BIOTEAM
Enabling Science

Building Data Systems with iRODS and Golang

John Jacquay | john@bioteam.net

BIOTEAM
Enabling Science

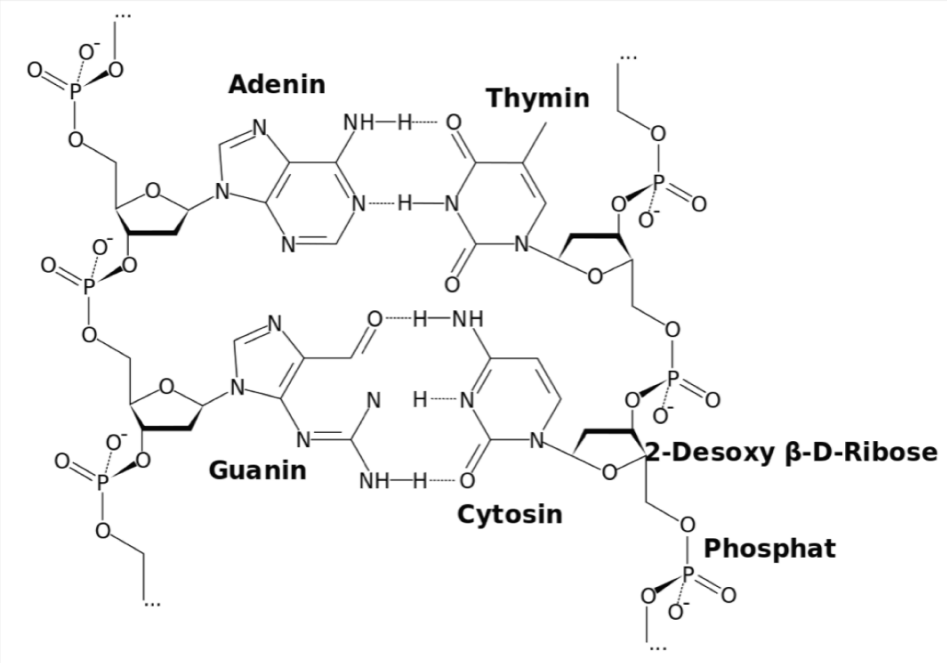
Presentation Road Map

- My vision of an iRODS data system in the domain of life sciences
  - Data formats & automated bioinformatics pipelines
  - Tiered storage
  - GUI for automation design, data discovery, delivery
  - Open-source
- Why Golang?
- Integration Points and Use Cases
  - Client – A deep dive into GoRODS
  - Microservices and iRODS DSL
  - Rule Engine
  - RPC API: Client & Server
- That's it!

BIOTEAM
Enabling Science

iRODS In The Domain of Life Sciences

DNA Sequencing



BIOTEAM
Enabling Science

iRODS In The Domain of Life Sciences

File formats for High-Throughput sequencing

- Instrument generates BCL
- Utility translates raw BCL to FASTQ
- QC is run
- Cleaning / trimming takes place
- Assembly, Alignment
- FASTQ to BAM/SAM
- BAM/SAM -> diff -> VCF

- Raw: BCL
- Primary: FASTQ
- Secondary: SAM, BAM
- Tertiary: VCF

BIOTEAM
Enabling Science

iRODS In The Domain of Life Sciences

Tiered Storage of Sequencing Data: Optimizing processing & performance

Raw: BCL

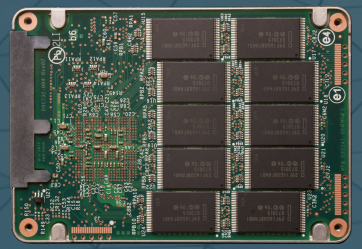
Primary: FASTQ

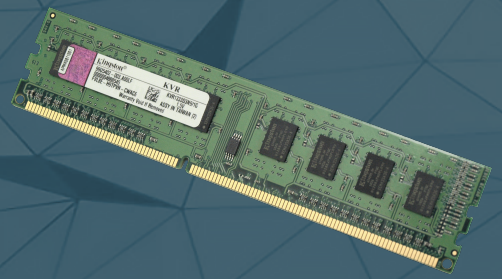
iRODS Resource
(Consumer)

Secondary: SAM, BAM

Tertiary : VCF

iRODS Resource
(Consumer)



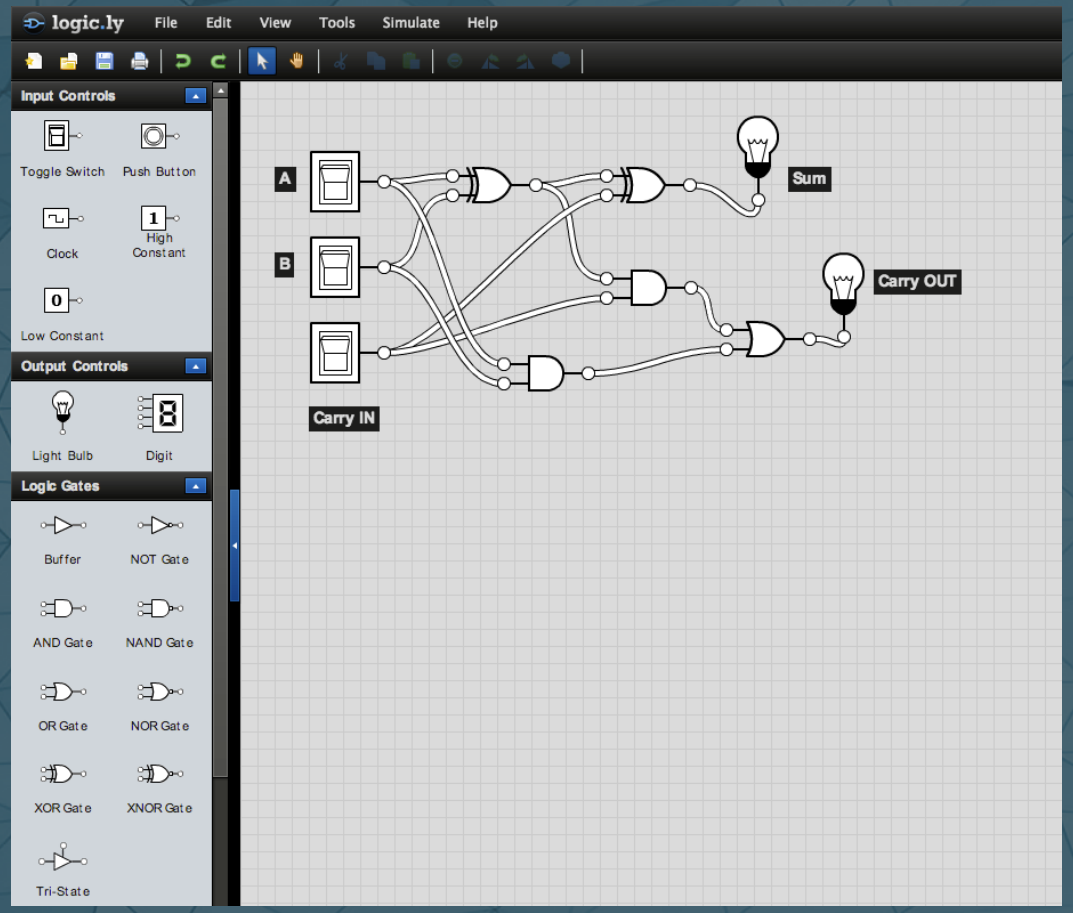


BIOTEAM
Enabling Science

iRODS In The Domain of Life Sciences

The full picture we're working towards

- Web UI: Primary Interface
  - Invoke workflows of rules / microservices / processes
  - View status and output
  - Search and discover in-flight & processed data
- Rule Engine: Data "Brain"
  - Visually compose workflow automation
  - Code for automated workflows
  - Invoke Multi-step Data Processing Pipelines
  - Distributed computation
  - Tiered Storage for data at-rest and processing
  - Geographically distributed Backups



BIOTEAM
Enabling Science

Open-source

- Add to the sum of human knowledge and computational ability
- We can work faster and achieve more together
- No sense in reinventing the wheel, unless you're inventing a better wheel
- If an idea is worth creating, it's worth sharing

BIOTEAM
Enabling Science

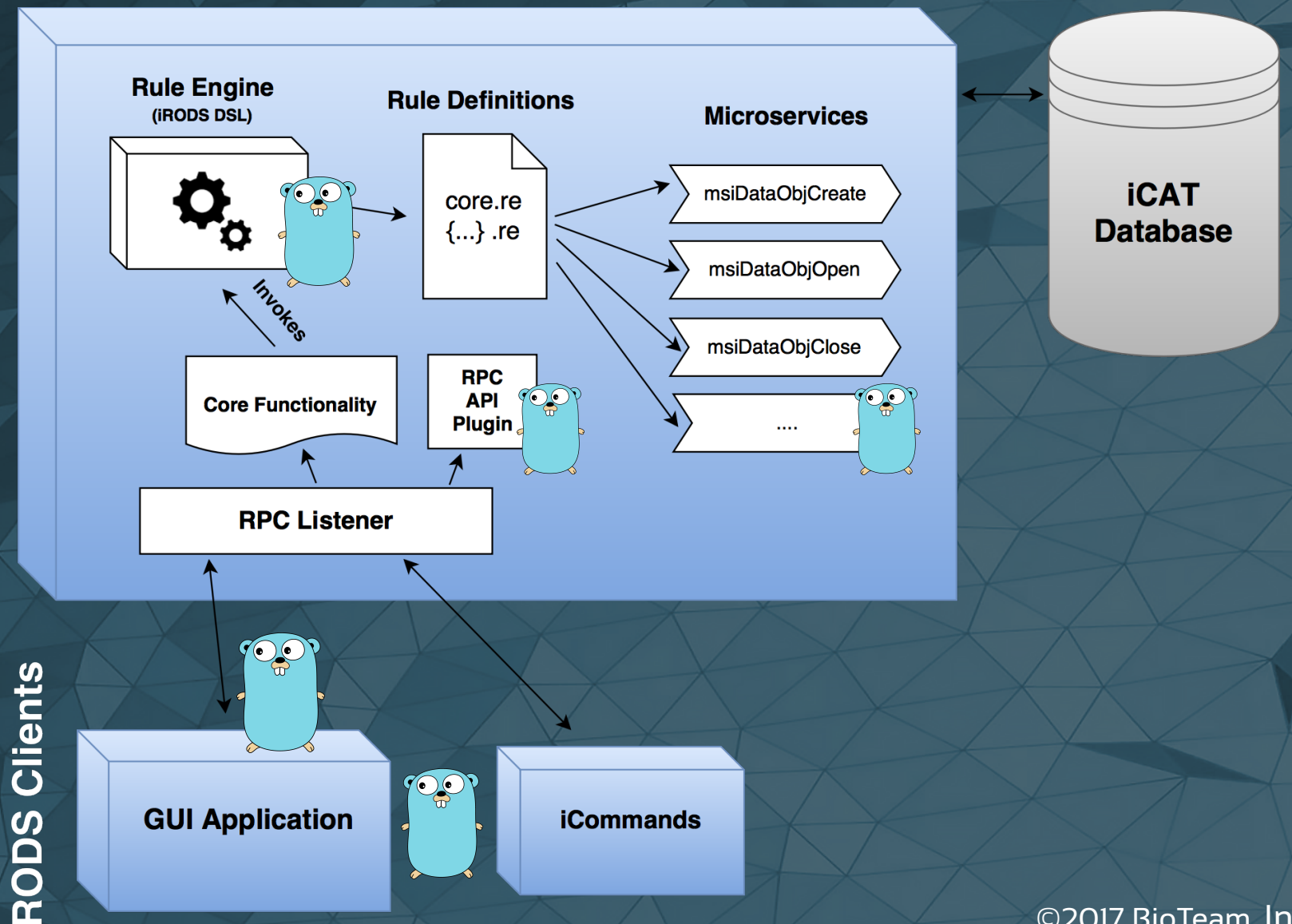
But Why Golang?

A biased explanation of the Golang programming language

- Golang is John's favorite programming language
- It plays nicely with C and C++ (via cgo!)
- It's fast
- Simple design
- Encourages consistency (idiomatic go code), modularity (via packages), sturdy code (forced error handling)
- Standard library
- Standard toolchain – profiling, testing, etc…
- Community
- Concurrency model for dummies
- Deployment
- End goal: Commoditize iRODS development for the Golang community

BIOTEAM
Enabling Science

iRODS Integration Points: Visual



BIOTEAM
Enabling Science

iRODS Integration Points

Common use cases

- iRODS Client
  - RESTful APIs
  - WebDAV
  - Web applications
  - Remote integration (decoupled distributed compute)

- Microservices & iRODS Rule Language DSL
  - Automated data processing
  - Automated business logic and data management
  - Replica based distributed computation

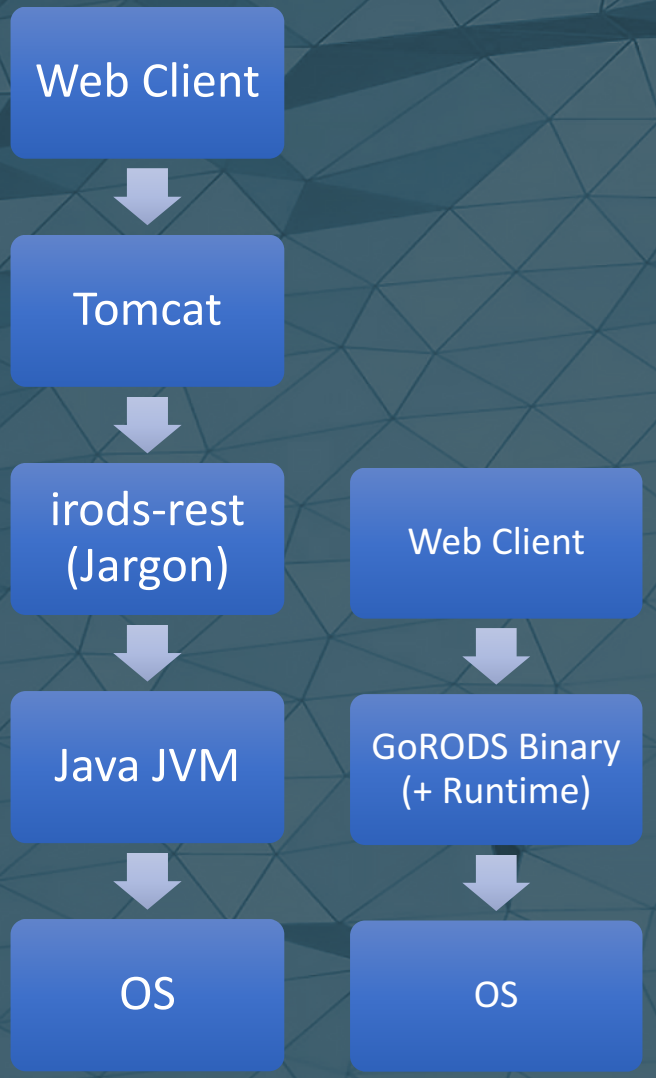
- RPC API
  - Deep integration
  - Side-loaded business logic
  - Full DB access
    (good for additional features for stateless clients)

- Rule Engine
  - Full integration into iRODS rule engine component, a step above microservices
  - DRY event notifications -> WebSocket server -> Web UI

Integration Point: iRODS Client

GoRODS: https://github.com/jjacquay712/GoRODS

- iRODS C API Binding via cgo
- Supports latest 4.2.1
- Efficient options for memory management
- Working example for HTTP User Interface
  - Byte range support (concurrency for large downloads, "chunked downloads")
- Fully documented at godoc.org



BIOTEAM
Enabling Science

Integration Point: iRODS Client

GoRODS: https://github.com/jjacquay712/GoRODS

```go
package main

import (
    "fmt"
    "github.com/jjacquay712/GoRODS"
    "log"
)

func main() {

    client, conErr := gorods.New(gorods.ConnectionOptions{
        Type: gorods.UserDefined,

        Host: "localhost",
        Port: 1247,
        Zone: "tempZone",

        Username: "rods",
        Password: "password",
    })

    // Ensure the client initialized successfully and connected to the iCAT server
    if conErr != nil {
        log.Fatal(conErr)
    }
}
```

Creating a Connection to iRODS

Integration Point: iRODS Client

GoRODS: https://github.com/jjacquay712/GoRODS

```go
// Open a collection reference for /tempZone/home/rods
if openErr := client.OpenCollection(gorods.CollectionOptions{
    Path: "/tempZone/home/rods",
}, func(col *gorods.Collection, con *gorods.Connection) {

    // Output collection's string representation
    fmt.Printf("String(): %v \n", col)

    // Loop over the data objects in the collection, print the file name
    col.EachDataObj(func(obj *gorods.DataObj) {
        fmt.Printf("%v \n", obj.Name())
    })

    // Loop over the subcollections in the collection, print the name
    col.EachCollection(func(subcol *gorods.Collection) {
        fmt.Printf("%v \n", subcol.Name())
    })

}); openErr != nil {
    log.Fatal(openErr)
}
```

Looping over collection objects

Output:

Collection: /tempZone/home/rods
- C: pemtest
- C: source-code
- C: test
- d: hello.txt
- d: mydir1.tar

Integration Point: iRODS Client

GoRODS: https://github.com/jjacquay712/GoRODS

```go
mountPath := "/irods/"

// Setup the GoRODS FileServer
fs := gorods.FileServer(gorods.FSOptions{
    Path:   "/tempZone/home/rods",
    Client: client,
    Download: true,
    StripPrefix: mountPath,
})

// Create the URL router
mux := http.NewServeMux()

// Serve the iRODS collection at /irods/
mux.Handle(mountPath, http.StripPrefix(mountPath, fs))

// Start HTTP server on port 8080
log.Fatal(http.ListenAndServe(":8080", mux))
```

Serving over HTTP

BIOTEAM
Enabling Science

Integration Point: iRODS Client

GoRODS: https://github.com/jjacquay712/GoRODS

Collection: /tempZone/home/r

John (BioTeam)

localhost:6060/irods/src/github.com/jjacquay712/GoRODS/

GoRODS HTTP File Server

src > github.com > jjacquay712 > GoRODS

/tempZone/home/rods/src/github.com/jjacquay712/GoRODS

Create Collection | Upload Data Object

Data Object "build.sh"

Metadata | ACL

Attribute	Value	Units
language	golang	lang

A: Attribute V: Value U: Units Add AVU

Close

Name		
..		
.git		
aaron		
lib		
.gitattributes		
.gitignore		
acl.go		
build.sh		
client.go	4.3 KiB	Data Object
client_test.go	625 bytes	Data Object
collection.go	28.7 KiB	Data Object
collection_test.go	1.0 KiB	Data Object
connection.go	29.8 KiB	Data Object
connection_test.go	882 bytes	Data Object
dataobj.go	31.7 KiB	Data Object
dataobj_test.go	3.3 KiB	Data Object
errors.go	1.2 KiB	Data Object
group.go	11.4 KiB	Data Object
helpers.go	3.3 KiB	Data Object
HOWTO.md	23.5 KiB	Data Object
httphandler.go	5.1 KiB	Data Object

Serving over HTTP: Output

BIOTEAM
Enabling Science

Integration Point: iRODS Client

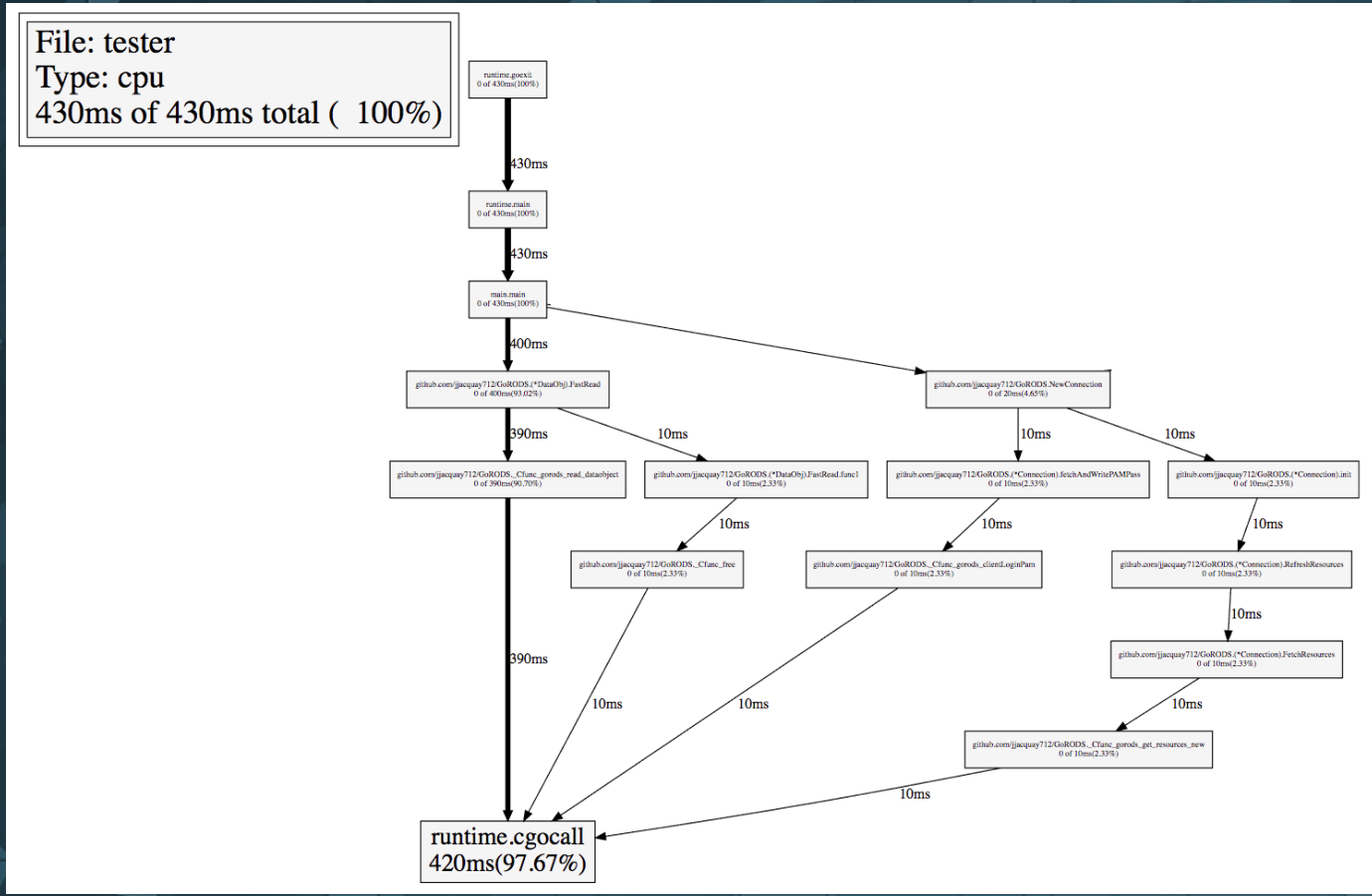
GoRODS: https://github.com/jjacquay712/GoRODS

```
1117        outBuff := make(chan *ByteArr, 100)
1118
1119        go func() {
1120            if readEr := obj.ReadChunkFree(10240000, func(chunk *ByteArr) {
1121                outBuff <- chunk
1122            }); readEr != nil {
1123                log.Print(readEr)
1124
1125                handler.response.WriteHeader(http.StatusInternalServerError)
1126                handler.response.Write([]byte("Error: " + readEr.Error()))
1127
1128            }
1129
1130            close(outBuff)
1131        }()
1132
1133        for b := range outBuff {
1134            handler.response.Write(b.Contents)
1135            b.Free()
1136        }
1137    }
1138
```

Reading Data / Concurrency

Integration Point: iRODS Client

GoRODS: https://github.com/jjacquay712/GoRODS

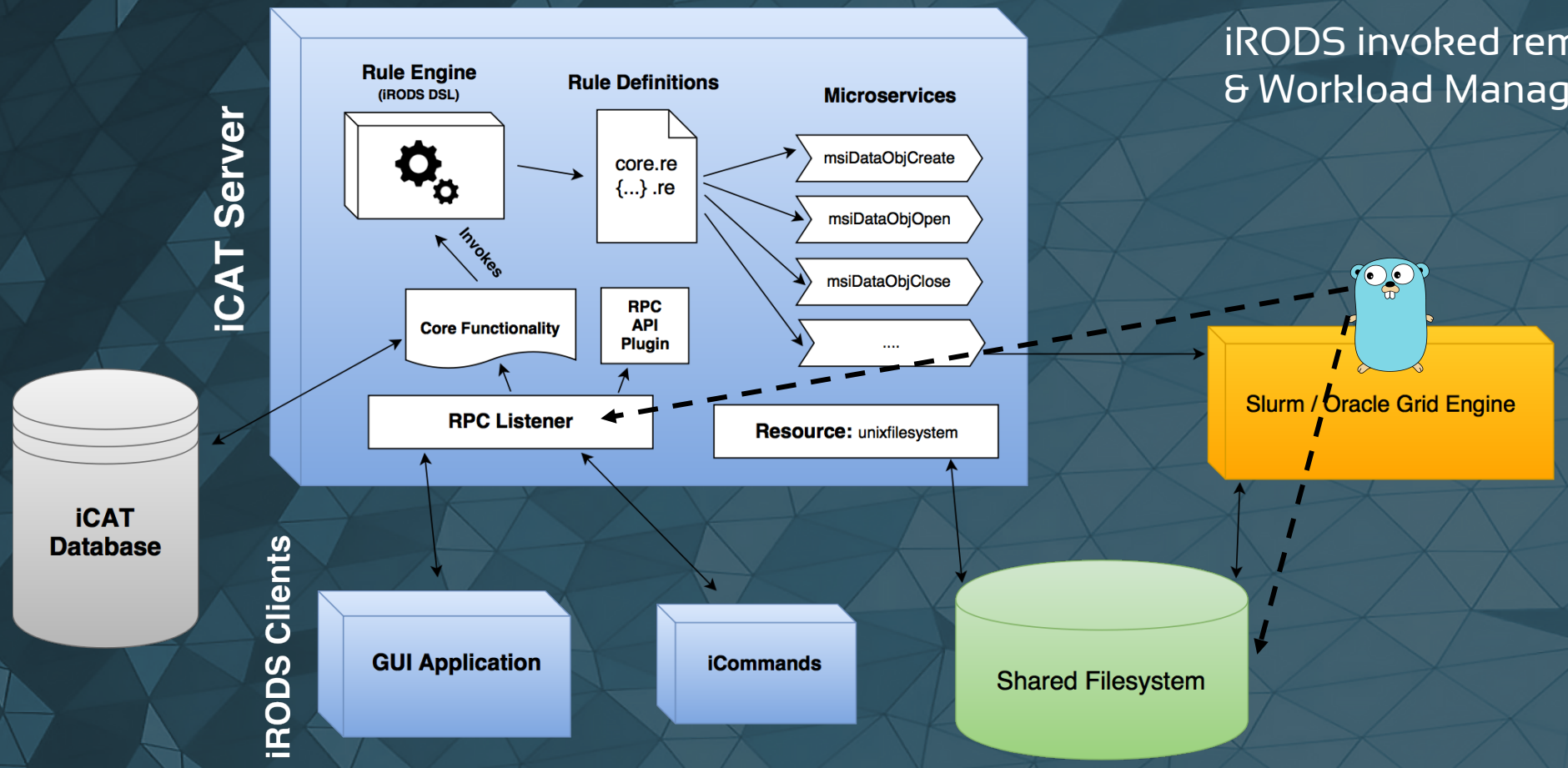


Golang CPU Profiling

Integration Point: iRODS Client

GoRODS: https://github.com/jjacquay712/GoRODS

iRODS invoked remote compute
& Workload Manager integration



BIOTEAM
Enabling Science

iRODS Integration Points

& Common use cases

- iRODS Client
  - RESTful APIs
  - WebDAV
  - Web applications
  - Remote integration (decoupled distributed compute)

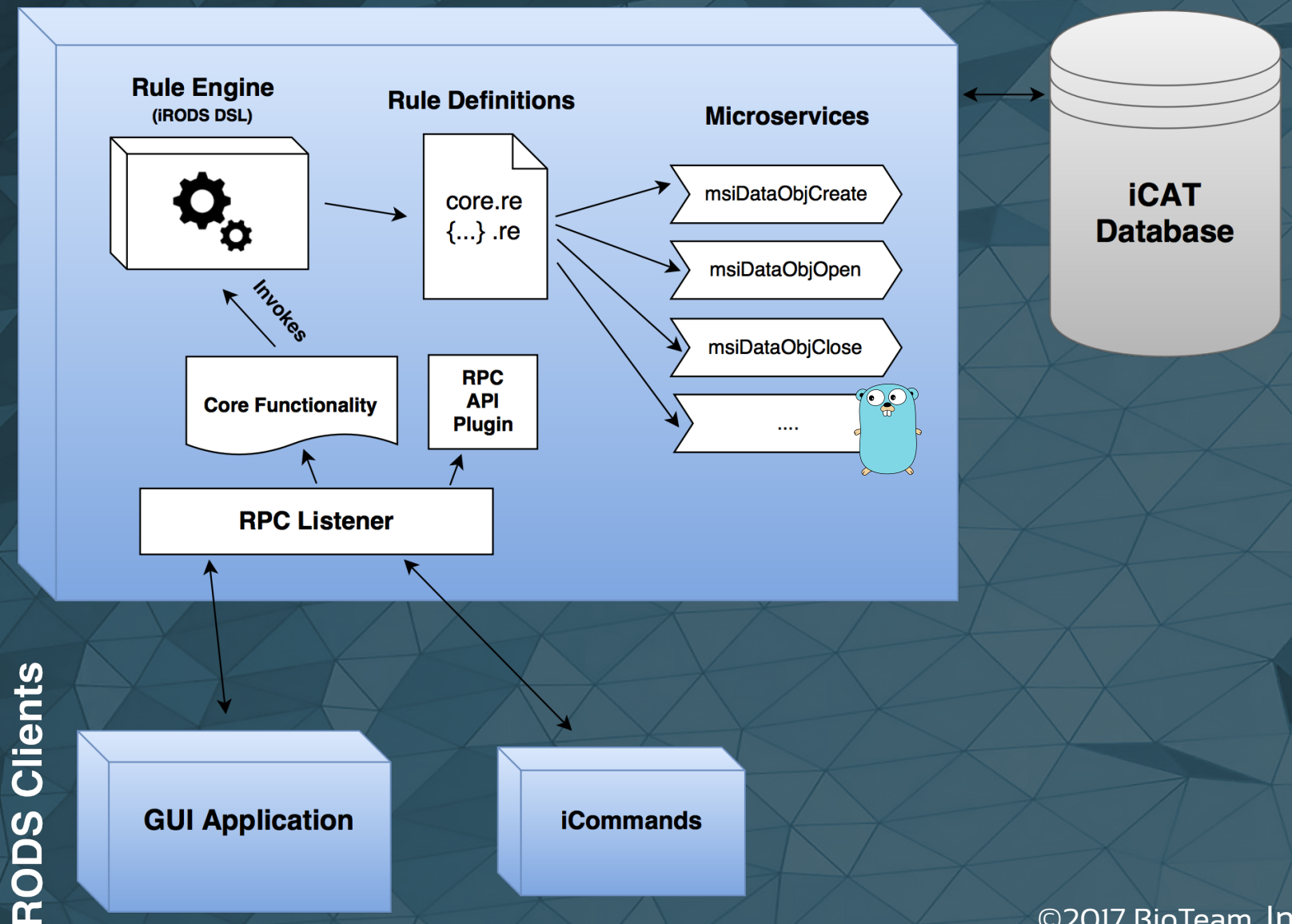
- RPC API
  - Deep integration
  - Side-loaded business logic
  - Full DB access
    (good for additional features for stateless clients)

- Microservices & iRODS Rule Language DSL
  - Automated data processing
  - Automated business logic and data management
  - Replica based distributed computation

- Rule Engine
  - Full integration into iRODS rule engine component, a step above microservices
  - DRY event notifications -> WebSocket server -> Web UI

BIOTEAM
Enabling Science

iRODS Integration Points: Visual



BIOTEAM
Enabling Science

Integration Point: Microservices & iRODS Rule Language DSL

https://github.com/jjacquay712/irods-ugm-2017

Introducing: GoRODS/msi

- Everything is on GitHub, ready to go. Go get it!
- Just spin up the Vagrant machine (2.5GB box file)
- Uses new GoRODS "msi" subpackage, written just for you!

```
    // Associate metadata to data object
    if err := msi.Call("msiAssociateKeyValuePairsToObj", labelsKVP, imageFilePath, "-d"); err != nil {
        log.Print(err)
        return msi.SYS_INTERNAL_ERR
    }

```

- Simple example "msibasic_example" returns CSV metadata as KVP struct
  - Testing example
- Advanced example implements Google Cloud Vision and Translation APIs to extract meaningful metadata, optionally compresses data objects

BIOTEAM
Enabling Science

Integration Point: Microservices & iRODS Rule Language DSL

https://github.com/jjacquay712/irods-ugm-2017

```go
package main

import (
    "encoding/csv"
    "github.com/jjacquay712/GoRODS/msi"
    "io"
    "log"
    "strings"
    "unsafe"
)

// #cgo CFLAGS: -I/usr/include/irods
// #cgo LDFLAGS: -lirods_server -lirods_common -lpthread
/*
#include "msParam.h"
#include "re_structs.h"
*/
import "C"

//export BasicExample
func BasicExample(inputParam *C.msParam_t, outputParam *C.msParam_t, rei *C.ruleExecInfo_t) int {

    // Setup GoRODS/msi
    msi.Configure(unsafe.Pointer(rei))

    // Convert *C.msParam_t to golang types
    inputCSV := msi.ToParam(unsafe.Pointer(inputParam)).String()
    outputKVP := msi.ToParam(unsafe.Pointer(outputParam)).ConvertTo(msi.KeyValPair_MS_T)

    // Set output KVP
    outputKVP.SetKVP(GetKVPMap(inputCSV))

    return msi.SUCCESS
}
```

Basic Example

Input:
2 Column CSV

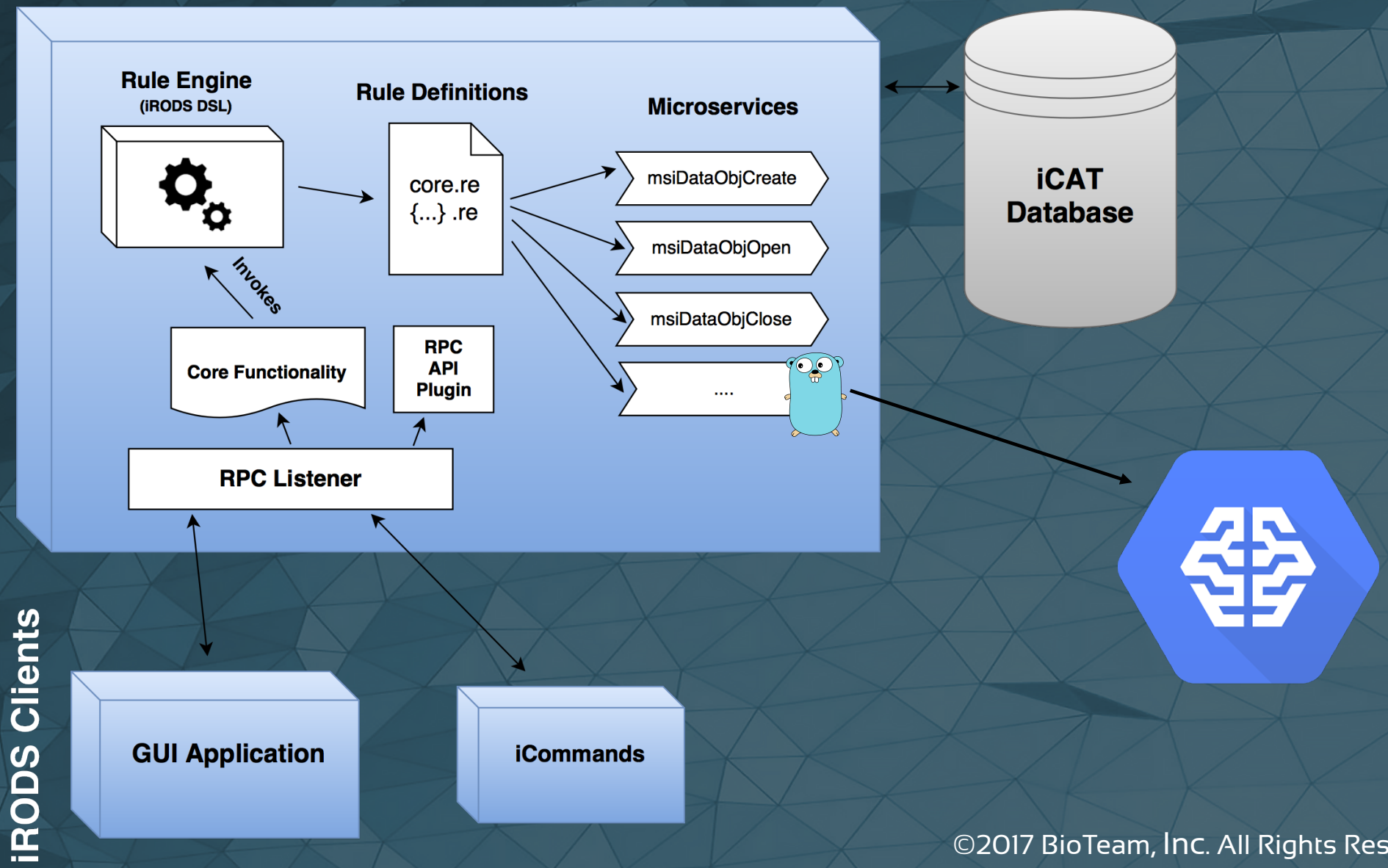
Output:
Key value pair structure (for metadata assignment)

```
TestBasicExample {
    msibasic_example("keytest,valuetest", *outKVP);
    msiPrintKeyValPair("stderr", *outKVP)
}

INPUT null
OUTPUT ruleExecOut
```

BIOTEAM
Enabling Science

iRODS Integration Points: Visual



BIOTEAM
Enabling Science

Integration Point: Microservices & iRODS Rule Language DSL

https://github.com/jjacquay712/irods-ugm-2017

GoRODS/msi

Go's Magical Interfaces

```
56      if enableGzip {
57          gzipDataObjPath := imageFilePath + ".gz"
58          gzipDesc := msi.NewParam(msi.INT_MS_T)
59
60          if err := msi.Call("msiDataObjCreate", gzipDataObjPath, "", gzipDesc); err != nil {
61              log.Print(err)
62              return msi.SYS_INTERNAL_ERR
63          }
64
65          gzipDataObj := msi.NewObjReaderFromDesc(gzipDesc)
66          defer gzipDataObj.Close()
67
68          gzWriter := gzip.NewWriter(gzipDataObj)
69          defer gzWriter.Close()
70
71          origDataObj, err := msi.NewObjReader(imageFilePath)
72          if err != nil {
73              log.Fatalf("Failed to read file: %v", err)
74          }
75          defer origDataObj.Close()
76
77          io.Copy(gzWriter, origDataObj)
78
79          imageFilePath = gzipDataObjPath
80      }
```

Advanced Example

Input:
iRODS Data Object Path

Output:
Assigns metadata from APIs to data object, and optionally compresses the object.

BIOTEAM
Enabling Science

Integration Point: Microservices & iRODS Rule Language DSL

https://github.com/jjacquay712/irods-ugm-2017

GoRODS/msi

Go's Magical Interfaces

```
196     file, err := msi.NewObjReader(filepath)
197     if err != nil {
198         log.Fatalf("Failed to read file: %v", err)
199     }
200     defer file.Close()
201
202     image, err := vision.NewImageFromReader(file)
203     if err != nil {
204         log.Fatalf("Failed to create image: %v", err)
205     }
206
207     labels, err := visionClient.DetectLabels(ctx, image, nil, 10)
208     if err != nil {
209         log.Fatalf("Failed to detect labels: %v", err)
210     }
```

Advanced Example

Input:
iRODS Data Object Path

Output:
Assigns metadata from APIs to data object, and optionally compresses the object.

GoRODS/msi

Memory Management, take #3. This time, it's for real

```go
// NewParam creates a new *Param, with the provided type string
func NewParam(paramType ParamType) *Param {
    p := new(Param)

    p.rodsType = paramType

    cTypeStr := C.CString(string(paramType))
    defer C.free(unsafe.Pointer(cTypeStr))

    p.ptr = C.NewParam(cTypeStr)

    runtime.SetFinalizer(p, func(param *Param) {
        C.FreeMsParam(param.ptr)
    })

    return p
}
```

Integration Point: Microservices & iRODS Rule Language DSL

https://github.com/jjacquay712/irods-ugm-2017

msiextract_image_metadata: Let's run some data through!

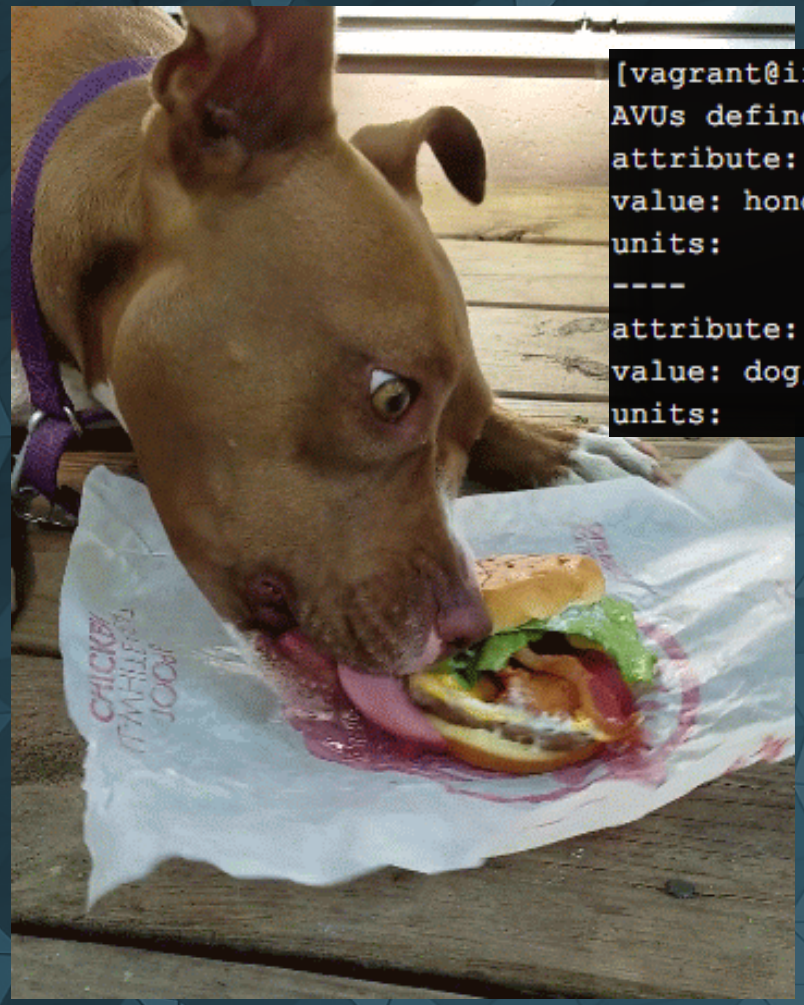
```
attribute: exif_PhotometricInterpretation
value: 2
units:
----
attribute: exif_XResolution
value: "720000/10000"
units:
----
attribute: exif_YResolution
value: "720000/10000"
units:
----
attribute: exif_PixelYDimension
value: 630
units:
----
attribute: exif_ThumbJPEGInterchangeFormat
value: 450
units:
----
attribute: exif_ImageWidth
value: 840
units:
```

```
[vagrant@irods-icat irods-ugm-2017]$ imeta ls -d gopher.jpg.gz
AVUs defined for dataObj gopher.jpg.gz:
attribute: tags_english
value: mammal,vertebrate,wildlife,squirrel,fauna,whiskers,prairie dog,marmot,rodent,prairie
units:
----
attribute: tags_dutch
value: zoogdier,gewerveld,dieren in het wild,eekhoorn,fauna,bakkebaarden,prairiehond,marmot,knaagdier,prairie
units:
```

BIOTEAM
Enabling Science

Integration Point: Microservices & iRODS Rule Language DSL

https://github.com/jjacquay712/irods-ugm-2017



```
[vagrant@irods-icat irods-ugm-2017]$ imeta ls -d pretzel.gif.gz
AVUs defined for dataObj pretzel.gif.gz:
attribute: tags_dutch
value: hond,zoogdier,gewerveld,Hond zoals zoogdier,Pit bull,Amerikaanse pit bull terrier
units:
----
attribute: tags_english
value: dog,mammal,vertebrate,dog like mammal,pit bull,american pit bull terrier
units:
```

BIOTEAM
Enabling Science

Integration Point: Microservices & iRODS Rule Language DSL

https://github.com/jjacquay712/irods-ugm-2017

```
[vagrant@irods-icat irods-ugm-2017]$ imeta ls -d jason.jpg
AVUs defined for dataObj jason.jpg:
attribute: tags_english
value: long luscious hair,facial hair,face,nose,chin,beard,forehead,eyebrow,hairstyle,head,mouth,hair metal band
units:
----
attribute: tags_dutch
value: lang lekker haar,gezichtshaar,gezicht,neus,kin,baard,voorhoofd,wenkbrauw,kapsel,hoofd,mond,haar metalen band
units:
```

BIOTEAM
Enabling Science

iRODS Integration Points

& Common use cases

- iRODS Client
  - RESTful APIs
  - WebDAV
  - Web applications
  - Remote integration (decoupled distributed compute)

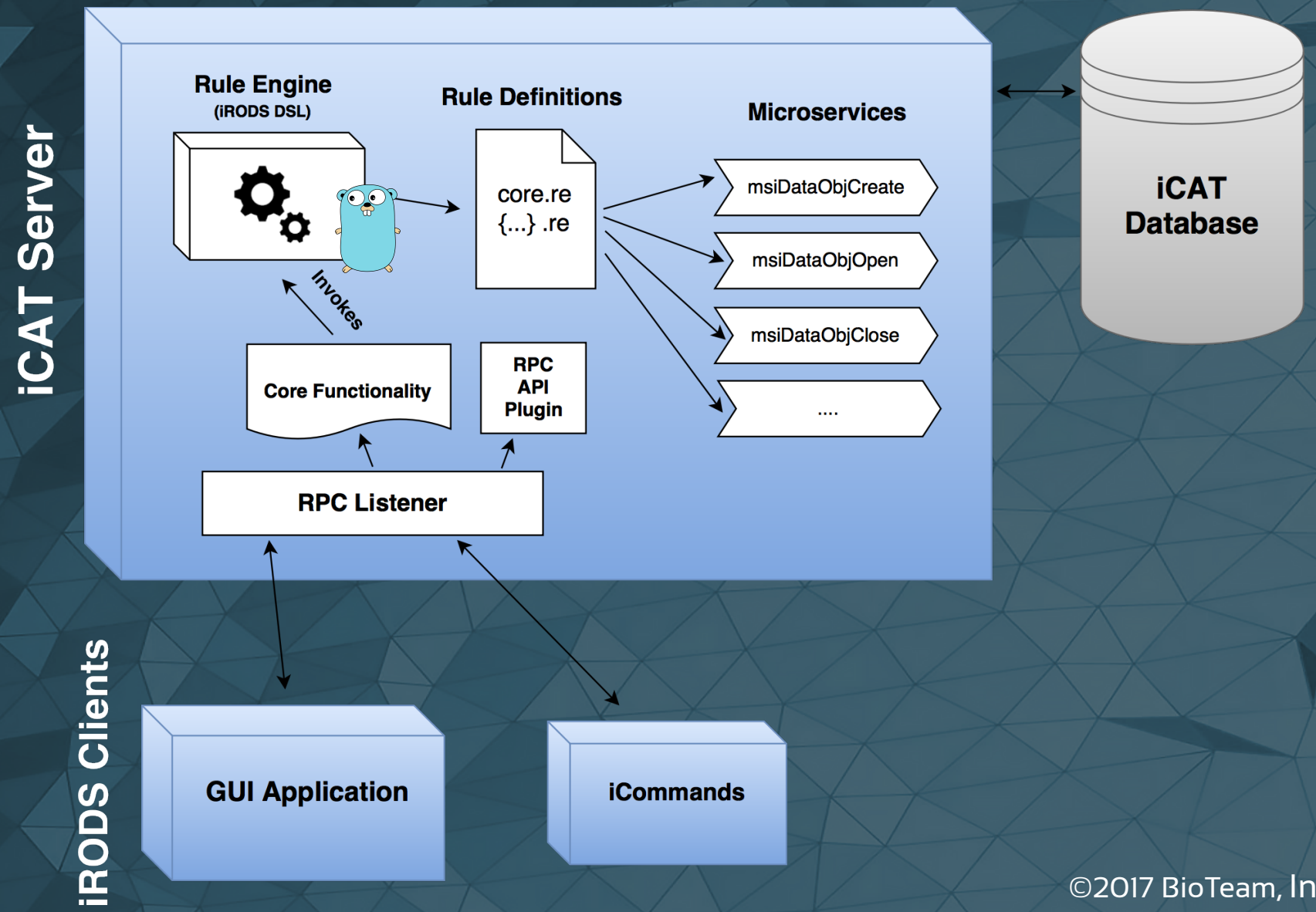
- RPC API
  - Deep integration
  - Side-loaded business logic
  - Full DB access (good for additional features for stateless clients)

- Microservices & iRODS Rule Language DSL
  - Automated data processing
  - Automated business logic and data management
  - Replica based distributed computation

- Rule Engine
  - Full integration into iRODS rule engine component, a step above microservices
  - DRY event notifications -> AMQP -> WebSocket server -> Web UI

BIOTEAM
Enabling Science

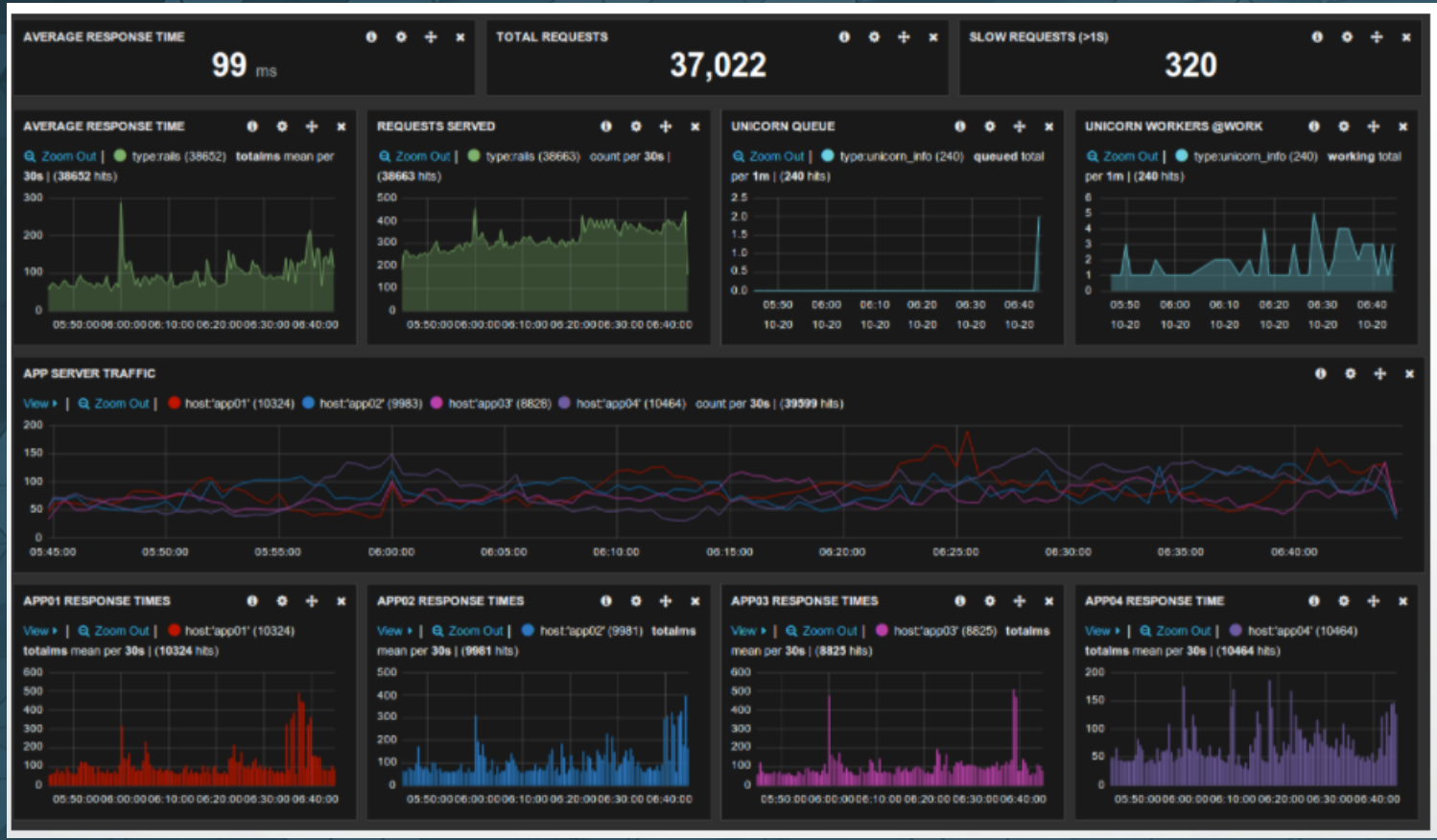
iRODS Integration Points: Visual



BIOTEAM
Enabling Science

Integration Point: Rule Engine

Kibana



BIOTEAM
Enabling Science

```go
package main

import (
    "github.com/xiam/exif"
    re "gore"
    "log"
    "path"
    "strings"
)

func init() {

    re.Configure(re.Opts{
        Verbose:          true,
        LoadDefaultRules: true,
    })

    re.RegisterRules(re.Rules{
        "acPostProcForPut": func(ps re.Params, cb *re.Callback) *re.Error {

            sess := cb.SessionVars()

            objPath := sess.Get("objPath").String()
            physPath := sess.Get("filePath").String()

            ext := path.Ext(objPath)

            if ext == ".jpg" {
                if data, err := exif.Read(physPath); err == nil {

                    metaStr := ExifToKVPStr(&data)

                    meta := re.NewMsParam(re.KeyValPair_MS_T)

                    if er := cb.Call("msiString2KeyValPair", metaStr, meta); !er.Ok() {
                        return er
                    }

                    log.Print(meta.String())

                    if er := cb.Call("msiAssociateKeyValuePairsToObj", meta, objPath, "-d"); !er.Ok() {
                        return er
                    }
                } else {
```

default_rules.go

```go
package gore

import (
    "log"
)

var defaultRules Rules = Rules{
    "printHello": func(ps Params, cb *Callback) *Error {
        return cb.Call("print_hello")
    },
    "acPreConnect": func(ps Params, cb *Callback) *Error {

        ps.Each(func(p RodsObj) {
            if p.Type() == StdStringPtr {
                strPtr := p.(*RodsStdStringPtr)

                strPtr.Set("CS_NEG_DONT_CARE")
            }
        })

        return Success()
    },
    "acCreateUser": func(ps Params, cb *Callback) *Error {

        var err *Error = cb.Call("acPreProcForCreateUser")

        err = cb.Call("msiCreateUser")
        if !err.Ok() {
            cb.Call("msiRollback")
            return err
        }

        info := cb.SessionVars()

        otherUserName, _ := info.Get("userNameProxy")

        if otherUserName.String() != "anonymous" {
            err = cb.Call("acCreateDefaultCollections")

            err = cb.Call("msiAddUserToGroup", "public")
            if !err.Ok() {
                cb.Call("msiRollback")
                return err
            }
        }

        cb.Call("msiCommit")

        return cb.Call("acPostProcForCreateUser")
```

BIOTEAM
Enabling Science

iRODS Integration Points

& Common use cases

- iRODS Client
  - RESTful APIs
  - WebDAV
  - Web applications
  - Remote integration (decoupled distributed compute)

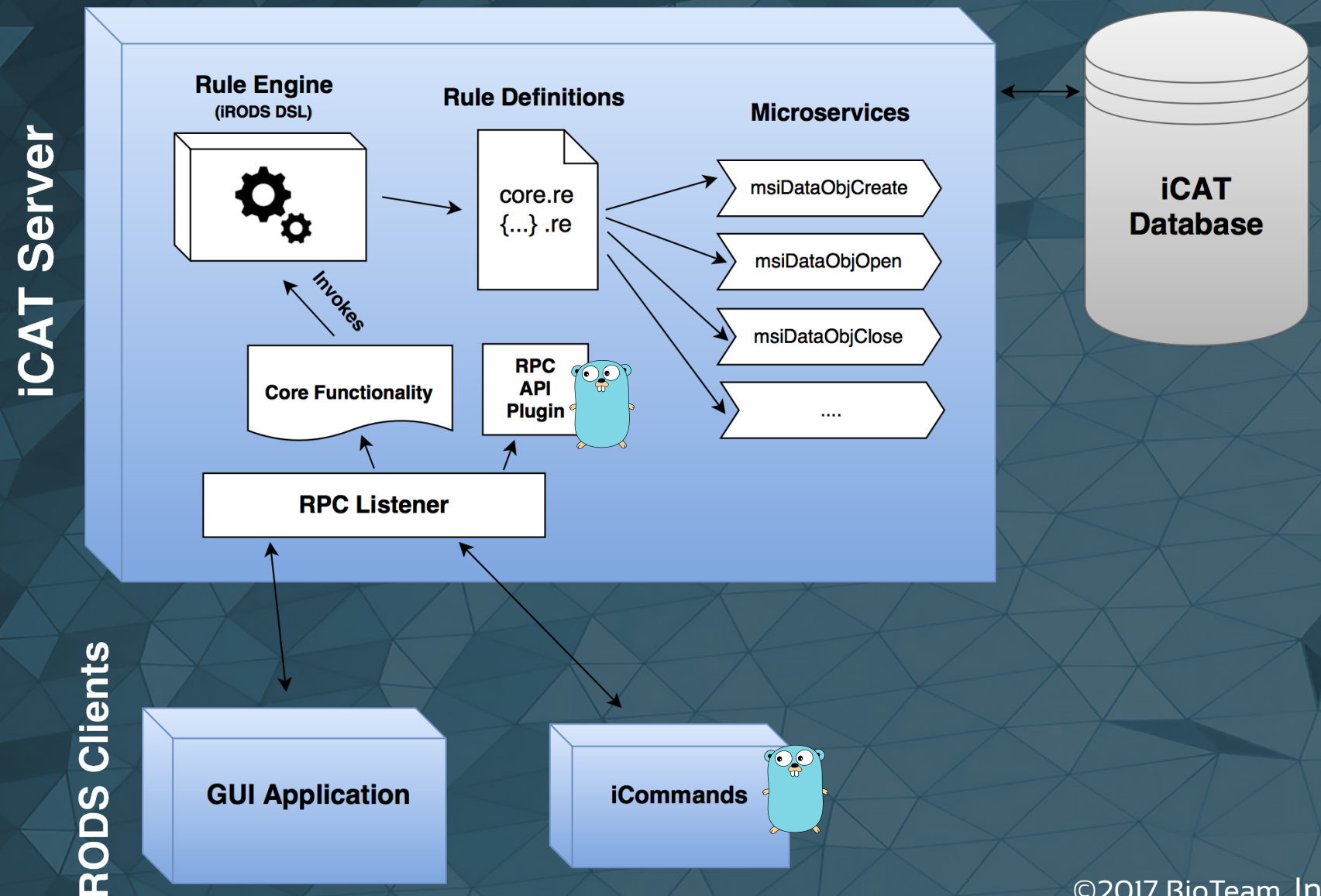
- RPC API
  - Deep integration
  - Side-loaded business logic
  - Full DB access (good for additional features for stateless clients)

- Microservices & iRODS Rule Language DSL
  - Automated data processing
  - Automated business logic and data management
  - Replica based distributed computation

- Rule Engine
  - Full integration into iRODS rule engine component, a step above microservices
  - DRY event notifications -> WebSocket server -> Web UI

BIOTEAM
Enabling Science

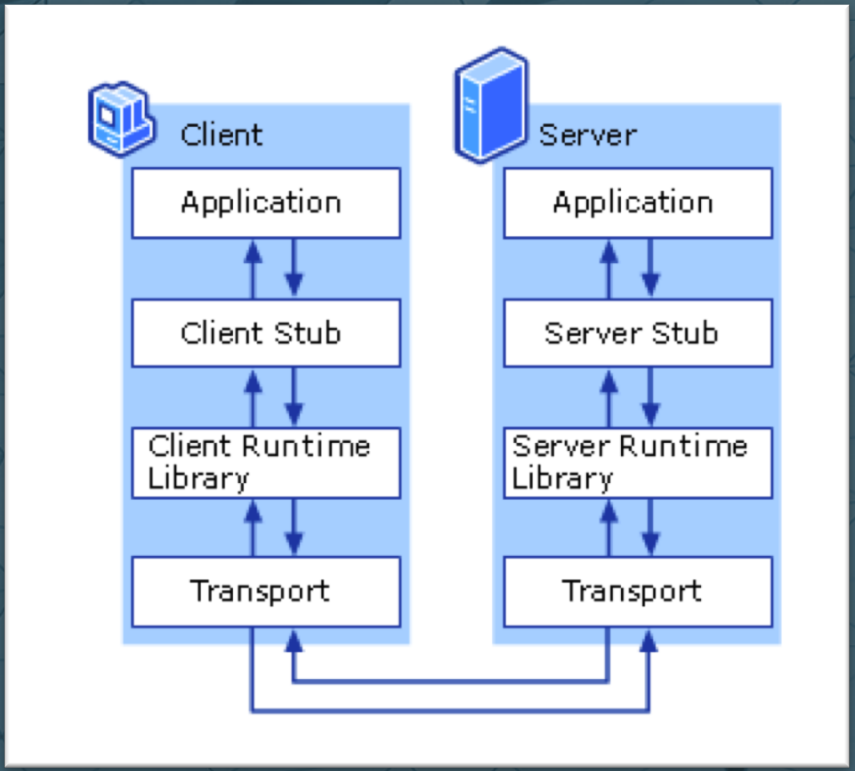
iRODS Integration Points: Visual



Integration Point: RPC API

Still figuring this out myself

- Similar binding as rule engine integration
- Low-level binding to C, C++ data structures
  - Not just iRODS C API functions
- Client & Server component



BIOTEAM
Enabling Science

The End

GoRODS: https://github.com/jjacquay712/GoRODS

GoRODS/msi: https://github.com/jjacquay712/GoRODS/msi

UGM 2017 Microservice Demo

https://github.com/jjacquay712/irods-ugm-2017

Special Thanks To:
BioTeam, University of Florida, Product Development Team, Aaron Gardner, Alex Oumantsev, iRODS Consortium, My Dog Pretzel