eZ XROW Cluster for Amazon

Preparation

For preparation you can provide details for the following questions and information relevant your cluster.

Common questions:

- What are all the hostnames of your website?
- Does you website use SSL? If so, provide your private key and certificates.
- Does your website want to use AWS SES as a outgoing email service? If so, provide a list of one or more sender addresses like webmaster@domain.com.
- How much storage do you need for the Database and the NFS?

Technical questions:

- Do you want SOLR replication on the web nodes?
- Do you want NFS replication?
- What is the redundancy you want?
- How can we aquire the sources and the migration data?

Tasks:

- Define bootstraping
- Define backup strategy
- Define talk about failover scenarios
- Define autoscaling

Getting Started

Preparation in the AWS Console

1. Create security / firewall rules. Open Port 80, 443, 22
2. Create a key pair
3. Create a RDS, if needed

We will now start our first instance and all others after wards.

1. Login to the AWS console
2. Choose your target region e.g. EU-WEST
3. Go to EC2->AMIs and select private images
4. Select the "eZ Cluster EL6" image with the newest date and press launch
5. Choose a instance type like m1.large and press continue
6. Cut and paste your ezcluster.xml into the field User Data and press continue
7. Fill the tag "Name" with the identifier of the instance your want to lauch according to your ezcluster.xml e.g. "WEB1"
8. Start over to start the rest of the instances defined in your ezcluster.xml.

Final steps after your cluster has been started the first time.

1. Get storage going in a multi master scenario. Execute on the first storage server:

   ```
   drbdadm -- --overwrite-data-of-peer primary storage
   ```

   and wait till the drives are synced completely. Check with

   ```
   cat /proc/drbd
   ```

2. Copy the var directory

   ```
   rsync -ruzWz -e "ssh -i key.pem" --size-only --force --progress --
   exclude '*/cache/*' --delete /path/on/my/old/server ec2-
   user@STORAGEHOST:/mnt/nas/var.live
   ```

3. Dump your database

   ```
   mysqldump -n --default-character-set=utf8 --opt --single-transaction
   -hlocalhost -umyusername -pmypassword dabasename > dump.sql
   ```

4. Inject the database
5. Index the SOLR on the SOLR Master.

   ```
   php extension/ezfind/bin/php/updatesearchindexsolr.php -
   sSITEACCESSNAME --clean-all --conc=2 --php-
   exec=/usr/local/zend/bin/php
   ```

6. Clear the cache from the DFS table.

   ```
   DELETE FROM ezdfsfile WHERE name REGEXP
   '[[.slash.]]cache[[.slash.]]';
   ```

   or delete with

   ```
   php bin/php/ezcache.php --clear-all --purge
   ```

General Description of the eZ Cluster Image

The Image create is a general purpose image. It can serve all required needs within the cluster.
Operating System

- OS: Centos 5.7 / 6.0
- Extra Packages
  - Varnish 3.0
  - Zend Server CE 5.5 modded by xrow
  - eZ Find rpm by xrow
  - eZ ODF rpm by xrow, not installed because openoffice not available over yum
  - PEAR AWS/SDK, Zeta Components
  - ezlupdate by xrow

Cluster Definition

The role or purpose of the image can be defined through user parameters or tagging. The most reasonable use is via userparameter. See an example XML file which you can supply as a user parameter at startup.

The Cluster Service

The services required are started through a init.d service called ezcluster.

You can start and stop this service like any regular service. This service will start automaticly at boot. It will just act different depending on what role you have assigned the Maschine. "ezcluster" should be always run as root user.

```
/etc/init.d/ezcluster start
/etc/init.d/ezcluster stop
```

The cluster configfile (ezcluster.xml)

cluster tag

```
<cluster lb="clustername" zones="eu-west-1a" update="yes" debug="no"
sites_storage="300">
```

instance tag

```
    <instance name="WEB1" bind="50.17.183.26" database-
slave="mysql://slave:password@mysql-read-1.us-east-1.rds.amazonaws.com">
            <role>web</role>
            <role>solr-slave</role>
        </instance>
```

Creating a new custom image

To create a new snapshot of an image please execute this command. This method is usually not needed and recommended you should try to configure maschi.
nes via bootstrapping.

```
ezcluster imagecreate
```

Ports and Services

Only HTTP/HTTPS on port 80 and 443 should be exposed to the outside world.

- Varnish Port 80 (HTTP)
- Apache Port 8080 (HTTP)
- Apache SSL Port 8080 (HTTPS)
- Solr Port 8983 (HTTP)
- Mysql 3306

- NFS 2049
- DRBD 7789
- OCFS2 7777
- 5404, 5405 UDP corosync/cman (Cluster Manager)

eZ Cluster Image and eZ Publish

Directories & Files

- /mnt/storage is the path were the STORAGE Volumne is mounted
- /mnt/nas is the mount point of the NAS
- /mnt/ephemeral is the mount point of the free ephemeral AWS Storage that is clear on each reboot. A large instance has about 400 GB.
- /var/www/sites is the directory where all eZ Publish installation directories are
- /etc/httpd/sites is the directory where all the vhosts reside

The logic behind this is scripted in PHP. If you need to know more details please look at following files.

- /bin/ezcluster (CLI management interface)
- /etc/init.d/ezcluster (service deamon script)
- /usr/share/ezcluster/* (Management libraries, templates and other scripts)
- /etc/ezcluster/ezcluster.xml (Main configuration file, copied from supplied user data)

Files

- /etc/varnish/ezcluster.vlc is the Varnish configuration file for the cluster
- /etc/drbd.conf is the config file for DRBD
- /etc/ocfs2/cluster.conf is the config file for OCFS2

Roles

Following roles are available and can be assigned to a instance. After changing a roles you need to restart ezcluster.

- web (runs varnish and httpd)
- dev (development instance runs web, database)
- storage (runs the NFS)
- database (runs the database)
- admin (runs the crons)
- solr
- solr-slave

About Varnish

Caching is enabled by default. Varnish will evaluale the caching headers and act accordinly. You can disable caching by using a hostname that starts with nocache (e.g. nocache.live.aws.example.com). You can disable caching by adding "Header add X-Varnish-Control "disabled"" to the .htaccess.

SOLR

Is available over http://[INTERNALIP/EXTERNALIP]:8983/solr/ for the local maschines.

Solr Admin Interface

```
lynx http://localhost:8983/solr/admin
```

Halt/disable replication for rebuild of the index

1. Disable replication ausschalten on all solr hosts
2. `lynx http://localhost:8983/solr/admin/replication/ -> Button "Disable Poll"`
3. Execute on the server with the solr master / admin server
4. `nohup php extension/ezfind/bin/php/updatesearchindexsolr.php -s [YOURSITEACCESS] --allow-root-user --clean-all  --conc=3 --php-exec=/usr/local/zend/bin/php &`
5. Enable replication
6. `lynx http://localhost:8983/solr/admin/replication/ -> Button "Enable Poll"`

STORAGE

We have an active / active NFS Server pair synced with DRBD and OCFS2 on top. Meaning both NFS are MASTER.

Ideally we would like to have some Loadbalanced/MASTER/SLAVE eZ Publish Driver that could properly make use of the ACTIVE / ACTIVE NFS. In case the current NFS fails just run "ezcluster restart" on the webserver and it connects to another active NFS.

Resolve DRBD split-brain recovery manually

After split brain has been detected, one node will always have the resource in a StandAlone connection state. The other might either also be in the StandAlone state (if both nodes detected the split brain simultaneously), or in WFConnection (if the peer tore down the connection before the other node had a chance to detect split brain).

At this point, unless you configured DRBD to automatically recover from split brain, you must manually intervene by selecting one node whose modifications will be discarded (this node is referred to as the split brain victim). This intervention is made with the following commands:

1. /etc/init.d/ocfs2 stop
2. drbdadm secondary storage
3. drbdadm disconnect storage
4. drbdadm -- --discard-my-data connect storage
5. /etc/init.d/ocfs2 start

On the other node (the split brain survivor), if its connection state is also StandAlone, you would enter:

1. ezcluster stop
2. drbdadm connect storage
3. ezcluster start

You may omit this step if the node is already in the WFConnection state; it will then reconnect automatically.

If all else fails and the machines are still in a split-brain condition then on the secondary (backup) machine issue:

drbdadm invalidate storage

Starting up new Standalone Instances

Create one off latest image
Give it a "Name"
Define "ROLES" or hand over the contents of ezcluster.xml as user data of the instance. Preferred is handing over ezcluster.xml.

```
For Storage ROLES=STORAGE
    For Webserver ROLES=WEB
    For Development ROLES=DEV
```

Starting up a Development Instance

```
Start a New Instance
    Apply tag name "ROLES" wiht value "DEV"
```

MYSQL

You can set certain my specific values to optimize MySQL execution.

```
        <database dsn="mysql://xrow:openpass@TEST-US-EAST/xrow">
          <setting name="table_cache">4000</setting>
        </database>
Host: localhost
    User: root
    No Password
```

Creating a new IMAGE

```
Start latest saved Image
    Apply changes
    run "ezcluster createimage" when you are logged on the shell
```

SES (Simple Email Service) with eZ Publish

1. Register your sending email with the ezcluster commandline tool.
2. `ezcluster verify bjoern@xrow.de`
3. Copy the SES Transport into eZ Publish and regenerate autoloads.

4. wget http://s3-eu-west-1.amazonaws.com/xrow/downloads/ezcluster/xrowsestransport.php
5. mv xrowsestransport.php lib/ezutils/classes
6. php bin/php/ezpgenerateautoloads.php -e -k
7. Setup the SES Transport in eZ Publish
8. [MailSettings]
9. EmailSender=bjoern@xrow.de
10. AdminEmail=bjoern@xrow.de
11. Transport=ses
12. TransportAlias[ses]=xrowSESTransport
13. Clear the ini caches eZ Publish
14. php bin/php/ezcache.php --allow-root-user --clear-all --purge
15. Configure SES for your SPF, Sender ID and/or DKIM (optional) http://docs.amazonwebservices.com/ses/latest/DeveloperGuide/index.html?SPFSenderIDDKIM.html

Cronjobs

You can assing cron to either an environment or instance tag. Cron jobs defined for an environment are just executed on the server that has the admin role. Crons with in an instance will be active on each instance you define them for.

```
<instance name="DEV1">
    <role>dev</role>
    <cron timing="0-55/5 * * * *" cmd="/some/command"/>
</instance>
<instance name="STORAGE1" volume="STORAGE1">
     <role>admin</role>
</instance>
<environment name="live">
<cron timing="15 * * * *" group="-s adminsiteaccess rssimport" />
    <cron timing="0-55/5 * * * *" cmd="/some/command"/>
<environment>
```

Bootstrapping

Once an image boots up you can add procedures to auto create you sites and fill you webroots with source files.

The current working directory of the script will be the webroot.

```
<environment name="live">
 <bootstrap>
  <script>
  <![CDATA[
  #!/bin/sh
  echo "created some stuff"
  ]]>
  </script>
 </bootstrap>
...
```

Testing / Cheat sheet

Tesing like ELB, 200 should be there
curl -A ELB-HealthChecker -I http://localhost/