



# IDUG *VIRTUAL*

2021 EMEA Db2 Tech Conference

## Running Db2 in Containers: Tips and Tricks from the Real World

Ember Crooks

# Agenda

- What containerization means
- Advantages and disadvantages of running Db2 in containers
- Review associated technologies
  - Docker
  - Kubernetes
  - Helm Charts and YAML
- Overview of tools that are often needed along with containerization
- Lessons learned about Db2 in containers in the real world



# Basics: Db2 in a Container

# Install Docker Desktop

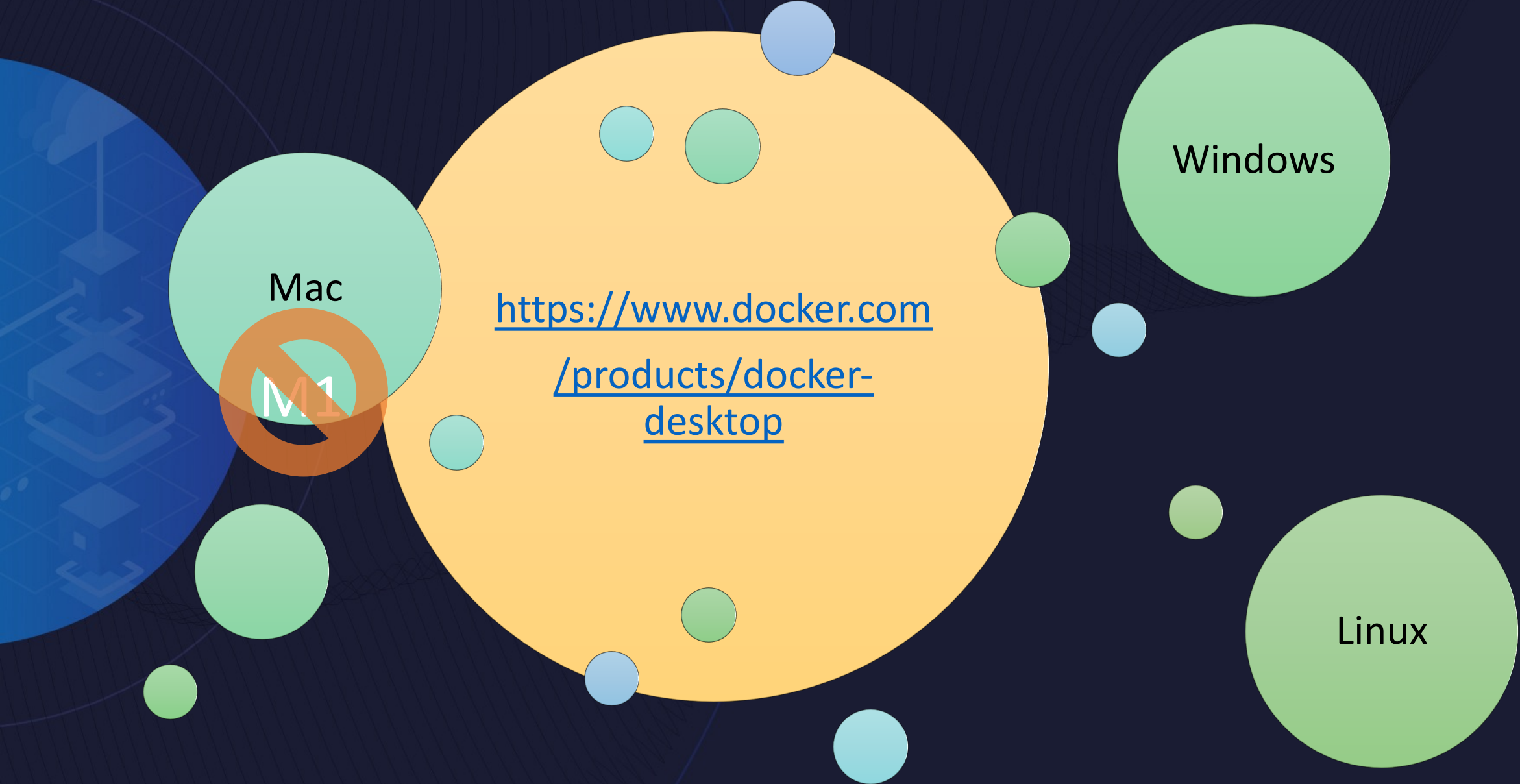
Mac

M1

[https://www.docker.com  
/products/docker-  
desktop](https://www.docker.com/products/docker-desktop)

Windows

Linux





# List and Pull Docker Images

```
$ docker image list
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
$ docker pull ibmcom/db2
Using default tag: latest
latest: Pulling from ibmcom/db2
93156a512b98: Pull complete
f8c518873786: Pull complete
5d4974261da2: Pull complete
2d3a12d55319: Pull complete
d8d137bd0181: Pull complete
0b0c43213599: Pull complete
650e3bc372c5: Pull complete
8e1c790df7a2: Pull complete
397ac3fddb7e: Pull complete
37df0a98c95f: Pull complete
ebdee5ddf728: Pull complete
a7b63a97ead4: Pull complete
654ebc840f5c: Pull complete
Digest: sha256:54355ddc5d8e5b890141ff863083fb3e37168fed8d66bbc4cdf6b73b704d4389
Status: Downloaded newer image for ibmcom/db2:latest
docker.io/ibmcom/db2:latest
$ docker image list
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
ibmcom/db2    latest    a6a5ee354fb1   2 weeks ago   2.95GB
$ █
```

# Docker Run Command for Db2

```
docker run
  -itd
  --name mydb2
  --privileged=true
  -p 50000:50000
  -e LICENSE=accept
  -e DB2INST1_PASSWORD=db2rock$
  -e DBNAME=testdb
  -v /Users/ecrooks/db:/database
  ibmcom/db2
```

# Run Docker Container

```
$ docker run -itd --name mydb2 --privileged=true -p 50000:50000 -e LICENSE=accept -e DB2INST1_PASSWORD=db2rock$ -e DBNAME=testdb -v /Users/ecrooks/db:/database ibmcom/db2 1db2fc367b336f19a9aae91358a1fa7de8d84e7ecc6928696ff53a0154e66b21
```

```
$ docker logs mydb2 --follow
```

```
(*) Previous setup has not been detected. Creating the users...
```

```
(*) Creating users ...
```

```
useradd: warning: the home directory already exists.
```

```
Not copying any file from skel directory into it.
```

```
useradd: warning: the home directory already exists.
```

```
Not copying any file from skel directory into it.
```

```
(*) Preparing the environment before updating the instance ...
```

```
(*) Fixing /etc/services file for DB2 ...
```

```
(*) Fixing db2nodes file configuration ...
```

```
08/26/2021 19:53:35      0      0    SQL1032N  No start database manager command was issued.
```

```
SQL1032N  No start database manager command was issued.  SQLSTATE=57019
```

```
(*) Creating instance ...
```

```
DBI1446I  The db2icrt command is running.
```

```
...
```

```
DB2 State : Available
```

```
DB2 has been started
```

```
ssh-keygen: generating new host keys: RSA1 RSA DSA ECDSA ED25519
```

```
/var/db2_setup/include/db2_common_functions: line 539: /usr/bin/supervisord: No such file or directory
```

```
(*) All databases are now active.
```

```
(*) Setup has completed.
```

# Listing and Entering Docker Containers

```
$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
1db2fc367b33	ibmcom/db2	"/var/db2_setup/lib/..."	7 minutes ago	Up 7 minutes	22/tcp, 55000/tcp, 60006-60007/tcp, 0.0.0.0:50000->50000/tcp, :::50000->50000/tcp
mydb2					

```
$ docker exec -it mydb2 bash -c "su - db2inst1"
```

```
Last login: Thu Aug 26 19:54:35 UTC 2021 on pts/0
```

```
[db2inst1@1db2fc367b33 ~]$
```





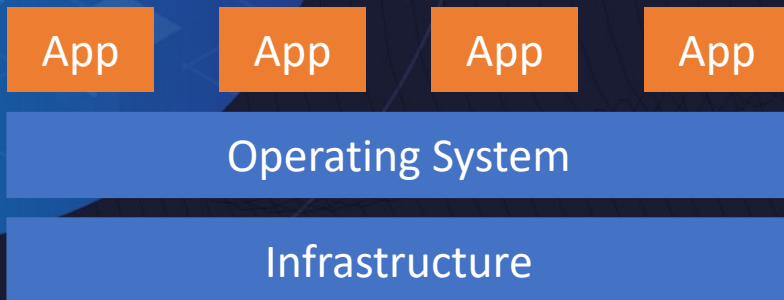
# What Containerization Means

# What is a Container?

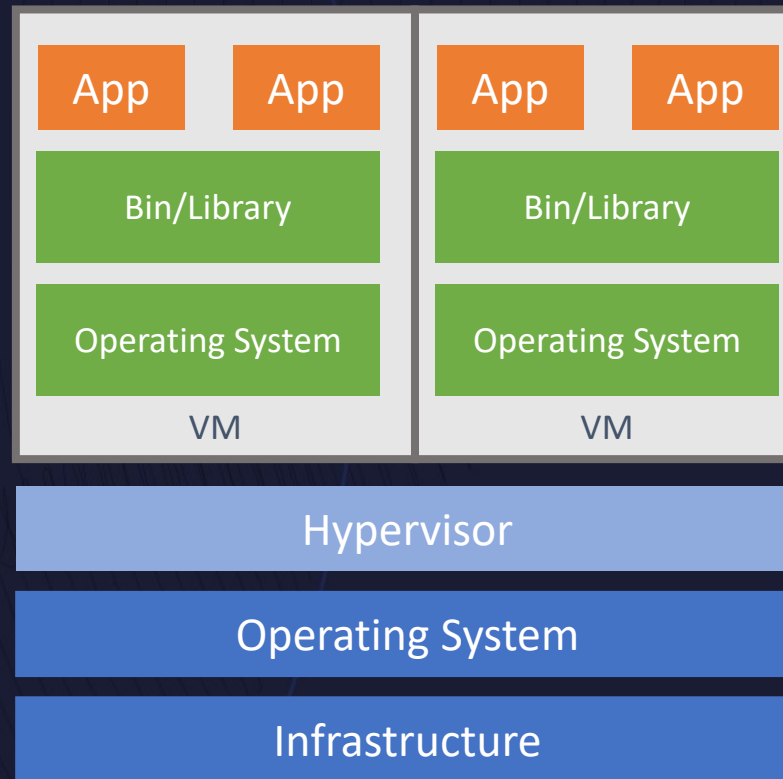
Isolated area of an OS with resource usage limits



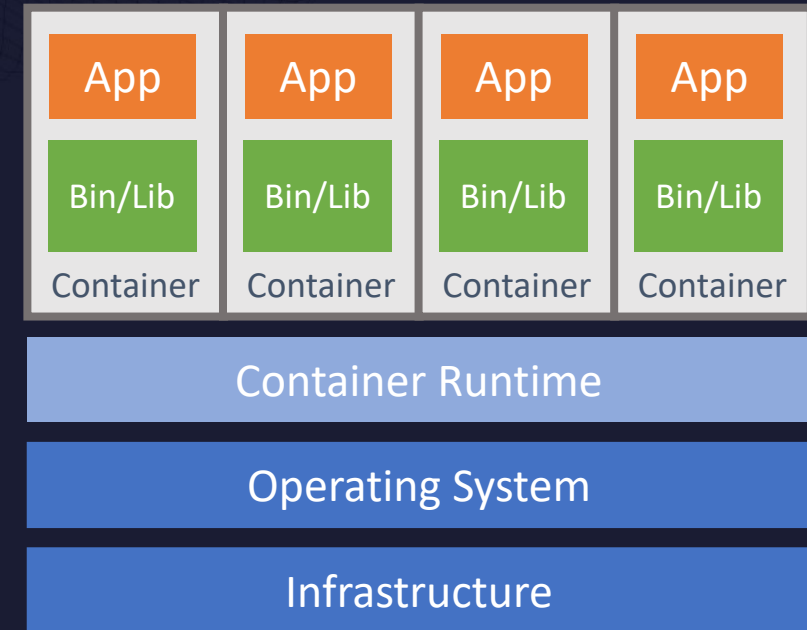
## Bare Metal



## Virtual Machines



## Containers



# Containers and Persistence

Container1

Container2

Container3

Container4

Container5

Container6

# Concepts of Containers



## Image

- Read-Only Files
- JSON Manifest



## Container



## Container



## Container



# Containerizing Db2



## Reasons to Containerize

- New environments
- Integration into DevOps
- Fast fix packs and upgrades
- Build similar databases at scale



## Reasons NOT to Containerize

- Resources on hosts
- Critical performance databases
- Skills gap in DBAs
- Lack of production support

# Where do Db2 Containers Come From?

## Db2 Community Container

[ibmcom/db2](https://ibmcom/db2)

- Limitations:
  - Privileged mode
  - One per host
  - Not for prod



## Db2u

- Truly cloud native
- Many containers working together
- Only available on OpenShift

EKS and  
AKS in  
2022



## Create Your Own

- Complex
- Flexible
- Learn lessons the hard way
- Not for prod



# Running Db2 in Containers for Production

- IBM has announced they will offer support for db2u on AKS and EKS in the first half of this year, with the possibility of other environments such as native Kubernetes, Google Cloud containers, and Rancher to follow.
- For production, the only way Db2 in containers is supported is on OpenShift



# Details on db2u



# Db2u

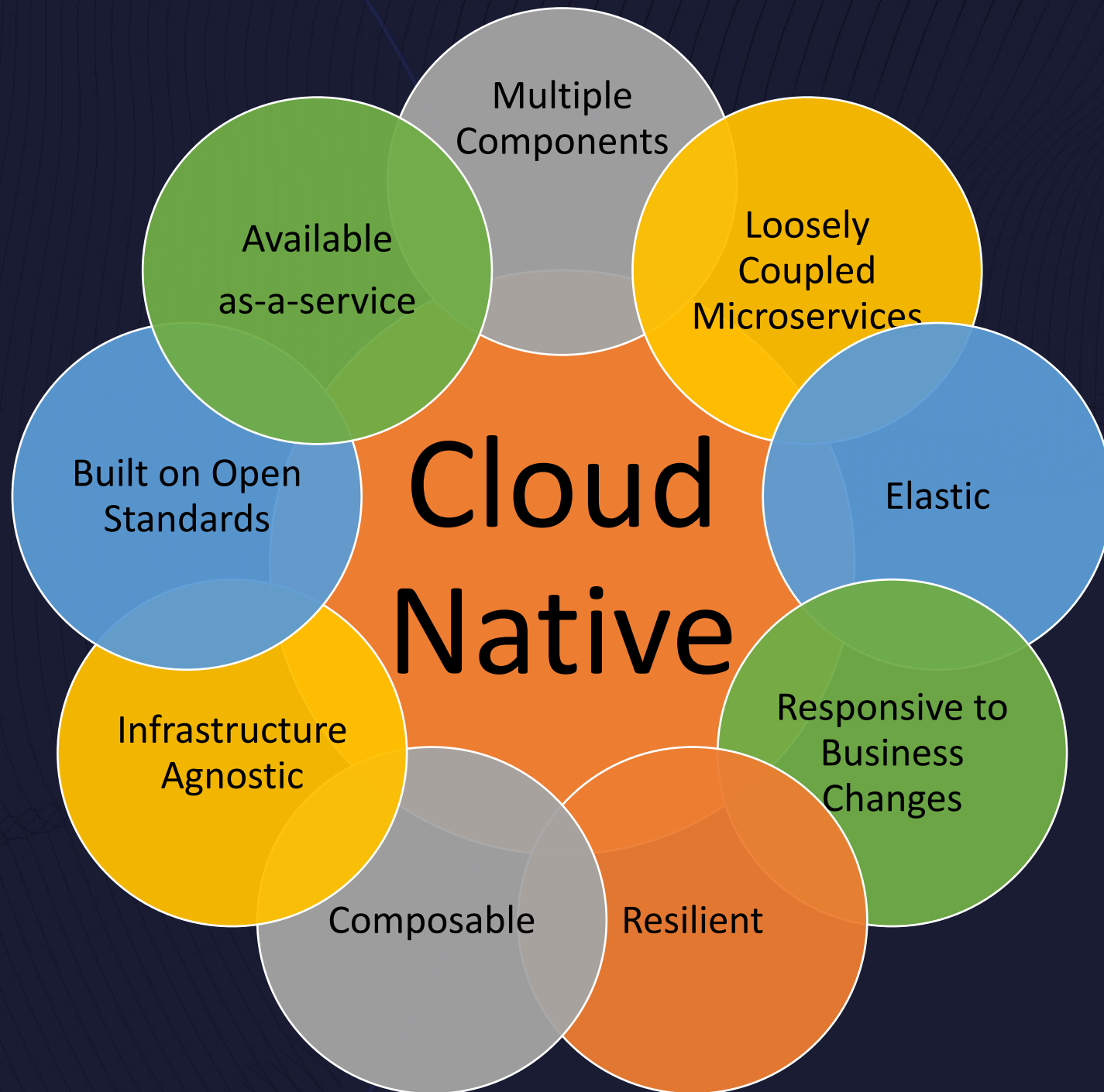
- Operator

- Custom Kubernetes controller that uses custom resources (CR) to manage applications and their components
- Monitor application as it runs, and can back up data, recover from failures, and upgrade the application over time, automatically
- Interaction via API

- Containers

- Db2
- Database
- Console
- LDAP





# db2u Technical Preview

- Works on Amazon EKS and MiniKube(and with modifications on k3d)
- Described here: <https://medium.com/@baheer/db2-on-kubernetes-8d715546f586>
- Tech Preview
  - Not supported for production yet
  - No full documentation (Refer to [documentation for db2u on OpenShift](#))
  - Suitable for limited PoC

# Db2u Tech Preview High Level Steps

- Install a version of K8's that works on the target environment
- Pull and install the Operator Lifecycle Manager
- Create namespace, and pull and instantiate the db2u operator
- Configure the YAML for the db2u CRD
- Use the db2u CRD YAML to spin up a db2 instance and database
- Set details for database connection
- Connect to database and use normally



# Advantages and Disadvantages of Running Db2 in Containers



# Common Container Use Cases



## “Database Server”

Serves the role of a VM or Server



## Maintenance

Client with scripts for maintaining databases



## Developer Local

Representative data/structure developers can run locally



## DBA Testing

Developer local or generic container for POC and experimentation



# Why Containerize Db2?

Easy for application team  
to spin up new  
environments



Build similar databases at  
scale



Speed of Fix Packs and  
Upgrades



Easy integration into  
DevOps

# Why NOT Containerize Db2?

More resources than  
a host/node



Limited options for  
prod



Lack of DBA skills



This DB is a pet, not  
cattle



# Review of Technologies Associated with Containerization

Docker

Kubernetes

Helm Charts and Yaml

# Kubernetes



Often used with orchestrator such as Rancher or OpenShift



Group services, including a db service together



Manages common things: Storage, Secrets, and Network



Abbreviated K8s

# Helm Charts



Used to define services and groups of related services



Consists of layers of text files which define properties



Example:

db2-chart

wcs9-chart, which requires db2-chart





# Tools Often Used Along With Containerization

Code Version Control

Database Version Control

Automation

# Code Version Control

Examples:



- GitHub
- Git
- SVN



Changes to:



- Infrastructure code
- Automation code
- Monitoring code

# Example of Files Changed: Changing Backup Methodology

- Container build script *x 5*
- Db2 helm chart *x 4*
- App helm chart *x 3*
- Secrets added (manually – x12)
- Script in Db2 maintenance container
- New Jenkins file to run modified backup syntax
- Jenkins DSL file to use the New Jenkins file *x 3*

*= 17!*

# GitHub Basics

## Repository (Repo)

Related set of code that is built/deployed together

## Branch

Different features or environments of the repo's code

## Fork

A repo that is a personal copy of the code

## Commit

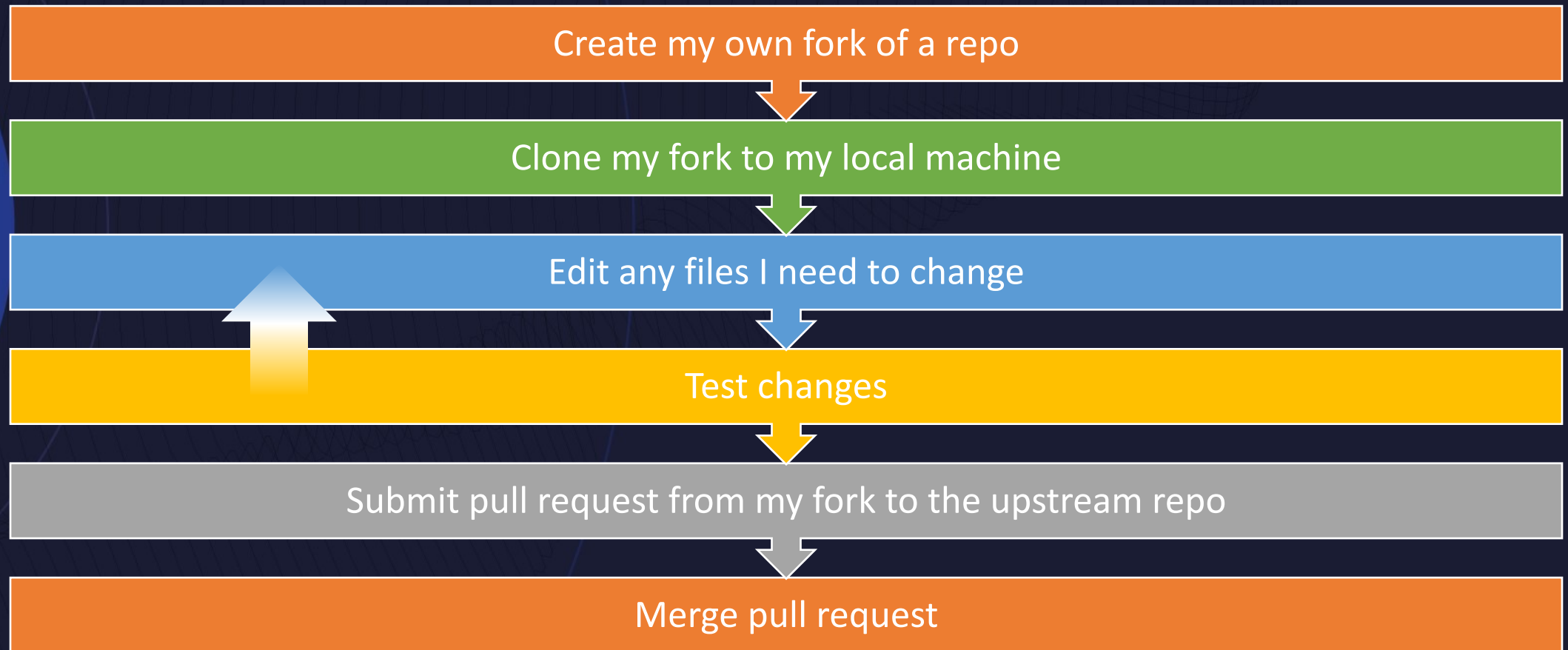
Taking code from one repo and adding it to another

## Pull Request

Asking to integrate code into a repository



# GitHub Process



# GitHub Resources

[GitHub desktop](#)

[GitHub basics](#)

[Great basic best practices video](#)

[Git game](#)

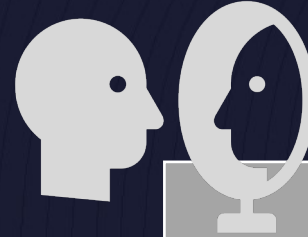
# Database Version Control

## Examples:



- Liquibase
- Flywheel

## Purpose:



- Make all database structure and ddl changes
- Couple database structure/ddl changes with code deploys
- Manage rollback of database changes
- Ensure the schema of related databases are in sync
- Manage the data of configuration tables

# Sample Liquibase SQL File (Data)

```
--liquibase formatted sql
--changeset ecrooks:xdbmaintconf runOnChange:true
-- This SQL file contains all entries to insert into xdbmaintconf

--changeset ecrooks:xdbmaintconf_allwcs context:"PRJ-WCS-WCS9-DEV, PRJ-WCS-WCS9-QA" runOnChange:true
MERGE
INTO dbamon.xdbmaintconf s
USING
    ( SELECT maint_category, object, type, sequence, check_query, thresh_direction, thresh_type, num_days, num_thresh, schema FROM (
        -- dbclean
        SELECT 'dbclean' maint_category, rtrim(OBJECTNAME) object, rtrim(TYPE) type, rtrim(SEQUENCE) sequence,
        replace(replace(statement,'delete from', 'select count(*) from'), 'DELETE FROM', 'select count(*) from') check_query, '='
        thresh_direction, 'retention' thresh_type, 2+1 num_days, 0 num_thresh, 'WSCOMUSR' schema from wscomusr.cleanconf where
        objectname='address' and type='obsolete' UNION ALL

        -- Start: last dummy query so we dont have to worry about ensuring last row does not have 'UNION ALL', insert/update rows above
        SELECT NULL maint_category, NULL object, NULL type, NULL sequence, NULL check_query, NULL thresh_direction, NULL thresh_type,
        NULL num_days, NULL num_thresh, NULL schema FROM sysibm.sysdummy1 WHERE 1=0 -- evals to false to return 0 rows
        -- End: last dummy query so we dont have to worry about ensuring last row does not have 'UNION ALL')
    ) t
ON
    ((s.maint_category = t.maint_category AND s.object = t.object AND s.type = t.type AND s.schema = t.schema AND s.sequence =
    t.sequence)
    OR
    (s.maint_category = t.maint_category AND s.schema = t.schema AND s.sequence = t.sequence AND s.object is NULL AND s.type is NULL))
WHEN MATCHED THEN
    UPDATE SET
        s.check_query      = t.check_query,
        s.thresh_direction = t.thresh_direction,
        s.num_days         = t.num_days,
        s.num_thresh       = t.num_thresh
WHEN NOT MATCHED THEN
    INSERT (s.maint_category, s.object, s.type, s.check_query, s.thresh_direction, s.sequence, s.thresh_type, s.num_days, s.num_thresh, s.schema)
    VALUES (t.maint_category, t.object, t.type, t.check_query, t.thresh_direction, t.sequence, t.thresh_type, t.num_days, t.num_thresh, t.schema)
;
--rollback select null from sysibm.sysdummy1;
```



# Sample Liquibase SQL File (Create Object)

```
--liquibase formatted sql

--changeset ecrooks:xdbmaintconf_create

create table dbamon.xdbmaintconf (
    maint_id integer generated by default as identity
    , maint_category VARCHAR(40) NOT NULL
    , object VARCHAR(40)
    , type VARCHAR(40)
    , sequence smallint NOT NULL default 0
    , check_query VARCHAR(4000) NOT NULL
    , thresh_direction CHAR(2) NOT NULL
    , thresh_type VARCHAR(40) NOT NULL
    , num_days bigint
    , num_thresh int NOT NULL
    , schema VARCHAR(128)
    , description generated always as (CASE
        WHEN thresh_type='existence' then 'Verify
there are ' || rtrim(thresh_direction) || ' ' || rtrim(char(num_thresh)) || ' records within the last '
|| rtrim(char(num_days)) || ' days for ' || coalesce( rtrim(maint_category) || ' ' || rtrim(object) || '
' || rtrim(type), rtrim(maint_category) || ' ' || rtrim(type), rtrim(maint_category)) || '.'
        WHEN thresh_type='retention' then 'Verify
there are ' || rtrim(char(num_thresh)) || ' records or less which are older than ' ||
rtrim(char(num_days)) || ' days for ' || coalesce( rtrim(maint_category) || ' ' || rtrim(object) || ' '
|| rtrim(type), rtrim(maint_category) || ' ' || rtrim(type), rtrim(maint_category)) || '.' end)
    , PRIMARY KEY (maint_id)
);

create unique index dbamon.ix_xdbmaintconf01 on dbamon.xdbmaintconf ( maint_category, sequence, object,
type, schema ) allow reverse scans;

--changeset ecrooks:xdbmaintconf_create_correction
alter table dbamon.xdbmaintconf alter column schema set default 'DBAMON';
alter table dbamon.xdbmaintconf alter column schema set not null;
CALL SYSPROC.ADMIN_CMD ('REORG TABLE dbamon.xdbmaintconf');
```



# Sample Liquibase SQL File (Conditional Create Object)

```
--liquibase formatted sql
--changeset ecrooks:lockevmon_bp runOnChange:false
--preconditions onFail:MARK_RAN onError:HALT
--precondition-sql-check expectedResult:0 SELECT COUNT(*) FROM
syscat.bufferpools WHERE bpname = 'BUFF32K'
CREATE BUFFERPOOL BUFF32K DEFERRED SIZE AUTOMATIC PAGESIZE 32
K;

--changeset ecrooks:lockevmon_ts runOnChange:false
--preconditions onFail:MARK_RAN onError:HALT
--precondition-sql-check expectedResult:0 SELECT COUNT(*) FROM
syscat.TABLESPACES WHERE tbspace = 'DBA32K'
CREATE LARGE TABLESPACE DBA32K PAGESIZE 32 K BUFFERPOOL
BUFF32K;

--changeset ecrooks:lockevmon runOnChange:false
--preconditions onFail:MARK_RAN onError:HALT
--precondition-sql-check expectedResult:0 SELECT COUNT(*) FROM
syscat.EVENTMONITORS WHERE EVMONNAME='DBA_DEADLOCK'
CREATE EVENT MONITOR DBA_DEADLOCK FOR LOCKING WRITE TO
UNFORMATTED EVENT TABLE (TABLE DBAMON.DBA_DEADLOCK IN DBA32K)
AUTOSTART;
COMMIT;
set event monitor DBA_DEADLOCK state=1;
```

# Sample Liquibase SQL File (Grants)

```
--liquibase formatted sql
```

```
--changeset ecrooks:grant_friday context:"PRJ-WCS-WCS9-DEV,  
PRJ-WCS-WCS9-QA, PRJ-WCS-WCS9-STAGE, PRJ-PRJ-WCS9-LT, PRJ-WCS-  
WCS9-PROD" runOnChange:true
```

```
-- This SQL file contains all entries to grant permissions to  
somebot
```

```
-- somebot is a member of SYSMON_GROUP
```

```
grant connect on database to user somebot;  
grant select on schema.orders to user somebot;  
grant select on schema.users to user somebot;  
grant select on schema.member to user somebot;  
grant select on schema.EMSPOT to user somebot;  
grant select, update on schema.DMEMSPOTDEF to user somebot;  
grant select on schema.address to user somebot;  
grant select on schema.userdemo to user somebot;  
grant select on schema.orderitems to user somebot;  
grant select on schema.stlocattr to user somebot;
```

# Automation

## Example



- Jenkins
- GitLab
- Ansible

## Purpose:




- Schedule regularly-occurring tasks like database maintenance
- Allow others to trigger database jobs (maintenance, data loads)
- Provide a centralized location to review success of all jobs



# Jenkins

## (1|3)

**Jenkins**

3

search

Ember Crooks | log out

Jenkins DatabaseMaintenance [ENABLE AUTO REFRESH](#)

Up

Status

Configure

New Item

Delete Folder

People

Build History

Project Relationship

Check File Fingerprint

Move

Rename

Config Files

Credentials

New View

Build Queue

No builds in the queue.









Build Executor Status

master




DatabaseMaintenance

[add description](#)

All +


S	W	Name ↓	Last Success	Last Failure	Last Duration	Built On
		DB2	N/A	N/A	N/A	
		<a href="#">db2db-dbclean-dsl</a>	8 days 19 hr - <a href="#">#25</a>	N/A	2.9 sec	 <a href="#">docker-agent8</a>
		<a href="#">db2db-maintenance-dsl</a>	20 days - <a href="#">#85</a>	N/A	7.1 sec	 <a href="#">docker-agent9</a>
		<a href="#">db2db-maintmon-dsl</a>	10 mo - <a href="#">#28</a>	N/A	3.2 sec	 <a href="#">docker-agent6</a>
		<a href="#">db2db-oms-maintmon-dsl</a>	5 mo 15 days - <a href="#">#9</a>	N/A	6 sec	 <a href="#">docker-agent7</a>
		<a href="#">db2db-oms10maint-dsl</a>	20 days - <a href="#">#57</a>	N/A	8.9 sec	 <a href="#">docker-agent1</a>
		<a href="#">db2db-reporting-dsl</a>	4 mo 13 days - <a href="#">#2</a>	N/A	4.2 sec	 <a href="#">docker-agent3</a>
		<a href="#">db2db-spgnmaint-dsl</a>	21 days - <a href="#">#2</a>	N/A	2.5 sec	 <a href="#">docker-agent11</a>
		MariaDB	N/A	N/A	N/A	
		<a href="#">mariadb-maintenance-dsl</a>	2 mo 22 days - <a href="#">#11</a>	1 yr 5 mo - <a href="#">#6</a>	13 sec	 <a href="#">docker-agent8</a>
		Reporting	N/A	N/A	N/A	

Icon: [S](#) [M](#) [L](#)

[Legend](#)  [RSS for all](#)  [RSS for failures](#)  [RSS for just latest builds](#)


# Jenkins

## (2|3)







 **Jenkins** 3  [Ember Crooks](#) | [log out](#)

Jenkins >  > DatabaseMaintenance > DB2 > [ENABLE AUTO REFRESH](#)




[Up](#)  
[Status](#)  
[Configure](#)  
[New Item](#)  
[Delete Folder](#)  
[People](#)  
[Build History](#)  
[Project Relationship](#)  
[Check File Fingerprint](#)  
[Move](#)  
[Rename](#)  
[Config Files](#)

 **DB2**

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration	Built On
		<b>PRJ1</b>	N/A	N/A	N/A	
		<b>PRJ2</b>	N/A	N/A	N/A	
		<b>PRJ3</b>	N/A	N/A	N/A	

Icon: [S](#) [M](#) [L](#)


[Legend](#)  [RSS for all](#)  [RSS for failures](#)  [RSS for just latest builds](#)

[add description](#)



# Jenkins

## (3 | 3)



# Jenkins

3


search

Ember Crooks | log out

Jenkins DatabaseMaintenance

[ENABLE AUTO REFRESH](#)











































[Up](#)  
[Status](#)  
[Configure](#)  
[New Item](#)  
[Delete Folder](#)



## PRJ3

[add description](#)

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration	Built On
		<a href="#">backup-lt</a>	8 hr 58 min - <a href="#">#10</a>	N/A	10 min	
		<a href="#">backup-prod</a>	8 hr 33 min - <a href="#">#9</a>	N/A	29 min	
		<a href="#">backup-qa</a>	8 hr 18 min - <a href="#">#11</a>	N/A	6 min 3 sec	
		<a href="#">backup-stage</a>	8 hr 59 min - <a href="#">#11</a>	N/A	14 min	
		<a href="#">dbclean-dev</a>	8 hr 59 min - <a href="#">#10</a>	8 days 8 hr - <a href="#">#1</a>	10 min	
		<a href="#">dbclean-lt</a>	10 hr - <a href="#">#9</a>	N/A	3 min 40 sec	
		<a href="#">dbclean-prod</a>	10 hr - <a href="#">#9</a>	N/A	16 min	
		<a href="#">dbclean-qa</a>	10 hr - <a href="#">#9</a>	N/A	5 min 21 sec	
		<a href="#">dbclean-stage</a>	10 hr - <a href="#">#10</a>	N/A	13 min	
		<a href="#">drift-dev</a>	5 hr 18 min - <a href="#">#12</a>	N/A	48 sec	
		<a href="#">drift-lt</a>	5 hr 55 min - <a href="#">#9</a>	N/A	45 sec	
		<a href="#">drift-prod</a>	5 hr 3 min - <a href="#">#9</a>	N/A	46 sec	
		<a href="#">drift-qa</a>	2 days 5 hr - <a href="#">#7</a>	5 hr 37 min - <a href="#">#9</a>	45 sec	
		<a href="#">drift-stage</a>	5 hr 57 min - <a href="#">#9</a>	N/A	45 sec	

New View

Build Queue

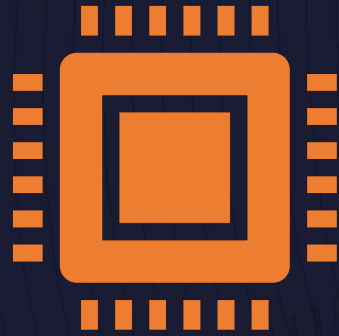
No builds in the queue.

Build Executor Status



# Lessons Learned Running Db2 in (Custom-Built) Containers

# Understand Resource Constraints



CPU



Memory

Host memory  
vs. Container  
memory

The diagram features a central title 'Limit Db2's Use of Memory' surrounded by three colored boxes: an orange box at the top, a yellow box at the bottom left, and a grey box at the bottom right. These boxes are connected by a thin, light-colored circular line. The background is dark blue with a subtle grid pattern and a large blue circular graphic on the left side containing abstract geometric shapes.

## Limit Db2's Use of Memory

May cause  
instance crashes  
at random times

IBM's image  
does this for  
you



Persistence



Stateful Set



Disk



# Build Reusable and Understand Inheritance



Containers



Helm Chart

Storage



Encryption

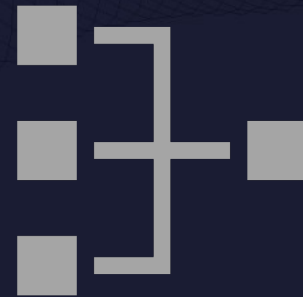


Expanding Volumes

# Use Ephemeral Containers When Possible



Maintenance



Clients

# Resources for Learning about Containers

- Great free interactive tutorials: <https://www.katacoda.com/>





# IDUG *VIRTUAL*

2021 EMEA **Db2** Tech Conference

Ember Crooks  
Sherwin Williams  
[ember.crooks@gmail.com](mailto:ember.crooks@gmail.com)



Please fill out your session evaluation!