

Developer Cookbook

- [How to start a replication in MYSQL](#)
- [How to generate a self encrypted SSL file for non wildcard domains with certbot](#)
- [Τι είναι το proxysql και πως μπορεί να χρησιμοποιηθεί](#)
- [Τι είναι το redis και πως μπορεί να χρησιμοποιηθεί](#)
- [Τι να προσέξουμε σε alter table σε παραγωγική βάση | MYSQL](#)
- [How to do safely ALTER TABLE for changes in column types with pt-online-change-schema](#)
- [How to swap a replication and a master database with each other efficiently](#)
- [Αντικατάσταση Google CAPTCHA με cloudflare turnstile](#)
- [Generate SSL certificate for Wildcard](#)
- [How to run mysql clone on a server](#)
- [emblem ergani disaster response for traffic](#)
- [Automate git pull securely](#)
- [Οδηγός setup για το cookie widget \(semantic cookie\)](#)

How to start a replication in MYSQL

1. Backup master data to a file

Για έλεγχο χώρου κάνουμε:

```
1. df -h
```

Και έπειτα εφόσον έχουμε αρκετό χώρο κάνουμε

```
2. mysqldump -u root -p emplemdb --single-transaction --master-data=2 --routines --events --triggers | gzip > emplemdb.sql.gz
```

Το --master-data=2 περιέχει δεδομένα για το σημείο από το οποίο θα ξεκινήσει το replication να τρέχει

Το --single-transaction σε innodb tables κάνει το backup χωρίς να χρειαστεί να LOCK τα table

ΠΡΟΣΟΧΗ: ΝΑ ΓΙΝΕΤΑΙ ΕΛΕΓΧΟΣ ΟΤΙ ΥΠΑΡΧΕΙ ΧΩΡΟΣ ΣΤΟ ΣΕΡΒΕΡ ΠΡΙΝ ΓΙΝΕΙ ΤΟ BACKUP. ΑΝ ΕΧΕΙΣ 0% ΧΩΡΟ ΕΧΕΙΣ ΠΡΟΒΛΗΜΑ

2. Create replication user on master db

```
CREATE USER 'repl'@'%' IDENTIFIED BY 'StrongPassword';  
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';  
FLUSH PRIVILEGES;
```

3. Move the database you backed up from one server to another

ΠΡΟΣΟΧΗ: ΝΑ ΓΙΝΕΤΑΙ ΕΛΕΓΧΟΣ ΟΤΙ ΥΠΑΡΧΕΙ ΧΩΡΟΣ ΣΤΟ ΣΕΡΒΕΡ ΠΡΙΝ ΓΙΝΕΙ Η ΜΕΤΑΦΟΡΑ. ΑΝ ΕΧΕΙΣ 0% ΧΩΡΟ ΕΧΕΙΣ ΠΡΟΒΛΗΜΑ

```
scp dump.sql replica:/
```

Επίσης υπάρχει περίπτωση τα bin logs να πιάσουμε πολύ χώρο. Για διαγραφή των bin logs μπορεί να χρησιμοποιηθεί το

```
PURGE BINARY LOGS BEFORE NOW();
```

4 Import database to replication server

```
gunzip < emplemdb.sql.gz | mysql -u root -p emplemdb
```

Αυτή η διαδικασία μπορεί να πάρει αρκετή ώρα ανάλογα το σερβερ

5. Point replica to master

```
CHANGE REPLICATION SOURCE TO  
SOURCE_HOST='PRIMARY_IP',  
SOURCE_USER='repl',  
SOURCE_PASSWORD='StrongPassword',  
SOURCE_LOG_FILE='mysql-bin.000012',  
SOURCE_LOG_POS=45678;
```

Το SOURCE_LOG_FILE και το SOURCE_LOG_POS το βρίσκεις στο κάνωντας head -n 50 στο backup αρχείο που έχουμε πάρει.

TIPS & INFO

1. Δεν μπορείς να κάνεις στο replication alter table και να βρίσκεται σε διαφορετική κατάσταση απο τον παραγωγικό πίνακα. Δε το επιτρέπει η MYSQL ακόμα και αν μιλάμε για ίδιο είδος column type σε ίδιο ειδος διαφορετικού μεγέθους

How to generate a self encrypted SSL file for non wildcard domains with certbot

```
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
sudo certbot --apache
```

Για να μπορέσει να βρει το certification το certbot θα πρέπει να βρίσκεται στα sites-available το domain με τον παρακάτω τρόπο στο

<VirtualHost *:80>

ServerName emblem-ergani.gr

ServerAdmin webmaster@localhost

DocumentRoot /var/www/html/public

ErrorLog \${APACHE_LOG_DIR}/error.log

CustomLog \${APACHE_LOG_DIR}/access.log combined

ErrorDocument 404 /error-404.html

RewriteEngine on

RewriteCond %{SERVER_PORT} ^80\$

RewriteRule ^(.*)\$ https://%{SERVER_NAME}%{REQUEST_URI} [L,R]

RewriteCond %{HTTP_HOST} ^www\.(.*)\$ [NC]

RewriteRule ^(.*)\$ https://%1/\$1 [R=301,L]

</VirtualHost>

Το παραπάνω δείχνει ως root αρχείο το /var/www/html/public. Το certbot θα κοιτάξει το ServerName emblem-ergani.gr και θα δημιουργήσει το ssl conf.

Τέλος πρέπει να κάνεις

service apache2 restart

-- Θα πρέπει να δειχνει το A record του DNS στο server σου αλλιως δε θα μπορέσει να γίνει πιστοποίηση

?? ?????? ?? proxysql ??? ??? ???????? ?? ??????????????????

Το **ProxySQL** είναι ένα **high-performance proxy** για **MySQL** και **MariaDB**. Στην ουσία λειτουργεί ως ενδιάμεσος μεταξύ των εφαρμογών σου και των βάσεων δεδομένων σου και προσφέρει αρκετά πλεονεκτήματα. Ας το εξηγήσουμε αναλυτικά:

1?? ?? ??????

- Δέχεται **SQL queries** από την εφαρμογή σου και τα προωθεί στους κατάλληλους **MySQL servers**.
 - Μπορεί να λειτουργήσει ως **load balancer**, δηλαδή να μοιράζει τα queries ανάμεσα σε πολλούς servers για καλύτερη απόδοση.
 - Υποστηρίζει **replication-aware routing**, δηλαδή μπορεί να στείλει τα **read queries** στους replicas και τα **write queries** στον master.
 - Παρέχει **caching**, **connection pooling**, και **query routing** για καλύτερη απόδοση και λιγότερα connections στη βάση.
-

2?? ?????? ??????????????????

- **Load balancing:** Μοιράζει τα queries σε πολλούς servers για υψηλή διαθεσιμότητα και απόδοση.
 - **Query routing:** Καθορίζει σε ποιο server θα πάει κάθε query (π.χ., writes στον master, reads στους replicas).
 - **Connection pooling:** Κρατάει ανοιχτές συνδέσεις για να μην ανοίγουν/κλείνουν συνεχώς.
 - **Query caching:** Αποθηκεύει προσωρινά αποτελέσματα queries για να μειώσει το load στη βάση.
 - **Failover & High Availability:** Αν πέσει ένας server, τα queries μπορούν να συνεχίσουν να εκτελούνται στους υπόλοιπους.
-

3?? ??? ??????????????????

- Σε συστήματα με πολλούς **MySQL servers** (master + replicas).
- Όταν θέλεις **απλή κλιμάκωση** (scale reads) χωρίς να αλλάζεις την εφαρμογή.

- Για **monitoring** και **statistics** των queries, καθώς μπορεί να καταγράφει ποιες queries πάνε που.

=====

Στα συστήματά μας το χρησιμοποιούμε κυρίως για να έχουμε λιγότερα max connections στη βάση δεδομένων. Αυτό που κάνει είναι αντί να ανοίγει νέο connection για κάθε νέο αρχείο PHP που ανοίγει κρατάει ένα ζωντανό connection και το διατηρεί μεταξύ κλήσεων επιταχύνοντας πολύ τη διαδικασία σύνδεσης και δεν χρειάζεται επιπλέον connections (tcp ktcp)

Γιατί δε το χρησιμοποιούμε για load balancing?

Εμείς στα πρότζεκτ μας τώρα δεν κάνουμε transactions για τα CRUD actions αλλά τα κάνουμε απευθείας. Το proxysql για να μπορεί να διανείμει σωστά τα select και τα writes θα πρέπει να έχουμε transactions. Αν σε κάποιο προτζεκτ μελλοντικά τρέξουμε με transactions (που πρέπει) θα μπορούμε με proxysql να δηλώσουμε 2+ servers και με μία σύνδεση να αναλαμβάνει το proxysql το load balancing

?? ????? ?? redis ??? ??? ??????? ?? ?????????????????

Το redis είναι μια in-memory key-value βάση δεδομένων η οποία χρησιμοποιείτε κυρίως για διαδικασίες caching. Ως cache την χρησιμοποιούμε και εμείς σε σελίδες όπως το mydata.semantic.gr & ergani.emblem.gr. Δεδομένα που παραμένουν συχνά ίδια μπορούμε να τα βάλουμε στο Redis ώστε αντί να γίνεται επικοινωνία με τη βάση δεδομένων και με το σκληρό δίσκο κατεπέκταση να γίνεται το διάβασμα των δεδομένων απο τη μνήμη η οποία είναι πολύ πιο γρήγορη.

Επιπλέον αν έχουμε πολλαπλές εφαρμογές που μπορεί να βρίσκονται σε άλλους σερβερ και πρέπει να επικοινωνήσουμε με τον κεντρικό σέρβερ αν έχεις ένα redis μπορείς να κρατήσεις "ζεστά" τα δεδομένα που σε ενδιαφέρουν άμεσα χωρίς επικοινωνία με τη βάση.

Ειδικά σε σερβερ με πολύ διαθέσιμη RAM είναι αρκετά εύκολη επιλογή το Redis για ελάφρυνση του φόρτου της βάσης δεδομένων και επιτάχυνση της εφαρμογής. Παρακάτω είναι η τεχνοτροπία που χρησιμοποιείτε στο ergani.emblem.gr για να διαβάζουμε και να γράφουμε στη cache

```
public function getLimitedNotificationsByUser(int $iduser): array
{
    $redisClient = $this->getRedisClient();
    $cacheKey = "NotificationsModel:getLimitedNotificationsByUser:" . $iduser . ":" .
($ _SESSION['usertype'] ?? 'prod');
    $cacheTTL = 1800; // seconds
    $cachedData = $redisClient->get($cacheKey);

    if ($cachedData !== false) {
        return json_decode($cachedData, true);
    }

    $arr = $this->SelectSqlSlave("SELECT * FROM notifications WHERE iduser = ? ORDER BY
seen ASC, id DESC LIMIT 30 ", ['i', $iduser]);

    $redisClient->setex($cacheKey, $cacheTTL, json_encode($arr));

    return $arr;
}
```

Στο cache key βάζουμε όλες τις μεταβλητές της λειτουργίας ώστε να μην cacharουμε δεδομένα απο άλλους χρήστες και πάρουμε δεδομένα απο άλλους. Ελέγχουμε αρχικά αν υπάρχουν στην cache τα δεδομένα βάση κλειδιού και αν υπάρχουν επιστρέφουμε τα δεδομένα της cache. Διαφορετικά κάνουμε το select στη βάση και επιστρέφουμε τα δεδομένα της βάσης. Τέλος εφόσον πήραμε τα δεδομένα απο τη βάση τα κάνουμε cache.

Είναι σημαντικό σε περίπτωση που γίνουν αλλαγή στα δεδομένα να κάνουμε invalidate της cache

```
public function add(int $iduser, string $message): array
{
    $redisClient = $this->getRedisClient();
    $cacheKey = "NotificationsModel:getLimitedNotificationsByUser:" . $iduser . ":" .
($_SESSION['usertype'] ?? 'prod');
    $redisClient->del($cacheKey);

    return $this->otherSql("INSERT INTO notifications (type, iduser, seen, message) VALUES
(?,?,?,?)", ["siis", "Info", $iduser, 0, $message]);
}
```

?? ?? ?????????????? ?? alter table ?? ?????????????? ????? | MYSQL

Το alter table είναι μια διαδικασία που αν δεν γίνει προσεκτικά μπορεί να φέρει σε κατάσταση WRITE LOCK του table για πολύ ώρα με αποτέλεσμα να μη μπορεί να γράψει κανείς στη βάση. Θα πρέπει να προσέξουμε σε περίπτωση που κάνουμε alter table αν κάνουμε προσθήκη column ή αλλαγή column ή drop index κτλπ.

?????? column type ?? ????? column type ???? ?????????????? ????????

Είναι ασφαλές διαδικασία που γίνεται γρήγορα και δε θα επιφέρει lock για μεγάλο χρονικό διάστημα. Μπορεί να γίνει σε παραγωγική βάση

?????? column type ?? ????? column type ?? ?????????????? ??????. ??
VARCHAR -> TEXT

Οι αλλαγές σε διαφορετικό column type μπορούν να αργήσουν πάρα πολύ καθώς χρειάζεται να γίνει διαγραφή ολοκληρω του table και μεταφορά σε άλλο table όπου ο πίνακας επαναδημιουργείται. Θα πρέπει να γίνει προσεκτικός έλεγχος αν θέλουμε να κάνουμε τέτοια αλλαγή και να σκεφτούμε το χρόνο που μπορεί να διαρκέσει. Συνήθως τέτοια αλλαγές δε πρέπει να γίνουν ποτέ σε παραγωγικό πίνακα

????????? column

Μπορεί να γίνει με ασφάλεια σε παραγωγικό πίνακα

How to do safely ALTER TABLE for changes in column types with pt-online-change-schema

pt-online-schema-change will do alter tables without affecting at all the production table locks. It runs completely silent and is pretty safe.

If we want to check what exactly the pt-online-schema-change will do you can first do a dry run which will output what exactly will run

```
pt-online-schema-change --alter "MODIFY COLUMN xmlSent MEDIUMTEXT" --dry-run --print --alter-foreign-keys-method=auto D=emplemdb,t=novusXMLTrans -u root -p"!....!"
```

ΠΡΟΣΟΧΗ: ΝΑ ΓΙΝΕΤΑΙ ΕΛΕΓΧΟΣ ΟΤΙ ΥΠΑΡΧΕΙ ΧΩΡΟΣ ΣΤΟ ΣΕΡΒΕΡ. ΑΝ ΕΧΕΙΣ ΛΙΓΟ ΧΩΡΟ ΕΧΕΙΣ ΠΡΟΒΛΗΜΑ

To actually run it later on

```
pt-online-schema-change --alter "MODIFY COLUMN xmlSent MEDIUMTEXT" --execute --print --alter-foreign-keys-method=auto --progress percentage,1 D=emplemdb,t=novusXMLTrans -u root -p"!....!"
```

How to swap a replication and a master database with each other efficiently

On the master database

1. set the master database on read only

```
SET GLOBAL super_read_only = ON;  
SET GLOBAL read_only = ON;
```

On slave database

find the last bin log position and file to later make this the replica

```
SHOW MASTER STATUS;
```

```
mysql> SHOW MASTER STATUS;
```

File	Position	Binlog_Do_DB	Binlog_Ignore_DB	Executed_Gtid_Set
mysql-bin.011112	5895164	emplemdb		

```
1 row in set (0.00 sec)
```

Save this output to a text file

stop the slave status and check the read only status. It should stop being read only

```
STOP SLAVE;
```

```
RESET SLAVE ALL;
```

Now this is your new master database

Change the app credentials to look at the slave database

On the old server setup the slave status and start slave

```
CHANGE REPLICATION SOURCE TO  
SOURCE_HOST='PRIMARY_IP',
```

```
SOURCE_USER='repl',  
SOURCE_PASSWORD='StrongPassword',  
SOURCE_LOG_FILE='mysql-bin.000012',  
SOURCE_LOG_POS=45678;
```

```
start slave;
```

???????????? Google CAPTCHA ?? cloudflare turnstile

1. Σύνδεση στο <https://dash.cloudflare.com/>
2. Μπαίνουμε στο application security > turnstile
3. Κάνουμε click add widget
4. Προσθέτουμε hostnames
5. Αποθηκεύουμε το widget και κρατάμε το public/private key που μας δίνει γιατί θα τα χρησιμοποιήσουμε αργότερα

```
6. <body>
    <div id="captcha" data-size="invisible"></div>

    <button id="send">Send</button>

    <script
src="https://challenges.cloudflare.com/turnstile/v0/api.js?onload=onloadTurnstile&render=explicit" async defer></script>

    <script>
        let widgetId;

        // Cloudflare Turnstile (global callback referenced by the API script URL)
        window.turnstileWidgetId = null;
        window.turnstileToken = "";
        window.turnstileAutoSubmit = false;

        window.setSignInDisabled = function (disabled) {
            const btn = document.getElementById("sign-in");
            btn.disabled = !!disabled;
            btn.style.opacity = disabled ? "0.7" : "1";
            btn.style.cursor = disabled ? "not-allowed" : "pointer";
        };

        window.onloadTurnstile = function () {
            const render = () => {
```

```

const el = document.getElementById("cf-turnstile");

window.turnstileWidgetId = turnstile.render(el, {
  sitekey: "PUBLIC_KEY",
  // If you want to use ONLY your own "Είσοδος" button, keep this
invisible.

  size: "invisible",
  callback: function (token) {
    window.turnstileToken = token || "";
    window.setSignInDisabled(false);
    window.turnstileAutoSubmit = false;
    document.getElementById("sign-in")?.click();
  },
  "expired-callback": function () {
    window.turnstileToken = "";
    window.turnstileAutoSubmit = false;
    window.setSignInDisabled(false);
  },
  "error-callback": function () {
    window.turnstileToken = "";
    window.turnstileAutoSubmit = false;
    window.setSignInDisabled(false);
  },
});

if (document.readyState === "loading") {
  document.addEventListener("DOMContentLoaded", render);
} else {
  render();
}

document.getElementById("send").addEventListener("click", () => {
  event.preventDefault();

  let fields = {
    $email: document.getElementById("email"),
    $password: document.getElementById("password"),
  }

```

```

};

const turnstileToken =
  window.turnstileToken ||
  (window.turnstileWidgetId !== null && typeof turnstile !== "undefined"
    ? turnstile.getResponse(window.turnstileWidgetId)
    : "");

if (!turnstileToken) {
  // Disable your login button while captcha runs
  window.setSignInDisabled(true);
  window.turnstileAutoSubmit = true;

  // For invisible Turnstile, execute it now
  try {
    turnstile.execute(window.turnstileWidgetId);
  } catch (e) {
    window.setSignInDisabled(false);
    window.turnstileAutoSubmit = false;
  }
  return;
}

// Disable while the login request is in-flight (prevents double submit)
window.setSignInDisabled(true);

postData("/api.php?op=signIn", {
  email: fields.$email.value,
  password: fields.$password.value,
  turnstileToken: turnstileToken,
  "cf-turnstile-response": turnstileToken,
})
});
</script>
</body>

```

7. Βάζουμε στο script πάνω και όταν κάνουμε send έχουμε πλέον το token που θα σταλθεί στο backend
8. Στο backend μετά κάνουμε το παρακάτω για επιβεβαίωση οτι είναι ok να προχωρήσουμε

```
9. $token = $_POST['cf-turnstile-response']; // <-- Token from Turnstile
    $secret = "YOUR_SECRET_KEY";

    $verify = curl_init("https://challenges.cloudflare.com/turnstile/v0/siteverify");
    curl_setopt($verify, CURLOPT_POST, true);
    curl_setopt($verify, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($verify, CURLOPT_POSTFIELDS, [
        'secret' => $secret,
        'response' => $token
    ]);

    $response = json_decode(curl_exec($verify), true);

    if ($response["success"]) {
        echo "HUMAN VERIFIED!";
    } else {
        echo "FAILED";
    }
}
```

Μερικές επιπλέον επεξηγήσεις για το παραπάνω. Κάνουμε οπω είναι copy paste αυτο το κομματι

```
let widgetId;

// Cloudflare Turnstile (global callback referenced by the API script URL)
window.turnstileWidgetId = null;
window.turnstileToken = "";
window.turnstileAutoSubmit = false;

window.setSignInDisabled = function (disabled) {
    const btn = document.getElementById("sign-in");
    btn.disabled = !!disabled;
    btn.style.opacity = disabled ? "0.7" : "1";
    btn.style.cursor = disabled ? "not-allowed" : "pointer";
};

window.onloadTurnstile = function () {
    const render = () => {
        const el = document.getElementById("cf-turnstile");
```

```

window.turnstileWidgetId = turnstile.render(el, {
  sitekey: "PUBLIC_KEY",
  // If you want to use ONLY your own "Είσοδος" button, keep this invisible.
  size: "invisible",
  callback: function (token) {
    window.turnstileToken = token || "";
    window.setSignInDisabled(false);
    window.turnstileAutoSubmit = false;
    document.getElementById("sign-in)?.click();
  },
  "expired-callback": function () {
    window.turnstileToken = "";
    window.turnstileAutoSubmit = false;
    window.setSignInDisabled(false);
  },
  "error-callback": function () {
    window.turnstileToken = "";
    window.turnstileAutoSubmit = false;
    window.setSignInDisabled(false);
  },
});

if (document.readyState === "loading") {
  document.addEventListener("DOMContentLoaded", render);
} else {
  render();
}
};

```

Δε θα χρειαστεί να αλλάξουμε κάτι εδώ εκτός από το `document.getElementById("sign-in")` να δείχνει το κουμπί σύνδεσης. και έπειτα παίρνουμε όπως είναι αυτό το κομμάτι πριν το request σύνδεσης:

```

const turnstileToken =
  window.turnstileToken ||
  (window.turnstileWidgetId !== null && typeof turnstile !== "undefined"
    ? turnstile.getResponse(window.turnstileWidgetId)
    : "");

```

```
if (!turnstileToken) {  
    // Disable your login button while captcha runs  
    window.setSignInDisabled(true);  
    window.turnstileAutoSubmit = true;  
  
    // For invisible Turnstile, execute it now  
    try {  
        turnstile.execute(window.turnstileWidgetId);  
    } catch (e) {  
        window.setSignInDisabled(false);  
        window.turnstileAutoSubmit = false;  
    }  
    return;  
}  
  
// Disable while the login request is in-flight (prevents double submit)  
window.setSignInDisabled(true);
```

Και απλα συμπεριλαμβανουμε στην αποστολη μετα το

turnstileToken

Generate SSL certificate for Wildcard

```
sudo certbot certonly --manual --preferred-challenges=dns --email  
thanasimpalatsoukas@gmail.com --agree-tos -d "*.emblem-ergani.gr"
```

How to run mysql clone on a server

1. Prerequisites

Before you start, ensure both servers meet these requirements:

- **Version:** Both must be MySQL 8.0.17 or higher.
- **Operating System:** Must be the same (e.g., both Linux).
- **Plugins:** The `clone_plugin` must be installed on both.

2. Step-by-Step Configuration

Phase A: Prepare the Donor (Primary)

Log into your **Primary** server and run:

SQL
❏ ❏

```
-- Install the plugin
INSTALL PLUGIN clone SONAME 'mysql_clone.so';

-- Create a replication user with cloning permissions
CREATE USER 'clone_user'@'%' IDENTIFIED BY 'password';
GRANT BACKUP_ADMIN ON *.* TO 'clone_user'@'%';
```

Phase B: Prepare the Recipient (Replica)

Log into your **New Replica** server and run:

SQL
■ ■

```
-- Install the plugin
INSTALL PLUGIN clone SONAME 'mysql_clone.so';

-- Allow the recipient to be overwritten by the donor
SET GLOBAL clone_valid_donor_list = 'primary-ip-address:3306';
```

Phase C: Execute the Clone

On the **Recipient** server, trigger the data transfer. **Warning:** This will delete all existing data on the recipient and restart the MySQL service automatically after completion.

SQL
■ ■

```
CLONE INSTANCE FROM 'clone_user'@'primary-ip-address':3306 IDENTIFIED BY 'password';
```

3. Completing the Replication

The beauty of the Clone plugin is that it automatically captures the **Binary Log** coordinates (GTID or filename/position) during the transfer.

Once the recipient restarts, log back in and check the status:

SQL
■ ■

```
SELECT BINLOG_FILE, BINLOG_POS FROM performance_schema.clone_status;
```

Option 1: If using GTIDs (Recommended)

If your primary uses GTIDs, replication might start automatically or simply require:

SQL
■ ■

```
CHANGE REPLICATION SOURCE TO
    SOURCE_HOST='primary-ip-address',
    SOURCE_USER='repl_user',
    SOURCE_PASSWORD='password',
    SOURCE_AUTO_POSITION=1;
START REPLICA;
```

Option 2: If using Binary Log Position

SQL
■ ■

```
-- Use the coordinates found in performance_schema.clone_status
CHANGE REPLICATION SOURCE TO
    SOURCE_HOST='primary-ip-address',
    SOURCE_USER='repl_user',
    SOURCE_PASSWORD='password',
    SOURCE_LOG_FILE='binlog.00000X',
    SOURCE_LOG_POS=123;
START REPLICA;
```

emblem ergani disaster response for traffic

Εαν βρεθουμε στη δυσχερη θέση να πεσει ο κεντρικος σερβερ που φιλεξει το emblem ergani (88.198.203.86) έχουμε έτοιμο 2ο σερβερ ο οποίος είναι ετοιμος να λαβει τη θέση του (167.235.135.104).

Βήματα για την μεταφορα λειτουργιας απο τον 88.198.203.86 -> 167.235.135.104

- Αλλαγή A Record στον 167.235.135.104

- Ενεργοποιηση apache conf a2ensite

a2ensite /etc/apache/sites-available/ergani.emblem.gr.conf

Υπαρχει μια περιπτωση εδω να εχουμε modules στο ergani.emblem.gr.conf τα οποια δεν υπαρχουν στο σερβερ. Κατεβαστε τα

- Δημιουργια νεων SSL certificates με certbot ή μεταφορα certs απο το προηγουμενο σερβερ

- Ανοιγμα cronjob και κανε comment out αυτα που βαλαμε σε comments για εκτακτη αναγκη

Automate git pull securely

? Best setup for your repo server (recommended): SSH deploy key

This is the standard production approach.

1. On the server: create an SSH key

```
ssh-keygen -t ed25519 -C "emblem-ergani-server"
```

Press Enter for default path:

```
/root/.ssh/id_ed25519
```

2. Show the public key

```
cat ~/.ssh/id_ed25519.pub
```

Copy it.

3. Add it to your GitHub repo

Go to:

☐ Repo → **Settings** → **Deploy keys** → **Add key**

Paste the key.

Important choice:

- ✓ Enable “**Allow write access**” if server will push
 - Otherwise leave it read-only (safer for production pulls)
-

4. Switch repo remote to SSH (on server)

Inside your project:

```
git remote set-url origin git@github.com:semantic-software/emblem-ergani.git
```

5. Test authentication

```
ssh -T git@github.com
```

You should see:

```
Hi semantic-software! You've successfully authenticated...
```

6. Now automation works (no password ever)

```
git pull origin main
```

? Automate it (cron / deploy hook)

Example cron (pull every minute)

```
crontab -e
```

```
* * * * * cd /var/www/emblem-ergani && git pull origin main
```

?????? setup ??? ?? cookie widget (semantic cookie)

?????? setup ??? ?? cookie widget

1. Βάλε το widget script στο τέλος της σελίδας Τοποθέτησέ το μία φορά, λίγο πριν το κλείσιμο του `</body>`: `html <script src="https://cdn.emblem.gr/semantic-cookie/cookie-widget.js"></script>`
2. Σημαντικό για scripts που θες να μπλοκάρονται μέχρι consent Τα tracking scripts πρέπει να δηλώνονται ως **text/plain** (όχι κανονικό JS), με `data-*` attribute κατηγορίας.
Παράδειγμα: `html <script type="text/plain" data-cookiecategory="analytics"> // Facebook Pixel, GA, Clarity, κτλ. </script>` Μπορείς αντί για `analytics` να βάλεις άλλη κατηγορία, ανάλογα με το script.
3. Consent update προς Google (προαιρετικό αλλά προτεινόμενο) Αν χρησιμοποιείς GTM/gtag, κράτησε listener για το event `cookieConsentSaved`: `html <script> document.addEventListener('cookieConsentSaved', function(event) { var consent = event && event.detail && event.detail.consent ? event.detail.consent : null; if (!consent || typeof gtag !== 'function') return; var marketingGranted = !!consent.marketing; var analyticsGranted = !!consent.analytics; gtag('consent', 'update', { ad_storage: marketingGranted ? 'granted' : 'denied', ad_user_data: marketingGranted ? 'granted' : 'denied', ad_personalization: marketingGranted ? 'granted' : 'denied', analytics_storage: analyticsGranted ? 'granted' : 'denied', functionality_storage: 'granted', personalization_storage: marketingGranted ? 'granted' : 'denied', security_storage: 'granted' }); }); </script>`
4. Σειρά φόρτωσης που προτείνεται 1. Consent Mode default (denied by default) 2. GTM/gtag 3. Listener `cookieConsentSaved` 4. Cookie widget script (πριν το `</body>`)
5. Γρήγορο checklist - Το widget μπαίνει μόνο μία φορά. - Τα scripts που θέλεις να ελέγχονται από consent είναι `type="text/plain"` και έχουν `data-cookiecategory="..."`. - Με accept analytics/marketing, ενημερώνεται σωστά το consent στο gtag. - Με reject, τα αντίστοιχα storage μένουν `denied`.