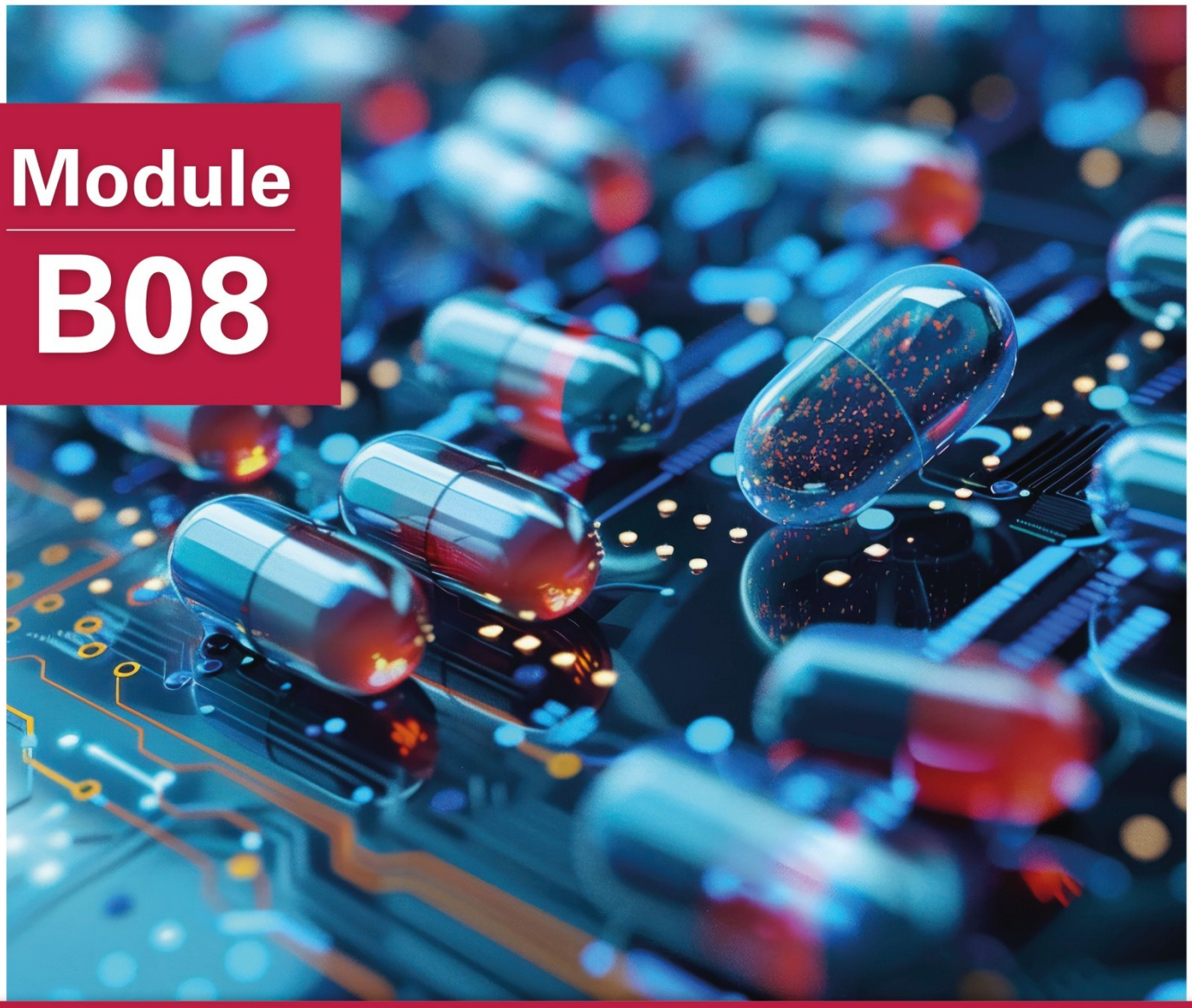
AfricaCDC
Centres for Disease Control and Prevention

Safeguarding Africa's Health



Module B08

Bioinformatics Foundational Course

Virtual Environments and Containerization

NGS Academy for the Africa CDC

ASLM
AFRICAN SOCIETY FOR LABORATORY MEDICINE

H3ABioNet
Pan African Bioinformatics Network for H3Africa

ASLM ACADEMY

Module B08

Virtual Environments and Containerization

back to the table of modules

Module last updated:
December 2024

Suggested or approximate number of sessions	4
Suggested or approximate total learning time	8-10 hours
Target audience	Bioinformaticians
Delivery format	Lectures, videos, practicals
Level of the module	Intermediate

Contributors

George Githinji, Yiqun Li, Suresh Maslamoney and Perceval Maturure.

Suggested prerequisite module(s)

- Module B02. Introduction to Version Control

Module description

Containers represent a method for encapsulating software dependencies and resource access, such as files and network communication, in a standardized manner. They operate within the context of your computer's underlying hardware components, such as the CPU, RAM, and hard drive, which are essential for its operation but typically remain unseen in everyday use. Container systems like Docker function as specialized tools installed on your computer, facilitating these capabilities. Drawing an analogy to shipping containers, before their introduction, the loading and unloading of cargo ships were cumbersome processes prone to errors and confusion. Similarly, software containers, like their physical counterparts, uphold the integrity of their contents during transit, streamlining the description and creation of complete software systems.

Docker, in particular, stands out as a versatile tool for crafting containers. While other containerization tools exist, Docker simplifies the deployment process by allowing containers to be seamlessly deployed onto any computer with Docker installed, known as the 'container host'. Containers exemplify a form of virtualization, wherein a second "virtual" computing environment operates alongside and is accessible from the primary or host computer. Virtual machines (VMs) offer another instance of virtualization, typically containing an entire operating system and filesystem, requiring boot-up akin to a physical computer. Notably, containers are regarded as lightweight versions of VMs, leveraging the underlying Linux kernel and a specific Linux flavor, alongside the filesystem encapsulated within. This module introduces virtual environments and containerization of software. In this module, participants are also introduced to the following topics and/or concepts:

- What is virtualization?
- Software dependency management using Conda
- Introduction to BioConda
- What are containers, what options are there (Docker, Singularity) and why are they useful?
- Introducing the Docker Command Line. How do I know Docker is installed and running?
- How to interact with Docker
- Exploring and running containers, interacting with Docker containers and container images on your computer
- Cleaning up containers
- Managing containers and container images
- What is the Docker Hub, finding containers on Docker hub
- Creating container images
- Documenting the ‘recipe’ for a Docker container image
- Creating more complex container images
- Examples of using container images in practice
- Containers in research workflows: reproducibility and granularity.

This module will also serve as an introduction to Conda for scientists, particularly those working with data. Conda is an open-source package and environment management system that operates on Windows, macOS, and Linux. It is designed to simplify the installation, running, and updating of packages and their dependencies. Furthermore, Conda provides a convenient way to create, save, load, and switch between environments on your local computer. While Conda was initially developed for Python programs, it is versatile enough to package and distribute software for various programming languages, including R, Ruby, Lua, Scala, Java, JavaScript, C/C++, and FORTRAN. This module will motivate the use of Conda as a powerful development tool for constructing and sharing project-specific software environments. By utilizing Conda, scientists can establish reproducible workflows in (data) science, facilitating the seamless sharing and collaboration of their projects.

Key points to know

- Almost all software depends on other software components to function, but these components have independent evolutionary paths.
- Small environments that contain only the software that is needed for a given task are easier to replicate and maintain.
- Critical systems that cannot be upgraded, due to cost, difficulty, etc. need to be reproduced on newer systems in a maintainable and self-documented way.
- Virtualization allows multiple environments to run on a single computer.
- Containerization improves upon the virtualization of whole computers by allowing efficient management of the host computer's memory and storage resources.
- Containers are built from 'recipes' that define the required set of software components and the instructions necessary to build/install them within a container image.
- Docker is just one software platform that can create containers and the resources they use.
- Understanding of what Docker containers are, why they are useful and the common terminology used
- Have a working Docker installation on your local system to allow you to use containers
- Understand how to use existing Docker containers for common tasks
- Be able to build your own Docker containers by understanding both the role of a Dockerfile in building containers, and the syntax used in Dockerfiles
- Understand how to manage Docker containers on your local system
- Appreciate issues around reproducibility in software, understand how containers can address some of these issues and what the limits to reproducibility using containers are.

Module learning outcomes

On completion of this module, participants will have a basic knowledge of, or will be able to:

- Understand software dependences and configuration management.
- Explain the advantages of containerization.
- Explain how using containers can solve software configuration problems.
- String software together using a workflow language.
- Containerize the workflow for deployment on different computing environments.

Module assessments

Module practical: Practical available on the ASLM platform
Module quiz: Assessment questions available on the ASLM platform

Module resources

- NGS Academy | GitHub - Resources
- WCS & COG-TRAIN | Slides - Data tools and pipelines
- Carpentries Incubator | GitHub - Reproducible Computational Environments Using Containers: Introduction to Docker
- PLOS One | Article - Managing genomic variant calling workflows with Swift/T

Acknowledgements

We would like to thank the following individuals, in alphabetical order of last name, for their valuable time and effort spent in designing (i.e., drafting, reviewing, and refining) this module: **George Githinji, Yiqun Li, Suresh Maslamoney and Perceval Maturure.**

Furthermore, we would like to thank the following institutions, societies, journals and individuals from whom we sourced open-access resources, used in this module:
Carpentries Incubator, COVID-19 Genomics UK Consortium, Public Library of Science, The Next Generation Sequencing Academy for the Africa Centres for Disease Control and Prevention, Wellcome Connecting Science; Azza Ahmed, Yan Asmann, Faisal Fadlelmola, Jacob Heldenbrand, Daniel Katz, Katherine Kendig, Matthew Kendzior, Tiffany Li, Liudmila Mainzer, Yingxue Ren, Elliott Rodriguez, Matthew Weber, Justin Wozniak, Jennie Zermeno.