# Scaling R Shiny Apps to Multiple Concurrent Users in a Secured HPC Environment Using Open OnDemand

Ohio Supercomputer Center: Eric Franz (efranz@osc.edu), Morgan Rodgers, Trey Dockendorf, Alan Chalker, Doug Johnson, David E. Hudak
Biomedical Informatics Shared Resource, The Ohio State University: Venkat S. Gadepalli, Maciej Pietrzak, Hatice G. Ozer, Amy Webb

## Introduction

### What is Open OnDemand?

Open OnDemand is an open source project to provide web based access to HPC resources. OnDemand's "Interactive Apps" support launching web applications like Jupyter and RStudio on cluster compute nodes, proxying HTTP requests to those apps, and secure user separation of both app and data access enforced at the OS level.

### What is R Shiny?

R is a programming language for statistical computing and graphics. Many libraries contributed by users have transformed R into an ideal tool for data wrangling, analysis, and as well visualization.

Shiny is a web application R package which empowers research and data scientists to deploy their analysis results in an interactive environment without requiring the expertise of a software developer.

### Client Need

Staff at the Biomedical Informatics Shared Resources (BISR) at The Ohio State University work with researchers and clinicians at the Wexner Medical Center supporting high throughput high dimensional biological data analysis needs such as Next Generation Sequencing (NGS) data. However, the gigabytes of data from this technique offers challenges in storage, computational resources, analysis and interpretation. Moreover, the solutions and advances in information technology working towards solving big data challenges are not widely adopted by the research and clinical community. Hence it is important to leverage existing open source tools and build customized tools for specific researcher to view and interact with genomics data and analysis results.

### Project Requirements

BISR's RNA sequencing analysis pipelines run on OSC clusters involves:

(1) Assessing the raw RNA sequencing data for their quality using software's tools such as FASTQC, MULTIQC.

(2) Cleaning the data as required using tools such as Cutadapt or Trimmomatic.

(3) Aligning the tidy data to Ensembl's GRCh38 reference genome using HISAT2 program.

(4) Assessing the quality of alignment RSeQC and picard.

(5) Generating a single .Rdata object using custom shell and R scripts from multiple data files produced by the pipeline for use in a web based interactive report using Shiny.

As a result of running these pipelines, BISR has multiple datasets that are already at OSC and new datasets are produced. BISR plans to build multiple interactive report apps using Shiny to enable research analysts to analyze the datasets. Hence, it made sense to build a visualization solution hosted at OSC. The requirements for the Shiny visualization solution are:

(1) Provide a high-level interface for users (both the BISR admin user and the analyst user).

(2) Analysts only see datasets they are granted access to.

(3) Analysts only see apps they are granted access to.

(4) Analysts can launch a Shiny app with one of several approved datasets.

### Solution Overview

The Shiny visualization service for BISR at OSC supports two workflows: the analyst workflow and the admin workflow.

- Analyst workflow enables a user to launch and connect to a Shiny app through OnDemand.
- Admin workflow enables a user to grant another user access to datasets and Shiny apps using a custom admin tool through OnDemand.
- Shiny apps run in the context of an HPC batch job.

### Benefits

The cost of supporting this service using OnDemand is lower than building or managing a stand-alone solution:

- An existing OnDemand installation at an HPC center will already have built solutions for authentication, authorization, accounting, and monitoring that can be leveraged by any OnDemand app.
- This approach leverages the HPC batch scheduler and its use of containerization in batch jobs to isolate and control the Shiny processes and OS file system permissions to control access to apps and data.
- This approach succeeds in shifting the responsibility for ensuring the production availability of the service from BISR to OSC, which enables BISR to focus more on its domain of work.
- BISR can keep all of the data produced by their NGS pipelines at OSC, deploying visualization options alongside datasets.

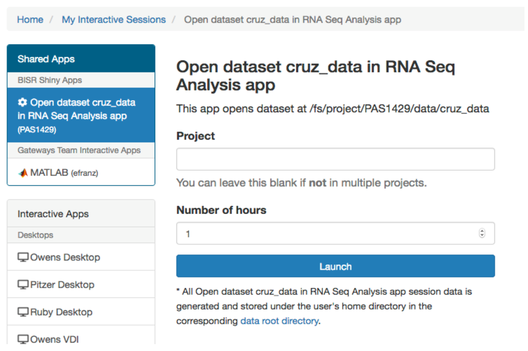
## Solution

### Analyst Workflow

The analyst workflow enables a user to launch a Shiny app through OnDemand:

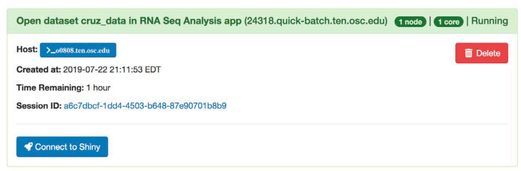
(1) The user logs into OnDemand and selects the "**Shiny launcher**" app from the dashboard menu that the user wants to start (there is one launcher app per Shiny app).

(2) The user is presented a web form (seen below) to configure the launcher by specifying the wall time of the session or batch job which defaults to an hour, and selects the dataset to load if multiple datasets are available to load. Then the user submits the form.



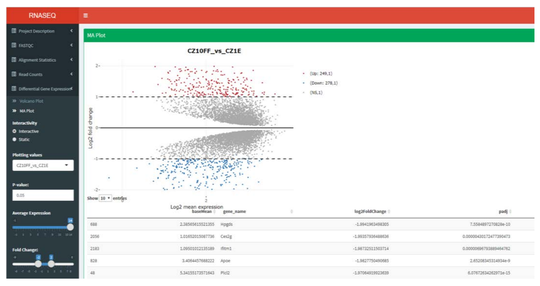

(3) A batch job is submitted and rapidly starts on OSC's "Quick batch" cluster. The status and id of the job are presented to the user on a job card.

(4) When the job start the server, a connect button appears in the OnDemand when connection information is available.




(5) The user clicks the connect button and a new browser window opens, connecting the user to the Shiny app, running on a compute node.

(6) The user performs the analysis using the app.




### Admin Workflow

The admin workflow enables a user to grant another user access to datasets and Shiny apps. By selecting a user, dataset, and app, the admin user can create a mapping between the three, with the goals to enable user access to the specified dataset and app, and to specify that the dataset should be loadable by the chosen app.

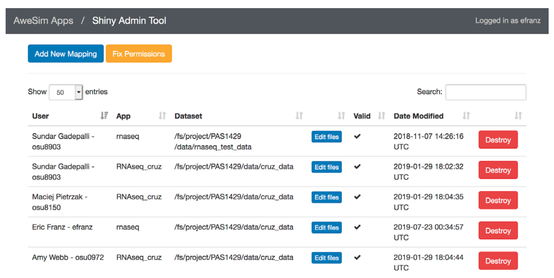
(1) The admin user logs into OnDemand

(2) Selects the "Shiny Admin" app from the dashboard

(3) Clicks the "Add new mapping" button

(4) The admin user is presented with a list of users (analysts), datasets, and Shiny launcher apps to choose from. The admin user clicks Save to save the mapping

(5) Clicks the "Fix Permissions" button to modify the permissions of the apps and datasets to reflect the modified mappings list.




### Shiny Report App

- We used **packrat** as a dependency manager to handle the many libraries and dependencies required to facilitate deployment of Shiny apps across different operating systems (Windows, MacOS, Linux) to avoid conflicts.
- We used the concepts of Shiny modules and Shiny reactive programming to modularize app functionality which made our code more manageable and extendable to other apps.
- We incorporated a JSON-based configuration file to customize the Shiny user-interface (UI) contents. This allows an admin or analyst to customize their Shiny UI without expertise in Shiny programming.

## Implementation

### Authorization

OnDemand maps authenticated requests to system accounts, requiring each authenticated user to have their own unique account. To handle an incoming request, OnDemand launches a per user NGINX web server on the OnDemand web node and reverse-proxies the request to that server. Each Per User NGINX (PUN) web server provides a secure per-user endpoint for communication with the browser. The PUN is configured using the Passenger application server to serve web applications in Ruby, Node, and Python running on the web node. Two OnDemand Passenger apps facilitate the Shiny visualization service for BISR.

The first is the OnDemand dashboard app, which provides a launch interface for accessing other apps. BISR admins use the launch interface to access the admin tool for managing access to Shiny apps and datasets, and BISR users use the launch interface to start a Shiny visualization session through OnDemand's interactive app plugin feature.

The second OnDemand passenger app that facilitates the Shiny visualization service is the admin tool. Manually ensuring that the permissions of the apps and the datasets are properly set is prone to error. The admin tool addresses this problem by providing admins a web interface to correctly set permissions to the Shiny apps and datasets deployed to BISR's project space directory.

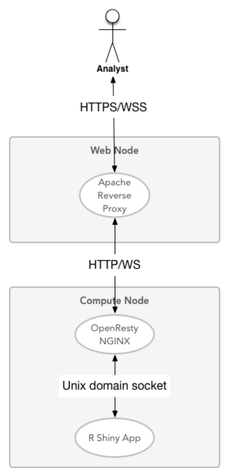
### Scalability

A user starts a Shiny app by submitting a batch job using the OnDemand dashboard. The HPC batch scheduler is responsible for ensuring the resources available for the Shiny apps are not overloaded. If too many users attempt to launch a Shiny app, some jobs may be queued till resources become available. Thus, the number of concurrent running Shiny apps scales with the number of jobs the HPC batch scheduler can concurrently run. Utilizing the HPC batch scheduler means that Shiny app jobs will still compete with other jobs for limited system resources. Queue times introduced by this competition are undesirable for interactive apps like the Shiny apps, where there is a need for deterministic and immediate start time.

To address this problem OSC has allocated resources for interactive apps by creating a dedicated batch environment called "Quick batch" that has a small number of compute nodes that execute interactive app jobs exclusively. Characteristics of the Quick batch environment include extremely short scheduler execution intervals, oversubscription of processors, but dedicated access to memory. The end result is users can get access to running dedicated instances of the Shiny apps in under a minute.

### Middleware

OnDemand provides a reverse proxy to web and VNC servers running on compute nodes. The OnDemand reverse proxy is only accessible by authenticated OnDemand users, but OnDemand does not enforce any extra authorization to ensure only the user who started the Shiny app's web server can connect to it. As a result, each server started on a compute node by a batch job needs to provide authentication to ensure that the server only accepts requests from the user that started it.




We introduced a custom middleware server to start alongside the R Shiny app (as shown at right). The custom middleware server is an OpenResty NGINX server that has custom Lua code to provide basic authentication and location blocks to handle reverse proxying HTTP and WebSocket requests to the Shiny app. The OpenResty NGINX package is installed in a Singularity image which made it easy to test and deploy this middleware solution without installing the OpenResty NGINX package on OSC's compute nodes. One of the benefits of using Singularity to package this middleware solution is we have the freedom to replace this with a better middleware solution for handling authentication for these types of apps.

### Shiny Launcher

The Shiny Launcher allows users to specify the job's walltime limit in number of hours and a project to bill against, a requirement for some users at OSC to provide when submitting jobs. The app fetches from the admin tool's mappings database the path to the dataset shared with the user that is intended to be loaded by this app. If multiple datasets can be loaded, a dropdown appears allowing users to select which dataset to load. When the user clicks the launch button a batch job is submitted, with the dataset specified as an environment variable in the job so that when the Shiny app starts it can determine what dataset to load by inspecting this environment variable. When the job starts it

(1) starts the Shiny app listening on a Unix domain socket owned by the user,

(2) generates an NGINX config for the custom middleware,

(3) starts the NGINX middleware with this config using a Singularity image that contains an installation of the OpenResty NGINX package, listening on an open port.

Once the NGINX server starts, the job writes a connection.yml file with host, port, and password of the NGINX server to the job's working directory which is mounted via NFS on both the compute node and the web node. This enables the OnDemand dashboard to build the connection button so the user can connect to the server.

## Future Work

### Status

We are in the evaluation phase of this project. The admin tool, Shiny launcher and Shiny app RNASeq (screenshot below), an interactive genomics data analysis report, are all deployed and in use by the six members of the BISR team with 8 projects, where each project may contain multiple datasets. BISR is continuing to refine the RNASeq app. In the next 18 months BISR will be developing a second Shiny app to support around 18 projects with 21-33 end users targeted.

### Related Work

The RStudio company provides several commercial offerings to support deploying Shiny apps in self hosted production environments:

- Shiny Server Pro provides a dashboard for launching Shiny apps and the ability to authorize users for access to specific apps. BISR first considered self hosting Shiny Server or Shiny Server Pro but were concerned about the cost of setting up authentication and authorization properly for their needs. Authorizing dataset access is a problem Shiny Server does not address. OSC also considered utilizing the open source version of Shiny Server in OnDemand to launch and proxy to the Shiny apps. However, the availability of Singularity on OSC systems made using NGINX as a reverse proxy to the Shiny library's basic web server easier to implement and likely more performant.
- The new RStudio Connect enables multiple users to use RStudio IDE to publish apps, R Markdown documents along with other document formats and may be a viable commercial alternative to this solution.
- Kubernetes or other container orchestration software could be a viable alternative to OnDemand for supporting launching multiple Shiny R applications.

### Future Scalability Work

Improving the performance of OnDemand on a single web node and adding support for horizontal scaling of OnDemand across multiple web nodes would address these problems:

- The number of users that can concurrently attempt to start a Shiny app are restricted by the number of users that can concurrently submit a job through the OnDemand dashboard.
- The number of concurrent users submitting jobs through a single OnDemand instance is limited to the order of 100s because each user is provided their own dedicated NGINX web server.
- The OnDemand PUN is configured to shut down Passenger apps after 5 minutes of inactivity, so this can help to work around this constraint by enabling more users to start Shiny apps while others are currently using them.

### Future Scheduler Work

OnDemand's dependence on an HPC batch scheduler and OS accounts for each user introduce some challenges:

- It can be problematic if Shiny app batch jobs exceed the HPC resources available as jobs will be queued and access to a Shiny app visualization session may not be immediate.
- Without dedicated hardware, Shiny jobs may compete for compute time with other jobs on the same node
- Installing this Shiny visualization service in a non-HPC environment may not be straightforward as it would require the installation of a batch scheduler such as Slurm.

Enabling interactive work without the need for an HPC batch scheduler in OnDemand will address these problems and is the subject of future work.

### Other Potential Future Work

The OnDemand platform is lacking a few features that would have made this work easier:

- An integrated DSL or app with a declarative approach to specifying permissions, including access control lists, may have reduced the cost to building the custom admin tool.
- A fast and reliable abstraction for securing TCP communication between the web node and servers running on the compute nodes would eliminate the need to provide the OpenResty custom middleware solution.

This solution does not provide the ability to offer public anonymous access to running Shiny apps. Since user isolation is the primary feature of OnDemand the development roadmap does not include adding support for anonymous public access to data and apps at this time.

### Acknowledgments

This work is supported by the National Science Foundation of the United States under the awards NSF SI2-SSE-1534949 and CSSI-Frameworks-1835725.

Ohio Supercomputer Center. 1987. Ohio Supercomputer Center. Columbus OH: Ohio Supercomputer Center. http://osc.edu/ark:/19495/f5s1ph73.

### Additional Information

OSC maintains a program website at

https://openondemand.org