{golem}

Production-Grade Shiny Apps

Retrouvez les slides : https://speakerdeck.com/colinfay

Colin Fay - ThinkR

$ whoami

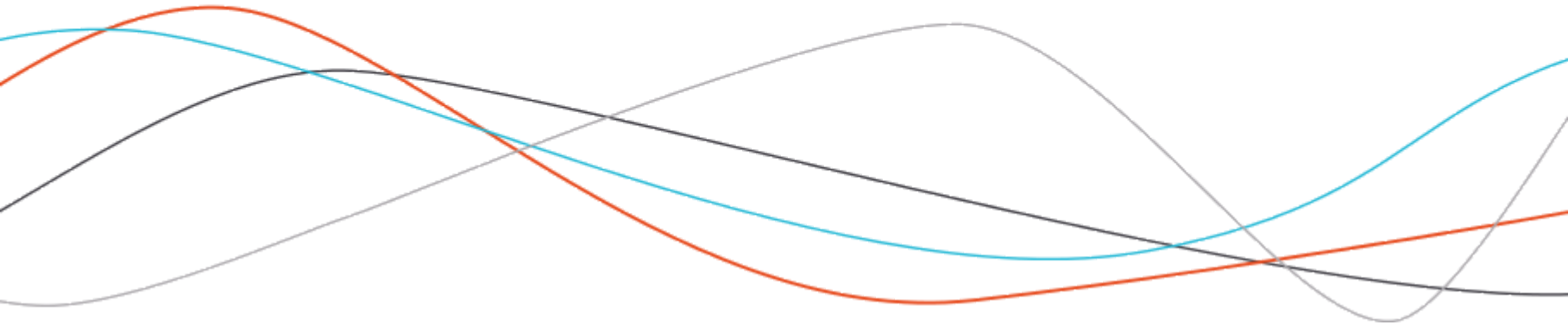
Colin FAY

Data Scientist & R-Hacker at ThinkR, companie française spécialisée en Data Science & langage R.

Développeur open source hyperactif.

- http://thinkr.fr
- http://rtask.thinkr.fr
- http://twitter.com/_colinfay
- http://github.com/colinfay

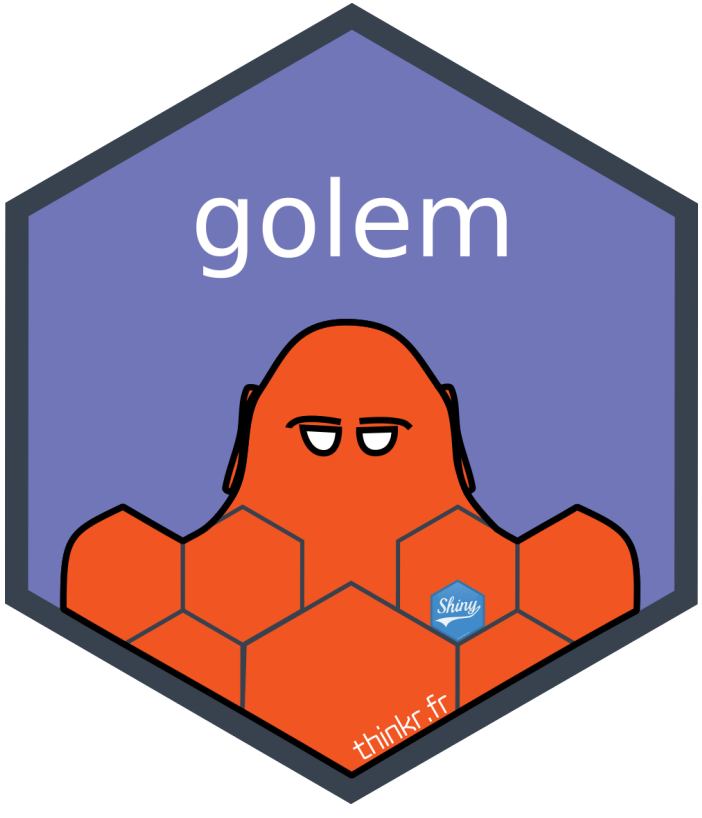
ThinkR



ThinkR

Data Science & R engineering.

- Formation
- Software Engineering
- R en production
- Consulting



> A Framework for Building Robust Shiny Apps

C'est quoi {golem}?

`{golem}` est un **package R** destiné à la création d'application Shiny pour la production.

Utiliser ce package impose un cadre relativement strict, mais permet de se passer des réflexions techniques d'infrastructure. Même si le package semble imposant, en pratique le workflow est assez simple à suivre.

Installer {golem}

```
install.packages("golem")
# DEV VERSION
# install.packages("remotes")
remotes::install_github("Thinkr-open/golem")
```

> Notes : il y a 1000 façon de faire une application Shiny, mais seulement une poignée de façon de le faire bien. {golem} est une approche qui a fait ses preuves en production.

{golem}

- CRAN version :

```
packageVersion("golem")
```

```
[1] '0.1'
```

- dev version :

```
x <- tempfile()
download.file("https://raw.githubusercontent.com/ThinkR-
open/golem/dev/DESCRIPTION", x)
desc::desc_get_version(x)
```

```
[1] '0.1.0.9600'
```

Historique


```
xml2::read_html("https://github.com/ThinkR-open/golem") %>%
  rvest::html_nodes(".overall-summary-bottomless") %>%
  as.character() %>%
  htmltools::HTML()
```

- 743 commits
- 28 branches
- 0 packages
- 1 release
- Fetching contributors
- View license

```
cranlogs::cran_downloads(
  "golem",  from = "2019-08-01", to = Sys.Date() - 1
) %>%
  extract("count") %>% sum()
```

```
[1] 5051
```

Pourquoi {golem}?

Pourquoi utiliser `{golem}`?

- Automatisation des tâches ~~ennuyeuses~~ répétitives
- Travailler avec des outils fiables
- Gagner du temps de dev
- Simplifier le déploiement
- Travailler avec un standard en équipe

`{golem}` chez ThinkR:

- D'abord construit à partir d'un besoin interne, aujourd'hui utilisé au quotidien (Je suis l'utilisateur #1 de `{golem}`)
- Nous avions besoin d'outils fiables et cohérents pour le déploiement dans les environnements de nos clients
- Construire et partager les bonnes pratiques au niveau mondial
- Promouvoir R & Shiny en production

Shiny App As a Package

Qu'est-ce qu'une application "prod-ready" ?

- Metadata (`DESCRIPTION`)
- Divisée en fonctions (`R/`)
- Testée (`tests/`)
- Avec ses dépendences (`NAMESPACE`)
- Documentée (`man/` & `vignettes`)

Donc, un 📦

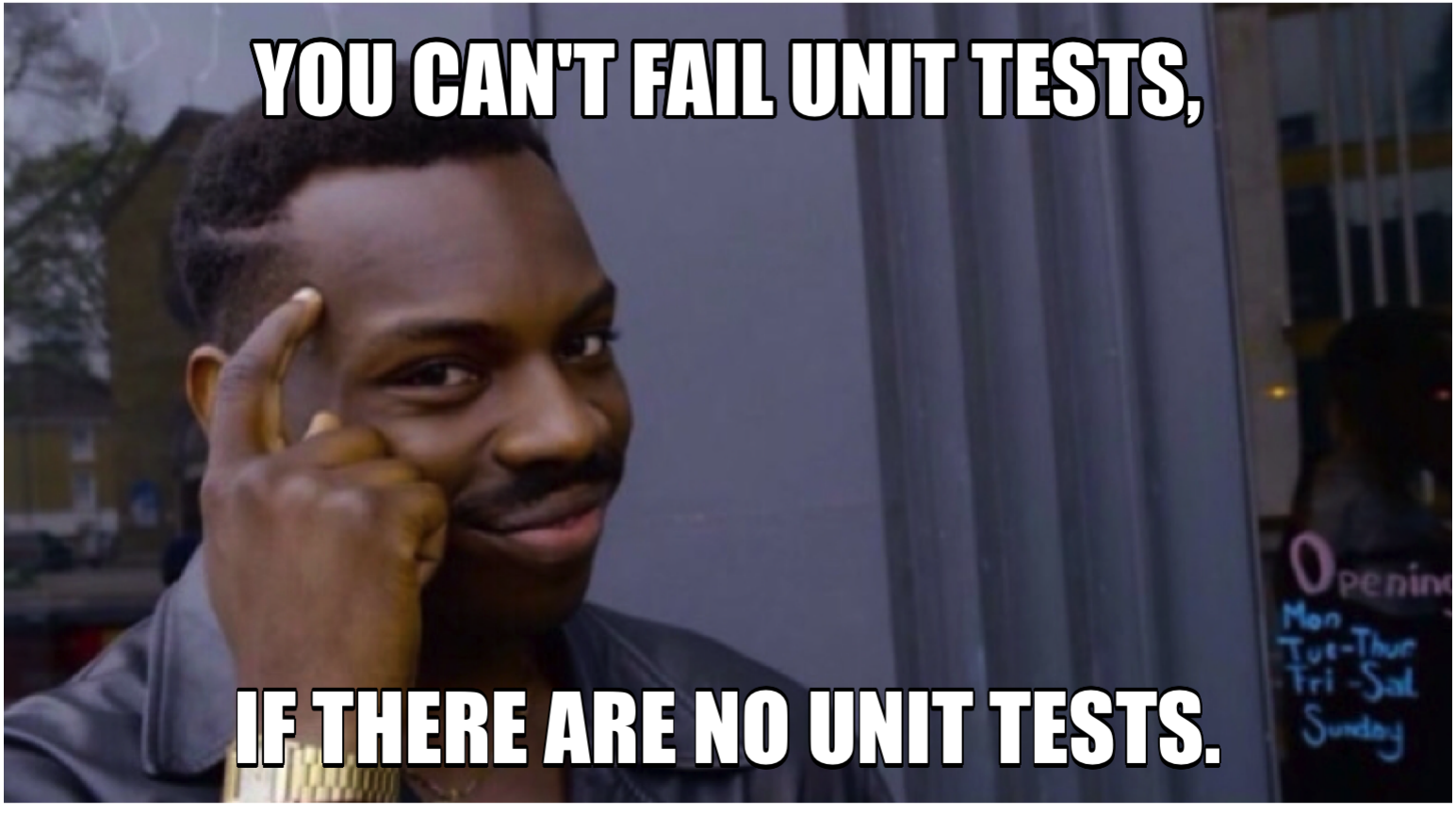
Shiny App As a Package

Le côté positif : tout ce que vous savez sur le développement de packages fonctionne avec `{golem}`.

Notamment :

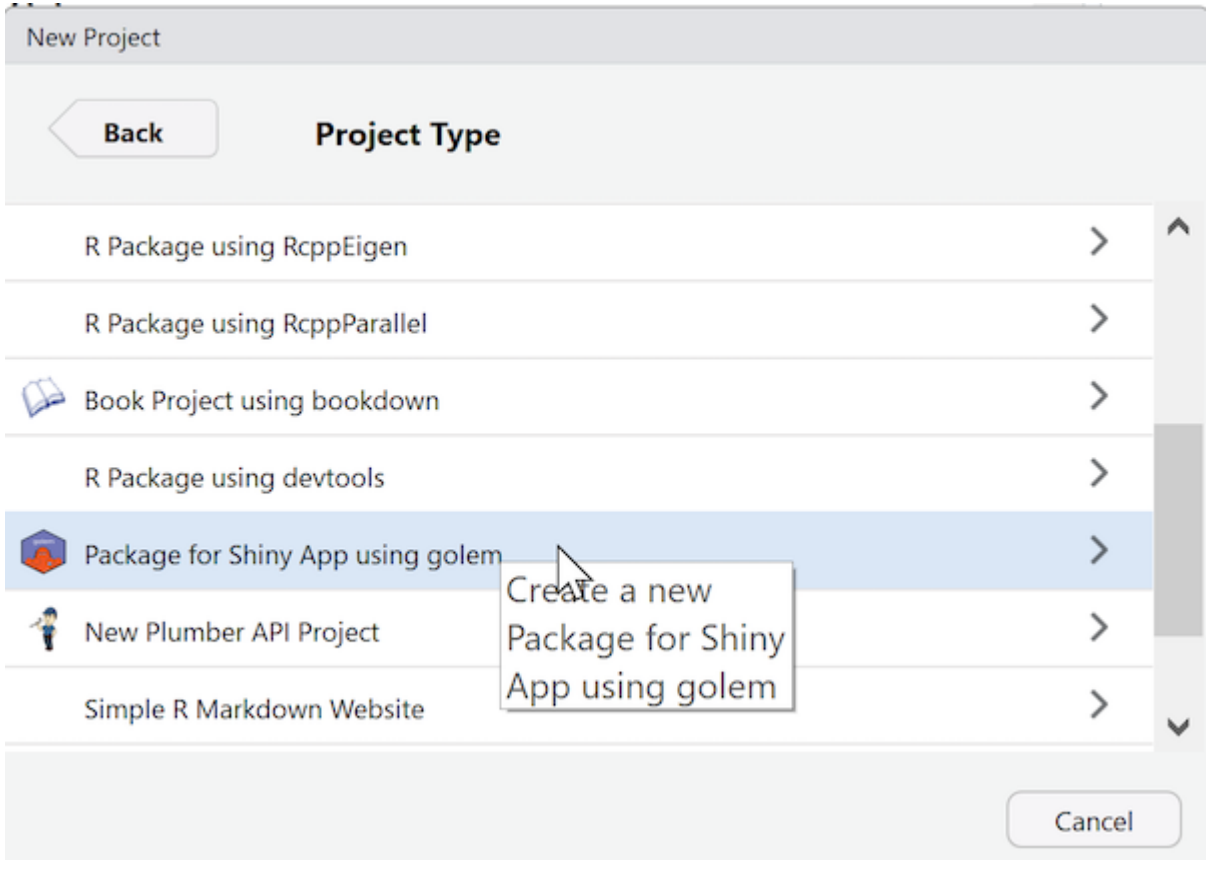
- 📖 Documentation
- 🧹 Testing

À propos des tests



Comprendre {golem}

Lancer {golem}



La structure d'un {golem}


```
fs::dir_tree("golex")
```

```
golex
├── DESCRIPTION
├── NAMESPACE
├── R
|   ├── app_server.R
|   ├── app_ui.R
|   └── run_app.R
├── dev
|   ├── 01_start.R
|   ├── 02_dev.R
|   ├── 03_deploy.R
|   └── run_dev.R
├── inst
|   └── app
|       └── www
|           └── favicon.ico
└── man
    └── run_app.Rd
```

- DESCRIPTION & NAMESPACE : Méta données du package.
- dev/ : outils de dev.
- inst/app : on ajoutera des fichiers externes dans www/. On ne touche pas à ui.R et server.R.
- man : documentation de l'app, se génère automatiquement
- R/app_server.R, app_ui.R : ce sont les deux seuls fichiers que l'on va remplir.
- R/run_app.R & onload.R : fonction qui va lancer l'app, et la configurer.

La structure d'un {golem}

app_server.R

```
#' @import shiny
app_server <- function(input, output,session) {
  # List the first level callModules here
}
```

Cette première fonction contient la logique de votre serveur. Cette fonction peut être considérée comme un remplacement du contenu de la fonction que vous avez dans votre `server.R`.

La construction d'une application Shiny complexe implique généralement l'utilisation de modules Shiny. Si c'est le cas, vous allez ajouter une série de `callModule()`, ceux que vous obtiendrez tout en bas du fichier créé avec `golem::add_module()`.

La structure d'un {golem}

app_ui.R

```
#' @import shiny
app_ui <- function() {
  tagList(
    # Leave this function for adding external resources
    golem_add_external_resources(),
    # List the first level UI elements here
    fluidPage(
      h1("golex")
    )
  )
}
```

Ce morceau du `app_ui.R` est conçu pour recevoir la contrepartie de ce que vous mettez dans votre serveur.

La structure d'un {golem}

app_ui.R

```
#' @import shiny
golem_add_external_resources <- function(){

  addResourcePath(
    'www', system.file('app/www', package = 'golex')
  )

  tags$head(
    golem::activate_js(),
    golem::favicon()
    # Add here all the external resources
    # If you have a custom.css in the inst/app/www
    # Or for example, you can add shinyalert::useShinyalert() here
    #tags$link(rel="stylesheet", type="text/css", href="www/custom.css")
  )
}
```

La deuxième partie de ce fichier contient la fonction `golem_add_external_resources()`, qui est utilisée pour ajouter des ressources externes.

La structure d'un {golem}

run_app.R

```
#' Run the Shiny Application
#'
#' @export
#' @importFrom shiny shinyApp
#' @importFrom golem with_golem_options
run_app <- function(...) {
  with_golem_options(
    app = shinyApp(ui = app_ui, server = app_server),
    golem_opts = list(...)
  )
}
```

Cette fonction `run_app()` est celle que vous utiliserez pour lancer l'application.

Elle est dans `with_golem_options()`, qui vous permet de passer des arguments à la fonction `run_app()`, qui sera plus tard utilisable avec `golem::get_golem_options()`.

La structure d'un {golem}

`inst/app/www/`

Contient des fichiers externes, notamment ceux créés avec :

- `golem::add_css_file()`
- `golem::add_js_file()`
- `golem::add_js_handler()`
- `golem::use_favicon()`

À propos de dev/

About the dev/ folder


```
fs::dir_tree("golex/dev")
```

```
golex/dev
├── 01_start.R
├── 02_dev.R
├── 03_deploy.R
└── run_dev.R
```

Fichiers qui regroupent le flux de travail :

- `01_start.R` : lancé une fois au début du projet
- `02_dev.R` : développement au jour le jour
- `03_deploy.R` : à utiliser avant d'envoyer à prod
- `run_dev.R` : pour relancer votre application pendant le développement

01_start.R

- `golem::fill_desc()`
- `golem::set_golem_options()`
- `golem::use_recommended_tests()`
- `golem::use_recommended_deps()`
- `golem::use_favicon()`
- `golem::use_utils_ui()` & `golem::use_utils_server()`

02_dev.R

```
golem::add_module( name = "my_first_module")
```

```
✓ File created at R/mod_my_first_module.R
■ Go to R/mod_my_first_module.R
```

02_dev.R

```
# Module UI

#' @title   mod_my_first_module_ui and mod_my_first_module_server
#' @description  A shiny Module.
#'
#' @param id shiny id
#' @param input internal
#' @param output internal
#' @param session internal
#'
#' @rdname mod_my_first_module
#'
#' @keywords internal
#' @export
#' @importFrom shiny NS tagList
mod_my_first_module_ui <- function(id){
  ns <- NS(id)
  tagList(

  )
}
```

02_dev.R

```
# Module Server

#' @rdname mod_my_first_module
#' @export
#' @keywords internal

mod_my_first_module_server <- function(input, output, session){
  ns <- session$ns
}

## To be copied in the UI
# mod_my_first_module_ui("my_first_module_ui_1")

## To be copied in the server
# callModule(mod_my_first_module_server, "my_first_module_ui_1")
```

02_dev.R

- `golem::add_js_file( "script" )`
- `golem::add_js_handler( "handlers" )`
- `golem::add_css_file( "custom" )`

```
golem::add_js_handler( "handlers" )
```

```
$( document ).ready(function() {
  Shiny.addCustomMessageHandler('fun', function(arg) {

  })
});
```

http://connect.thinkr.fr/js4shinyfieldnotes/

JavaScript 4 Shiny - Field Notes

Colin Fay

Last built on 2019-11-29

Field Notes on JavaScript for Shiny Users

About this bookdown

This Bookdown is the content of an informal JavaScript training given inside ThinkR. It's in no way supposed to be a complete JavaScript course nor even a complete book. Most of the content is composed of pieces of code + comments, without complete sentences.

Please read it with this in mind.

Also, this Bookdown is `{shiny}` -centric so it focuses on things that can be useful when building Shiny application.

What this book covers : bascic JavaScript objects, adding JavaScript to Shiny apps, the DOM elements & DOM events, jQuery, `this` and attributes, and building custom inputs for Shiny.

You'll find in the "Examples" section pieces of code written in real life Shiny Apps.

The Read More part points to external resources about JavaScript and Shiny.

03_deploy.R

To RStudio Products

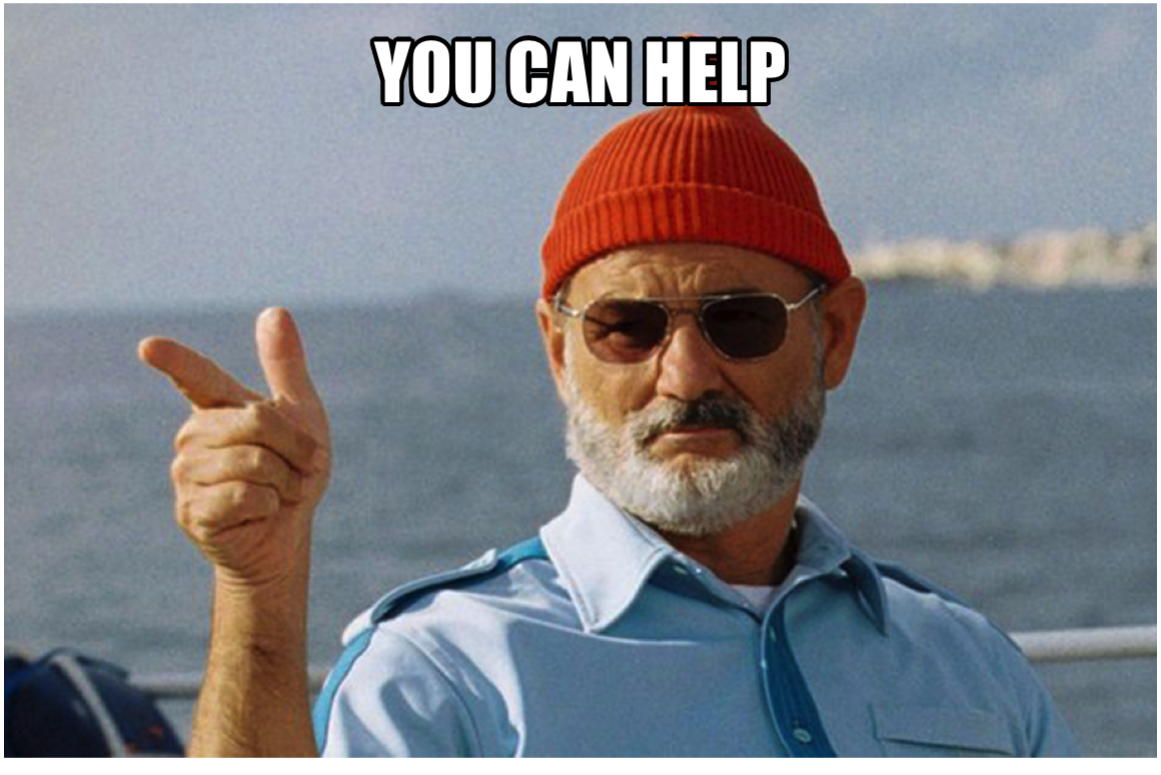
- `golem::add_rstudioconnect_file()`
- `golem::add_shinyappsio_file()`
- `golem::add_shinyserver_file()`

To Docker

- `golem::add_dockerfile()`
- `golem::add_dockerfile_shinyproxy()`
- `golem::add_dockerfile_heroku()`

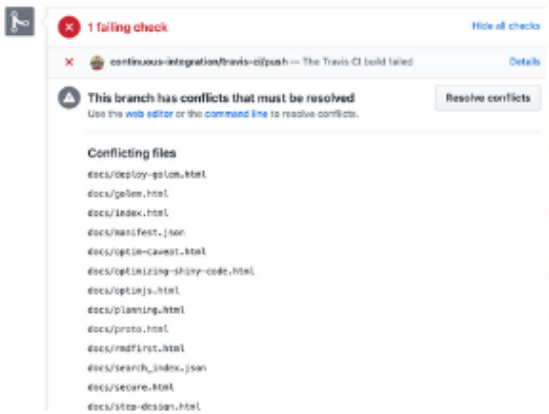
Le futur de {golem}

https://github.com/ThinkR-open/golem/issues

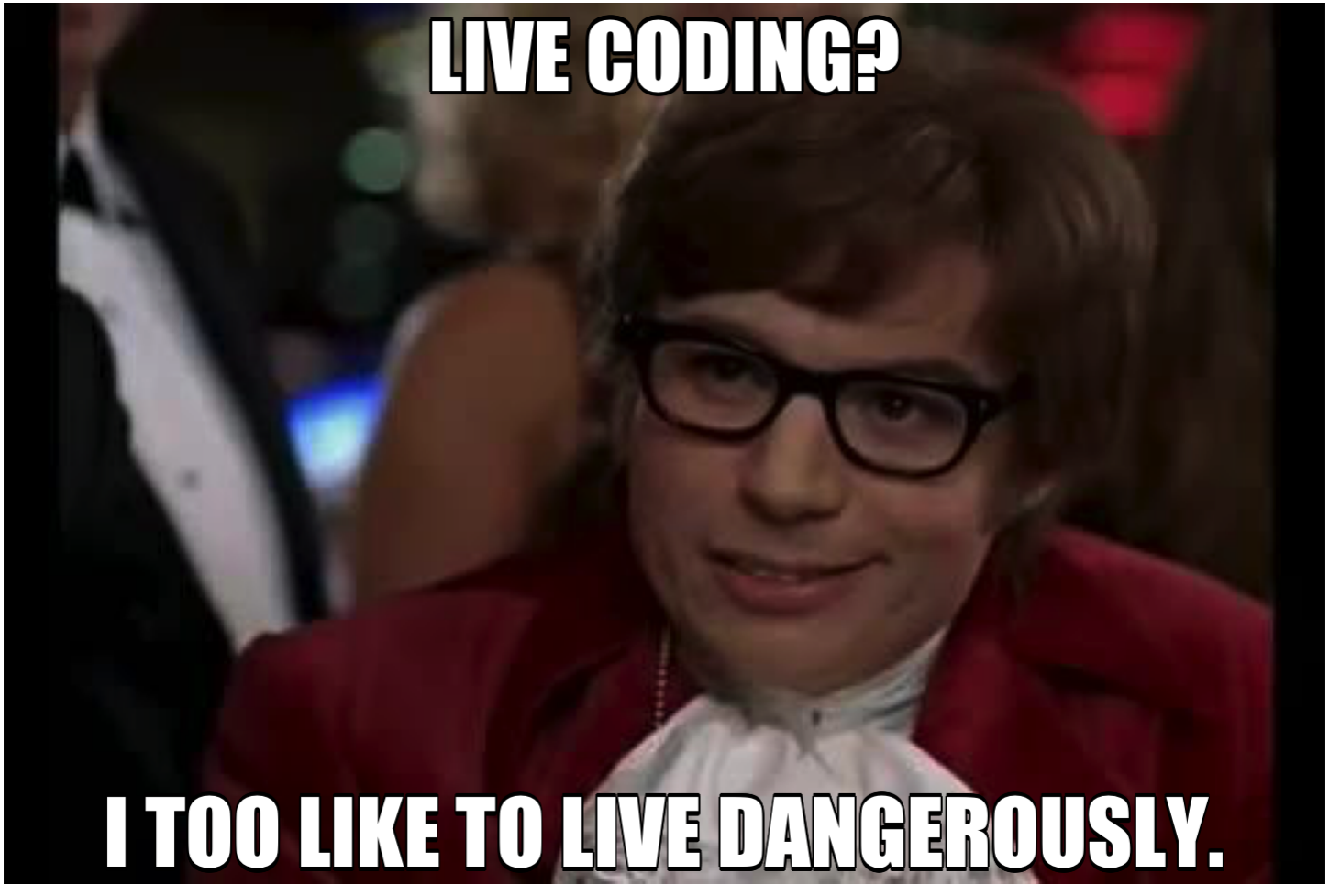


Ce que vous pouvez faire

- Passez le mot : tweets, messages de blog, parlez à vos amis et à votre famille de `{golem}`.
- Ouvrez des issues lorsque vous rencontrez un bug
- Donnez votre avis sur des choses que vous pourriez trouver bizarres
- Issues si vous avez des idées / demandes de fonctionnalités



Demo time 🎉



Thx! Questions?

Colin Fay

Online

- colin@thinkr.fr
- http://twitter.com/_colinfay
- http://twitter.com/thinkr_fr
- https://github.com/ColinFay
- https://thinkr.fr/
- https://rtask.thinkr.fr/
- https://colinfay.me/

Related projects

- building-shiny-apps-workflow
- {golem}
- {shinipsum}
- {fakir}
- {shinysnippets}