Administração de Sistemas (ASIST)

Aula Teórico Prática 05 - 2018/2019

Administração de servidores.
Acesso remoto. Gestão de software.
Scripts. Execução programada.
Storage centralizada - SAN - iSCSI.

Gestão de servidores – acesso remoto

Uma vez que os endereços IP dos servidores estejam definidos, e de preferência registados no DNS, o administrador pode optar por usar protocolos de aplicação apropriados para aceder e administrar o servidor e abandonar a utilização da consola.

Servidores Linux: sem interface gráfica, a forma de acesso mais adequada é a utilização do protocolo SSH (Secure Shell). Para o efeito basta instalar um servidor SSH, tipicamente o ***OpenSSH Server***. O SSH é uma emulação de terminal efetuada sobre uma ligação TCP protegida por TLS (*Transport Layer Security*). Neste tipo de utilização o servidor não é autenticado por certificado de chave pública. Quando o utilizador liga a um servidor, este fornece a sua chave pública RSA, se for desconhecida, o utilizador é alertado.

Servidores Windows: para instalações com interface gráfica (e mesmo server core), a forma mais simples de acesso remoto é através do protocolo RDP (*Remote Desktop Protocol*). Basta ativar através do *Server Manager* e usar um cliente apropriado (*Remote Desktop Connection* - RDC / Ligação a Ambiente de Trabalho Remoto). Outra forma de gestão remota dos servidores é através do *Server Manager*. Esta aplicação permite a gestão não apenas do servidor local onde é executada, mas também outros servidores através da rede. Torna-se especialmente útil para gerir um conjunto de servidores pertencentes ao mesmo domínio Windows.

Gestão de software

Após a instalação inicial do servidor torna-se fundamental dispor de ferramentas que permitam de forma expedida a sua atualização e instalação de novo software necessário.

Linux: para a maioria das distribuições existe um conjunto de repositórios na internet através dos quais são fornecidas atualizações e acesso a diversos pacotes de software que se pretenda instalar. A aplicação/serviço usada para esse efeito depende da distribuição, no caso da família Debian na qual se integra o Ubuntu, a gestão de pacotes de software (*package management system*) é controlada através do comando **apt** (*Advanced Package Tool*).

Windows Server: a maioria do software essencial está disponível localmente após a instalação, pode ser gerido através do ***Server Manager*** adicionado/removendo roles e *features*. Um role é um conjunto de pacotes de software que permite desempenhar determinada função. *Features* são pacotes de software que não fazendo parte de um role desempenham outras funções que podem estar relacionadas com vários roles. É normal a instalação de um role obrigar à instalação de algumas *features*. Um servidor Windows pode desempenhar o WSUS Server role (*Windows Server Update Services*) este serviço permite disponibilizar localmente um serviço de atualização de software para os vários equipamentos locais.

Linux - APT (Advanced Package Tool)

Nas distribuições da família Debian o gestor de pacotes de software é o **apt**. A instalação de atualizações pode ser também automatizada através da instalação do pacote **unattended-upgrades**.

Comandos **apt** essenciais:

Comando	Descrição
apt update	Atualiza a lista de pacotes disponíveis descarregando dos vários repositórios as listas de pacotes existentes.
apt install {NOME-DO-PACOTE}	Solicita a instalação do pacote.
apt remove {NOME-DO-PACOTE}	Solicita a remoção do pacote, os ficheiros de configuração alterados pelo administrador não são removidos.
apt purge {NOME-DO-PACOTE}	Solicita a remoção do pacote incluindo os ficheiros de configuração (válido mesmo que o pacote já tenha sido removido).
apt dist-upgrade	Solicita uma atualização geral de todos os pacotes instalados.
apt autoremove	Solicita a remoção de todos os pacotes instalados que já não estão a ser utilizados (pacotes obsoletos que já foram substituídos por outros).

Linux - APT (Advanced Package Tool)

Comando	Descrição
apt list [--installed] [NOME-DO-PACOTE]	Sem opções, lista todos os pacotes existentes. Com a opção –installed, apenas pacotes instalados. É possível especificar um nome de pacote ou um padrão. Utilizando um padrão torna-se simples procurar um pacote que se pretenda. Por exemplo para obter uma lista de todos os pacotes existentes cujo o nome contém a palavra **libnss** pode ser usado o comando: **`apt list *libnss*`**
apt search {STRING}	Permite procurar pacotes através de um STRING, serão procuradas correspondências do STRING no nome e também na descrição detalhada do pacote.
apt show {NOME-DO-PACOTE}	Apresenta todos os detalhes sobre o pacote indicado.

O **apt** usa o **dpkg** (package manager for Debian) nem todas as possibilidade do dpkg estão disponíveis no apt. Frequentemente, durante a instalação de um pacote são executados scripts de instalação que solicitam ao utilizador as configurações a aplicar. Se depois da instalação for necessário alterar essas configurações, pode ser usado o comando:

`dpkg-reconfigure {NOME-DO-PACOTE}`

WSUS(*Windows Server Update Services*)

Como qualquer sistema operativo, no Windows Server devemos automatizar as atualizações, a opção recomendada é **descarregar e instalar automaticamente as atualizações**, para manter maior controlo podemos optar por **descarregar atualizações e pedir a confirmação antes de as instalar**.

A instalação de novas aplicações num servidor Windows para além das que estão disponíveis através do *Server Manager* (*roles and features*) deve ser feita com cautela, de preferência deverão ser aplicações da Microsoft cuja compatibilidade seja garantida para a versão do sistema operativo em causa.

Para descarregar pacotes de instalação usando o Internet Explorer, o administrador terá de desativar o modo ESC (**IE ESC – *Internet Explorer Enhanced Security Configuration***) através do Server Manager.

O WSUS Server role pode ser ativado num Windows Server para obter atualizações da Microsoft e disponibilizar as mesmas para as máquinas Windows locais. O WSUS pode ser configurado para obter atualizações para diversos tipos de sistemas operativos e linguagens. É possível controlar as atualizações que serão aplicadas a diferentes grupos de computadores locais e a hora de aplicação. Os computadores locais são configurados através de uma *Group Policy* para obterem as atualizações do servidor WSUS local.

WDS (*Windows Deployment Services*)

O WDS permite administrar um parque informático constituído por um grande número de máquinas Windows. O WDS só pode ser aplicado no contexto de um domínio *Active Directory*, exige também um servidor DHCP e postos de trabalho em que as interfaces de rede suportem PXE (*Preboot eXecution Environment*).

Permite:

- A instalação *unattended* do sistema operativo a partir de um *boot* de rede (PXE). A imagem do disco pode ser distribuída por multicast, efetuando a instalação em muitas máquina simultaneamente.
- O sistema operativo é gerido numa imagem residente no servidor WDS designada **imageX**. A imagem pode ser criada num posto de trabalho e copiada para o servidor ou totalmente criada no servidor.
- No servidor WDS é possível alterar a instalação (**imageX**), nomeadamente instalar software e drivers e aplicar automaticamente as alterações a todos os postos de trabalho (deploy).
- Durante o processo de instalação em cada posto de trabalho, são executados procedimentos de instalação do sistema operativo tendo em vista a sua adaptação ao hardware.
- Durante a instalação o nome da máquina é definido corretamente e a máquina pode ser automaticamente inserida no domínio Windows.

Scripts

Atualmente este é um conceito muito abrangente. Os *scripts* começaram pode ser *batch files*, ou seja ficheiros de texto com uma sequencia de comandos, o *batch file* pode ser invocado provocando a execução sequencial dos comandos (execução em lote).

Atualmente todas as linguagens de *scripting* incluem também comandos que permitem definir variáveis, estruturas de decisão e repetição assemelhando-se muito a verdadeiros programas. O conceito atual de script está fortemente associado ao facto de ser executado por um programa interpretador e não se tratar de um programa compilado.

Para que um *script* (ficheiro de texto), possa ser diretamente executado pelo sistema operativo sem necessidade de invocar explicitamente o programa interpretador é necessário que o sistema operativo o reconheça como tal.

Windows: o tipo de conteúdo de cada ficheiro é identificado através do seu nome, mais exatamente da sua **extensão** (conjunto de 3 carateres no final do nome após o ponto). Para cada extensão o sistema operativo tem definida a forma de lidar com o ficheiro, no caso dos *scripts*, o programa interpretador que deve executar para o processar.

Linux: o tipo de conteúdo de cada ficheiro é identificado através do ***magic number*** do seu conteúdo, o nome é completamente ignorado.

Unix - Magic number

Os sistemas operativos Unix identificam o tipo de conteúdo dos ficheiros através de ***magic numbers***. Um ***magic number*** é um conjunto de bytes com valores fixos existentes em determinadas posições fixas do ficheiro que permite identificar o seu conteúdo.

Na Shell, o comando **file** permite determinar de que tipo é um objeto do sistemas de ficheiros. Depois de verificar que não se trata de uma pasta, uma ligação simbólica ou algum tipo de objeto especial, verifica o seu conteúdo e confronta-o com os ***magic number*** conhecidos.

O *magic number* que identifica um script é constituído pelo caracteres **#!** nas duas primeiras posições do ficheiro. Esta sequência no inicio de um ficheiro é designada **shebang**, imediatamente a seguir, na mesma linha do ficheiro de texto, deve ser identificado o programa interpretador que deve ser usado para processar o conteúdo.

```
#!/bin/bash
--- --
-- -- --
```

```
#!/usr/bin/perl
--- --
-- -- --
```

```
#!/bin/sh
--- --
-- -- --
```

```
#!/bin/csh
--- --
-- -- --
```

```
#!/usr/bin/pyton
--- --
-- -- --
```

Adicionalmente o ficheiro deve ter a permissão execute para poder ser diretamente executado pelo sistema operativo. Note-se que as linhas começadas pelo caractere # são consideradas comentários pelos interpretadores pelo que a linha do shebang não interfere com a execução do script.

```
#!/usr/bin/php
--- --
-- -- --
```

Linux – Shell scripts

Trata-se de scripts que são executados pelos interpretadores de linha de comando habitualmente designados Shell, destacando-se a *Bourne Shell* (sh/bash) e a C Shell (csh/tcsh), entre outras.

Se um ficheiro começa pelo *shebang*, mas nessa linha não se identifica o programa interpretador, será usada a *default* Shell, normalmente a **bash.**

Tanto a *Bourne Shell* tradicional como a C Shell estão disponíveis em qualquer instalação Linux pelo que constituem uma ferramenta importante que está sempre disponível para o administrador automatizar tarefas, basta um editor de texto.

Os Scripts escritos em C Shell utilizam uma sintaxe bastante semelhante à linguagem de programação C, tornam-se interessantes para administradores que trabalham habitualmente com essa linguagem.

Outros tipos de linguagens interpretadas como *Perl* e *Pyton* possuem potencialidades muito maiores, com bibliotecas bastante extensas, mas exigem a familiarização do administrador.

O exemplo apresentado à direita, escrito em C Shell, imprime todas as potências de dois inferiores a 1100.

```
#!/bin/csh
echo "Starting"
@ num = 1
while($num<1100)
 echo $num
 @ num = 2 * $num
end
```

Linux – Shell scripts

O exemplo seguinte, escrito em bash, recebe um número como argumento na linha de comando (${1} é o primeiro argumento) e imprime o valor do fatorial desse número.

```
#!/bin/bash
echo "Calculador de fatorial"
if [ -z "${1}" ]
        then
        echo "Deve fornecer o argumento (inteiro positivo)"
        exit 1
fi
NUM=${1}
FACT=1
while [ ${NUM} -gt 1 ]
        do
        FACT=$((${FACT}*${NUM}))
        NUM=$((${NUM}-1))
done
echo "Fatorial de ${1} = ${FACT}"
#### DONE
```

A colocação de chavetas a envolver os nomes das variáveis referenciadas não é obrigatória. Os espaços colocados no contexto dos testes de condições (if e while) são fundamentais.

Scripts em Windows – WSH (Windows Script Host)

Nos sistemas operativos Windows os scripts são identificados através da sua extensão:

Extensão	Tipo de script
.BAT .CMD	Ficheiros processados pelo interpretador de linha de comandos (cmd.exe).
.JS .JSE	**JScript** – Linguagem interpretada baseada no Microsoft Visual J++ 6.0.
.VBS .VBE	**VBScript** – Linguagem interpretada baseada no Microsoft Visual Basic 6.0.
.WSF	**Windows Script File** – Utiliza XML para suportar diferentes linguagens no mesmo script, por exemplo JScript e VBScript.
.PS1	**Windows PowerShell** shell script

Os sistemas Windows atuais dispõem de uma plataforma de suporte à execução de scripts designada ***Windows Script Host*** (WSH).

Esta plataforma permite às linguagens interpretadas compatíveis, nomeadamente JScript e VBScript, o acesso a um vasto conjunto de funcionalidades através de objetos que podem utilizar. Através do WSH é por exemplo possível utilizar o WMI (*Windows Management Instrumentation*) e o ADSI (*Active Directory Service Interfaces*).

WSH (Windows Script Host) - Exemplo

Neste exemplo em JScript é usado o WSH para aceder ao WMI e obter uma lista de contas de utilizador. Para cada conta é apresentada sucessivamente uma janela com o nome de utilizador e a data do último login:

```
LoginProfiles = GetObject("winmgmts:").InstancesOf ("Win32_NetworkLoginProfile");
for(e = new Enumerator(LoginProfiles) ; !e.atEnd() ; e.moveNext())
{
   Profile = e.item();
   lastLog="never";
   if(Profile.LastLogon!=null) lastLog=Profile.LastLogon;
   WScript.Echo(Profile.Name + " : " + lastLog);
}
```

O exemplo seguinte, em VBScript faz o mesmo:

```
Set LoginProfiles = GetObject("winmgmts:").InstancesOf ("Win32_NetworkLoginProfile")
   for each Profile in LoginProfiles
      WScript.Echo Profile.Name
      WScript.Echo Profile.LastLogon
   next
```

WSF (Windows Script File) - Exemplo

Um script WSF utiliza XML e pode conter diferentes linguagens:

```
<job>
<script language="VBScript">
  MsgBox "Eu agora estou a executar VBScript, mas a seguir vou executar JScript"
</script>
<script language="JScript">
  WSH.echo("Agora já estou a executar JScript e vou mostrar as contas de utilizador.");

  LoginProfiles = GetObject("winmgmts:").InstancesOf ("Win32_NetworkLoginProfile");
  num=0;
  for(e = new Enumerator(LoginProfiles) ; !e.atEnd() ; e.moveNext())
   {
   Profile = e.item();
   lastLog="never";
   if(Profile.LastLogon!=null) lastLog=Profile.LastLogon;
   WScript.Echo(Profile.Name + " : " + lastLog);
   num++;
   }
  WScript.Echo("Foram apresentadas " + num + " contas de utilizador");
  WScript.quit();
</script>
</job>
```

Windows PowerShell

A PowerShell disponibiliza um conjunto suplementar de comandos designados ***cmdlets***, permitem administrar todas as funcionalidades do servidor. Foi concebida com o propósito de facilitar as tarefas de administração dos servidores Windows.

A quantidade de cmdlets é imensa, os nomes são orientados pelo verbo da ação pretendida, por exemplo para consultar o estado são usados comandos começados por **Get-**, para alterar comandos começados por **Set-**.

Desde logo, o comando **Get-help** pode ser usado para obter ajuda, por exemplo **Get-help service** permite obter uma listas dos cmdlets cujo nome contém **service**. Ajuda detalhada sobre o cmdlet pode depois ser pedida diretamente, por exemplo **Get-help Start-service**.

A PowerShell também dispõe de *autocomplete* através da tecla TAB.

O desenvolvimento de scripts é muito facilitado pelo ***Integrated Scripting Environment*** (ISE). Trata-se de uma ferramenta em modo gráfico de desenvolvimento de scripts PowerShell com assistência direta disponível, desde logo com informação sobre todos os cmdlets disponíveis.

Os scripts (ficheiros com extensão .PS1) são editados no ISE e podem desde logo ser testados no próprio ambiente. O ambiente de desenvolvimento de scripts PowerShell ISE encontra-se disponível nas ***Administrative Tools*** do Windows Server.

Windows PowerShell script – exemplos WMI

```
# WMI query to list all properties and values of the Win32_BIOS class
# http://www.robvanderwoude.com

param( [string]$strComputer = "." )

$colItems = get-wmiobject -class "Win32_BIOS" -namespace "root\CIMV2" -computername $strComputer

foreach ($objItem in $colItems) {
          write-host "Name                            :" $objItem.Name
          write-host "Version                         :" $objItem.Version
          write-host "Manufacturer                    :" $objItem.Manufacturer
          write-host "SMBIOSBIOS Version              :" $objItem.SMBIOSBIOSVersion
          write-host
}

# WMI query to list all properties and values of the Win32_Printer class
# http://www.robvanderwoude.com

$colItems = get-wmiobject -class "Win32_Printer" -namespace "root\CIMV2" -computername $strComputer

foreach ($objItem in $colItems) {
          write-host "Name                            :" $objItem.Name
          write-host "Default                         :" $objItem.Default
          write-host "Network                         :" $objItem.Network
          write-host "Port Name                       :" $objItem.PortName
          write-host "Driver Name                     :" $objItem.DriverName
          write-host "Server Name                     :" $objItem.ServerName
          write-host "Share Name                      :" $objItem.ShareName
          write-host
}
```

Execução programada de tarefas

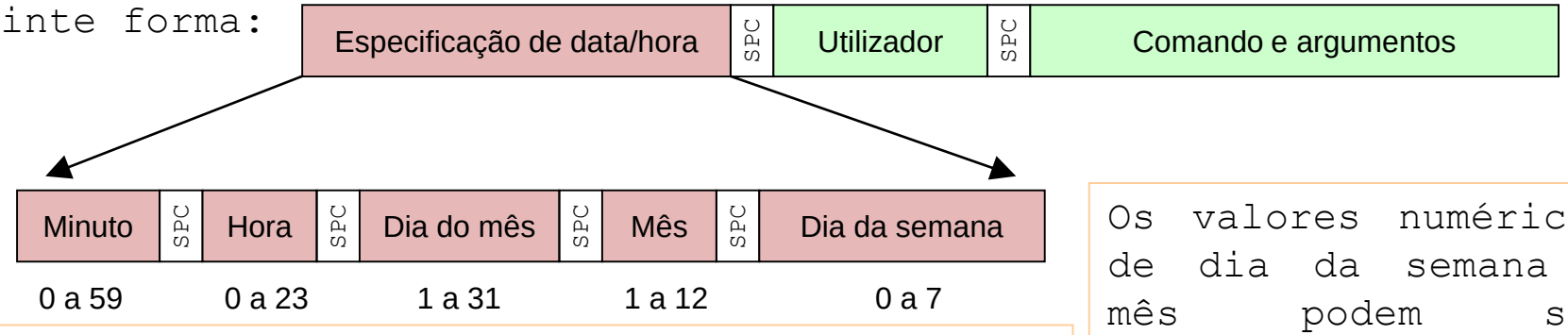
A execução programada de tarefas é uma ferramenta fundamental na administração de servidores, permite ao administrador definir a execução de aplicações e scripts em modo *unattended* em horas apropriadas. Um exemplo típico são operações de cópia de segurança.

Windows: a aplicação gráfica ***Task Scheduler*** permite programar tarefas. É possível especificar uma grande variedade de parâmetros relativamente à execução de tarefas, desde logo a conta de utilizador que será usada e condições especificas que poderão condicionar a execução da tarefa. Além de uma programação horária ao segundo, as tarefas também podem ser programadas para serem executadas em outros tipos de situações, como por exemplo no *boot* da máquina ou quando a máquina está *idle*. Também é possível definir uma data de expiração da tarefa, a tarefa não será eliminada mas ficará inativa a partir dessa data e não será executada.

Linux: nos sistemas Unix em geral o programador de tarefas mais usado é o **cron**. Trata-se de um serviço em execução permanente que de minuto a minuto lê o ficheiro de configuração **/etc/crontab** e executa as tarefas aplicáveis nesse minuto. Através das definições neste ficheiro podemos estabelecer horários de execução baseados na hora, minuto, dia do mês e dia da semana. A granularidade mínima é portanto o minuto.

Linux – Serviço cron

O serviço **cron** permite a execução programada de tarefas no Linux. Normalmente o administrador define as tarefas no ficheiro de texto **/etc/crontab**. Cada linha representa uma tarefa a ser executada na seguinte forma:



Os valores de data/hora podem ser intervalos ou enumeração dos valores separados por virgulas. O símbolo ***** representa a enumeração de todos os valores possíveis. Por exemplo para minutos, a enumeração de todos os valores possíveis pode ser feita na forma **0-59** ou *****.

Os valores numéricos de dia da semana e mês podem ser substituídos por nomes. Nos dias da semana os valores **0** e **7** são duas alternativas para especificar **domingo**.

Em substituição da especificação de data/hora também pode podem ser usados *strings* especiais, um deles é **@reboot**, que provoca a execução do comando logo após o arranque do serviço **cron**.

De minuto a minuto o serviço **cron** lê este ficheiro e compara a data/hora atual com as especificações existentes no ficheiro. Para cada correspondência encontrada executa o comando usando a conta de utilizador especificada.

Linux – exemplo /etc/crontab

```
*/5 * * * * root /usr/bin/mrtg /etc/mrtg/mrtg.cfg
5,35 * * * * root /etc/LinuxHealth >/dev/null 2>&1 &
45 2 * 8 6 root /root/make-backup >/dev/null 2>&1 &
```

As enumerações de valores podem ser associadas a um ***step***, no exemplo acima, nos minutos, ***/5** (equivalente a **0-59/5**) é equivalente a **0,5,10,15,20,25,30,35,40,45,50,55.** Neste caso (como as restantes especificações são *****) a tarefa será executada de cinco em cinco minutos, sempre que o valor data/hora tenha um número de minutos correspondente.

O ficheiro **/etc/crontab** apenas pode ser editado pelo administrador, no entanto o serviço *CRON* também está acessível aos outros utilizadores através de ficheiros de configuração normalmente guardados em **/var/spool/cron/**. Estes devem ser manipulados indiretamente através do comando **crontab –e**.

O formato é idêntico ao do **/etc/crontab**, mas o nome do utilizador (6º campo) é omitido pois está implícito, as tarefas de um utilizador definidas desta forma são sempre executadas usando a conta desse utilizador. O acesso dos utilizadores ao serviço *CRON* pode ser controlado pelo administrador através dos ficheiros **/etc/cron.allow** e **/etc/cron.deny**.

Storage centralizada – SAN e NAS

A disponibilização centralizada através da rede de dispositivos de armazenamento externo (*storage*) pode ser realizada através de duas abordagens distintas:

SAN (*Storage Area Network*) – Disponibilização de discos lógicos. Sob o ponto de vista do servidor de *storage* o disco lógico (LUN - *Logical Unit*) pode ser um disco inteiro, uma partição ou volume e até um ficheiro. **Sob o ponto de vista do cliente o disco lógico é tratado da mesma forma que um qualquer outro disco local.** Para um disco lógico recém criado, o cliente terá de criar uma tabelas de partições, formatar as partições e só depois as poderá usar, montando-as ou associando-lhes letras de drive no caso do Windows. Existe hardware especializado para criar este tipo de infraestrutura, notoriamente o **fibre channel.** Existem também implementações que podem funcionar sobre hardware corrente, o **iSCSI** que pode operar sobre qualquer rede IP independentemente do hardware subjacente.

NAS (*Network Attached Storage*) – Disponibilização de pastas. Corresponde ao vulgar conceito de partilha de pastas. No servidor NAS pastas do sistema de ficheiros local são disponibilizadas aos clientes, ao contrário dos discos lógicos a responsabilidade de gerir o sistema de ficheiros cabe ao servidor. Os sistemas Windows utilizam o protocolo (SMB/CIFS) para **partilhar pastas.** Os sistemas Unix usam o protocolo NFS para **exportar pastas.** A terminologia é diferente, mas o conceito é o mesmo.

SAN (Storage Area Network) - iSCSI

O protocolo ***iSCSI*** permite criar uma SAN sobre qualquer infraestrutura de rede IP, não exige hardware especial. Servidores Windows e servidores Linux podem ser **servidores de *storage* (*targets*)** ou **clientes de *storage* (*initiators*)** que usam os discos lógicos.

Os nós intervenientes no *iSCSI* (*targets* e *initiators*) são identificados através de nomes especiais, normalmente no formato **IQN (*iSCSI Qualified Name*)**, na forma:

iqn.{YYYY-MM}.{nome-dns-invertido}:{nome}

Onde **{YYYY-MM}** representa o ano e mês em que foi criado o domínio DNS, **{nome-dns-invertido}** representa o nome DNS do domínio invertido e **{nome}** é um nome arbitrário que garante que o conjunto é único, por exemplo o nome do nó (no caso de um *target* habitualmente este nome também identifica o disco lógico).

Exemplo de IQN de um target: **iqn.2000-08.pt.ipp.isep.dei:server1-lun0**

Os IQN usados pelos *targets* podem ser descobertos através da função *discovery*, desde que se conheça os seus endereço IP ou nomes DNS.

O acesso dos *initiators* aos targets é controlado através de *username* e *password* sendo usado o algoritmo CHAP.

iSCSI Initiator – Windows

Os sistemas Windows atuais (incluindo o Windows Server) pode usar discos lógicos disponibilizados por targets, ou seja operar como *initiator*. Para gerir esta funcionalidade deve ser usada a aplicação **iSCSI Initiator** disponível nas **Administrative Tools**.

O target pode ser adicionado através do endereço IP ou nome DNS respetivo, é necessário então ligar ao target (*Connect*) definindo o *username* e *password* para autenticação via CHAP (opções avançadas do *Connect*) tal como foram definidos pelo target.

Se o target for adicionado à lista de favoritos, a ligação será feita automaticamente sempre que a máquina arranca.

Após a ligação ao target, podemos optar por configurar automaticamente todos os discos lógicos (volumes) ou apenas alguns dos disponibilizados no target. Estando configurados os volumes passam a ser novos discos locais disponíveis.

Tratando-se de um disco vazio recém criado, através do **Computer Management** devemos colocar o disco on-line e inicializa-lo (criar a tabela de partições) após o que podem ser criadas novas partições para serem formatadas e atribuídas letras de drive.

Para o **Computer Management** tudo se passa como se tratasse de um disco físico local. O disco pode também ser convertido para ***dynamic***.

iSCSI Target – Windows

O **Windows Server 2012** pode funcionar como servidor de *storage* iSCSI (ou seja target iSCSI). Para o efeito terão de ser instalados os roles **File Server** e **iSCSI Target Server** (encontram-se no conjunto de roles ***File and Storage Services***). A *feature* **iSNS Server** (*discovery services*) também será instalada.

Através do **Server Manager** é então possível criar discos lógicos a disponibilizar pelo target.

Apenas são suportados discos virtuais, ou seja, discos que são armazenados sob a forma de um ficheiro de grande dimensão (correspondente ao tamanho do disco).

Ao criar o disco virtual temos de definir a sua dimensão, mais tarde pode ser expandido se necessário, mas não pode ser encolhido.

Os discos virtuais podem ser criados de várias formas, ***Fixed Size*** significa que será criado imediatamente um ficheiro com a dimensão correspondente ao tamanho do disco. ***Dynamically Expanding*** leva a que o tamanho do ficheiro vá crescendo à medida que vão sendo colocados dados no disco. ***Differencing*** permite criar um novo disco usando como base um disco já existente, o disco já existente é preservado no estado inicial e as alterações relativamente a ele são registadas no novo disco. Depois de criado o disco virtual pode então ser definido o target correspondente.

iSCSI Target – Linux

No **Ubuntu 16.04** o pacote que permite criar um target iSCSI é o **iscsitarget**. Após a sua instalação a configuração é realizada através do ficheiro **/etc/iet/ietd.conf**, neste ficheiro podem ser declarados nomes de targets (IQN) correspondentes a discos a disponibilizar.

Um disco lógico pode ser um disco físico, por exemplo /dev/sdc, também pode ser uma partição, por exemplo /dev/sdb3, ou à semelhança do Windows Server também pode ser um ficheiro que funciona como disco virtual. Ao definir o target são também definidas as credenciais de autenticação através do CHAP. Exemplo:

```
Target iqn.2015-10.pt.ipp.isep.dei:storage.lun0
        IncomingUser root PASSWORD
        OutgoingUser
        Lun 0 Path=/home/storage/lun0.disk,Type=fileio
```

Neste caso este target é um disco virtual armazenado no ficheiro **/home/storage/lun0.disk**, este ficheiro terá de ser previamente criado com a dimensão pretendida para o disco lógico, podendo ser preenchido inicialmente com quaisquer dados. Posteriormente ele será formatado pelo *initiator* que o utilizar.

No Ubuntu, para o serviço **iscsitarget** arrancar terá de ser *enabled* no ficheiro **/etc/default/iscsitarget**.

iSCSI Initiator – Linux

No **Ubuntu 16.04** o pacote que permite usar discos lógicos disponibilizados por targets iSCSI é o **open-iscsi** (Open-iSCSI project http://www.open-iscsi.org/). O primeiro passo é descobrir os nomes dos targets através do comando:

iscsiadm -m discovery -p {SERVER} -t st

Onde **{SERVER}** é o endereço IP ou nome DNS da máquina target, identificado o nome IQN do disco lógico pretendido, é necessário definir o método de autenticação (CHAP), o username e a password para esse disco lógico.

```
iscsiadm -m node -T {IQN} –p {SERVER} –o update –n node.session.auth.authmethod –v CHAP
iscsiadm -m node -T {IQN} –p {SERVER} –o update –n node.session.auth.username –v {USERNAME}
iscsiadm -m node -T {IQN} –p {SERVER} –o update –n node.session.auth.password –v {PASSWORD}
iscsiadm -m node -T {IQN} –p {SERVER} -o update -n node.startup -v automatic
```

Estes dados são guardados e serão usados para ligar ao disco lógico {IQN}. Para ligar ao target basta agora usar o comando:

iscsiadm -m node –login -T {IQN} –p {SERVER}

Os targets configurados com os atributos fornecidos ficam guardados dentro da pasta **/etc/iscsi/nodes** de forma persistente, os ficheiros de configuração podem ser editados manualmente. No exemplo acima foi definido **node.startup = automatic**, quando a máquina arrancar a ligação a este target (login) será realizada automaticamente.

iSCSI Initiator – Linux

Por cada disco lógico a que o sistema se liga (login no target), surge no sistema um novo disco, por exemplo se o sistema apenas possuía o disco **/dev/sda** surgirá um novo disco **/dev/sdb.**

O sistema operativo trata este novo disco como se fosse um disco físico local, embora na realidade seja um disco lógico iSCSI. Se é um disco recém criado (no target) deverá ser usado o comando **fdisk** para definir as partições que depois poderão ser formatadas com o comando **mkfs**. Estando formatadas podem então ser montadas e eventualmente registadas no **/etc/fstab** para serem montadas sempre que o sistema arranca.

A identificação de disco local atribuída pode ser um problema quando são usados vários discos lógicos iSCSI. A atribuição de discos locais é feita pela ordem de ligação, manter consistentemente os mapeamentos entre identificadores de discos locais e targets exige mais algum trabalho. Por exemplo se um *target* não está disponível, os identificadores de disco local para os outros **targets** poderão ser diferentes.

Uma vantagem de se utilizar um protocolo standard como o iSCSI é a interoperabilidade, por exemplo *targets* Windows podem ser utilizados por *initiators* Linux e *targets* Linux podem ser usados por *initiators* Windows.