Contents

Standard OZP (Next Generation) Deployment

Scope

This document covers operational system deployment plans, including
systems intended for pre-production testing (e.g. Integration/Staging/QA systems).

Overview

This document provides details on the intended deployment for OZP within an Enterprise
network.

This information will be useful for the Systems/Network Administration and Operations
teams, but will also be useful for system developers when troubleshooting, or when
identifying usage scenarios for feature development.

Document Layout

The document will be ordered as:

- Deployment Goals
- Deployment Architecture
- Platform Stack
- Component-Platform Mapping
- Deploying OZP

Deployment Goals

- 99.99% Availability
- Minimal Performance Degradation under high load
- Efficient use of the utility resources (minimization of operational costs)
- Capable of handling > 25,000 users simlutaneously

Deployment Architecture

The architecture chosen to acheive the above stated goals was a multi-tiered, highly modular architecture, consisting of:

- Tier 0: Load Balancing
- Tier 1: Web Content Distribution
  - Module A: Static Web Content Distribution
  - Module B: Dynamic Web Content Distribution
- Tier 2: Middleware Analysis & Processing
- Tier 3: Data Persistence
  - Relational Data (RDBMS)

Platform Stack

General

OZONE uses an integration of multiple "stacks" - a collection of independently managed software components, integrated together to form a single, logical service - instead of a single platform stack

Each stack consist of a common base:

- CentOS 6 x86-64

Non-developed Software Used

The following set of (non-developed) software was used to create the stacks associated with each
tier.

- OpenResty/NGINX 1.7.10
- Apache HTTPD 2.4
- Java Development Kit 1.7

  - Note: Preference is for Oracle/Sun's JDK, but OpenJDK will also work

- Apache Tomcat 7.0
- HAProxy 1.5
- MariaDB 5.5
- PHP 5.6

Tier Platform Stacks

The stacks listed below use the non-versioned name of software. Refer to the above list for specific version information.

User Service	Metrics
HAProxy	HAProxy
Apache HTTPD	Apache HTTPD
Apache Tomcat	PHP
MariaDB	MariaDB

Tier 0: Load balancing

This tier uses the following platform stack:

- CentOS
- HAProxy

Tier 1: Web Content Distribution

Module A: Static Web Content Distribution

- CentOS
- Apache HTTPD

Module B: Dynamic Web Content Distribution

- CentOS
- Apache HTTPD

Tier 2: Middleware Analysis & Processing

- CentOS
- Apache Tomcat

Tier 3: Data Persistence

- CentOS
- MariaDB

Deploying OZP

Non-Developed Software Preparation

No special preparation is required.

Developed Software Preparation

In order to maximize the reach of the software, the OZP-Rest component is released using the default security module which does not
provide X.509v3 based PKI authentication and authorization. If this feature is to be used, the OZP-Rest component must be customized.

Additional Software Required

- OZP Security Plugin
  - This plugin will need to be built prior to customization. The resulting JAR file will be used in the customization steps.

Customization Procedure

Customizing OZP Rest for X.509v3 PKI

Deployment Overview

Hardware & Network Requirements

Hardware

The Apps Mall OZP Next Generation Platform deployment has the following hardware requirements:

- Backend Apache Tomcat (JEE) Component
  - 2 VMs with
    - 2 CPUs

        * 8GB RAM
        * 40GB Disk Space

- MariaDB (RDBMS) Component
- 1 VM with
    - 4 CPU
    - 16GB RAM
    - 60GB Disk
- NGINX (Static Web Content Distribution) Component
- 2 VMs with
    - 2 CPU
    - 2GB RAM
    - 60GB Disk
- NGINX (Dynamic Web) Component
- 1 VM with
    - 2 CPU
    - 4GB RAM
    - 70 GB Disk
- HAProxy (Load Balancing) Component
- 2 VMs with
    - 2 CPU
    - 4GB RAM
    - 40GB Disk

Networking

Deployment of the OZP Next Generation platform has been tested within a private VLAN that explicitly exposes various endpoints using Firewalls and Network Address Translations (NAT).

The only requirements for Networking are:

- VLAN/private network
- Firewall with NAT support

Security Considerations & Requirements

All external-faced endpoints will make use of SSL (TLSv1+). In order to provide this function, all external IPs and Hostnames will require an associated SSL certificate from a Trusted Certificate Authority.

Installation & Configuration Overview

1. Deploy MariaDB (RDBMS)
2. Deploy OZP Rest Backend
3. Deploy ElasticSearch Cluster
4. Deploy OZP Rest Web Application
    (a) For Each VM running OZP Rest Web Application:
        i. Deploy Server PKI
        ii. Deploy Apache Tomcat
        iii. Deploy OZP-REST
5. Deploy Metrics Service
    (a) Deploy Server PKI
    (b) Deploy Apache HTTPD with SSL
    (c) Deploy Metrics application
6. Deploy Client Application components (HUD, WebTop, Center, and IWC)
7. For each VM hosting the client components:
    (a) Deploy Apache HTTPD with SSL
    (b) Deploy IWC and Client components
8. Deploy load balancers
    (a) Deploy HAProxy to Static Content LB VM
    (b) Configure HAProxy backends to balance over each VM in step 4.1
    (c) Deploy HAProxy to Rest Backend LB VM
    (d) Configure Rest Backend LB VM to balance over each VM in step 2.2.1

Customizations: OZP Rest WAR

Overview

The OZP Rest WAR is delivered as a FOSS, general purpose web application supporting basic authentication.
For Enterprise deployments using X.509v3 PKI, the web app requires additional customizations.
policies.

Resources

- OZP IAA Plugin

  - Source Code Repository: https://github.com/ozone-development/ozp-security

- Apache HTTPClient 4.3 Java JARs
  - http://archive.apache.org/dist/httpcomponents/httpclient/binary/httpcomponents-client-4.3.6-bin.tar.gz

Prerequisites

- Installed Software:
  - unzip
  - java 1.7+ (OpenJDK is fine)
- ozone-security-*.jar built from sources

Customization Overview

1. Explode WAR file to temporary directory
2. Replace security plugin with OZP IAA Plugin
3. Add Apache HTTPClient JARs
4. Repack WAR

Customization Process

1. Explode WAR file to temporary directory

```
mkdir /tmp/extract/ozp-rest
cd /tmp/extract/ozp-rest
unzip <path-to-ozp-rest-war>
```

2. Replace security plugin with OZP IAA Plugin

```
cd /tmp/extract/ozp-rest
rm -f WEB-INF/lib/ozone-security*.jar
cp <path-to-ozp-security-repo>/target/ozone-security*.jar WEB-INF/lib/
```

3. Add HTTPClient JARs

```
tar -C WEB-INF/lib --strip-components=2 -zxf <path-to-httpclient-archive>/httpcomponent
    httpcomponents-client-4.3.6/lib/httpclient-4.3.6.jar \
    httpcomponents-client-4.3.6/lib/httpmime-4.3.6.jar \
    httpcomponents-client-4.3.6/lib/fluent-hc-4.3.6.jar \
    httpcomponents-client-4.3.6/lib/httpclient-cache-4.3.6.jar \
```

```
httpcomponents-client-4.3.6/lib/httpcore-4.3.3.jar \
httpcomponents-client-4.3.6/lib/commons-logging-1.1.3.jar \
httpcomponents-client-4.3.6/lib/commons-codec-1.6.jar
```

4. Repack WAR

```
jar cf /tmp/extract/ozp-rest-x509.war *
```

Installation and Configuration: Server PKI

Software Versions

- N/A

Prerequisites

- Signed PKI certificate and associated private key are available.
- Java JDK is installed (need keytool)
- OpenSSL is installed

Outcome

- Server PKI certificates and private keys will be installed securely on the server
- Server certificates are packaged in Java Keystore and PKCS12 format

Overview

1. Install signed certificate and private key
2. Create Keystores

Assumptions

- None

Procedure

1. Install signed certificate and private key

```
install -o root -g root -m 444 <path-to-signed-certificate>.crt /etc/pki/tls/certs/$(ho
install -o root -g root -m 400 <path-to-private-key>.key /etc/pki/tls/private/$(hostnam
```

2. Create Keystores

- Create Keystore directory

```
install -o root -g root -m 750 -d /etc/pki/keystores
```

- Create PKCS12 Keystore

```
cd /etc/pki/keystores
openssl pkcs12 -export -out $(hostname --fqdn).p12 -in /etc/pki/tls/certs/$(hostname --
```

NOTE: provide a password for the private key, and then again for the PKCS12 keystore

- Create Java Keystore

```
cd /etc/pki/keystores
keytool -importkeystore -srckeystore /etc/pki/tls/keystores/$(hostname --fqdn).p12 -src
```

NOTE: provide a password for the private key, and then again for the JKS

Installation and Configuration: MariaDB

Software Versions

- MariaDB 5.5.42

Prerequisites

- Available Large disk storage area (>10GB)
- YUM repo with MariaDB RPM
  - Where internet access is available, install the yum repo per the instructions at https://mariadb.com/kb/en/mariadb/yum/
  - For private networks, the following MariaDB RPMs should be downloaded, and loaded into a custom YUM repo

    * MariaDB-5.5.42-centos6-x86_64-client.rpm
    * MariaDB-5.5.42-centos6-x86_64-common.rpm
    * MariaDB-5.5.42-centos6-x86_64-compat.rpm
    * MariaDB-5.5.42-centos6-x86_64-server.rpm
    * MariaDB-5.5.42-centos6-x86_64-shared.rpm
- Root DB user password

Overview

1. Install Package from YUM repo
2. Create Data Storage Folder
3. Configure MariaDB
4. Configure Firewall to allow ingress
5. Configure and Start Service
6. Initialize Database Server
7. Verify MariaDB

Assumptions

- /data is already created with 755 root:root permissions, on a partition with >10GB free space

Procedure

1. Install Package from YUM repo

   ```
   yum install MariaDB-server
   ```

2. Create Data Storage Folder

   ```
   install -o mysql -g mysql -m 775 -d /data/dbs
   ```

3. Configure MariaDB

   - Update /etc/my.cnf with the following entries:
     ```
     [server]
     datadir=/data/dbs
     ```

4. Configure Firewall to allow ingress to node

   - Add the following line to `/etc/sysconfig/iptables` before any REJECT rules

     ```
     `A INPUT -m state --state NEW -m tcp -p tcp --dport 3306 -j ACCEPT`
     ```

   - Reload the rules:

```
`service iptables reload`
```

5. Configure and Start Service

```
chkconfig mysql on
service mysql start
```

NOTE: if you receive a failure when starting, you will need to check permissions on the data directory (/data/dbs) to ensure the `mysql` *user can read and write to it.*

1. Initialize Database Server

```
sudo -l -u mysql mysql_install_db --datadir=/data/dbs
```

2. Verify MariaDB

Installation and Configuration: ElasticSearch Node

Prerequisites

- Java JDK Installed on system
  - Oracle JDK is recommended; OpenJDK 1.7+ is sufficient
- Available Large disk storage area (>10GB)
- YUM repo with ElasticSearch RPM
  - Where internet access is available, the main ElasticSearch repo can be used per these instructions: http://www.elastic.co/guide/en/elasticsearch/reference/1.4/setup-repositories.html#_yum
  - For private networks, the ElasticSearch RPM should be downloaded, and loaded into a custom YUM repo

Overview

1. Install Package from YUM repo
2. Create Data Storage Folder
3. Configure ElasticSearch
4. Configure Firewall to allow ingress to node
5. Configure and Start Service
6. Verify ElasticSearch Service is Running

Assumptions

- The /data partition is pre-created on a disk with >10G available

Notes and Considerations

This particular ElasticSearch deployment uses unicast "discovery" - all nodes must be listed prior to starting the cluster. This method should be usable across all infrastructure platforms, including those that disable multicast (the default discovery method of ElasticSearch).

Use of the multicast/dynamic discovery capabilities is up to the deployer, but will not be described here.

Procedure

1. Install Package from YUM repo

   `yum install elasticsearch`

2. Create Data Storage Folder

   `install -o elasticsearch -g elasticsearch -m 775 -d /data/es`

3. Configure ElasticSearch

   - Search for the associated keys (text before the ':'), uncomment if necessary, and update with the associated value, substituting locally relevant values for anything between angle brackets:
     - `cluster.name: ozp-<dev|int|prod>-search`
     - `index.number_of_replicas: 2`
     - `path.data: /data/es`
     - `path.work: /var/lib/elasticsearch`
     - `path.logs: /var/log/elasticsearch`
     - `bootstrap.mlockall: true`
     - `discovery.zen.ping.multicast.enabled: false`
     - `discovery.zen.ping.unicast.hosts: ["<es-node1", "<es-node2>"]`
       - *Note: List all nodes here, including the one you are configuring*

4. Configure Firewall to allow ingress to node

   - Add the following line to `/etc/sysconfig/iptables` before any REJECT rules

     `` `A INPUT -m state --state NEW -m tcp -p tcp --dport 9200:9400 -j ACCEPT` ``

- Reload the rules:

  `` `service iptables reload` ``

5. Configure and Start Service

   ```
   chkconfig elasticsearch on
   service elasticsearch start
   ```

6. Verify elasticsearch service

   ```
   curl http://localhost:9200/_cluster/health
   ```

Installation and Configuration: Apache Tomcat

Software Versions

- Apache Tomcat 7.0.33

Prerequisites

- YUM repo with Apache Tomcat RPM
  - Use the standard (pre-loaded) and EPEL repositories (`yum install -y epel-release`)
- PKI/SSL Certificates Keystore and Trust Store ready and available

Outcome

Apache Tomcat will be running on the local system on port 8443 and 8009. Connections
on port 8443 will be encrypted using SSL, and 8009 connections will use the Apache AJP
communications protocol.

No web apps will be started or running.

Overview

1. Install Package from YUM repo
2. Configure Tomcat Connector (`/usr/share/tomcat/conf/server.xml`)
3. Configure JVM options for SSL and Heap Memory (`/etc/sysconfig/tomcat`)
4. Configure Firewall to allow ingress
5. Configure and Start Service

Assumptions

- Keystore is stored in `/etc/pki/tls/keystores/<fqdn>.jks` and is password protected
  - Where `<fqdn>` is the fully-qualified domain name of the local host (e.g. appsvr01.example.com).
- Trust Store is stored at `/etc/pki/CA/ozp-trusts.jks` and is password protected

Procedure

1. Install Package from YUM repo

   ```
   yum install -y tomcat
   ```

2. Configure Tomcat Connector (/usr/share/tomcat/conf/server.xml)

   - Under the `<Service name="Catalina">` section in `/usr/share/tomcat/conf/server.xml`, add the following entry:

     ```
     <Connector
         protocol="HTTP/1.1"
         port="8443" maxThreads="200"
         scheme="https" secure="true" SSLEnabled="true"
         keystoreFile="/etc/pki/tls/keystores/<fqdn>.jks"
         keystorePass="<keystore-password>"
         truststoreFile="/etc/pki/CA/ozp-trusts.jks"
         truststorePass="<truststore-password>"
         clientAuth="want" sslProtocol="TLS"/>
     ```

   NOTE 1: We need to set `clientAuth` *to "want" instead of "true" due to the use of the HTTP OPTIONS request, which, by the official standard, will* not *pass any kind of authentication information; it is expected to be able to anonymously request OPTIONS.*

   NOTE 2: The AJP connector is already available in the default server.xml file. No additional configuration is required to enable it.

3. Configure JVM options for SSL and Heap Memory (`/etc/sysconfig/tomcat`)

   - Add the following entries to the `/etc/sysconfig/tomcat` file:

     ```
     export JAVA_OPTS="${JAVA_OPTS} -Xms2g -Xmx2g -XX:MaxPermSize=512m"
     export JAVA_OPTS="${JAVA_OPTS} -Djavax.net.ssl.keyStore=/etc/pki/tls/keystores/
     export JAVA_OPTS="${JAVA_OPTS} -Djavax.net.ssl.keyStorePassword='"$(echo <keyst
     export JAVA_OPTS="${JAVA_OPTS} -Djavax.net.ssl.trustStore=/etc/pki/CA/ozp-trust
     export JAVA_OPTS="${JAVA_OPTS} -Djavax.net.ssl.trustStorePassword=<truststore-p
     ```

4. Configure Firewall to allow ingress to node

   - Add the following line to `/etc/sysconfig/iptables` before any REJECT rules

     ```
     A INPUT -m state --state NEW -m tcp -p tcp --dport 8443 -j ACCEPT
     A INPUT -m state --state NEW -m tcp -p tcp --dport 8009 -j ACCEPT
     ```

   - Reload the rules:

     ```
     service iptables reload
     ```

5. Configure and Start Service

   ```
   chkconfig tomcat on
   service tomcat start
   ```

Installation & Configuration: ozp-rest

Prerequisites

- Customized ozp-rest WAR
- PKI certificates for SSL
- ElasticSearch Cluster deployed and accessible from ozp-rest host
- MariaDB setup with
  - empty database `ozp` created
  - User `ozpuser` created with full privileges on `ozp` database, and access allowed from OZP-Rest host(s)
  - You have the connection information for `ozpuser`
- Access to MariaDB client `mysql` (from a separate host is preferred).

Resources

- Database Schema @
- Initial Database data @

Deployment Overview

1. Install Apache Tomcat
2. Install OZP server-level configuration files
3. Configure SSL on Tomcat
4. Load Schema into MariaDB
5. Load initial data into MariaDB
6. Install (deploy) the customized ozp-rest WAR
7. Start Tomcat

Deployment Detailed Instructions

Notes:

- Perform all steps as root
- For CentOS 6, the directory is normally located at /usr/share/tomcat

1. Install Apache Tomcat

   ```
   yum install -y tomcat
   ```

2. Install OZP server-level configuration files

   - /lib/SecurityConfig.properties
   - /lib/SecurityContext.xml
   - /lib/MarketplaceConfig.groovy

3. Configure SSL on Tomcat

   - Configure the JVM to use the PKI certificates
   - This requires setting the `javax.net.ssl.*` properties, which will make
     the client certificate accessible to web apps
   - This allows the web app to call out to PKI authenticated services
   - Configure the HTTPS connector in server.xml
   - This provides the ingress encryption (Client -> Tomcat server)

4. Load Schema into MariaDB

   ```
   mysql -u ozpuser -h <mariadb-host> -D ozp < <DB-Schema-File>
   ```

5. Load Initial Data into MariaDB

   ```
   mysql -u ozpuser -h <mariadb-host> -D ozp < <DB-Data-File>
   ```

6. Install (deploy) the customized ozp-rest WAR

   - From a web location

     ```
     wget -O <tomcat-root>/webapps/ozp-rest.war http://infra-adm01/cfgmgmt/apps/ozp-rest
     chown tomcat:tomcat <tomcat-root>/webapps/ozp-rest.war
     ```

   - From a local download of the WAR

     ```
     install -o tomcat -g tomcat -m 644 <local-path>/ozp-rest.war
     <tomcat-root>/webapps/ozp-rest.war
     ```

7. Start Tomcat and enable auto-start

   ```
   chkconfig tomcat on
   service tomcat start
   ```

8. Check for errors and attempt to resolve

    - Review the following logs in /logs/
        - `catalina.out`
        - `marketplace.log`
        - `stacktrace.log`
        - `localhost.<date>.log`

Installation and Configuration: Apache HTTPD (secured)

Software Versions

- Apache HTTPD 2.2

Prerequisites

- Server PKI is installed per [install-server-pki] instructions
- Server PKI (private key and certificate) is accessible by the apache user.
- Certificate Authority file (CA Bundle - all desired CA's listed)
- Available disk storage area is available outside of the /var partition for use as a DocumentRoot location (content, media, & data files)

Outcome

Apache HTTPD is running and serving a secure virtual host on port 443 (default HTTPS port), which is configured
to only allow 2-way SSL connections.

Overview

1. Install Apache HTTPD with mod_ssl
2. Create DocumentRoot directory
3. Configure HTTPD - Basics
4. Configure HTTPD - SSL
5. Configure Firewall to allow ingress to node
6. Configure and start HTTPD

Assumptions

- CentOS 6 using YUM for package management
- iptables is in use and running and already configured to allow existing connections
- Available disk storage filesystem root is at /data mount point, which is owned by the `root` user.
- Certificate Authority/Trust Store is name `/etc/pki/CA/ozp-ca.crt`
- As a general process, the Private Server Key is assumed to be passphrase-less, but access controlled such that only the
  `apache` user can access it.
  - It is recommended that a stronger security mechanism is in place for production deployments - e.g. passphrase protected private key, and use
    of a mechanism to provide the passphrase to Apache HTTPD

Procedure

NOTE: substitute fully qualified domain name of host for instances of `<FQDN>` in procedures below

1. Install Apache HTTPD with mod_ssl
   `yum install httpd mod_ssl`
2. Create DocumentRoot directory
3. Create root directory on a filesystem separate from /var, owned by `apache` user
   `sudo install -o apache -g apache -m 0755 -d /data/www`
4. Copy the default document root layout and files to new document root directory
   `sudo cp -pr /var/www/* /data/www`
5. Configure HTTPD - Basics
   (a) Modify main HTTPD configuration file `/etc/httpd/conf/httpd.conf`:
   - Search and replace all instances of `/var/www/html` with `/data/www/html`
6. Configure HTTPD - SSL
7. [Optional] Generate a private key without a passphrase for the `apache` user only:

```
openssl rsa -in /etc/pki/tls/private/<FQDN>.key -out /etc/pki/tls/private/<FQDN>.nopp.k

chmod 400 /etc/pki/tls/private/<FQDN>.nopp.key

setfacl -m user:apache:r /etc/pki/tls/private/<FQDN>.nopp.key
```

8. Modify `/etc/httpd/conf.d/ssl.conf`:
   - Set the protocol and encryption levels to high
     ```
     SSLProtocol all -SSLv2
     SSLCipherSuite HIGH:!aNULL:!MD5
     ```
   - Set the SSL certificate and Trusts:
     ```
     SSLCertificateFile /etc/pki/tls/certs/<FQDN>.crt

     SSLCertificateKeyFile /etc/pki/tls/private/<FQDN>.nopp.key
     SSLCACertificateFile /etc/pki/CA/ozp-ca.crt
     ```
9. Configure Firewall to allow ingress to node
   - Add the following line to `/etc/sysconfig/iptables` before any REJECT rules
     ```
     A INPUT -m state --state NEW -m tcp -p tcp --dport 443
     -j ACCEPT
     ```
   - Reload the rules:
     ```
     service iptables reload
     ```
10. Configure and Start Service
    ```
    chkconfig httpd on
    service httpd start
    ```

Installation and Configuration: OZP IWC and Client Components

Software Versions

- Apache HTTPD 2.2/OpenResty 1.7.0
- OZP WebTop 1.0
- OZP Center 1.0
- OZP HUD 1.0
- OZP IWC 1.0

Prerequisites

- Either NGINX/OpenResty (secured) or Apache HTTPD (secured) is installed on the server
- URL for
- OZP primary entrypoint
- OZP Rest backend
- OZP Metrics Service

Outcome

The OZP IWC, WebTop, Center, and HUD mobile code will be available for clients to download
and run from a secure Apache HTTD instance. Client X.509 certificates will be required
to access and run the mobile code components.

Notes

- It is expected that the URL used for all of the services is an alias (CNAME) that points to a
  loadbalancer.

Overview

1. Create OZP Primary Configuration
2. Install IWC
3. Install OZP WebTop, Center, and HUD
4. Configure IWC
5. Configure OZP WebTop, Center, and HUD

Assumptions

- CentOS 6 using YUM for package management
- IWC and Client Components are the packaged distributions (tar/gzipped), not a copy of the source code

Notes about Procedure

The procedure below uses the following variables:

Variable	Represents	Example
@PRIMARY_URL@	Primary URL to service	https://ozp.example.com
@BACKEND_URL@	Root URL for OZP-Rest	https://ozp.example.com/rest
@METRICS_URL@	URL for Metrics service	https://metrics.example.com
@FEEDBACK_EMAIL@	Email address for user feedback	ozp-support@ozp.example.com
@DEV_URL@	URL to development info/resources	https://ozp-dev.example.com

Procedure

1. Create OZP Primary Configuration
   (a) Create a file named `OzoneConfig.js` with the following content:

```
window.OzoneConfig = {
      'API_URL': '@BACKEND_URL@/api',
      'CENTER_URL': '@PRIMARY_URL@/center',
      'DEVELOPER_RESOURCES_URL': '@DEV_URL@',
      'FEEDBACK_ADDRESS': '@FEEDBACK_EMAIL@',
      'HUD_URL': '@PRIMARY_URL@/hud',
      'IWC_URL': '@PRIMARY_URL@/iwc',
      'METRICS_URL': '@METRICS_URL@',
      'WEBTOP_URL': '@PRIMARY_URL@/webtop',
      'HELP_DOCS': {
          'Help doc 1': '/path/to/document',
          'Help doc 2': '/path/to/document',
          'Help doc 3': '/path/to/document',
          'Help doc 4': '/path/to/document',
          'Help doc 5': '/path/to/document'
      },
      'HELP_VIDEOS': {
          'Video 1': '/path/to/video',
          'Video 2': '/path/to/video',
          'Video 3': '/path/to/video',
          'Video 4': '/path/to/video',
          'Video 5': '/path/to/video'
      }
  };
```

2. Install IWC
3. Create the IWC entrypoint under the web server document root
   `install -m 755 -o root -g root -d /data/www/html/iwc`
4. Move the contents of the dist subfolder to the IWC entrypoint

   `tar -C /data/www/html/iwc --strip-components=1 -zxf <path-to-iwc-archive>`
5. Install OZP WebTop, Center, and HUD

6. Create the WebTop, Center, and HUD entrypoints under the web server document root

   ```
   install -m 755 -o root -g root -d /data/www/html/{webtop,hud,center}
   ```
7. Extract the contents of the WebTop archive to the `webtop` folder

   ```
   tar -C /data/www/html/webtop --strip-components=1 -zxf
   <path-to-webtop-archive>
   ```
8. Extract the contents of the Center archive to the `center` folder

   ```
   tar -C /data/www/html/center --strip-components=1 -zxf
   <path-to-center-archive>
   ```
9. Extract the contents of the HUD archive to the `hud` folder

   ```
   tar -C /data/www/html/hud --strip-components=1 -zxf <path-to-hud-archive>
   ```
10. Set permissions on folders
    ```
    setfacl --recursive -m user:apache:rx /data/www/html
    setfacl --recursive -d -m user:apache:rx /data/www/html
    ```
11. Configure IWC
12. Modify the `ozpIwc.apiRootUrl` variable in `/data/www/html/iwc/iframe_peer.html`
    ```
    ozpIwc.apiRootUrl=@BACKEND_URL@/api
    ```
13. Configure OZP Webtop, Center, and HUD
14. Overwrite the `OzoneConfig.js` file in each of the component top-level directories with the one created in step 1.

    ```
    cp <path-to-custom-ozoneconfig.js> /data/www/html/webtop/OzoneConfig.js

    cp <path-to-custom-ozoneconfig.js> /data/www/html/center/OzoneConfig.js

    cp <path-to-custom-ozoneconfig.js> /data/www/html/hud/OzoneConfig.js
    ```

Installation & Configuration: ozp-metrics

Prerequisites

- If using CentOS/RedHat 6, you will need to locate a 3rd party PHP 5.6 YUM repository
- SSL Server Certificate (including Private Key)

Resources

- PHP 5.6 for CentOS/RedHat 6.6 can be installed using the Webtatic EL repository

** Instructions at https://webtatic.com/packages/php56/
- OZP-Metrics @

Assumptions

- The OS is CentOS 6
- EPEL repository is enabled
- The instructions below will assume the use of the Webtatic PHP 5.6 packages on CentOS 6

Deployment Overview

1. Install Apache HTTPD 2.2 with mod_ssl
2. Install PHP 5.6
3. [Optional] Create a web server document root directory
4. Configure Apache HTTPD - primary config
5. Configure PHP for Apache HTTPD
6. Configure Apache HTTPD SSL
7. Deploy Metrics application to Document Root
8. Configure Metrics
9. Start Apache HTTPD
10. Verify Metrics

Deployment Steps

1. Install Apache HTTPD 2.2 with mod_ssl
   `sudo yum install httpd mod_ssl`
2. Install PHP 5.6 modules

   ```
   sudo yum install php56w php56w-cli php56w-mbstring php56w-pdo
   php56w-gd php56w-common php56w-mysql php56w-xml
   ```
3. [Optional] Create a web server document root directory
   `install -m 755 -d /data`

   ```
   install -m 755 -o apache -g apache -d /data/www/{cgi-bin,html,error,icons}
   ```
4. Configure Apache HTTPD - primary config

- Open /etc/httpd/conf/httpd.conf for editing
  - [If step 3 was done] Replace all instances of default document root `/var/www/html` with custom document root `/data/www/html`
  -

Installation and Configuration: OZP IWC and Client Components

Software Versions

- haproxy 1.5.2

Prerequisites

- URL for

Outcome

The OZP IWC, WebTop, Center, and HUD mobile code will be available for clients to download
and run from a secure Apache HTTD instance. Client X.509 certificates will be required
to access and run the mobile code components.

Notes

Overview

1. Install haproxy
2. Configure haproxy
3. Start haproxy

Assumptions

- CentOS 6 using YUM for package management

Notes about Procedure

Procedure

1. Install haproxy
   ```
   sudo yum install -y haproxy
   ```
2. Configure haproxy