jlm Documentation

Release 0.2.0-DEV

Takafumi Arakaki

Feb 10, 2020

CONTENTS

CHAPTER

ONE

INSTALLATION

1.1 Using Julia's package manager

```
(v1.1) pkg> add JuliaManager
...

julia> using JuliaManager

julia> JuliaManager.install_cli()
...
```

You need to add *~/.julia/bin* to *$PATH* as would be messaged if it not.

Note: This doesn't work in Windows.

1.2 Using pip (experimental)

```
$ pip install --upgrade 'https://github.com/tkf/JuliaManager.jl/archive/master.zip
↪#egg=jlm&subdirectory=jlm'
```

Note: This may work in any platform.

CHAPTER

TWO

EXAMPLES

2.1 Standard usage

```
$ cd PATH/TO/YOUR/PROJECT

$ jlm init
...

$ jlm run
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _    |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |   |
  | | |_| | | | (_| |   |  Version 1.1.0 (2019-01-21)
 _/ |\__'_|_|_|\__'_|   |  Official https://julialang.org/ release
|__/                    |

julia>
```

2.2 Using MKL.jl-patched Julia and standard Julia side-by-side

MKL.jl is a convenient way of using Intel's Math Kernel Library (MKL) with Julia. However, it still has some caveats and difficult to use standard Julia installation since precompilation cache are shared. This problem can be avoided by using `jlm` to separate compilation cache paths for MKL.jl and non-MKL.jl Julia. This way, both Julia installations can be used simultaneously without invoking repeated precompilation.

As MKL.jl overwrites its Julia installation, you need to create a dedicated Julia installation. Suppose it's done by

```
$ mkdir -p ~/opt/julia-mkl
$ cd ~/opt/julia-mkl
$ cd tar xf ~/Downloads/julia-1.1.0-linux-x86_64.tar.gz
```

Then create a project and install MKL.jl in it. Note that it is better be done in a separate project to avoid installing MKL.jl where standard (non-MKL.jl) Julia may accidentally instantiate and build it. This isolation is done by `--project=.`:

```
$ cd PATH/TO/PROJECT
$ ~/opt/julia-mkl/julia-1.1.0/bin/julia \
    --startup-file=no --compiled-modules=no --project=.
...
```

(continues on next page)

(continued from previous page)

```
(PROJECT) pkg> add https://github.com/JuliaComputing/MKL.jl
```

You may also need to run `pkg> build MKL`.

Finally, use *jlm* to isolate precompilation cache:

```
$ jlm init ~/opt/julia-mkl/julia-1.1.0/bin/julia
...

$ jlm run --project=.
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.1.0 (2019-01-21)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |
```

(This may cause (re)compilation of cache files if you import some packages in `~/.julia/config/startup.jl`.)

In Julia REPL, you can check if *jlm* is using the correct version of Julia by

```
julia> Base.julia_cmd().exec[1]
"/home/USER/opt/julia-mkl/julia-1.1.0/bin/julia"

julia> using LinearAlgebra

julia> BLAS.vendor()
:mkl
```

CHAPTER

THREE

MANUAL

Command line interface to manage Julia's system images.

```
usage: jlm [-h] [--version] [--dry-run] [--verbose] [--pdb] [--jlm-dir PATH]
           {run,init,set-default,set-sysimage,unset-sysimage,create-default-sysimage,
↪install-backend,update-backend,info,locate,ijulia-kernel,install-ijulia-kernel}
           ...
```

3.1 Named Arguments

--version show program's version number and exit

--dry-run Default: False

--verbose, -v Default: False

--pdb Default: False

--jlm-dir Specify the `.jlm` directory which is created by `jlm init` and stores information for `jlm`. By default, `.jlm` directory found in the nearest "ancestor" directory is used. Run `jlm locate dir` to locate the actual directory that would be used.

3.2 Sub-commands:

3.2.1 run

Run `julia` executable with appropriate system image.

```
jlm run [-h] [julia] [arguments [arguments ...]]
```

Positional Arguments

julia The name of Julia executable on `$PATH` or a path to the Julia executable.

arguments Arguments and options passed to `julia`. Non-option like argument (i.e., the ones *not* starting with `-`) following `run` is always interpreted as a Julia executable. To pass a file path to Julia, use `--` as the first argument to `run`; i.e. `jlm ... run -- PATH/TO/FILE.jl ...`. If you pass `julia` to `run`, there is no need to pass `--` since the argument parsing for `jlm` automatically ends at this point.

If argument `julia` is not given, the default executable configured by `jlm init` is used.

3.2.2 init

Initialize `jlm`.

```
jlm init [-h] [--sysimage SYSIMAGE] [julia]
```

Positional Arguments

julia The name of Julia executable on `$PATH` or a path to the Julia executable.

Named Arguments

--sysimage, -J The path to system image.

`jlm init` does:

- Create a data store (`.jlm` directory).
- Install `JuliaManager.jl` if it is not installed for `<julia>`.
- Compile the "patched" default system image (see note below) for `<julia>` if not already found and `--sysimage|-J` is not given. This can be done separately by `jlm compile-default-sysimage`.
- Set the system image to be used for `<julia>`. This can be re-done later by `set-sysimage`.

Note: `jlm` compiles the system image with a patch that does Suggestion: Use different precompilation cache path for different system image by tkf · Pull Request #29914 · JuliaLang/julia

3.2.3 set-default

Set default Julia executable to be used.

```
jlm set-default [-h] julia
```

Positional Arguments

julia The name of Julia executable on `$PATH` or a path to the Julia executable.

3.2.4 set-sysimage

Set system image for `juila`.

```
jlm set-sysimage [-h] [julia] sysimage
```

Positional Arguments

julia The name of Julia executable on `$PATH` or a path to the Julia executable.

sysimage The path to system image.

3.2.5 unset-sysimage

Unset system image for `juila`.

```
jlm unset-sysimage [-h] [julia]
```

Positional Arguments

julia The name of Julia executable on `$PATH` or a path to the Julia executable.

3.2.6 create-default-sysimage

Compile default system image for `julia`.

```
jlm create-default-sysimage [-h] [--force] [julia]
```

Positional Arguments

julia The name of Julia executable on `$PATH` or a path to the Julia executable.

Named Arguments

--force, -f Re-compile default system image for `julia` even if it already exists.

Default: False

3.2.7 install-backend

Install JuliaManager.jl for this `julia`.

```
jlm install-backend [-h] [julia]
```

Positional Arguments

julia The name of Julia executable on `$PATH` or a path to the Julia executable.

3.2.8 update-backend

Update JuliaManager.jl for this `julia`.

```
jlm update-backend [-h] [julia]
```

Positional Arguments

julia The name of Julia executable on `$PATH` or a path to the Julia executable.

3.2.9 info

Print information about jlm setup.

```
jlm info [-h]
```

3.2.10 locate

Show paths to related files and directories

```
jlm locate [-h] {sysimage,base,dir,home-dir} ...
```

Sub-commands:

sysimage

Print system image that would be used for `julia`.

```
jlm locate sysimage [-h] [julia]
```

Positional Arguments

julia The name of Julia executable on `$PATH` or a path to the Julia executable.

base

Print directory for which `jlm init` was executed.

```
jlm locate base [-h]
```

dir

Print directory in which `jlm` information is stored.

```
jlm locate dir [-h]
```

home-dir

Print directory in which `jlm` global information is stored.

```
jlm locate home-dir [-h]
```

3.2.11 ijulia-kernel

An entrypoint to be called from Jupyter frontends.

```
jlm ijulia-kernel [-h] [--julia [JULIA]] [--julia-option JULIA_OPTION]
                  connection_file
```

Positional Arguments

connection_file

Named Arguments

--julia The name of Julia executable on `$PATH` or a path to the Julia executable.

--julia-option

This command is not meant to be directly used. It is an entrypoint to be invoked from `kernel.json`.

3.2.12 install-ijulia-kernel

Install a Jupyter kernel that launches IJulia via jlm.

```
jlm install-ijulia-kernel [-h] (--name NAME | --output-dir PATH)
                          [--display-name DISPLAY_NAME] [--dont-store-jlm-dir]
                          [--julia-option JULIA_OPTION] [--jupyter JUPYTER]
```

Named Arguments

--name Name of the kernel spec. Typically, kernel spec is going to be written into `~/.local/share/jupyter/kernels/NAME/kernel.json`.

--output-dir Directory in which `kernel.json` is written.

--display-name Name of the kernel to be shown in Jupyter Lab, Notebook, etc. Default to NAME.

--dont-store-jlm-dir Do not –jlm-dir if specified. This means that `.jlm` directory is searched during run-time.

Default: True

--julia-option Option to be passed to Julia at run-time. It can be specified multiple times.

--jupyter Jupyter command line program. This is used to locate the directory in which kernel should be stored. This is ignored if `--output-dir"` is given.

Default: “jupyter”