Buildout: crear y desplegar entornos reproducibles en Python

Aitzol Naberan - @aitzol

PyconEs – Zaragoza - 09-10-2014

CodeSyntax

Azitain industrialdea, 3-K
E-20600 Eibar
Tel.: +34 943821780

Internet Solutions
www.codesyntax.com

CodeSyntax

¿Quienes somos?

¿Qué hacemos?

Web!

Bien, vale, pero..

- Sitios de noticias colaborativos
- Sitios para comunidades
- Portales corporativos e intranets
- Sitios multilingües
- I + D

¿Como lo hacemos?

De donde venimos

Allá por el 2001...

Zope

- Definición oficial: Servidor de aplicaciones web orientada a objetos, gratuita, open-source, escrita en Python
- Podemos resumirlo como una manera de publicar objetos python en un entorno web.
- Originalmente desarrollado para mejorar/reemplazar la programación CGI

Zope

- Introdujo el concepto de base de datos orientada a objectos (ZODB) en el 2001
- Programación through-the-web(en el navegador), pero no se lo digáis a nadie
- Versión restringida del interprete de python por razones de seguridad
- DTML y ZPT como lenguaje de plantillas

Zope .../2.5/2.6

- Un fichero tar
- Las primeras versiones incluían su propio interprete python (2.1)
- El desarrollo se hacía mediante "Products":carpetas con un __init__.py y algo de código de inicialización
- Desplegar el desarrollo en una carpeta y reiniciar

Zope 2.7/2.8/2.9

- Desde un único fichero -> instalación
- ./configure && make && make install
- Python del sistema/custom
- Zope instance: una instalación, N instancias
- Instancias ZEO para desacoplar la base de datos

Pero.... necesitábamos más

Plone es un CMS

- Simple y potente
- Flexibilidad en el flujo de publicación
- Seguridad granular (users, groups, roles, permissions)
- Extensible
- Usabilidad

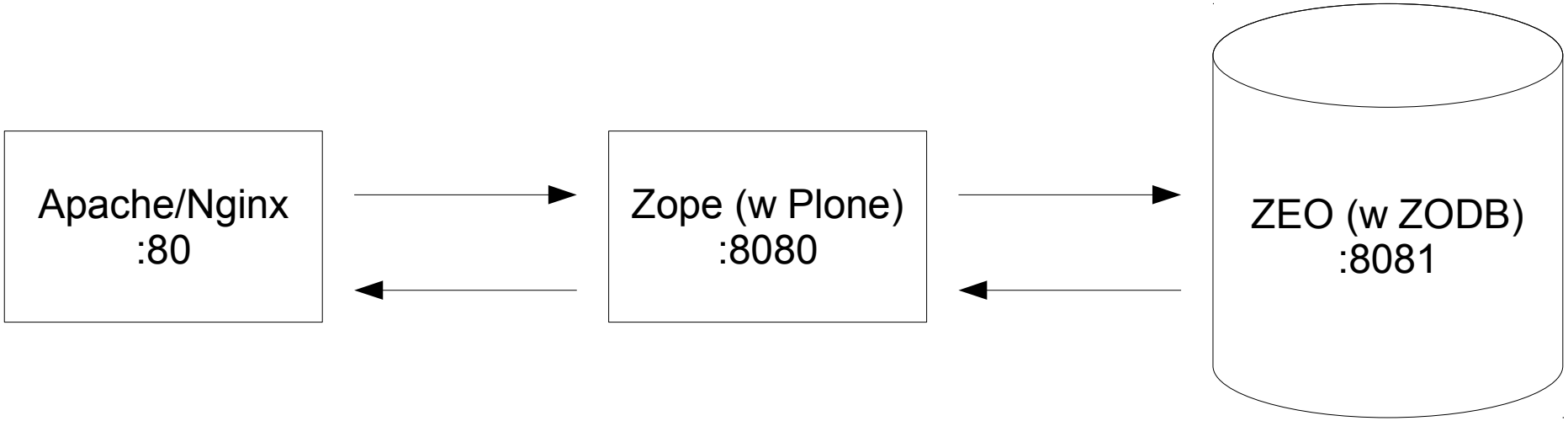
Pero la instalación...

- Plone 1.x, 2.0, 2.1, 2.5, 3.0, 3.1:
  - Tarball → Zope 2.6, 2.7, 2.8, 2.9 and 2.10
- Plone 3 (2007) and 3.1 (2008):
  - Zope → Zope 3: small reusable components
- Plone 3.2 (2009):
  - Tarball → eggs
  - Everyone started developing “python eggs” for Plone

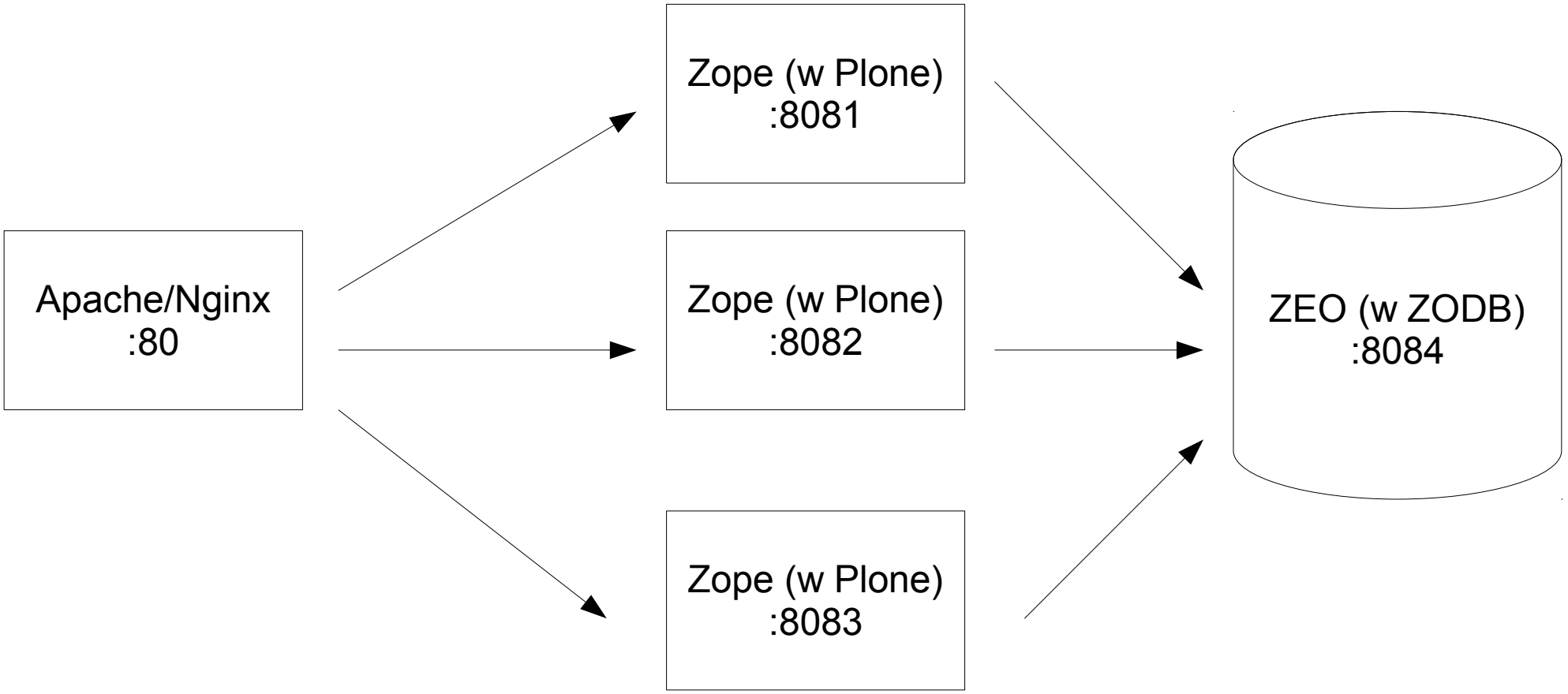
Instalación compleja

- Con la nueva distribución de eggs, la instalación ya no es descomprimir un tar
- Setup.py instala muchos eggs:
  - Plone 3.x: +150
  - Plone 4.x: +250
- Cabe la posibilidad de tener instalaciones complejas

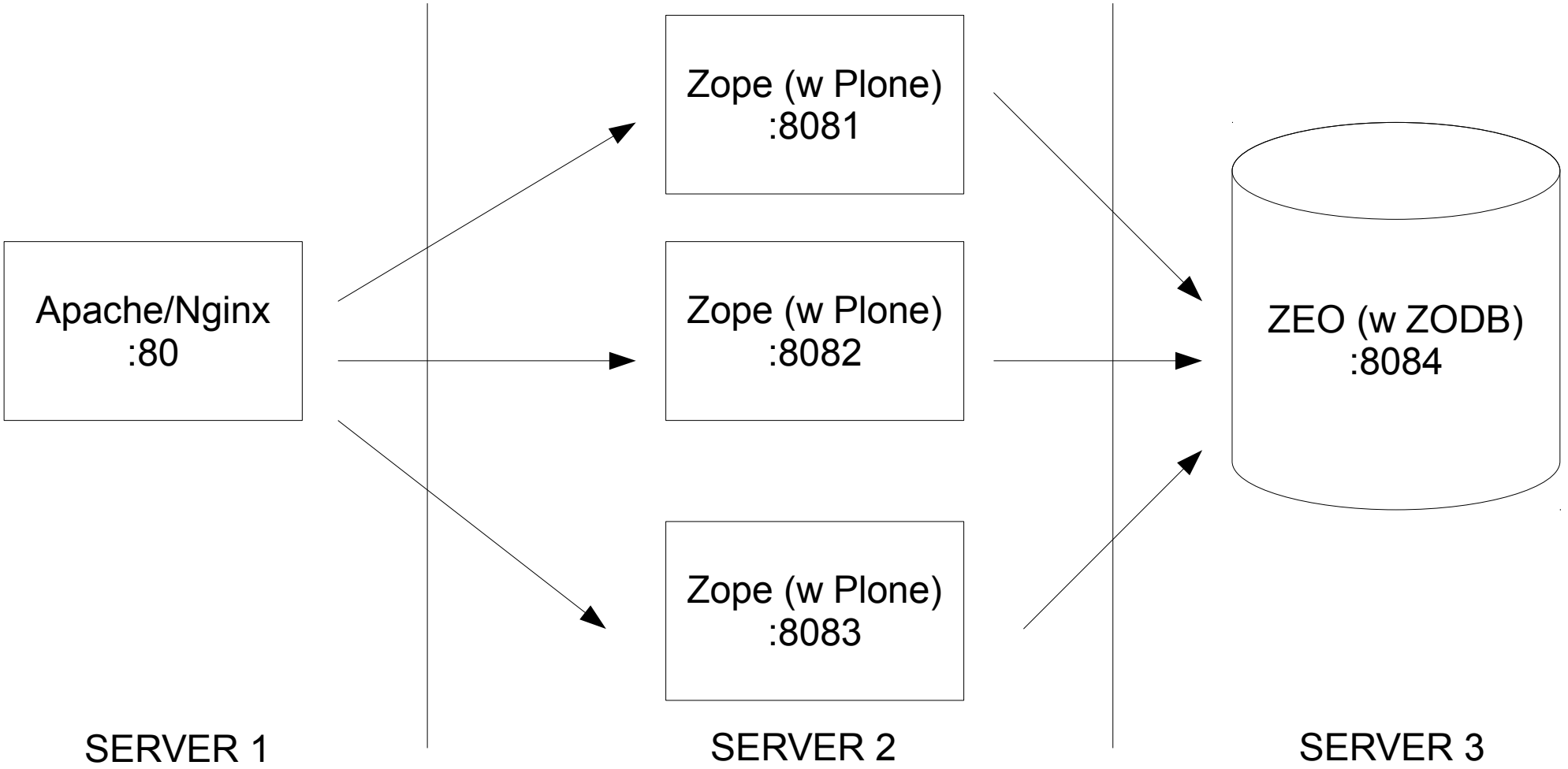
Arquitectura Plone



Arquitectura Plone



Arquitectura Plone



Necesidades

- Despliegue sencillo (2-3 comandos)
- Reproducible:
  - Necesitamos ejecutar el procedimiento bastantes veces (cada vez que añadimos un nuevo egg)
  - Necesitamos ejecutarlo en múltiples servidores (propio ,desarrollo , producción, ...)

Buildout

- Jim Fulton (creador de Zope), crea zc.buildout
- Para solventar 2 problemas:
  - Application-centric assembly and deployment
  - Repeatable assembly of programs from python eggs

¿Cómo lo soluciona buildout?

- Fichero de configuración en formato ini (buildout.cfg)
- Ejecución ordenada de partes (parts)
- Cada parte ejecuta una "receta":
  - Una receta es un huevo python especializado en una tarea (filosofía unix)
- Y ya

Ejemplo (mínimo)

```
[buildout]
parts =
    installation
[installation]
recipe = zc.recipe.egg
eggs = feedparser
interpreter = custompy
```

Crea un interprete python (llamado custompy) con feedparser en el path

Ejemplo: Plone

```
[buildout]
extensions =
    mr.developer
show-picked-versions = true
parts =
    instance
    zeo
    omelette
    zopepy
extends =
    http://dist.plone.org/release/4.3.3/versions.cfg
```

(show plonebuildout.cfg)

CodeSyntax

Basado en http://xkcd.com/303

Entornos reproducibles

- Conviene fijar las versiones exactas de todo lo que se usa
- Defines que te descargas desde donde
- La estructura del sistema de ficheros es siempre la misma
- Si todas las versiones se fijan correctamente, siempre se obtiene la misma instalación

¿Como desarrollamos?

- No tienen porque tener versión:
  - Podemos cambiar de las versiones fijadas a checkouts (svn, git, hg, …) en el sistema de ficheros utilizando una extensión: mr.developer
  - Las extensiones son similares a las recetas
- Activar/desactivar eggs en desarrollo

Y en eso llegó Django

- Definición: high-level python web framework that encourages rapid development and clean, pragmatic desing.
- Normalmente se utiliza para desarrollar soluciones rápidamente contra storage SQL

P: ¿Y usáis *eso* con Django?

- R:Si !
- P: ¿Por qué?
- R: Esencialmente porque llegamos a Django después de Plone, y ya conocíamos como funcionaba. De esta manera utilizamos una única herramienta para todos nuestros despliegues

P: ya pero...

- R: ¿no os he convencido?...
  - Un único sistema de despliegue es preferible a dos.

Ejemplo: Django

```
[buildout]
parts =
      django
      staticfiles
      cron
```

(show djangobuildout.cfg)

Extender buildout

- Las partes se pueden extender desde otros ficheros o mediante opciones (OOP concept)
- Se puede extender over-the-net (extends= http://myserver/buildout.cfg)
- Known Good Set of eggs for proper installation
- Crear tu propia receta

¿Qué hacen las recetas?

- Plone:
  - Crean una instancia de Zope y ponen todos los eggs en el path
  - Configurar puertos, base de datos, blob files, tamaños de cache...
  - Crear instancias de ZEO para desacoplar la base de datos (configurar puertos, paths, etc)

¿Qué hacen las recetas?

- Django:
  - Crea un intérprete con django instalado en el path (wrapper de python manage.py)
  - Ejecución de collect static
  - Crea el script WSGI

¿Qué hacen las recetas?

- General:
  - Ficheros de configuración para Nginx/Apache (basado en plantillas)
  - Descarga paquetes de SCV (svn/git/hg)
  - Descargar tarballs
  - Crear tareas cron
  - Instalar y configurar supervisord

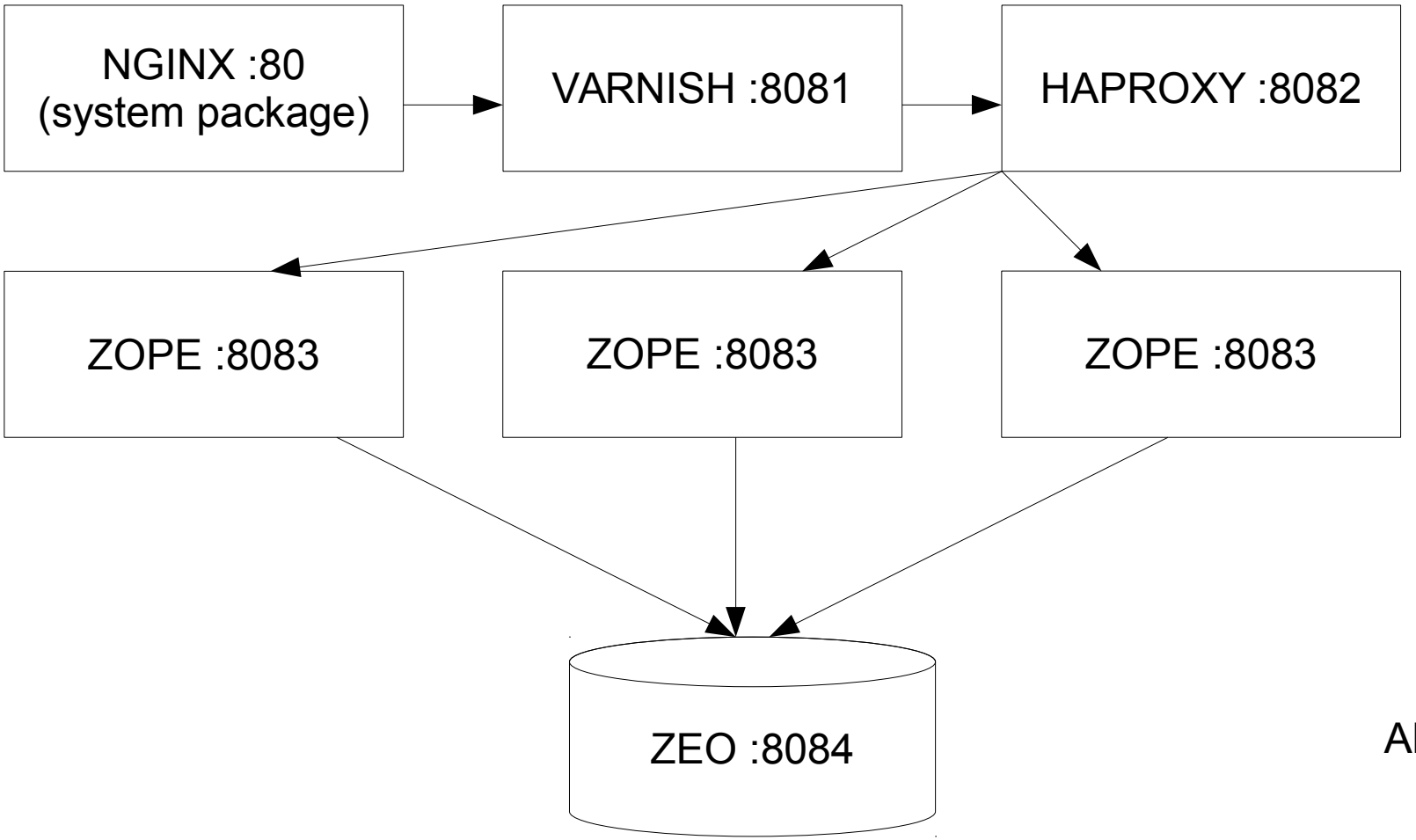
¿Qué hacen las recetas?

- General:
  - Configurar balanceadores (pound/haproxy)
  - Actualizar ficheros *po* y subirlos a Google Docs para su traducción.
  - Crear links a los huevos activos instalados para poder encontrarlos fácilmente (*omellete)*
  - ...

Ejemplo reales

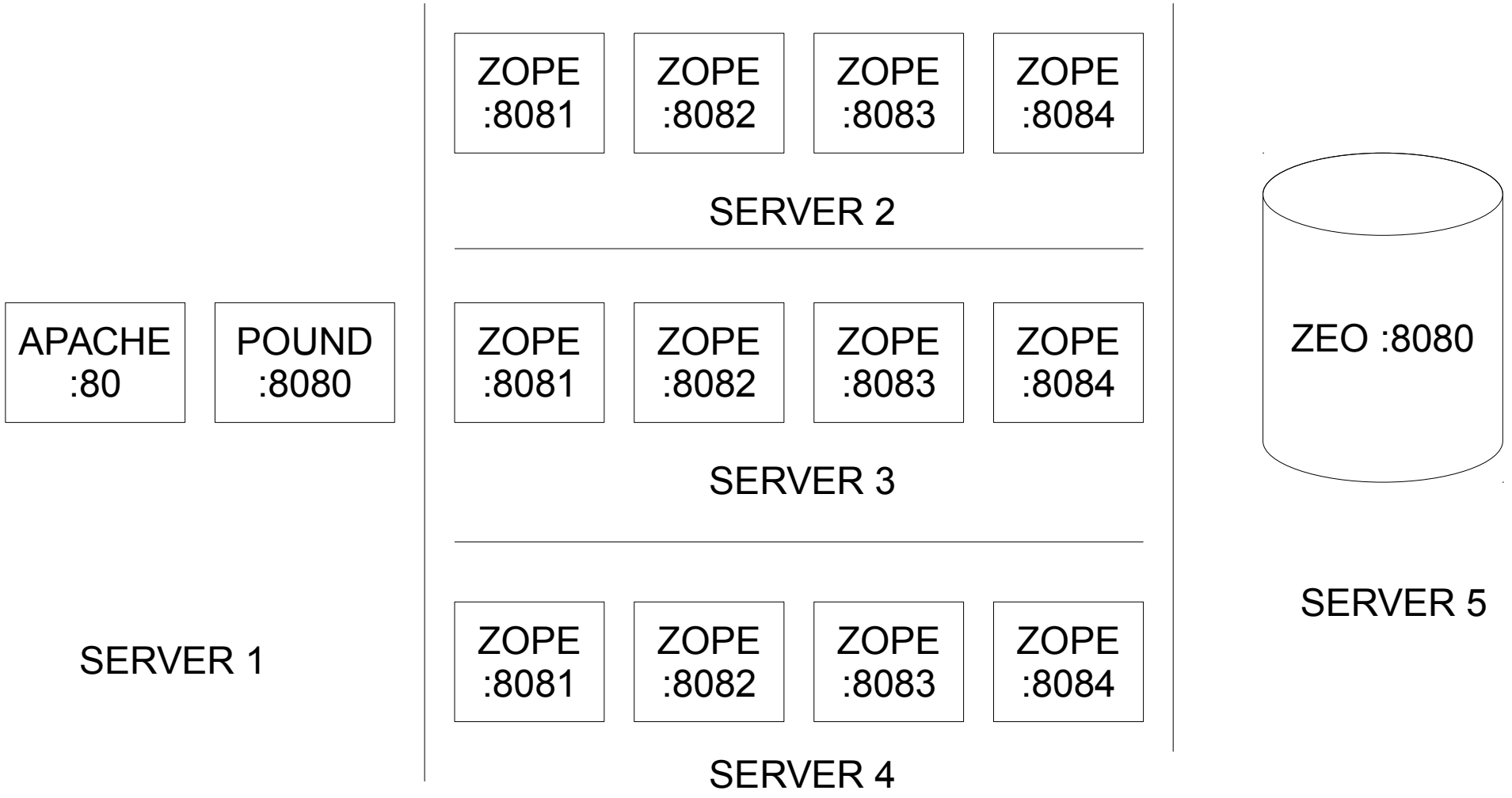
- +120 proyectos con buildout en nuestros servidores
- Todos los desarrolladores conocen como usarlo:
  - ./bin/buildout -vv
- Es predecible

Ejemplo: www.bertsozale.com



All in one server

Ejemplo: EEA


APACHE
:80
POUND
:8080
SERVER 1
ZOPE
:8081
ZOPE
:8082
ZOPE
:8083
ZOPE
:8084
SERVER 2
ZOPE
:8081
ZOPE
:8082
ZOPE
:8083
ZOPE
:8084
SERVER 3
ZOPE
:8081
ZOPE
:8082
ZOPE
:8083
ZOPE
:8084
SERVER 4
ZEO :8080
SERVER 5

Ejemplo: EEA

- 4 servidores, todo configurado con un único fichero buildout.
- https://github.com/eea/esdrt.buildout
- Los sysadmin de EEA Sysadmins adoran la configuración con buildout
  - Repetible
  - No necesitan permisos de root para la instalación

The next cool thing (tm)

- Cada día aparecen nuevas herramientas para la automatización de despliegues
- Fabric, Docker, …
- La inclusión de nuevas herramientas en el equipo puede llegar a ser muy costosa
- ¿Que hacer?
  - El uso de buildout no excluye el uso de fabric (por ejemplo).
  - ¿Fichero fab sencillo? → úsalo
  - ¿muchas lineas de despliegue? → prueba buildout

Buildout & fabric

- Fabric también se usa para la automatización y despliegue de entornos
- El uso de buildout no excluye el uso de fabric (¿he comentado esto?)
- Ej: usar fabric para actualizar y ejecutar buildout:
  - Actualizar el fichero buildout.cfg desde SCV
  - Ejecuta budout -vv
  - Reinicia servicios

Buildout & fabric

- Algunos proyectos de la EEA lo usan
- Nosotros no, ssh + run buildout
- Experiencias:
  - http://tinyurl.com/fab-buildout
  - http://tinyurl.com/fab-buildout-so

Resumiendo

- Configura y despliega tanto proyectos simples como complejos
- Repetible → la instalación siempre es igual en distintos entornos
- Pensado sobre todo para aplicaciones python(Plone, Zope, Django, Pyramid), pero es posible usarlo para otras cosas: Apache, Pound, HAProxy, Memcached, …

¿Preguntas?