



# iCanCloud

# Quick Installation

# Guide

**Jesús Carretero Pérez**

**Gabriel González Castañé**

**Javier Prieto Cepeda**

**Grupo de Arquitectura de Computadores**

**Universidad Carlos III de Madrid**



## Table of contents

1	Introduction .....	3
2	What is iCanCloud? .....	4
2.1	iCanCloud features .....	5
2.2	iCanCloud design.....	7
3	Installing the iCanCloud simulation platform.....	10
3.1	Install OMNeT++ .....	10
3.2	Install INET. ....	12
3.3	Install iCanCloud .....	12

# 1 Introduction

Hi iCanCloud user! This is the first version of the first manual of iCanCloud. The goal of this quickguide is to show you how to compile and use iCanCloud, and also give you a brief overview of this simulation tool focused on cloud computing systems.

Of course, we are (continuously) planning to improve this guide both by adding new sections and by completing the existing ones. If you have any suggestion, please send us e-mail[1].

Due to the fact that iCanCloud is developed on the top of OMNeT++, we strongly recommend you to read the Manual provided by the OMNeT++ framework, which can be found in `~/omnetpp-4.X/doc/Manual.pdf`. Also, the HTML version of that manual can be visited [here](#).

Currently we are working very hard to release a new version by adding new features of iCanCloud with a fantastic IDE.

Enjoy!

The iCanCloud team.

[1] Email: [icancloud@arcos.inf.uc3m.es](mailto:icancloud@arcos.inf.uc3m.es)

## 2 What is iCanCloud?

Simulation techniques have become a powerful tool for deciding the best starting conditions on “pay-as-you-go” scenarios. This is the case of public cloud infrastructures, where a given number and type of virtual machines are instantiated during a specified time, being this reflected in the final budget. In order to develop new proposals aimed at different topics related to cloud computing (for example, datacenter management, or provision of resources), a lot of work and money is required to set up an adequately sized testbed including different datacenters from different organizations and public cloud providers. Even if automated tools exist to do this work, it would still be very difficult to produce performance evaluation in a repeatable and controlled manner, due to the inherent variability of the cloud. Therefore, it is easier to use simulation as a tool for studying complex scenarios.

The ever-increasing complexity of computing systems has made simulators a very important choice for designing and analyzing large and complex architectures. In the field of cloud computing, simulators become especially useful for calculating the trade-offs between cost and performance in pay-as-you-go environments. Hence, this work describes a flexible and scalable simulation platform for modeling and simulating large environments that represent, both actual and non-existent cloud computing architectures.

iCanCloud is a simulation platform aimed to model and simulate cloud computing systems, which is targeted to those users who deal closely with those kinds of systems. The main objective of iCanCloud is to predict the trade-offs between cost and performance of a given set of applications executed in a specific hardware, and then provide to users useful information about such costs. However, iCanCloud can be used by a wide range of users, from basic active users to developers of large distributed applications.

This simulation framework has been developed on the top of OMNeT++ and INET frameworks. Thus, both frameworks are needed to execute and develop new modules for iCanCloud. For more information, please visit the [OMNeT home page](#).

Although each user is intersected on different features provided by the cloud, all of them have the same objective: optimizing the trade-off between cost and performance, which is the real hard task

iCanCloud tries to alleviate. Thus, this simulation platform provides a scalable, flexible, fast and easy-to-use tool, which let users, obtain results quickly in order to help to take a decision for paying a corresponding budget of machines.

The proposed features are desirable for any simulation platform, but their meaning can be blurry depending on the context in which they are used. Scalability means whether the corresponding simulator is able to simulate large-scale systems without losing performance. Likewise, performance determines the speed which a simulator executes a corresponding simulation. In general, the larger the size of the architecture to be simulated, the greater the time needed to execute the simulation. Moreover, a flexible simulator must let users build environments easily, using several component models with different levels of detail. In fact, the proposed hypervisor model let users to integrate any cloud brokering policy to manage a set of fully customizable VMs. Thus, different brokering policies can be fully customized by using this simulation platform.

The simulated cloud computing scenarios are modeled using a set of existent components provided by iCanCloud; they represent the behavior of real components that belong to real architectures like disks, networks, memories, file systems, etc. Those components are hierarchically organized within the repository of iCanCloud, which compose the core simulation engine. Besides designing simulated environments using components provided by iCanCloud, new components can be added to its repository. Moreover, iCanCloud allows an easy substitution of components for a particular component. Those interchangeable components can differ in level of detail (to make performance versus accuracy trade-offs), in the functional behavior of the component, or both.

## 2.1 iCanCloud features

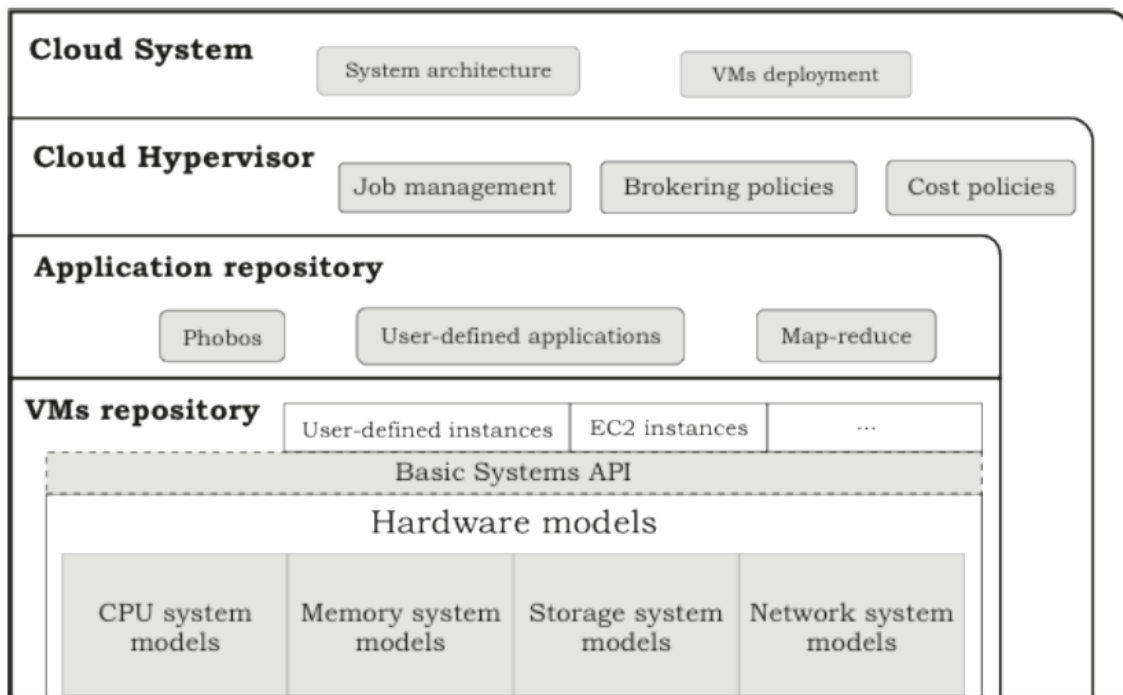
The most remarkable features of the iCanCloud simulation platform include the following:

- Both existing and non-existing cloud computing architectures can be modeled and simulated.
- A flexible cloud hypervisor module provides an easy method for integrating and testing both new and existent cloud brokering policies.

- Customizable VMs can be used to quickly simulate uni-core/multi-core systems.
- iCanCloud provides a wide range of configurations for storage systems, which include models for local storage systems, remote storage systems, like NFS, and parallel storage systems, like parallel file systems and RAID systems.
- iCanCloud provides a user-friendly GUI to ease the generation and customization of large distributed models. This GUI is especially useful for: managing a repository of pre-configured VMs, managing a repository of preconfigured Cloud systems, managing a repository of pre-configured experiments, launching experiments from the GUI, and generating graphical reports.
- iCanCloud provides a POSIX-based API and an adapted MPI library for modelling and simulating applications. Also, several methods for modelling applications can be used in iCanCloud: using traces of real applications; using a state graph; and programming new applications directly in the simulation platform.
- • New components can be added to the repository of iCanCloud to increase the functionality of the simulation platform.

## 2.2 iCanCloud design

The basic idea of a cloud computing system is to provide users a pseudocustomizable hardware environment where they can execute specific software. Therefore, in order to model entire cloud computing systems, the architecture of iCanCloud has been designed based on this principle. Thus, Figure 1 shows the layered architecture of iCanCloud.



**Figure 1 Basic layered schema of iCanCloud architecture**

The bottom of the architecture consists of the hardware models layer. This layer basically contains the models that are in charge of modeling the hardware parts of a system, like disk drives, memory modules and CPU processors. Using those models, entire distributed systems can be modeled and simulated. In turn, this section consists of four groups, where each corresponds to a specific basic system: processing system (CPU), memory system, storage system, and network system.

The basic system's API module is directly connected with the hardware models layer. Basically this module contains a set of system calls which are offered as an API (Application Programming Interface) for all applications executed in a VM modeled using iCanCloud. Thus, those system calls provide the interface between applications and the services provided by the hardware models. Moreover, researchers can write applications to be simulated in iCanCloud using this API. In



order to maintain a certain degree of compatibility, this API pretends to be a subset of POSIX.

Upper layer consists of a VMs repository. This repository contains a collection of VMs previously defined by the user. Initially, the iCanCloud simulator provides few models of existing VMs in well-known clouds like Amazon (EC2). Moreover, users can add, edit or remove VMs from this repository. Each VM is modeled by configuring the corresponding underlying hardware models for each basic system.

In a cloud system, the VM is the most relevant component. Similarly in iCanCloud, a VM is a building block for creating cloud systems. The key of this simulation platform is modularity, which let nested complex modules using other modules previously defined. Thence, the basic idea of iCanCloud consists on using VMs modules for building entire cloud computing systems.

In those systems, VMs are in charge of hiding the hardware details, providing to users a logic view that corresponds with the user requirements. Thus, the VMs models defined in this layer use the previously defined hardware components defined in the bottom layer.

Otherwise, the application repository contains a collection of pre-defined applications customized by users. Similarly to the repository of VMs, initially this repository provides a set of pre-defined application models. Those models will be used in order to configure the corresponding jobs that will be executed in a specific instance of a VM in the system. Moreover, new application models can be easily added to the system, because iCanCloud provides an API in order to ease the development of new application models.

Upper layer, called cloud hypervisor, consists of a module in charge of managing all incoming jobs and the instances of VMs where those jobs are executed. Once a job finishes its execution, this module sets as idle the VMs where that job has been executed, and then re-assign the available resources in the system to execute the remaining jobs. This module also contains cost policies in order to assign incoming jobs to a specific instance calculated by the corresponding heuristic.

Finally, at the top of the architecture is the cloud system module. This module contains a definition of the entire cloud system, which basically consists on the definition of the hypervisor, and the definition of each VM that composes the system.

However, the key innovation of this simulation framework lies in a modular and flexible design. Figure 2 shows the UML 2.3 class diagram of the iCanCloud simulation platform.

This model is split in two different parts. On the one hand, dark grey squares represent the hardware part of the cloud environment. On the other hand, the light grey squares represent those modules in charge of managing the cloud system. The VM class acts as a link among the physical resources of the cloud environment, such as nodes and networks, and the resources used by users, giving them an illusion of using directly the physical resources to execute the corresponding jobs. Thus, depending of the configuration of each VM, those are mapped to the physical nodes existent in the cloud system model. The main module of this section is the hypervisor, which is the center piece of the system.

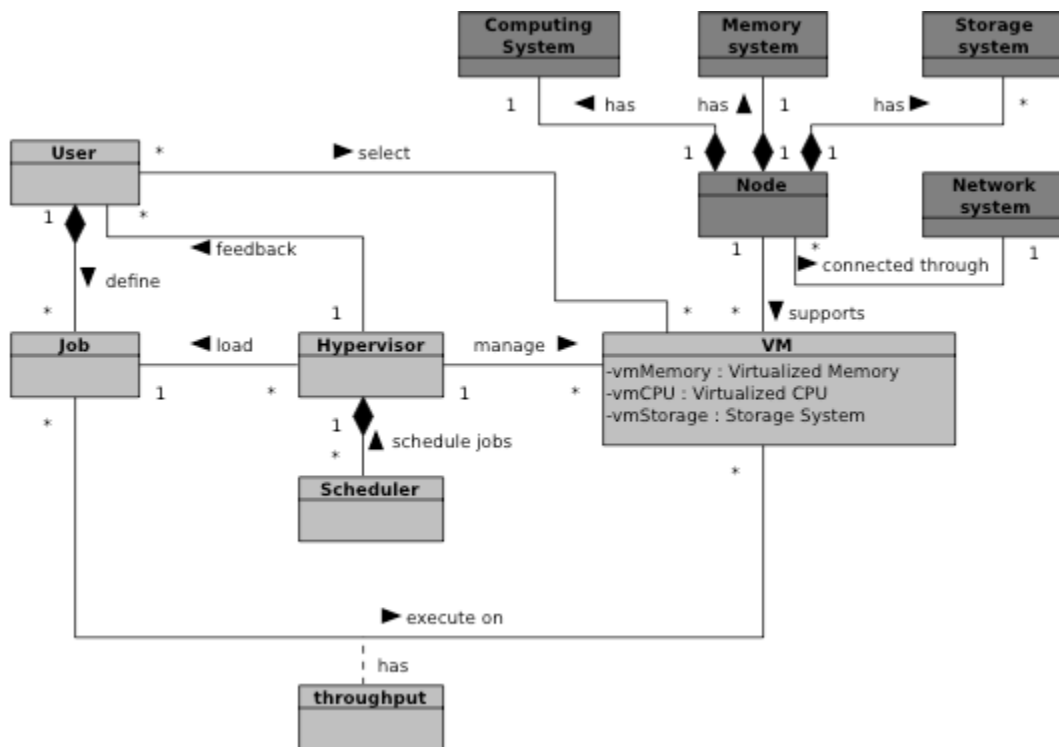


Figure 2 UML class diagram of iCanCloud simulation platform

### 3 Installing the iCanCloud simulation platform

This section provides the steps required to install iCanCloud on Linux Distributions.

First of all, iCanCloud needs both OMNeT++ and INET simulation frameworks, which are provided in the same package that iCanCloud. If you prefer to install a newer version of those frameworks, or just to obtain more information about them, please visit <http://www.omnetpp.org/>.

Install process:

Download iCanCloud from <http://www.arcos.inf.uc3m.es/~icancloud/downloads.html>.

Extract iCanCloud. Now, a folder called omnetpp-4.6 is generated. This folder contains the OMNeT++ framework, the INET framework and iCanCloud:

```
$ tar -zxvf file.tar.gz
```

#### 3.1 Install OMNeT++

Install OMNeT++. There is a very detailed installation guide in `~/omnetpp-4.X/doc/installGuide.pdf`. Installing OMNeT requires (Flex, bison, tcltk ..) .

Before starting the installation, refresh the database of available packages. Type in the terminal:

```
$ sudo apt-get update
```

To install the required packages, type in the terminal:

```
$ sudo apt-get install build essential gcc g++ bison flex  
perl \ tcl-dev tk-dev libxml2-dev zlib1g-dev default-jre \  
doxygen graphviz libwebkitgtk-1.0-0 openmpi-bin \  
libopenmpi-dev libpcap-dev
```

At the confirmation questions (Do you want to continue? [Y/N]), answer Y.

Later, add folder bin to to system PATH.

```
$ vim ~/.bashrc
```

Go to the final of file and add:

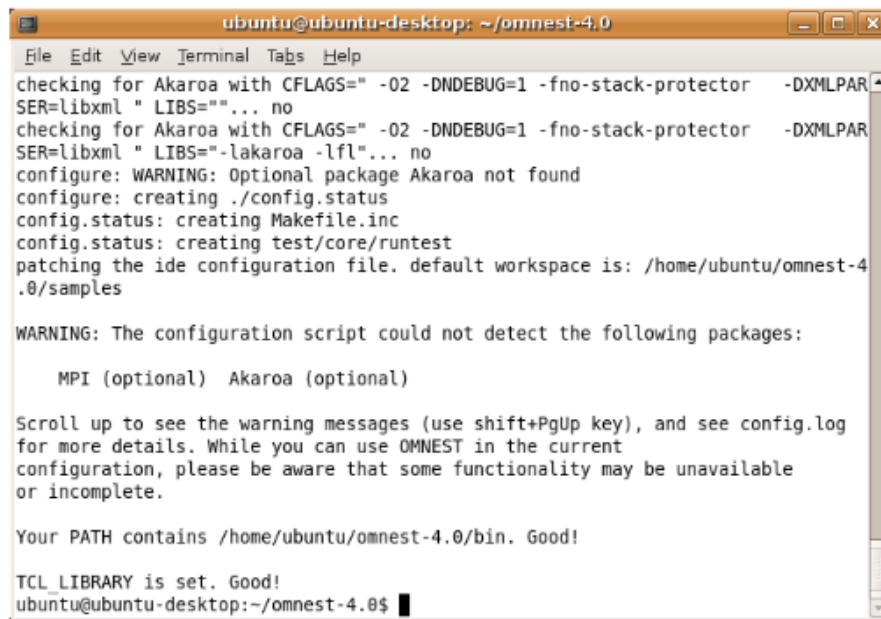
```
export PATH=$PATH:$HOME/omnetpp-4.6/bin
```

\*If you don't have omnet in your personal folder (/home/yourUser), you must insert de route of the folder that have omnetpp-4.6.

Now, go to omnetpp-4.6 directory and execute the configure script:

```
$ cd omnetpp-4.6  
$ ./configure
```

If you make correct this steps, you must see:

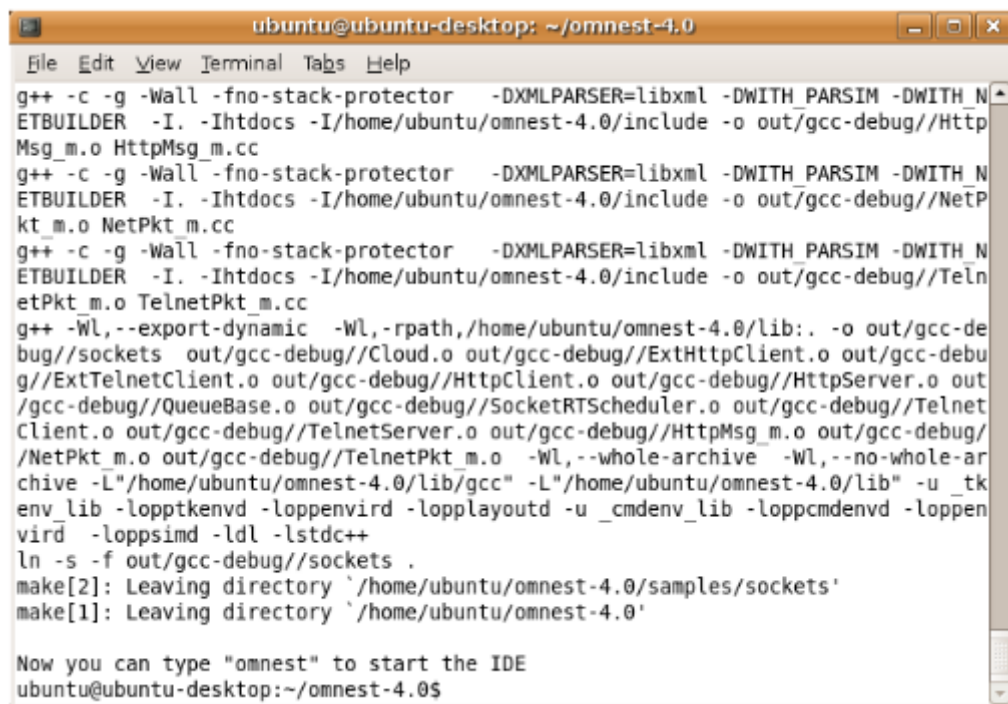


```
ubuntu@ubuntu-desktop: ~/omnest-4.0  
File Edit View Terminal Tabs Help  
checking for Akarua with CFLAGS=" -O2 -DNDEBUG=1 -fno-stack-protector -DXMLPAR  
SER=libxml " LIBS=""... no  
checking for Akarua with CFLAGS=" -O2 -DNDEBUG=1 -fno-stack-protector -DXMLPAR  
SER=libxml " LIBS="-lakarua -lfl"... no  
configure: WARNING: Optional package Akarua not found  
configure: creating ./config.status  
config.status: creating Makefile.inc  
config.status: creating test/core/runtest  
patching the ide configuration file. default workspace is: /home/ubuntu/omnest-4  
.0/samples  
  
WARNING: The configuration script could not detect the following packages:  
  
    MPI (optional)  Akarua (optional)  
  
Scroll up to see the warning messages (use shift+PgUp key), and see config.log  
for more details. While you can use OMNEST in the current  
configuration, please be aware that some functionality may be unavailable  
or incomplete.  
  
Your PATH contains /home/ubuntu/omnest-4.0/bin. Good!  
  
TCL_LIBRARY is set. Good!  
ubuntu@ubuntu-desktop:~/omnest-4.0$
```

When the configure script finish, execute make command:

```
$ make
```

When finish, you must see:



```
ubuntu@ubuntu-desktop: ~/omnest-4.0
File Edit View Terminal Tabs Help
g++ -c -g -Wall -fno-stack-protector -DXMLPARSER=libxml -DWITH_PARSIM -DWITH_N
ETBUILDER -I. -Ihtdocs -I/home/ubuntu/omnest-4.0/include -o out/gcc-debug//Http
Msg_m.o HttpMsg_m.cc
g++ -c -g -Wall -fno-stack-protector -DXMLPARSER=libxml -DWITH_PARSIM -DWITH_N
ETBUILDER -I. -Ihtdocs -I/home/ubuntu/omnest-4.0/include -o out/gcc-debug//NetP
kt_m.o NetPkt_m.cc
g++ -c -g -Wall -fno-stack-protector -DXMLPARSER=libxml -DWITH_PARSIM -DWITH_N
ETBUILDER -I. -Ihtdocs -I/home/ubuntu/omnest-4.0/include -o out/gcc-debug//Teln
etPkt_m.o TelnetPkt_m.cc
g++ -Wl,--export-dynamic -Wl,-rpath,/home/ubuntu/omnest-4.0/lib: -o out/gcc-de
bug//sockets out/gcc-debug//Cloud.o out/gcc-debug//ExtHttpClient.o out/gcc-debu
g//ExtTelnetClient.o out/gcc-debug//HttpClient.o out/gcc-debug//HttpServer.o out
/gcc-debug//QueueBase.o out/gcc-debug//SocketRTScheduler.o out/gcc-debug//Telnet
Client.o out/gcc-debug//TelnetServer.o out/gcc-debug//HttpMsg_m.o out/gcc-debug/
/NetPkt_m.o out/gcc-debug//TelnetPkt_m.o -Wl,--whole-archive -Wl,--no-whole-ar
chive -L"/home/ubuntu/omnest-4.0/lib/gcc" -L"/home/ubuntu/omnest-4.0/lib" -u _tk
env_lib -lopptkenvd -loppenvird -lopplayoutd -u _cmdenv_lib -loppcmdenvd -loppen
vird -loppsind -ldl -lstdc++
ln -s -f out/gcc-debug//sockets .
make[2]: Leaving directory `/home/ubuntu/omnest-4.0/samples/sockets'
make[1]: Leaving directory `/home/ubuntu/omnest-4.0'

Now you can type "omnest" to start the IDE
ubuntu@ubuntu-desktop:~/omnest-4.0$
```

### 3.2 Install INET.

Next step, is the installation of INET. First, change to the INET directory (into the omnetpp-4.6 folder), and type:

```
$ make makefiles
```

When finish, type:

```
$ make
```

### 3.3 Install iCanCloud

Next step, is the installation of iCanCloud. First, change to the iCanCloud directory (into the omnetpp-4.6 folder), and type:

```
$ make
```

Now, you should have everything ready to start iCanCloud.