

Open APIs
for Open
Minds

Deploy Mediawiki using Fiware Lab facilities

José Ignacio Carretero Guarde

R&D Engineer at Telefonica I+D. In charge of Spain FIWARE Lab Node

joseignacio.carreteroguarde@telefonica.com

Mediawiki and FIWARE Lab APIs (Openstack)

What Mediawiki has to do with FIWARE Lab (I)

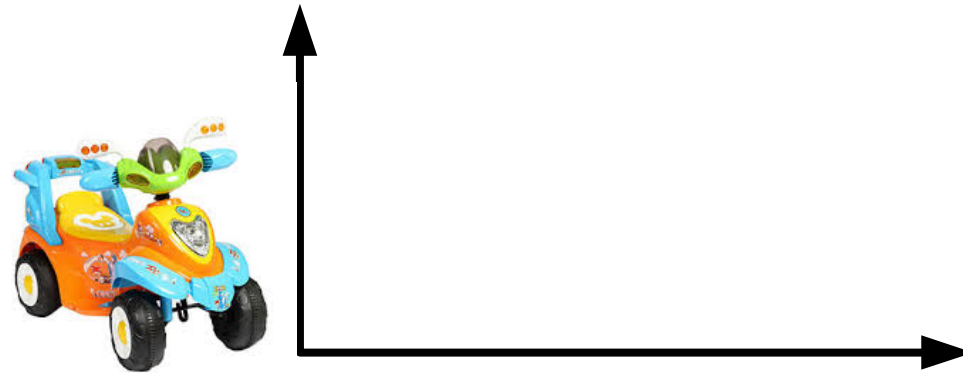


Nothing?



What Mediawiki has to do with FIWARE Lab (II)

...But it could be good vehicle to show some things about FIWARE Lab IaaS reference Implementation GE.



Installing an Application: Mediawiki

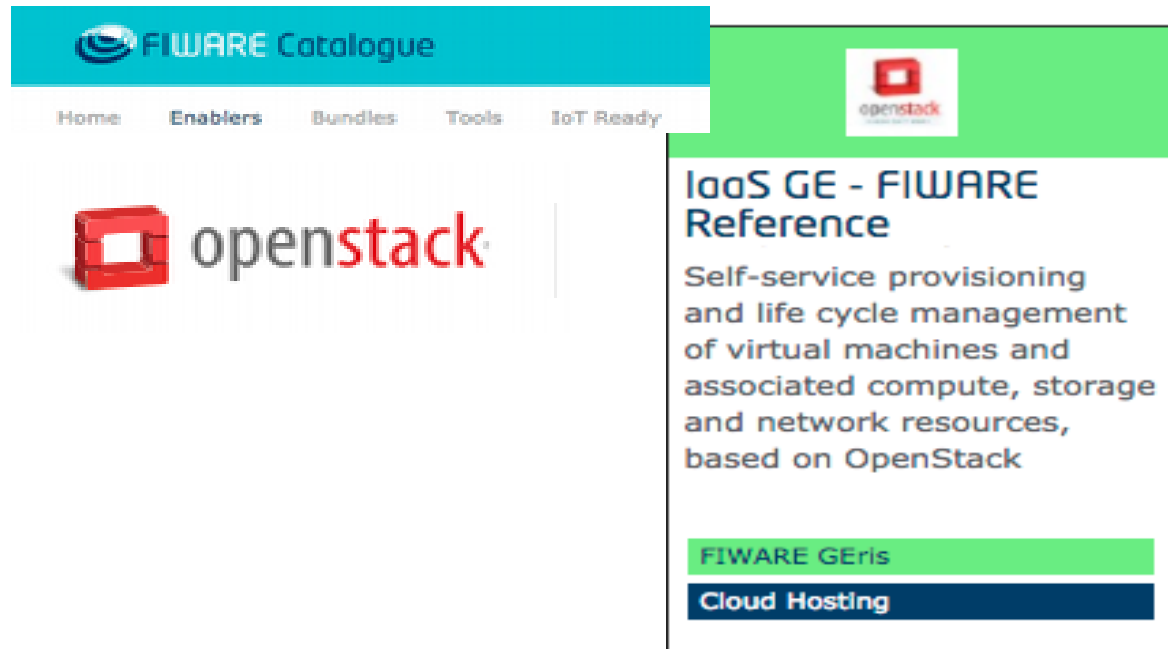
- Why Wikimedia? – If it has nothing to do with FIWARE.
 - It has a backend Database (Mysql)
 - It has a Frontend Web Server (Apache)
 - I Allows me to show some tools: IaaS GE.
- I Know how it Works, So I can automate it.
 - Images and files are stored in Apache (.../wiki/images)
 - Data in the database is stored in /var/lib/mysql

IaaS GE – FIWARE Reference Implementation

- <https://catalogue.fiware.org/enablers/iaas-ge-fiware-reference-implementation/documentation>
 - “We are using OpenStack Vanilla release for the IaaS GE.”

- Openstack APIs work:

- Compute APIs
- Block Storage APIs
- Image Service
- Object Storage



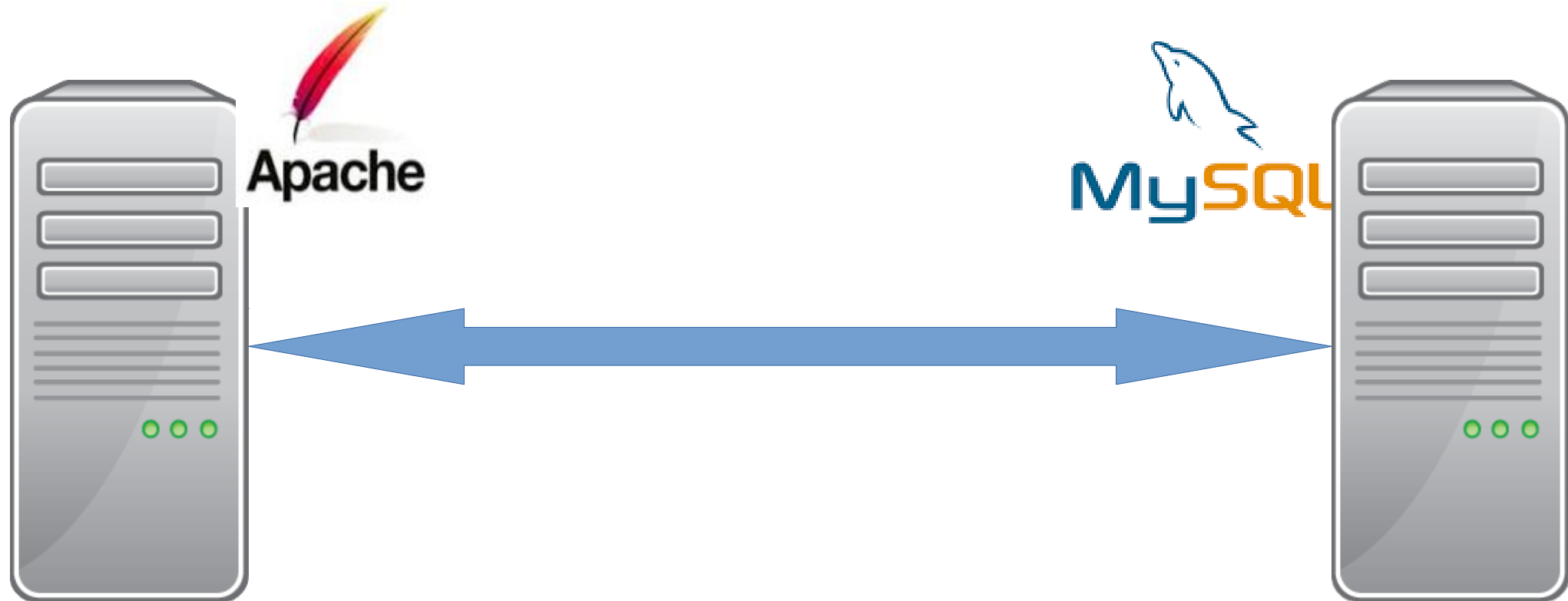
- There are CLIs written in Python...

How to locally install the CLIs

```
# Assuming Python 2.7, pip and virtualenv installed  
virtualenv osdemo  
source osdemo/bin/activate  
pip install python-openstackclient  
pip install python-swiftclient
```

A First Approach to the Installation Process

Installing an Application: Mediawiki (I)



Create a first VM

- Virtual Machines – Instances
 - Instances are managed by Nova.
- We need a few things before deploying an instance
 - A Keypair
 - In order to be Able to SSH our instances
 - Security Groups/Rules
 - Because OpenStack acts like a closed Firewall for every instance.

Nova: Key Pairs and Security Groups

Creating the keypair

```
nova keypair-add summitkp | tee summitkp && chmod 400 summitkp
```

Creating the security group and rules

```
nova secgroup-create openthings opens_some_ports
```

```
nova secgroup-add-rule openthings tcp 22 22 0/0
```

```
nova secgroup-add-rule openthings tcp 80 80 0/0
```

```
nova secgroup-add-rule openthings icmp -1 -1 0/0
```

Creating the security group and rules (What I did!):

```
nova secgroup-create allopen open_everything
```

```
nova secgroup-add-rule allopen tcp 1 65535 0/0
```

```
nova secgroup-add-rule allopen ucp 1 65535 0/0
```

```
nova secgroup-add-rule allopen icmp -1 -1 0/0
```

Create a first VM (ii)

- Boot our first instance with some parameters
 - (At least) One network ID to allow networking
 - Flavor – The size of the Instance
 - Our Security Group – our firewall configuration.
 - A Keypair – To SSH our instance
 - The base image – I'll use an Ubuntu in the demos.
- Assign a floating IP to some of our instances
 - A Floating IP is needed to SSH our instances. We might not be able to SSH private IPs
 - 1 Floating IP is allowed per User.

Nova: Boot and add a floating IP

Find the network ID

```
NETUUID=$(openstack network list | awk '/ node-int-net-01 / {print $2}')
```

Boot Virtual Machine

```
nova boot --nic net-id=$NETUUID --image base_ubuntu_14.04 \  
  --key-name summitkp \  
  --security-groups allopen \  
  --flavor m1.small mydatabase
```

Creating a floating IP:

```
PUBLIC_EXT_NET=public-ext-net-01
```

```
openstack floating ip create $PUBLIC_EXT_NET
```

Associate the IP to our instance:

```
nova list floating ip create $PUBLIC_EXT_NET
```

```
openstack floating list
```

```
nova floating-ip-associate mydatabase 130.206.112.0
```

Nova: Consoles

Get console URL

```
nova get-vnc-console mydatabase novnc
```

Creating the security group and rules

```
nova console-log mydatabase
```

Cinder: Ephemeral Vs Persistent disks

- Compute is thought to Compute
 - When Instances die, Ephemeral disks die
- Persistent Disk – Block Storage
 - Cinder manages Persistent Disks
 - The disks, once created can be attached to instances
 - Instances must format (once) and mount the disks before use
 - Persistent disk will survive Instances.
- A way to Keep data apart from computation.

Cinder: Persistent Disk Creation

Create a Persistent disk

```
openstack volume create --size 1 myvolume
```

```
openstack volume list
```

```
VOLUMEID=$(openstack volume list | awk '/ myvolume / {print $2}')
```

Attaching the volume to the instance

```
nova volume-attach mydatabase $VOLUMEID
```

Or attaching at boot time...

```
nova boot --nic net-id=$NETUUID --image base_ubuntu_14.04 \  
  --key-name summitkp \  
  --security-groups allopen \  
  --block-device-mapping vdb=$VOLUMEID \  
  --flavor m1.small mydatabase
```


Automate: Why?

- Increase Productivity
 - Automated processes reduce defects
 - Reduce Human Errors
 - Processes are run effortlessly
 - Processes are more flexible => Changes are easier
 - Increase satisfaction
 - ... You can write many other things here
- ... And here.

Automate: Using GUI (Cloud Portal)

Easy to use

Launch New Instance Actions

Images

Name	Actions
base_centos_6	Launch
base_ubuntu_14.04	Launch
base_ubuntu_12.04	Launch
base_debian_7	Launch
base_debian_8	Launch

Launch Instances

1. Details 2. Access & Security 3. Networking 4. Post-Creation 5. Summary

Instance Name *
tooslowtobegoodenough

Flavor
m1.small

Instance Count *
1

Description
Specify the details for launching an instance. The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name	m1.small
VCPUs	1
Root Disk	20 GB
Ephemeral Disk	0 GB
Total Disk	20 GB
RAM	2048 MB

Project Quotas

Instance Count (0)	3 Available
VCPUs (0)	6 Available
Memory (0 MB)	25000 MB Available

* Mandatory fields. Cancel Next

(maybe...)
Too many clicks

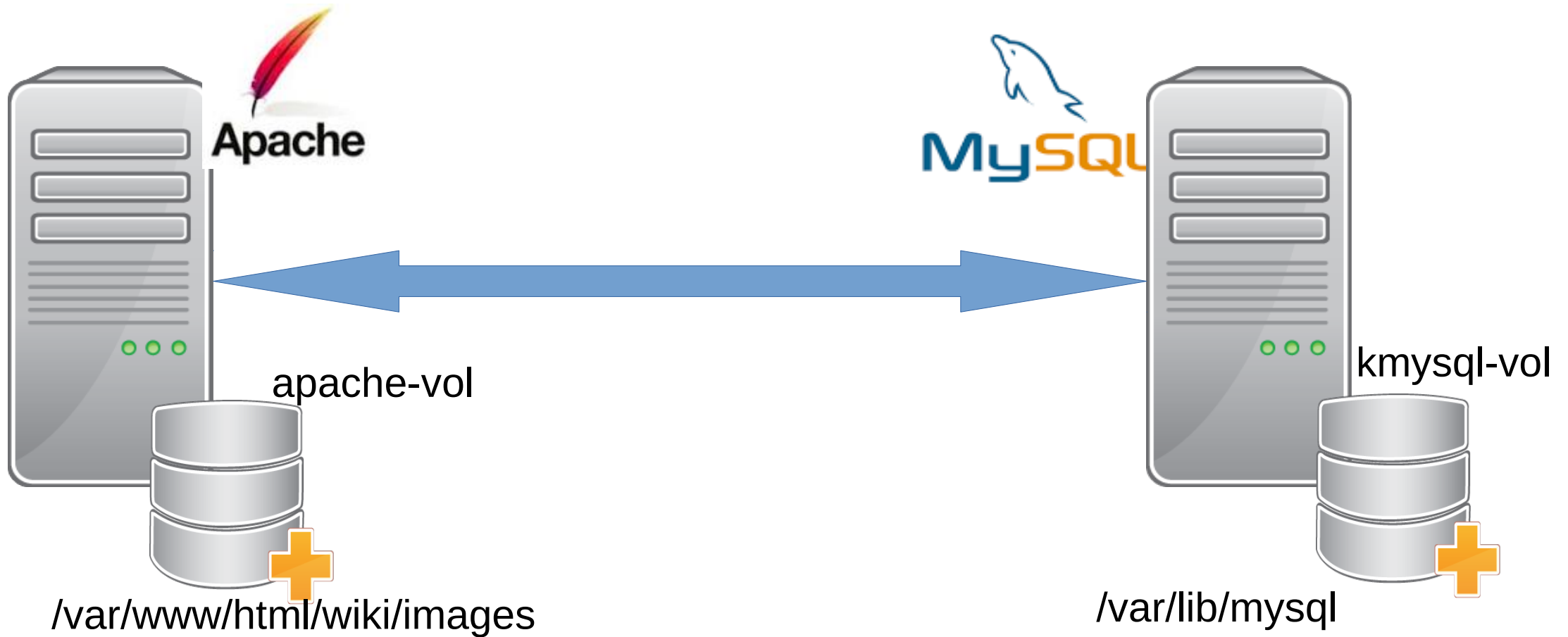
(maybe)...
Too error prone

(almost)...
Impossible automation

Automate: How?

- Tools to automate
 - Puppet/Chef
 - An Agent is required in the instance
 - Ansible
 - Many features, even some for OpenStack
 - Some OpenStack Tools
 - Heat / Murano
 - Ad Hoc Scripts
 - This time it was my option, so I can show some CLI commands

Installing an Application: Mediawiki (II)



One Installation Process: Database

- Create the Persistent Disk (if it doesn't exist)
- Boot the VM attaching the Persistent Disk and injecting a Script
- The Script
 - Formats the Persistent Disk and mounts it (using /etc/fstab)
 - /var/lib/mysql
 - Installs MySQL-Server software
 - Creates Database and user for the Wiki.

One Installation Process: Apache

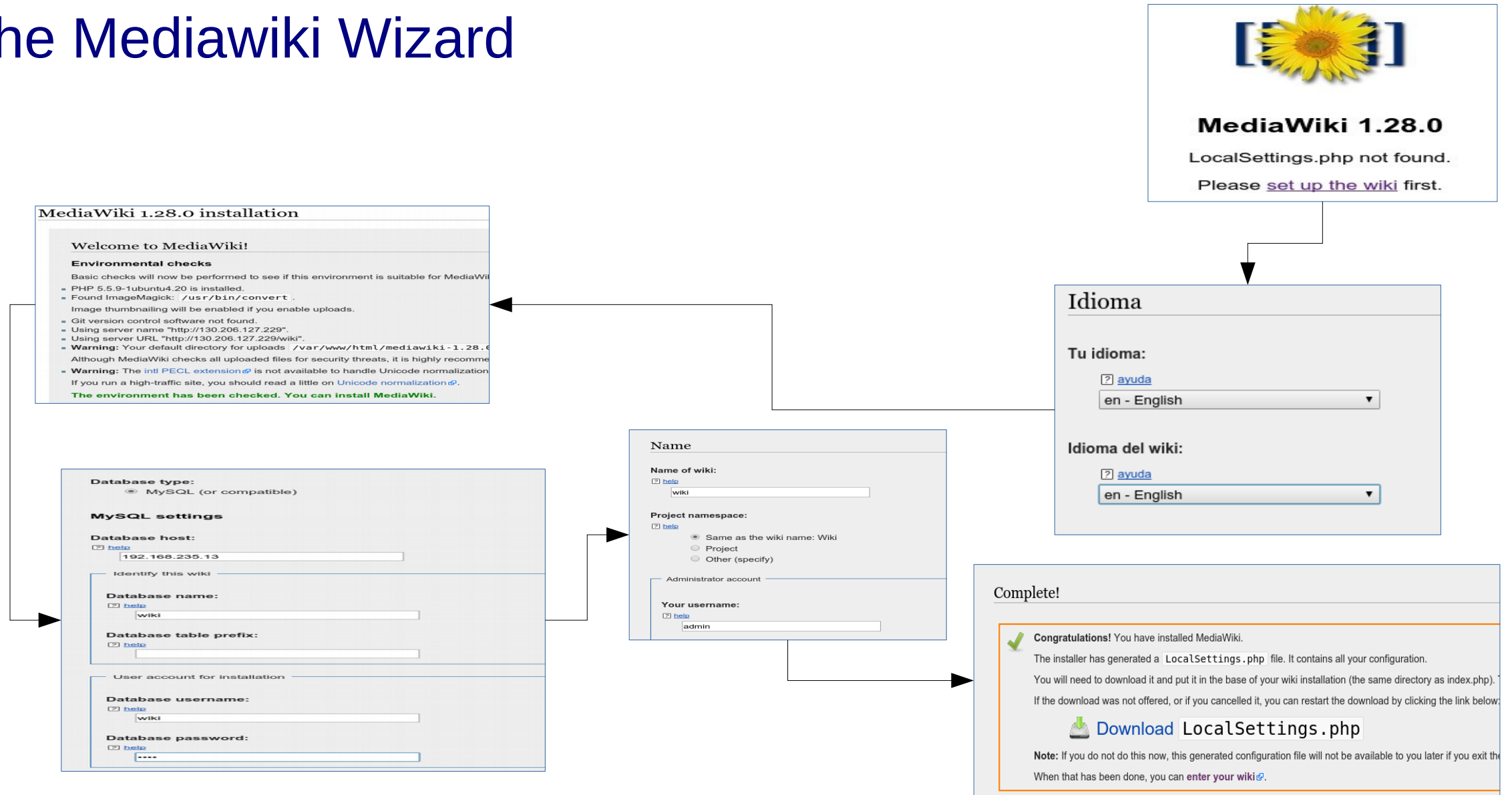
- Create the Persistent Disk (if it doesn't exist)
- Boot the VM attaching the Persistent Disk and injecting a Script
- The Script
 - Formats the Persistent Disk
 - Installs Apache, php5, libs, etc
 - Downloads Wikimedia and untars it `/var/www/html`
 - Downloads SyntaxHighlight_plugin and untars it.
 - Mounts the disk (`/var/www/html/wiki/images`)
- Creates a FloatingIP and associates it to the instance

The new boot command for Nova

Or attaching at boot time...

```
nova boot --nic net-id=$NETUUID --image base_ubuntu_14.04 \  
  --key-name summitkp \  
  --security-groups allopen \  
  --block-device-mapping vdb=$VOLUMEID \  
  --user-data oneScript \  
  --flavor m1.small mydatabase
```


The Mediawiki Wizard



Can't automate everything... but almost

- Mediawiki installation can't be automated... but
- I can automate LocalSettings.php configuration
- The Script
 - Gets the Public IP of our apache
 - Sets the Logo, Extensions of Documents which can be uploaded
 - Configures SyntaxHighlight extension
 - Uploads LocalSettings.php and logo.jpg

Some other tools: Images and Object Storage

Snapshots from instances

- I can take Snapshots of my instances
 - So I can boot a preconfigured Instance
 - Glance - The image service is used to create Snapshots
- I've taken an Snapshot of my Database Instance
 - The process takes some time...
 - It is a complicated process that involves many subprocesses


Image Commands

```
# Create an Snapshot from an Instance  
nova image-create krtmysql krtmysql-snp
```

```
# List images  
openstack image list
```

```
# Delete images  
openstack image delete $IMAGE_ID
```

Object Storage

- It stores static Objects
 - We can retrieve the objects in the future
 - Swift is the reference implementation of FIWARE's Object Storage GE
 - <https://catalogue.fiware.org/enablers/object-storage-ge-fiware-implementation>
 - Objects are Stored in containers
- I've uploaded my Preconfigured Mediawiki in a .tgz file to Object Store
 - Let's use this in next Mediawiki installations (as a demo)  FIWARE

Swift Commands

Create a new Container

```
swift post summit
```

Upload an object to the Container

```
swift upload summit wiki.tgz
```

List Containers, list objects from a container

```
swift list
```

```
swift list summit
```

Retrieve an object from a container

```
swift download summit wiki.tgz
```

Swift Commands (to automate)

Retrieve an object from a container.. In some scripts.

```
token=$(openstack token issue | \
    awk '/ id / || / project_id / {print $4}')
```

```
TOKEN=${token[0]}
```

```
TENANT_ID=${token[1]}
```

```
URL=http://130.206.112.3:8080/v1
```

...

Use the token, and the tenant.

```
swift --os-auth-token $TOKEN --os-storage-
```

```
url=$URL/AUTH_$TENANT_ID download summit wiki.tgz
```


<https://github.com/jicarretero/SummitMalaga2016>

| Thank you!

<http://fiware.org>

Follow @FIWARE on Twitter

