- Home
- Help
- Docker for Bioconductor

Docker containers for Bioconductor

Docker packages software into self-contained environments, called containers, that include necessary dependencies to run. Containers can run on any operating system including Windows and Mac (using modern Linux kernels) via the Docker engine.

Containers can also be deployed in the cloud using Amazon Elastic Container Service, Google Kubernetes Engine or Microsoft Azure Container Instances

Quick start

1. Install Docker

2. Run container with Bioconductor and RStudio

```
docker run \
    -e PASSWORD=bioc \
    -p 8787:8787 \
```

```
bioconductor/bioconductor_docker:devel
```

This command will run the docker container `bioconductor/bioconductor_docker:devel` on your local machine.

RStudio will be available on your web browser at `http://localhost:8787`. The USER is fixed to always being `rstudio`. The password in the above command is given as `bioc` but it can be set to anything. `8787` is the port being mapped between the docker container and your host machine. NOTE: password cannot be `rstudio`.

The user is logged into the `rstudio` user by default.

Why use Containers

With Bioconductor containers, we hope to enhance

- **Reproducibility**: If you run some code in a container today, you can run it again in the same container (with the same tag) years later and know that nothing in the container has changed. You should always take note of the tag you used if you think you might want to reproduce some work later.
- **Ease of use**: With one command, you can be running the latest release or devel Bioconductor. No need to worry about whether packages and system dependencies are installed.
- **Convenience**: Easily start a fresh R session with no packages installed for testing. Quickly run an analysis with package dependencies not typical of your workflow. Containers make this easy.

Our aim is to provide up-to-date containers for the current release and devel versions of Bioconductor, and some older versions. Bioconductor's Docker images are stored in Docker Hub; the source Dockerfile(s) are on Github.

Our release images and devel images are based on the Rocker Project - rocker/rstudio image and built when a Bioconductor release occurs.

Goals for new container architecture

A few of our key goals to migrate to a new set of Docker containers are,

- to keep the image size being shipped by the Bioconductor team at a manageable size.
- easy to extend, so developers can just use a single image to inherit and

build their docker image.

- easy to maintain, by streamlining the docker inheritance chain.
- Adopt a “best practices” outline so that new community contributed docker images get reviewed and follow standards.
- Adopt a deprecation policy and life cycle for images similar to Bioconductor packages.
- Replicate the Linux build machines (*malbec2*) on the `bioconductor/bioconductor_docker:devel` image as closely as possible. While this is not fully possible just yet, this image can be used by maintainers who wish to reproduce errors seen on the Bioconductor Linux build machine and as a helpful debugging tool.

Current Containers

For each supported version of Bioconductor, we provide

- **bioconductor/bioconductor_docker:RELEASE_X_Y**
- **bioconductor/bioconductor_docker:devel**

Bioconductor’s Docker images are stored in Docker Hub; the source Dockerfile(s) are in Github.

Deprecation Notice

For previous users of docker containers for Bioconductor, please note that we are deprecating the following images. These images were maintained by Bioconductor Core, and also the community.

Legacy Containers

These images are NO LONGER MAINTAINED and updated. They will however be available to use should a user choose. They are not supported anymore by the Bioconductor Core team.

Bioconductor Core Team: bioc-issue-bot@bioconductor.org

- bioconductor/devel_base2
- bioconductor/devel_core2
- bioconductor/release_base2

- bioconductor/release_core2

Steffen Neumann: sneumann@ipb-halle.de, Maintained as part of the "PhenoMeNal, funded by Horizon2020 grant 654241"

- bioconductor/devel_protmetcore2
- bioconductor/devel_metabolomics2
- bioconductor/release_protmetcore2
- bioconductor/release_metabolomics2

Laurent Gatto: lg390@cam.ac.uk

- bioconductor/devel_mscore2
- bioconductor/devel_protcore2
- bioconductor/devel_proteomics2
- bioconductor/release_mscore2
- bioconductor/release_protcore2
- bioconductor/release_proteomics2

RGLab: wjiang2@fredhutch.org

- bioconductor/devel_cytometry2
- bioconductor/release_cytometry2

First iteration containers

- bioconductor/devel_base
- bioconductor/devel_core
- bioconductor/devel_flow
- bioconductor/devel_microarray
- bioconductor/devel_proteomics
- bioconductor/devel_sequencing
- bioconductor/devel_metabolomics
- bioconductor/release_base
- bioconductor/release_core
- bioconductor/release_flow
- bioconductor/release_microarray
- bioconductor/release_proteomics
- bioconductor/release_sequencing
- bioconductor/release_metabolomics

Reason for deprecation

The new Bioconductor Docker image `bioconductor/bioconductor_docker` makes it possible to easily install any package the user chooses since all the system

dependencies are built in to this new image. The previous images did not have all the system dependencies built in to the image. The new installation of packages can be done with,

```
BiocManager::install(c("package_name", "package_name"))
```

Other reasons for deprecation:

- the chain of inheritance of Docker images was too complex and hard to maintain.
- Hard to extend because there were multiple flavors of images.
- Naming convention was making things harder to use.
- Images which were not maintained were not deprecated.

Reporting Issues

Please report issues with the new set of images on GitHub Issues or the Bioc-devel mailing list.

These issues can be questions about anything related to this piece of software such as, usage, extending Docker images, enhancements, and bug reports.

Using the containers

A well organized guide to popular docker commands can be found here. For convenience, below are some commands to get you started. The following examples use the `bioconductor/bioconductor_docker:devel` image.

Note: that you may need to prepend `sudo` to all `docker` commands. But try them without first.

Prerequisites: On Linux, you need Docker installed and on Mac or Windows you need Docker Toolbox installed and running.

List which docker machines are available locally

```
docker images
```

List running containers

```
docker ps
```

List all containers

```
docker ps -a
```

Resume a stopped container

```
docker start <CONTAINER ID>
```

Shell into a running container

```
docker exec -it <CONTAINER ID> /bin/bash
```

Shutdown container

```
docker stop <CONTAINER ID>
```

Delete container

```
docker rm <CONTAINER ID>
```

Delete image

```
docker rmi bioconductor/bioconductor_docker:devel
```

Running the container

The above commands can be helpful but the real basics of running a Bioconductor Docker involves pulling the public image and running the container.

Get a copy of public docker image

```
docker pull bioconductor/bioconductor_docker:devel
```

To run RStudio Server:

```
docker run -e PASSWORD=<password> \
        -p 8787:8787 \
        bioconductor/bioconductor_docker:devel
```

You can then open a web browser pointing to your docker host on port 8787. If you're on Linux and using default settings, the docker host is `127.0.0.1` (or `localhost`, so the full URL to RStudio would be `http://localhost:8787`). If you are on Mac or Windows and running `Docker Toolbox`, you can determine the docker host with the `docker-machine ip default` command.

In the above command, `-e PASSWORD=` is setting the RStudio password and is required by the RStudio Docker image. It can be whatever you like except it cannot be `rstudio`. Log in to RStudio with the username `rstudio` and whatever password was specified.

If you want to run RStudio as a user on your host machine, in order to read/write files in a host directory, please read this.

NOTE: If you forget to add the tag `devel` or `RELEASE_X_Y` while using the `bioconductor/bioconductor_docker` image, it will automatically use the `latest` tag which points to the latest RELEASE version of Bioconductor.

To run R from the command line:

```
docker run -it --user rstudio bioconductor/bioconductor_docker:devel R
```

To open a Bash shell on the container:

```
docker run -it --user rstudio bioconductor/bioconductor_docker:devel bash
```

Note: The `docker run` command is very powerful and versatile. For full documentation, type `docker run --help` or visit the help page.

[Back to top]

Mounting Additional Volume

One such option for `docker run` is `-v` to mount an additional volume to the docker image. This might be useful for say mounting a local R install directory for use on the docker. The path on the docker image that should be mapped to a local R library directory is `/usr/local/lib/R/host-site-library`.

The follow example would mount my locally installed packages to this docker directory. In turn, that path is automatically loaded in the R `.libPaths` on the docker image and all of my locally installed package would be available for use.

- Running it interactively,

  ```
  docker run \
      -v /home/my-devel-library:/usr/local/lib/R/host-site-library \
      -it \
      --user rstudio \
      bioconductor/bioconductor_docker:devel
  ```

  without the `--user rstudio` option, the container is started and logged in as the `root` user.

The `-it` flag gives you an interactive tty (shell/terminal) to the docker container.

- Running it with RStudio interface

```
docker run \
    -v /home/my-devel-library:/usr/local/lib/R/host-site-library \
    -e PASSWORD=password \
    -p 8787:8787 \
    bioconductor/bioconductor_docker:devel
```

[Back to top]

Modifying the images

There are two ways to modify these images:

1. Making changes in a running container and then committing them using the `docker commit` command.

```
docker commit <CONTAINER ID> <name for new image>
```

2. Using a Dockerfile to declare the changes you want to make.

The second way is the recommended way. Both ways are documented here.

Example 1:

My goal is to add a python package 'tensorflow' and to install a Bioconductor package called 'scAlign' on top of the base docker image i.e bioconductor/bioconductor_docker:devel.

As a first step, my Dockerfile should inherit from the `bioconductor/bioconductor_docker:devel` image, and build from there. Since all docker images are Linux environments, and this container is specifically 'Debian', I need some knowledge on how to install libraries on Linux machines.

In your new `Dockerfile`, you can have the following commands

```
# Docker inheritance
FROM bioconductor/bioconductor_docker:devel

# Update apt-get
RUN apt-get update \
        ## Install the python package tensorflow
        && pip install tensorflow               \
        ## Remove packages in '/var/cache/' and 'var/lib'
        ## to remove side-effects of apt-get update
        && apt-get clean \
```

```
        && rm -rf /var/lib/apt/lists/*

# Install required Bioconductor package
RUN R -e 'BiocManager::install("scAlign")'
```

This `Dockerfile` can be built with the command, (note: you can name it however you want)

```
docker build -t bioconductor_docker_tensorflow:devel .
```

This will let you use the docker image with 'tensorflow' installed and also `scAlign` package.

```
docker run -p 8787:8787 -e PASSWORD=bioc bioconductor_docker_tensorflow:devel
```

Example 2:

My goal is to add all the required infrastructure to be able to compile vignettes and knit documents into pdf files. My `Dockerfile` will look like the following for this requirement,

```
# This docker image has LaTeX to build the vignettes
FROM bioconductor/bioconductor_docker:devel

# Update apt-get
RUN apt-get update \
        && apt-get install -y --no-install-recommends apt-utils \
        && apt-get install -y --no-install-recommends \
        texlive \
        texlive-latex-extra \
        texlive-fonts-extra \
        texlive-bibtex-extra \
        texlive-science \
        texi2html \
        texinfo \
        && apt-get clean \
        && rm -rf /var/lib/apt/lists/*

## Install BiocStyle
RUN R -e 'BiocManager::install("BiocStyle")'
```

This `Dockerfile` can be built with the command,

```
docker build -t bioconductor_docker_latex:devel .
```

This will let you use the docker image as needed to build and compile vignettes for packages.

```
docker run -p 8787:8787 -e PASSWORD=bioc bioconductor_docker_latex:devel
```

[Back to top]

Singularity

The latest `bioconductor/bioconductor_docker` images are available on Singularity Hub as well. Singularity is a container runtime just like Docker, and Singularity Hub is the host registry for Singularity containers.

You can find the Singularity containers collection on this link https://singularity-hub.org/collections/3955.

These images are particularly useful on compute clusters where you don't need admin access. You need to have the module `singularity` installed. See https://singularity.lbl.gov/docs-installation (contact your IT department when in doubt).

If you have Singularity installed on your machine or cluster are:

Inspect available modules

```
module available
```

If Singularity is available,

```
module load singularity
```

Please check this link for specific usage instructions relevant to Singularity containers and their usage https://www.rocker-project.org/use/singularity/.

Microsoft Azure Container Instances

If you are a Microsoft Azure user, you have an option to run your containers using images hosted on Microsoft Container Registry.

> Microsoft Container Registry (MCR) is the primary Registry for all Microsoft Published docker images that offers a reliable and trustworthy delivery of container images with a syndicated catalog

Using containers hosted on Microsoft Container Registry

You can learn more about the `bioconductor_docker` image hosted on Micosoft Container Registry here.

Pull the `bioconductor_docker` image from Microsoft Container Registry, specifying your `tag` of choice. Check here for the list of tags under "Full Tag Listing":

```
docker pull mcr.microsoft.com/bioconductor/bioconductor_docker:<tag>
```

To pull the latest image:

```
docker pull mcr.microsoft.com/bioconductor/bioconductor_docker:latest
```

Example: Run RStudio interactively from your docker container

To run RStudio in a web browser session, run the following and access it from `127.0.0.1:8787`. The default user name is “rstudio” and you can specify your password as the example below (here, it is set to ‘bioc’):

```
docker run --name bioconductor_docker_rstudio \
        -v ~/host-site-library:/usr/local/lib/R/host-site-library \
        -e PASSWORD='bioc'                                  \
        -p 8787:8787                                        \
        mcr.microsoft.com/bioconductor/bioconductor_docker:latest
```

To run RStudio on your terminal:

```
docker run --name bioconductor_docker_rstudio \
        -it                                                 \
        -v ~/host-site-library:/usr/local/lib/R/host-site-library \
        -e PASSWORD='bioc'                                  \
        -p 8787:8787                                        \
        mcr.microsoft.com/bioconductor/bioconductor_docker:latest R
```

[Back to top]

Use Azure Container Instances to run bioconductor images on-demand on Azure

Azure Container Instances or ACI provide a way to run Docker containers on-demand in a managed, serverless Azure environment. To learn more, check out the documentation here.

Run bioconductor images using ACI

Prerequisites:

1. An Azure account and a subscription you can create resources in
2. Azure CLI
3. Create a resource group within your subscription

You can run Azure CLI or “az cli” commands to create, stop, restart or delete container instances running any bioconductor image - either official images by bioconductor or images available on Microsoft Container Registry. To get started, ensure you have an Azure account and a subscription or create a free

account.

Follow this tutorial to get familiar with Azure Container Instances.

To run the bioconductor image hosted on Microsoft Container Registry or MCR, create a new resource group in your Azure subscription. Then run the following command using Azure CLI. You can customize any or all of the inputs. This command is adapted to run on an Ubuntu machine:

```
az container create \
        --resource-group resourceGroupName \
        --name mcr-bioconductor \
        --image mcr.microsoft.com/bioconductor/bioconductor_docker \
        --cpu 2 \
        --memory 4 \
        --dns-name-label mcr-bioconductor \
        --ports 8787 \
        --environment-variables 'PASSWORD'='bioc'
```

When completed, run this command to get the fully qualified domain name(FQDN):

```
az container show \
        --resource-group resourceGroupName \
        --name mcr-bioconductor \
        --query "{FQDN:ipAddress.fqdn,ProvisioningState:provisioningState}" \
        --out table
```

Here we expose port `8787` on this publicly accessible FQDN. You may have to choose a different “dns-name-label” to avoid conflicts. By default, the username for RStudio is “rstudio” (similar to the official bioconductor docker image). Here we set the password for RStudio to ‘bioc’ in the environment variable configuration. The `--cpu` and `--memory` (in GB) configurations can also be customized to your needs. By default, ACI have 1 cpu core and 1.5GB of memory assigned.

To learn more about what you can configure and customize when creating an ACI, run:

```
az container create --help
```

Mount Azure File Share to persist analysis data between sessions

To ensure that the data persists between different analysis sessions when using Azure Container Instances, you can use the feature to mount Azure file share to your container instance. In this example, we will create an ACI that mounts the “/home/rstudio” directory in RStudio to an Azure File Share.

Prerequisites:

1. An Azure account and a subscription you can create resources in
2. Azure CLI
3. Create a resource group within your subscription

Now, run the following Azure CLI commands to:

1. Create an Azure Storage account
2. Create an Azure file share
3. Get the storage account key

```
# Change these four parameters as needed
ACI_PERS_RESOURCE_GROUP=resourceGroupName
ACI_PERS_STORAGE_ACCOUNT_NAME=storageAccountName
ACI_PERS_LOCATION=eastus
ACI_PERS_SHARE_NAME=fileShareName

# Step1: Create the storage account with the parameters
az storage account create \
        --resource-group $ACI_PERS_RESOURCE_GROUP \
        --name $ACI_PERS_STORAGE_ACCOUNT_NAME \
        --location $ACI_PERS_LOCATION \
        --sku Standard_LRS

# Step2: Create the file share
az storage share create \
        --name $ACI_PERS_SHARE_NAME \
        --account-name $ACI_PERS_STORAGE_ACCOUNT_NAME

# Step3: Get the storage account key
STORAGE_KEY=$(az storage account keys list \
        --resource-group $ACI_PERS_RESOURCE_GROUP \
        --account-name $ACI_PERS_STORAGE_ACCOUNT_NAME \
        --query "[0].value" --output tsv)
echo $STORAGE_KEY
```

Here is an example command to mount an Azure file share to an ACI running bioconductor. This command is adapted to run on an Ubuntu machine:

```
az container create \
        --resource-group resourceGroupName \
        --name mcr-bioconductor-fs \
        --image mcr.microsoft.com/bioconductor/bioconductor_docker \
        --dns-name-label mcr-bioconductor-fs \
        --cpu 2 \
        --memory 4 \
        --ports 8787 \
```

```
        --environment-variables 'PASSWORD'='bioc' \
        --azure-file-volume-account-name storageAccountName \
        --azure-file-volume-account-key $STORAGE_KEY \
        --azure-file-volume-share-name fileShareName \
        --azure-file-volume-mount-path /home/rstudio
```

When completed, run this command to get the fully qualified domain name or FQDN:

```
az container show \
        --resource-group resourceGroupName \
        --name mcr-bioconductor-fs \
        --query "{FQDN:ipAddress.fqdn,ProvisioningState:provisioningState}" \
        --out table
```

Here we expose port 8787 on this publicly accessible FQDN. You may have to choose a different “dns-name-label” to avoid conflicts. By default, the username for RStudio is “rstudio” (similar to the official bioconductor docker image). Here we set the password for RStudio to ‘bioc’ in the environment variable configuration. The “–cpu” and “–memory” (in GB) configurations can also be customized to your needs. By default, ACI have 1 cpu core and 1.5GB of memory assigned. Here, we also mount RStudio “/home/rstudio” directory to a persistent Azure file share named “fileShareName” in the storage account specified. When you stop or restart an ACI, this data will not be lost.

Stop, Start, Restart or Delete containers running on ACI

You can run Azure CLI commands to stop, start, restart or delete container instances on Azure. You can find all the commands and options here.

Replace `containerName` and `resourceGroupName` in the following CLI commands.

Stop the container instance

```
az container stop -n containerName -g resourceGroupName
```

Start the container instance

```
az container start -n containerName -g resourceGroupName
```

Restart the container instance

```
az container restart -n containerName -g resourceGroupName
```

Delete the container instance

```
az container delete -n containerName -g resourceGroupName
```

To not be prompted for confirmation for deleting the ACI:

```
az container delete -n containerName -g resourceGroupName -y
```

To troubleshoot any issues when using Azure Container Instances, try out the recommendations here. For feedback or further issues, contact us via email.

[Back to top]

How to Contribute

There is a comprehensive list of best practices and standards on how community members can contribute images here.

link: https://github.com/Bioconductor/bioconductor_docker/blob/master/best_practices.md

Acknowledgements

Thanks to the rocker project for providing the R/RStudio Server containers upon which ours are based.

Contact us: support.bioconductor.org

Search: