**Cloudlink部署指南**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本号： | V1.0.0 | 文档编号： |  |
| 文档密级： | 保密 | 归属部门/项目： | 系统工程部 |
| 产品名： | Cloudlink部署指南 | 子系统名： |  |
| 编写人： | 刘晔 | 编写日期： | 2015-11-25 |

万邑通信息科技有限公司

内部资料 注意保密

**修订记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订人** | **修订日期** | **修订内容** |
| V1.0.0 | 刘晔 | 2015-11-25 | 创建 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. 部署架构 4](#_Toc436229827)

[2. 硬软件需求 5](#_Toc436229828)

[3. RabbitMQ安装 5](#_Toc436229829)

[3.1. 安装erlang语言环境 5](#_Toc436229830)

[3.1.1. 安装erlang 5](#_Toc436229831)

[3.1.2. 验证erlang 5](#_Toc436229832)

[3.2. 安装rabbitmq 6](#_Toc436229833)

[3.2.1. 下载rabbitmq 6](#_Toc436229834)

[3.2.2. 安装rabbitmq 6](#_Toc436229835)

[3.2.3. 配置rabbitmq自启动 6](#_Toc436229836)

[3.2.4. 验证rabbitmq 6](#_Toc436229837)

[3.3. 配置rabbitmq 6](#_Toc436229838)

[3.3.1. 环境配置 6](#_Toc436229839)

[3.3.2. 启动/停止/查看状态rabbitmq 6](#_Toc436229840)

[3.3.3. 启用web管理界面 7](#_Toc436229841)

[3.3.4. 安装shovel插件 7](#_Toc436229842)

[3.3.5. 创建用户 7](#_Toc436229843)

[3.3.6. 修改用户密码 7](#_Toc436229844)

[3.3.7. 分配权限 7](#_Toc436229845)

[3.3.8. 授权登录 7](#_Toc436229846)

[3.4. RabbitMQ集群配置 7](#_Toc436229847)

[3.4.1. 编辑Hostname 7](#_Toc436229848)

[3.4.2. 同步erlang cookie 8](#_Toc436229849)

[3.4.3. 启动各节点上的rabbitmq 8](#_Toc436229850)

[3.4.4. 组成集群 9](#_Toc436229851)

[3.4.5. 设置镜像队列策略 9](#_Toc436229852)

[3.4.6. 搭建负载均衡 9](#_Toc436229853)

[3.4.7. 注意事项 9](#_Toc436229854)

[4. 云链配置 10](#_Toc436229855)

[4.1. Exchange配置 10](#_Toc436229856)

[4.1.1. 新建exchange 10](#_Toc436229857)

[4.1.2. 配置exchange绑定关系 10](#_Toc436229858)

[4.1.3. 建立shovel关系 10](#_Toc436229859)

[5. 监控 13](#_Toc436229860)

# 部署架构



# 硬软件需求

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IDC | 设备名称 | 配置简要 | 安装软件(网络服务) | 域名 |
| 广州IDC | Rabbitmq服务器2台 | CPU: Xeon E5 四核\* 1  内存: 8G RAM,200G硬盘 | Centos 6.x ext4，  RabbitMQ 3.5.x |  |
| US-AWS | Rabbitmq服务器2台 | CPU: Xeon E5 二核\* 1  内存: 8G RAM | Centos 6.x ext4，  RabbitMQ 3.5.x |  |
| AU-AWS | Rabbitmq服务器2台 | CPU: Xeon E5 二核\* 1  内存: 8G RAM | Centos 6.x ext4，  RabbitMQ 3.5.x |  |
| UK-AWS | Rabbitmq服务器2台 | CPU: Xeon E5 二核\* 1  内存: 8G RAM | Centos 6.x ext4，  RabbitMQ 3.5.x |  |

# RabbitMQ安装

## 安装erlang语言环境

### 安装erlang

centos7.x：

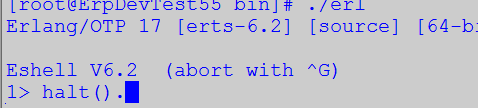
rpm -Uvh <http://download.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm>

centos6.x

rpm -Uvh <http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm>

### 验证erlang

执行erl可以看到erlang的详细信息，执行halt().命令退出；



## 安装rabbitmq

### 下载rabbitmq

wget http://www.rabbitmq.com/releases/rabbitmq-server/v3.5.6/rabbitmq-server-3.5.6-1.noarch.rpm

### 安装rabbitmq

rpm --import https://www.rabbitmq.com/rabbitmq-signing-key-public.asc

yum install rabbitmq-server-3.5.6-1.noarch.rpm

### 配置rabbitmq自启动

chkconfig rabbitmq-server on

### 验证rabbitmq

rabbitmq-server -detached 后台启动rabbitmq

telnet 5672端口或打开rabbitmq控制台页面验证rabbitmq是否启动成功

<http://xxxx.xxx.xxx.xxxx:15672/>，默认用户名和密码为guest

## 配置rabbitmq

### 环境配置

参考文档<https://www.rabbitmq.com/configure.html>

环境变量初始值通过文件rabbitmq-env.conf来配置，rabbitmq-env.conf默认在/etc/rabbitmq，

且位置不可更改。rabbitmq-env.conf中的每项都以RABBITMQ\_为前缀，常用参数如下：

RABBITMQ\_NODE\_IP\_ADDRESS= //IP地址，默认bind所有地址，指定地址bind指定网络接口

RABBITMQ\_NODE\_PORT= //TCP端口号，默认是5672

RABBITMQ\_NODENAME= //节点名称。默认是rabbit

RABBITMQ\_MNESIA\_BASE= //mnesia所在路径，通常指向数据盘路径

RABBITMQ\_LOG\_BASE= //日志所在路径，默认为/var/log/rabbitmq

### 启动/停止/查看状态rabbitmq

* service rabbitmq-server stop/start/status
* rabbitmqctl stop //停止

rabbitmq-server -detached //后台启动

### 启用web管理界面

rabbitmq-plugins enable rabbitmq\_management

### 安装shovel插件

rabbitmq-plugins enable rabbitmq\_shovel

rabbitmq-plugins enable rabbitmq\_shovel\_management

### 创建用户

rabbitmqctl add\_user admin password //admin为用户名，password为密码，根据需要修改

### 修改用户密码

rabbitmqctl change\_password admin newpassword //admin为用户名，newpassword为新密码，根据需要修改

### 分配权限

rabbitmqctl set\_permissions -p "/" admin ".\*" ".\*" ".\*" //admin为用户名，根据需要修改

### 授权登录

rabbitmqctl **set\_user\_tags admin administrator //admin为用户名，根据需要修改**

## RabbitMQ集群配置

集群参考文档<https://www.rabbitmq.com/clustering.html>

### 编辑Hostname

示例环境：二台主机，主机名和IP如下

主机名 IP

*rabbit1* 192.168.130.102

*rabbit2* 192.168.130.103

rabbit1$ ping rabbit1

rabbit1$ ping rabbit2

rabbit2$ ping rabbit1

rabbit2$ ping rabbit2

* 修改hostname

hostname *rabbit1*

hostname *rabbit2*

* 编辑/etc/hosts

192.168.130.102 *rabbit1*

192.168.130.103 *rabbit2*

### 同步erlang cookie

Rabbitmq基于erlang语言，erlang语言分布式基于cookie进行集群，rabbitmq集群首先要确保每台rabbitmq服务器的.erlang.cookie文件相同。Erlang Cookie 文件：/var/lib/rabbitmq/.erlang.cookie。这里将 rabbitmq1 的该文件复制到rabbitmq2，由于这个文件权限是400，所以需要先修改 rabbitmq1中的该文件权限为 777：

# chmod 777 /var/lib/rabbitmq/.erlang.cookie

然后将 *rabbit1*中的该文件拷贝到 *rabbit2*，最后将权限和所属用户/组修改回来：

#scp /var/lib/rabbitmq/.erlang.cookie root@rabbit2:/var/lib/rabbitmq/.erlang.cookie

# chmod 400 /var/lib/rabbitmq/.erlang.cookie

# chown rabbitmq /var/lib/rabbitmq/.erlang.cookie

# chgrp rabbitmq /var/lib/rabbitmq/.erlang.cookie

### 启动各节点上的rabbitmq

rabbit1$ rabbitmqctl stop

rabbit1$ rabbitmq-server -detached

rabbit2$ rabbitmqctl stop

rabbit2$ rabbitmq-server –detached

 可以在各个主机上执行以下命令查看集群状态,  此时看到的结果应该是各个RabbitMQ实例都是独自运行的.

rabbit1$ rabbitmqctl cluster\_status

Cluster status of node rabbit@rabbit1 ...

[{nodes,[{disc,[rabbit@rabbit1]}]},{running\_nodes,[rabbit@rabbit1]}]

...done.

### 组成集群

需要将rabbit2主机上的rabbitmq加入rabbit1主机上运行的RabbitMQ集群，通过在rabbit1主机上执行"rabbitmqctl cluster\_status"后我们可以得到rabbit1上运行的集群的名字：

{cluster\_name,<<"rabbit@rabbit1">>}

将 *rabbit2*与 *rabbit1*组成集群, 在*rabbit2*主机上执行以下shell命令：

rabbit2$ *rabbitmqctl stop\_app*

Stopping node rabbit@rabbit2 ...done.

rabbit2$ *rabbitmqctl join\_cluster rabbit@rabbit1*

Clustering node rabbit@rabbit2 with [rabbit@rabbit1] ...done.

rabbit2$ *rabbitmqctl start\_app*

Starting node rabbit@rabbit2 ...done.

之后，在rabbit1和rabbit2主机上执行以下命令来查看是否集群配置成功。

rabbit1$ *rabbitmqctl cluster\_status*

Cluster status of node rabbit@rabbit1 ...

[{nodes,[{disc,[rabbit@rabbit1,rabbit@rabbit2]}]},

{running\_nodes,[rabbit@rabbit2,rabbit@rabbit1]}]

...done.

rabbit2$ *rabbitmqctl cluster\_status*

Cluster status of node rabbit@rabbit2 ...

[{nodes,[{disc,[rabbit@rabbit1,rabbit@rabbit2]}]},

{running\_nodes,[rabbit@rabbit1,rabbit@rabbit2]}]

...done.

### 设置镜像队列策略

在任意一个节点上执行

# rabbitmqctl set\_policy ha-all "^" '{"ha-mode":"all"}' 将所有队列设置为镜像队列，即队列会被复制到各个节点，各个节点状态保持一直。

### 搭建负载均衡

广州IDC安装并配置HAProxy，AWS配置EBL

### 注意事项

cookie在所有节点上必须完全一样，同步时一定要注意。

erlang是通过主机名来连接服务，必须保证各个主机名之间可以ping通。可以通过编辑/etc/hosts来手工添加主机名和IP对应关系。如果主机名ping不通，rabbitmq服务启动会失败。

# 云链配置

## Exchange配置

### 新建exchange

每个数据中心的rabbitmq都新建好winit\_send,winit\_receive二个exchange,其中：

winit\_send用于发送消息，Type指定为topic，durable指定为true

winit\_receive用于接收消息，Type指定为direct，durable指定为true

### 配置exchange绑定关系

* **中国rabbitmq**

打winit\_send这个exchange,使用GZIDC.#这个bind routing key建立到winit\_receive的绑定关系

* **美国rabbitmq**

打winit\_send这个exchange,使用USIDC.#这个bind routing key建立到winit\_receive的绑定关系

* **澳洲rabbitmq**

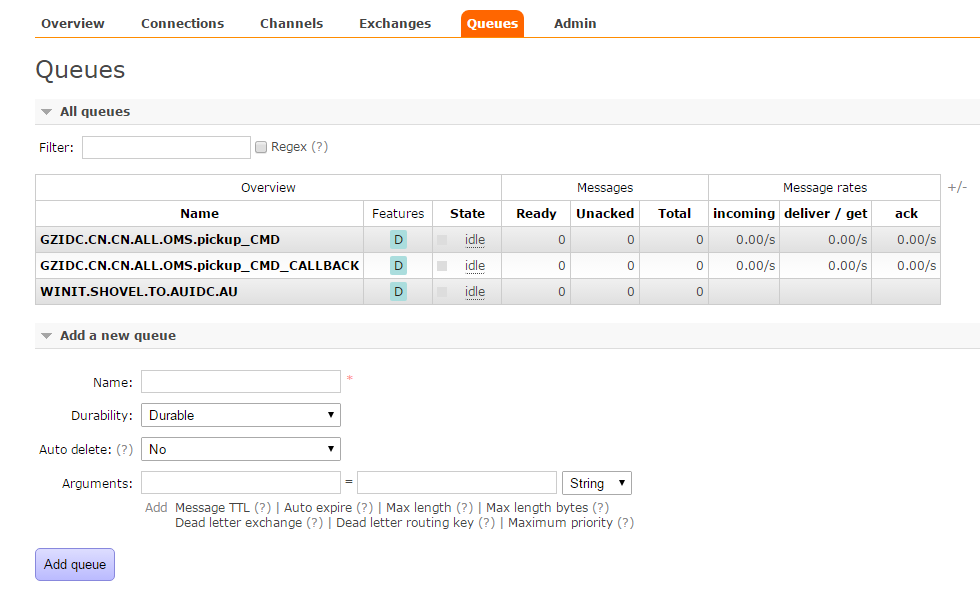
打winit\_send这个exchange,使用AUIDC.#这个bind routing key建立到winit\_receive的绑定关系

* **欧洲rabbitmq**

打winit\_send这个exchange,使用UKIDC.#这个bind routing key建立到winit\_receive的绑定关系

### 新建queue以及绑定关系

进入相关数据中心的rabbitmq web控制台，点击Queues，按下面页面分别对数据中心的rabbitmq进行queue配置：



* 本章节所有队列的创建，除名称外，其他参数都统一使用如下配置：

Durability：Durable

Auto Delete：No

Arguments：无

* 中国rabbitmq
* **Name：WINIT.SHOVEL.TO.AUIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用AUIDC. #这个bind routing key建立到winit\_send的绑定关系。

* **Name：WINIT.SHOVEL.TO.USIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用USIDC. #这个bind routing key建立到winit\_send的绑定关系。

* **Name：WINIT.SHOVEL.TO.UKIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用UKIDC. #这个bind routing key建立到winit\_send的绑定关系。

* 美国rabbitmq
* **Name：WINIT.SHOVEL.TO.AUIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用AUIDC.#这个bind routing key建立到winit\_send的绑定关系。

* **Name：WINIT.SHOVEL.TO.GZIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用GZIDC.#这个bind routing key建立到winit\_send的绑定关系。

* **Name：WINIT.SHOVEL.TO.UKIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用UKIDC.#这个bind routing key建立到winit\_send的绑定关系。

* 澳洲rabbitmq
* **Name：WINIT.SHOVEL.TO.USIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用USIDC.#这个bind routing key建立到winit\_send的绑定关系。

* **Name：WINIT.SHOVEL.TO.GZIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用GZIDC.#这个bind routing key建立到winit\_send的绑定关系。

* **Name：WINIT.SHOVEL.TO.UKIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用UKIDC.#这个bind routing key建立到winit\_send的绑定关系。

* 英国rabbitmq
* **Name：WINIT.SHOVEL.TO.AUIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用AUIDC.#这个bind routing key建立到winit\_send的绑定关系。

* **Name：WINIT.SHOVEL.TO.GZIDC**

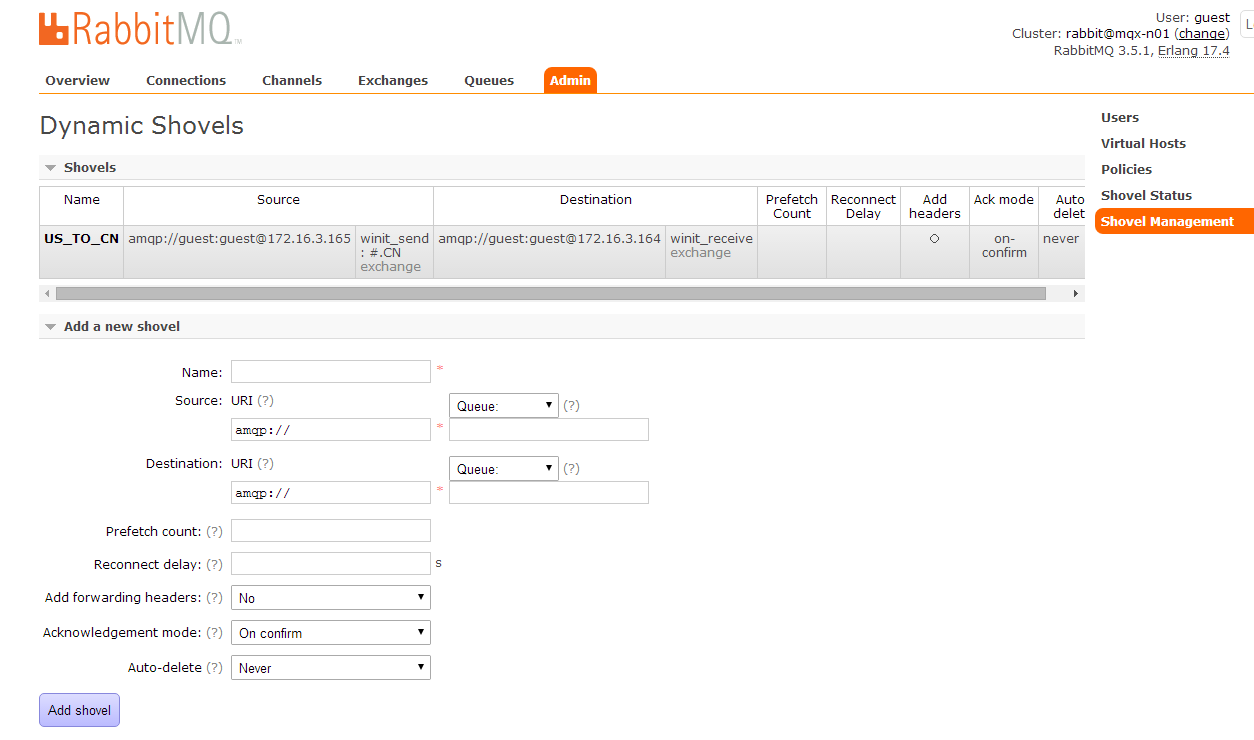
1. 创建队列。
2. 点击队列名称打开队列，使用GZIDC.#这个bind routing key建立到winit\_send的绑定关系。

* **Name：WINIT.SHOVEL.TO.USIDC**

1. 创建队列。
2. 点击队列名称打开队列，使用USIDC.#这个bind routing key建立到winit\_send的绑定关系。

### 建立shovel关系

进入相关数据中心的rabbitmq web控制台，点击admin/shovel management，按下面页面分别对数据中心的rabbitmq进行shovel配置：



* 中国rabbitmq
* **Name：USIDC\_TO\_GZIDC**

Source: amqp//用户名:密码@美国服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.GZIDC

Destination: amqp//用户名:密码@中国服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

Prefetch count:100 (后面的此配置项目都填100)

Reconnect delay:5 (后面的此配置项目都填5)

* **Name：AUIDC\_TO\_GZIDC**

Source: amqp//用户名:密码@澳洲服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.GZIDC

Destination: amqp//用户名:密码@中国服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* **Name：UKIDC\_TO\_GZIDC**

Source: amqp//用户名:密码@欧洲服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.GZIDC

Destination: amqp//用户名:密码@中国服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* 美国rabbitmq
* **Name：GZIDC\_TO\_USIDC**

Source: amqp//用户名:密码@中国服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.USIDC

Destination: amqp//用户名:密码@美国服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* **Name：AUIDC\_TO\_USIDC**

Source: amqp//用户名:密码@澳洲服务器ip:端口/，选择Queue后，WINIT.SHOVEL.TO.USIDC

Destination: amqp//用户名:密码@美国服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* **Name：UKIDC\_TO\_ USIDC**

Source: amqp//用户名:密码@欧洲服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.USIDC

Destination: amqp//用户名:密码@美国服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* 澳洲rabbitmq
* **Name：GZIDC\_TO\_AUIDC**

Source: amqp//用户名:密码@中国服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.AUIDC

Destination: amqp//用户名:密码@澳洲服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* **Name：USIDC\_TO\_AUIDC**

Source: amqp//用户名:密码@美国服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.AUIDC

Destination: amqp//用户名:密码@澳洲服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* **Name：UKIDC\_TO\_AUIDC**

Source: amqp//用户名:密码@欧洲服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.AUIDC

Destination: amqp//用户名:密码@澳洲服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* 欧洲rabbitmq
* **Name：GZIDC\_TO\_UKIDC**

Source: amqp//用户名:密码@中国服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.UKIDC

Destination: amqp//用户名:密码@欧洲服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* **Name：USIDC\_TO\_UKIDC**

Source: amqp//用户名:密码@美国服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.UKIDC

Destination: amqp//用户名:密码@欧洲服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

* **Name：AUIDC\_TO\_UKIDC**

Source: amqp//用户名:密码@澳洲服务器ip:端口/，选择Queue后，填入WINIT.SHOVEL.TO.UKIDC

Destination: amqp//用户名:密码@欧洲服务器ip:端口/，选择Exchange后，填入winit\_receive,routing key不用填

# 监控

rabbitmq日志：/var/log/rabbitmq/

web管理：http://xxx.xxx.xxx.xxx:15672