R & D Workshop
Tomcat in the Cloud
Initial Plan

Ismaïl Senhaji
ismail.senhaji@unifr.ch

Guillaume Pythoud
guillaume.pythoud@unifr.ch

Supervised by
Jean-Frederic Clere
jclere@redhat.com

March 26, 2017

Contents

1 Project description

1.1 Project and context

For this project, we have been assigned to *Red Hat*[1]. Red Hat is an international software company, specializing in open source. Among others, they develop the *OpenShift Container Platform*[2], which is full-stack cloud management solution.

We will work on the *Tomcat*[3] Java web server. Our task is to extend Tomcat in order to make session replication work in a cloud environment.

As it stands, session replication in Tomcat is not cloud ready yet. The main issue is that it uses multicast which only works in a local network. In a cloud deployment, instances can be distributed over the internet. This calls for the development of a new session replication solution.

1.2 Goals and objectives of the project

The major goal of the project to make Tomcat's session work in the cloud. This goal is divided in 3 steps:

1. Study existing methods
2. Implement a new solution based on one of those methods
3. Test the implementation

To test our implementation, we will develop a Java web application. It will simply display these informations:

- The ID of the Tomcat instance that served the request
- The user's session ID
- A counter that is incremented on every request.

A secondary goal is to build and configure a Raspberry Pi cluster, running OpenShift. If successful, it will be presented as a demonstration at the end of the project.

Another important goal is to provide detailed documentation for future users. We will write a "Quick Start Guide" detailing the steps to follow to have a working Tomcat cluster running in an OpenShift cloud.

[1]http://www.redhat.com/en/about
[2]https://www.openshift.com/container-platform/index.html
[3]https://www.openshift.com/container-platform/index.html

2 Project organisation

We are working on this project as a team of two. We plan on reserving two days per week to work exclusively on this project, and will aim to meet at least once a week to discuss issues and synchronize our efforts. All development will take place on the GitHub repository[4] we set up for this project. The repository's wiki contains a logbook that will periodically be updated during the project.

2.1 Responsibility distribution

- Ismaïl Senhaji, Student
  ismail.senhaji@unifr.ch
- Guillaume Pythoud, Student
  guillaume.pythoud@unifr.ch
- Jean-Frédéric Clere, Project supervisor at Red Hat
  jclere@redhat.com
- Dr. Hugues Mercier, Workshop coordinator
  hugues.mercier@gmail.com

2.2 Interactions with the client

Communication with Jean-Frederic Clere is done mainly via e-mail. As the Red Hat office is located in Neuchâtel, we also have the possibility to meet there in person to discuss things more thoroughly, and in fact we have already done so twice.

Jean-Frederic Clere has been given access to our GitHub repository and can monitor our progress.

3 Methodology

Most development will be done in Java, since Tomcat itself is written in Java. But we suspect that most time will be spent on reading documentation and code than on writing code.

As mentioned above, all our work is available on our GitHub repository, be it code, documentation or simply comments and ideas.

For testing our solutions, we will use MiniShift[5]. MiniShift is a tool that deploys a "local" (i.e consisting of a unique machine) OpenShift cluster on any computer. This will allow us to run tests on our laptops, saving the time necessary to deploy a real OpenShift installation. Only towards the

[4]https://www.openshift.com/container-platform/index.html
[5]https://www.openshift.org/minishift/

end of the project, when everything else is done, will we finally deploy our solution on a real cluster.

3.1 State of the art

Our research has shown that Tomcat has two built-in session replication solutions that could be built on[6]:

1. *DeltaManager*: this session manager replicates session data to all other instances in a cluster. Peer discovery can be either be done statically or dynamically. In static mode, a list of peers is hardcoded in Tomcat's configuration file. In dynamic mode, peers are discovered using multicast.

2. *PeristanceManager*: session data is stored in the filesystem or in a JDBC-capable database.

Paths to explore are modifying DeltaManager to discover peer using a different method (since multicast over the internet isn't feasible), or configure PersistanceManager to use a distributed datastore. The first solution might be more interesting from a performance point of view.

Finally, if none of the solutions above turn out fruitful, there still are other paths to explore: there are solutions that bypass Tomcat's session manager and use an external one instead. The *Spring Framework*[7] offers such a solution, and is advertised to have built-in support for distributed datastores such as *Redis* or *Hazelcast*[8]. Another solution ports WildFly's session manager to Tomcat[9].

While these solutions using external session managers are known to work, we will mainly concentrate on those based on Tomcat, as they avoid relying on too many external dependencies. But we consider them as a "Plan B" if all else fails.

4 Constraints and elements of risk

In this project, we face two major risks:

The first is getting lost in the complexity of the Tomcat and OpenShift ecosystems. Fortunately, these projects are usually well-documented. The open source community is also very helpful: we can find support on discussion forums and IRC chatrooms. And since the projects are open source, we can study the source code to gain insight on its inner workings.

[6]https://tomcat.apache.org/tomcat-8.5-doc/cluster-howto.html

[7]http://projects.spring.io/spring-framework/

[8]http://docs.spring.io/spring-session/docs/current/reference/html5/

[9]https://github.com/wildfly-clustering/wildfly-clustering-tomcat

The second risk is not being able to produce a working solution in time. Our main target is to develop a solution based on DeltaManager of PersistanceManager as explained above. If this turns out to be too time-consuming, technically difficult, or plain impossible, we will fall back to "Plan B" (using an external session manager). We will make this choice before the midterm presentation.

In case we opt for this fallback solution, we will document the reason for our failures, in the hope that these pitfalls might be avoided by someone attempting the same task in the future.

Furthermore, if we fall behind our schedule, we can cancel installing a Raspberry Pi cluster, as this is not a hard requirement.

5 Deliverable goods

At the end of this project we will deliver the following items:

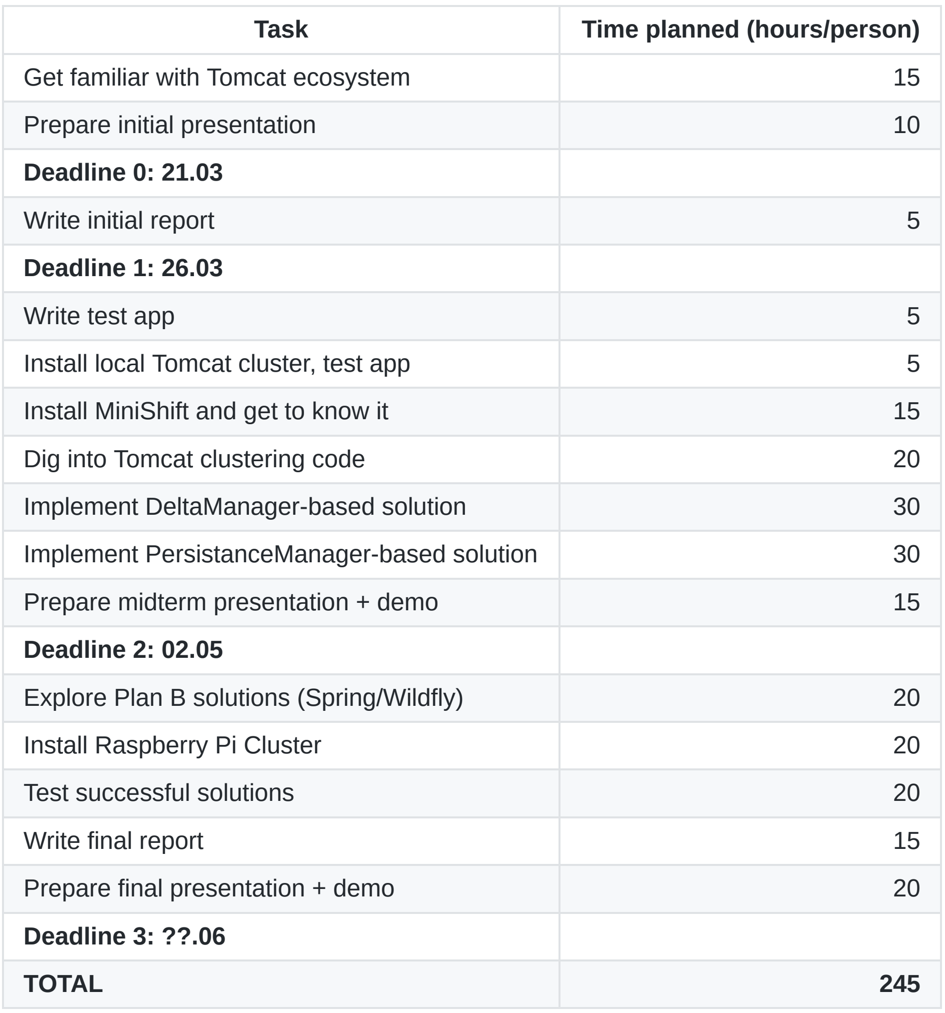
1. A simple Java Web Application to test session replication
2. Source code of our modifications to the Tomcat server
3. Documentation in the form of a "Quick Start Guide", explaining how to easily run Tomcat using our modifications on an OpenShift cloud
4. (optional) A working demonstration of our work running on a Raspberry Pi cluster.

Items 1–3 will be available on our GitHub repository, item 4 will be presented during the final presentation.

6 Efforts and schedule

Our workplan is available on our GitHub repository[10] and is reproduced in figure 1.

[10]https://github.com/iSma/tomcat-in-the-cloud/wiki/Workplan



Task	Time planned (hours/person)
Get familiar with Tomcat ecosystem	15
Prepare initial presentation	10
Deadline 0: 21.03	
Write initial report	5
Deadline 1: 26.03	
Write test app	5
Install local Tomcat cluster, test app	5
Install MiniShift and get to know it	15
Dig into Tomcat clustering code	20
Implement DeltaManager-based solution	30
Implement PersistanceManager-based solution	30
Prepare midterm presentation + demo	15
Deadline 2: 02.05	
Explore Plan B solutions (Spring/Wildfly)	20
Install Raspberry Pi Cluster	20
Test successful solutions	20
Write final report	15
Prepare final presentation + demo	20
Deadline 3: ??.06	
TOTAL	**245**

Figure 1: Workplan