安装

这里是基于centos7

参考文档：

<https://learn.hashicorp.com/tutorials/consul/get-started-agent?in=consul/getting-started>

1、下载：

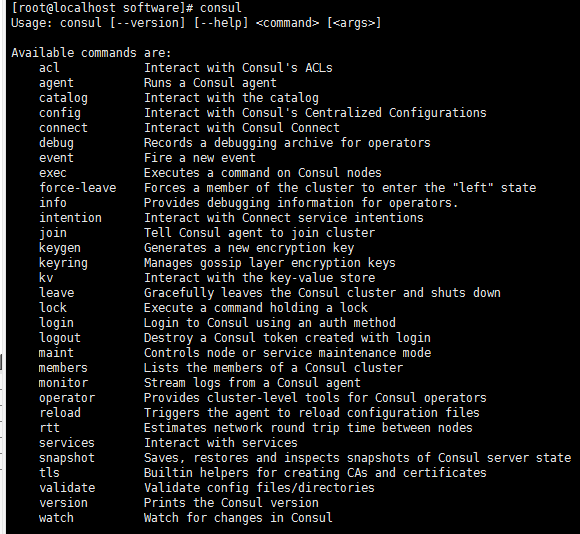
#wget https://releases.hashicorp.com/consul/1.8.0/consul\_1.8.0\_linux\_amd64.zip

配置path

export CONSUL\_HOME=/home/software/consul

export PATH=$PATH:$CONSUL\_HOME

验证：出现下面的信息，就算安装完成了。



启动consul agent

development mode：not secure or scalable. 开发模式具有基本的功能，所有持久化都被关闭。

server/client mode

**$** consul agent -dev

查看consul数据中心成员

**$** consul members

命令行是在client上执行的。返回的数据，返回的数据不以定和server端一致。agent和server端数据是最终一致性。要获取准确数据，通过api请求server

consul server 运行在8500端口

consul dns server 默认运行在8600端口

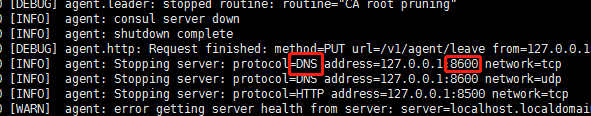
停止consul agent

**$** consul leave

Graceful leave complete



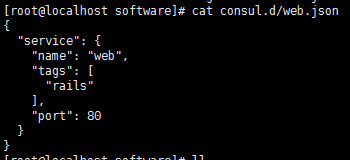
两个动作：1、leave the consul datacenter。2、shut down



这个日志也应证了端口对应

gracefully shut down consul agent

定义一个服务



重启agent

consul agent -dev -enable-script-checks -config-dir=./consul.d

允许健康检查，指定配置文件目录

service catalog —— 服务目录

服务没有运行也可以注册服务。consul会把运行的服务和端口联系起来。

多个agent consul datacenter情况，每个服务通过本地的consul client注册，client转发注册信息到consul servers，服务信息保存在sever中。

查询服务可以通过DNS接口和HTTP api

用consul注册的服务的dns名称是NAME.service.consul 这个Name就是注册服务用的名称，默认的，所有dns名称都在consul命名空间下，尽管可以配置

**$** dig @127.0.0.1 -p 8600 web.service.consul



要显示端口：增加SRV



其中的localhost.localdomain.node.dc1.consul. 这个是节点名

通过tags来过滤服务，基于tag的服务查询是TAG.NAME.service.consul 。例如，查询有rails tag的服务。

dig @127.0.0.1 -p 8600 rails.web.service.consul

通过http api查询服务

curl http://localhost:8500/v1/catalog/service/web

列出服务所有节点的地址

查找健康的服务实例

**$** curl 'http://localhost:8500/v1/health/service/web?passing'

更新服务

通过为服务注册一个health check来更新服务

**$** echo '{

"service": {

"name": "web",

"tags": [

"rails"

],

"port": 80,

"check": {

"args": [

"curl",

"localhost"

],

"interval": "10s"

}

}

}' > ./consul.d/web.json

服务定义的check部分添加了基于脚本的健康检查，它会尝试没10秒连接web service， 通过url。跟用户启动consul 进程一样的效果。

如果命令退出码>=2，健康检查失败，服务不健康。exit code = 1 是warning状态。

**$** consul reload

Configuration reload triggered

通过sonsul service mesh 连接 service

consul连接服务

sidecar proxies that you deploy locally with each service instance

local proxies that control network traffic between service instances

这种部署方式就是一个service mesh

Consul's service mesh 以前有的地方叫Consul Connect。这种叫法不准确的，慢慢修正过来。

consul service mesh

1、保证服务安全和观察服务之间的交互不用改服务的代码

2、consul 配置 sidecar proxies 在服务之间建立相同的TLS，是否允许服务之间通过注册名交互。

consul service mesh 控制着服务和服务之间的通信。他也可以收集服务的指标，然后输出到第三方聚合器，比如Prometheus

启动一个consul不知道的服务（start a consul-unaware service）

用socat来模拟一个服务。

**$** socat -v tcp-l:8181,fork exec:"/bin/cat"

用netcat来访问服务。

**$** nc 127.0.0.1 8181

定义注册服务

**$** echo '{

"service": {

"name": "socat",

"port": 8181,

"connect": {

"sidecar\_service": {}

}

}

}' > ./consul.d/socat.json

启动一个服务实例代理

**$** consul connect proxy -sidecar-for socat

注册一个依赖服务和代理

定义依赖服务

**$** echo '{

"service": {

"name": "web",

"connect": {

"sidecar\_service": {

"proxy": {

"upstreams": [

{

"destination\_name": "socat",

"local\_bind\_port": 9191

}

]

}

}

}

}

}' > ./consul.d/web.json

the connect部分指定了一个附接盒，它不是空的了，指定了web的上游依赖socat，代理监听的端口号9191， 和socat建立mTLS连接。

如果我们运行了一个真正的web service，它和它的代理交互通过一个回路地址。代理会加密他的数据，然后通过网络发送到socat的附接盒代理，socat的代理会解密数据，然后通过回路地址在8181端口本地发送给socat。因为这里没有web service运行，需要假装有一个web service在指定的端口9191上和其代理交互。

用附接盒注册的配置启动web proxy

**$** consul connect proxy -sidecar-for web

用intentions控制交互

intentions定义了允许哪些服务连接其他的服务。上面的连接成功，是因为在开发模式中，ACL（access control list）默认allow all。

创建intention。指定策略，源服务，目标服务

**$** consul intention create -deny web socat

删除intention

**$** consul intention delete web socat

intentions允许你分割网络像传统的firewall，但是它依赖于服务的逻辑名（web，socat），而不是每个服务实例的ip地址。

用consul KV存储数据。

consul的key value 存储。可以用来动态的配置应用，调节服务，管理leader选举，作为一个数据后端库。还有其他很多应用。

和consul key value store交互有两种方式：the HTTP API and the CLI.

下面我们用CLI来探索consul key value store

添加kv 数据

**$** consul kv put redis/config/minconns 1

**$** consul kv put redis/config/maxconns 25

增加一个flag

**$** consul kv put -flags=42 redis/config/users/admin abcd1234

查询key的value

consul kv get redis/config/minconns

获取key-value的额外的metadata信息

**$** consul kv get -detailed redis/config/users/admin

列出store中的所有keys

**$** consul kv get -recurse

删除数据

**$** consul kv delete redis/config/minconns

批量操作keys

**$** consul kv delete -recurse redis

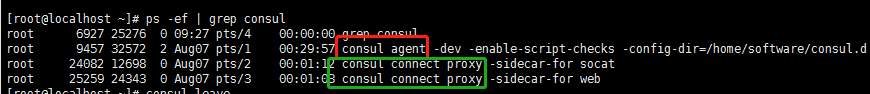
批量删除redis前缀的keys

修改已经存在的key-value

**$** consul kv put foo bar

consul UI

If you were running Consul in production you would need to enable the UI in Consul's configuration file or using the -ui command line flag, but because your agent is running in development mode, the UI is automatically enabled.



要访问ui 需要在前面的启动命令，加上 -client 0.0.0.0，如下：

consul agent -dev -enable-script-checks -config-dir=/home/software/consul.d -client 0.0.0.0

ui访问地址：<http://10.0.92.111:8500/ui>

创建一个本地consul 数据中心

启动一个新的consul agent，它是不知道其他的agent的。数据中心对一个成原来说很重要。

agents相互了解有两种方法：

添加一个新的agent到已存在的datacenter（client or server），在datacenter中给出其他agent的ip地址，这样新的agent就加入到datacenter中了。一旦agent是一个datacenter的一个成员，他会自动的通过广播的方式知道其他agent。

设置环境

consul是一个分布式程序，每台机器一个agent。这里用vagrant跑虚拟机来模拟分布式环境。

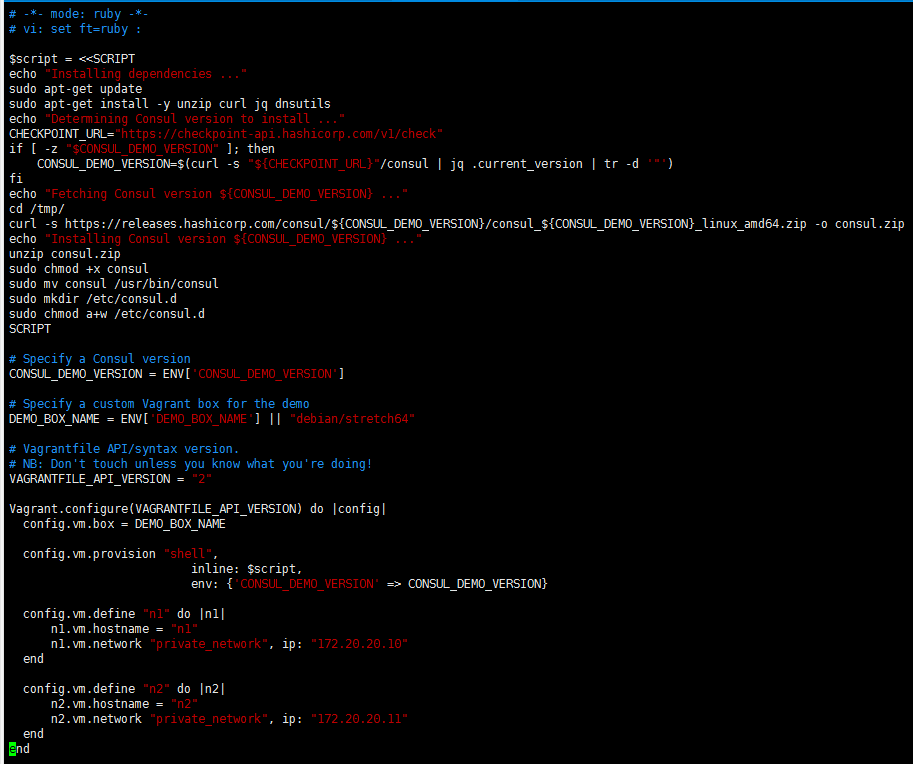
创建一个目录，放vagrant配置文件

**$** mkdir consul-getting-started-join

创建一个文件Vagrantfile

|  |
| --- |
| # -\*- mode: ruby -\*-  # vi: set ft=ruby :  $script = <<SCRIPT  echo "Installing dependencies ..."  sudo apt-get update  sudo apt-get install -y unzip curl jq dnsutils  echo "Determining Consul version to install ..."  CHECKPOINT\_URL="https://checkpoint-api.hashicorp.com/v1/check"  if [ -z "$CONSUL\_DEMO\_VERSION" ]; then  CONSUL\_DEMO\_VERSION=$(curl -s "${CHECKPOINT\_URL}"/consul | jq .current\_version | tr -d '"')  fi  echo "Fetching Consul version ${CONSUL\_DEMO\_VERSION} ..."  cd /tmp/  curl -s https://releases.hashicorp.com/consul/${CONSUL\_DEMO\_VERSION}/consul\_${CONSUL\_DEMO\_VERSION}\_linux\_amd64.zip -o consul.zip  echo "Installing Consul version ${CONSUL\_DEMO\_VERSION} ..."  unzip consul.zip  sudo chmod +x consul  sudo mv consul /usr/bin/consul  sudo mkdir /etc/consul.d  sudo chmod a+w /etc/consul.d  SCRIPT  # Specify a Consul version  CONSUL\_DEMO\_VERSION = ENV['CONSUL\_DEMO\_VERSION']  # Specify a custom Vagrant box for the demo  DEMO\_BOX\_NAME = ENV['DEMO\_BOX\_NAME'] || "debian/stretch64"  # Vagrantfile API/syntax version.  # NB: Don't touch unless you know what you're doing!  VAGRANTFILE\_API\_VERSION = "2"  Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config|  config.vm.box = DEMO\_BOX\_NAME  config.vm.provision "shell",  inline: $script,  env: {'CONSUL\_DEMO\_VERSION' => CONSUL\_DEMO\_VERSION}  config.vm.define "n1" do |n1|  n1.vm.hostname = "n1"  n1.vm.network "private\_network", ip: "172.20.20.10"  end  config.vm.define "n2" do |n2|  n2.vm.hostname = "n2"  n2.vm.network "private\_network", ip: "172.20.20.11"  end  end |

这个文件告诉Vagrant创建2个虚拟机，并且预装了Consul binary



启动两个虚拟机

**$** vagrant up

虚拟机的过程调过。

启动consul agent

$ consul agent \

-server \

-bootstrap-expect=1 \

-node=agent-one \

-bind=10.0.92.111 \

-data-dir=/home/software/consul-data \

-config-dir=/home/software/consul.d

用server mode启动agent

参数：

-server 指定agent以server mode 启动

-bootstrap-expect 告诉consul server，datacenter应该一共有几个servers。所有的servers启动副本日志之前都会等待这么多数量server注册，副本日志保证所有servers的数据一致性。

因为只启动了一个只有一个server的datacenter，所以设置值为1.

-node name datacenter中的每个node都必须有一个唯一的名字。默认，使用机器的hostname，通常改写，设置为agent-one

-bind address 这个就是agent监听来和其他consul成员通信的地址。它必须是一个datacenter所有节点都能访问的地址。如果没有设置bind address，consul会尝试所有ipv4接口，如果找到多个私有ip会启动失败。因为生产servers通常有多个接口，应该总是提供一个bind address。

data-dir 告诉consul agent 应该把状态存放在这里。它可能会包括如servers和clients的ACL tokens这样的敏感数据。生产部署应当小心这个目录的权限，一般/tmp/consul

config-dir 告诉consul从哪儿去找他的配置文件，一般是/etc/consul.d

在另一台机器在启动一个，以client模式（不是dev也不是server）

$ consul agent \

-node=agent-two \

-bind=10.0.92.140 \

-enable-script-checks=true \

-data-dir=/home/software/consul-data \

-config-dir=/home/software/consul.d

现在启动了2个consul agent。一个server模式，一个是client模式。这两个agent都不知道对方，他们各自拥有自己的只有一个node的datacenter。

查看量个机器上的consul members





连接这两个agents

multi-agent datacenter

在sever机器上执行

$ consul join 10.0.92.140

join的时候默认用的端口是8301，需要保证8301这个端口开放。

 the client had been throwing warnings and errors indicating that no servers were available. When the client learned about the server, it stopped throwing errors and synced its node information.

consul client没有一个server的是不能正常工作的。所有的datacenter必须有一个agent用server模式运行，consul才能运行正常。

如果datacenter中有多个server，超过半数的servers与其他server一直保持通信，datacenter才能运行正常。

要加入一个datacenter，一个consul agent只需要知道另外一个agent就可以了，不管是client还是server。加入datacenter后，agents会自动的相互gossip来传播全量的membership information

自动加入

用-join flag 或者start\_join setting

查询node

DNS interface或者http API

DNS interface

名称结构是：NAME.node.consul or NAME.node.DATACENTER.consul。如果没有datacenter，consul只会search本地datacenter。

$ dig @127.0.0.1 -p 8600 agent-two.node.consul

问题：

1、consul 上注册的服务不可用，怎么立即清除？

某个服务曾经用用了某个端口，后来换成另一个应用使用这个端口，因为端口和ip没变，consul认为以前的服务还正常。导致客户端请求以前的服务失败。

删除服务：

命令：

consul services deregister -id=consul-config-center-8082

api：

$ curl \

--request PUT \

http://10.0.92.111:8500/v1/agent/service/deregister/sdfsdfsdf-8083