# NIFI cluster

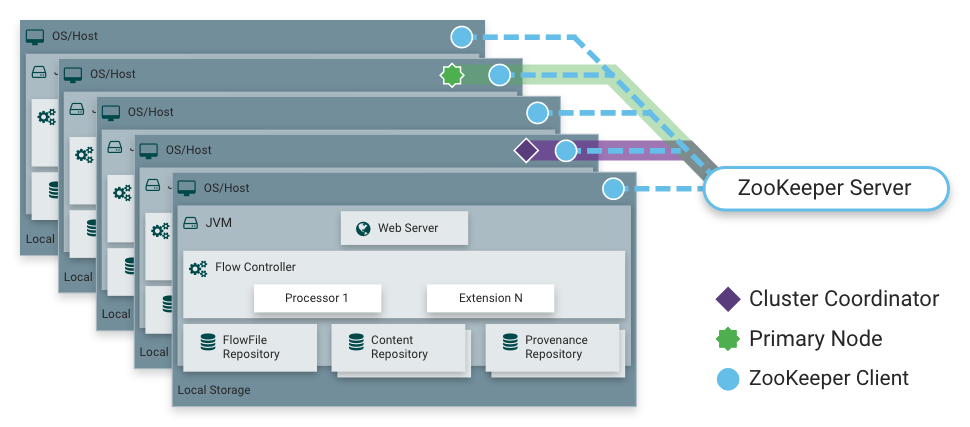
## 为什么使用集群

1. High-Availability
2. High-Throughput

如何集群化？

使用Apache Zookeeper提供自动选举不同的集群角色（clustering-related roles）

## NIFI集群架构



NIFI采用Zero-Master Clustering paradigm （after version 1.0.0）

集群中的每个节点对数据执行相同的任务，但每个节点都在不同的数据集合上运行。

在NIFI中有两个不同的角色，这两个角色都是通过Zookeeper自动选举的，第一个角色是**Primary Node，第**二个角色是**Cluster Coordinator。**

### ****Cluster Coordinator****

集群协调器是NIFI集群中的一个节点。

集群协调器会监控NIFI的所有节点，所有节点都会向其发送心跳，如果失败则被标记为Disconnected。

集群上的所有节点向集群协调器汇报状态信息

集群协调器负责执行管理允许哪些节点在集群中，并为新加入的节点提供最新的flow。

DataFlow Manager管理集群中的数据流时，他们可以通过集群中任何节点的用户界面进行此操作。 所做的任何更改都会复制到群集中的所有节点。

### Primary Node

主节点，每个集群上都有一个主节点，由zookeeper自动选举出来；

这个节点上可以运行隔离处理器（Isolated Processors）

### Isolated Processors

在NIFI集群中，相同的数据流在所有的节点上运行，因此，流中的每个组件都在每个节点上运行。但是在某些情况下DFM不希望每个处理器都在所有的节点上运行；例如GetSFTP处理器从远程目录中提取数据，如果每个节点都从远程目录提取数据，则会造成数据重复，因此DFM可以将主节点上的GetSFTP配置为独立运行

可以引入数据并且通过适当的数据流配置，在集群中的其余节点间进行负载均衡

### Heartbeats

节点通过“心跳”将其健康和状态发送给集群协调器；

缺省情况下，节点每5s发送一次心跳，如果在40s内没有接收到心跳则由于”缺少心跳”而断开节点



### Flow Election

当一个集群首次启动时，NiFi必须确定哪个节点具有“正确”版本的流。这是通过对每个节点具有的流进行投票来完成的。当一个节点试图连接到一个集群时，它为集群协调器提供了一个本地流（local flow）的副本。如果尚未选择“正确的”流，则将该节点的流与每个其他节点的流进行比较。如果另一个节点的流匹配这个流，则为此流投票。如果没有与其他节点有相同的流，那么这个流将被添加候选的流池中。经过一段时间后（通过nifi.cluster.flow.election.max.wait.time属性进行配置）或者某些节点数已投票（通过设置nifi.cluster.flow.election.max进行配置）。一个流被选为流的“正确”副本。所有具有不兼容流的节点将从群集中断开连接，而具有兼容流的节点将继承群集的流。选举是根据“大众投票”进行的，但要注意的是，除非所有流都是空的，否则赢家将永远不会成为"empty flow"。这允许管理员删除节点的flow.xml.gz文件并重新启动节点，知道节点的流不会被投票为“正确的”流，除非找不到其他流。

## 集群部署（simple）

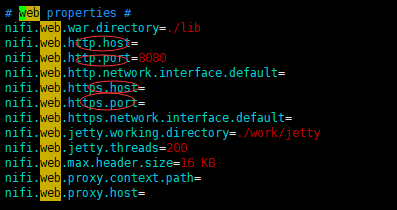
由三个NIFI实例组成的简单集群

注意：每个节点的配置文件尽量保持一致

### web properties section (nifi.properties)

设置http或者https的运行端口 （部署样例中http端口设置为8090）

设置http或者https的host



### cluster node properties(nifi.properties)

设置nifi.cluster.is.node为true

设置nifi.cluster.node.address，默认为localhost

设置nifi.cluster.node.protocol.port，端口大于1024(部署实例中设置为28001)

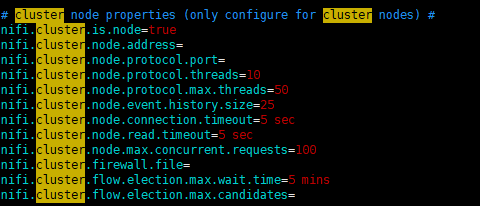
设置nifi.cluster.node.protocol.threads，默认值为10，为与其他节点通信时的线程数，理解为线程池的core size

设置nifi.cluster.node.protocol.max.threads，默认值为50，为与其他节点通信时的最大线程数，理解线程池线程数只能够增长到这个值

设置nifi.zookeeper.connect.string，连接zookeeper的字符串，用逗号隔开的hostname:port列表，包含所有的ZooKeeper quorum

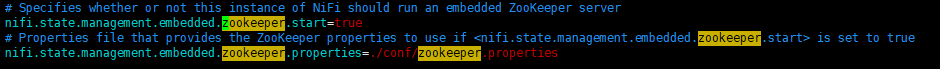
设置nifi.zookeeper.root.node，在zookeeper中的root节点

设置nifi.cluster.flow.election.max.wait.time，默认5分钟，flow通过选举变为“correct”版本的flow的等待时间



1. 如果使用NIFI内置的zookeeper（optional）

设置nifi.state.management.embedded.zookeeper.start为true



### 内置zookeeper配置 （optional）

1、./conf/zookeeper.propeties配置如下：

clientPort=12181

initLimit=10

autopurge.purgeInterval=24

syncLimit=5

tickTime=2000

dataDir=./state/zookeeper

autopurge.snapRetainCount=30

# 不同机器使用不同IP

server.1=10.6.6.82:12888:13888

server.2=10.6.6.83:12888:13888

server.3=10.6.6.84:12888:13888

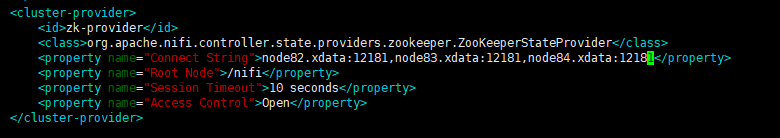
2、设置myid文件

设置文件目录：mkdir -p ${nifi\_home}/state/zookeeper

设置节点2的myid文件：echo "2" > ${nifi\_home}/state/zookeeper/myid

注：三个节点上都需要这个myid文件，节点一为1，节点二为2，节点三为3

3、更新state



## 集群部署（secured）

搭建安全的NIFI集群，并集成LADP完成身份验证以及用户授权

要在集群环境下使用LADP完成身份验证和用户授权，首先集群必须是secured的！

所以分两部分完成：一、部署安全的NIFI集群；二、集成LADP

以下是集群信息：

|  |  |  |
| --- | --- | --- |
| Hostname | NIFI版本 | NIFI-TKIT版本 |
| node82.xdata | 1.9.0 | 1.9.2 |
| node83.xdata | 1.9.0 | 1.9.2 |
| node84.xdata | 1.9.0 | 1.9.2 |

### Secured cluster setup

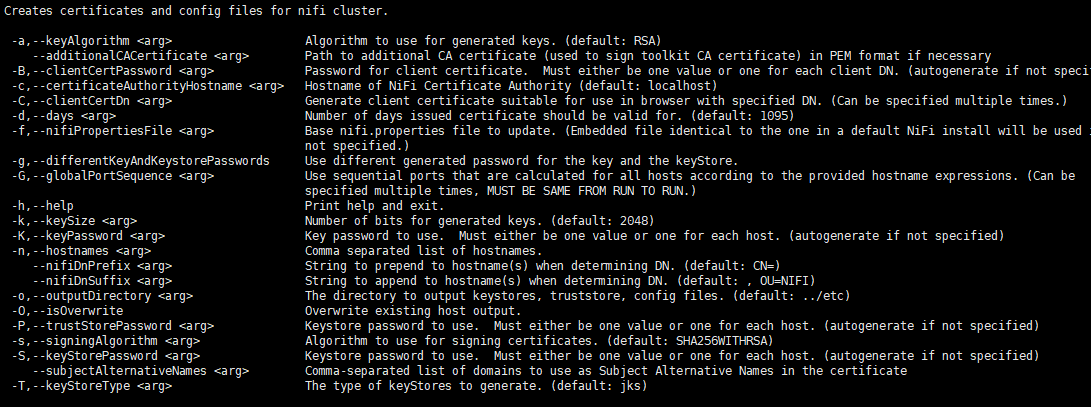
#### 下载nifi-toolkit-1.9.2-bin.tar.gz工具

wget <http://mirrors.tuna.tsinghua.edu.cn/apache/nifi/1.9.2/nifi-toolkit-1.9.2-bin.tar.gz>

并解压

首先查看工具tls使用方式

tls-toolkit.sh standalone|server|client --help



#### 解压nifi和nifi-toolkit

解压下载的nifi和toolkit，并创建一个etc目录，用于存放toolkit生成的文件



#### 启动ca server

在集群中任选一台，启动ca server，用于生成证书；注意请保证server的端口不要和其他服务冲突

执行：

../nifi-toolkit-1.9.2/bin/tls-toolkit.sh server -c node82.xdata -t sugonDippreventmitm -p 9999

#### 集群节点证书

为了保证节点之间通信的安全性，为节点生成证书；

在每个节点上一步中创建的etc目录下执行：

../nifi-toolkit-1.9.2/bin/tls-toolkit.sh client -c node82.xdata -t sugonDippreventmitm -p 9999

其中-c是ca sever的地址 -t是token，并且要保证大于16byte；

在ect目录下会存在如下文件：



文件会在nifi.properties中被使用，并且会默认使用hostname生成DN，例如node82.xdata节点DN为：CN=node82.xdata, OU=NIFI

#### 浏览器访问证书

同生成集群节点一样，首次使用浏览器访问时，需要添加浏览器访问证书

在etc目录下创建ui目录并在ui目录中执行：

/nifi-toolkit-1.9.2/bin/tls-toolkit.sh client -c node82.xdata -t sugonDippreventmitm -p 9999 -D "CN=sugonDip,OU=NIFI" -T PKCS12

在ui目录下会生成如下文件：



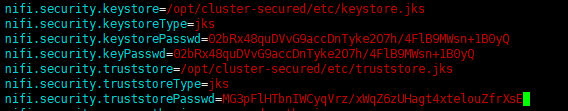
使用-T格式化证书，方便在浏览器导入

#### 配置nifi.properties（依次配置所有节点）

在nifi.properties的security段落，配置相应的属性；

文件位置在etc目录下，相应的config信息（如密码）在config.json文件中

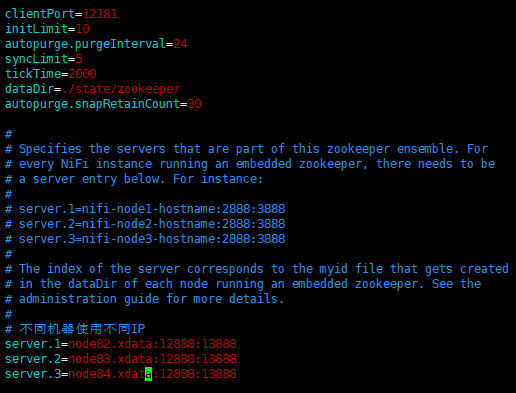
如下：



#### 集群配置

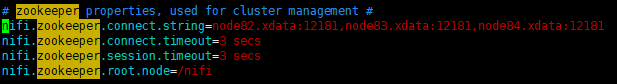
第一、使用内置zookeeper ./conf/zookeeper.properties

如下配置（所有节点保持一致）：



第二、nifi.properties（以下内容所有节点保持一致）



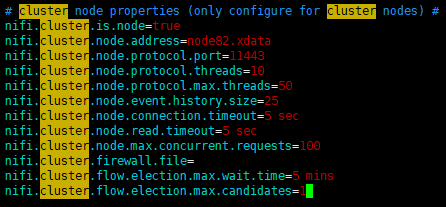


第三、创建myid（三个节点myid的内容分别为1，2，3，代表zookeeper server的id）

在nifi目录下：

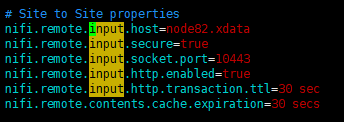
*mkdir ./state  
mkdir ./state/zookeeper  
echo 1 > ./state/zookeeper/myid*

第四、nifi.properties中配置



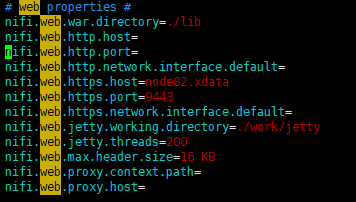
注：

*nifi.cluster.is.node=true  
nifi.cluster.node.address=值为每个节点的hostname*



注:

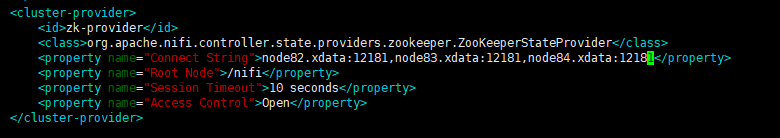
*nifi.remote.input.host=值为每个节点的hostname*



注：

设置host和port，http和https同时只能配置一个，要么是http为空要么是https为空

第五、更新state配置（所有节点保持一致）



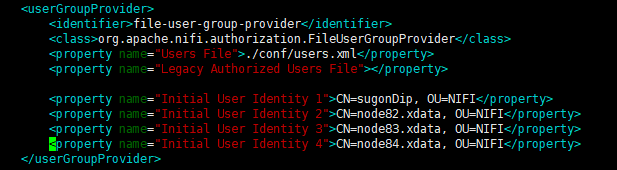
#### 设置authorizers.xml

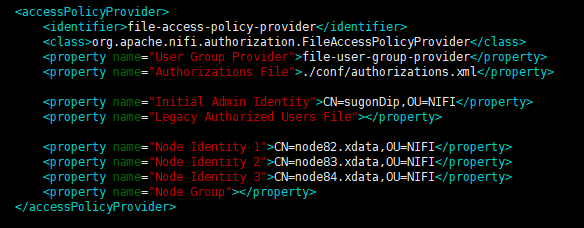
配置ui的初始访问权限和集群节点的访问权限

Initial Admin Identity为生成浏览器证书时的DN

Node Identity为集群节点生成证书时的DN

以上DN都可以在相应的config.properties中找到





注意：DN配置时，CN和OU之间默认是有空格的

#### 浏览器证书管理

以chrome浏览器为例：

在浏览器的设置-》安全-》证书管理中导入etc/ui/目录中生成的keystore.pkcs12文件，密码为config.json中的keyStorePassword

#### 浏览器访问

<https://10.6.6.82:9443/nifi/>

### 集成LADP

集成LADP需要修改的文件有如下：

* ./conf/nifi.properties
* ./conf/login-identity-providers.xml

在nifi.properties中需要设置

nifi.login.identity.provider.configuration.file

nifi.security.user.login.identity.provider

其中nifi.security.user.login.identity.provider=ladp-provider的值为配置file中的provider的identifier；

## 参考文档

<https://pierrevillard.com/2016/08/13/apache-nifi-1-0-0-cluster-setup/>

<https://pierrevillard.com/2016/11/29/apache-nifi-1-1-0-secured-cluster-setup/>

<https://pierrevillard.com/2017/01/24/integration-of-nifi-with-ldap/>

<https://bryanbende.com/development/2016/08/17/apache-nifi-1-0-0-authorization-and-multi-tenancy>

## Secured cluster的重要配置属性

