ConPaaS: an integrated runtime environment for elastic cloud applications

Guillaume Pierre

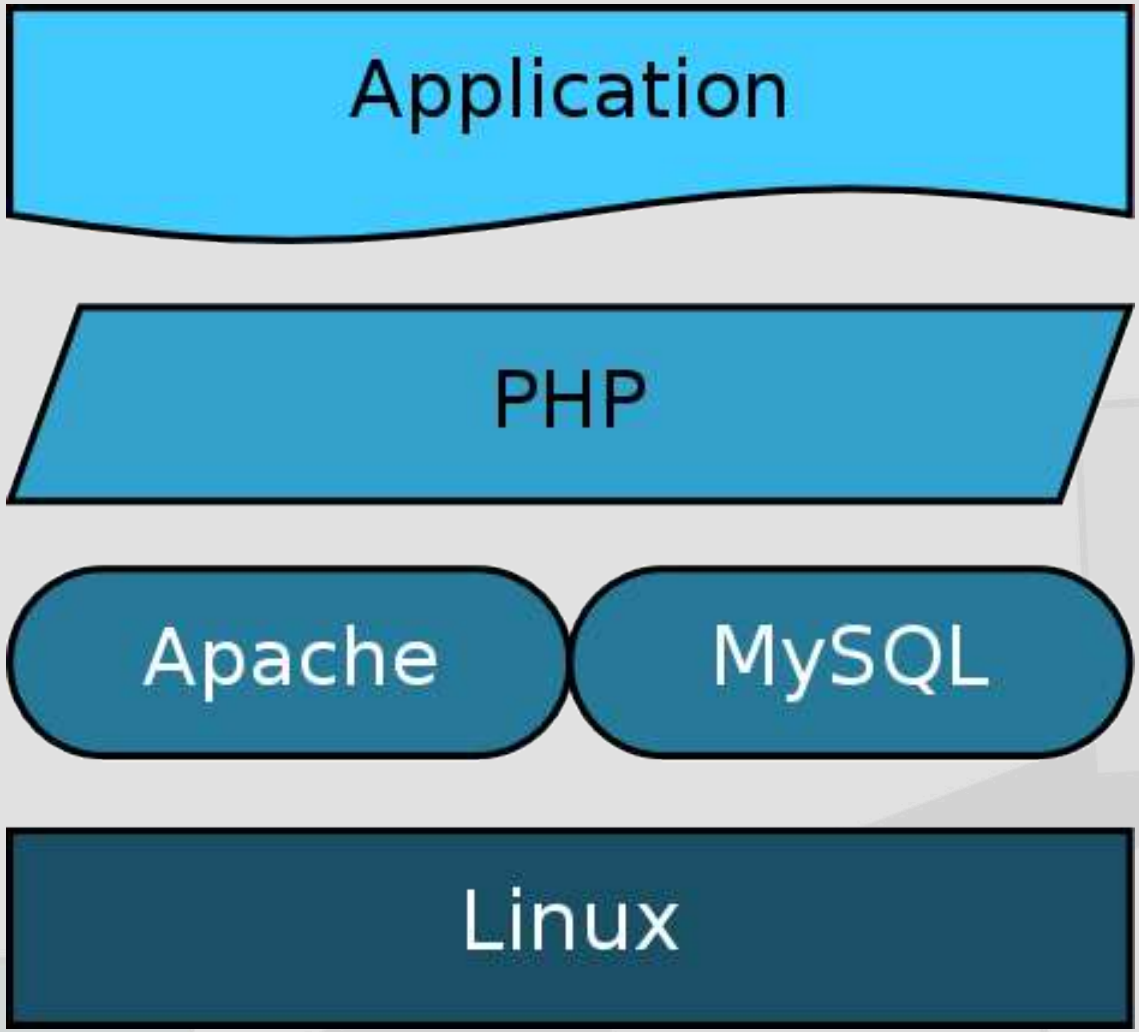
www.conpaas.eu

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

Deploying an application in the Cloud can be easy

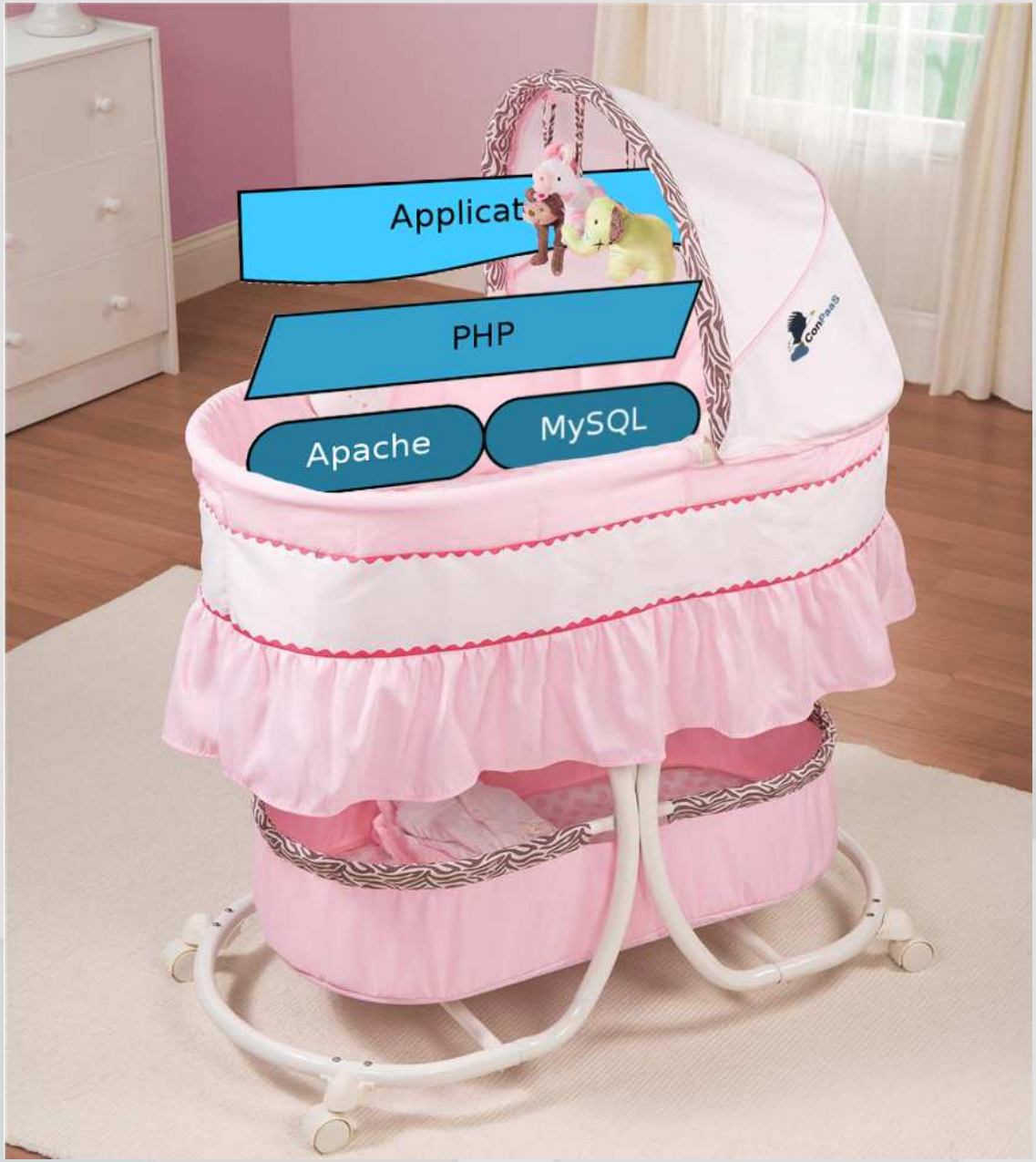
1. Choose a cloud provider
2. Start one virtual machine using a ready-made image
3. Install your software in the virtual machine
4. Snapshot the VM

What about a slightly more complex application?

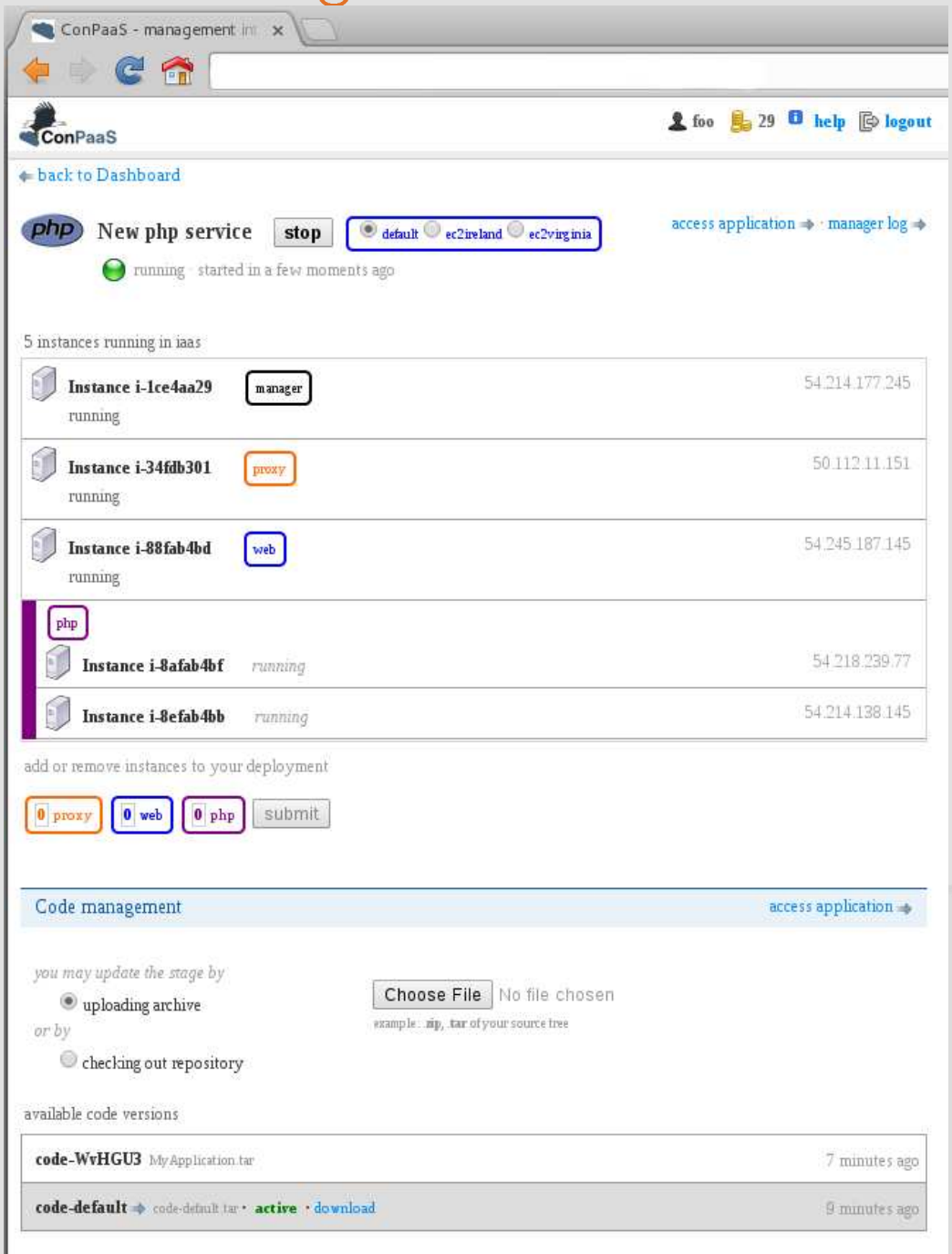


What about a slightly more complex application?

ConPaaS takes care of your applications

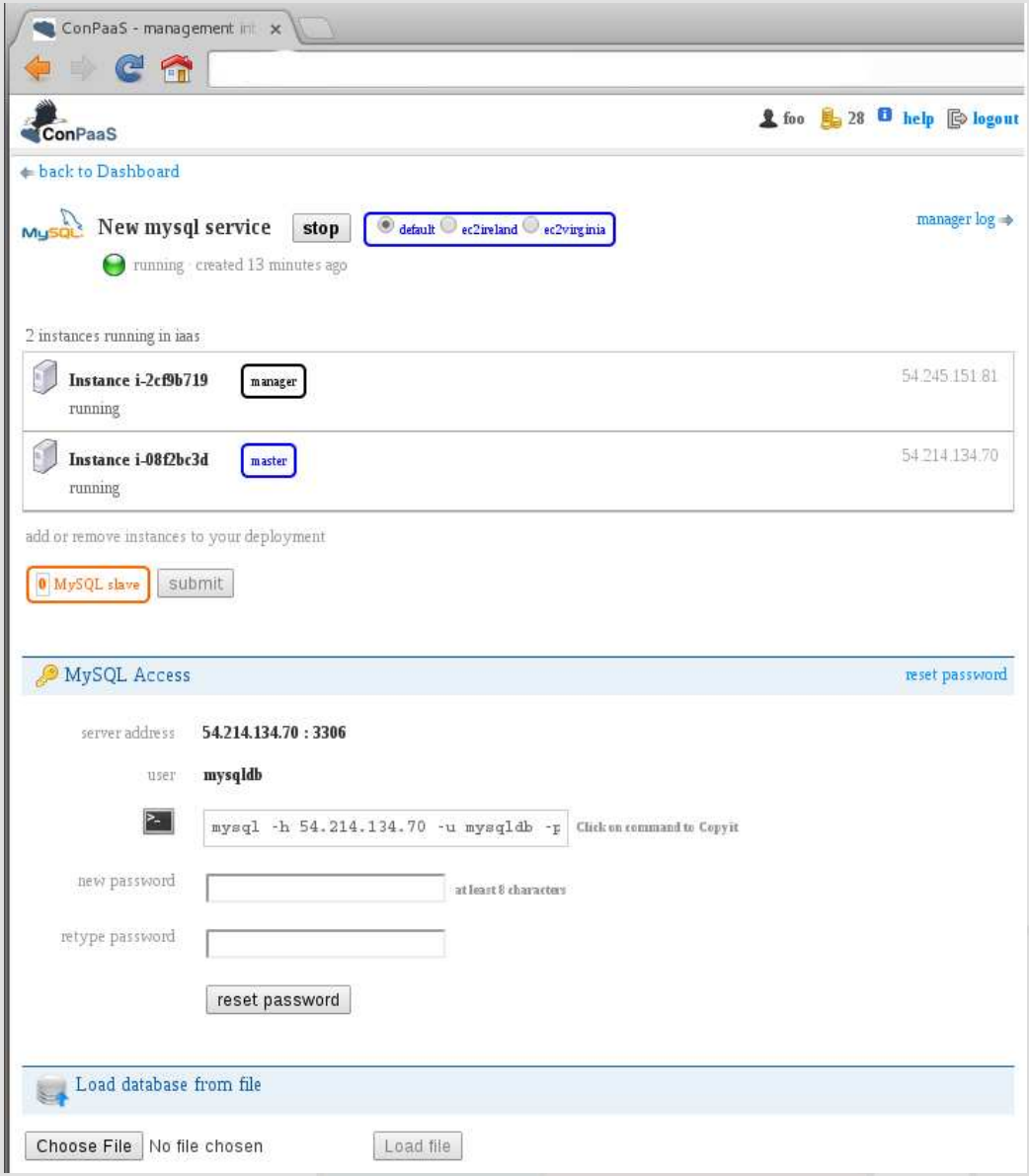


Web hosting services



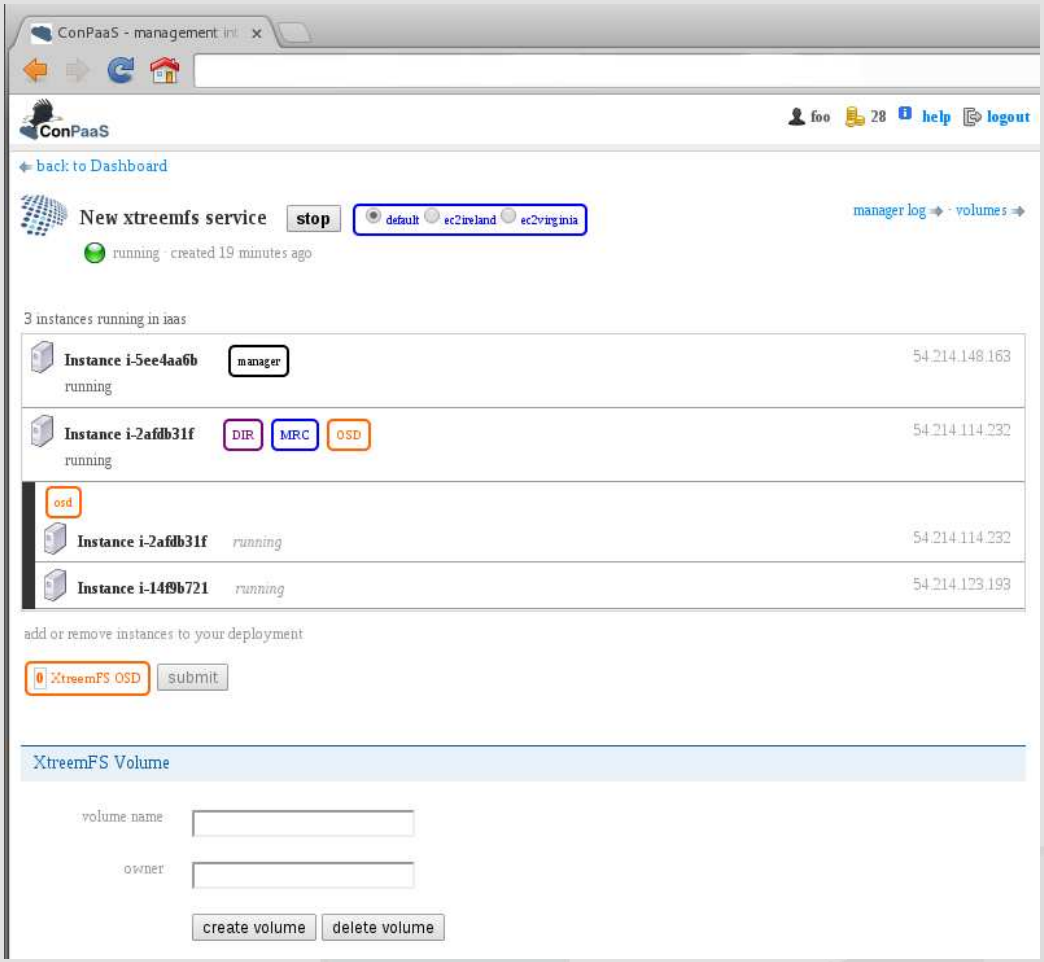
- Code upload (PHP, Java)
  - ZIP file upload, git push
  - Choose the version you want to run
- Seamless capacity control
  - Add/remove processing power in one click
- Coordinated reconfigurations
  - No service interruption even during reconfigurations

Database services



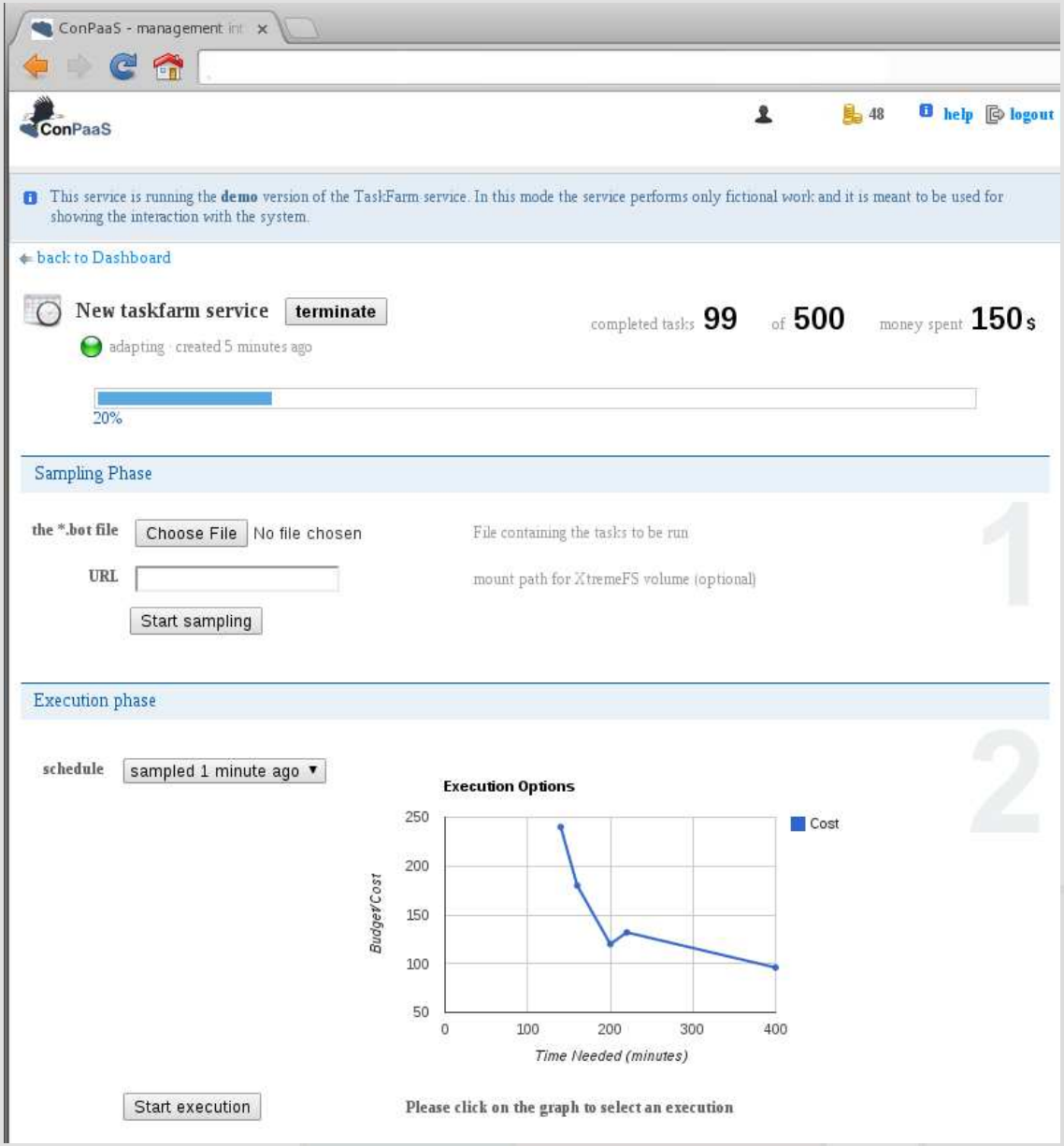
- Relational (MySQL) and NoSQL (Scalarix)
- Data upload
  - Choose your administrator password
  - Upload a database dump
- Automatic replication
  - Add/remove replicas in one click

File system service



- Mount the file system
  - From your client machine
  - From other ConPaaS services
- Read-write data as in a local file system
  - Full POSIX support
- Automated capacity scaling
  - Add/remove storage nodes in one click

High-performance computing services

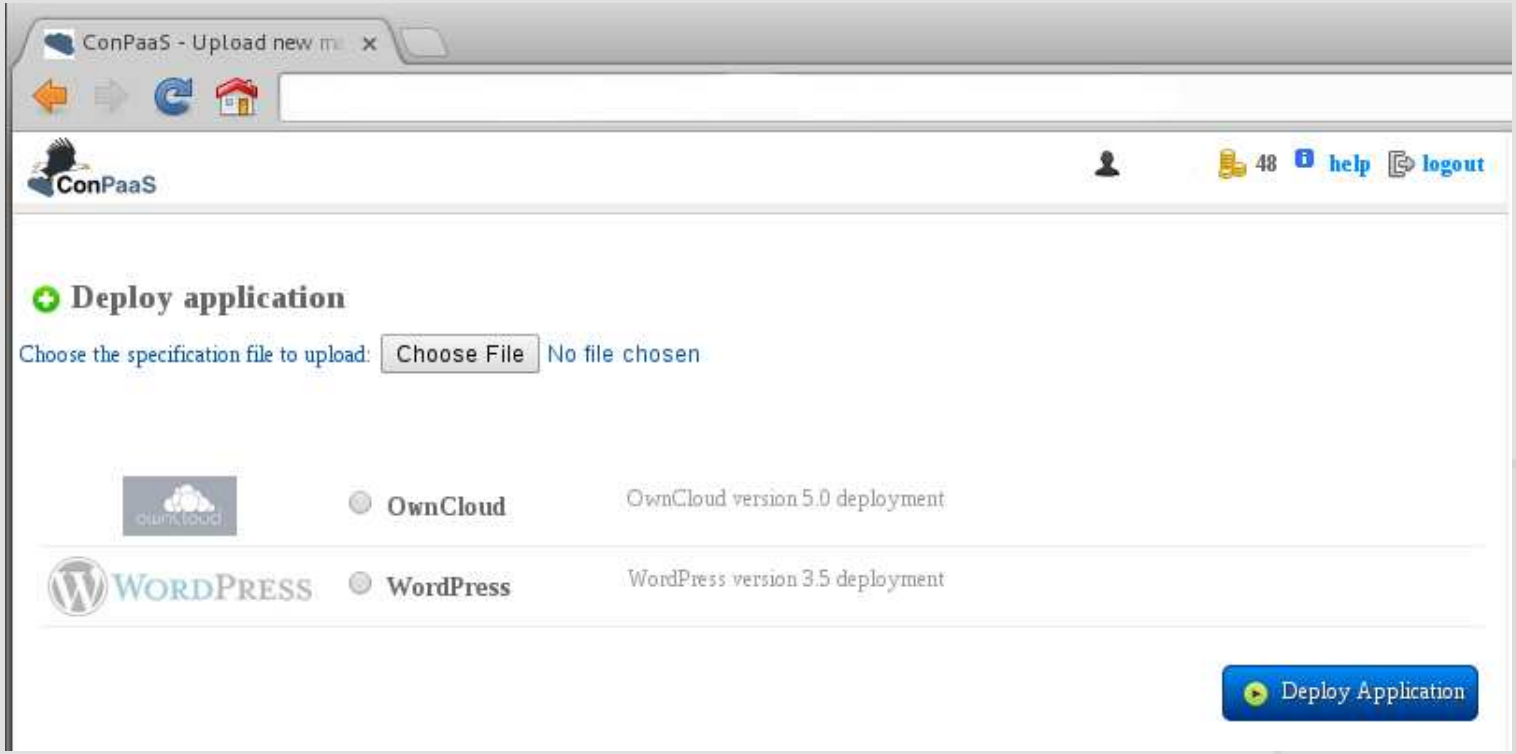


- TaskFarming for scientific workflows and massive batch executions
  - Control the tradeoff between cost and execution time
- MapReduce for BigData processing
  - Based on Hadoop

Multi-cloud deployments

- ConPaaS can use multiple underlying clouds simultaneously
  - For example: OpenNebula in my private cluster, EC2 in Virginia, and EC2 in Oregon
- Each virtual machine can be started in any cloud
- All virtual machines are connected using a multi-point VPN
  - Firewall traversal

Automated application deployment



- ConPaaS can deploy entire applications in one click
- A manifest file specifies all relevant information about the application
  - List of services, code/data to upload, configuration
- Users can use ready-made manifests or upload their own

Release Timeline

Oct 2010: Beginning of the project

Apr 2012: ConPaaS-0.9.0

- Five services: PHP, Java, MySQL, Scalarix, MapReduce
- Support of Amazon EC2 and OpenNebula clouds

Oct 2012: ConPaaS-1.0.0

- TaskFarming and Selenium services
- Secure control communications
- GIT-based code uploads

Feb 2013: ConPaaS-1.1.0

- XtreemFS service
- Full control via command-line tools

Jun 2013: ConPaaS-1.2.0

- Multi-cloud support
- VPN for internal application communications
- Automated deployment of entire applications
- Performance monitoring

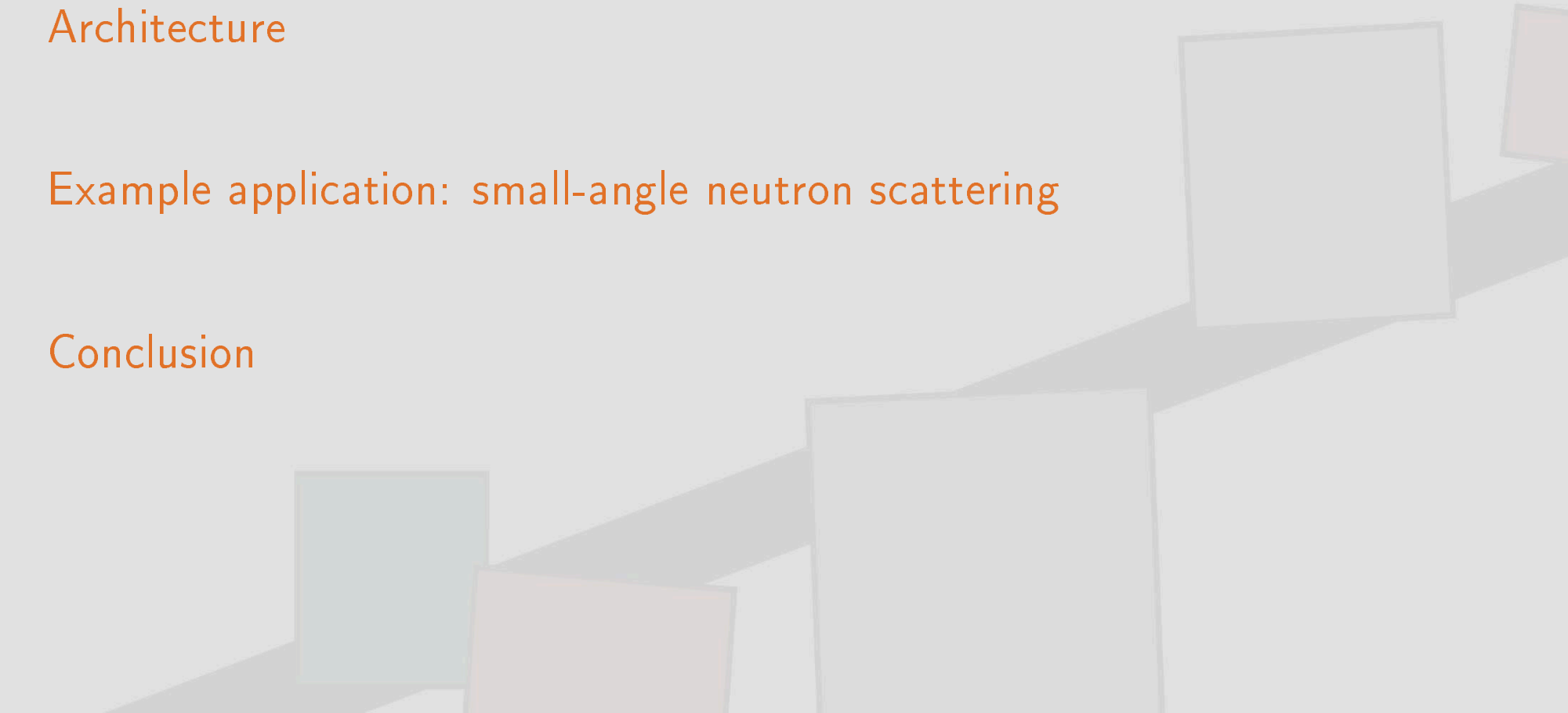
Agenda

Introduction

Architecture

Example application: small-angle neutron scattering

Conclusion



Agenda

Introduction

Architecture

Example application: small-angle neutron scattering

Conclusion

ConPaaS architecture

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

Services

ConPaaS Services

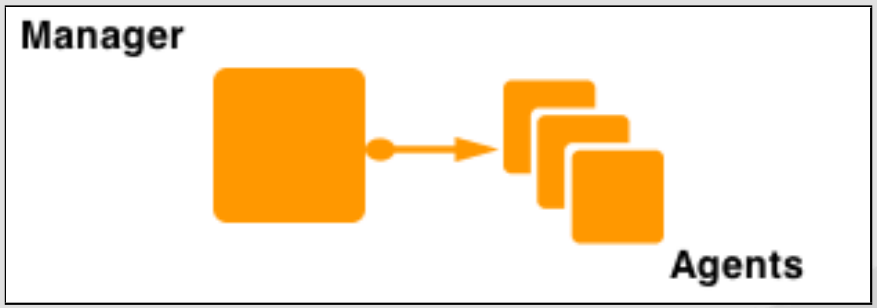
To support a diverse set of use cases we have introduced the concept of **Services**.

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

ConPaaS Services

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

The Manager is responsible for adding and removing agents, (re-)configuring them, as well as for other administrative tasks.



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.

- A manager class
- An agent class

ConPaaS Services

Different ConPaaS services actually have a lot in common.

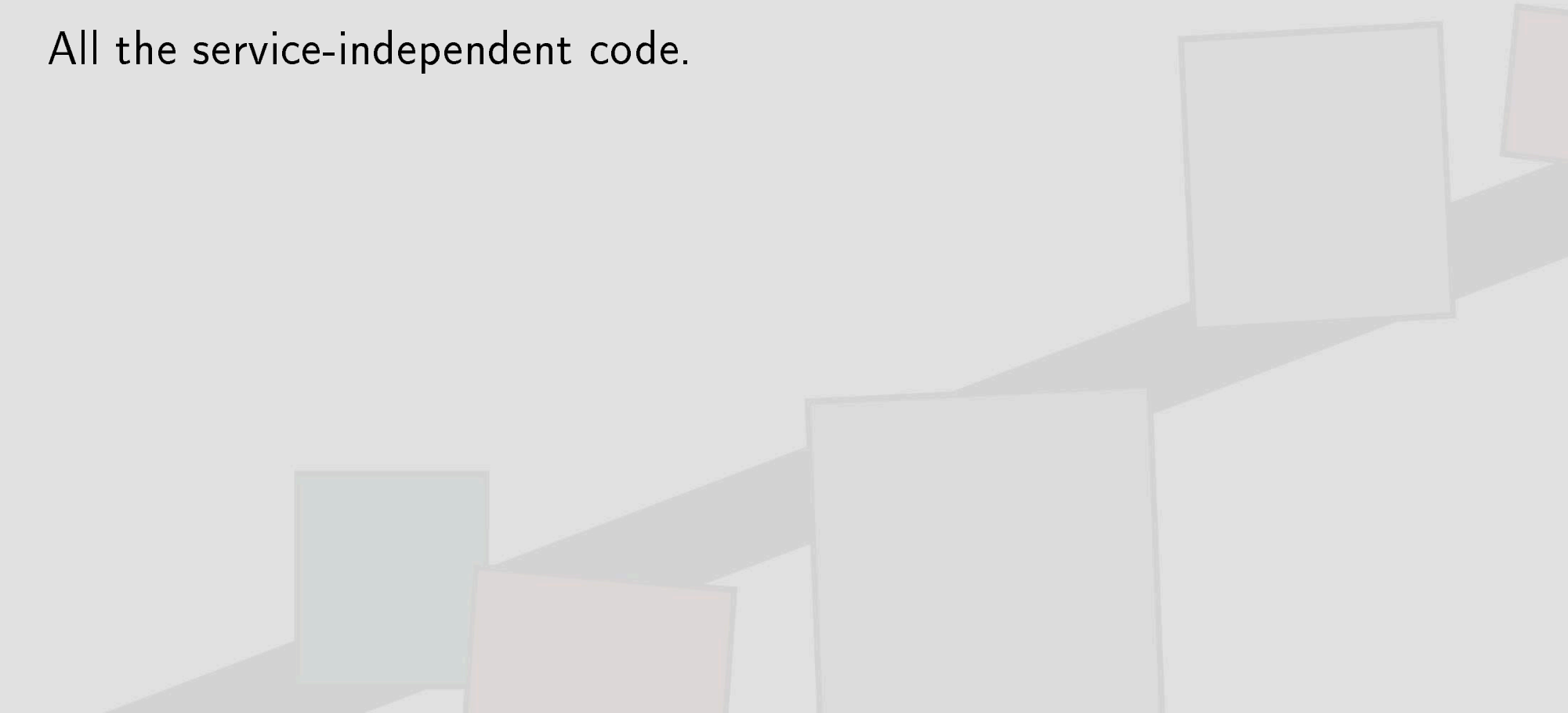
Manager and Agent classes do not have to be written completely from scratch.

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

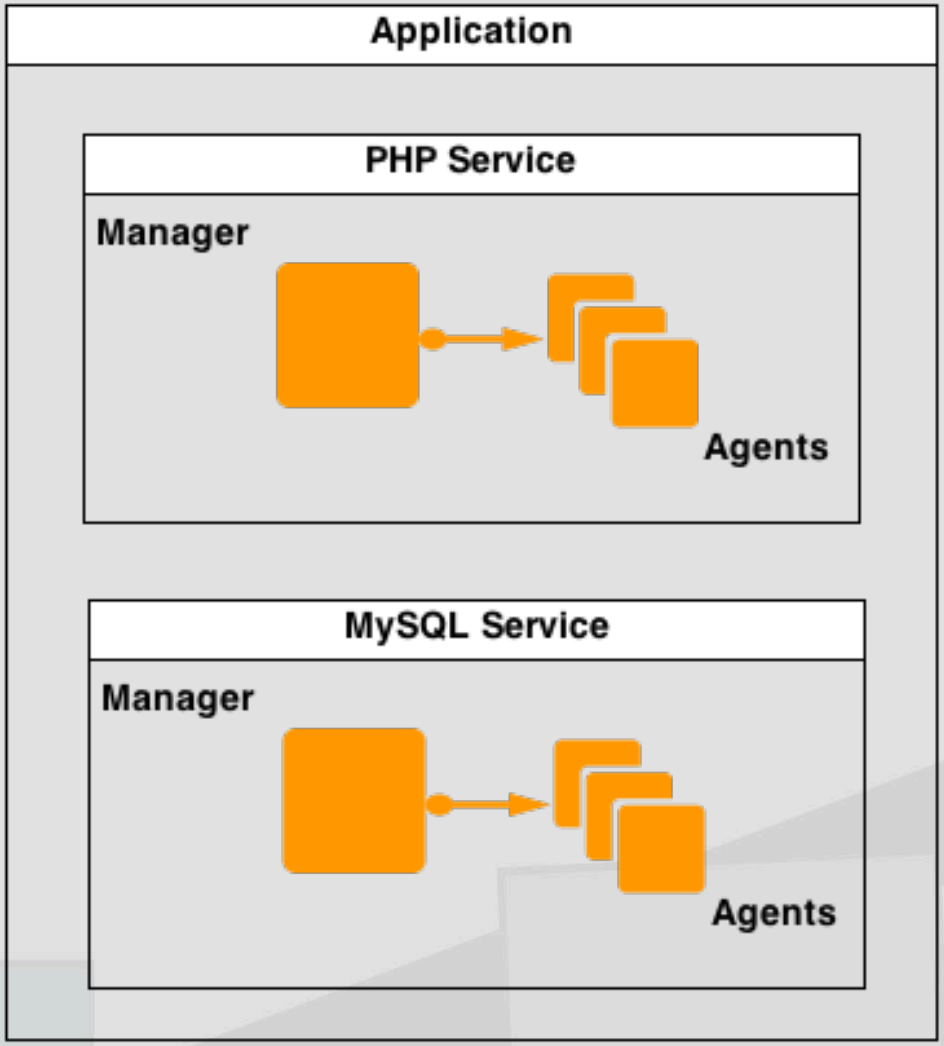
Applications

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

Users want to run WordPress.

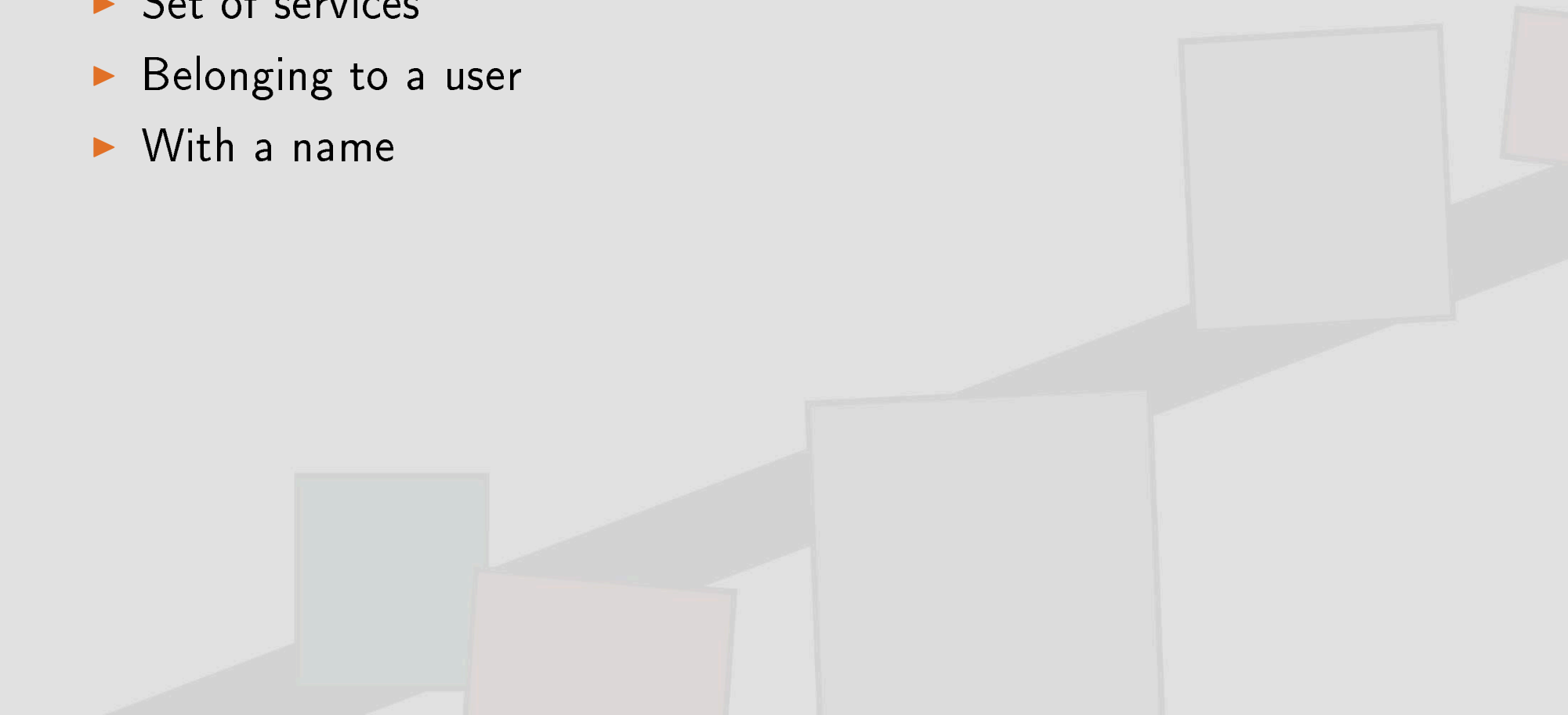
We need to put services together.

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

Handles the life-cycle of ConPaaS applications.

Director

Director

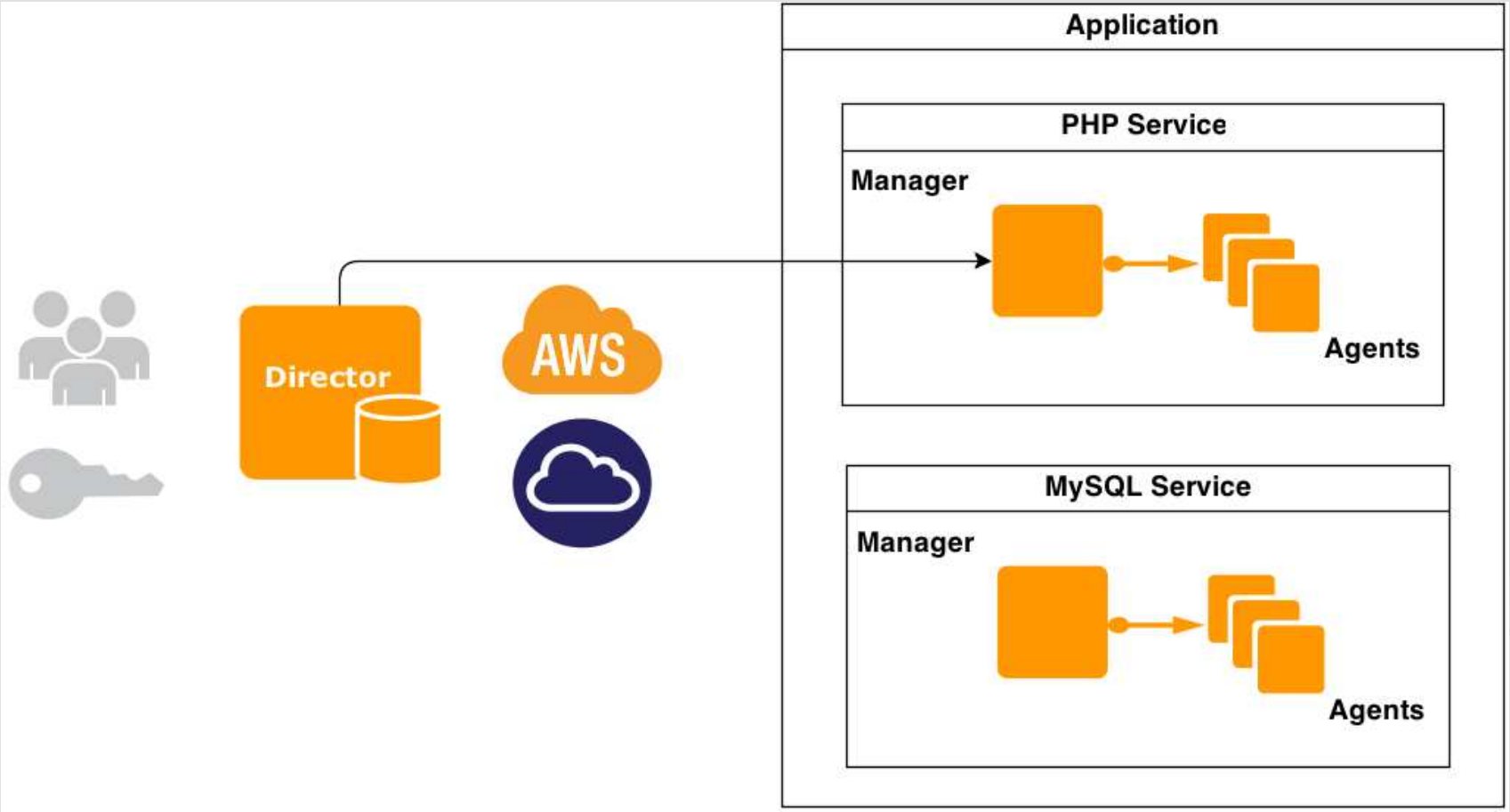
- Python/Flask application
- Deployed to Apache
- HTTPS/JSON
- ConPaaS API server

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.

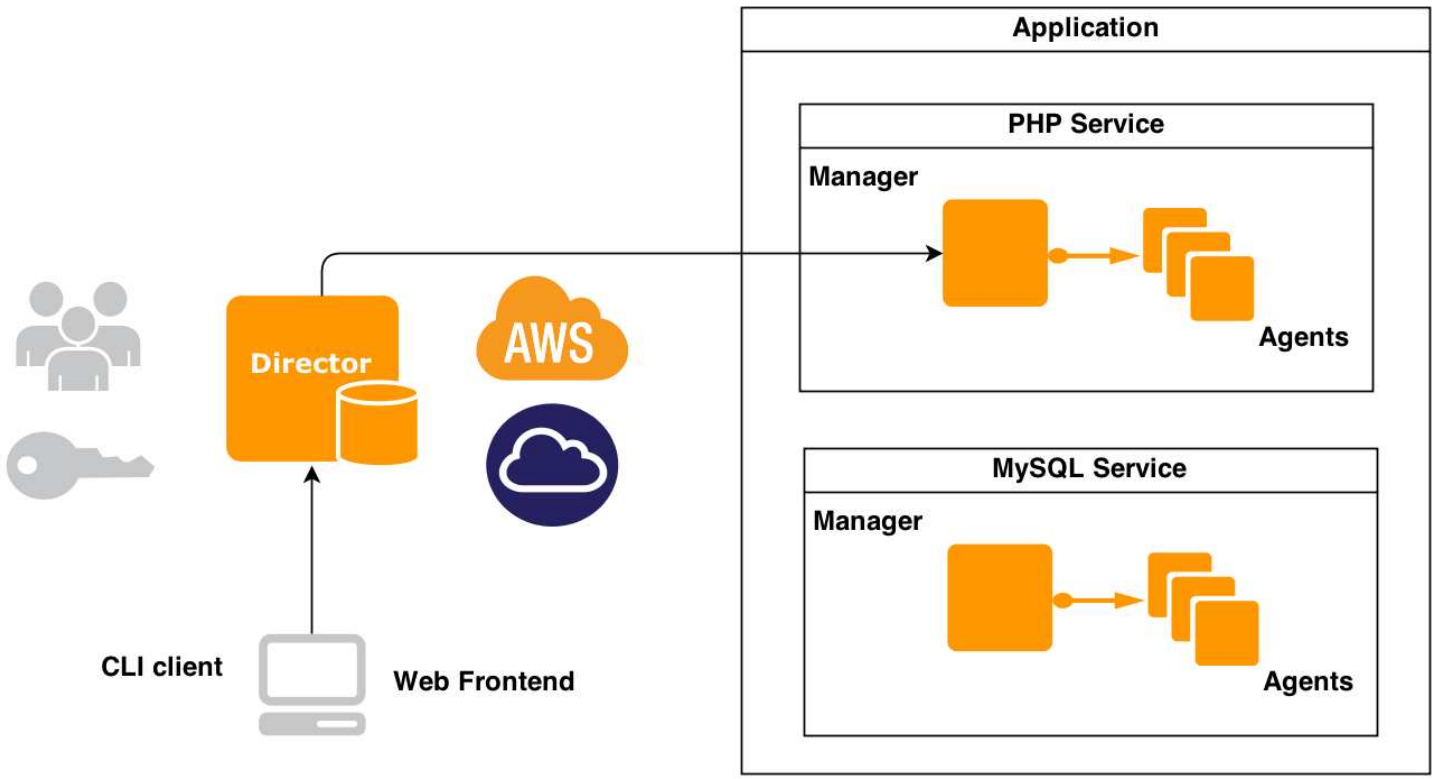
CLI client

```
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

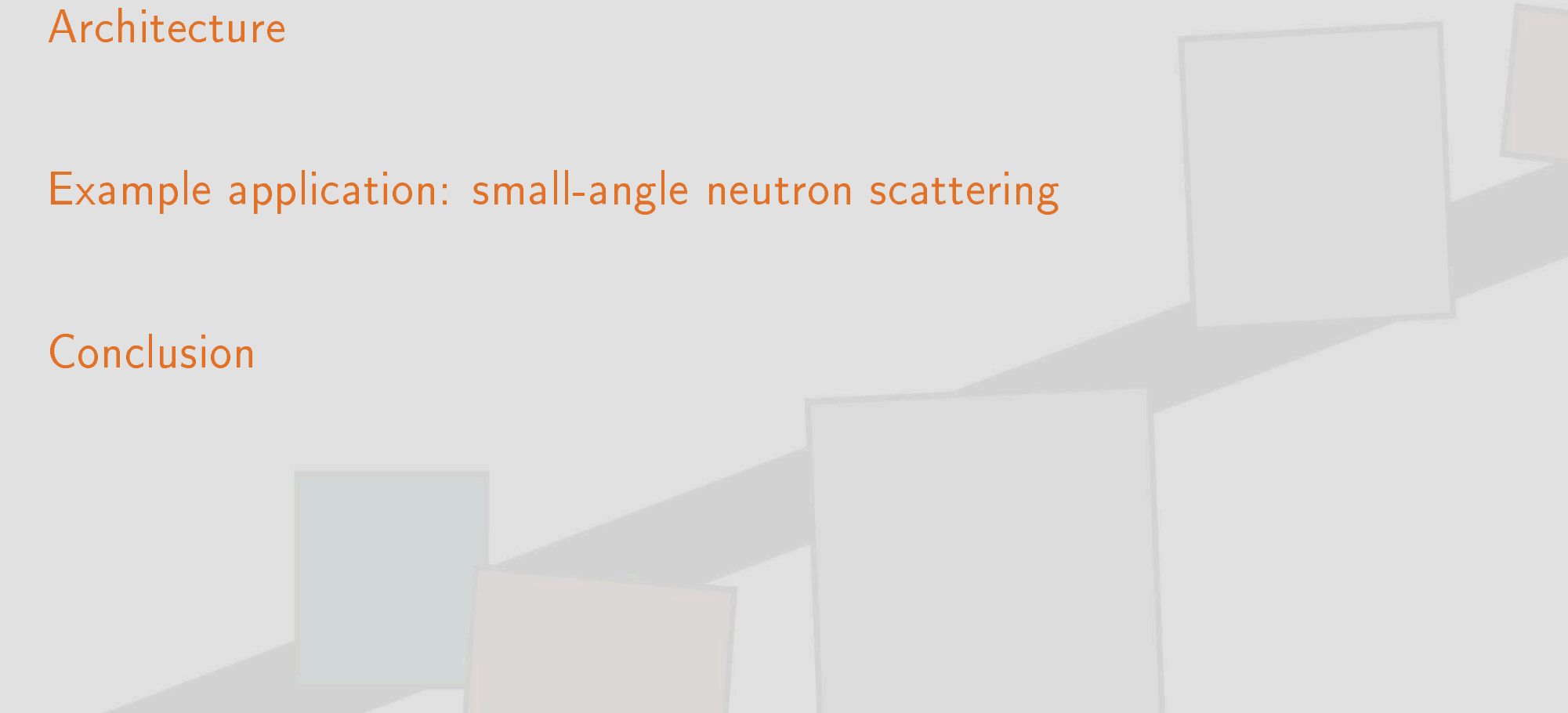
Agenda

Introduction

Architecture

Example application: small-angle neutron scattering

Conclusion



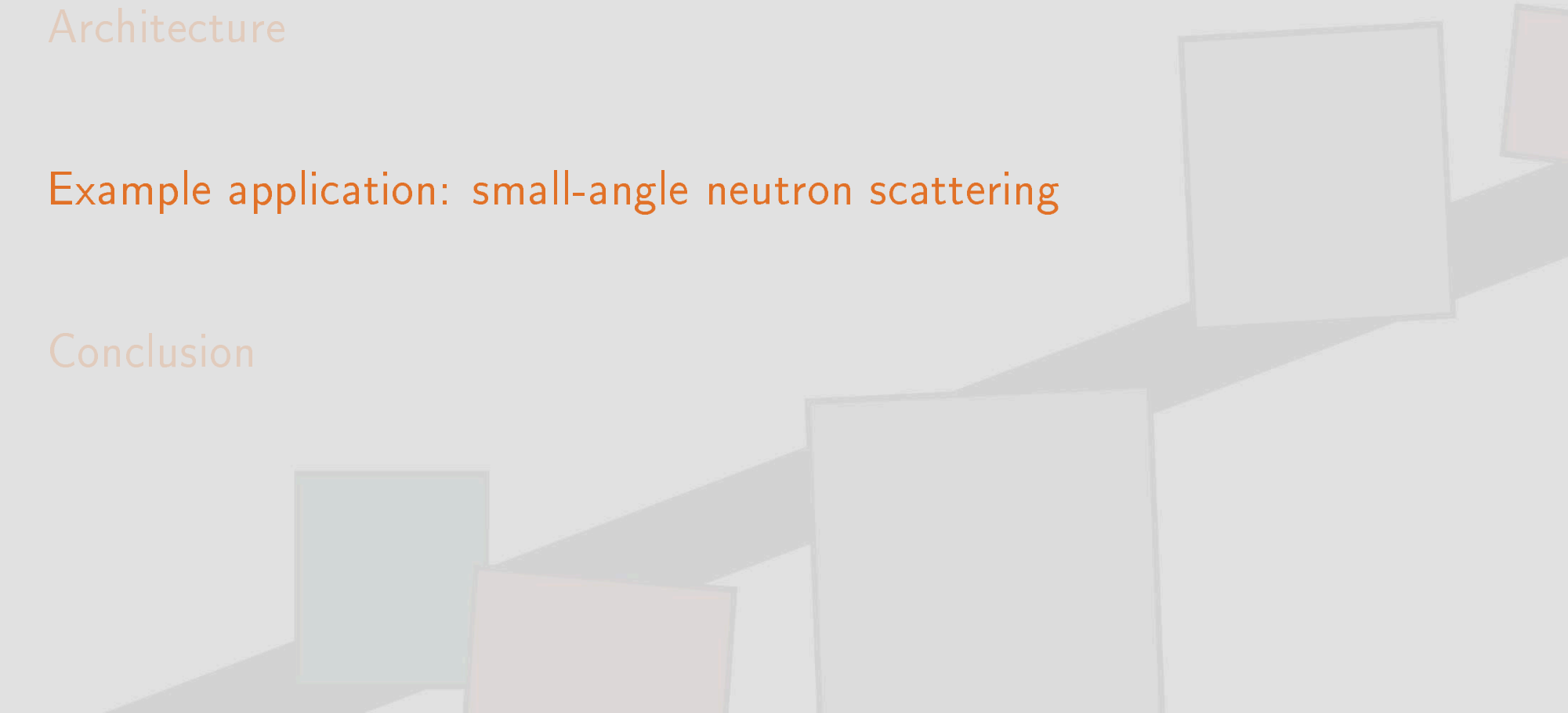
Agenda

Introduction

Architecture

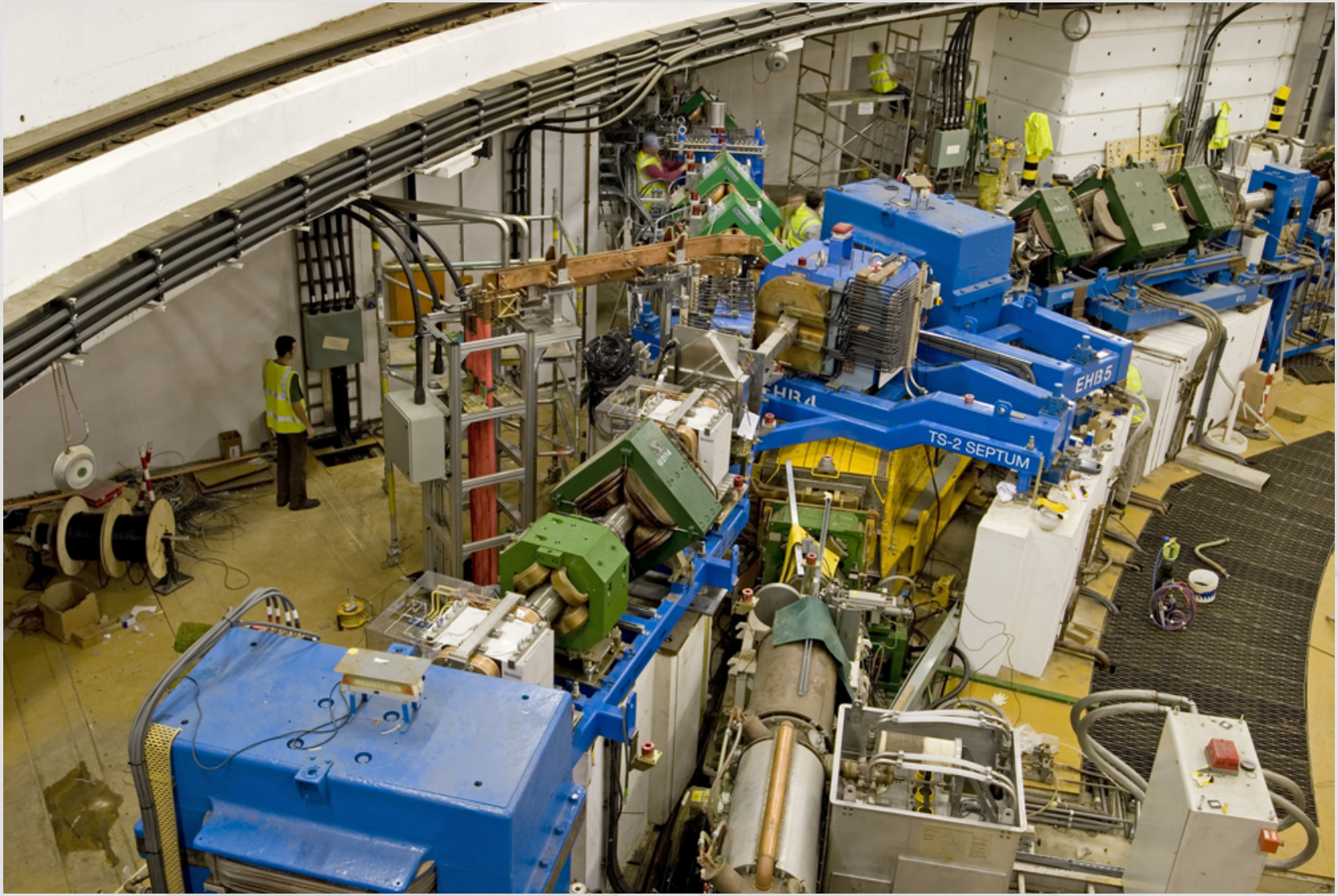
Example application: small-angle neutron scattering

Conclusion



Data comes from this...

Small-angle neutron scattering



What for...

- To determine the molecular structure of biological samples
- Common method to study proteins in biology
- To make better soaps!

How does it work?

- Need to find models that fit SANS data
- Does that molecule look like a sphere? or a cylinder?
- If so, what diameter fits better? or what height?
- Long story short, it is parameter sweep problem.
- `https://pypi.python.org/pypi/pybiosas`

The parameter sweep

- Parameter sweeps are sweet!
- Tasks with no dependency...
- Depending on the SANS problem: 1,000 to 100,000 tasks
- On modest hardware, 1,000 tasks take <= an hour.

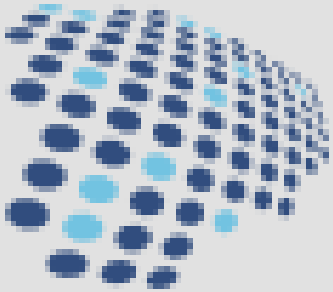
The scientist

1. Goes to that big machine and does his experiment
2. Looks at the data
3. Defines the range for parameters
4. Runs the sweep - wait
5. Looks at the result
6. Good? Done. Not good? Go to 3.

Scientist requirements

- Execution environment
- Short sweeps:
  - Interactiveness is important
  - Continuous stream of jobs
- Long sweeps:
  - 100 hours = 6$ on EC2 m1.small
  - Need to optimize between time and cost
- Storage:
  - Non-persistent: Sweeps
  - Persistent: SANS data and sweeps with good results

ConPaaS to the rescue



- Taskfarming service for the parameter sweep
- XtreemFS service for scalable and reliable storage
- Unified and clean interface for:
  - Starting/stopping services
  - Scaling services
  - Getting status of various services

Taskfarming service in ConPaaS

- Bag-of-tasks scheduler on VM nodes
- Parameter sweeps are bags of tasks
- Sampling/execution phase
- Optimizes between time and cost[1]

[1] Oprescu et al. Budget Estimation and Control for Bag-of-Tasks Scheduling in Clouds. Parallel Processing Letters, 2011.

XtreemFS service in ConPaaS

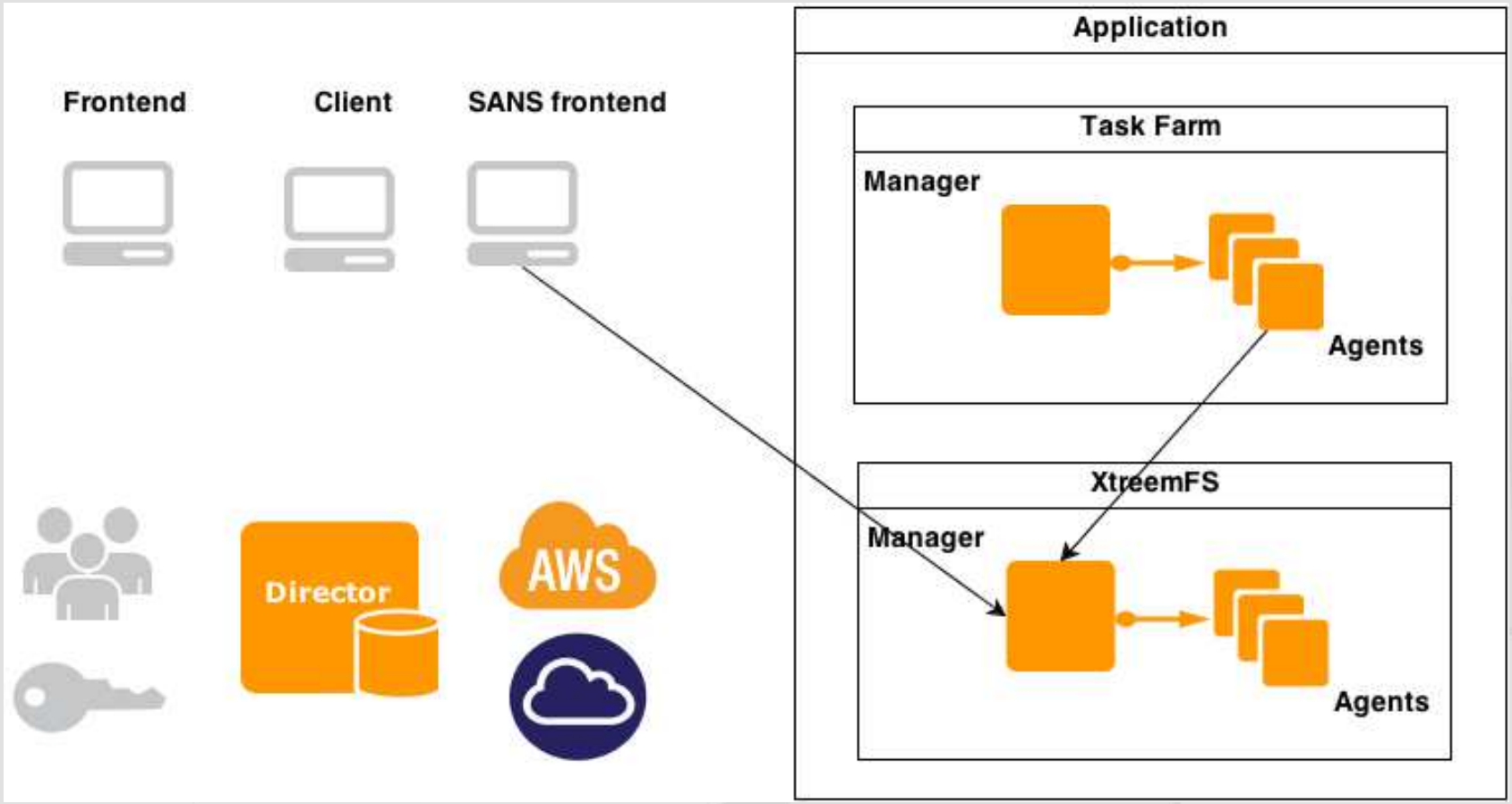
- Distributed object-based filesystem[2]
- Fully-integrated in ConPaaS
- Can add storage nodes on the fly
- Currently, only non-persistent storage on VM image disk
- Work in progress:
  - Support for scratch disks (more disk space)
  - Support for persistent storage (e.g. over Amazon EBS)

[2] Hupfeld et al. XtreemFS - a case for object-based storage in Grid data management. VLDB, 2007.

The ConPaaS director interface

- A unified interface to run commands on:
  - ConPaaS director: start/stop services
  - Services: scale, service status (e.g. tasks done)
- Users of this interface:
  - ConPaaS web frontend
  - ConPaaS client tools
  - And now: SANS modeling frontend!
- SANS modeling frontend
  - Specialized interface
  - Interactive workflow functionality described earliers
  - Functionality to analysis results of sweeps (e.g. clustering)
  - Lacks ConPaaS XtreemFS support (uses an external one)

Putting it all together



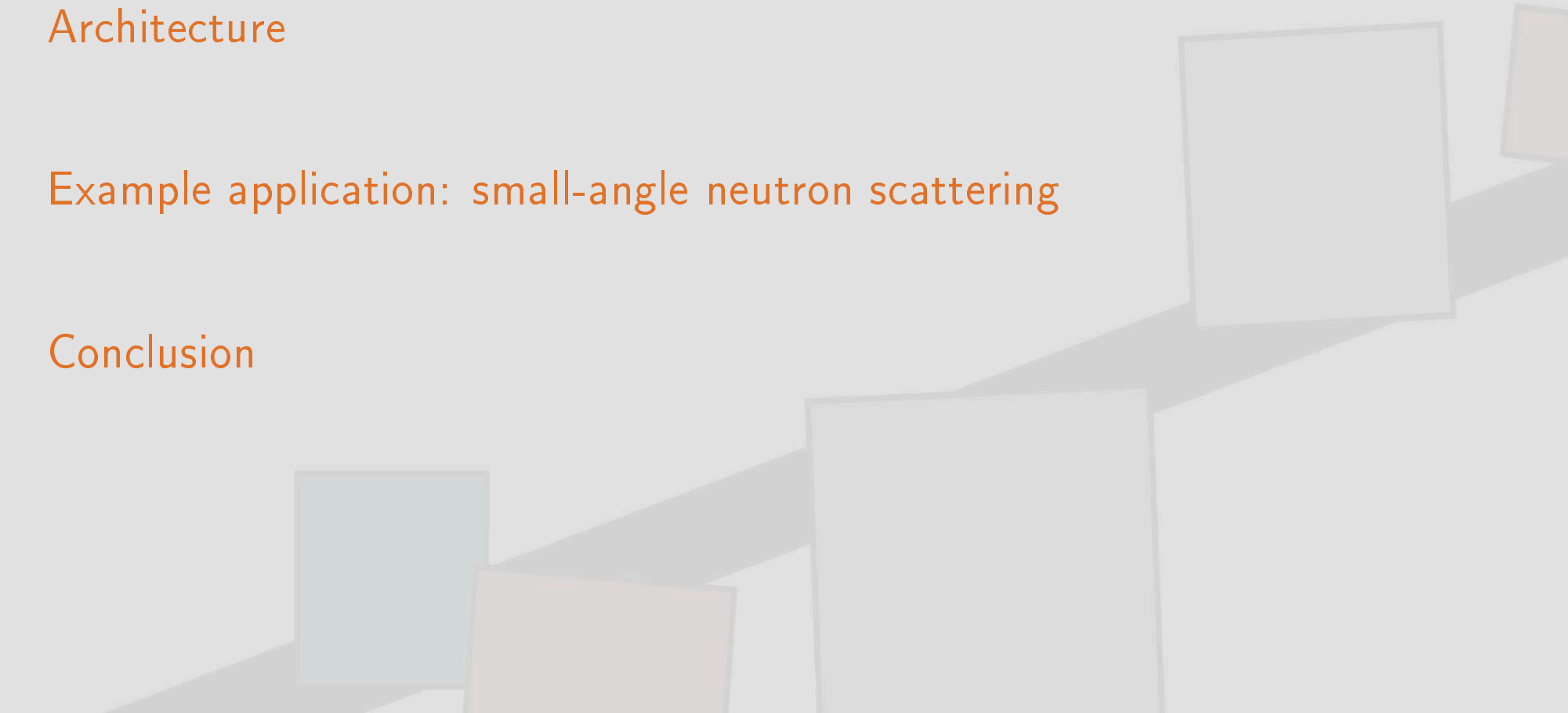
Agenda

Introduction

Architecture

Example application: small-angle neutron scattering

Conclusion



Agenda

Introduction

Architecture

Example application: small-angle neutron scattering

Conclusion

Conclusion

- An integrated runtime environment for elastic cloud applications
  - Standard services can be composed to build cloud applications
  - Extending ConPaaS with extra services is quite easy
- Application domains: Web apps, scientific apps
  - Soon: data stream processing
- Mature
- Open-source (BSD licence)

Free trial: `https://online.conpaas.eu`

People behind ConPaaS

Ismail El Helw

Services, Managers, Agents

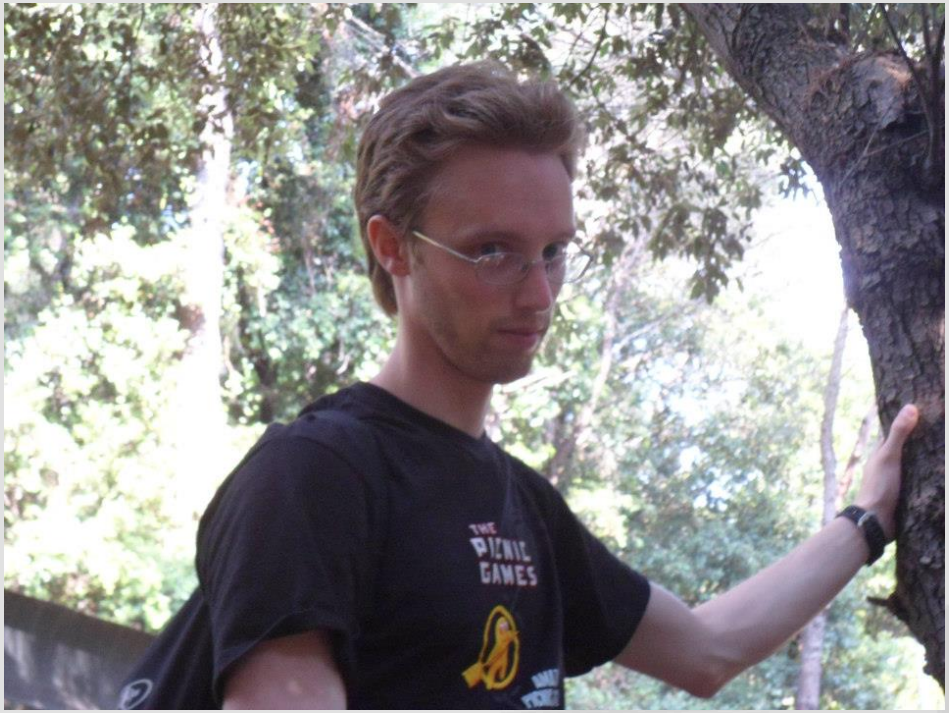
People behind ConPaaS

Adriana Szekeres

Security

People behind ConPaaS

Francesco Allertsen



Applications, Manifest

People behind ConPaaS

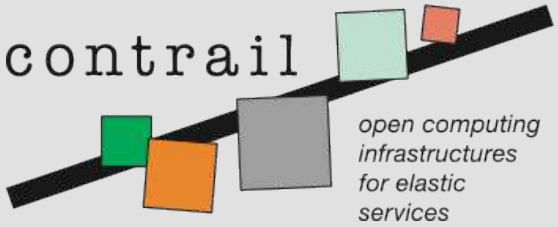
Emanuele Rocca

Director, IPOP integration

People behind ConPaaS

Claudiu Gheorghe

Frontend



contrail is co-funded by the
EC 7th Framework Programme

Funded under: FP7 (Seventh Framework Programme)
Area: Internet of Services, Software & Virtualization (ICT-2009.1.2)
Project reference: FP7-IST-257438
Total cost: 11.29 million euro
EU contribution: 8.3 million euro
Execution: From 2010-10-01 till 2013-09-30
Duration: 36 months
Contract type: Collaborative project (generic)