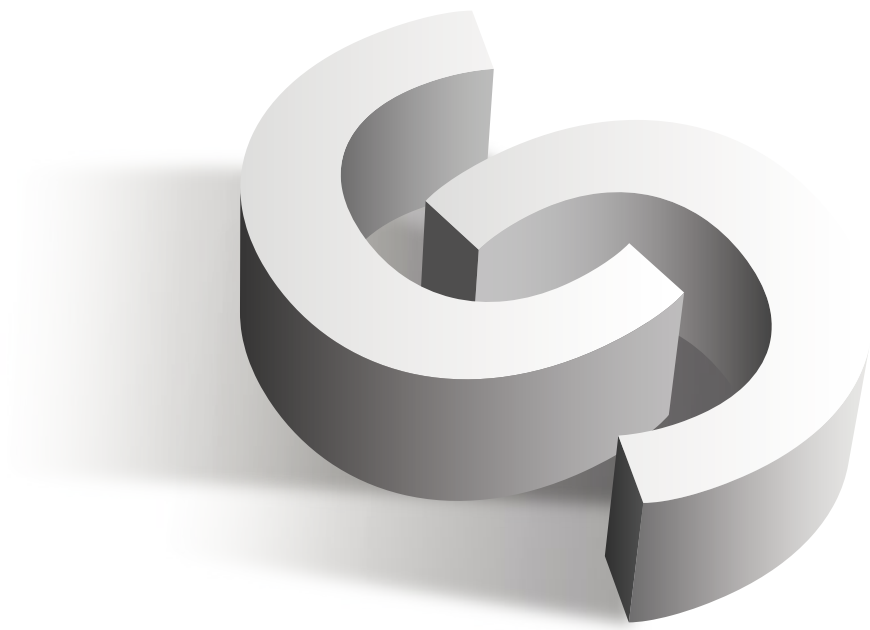


severalnines

# Disaster Recovery Planning for MySQL & MariaDB







# Table of Contents

<b>Introduction</b>	<b>4</b>
<b>Business considerations for Disaster Recovery</b>	<b>5</b>
Is 100% Uptime Possible?	5
Analysing risk	6
Assessing business impact	7
<b>Defining Disaster Recovery?</b>	<b>8</b>
Outage Timeline	8
Recovery Time Objective	9
Recovery Point Objective	9
RPO + RTO = 0 ?	9
<b>Disaster Recovery Tiers</b>	<b>10</b>
1. No Offsite Data	10
2. Database Backup with no Hot Site	10
3. Database Backup with Hot Site	14
4. Asynchronous Replication to Hot Site	16
5. Synchronous Replication to Hot Site	18
<b>In Conclusion</b>	<b>22</b>
<b>About ClusterControl</b>	<b>23</b>
<b>About Severalnines</b>	<b>23</b>
<b>Related Whitepapers</b>	<b>24</b>

# Introduction

The cost of downtime can vary significantly between different organizations, and in some cases, it may be enough to cause a company to go out of business. To mitigate the impact of downtime, organizations need an appropriate disaster recovery plan in place. But how much should a business invest? Designing a highly available system comes at a cost, and not all businesses and certainly not all applications need five 9's availability.

The best disaster recovery strategy for an application largely depends on its importance to the business, and more specifically, RTO (Recovery Time Objective) and RPO (Recovery Point Objective). RTO is the maximum period of time within which an application must be restored after a disruption. RPO is the determined maximum period of time that can pass during which data is lost. Can the business afford to lose 5 hours of data, or no more than 5 minutes? Can it be down for 4 hours, or at most 15 minutes? Knowing these numbers will go a long way in helping IT determine a disaster recovery strategy, as well as the best database solution to support it.

Therefore, disaster recovery can be implemented at different levels. They can be anything from periodic full backups that are archived offsite, to multi-datacenter setups with synchronous data replication. What is right for the business will vary by mission-criticalness.

As we will see in this whitepaper, outages are inevitable but understanding the timeline of an outage can help us better prepare, diagnose and recover from one. With regards to the database, different mechanisms can be implemented as part of a DR plan in order to prepare and respond to an outage. Higher levels of DR require increasing amounts of eventualities that one would have to plan for. We will look at the different levels, and specifically at the database mechanisms required for each level. Finally, we will see how these mechanisms can be fully automated with ClusterControl, a management platform for open source database systems.

# Business considerations for Disaster Recovery

## Is 100% Uptime Possible?

Is 100% uptime and availability possible? If it is possible, then why compromise on the SLA? Unfortunately, there is no such thing as 100% uptime.

One might be misled to believe that 100% is possible. For instance, hosting companies make a big deal about uptime guarantees. Those who actually offer an 100% uptime guarantee will usually specify exceptions:

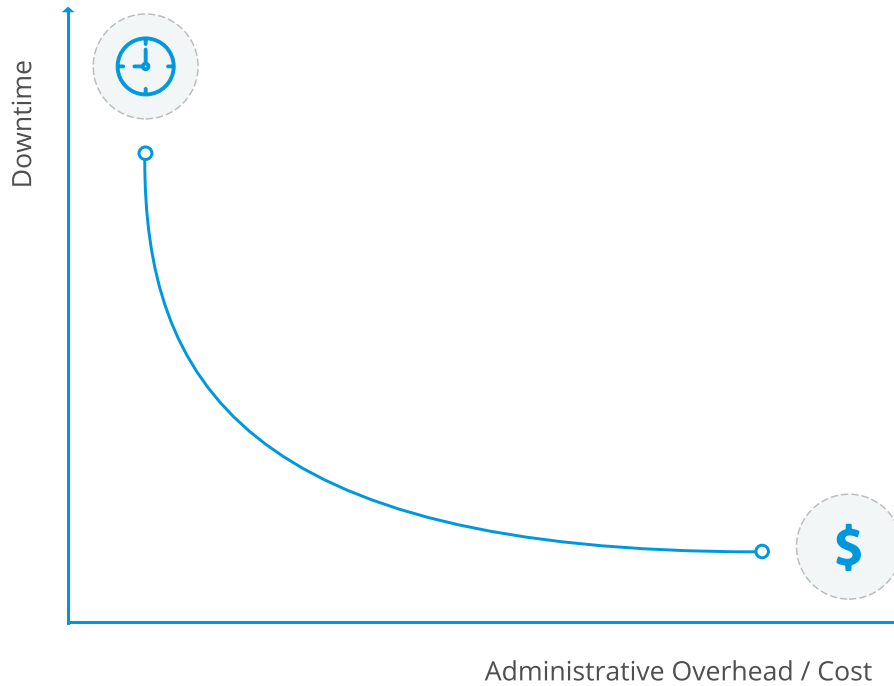
- Planned outages (e.g. server or network maintenance)
- Failure of network, power or facilities delivered by an upstream provider
- DOS attacks, hacker activity or other malicious events
- Acts of God (e.g., weather related - hurricane, flood)

Some providers may offer compensation in order to honour the '100% guarantee', for instance in the form of service credits. While this is attractive in a marketing pitch, it certainly won't stop your service from going offline or compensate you for lost business.

So, if we cannot achieve 100% uptime, what does it take to get close to 100%? Reducing downtime comes down to RTO (Recovery Time Objective) and RPO (Recovery Point Objective). The former refers to the last point in time that, in our case the database, can be recovered to. The latter is about how quickly databases can be recovered and operations resumed.

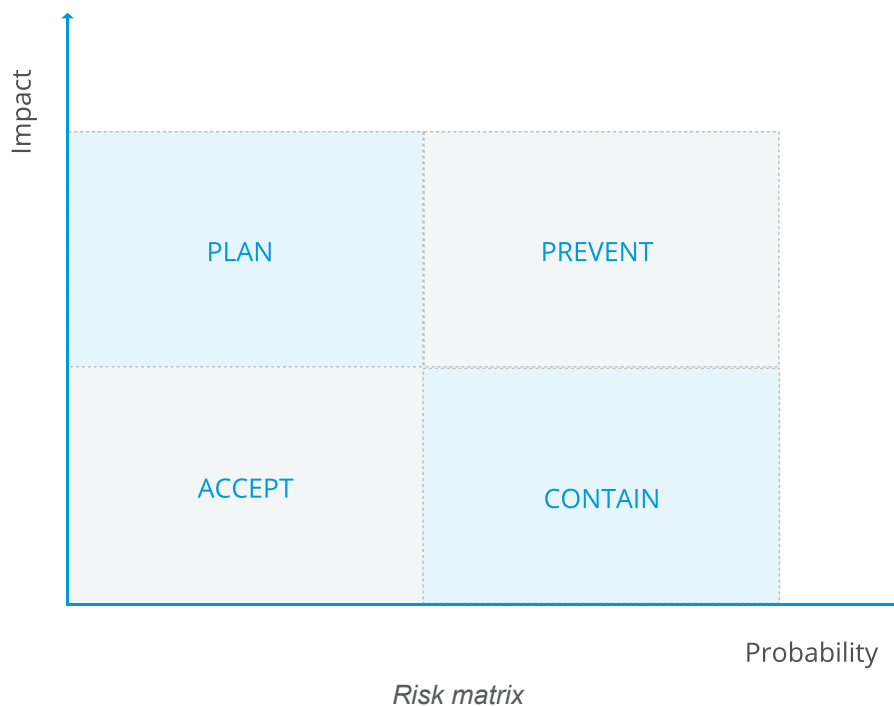
To minimize data loss, we need to have multiple copies of data in multiple places. We design our infrastructure in different layers and abstract each layer from the one below it. For instance, we build clusters of database instances so we can protect ourselves against a hardware failure. We replicate databases across datacenters so we can protect ourselves against a datacenter failure. Every additional layer adds complexity, which can become a nightmare to manage. And then, there is the cost.

Typically, the lower the downtime, the more the database will cost to run. We have to mitigate issues on several fronts, so we build redundancy into all the different layers. And because low downtime numbers often come with an increase in infrastructure complexity, the associated administrative overhead follows a similar curve. A high-availability database might find you managing a distributed setup with physically separated datacenters, redundant replication channels, management of data consistency during failovers, and more.



## Analysing risk

Outages can be costly, there is often a financial impact and they can do harm to the business. One common misconception from system owners is the difficulty in achieving high availability, and how both cost and complexity can escalate exponentially. There are so many internal and external threats and vulnerabilities that could negatively impact IT assets - from server failures, data corruption caused by software bugs, and hacked systems to human error or entire datacenter failures caused by fire or flood. Risks need to be analysed, with a framework that helps decide what action to take.



For instance, it is possible to use past statistics that show how often a datacenter loses power or network connectivity.

Based on that, we would know whether:

1. it rarely happens, but the impact is low - so we can live with the risk of it going down
2. it rarely happens, but the impact is high - so we need to plan the steps on how to address it
3. it happens frequently, but the impact is low - so will take appropriate steps to minimize the likelihood of it happening
4. it happens frequently, and the impact is high - so we will actively work on mitigating the risk

## Assessing business impact

Some applications are more important than others because they serve the business as a whole. To assess the importance of a database, we need to understand what the data means to the business and what business function it enables. What would the impact be on the organization if that business function cannot be performed? Which applications and databases does that function rely on?

Database	Criticality	Nº of users affected	RTO	RPO	Owner
Authentication	High	High	5 min	4 hours	Security team
Website DB	Medium	Medium	12 min	2 hours	DBA team
Reporting DB	Low	Low	8 hours	< 1 min	DBA team

*Classifying databases by criticality*

Business impact is not always easy to measure. SLA breach leading to fines would be straightforward. Lost sales could be quantifiable based on previous statistics, but it could also mean lost customers or reputation - the impact of which would be hard to put a number on.

# Defining Disaster Recovery?

A disaster usually causes an outage, which means system downtime and potential loss of data. Once we have detected the outage, we trigger our DR plan to recover from it.

But first, let's illustrate all this with a little story. Let's imagine a train with passengers going from Stockholm to Oslo, a journey that takes just over 5 hours. The train breaks down about an hour after leaving Stockholm.

The breakdown is the disaster, which is the point in time when the service stopped.

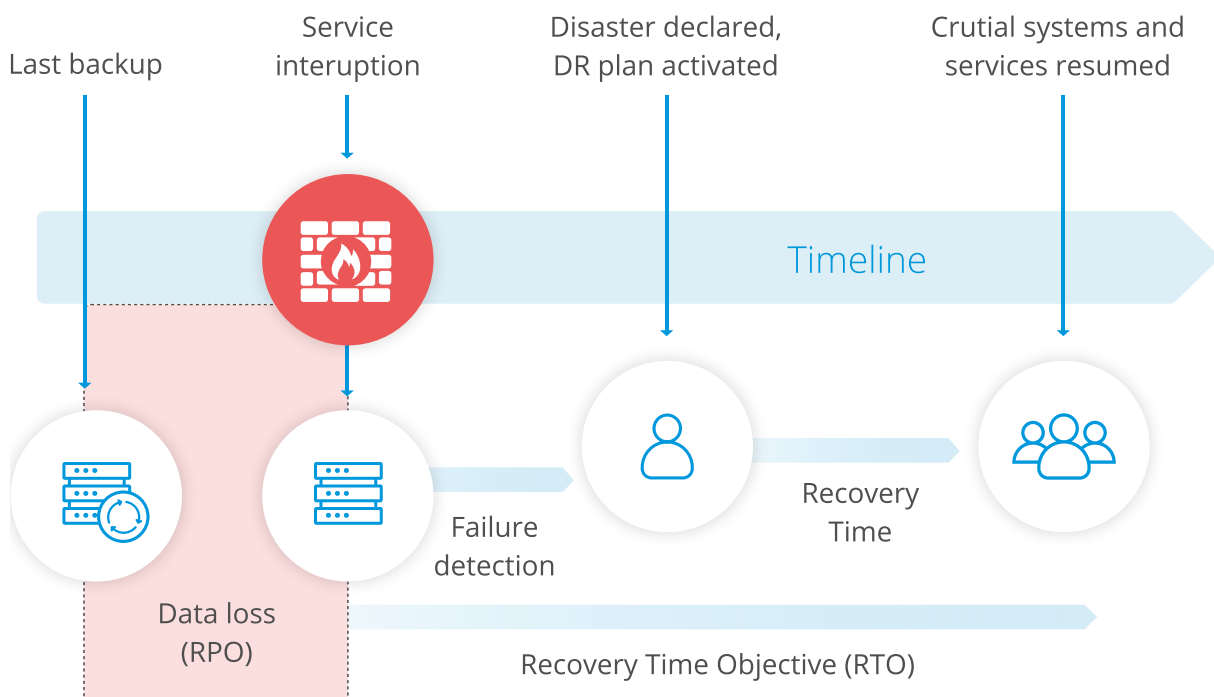
The disaster recovery plan for the train is the process of calling for help, troubleshooting the problem, getting spare parts and repairing the train, so it can continue its journey to Oslo.

DR is not to be confused with Business Continuity (BC), even though the terms are often used together. BC would be the process of arranging for a replacement train, or some alternative means of transport, so the services can continue between Stockholm and Oslo.

## Outage Timeline

The focus on uptime and minimizing the risk for failures usually translates into defining and meeting recovery point objectives (RPOs) and recovery time objectives (RTOs) for the different layers of the infrastructure.

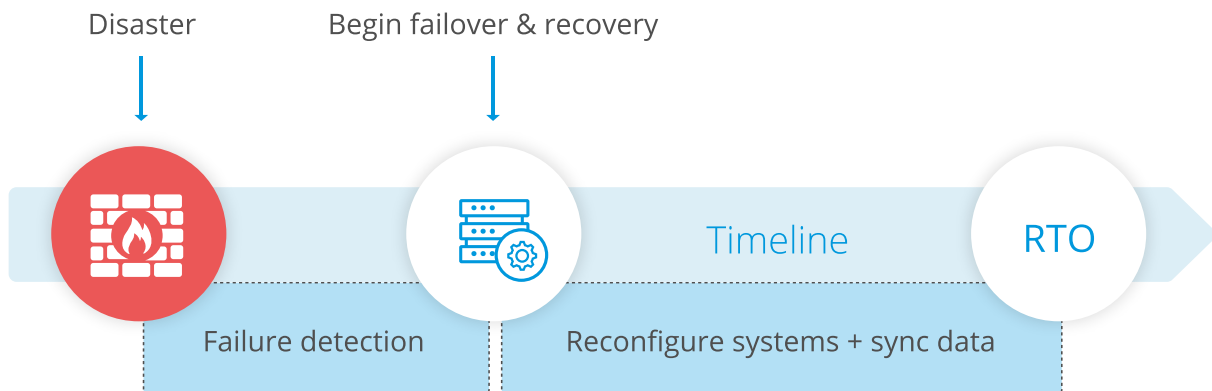
The below diagram describes the timeline of an outage.





## Recovery Time Objective

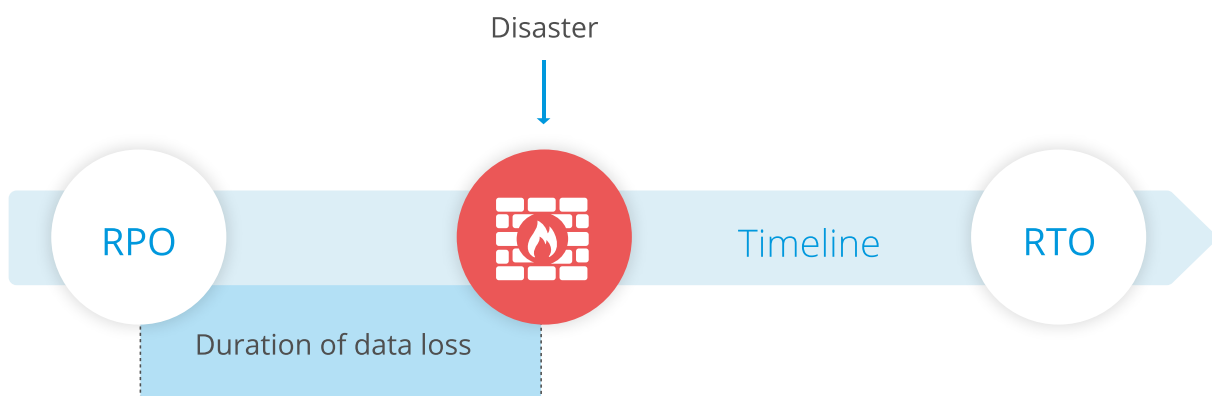
Recovery Time Objective (RTO) is the maximum acceptable length of time that your database can be offline. This includes the time to detect the failure, understand what it is and activate failover/recovery procedures.



## Recovery Point Objective

Recovery Point Objective (RPO) is the maximum acceptable length of time during which committed data in the database might be lost due to a major incident. The metric does not address the amount of data lost, or the quality of the data.

RPO will vary based on the type of data. Frequently modified customer data could have an RPO of just a few minutes, whereas less critical, infrequently modified data could have an RPO of several hours.



## RPO + RTO = 0 ?

Is it possible to have zero RPO and zero RTO? Considering the amount of eventualities that one would have to plan for in order to get there, zero downtime or 100% uptime seems more like a utopian dream. That is why high uptime is often measured in a number of 9's.

Meeting high uptime goals for infrastructure is extremely complicated - it requires application clustering, database clustering, storage clustering, network bonding, server load balancing and traffic management, file replication and clustering, database replication, monitoring, split brain prevention, site to site failovers, data integrity, security,... the list goes on.

# Disaster Recovery Tiers

As we mentioned earlier, there is no one size fits all solution. The type of database, high availability configuration and management software we employ will play an important role on what we can deliver, simply by virtue of the features available. For instance, there is no shortage of options in the MySQL and MariaDB ecosystem for meeting different levels of DR requirements. In this chapter, we will cover how different tiers of DR can be addressed by database technology. We'll see specifically how ClusterControl can be employed in each DR tier.

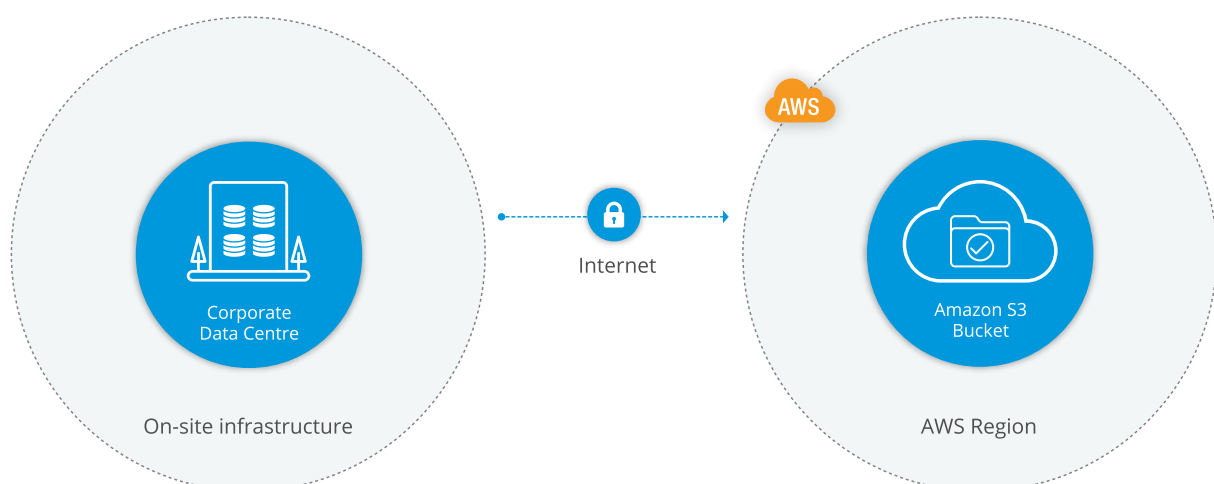
## 1. No Offsite Data

In this tier, DR has not been considered. There is no backup of data or systems. In case of disaster, it may not be possible to recover the data.

For companies in this category, we would recommend to keep your fingers crossed ...

## 2. Database Backup with no Hot Site

At this level, the database is backed up and the backup files are archived offsite. In case of failure of the primary site, the backup files would be safe and secure off-site. However, there are no systems to restore the backups on, in case the main site would fail. In the example below, AWS is only used as the offsite archive for backups. We'll assume the applications/databases cannot be installed on AWS (hence the 'no Hot Site' label).



*Uploading and archiving backups on AWS S3*

Depending on how often the backups are created and shipped, one must be prepared to accept hours or days of data loss. For larger databases, it might not be feasible to do frequent full backups as they can be time consuming. Taking incremental backups is a

way of decreasing the amount of time that it takes to do a backup. Incremental backups only backup the data that has changed since the previous backup. These two types of backups can be combined in order to have more up-to-date data in the backups, while limiting any performance degradation of our production database. Note though that incremental backups can be time consuming to restore, thus increasing RTO.

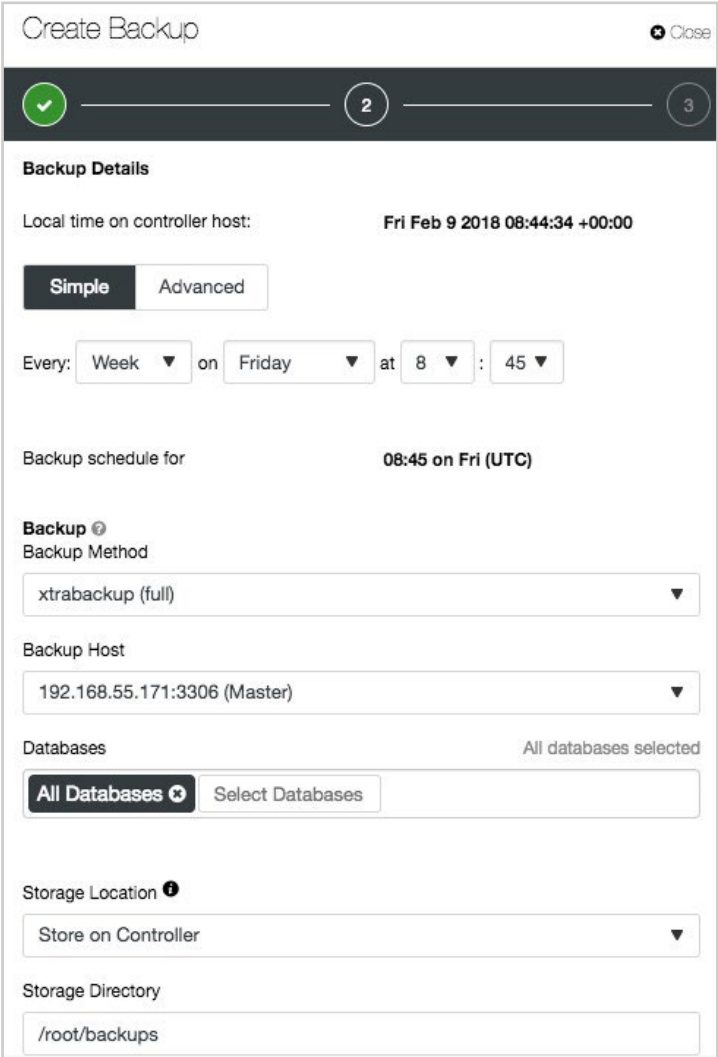
To further decrease RPO, we might want to consider backups that are PITR-compatible (Point in Time Recovery). We will see this in the next DR tier.

Scheduling a backup is not enough, backup data also needs to be verified for consistency and integrity. The concept of Schrödinger's backup illustrates the point - "The condition of any backup is unknown until a restore is attempted". A notification that our backup was successfully completed and uploaded offsite is a good start, but the true test comes when a restore is required.

When creating backups, we can create physical or logical copies of the database. Different types of backups are useful for recovery from different failure scenarios.

Another consideration is encryption. Backups are at risk if they are not encrypted. When managed appropriately, encryption provides a layer of security as well as peace of mind that, should any backup media fall into the wrong hands, the likelihood of data exposure is slim.

ClusterControl provides backup features that take into account all the above considerations. It supports a mixture of methods, both physical and logical.



Create Backup Close

1 2 3

**Backup Details**

Local time on controller host: **Fri Feb 9 2018 08:44:34 +00:00**

**Simple** Advanced

Every: Week on Friday at 8 : 45

Backup schedule for **08:45 on Fri (UTC)**

**Backup** ?

Backup Method

xtrabackup (full) ▼

Backup Host

192.168.55.171:3306 (Master) ▼

Databases All databases selected

All Databases × Select Databases

Storage Location !

Store on Controller ▼

Storage Directory

/root/backups

The backup files can be encrypted.

Create Backup Close

Backup Settings

Desync node during backup

Backup Locks

Xtrabackup Parallel Copy Threads

Network Streaming Throttle Rate (MB/s)

Use PIGZ for parallel gzip

Enable Encryption

Retention  31 days (Default)  Custom  Keep Forever

One important follow-up item is the state of the created backup. ClusterControl provides email notifications and will notify about the status.

Category Settings

Select how you want alarms/events delivered

Alarm/Event Category	CRITICAL	WARNING	INFO
All Event Categories	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Custom"/>
Network	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
CmonDatabase	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
Mail	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
Cluster	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
ClusterConfiguration	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
ClusterRecovery	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
Node	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
Host	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
DbHealth	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
DbPerformance	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
SoftwareInstallation	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>
<b>Backup</b>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Digest"/>
Unknown	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>	<input type="text" value="Deliver"/>

Backup files can be automatically uploaded to the cloud (AWS S3, Google Cloud Storage and Azure Storage). Note that it is still important to keep at least a copy of the latest backup (hopefully verified as well) in the datacenter, since in case the

infrastructure comes up again, it is much faster to restore from local backup files as opposed to having to download them from an external service first, before being able to start recovery.

## Create Backup Close

✓ ✓ 3

### Cloud Settings

Cloud Credentials Profile ?

████████ AWS

Select existing bucket **Create new bucket**

mysqldump-db-shops **Create**

Root Location/Folder (Optional, will be saved to root folder if empty) ?

Enter Location

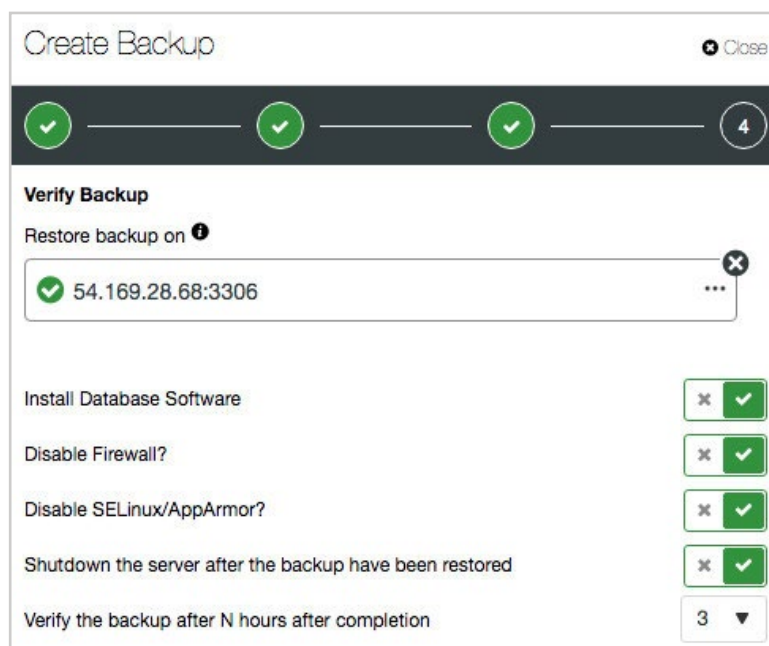
Subfolder ?

/

Backup details can be viewed from the backup list.

Backup List (1-2 out of 2)	Status	Method	Databases	Storage Location	Size
<b>Backup Set: 3</b> Restore More Actions	Completed at 11/07/2017 @ 8:42AM (UTC)	xtrabackup (full)	all		270.2 MB
<b>Full Backups</b>				<b>Backup Details</b>	
Local Storage 10.0.0.156/root/backups/BACKUP-3				Host: 10.0.0.25 Included Databases: all (270.2 MB)	
<b>Backup Set: 1</b> Restore More Actions	Completed at 11/07/2017 @ 7:12AM (UTC)	mysqldump (pitr compatible)	all		220.9 MB

When scheduling a backup, it is possible to schedule it for automatic restore verification.

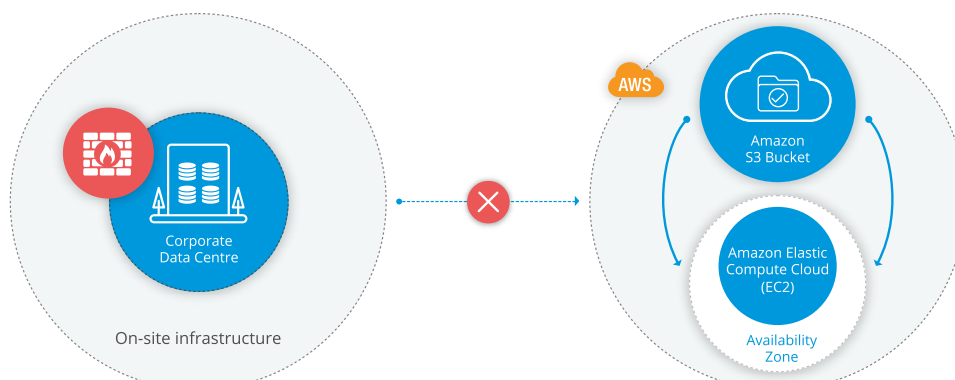


For this level of DR, we can expect long RTO. The infrastructure needs to be re-created somewhere, so we'd need to find a datacenter with servers. After that, the backup files need to be transferred to the new database servers and then restored.

### 3. Database Backup with Hot Site

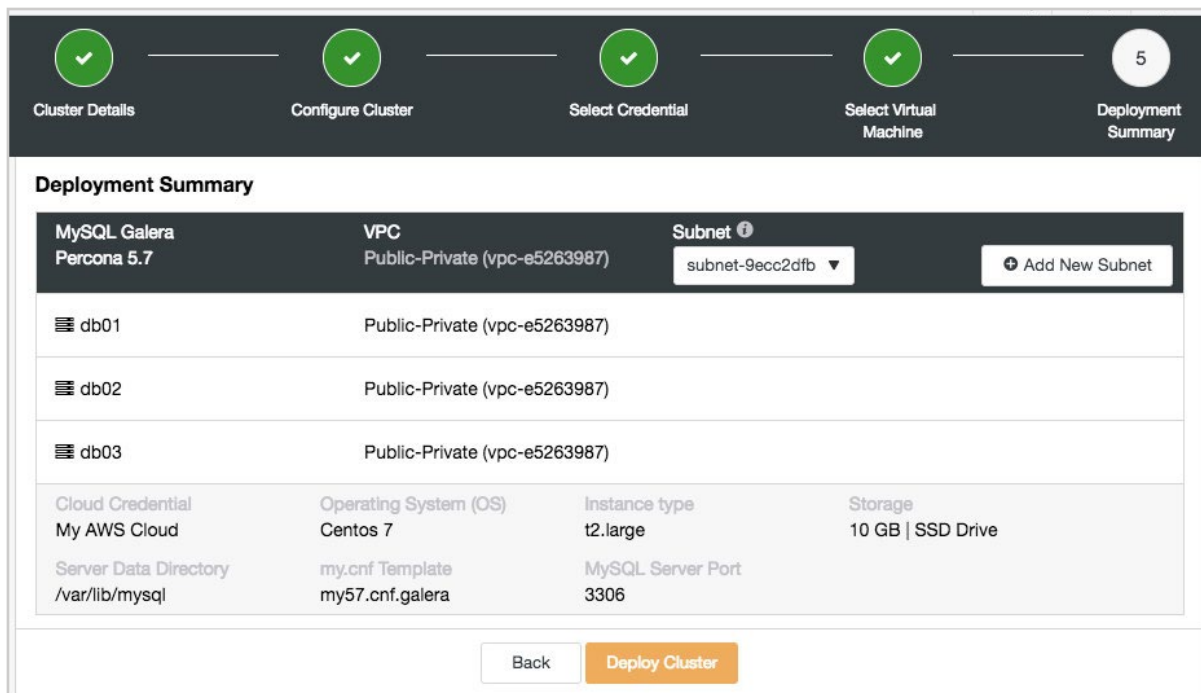
At this level, we are making regular backups but we also have a second site with infrastructure that can be used to restore systems in the event of disaster. Using this solution, we will have to re-install our database systems (as well as applications, etc.) from scratch and restore data. Recovery time will be predictable, since we have the infrastructure and are able to run tests beforehand. In the example below, the infrastructure can be re-created on AWS, which makes it a hot site. Pre-configured Amazon Machine Images (AMIs) can be used to quickly provision the application environment when needed.

RPO can be reduced by taking backups that are PITR compatible (Point in Time Recovery). The backups would include binary log files, so the database can be restored at an arbitrary time.

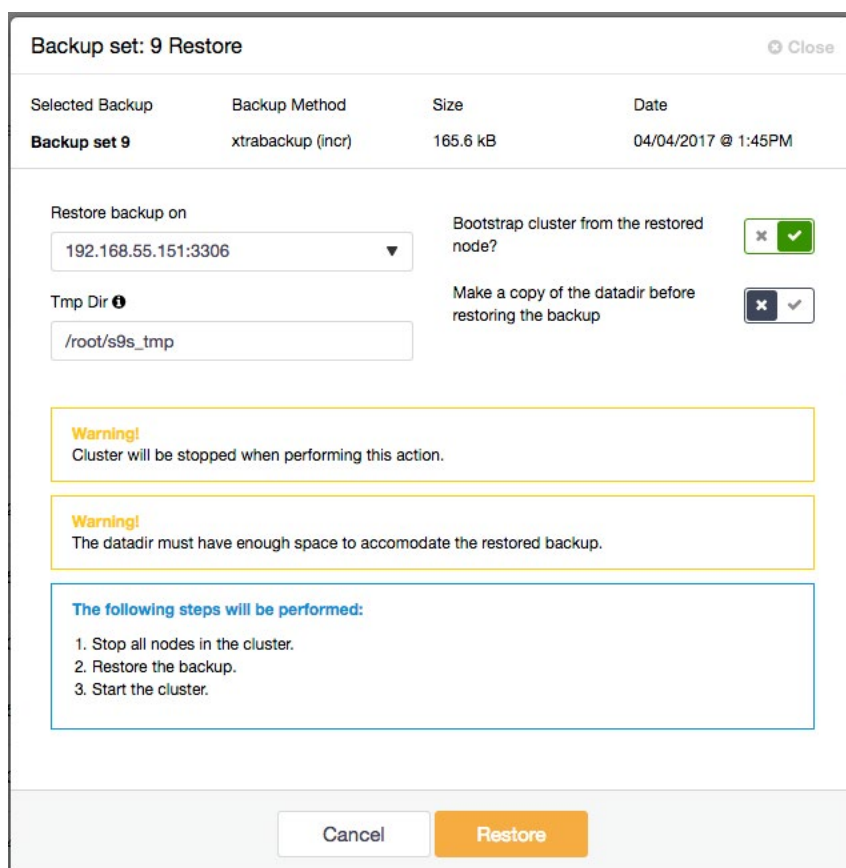


Uploading backups in AWS, with possibility to re-create the database on EC2 infrastructure

ClusterControl can be used to deploy the database. For MySQL and MariaDB, it supports all the major types of high availability setups (MySQL Replication, Galera Cluster, MySQL NDB Cluster, MySQL Group Replication). Through its cloud integration, it is able to automatically spin up instances on AWS, Azure and Google Cloud before deploying the databases. The screenshot below shows the deployment of a Galera Cluster on AWS.



Once the database is deployed, it is time to restore the latest backup.





For lower RPO, we would use Point in Time Recovery. There will be two options for that - "Time Based" and "Position Based". For "Time Based", it is enough to pass the day and time. "Position Based" is a more precise way to restore, one can pass the exact position (statement) up to which we want to restore.

Restore Backup Close

1 2 3 4

Select full backup to be restored

Backup Set 12 / xtrabackup (incr) / 19/04/2018 @ 3:17PM

Backup method and type	Size	Date
xtrabackupincr	37.6 MB	2018-04-19 13:18:04 (UTC)

Select where you want to restore this backup from

10.0.2.15:/root/backups/BACKUP-12

Point In Time Recovery (PITR) new ✓

Time Based Position Based

Binary Log Name ?

Enter binary log name

Log Stop Position ?

Enter stop log position

The entire process with step by step instructions is described in the blog - [Full Restore of a MySQL or MariaDB Galera Cluster from Backup](#).

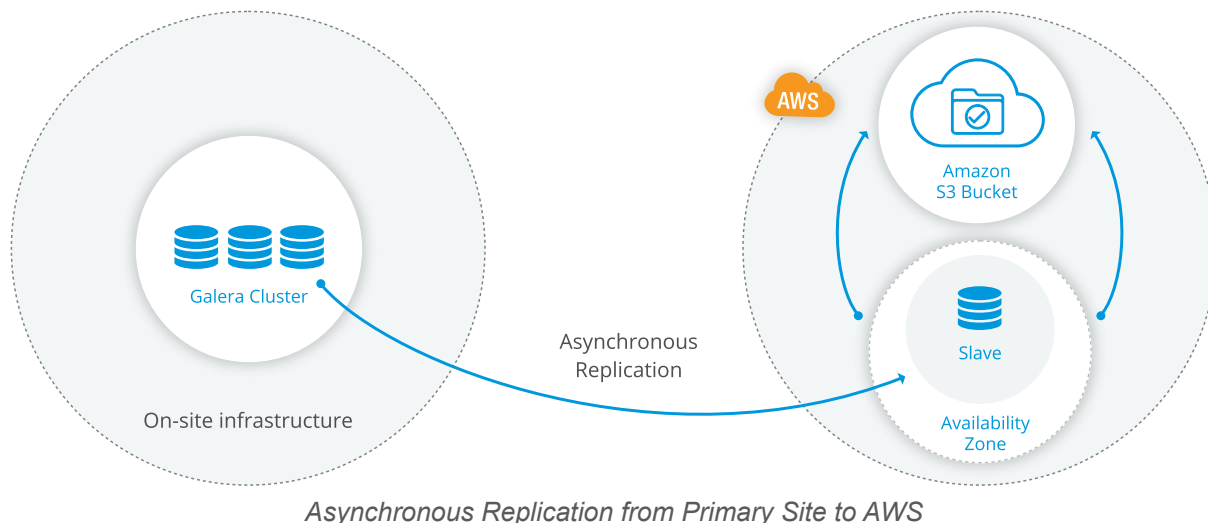
This level of DR has a similar RPO as the previous one, with perhaps a shorter RTO - since the infrastructure can be instantiated on-demand. Since the backup files would be within the same network, that should speed up the data recovery phase.

## 4. Asynchronous Replication to Hot Site

At this level, the hot site has backup infrastructure running. It is continuously kept up-to-date with the primary site, so it has a near-realtime copy of data. The main requirement is low RTO. The backup infrastructure has "almost current" data, enabling fast failover.

In the event of a disaster, there will be little to no data loss. The service can be up and running with little downtime. Note that backups are still important here, in case we would lose both the active and the standby system - for instance, if an operator would happen to drop a table by mistake, and the error would propagate to the standby.





Low RPO and RTO comes at a cost. This level of DR requires dedicated hardware in the hot site, ready to take over at any time.

To keep costs low, it is possible to do a minimal configuration of applications and databases in the DR site. For instance, the primary site could be running a database cluster, e.g., 3 nodes, replicating asynchronously to a single slave on the DR site. The same would be true for the rest of the infrastructure, e.g., the primary site would have a cluster of application servers whereas the hot site might just have one instance. This is an economical way of providing at least a minimal service.

However, note that the minimal capacity of the hot site might also mean there are insufficient resources available to provide service to all users, with acceptable quality. Overload controls need to be in place, so as to either switch off certain non-critical functions, or to offer minimally acceptable service to selected users. A degraded service would be offered until the primary site is repaired and service returned to normal. Alternatively, the hot site can be dynamically scaled out (e.g., turn the slave into a cluster, and add more application servers) to increase capacity so it can handle the full workload. Overload control is crucial here, so as to avoid the risk of flooding the hot site and bringing it down when traffic is failed over to it.

ClusterControl can be used to add an asynchronous slave to an existing setup, as we can see in the screenshot below. Note that the slave can also be delayed. A mistake on the master, for example an UPDATE without a WHERE clause, will be replicated nearly instantly to our slave with disastrous consequences. Having the slave server lagging behind can help, as we would be able to roll back the delayed slave to the time just before the destructive command.

Note that, since a delayed slave is lagging behind by a number of minutes (e.g., 30 minutes), we cannot just instantaneously switch over to it or else we would lose that 30 minutes window of data. A regular slave would save us from a number of failures that would take down our active cluster, whereas a delayed slave would be useful in case of operator error. Therefore in some cases, we might need to have both a normal slave server as well as a delayed slave on the hot site in order to address these different risks.

Using ClusterControl, the slave can be built from a freshly streamed backup from the master to the slave. Or the slave can also be staged from an existing backup, before it is connected to the master and catches up with it.

Add Replication Slave
Close

New Replication Slave

Existing Replication Slave

Master Hostname / IP

Netcat port

Slave Hostname / IP

Port

Delay the slave ?

Rebuild from a Backup ?

Install the Slave server

MySQL server slave installation

Disable Firewall?

? The slave will be setup from a streamed XtraBackup from the

Cancel

Add

✔ Completed  
at 2018-05-16 16:25:53 (CEST)

Backup:

✔ Select a Backup

- Backup Set 1935 / mysqldump / 18/05/2018 @ 3:00PM
- Backup Set 1932 / xtrabackup (full) / 18/05/2018 @ 2:15PM
- Backup Set 1931 / xtrabackup (full) / 17/05/2018 @ 5:30PM
- Backup Set 1930 / xtrabackup (full) / 17/05/2018 @ 4:25PM
- Backup Set 1928 / mysqldump / 17/05/2018 @ 3:00PM
- Backup Set 1925 / xtrabackup (full) / 17/05/2018 @ 2:15PM
- Backup Set 1924 / xtrabackup (full) / 16/05/2018 @ 5:30PM
- Backup Set 1923 / xtrabackup (full) / 16/05/2018 @ 4:25PM
- Backup Set 1921 / mysqldump / 16/05/2018 @ 3:00PM
- Backup Set 1918 / xtrabackup (full) / 16/05/2018 @ 2:15PM
- Backup Set 1917 / xtrabackup (full) / 15/05/2018 @ 5:30PM
- Backup Set 1916 / xtrabackup (full) / 15/05/2018 @ 4:25PM
- Backup Set 1914 / mysqldump / 15/05/2018 @ 3:00PM
- Backup Set 1911 / xtrabackup (full) / 15/05/2018 @ 2:15PM
- Backup Set 1910 / xtrabackup (full) / 14/05/2018 @ 5:30PM
- Backup Set 1909 / xtrabackup (full) / 14/05/2018 @ 4:25PM
- Backup Set 1907 / xtrabackup (full) / 13/05/2018 @ 5:30PM
- Backup Set 1906 / xtrabackup (full) / 13/05/2018 @ 4:25PM
- Backup Set 1904 / mysqldump / 13/05/2018 @ 3:00PM
- Backup Set 1901 / xtrabackup (full) / 13/05/2018 @ 2:15PM
- Backup Set 1899 / xtrabackup (full) / 13/05/2018 @ 2:00AM
- Backup Set 1898 / xtrabackup (full) / 12/05/2018 @ 5:30PM
- Backup Set 1897 / xtrabackup (full) / 12/05/2018 @ 4:25PM
- Backup Set 1895 / mysqldump / 12/05/2018 @ 3:00PM

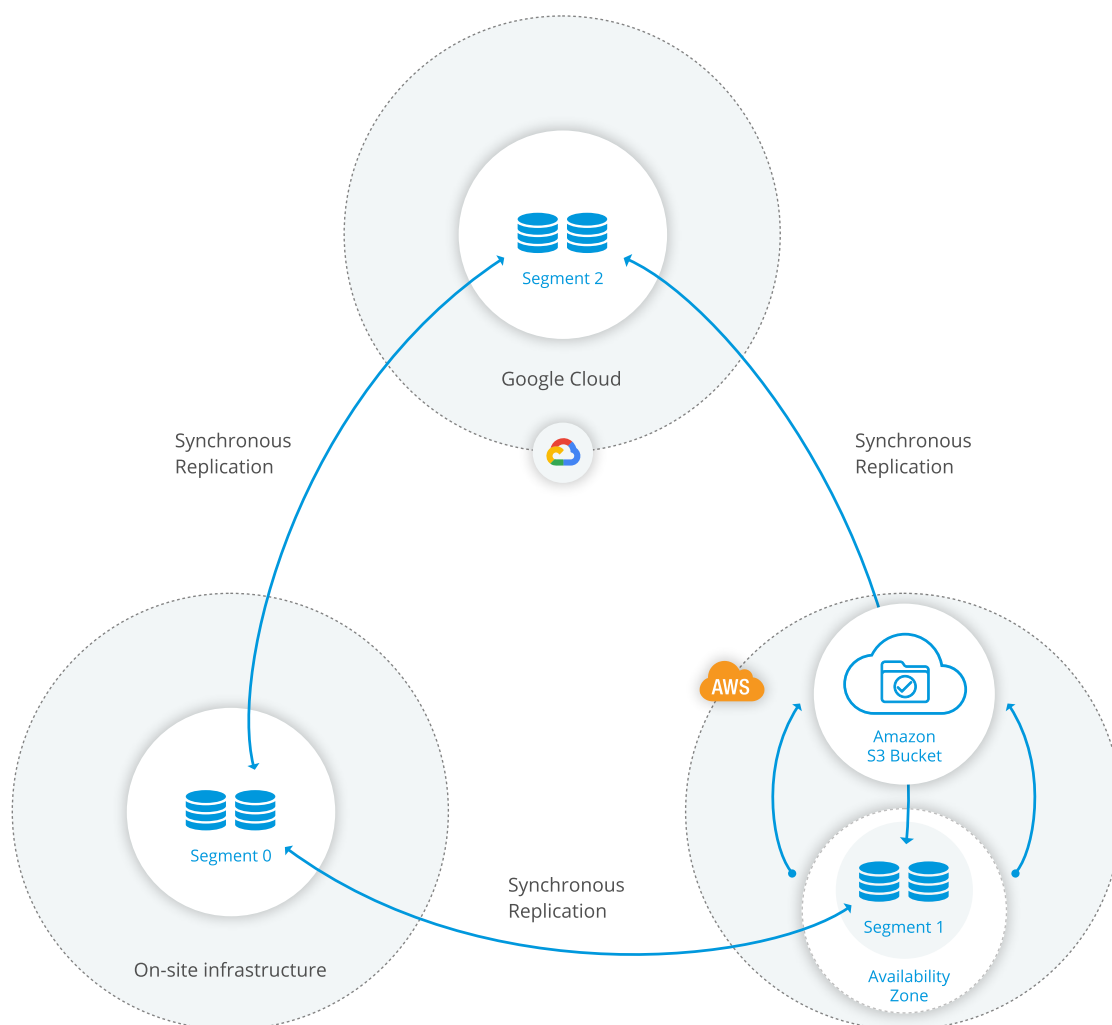
Finally, note that the failover to the hot site is manual. The DR team will be in charge of moving traffic over to the hot site. Once the primary site is repaired and it comes back up again, we will have to reverse the replication flow to synchronize data. Failback is possible once the data on the primary site is up-to-date.

## 5. Synchronous Replication to Hot Site

Finally, for the highest tier of DR, we would have redundant copies of our data in at least 2 hot sites. This is for businesses aiming at minimal RTO and RPO. Data is synchronously replicated across 3 sites. All the 3 sites are 'masters', i.e., they are all able to provide service to users. This is for businesses with little or no tolerance for data loss and who need to restore data to applications rapidly. Failover can be almost instantaneous, since the sets of data on the primary site and the hot site have the same

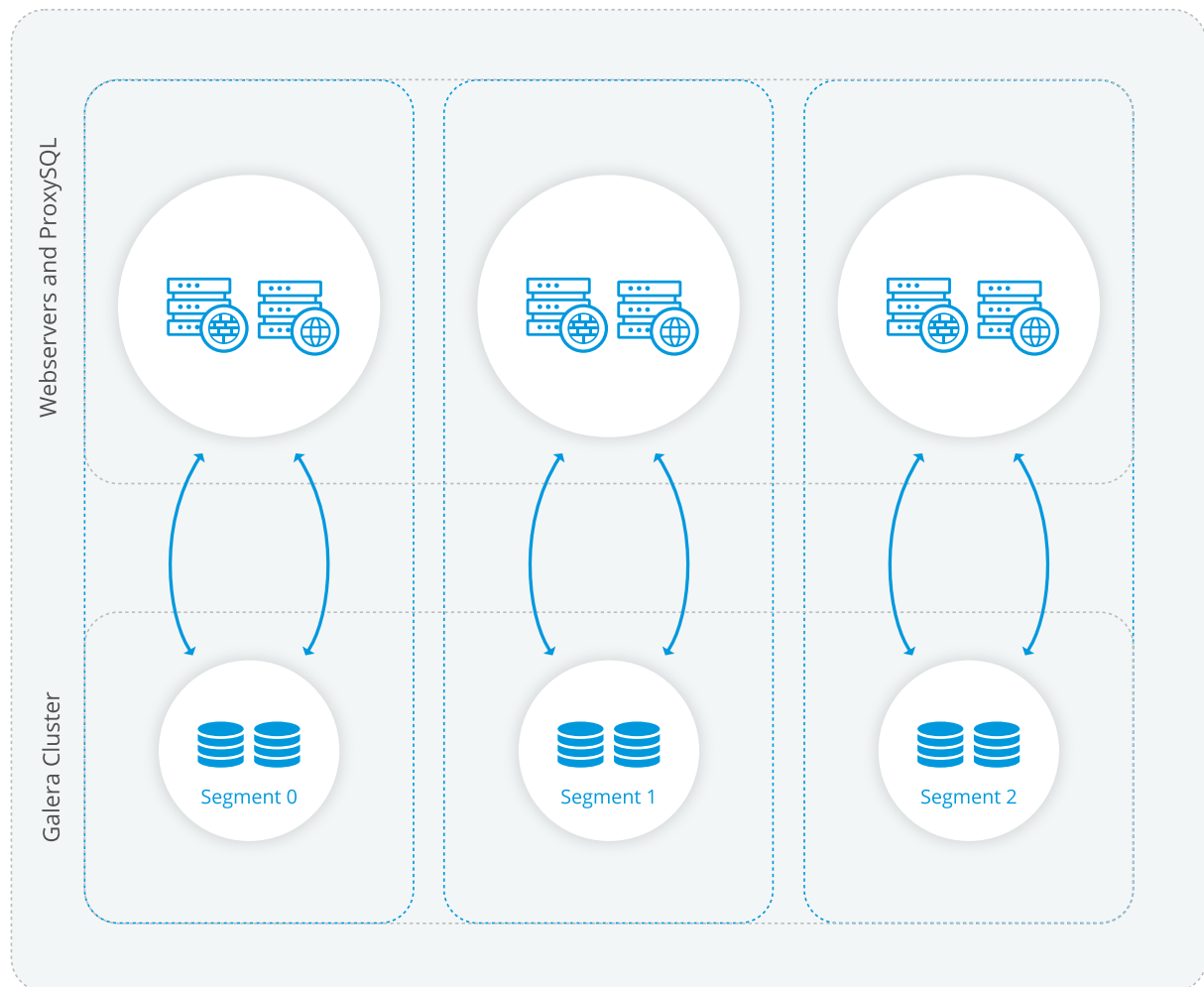
transactional state. Failure detection time will be the main culprit that adds to the RTO here. But once the failure is detected, failover is instantaneous and automatic.

Note here that 2 sites will not be enough for failover to be automatic, as there is no way to detect a network partition and a resulting split brain situation. Split brain is when a cluster of nodes is partitioned into e.g. two smaller clusters after a temporary failure of network links. Two partitions of equal number of nodes are created, and each partition believes it is the only active cluster. This is a harmful state, as data is changed on either partition, without having been replicated to the peer. In this situation, it is likely that two diverging sets of data will be created, which cannot be trivially merged. Galera Cluster is a multi-master replication technology for MySQL and MariaDB. It operates in a way that, in case of network partition, only the partition holding the majority of nodes will be allowed to operate. Therefore, having 3 datacenters with 2 nodes each would ensure that the system can tolerate the failure of one datacenter. It is common to use datacenters in separate geographical regions, although in a cross-region scenario, there is more latency between the nodes due to the longer network channels.

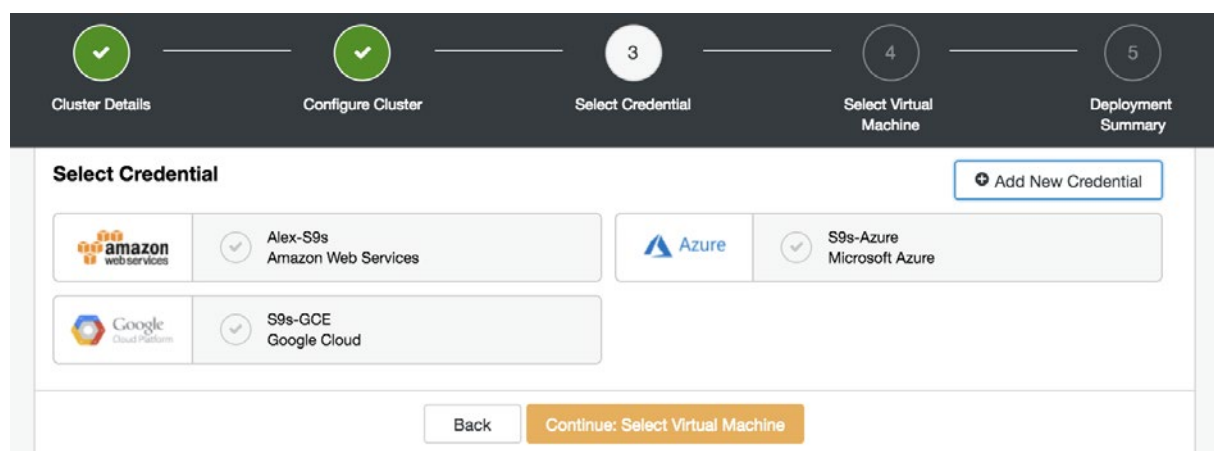


In the words of Amazon's CTO Vernal Vogel's, "Everything fails, all the time". And we know from experience that entire cloud regions can fail, regardless of vendor. Therefore, having the hot site(s) in a separate region can help so as to get away from dependencies that may exist between datacenters in the same region (e.g., Amazon Availability Zones). A multi-cloud DR strategy further reduces dependencies. We have seen previously that regional failures have sometimes cascaded to other regions, for instance, when a region gets a sudden surge in traffic as thousands of services are started at the same time.

Distributed database setups across multiple datacenters can be complex for applications to keep track of. For instance, failed instances or instances that are undergoing a recovery process should not be receiving traffic. Perhaps there are strategies around sending database updates to a specific set of nodes. Hence the need for a database proxy that can control traffic to the cluster, and at the same time abstract the complexity of the database layer from the applications.



ClusterControl supports the deployment of databases across the main cloud vendors - Amazon Web Services, Microsoft Azure and Google Cloud.



It also supports multiple database proxies on top of distributed database configurations. This allows administrators to easily set up routing rules for database traffic, and ensure that applications can easily connect to the setup via a single VIP.

The screenshot displays the ProxySQL monitoring dashboard. On the left, a sidebar lists various components: Controller (10.0.3.80), MySQL (10.0.3.50, 10.0.3.70, 10.0.3.60), HAProxy (10.0.3.80), Keepalived (10.0.3.70, 10.0.3.80), Garbd (10.0.3.80), MaxScale (10.0.3.80), and Proxysql (10.0.3.70, 10.0.3.80). The main area shows the ProxySQL 10.0.3.70 node status as 'Node is OK'. Below this, navigation tabs include Monitor, Top Queries, Rules, Servers, Users, Variables, and Scheduler Scripts. The 'ProxySQL Host Groups' section shows Hostgroup 20 and Hostgroup 10. Hostgroup 10 is active, with a table listing its members:

Hostname	Status	Conn. Used	Conn. Free
10.0.3.60 ...	✓ ONLINE	0	0
10.0.3.50 ...	✓ ONLINE	0	0
10.0.3.70 ...	✓ ONLINE	0	0

The 'ProxySQL Stats' section shows a 'Show Range' of '1 Hour Ago'. A line graph titled 'Questions' displays the number of questions per second over time. A tooltip for the current time, 'Mon, May 21, 2018 - 14:08:13', indicates 'questions: 0/sec'.

# In Conclusion

Organizations have historically deployed tape backup solutions as a means to protect data from failures, however the emergence of public cloud computing has also enabled new models with lower TCO than what has traditionally been available. It makes no business sense to abstract the cost of a DR solution from the design of it, so organizations have to implement the right level of protection at the lowest possible cost.

Those who require higher levels of DR would architect for availability, and the technology is available today for cross-region multi-master MySQL and MariaDB database systems. From a 2017 survey of over 1000 datacenter executives around the globe conducted by Uptime Institute<sup>1</sup>, over 68% have deployed some form of multi-site resiliency strategy. This is evidence that more companies are beginning to deliver mission-critical IT services through distributed data centers, either using synchronous or asynchronous replication to a hot site.

It is good to have failover by design, but actual failover tests are important to know that it actually works. As Netflix put it when they released their Chaos Monkey disaster testing system, ‘the best defense against major unexpected failures is to fail often’. There is probably not many organizations who can claim that<sup>2</sup>.

So what is holding these organisations back from testing their DR plan? The main reasons are down to cost and process complexity. There is an abundance of HA technologies designed to protect against different kinds of hardware or software failures. However, the more products we implement to protect against those, the more complex the environment becomes. Complexity makes systems harder to manage, and more prone to human errors. Complexity also has consequences in how DR processes are designed, and how ops teams interact with production systems when a DR plan is activated.

ClusterControl makes meeting the required SLA targets viable and cost-effective by wrapping the complexity of managing multiple operational procedures into a single product. We also expect the public cloud to become an integral part of the modern organization’s DR plan, because the amount of data that we’re having to manage and protect is increasing faster than IT budgets. Despite the costs involved, disaster recovery is not an option.

---

<sup>1</sup> [https://uptimeinstitute.com/webinars/2017\\_data-center\\_industry\\_survey\\_results](https://uptimeinstitute.com/webinars/2017_data-center_industry_survey_results)

<sup>2</sup> <https://www.zetta.net/resource/state-disaster-recovery-2016>

# About ClusterControl

ClusterControl is the all-inclusive open source database management system for users with mixed environments that removes the need for multiple management tools. ClusterControl provides advanced deployment, management, monitoring, and scaling functionality to get your MySQL, MongoDB, and PostgreSQL databases up-and-running using proven methodologies that you can depend on to work. At the core of ClusterControl is its automation functionality that lets you automate many of the database tasks you have to perform regularly like deploying new databases, adding and scaling new nodes, running backups and upgrades, and more. Severalnines provides automation and management software for database clusters. We help companies deploy their databases in any environment, and manage all operational aspects to achieve high-scale availability.

# About Severalnines

Severalnines provides automation and management software for database clusters. We help companies deploy their databases in any environment, and manage all operational aspects to achieve high-scale availability.

Severalnines' products are used by developers and administrators of all skills levels to provide the full 'deploy, manage, monitor, scale' database cycle, thus freeing them from the complexity and learning curves that are typically associated with highly available database clusters. Severalnines is often called the "anti-startup" as it is entirely self-funded by its founders. The company has enabled over 12,000 deployments to date via its popular product ClusterControl. Currently counting BT, Orange, Cisco, CNRS, Technicolor, AVG, Ping Identity and Paytrail as customers. Severalnines is a private company headquartered in Stockholm, Sweden with offices in Singapore, Japan and the United States. To see who is using Severalnines today visit:

<https://www.severalnines.com/company>



Deploy



Manage



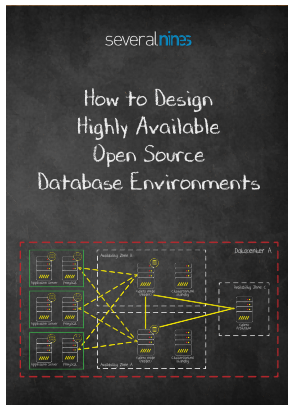
Monitor



Scale



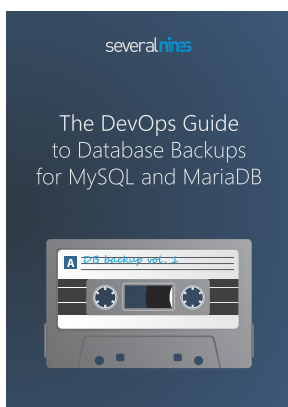
# Related Whitepapers



## How to Design Highly Available Open Source Database Environments

These days high availability is a must for any serious deployment. Long gone are days when you could schedule a downtime of your database for several hours to perform a maintenance. Making a database environment highly available is one of the highest priorities nowadays alongside data integrity. For a database, which is often considered the single source of truth, compromised data integrity can have catastrophic consequences. This whitepaper discusses the requirements for high availability in database setups, and how to design the system from the ground up for continuous data integrity.

[Download whitepaper](#)



## The DevOps Guide to Database Backups for MySQL and MariaDB

This whitepaper discusses the two most popular backup utilities available for MySQL and MariaDB, namely mysqldump and Percona XtraBackup. It further covers topics such as how database features like binary logging and replication can be leveraged in backup strategies. And it provides best practices that can be applied to high availability topologies in order to make database backups reliable, secure and consistent.

[Download whitepaper](#)



## Management and Automation of Open Source Databases

Proprietary databases have been around for decades with a rich third party ecosystem of management tools. But what about open source databases? This whitepaper discusses the various aspects of open source database automation and management as well as the tools available to efficiently run them.

[Download whitepaper](#)



Database outages are almost inevitable and understanding the timeline of an outage can help us better prepare, diagnose and recover from one. To mitigate the impact of downtime, organizations need an appropriate disaster recovery (DR) plan. This white paper provides essential insights into how to build such a plan, discussing the database mechanisms involved as well as how these mechanisms can be fully automated with ClusterControl, a management platform for open source database systems.



Deploy



Manage



Monitor



Scale