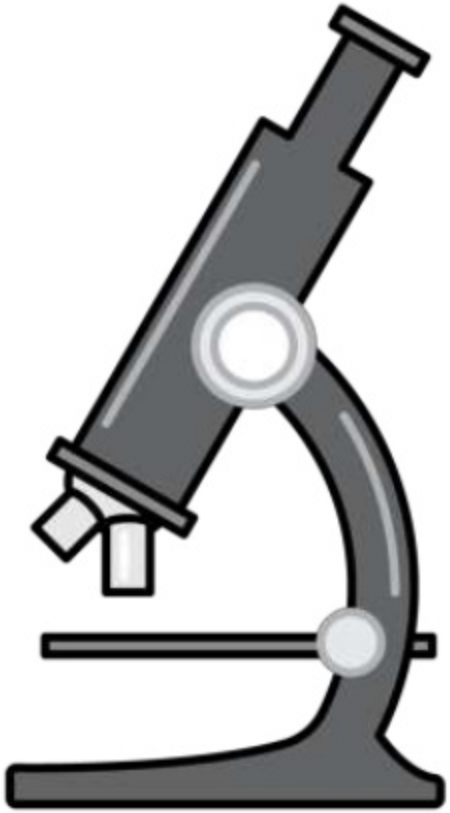
Deep Dive on Amazon Relational Database Service

Prahlad Rao, Solutions Architect

Sept 13, 2016

What to expect

- Amazon RDS overview (super quick)
- Security
- Metrics and monitoring
- High availability
- Scaling on RDS
- Backups and snapshots
- Migrating to RDS
- Q&A!



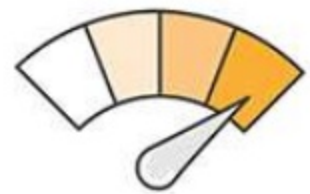
Amazon Relational Database Service (Amazon RDS)

No infrastructure management

Application compatibility

Instant provisioning

Cost-effective



Scale up/down

Amazon RDS engines

Commercial

Open source

Amazon Aurora

Selected Amazon RDS customers

Selected Amazon Aurora customers

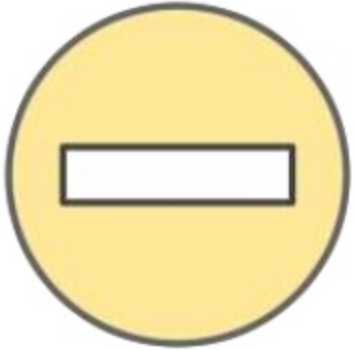
Trade-offs with a managed service

Fully managed host and OS

- No access to the database host operating system
- Limited ability to modify configuration that is managed on the host operating system
- No functions that rely on configuration from the host OS

Fully managed storage

- Max storage limits
  - Microsoft SQL Server—4 TB
  - MySQL, MariaDB, PostgreSQL, Oracle—6 TB
  - Aurora—64 TB
- Growing your database is a process

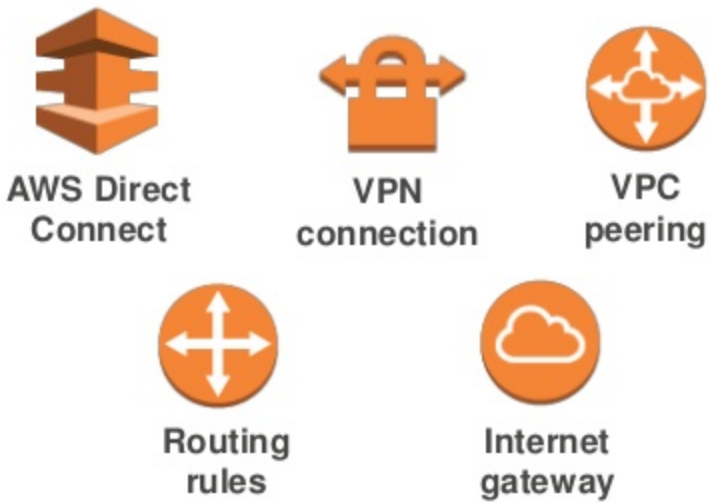


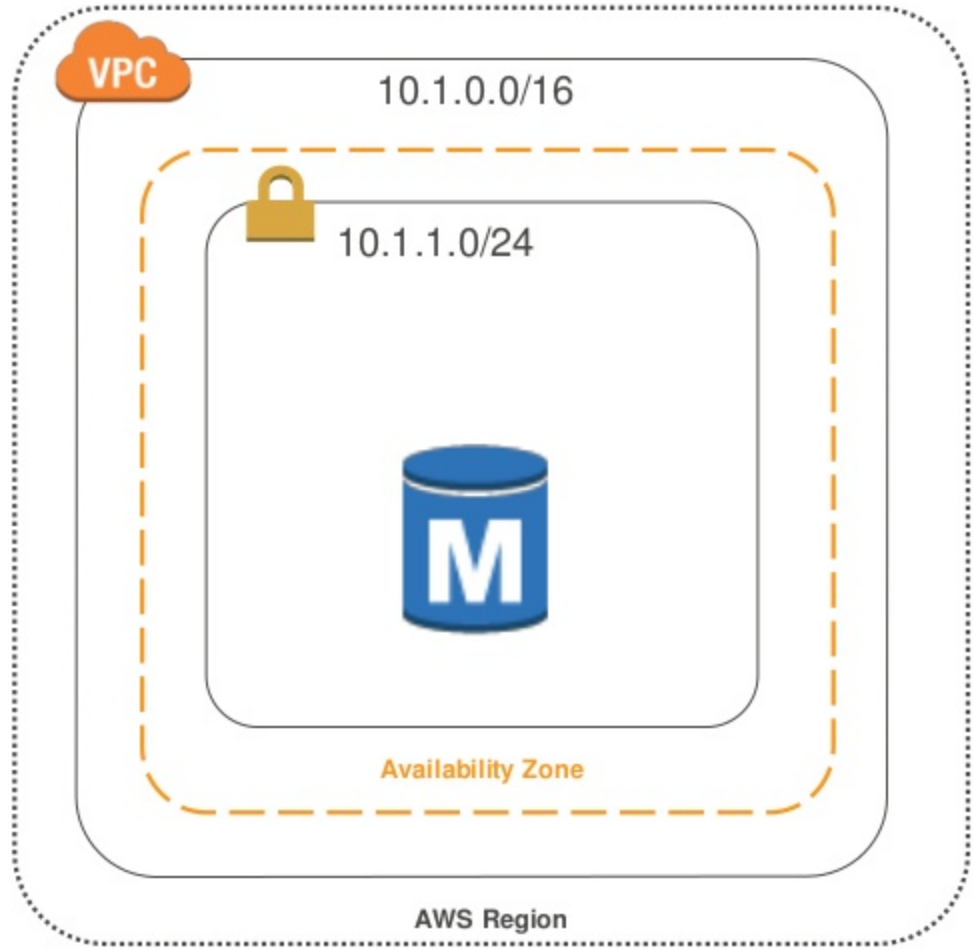
Security

Amazon Virtual Private Cloud (Amazon VPC)

Securely control network configuration

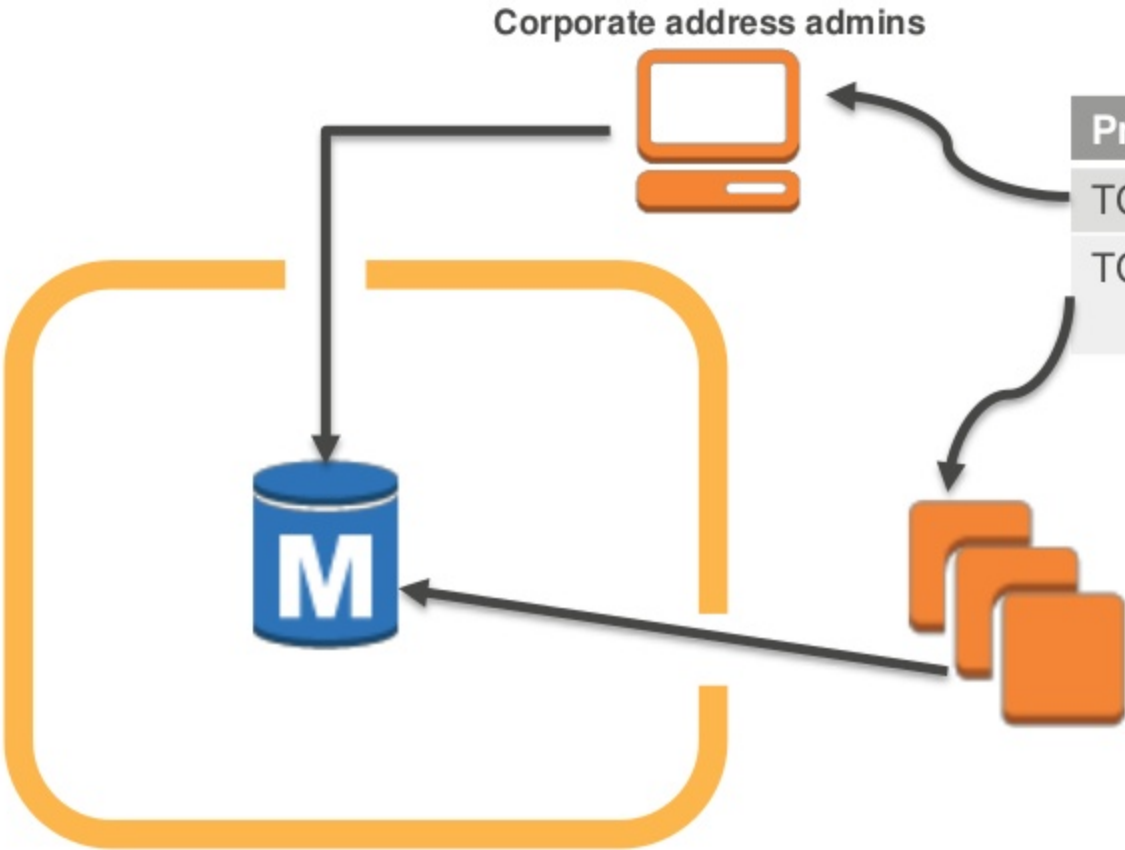
Manage connectivity





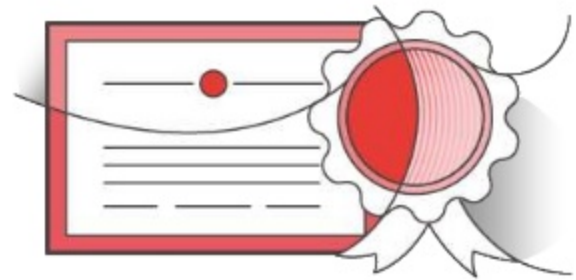
Security groups

Database IP firewall protection



Protocol	Port Range	Source
TCP	3306	172.31.0.0/16
TCP	3306	“Application security group”

Compliance



Singapore MTCS

27001/9001
27017/27018

Compliance

MySQL and Oracle

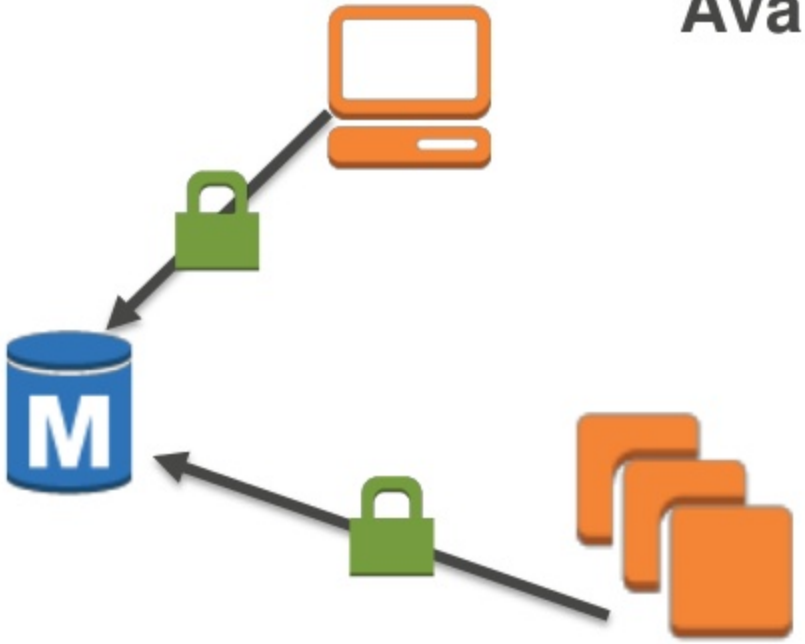
- SOC 1, 2, and 3
- ISO 27001/9001
- ISO 27017/27018
- PCI DSS
- FedRAMP
- HIPAA BAA
- UK government programs
- Singapore MTCS

SQL Server and PostgreSQL

- SOC 1, 2, and 3
- ISO 27001/9001
- ISO 27017/27018
- PCI DSS
- UK government programs
- Singapore MTCS

SSL

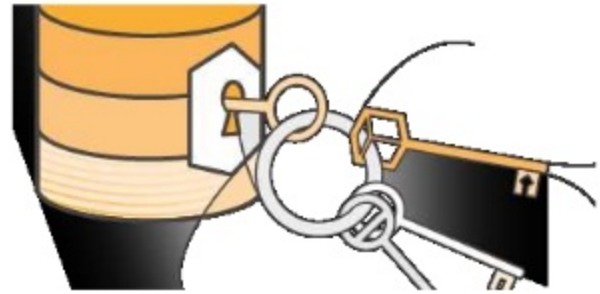
Database traffic encryption



Available for all six engines

At-rest encryption

- DB instance storage
- Automated backups
- Read Replicas
- Snapshots



- **Available for all six engines**
- **No additional cost**
- **Support compliance requirements**

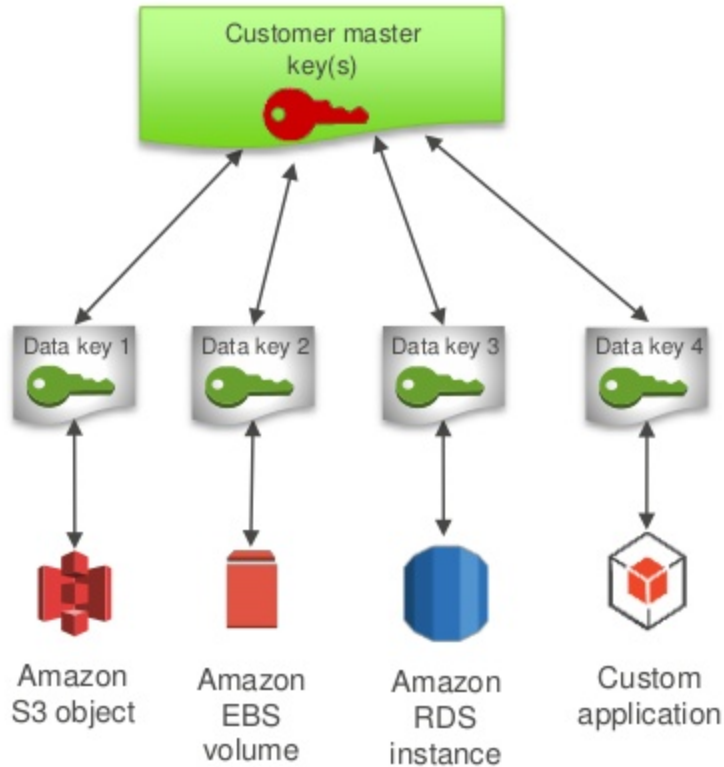
AWS KMS—RDS standard encryption

Two-tiered key hierarchy using envelope encryption:

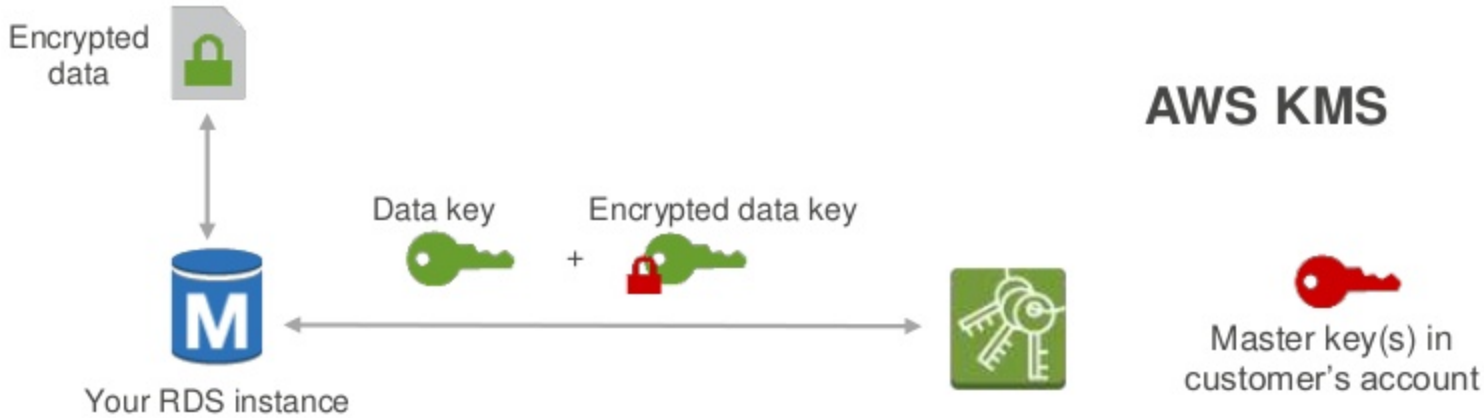
- Unique data key encrypts customer data
- AWS KMS master keys encrypt data keys

Benefits:

- Limits risk of compromised data key
- Better performance for encrypting large data
- Easier to manage small number of master keys than millions of data keys
- Centralized access and audit of key activity



How keys are used to protect your data



1. RDS instance requests encryption key to use to encrypt data, passes reference to master key in account
2. Client request authenticated based on permissions set on both the user and the key
3. A unique data encryption key is created and encrypted under the KMS master key
4. Plaintext and encrypted data key returned to the client
5. Plaintext data key used to encrypt data and then deleted when practical
6. Encrypted data key is stored; it's sent back to KMS when needed for data decryption

Enabling encryption

AWS Command Line Interface (AWS CLI)

```
aws rds create-db-instance --region us-west-2 --db-instance-identifier sg-cli-test \
--allocated-storage 20 --storage-encrypted \
--db-instance-class db.m4.large --engine mysql \
--master-username myawsuser --master-user-password myawsuser
```

```
aws rds create-db-instance --region us-west-2 --db-instance-identifier sg-cli-test1 \
--allocated-storage 20 --storage-encrypted  --kms-key-id xxxxxxxxxxxxxxxxxx \
--db-instance-class db.m4.large --engine mysql \ --master-username myawsuser
--master-user-password myawsuser
```

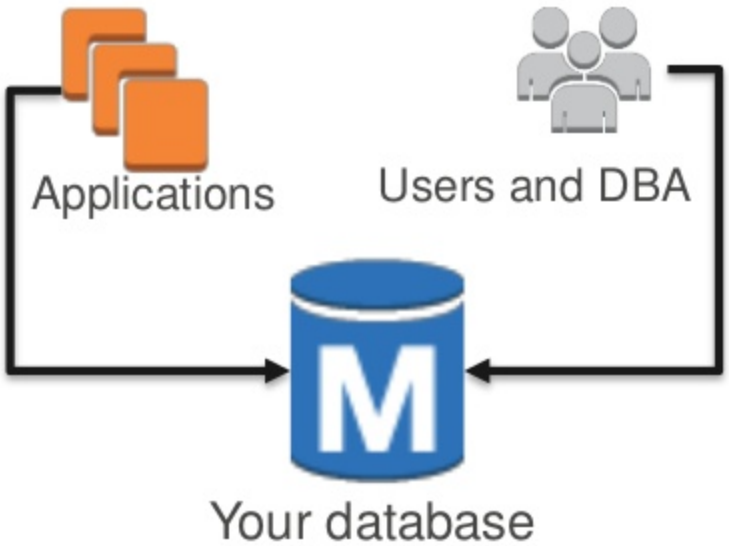
Amazon RDS + AWS KMS useful hints

- You can only encrypt on new database creation
- Encryption cannot be removed
- Master and Read Replica must be encrypted
- Unencrypted snapshots cannot be restored to encrypted DB
    - Aurora will allow this
    - You can create encrypted copies of your unencrypted snapshots
- Cannot restore MySQL to Aurora or Aurora to MySQL
- Cannot copy snapshots or replicate DB across regions

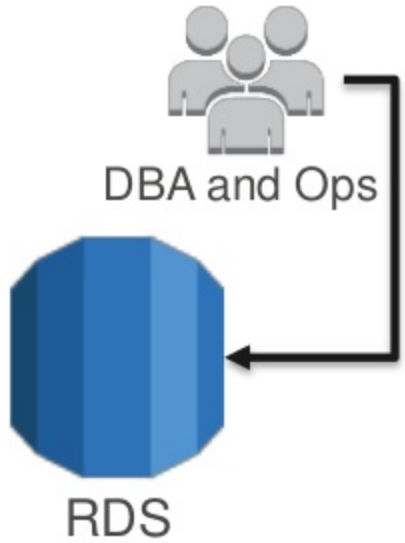
IAM governed access

You can use AWS Identity and Access Management (IAM) to control who can perform actions on RDS

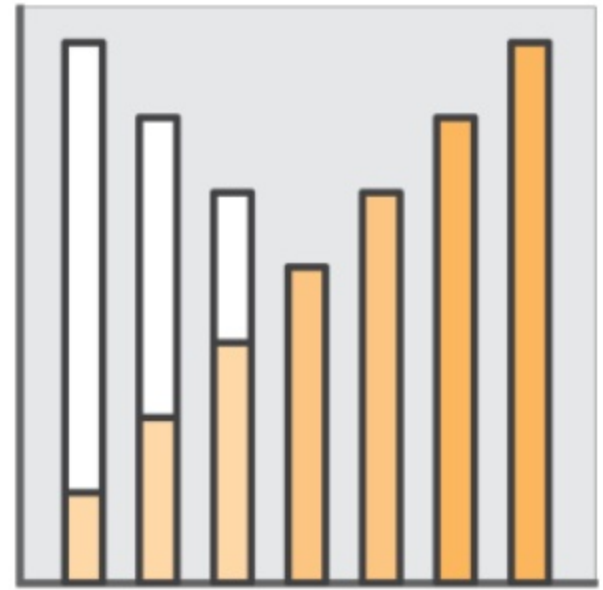
Controlled with database grants



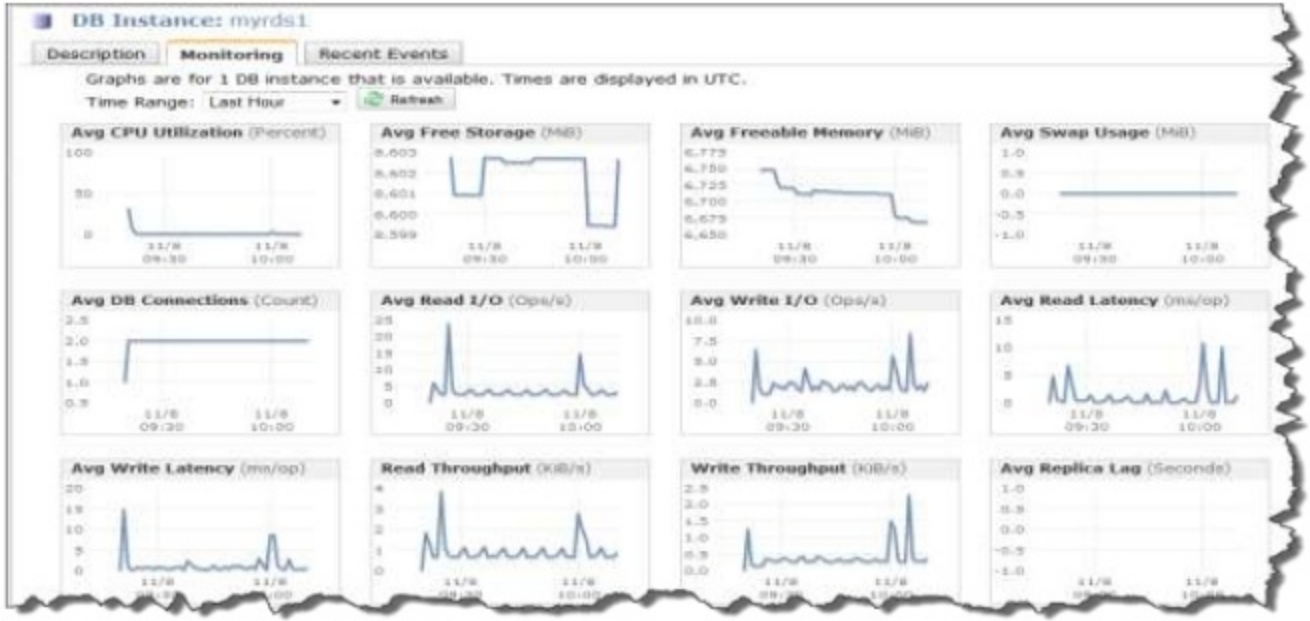
Controlled with IAM



Metrics and monitoring



Standard monitoring



Amazon CloudWatch metrics for Amazon RDS

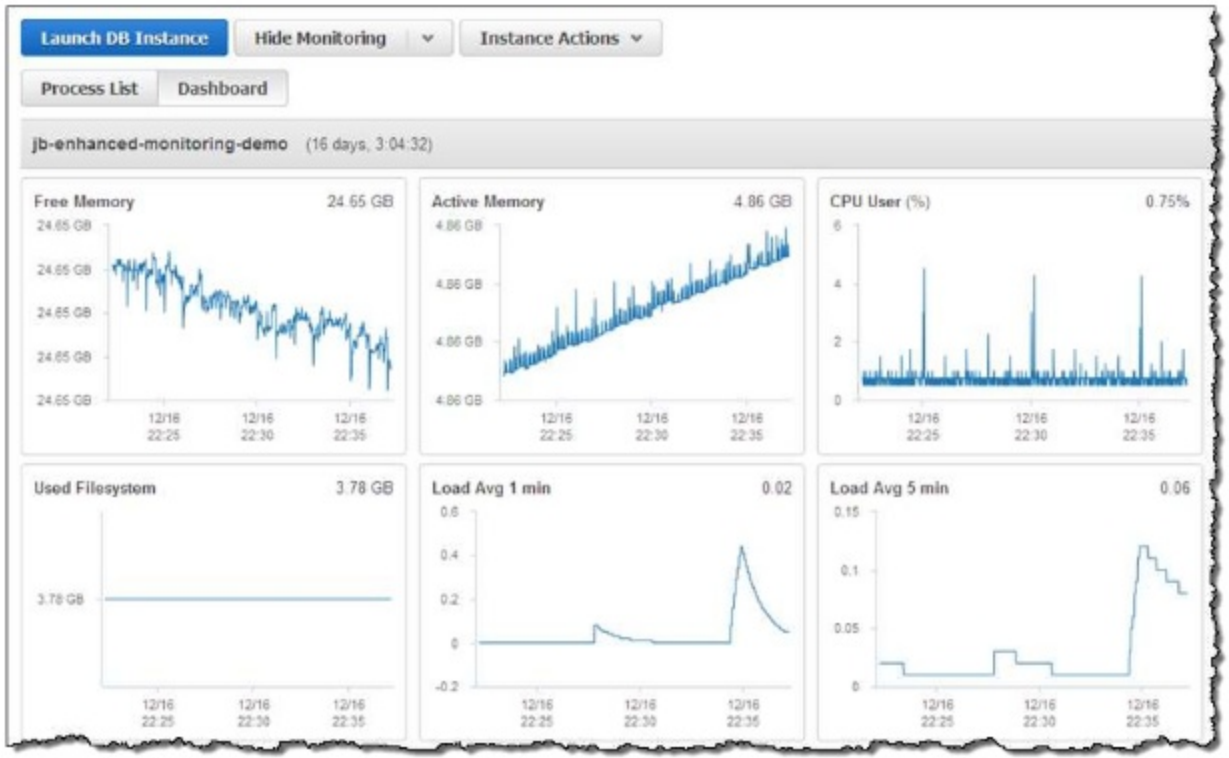
- CPU utilization
- Storage
- Memory
- Swap usage
- DB connections
- I/O (read and write)
- Latency (read and write)
- Throughput (read and write)
- Replica lag
- Many more

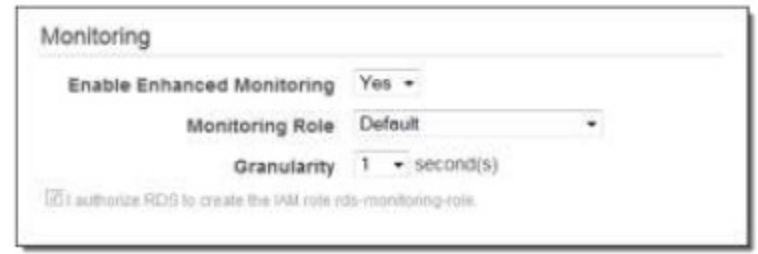
Amazon CloudWatch Alarms

- Similar to on-premises custom monitoring tools

Enhanced Monitoring

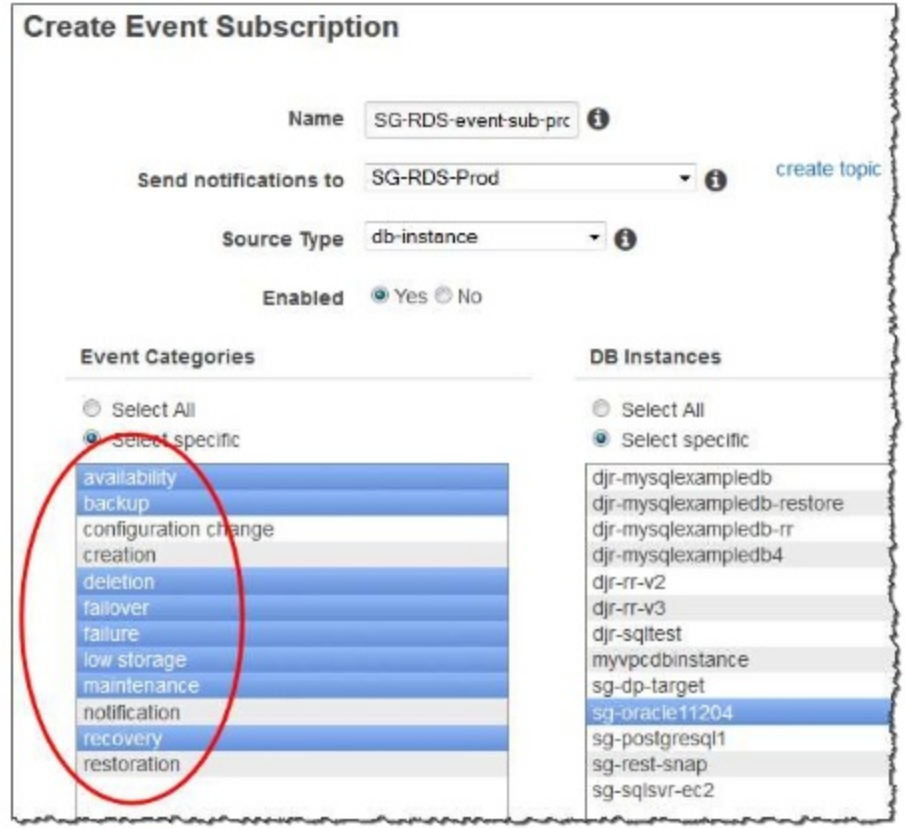
Access to over 50 new CPU, memory, file system, and disk I/O metrics as low as 1 second intervals



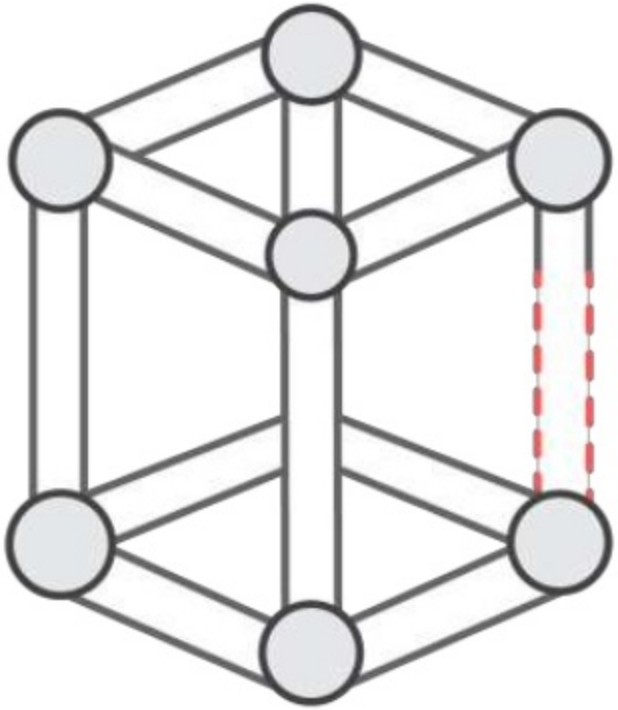


Event notifications

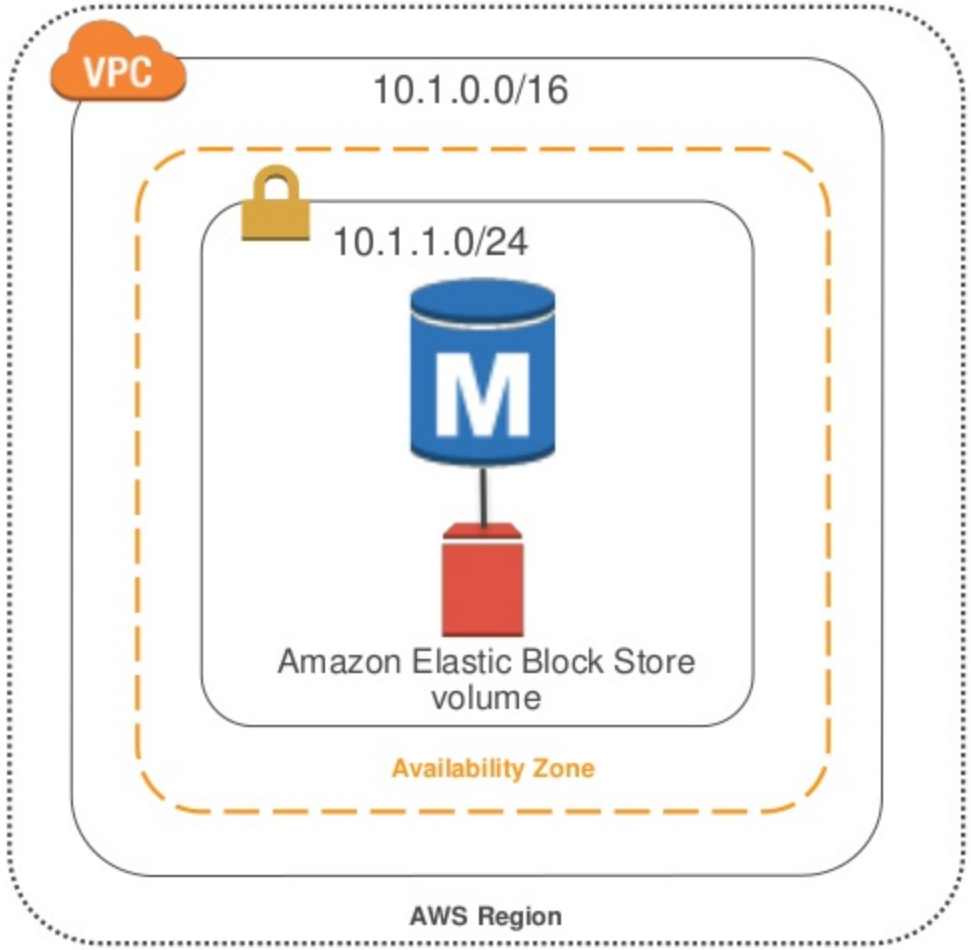
- Uses Amazon Simple Notification Service (Amazon SNS) to notify users when an event occurs
- 17 different event categories (availability, backup, configuration change, and so on)



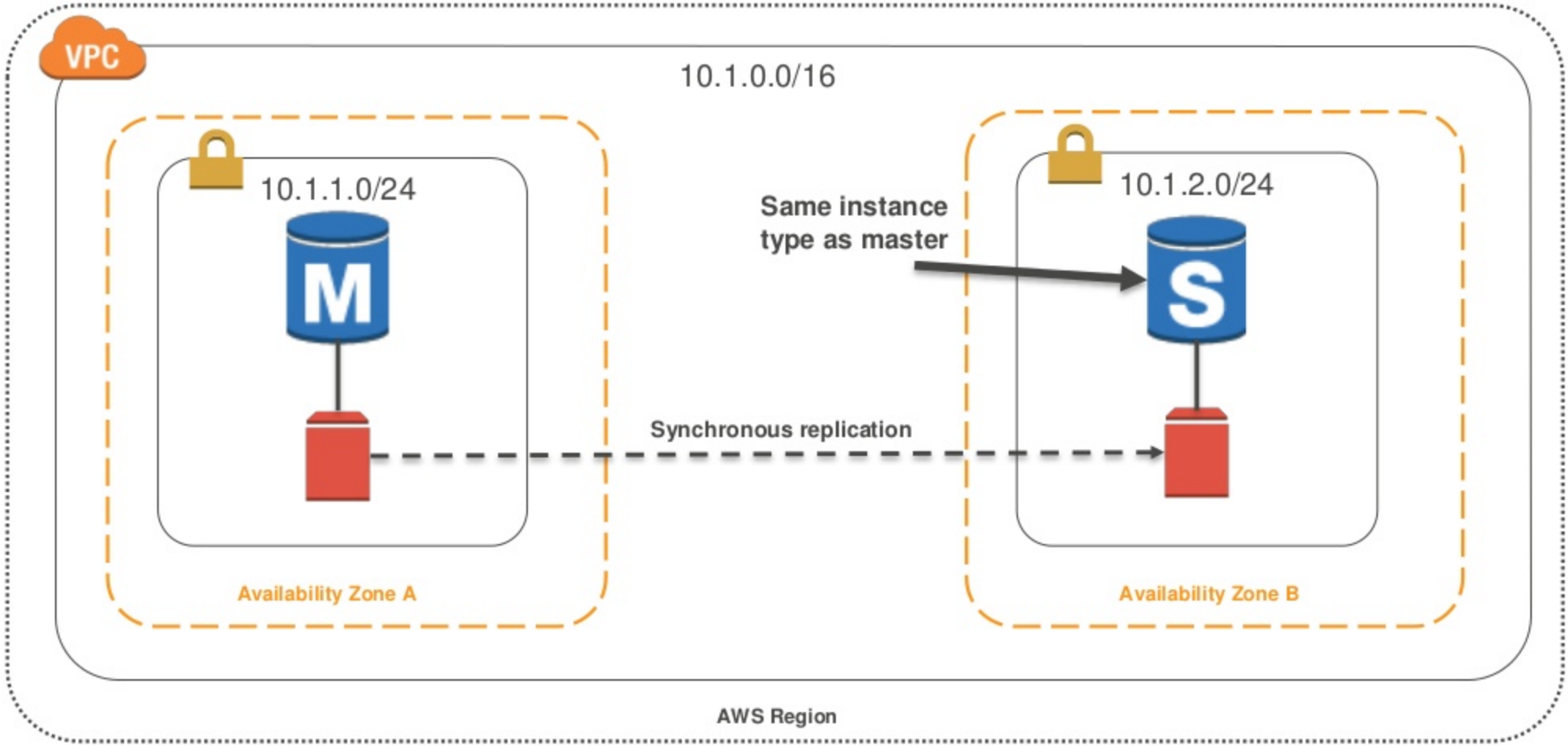
High availability



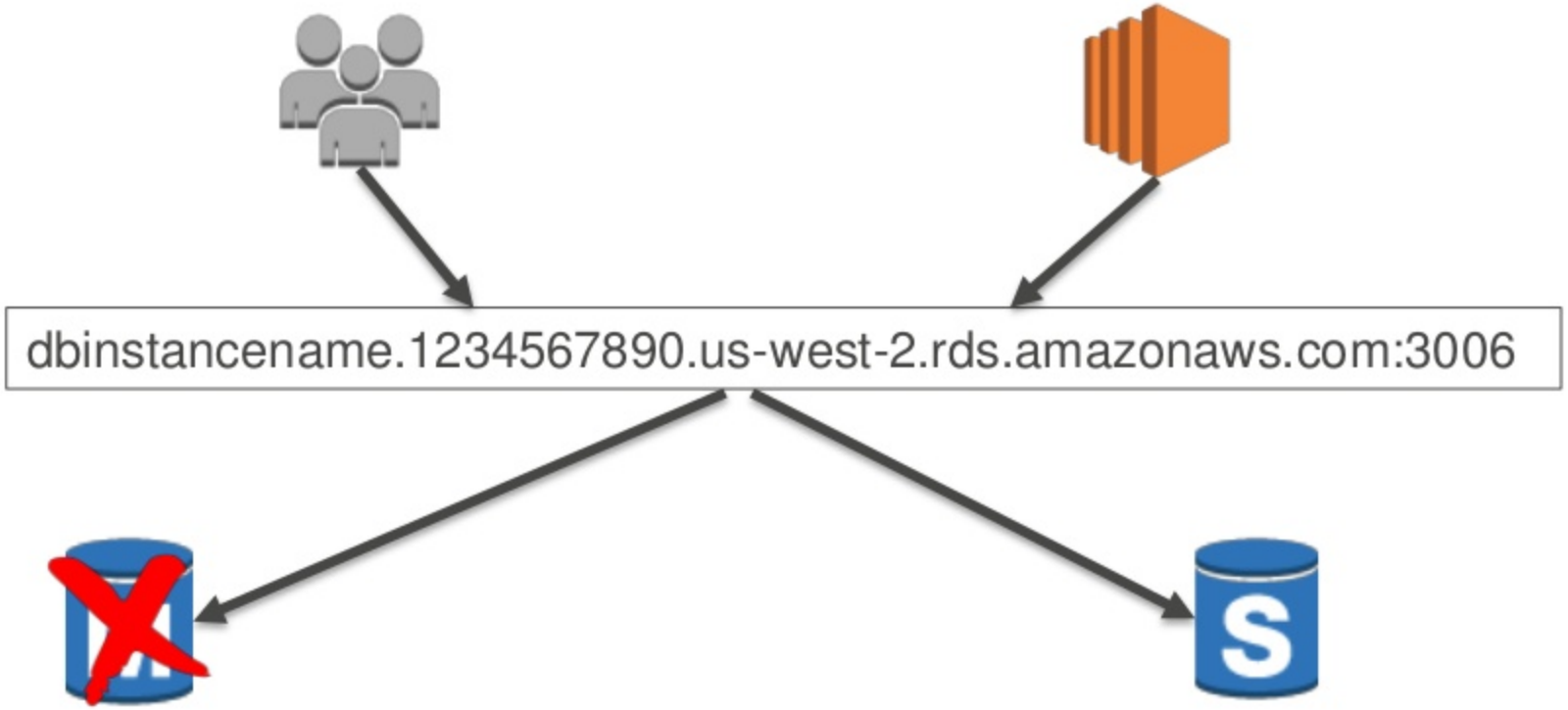
Minimal deployment—single AZ



High availability—Multi-AZ

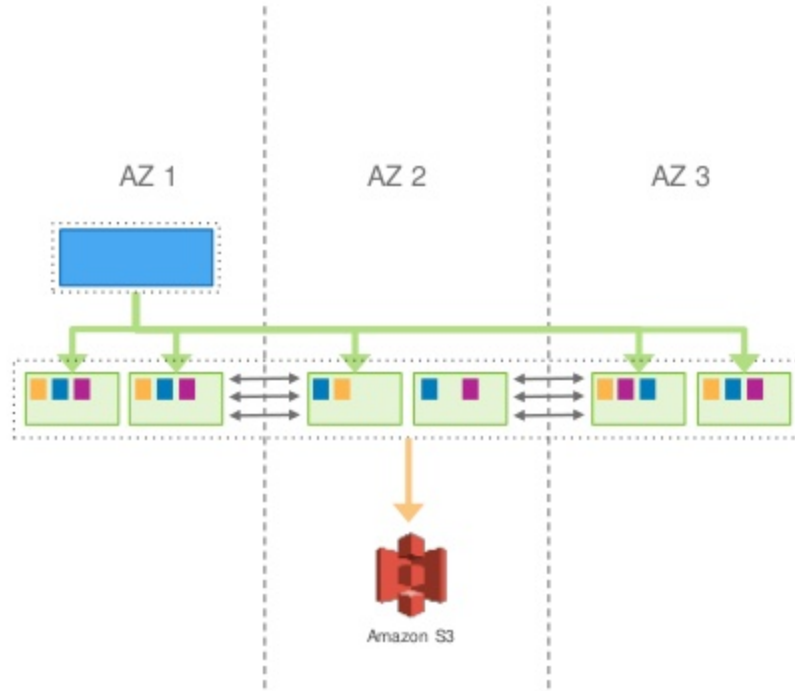


High availability—Multi-AZ to DNS



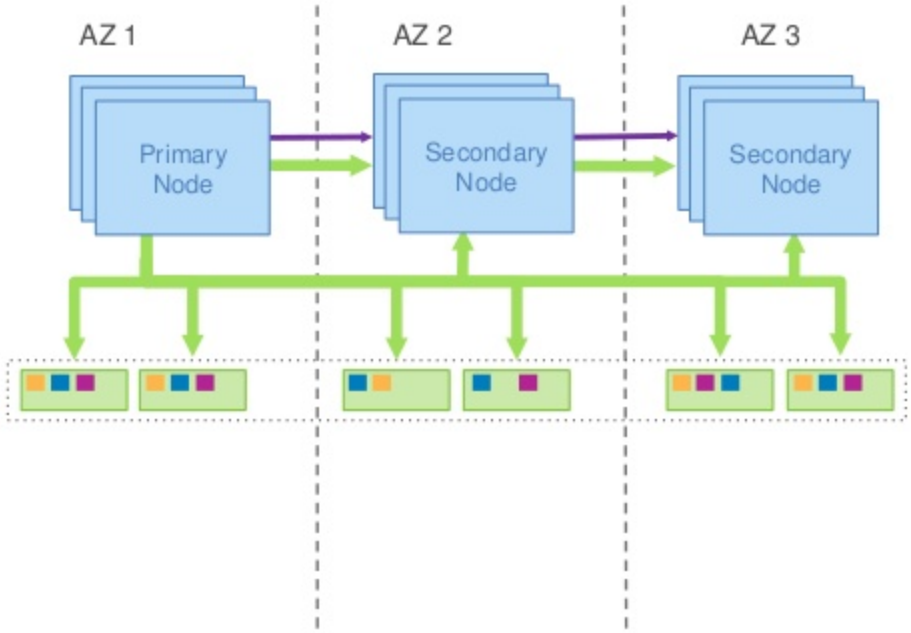
High availability—Amazon Aurora storage

- Storage volume automatically grows up to 64 TB
- Quorum system for read/write; latency tolerant
- Peer-to-peer gossip replication to fill in holes
- Continuous backup to Amazon S3 (built for 11 9s durability)
- Continuous monitoring of nodes and disks for repair
- 10 GB segments as unit of repair or hotspot rebalance
- Quorum membership changes do not stall writes

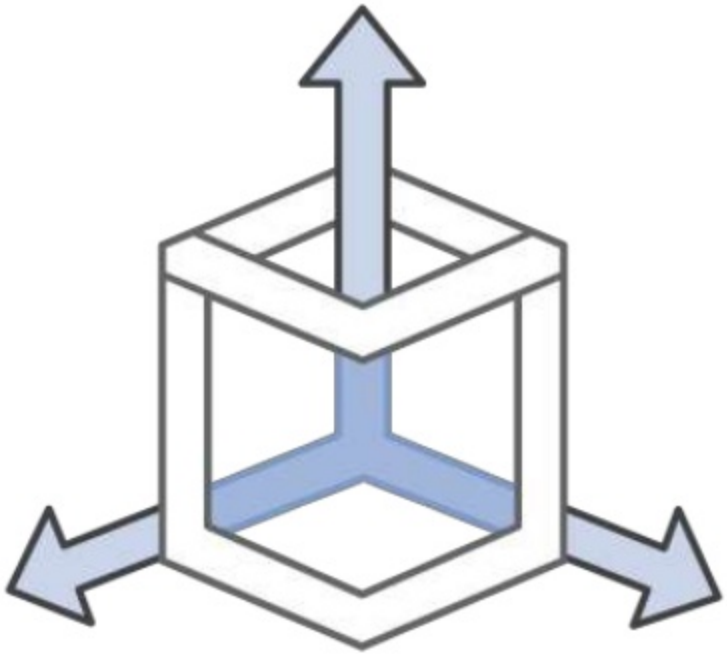


High availability—Aurora nodes

- Aurora cluster contains primary node and up to 15 secondary nodes
- Failing database nodes are automatically detected and replaced
- Failing database processes are automatically detected and recycled
- Secondary nodes automatically promoted on persistent outage, no single point of failure
- Customer application can scale out read traffic across secondary nodes



Scaling on RDS



Read Replicas

Bring data close to your customer's applications in different regions

Relieve pressure on your master node for supporting reads and writes

Promote a Read Replica to a master for faster recovery in the event of disaster

Read Replicas

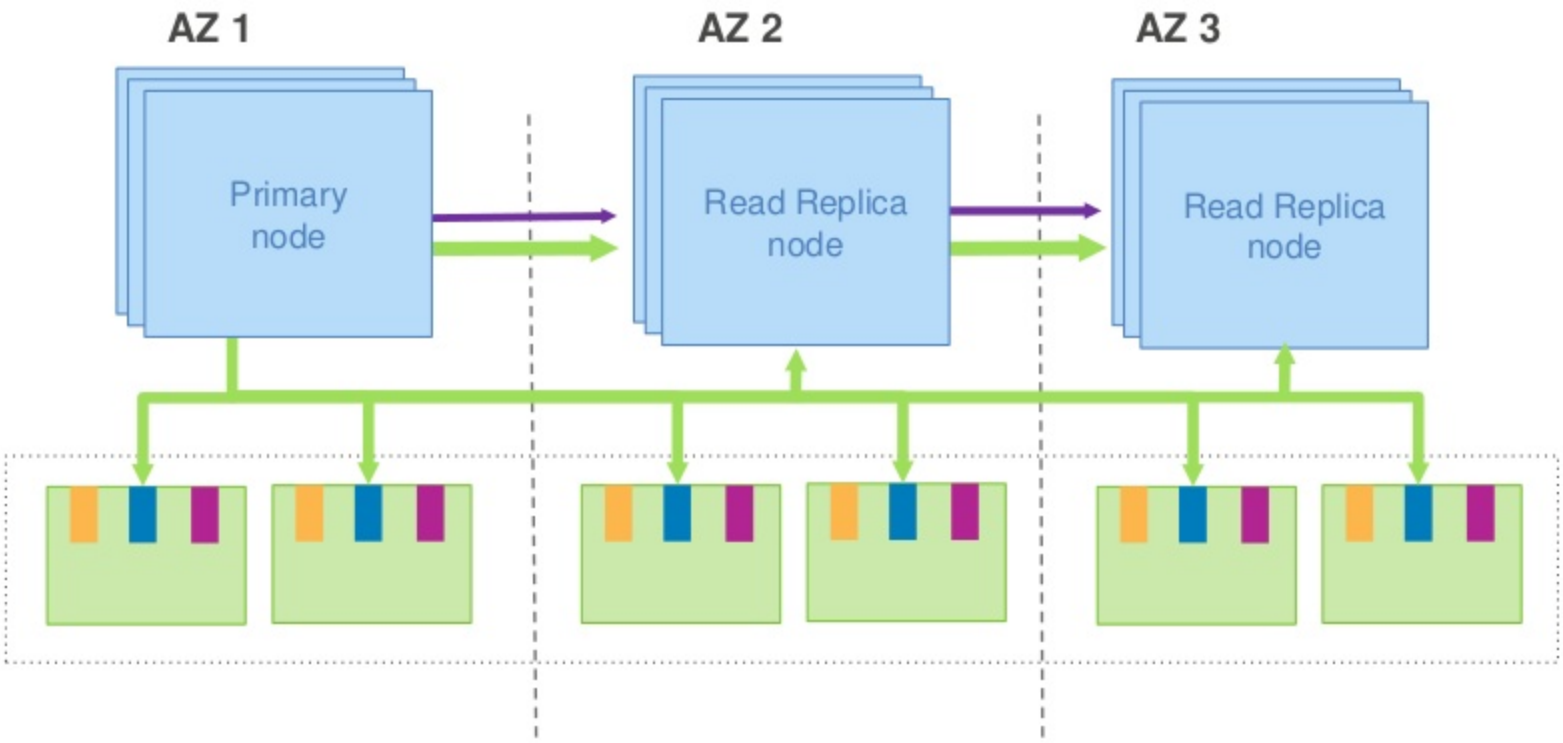
Within a region

- MySQL
- MariaDB
- PostgreSQL
- Aurora

Cross-region

- MySQL
- MariaDB
- PostgreSQL
- Aurora

Read Replicas for Amazon Aurora



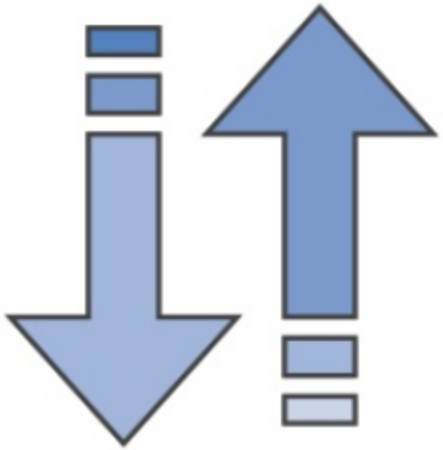
Read Replicas—Oracle and SQL Server

Options

- Oracle GoldenGate
- Third-party replication products
- Snapshots

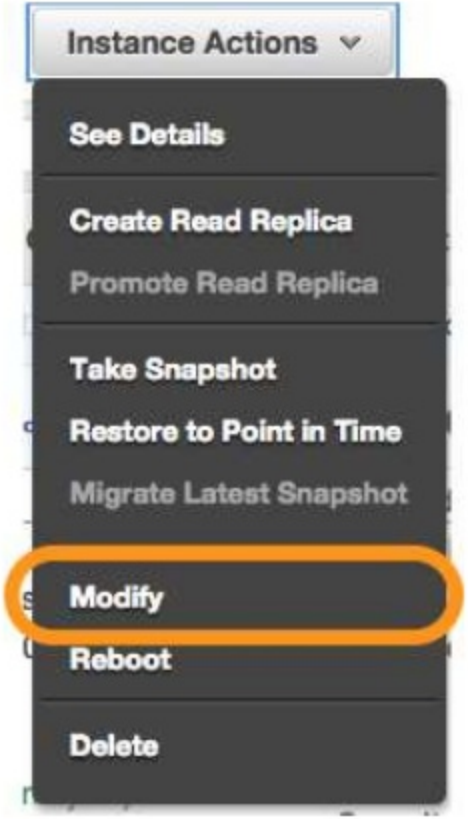
Scaling up—or down

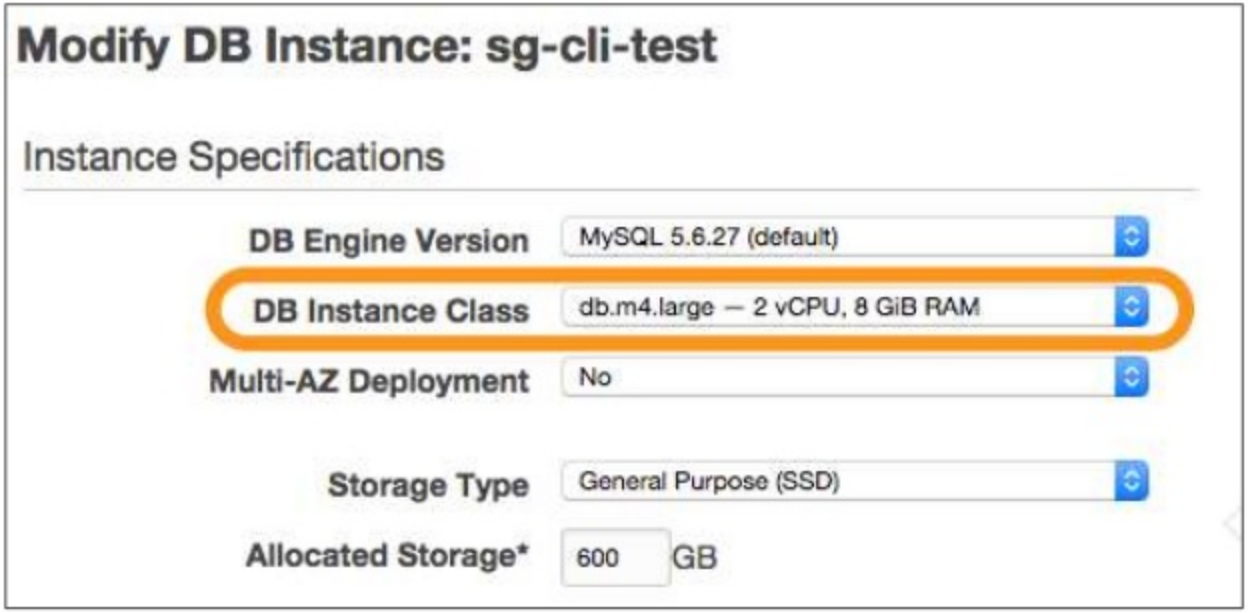
- Handle higher load or lower usage
- Control costs



Scaling up—or down

AWS Management Console





Scaling—single AZ

With single AZ deployment, the master takes an outage

Alarms and Recent Events

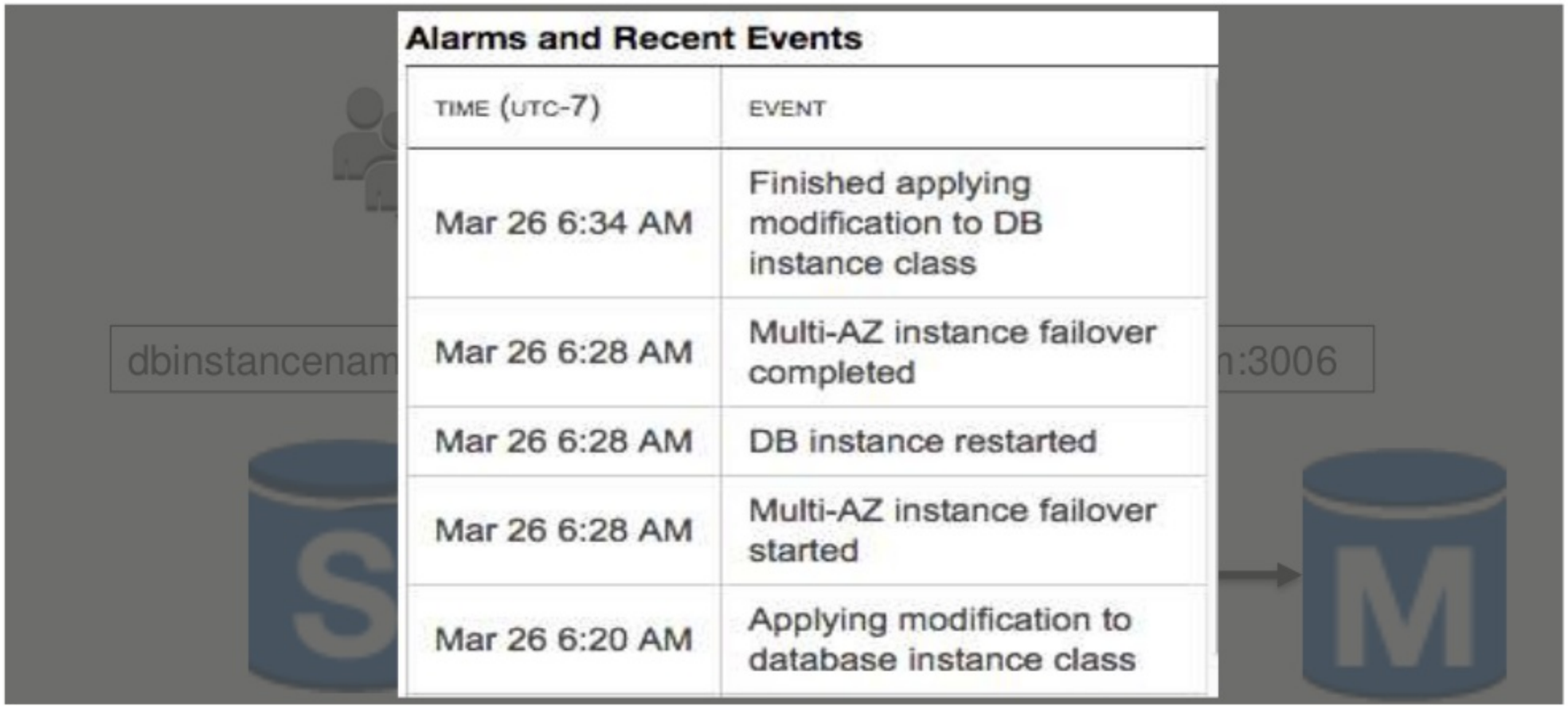
TIME (UTC-7)	EVENT
Mar 26 7:01 AM	DB instance restarted
Mar 26 7:00 AM	Finished applying modification to DB instance class
Mar 26 6:53 AM	Applying modification to database instance class

dbinstan

m:3006

Scaling—Multi-AZ

With Multi-AZ, the standby gets upgraded first



Alarms and Recent Events

TIME (UTC-7)	EVENT
Mar 26 6:34 AM	Finished applying modification to DB instance class
Mar 26 6:28 AM	Multi-AZ instance failover completed
Mar 26 6:28 AM	DB instance restarted
Mar 26 6:28 AM	Multi-AZ instance failover started
Mar 26 6:20 AM	Applying modification to database instance class

Scaling—automation

AWS CLI

```
aws rds modify-db-instance --db-instance-identifier sg-cli-test --db-instance-class
db.m4.large --apply-immediately
```

Scheduled CLI—cron

```
#Scale down at 8:00 PM on Friday
0 20 * * 5 /home/ec2-user/scripts/scale_down_rds.sh

#Scale up at 4:00 AM on Monday
0 4 * * 1 /home/ec2-user/scripts/scale_up_rds.sh
```

Scaling—automation

Scheduled—AWS Lambda

No server but still runs on a schedule!

```
import boto3

client=boto3.client('rds')

def lambda_handler(event, context):
    response=client.modify_db_instance(DBInstanceIdentifier='sg-cli-test',
                                       DBInstanceClass='db.m4.xlarge',
                                       ApplyImmediately=True)

    print response
```

Scaling—automation

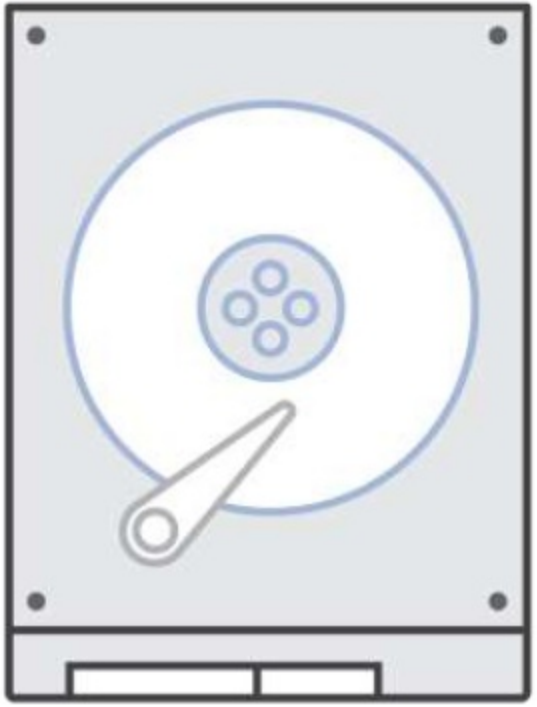
Metrics-based scaling

- Amazon CloudWatch and AWS Lambda!

Scaling—automation

```
import boto3
import json

client=boto3.client('rds')

def lambda_handler(event, context):

 message = event['Records'][0]['Sns']['Message']
 parsed_message=json.loads(message)
db_instance=parsed_message['Trigger']['Dimensions'][0]['value']
print 'DB Instance: ' + db_instance

 response=client.modify_db_instance(DBInstanceIdentifier=db_instance,
                    DBInstanceClass='db.m4.large',
                    ApplyImmediately=True)

print response
```

Backups and snapshots



Backups

MySQL, PostgreSQL, MariaDB, Oracle, SQL Server

- Scheduled daily backup of entire instance
- Archive database change logs
- 35 day retention for backups
- Multiple copies in each AZ where you have instances for a deployment

Aurora

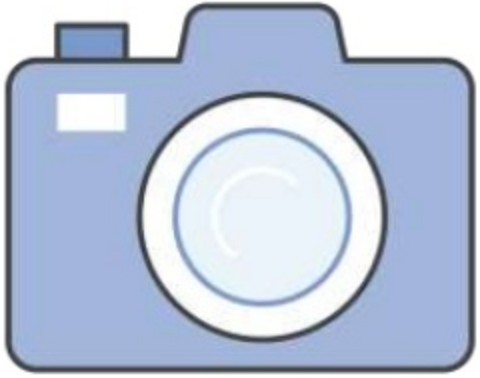
- Automatic, continuous, incremental backups
- Point-in-time restore
- No impact on database performance
- 35 day retention

Restoring

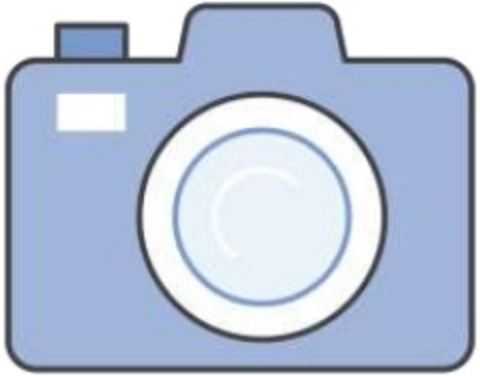
- Restoring creates an entire new database instance
- You define all the instance configuration just like a new instance

Snapshots

- Full copies of your Amazon RDS database that are different from your scheduled backups
- Backed by Amazon S3
- Used to create a new RDS instance
- Remain encrypted if using encryption



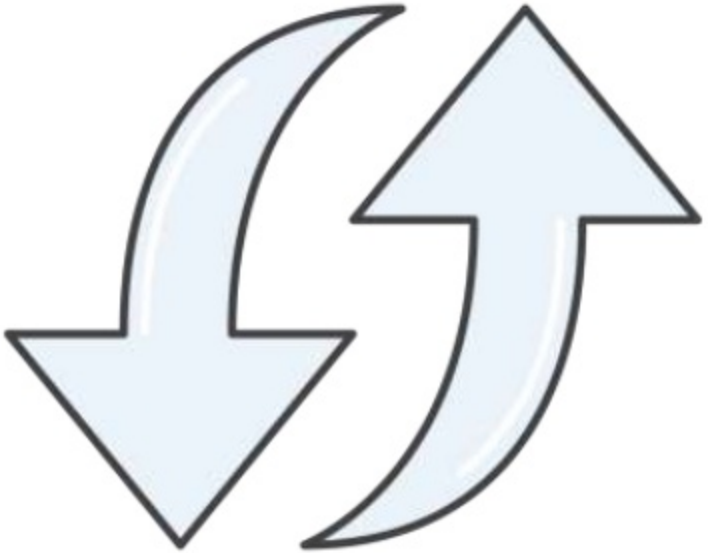
Snapshots



Use cases

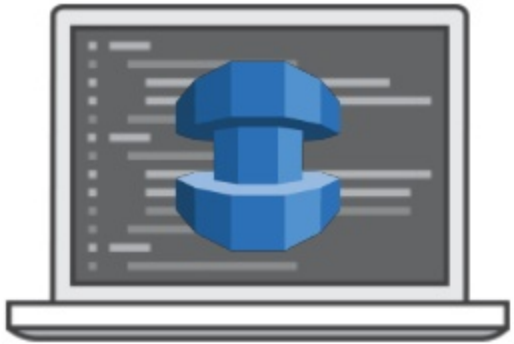
- Resolve production issues
- Nonproduction environments
- Point-in-time restore
- Final copy before terminating a database
- Disaster recovery
- Cross-region copy
- Copy between accounts

Migrating onto RDS



AWS Database Migration Service

- ✓ Move data to the same or different database engine
- ✓ Keep your apps running during the migration
- ✓ Start your first migration in 10 minutes or less
- ✓ Replicate within, to, or from Amazon EC2 or RDS



AWS Schema Conversion Tool

- ✓ Migrate from Oracle and SQL Server
- ✓ Move your tables, views, stored procedures, and data manipulation language (DML) to MySQL, MariaDB, and Aurora
- ✓ Highlight where manual edits are needed

Thank you!

amazon
web services