Progress®

Moving to OpenEdge 12: Lessons Learned and Best Practices

Shelley Chase, Software Fellow, Progress

Edsel Garcia, Software Architect, Progress

October 2019

“Support Adoption of OpenEdge Products for Customer Engagements…”

A-team

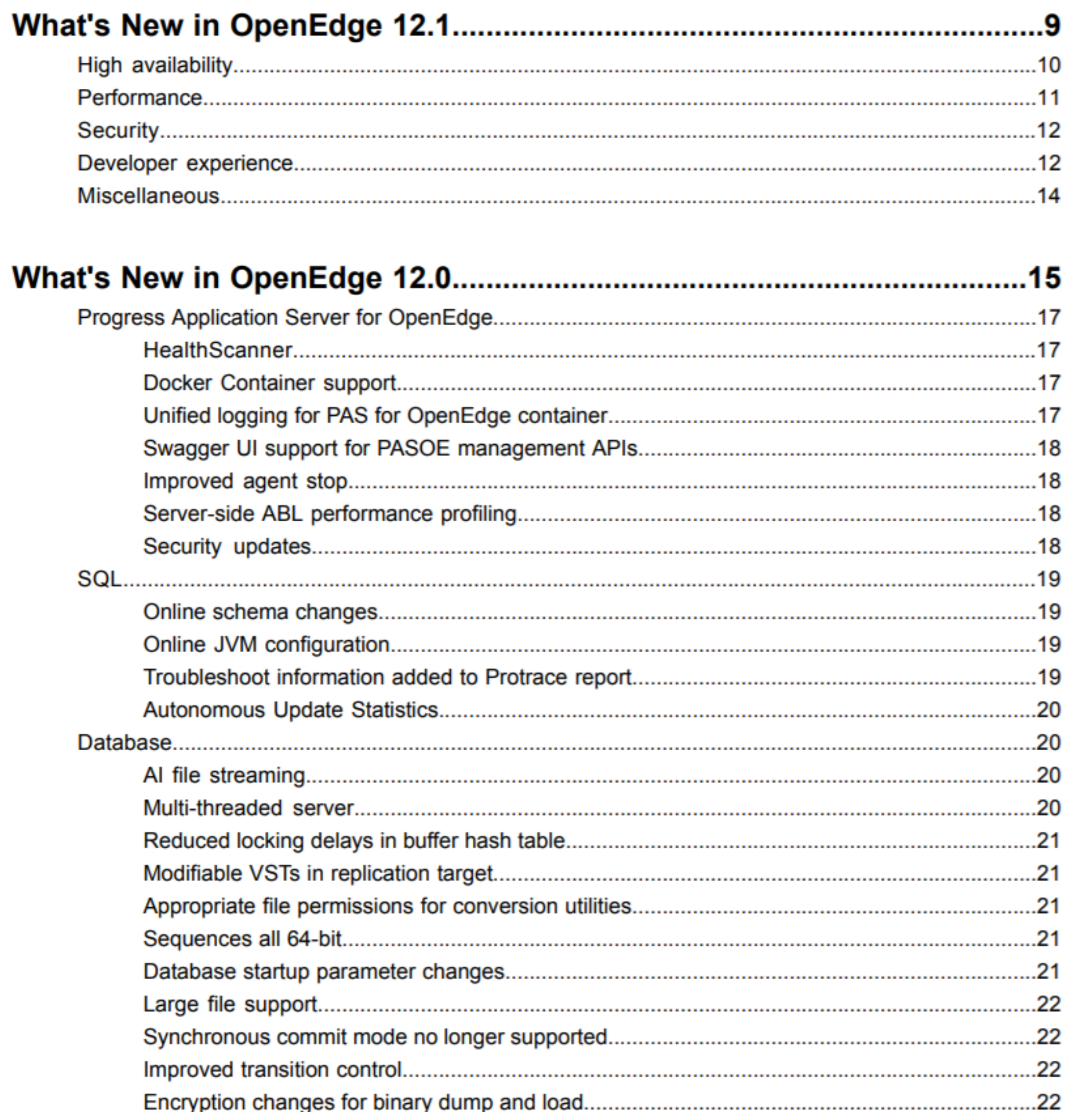
- Early Adopter Success Team for 12.x and PAS for OpenEdge
  - Migration from Classic AppServer
  - WebSpeed migration
  - Security
  - Performance Tuning
  - Diagnostics
  - CI/CD Pipeline

Progress

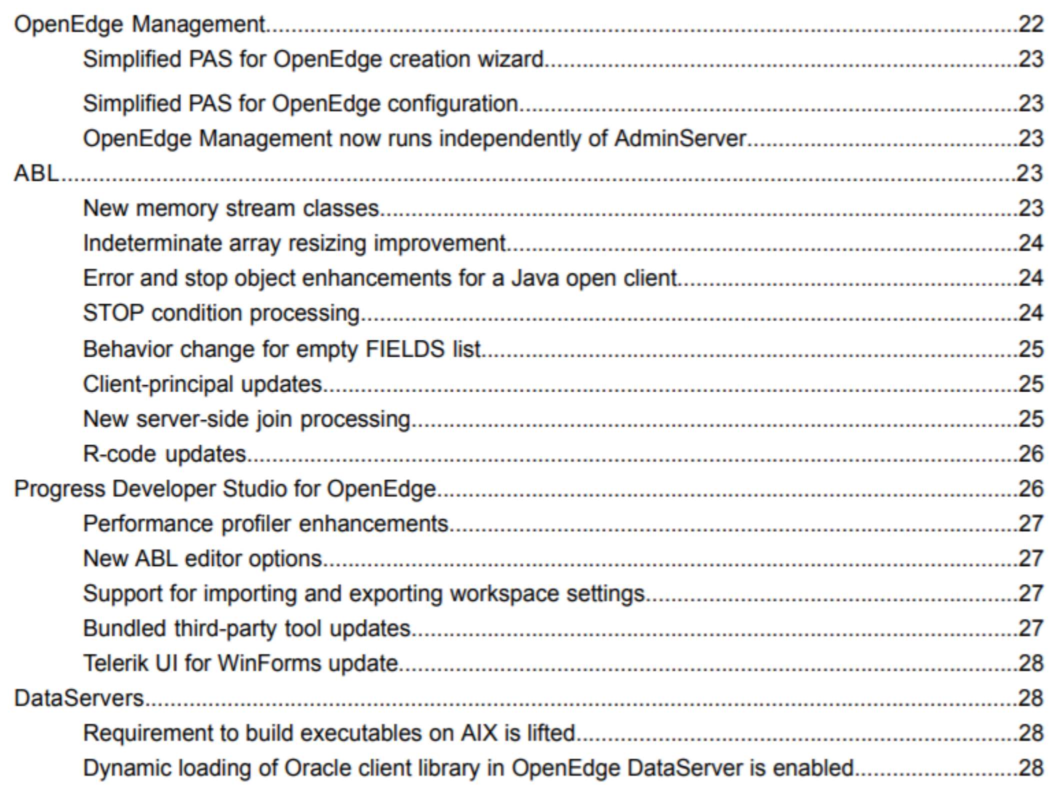
Progress®

OpenEdge 12.1 and 12.0

https://docs.progress.com/bundle/oe-pdfs-121/resource/openedge-whats-new.pdf



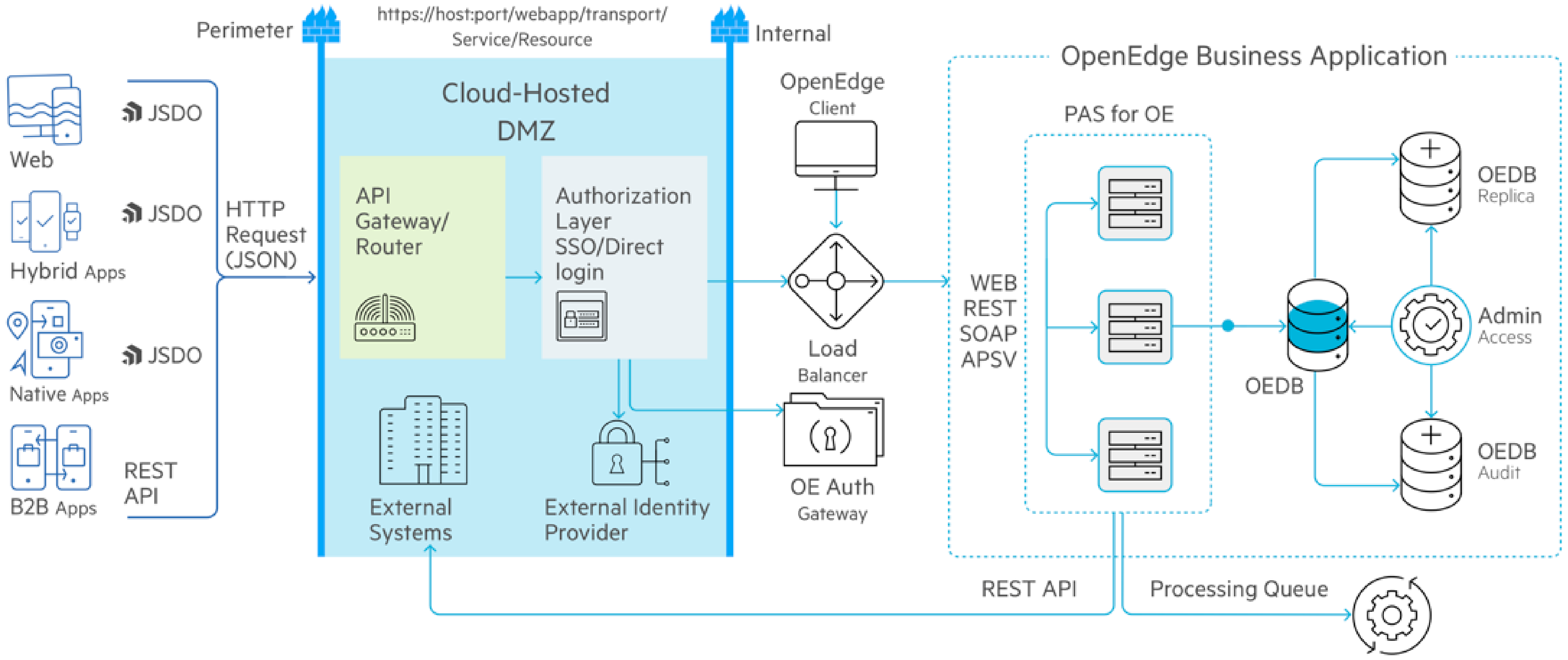
OpenEdge Platform



Target Architecture

OpenEdge Platform

Cloud-first Deployment Architecture Hosted

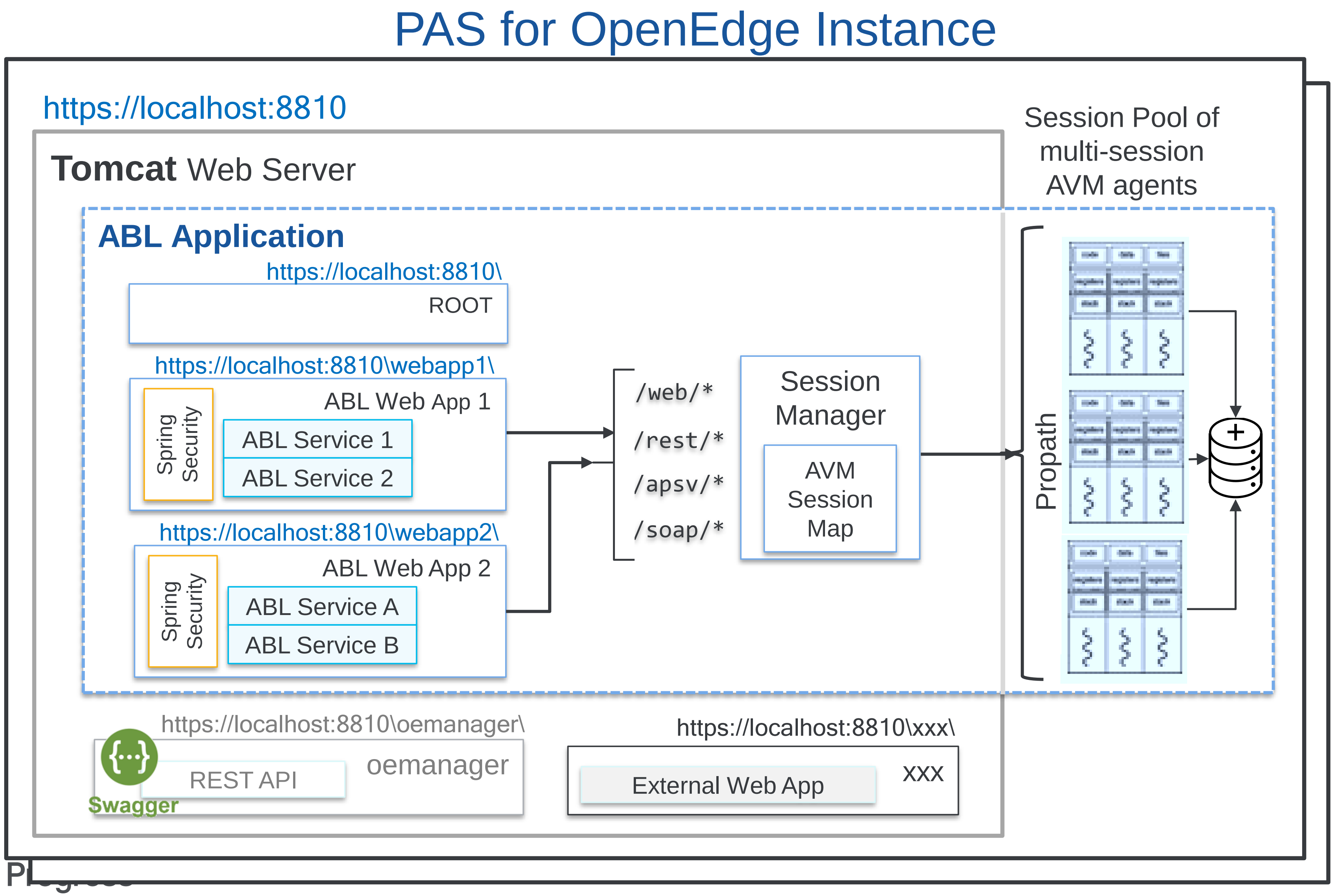


Progress Application Server for OpenEdge (PAS for OE)

OpenEdge Platform

- Cloud-ready, available as a Docker container for Linux (12.0)
- 12.1 Enhancements
  - Deferred logging *(troubleshooting)*
  - Get active requests API *(monitoring)*
  - Refresh agents *(high availability)*
  - Latest version of OpenSSL and Spring security

PAS for OE Architecture Recommendation



- By convention one tomcat Web Server holds one ABL Application (host/port)
  - Tomcat can host non-OE Web Apps
- Explicitly name Web Apps if more than one is needed for:
  - Security
  - Modularization
  - Monitoring traffic for billing, throughput, etc.

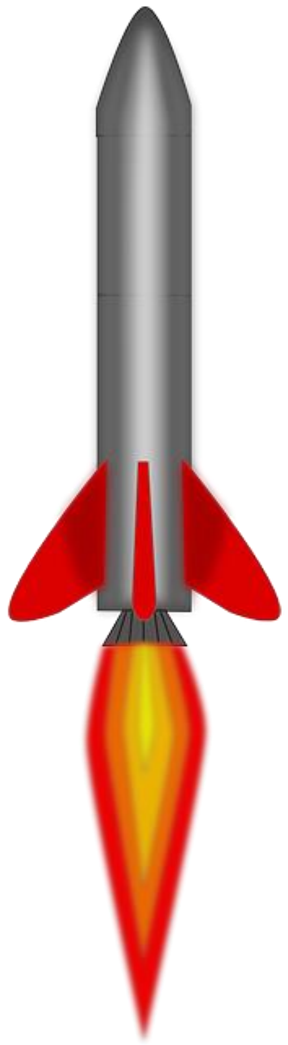
Enhanced OpenEdge RDBMS

OpenEdge Platform

- High availability with increased online operations
- Improved performance with new buffer pool hash table (BHT) latching
- Fault-tolerance with enhanced Replication
- Enhanced security cyphers for encryption
- New in 12.1
  - Param defaults changed to increase performance
    `-aibufs, -bibufs, -lruskips, -lru2skips, -pica, -prefetchDelay, -prefetchFactor, -prefetchNumRecs, -prefetchPriority, -Mm, -Mxs`
  - Modify database startup parameters online (added 40 new ones)
  - Allow non-structural schema changes online (field format, help string, label, etc.)
  - Server-side joins for dynamic queries forward-only, no-lock (FOR EACH was added in 12.0)
  - Replication: properties validation utility and Enhanced Replication Status in VSTs
  - Sequences increased to 32K
  - Extend and mark variable-length extents as fixed

OE RDBMS Best Practices

- Move from shared memory to Client-Server
  - Significant performance improvements make this more feasible
  - Multi-threaded database server
  - Kill of remote client can’t crash a database
  - Server-side joins
- Enable Replication
  - AI blocks transmitted to the targets as they are generated
  - Hot standbys (future: automatic db connection on failure)
- Use Pro2 for read-only access for reporting, etc.
  - Keep production DB at peak performance



Lessons Learned for WebSpeed Migration

WebSpeed Migration

- Scenario
  - WebSpeed Application (CGI and AJAX)
  - REST APIs provided by WebSpeed app
  - _users table
- Goal
  - Migrate customers from WebSpeed 10.2B to 11.7.x
  - Move all applications to 11.7.x PAS for OE and Spring security

Lessons Learned

- Supported WebSpeed WebObjects
  - Embedded SpeedScript
  - CGI Wrapper
- OpenEdge.Web.CompatibilityHandler
- WebHandler Support
- Conversion needed for:
  - HTML Mapped WebObjects
  - Customized web-disp.p

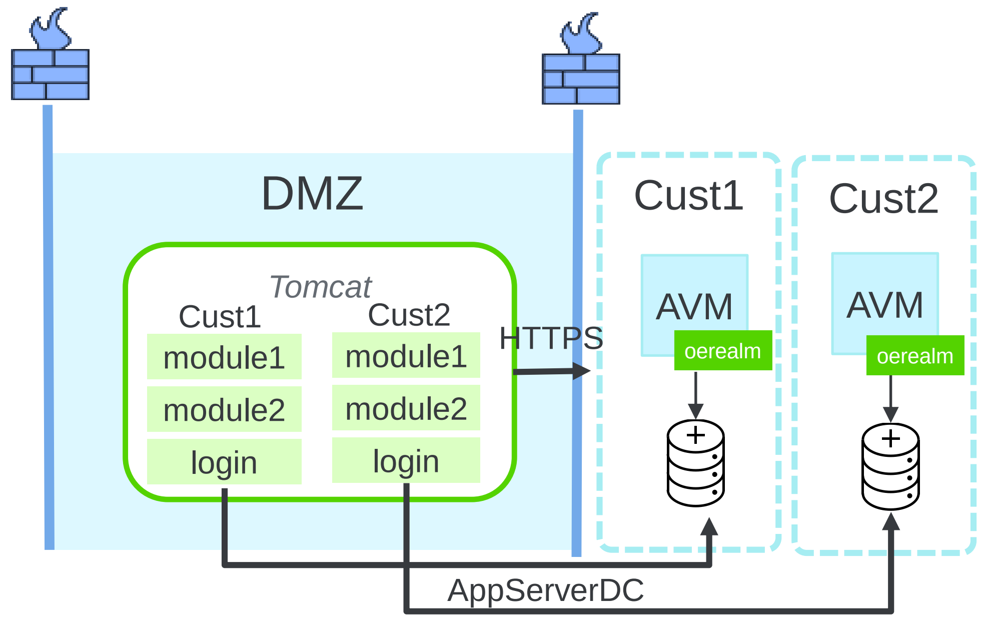
Progress®

Lessons Learned Migrating Classic AppServer

Migration to PAS for OE: Customer A

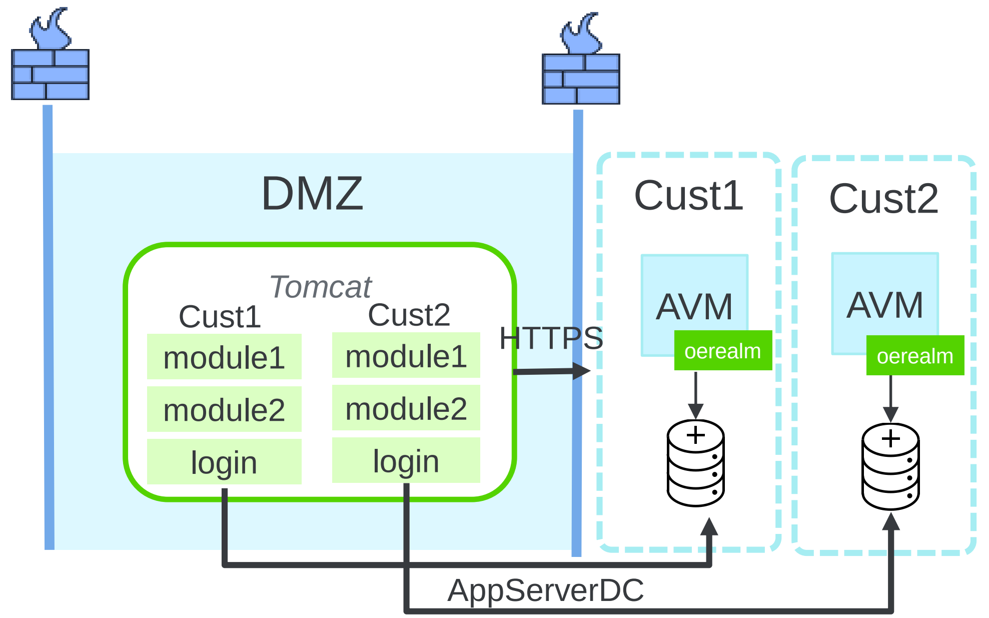
- Scenario
  - Large partner with a cloud-based product
  - Application hosted on dedicated AppServer and DB servers
    - A few high volume customers have their own on premise solution
  - OpenEdge 11.7 Classic AppServer
  - Using shared memory connections
- Goal
  - Migrate to PAS for OpenEdge on 11.7 first phase; limited re-architecture
  - Migrate to OpenEdge 12 second phase
  - Improve security and performance of application

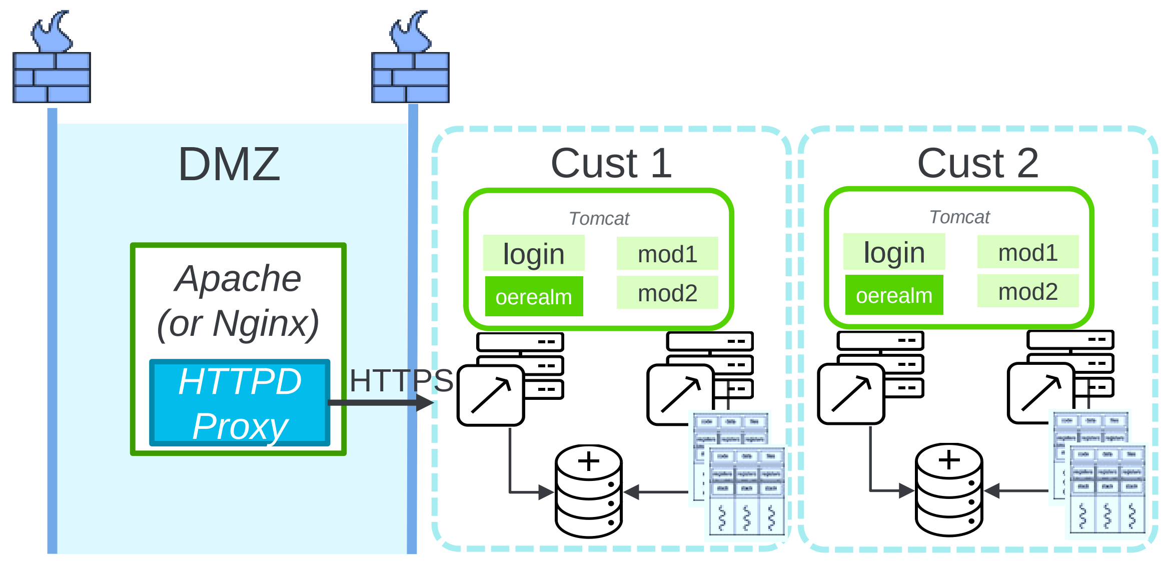
Architecture: Classic and PAS for OE (*high-level*)



- One Tomcat instance in DMZ shared by all customers
- Authentication done from DMZ using direct connect
- APIs defined in DMZ, requests use HTTP
- Business logic and DB behind firewall

Architecture: Classic and PAS for OE (*high-level*)





- One Tomcat instance in DMZ shared by all customers
- Authentication done from DMZ using direct connect
- APIs defined in DMZ, requests use HTTP
- Business logic and DB behind firewall

- One Tomcat instance for each customer behind firewall
- Requests routed by HTTPD (HTTPS)
  - Scalable thought load balancing
  - Version updates can be on-line
- Authentication, APIs are resolved behind the firewall

Progress

Lessons Learned

- Customer wanted multiple tenants to share PAS for OE instances due to concerns with Java/Tomcat resource consumption
  - A shared instance affected the SLAs for each customer
  - Resource usage not as bad as originally thought
- Deployment topology depend on many factors:
  - Prototypes often necessary to determine best architecture
  - Evaluate trade-offs in SLA, resources, and performance
- Identify evolution path for modernization
  - Identify milestones along the way and the work needed to get there

“Tests of classic state-free and PAS for OE

showed 1 classic broker and agent

used <u>only slightly less memory</u>

than 1 PAS for OE instance and 1 multi-session agent.”

- Customer X

Migration to PAS for OE: Example 2

- Large partner
- Cloud-based product deployed on 11.7.x
- Existing mature (self-built) build and deploy toolchain
- Existing mature customer support environment

Goals

- Evaluate moving to OE 12
- Evaluate moving from Classic AppServer to PAS for OE
- Changes to database connections initially **not** considered (ie SSJ)

Script Migration Steps

Existing system:

1. Run `paspropconv` to export Classic AppServer configs

New system:

2. Create an instance using `pasman create`
3. Merge exported properties using `oeprop -f`
4. Optionally add any other configurations using `pasman deploy` and `oeprop -f`

Export Classic AppServer configurations

```
paspropconv \
--ubrokerPropsFile     /path/to/ubroker.properties \
--ubrokerName          UBroker.AS.as-app1 \
# this becomes the abl-app name in the instance
--pasoeAppName         as-app1-pasoe \
# in this case, must match the abl-app name
-–pasoeWebAppName      as-app1-pasoe
```

https://docs.progress.com/bundle/qs-move-to-pasoe/page/About-this-guide.html

Setup PAS for OE Instances

```
# 1. Create a new PASOE instance
pasman create -f <ports> /path/to/instance/as-app1-pasoe

# 2. Merge     properties for the ROOT webapp
pasman oeprop -I as-app1-pasoe -f /path/to/as-app1-pasoe.merge

# For each of the Classic AppServers to merge into this PASOE Instance …
# 3. create the ABL in the instance (prop files and on disk)
pasman deploy -I as-app1-pasoe  \
              -a <abl-app-name> \   # In this case webapp name is ABL-App name
              $DLC/servers/pasoe/extras/oeabl.war <abl-app-name>

# 4. merge the exported properties
pasman oeprop -I as-app1-pasoe \
              -f /path/to/<abl-app-name>.<broker-name>.oemerge
```

Monitoring & Troubleshooting

Lots of moving parts

- 3 kinds of processes: Tomcat, MS Agent, DB server
- 3 kinds of sessions: HTTP, Spring, AVM/ABL
- Almost everything has its own log

Supportability

- ✓ Knowledge
- ✓ Experience

Monitoring

- ✓ More & different data (depth-of)
- ✓ More logs & different formats

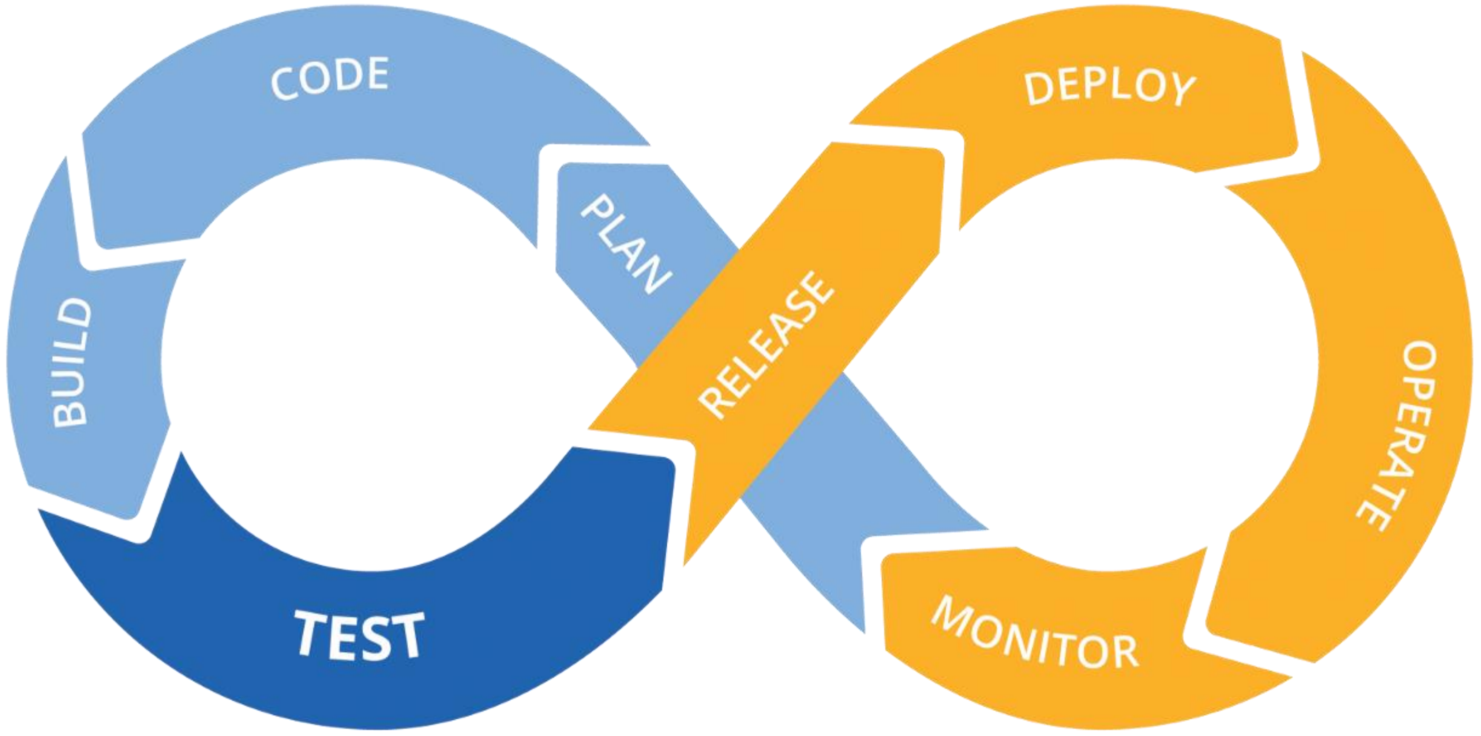
Lessons Learned

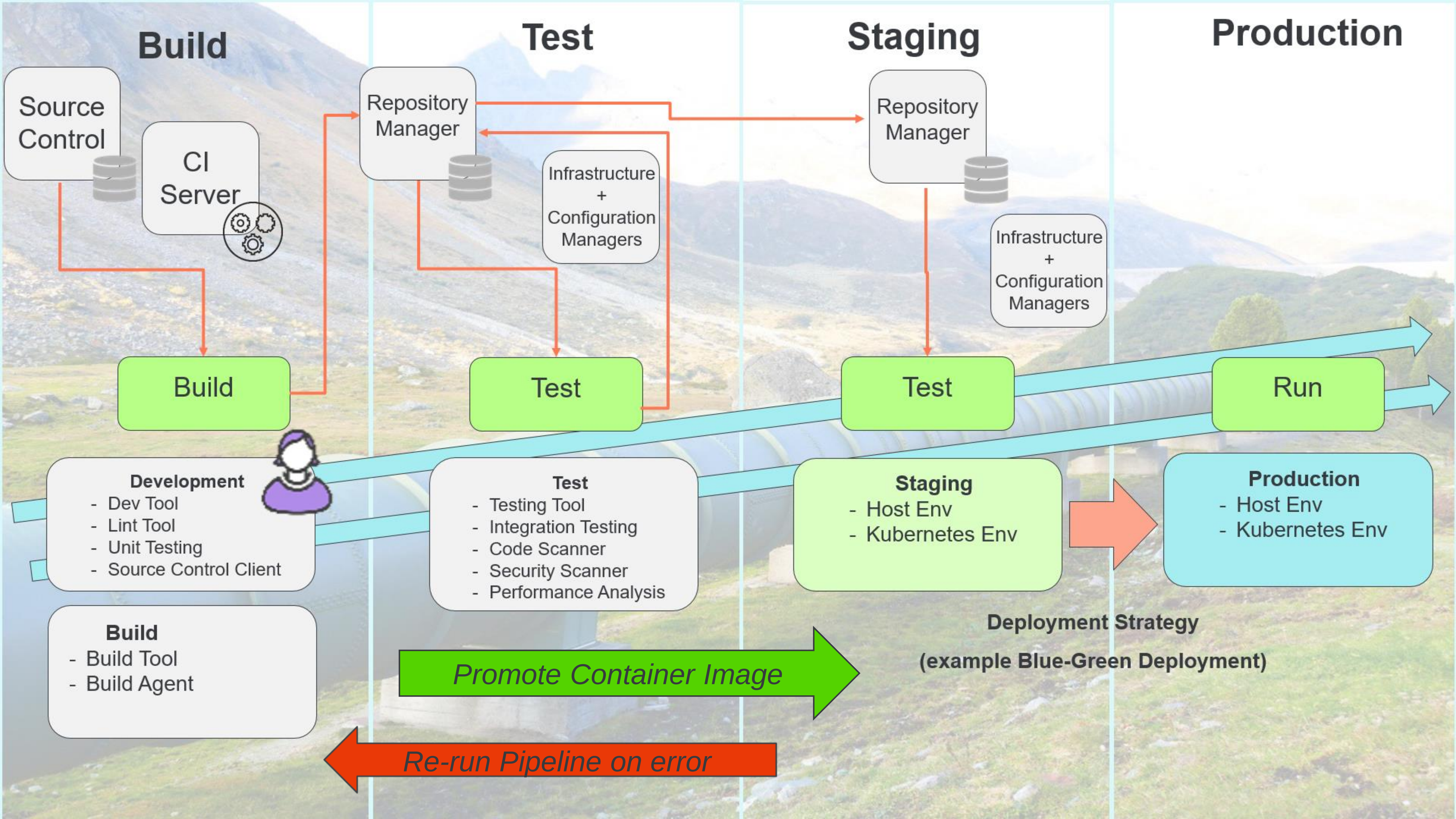
- Moving one piece is reasonably easy …
  - How to do this at scale across the org?
  - What changes are needed to the toolchain, for build, deploy, monitor?
- Keys to Monitoring
  - Tracing the path of a single request is key to know where and what can break
  - Log file formats differ, so scrapers/uploaders need to change
- Keys to Supportability
  - Training on the technology reasonably simple
  - Gaining experience in running applications on PAS for OE takes time

Progress®

Lessons on Building CI/CD Pipeline

Continuous Development Cycle




Build
Test
Staging
Production
Source Control
CI Server
Repository Manager
Infrastructure + Configuration Managers
Repository Manager
Infrastructure + Configuration Managers
Build
Test
Test
Run
Development
- Dev Tool
- Lint Tool
- Unit Testing
- Source Control Client
Test
- Testing Tool
- Integration Testing
- Code Scanner
- Security Scanner
- Performance Analysis
Staging
- Host Env
- Kubernetes Env
Production
- Host Env
- Kubernetes Env
Build
- Build Tool
- Build Agent
Promote Container Image
Deployment Strategy
(example Blue-Green Deployment)
Re-run Pipeline on error

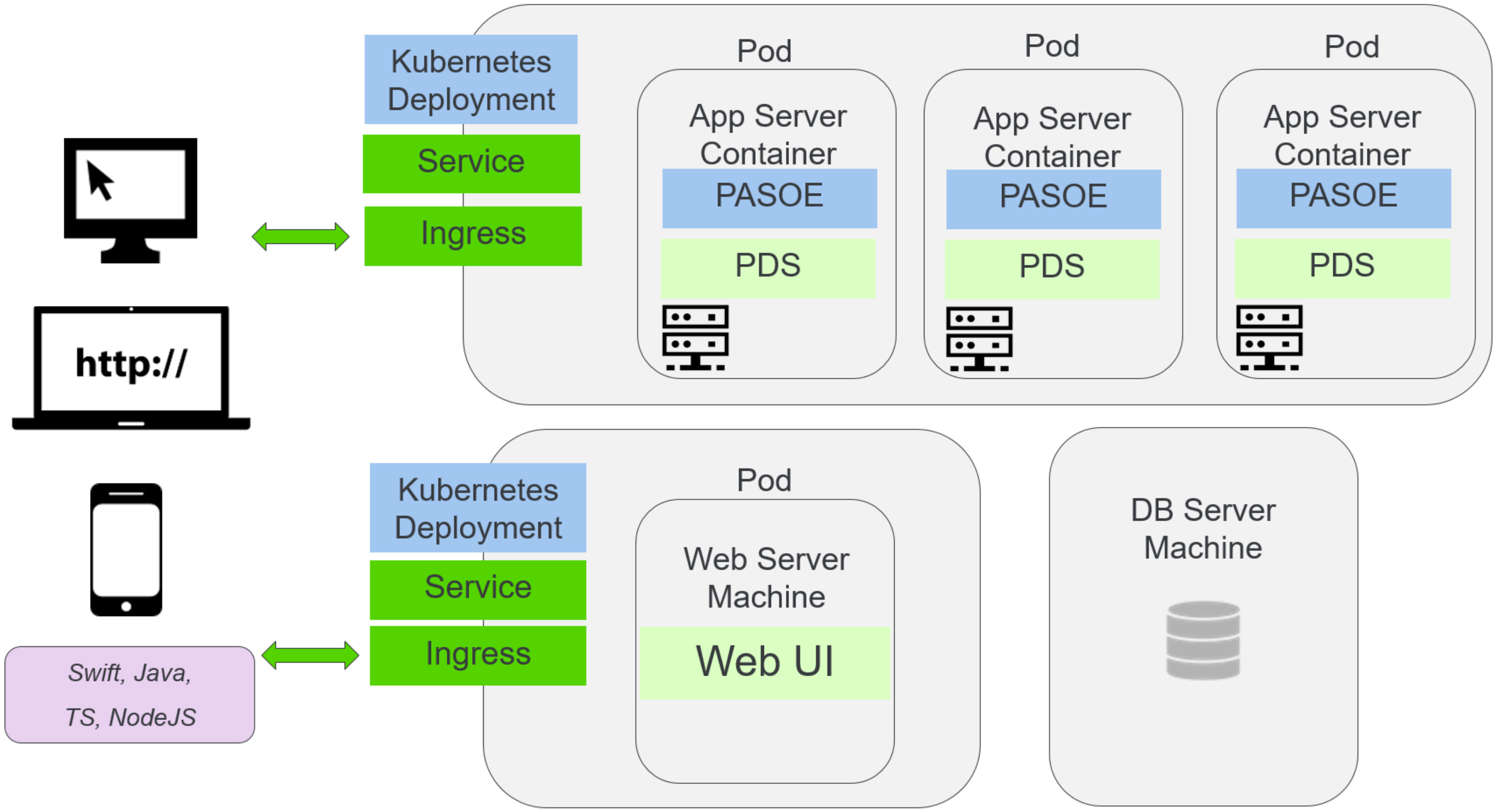
Getting Started

- Build a basic automated pipeline then iterate
  - Codify / Automate and use Infrastructure as Code principles
  - Code Coverage and Quality is key
- Focus on areas of the pipeline based on organization requirements
- A Maturity Model can help to understand the state of CI/CD:
  - http://bekkopen.github.io/maturity-model/
  - https://dzone.com/articles/continuous-delivery-the-holy-grail-of-cloud-app-ma

Lessons Learned

- Automate, automate, automate
- Code Quality
- Build pipeline and iterate
- Use maturity model
- Promote image
- Blue-Green Deployment for High Availability

Using Containers



Lessons Learned

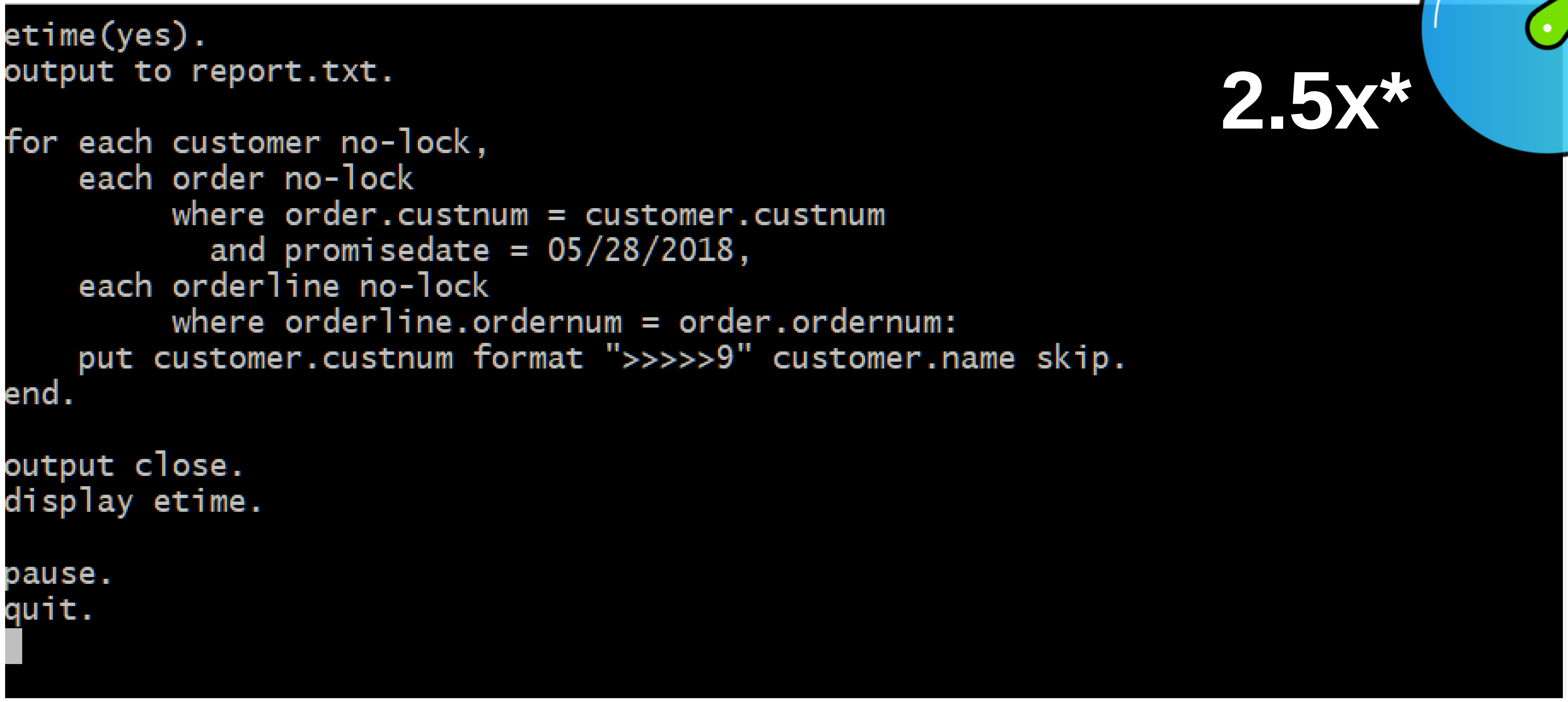
- Containers encapsulate application
- Using containers simplifies deployment
- Repeatable results
- Cluster Support
- Scalability
- High Availability

Lessons Learned on Performance

Server Side Joins (SSJ)

- Enabled by Default
- Client/Server
- FOR Statement
- FORWARD-ONLY Dynamic Query
- NO-LOCK
- Same Logical DB
- Up to 10 Tables

Server Side Joins



```
etime(yes).
output to report.txt.

for each customer no-lock,
    each order no-lock
         where order.custnum = customer.custnum
           and promisedate = 05/28/2018,
    each orderline no-lock
         where orderline.ordernum = order.ordernum:
    put customer.custnum format ">>>>>9" customer.name skip.
end.

output close.
display etime.

pause.
quit.
```

2.5x*

Lessons Learned

- Enable by default
- Reduction of data sent over the network
- FOR statement
- FORWARD-ONLY Dynamic Query
- QryInfo Logging

Progress®

Next Steps

Recommended Next Steps: Getting Started

OpenEdge Platform

- Move to OpenEdge 12
- Inventory of components and migration plan to PAS for OE
- Definition of Success

Infor Rapidly Transforms New App Ideas into Product Features

Mark Information

KRAS

Recommended Next Steps

OpenEdge Platform

- PAS for OpenEdge
  - Use separate PAS for OE instances based on SLA requirements
- REST Services
  - Recommend Web transport
- WebSpeed
  - Direct Migration using the Compatibility WebHandler
    - CGI Wrapper
    - HTML with embedded SpeedScript

Recommended Next Steps

OpenEdge Platform

- CI/CD
  - Build WAR file for Web Application from Command Line
  - Docker, Kubernetes
  - Blue-Green Deployment
- Performance
  - Write queries using multi-table joins to use Server Side Joins

Progress

Thank You.

Progress®