Hewlett Packard Enterprise

HPE Propel

Software version 2.20.p2

Distributed Propel Deployment Guide

Contents

Documentation release date: February, 2017

Software release date: July 2016

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hpe.com/.

This site requires that you register for an HPE Passport and to sign in. To register for an HPE Passport ID, click **Register** on the HPE Software Support site or click **Create an Account** on the HPE Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

Use the Search function to find documentation, whitepapers, and other information sources. To learn more about using the customer support site, go to: https://softwaresupport.hpe.com/documents/10180/14684/HP_Software_Customer_Support_Handbook/

Support

Visit the HPE Software Support site at: https://softwaresupport.hpe.com.

This website provides contact information and details about the products, services, and support that HPE Software offers.

HPE Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HPE support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HPE Passport user and to sign in. Many also require a support contract. To register for an HPE Passport ID, click **Register** on the HPE Support site or click **Create an Account** on the HPE Passport login page.

To find more information about access levels, go to: https://softwaresupport.hpe.com/web/softwaresupport/access-levels.

HPE Software Solutions Now accesses the HPE Software Solution and Integration Portal website. This site enables you to explore HPE Product Solutions to meet your business needs, includes a full list of Integrations between HPE Products, as well as a listing of ITIL Processes. The URL for this website is https://www.hpe.com/us/en/solutions.html.

Introduction

This document provides detailed steps on how to set up multiple HPE Propel 2.20 nodes for Distributed Propel Clustering using free and open source Ansible Playbooks technology. The intended audience of this document is enterprise technical IT staff that install, configure and support the Propel System.

Terminology

The following table contains descriptions of some common terms that you will find throughout this document.

Acronym / Terminology	Description
Ansible playbooks	Ansible playbooks are an open source software language that make it easier for an enterprise to run installation and setup scripts. For more information, see http://docs.ansible.com/ansible/playbooks.html.
DB VIP	Virtual IP address that does not correspond to an actual physical network interface (port). Primary use is by pgpool as a floating IP address.
Distributed Propel Cluster	A term used in this document to describe a cluster of Propel System servers (nodes) configured in such a way as to function as a single logical unit. The cluster provides both High Availability and Scalability of the Propel system.
Load balancer	A load balancer acts as a reverse proxy and distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) and reliability of applications.
Master and slave databases	PostgreSQL refers to a multiple node database setup as Master / Slave.
NGINX	High-performance load balancer open source software. The default supported configuration of Distributed Propel uses this product. www.nginx.com/load-balancer
OO	HPE Operations Orchestration. Enables enterprise scale IT process automation. This product is used by Propel.
OVA	A tar archive file package that contains an OVF directory of files. Open Virtualization Format (OVF) files are used for packaging and distributing virtual appliances (software) to be run in virtual machines.
pgpool	Middleware that supports PostgreSQL to provide connection pooling and replication.
PostgreSQL	Open source object oriented relational DBMS. www.postgresql.org
Propel DB Node – High Availability	If a Propel DB node or network route/connection to a node goes down in planned or unplanned outage, the Propel system is still available to users. If Master DB node breaks down, fault is automatically detected and the Slave is automatically promoted to Master, with no down time.
Propel Node - High Availability	When a Propel server (node) or network route/connection to a node or a service instance goes down in planned or unplanned outage, the Propel system is still available to users. Since session information is stored in the database, when failover/back happens there is no need for the user to re-login.

Propel Scalability	The ability to add Propel nodes to increase the scale of the Propel system. Nodes are typically added to either increase the number of users that can be supported or the volume of transactions that can be processed.
RabbitMQ	Open source message-broker software that implements Advanced Message Queueing Protocol (AMQP). www.rabbitmq.com

Distributed Propel cluster concepts

This chapter provides you with some important concepts in regard to a Distributed Propel Cluster.

Overview

There are a minimum of five nodes recommended for Distributed Propel: one load balancer (VIP), two Propel application nodes and two Propel database nodes. You can add additional Propel nodes as needed. A DB VIP must also be set up.

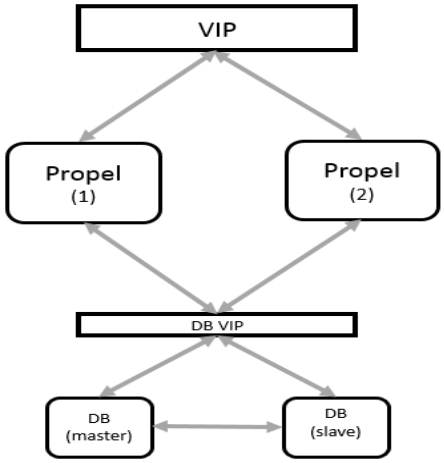


Figure 1 – Distributed Propel minimum configuration

Propel services communicate with each other over HTTP (RESTful) APIs. This allows the services to communicate with each other over the load balancer and enables further resilience inside of the Propel application stack.

The Propel PostgreSQL database will be clustered to provide redundancy. The default setup enables replication between two PostgreSQL DB nodes and provides master-slave automatic failover to the slave if the master goes down. HPE is not providing the enablement of a High Availability database which would imply scalability beyond two database nodes. Instead, we are only supporting a master-slave configuration.

SSL is important to the Propel system. Managing and generating signed certificates is important for the security of the system. The configuration implemented in the Distributed Propel scripts enables communication to always be encrypted. The default setup ensures that encryption is still enabled, but allows for self-signed certificates. In production systems, HPE recommends that only Certificate Authority-signed and trusted certificates are used.

Distributed Propel 2.20 configuration

After performing the steps in this document, the final Distributed Propel 2.20 configuration will resemble the following:

- On the Load Balancer node, only NGINX will run. During Distributed Propel configuration, this node is used to run the Ansible playbook scripts. The load balancer node is visible in the Propel Diagnostics. Diagnostics provides valuable information of the services that are accessible through the load balancer.
- On the two Propel application nodes, all Propel application services will run. The PostgreSQL instances should not run.
- On the two Propel DB nodes, all Propel services, OO and IDOL will be disabled. These nodes are only used for DB purposes and are clustered using pgpool-II.

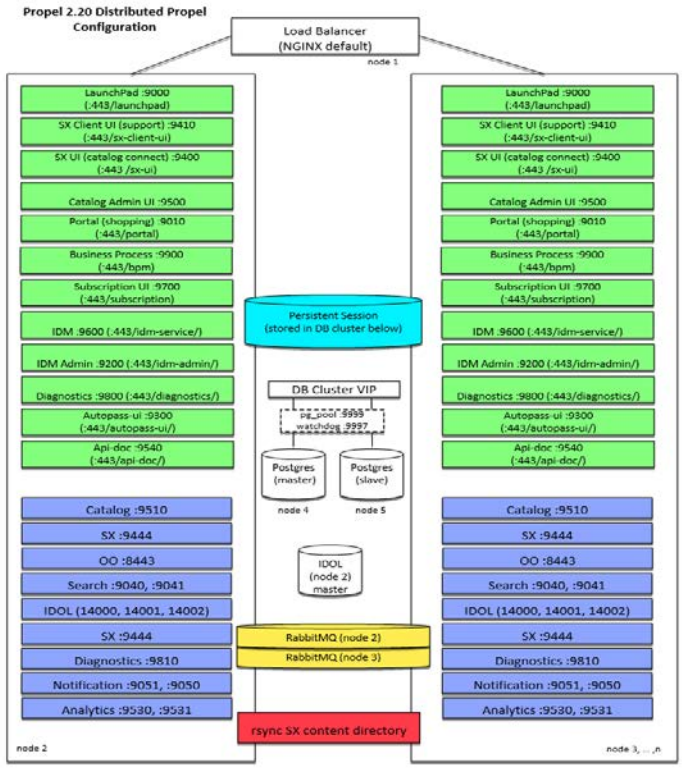


Figure 2: A Propel cluster with Load Balancer node, primary DB node, standby DB node, and two Propel nodes

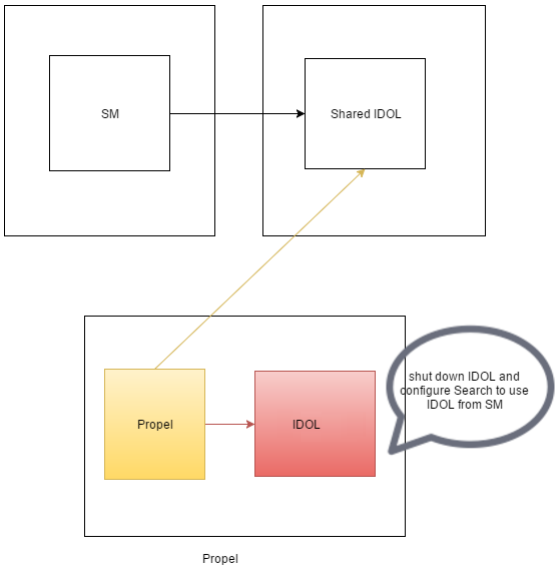
IDOL content servers

Deployment configurations for IDOL

There are two deployment configurations available for Propel and its use of IDOL:

1. IDOL is installed on Propel application node servers.
    A. The HPE Propel Ansible Playbook configures Mirrored IDOL as explained in the section right below.
2. Propel used in conjunction with Service Manager (SM) 9.50 and IDOL is installed on Service Manager application nodes.
    A. For details on the IDOL configuration in Service Manager, refer to the document *Service Manager 9.50 Sizing and Deployment Guide*.
    B. Change the Propel Search service on all application nodes to reference the Service Manager Shared IDOL instance. This should be done after all the installation steps have been completed in this document.
        a) For details on how to perform these steps, refer to the chapter *Configuring HPE Propel for Smart Analytics (IDOL)* in the *Propel Admin Guide*.

Figure 3 An HPE Propel configuration using shared IDOL instance on the Service Manager system



Mirrored IDOL in Propel

The default Propel configuration is to have each Propel node maintain its own instance of IDOL. This is problematic in a Distributed Propel system because the IDOL instances are not synchronized in any way. This document provides configuration changes that mirror the IDOL content servers so they are synchronized across multiple Distributed Propel nodes.

A distributed IDOL setup is composed of a few different components, the important ones for this discussion are as follows: Distributed Index Handler (DIH), Distributed Action Handler (DAH), and Content Servers. Mirroring the data between IDOL content servers allows for high availability by having multiple identical servers. If one server goes down, another exact failover copy goes online seamlessly.

The figures below show a simple mirroring architecture where the DIH replicates all changes to every IDOL Content Server. In this solution, the DAH uses a single IDOL instance as a primary server and only uses others as a backups if the primary fails. All actions are queued for the failed servers. Once the failed servers comes back up, all queued actions are replayed, but the primary server is not reverted to the original.

Figure 4 Before failure

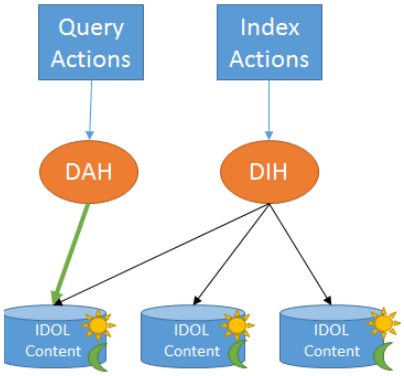


Figure 5 During failure

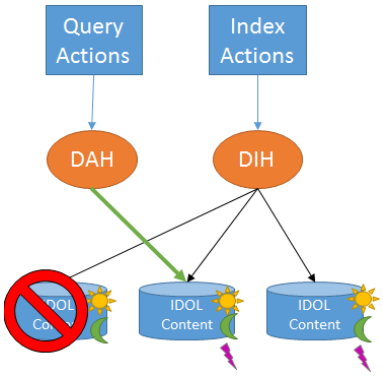
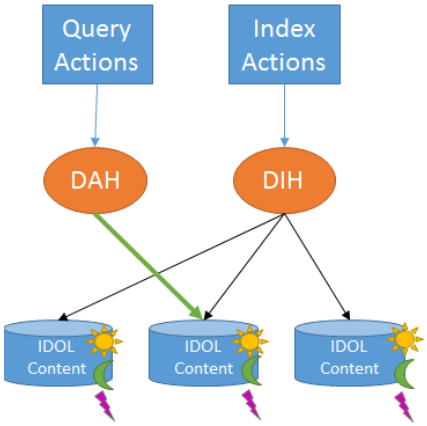


Figure 6 After failure



Load Balancing

The default of the Distributed Propel installation is to use an identical IDOL configuration on each node and rely on NGINX for load balancing. By configuring each instance of the Propel Search service to point to its local IDOL instance (that is identically configured), Search will point to the same IDOL server no matter which instance of Search is used, and seamlessly failover to the same IDOL if the primary goes down.

IDOL Configuration Changes

The IDOL configuration is identical on all the Propel nodes.

Note: It is important to use the actual IP addresses or hostnames of the computers in the configurations. Do not use 127.0.0.1 or localhost so that the configuration files can be copied across servers without errors.

The Distributed Propel scripts will implement the following changes to the main IDOL server configuration file /opt/hp/SmartAnalytics/IDOL/IDOLServer.cfg on each Propel node:

1. Whitelist all the IP addresses of all servers. In this example IP addresses 1.1.1.1, and 2.2.2.2 have been added in 5 locations:

```
[Service]

ServicePort=14002

//the IP address of clients that are permitted to send queries to IDOL.

//they should be altered to your Service Manager server address, and admin ip address range

ServiceStatusClients=*,127.0.0.1,::1,127.0.0.1,1.1.1.1,2.2.2.2

ServiceControlClients=*,127.0.0.1,::1,127.0.0.1,1.1.1.1,2.2.2.2

Access-Control-Allow-Origin=*


[Server]

. . .

//------------------------ Clients Settings-------------------------//

//the IP address of clients that are permitted to send queries to IDOL.

//they should be altered to your Service Manager server address, and admin ip address range

QueryClients=*,127.0.0.1,::1,127.0.0.1,1.1.1.1,2.2.2.2

AdminClients=*,127.0.0.1,::1,127.0.0.1,1.1.1.1,2.2.2.2

IndexClients=*,127.0.0.1,::1,127.0.0.1,1.1.1.1,2.2.2.2
```

2. Enable MirrorMode and remove any non-MirrorMode configurations. Here, MirrorMode has been set to true, all the DIH settings have been commented out, added "DistributionMethod=0.

```
//------------------------ Distributed Architecture -------------------------//

//To set up a distributed IDOL Server in non-mirrormode,

//please refer to the IDOL Server, DAH and DIH manual


[DistributionSettings]

StoredStateTokenLifetime=0

mirrormode=True

//DIH settings:

#DistributeByFields=True

#DistributeByFieldsCSVs=*/ContentStore

#UnknownFieldValueAction=Default

#UnknownFieldValueDefaultEngine=0

#DistributeOnMultipleFieldValues=True

//The following parameter is required for the DAH if mirrormode is set to false

#VirtualDatabases=1

#UseEngineAlias=true
```

```
// DAH Settings
DistributionMethod=0
```

3.Configure IDOLServers and remove Virtual Database Configurations.

- The "Number" field MUST equal the total number of Configured IDOL Servers.
- Each [IDOLServerN] section must be sequential starting at 0.
- IDOL will choose the first server (the one that has the smallest sequential number) as the primary.
- Do not use localhost or 127.0.0.1 for the Host so that the file can be copied to the servers successfully.

```
//This section is equivalent to the [engines] section
// in the DAH and DIH standalone configuration

[DistributionIDOLServers]
Number=2

[IDOLServer0]
Name=Content-Propel
Host=1.1.1.1
Port=10020

[IDOLServer1]
Name=Content-Propel-Failover-1
Host=2.2.2.2
Port=10020

// Remove or comment out all [vdbX] sections
```

4. As a precautionary measure, IDOL does not allow a user to switch from MirrorMode to non-MirrorMode and vice versa because it will result in data loss. Delete the /SmartAnalyitcs/IDOL/dih/main directory on each of the servers before restarting the service will result in losing all your data (indexes) stored in IDOL.

   - In the context of Propel, a simple catalog reindex will suffice for re-populating data indexes in IDOL:

     ```
     # cd /opt/hp/propel/catalog
     # java -jar lib/catalog.jar reindex
     ```

5.Restart the idol services

```
# cd /opt/hp
# servicectl restart idolserver
```

For testing purposes, you can disable a single Propel IDOL content instance to manually initialize the failover:

```
# cd /opt/hp
# servicectl stop idol-content-propel
```

Note: Currently the NGINX configuration does not have a healthcheck for both the Search service and DAH/DIH endpoints. Until the healthchecks are correctly configured, this solution will not support high availability if the DAH/DIH fails.

Setting up a Distributed Propel Cluster

This section provides details on how to set up a Distributed Propel cluster. In this document, it is assumed that five server (nodes) will be set up, but additional Propel cluster nodes could be added. (See Figure 2.)

Note: The Ansible playbook scripts are installed onto and run from the load balancer server.

High Level Overview of Process

This section provides an overview of the steps that must be performed to set up a Distributed Propel cluster.

1. Set up the five (or more) servers (nodes) to be used in your Distributed Propel Cluster:
    a. Install the Propel 2.20 OVA on all instances and upgrade to the latest 2.20 patch release. Alternatively, install the RHEL Propel 2.20.p2 on all instances.
    b. Configure the network.
2. Run the Distributed Propel Ansible Playbook.

System Prerequisites

The following prerequisites must be in place before you can start to follow the steps in this document to set up a Distributed Propel cluster. Obtain these packages from the corresponding HPE web site.

- Propel 2.20 OVA
- If applicable, the latest available patch and/or hotfixes for:
    - Propel 2.20 OVA or HPE Propel RHEL 2.20.p2
    - Distributed Propel 2.20 Ansible playbooks
- Five (or more) VMs available to provision for the Distributed Propel Cluster – see the HPE Propel 2.20.p2 document *Support Matrix* on HPE SSO for minimum hardware and software specifications.
- All virtual machines must have lower case hostnames. For example, PROPEL.com is NOT allowed, use propel.com instead.

Preparing all the nodes in the cluster

The following steps must be performed on each server (load balancer, Propel and DB nodes) in the Distributed Propel Cluster:

1. Provision each node in the cluster by deploying the HPE Propel 2.20 OVA or the RHEL Propel 2.20.p2 installer.
2. Configure networking for the HPE Propel VM:

    ```
    # cd /opt/hp/propel-install
    ```

    ```
    # ./configureNetwork.sh --hostname <Simple-Host-Name> --configuredhcp
    ```

    (or with static IP)
    The script will ask for a reboot. Select "Yes"
3. Configure SSL on the HPE Propel VM:

    ```
    # ./propel-ssl-setup.sh auto --hostname <FQDN> 2>&1 | tee ssl-setup.log
    ```
4. Run the HPE Propel installation utility:

    ```
    # ./setup.sh install <FQDN> 2>&1 | tee install.log
    ```
5. Patch each node to the latest available HPE Propel 2.20 Patch and Hotfix release.
6. Start Propel on each node:

    ```
    # propel start
    ```
7. Run the following command on each Propel application node as either root or rabbitmq user:

    ```
    # rabbitmqctl set_policy ha-all "notification-svc-queue" '{"ha-mode":"all","ha-sync-mode":"automatic"}'
    ```

8. If each node was deployed using the RHEL Propel 2.20.p2 installer, then perform these additional steps:

    A. Configure the propel user on all nodes.

        a) Password - If the propel user does not exist on the host before running the Propel setup.sh script, it will be created, however without the standard password. Before running the ansible playbooks, check that the propel user has a known password, which needs to be provided when the playbook starts. As root, run passwd propel and set the new password as prompted.

        b) Sudo - the propel user will also need to have sudo privileges. On Redhat, this should be using visudo as root. Find the line

        ```
        root ALL=(ALL) ALL
        ```

        Copy it to a new line right below it, and change the user to propel.

        ```
        propel ALL=(ALL) ALL
        ```

        Test that the propel user has sudo privileges by logging in over SSH with that user account and then run a command using sudo.

        Note: WE STRONGLY RECOMMEND USERS TO REMOVE THIS LINE FROM ALL MACHINES ONCE DISTRIBUTED PROPEL IS SUCCESSFULLY INSTALLED. LEAVING THIS LINE IN PRODUCTION CAN MAKE PROPEL MACHINES VULNERABLE.

Verify that you can launch Propel on all Propel_Cluster_Nodes

Before running the Distributed Propel scripts, you must ensure that Propel can be launched and standard workflows can be executed. At a minimum, run through the following workflows:

- As a super admin, launch each application on the Launchpad and verify they are working correctly. Go to the Diagnostics application and make sure that all components are healthy.
- As an organization admin, launch each application on the Launchpad and verify they are working correctly.
- As a consumer, launch Propel, shop for catalog items and place an order.

Note: After your patched 2.20 system has been verified to be working, it is highly recommended that you take a snapshot of your Distributed Propel Cluster nodes.

Preparing the Load Balancer server

The following steps must be performed to prepare the Load Balancer server. Connect to the LB node as user propel and run the following commands:

1. Released versions of Propel 2.20.x include the initial version of the Distributed HPE Propel Ansible playbooks. If there is a patch and hotfix for the Distributed HPE Propel 2.20 Ansible playbooks, copy the patch and hotfix to the folder /opt/hp/propel/contrib and untar it as follows (substitute appropriate file name below in the tar command):

    ```
    # cd /opt/hp/propel/contrib
    ```

    ```
    # tar xvf <Propel-2-20-Distributed-Propel-hotfix_or_patch>.tar
    ```

    The tar file contains the following sub-directory that will be extracted under:

    ```
    /opt/hp/propel/contrib:
    ```

    ```
    /propel-distributed.<VERSION>
    ```

2. Check the network connectivity and get the hosts keys. Verify the network connectivity between the Load Balancer Node (acting as Ansible Management Node) and all the Propel Node servers by making an ssh connection from the Load Balancer to the Propel servers (Cluster and DB nodes) using the FQDN. Remember to make an ssh connection to the Load Balancer server as well.
   The command ssh-copy-id:

    - Copies the local-host's public key to the remote-host's authorized_keys file.

- Assigns proper permission to the remote-host's home, ~/.ssh, and ~/.ssh/authorized_keys.
- Generate the private and public key (press Enter twice to create keys with no passphrase):

```
# ssh-keygen -t rsa -f ~/.ssh/id_rsa
```

- Run the ssh-copy-id commands for each node:
  Example connections:

```
# ssh-copy-id propel@propel-lb.swlab.net
# ssh-copy-id propel@propel-s1.swlab.net
# ssh-copy-id propel@propel-s2.swlab.net
# ssh-copy-id propel@propel-db1.swlab.net
# ssh-copy-id propel@propel-db2.swlab.net
```

  Note: After completing this step, ssh calls should be run without a password prompt.

3. Navigate to the /opt/hp/propel/contrib/propel-distributed.<version> folder.
4. Define Ansible Nodes (Hosts).
   - Copy the inventory/hosts.example file to inventory/hosts.default

```
# cp inventory/hosts.example inventory/hosts.default
```

   - In inventory/hosts.default, change fully qualified hostnames of all cluster nodes in [lb], [propel], [db_m], and [db_s] sections, IP address of the load balancer node in [lb_ip] section and virtual IP address of database cluster in [db_vip] section to values describing your actual configuration. Detailed descriptions can be found as follows.
     Node role descriptions:
     - [lb] - front-end load balancer address
     - [lb_ip] - IP address of the load balancer
     - [propel] - all Propel cluster nodes within the Propel cluster
     - [db_m] - Propel master DB node within the Propel cluster (one)
     - [db_s] - Propel slave DB node within the Propel cluster
     - [db_vip] - The VIP address for the PostgreSQL cluster. VIP is a virtual IP, so this is an address that uses a virtual adaptor on the pg_pool cluster. Pg_pool has a watchdog service that will float the IP between the cluster nodes to give a connection we can count on. This unused IP should be ping-able, reachable within the same subnet as the HPE Propel and DB nodes and will be linked to the primary Ethernet port (eth0) of the HPE Propel DB nodes
     - [*:children] - Support roles. Please do not change unless you know what you are doing.
5. Check your Ansible Node Hosts file.
   Verify that your Ansible node hosts file is set up correctly and recognized by Ansible. Run the following commands and verify the results.

```
# cd /opt/hp/propel/contrib/propel-distributed.<version>
# ansible propeld -u propel -m ping -c paramiko
```

   Note: For every host you may be asked if the fingerprint is correct. Type 'yes' and press enter. Next time, this command should finish without user input. The script should finish in seconds. If it takes longer time, it might be waiting for your input.

6. Copy the group_vars/propeld.yml.example file to group_vars/propeld.yml.

```
# cp group_vars/propeld.yml.example group_vars/propeld.yml
```

7. Online installation. Please follow this step if your cluster has access to the Internet:

   - Update Proxy Settings in group_vars/propeld.yml according to your corporate proxy settings:

     ```
     # vim group_vars/propeld.yml


     proxy_env:
       http_proxy: http://proxy.example.com:8080
       https_proxy: http://proxy.example.com:8080
     ```

8. Offline installation: Please follow these steps if you are installing Propel in an environment without access to the Internet. If you are installing Propel on custom RHEL machine please follow step 9↓.

   - Set *propeld.packages.download* to false in group_vars/propeld.yml.

     ```
     # vim group_vars/propeld.yml
     ```

   - Copy Propel Distributed scripts to the computer that is connected to the Internet and run *download_packages.sh* (you may need to make it executable by running 'chmod +x download_packages.sh'). Script should finish without any errors. If not, please check the script output, resolve possible issues and run it again. After it is successful run, the .packages directory should be populated with RPM packages that are required for installation.

   - Copy the .packages directory to the Propel Distributed scripts directory of the Load Balancer node and proceed with installation.

9. Offline installation on the custom RHEL computer:

   - Set *propeld.packages.enabled* to false in group_vars/propeld.yml.

     ```
     # vim group_vars/propeld.yml
     ```

   - Install the following packages (and their dependencies) to nodes with specified roles (described in step 4↑) :

     - glusterfs-server 3.8.4 - roles: *lb* and *propel*
     - nginx 1.10.2 - roles: *lb*
     - pgpool-II-pg95 3.4.8 - roles: *db_m* and *db_s*
     - ntp 4.2.6p5 - all nodes

10. Define an alternate net interface name on all Database nodes.
    By default, the database nodes will expect the ethernet interface to be named eth0. However, Redhat 7.2 now uses alternate and more secure naming conventions, such as eno16780032. Run 'ip a' and check the network interface name. If it is not eth0, it needs to be defined in group_vars/propeld.yml like this:

    ```
    postgresql:
      passwords:
        pgpool: "PASSWORD"
        postgres: "postgres"
        repl: "PASSWORD"
      interface: "eno16780032"
    ```

Running the Distributed Propel scripts

Before you follow the steps in this section, make sure you have completed the steps in the previous three sections. There are multiple steps to follow in order to setup a Distributed Propel Cluster. The process to execute these scripts is detailed below.

If you modified any of the Ansible YAML scripts, use the online parser to check for syntax errors before running them: http://yaml-online-parser.appspot.com/

NOTES:

- When prompted by the first scripts for the SUDO Password, you need to give the password of the *Propel* user on the target server.
- If any of the scripts are aborted before completion, it is safe to re-run them.

1. Run the following commands:

```
# cd /opt/hp/propel-distributed
# ansible-playbook propeld.yml -c paramiko --ask-become-pass –u propel 2>&1 | tee setup.out
```

2. If previous command successfully finishes, your Propel cluster should be installed and ready for use.

SSL setup

The Propel Cluster uses GlusterFS to synchronize Propel configuration files across whole cluster. Therefore all changes made to SSL configuration should be done on a single node (usually LB).

Replacing Generated HPE Propel SSL Certificates with CA-Signed Certificates

1. Connect to the Load Balancer node:

   ```
   # ssh propel@<lb_host>
   ```

2. Stop Propel:

   ```
   # ansible propel -a "propel stop"
   ```

3. Follow the steps in section 'Replacing Generated HPE Propel SSL Certificates with CA-Signed Certificates' in the Propel 2.20.p2 Administration Guide. Please ignore steps #1 - Stop HPE Propel and #14 - Restart HPE Propel, these steps are replaced by “stopping/starting whole cluster” from this guide.

4. Start Propel:

   ```
   # ansible propel -a "propel start"
   ```

Failover and Recovery

This section provides a list of potential failure scenarios and the steps to recover your Distributed Propel cluster environment.

Failure Scenarios	Recovery Action

Pgpool stops on standby database server	Verify that the standby is healthy:
	From the primary:
	`# sudo -u postgres psql -h <standby> -p 5432 -c 'select pg_is_in_recovery()'`
	` pg_is_in_recovery`
	`------------------`
	` t`
	`(1 row)`
	If this query returns "t" as above, then all that is required is to restart pgpool:
	From the standby:
	`# systemctl start pgpool.service`
	Confirm pgpool is running properly:
	`# systemctl status pgpool.service`

Pgpool stops on primary database server	Verify that the primary is healthy:
	From the standby:
	`# sudo -u postgres psql -h <primary> -p 5432 -c 'select pg_is_in_recovery()'`
	` pg_is_in_recovery`
	`------------------`
	` f`
	`(1 row)`
	If this query returns "f" as above, then all that is required is to restart pgpool:
	From the primary:
	`# systemctl start pgpool.service`
	Confirm pgpool is running properly:
	`# systemctl status pgpool.service`

Failure Scenarios	Recovery Action
PostgreSQL stops on standby database server	First, perform a backup of the primary database server. (See Single Node Propel Backup) Address the root cause of PostgreSQL's stoppage. Then, confirm that connectivity exists with primary: From primary: `# ssh <standby> service pgpool-II-94 status` <- should see normal pgpool status output -> From standby, restart PostgreSQL: `# systemctl status pgpool.service` From primary, confirm pgpool has attached to both primary and standby: `# sudo -u postgres psql -h <DB-VIP> -p 9999 -c "show pool_nodes"` `node_id	hostname	port	status	lb_weight	role` `---------+-----------+------+--------+-----------+---------` ` 0	<primary>	5432	2	0.500000	primary` ` 1	<standby>	5432	2	0.500000	standby`  The "role" column should contain the appropriate primary/standy value and the status column should be "2" for both nodes.  Confirm replication is active. From primary:  `# sudo -u postgres psql -h <primary> -p 5432 -c 'select sent_location, replay_location from pg_stat_replication'` `sent_location	replay_location` `---------------+-----------------` ` 7D/90004B0	7D/9000478` `(1 row)` Wait 60 seconds, and run the same command. Results should differ.

PostgreSQL stops on primary database server

Note:

- Recovery will involve a Propel service outage.
- After recovery, the primary and standby databases will swap roles. In these notes new-primary refers to the original standby, which has been promoted. Similarly, new-standby refers to the original primary, which has stopped.
- See diagram at the end of this section for illustration of how the system changes when a failover event occurs.

On the new-standby, stop pgpool and confirm PostgreSQL is stopped:

```
# systemctl stop pgpool.service
# systemctl status pgpool.service
```

Stop Propel and OO on all Propel nodes:

```
# propel stop
# systemctl stop central.service
```

Perform a backup of the new-primary database server. (See Single Node Propel Backup)

On the new-primary, restart PostgreSQL and pgpool:

```
# systemctl restart pgpool.service
# systemctl restart postgresql-9.5.service
```

Confirm that new-primary has been promoted:

```
# sudo -u postgres psql -h <new-primary> -p 5432 -c 'select
pg_is_in_recovery()'
 pg_is_in_recovery
-------------------
 f
(1 row)
```

Result should be 'f'.

```
# sudo -u postgres psql -h <DB-VIP> -p 9999 -c "show pool_nodes"
node_id |    hostname    | port | status | lb_weight |  role
---------+----------------+------+--------+-----------+---------
 0       | <new-standby> | 5432 | 3      | 0.500000  | standby
 1       | <new-primary> | 5432 | 2      | 0.500000  | primary
```

The "role" values should be reset. New-primary should have a status "2" (up), new-standby should have a status of "3" (down).

On load-balancer, create a new inventory file with primary and standby reversed. For example, if the original primary was db1.hpe.net and the original standby was db2.hpe.net, your new inventory would have this content:

```
[postgres]
db2.hpe.net  ansible_ssh_user=root
db1.hpe.net  ansible_ssh_user=root

[db_master_nodes]
db2.hpe.net  ansible_ssh_user=root

[db_slave_nodes]
db1.hpe.net  ansible_ssh_user=root
```

For this example, we assume that the new inventory file is `/opt/hp/propel-distributed/inventories/recovery_cluster`. Then, from the directory `/opt/hp/propel-distributed` on load-balancer, rerun Ansible playbook #2:

```
# ansible-playbook -i inventories/recovery_cluster
2_setup_pgpool_cluster.yml 2>&1 | tee recovery.out
```

Verify the new-primary and new-master are running. On load-balancer:

```
# sudo -u postgres psql -h <DB-VIP> -p 9999 -c "show pool_nodes"
 node_id |   hostname    | port | status | lb_weight |  role
---------+---------------+------+--------+-----------+---------
 0       | <new-primary> | 5432 | 2      | 0.500000  | primary
 1       | <new-standby> | 5432 | 2      | 0.500000  | standby
```

The “role” column should reflect the new server statuses. The “status” column should be “2” for both nodes. If status is not “2’ for both nodes, see Pgpool not attaching to nodes. Confirm replication is active:

```
# sudo -u postgres psql -h <new-primary> -p 5432 -c 'select
sent_location, replay_location from pg_stat_replication'
sent_location  | replay_location
---------------+-----------------
 7D/90004B0    | 7D/9000478
(1 row)
```

Wait 60 seconds, and run the same command. Results should differ.

On each Propel node, start Propel and start OO:

```
# propel start
# service central start
```

Verify that the mpp service has initialized properly and restart if necessary:

```
# service mpp status
```

Failure Scenarios	Recovery Action
Standby server down or unavailable	After addressing the root cause of the server outage, see failover scenario Pgpool stops on standby database server. Note: If the server shuts down abruptly, pgpool may not initialize properly. See troubleshooting note Pgpool not attaching to nodes.
Primary server down or unavailable	After addressing the root cause of the server outage, see failover scenario PostgreSQL stops on primary database server. Note: If the server shuts down abruptly, pgpool may not initialize properly. See troubleshooting note Pgpool not attaching to nodes.

Propel node down or unavailable	After addressing the root cause of the server outage, restart Propel and OO:
	
	`# propel stop`
	`# propel start`
	`# systemctl restart central.service`
	
	Verify that the Portal service has initialized properly and restart if necessary:
	
	`# systemctl status portal`

Load-balancer down or unavailable	After addressing the root cause of the server outage, restart nginx:
	
	`# service nginx restart`
	
	Note: If the backup image of the load balancer contains a Propel install, it may be necessary to stop Propel and OO:
	
	`# propel stop`
	`# systemctl restart central.service`
	
	Also, verify that no node processes are running:
	
	`# ps –ef | grep node`

The following diagram illustrates how a master- slave setup will change if a failover event occurs and then the recovery procedures are subsequently executed – this is the scenario in the table above - PostgreSQL stops on primary database server .

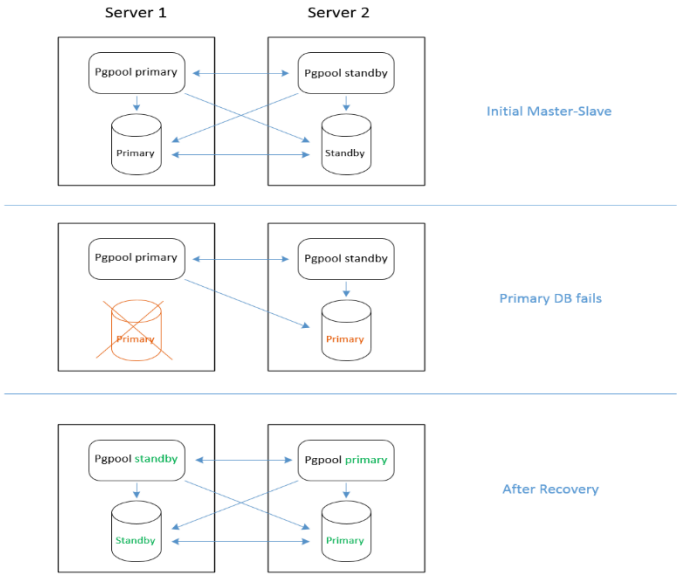


Figure 3: Master-Slave initial setup and setup after recovery

Disaster Recovery

Setting up a Propel Disaster Recovery cluster

This section assumes that a Propel Disaster Recovery (DR) Distributed Propel system has already been set up.

1. Make sure that Propel is stopped on Propel nodes on DR cluster.
2. On the DR cluster master DB node:
    1. [Optional] Back up /var/lib/pgsql/9.5/data.
    2. Delete the data directories:

        ```
        # rm -rf /var/lib/pgsql/9.5/data/*
        ```

    3. Set up replication from the Primary Cluster master DB node to the DR cluster master DB node:

        ```
        # su - postgres

        # pg_basebackup --dbname="postgresql://repl:replpass@<primary-cluster-master-db>/" -D /var/lib/pgsql/9.5/data -P -xlog-method=stream
        ```

    4. Create recovery.conf under /var/lib/pgsql/9.5/data:

        ```
        # vi /var/lib/pgsql/9.5/data/recovery.conf

        standby_mode = 'on'

        primary_conninfo = 'host=< primary-cluster-master-db> user=repl password=replpass'

        restore_command = 'cp /var/lib/postgresql/9.5/archive/%f %p'

        recovery_target_timeline='latest'

        trigger_file = '/tmp/pgsql.trigger'
        ```

        Changes the permissions:

        ```
        # chown postgres:postgres recovery.conf
        ```

    5. Restart Postgres on both DR master and slave nodes.

        ```
        # servicectl restart postgressql-9.5
        ```

3. Verification steps:
    1. On the DR cluster, make sure that both DB notes are standing by.
        1. Check the Master Postgres:

            ```
            # sudo -u postgres psql -h <dr-cluster-master- db > -p 5432 -c "SELECT pg_is_in_recovery()"


            pg_is_in_recovery -------------------

            t
            ```

        2. Check the Slave Postgres:

            ```
            # sudo -u postgres psql -h < dr-cluster-slave- db> -p 5432 -c "SELECT pg_is_in_recovery()"


            pg_is_in_recovery -------------------

            T
            ```

2. Verify that replication is happening to the DR cluster master from the Primary cluster master.
3. Replication from the DR cluster master to the DR cluster slave is stopped in this mode. Once the DR site is enabled as primary site, replication starts from master to slave.

Switching Propel to your Disaster Recovery cluster

1. Make sure that all nodes on the Primary cluster are down.
2. Rerun Ansible playbook script db.yml from DR cluster LB node from the directory /opt/hp/propel/contrib/propel-distributed.<version> on load-balancer:

   ```
   # ansible-playbook db.yml -c paramiko --ask-become-pass -u propel 2>&1 | tee recovery.out
   ```

   Verification steps:

   1. On the DR cluster, make sure that both DB notes are standing by.
      1. Check the Master Postgres:

         ```
         # sudo -u postgres psql -h <primary> - p 5432 -c "SELECT pg_is_in_recovery()"

         pg_is_in_recovery -------------------

         f
         ```

      2. Check the Master Postgres using DB- VIP:

         ```
         # sudo -u postgres psql -h <DB-VIP> - p 5432 -c "SELECT pg_is_in_recovery()"


         pg_is_in_recovery -------------------

         f
         ```

      3. Check the Slave Postgres:

         ```
         # sudo -u postgres psql -h <standby> - p 5432 -c "SELECT pg_is_in_recovery()"


         pg_is_in_recovery -------------------

         t
         ```

   2. Check if replication is happening to the DR cluster slave.
   3. Check pgpool (show pool_nodes). From primary DB of DR, make sure that pgpool has been attached to both the primary and standby.

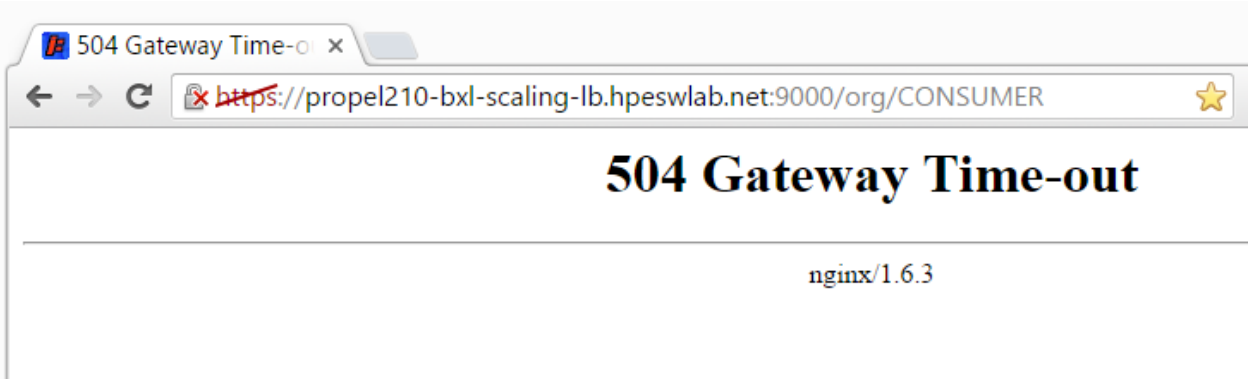
      ```
      # sudo -u postgres psql -h <DB-VIP> -p 9999 -c "show pool_nodes"

      node_id | hostname | port | status | lb_weight | role

      ------------+----------+------+-----+------------+----------

      0       | <primary> | 5432 | 2     | 0.500000 | primary

      1       | <standby> | 5432 | 2     | 0.500000 | standby
      ```

   4. The "role" column should contain the appropriate primary/standy value and the status column should be "2" for both nodes.
3. Start Propel nodes and restart Nginx on the DR cluster.
4. Login to the DR UI and check whether the data created on the Primary site is present on the DR.

Troubleshooting Hints and Tips

This section provides troubleshooting hints and tips that you may encounter during the setup of your Distributed Propel Cluster.

NGINX 504 Gateway Time-Out



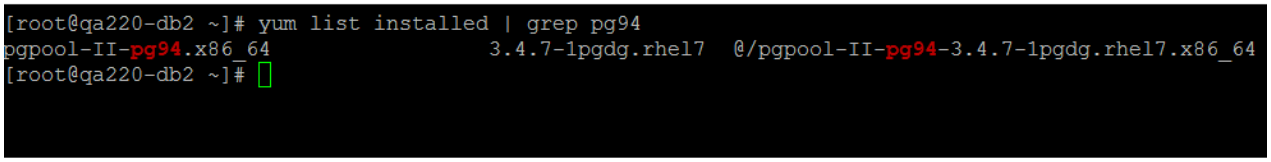
a. Check if pgpool is running and listening:

```
# systemctl status pgpool-II-94
# sudo -u postgres psql -h <DB VIP> -p 9999 –l
```

a. Check if IDM on node1/2 can connect to the DB (see logs in /var/log/propel/idm). If not, restart Launchpad and IdM. If they can connect to the DB, try the LB connection again. If that works, restart all other services on all propel nodes. Test the LB connection again.

Pgpool not starting

Make sure that version 3.4.7 is installed (example below shows 3.4.3). This is the version that HPE validated with in the Distributed Propel configuration.



You can update the `propel-distributed\roles\pgpool\tasks\main.yml` file for the pgpool role to force installing a specific version:

```
name: roles:pgpool Install older pgPool
yum: name=http://www.pgpool.net/yum/rpms/3.4/redhat/rhel-7-x86_64/pgpool-II-pg94-3.4.7-
1pgdg.rhel7.x86_64.rpm     state=installed
  ignore_errors: yes
```

Pgpool not attaching to nodes

When both databases are running, the “show pool_nodes” query should show a status of “2” for both nodes.

```
# sudo -u postgres psql -h <DB-VIP> -p 9999 -c "show pool_nodes"
 node_id | hostname  | port | status | lb_weight |  role
---------+-----------+------+--------+-----------+---------
 0       | <PRIMARY> | 5432 | 2      | 0.500000  | primary
 1       | <STANDBY> | 5432 | 3      | 0.500000  | standby
```

To obtain the expected result, try the following:

1) On the primary, restart pgpool:

```
# service pgpool-II-94 restart
```

On the standby. restart pgpool:

```
# service pgpool-II-94 restart
```

Check result:
```
# sudo -u postgres psql -h <DB-VIP> -p 9999 -c "show pool_nodes"
```

2) If the status is still incorrect, perform the following steps:
On the standby, stop pgpool:
```
# service pgpool-II-94 stop
```

On the primary, stop pgpool:
```
# service pgpool-II-94 stop
```

On the primary, confirm eth0:0 is down:
```
# ifdown eth0:0
```

On the primary, verify that pgpool exited gracefully
```
# rm –i /tmp/.s.PGSQL.9898
# rm –i /var/run/postgresql/.s.PGSQL.9999
```

On the primary, restart pgpool:
```
# service pgpool-II-94 start
```

Check result:
```
# sudo -u postgres psql -h <DB-VIP> -p 9999 -c "show pool_nodes"
```

If status is "2" for both nodes, restart pgpool on standby:
```
# service pgpool-II-94 start
```

3) If the status is still incorrect, perform the following steps:

On the standby, stop pgpool:
```
# service pgpool-II-94 stop
```

Confirm the status of the primary. Result should be "f":

```
# sudo -u postgres psql -h <Primary> -p 5432 -c 'select pg_is_in_recovery()'
 pg_is_in_recovery
-------------------
 f
(1 row)
```
Confirm the status of the standby. Result should be "t":

```
# sudo -u postgres psql -h <Standby> -p 5432 -c 'select pg_is_in_recovery()'
 pg_is_in_recovery
-------------------
 t
(1 row)
```

If these are incorrect, the issue is more likely with the configuration of PostgreSQL. Otherwise, perform these steps:

On the primary, run these commands using the *node_id* that reports a status of "3":
```
# /usr/pgpool-9.4/bin/pcp_detach_node -U pgpool -h localhost -p 9898 -W -n <node_id>
Password:
# /usr/pgpool-9.4/bin/pcp_attach_node -U pgpool -h localhost -p 9898 -W -n <node_id>
Password:
```

By default, the password is `pgpool`

Wait 60 seconds and check result:
```
# sudo -u postgres psql -h <DB-VIP> -p 9999 -c "show pool_nodes"
```

PostgreSQL queries on VIP fail

When one or both databases are up and pgpool is running, this error should not occur:

```
# sudo -u postgres psql -h <DB-VIP>-p 9999 -c 'SELECT now()'
psql: server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
```

To fix, follow the same steps in Pgpool not attaching to nodes.

"show pool_nodes" shows both databases

When one or both databases are up and pgpool is running, this error should not occur:

```
# sudo -u postgres psql -h <DB-VIP> -p 9999 -c "show pool_nodes"
 node_id | hostname  | port | status | lb_weight |  role
---------+-------------+------+--------+-----------+---------
 0       | <PRIMARY> | 5432 | 2      | 0.500000  | standby
 1       | <STANDBY> | 5432 | 2      | 0.500000  | standby
```

To fix, follow the same steps in Pgpool not attaching to nodes.

Load Balancer Node Information

nginx logs : /var/log/nginx

nginx conf : /etc/nginx/conf.d/virtual.conf

Command to restart nginx: service nginx restart

Database Node Information

The following section contains information about the DB node.

How to change postgreSQL to listen to all interfaces

http://www.postgresql.org/message-id/4B79CCFE.5040105@hogranch.com

1. Edit the `pg_hba.conf` file:
   `# su – postgres`
   `# vi /var/lib/pgsql/9.5/data/pg_hba.conf`
   host all all 0.0.0.0/0 trust
2. Edit the `postgresql.conf` file:
   `# vi /var/lib/pgsql/9.5/data/postgresql.conf`
   listen_address = '*'
3. Restart PostgreSQL:
   `# service postgresql-9.5 restart`

DB Log locations:

/var/lib/pgsql/9.5/data/pg_log

DB restart:

```
# service postgresql-9.5 restart
```

DB not responding:

If PostgreSQL runs out of space and does not respond:

http://blog.endpoint.com/2014/09/pgxlog-disk-space-problem-on-postgres.html

RabbitMQ Information

The following section provides some useful commands for RabbitMQ

Broker status

```
# rabbitmqctl status
```

SX: config for MQ

```
/opt/hp/propel/sx/WEB-INF/classes/config/infrastructure.json
```

Check if rabbitmq is running correctly

5671 is used by rabbit broker:

```
# netstat -an | grep 5671
```

25672 is used by rabbit to manage clustering:

```
# netstat -an | grep 25672
```

RabbitMQ failed to start on a node.

BOOT FAILED - Timeout contacting cluster nodes: [rabbit@awha22p4].

BACKGROUND -This cluster node was shut down while other nodes were still running.
To avoid losing data, you should start the other nodes first, then start this one. To force this node to start, first invoke
"rabbitmqctl force_boot". If you do so, any changes made on other cluster nodes after this one was shut down may be lost.

DIAGNOSTICS - attempted to contact: [rabbit@awha22p4]

If you see above type of error:

```
# rabbitmqctl force_boot
# rabbitmqctl start_app
```

Promoting Single node Propel installations to Propel Distributed

Propel Migrator can be utilized to accomplish the promotion of a single node to Propel Distributed. This section provides information on how to migrate a single node Propel environment to Propel Distributed.

Prerequisites for doing the migration are:

- Source Propel 2.10 or higher single node installation
- Target nodes which is a minimum of two application nodes, two database nodes and one load balancer node
- Ansible@1.9.4
- SSH
- nodejs 4.x / npm 2.x

It is preferred that migration is run before PropelD playbooks are run. Running migration prior to PropelD installation will simplify the process because some host names in configuration files can be changed by migrations from single node installation. Some hostnames will change to localhost where a load balancer host name is expected or another application node name is expected for proper PropelD functionality. In the case that migration is run after the PropelD installation, it is required to run PropelD scripts again, to ensure that hostnames are set correctly.

Migrator playbooks expect one ‘source_app‘ node from where configurations are taken. There is one ‘target_db‘ where new master database will be and a single ‘source_db‘ with postgres database from where databases are taken from. In this case, ‘source_db‘ is always the same as the ‘source_app‘ node.

It will have multiple ‘target_app‘ nodes that need to be stopped. Later after migrations are run, nodes will be restarted.

The target load balancer node is used to synchronize Propel configuration files to target application nodes. The migrator expects one ‘target_lb‘ where configurations are migrated into.

Also there are variables for ‘old_host‘ and ‘new_host‘. This will be the host name of a source single node Propel installation and target load balancer node of an PropelD cluster. Postgres ‘old_vip‘, ‘new_vip‘ and ports ‘old_vip_port‘, ‘new_vip_port‘ will be omitted as there is no VIP replacement needed.

Example migrator inventory is as follows:

```
[source_app]

$SOURCE_APP ansible_ssh_user=$MIGRATION_USER
ansible_ssh_private_key_file=./ssh/migration_rsa


[target_app]

$TARGET_APP1 ansible_ssh_user=$MIGRATION_USER
ansible_ssh_private_key_file=./ssh/migration_rsa

$TARGET_APP2 ansible_ssh_user=$MIGRATION_USER
ansible_ssh_private_key_file=./ssh/migration_rsa


[source_db]

$SOURCE_DB ansible_ssh_user=$MIGRATION_USER
ansible_ssh_private_key_file=./ssh/migration_rsa


[target_db]

$TARGET_DB ansible_ssh_user=$MIGRATION_USER
ansible_ssh_private_key_file=./ssh/migration_rsa
```

```
[target_lb:vars]
;old host is load balancer host name in PropelD
old_host=$SOURCE_APP
;new host is load balancer host name in PropelD
new_host=$TARGET_LOAD_BALANCER
To run migrator install Rollover:
# npm install
To run migration
# ./migrate.sh
```

Installation of reverse proxy on single node instance

Propel distributed scripts provide support for automated installation of an NGINX reverse proxy on a single node Propel instance. A reverse proxy hides the ports on which services are running and provides more user-friendly URLs. Context paths are used instead of ports. For example, *https://propel.com:9510/catalogs* will became *https://propel.com/catalog-ui/catalogs*. This section provides information on how to configure and execute Propel distributed scripts for such tasks.

Note: Old URLs, for example, *https://propel.com:9000/dashboard*, will become unavailable after the reverse proxy installation.

Prerequisites

Propel single node instance - this instance will be enhanced with an NGINX reverse proxy.

Setup

1. Connect to your Propel single node instance as root via SSH:

   ```
   # ssh root@your-propel-instance-fqdn
   ```

2. Go to the propel distributed scripts directory:

   ```
   # cd /opt/hp/propel/contrib/propel-distributed.<version>/
   ```

3. Copy the default configuration files:

   ```
   # cp inventory/hosts.example inventory/hosts.default
   # cp group_vars/propeld.yml.example group_vars/propeld.yml
   ```

4. In *inventory/hosts.default*, specify your propel node for the *lb* and *propel* groups:

   ```
   [lb]
   your-propel-instance-fqdn
   [lb_ip]

   [propel]
   your-propel-instance-fqdn

   [db_m]
   [db_s]
   [db_vip]
   [db:children]
   db_m
   db_s
   [propeld:children]
   lb
   propel
   db
   [internet:children]
   lb
   ```

5. In *group_vars/propeld.yml,* in thesection proxy_env, specify your http and https proxy if necessary. In this file

you can also specify port on which will be reverse proxy listening in section reverse_proxy. By default, port 443 is used.

```
proxy_env:
        http_proxy: https://your-corporate-proxy:8080
        https_proxy: https://your-corporate-proxy:8080
```

6. Copy the SSH key:

   ```
   # ssh-copy-id localhost
   ```

7. Make sure that ansible can connect to the propeld group. You may be asked if the fingerprint is correct. Type 'yes' and press enter. Command should then finish successfully.

   ```
   # ansible propeld -m ping
   ```

8. Run the ansible playbook:

   ```
   # ansible-playbook proxy.yml 2>&1 | tee proxy-setup.log
   ```

9. If the previous command finishes successfully, your Propel instance with reverse proxy is ready for use. Use *https:\\your-propel-instance-fqdn* or *https:\\your-propel-instance-fqdn:<port>* (if you have specified a port different from 443) to access it.

Note: In most cases, it is safe to re-run this playbook.