AFLOW V 3.1.111

```
*****************************************************************************************************
*                                                                                                   *
*                          aflow - STEFANO CURTAROLO Duke University 2003-2017                      *
*                              High-Throughput ab-initio Computing Project                          *
*                                                                                                   *
*****************************************************************************************************
LATEST VERSION OF THE FILE:                                  materials.duke.edu/AFLOW/aflow_scripting.pdf
*****************************************************************************************************

 AFLOW SCRIPTING MODE
  aflow --justbefore=string < something
        Prints all the strings (defined by EOL) present in the
        file (also stdout) "something" until the first occurence
        of "string".
  aflow --justafter=string < something
        Prints all the strings (defined by EOL) present in the
        file (also stdout) "something" after the first occurence
        of "string".
  aflow --justbetween=string_start[,string_stop] < something
        Prints all the strings (defined by EOL) present in the
        file (also stdout) "something" between the first occurence
        of "string_start" to the first occurence of "string_stop".
        If no string_stop is specified, the default is
     "START.string_start"
"STOP.string_start"

  aflow --qsub=N,file
        perform N times the command "qsub file" (useful for rapid aflow submissions)
  aflow --qdel=aaa,nnn:mmm,aaa,bbb,ccc
        qdels all the jobs specified between "," and the
        list of jobs between ":" in increments of 1.
        For instance  aflow --qdel 10:20,99,102
        qdels 10,11,12...20,99,102

  aflow --bsub=N,file
        perform N times the command "bsub < file" (useful for rapid aflow submissions)
  aflow --bkill=aaa,nnn:mmm,aaa,bbb,ccc
        bkills all the jobs specified between "," and the
        list of jobs between ":" in increments of 1.
        For instance  aflow --bkill 10:20,99,102
        bkills 10,11,12...20,99,102

  aflow --sbatch=N,file
        perform N times the command "sbatch file" (useful for rapid aflow submissions)
  aflow --scancel=aaa,nnn:mmm,aaa,bbb,ccc
        scancels all the jobs specified between "," and the
        list of jobs between ":" in increments of 1.
        For instance  aflow --scancel 10:20,99,102
        scancels 10,11,12...20,99,102

  aflow --kill=aaa,nnn:mmm,aaa,bbb,ccc
        kills all the jobs specified between "," and the
        list of jobs between ":" in increments of 1.
        For instance  aflow --kill 10:20,99,102
        kills -9 10,11,12...20,99,102

  aflow --multish    Very useful option for scripting.
         if you have a "file" containing a gazillion of instructions,
         one per line, such as
         command_perform ./directory1
         command_perform ./directory2
         ......
         command_perform ./directory..
         then the instruction
```

```
  aflow --multish --np=XX --FILE file
        will pipe the instruction in a push-pop list and feed XX cpus in a
        multithreaded environment. You can susbtitute "--np=XX" with "--npmax"
        (or omit it at all) and use the max number of available cores in the
        machine. Instead of --FILE you can use " --F | -F | --f | -f".
        If you omit the --FILE option, then the last argument will
        be taken as the name of the file.
        Note that due to the variable nature of the time requested for each line,
        you might loose causality in the whole process (most of the times
        you do not need it, though).
        These are examples of proper commands:
          aflow --multish --np=12 --FILE file   (run file in 12 cores)
          aflow --multish --np=12 file          (run file in 12 cores)
          aflow --multish file                  (run file in all your cores)

  aflow --multizip [--prefix=PREFIX] [--size=SSSS] [--add] --F file | -D directory1 directory2 ....
        If you have a huge amount of directories to zip then you can clusterize them, so that
        each subzip contains no more than SSSS entries.
        The "prefix=" is optional, the default is "m"
        The "size=" is optional, the default is 100.
        If you add  --add then, if zips are present, they will be added with the new files.
        This is a very useful option because many file systems do not allow big files.
        I usually use:
         aflow --multizip --prefix=magnetic --size=500 `find . -name EIGENVAL.bands.bz2`
        To make life easier, this option clears up the words
        "LOCK,aflow.in,OUTCAR.relax2.bz2,EIGENVAL.bands.bz2" from the directory names.

  aflow --multibzip2 --np=XX --FILE file file2 file3....
        if you have a huge amount of files to bzip2 then you can multithread the bzip2 so there are XX (np) co
        If --np=is not specified then the code will take 1 core.
        The filenames with ".bz2" extension are neglected.

  aflow --multibunzip2 --np=XX --FILE file.bz2 file2.bz2 file3.bz2....
        Same as --multibzip2 but for un-bzipping.
        The filenames without ".bz2" extension are neglected.

  aflow --multigzip --np=XX --FILE file file2 file3....
        if you have a huge amount of files to gzip then you can multithread the gzip so there are XX (np) core
        If --np=is not specified then the code will take 1 core.
        The filenames with ".gz" extension are neglected.

  aflow --multibunzip --np=XX --FILE file.gz file2.gz file3.gz....
        Same as --multigzip but for un-bzipping.
        The filenames without ".gz" extension are neglected.

  aflow --getTEMP [--runstat | --runbar | --refresh=X | --warning_beep=T | --warning_halt=T | --mem=XX ]
        If available, the command outputs the hostname and temperatures of the machine. Useful to find hardwar
        with --runstat the command continuously prints the temperature, refreshing every XX refresh seconds;
        with --runbar the command prints a bar with the temperature, refreshing every XX refresh seconds;
        with --refresh=X you can specify the refresh time (DEFAULT below)
        with --warning_beep=T, if the max temp goes beyond T(C, DEFAULT below), the command beeps the computer
        with --warning_halt=T, it the max temp goes beyond T(C, DEFAULT below), the command halts the computer
        with --max==X | --maxmem=XX, it kills vasp/mpivasp using more than XX% of memory.
        DEFAULT VALUES in aflow.h
        #define AFLOW_CORE_TEMPERATURE_BEEP_      56.0  // Celsius
        #define AFLOW_CORE_TEMPERATURE_HALT_      65.0  // Celsius, you need to run aflow as root to halt
        #define AFLOW_CORE_TEMPERATURE_REFRESH_   5.0   // seconds

  aflow --monitor [--mem=XX]
        This is a wrap up set of commands to be sent in the background so that the node is monitored for healt
        It kills vasp/mpivasp using more than XX% of memory.
        The default for XX is 95%/NCPUs, so even in the worst scenario there shoud be enough RAM to resuscitat
        a soon-to-be-frozen machine/node.

*******************************************************************************************
```

```
*                                                                                                  *
*                        aflow - STEFANO CURTAROLO Duke University 2003-2017                       *
*                            High-Throughput ab-initio Computing Project                           *
*                                                                                                  *
****************************************************************************************************
```