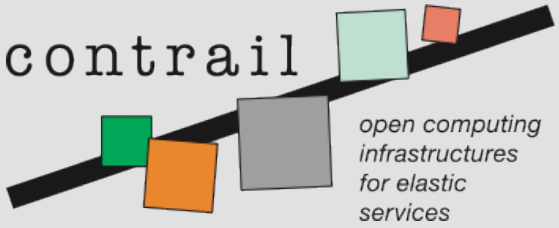


ConPaaS Architecture

Emanuele Rocca
Vrije Universiteit Amsterdam

June 13th 2013

contrail is co-funded by the EC
7th Framework Programme
under Grant Agreement nr.
257438

Software Architecture

The important stuff, whatever that is.
– Fowler, M.

The important stuff

- Services
- Managers
- Agents
- Core
- Applications
- Manifests
- Director
- CLI client
- Frontend

Software Architecture

Not a set of models or structures,
but the rationale behind them.

What is ConPaaS?

ConPaaS is a Free and Open Source Platform-as-a-Service system.

Platform-as-a-Service systems allow developers to deploy their applications "in the cloud".

What kind of applications?

Platform-as-a-Service systems tend to focus specifically on web applications.

ConPaaS takes a more generic approach, supporting different kind of applications and programming models.

Web applications

MapReduce

Bag-of-Tasks

Applications that execute independent parallel tasks.

Task Farming service.

What kind of applications?

Different types of applications and programming models can be combined at will.

For example, a scientific application can be written using the MapReduce programming model, while providing a web frontend written in PHP/MySQL.

What do they have in common?

They benefit by running on a arbitrary number of virtual machines.

- elasticity
- high availability

Other examples?

Client - server applications.

Deploying the server part is tedious and error-prone.

Developers/sysadmins might want to use ConPaaS instead.

Services

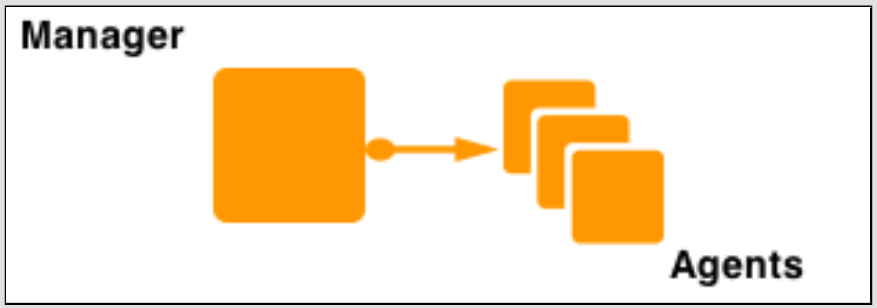
ConPaaS Services

To support such a diverse set of use cases we have introduced the concept of **Services**.

A service is composed by one **Manager** and multiple **Agents**.

ConPaaS Services

The Manager is responsible for adding and removing agents, as well as for other administrative tasks.



Agents do the real work and accept calls from their Manager only.

ConPaaS Services

Managers and Agents in practice:

- VM executing on a cloud provider running a manager/agent process
- TCP/IP process exposing methods via HTTPS/JSON
- Python class implementing the process behavior

ConPaaS Services: Manager

Exposed methods:

- startup
- get_logs
- add_nodes
- remove_nodes
- list_nodes
- get_node_info
- shutdown

ConPaaS Services: Manager

Service-specific, MySQL:

- load_dump
- set_password

ConPaaS Services: Agent

Exposed method:

- check_agent_process

ConPaaS Services: Agent

Service-specific, MySQL:

- setup_master
- setup_slave
- load_dump
- set_password

ConPaaS Services

Adding support for a new service to ConPaaS means writing two Python classes.

Why?

ConPaaS Services

Different ConPaaS services actually have a lot in common.

Manager and Agent classes do not have to be written completely from scratch.

Inherit from **ConPaaS Core**.

Core

ConPaaS Core

All the service-independent code.

ConPaaS Core

conpaas.core.manager.BaseManager

conpaas.core.agent.BaseAgent

ConPaaS Core

- IaaS
- HTTPS
- IPOP
- Ganglia
- ...

ConPaaS Core: multiple clouds

conpaas.core.clouds.base.BaseCloud

conpaas.core.clouds.opennebula.OpenNebulaCloud

conpaas.core.clouds.ec2.EC2Cloud

ConPaaS Core: IPOP

IP over P2P.

Easily deploy VPNs across multiple domains.

ConPaaS uses IPOP to create per-**application** VPNs.

Applications

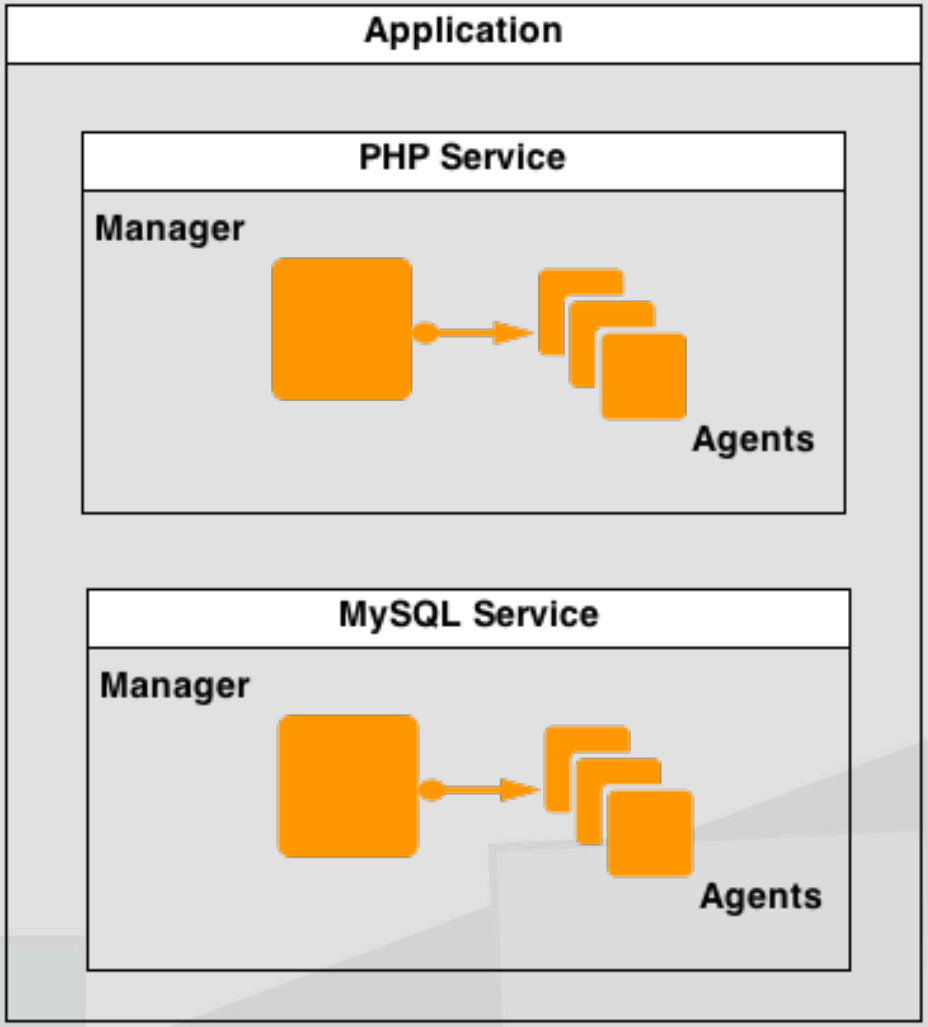
Applications

Users do not want to deploy a web server and a database system.

Users want to run WordPress.

We need to put services together.

Applications



Applications

- set of services
- belonging to a user
- with a name

Application Manifests

Files describing which services are needed to create a certain ConPaaS application.

JSON data format.

Application Manifests: Sudoku

```
{
 "Description" : "Sudoku example",

 "Services" : [
  {
   "ServiceName" : "PHP sudoku backend",
   "Type" : "php",
   "Start" : 0,
   "Archive" : "http://www.example.org/sudoku.tar.gz"
  }
 ]
}
```

Application Manifests: MediaWiki

```
{
 "Description" : "Wiki in the Cloud",

 "Services" : [
  {
   "ServiceName" : "Wiki-Webserver",
   "Type" : "java",
   "Archive" : "http://example.org/scalaris-wiki.war",
   "Start" : 1
  },
  {
   "ServiceName" : "Wiki-Database",
   "Type" : "scalaris",
   "Archive" : "http://example.org/wikipediadump",
   "Start" : 1
  }
 ]
}
```

Director

Director

Keeps track of:

- users
- credentials
- applications

Director

Director

- Python/Flask application
- Deployed to Apache
- HTTPS/JSON
- ConPaaS API server

Director

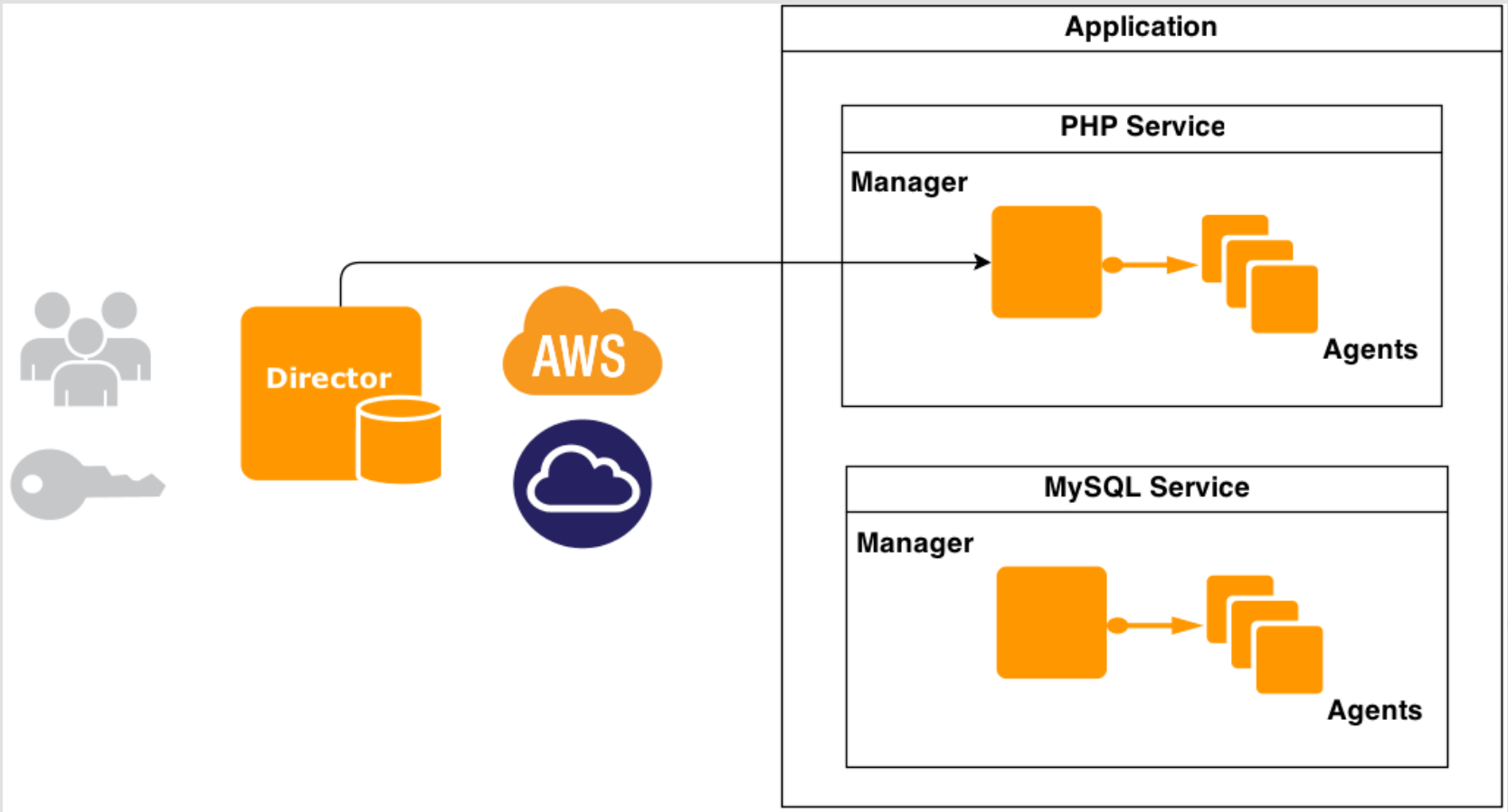
Handles the life-cycle of ConPaaS applications.

Director

Certification Authority issuing certificates for:

- users
- managers
- agents

Director


Application
PHP Service
Manager
Agents
MySQL Service
Manager
Agents
Director
AWS

Clients

CLI client

- Python CLI application
- HTTPS/JSON to talk with the Director
- Uses functions from Core
- Opens the door to "scriptability"

CLI client

```
Usage: /usr/local/bin/cpsclient.py COMMAND [params]
COMMAND is one of the following

    credentials                          # set your ConPaaS credentials
    listapp                              # list all applications
    available                            # list supported services
    clouds                               # list available clouds
    list              [appid]            # list running services under an application
    deleteapp         appid              # delete an application
    createapp         appname            # create a new application
    manifest          filename           # upload a new manifest
    download_manifest appid              # download an existing manifest
    create            servicetype [appid] # create a new service [inside a specific application]
    start             serviceid [cloud]  # startup the given service [on a specific cloud]
    info              serviceid          # get service details
    logs              serviceid          # get service logs
    stop              serviceid          # stop the specified service
    terminate         serviceid          # delete the specified service
    rename            serviceid newname  # rename the specified service
    startup_script    serviceid filename # upload a startup script
    usage             serviceid          # show service-specific options
ema@orion:~/dev/conpaas$
```

CLI client

Base class for service-independent code.

Specific classes for each service to implement service-dependent methods.

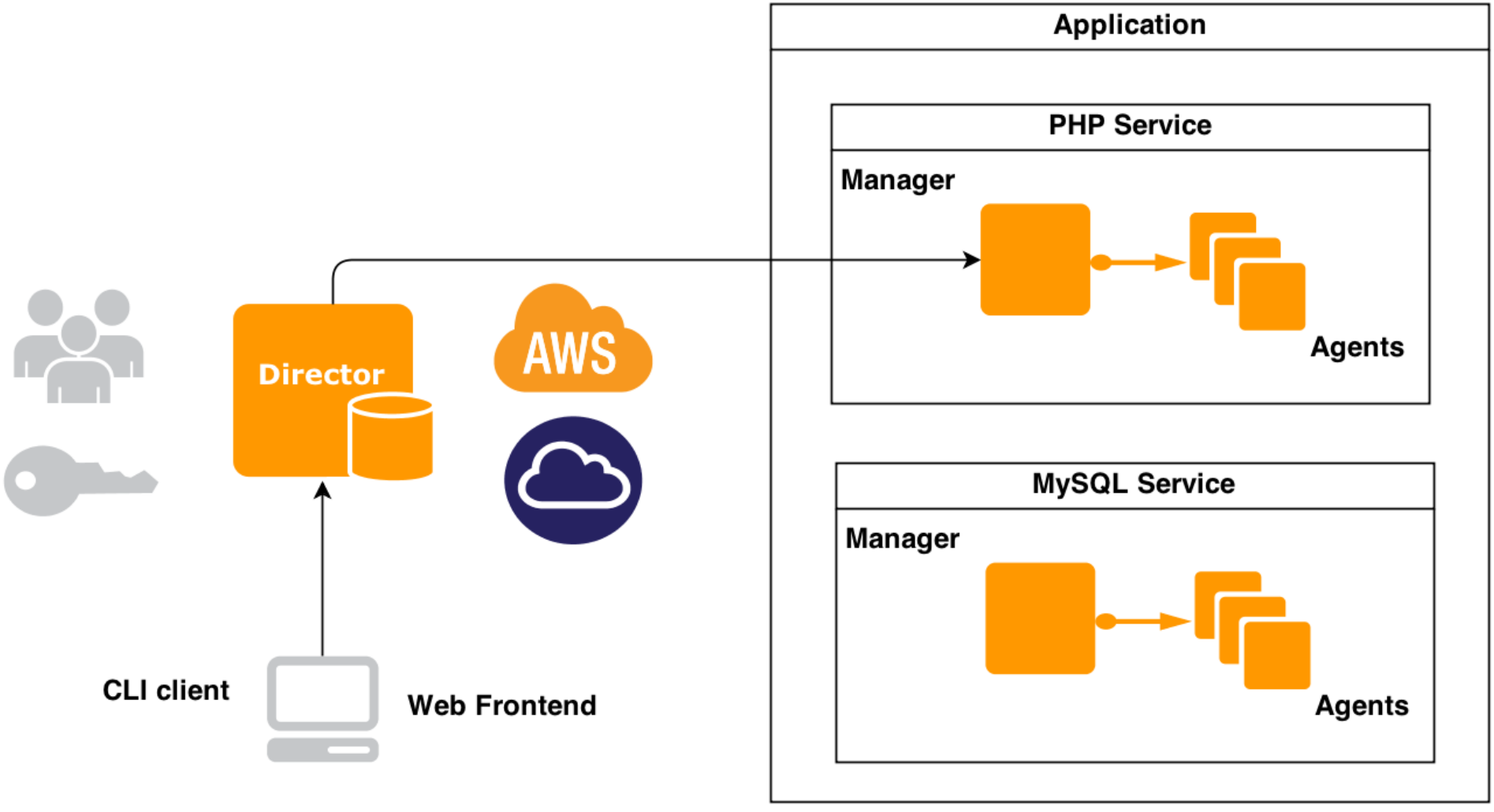
CLI client

```
ema@orion:~$ cpsclient.py create mysql
Creating new manager on 192.168.122.26...  done.
ema@orion:~$
ema@orion:~$
ema@orion:~$
ema@orion:~$
ema@orion:~$ cpsclient.py list
type  sid application_id vmid name              manager
-------------------------------------------------------------
mysql   1              1 1437 New mysql service 192.168.122.26
ema@orion:~$
ema@orion:~$
ema@orion:~$
ema@orion:~$
ema@orion:~$ cpsclient.py help 1
Usage: /usr/local/bin/cpsclient.py COMMAND [params]
COMMAND is one of the following

    credentials                          # set your ConPaaS credentials
    listapp                              # list all applications
    available                            # list supported services
    clouds                               # list available clouds
    list              [appid]            # list running services under an application
    deleteapp         appid              # delete an application
    createapp         appname            # create a new application
    manifest          filename           # upload a new manifest
    download_manifest appid              # download an existing manifest
    create            servicetype [appid] # create a new service [inside a specific application]
    start             serviceid [cloud]  # startup the given service [on a specific cloud]
    info              serviceid          # get service details
    logs              serviceid          # get service logs
    stop              serviceid          # stop the specified service
    terminate         serviceid          # delete the specified service
    rename            serviceid newname  # rename the specified service
    startup_script    serviceid filename # upload a startup script
    usage             serviceid          # show service-specific options
    set_password      serviceid password
    add_nodes         serviceid count [cloud]
    remove_nodes      serviceid count
ema@orion:~$
```

Frontend

- PHP web application
- HTTPS/JSON to talk with the Director
- GUI of ConPaaS


Application
PHP Service
Manager
Agents
MySQL Service
Manager
Agents
Director
AWS
CLI client
Web Frontend

People behind ConPaaS Core

Ismail El Helw

Services, Managers, Agents

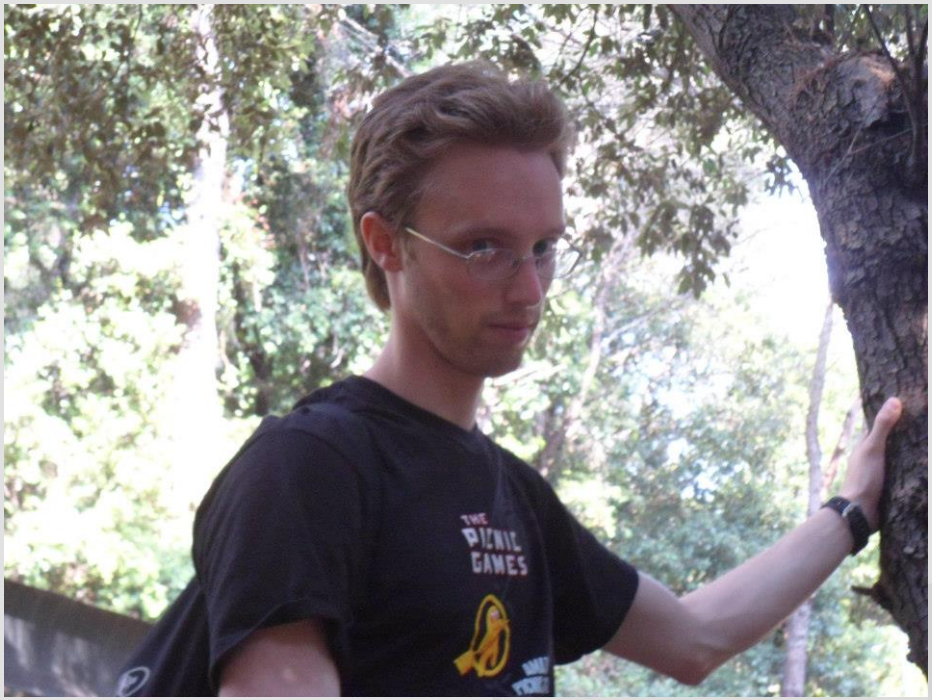
People behind ConPaaS Core

Adriana Szekeres

Security

People behind ConPaaS Core

Francesco Allertsen



Applications, Manifest

People behind ConPaaS Core

Emanuele Rocca

Director, IPOP integration

People behind ConPaaS

Claudiu Gheorghe

Frontend

Conclusions

1. **Services** to support different programming models
2. **Core** for service-independent parts
3. **Applications** to put Services together
4. **Director** to put it all together
5. **CLI client** for developers and scripts
6. **Frontend** for human beings