cron, Win Task Scheduler, Iptables

CSCI 4417/5417

Introduction to System Administration

East Tennessee State University
Department of Computing
Jack Ramsey, Lecturer

`cron`

Allows *nix users to schedule commands or scripts to run at given dates and times

Usually used for sysadmin jobs like backups or cleaning temporary directories

`cron` daemon runs in the background, checking **`/etc/crontab`** file, **`etc/cron.*`**, and /var/spool/cron directories

cron

```
jack@jram: ~
jack@jram:~$ ll /etc | grep "cron"
-rw-r--r--   1 root root     401 Jul  7  2014 anacrontab
drwxr-xr-x   2 root root    4096 Nov  4 07:46 cron.d/
drwxr-xr-x   2 root root    4096 Nov 18 07:47 cron.daily/
drwxr-xr-x   2 root root    4096 Jul 29 14:01 cron.hourly/
drwxr-xr-x   2 root root    4096 Jul 29 14:02 cron.monthly/
-rw-r--r--   1 root root     722 Jul  7  2014 crontab
drwxr-xr-x   2 root root    4096 Aug 12 08:03 cron.weekly/
jack@jram:~$
```

`cron`

Modify with **`crontab -e`**

Syntax:

min	hour	day	month	dow	script or command name to schedule
1	2	3	4	5	/path/to/command arg1 arg2

`cron`

Modify with **`crontab -e`**

Minute	0-59	
Hour	0-23	(24-hr clock; 0==midnight/23==11:00PM)
Day	0-31	
Day of the Week	0-7	(7 or 0 == Sunday)
/path/to/command	Script or command	

cron

```
root@jram: ~
  GNU nano 2.2.6                    File: /tmp/crontab.1mNQy7/crontab

#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
@reboot /usr/bin/ip-on
  0 0   *   *   7       sudo apt-get update -y
@reboot /home/jack/scripts/iptabset
```

cron Examples

System update every night at 12:05 AM

```
5 0 * * * apt-get update
```

cron Examples

Run script **`cleartemp.sh`** at 2:15 PM on the first of every month

```
15141 * * /root/scripts/cleartemp.sh
```

cron Examples

Run **phpscript.sh** at 10 PM on weekdays

```
0 221-5 * * /root/scripts/phpscript.sh
```

cron Operators

The asterisk (*) Any

Comma (,) List of values (eg., 0,2,4,6,8,10,12,14,16,18,20,22 in hours would signify every other hour)

Dash (-) Range of values (eg., 0-30)

Separator (/) Step value (eg., */2 in days would execute every other day of the month

`cron` Special Syntax

`@yearly` & **`@annually`** Runs the task at 12:00 AM on Jan 1 every year

`@daily & @midnight` Every day at 12:00 AM

`@weekly` Every week at 12:00 AM on Sunday

`@hourly` Runs the job at the top of every hour

`@reboot` Runs the job every time the machine reboots

`cron` Suppressing Job Output

By default, cron will send email to the executing user's email box

Any output or errors

If you don't want it to email, do this:

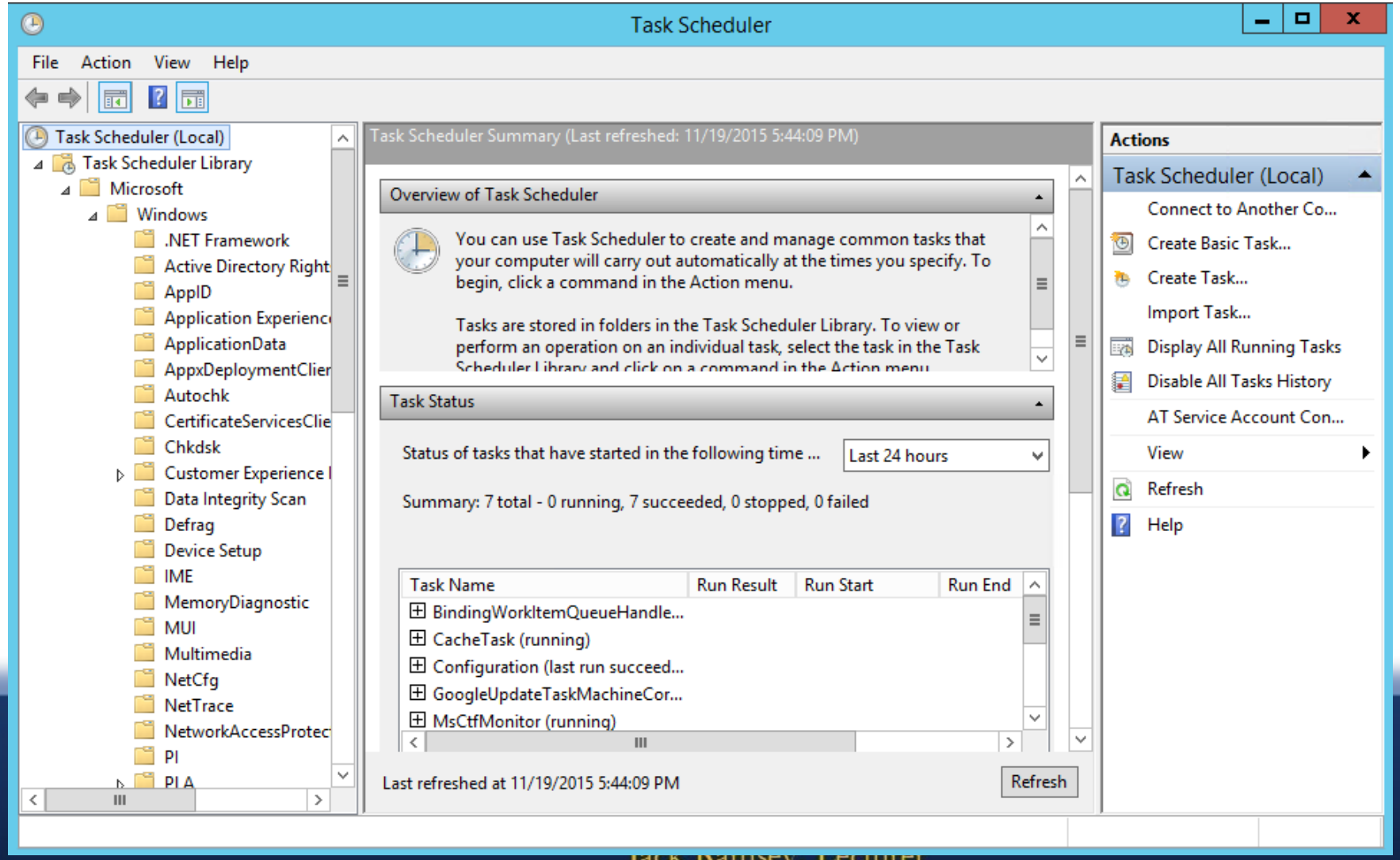
```
5  0  *  *  *  service mysql restart >/dev/null 2>&1
```

2>&1 redirects stderr (2) to stdout (1) so both of them get dumped into /dev/null

Windows Task Scheduler

East Tennessee State University
Department of Computing
Jack Ramsey, Lecturer

Windows Task Scheduler



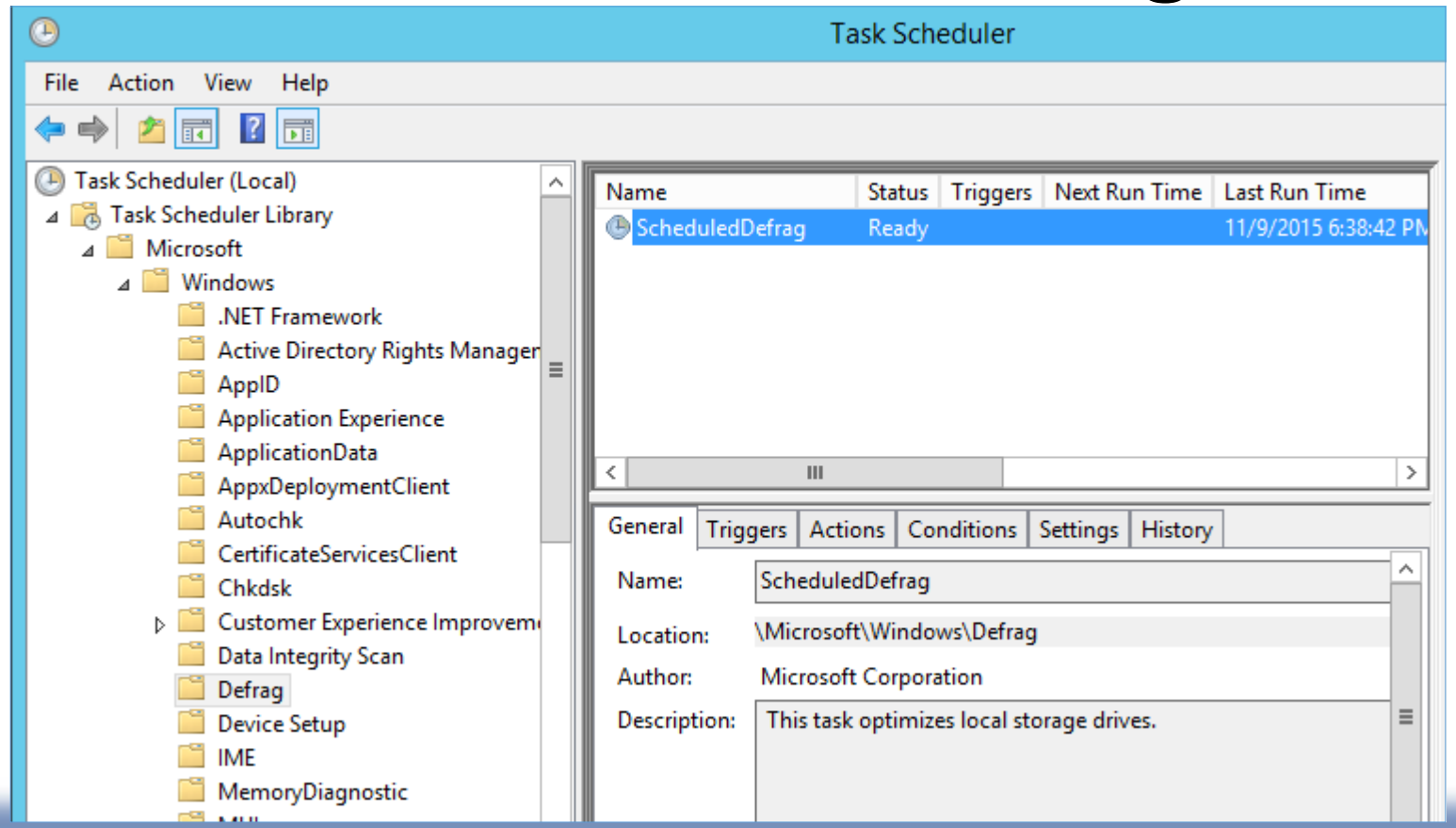
Windows Task Scheduler

Used to create, schedule, manage tasks

Tasks can be scheduled to run only when the user is logged in, or anytime

Check the status of existing tasks ->

Windows Task Scheduler - Defrag



Windows Task Scheduler - New Task

Task Scheduler

File Action View Help

Task Scheduler
Task Sched
Microso
Win

Create Task

General | Triggers | Actions | Conditions | Settings

Name:

Location: \Microsoft\Windows\Defrag

Author: RAMSEY\Administrator

Description:

Security options

When running the task, use the following user account:

RAMSEY\Administrator | Change User or Group...

Run only when user is logged on

Run whether user is logged on or not

Do not store password. The task will only have access to local computer resources.

Run with highest privileges

Hidden

Configure for: Windows Vista™, Windows Server™ 2008

OK | Cancel

Actions

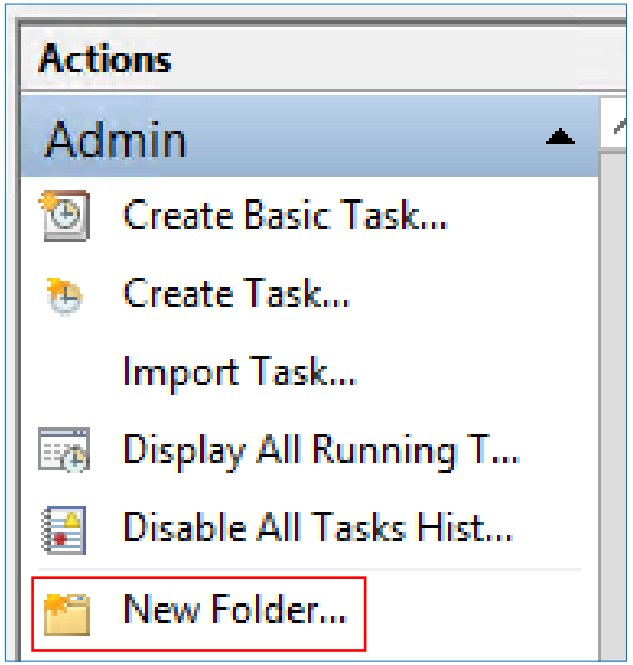
Defrag

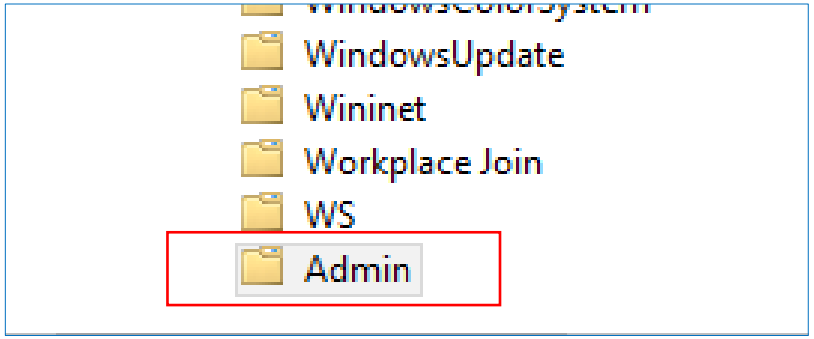
Create Basic Task...
Create Task...
Import Task...
Display All Running T...
Disable All Tasks Hist...
New Folder...
Delete Folder
View
Refresh
Help

Selected Item

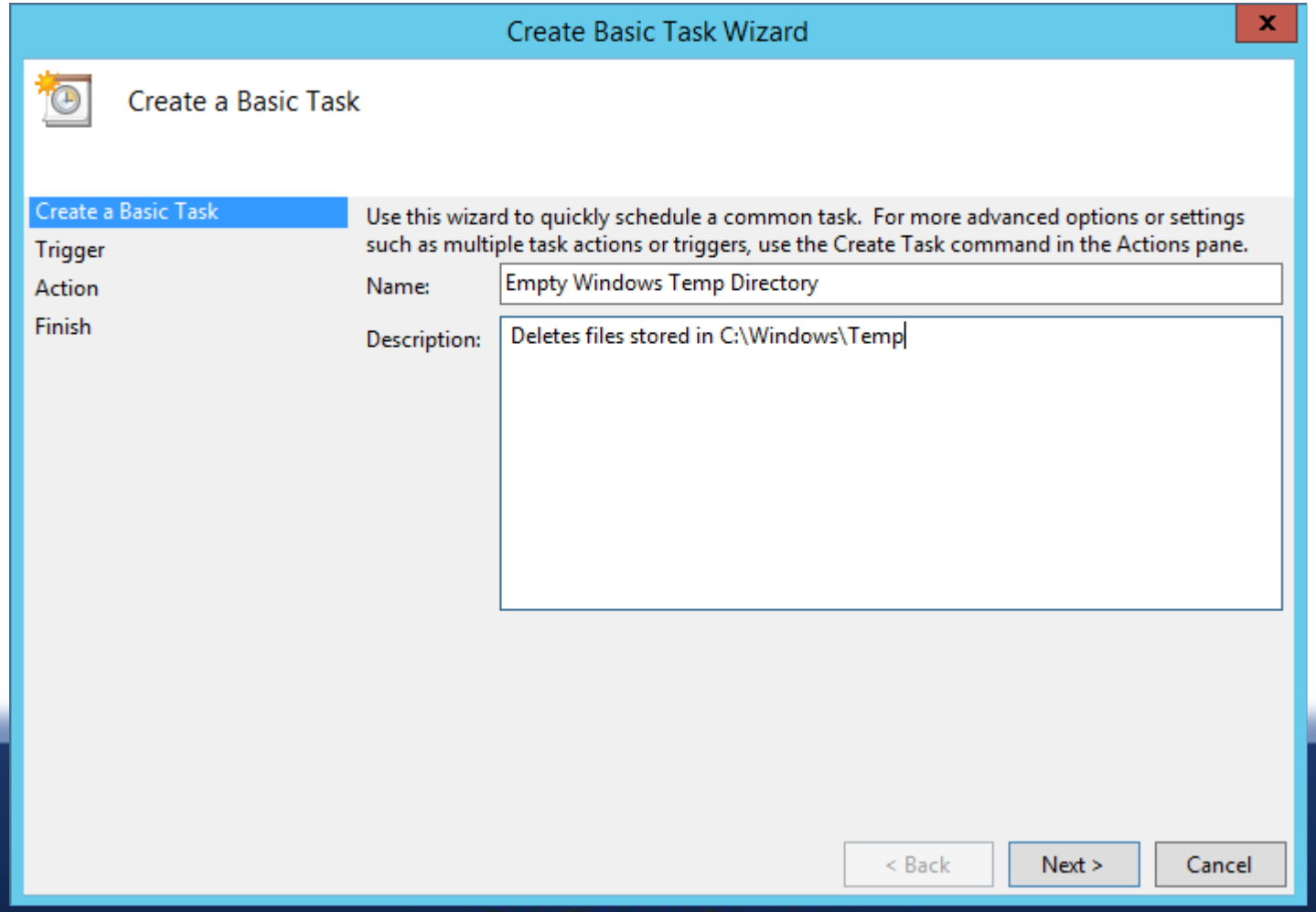
Run
End
Disable
Export...
Properties
Delete

Windows Task Scheduler - New Task

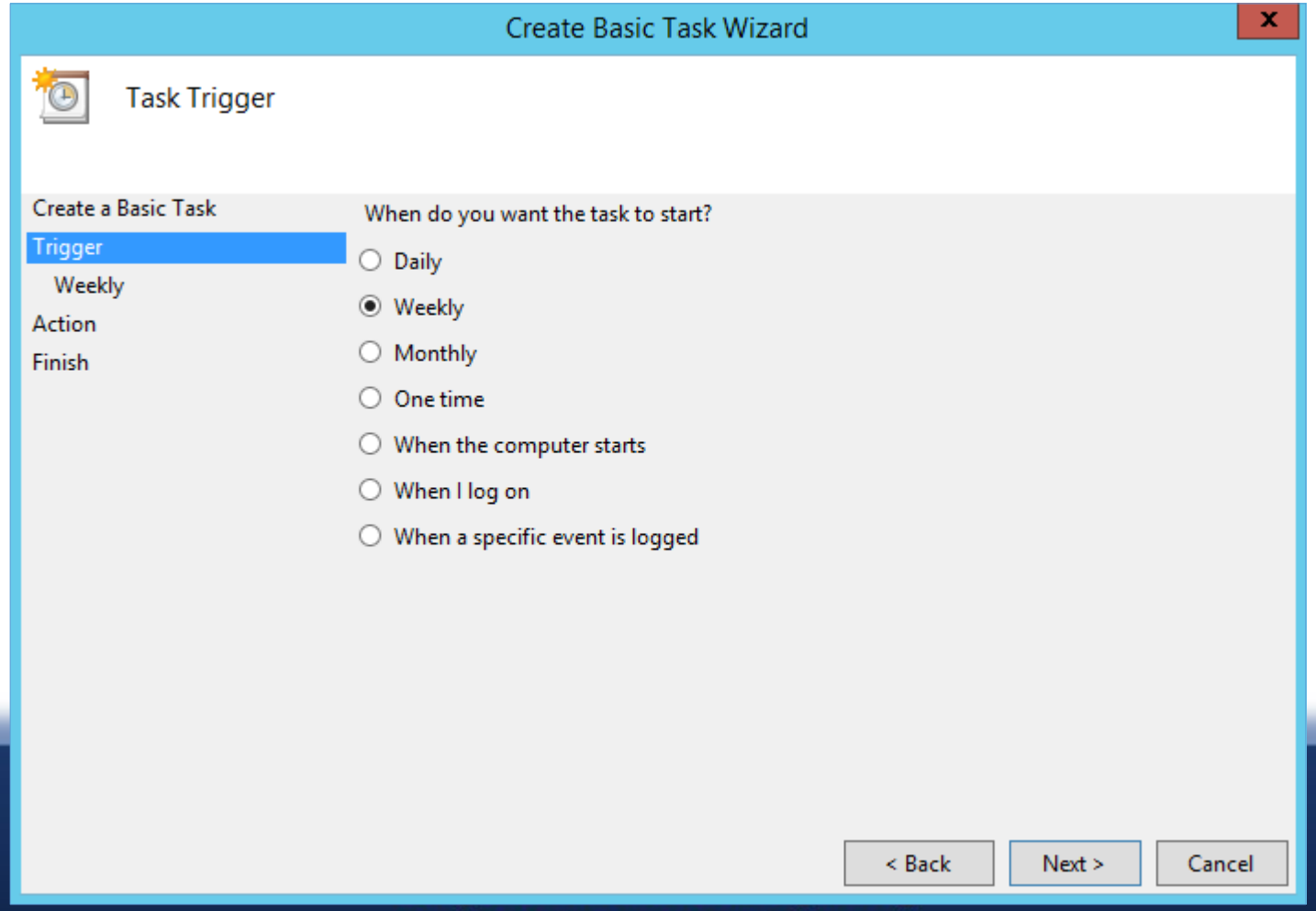




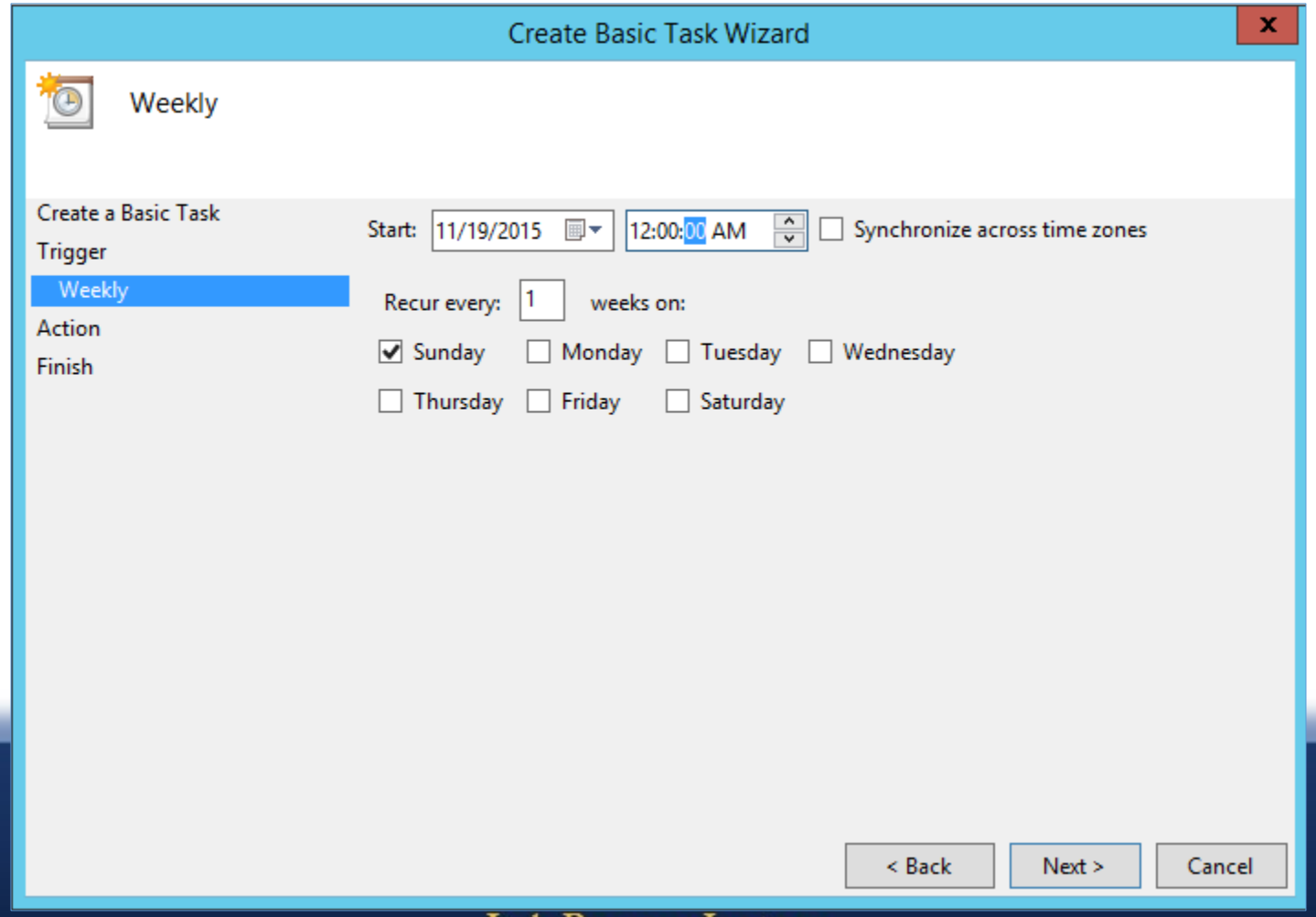
Windows Task Scheduler - New Task



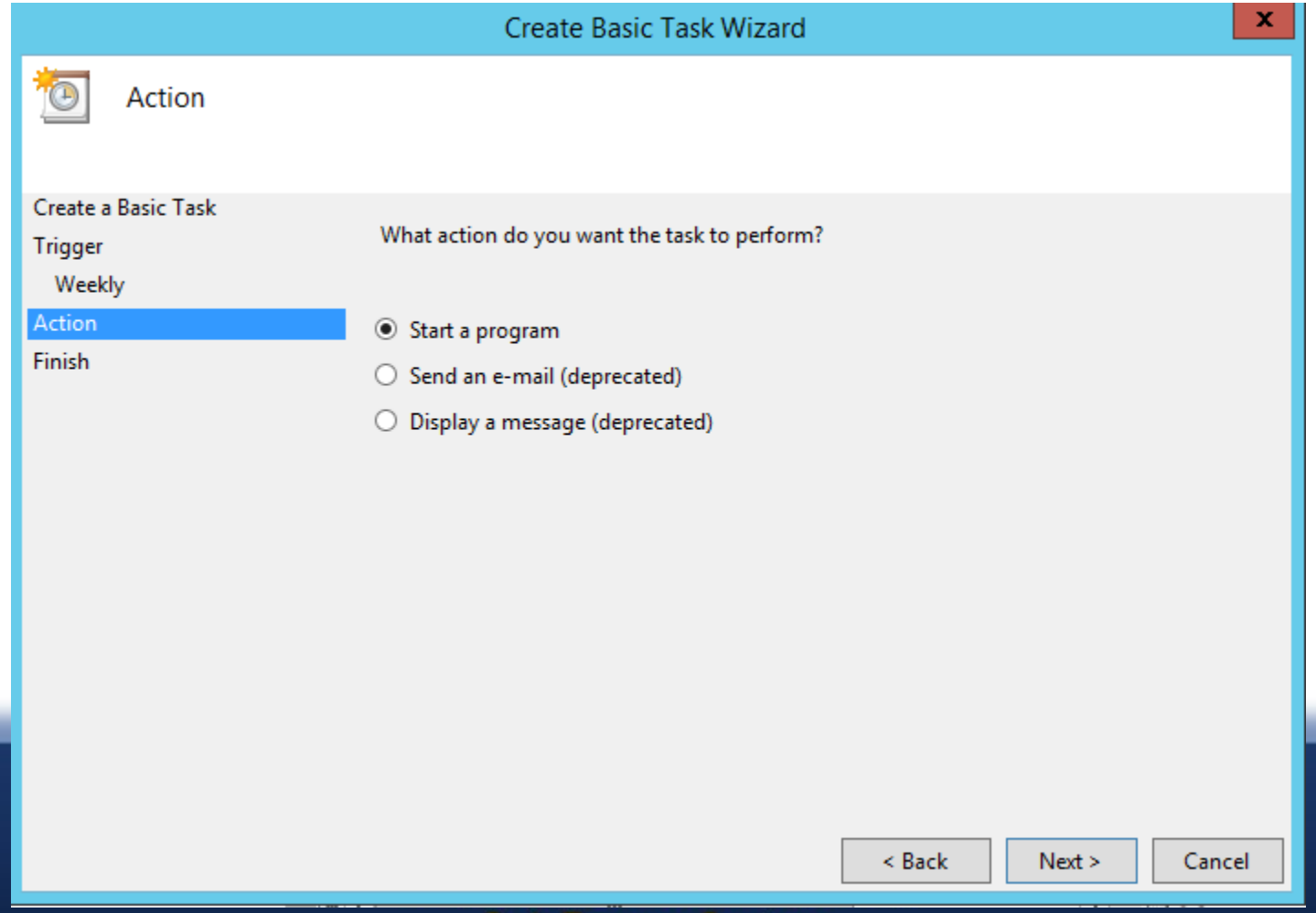
Windows Task Scheduler - New Task



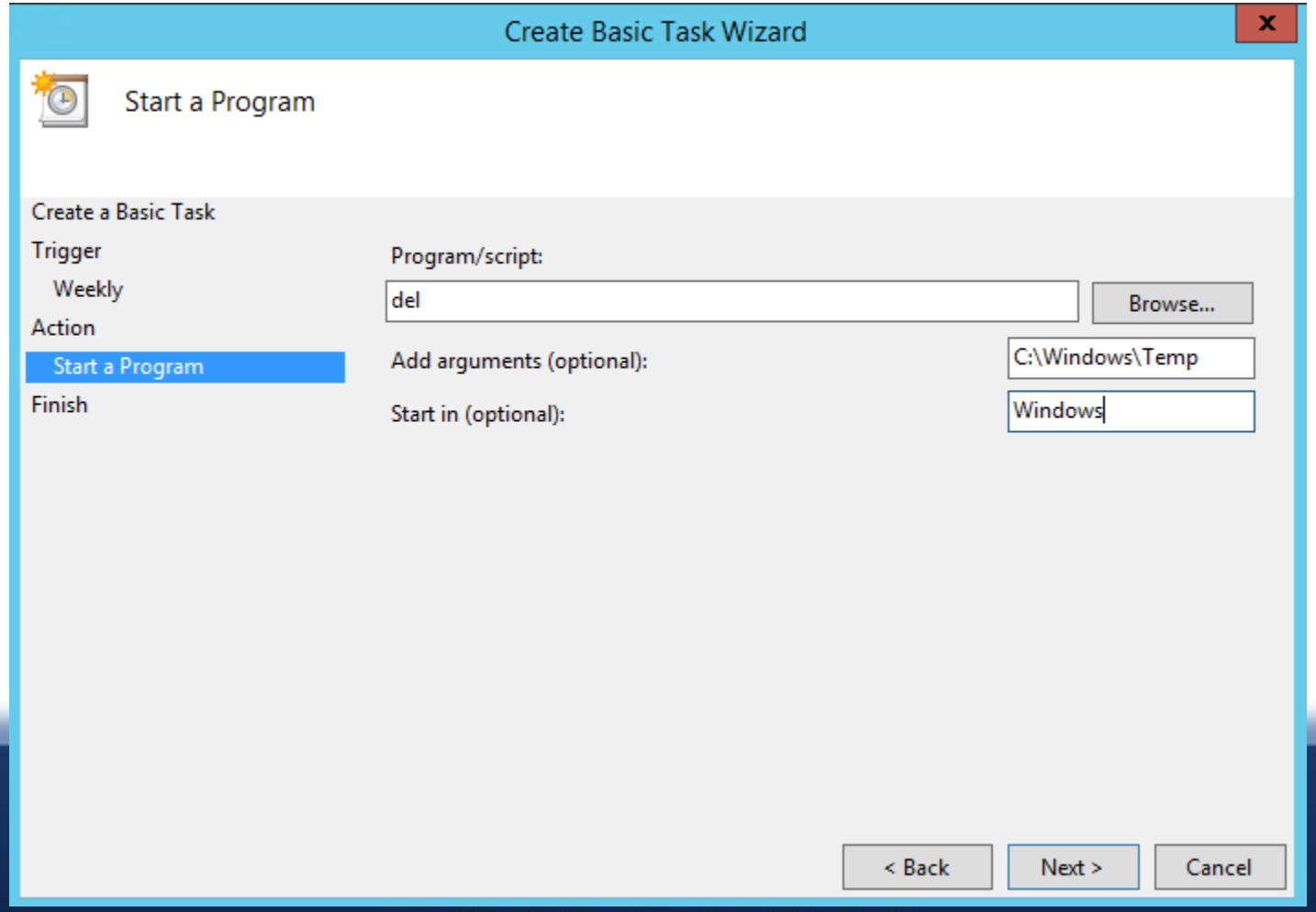
Windows Task Scheduler - New Task



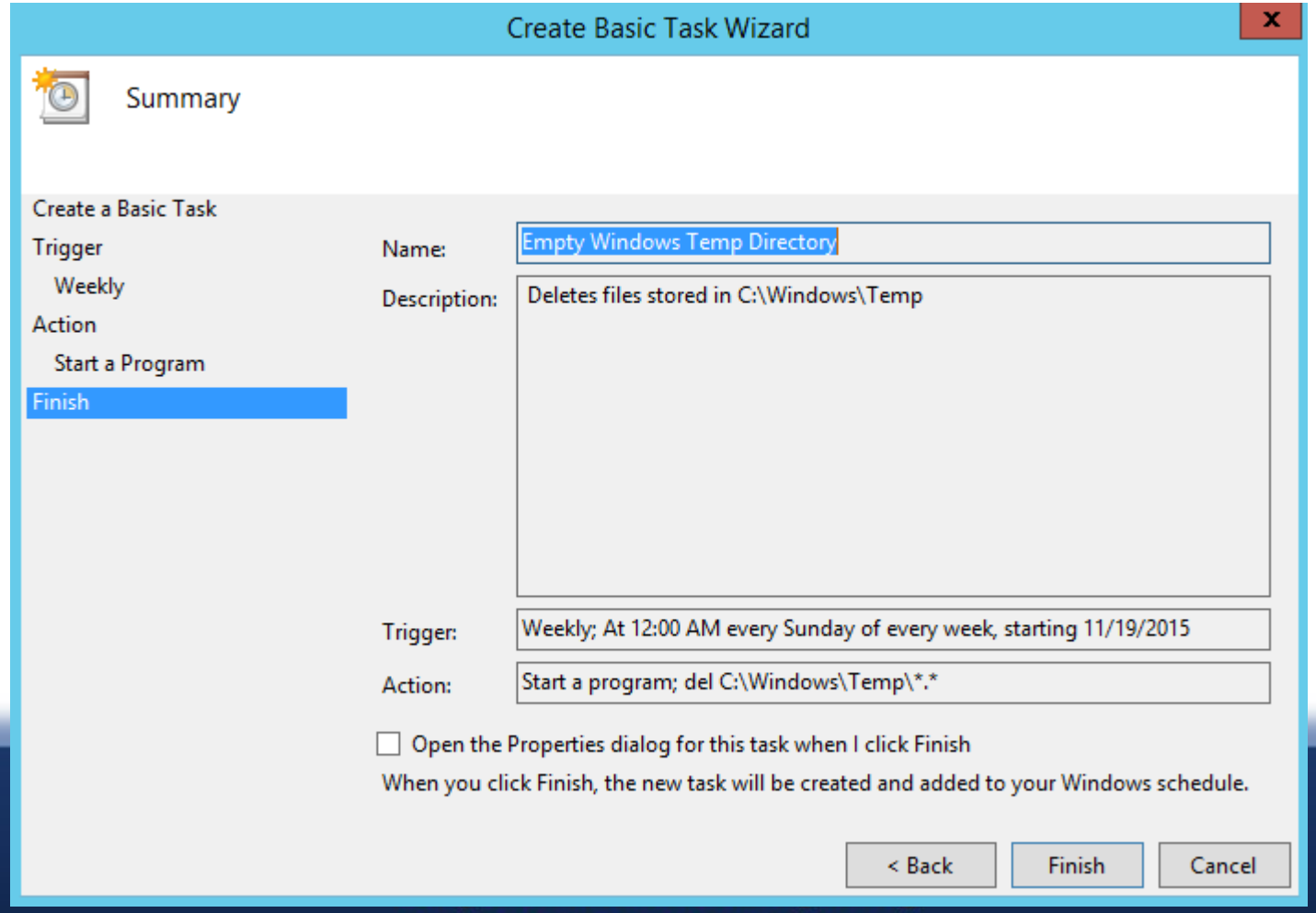
Windows Task Scheduler - New Task



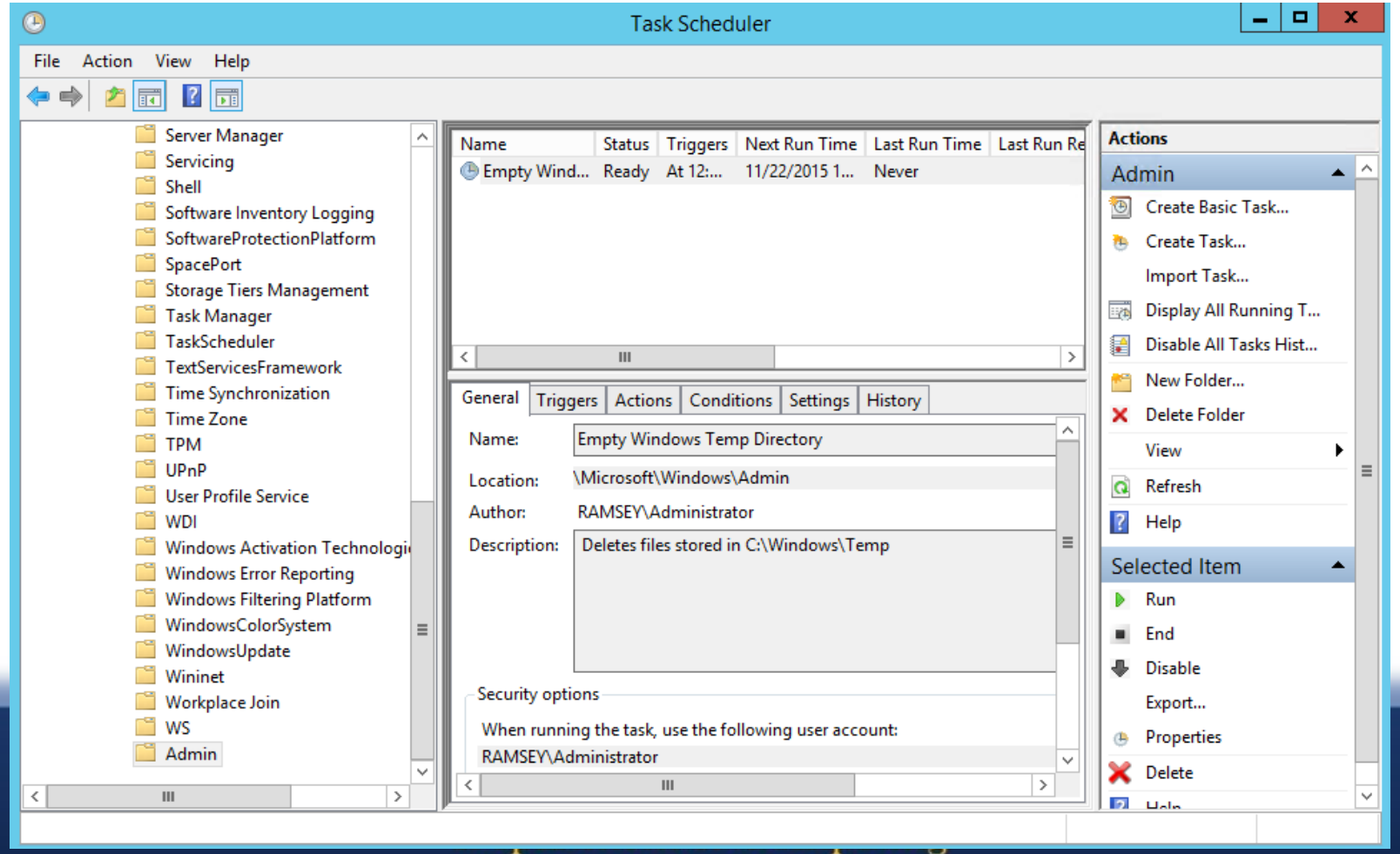
Windows Task Scheduler - New Task



Windows Task Scheduler - New Task



Windows Task Scheduler - New Task



IPTABLES

IPTABLES

Iptables is a standard firewall included in most Linux distributions by default

Can be used to create rules that block unwanted traffic

By default, Iptables rules do not persist across reboots (hence, the entry in **`crontab`** earlier. There are apt-get packages that can make rules persistent (e.g., **`iptables-persistent`**)

IPTABLES - View Rules

root@RamseyTemp: ~

```
root@RamseyTemp:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@RamseyTemp:~#
```

IPTABLES - View Rules

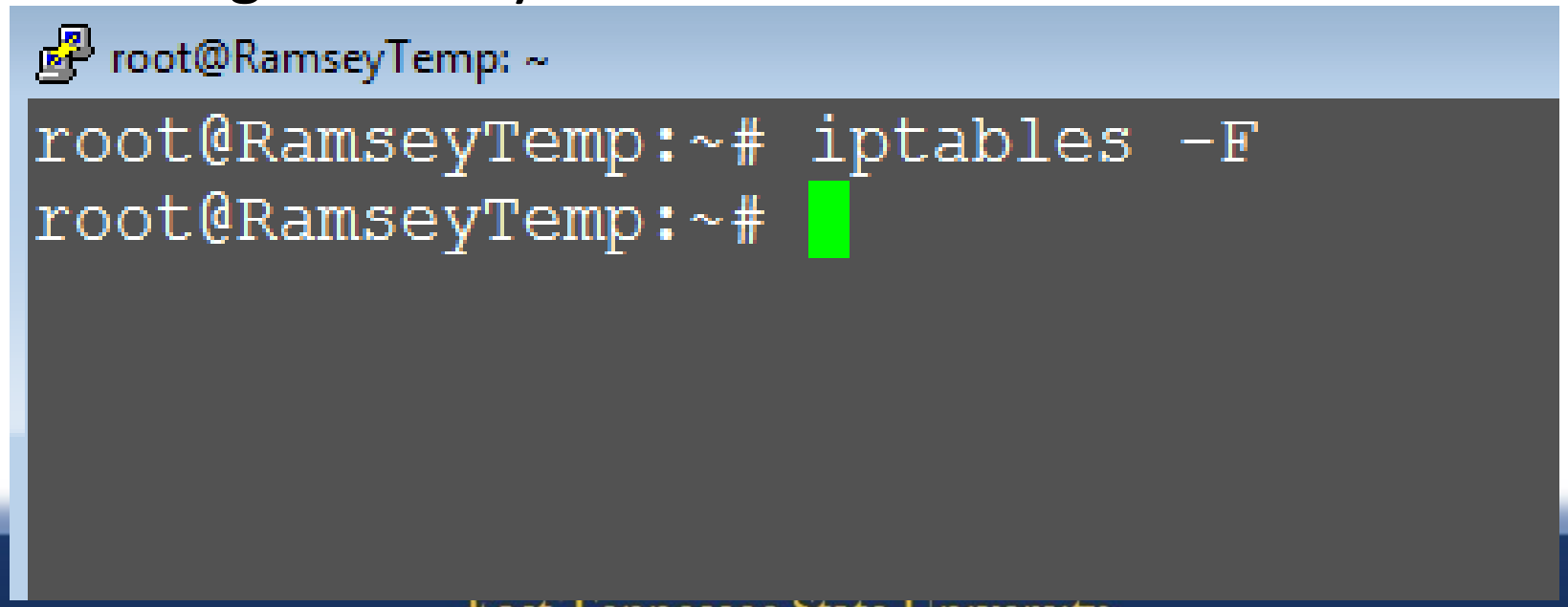
Can use the -S switch to display the basic syntax for rules. These are the default policies for INPUT, FORWARD, AND OUTPUT

root@RamseyTemp: ~

```
root@RamseyTemp:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
root@RamseyTemp:~#
```

IPTABLES - View Rules

If you DO already have rules in place and want to start over -- first make sure that the default INPUT & OUTPUT policies are set to ACCEPT if you're connecting remotely



IPTABLES - Make a Rule

First rule needed is:

```
root@RamseyTemp: ~
root@RamseyTemp:~# sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,REL
ATED -j ACCEPT
root@RamseyTemp:~#
```

-A Appends the rule to the end of the chain
-m conntrack Extension that provides extra capabilities
--ctstate Matches packets based on how they're related to packets seen before

IPTABLES - Make a Rule

First rule needed is:

```
root@RamseyTemp: ~
root@RamseyTemp:~# sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,REL
ATED -j ACCEPT
root@RamseyTemp:~#
```

-j ACCEPT Specifies the target of matching packets

IPTABLES - Make a Rule

```
root@RamseyTemp: ~
root@RamseyTemp:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@RamseyTemp:~#
```

IPTABLES - Other rules

```
root@RamseyTemp: ~
root@RamseyTemp:~# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
root@RamseyTemp:~#
```

-A Appends the rule to the end of the chain
INPUT The chain to add the rule to
-p Protocol
--dport 22 Port used by SSH

IPTABLES - Other rules

```
root@RamseyTemp: ~
root@RamseyTemp:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@RamseyTemp:~#
```

IPTABLES - Other rules

```
root@RamseyTemp: ~
root@RamseyTemp:~# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
root@RamseyTemp:~#
```

-A Appends the rule to the end of the chain
INPUT The chain to add the rule to
-p Protocol
--dport 22 Port used by SSH

IPTABLES - Other rules

```
root@RamseyTemp: ~
root@RamseyTemp:~# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
root@RamseyTemp:~#
```

-A Appends the rule to the end of the chain

INPUT The chain to add the rule to

-p Protocol

--dport 80 Port used by HTTP

IPTABLES - Other rules

```
root@RamseyTemp: ~
root@RamseyTemp:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@RamseyTemp:~#
```

IPTABLES - Implementing a Drop Rule

root@RamseyTemp: ~

```
root@RamseyTemp:~# iptables -A INPUT -j DROP
root@RamseyTemp:~#
```

-A	Appends the rule to the end of the chain
INPUT	The chain to add the rule to
-j DROP	Drops everything else

IPTABLES - Implementing a Drop Rule

root@RamseyTemp: ~

```
root@RamseyTemp:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@RamseyTemp:~#
```

Lab Next Week

So, we're going to join a Linux instance to the Windows domain we've already established

There are a number of steps:

- Launch instance (Amazon Linux or Ubuntu)
- Modify /etc/hosts file
- Modify hostname (different locations)
- Update / upgrade instance

Lab Next Week

So, we're going to join a Linux instance to the Windows domain we've already established

There are a number of steps:

- Install realmd
- Add additional necessary software packages
- Discover domain
- Join domain

Lab Next Week

So, we're going to join a Linux instance to the Windows domain we've already established

There are a number of steps:

- Modify sshd_config (authentication / password login)
- Add Domain Admins to sudoers
- Test authentication of Domain Admin

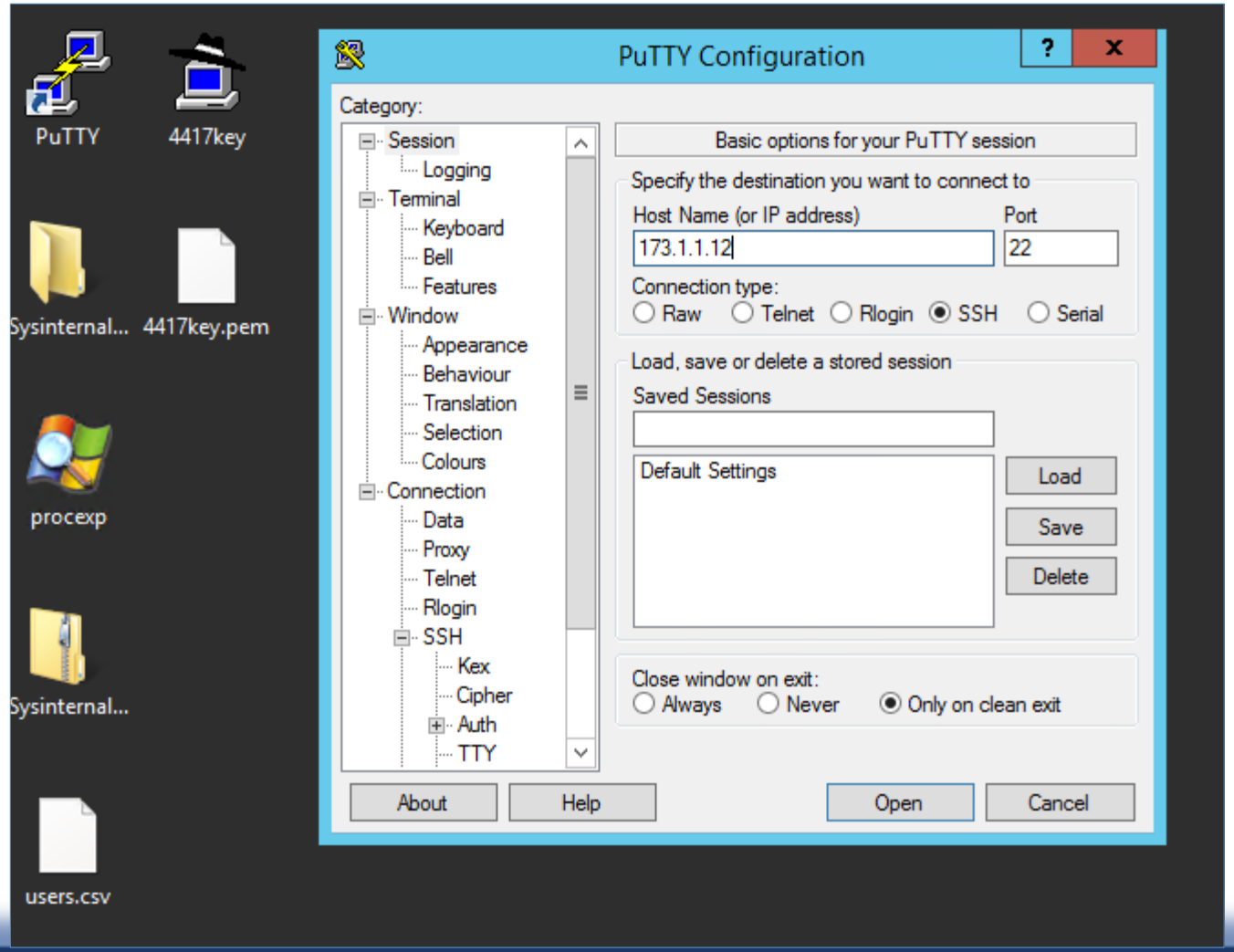
Lab Next Week

What we'll end up with is a Linux server that can be logged in to with accounts from the Domain Admin group (in addition to locally defined accounts, e.g., 'ubuntu' or 'ec2-user'

Lab Next Week

Launch instance (Amazon Linux or Ubuntu)

Log in from *lastname*DC01



Lab Next Week

ec2-user@ip-173-1-1-12:~

```
login as: ec2-user
Authenticating with public key "imported-openssh-key"

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2016.03-release-notes/
4 package(s) needed for security, out of 6 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-173-1-1-12 ~]$
```

Lab Next Week

Modify /etc/hosts file

(Add domain and domain controller info)

ec2-user@ip-173-1-1-12:~

```
  GNU nano 2.3.1                    File: /etc/hosts

127.0.0.1   localhost localhost.localdomain
173.1.1.20  ramsey.loc
173.1.1.20  ramseydc01.ramsey.loc
```

Lab Next Week

Modify hostname

/etc/sysconfig/network

```
ec2-user@ip-173-1-1-12:~
  GNU nano 2.3.1          File: /etc/sysconfig/network

NETWORKING=yes
HOSTNAME=ramseylinux02
NOZEROCONF=yes
NETWORKING_IPV6=no
IPV6INIT=no
IPV6_ROUTER=no
IPV6_AUTOCONF=no
IPV6FORWARDING=no
IPV6TO4INIT=no
IPV6_CONTROL_RADVD=no
```

Lab Next Wee

Install realmd

ec2-user@ramseylinux02:~

```
[ec2-user@ramseylinux02 ~]$ sudo yum install realmd
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
---> Package realmd.x86_64 0:0.16.1-5.5.amzn1 will be installed
--> Processing Dependency: adcli for package: realmd-0.16.1-5.5.amzn1.x86_64
--> Processing Dependency: samba-common for package: realmd-0.16.1-5.5.amzn1.x86
_64
--> Processing Dependency: sssd-common for package: realmd-0.16.1-5.5.amzn1.x86_
64
--> Processing Dependency: samba-winbind for package: realmd-0.16.1-5.5.amzn1.x8
6_64
--> Processing Dependency: oddjob-mkhomedir for package: realmd-0.16.1-5.5.amzn1
.x86_64
--> Processing Dependency: samba-winbind-clients for package: realmd-0.16.1-5.5.
amzn1.x86_64
--> Running transaction check
---> Package adcli.x86_64 0:0.7.5-4.4.amzn1 will be installed
--> Processing Dependency: cyrus-sasl-gssapi for package: adcli-0.7.5-4.4.amzn1.
x86_64
---> Package oddjob-mkhomedir.x86_64 0:0.31.5-4.17.amzn1 will be installed
--> Processing Dependency: oddjob = 0.31.5-4.17.amzn1 for package: oddjob-mkhome
dir-0.31.5-4.17.amzn1.x86_64
---> Package samba-common.noarch 0:4.2.10-6.33.amzn1 will be installed
```

Lab Next Weel

realm discover domain

ec2-user@ramseylinux02:~

```
[ec2-user@ramseylinux02 ~]$ sudo realm discover ramsey.loc
ramsey.loc
  type: kerberos
  realm-name: RAMSEY.LOC
  domain-name: ramsey.loc
  configured: no
  server-software: active-directory
  client-software: sssd
  required-package: oddjob
  required-package: oddjob-mkhomedir
  required-package: sssd
  required-package: adcli
  required-package: samba-common
[ec2-user@ramseylinux02 ~]$
```

Lab Next Week

Add additional packages

```
ec2-user@ramseylinux02:~
[ec2-user@ramseylinux02 ~]$ sudo nano /etc/ssh/sshd_config
[ec2-user@ramseylinux02 ~]$ sudo realm discover
realm: No default realm discovered
[ec2-user@ramseylinux02 ~]$ sudo realm discover ramsey.loc
ramsey.loc
  type: kerberos
  realm-name: RAMSEY.LOC
  domain-name: ramsey.loc
  configured: kerberos-member
  server-software: active-directory
  client-software: sssd
  required-package: oddjob
  required-package: oddjob-mkhomedir
  required-package: sssd
  required-package: adcli
  required-package: samba-common
  login-formats: %U@ramsey.loc
  login-policy: allow-realm-logins
[ec2-user@ramseylinux02 ~]$ sudo yum install oddjob oddjob-mkhomedir sssd adcli
samba-common
```

Lab Next Week

realm join domain

ec2-user@ramseylinux02:~

```
[ec2-user@ramseylinux02 ~]$ sudo realm join ramsey.loc
Password for Administrator:
[ec2-user@ramseylinux02 ~]$
```

Active Directory Users and Computers

File Action View Help

Active Directory Users and Com
- Saved Queries
- ramsey.loc
  - Builtin
  - Computers
  - DOMAIN - Groups
  - Domain Controllers
  - ForeignSecurityPrincipal
  - Managed Service Accou
  - Users

Name	Type	Description
RAMSEYLINUX01	Computer	
RAMSEYLINUX02	Computer	
RAMSEYMS01	Computer	
RAMSEYWINDOWSMS	Computer	
RAMSEYWINMS02	Computer	
RAMSEYWINMS04	Computer	
WIN-8QU6DCEQRGJ	Computer	

Lab Next Week

Modify /etc/ssh/sshd_config

Restert sshd when done

sudo service sshd restart

```
# Kerberos options
KerberosAuthentication yes
KerberosOrLocalPasswd yes
KerberosTicketCleanup yes
#KerberosGetAFSToken no
#KerberosUseKuserok yes

# GSSAPI options
GSSAPIAuthentication no
GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no
```

Lab Next Week

Add domain admins to sudoers

(not necessary, but gives sudo privileges to Domain Admins)

```
ec2-user@ramseylinux02:~
  GNU nano 2.5.3              File: /etc/sudoers

# %wheel        ALL=(ALL)       NOPASSWD: ALL

## Allows members of the users group to mount and unmo
## cdrom as root
# %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mn

## Allows members of the users group to shutdown this
# %users  localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here
#includedir /etc/sudoers.d

# Add ramsey.loc Domain Admins
%Domain\ Admins@ramsey.loc      ALL=(ALL:ALL) ALL
```

Questions?