IBM **CICS**
Transaction Server for z/OS

Modern Language Applications in CICS
Node.js and Java updates

Steve Fowlkes

fowlkes@us.ibm.com

@SAFowlkes

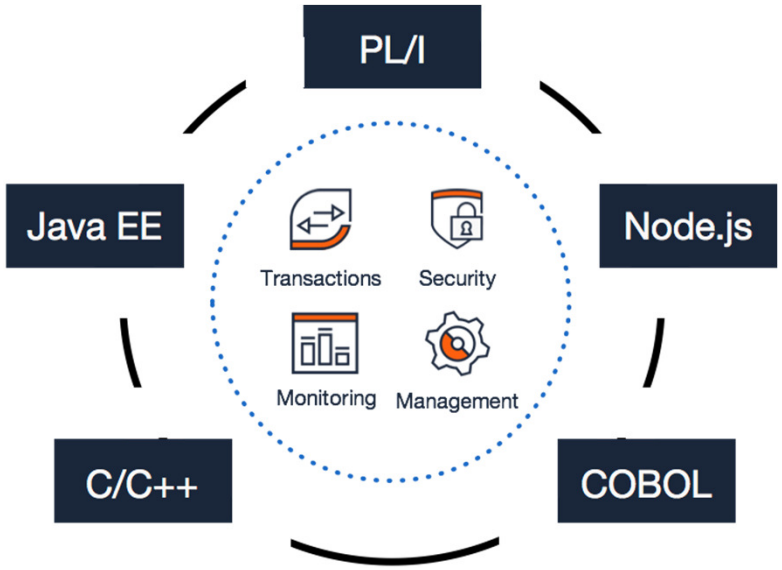
Session Agenda

- Support for Node.js in CICS
  - Introduction
  - Why Node.js in CICS

- Java in CICS
  - Why Java in CICS
  - Latest CICS updates

Unparalleled mixed-language application server

- **IBM CICS Transaction Server** has evolved to become the world's most powerful mixed language application server.
- Applications can share core programming contexts such as **transactionality**, **security**, **monitoring** and **management**, regardless of the language its components are written in, and take full advantage of IBM Z.
- CICS TS V5 allows developers to create incredible mixed-language applications, that include **Java EE 8 Full Platform** capabilities, with first-class interoperability.
- CICS TS V5.5 adds **Node.js** support.



IBM **CICS**
Transaction Server for z/OS

Support for Node.js applications

JavaScript™

First released in Netscape Navigator 2.0 in 1995

Scripting programming language for dynamic content

- Interact with backend servers
- Typically used with HTML and CSS
- User interactions, asynchronous API calls, …
- Language that is used within web browsers

Designed to complement Java with a similar syntax

Standardised as ECMAScript

- 1st edition 1997, latest 10th edition 2019

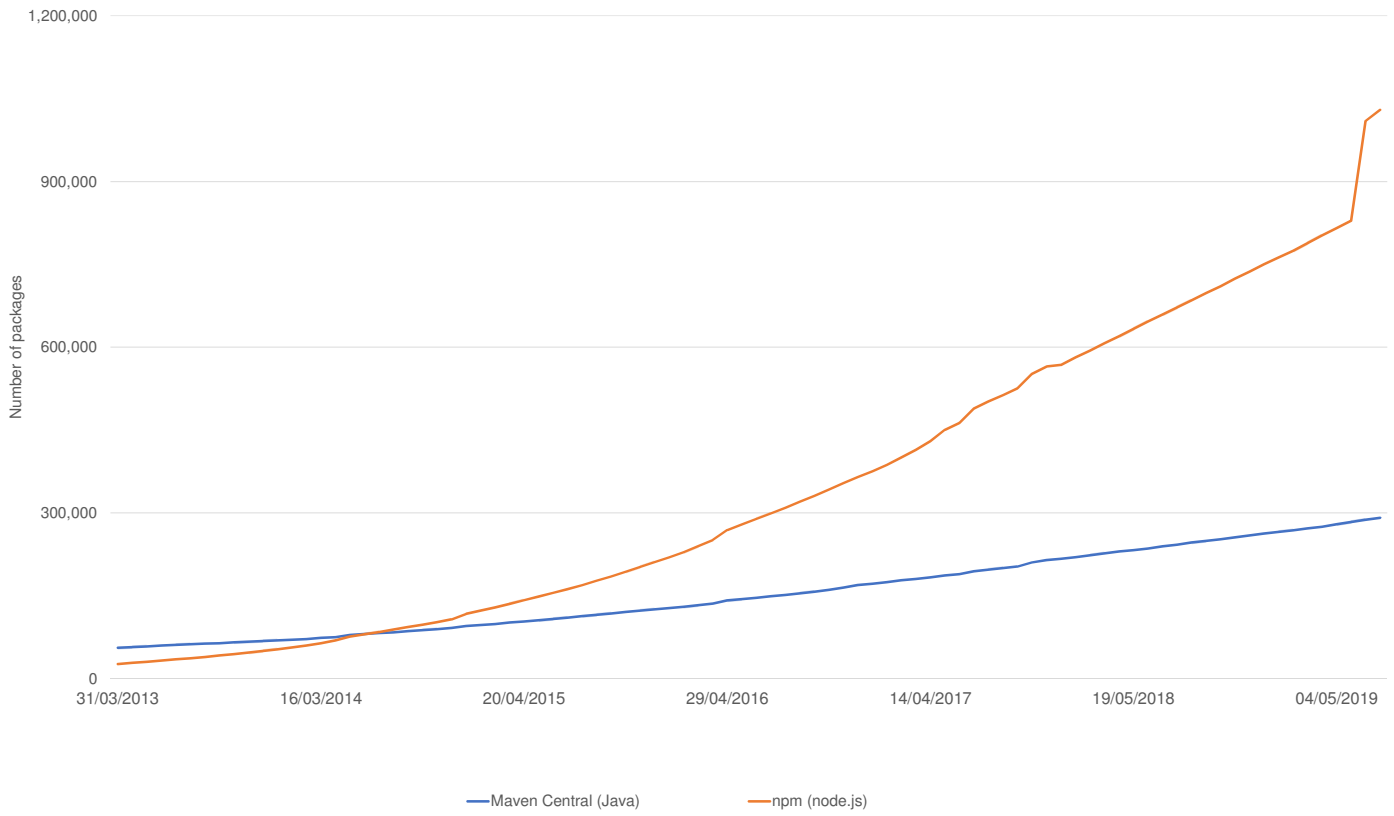
Most browsers embed a JavaScript runtime and compete on its performance

Why do people choose JavaScript?

- Reusable skills
- Huge amount of reusable packages
- Quick time to market
- Isomorphic apps
- High performance

The JavaScript ecosystem of packages

- JavaScript offers unparalleled reuse
- JavaScript has a culture of sharing and reuse
- The ecosystem continues to grow rapidly, coupled with regular releases of the core language specification



modulecounts.com

Node.js

Server-side JavaScript runtime platform

- Governed by the Node.js Foundation
- Built on Google's V8 JavaScript engine

Designed to build scalable network applications

- Best suited for data and I/O intensive applications
- Focused on high network bandwidth applications

Lightweight and efficient

- event-driven
- single-threaded
- non-blocking I/O model
  - leverages the underlying asynchronous I/O support in z/OS

Module-driven approach to application design and development

- Scalable and encourages agile practices
- Becoming a favored choice for digital transformation
  - has the ability to provide and aggregate REST services

Why use JavaScript and Node.js ?

JavaScript is ubiquitous – client, server, cloud, browsers, embedded systems

Large ecosystem of frameworks and tools for application development

- 950K+ modules available via the Node Package Manager (NPM) ecosystem
- Modules are already available for most tasks
  - saving considerable time for Node.js application developers

Fast moving, community driven

- High performance runtimes driven by competition in browsers
- “Battle tested” frameworks

JavaScript on servers

- Leverage huge JavaScript developer ecosystem
- Reuse components, tools, concepts, community

Node.js on z/OS

- Recently updated to Node.js version 12

- Inherit the acclaimed capabilities of IBM Z
  - With full connection to the IBM Z critical assets
  - Leverage the trusted environment of IBM Z
    - maximize the security and uptime of critical Node.js applications

- Co-locating data and the Node.js application on the same system will keep the data secure
  - Control what data will be exposed externally
  - While keeping all confidential data secure on the IBM Z platform

- Node.js runtime on z/OS has the same capabilities as on distributed platforms
  - Extended Node.js packaging and capabilities with z/OS specific extensions
    - the Node.js runtime is available via SMP/E installation
    - Node.js applications can be invoked from OMVS, USS, a JCL job

- Strong Tailored Fit pricing models are available

Why use Node.js in CICS ?

Host APIs and web applications that consume APIs and data on z/OS

- Add logic to existing APIs
- Aggregate APIs and data
- Reuse modules to access external APIs
- Focuses on the server side and exposing backend services as APIs in a programmatic way

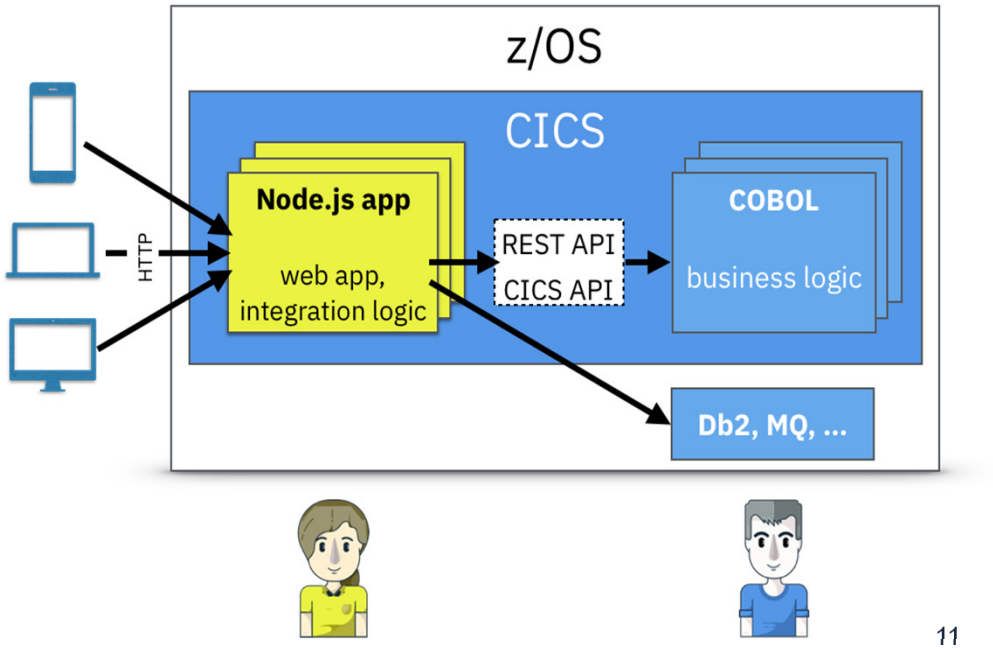
Co-location for optimized response times

Simplified deployment and management with CICS applications

Welcome a new set of API and front-end developers onto z/OS platform

CICS Developer Center for Q&A

APAR PH18618 for Node.js 12 support



Why use Node.js in CICS ?

- A Node.js application in CICS might aggregate calls to existing business logic functions
  - To provide a single service interface for a front-end application
  - Avoiding the need for the front-end application to make several network calls
  - A Node.js application can also add functionality to existing business logic
    - by calling external services or by using NPM modules

- Node.js applications can call services hosted in CICS in order to invoke existing business logic
  - These could be JSON or SOAP web services
  - Exposed by using CICS web services technology or z/OS Connect
- Node.js applications can call CICS services by using NPM modules
  - Used for making HTTP requests and for consuming JSON and SOAP web services

- When a Node.js application is hosted in the same CICS region as a JSON web service
  - A locally optimized transport can be used
  - Uses a cross-memory approach to call the service
  - Must use the ibm-cics-api module
  - Service must be exposed using CICS JSON web services technology
    - suitable PIPELINE and URIMAP resources must exist

The ibm-cics-api library

- The ibm-cics-api doesn't just work inside CICS
  - You can use it to call remotely into CICS too
- Develop the Node.js app on your own machine
  - Then test that same app without having to deploy it anywhere
- Enable in CICS by adding a TCPIPSERVICE in front of the URIMAP and PIPELINE that will be handling the request from Node.js
- The transport will now use HTTP rather than a cross-memory call

Node.js application in CICS

Add Node.js application to a CICS bundle

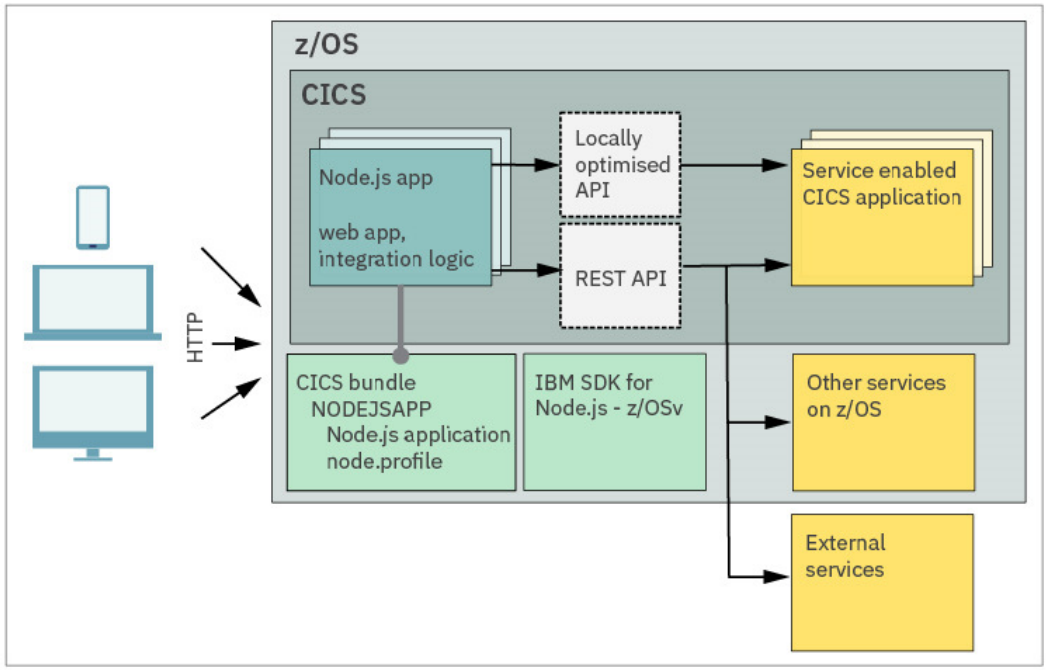
1. NODEJSAPP bundle part
2. Profile
3. Nodejs. application
   - Start script
   - Other application assets

Build CICS bundle from source and deploy to zFS

- CICS build toolkit
- CICS Explorer
- Zowe CLI
- Tag text files, otherwise EBCDIC assumed
- Run npm to resolve dependencies
- DFHDPLOY, CICS TS plug-in for UCD, ...

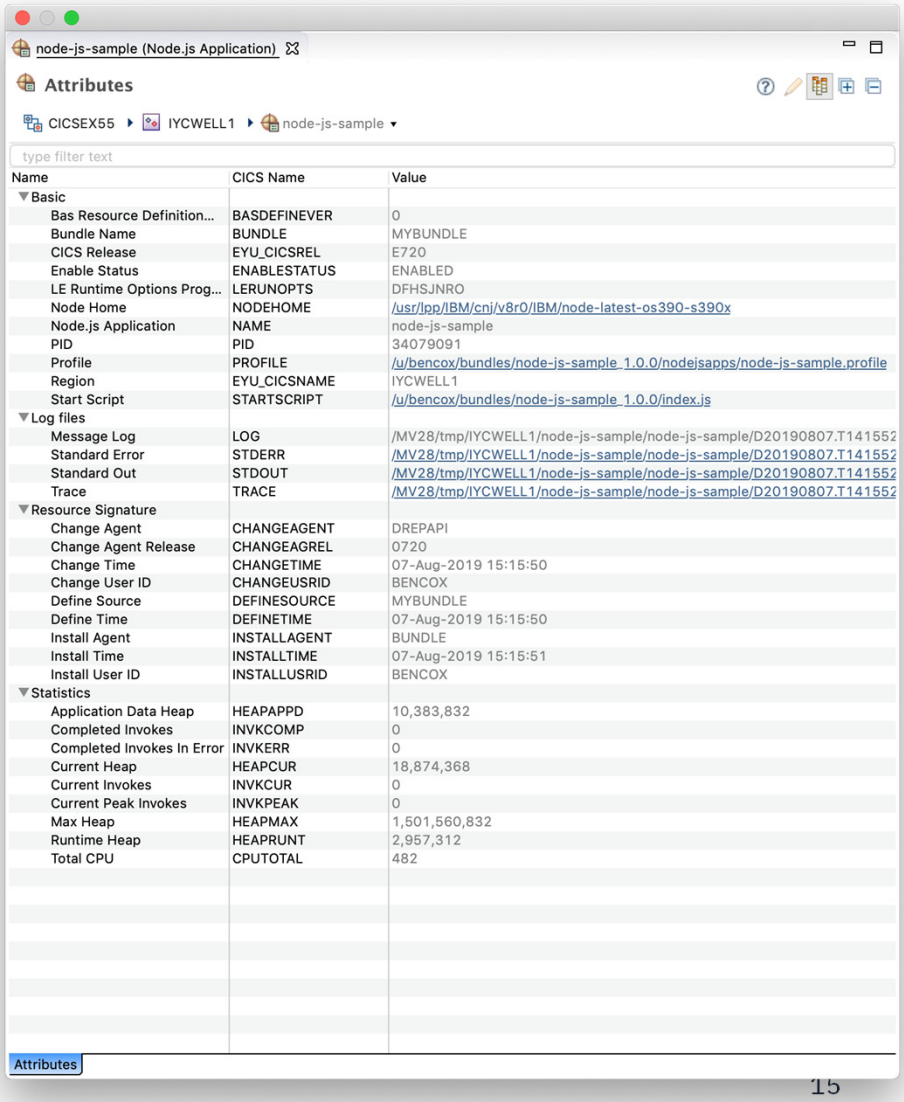
Lifecycle Management via CICS bundle as usual

- CEDA, CEMT, SPI, CICS Explorer, CMCI, ...
- Node.js app is running when bundle part enabled
- IBM SDK used by CICS to run the application
- Unix signals used by CICS to end application



Using Node.js in CICS

- The NODEJSAPP resource gives information on
  - Configuration
  - Status
  - Logs
  - Statistics

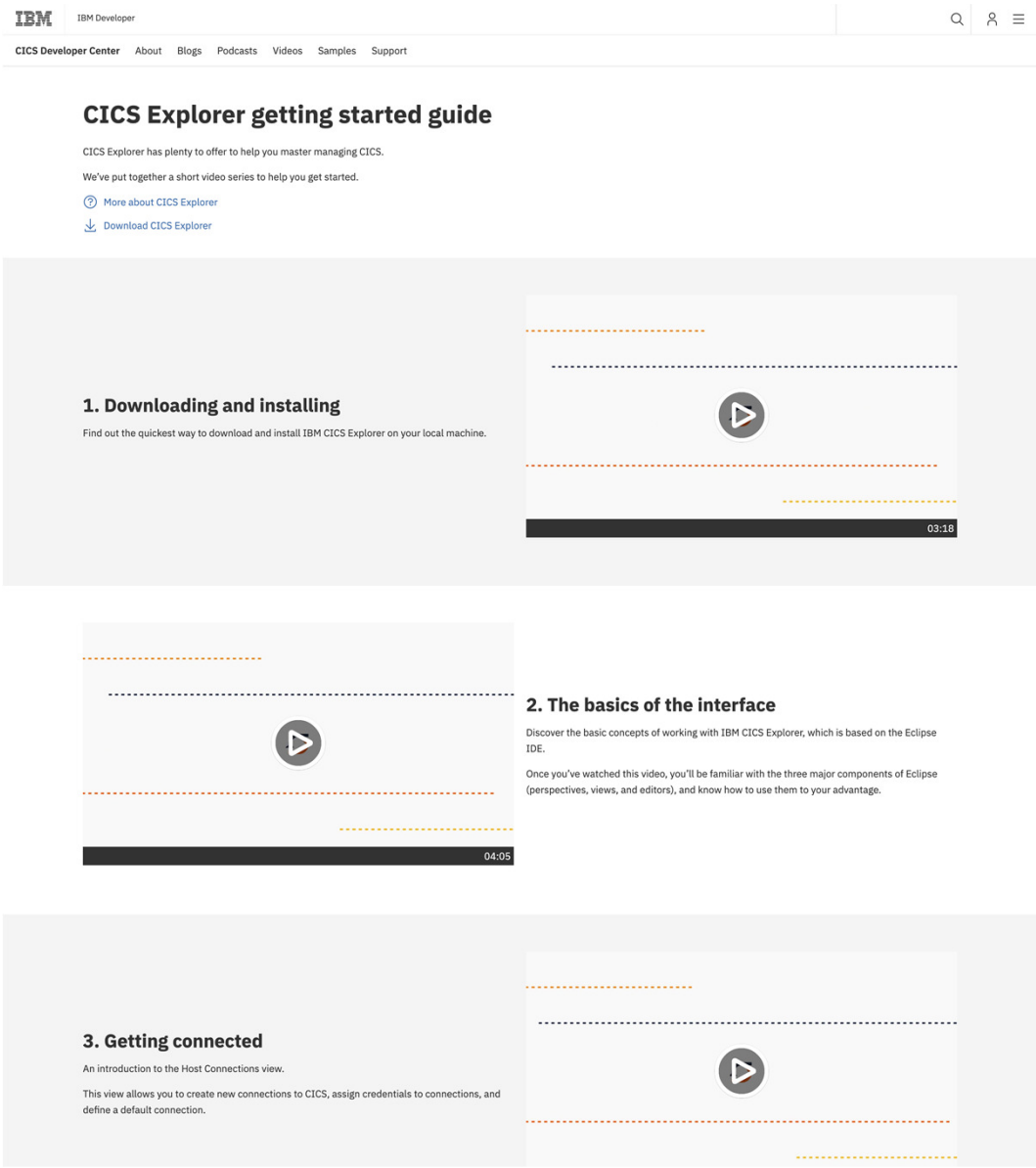


Developing a Node.js application for CICS

- Development of Node.js applications for CICS is, in general, like developing any other Node.js application
- Use common build and packaging techniques
  - look out for the rare package that uses natives and are not yet supported z/OS
- Node.js applications embedded in CICS can invoke programs in the rest of CICS
- This API is available in the NPM repository
  - npm install --save **ibm-cics-api**

Getting started with CICS Explorer

- We have a new 6-part video series to get you started with using CICS Explorer
- Just over half an hour takes you through
  - Downloading and installing
  - The basics of the interface
  - Getting connected
  - Using the main CICSplex Explorer view
  - Using resource views
  - Configuring resource views



Deploying to CICS

- Eventually, you will want to test the app in CICS
  - by creating a CICS bundle, exporting it to zFS, and installing it into CICS

- Can deploy using CICS Explorer
  - but Node.js developers typically do not already have CICS Explorer
  - expect to perform deployment via the command line

- Use the Zowe CLI with the zowe-cli-cics-deploy-plugin

Zowe CLI

- Zowe CLI offers a shell-based way of interacting with a z/OS machine, from your desktop
- Use interactively or embed in shell scripts
- Core capabilities around data sets, USS, jobs, TSO and console commands
- Extensible by plugins

- > zowe files dl ds "CICS.INFO.FILE"
- > zowe uss issue ssh "ls"
- > zowe jobs list jobs -o "*" -p "*"
- > zowe tso issue cmd "LU"
- > zowe console issue cmd "DISPLAY M"

Zowe CLI: CICS Deploy Plugin

- The zowe-cli-cics-deploy-plugin provides a way of deploying CICS bundles from the command line
  - Generate a CICS bundle from an existing workstation directory structure
  - Deploy a CICS bundle to a CPSM managed group of CICS regions
  - Undeploy a CICS bundle from a CPSM managed group of CICS regions

- Many more options than shown here
  - github.com/IBM/zowe-cli-cics-deploy-plugin
  - npm.im/zowe-cli-cics-deploy-plugin

- > zowe cics-deploy generate bundle

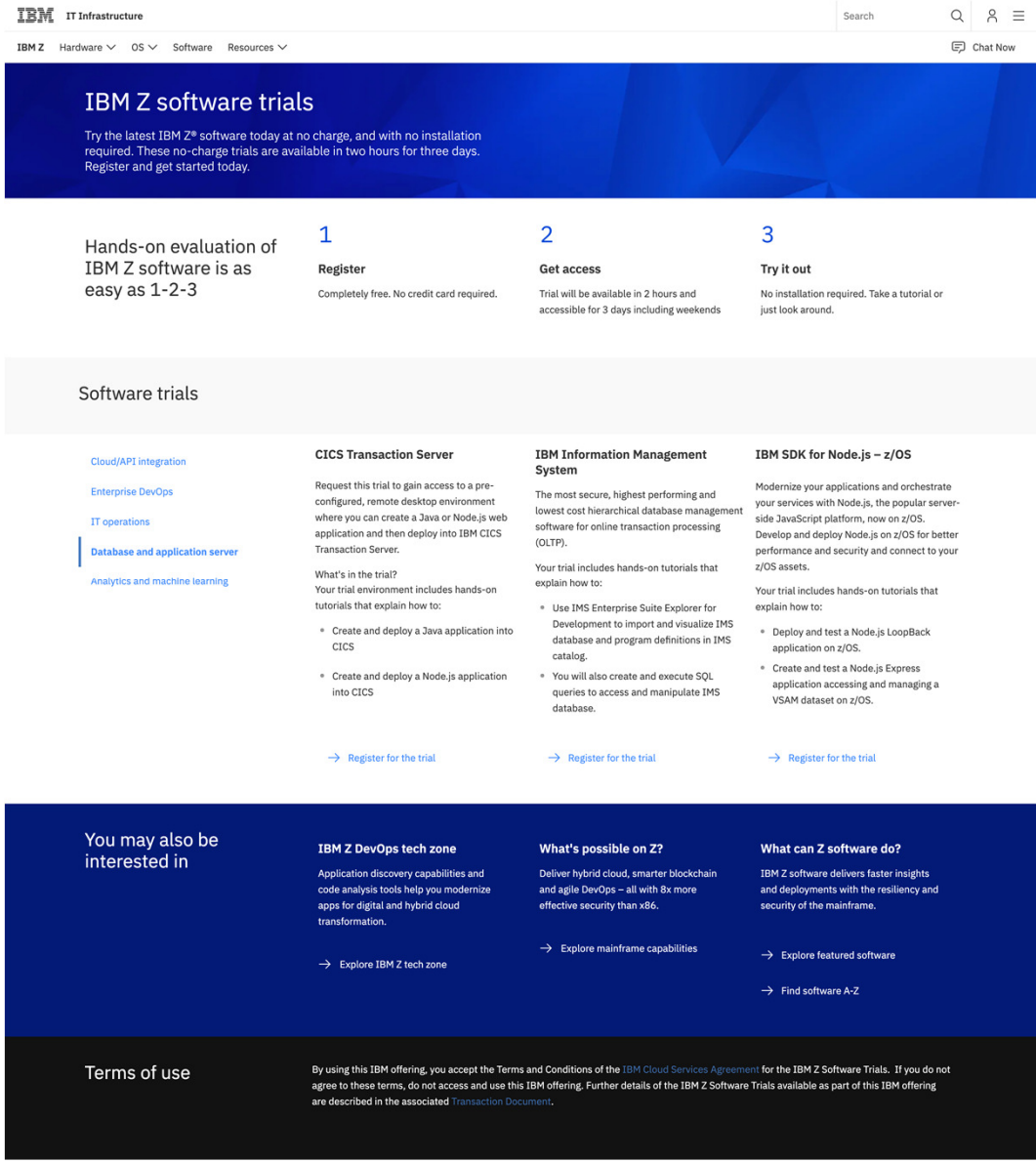
- > zowe cics-deploy push bundle
  --name MYBUNDLE
  --target-directory /u/me/bundles

Get started with Node.js on z/OS

- If you've not yet updated to CICS TS 5.5, or have other impediments to trying out Node.js on CICS, don't worry
  - Use IBM Z Trial at ibm.biz/ibmztrial

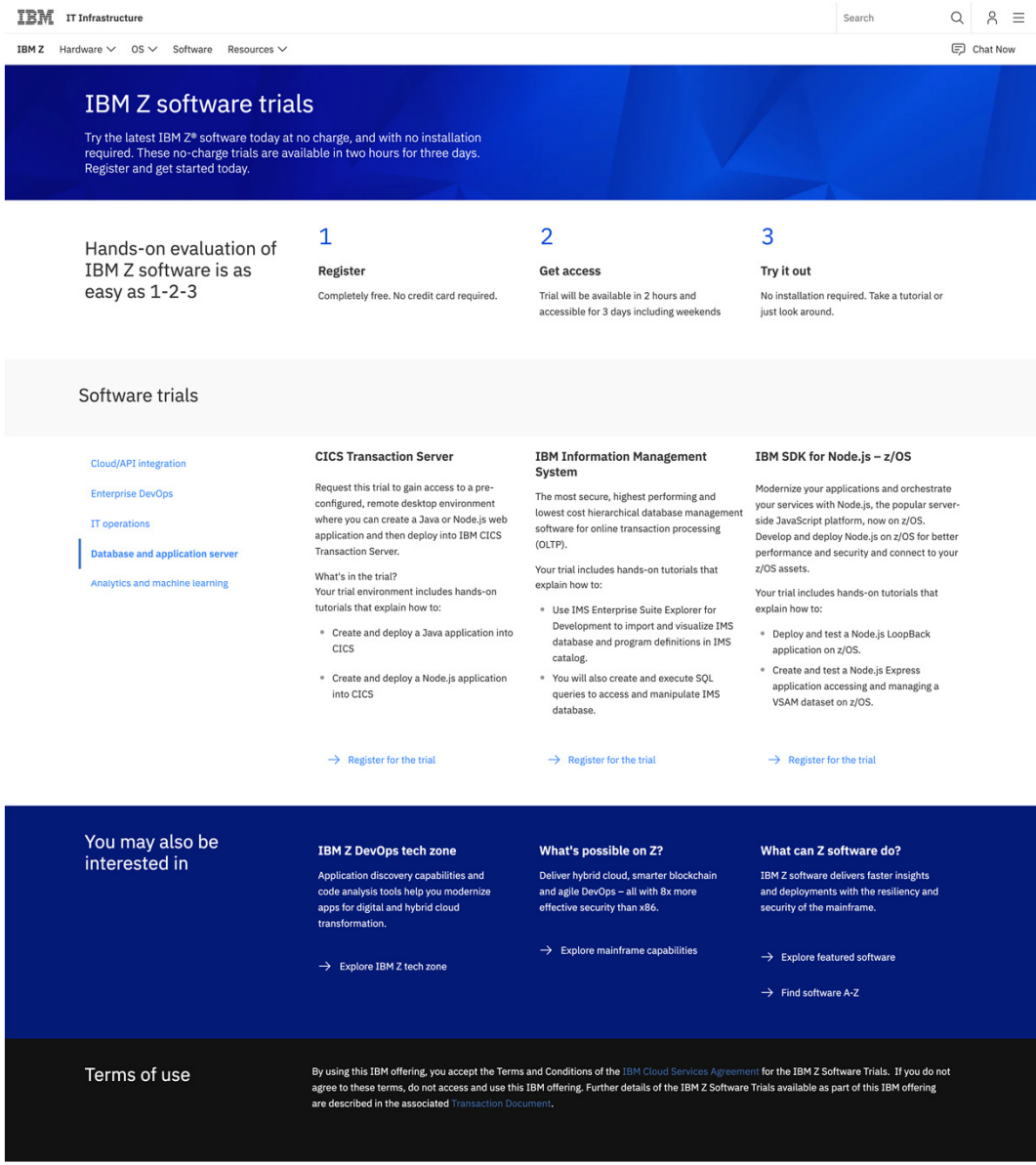
- CICS Transaction Server trial
  - Create and deploy a Node.js application into CICS

- IBM SDK for Node.js – z/OS trial
  - Deploy and test a Node.js LoopBack application on z/OS
  - Create and test a Node.js Express application accessing and managing a VSAM dataset on z/OS



Get started with Node.js on z/OS

- IBM Z Trial offers:
  - Many different products to evaluate
  - Available within a few hours
  - Configured for productive use
  - Guided scenarios
  - Absolutely free

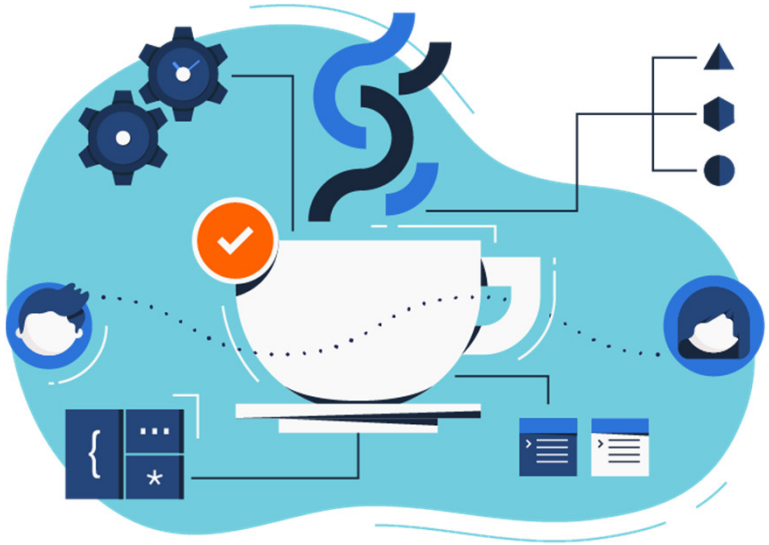


IBM **CICS**
Transaction Server for z/OS

Java in CICS

Java EE 8 Full Platform application support

- CICS TS supports Java applications that are written to the Java Enterprise Edition 8 (Java EE 8) Full Platform specification
  - Using the embedded version of IBM WebSphere Liberty
  - APAR PH15017 CICS TS V5.5
- Java applications that are hosted in CICS TS are integrated with CICS tasks by default
- A simple and powerful mechanism of modernizing CICS applications by using Java EE 8 features and capabilities



Support for Jakarta EE 8 Platform

- The CICS Liberty JVM server supports now supports the Jakarta Enterprise Edition (EE) 8

- The Jakarta EE 8 full platform technologies and specifications are an evolution of Java EE 8
  - Allows developers and applications to easily transition from Java EE to Jakarta EE
  - Available with APAR PH19704

- The promise of Jakarta EE is a community-driven open source model
  - More frequent releases than Java EE
  - Evolving more quickly to address the needs of modern applications

Java support in CICS

- CICS provides the tools and runtime to develop and run Java applications in a CICS JVM
  - Java applications can interact with CICS services and applications written in other languages
  - You can develop applications using the IBM CICS SDK for Java, Maven modules, or Gradle modules

- The CICS JVM server
  - Eligible Java workloads can run on specialty engine processors
    - reducing the cost of transactions
  - Different types of work such as threadsafe Java programs and web services
  - Application life cycle can be managed in the OSGi framework
    - no need to cycle the JVM server
  - Java applications that are packaged using OSGi can be ported more easily between CICS and other platforms
  - Java EE applications can be deployed into the Liberty JVM server

IBM CICS SDK for Java – The CICS Explorer

- The IBM CICS SDK for Java is included with the CICS Explorer
  - Provides support for developing and deploying applications that comply with the OSGi Service Platform specification

- The IBM CICS SDK for Java EE, Jakarta EE and Liberty is included as an option with the CICS Explorer
  - Supports packaging of Liberty applications into CICS bundles that can be deployed to CICS

- The OSGi Service Platform provides a mechanism for developing applications using a component model
  - Deploy applications into a framework as OSGi bundles
    - an OSGi bundle is the unit of deployment for an application component
      - contains version control information, dependencies, application code

- The IBM CICS SDK for Java allows development of Java applications for any supported release of CICS
  - The SDK includes the Java CICS library (JCICS) to access CICS services
  - Examples to get started with developing applications for CICS

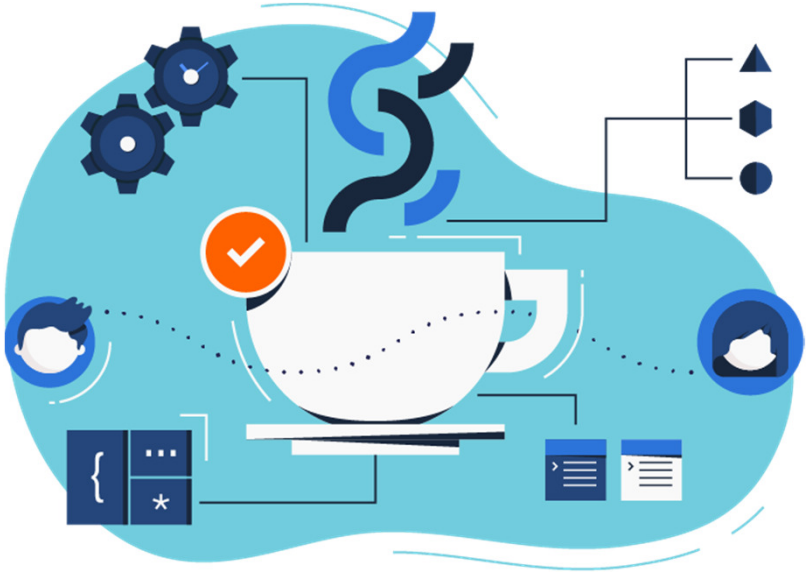
Using Maven or Gradle

- You can use popular build tools such as Maven and Gradle to create your own scripts for building and deploying CICS Java programs
  - An alternative to the IBM CICS SDK for Java
  - Easy management of dependencies
    - Java developers can add the required versions of the Java CICS APIs and the CICS annotation processor to the Java dependencies
  - More freedom when choosing the development environment
    - Maven and Gradle support most Java IDEs
      - such as Eclipse, IntelliJ IDEA, and Visual Studio Code
  - Better integration into a build toolchain
    - Maven and Gradle integrate smoothly with other automation tools such as Jenkins and Travis CI

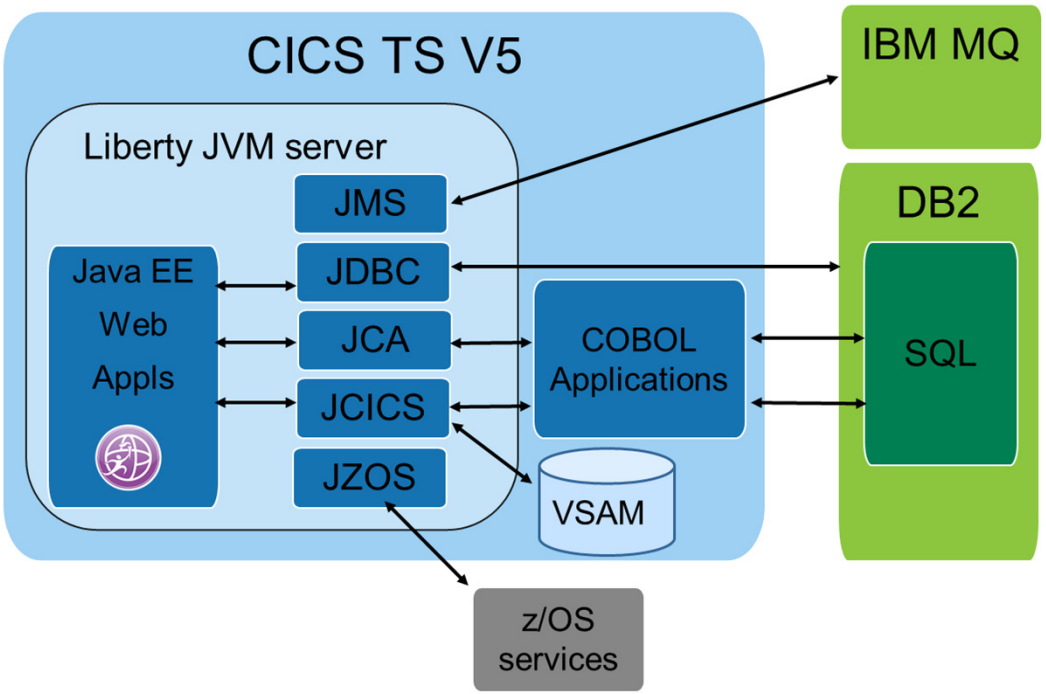
- The following artifacts are available on Maven Central
  - The Java CICS class library (JCICS)
    - Provides the EXEC CICS API support for Java applications in CICS TS
  - The CICS annotations library and the CICS annotation processor
    - Provides support that enables CICS programs to invoke Java applications in a Liberty JVM server
  - A bill of materials (BOM)
    - Defines the versions of the other artifacts to ensure that they are at the same CICS TS level

The types of JVM server in CICS

- Aside from a couple of more niche use cases
  - JVM servers form these categories
    - OSGi JVM server
    - Liberty JVM server

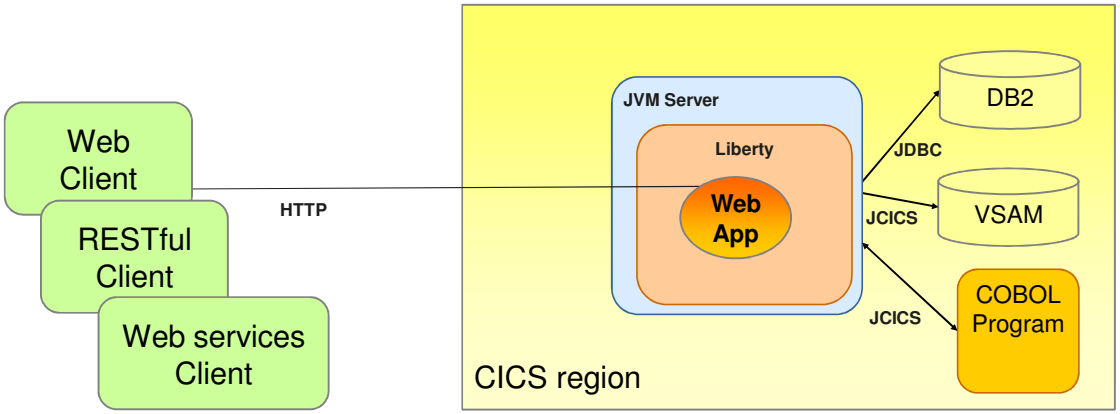


Java EE 8 Full Platform in CICS



- ✓ The full **Java EE 8 profile** supported in an integrated Liberty JVM server
- ✓ **JMS** support for MQ in client mode
- ✓ **JDBC** and **SQLJ** for Db2 data sources and other relational databases
- ✓ **JCICS** to provide access to CICS API including linking to other CICS programs
- ✓ **JCA** local ECI adapter supports porting of CICS TG ECI applications into CICS
- ✓ **JZOS** provides access to z/OS services such as console, files

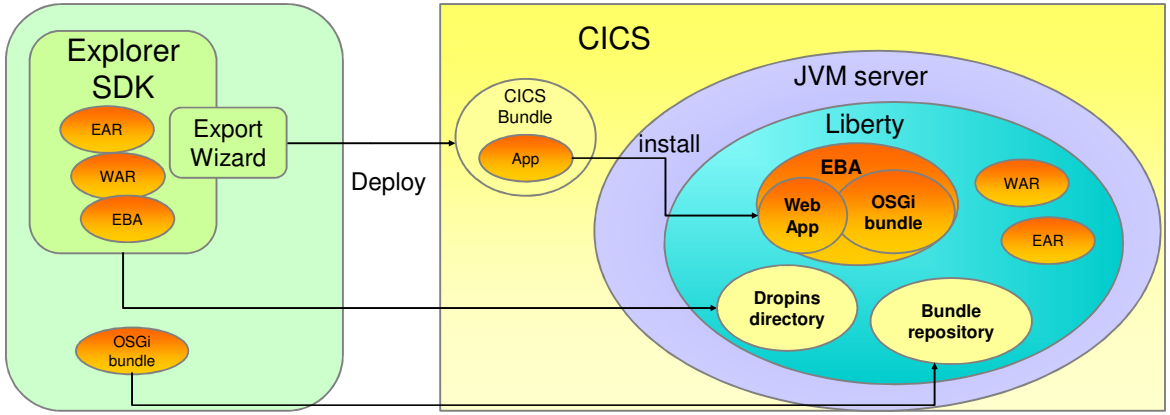
Why use Liberty in CICS?



- **Porting web** application to z/OS
  - JEE8 Web profile and JCA local ECI - 'lift and shift' porting from other JEE servers
- New **integration logic** for existing CICS services
  - Restful services or SOAP Web services interfacing existing CICS components
- **Java business logic** in CICS
  - Access to DB2 data – JDBC, EJBs, JPA or VSAM/JCICS

Application deployment – Liberty JVM server

- Liberty dropins directory
  - For development/testing
- CICS Bundle resource
  - Web application(WAR), JEE archive (EAR), OSGi Application(EBA)
  - CICS-managed application deployment
- Liberty application definition
  - For manual deployment via server.xml
- Liberty shared bundle repository or global library
  - For shared components



CICS JVM profiles

Include & share common configuration when cloning JVM servers across CICS regions

- For example unique ports, database configuration or log settings

```
%INCLUDE=<file_path>
```

Append to variables are built up over multiple lines

```
OSGI_BUNDLES=&CLONEDIR;/mybundle.jar
+OSGI_BUNDLES=/newpath/mybundle2.jar
```

... is equivalent to ...

```
OSGI_BUNDLES=&USSHOME;/&JVMSERVER;/bundles/mybundle.jar,/newpath/mybundle2.jar
```

Including server.xml snippets

- Inject Liberty configuration into server.xml

- A new JVM profile option LIBERTY_INCLUDE_XML is provided
  - to enable Liberty to load shared configuration
  - making it easier to administer, clone, and control Liberty JVM servers

- You can now use the LIBERTY_INCLUDE_XML property in JVM profiles
  - specify files that CICS will add <include> tags for

- In JVM profile
  `LIBERTY_INCLUDE_XML=<file>`

Management

CICS bundle status reflects Liberty application status

- CICS bundle with Web application bundle part remains in ENABLING state
  - until applications are installed in Liberty
- More robust application deployments

Extended CICS JVM server message

```
LOG_LEVEL=INFO | WARNING | ERROR | NONE
```

- New dfhjvmlog zFS file for CICS JVM server information, warnings and errors
- Can be redirected to MVS JES DD
- For example
  - a value of NONE suppresses all output
  - a value of WARNING gives log entries of warning level and above
  - the default value is INFO

More than one Liberty JVM server per CICS region

Multiple secure Liberty servers in a CICS region

- Provides improved application isolation or scalability without increasing number of regions
- Each Liberty server can have its own configuration and lifecycle – ideal for developers

Wait for Liberty angel process * (also in V5.4 APAR PI92676)

`-Dcom.ibm.ws.zos.core.angelRequired=true`

- More robust CICS start-up and IPL procedures
- Ensure that a Liberty JVM server will connect to a Liberty angel process
  - before reaching the ENABLED state
- Integrates with named Liberty angel process `-Dcom.ibm.ws.zos.core.angelName`

JSON Web Token

Liberty JWT feature

- Programmatically parse, build and verify JWT tokens in Java applications
- Provides for authentication using digitally signed web tokens
- Also available on CICS TS V5.3 and 5.4 with APAR PI91554

OpenID Connect Client feature

- Configure Liberty server to authenticate a request using a JWT token without writing any code
- Supports identity mapping
  - Map Subject in JWT to local registry user
  - Map distributed identity to SAF registry user via RACMAP

Liberty Admin Center

- The Liberty Admin Center is now supported in CICS

- It is a site built into Liberty that allows you to
  - View and configure server.xml and related files
  - Examine applications running in the server
  - View live statistics about heap, CPU, and threads

- Available on CICS TS 5.5 with PH08321

Java 8 Required for CICS TS V5

CICS TS V5.5 requires IBM 64-bit Java SDK for z/OS, V8

IBM 64-bit SDK for z/OS, Java Technology Edition V7.0.0 and V7.1.0

- To be withdrawn from service on September 30, 2019
- Withdrawal Announcement 916-121 – August 2, 2016

WebSphere Application Server Liberty

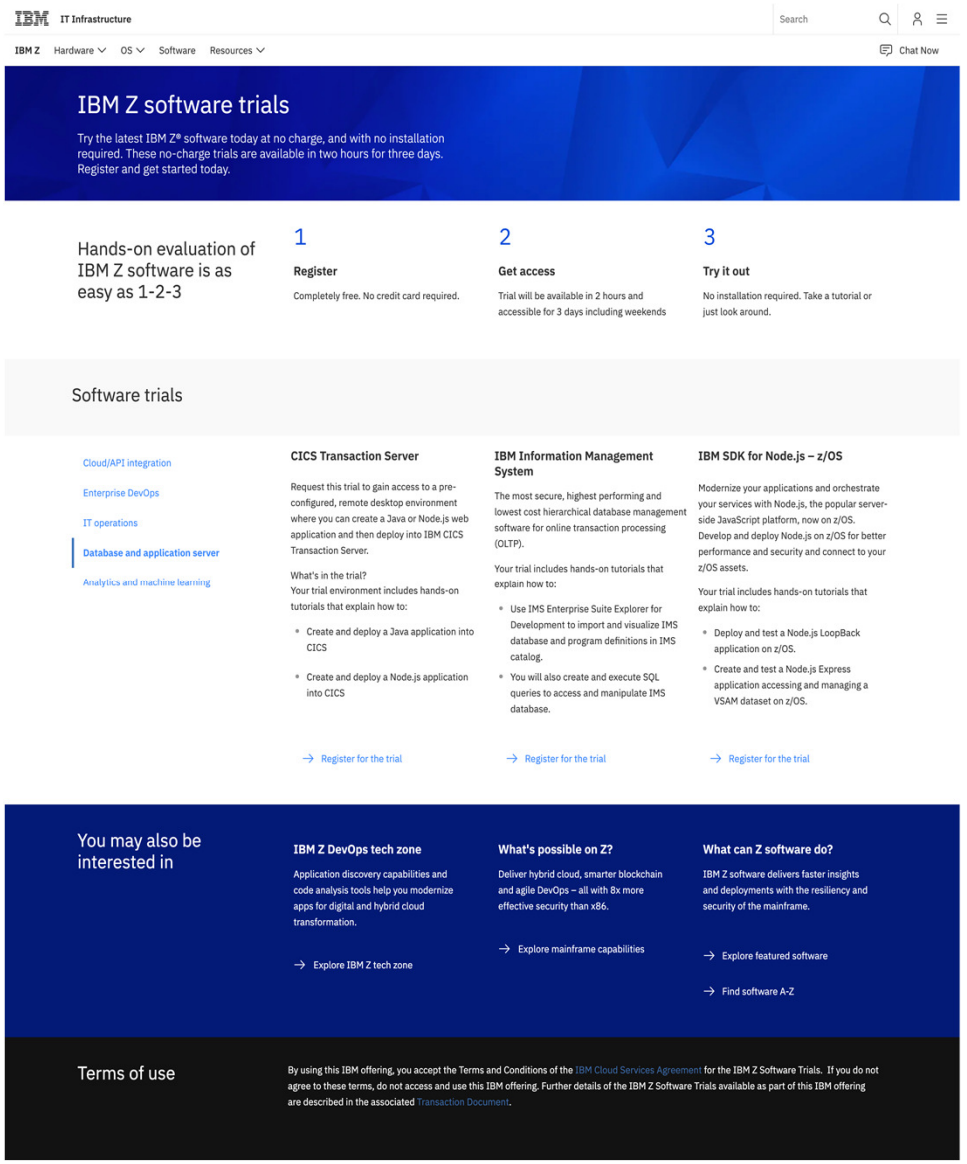
- WebSphere Application Server Liberty base – Removal Notices
- From September 2019, Liberty with fix pack 19.0.0.9 onwards will only run with Java 8 SDK

For details see article Java 8 recommended for CICS TS V5

Get started with Java in CICS

- If you have not yet configured CICS for Java, or have other impediments to trying out Java
  - use IBM Z Trial at ibm.biz/ibmztrial

- CICS Transaction Server trial
  - Create and deploy a Java application into CICS

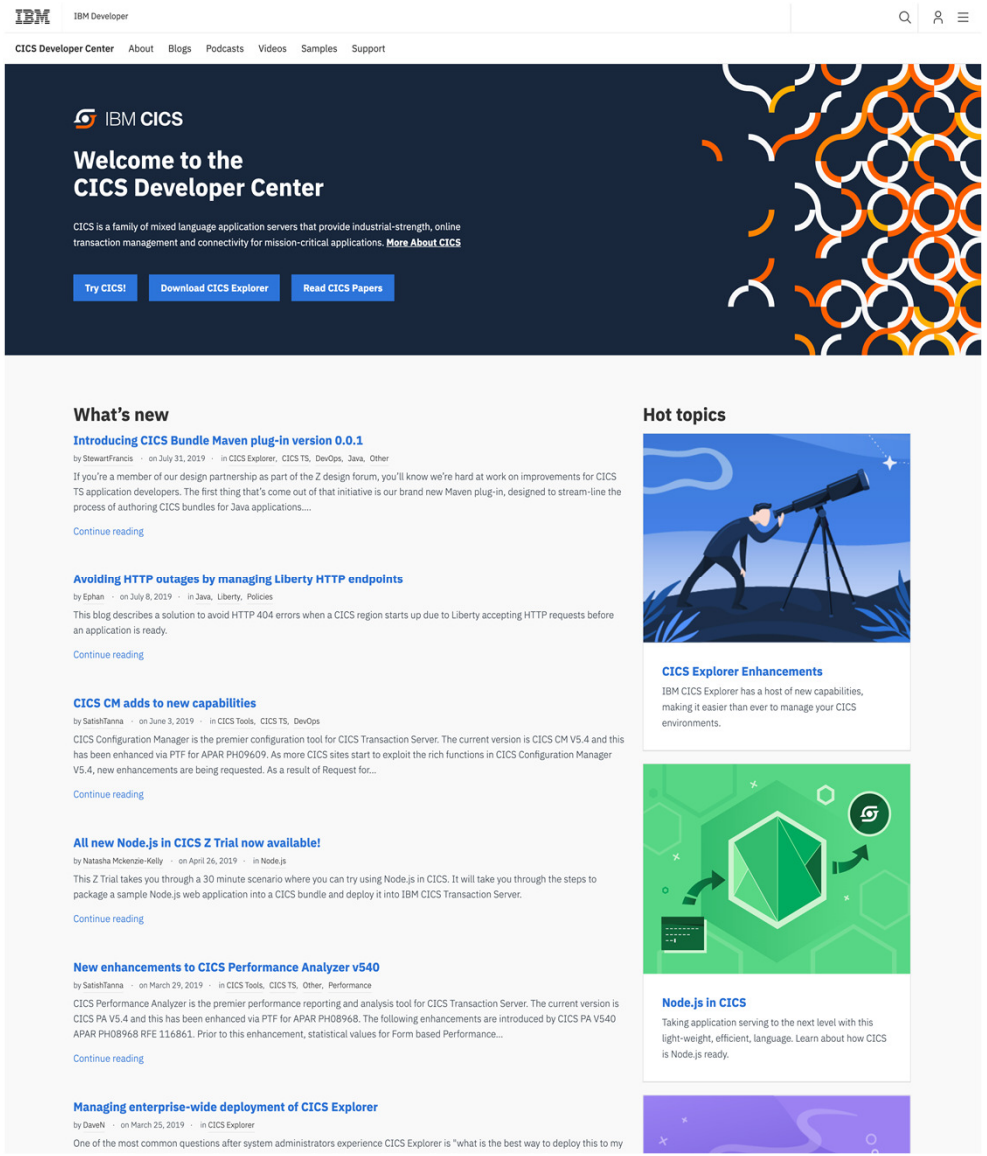


Get started with Java on CICS

- We have video series aimed at getting started with Java on CICS
  - Developing a RESTful Web application for Liberty in CICS
  - Architecting Java solutions for CICS
  - Extending a CICS web application using JCICS

Keeping up to date

- For all the latest developments subscribe to the blogs at developer.ibm.com/cics



CICS and Java Wildfire Workshop (CICSJAV1)

- This 2.5 day Wildfire workshop details CICS Java support
  - Including development, debugging, diagnostic, and tuning information
  - Upon completion of this workshop, you should have a better understanding of how to use Java in your CICS environment
  - This workshop will benefit Application Programmers, Systems Programmers, and Enterprise Architects

- Development of new Java applications is simplified
  - Due the mixed language environment of CICS and accessibility to your CICS resources
  - You can take a pure Java approach, or utilize CICS's many aids such as JSON parsing, Web Services support, or access to VSAM files and DB2

- Support for the Liberty Profile running in CICS
  - Offers the ideal integration of Java
  - Java specifications are supported
    - Java Servlets, JavaServer Pages, JAX-WS, and JAX-RS, while fully integrating into the CICS environment

- Lectures
  - CICS API for Java (the equivalent of the EXEC CICS API)
  - CICS and OSGi
  - CICS and the Liberty Profile
  - CICS and DB2 for Java
  - CICS/Java Application Tuning and Debugging

- Lab Exercises
  - The CICS Explorer
  - Writing a simple Java application in CICS
  - Using Servlets and JSPs in CICS
  - Writing a JAX-RS Java application for CICS
  - Writing a JAX-WS Java application for CICS
  - Adding a REST/JSON Interface to a Java program using z/OS Connect in CICS
  - Enabling CICS APIs using z/OS Connect EE