

Getting Started

Laptop/DelftBlue

Course repository

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

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
```

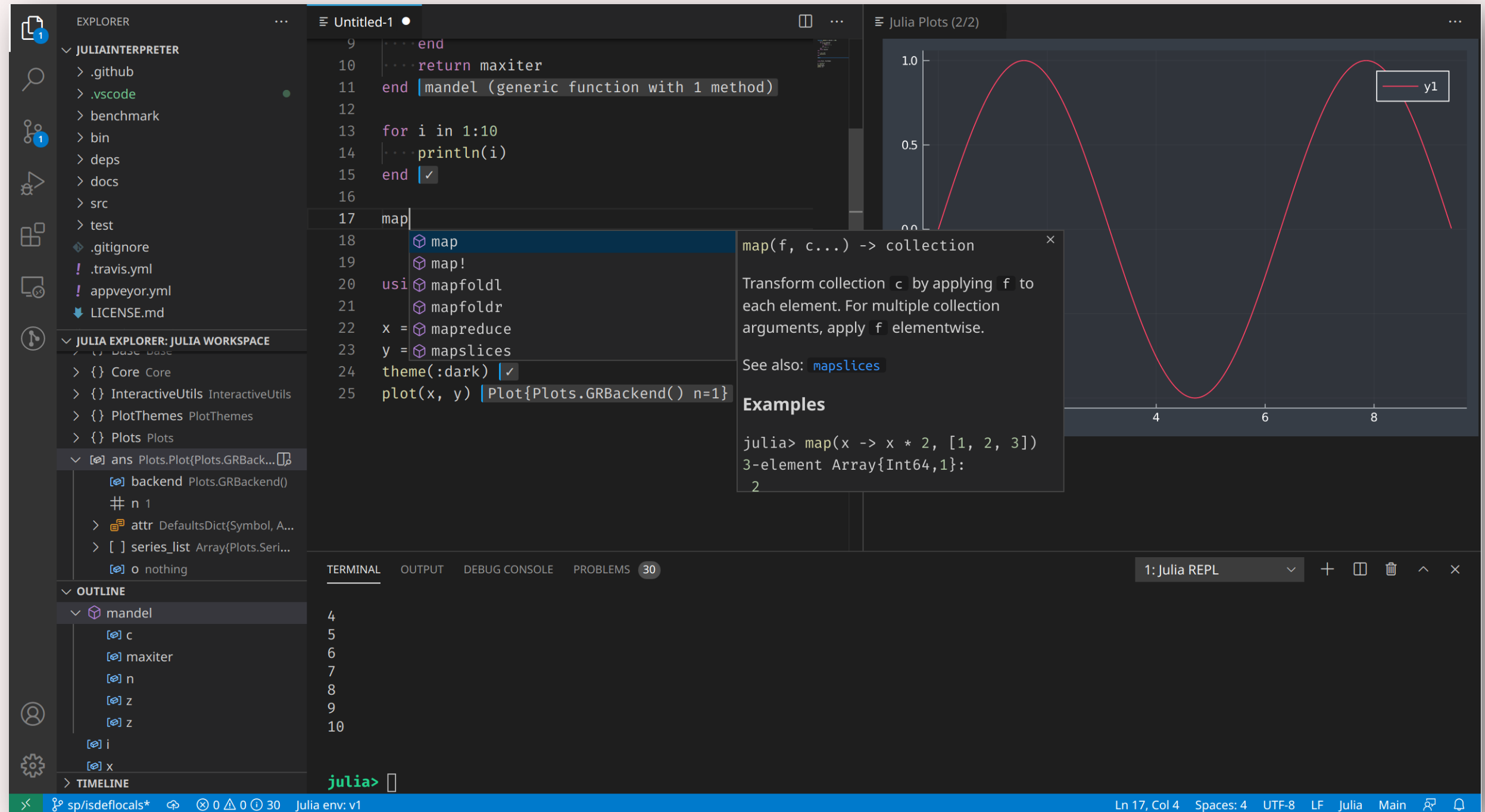
```
module load juliahpc
```

```
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 installed. The Explorer on the left shows the project structure, including the Julia workspace and various folders like `Core`, `InteractiveUtils`, `PlotThemes`, `Plots`, and `Plots.Plots.GRBackend()`. The Editor in the center shows a Julia script with a plot function. The Output panel at the bottom shows the Julia REPL output. The Plots panel on the right shows a plot of a sine wave.

```
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
18      map
19      map!
20  use mapfoldl
21      mapfoldr
22  x = mapreduce
23  y = mapslices
24  theme(:dark)
25  plot(x, y) Plot{Plots.GRBackend() n=1}
```

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

1: Julia REPL

Ln 17, Col 4 Spaces: 4 UTF-8 LF Julia Main