SIESTA: building, deployment and execution

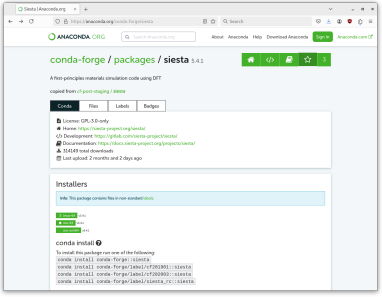
Dr. José María Escartín Esteban

21st November 2025

MAX DRIVING THE EXASCALE TRANSITION
ICN2 Institut Català de Nanociència i Nanotecnologia
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación
CSIC Consejo Superior de Investigaciones Científicas

Conda

- Conda is a package manager.
- Conda-Forge is community repository of recipes and packages.
- SIESTA available on Conda-forge: `https://anaconda.org/conda-forge/siesta`




```
$ conda install -c conda-forge siesta=5.4.1=*openmpi*
```

- Strength: easy access to SIESTA: somebody else built the package, so you can directly deploy.
- Weakness: a Conda package may run on a range of CPUs $\Longrightarrow$ package not optimized for your particular instruction set $\Longrightarrow$ SIESTA not as performant as it could be.

Question 0:
How do I get to run SIESTA *efficiently* on a computer?

The full seven steps

1. **Check** the hardware requirements.
2. **Prepare** the build environment.
3. **Download** the SIESTA source code.
4. **Build** SIESTA, its utilities, and any needed dependencies.
5. **Test** the binaries.
6. **Deploy** the files you built (binaries, libraries, etc.).
7. **Run** the executables in the correct environment.

Question 1:
Can I run SIESTA on any computer?

Hardware requirements

- SIESTA can work on a broad variety of computer architectures, from a Raspberry PI to massively parallel supercomputers.
- Recommended minimum requirements:
  - 1 GB RAM (although small atomic structures can be studied with less).
  - 2 GB disk storage, in particular if you want to run the SIESTA test suite.
- SIESTA will likely work on any CPU, but bear in mind that most development and testing is done on x86_64 (Intel/AMD) and ARMv8 architectures.

Question 2:
What software do I need to build SIESTA?

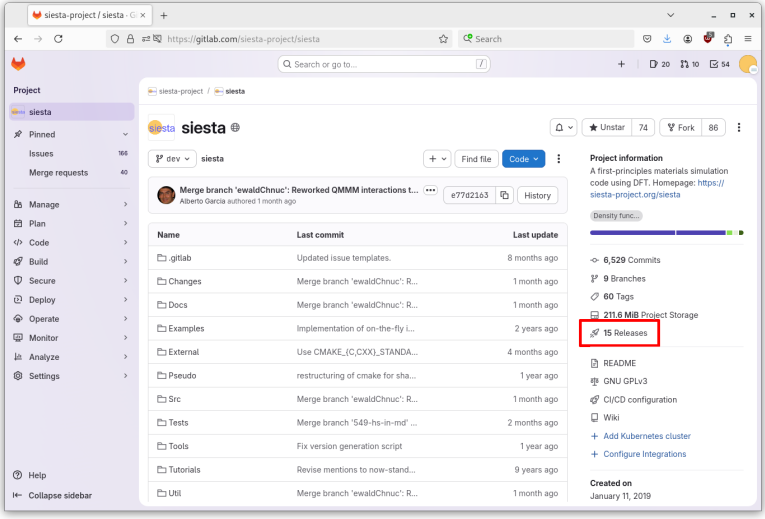
Building software requirements

- CMake version $\geq$ 3.20 (released in March 2021).
- A Fortran compiler with full support of the Fortran 2003 standard and partial support of the Fortran 2008 standard (gfortran, ifort, CRAY Fortran, etc.).
  Note: mixing old compilers with new hardware is usually a bad idea.
- A C/C++ companion compiler.
- If you want to build a parallel version of SIESTA, you will also need a MPI distribution (including development files).
- If you want to offload to a Nvidia (/AMD) GPU, you will need the corresponding CUDA (/HIP) distribution.
- If you want to build the SIESTA user manual, you will also need a LaTeX distribution.
- If you want to automatically download the source code of dependencies, a non-antique version of git.

Question 3:
Where can I find the authoritative source code of SIESTA?

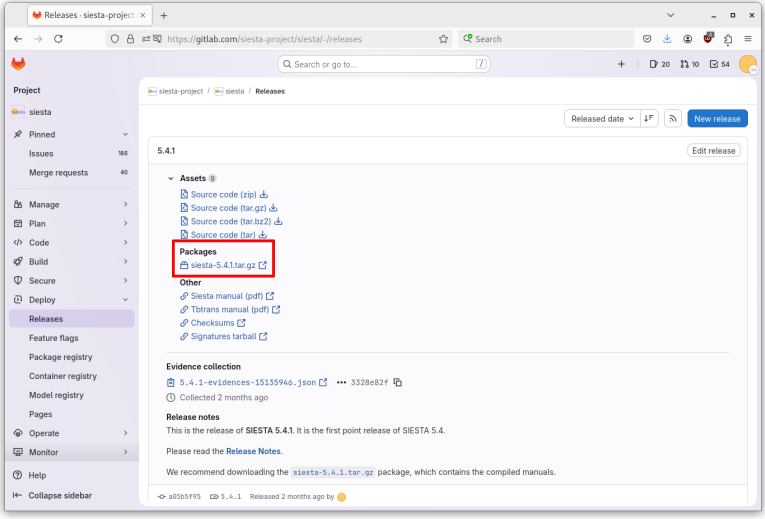
GitLab: SIESTA

https://gitlab.com/siesta-project/siesta/



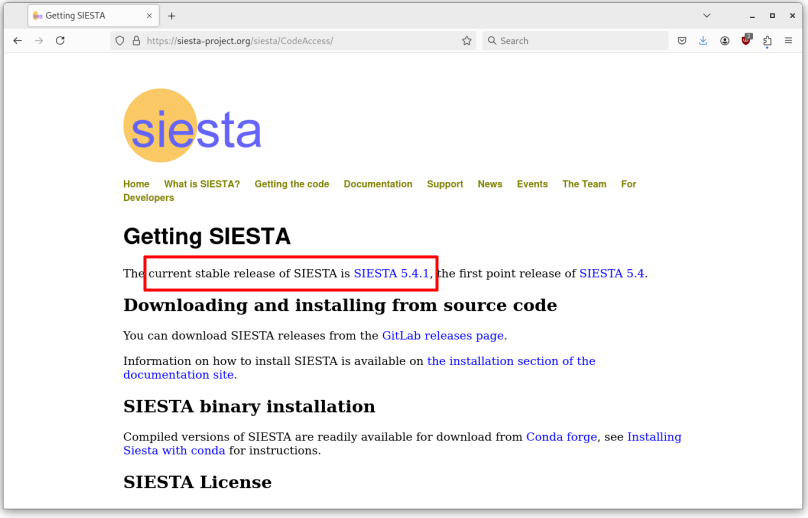
GitLab: SIESTA releases

https://gitlab.com/siesta-project/siesta/-/releases/



Which version of SIESTA should I use?

We strongly recommend that you use the current **stable release** of SIESTA.



Should I use SIESTA stable even if I am an advanced user?

Advanced users that want to try arbitrary branches or versions of SIESTA should really avoid the GitLab download button, and interact with the repository using git (git version ≥ 2.13).

These non-release versions are generally **unsupported** by the SIESTA development team.

Question 4:
How do I build SIESTA?

Building SIESTA 5

Download tarball:

```
$ wget https://gitlab.com/siesta-project/siesta/-/releases/5.4.1/downloads/siesta-5.4.1.tar.gz
```

Extract files:

```
$ tar -xvvzf siesta-5.4.1.tar.gz
```

Enter source directory:

```
$ cd siesta-5.4.1
```

Initialize build directory:

```
$ cmake -S. -B_build
```

Build:

```
$ cmake --build _build -j 4
```

Basic building options

- Specify Fortran compiler

  `FC=gfortran cmake ...`
- Specify Fortran compiler flags

  `cmake -DFortran_FLAGS='-O3 -march=native'`
- Explicitly enable/disable MPI (default: `ON` if MPI compiler found, otherwise `OFF`):

  `cmake ... -DSIESTA_WITH_MPI=ON|OFF`
- Explicitly enable/disable OpenMP (default: `OFF`):

  `cmake ... -DSIESTA_WITH_OPENMP=ON|OFF`
- Specify toolchain file (some samples in `Config/cmake/toolchains/`, but this directory will probably be superseded soon by the Build Tools repository `https://gitlab.com/siesta-project/ecosystem/build-tools`):

  `cmake ... -DSIESTA_TOOLCHAIN=/path/to/toolchain/file`

Check the **SIESTA manual** for details about all the building options.

Question 5:
How do I test my SIESTA executable?

Testing SIESTA

- The `ctest` command provided by CMake allows to test SIESTA and its dependencies. See `Tests/README` for more details.
- List (`-N`) all the execution tests [i.e., excluding (`-E`) the verification tests]:

```
$ cd _build
$ ctest -N -E verify
Test project /home/jme52/SIESTA/siesta-5.2.0/_build
  Test   #1: fdf-sample-test
  Test   #2: fdf-units-test
  Test   #3: fdf-ambiguous-units-test
  Test   #4: XMLF90_SAX_Features
  ...
  Test #166: siesta-00.BasisSets-ghost_atom_mpi4
  Test #167: siesta-01.PseudoPotentials-psf_mpi4-mkdir
  ...
  Test #579: citations-test

Total Tests: 579
```

- A less comprehensive set of tests can be chosen by selecting (`-L`) the `simple` ones:

```
$ ctest -N -L simple -E verify
```

Testing SIESTA

Expected output when you execute the tests:

```
$ ctest -L simple -E verify
Test project /home/jme52/SIESTA/siesta-5.4.1/_build
      Start 143: siesta-00.BasisSets-default_basis_mpi4-mkdir
 1/90 Test #143: siesta-00.BasisSets-default_basis_mpi4-mkdir ..............   Passed    0.01 sec
      Start 144: siesta-00.BasisSets-default_basis_mpi4-fdf-setup
 2/90 Test #144: siesta-00.BasisSets-default_basis_mpi4-fdf-setup ..........   Passed    0.01 sec
      Start 145: siesta-00.BasisSets-default_basis_mpi4
 3/90 Test #145: siesta-00.BasisSets-default_basis_mpi4 ....................   Passed    1.61 sec
      Start 167: siesta-01.PseudoPotentials-psf_mpi4-mkdir
 4/90 Test #167: siesta-01.PseudoPotentials-psf_mpi4-mkdir .................   Passed    0.01 sec
 ...
100% tests passed, 0 tests failed out of 90

Label Time Summary:
simple    =  84.38 sec*proc (28 tests)

Total Test time (real) =  84.93 sec
```

Question 6:
How do I deploy SIESTA?

Tell cmake where to install SIESTA, and install it there:

```
$ cmake -S. -B_build -DCMAKE_INSTALL_PREFIX=/path/to/installation
$ cmake --build _build -j 4
$ cmake --install _build
$ ls /path/to/installation
   bin  include  lib64  share
```

Then make your environment aware of this installation:

```
$ cat siestarc.sh
#!/bin/sh

LD_LIBRARY_PATH="/path/to/installation/lib:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH

PATH="/path/to/installation/bin:$PATH"
export PATH
```

Question 7:
How do I execute SIESTA?

Execution of SIESTA

Serial:

```
$ siesta < input.fdf > output.txt
```

MPI:

```
$ mpirun -np 12 siesta < input.fdf > output.txt
```

OpenMP:

```
$ OMP_NUM_THREADS=4 mpirun -np 12 siesta < input.fdf > output.txt
```

Read the documentation of your HPC facility and your MPI manual for advanced options (pinning, etc.)

Question 8:
Check, prepare, download, build, test, deploy, and run: can I take any shortcuts?

HPC facility modules

If you are running SIESTA on a HPC facility, check first if SIESTA is already installed!

We are working on SIESTA modules for all EuroHPC partitions.

Question 9:
Are there other methods or frameworks to build and/or deploy?

Other frameworks

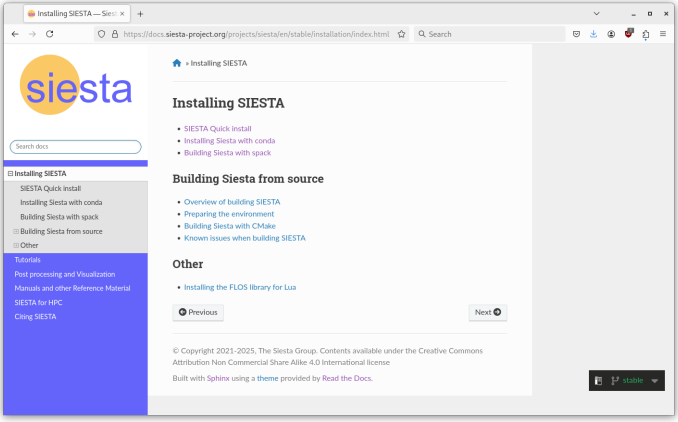
Spack

Recipes available in the Build Tools repository of the SIESTA Project:
`https://gitlab.com/siesta-project/ecosystem/build-tools`

Question 10:
Where can I learn more?
How can I get help if I need it?

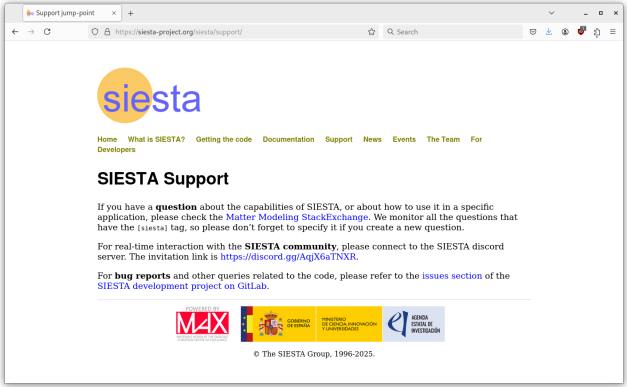
SIESTA documentation

https://docs.siesta-project.org/projects/siesta/en/stable/installation/



SIESTA support

https://siesta-project.org/siesta/support/



Thank you for your attention.