PHP: Vom Entwicklerbaukasten zur Enterprise-Plattform

The Architecture Gathering

Kore Nordmann (@koredn)

13th October 2016

Qafoo
passion for software quality

THE ARCHITECTURE GATHERING

Hi, I’m Kore (@koredn)

Qafoo
passion for software quality

THE LANGUAGE WARS

THE LANGUAGE WARS

A NEW HOPE

A Brief History

- 1995: Set of Perl scripts by Rasmus
- 1998: PHP 3: Rewrite by **Ze**ev & A**nd**i (Zend)
  - PHP: Hypertext Preprocessor
  - C-like standard library
- 2000: PHP 4: Dedicated virtual machine (Zend Engine)
- 2004: PHP 5: Actual object model
- 2009: PHP 5.3: Sane garbage collection
- 2015: PHP 7: Massive performance improvements


No Excuses – But Growing Up Can Be Hard

Evolution Of PHP

- Mixing paradigmes since 1995
  - Procedural from the beginning
  - Structs with functions since PHP 4
  - Object Orientation since PHP 5
  - We even got `goto` (since PHP 5.3)
- Horrible inconsistencies in standard library
  - `str_split()` vs. `strlen()` vs. `htmlspecialchars_decode()` vs. `IntlBreakIterator implements Traversable`

How can PHP power 80%[1] of the web?

[1] https://w3techs.com/technologies/overview/programming_language/all

THE LANGUAGE WARS

A NEW HOPE

ATTACK OF THE CLONES

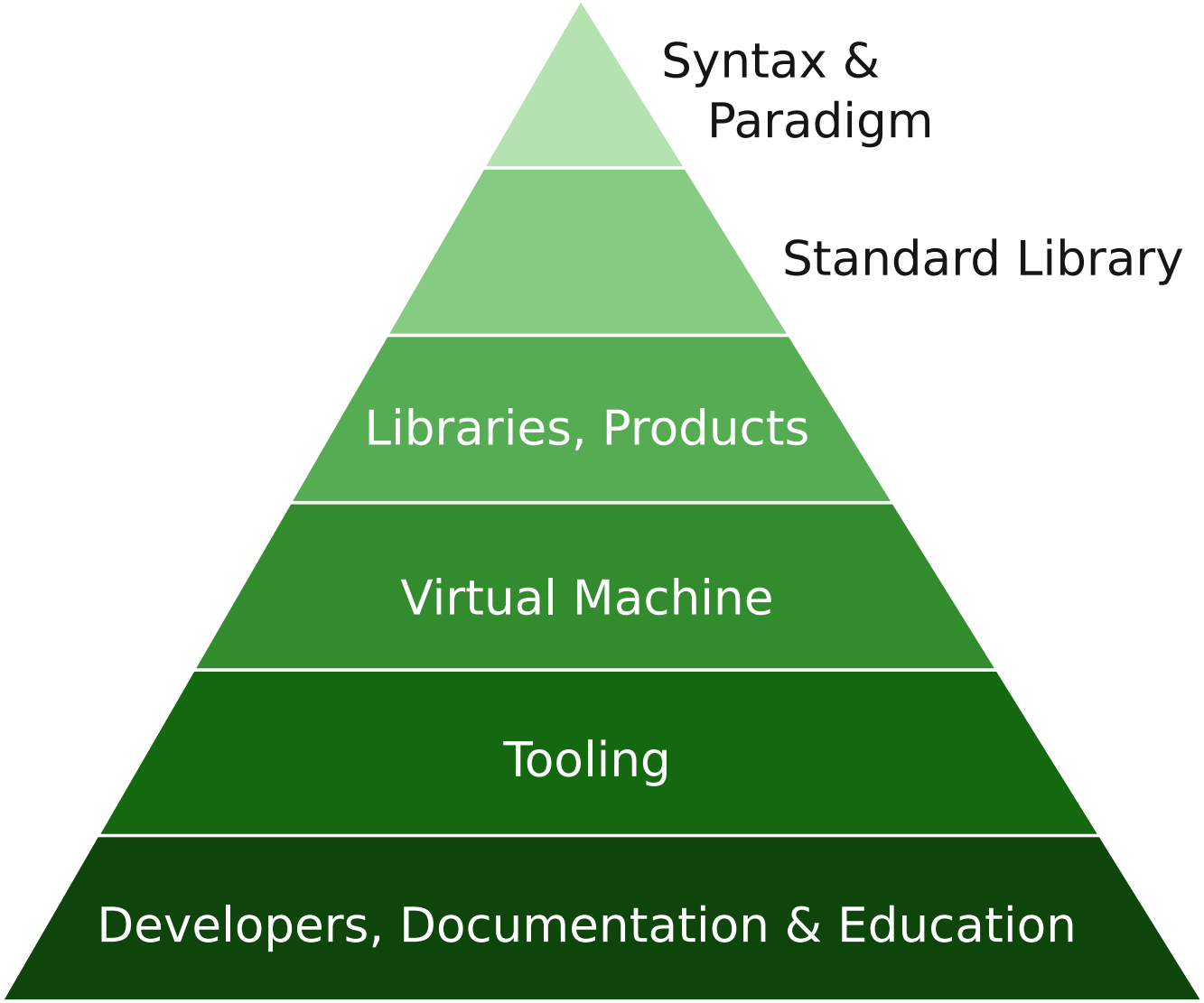
Attack Of The Clones

There was nothing else in 2000 (except Perl)

- But then there was:
  - Ruby On Rails
  - Django & Zope (Python)
  - ASP.net
  - Java Server Faces
  - ...

How can PHP still power 80% of the web?

It Is Not About The Language



Qafoo
passion for software quality

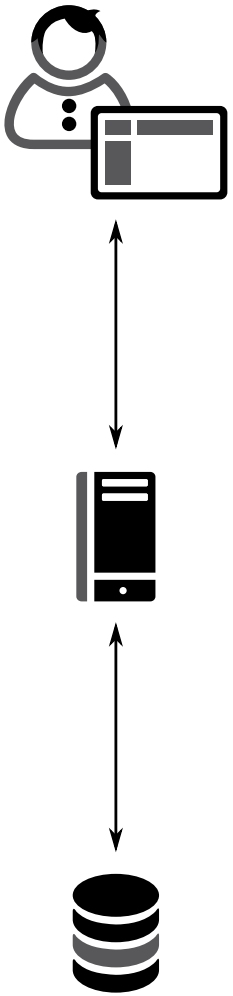
It Is Not About The Language

How does PHP power 80% of the web?

THE LANGUAGE WARS

A NEW HOPE
ATTACK OF THE CLONES
THE FORCE AWAKENS

Where Is PHP Used?



Architecture: It’s All Tradeoffs


Performance


Security

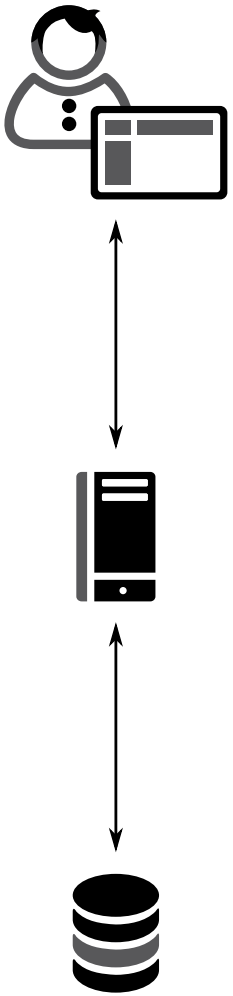

Availability


Modifiability


KRUPS
MADE IN THE IRISH REPUBLIC

Performance

Where Is PHP Used?



Number Crunching [2]

Language	CPU	Slower	Version
C++ (-O2)	0.973s	–	g++ 6.1.1
Java 8 (non-std lib)	1.126s	15%	1.8.0_102
Python 2.7 + PyPy	1.514s	55%	PyPy 5.4.0
Go	2.757s	183%	1.7
C++ (not optimized)	2.954s	203%	g++ 6.1.1
PHP 7.0	6.739s	592%	7.0.10
Javascript (nodejs)	7.202s	639%	4.3.1
Java 8 (see notes)	12.200s	1,153%	1.8.0_102
Ruby	13.147s	1,250%	2.3.1
Python 3.5	17.895s	1,738%	3.5.2
Python 2.7	23.691s	2,334%	2.7.12
Perl	25.562s	2,526%	5.22.2
PHP 5.6	68.784s	6,020%	5.6.17

[2] https://blog.famzah.net/2016/09/10/cpp-vs-python-vs-php-vs-java-vs-others-performance-benchmark-2016-q3/

Single Node Performance

- Performance does not match compiled code or good VMs
- Basically no support for threads
  - Experimental async I/O support: ReactPHP[3] (still single-threaded, like node.js)
  - "Experimental" threading support[4]
- Only basic support for forks[5]

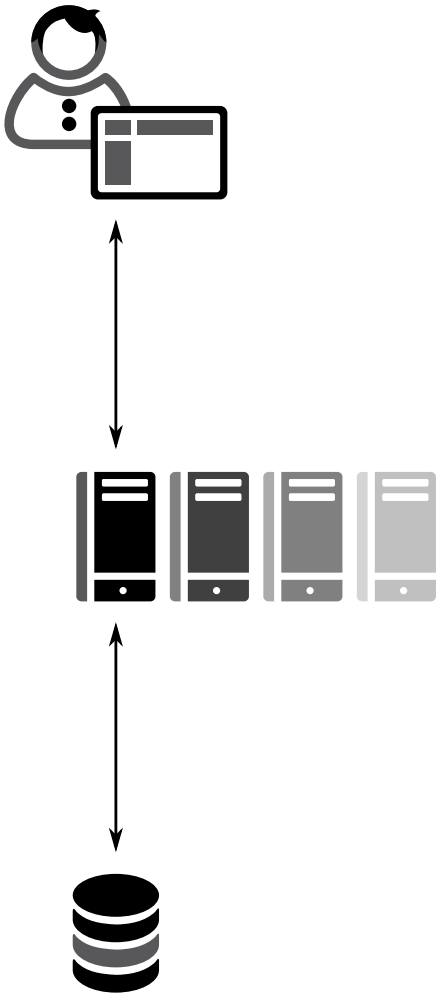
Do not use.

[3] http://reactphp.org/

[4] https://pecl.php.net/package/pthreads

[5] http://docs.php.net/pcntl

Horizontal Scalability



HTTP / REST Are Built For Scalability [6]

LCoDC$SS

[6] https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

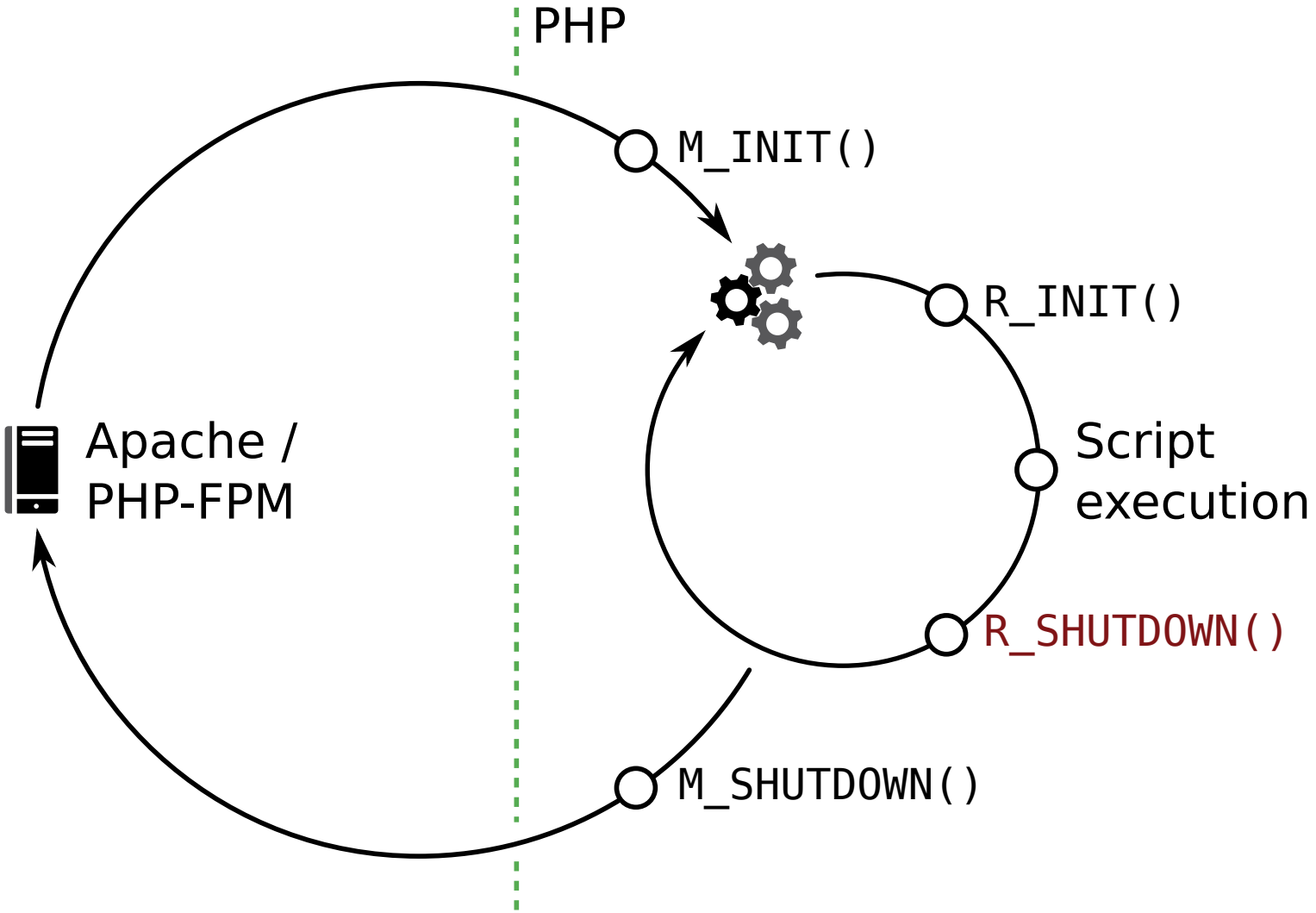
HTTP / REST Are Built For Scalability [6]

Layered Code on Demand Client Cached
Stateless Server

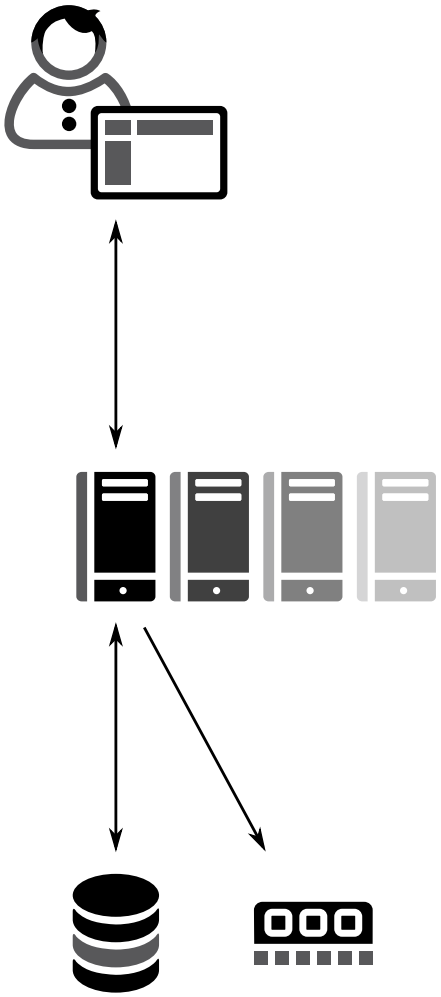
PHPs Shared Nothing Architecture

[6] https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

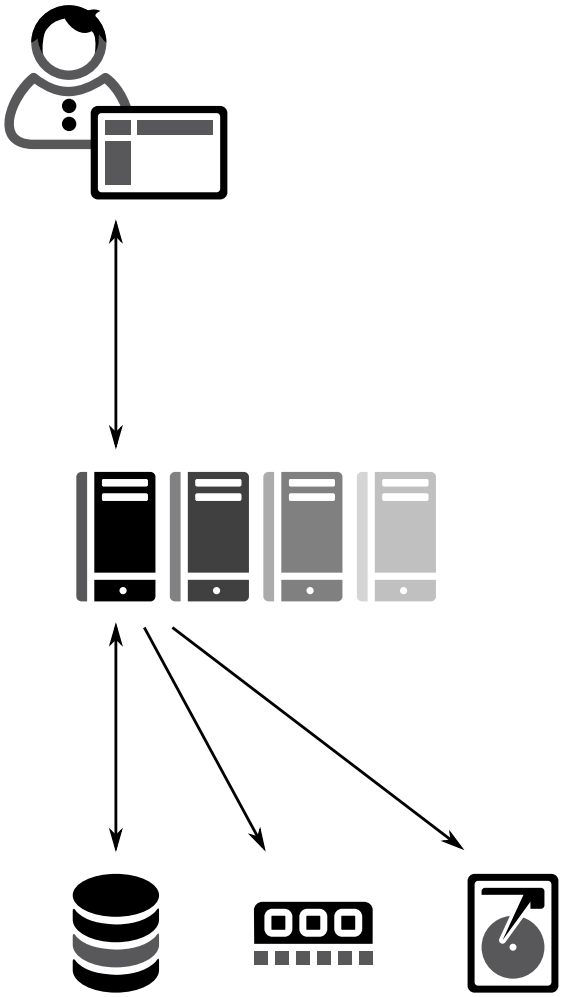
PHP Is Built For Shared Nothing



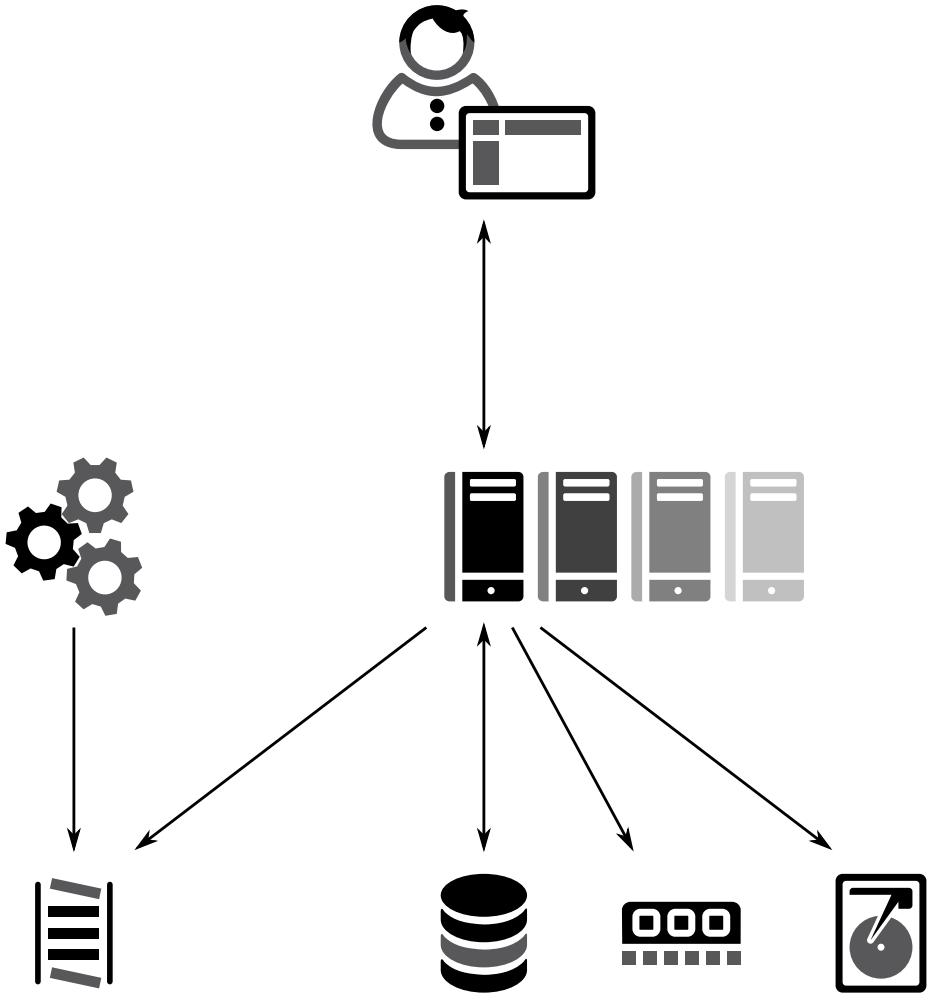
Pattern: Sessions



Pattern: Binary Data



Pattern: Offline Jobs



Summary

- PHP applications are usually scaled to multiple application servers from the very beginning
- Developers know the challenges and frameworks embrace them
- Servers are usually commodity hardware
- Do **not** use for:
  - Application Servers with shared object graph (Node.js?, Java, . . .)
  - WebSockets (Node.js?, Go, Erlang/OTP, Java, . . .)
  - Number Crunching (C, C++, Go, Java, . . .)


Security

Security

- “No” issues here
- Issues were caused by default configurations leading to stupid code
- Issues are caused by unaware developers
  - Maybe related to weak dynamically typed language
- It is not trivial to write insecure code with modern frameworks. . .

Low entry barrier is a double–edged sword. . .

Availability

Availability

- Horizontal scaling with shared nothing
  - Everything else must be highly available, too
- Resilience
  - "`NullPointer` exceptions" are finally catchable since PHP 7 (almost all errors are)
  - Management processes basically do not die
  - Just throw away unstable servers
    - Server provisioning is the default (Ansible, Puppet, Chef, . . .)
- Monitoring
  - Error monitoring with libraries like Monolog[7]
  - Application performance and error metrics with Tideways, NewRelic, AppDynamics, . . .

[7] `https://github.com/Seldaek/monolog`

Modifiability

Modifiability

- Hackability
  - (I think) The reason Wordpress is as big as it is...
  - ...like it or hate it – double-edged sword again.
- Trivial deployments
  - Just put new source on server and change link to new source directory...
  - Maybe tell opcode-cache, if `fstat` is disabled
- Tooling
  - Composer[8]: Sane dependency management
  - PHPStorm[9]: IDE with all the bits

[8] https://getcomposer.org

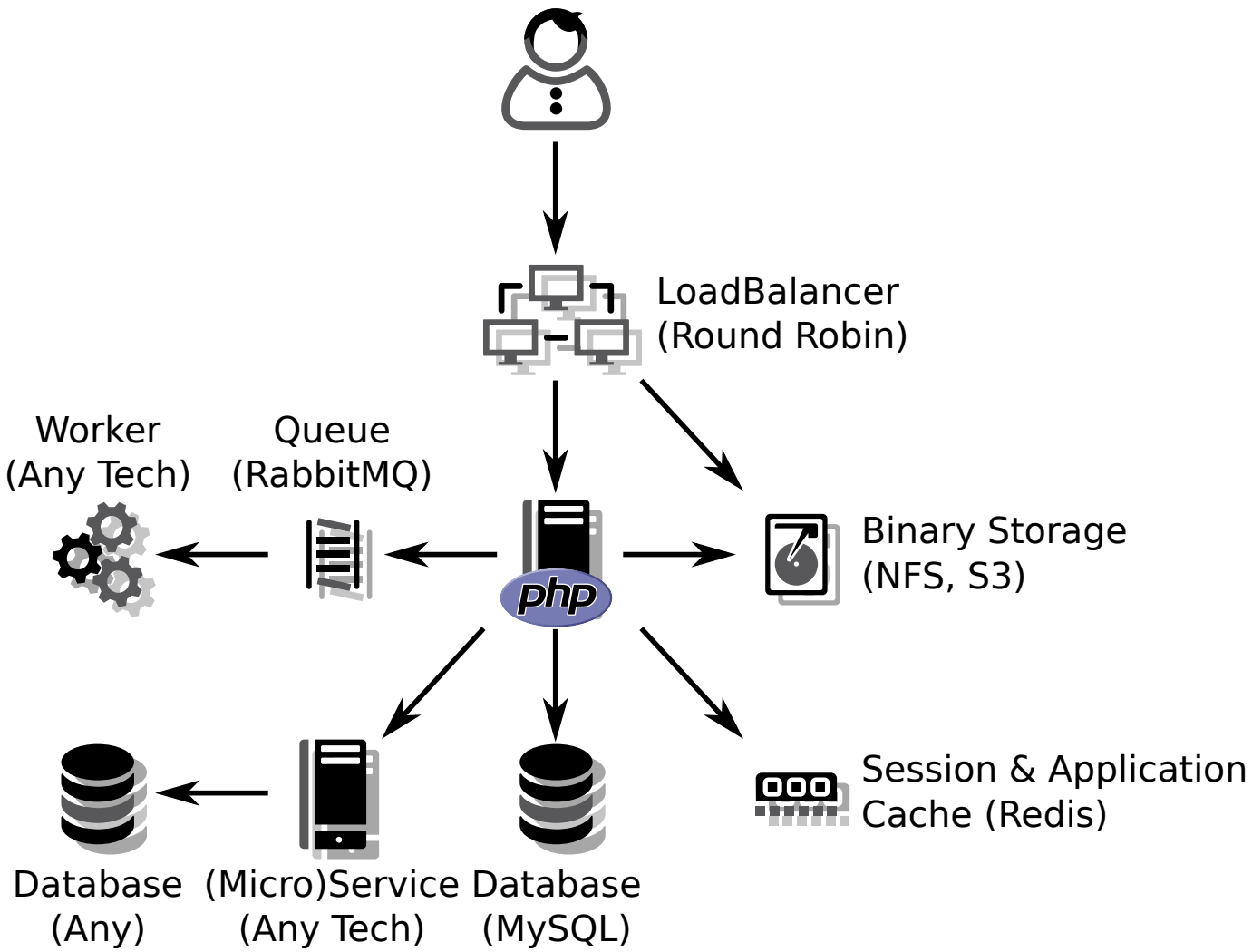
[9] https://www.jetbrains.com/phpstorm/

Quality Assurance

- **PHPUnit:** (Unit) Tests with PHPUnit are default for “all” libraries
- **PHPMD:** Mess detector exposes the common violations (coupling, size, complexity, . . .)
- **PHP_CodeSniffer:** Verification of codings standards (PSR-2)
- **PHPCS:** Detects copy-pasted code
- **Build Systems:** Phing (ant), Phake, . . .
- **CI:** TravisCI for Open Source, Jenkins, ContinuousPHP, . . .
- . . .


We Are Watching

Anatomy Of An Enterprise PHP Application




Relax

THE LANGUAGE WARS

...AND THEY ALL LIVED HAPPILY EVEN AFTER.

Qafoo
passion for software quality

THANK YOU

Rent a quality expert
qafoo.com

Ask Anything. . .