Power your journey to AI with IBM Cloud Pak for Data DataStage

Tech-talk: Tech Talk: DataStage on Cloud Pak for Data - Unlimited scaling for your workloads with a reduced total cost of ownership

Scott Brokaw
Offering Management - Data Integration
slbrokaw@us.ibm.com

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.

DataStage Modernization

Value in modernizing with Cloud Pak for Data

ONE
Reduced infrastructure management effort of 65% to 85%*

- Cloud-ready Data Integration Architecture for AI built on containers and microservices
- DataOps ready through out-of-the-box integration with governance, BI, data virtualization and data science

TWO
Save up to 30% of workload execution time

- Performance gains in heavy workload and resource contention situations
- Design once, run anywhere at extreme scale

THREE
Reduce cost of operations by up to 50%

- Meet mission critical SLAs through automatic failure resolution
- Leverage existing DataStage investments in skills and assets – no costly retraining required

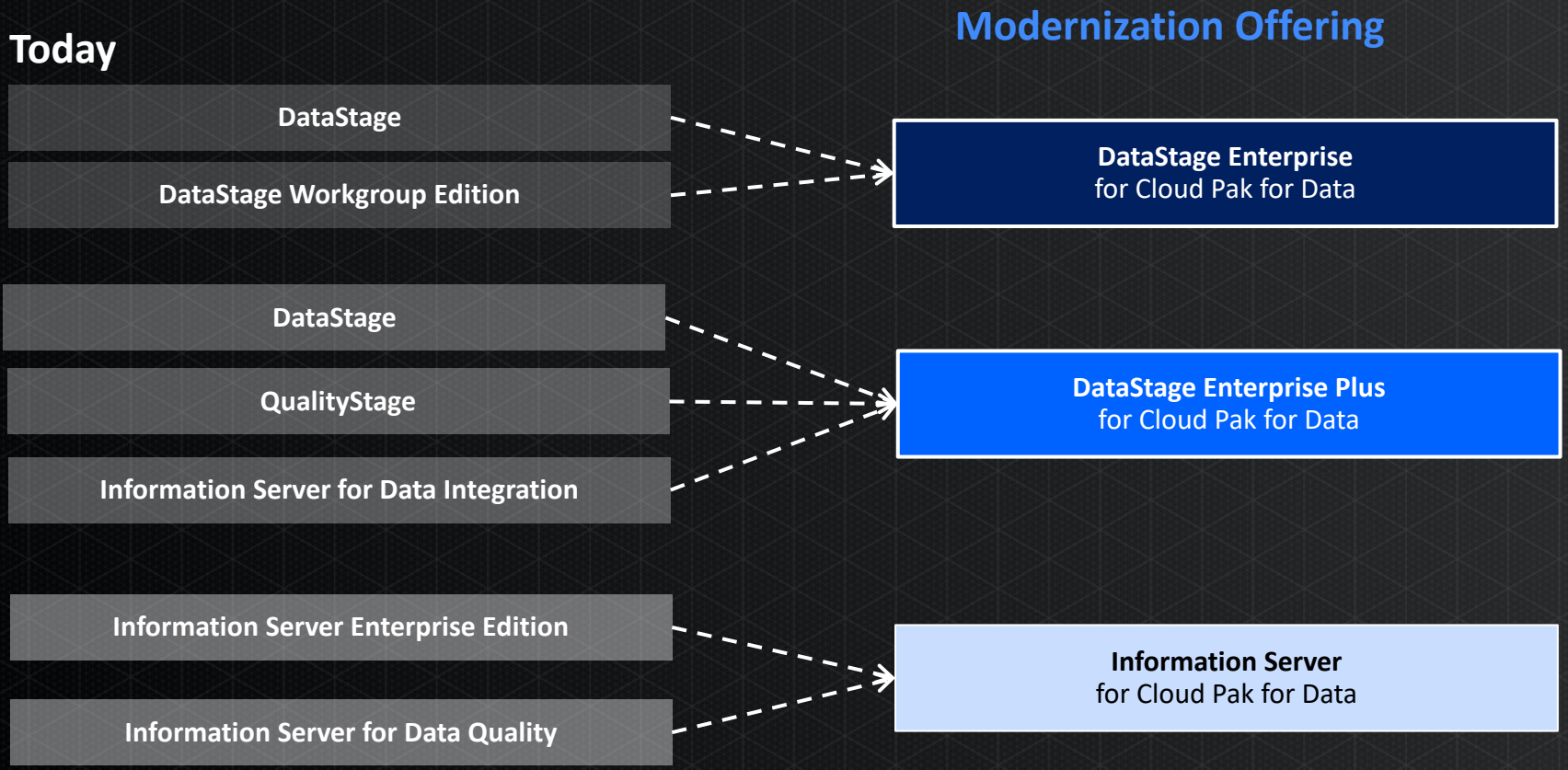
FOUR
Remove network bottlenecks with co-located Netezza or Db2 Warehouse on Cloud Pak for Data System

"One of the great things about the Cloud Pak for Data System is the speed with which we'll be able to launch and scale our analytics platform. The integrated stack contains what we need to improve data quality, catalog our data assets, enable data collaboration, and build/operationalize data sciences. **We're able to move quickly with design, test, build and deployment of new models and analytical applications**."

Steve Lueck
Vice President, Data Management
Associated Bank

*TEI report: https://www.ibm.com/downloads/cas/V5GNQKGE "Reduced infrastructure management effort: 65% to 85%" link

Paths to Modernize



How entitlements traded up to Modernization Upgrade can be allocated

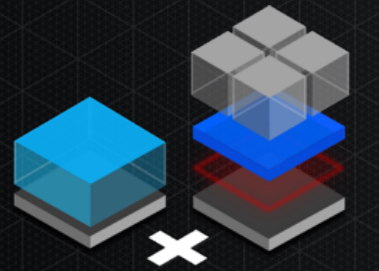
Scenario: Your existing DataStage

- Today: 7000 PVUs of DataStage Standalone (Prod)
- At renewal: trade-up to DataStage Enterprise Upgrade
- Get at a minimum: 100 VPCs of DataStage Enterprise + 100VPCs of Cloud Pak for Data

• Once traded-up, you can allocate this entitlement in an infinite number of ways, some shown below:

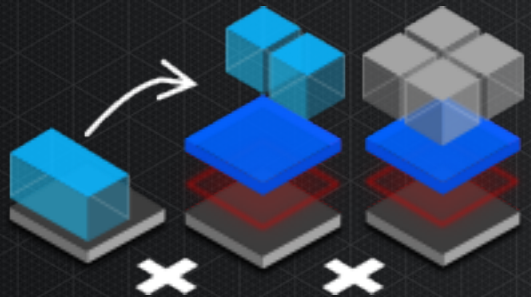
Example

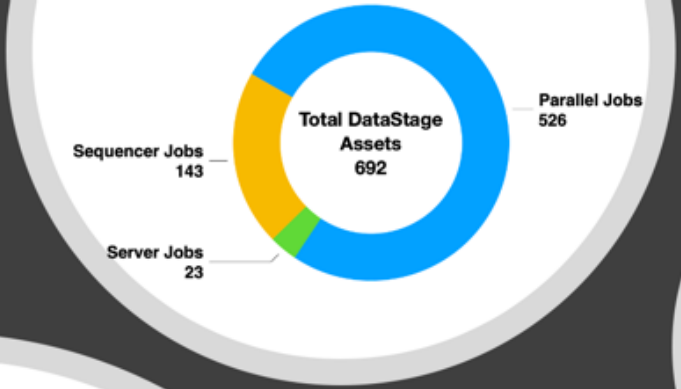
Trade-up license entitlement but workload still runs on stand-alone offering

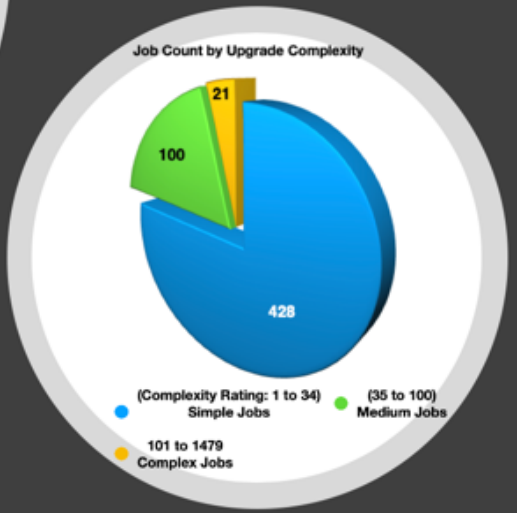


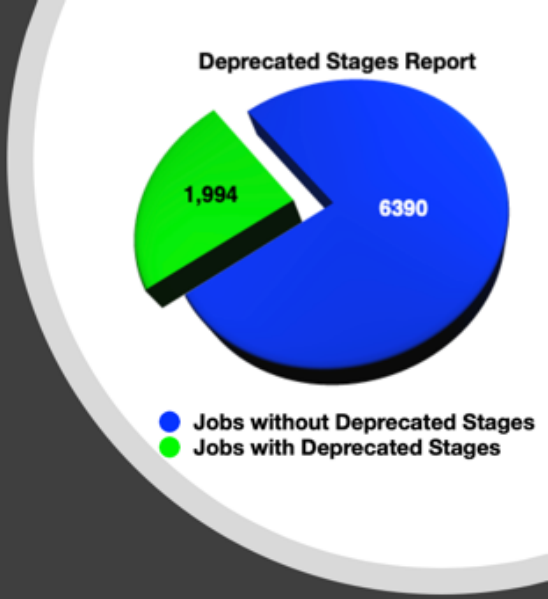
Example

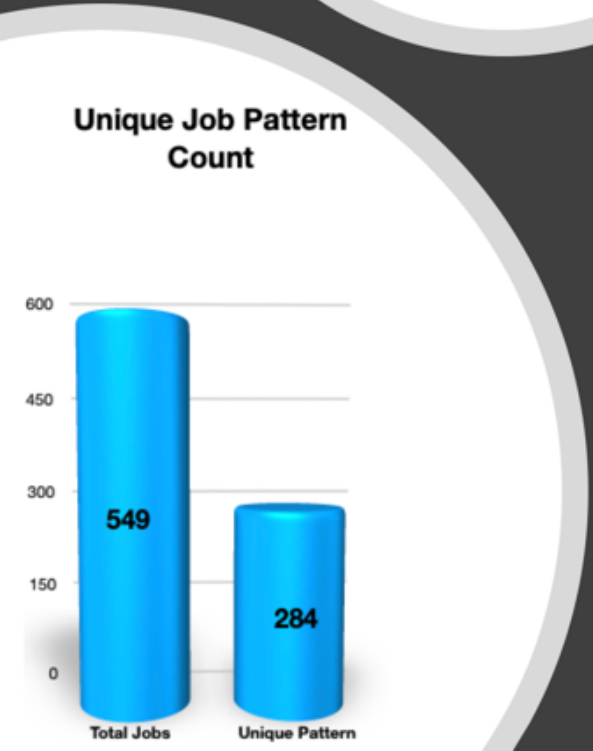
Trade-up license entitlement and move workload to extension gradually







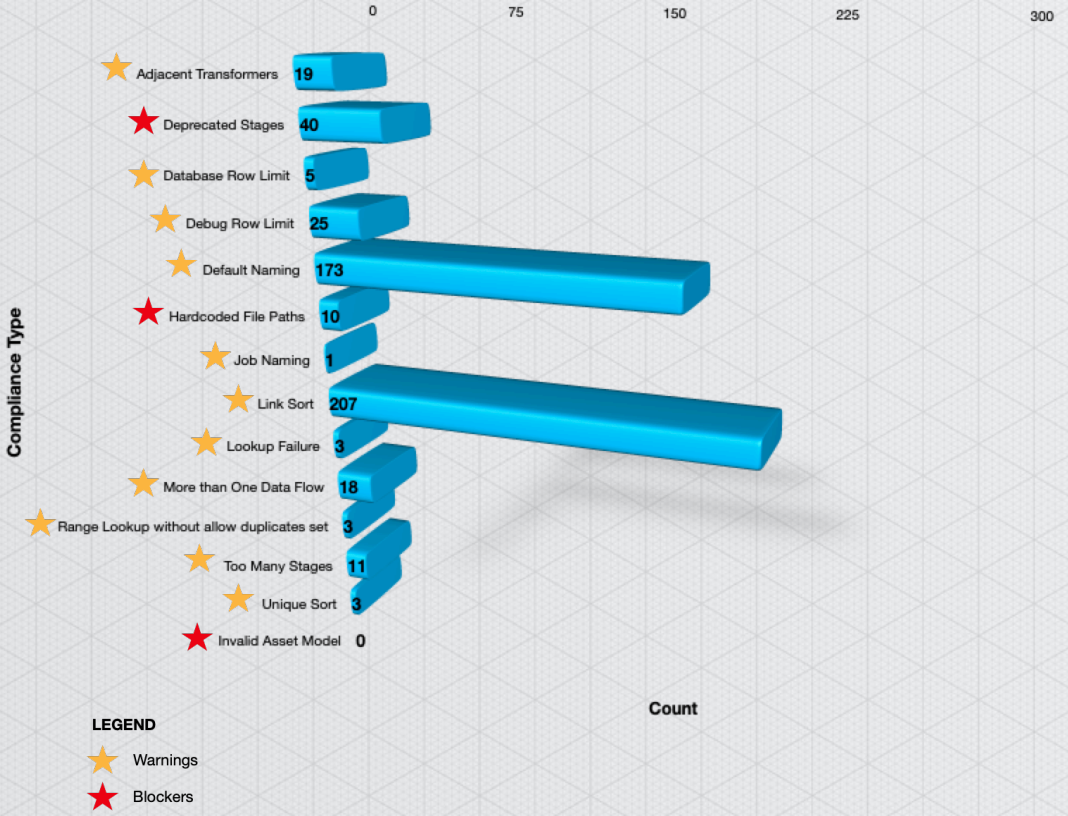




DataStage Upgrade Assessment Report Card

IBM

DataStage Compliance Report Summary



Cloud Pak for Data

1. Services Ecosystem

With a click, access and deploy an ecosystem of 45+ analytics services and templates from IBM and third parties.

2. Platform Interface

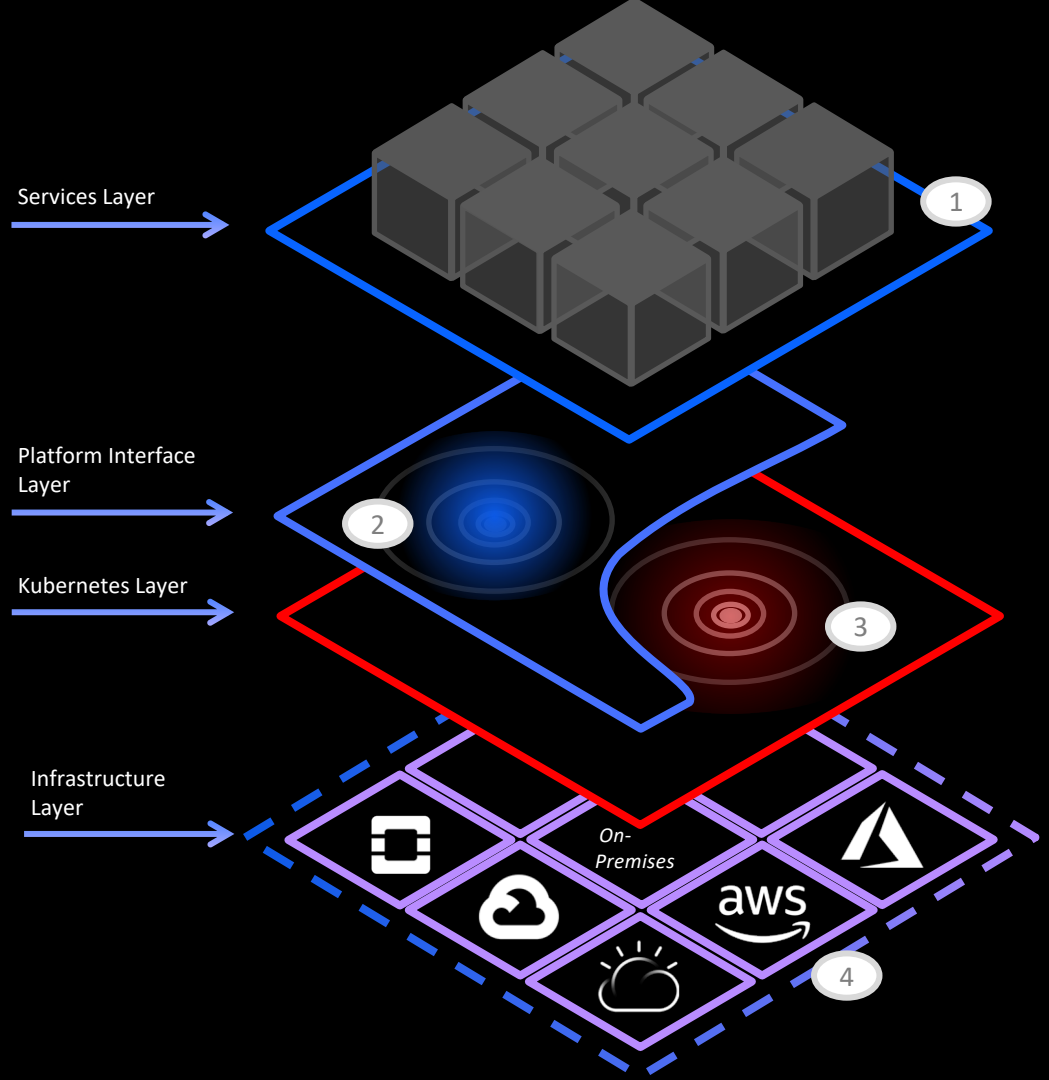
Speed time-to-value with a single user experience that integrates data management, data governance and
analysis for greater efficiency and improved use of resources.

3. Red Hat **OPEN**SHIFT®

Leverage the leading hybrid cloud, enterprise container platform for an innovative and fast deployment strategy

4. Any Cloud

Avoid lock-in and leverage all cloud infrastructures with our multi-cloud approach.



Cloud Pak for Data DataStage

Multi-cloud scalability and elasticity

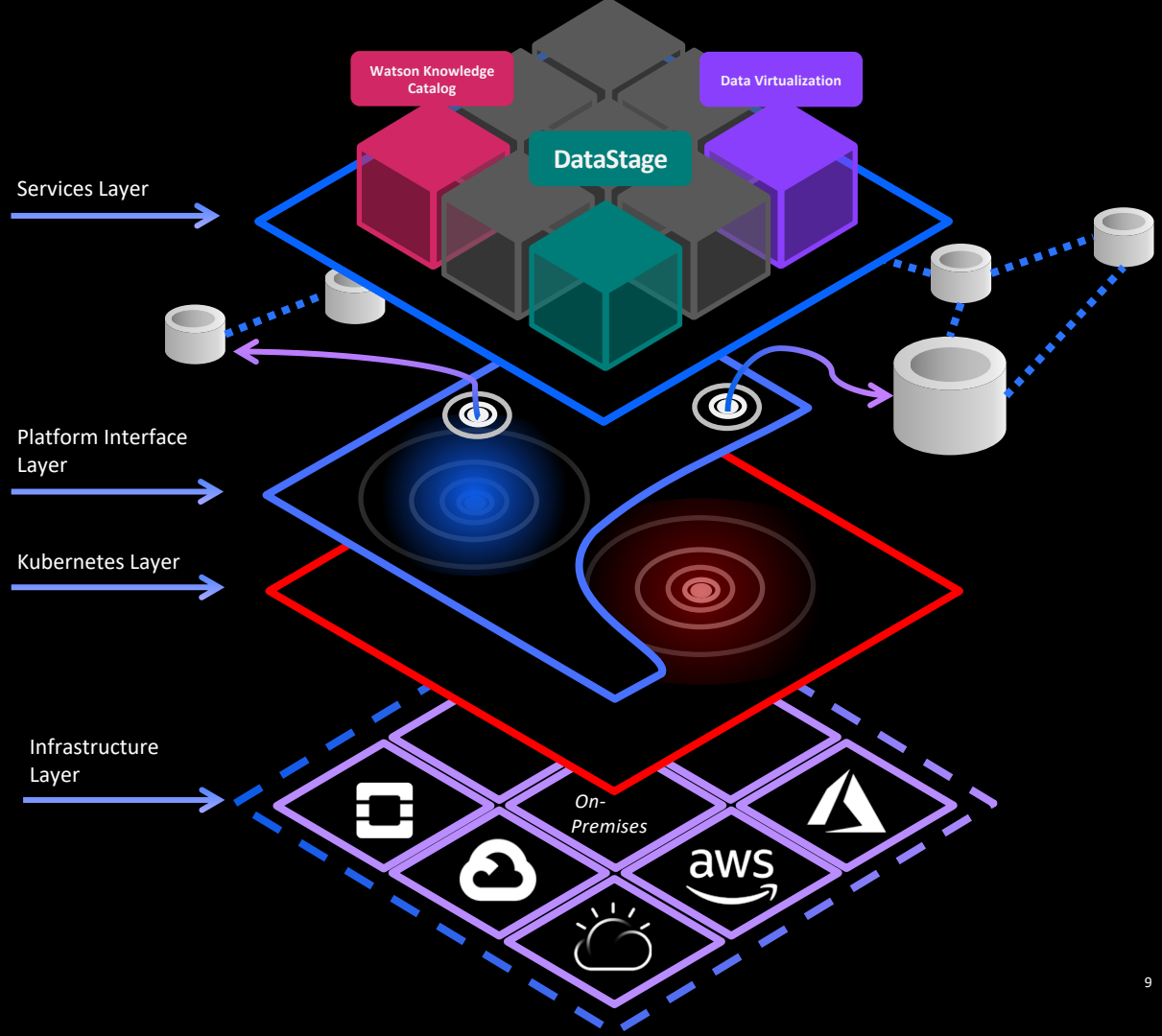
- Design once, dynamically run anywhere with built-in automatic workload balancing, parallelism and dynamic scalability

DataOps and DevOps enabled

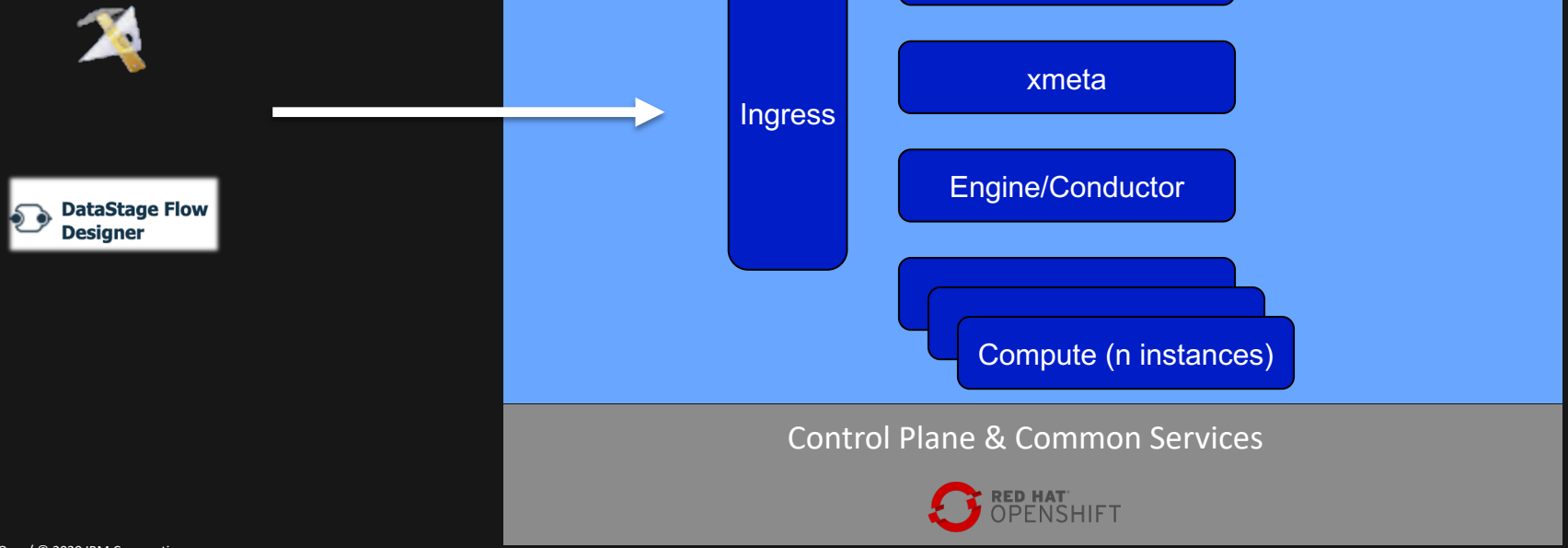
- Built-in resiliency, easy operation and CI/CD

Accelerate AI initiatives

- Automating Data Integration for faster ROI



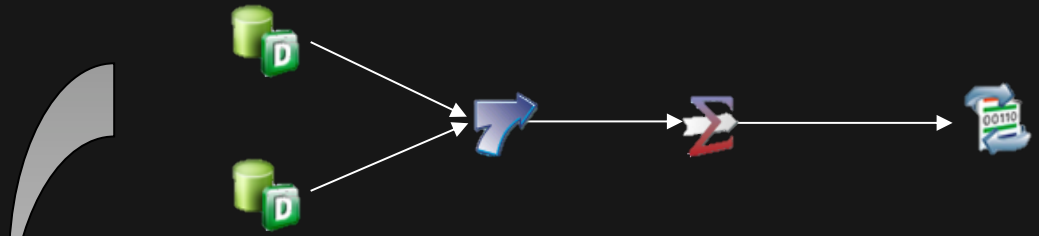
Cloud Pak for Data DataStage



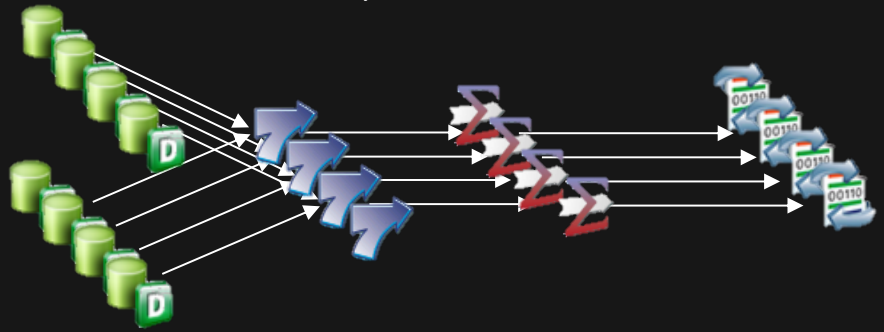
DataStage Parallel Engine

Job design versus execution

User assembles the flow using DataStage Designer



… at runtime, this job runs in parallel for any configuration
(1 node, 4 nodes, *N* nodes)



No need to modify or recompile the job design!

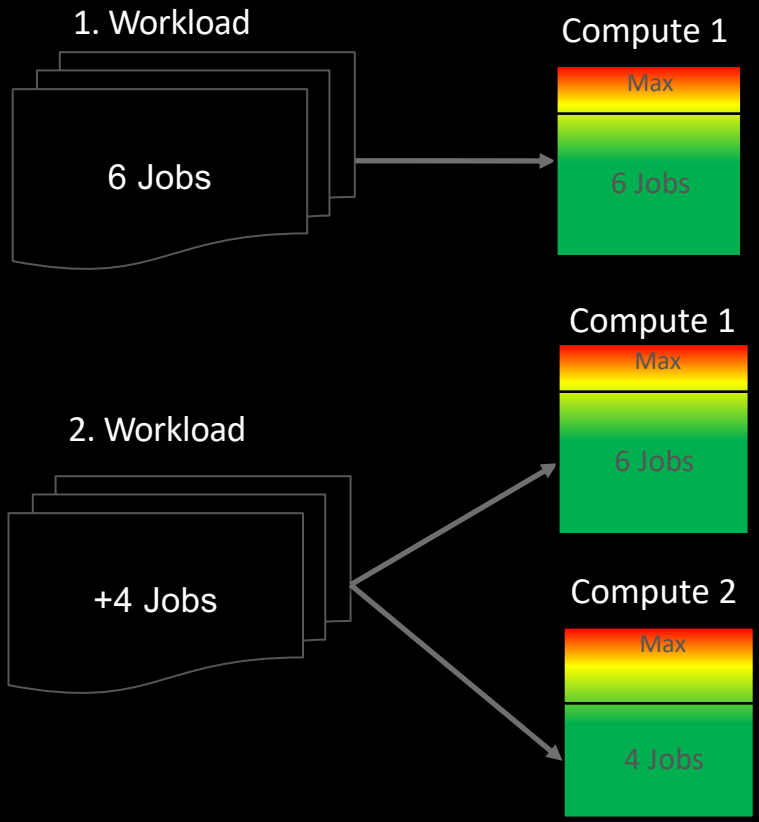
Built-in automatic workload balancing and best of breed parallel engine

Unlimited scaling (horizontal, vertical) using PX engine

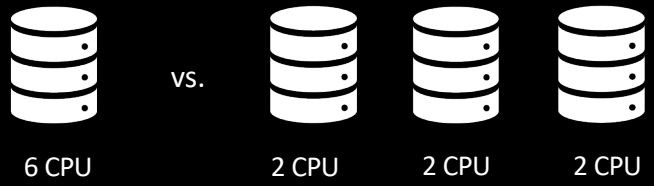
Automatic load balancing to maximize throughput and minimize resource congestion

Supports to run resource intensive workloads in parallel pipelining

Built on container architecture to allow for handling of any data volume and execution on any environment



Performance of DataStage for Cloud Pak for Data

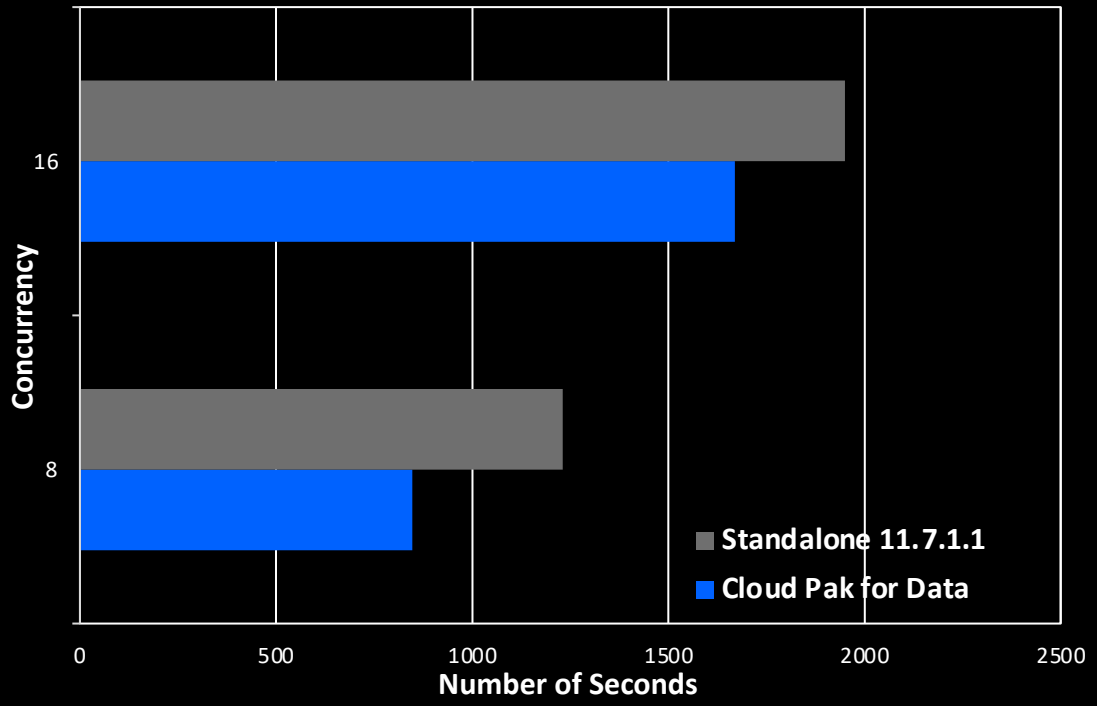


Objective:

- Validate performance during execution windows of resource contention
- Demonstrate value of default execution of Massively Parallel Processing (MPP)

Confirmed Result:

- Significant reduction in runtime on DataStage Cloud Pak for Data
- Delivers more evenly balanced and distributed workload



ds-engine-compute StatefulSet

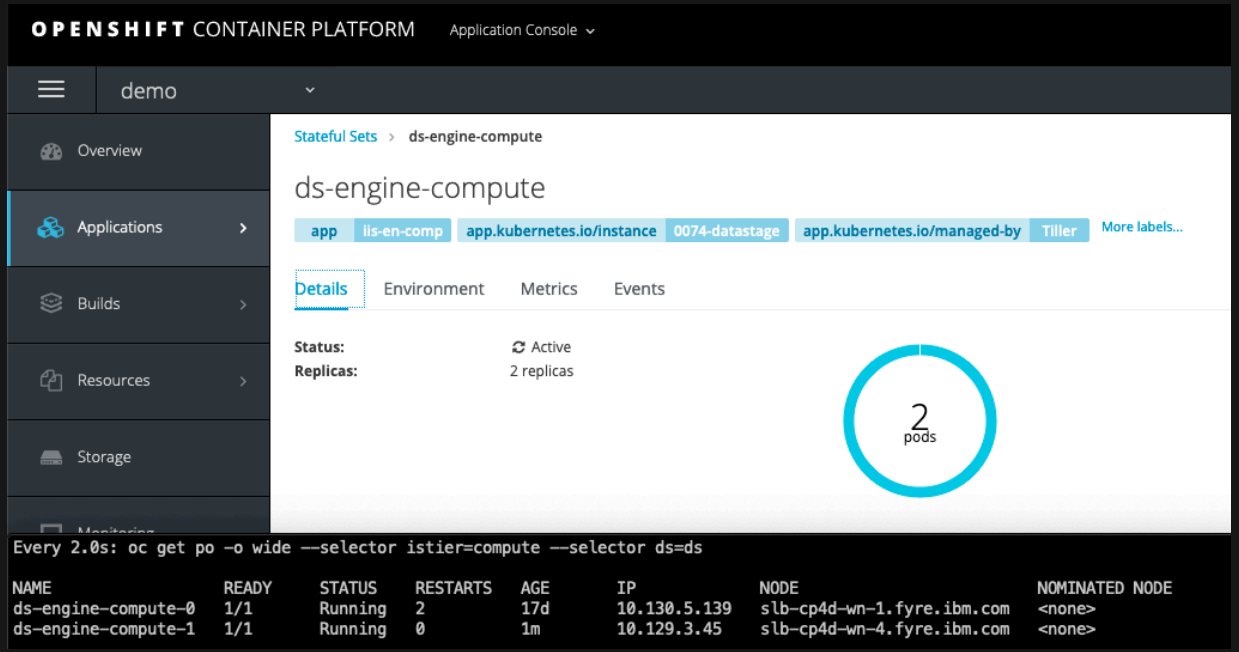
```
# oc get sts ds-engine-compute

NAME                DESIRED   CURRENT   AGE
ds-engine-compute       2         2     49d
```

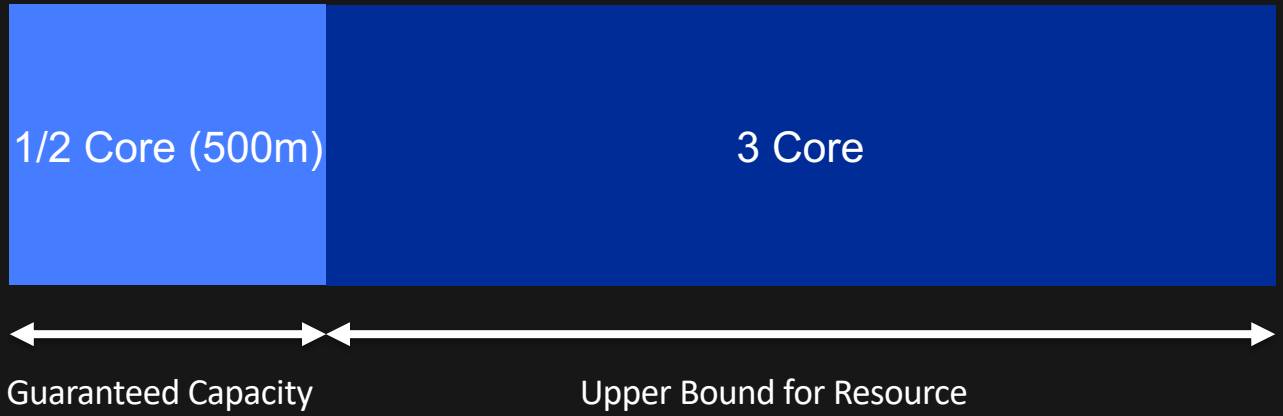
<u>Resource Limit/Requests:</u>

Limits:
- cpu: 3
- memory: 12Gi

Requests:
- cpu: 400m
- memory: 1500Mi



Resource Requests/Limits



Dynamic Workload-balancing
PXRuntime Project

DataStage Flow Designer

1

Persisted

xmeta

DataStage Flow Designer

{REST:API}

2

Compiled

PXRuntime Project

Engine/Conductor

Projects/dstage1/jobs/JobName

$APT_ORCHOME/bin/pxjob

{REST:API}

DataStage Flow Designer

3

Run

```
{
  node "node1"
  {
    fastname "is-en-conductor-0.en-cond"
    pools "conductor"
    ...
    ...
  }
  node "node2"
  {
    fastname "$pod"
    pools ""
    ...
    ...
    instances 2
  }
}
```

Engine/Conductor

Compute (n instances)

pxjob

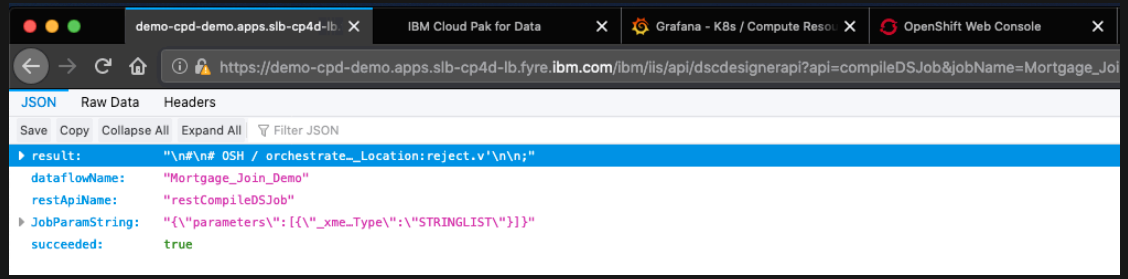
- Only supported CLI tool to run jobs in a PXRuntime Project
- $APT_ORCHHOME/bin/pxjob
- Syntax equivalent to dsjob
- Should be able to find/replace dsjob with pxjob

```
/opt/IBM/InformationServer/Server/PXEngine/bin/pxjob

usage: pxjob [-verbose] [-pxhost host] [-pxport port]
          [-domain domain_name -user username -password password -server enginename]
          [-authfile credentials_filename]
          [-file credentials_filename domain enginename]
          <command> [arguments]
<commands> include:
 -run [-mode <NORMAL | RESET | RESTART | VALIDATE>]
     [-queue <queue name>]
     [-paramfile <filename>]
     [-param <name>=<value>]
     [-warn <n>]
     [-wait]
     [-opmetadata <TRUE | FALSE>]
     [-jobstatus]
     [-sla <sla seconds>]
     <project name> <job name>
 -stop [-useid] <project name> <job name | job id>
 -lqueues
 -lprojects
 -ljobs [-status status_list] <project name>
 -linvocations [-useid] <project name> <job name | job id>
 -lstages [-useid] <project name> <job name | job id>
 -llinks [-useid] <project name> <job name | job id> <stage>
 -projectinfo <project name>
 -jobinfo [-useid] <project name> <job name | job id>
 -stageinfo [-useid] <project name> <job name |job id> <stage>
 -linkinfo [-useid] <project name> <job name> <stage> <link>
 -lparams [-useid] <project name> <job name | job id>
 -paraminfo [-useid] <project name> <job name | job id> <param>
 -log [-info | -warn] [-useid] <project name> <job name | job id>
-logsum [-type <INFO | WARNING | ERROR | FATAL | REJECT | STARTED | RESET | BATCH>] [-max num] [-useid] <project name> <job name|job id>
 -logdetail [-full] [-useid] <project name> <job name | job id> <first event id> [<last event id>]
 -logdetail [-full] [-useid] <project name> <job name | job id> [-wave <wave no>]
 -lognewest [-useid] <project name> <job name | job id> [<event type>]
    event type = INFO | WARNING | ERROR | FATAL | REJECT | STARTED | RESET | BATCH
 -report [-useid] <project> <job|jobid> [report type>]
  report type = BASIC | DETAIL | XML
 -purge [-useid] [-runs <n> | -days <n>] <project name> <job name | job id>
 -jobid <jobid> <project name> <job name>
 -schedulejob -time <MM:HH> -type <type name> -days <days list>
          [-queue <queue name>] [-warn <n>] [-param <name>=<value>] <project name> <job name>
 -rescheduleljob -time <MM:HH> -type <type name> -days <days list> -id <schedule id>
          [-queue <queue name>] [-warn <n>] [-param <name>=<value>] <project name> <job name>
 -scheduledelete <schedule id> <project name> <job name>
 -lschedules <project name> <job name>
```

REST API

- Compile Jobs!
- Run Jobs
- Status of Jobs
- Import Assets



[REST API Documentation]

Static APT_CONFIG_FILE

```
{
  node "node1"
  {
    fastname "is-en-conductor-0.en-cond"
    pools "conductor"
    resource disk "/opt/IBM/InformationServer/Server/Datasets" {pools ""}
    resource scratchdisk "/opt/IBM/InformationServer/Server/Scratch" {pools ""}
  }
  node "node2"
  {
    fastname ”ds-engine-compute-0.conductor-0"
    pools ""
    resource disk "/opt/IBM/InformationServer/Server/Datasets" {pools ""}
    resource scratchdisk "/opt/IBM/InformationServer/Server/Scratch" {pools ""}
  }
  {
    fastname ”ds-engine-compute-1.conductor-0"
    pools ""
    resource disk "/opt/IBM/InformationServer/Server/Datasets" {pools ""}
    resource scratchdisk "/opt/IBM/InformationServer/Server/Scratch" {pools ""}
  }
}
```


Dynamic APT_CONFIG_FILE

```
{
  node "node1"
  {
    fastname "is-en-conductor-0.en-cond"
    pools "conductor"
    resource disk "/opt/IBM/InformationServer/Server/Datasets" {pools ""}
    resource scratchdisk "/opt/IBM/InformationServer/Server/Scratch" {pools ""}
  }
  node "node2"
  {
    fastname "$pod"
    pools ""
    resource disk "/opt/IBM/InformationServer/Server/Datasets" {pools ""}
    resource scratchdisk "/opt/IBM/InformationServer/Server/Scratch" {pools ""}
    instances 2
  }
}
```

Dynamic Workload-balancing
Traditional Projects

Targeted for Q3 – Cloud Pak for Data 3.5

1

Persisted

DataStage Flow Designer

xmeta

DataStage Flow Designer

2

Compiled

Traditional Project

Engine/Conductor

RT_SCxx, RT_LOGxx, etc

{REST:API}

#DSHOME/bin/dsjob

{REST:API}

DataStage Flow Designer

3

Run

```
{
  node "node1"
  {
    fastname "is-en-conductor-0.en-cond"
    pools "conductor"
    ...
    ...
  }
  node "node2"
  {
    fastname "$pod"
    pools ""
    ...
    ...
    instances 2
  }
}
```

Engine/Conductor

Compute (n instances)