



Architecture-Driven Software Modernization

Documentation, Transformation & Refactoring

Mission

Through technical ingenuity, dedication, and collaboration, we automate the modernization of high-value software, advancing organizations into a better business and technology reality.

Vision

Creating a world where organizations aren't limited by technology.

About TSRI

Headquartered in Kirkland, WA

- Offices in Tucson, AZ and Washington, DC

Over 250+ Automated Modernization Projects since 2000

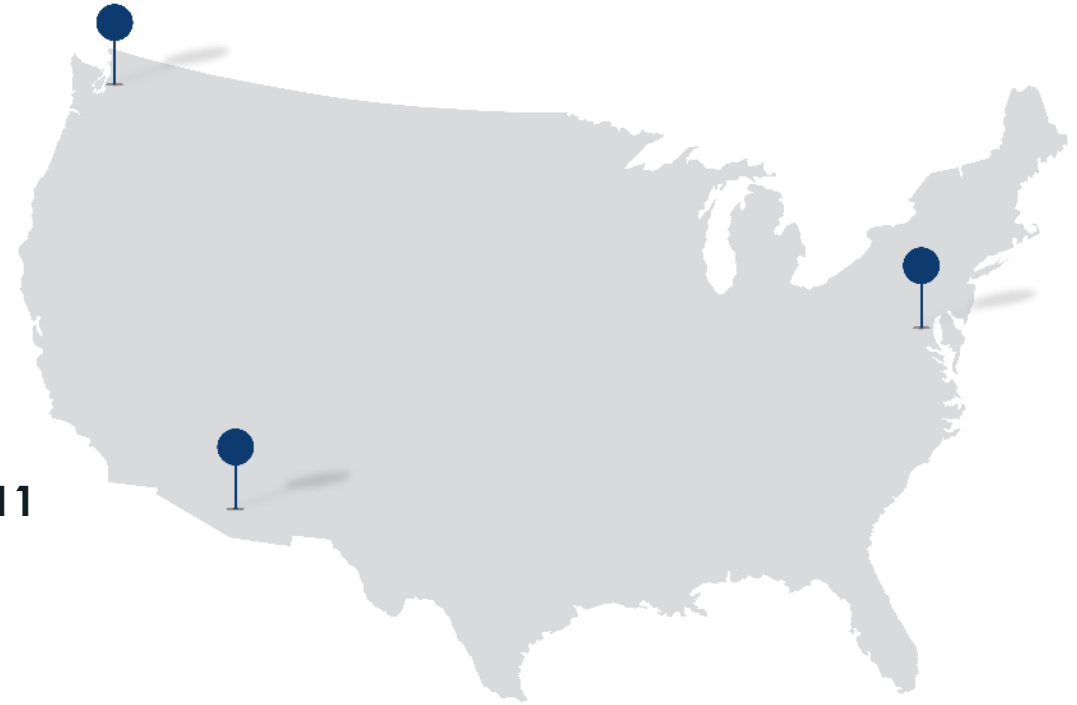
- Customer Satisfaction and Reference Accounts
- Award-Winning Technology Including Stevens Award in 2011

Technology Rooted in Early Artificial Intelligence Projects

- 1988 - 1994 Boeing Artificial Intelligence Lab
- 1983 USAF Knowledge Based Software Assistance (KBSA) Program

Member of the Object Management Group (OMG) Architecture-Driven Modernization (ADM) Task Force

- Author of Generic Abstract Syntax Tree Meta-Modeling (GASTM) Standard (adopted 2009),
- Author of Structured Patterns Meta-Model Standard (adopted 2015)
- Philip Newcomb:
 - Chair of Architecture Driven Modernization Task Force (ADMTF)
 - Author of *Information Systems Modernization: Architecture Driven Modernization Case Studies*, 2010



Customers



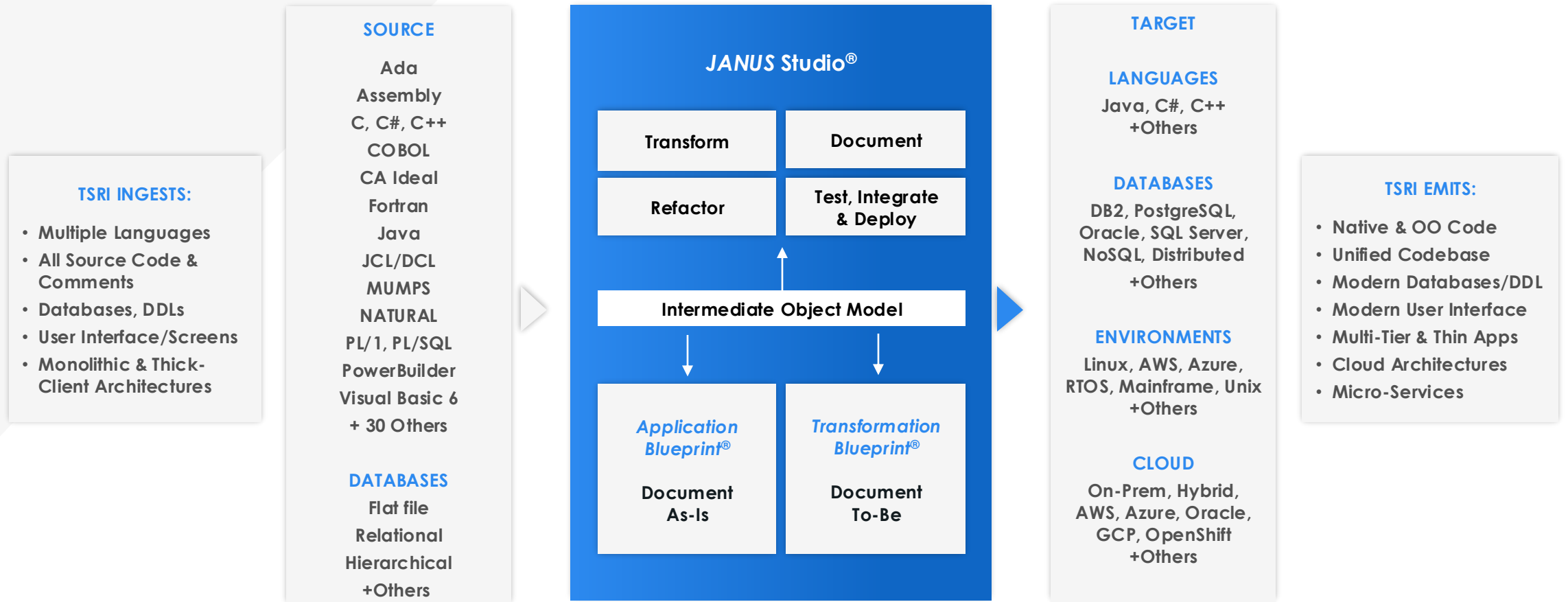
Featured Projects



Software Application	Source	Target	LOC	Total Time
TOPSKY – EUROCAT Air Traffic Management Systems	Ada	Java & C++	638k	9 Months
Amdocs - Sprint Billing System	Micro Focus COBOL	C	5.1M	7 Months
United States Air Force AFLCMC - SBSS ILS-S Logistics System	Unisys COBOL	Java	1.3M	10 Months
House & Urban Development (HUD) – CHUMS, F42d, CAIVRS, LOCCS	Unisys COBOL	Java	1.5M	10 Months
Boeing - Internal Billing Systems – PCOS, IBAS, FSIT	IBM COBOL & JCL	C# & Python	3.8M	11 Months
International Clothing Retailer – Documentation & Modernization	IBM COBOL & JCL	C# & Python	1.2M	6 Months
HCSC BlueCross BlueShield - Healthcare Provider System	PowerBuilder, MagnaX	Java	1.2M	3 Months
Naval Undersea Warfare Center - Weapons Control System (WCS)	Ada	C++	800k	7 Months
Core Automated Maintenance System (CAMS)	IBM COBOL	Document & BRE	1.0M	6 Months
Korean Air Force - F-16 Heads Up Display & Avionics	Jovial	C++	500k	6 Months
Boeing - Wiring System (WIRS)	IMS COBOL & JCL	C++ & Python	1.3M	12 Months
US Navy R-Supply Module, NTCSS System	PowerBuilder	Java	700k	12 Months
Veteran's Health Administration Fileman, WorldVistA & OpenVistA	MUMPS	Java	2.5M	6 Months
US Air Force Joint Mission Planning System (JMPS)	VB6	C#	776k	9 Months
Ballistic Missile Early Warning System – COBRA DANE	Ada & Fortran	C++	380k	8 Months
Oregon Public Employee Retirement System (OPERS)	IBM COBOL & JCL	C# & Python	250k	4 Months
Advanced Field Artillery Tactical Data System (AFATDS)	Ada	Java	5.5M	9 Months
Danish Government Digital Infrastructure Modernization	PL/1 & JCL	C# & Python	750k	12 Months*
Hundreds of Other Successful Projects Completed	35+ Source Languages	11+ Targets	1+ Billion	<i>Fast!</i>

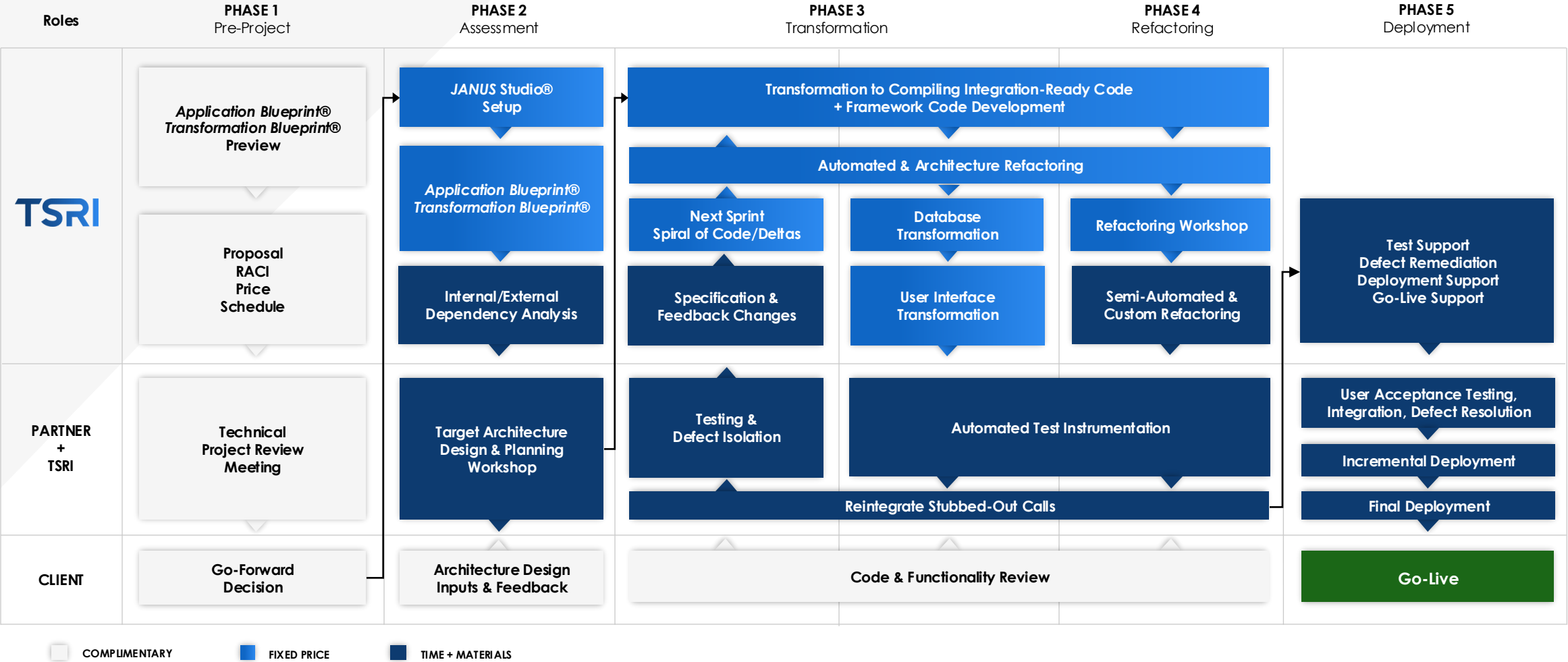
Modernization Method – Formal Methods AI

Assessment, Documentation, Transformation & Refactoring are done at 99.9X% automation with the lowest risk & minimal business disruption.



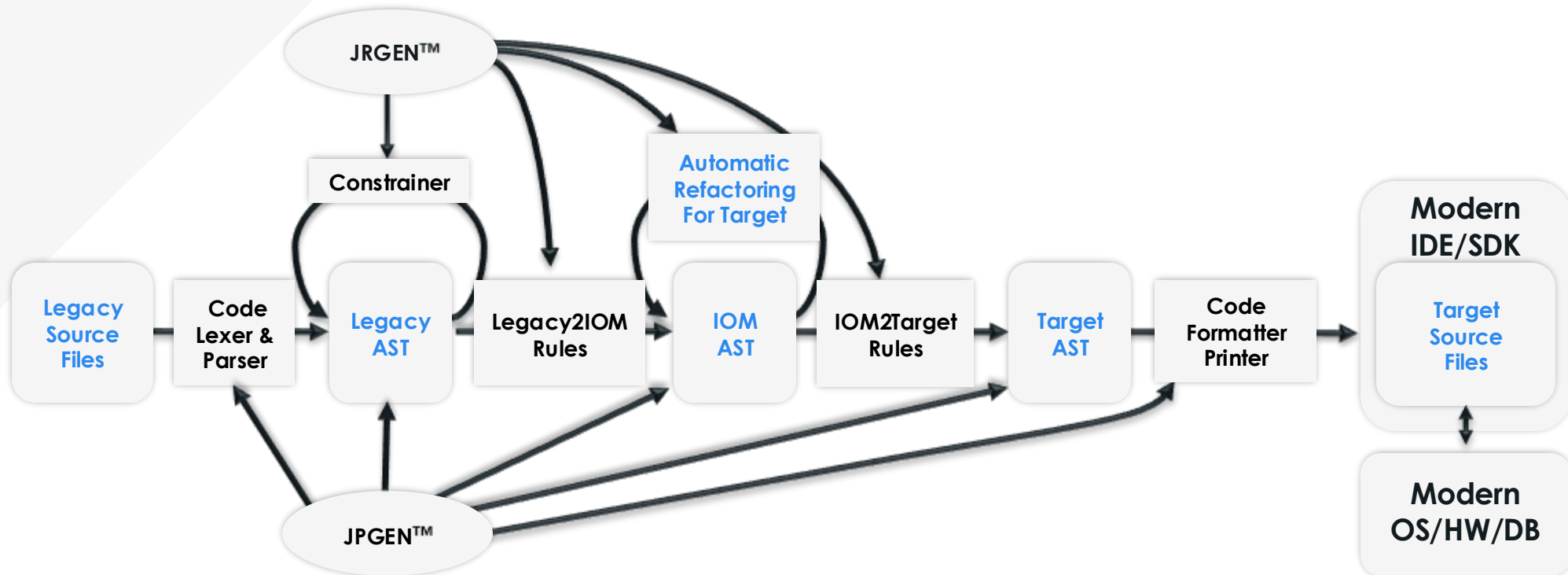
Our solution is flexible & feedback-driven to achieve the best possible output. All change is iterative & accomplished by transformation & refactoring rules applied to models.

Modernization Phases



TSRI Technology and Process

JRGEN™ is TSRI's Rule-Based Transformation Engine
Matches patterns & transforms source code AST models to target code models



JPGEN™ is TSRI's Grammar Specification Engine
Parses code, generates models (ASTs), prints & formats code from ASTs

Modernization Automated. Business Goals Accelerated.

Using TSRI's proven, automated, and flexible modernization solutions, our customers:

- Increase Business Agility
- Reduce TCO and O&M
- Rapidly Realize Business Goals



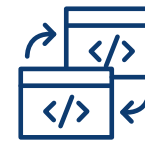
- ▶ Assessment, Discovery, & Documentation
- ▶ Gain insights into code and reduce O&M costs



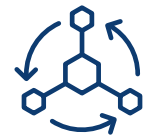
- ▶ Code, UI, & Database Migration to the Cloud
- ▶ Increase resilience, agility, and scalability



- ▶ 99.9X% Automated Modernization
- ▶ Reduce risk and rapidly realize business goals



- ▶ Automated Code Refactoring
- ▶ Improve code quality, maintainability, readability performance, and security



- ▶ Iterative, Architecture, & Model-Driven Approach
- ▶ Deploy containerized cloud-native applications

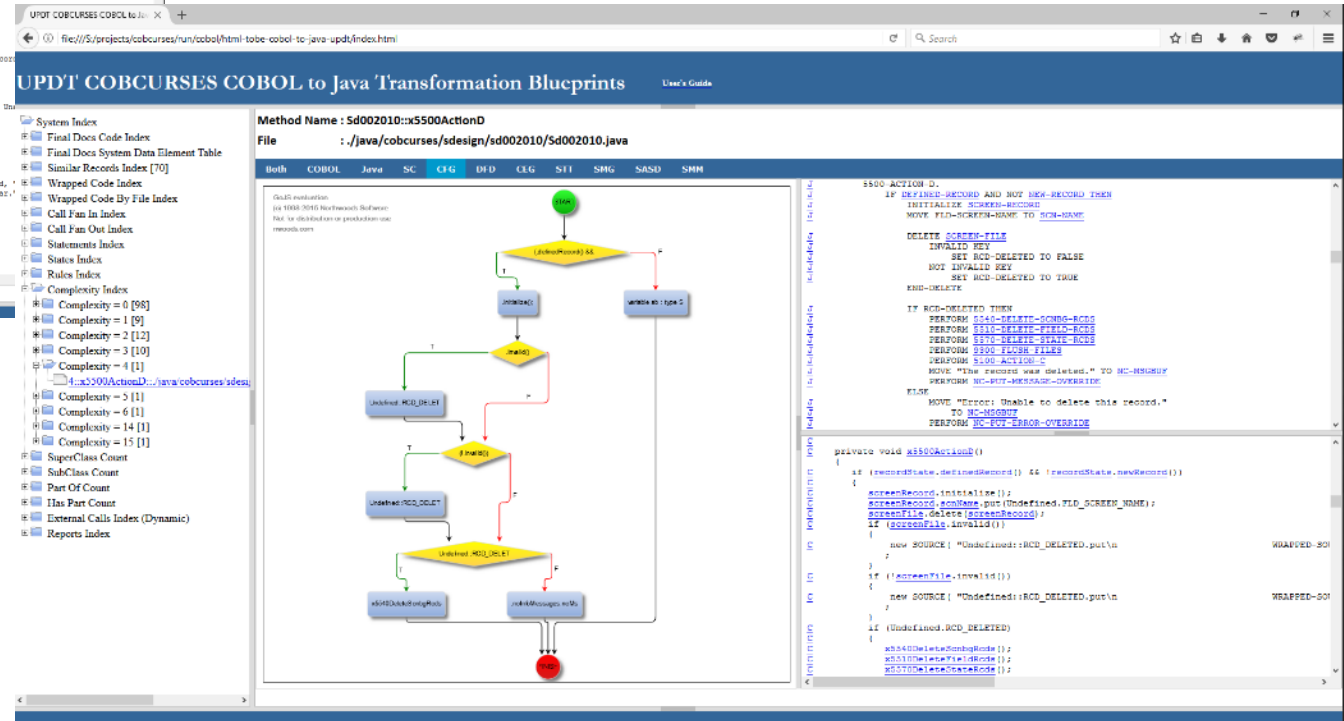
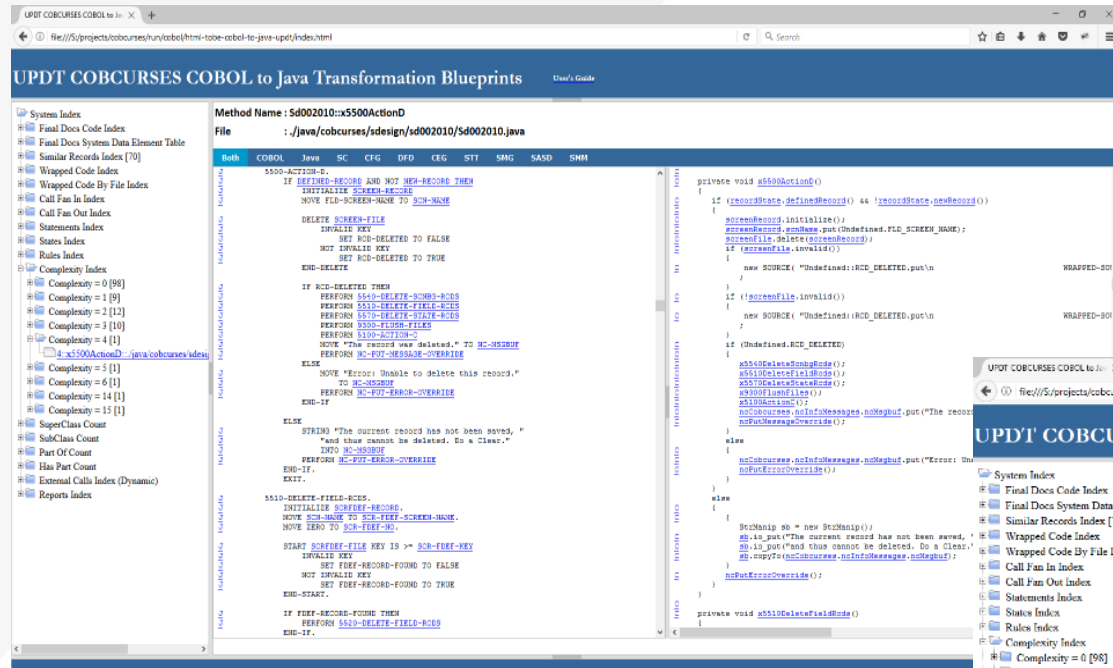


- ▶ Accelerated Modernization Journey
- ▶ Quickly realize ROI, with minimal business disruption

TSRI Automated Code Level Documentation

- HTML Based Documentation Included with Project
- *Application Blueprint®* - “As-Is” Documentation of Source Code with Charts and Graphs
- *Transformation Blueprint®* - “To-Be” Side-by-Side, Hyperlinking Source Code & Target Code

- **Structure & Data Flow**
- **Control Flow & Cause Effect Models**
- **Complexity Analysis**
- **External Interfaces Called**
- **Dead & Redundant Code Analysis**
- **Similar & Identical Code Analysis**



Air Force Lifecycle Management Center Legacy Screens

Active Civil Engineering Work Order Directory						
Work Order	Title	Cust Acct	Facility	WO Ind	Tracking Indicator	Tracking Status
TYMXA						
TYMXA 10085	CONSTRUCT JOGGING	900	TYMX 07002	C	CEP	CONTRACT
TYMXA 10086	PAINT 1ST FLR OFFI	330	TYMX 00028	Y	SHC	CONT REQ
TYMXA 10095	NEW FAC FOR 902FSS	900	TYMX 00000	C	CEP	CONTRACT
TYMXA 10097	REPAINT EXTR OF KE	220	TYMX 01039	C	CEP	CONTRACT
TYMXA 10099	MOUNT MONITOR TO W	606	TYMX 00977	Y	SHC	CLOSED
TYMXA 10121	INSTL MONITORS/SHE	606	TYMX 00499	N	SHC	CLOSED
TYMXA 10122	STRUCTURAL ASSESSM	220	TYMX 00156	C	CEP	CONTRACT
TYMXA 10127	MOUNT PHONE FRNT E	606	TYMX 00667	N	SHC	CONT REQ
TYMXA 10129	INSTL BRIDGE CRANE	723	TYMX 00076	N	CEP	REVIEW
TYMXA 10130	REPLACE WATER SOFT	220	TYMX 00999	C	CSU	CLOSED
TYMXA 10131	RMV 24 CASS STATIO	900	TYMX 11101	C	CSU	CLOSED
TYMXA 10134	INSTL 6 SEPARATE S	900	TYMX 11301	C	CEP	CONTRACT
TYMXA 10142	INCREASE PAD SIZE	606	TYMX 00499	A	CEOHP	CLOSED
TYMXA 10143	INSTL BREAK AREA P	606	TYMX 00663	D	CEOHP	SCOPING
TYMXA 10144	INSTAL BREAK AREA	606	TYMX 00667	A	CSU	WORC

Position the cursor and press ENTER to display the entire record.

(1)Keys	(2)First	(4)Prev	(5)Next	(6)Down	(7)Up	(8)Find
	(11)Add		(13)Help	(14)Path	(15)Print	(16)Retrn
(17)Menu	(18)Default	(19)Limits	(27)LOGST	(30)Histry	(31)Rpts	(32)Exit

Active CE Work Order/Request

Page: 1

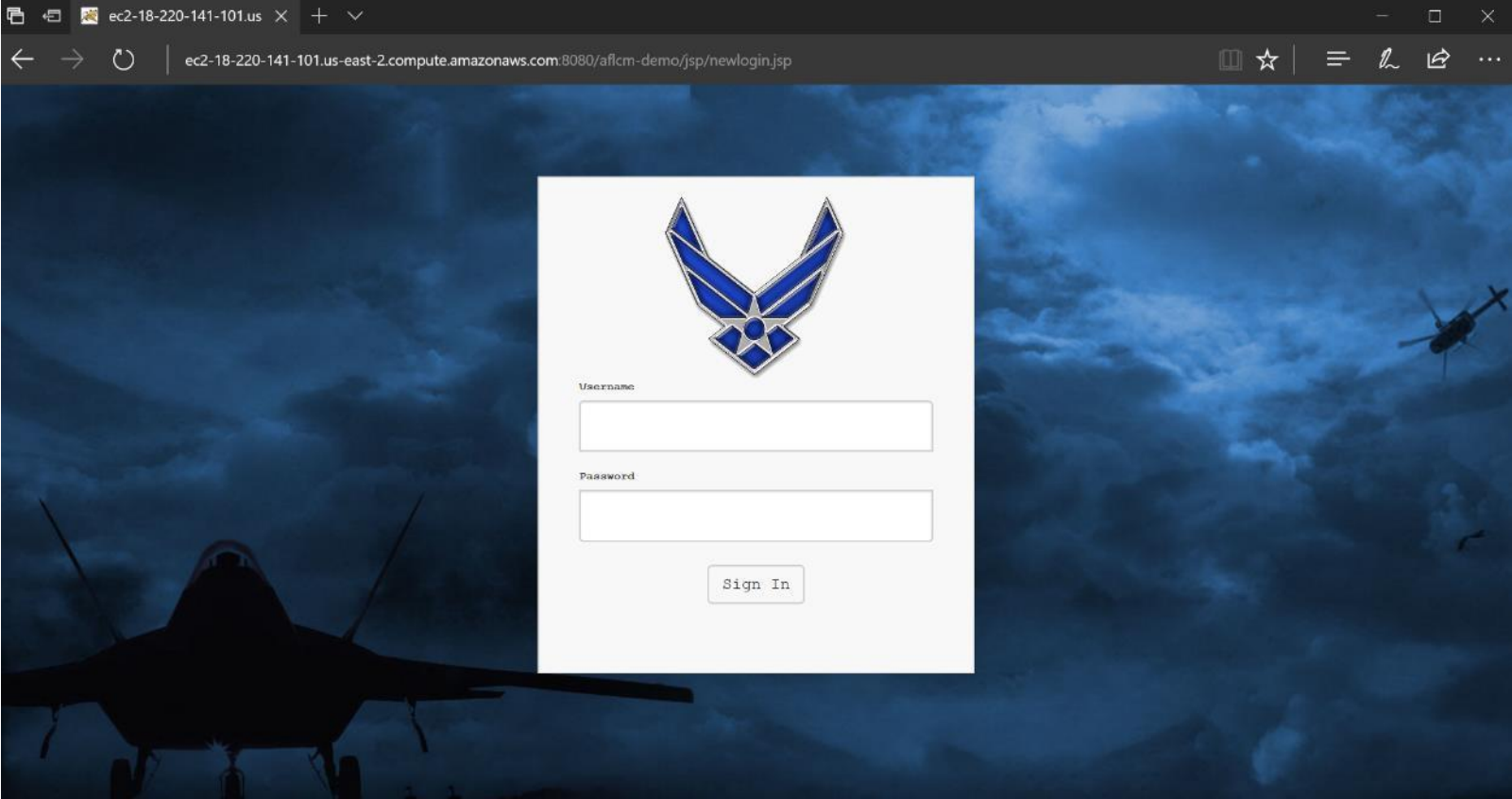
WO/Request #:	TYMX A 10142	INCREASE PAD SIZE BREAK AREA	Rec Status: C
WO Indicator:	A IN-SERVICE WORK	Tracking Location:	CEOHP
Work Class:	R Repair	Tracking Status:	CLOSED
Labor Code-LUC:	18	Inst/Facility #:	TYMX 00499
AF Account Code:	51070	Requester's Name:	MR GABEL
RRI Code:	T8	Organization:	AFPC
What's Reimb? All: X	Mat:	Office Symbol:	CC
Lab:	Contr:	Telephone Number:	565-3945
Gen/Sen Officer:	(G/S)	Facility Manager:	GABEL, MICHAEL
Special Interest:		Fac Mgr Orgn:	CC
Customer Account:	606	Fac Mgr Phone:	5652787
Cust/CE Priority:	/	Work Location:	OUTSIDE FAC 499
Work/Org Priority:	4 /	Type of Service:	...
Project Number:			
Infrastruct. Code:	...		
JOCAS JON:			
Deficiency: Fire	OSHA	Other	Travel/Work Zone: 1 /
Shops Assigned:	3	RAC:	
CPP :		Capitalization:	N Vouch #:

(1)Keys	(2)First	(3)Desc	(4)Prev	(5)Next	(8)Find
(9)Modify				(13)UWCPUD	(15)Print
(17)Menu	(18)Remrks	(19)Logs	(20)PvScrn	(21)NxScrn	(22)Shops
					(32)Exit

Note: in update/modify mode, highlighted field has select/list of values

Mainframe to Cloud Demonstration

Migration	Legacy	Cloud	Version
OS	z/OS	RHEL	6.5
Hardware	IBM Mainframe	EC2	T2.medium or T2.large
Web Server	N/A	Java 8 Tomcat	1.8
		Apache	2.0
Database	IBM DB2	Amazon MySQL	5.7
DNS	N/A	DNS	
Code	COBOL	Java	8.0
GUI	3270 Greenscreen	TypeScript	2.0
		HTML5	5.0
		CSS3	3.0
		Bootstrap	3.0



Cloud Architecture Target Options:



Example UI Modernization

Legacy 3270


Active Civil Engineering Work Order Directory						
Work Order	Title	Cust Acct	Facility	WO Ind	Tracking Indicator	Tracking Status
TYMXA						
TYMXA 10085	CONSTRUCT JOGGING	900	TYMX 07002	C	CEP	CONTRACT
TYMXA 10086	PAINT 1ST FLR OFFI	330	TYMX 00028	Y	SHC	CONT REQ
TYMXA 10095	NEW FAC FOR 902FSS	900	TYMX 00000	C	CEP	CONTRACT
TYMXA 10097	REPAINT EXTR OF KE	220	TYMX 01039	C	CEP	CONTRACT
TYMXA 10099	MOUNT MONITOR TO W	606	TYMX 00977	Y	SHC	CLOSED
TYMXA 10121	INSTR MONITORS/SHE	606	TYMX 00499	N	SHC	CLOSED
TYMXA 10122	STRUCTURAL ASSESSM	220	TYMX 00156	C	CEP	CONTRACT
TYMXA 10127	MOUNT PHONE FRNT E	606	TYMX 00667	N	SHC	CONT REQ
TYMXA 10129	INSTR BRIDGE CRANE	723	TYMX 00076	N	CEP	REVIEW
TYMXA 10130	REPLACE WATER SOFT	220	TYMX 00999	C	CSU	CLOSED
TYMXA 10131	RMV 24 CASS STATIO	900	TYMX 11101	C	CSU	CLOSED
TYMXA 10134	INSTR 6 SEPARATE S	900	TYMX 11301	C	CEP	CONTRACT
TYMXA 10142	INCREASE PAD SIZE	606	TYMX 00499	A	CEOHP	CLOSED
TYMXA 10143	INSTR BREAK AREA P	606	TYMX 00663	D	CEOHP	SCOPING
TYMXA 10144	INSTAL BREAK AREA	606	TYMX 00667	A	CSU	WORC

Position the cursor and press ENTER to display the entire record.

(1)Keys	(2)First	(4)Prev	(5)Next	(6)Down	(7)Up	(8)Find
(11)Add	(13)Help	(14)Path	(15)Print	(16)Retrn		
(17)Menu	(18)Default	(19)Limits	(27)LOGST	(30)Histry	(31)Rpts	(32)Exit



React.js or Angular.js



AFLCMC APPLICATION DEMO

Active Civil Engineering Work Order Directory						
Work Order	Title	Cust Acct	Facility	WO Ind	Tracking Indicator	Tracking Status
AJXFA 00011	CEMAS STORE STOCK	022	AJXF	L	MAT	MATL REQ
AJXFA 00012	CEMAS RESIDUAL	022	AJXF	A	MAT	MATL REQ
AJXFA 00013	CEMAS SNOW REMOVAL	022	AJXF	L	MAT	MATL REQ
AJXFA 00014	CEMAS ASPHALT AND	022	AJXF	L	MAT	MATL REQ
AJXFA 00015	CEMAS POOL CHEMICA	022	AJXF	L	MAT	MATL REQ
AJXFA 00016	CEMAS HVAC CHEMICA	022	AJXF	L	MAT	MATL REQ
AJXFA 00017	CEMAS GRASS SEED	022	AJXF	L	MAT	MATL REQ
AJXFA 01011	RWP SWEEP AIRFIELD	022	AJXF 90002	O	PAVEMENTS	IN PROG
AJXFA 01015	OPS UTILITY	022	AJXF 70841	O	UTILITIES	IN PROG
AJXFA 01016	RWP-SWEEP BASE ROA	022	AJXF MULTI	O	PAVEMENTS	IN PROG
AJXFA 01018	OPS-SUMMER POOLS	029G	AJXF MULTI	O	UTILITIES	IN PROG
AJXFA 01019	RWP-ENTOM BASE ARE	022	AJXF 99003	O	ENTOMOLOGY	IN PROG
AJXFA 01020	RWP-ENTOM AAFES	100	AJXF MULTI	O	ENTOMOLOGY	IN PROG
AJXFA 01023	RWP-ENTOM PESTCONT	022	AJXF MULTI	O	ENTOMOLOGY	IN PROG
AJXFA 01026	OPS SNOW RMVL A AN	022	AJXF 99003	O	PAVEMENTS	IN PROG

Position the cursor and press ENTER to display the entire record.

(1)Keys	(11)Add	(5)Next	(7)Up	(8)Find
(13)Help	(14)Path	(15)Print	(16)Retrn	
(17)Menu	(18)Default	(19)Limits	(27)LOGST	(30)Histry
(31)Rpts	(32)Exit			

ENTER

(1)

(2)

[3]

(4)

(5)

(6)

(7)

(8)

(9)

[10]

[11]

[12]

[13]

[14]

[15]

[16]

[17]

[18]

[19]

[20]

[21]

[22]

[23]

[24]

[25]

[26]

[27]

[28]

[29]

[30]

[31]

[32]

Cursor :
4 11

TSRI Automated Refactoring

Starts the Reengineering of Target Components

- **Automatic Refactoring**
 - Removes dead and redundant code and data
- **Semi-automatic Refactoring**
 - Merges and consolidates duplicate code and data
 - Reorganizes and improves design of code and data
 - Removes “As-Is” flaws from “To-Be” software

Creates reusable components for:

- **Optimization, packaging, and redistribution**
- **Micro-services, Rest calls & reusable services**
- **Integration into/with modern**
 - Parallel
 - Multi-processor
 - Distributed Environments & Databases
 - N-Tier operational environments

Refactoring Benefits

- Improves software maintainability
- Reduces Maintenance & Operating Costs
- Enhances software performance
- Supports component-based reusability
- Supports consolidation of “Stove-Pipe” systems

Semi-Automated & Custom Refactoring Examples

CODE

- Dead & Unused Code Elimination
- Commenting & Marking Operations
- User Guided & Automated Refactoring to Modularize Design
- Rename All References To Identifiers Across Program Class, Method, Field, Variables & Import
- Automated Modularization Based On Entry points & Call-tree
- Any User-specified Package, Class, Method, Variable Or
- Reduce Number Of Formal Parameters:
- Package Multiple Formal Parameters Into A Class
- Define Default-value Instances For Classes & Merge Into Class Constructors
- Convert Global Variables Into Class Data Members Referenced As Member Data Or Pass As Function Arguments
- Create Accessor Methods For Data Members
- Replace Direct Data References With Data
- Replace User-defined Types With Native Built-in Types
- Minimize Class Member Visibility
- Make Method/Field Private Instead Of Public

FUNCTION

- Merge And Consolidate Redundant & Duplicate Code, Classes, Methods & Code Blocks
- Consolidate Similar Statements, Classes, Types, Methods & Data Members
- Merge Similar Code Blocks, Code Statement Slices Into Methods:
- Extract Dissimilarities (Literals & Variables) To Parameters Of Methods
- Create Methods From Heuristically-Specific Code Slices
- Generate New Classes As Directed By Heuristic
- Functions Or Re-factoring Plan
- Extract-related Statements Detected By
- Code Pattern Analysis
- Evolve Code, Function & Architecture via Iterative Re-factoring
- Carry Out 3rd Party Refactoring Plans & Specifications

ARCHITECTURE

- Generate New Class Hierarchy
- Re-componentize Classes & Packages To Improve Coupling & Cohesion, & Segregate Business From Technical Logic
- Extract Subclasses From Superclass
- Consolidate Superclass From Similar Subclasses
- Move Classes Between Packages to Create More Modular Code
- Move Members (Method Or Field) Between Classes to Create More Functionally Cohesive Code
- Re-factor Derived Component Architecture Layers To Segregate Client-side Web-browser & Ui Code from
- Server-side Data Manipulation & Access Code
- Generate Multi-tier Application Architecture
- Separate High-level Business Logic From Client-side Presentation & Low-Level Db Definition & Manipulation
- SOA and Web-enablement Refactoring
- Create Flexible & Extensible Components To Support Future Enhancements
- Consolidate Code into Reusable Component Oriented Refactoring.
- Introduce SOA And Web-enablement Refactoring.
- Re-architect And Redesign
- Create Microservices

Redesigning & Reengineering IT Architectures

N-TIER ARCHITECTURE

- Extract Horizontal Services
- Data Access Layers:
File Descriptors Converted Into Data Layer Classes
I/O Statements (SQL or I/O) Converted Into
Method Of Data Layer Classes
- User Interface Layers: Screens Converted Into
Display Layer Classes Screen Statements
Converted Into Methods Of Display Layer Classes

WEB & MICRO SERVICE

- SOAP, WSDL, UDDI, XML-RPC, UBR, XHTML, JavaScript
- Microsoft Azure/.NET
- LAMP, IBM Bluemix
- AWS
- WebSphere, Sun JES
- Construction of SOA Services
- Construction of SOA Interfaces

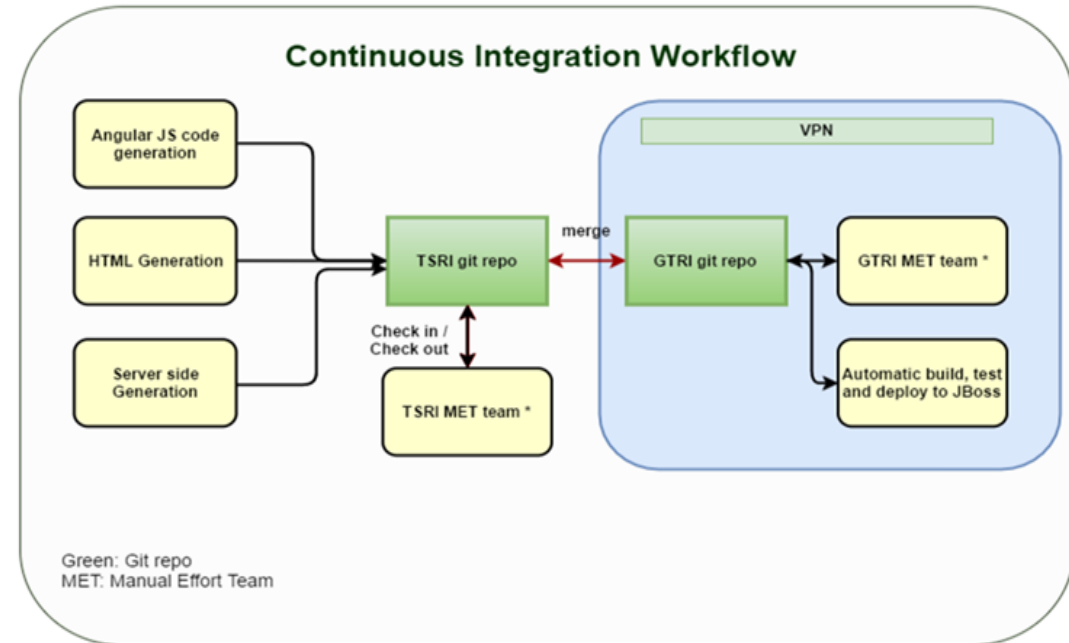
TARGET FRAMEWORK

- Component Oriented Refactoring
- Map to JEE Components
- (JDBC, JSP, JavaScript, HTML)
- Map to Microsoft .NET Components (ADO, ASP)
- Map to GCSS Component Framework
- Map to NNCI Component Framework
- Map to Angular, Spring Framework
- Map to Modular Open Architectures

TSRI Continuous Improvements

Continuous Integration and innovation in DevOps Transformation Innovation:

- Integration of multiple technologies and tools
- Support for automated scripting (Jenkins)
- Support for Automated testing
- Adoption of Best of Class tools and process
(BitBucket, Agile, Jira, Sprint/Kanban)
- DevOPS
 - CodePipeline, CodeBuild, Code Deploy
- Remote merging and integration – Container



Test Process:

Test Planning

- **Identify All Test Areas**
 - Review current AS-IS test assets
 - Categorize new vs transformed code, assess risk
 - Functional Equivalence Testing, As-Is comparative To-Be test
 - Framework and new content test cases
- **Most Tests Methods understood, automation everywhere possible**
 - Often limited by docs and key SME availability.
- **Tests Planned and Tasks defined, scheduled and aligned with project**

Test Execution

- **Baseline on legacy compare Functional Equivalence in target. TSRI Batch test orchestration framework**
- **UI Automated with Selenium**
- **API testing when applicable, Live Record and Playback in test**
- **JUnit, Python Unit Test framework, test automation development. TSRI JANUS meta data for test cases Code quality - SonarQube**
- **Track progress with test templates and in JIRA**

Experience & Partnership in System Test

- **Performance testing, Scalability/Load**
- **Third party Security Vulnerability**

Testing Considerations with Automated Modernization

- **Level of Test Required is Different from typical Enterprise App Dev**
 - Transformation code is a direct representation from a language neutral and independent object model sourced for legacy language.
 - Generated w/ automated process so eliminates random implementation errors
 - There are no changes in design or business logic
- **Changes can introduce issues**
 - Modifications to target source may breaks representation/linkage to the original legacy source.
- **Black box testing is more appropriate**
 - White box and case/procedural instrumentation/testing can inject differences
 - Leverage existing Selenium automated client software testing
 - Combination of several black box approaches most demanding/best coverage
- **Transformation is to be used for testability**
 - Automation of Equivalence partitioning, testing
 - COBOL telemetry instrumentation source and target analysis
 - Interface between COBOL and JCL for improved unit and system testability
 - API level testing record and playback
 - Instrumentation of SQL instruments integration/migration issues

Testing: Plan, Execute, Report

- **Test Planning**
 - Identify All Test Areas
 - Review current AS-IS test assets
 - Categorize new vs transformed code, assess risk
 - As-Is comparative To-Be test
 - Framework and new content test cases
 - Tests Methods understood, automation everywhere possible
 - Identify limitation of docs and key SME availability.
 - Tests Planned and Tasks defined, scheduled and aligned with project
- **Test Execution**
 - UI Automated with Selenium
 - API testing when applicable, Live Record and Playback in test
 - NUnit, Python unittest test framework development and experiences
 - Robotest or custom Python scripts
 - Comparative Testing
 - Track progress in JIRA
 - Third Party Collaboration
 - Performance testing, Scalability/Load
 - Third party Security Scans and Testing
- **Report, Monitor against Key Objects**

Testing: Automation

Automated test procedures write, execute review

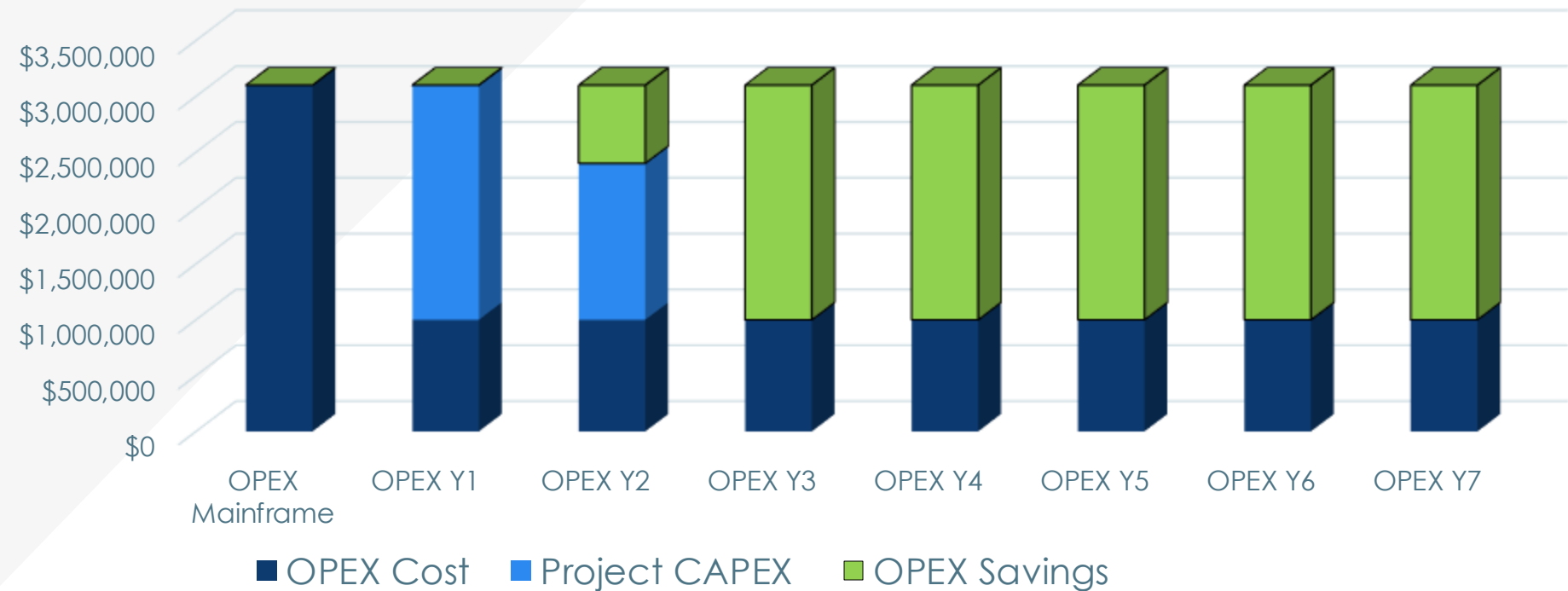
- Unit tests
- Selenium
- Postman test tool used for initial releases
- Python Test Scripts

Benefits

- Instantaneous test results for each software delivery
- No dedicated personnel required to perform testing manually
- Removes testing errors in performing manual tests
- Ensures consistent, repeatable tests steps for multiple environments and instantiations
- All code delivered to CM was automatically tested including full regression testing
- Each release verified through full automated test steps including full regression testing

Business Case - Actual Customer Example

Financial Analysis – Modernization of Medium-Sized Mainframe System



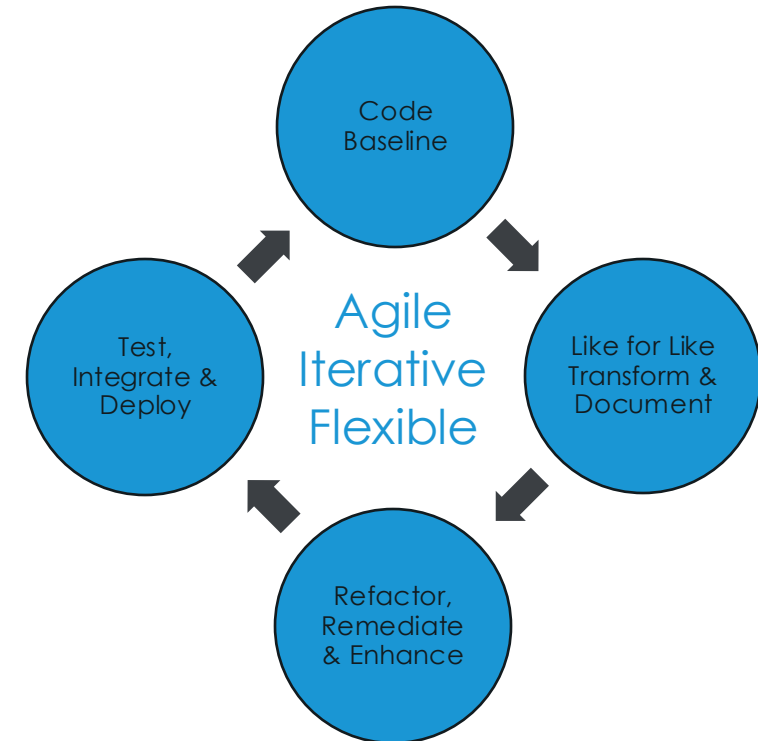
Project Summary: TSRI modernized a medium-sized mainframe system for a European government system used by over 11 million people. TSRI completed transformation to a modern cloud-enabled target, automated documentation, refactoring, and also completed automated functional, behavioral, and other testing.

Project Length: 1 Year
Total Project Cost: \$3.5M
Savings (7 Years): \$11.2M
ROI Achieved: 21 Months

Customer eliminated legacy technical debt, achieved OPEX savings of greater than \$2.1 million per year versus prior mainframe operations (ROI within 21 months), and moved system to a flexible, maintainable modern architecture.

TSRI Capabilities

- **Modernization of Legacy Systems to Customer Choice of Multi-Tier Architectures**
 - Targets Native, Object-Oriented, License Fee-Free Code
 - Targets All Major Modern Languages
- **Modernization of Flat File, Hierarchical, and Other Legacy Databases**
 - Targets Any Modern Database, DAO Layer
- **Modernization of Legacy User Interface (UI) to Web or Other Modern UI**
- **Documentation including “As-Is” and “To-Be” Documentation**
- **Refactoring to Improve Maintainability, Security, Performance, or Other**
- **Custom Pattern-Based Changes**
- **Migration to Modern Cloud Architectures**
 - Target AWS, Cloud Foundry, Azure, Bluemix, OpenStack, Milcloud or Others
 - Including SOA and Microservices Architectures, RESTful Interfaces, and Containers
- **Testing & Integration Support Solutions**
 - Test Telemetry Injection to Support Functional Testing



List of Source & Target Technologies for Automated Conversion

Source Languages

Ada
Assembly (HLASM, BAL)
Basic
C, C#, C++
CICS
CA-IDEAL
COBOL (many dialects)
COOLGEN & Other Generators
Dec Basic
EasyTrieve
Fortran
Java
JCL, DCL, Other Control Lang.
Jovial
MagnaX
MUMPS
Natural
PL/1
PL/SQL
PowerBuilder
Progress 4GL
RPG
Rust
SQL
VAX Basic
Visual Basic 6

Legacy Databases

Flat File Databases
Hierarchical Databases
Relational Databases

ISAM
VSAM
IMS
DB2
SQL
Sybase
Adabas
+ Other Databases

Stored Procedures
Triggers
Views

Legacy UI & Screens

BMS Maps
MFS Screens
Data Windows
Conversational
Pseudo-Conversational
CICS
In-lined Screen Code
+ Others UI Elements

Target Languages

C
C#
C++
Java
Java J2EE
Java J2SE
Angular
TypeScript
HTML5
EGL
VB.NET
PL/SQL
Python
Rust
+ Other Languages

Target Databases

Microsoft SQL Server
Oracle
PostgreSQL
MariaDB
MongoDB
NoSQL
Aurora
Distributed Databases
+ Other Databases

Target Platforms & Architectures

Multi-Tier,
Thin-Client Architectures,
DAO layer,
UI/Presentation layer,
Application layer,

Modern Frameworks :
Bootstrap,
HTML5,
CSS3,
Angular,
React,
Google Guice,
Google Closure

Web-Enablement
Cloud-Enablement

Fully Automated Introduction of:
RESTful interfaces,
Micro-Services
Cloud Architectures (AWS, On-Prem & Hybrid Clouds)
Cloud Native (AWS & TSRI services)

Mobile Enablement & Big Data

Automation Levels Across Languages



COBOL Source, Java Emissions, Java Emission Modified in Delivery (Pre-Refactoring)

Application 1:

COBOL LOC: 79,933
Java LOC: 120,656
Hand Modified LOC: 384
 $384/120,656 = .003$ **99.97%**

Application 2:

COBOL LOC: 361,385
Java LOC: 510,996
Hand Modified LOC: 34
 $34/510,996 = .00006$ **99.994%**

Application 3:

PL/1 LOC: 785,215
C# LOC: 798,899
Hand Modified LOC: 3
 $3/785,215 = .00002$ **99.998%**

Application 4:

MUMPS LOC: 3,928,813
Java LOC: 4,229,274
Hand Modified LOC: 4167
 $4,167/4,229,274 = .000572$ **99.902%**

Applications 4+ beyond: 99.9997%+ Automation with Minimal Adaptation

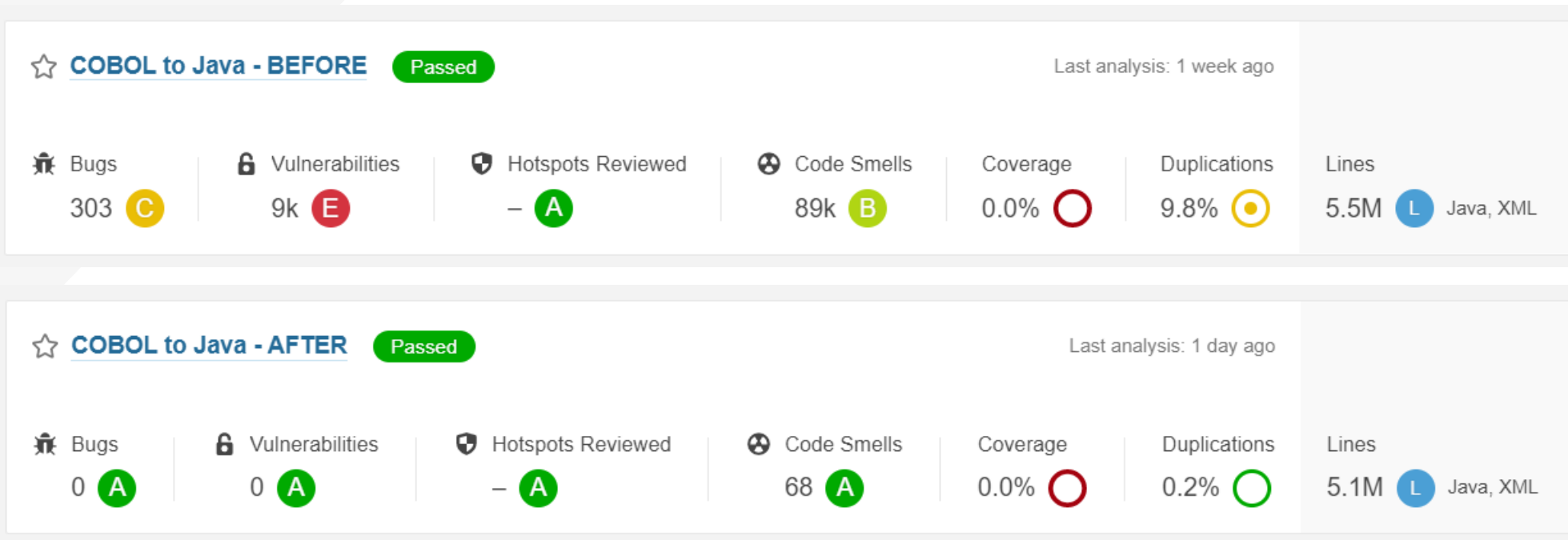
Why is Automation Important and What Are the Benefits of an Automated Modernization?

- Reduced Overall Budget, ~80% savings of TCO, Resource Availability & Cost of Resources
- Accelerated Schedule, Faster Time-to-Market/Deployment, Increased Agility to Market Needs
- Rapid & Iterative Process to Customer Specifications to improve code quality & maintainability
- Reuse of Transformation & Refactoring on Application Portfolios with Similar Technologies
- Significantly Fewer Defects/Errors (<1 defect every 20,000 LOC – covered under Warranty)
- Average Code, Database & UI transformation completed in 3-4 months
- Automated code refactoring accelerates manual improvements across millions of lines of code

Quality Refactoring: COBOL to Java

- Continuous & iterative refactoring improves vulnerabilities & maintainability
- Rules are applied to address the entire codebase & reused on subsequent applications
- Automation saves time, money, & resources by achieving improvements in a few minutes, what would take thousands of hours while reducing technical debt

Banking customer example below with 24,500 (\$1,225,000) hours of technical debt:



Iteration 1

<23 hours of refactoring

Iteration 2

Python – Emissions Quality and Progress

Initial python emissions were very good,
and still are.

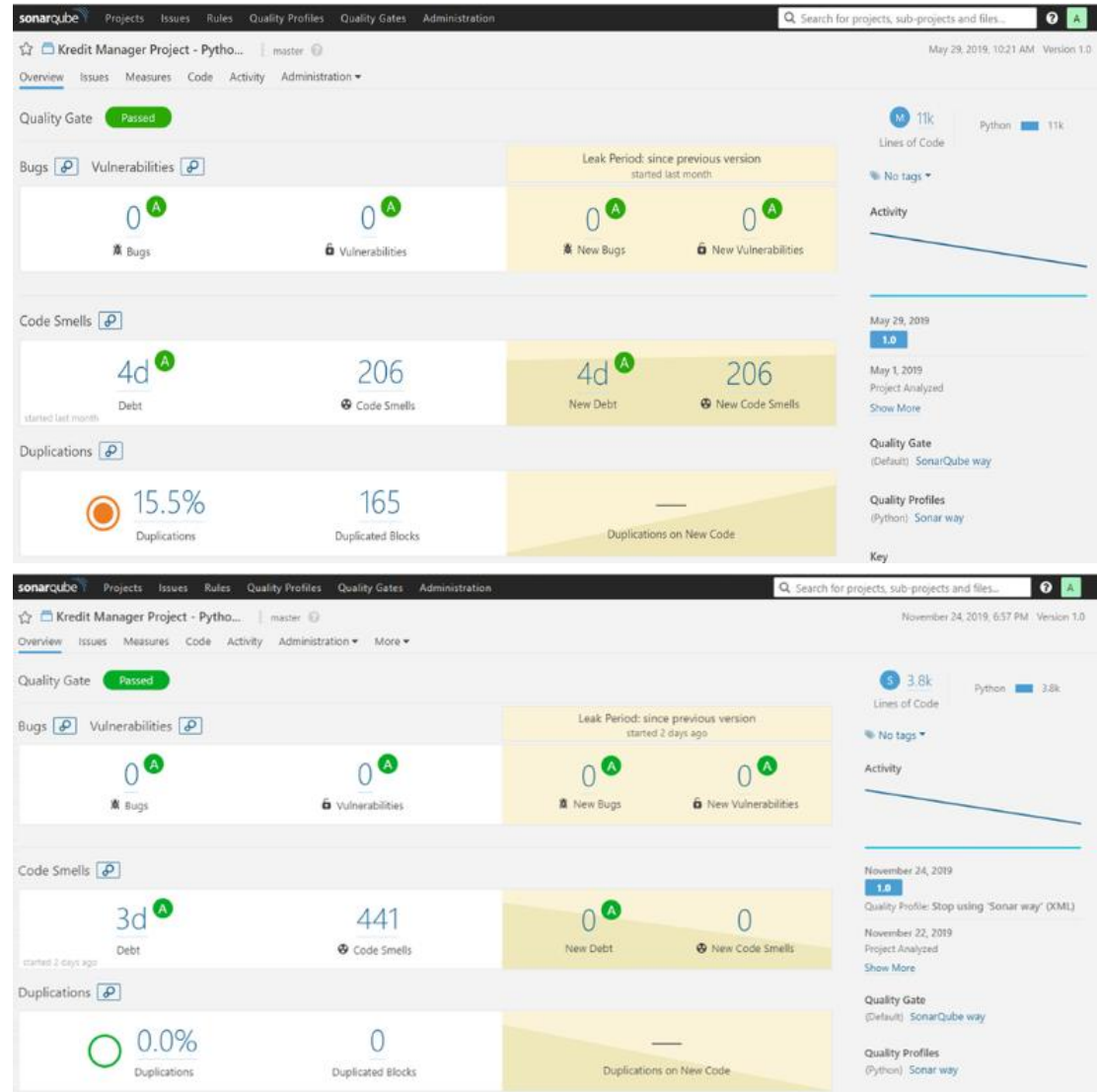
- No Bugs or Vulnerabilities

The strength of the JCL to Python transform
has remained excellent since project start

- Very small amounts of debt

Note: Bugs, smells and debt fluctuate as a
result of refactoring

- TSRI will continue to review with DB to
support *practical* resolutions



USAF Standard Base Supply System (SBSS) ILS-S

Program Description

- Integrated Logistic System
- Utilized by the United States Air Force, Air Force reserve, and Air National Guard.
- Mission critical system used by 18,000 users.
- Used at 260 locations.
- 54-year-old system written in legacy languages and on decaying technology.
- \$30B of USAF inventory tracked through SBSS system.
- Operating expenses of \$16.5M+

Program Approach

- Approximately 1.5M LOC COBOL and C Transformed to Java using automation.
- Utilized the OMG® GASTM and UML Standards.
- Completed ahead of schedule, under budget.
- Included move to cloud, big data, mobile release.
- Modern Web application, reduced O&M cost, increased availability/faster time to market.

TSRI



Key Accomplishments/Status

- Containerized deployment on USAF Cloud One AWS GovCloud.
- Transformed 100% of the code using 99.97% automation.
- \$25M hosting cost reduction & savings to \$3M TCO per year
- Less than 1 defect per 21,000 LOC – covered under warranty.
- Identical record and DAO method consolidation.
- Rearchitected to AWS cloud allowing for disaster recovery and AWS services.

HUD Unisys COBOL to Java Modernization Factory

Program Description

- TSRI & Partners modernized 4 major HUD applications over the course of 2 years.
- First major modernization out of the MGT Act as part of the TMF Funding.
- Last remaining applications at HUD on the Unisys mainframe
- Aged system written in Unisys COBOL & on decaying technology.
- Extremely high visibility at the GAO & Federal government levels.

Program Approach

- Approximately 1,700,000 LOC Unisys 2200 COBOL Transformed to Java (over 4 applications) using model-driven automation, refactoring & documentation.
- Completed ahead of schedule, under budget.
- Intensive refactoring to adhere to coding standards (PMD/Checkstyle).
- Increased efficiency, automation, and accuracy in each subsequent application (factory approach through 4 applications).
- Continued support through TSRI Support & Maintenance Agreement (SMA).



Key Accomplishments/Status

- Transformed using 99.98% automation.
- All 4 applications through transformation phase in the first year.
- Copybook and DAO method consolidation.
- 4 of 4 applications now in production as of July 2021 & December 2021.

• Deployed with .NET Core on Microsoft Azure Cloud
Media resources: [Fed. News](#) 2021 // [Fed. News](#) 2020 // [Fed. News](#) 2019

USAF Reliability & Maintainability Information System (REMIS)

Program Description

- Utilized by the United States Air Force Technical maintenance teams and its contractors.
- 250,000 daily users
- Aging and critical data system for reporting the entire USAF equipment inventory.
- Re-modernize a system transformed from COBOL to C++ to COBOL and C++ to Java.
- Little to no documentation or understanding of the system.

Program Approach

- 3.1M LOC COBOL and C++ transformed to modern Java.
- Utilized the OMG® GASTM and UML Standards.
- Documentation produced to reduce maintenance effort.
- Create a modern and maintainable system on reliable infrastructure.
- Integration into the Global Combat Support System framework.

TSRI



Key Accomplishments/Status

- Transformed using near 100% automation.
- Project completed 2 months early, \$449,000 under budget.
- Deployed at more than 1,000 Air Force sites worldwide
- Reduced O&M cost by 60%, enabled use of less expensive resources & less expensive infrastructure.
- Second full modernization of REMIS system by TSRI in 10 years.

US Navy PMW150 – (NTCSS System): PowerBuilder to Java

Program Description

- Utilized by the United States Navy and its contractors.
- Influential to Pentagon level leadership and decision makers.
- Essential legacy application for managing maintenance of ships and aircraft.
- Migration from unstable and unsupported technologies to modern languages/architectures.
- Demonstrate capability of using automation to modernize PowerBuilder language DoD systems



Program Approach

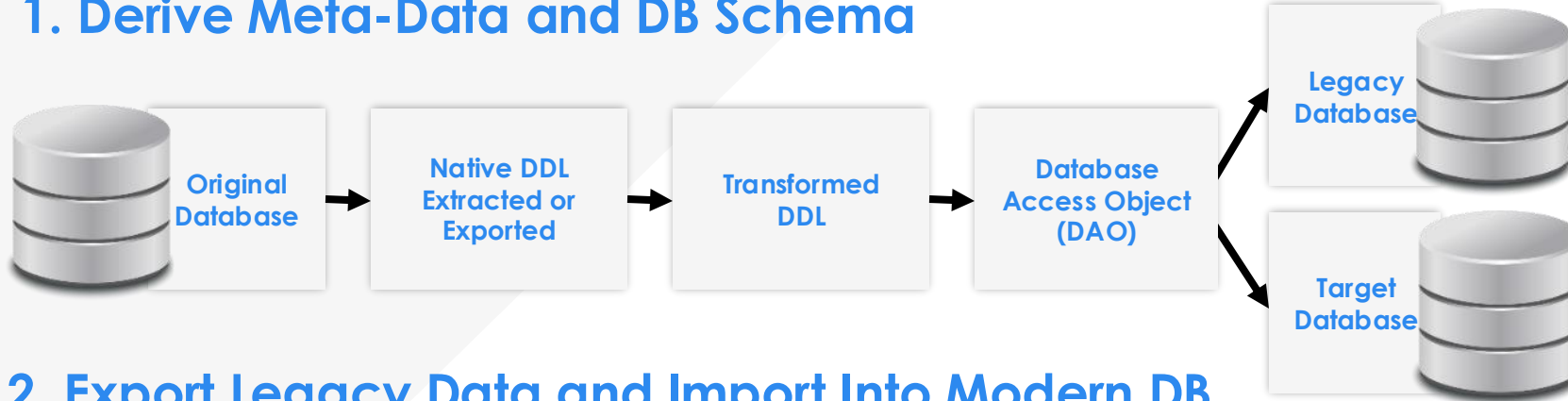
- Automated modernization of highly complex PowerBuilder system to Java, including Data Window Migration.
- Utilized the OMG® GASTM and UML Standards.
- Transformed over 700,000 LOC of PowerBuilder, including resolution to support 20 critical test cases
- Successfully transformed a highly complex US Navy module to Java.

Key Accomplishments/Status

- Transformed using near 100% automation.
- Modular and open layered architecture for ease of use/maintainability.
- Demonstrated the ability to use automation to move 4GL language systems into modern multi-tier architectures

TSRI Automated Database Migration

1. Derive Meta-Data and DB Schema



DDL is transformed and migrated.

- There is the opportunity to look at modernizing field types and filtering aged data.
- The model for the new database represents the same older schema as it must be, to run the legacy application logic.
- Dual DAO can support multiple Databases

ETL script for migrating the data itself.

- This is typically done with test data or scrubbed data for the test and is required for ongoing support for the project.
- The process is done multiple times and characterized.
- Experience in leveraging 3rd party tools including IBM utilities and Microsoft's SSIS.

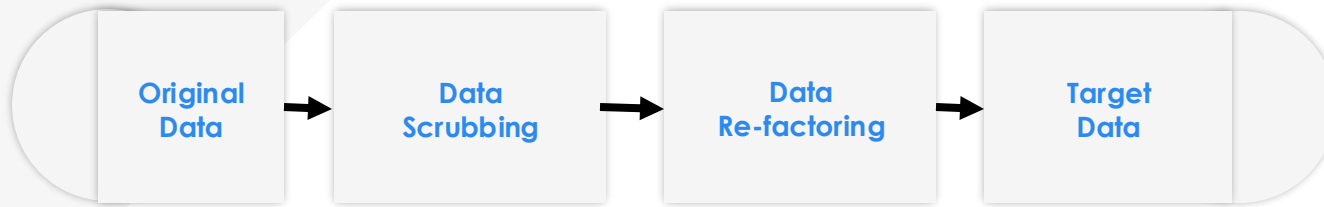
Complete end to end database validation

- Extraction from legacy database extraction and modern database represented in common file format where databases are compared.
- Independent validation of the entire process. The process is automated and can be repeated multiple times.
- Test cases exercising the application require that the data schema and tables are correct an additional separate validation of data set.

Experience in migrating the data,

- Synchronizing latest data changes and meet GoLive time windows.

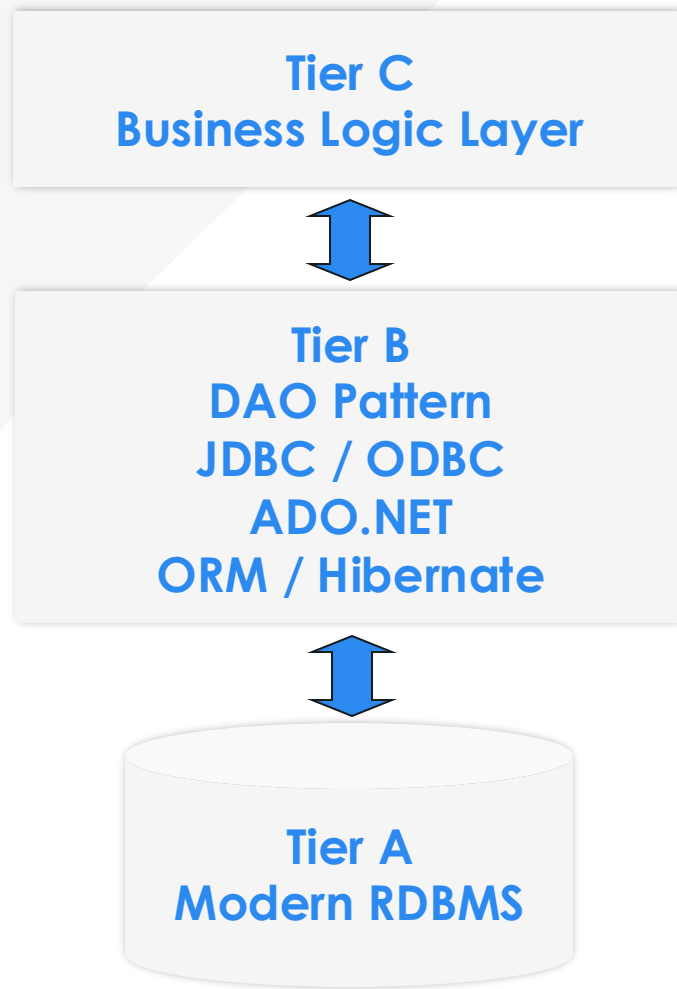
2. Export Legacy Data and Import Into Modern DB



3. Round-Trip Data To Prove Equivalence



3 – Tier Legacy Data Base Architecture



DDL is transformed and migrated.

- There is the opportunity to look at modernizing field types and filtering aged data.
- The model for the new database represents the same older schema as it has to be, to run the legacy application logic.
- Dual DAO can support multiple Databases

ETL script for migrating the data itself.

This is typically done with test data or scrubbed data for the test and is required for ongoing support for the project.

The process is done multiple times and characterized.

Experience in leveraging 3rd party tools including IBM utilities and Microsoft's SISS.

Complete end to end database validation

Extraction from legacy database extraction and modern database represented in common file format where databases are compared.

Independent validation of the entire process. The process is automated and can be repeated multiple times.

Test cases exercising the application require that the data schema and tables are correct an additional separate validation of data set.

Experience in migrating the data,

Synchronizing latest data changes and meet Go-Live time windows.

TSRI Migration to the Cloud – 3-Step Approach

1. **Migrate Monolithic Applications to Service-Oriented Applications**
 2. **Service-Oriented to Native-Cloud Services**
 3. **Native-Cloud, Docker, Kubernetes & Containers, Lambda/API Gateway- Server-less Framework**
- **Future State Cloud Target Support**
 - True source to target replication of framework, business logic and UI.
 - Library approach for legacy framework, control and batch
 - Refactoring/Mapping to move all JCL, external type functionalities to Cloud library
 - **Automatic Transformation Ensures no Loss in Functionality**
 - True source to target replication of framework, business logic and UI.
 - Library approach for legacy framework, control and batch
 - Refactoring/Mapping to move all JCL, external type functionalities to Cloud library

TSRI Modernization to the Cloud Overview

TSRI has already completed modernizing mainframe, mid-range and thick-client legacy applications to AWS, Azure, Oracle, Google Cloud & OpenShift.

These already created transformation and refactoring rules can immediately applied to the current modernization strategy or at later stage as well.

With minimal adaption, TSRI can move the currently modernized applications to the cloud using cloud-specific refactorings – targeting services, micro-services, containerization, etc...

Cloud-specific refactorings will be reused on all follow-on applications (one-time Engineering Effort for significant reuse)

TSRI can also adapt its process to other cloud architectures as well.



Amazon Web Services & Microsoft Azure Cloud

Amazon Web Services

- TSRI has been a long-standing Amazon Web Services partner
- DocsRev, Documentation-as-a-Service, is an AWS Cloud Service
- TSRI has moved multiple COBOL, JCL systems to the AWS Cloud



Advanced
Technology
Partner

Microsoft Azure

- TSRI modernized multiple critical systems to Azure (TJX, eBoks, Boeing)
- Languages such as COBOL, JCL, VB6, VB .NET were modernized to C#
- TSRI is a Microsoft Partner Network partner

Microsoft
Azure

Certified

TSRI Cloud Migration Highlights

TSRI has supported migration to Multiservice Architectures, Deployed in Multiple Cloud Environments

- Multiple DAO data layer injection allowing for interface to legacy and modernized data servers
- Encapsulation in containers with Rest service enter to Business Layer, transformed programs
- Rest API for common externals, bridges and legacy interfaces.
- Modernized GUI, supportable and extendable browser presentation layer.

Batch Programs

- Unique REST API (Batch) Endpoints, Batch programs now get exposed through REST endpoints to a python client
- Quintessential in getting batch programs to be executed on containers, separate from their corresponding python clients.
- Introducing filters and routing middleware to ensure certain containers exposed only certain batch programs.
- A status monitor allow for monitoring status of long running jobs that were invoked asynchronously. Easily integrated with Business Apps or 3rd party schedules and under management.

Online Programs

- Generated API interfaces for business logic program and exposed them through adapter classes to be called by the web service
 - Business logic programs decoupled from the web controllers for the program to be injected through service providers.
 - Allows for introducing custom pre- and post-execution codes for each transformed program, no altering the business logic.

Externals & Framework

- TSRI has over 150 libraries for target language, handling difficult, unique and varied data types issues.
- TSRI has library of externals and interfaces to support common externals API including DB Message Bridge

Cloud Migration

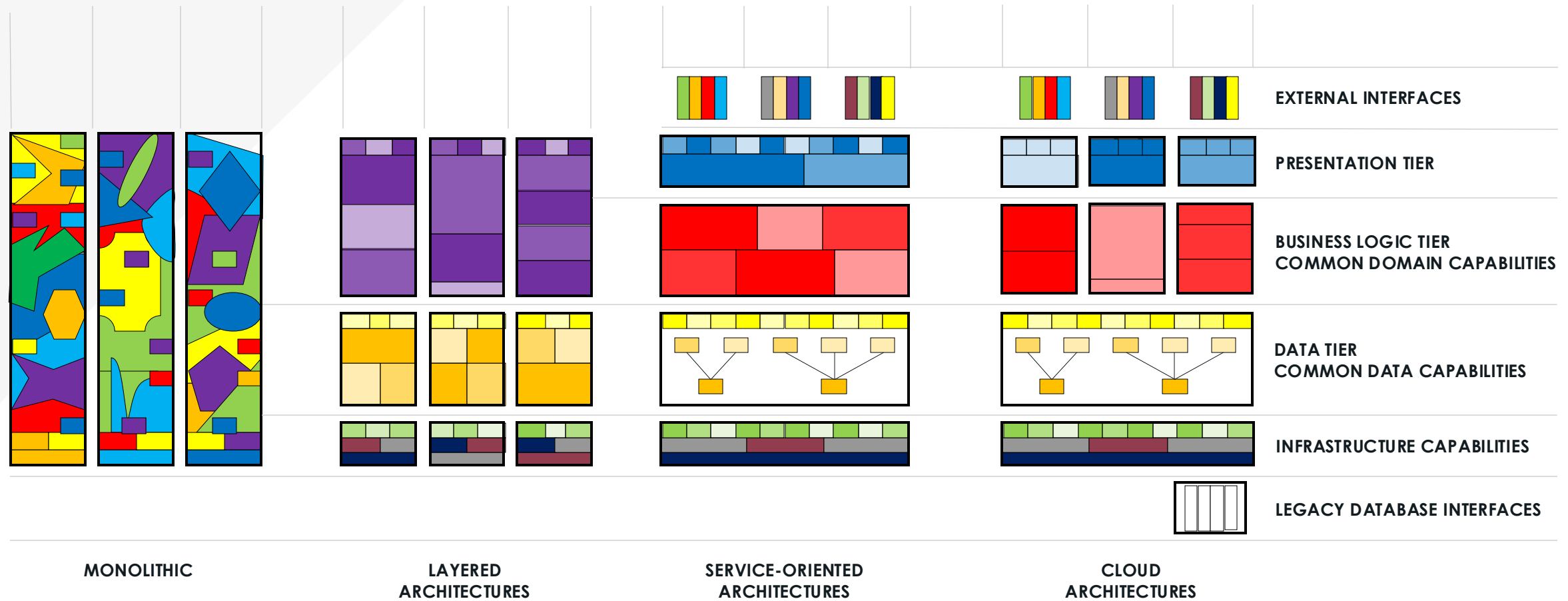
Migrate Monolithic Applications to Service-Oriented Applications

- Assessment, Analysis, Target Architecture and External Dependency Resolutions
- Transformation Sprints, Spirals of Increasing Size and Complexity, Deployment, Integration Testing
- Comparative Testing, Regression Testing, Refactoring for Performance and Code Quality
- CI/CD Multi-environment Deployment for QA, Scalability/Performance, Pilot and Production
- Deploy Docker, Kubernetes, Containers, AWS Services, API Gateway

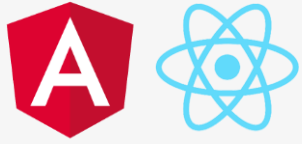
Automatic Transformation Maintains Business Logic, Ensures No Loss in Functionality - with Code Warranty

- 1:1, Like-for-Like, Source-to-Target Replication of Business Logic, Workflow and UI
- Library Approach for Legacy: Framework, Externals, Custom Implementations, Utility Functions to Cloud
- Automated Refactoring Applies Rules to Improve Codebase Without Changing Business Logic
- Allows for Changes to the Legacy Source Code at Any Point in the Project Without Business Stoppage

From Monolithic, Layered, SOA to Cloud Architectures



Java Technology Stack



UI
Service

Angular,
React,
Bootstrap,
HTML5, CSS,
jQuery,
AJAX, JSON

Application
Service

Java
Spring Boot

Spring Boot
embedded
Tomcat

Batch
Client

Python

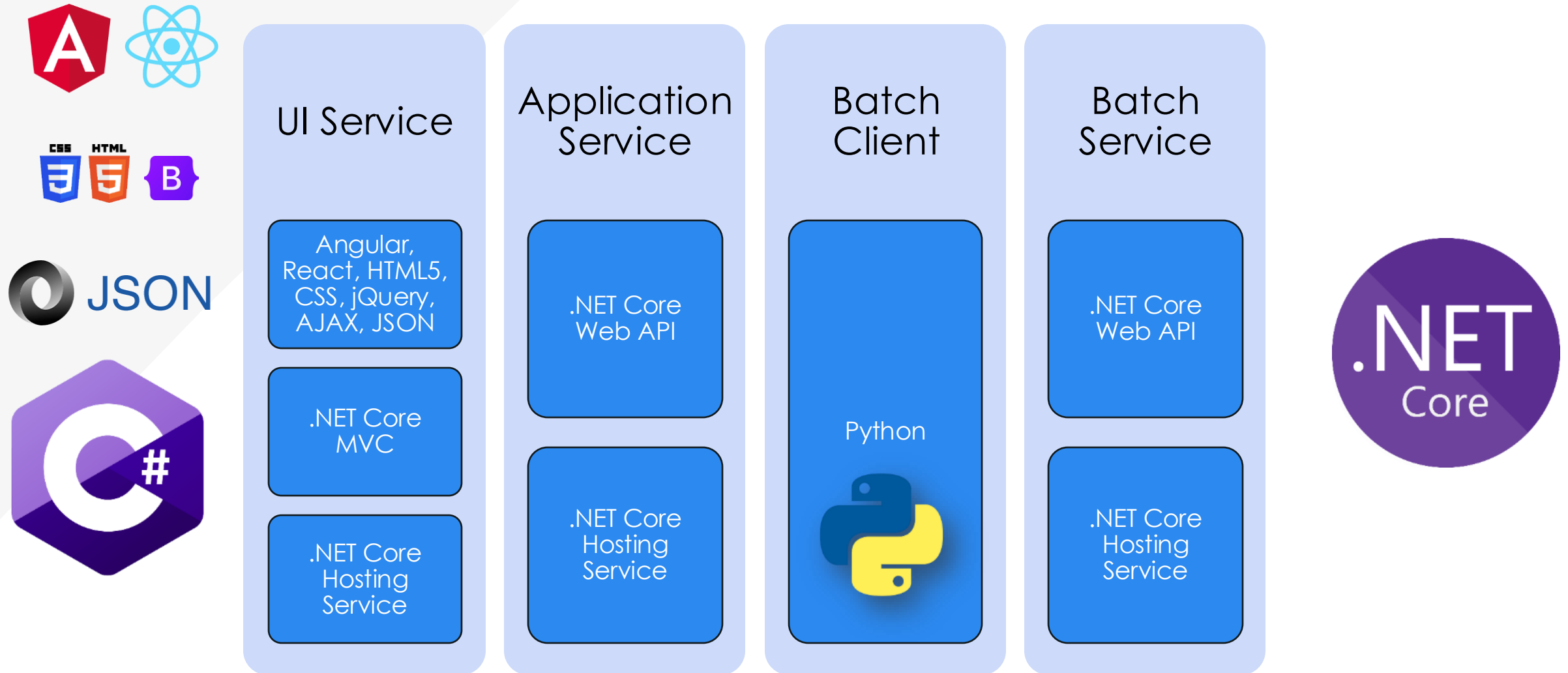


Batch
Service

Java
SpringBoot
or
SpringBatch



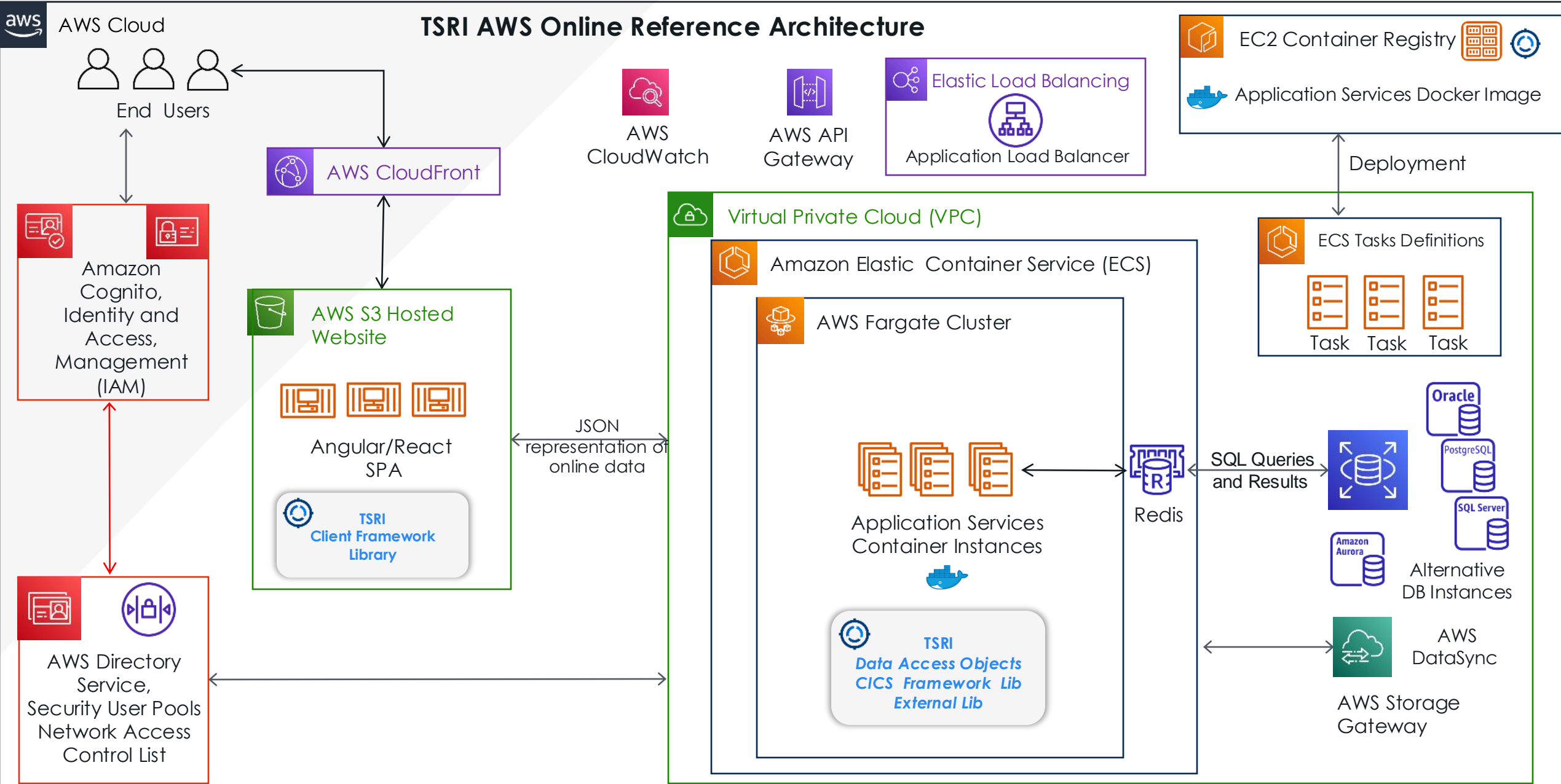
C# .NET Core Cloud Technology Stack





AWS Cloud

TSRI AWS Online Reference Architecture



SOFTWARE MODERNIZATION ASSURED

© TSRI. All rights reserved.

TSRI



TSRI AWS Batch Reference Architecture



AWS CloudWatch



EC2 Container Registry



Batch Services Docker Image

Deployment



ECS Tasks Definitions



Task



Task



Virtual Private Cloud (VPC)



Amazon Elastic Container Service (ECS)



AWS Fargate Cluster

AWS Batch Scheduler



AWS API Gateway



AWS MQ



AWS Simple Queue Service (SQS)



Batch Job Python Script

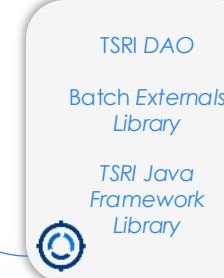
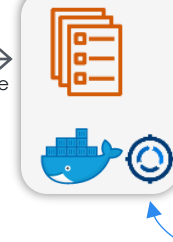
Batch Rest Services



Sync-Mode



Batch Externals Service



Batch Externals Library
TSRI Java Framework Library



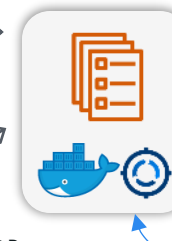
Async-Mode



TSRI Job Status Manager

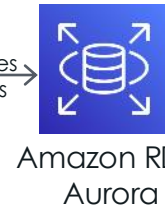


TSRI Job DB Query (Job_Id)



TSRI DAO
Batch Externals Library
TSRI Java Framework Library

SQL Queries and Results



Amazon RDS Aurora



Alternative DB Instances



AWS DataSync



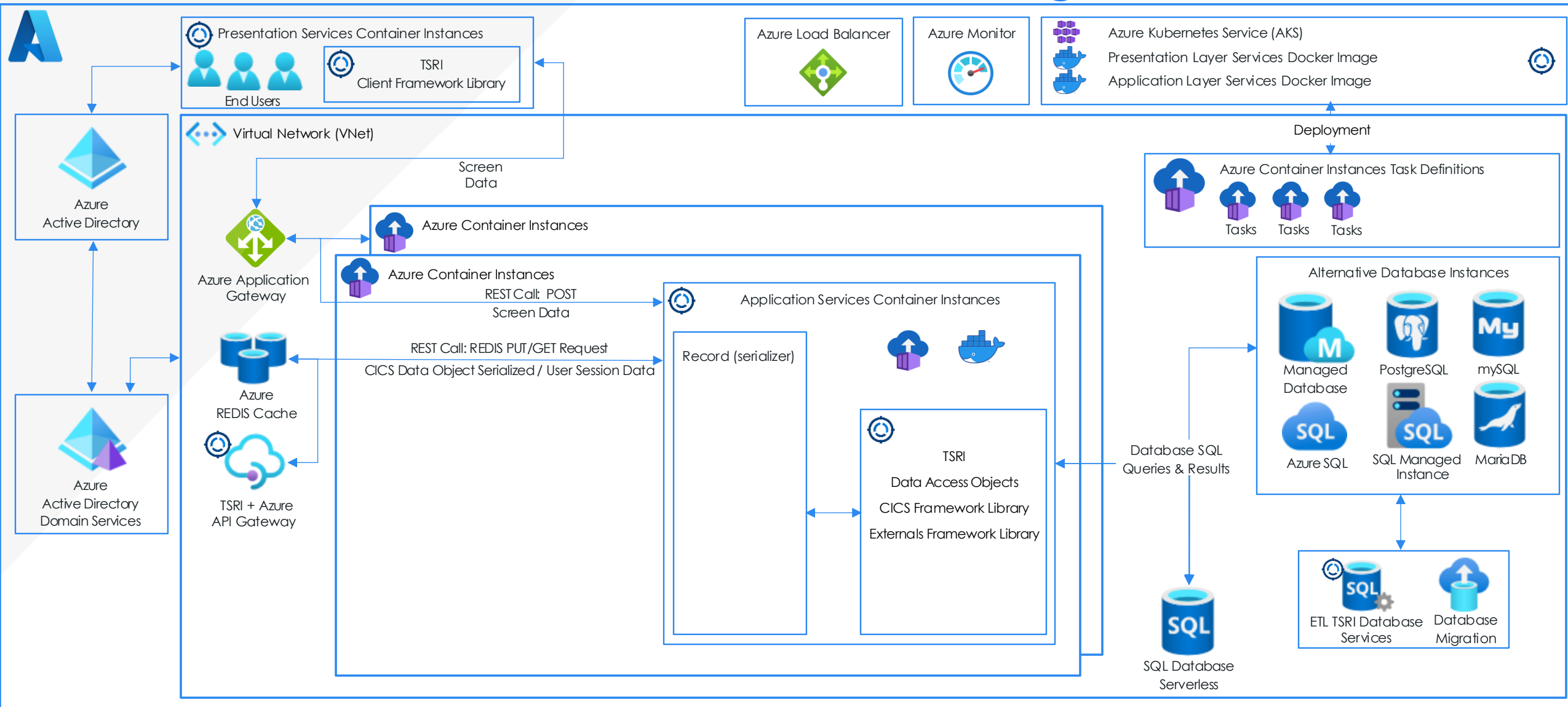
AWS Storage Gateway



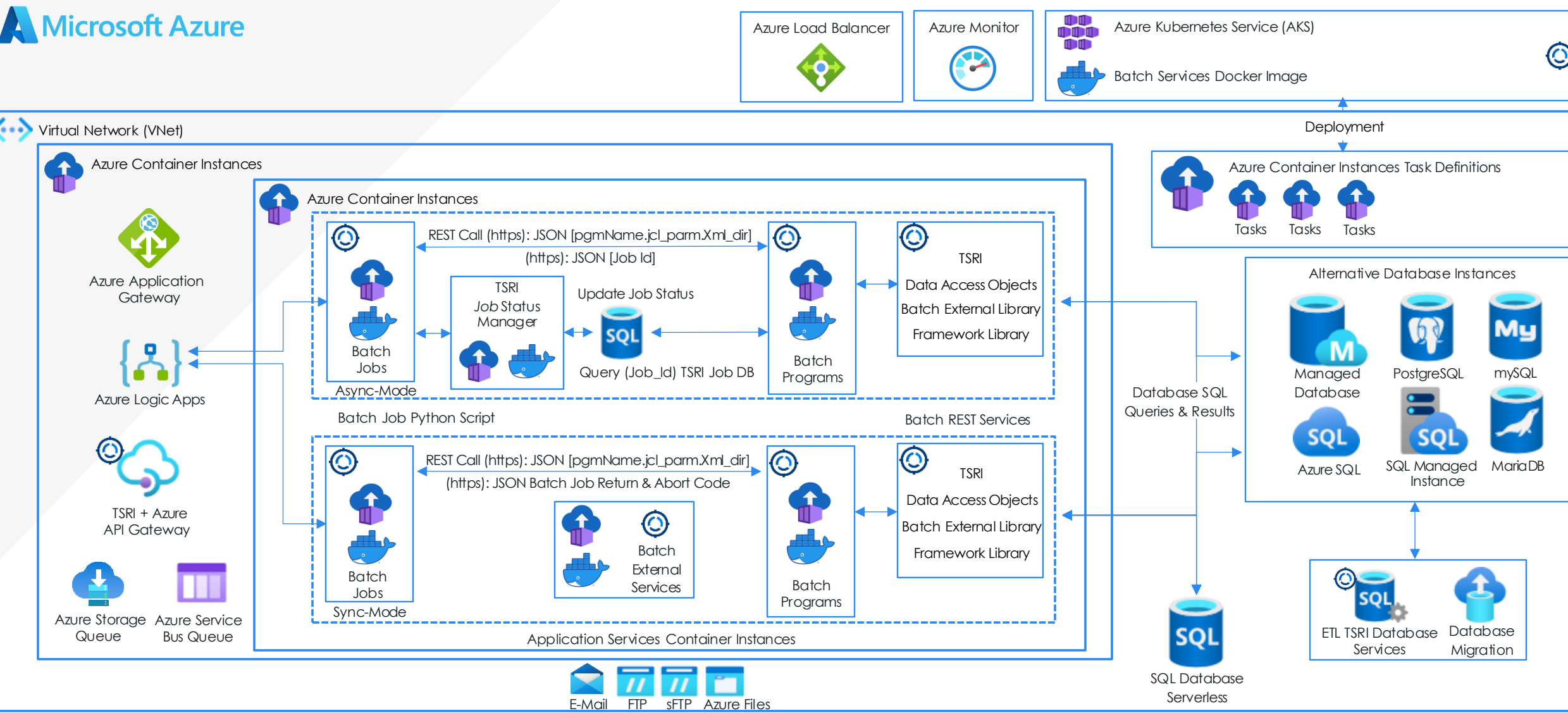
AWS Elastic File System (EFS)



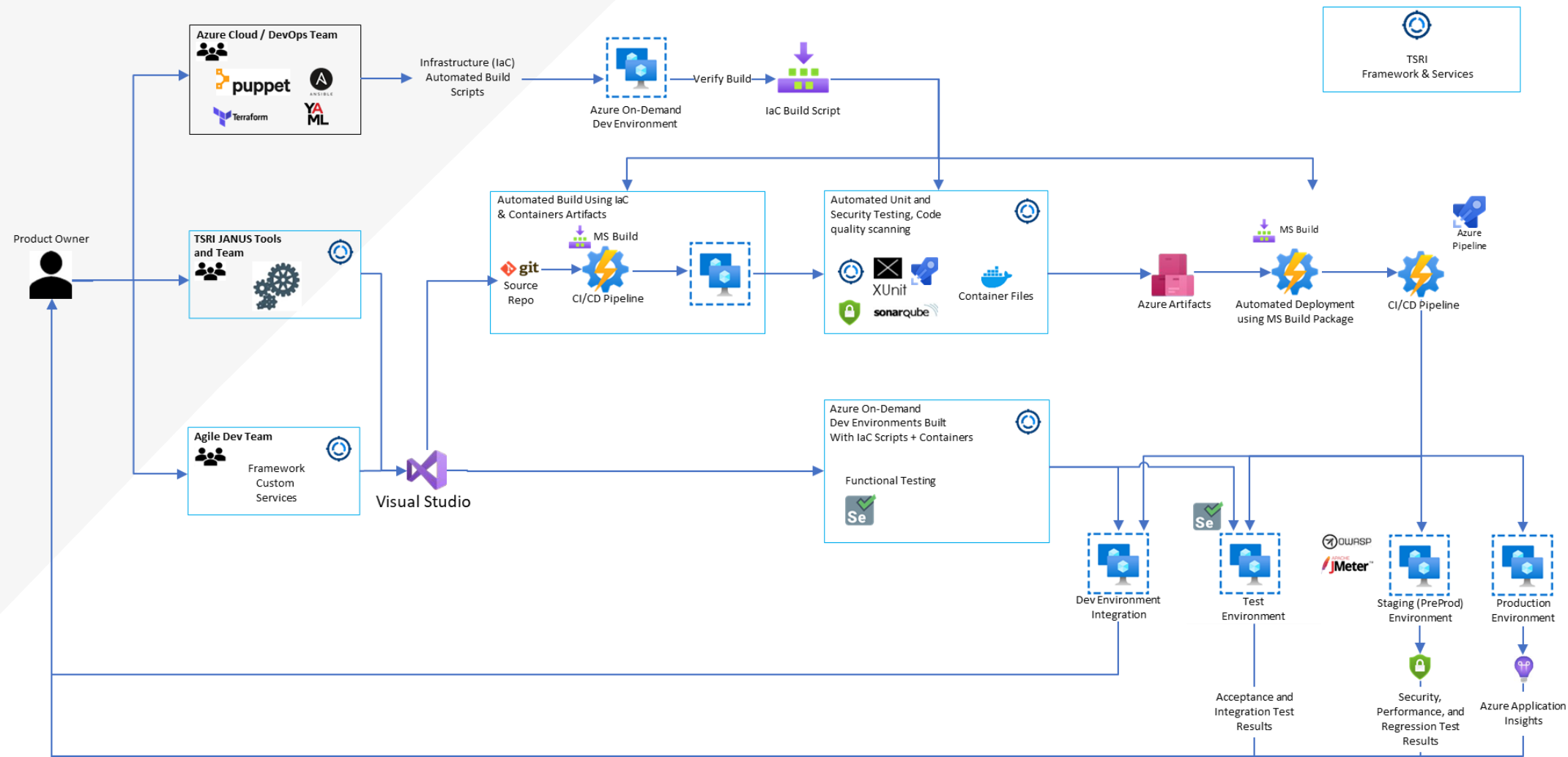
Microsoft Azure Online Cloud Architecture Diagram



Microsoft Azure Cloud Batch Architecture Diagram

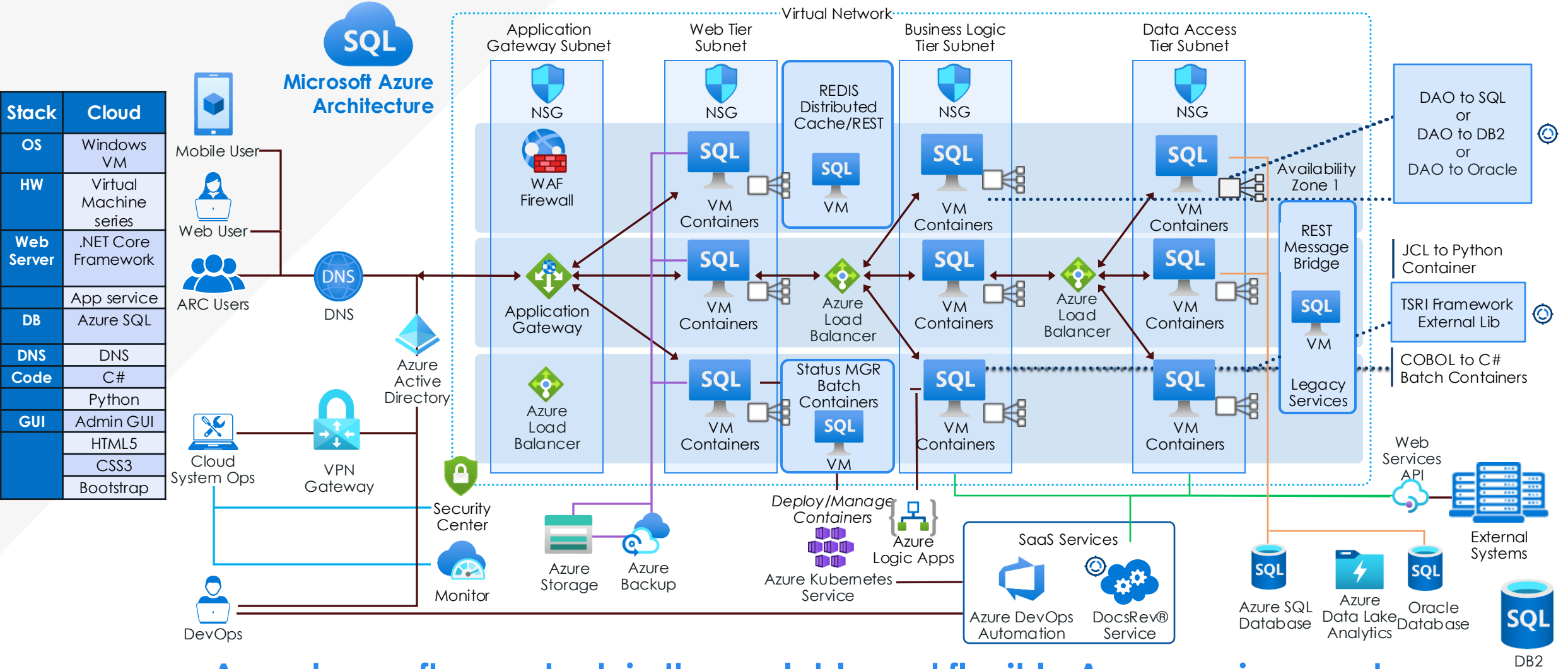


DevSecOps Pipeline on Azure Cloud



DevOps automation supports multiple development teams and continuous release cycles

Integration on Microsoft Azure Cloud



A modern software stack in the scalable and flexible Azure environment

COBOL to C# .NET Core on Microsoft Azure Cloud Case-Study

Multi-Phased Modernization

- 1st Phase to Service-Oriented App 2019
- 2nd Phase Refactoring to Cloud-Services in 2019

Program Description

- TJX, a Fortune 100 worldwide retailer with \$39 billion revenue, 4,300 stores and 270,000 employees
- System for executing unique buying process resulting in industry-leading inventory turnover
- Buying process viewed as the catalyst for years of unprecedented growth and profitability
- 20-year-old system written in legacy COBOL & JCL on languages and operating on expensive hardware
- Support and maintenance were not aligned with Company best practices



Program Approach

- COBOL & JCL transformed to C# & Python with near 100% automation (99.9x%)
- End-to-end traceability of fully automated conversion process to ensure complete preservation of business logic
- Architecture refactoring to target Azure services enabling DevOps aligned with SAFe Agile sprints

Key Accomplishments/Status

- Modernized code delivered directly to Azure CI/CD pipeline
- Batch program architecture with single container image to simplify deployment and administration
- REDIS Cache isolation to provide CICS session state to all containers
- Efficient HTML rendering enabling Angular front-end development

COBOL to Java on Amazon AWS Cloud Case-Study

COBOL to Java on AWS Cloud Case-Study

[Detailed Case-Study on AWS](#)

[Live Demo on AWS EC2 & ElasticBeanstalk \(Serverless\)](#)

[AWS on EC2](#)

[AWS on ElasticBeanstalk](#)

Multi-Phased Modernization (1st Phase to Service-Oriented App 2018, 2nd Phase Refactoring to Cloud-Services in 2018)

Program Description

- Utilized by the United States Air Force, Air Force reserve, and Air National Guard.
- Mission critical system used by 18,000 users.
- Used at 260 locations all over the world.
- 54-year-old system written in legacy COBOL & C languages & on decaying technology.
- \$30B of USAF inventory tracked through SBSS system.
- Operating expenses of \$16.500.000+

Program Approach

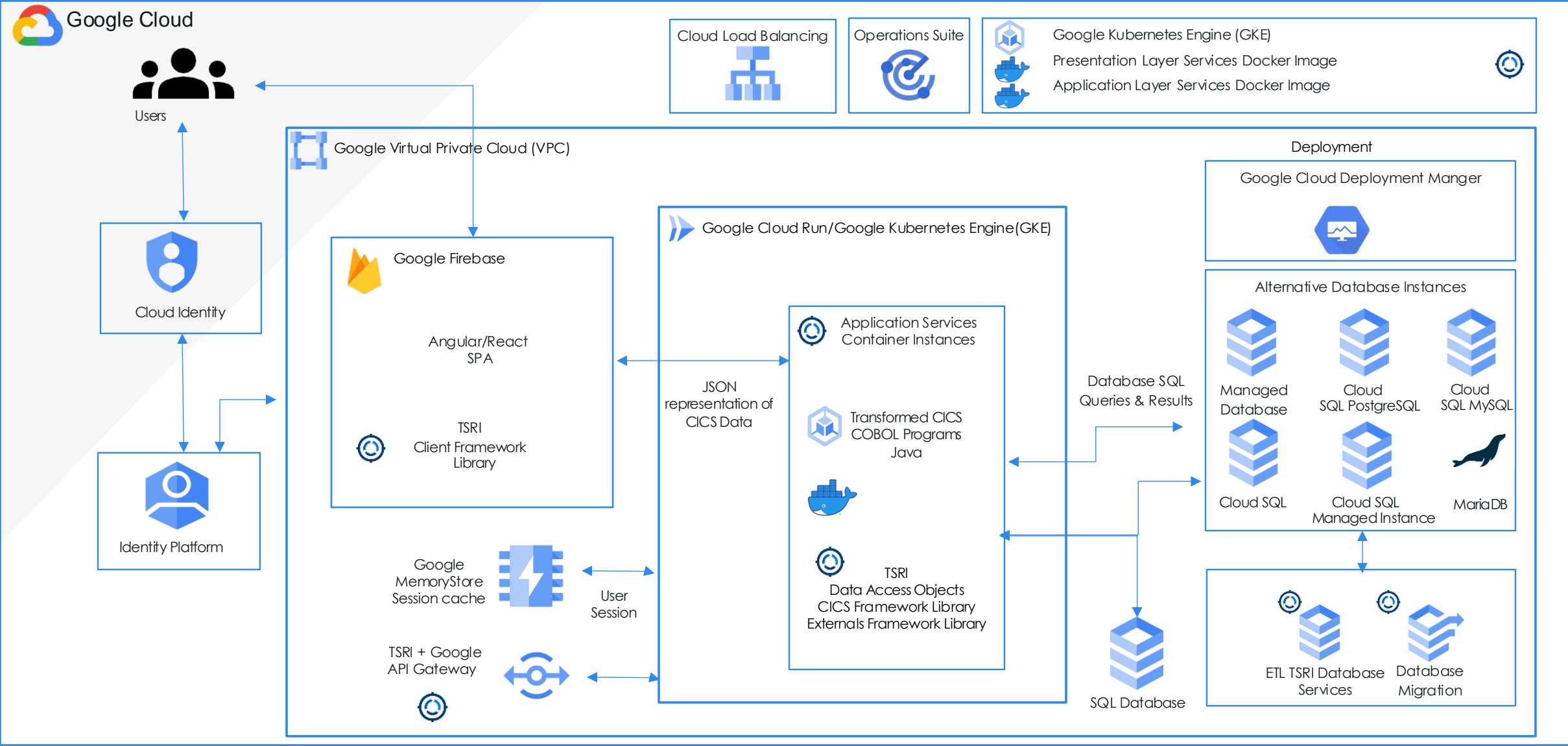
- Approximately 1,300,000 LOC COBOL & C Transformed to Java using near 100% automation.
- Completed ahead of schedule, under budget.
- Included move to cloud, big data, mobile release.
- Modern Web application, reduced O&M cost, increased availability/faster time to market.



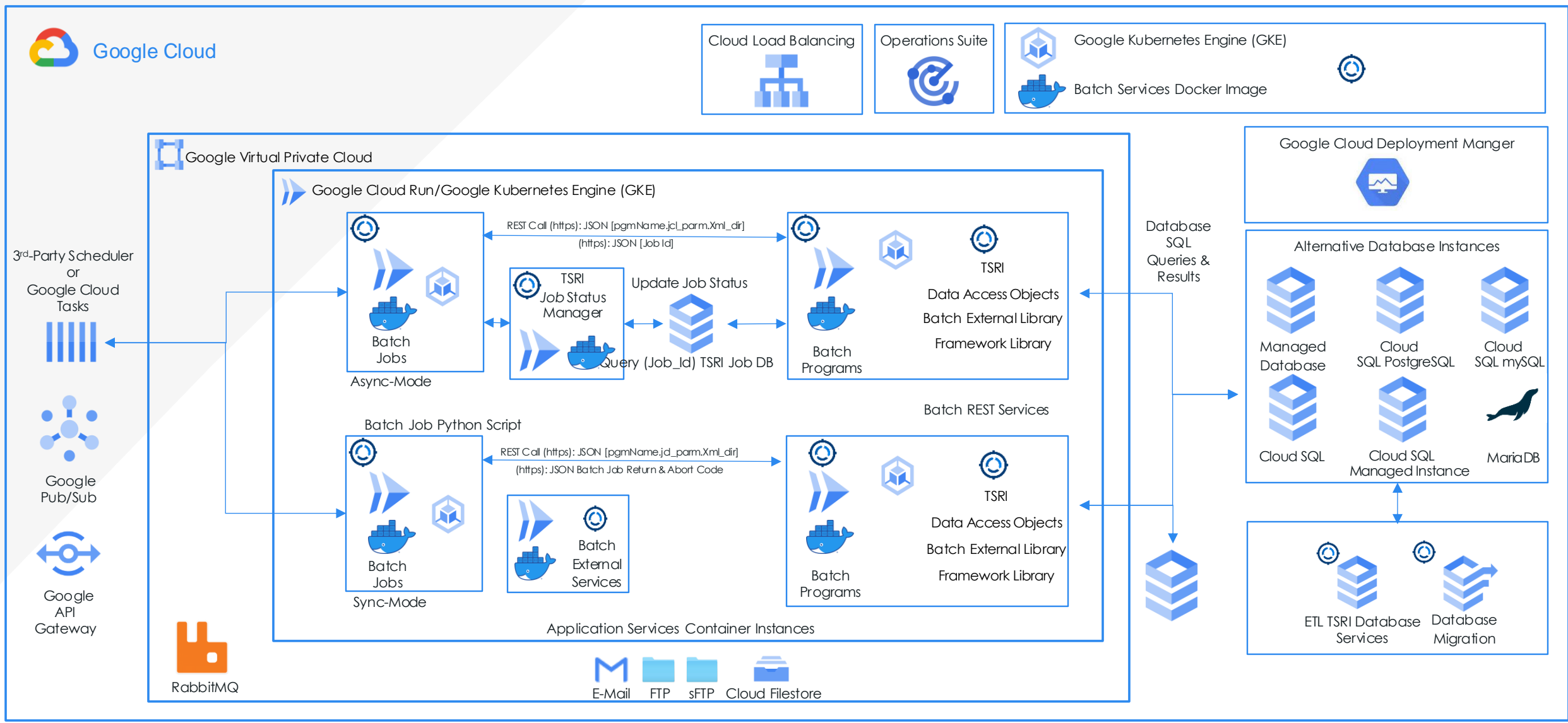
Key Accomplishments/Status

- Transformed using near 100% automation.
- 1 defect per 21,000 lines of transformed code.
- Identical record and DAO method consolidation.
- Rearchitected to AWS cloud allowing for disaster recovery and AWS services.

Google Cloud - Online Reference Architecture Diagram



Google Cloud - Batch Reference Architecture Diagram

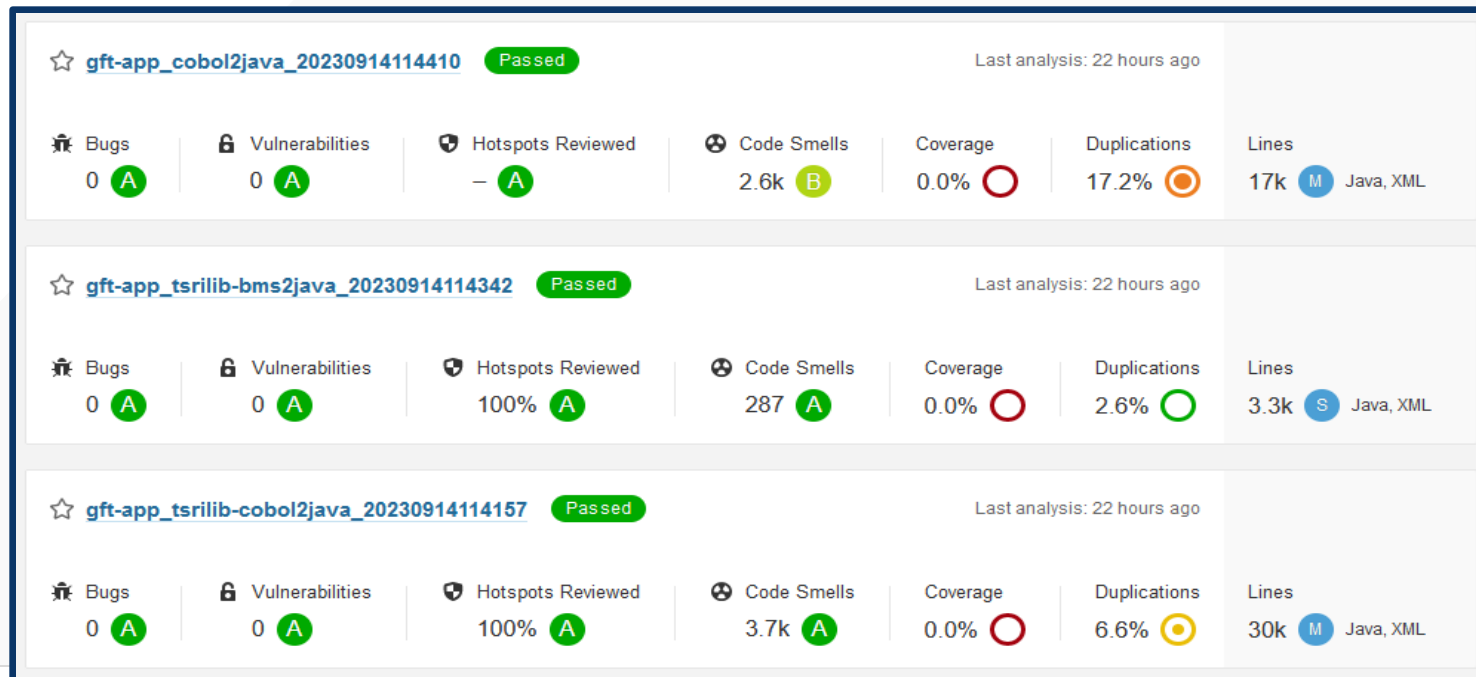


TSRI Container Support

- **REST API supports Containerized deployments where only communication is through network calls**
 - TSRI has been migrated over time from the .traditional WCF (Windows Communication Foundation) framework .to the modern .NET Core WEB API The .NET Core API allows for deployment on both windows and Unix containers
 - TSRI has also migrated to a multi tier approach that support presentation service separation from business logic, offering migrated programs with modern interfaces and maintainable
 - Online module for the transformed applications rely on user session data, there was a need to ensure that this session is consistent across all replicas.
 - The Redis distributed cache was introduced as an external component to maintain user sessions. We also took advantage of the cache to store user specific CICS data thereby eliminating the need for the presentation layer to send the entire CICS data across the http rest calls it makes to the business logic REST API
- **TSRI supports Docker containerization process, docker file template for .NET Linux/Windows, build and execution commands for the service intended to be deployed.**
- **Kubernetes provides orchestration and management of container deployment**
 - The resulting docker image can be pushed to Azure Container Registry and executed directly or via the Azure Kubernetes Service.
 - Kubernetes also allows for deployment specific configuration values to be specified in the .yaml file which can be read as environment variables by the programs at runtime.
 - Kubernetes allows for replication of containers. load balancing across these replicas.

Reservations Application Online/Batch DEMO Overview

- IBM Mainframe Application for making class reservations and accepting payments written in COBOL and JCL.
 - UI allows for entering and updating students/customers, making payments and seeing class availability
 - Batch jobs build current monthly calendar for center, extracts and categorizes data, maps classes and rooms between active and inactive centers, validates students/customer and employee and updates files
 - TSRI targets Java for Backend and Angular/Typescript for Front End, jobs were transformed to Python
 - TSRI Sort leveraged as external replacement
 - TSRI targets triple A Sonar scores for Bugs, Vulnerabilities and Hotspots



Automation Rates

COBOL to Java

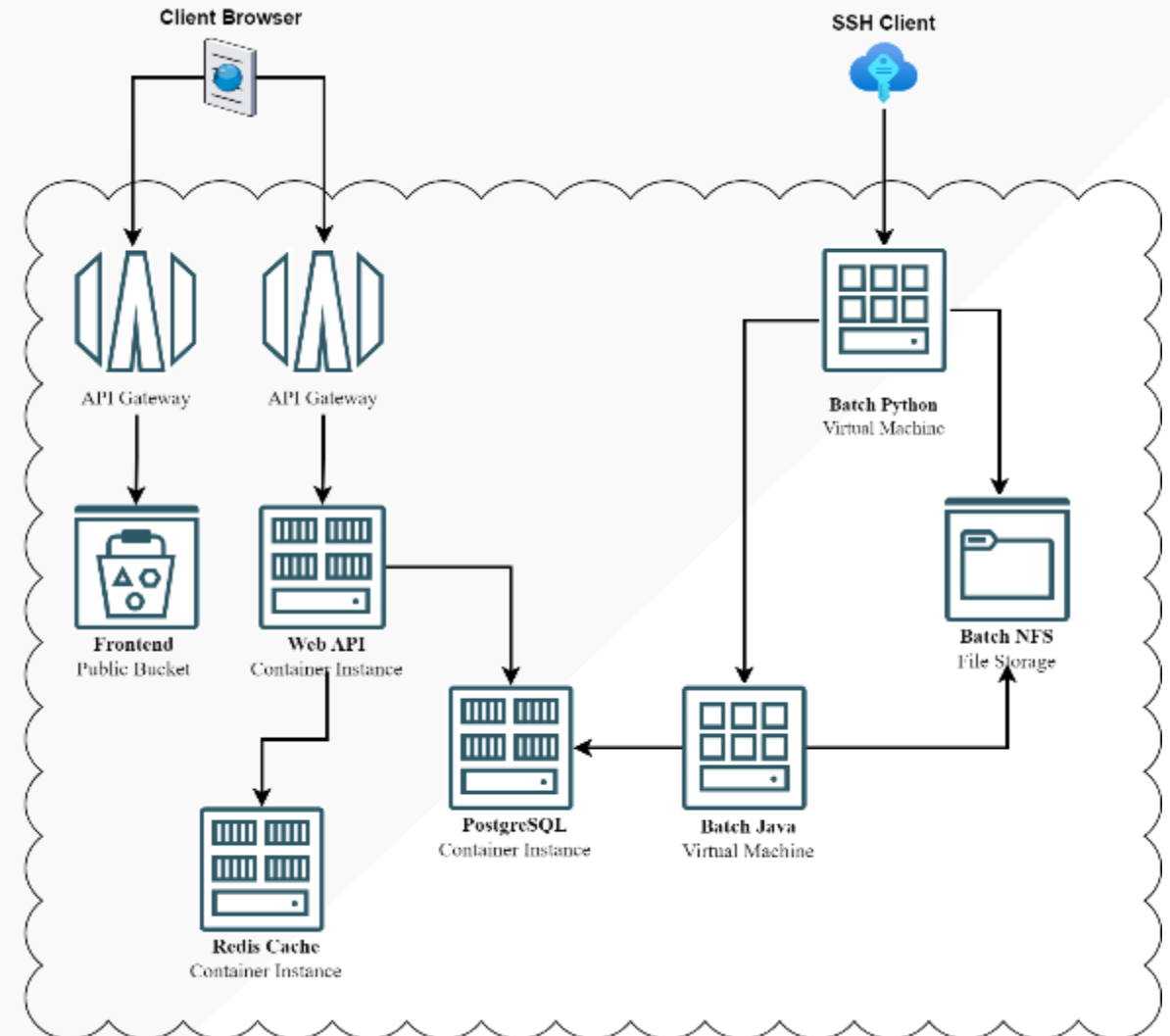
- 10,970 COBOL LOC
- 100.00% Automation
- 0 manually patches

JCL to Python

- 2,040 JCL LOC
- 99.71% Automation
- 12 manual patches

Demo Architecture Reservations Application Online and Batch

- COBOL to Java, JCL to Python, on OCI
- Deployed Angular Front-end framework with TSRI framework
- WebApi Containerized Backend
- Support session caching with Redis
- Python Batch Job deployed in VM
- Batch Containerized Backend
- Batch Network Files Share





Thank you for your time!

The Software Revolution, Inc. (TSRI)

11332 NE 122nd Way, Suite 300

Kirkland, WA 98034-6949

USA

Info@tsri.com

+1 (425) 284-2770



Scott K Pickett

Vice President, Product &
Service Delivery

skpickett@tsri.com



Bradley Charleson

Principal Account Executive

bcharleson@tsri.com



Joseph Prikhodko

Business Development
Manager

jprikhodko@tsri.com