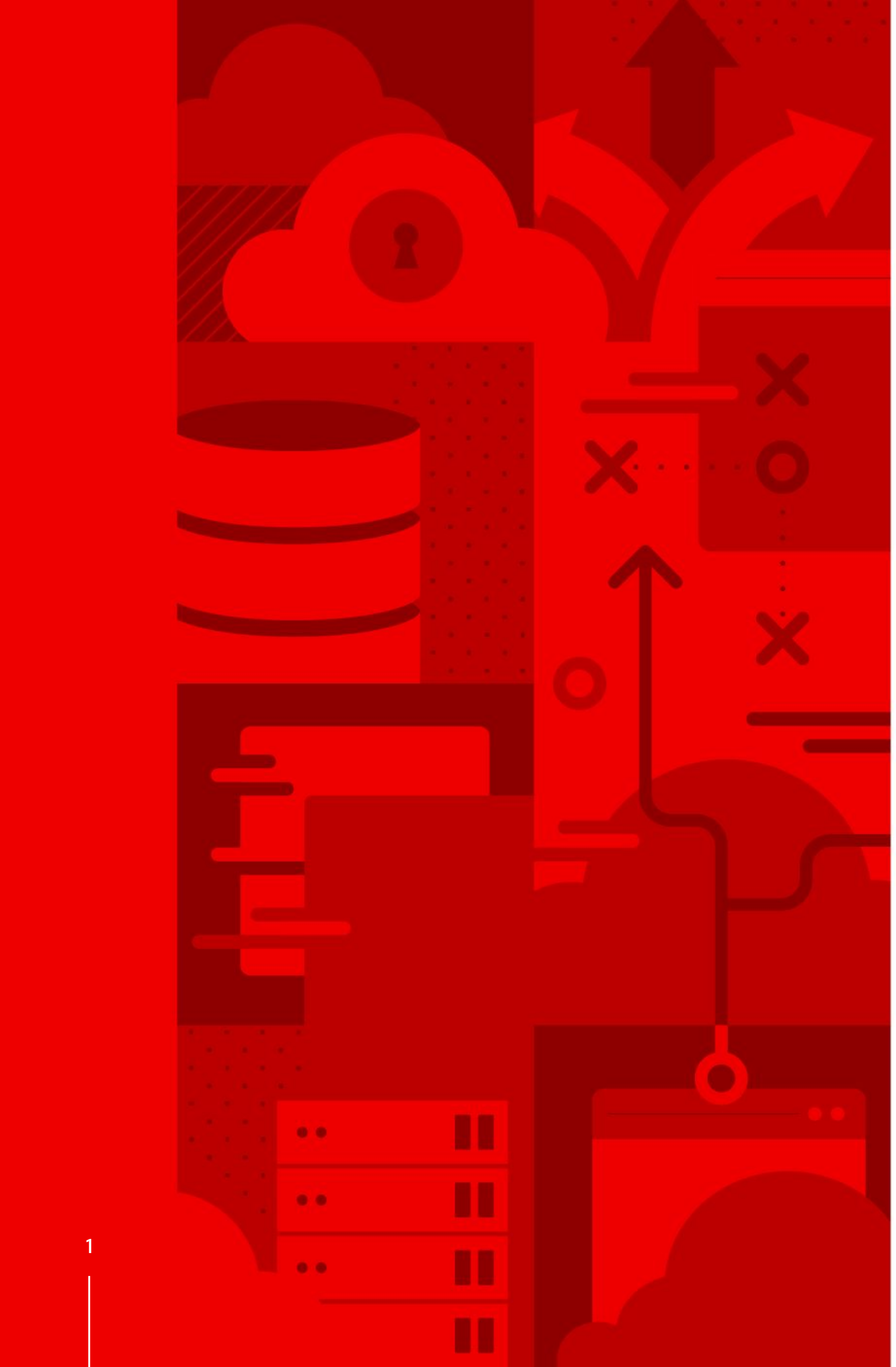


vfkit - A native macOS hypervisor written in go

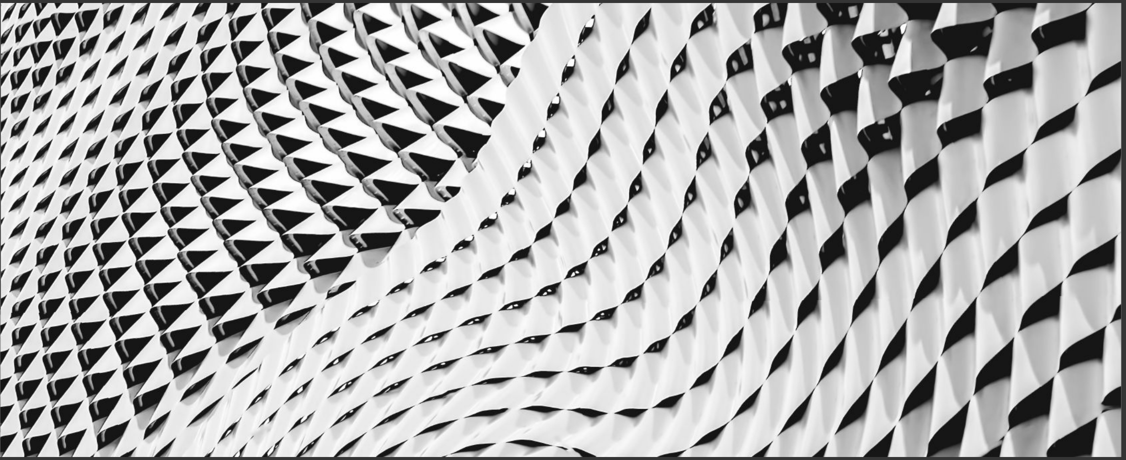
Christophe Fergeau
Senior Software Engineer

Agenda

What we'll discuss today

- Introduction
- Apple's Virtualization Framework
- vfkit
- Objective-C go bindings

Background



CRC

- https://github.com/crc-org/crc/
- Runs a local OpenShift cluster on a macOS/Windows/Linux machine
  - Red Hat OpenShift is a Kubernetes distribution
- The cluster runs in a virtual machine

macOS hypervisors

macOS Hypervisors (free software / command line)

- HyperKit
- QEMU
- ??

macOS hypervisors

macOS Hypervisors (free software / command line)

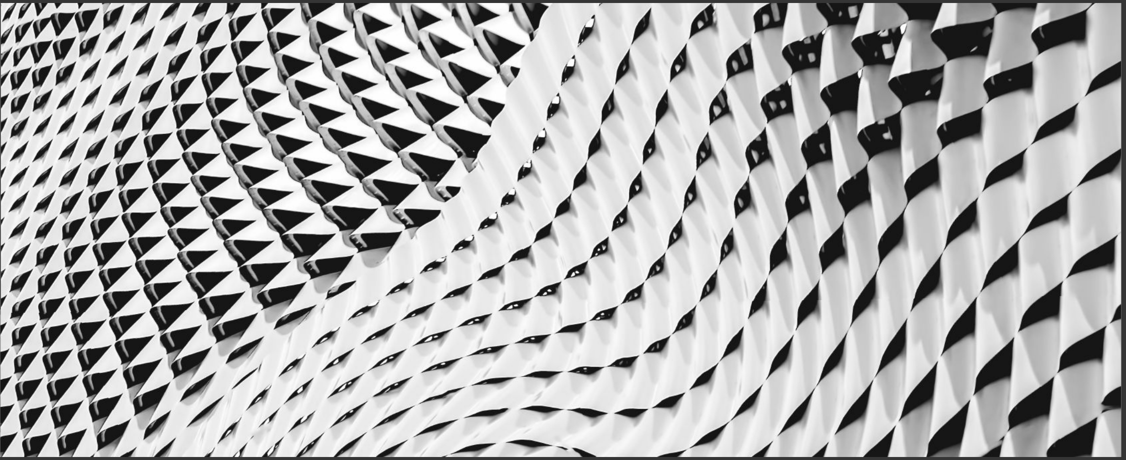
- ~~HyperKit~~ No support for Apple Silicon ARM CPUs
- QEMU
- ??

macOS hypervisors

macOS Hypervisors (free software / command line)

- ~~HyperKit~~ No support for Apple Silicon ARM CPUs
- ~~QEMU~~ Millions of lines of C code, and we would need to maintain our own builds, track CVEs, ..
- ??

Apple’s Virtualization Framework

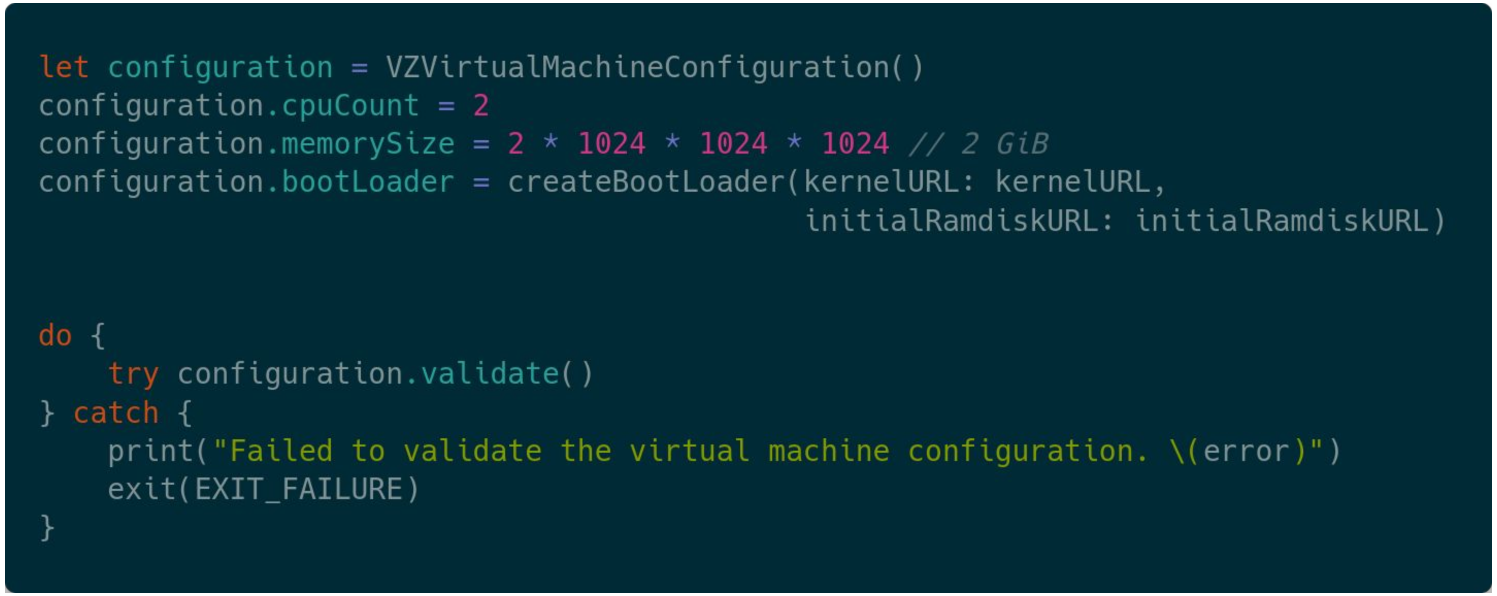


Apple's Virtualization Framework

- https://developer.apple.com/documentation/virtualization
- Available in macOS 11 and newer
- High-level API to create Linux and macOS virtual machines
- Can be used from Swift or Objective-C
- Just an API/a framework (library), not an end-user application

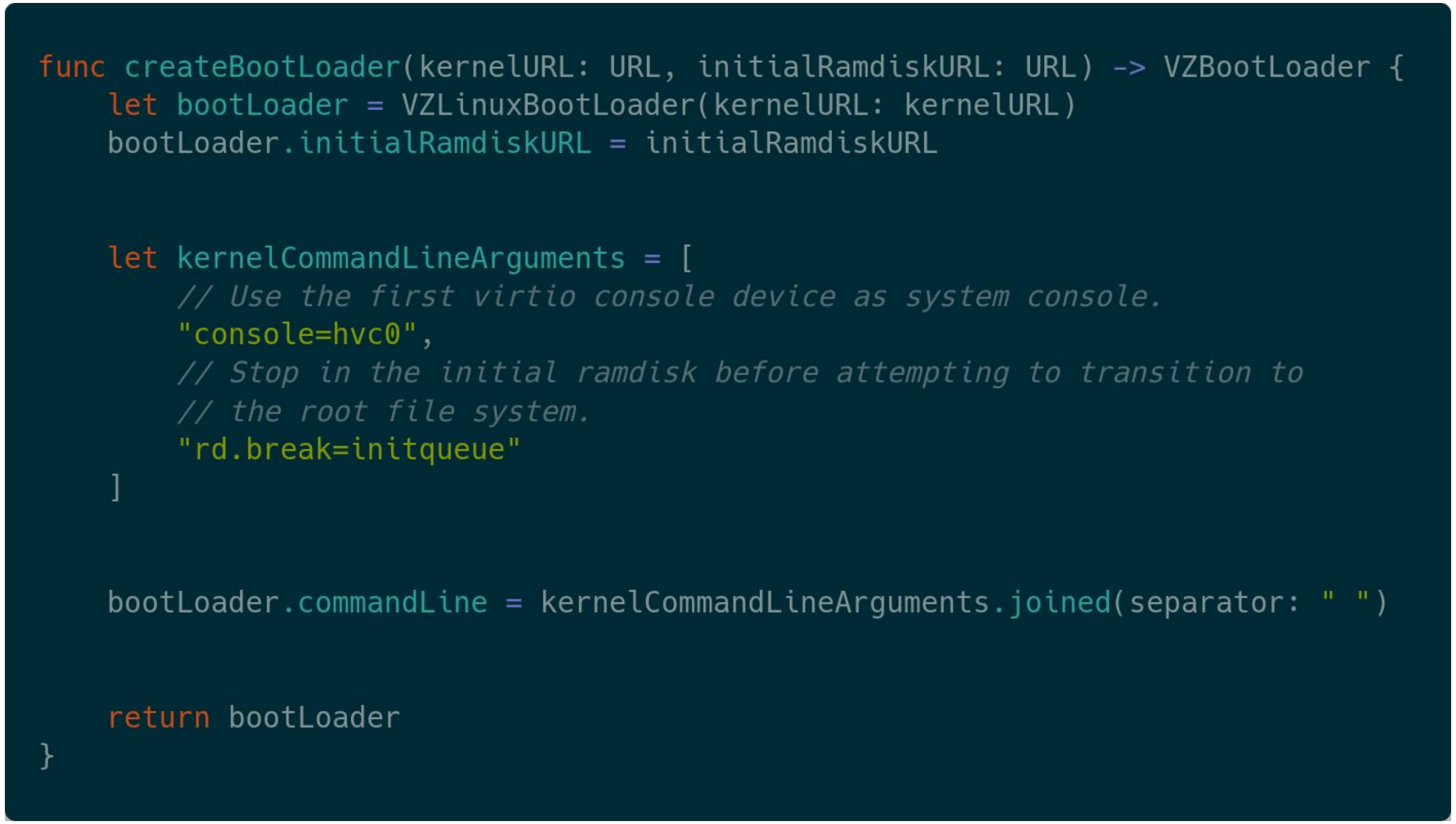
Apple’s Virtualization Framework

- Only supports devices needed in virtual machines, mostly virtio devices
- virtio-net, virtio-blk, virtio-serial, virtio-rng, …
- virtio-fs for file sharing between the host and the guest
- virtio-vsock for communication between the host and the guest using POSIX sockets
- Rosetta support to run amd64 linux binaries in arm64 linux guests



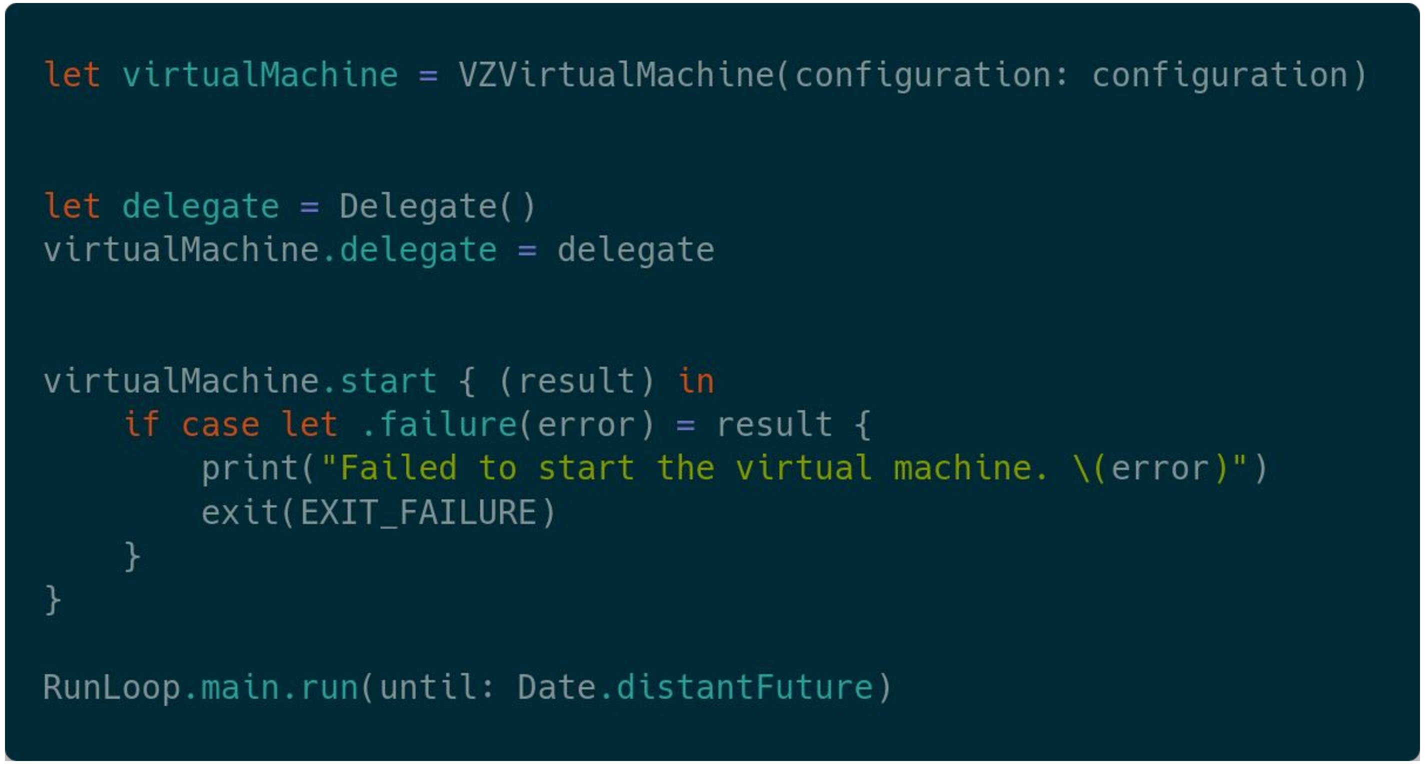
```
let configuration = VZVirtualMachineConfiguration()
configuration.cpuCount = 2
configuration.memorySize = 2 * 1024 * 1024 * 1024 // 2 GiB
configuration.bootLoader = createBootLoader(kernelURL: kernelURL,
                                            initialRamdiskURL: initialRamdiskURL)


do {
    try configuration.validate()
} catch {
    print("Failed to validate the virtual machine configuration. \(error)")
    exit(EXIT_FAILURE)
}
```

Configuration of the virtual machine



```
func createBootLoader(kernelURL: URL, initialRamdiskURL: URL) -> VZBootLoader {
    let bootLoader = VZLinuxBootLoader(kernelURL: kernelURL)
    bootLoader.initialRamdiskURL = initialRamdiskURL


    let kernelCommandLineArguments = [
        // Use the first virtio console device as system console.
        "console=hvc0",
        // Stop in the initial ramdisk before attempting to transition to
        // the root file system.
        "rd.break=initqueue"
    ]


    bootLoader.commandLine = kernelCommandLineArguments.joined(separator: " ")


    return bootLoader
}
```

Bootloader configuration

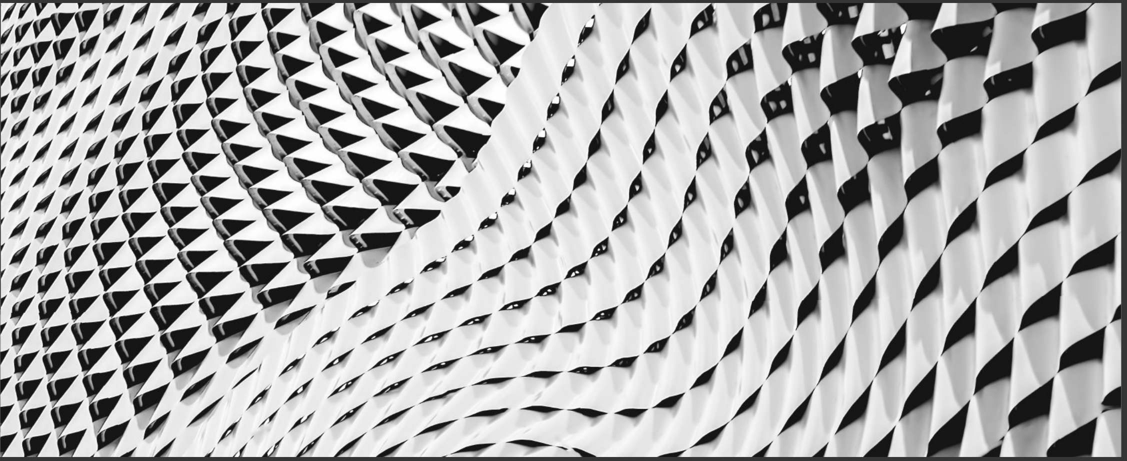


```
let virtualMachine = VZVirtualMachine(configuration: configuration)


let delegate = Delegate()
virtualMachine.delegate = delegate


virtualMachine.start { (result) in
    if case let .failure(error) = result {
        print("Failed to start the virtual machine. \(error)")
        exit(EXIT_FAILURE)
    }
}

RunLoop.main.run(until: Date.distantFuture)
```

Create the virtual machine and run it

What’s up with the Swift code and this non-free framework ? Aren’t we in the FOSDEM Go Devroom ?

vfkit



Code-Hex/vz

- https://github.com/Code-Hex/vz
- Written by Kei “Code-Hex” Kamikawa
- Go bindings for Apple’s Virtualization Framework
- MIT Licensing
- Follow closely new APIs added in macOS 12 and macOS 13

Code-Hex/vz

- Not enough!
- Need a long-lived process

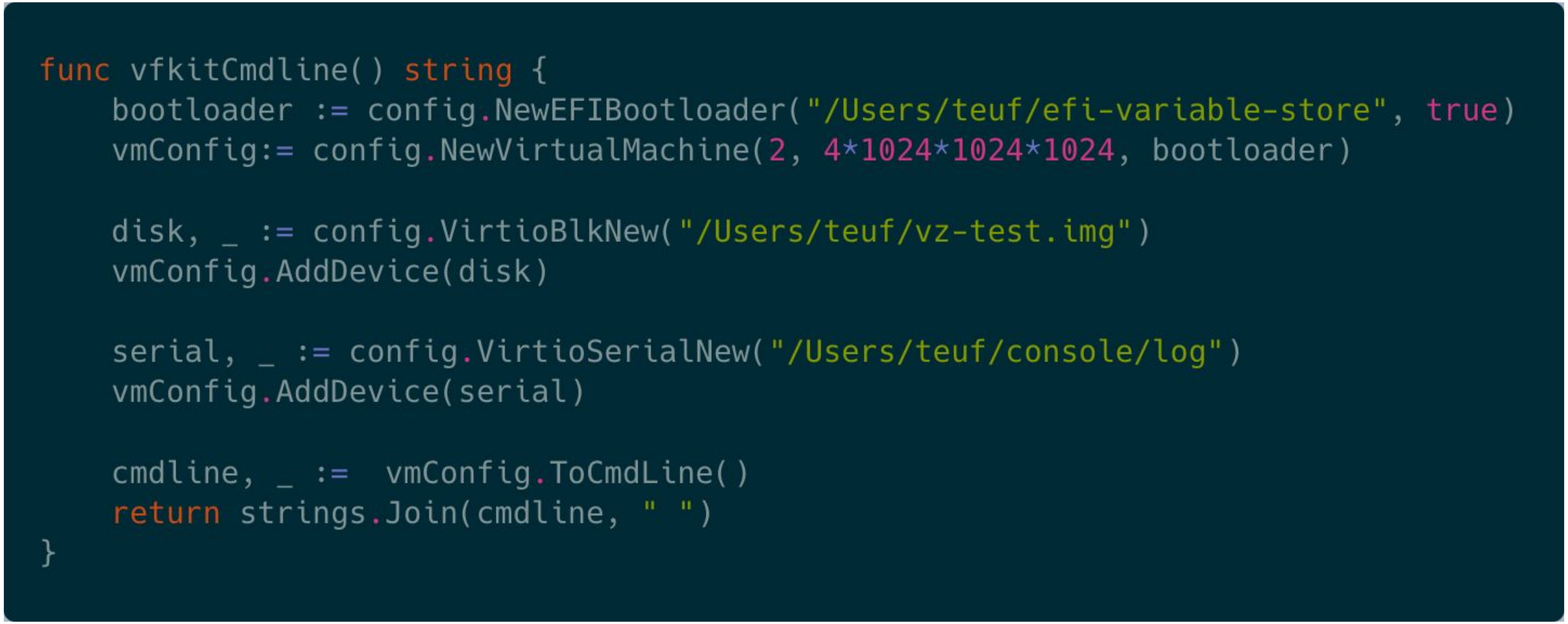
vfkit

- https://github.com/crc-org/vfkit
- Command-line tool which uses the Code-Hex/vz bindings
- Written in go
- Apache v2 Licensing

vfkit

```
./out/vfkit --cpus 2 --memory 2048 \
    --bootloader efi,variable-store=/Users/teuf/efi-variable-store,create \
    --device virtio-serial,stdio \
    --device virtio-fs,sharedDir=/Users/teuf,mountTag=dir0 \
    --device virtio-blk,path=/Users/teuf/vz-test.img \
    --device virtio-blk,path=/Users/teuf/Fedora-Server-x86_64-37-1.7.iso \
    --device virtio-rng \
    --device virtio-net,nat,mac=72:20:43:d4:38:62
```

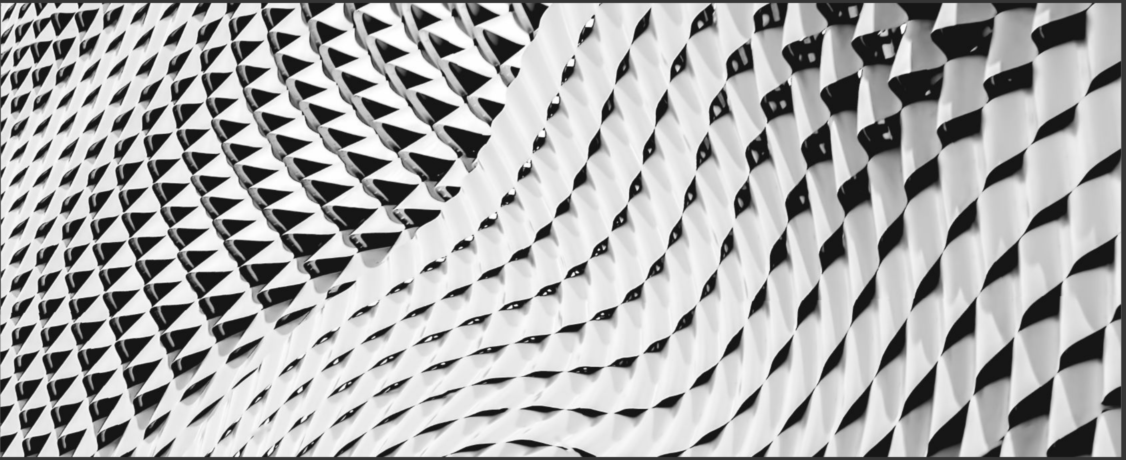
vfkit



```
func vfkitCmdline() string {
    bootloader := config.NewEFIBootloader("/Users/teuf/efi-variable-store", true)
    vmConfig:= config.NewVirtualMachine(2, 4*1024*1024*1024, bootloader)

    disk, _ := config.VirtioBlkNew("/Users/teuf/vz-test.img")
    vmConfig.AddDevice(disk)

    serial, _ := config.VirtioSerialNew("/Users/teuf/console/log")
    vmConfig.AddDevice(serial)

    cmdline, _ :=  vmConfig.ToCmdLine()
    return strings.Join(cmdline, " ")
}
```

- `github.com/crc-org/vfkit/pkg/config` provides a go API to generate this command line

Using Objective-C code from go



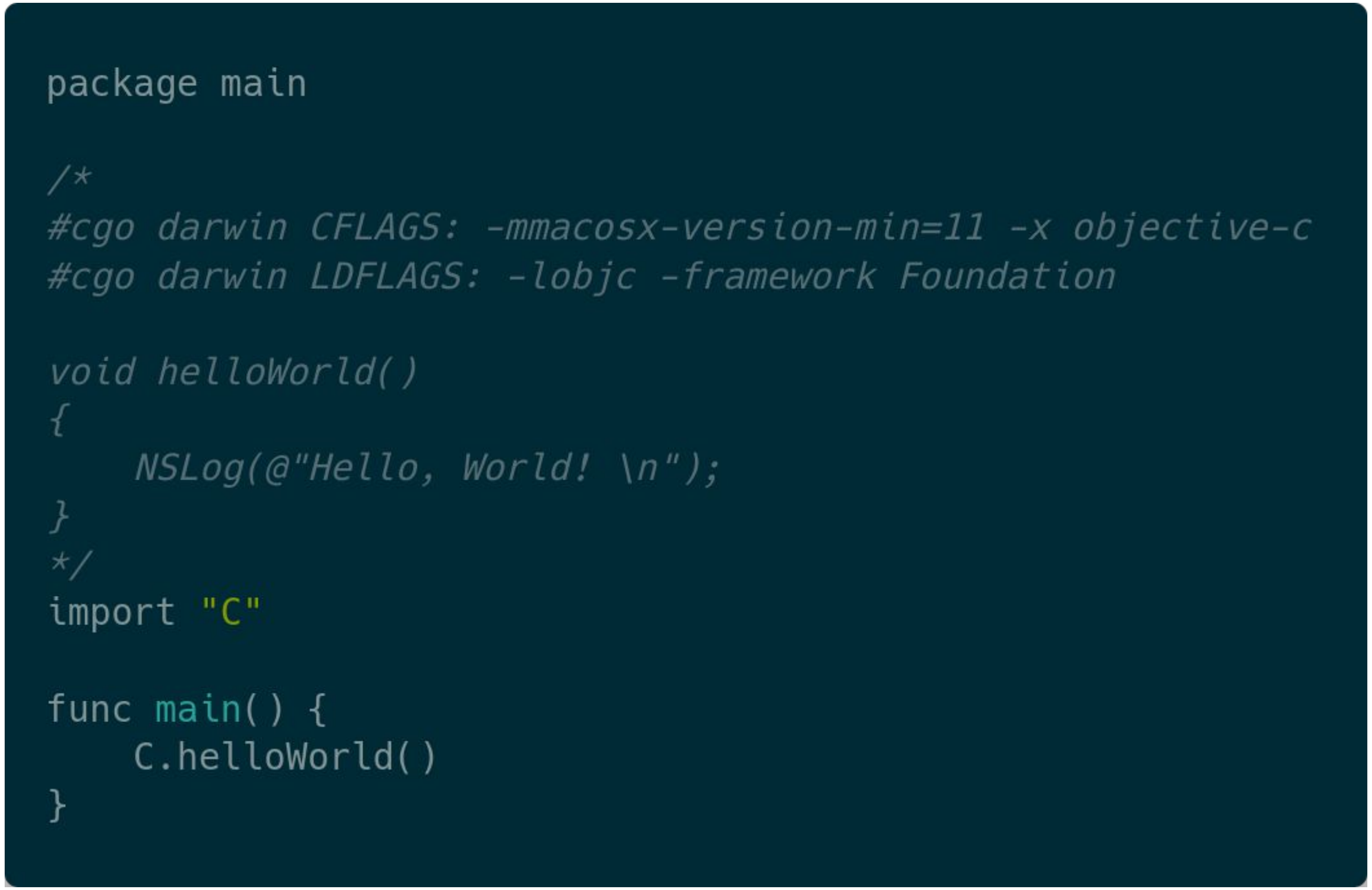
Go bindings for objc APIs

- Very similar to binding C code
  - `import “C”`
  - https://pkg.go.dev/cmd/cgo
  - https://pkg.go.dev/runtime/cgo
- Most of the interactions between go and Objective-C can be done through C types

Go bindings for objc APIs

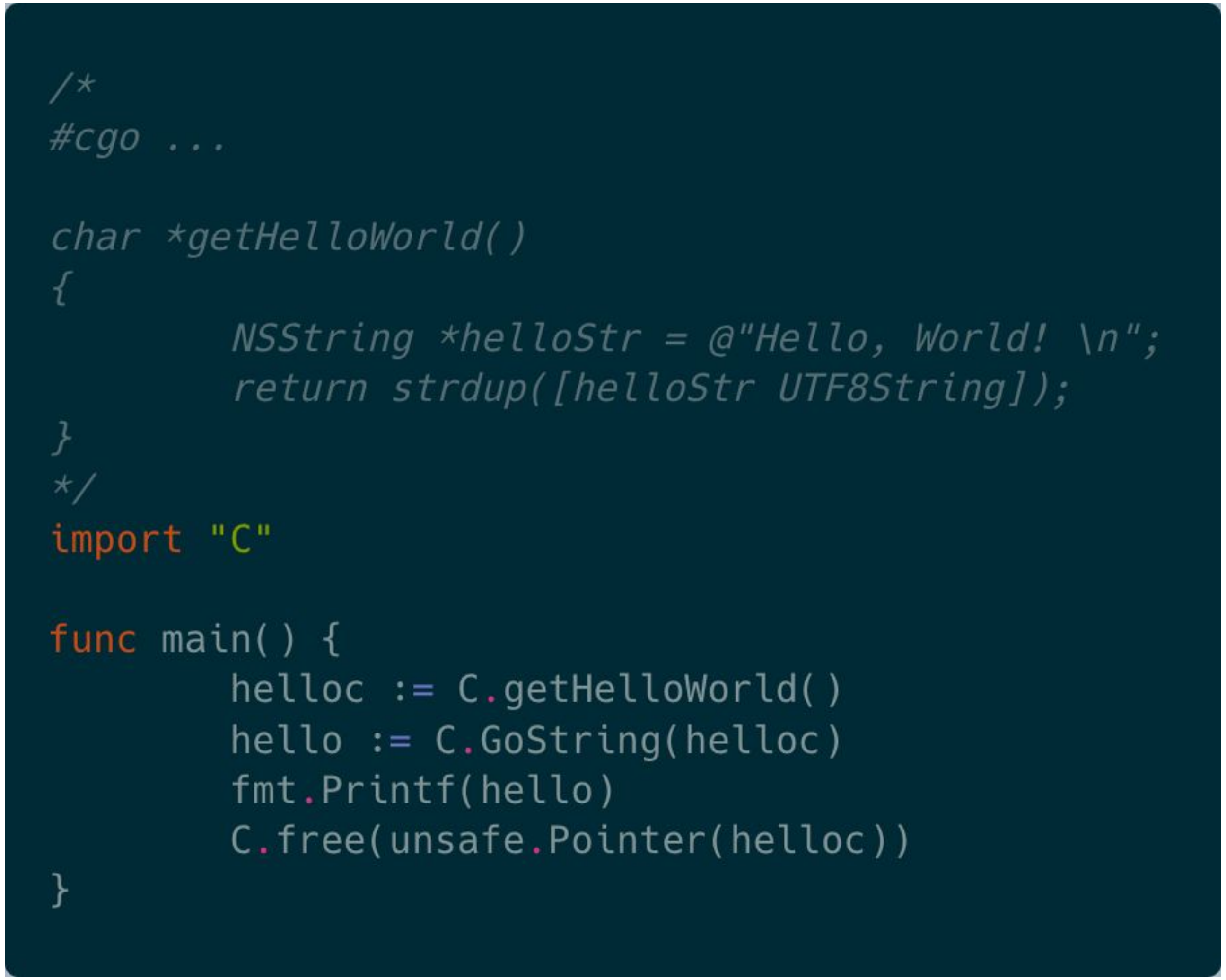
- arm64 and amd64 code can be built on the same machine
  - Go module to generate universal binaries
- Do not forget the ; at the end of each line!
- Compilation can get slow

Simple examples using strings



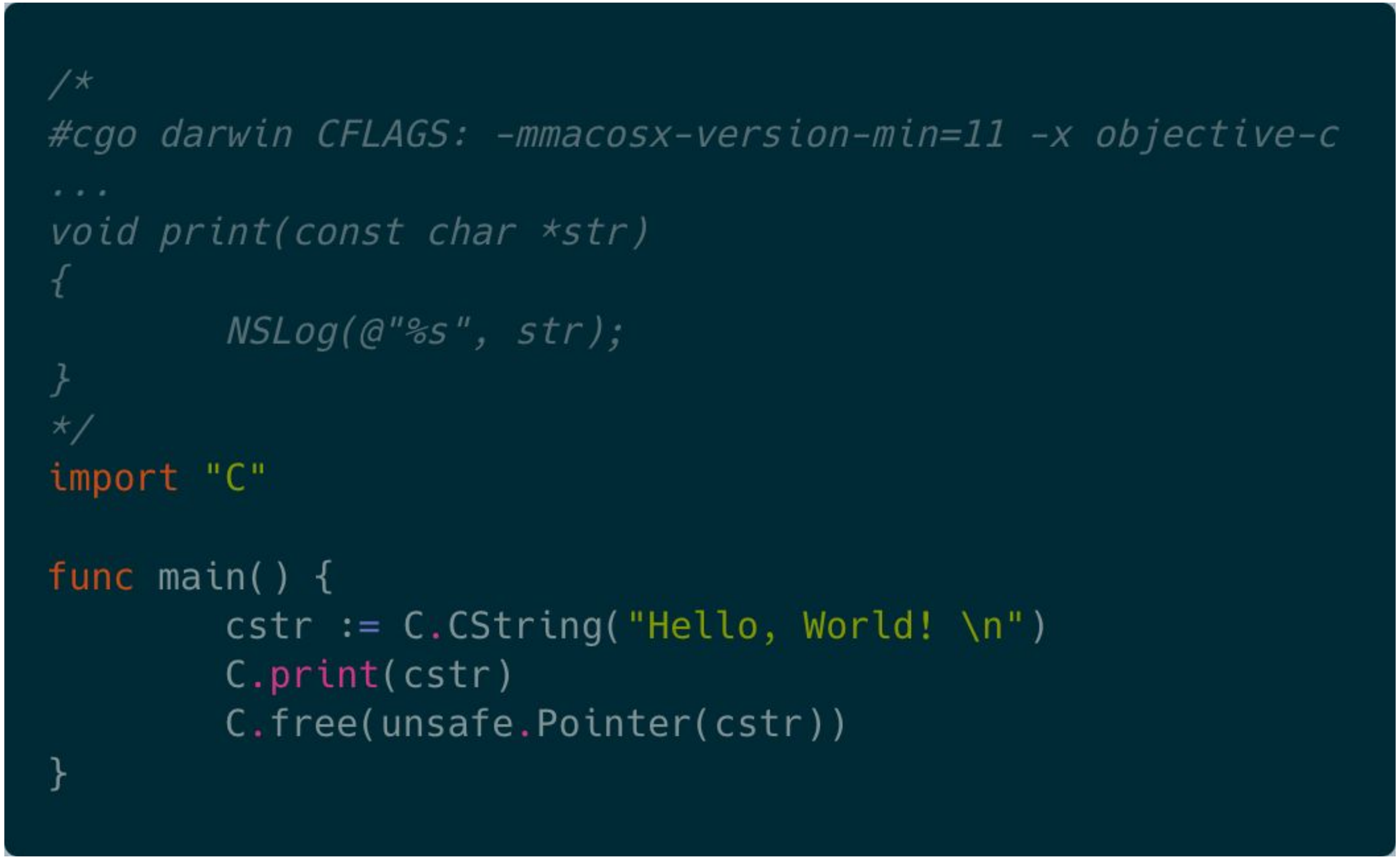
```
package main

/*
#cgo darwin CFLAGS: -mmacosx-version-min=11 -x objective-c
#cgo darwin LDFLAGS: -lobjc -framework Foundation

void helloWorld()
{
    NSLog(@"Hello, World! \n");
}
*/
import "C"

func main() {
    C.helloWorld()
}
```

Simple examples using strings (2)



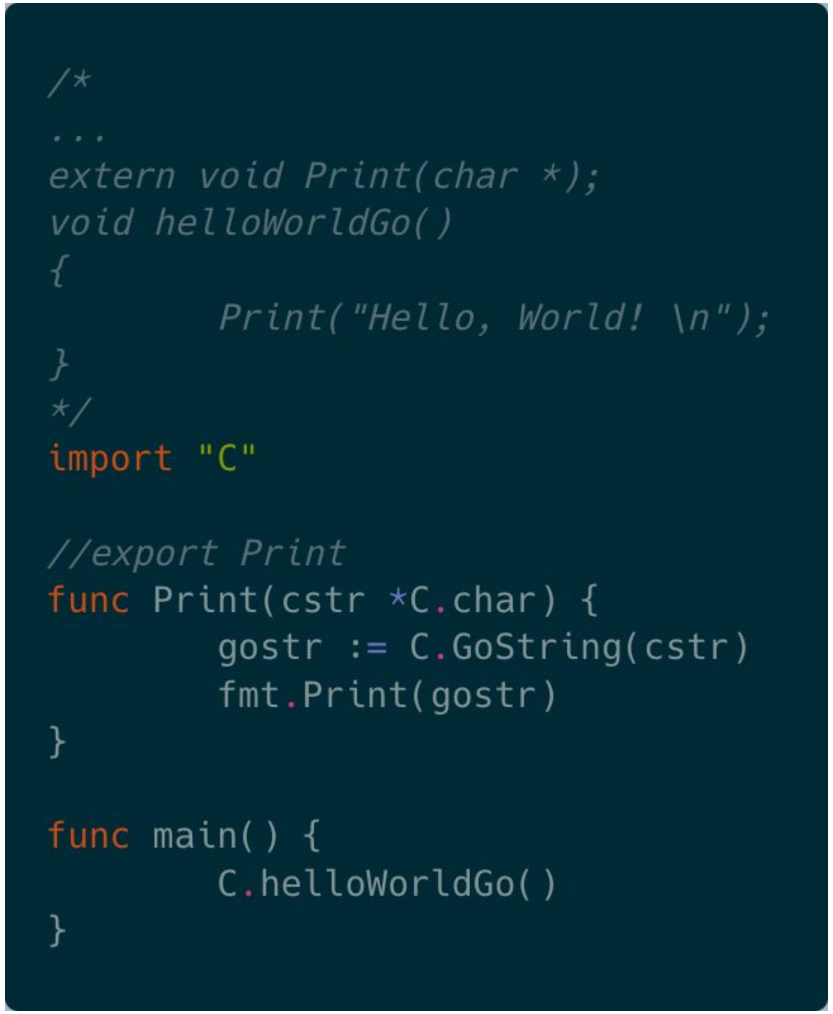
```
/*
#cgo ...

char *getHelloWorld()
{
        NSString *helloStr = @"Hello, World! \n";
        return strdup([helloStr UTF8String]);
}
*/
import "C"

func main() {
        helloc := C.getHelloWorld()
        hello := C.GoString(helloc)
        fmt.Printf(hello)
        C.free(unsafe.Pointer(helloc))
}
```

Simple examples using strings (3)



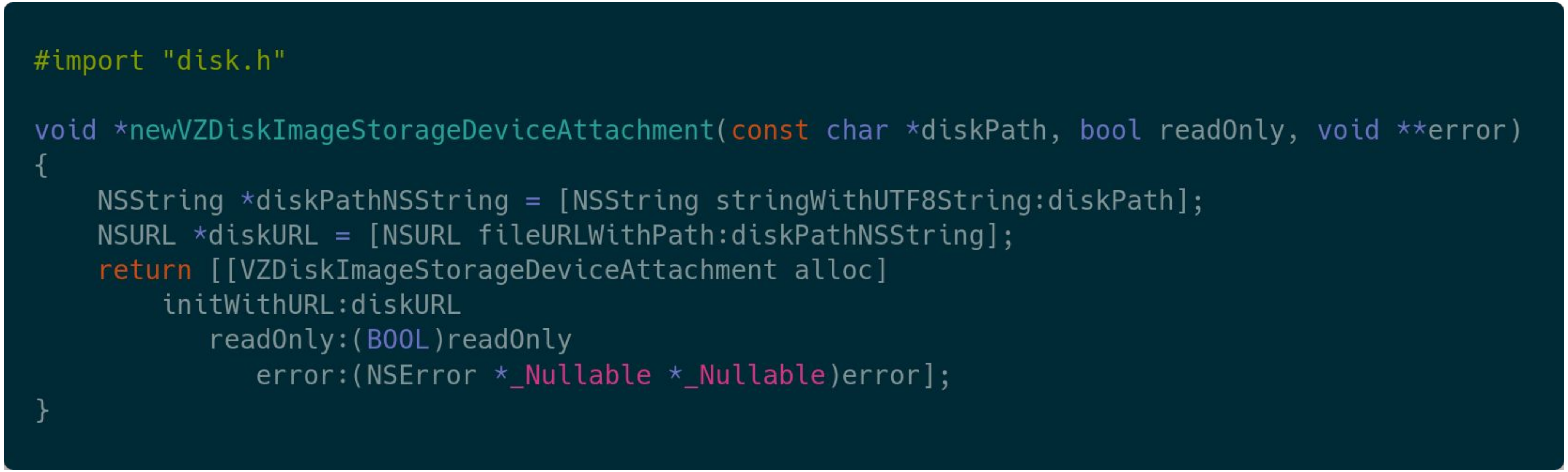
```
/*
#cgo darwin CFLAGS: -mmacosx-version-min=11 -x objective-c
...
void print(const char *str)
{
        NSLog(@"%s", str);
}
*/
import "C"

func main() {
        cstr := C.CString("Hello, World! \n")
        C.print(cstr)
        C.free(unsafe.Pointer(cstr))
}
```

Simple examples using strings (4)



```
/*
...
extern void Print(char *);
void helloWorldGo()
{
        Print("Hello, World! \n");
}
*/
import "C"

//export Print
func Print(cstr *C.char) {
        gostr := C.GoString(cstr)
        fmt.Print(gostr)
}

func main() {
        C.helloWorldGo()
}
```

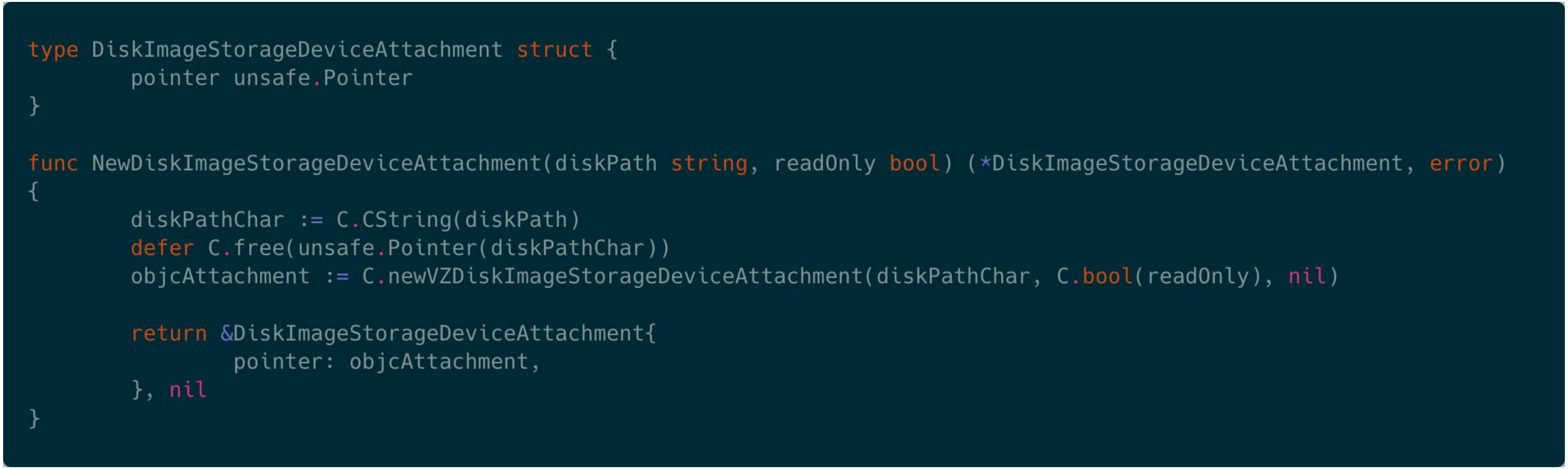
Calling virtualization framework methods from go



```
#import "disk.h"

void *newVZDiskImageStorageDeviceAttachment(const char *diskPath, bool readOnly, void **error)
{
    NSString *diskPathNSString = [NSString stringWithUTF8String:diskPath];
    NSURL *diskURL = [NSURL fileURLWithPath:diskPathNSString];
    return [[VZDiskImageStorageDeviceAttachment alloc]
        initWithURL:diskURL
           readOnly:(BOOL)readOnly
              error:(NSError *_Nullable *_Nullable)error];
}
```

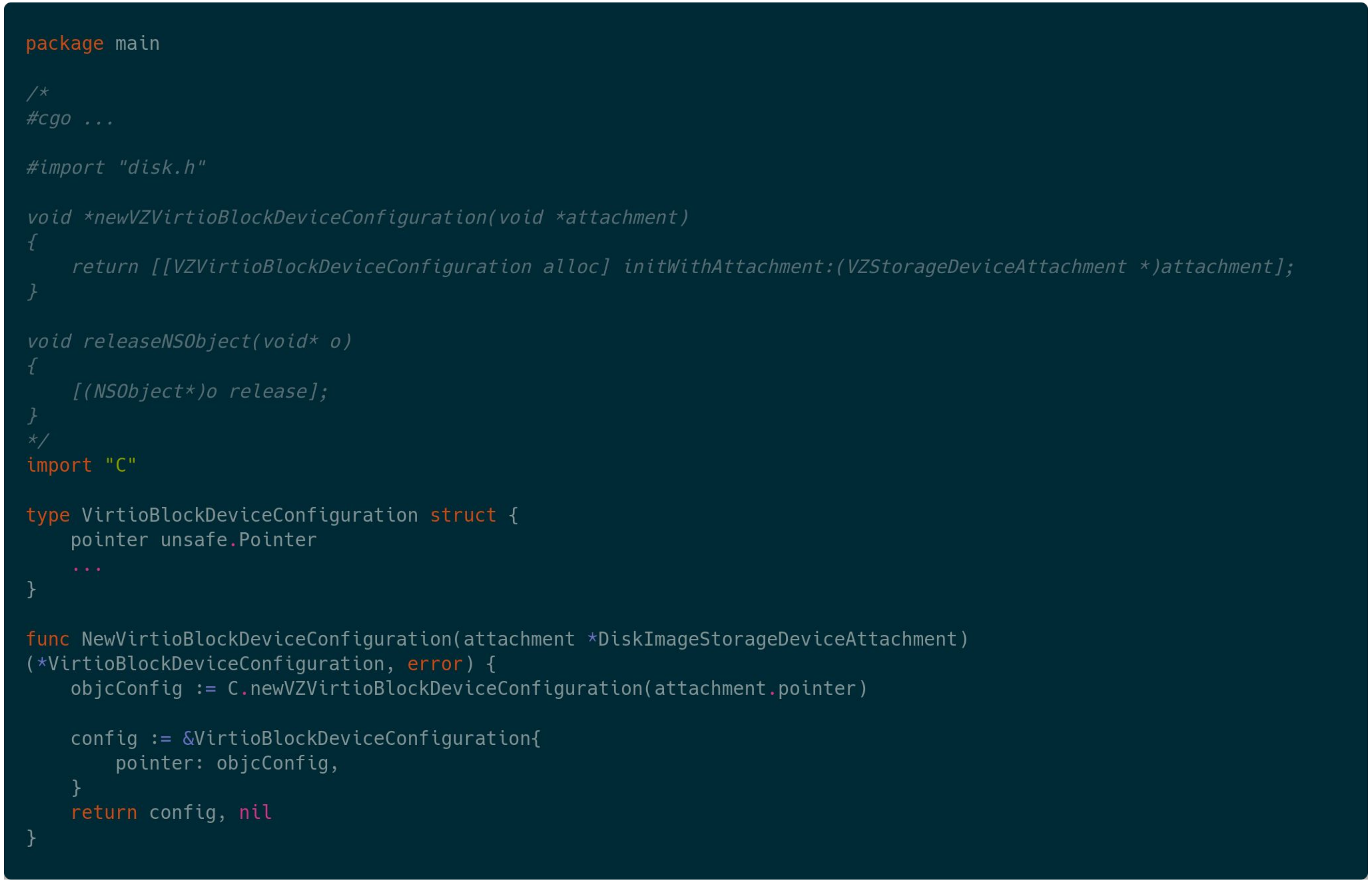
VZDiskImageStorageDeviceAttachment API documentation

Calling virtualization framework methods from go



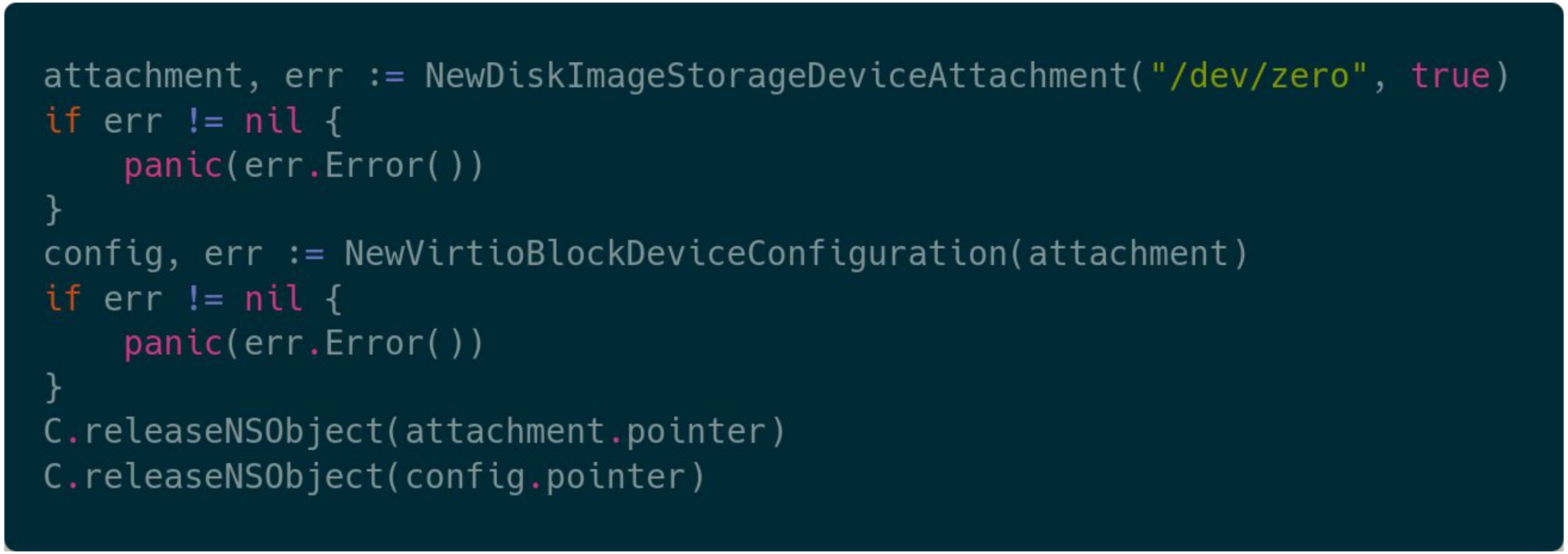
```
type DiskImageStorageDeviceAttachment struct {
        pointer unsafe.Pointer
}

func NewDiskImageStorageDeviceAttachment(diskPath string, readOnly bool) (*DiskImageStorageDeviceAttachment, error)
{
        diskPathChar := C.CString(diskPath)
        defer C.free(unsafe.Pointer(diskPathChar))
        objcAttachment := C.newVZDiskImageStorageDeviceAttachment(diskPathChar, C.bool(readOnly), nil)

        return &DiskImageStorageDeviceAttachment{
                pointer: objcAttachment,
        }, nil
}
```

Calling virtualization framework methods from go



```go
package main

/*
#cgo ...

#import "disk.h"

void *newVZVirtioBlockDeviceConfiguration(void *attachment)
{
    return [[VZVirtioBlockDeviceConfiguration alloc] initWithAttachment:(VZStorageDeviceAttachment *)attachment];
}

void releaseNSObject(void* o)
{
    [(NSObject*)o release];
}
*/
import "C"

type VirtioBlockDeviceConfiguration struct {
    pointer unsafe.Pointer
    ...
}

func NewVirtioBlockDeviceConfiguration(attachment *DiskImageStorageDeviceAttachment)
(*VirtioBlockDeviceConfiguration, error) {
    objcConfig := C.newVZVirtioBlockDeviceConfiguration(attachment.pointer)

    config := &VirtioBlockDeviceConfiguration{
        pointer: objcConfig,
    }
    return config, nil
}
```

Calling virtualization framework methods from go



```go
attachment, err := NewDiskImageStorageDeviceAttachment("/dev/zero", true)
if err != nil {
    panic(err.Error())
}
config, err := NewVirtioBlockDeviceConfiguration(attachment)
if err != nil {
    panic(err.Error())
}
C.releaseNSObject(attachment.pointer)
C.releaseNSObject(config.pointer)
```

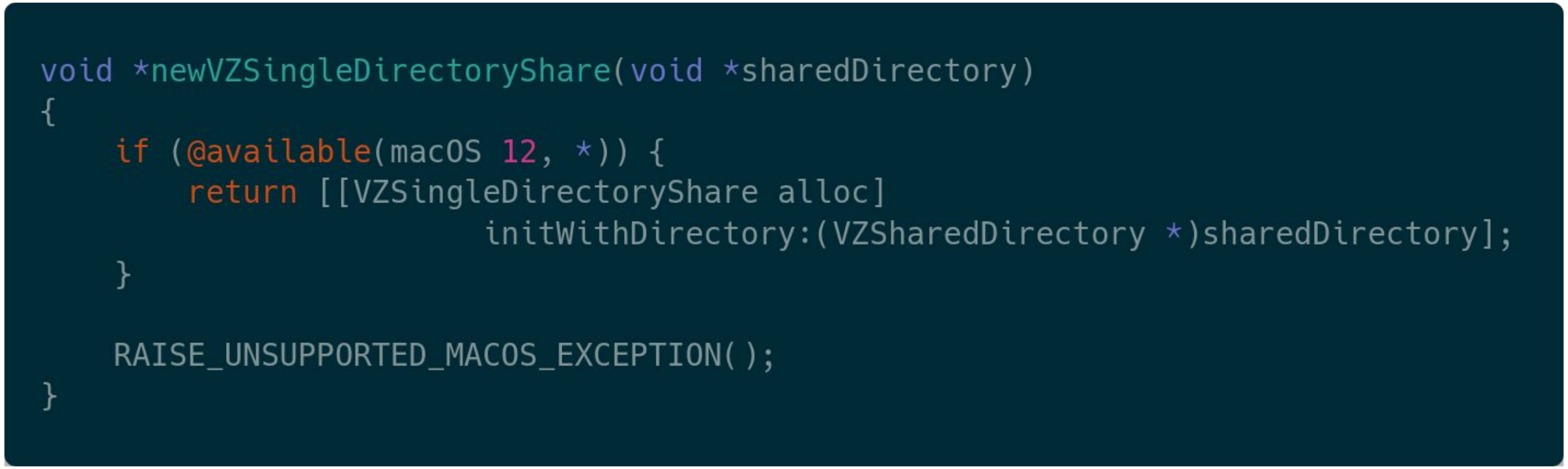
Memory management

- Link to Apple's memory management policy for objective-C
- You own any object you create
  - You create an object using a method whose name begins with `“alloc”`, `“new”`, `“copy”`, or `“mutableCopy”`
- You can take ownership of an object using `retain`
- When you no longer need it, you must relinquish ownership of an object you own
- You must not relinquish ownership of an object you do not own

Other Objective-C features

- `cgo.Handle` is useful for delegates
- Objective-C blocks
- Exceptions

API availability check with `@available`



```
void *newVZSingleDirectoryShare(void *sharedDirectory)
{
    if (@available(macOS 12, *)) {
        return [[VZSingleDirectoryShare alloc]
                        initWithDirectory:(VZSharedDirectory *)sharedDirectory];
    }

    RAISE_UNSUPPORTED_MACOS_EXCEPTION();
}
```

- To be used in combination with `-mmacosx-version-min=11`

Questions?

https://github.com/cfergeau/go-objc

github.com/cfergeau

cfergeau@redhat.com

Red Hat