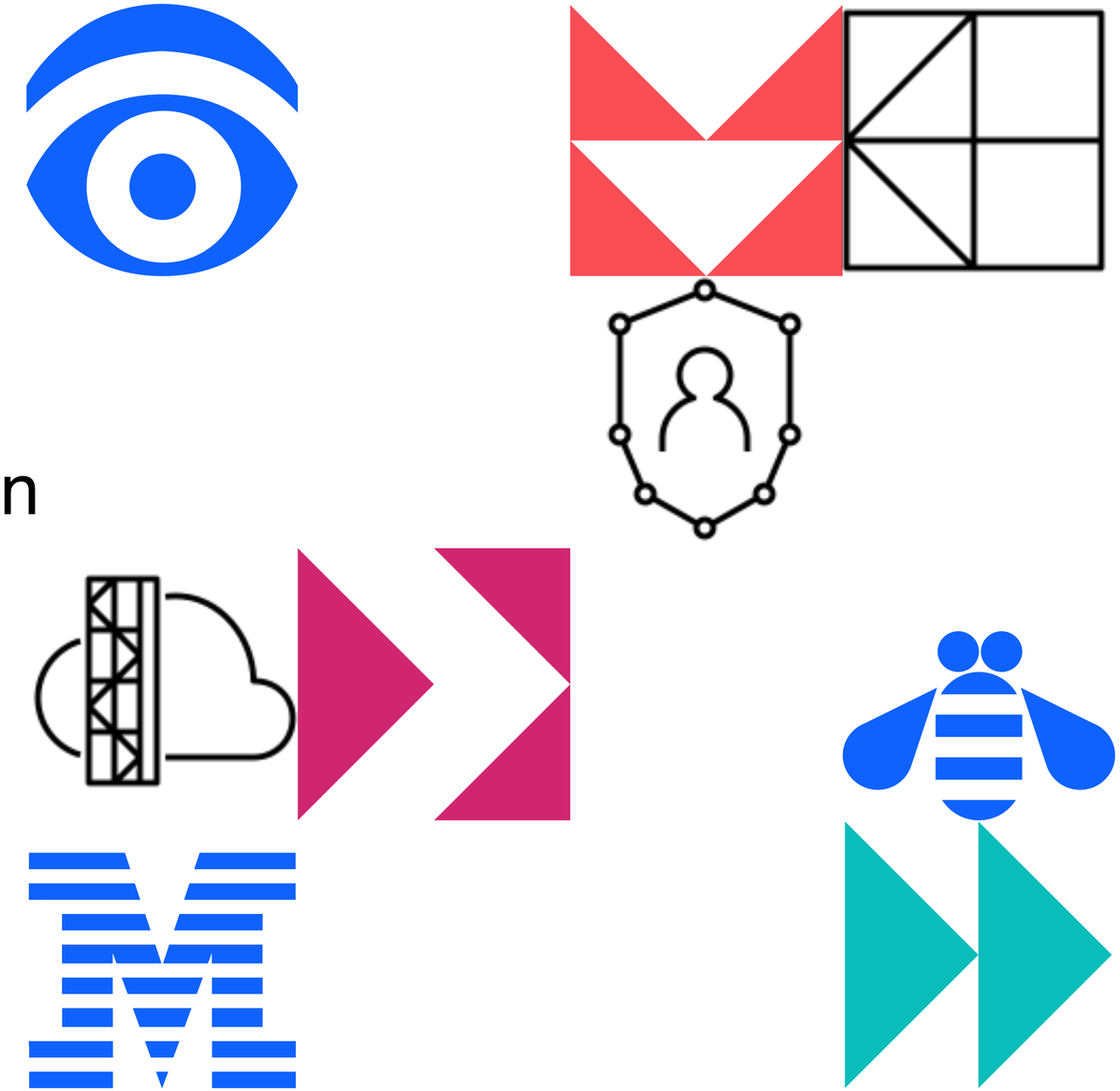
IBM TechXchange

Session **2847**

Two PyRACFs, One PyPi: Mainframe Open Source Collaboration



H.B. Kuiper
Wizard

IBM


TWO PYRACFS
CYOS
Z/OS
PYPI
PY/PI
PYTHON
PY/PI
PYTHON
PY/PI
TWO PYRACFS, ONE PYPI

Notices and disclaimers

This document is distributed “as is,” just like an untested JCL job—proceed at your own risk. We provide zero warranties, implied or express.

If your LPAR crashes, your batch job goes rogue, or your coffee machine stops working, we are not responsible. This includes, but is not limited to, loss of datasets, CICS meltdowns, REXX scripts spiraling out of control, or any profit that evaporates into thin air faster than your CPU credits on a misconfigured z/OS.

Customer examples are like carefully optimized assembler code: they show how some sysadmins have conquered the mainframe, but your mileage may vary.

Your results might differ based on your operating system, skill level, or whether you still run on COBOL from 1974.

Workshops and sessions may be crafted by independent mainframe gurus who dwell in the deepest corners of their data centers. Their ideas may not represent the views of zDevOps, especially if they suggest using punch cards (please, no).

Oh, and not all offerings are available in every galaxy. So if you're running on a legacy server in a parallel universe—sorry, no zDevOps support for you!

Any statements about future directions or plans for world domination with zDevOps are purely speculative, subject to spontaneous quantum shifts, and could be withdrawn at warp speed with no prior notice.

zDevOps, the zDevOps logo, and zdevops.com are proudly trademarked in multiple realities. Other product names might belong to us or some other tech overlord from the 21st century.

If you're really into copyright details, hyperspace jump to: www.zdevops.com/legal/copytrade.shtml.

Live long and compile!

SHARE Kansas City 2024

SHARE ORLANDO 2024 WINNERS

BEST OF THE BEST SESSION WINNERS

- **User Session:** "From Accidental Hacker to Ethical Wizard: Unmasking Anonymous Downloads and Responsible Disclosure" Henri Kuiper
- **Vendor Session:** "Turn "Oh No!" Into "No Problem" with Safeguarded Copy an IBM Z Batch Resiliency" Chris Taylor and Butch Rambish

BEST SESSION

Application Development

- User Session: Using Git with ISPF under z/OS - Yes you really can
  Speakers: Lionel Dyck and Henri Kuiper

Core Platform

- Vendor Session: Pragmatic Python for Pioneering Programmers
  Speakers: Steven Perva and Michael Fontanetta

New and Innovative Technologies

- Vendor Session: Using AI for understanding COBOL based applications
  Speaker: Gary Brach

Service Delivery

- User Session: No Longer a Myth: A Guide to Mainframe Buffer Overflows
  Speaker: Philip Young
- Vendor Session: How to Talk to Your Executives About Resiliency and Security
  Speaker: Rebecca Levesque

Strategic Partner

- WatchTower: A Whole New World of Mainframe Inclusive Observability
  Speakers: Angelika Heinric and Kai Kirsch
- Getting the Most Out of OSA and HiperSockets with z/OS Communications Server
  Speaker: Randall Kunkel

For those of you that don't know me

Henri Kuiper

wizard@zdevops.com / henri@mainframesociety.com
https://www.linkedin.com/in/wizardofzos/
@henrikuiper / @wizardofzos

IBM CHAMPION 2020 IBM

Belastingdienst

GSE GUIDE SHARE EUROPE

This is me


ALMELO
NETHERLANDS
mult


CRAYONS
BUT BEFORE WE BEGIN,

RACF

IRRSMO00

Jupyter

Pandas

PYTHON

PyPi


AND AS YOU MAY HAVE GUESSED,
THAT'S WHERE OUR STORY
BEGINS.

This story started a long time ago…

GSE GUIDE SHARE EUROPE

GSE UK Virtual Conference 2021

Virtually the best way to learn about Z

WE CREATE THE FUTURE OF IT

11 RACF Things worth addresing

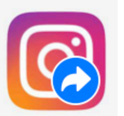
Henri Kuiper

zDevOps

November 2021

Session 1AA

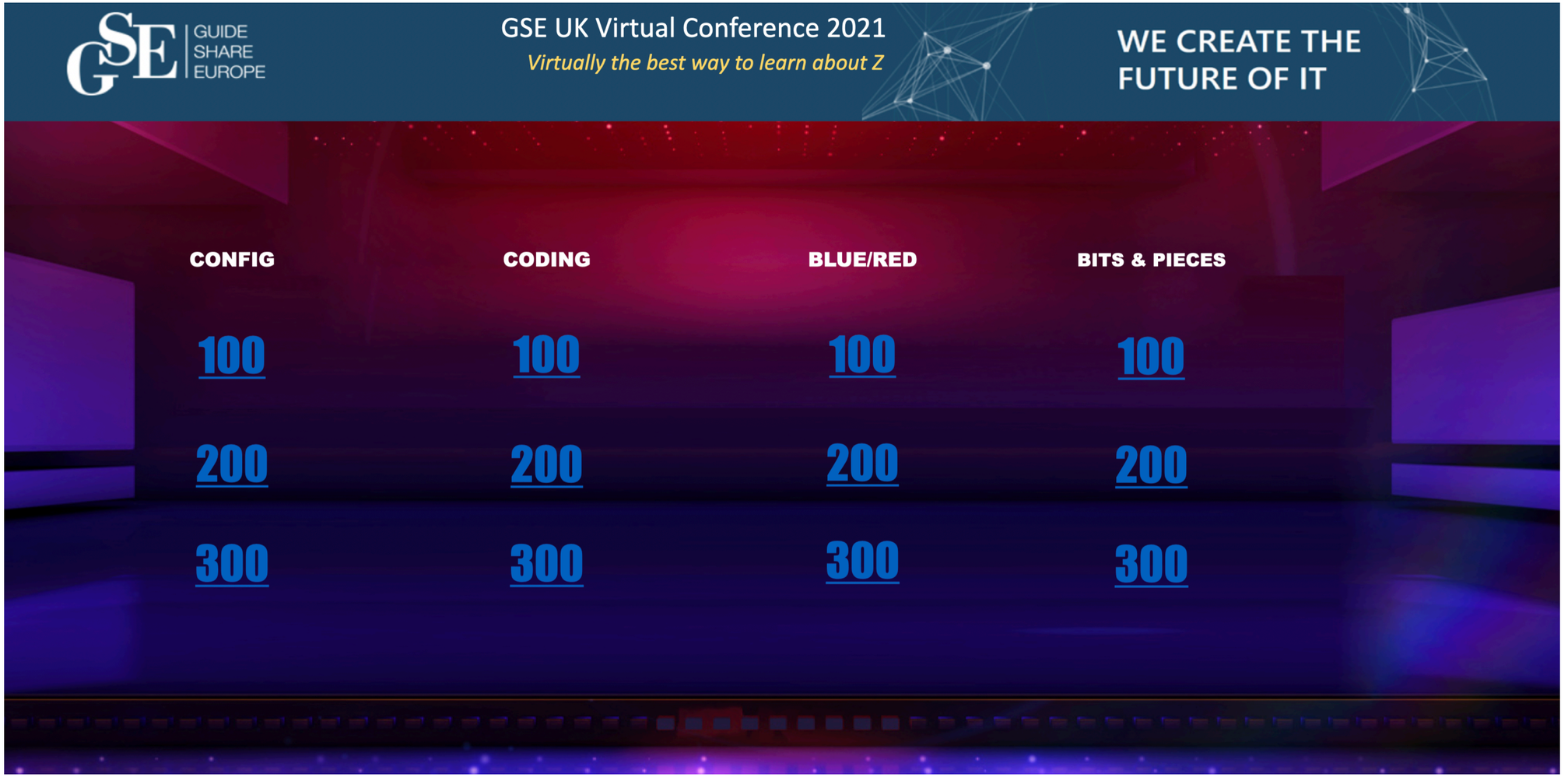
Not a QR code for proof of vaccination



GSE GUIDE SHARE EUROPE UK REGION

IBM

This story started a long time ago…



Parsing with Python???

```
# _*_ coding:utf-8 _*_
import json

def findOffsets(recordType, offsets):
    for offset in offsets:
        if offsets[offset]['record-type'] == recordType:
            return json.loads(json.dumps(offsets[offset]))
    return False

def recordType(unloadline):
    return

with open("offsets.json") as offsets_file:
        offsets = json.load(offsets_file)

i = 0
with open('IRRDBU00.txt', 'r') as infile:
            for line in infile:
                r = line[:4]
                if r == '0400':
                    # list datasets :)
                    dainfo = {}
                    model = findOffsets(r, offsets)
                    for field in model['offsets']:
                        start = int(field['start'])
                        end   = int(field['end'])
                        name  = field['field-name']
                        value = line[start-1:end]
                        dainfo[name] = value
                    print "%s, UACC=%s, OWNER=%s" % (dainfo['DSBD_NAME'],
                     dainfo['DSBD_UACC'],
                     dainfo['DSBD_OWNER_ID'])
```

DCD.S0W1.HZSPDATA, UACC=NONE, OWNER=IBMUSER
ADCDA.*.**, UACC=NONE, OWNER=ADCDA
ADCDB.*.**, UACC=NONE, OWNER=ADCDB
ADCDC.*.**, UACC=NONE, OWNER=ADCDC
..
DSN710.ARCHLOG2.D01351.T1945222.A0000008, UACC=READ, OWNER=SYS1
DSN710.ARCHLOG2.D01351.T1945222.B0000008, UACC=UPDATE, OWNER=DSN1MSTR
DSN710.ARCHLOG2.D01352.T1413499.A0000009, UACC=UPDATE, OWNER=DSN1MSTR
DSN710.ARCHLOG2.D01352.T1413499.B0000009, UACC=UPDATE, OWNER=DSN1MSTR
DSN710.ARCHLOG2.D01352.T1426322.A0000010, UACC=UPDATE, OWNER=DSN1MSTR
DSN710.ARCHLOG2.D01352.T1426322.B0000010, UACC=UPDATE, OWNER=DSN1MSTR
..
DSN710.ARCHLOG2.D02018.T2043554.A0000014, UACC=UPDATE, OWNER=DSN1MSTR
DSN710.ARCHLOG2.D02018.T2043554.B0000014, UACC=UPDATE, OWNER=DSN1MSTR
IBMUSER.IBMUSER.*.**, UACC=READ, OWNER=IBMUSER
UPD.*.**, UACC=NONE, OWNER=IBMUSER
USER1.*.**, UACC=NONE, OWNER=USER1
USER2.*.**, UACC=NONE, OWNER=USER2
USER3.*.**, UACC=NONE, OWNER=USER3
WORKIT.*.**, UACC=NONE, OWNER=IBMUSER
ZDO.*.**, UACC=NONE, OWNER=IBMUSER
ZDOHENRI.**.ZFS, UACC=READ, OWNER=ZDOHENRI
ZDOHENRI.*.**, UACC=NONE, OWNER=ZDOHENRI
ZDOHK.**.ZFS, UACC=READ, OWNER=ZDOHK
ZDOHK.*.**, UACC=NONE, OWNER=ZDOHK

9th of April 2022 (+5 months)

pyracf 0.0.1

Newer version available (0.8.8)

```
pip install pyracf==0.0.1
```

Released: Apr 9, 2022

Parsing IRRDBU00 unloads in panda dataframes.

Navigation

- Project description
- Release history
- Download files

Verified details

These details have been verified by PyPI

Project description

pyracf

parsing IRRDBU00 unloads like a boss

```
from pyracf import RACF
RACF.parse('path/to/irrdbu00')
RACF.users.info()
```

IBM

Do you do RACF reports?


```
newlist type=racf title="Profiles where IBMUSER is on access list"
 define acl subselect acl(id=ibmuser)
 s acl(id=ibmuser)
 sortlist class profile acl(aclaccess,7,"Access")


P R O F I L E   L I S T I N G   12 Aug 2008 00:20
Profiles where IBMUSER is on access list

Class    Profile key                                    Access
ACCTNUM  **                                             ALTER
DATASET  CBC.**                                         NONE
DATASET  CRMQARUN.ACCESS.B.**                           READ
DATASET  CRMQARUN.NOACCESS.B.**                         NONE
FACILITY $CNF.RACF                                      ALTER
FACILITY CKF.RACF                                       ALTER
STARTED  BLSJPRMI.*                                     ALTER
STARTED  CATALOG.*                                      ALTER
STARTED  CIC410A.*                                      ALTER
```

IBM

Didn’t Really Work For Me

- Sometimes I just take too long ‘crafting the perfect CARLa’
- Once the results are there, I’m spending too much time creating an XLS or other ‘sharable’ display of my findings.
- When I needed an update to the report, that pain starts all over again

Fast Forward to August 2023 (+20 Months)

Extending Python to manage Z: pyracf

Add

Create a new data set profile.

`DataSetAdmin.add()`

```
def add(
    self,
    data_set: str,
    traits: dict = {},
    volume: Union[str, None] = None,
    generic: bool = False
) -> Union[dict, bytes]:
```

Connect

Create or change a group connection.

`ConnectionAdmin.connect()`

```
def connect(self, userid: str, group: str, traits: dict = {}) -> Union[dict, bytes]:
```

	“pyracf”	“pyRACF”
Repository	https://github.com/wizardofzos/pyracf	https://github.com/ambitus/pyRACF
Documentation	Document the code? Why do you think they call it code?	Very extensive github.io docs at https://ambitus.github.io/pyracf/
Pip Installation	pip install piracy	Not Available
Goal	Make working with RACF unloads easier via Pandas/Python	Make issuing RACF commands easier via Python
Runtime	Python on Linux/Windows	Python on z/OS

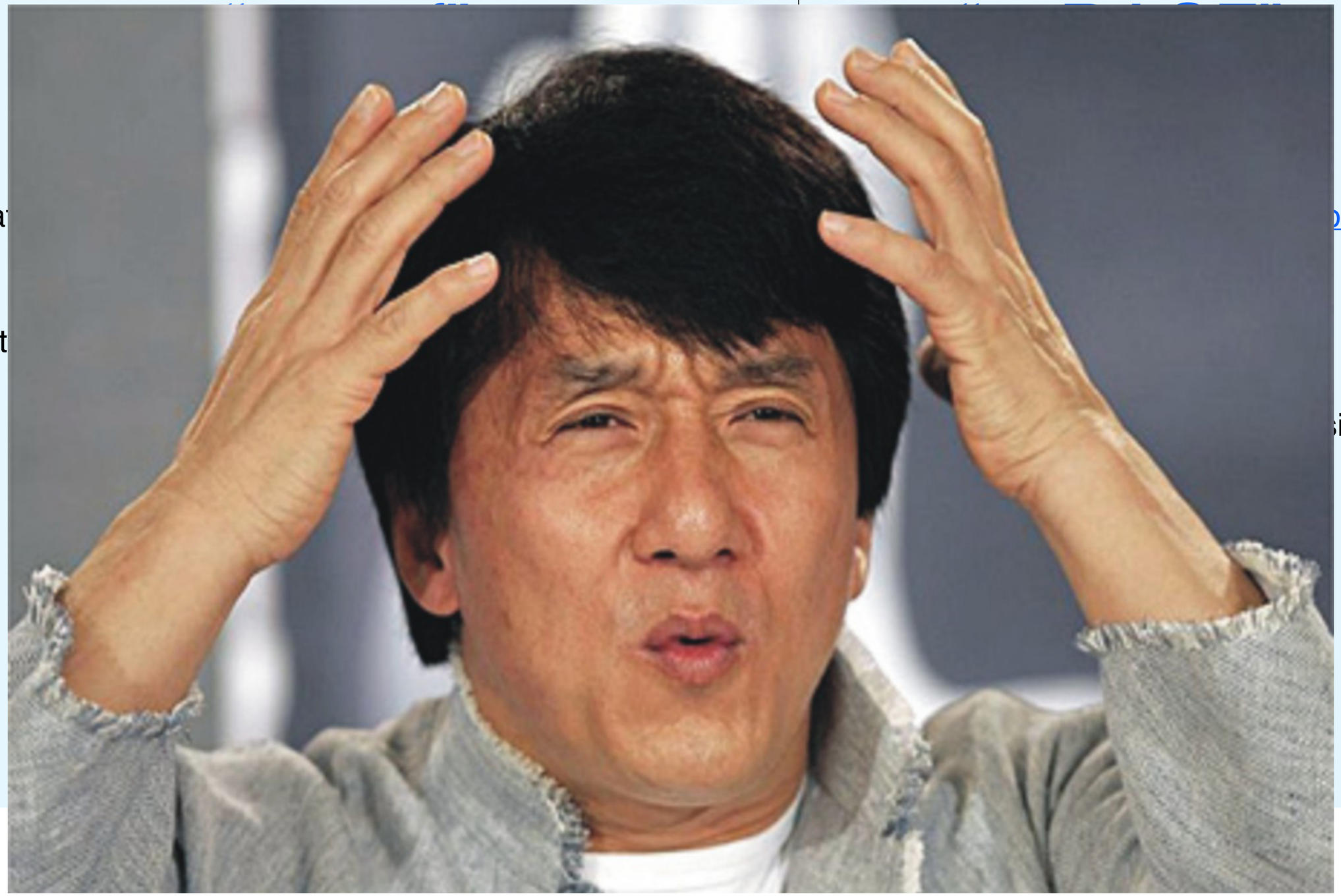
Repository

Documenta ps://

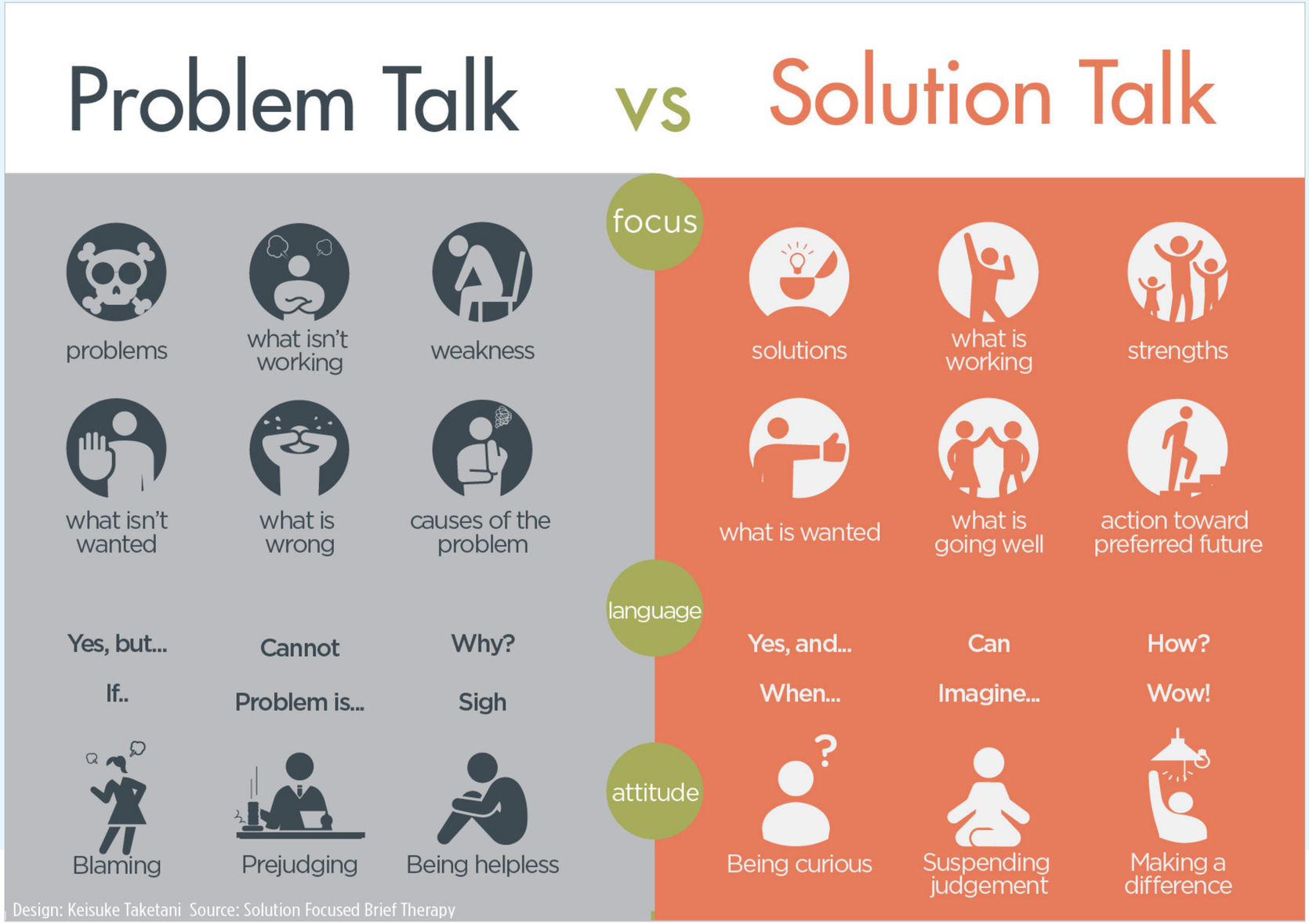
Pip Installat

Goal ier via

Runtime



So we had (a few) solution talk(s)



https://www.linkedin.com/pulse/problem-talk-vs-solution-keisuke-taketani/

And came up with a few options…

1

Just let Frank and his team pick a different name (Henri was first after all)

2

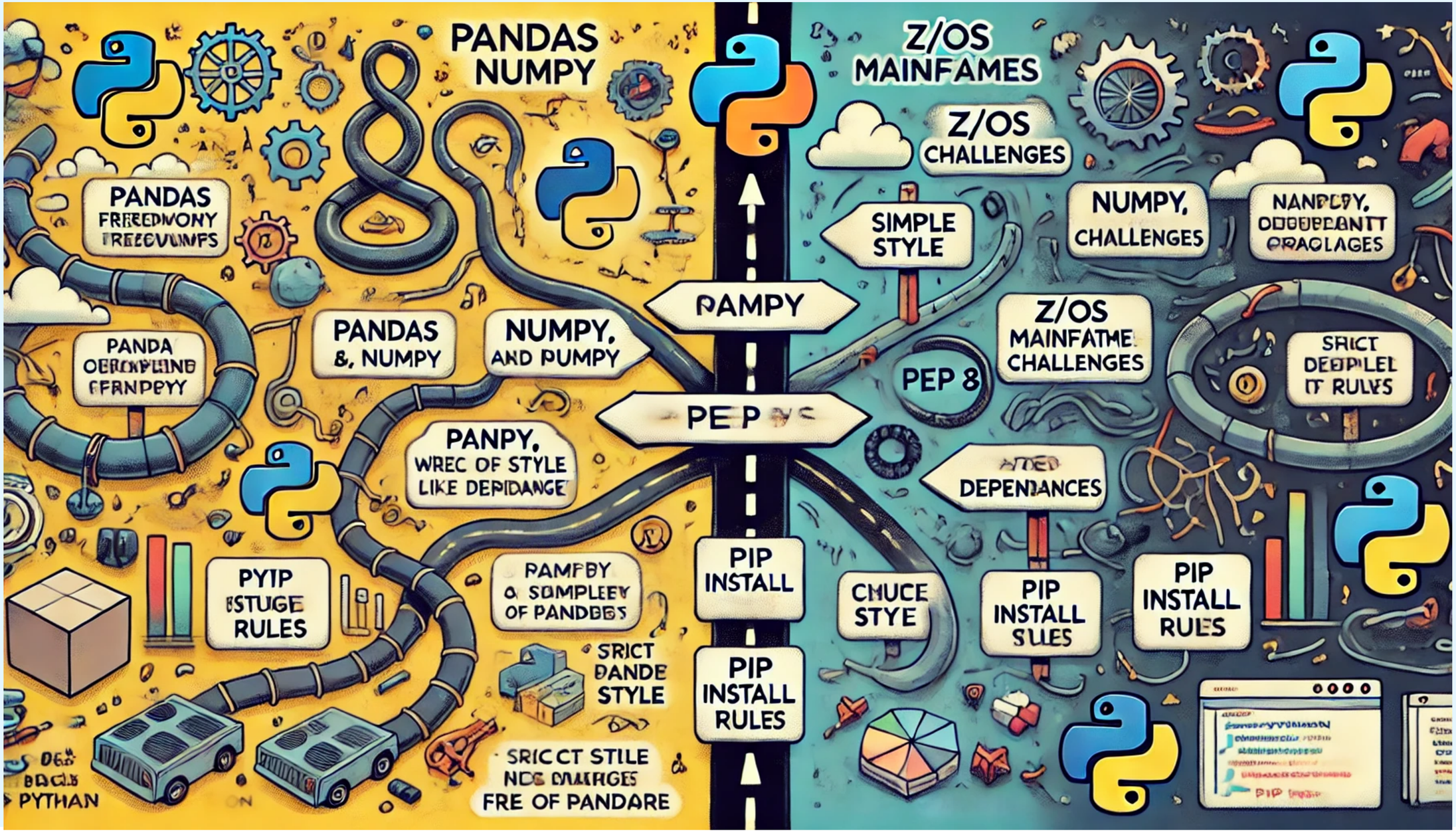
Just let Henri pick a different name (as it's the IBM guys anyway)

3

Make a 'generic' pyRACF that has two submodules (pyRACF-admin and pyRACF-reporter)

4

Merge the code bases into one pyRACF.

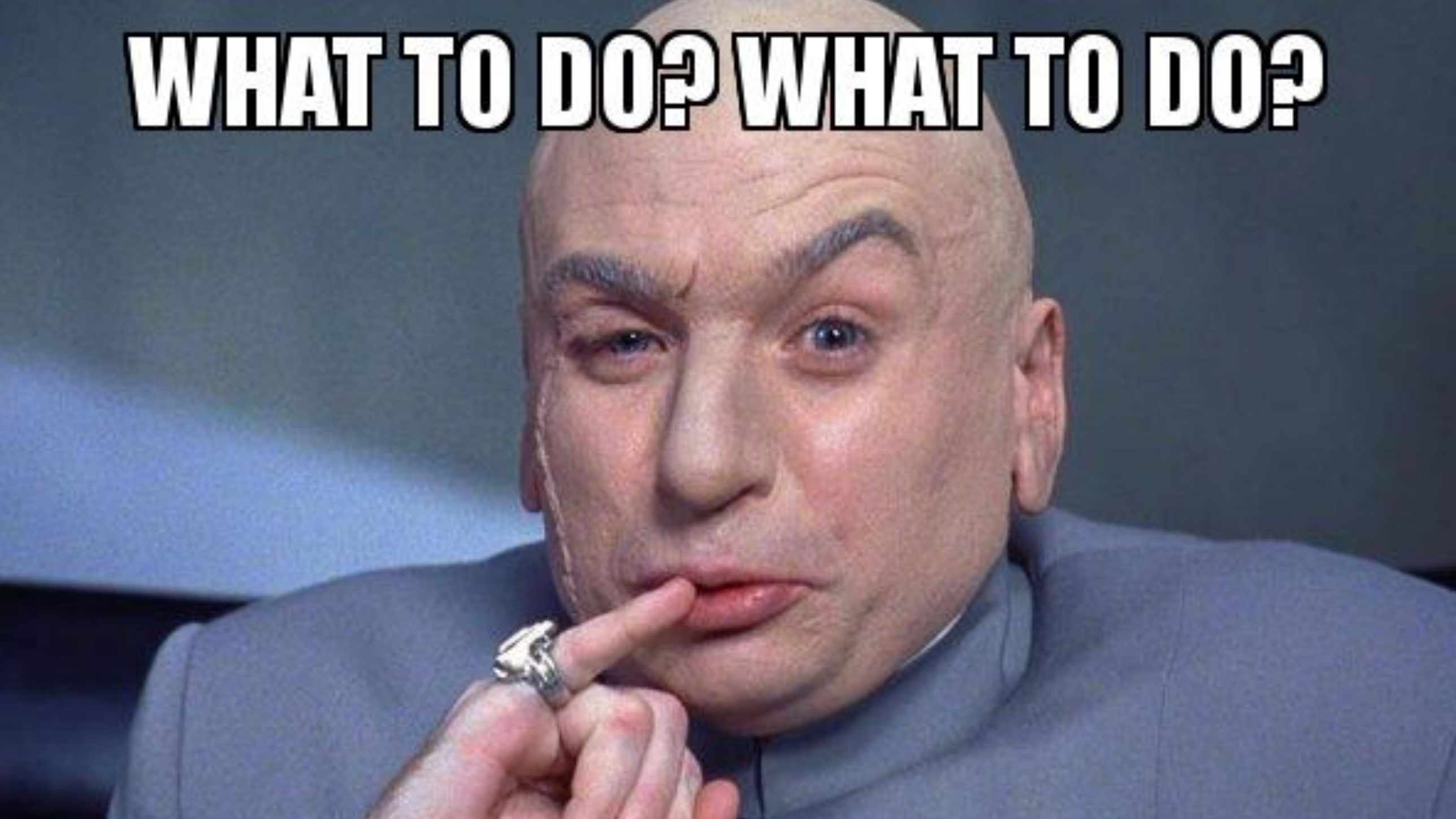

PANDAS
NUMPY
Z/OS
MAINFAMES
Z/OS
CHALLENGES
PANDAS
SIMPLE
STYLE
NUMPY,
CHALLENGES
PAMPY
Z/OS
MAINFATME
CHALLENGES
PANDAS
&, NUMPY
NUMPY,
AND PUMPY
PEP 8
PANPY,
WREC OF STYLE
LIKE DEPIDANGE
DEPENDANCES
PIP
INSTALL
PIP
INSTALL
RULES
PYIP
RULES
PIP
INSTALL
SUES
PIP
INSTALL
RULES
SRICT
DANDE
STYLE
PIP
INSTALL
RULES
SRICT STILE
FRE OF PANDARE
PYTHAN

Issue when trying #3 and or #4

- One version of pyRACF depended on Pandas (which in turn depends on NumPy), creating complications for z/OS users since Pandas wasn’t pip-installable without workarounds. Meanwhile, the other pyRACF version operated independently of Pandas altogether.

- One pyRACF adhered strictly to "PEP 8" coding guidelines, while the other was more casual, being "just a functional pile of Python code" (and yes, that second one was mine!).

- Initially, merging them under a single package seemed feasible (e.g., using from pyracf import pyracf-reporter vs. from pyracf import pyracf-admin). However, trying to make pip install behave differently based on runtime conditions became cumbersome—and Frank definitely isn’t a fan of anything clunky!


WHAT TO DO? WHAT TO DO?

I wanted more features

- DCOLLECT
- RMMEXTRACT
- Proclib members…
- Basically have Mainframe Data in Pandas Dataframe
- So MFPandas*was born

- *unintentional double entendre

PyPI <noreply@pypi.org> Fri, Jul 26, 9:25 AM
to wizardofzos

A new collaborator has been added to a project you own on PyPI:

Username: lcarcaramo
Role: Maintainer
Collaborator for: pyracf
Added by: wizardofzos

PyPI <noreply@pypi.org> Wed, Sep 25, 1:28 PM
to wizardofzos

A collaborator's role was changed on a project you own on PyPI:

Username: lcarcaramo
New role: Owner
Collaborator for: pyracf
Changed by: wizardofzos

PyPI <noreply@pypi.org> Thu, Sep 26, 8:23 AM
to wizardofzos

You have been removed as collaborator by lcarcaramo from pyracf on PyPI.

	“MFPandas”	“pyRACF”
Repository	https://github.com/wizardofzos/pyracf	https://github.com/ambitus/pyRACF
Documentation	Extensive documentation at https://mfpandas.readthedocs.io	Very extensive github.io docs at https://ambitus.github.io/pyracf/
Pip Installation	pip install mfpandas	Not Available
Goal	Make working with various Mainframe Data unloads easier via Pandas/ Python	Make issuing RACF commands easier via Python
Runtime	Python on Linux/Windows	Python on z/OS

PyRACF “Demo”

Slides provided by Frank (thanks Frank!)

(Check him out https://medium.com/@degilio)

Hope I’m not murdering the PyRACF explanation :)

pyRACF:

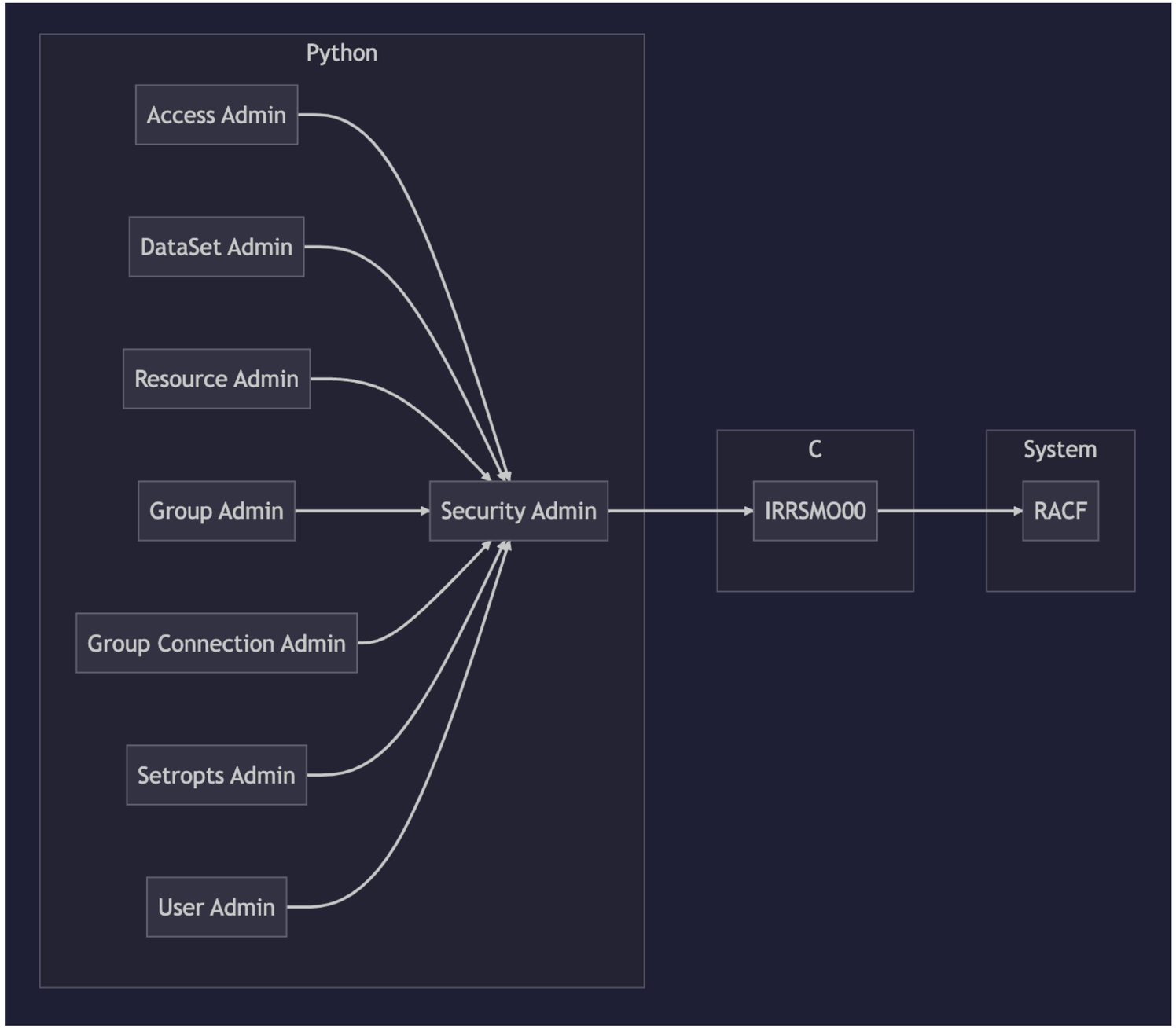
- Provides interfaces to perform z/OS security administrative tasks from python.
- Open-source developed by IBMers
- ***pyRACF*** *is currently in* ***Beta***

Link:

- pyRACF: https://github.com/ambitus/pyracf

pyRACF Dependencies*:*

- *z/OS* ***2.4*** *and higher.*
- ***R_SecMgtOper (IRRSMO00)****: Security Management Operations.*
  - *Authorization: For many common commands, READ access to the IRR.IRRSMO00.PRECHECK resource in the XFACILIT class is required.*
  - *IRRSMO00 authorization details here.*



Welcome to the Star Wars Universe

```
from pyracf import *
```

```
debug_flag=False
```

```
user_admin=UserAdmin(debug=debug_flag)
```

Let's Start by identifying some characters ¶

```
lukestraits = {
    "base:name" : "Luke Skywalker",
    "base:password": "4CBWITHU",
    "base:owner": "LUKESKY",
    "base:special": False,
    "base:operations": False,
    "omvs:uid": 16711680,
    "omvs:home_directory": "/u/lukesky",
    "omvs:default_shell": "/bin/sh",
    "tso:account_number": "D999",
    "tso:default_region_size" : 1096128,
    "tso:logon_procedure" : "IKJACCNT"
}
```

```
user_admin.add("LUKESKY",traits=lukestraits)
```

```
user_admin.extract("LUKESKY")
```

```
[5]: user_admin.add("LUKESKY",traits=lukestraits)
```

```
[5]: {'securityResult': {'user': {'name': 'LUKESKY',
    'operation': 'set',
    'requestId': 'UserRequest',
    'commands': [{'safReturnCode': 0,
      'returnCode': 0,
      'reasonCode': 0,
      'image': 'ADDUSER LUKESKY ',
      'messages': ['ICH01024I User LUKESKY  is defined as PROTECTED.']},
     {'safReturnCode': 0,
      'returnCode': 0,
      'reasonCode': 0,
      'image': "ALTUSER LUKESKY     NAME        ('Luke Skywalker') PASSWORD    (***
OMVS     (UID         (16711680) HOME        ('/u/lukesky') PROGRAM     ('/bin/sh'
OC        (IKJACCNT))"}]},
   'returnCode': 0,
   'reasonCode': 0,
   'runningUserid': 'run_20240614_0952.log'}}
```

```
[6]: user_admin.extract("LUKESKY")
```

```
[6]: {'securityResult': {'user': {'name': 'LUKESKY',
    'operation': 'listdata',
    'requestId': 'UserRequest',
    'commands': [{'safReturnCode': 0,
      'returnCode': 0,
      'reasonCode': 0,
      'image': 'LISTUSER LUKESKY ',
      'profiles': [{'base': {'user': 'lukesky',
        'name': 'luke skywalker',
        'owner': 'lukesky',
        'created': '6/14/2024',
        'defaultGroup': 'sys1',
        'passwordDate': None,
        'passwordInterval': 186,
        'passphraseDate': None,
```

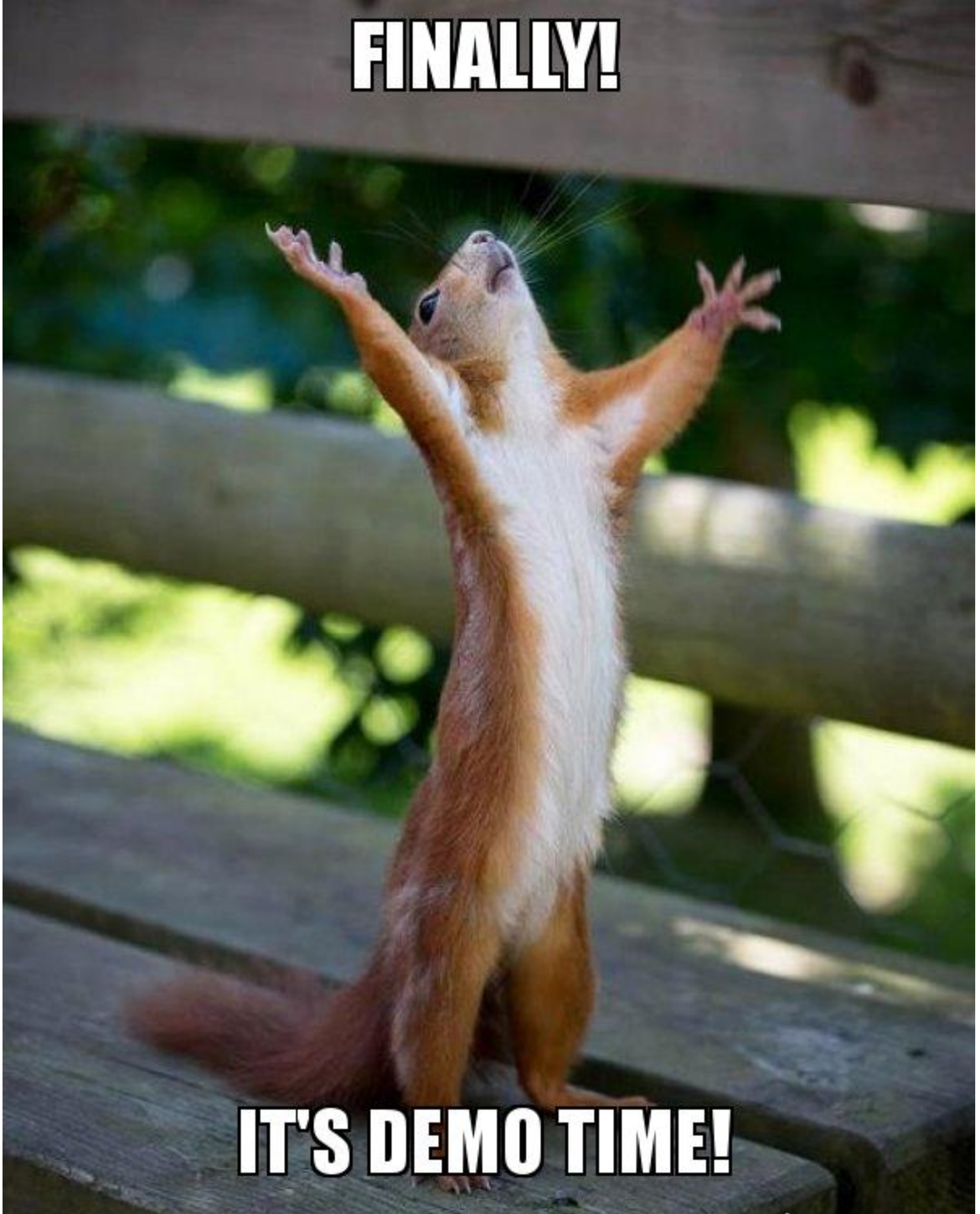
```
[ ]: anastraits = {
         "base:name": "Anakin Skywalker",
         "base:password": "4CBWITHU",
         "base:owner": "Anasky",
         "base:special": False,
         "base:operations": False,
         "omvs:uid": 65535,
         "omvs:home_directory": "/u/anasky",
         "omvs:default_shell": "/bin/sh",
         "tso:account_number": "D999",
         "tso:default_region_size" : 1096128,
         "tso:logon_procedure" : "IKJACCNT"
     }

[ ]: user_admin.add("ANASKY",traits=anastraits)

[ ]: user_admin.extract("ANASKY")
```

Now let's put em in some groups

```
[ ]: group_admin=GroupAdmin(debug=debug_flag)

[ ]: connection_admin=ConnectionAdmin(debug=debug_flag)

[ ]: connection_admin.connect("LUKESKY","JEDI")

[ ]: connection_admin.connect("ANASKY","JEDI")
```

```
[12]: connection_admin.connect("LUKESKY","JEDI")

[12]: {'securityResult': {'groupConnection': {'name': 'LUKESKY',
          'group': 'JEDI',
          'operation': 'set',
          'requestId': 'ConnectionRequest',
          'commands': [{'safReturnCode': 0,
            'returnCode': 0,
            'reasonCode': 0,
            'image': 'CONNECT LUKESKY  GROUP       (JEDI)'}]},
        'returnCode': 0,
        'reasonCode': 0,
        'runningUserid': 'run_20240614_0952.log'}}
```

```
[13]: connection_admin.connect("ANASKY","JEDI")

[13]: {'securityResult': {'groupConnection': {'name': 'ANASKY',
          'group': 'JEDI',
          'operation': 'set',
          'requestId': 'ConnectionRequest',
          'commands': [{'safReturnCode': 0,
            'returnCode': 0,
            'reasonCode': 0,
            'image': 'CONNECT ANASKY  GROUP       (JEDI)'}]},
        'returnCode': 0,
        'reasonCode': 0,
        'runningUserid': 'run_20240614_0952.log'}}
```



We’re gonna try a live demo…

https://github.com/wizardofzos/LasVegas-2874


Thank You For Your Attention!
Presentation is over!

Experience more here at IBM TechXchange

MANDATORY FINAL SLIDE of your presentation. ADD any recommended content related to yours here– amas, labs, sessions, but they must come AFTER your session, and not more than 3-4, Replace ones if you need room

Come to our IBM Z and LinuxONE Sandbox #850

Experience the world famous plexi and lego! Engage our SME's, demos, AMAs, community & skills

A snapshot of some of the great topics

AI for mainframe app dev: #2341, 3059, 3766, 1983

New chip set, simplification & zNext: #3761, 3063, 3520

Mainframe AI assistant for ops: #3767, 3060, 1950

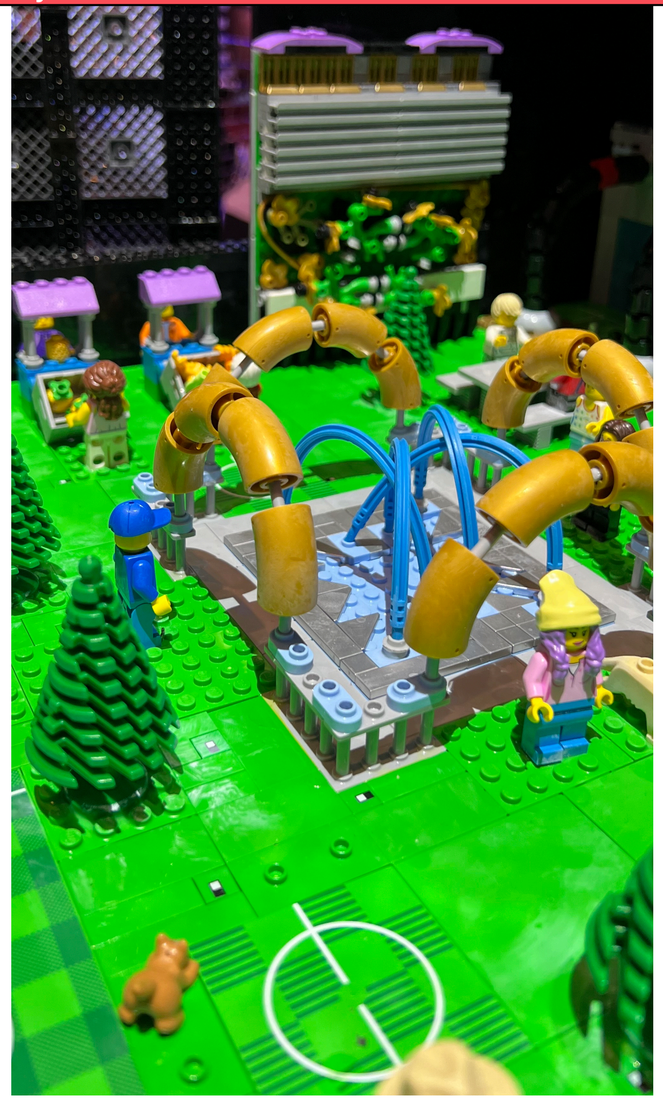
Threat detection & Cyber Vault: #2559, 3773

OTel, IntelliMagic & Instana: #3768, 3762, 1729

All things Data & AI: #3799, 3213, 1969, 3319, 1969

DevOps, testing & more #1186, 3212, 3058

Skills #3214, 1889,1890



IBM Z and LinuxONE content in TechXchange catalog