Webcast Website

From 0 to deployment

hello!

I am René Fernández

I am here to talk about the webcast website.
https://webcast.web.cern.ch

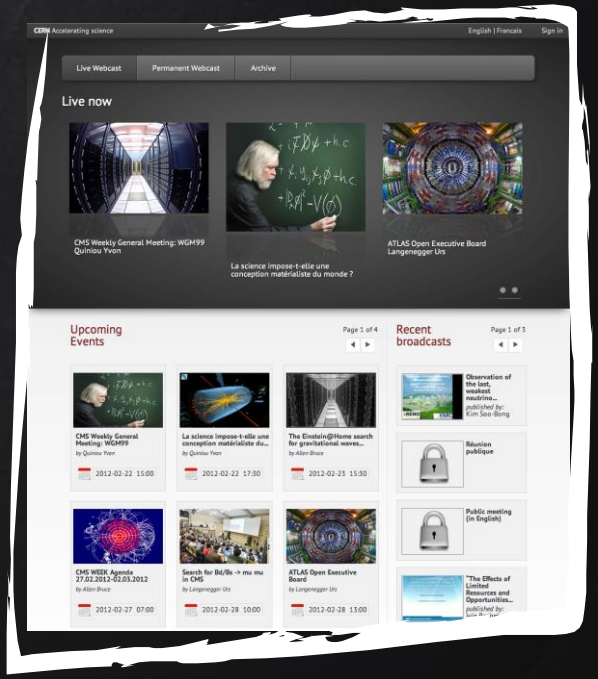
Introduction

- ✗1. A brief history
- ✗2. The beginning of the project
- ✗3 Starting the Development
- ✗4. Deployment Day

1. A brief history

- ✗ First one: Python on web services. Overloaded.
- ✗ Previous one on 2012
- ✗ Built using PHP and HMTL
- ✗ 7 developers over 6 years
- ✗ Hard to maintain

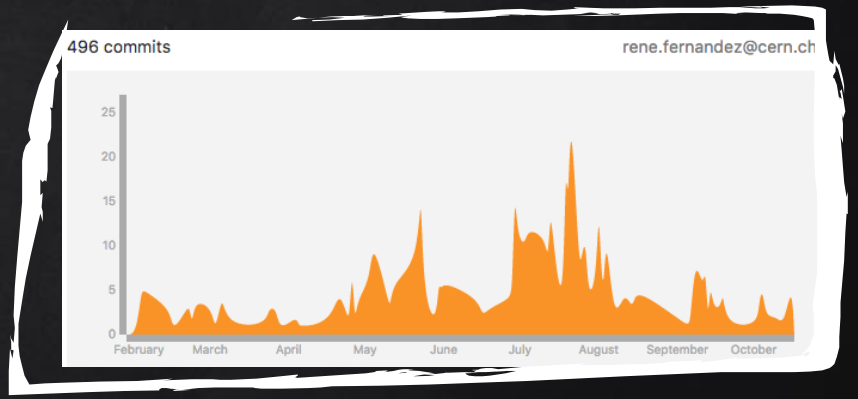
✗Contributors: Agon Bexheti, Nicola Tarocco, Simon Vocella, Marek Domaracky, Artur Friesen, Magdalena Zofia Peksa, René Fernández



Let’s build a new website!

The beginning of the project

- ✗ Started on February 2017
- ✗ Gathering the requirements
- ✗ Moving to Python stack
- ✗ Learning how the old website worked
- ✗ Moving to Openshift



It will be easy

myself


NO.

The –ALWAYS WRONG- very first glimpse

2 parts

OAuth

API

Frontend (users)

Backend (admins)

A simple UI

Same models

The real deal

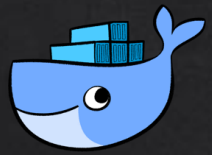
- Fronted & Backend
- Oauth
- Chrome, Firefox, Safari , IE and mobile...
- Javascript video player
- Integration with Indico
- Audiences
- Not so simple UI
- EOS
- Analytics
- Several API methods with different authentication each
- RSS Feeds
- Integration with CDS
- Integration with WOWZA streaming servers

Starting the development

How everything changed

Setting up the environment

```
  production:
    # This is the production Docker config
    build:
      context: ./web
      dockerfile: Dockerfile
    # The generated image will be named as
    # We shall not use tags at this point.
    image: webapp
  dev:
    # Development image that will extend t
    # so we won't need to change the produ
    build:
      context: ./web
      dockerfile: Dockerfile-dev
    # The generated image will be named as
    image: webapp:dev
    links:
      # We need to ensure that the product
      - production
  web:
    # The web application itself.
    restart: always
    image: webapp:dev
```



Dockerfile

Dockerfile-dev

webcast

webcast-test

Continuous Integration

Technologies Used (1/2)

PYTHON + FLASK

Used among other projects in our section and great for keeping logic and presentation separated.

POSTGRESQL + SQLALCHEMY

For the database we replaced MySQL.

EOS

Needed to share folders between projects.

Technologies Used (2/2)

THEOPLAYER

The video player we use in the website. It allows us to play live streams.

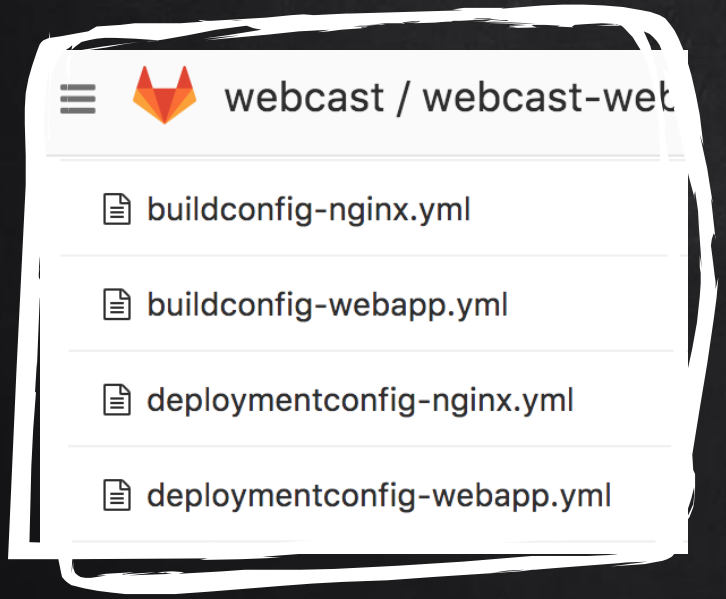
JQUERY+ FLASK ASSETS

Easy to use if the javascript we have is not very complex. Code can be merged and compressed easily.

SEMANTIC UI + SASS + PYSCSS

Advanced support for stylesheets also handled with Flask Assets.

The Openshift Configuration

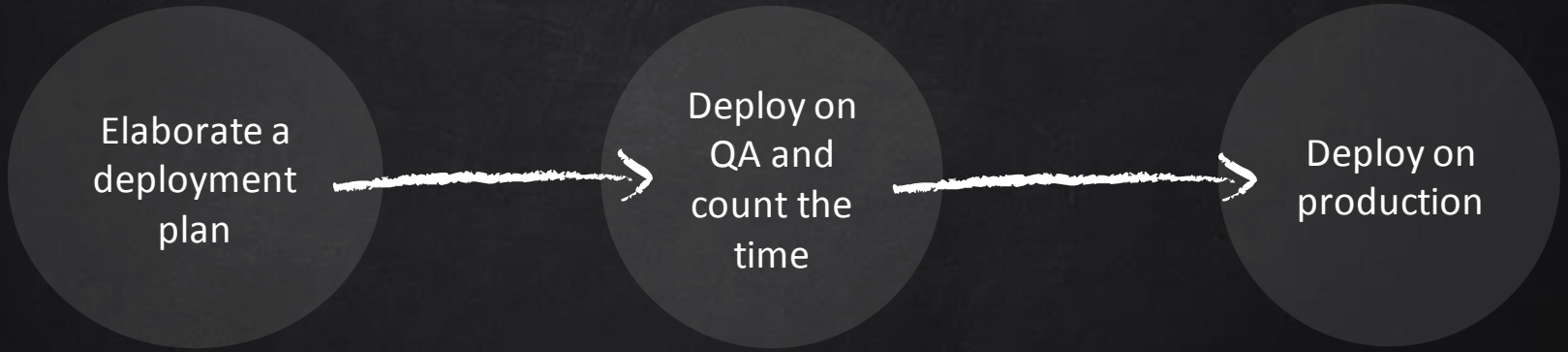


- ✗ 2 folders: dev & production
- ✗ 1 file per element: deployment config, build config, service & image stream

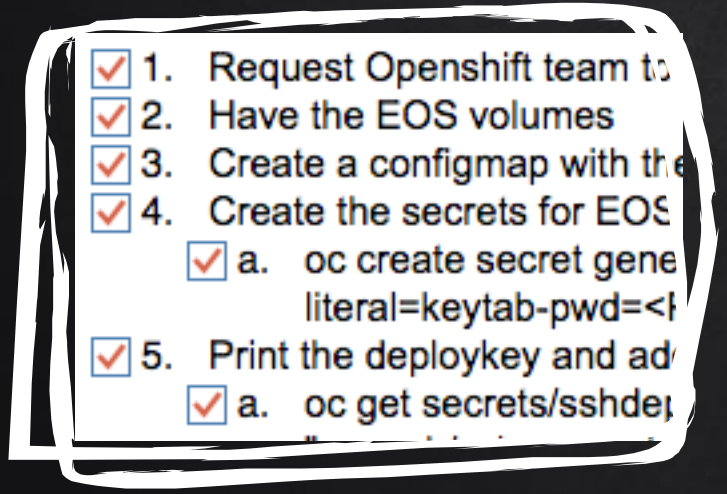
Deployment Day

When everything changed

Deployment Big Picture



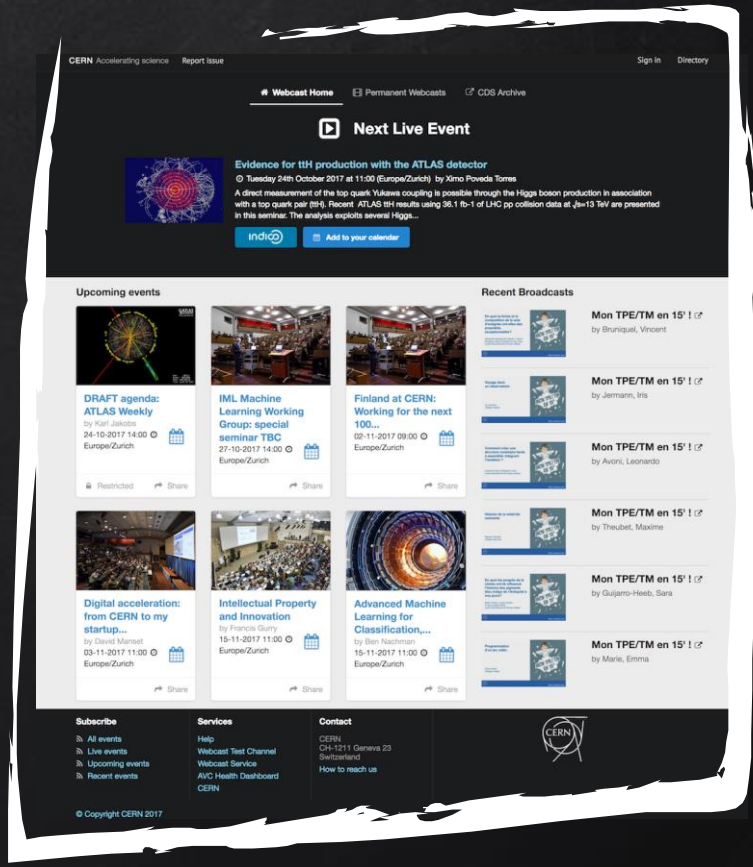
Deployment Details



- ✗ Some things can be done before
- ✗ If more people is involved, tell them your plan in advance
- ✗ Create an SSB
- ✗ Unable to do it transparently
- ✗ Be ready for trouble

The Result & The conclusion

- ✗ Know well what end users need
- ✗ A website is always more than website
- ✗ Know well the requirements.
- ✗ Be ready for failure.
- ✗ Perfection doesn't exist.
- ✗ Do not underestimate the project. Good things take time.
- ✗ You can do it! ☺



thanks!

Any questions?

rene.fernandez@cern.ch