Managing the Virtual Machine Lifecycle of the CernVM Project

Ioannis Charalampidis <ioannis.charalampidis@cern.ch>, Artem Harutyunyan, Dag Toppe Larsen, Jakob Blomer, Predrag Buncic
CERN PH/SFT

Introduction

Development and maintenance of a Virtual Machine Appliance involves several steps, ranging from the maintenance of a linux distribution, to the automated quality assurance of the appliance, or its dynamic contextualization at the run time.

In this poster we present a new solution developed within the CERN Virtual Machine (CernVM) project that unites these steps within a coherent framework. The framework allows to manage (and automate) the procedures that are performed throughout the lifetime of a CernVM appliance from within a common interface.

The lifecycle

The CernVM lifecycle is a continuous process that consists of the following steps:

1. Process the feedback from the previous release
2. Make necessary changes to the recipe for building the appliance
3. Update the individual software packages in the repository.
4. Build the virtual appliance.
5. Test the appliance.
6. Release it.

The problem

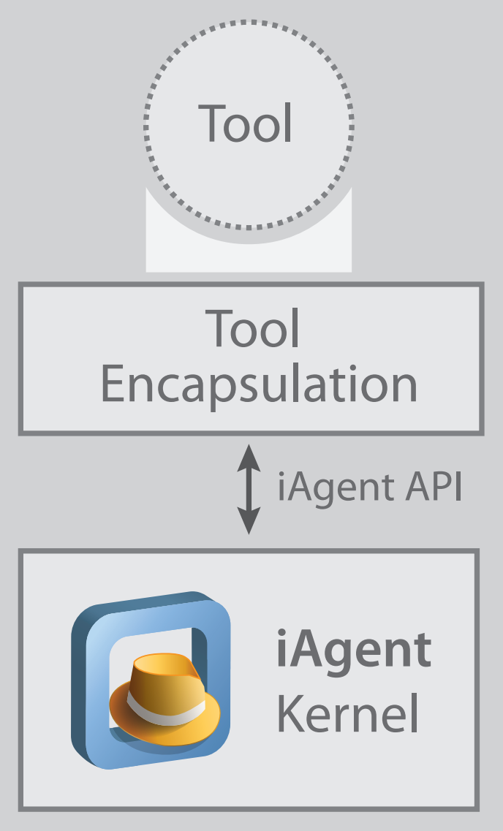
Some of these steps (such as building, or repository management), can be performed using existing tools. However for some others (such as testing) no generic tools existed.

Additionally, there is no apparent way to unify these steps within a single and coherent framework that would make it possible to automate the entire process.

The solution

We started by developing the missing software tools. Then we encapsulated each tool within a stand-alone component (built on top of the **iAgent** framework) that implements a common API, with well defined input and output.

The idea was to create a collection of components that can either be controlled independently, or could be chained together in order to repeatedly perform automated tasks.



An overview of the features and the framework components is shown in the diagram below:

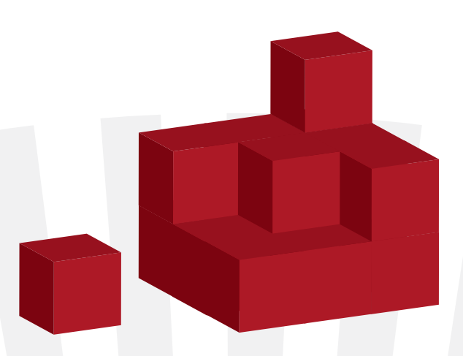
Topology agnostic
The iAgent instances are discovered dynamically. This means that new instances can be added without disrupting ongoing operations.

Job Agents

The iAgent instances comprise the server-side of the lifecycle management system. Each lifecycle task has one or more specialized agents associated with it.

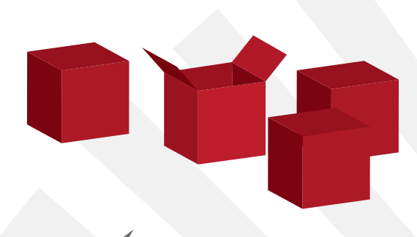
3. Build Agent

Builds a virtual machine disk image, according to the specified recipe and repository.



2. Repository Agent

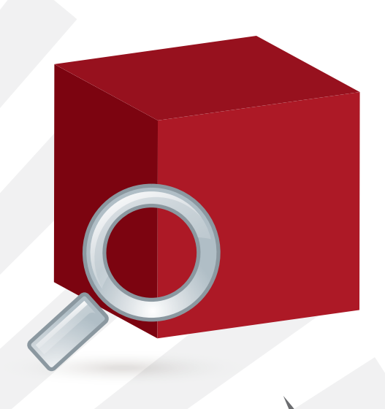
Updates the software repository, builds package groups, and triggers the build process.



Workflow subsystem
The agents execute the workflow by communicating with each other. Each node discovers the next node automatically. There is neither need for a central server nor for a predefined topology.

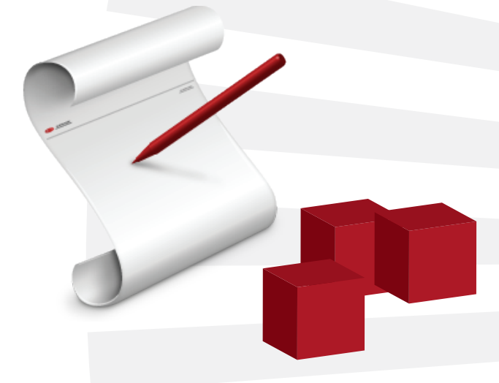
4. Test Agent

Instantiates and configures a virtual machine exactly like an end-user is expected to do, and runs an extensive test of functionality.



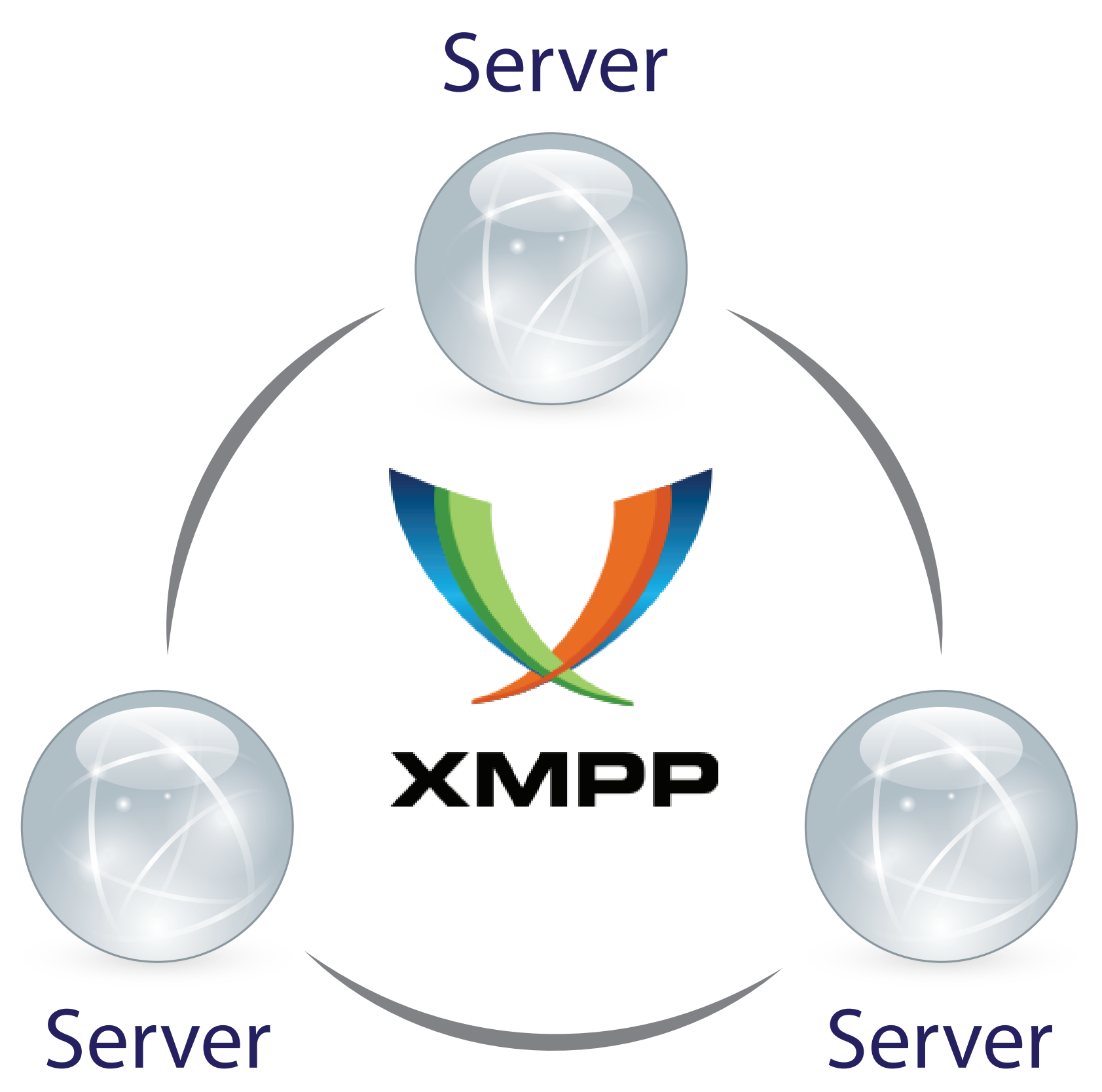
1. Configuration Agent

Holds the recipe for building the virtual machines and initiates the CernVM build cycle.



5. Release Agent

Publishes the virtual machine image



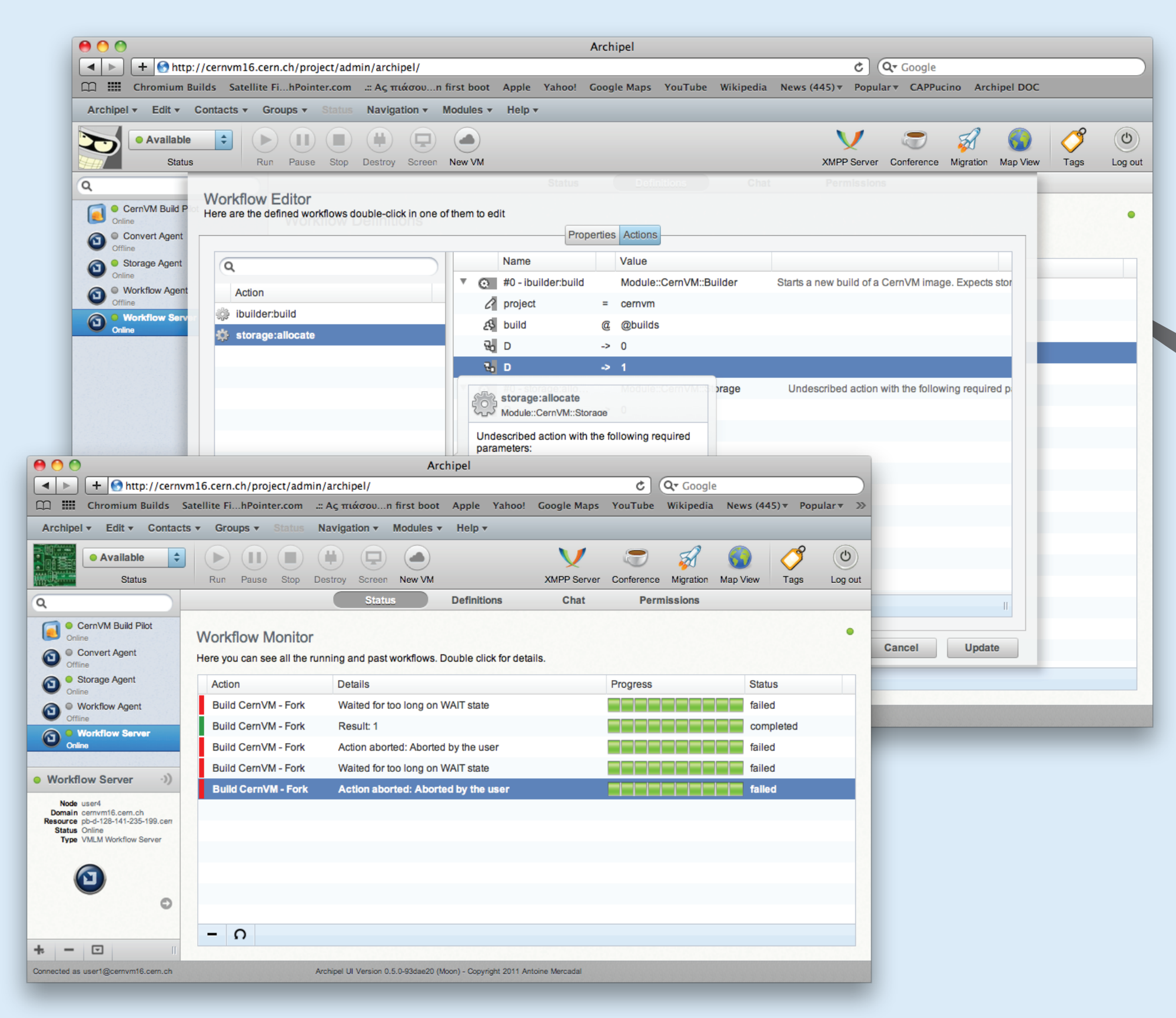
Built-in redundancy
The XMPP server provides load balancing and redundancy capabilities.

Management Clients

Since all the components are connected through XMPP they can all be managed by a specialized web client or even a simple chat terminal.

Archipel Client

We have created extensions to the stand-alone javascript interface of Archipel.

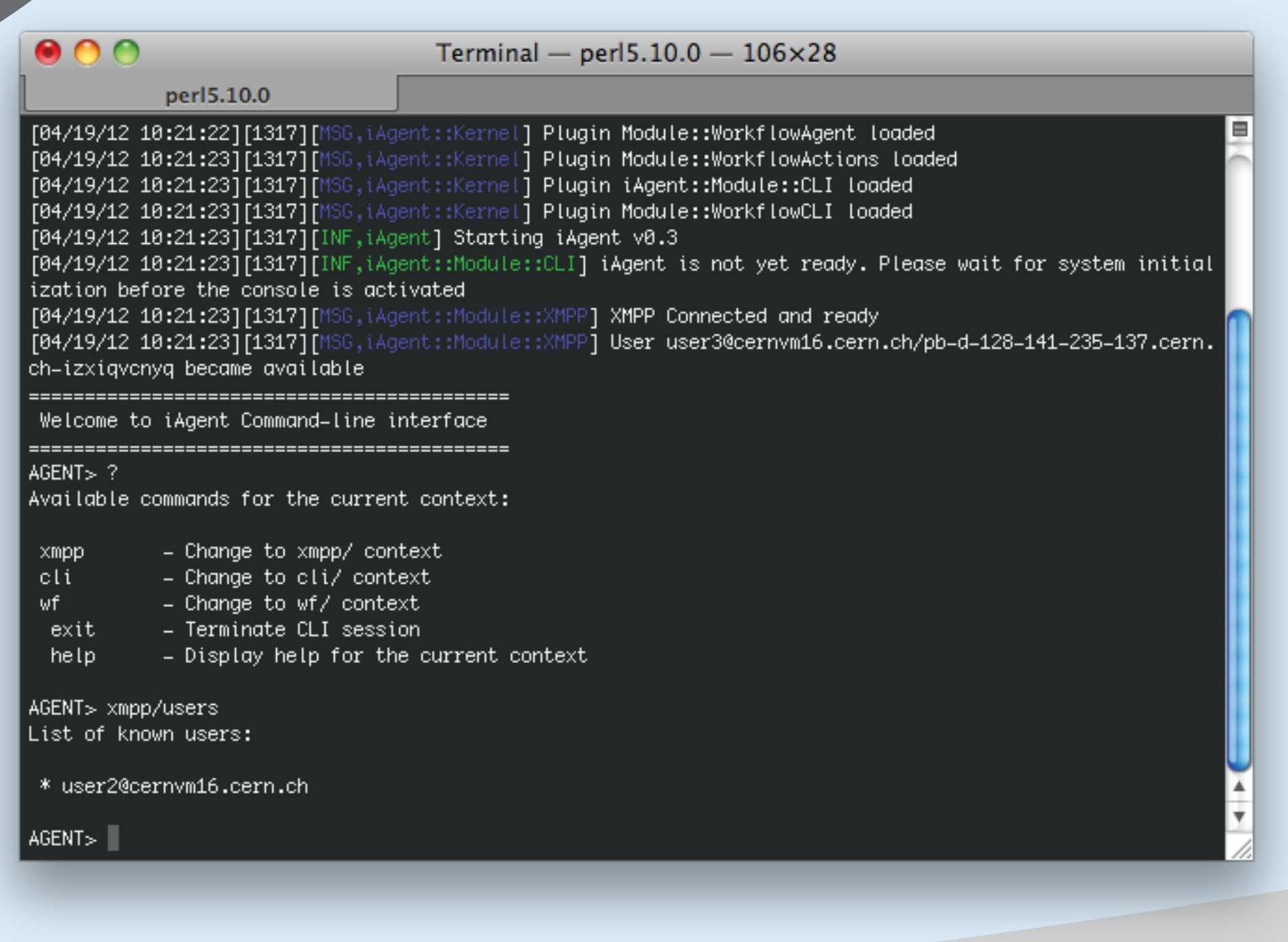


PicoClient

A portable client for mobile devices.

Command-Line Interface

The iAgent provides a built-in, interactive / batch CLI access to all of its local and distributed components.



```
Terminal — perl5.10.0 — 106×28
[04/19/12 10:21:22][1317][MSG,iAgent::Kernel] Plugin Module::WorkflowAgent loaded
[04/19/12 10:21:23][1317][MSG,iAgent::Kernel] Plugin Module::WorkflowActions loaded
[04/19/12 10:21:23][1317][MSG,iAgent::Kernel] Plugin iAgent::Module::CLI loaded
[04/19/12 10:21:23][1317][MSG,iAgent::Kernel] Plugin Module::WorkflowCLI loaded
[04/19/12 10:21:23][1317][INF,iAgent] Starting iAgent v0.3
[04/19/12 10:21:23][1317][INF,iAgent::Module::CLI] iAgent is not yet ready. Please wait for system initial
ization before the console is activated
[04/19/12 10:21:23][1317][MSG,iAgent::Module::XMPP] XMPP Connected and ready
[04/19/12 10:21:23][1317][MSG,iAgent::Module::XMPP] User user3@cernvm16.cern.ch/pb-d-128-141-235-137.cern.
ch-izxiqvcnyq became available
===========================================
 Welcome to iAgent Command-line interface
===========================================
AGENT> ?
Available commands for the current context:

 xmpp       - Change to xmpp/ context
 cli        - Change to cli/ context
 wf         - Change to wf/ context
  exit      - Terminate CLI session
  help      - Display help for the current context

AGENT> xmpp/users
List of known users:

 * user2@cernvm16.cern.ch

AGENT>
```

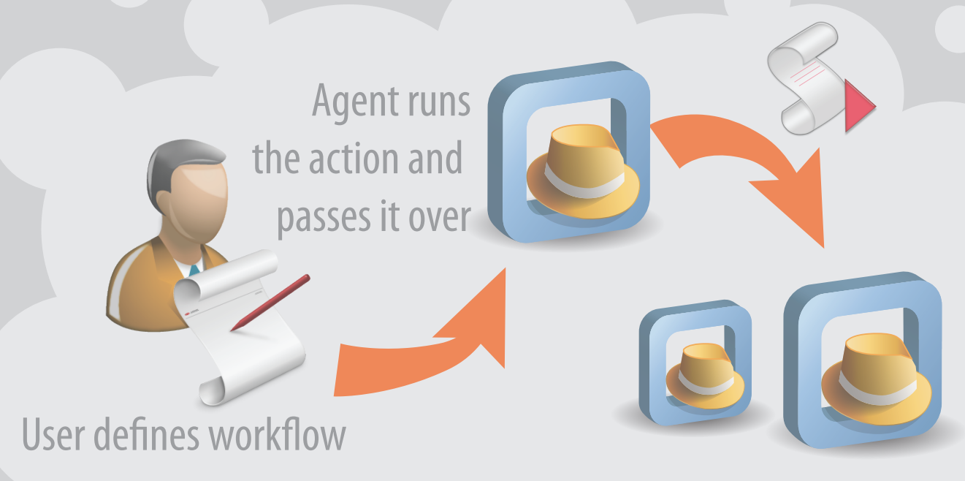
Technical details

The **iAgent framework** is written in Perl and consists of a simple kernel to which worker modules are plugged. The role of the framework is to abstract the communication mechanism and hide the workflow logic from the agents. This reduces the effort required to develop a new agent.

The **communication** is based on the Extensible Messaging and Presence Protocol (XMPP), that is widespread, very scalable, and easily extensible. For command messages we use Info-Query stanzas, and for resource discovery we use the Publish-Subscribe mechanism.

The **workflow** is initiated by the user, who defines it and hands the definition over to the first agent. After performing the relevant task the agent passes the workflow definition on to the next agent in the workflow chain.

Two important features of this approach are the run time discovery of agents, and elimination of the need for a centralized orchestrator.



The **client web applications** are written entirely in HTML5 + Javascript. The clients connect directly to an XMPP server through Bidirectional-streams Over Synchronous HTTP (BOSH) and provide a user-friendly interface to interact with the agents.

The mobile client is implemented using the SenchaTouch framework, whereas the deskop client is based on the front-end of the Archipel Project, and is written in Objective-J.

End users can also access the system using any standard XMPP messenger application.

Future plans

Our immediate plan is to connect this system with our "Long-term preservation of analysis software environment" project, in order to enable **on demand builds** of older CernVM versions.

Another important development would be addition of support for automated VM **deployment,** as well as on-the-fly **contextualization** of newly started VM instances.

We expect to add more features after performing tests of the system at a larger scale and receiving more feedback from users.

Resources and further reading

Get more information about this poster ...

1. iAgent framework svn: https://cernvm.cern.ch/project/trac/cernvm/browser/iagent
2. Mobile client svn: https://cernvm.cern.ch/project/trac/cernvm/browser/iagent-picolient
3. Archipel additions: https://cernvm.cern.ch/project/trac/cernvm/browser/iagent-archipel
4. The Archipel Project: http://archipelproject.org/
5. Project website: http://cernvm.cern.ch/portal/ibuilder
6. The XMPP standards foundation: http://xmpp.org/
7. Sencha Touch Framework: http://www.sencha.com/products/touch/
8. More details about this poster can be found at:
   https://indico.cern.ch/contributionDisplay.py?contribId=134&confId=149557
   or using the QRCode on the side.

See also our other CernVM posters/talks at this CHEP!