***Package*** *"****localshiny****"*

1. **Overview**

The R package *localshiny* is a deployment interface for LocalShiny web and LocalShiny app. LocalShiny web is an online service for hosting Shiny apps in the cloud. Developers, who want to share Shiny web applications built in R, use the R package *localshiny* to create the clones of their applications and then deploy them on the hosted service. LocalShiny app is designed to reproduce Shiny web apps on local computer machines by invoking the function for installation defined in the *localshiny* package.

**1.1 Package dependencies**

|  |  |
| --- | --- |
| **Package** | **Download Link** |
| renv | <https://rstudio.github.io/renv/> |
| httr | [https://CRAN.R-project.org/package=httr](https://cran.r-project.org/package=httr) |
| jsonlite | [https://CRAN.R-project.org/package=jsonlite](https://cran.r-project.org/package=jsonlite)s |
| zip | [https://CRAN.R-project.org/package=zip](https://cran.r-project.org/package=zip) |

**1.2 Supported os:**

Windows, Linux and Mac OS are currently supported.

**1.3 Package installation**

You can install the released version of *localshiny* package from GitHub with the following command in R:

*#install package dependencies*

*#install.packages(c("renv", "RCurl", "jsonlite", "zip"))*

*#install.packages("devtools")*

*devtools::install\_github("xxx/localshiny")*

To ensure you have successfully installed *localshiny*, try loading it into your R session.

*library(localshiny)*

1. **Share Shiny apps on the website**

Shiny apps are interactive web applications built with R which allow you to create highly effective data reports and visualizations where collaborators can browse a data set.[1] The *localshiny* package makes it simple for Shiny developers to deploy their apps straight from R session to the server hosted by LocalShiny. They will upload **Shiny apps as R scripts on a web page. *localshiny* will also create a lock file storing the state of recursive packages the app depends on and upload it together with the R scripts. Then users can reproduce it on their local machines with the information written in the lock file, and launch the app by running R scripts directly.**

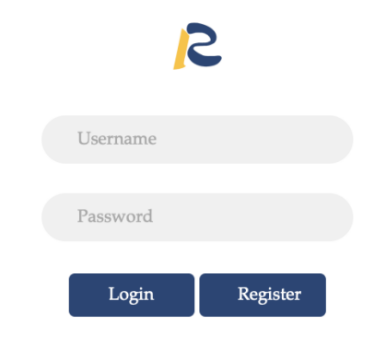
Prior to deploying applications, developers need to register their accounts on the LocalShiny web. Once they set up an account in LocalShiny web, the database will automatically generate a contiguous sequence of characters as a token. This token will be used to connect the user account to the package so that it can be used to deploy and manage applications on behalf of the account.

In this section, we’ll walk through all the steps of deploying a Shiny app on the LocalShiny website that lets you explore a set of functions available in the *localshiny* package.

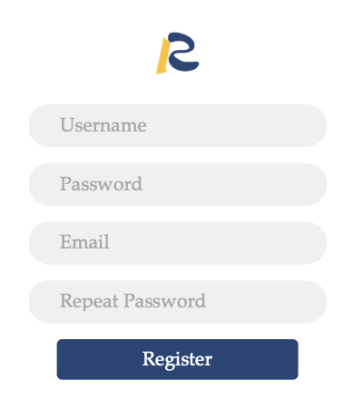
**2.1 Create an account and log in**

The first time you sign in, LocalShiny prompts you to sign up for a LocalShiny account.

1) Visit the LocalShiny web. In the upper-right corner of any page, click log in, then click register.



2) Register a new account using your username, password, and email address. Password is six or more characters, with no spaces. Then verify your email address. This verification allows you to retrieve your username or reset your password.



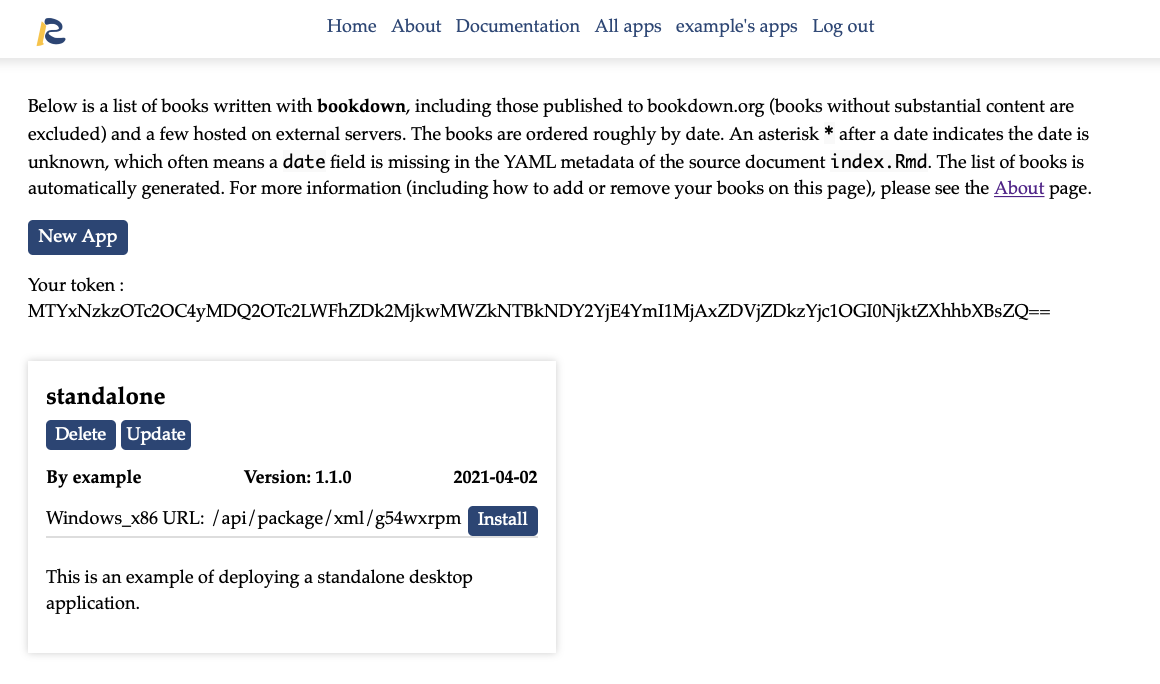
3) log in your account by clicking 'login' link at the top-right corner of the page.



**2.2 Configure the package**

After you log in, configure the package to authorize your account. Upon the successful authorization, your local installation of *localshiny* and your server account will be paired, and you’ll be able to deploy and manage applications using the package without further prompts for authentication.

Go to your account profile on which the token used to authorize your account is listed.



Copy the token and use *localshiny::login()* to configure the package.

*# copy the token then paste it into the R console.*

*localshiny::login("example",token="MTYxNxxxxxxxxXBsZQ==")*

Notice that after you log out of your account, the token will be invalid, and you need to retrieve a new one from the LocalShiny web and connect the package to your account again.

**2.3 Write a Shiny app that works**

Shiny app generally consists of two parts: the User Interface (UI) and the Server. The UI controls what is being displayed on the web page and how the components are laid out. The Server controls the data that will be displayed through the UI. You can define these two parts in one file, app.R, or separate them into two files, server.R and ui.R. Besides, you can also develop a package to build a Shiny app. Perhaps more work you have to do with it. However, it will greatly simplify further maintenance and scale your app to a broader audience. [1]Here comes an example to build a simple web app.

**server.R**

*# Define server logic to summarize and view selected dataset ----*

*server <- function(input, output) {*

*# Return the requested dataset ----*

*datasetInput <- reactive({ switch(input$dataset,*

*"rock" = rock,*

*"pressure" = pressure,*

*"cars" = cars) })*

*# Generate a summary of the dataset ----*

*output$summary <- renderPrint({ dataset <- datasetInput()*

*summary(dataset) })*

*# Show the first "n" observations ----*

*output$view <- renderTable({ head(datasetInput(), n = input$obs) })*

*}*

**ui.R**

*# Define UI for dataset viewer app ----*

*ui <- fluidPage(*

*# App title ----*

*titlePanel("Shiny Text"),*

*# Sidebar layout with a input and output definitions ----*

*sidebarLayout(*

*# Sidebar panel for inputs ----*

*sidebarPanel(*

*# Input: Selector for choosing dataset ----*

*selectInput(inputId = "dataset",*

*label = "Choose a dataset:",*

*choices = c("rock", "pressure", "cars")),*

*# Input: Numeric entry for number of obs to view ----*

*numericInput(inputId = "obs",*

*label = "Number of observations to view:",*

*value = 10)),*

*# Main panel for displaying outputs ----*

*mainPanel(*

*# Output: Verbatim text for data summary ----*

*verbatimTextOutput("summary"),*

*# Output: HTML table with requested number of observations ----*

*tableOutput("view") )*

*))*

Test that your application works by running it locally. Set your [working directory](http://www.rstudio.com/ide/docs/using/workspaces) to your project. The project to be deployed can be a .R file containing a Shiny app, or a project directory that *shiny::runApp()* can launch (i.e. a path to directory that contains a server.R and ui.R file). Then run:

*library(shiny)*

*shiny::runApp()*

If the Shiny app works, the system’s default web browser will be launched automatically after the app is started, and then you can upload it to the LocalShiny web.

**2.4 Delpoy apps on the web**

Upon you have configured the *localshiny* R package and prepared a Shiny app, you can use the function *localshiny::deployApp()* to upload the app under your account. The deployment function will snapshot the app to capture the state of a project's R package dependencies, zip up the project directory, and send it to the LocalShiny hosted service finally.

The *lcoalshiny::deployApp()* function will call *renv::snapshot()* to recursively detect all package dependencies for an app. This function parses all .R files in the application directory to determine what packages the application depends on; and for each of those packages what other packages they depend on. Dependencies are determined by parsing application source code and looking for calls to library, require, ::, and :::. Recursive dependencies are detected by examining the Depends, Imports, and LinkingTo, Suggests fields of DESCRIPTION file in the packages. All states of a project's R package dependencies will be captured and stored in a lock file. This lock file will be uploaded together with all the other application files.

If you want to add external packages to your project, you can force a package to be included dependency by inserting calls to require within your source directory. This code need not actually execute, but the packages in the file will be detected. For example, you could create a standalone file named dependencies.R with the following code:

dependencies.R

*require(ggplot2)*

*require(BH)*

This will force the *ggplot2* and *BH* packages listed in dependencies.R to be detected along with the rest of dependencies in your project.

In the next step, *lcoalshiny::deployApp()* will zip up your project directory and send it to the LocalShiny hosted service. While you upload an app, you are required to set the app name as a unique name within your account, defaults to the base name of the specified project directory. Once you upload the app on the website, the database will generate an app ID, which is unique among all distributed apps to identify your app. The app’s ID can be found in the URL while on the app’s page in a store’s control panel.

Run the following command to publish an app with *localshiny*:

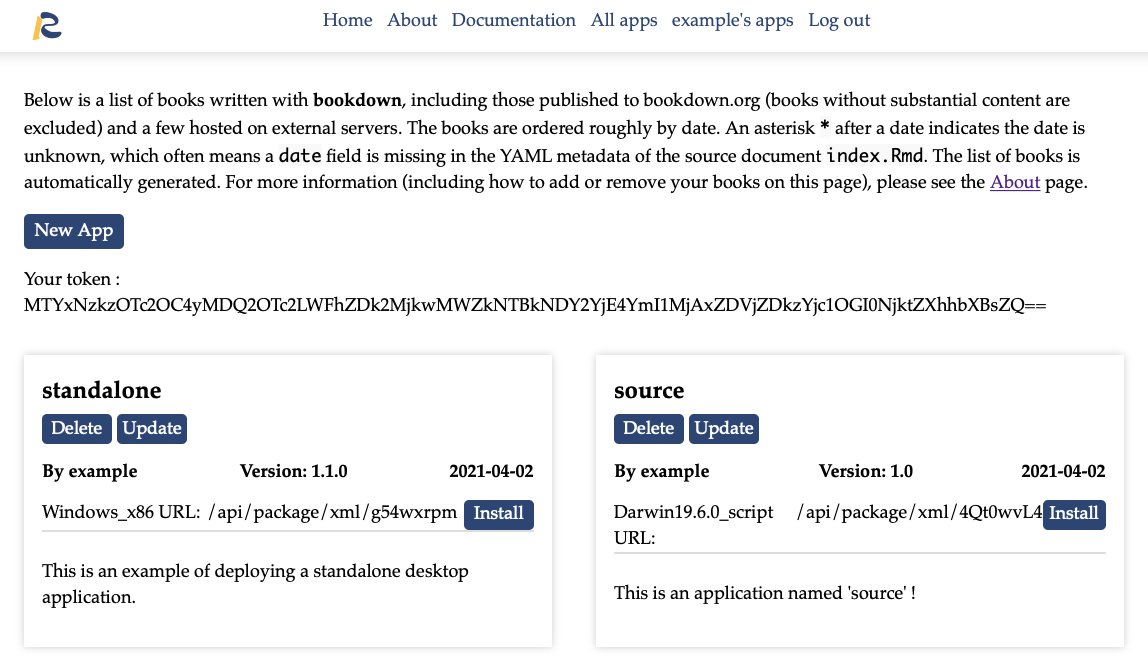
*# use deplpyApp() to distribute an app named "source"*

*#"path\_shiny" is set as your app directory.*

*# for each time to deploy an app, the parameter "username" cannot be omitted.*

*localshiny::deployApp(username="example",project="path\_shiny",name="source")*

Congratulations! You’ve deployed your app named ”source”.



**2.5 Delete apps**

You can also mange your app deployed on the website. Delete a currently deployed app with with *localshiny::deleteApp().*To use it, run

*# make sure that you have connected the package to your account*

*# localshiny::login("example", token=example\_token)*

*# if you want to delete app "source"*

*localshiny::deleteApp("source")*

**2.6 Update apps**

Your account can upload no more than ten apps. If you make changes to your code or app information, you can update it instead of deploying a new one. Notice that you cannot change your app name. If you have to do that, you need to delete it and deploy a new one.

We recommend you to use *localshiny::deployApp()*to update apps. In the function, set the app name as same as that of the application deployed before, reset the app information, and then you will update it.

#*“path\_shiny” is set as your app directory.*

*# update your app, you need to set the name as same as before*

*localshiny::deployApp(username="example",project="path\_shiny",name="source")*

*# change the app name , you need to deploy the app with a new name*

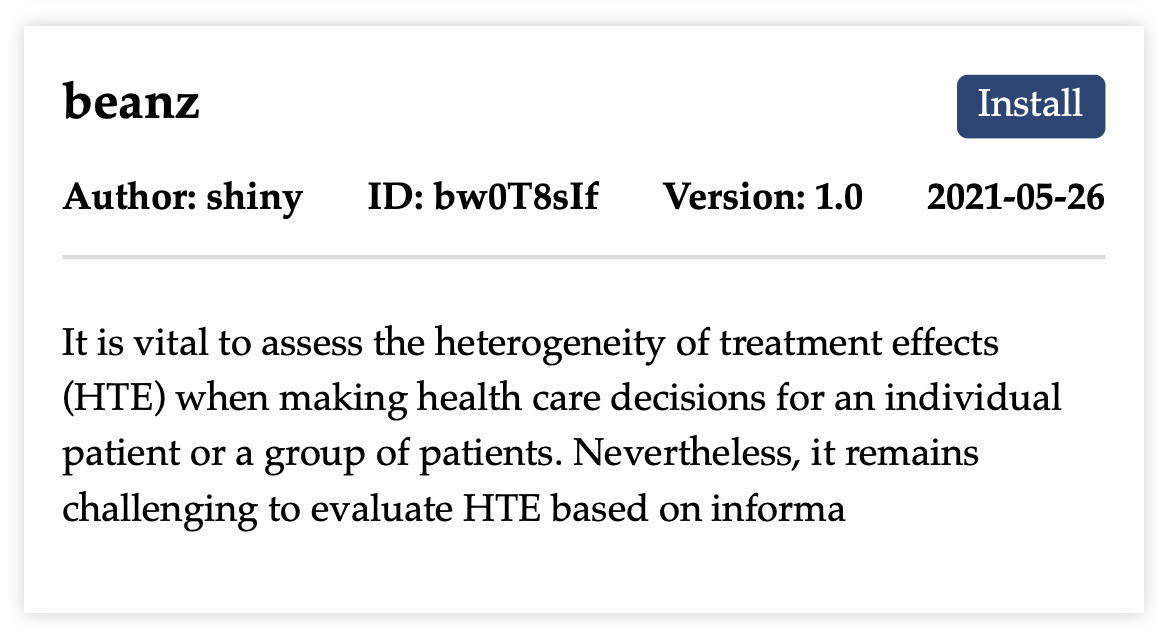
*localshiny::deployApp(username="example",project="path\_shiny",name="source1")*

**3. Install apps on local computers**

This section will go step-by-step on how to install a Shiny app published on the LocalShiny website via *localshiny*. Users do not need to know Shiny and they only need to call the function *localshiny::installApp()* to install apps on their local machine. If they want to launch Shiny apps installed from their R session, they are required to have R and Shiny installed on their computers.

1) Obtain an app ID

Users can open a web browser and go to the LocalShiny web. Visit the gallery of developer-submitted Shiny apps, and click through displayed apps to discover one they have an interest in. Copy the app ID on the app’s panel and then install the app associated with the app ID.



*# copy the appID*

*appID = "bw0T8sIf"*

2) Download and install an app

By calling the function *localshiny::installApp()*, you **can download the Shiny files of interest from the website, restore the app on your local machine with the information written in the lock file, and launch it by running R scripts. The function for installation can** automates all of these processes by running the following command:

*# set the appPath on which app files will be downloaded*

*mypath ="~/project/install/"*

*localshiny::installApp(appID =appID, appPath=mypath)*

1. Launch the app

Set the working directory at the app directory. Run the R scripts and then R will start an interactive session.

*setwd(paste(mypath, appID, sep="/"))*

*source("runscript.R")*

You’ve worked through the entire Shiny installation process.

**4. Links**

LocalShiny web： <https://www.localshiny.org>

localshiny package : https://github.com/localshiny/localshiny

**5. Reference**

[1] Chang W, Cheng J, Allaire J, Xie Y, McPherson J. Shiny: Web application framework for R. 2016. Available: http://CRAN.R-project.org/package=shiny