RDS PostgreSQL

A Journey Down the Amazon
Gabrielle Roth
SCALE 14x

About me

- “I use Postgres so I don't have to think”
- Co-founder & -leader of PDXPUG
- Pg user since 7.2? 7.4?
- Always on my hardware or VMs that were (nominally) under my control
- Been using RDS in production for about a year and a half now
- Currently work for RenewFinancial

Our Environment

- Many small databases
- Low but bursty tx
- Combo of RDS and self-hosted
- All Postgres (-1 MySQL)

Topics

- What is this RDS thing, anyway
- Basic setup
- Things that ROCK about RDS
- Things that are important to me that are missing or weird

AWS? EC2? RDS?

- AWS = Amazon Web Services
- EC2 = Elastic Compute Cloud
- RDS = Relational Database Services
- Postgres on EC2 = Pg on a hosted VM
- RDS Postgres = managed Pg offering
- https://www.expeditedssl.com/aws-in-plain-english

RDS Feature highlights

- Replication, failover, backups – I don't have to deal with configuring them
- Easy read replicas and encryption
- Scalability!
- Automatic patching & OS upgrades
- New! Point & click upgrade between (some) versions
- SDK supports many languages, including a robust CLI

Great, where do I sign up?

Why are you doing this?

- I want someone else to blame!
- To save money
- Easy setup, failover, restore
- To make things easier for your DBA

  ...or maybe get rid of your DBA

I'll save money!

- You pay for:
  - Instance
  - Storage
  - Data transfer out
  - Support
- Prices change frequently
- Amazon has a cost calculator:

http://calculator.s3.amazonaws.com/index.html

I'll save money!

Recommendations:

- Check out Trusted Advisor (part of support)
- Review your bills every month
- Set up an alert so you know when you're getting close to your limit
- Power down unused test instances
- Purchase reserved instances, but do the math

I want easy setup, failover, and restore!

WIN.

- Don't have to deal with:
  - Configuring replication
  - Monitoring replication
  - Recovering/cleaning up after a failover
  - Configuring or scheduling backups
- Read replicas are just a mouse click away
- Restore is so simple, it's ideal for spinning up quick instances for ad-hoc dev work, reporting, what have you.

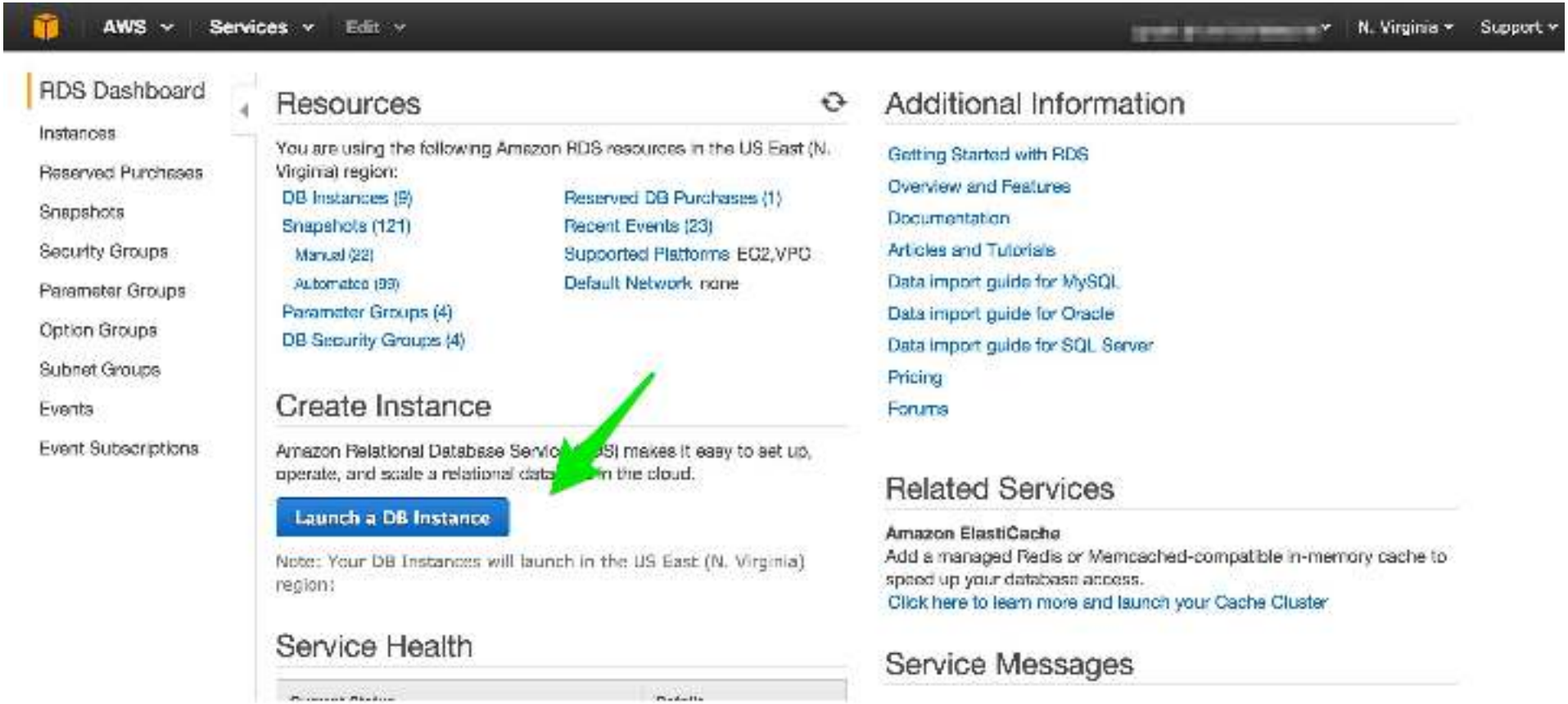
What is an “Instance”

- VM host, sort of
- No direct system access (no ssh)
- *Instance* managed via AWS tools (console, API)
  - Start up, power down, apply some configuration
- *Database access* only via psql, pgAdmin, etc

Identity & Access Management (IAM)

- Limit users' authority to manage the *instance*
  - Create/destroy instances, snapshots, etc
- Interactions with an *instance*, not a *database*
- Guard the keys closely
- Use CloudTrail to track activity

Setup Overview



CLI/API

- CLI toolkit and the API allow you to automate everything!
- Some things aren't very straightforward (eg downloading large log files)
- There are some things you can do from the CLI that you can't do from the console (eg review event notifications)

CLI/API

- Two CLIs: *rds* and *aws*
  - Support recommends the aws cli over the rds cli
  - They are very similar, but just different enough that it's aggravating to switch back and forth
- Download, install, configure
  - Java environment
  - ~/.aws/config and ~/.aws/credentials
- Advanced Usage of the AWS CLI

  www.youtube.com/watch?v=vP56l7qThNs

Create an instance - CLI

```
aws rds create-db-instance \
--db-instance-identifier gabs-db \
--engine postgres \
--engine-version 9.3.5 \
--master-username gabrielle \
--master-user-password my_excellent_password \
--db-parameter-group-name load-params \
--db-instance-class db.t2.small \
--allocated-storage 100 \
--no-multi-az \
--backup-retention-period 30 \
--no-publicly-accessible \
--db-subnet-group-name gabs-db-subnet \
--vpc-security-group-ids sg-xxxx
```

Create an instance - CLI

```
aws rds create-db-instance \
--db-instance-identifier gabs-db \
--engine postgres \
--engine-version 9.3.5 \
--master-username gabrielle \
--master-user-password my_excellent_password \
--db-parameter-group-name load-params \
--db-instance-class db.t2.small \
--allocated-storage 100 \
--no-multi-az \
--backup-retention-period 30 \
--no-publicly-accessible \
--db-subnet-group-name gabs-db-subnet \
--vpc-security-group-ids sg-xxxx
```

Create an instance - CLI

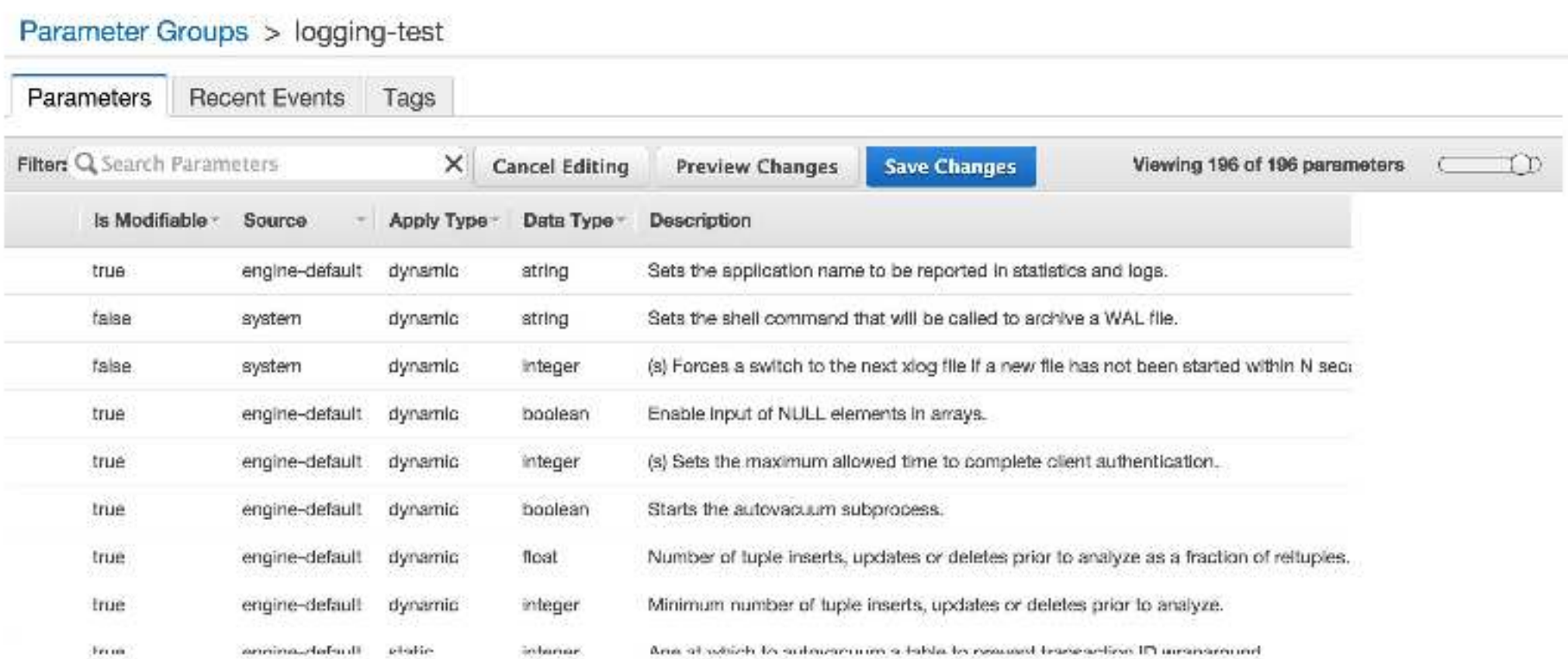
```
aws rds create-db-instance \
--db-instance-identifier gabs-db \
--engine postgres \
--engine-version 9.3.5 \
--master-username gabrielle \
--master-user-password my_excellent_password \
--db-parameter-group-name load-params \
--db-instance-class db.t2.small \
--allocated-storage 100 \
--no-multi-az \
--backup-retention-period 30 \
--no-publicly-accessible \
--db-subnet-group-name gabs-db-subnet \
--vpc-security-group-ids sg-xxxx
```

Parameter group, aka postgresql.conf, sort of

- It's all there
- You just can't change all of it.
- The GUI is not user friendly, but neither is the CLI.
- Keep the (Pg) docs handy.
- And this:

  www.davidmkerr.com/2013/11/tune-your-postgres-rds-instance-via.html

Parameters


Parameter Groups > logging-test
Parameters
Recent Events
Tags
Filter: Search Parameters
Cancel Editing
Preview Changes
Save Changes
Viewing 196 of 196 parameters
Is Modifiable
Source
Apply Type
Data Type
Description
true engine-default dynamic string Sets the application name to be reported in statistics and logs.
false system dynamic string Sets the shell command that will be called to archive a WAL file.
false system dynamic integer (s) Forces a switch to the next xlog file if a new file has not been started within N sec
true engine-default dynamic boolean Enable input of NULL elements in arrays.
true engine-default dynamic integer (s) Sets the maximum allowed time to complete client authentication.
true engine-default dynamic boolean Starts the autovacuum subprocess.
true engine-default dynamic float Number of tuple inserts, updates or deletes prior to analyze as a fraction of reltuples.
true engine-default dynamic integer Minimum number of tuple inserts, updates or deletes prior to analyze.

Parameters

- Parameters you can't modify:
  - Anything in the default parameter group
    - Create your own!
    - Create several! (They'll be available to all your instances.)
  - Anything to do with streaming rep
  - Several logging params (target file, format, log_line_prefix)
  - System layout (data directory, location of conf files)
  - Server encoding

Create an instance - CLI

```
aws rds create-db-instance \
--db-instance-identifier gabs-db \
--engine postgres \
--engine-version 9.3.5 \
--master-username gabrielle \
--master-user-password my_excellent_password \
--db-parameter-group-name load-params \
--db-instance-class db.t2.small \
--allocated-storage 100 \
--no-multi-az \
--backup-retention-period 30 \
--no-publicly-accessible \
--db-subnet-group-name gabs-db-subnet \
--vpc-security-group-ids sg-xxxx
```

Instance classes

- db.[class].[size]
- db.t1.micro: testing only (not current)
- db.t2.[size]: burst-capable (can max the CPU)
- db.m4.[size]: “standard”
- db.r3.[size]: memory optimized

Create an instance - CLI

```
aws rds create-db-instance \
--db-instance-identifier gabs-db \
--engine postgres \
--engine-version 9.3.5 \
--master-username gabrielle \
--master-user-password my_excellent_password \
--db-parameter-group-name load-params \
--db-instance-class db.t2.small \
--allocated-storage 100 \
--no-multi-az \
--backup-retention-period 30 \
--no-publicly-accessible \
--db-subnet-group-name gabs-db-subnet \
--vpc-security-group-ids sg-xxxx
```

Create an instance - CLI

```
aws rds create-db-instance \
--db-instance-identifier gabs-db \
--engine postgres \
--engine-version 9.3.5 \
--master-username gabrielle \
--master-user-password my_excellent_password \
--db-parameter-group-name load-params \
--db-instance-class db.t2.small \
--allocated-storage 100 \
--no-multi-az \
--backup-retention-period 30 \
--no-publicly-accessible \
--db-subnet-group-name gabs-db-subnet \
--vpc-security-group-ids sg-xxxx
```

Regions vs AZs

- A *Region* is a geographical area. US East, US West (2), EU West, EU Central, AP SE (2), AP NE, SA.
- An *Availability Zone* is an area within that region, e.g. us-east-1c
  - Think of it as a single DC.
- Multi-AZ means you failover to another area *within the same region*

Create an instance - CLI

```
aws rds create-db-instance \
--db-instance-identifier gabs-db \
--engine postgres \
--engine-version 9.3.5 \
--master-username gabrielle \
--master-user-password my_excellent_password \
--db-parameter-group-name load-params \
--db-instance-class db.t2.small \
--allocated-storage 100 \
--no-multi-az \
--backup-retention-period 30 \
--no-publicly-accessible \
--db-subnet-group-name gabs-db-subnet \
--vpc-security-group-ids sg-xxxx
```

Load your data

- docs.aws.amazon.com/AmazonRDS/latest/UserGuide/PostgreSQL.Procedural.Importing.html
- pg_restore or Amazon DMS
- Take a snapshot!
- VACUUM [FREEZE] ANALYZE;
- Change to your prod param group + add Multi-AZ; reboot.

Finishing touches

```
aws rds modify-db-instance \
--db-instance-identifier gabs-db \
--db-parameter-group-name load-params \
--multi-az \
--apply-immediately
...wait...
aws rds reboot-db-instance \
--db-instance-identifier gabs-db \
[--failover | --no-failover ]
```

Failover

- AWS handles replication for you
- Works well, but you will have a “brief” outage

Restore/PITR

It's easy!

...once you know how

(DR) Restore

- Choose the snapshot (or point in time) you want to recover from
- Restore it to a new instance
- Rename the old one to get it out of the way
  - ...and put it in the default security group, so nobody can access it
- Rename the new one to the desired instance name
- TEST IT before you destroy anything!
- Make a checklist and have regular fire drills with this process

Restore from a snapshot

```
aws rds restore-db-instance-from-db-snapshot \
--db-instance-identifier restore-test \
--db-snapshot-identifier rds:gabs-db-2015-02-05-08-05 \
...whatever other options you want...

...wait...
aws rds modify-db-instance \
--db-instance-identifier restore-test \
--db-parameter-group-name prod-param-group \
--vpc-security-group-ids sg-xxxx \
--apply-immediately
```

...and then run ANALYZE.

Point-in-Time Recovery (PITR)

```
aws rds restore-db-instance-to-point-in-time \
--source-db-instance-identifier gabs-db \
--target-db-instance-identifier gabs-db-well-hell \
--restore-time 2015-01-22T09:43:00Z \
...whatever other options you want …

...wait...
aws rds modify-db-instance \
--db-instance-identifier gabs-db-well-hell \
--db-parameter-group-name prod-param-group \
--vpc-security-group-ids sg-xxxx \
--apply-immediately
```

...and then run ANALYZE.

Restore/Recovery con't.

- Can sometimes take a while
- Choosing a different storage type can/will slow it down *a lot*
- You can't resize storage as part of this process
  - Storage can only be expanded, anyway

“I don't need a DBA.”

You need a DBA to:

Configure Pg appropriately

Choose appropriate instance size for your workload

Figure out what in [Sam Hill] the ORM is doing

Secure and audit databases

Ensure data quality

Tune queries

Mentor devs

...

Things to remember

- Some [important] Postgres features are not available.
- You are not the database superuser.
- This is not your system.
- “We've just come to accept a certain amount of unplanned downtime.”

Where to get help

- Purchase the support, at least at first
  - RDS support people ROCK.
- Hang out in the forums. Amazon folks monitor them pretty closely
- The copious documentation
  - But cross-reference your findings
- @dog_rates

Postgres features you may miss:
No pg_hba.conf

- Access is managed by “VPC”, Virtual Private Cloud + database security groups
- You can't control access per-database, -user, -source, or auth method, as you would with a pg_hba.conf
- No way to force SSL

Postgres features you may miss:
installing whatever extensions you want

- Choose from those AWS makes available
- They do add more periodically, and are responsive to community requests
- You may be able to install certain extensions via the old-fashioned way: SQL
- See www.databasesoup.com/2014/12/loading-pgpartman-on-rds-or-heroku.html

Currently available extensions

- Ignore output from this:

  ```
  SELECT * FROM pg_available_extensions ORDER BY name;
  ```

- Use this instead:

  ```
  SHOW rds.extensions;
  ```

btree_gin, btree_gist, chkpass, citext, cube, dblink, dict_int, dict_xsyn, earthdistance, fuzzystrmatch, hstore, intagg, intarray, ip4r, isn, ltree, pgcrypto, pgrowlocks, pgstattuple, pg_buffercache, **pg_stat_statements**, pg_trgm, plcoffee, plls, plperl, **plpgsql**, pltcl, plv8, **postgis**, postgis_tiger_geocoder, postgis_topology, **postgres_fdw**, sslinfo, **tablefunc**, test_parser, tsearch2, unaccent, uuid-ossp

- `SELECT name, version FROM pg_extension;`

You are not the database superuser.

- Can't pg_dumpall
  - DMS? aws.amazon.com/dms/
- Manual VACUUM skips certain tables
  - pg_database, pg_tablespace, ...
- <insufficient privs> in pg_stat_activity
- You don't get all necessary log messages
  - autovacuum (fixed in 9.4.5)
  - lock_waits (fixed in 9.4.3)
- REASSIGN … nope.

You are not the database superuser.

```
psql: FATAL: remaining connection slots are reserved
for non-replication superuser connections
```

You are not the database superuser.

- Good news!
  rds_superuser_reserved_connections available in 9.4.5.

- Or: hinky workaround, must plan in advance:

  ```
  ALTER database gabs_db CONNECTION LIMIT [x];
  ```

  Where x is something like:

  max_connections - superuser_reserved_connections - 3

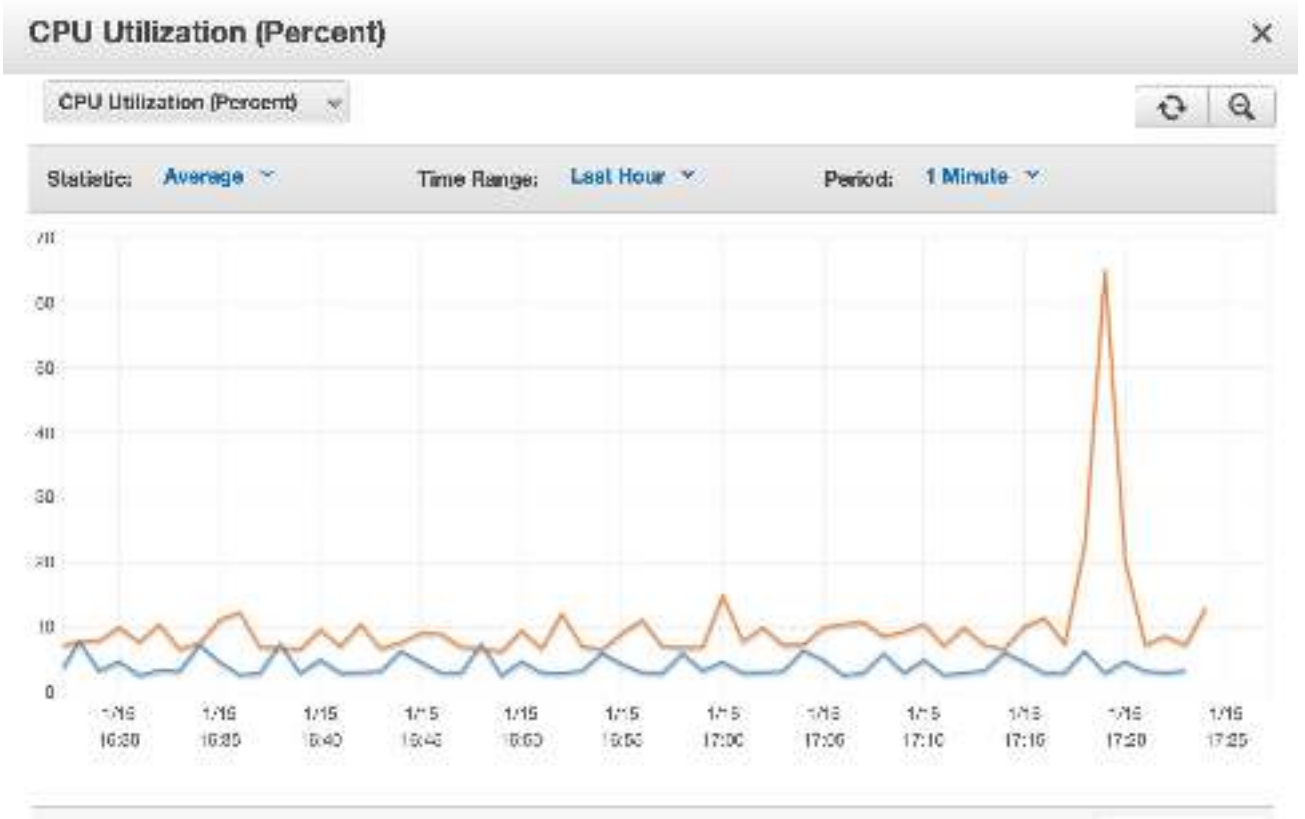
This is not your system.

- DB access only (psql, etc)
- Patches etc get applied for you (yay!)
- OS upgrades won't cause downtime (usually)
- Pg upgrades **will** cause downtime
  - Usually new features require a Pg upgrade
- Easy upgrades between certain versions only

Let's talk about monitoring.

- Cloudwatch metrics
  - Mostly “system” metrics; database connections
  - Can download existing
  - Can create your own

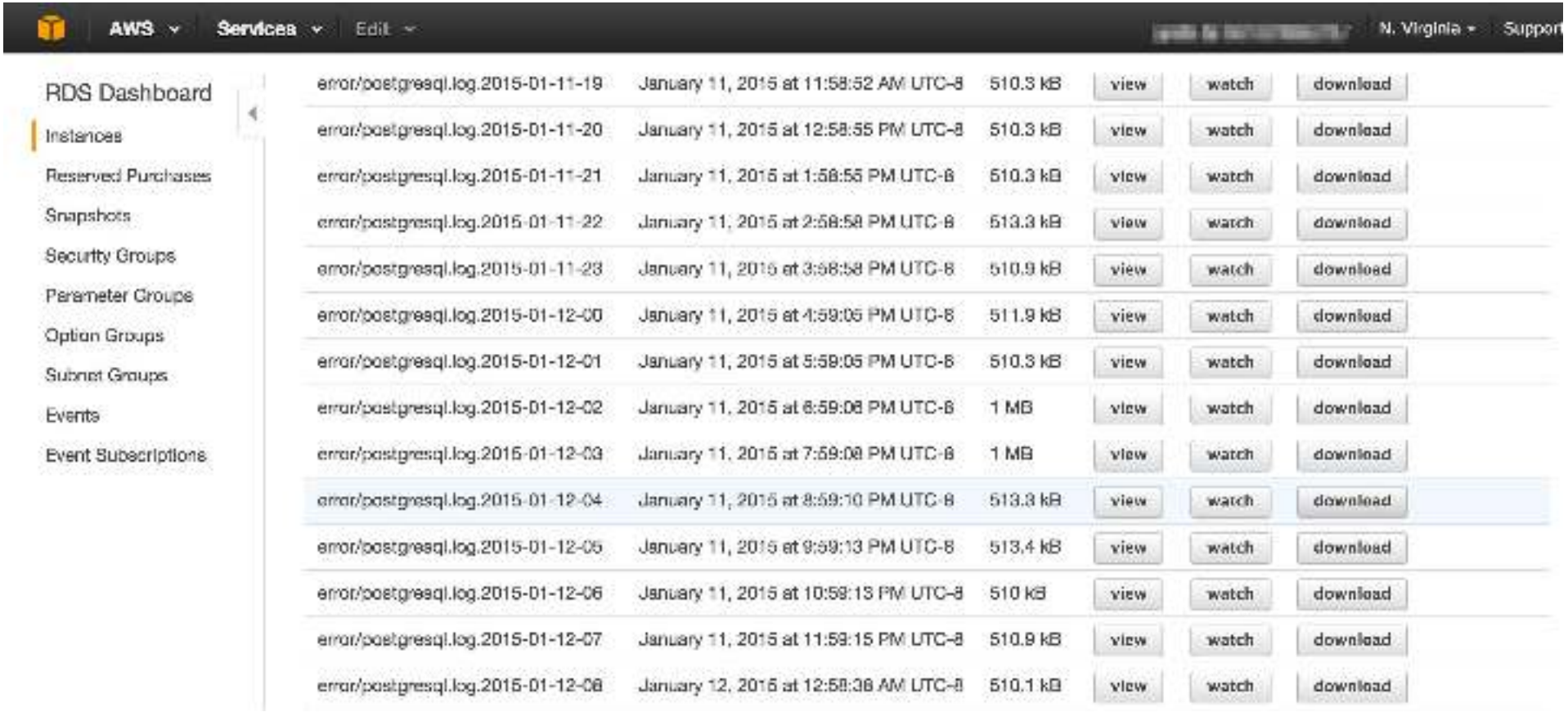
Cloudwatch metrics



Let's talk about monitoring (2).

- Pg logs
  - kind of a PITA to access
  - And you don't get to choose the format or the log_line_prefix
  - ...but you *can* make pgbadger work with it

“error” logs


AWS
Services
Edit
N. Virginia
Support
RDS Dashboard
Instances
Reserved Purchases
Snapshots
Security Groups
Parameter Groups
Option Groups
Subnet Groups
Events
Event Subscriptions
error/postgresql.log.2015-01-11-19 January 11, 2015 at 11:58:52 AM UTC-8 510.3 kB view watch download
error/postgresql.log.2015-01-11-20 January 11, 2015 at 12:58:55 PM UTC-8 510.3 kB view watch download
error/postgresql.log.2015-01-11-21 January 11, 2015 at 1:58:55 PM UTC-8 510.3 kB view watch download
error/postgresql.log.2015-01-11-22 January 11, 2015 at 2:58:58 PM UTC-8 513.3 kB view watch download
error/postgresql.log.2015-01-11-23 January 11, 2015 at 3:58:58 PM UTC-8 510.9 kB view watch download
error/postgresql.log.2015-01-12-00 January 11, 2015 at 4:59:05 PM UTC-8 511.9 kB view watch download
error/postgresql.log.2015-01-12-01 January 11, 2015 at 5:59:05 PM UTC-8 510.3 kB view watch download
error/postgresql.log.2015-01-12-02 January 11, 2015 at 6:59:08 PM UTC-8 1 MB view watch download
error/postgresql.log.2015-01-12-03 January 11, 2015 at 7:59:08 PM UTC-8 1 MB view watch download
error/postgresql.log.2015-01-12-04 January 11, 2015 at 8:59:10 PM UTC-8 513.3 kB view watch download
error/postgresql.log.2015-01-12-05 January 11, 2015 at 9:59:13 PM UTC-8 513.4 kB view watch download
error/postgresql.log.2015-01-12-06 January 11, 2015 at 10:59:13 PM UTC-8 510 kB view watch download
error/postgresql.log.2015-01-12-07 January 11, 2015 at 11:59:15 PM UTC-8 510.9 kB view watch download
error/postgresql.log.2015-01-12-08 January 12, 2015 at 12:58:38 AM UTC-8 510.1 kB view watch download

“error” logs


AWS
Services
Edit
N. Virginia
Support
RDS Dashboard
Instances
Reserved Purchases
Snapshots
Security Groups
Parameter Groups
Option Groups
Subnet Groups
Events
Event Subscriptions
Viewing Log: error/postgresql.log.2015-01-15-20 (4.7 kB)
text:
background:
grep this, buddy
Privacy Policy
Terms of Use
Feedback

“Unplanned downtime.”

- They can failover and fail/restart, and you may not get to know why
- If you're taking more than your share of resources, AWS will stop you (via instance restart/failover)
- Read the SLA
  - aws.amazon.com/rds/sla/
- Build failure handling into your architecture
- Have a good DR plan!

Let's talk (more) about backups and restores.

- Automated snaps once a day only at this time
- Automatic snapshots are destroyed when you destroy an instance
- Backups/snapshots are local AZ only
- Save your backups off to another region!
  - Don't forget to age them out, though
  - It's difficult to copy your snapshots completely off of Amazon's services

Copy snap to other region

```
aws rds copy-db-snapshot \

--source-db-snapshot-identifier rds:gabs-db-2016-01-05-08-05 \

--target-db-snapshot-identifier [ARN] \

--region us-east-1c
```

ARN:

```
arn:aws:rds:us-west-1:12345678910:snapshot:rds:gabs-db-2016-01-05-08-05
```

In conclusion...

Pros:

- Great for a dev env
- Easy setup
- Easy restore/PITR
- Easy failover
- Can automate testing/deploys: snapshot, test, rollback

Cons:

- Not as configurable
- You need your own monitoring
- Security concerns
- Who owns your data?
- “Stuff breaks and I don't get to know why”

Questions?

@gorthx
gorthx@gmail.com
gorthx.wordpress.com

Other conferences you may like

PgConfUS – 18-20 Apr 2016, NYC

http://www.pgconf.us

Postgres Open – Sept 2016, Dallas

http://postgresopen.org/

PgConf.EU – sometime, somewhere

http://pgconf.eu/

Thank you!

#pdxpug

Denish Patel

Grant McAlister

Magnus Hagander

Selena Deckelmann