



Galaxy Workflow Generator and Galaxy Flavors

Konstantin Riege

Tools

The screenshot shows the Galaxy Tools interface. On the left, a sidebar lists various tools categorized under 'Tools'. Some categories are collapsed, indicated by a minus sign icon. The visible categories include:

- search tools
- Get Data
- Send Data
- ENCODE Tools
- Lift-Over
- Text Manipulation
- Convert Formats
- FASTA manipulation
- Filter and Sort
- Join, Subtract and Group
- Extract Features
- Fetch Sequences
- Fetch Alignments
- Get Genomic Scores
- Operate on Genomic Intervals
- Statistics
- Graph/Display Data
- Regional Variation
- Multiple regression
- Multivariate Analysis
- Evolution
- Motif Tools
- Multiple Alignments
- Metagenomic analyses
- Human Genome Variation
- Genome Diversity
- EMBOSS
- NGS TOOLBOX BETA

Edit Attributes

Name: Join two Queries on data 3 and data 1

Info:

Database/Build: Click to Search or Select

Number of comment lines:

Save

Auto-detect

This will inspect the dataset and attempt to correct the above column values if they are not accurate.

Change data type

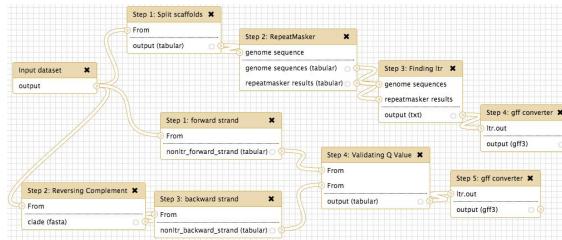
New Type: tabular

This will change the datatype of the existing dataset but not modify its contents. Use this if Galaxy has incorrectly guessed the type of your dataset.

Save

- ### History
- 1.8 Gb
- 14: Draw phylogeny on data 12
 - 13: Draw phylogeny on data 11
 - 12: Find lowest diagnostic rank on data 10
 - 11: Find lowest diagnostic rank on data 9
 - 10: Fetch taxonomic representation on data 8
 - 9: Fetch taxonomic representation on data 7
 - 8: s234 within 5% of max
 - 7: s1 within 5% of max
 - 6: Join two Queries on data 4 and data 2
 - 5: Join two Queries on data 3 and data 1
 - 4: s234 max bit score

Histories



Workflows

Tools

- search tools
- Get Data
- Send Data
- ENCODE Tools**
- Lift-Over
- Text Manipulation
- Convert Formats
- FASTA manipulation
- Filter and Sort
- Join, Subtract and Group
- Extract Features
- Fetch Sequences
- Fetch Alignments
- Get Genomic Scores
- Operate on Genomic Intervals
- Statistics
- Graph/Display Data
- Regional Variation
- Multiple regression
- Multivariate Analysis
- Evolution
- Motif Tools
- Multiple Alignments
- Metagenomic analyses
- Human Genome Variation
- Genome Diversity
- EMBOSS
- NGS TOOLBOX BETA

Edit Attributes

Name: Join two Queries on data 3 and data 1

Info:

Database/Build: Click to Search or Select

Number of comment lines:

Save

Auto-detect

This will inspect the dataset and attempt to correct the above column values if they are not accurate.

Change data type

New Type: tabular

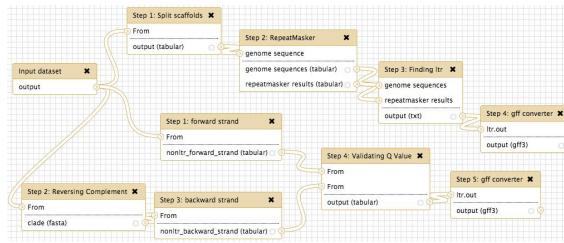
This will change the datatype of the existing dataset but not modify its contents. Use this if Galaxy has incorrectly guessed the type of your dataset.

Save

History

- 1.8 Gb
- 14: Draw phylogeny on data 12
- 13: Draw phylogeny on data 11
- 12: Find lowest diagnostic rank on data 10
- 11: Find lowest diagnostic rank on data 9
- 10: Fetch taxonomic representation on data 8
- 9: Fetch taxonomic representation on data 7
- 8: s234 within 5% of max
- 7: s1 within 5% of max
- 6: Join two Queries on data 4 and data 2
- 5: Join two Queries on data 3 and data 1
- 4: s234 max bit score

Histories



Workflows

Galaxy / Live

Analyze Data Workflow Visualization Shared Data Help Login or Register

History search datasets Unnamed history (empty)

This history is empty. You can load your own data or data from an external source

Jupyter lab for Python, R and Julia

RStudio with basic R packages

SPARQL query interface

<https://live.usegalaxy.eu>

Interactive environments

Galaxy

Tools Options

- search tools
- Get Data
- Send Data
- ENCODE Tools
- Lift-Over
- Text Manipulation
- Convert Formats
- FASTA manipulation
- Filter and Sort
- Join, Subtract and Group
- Extract Features
- Fetch Sequences
- Fetch Alignments
- Get Genomic Scores
- Operate on Genomic Intervals
- Statistics
- Graph/Display Data
- Regional Variation
- Multiple regression
- Multivariate Analysis
- Evolution
- Motif Tools
- Multiple Alignments
- Metagenomic analyses
- Human Genome Variation
- Genome Diversity
- EMBOSS
- NGS TOOLBOX BETA

Edit Attributes

Name: Join two Queries on data 3 and data 1

Info:

Database/Build: Click to Search or Select

Number of comment lines:

Save

Auto-detect

This will inspect the dataset and attempt to correct the above column values if they are not accurate.

Change data type

New Type: tabular

This will change the datatype of the existing dataset but not modify its contents. Use this if Galaxy has incorrectly guessed the type of your dataset.

Save

History Options

1.8 Gb

- 14: Draw phylogeny on data 12
- 13: Draw phylogeny on data 11
- 12: Find lowest diagnostic rank on data 10
- 11: Find lowest diagnostic rank on data 9
- 10: Fetch taxonomic representation on data 8
- 9: Fetch taxonomic representation on data 7
- 8: s234 within 5% of max
- 7: s1 within 5% of max
- 6: Join two Queries on data 4 and data 2
- 5: Join two Queries on data 3 and data 1
- 4: s234 max bit score

Histories

Tools



Interactive tours:
programmatically simulate **complex** user interaction
on next click

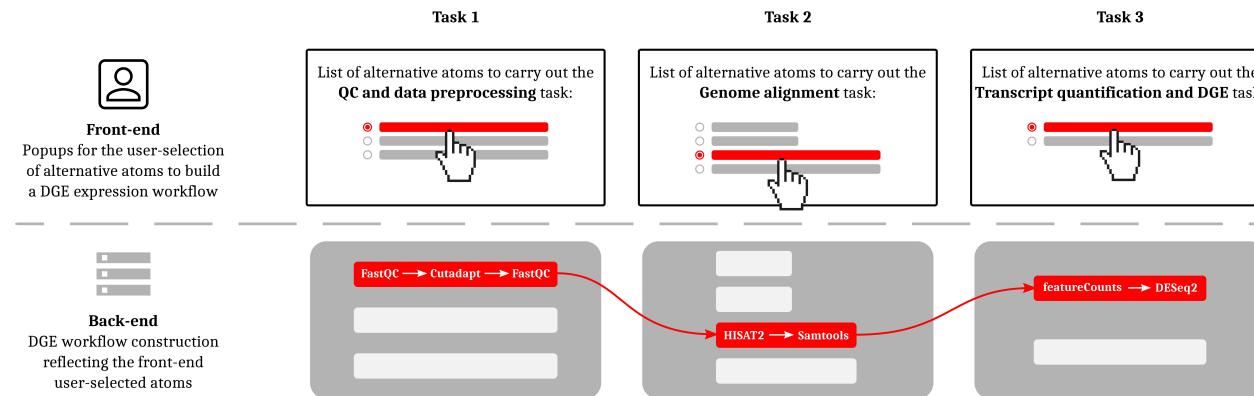
A screenshot of the Galaxy interface. At the top left is the "Galaxy / de.STAIR" header. Below it is a navigation bar with "Tools", "Upload Data", and "Visualization". A search bar says "search tools". On the left, a sidebar titled "Tools" lists various bioinformatics tools. A callout box highlights the "Import the sequence datasets" tool with the text "To start, click on". Below this is a "Send Data" section. The main workspace shows a "Workflow" tab with a history panel containing several green entries. A "Workflow" button is visible at the bottom.

Tools

A screenshot of the Galaxy interface showing the "Edit Attributes" and "Change data type" panels. The "Edit Attributes" panel has fields for "Name" (set to "Join two Queries on data 3 and data 1"), "Info", "Database/Build" (set to "Click to Search or Select"), "Number of comment lines" (checkbox), and "Save" and "Auto-detect" buttons. The "Change data type" panel has a "New Type" dropdown set to "tabular" and a note about inspecting the dataset. The "History" panel on the right shows a list of workflow steps, each with a green background and white text, such as "14: Draw phylogeny on data 12" and "1: s234 max bit score".

Histories

- **de.STAIR Atoms and Galaxy workflow generator plugin**
 - improved stability and crafted atomic interactive tours
 - extended functionality to support various HTML elements
 - integrated a new interaction layer as Galaxy plugin to
 - select and execute alternative tools which solve the same data analysis task (Atoms)
 - guide through parameters
 - extract workflows, citations and commands
 - ensure functionality without registration



<https://destair.leibniz-fli.de>

Galaxy / de.STAIR

Analyze Data Workflow Visualize ▾ Shared Data ▾ Help ▾ Login or Register Workflow generator Using 0%

Tools

search tools

- Upload Data
- Visualization
- Quality Control
- Utilities
- Text processing
- RNA-Seq
- BS/RRBS-Seq
- Annotation
- Workflows
- All workflows

Welcome to the Galaxy workflow generator!

To start, click on the deSTAIR icon in the top menu.

Privacy Policy

Implementation & testing

Tool selection

Type of analyses

Parameter optimization

Solution! Data output

What question I want to answer?
Data input

Get a guided tour with tool modularization by
deSTAIR

History

search datasets

de.STAIR Guide History (non-persistent!)

(empty)

This history is empty. You can load your own data or get data from an external source

How it works

A Galaxy framework is equipped with *alternative* Galaxy tools that are able to address the same biological problem in a different, sometimes more suitable way. Alternative tools can, for example, offer different sensitivity, speed, memory usage, as well as distinct sets of parameters and features.

For these reasons, to build Galaxy workflows, users are required to choose tools on the basis of their suitability within the context of the target analysis. However, shared best-practice training materials do not uniformly cover their usage, and the growing landscape of computational tools have not made this task any simpler.

<https://destair.leibniz-fli.de>

Screenshot of the Galaxy Workflow Generator interface for de.STAIR.

The main header bar includes: Galaxy / de.STAIR, Analyze Data, Workflow, Visualize, Shared Data, Help, Login or Register, Abort, and Using 0%.

The left sidebar lists: Tools, Upload Data, Visualization, Quality Control, Utilities, Text processing, RNA-Seq, RRBS/BS-Seq, Annotation, Workflow, and All workflows.

The central area features a "Welcome to the Galaxy workflow generator!" message with a "Privacy Policy" link and a "Get started" button.

A large diagram illustrates a workflow process involving "Data processing & analysis", "Differential gene expression", and "Differential gene control".

A callout box highlights the "Welcome to de.STAIR workflow generator" and "Which type of analysis do you want to perform?" section.

The analysis selection options are:

- Differential gene expression analysis (single-end reads)
- Differential gene expression analysis (paired-end reads)
- RRBS/BS-Seq analysis (paired-end reads)

Checkboxes for "Auto run tour" and a "Submit" button are present.

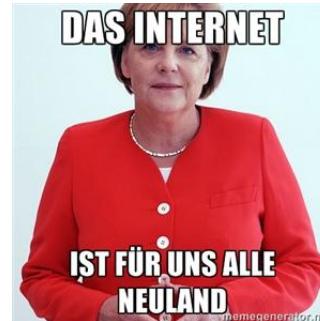
A detailed description of the "de.STAIR Differential gene expression analysis" is shown in a callout box:

In this analysis we will build an RNA-Seq workflow to find the most significantly expressed genes in human genome build HG38, given 4 control and 4 treatment data sets of single-end Illumina sequences.

The right sidebar shows a "History" section with a note: "This history is empty. You can load your own data or get data from an external source."

But I simply want to
know how to setup a
Galaxy instance





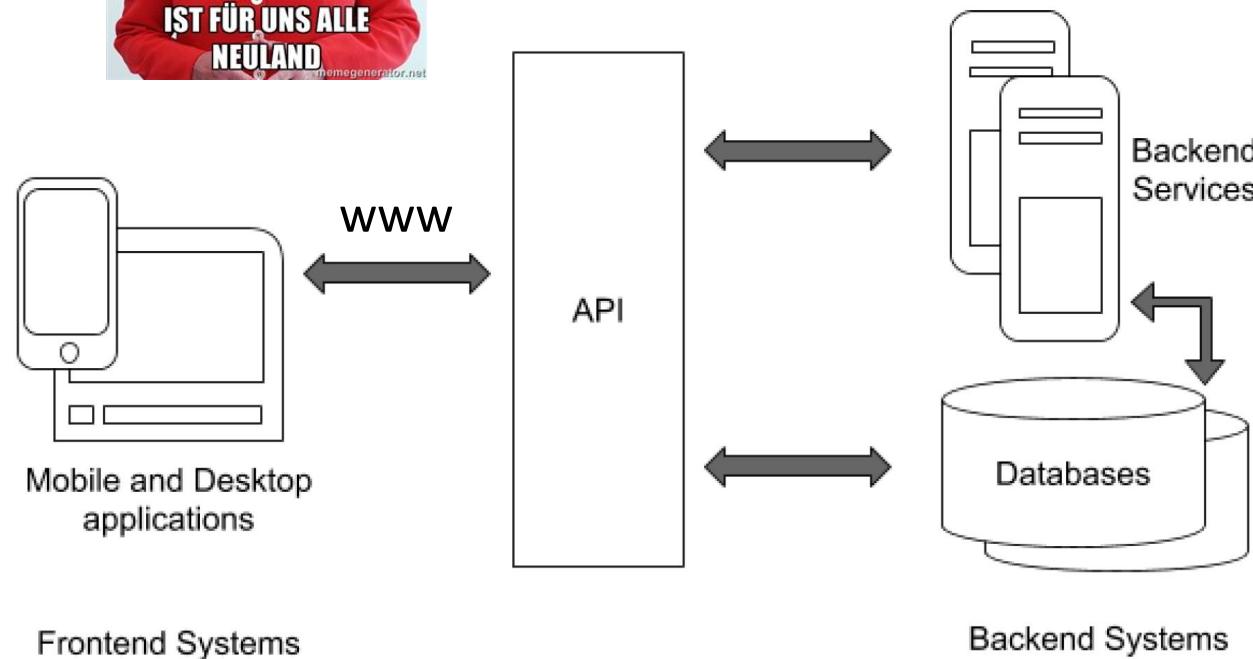
“The internet is unknown territory to all of us”

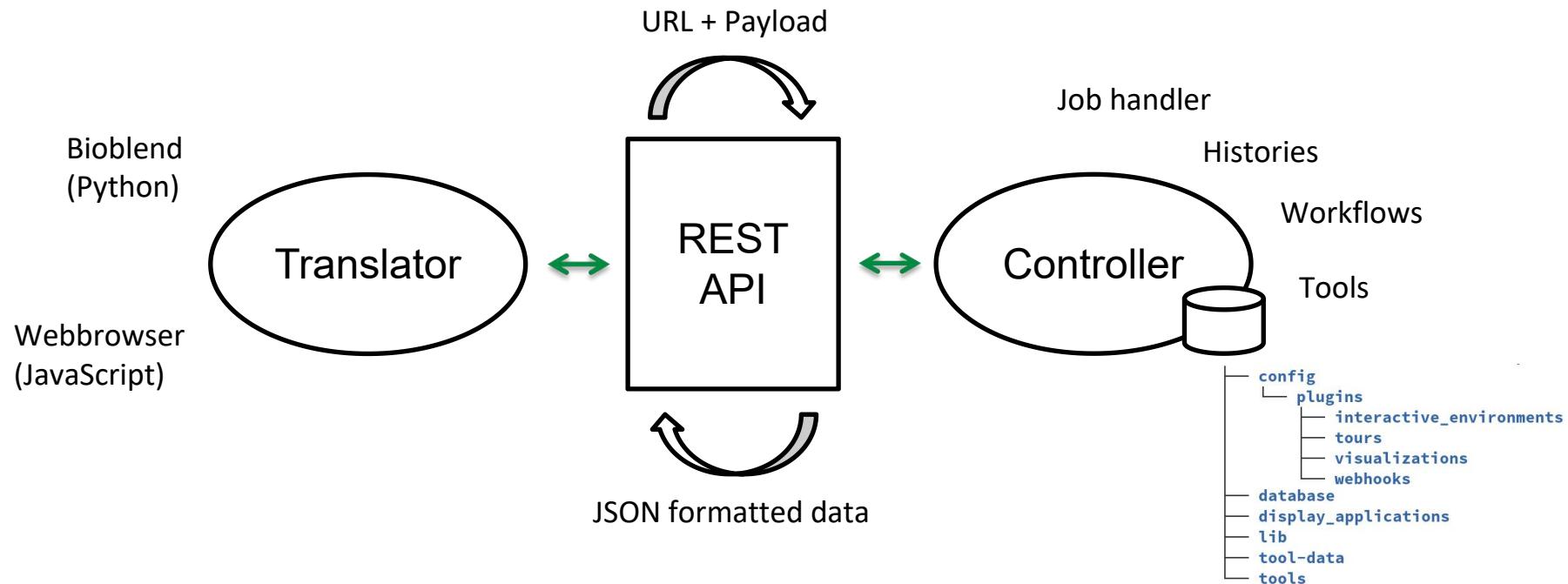
Merkel 2013, Press briefing topic: NSA monitoring



"The internet is unknown territory to all of us"

Merkel 2013, Press briefing topic: NSA monitoring





...

<https://destair.leibniz-fli.de/api/tours>

<https://destair.leibniz-fli.de/api/histories>

<https://destair.leibniz-fli.de/api/workflows>

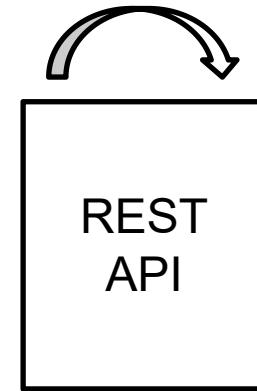
<https://destair.leibniz-fli.de/api/tools>

...

https://destair.leibniz-fli.de/api/datatypes?extension_only=False

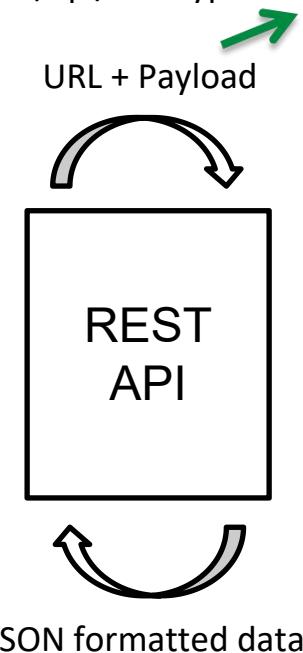


URL + Payload



```
[  
 {  
   "elems": [  
     {  
       "panel_section_name": "Upload Data",  
       "description": "convert Fastq data into unaligned BAM",  
       "labels": [  
         ],  
       "edam_operations": [  
         ],  
       "form_style": "regular",  
       "edam_topics": [  
         ],  
       "panel_section_id": "gettext",  
       "version": "2.18.2.1",  
       "link": "/tool\_runner?tool\_id=toolshed.g2.bx.psu.edu/repos%2Fdevteam%2Fpicard%2Fpicard\_FastqToSam%2F2.18.2.1",  
       "target": "galaxy_main",  
       "min_width": -1,  
       "model_class": "Tool",  
       "id": "toolshed.g2.bx.psu.edu/repos/devteam/picard/picard_FastqToSam/2.18.2.1",  
       "tool_shed_repository": {  
         "owner": "devteam",  
         "changeset_revision": "a1f0b3f4b781",  
         "name": "picard",  
         "tool_shed": "toolshed.g2.bx.psu.edu"  
       },  
       "name": "FastqToSam"  
     ],  
     {  
       "panel_section_name": "Upload Data",  
       "description": "charts the nucleotide distribution per cycle in a SAM or BAM dataset",  
       "labels": [  
         ...  
         https://destair.leibniz-fli.de/api/tours  
         https://destair.leibniz-fli.de/api/histories  
         https://destair.leibniz-fli.de/api/workflows  
         https://destair.leibniz-fli.de/api/tools  
         ...  
         https://destair.leibniz-fli.de/api/datatypes?extension_only=False  
       ]  
     }  
   ]  
 }
```

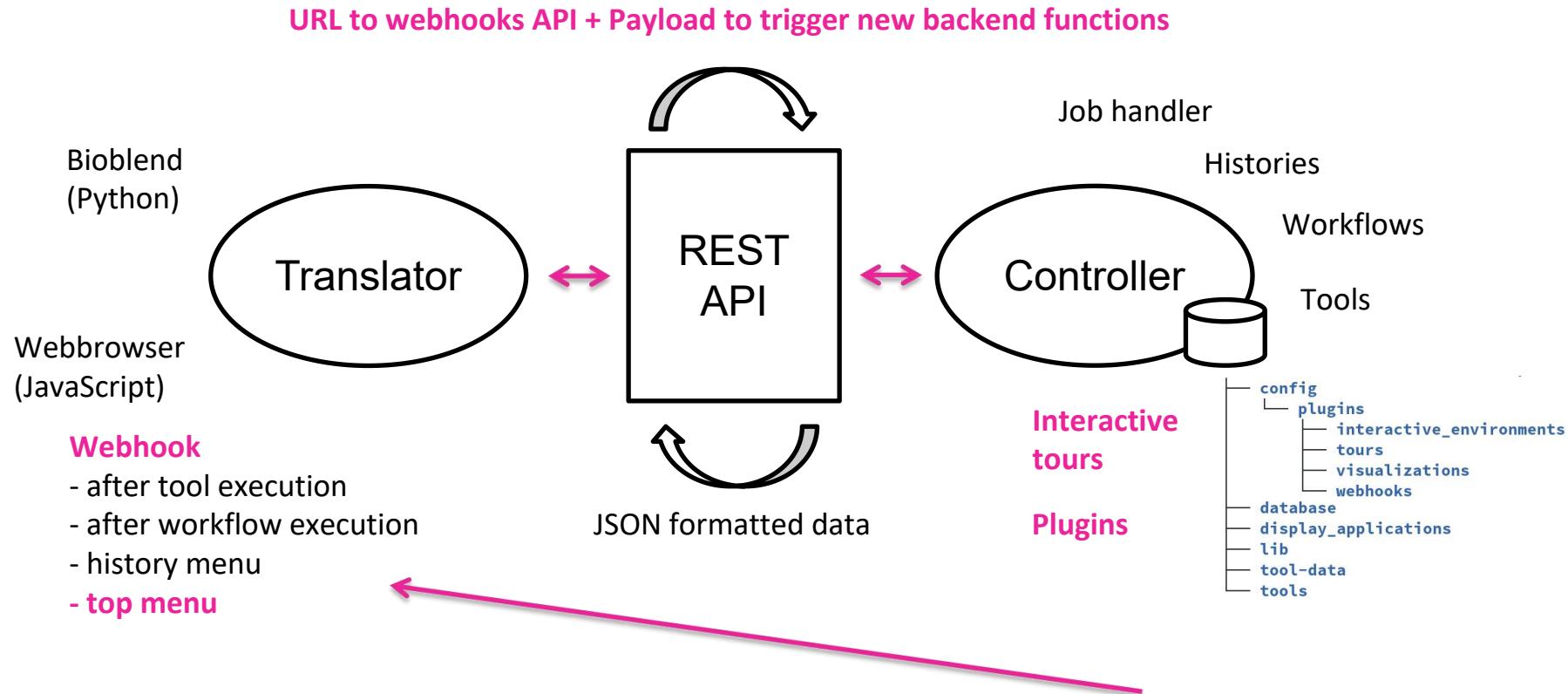
...
<https://destair.leibniz-fli.de/api/tours>
<https://destair.leibniz-fli.de/api/histories>
<https://destair.leibniz-fli.de/api/workflows>
<https://destair.leibniz-fli.de/api/tools>
...
https://destair.leibniz-fli.de/api/datatypes?extension_only=False



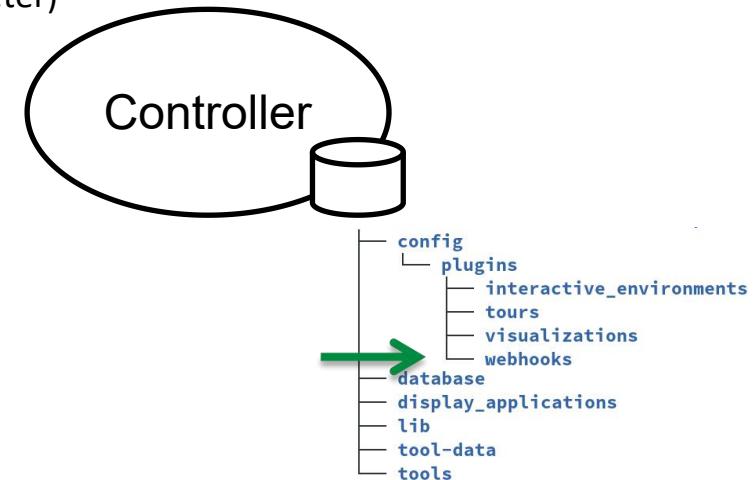
URL + Payload

REST
API

JSON formatted data



- Webhooks
 - JavaScript based frontend
 - triggers HTTP requests e.g. GET with payload (URL parameter) and receives JSON object string
 - Python based backend functions
 - Parse payload and uses its access to the controller objects to perform tasks and return results as JSON object string



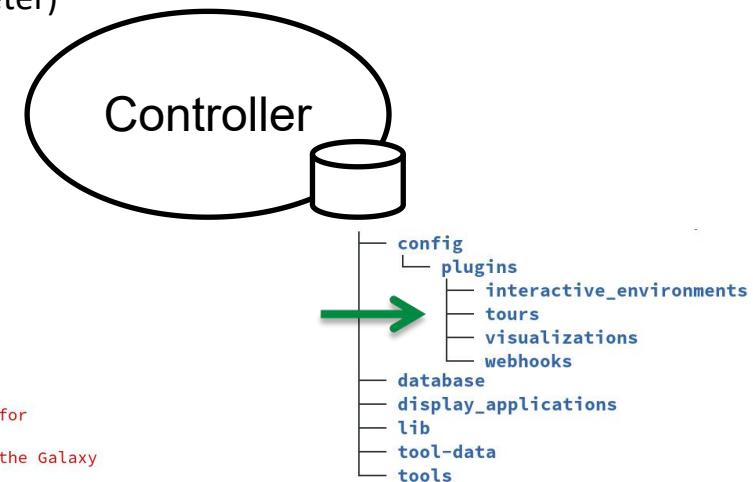
- Webhooks
 - JavaScript based frontend
 - triggers HTTP requests e.g. GET with payload (URL parameter) and receives JSON object string
 - Python based backend functions
 - Parse payload and uses access to controller object to perform tasks and return results as JSON object string

- Tours

- YAML files

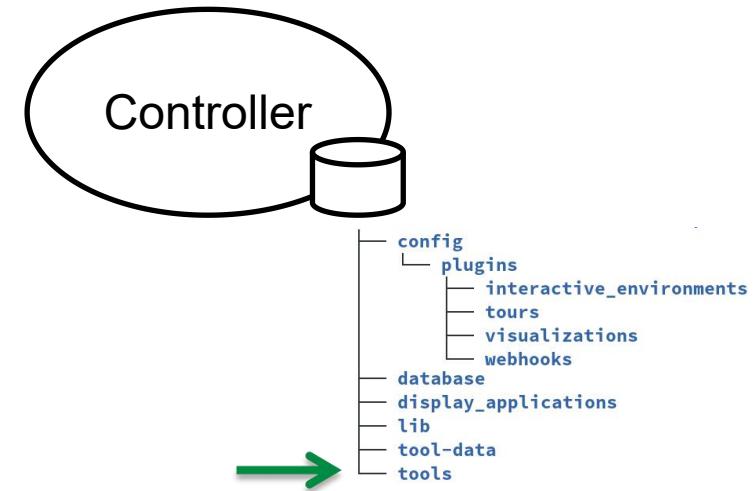
```
id: bs_pe_1a
name: 'BS-Seq analysis (paired-end reads)'
description: 'Data upload'
title_default: 'BS-Seq analysis (paired-end reads)'
steps:
  - title: '<b>Outline</b>'
    element: '#masthead:visible'
    content: 'In this tour we will upload NGS data to be analysed for
methylation levels.<br>
Here, we will leverage on the <b>Data upload</b> tool provided by the Galaxy
framework.'
    placement: bottom

  - title: '<b>Import the sequence datasets</b>'
    element: '.fa.fa-upload:visible'
    content: 'Import data into Galaxy.'
    placement: right
    onnextclick:
      - '.fa.fa-upload:visible'
    onloadwait:
      - element: '.history-size:contains("empty")' # must not check for visibility
        count: 1
```



- Tools
 - Binaries
 - XML wrapper to render command line options into graphical user interface

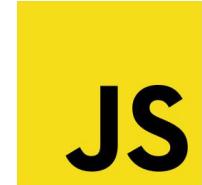
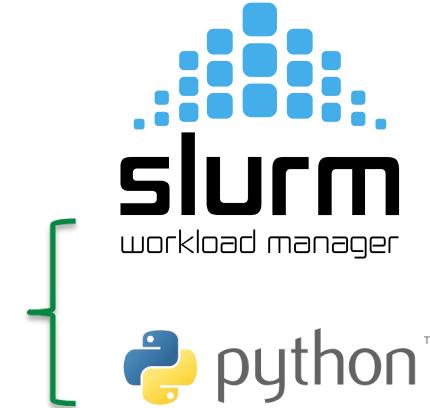
```
<tool id="unique_name" name="actual tool name" version="0.0.1">
  <requirements></requirements>
  <stdio><exit_code range="1:" /></stdio>
  <command><![CDATA[
    binary "$input_1" "$input_2" "$output"
  ]]></command>
  <inputs>
    <param type="data" name="input_1" format="csv" label="input table" />
    <param type="integer" name="input_2" value="0" label="input value" />
  </inputs>
  <outputs>
    <data name="output" format="csv" label="output table"/>
  </outputs>
  <tests></tests>
  <help>![CDATA[
    my binary does something
  ]]></help>
</tool>
```



But I simply want to
know how to setup a
Galaxy instance



HTTP/FTP Server
Job handler
Scheduler plugins

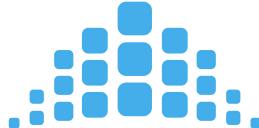




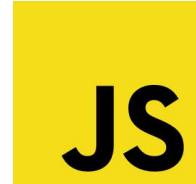
BioContainers



HTTP/FTP Server
Job handler
Scheduler plugins



workload manager



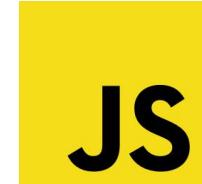
...scratching the surface



BioContainers



HTTP/FTP Server
Job handler
Scheduler plugins



ubuntu



docker

- Fused and stabilized via Docker
 - Dependencies and libraries
 - Necessary environment setting
 - Databases
 - Webservers
 - Software
 - Build systems and job handler
- A universe of techniques to avoid pitfalls in manual pipeline design

- Fused and stabilized via Docker
 - Dependencies and libraries
 - Necessary environment setting
 - Databases
 - Webservers
 - Software
 - Build systems and job handler
- A universe of techniques to avoid pitfalls in manual pipeline design

..lets unravel this!..

- The make build system
 - make is a tool used by programmers to define how software should be compiled and installed
 - It is also used as a way to store a set of commands to recall them later
 - Makefile is composed of rules to fulfill certain tasks and stores the associated shell commands



A screenshot of a terminal window titled "gioby@dayhoff: ~". The window contains the following text:

```
$: cat >Makefile
print_hello: echo 'Hello, world!'
```

Annotations with arrows and text labels explain the components of the Makefile rule:

- "Target of the rule" points to the word "print_hello".
- "This is a tabulation (not 8 spaces)" points to the tab character at the beginning of the line.
- "Commands associated with the rule" points to the command "echo 'Hello, world!'".
- "Makefile rule" is enclosed in a curly brace {} and points to the entire line "print_hello: echo 'Hello, world!'".

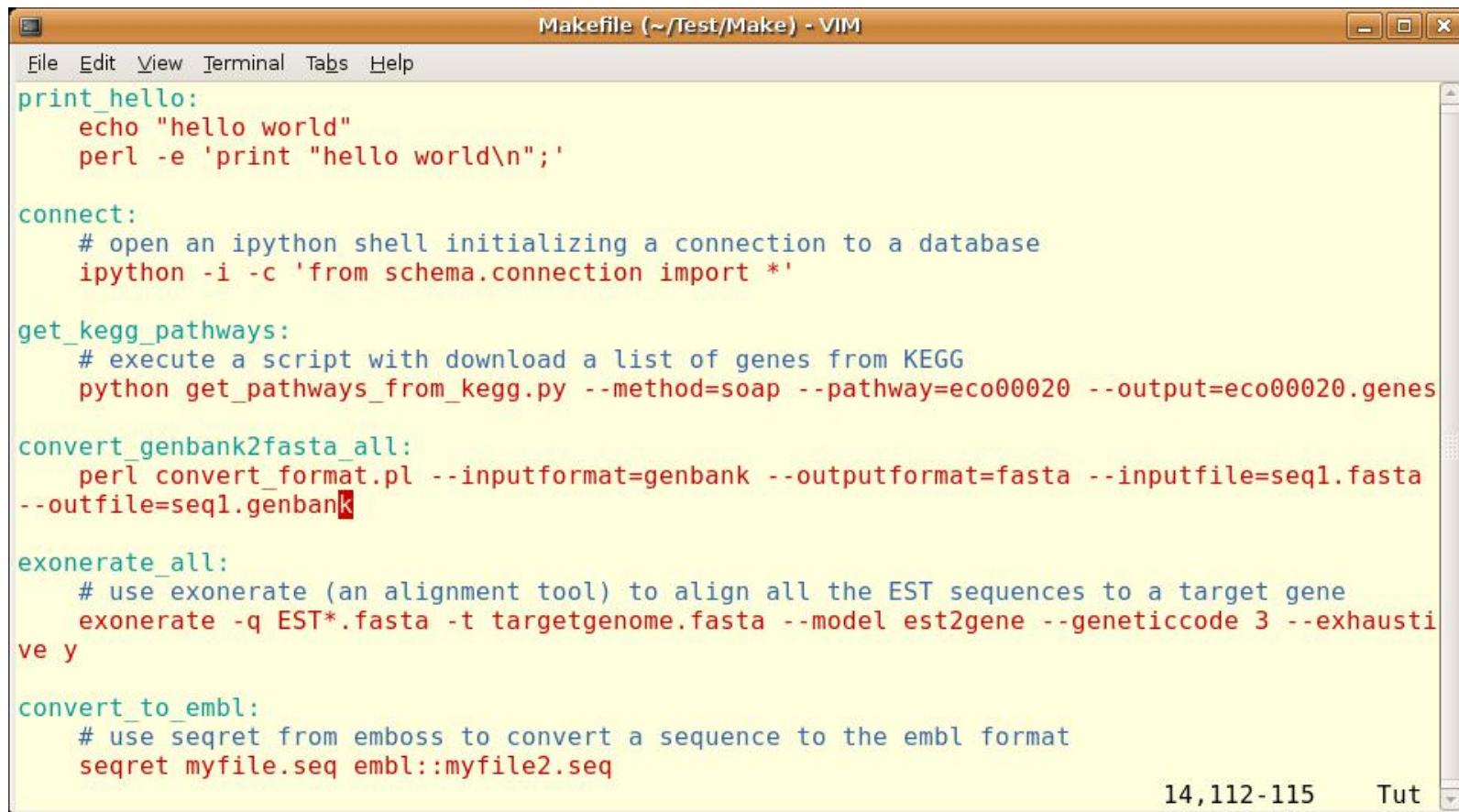
- The make build system
 - invoke with the command “make”
 - make looks for a file in the current directory called “Makefile”
 - When we type “make print_hello”, it executes the function (target) called “print_hello” in the Makefile



The screenshot shows a terminal window titled "gioby@dayhoff: ~/Test/Make". The terminal displays the following commands and output:

```
gioby@dayhoff:~/Test/Make$ cat >Makefile
print_hello:
    echo 'Hello, world!!'

gioby@dayhoff:~/Test/Make$ make print_hello
echo 'Hello, world!!'
Hello, world!!
gioby@dayhoff:~/Test/Make$
```



The screenshot shows a VIM editor window titled "Makefile (~/Test/Make) - VIM". The file contains a series of shell commands (make rules) for processing biological data. The syntax highlighting in VIM distinguishes between different command types and arguments.

```
print_hello:
    echo "hello world"
    perl -e 'print "hello world\n";'

connect:
    # open an ipython shell initializing a connection to a database
    ipython -i -c 'from schema.connection import *'

get_kegg_pathways:
    # execute a script with download a list of genes from KEGG
    python get_pathways_from_kegg.py --method=soap --pathway=eco00020 --output=eco00020.genes

convert_genbank2fasta_all:
    perl convert_format.pl --inputformat=genbank --outputformat=fasta --inputfile=seq1.fasta
    --outfile=seq1.genbank

exonerate_all:
    # use exonerate (an alignment tool) to align all the EST sequences to a target gene
    exonerate -q EST*.fasta -t targetgenome.fasta --model est2gene --geneticcode 3 --exhaustive

convert_to_embl:
    # use seqret from emboss to convert a sequence to the embl format
    seqret myfile.seq embl::myfile2.seq
```

The screenshot shows a VIM editor window titled "Makefile (~/Test/Make) - VIM". The file contains the following Makefile code:

```
print_hello:
    echo "hello world"
    perl -e 'print "hello world\n";'

connect:
    # open an ipython shell initializing a connection to a database
    ipython -i -c 'from schema.connection import *'

get_kegg_pathways:
    # execute a script with download a list of genes from KEGG
    python get

convert_genbank:
    perl convert_genbank.pl --outfile=seq1

exonerate_all:
    # use exon
    exonrate --outfile=seq1.fasta --exhaustive

ve y

convert_to_embl:
    # use seqret from emboss to convert a sequence to the embl format
    seqret myfile.seq embl::myfile2.seq
```

A green callout box highlights the first two targets: `print_hello` and `connect`. Inside the box, the following points are listed:

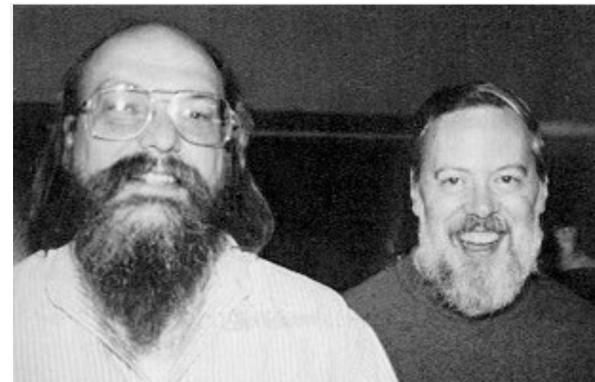
- pre-configure an operating system that can run all of these commands
- create an image out of it which can be shared and deployed and scales on various systems

On the right side of the VIM window, there are several files listed in a sidebar:
00020.genes
seq1.fasta
gene
--exhausti

Back in the days..



- What is an operating system?
 - The set of all the programs needed to use a machine
 - A software collection to coordinate all hardware components (kernel)
 - A software to draw windows (graphical interface)
 - Software to edit texts, browse the internet, install other software, etc..
- Way ahead of Windows: Unix
 - Officialy released 1969
 - ~30 programs/commands (ls, cp, mv,...)
 - Invented shell command structure



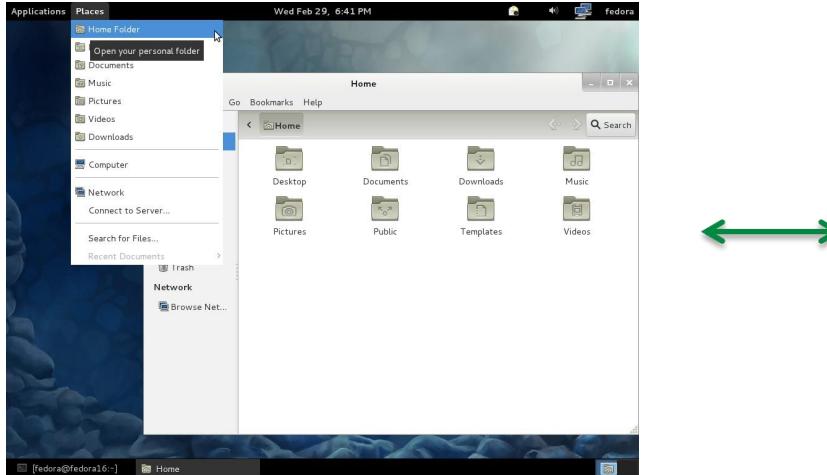
Ken Thompson and Dennis Ritchie

- 1991 GNU Linux implemented Unix kernel concepts as an open source operating system
 - Mac OS
 - Android
 - IoT devices
- Unix was rapidly adopted worldwide, and became the standard operating system, especially in universities due to free licenses
- Thanks to the adoption of GNU/Linux servers, internet grew considerably in 1991 - 1992
- The command interpreter “Unix shell” or “Bourne Again Shell” (Bash) became defacto standard



“Write programs that do one thing and do it well.
Write programs to work together. Write programs to handle
text streams, because that is a universal interface.”

(Doug McIlroy, Unix pipes inventor)



```
fedora@fedora16:~$ ls
Desktop  Downloads  Pictures  Templates
Documents  Music  Public  Videos
[fedora@fedora16 ~]$
```

- Command structure
 - tool name: ls, cp, rm, mv, grep, cat, head, tail, sort, paste, join, ...
 - options may or may not start with a minus or double minus (long version)

toolname option [argument] [option [argument] ..]

- textual representation enables to chain commands with the “pipe” symbol

```
ls --all | grep -v *.jpeg
```

- Resource for options
 - `toolname -h`
 - `toolname --help`
 - `man toolname`

LS(1) User Commands LS(1)

NAME
`ls` - list directory contents

SYNOPSIS
`ls [OPTION]... [FILE]...`

DESCRIPTION
List information about the FILES (the current directory by default). Sort entries alphabetically if none of `-cftuvSUX` nor `--sort`.
Mandatory arguments to long options are mandatory for short options too.

-a, --all
do not ignore entries starting with `.`

-A, --almost-all
do not list implied `.` and `..`

--author
with `-l`, print the author of each file

-b, --escape
print C-style escapes for nongraphic characters

--block-size=SIZE
use SIZE-byte blocks. See SIZE format below

-B, --ignore-backups
do not list implied entries ending with `~`

-c
with `-lt`: sort by, and show, ctime (time of last modification of file status information) with `-l`: show ctime and sort by name otherwise: sort by ctime

-C
list entries by columns

--color[=WHEN]
colorize the output. WHEN defaults to 'always' or can be 'never' or 'auto'. More info below

Manual page `ls(1)` line 1

- ls, cp, grep, cat, ... are programs shipped with the Linux distribution
- Third party software needs to be installed separately via
 - Linux Package Manager (administrator) - “App-Store”
 - Custom Package Manager (user)
 - Compilation from source (user, needs compiler and dependencies)
 - may be tricky and error prone

- Conda Is a free, community developed package manager
- Easily install software and dependencies as a user
- Software is retrieved from the online repository “Anaconda cloud”
- ~7500 Bioinformatics tools/libraries can be found in the bioconda channel



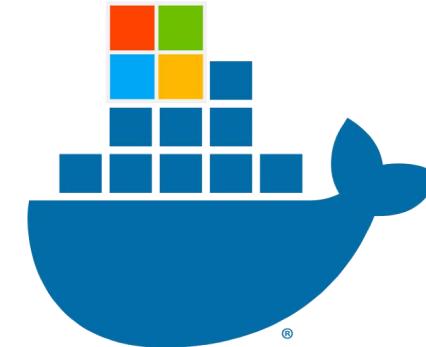
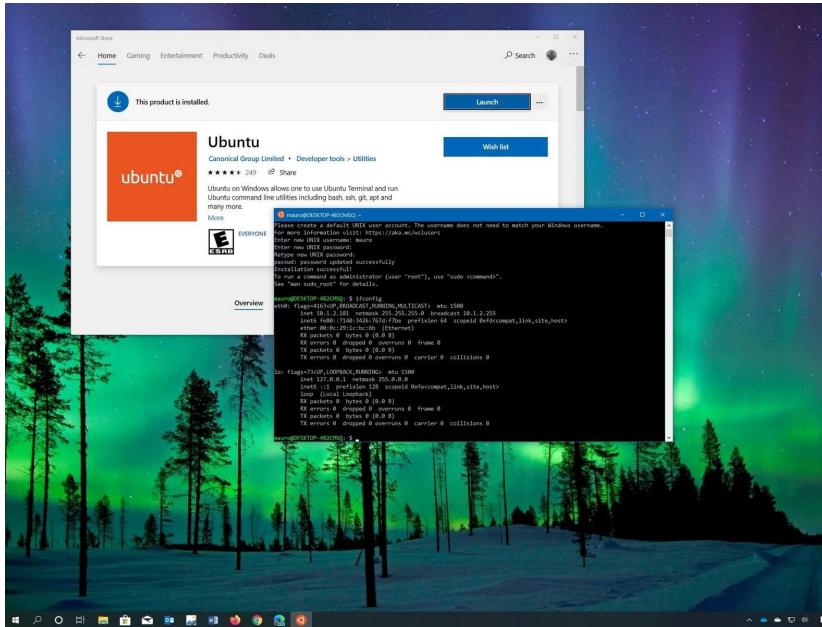
<https://conda.io>

```
bash /path/to/miniconda_setup.sh -b -p /path/to/conda
source /path/to/conda/bin/activate
conda install -c bioconda fastqc samtools trimmomatic bwa
samtools faidx /path/to/genome.fasta
fastqc /path/to/rna-seq.fastq
...

```

- Preconfigure the environment
 - As full grown image of an operating system for a virtual machine
 - Orchestrate container as sandbox environments
 - Azure container service
 - Amazon container server
 - Portainer
 - OpenShift
 - Kubernetes
 - **Docker**

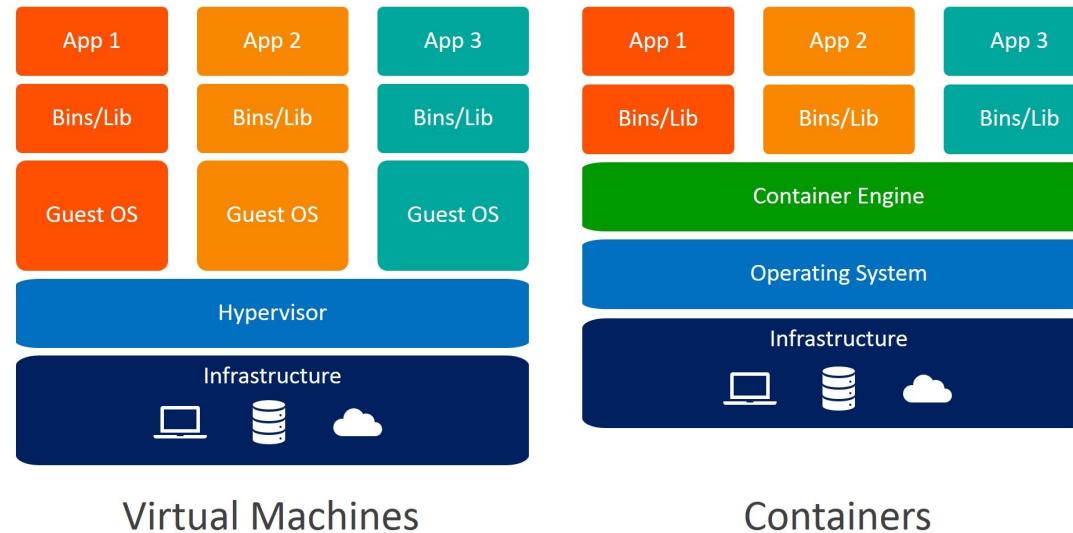
- Ubuntu on Windows 10 Subsystem for Linux (WSL) released 2016
- This significant architectural change as it is a full Linux kernel built by Microsoft, allowing Linux containers to run natively



- Docker is a virtualization technique to setup uniform environments in which necessary software can be installed
- The whole environment can be shipped as an “image”
- Docker images can be executes as “containers” on any operating system (Mac, Win, Linux, Unix)
 - Include own applications and run them everywhere



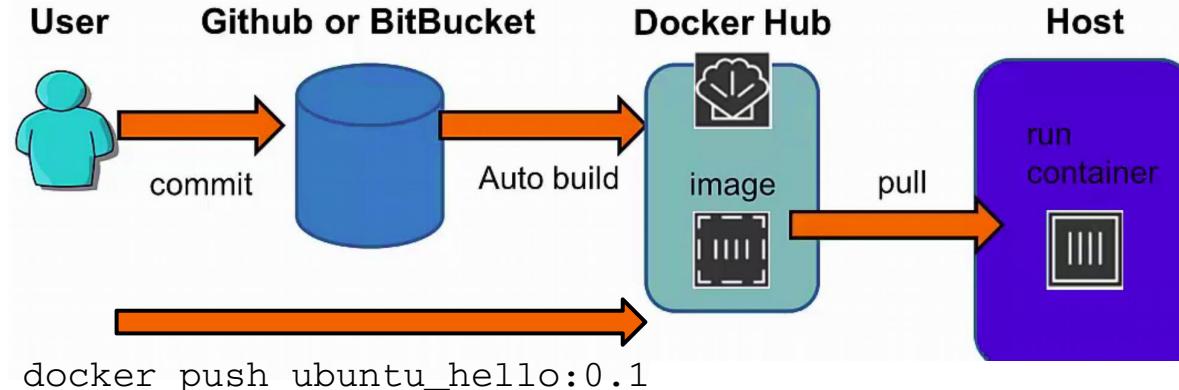
- Advantages of a container over a virtual machine
 - simple setup of a sandbox environment
 - better performance and lightweight due to shared kernel
 - bind mounts
 - no complex configuration of a hypervisor



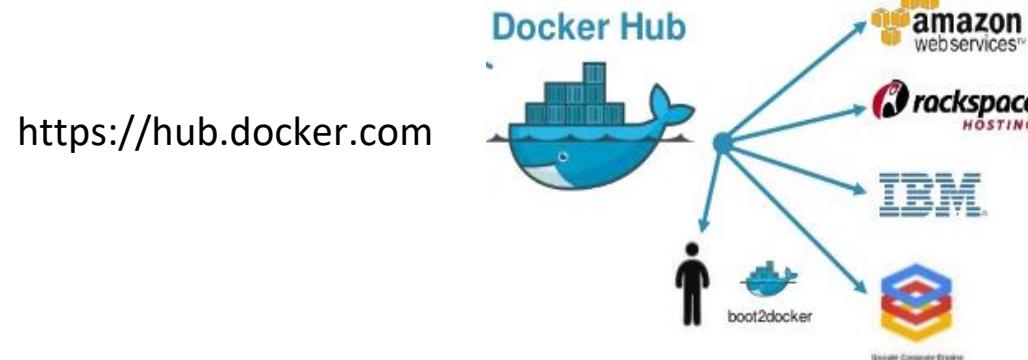
```
# script.sh  
echo "hello world" > /bind/mount/directory/hello.txt
```

```
# Dockerfile  
FROM ubuntu:latest  
ENV PATH=$PATH:/path/to/scripts  
COPY script.sh /path/to/scripts
```

```
docker build -t ubuntu_hello:0.1 /path/to/Dockerfile  
docker run --rm -d -v /local/directory:/bind/mount/directory \  
    ubuntu_hello:0.1 bash script.sh  
cat /local/directory/hello.txt
```



Deployment Platforms



```
docker pull ubuntu:latest          # pull remote image
docker images                      # list local images

docker run --rm ubuntu:latest ls -l    # run container, execute a command and
clean up after exit

docker run -i -t --name ubuntu ubuntu:latest bash # run and enter an interactive bash
# detach with ctr + p , ctr + q

docker run -d -i --name ubuntu ubuntu:latest bash # run interactive container detached
docker ps -a                                # list running and stoped containers

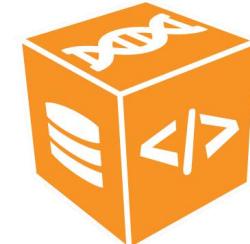
docker exec ubuntu ls -l                      # execute command in running container
docker exec -i -t ubuntu bash                 # open interactive bash to container

docker stop ubuntu                          # stop a running container
docker commit ubuntu ubuntu_modified:0.1     # commit as new image to run later
docker run --rm ubuntu_modified:0.1 ls -l

docker start ubuntu                        # resume a stopped container
docker exec -i -t ubuntu bash             # re-open interactive bash to container

docker rm -f ubuntu                       # nuke container
docker rmi -f ubuntu_modified:0.1         # nuke image
```

- Docker can be integrated into a pipeline
- BioContainers is a resource for dockerized bioinformatics tools



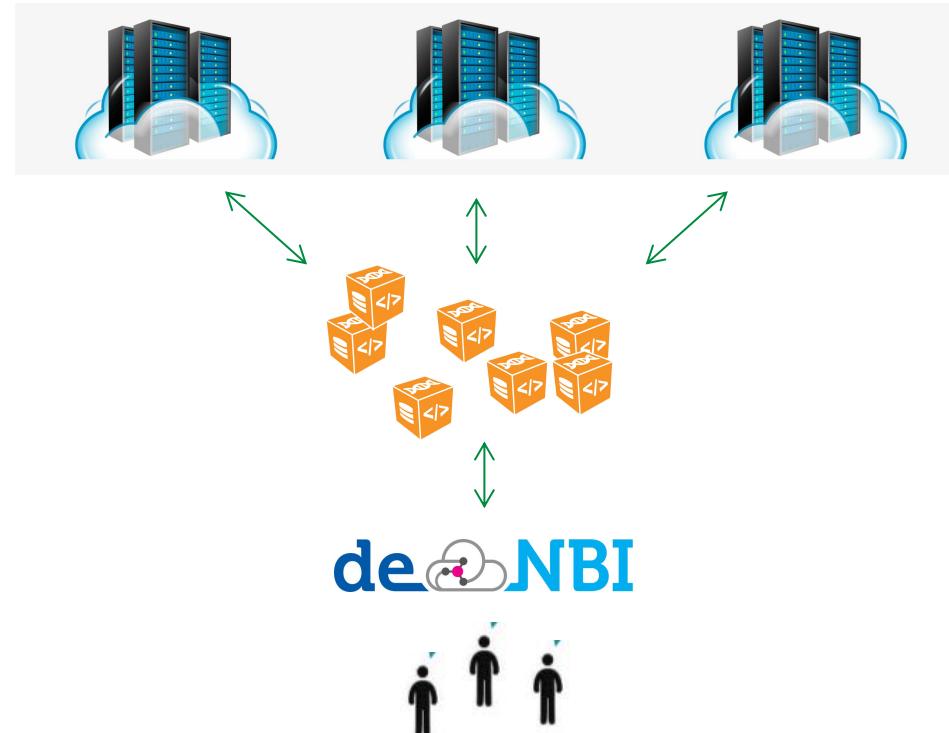
```
docker run --rm quay.io/biocontainers/samtools:1.12 samtools
```

```
cat file.sam | docker run --rm -i quay.io/biocontainers/samtools:1.12
samtools view -f 4 /mnt/file.sam | wc -l > number_unmapped.txt
```

```
docker run --rm -v /path/to/file.sam:/mnt/file.sam \
quay.io/biocontainers/samtools:1.12 \
samtools view -f 4 /mnt/file.sam | wc -l > number_unmapped.txt
```

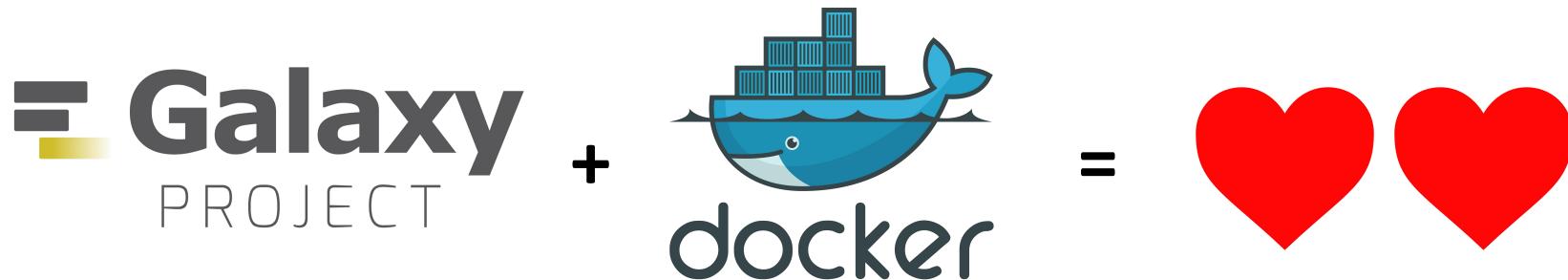
```
docker run --rm -v /path/to/sam-directory:/mnt \
quay.io/biocontainers/samtools:1.12 \
bash -c 'samtools view -f 4 /mnt/file.sam | wc -l > /mnt/number_unmapped.txt'
```

<https://cloud.denbi.de>



Thanks, but I want to know how
to setup a Galaxy instance

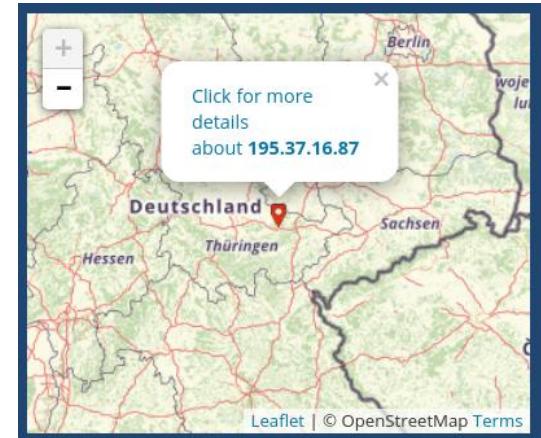




- Galaxy instance on Ubuntu base image is launching
 - BioContainers or tools from its toolshed mainly setpped via Conda
 - FTP-Server
 - Job and process controller and scheduler
 - Webserver
 - Database
 - User interface
 - Interface for interactive environments

- URL  DNS  ip
- localhost: 127.0.0.1
- http: port 80
- https: port 443

Type	Domain Name	IP Address	TTL
A	whatismyipaddress.com	104.16.154.36 Cloudflare, Inc. (AS13335)	5 min
A	whatismyipaddress.com	104.16.155.36 Cloudflare, Inc. (AS13335)	5 min

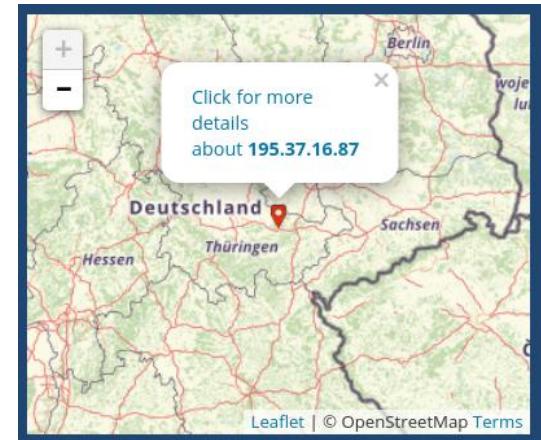


- URL \longleftrightarrow DNS \longleftrightarrow ip
- localhost: 127.0.0.1
- http: port 80
- https: port 443

Type	Domain Name	IP Address	TTL
A	whatismyipaddress.com	104.16.154.36 Cloudflare, Inc. (AS13335)	5 min
A	whatismyipaddress.com	104.16.155.36 Cloudflare, Inc. (AS13335)	5 min

run docker container and remap its http server to be accessible via port 8080

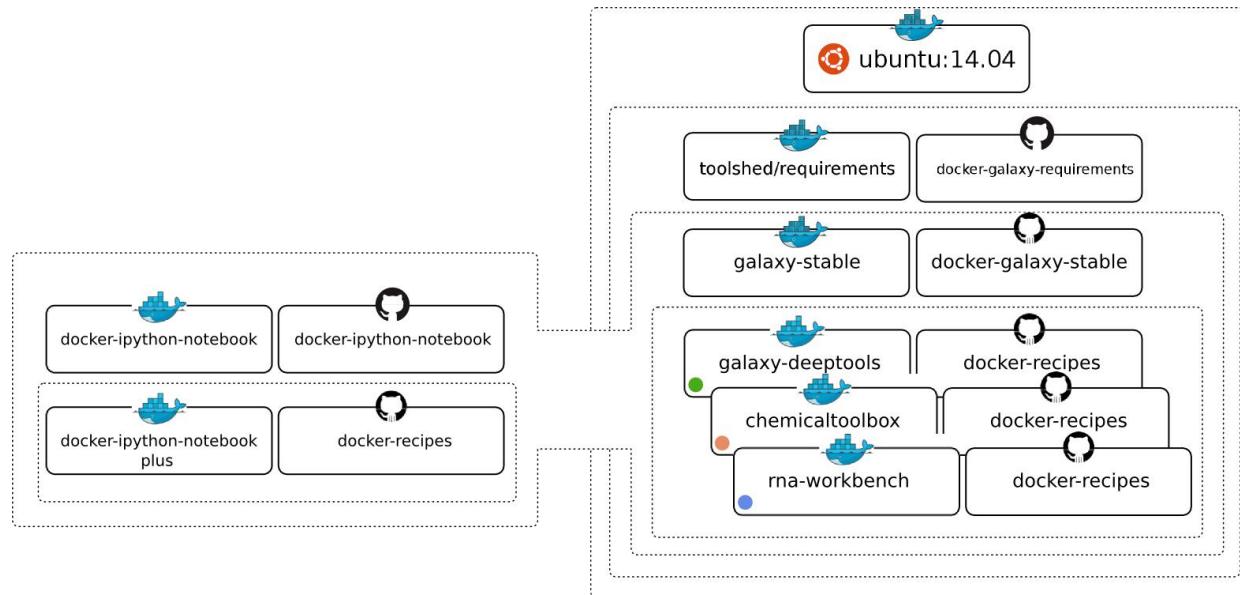
```
docker run -d -p 8080:80 \
quay.io/bgruening/galaxy:19.05
```



http://localhost:8080

The screenshot shows the Galaxy web interface running at <http://localhost:8080>. The page title is "Galaxy". The main content area displays the message "Hello, Galaxy is running!" and instructions to customize the page by editing `static/welcome.html`. It also provides links for "Configuring Galaxy" and "Installing Tools". Below this, there's a link to "Take an interactive tour" with options for "Galaxy UI", "History", and "Scratchbook". A sidebar on the left lists various tools under categories like "Get Data", "Collection Operations", "Text Manipulation", etc., and a "Workflows" section. The right sidebar shows a history panel titled "History" with an "Unnamed history" entry, which is currently empty. A note in the history panel encourages users to load their own data or get data from an external source.

- ..not so empty galaxy flavors



ASaIM (Metagenomics)
Galaxy Epigenetics
HiCExplorer

...

- Its mainly about config files
 - Scheduler configuration
 - List of toolshed tools
 - Galaxy configuration
 - Docker commands

Tools YAML

```
# https://galaxyproject.org/admin/config
#
# Config hackers are encouraged to check there before asking for help.
#
# wsgi:
#
#   The address and port on which to listen. By default, only listen to
#   localhost (galaxy will not be accessible over the network). Use
#   e.g. "0.0.0.0" to listen on all available network interfaces.
#   HTTPS: true
#   port: 19080
#
#   By default uwsgi allocates a very small buffer (4096 bytes) for the
#   headers of each request. If you start receiving "invalid request
#   block size" in your logs, it could mean you need a bigger buffer. We
#   recommend setting this to 16384.
#   buffer-size: 16384
#
#   Number of web server (worker) processes to fork after the
#   application has loaded. If this is set to greater than 1, thunder
#   threads should be enabled below.
#   processes: 1
#
#   Number of threads for each web server process.
#   threads: 4
#
#   Number of threads for serving static content and handling internal
#   routing requests.
#   offload-threads: 2
#
#   Mapping to serve static content.
#   static-map: /static/cfstyle/static/style/blue
#
#   Mapping to serve the remainder of the static content.
#   static-map: /static/cfstatic
#
#   Mapping to serve the favicon.
#   static-map: /favicon/favicon.ico
#
#   Enable the master process manager. Disabled by default for maximum
#   compatibility with CFEngine, but should be enabled for use with
#   Conda and/or production deployments.
#   master: false
#
#   Path to the application's Python virtual environment. If using Conda
#   a folder containing framework dependencies (not tools), do not set this.
#   virtualenv: venv
#
#   Path to the application's Python library.
#   pythonpath: lib
#
#   The entry point which returns the web application (e.g. Galaxy,
#   reports, etc.) that you are loading.
#   module: galaxy.wsgi:galaxy.wsgiapp:uwsgi_app()
```

Galaxy YAML

```
api-key-admin: http://localhost:8080
galaxy_instance_repository_dependencies: true
install_resolver_dependencies: false
install_tool_dependencies: false

tools:
  # quality control
  - name: fastqc
    owner: devteam
    tool_panel_section_label: "Quality Control"
    revisions:
      - revision_id: 1
        tool_panel_section_label: "Quality Control"
        - embedded_tools:
            - fastqc
  - name: trimgalore
    owner: devteam
    tool_panel_section_label: "Quality Control"
    revisions:
      - revision_id: 1
        tool_panel_section_label: "Quality Control"
        - embedded_tools:
            - trim_galore
  - name: primer3plus
    owner: devteam
    tool_panel_section_label: "Quality Control"
    revisions:
      - revision_id: 1
        tool_panel_section_label: "Quality Control"
        - embedded_tools:
            - primer3plus
  - name: samtools
    owner: devteam
    tool_panel_section_label: "Utilities"
    revisions:
      - revision_id: 1
        tool_panel_section_label: "Utilities"
        - embedded_tools:
            - samtools_filter
            - samtools_index
            - samtools_sort
            - samtools_flagstat
            - samtools_faidx
            - samtools_fixmate
            - samtools_regroup
            - deduplicate
```

```
api-key-admin: http://localhost:8080
galaxy_instance_repository_dependencies: true
install_resolver_dependencies: false
install_tool_dependencies: false

tools:
  # quality control
  - name: fastqc
    owner: devteam
    tool_panel_section_label: "Quality Control"
    revisions:
      - revision_id: 1
        tool_panel_section_label: "Quality Control"
        - embedded_tools:
            - fastqc
  - name: trimgalore
    owner: devteam
    tool_panel_section_label: "Quality Control"
    revisions:
      - revision_id: 1
        tool_panel_section_label: "Quality Control"
        - embedded_tools:
            - trim_galore
  - name: primer3plus
    owner: devteam
    tool_panel_section_label: "Quality Control"
    revisions:
      - revision_id: 1
        tool_panel_section_label: "Quality Control"
        - embedded_tools:
            - primer3plus
  - name: samtools
    owner: devteam
    tool_panel_section_label: "Utilities"
    revisions:
      - revision_id: 1
        tool_panel_section_label: "Utilities"
        - embedded_tools:
            - samtools_filter
            - samtools_index
            - samtools_sort
            - samtools_flagstat
            - samtools_faidx
            - samtools_fixmate
            - samtools_regroup
            - deduplicate
```

Dockerfile

Scheduler XML

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <!-- This file is used to define the scheduler and
       the way it interacts with the Galaxy instance. -->
  <!-- See https://galaxyproject.org/admin/config -->
  <!-- for more information. -->
  <!-- The scheduler is defined in the <scheduler> section. -->
  <!-- The Galaxy instance is defined in the <instance> section. -->
  <!-- The scheduler and instance sections are required. -->
  <!-- The <processes> section is optional. -->
  <!-- The <processes> section defines the number of worker
       processes to fork after the application has loaded. -->
  <!-- Thunder threads should be enabled below. -->
  <!-- The <processes> section is optional. -->
  <!-- The <processes> section defines the number of threads for
       each web server process. -->
  <!-- The <processes> section is optional. -->
  <!-- The <processes> section defines the number of threads for
       serving static content and handling internal routing requests. -->
  <!-- The <processes> section is optional. -->
  <!-- The <virtualenv> section defines the path to the application's
       Python virtual environment. If using Conda a folder containing
       framework dependencies (not tools), do not set this. -->
  <!-- The <virtualenv> section is optional. -->
  <!-- The <pythonpath> section defines the path to the application's
       Python library. -->
  <!-- The <pythonpath> section is optional. -->
  <!-- The <module> section defines the entry point which returns the
       web application (e.g. Galaxy, reports, etc.) that you are loading. -->
  <!-- The <module> section is optional. -->
```

Where can I find a blueprint
with simple, but working
config templates





GitHub

destairdenbi / **galaxy-workflow-generator**

forked from [bgruening/docker-galaxy/ngs-preprocessing](#)

```
git clone --recursive \
https://github.com/destairdenbi/galaxy-workflow-generator
```

```
<!DOCTYPE html>
<html lang="en">
  <div align="center">
    
    <h2>Welcome!</h2>
    <p>
      This is my Galaxy instance.
    </p>
  </div>
</html>
```



atoms @ d66a939
tools @ 4b5429a
web
webhooks @ a58f769
workflows @ daa49b2
.gitignore
.gitmodules
.travis.yml
CONTRIBUTING.md
Dockerfile
LICENSE
Makefile
README.md
data_managers.yaml
job_conf.xml
setup.sh
tools.yaml

<https://toolshed.g2.bx.psu.edu>

Repositories




Name	Synopsis
aurora_fastqc	Evaluate short reads with FastQC software on a single or a paired
aurora_fastqc_site	Evaluate short reads with FastQC software on a single or a paired
fastqc	Read QC reports using FastQC
fastqc_stats	Summary multiple FastQC into a single tabular line report
fastqc_workflow	FastQC workflow designed for use with Refinery Platform



Repository revision
21 (2019-05-24) repository tip
Select a revision to inspect and download versions of Galaxy utilities from this repository.

Repository fastqc
Name: fastqc 
Owner: devteam
Synopsis: Read QC reports using FastQC
Detailed description:
FastQC aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.
Content homepage: <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>
Development repository: <https://github.com/galaxyproject/tools-iuc/tree/master/tools/fastqc>
Link to this repository: <https://toolshed.g2.bx.psu.edu/view/devteam/fastqc/e7b2202befea> 
Clone this repository: hg clone <https://toolshed.g2.bx.psu.edu/repos/devteam/fastqc> 
Type: unrestricted
Revision: 21:e7b2202befea



```

11   - name: fastqc
12     owner: devteam
13     tool_panel_section_label: "Quality Control"
14     revisions:
15       - 'e7b2202befea'

```



- [atoms @ d66a939](#)
- [tools @ 4b5429a](#)
- [web](#)
- [webhooks @ a58f769](#)
- [workflows @ daa49b2](#)
- [.gitignore](#)
- [.gitmodules](#)
- [.travis.yml](#)
- [CONTRIBUTING.md](#)
- [Dockerfile](#)
- [LICENSE](#)
- [Makefile](#)
- [README.md](#)
- [data_managers.yaml](#)
- [job_conf.xml](#)
- [setup.sh](#)
- [tools.yaml](#)

<https://docs.galaxyproject.org/en/latest/admin/jobs.html>

- Multithreaded tools should be assigned to “slurmthreaded”

```
32      <destination id="slurm" runner="slurm">
33          <!-- <env file="/galaxy_venv/bin/activate"/> -->
34          <param id="nativeSpecification">--ntasks=1</param>
35      </destination>
36  →  <destination id="slurmthreaded" runner="slurm">
37          <!-- <env file="/galaxy_venv/bin/activate"/> -->
38          <param id="nativeSpecification" from_environ="GALAXY_CONFIG_PARALLEL_SLURM_PARAMS">--ntasks=2</param>
39      </destination>
40
41
42      <tools>
43          →  <tool id="cutadapt" destination="slurmthreaded"/>
44          →  <tool id="trim_galore" destination="slurmthreaded"/>
45          →  <tool id="trimmmomatic" destination="slurmthreaded"/>
46
47
48      <limits>
49          <limit type="walltime">999999:00:00</limit>
50          <limit type="destination_user_concurrent_jobs" id="slurm">999999</limit>
51          <limit type="destination_user_concurrent_jobs" id="slurmthreaded">999999</limit>
52          <limit type="destination_user_concurrent_jobs" id="local">999999</limit>
53          <limit type="destination_user_concurrent_jobs" id="localthreaded">999999</limit>
54
55      </limits>
```

atoms @ d66a939
tools @ 4b5429a
web
webhooks @ a58f769
workflows @ daa49b2
.gitignore
.gitmodules
.travis.yml
CONTRIBUTING.md
Dockerfile
LICENSE
Makefile
README.md
data_managers.yaml
job_conf.xml
setup.sh
tools.yaml

- create a sub-directory with unique name
 - place in the binary
 - create a .xml file named like the directory itself
 - for data types and more see <https://planemo.readthedocs.io>

```
<tool id="unique_name" name="actual tool name" version="0.0.1">
    <requirements></requirements>
    <stdio><exit_code range="1:" /></stdio>
    <command><![CDATA[
        binary "$input_1" "$input_2" "$output"
    ]]></command>
    <inputs>
        <param type="data" name="input_1" format="csv" label="input table" />
        <param type="integer" name="input_2" value="0" label="input value" />
    </inputs>
    <outputs>
        <data name="output" format="csv" label="output table"/>
    </outputs>
    <tests></tests>
    <help>![CDATA[
        my binary does something
    ]]></help>
</tool>
```



atoms @ d66a939
tools @ 4b5429a
web
webhooks @ a58f769
workflows @ daa49b2
.gitignore
.gitmodules
.travis.yml
CONTRIBUTING.md
Dockerfile
LICENSE
Makefile
README.md
data_managers.yaml
job_conf.xml
setup.sh
tools.yaml

```
docker build -t my_galaxy:0.1 /path/to/Dockerfile  
docker run -d -p 8080:80 --name galaxy my_galaxy:0.1
```

enable multithreading

```
[...] -e "GALAXY_CONFIG_PARALLEL_SLURM_PARAMS=--ntasks=8"  
-e "GALAXY_CONFIG_PARALLEL_LOCAL_NTASKS=8" [...]
```

bind mount location to persistent database

```
[...] -v /path/to/local/directory:/export [...] 
```

atoms @ d66a939
tools @ 4b5429a
web
webhooks @ a58f769
workflows @ daa49b2
.gitignore
.gitmodules
.travis.yml
CONTRIBUTING.md
Dockerfile
LICENSE
Makefile
README.md
data_managers.yaml
job_conf.xml
setup.sh
tools.yaml

```
docker exec galaxy bash -c 'echo $GALAXY_CONFIG_FILE'  
docker cp galaxy:/etc/galaxy/galaxy.yml /path/to/galaxy.yml
```

edit galaxy.yml - e.g. set require_login to false

```
docker cp /path/to/galaxy.yml galaxy:/etc/galaxy/galaxy.yml  
docker exec galaxy supervisorctl restart galaxy:
```

refresh browser

sometimes it is necessary to delete cookies and cache



Galaxy / de.STAIR

Tools

- search tools

Upload Data

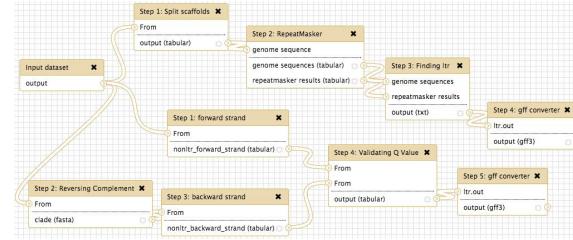
Visualization

Import the sequence datasets

Import data into Galaxy.

« Prev Next » End tour

TO START, CLICK ON



Galaxy

Analyze Data Workflow Shared Data Visualization Help User

Tools

- search tools
- Get Data
- Send Data
- ENCODE Tools
- Lift-Over
- Text Manipulation
- Convert Formats
- FASTA manipulation
- Filter and Sort
- Join, Subtract and Group
- Extract Features
- Fetch Sequences
- Fetch Alignments
- Get Genomic Scores
- Operate on Genomic Intervals
- Statistics
- Graph/Display Data
- Regional Variation
- Multiple regression
- Multivariate Analysis
- Evolution
- Motif Tools
- Multiple Alignments
- Metagenomic analyses
- Human Genome Variation
- Genome Diversity
- EMBOSS
- NGS TOOLBOX BETA

Edit Attributes

Name: Join two Queries on data 3 and data 1

Info:

Database/Build: Click to Search or Select

Number of comment lines:

Save

Auto-detect

This will inspect the dataset and attempt to correct the above column values if they are not accurate.

Change data type

New Type: tabular

This will change the datatype of the existing dataset but not modify its contents. Use this if Galaxy has incorrectly guessed the type of your dataset.

Save

History

1.8 Gb

- 14: Draw phylogeny on data 12
- 13: Draw phylogeny on data 11
- 12: Find lowest diagnostic rank on data 10
- 11: Find lowest diagnostic rank on data 9
- 10: Fetch taxonomic representation on data 8
- 9: Fetch taxonomic representation on data 7
- 8: s234 within 5% of max
- 7: s1 within 5% of max
- 6: Join two Queries on data 4 and data 2
- 5: Join two Queries on data 3 and data 1
- 4: s234 max bit score



fli

deSTAIR

Leibniz Institute on Aging –
Fritz Lipmann Institute

Universität
Rostock



Traditio et Innovatio

deNBI

GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE



Bundesministerium
für Bildung
und Forschung

ptj
Projekträger Jülich
Forschungszentrum Jülich



UNI
FREIBURG

We thank the Freiburg Galaxy Team and all Galaxy community trainers and developers for their great work.

We also thank the BMBF for the de.NBI-Partner program (Grant 031L0106) and the European Social Fund (ESF/14-BM-A55-0027/18).⁶³