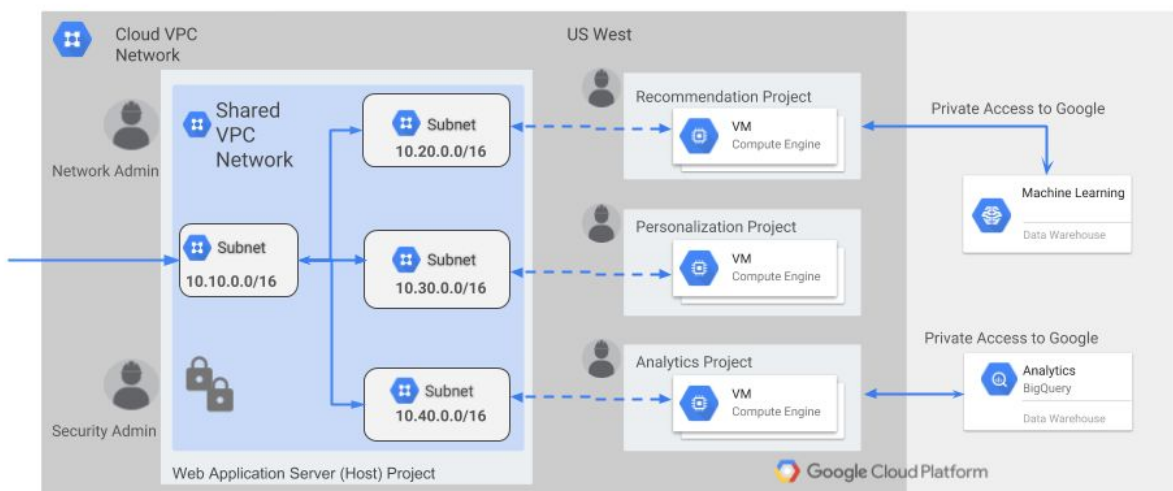


# Google Cloud中配置Shared VPC

## 一 Shared VPC介绍

Google Cloud中有一种特殊的VPC，叫Shared VPC。这个VPC可以共享给其他Project使用。如下图：

### Shared VPC: Configuring Network Topology



在Shared VPC的使用中，有两种Project：Host Project和服务Project。上图中Shared VPC Network所在的蓝色的Project即是Host Project；而VM所在的Project为Service Project。在这种架构中，Network的组件，包括VPC、Subnet、IP Address、防火墙等都在Host Project中的Shared VPC中定义，Shared VPC把这些资源共享给Service Project，在Service Project中就可以使用这些资源了。

## 二 Shared VPC的配置

下面介绍Shared VPC的具体配置，包括Shared VPC权限、Host Project的配置和服务Project的使用。

### 1 Shared VPC的权限

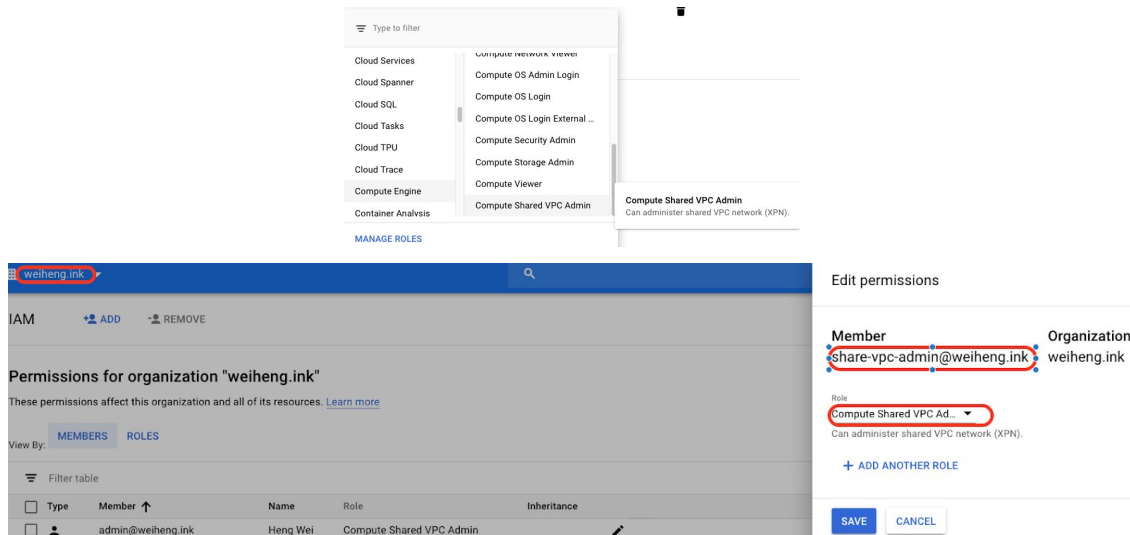
由于Shared VPC要在多个Project中共享资源，所以开启Shared VPC要求有Project在Organization下。Organization的开启方法请参考“Google Cloud IAM中添加自定义域名”。

## A. 资源准备：

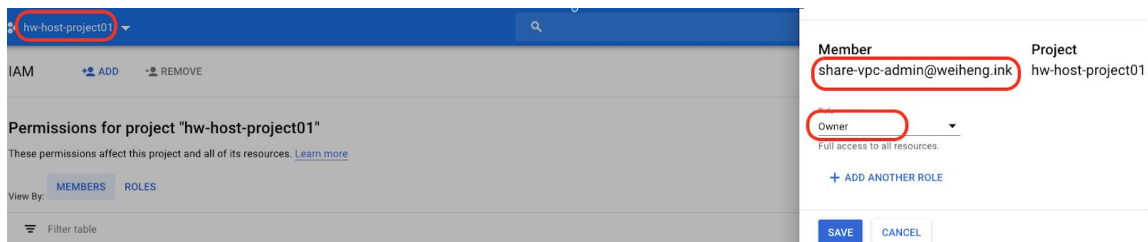
- 3个用户：share-vpc-admin, p1-vm-admin, p2-vm-admin
  - 3个Project：hw-host-project01, hw-svc-project01, hw-svc-project02
- 从名字可以看出各个用户和project的角色和功能。

## B. 具体授权的操作如下：

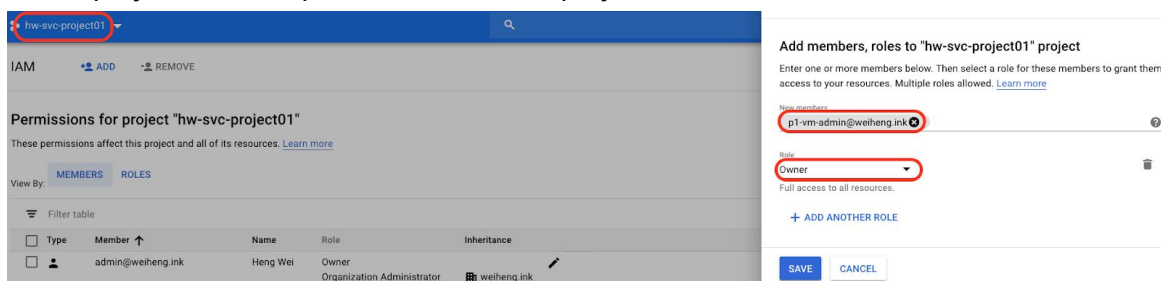
在weiheng.ink的组织下，给share-vpc-admin赋予Compute Shared VPC Admin（这个权限在Compute Engine下）的权限：



在hw-host-project的IAM中，给share-vpc-admin赋予：project owner权限：



在hw-svc-project01内，给p1-vm-admin赋予：project owner的权限：



类似的，在hw-svc-project02内给p2-vm-admin赋予project owner的权限。

## 2 Host Project中的配置

### A. 创建VPC

VPC 网络

VPC 网络

当前项目中没有任何本地 VPC 网络

创建 VPC 网络

### B.创建两个子网sharevlan1和sharevlan2

hw-host-project01

创建 VPC 网络

名称

corpvpn

说明 (可选)

子网

您可以利用子网在 Google Cloud 内创建专用云端拓扑。点击“自动”可在每个区域创建一个子网，点击“自定义”则可手动定义子网。[了解详情](#)

子网创建模式

自定义 自动

sharevlan1

新子网

名称

sharevlan2

添加说明

区域

asia-southeast1

IP 地址范围

10.2.0.0/16

创建次要 IP 范围

专用 Google 访问权限

开启

关闭

流日志

开启

关闭

完成

取消

+ 添加子网

## C. 开启 shared VPC



## D. 按照向导设置 Shared VPC

出现设置共享VPC的界面，共分三步：启用宿主项目，选择子网和授予权限：



## ← 设置共享的 VPC

☒ 启用宿主项目    ☒ 选择子网    3 授予权限

要共享您的子网，您需要向用户授予“计算网络用户”角色。以下是具体的操作方法：

1. 关联服务项目
2. 按角色选择用户

### 关联服务项目 <sup>?</sup>

可以在所选子网或宿主项目中向关联项目内的用户授予“计算网络用户”角色

按项目名称或 ID 过滤

<input type="checkbox"/> 项目名称 ^	项目 ID	标签
<input checked="" type="checkbox"/> hw-svc-project01	hw-svc-project01	
<input type="checkbox"/> hw-svc-project02	hw-svc-project02	

已选中 1 个项目

### 按角色选择用户 <sup>?</sup>

选择一个或多个角色。关联项目中拥有所选角色的用户将被授予选定子网或宿主项目的“计算网络用户”角色。

- ☒ 计算实例管理员 <sup>?</sup>
- ☒ 计算网络管理员 <sup>?</sup>
- ☒ 所有者 <sup>?</sup>
- ☒ 修改者 <sup>?</sup>

### Kubernetes Engine 访问权限 <sup>?</sup>

☐ 已启用

保存

取消

这样，sharevlan1就共享给了hw-svc-project01项目。

## E. 手工设置Shared VPC共享

也可以通过配置关联项目实现共享：

## ← 共享的 VPC 管理所有共享的 VPC

此项目 (hw-host-project01) 是一个宿主项目

它正在与任何关联的项目共享子网

共享的子网和权限    关联的项目

关联项目

取消关联项目

<input type="checkbox"/> 项目名称 ^	项目 ID	
<input type="checkbox"/> hw-svc-project01	hw-svc-project01	⋮

选择子网和相应的项目：

## ← 将项目附加到宿主项目

### 附加的项目

选择您要与其共享 VPC 网络的项目

按项目名称或 ID 过滤

<input checked="" type="checkbox"/> 项目名称 ^	项目 ID	标签
<input checked="" type="checkbox"/> hw-svc-project02	hw-svc-project02	

已选中 1 个项目

### VPC 网络权限

默认情况下，具有这些角色的用户能够使用您的 VPC 网络。您可以在以后自定义这些权限。

- ☒ 计算实例管理员 ?
- ☒ 计算网络管理员 ?
- ☒ 所有者 ?
- ☒ 修改者 ?

### Kubernetes Engine 访问权限 ?

☐ 已启用

### 共享模式

- ☐ 共享所有子网（项目级权限）  
此项目中的所有子网（包括将来创建的子网）将被共享。
- ☒ 个别子网（子网级权限）  
您要共享的单独的子网。将来创建的子网将不会自动被共享。

### 要共享的子网

项目“hw-host-project01”中的以下子网将与您选择的其他项目共享。

取消选中子网可移除所有权限。您可以在以后更改此设置。

<input type="checkbox"/> 子网 ^	区域	VPC 网络	IP 地址范围
<input type="checkbox"/> sharevlan1	asia-east1	corpvpc	10.1.0.0/16
<input checked="" type="checkbox"/> sharevlan2	asia-southeast1	corpvpc	10.2.0.0/16

将共享 1 个子网

这样配置以后，Sharevlan1共享给了hw-svc-project01，Sharevlan2共享给了hw-svc-project02。

## F. 创建防火墙规则

Shared VPC的防火墙规则需要集中统一在host project中配置，在Service Project中不能自行配置防火墙规则。

hw-host-project01

← 创建防火墙规则

名称

allow-ssh-icmp

说明 (可选)

日志

开启防火墙日志可能会生成大量的日志，这可能会增加 Stackdriver 的费用。[了解详情](#)

☐ 开启

☒ 关闭

网络

corpvpc

优先级

优先级可以从 0 到 65535 [检查其他防火墙规则的优先级](#)

1000

流量方向

☒ 入站

☐ 出站

对匹配项执行的操作

☒ 允许

☐ 拒绝

目标

网络中的所有实例

来源过滤条件

IP 地址范围

来源 IP 地址范围

0.0.0.0/0

次要来源过滤条件

无

协议和端口

☐ 全部允许

☒ 指定的协议和端口

☒ tcp : 22

☐ udp : 全部

☒ 其他协议 icmp

停用规则

创建

取消

### 3 在Service Project中使用Shared VPC的资源

在Service Project中，用户可以像使用自己创建的VPC资源一样使用Shared VPC中的网络资源。

使用p1-vm-admin用户登录， 在hw-svc-project01中在sharevlan1的子网中， 创建VM1：

hw-svc-project01

创建实例

名称

vm1

区域

asia-east1 (台湾)

地区

asia-east1-b

机器类型

进行自定义以选择核心、内存和 GPU 数量。

微型 (1 个共享 ...)

0.6 GB 内存

自定义

容器

☐ 将一个容器映像部署到此 VM 实例。[了解详情](#)

启动磁盘

新的 10 GB 标准永久性磁盘

映像

Debian GNU/Linux 9 (stretch)

更改

身份和 API 访问权限

服务帐号

Compute Engine default service account

访问权限范围

☒ 允许默认访问权限

☐ 允许所有 Cloud API 的全面访问权限

☐ 针对每个 API 设置访问权限

防火墙

添加标记和防火墙规则，允许来自互联网的特定网络流量

☐ 允许 HTTP 流量

☐ 允许 HTTPS 流量

配置为使用共享子网的模板将无法使用防火墙设置

管理

安全

磁盘

网络

单独租用

网络标记

(可选)

网络接口

网络接口

☐ 此项目中的网络

☒ (通过宿主项目 "hw-host-project01") 与我共享的网络

共享的子网

sharevlan1

主要内部 IP

临时 (自动)

显示别名 IP 范围

外部 IP

临时

网络服务层级

☒ 优质 (当前项目级层级, 更改)

☐ 标准 (asia-east1)

所选区域无法使用标准层级。标准层级目前可在以下区域使用: us-central1, us-east1, europe-west1。

IP 转发

关闭

公开 DNS PTR 记录

☐ 启用

PTR 域名

完成

取消

+ 添加网络接口

收起

您需要为此实例付费。[了解详情](#)

创建

取消

同样， 在hw-svc-project02中创建VM2。

hw-svc-project01

VM 实例

创建实例

导入 VM

刷新

启动

过滤 VM 实例

<input type="checkbox"/>	名称 ^	地区	建议	内部 IP	外部 IP	连接
<input checked="" type="checkbox"/>	vm1	asia-east1-b		10.1.0.2 (nic0)	35.221.145.41	SSH

hw-svc-project02

VM 实例

创建实例

导入 VM

刷新

启动

停止

过滤 VM 实例

<input type="checkbox"/>	名称 ^	地区	建议	内部 IP	外部 IP	连接
<input checked="" type="checkbox"/>	vm2	asia-southeast1-b		10.2.0.2 (nic0)	35.187.237.165	SSH

创建成功后， VM间可以相互ping通：



```

root@vml:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1460
    inet 10.1.0.2 netmask 255.255.255.255 broadcast 10.1.0.2
    inet6 fe80::4001:aff:fe01:2 prefixlen 64 scopeid 0x20<link>
    ether 42:01:0a:01:00:02 txqueuelen 1000 (Ethernet)
    RX packets 218 bytes 41779 (40.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 180 bytes 23690 (23.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@vml:~# ping 10.2.0.2
PING 10.2.0.2 (10.2.0.2) 56(84) bytes of data.
64 bytes from 10.2.0.2: icmp_seq=1 ttl=64 time=54.3 ms
64 bytes from 10.2.0.2: icmp_seq=2 ttl=64 time=49.7 ms
64 bytes from 10.2.0.2: icmp_seq=3 ttl=64 time=49.6 ms
64 bytes from 10.2.0.2: icmp_seq=4 ttl=64 time=49.7 ms
64 bytes from 10.2.0.2: icmp_seq=5 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=6 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=7 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=8 ttl=64 time=49.6 ms
64 bytes from 10.2.0.2: icmp_seq=9 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=10 ttl=64 time=49.6 ms
64 bytes from 10.2.0.2: icmp_seq=11 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=12 ttl=64 time=49.6 ms
64 bytes from 10.2.0.2: icmp_seq=13 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=14 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=15 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=16 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=17 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=18 ttl=64 time=49.6 ms
64 bytes from 10.2.0.2: icmp_seq=19 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=20 ttl=64 time=49.6 ms
64 bytes from 10.2.0.2: icmp_seq=21 ttl=64 time=49.5 ms
64 bytes from 10.2.0.2: icmp_seq=22 ttl=64 time=49.6 ms
64 bytes from 10.2.0.2: icmp_seq=23 ttl=64 time=49.5 ms
^C
--- 10.2.0.2 ping statistics ---
23 packets transmitted, 13 received, 43% packet loss, time 22267ms
rtt min/avg/max/mdev = 49.568/49.998/54.344/1.259 ms
root@vml:~#

root@vm2:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1460
    inet 10.2.0.2 netmask 255.255.255.255 broadcast 10.2.0.2
    inet6 fe80::4001:aff:fe02:2 prefixlen 64 scopeid 0x20<link>
    ether 42:01:0a:02:00:02 txqueuelen 1000 (Ethernet)
    RX packets 172 bytes 30131 (29.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 131 bytes 18866 (18.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@vm2:~# ping 10.1.0.2
PING 10.1.0.2 (10.1.0.2) 56(84) bytes of data.
64 bytes from 10.1.0.2: icmp_seq=1 ttl=64 time=51.0 ms
64 bytes from 10.1.0.2: icmp_seq=2 ttl=64 time=49.5 ms
64 bytes from 10.1.0.2: icmp_seq=3 ttl=64 time=49.6 ms
64 bytes from 10.1.0.2: icmp_seq=4 ttl=64 time=49.6 ms
64 bytes from 10.1.0.2: icmp_seq=5 ttl=64 time=49.5 ms
64 bytes from 10.1.0.2: icmp_seq=6 ttl=64 time=49.5 ms
64 bytes from 10.1.0.2: icmp_seq=7 ttl=64 time=49.6 ms
64 bytes from 10.1.0.2: icmp_seq=8 ttl=64 time=49.5 ms
64 bytes from 10.1.0.2: icmp_seq=9 ttl=64 time=49.6 ms
64 bytes from 10.1.0.2: icmp_seq=10 ttl=64 time=49.6 ms
64 bytes from 10.1.0.2: icmp_seq=11 ttl=64 time=49.6 ms
64 bytes from 10.1.0.2: icmp_seq=12 ttl=64 time=49.5 ms
64 bytes from 10.1.0.2: icmp_seq=13 ttl=64 time=49.6 ms
64 bytes from 10.1.0.2: icmp_seq=14 ttl=64 time=49.5 ms
^C
--- 10.1.0.2 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13023ms
rtt min/avg/max/mdev = 49.576/49.720/51.033/0.436 ms

```

## 三 总结

Google Cloud中的Shared VPC是一种集中管理网络资源的方法。在一个企业中，网络安全管理往往和系统管理、软件管理分开。为方便的对网络进行统一的管理和配置，便有了Shared VPC这样的功能。