

IMPLANTACIÓN DE SISTEMAS

JUEVES 24 DE JULIO DE 2025

OBEJETIVOS DE LA CLASE

- 01** Identificar los principales beneficios de las prácticas modernas de integración y despliegue.
- 02** Reconocer los retos más comunes en entornos de desarrollo actuales.
- 03** Comprender la importancia de la terminal en el flujo de trabajo.
- 04** Distinguir entre comandos básicos y avanzados de terminal.

1.2 BENEFICIOS Y RETOS EN ENTORNOS MODERNOS

BENEFICIOS EN ENTORNOS MODERNOS

Velocidad de Entrega

- **Despliegues frecuentes:** De meses a minutos.
- **Feedback rápido:** Detección rápida de problemas.
- **Time-to-market reducido:** Ventaja competitiva.

Calidad del Software

- **Automatización de pruebas:** Menos errores humanos.
- **Rollback rápido:** Vuelta atrás inmediata si hay problemas.
- **Entornos consistentes:** "Funciona en mi máquina" se elimina.

Colaboración Mejorada

- **Cultura DevOps:** Desarrollo y operaciones trabajando juntos.
- **Visibilidad:** Todo el equipo ve el estado del proyecto.
- **Responsabilidad compartida.**

Costos Reducidos

- **Menos tiempo manual:** Automatización de tareas repetitivas.
- **Detección temprana:** Bugs encontrados temprano.
- **Optimización de recursos:** Uso eficiente de infraestructura.

RETOS EN ENTORNOS MODERNOS

Complejidad Técnica

- **Curva de aprendizaje:** Nuevas herramientas y conceptos.
- **Infraestructura como código:** Gestión de infraestructura programática.
- **Complejidad de los microservicios:** Gestión de múltiples servicios.

Cultura organizacional

- **Resistencia al cambio:** Equipos acostumbrados a métodos tradicionales.
- **Brechas de habilidades:** Necesidad de capacitación constante.
- **Barreras de comunicación:** Entre equipos técnicos y de negocio.

Seguridad

- **Seguridad a toda velocidad:** Mantener seguridad con despliegues rápidos.
- **Seguridad de los contenedores:** Nuevas superficies de ataque.
- **Gestión secreta:** Gestión segura de credenciales.

Monitoreo y Observabilidad

- **Sistema distribuido:** Dificultad para debuggear sistemas distribuidos.
- **Fatiga por alerta:** Demasiadas alertas pueden ser contraproducentes.
- **Control del rendimiento:** Seguimiento en tiempo real.


2.0 MANEJO DE LA TERMINAL



¿Por que es importante la terminal?

“El uso de la terminal es esencial para automatizar tareas, ejecutar scripts, y administrar sistemas de forma eficiente, especialmente en entornos donde la velocidad y la confiabilidad son clave.”

The DevOps Handbook (2ª edición).



```
This example of
Single::ToString< >,
Single::ToString< String* >,
Single::ToString< IFormatProvider* >, and
Single::ToString< String*, IFormatProvider* >
generates the following output when run in the [en-US] culture.
A Single number is formatted with various combinations of format
strings and IFormatProvider.

IFormatProvider is not used; the default culture is [en-US]:
No format string: 11876.54
'N5' format string: 11.876.54000
'E' format string: 1.187654E+004
'E5' format string: 1.18765E+004

A CultureInfo object for [nl-NL] is used for the IFormatProvider:
No format string: 11876.54
'N5' format string: 11.876.54000
'E' format string: 1.187654E+004

A NumberFormatInfo object with digit group size = 2 and
digit separator = ',' is used for the IFormatProvider:
'N' format string: 1_18_76.54
'E' format string: 1.187654E+004
Press any key to continue . . .
```

RAZONES PRINCIPALES

1

EFICIENCIA

Comandos rápidos vs interfaces gráficas lentas.

2

AUTOMATIZACIÓN

Scripts y flujo de trabajo secuencial y automatizado.

3

SERVIDORES

La mayoría no tienen interfaz gráfica.

4

DEVOPS TOOLS

Muchas herramientas son CLI-first.

5

DEBUGGING

Acceso directo al sistema.

2.1 COMANDOS BÁSICOS Y AVANZADOS

Comandos de navegación

BÁSICOS

```
bash

pwd          # Print Working Directory
ls           # Listar archivos
ls -la       # Lista detallada con archivos ocultos
cd           # Change Directory
cd ~         # Ir al home
cd ..        # Subir un nivel
cd -         # Volver al directorio anterior
```

AVANZADOS

```
bash

find . -name "*.js" # Buscar archivos JavaScript
locate filename     # Buscar archivos en el sistema
which node           # Ubicación de un comando
whereis python       # Ubicaciones relacionadas con python
```

Comandos de manipulación de archivos

BÁSICOS

```
bash

touch file.txt           # Crear archivo vacío
mkdir proyecto           # Crear directorio
mkdir -p proyecto/src/js # Crear directorios anidados
cp file1.txt file2.txt   # Copiar archivo
mv oldname.txt newname.txt # Mover/renombrar
rm file.txt              # Eliminar archivo
rm -rf directory         # Eliminar directorio recursivamente
```

AVANZADOS

```
bash

rsync -av source/ dest/ # Sincronización avanzada
tar -czf backup.tar.gz dir/ # Crear archivo comprimido
tar -xzf backup.tar.gz    # Extraer archivo
chmod +x script.sh        # Dar permisos de ejecución
chown user:group file.txt  # Cambiar propietario
```

Comandos de visualización de contenido

BÁSICOS

```
bash

cat file.txt           # Mostrar contenido completo
less file.txt          # Ver archivo paginado
head -n 10 file.txt    # Primeras 10 líneas
tail -n 10 file.txt    # Últimas 10 líneas
tail -f logfile.log    # Seguir archivo en tiempo real
```

AVANZADOS

```
bash

grep "error" logfile.log # Buscar texto en archivo
grep -r "TODO" src/      # Buscar recursivamente
sed 's/old/new/g' file.txt # Reemplazar texto
awk '{print $1}' file.txt  # Procesar columnas
```

Comandos de sistema

BÁSICOS

```
bash

ps aux           # Procesos en ejecución
top              # Monitor de procesos
kill PID         # Terminar proceso
killall process_name # Terminar por nombre
df -h            # Espacio en disco
du -sh directory/ # Tamaño de directorio
```

AVANZADOS

```
bash

htop             # Monitor avanzado
netstat -tulpn   # Conexiones de red
ss -tulpn        # Conexiones (moderno)
lsof -i :8080     # Qué usa el puerto 8080
nohup command &  # Ejecutar en background
```

2.2 NAVEGACIÓN DE SISTEMAS DE ARCHIVOS Y EJECUCIÓN DE SCRIPTS

ESTRUCTURA DE DIRECTORIOS LINUX/UNIX

```
/
├── /home    # Directorios de usuarios
├── /var     # Archivos variables
├── /etc     # Archivos de configuración
├── /usr     # Programas de usuario
├── /opt     # Software opcional
├── /tmp     # Archivos temporales
└── /bin     # Binarios esenciales
```

TIPOS DE SCRIPTS

```
bash

# Script de Bash
./script.sh
bash script.sh

# Script de Python
python script.py
python3 script.py

# Script de Node.js
node script.js

# Hacer ejecutable
chmod +x script.sh
./script.sh
```

VARIABLES DE ENTORNO

```
bash

# Ver variables
env
echo $PATH
echo $HOME

# Establecer variables
export API_KEY="your_key_here"
export NODE_ENV="development"

# Variables en archivos .bashrc o .zshrc
echo 'export PATH=$PATH:/new/path' >> ~/.bashrc
```



DESARROLLO EN CLASE



Navegación básica

1. Ver dónde estamos
2. Listar contenido
3. Crear estructura de proyecto
4. Navegar y explorar

Manipulación de archivos

1. Crear archivo con contenido
2. Ver contenido
3. Copiar un archivo
4. Busquemos archivos
5. Buscar contenido

Script simple

1. Crear script
2. Dar permisos
3. Ejecutar

¿Sabías que..?

"¿Sabías que el 95% de los servidores web del mundo funcionan con Linux y no tienen interfaz gráfica? Esto significa que la terminal no es opcional, es esencial para cualquier desarrollador!"



MUCHAS GRACIAS