



{golem}

Engineering Production-Grade Shiny Apps

Colin Fay - ThinkR



\$ whoami

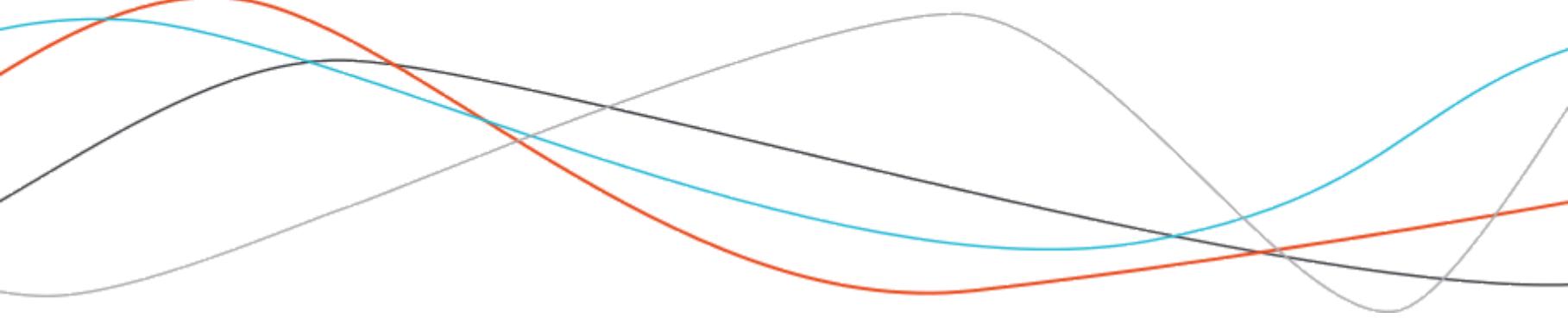
Colin FAY

Data Scientist & R-Hacker at ThinkR, a french company focused on Data Science & R.
Hyperactive open source developer.

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



ThinkR





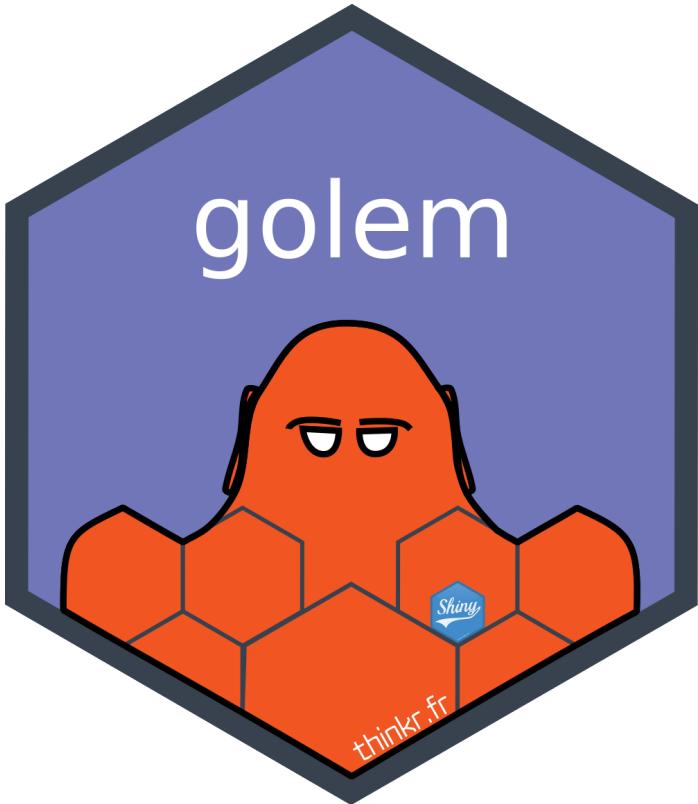
ThinkR

Data Science engineering, focused on R.

- Training
- Software Engineering
- R in production
- Consulting



{golem}



A Framework for Building Robust Shiny Apps



What is {golem}?

{golem} is an R package that provides a framework for building production-ready Shiny Applications.

The framework provided by {golem} is relatively strict, but allows to abstract away the technical points and pure engineering steps.

Install {golem}

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

Notes: there are a thousand ways to create a Shiny App, but very few ways to create a production-grade Shiny App. {golem} provides a framework to create what we believe is a production grade Shiny App.



{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'
```



{golem} history

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

- ⌚ 734 commits
- 🔖 26 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] 3610



Why {golem}?

Why using {golem}?

- Automate the ~~boring stuff~~ repetitive tasks
- Work with reliable tools
- Gain time developing
- Simplify deployment
- Standardize team work

About {golem} at ThinkR:

- First built out of internal need, today used on a daily basis (I'm the #1 {golem} user)
- We needed reliable and consistent tooling for deploying to our clients' environments
- Build and share good practices globally
- Promote R & Shiny in production



{golem} central philosophy

Shiny App As a Package

What's a "prod-ready" Shiny App?

- Comes with meta data (`DESCRIPTION`)
- Divided in functions (`R/`)
- Tested (`tests/`)
- With dependencies (`NAMESPACE`)
- Documented (`man/` & `vignettes`)

So, a a package



{golem} central philosophy

Shiny App As a Package

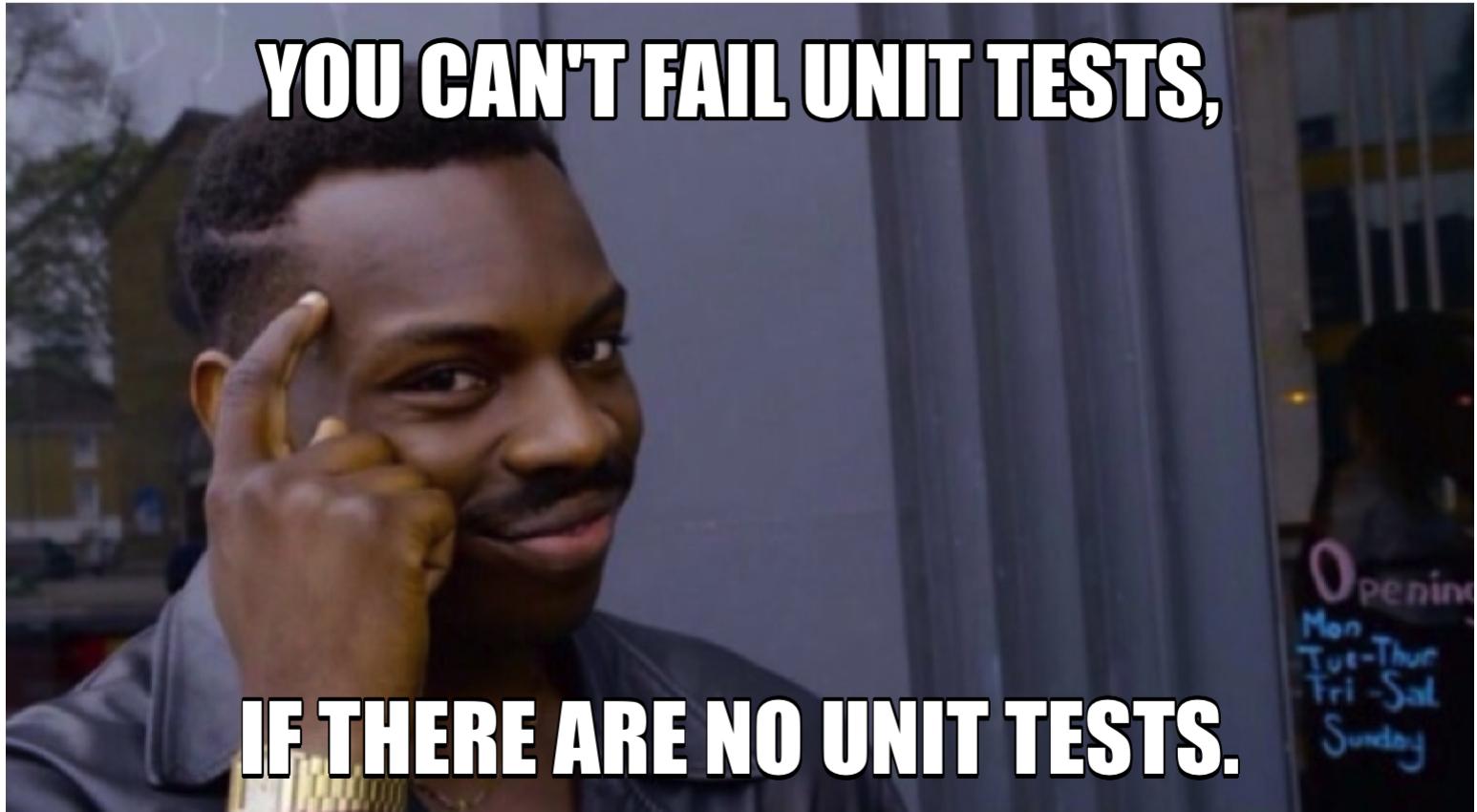
The plus side: everything you know about package development works with `{golem}`.

Notably:

- Documentation
- Testing

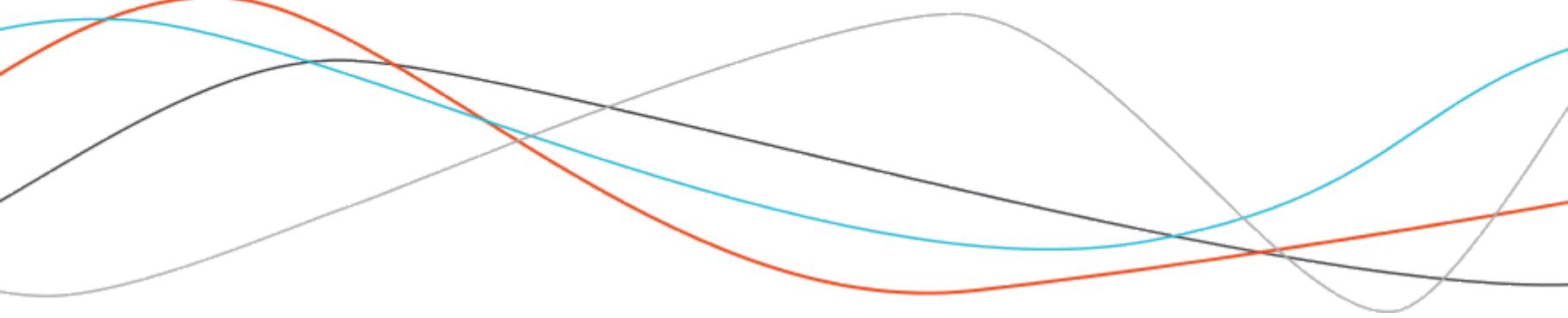


About testing





Understanding {golem}





Create a {golem}

New Project

Back **Project Type**

- R Package using RcppEigen > ^
- R Package using RcppParallel >
-  Book Project using bookdown >
- R Package using devtools > ^
-  Package for Shiny App using golem >
-  New Plumber API Project >
- Simple R Markdown Website > ^

Create a new
Package for Shiny
App using golem

Cancel



Understanding {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
```



Understanding {golem}

DESCRIPTION & NAMESPACE

The `DESCRIPTION` and `NAMESPACE` are standard package files (i.e. they are not `{golem}`-specific).

R/ & man/

The `R/` & `man/` folders are also the standard folders where you'll be putting all your app functions and documentations (not `{golem}`-specific).



Understanding {golem}

app_server.R

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

This first function contains your server logic. This function can be thought of as a drop in replacement for the content of the function you've got in your `server.R`.

Building a complex Shiny application commonly implies using Shiny modules. If so, you'll be adding there a series of `callModule()`, the ones you'll get on the very bottom of the file created with `golem::add_module()`.



Understanding {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")
    )
  )
}
```

This piece of the `app_ui.R` is designed to receive the counterpart of what you put in your server.

Everything here is to be put after the `# List the first level UI elements here` line.



Understanding {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")
  )
}
```

The second part of this file contains the `golem_add_external_resources()` function, which is used to add external resources.



Understanding {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(...))
}
```

This `run_app()` function is the one that you'll use to launch the app.

Wrapped inside `with_golem_options()`, which allows you to pass arguments to the `run_app()` function, which will later be callable with `golem::get_golem_options()`.



Understanding {golem}

inst/app/www/

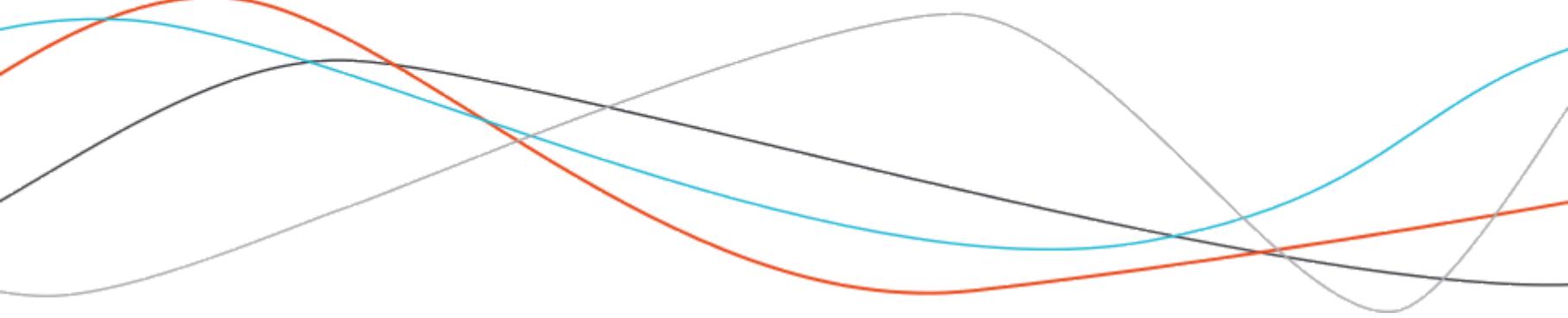
Host external files, notably the one created with:

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

(More on the later...)



About the dev/ folder





About the dev/ folder

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

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

Three files that bundle the golem workflow:

- `01_start.R`: run once at the beginning of the project
- `02_dev.R`: day to day development
- `03_deploy.R`: to use before sending to prod
- `run_dev.R`: to relaunch your app during development



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()`

And `{usethis}` commonly used calls



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

Build a skeleton for a Shiny Module



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) {
    })
});
```

And `{usethis}` commonly used calls.



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

JS 4 Shiny



[Field Notes on JavaScript for Shiny...](#)

[About this bookdown](#)

[Installing {bubble}](#)

1 A quick Intro to Base objects

[1.1 Launch a REPL](#)

[1.2 Assignment](#)

[1.3 Scalar objects](#)

[1.4 Array](#)

[1.5 Objects](#)

[1.6 Function](#)

[1.7 typeof](#)

[1.8 For loop](#)

[1.9 While loop](#)

2 JS in Shiny

[2.1 Add JS to Shiny](#)

[2.2 The JavaScript Shiny Object](#)

[2.3 Shiny JS Handlers](#)

3 DOM & DOM Events

[3.1 DOM elements](#)

[3.2 DOM elements](#)

[3.3 DOM events](#)

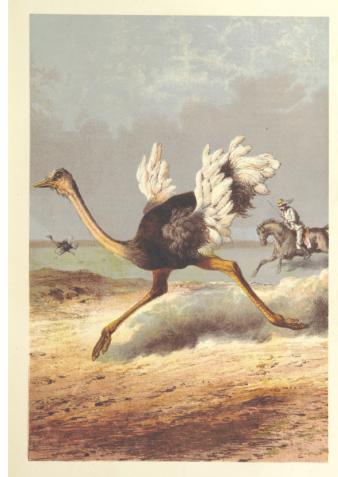
[3.4 Add Event Listeners](#)

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 : basic 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

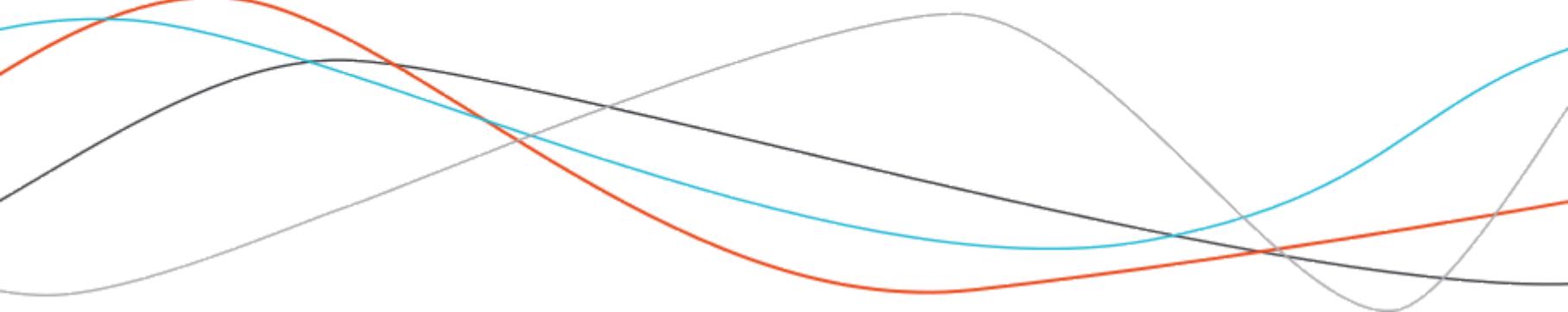
- `golem::add_rstudioconnect_file()`
- `golem::add_shinyappsi_file()`
- `golem::add_shinyserver_file()`

To Docker

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



What's new in {golem}





<https://github.com/ThinkR-open/golem.git@dev>

```
remotes::install_github("thinkr-open/golem", ref = "dev")
```

Docker

- Re-factoring of `Dockerfile` creation for faster, and more reliable deployment.
- Native guess of `sysreq`

Development

- `add_fct()` and `add_utils()` for utils and functions file creation (can be module specific)
- `use_external_js_file()` and `use_external_css_file()`
- Series of addins to navigate to file or to create a `ns()`
- New JS functions

And...

- Small changes and bug fixes (see the news file for more)



golem config

<https://github.com/ThinkR-open/golem/issues/109>

Scripted configuration

- Use a config file instead of `options`
- Easier sharing and normalization of configuration
- Rely on existing tooling (`{config}`) for production environments

The screenshot shows a GitHub issue page for the 'golem' repository. The header includes the repository name 'ThinkR-open / golem', a 'Issues 76' badge, and navigation links for Code, Issues, Pull requests, Projects, Wiki, Security, and Pull requests. Below the header are buttons for 'Edit' and 'New issue', and a 'Jump to bottom' link. The main content is an issue titled 'Use {config} #85', which is open and was created by ColinFay on June 18. The issue body contains text suggesting the use of the {config} package to set golem options, with a link to its documentation on cran.r-project.org. At the bottom of the issue view, there are interaction metrics: 2 likes, 1 comment, and 1 share.

Code Issues 76 Pull requests 6 Projects 1 Wiki Security Pull requests

Edit New issue Jump to bottom

Use {config} #85

Open ColinFay opened this issue on 18 Jun · 4 comments

Assignees

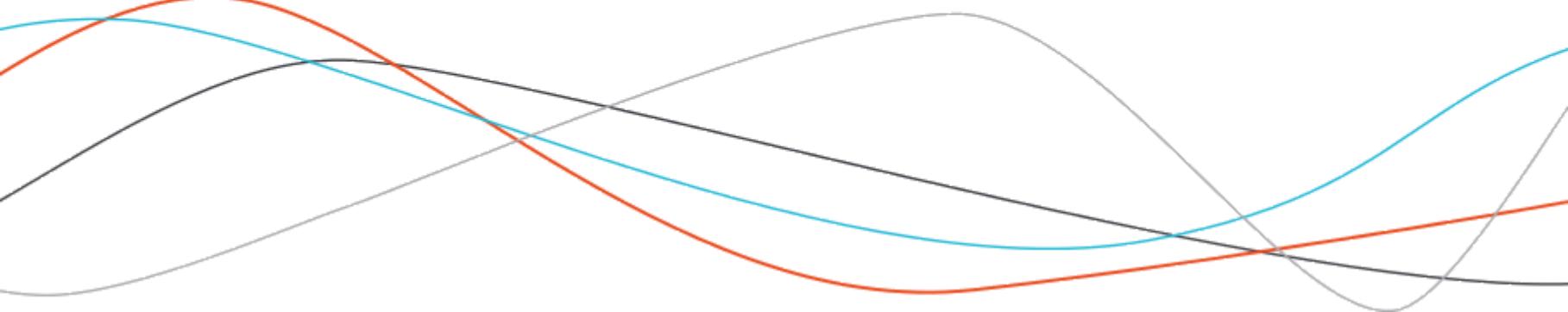
ColinFay on 18 Jun

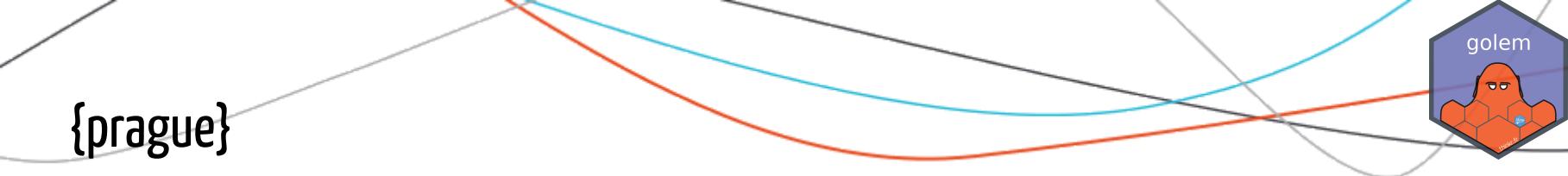
Just dropping the idea there, we should use the `{config}` package to set `golem` options (for example behaviour on `app.prod`)
cran.r-project.org/web/packages/config/vignettes/introduction.html

2 1 +



The future of {golem}





{prague}



Where golems come to life

<https://github.com/ThinkR-open/prague>

Lightweight, production focus version of the functions required to deploy a golem app to production.

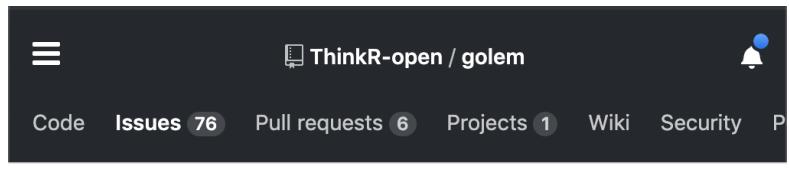
Faster and lighter deployment.

No breaking change for current {golem} apps.



golem templates

<https://github.com/ThinkR-open/golem/issues/109>



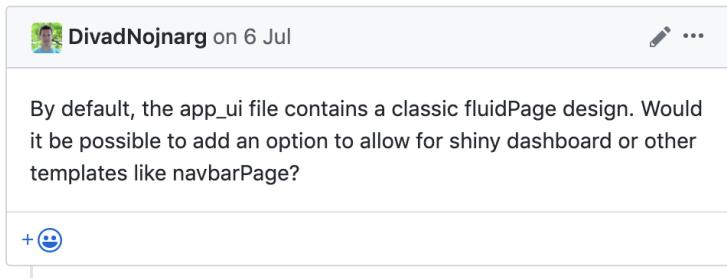
Templating projects

- Pass function with a template on app creation
- Easier and faster development in industry
- Sharing app template across a team or with external contractors

Customize app_ui function to allow other shiny templates #109

! Open DivadNojnarg opened this issue on 6 Jul · 1 comment

Projects ThinkR board





More deployment destinations

<https://github.com/ThinkR-open/golem/issues/317>

More supported platforms

- Simplify the deployment to other platforms
- Need feedback from the community: where do you usually deploy your apps?

The screenshot shows a GitHub issue page for the repository "ThinkR-open / golem". The title of the issue is "Add more deploy destinations #317". It is labeled as "Open" and was created by "ColinFay" now, with 0 comments. The issue body contains a message from ColinFay suggesting to add more deployment destinations for the {golem} package. He provides examples like Dokku, Google Cloud, AWS, and pm2. There is a "New issue" button at the top right of the page.

We should add more destinations for `{golem}` deployment.

For example:

- Dokku [dokku/dokku](#)
- Google Cloud [cloud.google.com/kubernetes-engine/docs/tutorials/hello-app](#)
- AWS [aws.amazon.com/getting-started/tutorials/deploy-docker-containers/?nc1=h_ls](#)
- pm2 [pm2.keymetrics.io](#)



Support for {renv}

<https://github.com/ThinkR-open/golem/issues/282>

The screenshot shows a GitHub issue page for the repository 'ThinkR-open / golem'. The title of the issue is 'Provide support for {renv} users #282'. The issue is marked as 'Open' and was created by DianeBeldame 21 days ago. It has 0 comments. The issue body contains a comment from DianeBeldame suggesting to 'Assess how to help {renv} users with renv.lock files'. ColinFay added this to the 'To do' list in Version 0.3.0 6 days ago. The GitHub interface includes navigation links like 'Edit' and 'New issue', and a 'Jump to bottom' link.

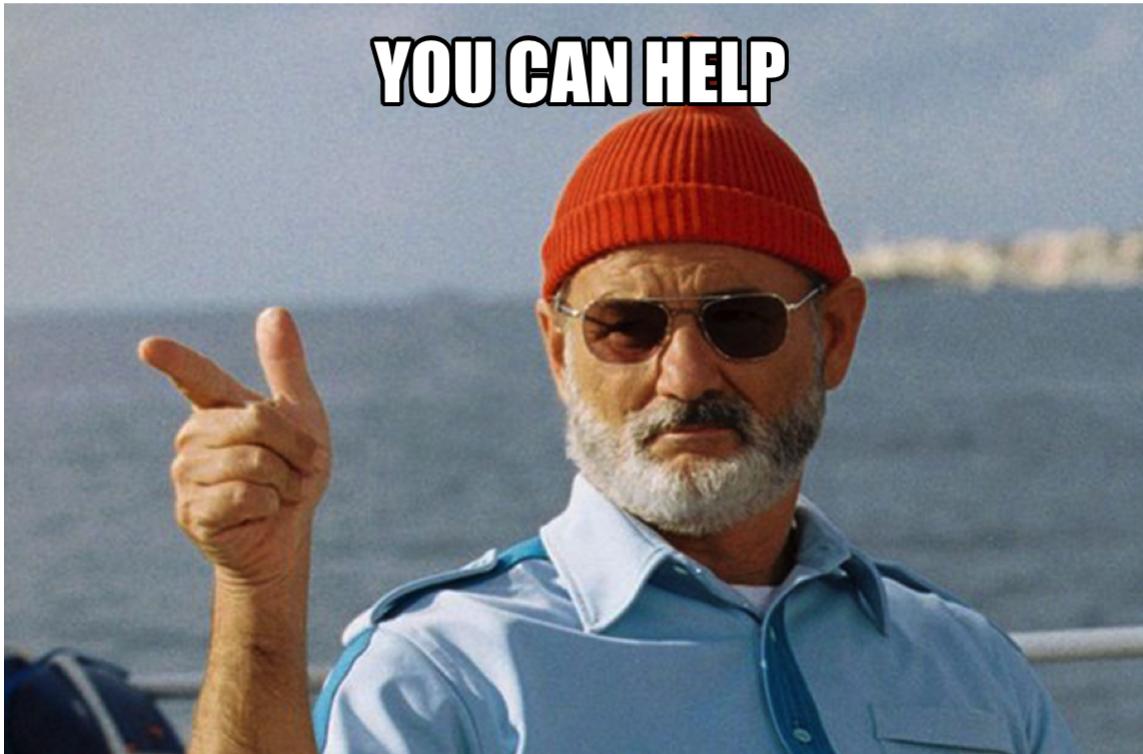
Integrate {renv}

- Better dependency management
- Integrates with RStudio products
- Works with Docker



{golem}'s future

All WIP and ideas are currently listed at <https://github.com/ThinkR-open/golem/issues>





What you can do

- Spread the word (and share stickers): tweets, blog posts, talk to your friends and family about `{golem}`
- Open issues when you encounter a bug
- Give feedback about things you might find weird
- Open issue if you have idea / feature requests

A screenshot of a GitHub pull request interface. At the top left, there's a red 'X' icon and the text '1 failing check'. Below it, another 'X' icon and the text 'continuous-integration/travis-ci/push -- The Travis CI build failed'. A warning message says 'This branch has conflicts that must be resolved' with a 'Resolve conflicts' button. On the left, under 'Conflicting files', a list includes: doc/deploy-golem.html, doc/golem.html, doc/index.html, doc/manIFEST.json, doc/optimize-caVest.html, doc/optimizing-shiny-code.html, doc/options.html, doc/planning.html, doc/proto.html, doc/nodFirst.html, doc/search_index.json, doc/secore.html, and doc/site-design.html.



Demo time



LIVE CODING?

I TOO LIKE TO LIVE DANGEROUSLY.



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}](#)