Universidade do Minho
Escola de Engenharia
Departamento de Informática

SPADE – Smart Python Agent Development Env.

Agentes e Sistemas Multiagente
Guilherme Barbosa, José Machado

ISLab
Synthetic Intelligence Lab

What is SPADE?

- A multi-agent systems platform written in Python (3.8 – 3.12) and based on instant messaging (XMPP).
- Complies with FIPA specifications for interoperable intelligent multi-agent systems
- Library that enables faster time-to-market for developing multi-agent distributed applications
  - Develop agents that can chat both with other agents and humans
- It is Open Source distributed under MIT license

ISLab
Synthetic Intelligence Lab

Summarized:

- Agent library with the purpose of simplifying the development of agents

SPADE provides:

- Agent Execution environment
- Provides a class library for multi-agent development
- Web-graphical administration and management tools for management and monitoring

SPADE Features:

- Provides agent life cycle management
- Message structure and transport management
- Support for agent code and execution state migration
- Agent presence notification (know the agent’s state in real-time)
- Secure multi-agent distributed applications
- Further plugins being developed for SPADE

ISLab
Synthetic Intelligence Lab

SPADE Architecture:

- Agents:
  - Composed of a collection of active components called “Agents”
  - Each agent is a peer and has a unique name/ID
  - Each agent “lives” in a XMPP Server
- Multi-agent Server:
  - Server responsible to enable agents communicating & negotiating between each other
  - One XMPP server can contain several agents running simultaneously
  - The platform provides a homogeneous layer that hides to agents the complexity and the diversity of the unerdying tires (Hardware, Operational System, etc.)

ISLab
Synthetic Intelligence Lab

SPADE Agent Model - Connection mechanism to the platform

- In order for the SPADE agent to establish a connection with the XMPP server, it requires:
  - An identified designated Jabber ID (or JID) – composed by agent_name@server_domain
  - A valid password

- The JID will be the name that uniquely identifies an agent in the platform
  - e.g., myagent@myprovide.com

- Connection establishment are handled internally by means of the XMPP protocol
  - i.e., User Registration & Authentication

ISLab
Synthetic Intelligence Lab

SPADE Agent Model - Message dispatcher

- Each SPADE agent has an internal message dispatcher component (acts as a mailman), e.g.:
  - When a message for the agent arrives, it places it in the correct “mailbox”
  - When the agent needs to send a message, the message dispatcher does the job, putting it in the communication stream

- The message dispatching is done automatically by the SPADE agent library whenever a new message arrives or is to be sent

ISLab
Synthetic Intelligence Lab

SPADE Agent Model - Behaviors

- A behaviour is a task that an agent can execute using repeating patterns
- SPADE provides predefined several behaviour types, which support to implement the different tasks that an agent can perform. These are:
  - *OneShotBehaviour* & *TimeoutBehaviour* – applied to perform casual tasks
  - *PeriodicBehaviour* & *CyclicBehaviour* - applied for performing repetitive tasks
  - *Finite State Machine (FSMBehaviour)* – applied for more complex behaviours to be built
- Each SPADE agent can run several behaviors simultaneously

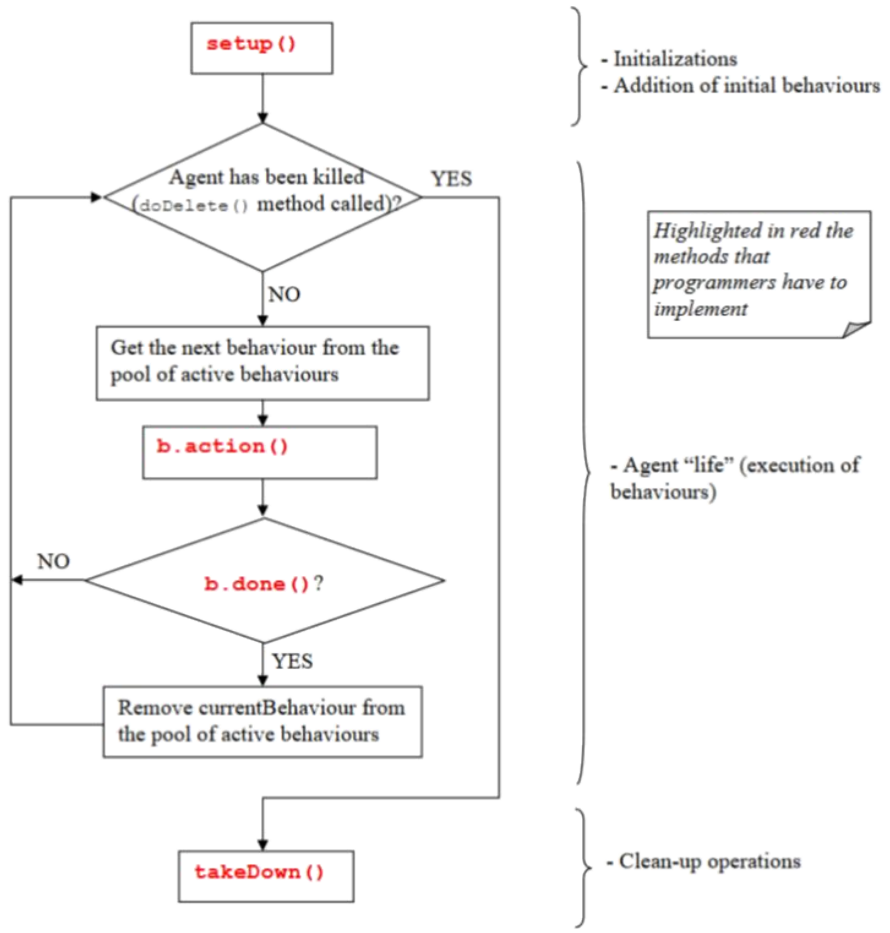
ISLab
Synthetic Intelligence Lab

Programmers Point of View

- SPADE provide a library of classes that programmers can use to develop their agents
  - *Agent* class (spade.agent.Agent): for creating a SPADE agent [e.g., *setup(), get()*]
  - *Behaviour* class (spade.behaviour.*): for carrying out the actual tasks [e.g., *run(), on_start(), on_end()*]
  - *Message* class (spade.message.Message): Agent Communication Language as defined by FIPA, following a syncronous message passing paradigm

ISLab
Synthetic Intelligence
Lab

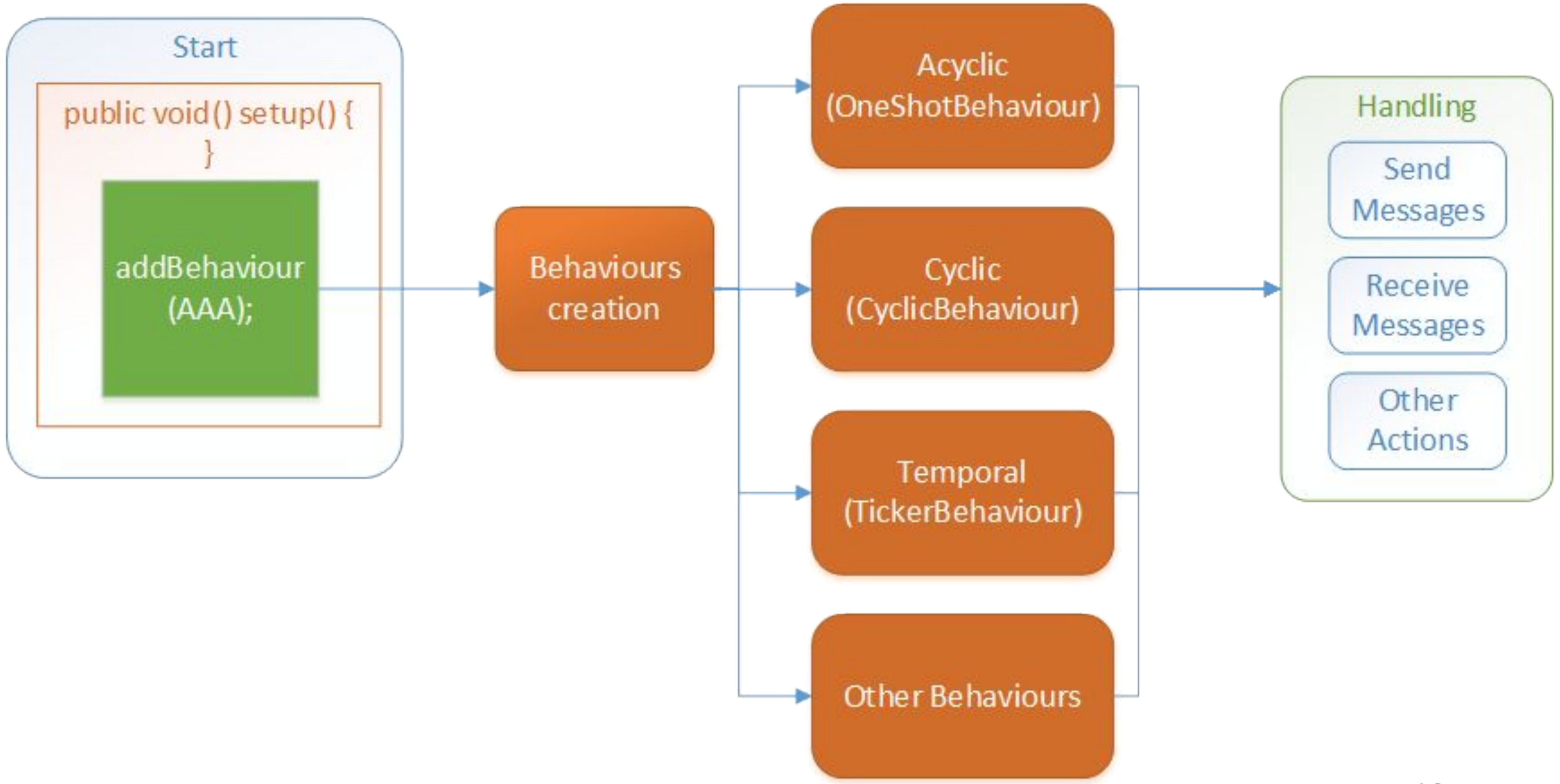
Agent Life Cycle



setup()
- Initializations
- Addition of initial behaviours
Agent has been killed (doDelete() method called)?
YES
NO
Get the next behaviour from the pool of active behaviours
b.action()
b.done()?
NO
YES
Remove currentBehaviour from the pool of active behaviours
Highlighted in red the methods that programmers have to implement
- Agent "life" (execution of behaviours)
takeDown()
- Clean-up operations

Synthetic Intelligence Lab

Execution Pipeline



ISLab
Synthetic Intelligence Lab

Performative - FIPA Compliant

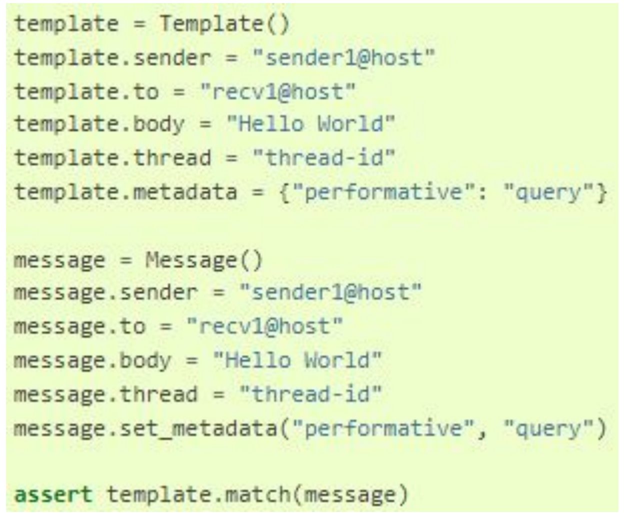
- FIPA - Foundation for Intelligent Physical Agents.
- Promotes standards for agent development.
- Delivers Messages Representation, Agents Structure, Messages Structure, etc.
- FIPA Message Types: ask, inform, request, propose, refuse, etc.

ISLab
Synthetic Intelligence Lab

SPADE Message & Template Structure:

- Message & Template structure share the same attributes:
  - Messages are applied to communicate between agents
  - Templates are applied to dispatch received messages to the behaviour that is waiting for that message

- Message & Template classes are composed by the attributes:
  - **to**: the JID string of the receiver of the message
  - **sender**: the JID string of the sender of the message
  - **body**: the body/content of the message, in string type
  - **thread**: the thread-id of the conversation
  - **metadata**: a (key, value) dictionary of strings to



```
template = Template()
template.sender = "sender1@host"
template.to = "recv1@host"
template.body = "Hello World"
template.thread = "thread-id"
template.metadata = {"performative": "query"}

message = Message()
message.sender = "sender1@host"
message.to = "recv1@host"
message.body = "Hello World"
message.thread = "thread-id"
message.set_metadata("performative", "query")

assert template.match(message)
```

ISLab
Synthetic Intelligence
Lab

Simple Agent Code

SPADE Agent Code

```
import spade

class DummyAgent(spade.agent.Agent):
    async def setup(self):
        print("Hello World! I'm agent {}".format(str(self.jid)))

async def main():
    dummy = DummyAgent("dummy@localhost", "your_password")
    await dummy.start()

if __name__ == "__main__":
    spade.run(main())
```

Execute Agent in XMPP Server

Execute Python Code (Terminal)

```
$ python dummyagent.py
Hello World! I'm agent your_jid@your_xmpp_server
$
```

Note

A SPADE agent is an asyncronous agent. That means that all the code to run an agent must be executed in an asyncronous loop. This is done by the `spade.run()` function. This function receives a coroutine as a parameter and runs it in an async loop. In our example, the `main()` coroutine is the one that is run in the loop.

ISLab
Synthetic Intelligence Lab

Agent Code with Behaviours

```
import time
import asyncio
from spade.agent import Agent
from spade.behaviour import CyclicBehaviour

class DummyAgent(Agent):
    class MyBehav(CyclicBehaviour):
        async def on_start(self):
            print("Starting behaviour . . .")
            self.counter = 0

        async def run(self):
            print("Counter: {}".format(self.counter))
            self.counter += 1
            await asyncio.sleep(1)

    async def setup(self):
        print("Agent starting . . .")
        b = self.MyBehav()
        self.add_behaviour(b)
```

SPADE Agent Code

CyclicBehaviour adding +1 per cycle to self.counter

CyclicBehaviour called by Agent setup funct.

Execute Python Code (Terminal)

```
$ python dummyagent.py
Agent starting . . .
Starting behaviour . . .
Counter: 0
Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 5
Counter: 6
Counter: 7
```

```
async def main():
    dummy = DummyAgent("dummy@localhost", "your_password")
    await dummy.start()
    print("DummyAgent started. Check its console to see the output.")

    print("Wait until user interrupts with ctrl+C")
    await wait_until_finished(dummy)

if __name__ == "__main__":
    spade.run(main())
```

ISLab
Synthetic Intelligence Lab

Create & Send Messages

```python
import time
from spade.agent import Agent
from spade.behaviour import OneShotBehaviour
from spade.message import Message


class SenderAgent(Agent):
    class InformBehav(OneShotBehaviour):
        async def run(self):
            print("InformBehav running")
            msg = Message(to="receiver@your_xmpp_server")     # Instantiate the message
            msg.set_metadata("performative", "inform")  # Set the "inform" FIPA performative
            msg.set_metadata("ontology", "myOntology")  # Set the ontology of the message content
            msg.set_metadata("language", "OWL-S")       # Set the language of the message content
            msg.body = "Hello World"                    # Set the message content

            await self.send(msg)
            print("Message sent!")

            # set exit_code for the behaviour
            self.exit_code = "Job Finished!"

            # stop agent from behaviour
            await self.agent.stop()

    async def setup(self):
        print("SenderAgent started")
        self.b = self.InformBehav()
        self.add_behaviour(self.b)
```

Create Message

Send Message

Execute Python Code (Terminal)

```
$ python sender.py
SenderAgent started
InformBehav running
Message sent!
Agent finished with exit code: Job Finished!
```

Start Sender Agent

```python
async def main():
    senderagent = SenderAgent("sender@your_xmpp_server", "sender_password")
    await senderagent.start(auto_register=True)
    print("Sender started")

    await spade.wait_until_finished(receiveragent)
    print("Agents finished")


if __name__ == "__main__":
    spade.run(main())
```

ISLab
Synthetic Intelligence Lab

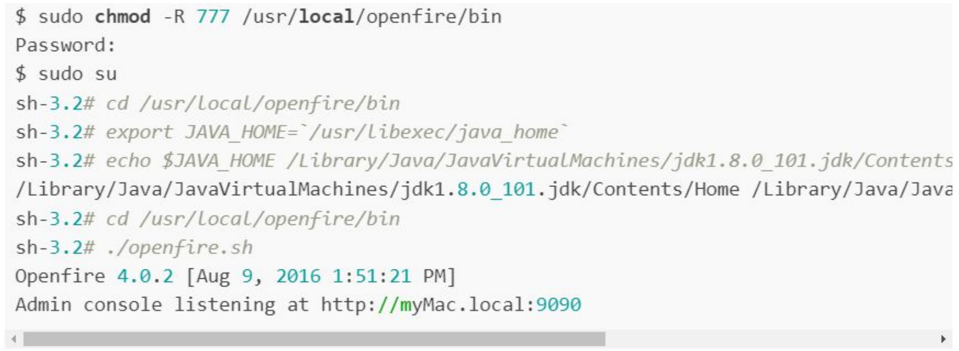
Programming Requirements:

1. Download & Install Java Development Kit (JDK)
2. Download XMPP Server (Openfire)
3. Installation & Configuration Tutorial of XMPP Server (Openfire)
4. Install SPADE Python Package (inside conda env.)
5. Multiagent System Development in Python:
    - Create new Python Project
    - In the Python script, import SPADE module
    - Follow the Quick Start guide to build our first SPADE agent

ISLab
Synthetic Intelligence
Lab

- Tutorial for Linux/Unix:
  - JDK & Openfire Guideline Tutorial

- Tutorial for Mac OS:
  - Homebrew Guideline Tutorial
  - JDK Guideline Tutorial
  - XMPP Guideline Tutorial
  - Openfire Guideline Tutorial
    - Install OpenFire
    - Terminal:

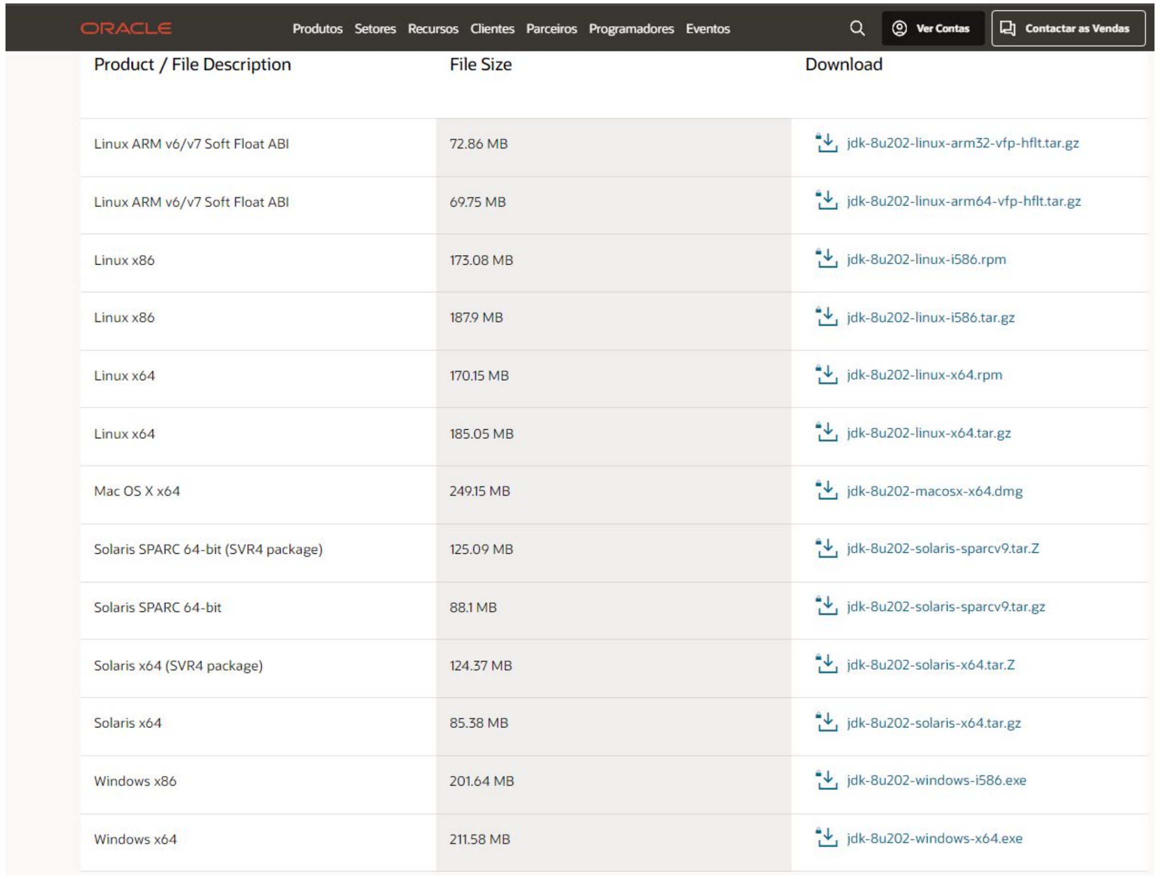


```
$ sudo chmod -R 777 /usr/local/openfire/bin
Password:
$ sudo su
sh-3.2# cd /usr/local/openfire/bin
sh-3.2# export JAVA_HOME=`/usr/libexec/java_home`
sh-3.2# echo $JAVA_HOME /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home /Library/Java/Java
sh-3.2# cd /usr/local/openfire/bin
sh-3.2# ./openfire.sh
Openfire 4.0.2 [Aug 9, 2016 1:51:21 PM]
Admin console listening at http://myMac.local:9090
```

Running these commands above one by one(setup the java_home variable in openfire.xml)

[Note]:
Grant the write permission to openfire.xml before you launching Openfire.

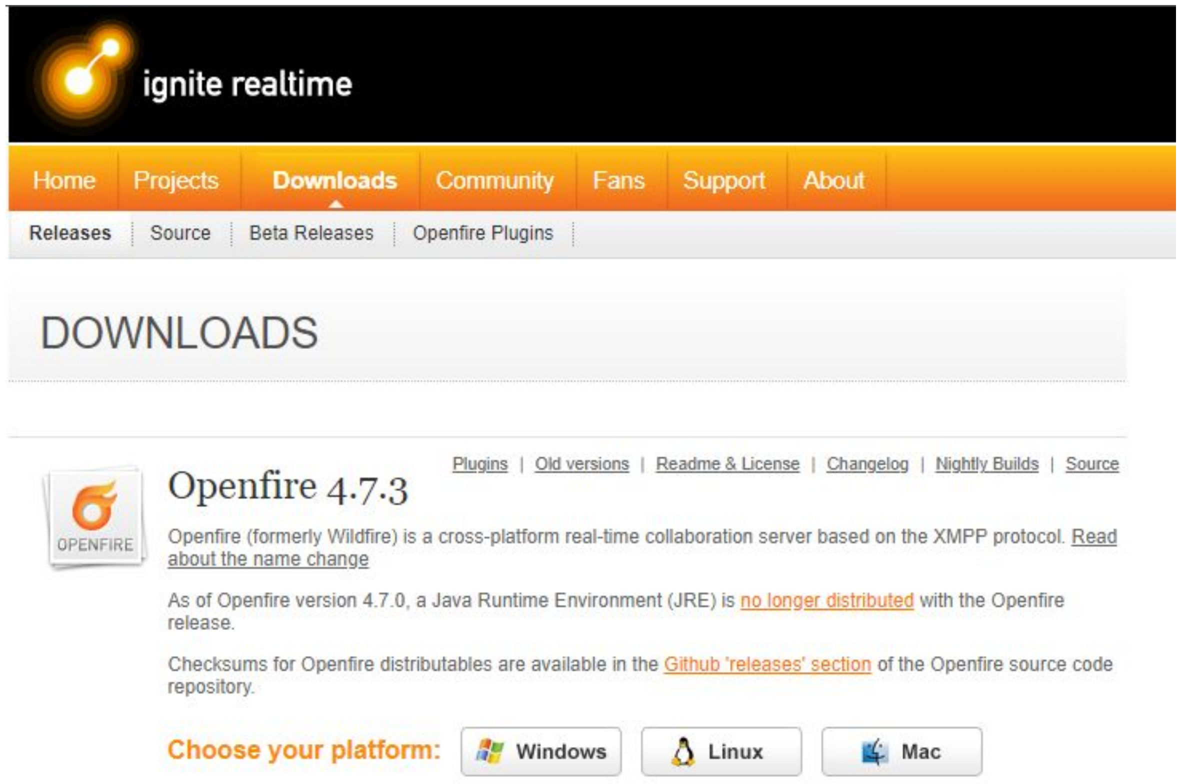
Download & Install Java Development Kit (JDK)


ORACLE
Produtos Setores Recursos Clientes Parceiros Programadores Eventos
Ver Contas
Contactar as Vendas

Product / File Description	File Size	Download
Linux ARM v6/v7 Soft Float ABI	72.86 MB	jdk-8u202-linux-arm32-vfp-hflt.tar.gz
Linux ARM v6/v7 Soft Float ABI	69.75 MB	jdk-8u202-linux-arm64-vfp-hflt.tar.gz
Linux x86	173.08 MB	jdk-8u202-linux-i586.rpm
Linux x86	187.9 MB	jdk-8u202-linux-i586.tar.gz
Linux x64	170.15 MB	jdk-8u202-linux-x64.rpm
Linux x64	185.05 MB	jdk-8u202-linux-x64.tar.gz
Mac OS X x64	249.15 MB	jdk-8u202-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.09 MB	jdk-8u202-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.1 MB	jdk-8u202-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	124.37 MB	jdk-8u202-solaris-x64.tar.Z
Solaris x64	85.38 MB	jdk-8u202-solaris-x64.tar.gz
Windows x86	201.64 MB	jdk-8u202-windows-i586.exe
Windows x64	211.58 MB	jdk-8u202-windows-x64.exe

ISLab
Synthetic Intelligence Lab

Download XMPP Server (Openfire)



DOWNLOADS

Plugins | Old versions | Readme & License | Changelog | Nightly Builds | Source

Openfire 4.7.3

Openfire (formerly Wildfire) is a cross-platform real-time collaboration server based on the XMPP protocol. Read about the name change

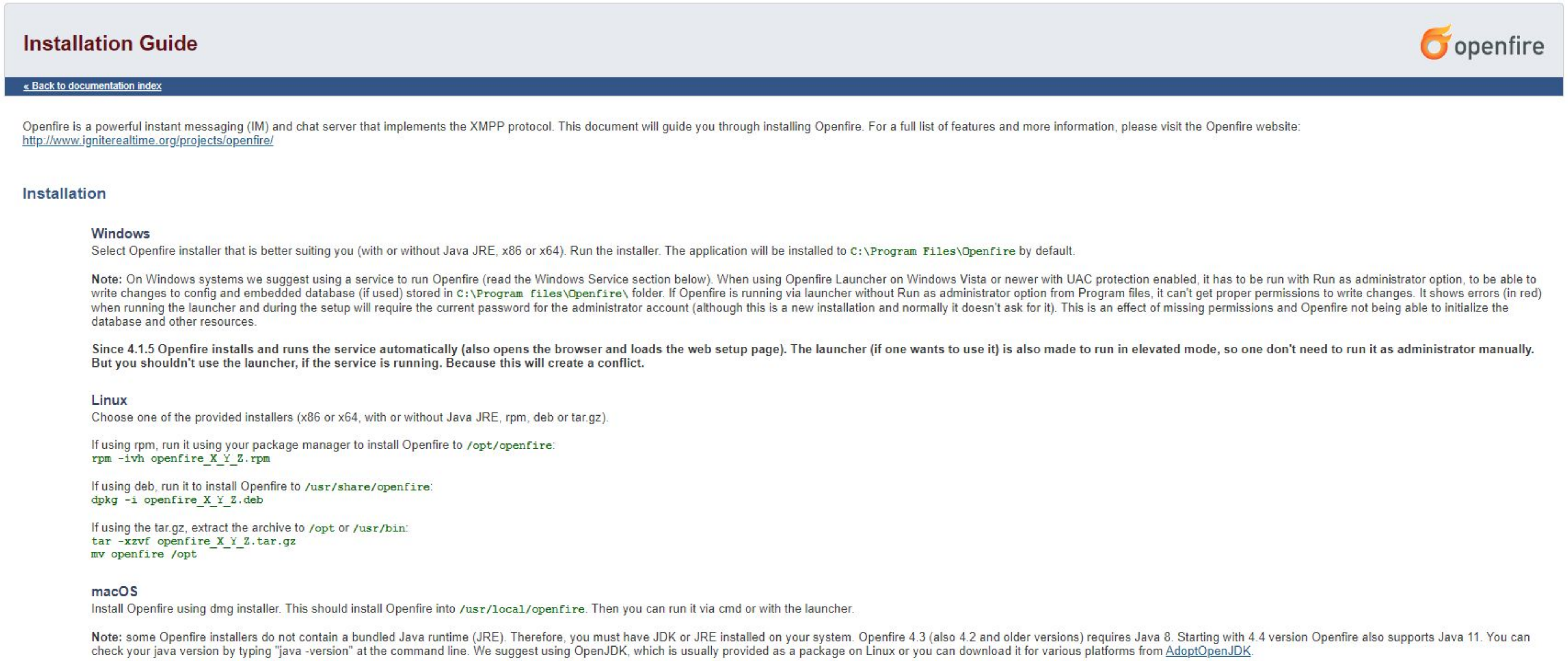
As of Openfire version 4.7.0, a Java Runtime Environment (JRE) is no longer distributed with the Openfire release.

Checksums for Openfire distributables are available in the Github 'releases' section of the Openfire source code repository.

Choose your platform: Windows Linux Mac

ISLab
Synthetic Intelligence Lab

Installation & Configuration Tutorial of XMPP Server (Openfire)



Installation Guide

openfire

« Back to documentation index

Openfire is a powerful instant messaging (IM) and chat server that implements the XMPP protocol. This document will guide you through installing Openfire. For a full list of features and more information, please visit the Openfire website: http://www.igniterealtime.org/projects/openfire/

Installation

Windows
Select Openfire installer that is better suiting you (with or without Java JRE, x86 or x64). Run the installer. The application will be installed to `C:\Program Files\Openfire` by default.

Note: On Windows systems we suggest using a service to run Openfire (read the Windows Service section below). When using Openfire Launcher on Windows Vista or newer with UAC protection enabled, it has to be run with Run as administrator option, to be able to write changes to config and embedded database (if used) stored in `C:\Program files\Openfire\` folder. If Openfire is running via launcher without Run as administrator option from Program files, it can't get proper permissions to write changes. It shows errors (in red) when running the launcher and during the setup will require the current password for the administrator account (although this is a new installation and normally it doesn't ask for it). This is an effect of missing permissions and Openfire not being able to initialize the database and other resources.

Since 4.1.5 Openfire installs and runs the service automatically (also opens the browser and loads the web setup page). The launcher (if one wants to use it) is also made to run in elevated mode, so one don't need to run it as administrator manually. But you shouldn't use the launcher, if the service is running. Because this will create a conflict.

Linux
Choose one of the provided installers (x86 or x64, with or without Java JRE, rpm, deb or tar.gz).

If using rpm, run it using your package manager to install Openfire to `/opt/openfire`:
```
rpm -ivh openfire_X_Y_Z.rpm
```

If using deb, run it to install Openfire to `/usr/share/openfire`:
```
dpkg -i openfire_X_Y_Z.deb
```

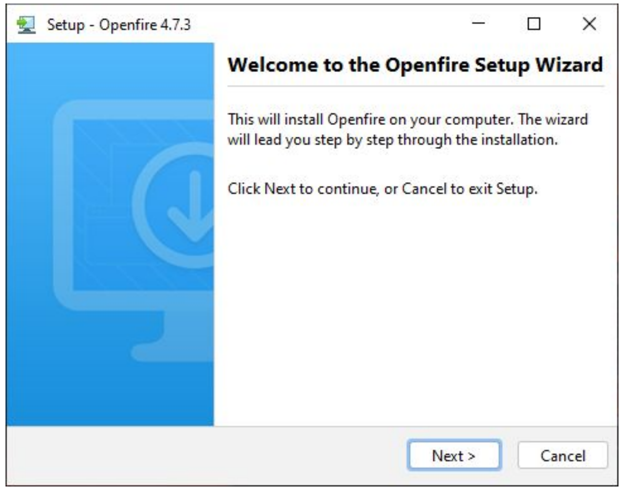
If using the tar.gz, extract the archive to `/opt` or `/usr/bin`:
```
tar -xzvf openfire_X_Y_Z.tar.gz
mv openfire /opt
```

macOS
Install Openfire using dmg installer. This should install Openfire into `/usr/local/openfire`. Then you can run it via cmd or with the launcher.

Note: some Openfire installers do not contain a bundled Java runtime (JRE). Therefore, you must have JDK or JRE installed on your system. Openfire 4.3 (also 4.2 and older versions) requires Java 8. Starting with 4.4 version Openfire also supports Java 11. You can check your java version by typing "java -version" at the command line. We suggest using OpenJDK, which is usually provided as a package on Linux or you can download it for various platforms from AdoptOpenJDK.

ISLab
Synthetic Intelligence Lab

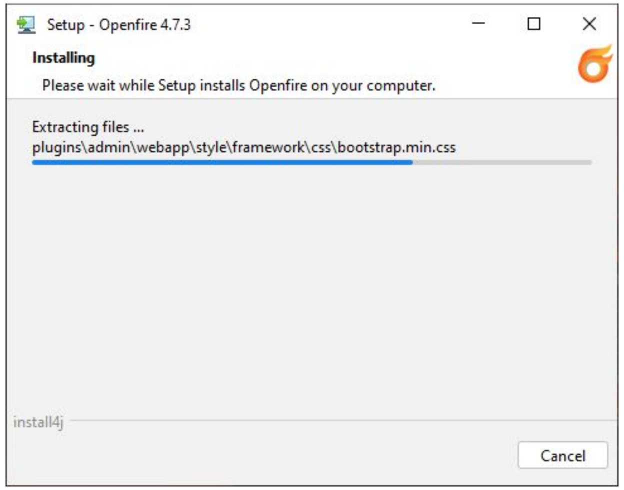
Installation & Configuration Tutorial of XMPP Server (Openfire)


Setup - Openfire 4.7.3

Welcome to the Openfire Setup Wizard

This will install Openfire on your computer. The wizard will lead you step by step through the installation.

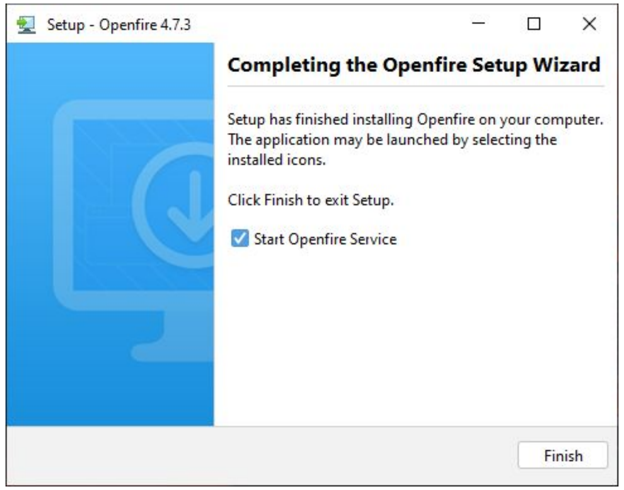
Click Next to continue, or Cancel to exit Setup.


Setup - Openfire 4.7.3

Installing

Please wait while Setup installs Openfire on your computer.

Extracting files ...
plugins\admin\webapp\style\framework\css\bootstrap.min.css


Setup - Openfire 4.7.3

Completing the Openfire Setup Wizard

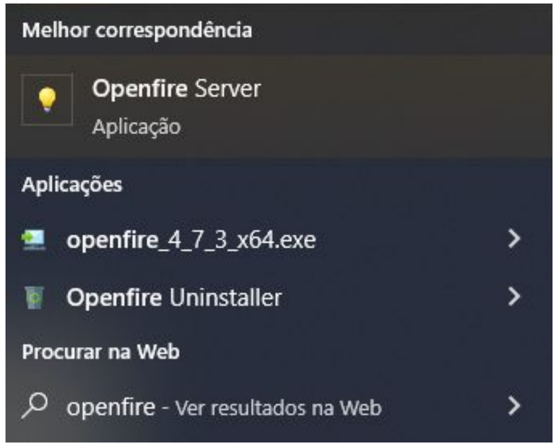
Setup has finished installing Openfire on your computer. The application may be launched by selecting the installed icons.

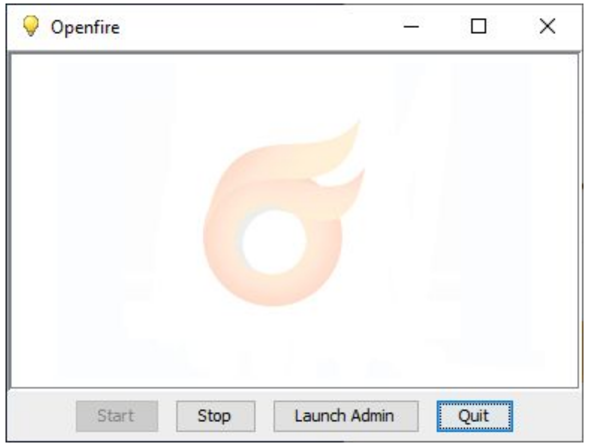
Click Finish to exit Setup.

[✓] Start Openfire Service

ISLab
Synthetic Intelligence Lab

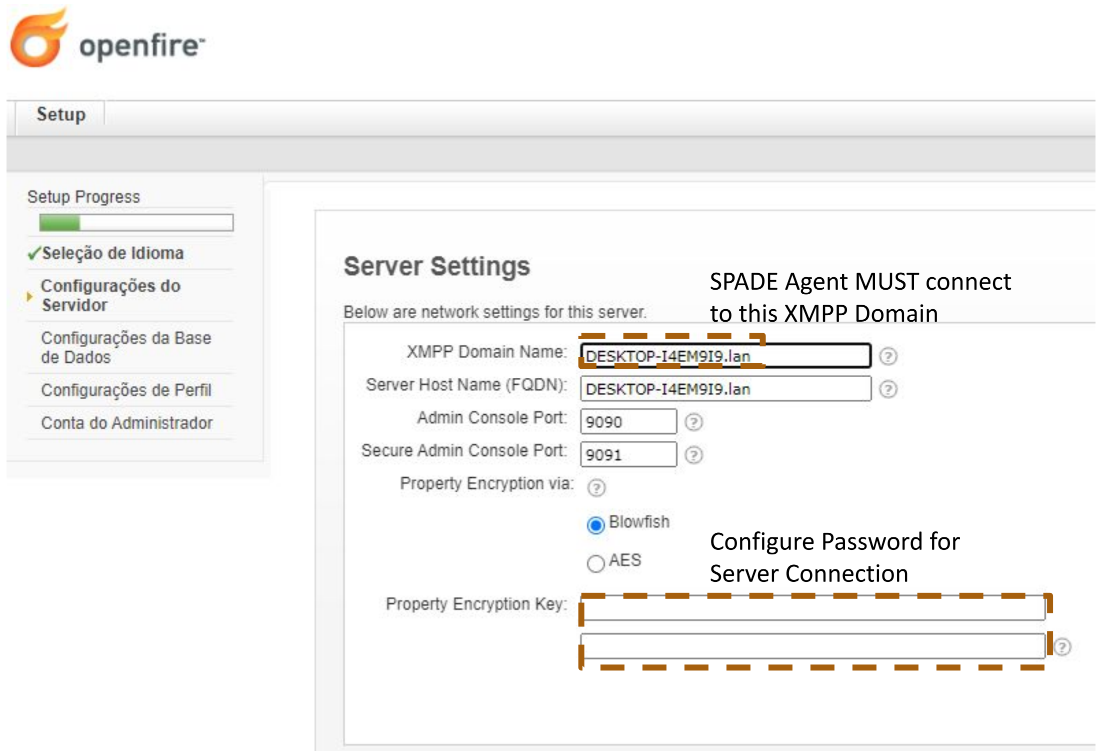
Installation & Configuration Tutorial of XMPP Server (Openfire)


Melhor correspondência
Openfire Server
Aplicação
Aplicações
openfire_4_7_3_x64.exe
Openfire Uninstaller
Procurar na Web
openfire - Ver resultados na Web



ISLab
Synthetic Intelligence Lab

Installation & Configuration Tutorial of XMPP Server (Openfire)



Setup Progress
✓Seleção de Idioma
Configurações do Servidor
Configurações da Base de Dados
Configurações de Perfil
Conta do Administrador

Server Settings

Below are network settings for this server.

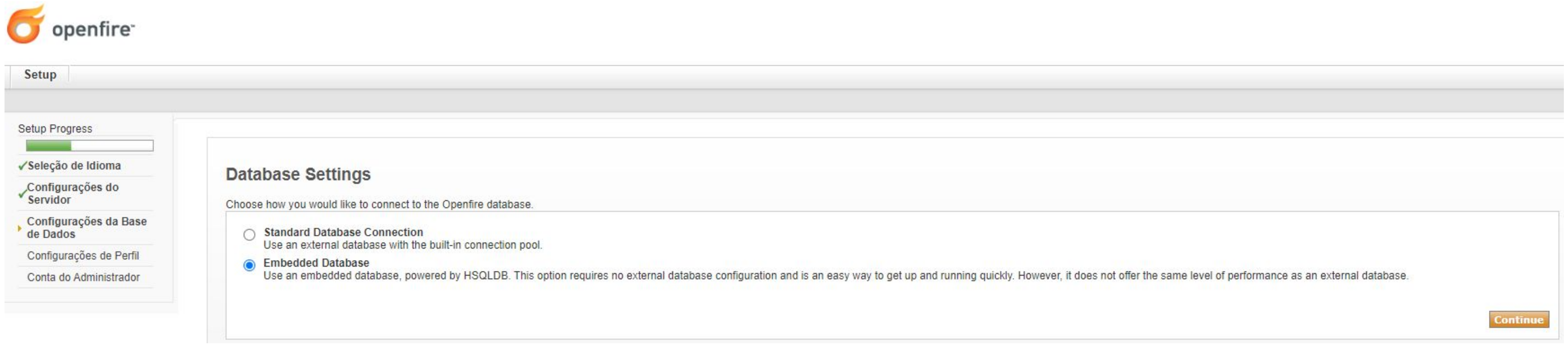
SPADE Agent MUST connect to this XMPP Domain

XMPP Domain Name:	DESKTOP-I4EM9I9.lan
Server Host Name (FQDN):	DESKTOP-I4EM9I9.lan
Admin Console Port:	9090
Secure Admin Console Port:	9091
Property Encryption via:	[✓ Blowfish] AES
Property Encryption Key:	

Configure Password for Server Connection

ISLab
Synthetic Intelligence
Lab

Installation & Configuration Tutorial of XMPP Server (Openfire)


openfire

Setup

Setup Progress

✓ Seleção de Idioma
✓ Configurações do Servidor
Configurações da Base de Dados
Configurações de Perfil
Conta do Administrador

Database Settings

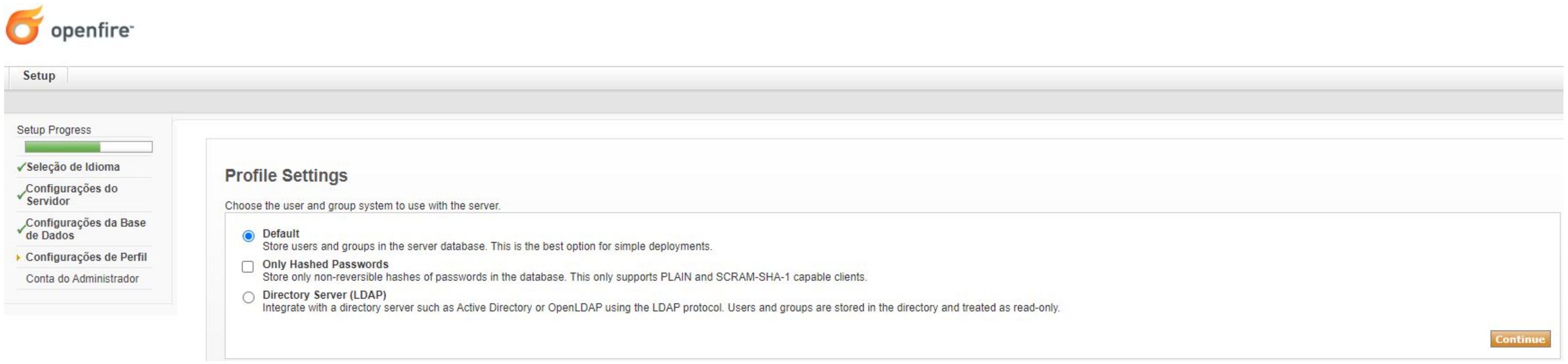
Choose how you would like to connect to the Openfire database.

○ **Standard Database Connection**
Use an external database with the built-in connection pool.

● **Embedded Database**
Use an embedded database, powered by HSQLDB. This option requires no external database configuration and is an easy way to get up and running quickly. However, it does not offer the same level of performance as an external database.

ISLab
Synthetic Intelligence Lab

Installation & Configuration Tutorial of XMPP Server (Openfire)



Setup Progress

- ✓ Seleção de Idioma
- ✓ Configurações do Servidor
- ✓ Configurações da Base de Dados
- Configurações de Perfil
- Conta do Administrador

Profile Settings

Choose the user and group system to use with the server.

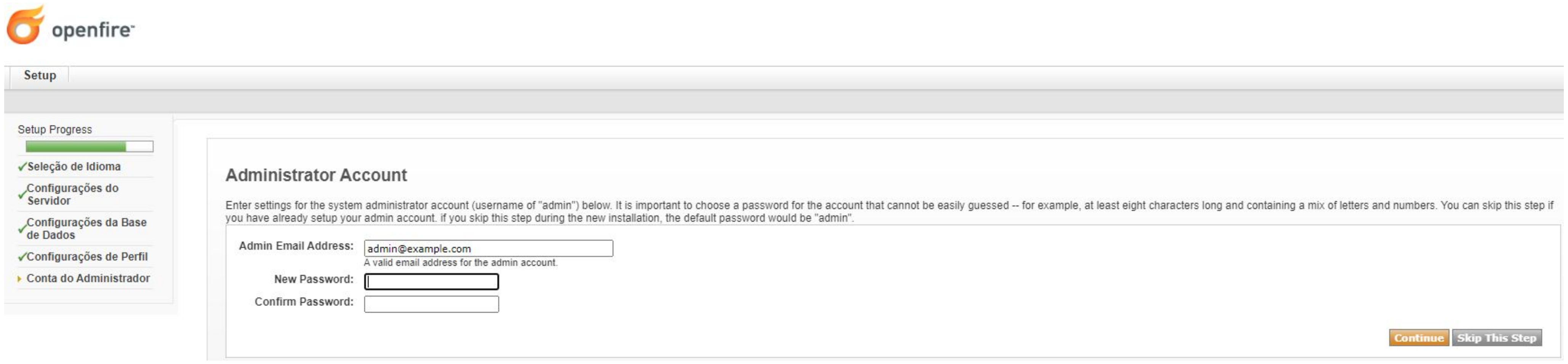
[✓ **Default**]
Store users and groups in the server database. This is the best option for simple deployments.

☐ **Only Hashed Passwords**
Store only non-reversible hashes of passwords in the database. This only supports PLAIN and SCRAM-SHA-1 capable clients.

○ **Directory Server (LDAP)**
Integrate with a directory server such as Active Directory or OpenLDAP using the LDAP protocol. Users and groups are stored in the directory and treated as read-only.

ISLab
Synthetic Intelligence Lab

Installation & Configuration Tutorial of XMPP Server (Openfire)



openfire

Setup

Setup Progress

✓Seleção de Idioma
✓Configurações do Servidor
✓Configurações da Base de Dados
✓Configurações de Perfil
Conta do Administrador

Administrator Account

Enter settings for the system administrator account (username of "admin") below. It is important to choose a password for the account that cannot be easily guessed -- for example, at least eight characters long and containing a mix of letters and numbers. You can skip this step if you have already setup your admin account. if you skip this step during the new installation, the default password would be "admin".

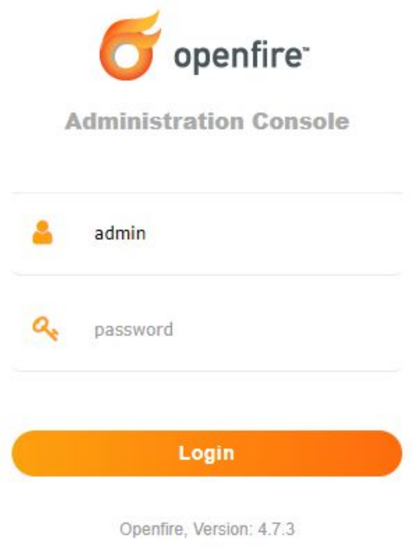
Admin Email Address:	admin@example.com A valid email address for the admin account.
New Password:	
Confirm Password:	

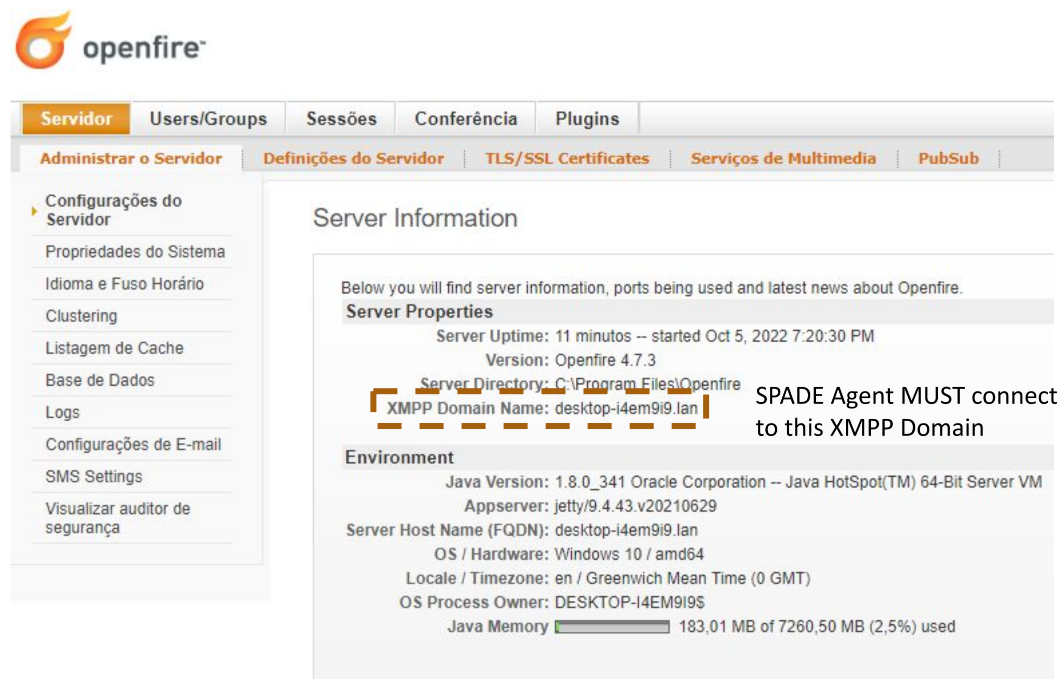
Continue | Skip This Step

Synthetic Intelligence Lab

Installation & Configuration Tutorial of XMPP Server (Openfire)

- Validate if XMPP Server is Configured and Online
- Open Browser: http://127.0.0.1:9090/
  - Username: admin
  - Password: admin





Server Information

Below you will find server information, ports being used and latest news about Openfire.

Server Properties

Server Uptime:	11 minutos -- started Oct 5, 2022 7:20:30 PM
Version:	Openfire 4.7.3
Server Directory:	C:\Program Files\Openfire
XMPP Domain Name:	desktop-i4em9i9.lan

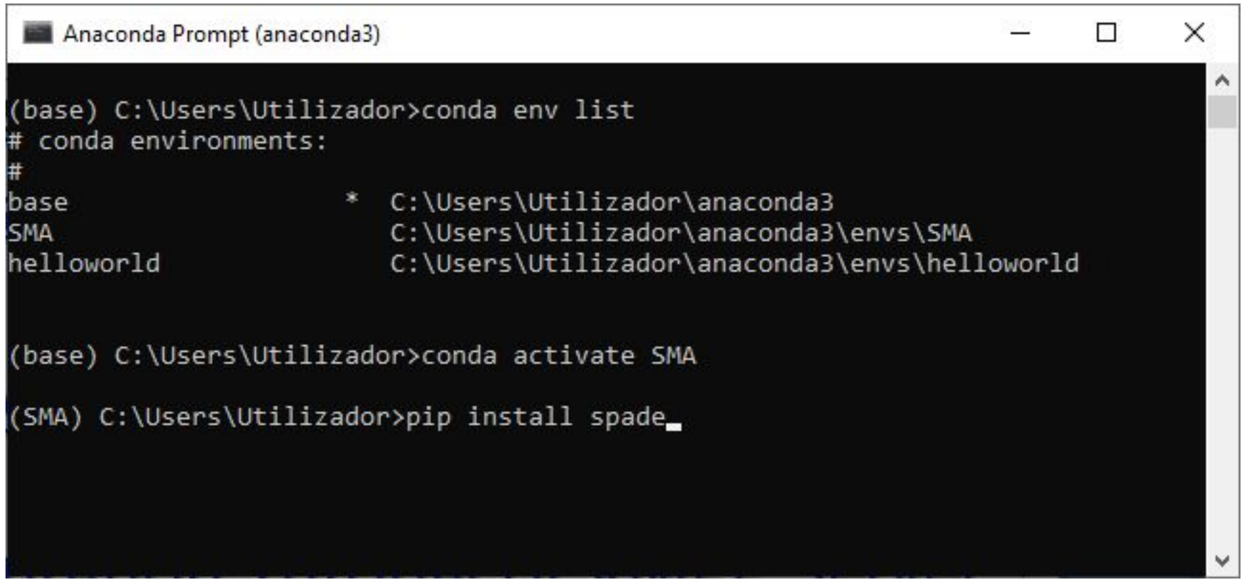
Environment

Java Version:	1.8.0_341 Oracle Corporation -- Java HotSpot(TM) 64-Bit Server VM
Appserver:	jetty/9.4.43.v20210629
Server Host Name (FQDN):	desktop-i4em9i9.lan
OS / Hardware:	Windows 10 / amd64
Locale / Timezone:	en / Greenwich Mean Time (0 GMT)
OS Process Owner:	DESKTOP-I4EM9I9$
Java Memory	183,01 MB of 7260,50 MB (2,5%) used

SPADE Agent MUST connect to this XMPP Domain

ISLab
Synthetic Intelligence Lab

Install SPADE Python Package (inside conda env.)



Anaconda Prompt (anaconda3)

```
(base) C:\Users\Utilizador>conda env list
# conda environments:
#
base                  *  C:\Users\Utilizador\anaconda3
SMA                      C:\Users\Utilizador\anaconda3\envs\SMA
helloworld               C:\Users\Utilizador\anaconda3\envs\helloworld


(base) C:\Users\Utilizador>conda activate SMA

(SMA) C:\Users\Utilizador>pip install spade_
```

Universidade do Minho
Escola de Engenharia
Departamento de Informática

SPADE – Smart Python Agent Development Env.

Agentes e Sistemas Multiagente

Guilherme Barbosa, José Machado