# AIC Valet Service Deployment Guide

Contents

[AIC Valet Service Deployment Guide 1](#_Toc457400651)

[1. Valet High Availability Design 3](#_Toc457400652)

[1.1. Valet High Availability Architecture 4](#_Toc457400653)

[2. Configuring HA Valet 5](#_Toc457400654)

[2.1. Prerequisites 5](#_Toc457400655)

[2.2. Configuring Valet on Host valet1 5](#_Toc457400656)

[2.2.1. HaValet 5](#_Toc457400657)

[2.2.2. Valet API 5](#_Toc457400658)

[2.2.3. Valet Engine 6](#_Toc457400659)

[2.2.4. Music 6](#_Toc457400660)

[2.2.5. Zookeeper 6](#_Toc457400661)

[2.2.6. Cassandra 6](#_Toc457400662)

[2.2.7. Supervisor (HaValet watchdog) 6](#_Toc457400663)

[2.2.8. Events Listener 6](#_Toc457400664)

[2.3. Configuring Valet on Host valet2 8](#_Toc457400665)

[2.3.1. HaValet 8](#_Toc457400666)

[2.3.2. Valet API 8](#_Toc457400667)

[2.3.3. Valet Engine 8](#_Toc457400668)

[2.3.4. Music 8](#_Toc457400669)

[2.3.5. Zookeeper 8](#_Toc457400670)

[2.3.6. Cassandra 8](#_Toc457400671)

[2.3.7. Supervisor (HaValet watchdog) 9](#_Toc457400672)

[2.3.8. Events Listener 9](#_Toc457400673)

[2.4. Configuring Valet on Host valet3 10](#_Toc457400674)

[2.4.1. Supervisor (HaValet watchdog) 10](#_Toc457400675)

[2.4.2. HaValet 10](#_Toc457400676)

[2.4.3. Valet API 10](#_Toc457400677)

[2.4.4. Valet Engine 10](#_Toc457400678)

[2.4.5. Music 10](#_Toc457400679)

[2.4.6. Zookeeper 10](#_Toc457400680)

[2.4.7. Cassandra 10](#_Toc457400681)

[2.4.8. Events Listener 11](#_Toc457400682)

[3. Installing valet\_os - adding Valet filter to OpenStack cluster - Nova, Heat. 12](#_Toc457400683)

[3.1. Nova 12](#_Toc457400684)

[3.2. Heat 13](#_Toc457400685)

[4. Appendix 13](#_Toc457400686)

[4.1. HAProxy configuration bigsite sample 13](#_Toc457400687)

[4.2. Silent SSH 14](#_Toc457400688)

[4.3. Health Checks (Loggers, Testing URLs) 15](#_Toc457400689)

[4.3.1. Zookeeper 15](#_Toc457400690)

[4.3.2. Cassandra 15](#_Toc457400691)

[4.3.3. Music 15](#_Toc457400692)

[4.3.4. Valet API 15](#_Toc457400693)

[4.3.5. Valet Engine 15](#_Toc457400694)

[4.3.6. HaValet 15](#_Toc457400695)

[4.3.7. Events Listener 15](#_Toc457400696)

# Valet High Availability Design

This document describes the distribution of the Valet components on 3 dedicated hosts and the different configuration of the various components on these hosts.

* Valet hosts should be instantiated from image found in : <http://valet.research.att.com/> (doc/aic)

Launch new instances based on this qcow file (choose flavor which can occupy ‘Ubuntu 14.04 for KVM VMs with ATT Supported Tools – SCO’ image)

Associate required Security Groups, Key Pair and Networking.

Valet-openstack – Nova and Heat Valet plugins archive for AIC Valet 1.0: <http://valet.research.att.com/> (doc/aic)

* Valet hosts have 3 execution setups:
* Valet1 runs all Valet components (Valet Engine in primary mode).
* Valet2 runs all Valet components (Valet Engine in secondary mode, no events listener).
* Valet3 runs only Music components (Zookeeper, Cassandra and Tomcat).
* The different Valet flavors – configuration-wise – are described [below](#_Configuring_Clustered_Valet)
* HAProxy receives Valet REST API calls on port 8090 from the Valet Heat Stack Lifecycle plugin and Nova Filter Scheduler plugin (ValetFilter). HAProxy balances Valet API calls between valet1 and valet2 (valet1:8090, valet2:8090).

It is assumed HAProxy already installed in AIC env.

* HAValet is running only on valet1 and Vale2 hosts.
* HAValet monitors the following:
* valet-api (local wsgi/pecan)
* Valet Engine – local and remote
* HAValet does not monitor Music components. Apache Tomcat, Zookeeper, and Cassandra are monitored via a cron job: /opt/valet/musicHealthCheck.sh.

Valet Engine HA: HAValet gives precedence to Valet Engine instance on the primary host.

HAValet queries Valet Engine for its priority. When two Engine instances are found, HAValet shuts down the instance with the lower priority.

When the Engine instance with the higher priority fails to start, HAValet starts the second instance.

* HAValet and the Event Listener (ostro-listener) are watched by supervisor (Linux watch-dog).

## Valet High Availability Architecture

HTTP:8090

HAProxy

valet2 valet1



Tomcat



HAValet

Valet API

Valet Engine

ZK



Cassandra



Supervisor

Supervisor completely

Valet API

HAValet

Valet Engine

s

Secondary

Primary

Events Listener

Service

Music

Tomcat

ZK

Cassandra

valet3



Tomcat

ZK

Cassandra

# Configuring HA Valet

## Prerequisites

* Define hosts ‘valet1’, ‘valet2’, ‘valet3’ in /etc/hosts file on 3 Valet hosts.
* Verify in the file /etc/ssh/sshd\_config *PasswordAuthentication* parameter set to yes:

PasswordAuthentication yes

* Verify Silent SSH works between the following hosts (i.e. $ ssh <host> does not require the password):
  1. valet1 🡪 valet1
  2. valet2 🡪 valet2
  3. valet1 🡨🡪 valet2

For the detailed silent SSH procedures please see [Appendix.](#_Silent_SSH)

## Configuring Valet on Host valet1

## HaValet

In /opt/valet/allegro/havalet/ha\_valet.cfg – set following properties in the ‘Ostro’ section:

stand\_by\_list=valet1, valet2

host=valet1

priority=1

user=<username of valet1 host *e.g. attcloud>*

## Valet API

In /opt/valet/allegro/valet\_api/config.py set *identity* parameters according to your cloud configuration:

identity = {

'config': {

'username': '<openstack username>', *for example, admin*

'password': '<password of the above user>',

'project\_name': '<openstack project name>',

'auth\_url': 'http://<keystone host URL>:5000/v2.0', *for example, localhost*

}

}

Set in /opt/valet/allegro/valet\_api/config.py *messaging* parameters according your RabbitMQ cloud configuration.

messaging = {

'config': {

'transport\_url': 'rabbit://<username>:<password>@<Rabbit MQ host>:5672/',

}

}

Take the relevant values of <username>, <password>, and <Rabbit MQ host> from /etc/nova/nova.conf (‘oslo\_messaging\_rabbit’ section) on Controller node.

## Valet Engine

Verify following parameters in /opt/valet/ostro/code/ostro\_server/ostro.cfg:

priority=1

control\_loc=<Controller IP>

ip=<valet1 IP>

db\_hosts = valet1, valet2, valet3

Set in /opt/valet/ostro/code/ostro\_server/ostro.auth authentication parameters according to your cloud configuration:

project\_name=<PROJECT\_NAME> for example, demo

admin\_tenant\_name=<must be identical to the project\_name above>

user\_name=<admin name of the project above> for example, admin

password=<password of the admin user above>

## Music

Verify /opt/valet/musicHealthCheck.sh has the execution permissions.

This script is called by cron daemon; verify ~/musicCronOut is updated every minute.

## Zookeeper

As is.

## Cassandra

On /opt/valet/apache-cassandra-2.1.1/conf/cassandra.yaml

Set broadcast\_address to the host name.

Verify listen\_address: is blank.

Kill Cassandra process and delete its data:

rm -rf /opt/valet/apache-cassandra-2.1.1/data/\*

## Supervisor (HaValet watchdog)

As is.

## Events Listener

In /etc/ostro-listener/ostro-listener.conf change the following values according to your cloud configuration:

host = < compute node on which RabbitMQ is running>

username = < RabbitMQ user>

port = < RabbitMQ connection port>

passwdfile = /etc/ostro-listener/passwd

In /etc/ostro-listener/passwd add line in the form of (this file must be owned and viewed by user root only):

<RabbitMQ host> <RabbitMQ password>

Refer to section “Ostro-listener” on <http://valet.research.att.com/> for the further reference.

Start ostro-listener service:

sudo service ostro-listener restart

## Configuring Valet on Host valet2

## HaValet

In /opt/valet/allegro/havalet/ha\_valet.cfg – set the following properties in ‘Ostro’ section:

stand\_by\_list=valet1, valet2

host=valet2

priority=2

user=<username of valet1 host *e.g. attcloud>*

## Valet API

Set in /opt/valet/allegro/valet\_api/config.py – in ‘music’ section

host: valet2

Set in /opt/valet/allegro/valet\_api/config.py identity parameters according your cloud configuration – as in valet1.

Set in /opt/valet/allegro/valet\_api/config.py messaging parameters according RabbitMQ your cloud configuration - as in valet1.

## Valet Engine

Set following parameters in /opt/valet/ostro/code/ostro\_server/ostro.cfg

Priority=2

control\_loc=<Controller IP>

ip=<valet2 IP>

replication\_factor=3

db\_hosts = valet1, valet2, valet3

Set in /opt/valet/ostro/code/ostro\_server/ostro.auth authentication parameters according your cloud configuration - as in valet1.

## Music

Verify execution permissions on /opt/valet/musicHealthCheck.sh

This script is called by cron daemon; verify ~/musicCronOut is updated every minute.

## Zookeeper

Set /var/zookeeper/myid – to contain – 2

## Cassandra

On /opt/valet/apache-cassandra-2.1.1/conf/cassandra.yaml

Set broadcast\_address to the host name.

Verify listen\_address: is blank.

Kill Cassandra process and delete its data:

rm -rf /opt/valet/apache-cassandra-2.1.1/data/\*

## Supervisor (HaValet watchdog)

Restarting Valet processes.

Stop supervisor:

sudo service supervisor stop

Kill ostro\_daemon and wsgi.py process

pgrep –f wsgi.py | sudo xargs kill -9

cd /opt/valet/ostro/code/ostro\_server ; sudo python ./ostro\_daemon.py stop

Start supervisor:

sudo service supervisor start

## Events Listener

Disable ostro-listener service

sudo service ostro-listener stop

sudo update-rc.d ostro-listener disable

## Configuring Valet on Host valet3

## Supervisor (HaValet watchdog)

Not running on valet3. Stop supervisor service.

sudo service supervisor stop

sudo update-rc.d supervisor disable

sudo mv /etc/supervisor/conf.d/HAValet.conf /etc/supervisor/conf.d/ HAValet.down

## HaValet

Not running on valet3. Stop the process.

pgrep –f havalet | sudo xargs kill -9

## Valet API

Not running on valet3. Stop the process.

pgrep –f wsgi.py | sudo xargs kill -9

## Valet Engine

Not running on valet3. Stop the process.

cd /opt/valet/ostro/code/ostro\_server ; sudo python ./ostro\_daemon.py stop

## Music

Verify execution permissions on /opt/valet/musicHealthCheck.sh

This script is called by cron daemon; verify ~/musicCronOut is updated every minute.

## Zookeeper

Set /var/zookeeper/myid – to contain - 3

## Cassandra

On /opt/valet/apache-cassandra-2.1.1/conf/cassandra.yaml

Set broadcast\_address to the host name.

Verify listen\_address: is blank.

Kill Cassandra process and delete its data:

rm -rf /opt/valet/apache-cassandra-2.1.1/data/\*

## Events Listener

Disable ostro-listener service

sudo service ostro-listener stop

sudo update-rc.d ostro-listener disable

# Installing valet\_os - adding Valet filter to OpenStack cluster - Nova, Heat.

Download the archive [VALET\_OPENSTACK\_1.0.tar.gz](https://tspace.web.att.com/files/app#/folder/54c30069-3570-4f74-8c6e-96f90ec8d06e) and unpack it in temporary location and change to the archive’s root directory:

$ tar -xzf VALET\_OPENSTACK\_1.0.tar.gz

Install the Valet Filter (need to be done on the each Nova and Heat hosts):

* Open the valet\_os folder:

$ cd ./VALET\_OPENSTACK\_1.0/valet\_os

* Add the line version = 1.0 to the file setup.cfg:

[metadata]

version = 1.0

* Perform the commands:

$ sudo git init

$ sudo pip install pbr==0.10.0

$ sudo pip install -e .

$ sudo python setup.py install

Refer to <http://valet.research.att.com/> (“valet-openstack”) for further details (also available as README in valet\_os directory)

## Nova

Add the following lines to /etc/nova/nova.conf:

Under the [DEFAULT] section (don’t drop already exists filters):

scheduler\_available\_filters = nova.scheduler.filters.all\_filters

scheduler\_available\_filters = valet\_os.nova.valet\_filter.ValetFilter

scheduler\_default\_filters = RetryFilter, AvailabilityZoneFilter, RamFilter, ComputeFilter, ComputeCapabilitiesFilter, ImagePropertiesFilter, ServerGroupAntiAffinityFilter, ServerGroupAffinityFilter, ValetFilter

Add [Valet] section:

[valet]

url = http://< IP of Valet node or its Load balancer>:8090/v1

failure\_mode = reject

admin\_tenant\_name = <set tenant name> for example, demo

admin\_username = <set tenant admin name> for example, admin

admin\_password = <password of the admin user above>

admin\_auth\_url = http://<keystone\_server\_ip>:35357/v2.0

Restart nova-scheduler using separate stop and start directives:

$ sudo service nova-scheduler stop

$ sudo service nova-scheduler start

## Heat

Perform the following changes /etc/heat/heat.conf:

Set the plugin\_dirs option in the [DEFAULT] section so that Heat can locate and use the Valet Stack Lifecycle Plugin. If plugin\_dirs is already present, separate entries by commas. The order of entries does not matter.

[DEFAULT]

plugin\_dirs = /opt/valet/allegro/valet\_os/etc/valet\_os/heat

stack\_scheduler\_hints = True

Add a [valet] section. This will be used by the Valet Stack Lifecycle Plugin:

**Note**: *valet\_api\_proxy* requires load balancing solution (e.g. haproxy)

[valet]

url = http://valet\_api\_proxy:8090/v1

Restart heat-engine using separate stop and start directives:

$ sudo service heat-engine stop

$ sudo service heat-engine start

# Appendix

Appendix items are illustrations only and AIC integration can configure below items as needed to support the AIC environment.

The below configurations were used in the BIG-SITE system supported by AT&T Labs Research.

## HAProxy configuration bigsite sample

HAProxy service installation (in case HAProxy is not already installed):

apt-get update

apt-get upgrade

apt-get install haproxy

set ENABLED=1 in /etc/default/haproxy

Starting the service (see conf sample below):

service haproxy start

HAProxy configuration bigsite sample:

maxconn 2000

errorfile 400 /etc/haproxy/errors/400.http

errorfile 403 /etc/haproxy/errors/403.http

errorfile 408 /etc/haproxy/errors/408.http

errorfile 500 /etc/haproxy/errors/500.http

errorfile 502 /etc/haproxy/errors/502.http

errorfile 503 /etc/haproxy/errors/503.http

errorfile 504 /etc/haproxy/errors/504.http

listen valet\_api 0.0.0.0:8090

mode http

stats enable

stats uri /haproxy?stats

balance roundrobin

no option http-server-close

http-check expect status 200

option forwardfor

server valet1 <valet1 IP>:8090 check

server valet2 <valet2 IP>:8090 check

## Silent SSH

In case needed this procedure explains how to establish silent ssh between two linux hosts.

Generate rsa key-pair for node user:

$ ssh-keygen

Append the generated keys on the **destination host**:

$ cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

Repeat this procedure for the root user:

$ sudo ssh-keygen

$ sudo cat /root/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

As some of the valet scripts are executed via ‘sudo’, silent ssh is required for both valet and root user. There for perform this procedure both as root and attcloud user.

Run ssh between hosts and hit ‘enter’ when asked to accept the destination host fingerprint

(sets the remote host fingerprint on local known\_hosts file):

attcloud@valet1:$ ssh attcloud@valet1

attcloud@valet1:$ sudo ssh attcloud@valet1

attcloud@valet1:$ ssh attcloud@valet2

attcloud@valet1:$ sudo ssh attcloud@valet2

attcloud@valet21:$ ssh attcloud@valet1

attcloud@valet2:$ sudo ssh attcloud@valet1

attcloud@valet21:$ ssh attcloud@valet2

attcloud@valet2:$ sudo ssh attcloud@valet2

## Health Checks (Loggers, Testing URLs)

Health checks should be carried out in the following order

### Zookeeper

Logger: /opt/valet/zookeeper.out

Verify process status: sudo /opt/valet/zookeeper-3.4.6/bin/zkServer.sh status

### Cassandra

Verify process status: sudo /opt/valet/apache-cassandra-2.1.1/bin/cqlsh -e "describe keyspace valet\_test"

### Music

Logger: /var/log/tomcat7/catalina.out

Service status: sudo service tomcat7 status

URL: http://:<valet host>:8080/MUSIC/rest/test

### Valet API

Logger: /var/log/havalet/Alegro\_#.log

URL: http//:<valet host>:8090/v1

### Valet Engine

Logger: /opt/valet/ostro/code/ostro\_server/ostro\_log/test.log

HA Logger: /var/log/havalet/Ostro\_#.log

### HaValet

Verifying HA process is alive: ps –ef | grep havalet

Loggers: /var/log/listener.out.log

/var/log/listener.err.log

/var/log/havalet

### Events Listener

Logger: /var/log/listener.out.log

/var/log/listener.err.log