Reporting

Multi customers and/or datacenters: REALMS

Shinken Enterprise's architecture like we saw allows us to have a unique administration and data location. All pollers the hosts are cut and sent to schedulers, and the pollers take jobs from all schedulers. Every one is happy.

Every one? In fact no. If an administrator got a continental distributed architecture he can have serious problems. If the architecture is common to multiple customers network, a customer A scheduler can have a customer B poller that asks him jobs. It's not a good solution. Even with distributed network, distant pollers should not ask jobs to schedulers in the other continent, it's not network efficient.

That is where the site/customers management is useful. In Shinken Enterprise, it's managed by the **realms**.

A realm is a group of resources that will manage hosts or hostgroups. Such a link will be unique: a host cannot be in multiple realms. If you put a hostgroup in a realm, all hosts in this group will be in the realm (unless a host already has the realm set, the host value will be taken).

A realm is:

- at least a scheduler
- at least a poller
- can have a reactionner
- can have a broker
- can have a recevier

In a realm, all realm pollers will take all realm schedulers jobs.

Important: there is only ONE arbiter (and a spare of course) for ALL realms. The arbiter manages all realms and all that is inside.

Sub-realms

A realm can have sub-realms. It doesn't change anything for schedulers, but it can be useful for other satellites and spares. Reactionners and brokers are linked to a realm, but they can take jobs from all sub-realms too. This way you can have less reactionners and brokers (like we soon will see).

The fact that reactionners/brokers (and in fact pollers too) can take jobs from sub-schedulers is decided by the presence of the manage_sub_realms parameter. For pollers the default value is 0, but it's 1 for reactionners/brokers.

An example

To make it simple: you put hosts and/or hostgroups in a realm. This last one is to be considered as a resources pool. You don't need to touch the host /hostgroup definition if you need more/less performances in the realm or if you want to add a new satellites (a new reactionner for example).

Realms are a way to manage resources. They are the smaller clouds in your global cloud infrastructure.

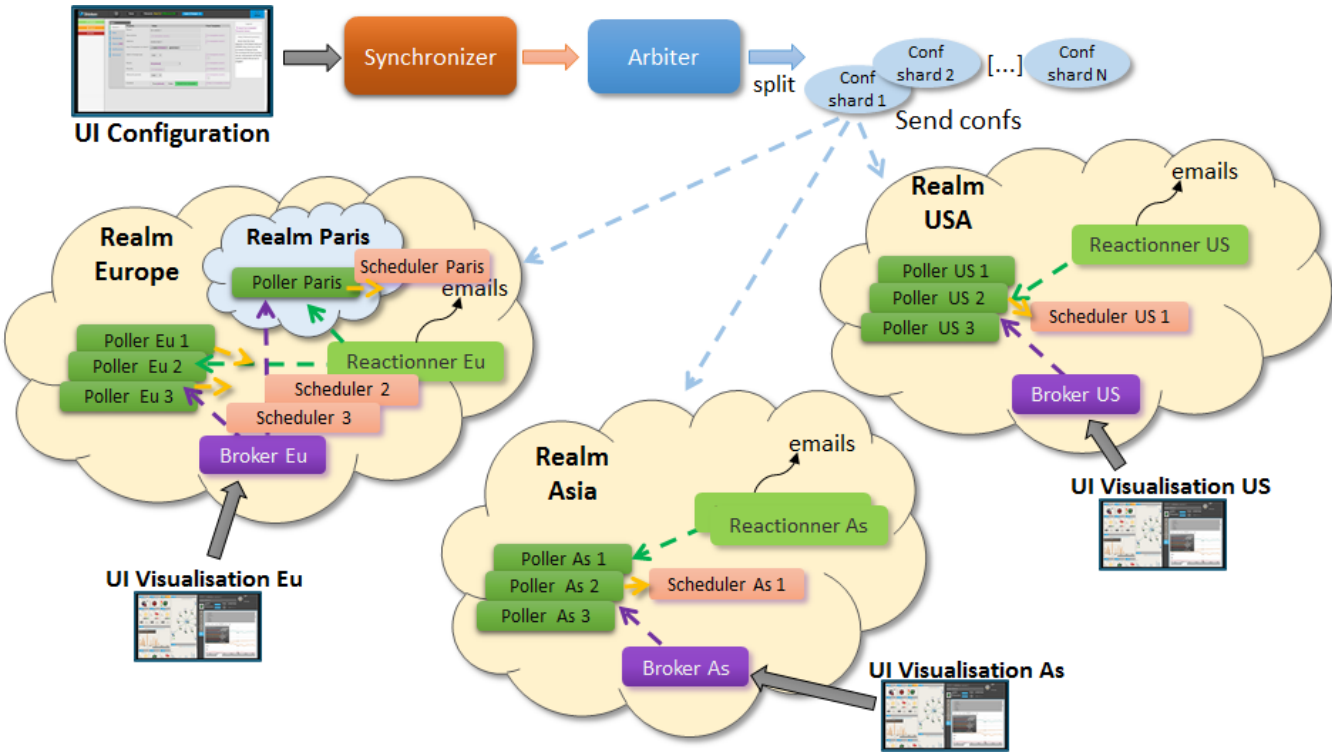
If you do not need this feature, that's not a problem, it's optional. There will be a default realm created and every one will be put into.

It's the same for hosts that don't have a realm configured: they will be put in the realm that has the "default" parameter.

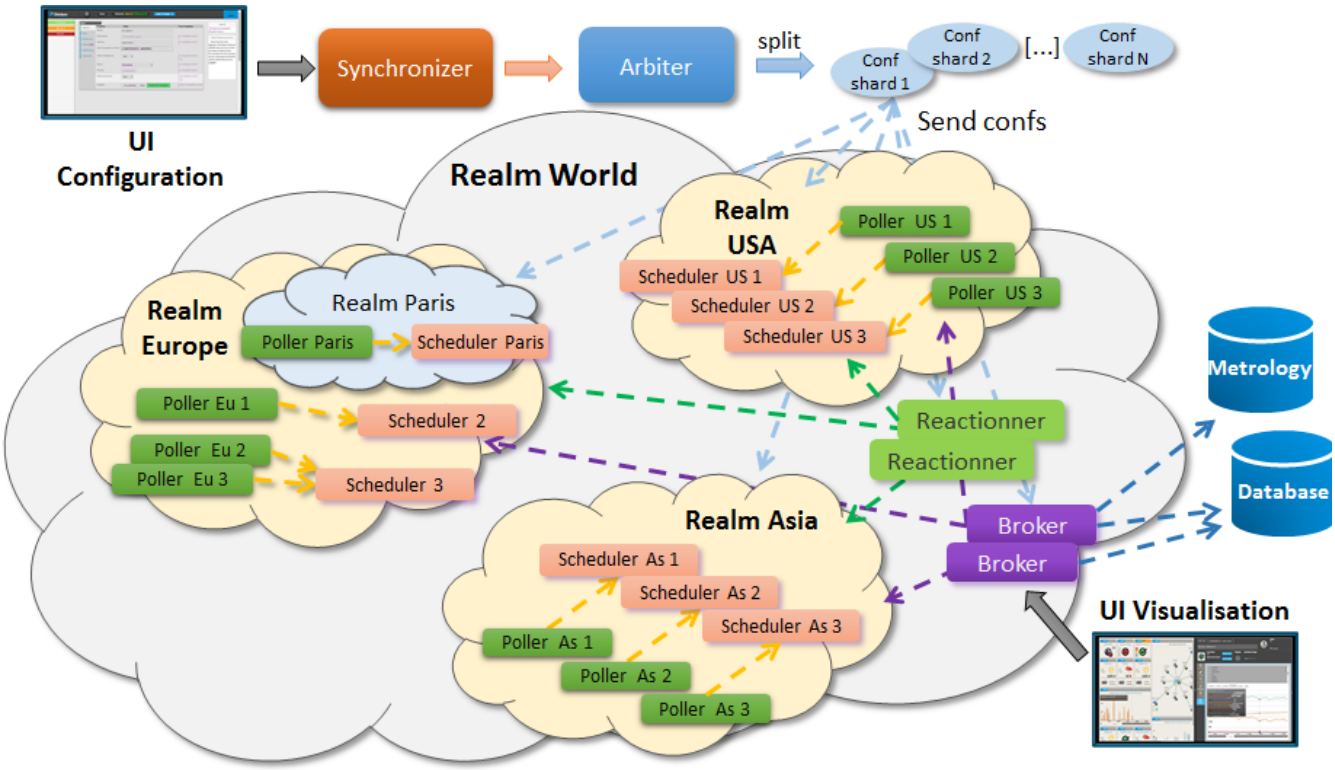
Picture example

Let's take two examples of distributed architectures around the world. In the first case, the administrator don't want to share resources between realms. They are distinct. In the second, the reactionners and brokers are shared with all realms (so all notifications are send from a unique place, and so is all data).

Here is the isolated one:



And a more common way of sharing reactionner/broker:



As you can see, all elements are in a unique realm. That's the sub-realm functionality used for reactionner/broker.

Variable Descriptions

Property	Default	Description
realm_name	N/A	This variable is used to identify the *short name* of the realm.
realm_members	N/A	This directive is used to list realm sub-realms.
broker_complete_links	0	If 1, then the brokers will be allowed to talk to schedulers taht already have a broker defined (in its realm or upper). So enabling this option allow a scheduler to export its data to all the broker in its realm level or upper, whatever their number are.
default	0	This directive is used to define if the realm is the default one or not. Only one default realm is allowed.

Example Definition

You first realm: World

```
define realm{
    realm_name     World
    realm_members  Europe,America,Asia
    default        0
}
```

World and its Sub-Realm (configuration)

Here is the configuration for the shared architecture:

Realm

```
define realm {
  realm_name      World
  # Now you define SUB REALMS of World
  realm_members   Europe,US,Asia
  # Element without explicit realm setting will be set in the World realm
  default         1
}

# We define our SUB REALMS
# EUROPE
define realm{
  realm_name       Europe
# This one have it's own SUB REALM
  realm_members    Paris
}
# Paris: sub realm for Europe
define realm{
  realm_name       Paris
}

# USA
define realm{
  realm_name       USA
}

# Asia
define realm{
  realm_name       Asia
}



# For example the daemons for the Paris realm

define scheduler{
  scheduler_name     scheduler_Paris
  realm              Paris
}

# Example of a TOP level realm (WORLD) daemon that can reach daemons of the SUB realms
# so will reach Europe, Paris, USA and Asia
define reactionner{
  reactionner_name     reactionner-master
  realm                World
}
```

And the host link into a realm will be on the host configuration page:

? Unknown Attachment

Multi levels brokers

In the previous samples, if you put numerous brokers into the realm, each scheduler will have only one broker at the same time. It was also impossible to have a common Broker in All, and one brokers in each sub-realms.

You can activate multi-brokers features with a realm parameter, the **broker_complete_links option** (0 by default).

You will have to enable this option in ALL your realms! For example:

Realm

```
define realm{
realm_name Europe
broker_complete_links 1
}
```

This will enable the fact that each scheduler will be linked to each brokers. This will make possible to have dedicated brokers in a same realm (one for the web interface, another for Graphite for example). It will also make possible to have a common Broker in "All", and one broker in each of its sub-realms (Europe, US and Asia). Of course the sub-brokers will only see the data from their realms, and the sub-realms (like Paris for Europe for example).