

# Getting Started

## Laptop/DelftBlue

# Course repository

→ <https://github.com/carstenbauer/JuliaDelft24>

# Remarks

→ `git pull`

→ `julia --version == 1.10.5`

If not:

→ `juliaup add 1.10`

→ `juliaup default 1.10`

# Julia on Your Laptop

# A basic Julia test

Open a terminal in [JuliaDelft24](#)

Start Julia

```
julia --project
```

Run this code

```
julia> using SysInfo  
julia> sysinfo()
```

**Julia on DelftBlue**

# Access DelftBlue via SSH

## Terminal

```
ssh <netid>@login.delftblue.tudelft.nl
```

## VS Code

```
Remote-SSH: Connect to Host...
```

# Loading software modules

What we need for the course

```
module use /projects/julia/modulefiles  
module use juliahpc  
module use nvhpc # MPI+CUDA
```



# A basic Julia test (on DelftBlue)

Open a terminal in

```
cd /scratch/$USER/JuliaDelft24
```

Start Julia

```
julia --project
```

Run this code

```
julia> using SysInfo  
julia> sysinfo()
```

# Jobs are scheduled with SLURM.

Submit a job:

```
sbatch job-script.sh
```

Check on your queued/running jobs:

```
squeue --me
```

# A few nodes are reserved for the course.

## 2 CPU nodes

2x Intel Cascade Lake

185 GB memory

48 cores total

## 2 GPU nodes

2x AMD Zen 2

250 GB memory

48 cores total

4x NVIDIA V100S



# Accessing compute nodes (with VS Code)

On the target node

```
module load code  
code tunnel
```

On your laptop

In VS Code:

```
Remote Tunnels: Connect to Tunnel
```

# Put the Julia depot on the parallel file system.

`JULIA_DEPOT_PATH` = where Julia stores stuff  
packages  
binary dependencies  
...

Why not `$HOME`?

Quotas

Can be read-only for compute jobs

# Julia VSCode extension requires a wrapper.

```
[...]
```

```
# Load modules
```

```
module use /projects/Julia/modulefiles
```

```
module load juliahpc
```

```
module load nvhpc
```

```
# Pass on all arguments
```

```
exec julia "${@}"
```

# Julia VS Code integration via extension.

The screenshot displays the VS Code interface with the Julia extension. The Explorer sidebar on the left shows the project structure, including folders like `.github`, `.vscode`, `benchmark`, `bin`, `deps`, `docs`, `src`, `test`, `.gitignore`, `.travis.yml`, `appveyor.yml`, and `LICENSE.md`. The Julia Explorer sidebar shows the workspace structure, including folders for `Core`, `InteractiveUtils`, `PlotThemes`, `Plots`, and a `Julia REPL`. The main editor shows a Julia script with the following code:

```
9     end
10     return maxiter
11 end mandel (generic function with 1 method)
12
13 for i in 1:10
14     println(i)
15 end
16
17 map
```

A tooltip for the `map` function is visible, providing the following information:

- `map(f, c...) -> collection`
- Transform collection `c` by applying `f` to each element. For multiple collection arguments, apply `f` elementwise.
- See also: `mapslices`
- Examples**
- `julia> map(x -> x * 2, [1, 2, 3])`
- `3-element Array{Int64,1}:`
- `2`

The bottom panel shows the Julia REPL with the output of the script:

```
4
5
6
7
8
9
10
julia>
```