Instalación y arranque del sistema

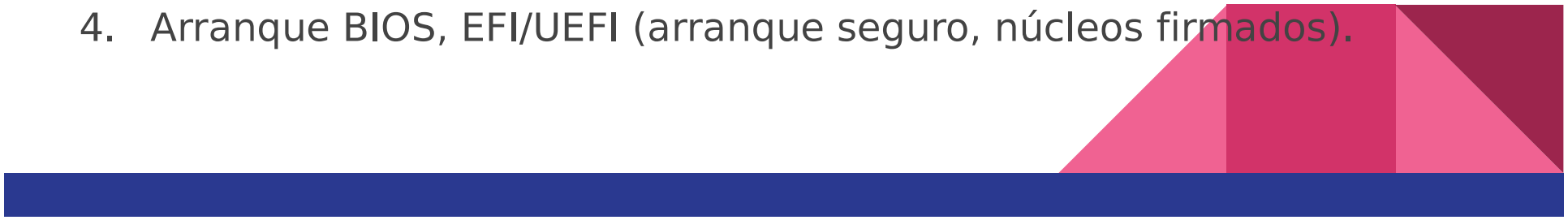
Instalación. Servicios, systemd

Instalación.

1. A partir de una imagen ISO. Esta puede estar en un USB.

```
dd bs=4M if=/ruta/a/distrolinux.iso of=/dev/sdX && sync
```

2. Algunos ISO son auto-arrancables permitiendo usar el PC sin necesidad de instalar nada en él.
3. Cargador de arranque:
   a. Grub
   b. systemd-boot
4. Arranque BIOS, EFI/UEFI (arranque seguro, núcleos firmados).

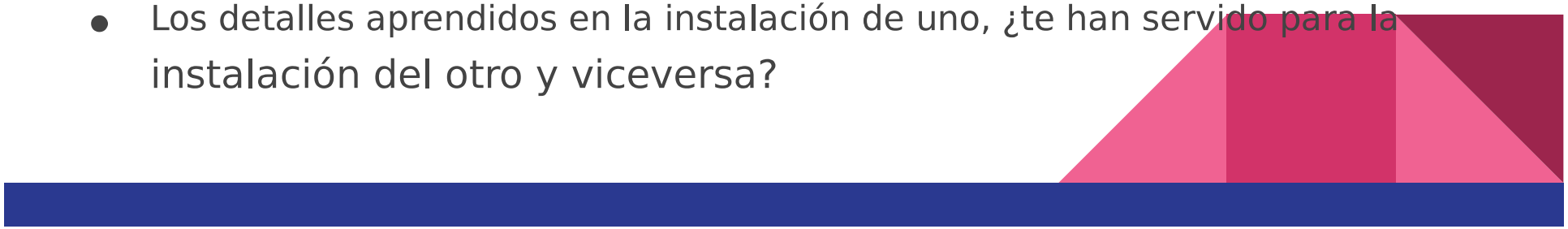


Instalación.

Puede variar de unos Sistemas Operativos a otros.

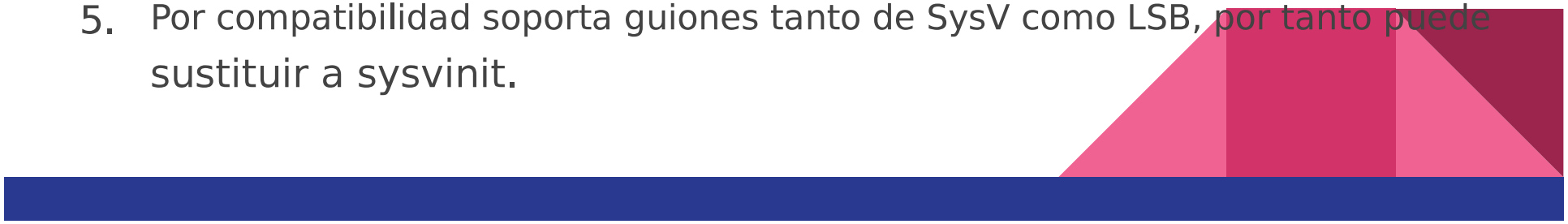
1. Descarga Debian-9.0 (amd64) y trata de instalarlo en una máquina virtual.
2. Prueba a hacer lo mismo p.e. con Ubuntu

- ¿Qué diferencias has observado?
- ¿Cuál te ha resultado más sencillo?
- Los detalles aprendidos en la instalación de uno, ¿te han servido para la instalación del otro y viceversa?



Servicios, systemd (I)

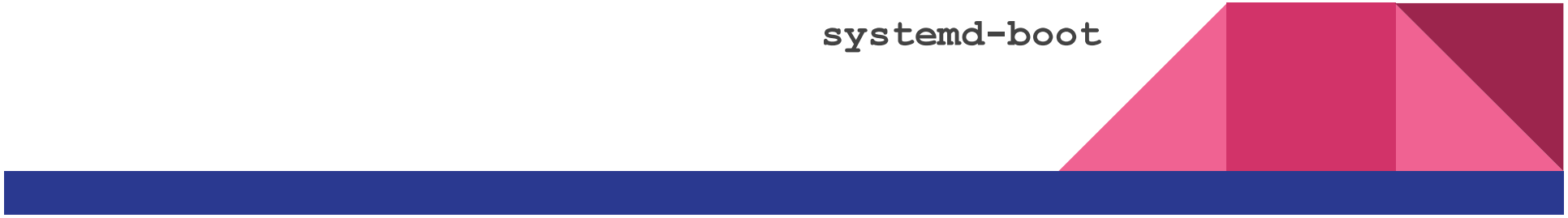
1. Comenzó siendo un gestor de inicio (**init**) del S.O. Básicamente un sustituto de **sysvinit** (ha sido algo traumático y en ciertos sectores tiene detractores).
2. Con el tiempo se ha ido expandiendo más allá de la gestión del arranque de la máquina.
3. Proporciona tanto un gestor de servicios como del propio sistema y se ejecuta con el **PID 1**.
4. Permite paralelizar muchos de los servicios en el arranque, puede activar servicios por socket o también por D-bus.
5. Por compatibilidad soporta guiones tanto de SysV como LSB, por tanto puede sustituir a sysvinit.



Servicios, systemd (II)

1. Gestiona puntos de montaje, servicio de logs.
2. Proporciona utilidades para controlar la configuración básica del sistema (hostname, date, locale, lista de usuarios logged-in así como de contenedores y máquinas virtuales en ejecución, cuentas del sistema, directorios *runtime* y ajustes -*settings*-).
3. Gestiona servicios para la configuración de la red, sincronización de fecha y hora a través de internet, redirección de logs y resolución de nombres.
4. Permite gestionar el arranque del S.O.: .

`systemd-boot`



Servicios, systemd (III)

- Interfaz con el usuario:
  a. `systemctl`:
     - Interacción con systemd.
     - Usa el interfaz tipo:

       `orden [opciones-orden] sub-orden [opciones-sub-orden]`
  b. `journalctl`:
     - Consulta de logs, errores producidos por algún componente de systemd.
- Configuración:
  a. Cuando se ejecuta como instancia del sistema: `/etc/systemd/system.conf`
  b. Cuando se ejecuta como instancia de usuario (`--user`): `/etc/systemd/user.conf`

Servicios, systemd (systemctl)

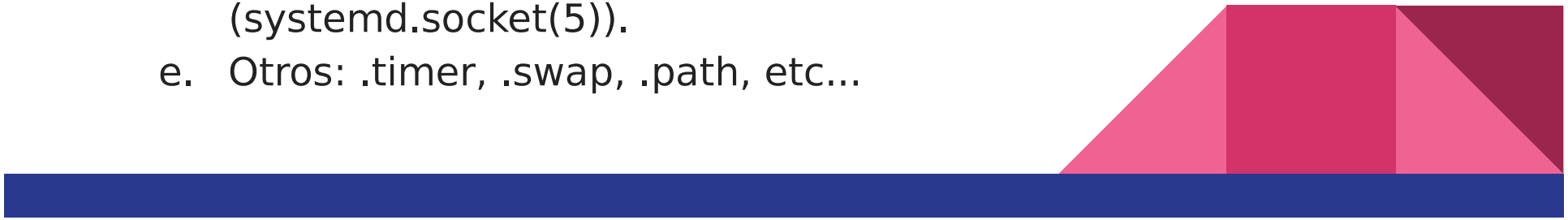
- Subórdenes básicas:
  a. `start, stop, kill, restart, reload (recarga conf.)`
  b. `status, list-dependencies`
  c. `enable, disable, is-enabled, mask, unmask`
  d. `daemon-reload # Reinicia el propio systemd`
  e. `--user # ~/.config/systemd/user`
  f. `--help (página de manual de la unidad),`
- Unidades *estáticas*.
- Se puede ejecutar sin ninguna sub-orden, en ese caso muestra una lista de servicios cargados y unidades que han fallado en su inicio.
  a. `systemctl`
  b. `systemctl list-units # Similar al anterior`
  c. `systemctl --failed`
- Los archivos de las unidades disponibles se pueden ver en `/usr/lib/systemd/system/` y `/etc/systemd/system/` (este último tiene prioridad). Se puede ver un listado de las unidades instaladas con: `systemctl list-unit-files`

Servicios, systemd (systemctl)

- Gestión de energía:
  a. `systemctl reboot`
  b. `systemctl poweroff`
  c. `systemctl suspend`
  d. `systemctl hibernate`
  e. `systemctl hybrid-sleep`

Servicios, systemd (unidades I)

- Las unidades pueden ser:
  a. **servicios** (.service): inician y controlan *daemons* (servicios, systemd.service(5))
  b. **puntos de montaje** (.mount), controla puntos de montaje (systemd.mount(5))
  c. **dispositivos** (.device), hace visibles dispositivos del sistema a syste puede ser usado en activación basada en dispositivos (systemd.device(5)).
  d. **sockets** (.socket): Usados para activación basada en *sockets* (systemd.socket(5)).
  e. Otros: .timer, .swap, .path, etc...



Servicios, systemd (unidades II)

- Al usar systemctl se tiene que especificar el nombre completo de la unic incluyendo el sufijo (*sshd.socket*). Pero existen atajos en las siguientes órdenes systemctl:
  - Si no se especifica el sufijo, systemctl asumirá que es .service. Por ejemplo, *netcfg* y *netcfg.service* son equivalentes.
  - Los puntos de montaje se traducirán automáticamente en la correspondiente unidad .mount (*/home será equivalente a home.mour*
  - Los dispositivos se traducen automáticamente en la correspondiente unidad .device (*/dev/sda2* es equivalente a *dev-sda2.device*).

Servicios, systemd (targets)

- Los targets u objetivos son similares a los **runlevels** o niveles de ejecución de *sysvinit*.

  ```
  systemctl list-units --type=target
  ```

- Para cambiar de target o nivel de ejecución usamos isolate:

  ```
  systemctl isolate graphical.target
  ```

- Son útiles para agrupar unidades y no ofrecen ninguna funcionalidad añ a las que éstas proporcionan.
- Al iniciarse el sistema systemd activa el target **default.target**.

Servicios, systemd (dependencias)

- Al describir una unidad o un objetivo podemos especificar diferentes tipo dependencias.
  a. Positivas: `Requires, Wants (weaker version of Requires)` .
  b. Negativas: `Conflicts`.
  c. De orden:
     - `After`
     - `Before`
- Si existe dependencia positiva entre dos unidades y no existe orden, se inician en paralelo.

Servicios, systemd (journalctl)

- systemd-journald (*/etc/systemd/journald.conf*)

```
journalctl -b                # Show all messages since last boot
journalctl -f                # Tail your logs
journalctl --since=yesterday # Show all messages produced since
yesterday
journalctl -pcrit            # Filter messages by priority
journalctl /bin/su           # Filter messages by program
journalctl --disk-usage      # The amount of space in use for journaling
```

- Mostrar entradas en el LOG del sistema para la unidad pasada como argumento: `journalctl -u UNIT-PATTERN`
- Dispones de un buen tutoria`journalctl` aquí.

Servicios, systemd (contenedores I)

- systemd-nspawn

```
# Crear un chroot con Debian Jessie en el
$ debootstrap jessie /srv/chroots/jessie http://http.debian.net/debian
# Ejecutar un shell, aprovechar para cambiar passwd de root
$ sudo systemd-nspawn -D /srv/chroots/jessie
# Arrancar el Sistema en el chroot
$ sudo systemd-nspawn -b -D /srv/chroots/jessie
# Estado del contenedor
$ sudo machinectl status jessie
```

- No olvides **`man machinectl`**
- Ejercicio: trata de crear un contenedor con Debian stretch.

Servicios, systemd (contenedores II)

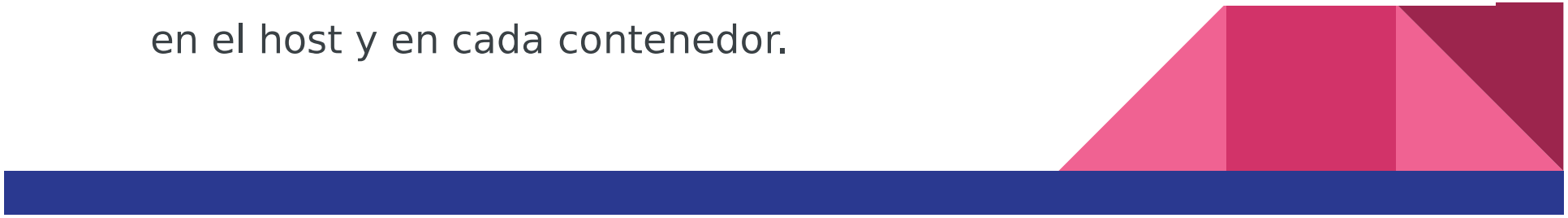
- Las órdenes de `systemd` admiten la opción **-M contenedor**, y se aplican en ese contenedor.
- Los nombres de los S.O. que puede crear **debootstrap** se encuentran en `/usr/share/debootstrap/scripts`.
- Para crear carpetas compartidas entre el *host* y un contenedor usamos la suborden `bind` de **machinectl**:

```
machinectl --mkdir bind jessie ~/datos
```

- Con la opción `--mkdir` crea el directorio para el punto de montaje si no existe.

Servicios, systemd (contenedores III)

- Si al intentar conectarte a un contenedor con **machinectl login** observas un error relacionado con PTY es porque *falta instalar en el contenedor* el paquete **dbus**.
- Para que el contenedor, al ser iniciado, tenga conexión de red debes arrancar el servicio **`systemd-networkd`**, **tanto en el host como en el contenedor**.
- Esto hace que en el host te puedas referir por el nombre del contenedor como su nombre en la red, p.e. `ssh jessie`.
- Si quieres que ésto último sea automático, debes habilitar (*enable*) el servicio en el host y en cada contenedor.



Servicios, systemd (contenedores IV)

- Si queremos arrancar aplicaciones gráficas en el contenedor y visualizarlas en el host debemos:
  a. Hacer visible el directorio del host donde el servidor guarda determinados archivos:

     ```
     sudo machinectl --mkdir bind jessie /tmp/.X11-unix
     ```

  b. Dar permiso de visualización a clientes de X11 locales:

     ```
     xhost +local:
     ```

  c. Comprobamos el valor de la variable de entorno en el HOST llamada **DISPLAY**:

     ```
     echo $DISPLAY
     ```

     Pues en el contenedor deberá tener un valor *similar*. Si en el host vale:

     ```
     $ echo $DISPLAY
     :0
     ```

     En el contenedor le daremos el valor:

     ```
     export DISPLAY=:0.0 # Es posible que funcione también con :0
     ```

  d. Los pasos **a** y **b** no son necesarios si nos conectamos al contenedor por **ssh** y usando la opción **-X**.

Servicios, systemd (análisis del arranque)

- **`systemd-analyze blame`**
  Muestra las unidades y su tiempo de arranque al iniciar el sistema.
- **`systemd-analyze critical-chain`**
  Muestra los caminos críticos en retrasos del arranque del sistema.
- **`systemd-analyze plot > boot.svg`**
- **`systemd-analyze dot > boot.dot`**, `(dot -T png boot.dot -o boot.png)`
- **`systemd-analyze dump`**. Muestra una serialización del estado del servidor.
- **`systemd-analyze verify unit`**$_1$**`...unit`**$_n$. Comprueba que las unidades pasadas como argumentos no tienen errores.

Servicios, systemd (systemd-boot)

- Evolución de `gummiboot`.
  Gestor de arranque UEFI.
- Sólo puede arrancar ejecutables EFI:
  - `Linux kernel` **`EFISTUB`**
  - `UEFI Shell`,
  - `GRUB`
  - `Windows Boot Manager`
- Instalación: `bootctl --path=`*`esp`* `install`, esp=*Efi System Partition*, suele estar montada en `/boot`
- Más información aquí.