Installation

Until version 2.2.x Z-Push was officially only distributed as a tarball. This source tarball will be further maintained for Z-Push 2.3, but it's highly recommended to use the newly available repositories and packages instead. If you are looking for the tarball, scroll to the end of the page. Starting with Z-Push 2.4.0 there won't be tarball releases.

Repository

With the release of Z-Push 2.3 a public repository is available.

Stages

There are different stages of packages available:

Stage	Stability	Description
final	stable	Contains the final releases of Z-Push 2.4.x. This corresponds to the `master` branch of GIT.
old-final	old-stable	Contains the final releases of Z-Push 2.3.x. This corresponds to the `release/2.3` branch of GIT.
pre-final	testing	Contains public alpha and beta releases of Z-Push. This corresponds to the `release/2.4` branch in GIT.
develop	experimental	Contains nightly builds. Use it to always have the latest code installed. **These packages are not recommended for production use** (unless you know exactly what you are doing and are aware of the risks). Corresponds to the `develop` branch in GIT.

Distributions

The final repository contains builds for the following distributions/architectures:

Distribution	Architectures	Comment
Debian 7.0	i586 x86_64	
Debian 8.0	i586 x86_64	
Debian 9.0	i586 x86_64	Since Z-Push 2.3.7
Fedora 26	i586 x86_64	Since Z-Push 2.4.0
Fedora 27	i586 x86_64	Since Z-Push 2.4.0
RHEL 6 + PHP 5.6	x86_64	
RHEL 7	x86_64	
RHEL 7 + PHP 5.6	x86_64	

SLES 12	ppc64le x86_64	
SLES 12 + PHP7	ppc64le x86_64	
Ubuntu 14.04	i586 x86_64	
Ubuntu 16.04	i586 x86_64	PHP7 support since Z-Push 2.3.4
Ubuntu 18.04	i586 x86_64	PHP7.2 support since Z-Push 2.4.3
Univention 4.0	i586 x86_64	
Univention 4.2	i586 x86_64	
Univention 4.3	i586 x86_64	
openSUSE Leap 42.3	i586 x86_64	
openSUSE Tumbleweed	i586 x86_64	

PHP 5.4 required

Z-Push 2.3, 2.4 and 2.5 require at least PHP 5.4. Due to this, packages for older distributions are not available.

Usage of repositories together with Zarafa/Kopano

Even though the the Z-Push Kopano backend is included in all of the above repositories, this backend can only be used in distributions that are also supported by Kopano/Zarafa and therefore provide php-mapi package.

URL

The base URL of the repository is:

```
http://repo.z-hub.io/z-push:/<stage>/<distribution>/
```

To e.g. add the stable repository for Ubuntu 16.04 use:

```
deb http://repo.z-hub.io/z-push:/final/Ubuntu_16.04/ /
```

To find the correct URL for your distribution, access:

http://repo.z-hub.io/z-push:/final/

Packages

Main packages

Name	Component	Provides	Comment
z-push-common	Core	-	Base components of Z-Push.
z-push-config-apache	Config	Apache configuration	Installs the Z-Push alias in the apache configuration and restarts the webserver.
z-push-config-nginx	Config	Nginx configuration	Add Nginx configuration file. Make sure you edit it to adapt your needs.
z-push-backend-caldav	Backend	Calendar	To get calendaring from a CalDav server.
z-push-backend-carddav	Backend	Contacts	To get contacts from a CardDav server.
z-push-backend-combined	Backend	-	To use different backends for different folder types.
z-push-backend-imap	Backend	Email	To get email from an imap server.
z-push-backend-kopano	Backend	Email, Calendar, Contacts, Tasks, Notes and GAB	Connect against a Kopano or ZCP server to get email, calendar, contacts, notes, tasks and search the GAB. **Dependencies** This package depends on the php-mapi packages provided by Kopano/Zarafa. These packages can be found for the community or via the repositories for Kopano subscribers.
z-push-backend-ldap	Backend	Contacts	To get contacts from an LDAP server.
z-push-backend-galsearch-ldap	SearchProvider	GAB	To search the GAB (Global Address Book) against an LDAP server.
z-push-ipc-memcached	IPC Provider	-	Stores volatile states (Interprocess communication) in a memcache server. Recommended large for multi-host setups and setups with more than 200 users.
z-push-ipc-sharedmemory	IPC Provider	-	Stores volatile states (Interprocess communication) in shared memory. Use on single host installations only. Recommended for setups with less than 200 users.
z-push-state-sql	StateMachine	-	Stores mobile states in a SQL database (e.g. mysql). Recommended for multi-host setups. Migration script available in Tools. More information see State Machines.

* GAB = Global Address Book.

Other Packages

Name	Component	Comment
z-push-autodiscover	Autodiscover	To be installed on the Z-Push host with a main backend.

z-push-kopano-gabsync	KOE GAB	Provides the GAB (Global Address Book) for the Kopano Outlook Extension. More information: Configuring GAB-Sync for Kopano OL Extension
z-push-kopano-gab2contacts	Tool	Contains a script that synchronizes the GAB (Global Address Book) into any contacts folder (e.g. in the public folder). This script is also able to update these contacts (e.g. via a daily cron job). Introduced with Z-Push 2.3.5.
z-push-kopano	Meta package	Installs the basic stack for Kopano (and ZCP). To run on apache install `z-push-config-apache`. Includes: `z-push-common, z-push-backend-kopano, z-push-ipc-sharedmemory` *Please note the additional dependencies for the z-push-backend-kopano package mentioned in the table above.*

Installation

This will guide you through the basic installation on example of Kopano/Zarafa.

The examples below show how to add the repository and install Z-Push using shared-memory and the file state machine (same as with Z-Push 2.2.x).

`z-push-top` and `z-push-admin` can be executed from any location (as `root`).

The main directory is `/usr/share/z-push`, the configuration files are located in `/etc/z-push`.

If you use Kopano on a different host or ZCP you need to adjust the `MAPI_SERVER` in `/etc/z-push/kopano.conf.php` to fit your environment.

Debian und Ubuntu based systems

Add repository

First of all, the repository needs to be added to your distribution.

E.g. for stable releases on Debian 8, create a new file in `/etc/apt/sources.list.d/z-push.list` with the content:

```
deb http://repo.z-hub.io/z-push:/final/Debian_8.0/ /
```

> To find the correct URL for your distribution, access:
>
> http://repo.z-hub.io/z-push:/final/

Then, download the repository key and it to the keychain.

```
wget -qO - http://repo.z-hub.io/z-push:/final/Debian_8.0/Release.key |
sudo apt-key add -
```

Install packages from repository

Assuming that Apache is your webserver, run:

```
sudo apt-get update
```

Then:

```
sudo apt-get install z-push-kopano z-push-config-apache
```

RHEL based systems

Add repository

First of all, the repository needs to be added to your distribution.

For the final releases on CentOS 7, create a new file in `/etc/yum.repos.d/z-push.repo` with the content:

```
[z-push]
name=Z-Push noarch Enterprise Linux 7 - $basearch
baseurl=http://repo.z-hub.io/z-push:/final/RHEL_7
failovermethod=priority
enabled=1
gpgcheck=0
```

Update the package list:

```
yum update
```

Note: If you are using RHEL_6_PHP_56 or RHEL_7_PHP_56 repository and memcached for inter process communication, make sure you have Remis repository enabled.

Install packages from repository

Assuming that Apache is your webserver and you want to install basic Z-Push with the Kopano backend, run:

```
yum install z-push-common z-push-config-apache z-push-backend-kopano z-push-ipc-sharedmemory
```

From Source

If you really want to install from the source tarball, the download is available here: http://z-push.org/download/

For information on how to install from source, please consult the INSTALL file in the tarballs main directory.

Instead of using the tarball or packages you can also use Z-Push from our GIT. See how to get the latest Z-Push code from GIT.

Speed optimizations

Opcache

Please check if your OS already has or is compatible with PHP opcache. This increases performance a lot as the PHP is compiled and cached. More information can be found here: https://kb.kopano.io/display/WIKI/Speeding+up+WebApp+and+Z-Push+with+PHP+opcache

To check your opcache usage have a look at this script: https://github.com/rlerdorf/opcache-status

States

The file state machine is used by default. It's the easiest to setup and fits most user profiles. As the states are just files they can easily be backup-ed and maintained.

There are a few downsides to the file state-machine, e.g. they are not clusterable. Some admins use NFS to share the states between several machines, which is required if you run several z-push systems and users can connect to any of them (like with a round-robin load balancer without session stickiness). NFS works but has some issues with stability. In this scenario we strongly recommend the usage of the sql state machine.

More about state machines here: State Machines

Inter process communication (IPC)

By default Z-Push installs the shared-memory IPC provider. It's the easiest to setup and a well fit for the most installations.

There are downsides using shared-memory IPC:

- PHP memory usage per request is about 5 to 10 MB higher than when using the memcache IPC provider

- It's not clusterable. If you use several z-push servers with shared states (like described above) the IPC data needs to be shared between these machines as well. The Shared-memory IPC doesn't allow that.
- It doesn't scale for hundreds of users.

You should use the memcache IPC provider in these cases. It requires the installation of a central memcache service, where all Z-Push instances connect to. It's a lightweight service that only requires a few MB of RAM (32 MB should be enough for most cases). You can install it on one of the Z-Push hosts or setup a dedicated machine (ideally with fail over).

It's not ideal if the IPC data is lost (e.g. when restarting the memcache service), but it's not a big problem either. Within a few requests the required data is built up again.

If your Z-Push server is running out of memory handling Z-Push requests, you should immediately switch to memcache IPC (can be running on the same system). There will be no downtime for users.

Related configuration topics

- Set Timezone
- Mobile Policies
- State Machines
- Fail2Ban support
- Syslog Integration