Cloud4Things: automatic provisioning of smart places infrastructure

Marcus Gomes, Miguel L. Pardal

Instituto Superior Técnico, Universidade de Lisboa
{marcus.paulino.gomes, miguel.pardal}@tecnico.ulisboa.pt

Abstract. Smart places are an ecosystem composed of sensors - e.g. RFID - actuators - e.g. automatic doors - and computing infrastructure - e.g. cloud servers - that are able to acquire data about the surrounding environment and use that data to improve the experience of the people using the place.The data acquired by the sensors needs to be collected, interpreted and transformed into information that is used to gather knowledge about the smart place. A complete example of a platform that allows the transformation of sensor data into information is Fosstrak, an open source RFID software that implements the EPC Network standards. In this paper we propose Cloud4Things, a solution that automates the provisioning of RFID software in the cloud by relying on configuration management tools that leverage existing stacks. We will perform a qualitative evaluation of our solution based on Docker containers with other solutions, such as full Virtual Machines, or tools implementing the TOSCA standards. The current prototype is able to support initial provisioning and day-to-day operations stages in the lifecycle of a smart place.

Keywords: Smart places, Automatic provisioning, Container-based virtualization, Configuration management tools, Cloud applications

1 Introduction

In recent years, computing is becoming more ubiquitous in the physical world. This notion where computational elements are embedded seamlessly in ordinary objects that are connected through a continuous network was introduced many years ago[1]. The progress towards ubiquitous computing has been slower than expected, technology advances such as the mobile Internet contributes to achieve this vision in which individual devices are able to communicate between themselves from any part of the world[2]. Recently, this ubiquitous world is close to becoming reality thanks to the Internet of Things and Cloud Computing [3]. In this vision, physical items are continuously connected to the virtual world and can act as remotely physical access points to Internet Services[4].

A common scenario where the Internet of Things paradigm is applied are smart places [5]. Smart places are an ecosystem composed of sensors - e.g. RFID tags - actuators - e.g. automatic doors - and computing infrastructure - e.g. cloud servers - that are able to acquire data about the surrounding environment and use that data to improve the experience of the people using the place [6].

The smart place life-cycle is composed of several stages, starting from the installation of sensors and readers in the smart place, provisioning the computing infrastructure of the smart place, upload the events that occur in the physical world to the servers, monitoring the quality of service (QoS) of the smart place and eventually deprovisioning the computing infrastructure, as illustrated on Figure 1.

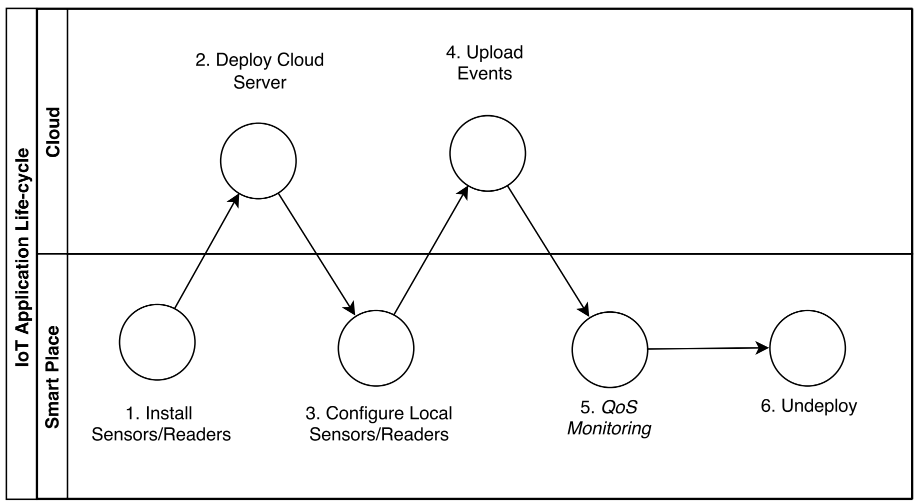

Fig. 1: Smart place life-cycle.

An example of a smart place is a smart warehouse. In the smart warehouse, products and objects are identified with RFID tags. The data transmitted by the tags is consumed by the computing infrastructure of the warehouse and then transformed into information. For instance, Fosstrak[1] can use this information to determine the objects that enter and leaves the smart warehouse.

The smart place infrastructure is divided essentially into two components: the infrastructure in location - the readers, sensors, power, network connection - and the computing infrastructure - servers, database. Is not practical the allocation of the physical infrastructure in the smart place itself. This makes the smart place cost ineffective, since there is a substantial increase in the resource consumption such as power and network connection, and also limits the scalability of the smart place. To solve those problems, an alternative is to allocate the computing infrastructure in the cloud. However, provisioning the computing

[1] Fosstrak is a complete example of a platform that can be used to transform the RFID data into information, since that implements most of the EPC Network standards.

infrastructure in the cloud still is a manual process that requires considerable effort and expertise to be executed.

In order to solve those problems we propose Cloud4Things, a solution that automates a set of stages of the smart place life-cycle, namely the provisioning of the computing infrastructure in the cloud by relying on configuration management tools that leverage existing cloud stacks. With Cloud4Things we want to support the life-cycle of smart place applications, from the initial provisioning to day-to-day operations.

1.1 Overview

The remainder of this paper is organized as follows. In Section 2 we present the related work in the area. Section 3 presents a description of our solution architecture and the current implemented prototype. In Section 4 we will perform a qualitative evaluation and compare our solution with the current approaches to provisioning a smart place infrastructure. Section 5 presents the conclusion of this paper.

2 Related Work

Computing infrastructure in location for IoT applications is cost ineffective and presents low scalability. Recently, the cloud paradigm allowed to move the computing infrastructure to the cloud providers, making possible to increase the application scalability and also to reduce in a significant way the cost related with the smart place infrastructure.

In RFID-based IoT applications, Guinard et al. [7] point out that the deployment of RFID applications are cost-intensive mostly because they involve the deployment of often rather large and heterogeneous distributed systems. As a consequence, these systems are often only suitable for big corporations and large implementations and do not fit the limited resources of small to mid-size businesses and small scale applications both in terms of required skill-set and costs. To address this problem, Guinard et al. propose a cloud-based solution that integrates virtualization technologies and the architecture of the Web and its services. The case of study presented in the paper consists of an IoT application that uses RFID technology to substitute existing Electronic Article Surveillance (EAS) technology, such as those used in clothing stores to track the products. In this scenario they applied the Utility Computing blueprint to the software stack - Fosstrak - required by the application using the Amazon Web Services platform and the Amazon EC2 service. To evaluate the Cloud-based solution, two prototypes was successfully implemented to prove that the pain points of the RFID applications can be relaxed by adopting the proposed solution.

However, provisioning applications for Internet of Things still is an issue, because to the Virtual Machines need to be manually configured and the deployment operation of those applications is specific for each cloud provider.

Amazon Web Services[2] (AWS) offers a variety of services to automate the provisioning of the IT infrastructure at the Amazon Elastic Computing Cloud (EC2). VM Import/Export is a service provided by AWS that enables to import virtual machine images from the development environment to EC2 instances and export them back to the on-premisies development environment. This offering allows to leverage the existing investments in the virtual machines that was built to meet the IT security, configuration management, and compliance requirements by bringing those virtual machines into EC2 as ready-to-use instances. The instances also can be exported to the on-premises virtualization infrastructure, allowing to deploy workloads across the IT infrastructure. Another service provided by AWS is Elastic Beanstalk that allows to quickly deploy and manage an application in the AWS cloud. The application is uploaded to AWS, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling and application health monitoring. Elastic Beanstalk supports several types of applications, including Java, Python, Ruby on Rails and Docker containers.

TOSCA (Topology and Orchestration Specification for Cloud Applications) [8] is a new cloud standard to formally describe the internal topology of application components and the deployment process of cloud applications. TOSCA is proposed in order to improve the reusability of service management processes and automate IoT application ratified by OASIS in deployment in heterogeneous environments. The structure and management of IT services is specified by a meta-model, which consists of a *Topology Template*, that is responsible for describing the structure of a service, then there are the *Artifacts*, that describe the files, scripts and software components necessary to be deployed in order to run the application, and finally the *Plans*, that defined the management process of creating, deploying and terminating a service. The correct topology and management procedure can be inferred by a TOSCA environment just by interpreting the topology template, this is known as "declarative" approach. Plans realize an "imperative" approach that explicitly specifies how each management process should be done. The topology templates, plans and artifacts of an application are packaged in a Cloud Service Archive (.csar file) and deployed in a TOSCA environment, which is able to interpret the models and perform the specified management operations. These .csar files are portable across different Cloud providers, which is a great benefit in terms of deployment flexibility. To demonstrate the feasibility of TOSCA in facilitating IoT application deployment, a typical application in building automation to control an Air Handling Unit (AHU) was used as scenario. The common IoT components, such as gateways and drivers will be modeled, and the gateway-specific artifacts that are necessary for application deployment will also be specified. By archiving the previous specifications and corresponding artifacts into a .csar file, and deploying it in a TOSCA environment, the deployment of AHU application onto various gateways can be automated. As a newly established standard to counter growing

[2] http://aws.amazon.com/

complexity and isolation in cloud applications environments, TOSCA is gaining momentum in industrial adoption as well as academic interest.

3 Cloud4Things

Cloud4Things is a solution that automates the provisioning of software for Internet of Things applications in the cloud. Our solution relies on configuration management tools that leverage existing software stacks - e.g RFID software. In Figure 2 we present the architecture of Cloud4Things.

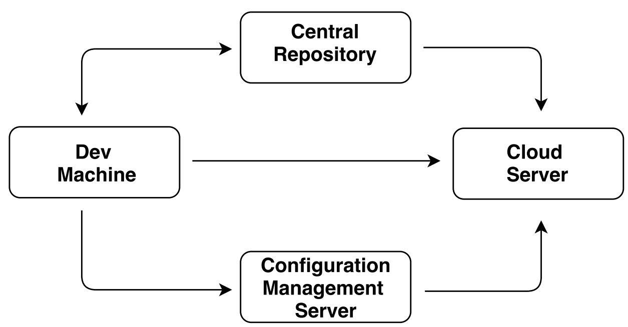

Fig. 2: Cloud4Things conceptual architecture.

In our architecture, the provisioning policies and software images of a smart place are defined and configured in a local environment and then uploaded to its respective repositories. When the provisioned request is performed - through a configuration management interface in a local environment - the configuration management (CM) client in the server node pulls the polices from the configuration management server, a centralized server that is responsible to maintain a consistent state of the provisioned nodes in the cloud. In order to enforce the polices, the CM client pulls the software images from a central repository and then performs the provisioning and configuration of the software. After provisioning the infrastructure the CM client periodically polls the CM server in order to determine if its current state is consistent with the most recent policy.

3.1 Implementation

The current implementation of Cloud4Things relies on the Chef[3] tool. The *recipes* that describe our infrastructure are based on *cookbooks* that are available on the Chef Supermarket[4]. These *recipes* describe how our software stack - are

[3] https://www.chef.io/
[4] https://supermarket.chef.io/

provisioned in the cloud instances. In our current prototype, we will use Docker containers to provisioning the smart place software and we chose to use Amazon Web Services as cloud provider. To provisioning the resources in the Amazon EC2 instances we will use *knife*, a command-line tool developed by Chef that provides an interface between a local Chef repository and the Chef server. The provisioning workflow is illustrated in Figure 3.

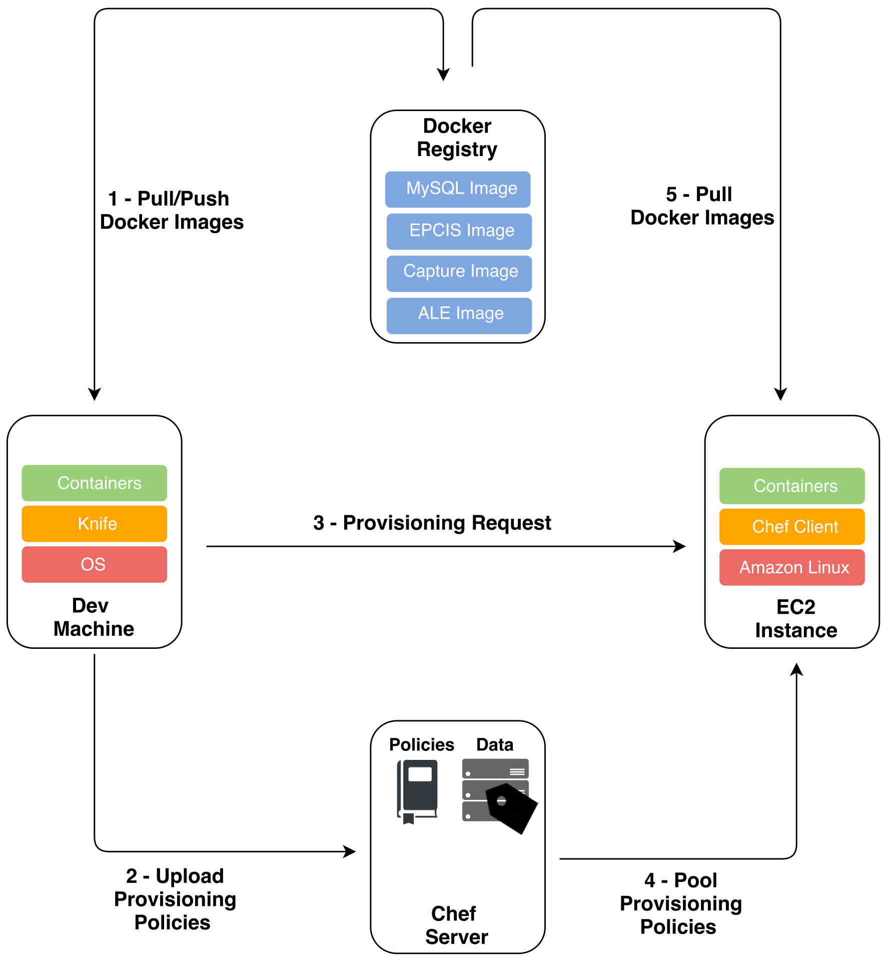

Fig. 3: Automatic provisioning workflow.

In a development environment the Docker images are built and then uploaded to the Docker Registry repository (1). The provisioning of the cloud resources is described in the cookbooks that are uploaded to the Chef server (2). The provisioning request (3) is performed using knife - knife has a plugin for EC2 that allows to describe the image type, the instance type and the policies that need to be applied on each provisioned node. Then the Chef client runs the configuration recipes that are pulled from the Chef server (4). In our solution these configuration recipes describe that our nodes must have a set of Docker containers running on it. The Chef client pulls the Docker images from the remote repository, build the containers based on those images and finally applies the configuration that is associated to each container.

Containers Docker[5] is an open source project to pack, ship and run any application as a lightweight container. Docker containers are *hardware-agnostic* and *platform-agnostic*, this means that these containers can run anywhere, from a laptop to a EC2 compute instance. Since Docker is based in Linux Containers (LXC), the virtualization is performed at operating-system level, different of hypervisor-based solutions where the virtualization is performed at hardware-level. While the effect of both types of virtualization are similar, the virtualization at the operating-system level provides significant benefits compared to hypervisor-based solutions[9]. Docker containers are small, they have low memory and CPU overhead, they also are portable between different virtualization environments.

In our solution, Docker containers are used to provisioning the software stack of the Fosstrak platform. A complete installation of Fosstrak requires a compatible Java SDK, a full MySQL database and a Apache Tomcat server. In order to improve the application scalability we are provisioning a single container for each component of the Fosstrak platform, the EPCIS repository, the Capture application, the ALE server, and also for the MySQL database.

By default each container runs a process that is isolated from the other processes that are executed in the same environment. In order to connect the different modules of the Fosstrak, our containers are linked through the *linking system*[6] provided by Docker. This mechanism creates a secure tunnel between the containers, allowing the recipient container to access select data about the source container. Another benefit that the Docker platform provides is the Docker Registry service, a public repository that stores Docker images used to create the containers. In our solution we built the Docker images of the Fosstrak modules and published them in Docker registry to later be used to create our containers.

Configuration Management Tools Chef is a configuration management tool that allows to describe the infrastructure as code. In that way it is possible to automate how the infrastructure is built, deployed and managed.

Chef architecture is composed of the Chef Server - that stores the recipes and other configuration data - and the Chef Client - that is installed in each server, VM or container, i.e, the nodes that are managed with Chef. The Chef client periodically pulls Chef server latest policy and state of the network, and

[5] https://www.docker.com
[6] https://docs.docker.com/userguide/dockerlinks/

if anything on the node is out of date, the client update its state in order to be consistent with the latest policy.

Chef was built from the ground with the cloud infrastructure in mind. With Chef, is possible to dynamically provision and de-provision the application infrastructure on demand to keep up with peaks in usage and traffic. For instance, Chef offers several plugins for provisioning cloud resources in different hosts such as Amazon EC2, Google Compute Engine and OpenStack.

4 Evaluation

In this section, we will summarize the stages of the smart place life-cycle that already can be automated with our solution. In our evaluation we will compare the software required to provisioning the smart place infrastructure and also discuss about the advantages of our solution compared with the others: full Virtual Machines and tools that implement the TOSCA standard.

4.1 Preliminary Results

As illustrated in Figure 1 the smart place life-cycle is composed of several stages. Our current prototype allows to automate some of these stages and our work in progress intend to cover the remaining stages. Table 1 presents a summary of the smart place life-cycle stages that our solution automates.

Stage	In Progress	Completed	Out of Scope
Install Readers/Sensors			X
Deploy Cloud Server		X	
Configure Readers/Sensors	X		
Upload Events	X		
QoS Monitoring		X	
Undeploy		X	

Table 1: Life-cycle automation summary.

Our current prototype can automate the provisioning and deprovision the smart place computing infrastructure. Regarding the *QoS* monitoring, our solution leverages a service available in AWS (CloudWatch), although we intend to perform the monitoring with a solution that is independent of a particular provider. The remaining stages (reader/sensors configuration and event upload) are in development - the automation of these stages will be available only for solutions that are based on the Fosstrak platform.

4.2 Container-based vs. VM-based solution

We want our provisioned stack to be as small as possible, without the performance of our solution being affected. The provisioning of the RFID software in Cloud4Things is performed through Docker containers. A alternative to provisioning the RFID software stack is to use Virtual Machines. However, full VM presents an overhead regarding the amount of resources that are consumed. In Figure 4 we illustrate a comparison of the stacks between our solution and a full VM solution.

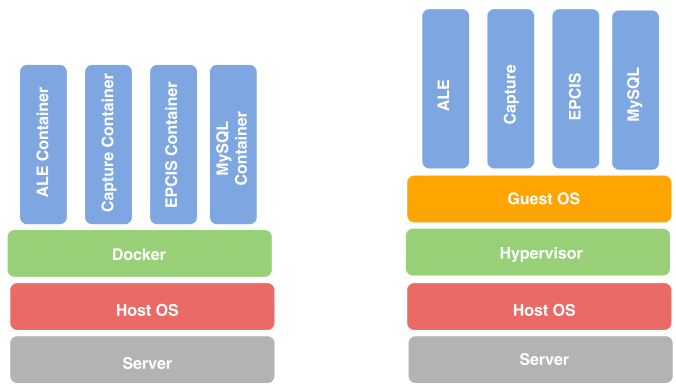

Fig. 4: Container vs. VM stack

The current implementation is running in a Amazon Linux based EC2 instance with a 8GB volume storage. After provisioning the stack our implementation is using ∼2.6GB of the available storage, which ∼1.4GB corresponding to the storage allocated by the Docker containers, as illustrated in Table 2.

Container	Size
MySQL Database	290 MB
EPCIS Repository	400 MB
Capture Application	381 MB
ALE Server	390 MB

Table 2: Docker containers size.

To compare our Docker-based approach with a full VM solution, we configured a Virtual Box VM running Ubuntu 14.04 LTS with the software stack required to have a full installation of Fosstrak. Compared with our implementation, a full VM approach requires almost twice the available storage - ∼4.7GB.

4.3 Cloud4Things vs. TOSCA-based tools

TOSCA is a powerful tool that allows to describe the internal topology and deployment process of cloud applications. However, to provisioning the smart place infrastructure with TOSCA, a special provisioning engine for TOSCA is needed. Currently there are several tools that implements a provisioning engine for TOSCA such as Ubuntu Juju[7], Cloudify[8] and the open-source implementation OpenTOSCA[9].

These tools allows the modeling of the application topology and deployment process in a more expressive way compared to using only configuration management tools. Although TOSCA standard is very promising, is not fully developed yet and does not support features that are used in our solution, such as container technologies.

5 Conclusion and Future Work

In this paper we propose Cloud4Things, a solution to automate the provisioning of the smart places infrastructure in the cloud. Our solution relies on configuration management tools to automate the provisioning the RFID software in the cloud providers. In the current approach, we decide to use Docker containers to provisioning the software stack due to the performance benefits that this technology offers. Our current prototype already is capable of automating some of the most important life-cycle stages of a smart place. Currently, our solution is centralized, i.e, all the containers that run the Fosstrak application are located in the same machine, a important component of our work will be to compare the current approach with a distributed solution. For the future work we want to collect some metrics to further compare the performance of our solution with the other solutions. For that, we will measure the scalability of the application regarding the quantity of events that occur in the physical world. Another aspect that is important is to verify if the latency of the events processing is acceptable.

References

1. M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.
2. J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
3. R. Cáceres and A. Friday, "Ubicomp systems at 20: progress, opportunities, and challenges," *IEEE Pervasive Computing*, vol. 11, no. 1, pp. 14–21, 2012.
4. F. Mattern and C. Floerkemeier, "From the internet of computers to the internet of things," in *From active data management to event-based systems and more*, pp. 242–259, Springer, 2010.

[7] http://www.ubuntu.com/cloud/tools/juju

[8] http://getcloudify.org/

[9] http://www.iaas.uni-stuttgart.de/OpenTOSCA/

5. L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.
6. D. Cook and S. Das, *Smart environments: technology, protocols and applications*, vol. 43. John Wiley & Sons, 2004.
7. D. Guinard, C. Floerkemeier, and S. Sarma, “Cloud computing, REST and mashups to simplify RFID application development and deployment,” in *Proceedings of the Second International Workshop on Web of Things*, p. 9, ACM, 2011.
8. F. Li, M. Vogler, M. Claeßens, and S. Dustdar, “Towards automated IoT application deployment by a cloud-based approach,” in *Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on*, pp. 61–68, IEEE, 2013.
9. D. Merkel, “Docker: Lightweight linux containers for consistent development and deployment,” *Linux J.*, vol. 2014, Mar. 2014.