

Formation intégration d'outils sous Galaxy

24/02/2015

Cyril Monjeaud & Yvan Le Bras

Plateforme Bio-informatique GenOuest

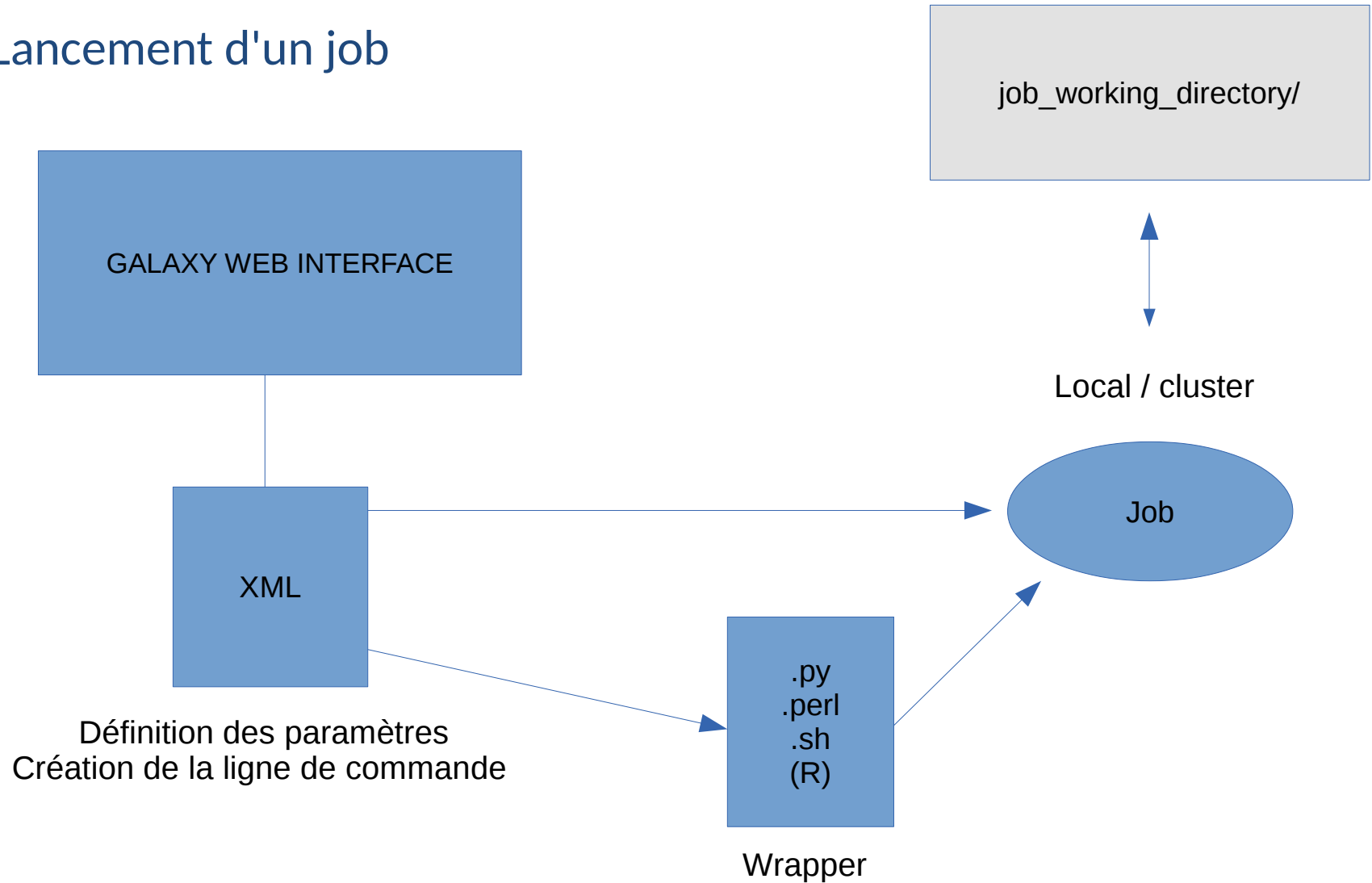
CNRS UMR 6074 IRISA-INRIA, Campus de Beaulieu, 35042 Rennes Cedex

http://github.com/cmonjeau/formation_galaxy

PLAN

- L'arborescence du serveur Galaxy
 - L'ajout d'outil dans Galaxy
 - Les descripteurs et la syntaxe
 - TP0. L'environnement de travail
 - TP1-5. Intégration d'outils “simples”
-
- TP6. Intégration d'un outils “complexe”
 - Présentation du toolshed
 - TP7. Déploiement d'un toolshed
 - TP8. Intégration d'un outil dans un toolshed
 - TP9. Intégration d'un outil avec une dépendance
 - Présentation du système de gestion des banques (by Anthony Brétaudeau)
 - TP10. Intégration d'un outil utilisant des banques (by Anthony Brétaudeau)

Lancement d'un job



L'arborescence d'un serveur Galaxy

La racine du server galaxy-dist/

```
/opt/galaxy-dist/  
├── buildbot_setup.sh  
├── CITATION  
├── client/  
├── config/  
├── contrib/  
├── create_db.sh*  
├── cron/  
├── database/  
├── display_applications/  
├── dist-eggs.ini  
├── doc/  
├── eggs/  
├── eggs.ini  
├── external_service_types/  
├── external_service_types_conf.xml  
├── extract_dataset_parts.sh*  
├── install_and_test_tool_shed_repositories.sh  
├── integrated_tool_panel.xml  
├── lib/  
├── LICENSE.txt  
├── locale/ └── manage_db.sh*  
├── manage_tools.sh  
├── openid/  
├── openid_conf.xml  
├── paster.log  
├── paster.pid  
├── README.txt  
└── rolling_restart.sh*
```

```
├── run_demo_sequencer.sh  
├── run_functional_tests.sh -> run_tests.sh*  
├── run_reports.sh*  
├── run.sh*  
├── run.sh.orig*  
├── run_tests.sh*  
├── run_tool_shed.sh*  
├── scripts/  
├── set_metadata.sh*  
├── static/  
├── templates/  
├── test/  
├── test-data/  
├── tool-data/  
├── tool_list.py  
└── tools/
```

- run.sh
 - permet de lancer le serveur Galaxy
- paster.log
 - fichier de log de Galaxy
- config/
 - configuration générales
 - liste des outils
 - lancement des jobs, etc.
- tools/
 - contient les outils de l'instance locale
- database/
 - contient les données (inputs / outputs)

Le dossier galaxy-dist/config/

/opt/galaxy-dist/config/

- data_manager_conf.xml.sample
- datatypes_conf.xml.sample
- demo_sequencer_wsgi.ini.sample
- disposable_email_blacklist.conf.sample
- external_service_types_conf.xml.sample
- **galaxy.ini**
- galaxy.ini.sample
- **job_conf.xml**
- job_conf.xml.sample_advanced
- job_conf.xml.sample_basic
- job_metrics_conf.xml.sample
- job_resource_params_conf.xml.sample
- migrated_tools_conf.xml
- migrated_tools_conf.xml.sample
- object_store_conf.xml.sample
- openid_conf.xml.sample
- plugins/
- reports_wsgi.ini.sample
- shed_data_manager_conf.xml
- shed_data_manager_conf.xml.sample
- **shed_tool_conf.xml**
- shed_tool_conf.xml.sample
- shed_tool_data_table_conf.xml
- shed_tool_data_table_conf.xml.sample
- **tool_conf.xml**
- tool_conf.xml.main
- tool_conf.xml.sample
- tool_data_table_conf.xml.sample

- tool_shed.ini.sample
- tool_sheds_conf.xml
- tool_sheds_conf.xml.sample
- workflow_schedulers_conf.xml.sample

- **galaxy.ini**
 - configuration générale du serveur Galaxy
- **tool_conf.xml**
 - liste des outils locaux de Galaxy
- **shed_tool_conf.xml**
 - liste des outils installés via un toolshed
- **job_conf.xml**
 - définition du lancement des jobs
 - environnement embarqué
 - exécution en local
 - exécution sous docker
 - exécution sur un cluster
 - drmaa
 - pbs

Le dossier galaxy-dist/tools/

```
/opt/galaxy-dist/tools/
├── cyril_tools/
│   ├── appendv2.py
│   ├── appendv2.xml
│   ├── append.xml
│   ├── convert.py
│   ├── convert.xml
│   ├── resize.xml
│   ├── rotate.sh
│   └── rotate.xml
├── evolution/
│   ├── add_scores.py*
│   ├── add_scores.xml
│   ├── codingSnps_filter.py*
│   ├── codingSnps.pl*
│   └── codingSnps.xml
├── extract/
│   ├── extract_genomic_dna.py
│   ├── extract_genomic_dna.xml
│   ├── liftOver_wrapper.py
│   └── liftOver_wrapper.xml
├── meme/
│   ├── fimo_wrapper.py
│   ├── fimo.xml
│   └── meme.xml
└── next_gen_conversion/
    ├── bwa_solid2fastq_modified.pl*
    ├── fastq_conversions.py
    └── fastq_conversions.xml
```

```
├── fastq_gen_conv.py
├── fastq_gen_conv.xml
├── solid2fastq.py
├── solid2fastq.xml
├── solid_to_fastq.py
├── solid_to_fastq.xml
└── visualization/
    ├── LAJ_code.py
    ├── LAJ.py
    └── LAJ.xml
```

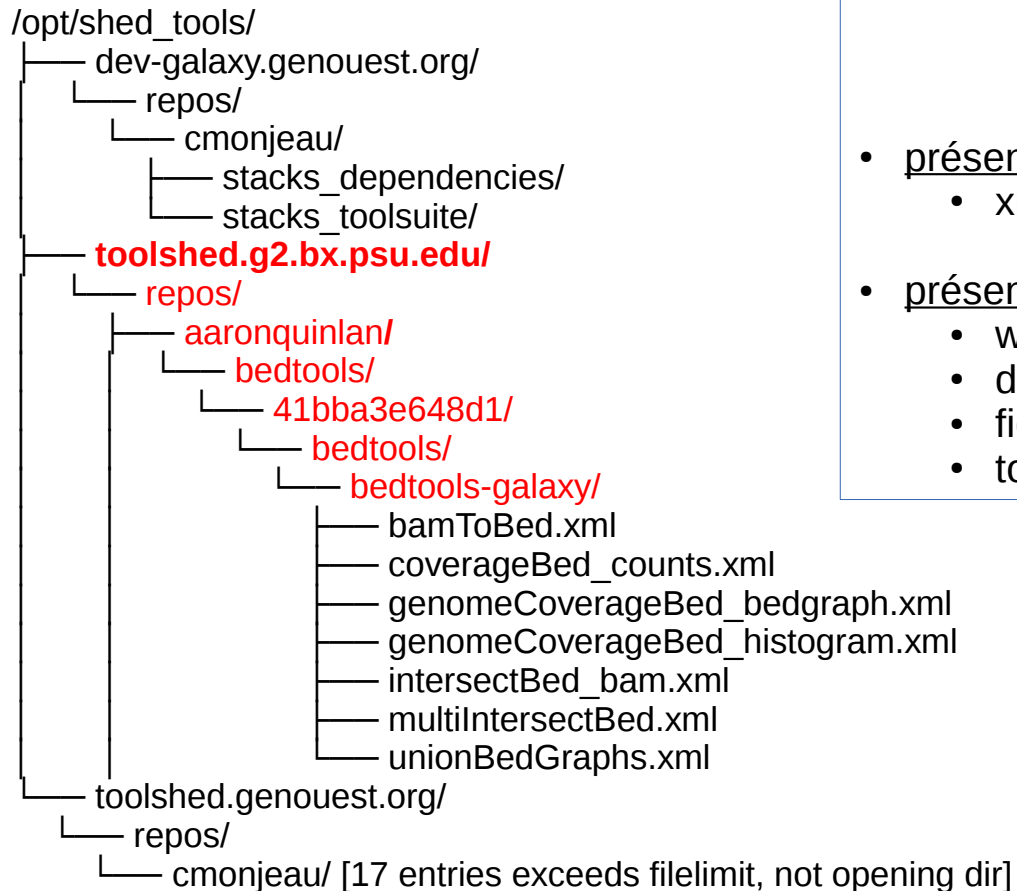
- trié en sous-dossier
- présence d'un descripteur par outil
 - xml
- présence (facultative) d'un wrapper associé
 - bash
 - python
 - perl
 - R

Le dossier galaxy-dist/database/

```
/opt/galaxy-dist/database/
├── citations/
├── data/
│   ├── container_file/
│   └── locks/
│       ├── 4/
│       └── e/
├── compiled_templates/ [36 entries exceeds filelimit, not opening dir]
├── files/
│   ├── 000/ [99 entries exceeds filelimit, not opening dir]
│   └── 001/ [74 entries exceeds filelimit, not opening dir]
├── info.txt
├── job_working_directory/
│   └── 000/
├── object_store_cache/
├── pbs/
├── tmp/ [96 entries exceeds filelimit, not opening dir]
└── whoosh_indexes/
```

- files/
 - localisation des fichiers (entrées / sorties)
 - rangés en sous-dossiers
 - 000/
 - 001/
 - etc.
 - nom de fichier : dataset_001.dat
- job_working_dir/
 - lieu de l'exécution des jobs
- tmp/
 - Dossier temporaire (utilisé pour multi-output)

Le dossier /opt/shed_tools





- architecture spécialisée
 - Source du toolshed
 - “repos”
 - owner du repository
 - nom du repository
 - revision du repository
 - nom du repository
 - fichiers
- présence d'un descripteur par outil
 - xml
- présence (facultative) de fichiers
 - wrappers
 - datatypes.xml.sample
 - fichiers .loc.sample
 - tool_data_table_conf.xml.sample

L'ajout d'outil dans Galaxy

Que faut-il faire pour ajouter un outil ?

- Ajout d'un outil local :
 - création / récupération de l'outil
 - descripteur
 - [wrapper]
 - ajout des fichiers dans le dossier tools/
 - ajout du chemin du descripteur dans config/**tool_conf.xml**
 - **Redémarrage du serveur**
- Ajout d'un outil présent dans un toolshed :
 - chercher son outil dans le toolshed (via admin)
 - suivre la procédure
 - Au final :
 - les fichiers sont téléchargés
 - le descripteur de l'outil est ajouté dans le fichier config/**shed_tool_conf.xml**

Tools 

search tools 

Get Data

Send Data

Lift-Over
[Convert genome coordinates between assemblies and genomes](#)

Text Manipulation

Filter and Sort

Join, Subtract and Group

Convert Formats

Extract Features

Fetch Sequences

Fetch Alignments

Get Genomic Scores

Statistics

Graph/Display Data

Evolution

Motif Tools

NGS: QC and manipulation

NGS: Mapping

NGS: RNA Analysis

NGS: SAM Tools

NGS: Simulation

Phenotype Association

NCBI Blast+

NGS: Bed Tools

NGS: Assembly

NGS: SNP Analysis

STACKS toolsuite

Archives Manipulation

Workflows

- [All workflows](#)

tool_conf.xml

```
<label id="monlabel" text="MonTitre" />
<section id="masection" name="MonTitreDeSection">
  <tool file="tools/descripteur.xml" />
</section>
```

tools/descripteur.xml

shed_tool_conf.xml

```
<section id="gettext" name="Get Data" version="">
  <tool file="cloud-
45.genouest.org/toolshed/repos/cmonjeau/test/e0266fbac9ec/test/discoSNP.xml"
guid="cloud-45.genouest.org/toolshed/repos/cmonjeau/test/discosnp/1.0.0">
    <tool_shed>cloud-45.genouest.org/toolshed</tool_shed>
    <repository_name>test</repository_name>
    <repository_owner>cmonjeau</repository_owner>
    <installed_changeset_revision>e0266fbac9ec</installed_changeset_revision>
    <id>cloud-45.genouest.org/toolshed/repos/cmonjeau/test/discosnp/1.0.0</id>
    <version>1.0.0</version>
  </tool>
</section>
```

Les descripteurs & la syntaxe

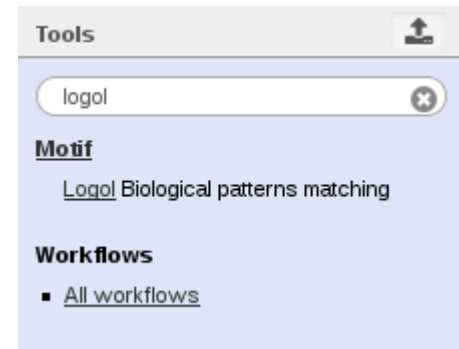
<http://wiki.galaxyproject.org/Admin/Tools/ToolConfigSyntax>

Un descripteur?

- Fichier xml (balises)
- **But** : création d'une ligne de commande
- **Contenu** :
 - Description de l'outil
 - définition des données “input”
 - data (présent dans l'historique de Galaxy)
 - boolean
 - integer
 - float
 - select
 - définir les données “output”
 - data (présent dans l'historique de Galaxy)
 - champ d'aide
- **Apparaître sous la forme d'un formulaire** -> interaction avec l'utilisateur

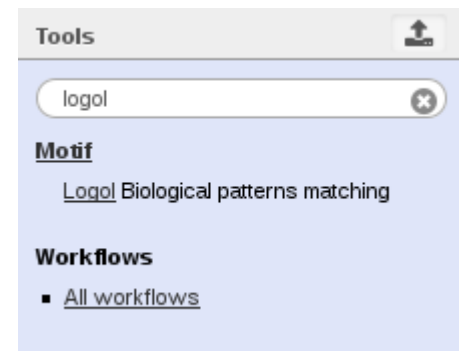
Balises les plus importantes

```
<tool id="logol_wrapper" name="Logol">
  <description>Biological patterns matching</description>
  <command interpreter="bash">
    logol.sh $options.input.options_input_selector $option_input.input_model $type > $match
  </command>
  <inputs>
    <conditional name="options_input">
      <param name="options_input_selector" type="select" label="Logol pattern type">
        <option value="model" selected="True">Logol pattern model</option>
        <option value="grammar">Logol pattern grammar</option>
      </param>
      <when value="model">
        <param name="input_model" type="data" format="lgd" label="Logol pattern model" help="Pattern model designed by LogolDesigner" />
      </when>
    </conditional>
    <param name="type" type="select" format="text" label="type">
      <option value="dna">DNA</option>
      <option value="rna">RNA</option>
      <option value="protein">PROTEIN</option>
    </param>
    <param name="max" type="integer" value="100" label="Maximum number of result matches"/>
    <param name="fasta" type="boolean" label="Add fasta conversion to result archive" checked="False" value="False"/>
  </inputs>
  <outputs>
    <data format="text" name="match" label="match : ${tool.name} on ${on_string}" />
  </outputs>
  <help>
  </help>
</tool>
```



Syntaxes les plus importantes

```
<tool id="logol_wrapper" name="Logol">
  <description>Biological patterns matching</description>
  <command interpreter="bash">
    logol.sh $options.input.options_input_selector $option_input.input_model $type > $match
  </command>
  <inputs>
    <conditional name="options_input">
      <param name="options_input_selector" type="select" label="Logol pattern type">
        <option value="model" selected="True">Logol pattern model</option>
        <option value="grammar">Logol pattern grammar</option>
      </param>
      <when value="model">
        <param name="input_model" type="data" format="lgd" label="Logol pattern model" help="Pattern model designed by LogolDesigner" />
      </when>
    </conditional>
    <param name="type" type="select" format="text" label="type">
      <option value="dna">DNA</option>
      <option value="rna">RNA</option>
      <option value="protein">PROTEIN</option>
    </param>
    <param name="max" type="integer" value="100" label="Maximum number of result matches"/>
    <param name="fasta" type="boolean" label="Add fasta conversion to result archive" checked="False" value="False"/>
  </inputs>
  <outputs>
    <data format="text" name="match" label="match : ${tool.name} on ${on_string}" />
  </outputs>
  <help>
  </help>
</tool>
```



Syntaxes les plus importantes

1 ligne de commande :

- retour ligne = espace
- conditionnelle = #if ... #endif
- commentaires = ##mon commentaire

Cheetah : possibilité de faire du code python

```
<tool id="logol_wrapper" name="Logol">
  <description>Biological patterns matching</description>
  <command interpreter="bash">
    logol.sh $options.input.options_input_selector $option_input.input_model $type > $match
  </command>
  <inputs>
    <conditional name="options_input">
      <param name="options_input_selector" type="select" label="Logol pattern type">
        <option value="model" selected="True">Logol pattern model</option>
        <option value="grammar">Logol pattern grammar</option>
      </param>
      <when value="model">
        <param name="input_model" type="data" format="lgd" label="Logol pattern model" help="Pattern model designed by LogolDesigner" />
      </when>
    </conditional>
    <param name="type" type="select" format="text" label="type">
      <option value="dna">DNA</option>
      <option value="rna">RNA</option>
      <option value="protein">PROTEIN</option>
    </param>
    <param name="max" type="integer" value="100" label="Maximum number of result matches"/>
    <param name="fasta" type="boolean" label="Add fasta conversion to result archive" checked="False" value="False"/>
  </inputs>
  <outputs>
    <data format="text" name="match" label="match : ${tool.name} on ${on_string}" />
  </outputs>
  <help>
  </help>
</tool>
```

Syntaxes les plus importantes

```
<tool id="logol_wrapper" name="Logol">
  <description>Biological patterns matching</description>
  <command interpreter="bash">
    logol.sh $options.input.options_input_selector $option_input.input_model $type > $match
  </command>
  <inputs>
    <conditional name="options_input">
      <param name="options_input_selector" type="select" label="Logol pattern type">
        <option value="model" selected="True">Logol pattern model</option>
        <option value="grammar">Logol pattern grammar</option>
      </param>
      <when value="model">
        <param name="input_model" type="data" format="lgd" label="Logol pattern model" />
      </when>
    </conditional>
    <param name="type" type="select" format="text" label="type">
      <option value="dna">DNA</option>
      <option value="rna">RNA</option>
      <option value="protein">PROTEIN</option>
    </param>
    <param name="max" type="integer" value="100" label="Maximum number of result matches"/>
    <param name="fasta" type="boolean" label="Add fasta conversion to result archive" checked="False" value="False"/>
  </inputs>
  <outputs>
    <data format="text" name="match" label="match : ${tool.name} on ${on_string}" />
  </outputs>
  <help>
  </help>
</tool>
```

Logol pattern type:
 Logol pattern model

Logol pattern model:
 Pattern model designed by LogolDesigner

Type of personal data file:
 DNA

Target sequence(s) to analyse:
 Personal fasta sequence

Read from file: 21: fasta: Check_sense on data 17
 Fasta sequence to analyse

Maximum number of result matches:
 100

Add fasta conversion to result archive:
☐

Add gff conversion to result archive:
☐

Search sequence in both directions:
☐

Maximum size of a match:

Maximum size of a spacer:

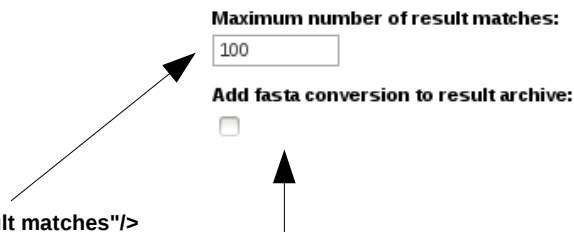
Maximum size of a word:

Minimum size of a word:

Allow sequence cut (if several models are defined in rule, all models will look for pattern in same sequence range):
☐

Syntaxes les plus importantes

```
<tool id="logol_wrapper" name="Logol">
  <description>Biological patterns matching</description>
  <command interpreter="bash">
    logol.sh $options.input.options_input_selector $option_input.input_model $type > $match
  </command>
  <inputs>
    <conditional name="options_input">
      <param name="options_input_selector" type="select" label="Logol pattern type">
        <option value="model" selected="True">Logol pattern model</option>
        <option value="grammar">Logol pattern grammar</option>
      </param>
      <when value="model">
        <param name="input_model" type="data" format="lgd" label="Logol pattern model" help="Pattern model designed by LogolDesigner" />
      </when>
    </conditional>
    <param name="type" type="select" format="text" label="type">
      <option value="dna">DNA</option>
      <option value="rna">RNA</option>
      <option value="protein">PROTEIN</option>
    </param>
    <param name="max" type="integer" value="100" label="Maximum number of result matches"/>
    <param name="fasta" type="boolean" label="Add fasta conversion to result archive" checked="False" value="False"/>
  </inputs>
  <outputs>
    <data format="text" name="match" label="match : ${tool.name} on ${on_string}" />
  </outputs>
  <help>
  </help>
</tool>
```

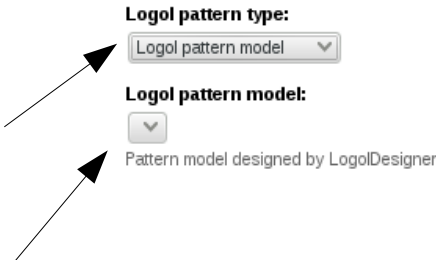


Maximum number of result matches:
100

Add fasta conversion to result archive:
☐

Syntaxes les plus importantes

```
<tool id="logol_wrapper" name="Logol">
  <description>Biological patterns matching</description>
  <command interpreter="bash">
    logol.sh $options.input.options_input_selector $option_input.input_model $type > $match
  </command>
  <inputs>
    <conditional name="options_input">
      <param name="options_input_selector" type="select" label="Logol pattern type">
        <option value="model" selected="True">Logol pattern model</option>
        <option value="grammar">Logol pattern grammar</option>
      </param>
      <when value="model">
        <param name="input_model" type="data" format="lgd" label="Logol pattern model" help="Pattern model designed by LogolDesigner" />
      </when>
    </conditional>
    <param name="type" type="select" format="text" label="type">
      <option value="dna">DNA</option>
      <option value="rna">RNA</option>
      <option value="protein">PROTEIN</option>
    </param>
    <param name="max" type="integer" value="100" label="Maximum number of result matches"/>
    <param name="fasta" type="boolean" label="Add fasta conversion to result archive" checked="False" value="False"/>
  </inputs>
  <outputs>
    <data format="text" name="match" label="match : ${tool.name} on ${on_string}" />
  </outputs>
  <help>
  </help>
</tool>
```



Logol pattern type:

Logol pattern model:

Pattern model designed by LogolDesigner

Syntaxes les plus importantes

```
<tool id="logol_wrapper" name="Logol">
  <description>Biological patterns matching</description>
  <command interpreter="bash">
    logol.sh $options.input.options_input_selector $option_input.input_model $type > $match
  </command>
  <inputs>
    <conditional name="options_input">
      <param name="options_input_selector" type="select" label="Logol pattern type">
        <option value="model" selected="True">Logol pattern model</option>
        <option value="grammar">Logol pattern grammar</option>
      </param>
      <when value="model">
        <param name="input_model" type="data" format="lgd" label="Logol pattern model" help="Pattern model designed by LogolDesigner" />
      </when>
    </conditional>
    <param name="type" type="select" format="text" label="type">
      <option value="dna">DNA</option>
      <option value="rna">RNA</option>
      <option value="protein">PROTEIN</option>
    </param>
    <param name="max" type="integer" value="100" label="Maximum number of result matches"/>
    <param name="fasta" type="boolean" label="Add fasta conversion to result archive" checked="False" value="False"/>
  </inputs>
  <outputs>
    <data format="text" name="match" label="match : ${tool.name} on ${on_string}" />
  </outputs>
  <help>
  </help>
</tool>
```



Syntaxes les plus importantes

```
<tool id="logol_wrapper" name="Logol">
  <description>Biological patterns matching</description>
  <command interpreter="bash">
    logol.sh $options.input.options_input_selector $option_input.input_model $type > $match
  </command>
  <inputs>
    <conditional name="options_input">
      <param name="options_input_selector" type="select" label="Logol pattern type">
        <option value="model" selected="True">Logol pattern model</option>
        <option value="grammar">Logol pattern grammar</option>
      </param>
      <when value="model">
        <param name="input_model" type="data" format="lgd" label="Logol pattern model" help="Pattern model designed by LogolDesigner" />
      </when>
    </conditional>
    <param name="type" type="select" format="text" label="type">
      <option value="dna">DNA</option>
      <option value="rna">RNA</option>
      <option value="protein">PROTEIN</option>
    </param>
    <param name="max" type="integer" value="100" label="Maximum number of result matches"/>
    <param name="fasta" type="boolean" label="Add fasta conversion to result archive" checked="False" value="False"/>
  </inputs>
  <outputs>
    <data format="text" name="match" label="match : ${tool.name} on ${on_string}" />
  </outputs>
  <help>
  </help>
</tool>
```



Syntaxes les plus importantes

```
<tool id="logol_wrapper" name="Logol">
  <description>Biological patterns matching</description>
  <command interpreter="bash">
    logol.sh $options.input.options_input_selector $option_input.input_model $type > $match
  </command>
  <inputs>
    <conditional name="options_input">
      <param name="options_input_selector" type="select" label="Logol pattern type">
        <option value="model" selected="True">Logol pattern model</option>
        <option value="grammar">Logol pattern grammar</option>
      </param>
      <when value="model">
        <param name="input_model" type="data" format="lgd" label="Logol pattern model" help="Pattern model designed by LogolDesigner" />
      </when>
    </conditional>
    <param name="type" type="select" format="text" label="type">
      <option value="dna">DNA</option>
      <option value="rna">RNA</option>
      <option value="protein">PROTEIN</option>
    </param>
    <param name="max" type="integer" value="100" label="Maximum number of result matches"/>
    <param name="fasta" type="boolean" label="Add fasta conversion to result archive" checked="False" value="False"/>
  </inputs>
  <outputs>
    <data format="text" name="match" label="match : ${tool.name} on ${on_string}" />
  </outputs>
  <help>
  </help>
</tool>
```

⚠ Double equal signs, ==, must be used as "equal to" (e.g., `c1 == 'chr22'`)

💡 **TIP:** Attempting to apply a filtering condition may throw exceptions if the data type (e.g., string, integer) in every line of the column a line, that line is skipped as invalid for the filter condition. The number of invalid skipped lines is documented in the resulting history.

💡 **TIP:** If your data is not TAB delimited, use *Text Manipulation->Convert*

Syntax

The filter tool allows you to restrict the dataset using simple conditional statements.

Columns are referenced with **c** and a **number**. For example, `c1` refers to the first column of a tab-delimited file. Make sure that multi-character operators contain no white space (e.g., `<=` is valid while `<=` is not valid).

When using 'equal-to' operator **double equal sign '==' must be used** (e.g., `c1=='chr1'`).

Non-numerical values must be included in single or double quotes (e.g., `c6=='*'`).

Filtering condition can include logical operators, but **make sure operators are all lower case** (e.g., `(c1!='chrX' and c1!='chrY')`).

Example

`c1=='chr1'` selects lines in which the first column is chr1

`c3-c2<100*c4` selects lines where subtracting column 3 from column 2 is less than the value of column 4 times 100

`len[c2.split(',')]<4` will select lines where the second column has less than four comma separated elements

`c2>=1` selects lines in which the value of column 2 is greater than or equal to 1

Numbers should not contain commas - `c2<=44,554,350` will not work, but `c2<=44554350` will

Some words in the data can be used, but must be single or double quoted (e.g., `c3=='exon'`)

TP0 : l'environnement de travail

Genocloud

Cf document annexe

TP1->5 : intégration d'outils « simples »

Transformation d'image

Quelques infos à savoir

- Ajout d'un nouvel outil, redémarrage nécessaire :
 - `sh /opt/galaxy-dist/run.sh --stop-daemon`
 - `sh /opt/galaxy-dist/run.sh --daemon`
- Modification du descripteur d'un outil :
 - Dans l'onglet **admin** - « reload a tool configuration »
- Modification du wrapper :
 - Rien à faire
- Si le descripteur xml n'est pas valide, l'outil ne sera pas chargé :
 - consultation du fichier paster.log : `cat /opt/galaxy-dist/paster.log | grep mon_outil`

La commande “convert”

- Utilisation de la commande “convert”
- Nécessite le package imagemagick (apt-get install imagemagick)
- Utile pour le traitement d'image
- Dans cette partie :

- Outil de redimensionnement d'images

`convert input_file -resize percentage output_file`

- Outil de rotation d'images

`convert input_file -rotate angle output_file`

- Outil de collage d'images

`convert -append input_file1 input_file2 ... output_file`

- Outil de conversion d'images

`convert input_file1 output_file`

TP1 : outil de redimensionnement d'images

```
convert input_file -resize percentage output_file
```

- 2 paramètres d'entrée
 - data
 - integer (0-100)
- 1 paramètre de sortie
 - data

Note des formateurs :

<command>

mon_programme \${ma_variable} -monoption \${ma_variable2}

</command>

TP2 : outil de rotation d'images

```
convert input_file -rotate angle output_file
```

- 2 paramètres d'entrée
 - data
 - integer (0-360)
- 1 paramètre de sortie
 - data
- utilisation d'un wrapper

Note des formateurs :

```
<command interpreter="perl">  
    mon_wrapper.pl ${ma_variable} ${ma_variable2}  
</command>
```

```
<param name="ma_variable" type="data" format="fasta">
```

Le multi-input files

1

- `<param name="gene_files_list" type="data" multiple="true" format="txt" label="Gene list file" />`
 - Passage directement de `$gene_files_list` dans la ligne de commande
 - Récupération d'une chaîne de caractère du type :

`/opt/galaxy-dist/database/files/000/dataset_001.dat,/opt/galaxy-dist/database/files/000/dataset_002.dat`

2

- `<repeat name="ma_liste" title="liste">`
 `<param name="input_file" type="data" format="fasta">`
 `</repeat>`

- `<command>`
 `ma_commande`
 `#for $gene in $ma_liste`
 `${gene.input_file}`
 `#end for`
 `</command>`

- Création d'une commande de type :

`ma_commande /opt/galaxy-dist/database/files/000/dataset_001.dat /opt/galaxy-dist/database/files/000/dataset_001.dat`

Le multi-input files

3

- `<param name="gene_files_list" type="data" multiple="true" format="txt" label="Gene list file" />`
- `<configfiles>`
 - `<configfile name="gene_files_list_config" >`
 - `#for $file in $gene_files_list:`
 - `${file}::${file.display_name}`
 - `#end for`
 - `</configfile>`
- `</configfiles>`
- Passage de `$gene_files_list_config` dans la ligne de commande
- `$gene_files_list_config = chemin vers un fichier != chaîne de caractère`
 - Contenu du fichier :
 - `/opt/galaxy-dist/database/files/000/dataset_001.dat::nom1`
 - `/opt/galaxy-dist/database/files/000/dataset_002.dat::nom2`

TP3 : outil de collage d'images (version 1)

```
convert -append input_file1 input_file2 input_file3 ... output_file
```

- x paramètres d'entrée
 - data
 - nombre inconnu
- 1 paramètre de sortie
 - data
- Utilisation de la balise **repeat**

Repeat

<command>

macommande

#for \$mon_element in \$ma_liste

-i \${mon_element.input}

#end for

</command>

<repeat name="ma_liste" title="liste">

 <param name="input" type="data" format="fasta">

</repeat>

TP4 : outil de collage d'images (version 2)

```
convert -append input_file1 input_file2 input_file3 ... output_file
```

- x paramètres d'entrée
 - data
 - nombre inconnu
- 1 paramètre de sortie
 - data
- Utilisation de l'attribut
 - multi="true"

multiple = "true"

```
<param name="gene_files_list" type="data" multiple="true" format="txt" />
```

```
$gene_files_list -> path_file_1,path_file2,path_file_3
```

Utilisation du multi-output files avec nombre de fichiers inconnus

1

- `<tool id="monid" name="myname" force_history_refresh="true" >`
- `<outputs>`
 - `<data format="txt" name="output" label=" log.txt" />`
- `</outputs>`
- `--output $output`
- `--output_id $output.id`
- `--new_file_path $__new_file_path__`

- `$__new_file_path__` correspond à un répertoire temporaire définit dans `galaxy.ini`
- Dans le wrapper, les fichiers de sorties doivent être envoyé dans ce répertoire
 - `new_file_path/primary_output.id_name_visible_txt`

Utilisation du multi-output files avec nombre de fichiers inconnus

2

- `<tool id="monid" name="myname" force_history_refresh="true" >`
- `<outputs>`
 - `<data format="txt" name="output" label=" log.txt" >`
 - `<discover_datasets pattern="__designation_and_ext__" directory="my_dir" visible="true" />`
 - `</data>`
- `</outputs>`

- Récupération de tous les fichiers présents dans le dossier "my_dir" (format de fichier définit selon l'extension):
 - `discover_datasets pattern="__designation_and_ext__"`
- Récupération de tous les fichiers présents dans le dossier "my_dir" et forcer le format :
 - `discover_datasets pattern="__designation__" ext="tabular"`
- Récupération de certains fichiers grâce à des expressions régulières (beurk mais pouvant être pratique):
 - `discover_datasets pattern="(P<designation>.+)\.tsv" ext="tabular"`

TP5 : outil de conversion d'une image en pdf, tiff, jpg

```
convert -append input_file1 input_file2 input_file3 ... output_file
```

- 1 paramètres d'entrée
 - data
- **x paramètres de sortie**
 - data

```
<outputs>
```

```
  <data format="txt" name="output" label=" log.txt" >
```

```
    <discover_datasets pattern="__designation_and_ext__" directory="my_dir" visible="true" />
```

```
  </data>
```

```
</outputs>
```

TP6 : intégration d'un outil « complexe »

Outil complexe?

- Outil complexe :
 - Multi-input
 - Multi-output
 - Conditions faisant varier le formulaire
- Intégration d'un outil « ultimate_tool » :
 - Redimensionner une image png
 - Tourner une images png
 - Coller plusieurs images png ensemble
 - Convertir une images png : jpg, pdf au choix.
- Nouveau :
 - Utilisation de la balise <conditionnal> pour les différentes “options”
 - Utilisation du type « boolean » pour les formats.

La balise conditionnal

```
command interpreter="python">
  #if $source.source_select=="database"
    blat_wrapper.py 0 $source.dbkey $input_query $output1 $iden $tile_size $one_off
  #else
    blat_wrapper.py 1 $source.input_target $input_query $output1 $iden $tile_size $one_off
  #end if
</command>
```

```
<conditional name="source">
  <param name="source_select" type="select" label="Target source">
    <option value="database">Genome Build</option>
    <option value="input_ref">Your Upload File</option>
  </param>
  <when value="database">
    <param name="dbkey" type="genomebuild" label="Genome" />
  </when>
  <when value="input_ref">
    <param name="input_target" type="data" format="fasta" label="Reference sequence" />
  </when>
</conditional>
```


La commande “convert”

- Utilisation de la commande “convert”
- Nécessite le package imagemagick
- Utile pour le traitement d'image

- Dans cette partie :

- Outil de redimensionnement d'images

`convert input_file -resize percentage output_file`

- Outil de rotation d'images

`convert input_file -rotate angle output_file`

- Outil de collage d'images

`convert -append input_file1 input_file2 ... output_file`

- Outil de conversion d'images

`convert input_file output_file`

Présentation du toolshed


Le toolshed Galaxy

- Appstore d'outils pour Galaxy
 - centralisation des outils
 - intégration rapide dans son instance
- Contient des répertoires contenant :
 - des outils / wrappers
 - des datatypes
 - des workflows
 - des data managers
- Plusieurs types de repositories
 - Unrestricted
 - Tool dependencies
 - Repositories dependencies

Le toolshed Galaxy

- Deux toolsheds principaux
 - toolshed main -> <https://toolshed.g2.bx.psu.edu>
 - toolshed test -> <https://testtoolshed.g2.bx.psu.edu/>
- Possibilité pour chacun de déployer son toolshed
 - Administration
 - Développement
 - Sécurité

Exemple de catégorie “Systems Biology”

 Galaxy Tool Shed

2570 valid tools on Aug 06, 2014

Search

- [Search for valid tools](#)
- [Search for workflows](#)

Valid Galaxy Utilities

- [Tools](#)
- [Custom datatypes](#)
- [Repository dependency definitions](#)
- [Tool dependency definitions](#)

All Repositories

- [Browse by category](#)

Available Actions

- [Login to create a repository](#)


Repositories Help User

Repositories in Category Systems Biology

Q

Name	Synopsis	Type	Metadata Revisions	Tools or Package Verified	Owner
cloudmap_in_silico_complementation	Perform in silico complementation analysis on multiple tabular snpEff output files	Unrestricted	1 (2012-11-01)	no	gregory-minevich
cluster3	A Wrapper for the Cluster3.0 program	Unrestricted	1 (2012-11-07) ▼	no	kellrott
ctcf_analysis	A tool for identification of CTCF sites	Unrestricted	6 (2013-04-25)	no	mkhan1980
dgidb_annotator	Annotates a tabular file with information from the Drug-Gene Interaction Database (http://dgidb.genome.wustl.edu/)	Unrestricted	3 (2014-03-07) ▼	no	devteam
matrix_normalization	normalization tool for matrix formatted data	Unrestricted	3 (2012-12-17) ▼	no	vnewton
primo_multiomics	Multi-omics module of Plant Research International's Mass Spectrometry (PRIMS) toolsuite	Unrestricted	6 (2014-08-01) ▼	no	pieterlukasse

Exemple d'outil “matrix_normalization”

 Galaxy Tool Shed

Repositories Help User

2570 valid tools on Aug 06, 2014

Repository Actions

Repository revision

3 (2012-12-17) repository tip

Select a revision to inspect and download versions of Galaxy utilities from this repository.

Repository 'matrix_normalization'

Sharable link to this repository:

https://toolshed.g2.bx.psu.edu/view/newton/matrix_normalization

Clone this repository:

hg clone https://toolshed.g2.bx.psu.edu/repos/newton/matrix_normalization

Name:

[matrix_normalization](#)

Type:

unrestricted

Synopsis:

normalization tool for matrix formatted data

Detailed description:

normalization tool for matrix formatted data

Revision:

[3:6e77048d4d88](#)

Owner:

ynewton

This revision can be installed:

True

Times cloned / installed:

102

Contents of this repository

Valid tools - click the name to preview the tool and use the pop-up menu to inspect all metadata

Name	Description	Version
Matrix Normalize	Matrix Normalize	3.0.0

Search

Search for valid tools

Search for workflows

Valid Galaxy Utilities

Tools

Custom datatypes

Repository dependency definitions

Tool dependency definitions

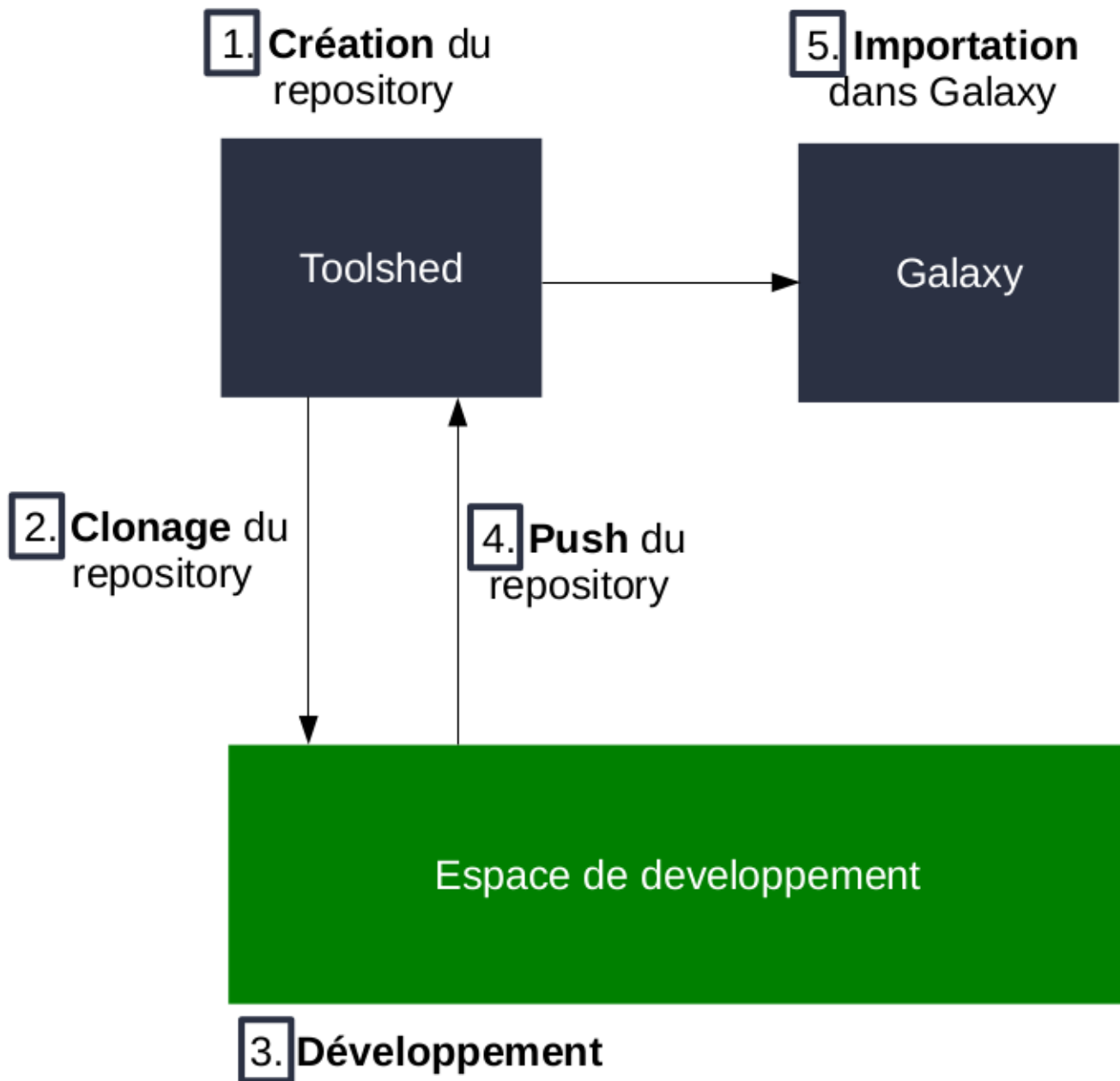
All Repositories

Browse by category

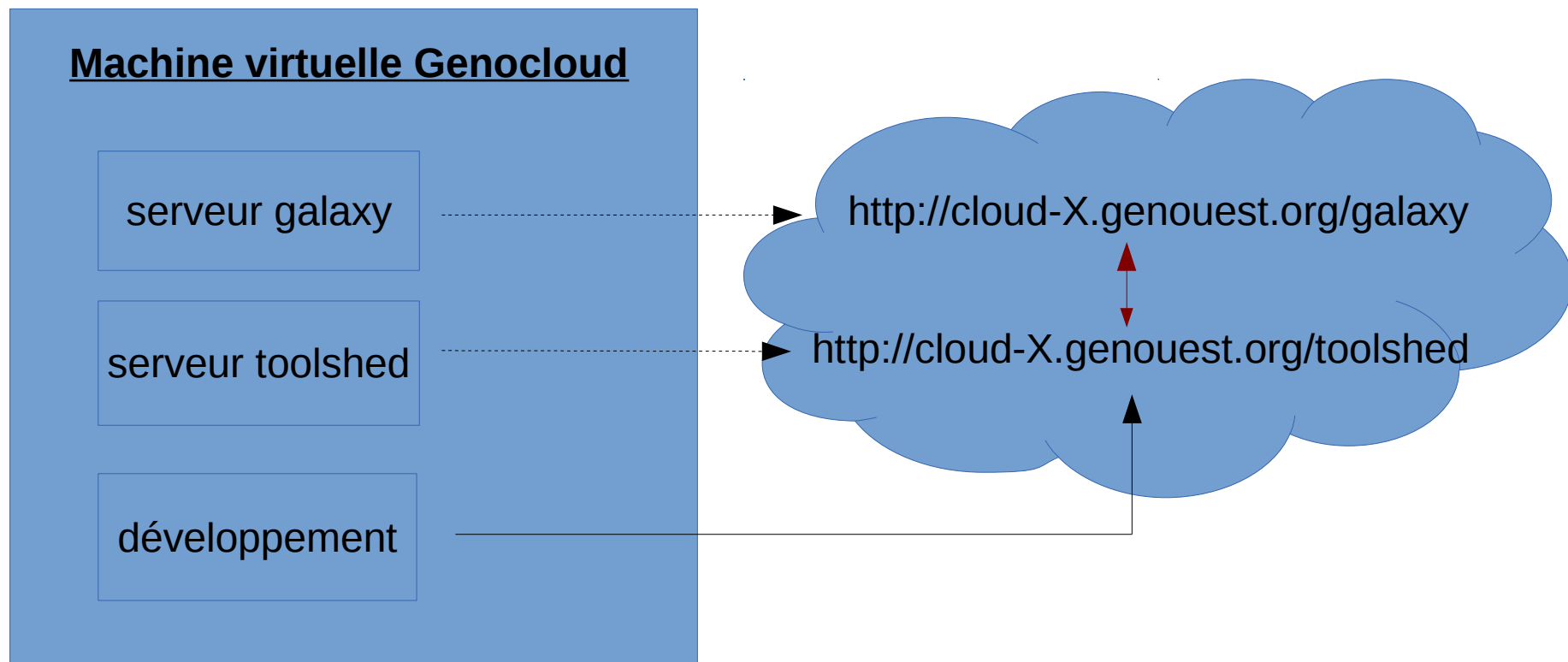
Available Actions

Login to create a repository

TP7. Déploiement d'un toolshed



Environnement



Installation de l'environnement

cf. document annexe

TP8. Intégration d'un outil dans un toolshed

Lors d'une intégration

- A chaque « hg push » sur le dépôt = une révision
 - A chaque installation d'une révision = un clone compté en plus
- Évitez lors du développement de multiplier les « hg push » :
 - Passez par le fichier tool_conf.xml
 - Premier « hg push » seulement lorsque l'outil est fonctionnel
- Pour installer votre outil du toolshed:
 - Admin > Search and browse tool shed repositories
 - Sélectionnez votre tool shed
 - Naviguez et installez

Lors d'une modification

- Pour la mise à jour de votre outil issus du toolshed
 - Admin > Managed installed tool shed repositories
 - Sur la flèche : Update tool shed status
 - Sur la flèche : Install latest revision / Get updates
- Pour « recharger » votre outil présent dans tool_conf.xml
 - Admin > Reload a tool's configuration
 - Sélectionnez votre outil
 - Reload

Intégration de l'outil “resize” de ce matin

- Quatre commandes sur le terminal:
 - hg clone
 - cp
 - hg add
 - hg commit
 - hg push
- Suppression dans le tool_conf.xml
- Installation via le toolshed


TP9. Intégration d'un outil avec sa dépendance

<https://wiki.galaxyproject.org/ToolDependenciesTagSets>

Les dependencies : étape 1/3

- Intégration d'un package spécial dans le toolshed
 - type = tool_dependencies
- Le repository nommé « comet_dependencies » contient uniquement **un fichier tool_dependencies.xml** définissant un package
- Dans ce fichier s'applique une succession d' « actions » :

```
<?xml version="1.0"?>
<tool_dependency>
  <package name="galaxy_commet" version="24.7.14">
    <install version="1.0">
      <actions>
        <action type="download_by_url">http://github.com/pierrepeterlongo/commet/archive/master.zip</action>
        <action type="shell_command">make</action>
        <!-- move directories into $INSTALL_DIR -->
        <action type="move_directory_files">
          <source_directory>bin</source_directory>
          <destination_directory>$INSTALL_DIR/bin</destination_directory>
        </action>
        <!-- move files into $INSTALL_DIR -->
        <action type="move_file">
          <source>dendro.R</source>
          <destination>$INSTALL_DIR/Rscript</destination>
        </action>
        <!-- create env.sh -->
        <action type="set_environment">
          <environment_variable name="PATH" action="prepend_to">$INSTALL_DIR/bin</environment_variable>
          <environment_variable name="RSCRIPTS" action="set_to">$INSTALL_DIR/Rscript</environment_variable>
        </action>
      </actions>
    </install>
  </package>
</tool_dependency>
```



Création d'un fichier env.sh embarqué par l'outil lors de l'exécution

Les dependencies : étape 2/3

- Au sein du repository contenant l'outil Galaxy développé doit s'ajouter un fichier **tool_dependencies.xml**

```
<?xml version="1.0"?>
<tool_dependency>
  <package name="galaxy_commet" version="24.7.14">
    <repository toolshed="http://toolshed.genouest.org" name="commet_dependencies"
owner="cmonjeau" prior_installation_required="True" changeset_revision="96f67cab9b21"/>
  </package>
</tool_dependency>
```

- Installation du package lors de l'installation de l'outil

Les dependencies : étape 3/3

- Au sein du descripteur de l'outil :

```
<requirements>  
  <requirement type="package" version="24.7.14">galaxy_commet</requirement>  
</requirements>
```

- Nécessaire pour l'embarquement automatique de l'environnement

Intégration de la dépendance de l'outil “resize”

- Désinstaller imagemagick
 - `apt-get remove imagemagick`
- Essayer l'outil : échec assuré
- Création de la dépendance imagemagick
 - Création du **package imagemagick** (tool_dependencies)
 - Modification de l'outil pour :
 - installer le package lors de son installation
 - Prendre un environnement précis lors de l'exécution

Le package imagemagick

- Création d'un nouveau repository
 - De type tool_dependencies
 - Contiendra les règles pour l'installation
- Téléchargement des sources :
 - <http://www.imagemagick.org/download/ImageMagick.tar.gz>
- Exécution de configure :
 - ./configure
- Compilation :
 - make install
- Ajout des bibliothèques dans le système
 - ldconfig /usr/local/lib
- Ajout de /usr/local/bin dans le \$PATH -> creation d'un env.sh

Actions

Présentation du système de gestion des banques

Powered by Anthony Brétaudeau

TP10 : Intégration d'un outil utilisant des banques

Powered by Anthony Brétaudeau (again)