BUILDING REFERENCE DEVELOPMENT ENVIRONMENT IN CLOUD USING FREENEST

Juha Sinkkonen

Bachelor's thesis
December 2012

Software Engineering
The School of Technology

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

KUVAILULEHTI

Tekijä(t) Sinkkonen, Juha	Julkaisun laji Opinnäytetyö	Päivämäärä 14.12.2012
	Sivumäärä 36	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty (X)
Työn nimi BUILDING REFERENCE DEVELOPMENT ENVIRONMENT IN CLOUD USING FREENEST		
Koulutusohjelma Ohjelmistotekniikka		
Työn ohjaaja(t) Salmikangas, Esa		
Toimeksiantaja(t) Rintamäki, Marko		

Tiivistelmä

Työssä tutkittiin kuinka pilviympäristö soveltuu ohjelmisto kehitykseen, käyttäen Jyväskylän ammattikorkeakoussa tomivan Skynest-projektin kehittämää FreeNEst ohjelmistokehitys alustaa. Työssä kartoitetaan pilviympäristön suurimmat ongelmat ja hyödyt ohjelmistokehitykselle erityisesti pienemmisssä projekteissa, sekä tutkitaan mahdollisia ratkaisumalleja kyseisten ongelmien ratkaisemiseksi. Suurimmat ongelmat pilviympäristön käytössä pienemmissä projeteissa ovat ympäristön pystyttämiseen tarvittava suuri investointi, hankala ylläpito, sekä mahdolliset tietoturva ongelmat. Tämän lisäksi työssä käytiin läpi ohjelmistokehitysympäristön rakentaminen pilviympäristöön FreeNest alustaa käyttäen, sekä pyrittiin arvioimaan FreeNest alustan tomivuutta tälläisessä ympäristössä. Käyttäen tätä ympäristöä mallina työssä pyrittiin arvioimaan löydettyjen ratkaisujen soveltuvuus käytössä.

Näitä ratkaisuja lähdettiin kartoittamaan käyttämällä ratkaisua jossa pilvi on rakennettu tavallisista tietokoneista yleisesti käytettyjen servereiden sijaan, tällä ratkaisulla pyrittiin pienentämään pilven rakentamiseen tarvittavaa investointia. Valittavasti tämä ratkaisu lisäsi pilviympäristön muita ongelmia, erityisesti tietoturva ongelmia mahdollisissa vikatilanteissa, ja toi uusia ongelmia. Koska tämä ratkausi oli muuten toimiva, työssä pyrittiin korjaamaan nämä ongelmat tämän tapaisessa ympäristössä.

Työn pääpaino oli teoreettisessa tutkimuksessa, jolla pyrittiin tuomaan esiin pilviympäristön mahdollisuudet erityisesti pienemmille ohjelmistokehittäjille. Työ ei ole tarkoitettu niinkään ohjeeksi kuinka luoda ohjelmistokehitysympäristö pilveen, vaan sen päätarkoitus on tarjota näkemys siitä mitä ohjelmistokehitysalustaa luotaessa tulisi ottaa huomioon ja kuinka välttää yleisempiä ongelmia sekä virheitä.

Avainsanat (asiasanat)
FreeNest, SkyNEST, Automatisoitutestaus, Git, Pilvipalvelut

Muut tiedot

Author(s)
Sinkkonen, Juha

Type of publication
Bachelor´s Thesis

Date
14.12.2012

Pages
36

Language
English

Confidential

( ) Until

Permission for web publication
(X)

Title
BUILDING REFERENCE DEVELOPMENT ENVIRONMENT IN CLOUD USING FREENEST

Degree Programme
Software Engineering

Tutor(s)
Salmikangas, Esa

Assigned by
Rintamäki, Marko

Abstract

This thesis studies how cloud environment can be used in software development when using FreeNest development platform developed in JAMK University of Applied Sciences under the SkyNest project. Thesis reviews cloud environment's biggest possibilities and problems for software development especially in smaller scale software projects, and study possible solutions to answer these problems. Main problems for using cloud based development environment for smaller software projects were, rather large initial investment required to setup the cloud environment, more difficult maintenance, and possible problems with data security. Furthermore thesis studies the process of building development environment in cloud using FreeNest development platform, and studies how FreeNest performs in this kind of environment. Using this environment thesis then evaluates the possible solutions for the problems of the cloud environment.

Thesis begins to find solutions for these problems by using a cloud where the cloud's structure is made by using regular desktop computers instead of commonly used server computers. This solution helps to keep down the initial investment required to setup cloud. However usage of kind of structure increases the effects of the other problems, especially the data security problems in case of hardware malfunction, and brings some new problems. Because this solution was otherwise very viable, thesis aims to correct these problems in this style of environment.

Main focus of the thesis is on the theoretical study, which focuses to study the cloud environments viability especially for the smaller software developers. This thesis is not meant be used detailed as guide how to setup the development environment in the cloud, but rather as study what to take in consideration, and how to avoid most common problems and mistakes when setting up the environment.

Keywords
FreeNest, SkyNEST, Automatic Testing, Git, Cloud services

Miscellaneous

Table of Contents

Illustration Index

TERMINOLOGY

Bugzilla - *“Bugzilla is a "Defect Tracking System" or "Bug-Tracking System". Defect Tracking Systems allow individual or groups of developers to keep track of outstanding bugs in their product effectively.”*
(Bugzilla, 03.11.2012)

Cloud - The whole cloud infrastructure, containing both the physical could structure and the cloud environment.

Cloud environment - The virtual network inside the cloud where the cloud's instances operate.

Cloud Software Program - *“The Cloud Software program (2010-2013) aims to significantly improve the competitive position of Finnish software intensive industry in global markets. Cloud Software program especially aims to pioneer in building new cloud business models, lean software enterprise model and open cloud software infrastructure.”*
(Cloud Software Program, 09.11.2012)

Cloud structure - The physical network of computers that form the structure of the cloud.

Framebuffer - *“A Framebuffer (or sometimes Framestore) is a video output device that drives a video display from a memory buffer containing a complete frame of data.”*
(Wikipedia, Framebuffer, 03.11.2012)

GIT - *“Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.“*
(Git, 03.11.2012)

IaaS (Infrastructure as a service) - *“IaaS is the most basic cloud service model that offers computers, as physical or more often as virtual machines, and other resources.”*
(Wikipedia, Cloud computing/Service models, 03.11.2012)

IRC (Internet Relay Chat) - *"Internet Relay Chat (IRC) is a protocol for real-time Internet text messaging (chat) or synchronous conferencing."*
(Wikipedia, Internet Relay Chat, 03.11.2012)

KVM (Kernel-based Virtual Machine) - *"KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V)."*
(KVM, 03.11.2012)

Linux - *"Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution."*
(Wikipedia, Linux, 03.11.2012)

OpenStack - *"OpenStack OpenStack is a global collaboration of developers and cloud computing technologists producing the ubiquitous open source cloud computing platform for public and private clouds."*
(OpenStack, 12.11.2012)

PaaS (Platform as a service) - *"In the PaaS model, cloud providers deliver a computing platform typically including operating system, programming language execution environment, database, and web server."*
(Wikipedia, Cloud computing/*Service models*, 03.11.2012)

RAID - *"Redundant array of independent disks, originally redundant array of inexpensive disks is a storage technology that combines multiple disk drive components into a logical unit."*
(Wikipedia, RAID, 03.11.2012)

Robot Framework - *"Robot Framework is a generic test automation framework for acceptance testing and acceptance test-driven development (ATDD)."*
(Robot Framework, 03.11.2012)

SaaS (Software as a service) - *"In this model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients."*
(Wikipedia, Cloud computing/*Service models*, 03.11.2012)

SkyNest - Project hosted by JAMK University of Applied Sciences, part of the TIVIT Cloud Software program.
(SkyNest, 12.11.2012)

StaaS (Storage as a service) - *"Storage as a service (STaaS) is a business model in which a large service provider rents space in their storage infrastructure on a subscription basis."*
(Wikipedia, Storage as a service, 03.11.2012)

Virtual machine - *"A virtual machine (VM) is a simulation of a machine (abstract or real) that is usually different from the target machine (where it is being simulated on)."*
(Wikipedia, Virtual Machine, 03.11.2012)

VNC (Virtual Network Computing) - *"Virtual Network Computing (VNC) is a graphical desktop sharing system that uses the RFB protocol (remote framebuffer) to remotely control another computer."*
(Wikipedia, Virtual Network Computing, 03.11.2012)

Wiki - *"A wiki is a website which allows its users to add, modify, or delete its content via a web browser usually using a simplified markup language or a rich-text editor."*
(Wikipedia, Wiki, 03.11.2012)

XVFB (X virtual framebuffer) - *"Xvfb or X virtual framebuffer is an X11 server that performs all graphical operations in memory, not showing any screen output."*
(Wikipedia, Xvfb, 03.11.2012)

XVNC - *"Xvnc is the Unix VNC server, which is based on a standard X server. Applications can display themselves on it as if it were a normal X display, but they will actually appear on any connected VNC viewers rather than on a physical screen."*
(XVNC, 03.11.2012)

1 INTRODUCTION

Commissioner

This thesis was commissioned for the FreeNest project that is a part of the JAMK University of Applied Sciences' SkyNest project, in order to evaluate FreeNest's viability for smaller scale software development in cloud environments.

About this thesis

The large software companies like the Microsoft have been using the cloud based development environments for awhile, for these companies the cloud environment offers great benefits. However for the smaller software companies usage of these kind of cloud environments have been so far rather impractical because of the huge initial investment that is required to setup the cloud environment, and because the tools available have been rather complicated to learn and use in smaller projects these companies usually .

The thesis studies viability of the cheap cloud environment build from normal desktop computers, and the progress of creating cloud based development environment that is simple to setup and easy to use by using FreeNest development tools. While thesis will study the viability of the cost effective cloud environment process of building and setting up that kind of environment is outside of the scope of this thesis, and will not be studied in detail.

The thesis is divided into two parts. In first part the environment is studied from theoretical view, with an analysis of its different strengths and weaknesses when compared to standard development environments, and different styles of environments available to us are discussed. A look is taken on to FreeNest as well as the kind of enhancements it offers for this style of software development.

In the second part of the thesis, the progress of setting up an example development environment and its setup and configuration procedures are discussed.

OBJECTIVES OF THIS THESIS

The objectives if this thesis are as follows:

- Learn different strengths and weaknesses of cloud based development environment when compared to standard environments.
- Study how FreeNest can be used as development tools in this style of development environment.
- Go through setup and configuration procedures of cloud based development environment.

2 TECHNOLOGIES

2.1 FreeNest

The FreeNest is an open source based project platform for software development. The FreeNest is currently developed at JAMK University of Applied Sciences by the FreeNest team as part of the SkyNest project, that operates under the TIVIT cloud software program. The goal of the FreeNest is to provide an easy to use working environment for software development teams, and to provide these teams all the necessary tools. (SkyNest, 12.11.2012)

Because the FreeNest is completely based on open source tools it provides a cost-effective solution for educational or commercial use.

Each installation of the FreeNest platform is designed to be used for a single project, and after the project has been completed the platform and all its contents can be destroyed. Because of this the FreeNest is designed so that new instances of the FreeNest can be launched quickly and efficiently as possible. This makes the FreeNest well suited platform for a cloud based development environment. (FreeNest, 12.11.2012)

FreeNest Modules

From version 1.4. onwards FreeNest has been developed using modular infrastructure based on separated Debian packages for each part of the FreeNest. By this split FreeNest is divided into two separated parts. FreeNest core is a package that contains and installs the basic FreeNest server and web interface as well as the required databases.

Other category is FreeNest plug-ins that either extend or enhance FreeNest's functions. These modules include systems like IRC servers, wiki or bug databases like Bugzilla. This allows easily scalable configurations as the users no longer has to install parts of the FreeNest they do not need in their projects.

2.2 Application Life Cycle Management

"Application Life cycle Management (ALM) is a continuous process of managing the life of an application through governance, development and maintenance. ALM is the marriage of business management to software engineering made possible by tools that facilitate and integrate requirements management, architecture, coding, testing, tracking, and release management." (Wikipedia, Application life cycle management, 03.11.2012)

According to ALM processes of software development is split in to several different steps, most commonly these steps are referred as; Requirement-, Design-, Implementation-, Verification- and Maintenance management.

Over the years of software development there has been multiple different methodologies for managing these steps. One of the oldest but still one of the most used one is so called waterfall model. The waterfall model is a sequential model where development flows through each step of the ALM in order (as illustrated in illustration 1).

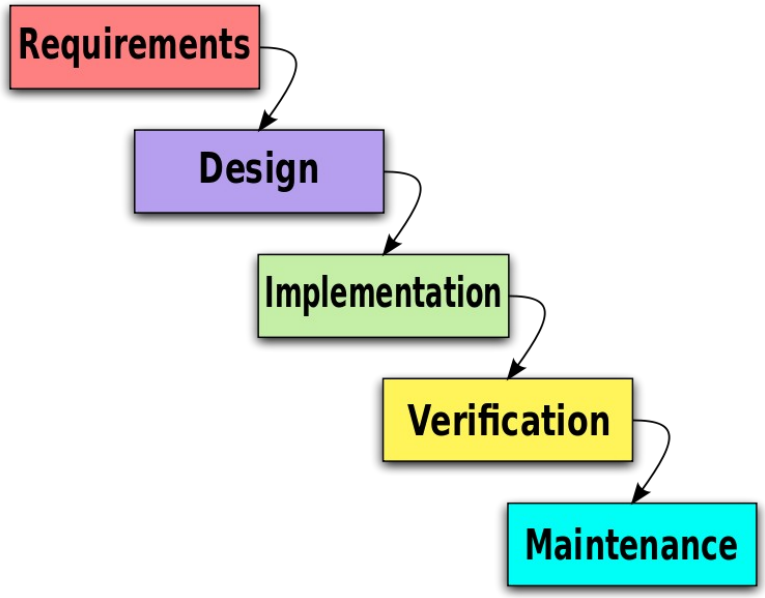

Illustration 1: Flow of the development in waterfall model (Wikipedia, Waterfall model)

While waterfall model can be effective for the ALM especially for smaller projects, because each development cycle goes through all the steps of the ALM before starting again, projects using the waterfall model can be slow to react to changes in the requirements or possible problems during the development process. (Wikipedia, Waterfall model)

To counter these slow reaction times it is better to use so called agile development methods. These methods allow the requirements of the project to evolve quickly. One of the most used agile development methods is scrum. In scrum development cycle is split into so called sprints that usually last between one week and one month. During each of these sprints developers work according to sprint backlog, that contains requirements that must be met during that sprint (illustrated in illustration 2). (Wikipedia, Scrum)

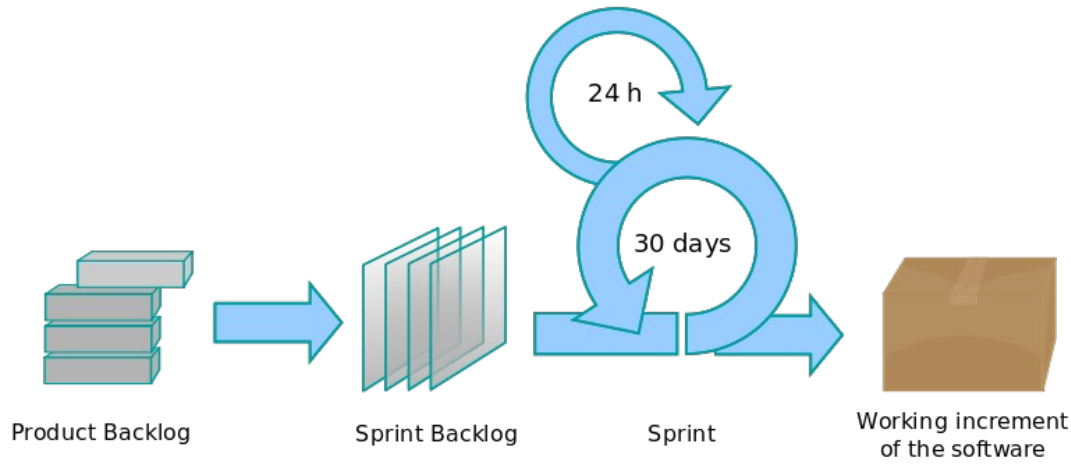

Illustration 2: Flow of the scrum process (Wikipedia, Scrum)

Because one sprint is relatively short, if there is change in the requirements, or problem in the development, developers can react to these changes quickly by including them into the sprint backlog for next sprint.

2.3 Cloud Environment

Cloud Structure or cloud environment is a special type network of computers, these computers are usually referred as cloud's nodes, each computer forming it's own node. While these computers are normal computers they do not operate as single units while they are a part of the cloud network, instead they work as a part of a large infrastructure that can be assigned to complete different tasks as whole depending on what types of services the cloud offers. (Wikipedia, Cloud Computing, 28.11.2012)

Structure of a cloud environment

The cloud is structured so that one of the cloud's nodes is chosen to be a cloud's master server. Even thou the cloud can be constructed from any computers available, it is highly recommended that this master server is powerful as possible. This is because the master server has to handle assigning cloud's resources (other nodes) while simultaneously handling all

the communications between nodes withing the cloud's internal network. This can put the master server under extreme stress, especially if there is a lot of simultaneous communications happening within the cloud. It is recommended that if it is possible that the cloud's master server should be an actual server, that has capability to handle these kind of network traffic and loads.

The cloud's other nodes form the cloud's resources, it is important to know that when the master server then assigns these resources it does not necessarily assign all the needed hardware from the same node. It is completely possible that certain instances have been assigned a CPU from another node and a RAM from another. The master server tries to match these hardware assignments closely as possible for the needs of an instance. For example, if certain instance informs the master server that it is going to need at least 2GHz of CPU power, the master server will check it's databases for available CPUs and then assigns that instance the CPU that is close as possible but still over 2GHZ that the instance requires. This is one of the most compelling strengths of using an cloud environment for software development. It allows development environment to quickly react and adjust to changes in requirements that software development has. For example if there is need for a new testing environment, as long as the cloud has enough free resources, setting up that environment is just a matter of telling master server to setup an new instance.

Some of the cloud environments even support adding new nodes to cloud on fly, meaning that if the cloud starts to run out of the resources, it is possible to just hook up a new computer into the cloud's network and it can then be used as node by master server without requiring to restart any of the cloud's computers.

The services of a cloud environment

In this thesis **IaaS** (Infrastructure as a service), **PaaS** (Platform as a service), **SaaS** (Software as a service) and **StaaS** (Storage as a service) types of cloud environments are mostly used. There are multiple other types of cloud environments but those are generally outside the scope of this thesis.

For this thesis writer assumes that there already is a working cloud environment that is ready to run instances, as processes of building and configuring cloud itself are outside the scope of this thesis.

3.1 Service layer models

Cloud model is usually split into three main service models that offer different kind of services for the client base of the cloud (illustrated in illustration 3). These three model are called; IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and finaly SaaS (Software as Service).

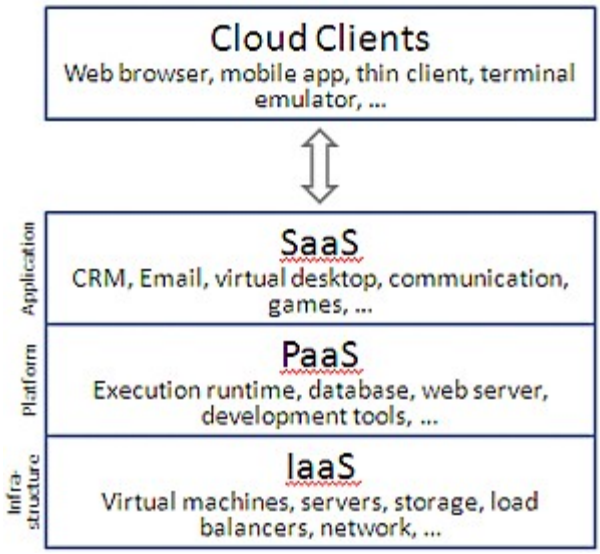

Illustration 3: Service models of Cloud (Cisco, 09.11.2012)

Most commonly offered service is SaaS, cloud can offer access to different kinds of software for clients, good example of this is websites or email. For software development SaaS can offer a virtual machine for developing software that an user can then use as normally. For an developer this is like connecting and using any other regular server or virtual machine, but in reality the server only exists virtually in cloud. (Wikipedia, Cloud computing/*Service models*, 03.11.2012)

The second model is PaaS, this usually includes services like databases and web servers. For developers this can mean database where they upload their code or it can even offer developers some development tools. (Wikipedia, Cloud computing/*Service models*, 03.11.2012)

The final model is IaaS, which can offer services like virtual machines or even whole network of machines. For developers these machines can then be used for example as testing platforms or to build projects. (Wikipedia, Cloud computing/*Service models*, 03.11.2012)

3.2 Instances in Cloud

An instance means a virtual machine that is run in the cloud environment. These virtual machines use some kind of disc or computer image as a model depending on the platform of the cloud. These images will also contain information about resources that the virtual machine is going to need.

These virtual machines are controlled by the cloud's master server. This master server handles creating or booting up these images when their services are requested by other machines either from inside the cloud or from an authorized outside user.

Depending on what platform the cloud is using images for these virtual machines can be either stored in the master server itself, in an outside storage server or the image can be uploaded when needed.

First two of these options allows the cloud to build a database of these images and any number of instances can be then launched based on those images. However this can require quite an extensive amounts of storage space depending on size and number of images that is going to be stored on master server. The last option does not require this kind of storage space, but because each image is stored in a server only as long as instance based on that image is running, each instance requires upload of it's image even if it uses same base image as one of the already running instances.

When the master server receives a request to boot up an image, it first checks the image for resources it needs, and then does another check to see if the cloud has enough available resources. If the cloud has enough resources available the master server then assign the virtual machine resources that are the best possible matches for the virtual machine's needs (as illustrated in illustration 4), and then links that hardware to that virtual machine using its own internal database.

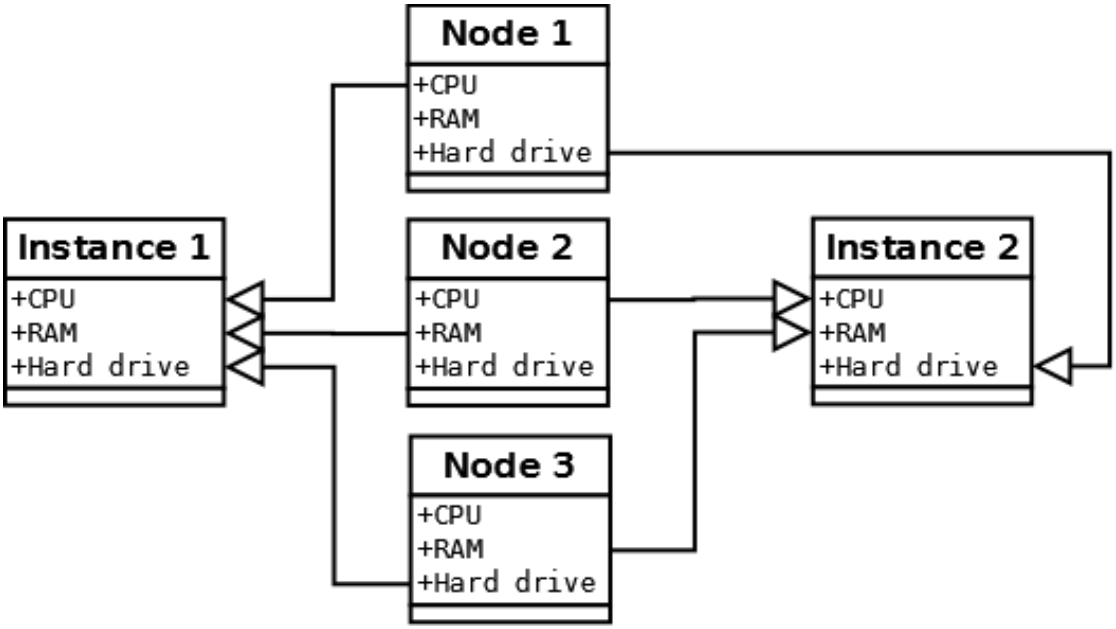

Illustration 4: Assigment of the hardware for the instance in the cloud

The master server then starts to boot up the virtual machine similarly to any other virtual machine. When the virtual machine is booted up, the master server then assign an IP for it, and notifies that the machine is now ready to be used and that it can be connected using that IP.

When the virtual machine is no longer needed and can be removed from the cloud, request for its removal is sent either by user or automatically because it was designed to be terminated after shutting down.

When the master server receives this request, it first check if the machine is still running and if it is it then sends terminate signal to it to shut it down. After the virtual machine is shut down, the master server then releases the hardware that was assigned for that virtual machine so it can be used by another instance, and finally either deletes or stores the image according to its settings.

It should be noted that no changes will be made to this image when it is assigned to an instance, all the data that is stored in virtual machine's assigned hard drive and all the changes that occur take place only inside that virtual machine and once the hardware that was assigned to an virtual machine is released all those changes and data will be lost.

Because of this it is vital that the cloud's master server is secured because if the master server loses its database that contains information about virtual machines and hardware assigned to them all the data in those virtual machines is lost. While the actual data still exists in hard drives of the cloud's nodes, it is impossible to tell how these hard drives were assigned and thus it is impossible to recover the data.

Also it is important to remember that any data stored in the virtual machine will be lost when master server deletes that instance. It is important that any virtual machines used to store data, may it be files or databases inside the cloud, must be secured so that the master sever will not accidentally delete those instances and their data. Because of this it is recommended that if instances must store and manipulate data, that data should be stored in either a separated instance that is configured specially as data storage or in a server that is located outside of the cloud.

It is also recommended that any vital information is backed up regularly and stored outside of the cloud, thus in case of a major hardware malfunction damages can be minimized.

3.3 FreeNest in Cloud

Installing and running FreeNest in a cloud environment does not have any major differences to running it in regular servers. All thought it is important to note that because some parts of FreeNest are used to store data like programs source codes, all instances that run FreeNest should be secured

This can be done in few different ways, first FreeNest can be run regularly in one computer or in this case instance, and that instance is to be just secured in the same way as any other instance that is used to store data is secured in the cloud environment. While this is the easiest and most resource efficient solution, it is also the most unreliable one, and any malfunctions can have catastrophic consequences.

Because all the data that FreeNest is going to handle is stored in same the instance, if something happens to that instance or the master server loses data about that instance, all the data that was stored in that FreeNest is lost. Needless to say, in worst case this can mean that the project must be started over from scratch. Even if the node running the FreeNest and the master server are secured really well it is still recommended to go with one of the other solutions, as they provide greatly increased data security while having a relatively small impact on resources that are required for the FreeNest.

The second way for installing FreeNest is to separate some or all databases in FreeNest into their own instances. While this does increase the number of required instances and thus the hardware that FreeNest requires from the cloud, it also provides greatly increased security. In case that one of the cloud's nodes suffer major hardware malfunction only the databases that happen to use that node's hardware will crash. All thought most cloud environments can recover automatically if one of their nodes crashes, if the hard drive that was assigned to the instance is no longer available, all the data will be lost at least temporarily. When the node that contains the instance's original hard drive is brought back online, if data in hard drive was not damaged, that hard drive can then be reassigned to its instance and it can then operate as normally.

It is clear that this method too has it weaknesses. Because the data stored in cloud can be some what volatile, it is recommended that all the critical data is regularly backed up and then stored outside of the cloud in a separated server. This can be done same way as any other regular server back up, as the cloud's instances can communicate to outside of the cloud's network as normally if allowed by master server.

The third and most secure way while still storing all the data in the cloud itself, is to use a special nodes for the instances that are going to be used to store and handle the data in the cloud. These nodes are automatically backed up by cloud in several different ways.

Firstly cloud assigns the hard drives for these nodes so that the hard drives from multiple nodes form a single RAID enabled hard drive, this way if hard drive from one of these nodes fails, no data will be lost because of the RAID protects it. Major weakness of this is that because the RAID adds redundancy, number of hard drives that are required also increases rapidly.

Secondly cloud can be set to automatically mirror the instances so that even if the instance and nodes assigned to it are completely destroyed, it can be instantly be replaced by its clone. This method offers the most security and because the instances can be instantly replaced by their mirrors, it is extremely unlikely that any major down time will occur. Of course this method also has its price. Because the nodes are mirrored perfectly, the hardware requirements of the nodes using this method are effectively doubled.

Choosing right the method for each instance is a careful project, and has to be balanced between the required security and the cloud's available resources. Of course not every instance in the cloud has to use the same security methods. When instance is send to the master server, it can be configured to use any of these methods, and the master server then chooses the required hardware for it, if cloud has required resources available.

3 CLOUD BASED DEVELOPMENT COMPARED TO NORMAL ENVIRONMENT

3.1 Development environment in normal network

Development environments in normal networks are easy to maintain and they work well for a single project or if multiple projects that have very similar environments, if the development environment evolves between or during the project it can cause major problems. Because in the regular development environment the development steps are handled by normal computers they must be reconfigured and in some cases completely reinstalled if there is any changes in the development environment. This can cause huge loss of time and can counter the time won during the initial setup of the network.

3.2 Development environments in cloud based network

Because of the flexibility and scalability of cloud environment there can be numerous different environment configurations available for software development, Instead of trying to go through all of these combinations a few of the most common configurations are looked into more in detail.

While all of these configurations are viable development environments, they all have their own strengths and weaknesses. In order to choose environment that is going to be best fit for project there are several points that must be considered. While there are many things to take into consideration, two most important ones are:

- The resources available in the cloud
- what automated processes like testing, does the project require.

Basic configuration with development tools and version control

This is the very basic configuration that uses one or two instances in the cloud that runs the FreeNest platform and it's databases (as illustrated in illustration 5).

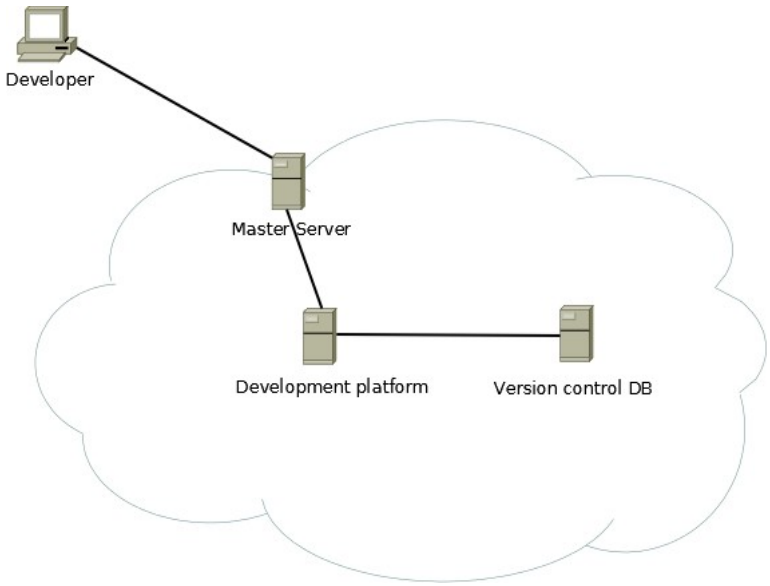

Illustration 5: Basic development configuration in cloud

While this environment does not provide any automatized testing or building it still provides full FreeNest platform and all the tools it provides.

Normally this setup uses only one FreeNest instances but in some cases it might be useful and desired to separate version control database to a different instance. This provides added security in case of a malfunction but also helps to provide an easy way to allow outside access to version control database with out compromising the whole FreeNest platform to threats from outside of the cloud. Developers use their workstations to connect to FreeNest and its version control databases and then do the work locally. Even if this kind of development structure uses only one instance, it is very useful for small and relatively simple projects that does not require robust testing or building environments.

Because of the nature of cloud environment and minimal need for configuration of these single instance environments, it is really easy and fast to launch them for those smaller projects. For example if a project is designed to last only for a short period of time instead of spending days to setup and configure environment and different databases, for a project this kind of a single instance can be launched in cloud in a matter of minutes and provide already configured tools and databases. This can greatly reduce development time and cost.

Basic configuration with added building robot

Compared to simple single instance solution this type of environment adds new instance into the cloud, that instance can then handle building projects either manually or automatically (as illustrated in illustration 6).

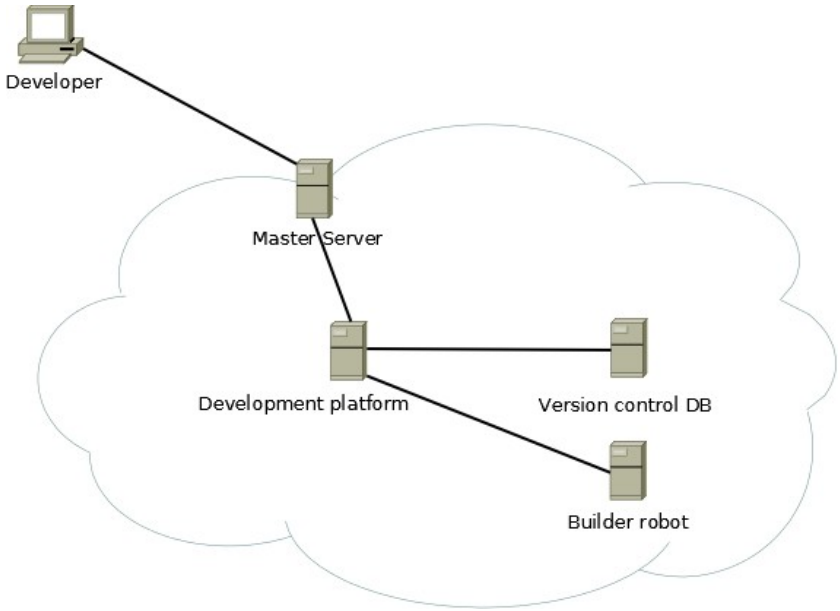

Illustration 6: Added build robot in cloud

The development progress in this type of environment is mostly the same as a simpler solution without the build robot. Users still use their local workstations to connect databases and services provided by FreeNest, however instead of using their local workstations to build their projects users use a builder robot in the cloud. Build robot can be configured to work either manually or automatically.

In the manual mode when developers upload a new version of the project into the version control database, developers then must manually order a robot to build that project. The robot then builds the projects according to the settings assigned to that particular project and when it is completed it uploads the project to either into the FreeNest project storage or into the temporary storage where the developer can then move it to a desired location.

In the automatic mode a builder robot receives a signal from the version control database when a new version of the project is uploaded. The robot will then automatically build this new version similarly to the manual version and upload it to a designated storage.

Both of these models have their weaknesses and strengths. While automatic build process will reduce work load of the developers, in some cases it might cause problems. For example, if there are multiple developers working on the same project and they are rapidly uploading new versions to version control database, these constant build orders may cause unnecessary stress to the builder robot.

This can be avoided by having separate version control databases for project in constant development and another database for bigger releases that contain multiple of smaller enhancements. When using this method developers can manually order build to their rapid development databases when it is necessary, and when they finish the current task they can upload a new version to the release database where the robot will automatically build it. This type of environment is especially useful for cross-platform projects where the program is simple enough that it does not require constant automated testing after each new build.

The cloud environment makes it possible to have multiple build robots configured for all different platforms that can then simultaneously build a project for all desired platforms as build is ordered. For example, if there is a situation where a new build robot must be introduced or an old one replaced, the modularity of cloud environment allows to perform these task easily and quickly as it is just a matter of bringing a new instance into the cloud environment. When compared to traditional solutions where whole computers must be reinstalled this will greatly reduce the workload and expenses.

Configuration with automated testing environment

This is the most common solution; it offers same benefits as version without testing environment; however added testing environment adds a possibility to do automated or manual tests easily. This solution is going add at least two instances to cloud environment when compared to last solution, it requires the test robot that runs the tests and at least one test target that the robot uses to install and test software (as illustrated in illustration 7).

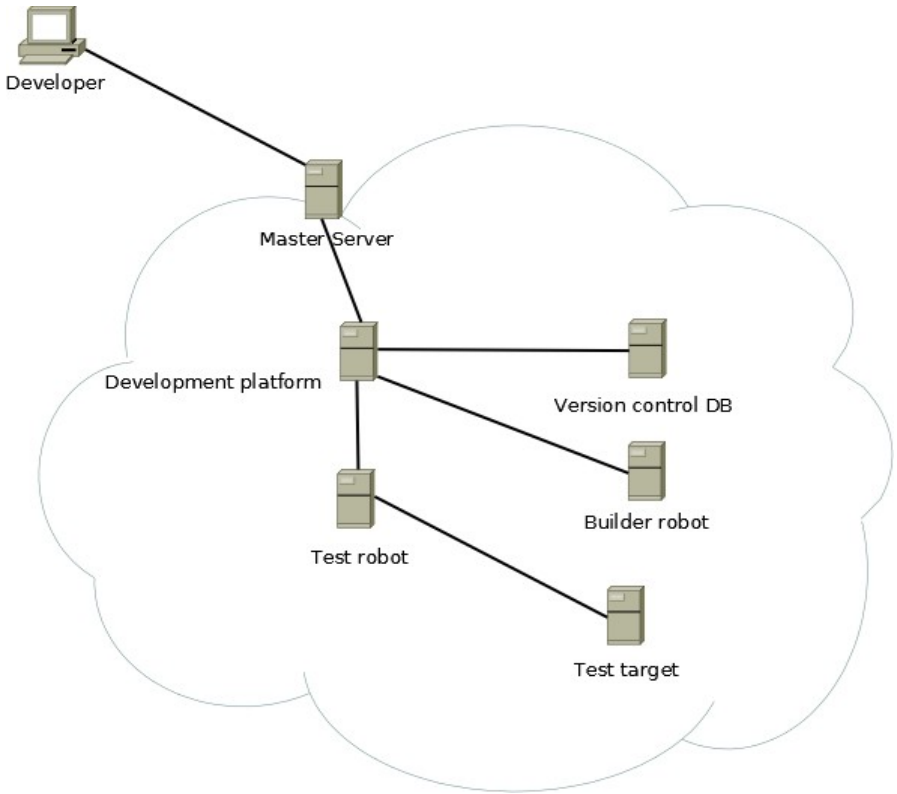

Illustration 7: Solution with added test robot and one test target

While it may seem that it would be always better to choose a solution with this type of testing environment, it is good to remember that a testing environment can require additional hardware and maintenance. So if this type of robust testing environment is not needed it may be more economical to choose configuration without it. Of course, if the cloud that is used to run the solution already has resources to run this kind of solution there really is no reason to not to. Since instances in cloud only exist, and thus only take up resources when they are in use, adding testing environment to your solution does not reserve anymore of cloud's resources than a solution with out it.

3.3 Strenghts and weaknesses of cloud environment

From an outside view process of developing software using cloud based environment does not different that much from using any other method. From the users' point of view they still connect to their databases and other resources as they would normally do. From an inside there are quite a few very important differences that should be addressed when planning and setting up the cloud based development environment.

Strengths of cloud based environment

The main strength of using cloud based environment for software development compared to regular model has to be its modularity.

The ability to switch out different target machines for testing, or build robots for different platforms almost instantly, and without requiring intensive configurations is almost priceless. Not only does this allow developers to react quickly and easily to possible changes in the development requirements, it also saves resources.

Another major advance that comes from the cloud's modularity is its ability to withstand hardware malfunctions. For example if there is a node that is currently hosting build slave for a project breaks, the cloud can automatically assign that instance to new node almost instantly. The broken node can then be removed from the cloud, fixed and introduced back into the cloud without interrupting development process.

Of course this does not mean that the cloud is in any way immune to hardware malfunctions; some malfunctions may even cause greater damage than in a regular environment; but more about this is discussed in the weaknesses section.

Weaknesses of cloud based environment

The main weakness and argument that is most commonly used against all cloud based systems, and especially cloud storage systems (and cloud based development environments since those use the same style of storage), is that cloud based system's data security is somewhat questionable.

While a cloud environment is not really any weaker or stronger against attacks than any normal network it has several weak points that if exploited could allow a possible attacker to gain ability to either steal or destroy sensitive data handled inside the cloud.

As mentioned in the previous chapter certain types of hardware malfunctions can also cause severe damage to a cloud system and the data it handles. The most prominent of these weak points when considering both attackers and malfunctions is the cloud's main server that handles assigning cloud's instances their resources and jobs.

In case of a hardware malfunction if this server is not properly secured, an simple hard drive failure can lead to the loss of all the data that cloud's instances were handling. This happens because instance's data it self is contained in a separated physical hard drive, and if the cloud's master server loses its knowledge where that data is located it no longer can be assigned for that instance. Therefore while the data still exists in a hard drive it is almost

impossible to recover it, and if the cloud is configured so that instances' virtual hard drives can share physical hard drive (for example one large physical hard drive can contain several smaller virtual hard drives for cloud's instances), the recovery becomes even more challenging.

Fortunately it is relatively easy to secure your cloud environment against these types of malfunctions. The first priority should always be securing the cloud's master server and especially its hard drives. This can be done in the same way as in any other server, by adding redundant hardware for system critical components and especially for hard drives. While a malfunction of a node in cloud is not as critical, losing hard drive can still cause loss of data. In case of nodes and virtual machines there are two ways to prevent loss of data in case of hard drive malfunction. Either instance's virtual hard drive can be formed as RAID from hard drives from multiple nodes thus preventing data loss if one of those hard drives fails. Other way is to install a RAID configured hard drives to the nodes themselves and then configure the cloud server to see those hard drive as normal.

Both ways have their advances, in the first case the nodes do not require any additional physical hardware, so it is cheaper to set it up, but because each instance requires an additional virtual hard drive for the RAID it can limit the number of possible simultaneous instances. The reason why the second option is preferred is that while the data is secured in both cases, in the first case the node's hard drive failure will cause that node to crash, and every instance that was using any resources of that node will crash as well.

The second major issue with cloud environment is privacy. Cloud environment is often criticized for how easy it is to company hosting the cloud service to access and modify the information stored in a cloud. Of course this is not such a large issue when using a private cloud for internal development, however it should still be considered if some level of outside access is offered to the cloud. For example if you allow the programmers to connect into the cloud using their personal computers, and then one those computers are compromised it can lead to an massive security breach.

Then there is, of course the problem of the hackers. Securing cloud environment against outside attacks does not really differ from any normal network; therefore these issues can be resolved by using the same security methods as for any other network.

Only major difference presents itself if the cloud's master server is compromised to an outsider attacker; they can cause much larger damage than what they could do by accessing a single computer in a normal network.

3.4 Possible solutions for the problems of cloud environment

If a cloud environment is built and secured correctly it can be a major boon to any software development company. There are, however many steps that can raise issues and questions that must be addressed before starting the process of migrating company's systems into the cloud.

Issues regarding possible hardware malfunctions are rather unique to a cloud based development environment but they can be solved rather easily by categorizing nodes in a cloud environment to different tiers according to level the of redundant hardware they contain.

Tier 1 nodes should be almost completely secured to a point where any hardware malfunction can be recovered from. For keeping expenses of setting up the cloud this tier should only consist of a master server of the cloud, and nodes running the most important databases like version control or Debian package databases.

Tier 2 nodes should have at least RAID hard drive back ups in case of malfunctions. These nodes should be prioritized to run instances that will be run for a long time and which are required to have minimal downtime like web services, or different servers like IRC. (Wikipedia, RAID, 03.11.2012)

Tier 3 nodes can be thought as work force of the cloud and as such do not require any redundant hardware. These nodes should run instances that will do few relatively quick jobs and can then be killed. These instances can be different kinds of build robots, target machines for testing, testing robots to name a few examples.

By using this kind of tiered nodes and taking into account the needed instances, the hardware requirements for the cloud can be easily measured, and expenses of setting up a cloud based development environment can be kept in check, while still providing all the strengths of the cloud environment, and minimizing the damage in case of hardware malfunction.

As long as the cloud is built and maintained correctly, and the servers are well secured, hackers and other security problems should not cause any more work than maintaining a regular network.

4 SETTING UP THE DEVELOPMENT ENVIRONMENT

4.1 Overview

In order to test how the cloud based development environments perform, cloud based development environment for developing Android software was build, on a cloud environment that was constructed using only regular desktop computers.

The FreeNest is used as the base for the development, and required development tools for developing Android software are installed. Addition to that automated testing and release robots are installed and configured, so that development process is going to be efficient as possible.

As for security FreeNest's most important databases are separated into their own instances, to further enhance the data security some of these databases will also be fully mirrored. Also these databases will be automatically backed up to a file server outside of the cloud in regular intervals.

Because of all of this the environment is going to require relatively large amount of resources from the cloud. All in all following instances are required:

- FreeNest Base
- FreeNest's Databases, including wiki, user information and Bugzilla
- GIT version Control database
- database for software packages in development
- database for released software packages
- build robot
- testing robot
- testing environment.

That is eight instances as base, but because number of these instances are going require different levels of backups the final number of required nodes is going to be higher.

All four of the database nodes are going to be mirrored so each one of those instances are going to require complete node as an backup, additionally the databases for the software packages are going to be backed up to an server outside of the cloud, however this does not add requirements for the cloud itself so it can be ignored for now. FreeNest base node is also going to be backed up, but only by assigning its hard drive from node that has RAID hard drive.

All in all this particular environment requires twelve nodes and at least one of those nodes must have RAID enabled hard drive. However this does not include the master server of the cloud. Illustration 8 shows the final setup of the nodes.

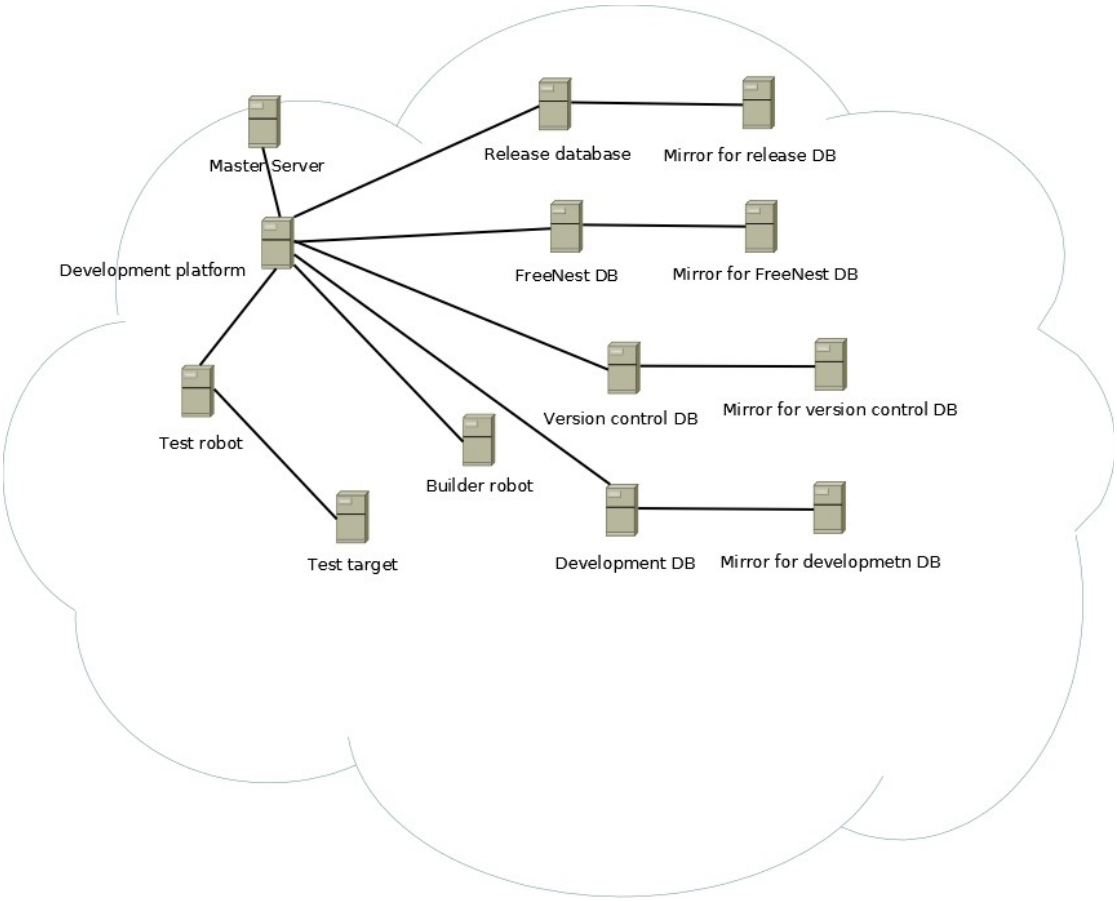

Illustration 8: Environment setup for the example

This setup is an example of very large and robust development environment that is meant to show the full potential of the cloud based environment, most software projects are not going to require this level of automatisation or backups, and thus can be done in much smaller scale clouds.

4.2 Setup procedure

As setting up cloud environment itself is outside the scope of this thesis, this example assumes that there already is a cloud environment that is ready to run different instances. In this example open stack based cloud environment is used as platform for the cloud environment.

Preparations

The first step is to determine which kind of developing environment is used. As discussed in last chapter, this example is going to use very large and robust environment that is configured to automatically handle building, testing and release procedures. This is done so that this example can offer an look to the full potential of the cloud based environment and features it offer to the software development.

Creating the virtual machine images

After deciding the desired configuration it is time to start to build images for different instances that are going to run in cloud.

The development environment is going to consist eight instances the are going to require a total of twelve nodes. Building images for these instances can differentiate depending on what software the target cloud environment uses and what platform is used while creating those images, in case of this example the cloud is running using Open stack and images are based on Linux so standard KVM images are used. (OpenStack, 12.11.2012)

The creation of these images is not any different from creating normal virtual machine images, in this case since all images are headless machines each image is going to use normal Ubuntu server as it's base.

There are few ways to prepare these images, depending on how much work can be done before the image is actually introduced into the cloud. In most cases all services and programs can be fully installed while making the image, but it should be noted that because the hardware and IP address of the machine may be different when it is run in cloud, it may be required to install some software after the image has been introduced in the actual cloud environment. Also because in some cases the order of which software is installed can matter, can the software that can not be installed when image created block installation of other software.

If most or all of the image can be installed before introducing it to the cloud environment it is beneficial to do so because trouble shooting can be easier while creating the image. However if image contains a lot of software that either can not be installed while the image is being created or is dependent on order in which software is installed it might be easier to create an image that contains installation files and script that installs that software. This script can then be configured to automatically be executed on first launch, and all the software should install automatically when the cloud boot ups the instance based on that image for first time. Of course there are some software that are going to require either manual installation or at least some configuration after the image has been introduced into the cloud.

The only difference from this is the build robot and target machine for testing. Because this example is based on android development environment, the builder robot is going to require a screen to correctly run the Android emulator that is required for building the program. The target machine for testing is also going to require a monitor in order to run an emulator for testing purposes.

Problem with this is that instances with physical monitors cannot be run in the cloud, this is resolved by installing virtual monitors to the headless images. In this case XVFB is used to create virtual frame buffers that act like regular monitors, and then XVNC is used to “stream” this frame buffer over VNC protocol. While streaming this frame buffer is not required, it helps in case that some trouble shooting is required. The stream can be connected from any computer that can access the cloud's network, and by using that stream it can be seen what the emulator is doing. This also helps in testing as it is possible to visually see what the testing robot is doing. (Wikipedia, Xvfb, 03.11.2012)

Introducing images to the cloud environment

After all the necessary images have been created the next step is to start implementing those images to the cloud environment. Depending on what platform the cloud uses this step can be done in multiple different ways.

In this case images are simply send to the cloud's master server that then stores these base images for later use. After the master server has received an image, instance based on that image can then be launched, and if the image works the virtual machine can then be connected by creating a ssh connection to IP that master server assigned to that instance.

It is important to remember that if that instance requires any special hardware or services from cloud, depending on cloud's platform it must be specified when ordering the cloud's master server to launch that instance for first time, or when image is uploaded to the master server.

For this example, four of the images are going to require full cloning and one of the images requires an hard drive with an RAID. These specifications are send to to cloud's master server when instance is launched for the first time and before it has been assigned any hardware.

Post launch installations

After all the images has been successfully launched, the next step is to perform any software installations that cloud not be done while images were created or could not be automated by creating scripts.

These softwares usually include systems like databases that require password configuration during installation, or software that requires to keep track of the machines hardware, IP address or network connections (for example firewalls).

It should be noted that if software that is being installed does not have headless installation available it might be required to temporarily to install some kind of virtual frame buffer so that installation may be completed.

While in this example all the software can be installed during the creation of the images, these are important things to remember if some post launch installations are required.

4.3 Configuration

Configuration process of cloud environment's instances doesn't really differ that much from configuration of regular network of computers, as the user connects to them using ssh connection like they would to any regular computer.

There are couple things worth to be mentioned, The configuration of IP addresses and cases where monitor is required for configuration of a specific software. For IP addresses it is important to make sure that right IP is used for communications inside the cloud. Instances internal IP should be used for communications inside the cloud, but if the instance must communicate to outside the cloud's internal network its public IP should be used. For software that are going to require a monitor for the configuration, XVNC or any similar program can be used, as it allows instance to transmit its frame buffer over VNC connection.

In this example both the Android test slaves and the Android builder robot are both going to require a monitor for their configuration, however since both of these instances already require monitors for the android emulator to work properly, XVNC installed for that can be also be used for the configuration needs.

4.4 Using distributed development environment

As mentioned in earlier chapters, from the user point of view a cloud based environment looks like any other network, thus for the developers usage of the cloud environment is exactly the same as regular a development environment with same automatisations.

However for a maintenance team, a cloud environment can be a bit more complicated to maintain than a regular environment. Most of this comes from the cases where the cloud's nodes have hardware malfunctions. As discussed in chapter 7.2, these malfunctions can cause rather massive damage to the projects using the cloud. Because of this the maintenance team in charge of the cloud should always perform regular checks for the nodes, and replace any nodes that show any signs of breaking down immediately.

5 RESULTS

Overall the thesis met all of it's obejectives rather well. During the processes of studing the cloud enviroiments, we were able to find what are the major strenghts and weaknesses of using the cloud based development environment, and think possible solutions for these problems. Then by using the example development environment, we were able to show that by using these solutions the cloud based environment cloud be viable solution for a smaller scale software development. We were also able to study the usage of FreeNest as development platform, and evaluate it's performance as development platform in cloud environment.

Viability of cloud environment

Results of this example were mixed, while it is entirely possible for even a small software company to setup an cloud environment relatively cheaply using normal desktop computers. The tools for maintaining the cloud build this way are still relatively new and under developed. Thus while cost of setting up an cloud environment can be cut down by usage of normal computers instead of servers, costs and work required to maintain the cloud environment can be still quite high. Other problems that rise from using normal computers are security and processing power. These problems are especially clear in cloud's master server as even very powerful desktop computer just can not handle the network traffic required in order for the cloud to operate smoothly. However these problems can be circumvented by using mixture of server and desktop computers. Even in case where only the cloud's master server is replaced with an actual server it can noticeably boost cloud's performance, to the levels where it's usage comes viable again. Of course this still leaves the problem with security, if cloud environment is build using only desktop computers that don't have any redundant hardware, even small hardware malfunction can cause the whole cloud environment to collapse. Solution of using tiered nodes to offer an redundancy for those instances that require it, while still keeping the cloud as cost-effective as possible was success. By using this kind of scaling security, it is possible to offer almost same level of security as server based environment, while still keeping cost of setting up the environment low.

By utilizing these solution only major weaknesses of the cloud environment is the somewhat complicated maintenance and setup procedures. Of course it should be noted that these problems are present when any major changes are made to the development environment, and after new environment is learned the maintenance becomes easier and faster.

Overall while there are still some problems for adopting cloud based developing environment, it offers great benefits for any software company, and tools and procedures to setup and maintain the cloud environment advance rapidly. Even if it might not be fully viable at the moment for every software developer to switch to cloud based environment. It is clear from benefits it offers, that in near future cloud based development will become more and more common as the process of adopting it becomes more and more accessible for even smaller software companies.

Usage of FreeNest in cloud environment

FreeNest performed really well in this kind of environment, and there was not any major problems. Because the FreeNest is an open source based platform, every tool it offers is under constant and rapid development, because of this any major bugs that could hinder the FreeNest can be fixed quickly. Another great strength that FreeNest has, because of its open source roots, is that any tool can be replaced rather easily if there is reason to do so.

The second important feature of the FreeNest is its modularity, this feature allows the FreeNest to be easily scaled to fit the needs of any development project, without bringing any unnecessary tools or features that might slow down the development. This also allows different modules to be developed separated from each other, allowing quick and efficient updates to those modules that needs them.

It is clear that currently the FreeNest offers great and very versatile development platform that can be used efficiently in both cloud and normal development environments. Because of this versatility, the FreeNest can be easily used in all levels of projects. On top of that the FreeNest has very active and skillful development team behind it, providing quick updates and bug fixes when needed.

6 CONCLUSION

The big software companies like Microsoft have been using cloud based development environments for a long time now. It is clear that cloud environment offers great benefits for software developers, however because of its weaknesses and the major initial investment required to setup the cloud environment it has not been a viable solution for a smaller software companies so far.

However because of the platforms like FreeNest and major development of cloud software like Openstack, setting up the cloud has become easier and easier. Coupled with the ability to use regular computer instead of expansive servers as cloud's nodes, cloud based development is slowly becoming viable solution for all software companies.

While using normal desktop computers instead of actual servers as cloud's nodes causes its own problems with reliability, as shown impact of this can be minimized by adding redundancy for the cloud's hardware. While this increases the cost of setting up the cloud environment, because of the relatively cheap price of the computers used to build the cloud, the solution still stays as cost-effective way to setup the cloud environment.

While it may take some time until cloud environment can be considered as main development environment for smaller software companies, it is clear that benefits it offers are large enough to justify the investigation for using the cloud environment.

7 REFERENCES

Bugzilla, referenced 03.11.2012
http://www.bugzilla.org/about/

Cisco, What Cloud Computing Exact is? - Cisco & Cisco Network Hardware News and Technology, Referenced 09.11.2012
http://ciscorouterswitch.over-blog.com/article-what-cloud-computing-exact-is-102461240.html

Cloud Software Program, referenced 09.11.2012
http://www.cloudsoftwareprogram.org/cloud-software-program

FreeNest, referenced 12.11.2012
http://freenest.org/about

Git, referenced 03.11.2012
http://git-scm.com/

KVM, referenced 03.11.2012
http://www.linux-kvm.org/page/Main_Page

OpenStack, referenced 12.11.2012
http://www.openstack.org/

Robot Framework, referenced 03.11.2012
http://code.google.com/p/robotframework/

SkyNest, referenced 12.11.2012
http://www.jamk.fi/projektit/1233

XVNC, referenced 03.11.2012
http://www.hep.phy.cam.ac.uk/vnc_docs/xvnc.html