Localización e internacionalización de lsard VDI

Lucas Dos Santos Pereira
Cristian Berdún Gómez
INS Puig Castellar

Introducción **4**
Contexto y justificación del Trabajo 4
Objetivos del Trabajo 4
Planificación del proyecto 5
Diagrama de Gantt 6
Enfocamiento y método seguido 7
Breve sumario de productos obtenidos 7
Breve descripción de los otros capítulos de la memoria 8

1. Virtualización **9**
1.2. Historia de la virtualización 10
1.3. ¿Cómo funciona la virtualización? 12
1.4. Tipos de virtualización 13
1.4.1. Virtualización de datos 13
1.4.2. Virtualización de escritorios 13
1.4.3. Virtualización del sistema operativo 14
1.4.4. Virtualización de las funciones de red 15

2. KVM/Libvirt **16**
2.1. ¿Cómo funciona KVM? 16
2.2. Implementar KVM 16
2.3. Instalación de KVM 17
2.4. Creación de las máquinas virtuales 18

3. ¿Que es Isard VDI? **20**

4. Instalación de Isard VDI en MV (paso por paso) **22**
4.1. Pasos a realizar en virt-manager 22
4.2. Pasos a realizar en Ubuntu server 26

5. Localización e internacionalización **32**
5.1 Internacionalización 32
5.2. Localización 33
5.2.1. Importancia de la localización 34
5.2.2 Las diferentes etapas de la localización 35
Primera etapa 35
Segunda etapa 35
Tercera etapa 36

6. Ficheros de traducción POT, PO y MO **37**
6.1. Gettext 37
6.1.1. Pot 37
6.1.2. Po 37
6.1.3. Mo 38
6.2. Aplicaciones para generar ficheros POT, PO y MO 38
6.2.1. Launchpad 38
6.2.2. Pootle 38
6.2.3. Babel 39
7. Traducción de los templates. **40**
8. Conclusiones **43**
9. Bibliografia **44**

Introducción

Contexto y justificación del Trabajo

Isard VDI es una nueva propuesta que hemos descubierto gracias a nuestro profesor Victor Carceler, Isard es una implementación de código abierto basada en la virtualización KVM de Linux y en los dockers. Debido a que es bastante reciente y apenas está extendido solamente podemos encontrarlo en Inglés y por ello vamos a traducirlo al Catalán y Castellano mediante un compilador que en nuestro caso será Babel.

Isard tiene muchas ventajas, está orientado a profesores y alumnos autónomos con sus escritorios, tiene una transición sencilla a VDI en el aula y sobretodo una gran reducción de costes. Por eso creemos que traducirlo a otras lenguas, en nuestro caso Catalán y Castellano hará que sea accesible a más gente y así podremos darle reconocimiento a Isard y con ello contribuir al desarrollo de este proyecto.

Objetivos del Trabajo

Tenemos varios objetivos muy claros pero también hay otros que derivan de los objetivos principales que nos parecen también muy interesantes, son los siguientes:

- Estudio de la virtualización
- Estudio de KVM/Libvirt
- Estudio de IsardVDI
- Localización de software e internacionalización
- Estudio de Babel
- Importación de cadenas en Babel
- Traducción de cadenas
- Commit a GitHub
- PullRequest al proyecto original
- Comunicación con los desarrolladores
- Memoria
- Presentación
- Instalación de IsardVDI con la interfaz traducida

Planificación del proyecto

Recursos necesarios

Para realizar nuestro proyecto no harán falta muchos recursos ya que todas las herramientas que utilizaremos son de Software libre, como mucho lo que necesitaremos será un ordenador con acceso a varias Máquinas virtuales y con capacidad suficiente para poder albergar Isard VDI.

Los recursos libres que utilizaremos serán:

Babel:

Podemos descargar Babel a través de:
http://babel.pocoo.org/en/latest/installation.html
Solamente tenemos que seguir estas instrucciones de instalación.

Isard VDI:

Podemos obtener Isard VDI por:
https://github.com/isard-vdi/isard
Seleccionamos el .zip de Isard y lo descargamos

GitHub:

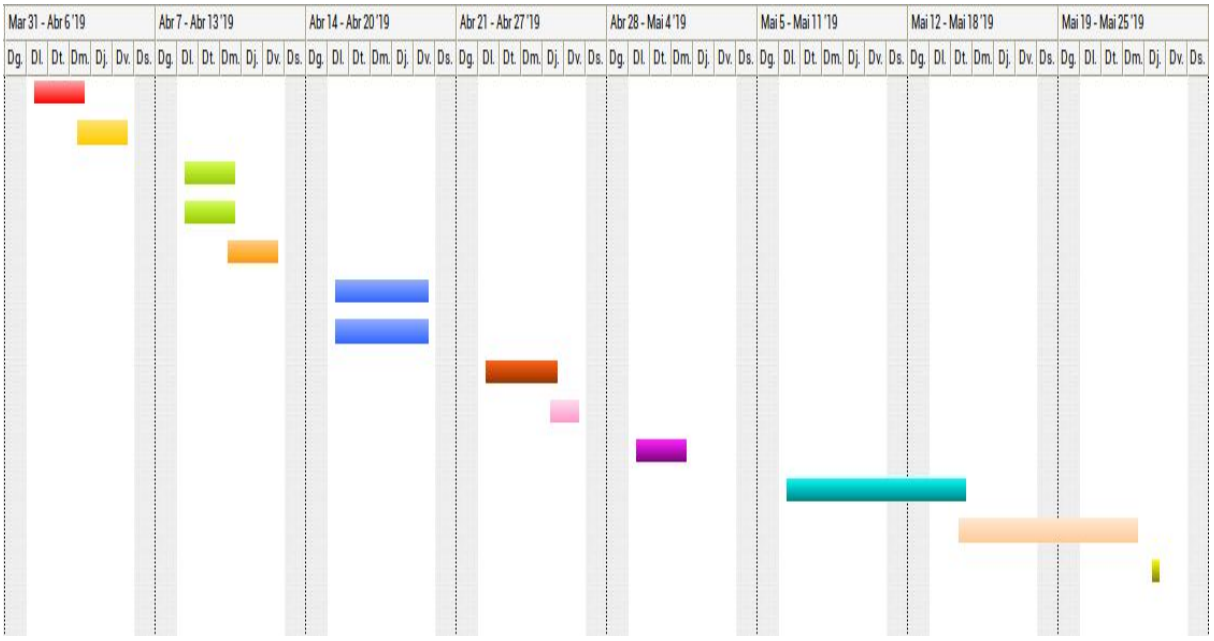
Podemos conseguir GitHub:
https://desktop.github.com/
También podemos utilizarlo por su página web que podemos hacer login con nuestro usuario y password y utilizarlo online.

KVM/Libvirt:

Podemos adquirir KVM/Libvirt:
https://wiki.debian.org/KVM#Installation
Seguimos los pasos de la Wiki para instalarlo.

Diagrama de Gantt

	Nom	Durada	Inici	Fi
1	Estudio de la virtualización	24hores?	04/01/2019	04/03/2019
2	Estudio de KVM/Libvirt4	24hores?	04/03/2019	04/05/2019
3	Estudio de IsardVDI	24hores	04/08/2019	04/10/2019
4	Localización de software e internacionalización	3dies?	04/08/2019	04/10/2019
5	Estudio de Babel	3dies?	04/10/2019	04/12/2019
6	Importación de cadenas en Babel	5dies?	04/15/2019	04/19/2019
7	Traducción de cadenas	5dies?	04/15/2019	04/19/2019
8	Commit a GitHub	4dies?	04/22/2019	04/25/2019
9	PullRequest al proyecto original	2dies?	04/25/2019	04/26/2019
10	Comunicación con los desarrolladores	3dies?	04/29/2019	05/01/2019
11	Memoria	7dies?	05/06/2019	05/14/2019
12	Presentación	7dies?	05/14/2019	05/22/2019
13	Instalación de IsardVDI con la interfaz traducida	1dia?	05/23/2019	05/23/2019



Para tener una mejor vista del diagrama hemos colgado junto con este un archivo PDF llamado **DiagramaGantt.pdf** donde se puede ver de manera más clara.

Enfocamiento y método seguido

En este trabajo nos vamos a enfocar en distintos objetivos, utilizaremos la ayuda de la web, foros, posts, etc. El enfocamiento se tratará de ser constantes a la hora de hacer los objetivos de trabajo. Utilizaremos varias aplicaciones, programas para realizar nuestro trabajo

Y el método seguido es:

- Hacer una introducción que será la primera entrega del trabajo, que estará formada con índice, planificación del proyecto, productos obtenidos, conclusión, etc.
- Luego de entregar la primera entrega comenzamos con la segunda entrega que consta de la memória medio hecha, con la información casi ya definida y mostrar un uso “beta” para ver si estamos avanzando correctamente.
- Y finalmente después de haber entregado la segunda entrega comenzará la tercera que consiste en acabar la memoria, finalizar toda la información, recursos y las máquinas virtuales.

Breve sumario de productos obtenidos

Los productos que hemos obtenido han sido:

- Máquinas virtuales
- Isard VDI
- Babel
- Ubuntu Desktop
- Memoria
- Diagrama de Gantt

Breve descripción de los otros capítulos de la memoria

Una vez ya sabemos todos los capítulos de la memoria podemos hacer una descripción de cada uno:

1. Introducción: Consta de:
   - 1.1 Contexto y justificación del trabajo: (preguntas como, porqué es un tema relevante, que resultado se quiere obtener,etc)
   - 1.2 Objetivos del trabajo: (Un listado de los objetivos del trabajo que queremos y tenemos que realizar)
   - 1.3 Enfocamiento y método seguido: (Enfocar como hacer el proyecto y el método seguido los pasos que realizaremos para hacer nuestro proyecto)
   - 1.4 Planificación del proyecto: (Será simplemente como nos vamos a organizar el proyecto de forma que lo podamos planificar)
   - 1.5 Un resumen breve de los productos obtenidos: (Resumen breve de los productos obtenidos a lo largo del crédito)
   - 1.6 Descripción breve de los otros capítulos de la memoria: (Explicación de los contenidos de cada capítulo y su relación con el proyecto global)

1. Virtualización

La virtualización es una tecnología que permite crear servicios de TI útiles mediante recursos que normalmente se ejecutan en el hardware. Gracias a ello, permite utilizar toda la capacidad de una máquina física, pues distribuye sus capacidades entre varios usuarios o entornos.
Dicho de otra manera, se refiere a la utilización de los recursos de un ordenador, llamado Hypervisor o VMM (Virtual Machine Monitor) que crea una capa de abstracción entre el hardware de la máquina física (host) y el sistema operativo de la máquina virtual (virtual machine, guest), dividiéndose el recurso en uno o más entornos de ejecución.

Virtual machine monitor es una capa de software que maneja, gestiona y arbitra los cuatro recursos principales de un ordenador: CPU, Memoria, Dispositivos Periféricos y Conexiones de Red. Esto hace que se puedan tener varios ordenadores virtuales ejecutándose en el mismo ordenador físico.

"VMM en Windows 10"

El término virtualización es bastante antiguo, se viene usando desde 1960. Ha sido aplicado a diferentes aspectos y ámbitos de la informática, finalmente ha pasado a ser un componentes fundamental para la informática.

Existen diferentes formas de virtualización, desde virtualizar el hardware de servidor, el software de servidor, sesiones de usuario, aplicaciones y también crear máquinas virtuales en un ordenador de escritorio.

Los principales proveedores de software que han desarrollado tecnologías de virtualización que abarcan todas las opciones de virtualización: servidor, aplicaciones, escritorio. Tenemos por ejemplo Citrix, VMware y Microsoft. Estas compañías han diseñado soluciones específicas para virtualización, como XenServer, VMware ESX y Windows Server 2008 Hyper-V para la virtualización de servidores.

1.2. Historia de la virtualización

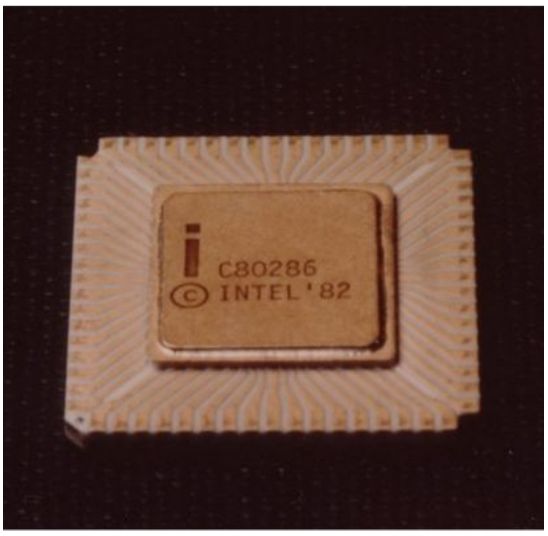
El tema de la virtualización no es nada nuevo. Durante la década de los 60 los equipos de informática de muchas empresas y entidades tenían el mismo problema y es que contaban con “superordenadores” o “mainframes[1]” de alto rendimiento que necesitaban “particionar lógicamente”, o utilizar para múltiples tareas ejecutadas al mismo tiempo lo que hoy se conoce como “multitasking” básicamente trabajar más de una aplicación o proceso simultáneamente. Es por esto que IBM desarrolló un método para crear múltiples “particiones lógicas” bastante similar a lo que se conoce actualmente como “máquinas virtuales” las cuales trabajaban independientemente una de las otras, y cada una utilizando los recursos provistos por el “mainframe”.

“Mainframe de IBM en el año 1965”

Sobre la década de los 80 y con la llegada de las máquinas x86, comenzó una nueva era de microordenadores[2], aplicaciones cliente-servidor, y “computación distribuida”, donde los enormes y potentes “mainframes” se comenzaron a cambiar por pequeños servidores y ordenadores personales de arquitectura x86, con “una caja diferente para cada uso”, lo que se convirtió rápidamente en el estándar de la industria.

[1] Computadora grande, potente y costosa, usada principalmente para el procesamiento de una gran cantidad de datos.

[2] Computadora pequeña, con un microprocesador como su unidad central de procesamiento CPU



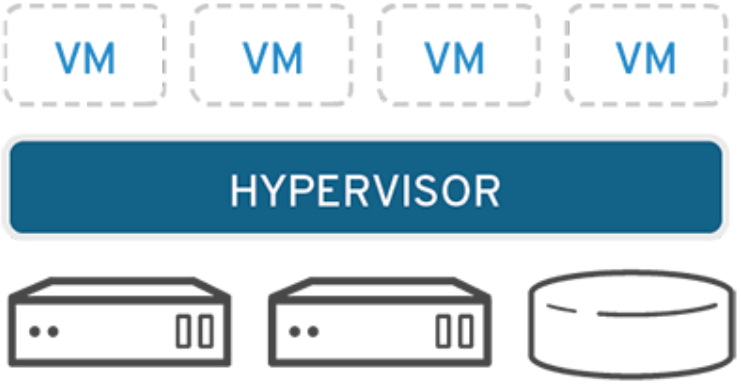
“Procesador 8086 considerado el primer x86 de la historia”

“Commodore Amiga comercializada entre 1985-1994”

Debido a esto, el tema de la virtualización vuelve a quedar prácticamente en el olvido. No es hasta finales de la década de los 90 que gracias al alto desarrollo del hardware se vuelve a caer en un razonamiento similar al que se llegó en los años 60 ya que el hardware de por aquel entonces era altamente eficiente, y utilizar cada “caja” para una sola aplicación sería un desperdicio de recursos, espacio, energía y dinero, y tampoco es conveniente asignarle múltiples usos o instalar varias aplicaciones en un solo servidor, por más de una razón ya que estas aplicaciones podrían ser conflictivas entre sí, o podrían necesitar diferentes configuraciones o incluso diferentes sistemas operativos, o tener diferentes necesidades de seguridad, entre otras cosas que podrían causar problemas al ejecutar estas funciones al mismo tiempo. Es por esto que se vuelve a pensar en resurgir la idea de dividir el hardware, de tal manera que funcione como múltiples servidores independientes pero compartiendo los recursos de un mismo servidor físico. Y es a partir de esto que nace lo que hoy conocemos como “Virtualización”.

1.3. ¿Cómo funciona la virtualización?

Primeramente el software denominado hipervisor separa los recursos físicos de los entornos virtuales, es decir, todo lo que necesitan los recursos. Los hipervisores pueden conformarse como elementos principales de un sistema operativo (como un ordenador portátil) o se pueden instalar directamente en el hardware (como un servidor), que es la forma en que la mayoría de las empresas virtualiza. Los hipervisores toman los recursos físicos y los dividen de manera tal que los entornos virtuales puedan usarlos.

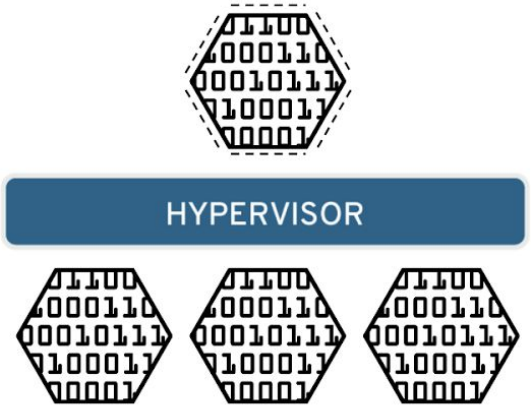


Los recursos se dividen según las necesidades, desde el entorno físico hasta los numerosos entornos virtuales. Los usuarios interactúan con los cálculos y los ejecutan dentro del entorno virtual (generalmente denominado máquina de guest o máquina virtual). La máquina virtual funciona como un archivo de datos único. Al igual que cualquier archivo digital, se puede migrar de un ordenador a otro, abrir en cualquier computadora y prever que funcione de la misma manera.

Cuando el entorno virtual se está ejecutando y un usuario o programa emite una instrucción que necesita recursos adicionales del hardware, el hipervisor transmite la solicitud al sistema físico y guarda los cambios en la caché. Todo esto sucede casi a la misma velocidad que si este proceso se realizará dentro de la máquina física.

1.4. Tipos de virtualización

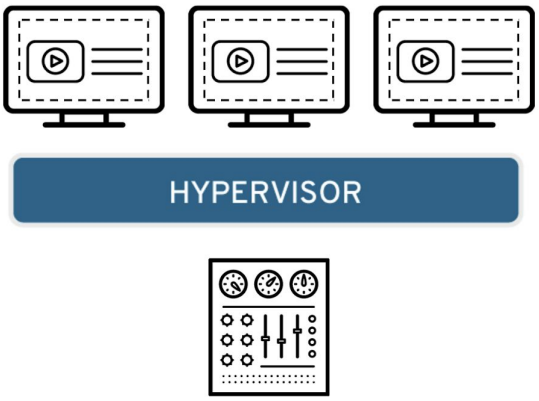
1.4.1. Virtualización de datos



Los datos que se encuentran repartidos por todas partes se pueden juntar en una fuente única. La virtualización de datos permite que las empresas los traten más fácilmente y como si fueran una cadena de suministro, de esta manera, se obtiene la capacidad de procesamiento que permitirá reunir los datos de varias fuentes, se integran otras fuentes nuevas fácilmente, y se transforman los datos de acuerdo con las necesidades de los usuarios. Las herramientas de virtualización de datos (como Red Hat® JBoss® Data Virtualization) trabajan con múltiples fuentes de datos y permiten tratarlas como una sola, y proporcionan los datos necesarios, de la forma requerida y en el momento justo, para cualquier aplicación o usuario.

1.4.2. Virtualización de escritorios

La virtualización de escritorios se confunde fácilmente con la virtualización de sistemas operativos. La virtualización de SO permite implementar múltiples sistemas operativos en una sola máquina en cambio la virtualización de escritorios permite que un administrador central o una herramienta de administración automatizada implementen entornos simulados de escritorio en cientos de máquinas físicas al mismo tiempo.

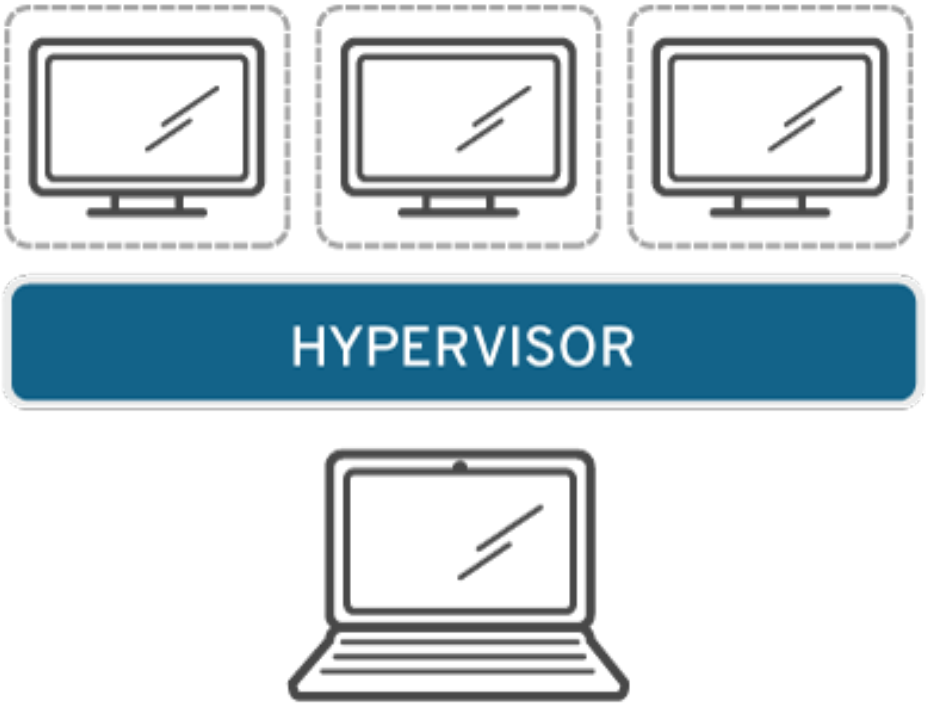


A diferencia de los entornos de escritorio tradicionales que se instalan, configuran y actualizan físicamente en cada máquina, la virtualización de escritorios permite que los administradores realicen configuraciones, actualizaciones y controles de seguridad de forma masiva en todos los escritorios virtuales.

1.4.3. Virtualización del sistema operativo

La virtualización del sistema operativo se realiza en el kernel[3], es decir, los administradores de tareas centrales de los sistemas operativos. Es una forma útil de ejecutar los entornos Linux y Windows de manera paralela. Las empresas también pueden instalar sistemas operativos virtuales en los ordenadores, los beneficios de esto son:

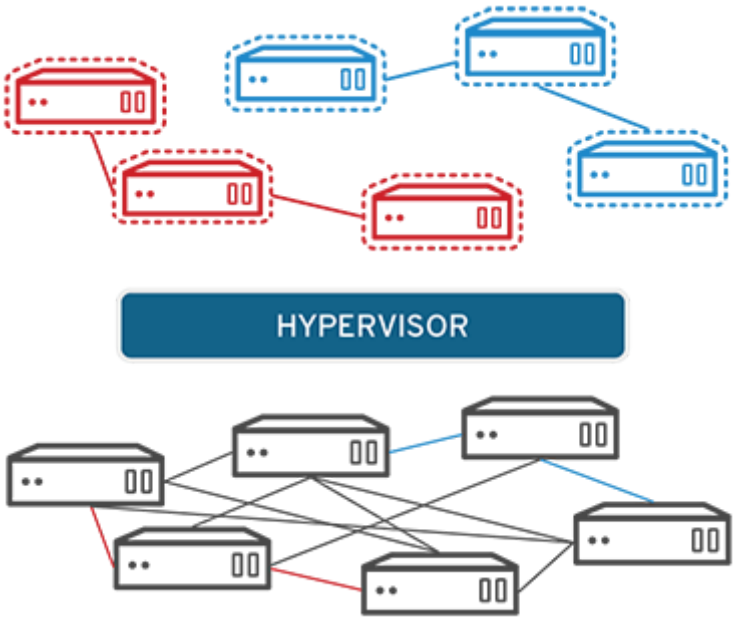
- Reduce el costo del hardware en masa, debido a que los ordenadores no requieren capacidades tan inmediatas.
- Aumenta la seguridad porque todas las instancias virtuales se pueden supervisar y aislar.
- Limita el tiempo que se destina a los servicios de TI, como las actualizaciones de software.



[3] Software que constituye una parte fundamental del sistema operativo, y se define como la parte que se ejecuta en modo privilegiado

1.4.4. Virtualización de las funciones de red

La virtualización de las funciones de red (NFV) separa las funciones clave de una red (como los servicios de directorio, el uso compartido de archivos y la configuración de IP) para distribuirlas en los entornos. Cuando las funciones del software se independizan de las máquinas virtuales en las que se encontraban, las funciones específicas se pueden empaquetar en una nueva red y asignarse a un entorno. La virtualización de redes reduce la cantidad de componentes físicos (como conmutadores, routers, servidores, cables...) que se necesitan para crear varias redes independientes y es muy popular en el sector de las telecomunicaciones.



2. KVM/Libvirt

Kernel-based Virtual Machine o KVM, en español, Máquina virtual basada en el núcleo. Es una solución para implementar la virtualización completa con Linux. Está formada por un módulo del núcleo (con el nombre kvm.ko) , siendo en su totalidad software libre. El componente KVM para el núcleo está incluido en Linux desde la versión 2.6.20.

KVM permite ejecutar máquinas virtuales utilizando imágenes de disco que contienen sistemas operativos sin modificar. Cada máquina virtual tiene su propio hardware virtualizado: una tarjeta de red, discos duros, tarjeta gráfica, etc.

KVM fue creado, por Qumranet y en 2008 esta fue adquirida por Red Hat Inc2. Actualmente el software es mantenido por Open Shift.

Por otra parte cuando hablamos de KVM es inevitable comentar sobre libvirt. Libvirt es una biblioteca para interactuar con las capacidades de virtualización de Linux independientemente de la solución de virtualización que hay debajo (KVM, XEN[4], etc), con la que arrancar/parar/etc máquinas virtuales.

2.1. ¿Cómo funciona KVM?

KVM convierte a Linux en un hipervisor de tipo 1[5]. Todos los hipervisores necesitan algunos componentes al nivel del sistema operativo (por ejemplo, administrador de memoria, planificador de procesos, pila de entrada o salida (E/S), controladores de dispositivos, gestión de seguridad, pila de red y más) para ejecutar las máquinas virtuales. KVM cuenta con todos estos componentes porque es parte del kernel de Linux. Cada máquina virtual se implementa como un proceso regular de Linux, programada por el planificador estándar de Linux con hardware virtual dedicado como tarjeta de red, adaptador de gráficos, CPU, memoria y discos.

2.2. Implementar KVM

Para implementar KVM hay que ejecutar una versión de Linux lanzada después de 2007 y que deba instalarse en hardware x86 que sea compatible con capacidades de virtualización. Si lo anteriormente mencionado ya se ha realizado, entonces todo lo que hay que hacer es cargar dos módulos existentes (un módulo host del kernel y un módulo específico del procesador), un emulador y cualquier controlador que lo ayude a ejecutar sistemas adicionales.

[4] Monitor de máquina virtual de código abierto desarrollado por la Universidad de Cambridge.

[5] Se caracteriza porque este software se instala directamente sobre el equipo

2.3. Instalación de KVM

Estamos hablando de una solución de virtualización para utilizar en los sistemas Linux. KVM utiliza Virt-Manager como administrador de máquinas virtuales y un Qemu modificado como hypervisor. Para instalarlo lo primero que debemos saber es si nuestro procesador es compatible. Para ello introduciremos los siguientes comandos:

$ cat /proc/cpuinfo | grep vmx # para CPUs Intel
$ cat /proc/cpuinfo | grep svm # para CPUs AMD

Si el sistema no nos devuelve nada significa que no tenemos soporte de virtualización en nuestro procesador o no lo tenemos activado. En caso afirmativo, instalamos los paquetes necesarios:

$ sudo apt-get install kvm libvirt-bin ubuntu-vm-builder bridge-utils

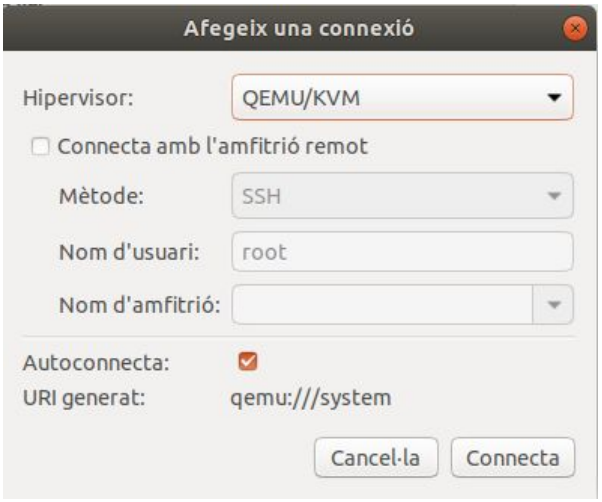
 # E incluimos el usuario en el grupo de kvm

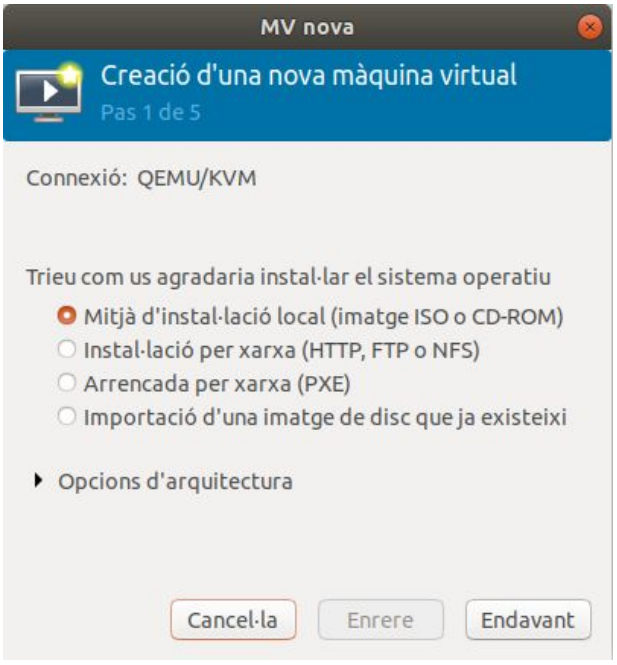
$ sudo adduser $USER kvm

Una vez realizado todos estos pasos tendremos que reiniciar nuestro ordenador, dependiendo de la versión de Ubuntu que estemos utilizando. Si todo ha ido bien, aparecerá una nueva entrada en nuestro menú Aplicaciones/Herramientas del Sistema/Administrador de Máquina Virtual desde donde se comenzará la instalación de las máquinas virtuales.

2.4. Creación de las máquinas virtuales

Primeramente debemos abrir el Administrador de Máquina Virtual donde tendremos que crear una conexión, en caso de que no nos aparezca por defecto. Para ello lo único que se debe seleccionar es QEMU[6] como hipervisor y local como conexión. Tras seleccionar la nueva entrada creada aparecerá activo el botón Nuevo, donde tras pulsarlo se accede al Asistente para la creación de máquinas virtuales.





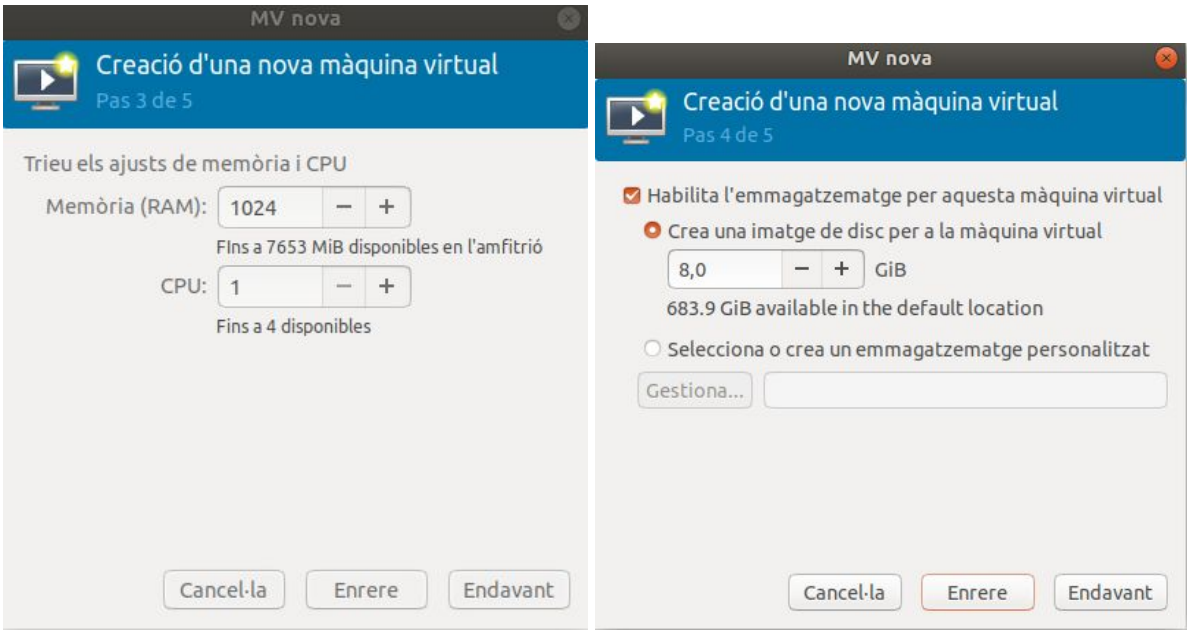
Una vez creada la conexión anterior ya podemos proceder a crear la Máquina virtual, se resume en 5 sencillos pasos, en el primer paso podemos elegir que tipo de instalación de Sistema Operativo queremos.
Tenemos 4 opciones principales que van desde:

- Imagen ISO o CD-ROM
- Por red (HTTP, FTP o NFS)
- PXE[7]
- Importación de imagen existente

Importante mencionar que también podemos seleccionar que tipo de arquitectura queremos tener.

[6] Emulador de procesadores basado en la traducción dinámica de binarios
[7] **(Preboot eXecution Environment)** entorno para arrancar e instalar el sistema operativo en computadoras a través de una red

Independientemente de la opción elegida anteriormente en los siguientes pasos nos pedirán los ajustes de memória y la CPU, el tamaño del disco y el nombre de la máquina virtual cómo vemos a continuación:

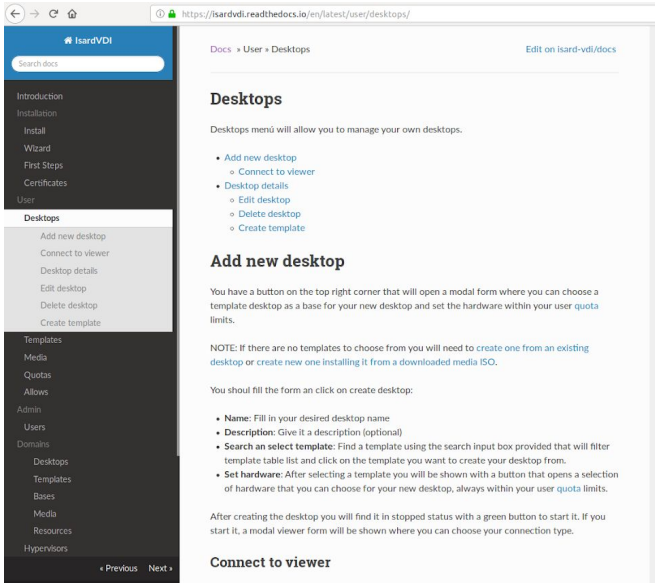


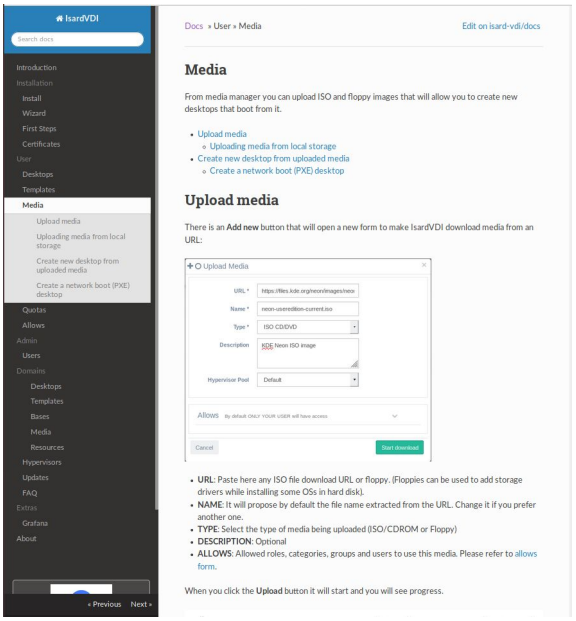
Una vez creada la máquina virtual tendremos que instalar el sistema operativo que hayamos elegido al igual que haríamos en este caso con un ordenador al que le acabamos de instalar el disco duro. Este proceso llevará más o menos tiempo en función de la memoria y procesador que hayamos asignado a nuestra máquina virtual, como si lo instalamos a través de CD, de una imagen *.ISO o de la red que son las opciones disponibles.

3. ¿Que es Isard VDI?

Isard VDI es una Implementación de VDI de código abierto basada en la virtualización KVM de Linux y en los dockers.

Isard es un software gratuito de open source que nos permite administrar escritorios virtuales. Contiene las funciones:





- Poder acceder/conectarnos desde un navegador web.
- Nos permite poder introducir USBs dentro de un escritorio virtual.
- Nos permite utilizar el sonido dentro de un escritorio virtual.
- Nos permite descargar escritorios precreados, isos y recursos desde el menú de actualizaciones.
- Crear escritorios personalizados utilizando isos descargadas desde el menú de actualizaciones
- Permite gestionar usuarios
- Gracias a la creación incremental de discos, todo de descargar escritorios personalizados, isos, etc se puede hacer en minutos.
- Permite poder agregar más hipervisores al grupo y dejar que Isard VDI decida dónde iniciar cada escritorio.
- Isard VDI ahora funciona fuera de la caja con docker y docker-compose
- Instalación muy completa para Debian 9, Fedora 28-29, Ubuntu Server, y muchas más configuraciones.
- Fáciles pasos a seguir para la instalación
- Podemos añadir escritorios
- Podemos conectar al visor
- Podemos ver los detalles del escritorio

- Podemos editar el escritorio
- Podemos borrar el escritorio
- Podemos crear una plantilla
- Subir media
- Subir media desde el disco local
- Crear escritorios de la media subida
- Crear un escritorio de arranque de red
- Cuotas
- Realizar cuotas para usuarios
- Modificar las cuotas de “escritorios”
- Modificar las cuotas de “ejecutamiento”
- Modificar las cuotas de “plantillas”
- Modificar las cuotas de “media / isos”
- Modificar las cuotas de “CPUs”
- Modificar las cuotas de “memoria”
- Edición el apartado de escritorios
- Hipervisores
- Hipervisores externos
- Instalar hypervisor KVM
- Añadir claves ssh para el nuevo hipervisor
- Rendimiento de discos
- Rendimiento de red
- Actualizaciones
- Actualizaciones de “dominios”
- Actualizaciones de “media”
- Actualizaciones de “gráficos”
- Recomendación de actualizaciones
- Administrar FAQ
- Administrar Grafana

4. Instalación de Isard VDI en MV (paso por paso)

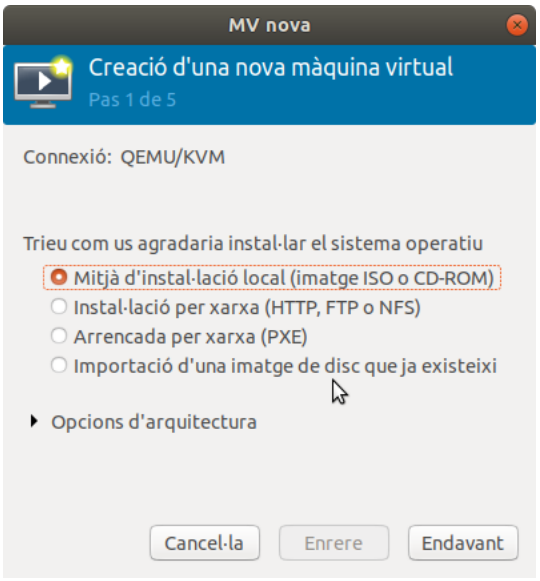
Antes de empezar la creación de la máquina virtual con Isard VDI es necesario descargar la imágen .iso de ubuntu server, sirve cualquier server pero nosotros hemos utilizado el 18.04.02.

Abrir “virt-manager” y crear una nueva MV con la .iso descargada anteriormente y con:

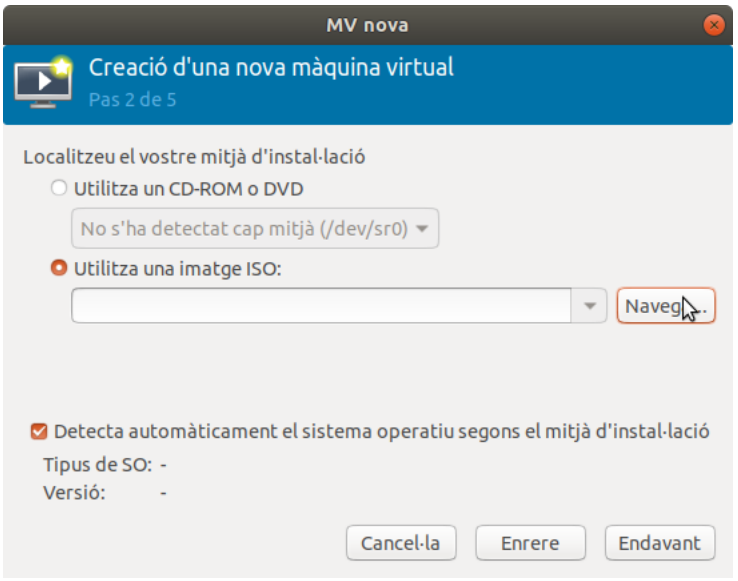
- 4096 Mb de RAM
- 4 CPU
- 100 GiB de Disco duro
- isard-vdi (nombre)

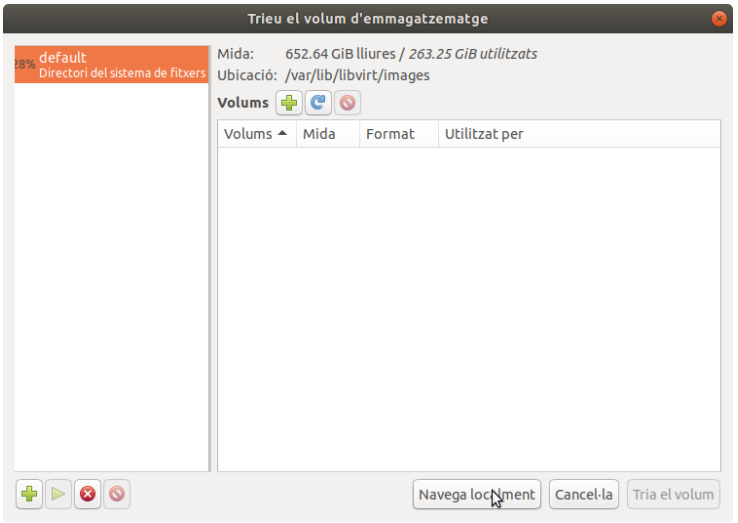
4.1. Pasos a realizar en virt-manager

Primeramente una vez se tiene la conexión establecida en virt-manager procedemos a crear una nueva máquina virtual mediante la instalación local con una imagen .iso.

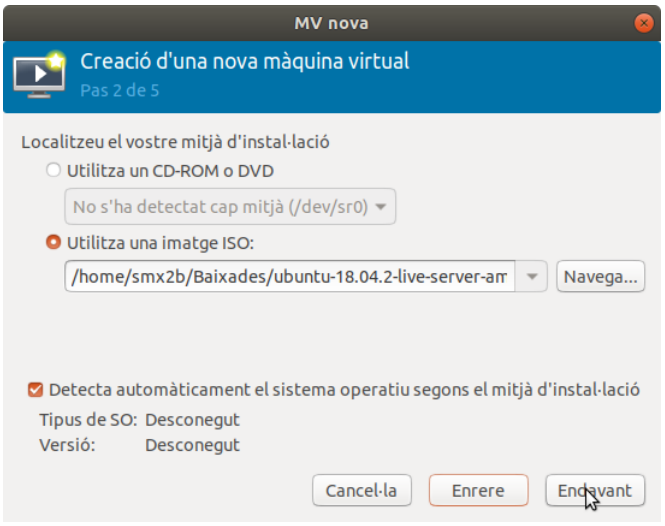


Una vez realizado el paso 1 pasamos a seleccionar la imagen .iso de ubuntu server que se ha descargado anteriormente desde la página oficial de ubuntu.



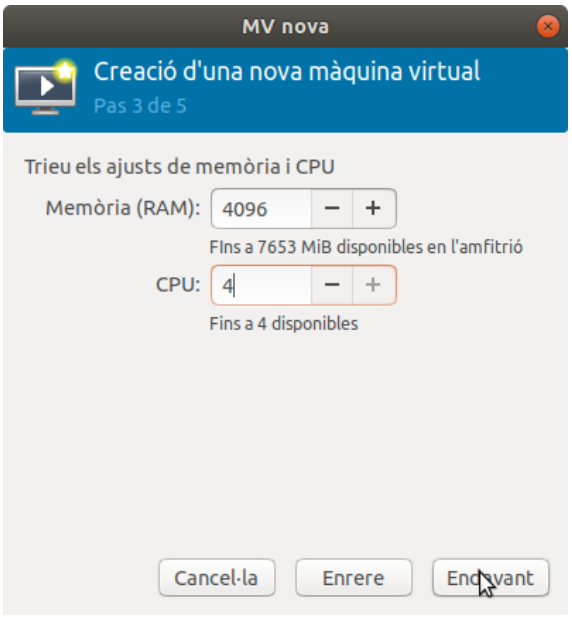


En el botón “Navegar” podemos seleccionar la ruta de manera local donde tengamos guardada la iso de ubuntu server.



Si hemos realizado hasta ahora bien los pasos deberíamos ver lo mismo que vemos en la imagen de la izquierda pero con diferente ruta.

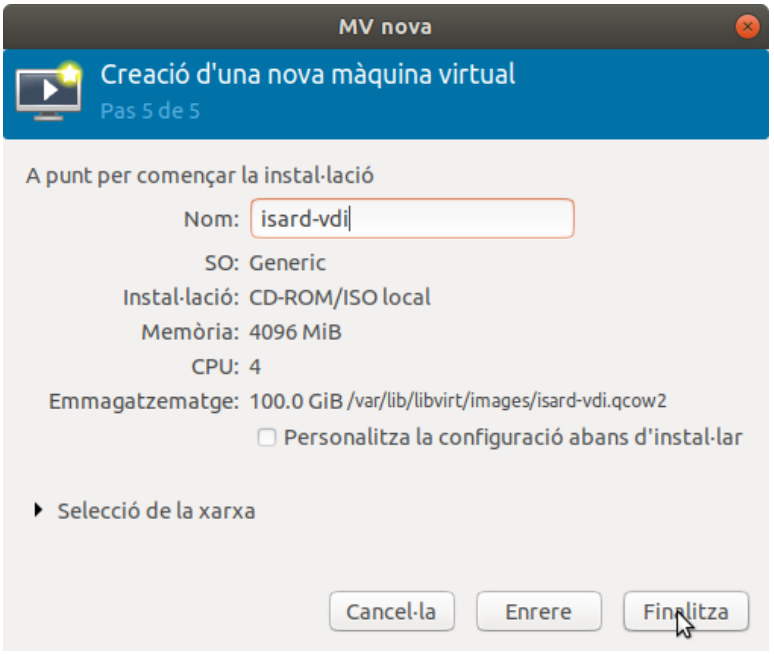
El siguiente paso se trata de elegir los ajustes de memória y de CPU, en nuestro caso necesitaremos 4 GiB de RAM y 4 CPU's para poder trabajar bien con Isard VDI



Los últimos pasos de la creación de la máquina virtual son bastante simples, en el paso 4 solamente habrá que ajustar el tamaño del disco para la MV, para nuestro Isard con 100 GiB es más que suficiente.

Por último el paso 5 se trata solamente de introducir el nombre queramos ponerle a nuestra máquina virtual a parte de esto en la pestaña nos mostrará un pequeño resumen de las opciones seleccionadas anteriormente.

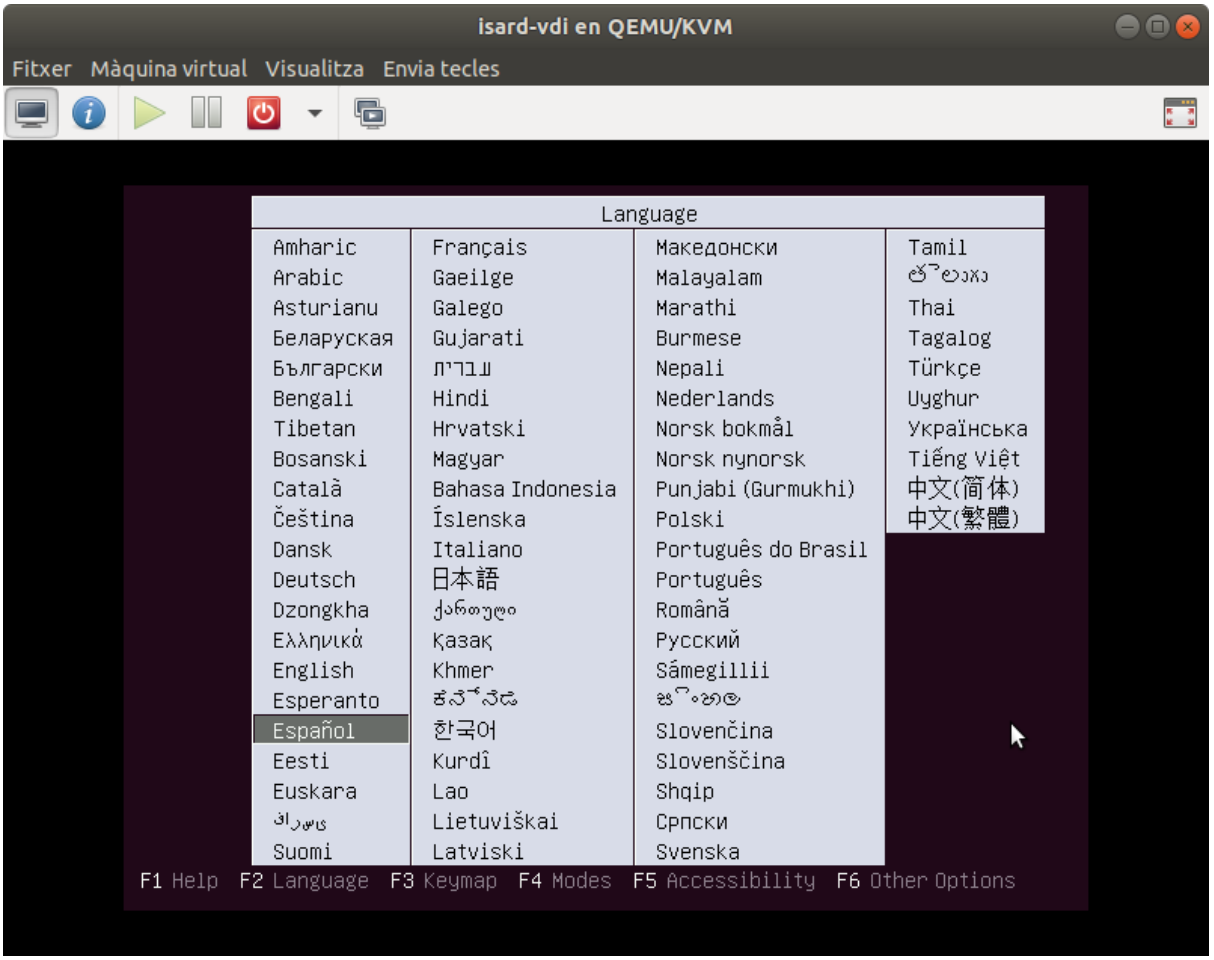
"Penúltimo paso de la creación"



"Último paso de la creación"

4.2. Pasos a realizar en Ubuntu server

Después de realizar los anteriores pasos, si se ha configurado correctamente, la máquina se ejecutará y aparecerá la siguiente ventana:



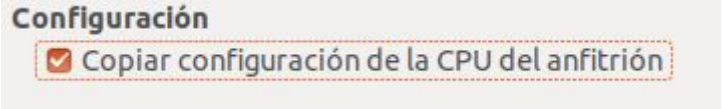
Seleccionamos el idioma Español y pulsamos enter.

Seleccionamos la opción de “Instalar Ubuntu Server”. Y seguimos todos los pasos de la instalación para instalar nuestro servidor.

Una vez reiniciada la máquina con el Ubuntu instalado debemos seleccionar estas opciones sobre el Disco virtual de nuestra máquina:

- Bus de disco: VirtIO
- Modo cache: unsafe

- copiar configuración de la CPU del anfitrión



El siguiente paso es dentro del ubuntu 18.04 clonar con git clone Isard VDI:

git clone https://github.com/isard-vdi/isard

```
usuario@isard:~$ git clone https://github.com/isard-vdi/isard
Cloning into 'isard'...
remote: Enumerating objects: 14107, done.
remote: Total 14107 (delta 0), reused 0 (delta 0), pack-reused 14107
Receiving objects: 100% (14107/14107), 59.85 MiB | 2.21 MiB/s, done.
Resolving deltas: 100% (8022/8022), done.
usuario@isard:~$ _
```

Después instalamos docker compose con snap y con apt:

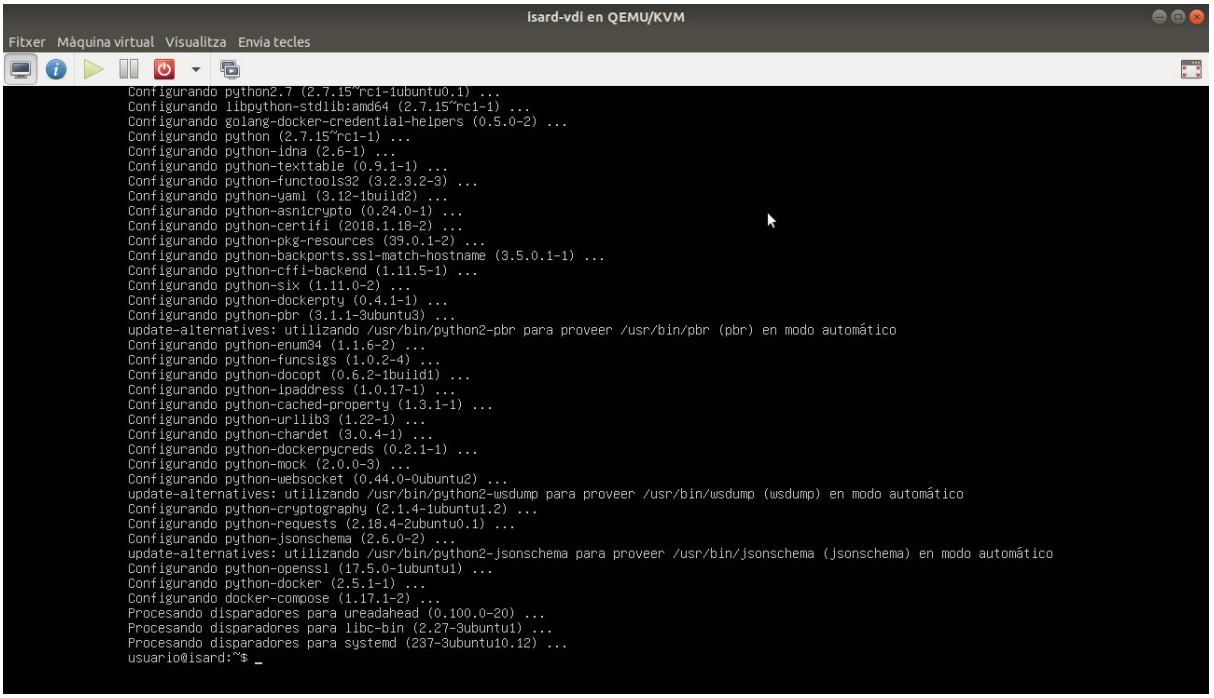
- wget:https://raw.githubusercontent.com/isard-vdi/isard/master/docker-compose.yml

```
usuario@isard:~$ wget https://raw.githubusercontent.com/isard-vdi/isard/master/docker-compose.yml
--2019-05-20 11:23:11--  https://raw.githubusercontent.com/isard-vdi/isard/master/docker-compose.yml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.132.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.132.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1732 (1,7K) [text/plain]
Saving to: ‘docker-compose.yml’

docker-compose.yml             100%[=========================================================>]   1,69K  --.-KB/s    in 0s

2019-05-20 11:23:11 (61,5 MB/s) - ‘docker-compose.yml’ saved [1732/1732]

usuario@isard:~$ _
```

- sudo snap install docker

```
usuario@isard:~$ sudo snap install docker
[sudo] password for usuario:
Start snap "docker" (384) services                                                                                  |
[ 1039.848697] aufs aufs_fill_super:912:mount[2773]: no arg
Run configure hook of "docker" snap if present                                                                      /
[ 1039.887102] overlayfs: missing 'lowerdir'
docker 18.06.1-ce from Canonical✓ installed
usuario@isard:~$ _
```

- sudo apt install docker-compose



- docker-compose pull
- docker-compose up -d

```
version: "3.5"
services:
  isard-database:
    container_name: isard-database
    volumes:
      - "/opt/isard/database:/data"
      - "/etc/localtime:/etc/localtime:ro"
    networks:
      - isard_network
    image: rethinkdb
    restart: unless-stopped
    logging:
        driver: none

  isard-nginx:
    container_name: isard-nginx
    volumes:
      - "/opt/isard/certs/default:/etc/nginx/external"
      - "/opt/isard/logs/nginx:/var/log/nginx"
      - "/etc/localtime:/etc/localtime:ro"
    ports:
      - "80:80"
      - "443:443"
    networks:
      - isard_network
    image: isard/nginx:1
    restart: unless-stopped
    depends_on:
      - isard-app

  isard-hypervisor:
    container_name: isard-hypervisor
    volumes:
      - "sshkeys:/root/.ssh"
      - "/opt/isard:/isard"
      - "/opt/isard/certs/default:/etc/pki/libvirt-spice"
      - "/etc/localtime:/etc/localtime:ro"
    ports:
      - "5900-5999:5900-5999"
      - "55900-55999:55900-55999"
    networks:
      - isard_network
    image: isard/hypervisor:1
```

Editamos el fichero docker-compose.yml y borramos la última línea.

```
networks:
  isard_network:
    external: false
    name: isard_network
"docker-compose.yml" 73L, 1732C                                69,0-1      Final
```

```
networks:
  isard_network:
    external: false
~
:wq_
```

- docker-compose pull

```
isard login: [   12.793172] cloud-init[1349]: Cloud-init v. 18.4-0ubuntu1~18.04.1 running 'modules:f
inal' at Thu, 30 May 2019 13:54:56 +0000. Up 12.67 seconds.
[   12.793310] cloud-init[1349]: Cloud-init v. 18.4-0ubuntu1~18.04.1 finished at Thu, 30 May 2019 13
:54:56 +0000. Datasource DataSourceNoCloud [seed=/var/lib/cloud/seed/nocloud-net][dsmode=net].  Up 1
2.78 seconds
usuario
Password:
Last login: Thu May 30 13:36:29 UTC 2019 on tty1
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-50-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu May 30 13:55:03 UTC 2019

  System load:  0.35              Processes:              92
  Usage of /:   6.0% of 97.93GB   Users logged in:        0
  Memory usage: 21%               IP address for enp0s3:  192.168.1.96
  Swap usage:   0%                IP address for docker0: 172.17.0.1


Pueden actualizarse 111 paquetes.
43 actualizaciones son de seguridad.


usuario@isard:~$ sudo docker-compose pull
[sudo] password for usuario:
Pulling isard-database (rethinkdb:latest)...
latest: Pulling from library/rethinkdb
db0035920883: Extracting [==============================================>    ]  50.69MB/54.39MB
1cdafef2e87a: Download complete
47c168b9ade3: Download complete
b5408f18adbd: Download complete
98fac7b641fa: Download complete
```

```
Pulling isard-hypervisor (isard/hypervisor:1)...
1: Pulling from isard/hypervisor
cd784148e348: Pull complete
6fe13df39e2f: Pull complete
92b36d31b3f4: Pull complete
b2f8fd23e284: Pull complete
db0a65ce04b5: Pull complete
09352fb80fb9: Pull complete
c3fc870cf071: Pull complete
ea74c07b8121: Pull complete
87dc89728c1a: Pull complete
77a8014baa34: Pull complete
8f1440eb0c7a: Pull complete
7210c410a287: Pull complete
a3c52dd9850f: Pull complete
399c723ed296: Pull complete
1e6031e48da6: Pull complete
06b3095b3ef6: Pull complete
c36757f438ee: Pull complete
4cf1e983f91a: Pull complete
7de55504c968: Pull complete
ef8467b69edb: Pull complete
c07b8eb573d7: Pull complete
8f4cdd821a2e: Pull complete
2064341eebd7: Pull complete
802aec6225e9: Pull complete
028d90798a51: Pull complete
b928f4e87d7c: Pull complete
34611e08c64d: Pull complete
0b0ac4cef32b: Pull complete
bb84b7e4f2f2: Pull complete
03828c6340a3: Pull complete
319711229368: Pull complete
Digest: sha256:35ee4c819f560533b6a7d77b18d1ad9ee76bb885bf89fca5ba06b5b54ffa6877
Status: Downloaded newer image for isard/hypervisor:1
Pulling isard-app (isard/app:1)...
```

- docker-compose up -d

```
usuario@isard:~$ sudo docker-compose up -d
Creating isard-hypervisor ...
Creating isard-database ...
Creating isard-database
Creating isard-database ... error

ERROR: for isard-database  Cannot start service isard-database: error while creating mount source pa
Creating isard-hypervisor ... error

ERROR: for isard-hypervisor  Cannot start service isard-hypervisor: driver failed programming extern
al connectivity on endpoint isard-hypervisor (be1c2a4381a216e07c3d38ebf8aa156d8bbcf2ca94bcc612a04cb2
653e055a91):  (iptables failed: iptables --wait -t nat -A DOCKER -p tcp -d 0/0 --dport 5918 -j DNAT
--to-destination 172.18.0.3:5918 ! -i br-d63cf8551e68:  (fork/exec /sbin/iptables: resource temporar
ily unavailable))

ERROR: for isard-database  Cannot start service isard-database: error while creating mount source pa
th '/opt/isard/database': mkdir /opt/isard: read-only file system

ERROR: for isard-hypervisor  Cannot start service isard-hypervisor: driver failed programming extern
al connectivity on endpoint isard-hypervisor (be1c2a4381a216e07c3d38ebf8aa156d8bbcf2ca94bcc612a04cb2
653e055a91):  (iptables failed: iptables --wait -t nat -A DOCKER -p tcp -d 0/0 --dport 5918 -j DNAT
--to-destination 172.18.0.3:5918 ! -i br-d63cf8551e68:  (fork/exec /sbin/iptables: resource temporar
ily unavailable))
ERROR: Encountered errors while bringing up the project.
usuario@isard:~$ _
```

Activamos la virtualización anidada en /etc/modprobe.d/kvm.conf con la linea:

options kvm_intel nested=1

```
usuario@isard:~$ sudo vi /etc/modprobe.d/kvm.conf_
```

```
options kvm_intel nested=1
```

Por último reiniciamos la máquina y volvemos a activar el docker con el comando:

systemctl enabled docker-compose

```
usuario@isard:~$ sudo init 6_
```

```
usuario@isard:~$ sudo systemctl enable docker.service
[sudo] password for usuario:
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-instal
l.
Executing: /lib/systemd/systemd-sysv-install enable docker
usuario@isard:~$ _
```

Ya podemos acceder a la configuración de Isard VDI.

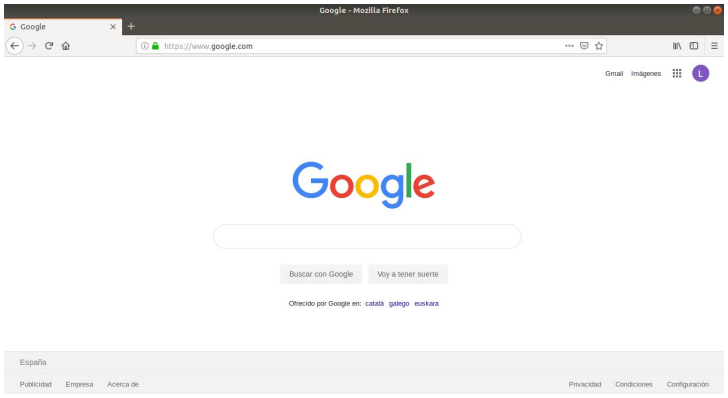
5. Localización e internacionalización

Cuando hablamos de Internacionalización nos referimos al proceso de diseñar un software de tal manera que pueda adaptarse a diferentes idiomas y regiones sin la necesidad de realizar ningún cambio de ingeniería ni en el código.

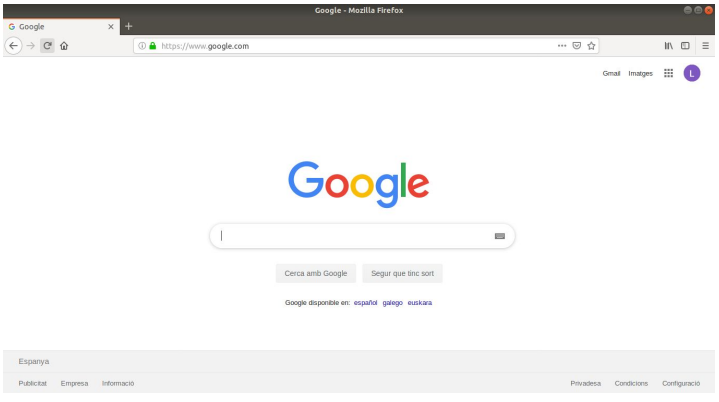
En cambio la localización es el proceso de adaptar un software para una región específica mediante la agregación de componentes específicos de un archivo denominado “locale” y la traducción de los textos. También se le puede denominar regionalización.
Es bastante habitual realizar esta práctica en inglés (sobre todo en el sector de la informática).

5.1 Internacionalización

Podemos definir la internacionalización como un proceso mediante el cual se prepara un elemento para permitir que se adapte a diferentes regiones. En el caso de los programas informáticos nos referimos a preparar el programa para poder traducirlo tanto a varios idiomas cómo utilizar monedas diferentes o usar distintos formatos de fecha etc.
Podemos decir que un programa informático está internacionalizado cuando nos permite adaptarlo a diferentes regiones.



“Página web de Google traducida al castellano”

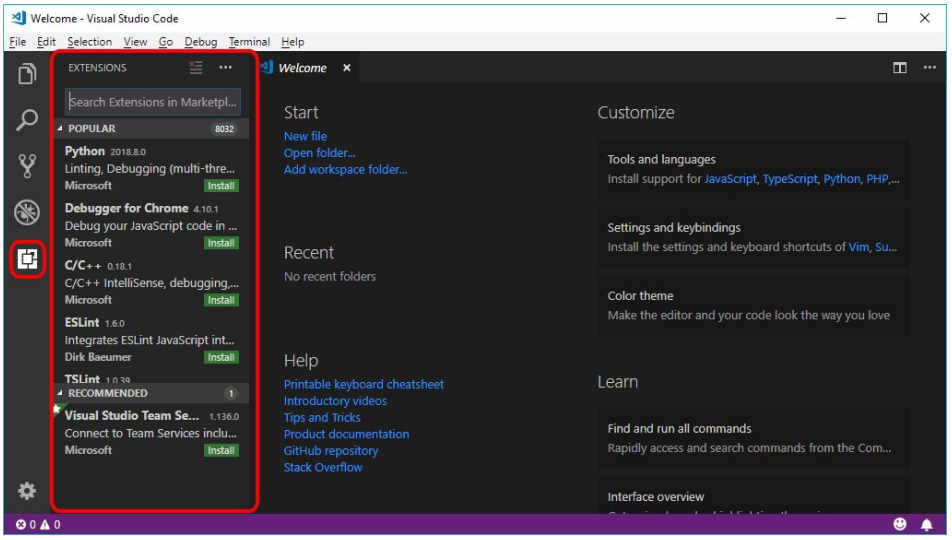


“Página web de Google traducida al catalán”

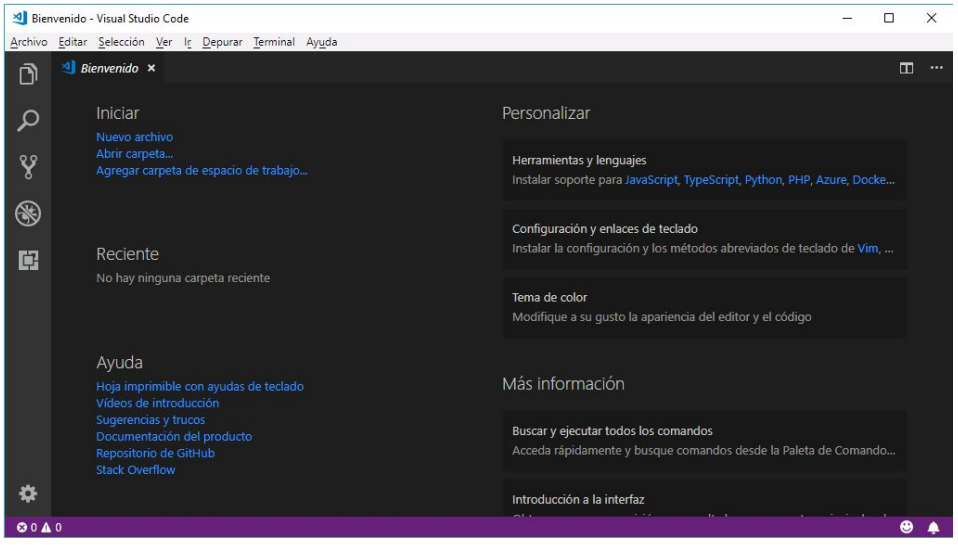
5.2. Localización

Para poder realizar la localización o regionalización de un software antes tenemos que saber si este ha sido internacionalizado.

La localización es el proceso mediante el cual un software internacionalizado se configura para una determinada región, aprovechando las opciones que la internacionalización realizada anteriormente ha permitido. Por ejemplo, la internacionalización puede permitir utilizar distintos formatos de fecha, y en cambio la localización es seleccionar el adecuado para una región. Si decimos «un programa internacionalizado a España» nos estaríamos equivocando, ya que la internacionalización es genérica, también sería erróneo decir «un programa localizado» sin indicar antes o después a qué región se ha localizado el programa mencionado.



"Visual Studio Code en su idioma original"



"Visual Studio Code localizado a España"

5.2.1. Importancia de la localización

Si tenemos un software bien diseñado el cual le hemos dedicado esfuerzo, pasión y sabemos que tiene, la capacidad de impactar de manera positiva en determinado público, después de haber comprobado que funciona, el siguiente paso debe ser la proyección internacional del mismo.

Para esto, la localización ha de ser eficiente y funcional, debemos tener en cuenta que se trata de un proceso que va mucho más allá de la traducción del contenido, sino que habrá que realizar modificaciones en algunas funcionalidades tanto como de elementos de diseño, colores y etc.

En lugar de enfocarnos en la localización de un software también es importante fijarnos en que actualmente las personas que quieran utilizar un ordenador debe primero aprender el inglés pero en un país con bajas tasas de alfabetización lo mencionado anteriormente resulta un gran problema especialmente para las personas con bajos ingresos.
La localización tiene muchos beneficios como por ejemplo la reducción del conocimiento necesario para que los usuarios finales puedan utilizar una aplicación o directamente un equipo informático. Si permitimos que los empleados trabajen en su lengua madre facilitaremos enormemente el manejo de las nuevas tecnologías.

El idioma es un asunto que puede pasa desapercibido pero debemos prestarle la importancia que se merece. Un software en un idioma diferente al del público objetivo, está inevitablemente condenado a convertirse en un gran fracaso.

5.2.2 Las diferentes etapas de la localización

Primera etapa

En la primera etapa se analiza el paquete de localización del cliente, donde se encuentran los archivos ejecutables (en formato .exe, .com, .dll, .bat, .drv o .bin), el entorno de programación del software, posibles notas e instrucciones y etc.
A la hora de elegir el entorno de localización más indicado normalmente, se elige según el formato de los archivos. Para evitar traducir el código por accidente es necesario utilizar una plataforma de localización específica que permita bloquear esas cadenas y por lo tanto facilitar el trabajo. Pero si el cliente solo proporciona el archivo de texto, también se puede utilizar una herramienta de traducción asistida más genérica. Las notas para el traductor pueden indicar diversas especificaciones. Por ejemplo, el cliente puede solicitar no traducir los nombres comerciales de los productos o ciertas secuencias de caracteres, solicitar una traducción en español latino o señalar posibles problemas del proyecto.

Segunda etapa

La siguiente etapa es la relacionada con la traducción de contenidos textuales del software. Normalmente, se traducen primero los elementos que componen la interfaz gráfica, como por ejemplo los cuadros de diálogo, los menús y las cadenas. Luego, la ayuda en línea y los manuales del usuario.
Sin embargo, debido a la gran demanda de trabajo y los cortos plazos que solicitan los clientes, cada vez es más habitual ver a equipos de localización trabajando de forma simultánea.

Para verificar la calidad del trabajo se realiza una revisión de la traducción y pruebas en una etapa que se divide en tres niveles:

Prueba lingüística:

En la prueba lingüística se verifica la consistencia terminológica también se comprueba la visualización correcta del texto en relación los espacios y caracteres, y que todas las cadenas estén traducidas correctamente.

Prueba de la interfaz gráfica del usuario:

En la prueba de la interfaz gráfica se verifica que la versión localizada resulte agradable a nivel de estética.

Prueba funcional:

En la prueba funcional principalmente se realiza para garantizar que el software tiene las mismas funcionalidades que el original y sobretodo que sea compatible con los diferentes dispositivos de hardware a los que esté destinado la aplicación. Realizar las evaluaciones comentadas anteriormente es muy importante, ya que una localización incorrecta puede causar grandes pérdidas económicas y de imagen.

Tercera etapa

Finalmente, se suele pedir al equipo de localización que escriba un informe de cambios, donde redactarán los problemas pendientes por resolver que hayan surgido en alguna de las etapas del proyecto, y que por razones de costes y tiempo no pudieron solucionarse. Más adelante esta información recopilada se utiliza en la localización de versiones siguientes del software para optimizar las etapas del proceso.

6. Ficheros de traducción POT, PO y MO

Una vez ya sabemos que significan los términos localización e internacionalización podemos pasar a hablar de los ficheros de traducción con extensión .pot, .po y .mo.

6.1. Gettext

Existen diferentes tipos de herramientas a la hora de internacionalizar un software , una de ellas es gettext. Gettext es la biblioteca GNU[8] ampliamente usada para escribir programas con interfaz en múltiples idiomas. GNU Gettext tiene diversas implementaciones C, C++, Objective C, sh script, bash script, Python, Java, Pascal, Tcl, Perl y PHP.

Aquí es donde participan los archivos mencionados anteriormente debido a que Gettext utiliza diferentes tipos de ficheros:

6.1.1. Pot

(Portable Object Template, plantillas de objetos portables)

En el archivo con extensión .pot podemos encontrar todas las cadenas de texto traducibles que han sido extraídas anteriormente desde un fichero de código donde habrá un identificador (msgid) y un texto (msgstr) vacío, para generar este fichero se utiliza la funcionalidad xgettext pero también podemos hacerlo manualmente desde nuestro editor de textos favorito.

6.1.2. Po

(Portable Object, objetos portables)

Los .po son ficheros de texto que contienen identificadores (msgid) y el texto (msgstr) que debe aparecer en sustitución del identificador al igual que en los archivos .pot. Por ejemplo, el identificador podría ser “mensajeBienvenida” y el texto “Bienvenidos a nuestro trabajo, espero que disfrutes con nuestra traducción”. Esta traducción a partir de un fichero .pot se puede realizar fácilmente con herramientas de edición/traducción como por ejemplo Poedit. Los ficheros .po a diferencia de los .pot son específicos de un idioma.

[8] Colección de programas informáticos y un sistema operativo de tipo Unix, desarrollado por y para el Proyecto GNU

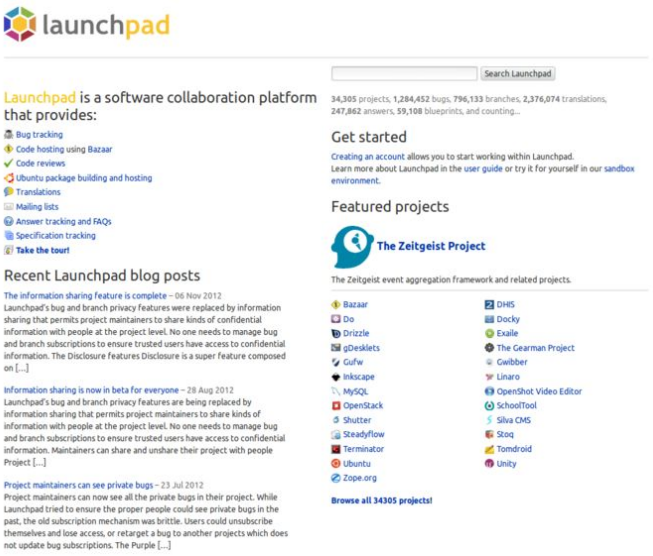
6.1.3. Mo

Finalmente tenemos los ficheros Po que son compilados para generar un archivo binario, con extensión .mo, este fichero nos permite una lectura más rápida de los textos. Una vez realizado este fichero ya se puede distribuir, según la configuración de idioma del programa, este seleccionará de forma adecuada el fichero .mo .

6.2. Aplicaciones para generar ficheros POT, PO y MO

6.2.1. Launchpad

Launchpad es una aplicación bastante antigua y es una Web que permite traducir los mensajes del programa sin tener que trabajar con ficheros POT o PO. El resultado es exportado directamente a un fichero MO. Actualmente, no es una buena opción pero antaño funcionaba bastante bien.



"Página web de Launchpad"

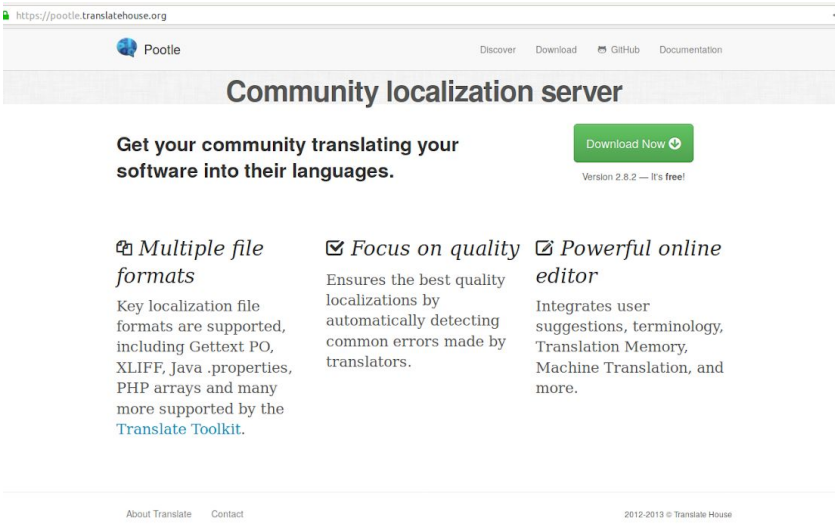
6.2.2. Pootle

Pootle es una herramienta de traducción en línea con una interfaz de gestión de traducción. Está escrito en Python usando el framework[9] Django y es un software libre, originalmente desarrollado y lanzado por Translate.org.za en el año 2004.

Pootle tiene su uso destinado a los traductores de software libre, pero es utilizable en otras situaciones. Su foco principal está en la localización de las interfaces gráficas de usuario aplicaciones, en oposición a la traducción de documentos.

[9] conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

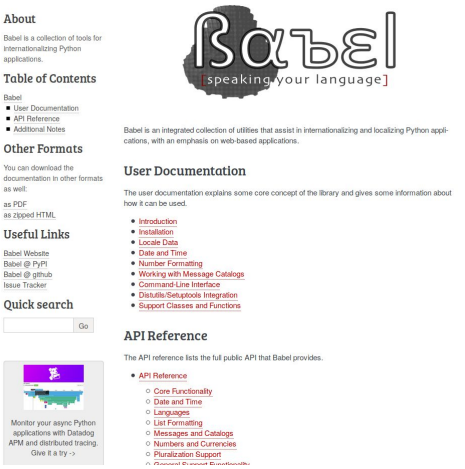
Puede desempeñar diversas funciones a la hora de la traducción. La más simple es que muestra estadísticas para el cuerpo de las traducciones alojadas en el servidor. Tiene un modo de sugerencia que permite a los usuarios hacer sugerencias y correcciones de traducción.



"Página web de Pootle"

6.2.3. Babel

Este es el más reciente de los tres mencionados, Babel es una herramienta herramientas para construir y trabajar con catálogos de mensajes gettext , y una interfaz de Python para el CLDR (Common Locale Data Repository), que proporciona acceso a varios nombres de visualización de la configuración regional, números localizados, formato de fecha, etc.



"Imagen de la página web de Babel"

7. Traducción de los templates.

Finalmente hemos traducido solamente los templates de Isard VDI debido al escaso tiempo que teníamos.
Para poder empezar la traducción primeramente hemos hecho un clon de Isard VDI mediante la herramienta git clone, para poder tener nuestra propia versión y trabajar más cómodamente. Una vez con este clon realizado hemos decidido dividir los ficheros a traducir de los templates, la manera de dividirlo ha sido muy sencilla gracias a la ayuda del otro grupo de compañeros que también realizaban este proyecto. Nosotros hicimos la traducción de los ficheros y directorio:

- Snippets **(directorio)**
- page_404.html
- page_500.html
- sidebar.html
- voucher_validation.htm l

Y el otro grupo realizó el resto de ficheros:

- Pages **(directorio)**
- base.html
- footer.html
- header.html
- login_disposables.html

Entre ambos grupos hemos realizado los ficheros .pot, .po y .mo, para trabajar simultáneamente y ahorrarnos trabajo hemos utilizado una herramienta online llamada “Codeshare” con la que hemos podido ir añadiendo las líneas de código a un mismo fichero formando así el .pot que más tarde clonamos para hacer el .po en otro fichero de “Codeshare”.

Los archivos .po los hemos realizado en dos idiomas diferentes, uno es el castellano y el otro el catalán, es importante destacar que tanto en el fichero .pot como los .po las cadenas estan ordenadas alfabeticamente y el fichero de donde las hemos sacado también. Para ordenarlo todo decidimos hacerlo mediante comentarios (#) cómo se puede ver a continuación:

```
#Las cadenas a traducir estan ordenadas alfabeticamentes.

#+---------- A ------------+

#: alloweds.html:
msgid "Allows"
msgstr ""

#: about.html:
msgid "Admin login"
msgstr ""
```

Ejemplo de las líneas del fichero.pot

```
#: desktops.html:
msgid "Add new"
msgstr ""
```

El orden de la traducción sería:

#: desktops.html → Es el fichero el cual contiene el texto que vamos a traducir
msgid “Add new” → Sería el identificador, el texto original que va a ser traducido al idioma que queremos traducir
msgstr “” → Sería el texto traducido al idioma que deseamos.

A partir del fichero .po hemos creado un fichero .pot que sería igual que el .po sólo que éste tendría el texto ya traducido a su idioma. De forma que en fichero .po está el texto sin la traducción y el fichero .pot estaría el texto + el texto traducido. Para la traducción en catalán y en castellano, lo hemos hecho en distintos documentos separados, aunque contengan casi la información, el documento con la traducción en castellano lo meteremos dentro del directorio “/es/” de español para indicar que es la traducción en español. Y para la traducción en catalán lo meteremos dentro del directorio “/ca/” indicando que proviene del catalán

Como vemos en las siguientes fotografías, en la foto de la izquierda sería el fichero .po con el texto a traducir y en la fotografía de la derecha sería el fichero .pot con el texto traducido.

```
#+---------- G ------------+

#: alloweds_add.html:
msgid "Groups"
msgstr ""

#: domain_graphs.html:
msgid "Graphs"
msgstr ""

#: domain_derivates.html:
msgid "Group"
msgstr ""

#: domain_genealogy.html:
msgid "Genealogy"
msgstr ""

#: sidebar.html:
msgid "General"
msgstr ""

#: user_profile.html:
msgid "Group:"
msgstr ""
```

```
#+---------- G ------------+

#: alloweds_add.html:
msgid "Groups"
msgstr "Grupos"

#: domain_graphs.html:
msgid "Graphs"
msgstr "Graficas"

#: domain_derivates.html:
msgid "Group"
msgstr "Grupo"

#: domain_genealogy.html:
msgid "Genealogy"
msgstr "genealógico"

#: sidebar.html:
msgid "General"
msgstr "General"

#: user_profile.html:
msgid "Group:"
msgstr "Grupo"
```

En la capturas de abajo se puede ver el mismo contenido pero en otro idioma que es el catalán

```
#+---------- G ------------+

#: alloweds_add.html:
msgid "Groups"
msgstr ""

#: domain_graphs.html:
msgid "Graphs"
msgstr ""

#: domain_derivates.html:
msgid "Group"
msgstr ""

#: domain_genealogy.html:
msgid "Genealogy"
msgstr ""

#: sidebar.html:
msgid "General"
msgstr ""

#: user_profile.html:
msgid "Group:"
msgstr ""
```

```
#+---------- G ------------+

#: alloweds_add.html:
msgid "Groups"
msgstr "Grups"

#: domain_graphs.html:
msgid "Graphs"
msgstr "Gràfics"

#: domain_derivates.html:
msgid "Group"
msgstr "Grup"

#: domain_genealogy.html:
msgid "Genealogy"
msgstr "Genealògic"

#: sidebar.html:
msgid "General"
msgstr "General"

#: user_profile.html:
msgid "Group:"
msgstr "Grup"
```

8. Conclusiones

Finalmente una vez realizado este proyecto podemos decir que hemos aprendido bastantes cosas nuevas por ejemplo conceptos como la virtualización que llevábamos viendo desde principio de curso pero gracias a la investigación necesaria para el crédito hemos descubierto muchísima información nueva sobre ello. El descubrimiento de KVM como alternativa a VirtualBox ha sido un gran punto debido a que no conocíamos esta opción y nos ha resultado bastante interesante.
Otra cosa que nos ha gustado es el hecho de poder contribuir en el crecimiento del proyecto Isard VDI, gracias a esto hemos podido conocer e informarnos sobre conceptos como la Localización e internacionalización y sobretodo entender la importancia que tienen tanto para la comodidad como para la adaptación en lugares de baja alfabetización.
Aunque no hemos podido dedicarle todo el tiempo que queríamos a este proyecto, igual nos hemos alegrado de participar en ello y de poder aprender por nuestra cuenta sobre esta nueva propuesta.

Finalmente podemos decir que el proyecto Isard VDI nos parece una muy buena idea y después de habernos informado y visto todo el trabajo que hay detrás de esta interfaz virtual de escritorio nos sentimos gratificados y orgullosos de participar en este proyecto aunque haya sido mínimamente.

9. Bibliografia

Localización e internacionalización:
https://es.wikipedia.org
https://somoslibres.org
https://qabiria.com

Ficheros de traducción POT, PO y MO:
https://es.wikipedia.org
http://www.xperimentos.com
http://www.linuxhispano.net

Virtualización:
https://es.wikipedia.org
https://www.josemariagonzalez.es
http://www.consultaunitpro.com

KVM/Libvirt:
https://www.redhat.com
https://www.ibm.com
http://www.sromero.org
https://es.wikipedia.org

Isard VDI:
https://isardvdi.readthedocs.io