CODE2842LV

vmware EXPLORE

Go Start Your Horizon Automation Journey

Sean Massey (He/Him)

Staff Multi-Cloud Solutions Architect

#vmwareexplore #CODE2842LV

Introductions

Hi…I’m Sean…

Staff Multi-Cloud Solutions Architect – VMware

I do “Cloudy EUC Things” with a focus on Multi-Cloud and Service Providers

VCDX-DTM #247

Live in Kimberly, WI

Blog: http://thevirtualhorizon.com

Mastodon: @seanpmassey@vmst.io

Instagram/Threads: seanpmassey

LinkedIn: https://www.linkedin.com/in/seanpmassey

Agenda

The Go Programming Language

Automating Horizon – A Little History

Getting Started with the Horizon REST API

Our Test Cases

- Getting Connection Server Information
- Finding Persistent Desktop Sessions that have been idle for two days and logging them out

Building Our Tools

- Testing Our API Calls and Workflow
- Creating Our Go Project
- Writing and Testing Our Tool

I’m not an expert in Go – I’ve been learning this as I *go* along

All code for this session is available on Github. The code is not the cleanest.

Why the Go Programming Language?

What is the Go Programming Language?

Go Programming Language Website: https://go.dev/

Statically-typed, high-level compiled programming language

- Data type of the variable is known at compile time
- Programmer must specify data type when declaring variable

Influenced by C and Python

Designed for Simplicity and Safety

Open Source and Supported by Google

Used in some small projects you've probably never heard of

Projects written in Go

- Docker
- Kubernetes
- Terraform
- Prometheus
- Grafana

Why Am I Using Go?

Starting Using Go to Write Tools for Managing My Kids Minecraft Servers

Picked it for a few reasons:

Low barrier to entry

- Works with VSCode
- Fairly easy to learn
- Module System that reminds me of PowerShell
- Easy Modularity

Compiled language – binaries don't require any prerequisites

- I know this can be done with Python and other languages, but there are no runtimes or additional tooling required to make multi-platform binaries

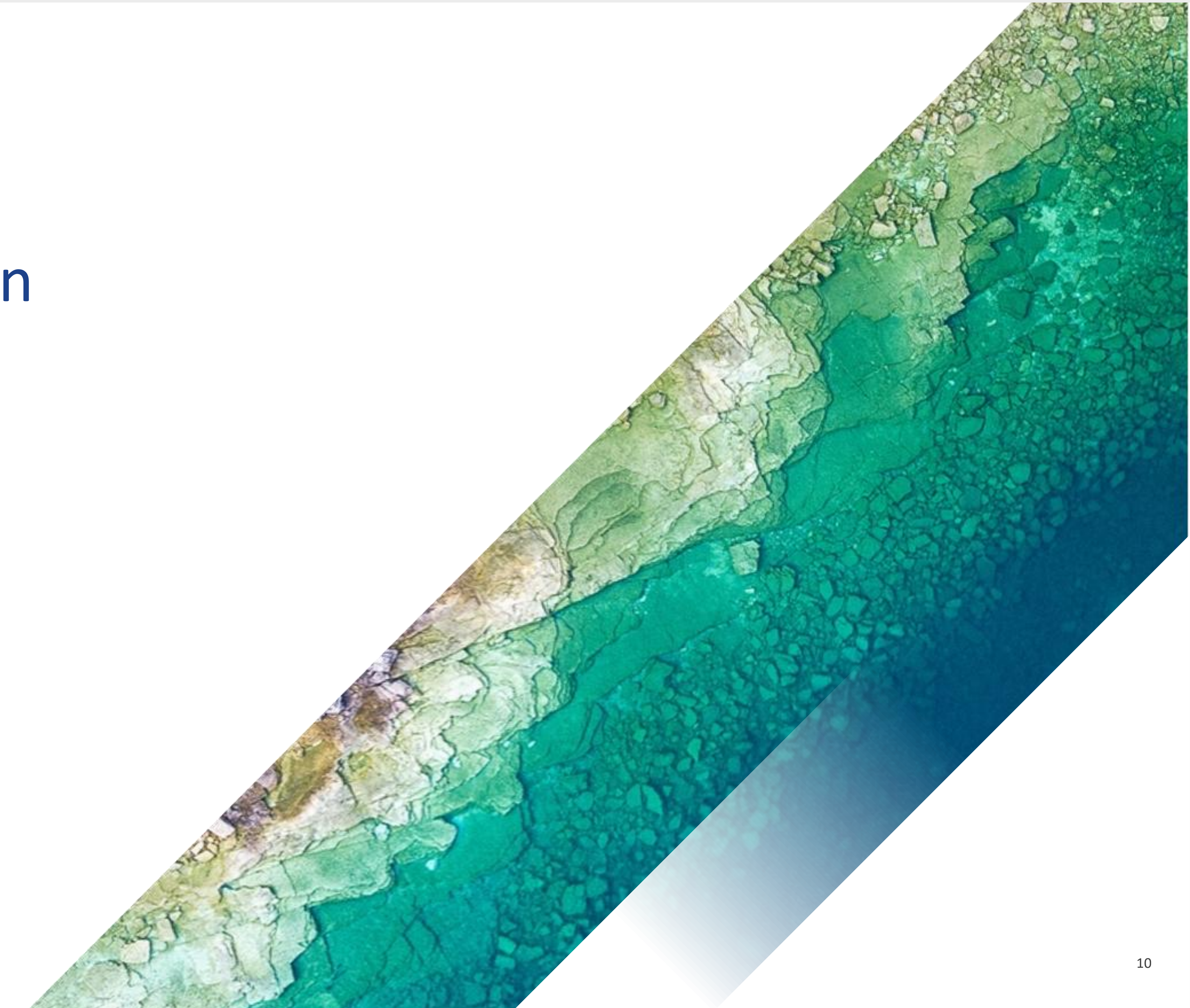
Multi-Platform – Can easily compile code for multiple operating systems and architectures

Seems easy to transition from a CLI tool to something web or API-based

Use the language that fits you and your business or technical needs

Automating Horizon

A Brief History



Horizon has not always been the easiest platform to automate

Horizon Automation Tools

Some are still used in various places

vdmadmin

A Windows CLI command to perform tasks that can’t be performed in the Admin UI

Limited in what it could do

Still used for some tasks includingKiosk Mode, enabling Login Profiler, removing failed connection servers and more.

https://docs.vmware.com/en/VMware-Horizon-7/7.13/horizon-administration/GUID-D66A2341-E672-48CC-8D19-16EB2285CEEF.html

Horizon PowerCLI

PowerCLI wrapper for the Horizon View API

Powerful and low-level access

Hard to consume

Community module that attempted to make consuming this easier

ADAM Database

Unsupported!!!!

Directly manipulating entries in the Horizon ADAM database that is shared between connection servers

Dangerous and not recommended because it can, and will, break things

Did I mention it’s unsupported?

Only mentioning this because it has been used to automate Horizon in the past, and you will find scripts that use this method…but I wouldn’t know anything about that…

The Horizon REST API addresses many automation challenges

What is the Horizon REST API

Versioned API that returns JSON objects

Allows admins to automate most tasks

VDI

Introduced in Horizon 7.10

Constantly being updated and iterated

Getting Started with Horizon Automation

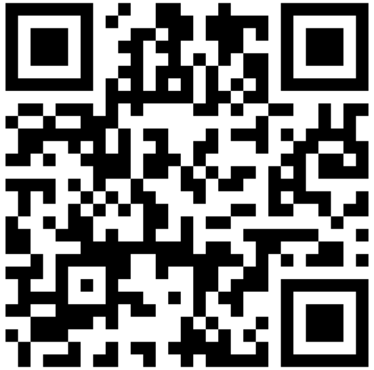
Using the Horizon REST API

Horizon REST API Resources

Key Resources to learn the API – In QR Code Format

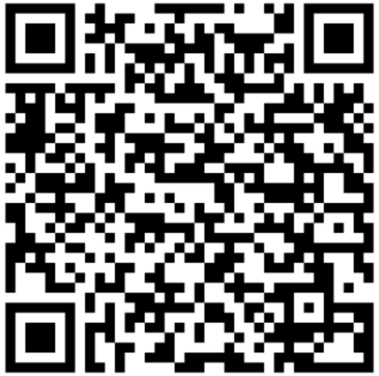
1

VMware Code Site



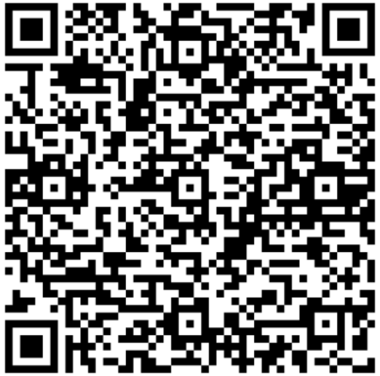
2

Horizon Postman Collection



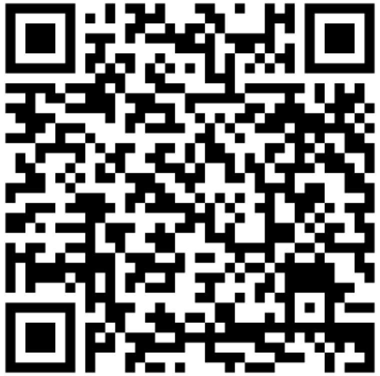
3

Pagination and Filter Guide (Word doc)



4

Techzone API Getting Started Guide



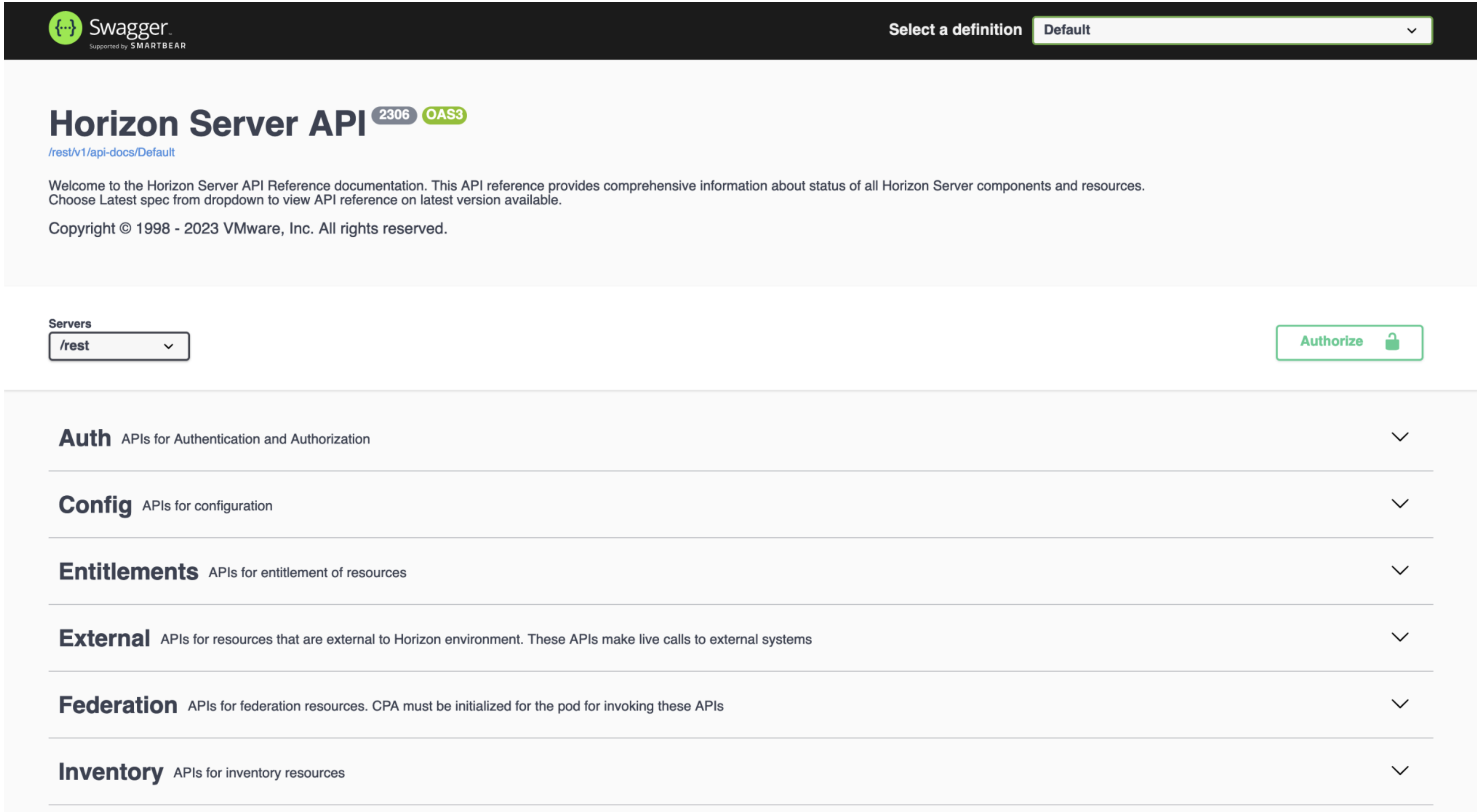
5

Local Swagger Page

Each Connection Server has a local swagger page

Connection Server Swagger Page

https://<fqdn-of-connectionserver>/rest/swagger-ui.html



Select a definition: Default

Horizon Server API 2306 OAS3

/rest/v1/api-docs/Default

Welcome to the Horizon Server API Reference documentation. This API reference provides comprehensive information about status of all Horizon Server components and resources.
Choose Latest spec from dropdown to view API reference on latest version available.

Copyright © 1998 - 2023 VMware, Inc. All rights reserved.

Servers: /rest

Auth APIs for Authentication and Authorization

Config APIs for configuration

Entitlements APIs for entitlement of resources

External APIs for resources that are external to Horizon environment. These APIs make live calls to external systems

Federation APIs for federation resources. CPA must be initialized for the pod for invoking these APIs

Inventory APIs for inventory resources

Horizon REST API Endpoints

Seven Endpoints

Auth – APIs Authentication and Authorization. Used to log in and out of Horizon

Config – APIs for managing Horizon configuration items including Connection Server general settings and Image Management Service

Entitlements – APIs for managing user entitlements to desktop and application pools

External – APIs for managing systems external to Horizon. This includes vCenter and Active Directory integrations

Federation – APIs for managing Cloud Pod Architecture

Inventory – Information for managing desktop and application pools including managing pools, machines, and sessions and performing CRUD operations

Monitor – Monitoring APIs. These are all GET calls

Authenticating to the Horizon REST API

Horizon REST API uses JSON Web Token (JWT) for authentication and authorization

Login call is a POST call to the following URL:

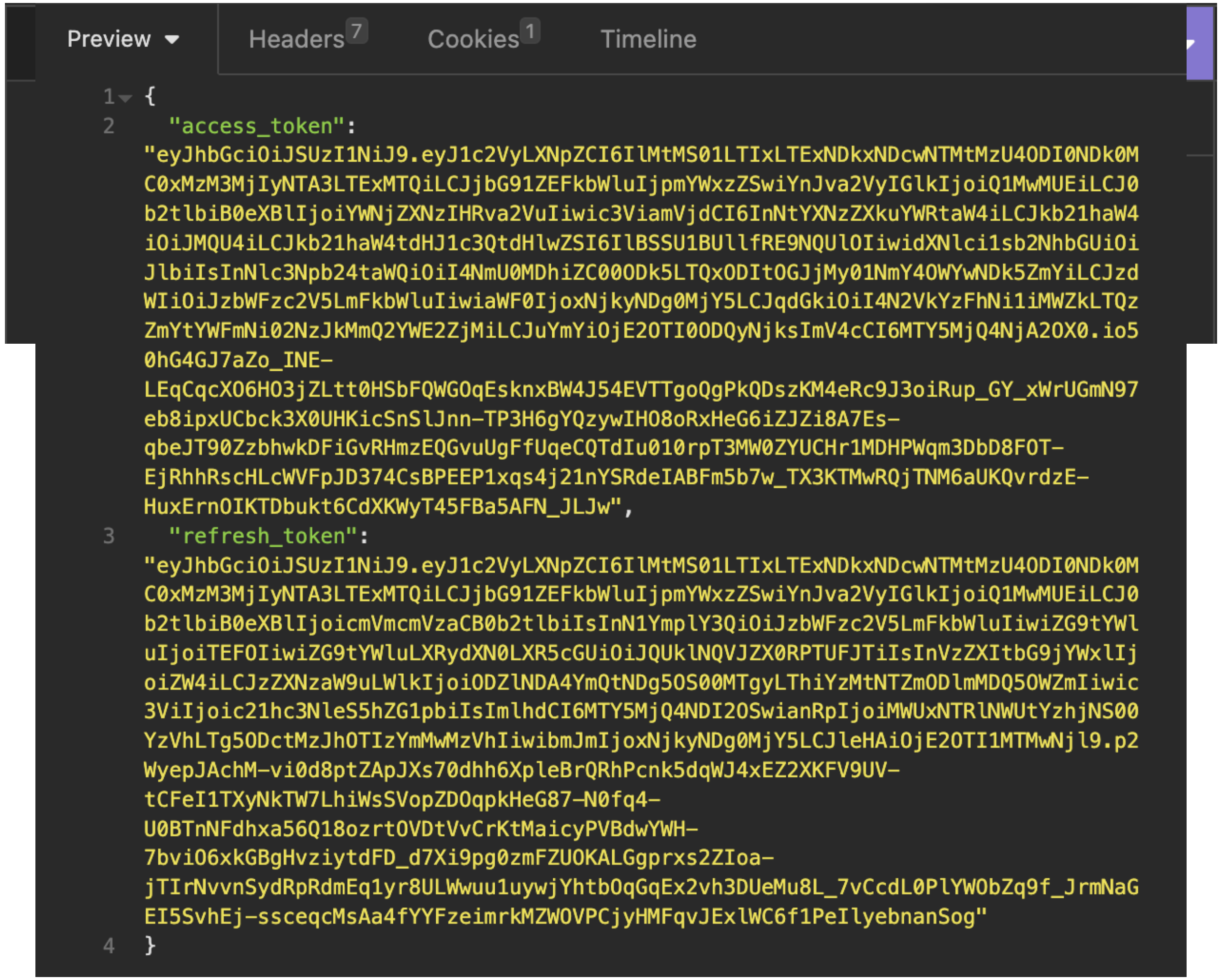
- https://**fqdn-of-connectionserver/**rest/login

JSON body that must contain the username, password, and domain

Returns an access_token and refresh_token

- Access_Token is good for 30 minutes and must be attached to all calls as a bearer token in the authentication header
- Refresh_Token is valid for 8 hours used to generate a new access_token if the current one expires

Can change the default token lifetimes by editing the Connection Server ADAM database…



```
{
  "access_token":
"eyJhbGciOiJSUzI1NiJ9.eyJ1c2VyLXNpZCI6IlMtMS01LTIxLTExNDkxNDcwNTMtMzU4ODI0NDk0M
C0xMzM3MjIyNTA3LTExMTQiLCJjbG91ZEFkbWluIjpmYWxzZSwiYnJva2VyIGlkIjoiQ1MwMUEiLCJ0
b2tlbiB0eXBlIjoiYWNjZXNzIHRva2VuIiwic3ViamVjdCI6InNtYXNzZXkuYWRtaW4iLCJkb21haW4
iOiJMQU4iLCJkb21haW4tdHJ1c3QtdHlwZSI6IlBSSU1BUllfRE9NQUlOIiwidXNlci1sb2NhbGUiOi
JlbiIsInNlc3Npb24taWQiOiI4NmU0MDhiZC00ODk5LTQxODItOGJjMy01NmY4OWYwNDk5ZmYiLCJzd
WIiOiJzbWFzc2V5LmFkbWluIiwiaWF0IjoxNjkyNDg0MjY5LCJqdGkiOiI4N2VkYzFhNi1iMWZkLTQz
ZmYtYWFmNi02NzJkMmQ2YWE2ZjMiLCJuYmYiOjE2OTI0ODQyNjksImV4cCI6MTY5MjQ4NjA2OX0.io5
0hG4GJ7aZo_INE-
LEqCqcXO6HO3jZLtt0HSbFQWGOqEsknxBW4J54EVTTgoQgPkQDszKM4eRc9J3oiRup_GY_xWrUGmN97
eb8ipxUCbck3X0UHKicSnSlJnn-TP3H6gYQzywIHO8oRxHeG6iZJZi8A7Es-
qbeJT90ZzbhwkDFiGvRHmzEQGvuUgFfUqeCQTdIu010rpT3MW0ZYUCHr1MDHPWqm3DbD8FOT-
EjRhhRscHLcWVFpJD374CsBPEEP1xqs4j21nYSRdeIABFm5b7w_TX3KTMwRQjTNM6aUKQvrdzE-
HuxErnOIKTDbukt6CdXKWyT45FBa5AFN_JLJw",
  "refresh_token":
"eyJhbGciOiJSUzI1NiJ9.eyJ1c2VyLXNpZCI6IlMtMS01LTIxLTExNDkxNDcwNTMtMzU4ODI0NDk0M
C0xMzM3MjIyNTA3LTExMTQiLCJjbG91ZEFkbWluIjpmYWxzZSwiYnJva2VyIGlkIjoiQ1MwMUEiLCJ0
b2tlbiB0eXBlIjoicmVmcmVzaCB0b2tlbiIsInN1YmplY3QiOiJzbWFzc2V5LmFkbWluIiwiZG9tYWl
uIjoiTEFOIiwiZG9tYWluLXRydXN0LXR5cGUiOiJQUklNQVJZX0RPTUFJTiIsInVzZXItbG9jYWxlIj
oiZW4iLCJzZXNzaW9uLWlkIjoiODZlNDA4YmQtNDg5OS00MTgyLThiYzMtNTZmODlmMDQ5OWZmIiwic
3ViIjoic21hc3NleS5hZG1pbiIsImlhdCI6MTY5MjQ4NDI2OSwianRpIjoiMWUxNTRlNWUtYzhjNS00
YzVhLTg5ODctMzJhOTIzYmMwMzVhIiwibmJmIjoxNjkyNDg0MjY5LCJleHAiOjE2OTI1MTMwNjl9.p2
WyepJAchM-vi0d8ptZApJXs70dhh6XpleBrQRhPcnk5dqWJ4xEZ2XKFV9UV-
tCFeI1TXyNkTW7LhiWsSVopZDOqpkHeG87-N0fq4-
U0BTnNFdhxa56Q18ozrtOVDtVvCrKtMaicyPVBdwYWH-
7bviO6xkGBgHvziytdFD_d7Xi9pg0zmFZUOKALGgprxs2ZIoa-
jTIrNvvnSydRpRdmEq1yr8ULWwuu1uywjYhtbOqGqEx2vh3DUeMu8L_7vCcdL0PlYWObZq9f_JrmNaG
EI5SvhEj-ssceqcMsAa4fYYFzeimrkMZWOVPCjyHMFqvJExlWC6f1PeIlyebnanSog"
}
```

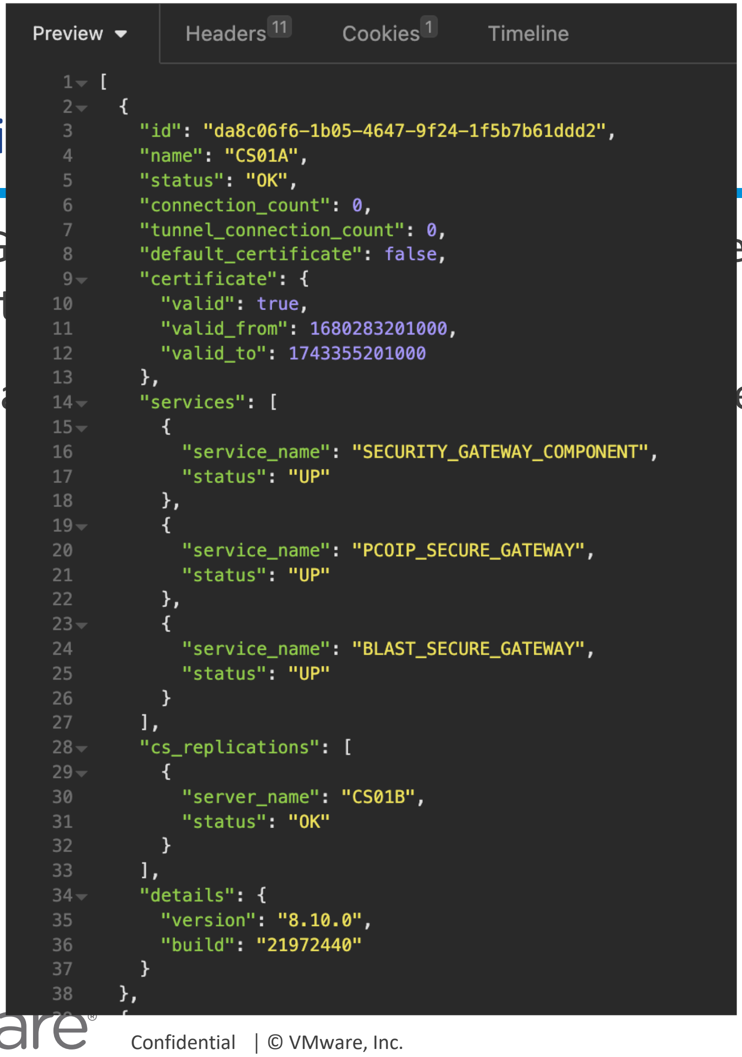
Using the Horizon REST API

Showcase Two Easy (-ish) Use Cases

Retrievi

Simple G ervers
endpoint

Returns erver
info



```
[
  {
    "id": "da8c06f6-1b05-4647-9f24-1f5b7b61ddd2",
    "name": "CS01A",
    "status": "OK",
    "connection_count": 0,
    "tunnel_connection_count": 0,
    "default_certificate": false,
    "certificate": {
      "valid": true,
      "valid_from": 1680283201000,
      "valid_to": 1743355201000
    },
    "services": [
      {
        "service_name": "SECURITY_GATEWAY_COMPONENT",
        "status": "UP"
      },
      {
        "service_name": "PCOIP_SECURE_GATEWAY",
        "status": "UP"
      },
      {
        "service_name": "BLAST_SECURE_GATEWAY",
        "status": "UP"
      }
    ],
    "cs_replications": [
      {
        "server_name": "CS01B",
        "status": "OK"
      }
    ],
    "details": {
      "version": "8.10.0",
      "build": "21972440"
    }
  },
```

Logging Out Stale Sessions on a Persistent Pool

Multi-Step Process targeting 3 API Calls

- GET /inventory/v7/desktop-pools – find persistent desktop pools
- GET /inventory/v1/sessions – find all disconnected sessions for this desktop pool
- POST /inventory/v1/sessions/action/logoff – Log off sessions that meet our criteria

Will require some advanced filtering

Prerequisites

API Testing Tool like Postman or Insomnia

Postman Collection

Non-Production Horizon Environment

User Account with permissions to perform the actions we want to automate

After testing the API calls, we'll look at how to create a Go app using them

Demo Time!

Consuming the Horizon REST API

Using the Go Programming Language

New Go Project:
Browse to your project folder
go mod init

This blog provides a great overview for creating a new Go project.

Using REST APIs with Go

net/http Module

Built-in Golang Module

Provides both http client and server features

Two primary ways to perform REST calls

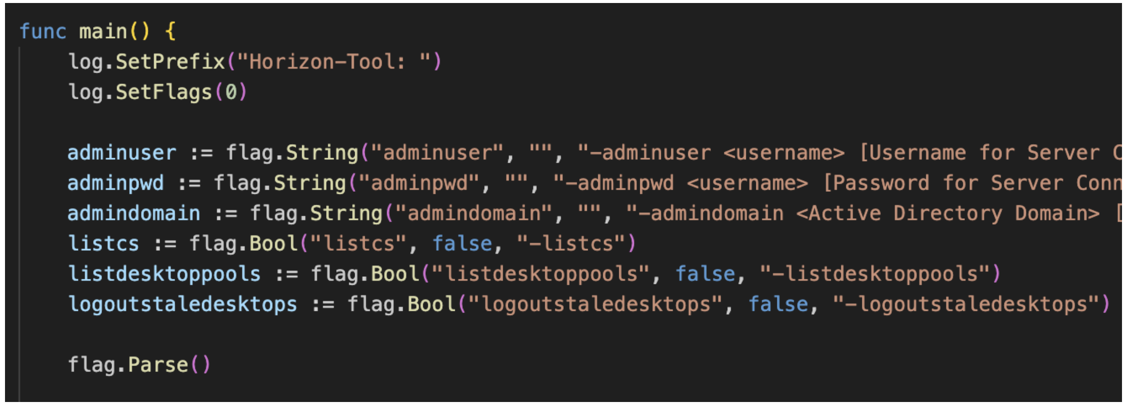
- http.newrequest – requires programmer to declare a new client, but allows headers to be set
- http.get and http.post – does not require a new client, but can't set headers

Using both techniques in different places

- http.get and http.post used for authentication calls
- http.newrequest used for all other requests to add required headers

There are 3rd-Party modules that may make it easier, but I'm not using these right now so I can understand how Go does things

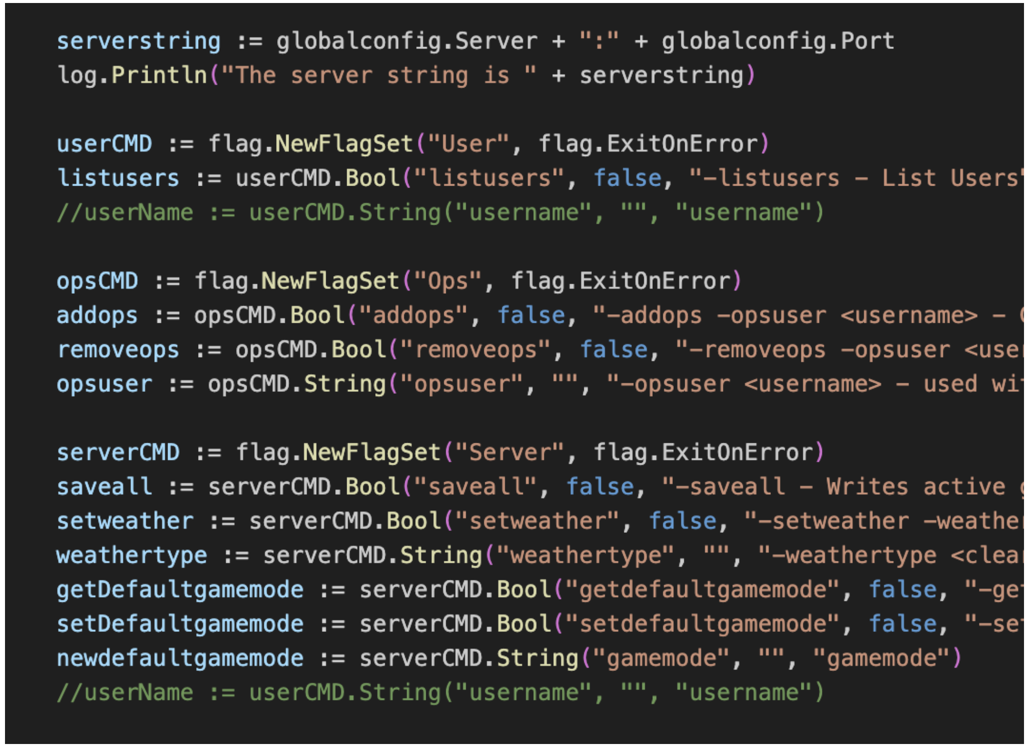
CLI Parameters



```
func main() {
    log.SetPrefix("Horizon-Tool: ")
    log.SetFlags(0)

    adminuser := flag.String("adminuser", "", "-adminuser <username> [Username for Server C
    adminpwd := flag.String("adminpwd", "", "-adminpwd <username> [Password for Server Conn
    admindomain := flag.String("admindomain", "", "-admindomain <Active Directory Domain> [
    listcs := flag.Bool("listcs", false, "-listcs")
    listdesktoppools := flag.Bool("listdesktoppools", false, "-listdesktoppools")
    logoutstaledesktops := flag.Bool("logoutstaledesktops", false, "-logoutstaledesktops")

    flag.Parse()
```



```
    serverstring := globalconfig.Server + ":" + globalconfig.Port
    log.Println("The server string is " + serverstring)

    userCMD := flag.NewFlagSet("User", flag.ExitOnError)
    listusers := userCMD.Bool("listusers", false, "-listusers - List Users
    //userName := userCMD.String("username", "", "username")

    opsCMD := flag.NewFlagSet("Ops", flag.ExitOnError)
    addops := opsCMD.Bool("addops", false, "-addops -opsuser <username> - 
    removeops := opsCMD.Bool("removeops", false, "-removeops -opsuser <use
    opsuser := opsCMD.String("opsuser", "", "-opsuser <username> - used wi

    serverCMD := flag.NewFlagSet("Server", flag.ExitOnError)
    saveall := serverCMD.Bool("saveall", false, "-saveall - Writes active 
    setweather := serverCMD.Bool("setweather", false, "-setweather -weathe
    weathertype := serverCMD.String("weathertype", "", "-weathertype <clea
    getDefaultgamemode := serverCMD.Bool("getdefaultgamemode", false, "-ge
    setDefaultgamemode := serverCMD.Bool("setdefaultgamemode", false, "-se
    newdefaultgamemode := serverCMD.String("gamemode", "", "gamemode")
    //userName := userCMD.String("username", "", "username")
```

Native Flags module to create CLI Parameters

Allows me to create one tool to handle multiple tasks

- Go's easy modularity enables this

Flags are declared at the beginning of the main function of main.go

Two ways to do Flags

- Flags – individual command line parameters
  flags.parse and if statements
- Flag Sets – groups of command line parameters
  os.Args[position] and select case

Haven't found a good way to integrate flags and flag sets

Working with JSON Results

Go has a concept of marshalling and unmarshalling JSON

- This is Go’s terminology for converting to and from JSON

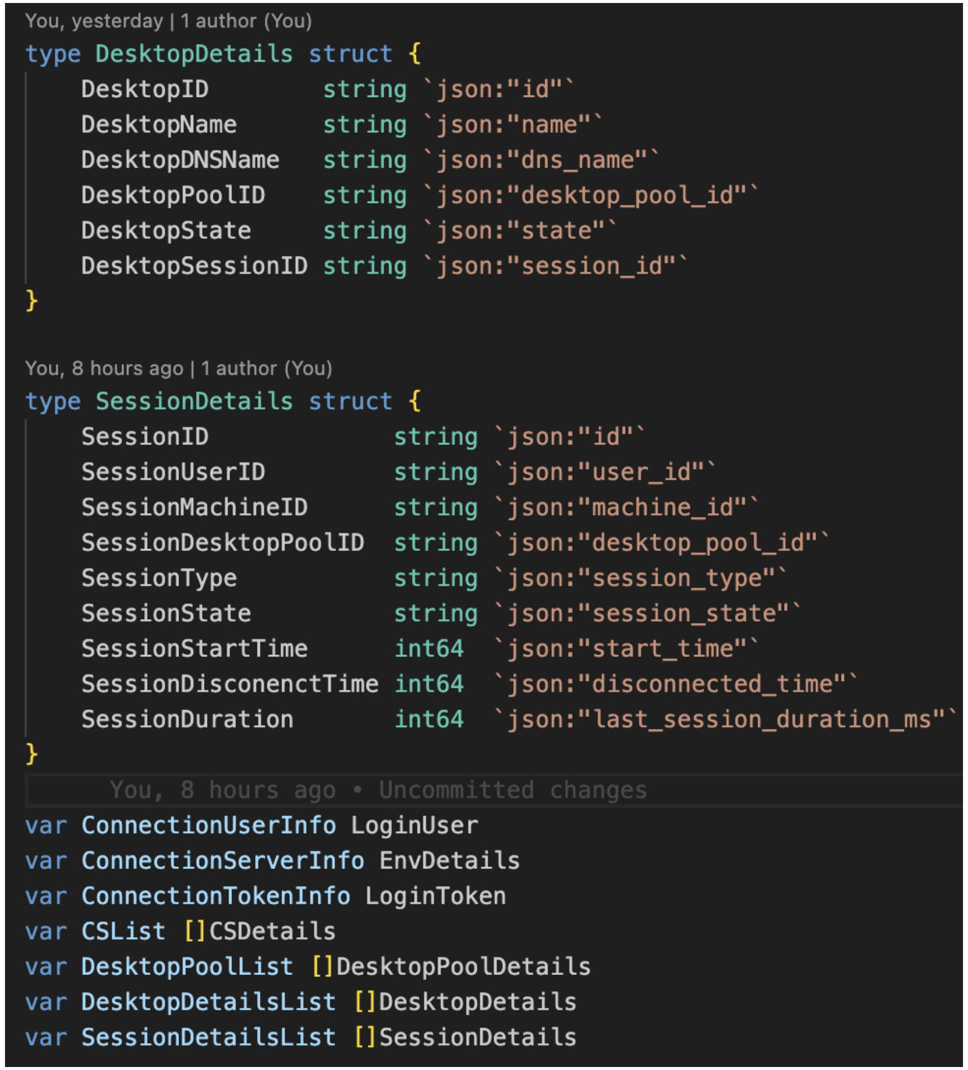
Go cannot dynamically unmarshal JSON objects

Manually create data structures - struct objects

Struct object is basically a grouping of variables

Map specific JSON fields to elements of a data structure

To use data structures in code, I must declare a variable of that data structure



```
You, yesterday | 1 author (You)
type DesktopDetails struct {
    DesktopID        string `json:"id"`
    DesktopName      string `json:"name"`
    DesktopDNSName   string `json:"dns_name"`
    DesktopPoolID    string `json:"desktop_pool_id"`
    DesktopState     string `json:"state"`
    DesktopSessionID string `json:"session_id"`
}

You, 8 hours ago | 1 author (You)
type SessionDetails struct {
    SessionID             string `json:"id"`
    SessionUserID         string `json:"user_id"`
    SessionMachineID      string `json:"machine_id"`
    SessionDesktopPoolID  string `json:"desktop_pool_id"`
    SessionType           string `json:"session_type"`
    SessionState          string `json:"session_state"`
    SessionStartTime      int64  `json:"start_time"`
    SessionDisconenctTime int64  `json:"disconnected_time"`
    SessionDuration       int64  `json:"last_session_duration_ms"`
}
      You, 8 hours ago • Uncommitted changes
var ConnectionUserInfo LoginUser
var ConnectionServerInfo EnvDetails
var ConnectionTokenInfo LoginToken
var CSList []CSDetails
var DesktopPoolList []DesktopPoolDetails
var DesktopDetailsList []DesktopDetails
var SessionDetailsList []SessionDetails
```

Horizon API and Time Conversion

We need to evaluate time values to determine which desktops get logged out

Horizon API uses Unix/Epoch time

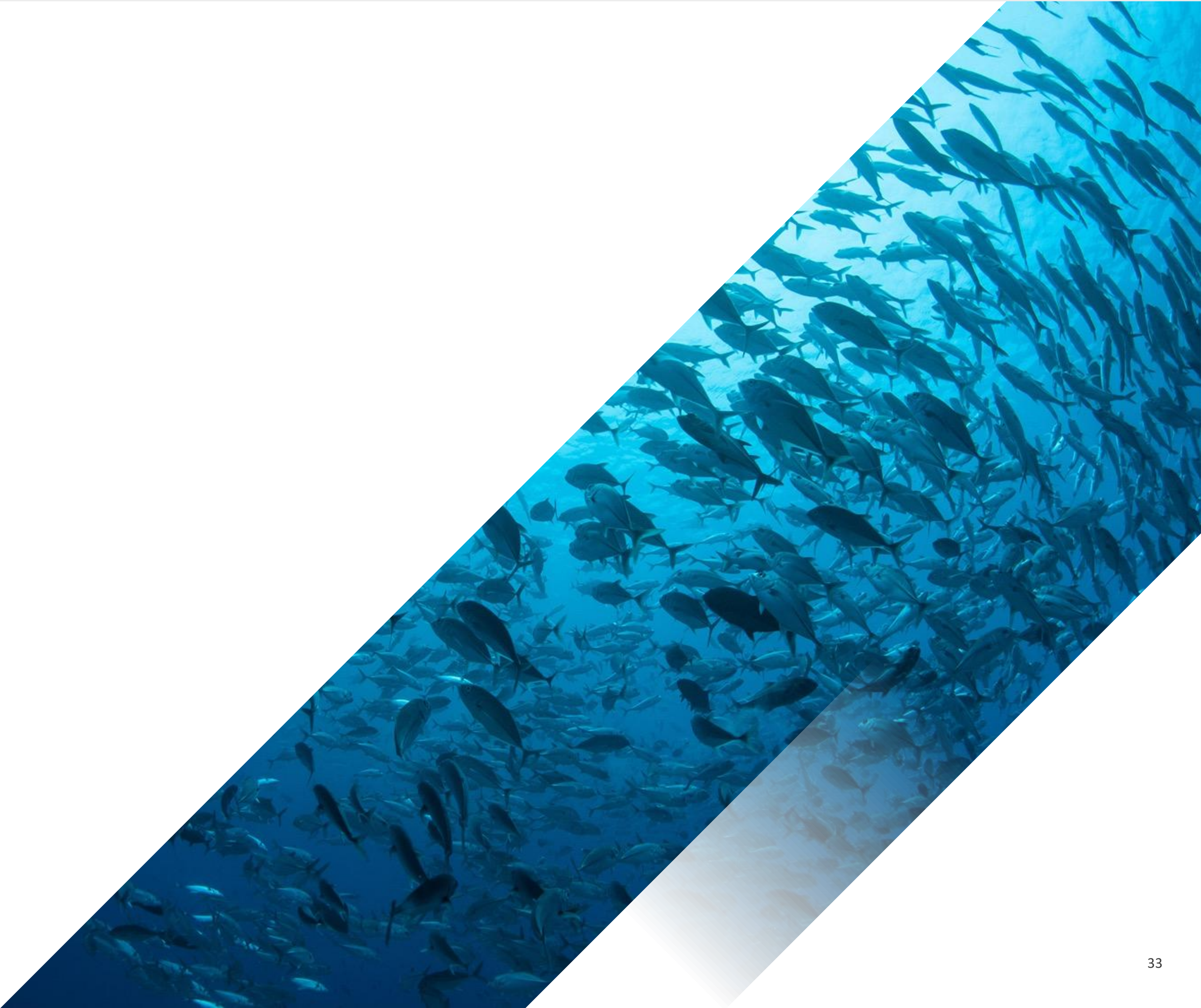
Go has a method for converting Unix time to standard time – time.Unix()

I struggled a lot with this, and I almost didn’t get this demo working because Go’s default method accepts seconds or nanoseconds, and Horizon returns microseconds

There is a new method for this – time.UnixMicro() that converts properly

Code Walkthrough

YOLO!!!



Where I Want to Go From Here

And Things to Try in the Future

Get more use cases enabled

- Automating Pool Creation and Image Push Operations
- Entitlements

Get the CLI Flags/Parameters working better

More modularity

Web/API Front-End

- Manage Multiple Horizon Deployments

3rd-Party Modules

- RESTY – 3rd-party REST API Client
- gjson – 3rd-party JSON module

All code for this session is available on Github. The code is not the cleanest.

vmware EXPLORE

Please take your survey.

vmware® EXPLORE

Thank You