**Oracle to Aurora migration (PAuth POC)**

STEP #1 (one-time): In both West/East regions, create new cluster parameter group aurora-postgresql14-cluster-parameters-pauth with family “aurora-postgresql14” and type “DB cluster parameter group”.

Update cluster parameter group with below parameter changes.

rds.logical\_replication = 1 # Enable logical replication by setting the parameter to 1.

pglogical.conflict\_resolution = last\_update\_wins

track\_commit\_timestamp = 1 # needed for last/first update wins conflict resolution

shared\_preload\_libraries = \*\*\*, pglogical # Add *pglogical* to the shared\_preload\_libraries parameter

max\_wal\_senders = 20

timestamp = US/Pacific

STEP #2 (one-time): In both West/East regions, create new instance parameter group aurora-postgresql14-instance-parameters-pauthwith family “aurora-postgresql14” and type “Parameter groups”.

Update instance parameter group with below static parameter change.

shared\_preload\_libraries = \*\*\*, pglogical # Add *pglogical* to the shared\_preload\_libraries parameter

STEP #3: (a) Create one Aurora PostgreSQL (14.3) regional cluster, in East region.

Version Aurora PostgreSQL 14.3

Cluster name: pauth-b-prf-east-poc

Master User: postgres

Master Password: \*\*\*\*\*

Instance class: db.r6g.2xlarge

VPC: select perf vpc

Select rds-ora-idx-b-subnet-group-prf subnet group

Select aurora-postgres-rds-prf-sg security group and remove default security group

Port 1526

Disable Enhanced monitoring

Select above cluster parameter group and instance group.

(b) Modify above cluster and add below tags

intuit:billing:appenv prf

intuit:identity:env prf

intuit:billing:user-app pauth

intuit:identity:app pauth

intuit:billing:component swimlane b

Name pauth-b-prf-east-poc

(c) Modify cluster and update cluster parameter group.

(d) Reboot instance (or) stop and start entire cluster.

STEP #4: (a) Create one Aurora PostgreSQL (14.3) regional cluster, in West region.

Version Aurora PostgreSQL 14.3

Cluster name: pauth-b-prf-west-poc

Master User: postgres

Master Password: \*\*\*\*\*

Instance class: db.r6g.2xlarge

VPC: select perf vpc

Select rds-ora-idx-b-subnet-group-prf subnet group

Select aurora-postgres-rds-prf-sg security group and remove default security group

Port 1526

Disable Enhanced monitoring

Select above cluster parameter group and instance group.

(b) Modify above cluster and add below tags

intuit:billing:appenv prf

intuit:identity:env prf

intuit:billing:user-app pauth

intuit:identity:app pauth

intuit:billing:component swimlane b

Name pauth-b-prf-west-poc

(c) Modify cluster and update cluster parameter group.

(d) Reboot instance (or) stop and start entire cluster.

STEP #5: Enable tunnelling and try to connect to West/East clusters, from PgAdmin tool.

ssh -L 3306:pauth-b-prf-east-poc.cluster-c4ik1tjadzub.us-east-2.rds.amazonaws.com:1526 bastion.identity-prf-us-east-2.a.intuit.com

ssh -L 5433:pauth-b-prf-west-poc.cluster-cjzdkv6oy3dy.us-west-2.rds.amazonaws.com:1526 bastion.identity-prf-us-west-2.a.intuit.com

( or )

Login into databases, from Perf OGG hub

eiamCli login; eiamCli getAWSTempCredentials -a 181299031239 -r AppOps -o 1 -p default; eiamCli aws\_ssh -a 181299031239 -p id\_rsa.pub -d ~/.ssh/

psql -h pauth-b-prf-east-poc.cluster-c4ik1tjadzub.us-east-2.rds.amazonaws.com -p1526 -U postgres -d postgres

psql -h pauth-b-prf-west-poc.cluster-cjzdkv6oy3dy.us-west-2.rds.amazonaws.com -p1526 -U postgres -d postgres

STEP #6: Create replication user, in all clusters.

On West/East databases,

create user pguser with password 'pguser123';

grant rds\_superuser to pguser;

STEP #7: On East database, create pglogical extension and create node. And create replication set.

create extension pglogical;

select pglogical.create\_node(node\_name :='pauthnode-perfeast', dsn := 'host=pauth-b-prf-east-poc.cluster-c4ik1tjadzub.us-east-2.rds.amazonaws.com port=1526 dbname=postgres user=pguser password=pguser123');

select pglogical.create\_replication\_set('pauth\_east\_west\_repl\_set');

select pglogical.replication\_set\_add\_all\_tables('pauth\_east\_west\_repl\_set', ARRAY['pauthown']);

STEP #8: On West database, create pglogical extension and create provider node. And create subscription.

create extension pglogical;

select pglogical.create\_node(node\_name :='pauthnode-perfwest', dsn := 'host=pauth-b-prf-west-poc.cluster-cjzdkv6oy3dy.us-west-2.rds.amazonaws.com port=1526 dbname=postgres user=pguser password=pguser123');

SELECT pglogical.create\_subscription(

subscription\_name := 'pauth\_east\_b\_to\_west\_b\_sub',

provider\_dsn := 'host=pauth-b-prf-east-poc.cluster-c4ik1tjadzub.us-east-2.rds.amazonaws.com port=1526 dbname=postgres user=pguser password=pguser123',

replication\_sets := ARRAY['pauth\_east\_west\_repl\_set'],

synchronize\_data := false,

forward\_origins := '{}' );

STEP #9: Test DMLs - inserts, updates, deletes - in one-way replication (from east to west)

select \* from pauthown.token where TOKEN\_ID='fb12ac3d-5c70-42f6-824a-a4034cf5c4e3';

select \* from pauthown.user\_connection where USER\_CONNECTION\_ID='42383e68-8f0c-426e-99ea-839bd4573d32';

update pauthown.token set LAST\_MOD\_TIME=CURRENT\_TIMESTAMP where TOKEN\_ID in ('fb12ac3d-5c70-42f6-824a-a4034cf5c4e3');

update pauthown.user\_connection set LAST\_MOD\_TIME=CURRENT\_TIMESTAMP where USER\_CONNECTION\_ID='42383e68-8f0c-426e-99ea-839bd4573d32';

select \* from PAUTHOWN.PC\_EVENTS where id='21489';

insert into PAUTHOWN.PC\_EVENTS ("id","EXTERNAL\_SRC\_ID","EXTERNAL\_REF\_ID","EXTERNAL\_REF\_NAME","EVENT\_TYPE","ACTIVE\_TIME","START\_TIME","END\_TIME","STATUS","CREATED","MODIFIED","RETRY\_COUNT","LAST\_MOD\_TIME")

values ('21489','1838','1484','jhon','game','2022-05-16 10:09:10.32442','2022-05-16 08:09:10.32442','2022-05-17 10:09:10.32442','started','2022-05-15 10:09:10.32442','2022-05-19 10:09:10.32442','2','2022-05-20 10:09:10.32442');

delete from PAUTHOWN.PC\_EVENTS where id='21489';

STEP #10: Set up bi-directional logical replication. Create replication set on East DB. Create subscriptions.

On West database,

select pglogical.create\_replication\_set('pauth\_west\_east\_repl\_set');

SELECT pglogical.replication\_set\_add\_all\_tables('pauth\_west\_east\_repl\_set', ARRAY['pauthown']);

On East database,

SELECT pglogical.create\_subscription(

subscription\_name := 'pauth\_west\_b\_to\_east\_b\_sub',

provider\_dsn := 'host=pauth-b-prf-west-poc.cluster-cjzdkv6oy3dy.us-west-2.rds.amazonaws.com port=1526 dbname=postgres user=pguser password=pguser123',

replication\_sets := ARRAY['pauth\_west\_east\_repl\_set'],

synchronize\_data := false,

forward\_origins := '{}' );

STEP #11: Test DMLs - inserts, updates, deletes - in bi-directional replication

select \* from oauth2own.token where TOKEN\_ID='BB5Av14XG1qagSOdlAsffIUGCcmY9rxCVF/iYjnQx0e6Q=';

select \* from oauth2own.client where CLIENT\_ID='prfqycGIgn9YngHFKP7LU5a2Z2osXCPD4eXQL77WMR6sGZ8BhC';

update oauth2own.token set LAST\_MOD\_TIME=CURRENT\_TIMESTAMP where TOKEN\_ID in ('BB5Av14XG1qagSOdlAsffIUGCcmY9rxCVF/iYjnQx0e6Q=');

update oauth2own.client set LAST\_MOD\_TIME=CURRENT\_TIMESTAMP where CLIENT\_ID='prfqycGIgn9YngHFKP7LU5a2Z2osXCPD4eXQL77WMR6sGZ8BhC';

select \* from OAUTH2OWN.SCOPE where SCOPE\_ID='sat-test22';

insert into OAUTH2OWN.SCOPE ("scope\_id", "scope\_ref", "description", "auth\_code\_max\_life", "access\_token\_max\_life", "refresh\_token\_max\_life", "opaque\_token\_required", "created", "modified", "min\_user\_aal", "rolling\_expiry\_max\_life", "rolling\_expiry\_interval", "last\_mod\_time") values ('sat-test22', 'Intuit.consumer.test.orchestrator', 'Mindy Web Service', 600, 3600,15552000, '1', '2022-04-27 20:29:20.724564', '2022-05-07 10:04:14.174667', null, 15552000, 15552000, '2016-05-07 10:04:14.244036');

delete from OAUTH2OWN.SCOPE where SCOPE\_ID='sat-test22';

STEP #12: Test CDR/conflicts in replication

select \* from oauth2own.token where TOKEN\_ID='AAr6CwRJiAJIen/34tBv1St9wVquxo1eDO+HxXMTOBkjk=';

select \* from oauth2own.client where CLIENT\_ID='prfqyC0rPw6RN2RiBDbeEUXLkAN3L5lgc7eUMmofxqo1DgNecS';

update oauth2own.token set LAST\_MOD\_TIME=CURRENT\_TIMESTAMP where TOKEN\_ID in ('AAr6CwRJiAJIen/34tBv1St9wVquxo1eDO+HxXMTOBkjk='); select \* from oauth2own.token where TOKEN\_ID='AAr6CwRJiAJIen/34tBv1St9wVquxo1eDO+HxXMTOBkjk=';

update oauth2own.client set LAST\_MOD\_TIME=CURRENT\_TIMESTAMP where CLIENT\_ID='prfqyC0rPw6RN2RiBDbeEUXLkAN3L5lgc7eUMmofxqo1DgNecS'; select \* from oauth2own.client where CLIENT\_ID='prfqyC0rPw6RN2RiBDbeEUXLkAN3L5lgc7eUMmofxqo1DgNecS';

select \* from OAUTH2OWN.SCOPE where SCOPE\_ID='test-cdr';

insert into OAUTH2OWN.SCOPE ("scope\_id", "scope\_ref", "description", "auth\_code\_max\_life", "access\_token\_max\_life", "refresh\_token\_max\_life", "opaque\_token\_required", "created", "modified", "min\_user\_aal", "rolling\_expiry\_max\_life", "rolling\_expiry\_interval", "last\_mod\_time") values ('test-cdr', 'Intuit.consumer.test.east', 'East Service', 600, 3600,15552000, '1', '2022-04-27 20:29:20.724564', '2022-05-07 10:04:14.174667', null, 15552000, 15552000, CURRENT\_TIMESTAMP); select \* from OAUTH2OWN.SCOPE where SCOPE\_ID='test-cdr';

insert into OAUTH2OWN.SCOPE ("scope\_id", "scope\_ref", "description", "auth\_code\_max\_life", "access\_token\_max\_life", "refresh\_token\_max\_life", "opaque\_token\_required", "created", "modified", "min\_user\_aal", "rolling\_expiry\_max\_life", "rolling\_expiry\_interval", "last\_mod\_time") values ('test-cdr', 'Intuit.consumer.test.west', 'West Service', 600, 3600,15552000, '1', '2022-04-27 20:29:20.724564', '2022-05-07 10:04:14.174667', null, 15552000, 15552000, CURRENT\_TIMESTAMP); select \* from OAUTH2OWN.SCOPE where SCOPE\_ID='test-cdr';

delete from OAUTH2OWN.SCOPE where SCOPE\_ID='test-cdr'; (delete at the same time)

delete from oauth2own.token where TOKEN\_ID='BB3MDbWABPndfyrQmN8Abz0ShJFZSeiqq7JVWsWSVBrKw='; (record exists in one region)

select TOKEN\_ID,LAST\_MOD\_TIME from oauth2own.token where TOKEN\_ID='BB+Z1rf+c425Xi6YkqwmAAHlPcYpNOAy7B5cGUgSyKzCc=';

update oauth2own.token set LAST\_MOD\_TIME=CURRENT\_TIMESTAMP where TOKEN\_ID in ('BB+Z1rf+c425Xi6YkqwmAAHlPcYpNOAy7B5cGUgSyKzCc='); (record exists in one region)

UPDATEROWEXISTS - Working fine

INSERTROWEXISTS - Working fine

DELETEROWEXISTS - Working fine

UPDATEROWMISSING - NOT working

DELETEROWMISSING - Working fine

STEP #13: DDL replication

In all West/East databases, create function, trigger and tables.

CREATE OR REPLACE FUNCTION pglogical\_assign\_repset()

RETURNS event\_trigger AS $$

DECLARE obj record;

BEGIN

FOR obj IN SELECT \* FROM pg\_event\_trigger\_ddl\_commands()

LOOP

IF obj.object\_type = 'table' THEN

IF obj.schema\_name = 'config' THEN

PERFORM pglogical.replication\_set\_add\_table('configuration', obj.objid);

ELSIF NOT obj.in\_extension THEN

PERFORM pglogical.replication\_set\_add\_table('default', obj.objid);

END IF;

END IF;

END LOOP;

END;

$$ LANGUAGE plpgsql;

CREATE EVENT TRIGGER pglogical\_assign\_repset\_trg

ON ddl\_command\_end

WHEN TAG IN ('CREATE TABLE', 'CREATE TABLE AS')

EXECUTE PROCEDURE pglogical\_assign\_repset();

create table oauth2own.test\_listing(

listid integer not null primary key,

sellerid integer not null,

eventid integer not null,

dateid smallint not null,

numtickets smallint not null,

priceperticket decimal(8,2),

totalprice decimal(8,2),

listtime timestamp);

In both West databases,

INSERT INTO oauth2own.test\_listing values

(10,24858,3375,1994,16,197.00,3152.00,'2022-06-17 09:44:54'),

(11,41053,3877,2146,5,24.00,120.00,'2022-11-16 11:59:10'),

(12,45635,4769,2032,26,65.00,1690.00,'2022-07-25 01:45:49'),

(13,30606,2147,1883,3,172.00,516.00,'2022-02-26 05:04:06');

In both East databases,

INSERT INTO oauth2own.test\_listing values

(211334,15593,275,2091,8,1236.00,9888.00,'2022-09-22 12:32:38'),

(211335,47790,1380,2143,6,1731.00,10386.00,'2022-11-13 06:07:17'),

(211336,22488,7938,2019,8,393.00,3144.00,'2022-07-12 12:09:01');

update oauth2own.test\_listing set listtime=CURRENT\_TIMESTAMP where listid=211335;

select \* from oauth2own.test\_listing order by listid;

drop table oauth2own.test\_listing CASCADE;

alter table OAUTH2OWN.APPLICATION\_GROUP alter column deleted set data type char(2);

SELECT table\_name, column\_name, data\_type,character\_maximum\_length,column\_default FROM information\_schema.columns WHERE table\_name = 'application\_group';

select APPLICATION\_GROUP\_ID,deleted,LAST\_MOD\_TIME from OAUTH2OWN.APPLICATION\_GROUP where application\_group\_id='cShVozFvXSCSecqpxBkDJxGFaTVCzKRfnhbiMIAZnLKV01XgKH';

update OAUTH2OWN.APPLICATION\_GROUP set deleted='11' where application\_group\_id='cShVozFvXSCSecqpxBkDJxGFaTVCzKRfnhbiMIAZnLKV01XgKH';

update OAUTH2OWN.APPLICATION\_GROUP set LAST\_MOD\_TIME=CURRENT\_TIMESTAMP where application\_group\_id='cShVozFvXSCSecqpxBkDJxGFaTVCzKRfnhbiMIAZnLKV01XgKH';

PoC 4-way replication setup with pglogical:

West A

(oauth2-a-prf-west-poc)

East A

(oauth2-a-prf-east-poc)

East B

(oauth2-b-prf-east-poc)

West B

(oauth2-b-prf-west-poc)

Aggregating all data from Oracle decom tasks in a single sheet:

<https://docs.google.com/document/d/1u75WIC1zpeGjO-JTBvM5-2Monm2rypknt_FzT-33ZYg> (Oracle Decomm Oauth2)

Oracle Decom - SyncUp <https://docs.google.com/document/d/1F2fAFV6zqAPqAjqwOWMTddRU1H4mR0Bh-5aKsKinFRE>

Oracle Decommissioning - options explored <https://docs.google.com/document/d/1zp5xXWlJkcu9Sl-sKPBGkk2pshXjTjw4BtwyWQDUGJ8>

Docs from meeting: <https://github.intuit.com/identity-connect/oauth2-service-config/blob/master/oauth2_service.properties> (search for sproc)

SCT conversion issues + DMS: <https://docs.google.com/document/d/1EXgoQmw8Jj3e0YBhy5eksdtNmu7YylV1hG8X2XvY3pI/view>

**Wiki pages:**

RPAS Oracle to Aurora migration using DMS - <https://wiki.intuit.com/display/Identity/Role+Assignment+Data+Migration+using+AWS+DMS>

# [EPR Aurora DB Migration Deployment PROD](https://wiki.intuit.com/display/EIS/EPR+Aurora+DB+Migration+Deployment+PROD) <https://wiki.intuit.com/display/EIS/EPR+Aurora+DB+Migration+Deployment+PROD>

# [Major version upgrade via Logical Replication](https://wiki.intuit.com/display/QBOEN/Major+version+upgrade+via+Logical+Replication) <https://wiki.intuit.com/display/QBOEN/Major+version+upgrade+via+Logical+Replication>

**AWS documents:**

Best practices for Amazon RDS PostgreSQL replication

<https://aws.amazon.com/blogs/database/best-practices-for-amazon-rds-postgresql-replication/>

PostgreSQL bi-directional replication using pglogical

<https://aws.amazon.com/blogs/database/postgresql-bi-directional-replication-using-pglogical/>

Installing SCT

<https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Installing.html>

Migration playbook, oracle to amazon aurora PostgreSQL

<https://d1.awsstatic.com/whitepapers/Migration/oracle-database-amazon-aurora-postgresql-migration-playbook-12.4.pdf>

<https://docs.aws.amazon.com/dms/latest/oracle-to-aurora-postgresql-migration-playbook/dms-mpb-oracle-to-aurora-postgresql.pdf>

sysdate use-cases during migration

<https://aws.amazon.com/blogs/database/converting-the-sysdate-function-from-oracle-to-postgresql/>

Schema migration using SCT

<https://catalog.us-east-1.prod.workshops.aws/workshops/77bdff4f-2d9e-4d68-99ba-248ea95b3aca/en-US/oracle-aurora/schema-conversion>

Data migration using DMS

<https://catalog.us-east-1.prod.workshops.aws/workshops/77bdff4f-2d9e-4d68-99ba-248ea95b3aca/en-US/oracle-aurora/data-migration>

<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.Replication.Logical.html>

<https://aws.amazon.com/blogs/database/part-2-upgrade-your-amazon-rds-for-postgresql-database-using-the-pglogical-extension/>

<https://docs.aws.amazon.com/dms/latest/oracle-to-aurora-postgresql-migration-playbook/chap-oracle-aurora-pg.sql.cursors.html>

Global indexes (in Oracle) are not supported in PostgreSQL.

Client – 1 GI on APP\_TOKEN

Token – 4 GIs on PARENT\_TOKEN\_ID, SECURITY\_CTX\_AUTH\_ID, CLIENT\_ID, SECURITY\_CTX\_REALM\_ID

Get rid of partitioning (on token)

Ref cursors -> verify/test… may change to some other cursor…

PostgreSQL doesn't support bitmap indexes

PostgreSQL does not support foreign keys referencing partitioned tables

Aurora Crash test

EC2 stress test – (stress -c 4)

LOB objects – PAuth has 2 LOBs => store them in S3, point to it.

SCHEMA\_NAME TABLE\_NAME COLUMN\_NAME COLUMN\_COUNT

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PAUTHOWN PARTNER PROTOCOL\_FEATURES\_CLOB 1

PAUTHOWN PARTNER\_HISTORY PROTOCOL\_FEATURES\_CLOB 1

Sequencing replication pglogical?

DMS

APN replication tools:

* HVR
* Striim
* Qlik (formerly Attunity)