How to solve error “Missing `secret_key_base` for 'production' environment” on Heroku (Rails 4.1)

I've created a rails app (rails 4.1) from scratch and I am facing a strange problem that I am not able to solve.

Every time I try to deploy my app on Heroku I get an error 500:

> Missing `secret_key_base` for 'production' environment, set this value in `config/secrets.yml`

The secret.yml file contains the following configuration:

```
secret_key_base: <%= ENV["SECRET_KEY_BASE"] %>
```

On Heroku I have configured an environment variable "SECRET_KEY_BASE" with the result of "rake secret" command. If I launch "heroku config", I can see the variable with the correct name and value.

Why am I still getting this error?

Thanks a lot

ruby-on-rails ruby heroku ruby-on-rails-4

asked Apr 20 '14 at 9:38

Paolo Laurenti
323 1 3 9

1 I'm having the exact same problem and would love to know why this is happening, too. If I figure out why, I'll post back with my solution. – danielricecodes Apr 21 '14 at 20:52

Is your config file called `secret.yml` or `secrets.yml` ? – James Apr 21 '14 at 21:53

It's secrets.yml – Paolo Laurenti Apr 22 '14 at 9:26

2 I configured again the .gitignore file with the one generated by rails and now everything works fine – Paolo Laurenti Apr 22 '14 at 9:27

10 Answers

Add `config/secrets.yml` to version control and deploy again. You might need to remove a line from `.gitignore` so that you can commit the file.

I had this exact same issue and it just turned out that the boilerplate `.gitignore` Github created for my Rails application included `config/secrets.yml` .

edited Jun 5 '14 at 14:49

answered Apr 21 '14 at 21:04

danielricecodes
1,347 6 11

should this be the new default for the gitignore? I just figure the Current version should be assumed and previous versions' adjustments described in the gitignore. – max Jun 9 '14 at 23:52

53 config/secrets.yml should NEVER be in the repo you can do.yml.sample and fill it in with fake data but for security, never do .yml in repos – user3379926 Jul 7 '14 at 19:20

5 @user3379926, within the context of a Rails app on Heroku, you can't pick and choose which files are included in version control and which are not. Rails 4.1 expects the secret configuration to exist otherwise the application will not run. If you have a way to resolve the issue posed in the Question above without resorting to committing the secrets.yml file in Git, please help improve this thread by providing that advice. – danielricecodes Jul 8 '14 at 14:51

1 Try using an ENV – quantumpotato Sep 10 '14 at 1:59

7 @danielricecodes you can manually set the value in an initializer. Something like `Rails.application.config.secret_key_base = ENV["SECRET_KEY_BASE"]` would work and remove the error without adding `secrets.yml` to source. – joshhepworth Dec 11 '14 at 21:44

I had the same problem and I solved it by creating an environment variable to be loaded every time that I logged in to the production server and made a mini guide of the steps to configure it:

https://gist.github.com/pablosalgadom/4d75f30517edc6230a67

I was using Rails 4.1 with Unicorn v4.8.2, when I tried to deploy my app it didn't start properly and in the unicorn.log file I found this error message:

"app error: Missing `secret_key_base` for 'production' environment, set this value in `config/secrets.yml` (RuntimeError)"

After some research I found out that Rails 4.1 changed the way to manage the secret_key, so if you read the secrets.yml file located at [exampleRailsProject]/config/secrets.yml you'll find something like this:

```
# Do not keep production secrets in the repository,
# instead read values from the environment.
production:
  secret_key_base: <%= ENV["SECRET_KEY_BASE"] %>
```

This means that rails recommends you to use an environment variable for the secret_key_base in your production server, in order to solve this error you should follow this steps to create an environment variable for Linux (in my case Ubuntu) in your production server:

1.- In the terminal of your production server execute the next command:

```
$ RAILS_ENV=production rake secret
```

This returns a large string with letters and numbers, copy that (we will refer to that code as GENERATED_CODE).

2.1- Login as root user to your server, find this file and edit it: $ vi /etc/profile

Go to the bottom of the file ("SHIFT + G" for capital G in VI)

Write your environment variable with the GENERATED_CODE (Press "i" key to write in VI), be sure to be in a new line at the end of the file:

```
export SECRET_KEY_BASE=GENERATED_CODE
```

Save the changes and close the file (we push "ESC" key and then write ":x" and "ENTER" key for save and exit in VI)

2.2 But if you login as normal user, lets call it example_user for this gist, you will need to find one of this other files:

```
$ vi ~/.bash_profile
$ vi ~/.bash_login
$ vi ~/.profile
```

These files are in order of importance, that means that if you have the first file, then you wouldn't need to write in the others. So if you found this 2 files in your directory "~/.bash_profile" and "~/.profile" you only will have to write in the first one "~/.bash_profile", because Linux will read only this one and the other will be ignored.

Then we go to the bottom of the file ("SHIFT + G" for capital G in VI)

And we will write our environment variable with our GENERATED_CODE (Press "i" key to write in VI), be sure to be in a new line at the end of the file:

```
export SECRET_KEY_BASE=GENERATED_CODE
```

Having written the code, save the changes and close the file (we push "ESC" key and then write ":x" and "ENTER" key for save and exit in VI)

3.- You can verify that our environment variable is properly set in Linux with this command:

```
$ printenv | grep SECRET_KEY_BASE
```

or with:

```
$ echo $SECRET_KEY_BASE
```

When you execute this command, if everything went ok, it will show you the GENERATED_CODE from before. Finally with all the configuration done you should be able to deploy without problems your Rails app with Unicorn or other.

When you close your shell terminal and login again to the production server you will have this environment variable set and ready to use it.

And thats it!! I hope this mini guide help you to solve this error.

Disclaimer: I'm not a Linux or Rails guru, so if you find something wrong or any error I will be glad to fix it!

edited Apr 29 at 19:17

ersamy
347 2 17

answered Oct 3 '14 at 2:18

Demi Magus
1,370 5 19

It seems, that Rails does't sees environment variable SECRET_KEY_BASE. printenv shows it, rails c production also displays it, if I inspect ENV. But, i has no effect, when I restarting Unicorn. The only way, that works now, is pasting it directly to secrets.yml – AntonAL Nov 17 '14 at 20:37

1 This worked for me. Thank you for your full explanation. I just learned that there are gem that exist for managing an app's environment variables. 'Dotenv' is one and 'foreman' for heroku. While it was education to fix the error manually this way, maybe using one of those gems will streamline the process? – ninja08 Apr 1 at 22:52

I'm glad that my answer was helpful, thanks for the gem options @ninja08, they definitively make the process easier, mainly for those who use capistrano or other incremental tool to manage the server :) – Demi Magus Apr 2 at 5:23

Following Demi Magus's excellent instructions, I did something like this: cd /var/www/rails; rvm use ext-rbx-2.5.2@rails; SKB_FILE=/var/www/.secret_key_base; echo "export SECRET_KEY_BASE=$(RAILS_ENV=production rake secret)" > $SKB_FILE; . $SKB_FILE; echo ". $SKB_FILE" | tee -a ~/.bashrc ~/.bash_profile; chmod o-rwx $SKB_FILE; – David Winiecki Apr 13 at 16:44

Wow, this is superb answer. Nice Work! – fruqi May 23 at 7:45

I'm going to assume that you do not have your `secrets.yml` checked into source control (ie. it's in the `.gitignore` file). Even if this isn't your situation, it's what many other people viewing this question have done because they have their code exposed on Github and don't want their secret key floating around.

If it's not in source control, Heroku doesn't know about it. So Rails is looking for `Rails.application.secrets.secret_key_base` and it hasn't been set because Rails sets it by checking the `secrets.yml` file which doesn't exist. The simple workaround is to go into your `config/environments/production.rb` file and add the following line:

```
Rails.application.configure do
    ...
    config.secret_key_base = ENV["SECRET_KEY_BASE"]
    ...
end
```

This tells your application to set the secret key using the environment variable instead of looking for it in `secrets.yml`. It would have saved me a lot of time to know this up front.

answered Oct 24 '14 at 4:58

Erik Trautman
1,227 1 9 20

7 This is the best answer. `Figaro` and `heroku_secrets` don't do anything unless Rails knows that `SECRET_KEY_BASE` lives in `ENV`. I've been struggling with this thinking that if the config var existed on Heroku, Rails would pick it up just by virtue of it existing, but now it seems blindingly obvious that Rails would need to know where to look. I've been wondering how I can have code on Github without having to worry about the secret key base thing; now I know. – flanger001 Jan 17 at 21:12

1 Agreed, I think the secrets.yml is superfluous with great gems like Figaro. – Joe May 2 at 19:11

I added my secrets.yml file to .gitignore and used the heroku_secrets gem as recommended by this post.

Running the following steps worked for me:

1. $ `heroku config`
2. copy value from SECRET_KEY_BASE
3. paste value to secrets.yml file in place of <%= ENV["SECRET_KEY_BASE"] %>
4. re-deploy

edited Jun 25 '14 at 21:46

answered May 26 '14 at 4:19

Ameet Wadhwani

391 2 5

1 where SECRET_KEY_BASE can be found? – Gediminas Jun 13 '14 at 8:46

1 When you run 'heroku config' it will generate a list of "Config Vars", one of which is the SECRET_KEY_BASE variable that you're looking for. – Ameet Wadhwani Jun 25 '14 at 21:47

2 This is a private repo solution only. This technically will work, but it will expose your secret key to the public if your repo is public. – ahnbizcad Jul 29 '14 at 0:17

2 this makes no sense; how come you got 4 upvotes? – Tommaso Barbugli Oct 27 '14 at 10:02

This worked for me. ssh into your production server and cd into your current directory. run **bundle exec rake secret** or **rake secret**. You will get a long string as an output. Copy that string.

Now run **sudo nano /etc/environment**.

Paste at the bottom of the file

export SECRET_KEY_BASE=**rake secret**

ruby -e 'p ENV["SECRET_KEY_BASE"]'

where **rake secret** is the string you just copied. Paste that copied string in place of **rake secret**

restart the server and test by running echo $SECRET_KEY_BASE.

answered Jan 27 at 12:22

sumitsv21

59 4

What I did : On my production server, I create a config file (confthin.yml) for Thin (I'm using it) and add the following information :

```
environment: production
user: www-data
group: www-data
SECRET_KEY_BASE: mysecretkeyproduction
```

I then launch the app with

```
thin start -C /whereeveristhefieonprod/configthin.yml
```

Work like a charm and then no need to have the secret key on version control

Hope it could help, but I'm sure the same thing could be done with Unicorn and others.

answered May 13 '14 at 15:25

Geraud Puechaldou

314 1 8

can you explain why/how this is working? The question was for heroku. Is thin an alternative, or is it compatible with heroku? – ahnbizcad Jul 29 '14 at 0:23

While you can use initializers like the other answers, the conventional Rails 4.1+ way is to use the `config/secrets.yml` . The reason for the Rails team to introduce this is beyond the scope of this answer but the TL;DR is that `secret_token.rb` conflates configuration and code as well as being a security risk since the token is checked into source control history and the only system that needs to know the production secret token is the production infrastructure.

You should add this file to `.gitignore` much like you wouldn't add `config/database.yml` to source control either.

Referencing Heroku's own code for setting up `config/database.yml` from `DATABASE_URL` in their Buildpack for Ruby, I ended up forking their repo and modified it to create `config/secrets.yml` from `SECRETS_KEY_BASE` environment variable.

Since this feature was introduced in Rails 4.1, I felt it was appropriate to edit `./lib/language_pack/rails41.rb` and add this functionality.

The following is the snippet from the modified buildpack I created at my company:

```
class LanguagePack::Rails41 < LanguagePack::Rails4

  # ...

  def compile
    instrument "rails41.compile" do
      super
      allow_git do
        create_secrets_yml
      end
    end
  end

  # ...

  # writes ERB based secrets.yml for Rails 4.1+
  def create_secrets_yml
    instrument 'ruby.create_secrets_yml' do
      log("create_secrets_yml") do
        return unless File.directory?("config")
        topic("Writing config/secrets.yml to read from SECRET_KEY_BASE")
        File.open("config/secrets.yml", "w") do |file|
          file.puts <<-SECRETS_YML
<%
raise "No RACK_ENV or RAILS_ENV found" unless ENV["RAILS_ENV"] || ENV["RACK_ENV"]
%>

<%= ENV["RAILS_ENV"] || ENV["RACK_ENV"] %>:
  secret_key_base: <%= ENV["SECRET_KEY_BASE"] %>
          SECRETS_YML
        end
      end
    end
  end

  # ...

end
```

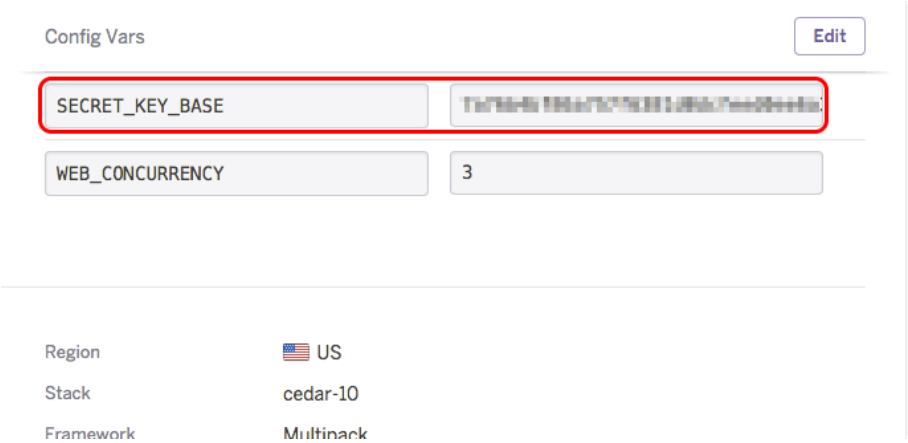
You can of course extend this code to add other secrets (e.g. third party API keys, etc.) to be read off of your environment variable:

```
...
<%= ENV["RAILS_ENV"] || ENV["RACK_ENV"] %>:
  secret_key_base: <%= ENV["SECRET_KEY_BASE"] %>
  third_party_api_key: <%= ENV["THIRD_PARTY_API"] %>
```

This way, you can access this secret in a very standard way:

```
Rails.application.secrets.third_party_api_key
```

Before redeploying your app, be sure to set your environment variable first:



Then add your modified buildpack (or you're more than welcome to link to mine) to your Heroku app (see Heroku's documentation) and redeploy your app.

The buildpack will automatically create your `config/secrets.yml` from your environment variable as part of the dyno build process every time you `git push` to Heroku.

EDIT: Heroku's own documentation suggests creating `config/secrets.yml` to read from the environment variable but this implies you should check this file into source control. In my case, this doesn't work well since I have hardcoded secrets for development and testing environments that I'd rather not check in.

edited Jun 25 at 23:25

answered Jun 25 at 23:19

stackunderflow

31 5

I have a patch that I've used in a Rails 4.1 app to let me continue using the legacy key generator (and hence backwards session compatibility with Rails 3), by allowing the secret_key_base to be blank.

```
Rails::Application.class_eval do
  # the key_generator will then use
ActiveSupport::LegacyKeyGenerator.new(config.secret_token)
  fail "I'm sorry, Dave, there's no :validate_secret_key_config!" unless
instance_method(:validate_secret_key_config!)
  def validate_secret_key_config! #:nodoc:
    config.secret_token = secrets.secret_token
    if config.secret_token.blank?
      raise "Missing `secret_token` for '#{Rails.env}' environment, set this value
in `config/secrets.yml`"
    end
  end
end
```

I've since reformatted the patch are submitted it to Rails as a Pull Request

answered Sep 28 '14 at 1:57

BF4

499 4 10

I've created `config/initializers/secret_key.rb` file and I wrote only following line of code:

```
Rails.application.config.secret_key_base = ENV["SECRET_KEY_BASE"]
```

But I think that solution posted by @Erik Trautman is more elegant ;)

Edit: Oh, and finally I found this advice on Heroku: https://devcenter.heroku.com/changelog-items/426 :)

Enjoy!

edited Dec 25 '14 at 11:35

answered Dec 25 '14 at 11:25

fadehelix

21 9

I had the same problem after I used the .gitignore file from https://github.com/github/gitignore/blob/master/Rails.gitignore

Everything worked out fine after I commented the following lines in the .gitignore file.

```
config/initializers/secret_token.rb
config/secrets.yml
```

answered Feb 27 at 17:55

user4608567

As is repeated everywhere, committing secrets.yml or secret_token.rb to git is NOT recommended. – cofiem Sep 23 at 14:41
