status = processing
```

class Video < ActiveRecord::Base

Video.status = processing

class Version < ActiveRecord::Base

before_save	mount_uploader VersionUploader
	process_in_background VersionUploadWorker
before_processing	Version.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Version.status = ready
after_processing_failure	Version.status = failed

class Video < ActiveRecord::Base

Video.status = processing

class Version < ActiveRecord::Base

before_save	mount_uploader VersionUploader
	process_in_background VersionUploadWorker
before_processing	Version.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Version.status = ready
after_processing_failure	Version.status = failed

class Version < ActiveRecord::Base

before_save	mount_uploader VersionUploader
	process_in_background VersionUploadWorker
before_processing	Version.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Version.status = ready
after_processing_failure	Version.status = failed

class Video < ActiveRecord::Base

Video.status = processing

class Version < ActiveRecord::Base

before_save	mount_uploader VersionUploader
	process_in_background VersionUploadWorker
before_processing	Version.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Version.status = ready
after_processing_failure	Version.status = failed

class Version < ActiveRecord::Base

before_save	mount_uploader VersionUploader
	process_in_background VersionUploadWorker
before_processing	Version.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Version.status = ready
after_processing_failure	Version.status = failed

class Version < ActiveRecord::Base

before_save	mount_uploader VersionUploader
	process_in_background VersionUploadWorker
before_processing	Version.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Version.status = ready
after_processing_failure	Version.status = failed

class Video < ActiveRecord::Base

Video.status = processing

class Version < ActiveRecord::Base

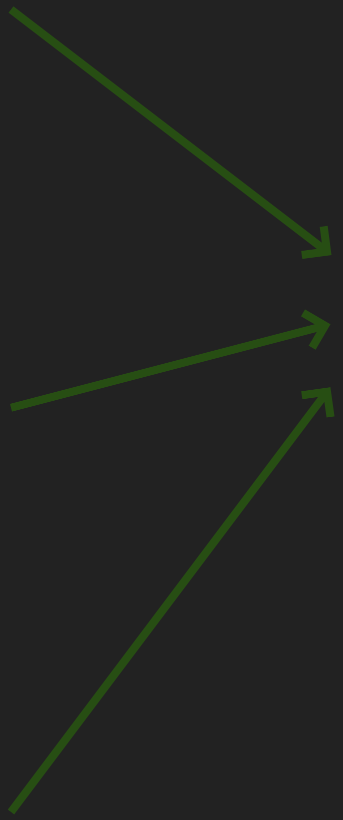
before_save	mount_uploader VersionUploader
	process_in_background VersionUploadWorker
before_processing	Version.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Version.status = ready
after_processing_failure	Version.status = failed

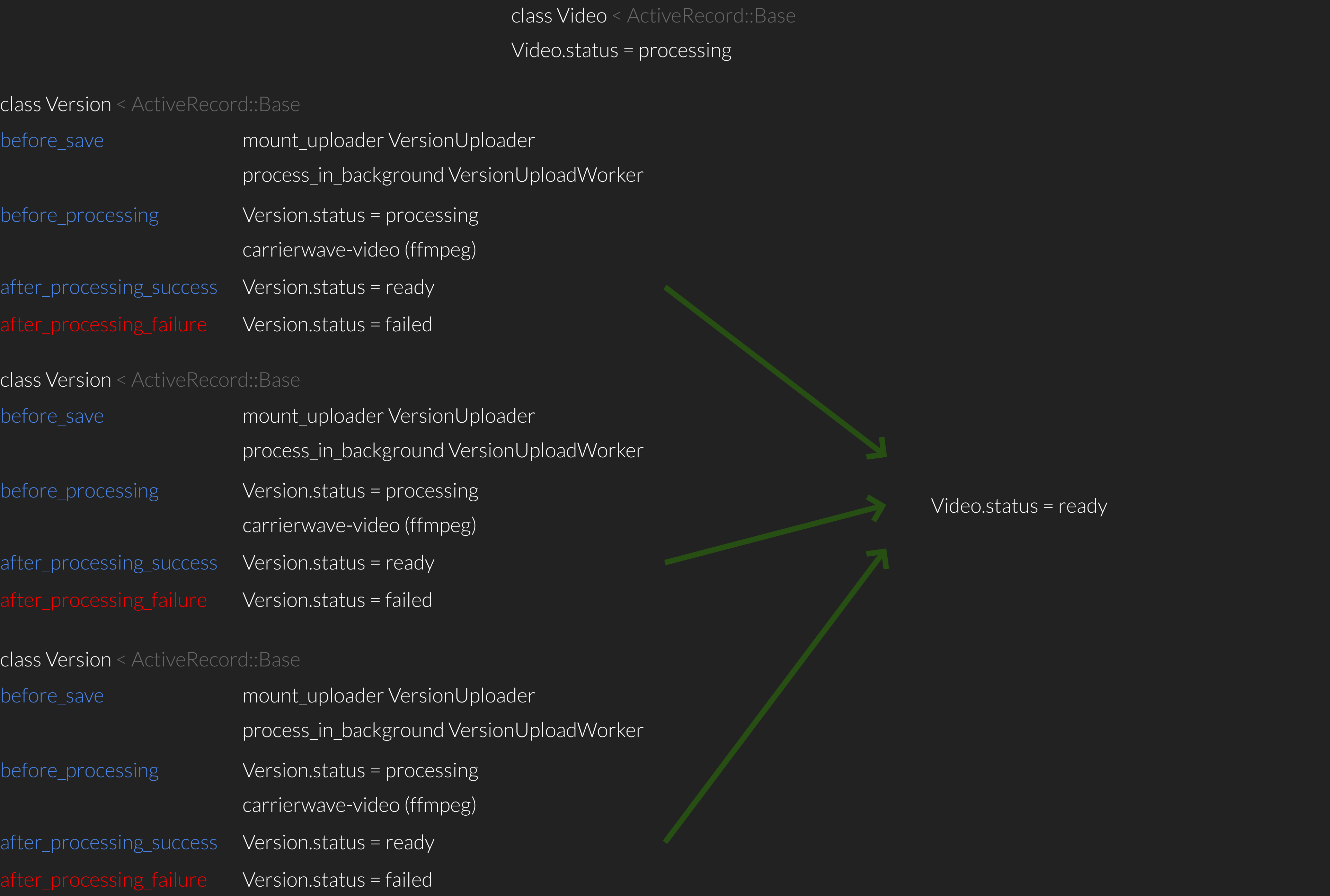
class Version < ActiveRecord::Base

before_save	mount_uploader VersionUploader
	process_in_background VersionUploadWorker
before_processing	Version.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Version.status = ready
after_processing_failure	Version.status = failed

class Version < ActiveRecord::Base

before_save	mount_uploader VersionUploader
	process_in_background VersionUploadWorker
before_processing	Version.status = processing
	carrierwave-video (ffmpeg)
after_processing_success	Version.status = ready
after_processing_failure	Version.status = failed





```
class Video < ActiveRecord::Base
Video.status = processing

class Version < ActiveRecord::Base
before_save                  mount_uploader VersionUploader
                             process_in_background VersionUploadWorker
before_processing            Version.status = processing
                             carrierwave-video (ffmpeg)
after_processing_success     Version.status = ready
after_processing_failure     Version.status = failed

class Version < ActiveRecord::Base
before_save                  mount_uploader VersionUploader
                             process_in_background VersionUploadWorker
before_processing            Version.status = processing
                             carrierwave-video (ffmpeg)
after_processing_success     Version.status = ready
after_processing_failure     Version.status = failed

class Version < ActiveRecord::Base
before_save                  mount_uploader VersionUploader
                             process_in_background VersionUploadWorker
before_processing            Version.status = processing
                             carrierwave-video (ffmpeg)
after_processing_success     Version.status = ready
after_processing_failure     Version.status = failed
```

THIS IS NOT WHAT I WAS PROMISED!

Video

Video

original version gets uploaded

Video original version gets uploaded store in local cache

Video original version gets uploaded store in local cache

ProcessingWorker

Video	original version gets uploaded	store in local cache
ProcessingWorker		checks for existing versions

Video original version gets uploaded store in local cache

ProcessingWorker checks for existing versions

creates missing ones

Video original version gets uploaded store in local cache

ProcessingWorker checks for existing versions

creates missing ones

starts processing based on the priority

(low quality video - higher priority because it's gonna be ready fastest)

Video original version gets uploaded store in local cache

ProcessingWorker checks for existing versions

creates missing ones

starts processing based on the priority

(low quality video - higher priority because it's gonna be ready fastest)

processing uses streamio-ffmpeg directly to pass custom arguments to ffmpeg based on the video format and allows transcoding from other versions rather than original file (much much faster)

Video original version gets uploaded store in local cache

ProcessingWorker checks for existing versions

creates missing ones

starts processing based on the priority

(low quality video - higher priority because it's gonna be ready fastest)

processing uses streamio-ffmpeg directly to pass custom arguments to ffmpeg based on the video format and allows transcoding from other versions rather than original file (much much faster)

uploads to S3

(in a separate worker once the version is ready)

Lesson #4

Simplest solutions are often best solutions

We would have avoided a lot of pain if we didn't try to do things The Rails Way™ but went with the simplest solution instead.

You live and learn!

But why didn't you use...

But why didn't you use...

- AWS Lambda - 15 minute of max execution time

But why didn't you use...

- AWS Lambda - 15 minute of max execution time
- Zencoder, Amazon Elastic Transcoder etc. - expensive

But why didn't you use...

- AWS Lambda - 15 minute of max execution time
- Zencoder, Amazon Elastic Transcoder etc. - expensive
- Docker - not that popular (and well supported in production) at the time

Tricks and trivia we've learned about processing video (and ffmpeg in general)

Tip #1
Copy streams!

When transcoding things, the best approach is to copy streams and match codecs as much as possible because it's way way faster.

Tip #2
Bring Your Own Arguments

When using streamio-ffmpeg just ignore all the DSL they have and go for direct passing of as many arguments as you can.

Tip #3
Optimize for streaming

ffmpeg has arguments that make video better at streaming.
The most important:
-movflags +faststart

Tip #4
There are no universal solutions

When transcoding videos from network, sometimes it's faster to download-then-process, sometimes to process on the fly. It heavily depends on the codec though.

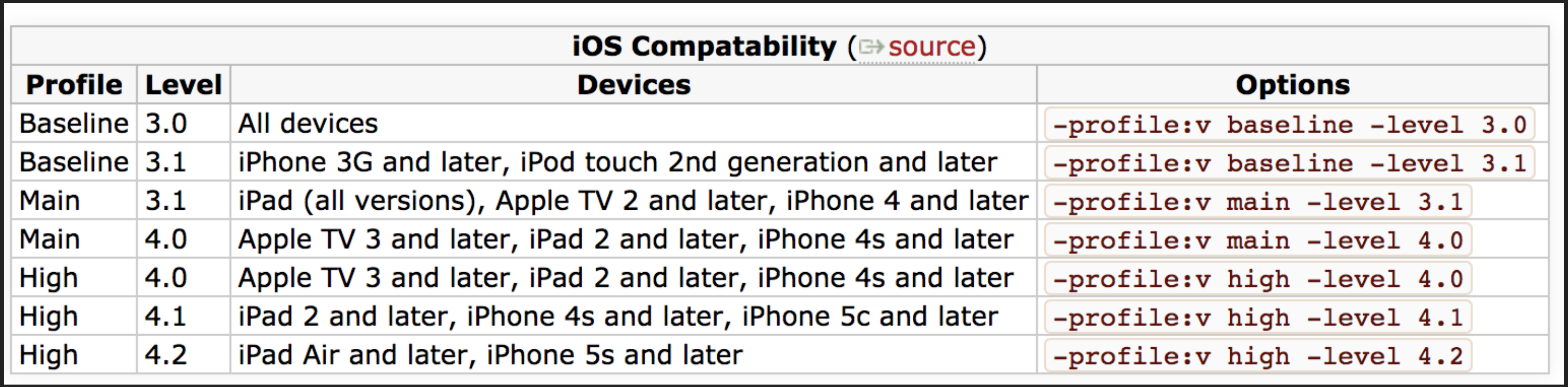
Tip #5
Use presets

ffmpeg already has presets for most common configuration options but it's a balance between speed and quality. Use fast presets for low resolution videos.

Tip #6
Use profiles

H.264 has profiles but not every device will support all of them.

Go with the highest one you can afford.



iOS Compatability (source)

Profile	Level	Devices	Options
Baseline	3.0	All devices	`-profile:v baseline -level 3.0`
Baseline	3.1	iPhone 3G and later, iPod touch 2nd generation and later	`-profile:v baseline -level 3.1`
Main	3.1	iPad (all versions), Apple TV 2 and later, iPhone 4 and later	`-profile:v main -level 3.1`
Main	4.0	Apple TV 3 and later, iPad 2 and later, iPhone 4s and later	`-profile:v main -level 4.0`
High	4.0	Apple TV 3 and later, iPad 2 and later, iPhone 4s and later	`-profile:v high -level 4.0`
High	4.1	iPad 2 and later, iPhone 4s and later, iPhone 5c and later	`-profile:v high -level 4.1`
High	4.2	iPad Air and later, iPhone 5s and later	`-profile:v high -level 4.2`

Tip #7
Always convert to YUV420

There are different pixel formats that define how color information is stored. Use -pix_fmt yuv420 for max compatibility in browsers.

Tip #8
Trust but verify

Sometimes file can be cut during transcoding and still technically be "valid", even if incomplete.

BUT...

Tip #9
Trust but verify manually

Be super-careful about invalidating the video or audio based on the duration given from ffprobe. It's sometimes approximated without a warning.

That's it folks!

UNTITLED KINGDOM

That's it folks!

UNTITLED KINGDOM

(yes, we're hiring)

That's it folks!

UNTITLED KINGDOM

(yes, we're hiring)
(no, we're not using callbacks anymore)