Be Smug, Debug

Why You Should be a Delve Power User

Sam Kamenetz

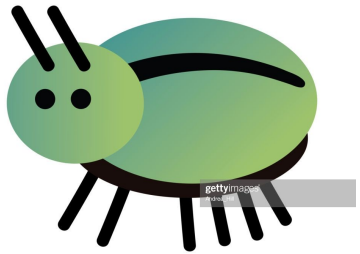
What we are covering

- How to reason about delve and debuggers
- Debug configurations for all types of projects
- Advanced delve and vscode features

Setting up a debugger is a time investment, but a *one time* investment

image: Flaticon.com

Why Debuggers are awesome



Code without a debugger:

```
fmt.Print(r)
fmt.Printf("r: %v+", r)
fmt.Printf("r2: %v", r.Handlers)
fmt.Printf("handlers: %v ", r.Handlers[0])
r.GET("/ping", func(c *gin.Context) {
    fmt.Printf(“here2”)
    c.JSON(200, gin.H{
        "message": some_package.GetMessage(),
    })
})
fmt.Printf("handlers: %+v+\n", r.Handlers.Last())
```

Debugger use cases

- Function that’s used everywhere only breaks when called by certain functions? Inspect the call stack!
- Tests failing? Debug the test code!
- working locally, but not in docker container? Debug the application as part of the container!
- Nil pointer dereference from nowhere? Set breakpoints!

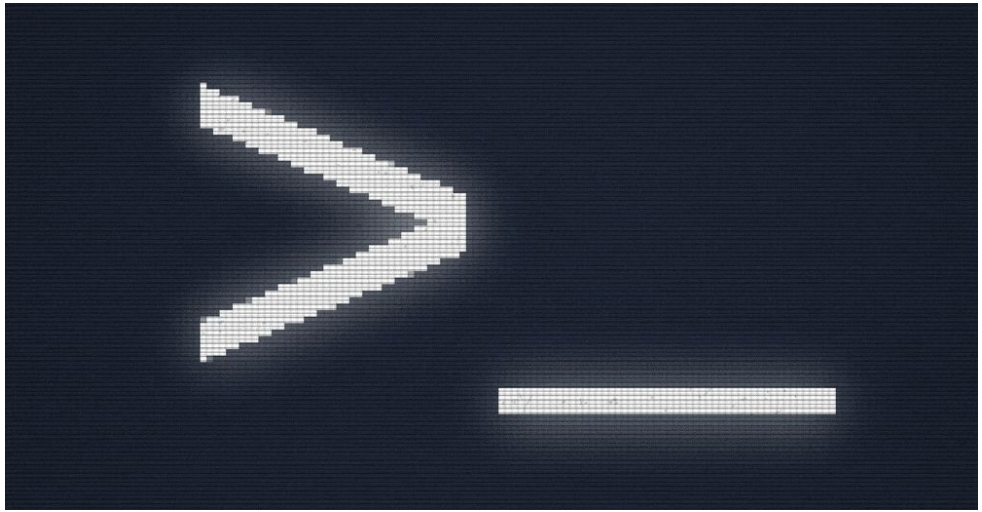
And More! Other Features I Haven’t Used Yet

- rr-debugger integration (allows replaying/rewinding of debugger sessions)
- vs-code’s native DAP support (more on this later)
- Starlark debug scripts

https://github.com/go-delve/delve

Aside on Tooling: Why the Delve’s CLI?

Not the most ergonomic tool, but best basis for understanding the debugger operations



Aside on tooling: why use VSCode over Goland

- Debug options much more configurable
- Easier to edit and share configurations (using launch.json)

Follow along with these examples!

https://github.com/samkam/go-debugging-example

Example 1: running local program

Launch.json attributes, explained.

https://github.com/golang/vscode-go/blob/master/docs/debugging.md#launchjson-attributes

https://code.visualstudio.com/docs/editor/debugging#_launchjson-attributes

If you bookmark anything from this talk, it should be these links!

Example 2: debugging a test

Example 3: tracing a program

See every function called!

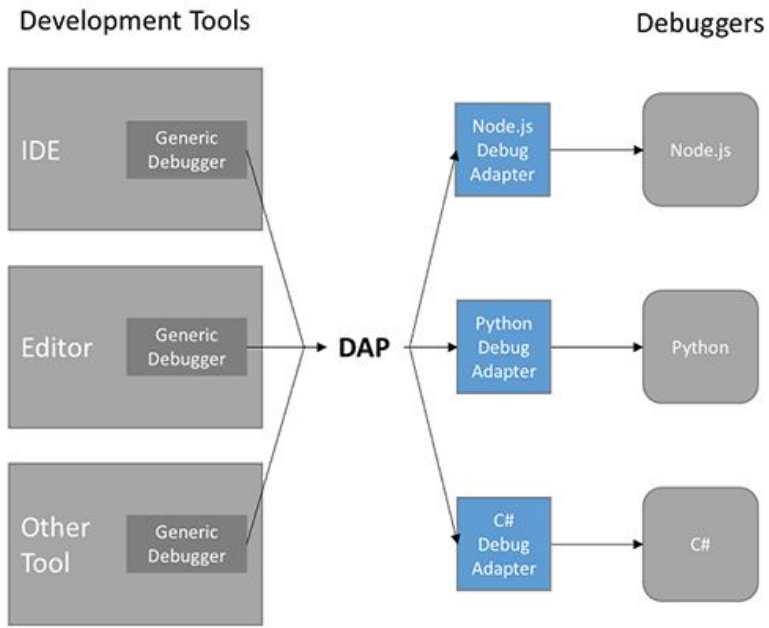
Architecture of delve: client server model

Courtesy of 2018 gophercon speaker
Alessandro Arzilli

https://speakerdeck.com/aarzilli/internal-architecture-of-delve?slide=35

Debugger Adaptor protocol (DAP)

Intermediary between vscode user interface and the actual debugger executable. Specifies a protocol for the purpose



Example 4: debugging an already running process

Example 5: debugging a remote host

Exploring the client server debugger model

Example 6: debugging a Docker container

Breaking down the docker debug command

```
docker run
    --rm
    --expose=40000
    --publish 40000:40000
    --publish 8080:8080
    --security-opt=seccomp:unconfined
    --name debug-example
    debug-example-app
```

```
dlv debug main.go
    --listen=:40000
    --headless=true
    --api-version=2
    -- somevalue
```

Pre-launch tasks

Use custom defined pre-launch tasks to automate set up for debugging (or even regular code execution)

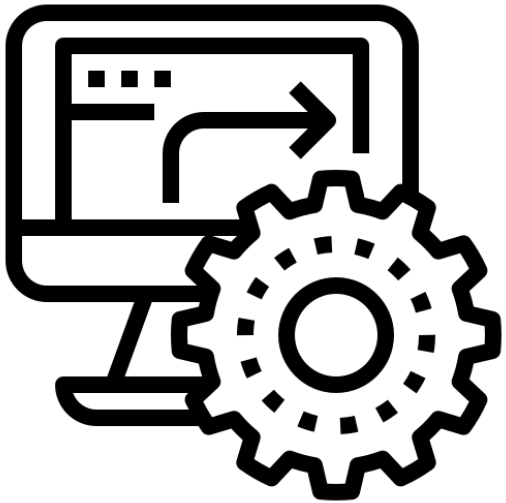


image: Flaticon.com

Summary

- Set up is a one time effort!
- Add your tasks.json and launch.json to your repo so your whole team benefits
- Most important links for configuring debuggers:
  - https://github.com/golang/vscode-go/blob/master/docs/debugging.md#launchjson-attributes
  - https://code.visualstudio.com/docs/editor/debugging#_launchjson-attributes
- Further reading and explanations in this demo repo:
  - https://github.com/samkam/go-debugging-example

Thank you!