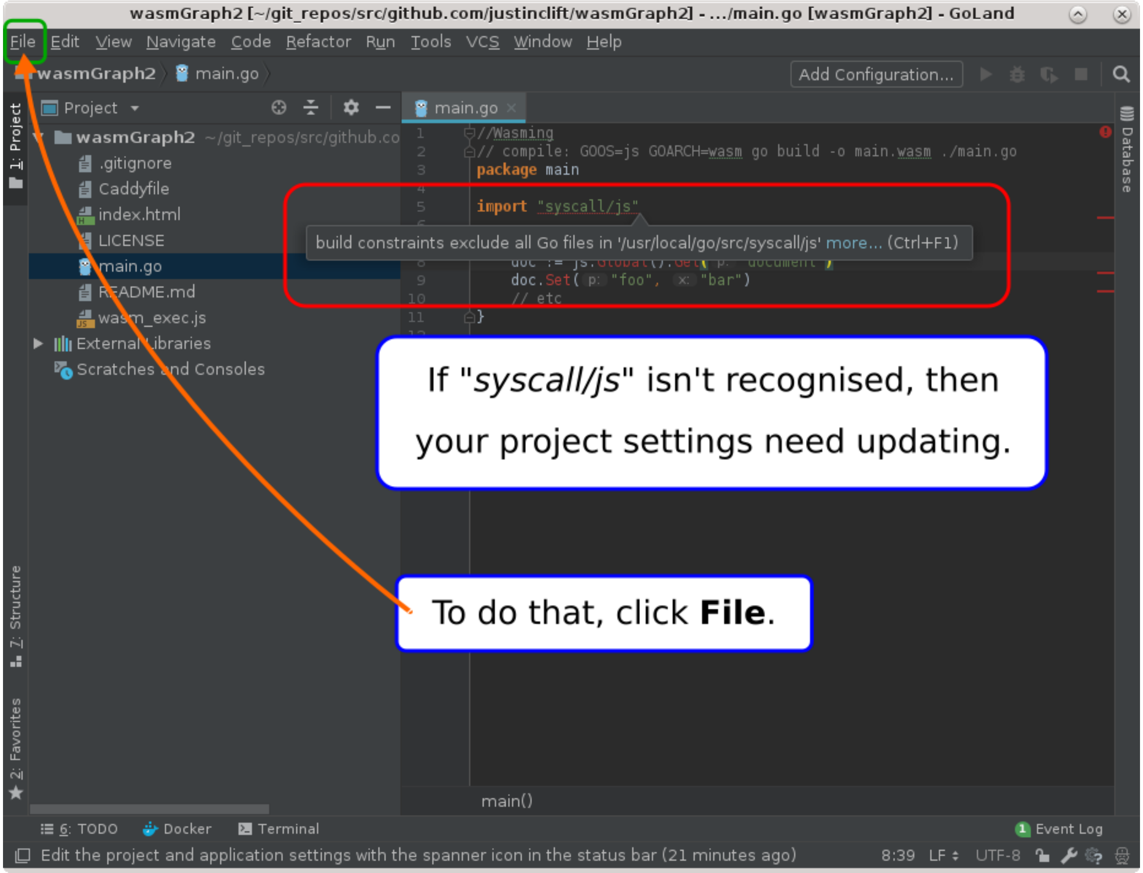
← Back to the main WebAssembly page.

Configuring GoLand for WebAssembly (Wasm) projects

Initial project configuration

When you first open or start a WebAssembly project in GoLand, it won't understand the "*syscall/js*" package.

That's easily fixable, by changing the **GOOS** and **GOARCH** values in the project settings, as per the screenshots below.



wasmGraph2 [~/git_repos/src/github.com/justinclift/wasmGraph2] - .../main.go [wasmGraph2] - GoLand

```
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"

    doc := js.Global().Get("document")
    doc.Set("foo", "bar")
    // etc
}
```

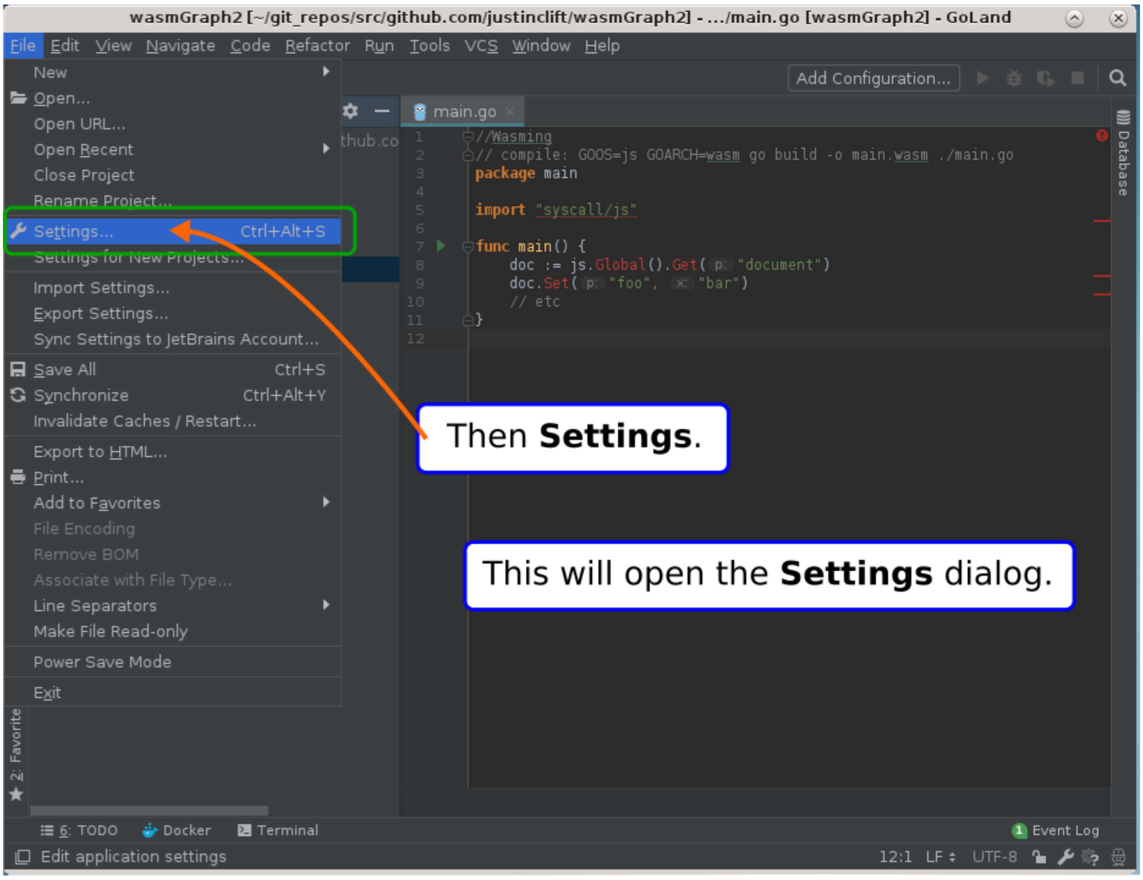
build constraints exclude all Go files in '/usr/local/go/src/syscall/js' more... (Ctrl+F1)

If "*syscall/js*" isn't recognised, then your project settings need updating.

To do that, click **File**.

Edit the project and application settings with the spanner icon in the status bar (21 minutes ago)

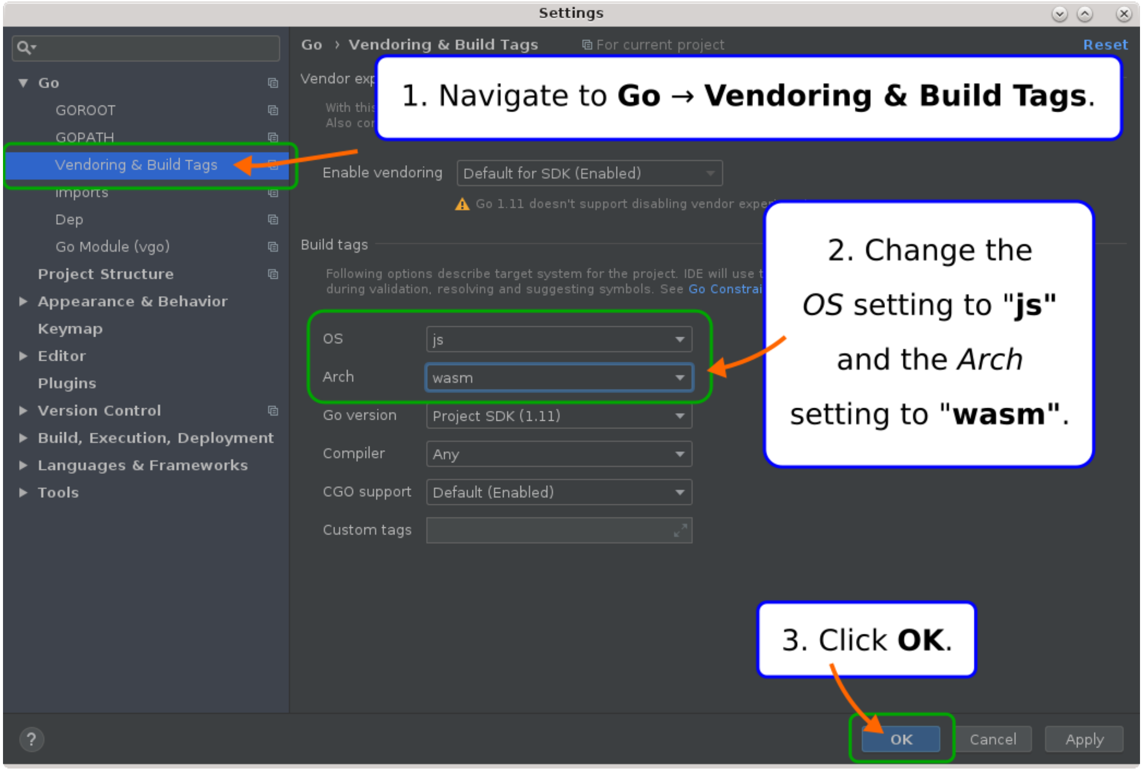
Note - The screenshot below shows how to access **Settings** on a Linux desktop. If you're using macOS, you'll probably need to access them through the standard macOS **Preferences** option.



```go
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"

func main() {
    doc := js.Global().Get("document")
    doc.Set("foo", "bar")
    // etc
}
```

Then **Settings**.

This will open the **Settings** dialog.



1. Navigate to **Go → Vendoring & Build Tags**.

Enable vendoring: Default for SDK (Enabled)

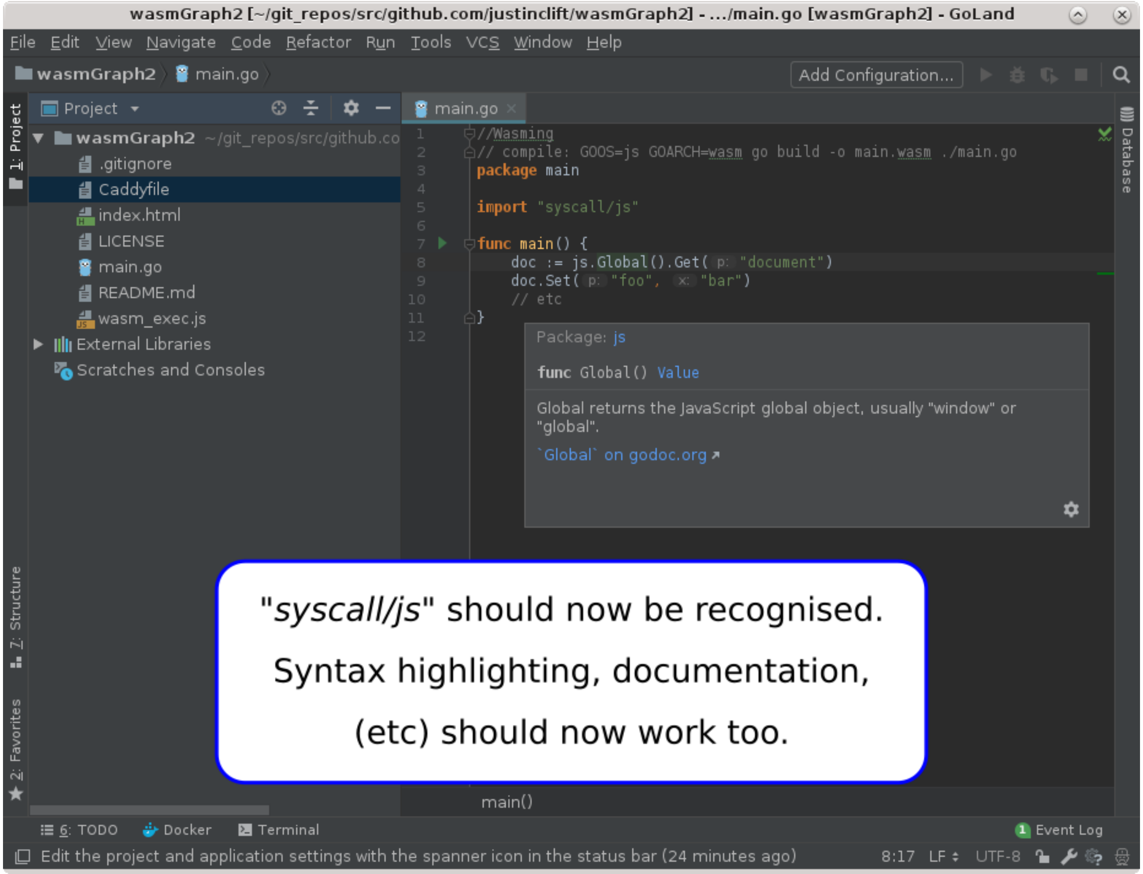
Go 1.11 doesn't support disabling vendor expe

Following options describe target system for the project. IDE will use t during validation, resolving and suggesting symbols. See Go Constrai

OS	js
Arch	wasm
Go version	Project SDK (1.11)
Compiler	Any
CGO support	Default (Enabled)
Custom tags	

2. Change the *OS* setting to "**js**" and the *Arch* setting to "**wasm**".

3. Click **OK**.



```
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"

func main() {
    doc := js.Global().Get( p: "document")
    doc.Set( p: "foo", x: "bar")
    // etc
}
```

Package: js

`func Global() Value`

Global returns the JavaScript global object, usually "window" or "global".

`Global` on godoc.org ↗

"*syscall/js*" should now be recognised.

Syntax highlighting, documentation,

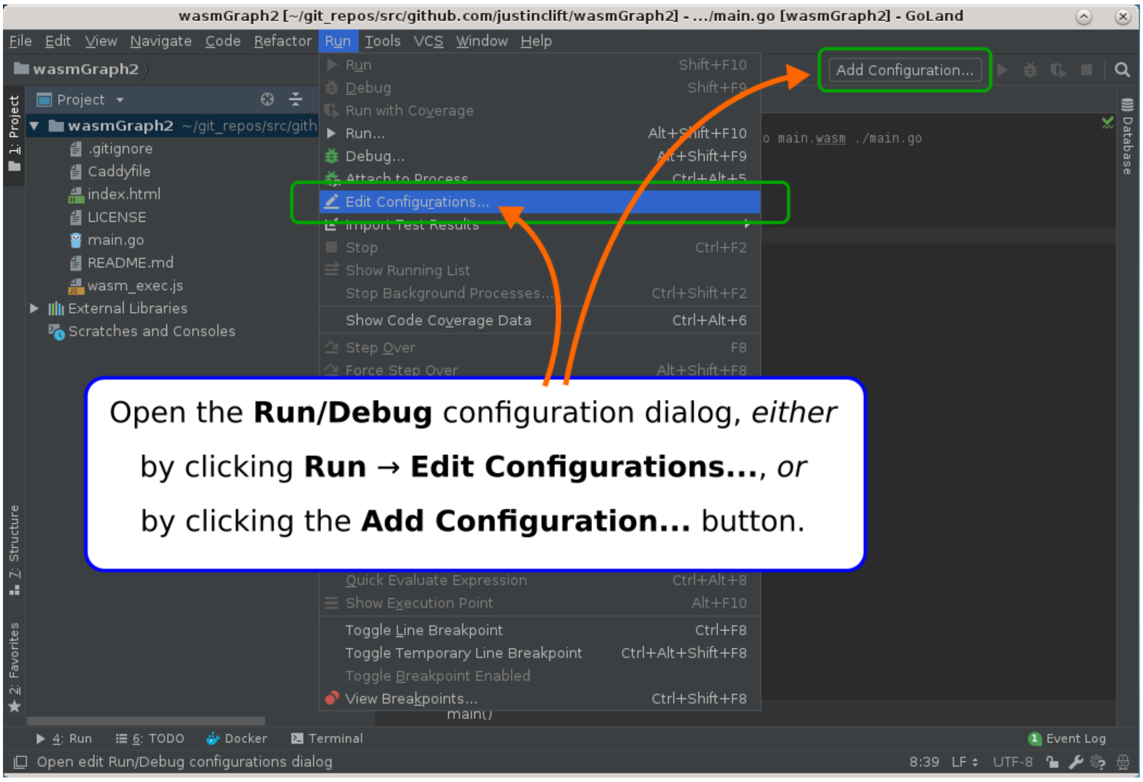
(etc) should now work too.

Edit the project and application settings with the spanner icon in the status bar (24 minutes ago)

Configuring Run/Debug settings

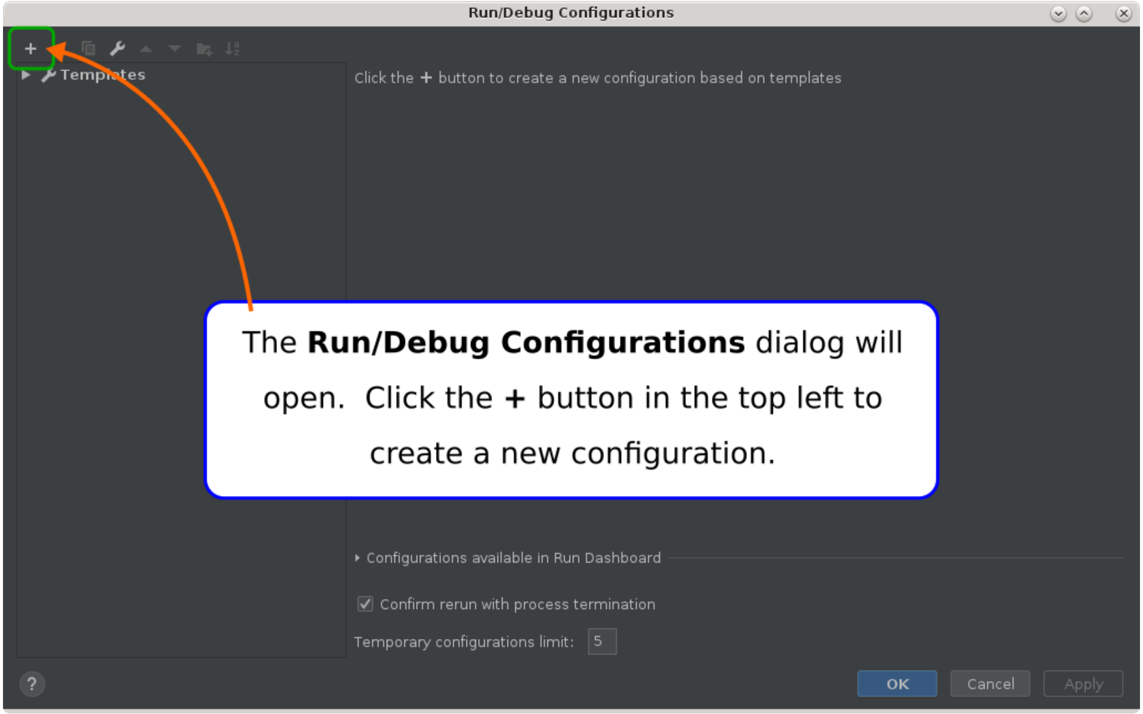
With the initial project settings changed, you'll probably want to configure the Run/Debug settings next.

That will let you recompile the .wasm file by just launching `Run` (`Shift`+`F10` on Linux).



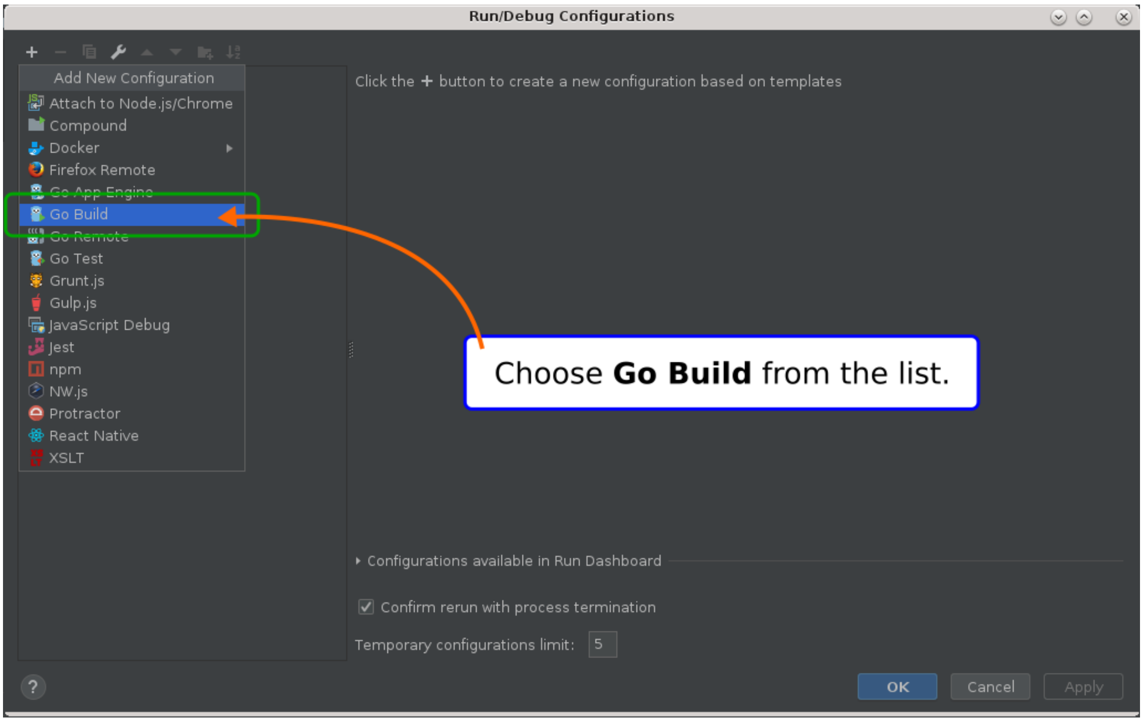
Open the **Run/Debug** configuration dialog, *either* by clicking **Run → Edit Configurations…**, *or* by clicking the **Add Configuration…** button.

wasmGraph2 [~/git_repos/src/github.com/justinclift/wasmGraph2] - .../main.go [wasmGraph2] - GoLand
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Add Configuration...
wasmGraph2
Project
wasmGraph2 ~/git_repos/src/gith
.gitignore
Caddyfile
index.html
LICENSE
main.go
README.md
wasm_exec.js
External Libraries
Scratches and Consoles
Run Shift+F10
Debug Shift+F9
Run with Coverage
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Process Ctrl+Alt+5
Edit Configurations...
Import Test Results
Stop Ctrl+F2
Show Running List
Stop Background Processes... Ctrl+Shift+F2
Show Code Coverage Data Ctrl+Alt+6
Step Over F8
Force Step Over Alt+Shift+F8
Quick Evaluate Expression Ctrl+Alt+8
Show Execution Point Alt+F10
Toggle Line Breakpoint Ctrl+F8
Toggle Temporary Line Breakpoint Ctrl+Alt+Shift+F8
Toggle Breakpoint Enabled
View Breakpoints... Ctrl+Shift+F8
o main.wasm ./main.go
main()
1: Project
7: Structure
2: Favorites
Database
4: Run
6: TODO
Docker
Terminal
Event Log
Open edit Run/Debug configurations dialog
8:39 LF UTF-8



The **Run/Debug Configurations** dialog will open. Click the **+** button in the top left to create a new configuration.

Run/Debug Configurations
+
Templates
Click the + button to create a new configuration based on templates
Configurations available in Run Dashboard
☑ Confirm rerun with process termination
Temporary configurations limit: 5
?
OK
Cancel
Apply

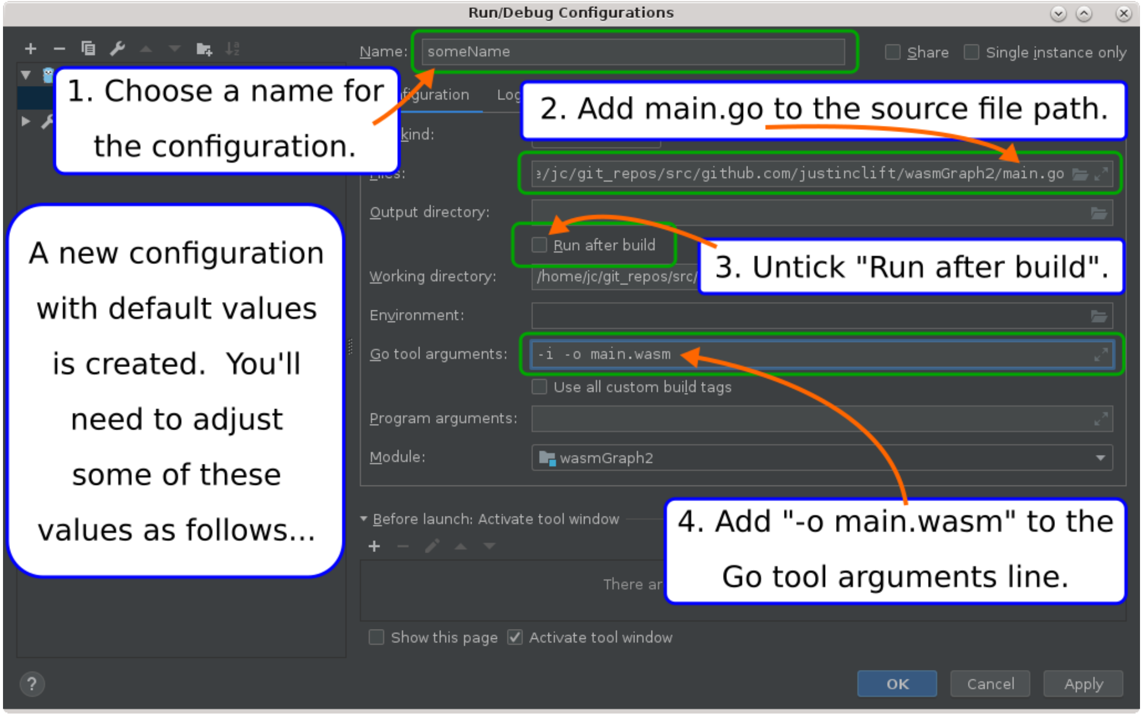


Click the + button to create a new configuration based on templates

Choose **Go Build** from the list.

☑ Confirm rerun with process termination

Temporary configurations limit: 5



Name: someName

□ Share □ Single instance only

1. Choose a name for the configuration.

2. Add main.go to the source file path.

Run kind:

Files: :/jc/git_repos/src/github.com/justinclift/wasmGraph2/main.go

Output directory:

□ Run after build

3. Untick "Run after build".

Working directory: /home/jc/git_repos/src/

Environment:

Go tool arguments: -i -o main.wasm

□ Use all custom build tags

Program arguments:

Module: wasmGraph2

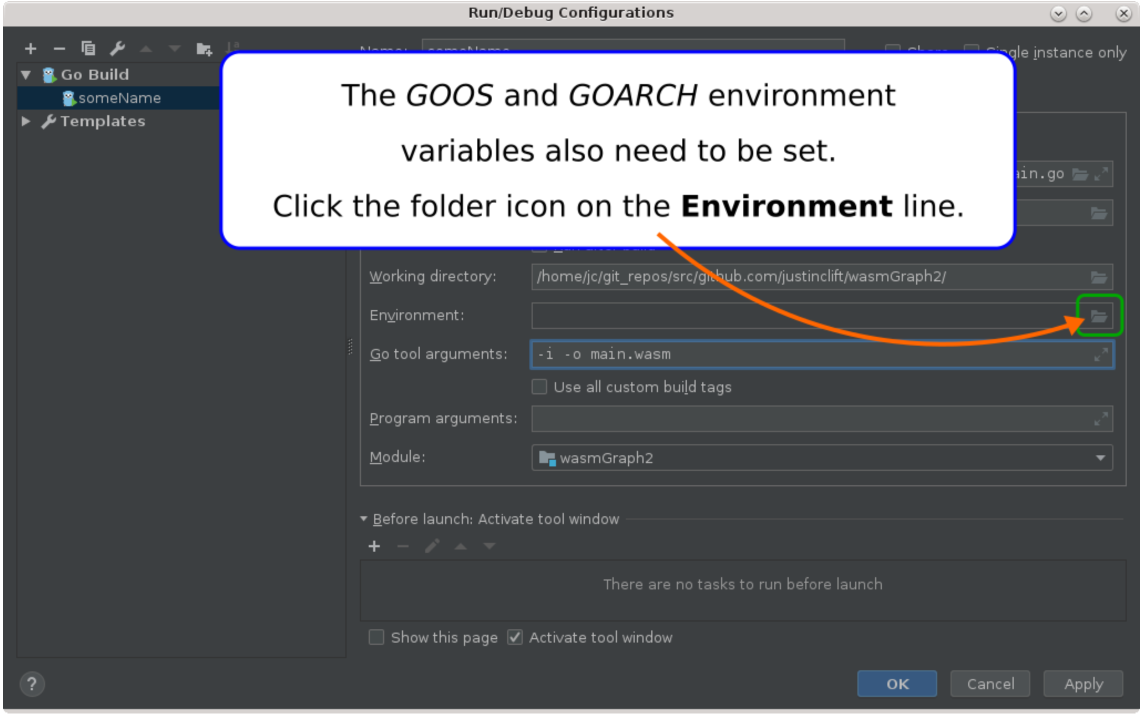
A new configuration with default values is created. You'll need to adjust some of these values as follows...

Before launch: Activate tool window

There ar

4. Add "-o main.wasm" to the Go tool arguments line.

□ Show this page ☑ Activate tool window



Run/Debug Configurations

Working directory: /home/jc/git_repos/src/github.com/justinclift/wasmGraph2/

Environment:

Go tool arguments: -i -o main.wasm

☐ Use all custom build tags

Program arguments:

Module: wasmGraph2

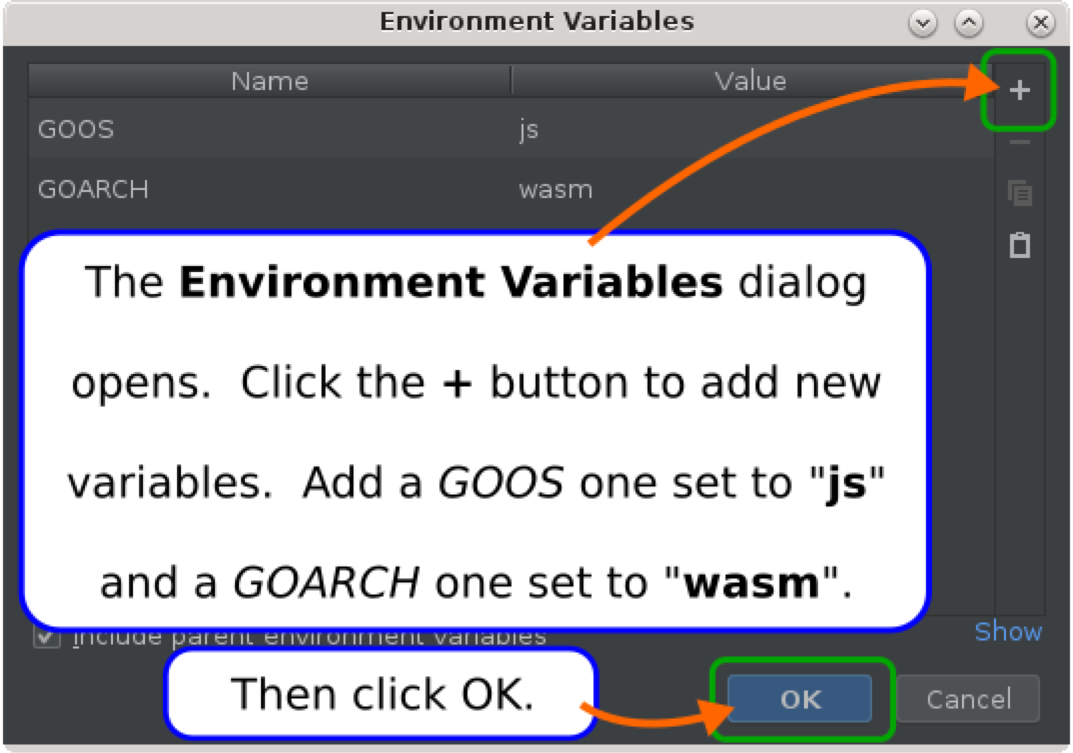
Before launch: Activate tool window

There are no tasks to run before launch

☐ Show this page ☑ Activate tool window

The *GOOS* and *GOARCH* environment variables also need to be set.

Click the folder icon on the **Environment** line.



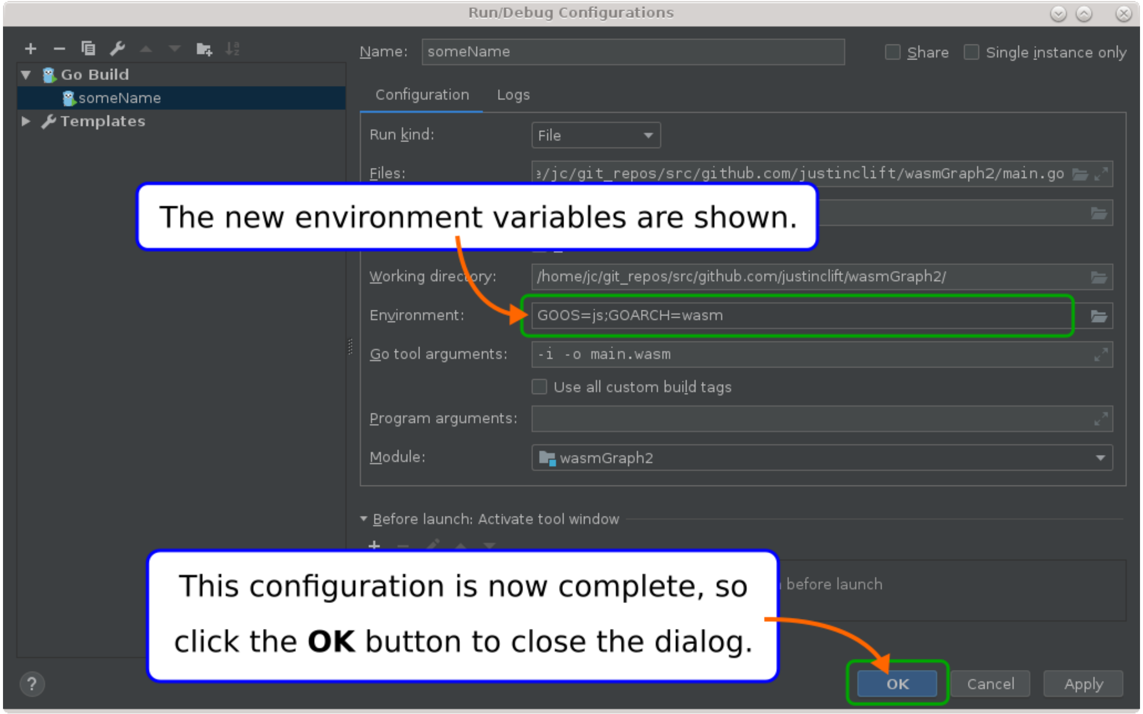
Environment Variables

Name	Value
GOOS	js
GOARCH	wasm

☑ Include parent environment variables

The **Environment Variables** dialog opens. Click the **+** button to add new variables. Add a *GOOS* one set to "**js**" and a *GOARCH* one set to "**wasm**".

Then click OK.



Name: someName ☐ Share ☐ Single instance only

Run kind: File

Files: ı/jc/git_repos/src/github.com/justinclift/wasmGraph2/main.go

Working directory: /home/jc/git_repos/src/github.com/justinclift/wasmGraph2/

Environment: GOOS=js;GOARCH=wasm

Go tool arguments: -i -o main.wasm

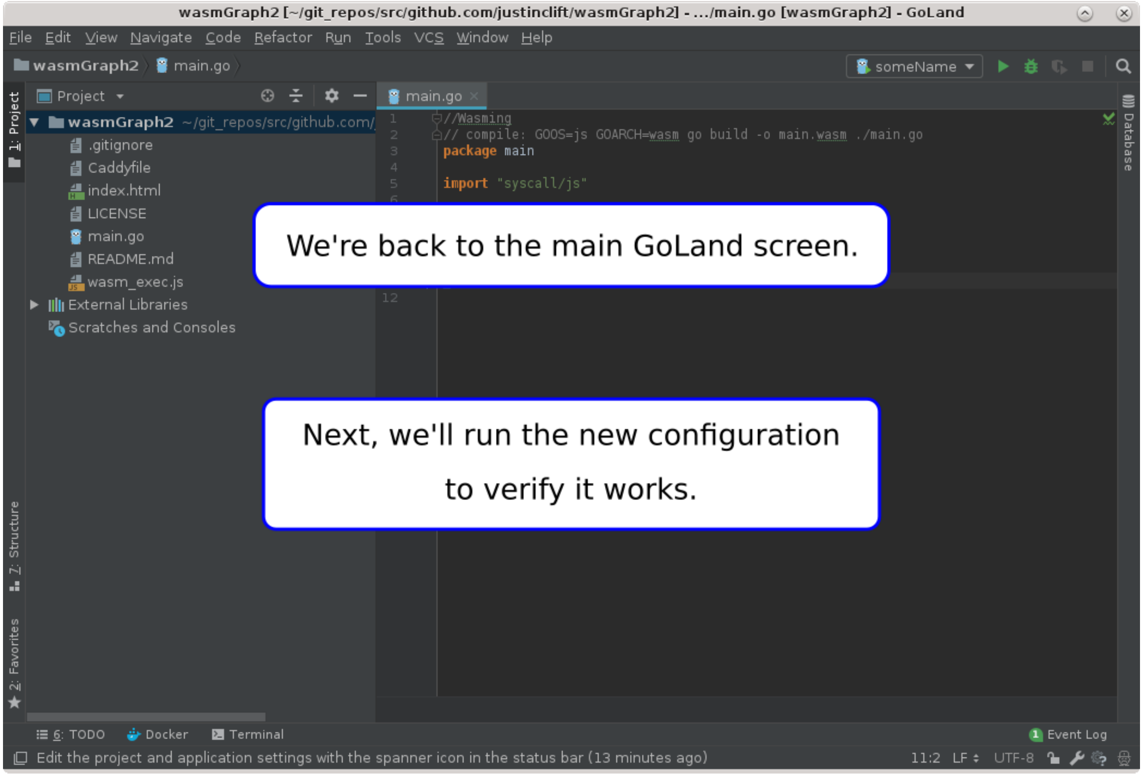
☐ Use all custom build tags

Program arguments:

Module: wasmGraph2

The new environment variables are shown.

This configuration is now complete, so click the **OK** button to close the dialog.

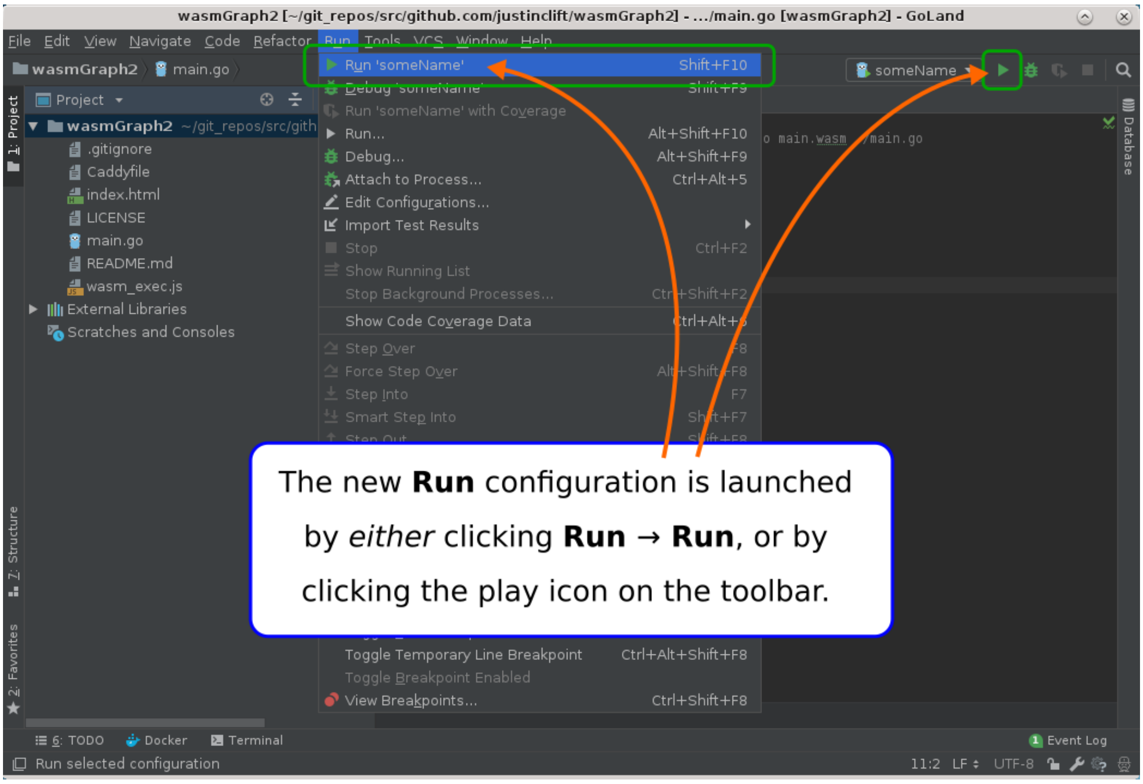


```
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"
```

Edit the project and application settings with the spanner icon in the status bar (13 minutes ago)

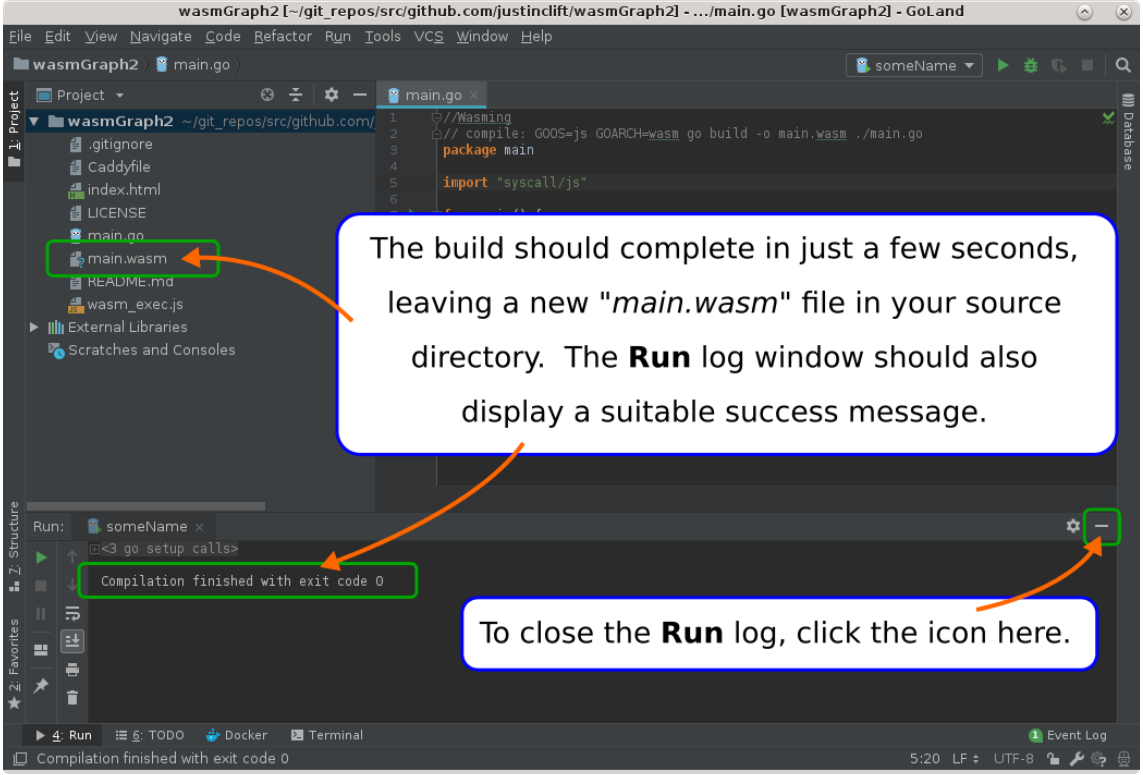
We're back to the main GoLand screen.

Next, we'll run the new configuration to verify it works.



wasmGraph2 [~/git_repos/src/github.com/justinclift/wasmGraph2] - .../main.go [wasmGraph2] - GoLand
File Edit View Navigate Code Refactor Run Tools VCS Window Help
wasmGraph2 main.go
someName
Project
wasmGraph2 ~/git_repos/src/gith
.gitignore
Caddyfile
index.html
LICENSE
main.go
README.md
wasm_exec.js
External Libraries
Scratches and Consoles
Run 'someName' Shift+F10
Debug 'someName' Shift+F9
Run 'someName' with Coverage
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Process... Ctrl+Alt+5
Edit Configurations...
Import Test Results
Stop Ctrl+F2
Show Running List
Stop Background Processes... Ctrl+Shift+F2
Show Code Coverage Data Ctrl+Alt+6
Step Over F8
Force Step Over Alt+Shift+F8
Step Into F7
Smart Step Into Shift+F7
Step Out Shift+F8
Toggle Temporary Line Breakpoint Ctrl+Alt+Shift+F8
Toggle Breakpoint Enabled
View Breakpoints... Ctrl+Shift+F8
6: TODO
Docker
Terminal
Event Log
Run selected configuration
11:2 LF UTF-8
1: Project
7: Structure
2: Favorites
Database

The new **Run** configuration is launched by *either* clicking **Run** → **Run**, or by clicking the play icon on the toolbar.



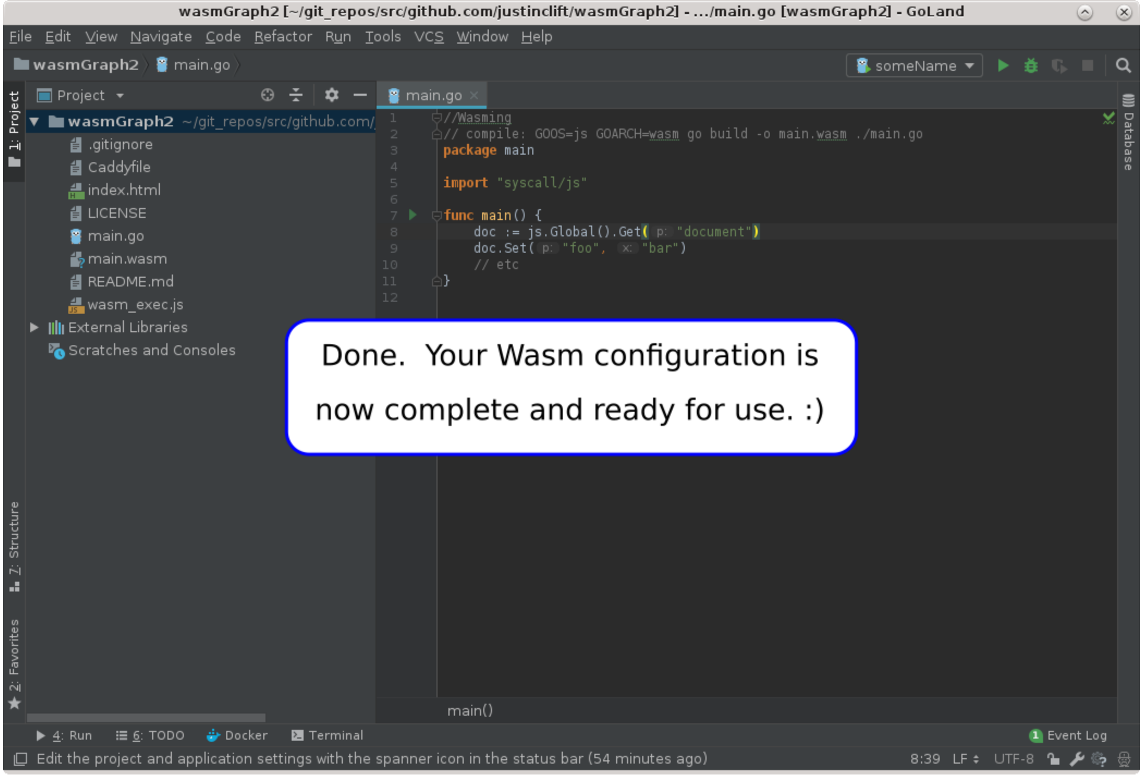
```
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"
```

```
<3 go setup calls>
Compilation finished with exit code 0
```

Compilation finished with exit code 0

The build should complete in just a few seconds, leaving a new "*main.wasm*" file in your source directory. The **Run** log window should also display a suitable success message.

To close the **Run** log, click the icon here.



wasmGraph2 [~/git_repos/src/github.com/justinclift/wasmGraph2] - .../main.go [wasmGraph2] - GoLand

```
//Wasming
// compile: GOOS=js GOARCH=wasm go build -o main.wasm ./main.go
package main

import "syscall/js"

func main() {
    doc := js.Global().Get( p: "document")
    doc.Set( p: "foo",  x: "bar")
    // etc
}
```

Done. Your Wasm configuration is now complete and ready for use. :)

Edit the project and application settings with the spanner icon in the status bar (54 minutes ago)   8:39 LF ÷ UTF-8

Finished, your GoLand setup should now be complete.

← Back to the main WebAssembly page.