MySQL Replication 101

Giuseppe Maxia
Continuent, Inc

about me - Giuseppe Maxia

- a.k.a. The Data Charmer
- QA Director at Continuent, Inc
- Long time hacking with MySQL features
- Formerly, community manager,db consultant, designer, coder.
- A passion for QA and open source
- Blogger
- http://datacharmer.blogspot.com

AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

AGENDA

Why replication	• **The web economy** • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

The world today is
dominated by the
web economy

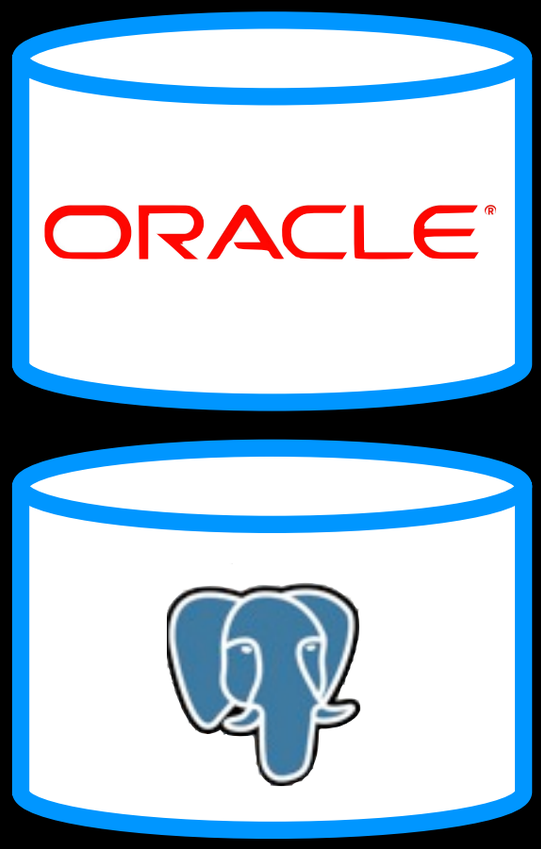
Databases are the backbone of the web economy

What database for the web?

What database for the web?

The most **powerful** database

What database for the web?

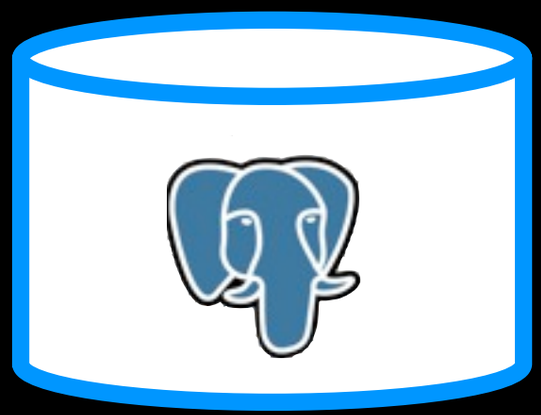


The most **powerful** database

The most **advanced** open source database

What database for the web?

The most **powerful** database

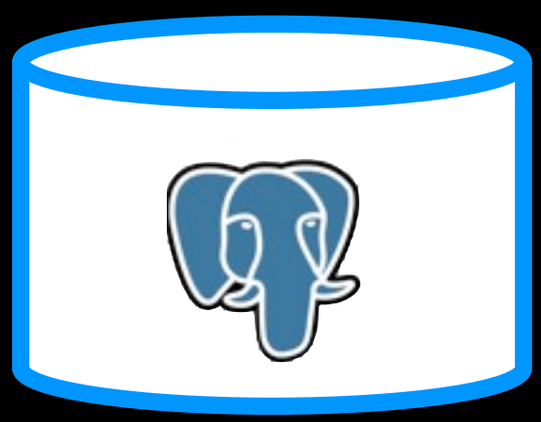


The most **advanced** open source database

The most **deployed** open source database

What database for the web?

The most **powerful** database



The most **advanced** open source database

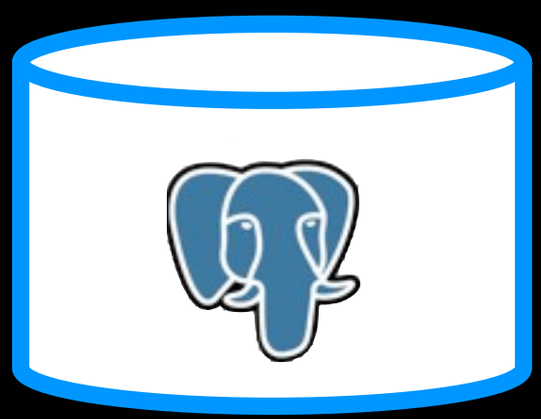
The most **deployed** open source database

The most **popular** open source database

Actually, **MySQL** databases are the backbone of the web economy

What database for the web?

No built-in replication



No built-in replication

No built-in replication

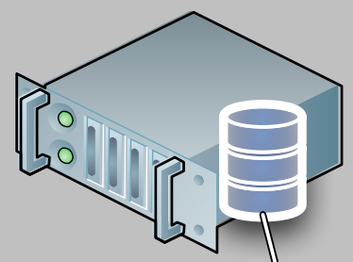
Built-in replication

More precisely, **MySQL REPLICATION** is the backbone of the web economy

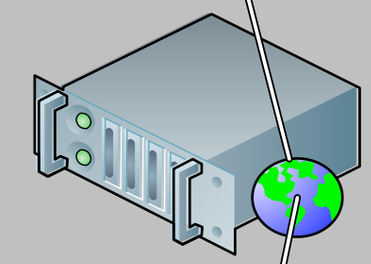
AGENDA

Why replication	• The web economy • **Scaling out**
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

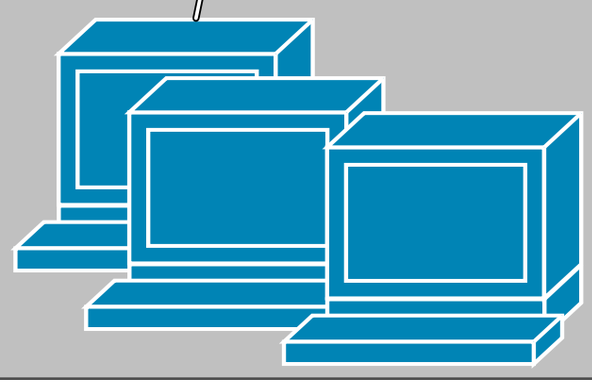
database server



r/w requests



web server

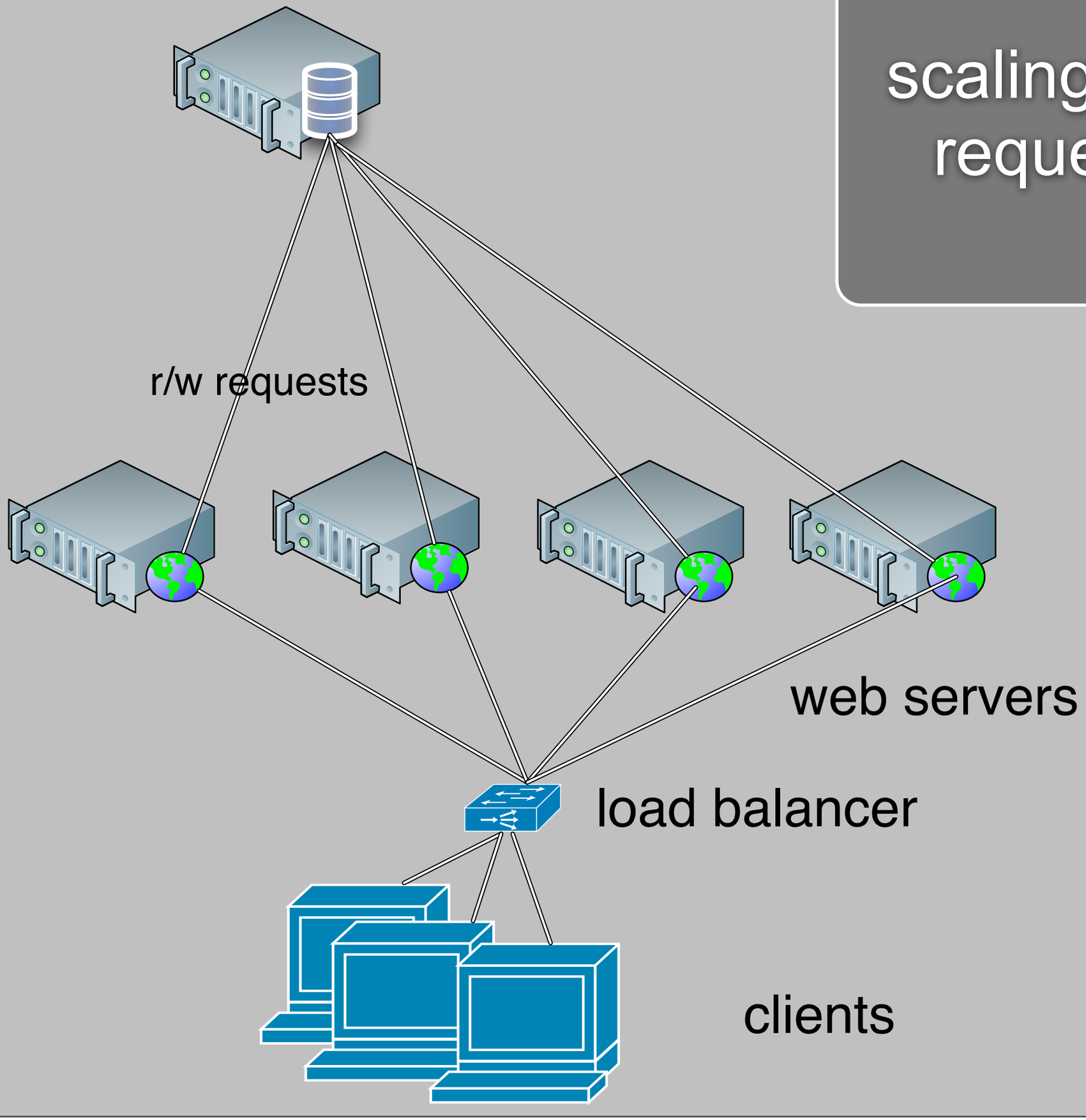


clients

a simple web application scheme

database server

scaling web requests



r/w requests

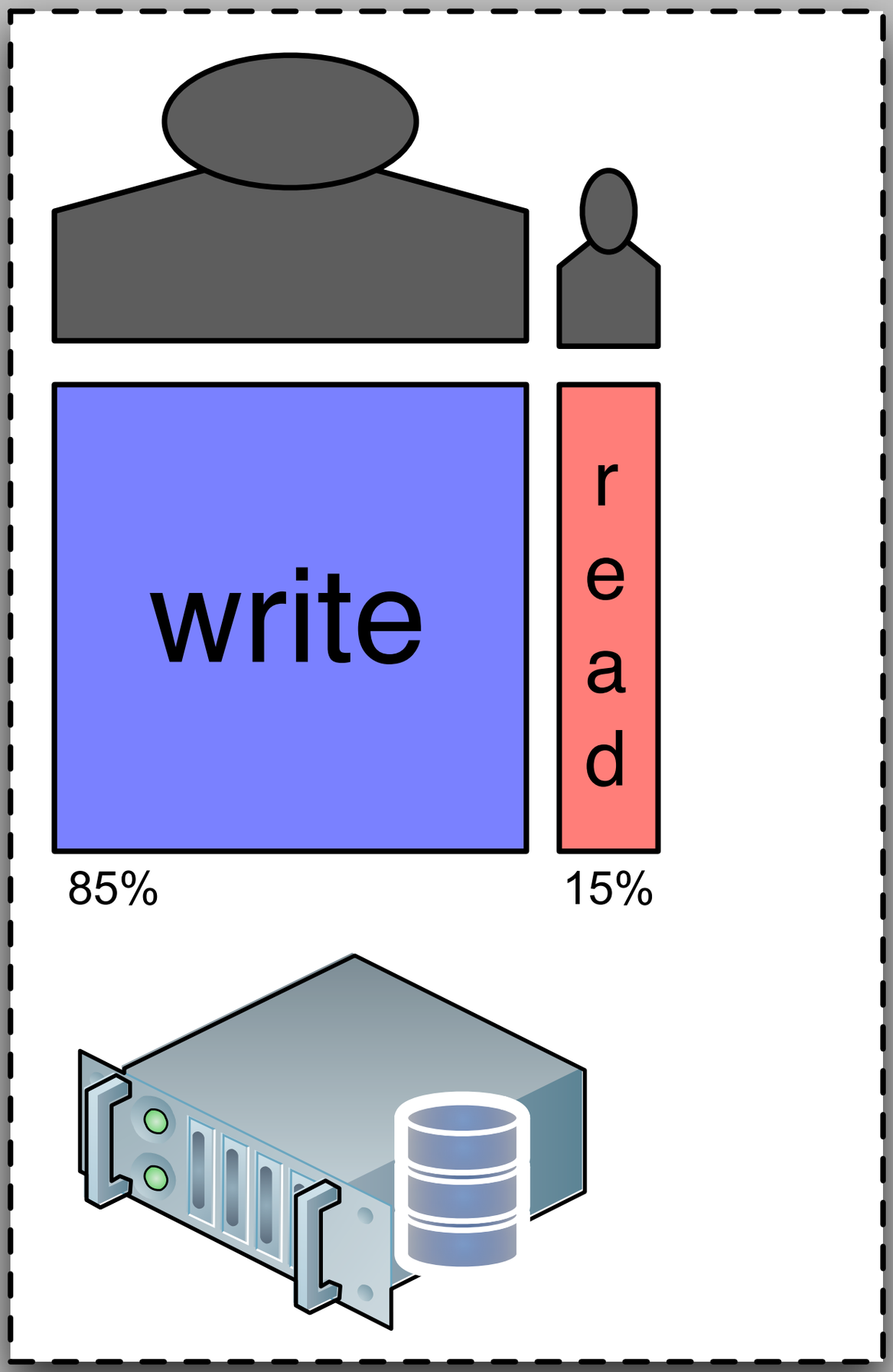
web servers

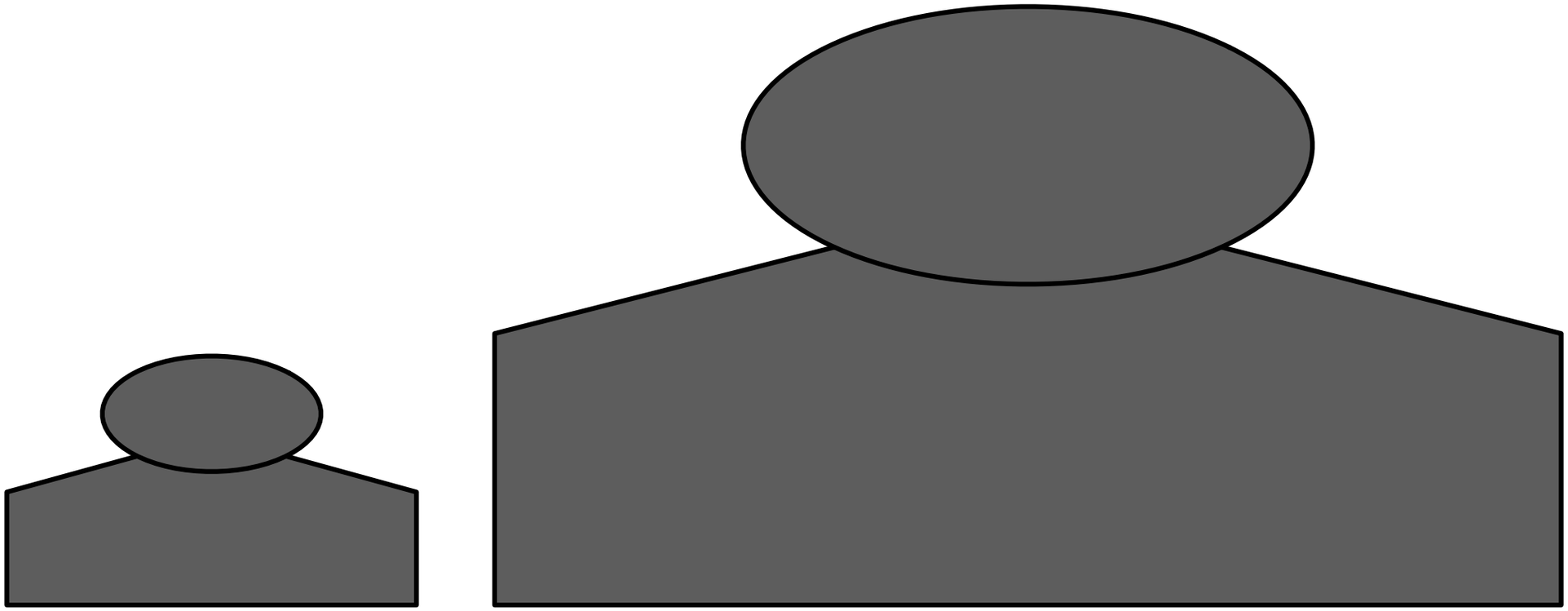
load balancer

clients

13

database load on a simple web application



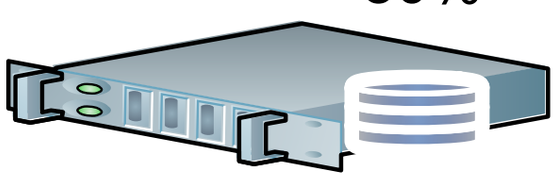


write

read

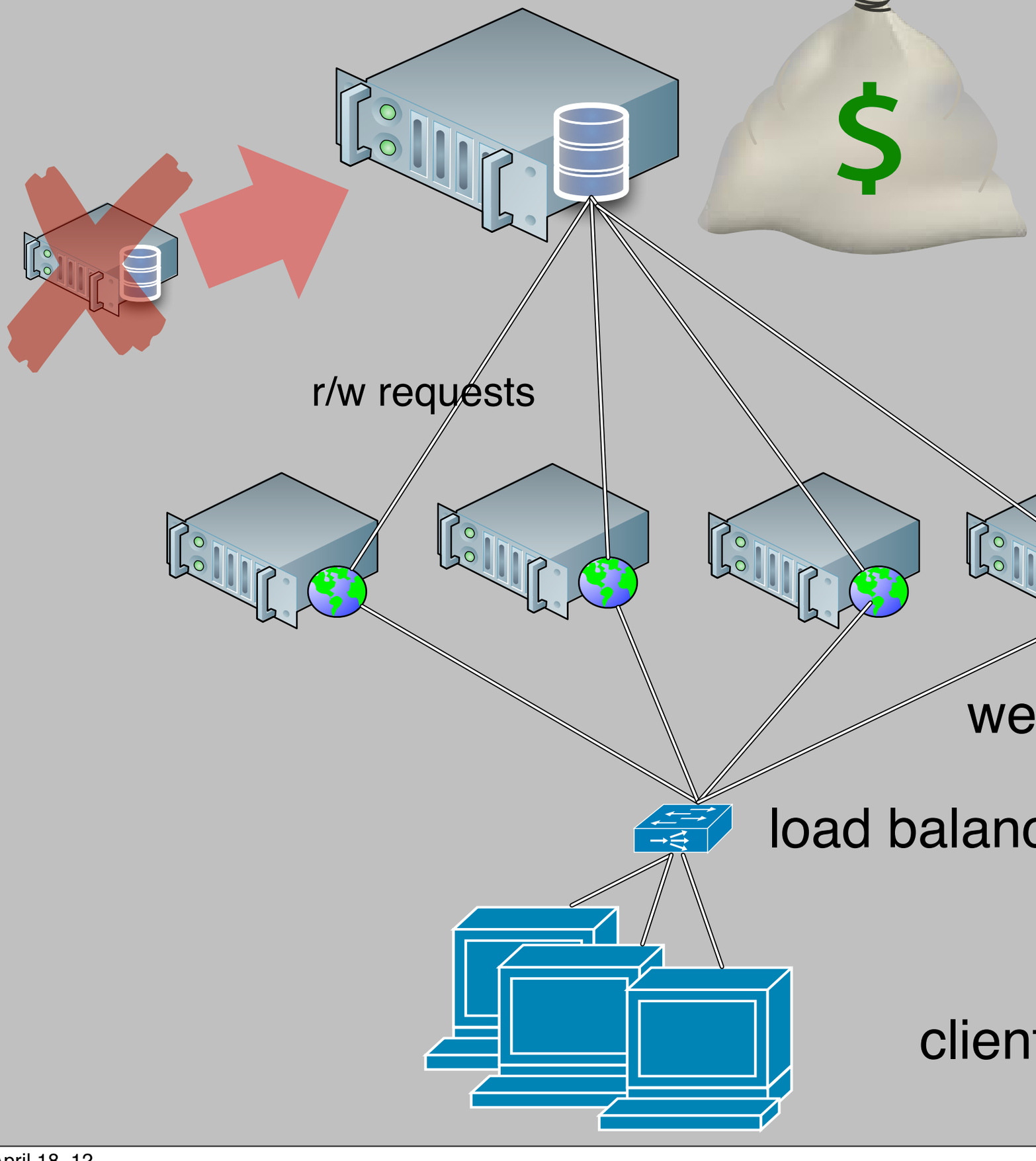
20%

80%



database load on a **successful** web application

database server



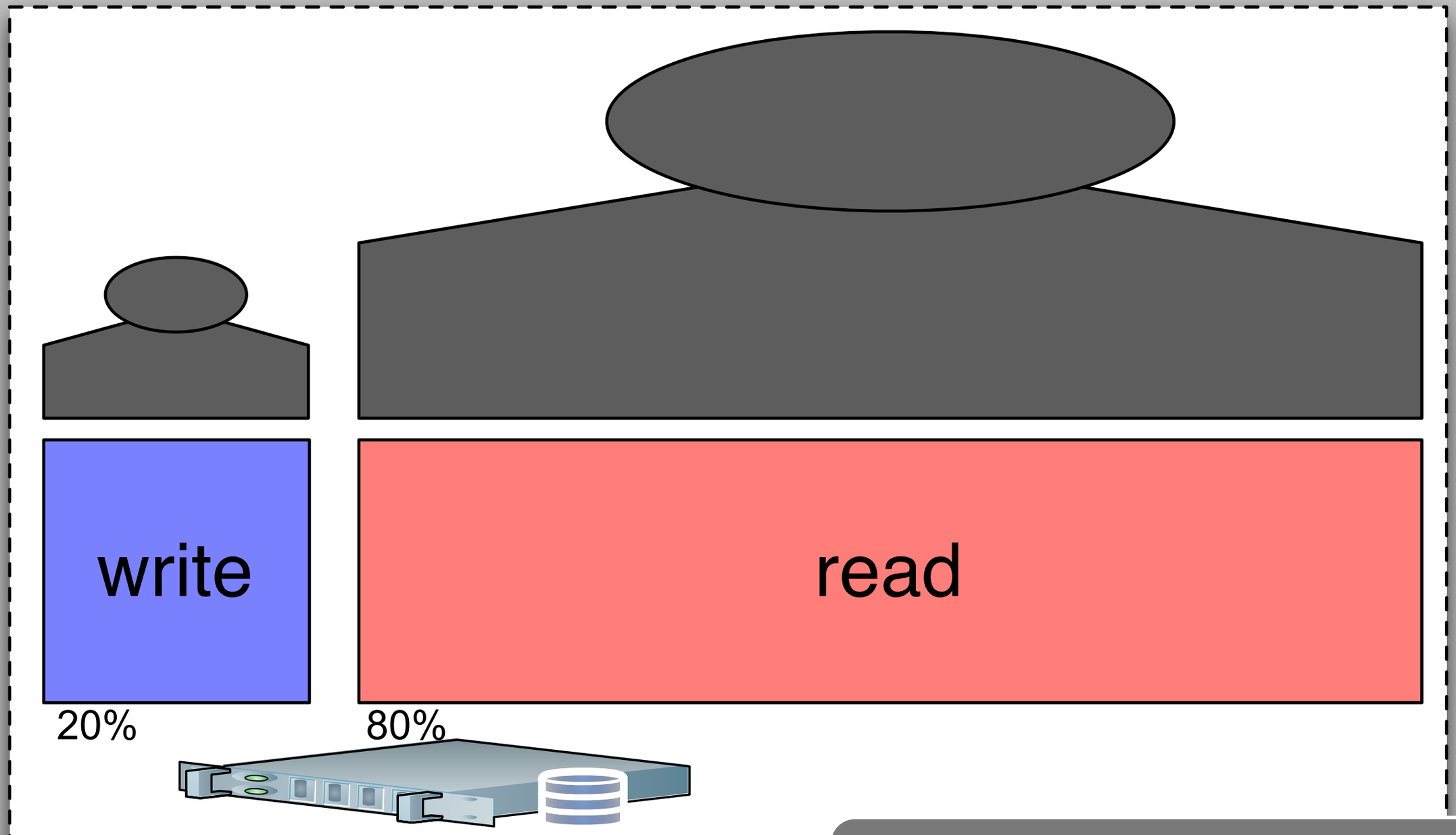
scaling up means buying a bigger database server

r/w requests

web servers

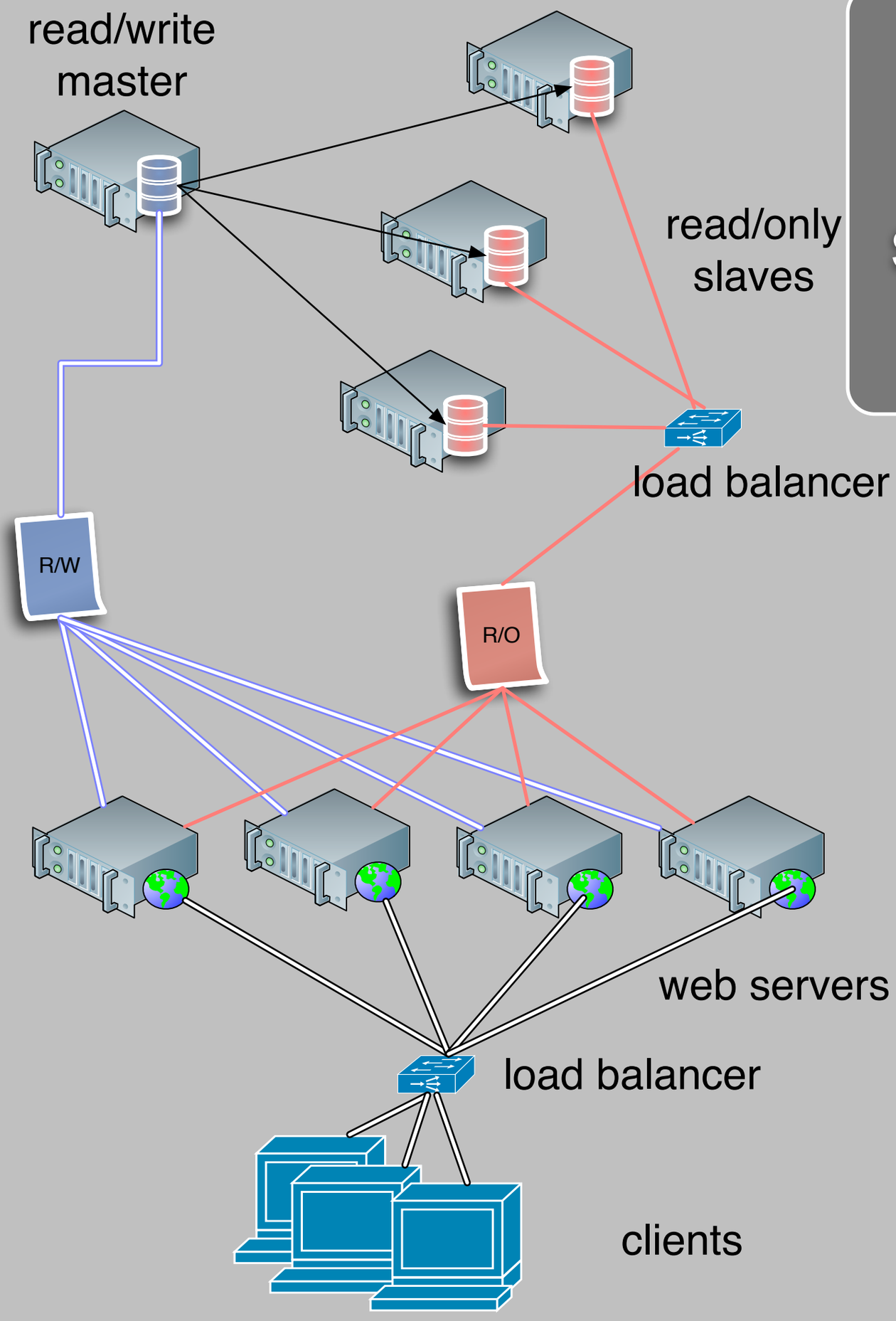
load balancer

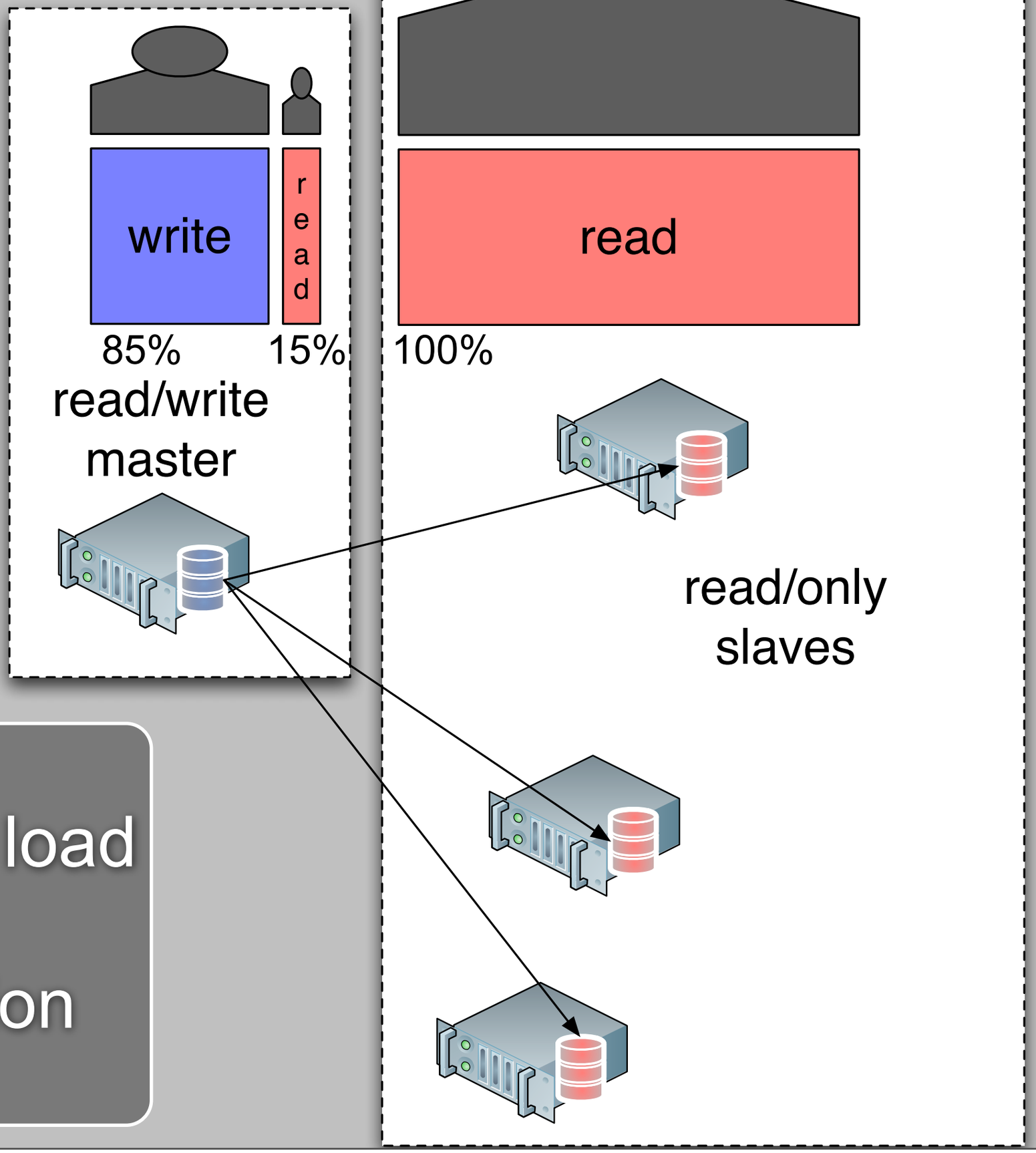
clients



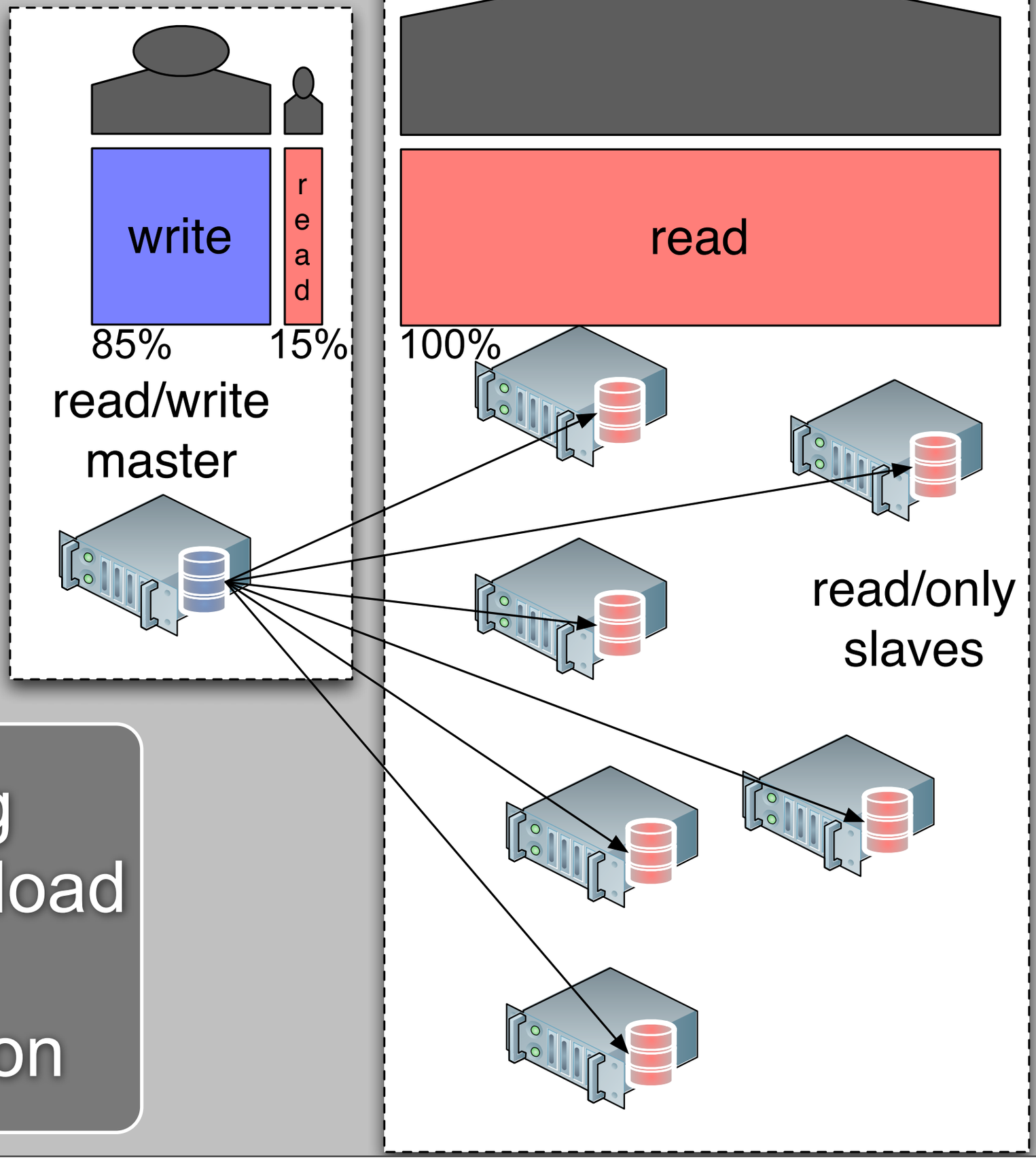
the bigger database server will eventually have the same problem

a web application scheme with replication





database load with replication



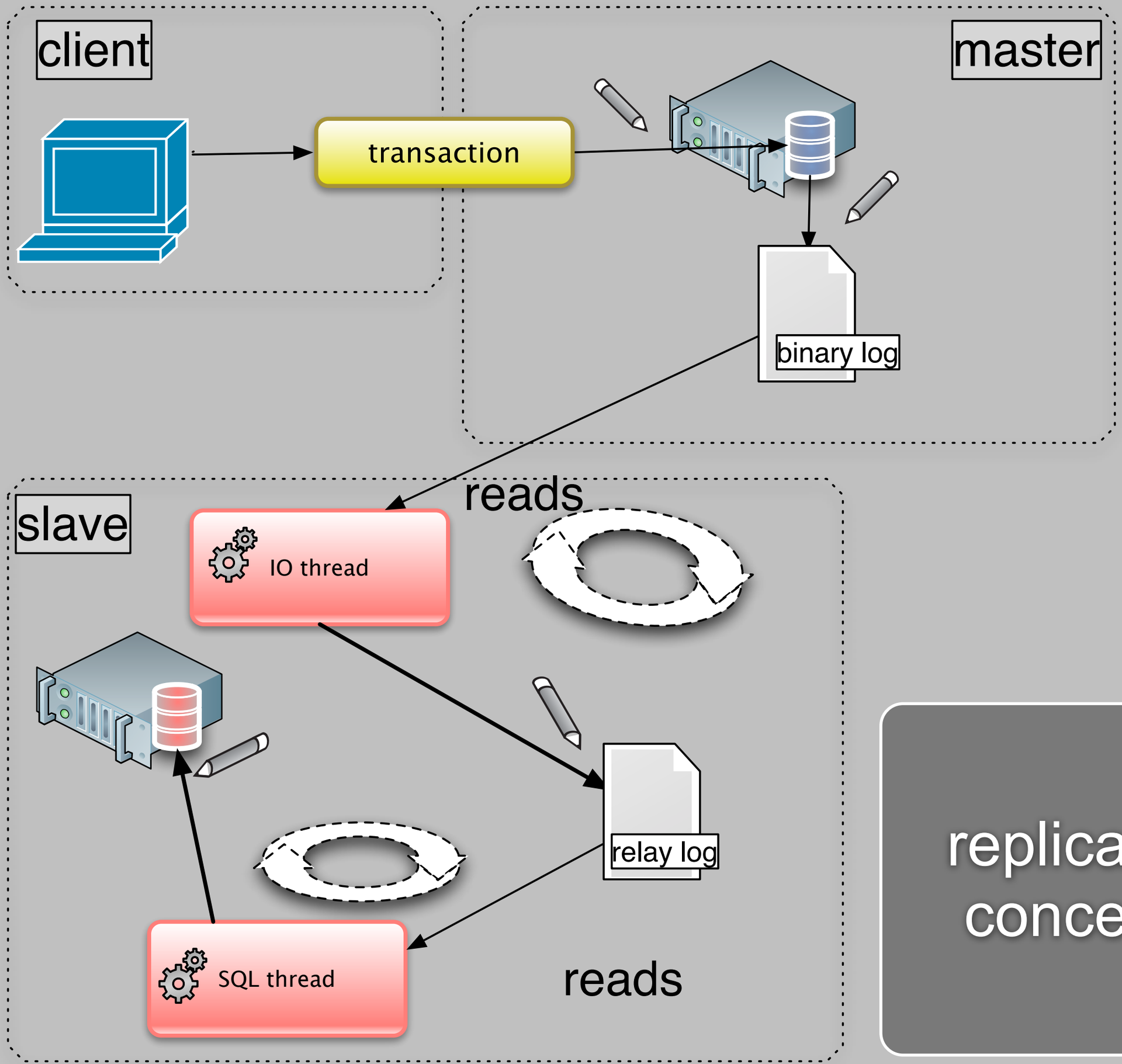
scaling database load with replication

Replication assessment

	without replication	with replication
database handling	**easy**	**harder**
performance	**high**	**lower (binary logs)**
Point in Time recovery	**none**	**easy**
failover	**none**	**possible**
write scaling	**none**	**minimal**
backup	**with downtime**	**without downtime**
read scaling	**none**	**easy**

AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

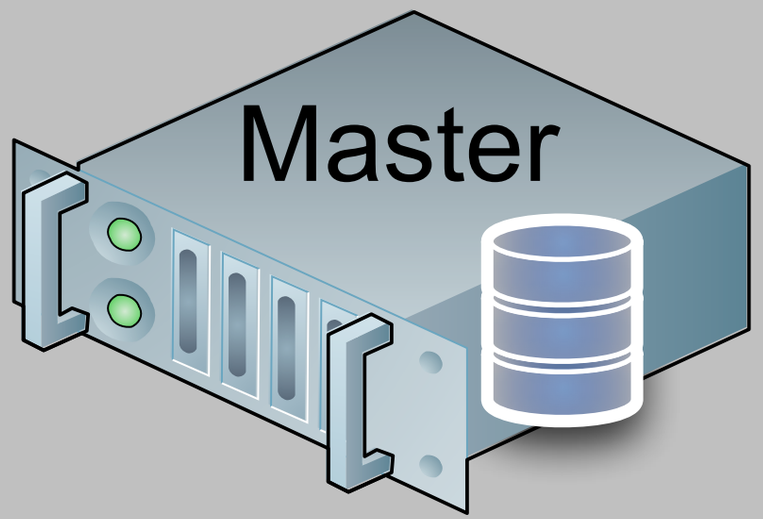


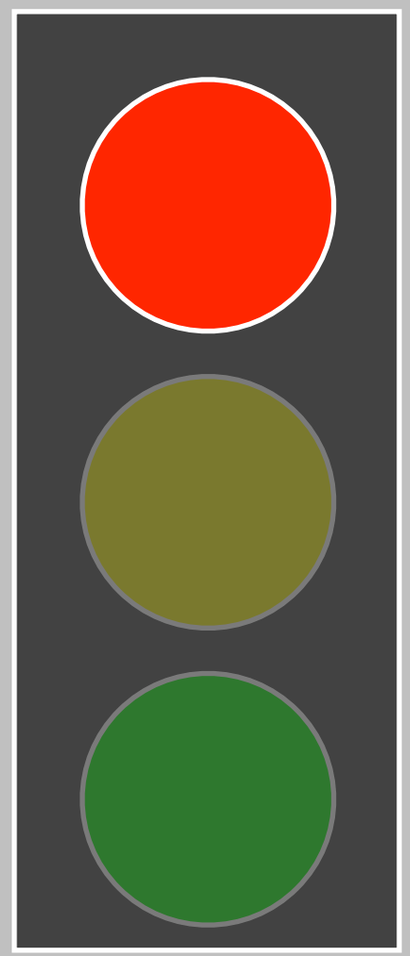
replication concepts

AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• **From single server to replication** • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

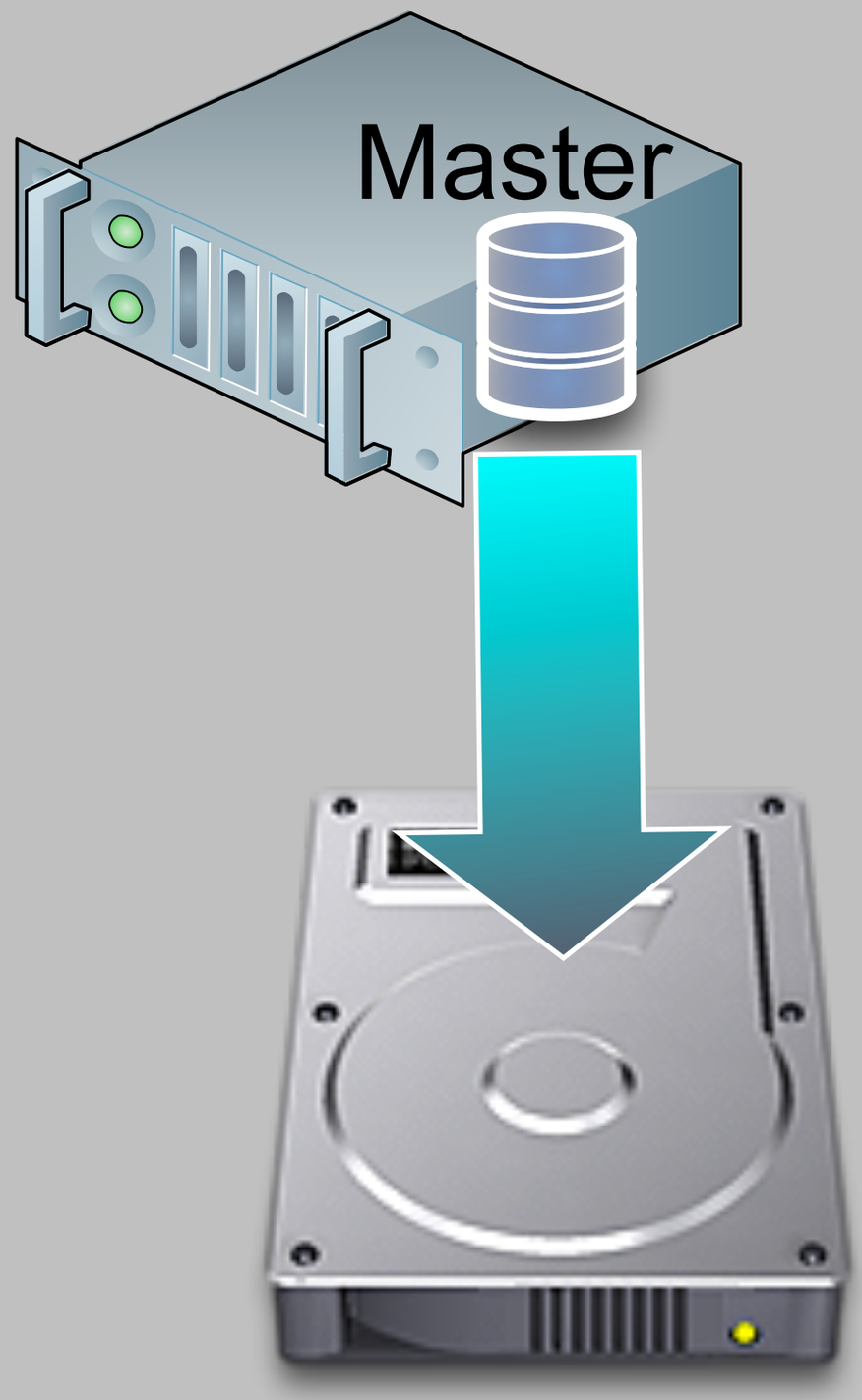
1 SHUT DOWN THE DATABASE SERVER





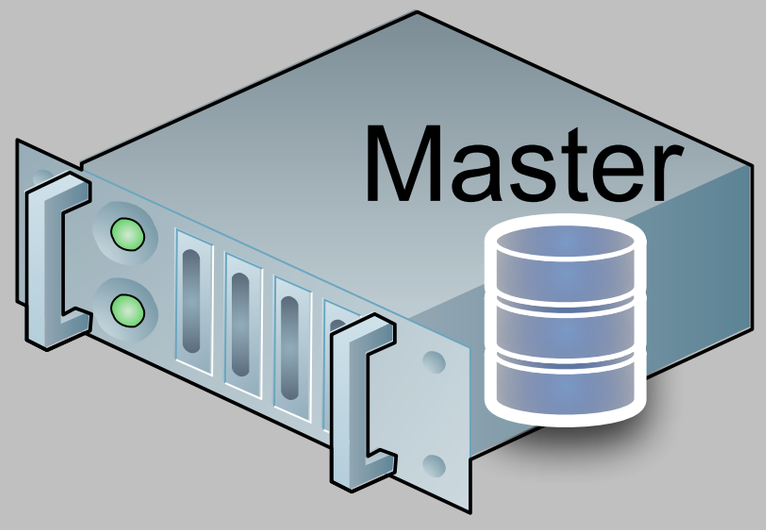
2

MAKE A BACKUP COPY



3

ENABLE THE MASTER

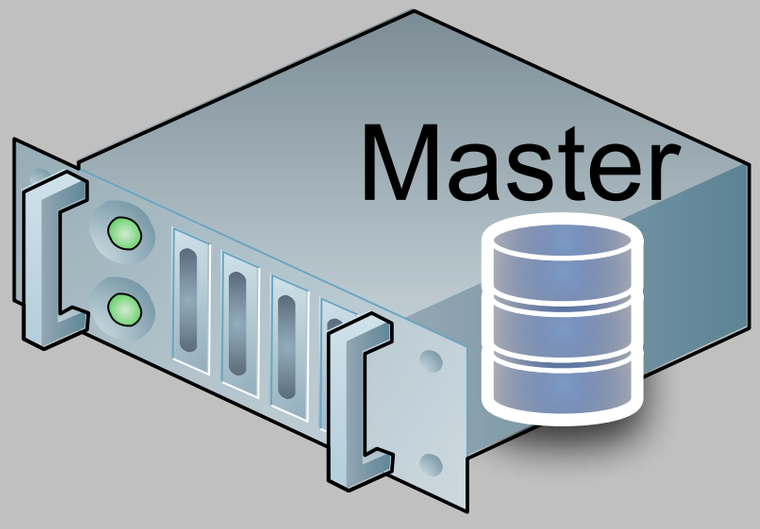


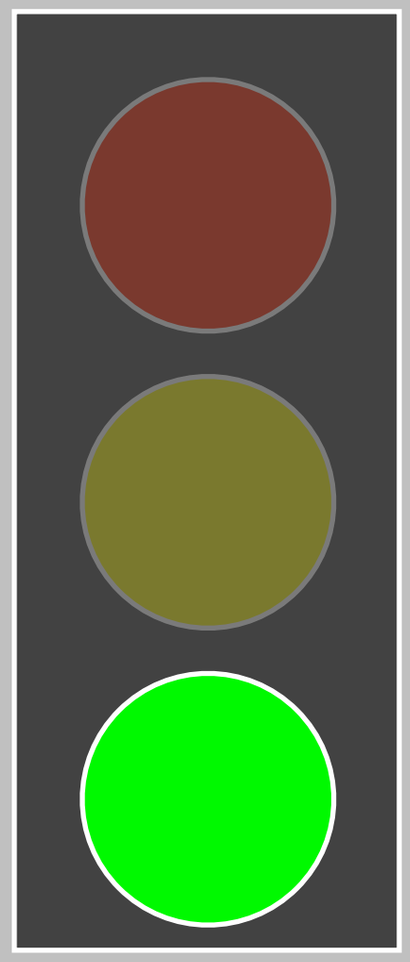
Configuration file

```
[mysqld]
log-bin=mysql-bin
server-id=1
```

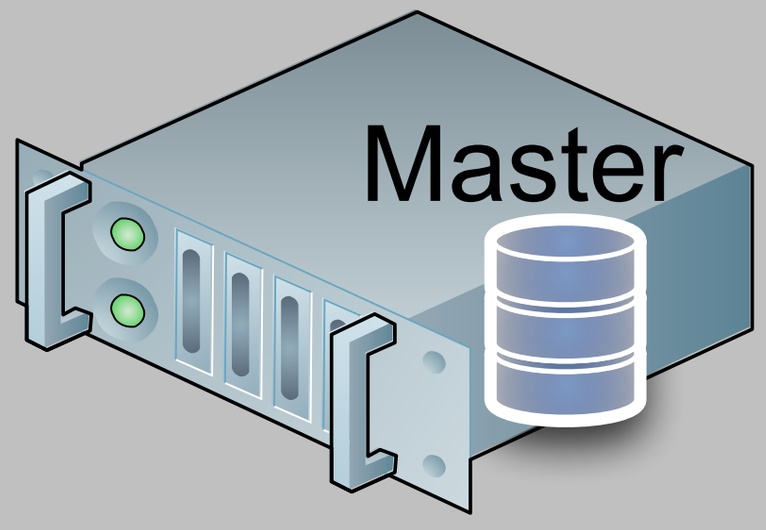
4

RESTART THE MASTER





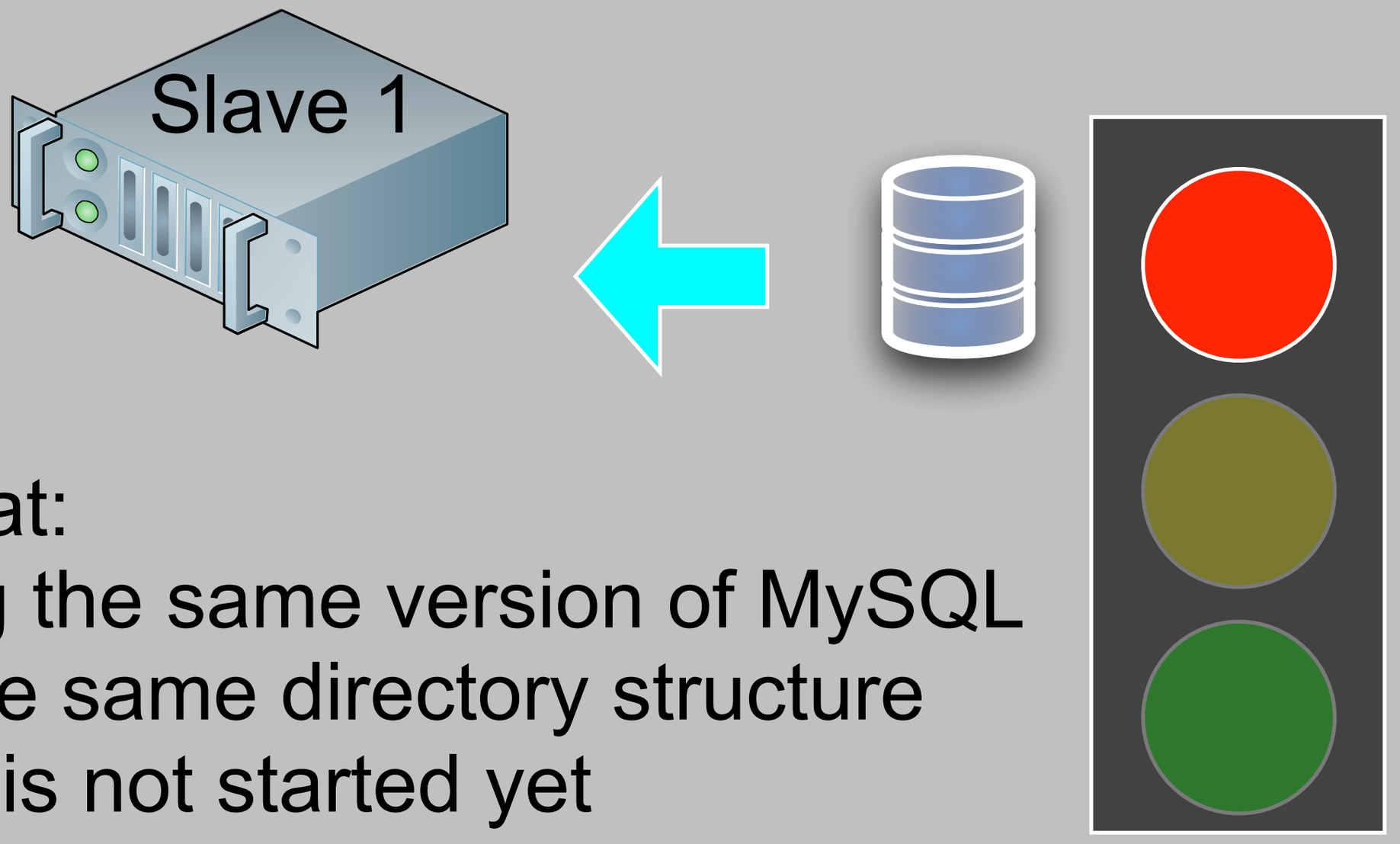
5 CREATE REPLICATION USER



SQL command

```
GRANT REPLICATION SLAVE ON *.*
to 'slave_user@'10.10.100.%'
IDENTIFIED BY 'slave_pass';
```

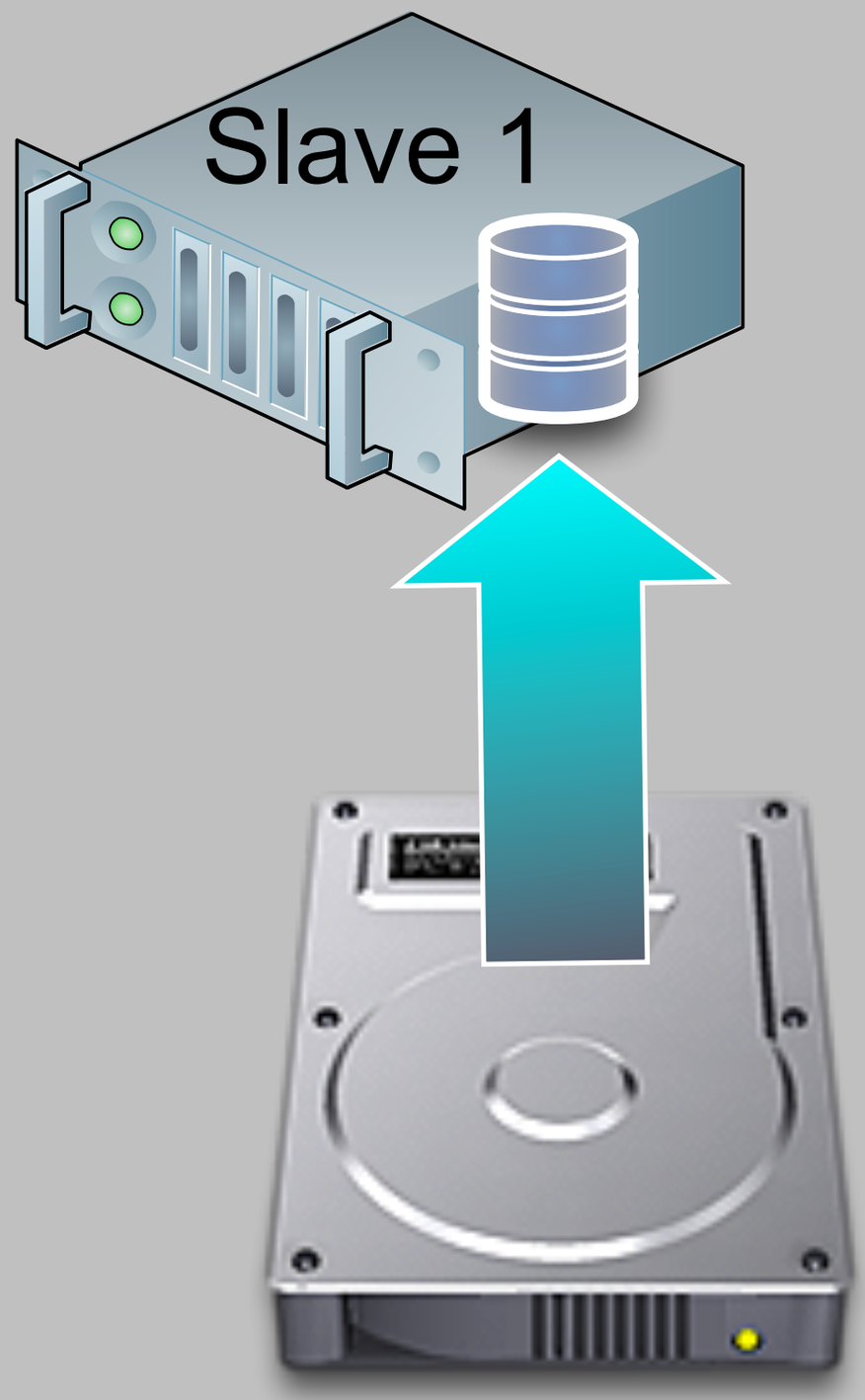
6 INSTALL MySQL on the slave



Make sure that:

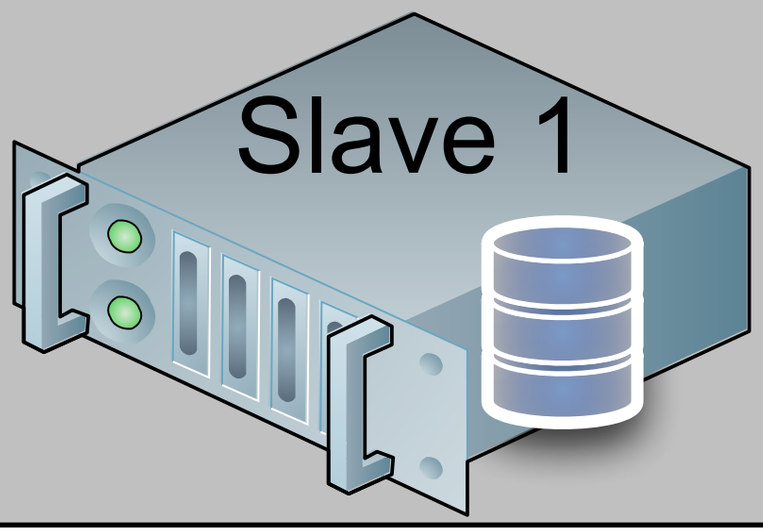
- You're using the same version of MySQL
- You have the same directory structure
- The server is not started yet

7 COPY THE MASTER DATA to the slave



8

ENABLE THE SLAVE

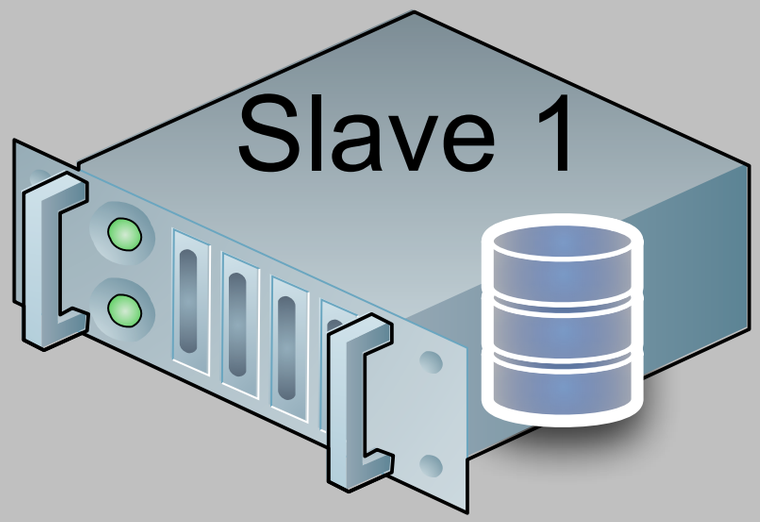


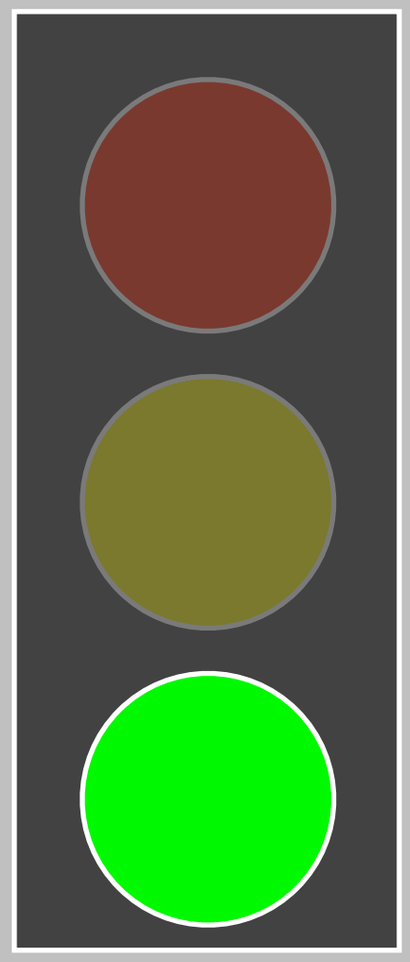
Configuration file

```
[mysqld]
server-id=2
relay-log=mysql-relay
read-only
# optional:
log-bin=mysql-bin
```

9

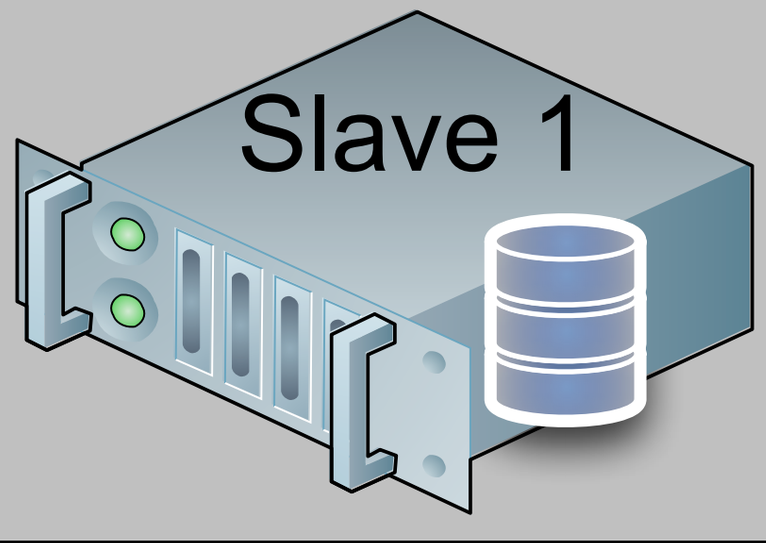
START THE SLAVE SERVER





10

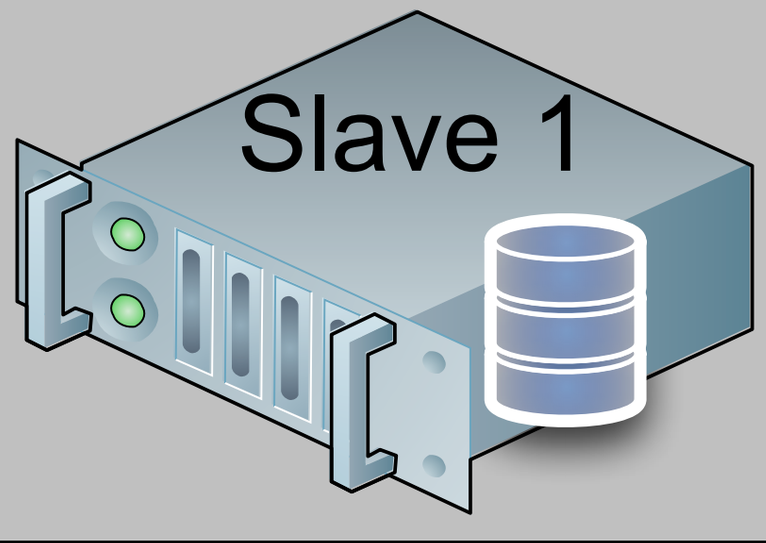
INITIALIZE THE SLAVE



SQL command

```
SET MASTER TO
MASTER_HOST=master_IP,
MASTER_PORT=3306,
MASTER_USER=slave_user,
MASTER_PASSWORD='slave_pwd';
```

11 START THE SLAVE SERVICE

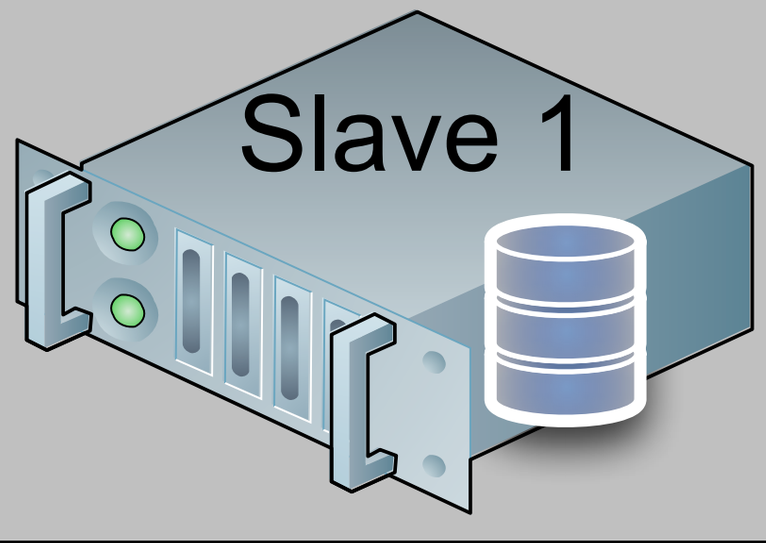


SQL command

```
START SLAVE;
```

12

CHECK THE SLAVE



SQL command

```
SHOW SLAVE STATUS \G

...
 Slave_IO_Running: Yes
Slave_SQL_Running: Yes
...
```

Troubleshooting

- **SHOW SLAVE STATUS says SLAVE_IO_RUNNING=No**
  - Make sure that the slave host can connect to the master
  - Make sure that master and slave have different Server-id
  - Check the error log of both master and slave

Testing the slave

- Create a table in the master.
- Make sure that the slave has replicated the table.
- Insert data in the master
- read that data in the slave

38

What if the master was already logging?

- You have two options:
  - Physical copy
  - Logical copy

Physical copy

- stop master
- make copy
- remove binary log files and index
- start master
- (alternative: use xtrabackup)

Logical copy

- mysqldump --all-databases --master-data

Common replication commands

- CHANGE MASTER TO
- SHOW MASTER STATUS
- SHOW SLAVE STATUS
- START SLAVE
- STOP SLAVE
- RESET MASTER (caution!)
- RESET SLAVE (caution!)

42

CHANGE MASTER TO basic syntax

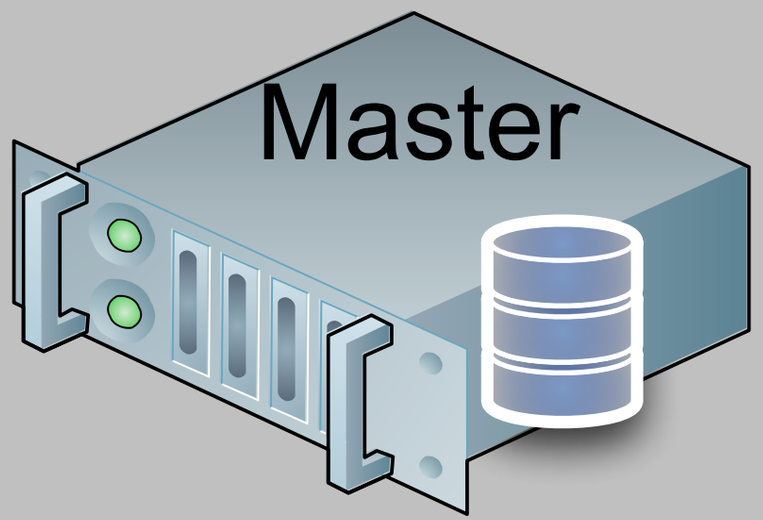
```
STOP SLAVE;

CHANGE MASTER TO
MASTER_HOST='hostname', # or IP
MASTER_PORT=3306,
MASTER_USER='slaveuser',
MASTER_PASSWORD='slavepassword',
MASTER_LOG_FILE='filename', # default: first binlog
MASTER_LOG_POS=123456;  # default: from the beginning

START SLAVE;
```

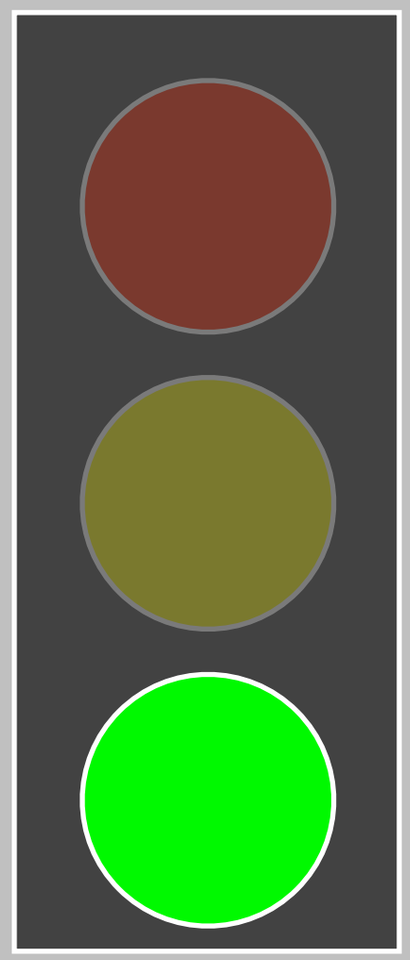
AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • **Adding a slave**
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

1

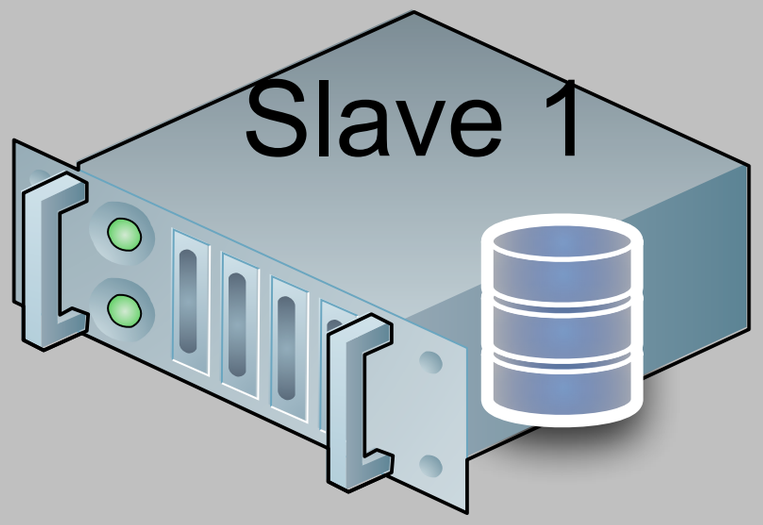
NO NEED TO STOP THE MASTER!

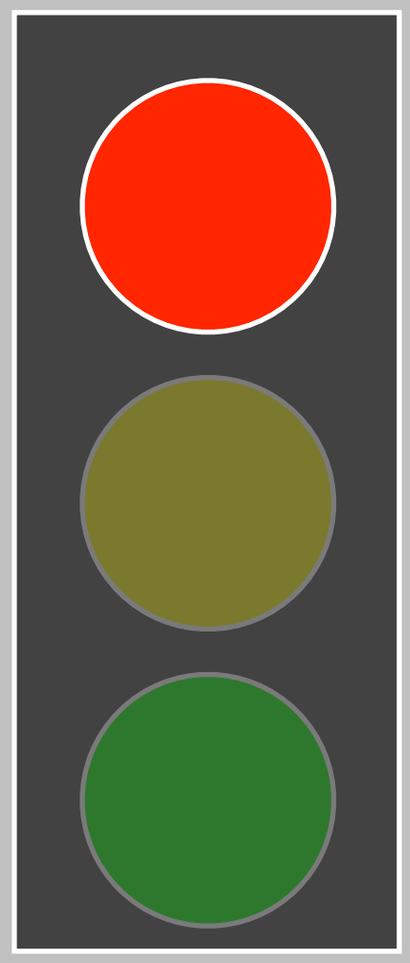




2

STOP THE SLAVE

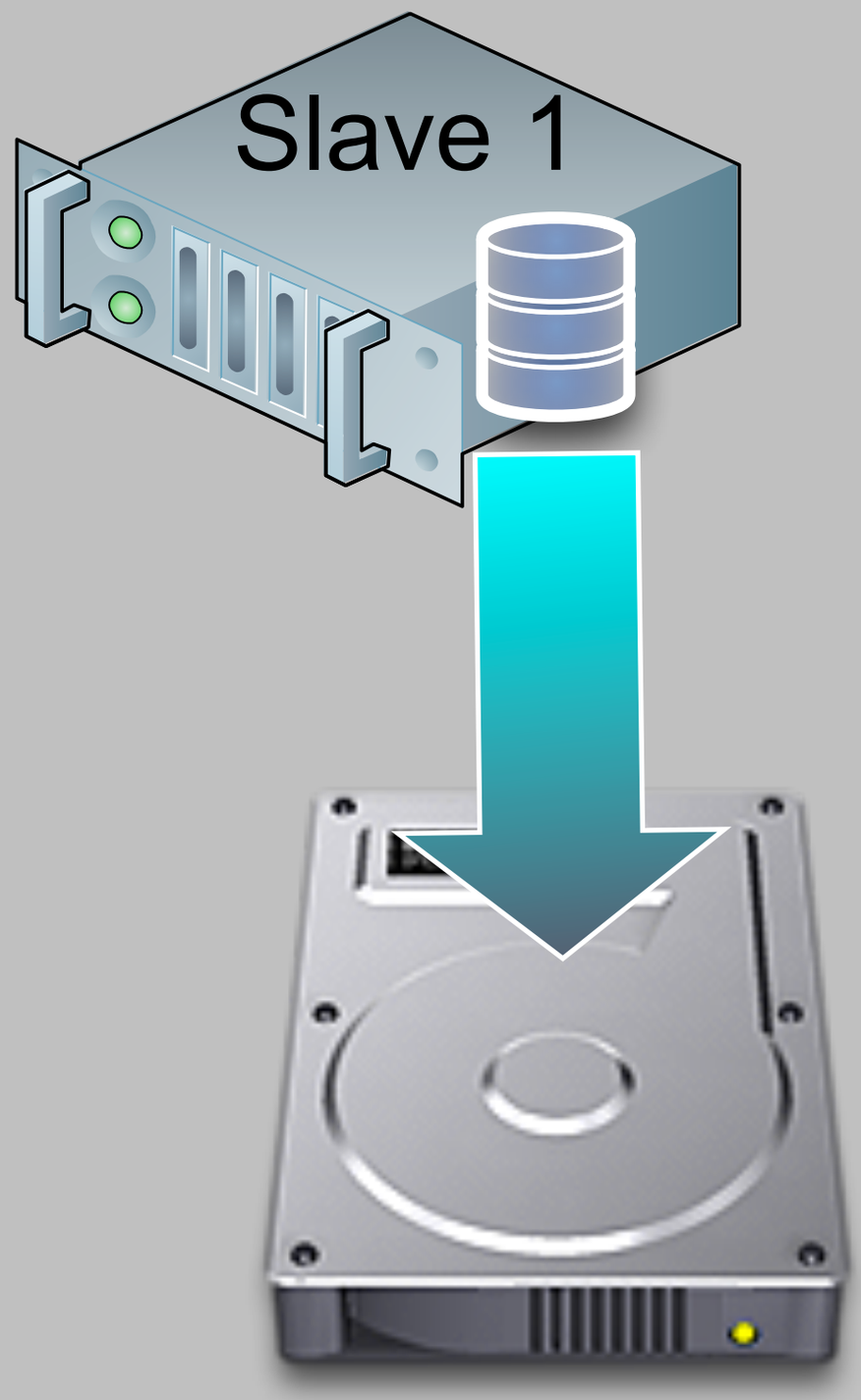




SQL command

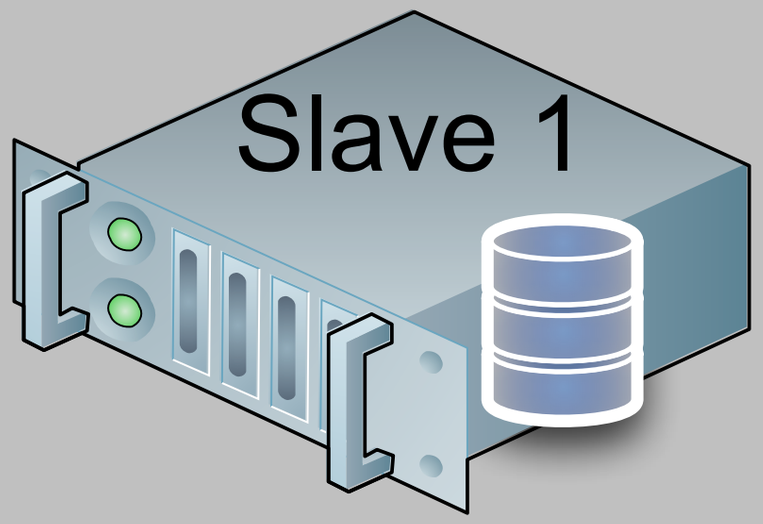
```
STOP SLAVE IO_THREAD;
# wait until the SQL_THREAD
# has done everything
STOP SLAVE SQL_THREAD;
# STOP THE SERVER
```

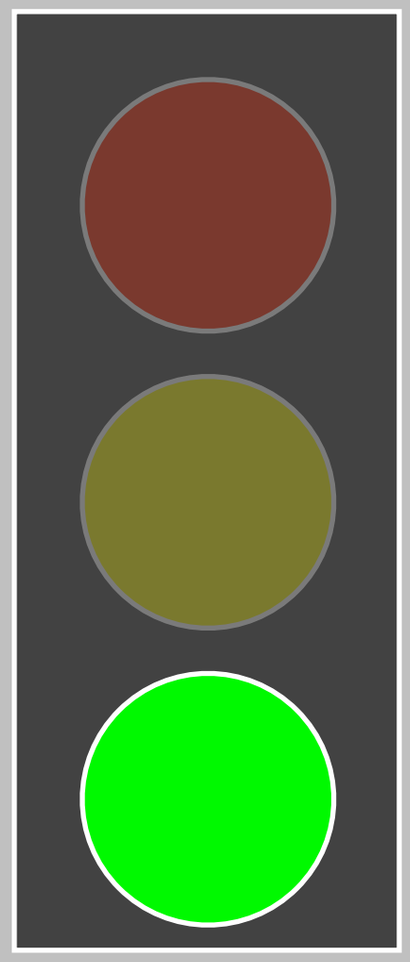
3 MAKE A COPY OF THE DATA DIRECTORY



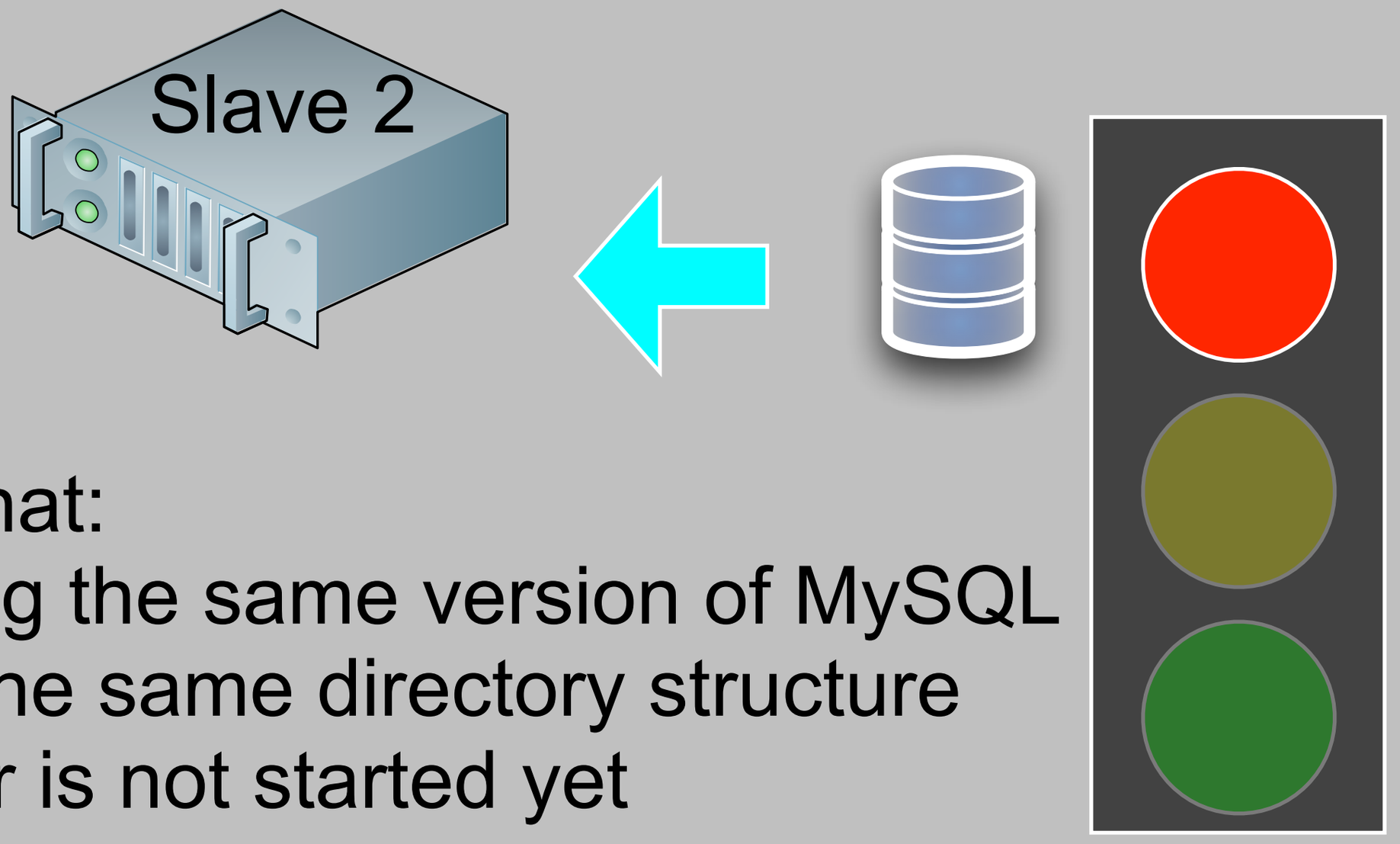
4

RESTART THE SLAVE





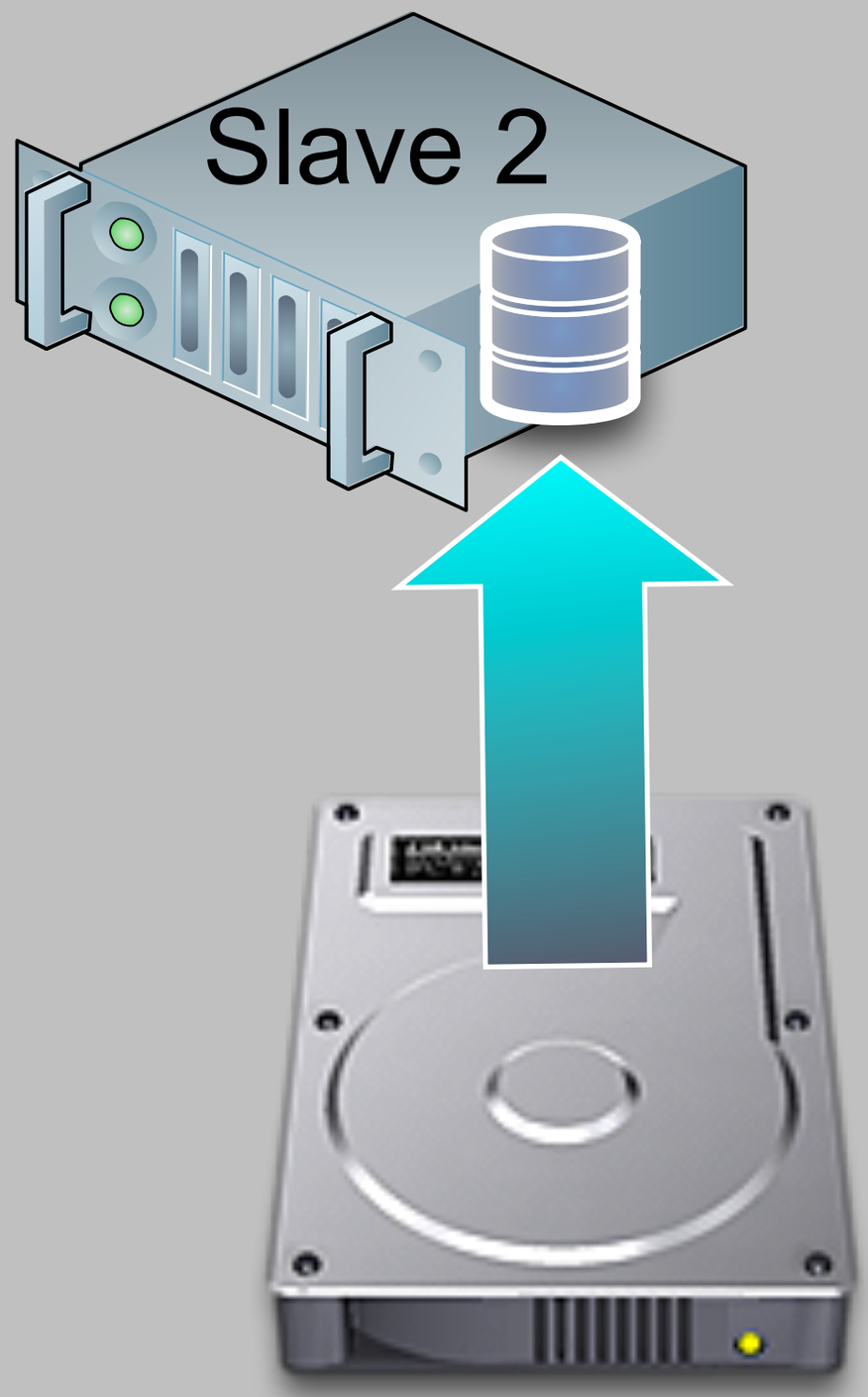
5 INSTALL MySQL on the new slave



Make sure that:

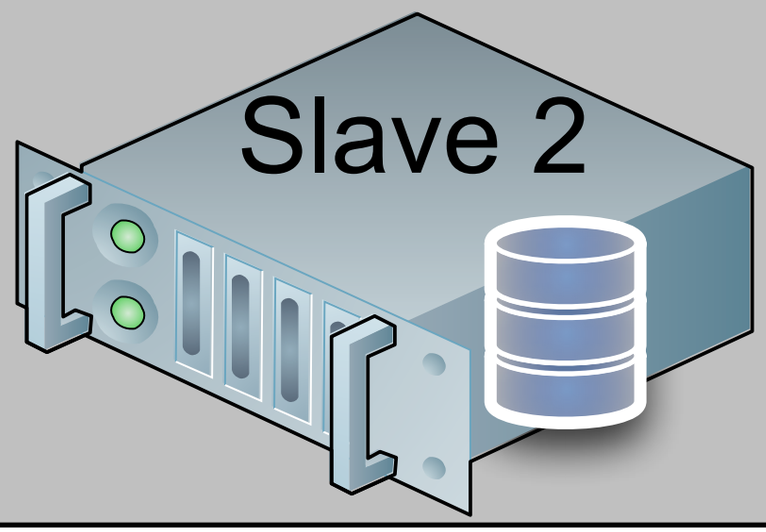
- You're using the same version of MySQL
- You have the same directory structure
- The server is not started yet

6 COPY THE old slave DATA on the slave



7

ENABLE THE NEW SLAVE



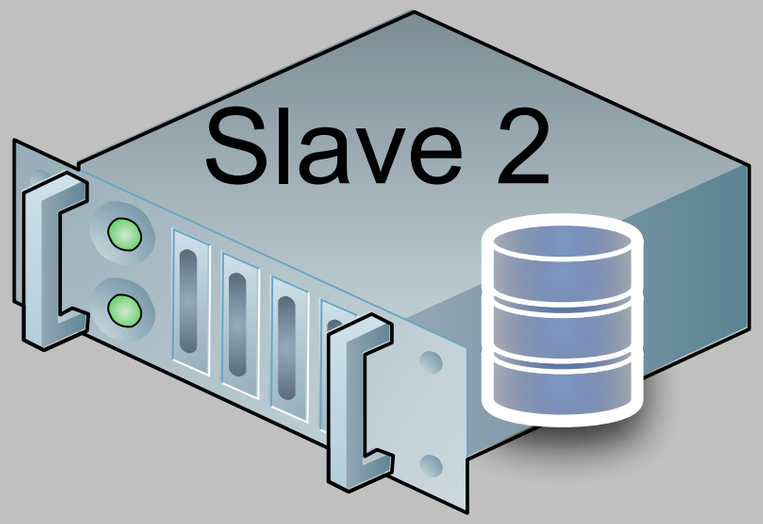
Configuration file

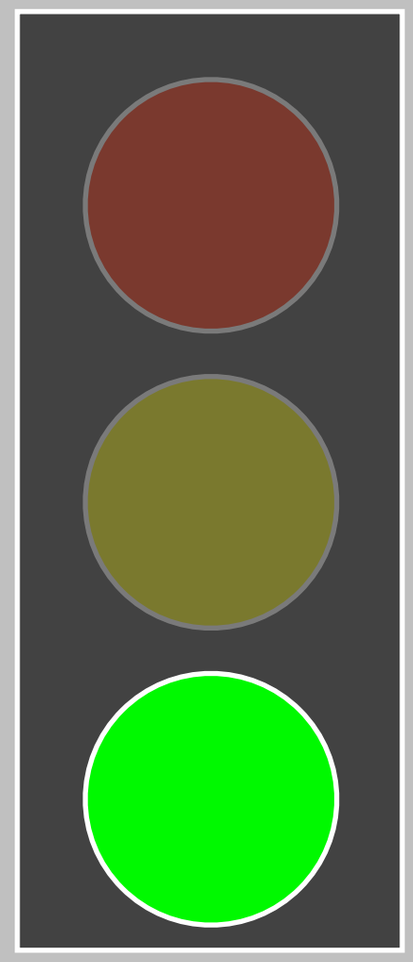
```
[mysqld]
server-id=3
relay-log=mysql-relay
read-only
# optional:
log-bin=mysql-bin
```

must be unique!

8

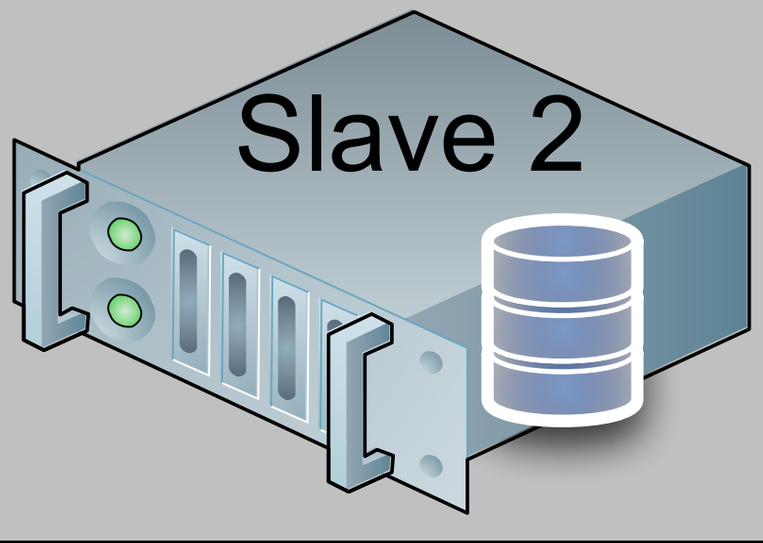
START THE NEW SLAVE





9

CHECK THE SLAVE



SQL command

```
SHOW SLAVE STATUS \G

...
 Slave_IO_Running: Yes
Slave_SQL_Running: Yes
...
```

Why it works

- No need to issue a CHANGE MASTER TO command.
- Because we cloned the old slave
- The new slave gets its parameters from the .info files in the data directory

Starting and stopping replication

Starting and stopping replication

```
START SLAVE;
```

Starting and stopping replication

```
START SLAVE;
START SLAVE SQL_THREAD;
```

Starting and stopping replication

```
START SLAVE;
START SLAVE SQL_THREAD;
START SLAVE IO_THREAD;
```

Starting and stopping replication

```
START SLAVE;
START SLAVE SQL_THREAD;
START SLAVE IO_THREAD;
START SLAVE UNTIL MASTER_LOG_FILE='filename',
MASTER_LOG_POS=xxxx;
```

Starting and stopping replication

```
START SLAVE;
START SLAVE SQL_THREAD;
START SLAVE IO_THREAD;
START SLAVE UNTIL MASTER_LOG_FILE='filename',
MASTER_LOG_POS=xxxx;
STOP SLAVE;
```

Starting and stopping replication

```
START SLAVE;
START SLAVE SQL_THREAD;
START SLAVE IO_THREAD;
START SLAVE UNTIL MASTER_LOG_FILE='filename',
MASTER_LOG_POS=xxxx;
STOP SLAVE;
STOP SLAVE SQL_THREAD;
```

Starting and stopping replication

```
START SLAVE;
START SLAVE SQL_THREAD;
START SLAVE IO_THREAD;
START SLAVE UNTIL MASTER_LOG_FILE='filename',
MASTER_LOG_POS=xxxx;
STOP SLAVE;
STOP SLAVE SQL_THREAD;
STOP SLAVE IO_THREAD;
```

Hijacking the agenda

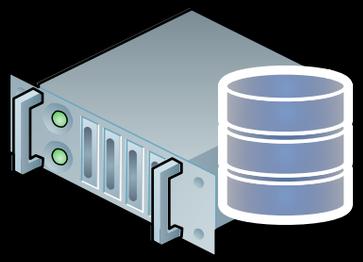
- MySQL Sandbox

MySQL Sandbox

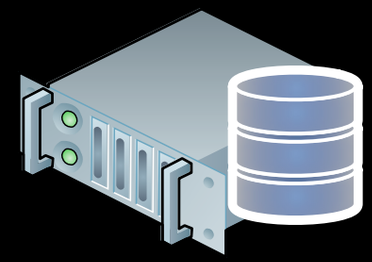
http://mysqlsandbox.net

- Free software (Perl under GPL)
- One (unix) host
- Many database servers
- Single or multiple sandboxes
- Customized scripts to use the servers
- Standard or circular replication
- Installs **IN SECONDS**

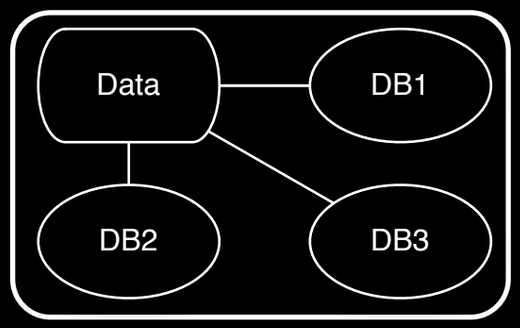
overview



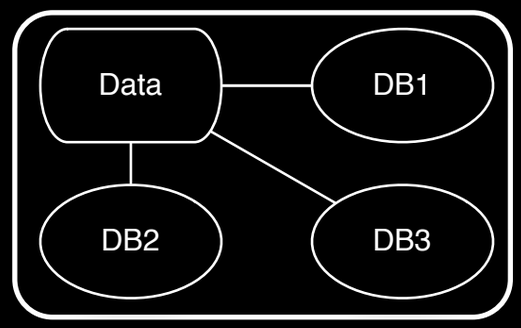
MySQL
server



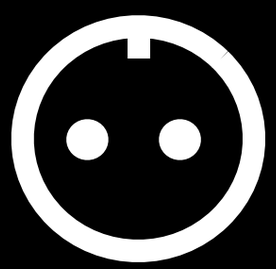
MySQL
server



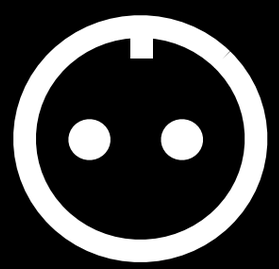
DATA DIRECTORY



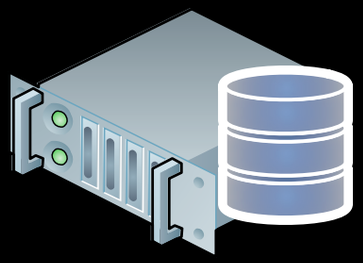
PORT



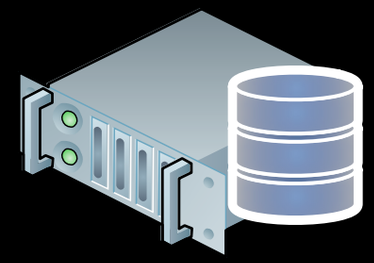
SOCKET



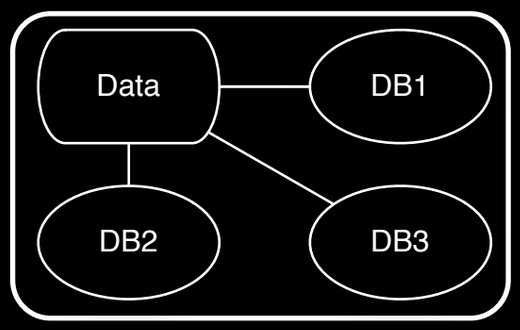
overview



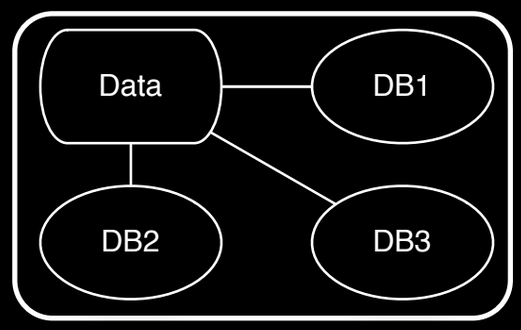
MySQL
server



MySQL
server



SAME
DATA
DIRECTORY?

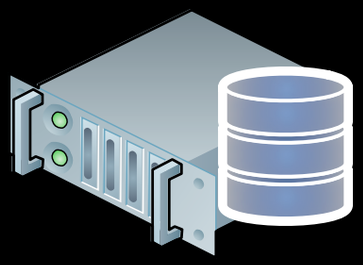


`/var/lib/mysql`

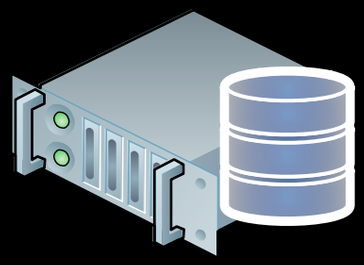
`/var/lib/mysql`

DATA CORRUPTION

overview

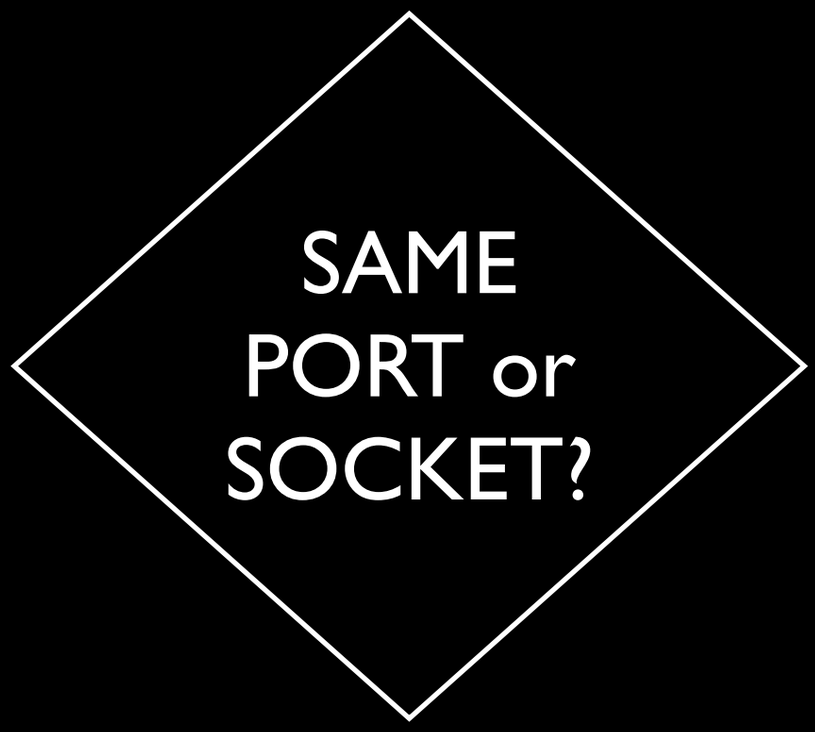


MySQL
server

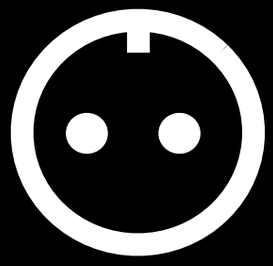


MySQL
server

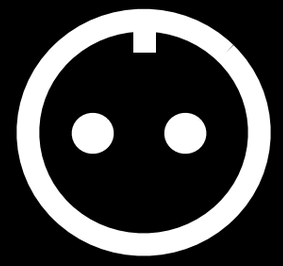
3306



3306



/tmp/mysql.sock



/tmp/mysql.sock

DOES NOT START

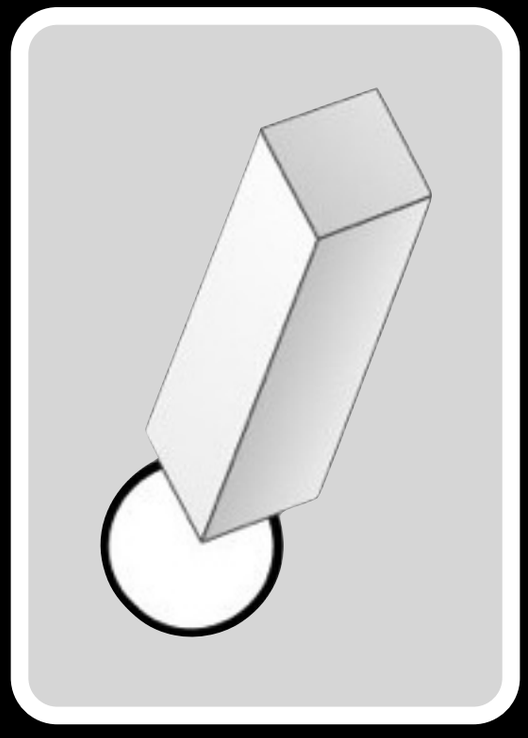
The hard way

The hard way

Read the manual

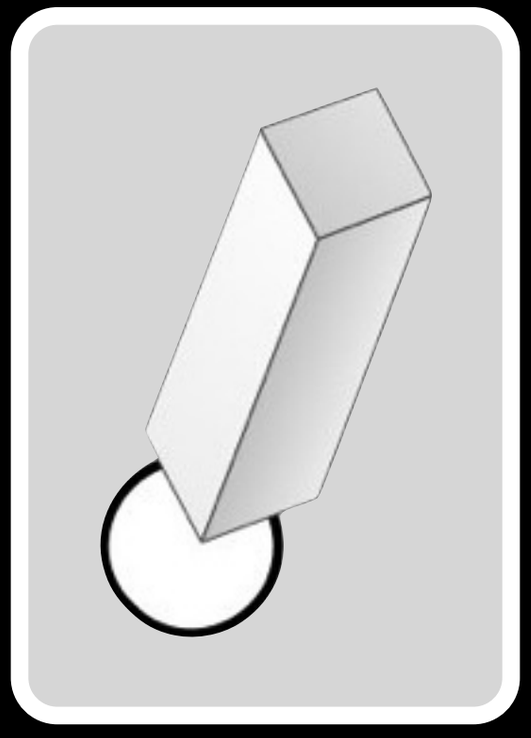
The hard way

Read the manual → **try to figure out what to change**



The hard way

Read the manual → **try to figure out what to change** → **Install**



The easy way

MySQL Sandbox

```
$ make_sandbox \
    /path/to/mysql-5.1.54_linux.tar.gz

# it should work always
```

The easier way

Prepare once

Install many times

```
# some
# preliminary
# work
```

```
$ make_sandbox 5.1.54
```

The easiest way

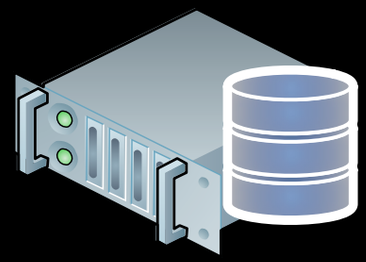
Prepare once

Install many times

```
# some
# preliminary
# work
```

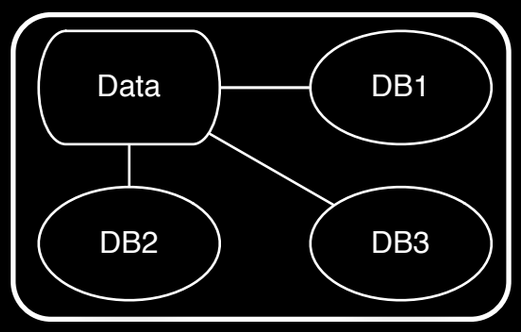
```
$ sb 5.1.54
```

MySQL Sandbox



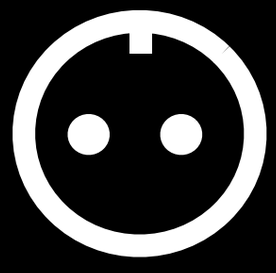
MySQL
server

VERSION



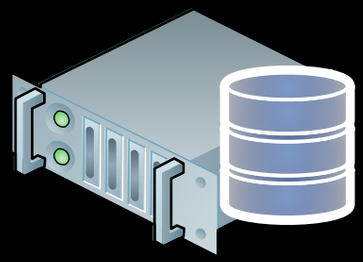
$SANDBOX_HOME/msb_VERSION/data

VERSION



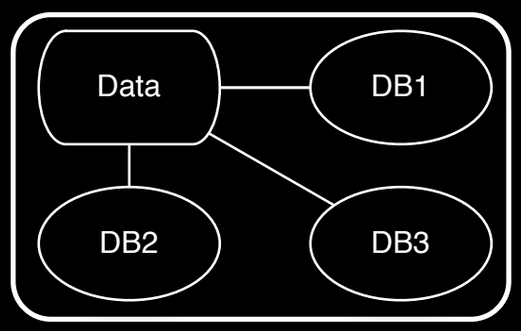
/tmp/mysql_VERSION.sock

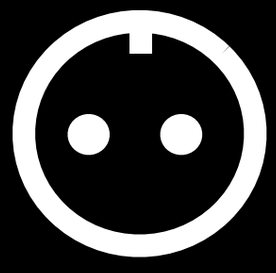
MySQL Sandbox



MySQL
server

5.1.54



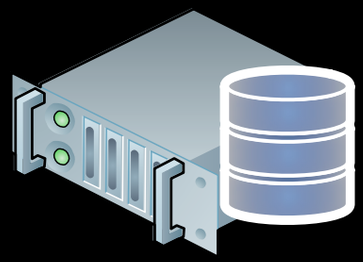


$SANDBOX_HOME/msb_5_1_54/data

5154

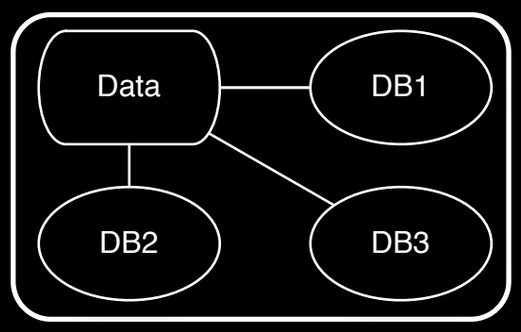
/tmp/mysql_5154.sock

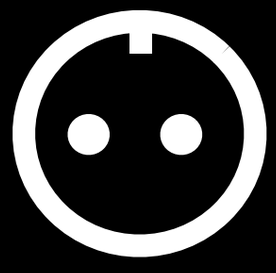
MySQL Sandbox



MySQL
server

5.5.9

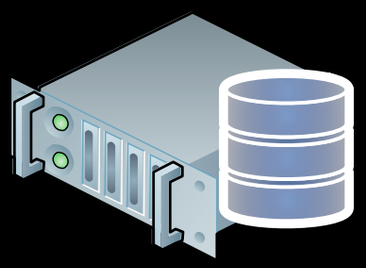




$SANDBOX_HOME/msb_5_5_09/data

5509

/tmp/mysql_5509.sock

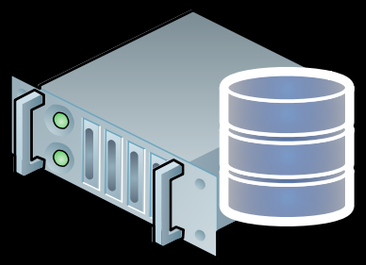


MySQL
server

Single Sandbox

customized scripts

```
start
stop
restart
status
clear
send_kill
use
```



MySQL
server

Multiple Sandbox

customized scripts

```
start_all
stop_all
restart_all
status_all
clear_all
send_kill_a
ll
use_all
```

m	n1
s1	n2
s2	n3

Where do you get it

- from CPAN

```
sudo su -
cpan MySQL::Sandbox
```

- from launchpad

```
http://launchpad.net/mysql-sandbox
```

The easy replication way

MySQL Sandbox

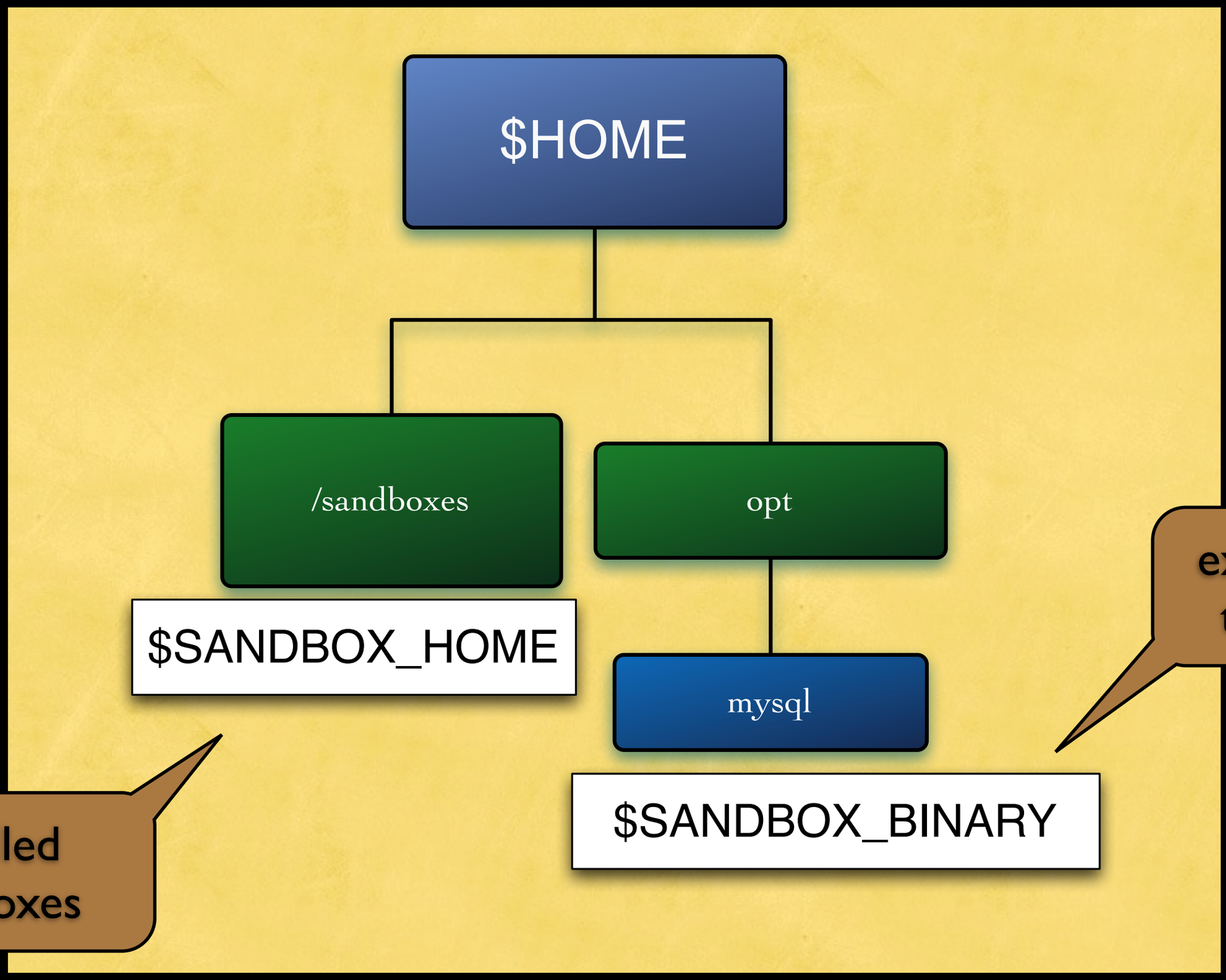
```
$ make_replication_sandbox \
    /path/to/mysql-5.1.54_linux.tar.gz
```

Prepare once

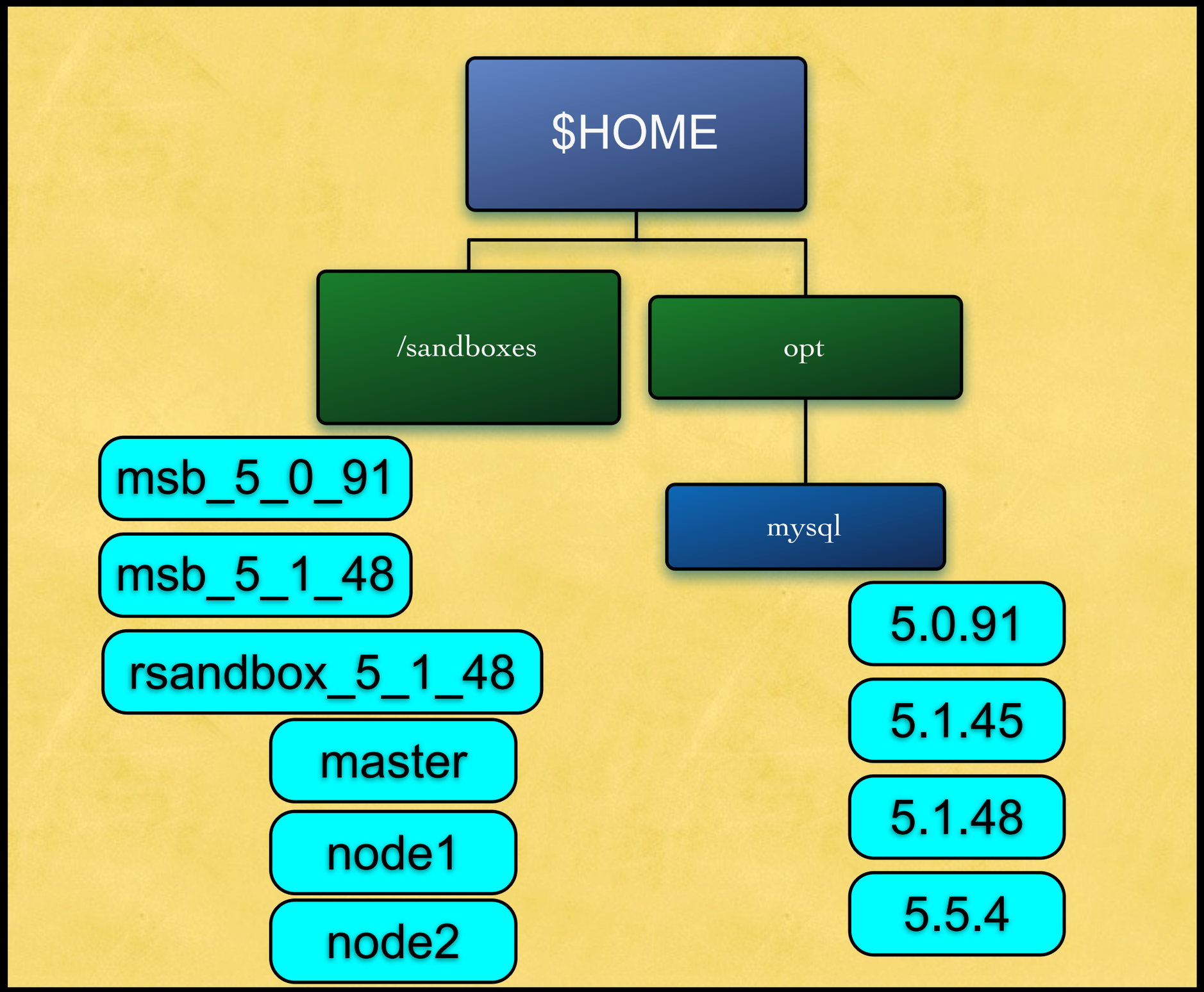
Install many times

```
# some                $ make_replication_sandbox
# preparation         5.1.54
```

default architecture



default architecture




COOKBOOK

creating a single sanbox

```
make_sandbox \
    /path/to/mysql-X.X.XX-OS.tar.gz
```

using a single sanbox

```
# after
# make_sandbox \
#   /path/to/mysql-X.X.XX-OS.tar.gz

$ cd $SANDBOX_HOME/msb_X_X_XX
$ ./use
```

creating a single sanbox with a specific options file

```
make_sandbox \
    /path/to/mysql-X.X.XX-OS.tar.gz \
    --my_file=/path/to/my.cnf
```

easily create a sandbox after the first one

The long way

```
$ cd $HOME/opt/mysql      # $SANDBOX_BINARY
$   gunzip -c \
  /path/to/mysql-5.1.34-osx10.5-x86.tar.gz \
   | tar -xf -
$ mv mysql-5.1.34-osx10.5-x86 5.1.34
$ make sandbox 5.1.34
```

easily create a sandbox after the first one

The short way

```
$ make_sandbox \
  path/to/mysql-5.1.34-osx10.5-x86.tar.gz \
  --export_binaries
```

starting a single sanbox

```
$ cd $SANDBOX_HOME/msb_X_X_XX
$ ./start
```

starting a single sanbox with temporary options

```
$ cd $SANDBOX_HOME/msb_X_X_XX
$ ./start --option=value
```

```
$ ./restart --option=value
```

```
$ ./start --key-buffer=20000000
```

creating a sandbox with custom port and directory

```
$ make_sandbox 5.1.34 \
    --sandbox_port=7800 \
    --sandbox_directory=mickeymouse
```

creating a sandbox with automatic port checking

```
$ make_sandbox 5.1.34 --check_port


# if 5.1.34 is free
#     port=5134
#     directory=msb_5_1_34
# else
#     port=5135 (or the first free)
#     directory=msb_5_1_34_a
```

create a replication sandbox

```
$ make_replication_sandbox \
  path/to/mysql-5.1.34-osx10.5-x86.tar.gz
```

create a circular replication sandbox

```
$ make_replication_sandbox \
  --circular=4 \
  path/to/mysql-5.1.34-osx10.5-x86.tar.gz
```

changing port to an existing sandbox

```
$ sbtool -o port \
    -s /path/to/source/sandbox \
    --new_port=XXXX
```

installing the innodb plugin

```
$ sbtool -o plugin \
    --plugin=innodb \
    -s /path/to/source/sandbox
```

creating a replication sandbox with new base port

```
$ make_replication_sandbox \
    --replication_directory=newwdir \
    --check_base_port 5.0.79

# Creates a replication directory under
# $SANDBOX_HOME/newdir
# The previous one is preserved.
# No conflicts happen
```

How to set up replication

DEMO

AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• **Binary log formats** • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

Binary log formats

- **Statement** based replication
  - default
  - available since 3.23

- **Row** based replication
  - introduced in 5.1
  - ROW or MIXED

Viewing the binary logs

```
# statement based replication

$ mysqlbinlog binary-log-name

# or, as a SQL command
SHOW BINLOG EVENTS IN 'binary-log-name';
```

```
# row based replication

$ mysqlbinlog --verbose \
  --base64-output=decode-rows binary-log-name
```

binary log examples

```
# statement based replication

mysqlbinlog binary-log-name


# row based replication

mysqlbinlog --verbose \
  --base64-output=decode-rows binary-log-name
```

AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • **What gets replicated and how** • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

what gets replicated

What	Statement-based	row-based

what gets replicated

What	Statement-based	row-based
insert/update/delete	statement	affected records

what gets replicated

What	Statement-based	row-based
insert/update/delete	statement	affected records
schema/table/view creation/drop	statement	statement

what gets replicated

What	Statement-based	row-based
insert/update/delete	statement	affected records
schema/table/view creation/drop	statement	statement
stored routine/trigger creation/drop	statement	statement

95

what gets replicated

What	Statement-based	row-based
insert/update/delete	statement	affected records
schema/table/view creation/drop	statement	statement
stored routine/trigger creation/drop	statement	statement
stored procedure call	statements executed inside SP	affected records

what gets replicated

What	Statement-based	row-based
insert/update/delete	statement	affected records
schema/table/view creation/drop	statement	statement
stored routine/trigger creation/drop	statement	statement
stored procedure call	statements executed inside SP	affected records
stored function call	function call	affected records

what gets replicated

What	Statement-based	row-based
insert/update/delete	statement	affected records
schema/table/view creation/drop	statement	statement
stored routine/trigger creation/drop	statement	statement
stored procedure call	statements executed inside SP	affected records
stored function call	function call	affected records
trigger execution	none: the slave runs the trigger	affected records moved. No trigger runs

what gets replicated

What	Statement-based	row-based
insert/update/delete	statement	affected records
schema/table/view creation/drop	statement	statement
stored routine/trigger creation/drop	statement	statement
stored procedure call	statements executed inside SP	affected records
stored function call	function call	affected records
trigger execution	none: the slave runs the trigger	affected records moved. No trigger runs
event creation	slaveside disabled event	slaveside disabled event

what gets replicated

What	Statement-based	row-based
insert/update/delete	statement	affected records
schema/table/view creation/drop	statement	statement
stored routine/trigger creation/drop	statement	statement
stored procedure call	statements executed inside SP	affected records
stored function call	function call	affected records
trigger execution	none: the slave runs the trigger	affected records moved. No trigger runs
event creation	slaveside disabled event	slaveside disabled event
event execution	statement	affected records

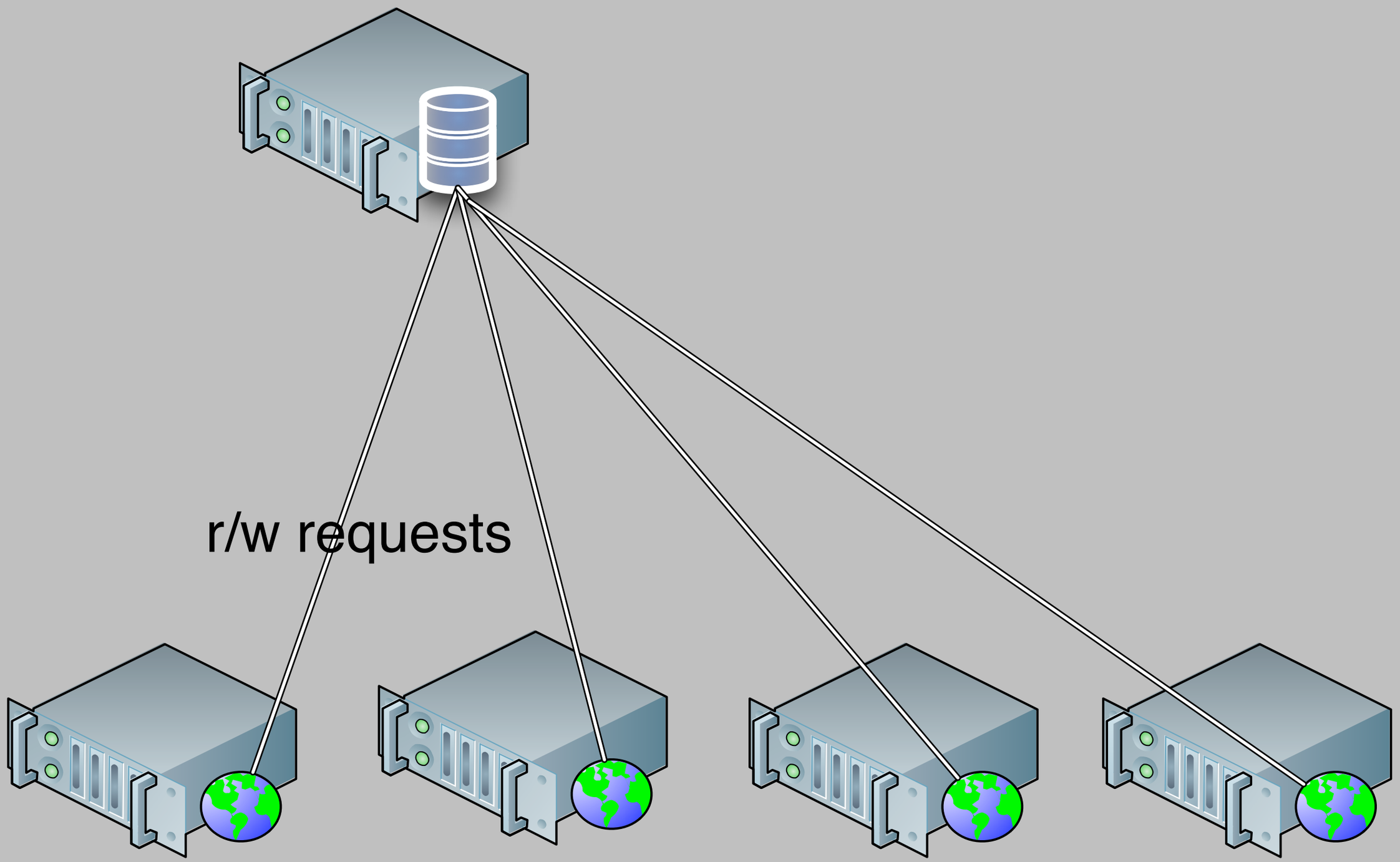
Binary logs in practice

DEMO

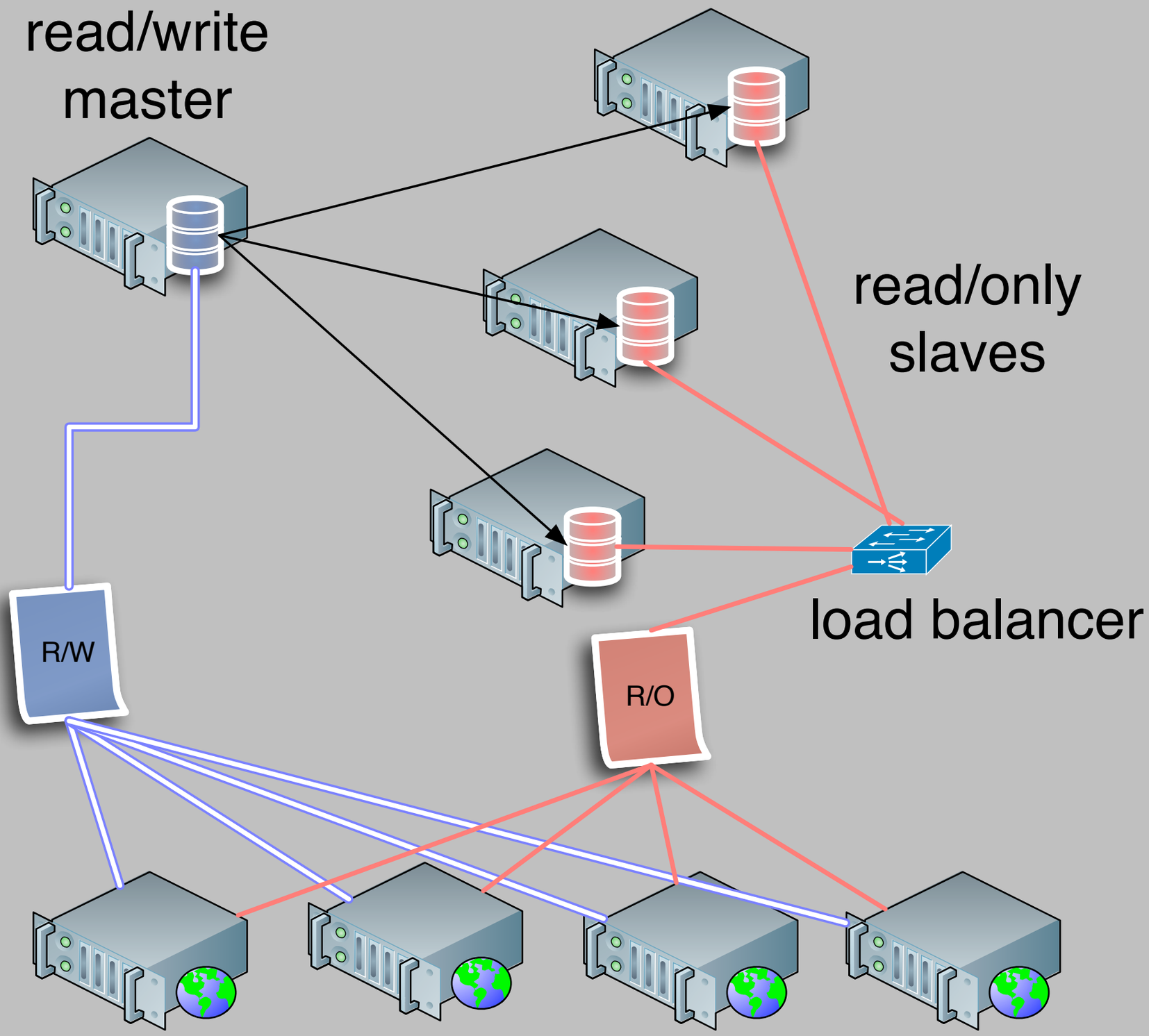
AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • **Replication awareness**
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

From single server application



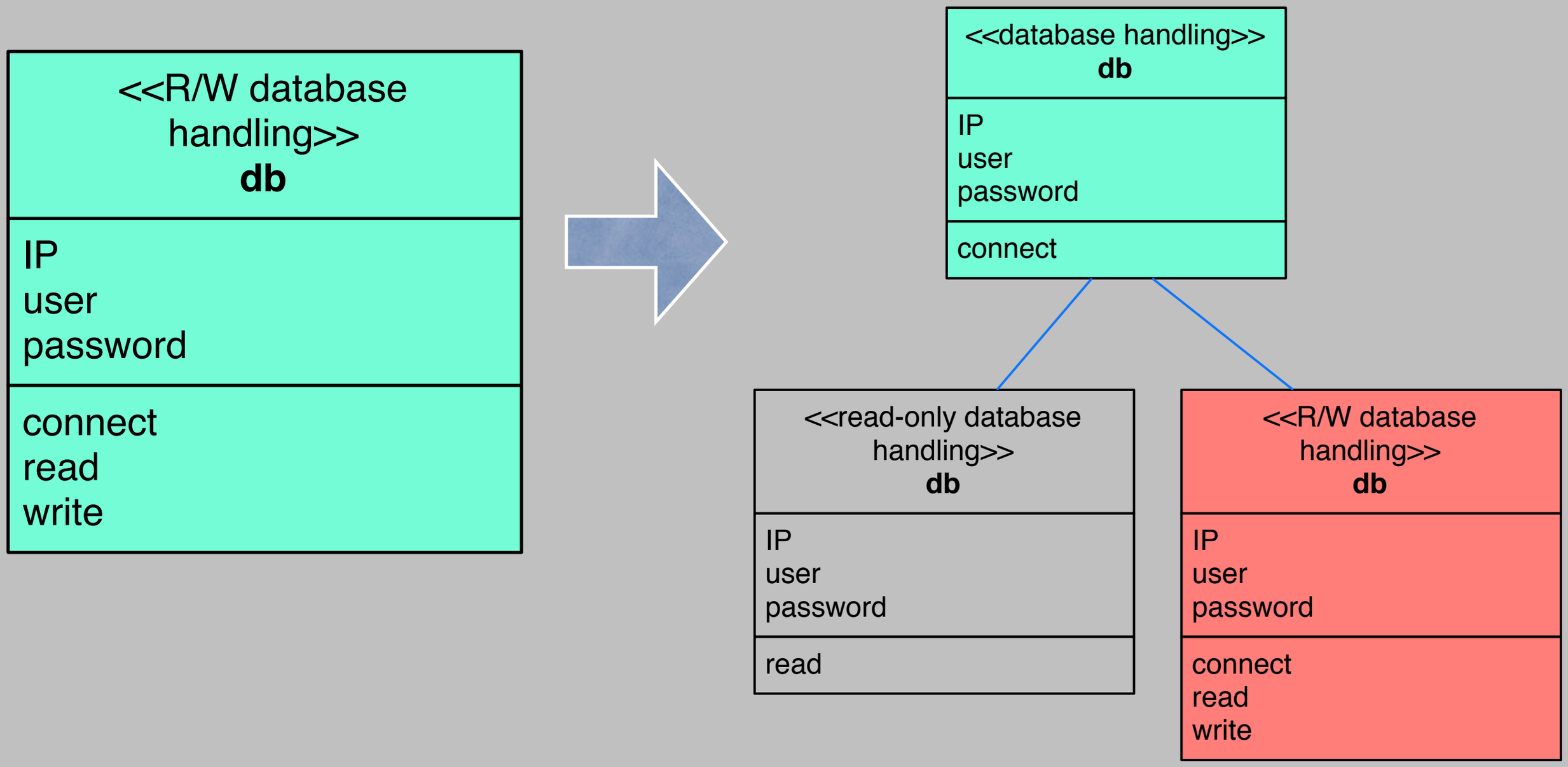
To replication-aware application



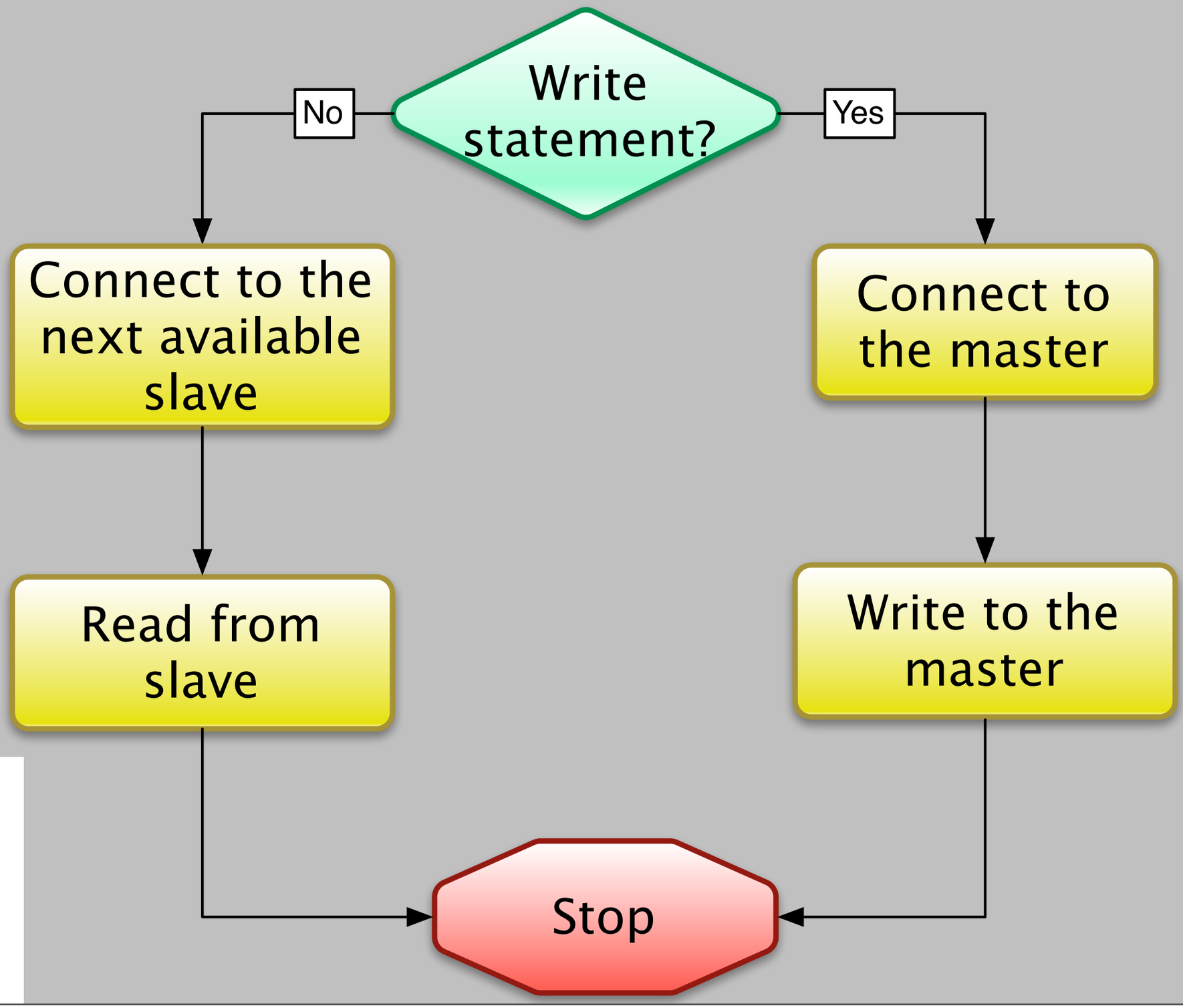
Single server application

```
$link = mysql_connect(
 $server_IP,
 'mysql_user',
 'mysql_password'
);
$result = mysql_query(
'INSERT INTO table_name (x) VALUES (1)',
$link
);
$result = mysql_query(
'SELECT * FROM table_name WHERE x=1',
$link
);
```

Making an application aware of replication



Using replication: the WRONG way

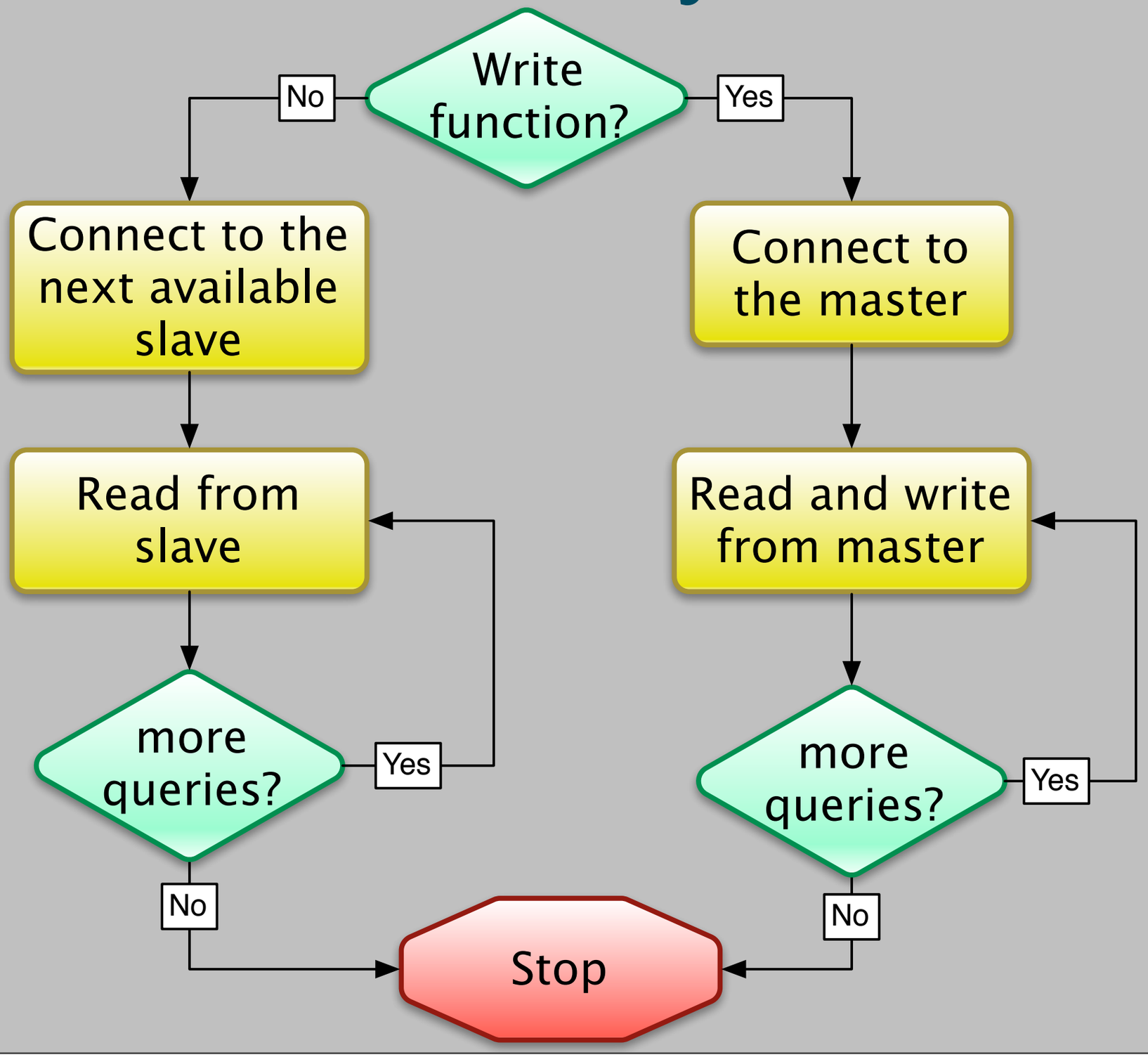


R/W split by statement

Why statement split is wrong

- Breaks transactions
- High risk of inconsistency
- Loses or corrupts data

Using replication: the RIGHT way



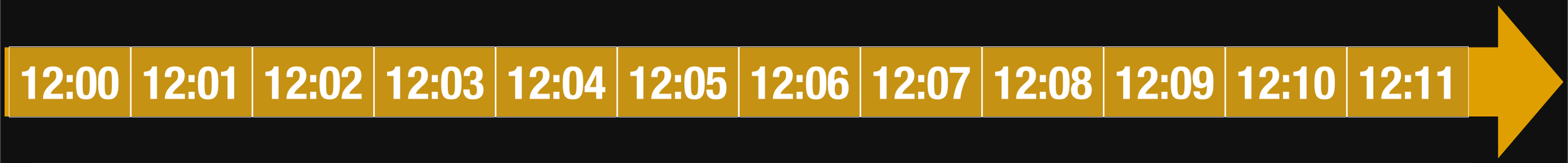
R/W split by function

Replication is single threaded

- Let's assume you have two queries on the master
- one query takes 3 minutes to complete
- the other takes 5 minutes
- Both start at 12:00 noon

Single thread replication

MASTER



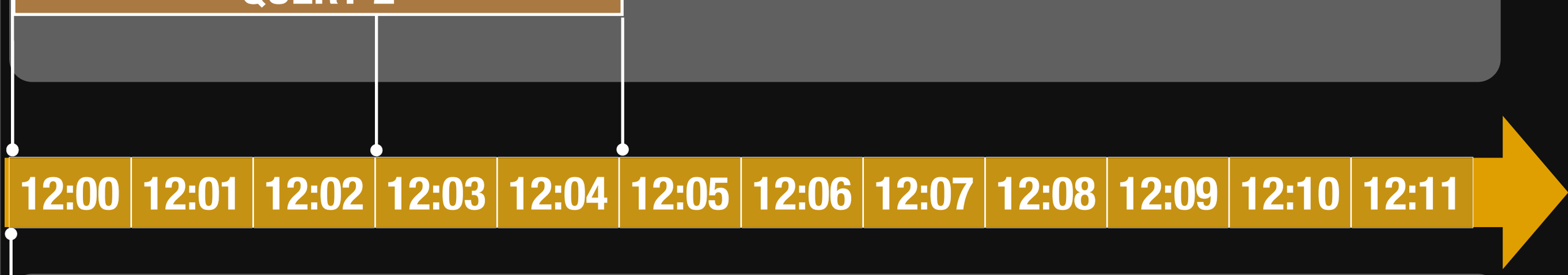
SLAVE

Single thread replication

MASTER

QUERY 1

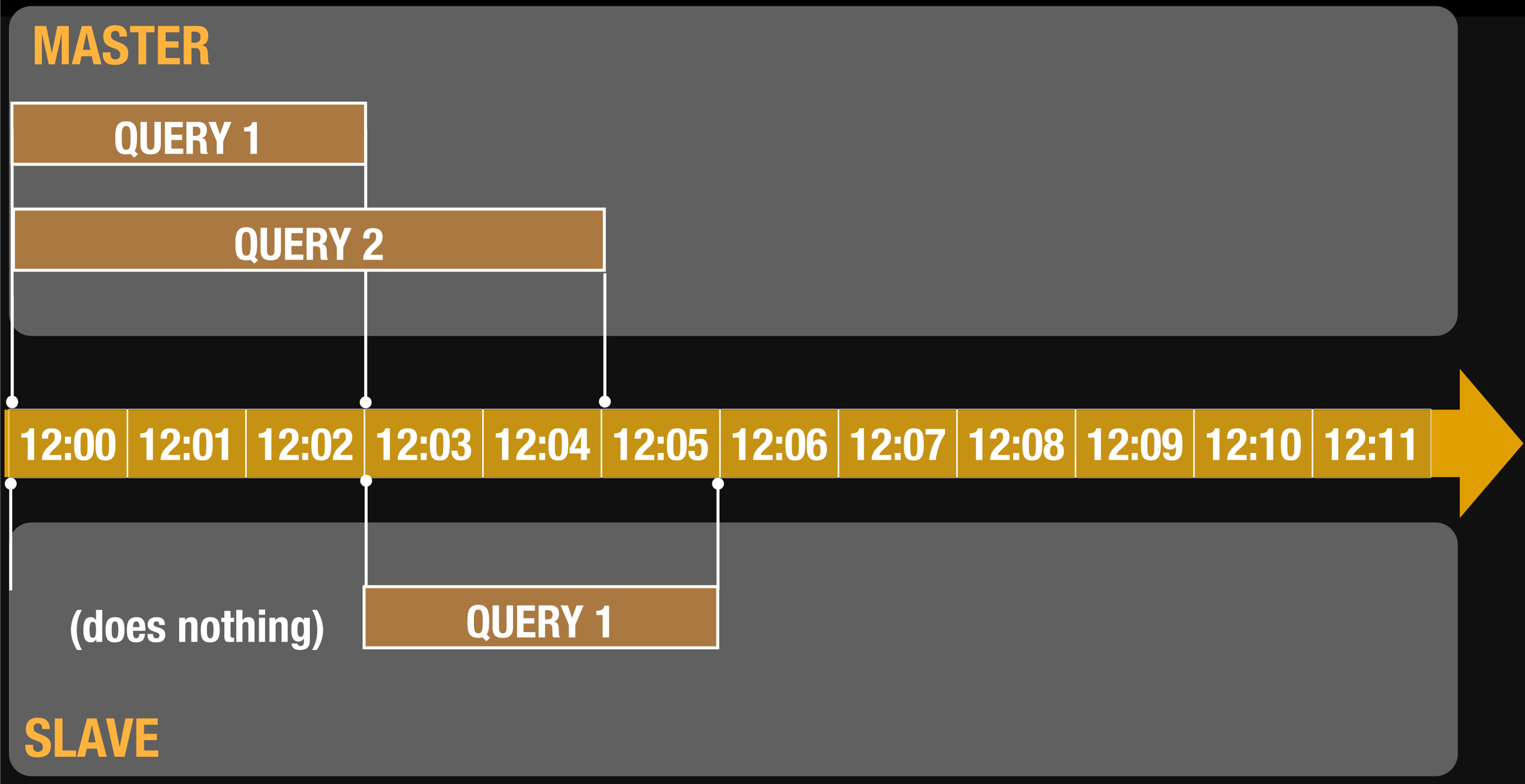
QUERY 2



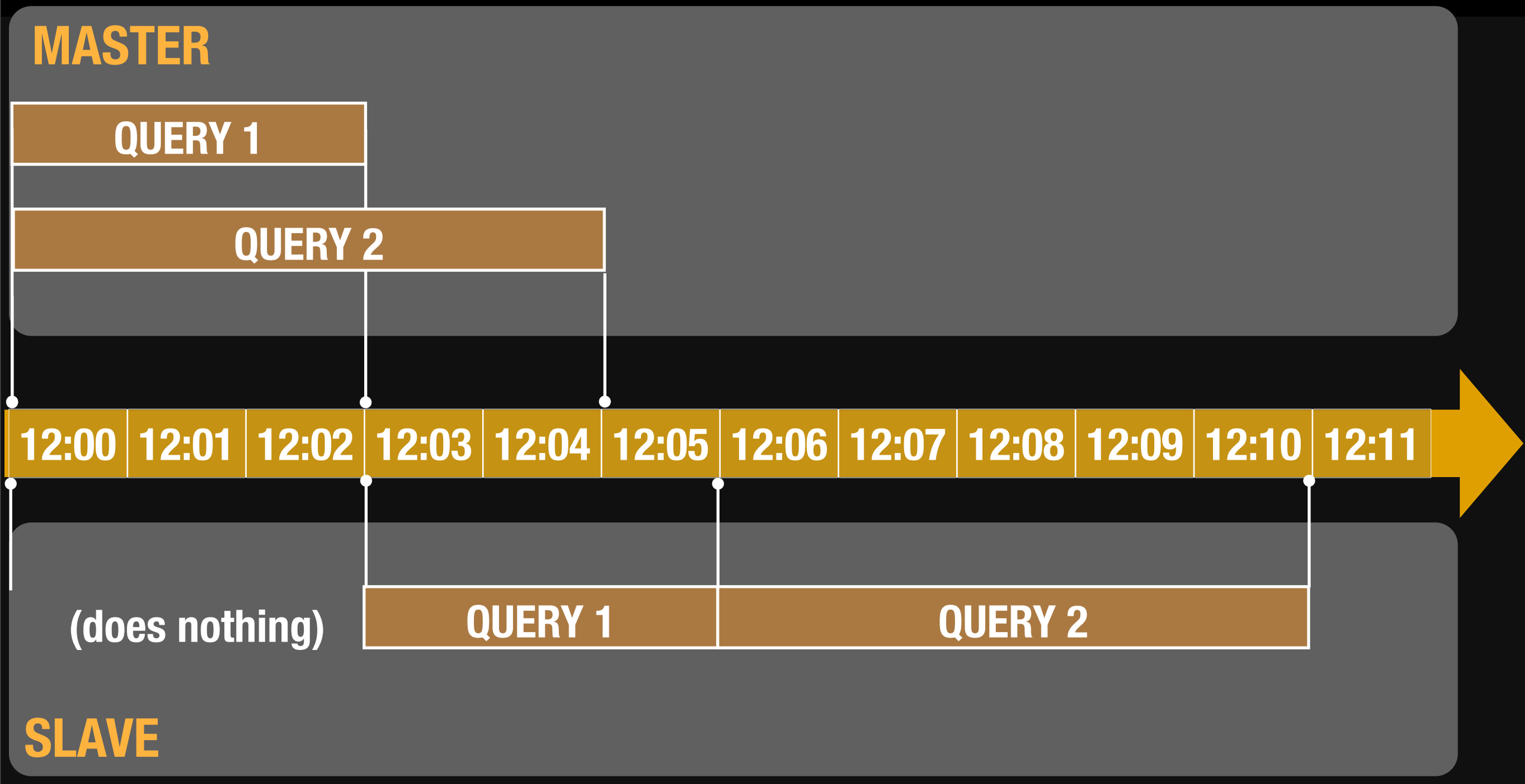
(does nothing)

SLAVE

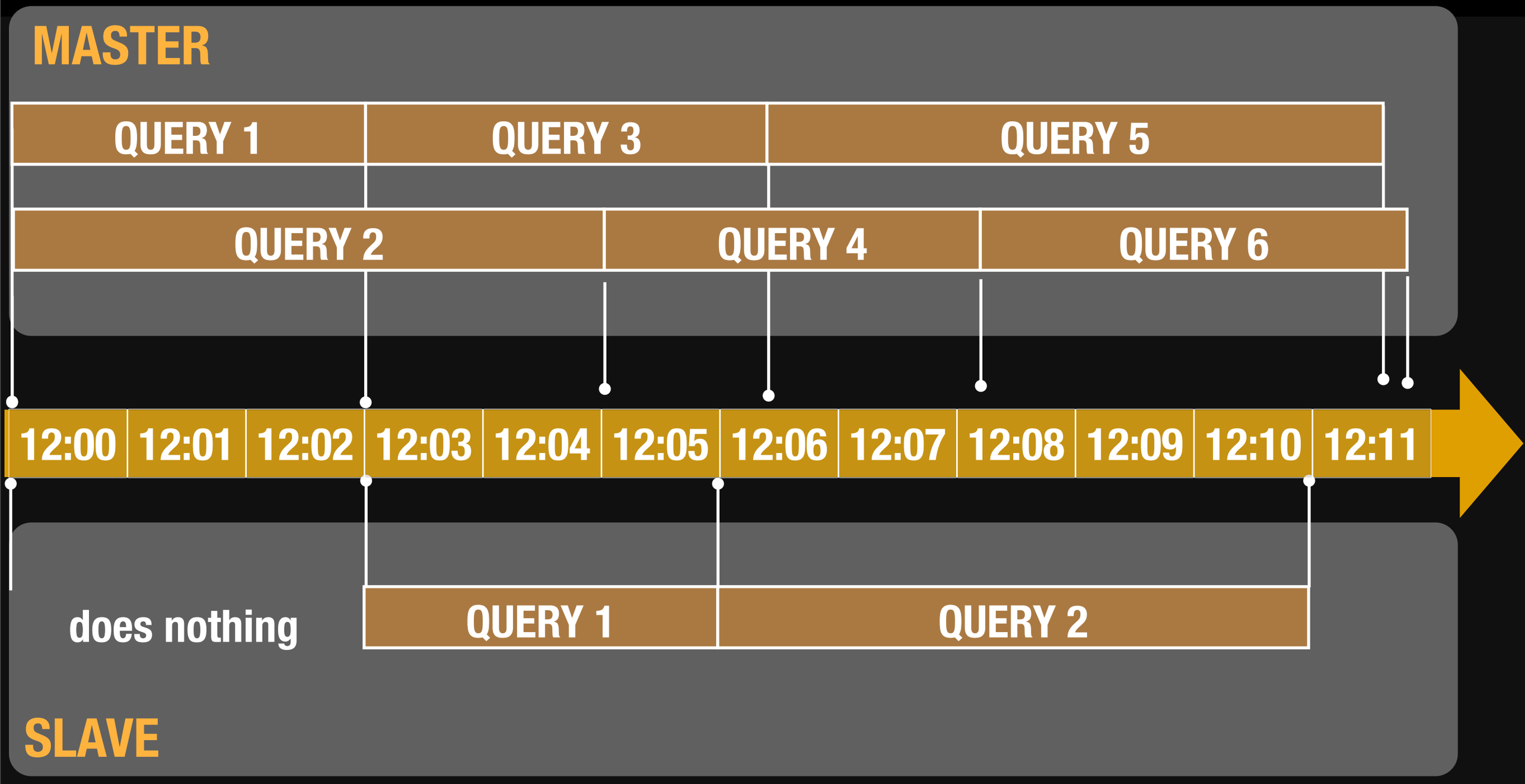
Single thread replication



Single thread replication



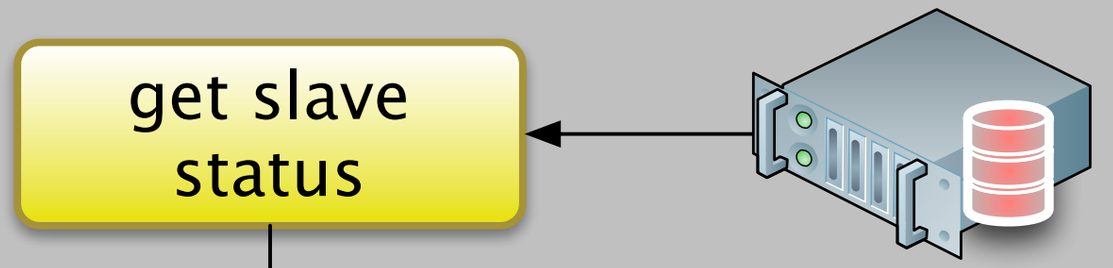
Single thread replication

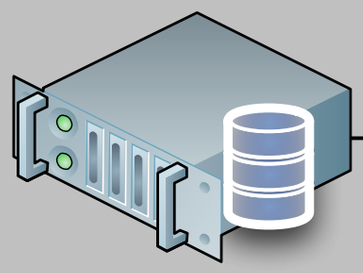


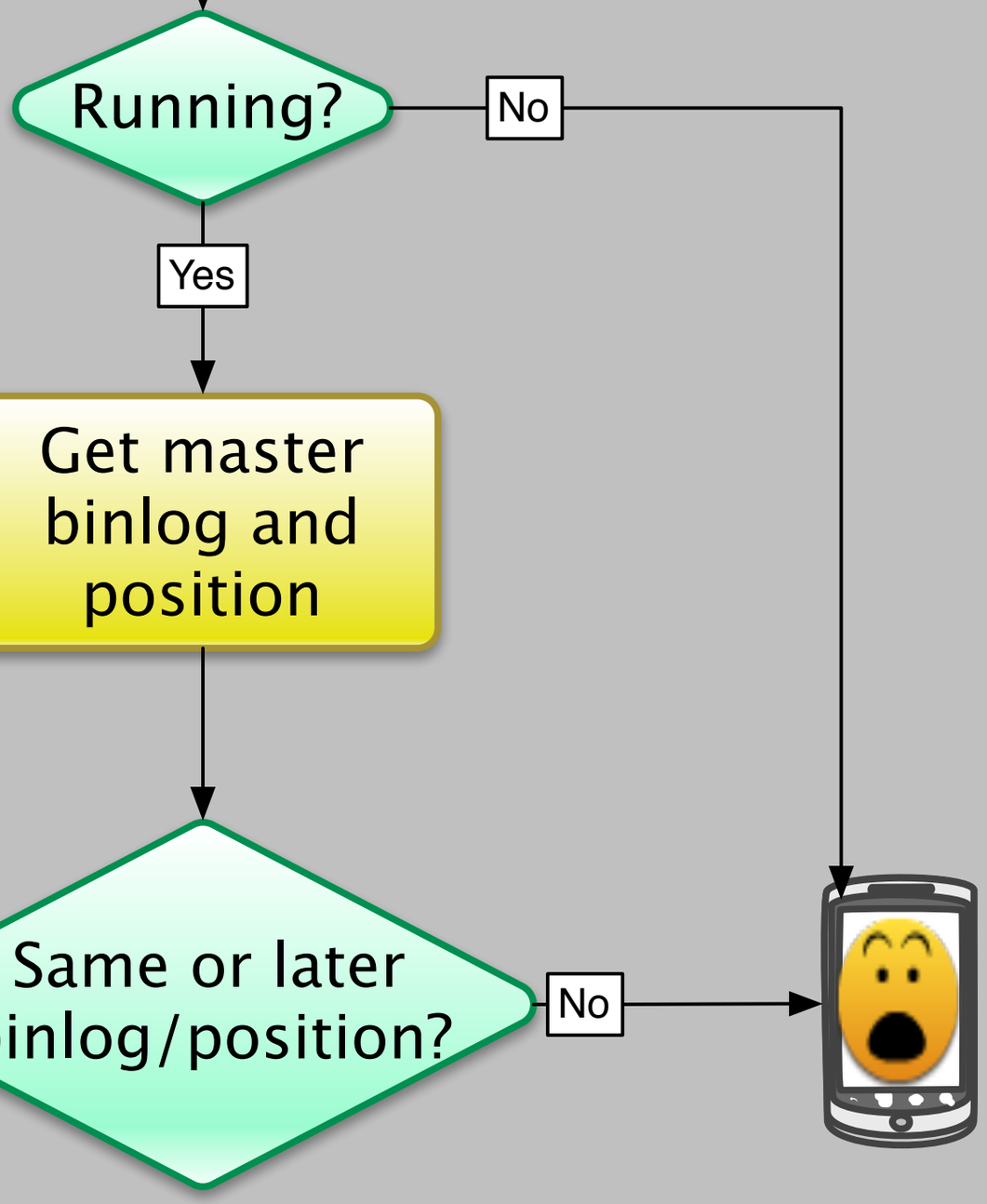
AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• **Monitoring** • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

Sample monitoring







commands for checking replication

```
master> SHOW MASTER STATUS
master> SHOW PROCESSLIST

slave> SHOW SLAVE STATUS
slave> SHOW PROCESSLIST
```

replication user seen in the master

```
master > show processlist\G
*************************** 1. row ***************
     Id: 2
   User: rsandbox
   Host: localhost:57011
     db: NULL
Command: Binlog Dump
   Time: 81625
  State: Master has sent all binlog to slave; waiting
for binlog to be updated
   Info: NULL
[...]
```

IO-thread in the slave

```
slave > show processlist\G
*************************** 1. row ***************
     Id: 2
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 124801
  State: Waiting for master to send event
   Info: NULL
[...]
```

SQL thread in the slave

```
slave > show processlist\G
[...]
*************************** 2. row ***************
     Id: 3
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 124712
  State: Slave has read all relay log; waiting for the
slave I/O thread to update it
   Info: NULL
[...]
```

monitoring replication

```
master> SHOW MASTER STATUS

slave> SHOW SLAVE STATUS
```

FULL SCRIPTS:

http://datacharmer.blogspot.com/2011/04/refactored-again-poor-mans-mysql.html

http://forge.mysql.com/tools/tool.php?id=6

show master status

```
            File: mysql-bin.000002
        Position: 78045744
    Binlog_Do_DB:
Binlog_Ignore_DB:
```

show slave status

```
Slave_IO_State: Waiting for master to send event
   Master_Host: 127.0.0.1
   Master_User: rsandbox
   Master_Port: 27371
  Connect_Retry: 60
Master_Log_File: mysql-bin.000002
Read_Master_Log_Pos: 78045744
Relay_Log_File: mysql_sandbox27372-relay-bin.000055
Relay_Log_Pos: 78045889
Relay_Master_Log_File: mysql-bin.000002
   Slave_IO_Running: Yes
  Slave_SQL_Running: Yes
```

show slave status

```
...
            Replicate_Do_DB:
        Replicate_Ignore_DB:
         Replicate_Do_Table:
     Replicate_Ignore_Table:
    Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
 ...
  . . .
```

show slave status

```
...
                Last_Errno: 0
                Last_Error:
              Skip_Counter: 0
       Exec_Master_Log_Pos: 78045744
           Relay_Log_Space: 78046100
...
      Seconds_Behind_Master: 0
...
               Last_IO_Errno: 0
               Last_IO_Error:
              Last_SQL_Errno: 0
              Last_SQL_Error:
```

AGENDA

Breaking (and fixing) replication

DEMO

AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • **Log management** • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

Logs rotation

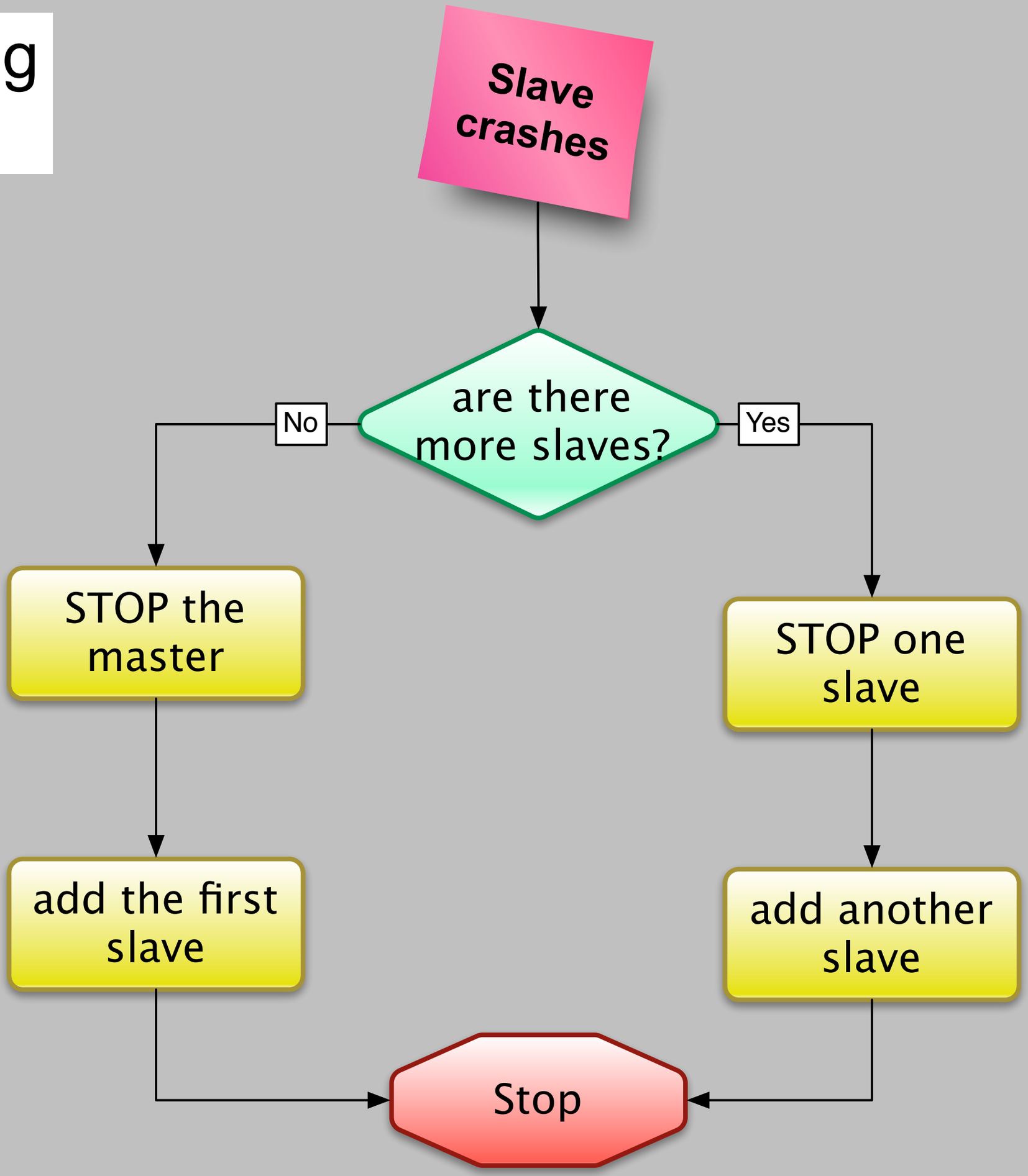
```
# server variables

max-binlog-size
expire-log-days

# logs commands
```

```
SHOW MASTER LOGS;
PURGE MASTER LOGS TO 'filename';
FLUSH [BINARY] LOGS;
```

AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • **Replacing a slave** • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

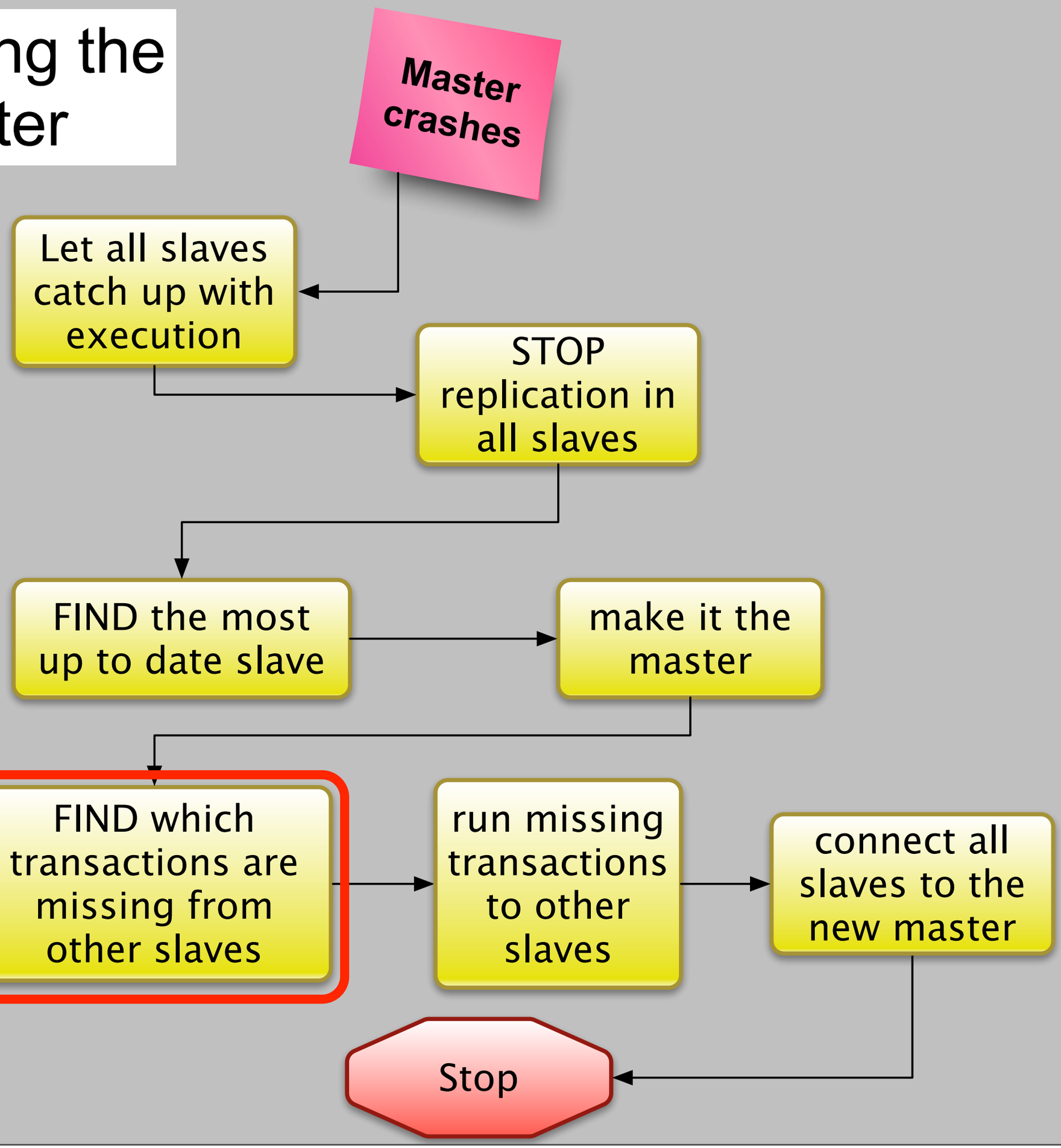
Replacing a slave



AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • **Replacing a master**
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

Replacing the master



Planned master switch

- Stop accepting writes
- Wait until all slaves have caught up

- Stop replication in all slaves
- Promote a slave to master
- Point all slaves to the new master

Changing a failed master

- Pre-requisite:
- log_bin and log_slave_updates must be enabled in all the slaves
- If not, there is more manual labor

Changing a failed master (1)

- Wait until all slaves have caught up
- Identify the most advanced slave
- Make that slave the new master
- ... so far, so good

Changing a failed master (2)

- For each remaining slave:
- Find the missing statements

- Find the **LAST** statement replicated by the slave
- Find the same statement in the new master binlog (*)
- get the position of the **NEXT** statement

(*) if log_slave_updates was not enabled, you need to convert the relay log statements to SQL and do the next step manually

Changing a failed master (3)

- For each remaining slave:
- Apply the missing statements
  - CHANGE MASTER TO master_host=”new_master_hostname”, master_port=new_master_port, master_log_file=”mysql-bin.xxxxxx”, master_log_pos=YYYY

Reasons for complexity

- No global transaction ID

What is a global transaction ID

- A unique identifier of a transaction
- Unique for the whole cluster, not for each node
- Generated by the ultimate source (the master)
- Does not change when the transaction goes through an intermediate master

Why should you care

- Failure recovery
- MySQL DOES NOT have it
- What can you do? Either wait for MySQL 5.6 or use Tungsten Replicator.

AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• **master replacement** • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

Slave = master replacement

- **if**:
  - there are no filters;
  - you are monitoring replication
  - you make sure data is consistent

Slave != backup

- There is no replacement for a good backup.
- Data loss due to a mistake in the master will propagate to the slaves.
- Disasters are always smarter than replication.
- Do your backups!
- Session on Wednesday at 12pm: **"Be a Data Management Hero with Good Backups!"**

AGENDA

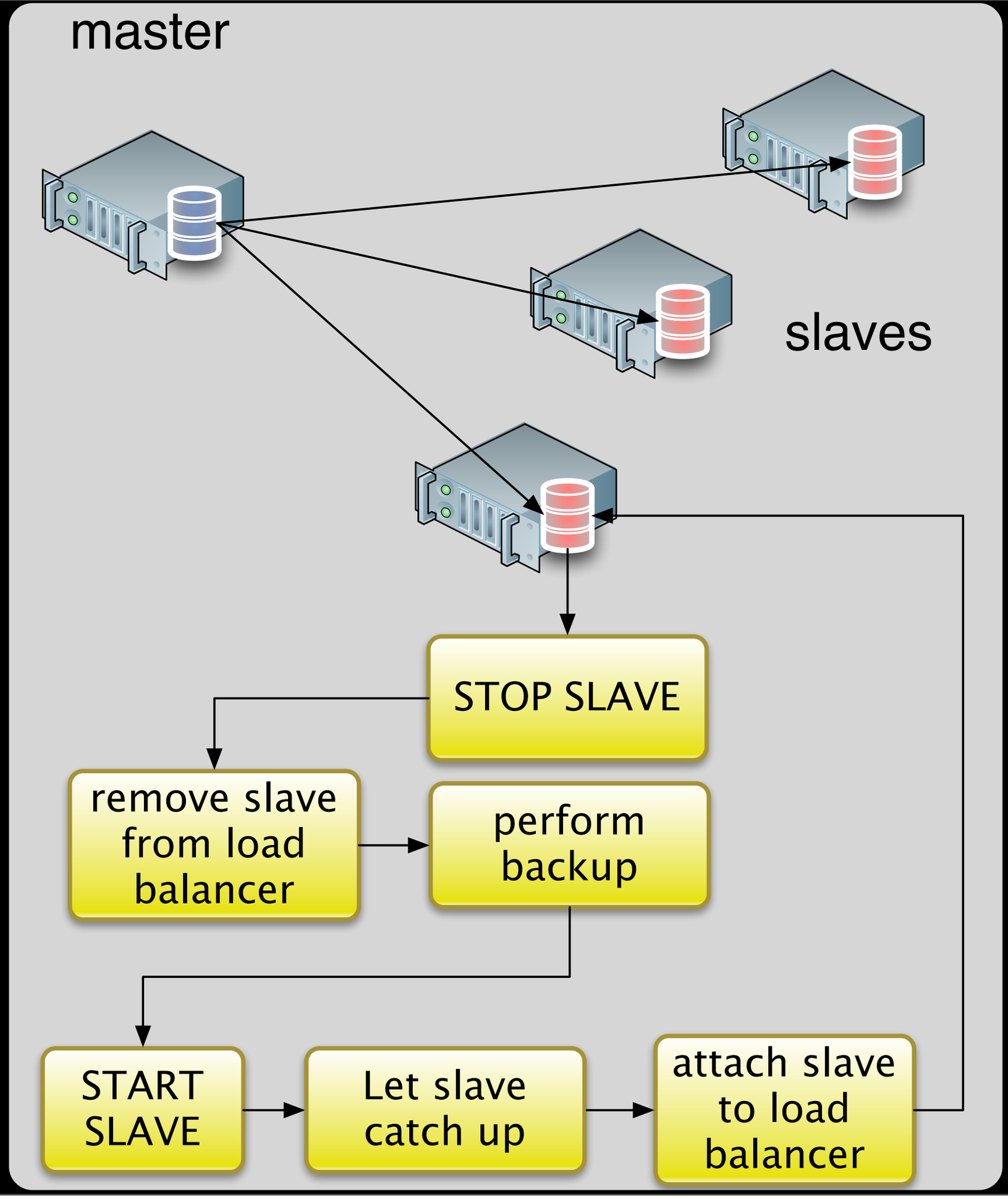
Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • **backup** • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

Replication

=

let the slave do the dirty work

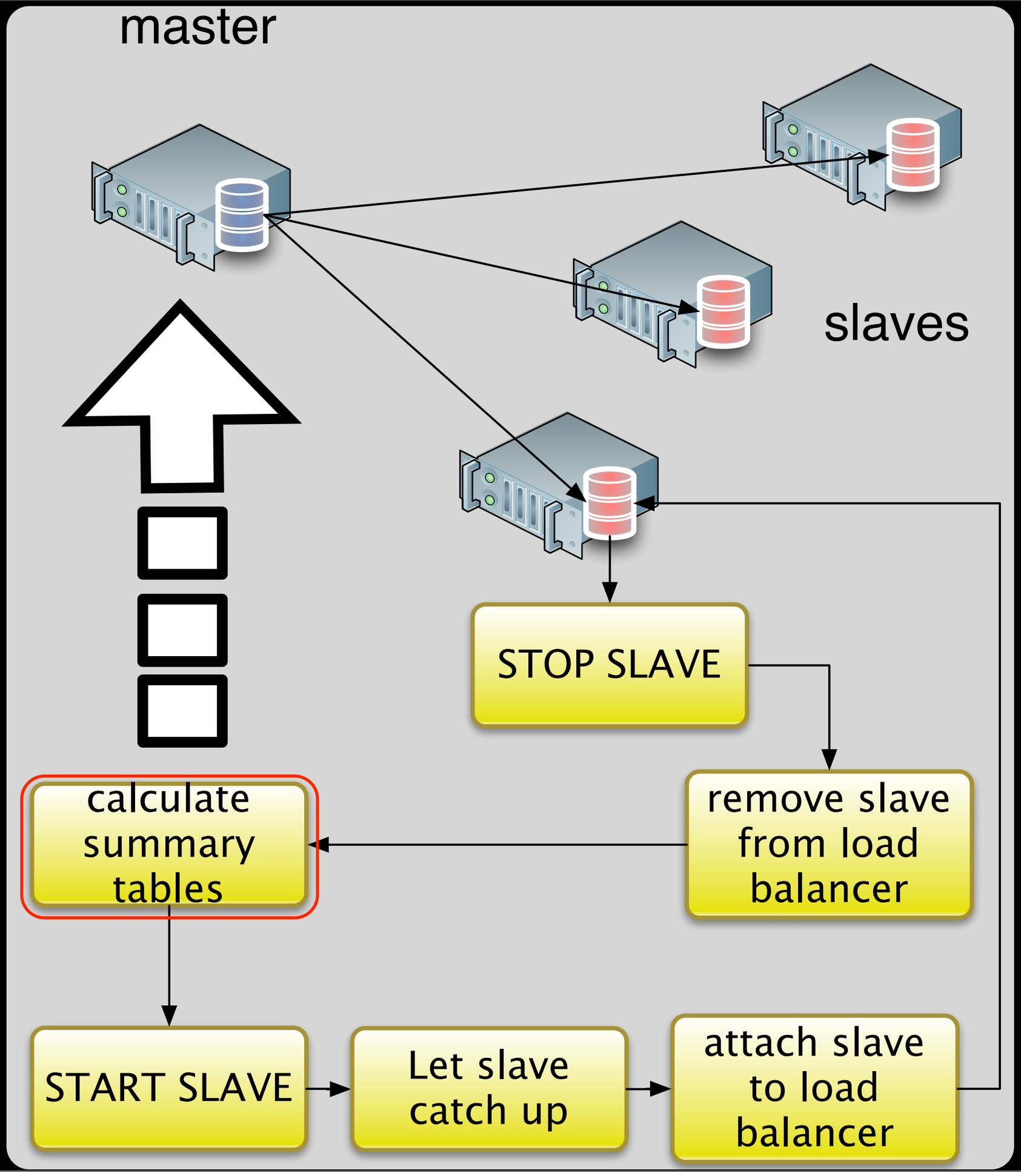
backup


master
slaves
STOP SLAVE
remove slave from load balancer
perform backup
START SLAVE
Let slave catch up
attach slave to load balancer

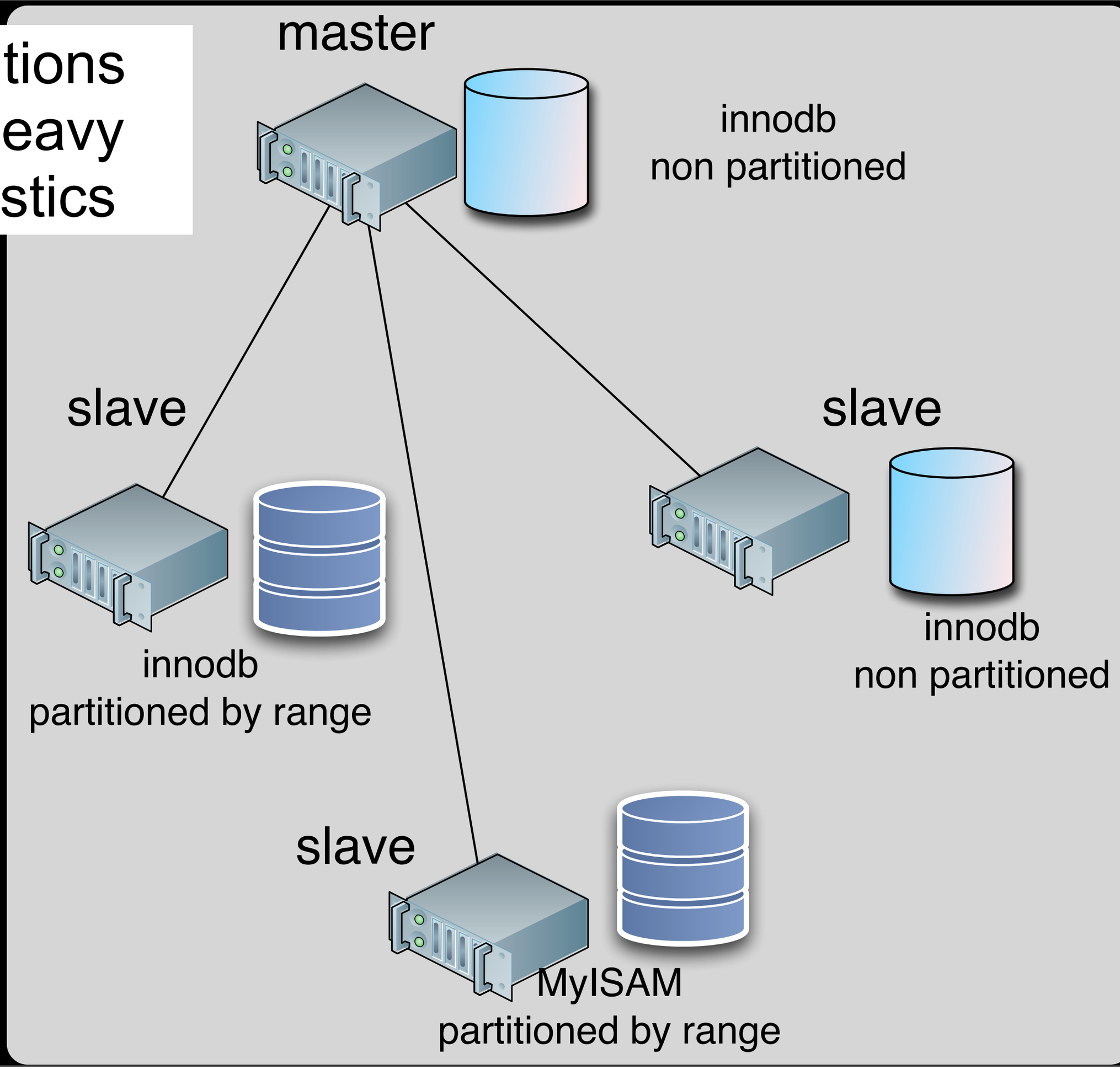
AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • **better reads**
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

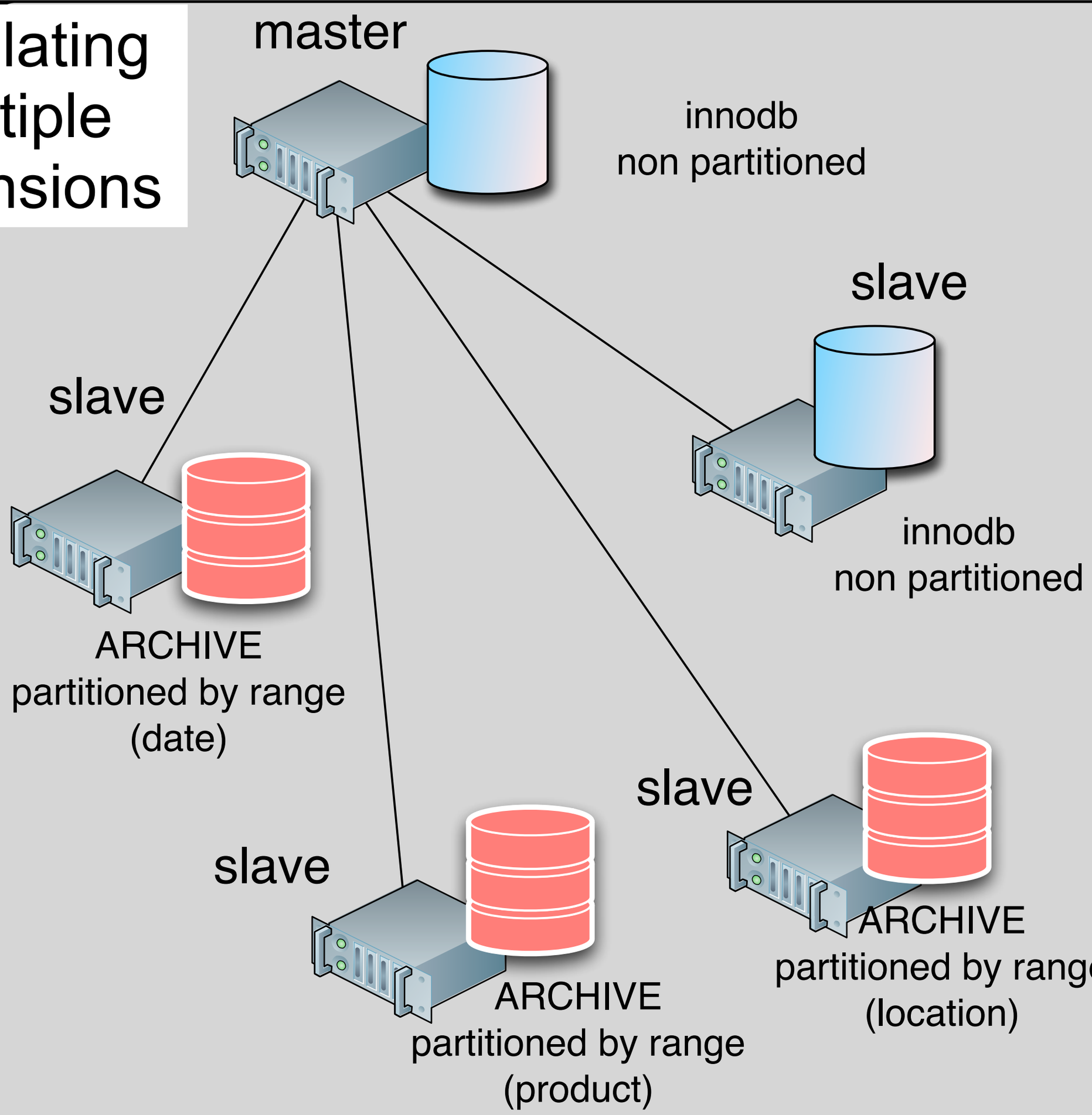
make summary tables



Partitions for heavy statistics



Simulating multiple dimensions



AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • More replication sessions

READ-ONLY slave

- Good practice: make a slave read-only
- Caveat 1: if promoting a slave, you need to remove the read-only option;
- Caveat 1: a user with SUPER privilege can write to a read-only slave

145

Default engine

- Caveat: default storage engine is not replicated.
- Example: master with default engine=Innodb and slave with default engine=MyISAM
- a CREATE TABLE without the ENGINE clause will use different engines on master and slave

146

AGENDA

master-to-master and circular replication

DEMO

Common filters

- on the master
  - binlog-do-db
  - binlog ignore-db
- on the slave
  - replicate-do-db
  - replicate-ignore-db

Common filters

- on the master
  - binlog-do-db
  - binlog ignore-db
- on the slave
  - replicate-do-db
  - replicate-ignore-db

DO NOT USE!

do Filters

What happens when everything goes right

transactions from client

all data comes to the master

binlog-do-*

The master logs only some of the transactions

replicate-do-*

The slave only gets some of the above

ignore Filters

transactions from client

all data comes to the master

binlog-ignore-*

The master logs all the transactions, except the ones that should be ignored

replicate-ignore-*

The slave further filters some of the above

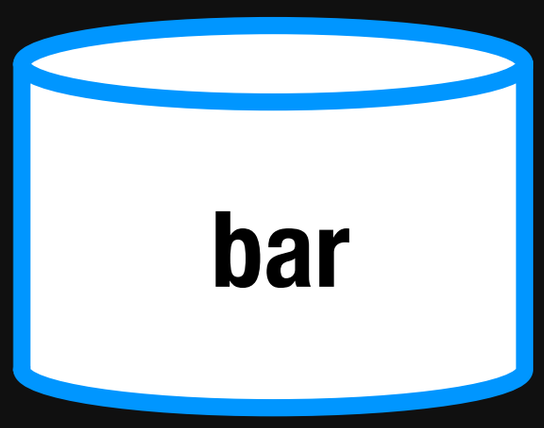
filters

What happens when something goes wrong

Master my.cnf:
binlog-do-db=foo


```
use foo;
insert into bar.t1 values (1)
```

it is replicated (breaks replication)




```
use bar;
insert into foo.t1 values (1)
```

is NOT replicated

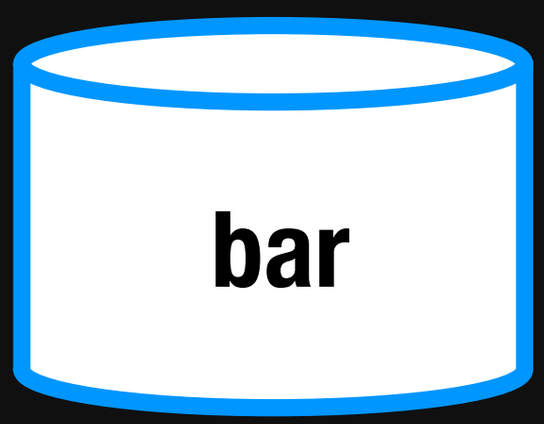
filters

What happens when something goes wrong

slave my.cnf:
replicate-do-db=foo


```
use foo;
insert into bar.t1 values (1)
```

it is replicated (breaks replication)




```
use bar;
insert into foo.t1 values (1)
```

is NOT replicated

"Safer" filters

- on the slave:
  - replicate-wild-do-table=db_name.%
  - replicate-wild-do-table=foo%.bar%
  - replicate-wild-ignore-table=db_name.%
  - replicate-wild-ignore-table=foo%.bar%

153

"Safer" filters

- on the slave:
  - replicate-wild-do-table=db_name.%
  - replicate-wild-do-table=foo%.bar%
  - replicate-wild-ignore-table=db_name.%
  - replicate-wild-ignore-table=foo%.bar%

DON'T MIX "do" and "ignore" filters

General rules of replication filters

- DON'T USE filters on the master.
- If you apply slave filters, the slave is not suitable for replacing a master or taking backups;
- Filters can break replication;
- Slave filters don't save bandwidth.

AGENDA

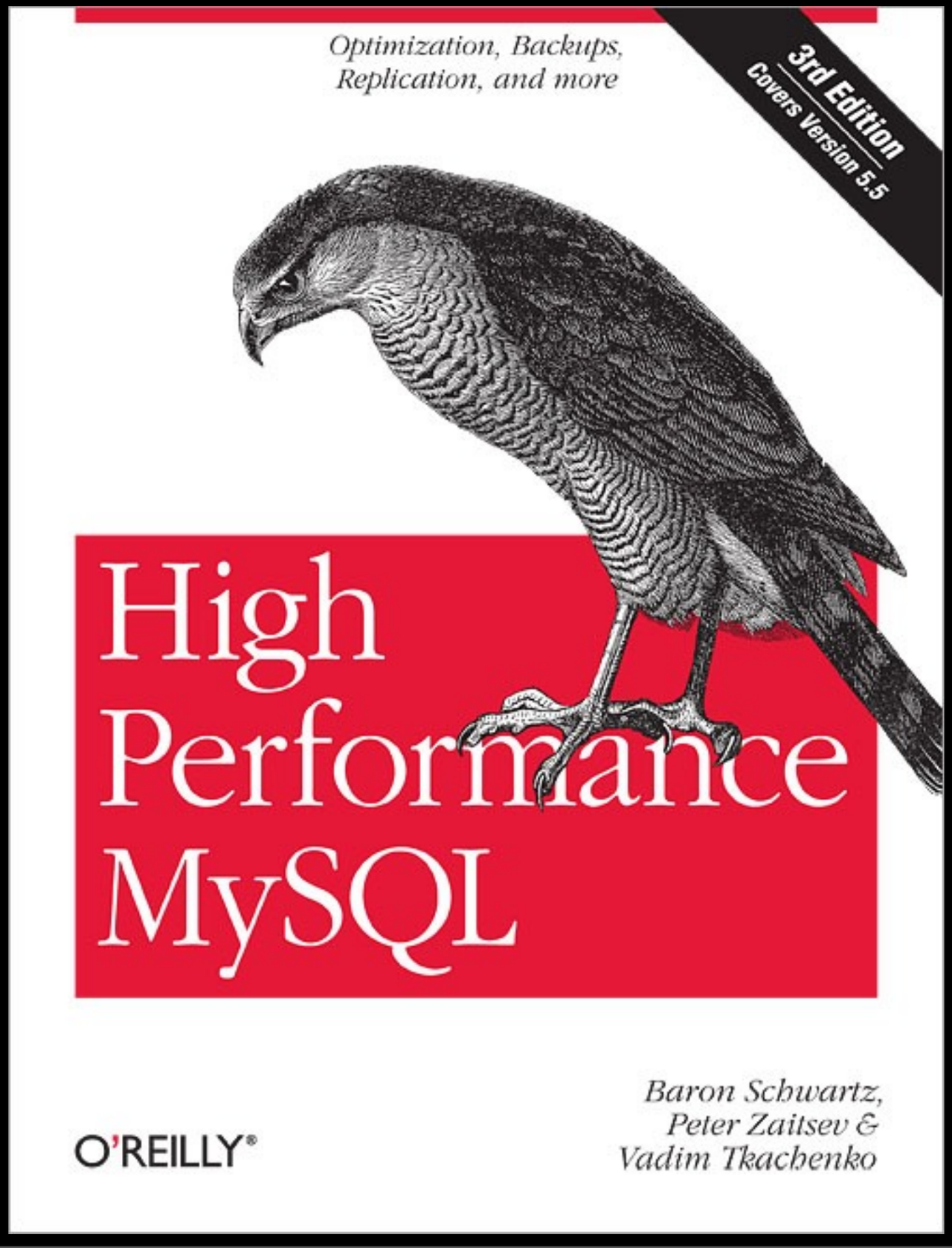
Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• **What to read** • More replication sessions

Read more

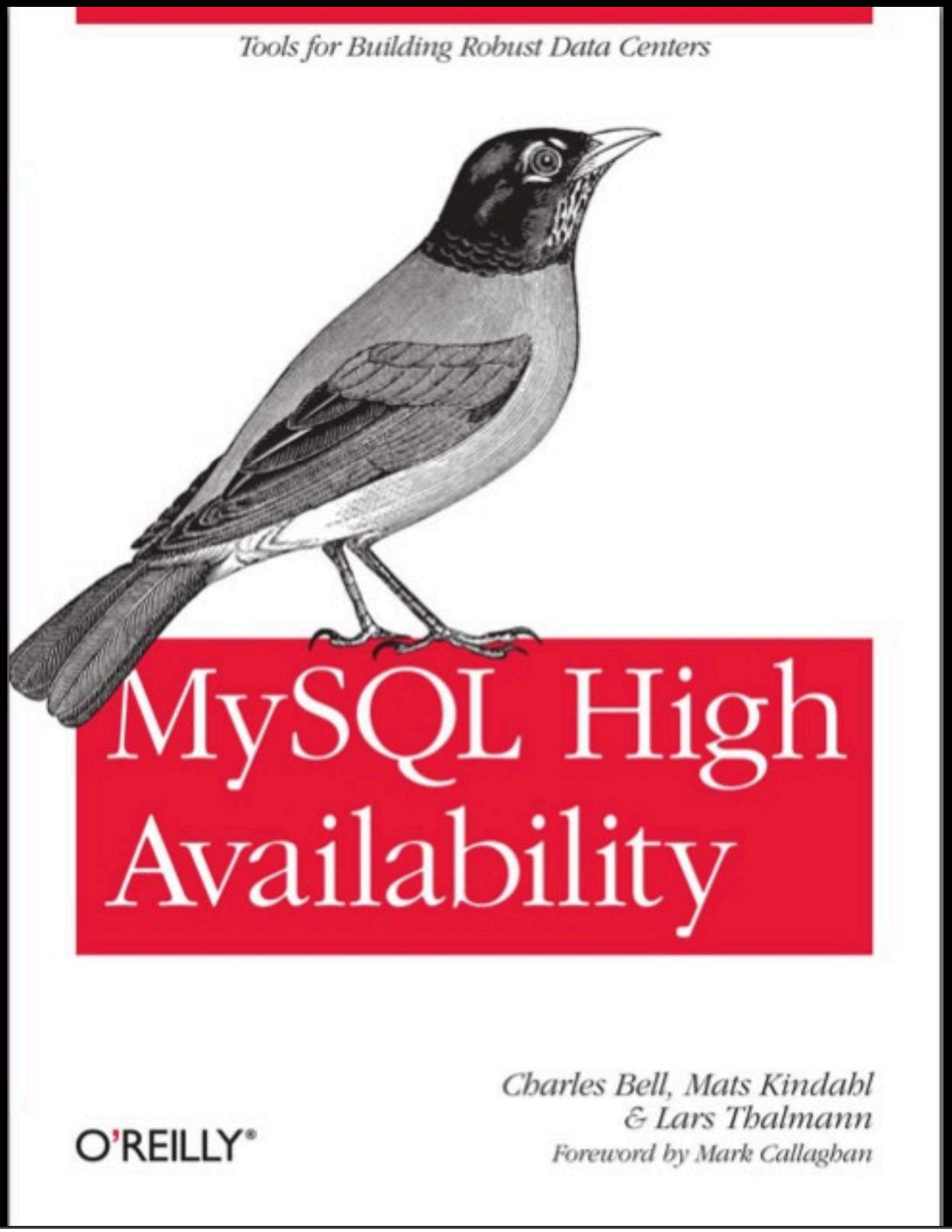
- The MySQL online manual

http://dev.mysql.com/doc

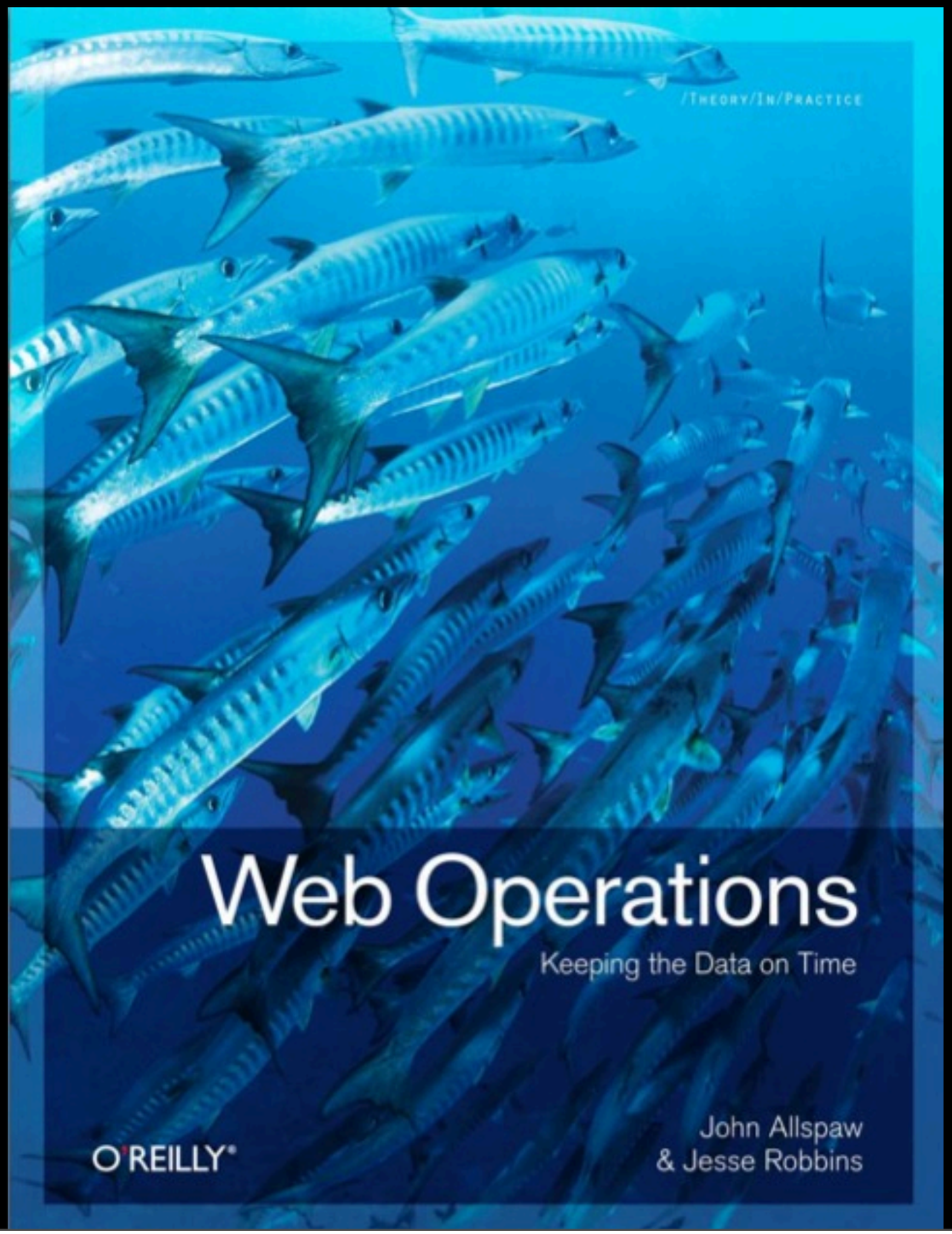
High Performance MySQL



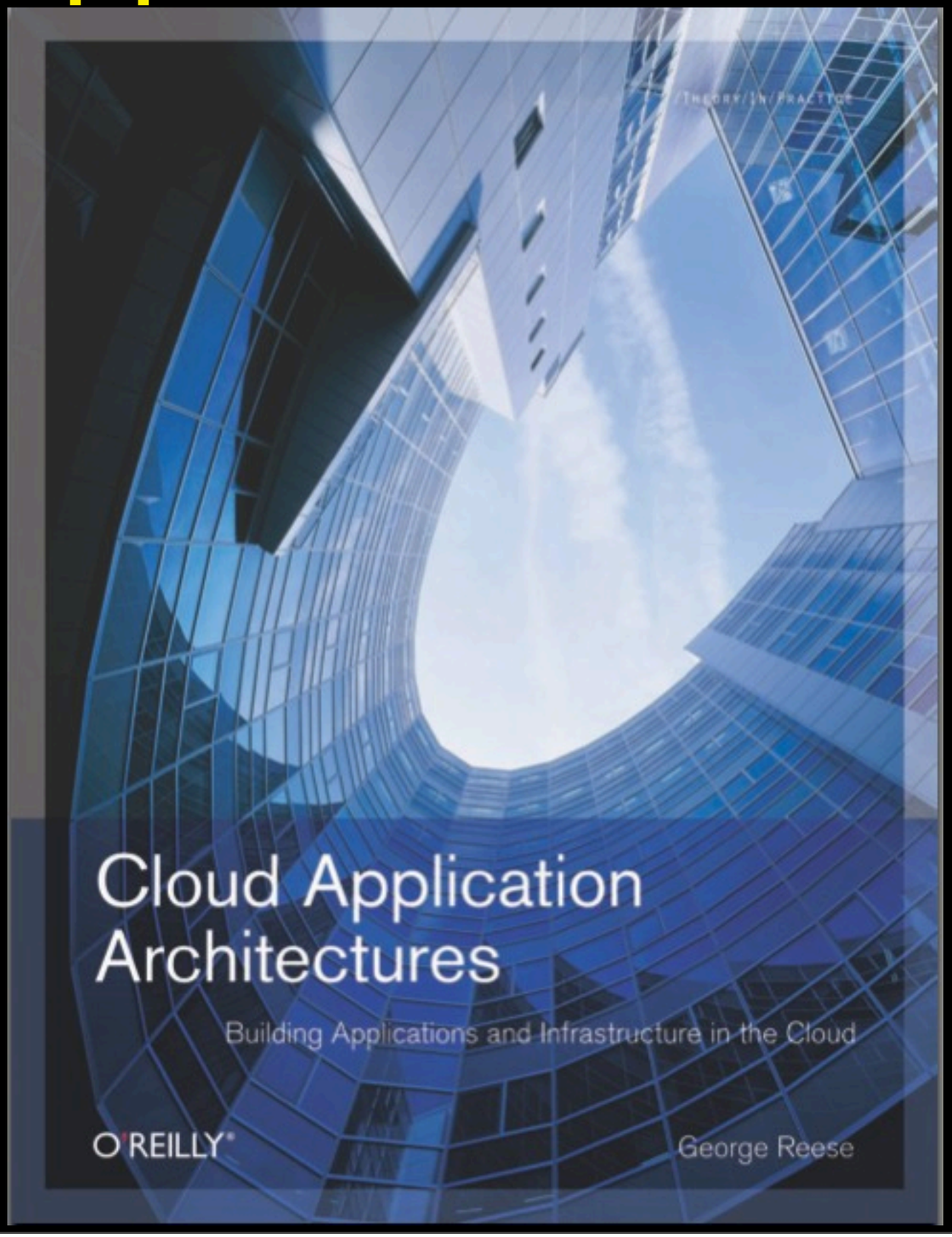
MySQL High Availability



Web Operations



Cloud Application Architectures



AGENDA

Why replication	• The web economy • Scaling out
How to set replication	• From single server to replication • Adding a slave
Using Replication	• Binary log formats • What gets replicated and how • Replication awareness
Managing replication	• Monitoring • Log management • Replacing a slave • Replacing a master
What Replication is for	• master replacement • backup • better reads
Gotchas, tips, and tricks	
More info	• What to read • **More replication sessions**

More replication sessions - Wednesday

- 11am Building a multi-master, multi-region database infrastructure in Amazon EC2
- 11am Performance practices for minimizing replication delay
- 1pm Diagnosing & Fixing MySQL Replication
- Replaying database load with Percona Playback
- 2pm Build simple and complex replication clusters with Tungsten Replicator
- 3:30pm What's new in MySQL 5.5 and 5.6 Replication

More replication sessions - Thursday

- 11am MySQL Replication: Pros and Cons
- 2pm Boost Your Replication Throughput with Parallel Apply, Prefetch, and Batching
- 3pm Verifying MySQL Replication Safely With pt-table-checksum 2.0

http://www.continuent.com

continuent

Database replication and clustering

WE ARE HIRING!

- Implementation/support engineer
- QA engineer
- Documentation writer

continuent

- http://www.slideshare.net/datacharmer

Bonus slides

- Semi-synchronous replication

semi-synchronous replication

- Available in 5.5 and higher
- Makes sure that at least one slave has copied the data.
- Increases reliability

transaction with regular replication

client

master

1

commit

execute

2

binary log

3

replication

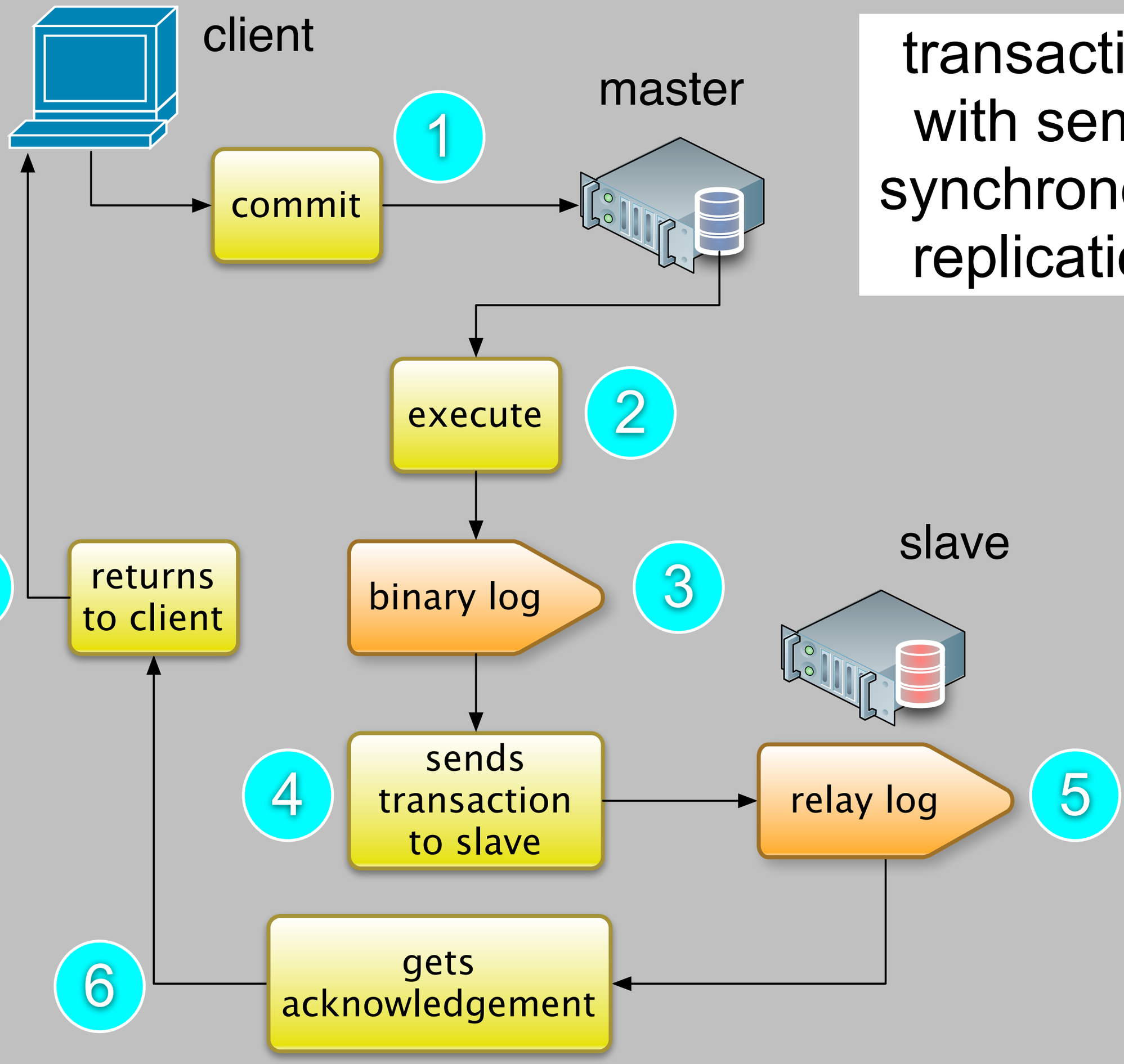
5

returns to client

4

slave

transaction with semi-synchronous replication



semi-synchronous replication in practice

- installation:
  - it’s a plugin.
  - Actually, two plugins

semi-synch replication install

```
# in the master
plugin-load=rpl_semi_sync_master=semisync_master.so
rpl_semi_sync_master_enabled=1


# in each slave
plugin-load=rpl_semi_sync_slave=semisync_slave.so
rpl_semi_sync_slave_enabled=1


# restart all servers
```

semi-synch replication check

```
# in the master
show variables like 'rpl_semi%';
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| rpl_semi_sync_master_enabled  | ON    |
| rpl_semi_sync_master_timeout  | 10000 |
| rpl_semi_sync_master_trace_level | 32    |
| rpl_semi_sync_master_wait_no_slave | ON    |
+-------------------------------+-------+
```

semi-synch replication check

```
show status like "rpl_semi_%tx";
+-----------------------------+-------+
| variable_name               | value |
+-----------------------------+-------+
| RPL_SEMI_SYNC_MASTER_NO_TX  | 0     |
| RPL_SEMI_SYNC_MASTER_YES_TX | 0     |
+-----------------------------+-------+
```

semi-synch replication test

```
master> create table t1 ( i int);
Query OK, 0 rows affected (0.01 sec)

master> show status like "rpl_semi_%tx";
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| Rpl_semi_sync_master_no_tx    | 0     |
| Rpl_semi_sync_master_yes_tx   | 1     |
+-------------------------------+-------+
```

disabling semi-synch

```
# for each slave

set global rpl_semi_sync_slave_enabled=0;
stop slave io_thread;
start slave io_thread;
```

disabled semi-synch replication test

```
master> insert into t1 values (1);
Query OK, 1 row affected (10.00 sec)

master> show status like "rpl_semi_%tx";
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| Rpl_semi_sync_master_no_tx    | 1     |
| Rpl_semi_sync_master_yes_tx   | 1     |
+-------------------------------+-------+
2 rows in set (0.00 sec)
```

disabled semi-synch replication test

```
master> insert into t1 values (2);
Query OK, 1 row affected (0.01 sec)

master> show status like "rpl_semi_%tx";
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| Rpl_semi_sync_master_no_tx    | 2     |
| Rpl_semi_sync_master_yes_tx   | 1     |
+-------------------------------+-------+
2 rows in set (0.00 sec)
```

re-enabling semi-synch

```
# in one slave

set global rpl_semi_sync_slave_enabled=1;
stop slave io_thread;
start slave io_thread;
```

reenabled semi-synch replication test

```
master> insert into t1 values (3);
Query OK, 1 row affected (0.01 sec)

master> show status like "rpl_semi_%tx";
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| Rpl_semi_sync_master_no_tx    | 2     |
| Rpl_semi_sync_master_yes_tx   | 2     |
+-------------------------------+-------+
2 rows in set (0.00 sec)
```