A NEW, SAFER, MAGENTO DEPLOYMENT

talk by angelo.maragna@gmail.com

THE PRESENTER

Angelo Maragna
Italian, 40y old, currently working at Silentnight here in Manchester as a **Magento Developer**
Coder since 1988 and professional developer since 2003.

Experience in:

- PHP development
- Windows server systems administration (from design to implementation, management and support)
- Windows and linux Web server administration
- ERP and CRM implementations (Italian ERPS, vTiger CRM)
- Door to door sales
- Freelance company websites developer (from sales, to graphic design, development and customer training)

- the story -

This presentation wants to provide some light on Magento's deployment process, a maybe different approach to many developed in the market, and a proof that hard work can still pay back on the new Magento 2.

Angelo Maragna

- the plot -

1. Magento concepts
2. Servers topology & old deployment
3. A new hope
4. Health and safety
5. Q&A

1. Concepts

1.1 - Type of changes

When you work on a Magento project, changes can be grouped in different areas:

- **Inner functions (backend - php files)**
- **Style enhancements (frontend - phtml, css, js)**
- **New classes and methods (backend - php files)**
- **Database alterations (backend - setup -php- files)**

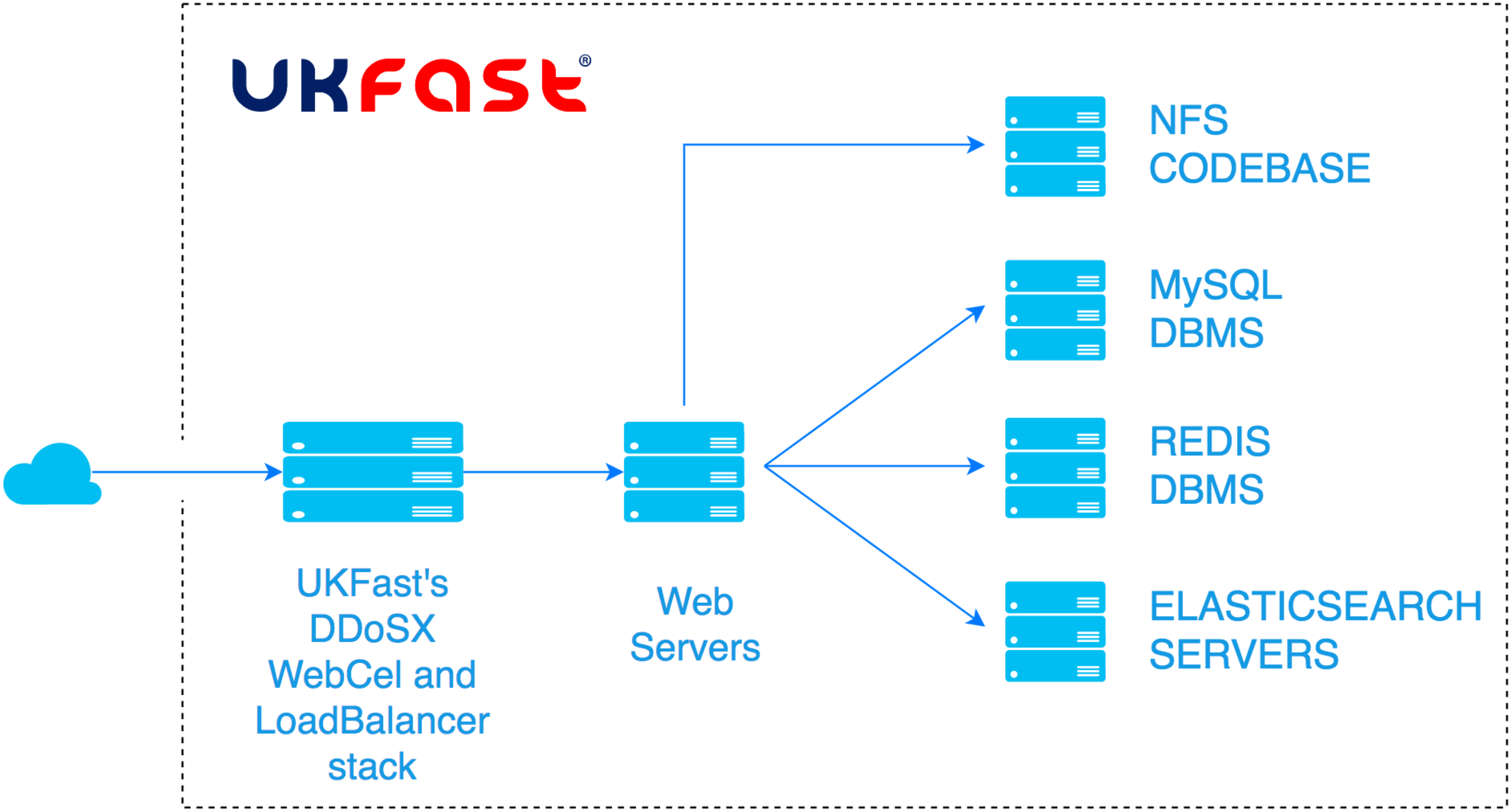
1.2 - Folders affected

- **pub/static/versionXXXXXX (css/js/html-js templates)**
- **generated (interceptors for public functions like plugins)**
- **var/cache (redis-01) (generated blocks and pages)**
- **var/session (redis-02)**

• • •

2. Server Topology

2.1 - Our live environment



UKFast

NFS
CODEBASE

MySQL
DBMS

REDIS
DBMS

ELASTICSEARCH
SERVERS

UKFast's
DDoSX
WebCel and
LoadBalancer
stack

Web
Servers

2.2 - The previous deployment

1. git pull
(partial codebase)

2. configure authentication files
(env.php - auth.json)

3. composer install

4. bin/magento setup:install

5. bin/magento deploy:mode:set production

6. zip and move the zip
on the production server

7. unzip to a temporary folder

8. maintenance mode on

9. swap the live folder with the temporary folder

10. bin/magento setup:upgrade —keep-generated

11. bin/magento cache:flush

12. maintenance mode off

• • •

2.2 - The previous deployment

1. git pull
(partial codebase)

2. configure authentication files
(env.php - auth.json)

3. composer install

4. bin/magento setup:install

5. bin/magento deploy:mode:set production

6. zip and move the zip
on the production server

7. unzip to a temporary folder

8. maintenance mode on

9. swap the live folder with the temporary folder

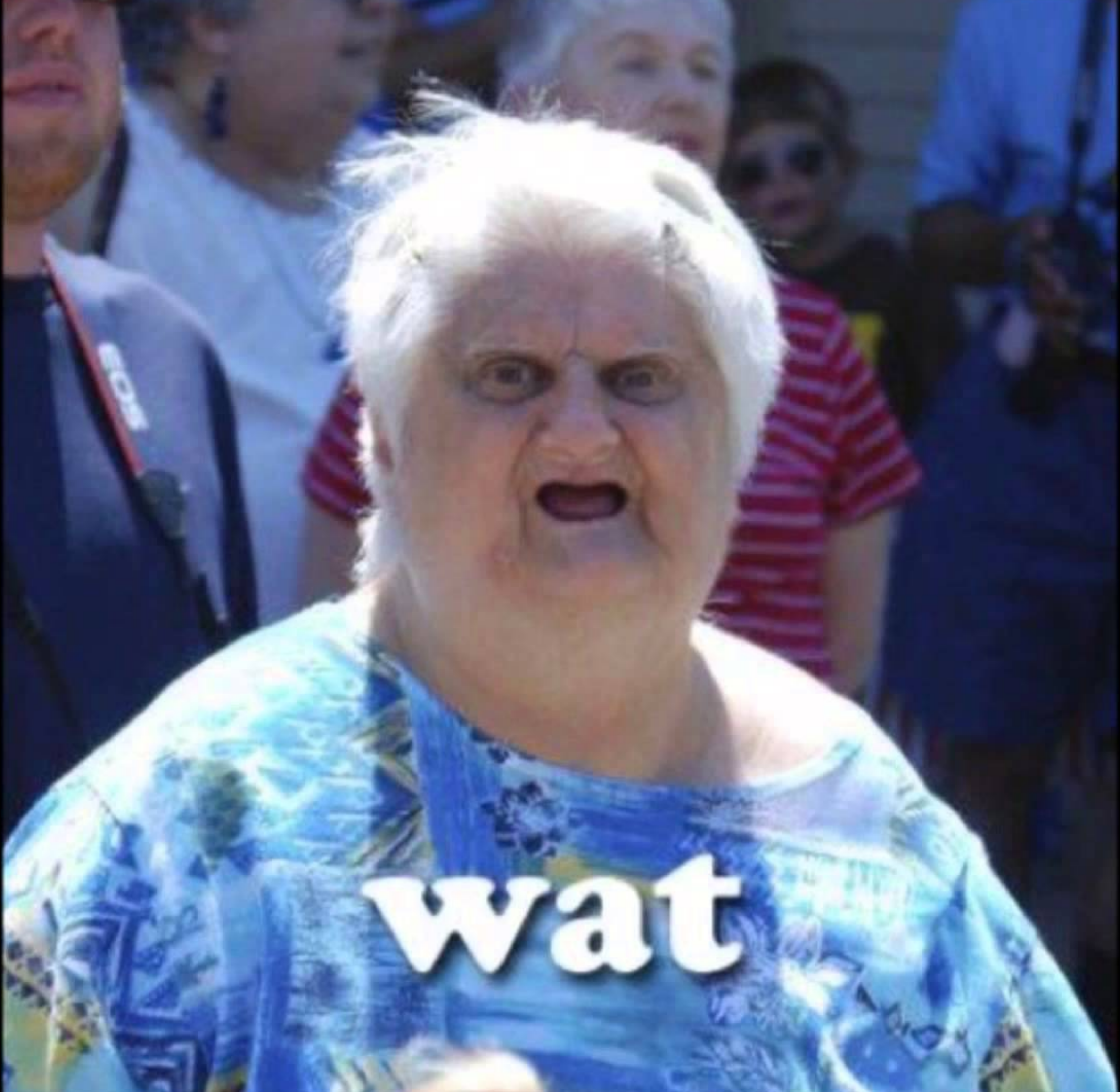
10. bin/magento setup:upgrade —keep-generated

11. bin/magento cache:flush

12. maintenance mode off

at least 2 minutes
of website offline

40 minutes
for a deployment


wat

2.2 - The previous deployment

1. git pull
(partial codebase)

2. configure authentication files
(env.php - auth.json)

3. composer install

4. bin/magento setup:install

5. bin/magento deploy:mode:set production

6. zip and move the zip
on the production server

7. unzip to a temporary folder

8. maintenance mode on

9. swap the live folder with the temporary folder

10. bin/magento setup:upgrade —keep-generated

11. bin/magento cache:flush

12. maintenance mode off

3. A NEW HOPE

3.1 - CHANGES EXAMPLES

We've seen before

- **Inner functions (backend - php files)**

  - - -

- **Style enhancements (frontend - phtml, css, js)**

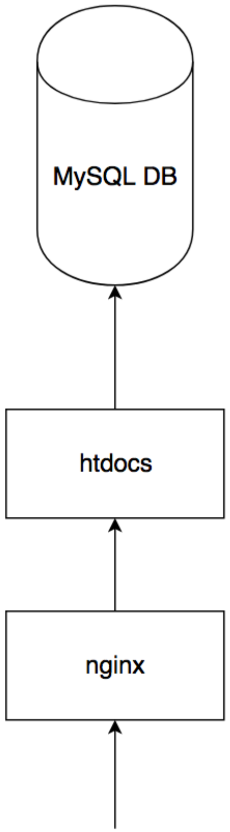
  - - -

- **New classes and methods (backend - php files)**
- **Database alterations (backend - setup -php- files)**

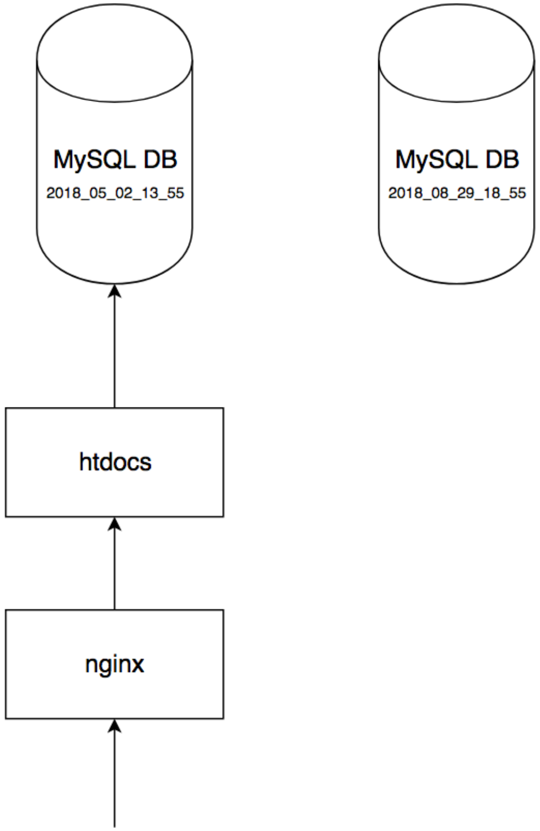
3.2 - CHANGES TYPOLOGY

- Minor changes (inner method functions or non cached pages)
- Static content (css / js / html-js templates)
- Full deployment (db upgrade required)

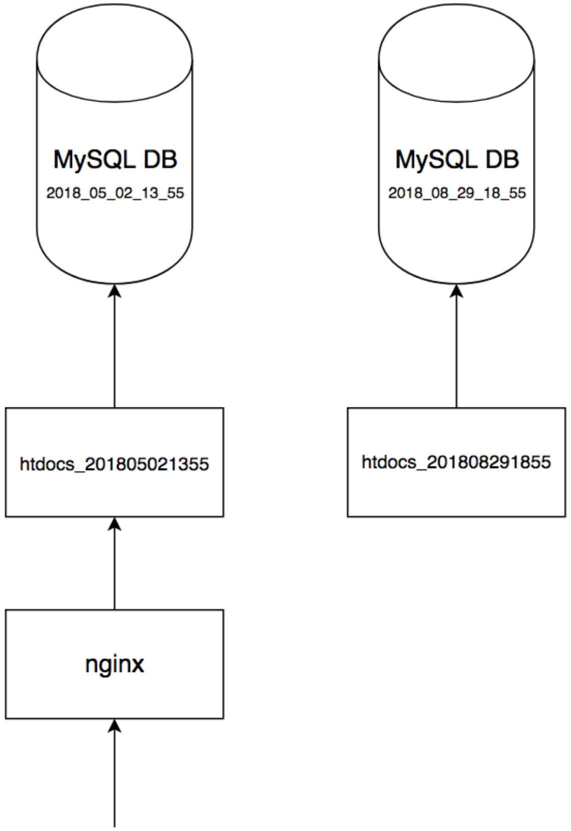
3.3 - NEW SAFER DEPLOYMENT DIAGRAM



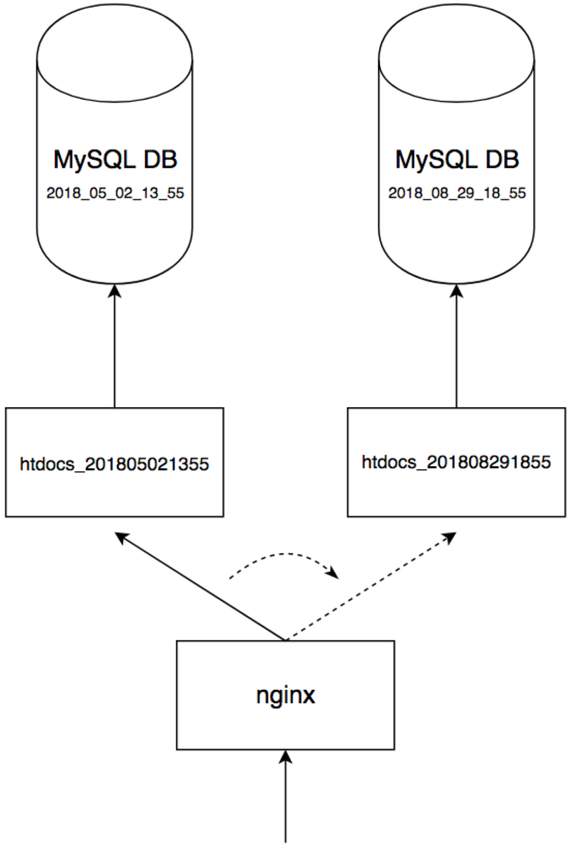
3.3 - NEW SAFER DEPLOYMENT DIAGRAM



3.3 - NEW SAFER DEPLOYMENT DIAGRAM



3.3 - NEW SAFER DEPLOYMENT DIAGRAM



3.4 - ADVANTAGES

- No downtime
- We can chose when to flush caches
- We can chose if to run db upgrades
- We don’t need the new database
- We can chose if to recompile everything or just recompile static content

3.5 - DEPLOYMENT STEPS

		MODE 1	**MODE 2**	**MODE 3**
	REF SRV	**(minor changes)**	**(static content)**	**(full deployment)**
UPDATE CODEBASE (GIT PULL)	DEVELOPMENT	X	X	X
INSTALL MAGENTO	DEVELOPMENT			X
REDEPLOY **STATIC CONTENT**	DEVELOPMENT		X	
BACKUP ACTIVE DB TO DB_TIMESTAMP	**DB SRV**	X	X	X
DISABLE CRONS	ADMIN SRV			X
DELIVER ARTIFACTS	**NFS SRV**	X	X	X
UNZIP & RESTORE PERMISSIONS	**NFS SRV**	X	X	X
CUSTOMISE DIRECTORIES AND FILES	**NFS SRV**	X	X	X
FLUSH CONFIGURATION	ADMIN SRV	X	X	
FULL CACHE FLUSH	ADMIN SRV			X
M UPGRADE	ADMIN SRV			X
SWAP SYM LINKS	ADMIN SRV	X	X	X
HEALTH CHECK	ADMIN SRV	X	X	X
ENABLE CRONS	ADMIN SRV			X
REGRESSION TESTS	ADMIN SRV	X	X	X

4. HEALTH & SAFETY

4.1- Advantages

- There is always a backup copy
- The website can't stay down
- We don't need to always recompile the code
- Deployment speed improved avoiding NFS operations
- No data can be lost
- No rush to debug issues on live environment

• • •

七転び八起き

"Fall seven times and stand up eight"

–old Japanese proverb

slides available on github.com/angelomaragna/zero-downtime-magento-deployment

THANKS

– Angelo Maragna

slides available on github.com/angelomaragna/zero-downtime-magento-deployment