PALOMINO

OPERATIONAL EXCELLENCE
FOR DATABASES

PostgreSQL on AWS

PgDay Bs As 2013
by Emanuel Calvo

Palomino - Service Offerings

- Monthly Support:
  - Being renamed to Palomino DBA as a service.
  - Eliminating 10 hour monthly clients.
  - Discounts are based on spend per month (0-80, 81-160, 161+
  - We will be penalizing excessive paging financially.
  - Quarterly onsite day from Palomino executive, DBA and PM for clients using 80 hours or more per month.
  - Clients using 80-160 hours get 2 New Relic licenses. 160 hours plus get 4.
- Adding annual support contracts:
  - Consultation as needed.
  - Emergency pages allowed.
  - Small bucket of DBA hours (8, 16 or 24)

For more information, please go to: [Spreadsheet]

amazon web services | Partner Network
ADVANCED TECHNOLOGY PARTNER

About me:

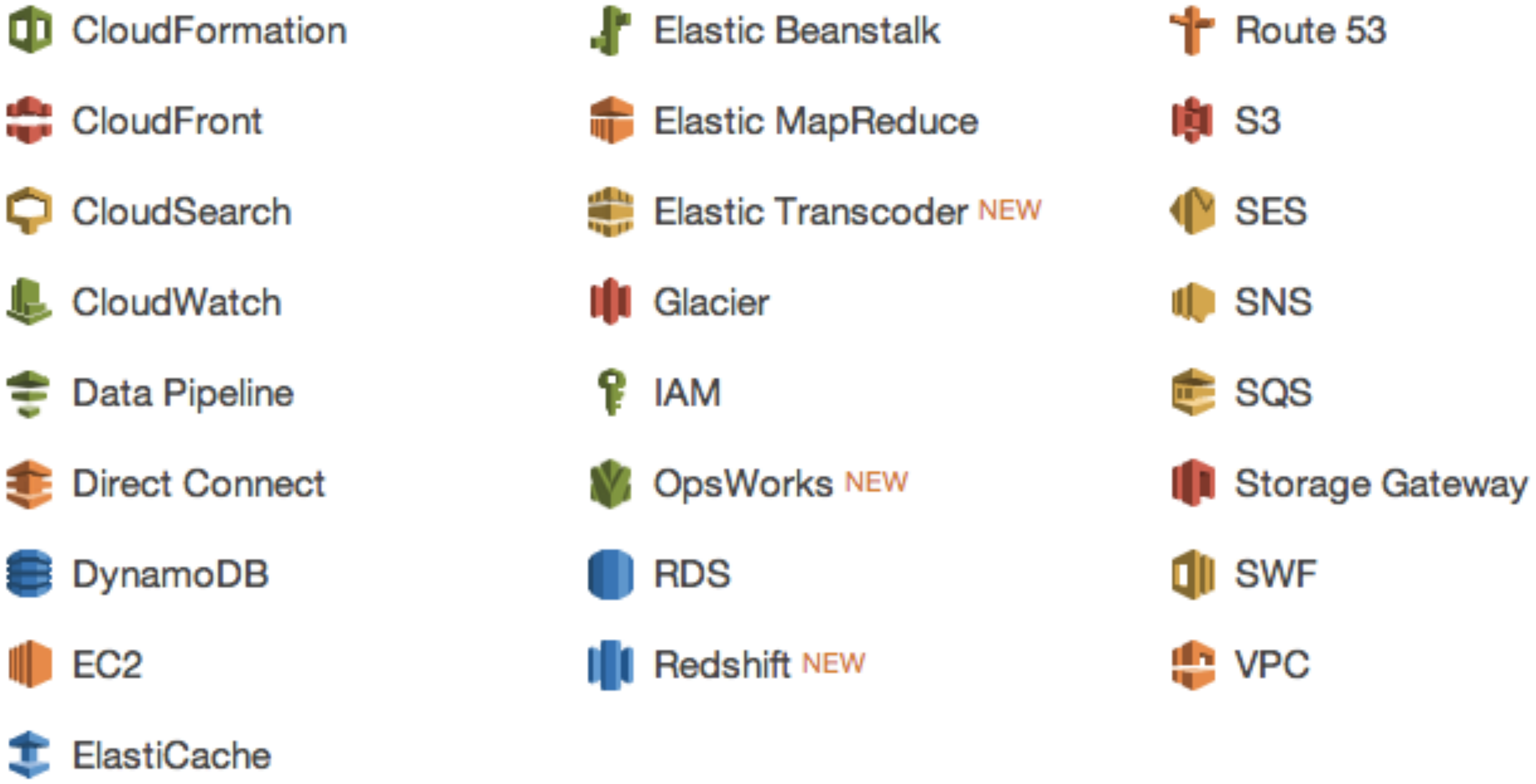
- Operational DBA at PalominoDB.
    - PostgreSQL teach leader, MySQL, MariaDB, Cassandra databases.
- Check out my LinkedIn Profile at: http://es.linkedin.com/in/ecbcbcb/
- Co-author of “RDBMS in the Cloud: PostgreSQL on AWS” (Amazon Library)
- Mercenary.

This talk is about:

- Main services available for Postgres implementation on AWS.
- Some advises you’ll appreciate.

Amazon Web Services

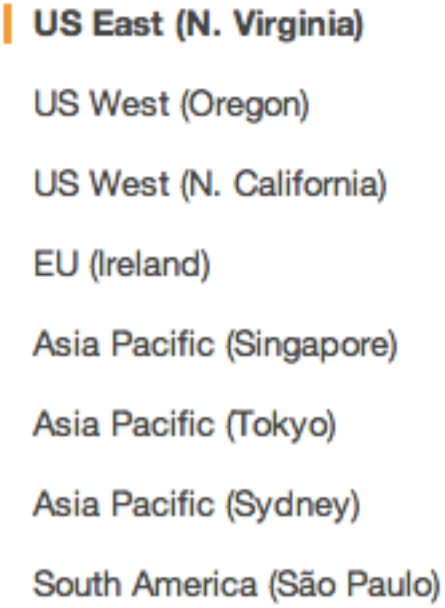
- Services provider.
- Services will depend on the regions (not all the features are available across the regions).



- CloudFormation
- CloudFront
- CloudSearch
- CloudWatch
- Data Pipeline
- Direct Connect
- DynamoDB
- EC2
- ElastiCache
- Elastic Beanstalk
- Elastic MapReduce
- Elastic Transcoder NEW
- Glacier
- IAM
- OpsWorks NEW
- RDS
- Redshift NEW
- Route 53
- S3
- SES
- SNS
- SQS
- Storage Gateway
- SWF
- VPC

Amazon Web Services (2)

- Regions



US East (N. Virginia)

US West (Oregon)

US West (N. California)

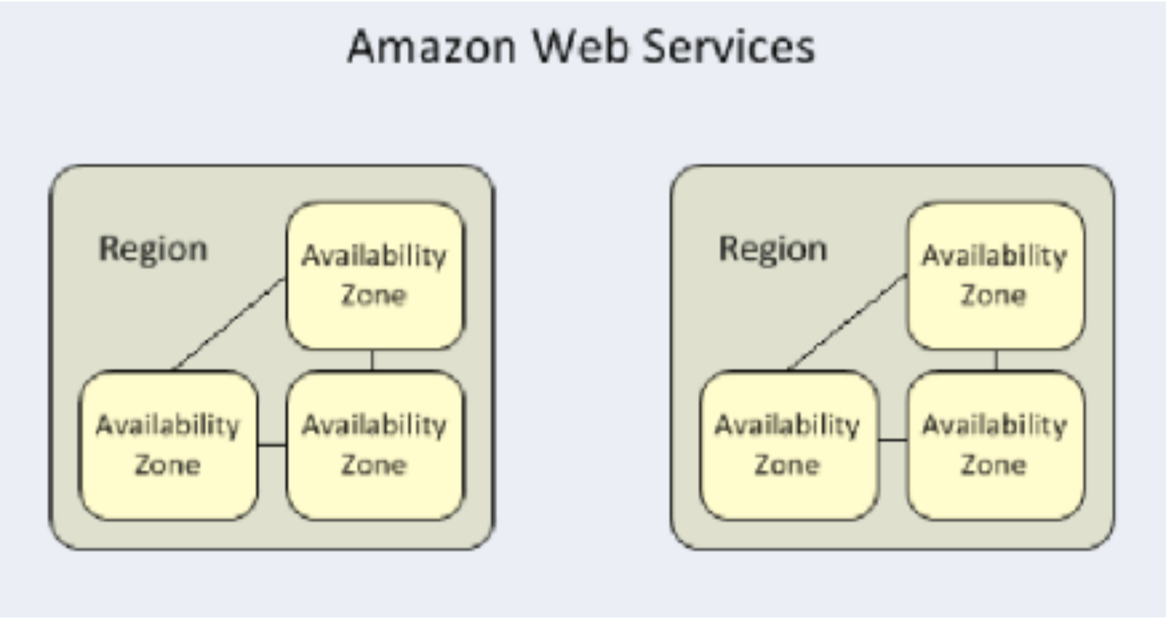
EU (Ireland)

Asia Pacific (Singapore)

Asia Pacific (Tokyo)

Asia Pacific (Sydney)

South America (São Paulo)



Where is Postgres in AWS?

- Redshift
  - PostgreSQL 8.0.2 on i686-pc-linux-gnu, compiled by GCC gcc (GCC) 3.4.2 20041017 (Red Hat 3.4.2-6.fc3), Redshift 1.0.516
  - Paraccel - http://www.paraccel.com/ http://en.wikipedia.org/wiki/ParAccel
  - Expensive (aprox 20u$s per day, one host). But there are more expensive services from other solutions using Hadoop.
  - Scale horizontally
- EC2
  - Yes, you can run an elephant here.
  - Call it “Cloud hosting”
  - Multi-region replication and WAN tunnel
  - Vertical scaling
  - You can use CC (Cluster Compute) instances.
- CloudWatch
  - Customizable monitoring
- No RDS for Postgres and there is no plan to be.

EC2

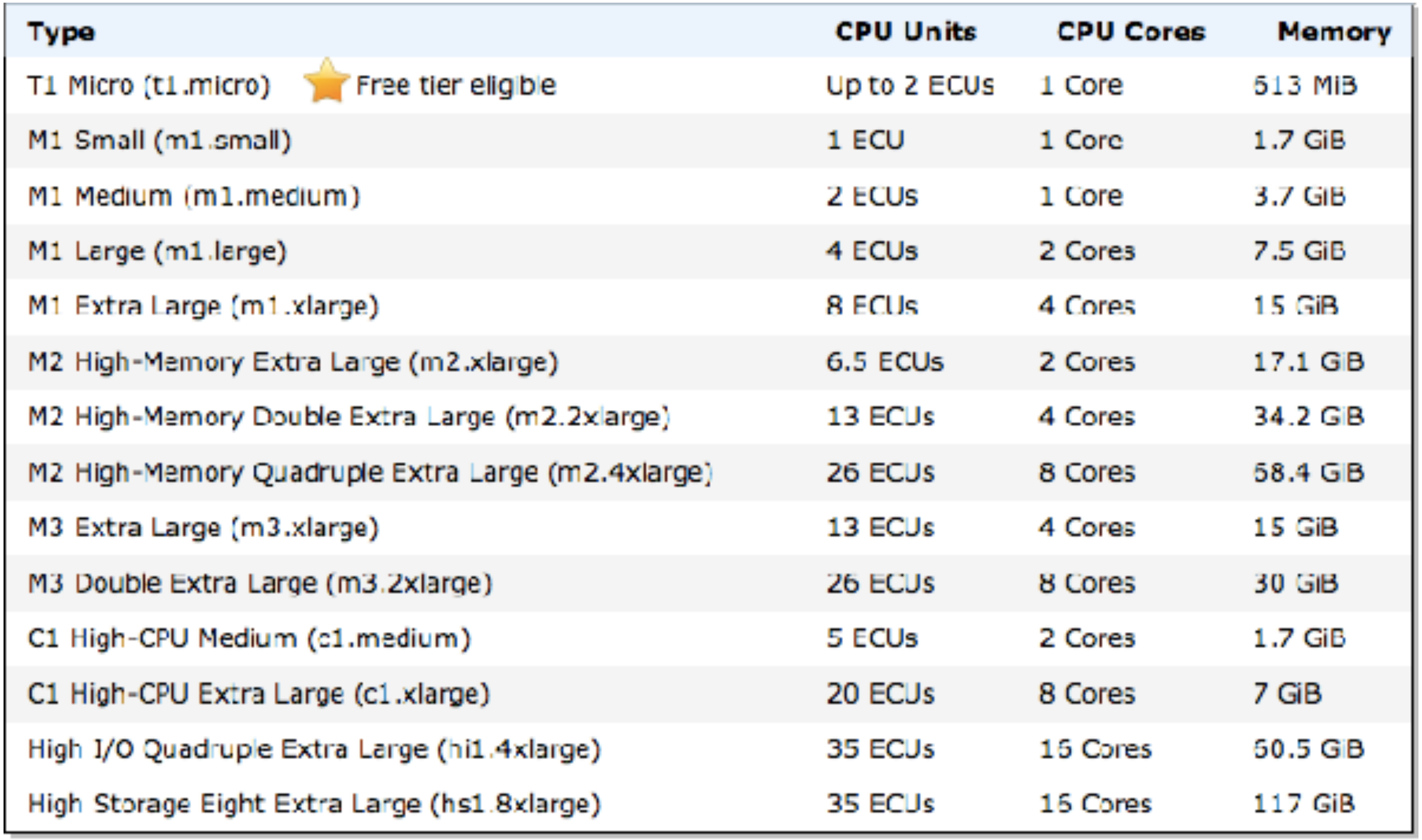
Running Postgres over the cloud, customized.

Before go to EC2, you may know:

- SLA (Service License Agreement): 99,95% ("Annual Uptime Percentage")
  - Real? 99,2%
- Prepare for easy failover and switchover
  - Never stay with only 1 host. Even Chuck Norris has a replica.
  - If you can go with a Master with Multi-AZ and PIOPS for durability, you'll rock!
- Prepare to scale up and down.
  - Reduce costs scaling down.
  - Add more gas to your servers scaling up.
- Go VLC from the beginning. ELBs for internal Load balancing.

EC2 Instances Sizes

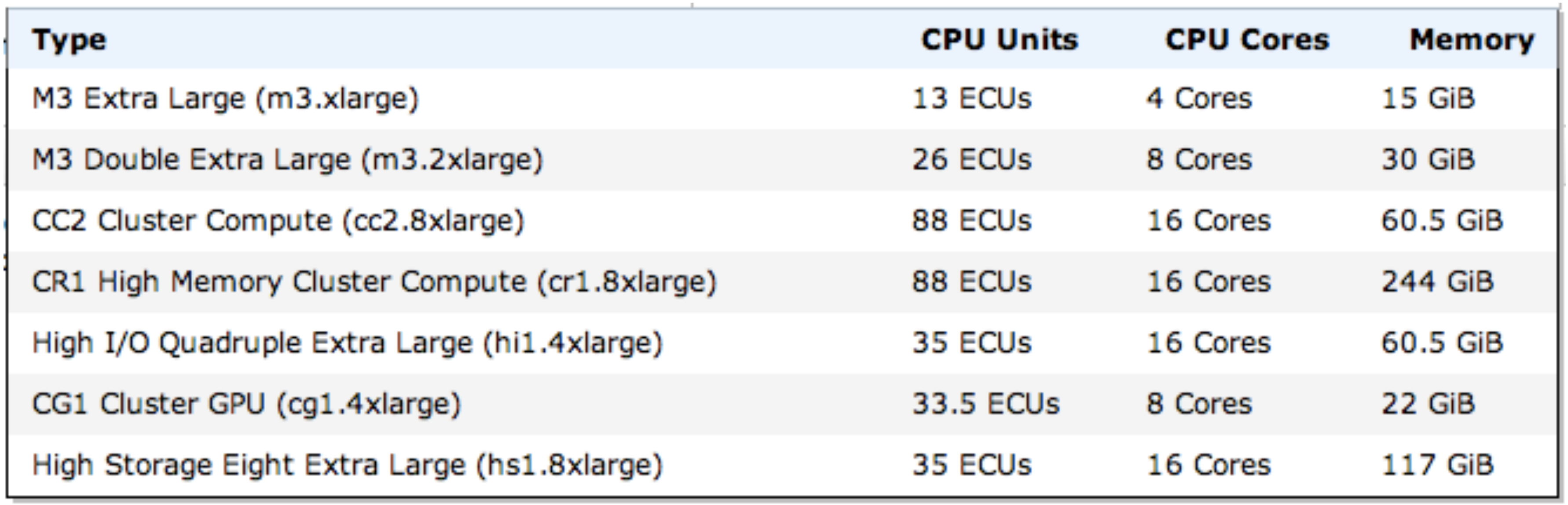
- Instances are classified by “sizes”



Type	CPU Units	CPU Cores	Memory
T1 Micro (t1.micro) ⭐ Free tier eligible	Up to 2 ECUs	1 Core	613 MiB
M1 Small (m1.small)	1 ECU	1 Core	1.7 GiB
M1 Medium (m1.medium)	2 ECUs	1 Core	3.7 GiB
M1 Large (m1.large)	4 ECUs	2 Cores	7.5 GiB
M1 Extra Large (m1.xlarge)	8 ECUs	4 Cores	15 GiB
M2 High-Memory Extra Large (m2.xlarge)	6.5 ECUs	2 Cores	17.1 GiB
M2 High-Memory Double Extra Large (m2.2xlarge)	13 ECUs	4 Cores	34.2 GiB
M2 High-Memory Quadruple Extra Large (m2.4xlarge)	26 ECUs	8 Cores	68.4 GiB
M3 Extra Large (m3.xlarge)	13 ECUs	4 Cores	15 GiB
M3 Double Extra Large (m3.2xlarge)	26 ECUs	8 Cores	30 GiB
C1 High-CPU Medium (c1.medium)	5 ECUs	2 Cores	1.7 GiB
C1 High-CPU Extra Large (c1.xlarge)	20 ECUs	8 Cores	7 GiB
High I/O Quadruple Extra Large (hi1.4xlarge)	35 ECUs	16 Cores	60.5 GiB
High Storage Eight Extra Large (hs1.8xlarge)	35 ECUs	16 Cores	117 GiB

Sizes for CC

- Instances are classified by “sizes”



Type	CPU Units	CPU Cores	Memory
M3 Extra Large (m3.xlarge)	13 ECUs	4 Cores	15 GiB
M3 Double Extra Large (m3.2xlarge)	26 ECUs	8 Cores	30 GiB
CC2 Cluster Compute (cc2.8xlarge)	88 ECUs	16 Cores	60.5 GiB
CR1 High Memory Cluster Compute (cr1.8xlarge)	88 ECUs	16 Cores	244 GiB
High I/O Quadruple Extra Large (hi1.4xlarge)	35 ECUs	16 Cores	60.5 GiB
CG1 Cluster GPU (cg1.4xlarge)	33.5 ECUs	8 Cores	22 GiB
High Storage Eight Extra Large (hs1.8xlarge)	35 ECUs	16 Cores	117 GiB

Prepare for the disaster

- EBS hang
- Database crash
  - (Don’t remember last time I saw this for Postgres, even in AWS)
- Common Human Errors:
  - *DROP DATABASE production;*
  - *DELETE FROM really_important_table;*
  - *DROP TABLE fire_me_if_you_can;*
- Don’t *overcloud*. And do not enter the *rage or panic states*.

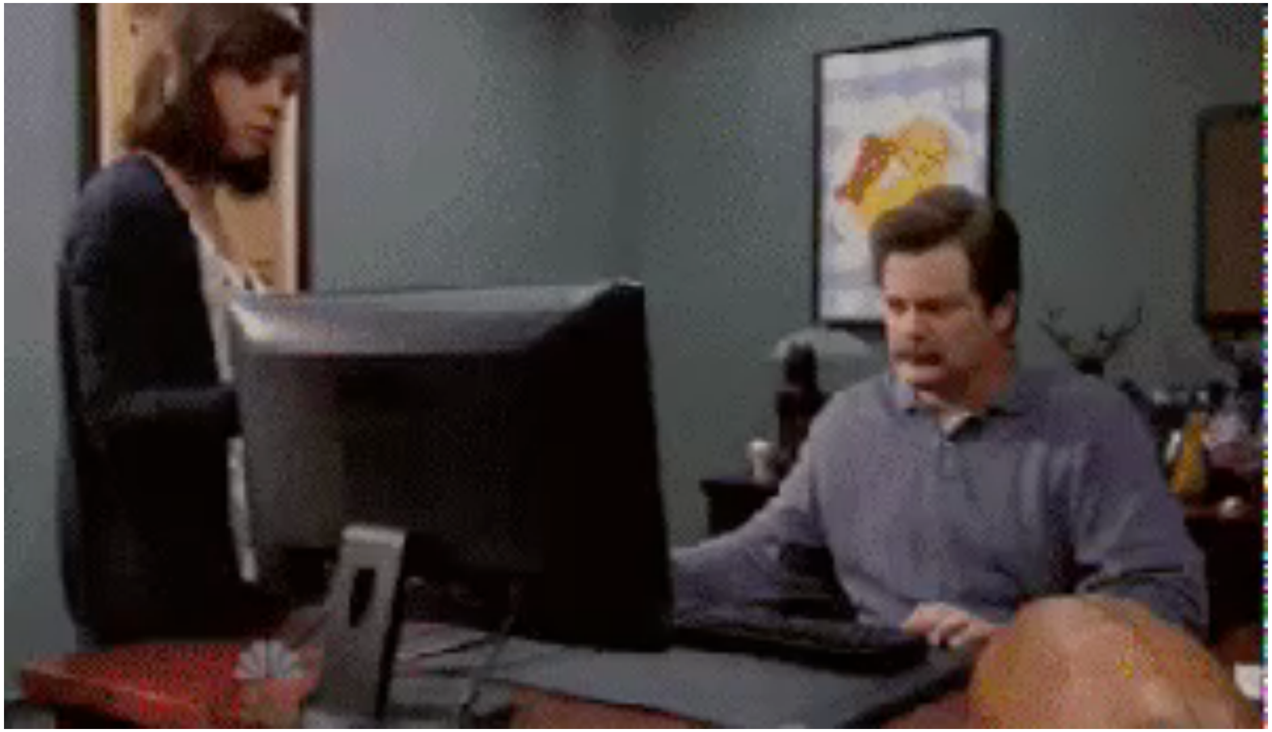
How?

- Multi AZ master with PIOPs
- Rebuild replicas after failover
  - *Oh, wait! 9.3 doesn’t need this for cascade slaves!*
- *Document step by step the PANIC MODEs in the corresponding runbook on your Confluence/Wiki.*
- *A bare metal minimal structure is sometimes a good approach.*

- *Obviously, there are non-conventional ways...*

Other non-conventional procedures

Rage



Storages

- S3
  - Store cheap and slow. Perfect for backup pieces.
  - Very durable (99.999999999%)
- Glacier
  - Even more slower, even more cheaper.
- EBS 100, 200, 400 IOPS
  - Lower cost and persistent.
  - *Use always EBS optimized option (available from m1.large and on)*
  - I/O unpredictable sometimes.
  - *(NUM_IOPS * BLOCK_SIZE) / 1024 = Megabytes/Sec*
    - That is 400 IOPS is 3MB at 8K block size
- PIOPS 1000,2000,4000
  - Want better performance? Guaranteed throughput.
  - Lower failure rate.
- SSD (ephemeral)
  - You don’t want to have you main servers with this setup, unless you’re very sure what you want.
  - 150.000 IOPS (YMMV)
  - Temporal tablespaces or unlogged tables

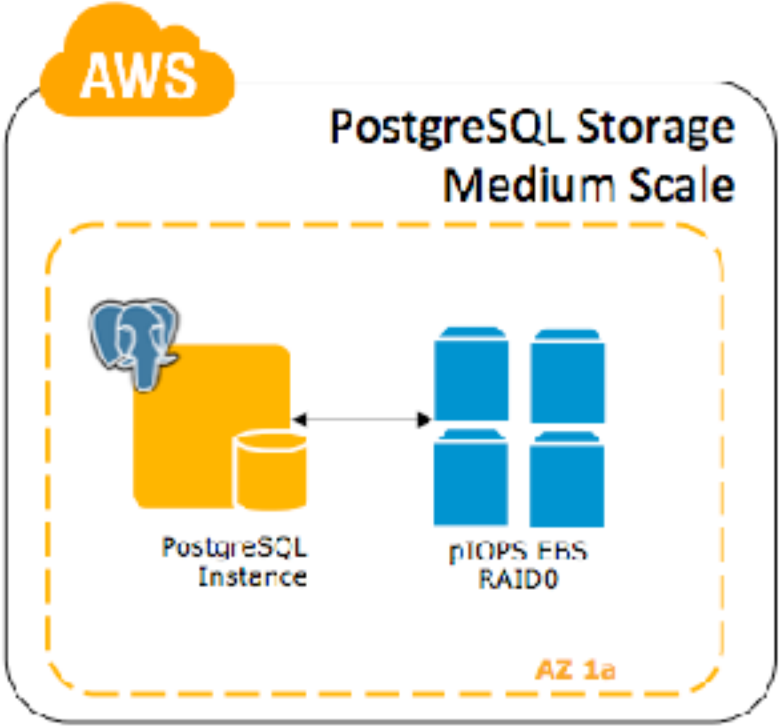
Storages (2)

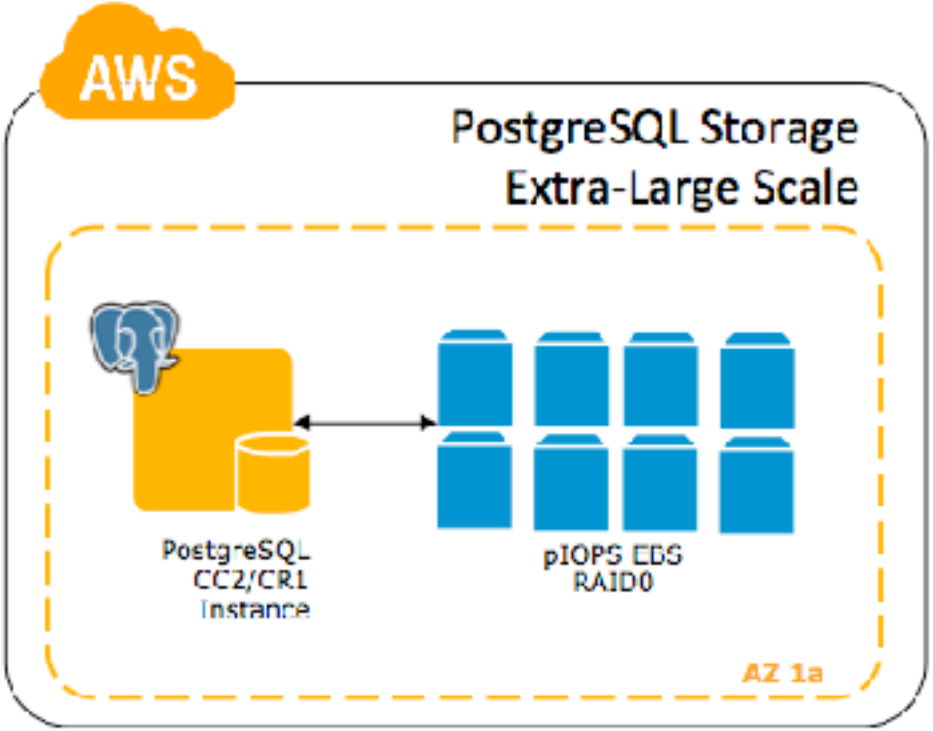
Internally, EBS manage
against a 8k writes, you
PostgreSQL compiled in
performance in a wide

More information at:
http://docs.aws.amazo
EBSPerformance.html

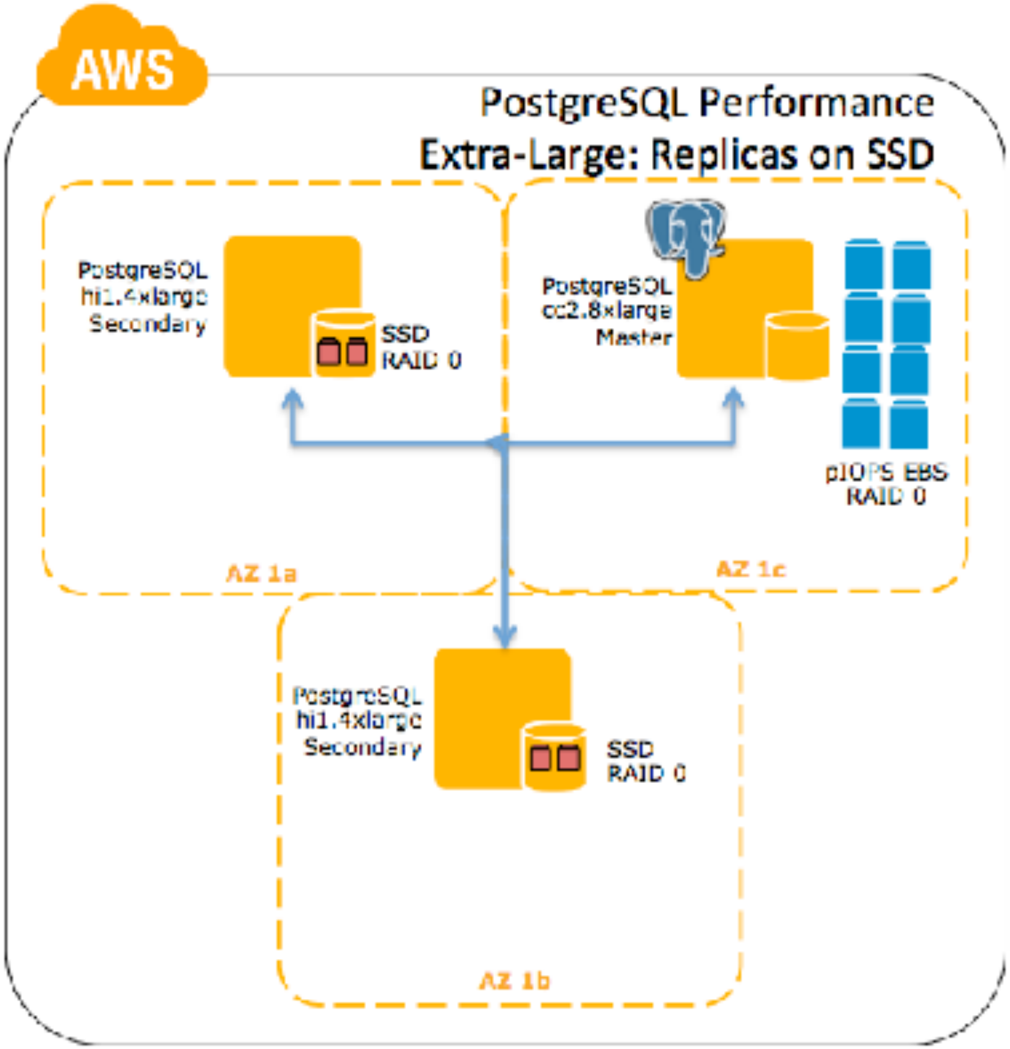
```
[root@ip-10-248-82-137 ~]# mkdir -p /data
[root@ip-10-248-82-137 ~]# mkfs.xfs /dev/sdb
meta-data=/dev/sdb               isize=256    agcount=4, agsize=1966080 blks
         =                       sectsz=512   attr=2
data     =                       bsize=4096   blocks=7864320, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =internal log           bsize=4096   blocks=3840, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0

[root@ip-10-248-82-137 ~]# mount -t xfs /dev/sdb /data
[root@ip-10-248-82-137 ~]# useradd postgres
```

Storages (2)


AWS
PostgreSQL Storage
Medium Scale
PostgreSQL
Instance
pIOPS EBS
RAID0
AZ 1a


AWS
PostgreSQL Storage
Extra-Large Scale
PostgreSQL
CC2/CR1
Instance
pIOPS EBS
RAID0
AZ 1a

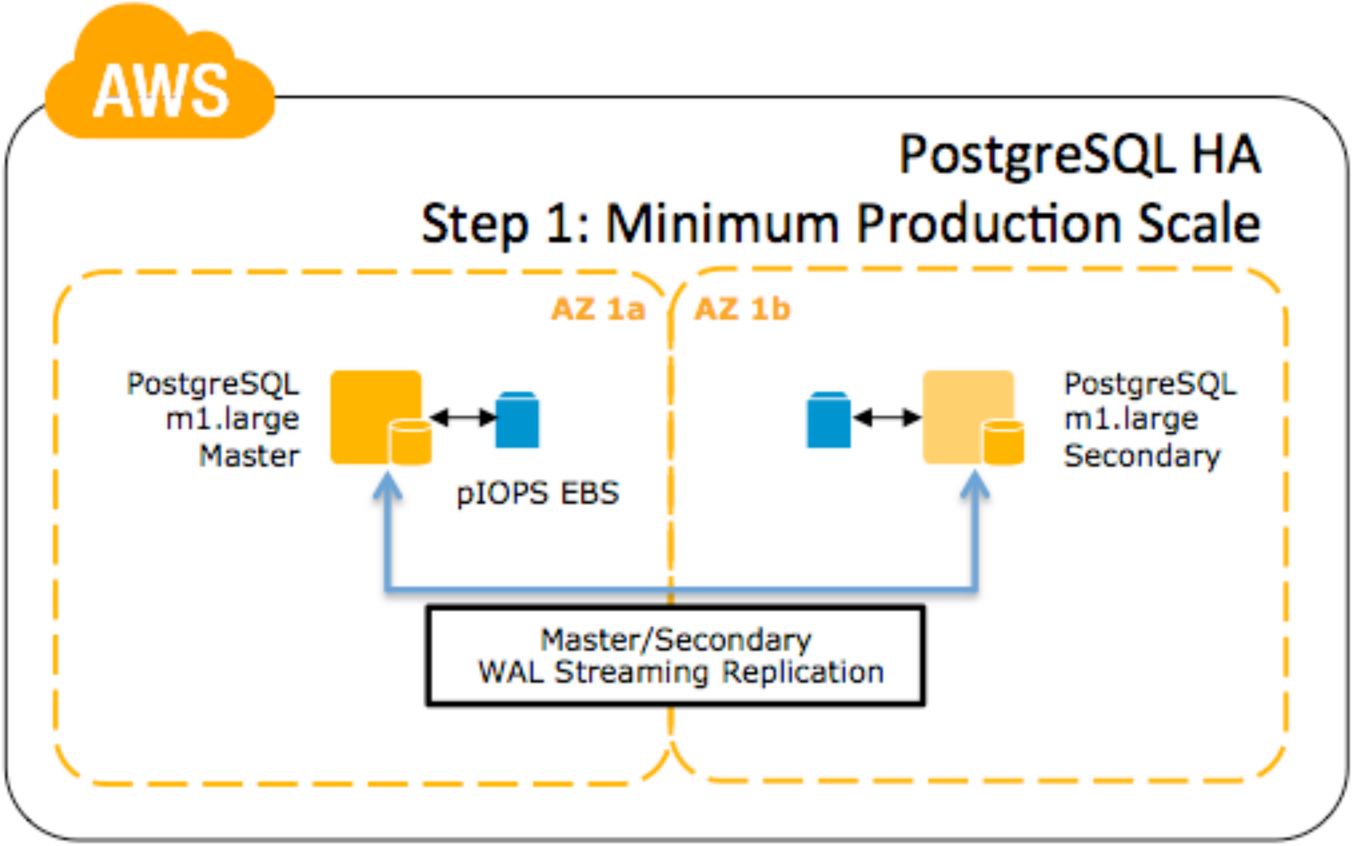
Storages (3)


AWS
PostgreSQL Performance
Extra-Large: Replicas on SSD
PostgreSQL
hi1.4xlarge
Secondary
SSD
RAID 0
PostgreSQL
cc2.8xlarge
Master
pIOPS EBS
RAID 0
AZ 1a
AZ 1c
PostgreSQL
hi1.4xlarge
Secondary
SSD
RAID 0
AZ 1b

Scale up

- Horizontally
  - Replicas
  - More replicas
  - More cascade slaves, specially if you are planning to run 9.3
  - Try to use CC (Cluster Compute technology)
  - Sharding
    - *What did you say? What? Excuse me, don't speak english.*
- Vertically
  - Increase the instance type
  - Try to fit your RSS in memory
  - Move to PIOPs (you can move back and forth from EBS)
    - http://aws.amazon.com/contact-us/ebs_volume_limit_request/

PostgreSQL Replication



PostgreSQL Replication (2)

- 2 ways to setup a slave:
  - pg_basebackup
  - Snapshot the EBS devices
    - I don’t recommend this. Snapshots don’t warranty consistency in a 100%.

- Don’t underestimate the bandwidth across the zones. For 200GB a snapshot took almost the same time as a pg_basebackup. Yeah.
- Really?
- Yeah. Told ya’.

Don’t overallocate! Scale down!

- Turn off PIOPS
- Kill non used slaves
- downgrade the instance type
- Maintain S3 clean.
- Archive your data.

… unless you're this kid:

Backups

- Use S3 to store pieces (you’ll need to encrypt)
  - Generate a pair-key

    -
      ```
      openssl req -x509 -nodes -days 100000 -newkey rsa:2048 -keyout private.pem -out public.pem -subj '/'
      ```
  - Encrypt a piece/full_plain_backup:

    -
      ```
      $PGBINHOME/pg_dumpall -U $DBUSER | gzip -c | openssl smime -encrypt -aes256 -binary -outform DEM -out backups.zip.enc $PUB_KEY_PATH  || { BACKUP_STATUS=5 ; echo"failed to dump database" ; mail "failed to dump database" $BACKUP_STATUS ;   }
      ```
  - Decrypt:

    -
      ```
      openssl smime -decrypt -in $TEMP_ENC_BACKUP -binary -inform DEM -inkey $PRIVATE_KEY -out $TEMP_DEC_BACKUP || { echo Decryption failed ; exit 1500; }
      ```
- Tools:
  - https://github.com/wal-e/wal-e

Snaphots

- Shoot the snapshot
  - `SELECT pg_start_backup(‘label’,true);`
  - `ec2-create-snapshot -d "postgres clon" vol-24592c0e`
  - `SELECT pg_stop_backup();`

Output:

```
SNAPSHOT snap-219c1308 vol-24592c0e pending 2012-12-03T01:34:12+0000
052088341151 10 postgres clon
```

Snaphots/Restore

```
$ ec2-describe-snapshots
SNAPSHOT snap-219c1308 vol-24592c0e completed 2012-12-03T01:34:12+0000 100%
052088341151 10 postgres clon

$ ec2-create-volume --snapshot snap-219c1308 --availability-zone eu-west-1c
VOLUME vol-eb1561c1 10 snap-219c1308 eu-west-1c creating
2012-12-03T10:13:44+0000

$ ec2-attach-volume -i i-96ec5edd -d /dev/sdc vol-eb1561c1
ATTACHMENT vol-eb1561c1 i-96ec5edd /dev/sdc attaching
2012-12-03T10:23:37+0000

$ dmesg | tail
[3889500.959401] blkfront: xvdc: barrier or flush: disabled
[3889500.961193] xvdd: unknown partition table
[root@ip-10-208-8-123 ~]# mkdir -p /data
[root@ip-10-208-8-123 ~]# chown -R postgres: /data
# echo “/dev/xvdd /data auto noatime,noexec,nodiratime 0 0” >> /etc/fstab
# mount -a
```

* Start postgres!

CloudWatch

- CloudWatch does not graph.
- Customizable monitor.
  - The following example shows how we can graph using a normal query.

```
$ ec2-monitor-instances i-08fe4e43
i-08fe4e43 monitoring-pending
# while true ; do CloudWatch-1.0.13.4/bin/mon-put-data --metric-name backends --
namespace Postgres --dimensions "InstanceId=i-08fe4e43" --value `psql -Upostgres -Atc
'SELECT sum(numbackends) FROM
pg_stat_database'` --timestamp `date +%Y-%m-%dT%H:%M:%S.000Z` ; sleep 60 ; done
# CloudWatch-1.0.13.4/bin/mon-list-metrics | grep -i backends
backends Postgres {InstanceId=i-08fe4e43}
# CloudWatch-1.0.13.4/bin/mon-get-stats backends --namespace Postgres --statistics
"Average,Maximum" --dimensions "InstanceId=i-08fe4e43" --start-time
2013-03-04T23:00:00.000Z
2013-03-05 13:15:00 1.0 1.0 None
2013-03-05 13:16:00 1.0 1.0 None
2013-03-05 13:17:00 1.0 1.0 None
2013-03-05 13:22:00 1.0 1.0 None
2013-03-05 13:23:00 1.0 1.0 None
2013-03-05 13:24:00 1.0 1.0 None
…
```

Configuration

Specific Amazon-configuration at the postgresql.conf:

```
random_page_cost = 1
```

And know is when you ask. Don’t. I’ll explain.

Tools

- Web client
  - Nice
  - Rage clicking
    - Yeah, next - next - click - click. Super-click for check snaphots process status or instances creation (sometimes hangs)
- API cli
  - Command line API
- External tools
  - https://github.com/timkay/aws (Perl)
  - https://github.com/aws/aws-cli (Python)

Featured documents

You may want to read the following links:

- RDBMS on the Cloud, Amazon Library: http://media.amazonwebservices.com/AWS_RDBMS_PostgreSQL.pdf
- Benchmarking PG on AWS 4000 PIOPs EBS instances: http://www.palominodb.com/blog/2013/05/08/benchmarking-postgres-aws-4000-piops-ebs-instances
- MySQL patters on AWS by Jay Edwards and Ben Black: http://www.slideshare.net/palominodb/mysql-patterns-in-aws#btnNext
- Laine Campbell at Velocity http://cdn.oreillystatic.com/en/assets/1/event/94/Using%20Amazon%20Web%20Services%20for%20MySQL%20at%20Scale%20Presentation.pdf

Redshift

An old elephant for massive processing.

Soup of features and links

- http://aws.amazon.com/documentation/redshift/
- Query parallelizing
- Distribution
- Data importation only available from
    - S3
    - DynamoDB
- Expensive
- Manages lower amount of data than Hadoop (i.e.) but is good for query with more frequently.
- Scaling is horizontal
    - Add nodes for increase performance
- http://docs.aws.amazon.com/redshift/latest/gsg/getting-started-create-sample-db.html
- http://www.slideshare.net/Hapyrus/amazon-redshift-is-10x-faster-and-cheaper-than-hadoop-hive

Different feelings

http://www.wired.com/wiredenterprise/2013/08/memsql-and-amazon/ -- Why Some Startups Say the Cloud Is a Waste of Money

http://programming.oreilly.com/2013/06/ins-and-outs-of-running-mysql-on-aws.html#! by Laine Campbell

Prices and Free tier*

Is not a topic of this talk, but you can find the prices here:
http://aws.amazon.com/ec2/pricing/

Free Tier*

As part of AWS’s Free Usage Tier, new AWS customers can get started with Amazon EC2 for free. Upon sign-up, new AWS customers receive the following EC2 services each month for one year:

750 hours of EC2 running Linux/UNIX or RHEL Micro instance usage

750 hours of EC2 running Microsoft Windows Server Micro instance usage

750 hours of Elastic Load Balancing plus 15 GB data processing

30 GB of Amazon EBS Standard volume storage plus 2 million IOs and 1 GB snapshot storage

15 GB of bandwidth out aggregated across all AWS services

1 GB of Regional Data Transfer

Reality

Colleagues reaction when AWS isn't available/region outage and they are running with their own iron:

Thanks!

Contact us!
We are hiring!
emanuel@palominodb.com