**Refresh Orchestrator for Amazon RDS and Amazon Aurora**

**Guidelines (v1)**

The AWS Step Functions state machine has different states, and for some them (the ones in **bold**) is necessary provide the required inputs:

PRE-RESTORE

**delete-replicas**

wait-delete-replicas

**check-status-deletereplicas**

outcome-delete-replicas

**stop-old-database**

wait-shutdown

**check-status-shutdown**

outcome-shutdown

RESTORE

**perform-restore**

wait-restore

**check-status-restore**

outcome-restore

**create-instance**

wait-create-instance

**check-status-create-instance**

outcome-create-instance

POST-RESTORE

**delete-old-database**

wait-delete

**check-status-delete**

outcome-delete

**delete-old-cluster**

wait-delete-cluster

**check-status-delete-cluster**

outcome-delete-cluster

**rename-cluster**

wait-rename-cluster

**check-status-rename-cluster**

outcome-rename-cluster

**rename-database**

wait-rename

**check-status-rename**

outcome-rename

**fix-tags**

wait-tags

**check-status-tags**

outcome-tags

**create-read-replicas**

wait-replicas

**check-status-replicas**

outcome-replicas

**create-secret**

wait-secret

**check-secret**

outcome-secret

**change-master-pwd**

wait-pwd

**check-pwd**

outcome-pwd

**rotate-master-pwd**

wait-rotate-pwd

**check-rotate-pwd**

outcome-rotate-pwd

engine-choice

**runscripts**

wait-run-scripts

**check-run-scripts**

outcome-run-scripts

**update-dynamodb**

**send-msg**

RESTORE-COMPLETE

update-dynamodb-failure

send-msg-failure

RESTORE-FAILED

For each state that requires inputs, you can find here the following elements:

* Description
* The syntax to declare the input
* The syntax to skip the step
* Notes
* Examples

**[ delete-replicas ]**

**Description**

This state deletes all the read replicas of the existing target instance

**Syntax (input)**

"deletereplicas": {

"dbservice": "aurora|rds",

"cluster": "<string>",

"dbinstance": "<string>",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"cluster": "<string>",

"dbinstance": "<string>",

"checktodo": "checkdeletereplicas",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"deletereplicas": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* **cluster** is valid only for Aurora
* **dbinstance** is valid only for RDS
* **wait\_time** is in seconds

**Examples**

**- Aurora cluster with one or more read replicas:**

"deletereplicas": {

"dbservice": "aurora",

"cluster": "auposclud",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"cluster": "auposclud",

"checktodo": "checkdeletereplicas",

"torun": "true"

},

"torun": "true"

}

**- RDS instance with one or more read replicas:**

"deletereplicas": {

"dbservice": "rds",

"dbinstance": "",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "",

"checktodo": "checkdeletereplicas",

"torun": "true"

},

"torun": "true"

}

**[ stop-old-database ]**

**Description**

This state stops the existing RDS instance

**Syntax (input)**

"stopdb": {

"dbservice": "rds",

"dbinstance": "<string>",

"wait\_time": <integer>,

"check": {

"dbservice": "rds",

"dbinstance": "<string>",

"checktodo": "checkstopdb",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"stopdb": {

"wait\_time": "integer",

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* Aurora doesn’t support this state, it must be skipped
* This step must be skipped if the database you want to restore doesn’t exist yet
* **wait\_time** is in seconds

**Examples**

**- RDS MySQL instance:**

"stopdb": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"checktodo": "checkstopdb",

"torun": "true"

},

"torun": "true"

}

**[ perform-restore ]**

**Description**

This state performs the restore of a new RDS instance or of a new Aurora cluster

**Syntax (input)**

"restore": {

"dbservice": "aurora|rds",

"source": "<string>",

"sourcecluster": "<string>",

"target": "<string>",

"targetcluster": "<string>",

"restoretype": "fromsnapshot|restorepoint|latestpoint|fastcloning",

"engine": "aurora|aurora-mysql|aurora-postgresql",

"engineversion": "<string>",

"application": "<string>",

"restoretime": "%Y-%m-%d %H:%M:%S",

"snapshot": "<string>",

"backtrack": <integer>,

"environment": "<string>",

"port": <integer>,

"subgrp": "<string>",

"iamdbauth": "True|False",

"cwalogs": "<logname1>,< logname2>,<lognameN>",

"copytagstosnap": "True|False",

"dbparamgrp": "<string>",

"cluparamgrp": "<string>",

"deletionprotection": "True|False",

"secgrpids": "<string>",

"multiaz": "True|False",

"dbname": "<string>",

"dbclass": "<string>",

"autominor": "True|False",

"storagetype": "gp2|io1",

"iops": <integer>,

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"dbinstance": "<string>**temp**",

"checktodo": "checkrestore",

"torun": "true"

},

"torun": "true"

}

**Syntax (skip)**

"restore": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* The following parameters are only valid for Aurora: **sourcecluster**, **targetcluster**, **engine**, **engineversion**, and **cluparamgrp**
* The following parameters are only valid for RDS: **source**, **target**, **dbparamgrp**, **multiaz**, **dbname**, **dbclass**, **autominor** and **storagetype**
* **engine** must be specified only for Aurora and if **restoretype** is set to "fromsnapshot"
* Here the meaning of the possible values for **engine**:

MySQL 5.6-compatible Aurora => aurora

MySQL 5.7-compatible Aurora => aurora-mysql

PostgreSQL-compatible Aurora => aurora-postgresql

* if **engine** is set to "aurora" or "aurora-mysql", **backtrack** is required
* **engineversion** must be specified only for Aurora and if **restoretype** is set to "fromsnapshot"
* **engineversion** must have a valid value, use the following commands to list them:

[if engine = aurora]

aws rds describe-db-engine-versions \

--engine aurora \

--query "DBEngineVersions[].EngineVersion"

[if engine = aurora-mysql]

aws rds describe-db-engine-versions \

--engine aurora-mysql \

--query "DBEngineVersions[].EngineVersion"

[if engine = aurora-postgresql]

aws rds describe-db-engine-versions \

--engine aurora-postgresql \

--query "DBEngineVersions[].EngineVersion"

* **dbname** is not required and currently is ignored by the code
* **cwalogs** must contain only the list of the supported logs:

[if engine = oracle]

"trace,audit,alert,listener"

[if engine = mysql]

"audit,error,general,slowquery"

[if engine = mariadb]

"audit,error,general,slowquery"

[if engine = postgresql]

"postgresql,upgrade"

[if engine = aurora-mysql]

"audit,error,general,slowquery"

[if engine = aurora-postgresql]

"postgresql,upgrade"

* **cluparamgrp** must refer to an existing Cluster Parameter Group and it should be consistent with the engine specified with **engine**
* **dbparamgrp** must refer to an existing DB Parameter Group and it should be consistent with the database engine and version involved
* if **restoretype** is set to "restorepoint", **restoretime** is required
* if **restoretype** is set to "fromsnapshot", **snapshot** is required
* **restoretime** must be specified in the following format: %Y-%m-%d %H:%M:%S, if doesn’t fall within the restore window of the source database the operation will fail
* **snapshot** is the snapshot ARN, if the snapshot specified doesn’t exist or is not accessible the operation will fail:

Example: arn:aws:rds:us-east-1:12345678901:snapshot:rds:orgammap-2020-05-26-10-09

* **dbclass** must have a valid value for an RDS instance, see the following link:

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html#Concepts.DBInstanceClass.Types>

* **iamdbauth** can be "true" only for the engines which support the feature
* If **storagetype** is set to "io1", **iops** must be set as well
* **iops** must be compatible with the allocated storage for the instance. See the following link:

Section **Provisioned IOPS SSD Storage**

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html#USER_PIOPS>

* In the "check" section **dbinstance** needs the "temp" string at the end of it
* **wait\_time** is in seconds

**Examples**

**- Aurora MySQL cluster restored with fast-cloning:**

"restore": {

"dbservice": "aurora",

"sourcecluster": "aumyclup",

"targetcluster": "aumyclud",

"restoretype": "fastcloning",

"engine": "aurora",

"application": "app5",

"environment": "development",

"backtrack": 0,

"port": 3308,

"subgrp": "default",

"iamdbauth": "False",

"cwalogs": "audit,error,general,slowquery",

"copytagstosnap": "True",

"cluparamgrp": "default.aurora5.6",

"deletionprotection": "False",

"secgrpids": "sg-0d2ed6510c66ee394",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"cluster": "aumycludtemp",

"checktodo": "checkrestore",

"torun": "true"

},

"torun": "true"

}

**- RDS MySQL instance restored to the latestpoint:**

"restore": {

"dbservice": "rds",

"source": "mysqlinstp",

"target": "mysqlinstd",

"restoretype": "latestpoint",

"application": "app1",

"environment": "development",

"port": 3306,

"subgrp": "db-sub-grp-virginia",

"iamdbauth": "False",

"cwalogs": "error,general,slowquery",

"copytagstosnap": "True",

"dbparamgrp": "default.mysql5.7",

"deletionprotection": "False",

"secgrpids": "sg-09ea3b351f448b6b2",

"multiaz": "False",

"dbclass": "db.t3.micro",

"autominor": "False",

"storagetype": "gp2",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mysqlinstdtemp",

"checktodo": "checkrestore",

"torun": "true"

},

"torun": "true"

}

**- RDS Oracle instance restored from snapshot:**

"restore": {

"dbservice": "rds",

"source": "orgammap",

"target": "orainstd",

"restoretype": "fromsnapshot",

"snapshot": "arn:aws:rds:us-east-1:123456789012:snapshot:rds:orgammap-2020-05-26-10-09",

"application": "app3",

"environment": "development",

"port": 1522,

"subgrp": "default",

"iamdbauth": "False",

"cwalogs": "trace,audit,alert,listener",

"copytagstosnap": "True",

"dbparamgrp": "default.oracle-ee-12.2",

"deletionprotection": "False",

"secgrpids": "sg-0d2ed6510c66ee394",

"multiaz": "False",

"dbclass": "db.t3.medium",

"autominor": "False",

"storagetype": "gp2",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "orainstdtemp",

"checktodo": "checkrestore",

"torun": "true"

},

"torun": "true"

}

**- RDS MariaDB instance restored to restore point with provisioned IOPS storage:**

"restore": {

"dbservice": "rds",

"source": "mariainstp",

"target": "mariainstd",

"restoretype": "restorepoint",

"restoretime": "2020-05-26 18:24:00",

"application": "app4",

"environment": "development",

"port": 3308,

"subgrp": "default",

"iamdbauth": "False",

"cwalogs": "audit,error,general,slowquery",

"copytagstosnap": "True",

"dbparamgrp": "default.mariadb10.2",

"deletionprotection": "False",

"secgrpids": "sg-0d2ed6510c66ee394",

"multiaz": "False",

"dbname": "mysqldbd",

"dbclass": "db.t3.medium",

"autominor": "False",

"storagetype": "io1",

"iops": 1000,

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mariainstdtemp",

"checktodo": "checkrestore",

"torun": "true"

},

"torun": "true"

}

**[ create-instance ]**

**Description**

This state creates a new Aurora instance within the restored Aurora cluster

**Syntax (input)**

"createinstance": {

"dbservice": "aurora",

"cluster": "<string>**temp**",

"dbinstance": "<string>",

"application": "<string>",

"environment": "<string>",

"dbclass": "<string>",

"engine": "aurora-postgresql",

"subgrp": "<string>",

"dbparamgrp": "<string>",

"autominor": "True|False",

"copytagstosnap": "True|False",

"perfinsights": "True|False",

"perfinsightsretention": 7|731,

"perfinsightskmskeyid": "<string>",

"enhancedmon": "True|False",

"enhancedmoninterval": 0|1|5|10|15|30|60,

"enhancedmonrolearn": "<string>",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora",

"dbinstance": "<string>**temp**",

"checktodo": "checkcreate",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"createinstance": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* RDS doesn’t support this state, it must be skipped
* **cluster** needs the "temp" string at the end of it
* **dbinstance** needs the "temp" string at the end of it (but only in the "check" section)
* **dbclass** must have a valid value for an Aurora instance, see the following link:

<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Concepts.DBInstanceClass.html#Concepts.DBInstanceClass.Types>

* **engine** must have a valid value, see the following list:

aurora (for MySQL 5.6-compatible Aurora)

aurora-mysql (for MySQL 5.7-compatible Aurora)

aurora-postgresql

mariadb

mysql

oracle-ee

oracle-se2

oracle-se1

oracle-se

postgres

sqlserver-ee

sqlserver-se

sqlserver-ex

sqlserver-web

* **dbparamgrp** must refer to an existing DB Parameter Group and it should be consistent with the engine specified with **engine**
* **enhancedmoninterval** and **enhancedmonrolearn** must be set only if **enhancedmon** is set to "True"
* For **enhancedmonrolearn**, if enhanced monitoring was activated at least one time, you should have the default role "arn:aws:iam::<account\_number>:role/rds-monitoring-role"
* **perfinsightsretention** and **perfinsightskmskeyid** must be set only if **perfinsights** is set to "True"
* For **perfinsightskmskeyid**, if you use the default AWS managed key you can specify the Key ID for the key "aws/rds" (check in KMS)
* **wait\_time** is in seconds

**Examples**

**- Aurora PostgreSQL instance with enhanced monitoring enabled:**

"createinstance": {

"dbservice": "aurora",

"cluster": "auposcludtemp",

"dbinstance": "auposinstd",

"application": "app2",

"environment": "development",

"dbclass": "db.t3.medium",

"engine": "aurora-postgresql",

"subgrp": "default",

"dbparamgrp": "default.aurora-postgresql10",

"autominor": "False",

"copytagstosnap": "True",

"enhancedmon": "True",

"enhancedmoninterval": 60,

"enhancedmonrolearn": "arn:aws:iam::123456789012:role/rds-monitoring-role",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"dbinstance": "auposinstdtemp",

"checktodo": "checkcreate",

"torun": "true"

},

"torun": "true"

}

**- Aurora MySQL instance with Performance Insights enabled:**

"createinstance": {

"dbservice": "aurora",

"cluster": "aumycludtemp",

"dbinstance": "aumyinstd",

"application": "app5",

"environment": "development",

"dbclass": "db.t3.medium",

"engine": "aurora",

"subgrp": "default",

"dbparamgrp": "default.aurora5.6",

"autominor": "False",

"copytagstosnap": "True",

"enhancedmon": "False",

"perfinsights": "True",

"perfinsightsretention": 7,

"perfinsightskmskeyid": "fc59b5b9-193f-41c5-a196-fc0000000000",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"dbinstance": "aumyinstdtemp",

"checktodo": "checkcreate",

"torun": "true"

},

"torun": "true"

}

**[ delete-old-database ]**

**Description**

This state deletes the existing target instance

**Syntax (input)**

"delete": {

"dbservice": "aurora|rds",

"dbinstance": "<string>",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"dbinstance": "<string>",

"checktodo": "checkdelete",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"delete": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* This step must be skipped if the database you want to restore doesn’t exist yet
* **wait\_time** is in seconds

**Examples**

**- RDS MySQL instance:**

"delete": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"checktodo": "checkdelete",

"torun": "true"

},

"torun": "true"

}

**- Aurora instance:**

"delete": {

"dbservice": "aurora",

"dbinstance": "auposinstd",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"dbinstance": "auposinstd",

"checktodo": "checkdelete",

"torun": "true"

},

"torun": "true"

}

**[ delete-old-cluster ]**

**Description**

This state deletes the existing target Aurora cluster

**Syntax (input)**

"deletecluster": {

"dbservice": "aurora",

"cluster": "<string>",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora",

"cluster": "<string>",

"checktodo": "checkdeletecluster",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"deletecluster": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* RDS doesn’t support this state, it must be skipped
* **wait\_time** is in seconds

**Examples**

**- Aurora instance:**

"deletecluster": {

"dbservice": "aurora",

"cluster": "auposclud",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"cluster": "auposclud",

"checktodo": "checkdeletecluster",

"torun": "true"

},

"torun": "true"

}

**[ rename-cluster ]**

**Description**

This state renames the new restored Aurora cluster. The temporary name (which ends with the string "temp") is changed with the name that the cluster originally had

**Syntax (input)**

"renamecluster": {

"dbservice": "aurora ",

"cluster": "<string>**temp**",

"wait\_time": "integer",

"check": {

"dbservice": "aurora ",

"cluster": "<string>",

"checktodo": "checkrenamecluster",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"renamecluster": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* RDS doesn’t support this state, it must be skipped
* **cluster** needs the "temp" string at the end of it (but not in the "check" section)
* **wait\_time** is in seconds

**Examples**

**- Aurora cluster:**

"renamecluster": {

"dbservice": "aurora",

"cluster": "auposclud**temp**",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"cluster": "auposclud",

"checktodo": "checkrenamecluster",

"torun": "true"

},

"torun": "true"

}

**[ rename-database ]**

**Description**

This state renames the new restored instance. The temporary name (which ends with the "temp" string) is changed with the name that the instance originally had

**Syntax (input)**

"rename": {

"dbservice": "aurora|rds",

"dbinstance": "<string>**temp**",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"dbinstance": "<string>",

"checktodo": "checkrename",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

" rename ": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* **dbinstance** needs the "temp" string at the end of it (but not in the "check" section)
* **wait\_time** is in seconds

**Examples**

**- RDS MySQL instance:**

"rename": {

"dbservice": "rds",

"dbinstance": "mysqlinstd**temp**",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"checktodo": "checkrename",

"torun": "true"

},

"torun": "true"

}

**- Aurora instance:**

"rename": {

"dbservice": "aurora",

"dbinstance": "auposinstd**temp**",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"dbinstance": "auposinstd",

"checktodo": "checkrename",

"torun": "true"

},

"torun": "true"

}

**[ fix-tags ]**

**Description**

This state sets the right tags to the new restored instance and, with Aurora, to the new restored cluster

**Syntax (input)**

"fixtags": {

"dbservice": "aurora|rds",

"cluster": "<string>",

"cluarn": "<string>",

"dbinstance": "<string>",

"dbarn": "<string>",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"cluster": "<string>",

"cluarn": "<string>",

"dbinstance": "<string>",

"dbarn": "<string>",

"checktodo": "checkfixtags",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"fixtags": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* **wait\_time** is in seconds
* **cluster** and **cluarn** are only valid for Aurora
* **dbinstance** and **dbarn** are only valid for RDS

**Examples**

**- Aurora cluster:**

"fixtags": {

"dbservice": "aurora",

"cluster": "audeltaclud",

"cluarn": "arn:aws:rds:us-east-1:123456789012:cluster:audeltaclud",

"dbinstance": "audeltainstd",

"dbarn": "arn:aws:rds:us-east-1:123456789012:db:audeltainstd",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"cluster": "audeltaclud",

"cluarn": "arn:aws:rds:us-east-1:123456789012:cluster:audeltaclud",

"dbinstance": "audeltainstd",

"dbarn": "arn:aws:rds:us-east-1:123456789012:db:audeltainstd",

"checktodo": "checkfixtags",

"torun": "true"

},

"torun": "true"

}

**- RDS instance:**

"fixtags": {

"dbservice": "rds",

"dbinstance": "mybetainstd",

"dbarn": "arn:aws:rds:us-east-1:123456789012:db:mybetainstd",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mybetainstd",

"dbarn": "arn:aws:rds:us-east-1:123456789012:db:mybetainstd",

"checktodo": "checkfixtags",

"torun": "true"

},

"torun": "true"

}

**[ create-read-replicas ]**

**Description**

This state creates the read replicas to the new restored target instance

**Syntax (input)**

"createreplicas": {

"dbservice": "aurora|rds",

"cluster": "<string>",

"engine": "aurora-postgresql",

"subgrp": "<string>",

"dbparamgrp": "<string>",

"multiaz": "True|False",

"iamdbauth": "True|False",

"secgrpids": "<string>",

"storagetype": "gp2|io1",

"iops": "<integer>",

"dbinstance": "<string>",

"application": "<string>",

"environment": "<string>",

"autominor": "True|False",

"copytagstosnap": "True|False",

"replicas": "<replica1-name>**\_**<replica1-db-instance-class**\_**replica1-port>,<replicaN-name>**\_**<replicaN-db-instance-class**\_**replicaN-port>",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"replicas": "<replica1-name>,<replicaN-name>",

"checktodo": "checkreplicas",

"torun": "true"

},

"torun": "true"

}

**Syntax (skip)**

"createreplicas": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* The following parameters are only valid for Aurora: **cluster**, **engine**, **subgrp** and **dbparamgrp**
* The following parameters are only valid for RDS: **multiaz**, **iamdbauth**, **secgrpids** and **storagetype**
* **engine** must have a valid value, see the following list:

aurora (for MySQL 5.6-compatible Aurora)

aurora-mysql (for MySQL 5.7-compatible Aurora)

aurora-postgresql

mariadb

mysql

oracle-ee

oracle-se2

oracle-se1

oracle-se

postgres

sqlserver-ee

sqlserver-se

sqlserver-ex

sqlserver-web

* **dbparamgrp** must refer to an existing DB Parameter Group and it should be consistent with the engine specified with **engine**
* **dbclass** must have a valid value for an Aurora instance, see the following link:

<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Concepts.DBInstanceClass.html#Concepts.DBInstanceClass.Types>

* **replicas** must have replica names separated by a "," and, for each replica, information must be separated by a "\_" (in the "check" section **replicas** just needs the replica names separated by a ",")
* If **storagetype** is set to "io1", **iops** must be set as well
* **storagetype** must be compatible with the storage type of the primary instance created with the "perform-restore" state
* **iops** must be compatible with the allocated storage for the primary instance. See the following link:

Section **Provisioned IOPS SSD Storage**

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html#USER_PIOPS>

* **wait\_time** is in seconds

**Examples**

**- Aurora cluster with one read replica:**

"createreplicas": {

"dbservice": "aurora",

"cluster": "auposclud",

"engine": "aurora-postgresql",

"application": "app2",

"environment": "development",

"dbinstance": "auposinstd",

"autominor": "False",

"copytagstosnap": "True",

"subgrp": "default",

"dbparamgrp": "default.aurora-postgresql10",

"replicas": "auposinstd-b\_db.t3.medium",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"replicas": "auposinstd-b",

"checktodo": "checkreplicas",

"torun": "true"

},

"torun": "true"

}

**- RDS Oracle instance with one read replica with general purpose storage:**

"createreplicas": {

"dbservice": "rds",

"application": "app3",

"environment": "development",

"dbinstance": "orainstd",

"multiaz": "False",

"iamdbauth": "False",

"autominor": "False",

"storagetype": "gp2",

"secgrpids": "sg-0d2ed6510c66ee394",

"copytagstosnap": "True",

"replicas": "orainstd-b\_db.t3.medium\_1522",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"replicas": "orainstd-b",

"checktodo": "checkreplicas",

"torun": "true"

},

"torun": "true"

}

**- RDS Oracle with two read replicas with provisioned IOPS storage:**

"createreplicas": {

"dbservice": "rds",

"application": "app4",

"environment": "development",

"dbinstance": "mariainstd",

"multiaz": "False",

"iamdbauth": "False",

"autominor": "False",

"storagetype": "io1",

"iops": 1000,

"secgrpids": "sg-0d2ed6510c66ee394",

"copytagstosnap": "True",

"replicas": "mariainstd-b\_db.t3.medium\_3308,mariainstd-c\_db.t3.medium\_3308",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"replicas": "mariainstd-b,mariainstd-c",

"checktodo": "checkreplicas",

"torun": "true"

},

"torun": "true"

}

**[ create-secret ]**

**Description**

This state creates a new AWS Secrets Manager secret for the new restored RDS instance or Aurora cluster

**Syntax (input)**

"createsecret": {

"dbservice": "aurora|rds",

"dbinstance": "<string>",

"cluster": "<string>",

"temppwd": "<string>",

"secretname": "<string>",

"rotation": "true|false",

"lambdaarn": "<string>",

"rotationdays": <integer>,

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"secretname": "<string>",

"rotation": "true|false",

"checktodo": "checkcreatesecret",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"createsecret": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* **cluster** is only valid for Aurora
* **dbinstance** is only valid for RDS
* **temppwd** is a temporary password for the new restored RDS instance or Aurora cluster
* If **rotation** is set to "true", **lambdaarn** is required and should point to an existing Lambda function used to rotate the secret for the specific engine involved
* **wait\_time** is in seconds

**Examples**

**- Aurora cluster without secret:**

TBD

**- RDS MySQL instance:**

"createsecret": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"temppwd": "temppwd123",

"secretname": "/development/app1/mysqlinstd",

"rotation": "true",

"lambdaarn": "arn:aws:lambda:us-east-1:123456789012:function:awssoldb-secretmanager-rotation-MySQL",

"rotationdays": 1,

"wait\_time": 10,

"check": {

"dbservice": "rds",

"secretname": "/development/app1/mysqlinstd",

"rotation": "true",

"checktodo": "checkcreatesecret",

"torun": "true"

},

"torun": "true"

}

**[ change-master-pwd ]**

**Description**

This state changes the master password of the new restored instance and, with Aurora, to the new restored cluster

**Syntax (input)**

"changemasterpwd": {

"dbservice": "aurora|rds",

"cluster": "<string>",

"dbinstance": "<string>",

"temppwd": "<string>",

"secret": "true|false",

"secretname": "<string>",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"cluster": "<string>",

"dbinstance": "<string>",

"checktodo": "checkpwd",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"changemasterpwd": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* **cluster** is only valid for Aurora
* **dbinstance** is only valid for RDS
* **temppwd** is a temporary password for the new restored database and can be changed later
* If **secret** is set to "true", then **secretname** is required
* If **secret** is set to "true", once the master password is changed the related Secrets Manager secret is automatically changed too
* **wait\_time** is in seconds

**Examples**

**- Aurora cluster without secret:**

"changemasterpwd": {

"dbservice": "aurora",

"cluster": "auetaclud",

"temppwd": "temppwd123",

"secret": "false",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"cluster": "auetaclud",

"checktodo": "checkpwd",

"torun": "true"

},

"torun": "true"

}

**- RDS instance without secret:**

"changemasterpwd": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"temppwd": "temppwd123",

"secret": "false",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"checktodo": "checkpwd",

"torun": "true"

},

"torun": "true"

}

**- RDS instance with secret:**

"changemasterpwd": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"temppwd": "temppwd123",

"secret": "true",

"secretname": "/development/app1/mysqlinstd",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"checktodo": "checkpwd",

"torun": "true"

},

"torun": "true"

}

**[ rotate-master-pwd ]**

**Description**

This state rotates the Secrets Manager secret related to the master password

**Syntax (input)**

"rotatemasterpwd": {

"dbservice": "aurora|rds",

"cluster": "<string>",

"dbinstance": "<string>",

"secretname": "<string>",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"secretname": "<string>",

"temppwd": "<string>",

"checktodo": "rotatepwd",

"torun": "true|false"

},

"torun": "true|false"

}

**Syntax (skip)**

"rotatemasterpwd": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* **cluster** is only valid for Aurora
* **dbinstance** is only valid for RDS
* **temppwd** must have the same value of **temppwd** specified in the state "change-master-pwd"
* **wait\_time** is in seconds

**Examples**

**- Aurora cluster:**

"rotatemasterpwd": {

"dbservice": "aurora",

"cluster": "auposclud",

"secretname": "/development/gamma/auposclud",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"secretname": "/development/gamma/auposclud",

"temppwd": "temppwd123",

"checktodo": "rotatepwd",

"torun": "true"

},

"torun": "true"

}

**- RDS instance:**

"rotatemasterpwd": {

"dbservice": "rds",

"dbinstance": "mybetainstd",

"secretname": "/development/gamma/mybetainstd",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"secretname": "/development/gamma/mybetainstd",

"temppwd": "temppwd123",

"checktodo": "rotatepwd",

"torun": "true"

},

"torun": "true"

}

**[ run-scripts ]**

**Description**

This state runs the .sql scripts specified against the database instance

**Syntax (input)**

"runscripts": {

"dbservice": "aurora|rds",

"cluster": "<string>",

"dbinstance": "<string>",

"engine": "aurora-mysql|mysql|mariadb|oracle|aurora-postgresql|postgresql",

"access": "pwd|secret",

"temppwd": "<string>",

"secretname": "<string>",

"method": "lambda|ec2",

"bucketname": "<string>",

"prefix": "<string>/<string>",

"keys": "<filename1>,<filename2>,<filenameN>",

"wait\_time": <integer>,

"check": {

"dbservice": "aurora|rds",

"bucketname": "<string>",

"prefix": "<string>/<string>",

"checktodo": "runscripts",

"torun": "true"

},

"torun": "true"

}

**Syntax (skip)**

"runscripts": {

"dbservice": "aurora|rds",

"**engine**": "**aurora-mysql|aurora-postgresql|mysql|postgresql|mariadb|oracle|sqlserver**",

"wait\_time": 3,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* **cluster** is only valid for Aurora
* **dbinstance** is only valid for RDS
* If **access** is set to "pwd", then **temppwd** is required and it must have the same value of **temppwd** specified in the states "change-master-pwd" and "rotate-master-pwd"
* If **access** is set to "secret", then **secretname** is required
* **secretname** is the name of the Secrets Manager secret related to the master password of the instance
* Currently the supported values for **engine** are "aurora-mysql", "mysql" and "mariadb" (the others are simply ignored)
* A possible value for **prefix** could be "s3\_directory/dbinstance" or "s3\_directory/cluster"
* When you skip this state, you still need to specify a value for **engine**
* **wait\_time** is in seconds

**Examples**

**- Aurora MySQL cluster with Lambda to run the .sql scripts (master password provided by the user):**

"runscripts": {

"dbservice": "aurora",

"cluster": "aumyclud",

"engine": "aurora-mysql",

"access": "pwd",

"temppwd": "temppwd123",

"method": "lambda",

"bucketname": "awsolproj",

"prefix": "auroramysql/aumyclud",

"keys": "00test.sql,01test.sql",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"bucketname": "awsolproj",

"prefix": "auroramysql/aumyclud",

"checktodo": "runscripts",

"torun": "true"

},

"torun": "true"

}

**- Aurora MySQL cluster with Ec2 and Systems Manager to run the .sql scripts (master password provided by the user):**

"runscripts": {

"dbservice": "aurora",

"cluster": "aumyclud",

"engine": "aurora-mysql",

"access": "pwd",

"temppwd": "temppwd123",

"method": "ec2",

"bucketname": "awsolproj",

"prefix": "auroramysql/aumyclud",

"keys": "00testEc2.sql,01testEc2.sql",

"wait\_time": 15,

"check": {

"dbservice": "aurora",

"bucketname": "awsolproj",

"prefix": "auroramysql/aumyclud",

"checktodo": "runscripts",

"torun": "true"

},

"torun": "true"

}

**- RDS MySQL instance with Lambda to run the .sql scripts (master password provided by the user):**

"runscripts": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"engine": "mysql",

"access": "pwd",

"temppwd": "temppwd123",

"method": "lambda",

"bucketname": "awsolproj",

"prefix": "rdsmysql/mysqlinstd",

"keys": "00test.sql,01test.sql",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"bucketname": "awsolproj",

"prefix": "rdsmysql/mysqlinstd",

"checktodo": "runscripts",

"torun": "true"

},

"torun": "true"

}

**- RDS MySQL instance with Lambda to run the .sql scripts (master password stored in a secret):**

"runscripts": {

"dbservice": "rds",

"dbinstance": "mybetainstd",

"engine": "mysql",

"access": "secret",

"secretname": "/development/gamma/mybetainstd",

"method": "lambda",

"bucketname": "awsolproj",

"prefix": "rdsmysql/mybetainstd",

"keys": "00test.sql,01test.sql",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"bucketname": "awsolproj",

"prefix": "rdsmysql/mybetainstd",

"checktodo": "runscripts",

"torun": "true"

},

"torun": "true"

}

**[ update-dynamodb ]**

**Description**

This state inserts a new item into the DynamoDB table used to track each cloning operation

**Syntax (input)**

"updatedynamodb": {

"dbservice": "aurora|rds",

"dbinstance": "<string>",

"application": "<string>",

"environment": "<string>",

"source": "<string>",

"restoretype": "fromsnapshot|restorepoint|latestpoint|fastcloning",

"snapshot": "<string>",

"restoretime": "<string>",

"tablename": "<string>",

"torun": "true|false"

}

**Syntax (skip)**

"updatedynamodb": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* if **restoretype** is set to "fromsnapshot", **snapshot** is required
* if **restoretype** is set to "restorepoint", **restoretime** is required
* **snapshot** is the snapshot ARN, if the snapshot specified doesn’t exist or is not accessible the operation will fail
* **restoretime** must be specified in the following format: %Y-%m-%d %H:%M:%S, if doesn’t fall within the restore window of the source database the operation will fail
* **wait\_time** is in seconds

**Examples**

**- Aurora cluster restored with fastcloning:**

"updatedynamodb": {

"dbservice": "aurora",

"dbinstance": "auposinstd",

"application": "app2",

"environment": "development",

"source": "auposinstp",

"restoretype": "restorepoint",

"restoretime ": "2020-05-01 23:00:00",

"tablename": "dbalignment-awssol",

"torun": "true"

}

**- Aurora cluster restored with restoretime:**

"updatedynamodb": {

"dbservice": "aurora",

"dbinstance": "auposinstd",

"application": "app2",

"environment": "development",

"source": "auposinstp",

"restoretype": "fastcloning",

"tablename": "dbalignment-awssol",

"torun": "true"

}

**- RDS instance restored with latestpoint:**

"updatedynamodb": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"application": "app1",

"environment": "development",

"source": "mysqlinstp",

"restoretype": "latestpoint",

"tablename": "dbalignment-awssol",

"torun": "true"

}

**- RDS instance restored with fromshapshot:**

"updatedynamodb": {

"dbservice": "rds",

"dbinstance": "xplbetainstd",

"application": "beta",

"environment": "development",

"source": "xplbetainstp",

"restoretype": "fromsnapshot",

"snapshot": "arn:aws:rds:us-east-1:1234567890:snapshot:xplbetainstp-2020-04-15-22-00",

"tablename": "dbalignment-awssol",

"torun": "true"

}

**[ send-msg ]**

**Description**

This state notifies users about the completion of the cloning operation using SNS

**Syntax (input)**

"sendmsg": {

"dbservice": "aurora|rds",

"application": "<string>",

"environment": "<string>",

"dbinstance": "<string>",

"source": "<string>",

"restoretype": "fromsnapshot|restorepoint|latestpoint|fastcloning",

"topicarn": "<string>",

"torun": "true|false"

}

**Syntax (skip)**

"sendmsg": {

"wait\_time": <integer>,

"check": {

"torun": "false"

},

"torun": "false"

}

**Notes**

* **wait\_time** is in seconds

**Examples**

**- Aurora cluster restored with fastcloning:**

"sendmsg": {

"dbservice": "aurora",

"application": "app2",

"environment": "development",

"dbinstance": "auposinstd",

"source": "auposinstp",

"restoretype": "fastcloning",

"topicarn": "arn:aws:sns:us-east-1:123456789012:NotifyMe",

"torun": "true"

}

**- RDS instance restored with latestpoint:**

"sendmsg": {

"dbservice": "rds",

"application": "app1",

"environment": "development",

"dbinstance": "mysqlinstd",

"source": "mysqlinstp",

"restoretype": "latestpoint",

"topicarn": "arn:aws:sns:us-east-1:123456789012:NotifyMe",

"torun": "true"

}

If you want to use **AWS Secrets Manager** in order to securely manage the secrets of the RDS instances or the Aurora clusters, you can follow the steps below:

1. Create a new secret with Secrets Manager
2. Fix the rotation of the secret
3. Rotate manually the secret
4. Run the state machine again but with different parameters for the states "changemasterpwd" and "rotatemasterpwd"

Here a complete example with an RDS MySQL instance:

1. Create a new secret with Secrets Manager

User name: **admin**

Password: **temppwd123**

DB instance: **mysqlinstd**

Secret name: **/development/app1/mysqlinstd**

Rotation: **enabled**

Rotation interval: **1 day**

Lambda function: **SecretsManager-mysql-rotation**

1. Fix the rotation of the secret

aws secretsmanager get-secret-value \

--secret-id "/development/app1/mysqlinstd" \

--region us-east-1

aws secretsmanager update-secret-version-stage \

--secret-id "/development/app1/mysqlinstd" \

--version-stage "AWSPENDING" \

--remove-from-version-id "15544409-b558-4992-b4b9-edxxxxxxxxxx" \

--region us-east-1

aws secretsmanager get-secret-value \

--secret-id "/development/app1/mysqlinstd" \

--region us-east-1

1. Rotate manually the secret
2. Run the state machine again but with different parameters for the states "changemasterpwd" and "rotatemasterpwd"

"changemasterpwd": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"temppwd": "temppwd123",

"secret": "true",

"secretname": "/development/app1/mysqlinstd",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"checktodo": "checkpwd",

"torun": "true"

},

"torun": "true"

}

"changemasterpwd": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"temppwd": "temppwd123",

"secret": "true",

"secretname": "/development/app1/mysqlinstd",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"checktodo": "checkpwd",

"torun": "true"

},

"torun": "true"

}

"rotatemasterpwd": {

"dbservice": "rds",

"dbinstance": "mysqlinstd",

"secretname": "/development/app1/mysqlinstd",

"wait\_time": 15,

"check": {

"dbservice": "rds",

"secretname": "/development/app1/mysqlinstd",

"temppwd": "temppwd123",

"checktodo": "rotatepwd",

"torun": "true"

},

"torun": "true"

}