GRASS GIS in the Cloud

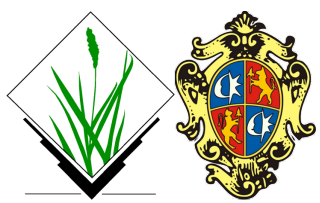
Luca Delucchi, Markus Neteler

Fondazione Edmund Mach – GIS & Remote Sensing Platform

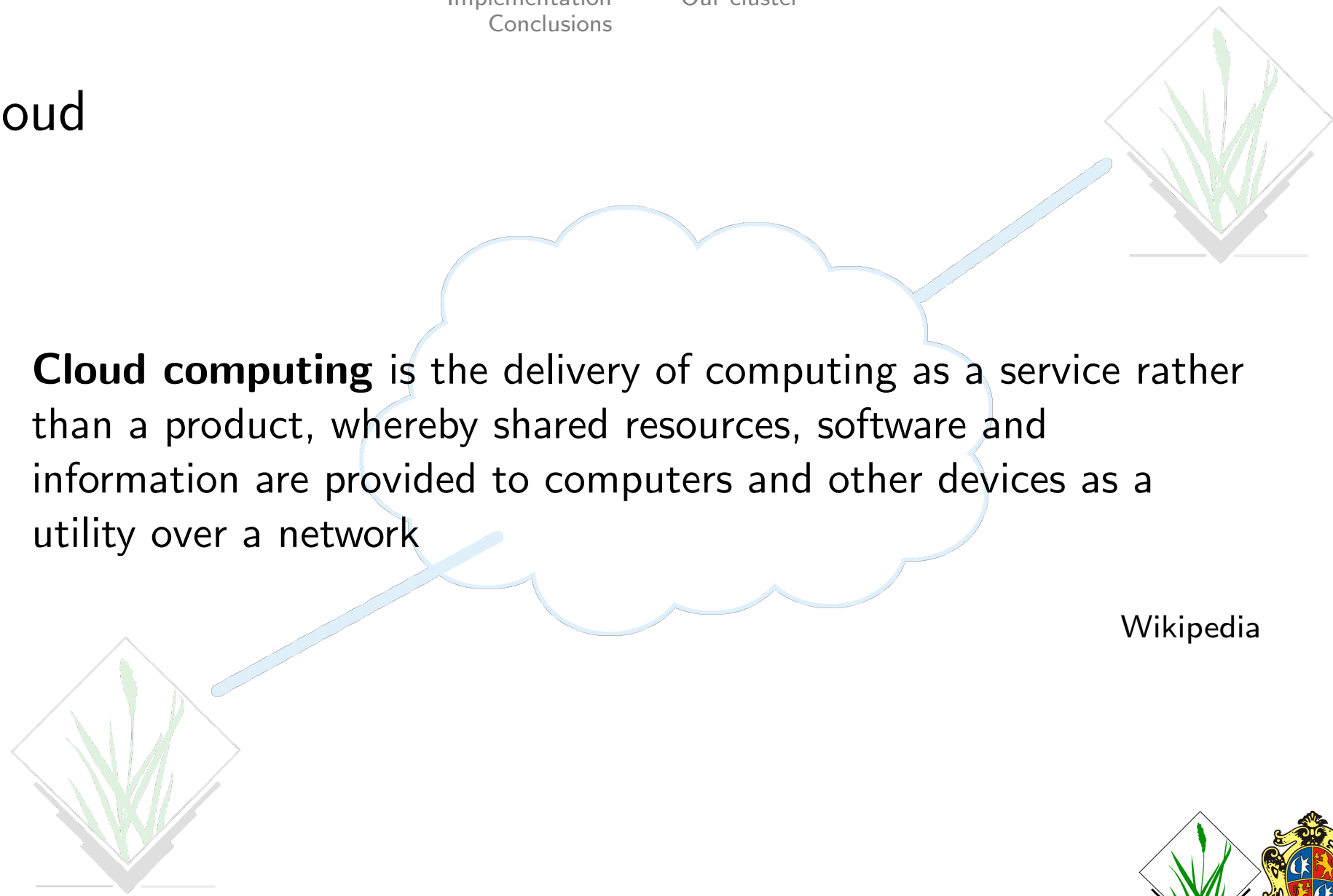
http://gis.cri.fmach.it

XIII Meeting GRASS e GFOSS

17th February 2012, Trieste (Italy)



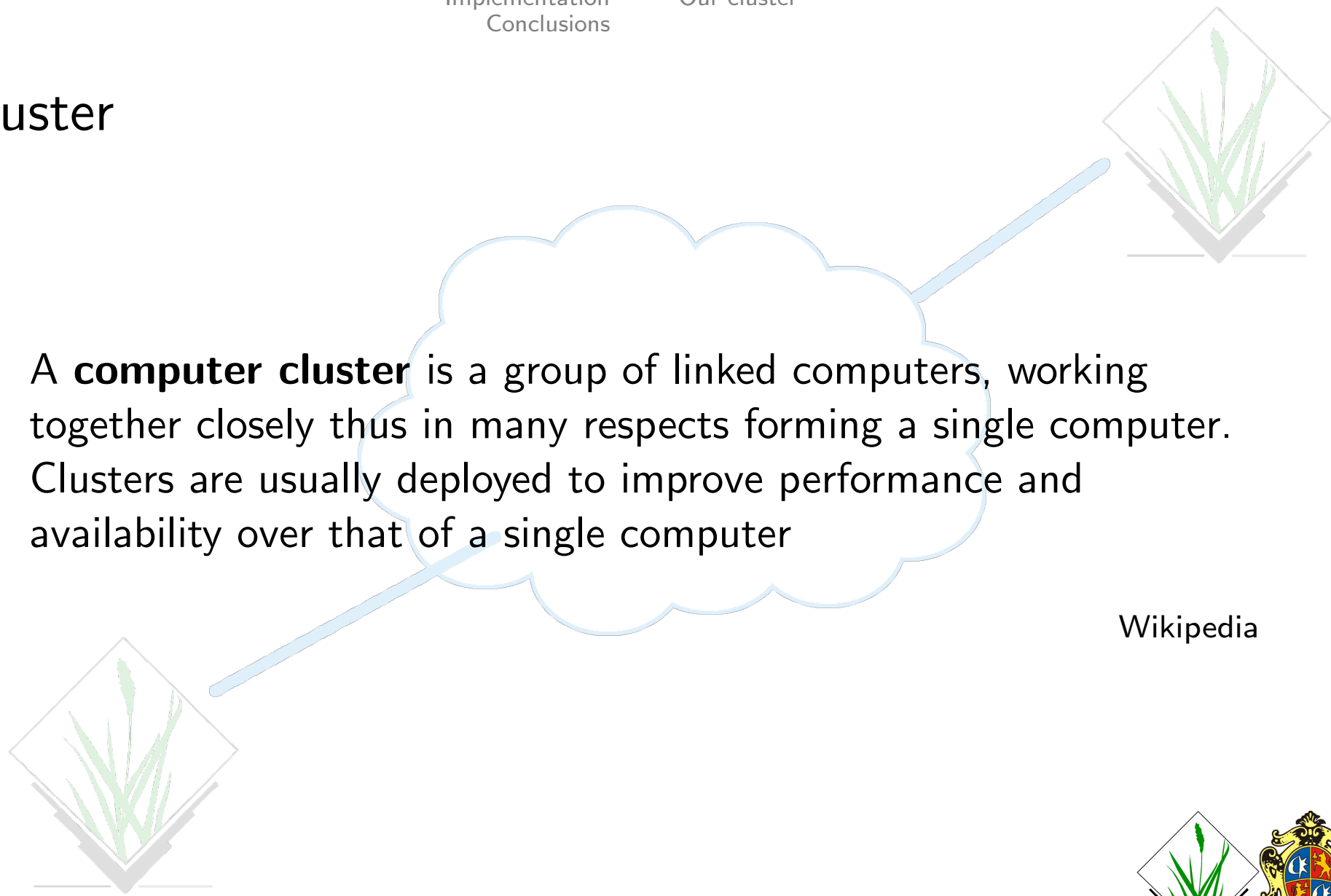
Cloud



Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software and information are provided to computers and other devices as a utility over a network

Wikipedia

Cluster



A **computer cluster** is a group of linked computers, working together closely thus in many respects forming a single computer. Clusters are usually deployed to improve performance and availability over that of a single computer

Wikipedia

Our cluster: infrastructure

FEM GIS cluster consists of

- 300 nodes
- 610 GB RAM
- 34 TB disk storage + 15 TB tape backup
- 10 Gb/s internal bus
- Scientific Linux 6.2
- **Grid Engine**

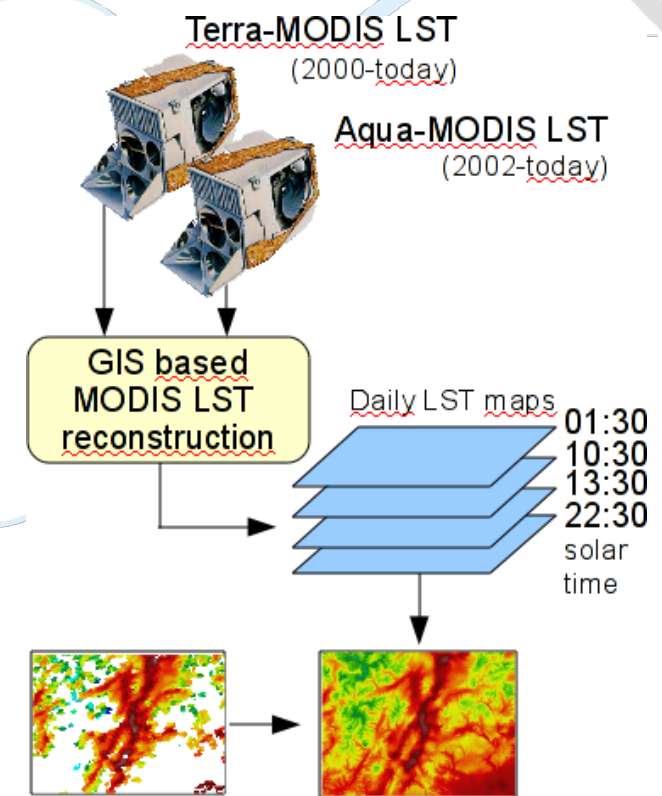
Our cluster: massive spatial data analysis

MODIS Land Surface Temperature (LST) map gap-filling (13,000 maps)

MODIS data analysis (LST, NDVI, Snow products)

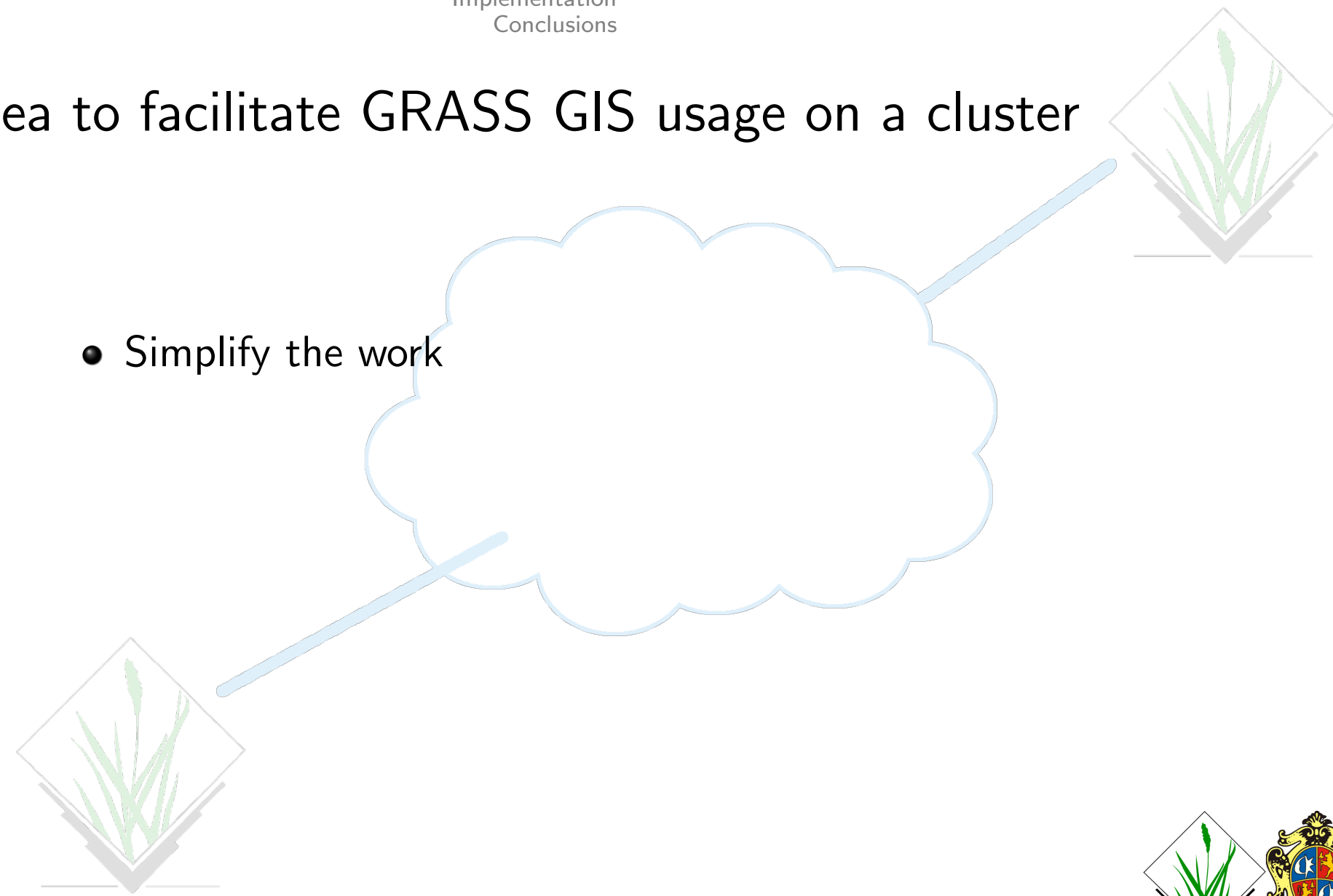
LiDAR data analysis

Solar energy calculations



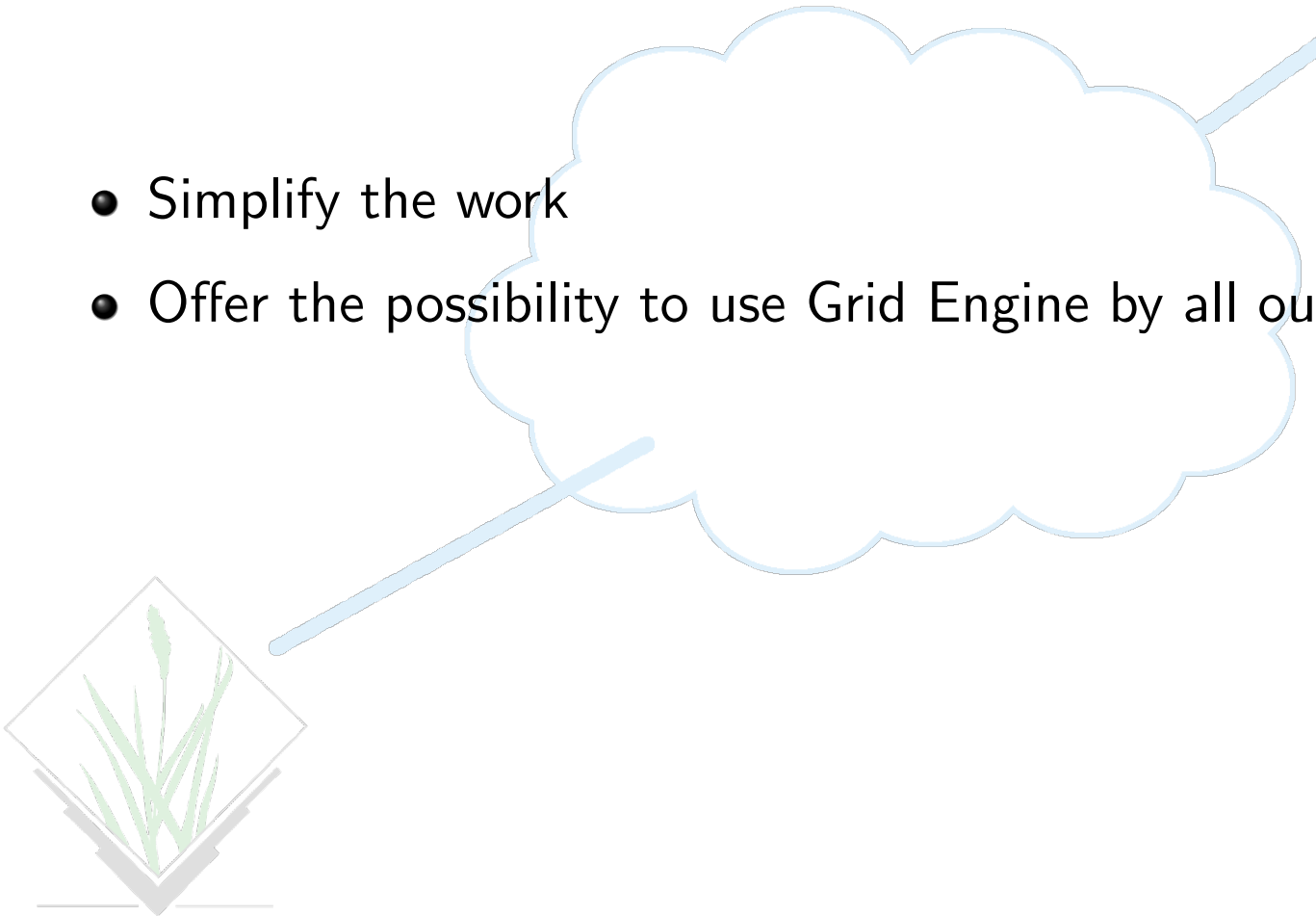
Neteler et al., 2011: Tiger mosquito, IJHG, doi:10.1186/1476-072X-10-49

Idea to facilitate GRASS GIS usage on a cluster



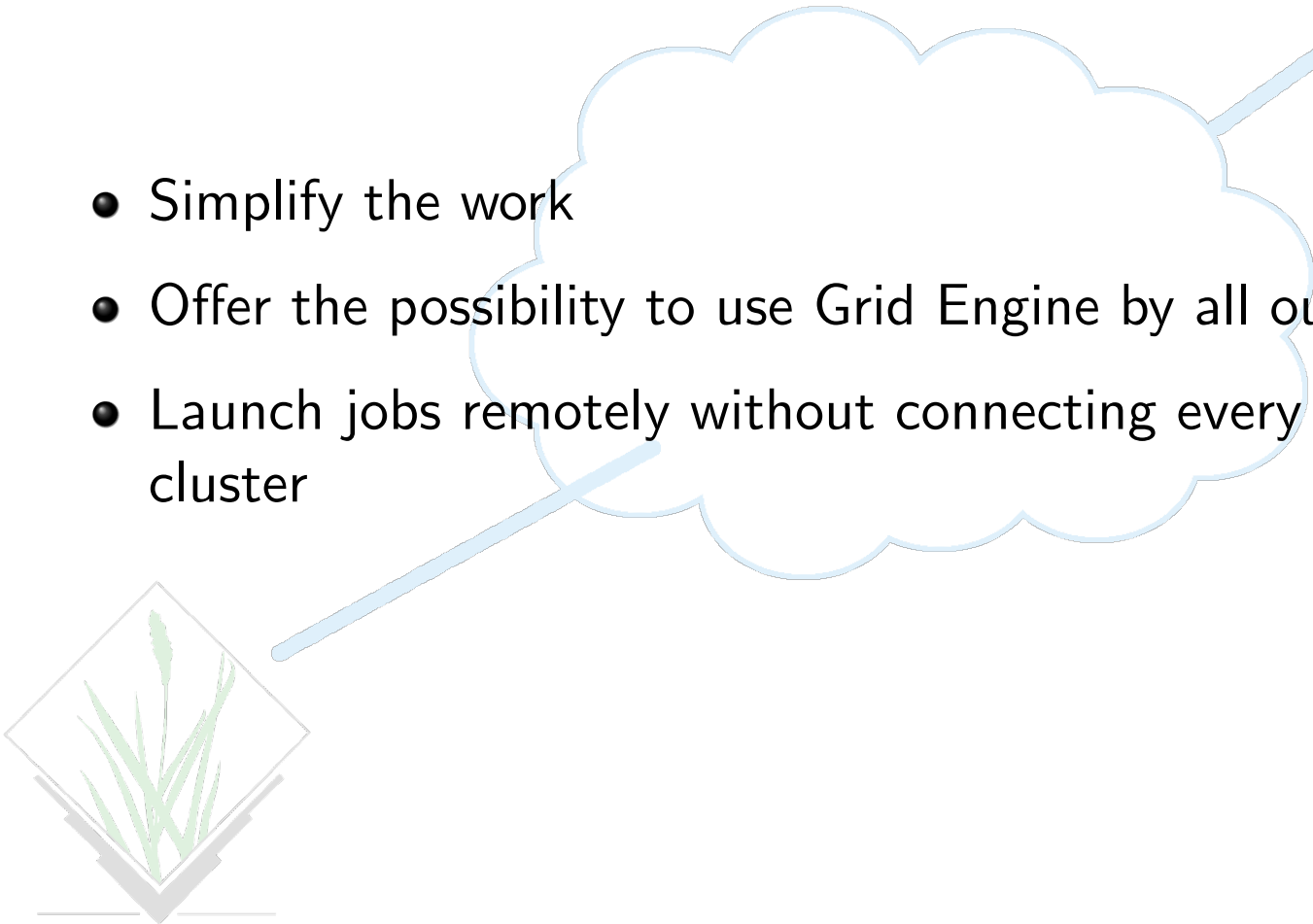
- Simplify the work

Idea to facilitate GRASS GIS usage on a cluster



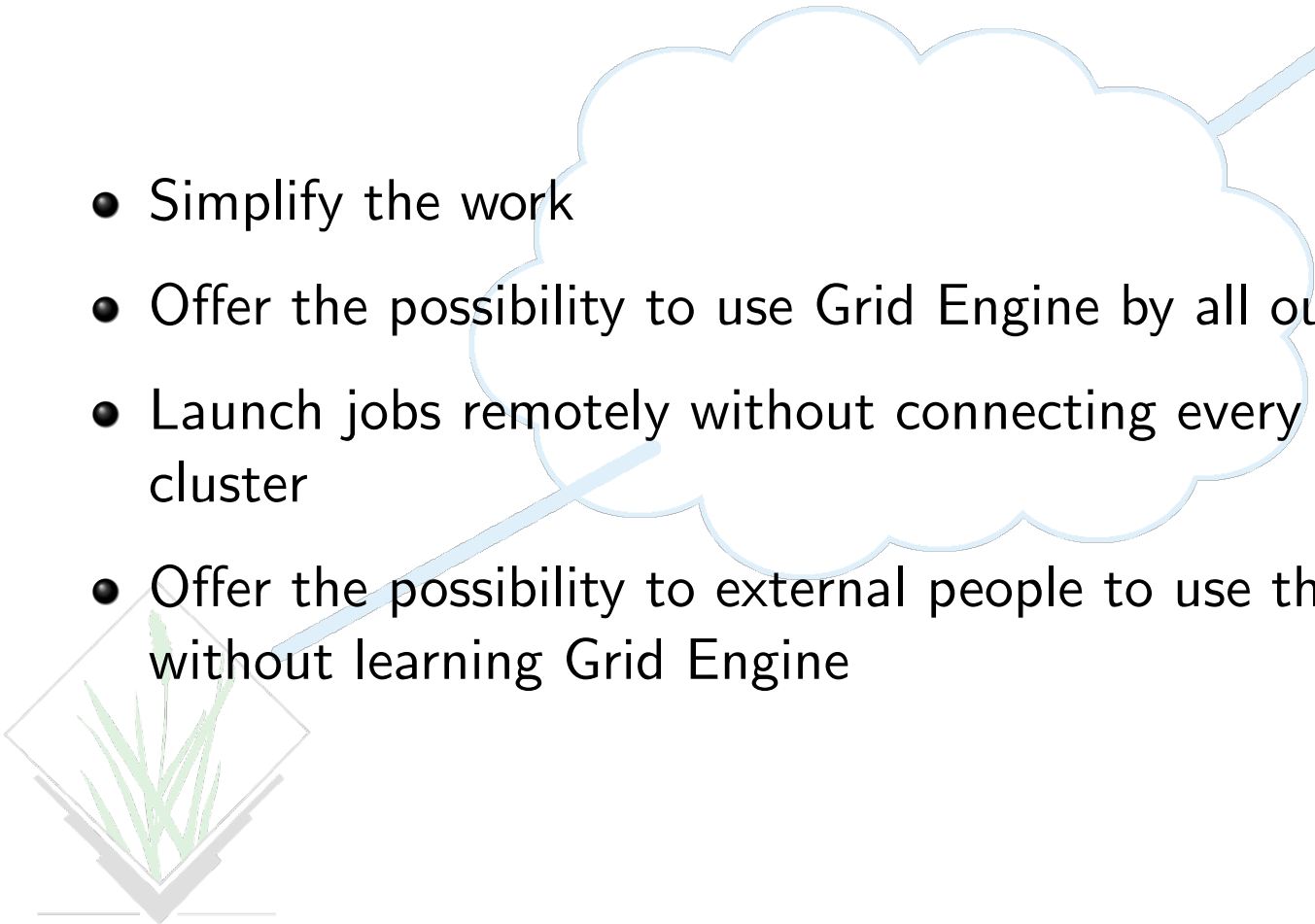
- Simplify the work
- Offer the possibility to use Grid Engine by all our colleagues

Idea to facilitate GRASS GIS usage on a cluster

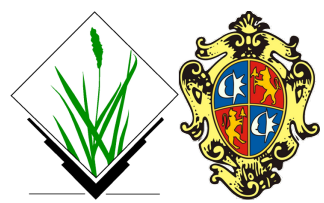


- Simplify the work
- Offer the possibility to use Grid Engine by all our colleagues
- Launch jobs remotely without connecting every time to the cluster

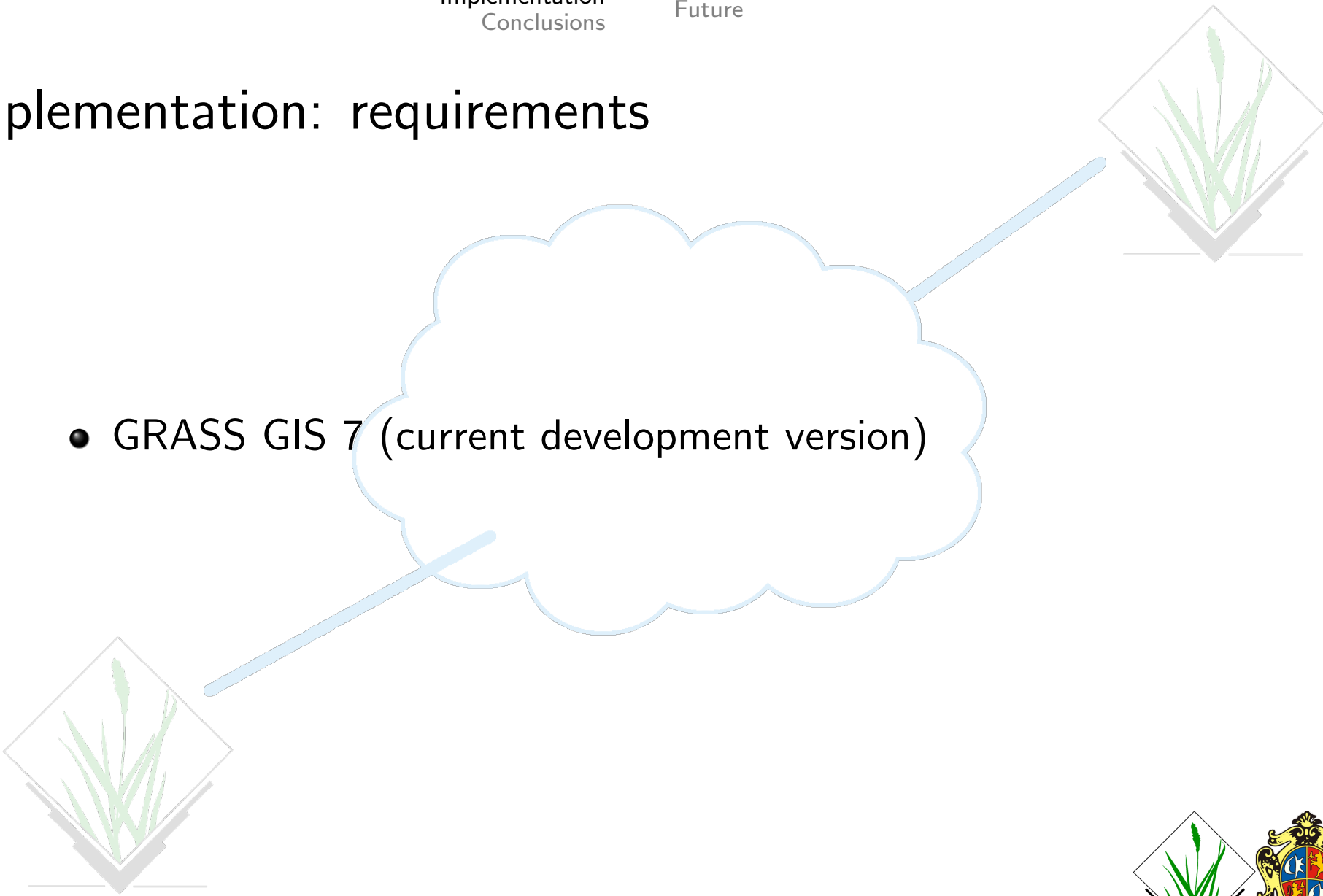
Idea to facilitate GRASS GIS usage on a cluster



- Simplify the work
- Offer the possibility to use Grid Engine by all our colleagues
- Launch jobs remotely without connecting every time to the cluster
- Offer the possibility to external people to use the cluster without learning Grid Engine

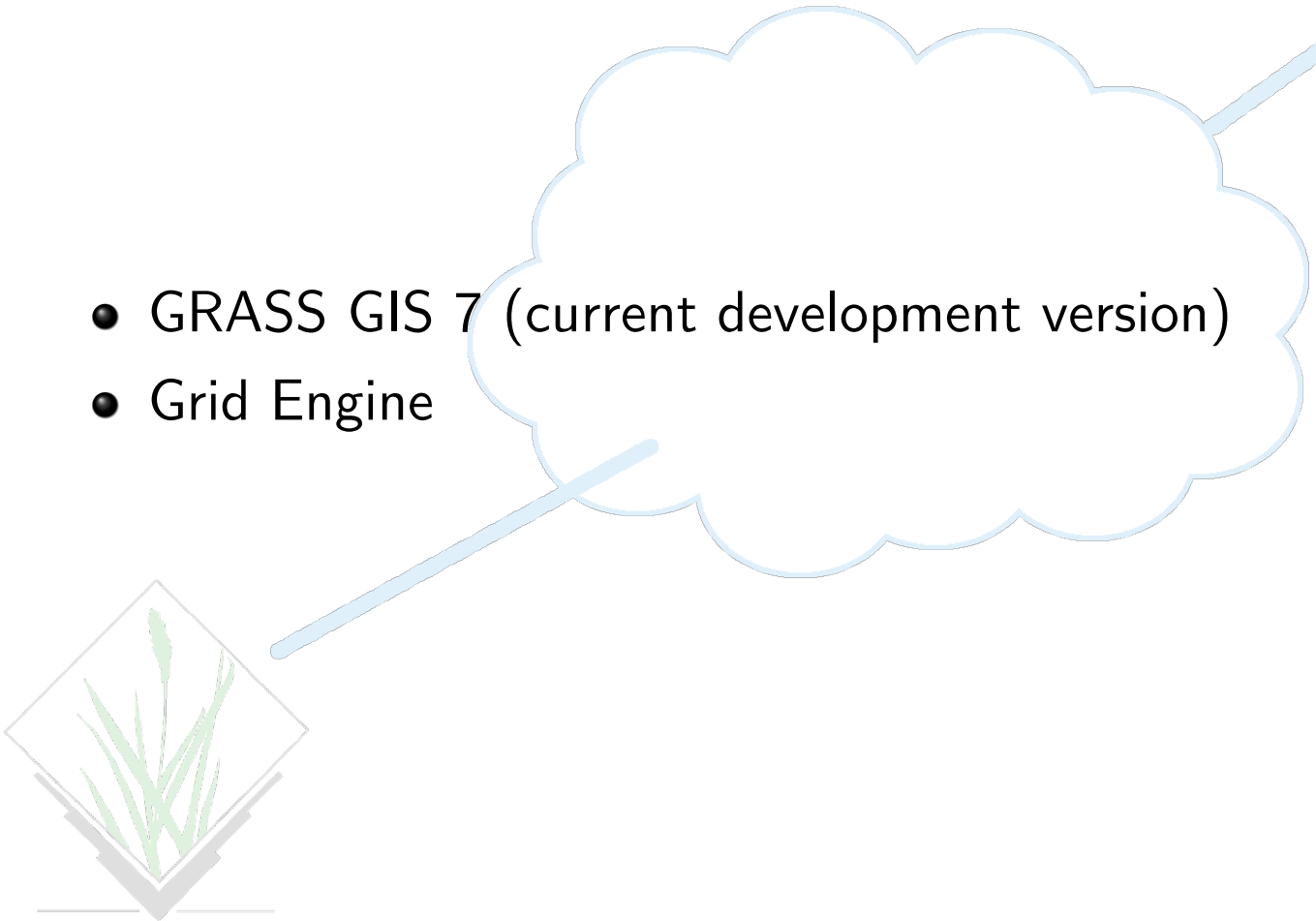


Implementation: requirements



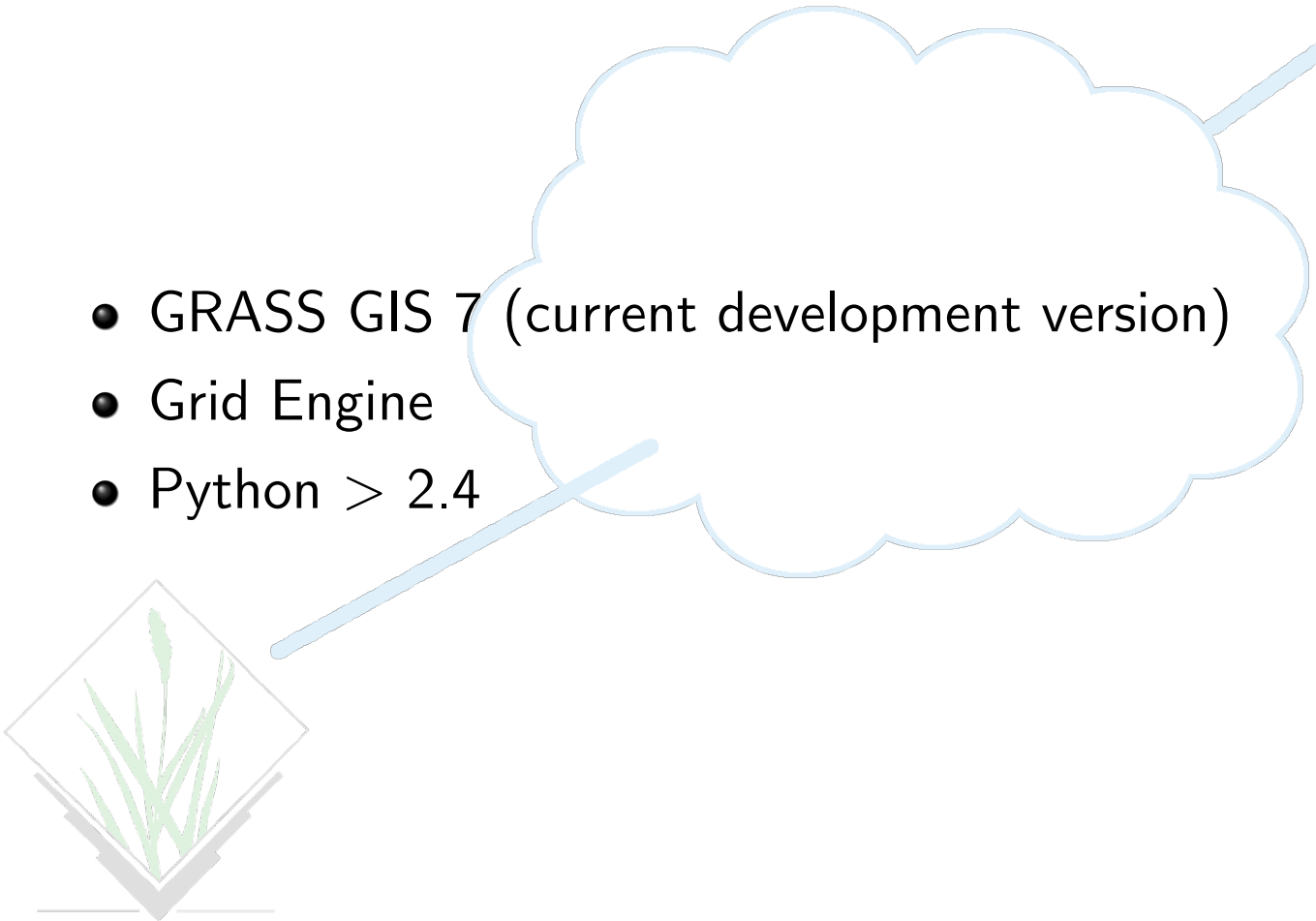
- GRASS GIS 7 (current development version)

Implementation: requirements



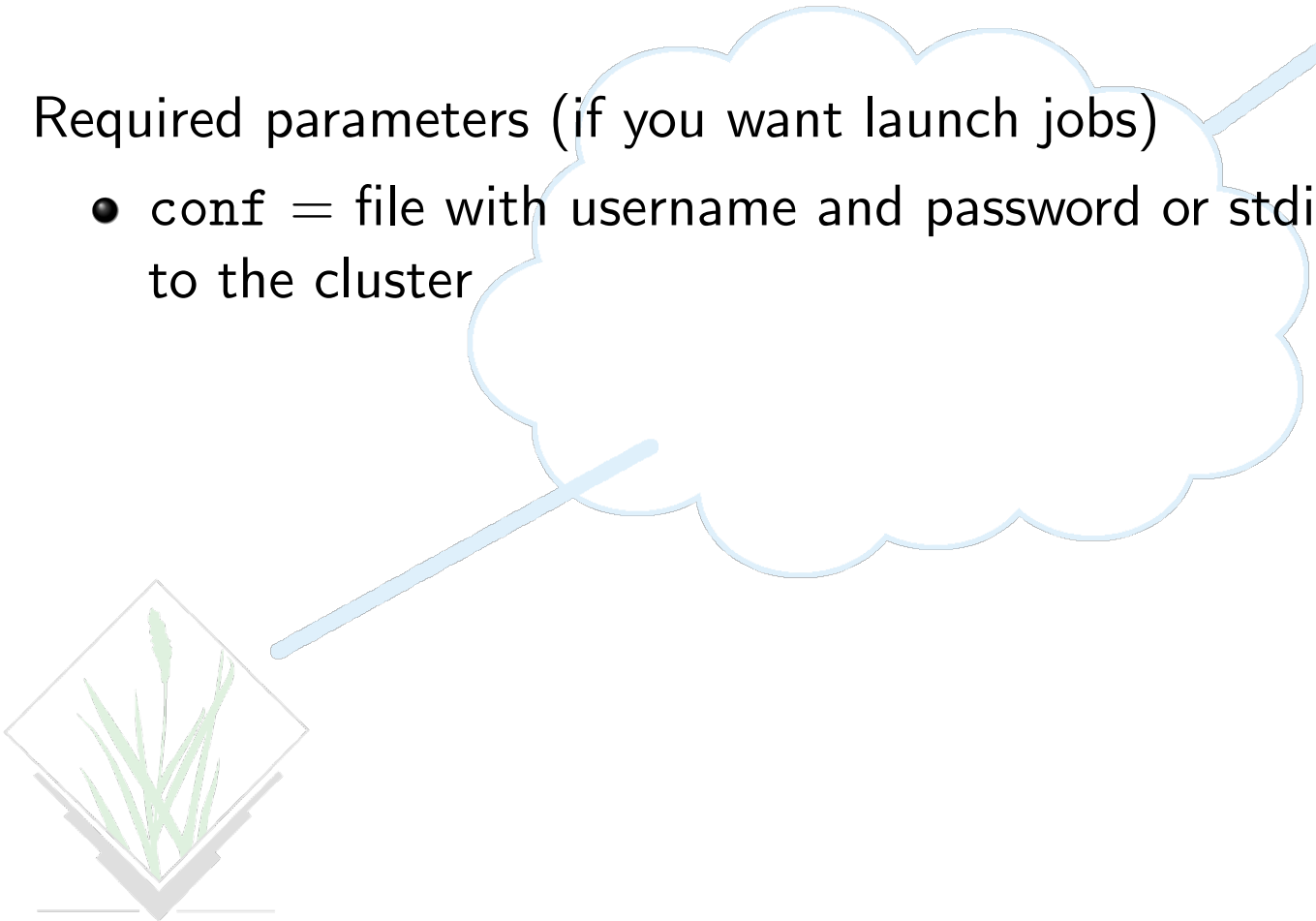
- GRASS GIS 7 (current development version)
- Grid Engine

Implementation: requirements



- GRASS GIS 7 (current development version)
- Grid Engine
- Python $>$ 2.4

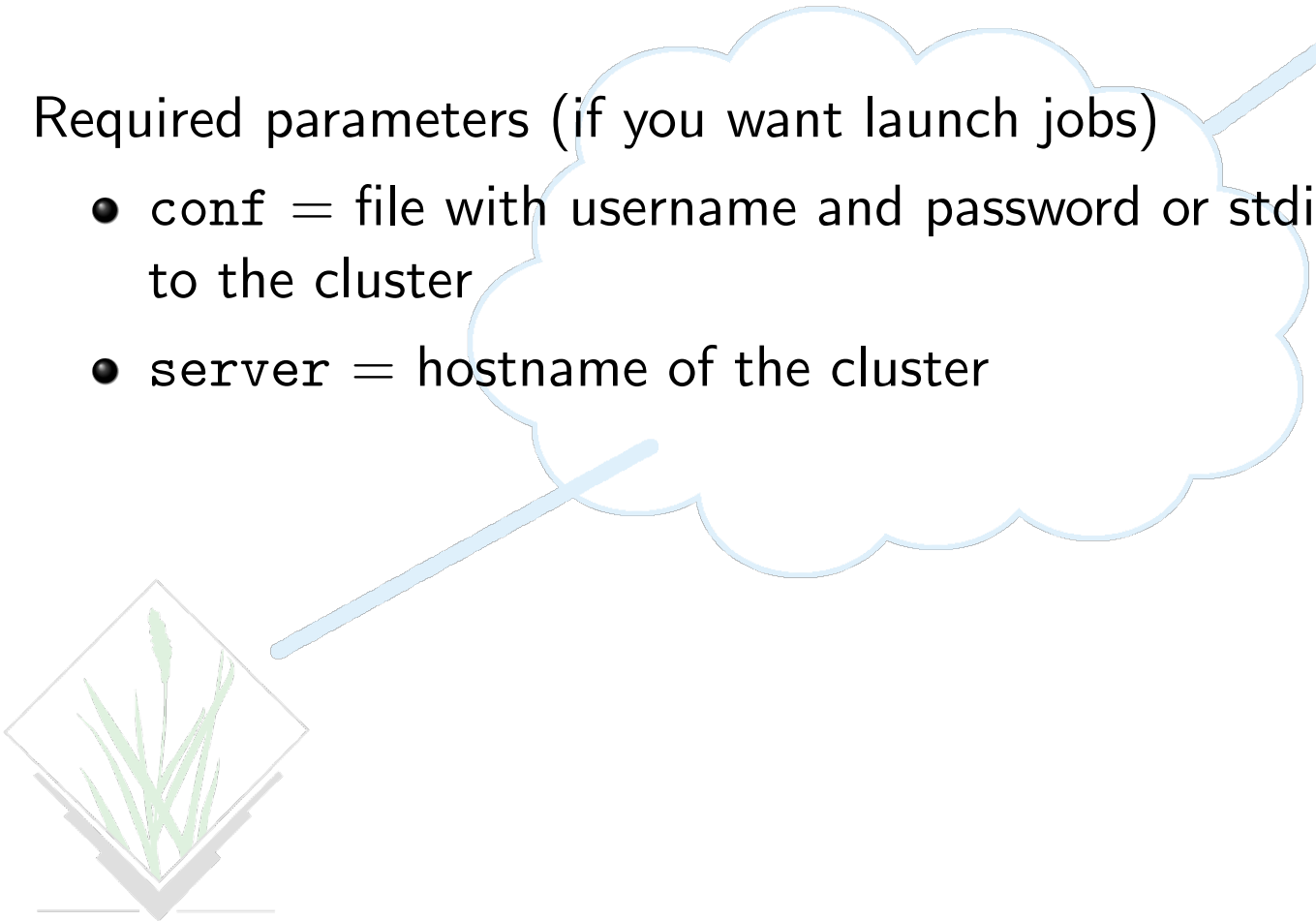
Implementation: how does it work?



Required parameters (if you want launch jobs)

- `conf` = file with username and password or stdin to connect to the cluster

Implementation: how does it work?



Required parameters (if you want launch jobs)

- `conf` = file with username and password or stdin to connect to the cluster
- `server` = hostname of the cluster

Implementation: how does it work?

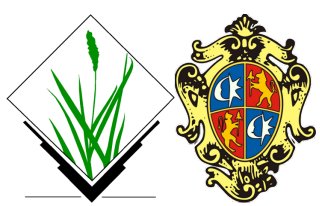
Required parameters (if you want launch jobs)

- `conf` = file with username and password or stdin to connect to the cluster
- `server` = hostname of the cluster
- `qsub_script` = file containing qsub script (template)

Implementation: how does it work?

Required parameters (if you want launch jobs)

- `conf` = file with username and password or stdin to connect to the cluster
- `server` = hostname of the cluster
- `qsub_script` = file containing qsub script (template)
- `grass_script` = own GRASS GIS script



Implementation: how does it work?

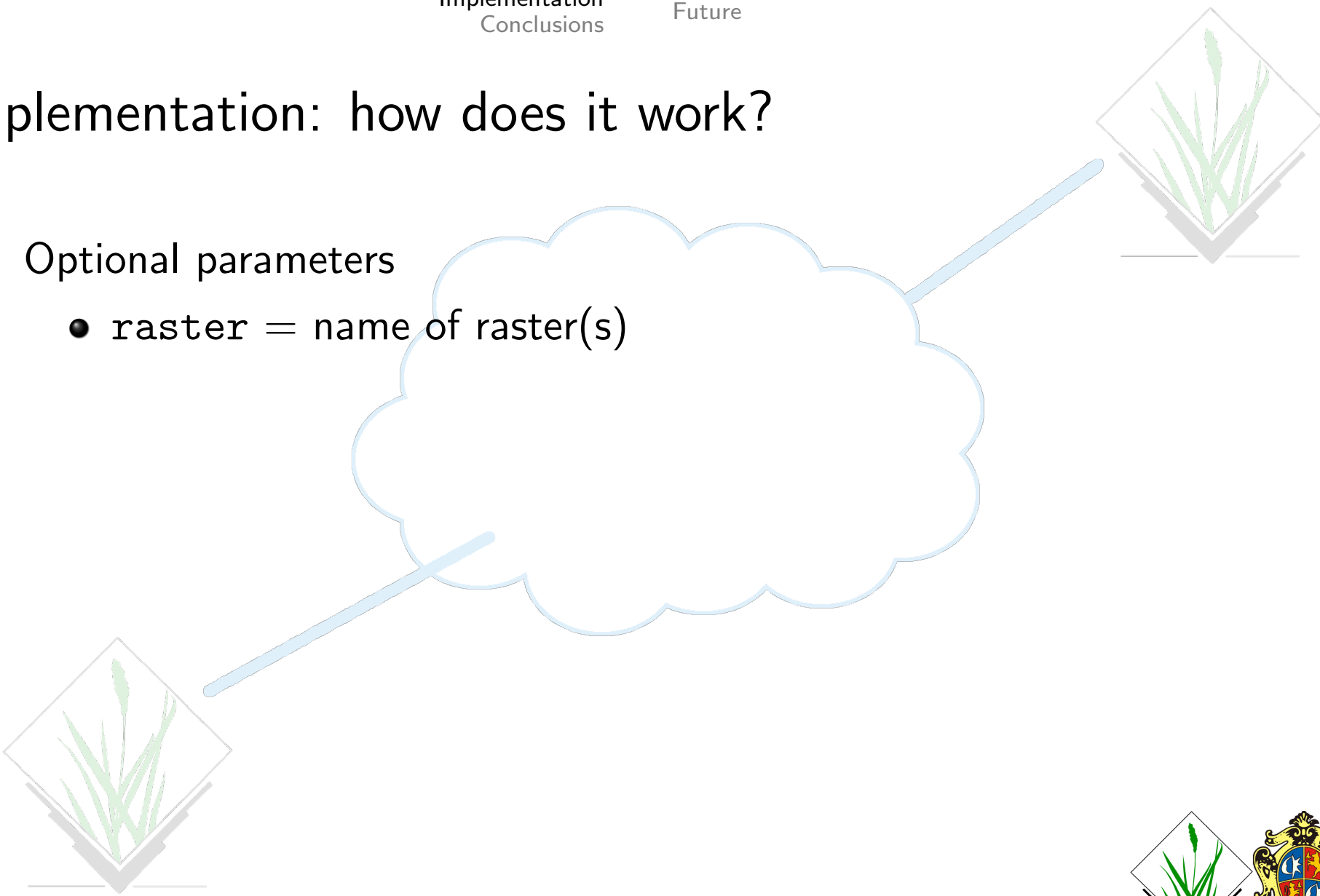
Required parameters (if you want launch jobs)

- `conf` = file with username and password or stdin to connect to the cluster
- `server` = hostname of the cluster
- `qsub_script` = file containing qsub script (template)
- `grass_script` = own GRASS GIS script

g.cloud conf=/tmp/passwd server=giscluster

qsub_script=test_launch_SGE_grassjob.sh grass_script=test_novariables.sh

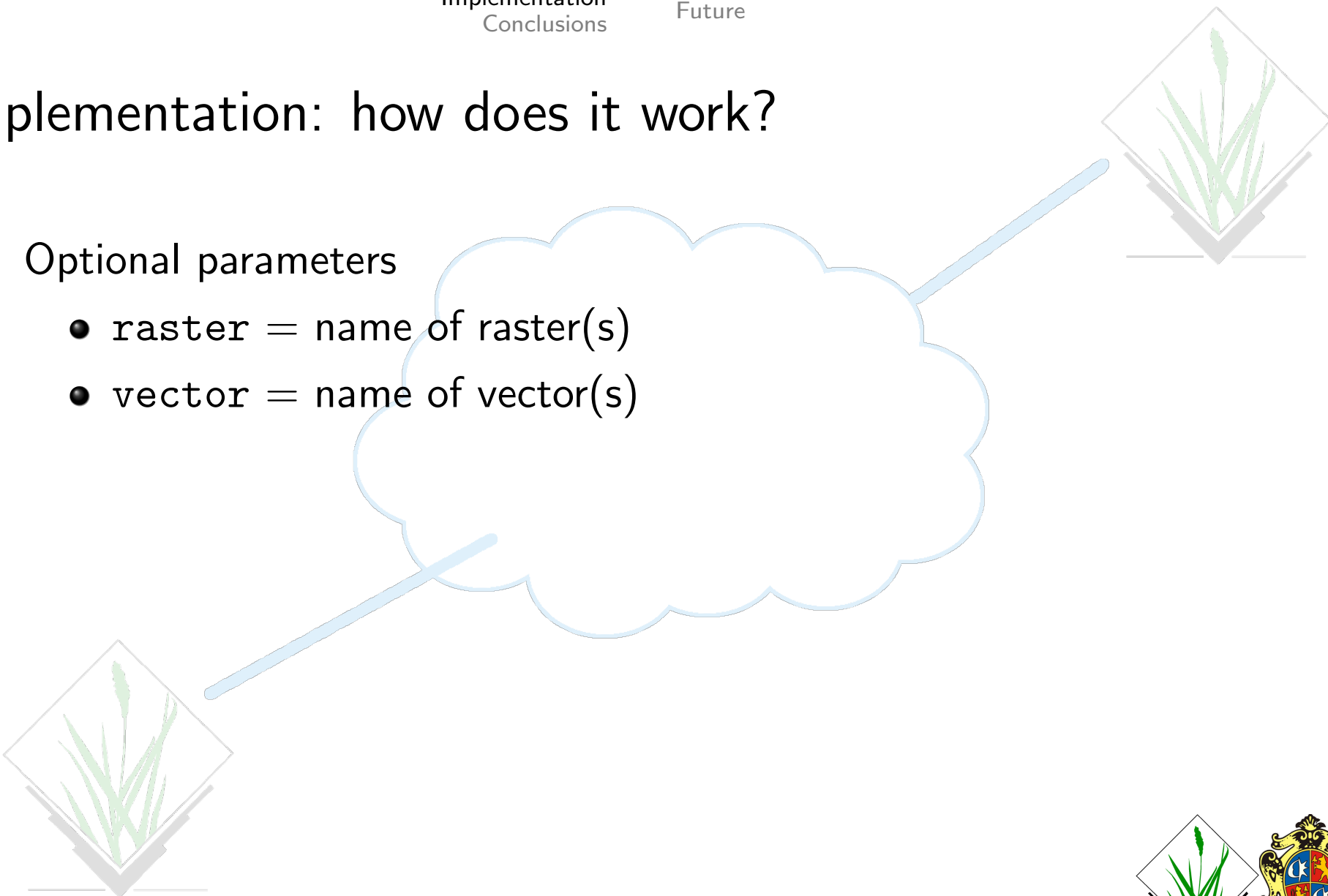
Implementation: how does it work?



Optional parameters

- `raster` = name of raster(s)

Implementation: how does it work?



Optional parameters

- `raster` = name of raster(s)
- `vector` = name of vector(s)

Implementation: how does it work?

Optional parameters

- `raster` = name of raster(s)
- `vector` = name of vector(s)
- `variables` = string with name and values of variables stated as Python dictionary "'key':['value0','value1']"

Implementation: how does it work?

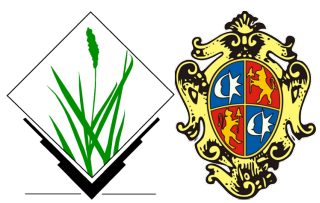
Optional parameters

- `raster` = name of raster(s)
- `vector` = name of vector(s)
- `variables` = string with name and values of variables stated as Python dictionary "'key':['value0','value1']"
- `email` = user's email address

Implementation: how does it work?

Optional parameters

- `raster` = name of raster(s)
- `vector` = name of vector(s)
- `variables` = string with name and values of variables stated as Python dictionary "'key':['value0','value1']"
- `email` = user's email address
- `reconnect` = reconnect with old job to see if it is finished and move the output data back on the client

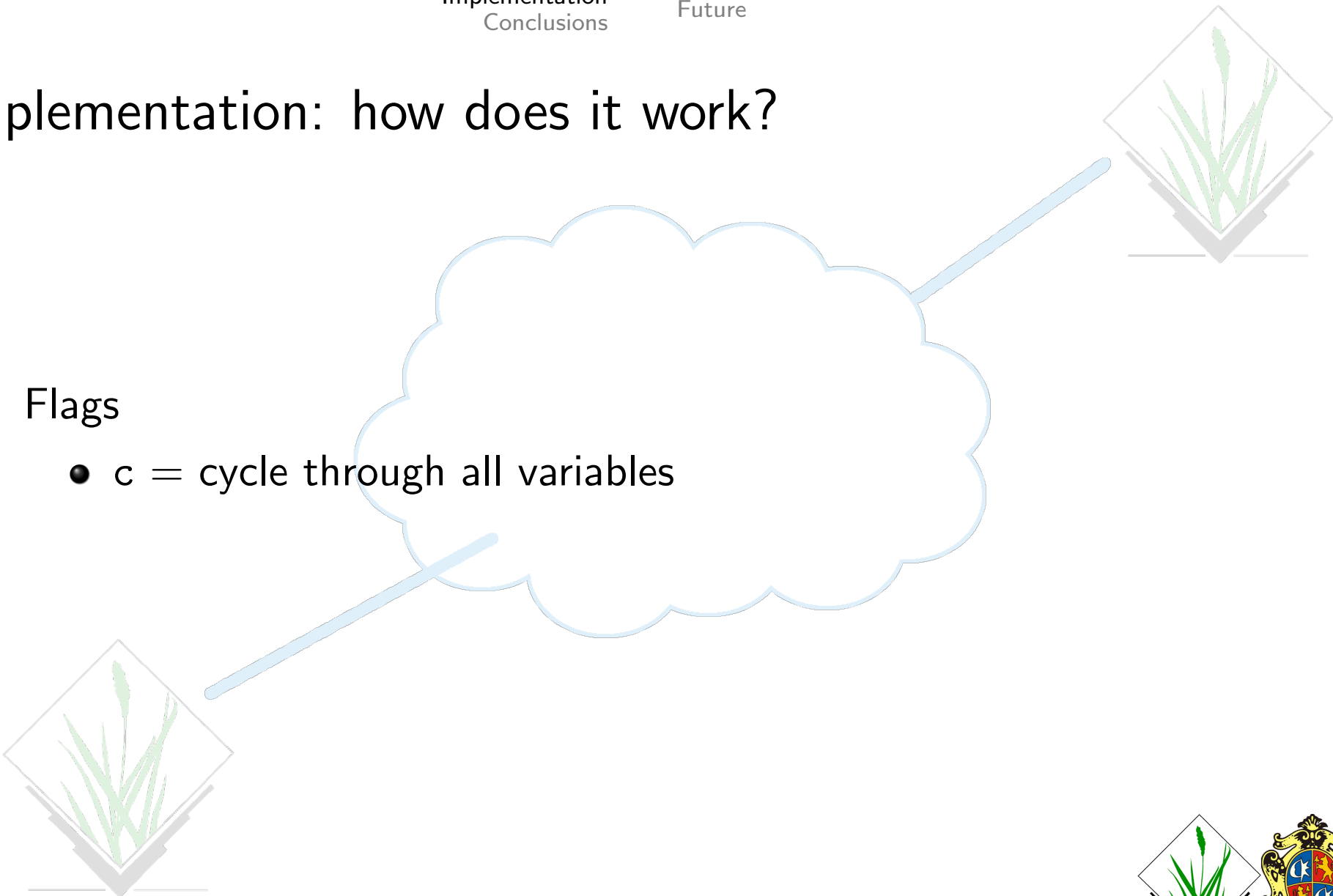


Implementation: how does it work?

Optional parameters

- `raster` = name of raster(s)
- `vector` = name of vector(s)
- `variables` = string with name and values of variables stated as Python dictionary "'key':['value0','value1']"
- `email` = user's email address
- `reconnect` = reconnect with old job to see if it is finished and move the output data back on the client
- `path` = path to the folder which must be accessible from cluster frontend and blades

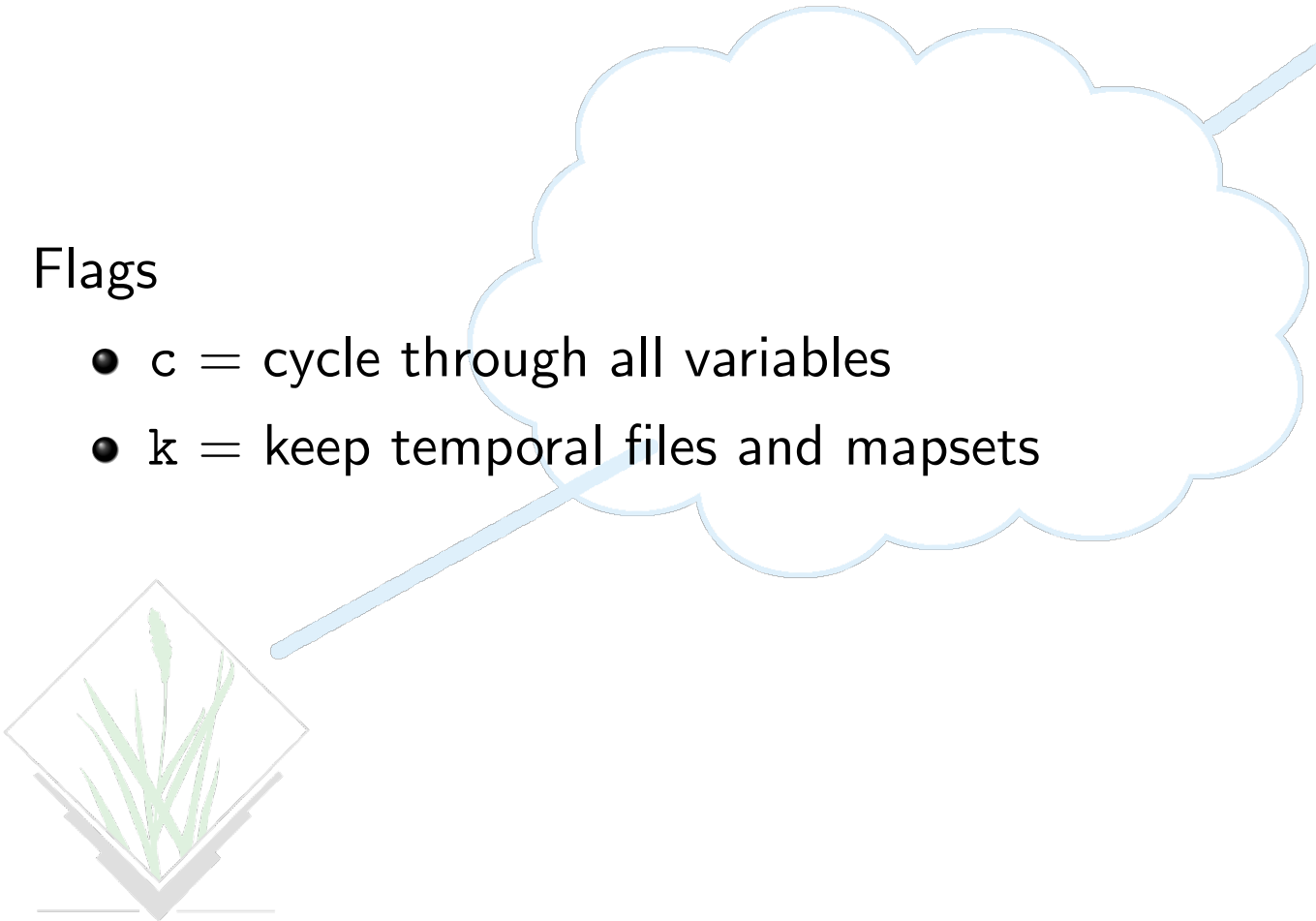
Implementation: how does it work?



Flags

- c = cycle through all variables

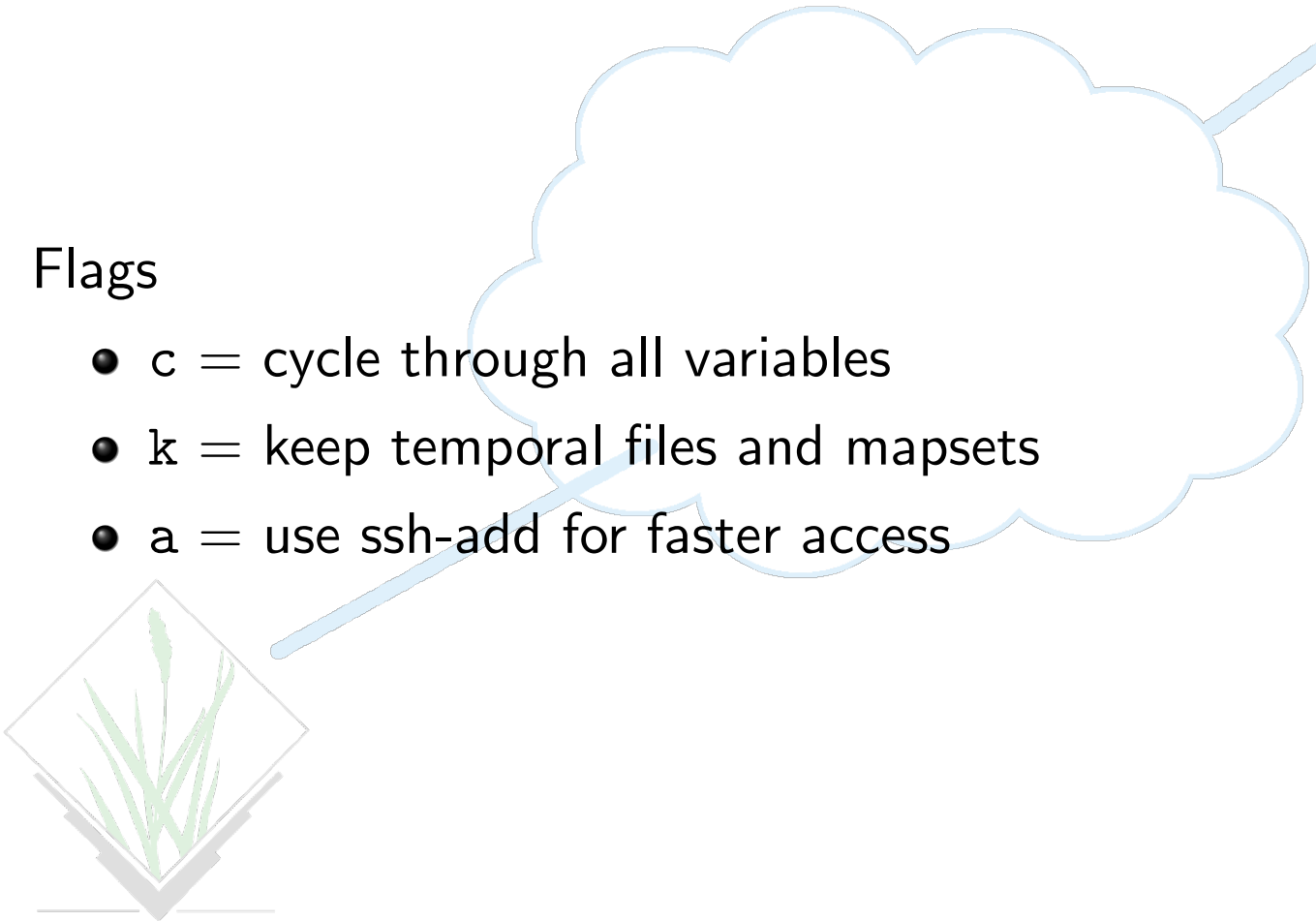
Implementation: how does it work?



Flags

- `c` = cycle through all variables
- `k` = keep temporal files and mapsets

Implementation: how does it work?



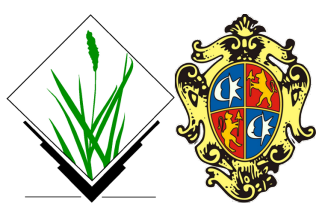
Flags

- c = cycle through all variables
- k = keep temporal files and mapsets
- a = use ssh-add for faster access

Implementation: more examples

North Carolina sample data

- r.texture example
  g.cloud conf=/tmp/passwd server=giscluster
  qsub_script=test_launch_SGE_grassjob.sh grass_script=test_onevariable_raster.sh
  variables="'TEXT' :
  ['asm','contrast','corr,var','idm','sa','se','sv','entr','dv','de','moc1','moc2']"
  raster=lsat7_2002_40



Implementation: more examples

North Carolina sample data

- r.texture example
  g.cloud conf=/tmp/passwd server=giscluster
  qsub_script=test_launch_SGE_grassjob.sh grass_script=test_onevariable_raster.sh
  variables="'TEXT' :
  ['asm','contrast','corr,var','idm','sa','se','sv','entr','dv','de','moc1','moc2']"
  raster=lsat7_2002_40
- r.sun example

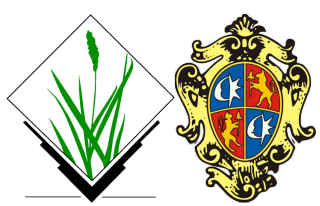
  g.cloud conf=/tmp/passwd server=giscluster

  qsub_script=test_launch_SGE_grassjob.sh grass_script=test_onevariable_sun.sh

  variables="'DOY' : [1,2,3,4,5,6,7,8,9,10]" raster=elevation

Implementation: more examples

- more variable example
  g.cloud conf=/tmp/passwd server=giscluster qsub_script=test_launch_SGE_grassjob.sh grass_script=test_morevariables_sun.sh variables="'NPOINT' : [10,20], 'BUFFERDIST' : [100,500]"
  use number point 10 with buffer distance 100; and number point 20 with buffer distance 500



Implementation: more examples

- more variable example
  g.cloud conf=/tmp/passwd server=giscluster qsub_script=test_launch_SGE_grassjob.sh grass_script=test_morevariables_sun.sh variables="'NPOINT' : [10,20], 'BUFFERDIST' : [100,500]"
  use number point 10 with buffer distance 100; and number point 20 with buffer distance 500
- more variable example

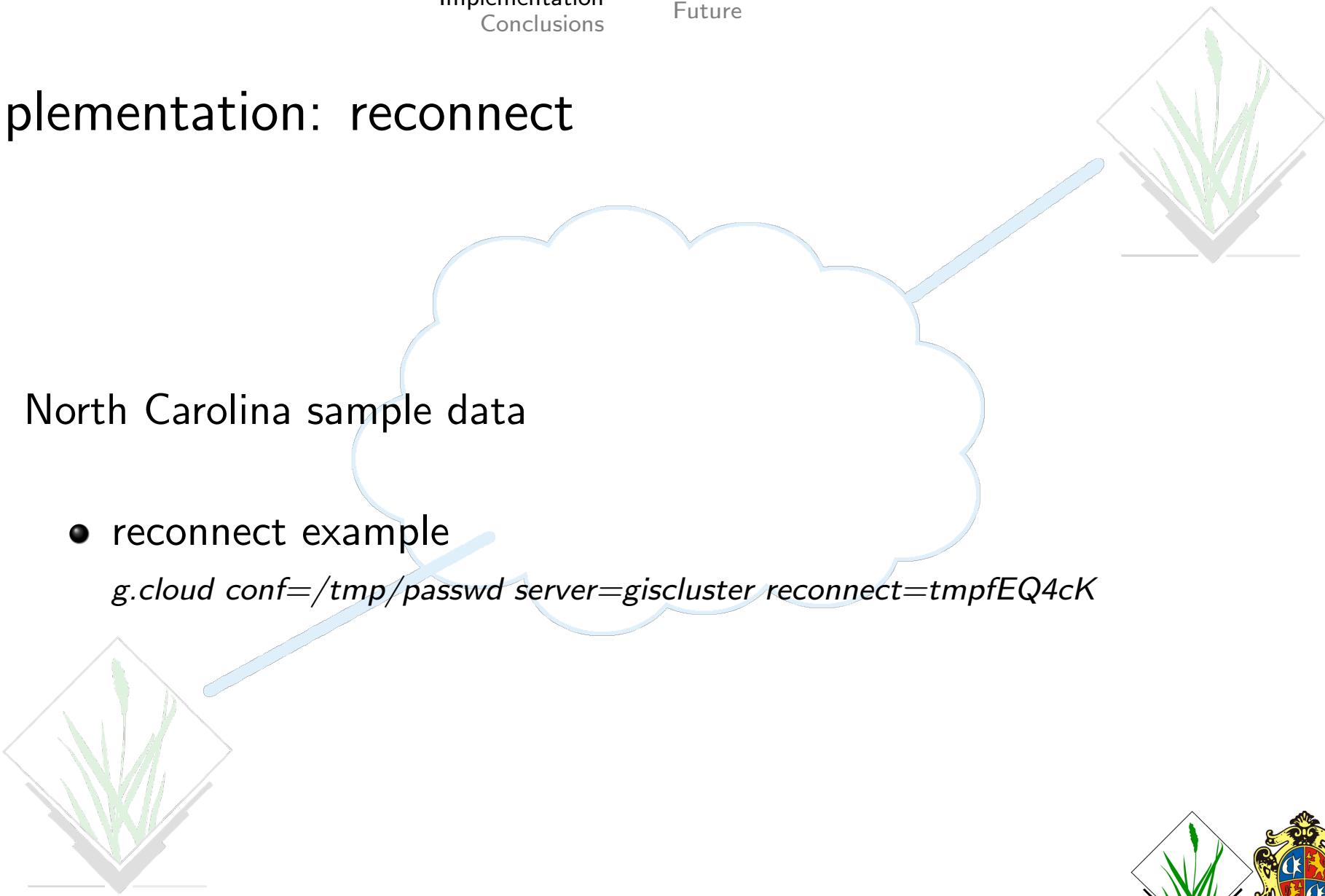
  g.cloud -c conf=/tmp/passwd server=giscluster

  qsub_script=test_launch_SGE_grassjob.sh grass_script=test_morevariables_sun.sh

  variables="'NPOINT' : [10,20], 'BUFFERDIST' : [100,500]"

  use number point 10 with buffer distance 100 and 500; number point 20 with buffer distance 100 and 500

Implementation: reconnect

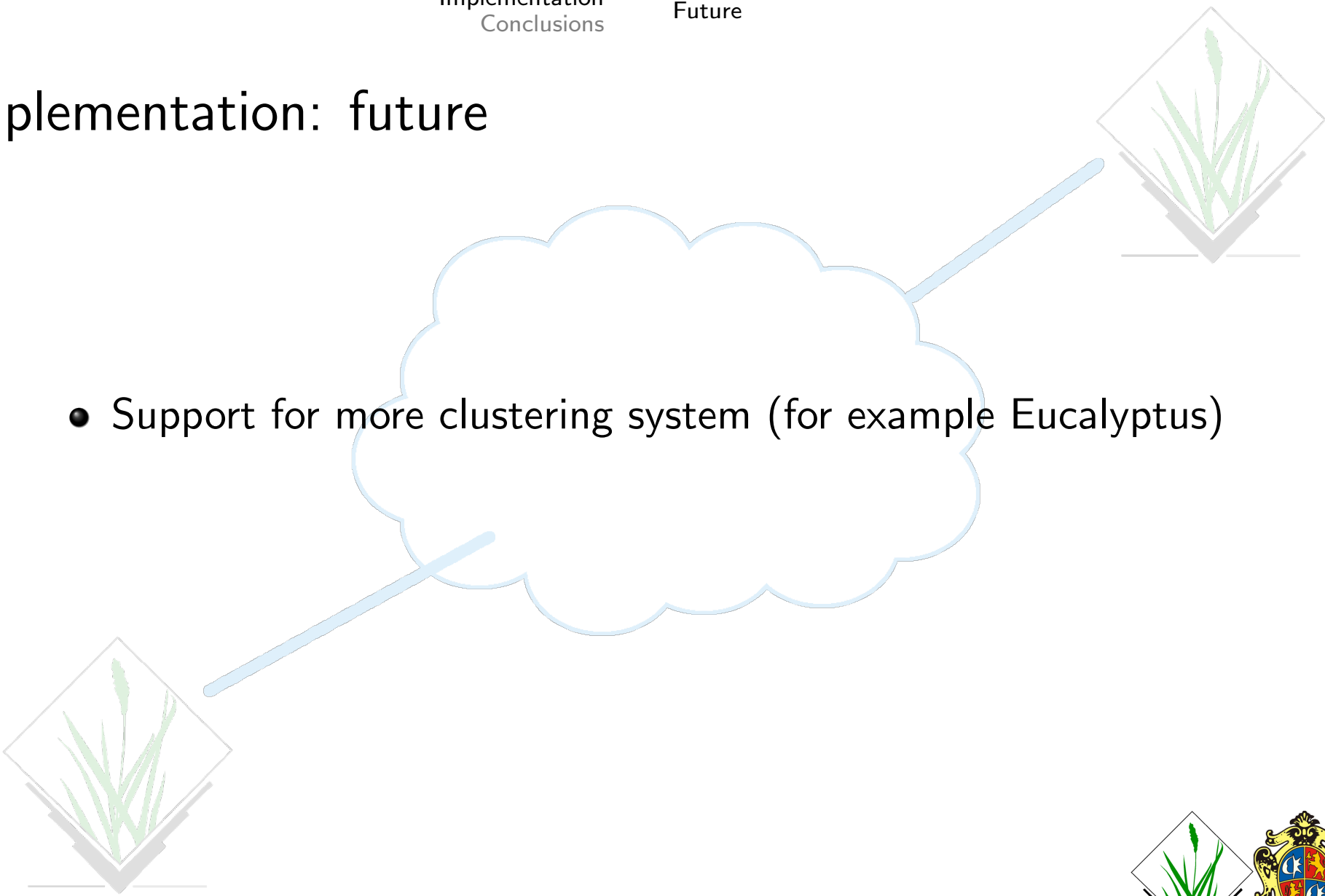


North Carolina sample data

- reconnect example

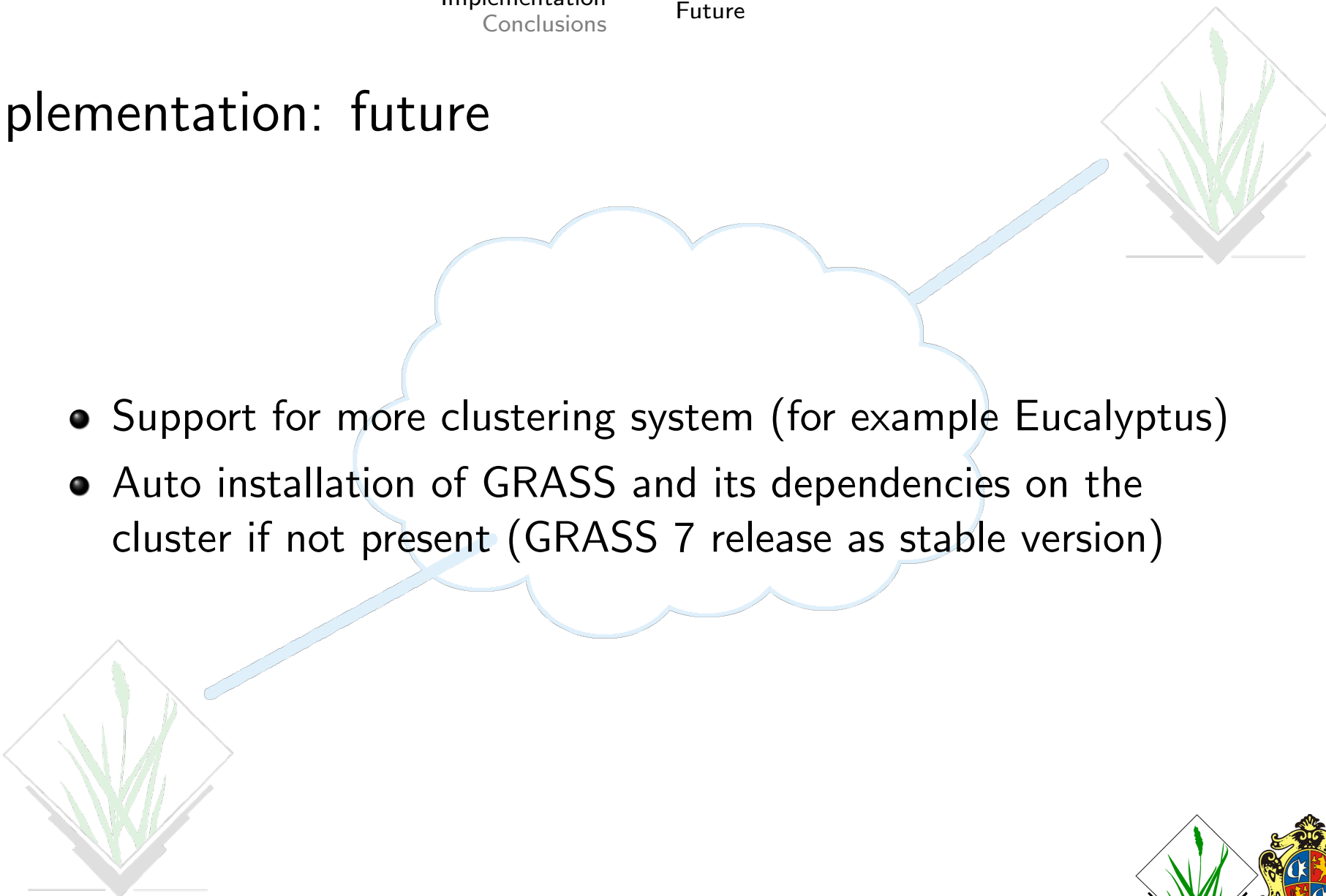
  g.cloud conf=/tmp/passwd server=giscluster reconnect=tmpfEQ4cK

Implementation: future



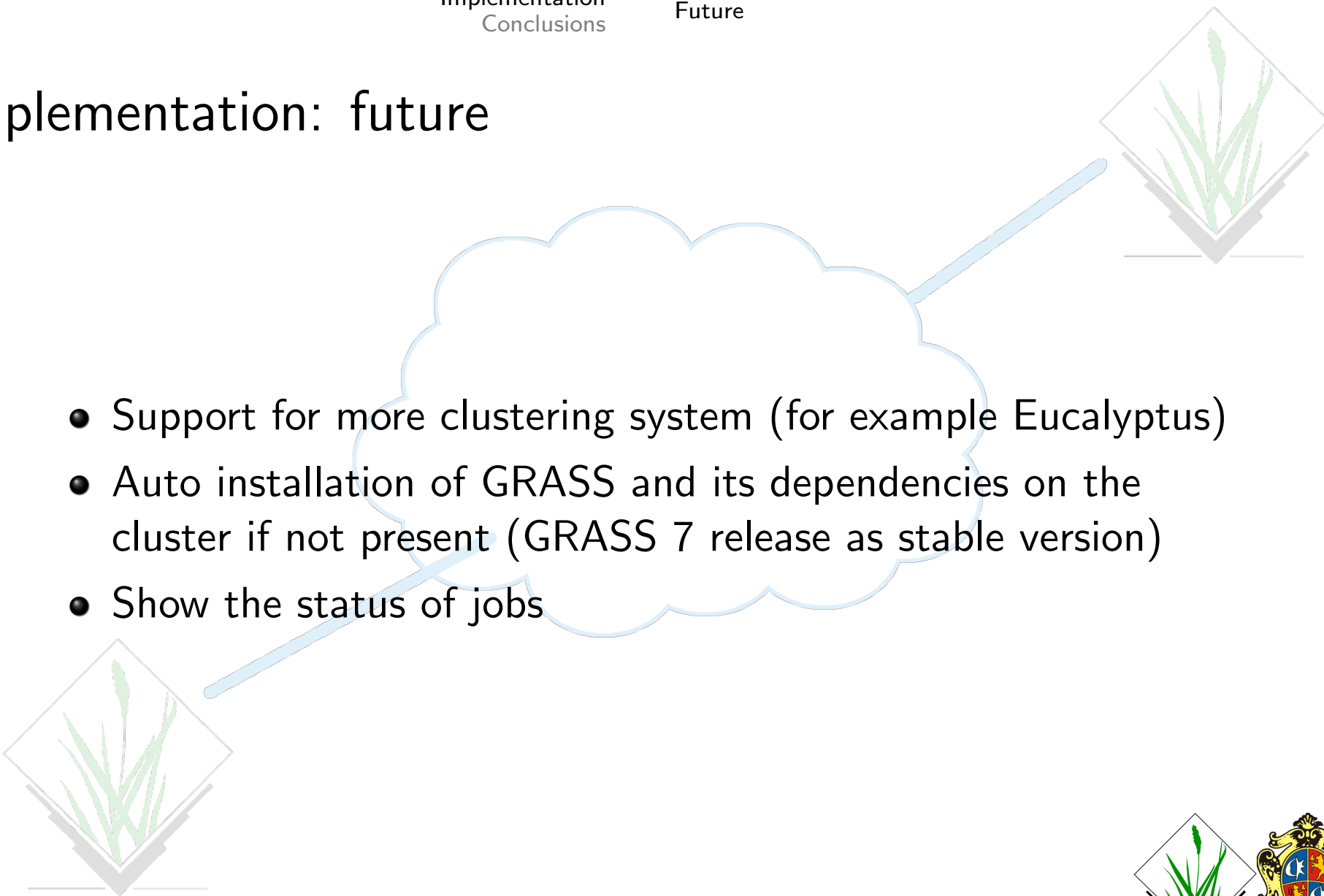
- Support for more clustering system (for example Eucalyptus)

Implementation: future



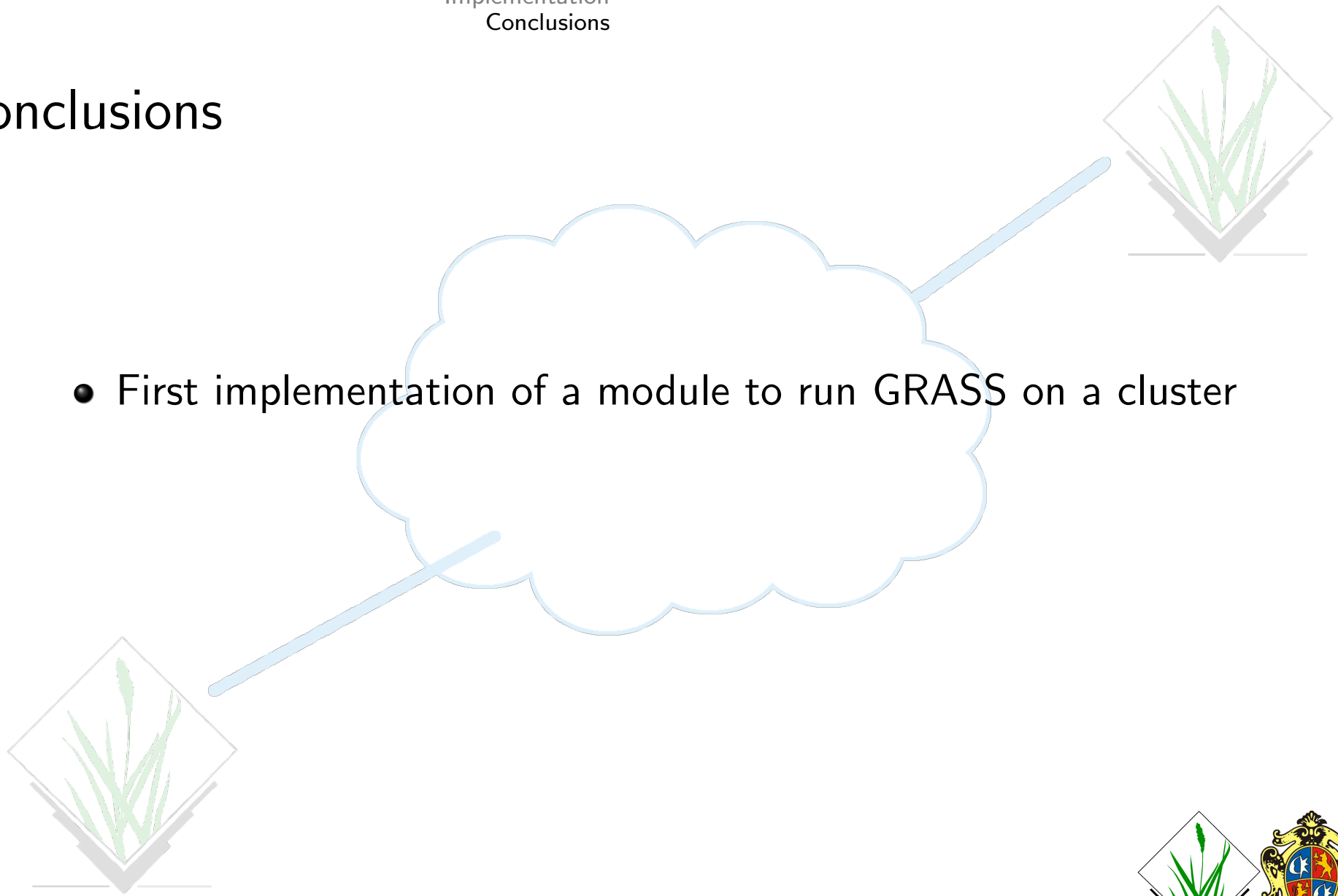
- Support for more clustering system (for example Eucalyptus)
- Auto installation of GRASS and its dependencies on the cluster if not present (GRASS 7 release as stable version)

Implementation: future



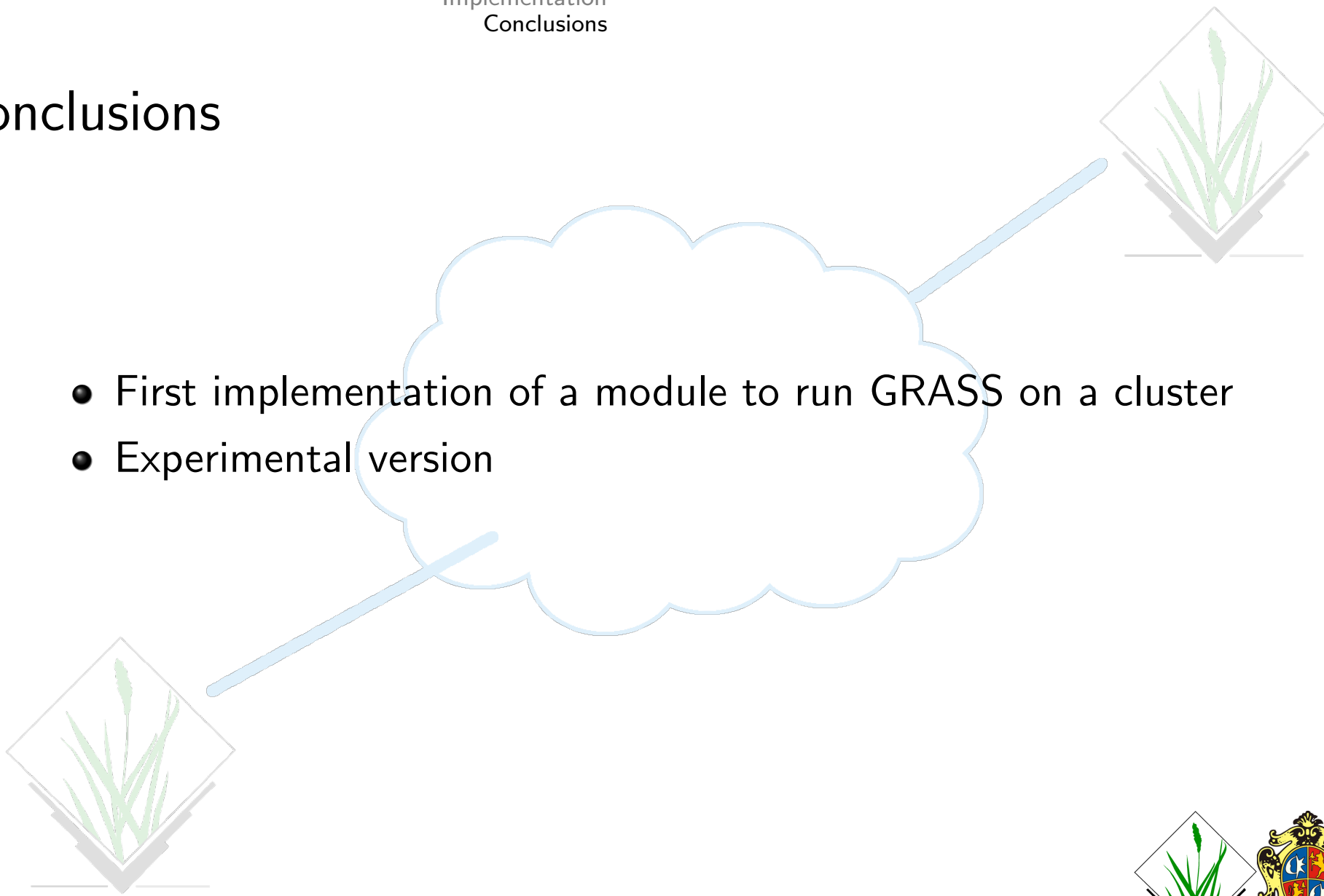
- Support for more clustering system (for example Eucalyptus)
- Auto installation of GRASS and its dependencies on the cluster if not present (GRASS 7 release as stable version)
- Show the status of jobs

Conclusions



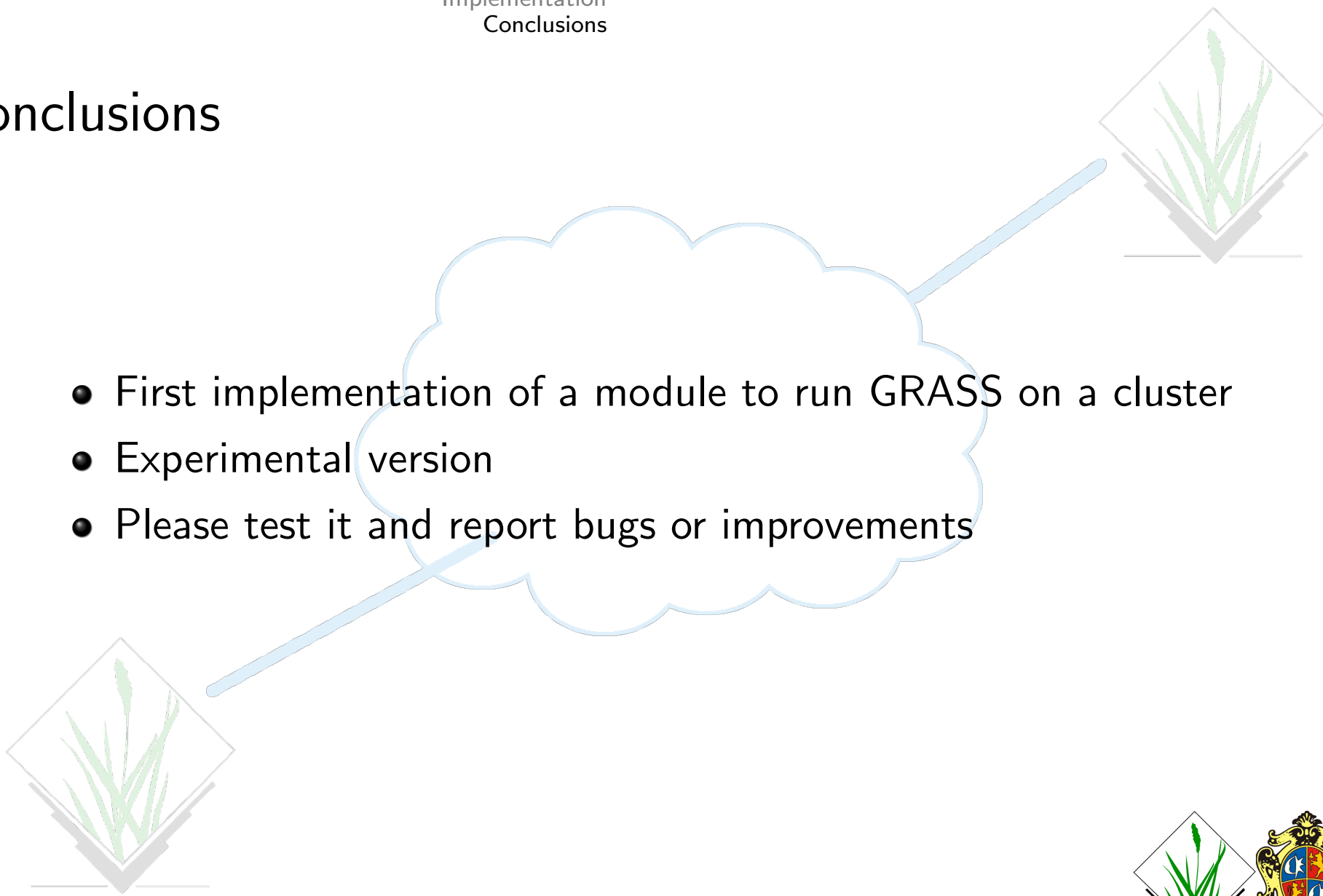
- First implementation of a module to run GRASS on a cluster

Conclusions



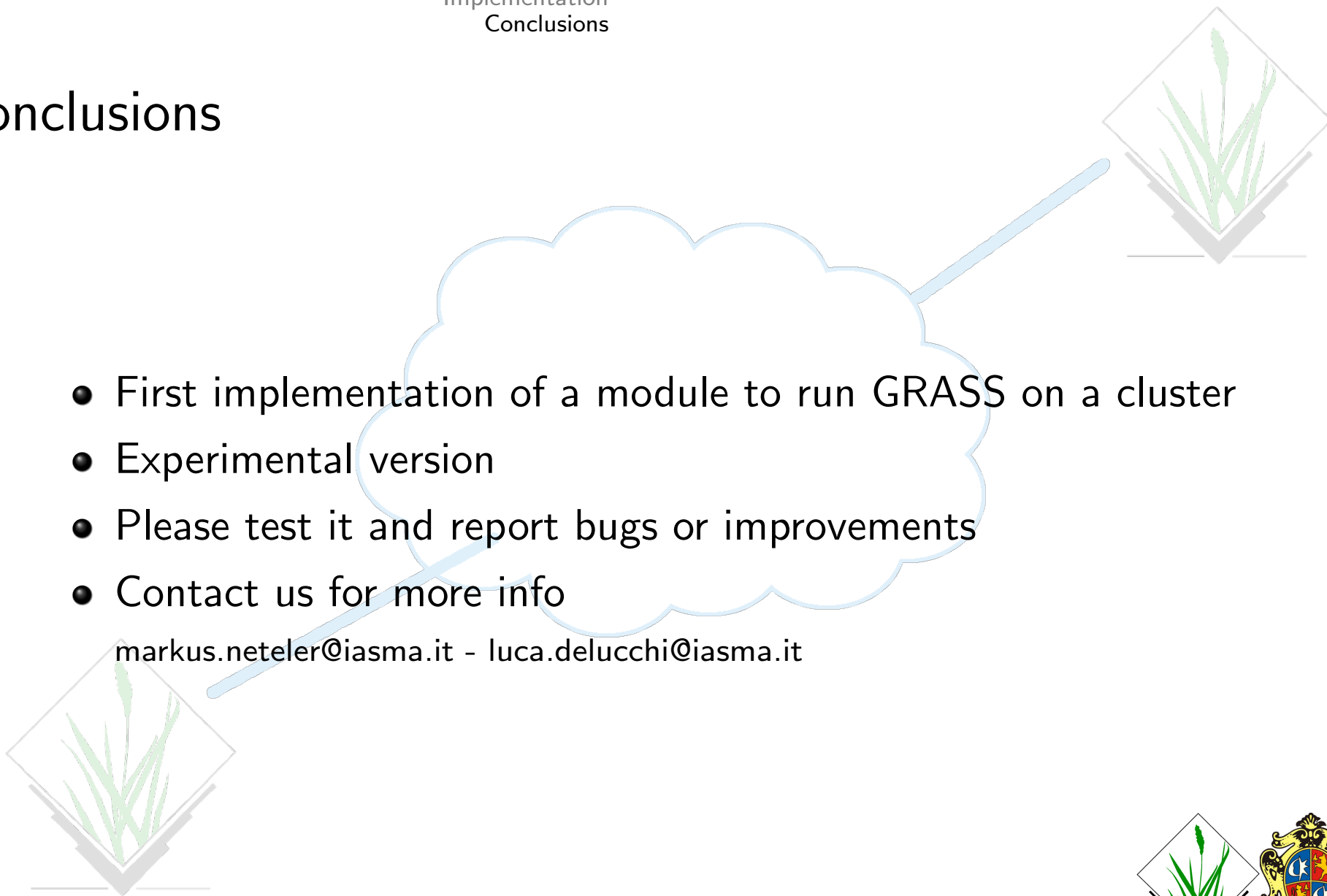
- First implementation of a module to run GRASS on a cluster
- Experimental version

Conclusions



- First implementation of a module to run GRASS on a cluster
- Experimental version
- Please test it and report bugs or improvements

Conclusions



- First implementation of a module to run GRASS on a cluster
- Experimental version
- Please test it and report bugs or improvements
- Contact us for more info

markus.neteler@iasma.it - luca.delucchi@iasma.it

Conclusions

Thanks for your attention

gis.cri.fmach.it