Db2 Click to Containerize

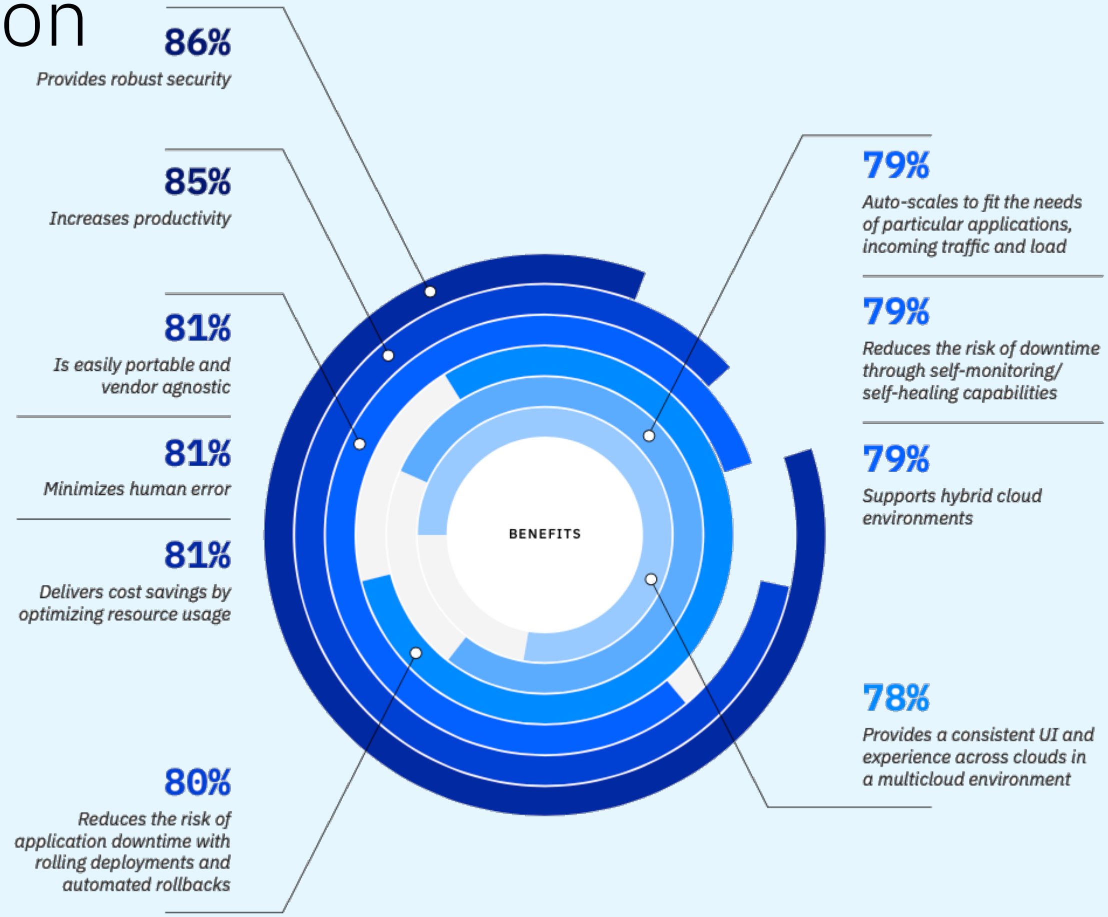
Db2 Night Show #247

George Baklarz
Principal Data & AI Technical Specialist – Americas

Agenda

- Introduction to Db2U and Containerization
- Db2 Click to Containerize
- What's New from Version 1
- Shift Scenarios
- Resources

Benefits of Containerization



86%
Provides robust security

85%
Increases productivity

81%
Is easily portable and vendor agnostic

81%
Minimizes human error

81%
Delivers cost savings by optimizing resource usage

80%
Reduces the risk of application downtime with rolling deployments and automated rollbacks

79%
Auto-scales to fit the needs of particular applications, incoming traffic and load

79%
Reduces the risk of downtime through self-monitoring/ self-healing capabilities

79%
Supports hybrid cloud environments

78%
Provides a consistent UI and experience across clouds in a multicloud environment

The Challenges of Containerizing Databases

If a containerized application has a built-in database, that database disappears when the container shuts down and is recreated from scratch when it boots back up again.

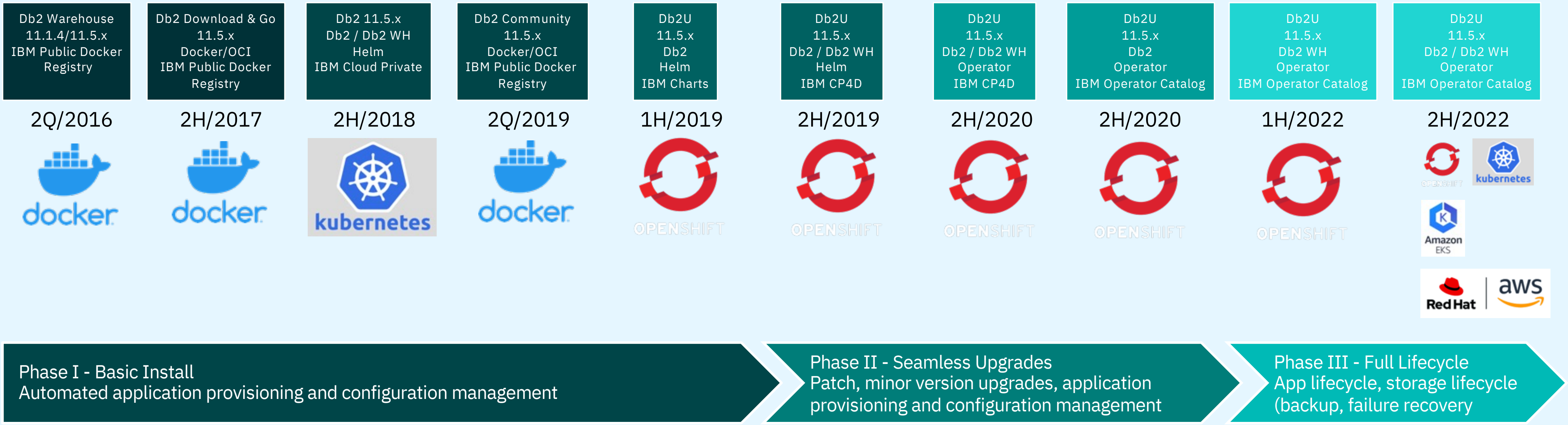
Many databases were not designed to be cloud native, compatible with containers or orchestrated by Kubernetes.

Because of the benefits of moving to a containerized environment, there’s a great deal of interest from enterprises in finding ways to run SQL and no-SQL databases on Kubernetes.

The trick is to provide developers with a data fabric that just works, without forcing developers to struggle with security, auditability or scalability.

https://thenewstack.io/databases-finally-get-containerized/

Db2 Containerization Evolution



Db2 Warehouse
11.1.4/11.5.x
IBM Public Docker Registry

2Q/2016

Db2 Download & Go
11.5.x
Docker/OCI
IBM Public Docker Registry

2H/2017

Db2 11.5.x
Db2 / Db2 WH
Helm
IBM Cloud Private

2H/2018

Db2 Community
11.5.x
Docker/OCI
IBM Public Docker Registry

2Q/2019

Db2U
11.5.x
Db2
Helm
IBM Charts

1H/2019

Db2U
11.5.x
Db2 / Db2 WH
Helm
IBM CP4D

2H/2019

Db2U
11.5.x
Db2 / Db2 WH
Operator
IBM CP4D

2H/2020

Db2U
11.5.x
Db2
Operator
IBM Operator Catalog

2H/2020

Db2U
11.5.x
Db2 WH
Operator
IBM Operator Catalog

1H/2022

Db2U
11.5.x
Db2 / Db2 WH
Operator
IBM Operator Catalog

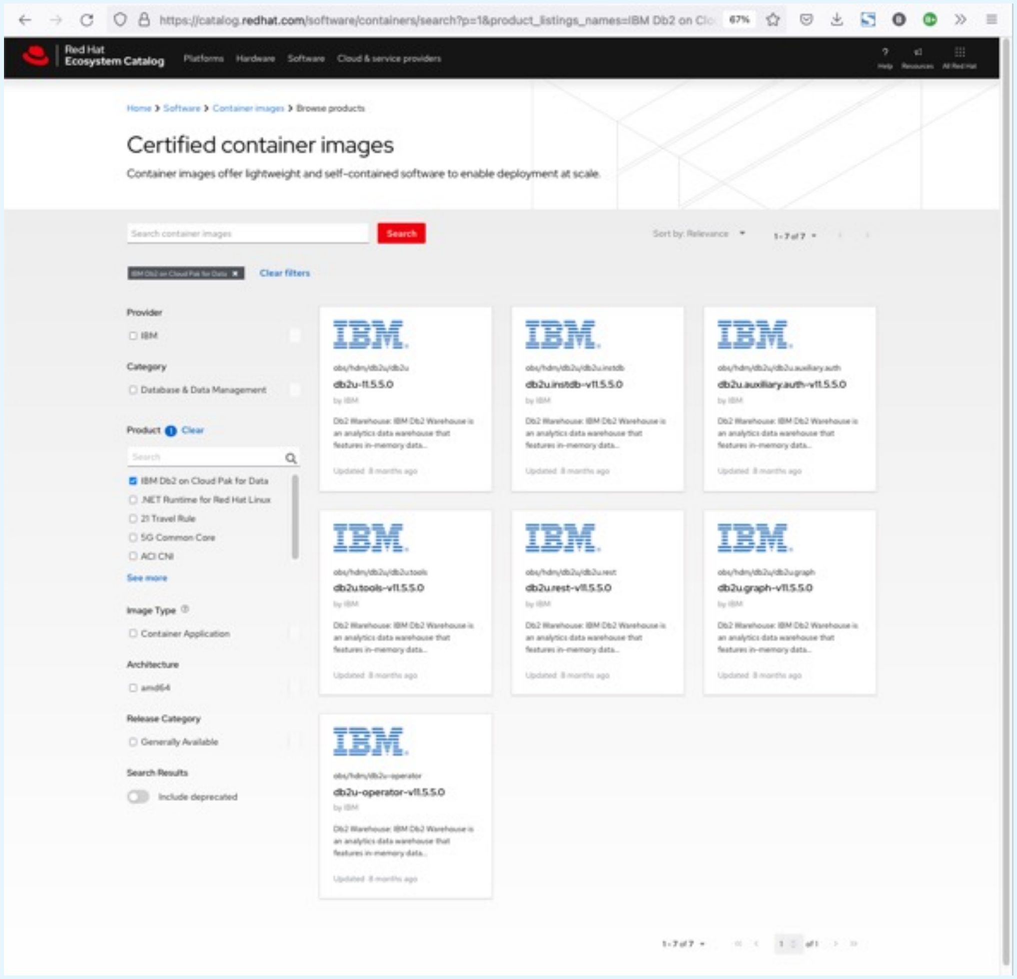
2H/2022

Phase I - Basic Install
Automated application provisioning and configuration management

Phase II - Seamless Upgrades
Patch, minor version upgrades, application provisioning and configuration management

Phase III - Full Lifecycle
App lifecycle, storage lifecycle (backup, failure recovery

Db2 Containerization Support



https://catalog.redhat.com/software/containers/search?p=1&product_listings_names=IBM Db2 on Clo 67%

Red Hat Ecosystem Catalog · Platforms · Hardware · Software · Cloud & service providers

Home › Software › Container images › Browse products

Certified container images

Container images offer lightweight and self-contained software to enable deployment at scale.

Search container images [Search]

Sort by: Relevance ▾ · 1 - 7 of 7 ▾

IBM Db2 on Cloud Pak for Data ✖ · Clear filters

Provider
- □ IBM

Category
- □ Database & Data Management

Product 1 Clear
- ☑ IBM Db2 on Cloud Pak for Data
- □ .NET Runtime for Red Hat Linux
- □ 21 Travel Rule
- □ 5G Common Core
- □ ACI CNI

See more

Image Type
- □ Container Application

Architecture
- □ amd64

Release Category
- □ Generally Available

Search Results
- Include deprecated

IBM
obs/hdm/db2u/db2u
db2u-11.5.5.0
by IBM
Db2 Warehouse: IBM Db2 Warehouse is an analytics data warehouse that features in-memory data...
Updated 8 months ago

IBM
obs/hdm/db2u/db2u.instdb
db2u.instdb-v11.5.5.0
by IBM
Db2 Warehouse: IBM Db2 Warehouse is an analytics data warehouse that features in-memory data...
Updated 8 months ago

IBM
obs/hdm/db2u/db2u.auxiliary.auth
db2u.auxiliary.auth-v11.5.5.0
by IBM
Db2 Warehouse: IBM Db2 Warehouse is an analytics data warehouse that features in-memory data...
Updated 8 months ago

IBM
obs/hdm/db2u/db2u.tools
db2u.tools-v11.5.5.0
by IBM
Db2 Warehouse: IBM Db2 Warehouse is an analytics data warehouse that features in-memory data...
Updated 8 months ago

IBM
obs/hdm/db2u/db2u.rest
db2u.rest-v11.5.5.0
by IBM
Db2 Warehouse: IBM Db2 Warehouse is an analytics data warehouse that features in-memory data...
Updated 8 months ago

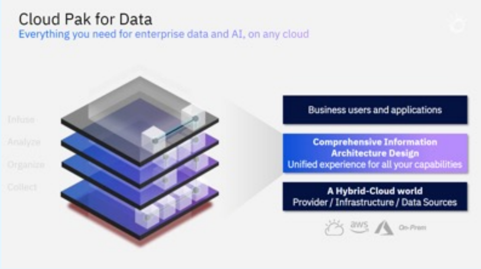
IBM
obs/hdm/db2u/db2u.graph
db2u.graph-v11.5.5.0
by IBM
Db2 Warehouse: IBM Db2 Warehouse is an analytics data warehouse that features in-memory data...
Updated 8 months ago

IBM
obs/hdm/db2u-operator
db2u-operator-v11.5.5.0
by IBM
Db2 Warehouse: IBM Db2 Warehouse is an analytics data warehouse that features in-memory data...
Updated 8 months ago

1 - 7 of 7 ▾ · 1 of 1

Red Hat OpenShift

kubernetes


Cloud Pak for Data
Everything you need for enterprise data and AI, on any cloud
Infuse
Analyze
Organize
Collect
Business users and applications
Comprehensive Information Architecture Design
Unified experience for all your capabilities
A Hybrid-Cloud world
Provider / Infrastructure / Data Sources
aws
On-Prem

Benefits of Db2u Containerization

Quicker time to value
Significantly reduce deployment time

Containerized deployment
Reduce operation costs by 26%[1]

Cloud-native database
Reduce infrastructure costs by 40%[2]

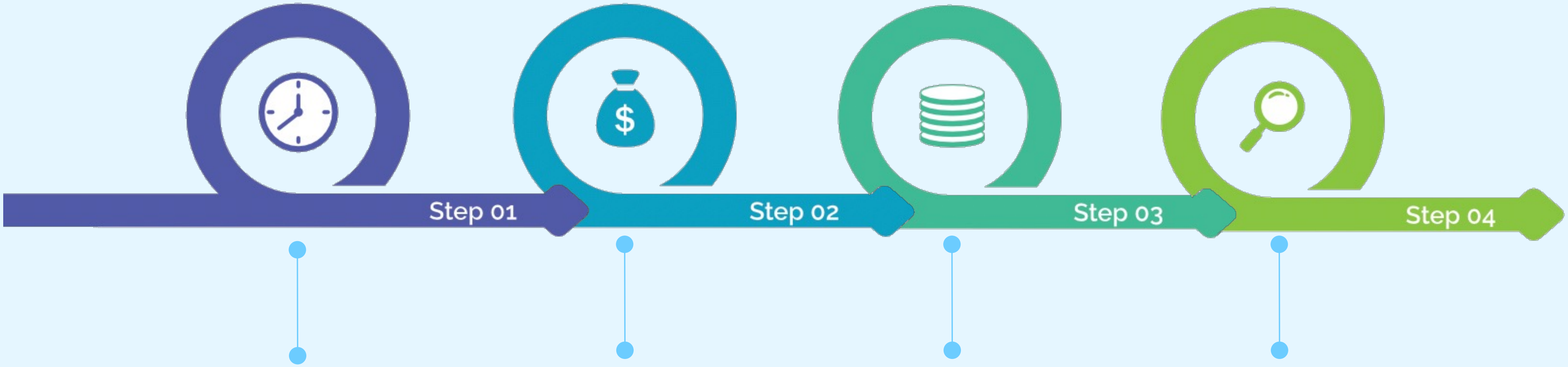
Data virtualization (CP4D)
Reduce data extraction and transformation requests up to 65%[3]

Storage cost savings
Save up to 50%[4]

Enhanced productivity
66% faster application development[5]

Metrics	Db2U on OpenShift
Install Time	[Automated] ▪ 5' – 10' (Db2 OLTP / WH SMP) ▪ 10' – 20 ' (WH MPP)
Upgrade Time	[Automated] 15' – 30' (Dependent on number of catalog objects)
[Db2 OLTP] HADR Takeover Time	[Automated] 2' – 4'
[Db2 WH MPP] Recovery on Db2 Failure	[Automated] ▪ Non-catalog POD failure: 3' + Db2 Crash Recovery time ▪ Catalog POD: 5 + Db2 Crash Recovery time
Recovery on Hardware Failure	[Semi-automated] OCP/k8s Node failure detection time + *Recovery on Db2 Failure* time
[Db2 WH MPP] Scaling Compute	[Semi-automated] 2 -3' + Db2 autoconfigure time

Migrating to Containers



Planning

How long will the move take?
Is there a migration required?
Do I need to decrypt the data?
Is the database offline?
Do my applications change?

Cost

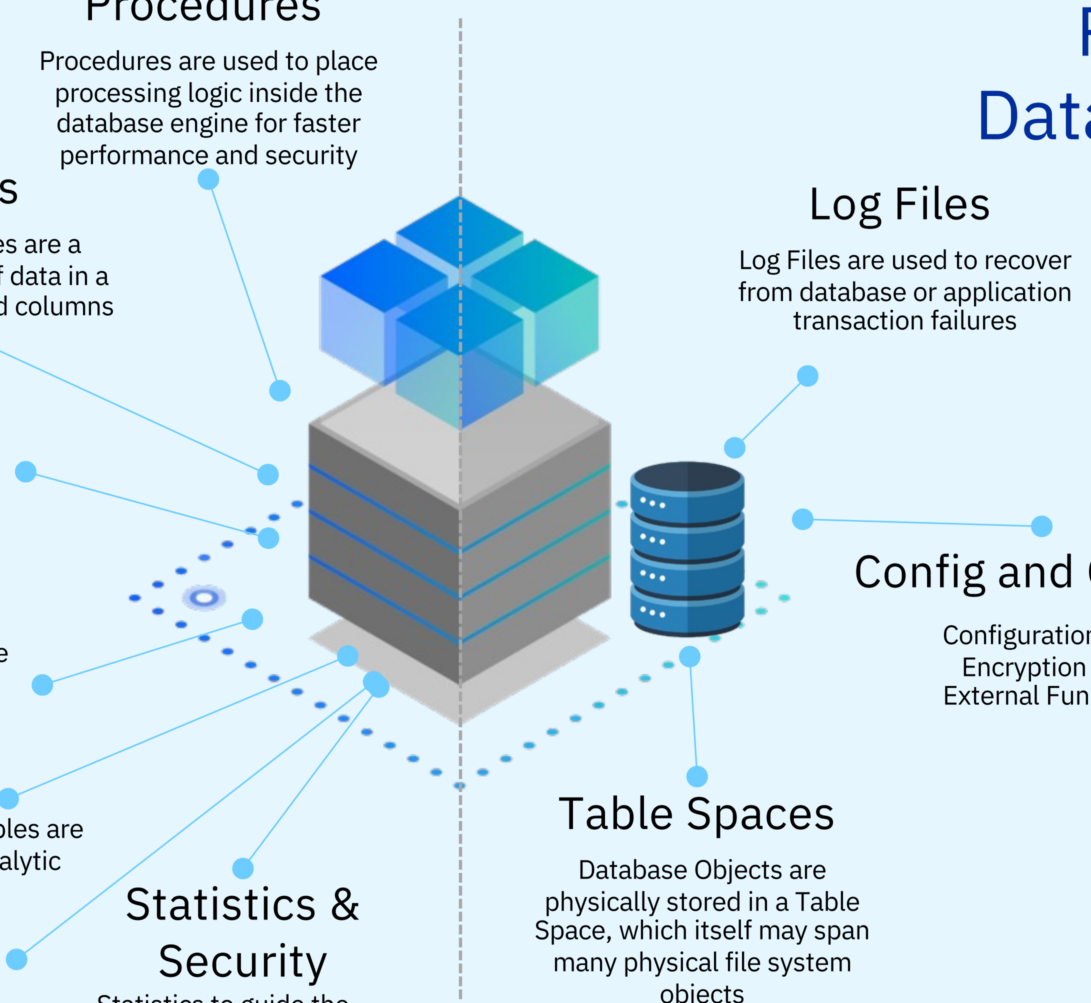
How much time will it take?
Do I need more resources?
How much of an outage?
Manual or Automated effort?

Data Movement

Is the transport secure?
Do I need to expose the data?
How fast can I move data?
Is the process recoverable?
Are the IOPs sufficient?

Verification

Do I need to verify all data?
Is optimization required?
Can applications connect?
Additional features needed?
Are the databases in sync?



Logical Database View

Physical Database View

Procedures

Procedures are used to place processing logic inside the database engine for faster performance and security

Tables

Database tables are a representation of data in a series of rows and columns

Views

Views are used to combine or redefine tables in a different logical view

Indexes

Indexes are used to improve the time required to find an answer set.

MQT

Materialized Query Tables are used to speed up Analytic queries

Other Database Objects

Buffer pools/Tablespaces/Parameters

Statistics & Security

Statistics to guide the Optimizer and Security Authorizations

Log Files

Log Files are used to recover from database or application transaction failures

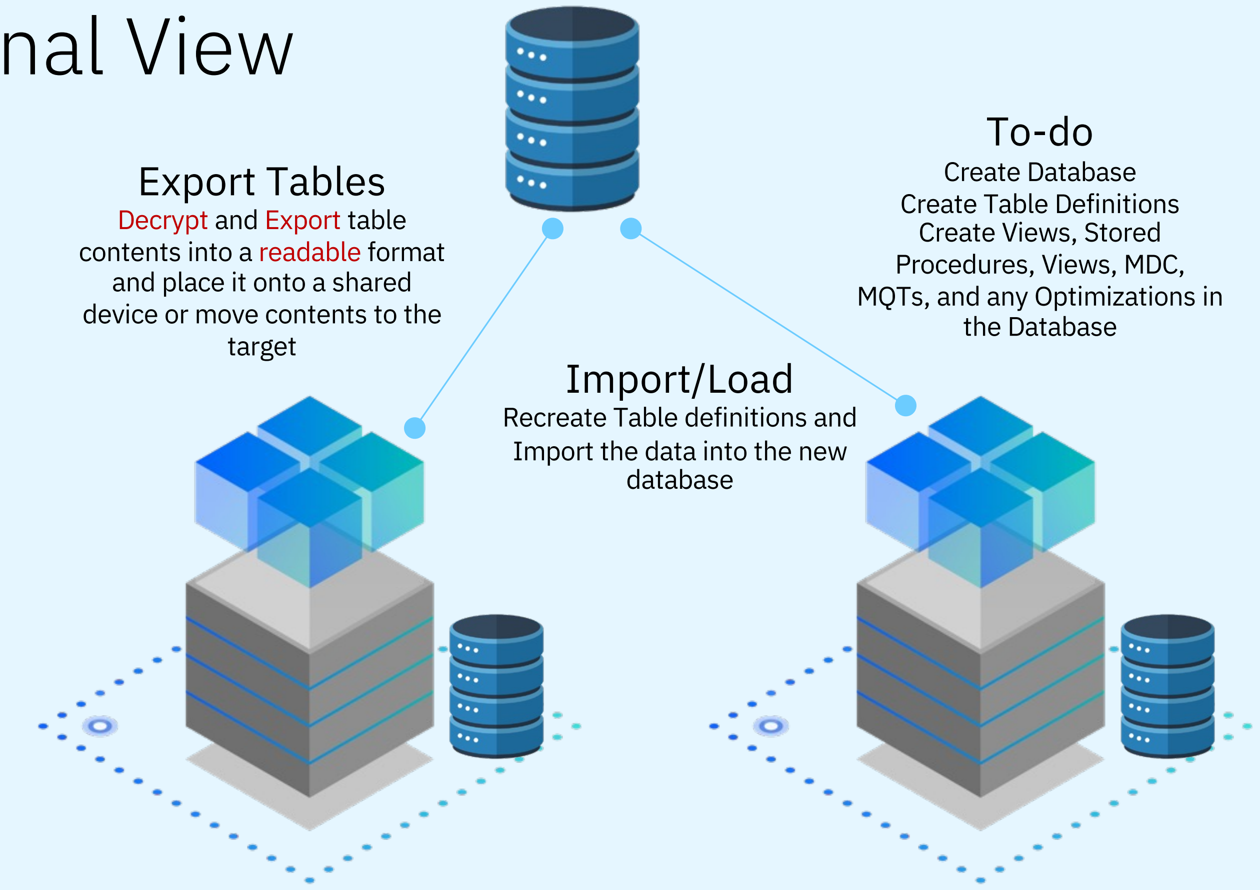
Config and Ctrl Files

Configuration Files
Encryption files
External Functions

Table Spaces

Database Objects are physically stored in a Table Space, which itself may span many physical file system objects

Migrating a Database
A Traditional View



Export Tables

Decrypt and Export table contents into a readable format and place it onto a shared device or move contents to the target

Import/Load

Recreate Table definitions and Import the data into the new database

To-do

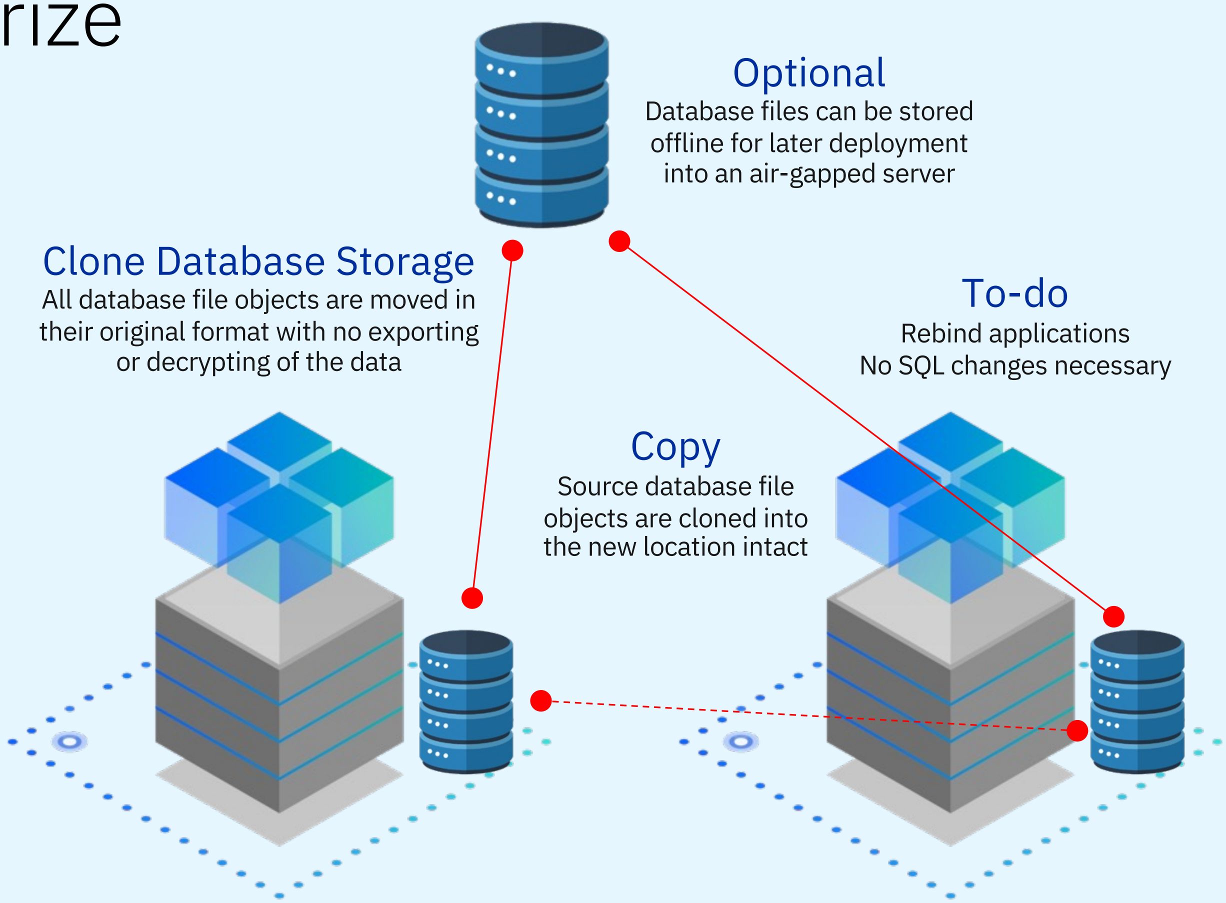
Create Database
Create Table Definitions
Create Views, Stored Procedures, Views, MDC, MQTs, and any Optimizations in the Database

Db2 Click to Containerize

The fastest way to move your data from an On-premise Db2 Linux system to a Cloud Environment

- Move to OpenShift, Kubernetes, or Cloud Pak for Data
- Containerize your data or move to a traditional server instance
- Command line or UI tool for flexibility
- With automatic database upgrades

More info: ibm.biz/c2cguide



Optional
Database files can be stored offline for later deployment into an air-gapped server

Clone Database Storage
All database file objects are moved in their original format with no exporting or decrypting of the data

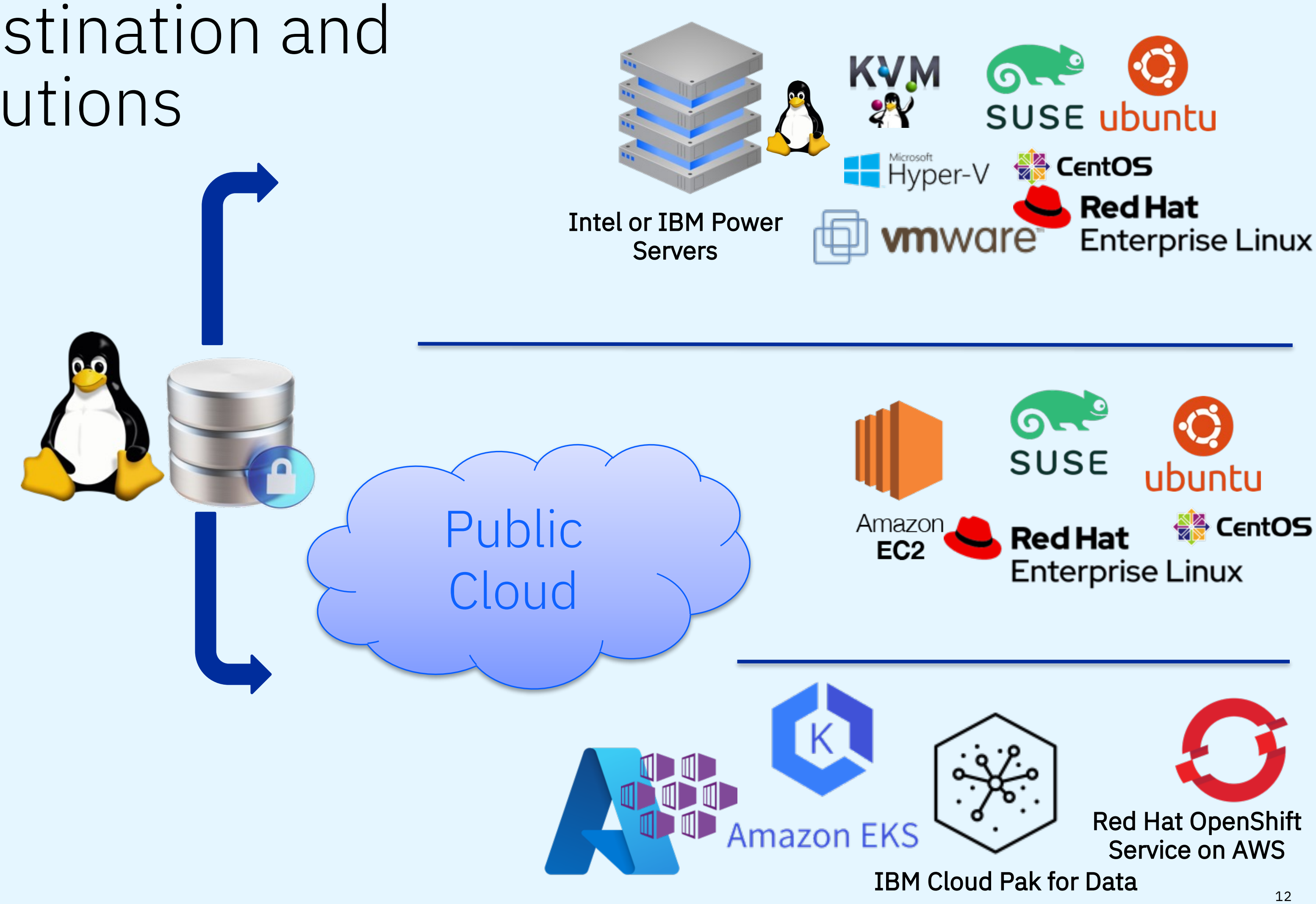
To-do
Rebind applications
No SQL changes necessary

Copy
Source database file objects are cloned into the new location intact

Choice of Destination and Linux Distributions

Db2 Linux Server x64 or powerLE

- 10.5 fp7
- 11.1
- 11.5



Intel or IBM Power Servers

KVM SUSE ubuntu Microsoft Hyper-V CentOS Red Hat Enterprise Linux vmware

Public Cloud

Amazon EC2 SUSE ubuntu Red Hat Enterprise Linux CentOS

Amazon EKS

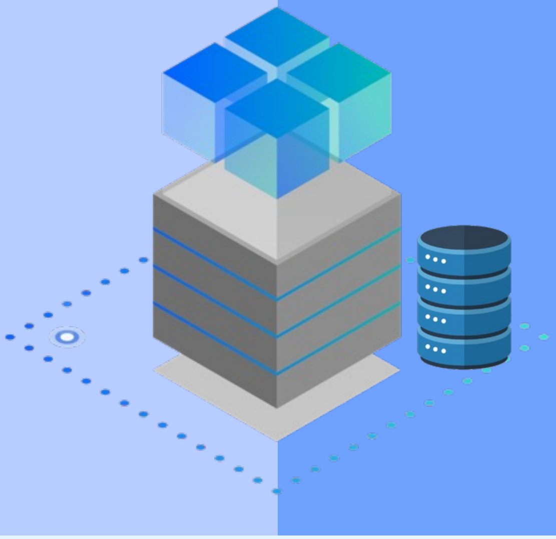
IBM Cloud Pak for Data

Red Hat OpenShift Service on AWS

Which Approach to Use?

Database Migration

Move between different O/S (AIX, zOS)
Change configuration and codepages
Unsupported database versions
All or subset of data
SQL Access only
Topology change (MLNs for DPF)
Object recreation required
Data must be exported and decrypted



Db2 Click to Containerize

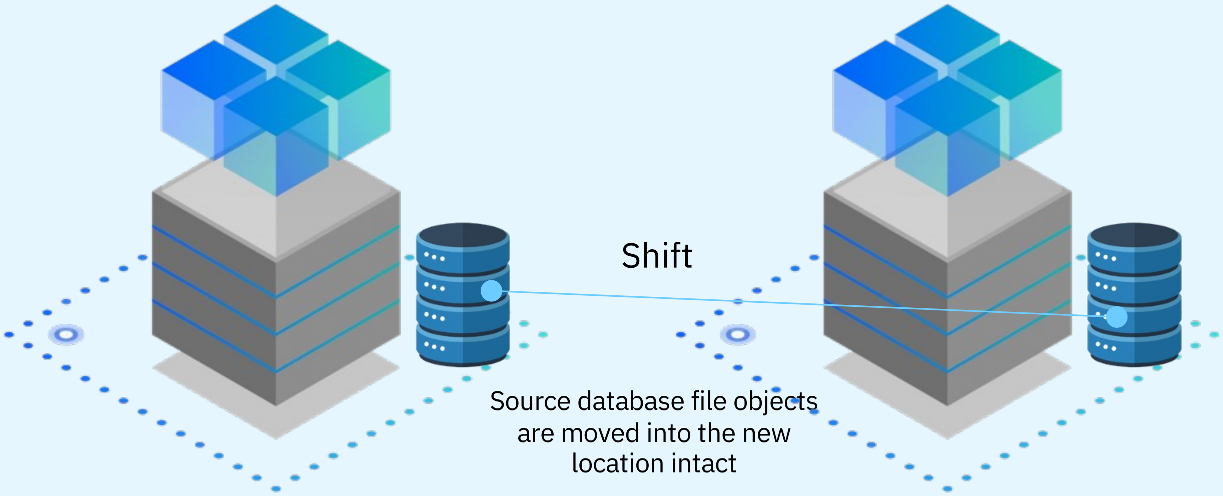
Move only Linux to Linux environment
All database features moved intact
Current releases only 10.5, 11.1, 11.5
Entire database movement
Direct file access, no SQL used
No change in configuration allowed
No rebuild of objects required
No exposure of the data

Demonstration Scenario

Table	Size
Database Size (RAW)	100Gb
Customers	2000000
Addresses	941566
Customer Demographics	1920800
Date Dimensions	73049
Household Demographics	7200
Income Bands	20
Items	202960
Promotions	1000
Stores	394
Store Sales	262185014
Time Dimensions	36000

Db2 CQW Database
m5a.2xlarge
8 VPC 32Gb
CentOS 7 AMI
Db2 11.5.4

EKS Cluster
Db2 Worker Node
r5b.4xlarge
16 VPC 128Gb
Db2 11.5.7



Shift

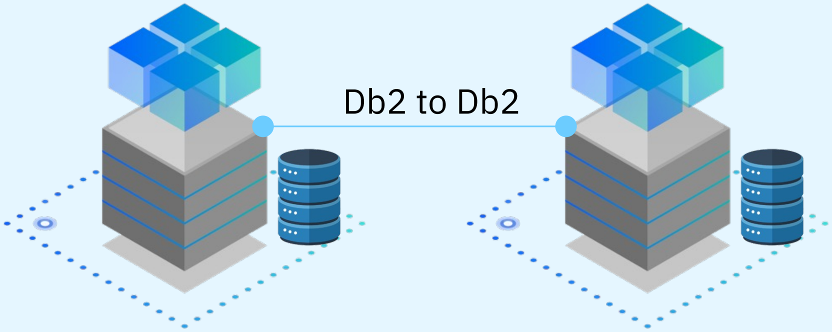
Source database file objects
are moved into the new
location intact

Shift Targets

- The Shift targets have been expanded in the newest release
  - Continues to support OpenShift 3.11, and 4+ clusters
  - CP4D V3.5 and V4+ are supported
  - New! Kubernetes Distributions (1.19+) are supported
  - New! MicroK8S, MiniKube for small systems
- Operating System Support
  - CentOS 6, 7, 8, RHEL 6, 7, 8, CentOS Stream
  - Ubuntu 18.04, 20.04. 22.04
  - Power LE
  - SUSE 15

Shift Modes

- The Shift mode has been expanded
  - Shift between Db2 instance and another Db2 instance (traditional Db2)
  - Create Database on target if it doesn't exist (Db2 instance only)
  - Set up HADR at target and initialize source and target (Pod or Instance)
  - Create cloned databases without requiring target information (Pod or Instance)
    - Useful in air-gapped environments
  - Clone copy to and from a POD
  - Reverse shift (back-out) from POD back to an instance

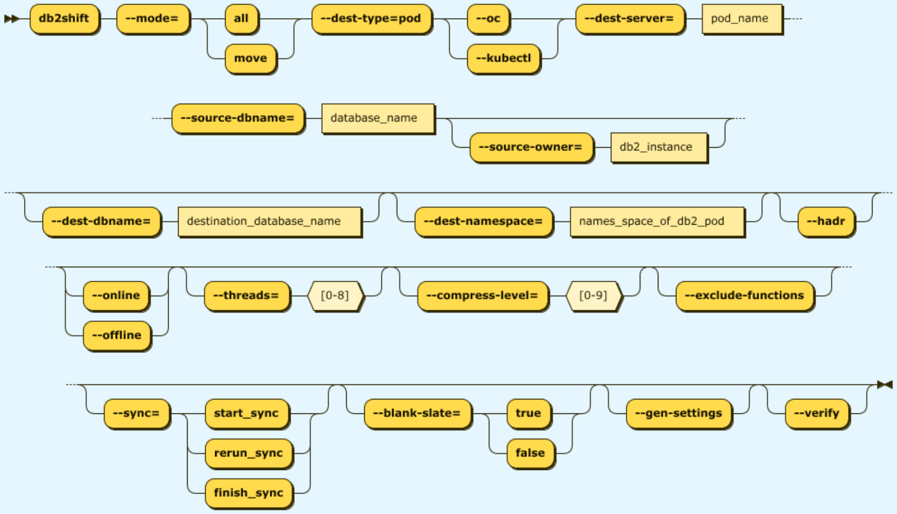


Db2 Shift Scenarios

- The Db2 Shift utility provides support for 9 scenarios
  - Shift Operations
    - Shift a Db2 database to OpenShift, Kubernetes or CP4D
    - Shift a Db2 database to another Db2 instance
  - Clone and Deploy a Database
    - Create a Cloned copy of the Db2 database for later deployment
    - Deploy a clone into an OpenShift, Kubernetes or CP4D container
    - Deploy a clone into another Db2 instance
  - HADR Operations
    - Initialize HADR between Source and Target POD
    - Initialize HADR between Source and Target Instance
    - Initialize DMC and LDAP Authentication for CP4D
  - Copy Cloned Databases to a POD

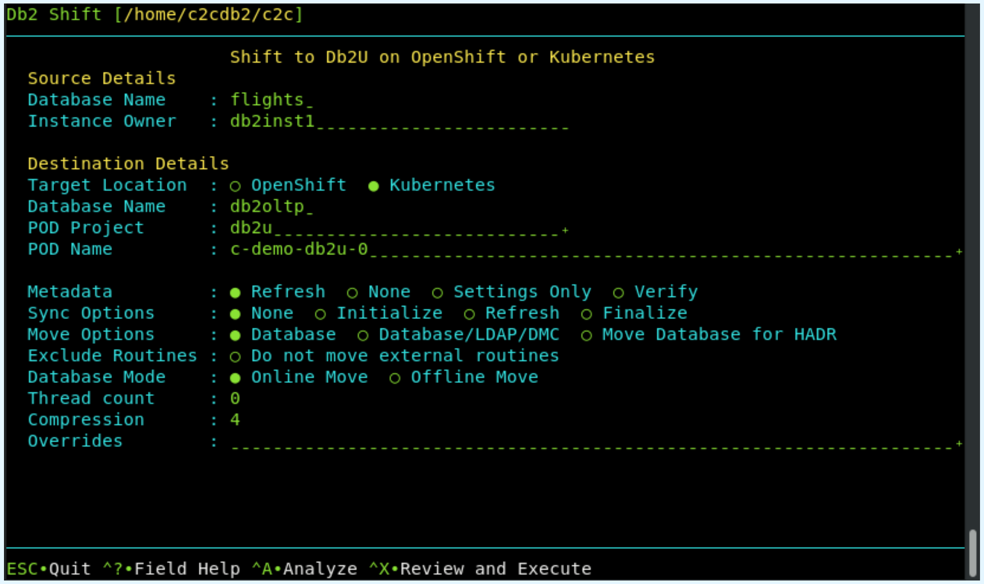
Containerize Db2 to OpenShift, Kubernetes, or CP4D

- Containerizing an existing Db2 database on an on-premise system to OpenShift, Kubernetes, or CP4D is the most common scenario for using Db2 Shift
- In order to move a Db2 database to a POD you will require the following information:
  - Source Database details
  - Destination location and POD details
  - Type of Containerization environment
  - Shift Options
- The target of the Db2 Shift operation can be OpenShift, a Kubernetes cluster, or Cloud Pak for Data



Containerize Db2 to OpenShift, K8s or CP4D Dialog

- The Source and Destination Details are the only required fields



```
Db2 Shift [/home/c2cdb2/c2c]

                     Shift to Db2U on OpenShift or Kubernetes
  Source Details
  Database Name     : flights_
  Instance Owner    : db2inst1________________________

  Destination Details
  Target Location   : ○ OpenShift  ● Kubernetes
  Database Name     : db2oltp_
  POD Project       : db2u___________________________+
  POD Name          : c-demo-db2u-0______________________________________________________+

  Metadata          : ● Refresh  ○ None  ○ Settings Only  ○ Verify
  Sync Options      : ● None  ○ Initialize  ○ Refresh  ○ Finalize
  Move Options      : ● Database  ○ Database/LDAP/DMC  ○ Move Database for HADR
  Exclude Routines  : ○ Do not move external routines
  Database Mode     : ● Online Move  ○ Offline Move
  Thread count      : 0
  Compression       : 4
  Overrides         : _____________________________________________________________________+




ESC•Quit ^?•Field Help ^A•Analyze ^X•Review and Execute
```

- The additional fields provide more options for the Shift process

Optional Fields

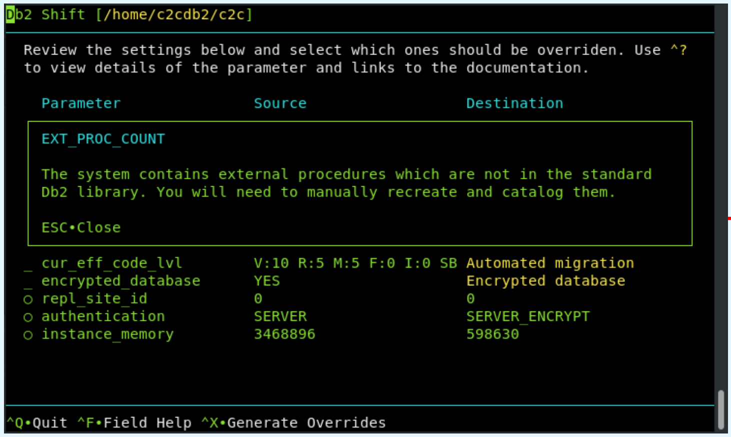
- Metadata
  - Used for gathering information on the source and target destination
  - No need to use this unless you want to test the connections between the source and the target (verify) or you are shifting a database that needs to be offline (upgraded)
- Sync Options
  - By default, all shifts are performed from beginning to end
  - You can shift incrementally by using the Initialize/Refresh/Finalize Options
- Move Options
  - Use Database for OpenShift and K8s, Database/LDAP/DMC for CP4D, and HADR if you want the target POD created as a standby for the primary database
- Exclude Routines
  - Do not move any external routines found on the source database library path

Optional Fields (Continued)

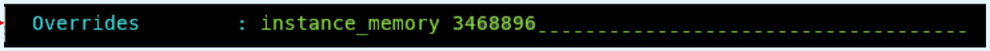
- Database Mode
  - The database can be online during a shift operation (11.5 only)
  - For databases that require an upgrade (10.5 and 11.1), the database needs to be offline for the shift to work properly (Db2 requirement)
- Thread Count
  - The Db2 shift utility can use up to 8 threads when communicating with the target location
  - The more threads, the more throughput, but at the expense of CPU resources
  - The limit is usually the Ethernet performance, not the threads
- Compression
  - The data stream can be compressed when communicating with the target
  - If the network is >1Gb/sec then there is no benefit from using this setting
  - Anything less than 1Gb/sec may find compression helps at the expense of CPU cycles

Optional Fields (Continued)

- Overrides
  - All source database settings are retained at the target location
  - Instance settings do NOT move to the target
    - You may need to adjust the target's instance settings
    - This field provides a way of setting instance values that will be updated during the shift process
  - The Analyze option (^A) will provide a summary of the differences between the source and target and allow you to select which settings to override



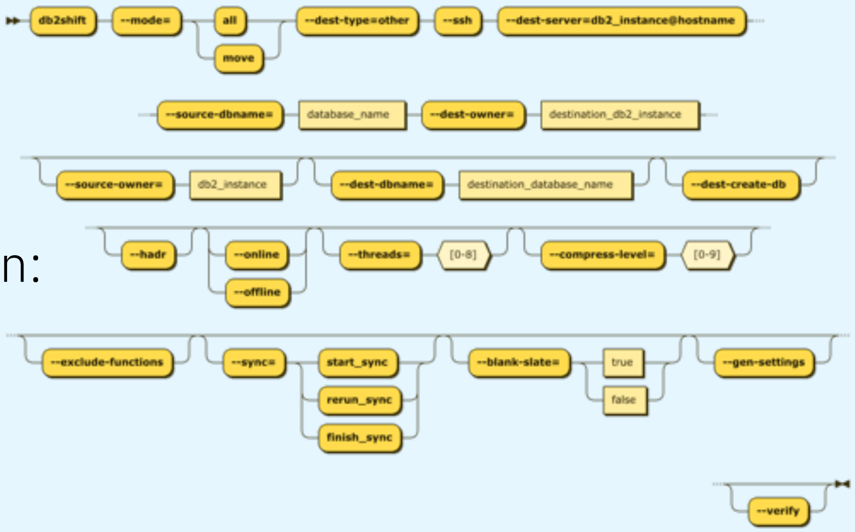
```
Db2 Shift [/home/c2cdb2/c2c]

  Review the settings below and select which ones should be overriden. Use ^?
  to view details of the parameter and links to the documentation.

    Parameter                Source                   Destination
   ┌──────────────────────────────────────────────────────────────────────────┐
   │EXT_PROC_COUNT                                                            │
   │                                                                          │
   │The system contains external procedures which are not in the standard     │
   │Db2 library. You will need to manually recreate and catalog them.         │
   │                                                                          │
   │ESC•Close                                                                 │
   └──────────────────────────────────────────────────────────────────────────┘
  _ cur_eff_code_lvl         V:10 R:5 M:5 F:0 I:0 SB Automated migration
  _ encrypted_database       YES                     Encrypted database
  ○ repl_site_id             0                       0
  ○ authentication           SERVER                  SERVER_ENCRYPT
  ○ instance_memory          3468896                 598630


^Q•Quit ^F•Field Help ^X•Generate Overrides
```



```
Overrides          : instance_memory 3468896_____________________________________
```

Shift Db2 to another Instance

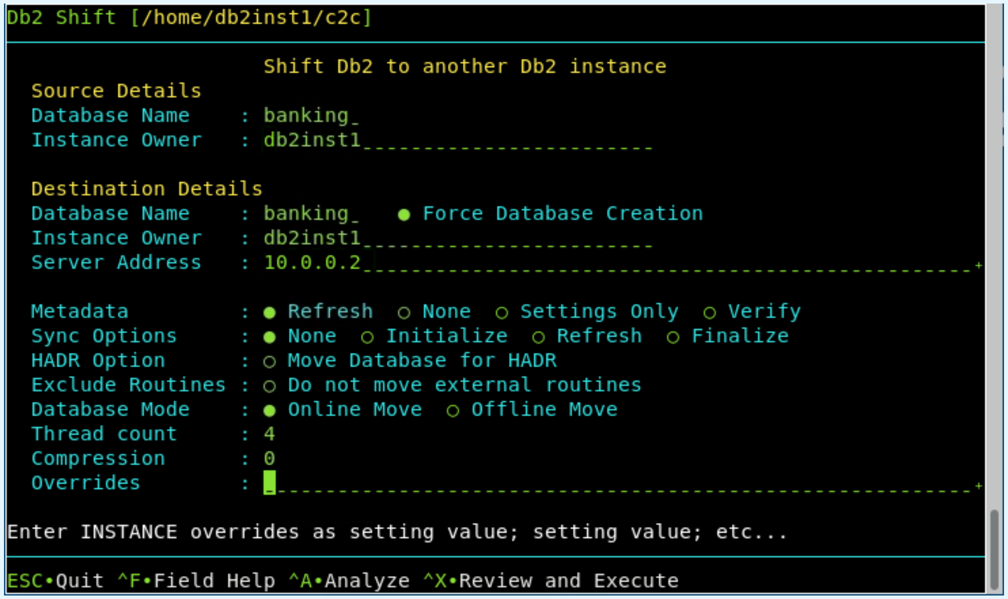
- This format of the Db2 Shift will take an existing Db2 database on an on-premise system, and shift it to another traditional Db2 system to:
  - Another on-premise server
  - A Cloud virtual machine (i.e. EC2)
- This does not containerize Db2!
- The command requires the following information:
  - Source Database details
  - Destination location
  - Shift Options



- The db2shift program assumes that you are currently connected to the instance that has the Db2 database and have ssh connectivity to the target server

Shift Db2 to another Instance

- The Source and Destination Details are the only required fields

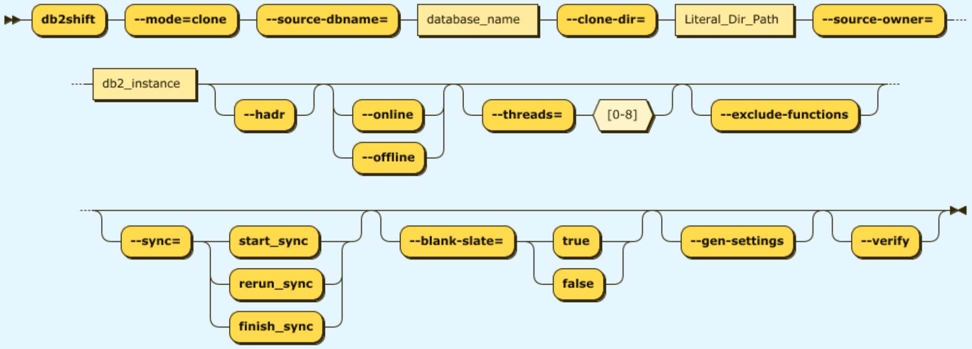


```
Db2 Shift [/home/db2inst1/c2c]

                     Shift Db2 to another Db2 instance
  Source Details
  Database Name     : banking_
  Instance Owner    : db2inst1________________________

  Destination Details
  Database Name     : banking_    ● Force Database Creation
  Instance Owner    : db2inst1________________________
  Server Address    : 10.0.0.2__________________________________________________+

  Metadata          : ● Refresh  ○ None  ○ Settings Only  ○ Verify
  Sync Options      : ● None  ○ Initialize  ○ Refresh  ○ Finalize
  HADR Option       : ○ Move Database for HADR
  Exclude Routines  : ○ Do not move external routines
  Database Mode     : ● Online Move  ○ Offline Move
  Thread count      : 4
  Compression       : 0
  Overrides         : _____________________________________________________________+

Enter INSTANCE overrides as setting value; setting value; etc...

ESC•Quit ^F•Field Help ^A•Analyze ^X•Review and Execute
```

- The FORCE DATABASE CREATION option will create the database at the target
- The additional fields provide more options for the Shift process and are identical to the Db2 Shift to POD command

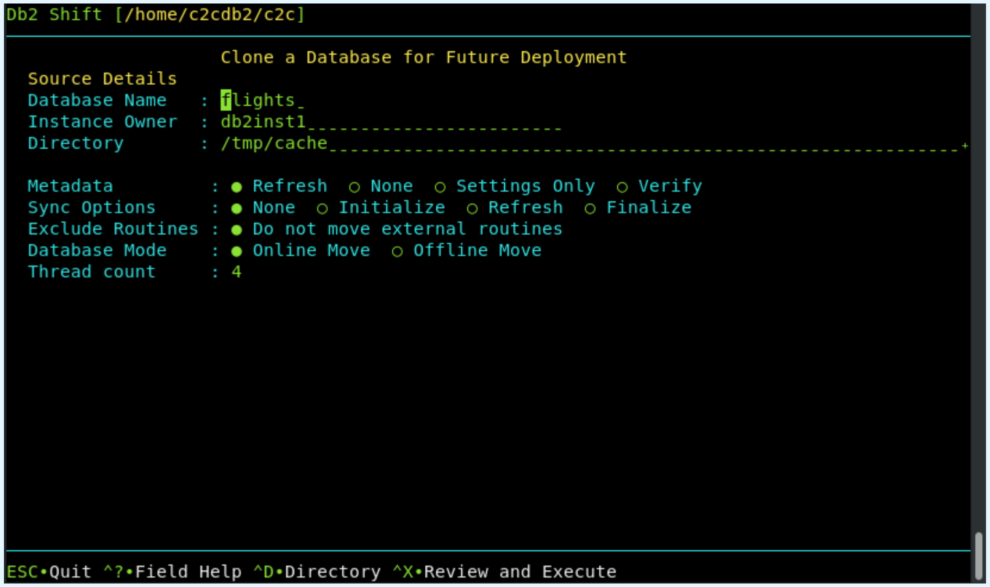
Clone a Database

- The Db2 Shift clone option is used to take an existing Db2 database that is currently on-premise, and clone it into a directory
  - This cloned database can be transported to another server and deployed at that location
  - The advantage of cloning is that the destination does not need to be connected to the source location and the deployment of the clone can be done at a more convenient time
- The Db2 Shift program requires the following information:
  - Source Database details
  - Clone Options
- The destination details are not required to clone a database



Create a Clone of a Database

- The Source is required as well as the directory to place the cloned copy

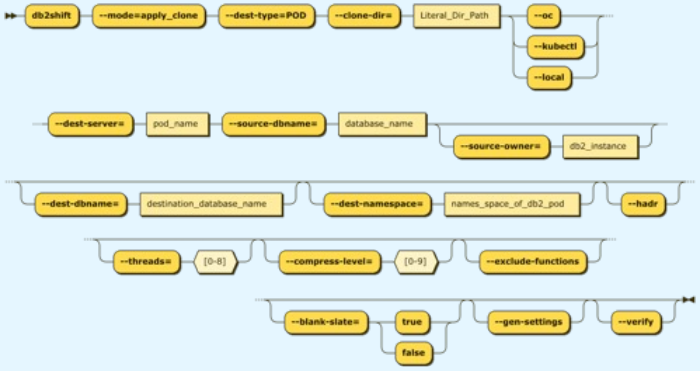


```
Db2 Shift [/home/c2cdb2/c2c]

                    Clone a Database for Future Deployment
  Source Details
  Database Name    : flights_
  Instance Owner   : db2inst1________________________
  Directory        : /tmp/cache___________________________________________________________+

  Metadata          : ● Refresh  ○ None  ○ Settings Only  ○ Verify
  Sync Options      : ● None  ○ Initialize  ○ Refresh  ○ Finalize
  Exclude Routines  : ● Do not move external routines
  Database Mode     : ● Online Move  ○ Offline Move
  Thread count      : 4

ESC•Quit ^?•Field Help ^D•Directory ^X•Review and Execute
```

- The additional fields provide more options for the Shift process and are identical to the Db2 Shift to POD command
- HADR and Compression options are not used

Deploy a Clone to a POD

- This Db2 Shift option will take a database clone and deploy it into a Db2u pod running on OpenShift, Kubernetes or CP4D
- The requirements for deploying to a POD are identical to a direct shift
- The command requires the following information:
  - Database name and clone location
  - Destination POD details
  - Type of Containerization environment
  - Shift Options

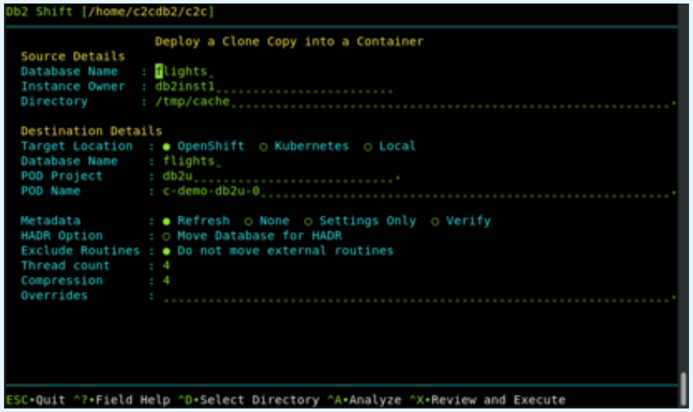

db2shift --mode=apply_clone --dest-type=POD --clone-dir= Literal_Dir_Path --oc --kubectl --local

--dest-server= pod_name --source-dbname= database_name --source-owner= db2_instance

--dest-dbname= destination_database_name --dest-namespace= names_space_of_db2_pod --hadr

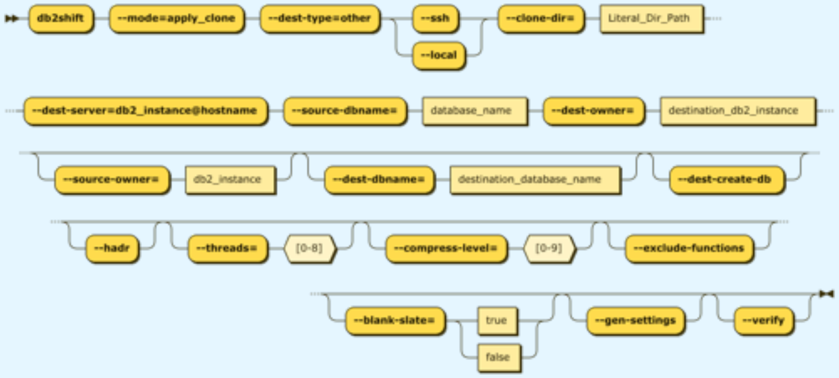
--threads= [0-8] --compress-level= [0-9] --exclude-functions

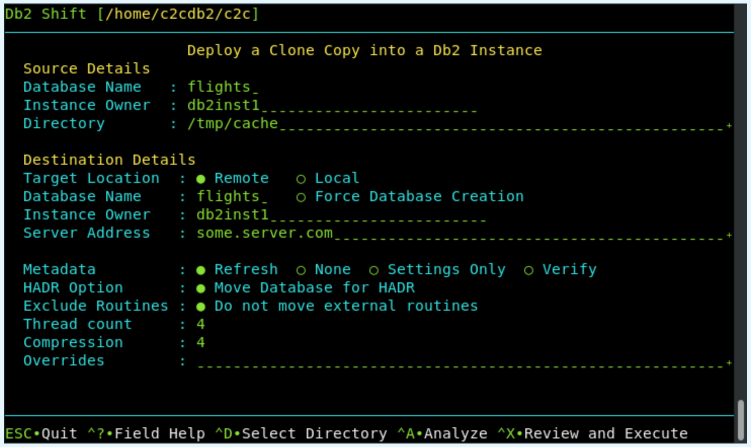
--blank-slate= true false --gen-settings --verify


```
Db2 Shift [/home/c2cdb2/c2c]

                    Deploy a Clone Copy into a Container
  Source Details
  Database Name    : flights_
  Instance Owner   : db2inst1........................
  Directory        : /tmp/cache..........................................................

  Destination Details
  Target Location   : ● OpenShift  ○ Kubernetes  ○ Local
  Database Name     : flights_
  POD Project       : db2u...........................
  POD Name          : c-demo-db2u-0..................................................

  Metadata          : ● Refresh  ○ None  ○ Settings Only  ○ Verify
  HADR Option       : ○ Move Database for HADR
  Exclude Routines  : ● Do not move external routines
  Thread count      : 4
  Compression       : 4
  Overrides         : ..........................................................................

ESC-Quit ^?-Field Help ^D-Select Directory ^A-Analyze ^X-Review and Execute
```

Deploy a Clone an Instance

- This Db2 Shift option will take a database clone and deploy it into another Db2 instance running natively (in any environment including Cloud VMs)
- The requirements for deploying to an Instance are identical to a direct shift
- The command requires the following information:
  - Database name and clone location
  - Destination instance details
  - Shift Options

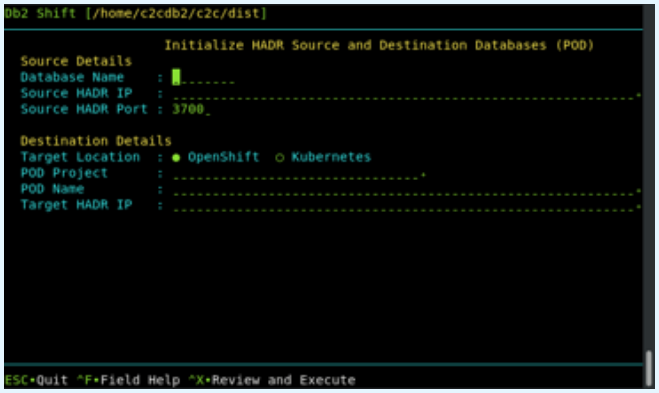




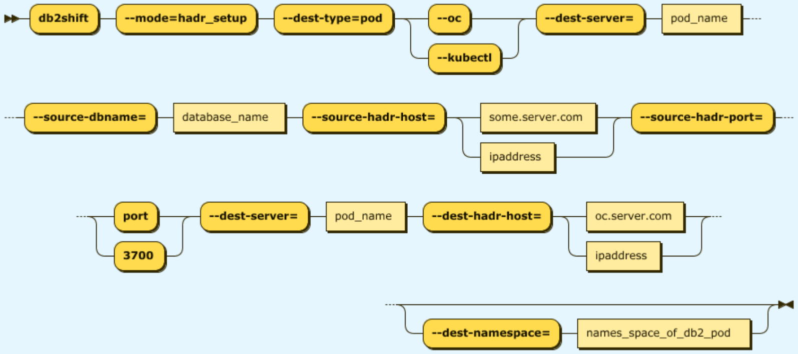
```
Db2 Shift [/home/c2cdb2/c2c]

                    Deploy a Clone Copy into a Db2 Instance
  Source Details
  Database Name   : flights_
  Instance Owner  : db2inst1________________________
  Directory       : /tmp/cache_______________________________________________

  Destination Details
  Target Location  : ● Remote   ○ Local
  Database Name    : flights_   ○ Force Database Creation
  Instance Owner   : db2inst1________________________
  Server Address   : some.server.com_________________________________________

  Metadata         : ● Refresh  ○ None  ○ Settings Only  ○ Verify
  HADR Option      : ● Move Database for HADR
  Exclude Routines : ● Do not move external routines
  Thread count     : 4
  Compression      : 4
  Overrides        : __________________________________________________________

ESC•Quit ^?•Field Help ^D•Select Directory ^A•Analyze ^X•Review and Execute
```

Initialize HADR to a POD

- This Db2 Shift option will take a source and destination (POD) database and start the HADR service between them
  - The Db2u pod must have been created with the HADR option during the shift step
  - Note that a destination HADR port is not required since it is automatically generated for you by the Db2 Shift program
- The command requires the following information:
  - The source database name and server
  - The destination POD and server details

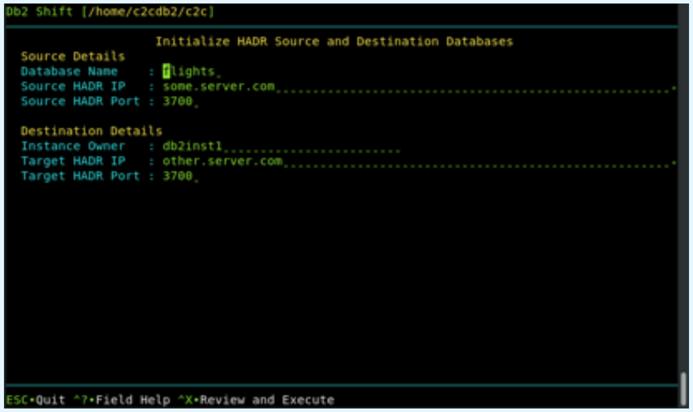


```
Db2 Shift [/home/c2cdb2/c2c/dist]

                    Initialize HADR Source and Destination Databases (POD)
  Source Details
  Database Name     : _.......
  Source HADR IP    : ..........................................................
  Source HADR Port  : 3700_

  Destination Details
  Target Location   : ● OpenShift   ○ Kubernetes
  POD Project       : ..............................
  POD Name          : ..........................................................
  Target HADR IP    : ..........................................................

ESC•Quit ^F•Field Help ^X•Review and Execute
```

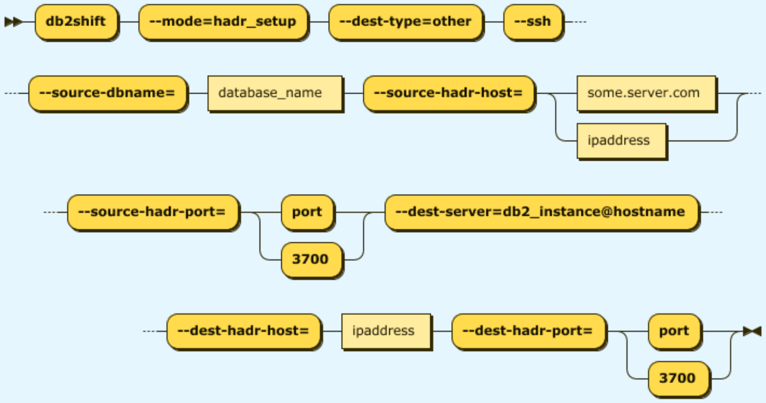


Initialize HADR to an Instance

- This menu is similar to the previous one where the HADR service is setup between the source Db2 database and another Db2 instance
  - The Db2 database on the target system must have been created with the HADR option during the Shift operation
- The command requires the following information:
  - The source database name and server
  - The destination server details



```
Db2 Shift [/home/c2cdb2/c2c]

                    Initialize HADR Source and Destination Databases
  Source Details
  Database Name    : flights_
  Source HADR IP   : some.server.com_.........................................
  Source HADR Port : 3700_

  Destination Details
  Instance Owner   : db2inst1_.......................
  Target HADR IP   : other.server.com_........................................
  Target HADR Port : 3700_

ESC•Quit ^?•Field Help ^X•Review and Execute
```



db2shift --mode=hadr_setup --dest-type=other --ssh

--source-dbname= database_name --source-hadr-host= some.server.com | ipaddress

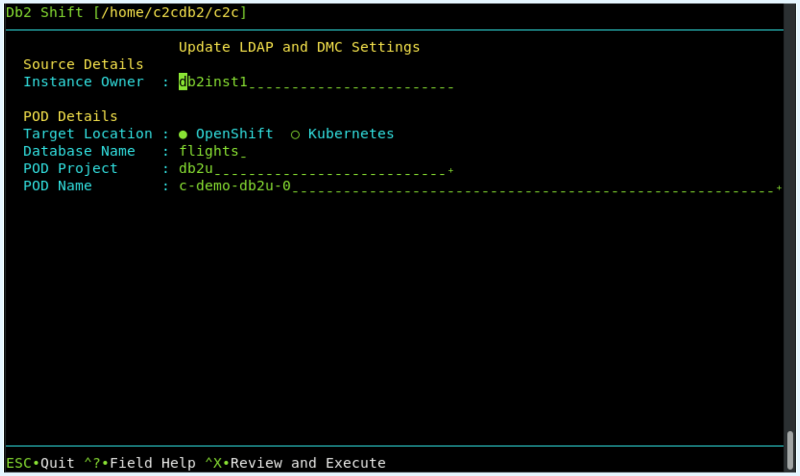
--source-hadr-port= port | 3700 --dest-server=db2_instance@hostname

--dest-hadr-host= ipaddress --dest-hadr-port= port | 3700

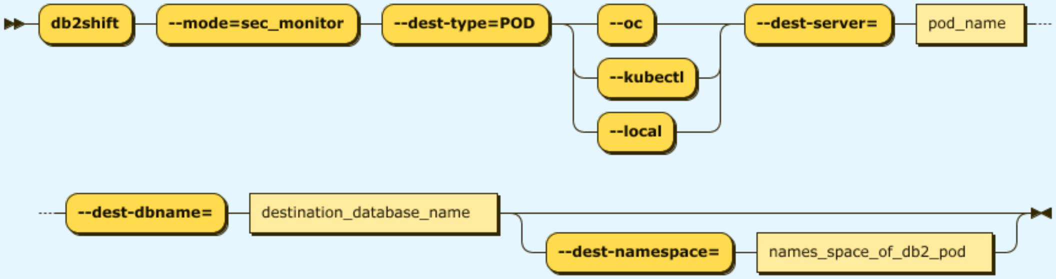
Initialize LDAP and DMC (CP4D Only)

- Db2 Shift will automatically add the appropriate userids to the LDAP service and reset the Data Management Console (DMC) when shifting to CP4D
  - If you choose to shift a database and set it up as an HADR secondary, the LDAP and DMC setup cannot be performed until after you have converted the secondary POD into the primary server

- You must supply the name of the database, POD name and project as well as the name of the instance owner to update the LDAP and DMC settings



```
Db2 Shift [/home/c2cdb2/c2c]

                         Update LDAP and DMC Settings
  Source Details
  Instance Owner  : db2inst1________________________

  POD Details
  Target Location : ● OpenShift  ○ Kubernetes
  Database Name   : flights_
  POD Project     : db2u___________________________+
  POD Name        : c-demo-db2u-0_________________________________________________________+

ESC•Quit ^?•Field Help ^X•Review and Execute
```

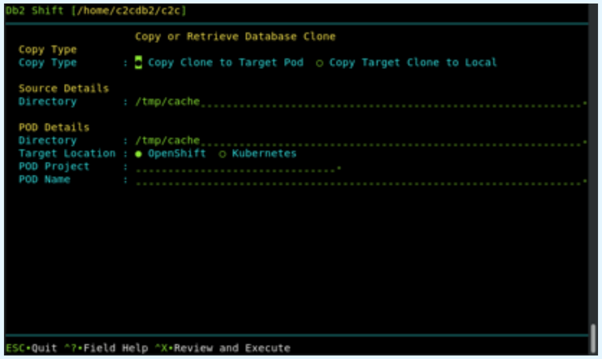


db2shift --mode=sec_monitor --dest-type=POD --oc --kubectl --local --dest-server= pod_name

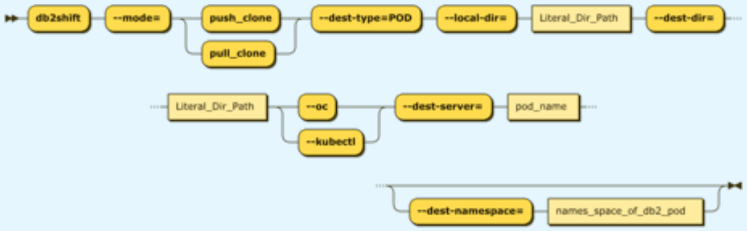
--dest-dbname= destination_database_name --dest-namespace= names_space_of_db2_pod

Copy a Clone to a POD

- This Db2 Shift command provides a feature that allows a user to copy an existing database clone copy to a POD, or to retrieve a database clone from a POD
- Once a database clone has been generated, the copy can be moved to any location and then deployed locally
  - This option provides a convenient way of copying the database using Db2 Shift without having to use OpenShift or Kubernetes commands
- The panel requires the following information:
  - Type of copy (From Source to Target or Target to Source)
  - Source cloned database directory
  - Target cloned database directory
  - The destination POD and server details

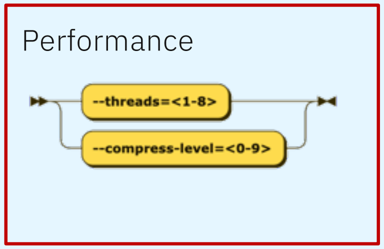


```
Db2 Shift [/home/c2cdb2/c2c]

                   Copy or Retrieve Database Clone
  Copy Type
  Copy Type       : ● Copy Clone to Target Pod  ○ Copy Target Clone to Local

  Source Details
  Directory       : /tmp/cache_______________________________________________

  POD Details
  Directory       : /tmp/cache_______________________________________________
  Target Location : ● OpenShift  ○ Kubernetes
  POD Project     : ______________________________
  POD Name        : __________________________________________________________

ESC•Quit ^?•Field Help ^X•Review and Execute
```



Db2 Shift Performance

- There are two settings which will impact the performance of a Shift operation:
  - Threads - the amount of parallelism that Db2 Shift will use
  - Compression - How much compression will be applied to the data stream during shifts
- The impact of these two parameters can be substantial
- Deciding what setting to use is based on network speed, CPU capacity and overall load on the source system

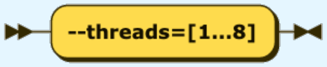


Performance

--threads=<1-8>

--compress-level=<0-9>

Threading

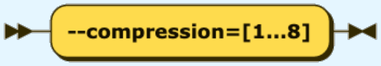
- Syntax: 

--threads=[1...8]

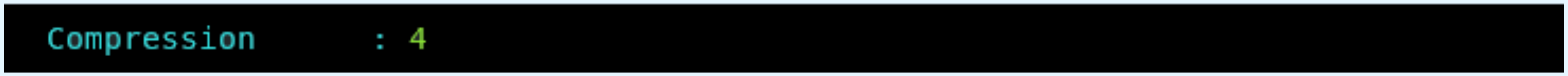
```
Thread count      : 4
```

- The copy phase of the Db2 Shift program can use multiple threads to transmit data to a destination
- This setting allows you to increase the parallelism up to 8 threads
- As you increase the number of threads, the amount of data being transmitted increases, at the expense of greater CPU usage and network congestion
- The default value is 4 which strikes a balance between overhead and network performance

Compression

- Syntax: 

--compression=[1...8]



```
Compression        : 4
```

- RSYNC compresses the data during the transfer process to allow for faster movement of data
- The amount of compression can be adjusted from 0 to 9 with 0 turning off compression and values between 1 and 9 increasing the amount of compression applied to the data
  – A value of 4 has been found to be a good compromise between compression overhead and data size on slow networks (< 1Gb/s)
  – For high-speed networks, a value of 0 is recommended unless there is a requirement to reduce network traffic
- Higher compression values will result in more CPU usage and may not significantly reduce the size of the data stream

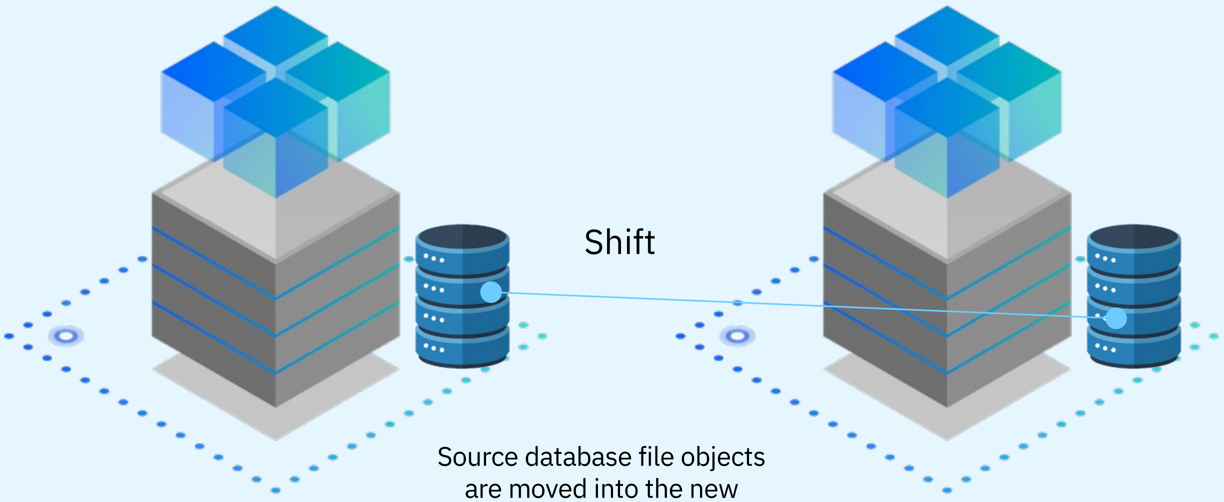
Threading Benchmark

- The following system was created on AWS to test the transfer from a native Db2 instance to an EKS (Kubernetes) cluster that contained a Db2u pod
  - Db2 Shift Scenario: Db2 on-premise to Db2 on Kubernetes (POD) with no compression

Table	Size
Database Size (RAW)	100Gb
Customers	2000000
Addresses	941566
Customer Demographics	1920800
Date Dimensions	73049
Household Demographics	7200
Income Bands	20
Items	202960
Promotions	1000
Stores	394
Store Sales	262185014
Time Dimensions	36000

Db2 CQW Database
m5a.2xlarge
8 VPC 32Gb
CentOS 7 AMI
Db2 11.5.4

EKS Cluster
Db2 Worker Node
r5b.4xlarge
16 VPC 128Gb
Db2 11.5.7

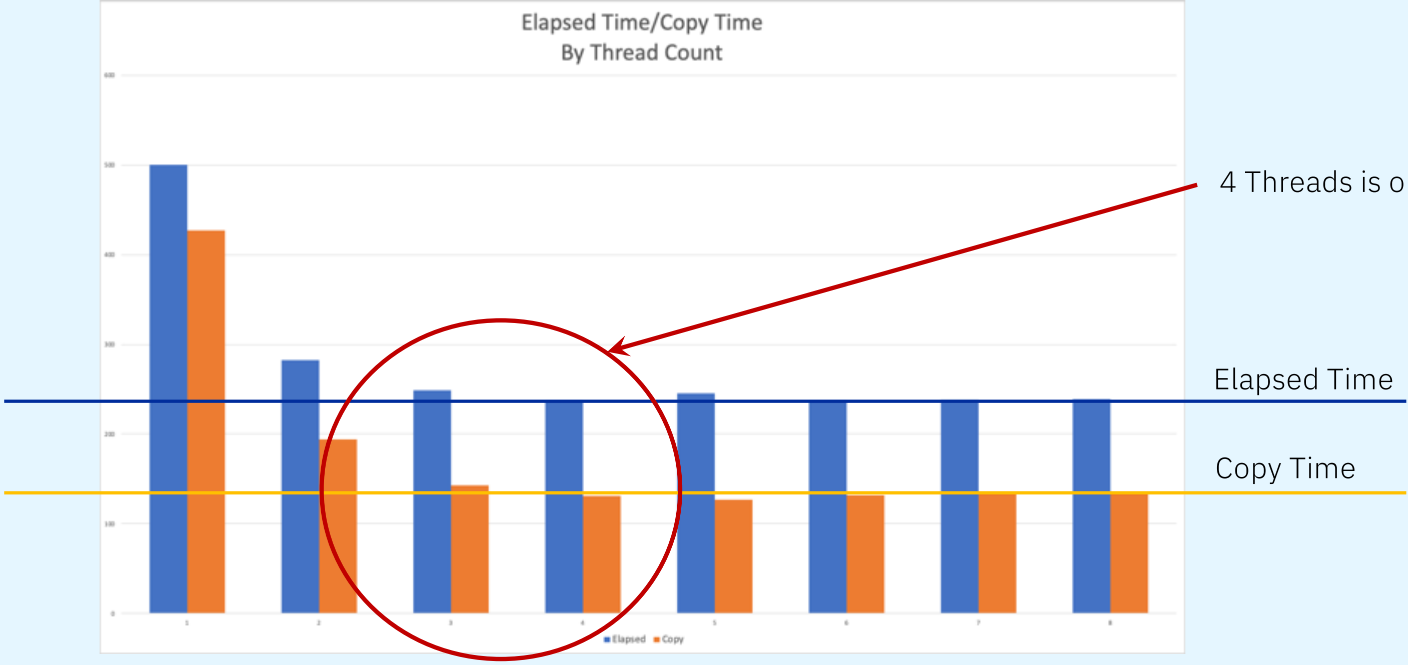


Shift

Source database file objects
are moved into the new
location intact

Elapsed Time

- 289M rows of transactional database with database size of 50Gb
- Best elapsed time 237 secs, with 131 sec for the data transfer
- Why 4 threads?



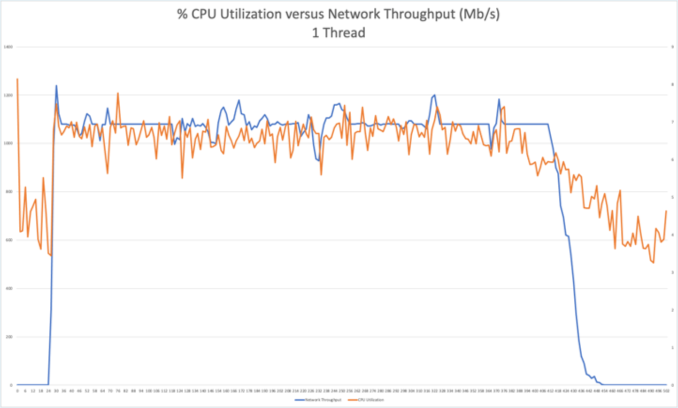
4 Threads is optimal

Elapsed Time

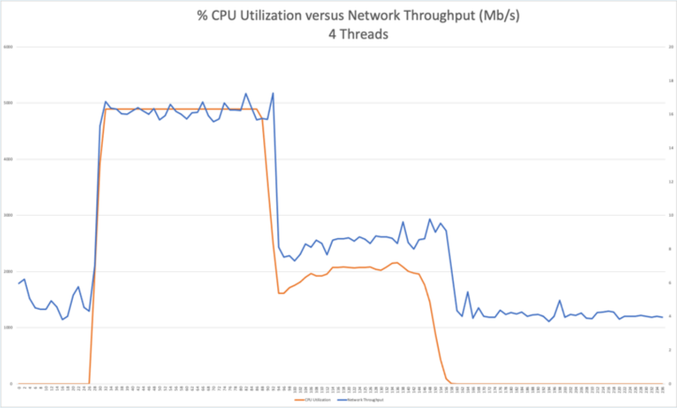
Copy Time

Combined CPU and Network

- CPU Usage is directly correlated to Network Throughput



1 Thread



4 Threads

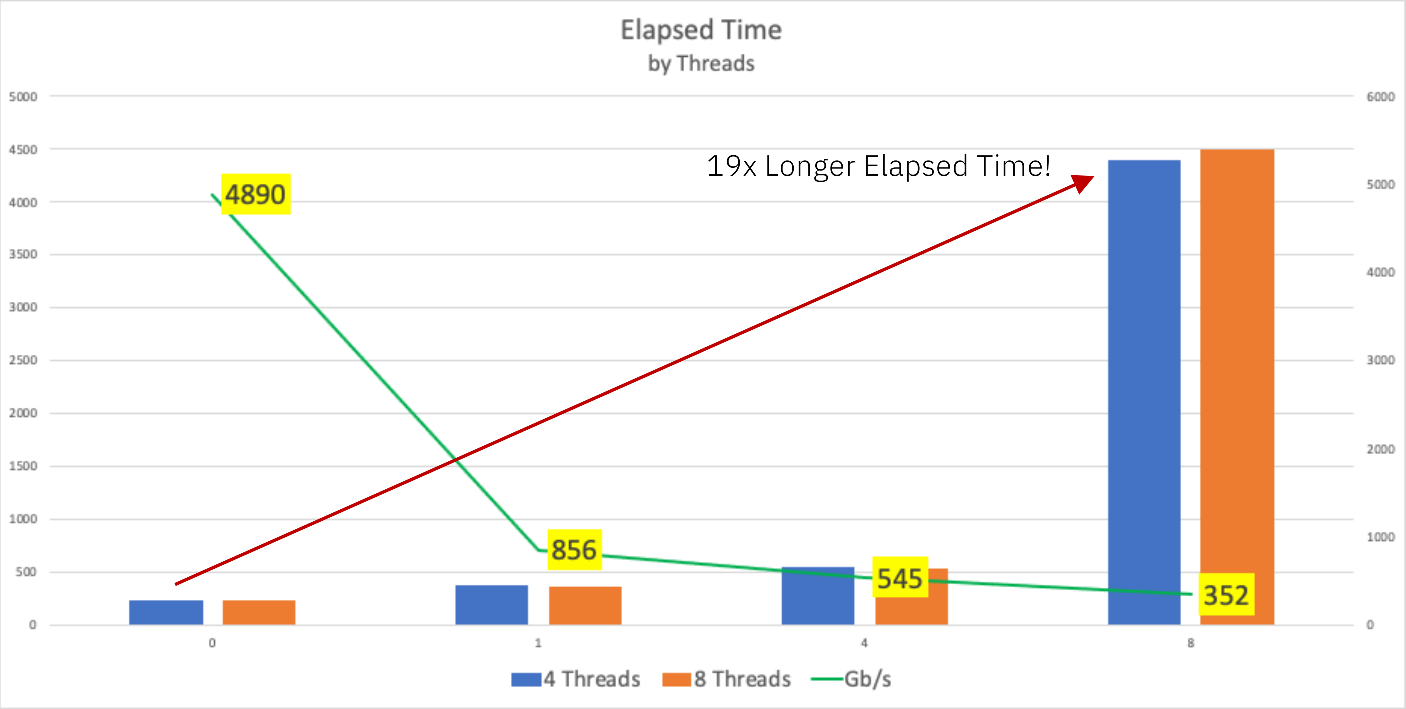
- Maximum network throughput
  – 1 thread was 1240 Mb/sec
  – 4 threads was 4890 Mb/sec
- Throughput was capped by processor speed and network limit of 5Gb/s

Threading Conclusion

- A CPU thread has a limit on how much data it can push onto the network
  - Running a test on a single thread will determine how many cores you can effectively use during a Shift run
- Divide the network capacity by the single core performance to determine the optimal number of threads to use
- Example:
  - Throughput of one thread is approximately 1.2Gb/s
  - Network limit is 5Gb/s
  - 5/1.2 is approximately 4 threads
- This can also be used to determine your total copy time
  - Database Size/(Network Limit/8) = elapsed time
  - 50 GB/(5Gbs/8) = 50GB/(.625GBs) = 80s with ideal conditions
  - Tests results were 50GB/(4.89Gbs/8) = 128s ideal (131 observed)

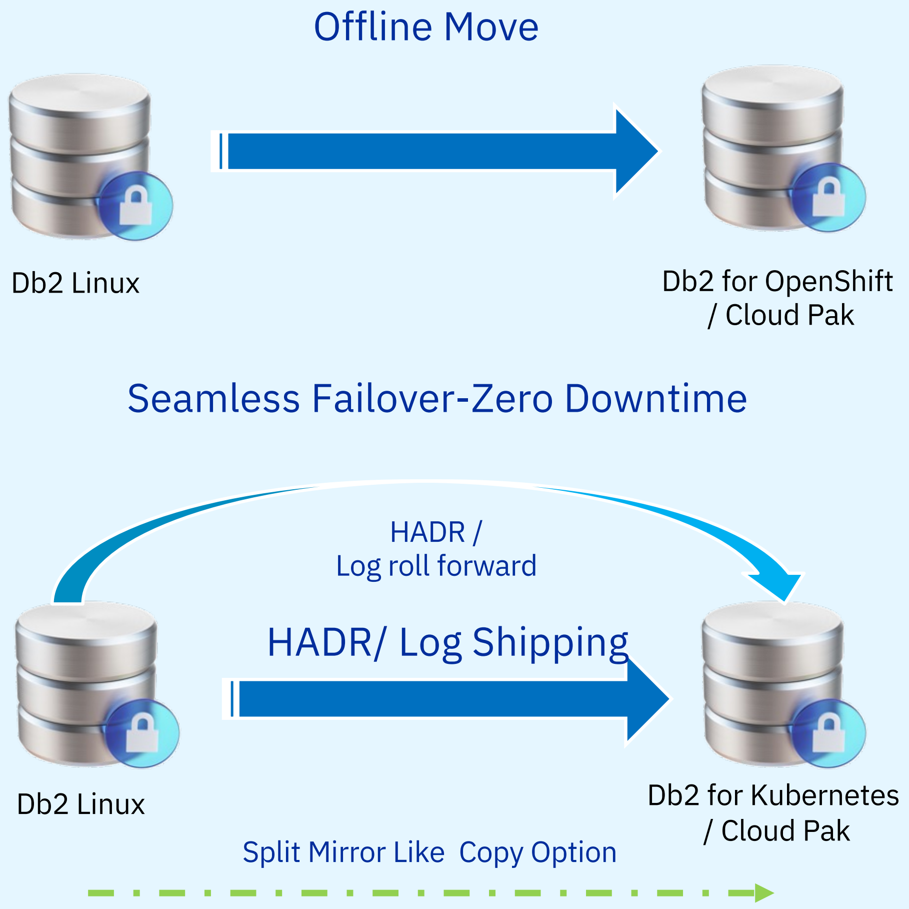
Compression

- The Compression setting has a negative impact on elapsed time
  - It will reduce the amount of network traffic but at the expense of elapsed time and CPU usage

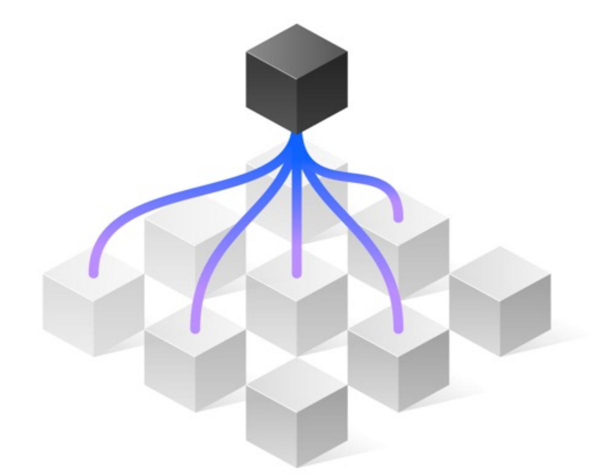


Summary

- Moves Databases to and from:
  - Normal Db2 Instance
  - Containerized Db2 Instance
- Support Multi-tenant Db2
- Supports offline Clones
- Inflight Version Upgrade
- Db2 Warehouse / OLTP
- Volumes = GBs to Terabytes
- Cloud or Local Deployments
  - IBM Cloud Pak For Data Deployments
  - Kubernetes
  - OpenShift
- Tunable Transportation


Offline Move
Db2 Linux
Db2 for OpenShift / Cloud Pak
Seamless Failover-Zero Downtime
HADR / Log roll forward
HADR/ Log Shipping
Db2 Linux
Db2 for Kubernetes / Cloud Pak
Split Mirror Like Copy Option

Resources



Db2 Click to Containerize

Visit ibm.biz/c2cdownload for more details on the Technical Preview of Db2 Click to Containerize

Documentation is available on ibm.biz/c2cebook (PDF) or ibm.biz/c2cguide (HTML)

Self-paced lab is available for business partners and IBMers at ibm.biz/c2cv2lab. Lab details are described in the following PDF ibm.biz/c2ck8slab

Single page fact sheet can be found at ibm.biz/c2cfactsheet

Thank you

© Copyright IBM Corporation 2022

IBM Corporation
New Orchard Road, Armonk, NY 10504

Produced in the United States of America
March 2021

IBM, the IBM logo, ibm.com, IBM Cloud, IBM Cloud Pak, Db2, InfoSphere, DataStage, Cognos and IBM Watson are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Red Hat®, and OpenShift® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions. It is the user’s responsibility to evaluate and verify the operation of any other products or programs with IBM products and programs. THE INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT. IBM products are warranted according to the terms and conditions of the agreements under which they are provided.

The client is responsible for ensuring compliance with laws and regulations applicable to it. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the client is in compliance with any law or regulation.

Statement of Good Security Practices: IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure, and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a lawful, comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY