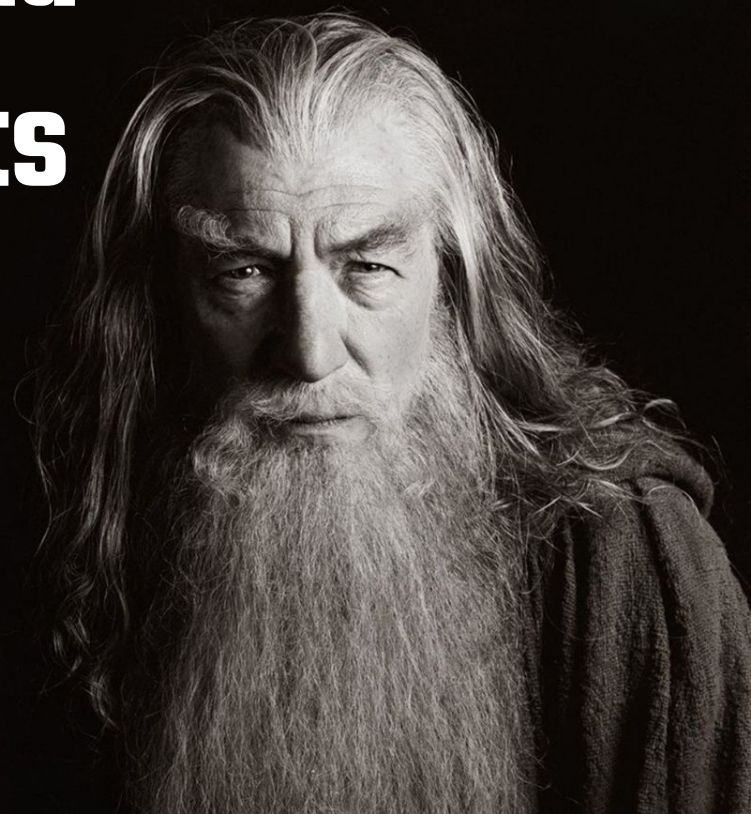
Protect sensitive data with Symfony secrets



Boozt.com
PLATFORM
POWER TO THE PEOPLE

Keep our secrets in secret

All sensitive data like passwords, secret tokens and API keys should not be stored as a plain text in the code. So no one (Gitlab, developers, etc.) has access to them.

How does it work

1. Generate a pair of encryption keys.
2. Encrypt secrets with the public key.
3. Use the private key on the production server to decrypt secrets.
4. Profit!

Step 0
Prepare the project

First of all we need to move all sensitive values to the environment variables.

Symfony provides secrets like a normal environment variables.

.env

```
MY_SECRET_TOKEN="asdad2342^#$g"
```

bootstrap.php

```
require __DIR__ . '/vendor/autoload.php';
(new Dotenv(true))->load(__DIR__ .'/.env');
```

parameters.yaml

```
parameters:
  token: '%env(MY_SECRET_TOKEN)%'
```

Step 1
Generate the encryption keys

We need to generate encryption keys for each environment:

```
$ APP_RUNTIME_ENV={env}
$ php bin/console secrets:generate-keys
```

It generates a pair of keys:

```
config/secrets/{env}/{env}.decrypt.private.php
config/secrets/{env}/{env}.encrypt.public.php
```

Potential pitfalls

Add the production's private key file to the `.gitignore`. It should never be in the repo.

Use different key pairs for different environments.

Change nothing in the `config/secrets/*` directories manually. All files there are auto generated.

Step 2
Encrypt the secrets

We can add a new or update an existing secret:

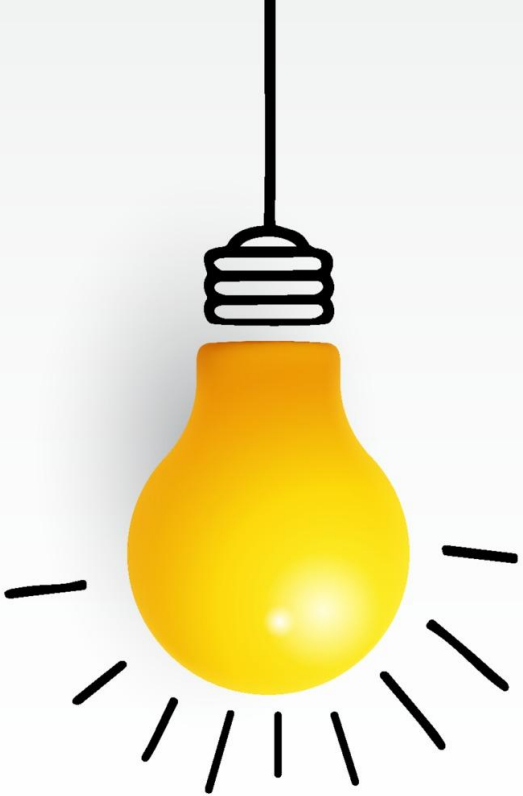
```
$ APP_RUNTIME_ENV={env}
$ php bin/console secrets:set {name}
```

Symfony will prompt the value, encrypt it and put it into the `config/secrets/{env}` directory.

Benefit

We do not need the private key for adding or updating secrets. So we can do it easily by ourselves.

No tickets, no waiting, no delais!



Potential pitfalls

Environment variables have higher priority than secrets. The secret will be overridden with the environment variable with the same name.

Step 3
Deliver the private key to production

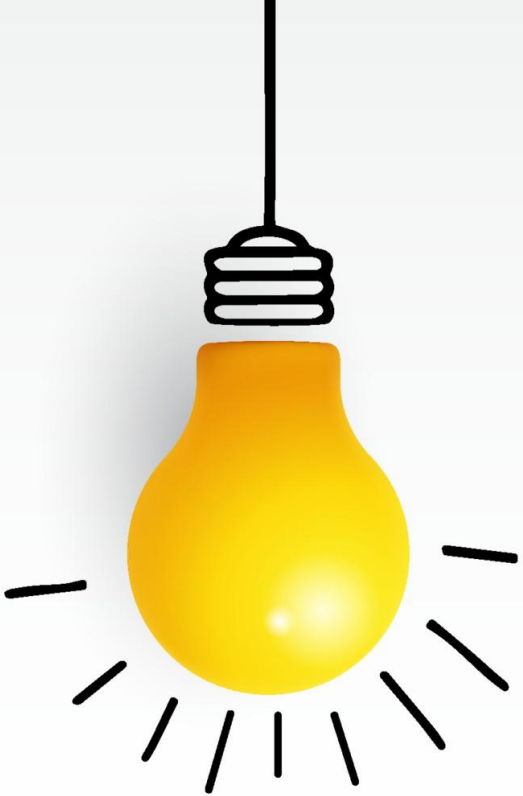
We have to keep the production private key safely out of the repo!

It should be delivered to production and be placed in the:
`config/secrets/{prod-env}/{prod-env}.decrypt.private.php`

Symfony will use it automatically. We don’t need to do anything else.

Alternative

You can pass the private key with the `SYMFONY_DECRYPTION_SECRET` environment variable instead of the file.



Step 4 (optional but strongly recommended)

Decrypt secrets on production

Run to decrypt all secrets and put them to the `.env.{env}.local` file:

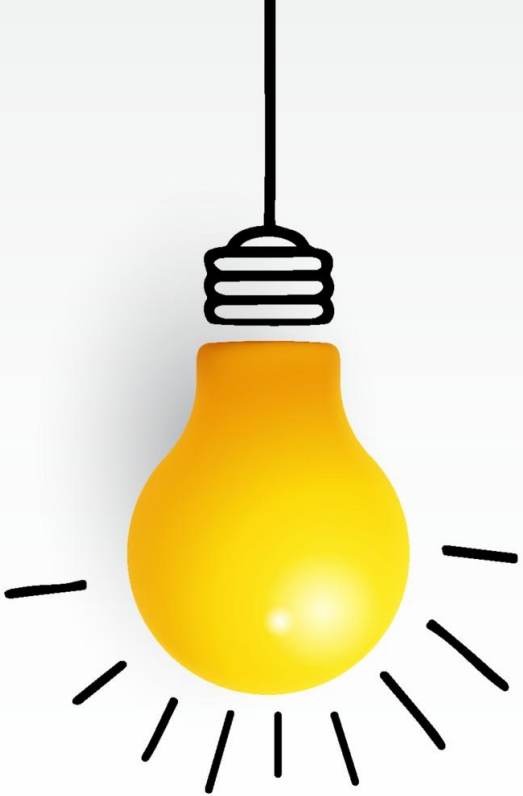
```
$ php bin/console secrets:decrypt-to-local --force
```

It allows Dotenv to put all secrets into environment variables.

Benefit

It allows other non-Symfony php code to access secrets with the `getenv()` function.

It will decrease the load on the server.



Potential pitfalls

Dotenv should be configured and work correctly to load secrets from the `.env.{env}.local` file properly.

Step 5
Enjoy safety (But never relax)

You are awesome! There are two pies on the shelf. Take the one in the middle. It’s yours.

Questions?