Software installation

Installing from source
Installing with apt package manager
Installing with Conda

Packaging systems

Linux software is distributed through packaging systems

Different Linux distributions have different packaging systems that are not compatible with each other

Debian-style: Debian, Ubuntu, Mint

Red-hat style: Fedora, CentOS, Red Hat, openSUSE

Debian style packages have the extension ***.deb*** and Red-hat - ***.rpm***

All software for Linux is found on Internet as package files (.deb, .rpm files) or as source code for manual installation

Linux software is usually available through ***repositories*** or ***repos*** that contain thousands of packages

Linux can interact with repos through a package manager to download and install software

Installing software from source

- Linux software is **open source**, meaning that we can view, download, edit the source code
- We can install the software from the source
- Source code is a set of instructions written in human readable programming language
- This code can be interpreted directly in the case of interpreted languages (perl, python, bash, R) or compiled as machine code (C++, C, java)
- Linux system must contain appropriate interpreters or compilers to interpret or compile source code
- Software required to install from source are called development tools, for example **GNU coreutils**, **gcc, GNU tar, gunzip, bunzip2, make**
- Usually, your system will come with development tools, but they can also be downloaded and installed

Installing software from source

Practice installing software from source

We will install bedtools2 https://github.com/arq5x/bedtools2

Bedtools2 is a versatile toolset for operations in genomic intervals, scan through their github to know more

Search for bedtools2 github and find releases:

src	Setting Release-Version v2.31.0	4 days ago
test	add test for (fixed) #919	4 months ago
tutorial	fix typo in tutorial	3 years ago
.gitignore	use htslib/faidx.h to get fasta sequences	6 months ago
LICENSE	change to MIT license	4 years ago
Makefile	Automatically detect if python2 exists	2 years ago
README.md	Update README.md	3 years ago

Releases 22

bedtools version 2.31.0 Latest
4 days ago

+ 21 releases

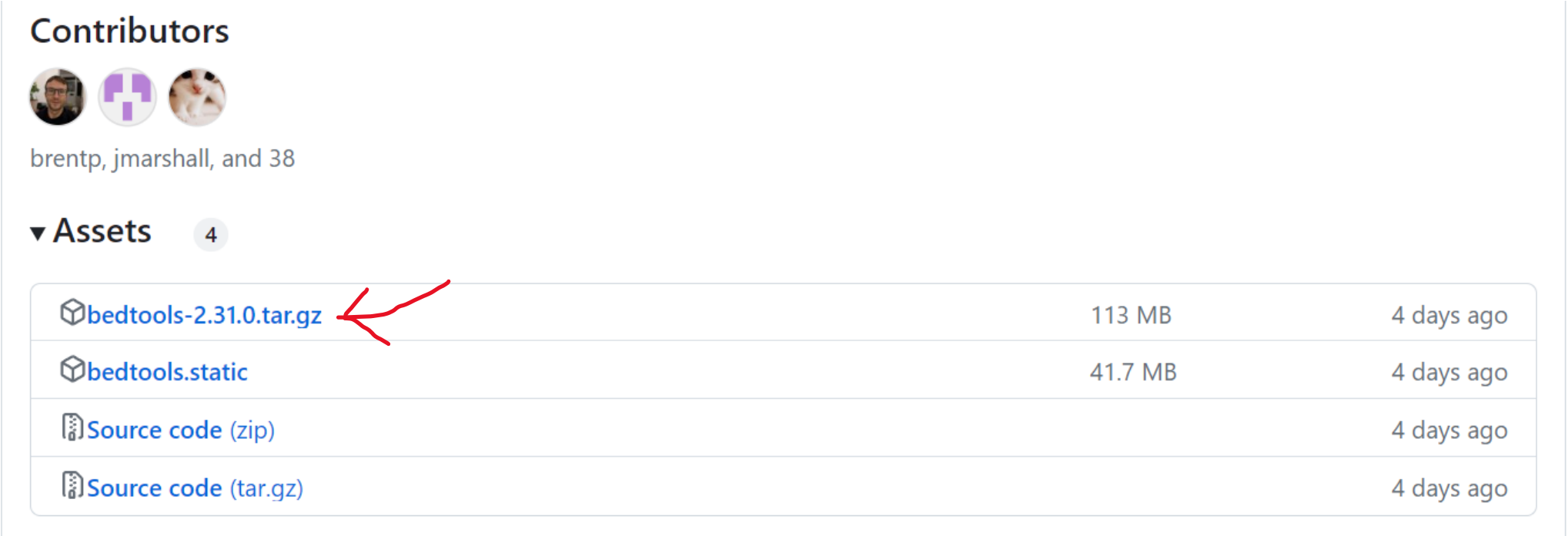
Packages

No packages published

Installing software from source

Practice installing software from source

Click on releases and locate compressed source code for the latest stable release



Right click on **.tar.gz** file and copy link address

Installing software from source

Practice installing software from source

Create **programs/** directory in your home folder and **cd** there

```
$ cd ~
$ mkdir programs
$ cd programs
```

Download **bedtools2** tarball

```
$ wget https://github.com/arq5x/bedtools2/releases/download/v2.31.0/bedtools-2.31.0.tar.gz
```

Read installation instructions https://bedtools.readthedocs.io/en/latest/content/installation.html

Untar and unzip **bedtools2.tar.gz**

```
$ tar -xzvf bedtools-2.31.0.tar.gz
$ cd bedtools2/
$ make # compile binaries
```

Installing software from source

Practice installing software from source

Create **programs/** directory in your home folder and **cd** there

```
$ cd ~
$ mkdir programs
$ cd programs
```

Download **bedtools2** tarball

```
$ wget https://github.com/arq5x/bedtools2/releases/download/v2.31.0/bedtools-2.31.0.tar.gz
```

Read installation instructions https://bedtools.readthedocs.io/en/latest/content/installation.html

Untar and unzip **bedtools2.tar.gz**

```
$ tar -xzvf bedtools-2.31.0.tar.gz
$ cd bedtools2/
$ make # compile binaries
$ ls -lh bin/ # take a look at compiled binaries
$ ./bin/bamToBed -h # test if it works
```

Installing software from source

Practice installing software from source

- Now we can copy **bedtools2/** binaries to **/usr/local/bin** (you will require) or you can simply add **bedtools2/bin** directory to your PATH

- Most of software tools require **dependencies**

- **Dependencies** are other software tools or code libraries that our software tools requires to work

- The presence of dependencies, for example I/O libraries, or other shared libraries is checked during the installation process

- Missing dependencies need to be installed, otherwise you will encounter installation errors

- Some software will need you to run 3 step installation process that includes running ***configure*** script that detects installed dependencies, followed by ***make*** that compiled binaries and ***make install*** that makes the binaries available in your path

Installing with package tools

- Package files for Debian flavor of Linux, like Ubuntu, have *.deb* extension or *.rpm* on Red Hat systems (Fedora, CentOS)
- Packages can be handled by **low-level** and **high-level** package tools
- Low-level tools are ***dpkg*** on Debian and ***rpm*** on Red Hat
- Low-level tools handle tasks of installing and removing packages
- High-level tools are ***apt, apt-get and aptitude*** on Debian or ***yum*** and ***dnf*** on Red Hat
- High level tools conduct searches for metadata and dependency resolution

Use ***apt*** to search repositories

```
$ sudo apt-get update
$ apt-cache search gedit | less # search for a software package gedit – a lightweight text editor
```

Installing with package tools

Practice installation with a package manager

$ sudo apt-get install gedit # install the package

Search and install fastx-toolkit http://hannonlab.cshl.edu/fastx_toolkit/
$ apt-cache search fastx
$ sudo apt-get install fastx-toolkit # install fastx toolkit

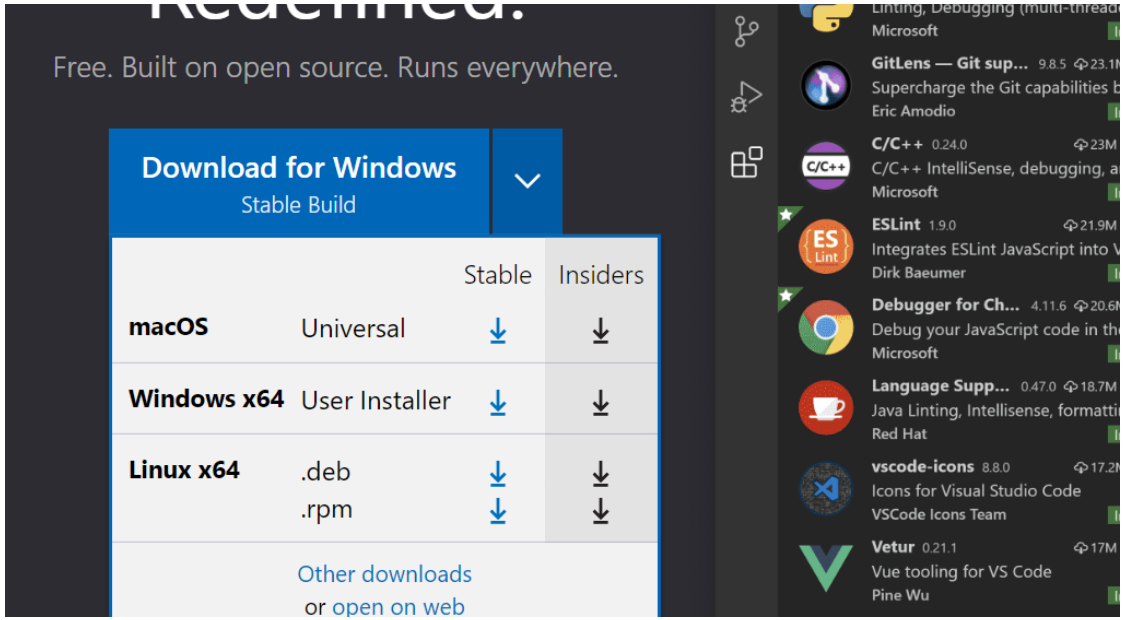
What tools are there?
http://hannonlab.cshl.edu/fastx_toolkit/commandline.html

Test if the installation worked
$ fastq_to_fasta -h

Installing with package tools

We can also use low-level package tools to install directly from package files downloaded to your system

- In some cases, the software is available for download as a package file like *.deb* or *.rpm*
- Let's download and install Visual Studio Code for Linux https://code.visualstudio.com/
- Visual Studio Code is a popular code editor we can later use to edit our code



Installing with package tools

Download Visual Studio Code *.deb* file to your system and install it using ***dpkg***

$ sudo dpkg --install code_1.77.3-1681292746_amd64.deb

$ rm code_1.77.3-1681292746_amd64.deb # remove the .deb file, we don't need anymore

Updating packages

$ sudo apt-get update

$ sudo apt-get upgrade

We can also update specific package from .deb file that contains a more up-to-date version

```
dpkg -i <package_name>
```

List installed packages

$ dpkg -l

$ dpkg -l | grep fastx

Is package installed

$ dpkg -s fastx-toolkit

Installing with package tools

Show information about a package

```
$ apt-cache show fastx-toolkit
```

Find which packages owning files

```
$ dpkg -S fastq_to_fasta
```

Uninstall packages

```
$ sudo apt purge fastx-toolkit
```

Environment management with Conda

- **Conda** is a better way to install and manage bioinformatics software
- In this course we will try to use **conda** install all of the required software
- Why conda? It is the best way to deal with conflicting dependencies
- Almost every software we are installing requires other software (dependency) to work, that other software also requires its own and so on.
- Imagine that we have program A that requires a library shared with a program B, however program B need an older version of the library that A cannot work with. Now we have dependency conflict, and we will need to jump lots of hoops to get both programs to work
- Conda is an environment manager that makes sure we don’t need to deal with conflicting dependencies

Environment management with Conda

- Conda handles dependency-related problems by setting up isolated package installations called **environments**

- I tend to create separate environments for the projects in different domains of NGS, such as RNA-seq, metagenomics, variant calling etc. It maybe worthwhile to create a separate environments for separate NGS analysis projects. Sometimes it may be necessary to create separate environment for old problematic software

- Conda is both a package manager that performs the same functions as **apt,** and an environment manager at the same time

Let's set up **conda** on our system

We will install a light-weight variation of conda called **miniconda**

Download **miniconda** installation script
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

Environment management with Conda

Run the installation script

```
$ bash Miniconda3-latest-Linux-x86_64.sh
```

The installation script will create **miniconda/** directory in your home directory, all of the packages installed through conda will be located there

Activate conda

```
$ conda activate # this will launch a base environment
```

We will create a dedicated environment for this course

```
$ conda create -n course
```

List environments

```
$ conda info --envs
```

Activate **course** environment

```
$ conda activate course
```

Environment management with Conda

Install bioinformatics software with conda

Simply google “conda package_name” and it will lead you to **bioconda** page with the instructions

bioconda is a repository of thousands of bioinformatics packages

Let’s install **fastx-toolkit** with conda

Search **conda + fastx toolkit**

linux-64 v0.0.14

osx-64 v0.0.14

conda install

To install this package run one of the following:

```
conda install -c bioconda fastx_toolkit
```

```
conda install -c "bioconda/label/cf201901" fastx_toolkit
```

Environment management with Conda

Install the package

```
$ conda install -c fastx_toolkit
```

Deactivate environment

```
$ conda deactivate
```

Remove the environment

```
$ conda env remove --name rnaseq
```