[7] LPMX: A pure rootless composable container system

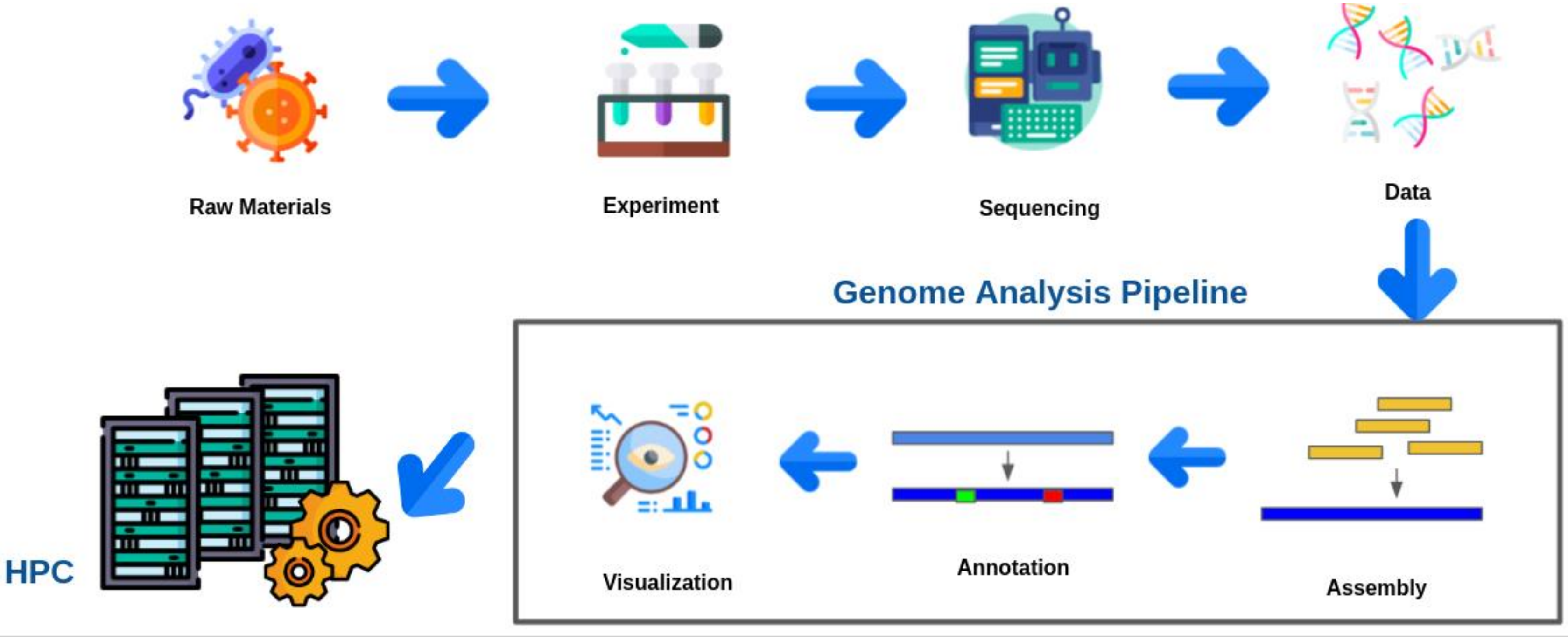
楊　旭 （ヨウ　キョク）
東京大学新領域創成科学研究科 メディカル情報生命専攻
笠原研究室

LPMX
https://github.com/jasonyangshadow/lpmx

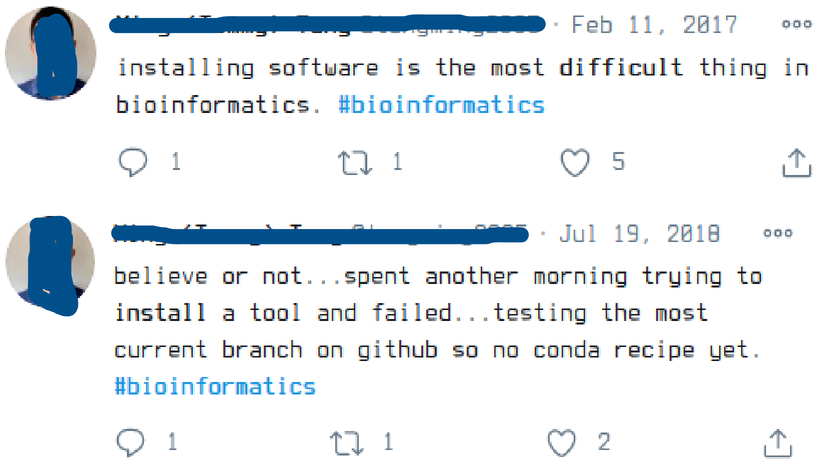
CBMS
Computational Biology and Medical Sciences
DEPARTMENT OF
COMPUTATIONAL BIOLOGY AND MEDICAL SCIENCES,
GRADUATE SCHOOL OF FRONTIER SCIENCES,
THE UNIVERSITY OF TOKYO

[7] LPMX: A pure rootless composable container system

With the advancement in sequencing technologies such as Next Generation Sequencing(NGS), huge amounts of genomic data is being generated at a fast rate. Scientists have to use **pipelines** to automate the analysis procedure and use **High-Performance Computing (HPC) infrastructures** to accelerate the analysis



Bioconductor:	**1974+ packages**
Bioconda:	**3500+ packages**
Biocontainer:	**1000+ images**



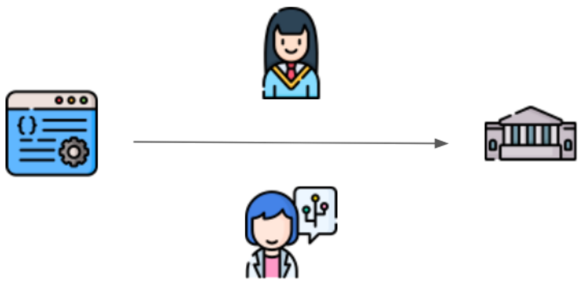
[redacted] · Feb 11, 2017 ···

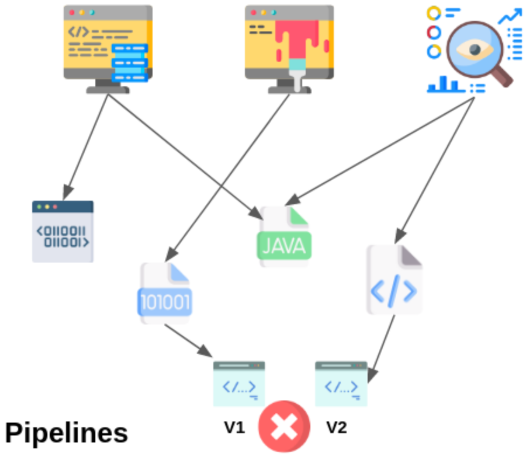
installing software is the most difficult thing in bioinformatics. #bioinformatics

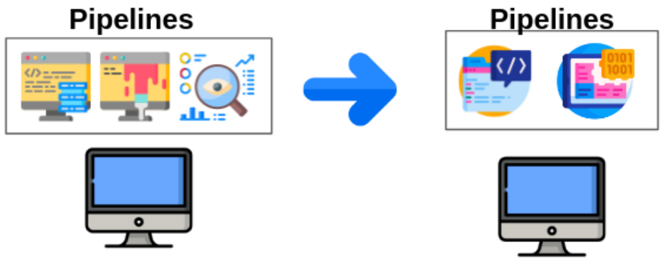
[redacted] · Jul 19, 2018 ···

believe or not...spent another morning trying to install a tool and failed...testing the most current branch on github so no conda recipe yet. #bioinformatics

[7] LPMX: A pure rootless composable container system







Tools are developed by students / researchers often lack maintenance

- For example: GeneWise
- GeneWise was released in 2004 (ubuntu 10.04), and stopped maintenance since version 2.2.3
- Other tools, e.g. PASA2, CEGMA, depend on it
- To compile GeneWise, it will cost researchers a large amount of time to set up the environment

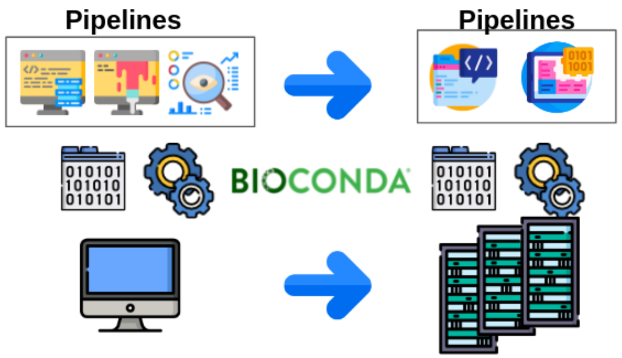
Dependency Hell Issue, arises when different incompatible versions of shared libraries are required by different packages

- For example: Bioconda
- When a package exist in multiple channels and is out of date in the channel with the highest priority. Other packages depending on this package will report incompatible error
- Cannot automatically resolve conflicting packages in the same namespace. For example, Illumina/manta depends Python2, any other packages depending on Python3 will be conflicted.

Difficult to reproduce the runtime environment, users have to run software in compatible runtimes across different machines

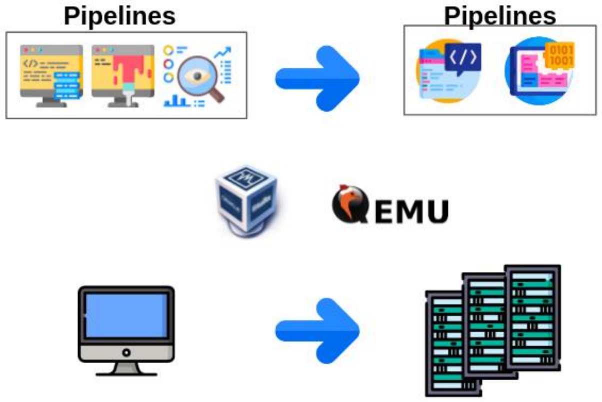
- For example: Difference between OS, VM platforms, Container platforms (Singularity[1])
- It is often difficult to migrate tools from one platform to another due to the requirements for installing compatible platforms and runtimes

[7] LPMX: A pure rootless composable container system



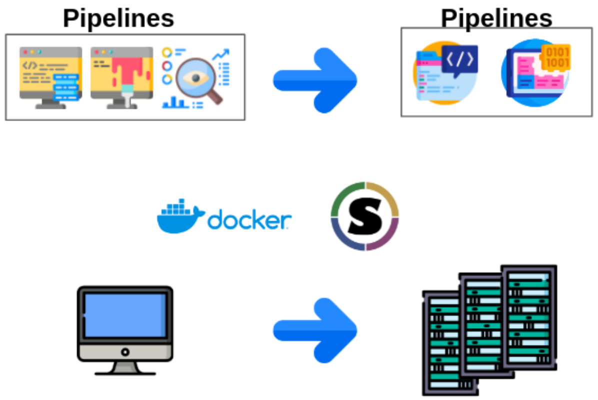
Distribute source tar-ball, compiling source code each time

- Pros
  - Convenient for developers
  - Optimized for current machines
  - If dependencies and libraries are already satisfied, then no need to ask for the admin's help
- Cons
  - Dependencies and libraries are usually not satisfied
  - If packages are conflicting with host ones, then they will fail
  - Weak isolation, libraries in the host have impact on target compiling tools



Use VM to distribute pipelines

- Pros
  - Best isolation, including an entire OS
  - Single file delivery
  - Works on old Linux kernel
- Cons
  - Inflexibility, usually does not support layered file system, so changing a file will result in an entire new package
  - Have to ask the server admin to install the same VM platform
  - Performance overhead, e.g. I/O
  - Pre-allocated computing resources (CPU, memory, network bridges)
  - Security concern

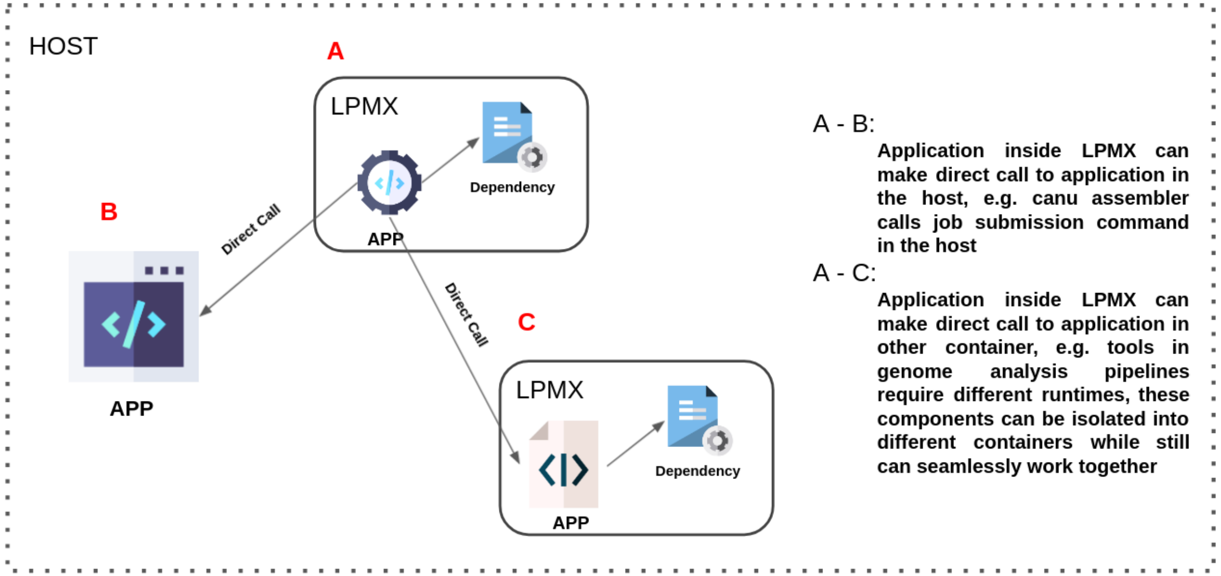


Use container technology to distribute pipelines

- Pros
  - Good isolation, lightweight
  - Support layered file system
  - Flexible, CGroup resource isolation
  - Many out-of-box images
- Cons
  - Have to ask the server admin to install compatible container platforms
  - Require modern Linux kernels
  - Maintenance cost
  - Security concern (Docker Vulnerability)

[7] LPMX: A pure rootless composable container system

- For genome analysis pipelines, researchers usually use a bunch of tools and scripts to shred data files
  - Different tools and scripts require different runtime environments, larger chance to have conflicting packages (Bioconda not work)
- The input and output of these data files are trusted, and the output of previous step is usually used as the input of next steps
  - Mainly handle the file system isolation (Bioconda not work)
- Dependencies and libraries required by these tools and scripts are varieties since they are developed by different teams in past decades.
  - We need to separate tools into different containers to meet their runtime requirements, meantime, we need to handle the communication among these tools (All containers not work)
- Many pipelines supports GPGPU and requires batch job engines
  - We have to support GPGPU as well as batch job systems (All containers not work on batch engine)
- Reduce the cost for negotiation with the server administrator for installing container platforms (HPC)
  - Rootless support (Singularity not work)
- Do modifications inside containers (optional)
  - Read-write container rather than read-only (Singularity not work)
- Robust to old Linux kernel (HPC)
  - It is quite difficult to persuade the server administrator to upgrade Linux kernel, robust to Linux kernel (Docker, Podman not work)
- Support multiple image formats
  - Compatible with different implementations (Docker, Podman not work)



A - B:
Application inside LPMX can make direct call to application in the host, e.g. canu assembler calls job submission command in the host

A - C:
Application inside LPMX can make direct call to application in other container, e.g. tools in genome analysis pipelines require different runtimes, these components can be isolated into different containers while still can seamlessly work together

Composability, refers to the ability that allows processes running inside containers to make calls to other processes in the host or other containers

[7] LPMX: A pure rootless composable container system

Local Package Manager X (LPMX)

A pure rootless composable container system

https://github.com/jasonyangshadow/lpmx

LPMX
https://github.com/
jasonyangshadow/lpmx

- ✓ Pure rootless
- ✓ Robust to kernel version
- ✓ Zero security risk
- ✓ Support Docker & Singularity Images
- ✓ Composability feature
- ✓ Use in restricted environment
- ✓ Low learning curve
- ✓ Easy to access hardware resources
- ✓ Open source and embrace community