Getting Started
This guide introduces VICE, a visual and interactive computing environment which is the latest feature in CyVerse’s Discovery Environment (DE) for running interactive apps. This guide instructs users on how to do basic functions of VICE, e.g., how to launch interactive apps, run your analysis on interactive apps, build interactive apps etc.
VICE which stands for Visual Interactive Computing Environment is a component of CyVerse’s data science workbench, called the Discovery Environment (DE). VICE allows users to launch web-based applications from the DE. Once an application is launched, users can access the VICE app through a linked URL, they can bring their data from the CyVerse Data Store into the application, or pull data from anywhere on the web using standard requests (curl wget git).
1.1 What is the big idea?

The CyVerse DE hosts a large number of Command Line (CLI) applications for researchers to perform Bioinformatic and data analysis. **VICE** introduces graphic user interfaces (GUIs) and common Integrated Development Environments (IDEs) such as Project Jupyter Notebooks & Lab, Rstudio, Shiny Apps and Linux Desktop. 

**VICE** allows exploratory bioinformatic and geoinformatic data analysis as well as the ability to run ad-hoc scripts. By working on an advanced, scalable cyberinfrastructure, you are essentially moving your algorithms and code to the center where data and compute live. CyVerse operates on an Internet2 backbone, meaning your data are transferred at rates which likely exceed your local ISP or campus network.
1.2 How is VICE different from other DE apps?

The current apps in the DE are non-interactive, meaning the user selects parameters and data for a particular analysis, and submits the job for execution on platforms (Condor, HPC via Agave). When the process completes, the user is notified and they can view their analysis results in a folder. Any desired changes in results requires the user to change analysis parameters and run the job again to full completion. But exploratory data analysis (EDA) requires user to click and interact with running applications (i.e. Data Scientists need a Workbench). Availability of computational notebooks (Jupyter, Zeplin) and Rstudio’s Shiny allow users to readily share analysis in a reproducible manner and technologies like Javascript, WebGL, and others are making the web browser an extremely capable workbench.

VICE lets users interact with their data and do analyses in their favorite programming language in one place in an iterative way. Researchers can now explore their datasets interactively by easily changing parameters of selected analysis applications without having to download data from storage to an active workspace.

1.3 Conclusion

VICE combines the cyberinfrastructure power of CyVerse with the interactivity of modern web-based IDE and data science software, allowing researchers to work seamlessly with their data within a single, high-performance data science workbench.
In order to use VICE, you will need access to the following services/software

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Preparation/Notes</th>
<th>Link/Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyVerse account</td>
<td>You will need a CyVerse account to complete this exercise</td>
<td>Register</td>
</tr>
<tr>
<td>Discovery Environment access</td>
<td>You will get access to Discovery Environment by default. Check the ‘My Services’ tab; if Discovery Environment is not listed, click the ‘Available’ tab, locate Discovery Environment and click the link to request access.</td>
<td>CyVerse User Portal</td>
</tr>
</tbody>
</table>

Fix or improve this documentation:

- On Github: Repo link
- Send feedback: Tutorials@CyVerse.org
Currently, VICE apps are categorized broadly into three different spaces:

1. Integrated Development Environments (Jupyter Lab and RStudio)
2. Interactive apps (Shiny, WebGL, HTML5)
3. Virtual Desktop Environment (w/ Apache Guacamole, VNC, XPRA)

Each of these serve a different data science purpose.

- If you are interested in writing Python, Julia, Spark, C++ based data analysis or visualization, Jupyter is most appropriate.
- If you are interested in creating analyses written in R, RStudio would be more appropriate.
- If you have a pre-built analysis such as Shiny built in R, Shiny is used.
- If you want to work in other Linux-based software, you can launch a virtual desktop (KDE, Mate, XFCE4, Gnome, etc) and work with your tools interactively.

### 3.1 1. What is JupyterLab?

**JupyterLab** is an interactive development environment for working with notebooks, code and data.

JupyterLab enables you to use text editors, terminals, file viewers, and other custom components side-by-side with notebooks in a tabbed work area. JupyterLab provides a high level of integration between notebooks, documents, and activities:

- Drag-and-drop to reorder notebook cells and copy them between notebooks
- Run code blocks interactively from text files (.py, .R, .md, .tex, etc.)
- Link a code console to a notebook kernel to explore code interactively without cluttering up the notebook with temporary scratch work
- Edit popular file formats with live preview, such as Markdown, JSON, CSV, Vega, VegaLite, and more
3.1.1 1.1 What is a Jupyter Notebook?

The Notebook (formerly IPython Notebook) is Project Jupyter’s flagship project for creating reproducible computational narratives.

It enables users to create and share documents that combine live code with narrative text, mathematical equations, visualizations, interactive controls, and other rich output.

Notebook documents (or “notebooks”) are documents produced by the Jupyter Notebook App, which contains both computer code (e.g., python, r, julia) and rich text elements (paragraph, equations, figures, comments, images, links, etc.).

3.1.2 1.2 JupyterLab VICE

The apps are based on official integrated into the CyVerse Discovery Environment (DE).

3.2 2. What is RStudio?

RStudio is a free and open source integrated development environment for R, a programming language for statistical computing and graphics. Some of its features include:

- Customizable workbench with all of the tools required to work with R in one place (console, source, plots, workspace, help, history, etc.)
- Syntax highlighting editor with code completion
- Execute code directly from the source editor (line, selection, or file)
- Full support for authoring Sweave and TeX documents
- Runs on all major platforms (Windows, Mac, and Linux) and can also be run as a server, enabling multiple users to access the RStudio IDE using a web browser

3.2.1 2.1 RStudio VICE

The apps are based on official.

3.3 3. What is Shiny?

Shiny is an open source R package that provides an elegant and powerful web framework for building web applications using R. Shiny helps you turn your analyses into interactive web applications without requiring HTML, CSS, or JavaScript knowledge. Some of its features include:

- Build useful web applications with only a few lines of code—no JavaScript required
- Shiny applications are automatically “live” in the same way that spreadsheets are live. Outputs change instantly as users modify inputs, without requiring you to reload your browser
- Shiny user interfaces can be built entirely using R, or can be written directly in HTML, CSS, and JavaScript for more flexibility
- Works in any R environment (Console R, Rgui for Windows or Mac, ESS, StatET, RStudio, etc.)
- Attractive default UI theme based on Twitter Bootstrap
• A highly customizable slider widget with built-in support for animation
• Pre-built output widgets for displaying plots, tables, and printed output of R objects
• Fast bidirectional communication between the web browser and R using the websockets package
• Uses a reactive programming model that eliminates messy event handling code, so you can focus on the code that really matters
• Develop and redistribute your own Shiny widgets that other developers can easily drop into their own applications

3.3.1 3.1 Shiny VICE

The apps are based on official.

3.4 4. What is Ubuntu Desktop?

The Ubuntu Desktop has a full Guacamole installation and Ubuntu XFCE desktop. This allows users to have a simple all-in-one desktop through their web browser. Users can run any interactive or visualization tool that can run on the most recent Linux distros. Solutions to support the inevitable array of Linux applications that users will want. Potential options include:

• Separate image per application
• Network fs (e.g. NFS, Ceph, etc) containing all applications
• Per-application network fs
• On-demand installation of application via script/ansible

3.4.1 4.1 Ubuntu Desktop VICE

Linux Desktops using Apache Guacamole, and Xpra have been integrated into the DE.

Fix or improve this documentation:

• On Github: Repo link
• Send feedback: Tutorials@CyVerse.org
4.1 1. Search a Jupyter Lab

First log-in CyVerse DE

After you login to DE, open the Apps window and search for Jupyter Lab with key word *Jupyter* or *JupyterLab*.
4.2 2. Launch analysis

Launch the Jupyter Lab app by clicking **launch analysis**. Before you launch, you can either drag and drop or browse the files that you want to use with Jupyter. There is currently no restriction of how many files and size of the files that can be launched along with JupyterLab app.

**Note:** The input files and/or folders can be selected under the ‘Parameters’ tab.

**Tip:** If you have a working Jupyter workbook, you can import it into the app using input files and/or folder.

4.3 3. Navigate to JupyterLab url

Unlike regular DE apps once the analysis starts running you will get an url. Click on your notifications, and then by clicking on the “Access your running Analysis here” url (check that it is the jupyter app) you will be redirect to a page with a welcome message.
Important: The app might take a little bit to get everything set up. In the meantime you will see the welcome page.

4.3. Navigate to JupyterLab url
You will be automatically redirected to the app from the welcome page after it finishes setting up.

The Jupyter Lab Interface: Jupyter Lab provides flexible building blocks for interactive, exploratory computing. While Jupyter Lab has many features found in traditional integrated development environments (IDEs), it remains focused on interactive, exploratory computing. The Jupyter Lab interface consists of a main work area containing tabs of documents and activities, a collapsible left sidebar, and a menu bar. The left sidebar contains a file browser, the list of running kernels and terminals, the command palette, the notebook cell tools inspector, and the tabs list.

More information about the Jupyter Lab can be found here.

4.4 4. Create Jupyter notebook

Jupyter notebooks are documents that combine live runnable code with narrative text (Markdown), equations (LaTeX), images, interactive visualizations and other rich output. Jupyter notebooks (.ipynb files) are fully supported in JupyterLab.

If you want to create a notebook, you can do so by clicking the + button in the file browser and then selecting a kernel in the new Launcher tab. Currently there are 3 different notebooks available - Python3, Julia and R. Click on Python 3 under Notebook section in the JupyterLab Interface, which will open a new Jupyter Notebook. A new file is created with a default name. Rename a file by right-clicking on its name in the file browser and selecting “Rename” from the context menu.

To know more about notebooks in JupyterLab click here

Tip: To open the classic Notebook from Jupyter Lab, select “Launch Classic Notebook” from the Help Menu.
Note: There are plenty other cool stuff that you can do in Jupyter Lab such as using consoles, using terminal and using text editor.

4.5 5. Write your code

Once you open a new notebook, you can start writing your code, put markdown text, generate plots, save plots etc.

4.6 6. Complete and Save Outputs

After finishing your analysis, you can save outputs to data store by clicking the Analysis window, then select the VICE analysis that you are running and select Complete and Save Outputs under the “Analyses” button.
After you had done this, you can find the outputs that you generated (if any) using the same steps as before, but this time selecting ‘Go To Output Folder’.

**Warning:** Currently, VICE can run for 48 hrs beyond which the apps will be terminated. If you have opted for email notifications from DE, then you’ll get a notification 1 day before and 1 hour before the app gets terminated. If you want to extend the time, you need to login to [http://cyverse.run](http://cyverse.run), find your analysis and then click the hour glass which automatically extends the app run time to 3 more days.

Fix or improve this documentation

- On Github: [Repo link](#)
- Send feedback: Tutorials@CyVerse.org

## 4.7 7. Jupyter-lab with SQL

Now you can run SQL queries inside a notebook. Here is a quick launch
RStudio is a free, open source IDE (integrated development environment) for R. Its interface is organized so that the user can clearly view graphs, data table, R code and output at the same time. It also offers an Import-Wizard-like feature that allows users to import CSV, Excel, SAS (.sas7bdat), SPSS (.sav), and Stata (*.dta) files into R without having to write the code to do so.

5.1 1. Running Rstudio App

First log-on CyVerse DE, open the Apps window and find the Rstudio Lab app by searching for ‘rstudio’.
5.2 2. Launch analysis

Launch the Rstudio app by selecting an example folder and then clicking launch analysis. You can select different input files and/or folder.

Tip: You can use input files to import a script into the app.
To use a folder as an input...

**Note:** You will not see any files when selecting the folder if you selected input by folder. Rest assured that they will be there once the app begins to run.

Launch the analysis after you are finished selecting the input files (if any).
5.3 3. Navigate to rstudio app url

After the analysis starts running, open your notifications and click on the ‘Access your running Analysis here url’.
5.4 4. Launch Rstudio

In the new URL, enter ‘rstudio’ for both username and password
5.5 5. Write/Run your code

In the Rstudio script section, you can write your code, generate plots, save plots etc.

**Tip:** As a first step, check that the files you wanted to import are in the app. Go to the bottom right of the app, and check under ‘Files’ for your files.
5.6 6. Complete your analysis

Complete your analysis by clicking the Analysis window, then select the rstudio analysis and click ‘Complete and Save Outputs’ option under “Analyses” button.

After you had done this, you can find the outputs that you generated (if any) using the same steps as before, but this time selecting ‘Go To Output Folder’.

**Warning:** Currently, VICE can run for 48 hrs beyond which the apps will be terminated. So make sure you run your analysis before 48 hrs.

Fix or improve this documentation

- On Github: Repo link
- Send feedback: Tutorials@CyVerse.org

---

5.6. 6. Complete your analysis
Shiny is an R-based web application for interactive data analysis, exploration and visualization. It was built to execute R code dynamically based on user input. Since most JavaScript code is autogenerated by the environment, basic R scripting knowledge is sufficient for Shiny apps.

For running a Shiny App in CyVerse Discovery Environment, you need `app.R` script/file. The following are steps for running an example Rshiny App (`mpg`) that is available on the Rstudio site within the Discovery Environment.

### 6.1 1. Find your Rshiny app

Navigate to Rstudio github to find the Rshiny app that you are interested in. Here is an example of one such Rshiny app that we will be using to run in the DE.
6.2 2. Find the `app.R` script

Inside the folder for `mpg` you will find the `app.R` file. Click on the `Raw` button and then copy the contents of the `app.R` file.
6.3 3. Log-in to CyVerse DE

Next, log-in to CyVerse DE. Click on the Data window and create a folder rshiny-mpg specific for the app
6.4 4. Create new *app.R* script in DE

Under File menu, click the New R Script *app.R*, paste the contents and name it as *app.R*
6.5 5. Search Rshiny app

The Rshiny app can be launched by first searching for Rshiny app in the search bar in the Apps window.
6.6 6. Select the *app.R* script

Under Input files of the app, click the "add" button and then select the *app.R* file that you created earlier.
6.7 7. Launch Rshiny App

After you select the `app.R` file, launch the Rshiny app by clicking launch analysis
6.8 8. Navigate to Rshiny url

After the analysis starts running, click on the “Access your running Analysis here url”
Chapter 6. Shiny
6.9 9. Cancel your analysis

Cancel your analysis by clicking the Analysis window, then select the Rshiny analysis and click Cancel Analysis option under “Analyses” button
6.10 References/Tutorials

- Tutorial for those new to Shiny - https://bims.fun/r-shiny.html#fn1
- Collection of posts covering the basics of layout, reactivity and extensions for Shiny - https://bims.fun/r-shiny.html#fn2
- R-Bloggers articles featuring Shiny apps - https://bims.fun/r-shiny.html#fn4
- RStudio webinar slides for getting started with Shiny - https://bims.fun/r-shiny.html#fn5

Fix or improve this documentation:

- On Github: Repo link
- Send feedback: Tutorials@CyVerse.org
Sharing VICE apps with collaborators

You can share your running VICE workspace with colleagues (with a CyVerse account) who can see and edit your notebooks, logs, and outputs.

To share your running workspace
Chapter 7. Sharing VICE apps with collaborators
Opening workspaces shared with you
Chapter 7. Sharing VICE apps with collaborators
There are also different ways to share a VICE app without sharing a running instance of that app.

- Apps that have been made public in the Discovery Environment can be shared with the public app’s URL.
- Unpublished apps (those in your private workspace that have not yet been made public) can be shared with specific users or teams. See Sharing your App or Workflow and Editing the User Manual.
- A Quick Launch configuration can be created, and then the URL to the Quick Launch can be shared.

Creating and sharing Quick Launch configurations

Quick Launches provide a way to set default parameter values for an analysis, which can make it much easier to launch similar jobs without having to select the parameter values that the jobs have in common for every new analysis.

To create a Quick Launch, open the app as if you are going to run it, fill in any default parameters for this Quick Launch, then click the Create Quick Launch button instead of the Launch Analysis button.
Then you can name the Quick Launch and make it public.

To share the link for a public Quick Launch, or to copy its badge code for embedding in web pages or in repo markdown files, click the **Quick Launch** menu item from the app’s 3-dot menu in the **Apps window**.
Note that private Quick Launch configurations cannot be shared, and clicking on them from this listing simply opens the app launch dialog with its parameters pre-filled, according to the Quick Launch configuration.

If you don’t need to save any default inputs or parameters, you can also use the app’s public URL for sharing the VICE app, which acts the same as a Quick Launch, but it doesn’t have any parameters pre-filled.

---

**Fix or improve this documentation:**

- On Github: [Github Repo Link]
- Send feedback: Tutorials@CyVerse.org

---
CyVerse hosts many popular data science applications, e.g. Jupyter Lab, RStudio, and Shiny. These applications can be started in the Discovery Environment, and the researcher can install additional packages to the running application.

In cases where the installation may be complex, long, or require additional system administrator level access, the researcher can use the existing CyVerse container as a base image for their own new container. The researcher can add their own packages and then deploy the new app in the Discovery Environment.
Guidelines for adding interactive tools in DE

9.1 Prerequisites

Adding VICE Tools in DE is different from non-interactive Tools. VICE applications like Jupyter and RStudio run on an open port for enabling their web UI.

1. Ensure that the listen port for the web UI has a sane default and is set in the Dockerfile.
2. The working directory is set
3. All commonly needed dependencies are installed in the container image - you will not have root privileges later
4. The default user set
5. Disable any additional authentication (CyVerse provides CAS authentication and authorization).
6. URLs will work sanely behind a reverse proxy. If they don’t, you may need to add nginx to the container.

9.2 Community images as your base image

If you need to set the configurations at all (see above), you’ll need to create a new Dockerfile that uses the community-provided image as a base. Your new Dockerfile should deal with custom configurations and dependency installations.

- Jupyter Lab (https://hub.docker.com/r/cyversevice/jupyterlab-base)
- RStudio Verse (https://hub.docker.com/r/cyversevice/rstudio-verse)
- Shiny Verse (https://hub.docker.com/r/cyversevice/shiny-verse)

See some examples of VICE apps that uses community images as base image in the Dockerfile

- Patmatch (https://github.com/fomightez/patmatch-binder/tree/master/vice)
9.3 Test your Docker image

Since the images are built based using Dockerfile, make sure you test the Dockerfile before providing it to us. Dockerfile must have Entrypoint. If you cannot provide us the Dockerfile, you can request integration of the app by doing a tool request.

**Fix or improve this documentation:**

- On Github: Repo link
- Send feedback: Tutorials@CyVerse.org
Once you build your Docker image (following the guidelines), the next step is building the Tool. For this you’ll need a Docker image name, any port numbers PORT, User ID UID, working directory WORKDIR, and ENTRYPOINT.

10.1 Docker images

The Docker image of your tool is mandatory and it should be available on public registries such as Dockerhub or quay.io. Alternatively you can provide us the Dockerfile and we will build the Docker image for you. If there is no Dockerfile for the tool that you are interested in, then tell us what tool you are interesting in making us as VICE app.

10.2 Add Tool in DE

The final step in building the VICE tool is to fill up the “Add Tool” form in DE. Briefly here are the following steps.

- Log in CyVerse Discovery Environment
- Click on the Apps window and click Manage Tools button on the far right hand side of the window
- Click on Tools button and then finally Add Tools button

You’ll see a Add Tool form like this
• **Tool name** is the name of the tool. This will appear in the DE’s tool listing dialog. This is mandatory field. Eg. “jupyterlab-circos”
- **description** is a brief description of the tool. This will appear in the DE’s tool listing dialog. Eg. “Circos is a software package for visualizing data and information that was created by Martin Krzywinski”

- **version** is the version of the tool. This will appear in the DE’s tool listing dialog. This is mandatory field. Eg. “1.0”

- **Image name** is the name of the image specifier minus the image tag. The image must exist on Dockerhub or quay.io. This is mandatory field. E.g “fomightez/circos-vice”

- **Tag** is the image tag. If you don’t specify the tag, the DE will look for the “latest” tag which is the default tag.

- **Docker Hub URL** is the url of the image on the Dockerhub. E.g https://hub.docker.com/r/fomightez/circos-vice

- **Type** is the type of Tool. For VICE apps, chose “interactive”.

- **OSG Image Path** is path of the image on the OSG. You can skip this for interactive tools.

- **Entrypoint** is the Entrypoint for your tool. Entrypoint should be present in the Docker image and if not, you should specify it here.

- **Working Directory** this is the working directory of the tool and must be filled in with the value you gathered above. E.g /home/jovyan/vice

- **UID** is a number and must be filled in with the value you gathered from above. E.g 1000

- **Max CPU Cores** is the number of cores for your tool. Eg. 16

- **Memory Limit** is the memory for your tool. Eg. 64 GB

- **Min Disk Space** is the minimum disk space. Eg. 200 GB

- **Container Ports** must be a list of maps with only a single entry. The key in that entry must be container_port and should be filled in with the number value you gathered above.

  **Warning:** It is strongly recommended you do not set the bind to host as true for your added ports when creating a new App**

### 10.3 Creating VICE app for your tool

To create a new app, follow the instructions in here

**Important:** For VICE apps, be sure to check the box “Do not pass this argument to the command line” for each option you add (for VICE, this is usually just input files and folders.
Fix or improve this documentation:

- On Github: [Repo link](#)
- Send feedback: Tutorials@CyVerse.org
CyVerse provides base images of featured VICE apps here: https://github.com/cyverse-vice/

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Dockerfile</th>
<th>Quick Launch</th>
</tr>
</thead>
<tbody>
<tr>
<td>RStudio Verse</td>
<td>Rocker RStudio Verse latest</td>
<td>RStudio</td>
<td>Dockerfile</td>
<td></td>
</tr>
<tr>
<td>RStudio Geospatial</td>
<td>Rocker RStudio Geospatial latest</td>
<td>RStudio</td>
<td>Dockerfile</td>
<td></td>
</tr>
<tr>
<td>Jupyter Lab SciPy</td>
<td>Jupyter Lab w/ SciPy</td>
<td>Jupyter</td>
<td>Dockerfile</td>
<td></td>
</tr>
<tr>
<td>Jupyter Lab SQL</td>
<td>Jupyter Lab w/ SQL</td>
<td>Jupyter</td>
<td>Dockerfile</td>
<td></td>
</tr>
</tbody>
</table>

Fix or improve this documentation

- On Github: Repo link
- Send feedback: Tutorials@CyVerse.org
Frequently Asked Questions

1. What happens if my VICE app has been running for more than 48 hours?
2. Can I extend the 48hr time limit on the VICE app?
3. Can I request the CyVerse team to build the VICE app for my interactive tool if I don’t have Docker image?
4. I’m getting this error (or similar) with my docker file:

   You must set a unique PASSWORD (not ‘rstudio’) first! e.g. run with: docker run -e PASSWORD=<YOUR_PASS> -p 8787:8787 rocker/rstudio

   A: Make sure the environmental password is set for rStudio: ENV PASSWORD “rstudio1”. You can also try using this base image for rStudio: cyversevice/rstudio-base:latest or cyversevice/rstudio-verse:3.6.0. For bioconductor images, use upendradevisetty/bioconductor:1.0.

Fix or improve this documentation:

- On Github: Repo link
- Send feedback: Tutorials@CyVerse.org
CHAPTER 13

VICE Best practices

1. Smaller docker containers are better. Larger images take longer to transfer and load.
2. Use our base images for complex programs that require additional configuration files other than the Dockerfile.
3. Create robust documentation with as much metadata as possible.

Fix or improve this documentation:
• On Github: Repo link
• Send feedback: Tutorials@CyVerse.org
Tool Troubleshooting

14.1 1. Get the port

You’ll need to figure out the port that the tool uses to present its web interface. This is mandatory and you can integrate a tool without knowing the port it runs on. If you don’t know, you can find the ports that a container image exposes with this command

```bash
$ docker inspect <image-name>:<image-tag> -f '{{range $port, $val := .ContainerConfig.ExposedPorts}}{{$port}} {{end}}'
```

**Note:** Replace `<image-name>:<image-tag>` with your Docker image

It’s possible that multiple or no ports are listed. If that’s the case you’ll need to refer to the documentation for the tool to figure out the port it uses. Make a note of the port, you’ll need it later when integrating the tool in DE. Here are the tools and their ports for common tools such as Jupyter notebook, Rstudio and Shiny. If you are developing any applications based on these tools, you can use these ports while integrating the tool in DE.

<table>
<thead>
<tr>
<th>Type</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jupyter</td>
<td>8888</td>
</tr>
<tr>
<td>Rstudio</td>
<td>80</td>
</tr>
<tr>
<td>Shiny</td>
<td>3838</td>
</tr>
</tbody>
</table>

14.2 2. Get the UID of the tool’s user

You’ll need to figure out the UID of the user the app runs as. Many tools will start up as root and then use another user for the actual process, so it might take a little investigation to figure this out. To start this figure out the user that the container is configured to start up using:
If you’re lucky that will contain the numerical UID of the user. In that case you can make a note of the UID and move on. Otherwise you have more work to do. The User field can also be empty or set to the username. If its empty, then the user is root. If it’s a username then you’ll need to get the UID from inside the container.

To get the UID for a username run this:

```bash
$ docker run --rm -it --entrypoint "id" <image-name>:<image-tag> -u <username>
```

**Note:** Replace `<image-name>:<image-tag>` with your Docker image.

If the User field is empty or root, you need to be sure that the process inside the container actually runs as root. There are a few ways to check this:

- Fire up the container, exec into it, and do a `ps aux` to see the user the process is running as.

  ```bash
  $ docker run -d --name app <image-name>:<image-tag>
  $ docker exec -it app ps aux
  ```

- Print out the contents of `/etc/passwd` and check for hints:

  ```bash
  docker run --rm -it --entrypoint "cat" <image-name>:<image-tag> /etc/passwd
  ```

  **Note:** Alternatively check the documentation for the tool.

You’ll need the working directory for the process in the tool container, which may not correspond to the default working directory for the container.

To get the default working directory for the container run this:

```bash
$ docker inspect <image-name>:<image-tag> -f '{{.ContainerConfig.WorkingDir}}'
```

- If that prints out an empty string, then the default working directory is `/`
- If the container fires up as root but the tool runs as another user, then the working directory may need to be that user’s home directory.
- If the container changes to another directory after it starts up, then the working directory may need to be that directory.
- If all else fails, check the documentation and/or try out the container locally to figure out what it does.

**Important:** Keep in mind that the working directory is where the input files will be made available. Similar to UID, working directory is not mandatory but given JupyterLab’s default behavior of showing things in subdirectories of the
place it’s started. So if you’re loading notebooks and data from the Datastore, you want the working directory (where those files are loaded into the container) to be in the right spot.

Make a note of the working directory, you’ll need it later when putting together the JSON for the tool and app.
CHAPTER 15

Contact

15.1 Intercom

15.2 Ask Forum

- You can post your questions at Ask forum http://ask.cyverse.org

Fix or improve this documentation:

- On Github: Repo link
- Send feedback: Tutorials@CyVerse.org

Learning Center Home

Fix or improve this documentation

- On Github: Repo link
- Send feedback: Tutorials@CyVerse.org

Learning Center Home