Will Be Defined After the Brainstorming Phase⋆

Eirik Brandztæg[1,2] and Sébastien Mosser[1]

[1] SINTEF IKT, Oslo, Norway
[2] University of Oslo, Oslo, Norway
{firstname.lastname}@sintef.no

Abstract. ∼150 words expected. Will also be defined after the brainstorming phase. Must mention *(i)*the problem, *(ii)*the actual contribution and *(iii)*the obtained results.

1 Introduction

I'll write the introduction afterwards, when the content of the paper will be fixed.

- Cloud–computing research field [2]
- Model–driven engineering applied to the cloud

2 Challenges in the cloud

There are many challanges[1] to cloud deployments, such as **cloud environment configuration** and **automating the deployment**. To recognize these challanges an example application[3] has been utilized. The application (from here known as BankManager) is a featureless bank mananger system written in Grails[5], it supports creating users and bank accounts, moving money between bank accounts and users. BankManager is designed to be distributed between several nodes with two front-end applications connected to one back-end database as seen in Figure 1 with three nodes and one browser to visualize application flow. To prototype deployment of the design we used scripts and s3cmd[4] to create instances and deploy the software. From this prototype it became clear that details such as IP addresses, dependencies between instances, software configurations and level of technical competence could have large impact on resulting deployment. CloudML focuses on coping with challenges such as dynamic allocation and runtime information dependencies. Another barrier when deploying to the cloud that CloudML will handle is the anticipated level of technical compentence nessecary to complete deplyment.
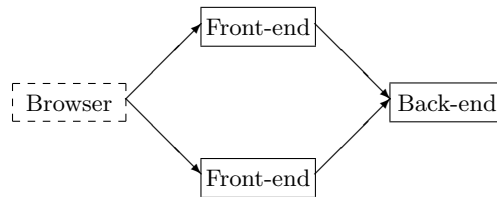
Fig. 1. Nodes sed for deploying BankManager

3 Give it a name too

Name: CloudML, cloudml-engine?

3.1 Showcase 1, the idea (multicloud template language)

- Model-driven approach (template in JSON)
  - Reusability
  - Design / Architecture decoupling
  - Less technical
- Small/Simple template
  - Pros
    * Less complexity
    * Robustness
    * Same between providers
  - Cons
    * Less Flexibility
    * Loss of features and services

3.2 Showcase 2, the engine

- Written in Scala, builds with Maven, works from Java
- Multicloud support
- Validates the example (Can create nodes)
- Can be hosted by third-parties with REST interface

Badish: No support for deployment - yet

If the contribution does not fit in a single section, it can be divided into two sections.

4 Validation & Experiments

- Three instances
    1. Frontend webapp
    2. Frontend webapp
    3. Backend database
- Setup works on
    - AWS EC2
    - Rackspace cloudservers

5 Related Works

- AWS CloudFormation (Amazon only)
- jclouds (Only through (more advance?) code)
- libcloud (Only through (more advance?) code)
- CA Applogic (only graphical, and inhouse)

6 Conclusions

I'll write the conclusions afterwards.

References

1. A. V. Konstantinou, T. Eilam, M.K.A.A.T.W.A., E.Snible: An architecture for virtual solution composition and deployment in infrastructure clouds. Tech. rep., IBM Research (2009)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (Feb 2009), http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html
3. Brandtzg, E.: Bank manager (2012), https://github.com/eirikb/grails-bank-example
4. Ludvig, M.: s3cmd (2012), http://s3tools.org/s3tools
5. SpringSource: Grails (2012), http://grails.org