Leveraging Go to Build a FaaS Platform

Brian Downs
2018

Who Am I?

1. Software Engineer
2. GolangPhoenix and BSD PHX Co-Organizer
3. Amatuer cook and gardener

Function as a Service

It’s all the rage. AWS, Google, Azure, and a number of others are offering these services.

There are also do it yourself FaaS platforms. OpenFaaS is a great example and Hashicorp Nomad can be a FaaS depending on config and use.

Why?

What Do We Need to Build Our FaaS Platform?

Core Components

- FreeBSD Jails
- ZFS
- Go

FreeBSD Jails

What is a FreeBSD?

- Free and open source operating system under the BSD Licensing
- Direct descendant from BSD UNIX
- Serves the majority of traffic on the internet
- Supports native ZFS, DTrace

Why not Linux?

What is a Jail?

- Kernel built-in
- Operating system virtualization
- Process segregation
- Contains its own users and

Jail Features

- provide the abstractions necessary to provide a secure execution environment for long and short lived processes
- allow for simple and complex networking
- simple to use

Common Uses for Jails

- Web stacks
  - Apache / Nginx
  - MySQL / Postgres
- Shell environments
- Network Services

Let’s take a look at a simple Jail.


ZFS
The Last Word in File Systems

ZFS

- Snapshots
- Instant Cloning
- Extremely fast

A look at a few ZFS commands

Snapshots

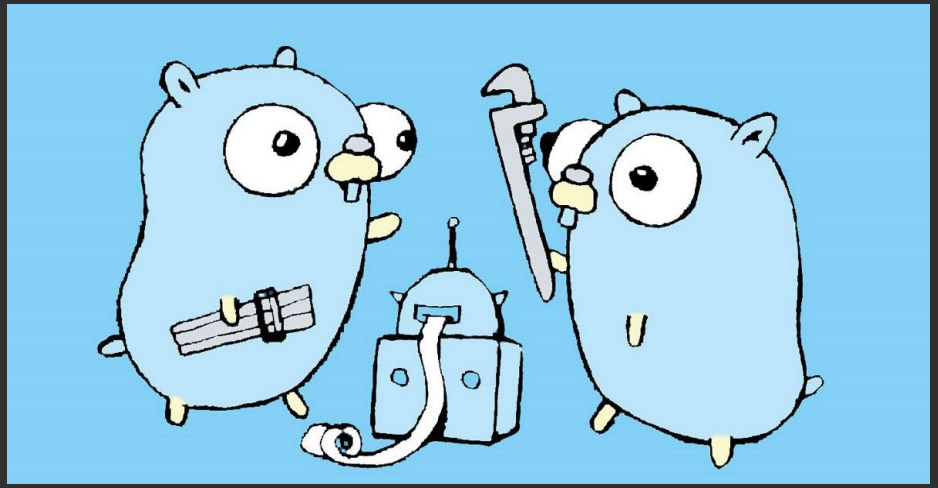
- Read-only copy of a dataset
- Near instant creation
- No disk space use unless data starts changing

Clones

- Read/write volumes
- Created from snapshots
- Near instant creation
- No disk space use unless data starts changing


GO

Why Go?



- Ability to iterate quickly
- Extremely fast compilations
- Build complex systems easily
- Extensive standard library

Go Components

- net/http
- text/template
- syscall
- Go compiler

DEMO!

Demo continued...

http://demo.skyisland.io:3280

Demo continued...

```
curl --silent -XPOST

http://demo.skyisland.io:3280/api/v1/function

-d '{"url": "github.com/briandowns/smile", "call": "Smile()"}'
```

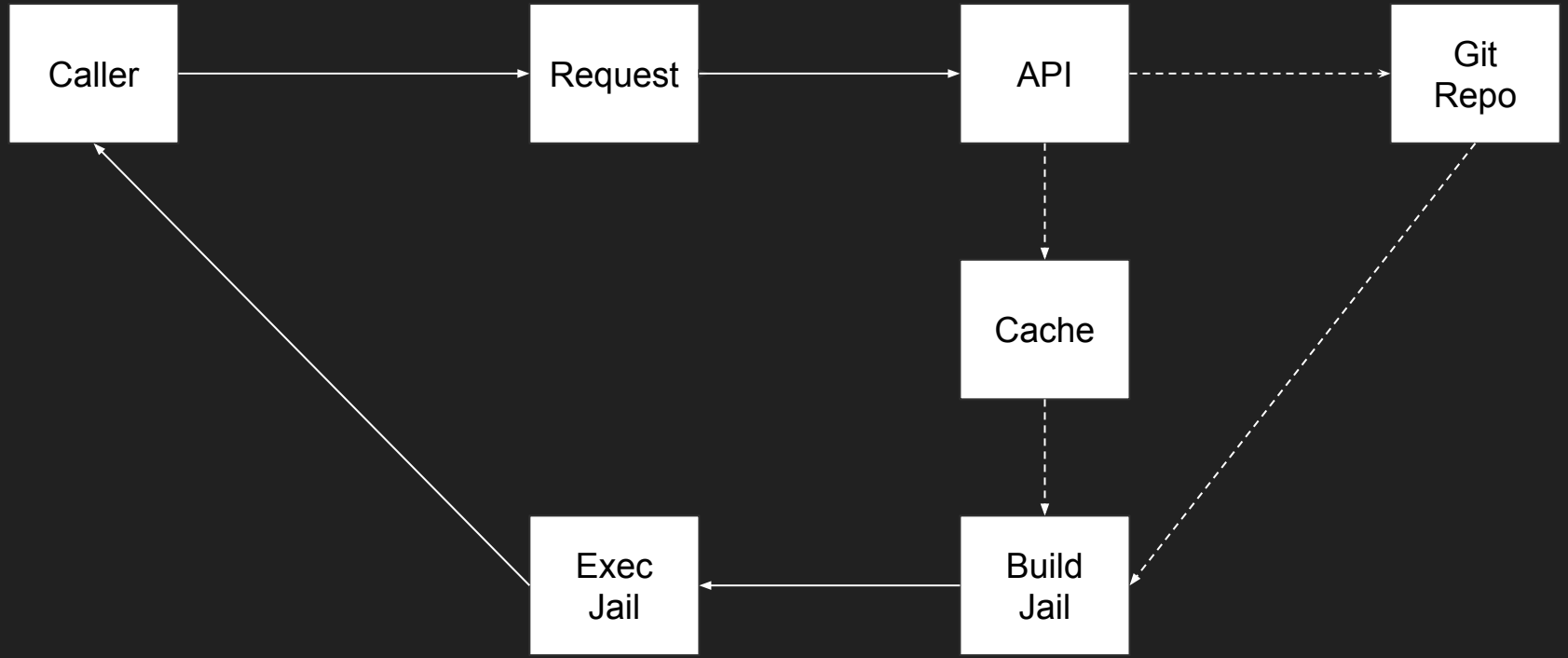
Demo continued...

{"timestamp":1527201010,"data":":)"}

Demo continued...

What all just happened?

Workflow



Build Your Own Function!

Let’s Build Our FaaS System!

Questions?

FaaS Components

- IP Management
- HTTP JSON REST API
- Cache Layer
- Client Code

IP Management

DHCP

Pros

DHCP continued...

- We don’t have to give a ****. It just works.

Cons

- We have to enable Raw Sockets
- Likely more functionality than necessary

IP Allocator

Pros

IP allocator continued...

- We don’t have to give a ****. It just works.

Cons

DHCP continued...

- We have to write an IP allocator...
- We have to manage a pool of IP’s.

To the code!

Further Considerations

IP Allocator continued...

- Use VNET

API

Health Check

API continued...

- GET - /healthcheck
  - Provide back an HTTP 200
  - Include git sha in response

Function Endpoint

API continued...

- POST - /api/v1/function
  - Trigger a function execution

Jail Operations

API continued...

- GET - /api/v1/admin/jails
  - Get a list of the running jails
- GET - /api/v1/admin/jail/{id}
  - Get the details for the given jail
- DELETE - /api/v1/admin/jail/{id}
  - Kill the jail with the given ID
- DELETE - /api/v1/admin/jails
  - Kill all jails

To the code!

Further Considerations

API continued...

- Rate Limiting
- Circuit Breaking
- Additional Authentication Methods

syscalls

What is a syscall?

- Short for system call
- A means of requesting kernel services
- Represented by an integer value
- 99% of O.S.’s provide wrappers in libc

Why do they matter?

Cache

Caching Strategies

- Keep git repo in build jail
- Save compiled binaries
- Save results from previously seen requests

Git Repo Cache

Cache continued...

- Depth 1 clone
- Stored in the build jail
- Reused for each build

Binary Cache

Cache continued...

- Keep compiled binaries somewhere they can be accessed quickly for execution

```
type BinaryCache struct {
	mu    sync.RWMutex
	cache map[string]string
}
```

Request Cache

Cache continued...

- Store the previous requests
- Prevent from hitting any jail process

To the code!

Further Considerations

Cache continued...

- Redis
- DynamoDB

Reusable Code

Client Libraries for Consuming Existing Code

- Go
- Python
- C
- Curl
- Javascript (soon (maybe))

Go

Client continued...

```
func main() {
	timeout := time.Second * 10
	client := skyisland.NewClient("http://demo.skyisland.io", 3280, timeout)
	res, err := client.Function("github.com/briandowns/smile", "Smile()")
	if err != nil {
		fmt.Println(err)
		os.Exit(1)
	}
	fmt.Printf("%+v\n", res.Data)
}

// :)
```

Python

Client continued...

```
client = Client("demo.skyisland.io", 3280)
data = client.function("github.com/mmcloughlin/geohash", "Encode(100.1, 80.9)")
print(json.loads(data)[‘data’])

# jcc92ytsf8kn
```

C

Client continued...

```
struct client_t *c = malloc(sizeof(struct client_t));

c->endpoint = "http://demo.skyisland.io:3280/api/v1/function";

char *url = "github.com/mmcloughlin/geohash";
char *call = "Encode(100.1, 80.9)";

res = function(c, url, call);
printf(“%s\n”, res->data);
free(c);
return 0;

// jbcs8zmd6shp
```

Curl

Client continued...

```
curl --silent \
    -XPOST http://demo.skyisland.io:3280/api/v1/function \
    -d ‘{
            “url”: “github.com/mmcloughlin/geohash”,
            “call”: “Encode(100.1, 80.9)”
        }’

# {“timestamp”: 1527193906, “data”: “jbcs8zmd6shp”}
```

To the code!

Further Considerations

- Decouple the build process from the exec process
- Allow for building binaries from a given Git hash
- ?

Where do we go next?

- Interpreter Embedding
  - Lua
  - Javascript
  - Python
- Clustering
- Cloud Provider Integrations
- Scheduled Function Execution
- Different Jail types

Conclusion

Through the use of ephemeral jails, a simple API, ZFS, and Go primitives, we can derive platforms that allow for the creation of dynamic workflows further extending the capabilities of distributed code execution.

Projects & Example Code

- FaaS Platform:
  - github.com/briandowns/sky-island

- FreeBSD Jail Syscall Package:
  - github.com/briandowns/jail

Questions?

Thank You

- John Moore
- Devin Teske
- Josh Baker
- Brandon Gibson
- Jeremy Maldonado

Brian Downs

github.com/briandowns

@bdowns328