# Appendix 1

# Requirements for running and launching Magento in case it is not hosted by us

## Introduction

This document describes the structure of the server environment needed for running a Magento store.

We also list the development technologies (CI, CD) we use as well as the requirements related to them in terms of hosting (to be applied in the actual server environment).

We will describe these requirements and prerequisites in the following sections. Having these fulfilled in technological environment is the basis of a smooth cooperation between the external hosting provider and AionHill's development team. Therefore if these requirements are not met on the provider's side, we cannot take responsibility for a flawless code performance.

This set of requirements does not include in detail installation and configuration of the hardware resources, the operating system, server software and other necessary infrastructure. Such guides and materials are publicly available on the Internet and the hosting service provider should have them at hand for safe and reliable server management. In this respect, AionHill cannot give any help to a third party since we are not prepared to provide consultation and support services of the kind.

## Prerequisites

For smooth cooperation, the following prerequisites should be met:

- Having a system capable of running Magento
- If the external provider does not have any experience in Magento hosting, setting up a demo or staging store is strongly recommended
- Git client software running on the system
- Installing and configuring a <DEPLOYKEY> in order to access AionHill's Git server
- PhpMyAdmin details for accessing the database

## Macros and other details

| Name of Macro | Value of Macro | Description |
|---|---|---|
| DEPLOYKEY | | Project related private key, AionHill sends it out to the client. |
| PROJECTNAME | | Project related name, AionHill sends it out to the client. |
| LOCAL.XML | | File needed for operating Magento |
| PROJEKT.CONF.JSON | | File needed for AION Utils operation. |

## Details required form the client

| Detail | Sample values |
|---|---|
| Accessing the admin and frontend interfaces of the demo stores (URL, access data) | http://shop.mycompany.com<br>https://shop.mycompany.com/admin<br>user name: admin<br>password: 1234abcd |

| | |
|---|---|
| PhpMyAdmin access (URL, access data) | https://shop.mycompany.com/devtools/PHPMyAdmin/<br>htpasswd user name: utils<br>htpasswd password: utils123<br>mysql user name: shop<br>mysql password: shop123 |
| Access route to the online store in the operating system | /var/www/shop |
| Access to the new store (URL, access data) | http://shop.mycompany.com<br>admin user name: admin<br>admin password: 1234abcd |
| Protocol of the new online store (http, https, both) | http, https |

# Installation Guides

## Installing the Web Application

The process of installing the web app is the following:

- *Installing <DEPLOYKEY>:*
    - this is to be inserted in the <userhome>/.ssh directory with the appropriate authentication:
      *hmod 600.ssh/authorized_keys && chmod 700.ssh*
    - *su – webserveruser –s /bin/bash*
    - *git clone ssh://git@office.aion.hu:20022/aion/<PROJEKTNAME>.git <webroot>*
    - (here, password should not be requested of the key settings are correct)
    - *cd <webroot>*
    - *git checkout master*
    - *Setting <LOCAL.XML>*
    - Insert the included local.xml file in the <webroot>/app/etc/ directory, while modifying the database parameters (setting the correct parameters).

# Suggestions

Please consider the following  suggestions:

- Backup
    - Saving backup files of the existing web directories
        - daily backup kept for 1-3 days
        - weekly backup kept for 3 months
        - Creating SQL database backup
        - daily backup kept for 1-3 days
        - weekly backup kept for 3 months
    - PhpMyAdmin application
        - should be protected by a password
        - should be available via an HTTPS protocol only
        - recommended access route: <SHOP>/devtools/PHPMyAdmin/
    - Redis Cache
        - It’s a memory based object caching service for reaching an optimal page speed.
    - Varnish Cache
        - It’s a full page cache server application for reaching an optimal page speed.

# Other useful information

## Magento hosting general system requirements

For finding out more on Magento system requirements, please visit:

- http://magento.com/resources/system-requirements
- http://magento.com/resources/previous-magento-system-requirements
- “magento-check.php” is a useful application checking Magento system requirements
    - https://gist.github.com/atomicpages/4383809

**Magneto system requirements can be tested with the help of a newly downloaded and installed basic Magento system. We strongly recommend performing this before signing a final contract with the service provider. If you already have a Magento shop, it can be used for testing the server environment.**

Magento can be downloaded here: https://www.magentocommerce.com/download

Installing Magento 1 demo store:

http://devdocs.magento.com/guides/m1x/install/installing_install.html

The following tasks need to be performed after installation:

- Frontend functionality
    - The demo store should appear perfectly both via http and https (if this is the considered protocol)
    - Backend functionality
    - Admin access should be tested. Here http and https need to work as well.

### Public key based ssh authentication

https://mediatemple.net/community/products/dv/204644740/using-ssh-keys-on-your-server

### Git

https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

### PhpMyAdmin

- http://phpmyadmin.net/

## Release and patch processes

During these processes we deploy the final and tested features to the servers

These include:

- Patches: Smaller fixes that do not have an impact on the database
- Releases: such more serious deployments that have an impact on the database

Project managers (POs) always inform the client about which of the two types of deployment are going to take place.

In the case of patches, the hosting service provider can perform the task with simple “git” commands.

In the case of releases, some extra caution is needed from all the participants, especially from the provider to make sure that the data of the application stay safe and can be rolled back to a previous version.

For every patch or release task, a maintenance window is needed scheduled by the client. At this pre-defined time the hosting provider puts the application to maintenance mode with close cooperation with AionHill’s developers.

Before performing patch and release projects, the hosting provider should create extra backups of the present application, both in terms of the database and the entire file structure in order to be able to restore these immediately if asked by the client or AionHill (e.g. in the case of a deployment failure).