severalnines

PCI compliance for MySQL & MariaDB with ClusterControl

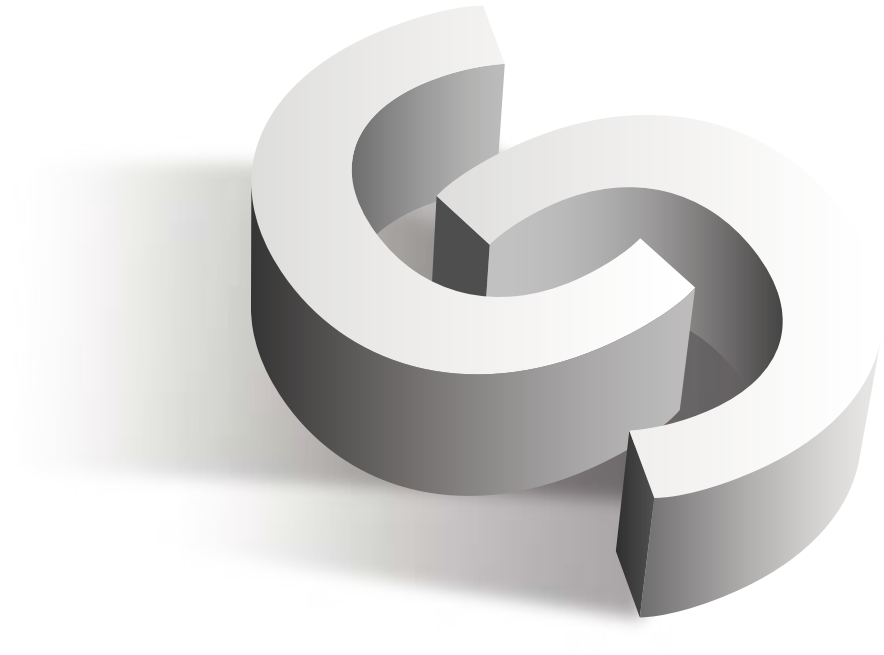


Table of Contents

Introduction

The Payment Card Industry Data Security Standard (PCI-DSS) is a set of technical and operational requirements defined by the PCI Security Standards Council (PCI SSC) to protect cardholder data. The standards apply to all entities that store, process or transmit cardholder data – with requirements for software developers and manufacturers of applications and devices used in those transactions. PCI data that resides in a MySQL or MariaDB database must adhere to these requirements, and database administrators must follow best practices to ensure the data is secured and compliant.

Nowadays the security of payment cards transactions is ensured on several fronts: on the cardholder side, EMV cards are protecting in-person transactions with a secure chip, and 3D Secure is authenticating online purchases. PCI-DSS comes into play as soon as the transaction data leaves the cardholder and, through the merchant, is stored on back-end systems until being processed by the banks.

The combination of those technologies and methods makes transactions more secure (as long as they are properly implemented). The technologies used by the cardholder are, by design, the simplest to use: physical ownership and a PIN to protect the card and a 2nd factor authentication for 3D Secure.

Most of the difficulty ends up in the parts covered by PCI-DSS, which covers a long chain of processing entities, from the merchant itself to the card processor then to the banks exchanging the money electronically. To keep track of the transaction, the cardholder data will have to be processed and stored several times. The major card brands currently demand that all those actors act in compliance with PCI-DSS during all steps, under the threat of fines. That compliance is not just a one-time operation or an off-the-shelf product. The environment where it will apply will often be heterogeneous (made up of various networked systems). It first has to be precisely defined, so that the limit of the network is known, and then, every interconnected devices have to be set up according to the requirements. This will cover network devices such as switches and routers, up to workstations and servers.

Severalnines ClusterControl can help to cover several aspects of compliance. PCI-DSS itself is a rather pragmatic approach to securing payment card data. For the sensitive data (CVV, PIN, ...), it is simple: they must never ever be stored. For other data, such as the card number or cardholder name, they can be kept as long as careful conditions are met. Such data is commonly stored into a database for processing, and that database must then be properly secured against intentional (or even unintentional) leakages, using techniques such as strong cryptography, access control and audit trails.

Scope of the document

This document focuses on PCI-DSS requirements for a database back-end managed by ClusterControl. It is not about PA-DSS, which is a standard that can be used by application vendors to secure their payment application. A PCI-DSS environment does not require using PA-DSS applications, and using PA-DSS applications by itself do not make an environment PCI-DSS compliant.

The version of PCI-DSS considered is 3.2, released in April 2016, currently in force at the time of writing. For clarity and consistency, the terminology used in this document will follow what is used in the standard itself.

PCI-DSS applies to every single component that can connect to the cardholder data environment (CDE). A database cluster managed by ClusterControl will only be one brick inside that CDE, usually a critical one, which will be the focus of both production constraints and PCI compliance. Typically, it will be located in an isolated network segment to reduce the scope of the compliance. That has some consequences, such as Internet access, which, if available, will be restricted and monitored.

One important part of PCI, easily overlooked when starting the compliance process, is that it concerns itself heavily with ***provability***. Doing things right is, of course, most important but PCI compliance means all requirements must be ***proven*** to be in place with procedures and evidence showing it.

> PCI compliance means all requirements must be ***proven*** to be in place with procedures and evidence showing it.

Auditors will look at how the CDE has been secured and ask for different pieces of evidence such as configuration files, screenshots of the applications, and emails. This means that even a perfectly secure CDE can fail compliance if the proper evidence trail is not available. The easier it is to provide evidence the better. In the case of a traditional database, and specifically a database cluster, this can be tricky. This is where ClusterControl will be helpful, both in securing the data, and providing the evidence that it is secure.

One of the recommended ways to achieve compliance is separation of duties; where implementing security and auditing security are tasks done by separate people. ClusterControl can help by making sure all actions are logged and these logs are then sent to a remote server.

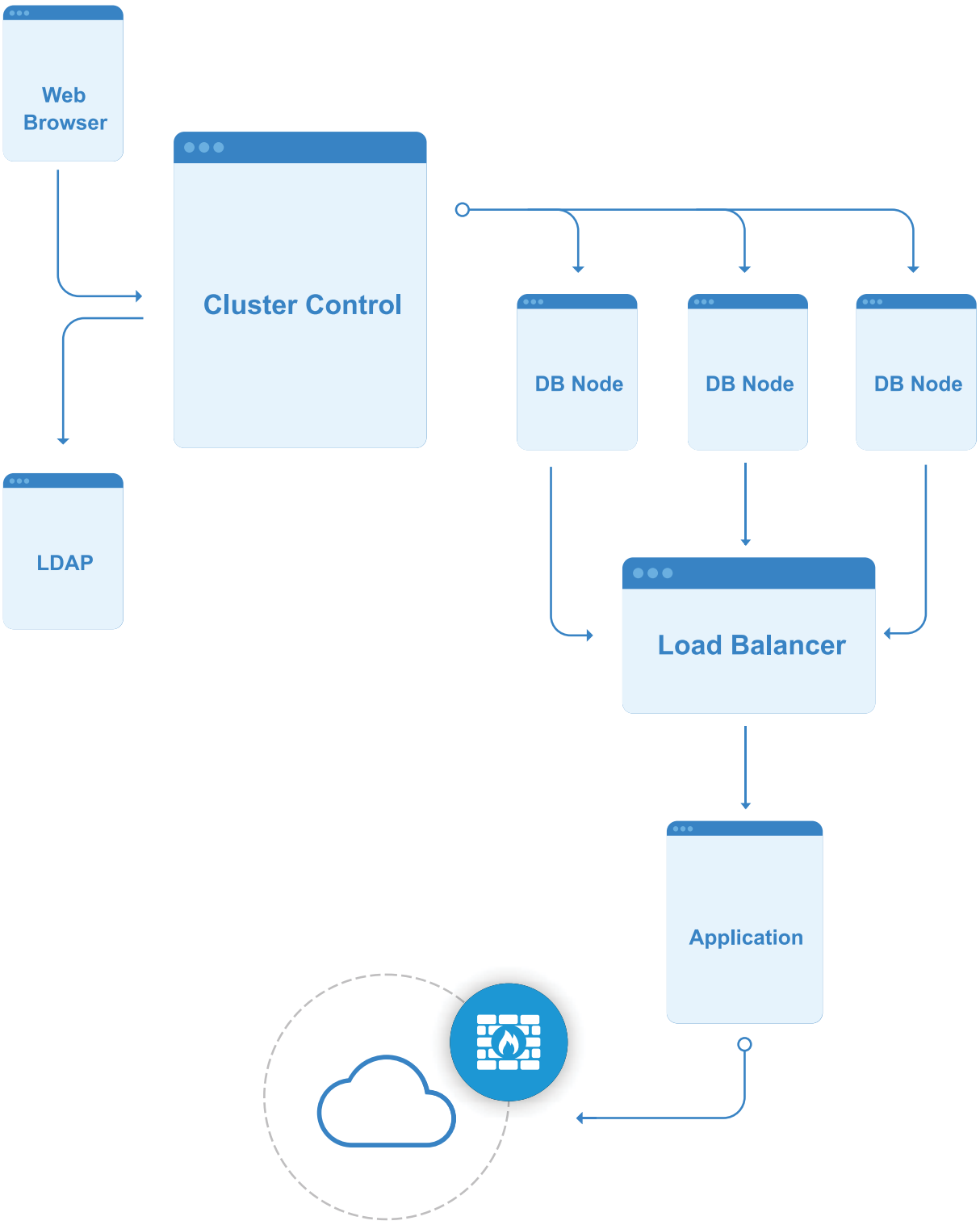
Auditors will check by various methods such as reading documentations, interviewing people, looking at logs, checking configurations, testing network connections, and more. With this process it is necessary to review a sampling of the complete environment as auditors can only work so fast and complex environments make it impossible to review each and every component in depth. However, weaknesses can be very obvious when database administrators have failed to give enough attention to some areas and those areas are left improperly secured. Sometimes it becomes just

easier to avoid the appearance of a weakness rather than having to spend time explain why it isn't one.[1]

With ClusterControl, the whole database environment can easily be viewed as a single entity. It becomes less necessary to do assessments for each cluster node separately as ClusterControl gives a single unified view of the whole cluster, showing the state and configuration of the database or load balancers. A complete check is possible in a single connection to its web interface.

[1] A common example is SHA-1: now forbidden to be used in some cases like TLS certificates, it is still considered secure as a MAC (Message Authentication Code) algorithm in SSH.

PCI Data Security Standard - The Requirements



Example of a secure database setup with ClusterControl

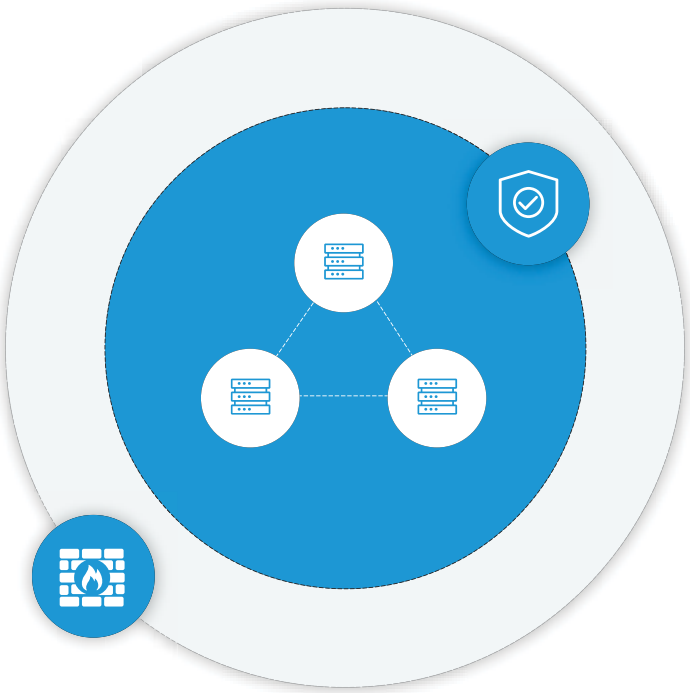
Build and maintain a secure network and systems

Requirement 1 - Install and maintain a firewall configuration to protect cardholder data

The network must be secured. This is to ensure that different flows of data are either allowed or blocked. At the core this means setting up firewalls and maintaining complete documentation of the network. The checks will be to ensure that no unneeded port is kept open, and that either incoming or outgoing connections are kept to what is absolutely needed.[1]

> **Req. 1.1** "Establish and implement firewall and router configuration standards"

A well-defined, restricted set of open ports must be defined when installing ClusterControl, as well as the database cluster it will manage. The allowed connections can be restricted only to the addresses of the nodes. [2]



Firewalling the systems

> **Req 1.2** "Build firewall and router configurations that restrict connections between untrusted networks and any system components in the cardholder data environment."

> **Req 1.3** "Prohibit direct public access between the Internet and any system component in the Cardholder Data Environment."

[1] ClusterControl makes use of Google Analytics, this can be disabled by the administrator

[2] A list of ports can be found here https://support.severalnines.com/hc/en-us/articles/212426063-Firewall-ports-

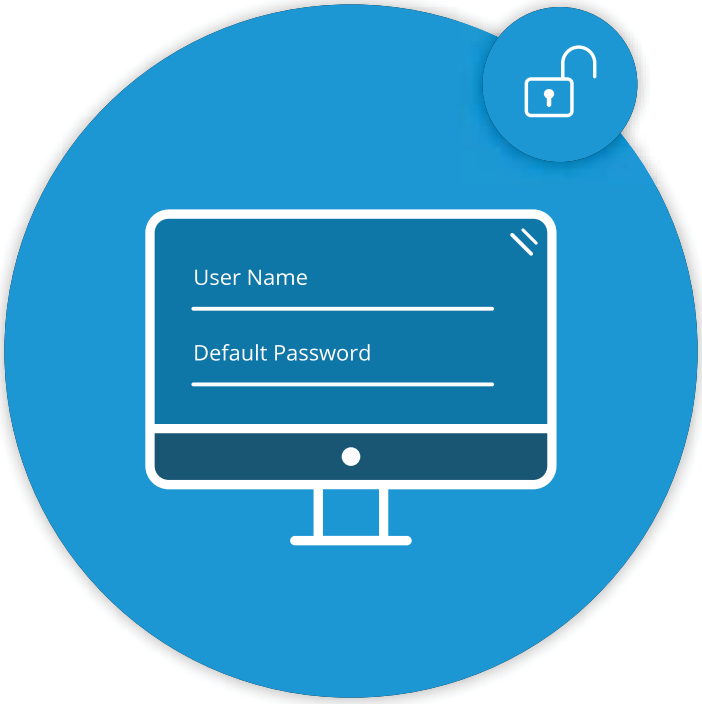
Access to and from the Internet must be restricted to the minimum needed, both for incoming and outgoing connections. While this is not directly under the responsibility of ClusterControl, a wrong configuration can impact its operation: too tight, it won't be able to open the needed connections to the database nodes and other elements; and too open, it will become a security risk.

Requirement 2 - Do not use vendor-supplied defaults for system passwords and other security parameters

Any vendor-supplied default for security parameters must be removed. For example, default passwords, passphrases, keys…

> **Req 2.1** "Always change vendor-supplied defaults and remove or disable unnecessary default accounts **before** installing a system on the network."

For MySQL and MariaDB the default root password is empty and care must be taken to set it to a proper value. In general, tools that ask to set passwords or generate keys upon installation guarantee this requirement is met. In the case of a cluster the only connection initially allowed by the firewall will be restricted to the ClusterControl node. Application and other systems can be allowed once the database has been secured.



Default passwords are a security risk

The installation of ClusterControl itself will ensure its application passwords are properly set. ClusterControl should be configured to access its user accounts from an LDAP directory, which will take care of ensuring strong passwords, rotation, and all other required policies. For example, they could include:

- Strong one-way hash, e.g. SHA512
- Minimum length
- Complexity

- History (one cannot reuse the last x passwords)
- Control of invalid login attempts (e.g., locking out a user for a certain period of time, or until unlocked by the administrator)

> **Req 2.2** "Develop configuration standards for all system components. Assure that these standards address all known security vulnerabilities and are consistent with industry-accepted system hardening standards."

2.2.1 requires that servers, physical or virtual, must have only one primary function: this fits in with how ClusterControl sees deployments, with each physical host or VM only hosting one database or proxy node.

2.2.3 requires that additional security be enabled, ClusterControl can help ensure that TLSv1.0 and below are not enabled to connect to the database nodes.

2.2.4 requires configuring security parameters to prevent misuse. ClusterControl can have application accounts passwords (such as those used for database access) stored in separate configuration files. This allows to have the configuration files kept e.g. in a central repository without including potentially sensitive elements.

> **Req 2.3** "Encrypt all non-console administrative access using strong cryptography."

Strong cryptography is required to protect every access. That means securing the httpd server it uses for the frontend GUI and ensuring that connections from applications to the database are using TLS as well. The ClusterControl frontend can be configured to only be accessed by HTTPS. ClusterControl can be used to configure the servers on the database nodes so they accept TLS connections, database users can then be created with 'REQUIRE SSL' to enforce the use of encrypted connections from the application to the databases.

Protect Cardholder Data

Requirement 3 - Protect Stored Cardholder Data

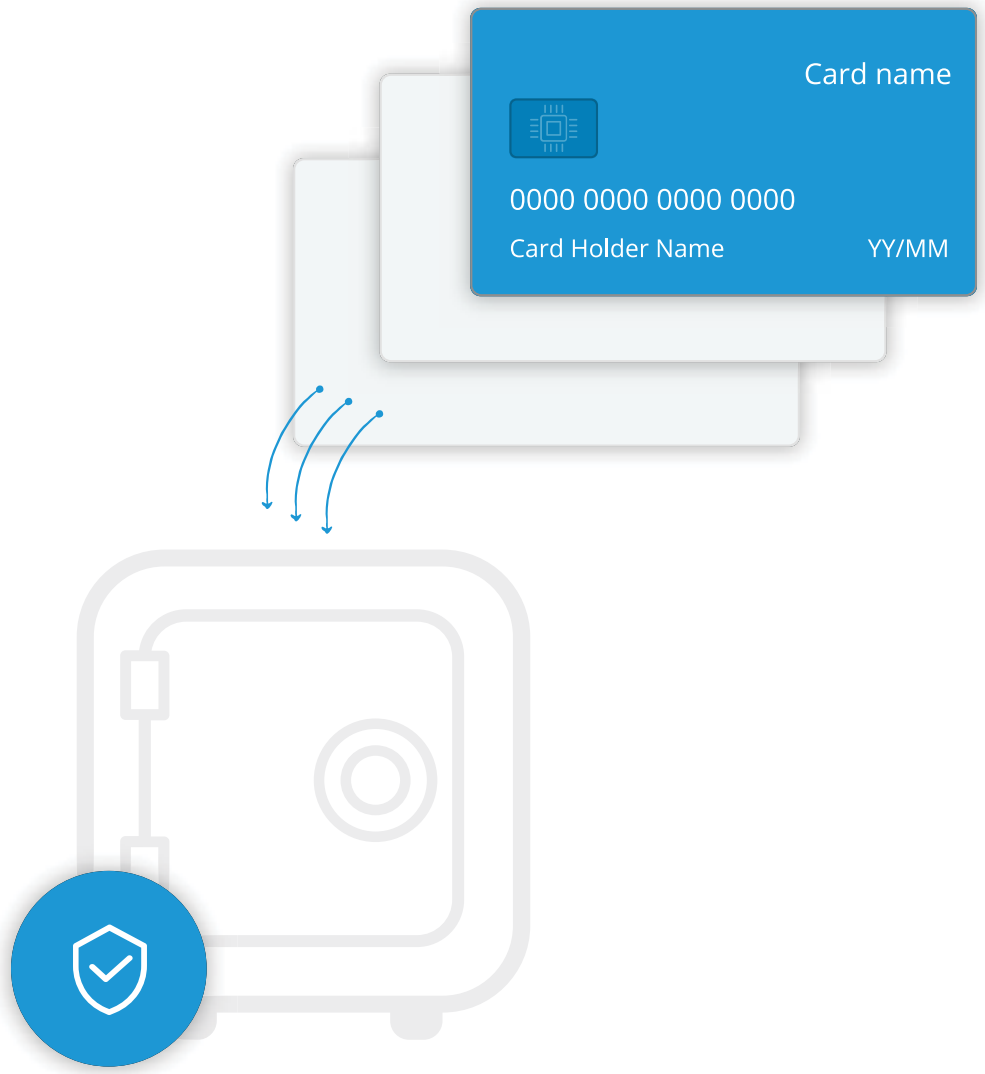
> **Req 3.2** "Do not store sensitive authentication data after authorization (even if encrypted). If sensitive authentication data is received, render all data unrecoverable upon completion of the authorization process"

This requirement gets to the core... protecting the cardholder data. In many cases, cardholder data is transient, and will not be stored at all. Authentication data, such as the PIN or the CVV2, must never be stored. The PAN, when it must be stored, must be either masked (by removing a sufficient number of its digits) or encrypted.

ClusterControl can help both directly and indirectly.

Directly by helping to understand how the database is structured, and what content may be potentially sensitive: an auditor can have a first look at how data is organized to check if anything is potentially sensitive.

Ultimately, the encryption of the data can be done at different levels, application, database, file system, disk, each covering a part of the the requirements.



Card information must be kept safe

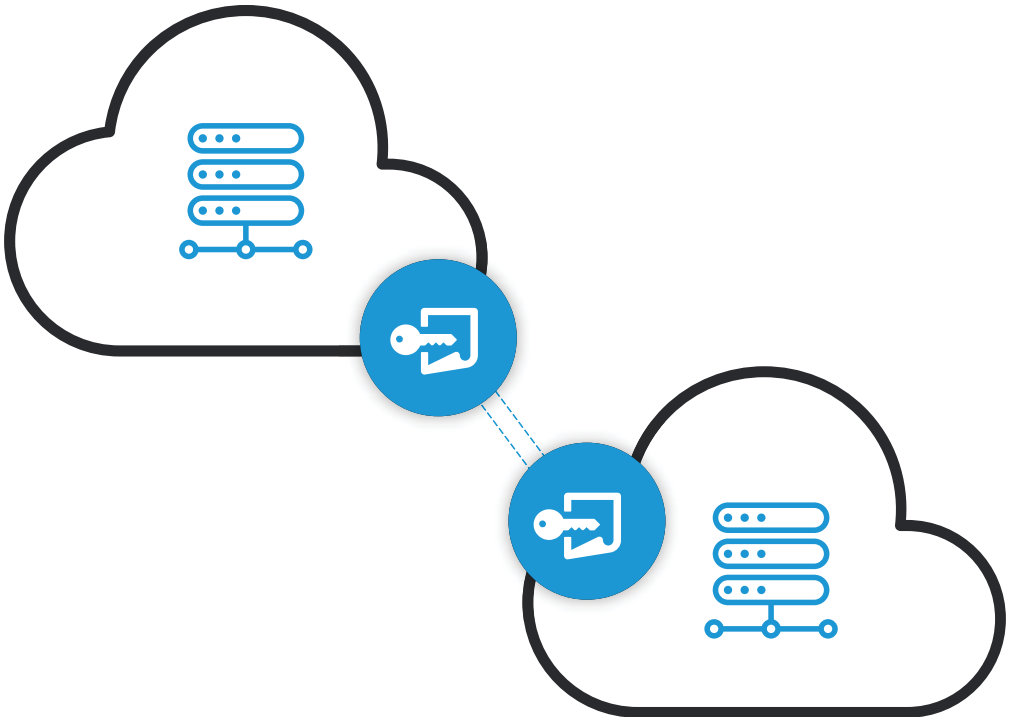
Indirectly to ensure that requests are not inadvertently stored in logs when they contain cleartext card data (such as when slow queries logs is enabled). Since compliance impose auditing database access, the risk of inadvertently saving sensitive data in a log file is always present.

Requirement 4 - Encrypt transmission of cardholder data across open, public networks

Nowadays, as soon as there are two separate locations for systems, public networks are involved to connect them. And this becomes even more obvious when using systems in a cloud.

> **Req 4.1** "Use strong cryptography and security protocols to safeguard sensitive cardholder data during transmission over open, public networks."

ClusterControl can help set up TLS between database nodes (replication traffic between databases) and to access nodes (database access between client applications and databases). This ensures that all requests to the database are encrypted.



Card data can only be transferred encrypted

Maintain a vulnerability management programme

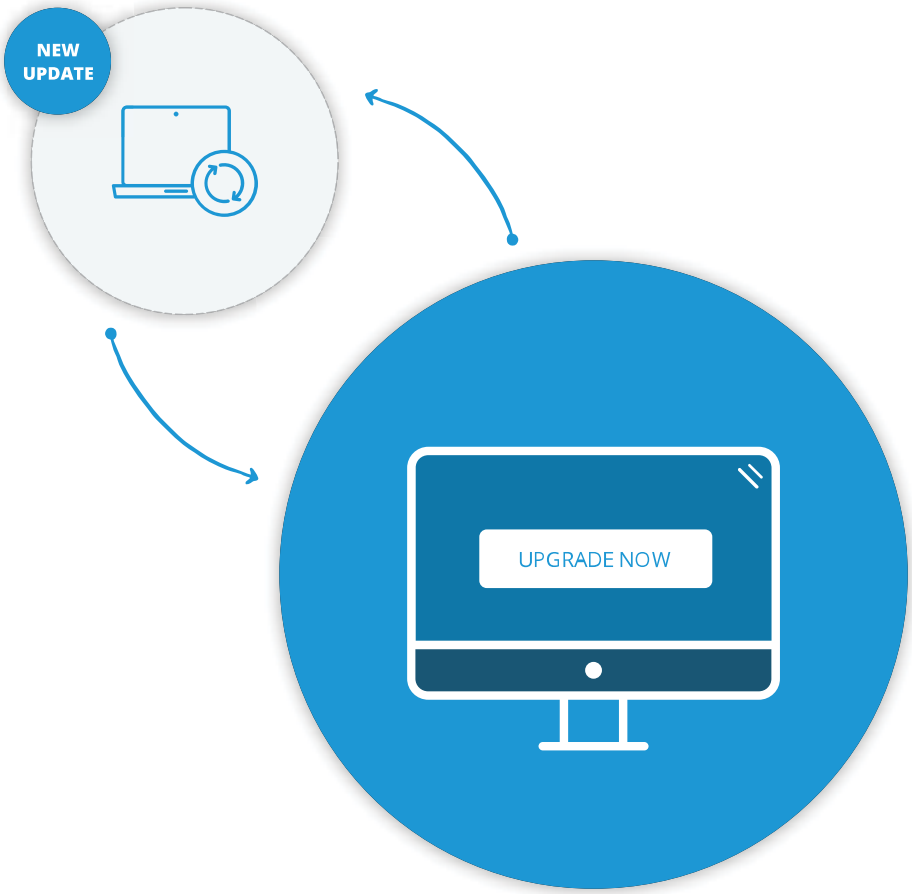
Requirement 5 - Protect all systems against malware and regularly update anti-virus software or programmes

It can only affect ClusterControl indirectly: antivirus software has to be installed, and auditors have started asking for them on Unix/Linux systems as well.

That means that care must be taken that it will not affect the performance of the database, or create false positives on certain file formats. If an antivirus scan is scheduled on the nodes, it must exclude the database files.

Requirement 6 - Develop and maintain secure systems and applications

Software vulnerabilities must be tracked and any that impact security must be patched quickly. In general this means that the patching schedule of PCI systems is faster than in the rest of the industry. Patches for critical vulnerabilities must be applied within one month of their availability, and within 3 months for non-critical ones.



Keeping systems up-to-date is a must

> **Req 6.1** "Establish a process to identify security vulnerabilities, using reputable outside sources for security vulnerability information, and assign a risk ranking (for example, as "high," "medium," or "low") to newly discovered security vulnerabilities."

> **Req 6.2** "Ensure that all system components and software are protected from known vulnerabilities by installing applicable vendor- supplied security patches. Install critical security patches within one month of release."

ClusterControl greatly eases the upgrade process of the database cluster. Currently, the rule is that any CVE with a score above 7 must be patched one month after the fix is available. With the complexity of modern software, and the scrutiny it attracts, this means in practice that systems in a PCI-DSS environment are patched often. To keep a database cluster reliable and available means having a clear process to perform a rolling upgrade makes it manageable and easier to live with: it minimizes the impact on the applications using it, to make those upgrade transparent to end users.

ClusterControl also allows to quickly check the currently running versions of the database server on each node: that makes it simpler to show to auditors that all versions are up to date. An Operational Report will show any software packages that can be upgraded on all managed database servers. It is a convenient way to view what versions of packages are installed, and if all of them are up-to-date.

> **Req 6.3** "Develop internal and external software applications (including web-based administrative access to applications) securely."

Applications must be developed securely. Any development-specific account must be removed when the application goes to production, as those have permissions for debugging. ClusterControl will allow to check that applications accounts are limited when running on the production environment.

> **Req 6.4** "Follow change control processes and procedures for all changes to system components."

Separation of the development/test environments and production environments means that a full ClusterControl stack should be installed on dedicated staging servers that will be used to check new deployments.

Preparing rollbacks and change also apply to ClusterControl (6.4.5.4, 6.4.6). Using its database backups is one way it can help as it will fit in roll back plan to minimize unavailability, should anything unexpected happen.

> **Req 6.5** "Address common coding vulnerabilities in software-development processes."

Though ClusterControl manages the database backend to keep it up and running, it doesn't actually handle any cardholder information. And while this particular requirement does not therefore apply to ClusterControl (nor can it help users with this requirement), its development process can absolutely gain from the Best Practices that it outlines.

ClusterControl builds on standard bricks from Linux distributions: Apache, PHP, ssh. This allows the experienced Severalnines developer team to follow the Best Practices as applied to those well-known elements, and to develop internal procedures to reduce security risks. In addition, the Severalnines support and development teams are in close contact with customer and user requests tagged as security requests, which can be analyzed and dealt with in a short timeframe.

Implement Strong Access Control Measures

Requirement 7 - Restrict access to cardholder data by business need to know

Strong access controls are required with a deny-all default and only business need-to-know permissions should be granted.

One important aspect of this is that system and database administrators should not be able to access the content of the databases while the users of those databases should only be limited to what they must access.

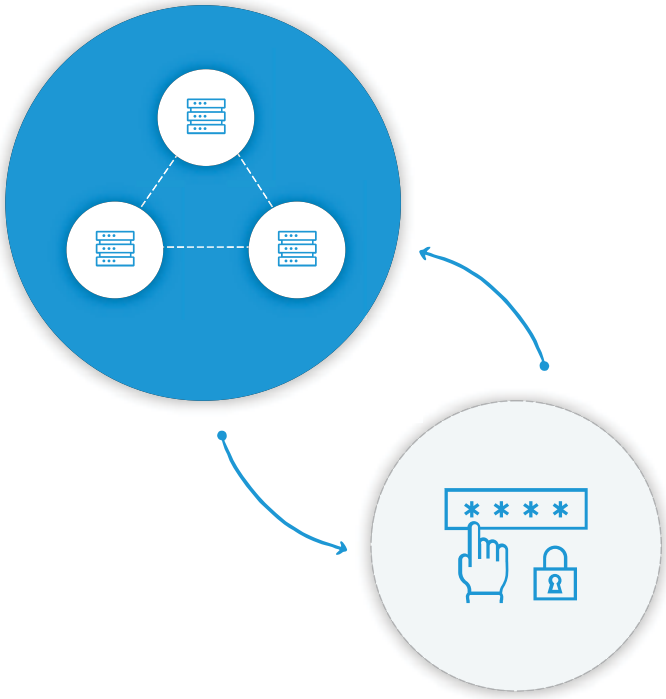
This is something that has to be proven both via the permission settings and the logs and these traces of the access must show business reasons for all accesses.

> **Req 7.1** "Limit access to system components and cardholder data to only those individuals whose job requires such access."

A common organization will revolve around a handful of different types of users:

- **root:** all-powerful, and therefore usually restricted to localhost connections, and for emergency use
- **Administrators:** personal accounts of users that are allowed to manage the database, but not access its content
- **Developers:** personal accounts for users that define the structure of the database
- **Application Accounts:** used only by applications and not by people, allowed to access and modify the content of only the databases the application needs to access

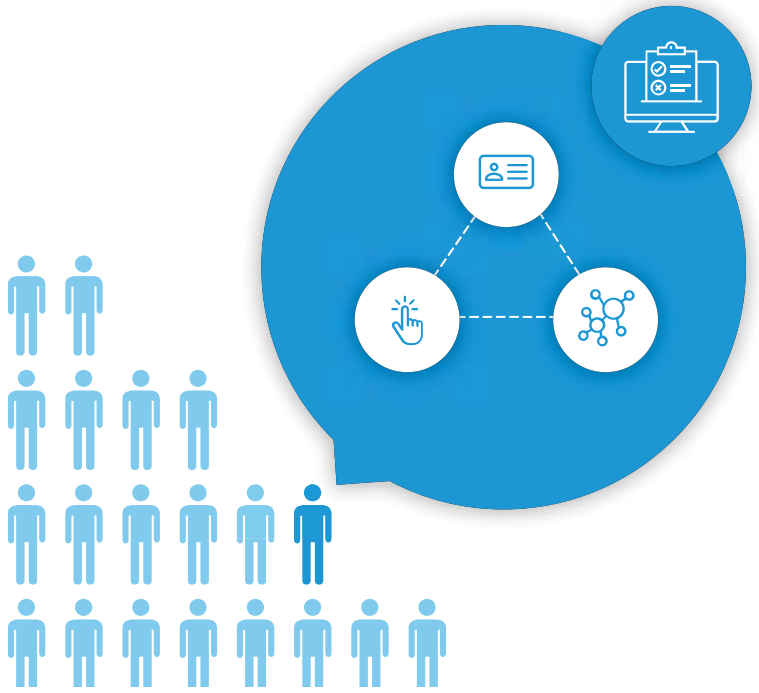
ClusterControl allows to check on SQL users and their access permissions at a glance for each node of the cluster to ensure that they are all limited to their exact needs including that the addresses are allowed to connect, which databases can be used, and the operations allowed on them.



Restricted access to the database, with only the required functionality required and not more

Requirement 8 - Identify and authenticate access to system components

There are a lot of constraints on authentication systems and procedures and their complexity is generally on the rise, which can be a technical challenge. The easiest way to deal with it is to connect it to an LDAP server, which will have its own user management procedures already in place. Examples of these include user lockouts after repeated failed attempts, locking of inactive accounts, and so on. However, ClusterControl has to do some tasks.



Each separate user has a unique personal identifier

> **Req 8.1.1** "Assign all users a unique ID before allowing them to access system components or cardholder data."

ClusterControl shows all existing database users on each node, allowing to prove that this requirement is properly met by listing all application and personal accounts at a glance.

> **Req 8.1.2** "Control addition, deletion, and modification of user IDs, credentials, and other identifier objects"

ClusterControl can keep track of its users' connections and operations to provide the required accountability for database access.

> **Req 8.5** "Do not use group, shared, or generic IDs, passwords, or other authentication methods,"

ClusterControl's listing of the SQL users allows to check that they are configured properly, particularly for the "no shared ID" rule.

> **Req 8.7** "All access to any database containing cardholder data (including access by applications, administrators, and all other users) is restricted."

Access to the database must be restricted to preset operations, except for DBAs. The list of SQL users shows what their permissions are on databases.

Requirement 9 - Restrict physical access to cardholder data

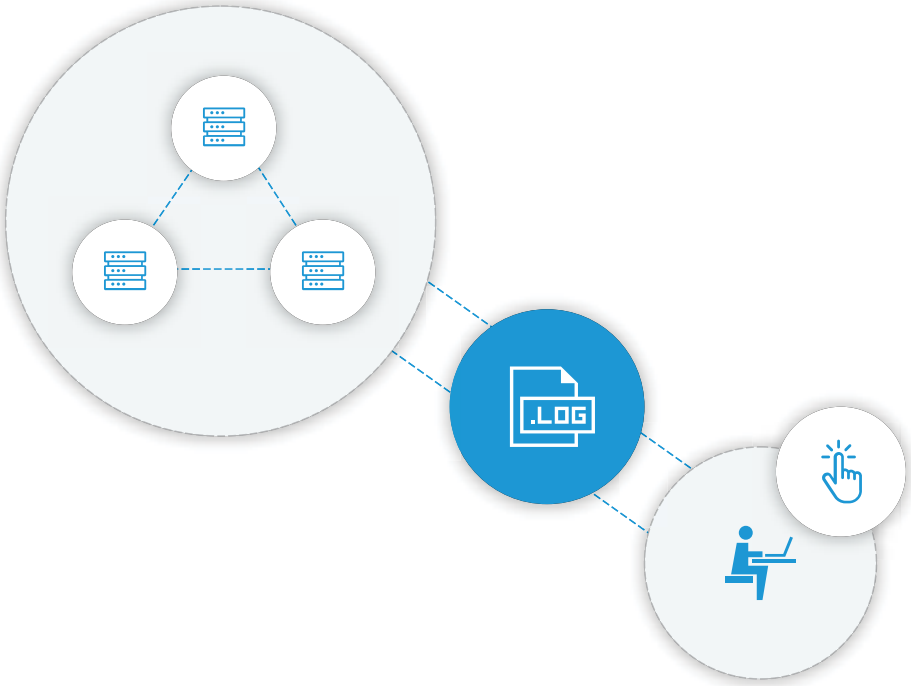
Not applicable for ClusterControl, as it cannot help to protect against physical access.

Regularly monitor and test networks

Requirement 10 - Track and monitor all access to network resources and cardholder data

Every action must be logged in a secure way. Modification of the configuration of the audit software itself must also be logged. This is where SQL databases are traditionally poor, since they mostly rely on applicative users. The application itself connects to the database and the access control is more or less expected to be done on the application side. However, manual manipulations on the databases are still common, if hopefully not frequently needed. One added difficulty is that requests can contains sensitive data, that must not be itself logged (secrets, PAN, ...). Typically, log transmission and storage is not encrypted, and the people checking on them are not the same as those working directly with the production data.

Both aspects can mean failing compliance: missing logs and sensitive data in logs. This is a difficult balance to achieve.



Connections to the database cluster will be logged

> **Req 10.1** "Implement audit trails to link all access to system components to each individual user."

ClusterControl helps to maintain an audit trail via its interface, and sends logs to a remote syslog server.

Requirement 11 - Regularly test security systems and processes

The security of systems must be tested. This includes vulnerability scans, penetration testing, and checks on the methodology used.

> **Req 11.2** "Run internal and external network vulnerability scans at least quarterly and after any significant change in the network"

ClusterControl is both directly and indirectly involved. Indirectly, because it sits at the top of a many elements, database, web server, secure shell, all of which can have known vulnerabilities. Directly, because it too can have CVE. It can be of help and it must not be a hindrance.

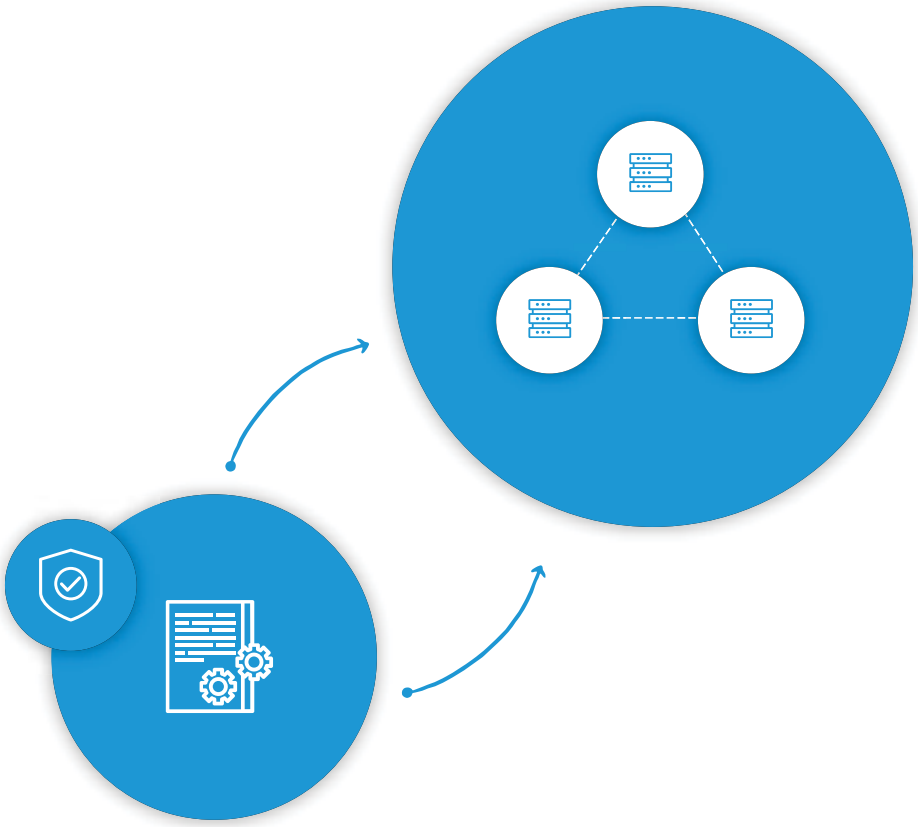
The version of databases can be determined and known issues reported. And ClusterControl itself must work with vendor-supported, up-to-date versions of the tools.

Maintain an Information Security Policy

Requirement 12 - Maintain an information security policy for all personnel

This is almost redundant, as the previous requirements already insists on having procedures, how the PCI auditors must check them, and how they are applied. It aims for the global security policy, and how all personnel must follow it.

Once again, this requirement is not only about writing the policy, but also proving it has been communicated to the persons that must follow it.



All database administration procedures need to be documented

> **Req 12.1** "Establish, publish, maintain, and disseminate a security policy."

ClusterControl helps by streamlining the use of the database cluster, it is not negligible as it allows to reduce the information load to be processed by people in a complex environment.

A smooth integration of CC itself is helping. Providing an easy to use interface saves time and reduces stress in case of a system failure.

Since there is, almost by definition, a lot of procedures to learn and apply, anything that makes their understanding and application easier is welcome. It means time saved, frustration avoided, and ultimately a better user commitment to the policies.

Appendix A2

> **A2.2** "Entities with existing implementations (other than as allowed in A2.1) that use SSL and/or early TLS must have a formal Risk Mitigation and Migration Plan in place."

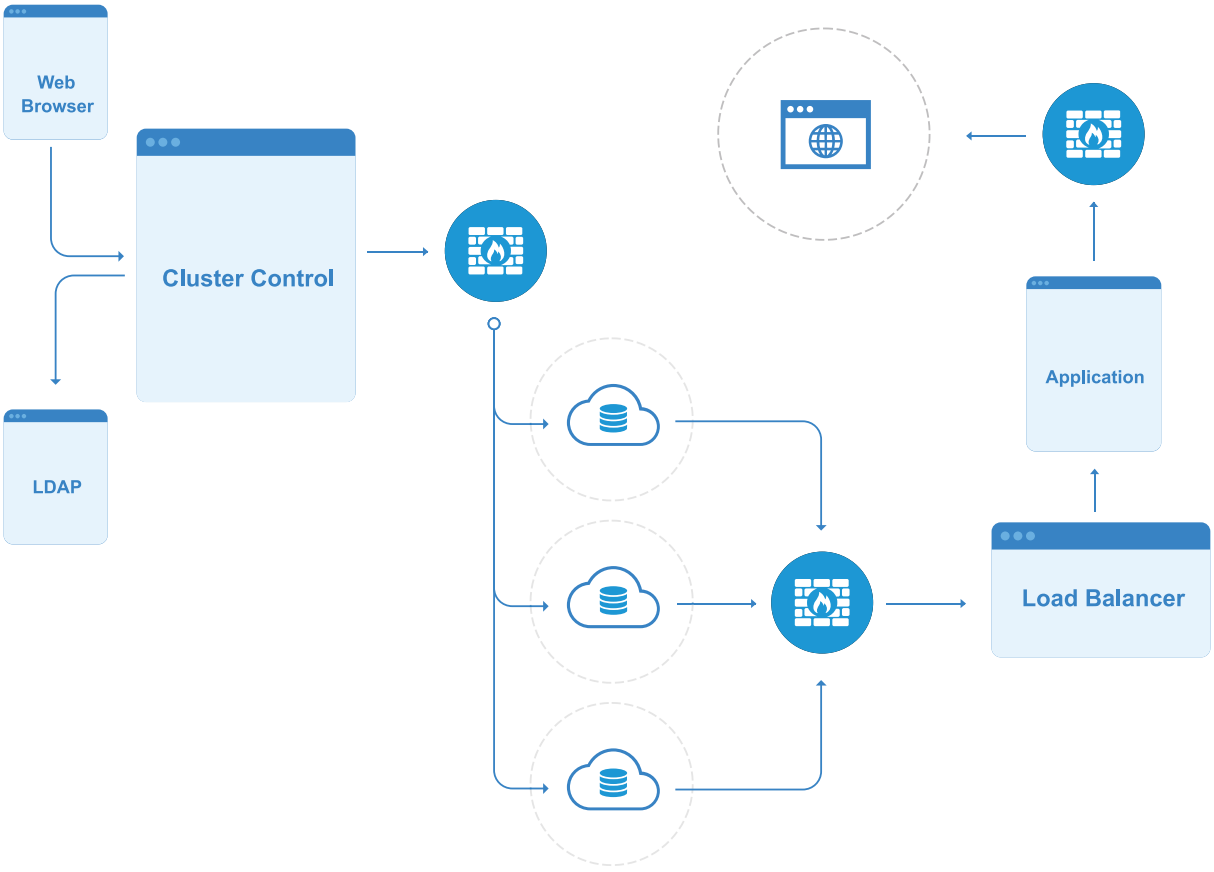
This one is relevant, as it covers the end-of-life of older SSL/TLS implementations (anything below and including TLSv1.0) which are not secure and must now be disabled before June 30, 2018.

Onsite vs Cloud usage

All points are similarly applicable whether the database is used onsite or in a public cloud. However, in the latter case, greater care will be needed since all elements are public (in the sense that they are potentially shared with others) there is little room left for compensating controls to reach compliance.

For example, there will not be an internal network that can be considered secure behind a perimeter firewall, nor the possibility to show how an individual server is physically secure in a locked cabinet.

Each node could be considered as an independent CDE, interconnected to others over public networks, it becomes even more important to streamline first their installation, then their management, and ultimately, that everything is provably in order.



Cloud deployments - similar, yet different

Summing it up...

Payment card security is not a fixed goal that can be reached by implementing a limited set of well-defined steps. It is, however, a worthy goal and one that stays elusive and is not easily reached (as shown by well-publicized data thefts from major companies).

It is meant to be a sustained effort from all parties involved. PCI-DSS itself is not set in stone, the PCI Council upgrades the standard yearly... keeping track of the evolution of technology, such as when encryption algorithms are not considered secure anymore and must be removed.

PCI-DSS implementers, those on the front line of card security, must not only keep track of those changes and adjust their environments accordingly. They must also make sure, year on year, that all requirements are satisfied, stay that way, and evidence is kept to prove it.

In practice, when starting from scratch, this can easily mean a spiraling amount of time spent on it, and ending up overwhelmed when using software that was not designed for compliance, often because it long predates PCI itself, as is the case for most database systems in use today.

That is why, as often as possible, reliable tools must be chosen to help with that compliance, easing out the crucial parts. Each time the compliance for one requirement can be shown to be implemented, working, and logged accordingly, time will be saved. If well-designed, it will only require regular software upgrades, a yearly review and a moderate amount of tweaking to follow the standard's evolution.

The future

As stated before, PCI-DSS evolves often, so that both tools and procedures must follow to keep environments compliant. ClusterControl will continue improving to keep making the management of clustered databases easier, as well as, their security. To reflect that, this white paper will also be updated when changes are required by the standard or because new features are introduced to help respect it.

And of course, each environment is different, with its own specific needs and constraints, and audits adjust to those difference, not following always the exact same pattern.

In other words, feedback is welcome, to continue building on past experiences.

About the Authors

Laurent Blume, Unix systems engineer

Laurent's career in IT started in 2000, his work since evolved from POS terminals for a jewelry store chain to infrastructure servers in a government aerospace R&D organization, even touching supercomputers. One constant throughout was the increasing need for security.

For the past 6 years, he has been in charge of first implementing, then keeping up with the PCI-DSS compliance of critical transnational payment authorization systems. Its implementation for databases has been an essential part of the task. For the last few years, it has expanded to the design and productization of MariaDB cluster backends for mobile contactless payments.

Vinay Joosery, CEO & Co-Founder, Severalnines

Vinay is a passionate advocate and builder of concepts and business around Big Data computing infrastructures.

Prior to co-founding Severalnines, Vinay held the post of Vice-President EMEA at Pentaho Corporation - the Open Source BI leader. He has also held senior management roles at MySQL / Sun Microsystems / Oracle, where he headed the Global MySQL Telecoms Unit, and built the business around MySQL's High Availability and Clustering product lines. Prior to that, Vinay served as Director of Sales & Marketing at Ericsson Alzato, an Ericsson-owned venture focused on large scale real-time databases.

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 it's automation functionality that let's 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 o ces in Singapore, Japan and the United States. To see who is using Severalnines today visit:

https://www.severalnines.com/company

Deploy

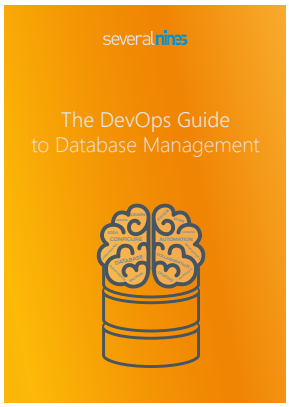
Manage

Monitor

Scale

Related Resources from Severalnines

Whitepapers



The DevOps Guide to Database Management

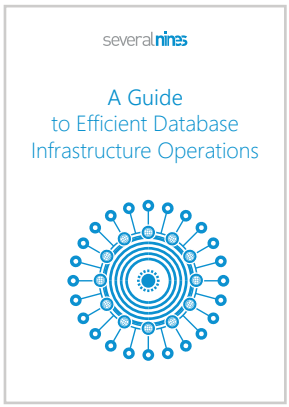
Relational databases are not very flexible by nature, while DevOps is all about flexibility. This creates many challenges that need to be overcome. This white paper discusses three core challenges faced by DevOps when it comes to managing databases. It also discusses how Severalnines ClusterControl can be used to address these challenges.

Download here

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 here



A Guide to Efficient Database Infrastructure Operations

Taking control of their data is every company's number one job. Database operations encompass a number of functions, including the initial deployment of a solution, configuration management, performance monitoring, SLA management, backups, patches, version upgrades and scaling. In this white paper, we will discuss the operational aspects of running database infrastructures, and how companies can make these more efficient.

Download here

severalnines

Deploy

Manage

Monitor

Scale