

System Administration 1

Prepared by Dr. Ahmed Omar

IT391

System Administration 1

Lecture 10



Managing Linux Groups & Controlling File Permissions

Lecture 10

IT391

System Administration 1

Prepared by
Dr. Ahmed Omar



Agenda

- ☐ Linux Groups
- ☐ Linux File Permission Control
- ☐ Monitoring and Managing Processes
- ☐ Disaster Recovery & Backup Strategies
- ☐ The Power of Automation

Linux Group

A **group** in Linux is a collection of user accounts that share common permissions and access rights to files, directories, and system resources.

- ❑ **Groups** provide an efficient way to manage permissions for multiple users without having to set individual permissions for each user.

Linux Group

Primary vs Secondary Groups

- ☐ Each user has one **primary group** (usually created automatically when the user account is made)
- ☐ Users can belong to multiple **secondary groups** for additional permissions

Linux Group

Group Information Storage:

Group definitions are stored in
/etc/group

cat /etc/group

```
ahmed@debiansrv1:~$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
Debian-gdm:x:121:
ahmed:x:1000:
gnome-initial-setup:x:995:
ali:x:1001:
omar:x:1002:
user1:x:1003:
tcpdump:x:122:
```

Create and manage groups

sudo groupadd [options] **groupname**

- ☐ **-f** exit successfully if the group already exists,
- ☐ **-g** use GID for the new group
- ☐ **-h** display this help message and exit
- ☐ **-K KEY=VALUE** override **/etc/login.defs** defaults
- ☐ **-o** allow to create groups with duplicate (non-unique) GID
- ☐ **-p PASSWORD** use this encrypted password for the new group
- ☐ **-r** create a system account
- ☐ **-R CHROOT_DIR** directory to chroot into
- ☐ **-P, PREFIX_DI** directory prefix
- ☐ **-U** list of user members of this group

Create and manage groups

Group GID

- **GID 0: root group**
- **System Groups (0-999)**
- **User Groups (1000+)**

```
ahmed@debiansrv1:~$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
Debian-gdm:x:121:
ahmed:x:1000:
gnome-initial-setup:x:995:
ali:x:1001:
omar:x:1002:
user1:x:1003:
tcpdump:x:122:
```

Create and manage groups

```
sudo groupadd developers
```

```
ahmed@debiansrv1:~$ sudo groupadd developers
ahmed@debiansrv1:~$ tail -5 /etc/group
ali:x:1001:
omar:x:1002:
user1:x:1003:
tcpdump:x:122:
developers:x:1004:
ahmed@debiansrv1:~$ █
```

Create and manage groups

Creating Groups with Specific GIDs

```
sudo groupadd -g 2000 sysadmin
```

```
ahmed@debiansrv1:~$ sudo groupadd -g 2000 sysadmin
ahmed@debiansrv1:~$ tail -5 /etc/group
omar:x:1002:
user1:x:1003:
tcpdump:x:122:
developers:x:1004:
sysadmin:x:2000:
ahmed@debiansrv1:~$
```

Create and manage groups

When you create a new group without specifying a GID, Linux automatically assigns the next available GID by continuing from the **highest** currently used GID within the allowed range.

```
ahmed@debiansrv1:~$ sudo groupadd netadmin
ahmed@debiansrv1:~$ tail -5 /etc/group
user1:x:1003:
tcpdump:x:122:
developers:x:1004:
sysadmin:x:2000:
netadmin:x:2001:
ahmed@debiansrv1:~$
```

Create and manage groups

Add user to group

```
sudo usermod -aG netadmin ahmed
```

Remove user from group

```
sudo gpasswd -d netadmin ahmed
```

```
ahmed@debiansrv1:~$ sudo usermod -aG netadmin ahmed
[sudo] password for ahmed:
ahmed@debiansrv1:~$ tail -5 /etc/group
user1:x:1003:
tcpdump:x:122:
developers:x:1004:
sysadmin:x:2000:
netadmin:x:2001:ahmed
ahmed@debiansrv1:~$
```

Create and manage groups

Change primary group

The **newgrp** command allows a user to temporarily switch their primary group (effective group) without logging out. This is useful when you need to create files with a different group ownership.

When you run **newgrp**, it:

- ✓ Starts a new shell session
- ✓ Changes your effective group ID to the specified group
- ✓ New files/directories will be owned by this group
- ✓ You must be a member of the target group

Create and manage groups

Change primary group

Basic Syntax : `newgrp [groupname]`

```
ahmed@debiansrv1:~$ newgrp sysadmin
ahmed@debiansrv1:~$ id
uid=1000(ahmed) gid=2000(sysadmin) groups=2000(sysadmin),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),100(users),106(netdev),111(bluetooth),113(lpadmin),116(scanner),1000(ahmed)
ahmed@debiansrv1:~$ █
```

```
ahmed@debiansrv1:~$ ls -l file1
-rw-r--r-- 1 ahmed ahmed 0 Aug 11 10:24 file1
ahmed@debiansrv1:~$ touch file2
ahmed@debiansrv1:~$ ls -l file2
-rw-r--r-- 1 ahmed sysadmin 0 Aug 11 10:37 file2
ahmed@debiansrv1:~$
```

Create and manage groups

Change group name

Linux doesn't have a direct "rename group" command, but you can effectively rename a group using the **groupmod** command to change the group name.

sudo groupmod -n new_name old_name

```
ahmed@debiansrv1:~$ tail -5 /etc/group
user1:x:1003:
tcpdump:x:122:
developers:x:1004:
sysadmin:x:2000:ahmed
netadmin:x:2001:ahmed
ahmed@debiansrv1:~$ sudo groupmod -n sysdev developers
ahmed@debiansrv1:~$ tail -5 /etc/group
user1:x:1003:
tcpdump:x:122:
sysadmin:x:2000:ahmed
netadmin:x:2001:ahmed
sysdev:x:1004:
ahmed@debiansrv1:~$
```

Create and manage groups

Change group id

You can change a group's GID using the ***groupmod*** command with the -g option.

```
sudo groupmod -g new_gid groupname
```

```
ahmed@debiansrv1:~$ tail -5 /etc/group
user1:x:1003:
tcpdump:x:122:
sysadmin:x:2000:ahmed
netadmin:x:2001:ahmed
sysdev:x:1004:
```

```
ahmed@debiansrv1:~$ sudo groupmod -g 1006 sysadmin
ahmed@debiansrv1:~$ tail -5 /etc/group
user1:x:1003:
tcpdump:x:122:
sysadmin:x:1006:ahmed
netadmin:x:2001:ahmed
sysdev:x:1004:
```

Create and manage groups

Removing Groups

You can remove (delete) groups using the *groupdel* command.

`sudo groupdel groupname`

```
ahmed@debiansrv1:~$ sudo groupdel sysdev
ahmed@debiansrv1:~$ tail -5 /etc/group
omar:x:1002:
user1:x:1003:
tcpdump:x:122:
sysadmin:x:1006:ahmed
netadmin:x:2001:ahmed
ahmed@debiansrv1:~$
```

File Permission Control

Default File Permission Assignment

File Owner and Group:

❑ **Owner (User):** The user who created the file

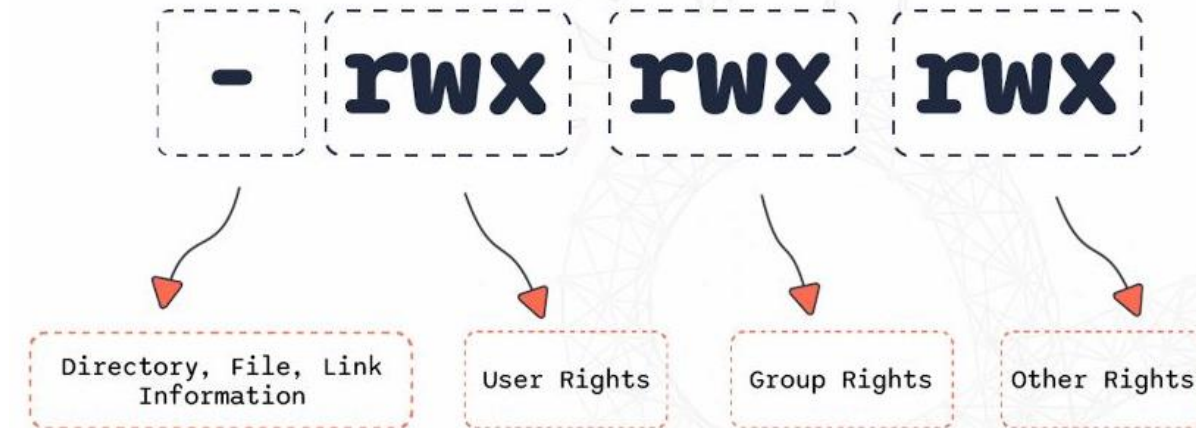
rw-: Owner (ahmed) permissions - read and write, no execute

❑ **Group:** The user's primary group (from /etc/passwd)

r--: Group (ahmed) permissions - read only

❑ **Others:** Everyone else on the system

r--: Others permissions - read only



```
ahmed@debiansrv1:~$ ls
Desktop  dir1  dir3  Documents  Downloads  Music  Pictures
ahmed@debiansrv1:~$ touch file1
ahmed@debiansrv1:~$ ls -l file1
-rw-r--r-- 1 ahmed ahmed 0 Aug 11 10:24 file1
ahmed@debiansrv1:~$
```

File Permission Control

- ❑ **Purpose** : Secure files/directories by modifying access rights for users, groups, and others

Permission Basics

▪ Three Entities:

- ❑ **u** → Owner
- ❑ **g** → Group
- ❑ **o** → Others
- ❑ **a** → all (user + group + others)

▪ Three Permissions:

- ❑ **r** = Read (4)
- ❑ **w** = Write (2)
- ❑ **x** = Execute (1)

`-rw-r--r-- 1 user group 4096 Aug 11 file.txt`

- ✓ Owner: **read+write**
- ✓ Group: **read**
- ✓ Others: **read**

File Permission Control

❑ What is chmod?

❑ **chmod** stands for **Change Mode** — it's the command used in Linux to set file and directory permissions.

❑ Symbolic Mode

- + Adds the given permission(s) without changing existing ones.
- Removes the given permission(s) but leaves others unchanged.
- = Replaces all permissions for the category with exactly what you specify.
- **chmod u+x file.txt** # Add execute for user
- **chmod g-w file.txt** # Remove write for group
- **chmod o=r file.txt** # Set read only for others
- **chmod a+rw file.txt** # Add read/write for everyone

File Permission Control

chmod

☐ Numeric Mode

Uses numbers:

r = 4, w = 2, x = 1, - = 0 Sum them to get permission values.

chmod 754 file.txt

- **7** → user: 4+2+1 = rwx
- **5** → group: 4+0+1 = r-x
- **4** → others: 4+0+0 = r--

Users Passwords

Show users passwords

- ❑ user passwords are not stored in plaintext for security reasons. Instead, they are stored as encrypted **hashes** in a protected file. You cannot "show" the actual passwords, only their hashed versions.
- ❑ Hashed passwords are stored in */etc/shadow*, which is readable only by root (or with sudo privileges)

`sudo cat /etc/shadow`

```
ahmed@debiansrv1:~$ sudo cat /etc/shadow
root:$y$j9T$5cVb9FnGjYNsbmly0sm93/$QylIoo0n/eCxqeFFaaw/TC3TgZZe2F65qEeaWB0g8T1:20290:0:99999:7:::
daemon:!:20290:0:99999:7:::
ahmed:$y$j9T$aXNiJKojofhMtZlS9Xmv/0$KJ6Za1b/o2/k62mqEgQgcBYMIQn8ydPHpssVaC1SKEA:20290:0:99999:7:::
ali:$y$j9T$XpNTRxA00.JXAEnGuqP301$qBx25qSiUnLDHqfMfrA4/fG86qnumeH1k20o/gj0S.0:20305:0:99999:7:::
omar:$y$j9T$zdzts3wEflrcdCYsRYkTo1$LpQcsuFB6SM0IvdgZpmEwSSBJXnGml1CRFz35sQ9Vd1:20305:0:99999:7:::
user1:$y$j9T$zNfUfoNJJ0B7qaRopUYUh0$efzGyXaKdE18HL0wPR.7.p0uFDTA9u3nFCV4hTqPVj.:20305:0:99999:7:::
```

Users Passwords

- **ahmed:\$y\$j9T\$aXNiJKojofhMtZlS9Xmv/0\$KJ6Zalb/o2/k62mqEgQgcBYMIQn8ydPHpssVaC1SKEA:20290:0:99999:7:::**

- ❑ Username: **ahmed**
- ❑ Hash Type: **\$y\$** = **yescrypt** algorithm (modern, secure)
- ❑ Cost Parameter: **j9T** (computational cost for yescrypt)
- ❑ Salt: **aXNiJKojofhMtZlS9Xmv/0** (unique random salt)
- ❑ Hash: **KJ6Zalb/o2/k62mqEgQgcBYMIQn8ydPHpssVaC1SKEA** (password hash)
- ❑ Last Password Change: **20290** - Days since January 1, 1970
- ❑ Minimum Days: **0** - User can change password immediately (no waiting period)
- ❑ Maximum Days: **99999** - Password effectively never expires
- ❑ Warning Days: **7** -User gets warned 7 days before password expiration

Configure Password Aging

Password aging is a way to control how long a user's password remains valid before they must change it. It's part of security best practices, preventing users from keeping the same password forever.

With password aging, you can:

- ☐ Set the **minimum number of days** before a user can change their password.
- ☐ Set the **maximum number of days** a password is valid.
- ☐ Set a **warning period** before expiration.
- ☐ Force password expiration immediately.

Configure Password Aging

☐ View a user's password aging info

`sudo chage -l username`

```
ahmed@debiansrv1:~$ sudo chage -l ahmed
[sudo] password for ahmed:
Last password change                : Jul 21, 2025
Password expires                     : never
Password inactive                    : never
Account expires                     : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
ahmed@debiansrv1:~$
```

Configure Password Aging

- ❑ Set maximum and minimum password days

`sudo chage -M 60 -m 5 omar`

- ❑ Set a warning before expiration

`sudo chage -W 7 omar`

Expire password immediately (forces change on next login)

`sudo chage -d 0 omar`

```
ahmed@debiansrv1:~$ sudo chage -M 60 -m 5 omar
ahmed@debiansrv1:~$ sudo chage -l omar
Last password change                : Aug 05, 2025
Password expires                    : Oct 04, 2025
Password inactive                   : never
Account expires                    : never
Minimum number of days between password change : 5
Maximum number of days between password change : 60
Number of days of warning before password expires : 7
```

Configure Password Aging

☐ Account Expiration

- ☐ Unlike password aging, account expiration means the entire account becomes inactive after a certain date — the user can't log in at all, even with a correct password.

`sudo chage -E 2025-12-31 ali`

```
ahmed@debiansrv1:~$ sudo chage -E 2025-12-31 ali
[sudo] password for ahmed:
ahmed@debiansrv1:~$ sudo chage -l ali
Last password change                : Aug 05, 2025
Password expires                    : never
Password inactive                    : never
Account expires                     : Dec 31, 2025
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
ahmed@debiansrv1:~$
```

Change User or Group Ownership

To change user or group ownership of files and directories in Linux, the primary commands is **chown** (change owner)

chown [user][:group] file_or_directory

chown -R user:group directory

```
ahmed@debiansrv1:~$ ls -l file1
-rw-r----- 1 ahmed ahmed 0 Aug 11 10:24 file1
ahmed@debiansrv1:~$ sudo chown admin1 file1
ahmed@debiansrv1:~$ ls -l file1
-rw-r----- 1 admin1 ahmed 0 Aug 11 10:24 file1
ahmed@debiansrv1:~$ sudo chown :admins file1
ahmed@debiansrv1:~$ ls -l file1
-rw-r----- 1 admin1 admins 0 Aug 11 10:24 file1
ahmed@debiansrv1:~$
```

Monitoring and Managing Processes

What is a Process in Linux?

A process is simply a **running instance of a program**.

Each process has:

- ☐ **PID** (Process ID) — unique identifier.
- ☐ **State** (running, sleeping, stopped, zombie).
- ☐ **Owner** (user who started it).
- ☐ **Priority** and **niceness** value.
- ☐ **Resource usage** (CPU, memory, I/O).

Monitoring and Managing Processes

The **ps** a command in Linux lists all running processes associated with terminals

This command output typically includes columns such as:

- **PID:** Process ID
- **TTY:** Terminal associated with the process
- **STAT:** Process state (e.g., S for sleeping, R for running)
- **TIME:** CPU time used by the process
- **COMMAND:** The command that started the process

Command	Description
ps a	Lists processes with terminals
ps u	Lists processes with user details
ps x	Lists processes without terminals
ps aux	Lists all processes with details

Monitoring and Managing Processes

```
ahmed@debiansrv1:~$ ps a
```

PID	TTY	STAT	TIME	COMMAND
1496	tty2	Ssl+	0:00	/usr/libexec/gdm-wayland-session /usr/bin/gnome-session
1517	tty2	Sl+	0:00	/usr/libexec/gnome-session-binary
2224	pts/0	Ss	0:00	bash
2462	pts/0	S	0:00	su - omar
2463	pts/0	S	0:00	-bash
2519	pts/0	S	0:00	su - ahmed
2521	pts/0	S	0:00	-bash
2739	pts/0	R+	0:00	ps a

```
ahmed@debiansrv1:~$ echo $$
```

```
2521
```

```
ahmed@debiansrv1:~$ ps au
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
ahmed	1496	0.0	0.3	159288	7740	tty2	Ssl+	22:15	0:00	/usr/libexec/gdm-wayland-session
ahmed	1517	0.0	1.0	297964	20340	tty2	Sl+	22:15	0:00	/usr/libexec/gnome-session-binar
ahmed	2224	0.0	0.2	8008	4872	pts/0	Ss	22:15	0:00	bash
root	2462	0.0	0.1	9072	3868	pts/0	S	22:23	0:00	su - omar
omar	2463	0.0	0.2	8104	5012	pts/0	S	22:23	0:00	-bash
root	2519	0.0	0.2	9072	3976	pts/0	S	22:38	0:00	su - ahmed
ahmed	2521	0.0	0.2	7972	4892	pts/0	S	22:38	0:00	-bash
ahmed	2742	500	0.2	11220	4852	pts/0	R+	23:24	0:00	ps au

```
ahmed@debiansrv1:~$
```

Monitoring and Managing Processes

Controlling Processes

kill -9 1234	# Force kill by PID
pkill -f "pattern"	# Kill by process name/pattern
kill -STOP 1234	# Pause process
kill -CONT 1234	# Resume process

CPU Usage Monitoring

Consistent high CPU usage can lead to performance degradation. Several command-line tools offer insights into CPU load.

- ❑ **top**: This ubiquitous tool provides a real-time, dynamic view of the processes running on a system. It displays a summary of system information, including load averages and a list of the most CPU-intensive tasks. You can press '1' within top to see statistics for individual CPU cores.

```
top - 00:58:14 up 1:49, 1 user, load average: 0.04, 0.08, 0.04
Tasks: 267 total, 1 running, 266 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.1 us, 1.7 sy, 0.0 ni, 94.9 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st
MiB Mem : 1932.4 total, 118.8 free, 1230.3 used, 757.4 buff/cache
MiB Swap: 975.0 total, 975.0 free, 0.0 used. 702.0 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1606	ahmed	20	0	3780228	308664	133460	S	10.9	15.6	1:42.86	gnome-shell
2193	ahmed	20	0	561612	54520	41596	S	2.3	2.8	0:11.33	gnome-terminal-
2944	ahmed	20	0	11604	5456	3320	R	1.0	0.3	0:01.54	top
15	root	20	0	0	0	0	I	0.3	0.0	0:03.66	rcu_preempt
451	root	20	0	241172	10816	7252	S	0.3	0.5	0:20.12	vmtoolsd
1743	ahmed	20	0	312252	11896	6736	S	0.3	0.6	0:03.96	ibus-daemon
2761	root	20	0	0	0	0	I	0.3	0.0	0:00.37	kworker/0:2-events

Failure Types

- ☐ **Hardware:** Disk failures, RAM errors, power supply issues
- ☐ **Software:** OS corruption, file system errors, bad updates
- ☐ **Human:** Accidental file deletion, improper commands (rm -rf)
- ☐ **Natural:** Fire, flood, power surges, physical site damage

Disaster Recovery & Backup

Backup Types

- ❑ **Full Backup** Complete copy of all selected data, creates independent backup sets, slower but simplest to restore.
- ❑ **Incremental** Only backs up changes since last backup (any type), fastest & smallest, but complex restore
- ❑ **Snapshot** Point-in-time image of file system state, fast creation, uses features like LVM, Btrfs, ZFS

Backup Strategy Example:

Sunday
Full Backup

Monday-Friday
Incremental

Monthly
Archive Full

Critical Changes
Snapshots

Disaster Recovery & Backup

❑ Common Linux Backup Tools

- **tar** → Archive & compress files
- **rsync** → Sync files/directories
- **dd** → Bit-by-bit copy (disk cloning)
- **cp** → Simple copy
- **Timeshift** → System snapshot for desktops

Disaster Recovery & Backup

❏ `tar -cf backup.tar.gz /home/user`

- c → create archive
- f → filename
- Restore : `tar -xzf backup.tar.gz -C /`

The Power of Automation

Automation in system management refers to the process of creating systems and software that reduce or eliminate the need for human intervention in IT services.

By automating routine tasks, system administrators can:

- ☐ Reduce human error and increase reliability
- ☐ Improve efficiency and response times
- ☐ Scale operations without proportional staff increases
- ☐ Focus on strategic initiatives rather than routine

Key Benefit: Time Savings

- ✓ Studies show that automation can reduce system administration workload by up to 70% for routine tasks.

The Power of Automation

The Linux **at** command is used to schedule one-time tasks or jobs to be executed at a specific time in the future. Unlike cron which is for recurring jobs, at is designed for single execution at a given time.

The general syntax is: **at [options] time**

After entering this command, you will get an interactive prompt (**at>**) to enter the commands you want to execute at the scheduled time. Press **Ctrl+D** to finish and save the job.

Not you need to install at package:

sudo apt update

sudo apt install at

```
ahmed@debiansrv1:~$ at now+2 minutes
warning: commands will be executed using /bin/sh
at Tue Aug 12 20:33:00 2025
at> touch myfile
at> mkdir mydir
at> <EOT>
job 3 at Tue Aug 12 20:33:00 2025
ahmed@debiansrv1:~$
```

Infrastructure as Code (IaC)

Automating Configuration with Ansible

- ▶ **Concept:** Replace manual SSH loops with desired state definitions in code files.
- ▶ **Tool:** **Ansible** agentless and uses simple YAML "Playbooks".
- ▶ **Consistency:** Eliminates "configuration drift" ensuring all servers are identical.
- ▶ **Scalability:** Configure 1,000 servers as easily as 1.
- ▶ **Version Control:** Track infrastructure changes with Git.



Containerization

Beyond Virtual Machines

Concept: While VMs virtualize hardware, Containers virtualize the OS, packaging apps with dependencies.

Advantage: Incredibly lightweight and fast. Starts in seconds, not minutes.

Relevance: Admins focus on container runtimes rather than full OS installs.



Container Orchestration (Kubernetes)

The challenge is managing hundreds of containers across multiple servers,
scaling, and management.

auto **K8s** deployment,



Self-healing

Automatically restarts failed containers
or kills those that don't respond to health
checks.



Auto-scaling

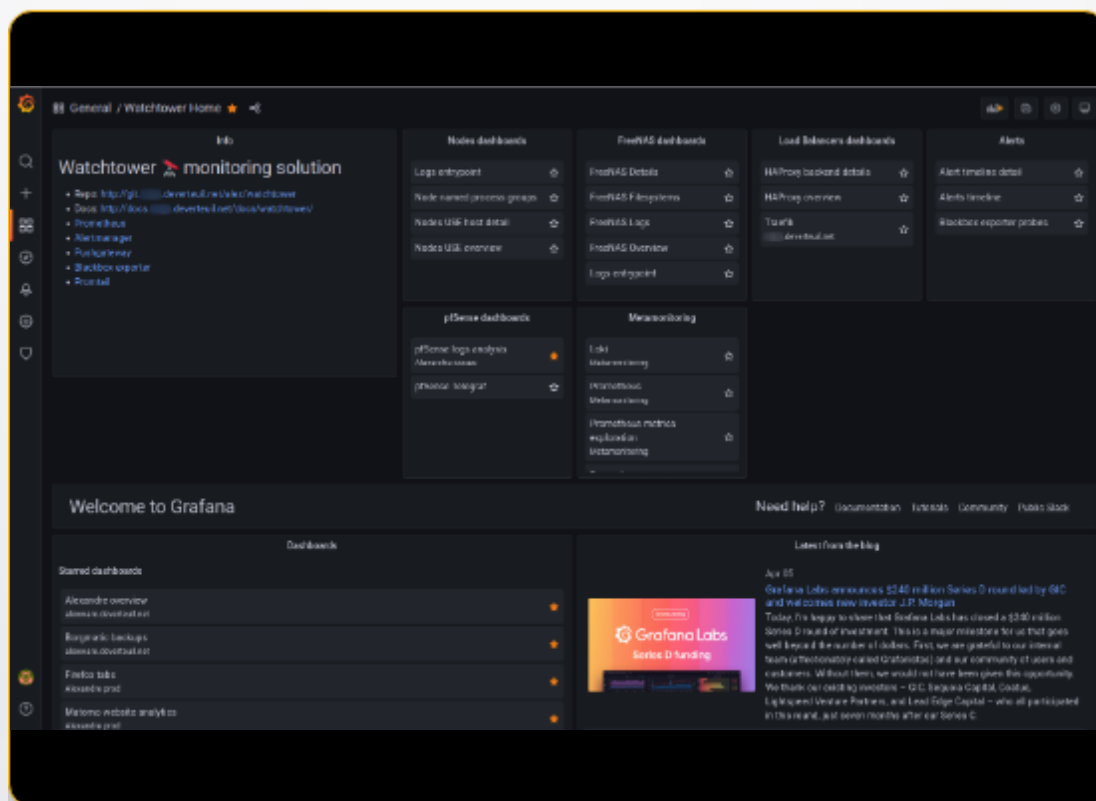
Automatically adds more containers
during traffic spikes to handle load
seamlessly.



Load Balancing

Intelligently distributes network traffic to
keep the deployment stable and
responsive.

Modern Observability



Prometheus & Grafana

- ▶ **Evolution:** Moving from "Is it down?" (Monitoring) to "Why is it down?" (Observability).
- ▶ **Collects** metrics like CPU, memory, and latency at specific intervals.
- ▶ **Visualizes** those metrics into interactive, real-time dashboards.
- ▶ **Impact:** Predict failures before they happen and view infrastructure health on a single screen.

Summary

- ☐ Linux Groups
- ☐ Linux File Permission Control
- ☐ Monitoring and Managing Processes
- ☐ Disaster Recovery & Backup Strategies
- ☐ The Power of Automation



Thank You