

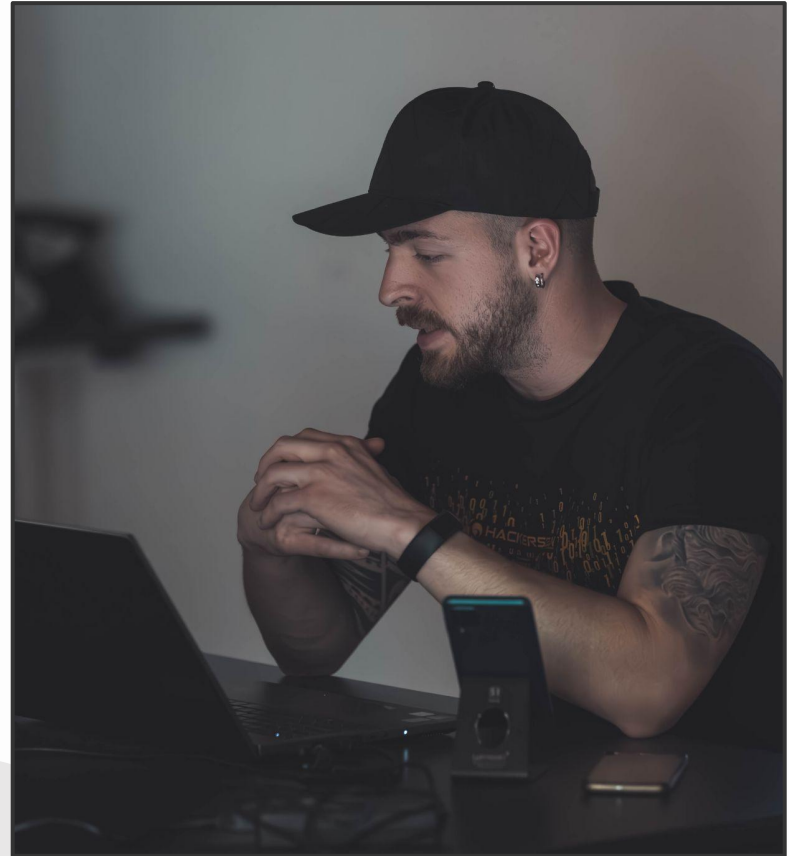
Swiss Knife for Go Debugging with VSCode

Ivan **ossan** Pesenti
Software Engineer
Golab 2024



About Me

- ❑ Name: Ivan Pesenti (AKA **ossan**)
- ❑ Age: 29
- ❑ Country: Italy
- ❑ Company: CrowdStrike
- ❑ Award: Docker Captain
- ❑ Engagements: speaker, mentor,
technical writer, course author
- ❑ Hobby: anime, tattoos, football





My Humble Goals

Stop These

Useless & Time consuming

No need to waste time until we have to debug anything.

Goland

I'm switching to Goland since VSCode doesn't allow me to properly debug my code.

`fmt.Println("")`

Let's add some more troubleshooting log entries.

Mission Players

VSCode

The Integrated Development Environment (**IDE**).

DAP

The Debug Adapter Protocol.

Go

The programming language.

Delve

The debugger.

01

Why Debugging?

Why It's Relevant



Codebase Knowledge

Get more familiar with the codebase you're working with. Follow execution paths to find out functions in the stack trace.



Detect Bugs

Find out where bugs happen and fix them. Allow you to slow down the code execution in specific parts to dig into them.



Inspection

Inspect variables' values or complex objects and how they change while code execution.

02

What Debugging?

All For One, and One For All!



01

**Unit Test
Functions**

02

**Integration
Test Functions**

03

**Command Line
Interface Apps**

04

**Local running
processes**

05

**Remote running
processes**

06

**Any piece of code
in your mind**

03

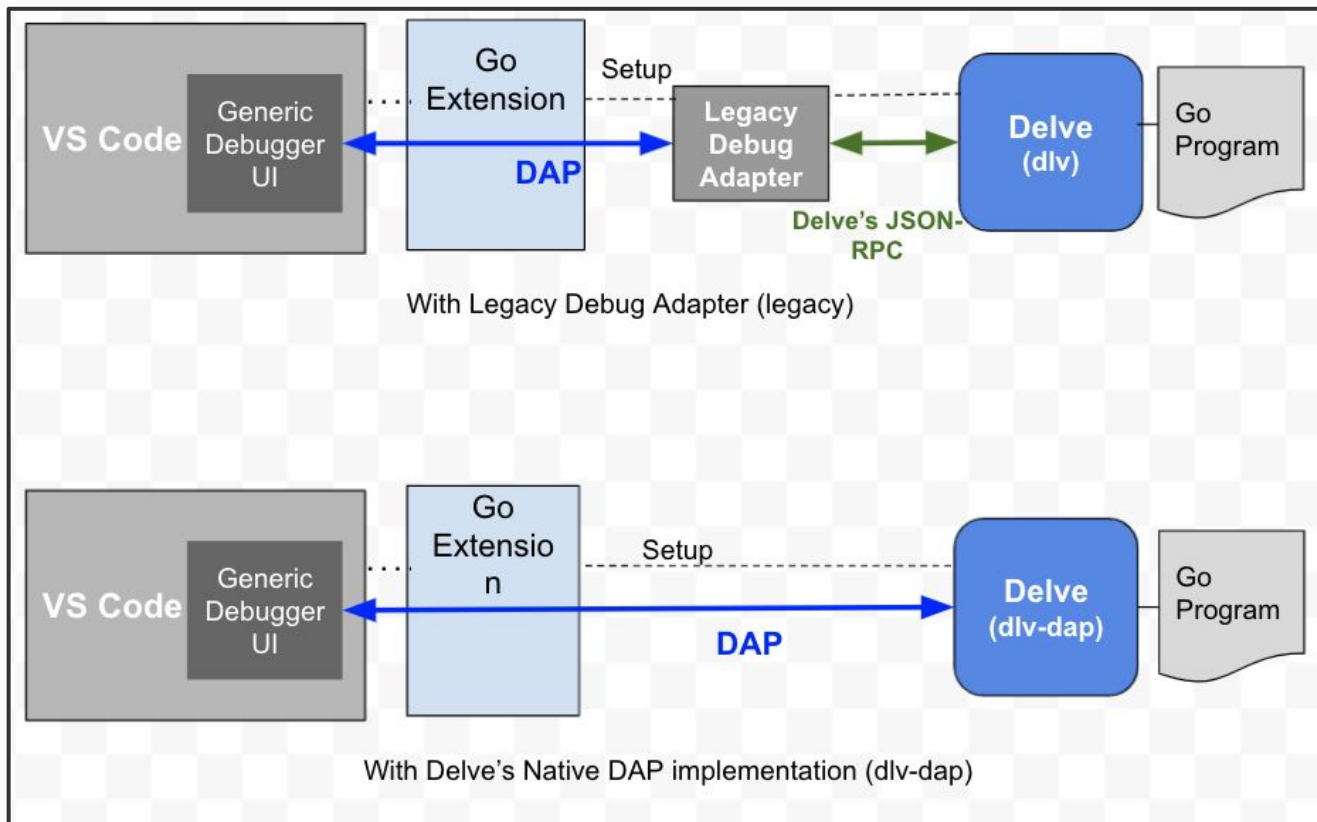
Debug Adapter Protocol

Debug Adapter Protocol



- ❑ Standard on how IDEs/tools communicate with concrete debuggers
- ❑ Utopian that concrete debuggers adapt to DAP (or not in the near future)
- ❑ An intermediate component will manage that
- ❑ Reduce effort to support debugging in new IDEs
- ❑ Allow to implement a generic debugger

Underlying Infrastructure



03

Delve

In a Nutshell

- ❑ The debugger for Go programs (written in Go)
- ❑ Part of the **Go Tools**
- ❑ Listen to commands through a CLI interface
- ❑ Can be also installed (and used) outside of Visual Studio Code

Delve Key Features



Configure

Customize the debug behavior.

Launch

Run and debug your programs.

Attach

Both to a local process or to a remote one.

Debug Session

Handle the debug session.

Breakpoints

Normal, conditional, function, and logpoint.

Inspection

Variables' values and stack trace calls.

04

Visual Studio Code

Why is Worth Using?

- ❑ Free
- ❑ open-source
- ❑ Completely customizable (appearances, extensions, shortcuts, and so on)
- ❑ Fits well for Go programs
- ❑ One-tool-for-all solution

Built-in Variables (not exhaustive)

Variable	Value
<code>\${workspaceFolder}</code>	The path of the folder opened in VS Code
<code>\${file}</code>	The current opened file
<code>\${fileDirName}</code>	The current opened file folder path
<code>\${cwd}</code>	Current Working Directory for the debugger

The *settings.json* File

- ❑ Home for all VS Code settings
- ❑ Every setting in User Preferences (GUI) is read from this file
- ❑ Editor and terminal behavior, themes, extensions, keyboard shortcuts
- ❑ Boost developers productivity and consistency

General Delve Config Setting



go-debug-vsc - settings.json

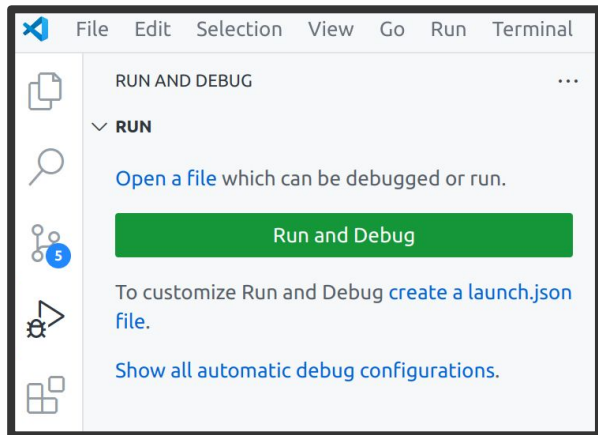
```
1  "go.delveConfig": {  
2    "debugAdapter": "dlv-dap" // or "legacy"  
3  }
```

The *launch.json* File 🤖🤨

- ❑ Selectable in Run View menu
- ❑ Holds a specific configuration to test, debug, launch, etc. a specific piece of code
- ❑ Different arguments for each scenario
- ❑ Prepare the input for the Delve tool

```
go-debug-vsc - launch.json  
  
1  {  
2    "name": "Launch Package",  
3    "type": "go",  
4    "request": "launch",  
5    "mode": "auto",  
6    "program": "${workspaceFolder}"  
7  }
```

The *launch.json* File Setup



```
go-debug-vsc - launch.json

1 {
2   // Use IntelliSense to learn about possible attributes.
3   // Hover to view descriptions of existing attributes.
4   // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5   "version": "0.2.0",
6   "configurations": []
7 }
```

Warm-Up 🐉



Debug StdIn



go-debug-vsc - launch.json

```
1 {
2   "name": "Launch Package",
3   "type": "go",
4   "request": "launch",
5   "mode": "auto",
6   "program": "${workspaceFolder}/main.go",
7   "console": "integratedTerminal"
8 }
```

```
main.go ms-billing  main.go ./ M X
main.go > main
4  func main() {
5  }
6
7  func main() {
8      fmt.Println("Type your name:")
9      var name string
10     fmt.Scan(&name)
11     fmt.Printf("Hi, %s!\n", name)
12 }
13
```

TERMINAL PROBLEMS DEBUG CONSOLE PORTS OUTPUT

```
ossan@personal:~/Projects/go-debug-vsc$ cd /home/ossan/Projects/go-debug-vsc ; /usr/bin/env GOPATH=/home/ossan/go /home/ossan/go/bin/dlv dap --client-addr=:46569 .
Type your name:
Ivan
Hi, Ivan!
█
```

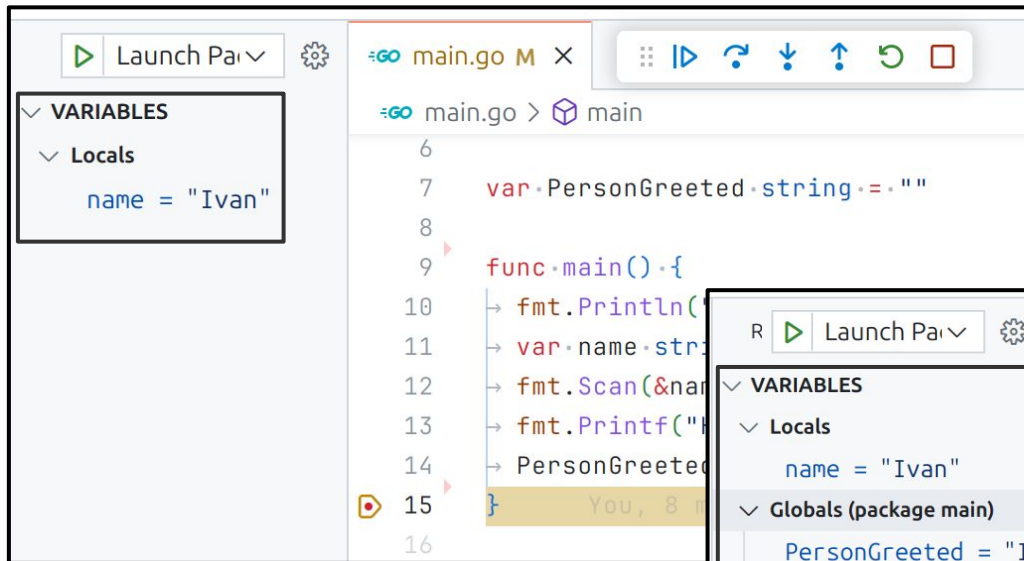

Global Variables



go-debug-vsc - launch.json

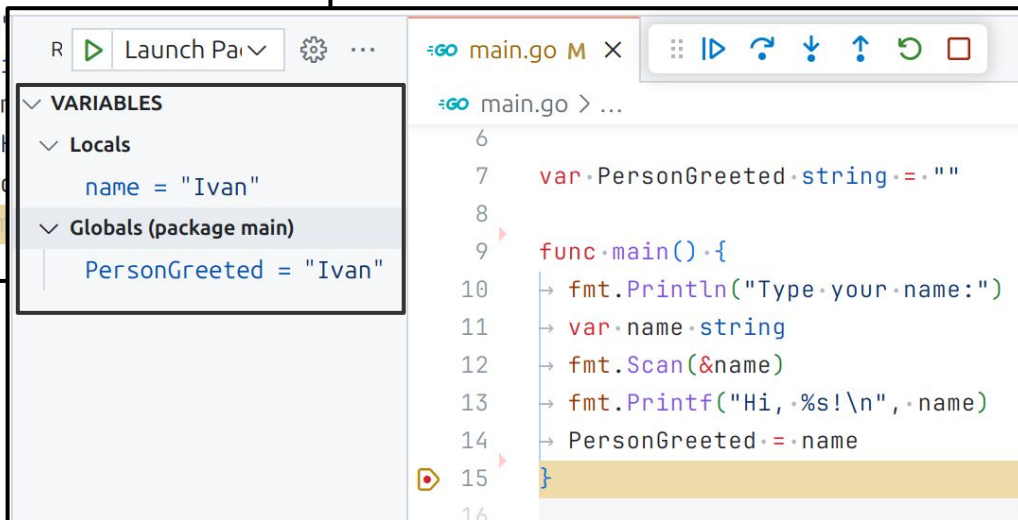
```
1  {  
2    "name": "Launch Package",  
3    "type": "go",  
4    "request": "launch",  
5    "mode": "auto",  
6    "showGlobalVariables": true,  
7    "program": "${workspaceFolder}/main.go",  
8    "console": "integratedTerminal"  
9  }
```

In Action



VS Code editor showing a Go program. The variable inspector on the left highlights the 'Locals' section, showing the variable `name` with the value `"Ivan"`.

```
main.go M X
main.go > main
6
7   var PersonGreeted string = ""
8
9   func main() {
10      fmt.Println("Type your name:")
11      var name string
12      fmt.Scan(&name)
13      fmt.Printf("Hi, %s!\n", name)
14      PersonGreeted = name
15   }
16
```



VS Code editor showing the same Go program. The variable inspector on the left highlights the 'Globals (package main)' section, showing the variable `PersonGreeted` with the value `"Ivan"`.

```
main.go M X
main.go > ...
6
7   var PersonGreeted string = ""
8
9   func main() {
10      fmt.Println("Type your name:")
11      var name string
12      fmt.Scan(&name)
13      fmt.Printf("Hi, %s!\n", name)
14      PersonGreeted = name
15   }
16
```

Debug Console REPL



```
go-debug-vsc - main.go

1 // func to evaluate in "Debug Console REPL"
2 func getEvenNumbers(max int) (res []int) {
3     for i := range max {
4         if i%2 == 0 {
5             res = append(res, i)
6         }
7     }
8     return
9 }

10
11 func main() {
12     fmt.Println("Type your name:")
13     var name string
14     fmt.Scan(&name)
15     fmt.Printf("Hi, %s!\n", name)
16     PersonGreeted = name
17     getEvenNumbers(3) // this call is mandatory
18 }
```

```
18
19 func main() {
20     fmt.Println("Type your name:")
21     var name string
22     fmt.Scan(&name)
23     fmt.Printf("Hi, %s!\n", name)
24     PersonGreeted = name
25     getEvenNumbers(3) // this call is mandatory
26 }
27
```

TERMINAL PROBLEMS TEST RESULTS PORTS GITLENS OUTPUT DEBUG CONSOLE

```
Type 'dlv help' for list of commands.
call getEvenNumbers(8)
✓ []int len: 4, cap: 4, [0,2,4,6]
  ✓ res: []int len: 4, cap: 4, [0,2,4,6]
    [0]: 0
    [1]: 2
    [2]: 4
    [3]: 6
call getEvenNumbers(3)
✓ []int len: 2, cap: 2, [0,2]
  ✓ res: []int len: 2, cap: 2, [0,2]
    [0]: 0
    [1]: 2
```

Well... Let's Practice!



A Sample Unit Test



go-debug-vsc - invoice_test.go

```
1 func TestCalculateAmount(t *testing.T) {
2     // Arrange
3     testSuite := []struct {
4         // ... omitted for brevity
5     }{
6         // ... omitted for brevity
7         {
8             name:          "HappyPath",
9             itemsPurchased: map[string]int{"mobile phone": 2},
10            itemsInStock:   map[string]int{"mobile phone": 40, "TV": 20},
11            itemsPrices:    map[string]float64{"mobile phone": 350.00, "TV": 500.00},
12            amount:         700.00,
13            wantErr:        nil,
14        },
15    }
16    for _, tt := range testSuite {
17        t.Run(tt.name, func(t *testing.T) {
18            // Act
19            amount, err := payment.CalculateAmount(tt.itemsPurchased, tt.itemsInStock, tt.itemsPrices)
20            // Assert
21            assert.Equal(t, tt.amount, amount)
22            assert.Equal(t, tt.wantErr, err)
23        })
24    }
25 }
```

Conditionals Breakpoints ?

```
43 → {
44 →   name: ..... "PriceMustBePositive",
45 →   itemsPurchased: ..map[string]int{"mobile phone": 1},
46 →   itemsInStock: ...map[string]int{"mobile phone": 40, "TV": 20},
47 →   itemsPrices: ....map[string]float64{"mobile phone": -5.00, "TV": 500.00},
48 →   amount: ..... 0.00,
49 →   wantErr: ..... errors.New("price cannot be zero or less"),
50 → },
51 → {
52 →   name: ..... "HappyPath",
53 →   itemsPurchased: ..map[string]int{"mobile phone": 2},
54 →   itemsInStock: ...map[string]int{"mobile phone": 40, "TV": 20},
55 →   itemsPrices: ....map[string]float64{"mobile phone": 350.00, "TV": 500.00},
56 →   amount: ..... 700.00,
57 →   wantErr: ..... nil,
58 → },
59 → }
60 → for _, tt := range testSuite {
61 →   t.Run(tt.name, func(t *testing.T) {
62 →     // Act
63 →     amount, err := payment.CalculateAmount(tt.itemsPurchased, tt.itemsInStock, tt.
64 →     itemsPrices)
65 →     // Assert
66 →     assert.Equal(t, tt.amount, amount)
67 →     assert.Equal(t, tt.wantErr, err)
68 →   })
69 → }
```

Expression tt.name == "HappyPath"

You, 3 weeks ago • debug unit test

Debug With *launch.json*



go-debug-vsc - launch.json

```
1  {
2    "name": "debug-unit-test",
3    "type": "go",
4    "request": "launch",
5    "mode": "test",
6    "program": "${fileDirname}",
7    "args": [
8      "-test.v",
9      "-test.run",
10     "TestCalculateAmount" // or "TestCalculateAmount/HappyPath"
11  ]
12 }
```

The Integration Test Suite



go-debug-vsc - sheet_test.go

```
1 func (s *InventoryTestSuite) TestAddItemGivesErrWhenQtyLessThanZero() {
2     err := inventory.AddItem(s.gormDbClient, &inventory.Item{Name: "mobile phone", Quantity: -4})
3     require.Error(s.T(), err)
4 }
5
6 func (s *InventoryTestSuite) TestAddItemGivesErrWhenNameIsEmpty() {
7     err := inventory.AddItem(s.gormDbClient, &inventory.Item{Name: "", Quantity: 4})
8     require.Error(s.T(), err)
9 }
10
11 func (s *InventoryTestSuite) TestAddItemHappyPath() {
12     item := inventory.Item{Name: "mobile phone", Quantity: 4}
13     err := inventory.AddItem(s.gormDbClient, &item)
14     require.NoError(s.T(), err)
15     assert.NotZero(s.T(), item.ID)
16 }
```




It !



go-debug-vsc - launch.json

```
1  {
2    "name": "debug-integration-test",
3    "type": "go",
4    "request": "launch",
5    "mode": "test",
6    "program": "${fileDirname}",
7    "buildFlags": [
8      "-tags=integration"
9    ],
10   "args": [
11     "-test.v",
12     "-test.run",
13     "TestInventorySuite",
14     "-testify.m", // for a specific test
15     "TestAddItemHappyPath" // for a specific test
16   ]
17 }
```

Attach to Local Process



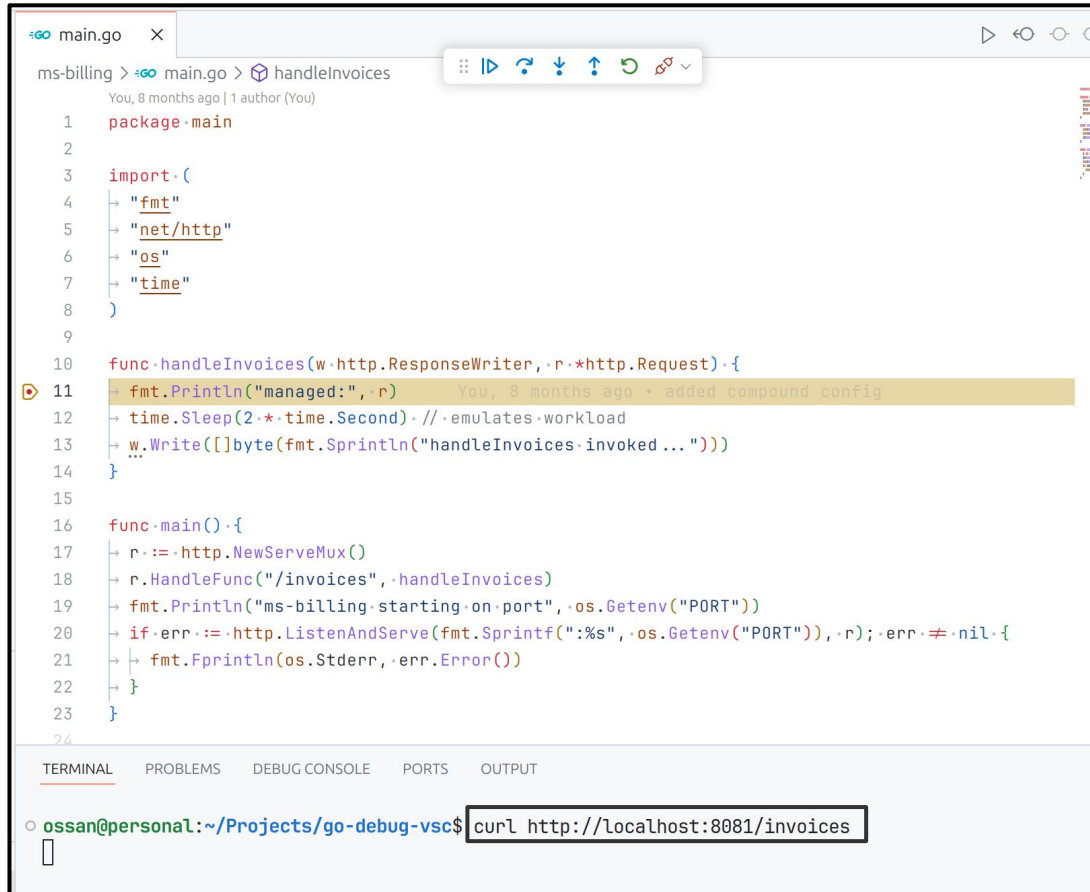
go-debug-vsc - build.sh

```
1  #!/bin/bash
2
3  cd /home/ossan/Projects/go-debug-vsc/ms-billing
4  echo 0 | tee /proc/sys/kernel/yama/ptrace_scope
5  go build -modfile=../go.mod -gcflags=all="-N -l"
6  PORT=8081 ./ms-billing
```



go-debug-vsc - launch.json

```
1  {
2    "name": "Attach Local Proc",
3    "type": "go",
4    "request": "attach",
5    "mode": "local",
6    "processId": 0 // selected via the Command Palette
7  }
```



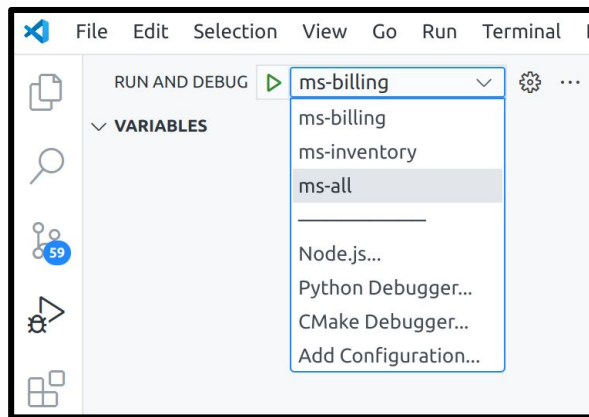
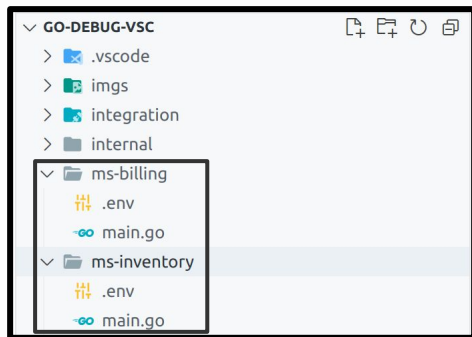
The image shows a VS Code editor window with a Go file named `main.go`. The code defines a package `main` and imports `fmt`, `net/http`, `os`, and `time`. It includes a `handleInvoices` function that prints a message, sleeps for 2 seconds, and writes a response. The `main` function sets up an HTTP server on port 8081, handling requests to `/invoices`. A terminal window at the bottom shows the command `curl http://localhost:8081/invoices` being executed.

```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6     "os"
7     "time"
8 )
9
10 func handleInvoices(w http.ResponseWriter, r *http.Request) {
11     fmt.Println("managed:", r)
12     time.Sleep(2 * time.Second) // emulates workload
13     w.Write([]byte(fmt.Sprintf("handleInvoices invoked ...")))
14 }
15
16 func main() {
17     r := http.NewServeMux()
18     r.HandleFunc("/invoices", handleInvoices)
19     fmt.Println("ms-billing starting on port", os.Getenv("PORT"))
20     if err := http.ListenAndServe(fmt.Sprintf(":%s", os.Getenv("PORT")), r); err != nil {
21         fmt.Fprintln(os.Stderr, err.Error())
22     }
23 }
24
```

Terminal output:

```
ossan@personal:~/Projects/go-debug-vsc$ curl http://localhost:8081/invoices
```

Compound Configuration



```
go-debug-vsc - launch.json
1 {
2   "configurations": [
3     {
4       "name": "ms-billing",
5       "type": "go",
6       "request": "launch",
7       "mode": "auto",
8       "program": "${workspaceFolder}/ms-billing/main.go",
9       "envFile": "${workspaceFolder}/ms-billing/.env",
10      "env": {
11        "PORT": "8081" // take precedence over the env file
12      }
13    },
14    {
15      "name": "ms-inventory",
16      // ... omitted for brevity
17    }
18  ],
19  "compounds": [
20    {
21      "name": "ms-all",
22      "configurations": [
23        "ms-billing",
24        "ms-inventory"
25      ],
26      "stopAll": true
27    }
28  ]
29 }
```

The image shows a VS Code editor window with a Go file named `main.go`. The code defines a package `main` with imports for `fmt`, `net/http`, `os`, and `time`. It includes a `handleInvoices` function that prints a message, sleeps for 2 seconds, and writes the message. The `main` function sets up an HTTP server on port 8081 and listens for requests. A comment on line 11 indicates that the sleep function is used to emulate a workload.

```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6     "os"
7     "time"
8 )
9
10 func handleInvoices(w http.ResponseWriter, r *http.Request) {
11     fmt.Println("managed:", r)
12     time.Sleep(2 * time.Second) // emulates workload
13     w.Write([]byte(fmt.Sprintf("handleInvoices invoked ...")))
14 }
15
16 func main() {
17     r := http.NewServeMux()
18     r.HandleFunc("/invoices", handleInvoices)
19     fmt.Println("ms-billing starting on port", os.Getenv("PORT"))
20     if err := http.ListenAndServe(fmt.Sprintf(":%s", os.Getenv("PORT")), r); err != nil {
21         fmt.Fprintln(os.Stderr, err.Error())
22     }
23 }
24
```

The terminal at the bottom shows the output of the `dap` server, indicating it is listening on port 8081.

```
Starting: /home/ossan/go/bin/dlv dap --listen=127.0.0.1:34161 --log-dest=3 from /media/ossan/Volume/conferer
DAP server listening at: 127.0.0.1:34161
Type 'dlv help' for list of commands.
ms-billing starting on port 8081
```

The Final Challenge



Docker Container



go-debug-vsc - Dockerfile.debug

```
1 FROM golang
2
3 EXPOSE 8080 2345
4
5 WORKDIR /app
6
7 COPY main.go ./
8
9 RUN CGO_ENABLED=0 go install -ldflags "-s -w -extldflags '-static'" \
10 && github.com/go-delve/delve/cmd/dlv@latest
11
12 ENV G0111MODULE=off
13 RUN CGO_ENABLED=0 go build -gcflags "all=-N -l" -o greeting-ms .
14
15 CMD [ "/go/bin/dlv", "--listen=:2345", "--headless=true", "--log=true", \
16 && "--accept-multiclient", "--api-version=2", "exec", "/app/greeting-ms" ]
```

Let's Setup



go-debug-vsc - run.sh

```
1  #!/bin/bash
2
3  docker build . --tag greeting-ms --file Dockerfile.debug
4  docker run -p 8080:8080 -p 2345:2345 greeting-ms
```



go-debug-vsc - launch.json

```
1  {
2    "name": "debug-docker-container",
3    "type": "go",
4    "request": "attach",
5    "debugAdapter": "legacy",
6    "mode": "remote",
7    "port": 2345,
8    "host": "127.0.0.1",
9    "trace": "info",
10   "showLog": true
11 }
```



```

9  func.main(){
10 → mux := http.NewServeMux()
11 → mux.HandleFunc("/greetings", func(w http.ResponseWriter, r *http.Request){
12 → → name := r.URL.Query().Get("name")
13 → → if name == "" {
14 → → | name := "unknown"
15 → → }
16 → → hostname, err := os.Hostname()
17 → → if err != nil {
18 → → | w.WriteHeader(http.StatusInternalServerError)
19 → → | w.Write([]byte(err.Error()))
20 → → return
21 → → }
22 → → w.Write([]byte(fmt.Sprintf("%q greets %q\n", hostname, name)))
23 → → })
24 → if err := http.ListenAndServe(":8080", mux); err != nil {
25 → | fmt.Println(err.Error())
26 → | os.Exit(1)
27 → }
28 }
29

```

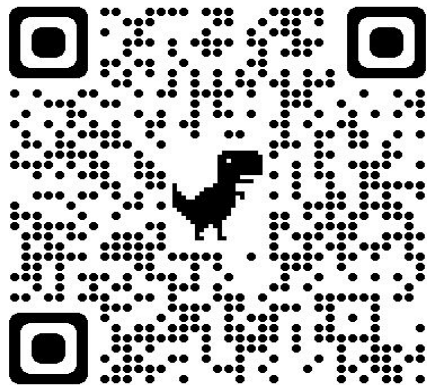
TERMINAL
 PROBLEMS
 DEBUG CONSOLE
 PORTS
 OUTPUT

o ossan@personal: ~/Projects/go-debug-vsc\$ curl http://localhost:8080/greetings?name=ossan



05

QR code and Link to the repository



t.ly/pDNHV

Thanks



ivan.pesenti.dev@gmail.com



[ossandev](#)



[ossan](#)



[ossan#2971](#)



[ossan](#)



[Ivan \(ossan\) Pesenti](#)



[ossan](#)



[ossan-dev](#)



[ossandev](#)