# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all trust

# IPv4 local connections:

host all all 127.0.0.1/32 trust

# IPv6 local connections:

host all all ::1/128 trust

# Allow replication connections from localhost, by a user with the

# replication privilege.

local replication all trust

host replication all 127.0.0.1/32 trust

host replication all ::1/128 trust

postgres=# set password\_encryption ='scram-sha-256';

SET

postgres=# CREATE ROLE finance WITH PASSWORD 'mostcommonpassword';

CREATE ROLE

postgres=# select substr(rolpassword,1,14) from pg\_authid where rolname ='finance';

substr

----------------

SCRAM-SHA-256$

(1 row)

postgres=#

postgres=# create role sales with password 'md5password';

CREATE ROLE

postgres=# select rolpassword from pg\_authid where rolname ='sales';

rolpassword

-------------------------------------

md5059f18f30cad64836a01f936c9ba7dd4

(1 row)

# All users (from 192.168.12.10) require SCRAM authentication to connect postgres database.

# TYPE DATABASE USER ADDRESS METHOD

host postgres all 192.168.12.10/32 scram-sha-256

8c85902e3a2c:data $ **initdb --help|grep auth**

-A, --auth=METHOD default authentication method for local connections

--auth-host=METHOD default authentication method for local TCP/IP connections

--auth-local=METHOD default authentication method for local-socket connections

8c85902e3a2c:data $

#All users from 192.168.54.1 server are rejected.

# TYPE DATABASE USER ADDRESS METHOD

host all all 192.168.54.1/32 reject

# "local" is for Unix domain socket connections only

local all all peer

host all all 192.168.10.22/24 ident map=my\_ident\_map

# MAPNAME SYSTEM-USERNAME PG-USERNAME

my\_ident\_map my\_os\_user ident\_db\_user

host    all    dbuser 0.0.0.0/0    ldap ldapserver=ldapserver.example.com ldapprefix="cn=" ldapsuffix=", dc=example, dc=com"

hostssl all all 0.0.0.0/0 md5

postgres=# \z titles

Access privileges

Schema | Name | Type | Access privileges | Column privileges | Policies

--------+--------+-------+-------------------+-------------------+----------

public | titles | table | | |

(1 row)

postgres=# grant select,insert on titles to sales;

GRANT

Time: 27.145 ms

postgres=# \z titles

Access privileges

Schema | Name | Type | Access privileges | Column privileges | Policies

--------+--------+-------+---------------------------+-------------------+----------

public | titles | table | postgres=arwdDxt/postgres+| |

| | | sales=ar/postgres | |

(1 row)

postgres=# begin;

BEGIN

postgres=# grant select on titles to sales;

GRANT

postgres=# \z titles

Access privileges

Schema | Name | Type | Access privileges | Column privileges | Policies

--------+--------+-------+---------------------------+-------------------+----------

public | titles | table | postgres=arwdDxt/postgres+| |

| | | **sales=r/postgres** | |

(1 row)

postgres=# **rollback**;

ROLLBACK

postgres=# \z titles

Access privileges

Schema | Name | Type | Access privileges | Column privileges | Policies

--------+--------+-------+---------------------------+-------------------+----------

public | titles | table | postgres=arwdDxt/postgres | |

(1 row)

ALTER DEFAULT PRIVILEGES IN SCHEMA <schema\_name> GRANT <privilege> ON TABLES TO <role>;

postgres=# GRANT update(emp\_no) ON titles TO sales;

GRANT

postgres=# \z titles

Access privileges

Schema | Name | Type | Access privileges | Column privileges | Policies

--------+--------+-------+---------------------------+--------------------+----------

public | titles | table | postgres=arwdDxt/postgres | emp\_no: +|

| | | | sales=w/postgres |

(1 row)

postgres=#

#this will revoke create privileges from all the users  
REVOKE CREATE ON SCHEMA public FROM public;

#you would need to explicitly grant privileges if you want a user with write permissions  
GRANT CREATE ON SCHEMA public to write;

#create read-only  
CREATE ROLE user\_readonly WITH PASSWORD 'mypassword';  
GRANT SELECT ON ALL TABLES IN SCHEMA public TO user\_readonly;

postgres=# ALTER TABLE accounts ENABLE ROW LEVEL SECURITY;

ALTER TABLE

postgres=# CREATE POLICY account\_managers ON accounts TO managers

postgres-# USING (manager = current\_user);

CREATE POLICY

postgres=> \c - postgres

psql (11.2, server 10.2)

You are now connected to database "postgres" as user "postgres".

postgres=# select count(1) from accounts ;

count

-------

2

(1 row)

postgres=# \c - managers

psql (11.2, server 10.2)

You are now connected to database "postgres" as user "managers".

postgres=> select \* from accounts ;

manager | employee | contact\_email | salary

----------+----------+---------------+--------

managers | employee | abc@abc.com | 100000

(1 row)

postgres=> select count(1) from accounts ;

count

-------

1

(1 row)

ALTER ROLE abc BYPASSRLS;

ALTER TABLE accounts FORCE ROW LEVEL SECURITY;

./configure --with-openssl

ssl = on # (change requires restart)

#ssl\_ciphers = 'ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH' # allowed SSL ciphers #ssl\_prefer\_server\_ciphers = on # (change requires restart) #ssl\_ecdh\_curve = 'prime256v1' # (change requires restart)

ssl\_cert\_file = '/etc/ssl/postgres/starry.io.crt' # (change requires restart)

ssl\_key\_file = '/etc/ssl/postgres/starry.io.key' # (change requires restart) ssl\_ca\_file = '' # (change requires restart)

#ssl\_crl\_file = '' # (change requires restart)

ssh -L 63333:localhost:5432 foo@bar.com

psql -h localhost -p 63333 postgres

postgres=# CREATE FUNCTION to\_avoid\_object\_drops()

postgres-# RETURNS event\_trigger

postgres-# LANGUAGE plpgsql AS $$

postgres$# DECLARE

postgres$# object record;

postgres$# BEGIN

postgres$# FOR object IN SELECT \* FROM pg\_event\_trigger\_dropped\_objects()

postgres$# LOOP

postgres$# RAISE EXCEPTION '% dropped object: % %.% %',

postgres$# tg\_tag,

postgres$# object.object\_type,

postgres$# object.schema\_name,

postgres$# object.object\_name,

postgres$# object.object\_identity;

postgres$# END LOOP;

postgres$# END

postgres$# $$;

CREATE FUNCTION

postgres=# CREATE EVENT TRIGGER to\_avoid\_object\_drops

postgres-# ON sql\_drop

postgres-# EXECUTE PROCEDURE to\_avoid\_object\_drops();

CREATE EVENT TRIGGER

postgres=# DROP TABLE join1; NOTICE: DROP TABLE dropped object: table public.join1 public.join1

NOTICE: DROP TABLE dropped object: index public.unq\_index\_join public.unq\_index\_join

NOTICE: DROP TABLE dropped object: type public.join1 public.join1

NOTICE: DROP TABLE dropped object: type public.\_join1 public.join1[]

NOTICE: DROP TABLE dropped object: toast table pg\_toast.pg\_toast\_16554 pg\_toast.pg\_toast\_16554

NOTICE: DROP TABLE dropped object: index pg\_toast.pg\_toast\_16554\_index pg\_toast.pg\_toast\_16554\_index

NOTICE: DROP TABLE dropped object: type pg\_toast.pg\_toast\_16554 pg\_toast.pg\_toast\_16554

ERROR: DROP TABLE dropped object: table public.join1 public.join1

CONTEXT: PL/pgSQL function to\_avoid\_object\_drops() line 7 at RAISE

CREATE FUNCTION test\_audit\_trig() RETURNS trigger

LANGUAGE plpgsql

AS $$

DECLARE

v\_dmltype text;

BEGIN

IF TG\_OP = 'INSERT' THEN

v\_dmltype = 'I';

ELSE

v\_dmltype = 'U';

END IF;

INSERT INTO auditing.test (col1, col2,....dmltype, change\_timestamp)

VALUES (NEW.col1, NEW.col2,....v\_dmltype, current\_timestamp)

RETURN NULL;

Attach trigger to a table:

CREATE TRIGGER test\_audit\_trig BEFORE INSERT OR UPDATE OR DELETE ON table\_name

FOR EACH ROW EXECUTE PROCEDURE test\_audit\_trig();

postgres=# CREATE TABLE crypt\_table(uname varchar, pwd\_crypt text, pwd\_md5 text);

CREATE TABLE

postgres=# create extension pgcrypto ;

CREATE EXTENSION

postgres=# INSERT INTO crypt\_table VALUES ('Robert', crypt('testpassword',gen\_salt('md5')),md5('testpassword'));

INSERT 0 1

postgres=# INSERT INTO crypt\_table VALUES ('Tom', crypt('testpassword',gen\_salt('md5')),md5('testpassword'));

INSERT 0 1

postgres=# select \* from crypt\_table ;

uname | pwd\_crypt | pwd\_md5

--------+------------------------------------+----------------------------------

Robert | $1$Y3iUMA6h$OUTGwuH7hoFJnOO48taNV1 | e16b2ab8d12314bf4efbd6203906ea6c

Tom | $1$IfATTihP$A78rkOIIEvkZ2LjcRT6hd1 | e16b2ab8d12314bf4efbd6203906ea6c

(2 rows)

postgres=# select uname from crypt\_table where uname='Tom' and pwd\_crypt=crypt('testpassword',pwd\_crypt);

uname

-------

Tom

(1 row)