目录

1 大概思路

- Nginx 集群之 WCF 分布式身份验证
- BasicHttpBinding、Ws2007HttpBinding
- Windows 证书生成公钥、私钥（x509 证书）
- 编写 WCF 服务、客户端程序
- URL 保留项
- 部署 WCF 服务程序到局域网内 3 台 PC 机
- Nginx 集群配置搭建
- SoapUI 和 WCF 客户端程序的运行结果
- 总结

2 Nginx 集群之 WCF 分布式身份验证

Nginx 是个轻量级的反向代理，当然，也有相应的 SSL 身份认证。本文主要采用一种自我寄宿的方式，使用 Nginx 集群，通过 windows 证书(X.509 证书)，讲述客户端如何访问服务器的方法。

本文源代码主要分类：

- HighlyConcurrentHosting

使用 BasicHttpBinding 的源代码（本文主要以这种方式进行 Nginx 集群）

- HighlyConcurrentHosting_Ws2007HttpBinding

使用 ws2007HttpBinding 的源代码（这是一种点对点的 Windows 认证方式）

- Nginx 配置

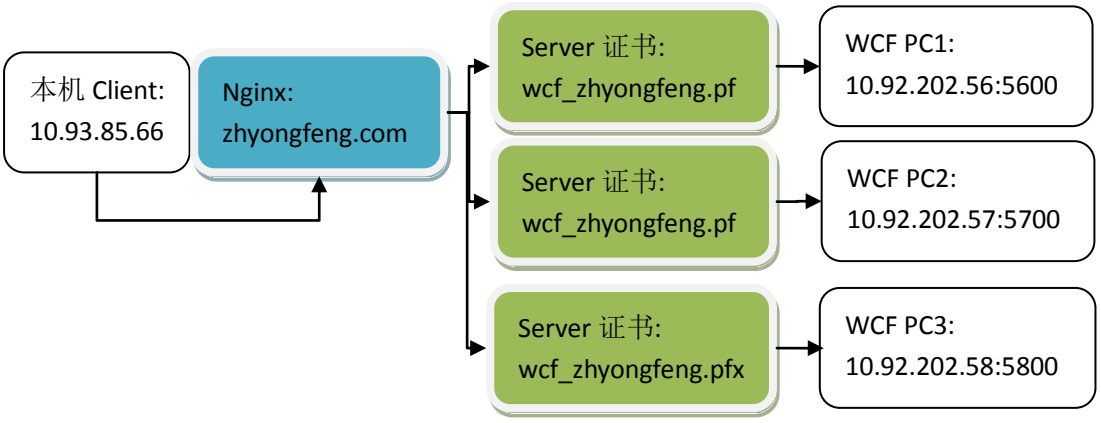
nginx.conf

- Windows 证书

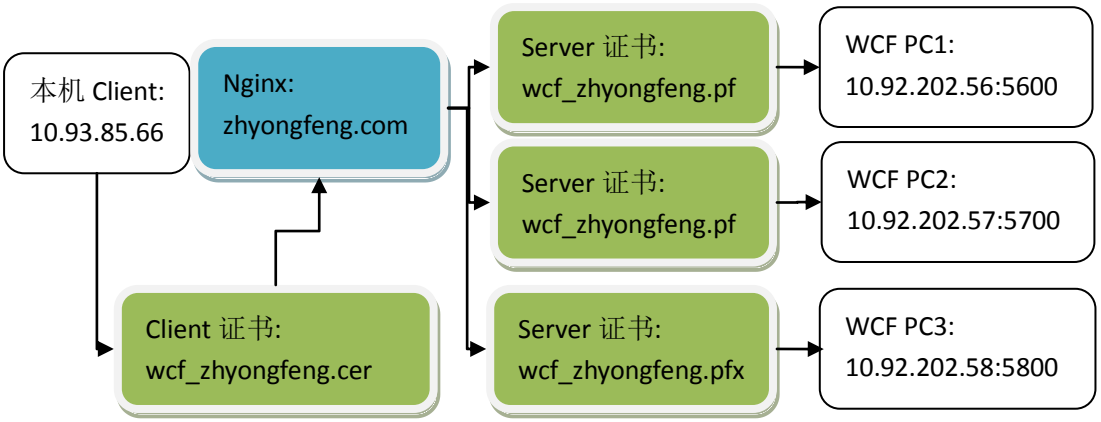
wcf_zhyongfeng.cer（客户端需要安装的证书）
wcf_zhyongfeng.pfx（服务器需要安装的证书）

以下是本文讲述的主要结构图：

客户端以 BasicHttpBinding 进行访问 Nginx，然后 Nginx 进行负载均衡，将消息分发到后端任意一台 WCF 的 PC 机，后端需要被访问的 WCF 服务器，都要安装 Windows 证书（仅被访问的服务器需要安装 wcf_zhyongfegn.pfx 证书）。



若以 ws2007HttpBinding 进行点对点 Windows 认证，则 Nginx 只能起到通过划分 IP 绑定特定一台服务器访问的作用，并不能起到集群负载均衡，同时除了服务器要 wcf_zhyongfeng.pfx 安装证书外，客户端也需要进行安装 wcf_zhyongfeng.cer 证书，这里不作为重点讲述。



3 BasicHttpBinding、ws2007HttpBinding

这里 WCF 的 Ningx 集群，主要用的是 BasicHttpBinding。BasicHttpBinding 的默认安全模式是 None，即没有任何安全设置，消息都以明文传送，对客户端也不进行验证。 但是 basicHttpBinding 绑定可以实现安全传输，也可以通过传输层和消息层来保证消息的安全性。basicHttpBinding 设置为 Transport 安全模式，传输层的安全是使用 IIS 的安全机制，比如基本身份验证、集成 windows 验证、SSL 安全通道等等

.NET Framework 3.5 介绍了一种用于 Web 服务交互称为 ws2007HttpBinding 绑定的新的绑定。这个绑定类似于 ws2007HttpBinding 绑定除了它支持最新的 WS-* 消息，安全，可信赖消息和事务标准。

ws2007HttpBinding 支持的标准：

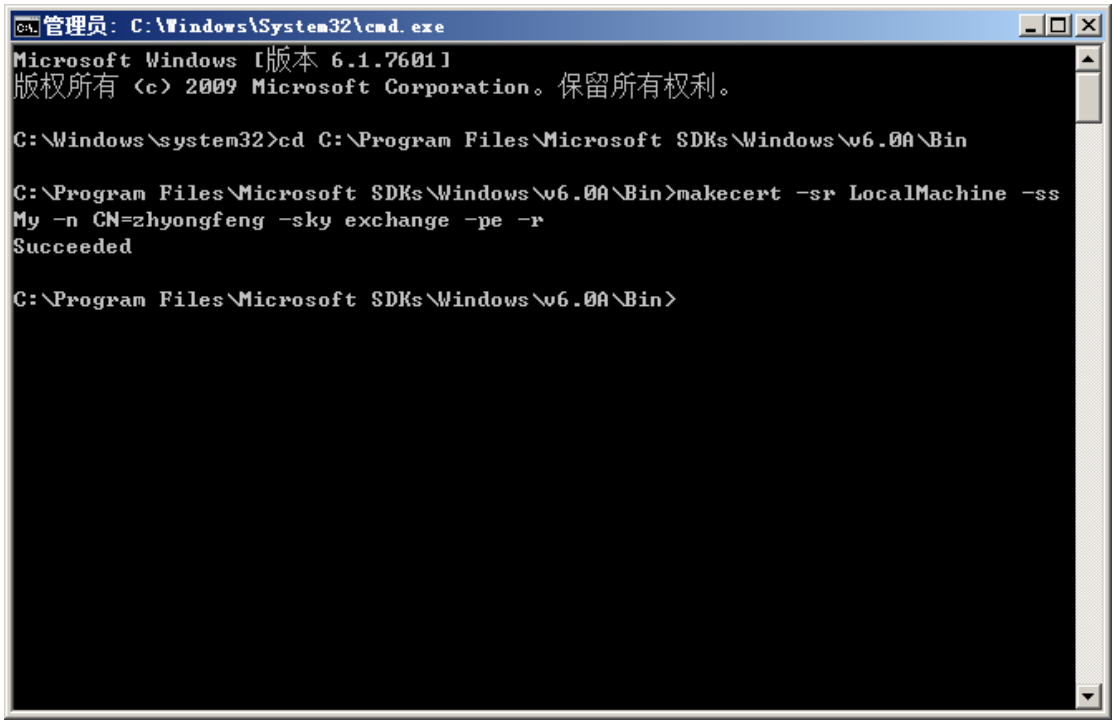
WS-SecureConversation v1.3	WS-Security 的扩展，为多个消息交换提供一个安全上下文
WS-Trust v1.3	WS-Security 的扩展，请求并标记问题，管理

	可依赖关系。
WS-SecurityPolicy v1.2	WS-Security 的安全断言，WS-Security 转换以及使用 WS-Policy 表达的 WS-Trust
Web Services Reliable Messaging v1.1	保证消息被传递，适当编码且不会重复接收的协议
Web Services Coordination v1.1	为分布式平台的动作合作提供协议的平台

4 Windows 证书生成私钥、公钥（X.509 证书）

进行 C:\Windows\system32，以管理员运行 cmd.ext

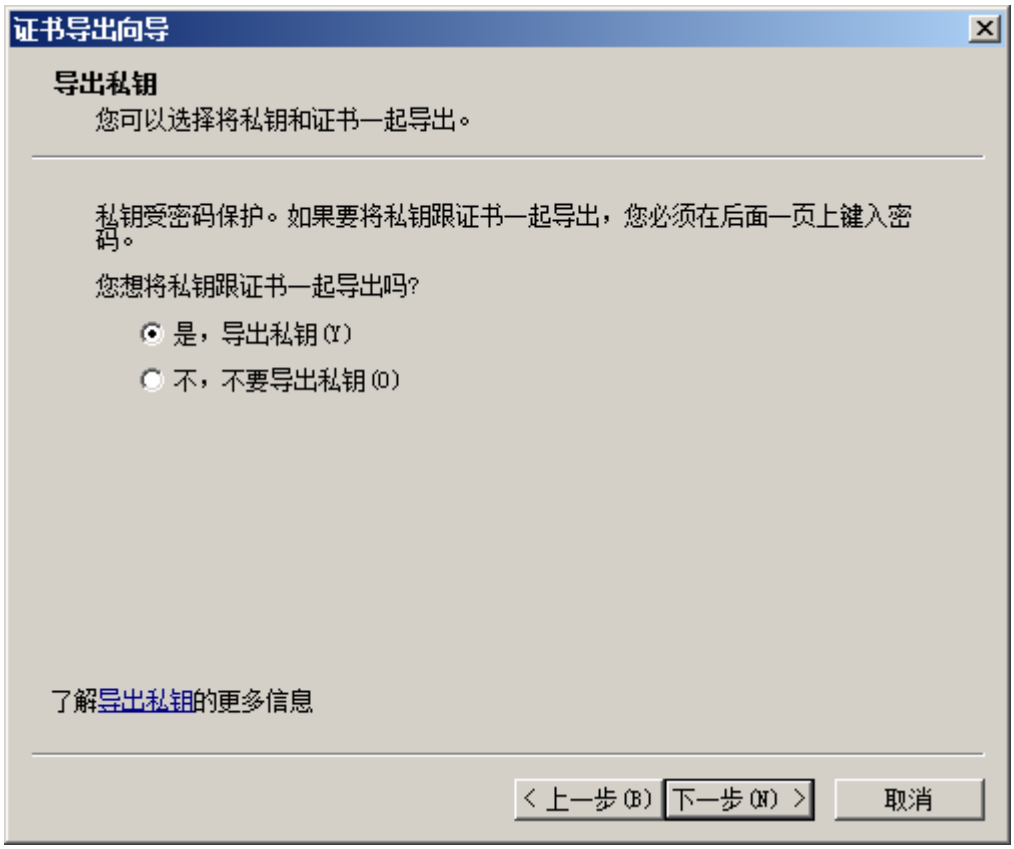
```
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Windows\system32>cd C:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin

C:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin>makecert -r -pe -n "CN=wcf_zhyongfeng" -ss My -sky exchange
Succeeded

C:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin>
```

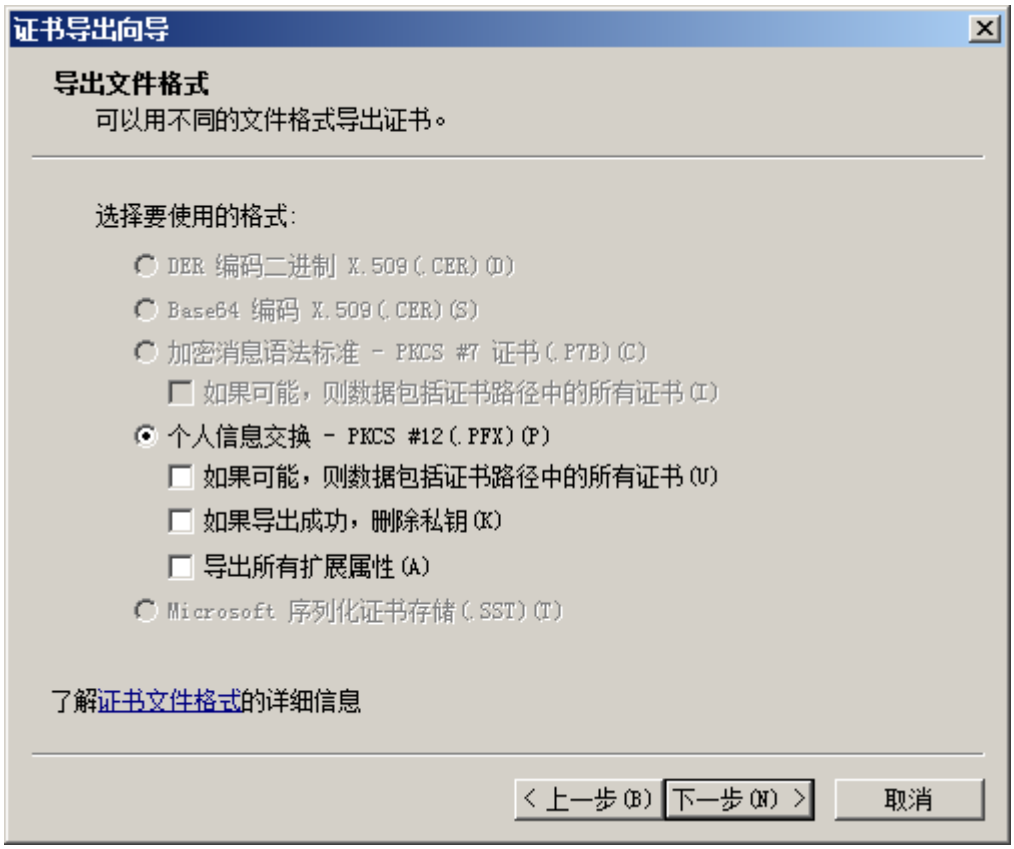


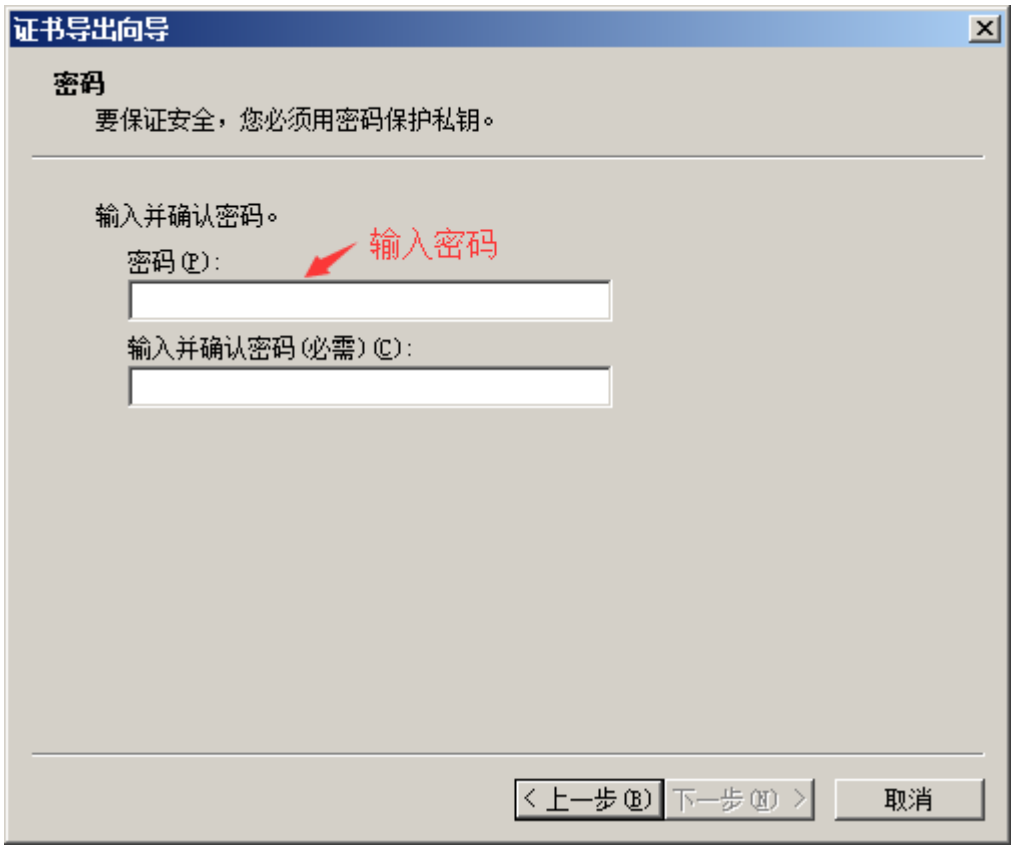
查看 Windows 证书的生成，开始->运行，输入：

certmgr.msc

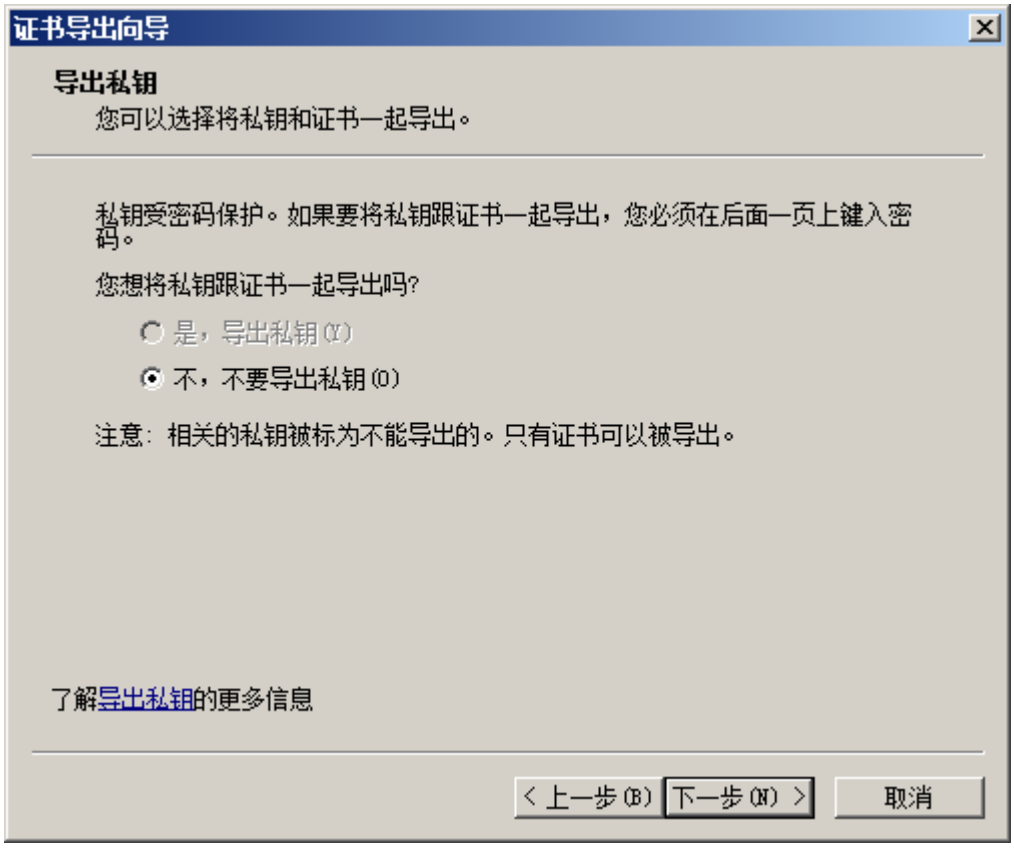
Windows 导出 wcf_zhyongfeng.pfx 服务端证书：

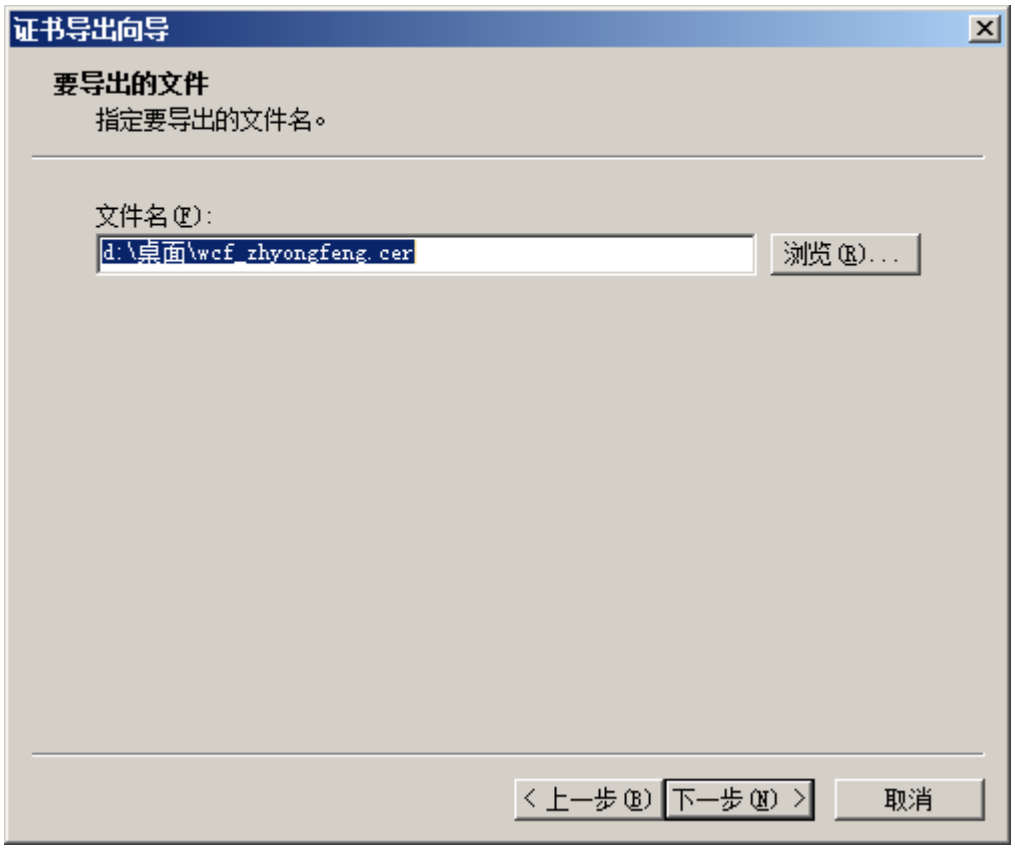






Windows 导出 wcf_zhyongfeng.cer 客户端证书：





5 编写 WCF 服务、客户端程序

- WCF 服务程序

Program.cs

```
using Service;
using System;
using System.ServiceModel;

namespace HighlyConcurrentHosting
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ServiceHost host = new ServiceHost(typeof(OutputSomething)))
            {
```

```
                host.Opened += delegate
                {
                    Console.WriteLine(host.Description.Endpoints[0].Address.Uri + "已经启动，按任意键终止服务！");
                };

                host.Open();
                Console.Read();
            }
        }
    }

    /// <summary>
    /// 证书验证帐户名，密码
    /// </summary>
    public class UserNamePasswordValidator : System.IdentityModel.Selectors.UserNamePasswordValidator
    {
        public override void Validate(string userName, string password)
        {
            if (userName != "zhyongfeng" || password != "123456")
            {
                throw new System.IdentityModel.Tokens.SecurityTokenException("Unknown Username or Password");
            }
        }
    }
}
```

服务端配置文件：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="metadataBehavior">
          <serviceMetadata httpGetEnabled="true"/>
          <!-- 要接收故障异常详细信息以进行调试，请将以下值设置为 true。在部署前设置为 false 以避免泄漏异常信息 -->
          <serviceDebug includeExceptionDetailInFaults="true"/>

          <serviceCredentials>
            <!--指定一个 X.509 证书，用户对认证中的用户名密码加密解密-->
            <!--C:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin，使用 makecert -r -pe -n "CN=wcf_zhyongfeng" -ss My -sky exchange-->
```

```
                <serviceCertificate                                findValue="wcf_zhyongfeng"
x509FindType="FindBySubjectName" storeLocation="CurrentUser" storeName="My"/>
                <clientCertificate>
                  <!--自定义对客户端进行证书认证方式 这里为 None-->
                  <authentication certificateValidationMode="None"></authentication>
                </clientCertificate>

                <!--自定义用户名和密码验证的设置-->
                <userNameAuthentication          userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="HighlyConcurrentHosting.UserNamePasswordValidato
r,HighlyConcurrentHosting" />
              </serviceCredentials>
            </behavior>
          </serviceBehaviors>
        </behaviors>

        <bindings>
          <basicHttpBinding>
            <!--这个是需要输入用户名密码的-->
            <binding name="YesCertificate">
              <security mode="TransportCredentialOnly">
                <transport clientCredentialType="Basic"></transport>
                <message clientCredentialType="UserName"/>
              </security>
            </binding>
          </basicHttpBinding>
    </bindings>

        <services>
          <service name="Service.OutputSomething" behaviorConfiguration="metadataBehavior" >
            <host>
              <baseAddresses>
                <add baseAddress="http://127.0.0.1:5600/hello"/>
              </baseAddresses>
            </host>
            <endpoint       binding="basicHttpBinding"       bindingConfiguration="YesCertificate"
contract="Service.Interface.IOutputSomething"/>
            <endpoint       binding="basicHttpBinding"       bindingConfiguration="YesCertificate"
contract="Service.Interface.IOutputSomethingCertificate" />
          </service>
        </services>
    </system.serviceModel>

    <startup>
```

```
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
</configuration>
```

- 客户端程序

Program.cs

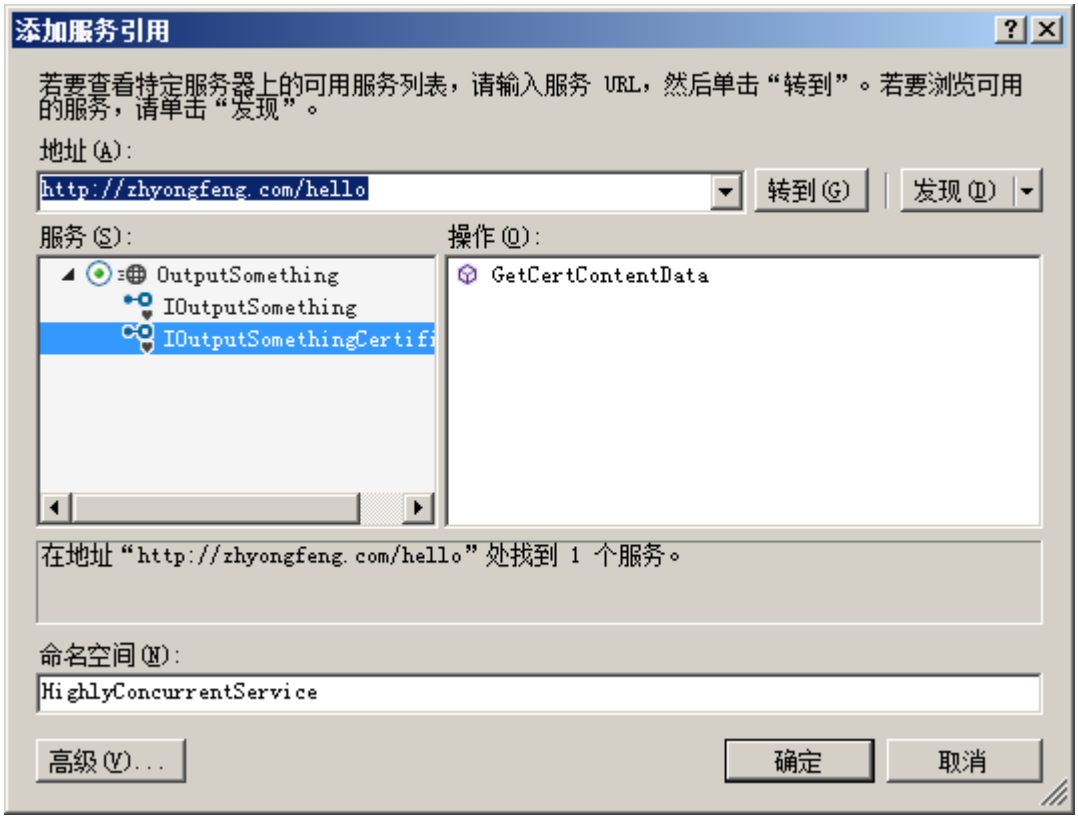
```
using HighlyConcurrentClient.HighlyConcurrentService;
using System;
using System.Net;

namespace HighlyConcurrentClient
{
    class Program
    {
        static void Main(string[] args)
        {

            string AddressIP = string.Empty;
            foreach (IPAddress _IPAddress in Dns.GetHostEntry(Dns.GetHostName()).AddressList)
            {
                if (_IPAddress.AddressFamily.ToString() == "InterNetwork")
                {
                    AddressIP = _IPAddress.ToString();
                }
            }
            Console.WriteLine(string.Format("本机 IP 是：{0}", AddressIP));
            using (OutputSomethingCertificateClient proxy = new OutputSomethingCertificateClient())
            {

                proxy.ClientCredentials.UserName.UserName = "zhyongfeng";
                proxy.ClientCredentials.UserName.Password = "123456";
                for (int i = 0; i < 20; i++)
                {
                    Console.WriteLine(proxy.GetCertContentData(i));
                }
            }
            Console.Read();
        }
    }
}
```

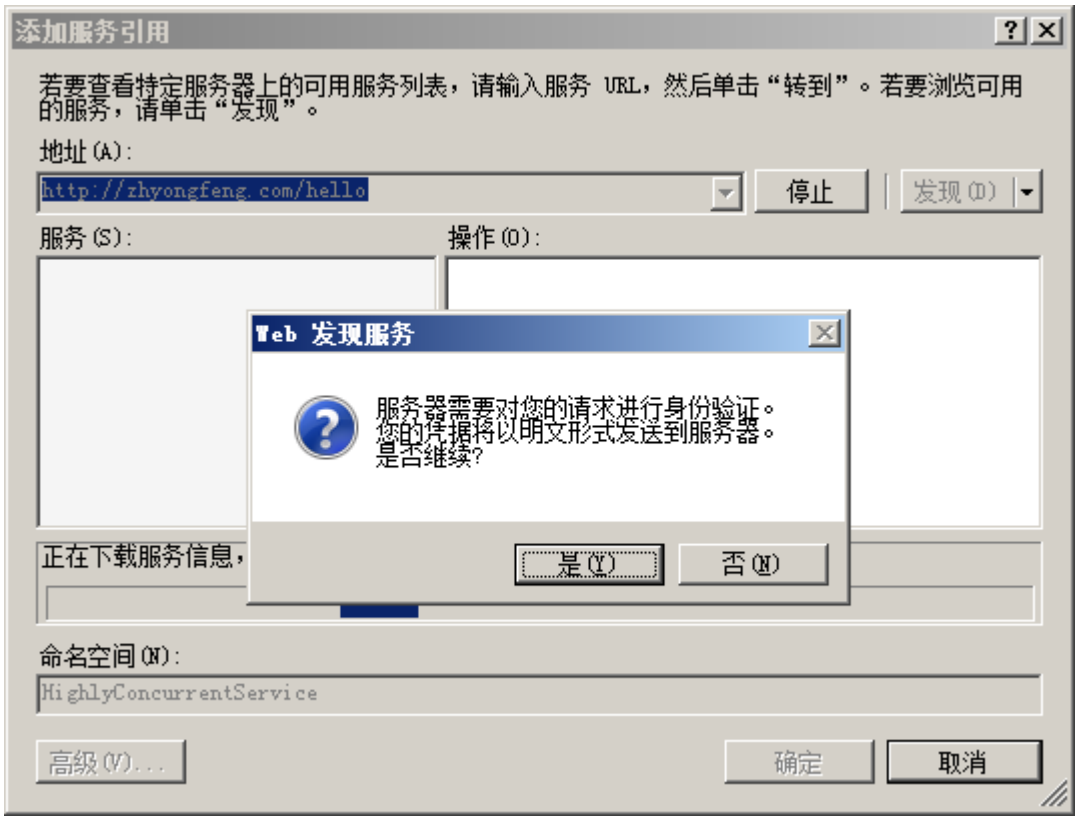
客户端配置文件：

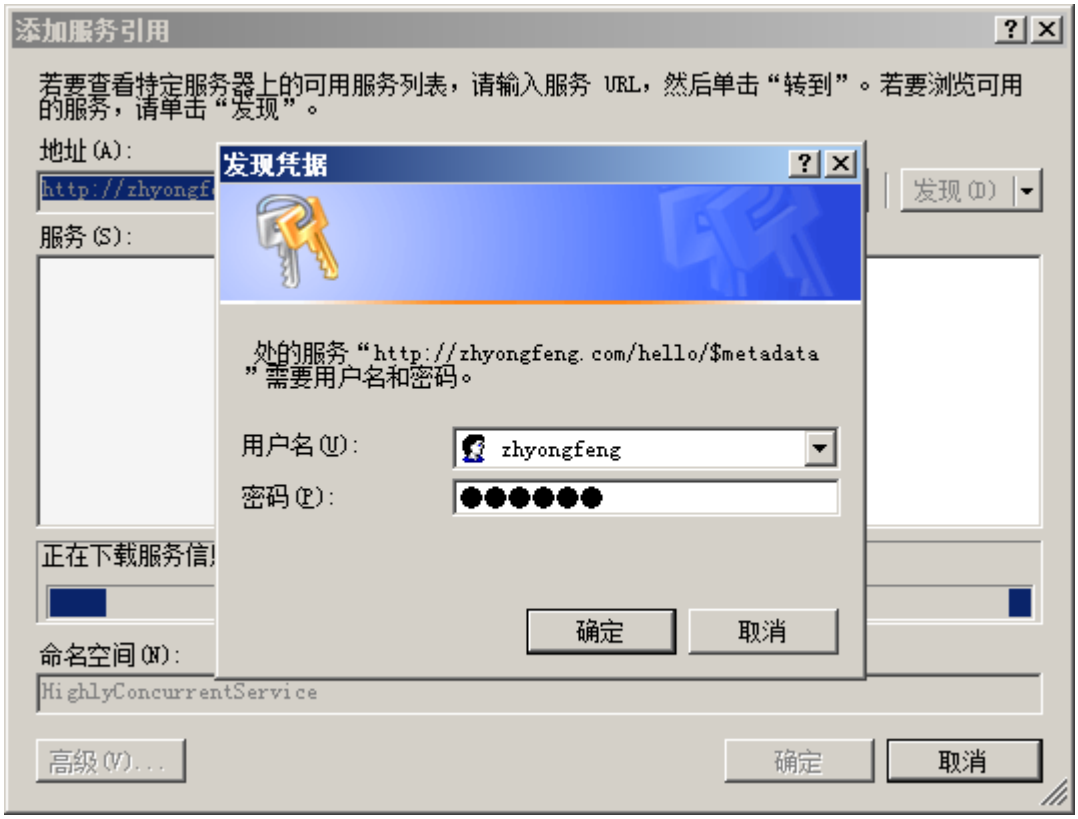
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
```

```
  <configSections>
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicHttpBinding_IOutputSomething">
          <security mode="TransportCredentialOnly">
            <transport clientCredentialType="Basic" />
          </security>
        </binding>
        <binding name="BasicHttpBinding_IOutputSomethingCertificate">
          <security mode="TransportCredentialOnly">
            <transport clientCredentialType="Basic" />
          </security>
        </binding>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://zhyongfeng.com/hello" binding="basicHttpBinding"
        bindingConfiguration="BasicHttpBinding_IOutputSomething"
contract="HighlyConcurrentService.IOutputSomething"
        name="BasicHttpBinding_IOutputSomething" />
      <endpoint address="http://zhyongfeng.com/hello" binding="basicHttpBinding"
        bindingConfiguration="BasicHttpBinding_IOutputSomethingCertificate"
        contract="HighlyConcurrentService.IOutputSomethingCertificate"
        name="BasicHttpBinding_IOutputSomethingCertificate" />
    </client>
  </system.serviceModel>
</configuration>
```

客户端添加引用时，会产生

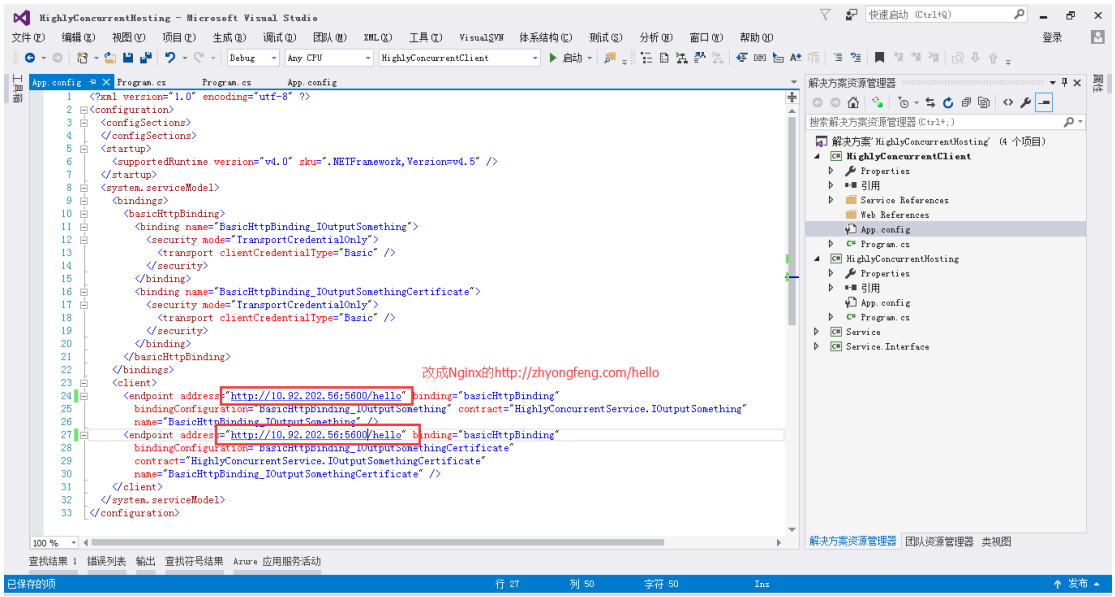






客户端添加服务引用后，Address 可能是某一台 PC 机的 IP 地址（例如：address="http://10.92.202.56:5600/hello"）这是需要修改为以下 Nginx 的地址 address="http://zhyongfeng.com/hello".

即如图所示：

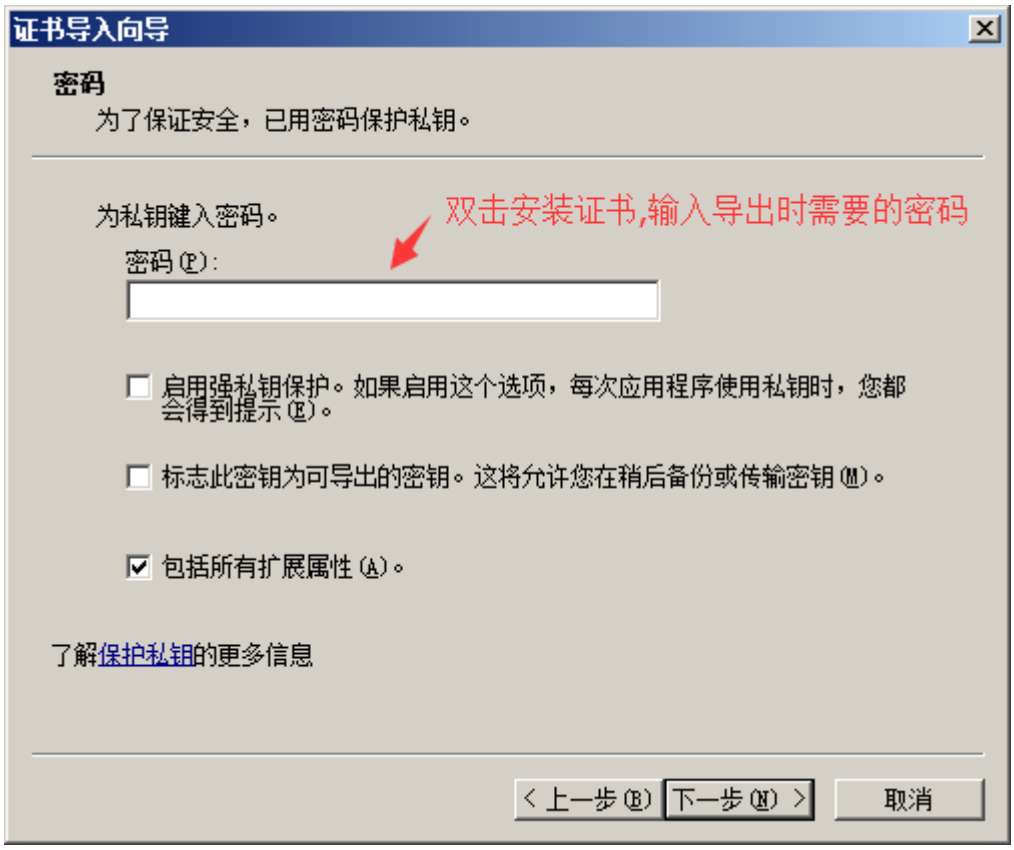


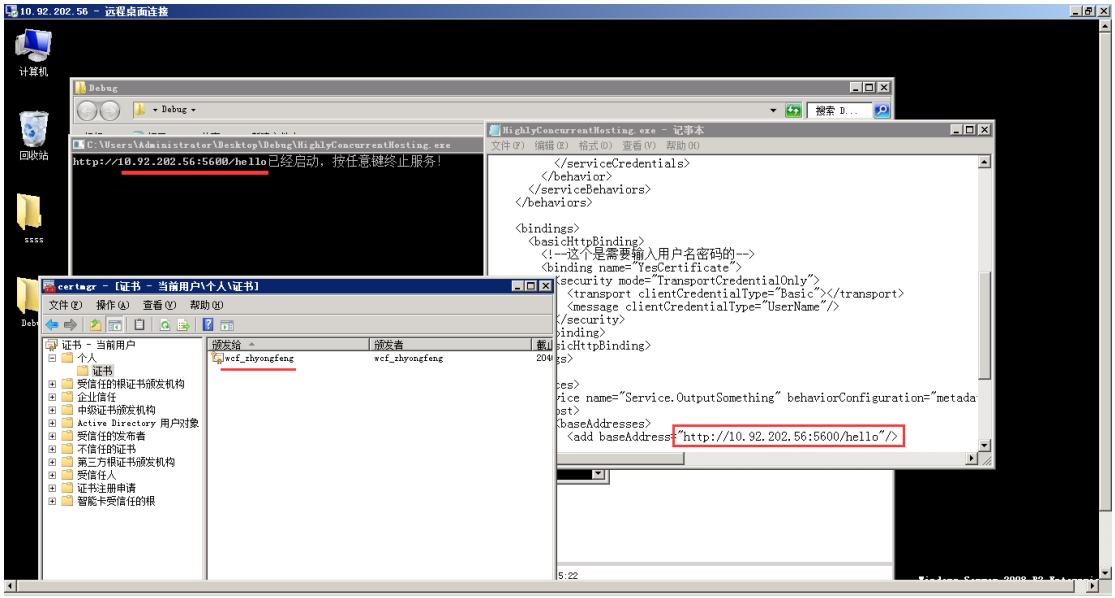
6 URL 保留项

详见：http://www.cnblogs.com/yongfeng/p/7851039.html

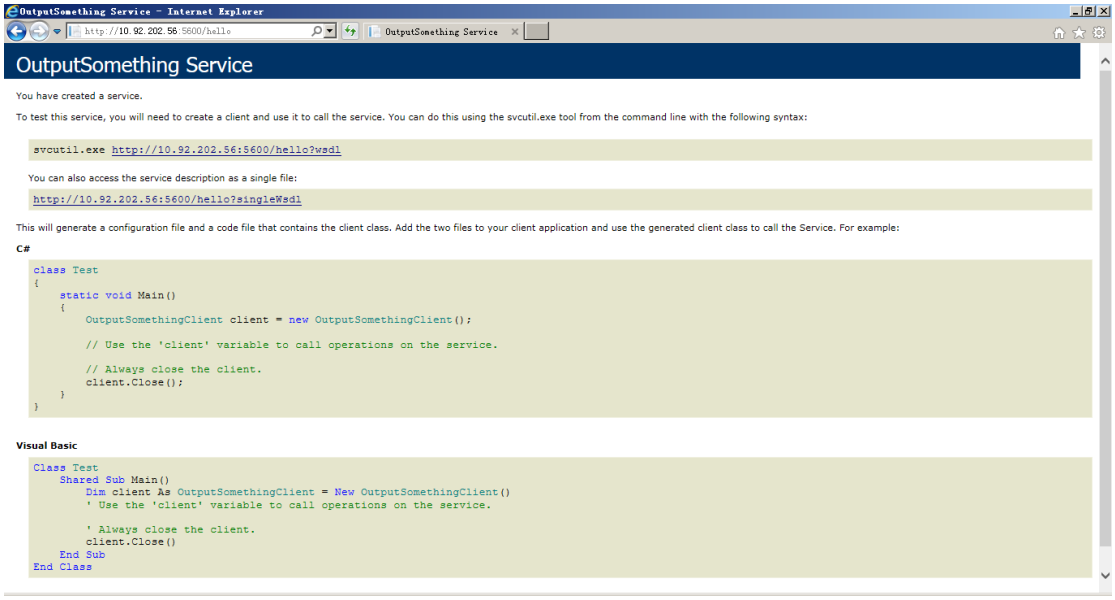
7 部署 WCF 服务程序到局域网内 3 台 PC 机

远程进行部署 WCF 服务程序时，需要双击安装服务器 wcf_zhyongfeng.pfx 证书、修改 config 三台机的配置文件：10.92.202.56:5600、10.92.202.57:5700、10.92.202.58:5800 然后启动远程计算机的 WCF 服务程序，运行效果如下：





本机 IE 上访问 WCF 服务端的运行效果：



8 Nginx 集群配置搭建

通过自主义域名 zhyongfeng.com：80 端口进行负载均衡集群访问，则访问 C:\Windows\System32\drivers\etc\hosts，添加下列“本机 IP 自定义的域名”：

```
10.93.85.66    zhyongfeng.com
```

针对 WCF 部署的多台 PC 机配置（设置了 proxy_connect_timeout 为 10s，如果其中一台机 down 掉了，可以转发到另一台机器）如下：

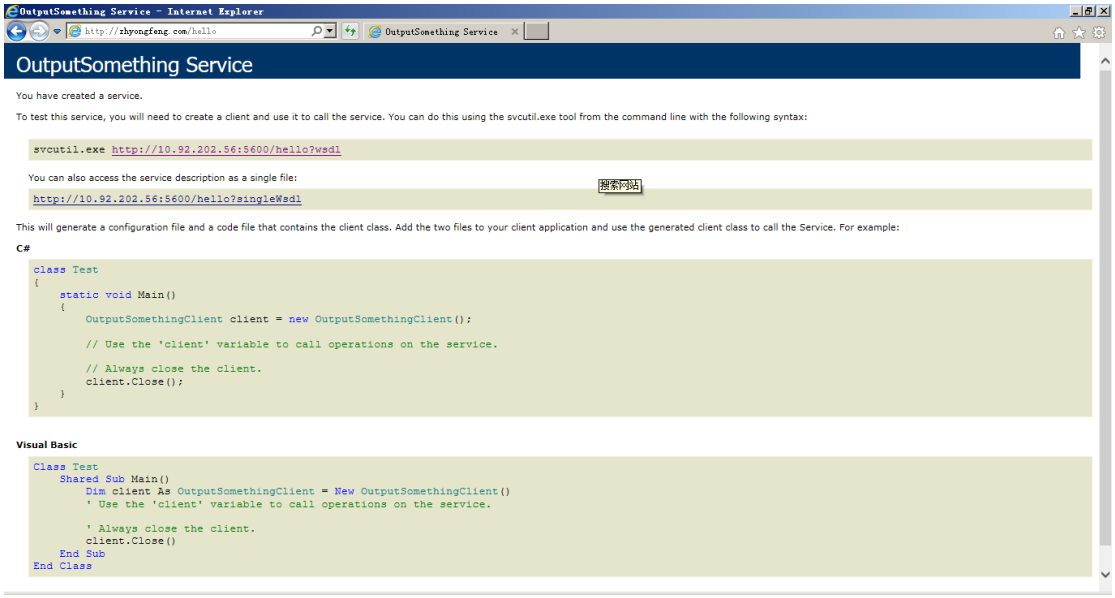
```
worker_processes    1;
events {
    worker_connections    1024;
}
http {
    include            mime.types;
    default_type    application/octet-stream;
    sendfile            on;
    keepalive_timeout    65;

    upstream zhyongfeng.com {
        server    10.92.202.56:5600;
        server    10.92.202.57:5700;
        server    10.92.202.58:5800;
    }
    server {
        listen            80;
        server_name    zhyongfeng.com;
        location / {
            proxy_pass    http://zhyongfeng.com;
```

```
                proxy_connect_timeout       10s;
            }
        }
}
```

运行 CMD：

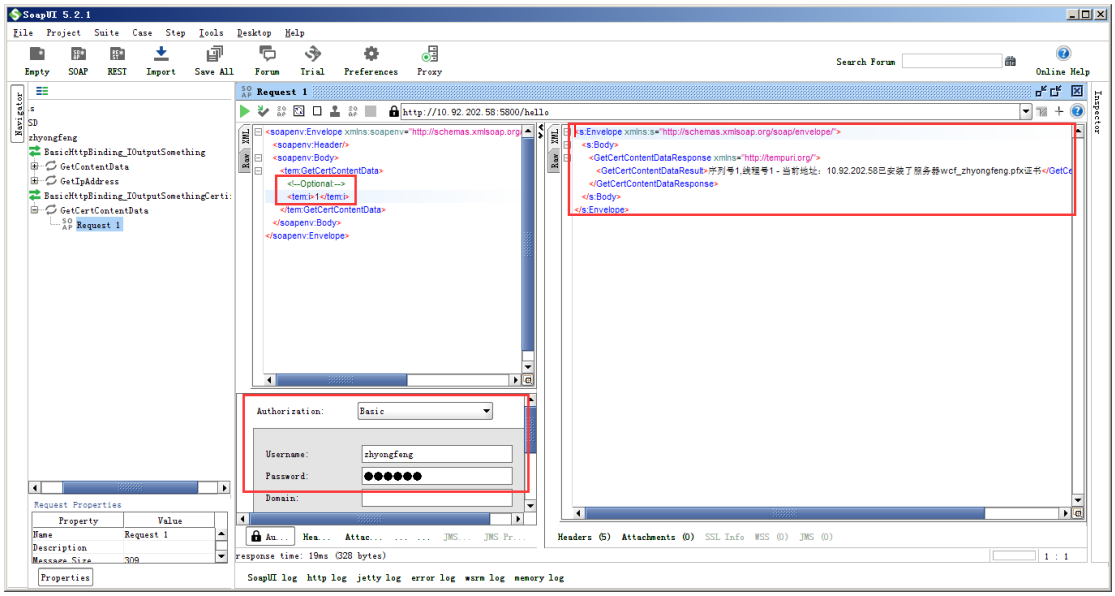
```
D:\DTLDownLoads\nginx-1.10.2>start nginx

D:\DTLDownLoads\nginx-1.10.2>nginx -s reload
```

访问 WCF 服务端：http://zhyongfeng.com/hello，运行结果：

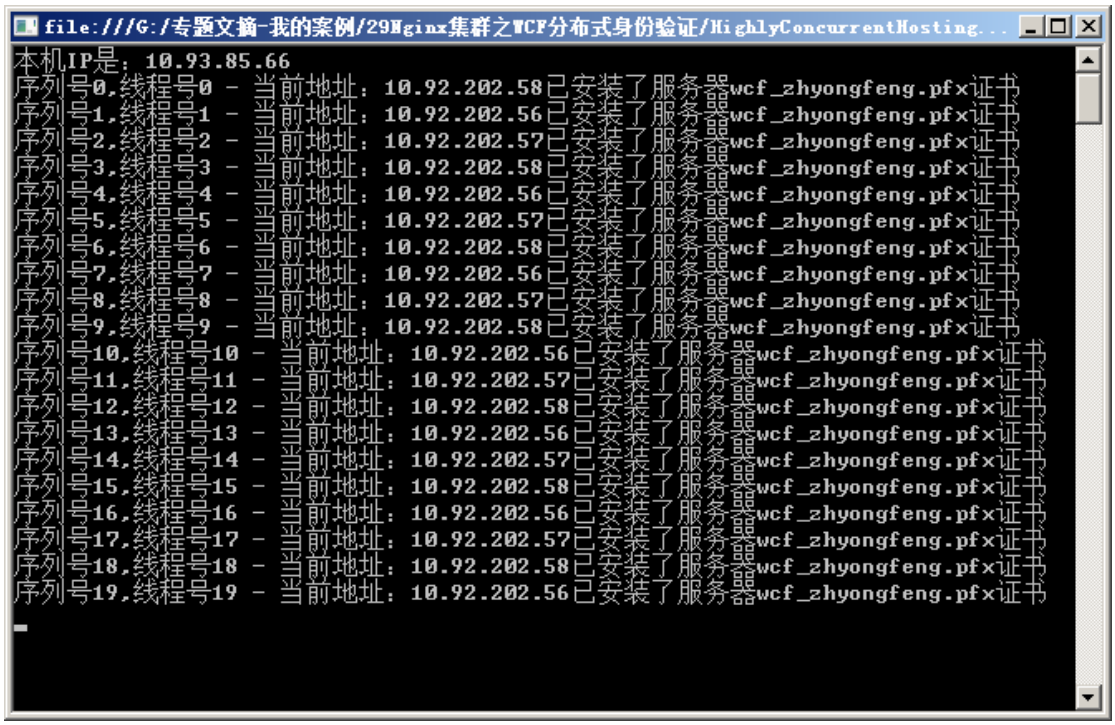


9 SoapUI 和 WCF 客户端程序的运行结果

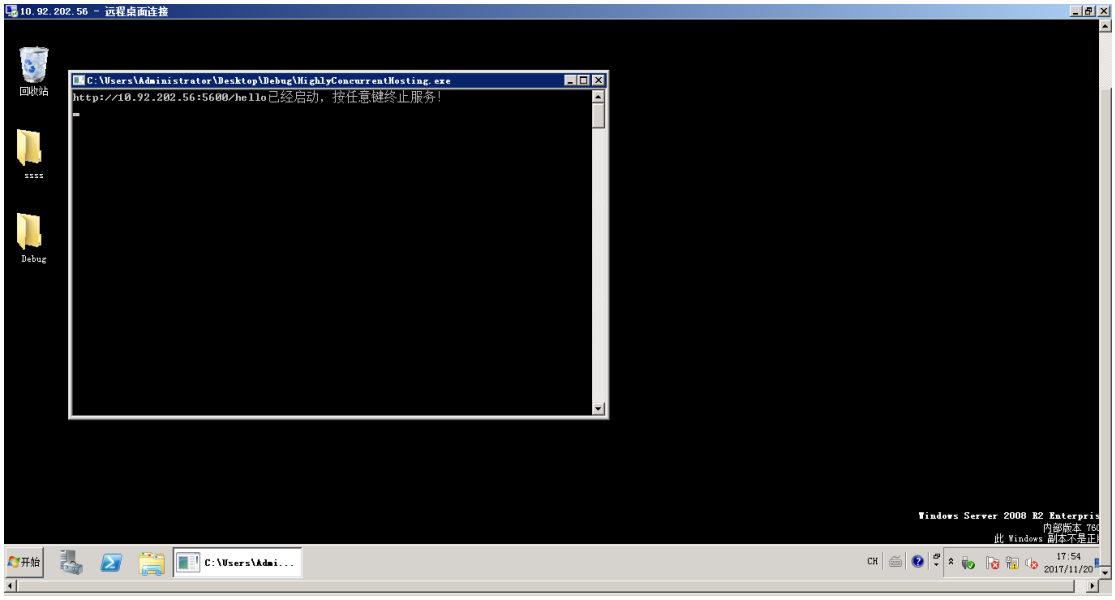
Soap 协议，可以使用 SoapUI 测试并添加 WCF 的 wsdl：http://zhyongfeng.com/hello?wsdl，运行效果如下：



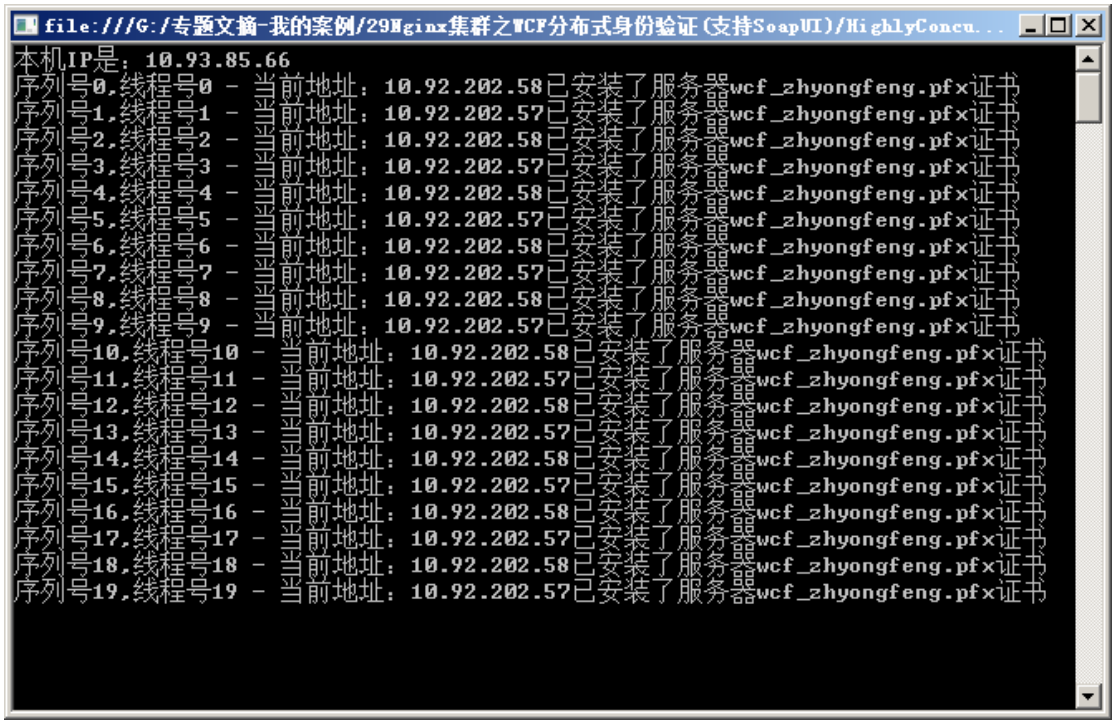
启动 WCF 客户端程序，运行效果图如下：



远程桌面关掉其中一台 10.92.202.56:5600 的 PC 机：



重新启动 WCF 客户端程序，因为 Nginx 配置文件设置了 proxy_connect_timeout 为 10s，则关闭的 PC 机 10.92.202.56:5600 在 10s 后会将它的消息转发给 10.92.202.57:5700，继续由其它 2 台 PC 机执行：



10 总结

通过使用 BasicHttpBinding，除了能让 WCF 客户端访问之外，还增加了 WSDL 的访问方式。Nginx 集群让 WCF 客户端具备用户名密码验证的同时，达到负载均衡分布式处理的效果。

源代码下载：
http://download.csdn.net/download/ruby_matlab/10126187

PDF 下载：
Nginx 集群之 WCF 分布式身份验证(支持 soap).pdf