KUBERMATIC

Marvin Beckers

Ephemeral Containers in Action

Running a Go Debugger in Kubernetes

Cloud_Native Rejekts [EU'23]

Marvin Beckers

Senior Software Engineer @ Kubermatic

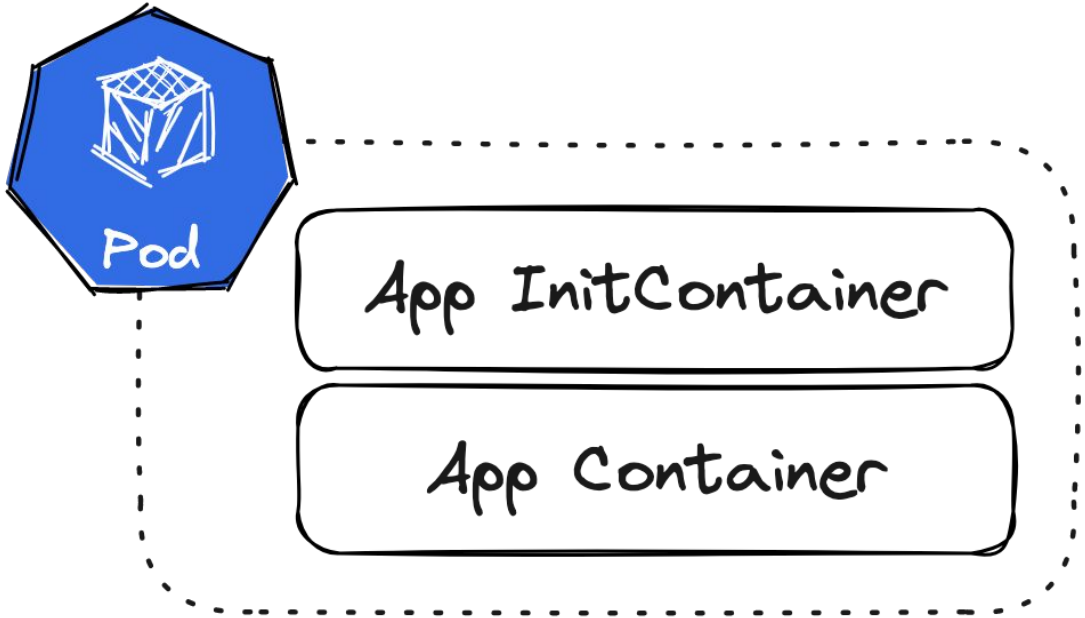
GitHub Slack embik

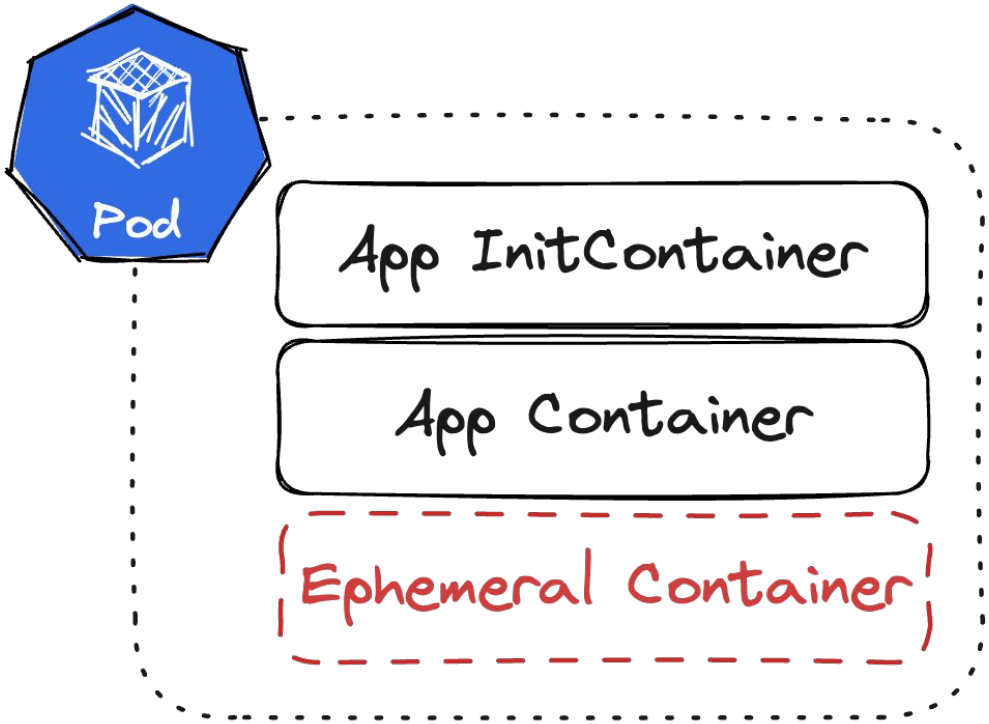
Agenda

1. A Primer on Ephemeral Containers
    a. Command Line Tools
2. Basics of Debugging Go
3. Remote Debugging
4. Delve as Ephemeral Container
5. Remote Debugging in VS Code
6. Live Demo

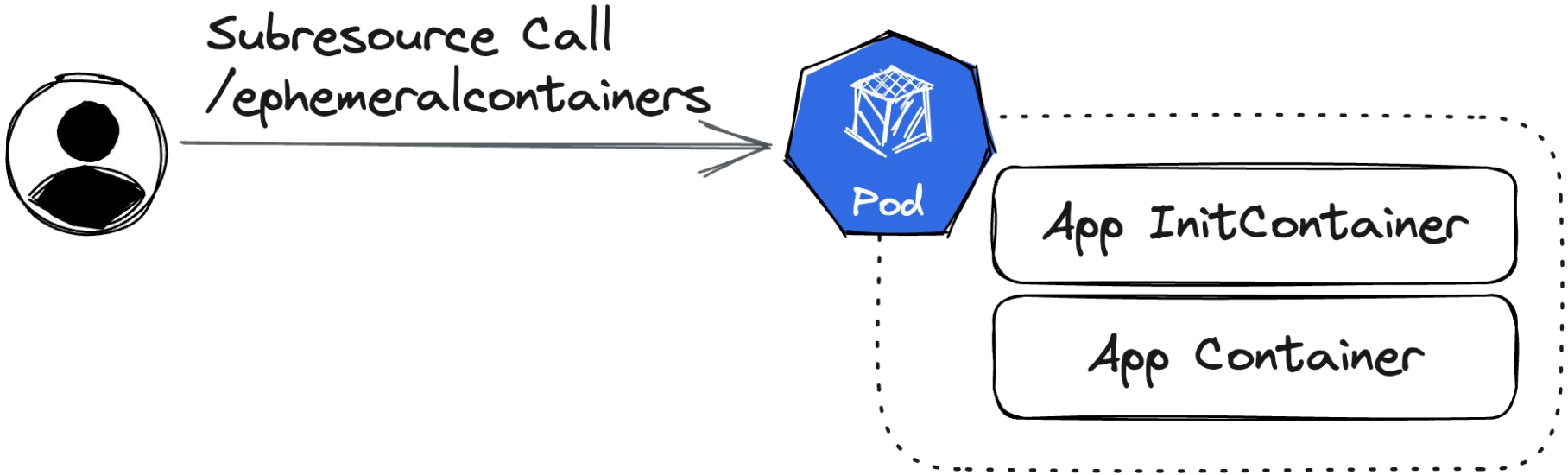
KUBERMATIC

A Primer on Ephemeral Containers


Pod
App InitContainer
App Container



With Kubernetes 1.25+




```
client.Pods(namespace).UpdateEphemeralContainers(...)
```

KUBERMATIC

Command Line Tools

```
$ kubectl debug <pod> -it \
    --image=busybox \
    --target=<container>
```

```
Defaulting debug container name to debugger-jwl89.
If you don't see a command prompt, try pressing enter.
/ #
/ #
/ # ps waux
PID   USER     TIME  COMMAND
    1 root      0:00 /bin/app
   30 root      0:00 sh
   36 root      0:00 ps waux
```

kubectl-ephemeral

- Simple custom kubectl plugin to pass YAML directly
- https://github.com/embik/kubectl-ephemeral

```
$ kubectl ephemeral <target pod name> \
  -f <path to ephemeral container>.yaml
  -c <target container name>
```

KUBERMATIC

Basics of Debugging Go

DELVE

A Debugger for the Go Programming Language

Compile Flags

```
$ go build -gcflags="all=-N -l"
```

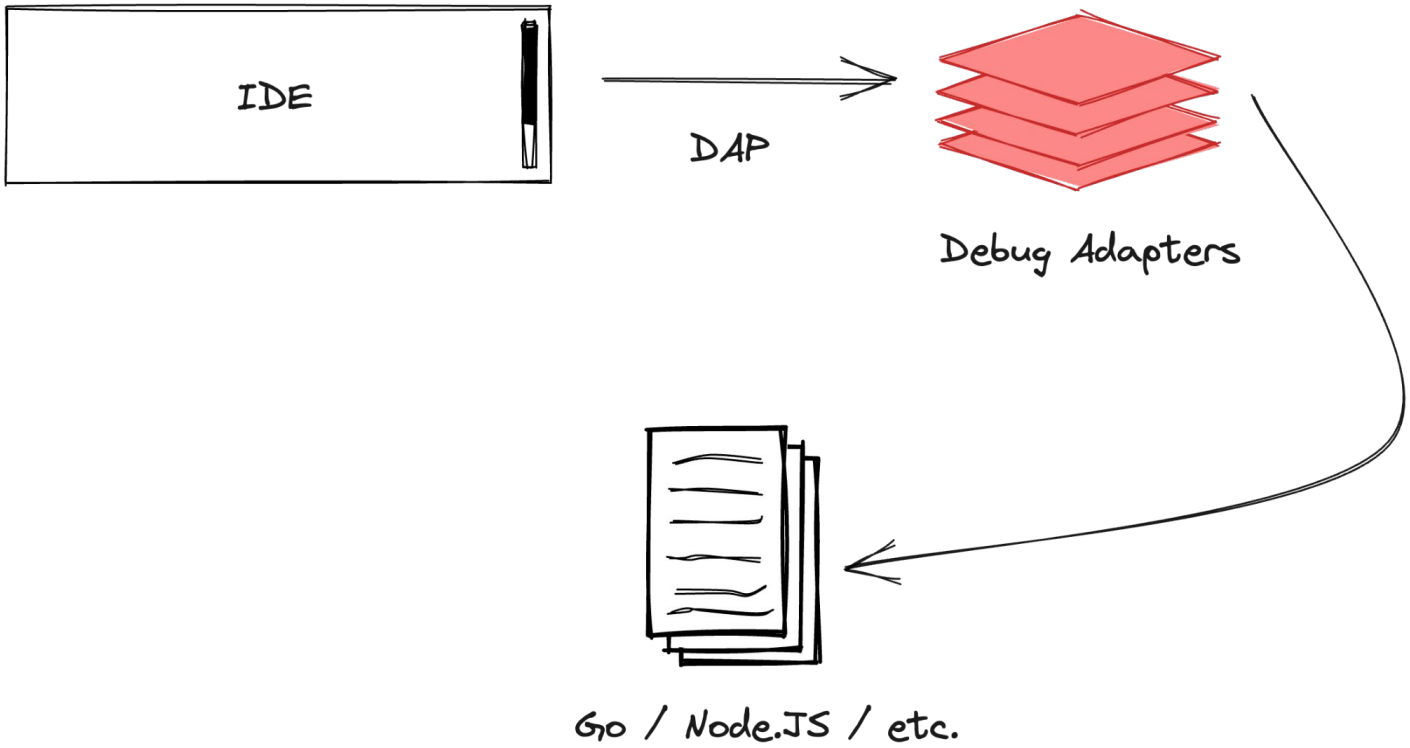
```
-gcflags '[pattern=]arg list'
    arguments to pass on each go tool compile invocation.
```

```
-N disable optimizations
-l disable inlining
```

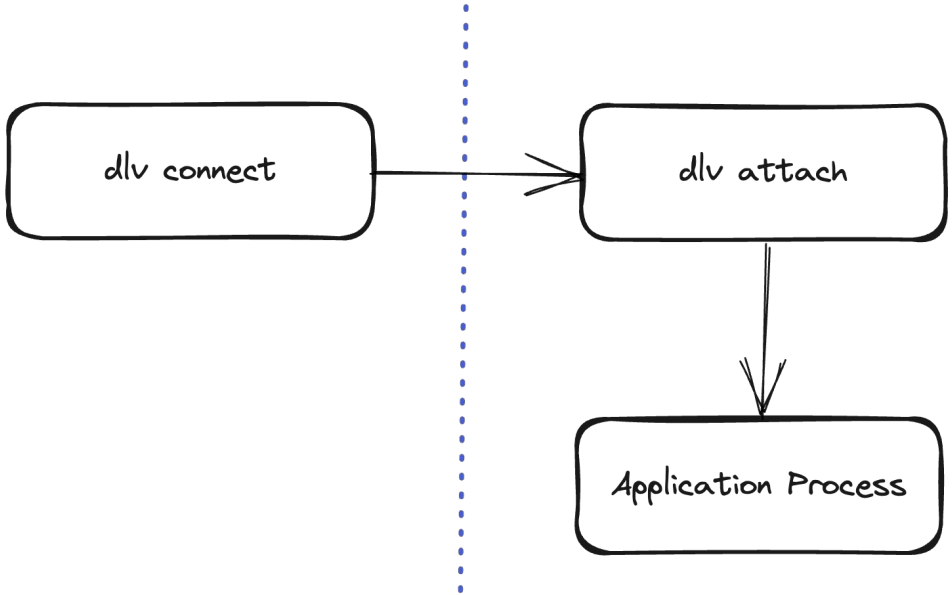
KUBERMATIC

Remote Debugging

DAP - Debug Adapter Protocol



Headless Delve Instance



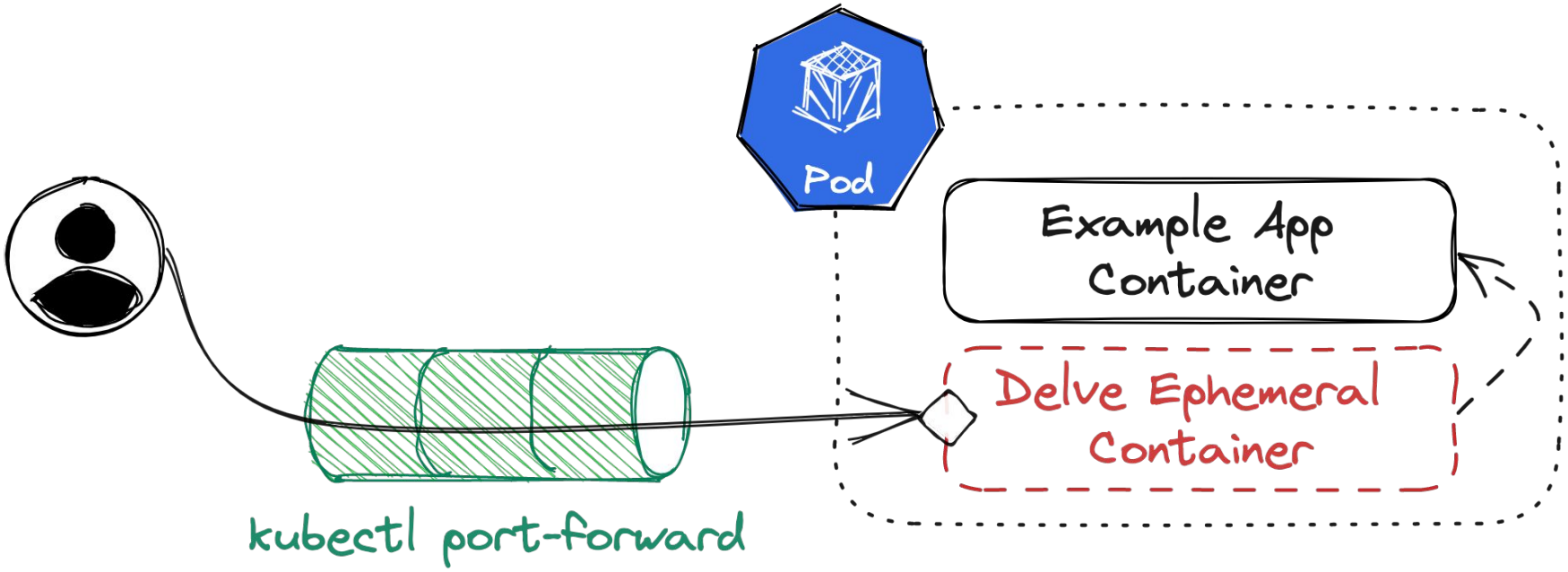

```
$ dlv attach <Target PID> --listen=:2345 --headless=true
```

KUBERMATIC

Delve as Ephemeral Container

EphemeralContainer YAML

```yaml
name: delve
image: quay.io/embik/dlv:v1.20.1
securityContext:
  privileged: true
command:
  - dlv
  - --listen=127.0.0.1:2345
  - --headless=true
  - --accept-multiclient
  - --api-version=2
  - attach
  - '1'
```




```
$ dlv connect localhost:2345
```

KUBERMATIC

Remote Debugging in VS Code

... or neovim (via nvim-dap)!

.vscode/launch.json

```
{
    "name": "Remote Attach",
    "type": "go",
    "request": "attach",
    "debugAdapter": "dlv-dap",
    "mode": "remote",
    "substitutePath": [
      { "from": "${workspaceFolder}", "to": "/build" }
    ],
    "port": 2345,
    "host": "127.0.0.1"
}
```

Application Dockerfile

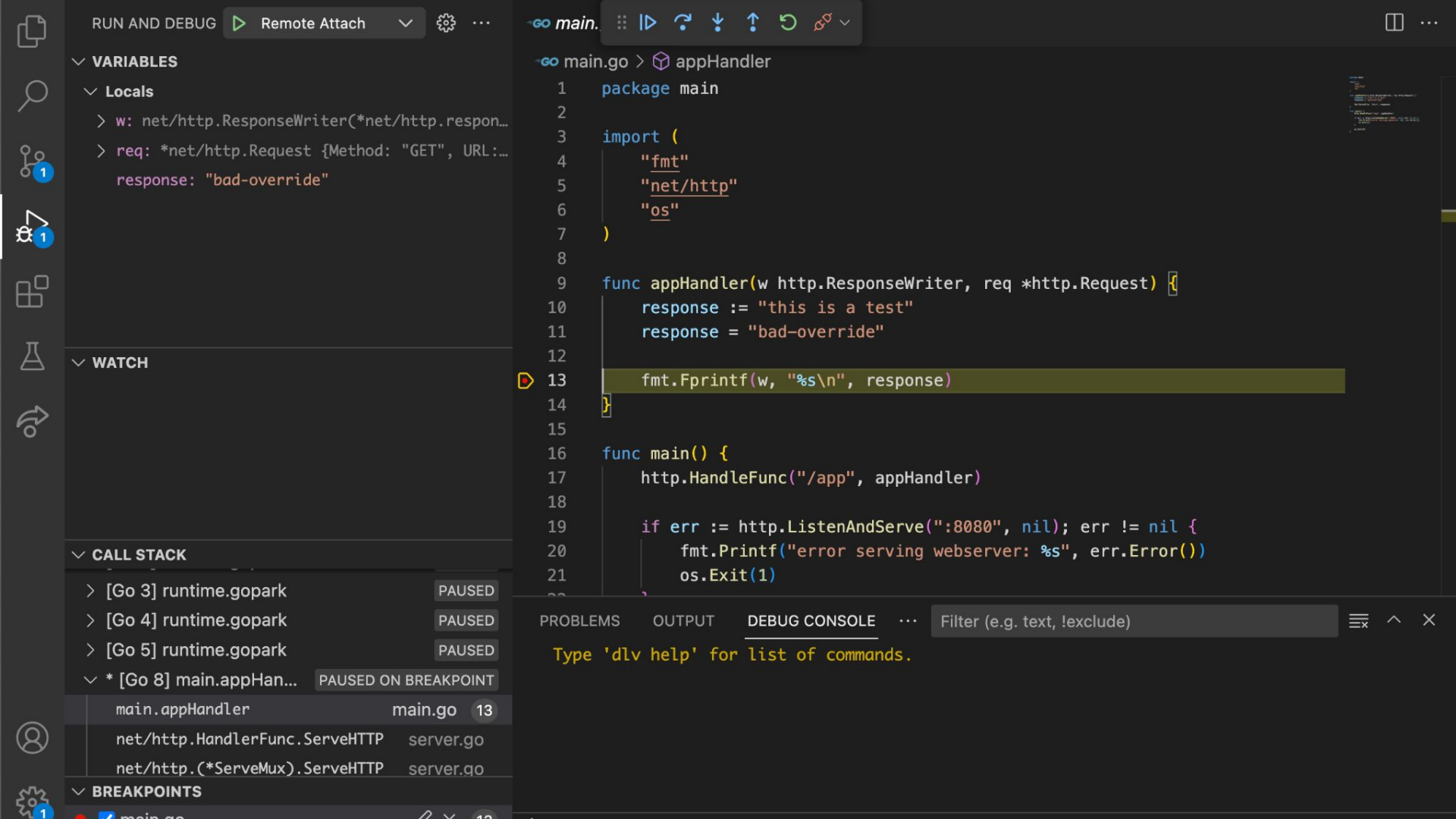
```
FROM golang:1.19.8 as builder

WORKDIR /build
COPY . .
RUN CGO_ENABLED=0 go build -gcflags="all=-N -l" -o app .

FROM alpine:3.16
LABEL maintainer="marvin@kubermatic.com"

COPY --from=builder /build/app /bin/app

ENTRYPOINT ["/bin/app"]
```


RUN AND DEBUG
Remote Attach
VARIABLES
Locals
w: net/http.ResponseWriter(*net/http.respon…
req: *net/http.Request {Method: "GET", URL:…
response: "bad-override"
WATCH
CALL STACK
[Go 3] runtime.gopark PAUSED
[Go 4] runtime.gopark PAUSED
[Go 5] runtime.gopark PAUSED
* [Go 8] main.appHan… PAUSED ON BREAKPOINT
main.appHandler main.go 13
net/http.HandlerFunc.ServeHTTP server.go
net/http.(*ServeMux).ServeHTTP server.go
BREAKPOINTS
main.go
main.go > appHandler
1 package main
2
3 import (
4 "fmt"
5 "net/http"
6 "os"
7)
8
9 func appHandler(w http.ResponseWriter, req *http.Request) {
10 response := "this is a test"
11 response = "bad-override"
12
13 fmt.Fprintf(w, "%s\n", response)
14 }
15
16 func main() {
17 http.HandleFunc("/app", appHandler)
18
19 if err := http.ListenAndServe(":8080", nil); err != nil {
20 fmt.Printf("error serving webserver: %s", err.Error())
21 os.Exit(1)
PROBLEMS
OUTPUT
DEBUG CONSOLE
Filter (e.g. text, !exclude)
Type 'dlv help' for list of commands.

```
Locals:
 ▶ w net/http.ResponseWriter = net/http.
 ▶ req *net/http.Request = *net/http.Req
   response string = "bad-override"
```

```
main.go:
 13 fmt.Fprintf(w, "%s\n", response)
```

```
* [Go 33] main.appHandler (Thread 10):
 main.appHandler main.go:13
 net/http.HandlerFunc.ServeHTTP server.g
 net/http.(*ServeMux).ServeHTTP server.g
 net/http.serverHandler.ServeHTTP server
 net/http.(*conn).serve server.go:1991
 net/http.(*Server).Serve.func3 server.g
```

```
No Expressions
```

```go
    "os"
)

func appHandler(w http.ResponseWriter, req *http.Request) {
    response := "this is a test"
    response = "bad-override"

    fmt.Fprintf(w, "%s\n", response)
}

func main() {
    http.HandleFunc("/app", appHandler)

    if err := http.ListenAndServe(":8080", nil); err != nil {
        fmt.Printf("error serving webserver: %s", err.Error())
        os.Exit(1)
    }
```

```
Type 'dlv help' for list of commands.
```

```
NORMAL  main  main.go                         utf-8 < unix < go  68%  17:5
```

GitHub Repositories

- https://github.com/embik/ephemeral-debugger-talk
- https://github.com/embik/kubectl-ephemeral

KUBERMATIC

Let’s see this in practice

Quick Live Demo

KUBERMATIC

Thank You!