目录

1 大概思路

- Nginx 集群之基于 Redis 的 WebApi 身份验证
- Redis 数据库
- Visualbox 虚拟机 ubuntu 下的 redis 部署
- 编写.NET WebApi 的 OnAuthorization 身份验证
- 编写.NET WebApi 的 ActionFilterAttribute 令牌验证
- 编写.NET WebApi 的服务端
- 编写.NET WebApi 的客户端
- 部署 WebApi 到本机
- Nginx 集群配置搭建
- 运行结果
- 总结

2 Nginx 集群之基于 Redis 的 WebApi 身份验证

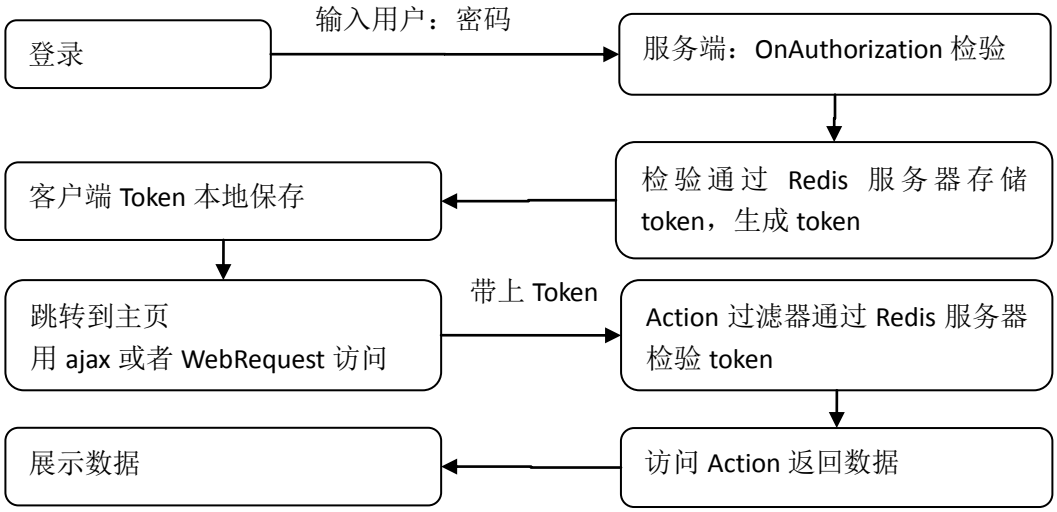
Nginx 在集群上使用 Redis 数据库进行身份验证，达到了支持集群、分布式。在此基础上能够实现单点登录、时效性的访问，结合 WebApi 最大限度地发挥了后台身份验证的管理。

基于 Redis 的原因有几个：第一点是 Redis 是基于内存的数据库访问起来比较快，能够设置数据库存储的时效性；第二点，Redis 数据库的语法比较简单、轻巧，在 Linux、Windows 服务器均可以一键安装完成；第三点，Redis 支持数据的备份，即 master-slave 模式的数据备份。

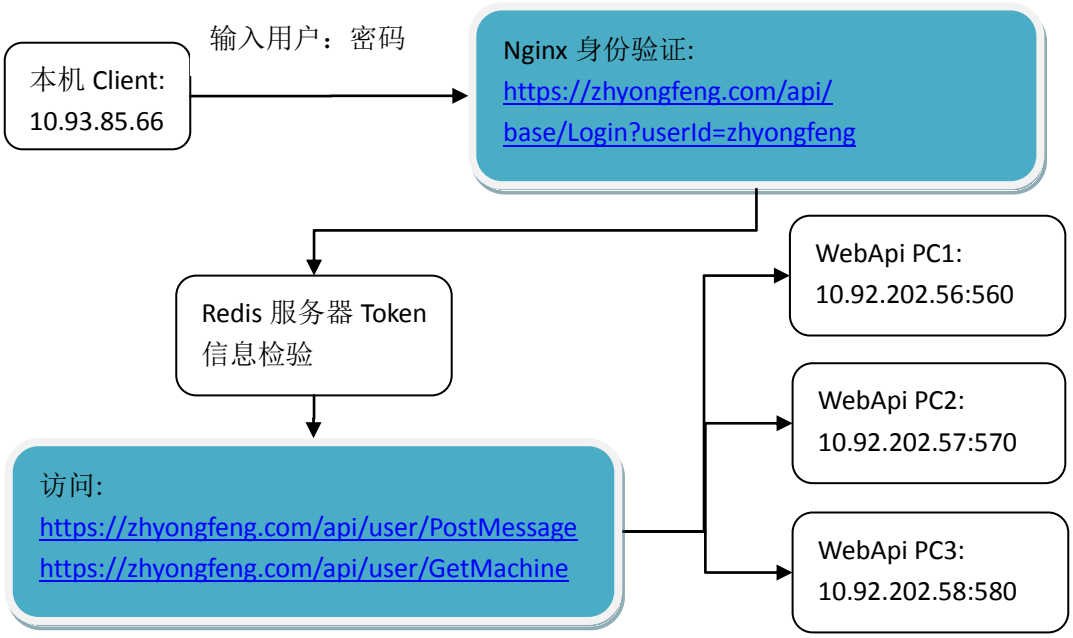
以下是本文讲述的主要结构图：

客户端输入用户名密码服务器，通过了用户名密码验证，其中一台 WebApi 服务器生成一个 Token 并返回 https 的响应。客户端收到 Token 保存在本地，带上 token 发出 ajax 请求、WebRequest 请求，经过 Action 过滤器的检验，访问 Action 并返回数据。

Token 令牌身份验证机制：



Nginx 集群之基于 Redis 的 WebApi 身份验证，如下图所示：



3 Redis 数据库

Redis 是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库，并提供多种语言的 API。它通常被称为数据结构服务器，因为值（value）可以是 字符串(String), 哈希(Map), 列表(list), 集合(sets) 和 有序集合(sorted sets)等类型。具体可以在以下网址进行学习：

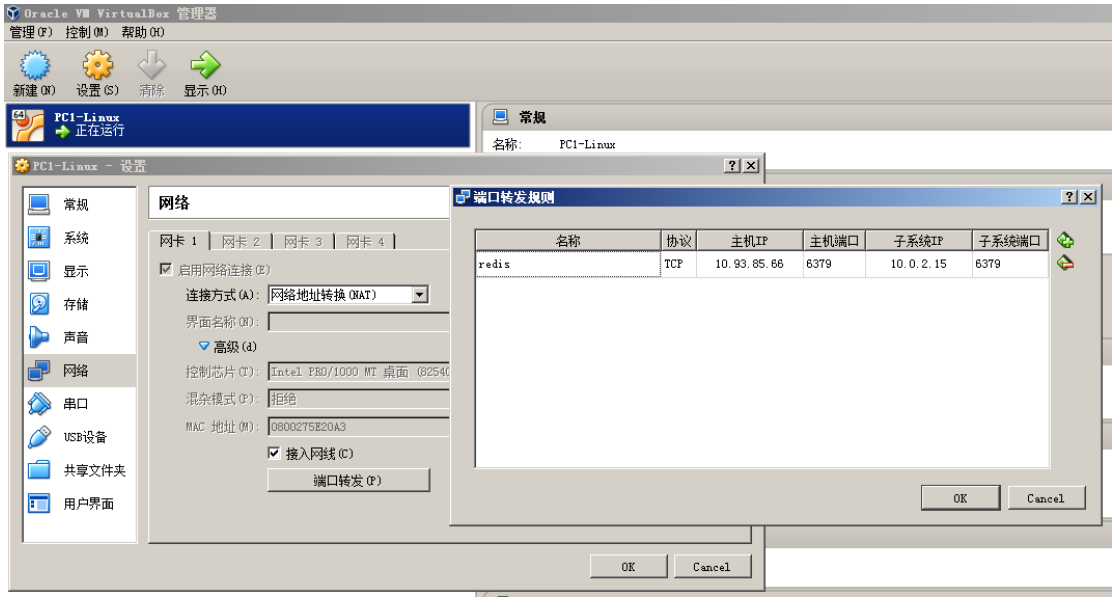
http://www.runoob.com/redis/redis-tutorial.html

4 Visualbox 虚拟机 ubuntu 下的 redis 部署

（已安装 Oracle VM VirtualBox、ubuntu-17.10-desktop-amd64.iso）
Redis 安装可参考：http://www.runoob.com/redis/redis-install.html

- 虚拟机

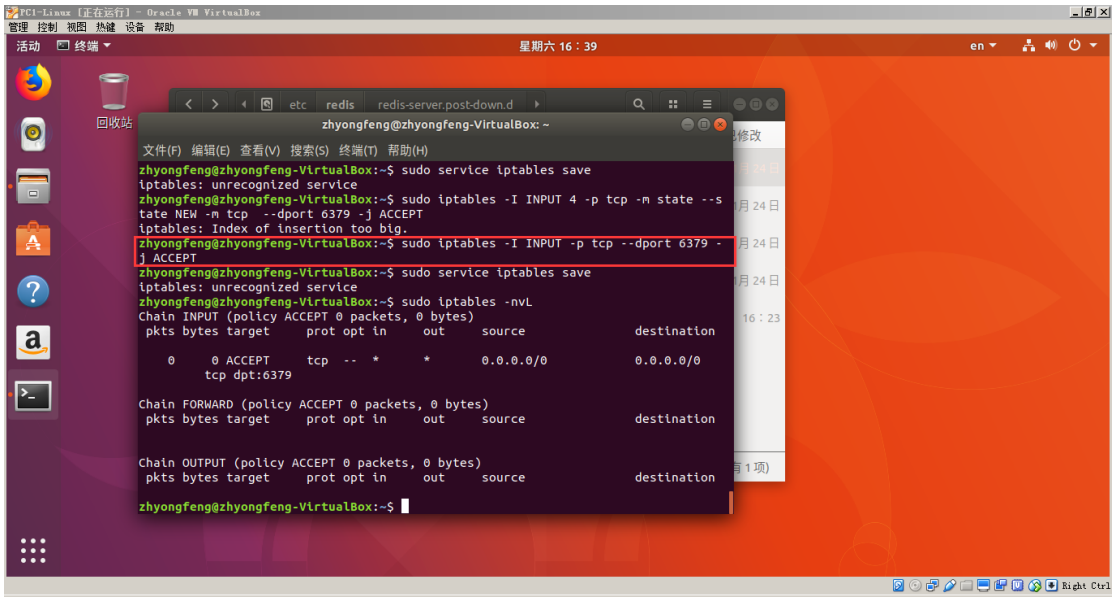
因网络的限制，本文采取的是“网络地址转换（NAT）”方式，进行 redis 数据库连接访问。有条件的可以采用“桥接网上”的方式，便可以使多台 PC 机在同一 redis 数据库进行访问，达到集群的效果。



Linux 的 ubuntu 打开 6379 端口：

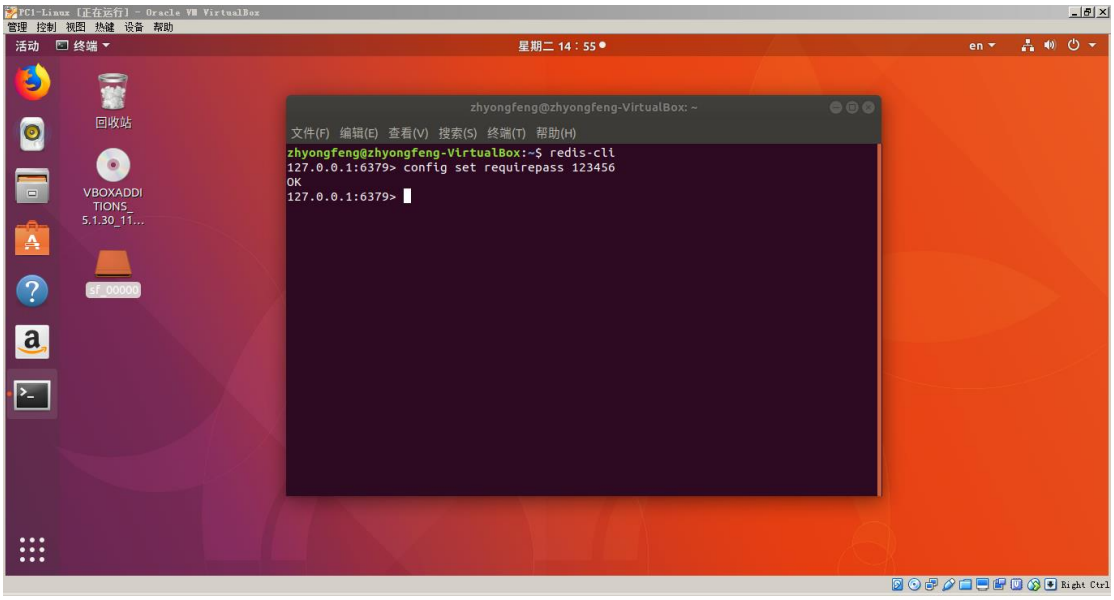
```
sudo iptables -I INPUT -p tcp --dport 6379 -j ACCEPT
```

具体如下所示：



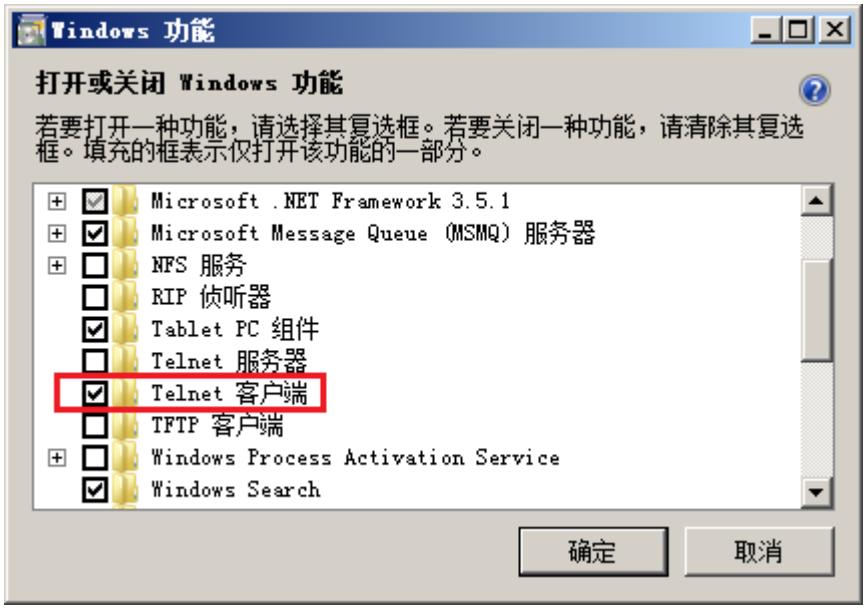
登录 Redis 服务器，设置密码：

```
redis-cli
config set requirepass 123456
```



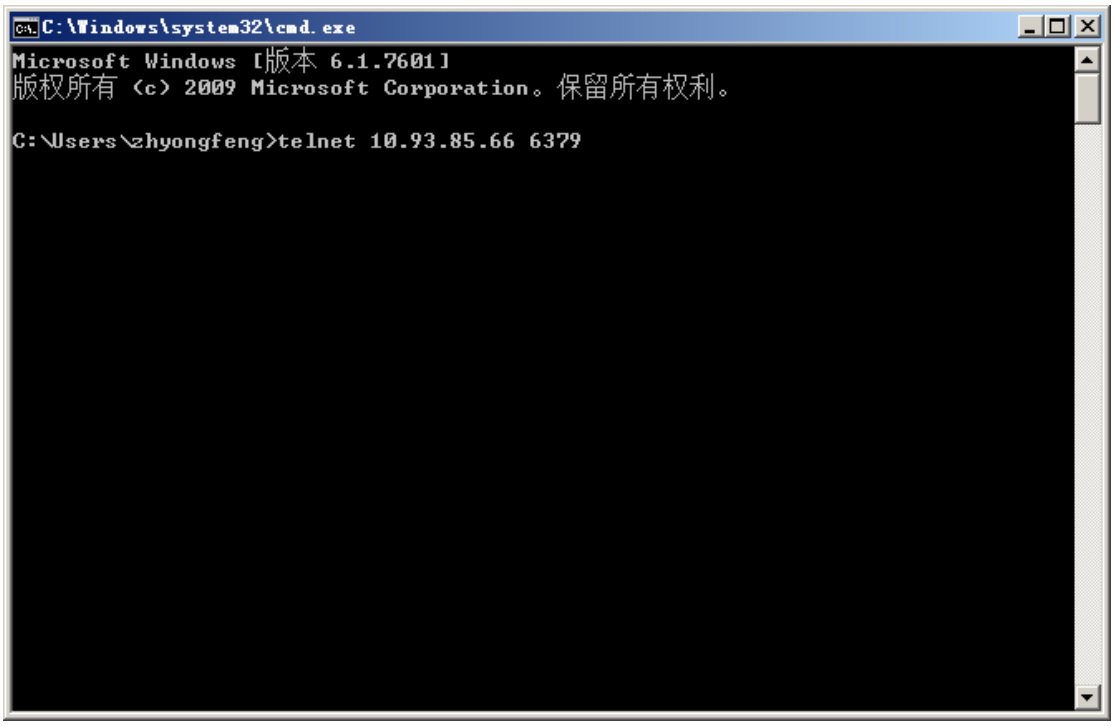
- 主机

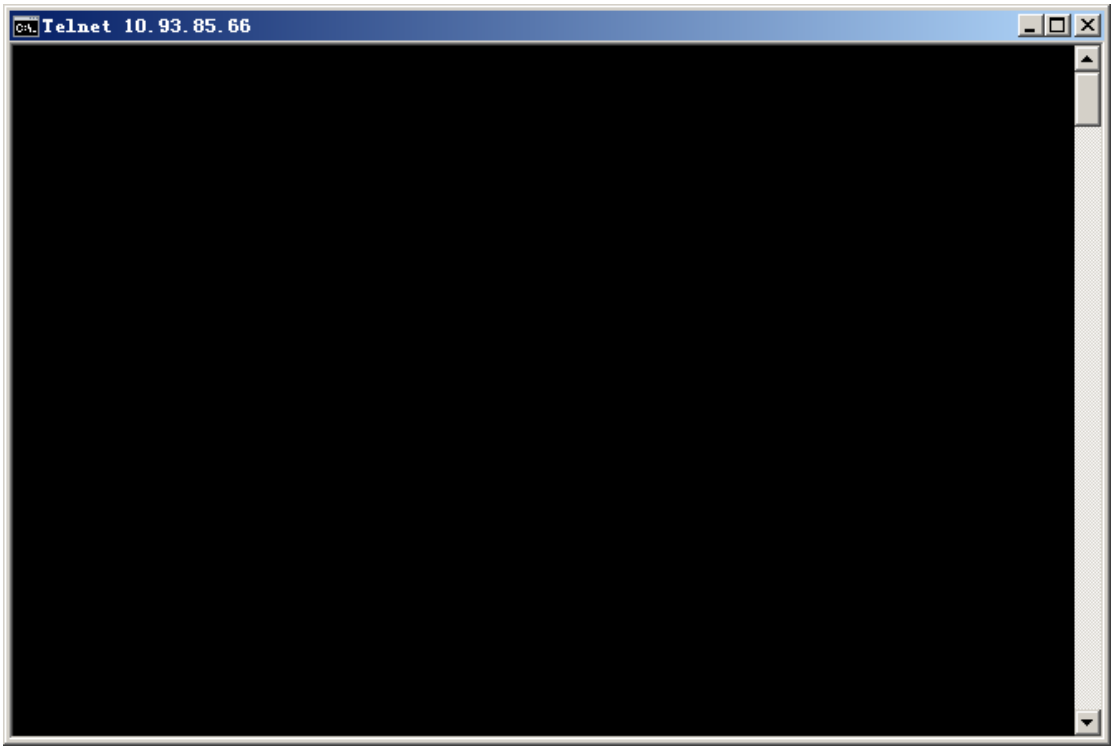
Windows 安装 Telnet 客户端，进行测试连接成功与否



运行 CMD，输入命令行如下：

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

C:\Users\zhyongfeng>telnet 10.93.85.66 6379
```





测试成功

5 编写.NET WebApi 的 OnAuthorization 身份验证

CustomAuthorizeAttribute.cs

```
using System.Web.Http;
using System.Web.Http.Controllers;

namespace SSLWebApi.Controllers
{
    public class CustomAuthorizeAttribute : AuthorizeAttribute
    {
        public override void OnAuthorization(HttpActionContext actionContext)
        {
            //判断用户是否登录
            if (actionContext.Request.Headers.Authorization != null)
            {
                string userInfo = System.Text.Encoding.Default.GetString(System.Convert.FromBase64String(actionContext.Request.Headers.Authorization.Parameter));
                //用户验证逻辑
                if (string.Equals(userInfo, string.Format("{0}:{1}", "zhyongfeng", "123456")))
                {
                    IsAuthorized(actionContext);
                }
                else
                {
                    HandleUnauthorizedRequest(actionContext);
                }
            }
            else
            {
                HandleUnauthorizedRequest(actionContext);
            }
        }
        protected override void HandleUnauthorizedRequest(System.Web.Http.Controllers.HttpActionContext actionContext)
        {
            var challengeMessage = new System.Net.Http.HttpResponseMessage(System.Net.HttpStatusCode.Unauthorized);
            challengeMessage.Headers.Add("WWW-Authenticate", "Basic");
            throw new System.Web.Http.HttpResponseException(challengeMessage);
```

```
            }
        }
}
```

生成 Token

BaseController.cs

```
using ServiceStack.Redis;
using SSLWebApi.Models;
using System;
using System.Web;
using System.Web.Http;

namespace SSLWebApi.Controllers
{
    /// <summary>
    /// BaseController 继承 BaseController 则需要身份验证
    /// </summary>
    [CustomAuthorize]
    [RoutePrefix("api/Base")]
    public class BaseController : ApiController
    {
        [HttpGet]
        [Route("Login")]
        public string Login(string userId)
        {
            if (HttpRuntime.Cache.Get(userId) == null)
            {
                return CreateToken(userId);
            }
            else
            {
                HttpRuntime.Cache.Remove(userId);
                return CreateToken(userId);
            }
        }

        /// <summary>
        ///  生成 token
        /// </summary>
        /// <param name="userId"></param>
        /// <returns></returns>
        private string CreateToken(string userId)
        {
            Token token = new Token();
            token.UserId = userId;
```

```
                token.SignToken = Guid.NewGuid();
                token.Seconds = 12;
                token.ExpireTime = DateTime.Now.AddSeconds(token.Seconds);

                //redis 使用 sudo iptables -A INPUT -p tcp --dport 6379 -j ACCEPT 打开 6379 的端
口
                //visualbox 虚拟机使用“设置->网络->高级->端口转发”创建商品规则
                //连接远程 visualbox 虚拟机 ubtuntu 的 redis，形成一个分布式的登录验证
                string remoteIp = "10.93.85.66";
                int remotePort = 6379;
                string remoteDbPassword = "123456";
                //RedisClient(地址,端口，密码，0)
                using (var client = new RedisClient(remoteIp, remotePort, remoteDbPassword, 0))
                {
                    string strToken = token.SignToken.ToString();
                    if (client.Exists(userId) > 0)
                        client.Del(userId);
                    if (client.Add(userId, strToken))
                    {
                        //设置 key 的过期时间为 12s
                        client.Expire(userId, 12);
                        return token.SignToken.ToString();
                    }
                    else
                        return string.Format("远程虚拟机{0}的 redis 创建 token 失败",
remoteIp);
                }
            }
        }
}
```

6 编写.NET WebApi 的 ActionFilterAttribute 令牌验证

WebApiSecurityFilter.cs

```
using ServiceStack.Redis;
using SSLWebApi.Models;
using System;
using System.Linq;
using System.Net.Http;
using System.Web;
```

```
using System.Web.Http.Controllers;
using System.Web.Http.Filters;

namespace SSLWebApi.Filter
{
    public class WebApiSecurityFilter : ActionFilterAttribute
    {
        public override void OnActionExecuting(HttpActionContext actionContext)
        {

            if (actionContext.ActionDescriptor.ActionName == "Login")
            {
                //登录成功则生成 token
                base.OnActionExecuting(actionContext);
                return;
            }
            else
            {
                //判断 token 令牌
                HttpRequestMessage request = actionContext.Request;
                string staffid = request.Headers.Contains("userid") ? HttpUtility.UrlDecode(request.Headers.GetValues("userid").FirstOrDefault()) : string.Empty;
                string timestamp = request.Headers.Contains("timestamp") ? HttpUtility.UrlDecode(request.Headers.GetValues("timestamp").FirstOrDefault()) : string.Empty;
                string nonce = request.Headers.Contains("nonce") ? HttpUtility.UrlDecode(request.Headers.GetValues("nonce").FirstOrDefault()) : string.Empty;
                string signature = request.Headers.Contains("signature") ? HttpUtility.UrlDecode(request.Headers.GetValues("signature").FirstOrDefault()) : string.Empty;

                if (String.IsNullOrEmpty(staffid) || String.IsNullOrEmpty(timestamp) || String.IsNullOrEmpty(nonce) || String.IsNullOrEmpty(signature))
                {
                    //令牌检验不通过
                    actionContext.Response = new HttpResponseMessage(System.Net.HttpStatusCode.Forbidden);
                    return;
                }
                else
                {
                    string remoteIp = "10.93.85.66";
                    int remotePort = 6379;
                    string remoteDbPassword = "123456";
                    using (var client = new RedisClient(remoteIp, remotePort, remoteDbPassword, 0))
```

```
                        {
                            //令牌检验是否存在这个用户
                            if (client.Exists(staffid) > 0)
                            {
                                string tempStr = System.Text.Encoding.UTF8.GetString(client.Get(staffid));
                                if (tempStr.Trim('"').Equals(signature))
                                {
                                    //时间转换成功、时间有效、token 值相等
                                    //令牌通过
                                    base.OnActionExecuting(actionContext);
                                    return;
                                }
                                else
                                {
                                    actionContext.Response = new HttpResponseMessage(System.Net.HttpStatusCode.Forbidden);
                                    return;
                                }
                            }
                            else
                            {
                                actionContext.Response = new HttpResponseMessage(System.Net.HttpStatusCode.Forbidden);
                                return;
                            }
                        }
                    }
                }
            }

    }
}
```

7 编写.NET WebApi 的服务端

UserController.cs

```
using System;
using System.Net;
using System.Web.Http;

namespace SSLWebApi.Controllers
{
```

```
    [RoutePrefix("api/User")]
    public class UserController : ApiController
    {
        /// <summary>
        ///  获取当前用户信息
        /// </summary>
        /// <param name="msg"></param>
        /// <returns></returns>
        [HttpPost]
        [Route("PostMessage")]
        public string PostMessage(dynamic obj)
        {
            return string.Format("当前输入的消息是:{0}", Convert.ToString(obj.msg));
        }

        [Route("GetMachine")]
        public string GetMachine()
        {
            string AddressIP = string.Empty;
            foreach (IPAddress _IPAddress in
Dns.GetHostEntry(Dns.GetHostName()).AddressList)
            {
                if (_IPAddress.AddressFamily.ToString() == "InterNetwork")
                {
                    AddressIP = _IPAddress.ToString();
                }
            }
            return string.Format("当前 WebApi 部署的 IP 是：{0}", AddressIP);
        }
    }
}
```

8 编写.NET WebApi 的客户端

WebApiHelper.cs

```
using Newtonsoft.Json;
using System;
using System.IO;
using System.Net;
using System.Text;

namespace SSLWebApiClient.Common
{
```

```
public class WebApiHelper
{
        /// <summary>
        /// Post 请求
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="url">url</param>
        /// <param name="data">数据</param>
        /// <param name="userid">帐户</param>
        /// <param name="signature">数字签名</param>
        /// <returns></returns>
        public static string Post(string url, string data, string userid, string signature)
        {
                return PostData(url, data, userid, signature);
        }

        /// <summary>
        /// Post 请求
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="url">url</param>
        /// <param name="data">数据</param>
        /// <param name="userid">帐户</param>
        /// <param name="signature">数字签名</param>
        /// <returns></returns>
        public static T Post<T>(string url, string data, string userid, string signature)
        {
                return JsonConvert.DeserializeObject<T>(Post(url, data, userid, signature));
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="webApi"></param>
        /// <param name="queryStr"></param>
        /// <param name="userid"></param>
        /// <param name="signature"></param>
        /// <returns></returns>
        public static string Get(string webApi, string queryStr, string userid, string signature)
        {
                return GetData(webApi, queryStr, userid, signature);
        }

        /// <summary>
```

```
        /// Get 请求
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="webApi"></param>
        /// <param name="query"></param>
        /// <param name="queryStr"></param>
        /// <param name="userid"></param>
        /// <param name="signature"></param>
        /// <returns></returns>
        public static T Get<T>(string webApi, string queryStr, string userid, string signature)
        {
            return JsonConvert.DeserializeObject<T>(GetData(webApi, queryStr, userid, signature));
        }

        /// <summary>
        /// 获取时间戳
        /// </summary>
        /// <returns></returns>
        private static string GetTimeStamp()
        {
            TimeSpan ts = DateTime.Now - new DateTime(1970, 1, 1, 0, 0, 0, 0);
            return ts.TotalSeconds.ToString();
        }


        /// <summary>
        /// 获取随机数
        /// </summary>
        /// <returns></returns>
        private static string GetRandom()
        {
            Random rd = new Random(DateTime.Now.Millisecond);
            int i = rd.Next(0, int.MaxValue);
            return i.ToString();
        }
        /// <summary>
        /// Post 请求
        /// </summary>
        /// <param name="url"></param>
        /// <param name="data"></param>
        /// <param name="userid">用户名称</param>
        /// <param name="signature">数字签名</param>
        /// <returns></returns>
```

```
        private static string PostData(string url, string data, string userid, string signature)
        {
            try
            {
                byte[] bytes = Encoding.UTF8.GetBytes(data);
                HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);

                string timeStamp = GetTimeStamp();
                string nonce = GetRandom();
                //加入头信息
                //当前请求用户
                request.Headers.Add("userid", userid);
                //发起请求时的时间戳（单位：秒）
                request.Headers.Add("timestamp", timeStamp);
                //发起请求时的时间戳（单位：秒）
                request.Headers.Add("nonce", nonce);
                //当前请求内容的数字签名
                request.Headers.Add("signature", signature);

                //写数据
                request.Method = "POST";
                request.ContentLength = bytes.Length;
                request.ContentType = "application/json";
                request.GetRequestStream().Write(bytes, 0, bytes.Length);
                //读数据
                request.Timeout = 300000;
                request.Headers.Set("Pragma", "no-cache");
                HttpWebResponse response = (HttpWebResponse)request.GetResponse();
                Stream streamReceive = response.GetResponseStream();
                StreamReader streamReader = new StreamReader(streamReceive,
Encoding.UTF8);
                string strResult = streamReader.ReadToEnd();

                //关闭流
                //reqstream.Close();
                streamReader.Close();
                streamReceive.Close();
                request.Abort();
                response.Close();
                return strResult;
            }
            catch (Exception ex)
            {
                return ex.Message;
```

```
            }
        }

        /// <summary>
        /// Get 请求
        /// </summary>
        /// <param name="webApi"></param>
        /// <param name="queryStr"></param>
        /// <param name="userid"></param>
        /// <param name="signature"></param>
        /// <returns></returns>
        private static string GetData(string webApi, string queryStr, string userid, string signature)
        {
            try
            {
                HttpWebRequest request = (HttpWebRequest)WebRequest.Create(webApi + "?" + queryStr);
                string timeStamp = GetTimeStamp();
                string nonce = GetRandom();
                //加入头信息
                //当前请求用户
                request.Headers.Add("userid", userid);
                //发起请求时的时间戳（单位：秒）
                request.Headers.Add("timestamp", timeStamp);
                //发起请求时的时间戳（单位：秒）
                request.Headers.Add("nonce", nonce);
                //当前请求内容的数字签名
                request.Headers.Add("signature", signature);

                request.Method = "GET";
                request.ContentType = "application/json";
                request.Timeout = 90000;
                request.Headers.Set("Pragma", "no-cache");
                HttpWebResponse response = (HttpWebResponse)request.GetResponse();
                Stream streamReceive = response.GetResponseStream();
                StreamReader streamReader = new StreamReader(streamReceive, Encoding.UTF8);
                string strResult = streamReader.ReadToEnd();

                streamReader.Close();
                streamReceive.Close();
                request.Abort();
                response.Close();
```

```
                return strResult;
            }
            catch (Exception ex)
            {
                return ex.Message;
            }
        }

    }
}
```

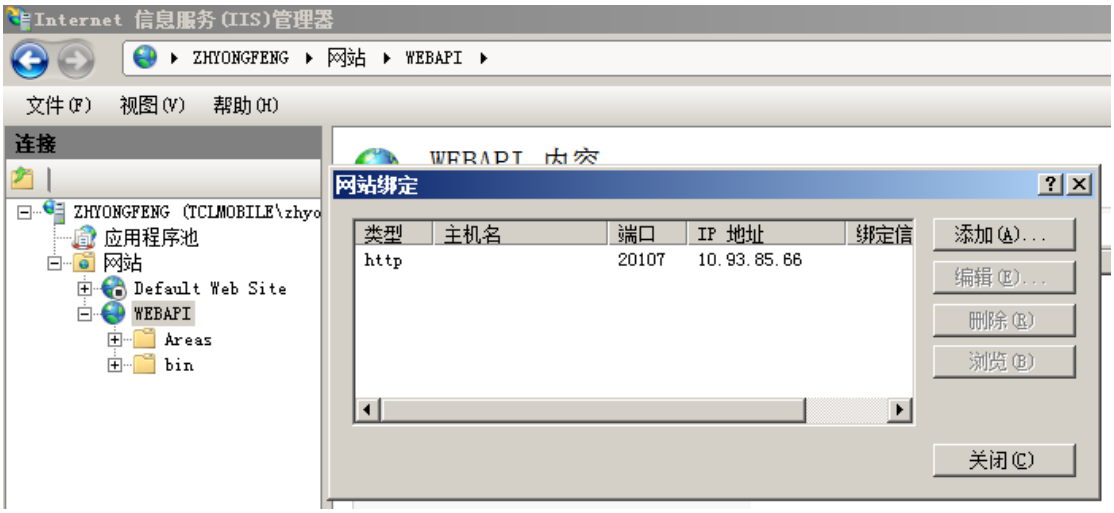
Program.cs

```
using System;
using System.IO;
using System.Net;
using System.Text;
using Newtonsoft.Json;
namespace SSLWebApiClient
{
    class Program
    {
        static void Main(string[] args)
        {
            string basicUrl = "http://zhyongfeng.com";
            string html = string.Empty;
            for (int i = 0; i < 1; i++)
            {
                //https 协议基本认证 Authorization
                string url = basicUrl + "/api/base/Login?userId=zhyongfeng";
                ServicePointManager.ServerCertificateValidationCallback = delegate { return
true; };
                HttpWebRequest req = (HttpWebRequest)WebRequest.Create(url);
                NetworkCredential credential = new NetworkCredential("zhyongfeng",
"123456");
                req.Credentials = credential;
                HttpWebResponse response = (HttpWebResponse)req.GetResponse();
                Stream responseStream = response.GetResponseStream();
                StreamReader streamReader = new StreamReader(responseStream,
Encoding.UTF8);
                html = streamReader.ReadToEnd().Replace("\"", "");
                Console.WriteLine("Redis Token 服务器保存时间为 12s");
                Console.WriteLine(String.Format("Redis 服务器返回的 Token 值为:{0}",
html));
            }
            //token 设置了 12s 有效期
```

```
                for (int j = 0; j < 5; j++)
                {
                    System.Threading.Thread.Sleep(1000);
                    string url = basicUrl + "/api/user/PostMessage";
                    Console.WriteLine(Common.WebApiHelper.Post(url,
JsonConvert.SerializeObject(new { msg = "hello" }), "zhyongfeng", html));
                }
                for (int j = 0; j < 10; j++)
                {
                    System.Threading.Thread.Sleep(1000);
                    string url = basicUrl + "/api/user/GetMachine";
                    Console.WriteLine(Common.WebApiHelper.Get(url,      null,      "zhyongfeng",
html));
                }
                Console.Read();
            }

        }
}
```

9 部署 WebApi 到本机

将 WebApi 部署到以下 10.93.85.66（因网络限制，所以这里只做单个集群，虚拟机 ubuntu 中的数据 redis 主要是 NAT 网络连接方式，使用端口转发进行访问）



Hello, welcome to web api!

10 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
```

Nginx 的集群配置：

```
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    10.93.85.66:20107;
    }
    server {
        listen            80;
        server_name    localhost;
        location / {
            proxy_pass    http://zhyongfeng.com;
        }
        error_page    500 502 503 504    /50x.html;
        location = /50x.html {
```

```
                root     html;
            }
        }
}
```

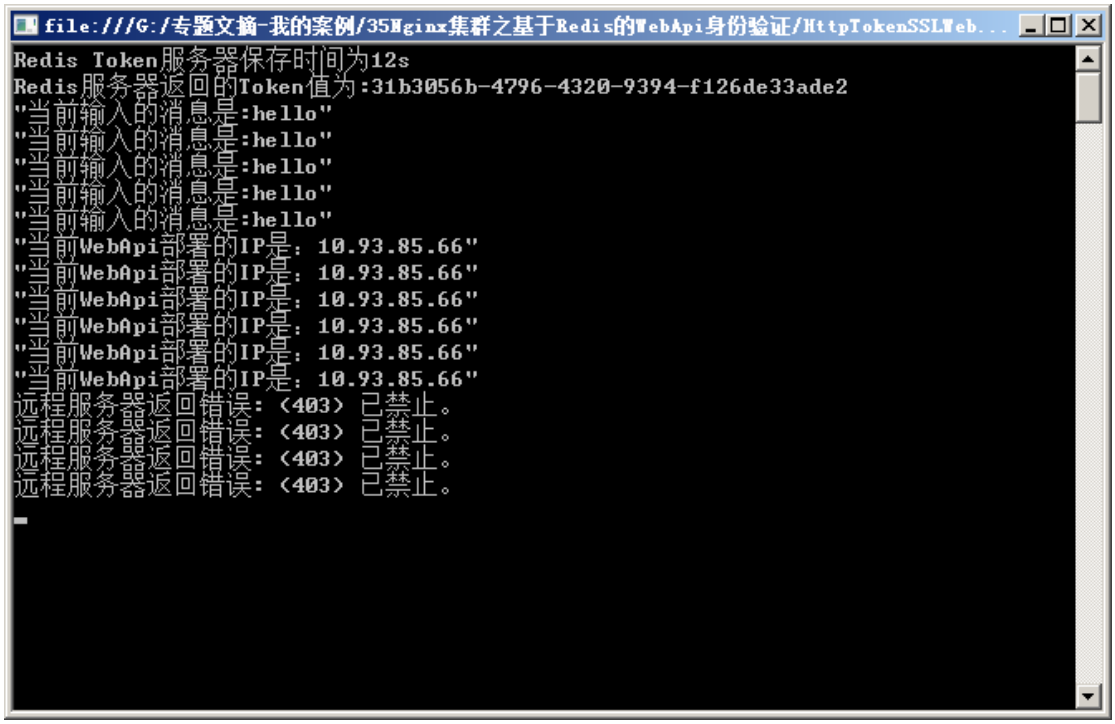
运行 CMD：

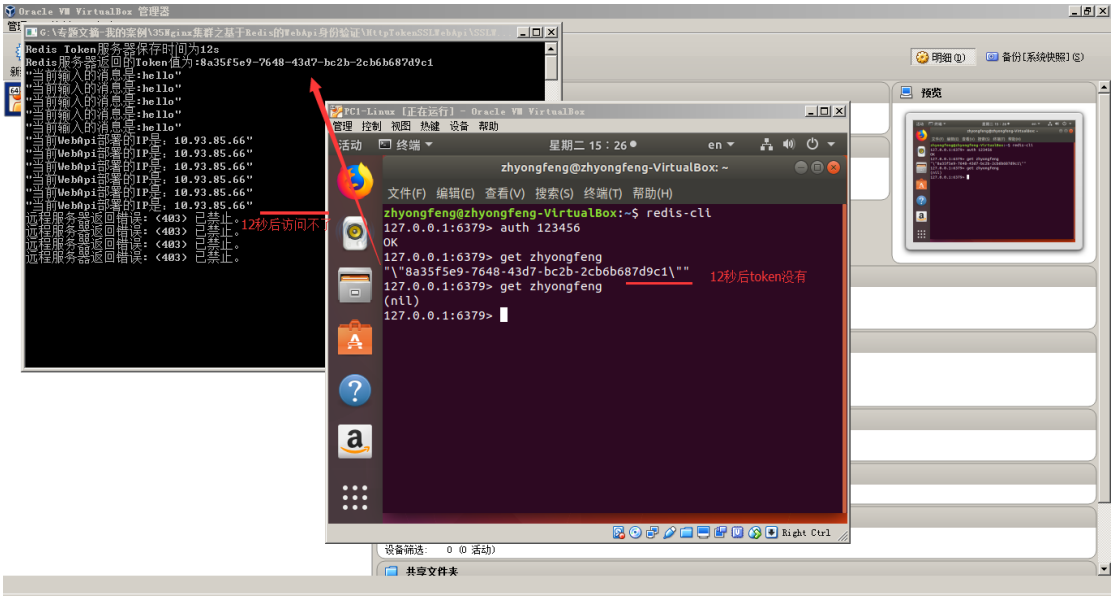
```
D:\DTLDownLoads\nginx-1.10.2>start nginx

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

11 运行结果

访问集群：http://zhyongfeng.com

WebApi 经过 Action 过滤器，生成了 Token 值，并存储到 Redis，运行结果如下：





12 总结

Nginx 集群使用 Redis 数据库，客户端利用 http basic 身份验证，访问 WebApi 获得 Token 并将 Token 存储到 Redis 内在数据库，通过 Token 值获取相应的权限数据，这样子可以做到单点登录，集群分布式的身份验证效果。既方便了用户在整个业务领域的系统操作，同时可以为整个公司、集团等各个区域的系统进行统一有效的身份验证管理。

源代码下载：
http://download.csdn.net/download/ruby_matlab/10155491

PDF 下载：
Nginx 集群之基于 Redis 的 WebApi 身份验证.pdf