# Modernizing business critical applications with open languages on z/OS

—

Day 2

Deep-dive and demos of the open-source languages on z/OS – Go, Node.js, and Python

# Presenters

## Node.js

**Wayne Zhang**

**Software Developer, z/OS Node.js**

**Yves Tolod**

**IBM Z Client Technical Professional**

## Python

**Austin Wells**

**Python Compiler Developer**

**Waleed Q Khan**

**Python Compiler Developer**

## Go

**Bill O'Farrell**

**Team lead for Go Language**

**Mahdi Hosseini**

**Go Compiler Developer**

# Panelists

**Jennifer Rowan**
**Offering Manager - Python and Node.js on z/OS**

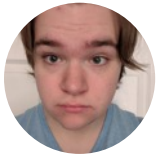
**Wayne Zhang**
**Software Developer, z/OS Node.js**

**Rosanne Jolin**
**Manager and Delivery Manager, Python and Node.js for z/OS**

**Yves Tolod**
**IBM Z Client Technical Professional**

**Waleed Q Khan**
**Python Compiler Developer**

**Chad McIntyre**
**IBM Open Enterprise Python for z/OS Dev Lead**



**Austin Wells**
**Python Compiler Developer**

**James Tang**
**Offering Manager – Go and Java on z/OS**

**Larry Lindsay**
**Senior Development Manager and Security Architect**

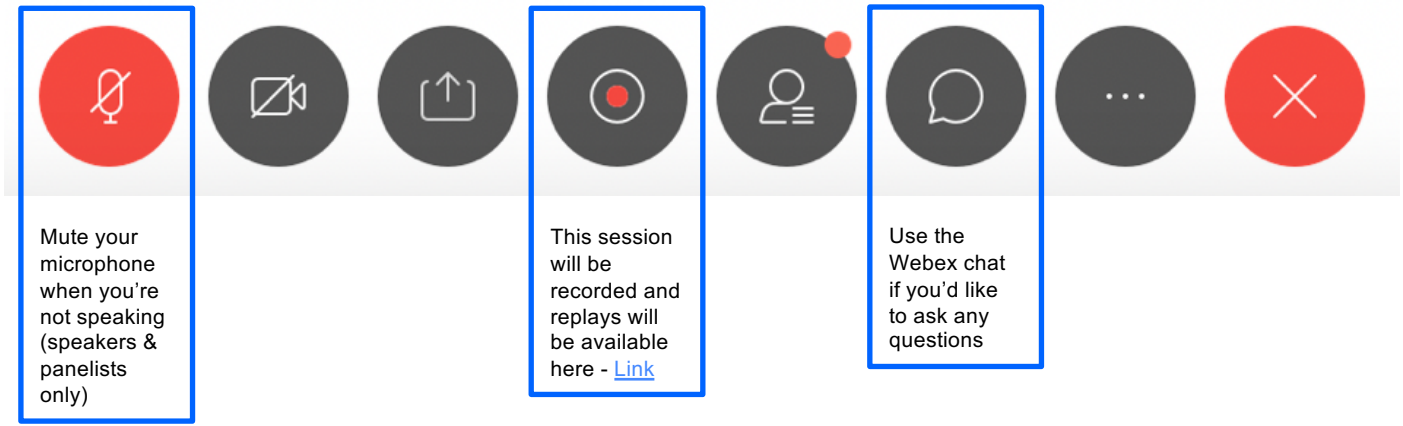
**Bill O'Farrell**
**Team lead for Go Language**

# Instructions

- If you do not hear audio, check/change your audio connection by clicking either the Phone icon in the Webex controls (if available) or the (...) icon then Switch Audio

- Our technical team is standing by and ready to answer your questions through the Q&A panel. Please use the Q&A panel to post your questions.

- The slides for this presentation and some other resources are available for download in the chat window

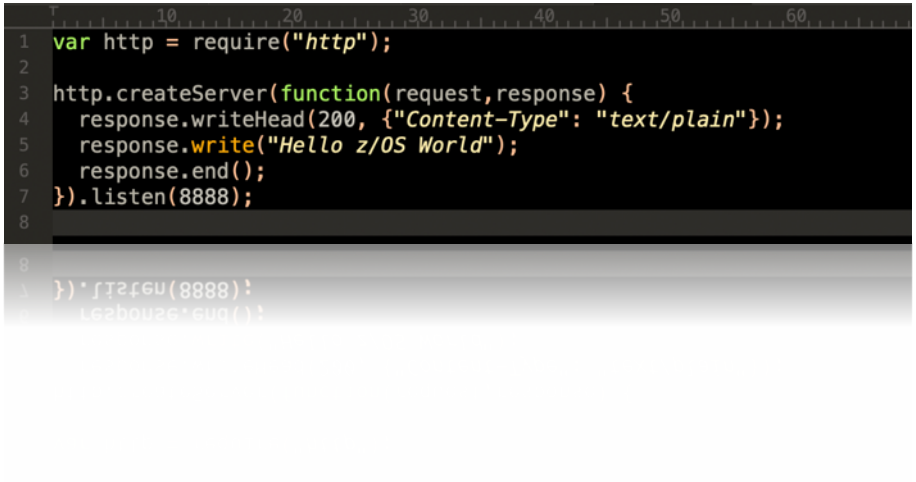


# IBM Open Enterprise SDK for Node.js

# What is Node.js?

- **Server side JavaScript runtime**
  Designed to build scalable network applications
- Lightweight and efficient
- Uses an event-driven, single-threaded, non-blocking I/O model
- Best suited for data-intensive (i.e. I/O bound) applications
- Provides a module-driven, highly scalable approach to application design and development that encourages agile practices

## ‘Hello World’ Web Application

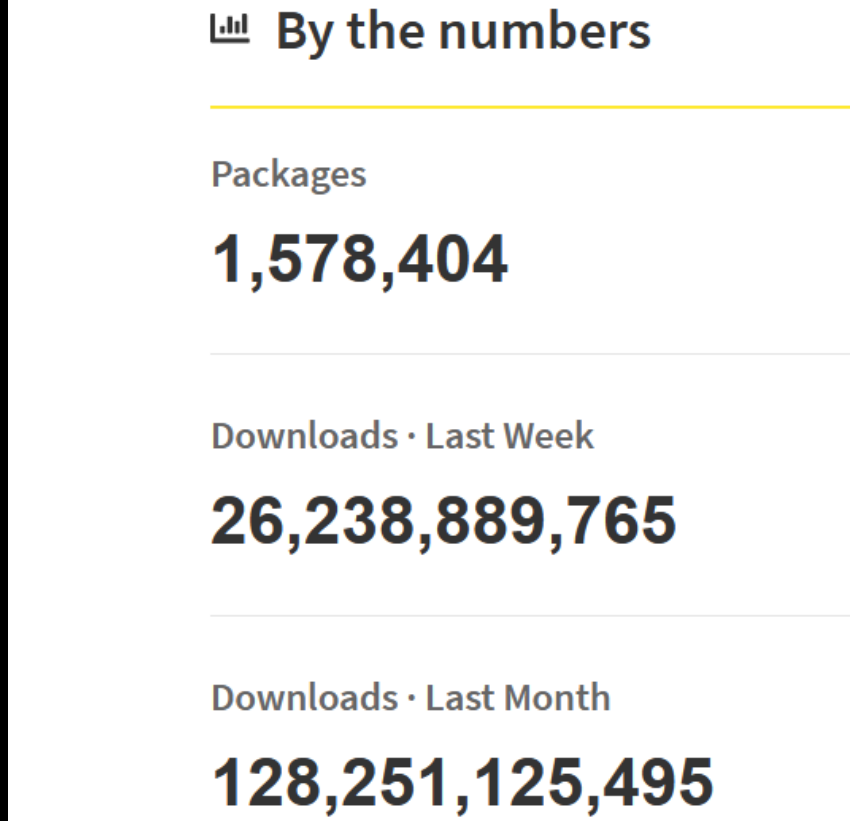


```
var http = require("http");

http.createServer(function(request,response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("Hello z/OS World");
  response.end();
}).listen(8888);
```

# Largest repository of modules

- **NPM (Node Package Manager)**
- **Repository of community contributed modules**
- **1.5M modules and growing**
- **3x growth rate vs other runtimes**



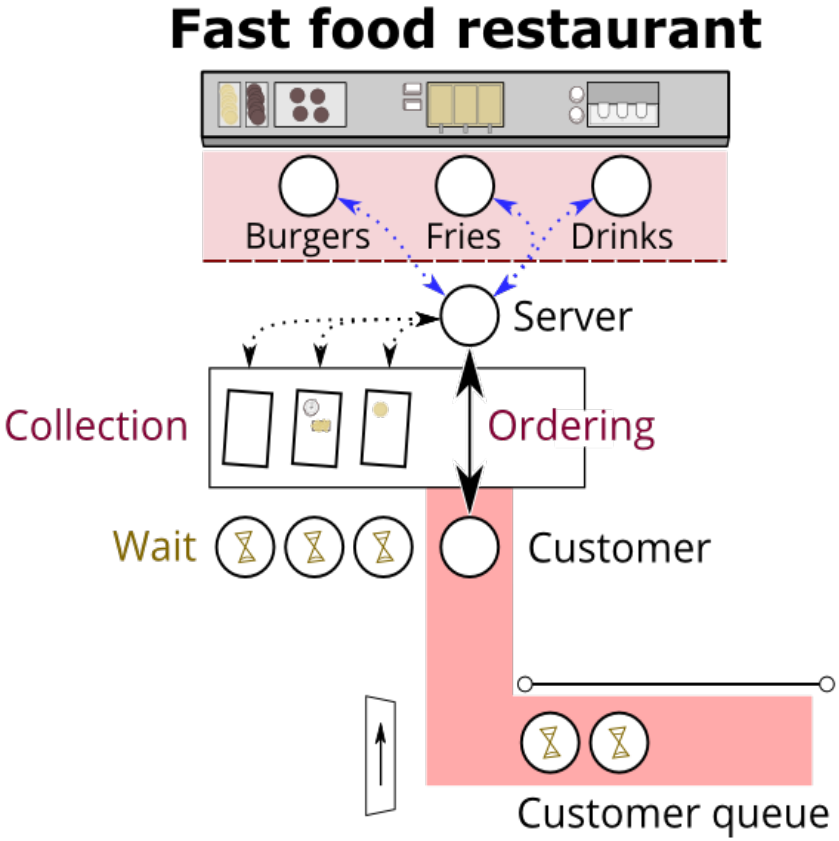

Data source: Link

# How Node.js Works?

## The Event Loop

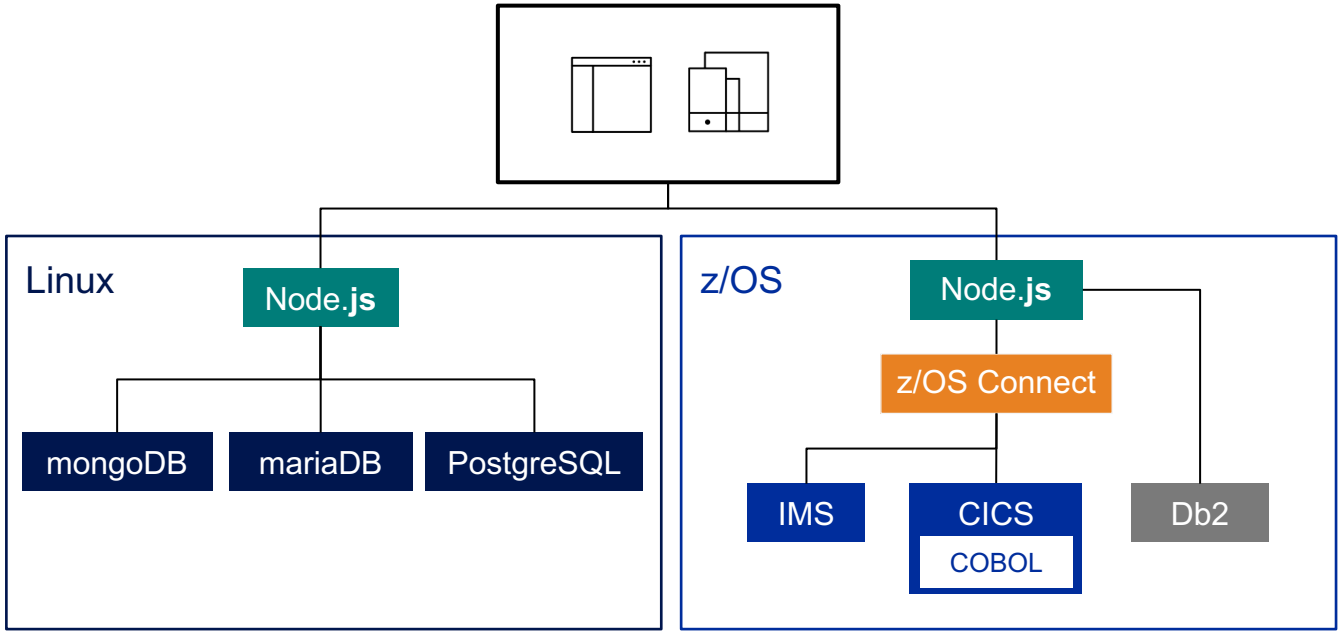
Allows Node.js to perform non-blocking and asynchronous operations

Node.js is a Single- threaded Application
Supports concurrency via events and callbacks
Loop that listens for events, and then triggers a callback function when one of those events is detected



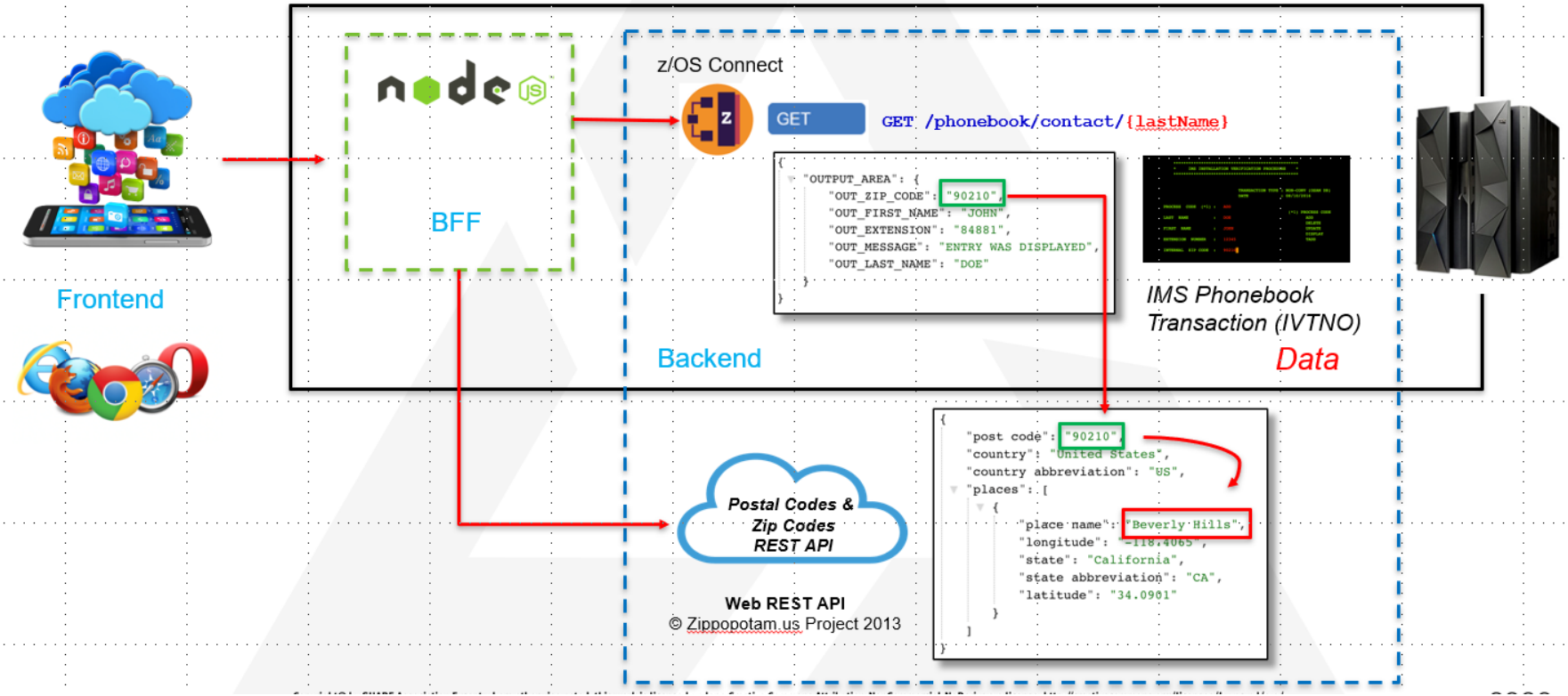
# Why use Node.js on z/OS

Co-locate applications and data on IBM Z to deliver better throughput and response time



# Why use Node.js on z/OS?

*Put your Backend for Frontend (BFF) closer to your data*



# Connect to your z/OS assets

**z/OS Connect Enterprise Edition (EE)**

- Access z/OS assets that are exposed through IBM z/OS Connect EE
- zosconnect-node: link
- Loopback connector for z/OS Connect EE: link

**CICS**

- Can deploy Node.js application in CICS the same way as COBOL: link

**Db2**

- Provide direct access to Db2 on z/OS via npm module: link

**VSAM**

- Interact with z/OS VSAM datasets and records via npm module: here

# Connect to your z/OS assets

**z/OS Node Accessor**

- Module to interact with z/OS MVS dataset and z/OS UNIX files and simple JCL operations: link

**zREXX**

- It calls z/OS REXX scripts residing in PDS from Node.js: link

**RACF**

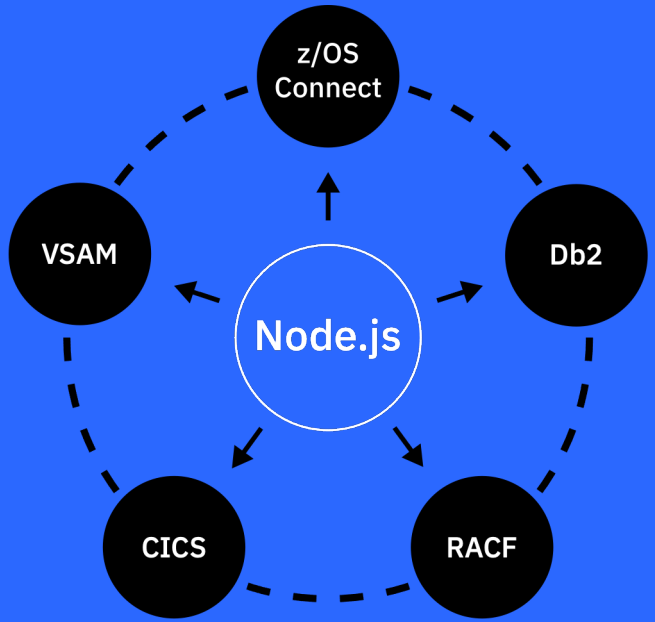
- This Node.js module enables your application to validate against RACF: link

**BPXWDYN** Dynamic Allocation

- Node.js interface to the BPXWDYN program, used for dynamic allocation: link

**zCRYPTO** - Interface to RACF Keyrings

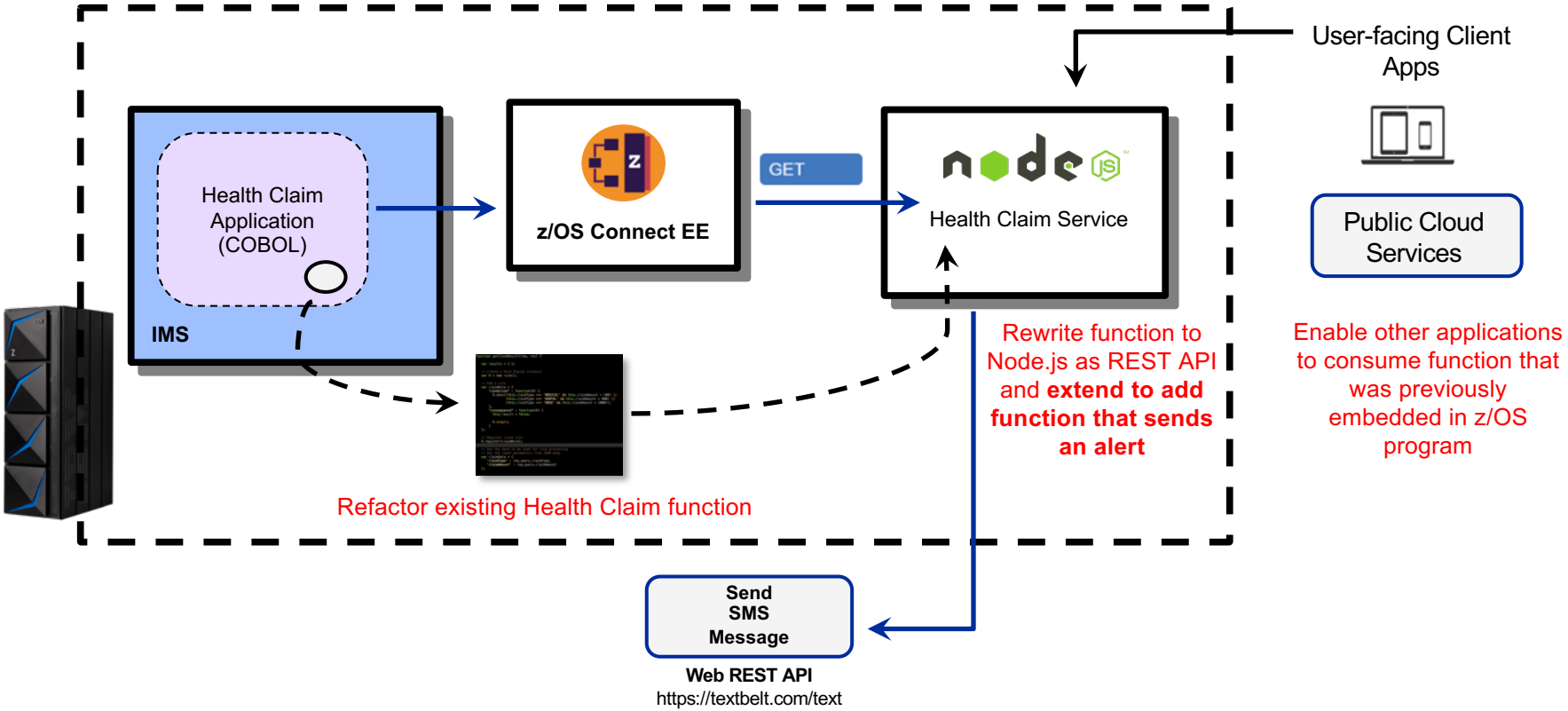
- Provides APIs to RACF key rings link



# Demo time!

# Refactor elements of an existing z/OS application into discrete services

*Rewriting a COBOL function as a Microservice in Node.js and invoked as a REST API*



# Demo Take-aways

- Revitalize existing monolithic application with APIs
- Develop new functions using open languages and invoked as APIs to extend existing core applications on z/OS
- Refactor specific functions of existing z/OS application using open languages to simplify maintenance

# IBM Open Enterprise SDK for Python

# What is Python?

Python is an interpreted, object-oriented high-level language with simple and easy to learn syntax.

Python encourages code reuse through packages & modules, which are available through the Python Package Index (PyPI) repository, where users can share open-source projects.

# Extensive Package Ecosystem

The Python Package Index (PyPI) is a repository of community contributed packages  for the Python programming language

There are over 200,000 packages covering all domains (web hosting, scientific computing, databases, etc.)

# What's included in IBM Open Enterprise SDK for Python

**Full port of Python™ 3.9 to z/OS®**
**UNIX Systems Services:**

- The cpython interpreter
- The Python standard library
- Enhancements to support interaction with EBCDIC data
- Support for ctypes and calling native language libraries
- Distutils customization to enable packaging Python applications on z/OS

# z/OS tailored PyPI packages

- numpy tuned for IBM Z® hardware
- cffi
- cryptography
- ebcdic
- zos_util (IBM Python module for file tagging utilities for z/OS)
- pycparser
- six
- pip
- setuptools

# Why Python for z/OS?

IBM Z clients can develop applications with IBM Open Enterprise SDK for Python in the same way as they would on any other platform.

Simple, easy to learn, vast group developers readily available

**By Co-locating** Python applications with **critical assets** (i.e. applications and data) on z/OS you can increase throughput and maintain better security

- Perform advanced analytics from the same machine where the data is stored

**Extend existing code**

- CFFI/Ctypes
- Extensions
- Embedding
- JCL with Z Open Automation Utilities (ZOAU)
- Call from REXX, JCL and etc.

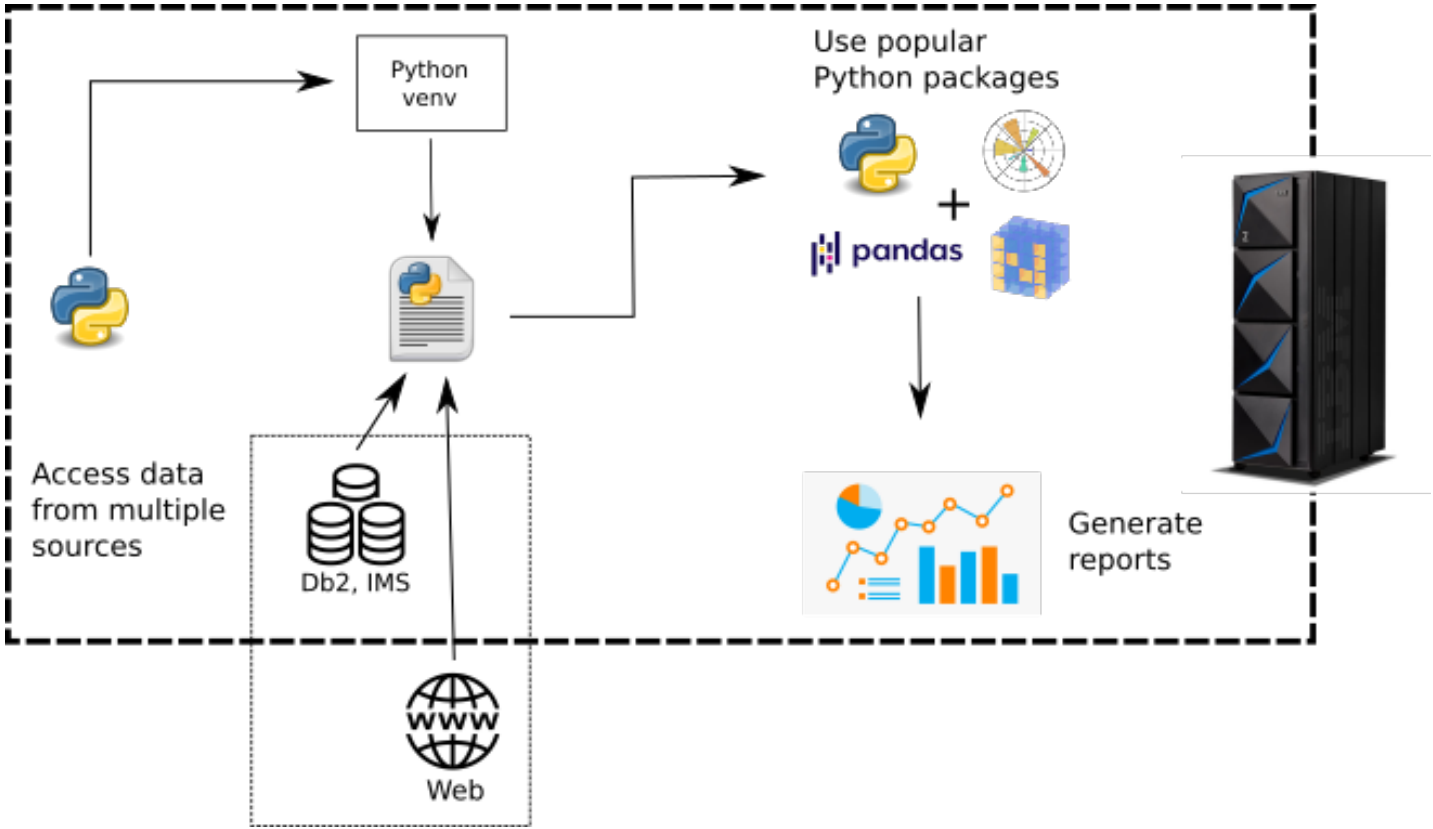
# Ansible

**Infrastructure management tool:**

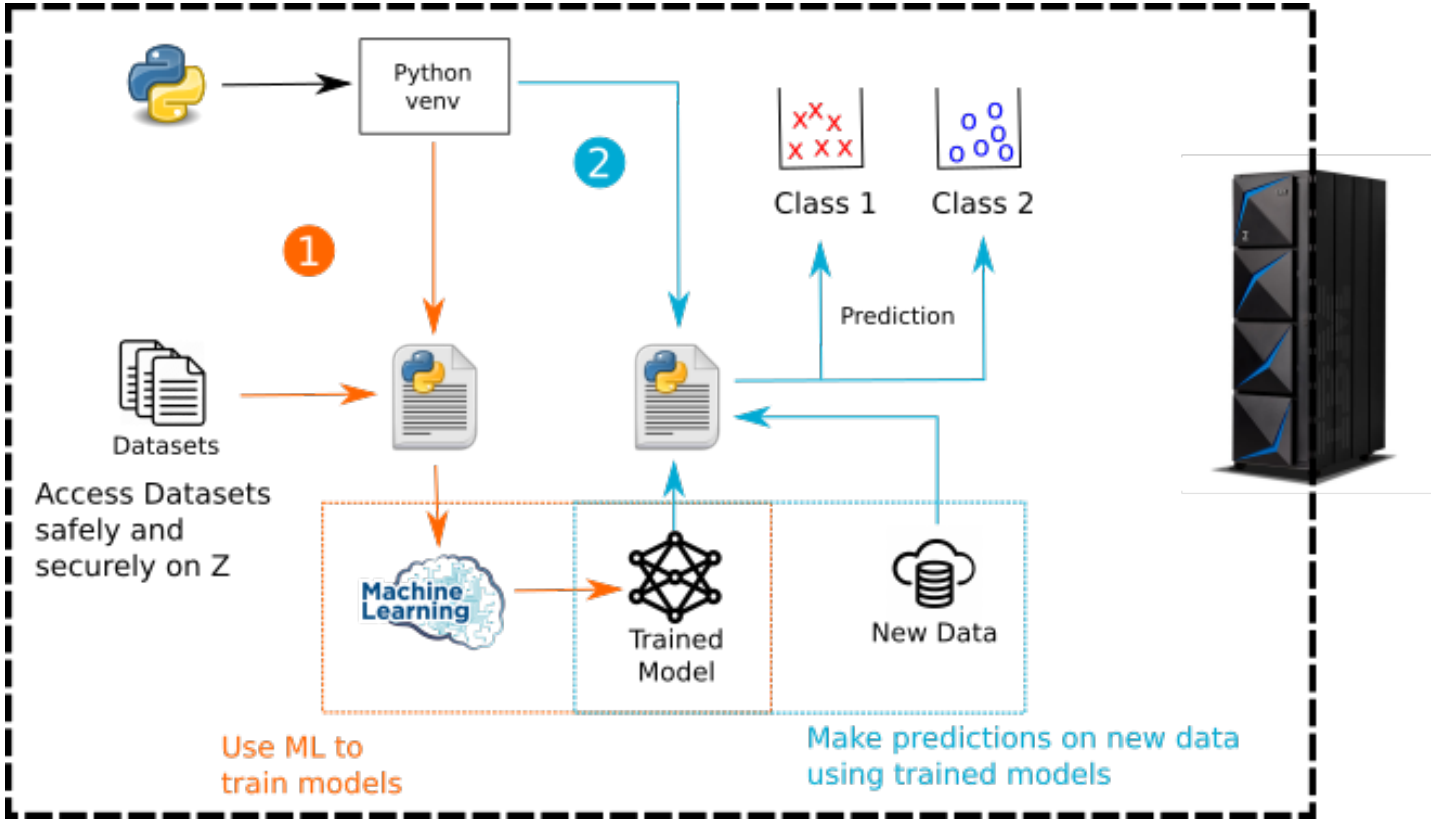
- Written in python
- Can be used to deploy apps, manage machines and more
- Runs entirely through python over ssh

# Demo time!

# Demo 1: Data Analysis with Python



# Demo 2: Machine Learning with Python



# Demo Take-aways

- Safely and securely gain access to dataset contents on z machines with a few lines of python code
- Access data, analyze and generate reports on Z, similar to how you would on a distributed system
- Deploy your machine learning models on the same machine where your data resides

# IBM Open Enterprise SDK for Go

# Go compiler on z/OS

## What

- New language, Go 1.0 released by Google in 2012, now at 1.16
- Design goals:
  - Efficient code, native compiler
  - No included files
  - Interface to C
  - Simple syntax, static typing
  - Built-in framework for testing and benchmarking
  - Built-in concurrency
  - Built in networking
  - Garbage collection
  - Portable community contributions

## Why

efficiency + productivity

Get out of “dependency hell”

When you need to call. C libraries

Programmers can “keep it in their heads”

No excuses for not writing tests as you code

Full use of multi-core power with goroutines, channels and synchronization

All common networking and security protocols + built-in http server

Because modern GC is fast and efficient and prevents obscure memory errors

Thousands of open-source, endian-independent packages available

# Large Ecosystem

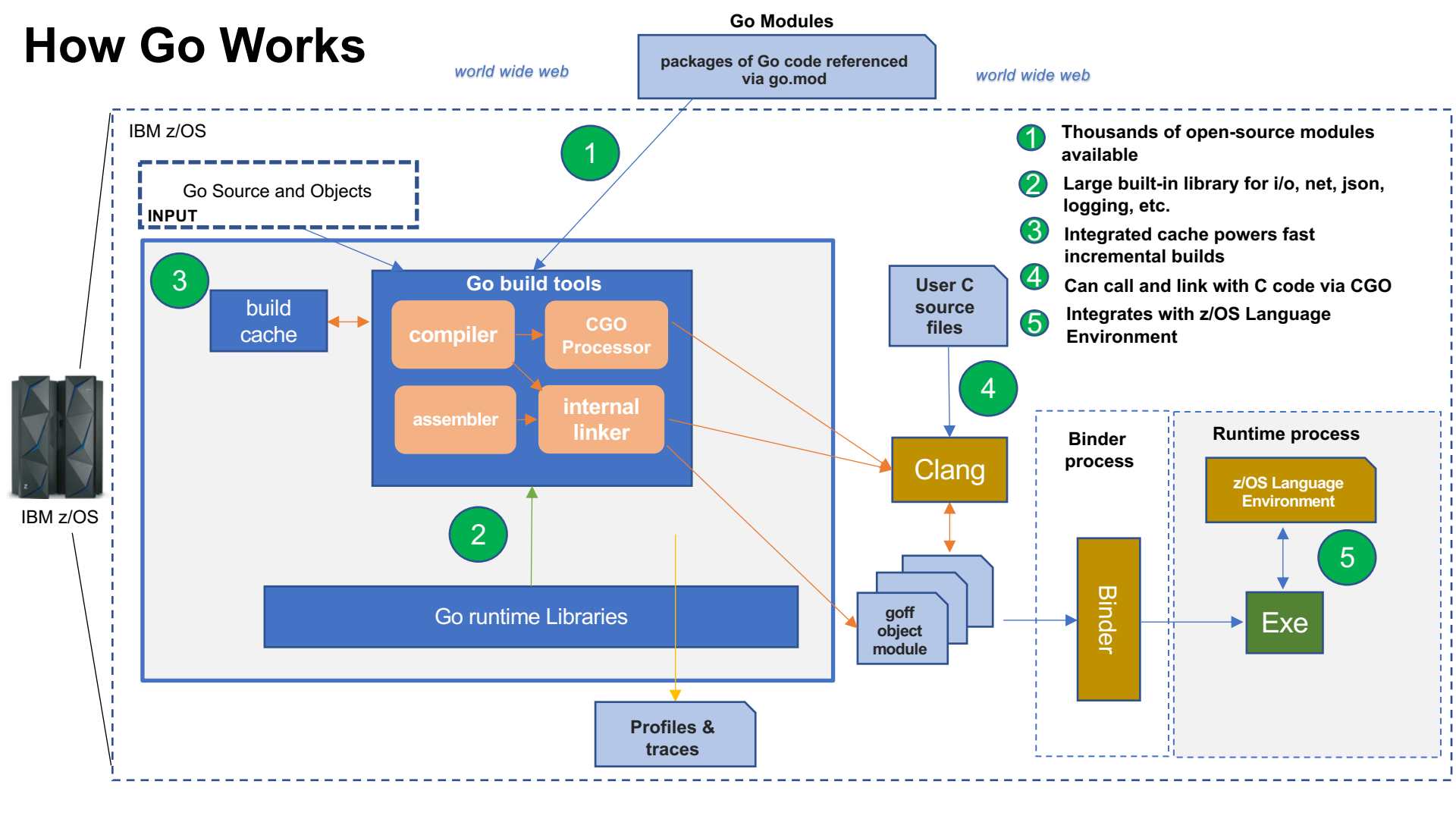
Extensive standard library and growing list of third-party packages

Standard and 3rd party libraries - cryptology, archiving, mail operations, encoding/decoding, networking, error manipulation, OS interfaces, and other popular functions. These packages encourage **code reuse and expedite time to market**.

Go community – additional freely available packages available just by specifying package in import section

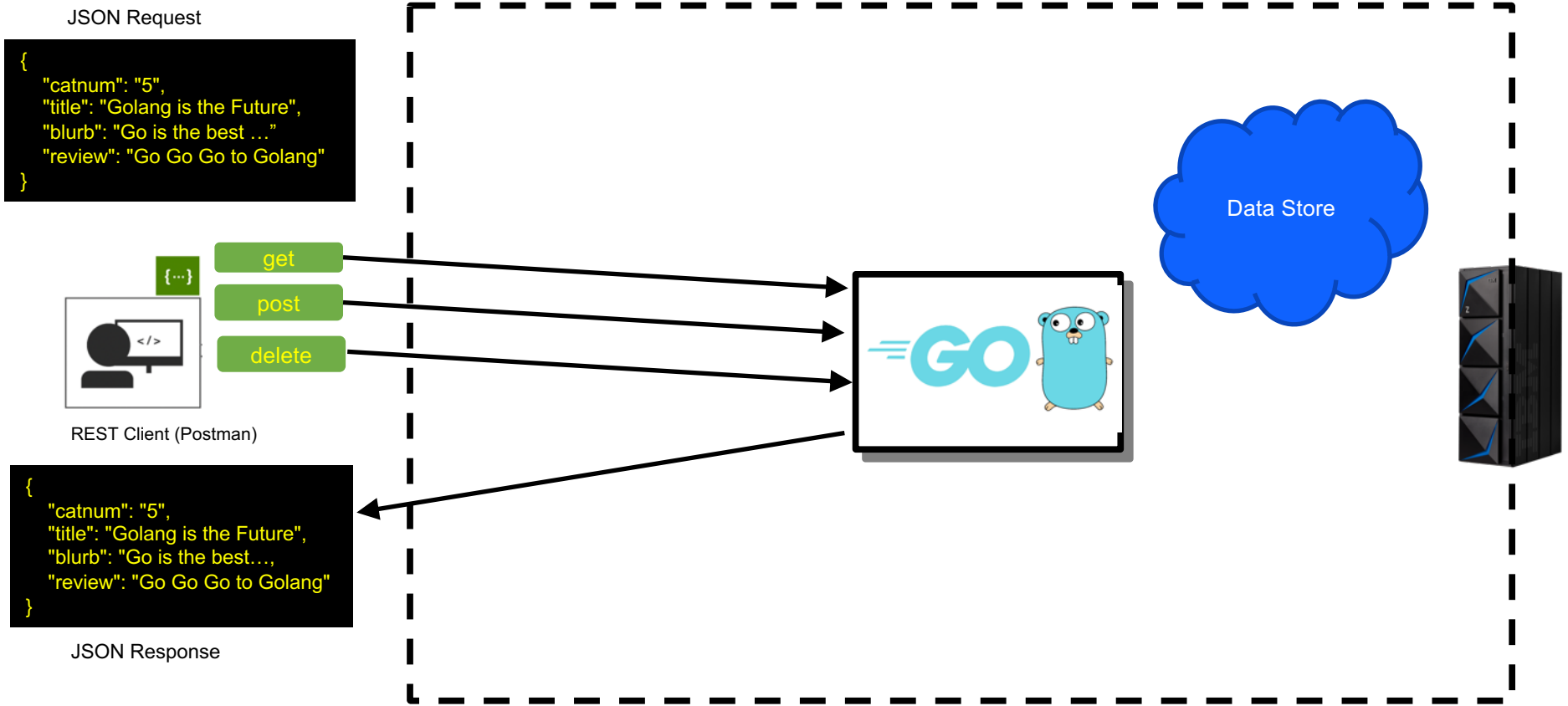
Developers can implement business functions with fewer lines of code, which help **shorten development times and reduce costs**. The continued contribution of modules from the community ensures a steady stream of new libraries and tools

# How Go Works


Go Modules
packages of Go code referenced via go.mod
world wide web
world wide web
IBM z/OS
Go Source and Objects
INPUT
1
Thousands of open-source modules available
2
Large built-in library for i/o, net, json, logging, etc.
3
Integrated cache powers fast incremental builds
4
Can call and link with C code via CGO
5
Integrates with z/OS Language Environment
3
build cache
Go build tools
compiler
CGO Processor
assembler
internal linker
User C source files
4
Clang
2
Go runtime Libraries
goff object module
Binder process
Binder
Runtime process
z/OS Language Environment
5
Exe
Profiles & traces
IBM z/OS

# Demo time!

# Create a RESTful Server



# Demo Take-aways

- Built in Go libraries for many common functions, including json, i/o, logging, webserver
- Use of community-provided module – no porting needed (”mux”)
- Very little code needed to setup a working REST server
- Tight integration with json
- Module versioning is a first-class capability – stability for completed code


謝謝
DZIĘKUJĘ CI
TAPADH LEIBH
KEA LEBOHA
NGIYABONGA
БАЯРЛАЛАА
MISAOTRA ANAO
TEŞEKKÜR EDERIM
WHAKAWHETAI KOE
DANKIE
TERIMA KASIH
DANKON
TANK
TAPADH LEAT
СПАСИБО
GRAZIE
MATUR NUWUN
ХВАЛА ВАМ
MULȚUMESC
РАКМЕТ СІЗГЕ
고맙습니다
GRAZIE
شكرا
FAAFETAI
GO RAIBH MAITH AGAT
ESKERRIK ASKO
БЛАГОДАРЯ
GRACIAS
THANKYOU
HVALA
ТИ БЛАГОДАРАМ
TEŞEKKÜR EDERIM
DANK JE
ΕΥΧΑΡΙΣΤΩ
GRATIAS TIBI
OBRIGADO
TAK
DANKE
AČIŪ
SALAMAT
MAHALO IĀ ʻOE
TAKK SKAL DU HA
MERCI
RAHMAT
MERCI
GRAZZI
ÞAKKA ÞÉR
ありがとうございました
DI OU MÈSI
ĎAKUJEM
HATUR NUHUN
PAXMAT САГА
SIPAS JI WERE
TERIMA KASIH
CẢM ƠN BẠN
UA TSAUG RAU KOJ
WAZVIITA
ТИ БЛАГОДАРАМ
СИПОС
GRÀCIES
SALAMAT
KÖSZÖNÖM
MAHADSANID
HVALA
ДЗЯКУЙ
FALEMINDERIT

# Notices and disclaimers



**U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

# Notices and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

# Please note

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at IBM’s sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those
stated here.


IBM